JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Ing. Marco Praher, BSc

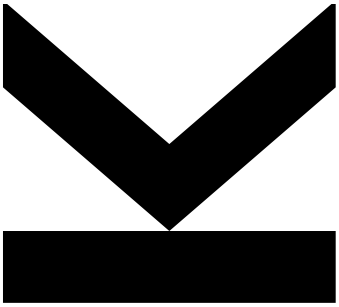
Submitted at
Institute of Networks and Security

Supervisor
Univ.-Prof. Priv.-Doz. DI Dr. Rene Mayrhofer

Co-Supervisor
Michael Hölzl, MSc

January 2017

Utilizing Bluetooth keyboard emulation and optical character recognition to securely transfer passwords



Master Thesis
to obtain the academic degree of
Diplom-Ingenieur
in the Master's Program
Computer Science

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Abstract

The rising popularity of online services, propelled in part due to the success of smartphones and tablets, has resulted in a situation where a user typically has data spread across multiple services. This data is often personal and sensitive like photos, videos, documents, e-mails, contacts, calendars and so on. The access to this data is usually protected by a combination of username and password chosen by the user. Using a different and strong password for every account is the user's responsibility to prevent adversaries from gaining access. Unfortunately users tend to choose weak passwords, especially if they need to remember and type them regularly.

This work proposes a solution that employs an Android device to manage all credentials of an user. This includes safely storing them on the device, efficiently finding credentials and transferring them to another devices. This transfer happens only on demand for individual credentials in a secure manner. Additional support for adding new credentials exists, in case the user creates a new account.

To create the solution different technologies are examined. Their applicability and feasibility to use in an Android application is determined. To transfer credentials between devices the Bluetooth human interface device profile is utilized. Safekeeping of user credentials is implemented by integrating a password manager. Further the application of optical character recognition is considered, to simplify searching entries in the password manager.

Abstrakt

Die steigende Beliebtheit von Online-Services, vorangetrieben unter anderem durch die Beliebtheit von Smartphones und Tablets, hat dazu geführt, dass ein Benutzer typischerweise Daten bei mehreren Anbietern verteil hat. Die dort gespeicherten Daten sind oftmals sensible und von persönlicher Natur wie z.B. Fotos, Videos, Dokumente, E-Mails, Kontakte, Kalender und ähnliche. Der Zugriff auf diese Daten ist üblicherweise durch eine vom Benutzer gewählte Kombination aus Benutzernamen und Passwort geschützt. Der Benutzer ist dafür verantwortlich für jeden Account ein unterschiedliches und starkes Passwort zu wählen. Dadurch wird der Zugriff durch Unbefugte verhindert. Jedoch tendieren Benutzer dazu einfache Passwörter zu wählen und diese auch bei mehren Accounts zu verwenden, besonders falls diese im Gedächtnis behalten werden und oftmals eingegeben werden müssen.

Diese Arbeit präsentiert eine Lösung die ein Android-Gerät dazu verwendet um alle Zugangsdaten eines Benutzers zu verwalten. Dies umfasst das sichere Speichern am Gerät, effizientes auffinden benötigter Daten und die Übertragung zu einem anderen Gerät. Die Übertragung findet dabei nur bei Bedarf für individuelle Zugangsdaten in einem sicheren Verfahren statt. Zusätzlich besteht eine Möglichkeit neue Zugangsdaten hinzuzufügen, falls der Benutzer einen neuen Account erstellt.

Zur Realisierung dieser Lösung werden mehre Technologien betrachtet und ihre Anwendbarkeit innerhalb einer Android-Anwendung. Für die Übertragung von Zugangsdaten zwischen Geräten wird das Bluetooth Human Interface Device Profil eingesetzt. Für das sichere Speichern der Zugangsdaten wird ein Passwort-Manager integriert. Außerdem wird optische Zeichenerkennung (optical character recognition) verwendet, um das Auffinden von Einträgen im Passwort-Manager für den Benutzer zu vereinfachen.

Contents

Figures

Listings

Tables

1 Introduction

1.1 Motivation

The popularity of online services is steadily rising. The Internet is used for a wide variety of different tasks like shopping, communicating, banking, photo/video sharing and safekeeping of personal data. All this tasks involve private information the user wants to protect from illicit access. But exactly this continually growing amount of data is very attractive to attackers. Therefore the number of data breaches has risen steadily over the last ten years. The number of data breaches concerning the United States clearly show this trend. They went up from 157 incidents in 2005, with 66.9 million exposed records, to 781 incidents in 2015, with 169 million exposed records.[59] Famous examples of data breaches, widely covered in the press, that happened in the last four years include big companies like Adobe[36], Anthem Inc[3], Ashley Madison[66], Citigroup[37], Ebay[51], Facebook[14], Google Gmail[52], Nintendo[38], Sony PlayStation Network[50] and Twitter[67].

As a result, some of those data breaches leaked personal data to attackers or the public. In some instances this data also included passwords, either in clear text or in a hashed form. If a user, who's data got leaked, uses the same password for a different service, that account is also potentially threatened. Therefore it is highly recommended to use a different password for every account. Yet a study found that 73% of all interviewed people reuse passwords for multiple or all accounts.[44]

Similarly problematic is the use of unsafe passwords. That means a password that is either too short, too simple, or easy guessable. Despite this a 2015 survey found that "123456", "password", and similar easy guessable passwords are still very popular and often used.[61]

So why is the use of unsafe passwords still common, despite the wide public coverage of those data leaks? The reason for that is not hard to guess. It is difficult to remember multiple passwords, especially if they are long and complex. Furthermore typing secure passwords takes longer and is more inconvenient. Password managers aim to mitigate those problems by generating strong passwords, (safely) storing the login credential, and automatically inputing the information whenever necessary. This takes away the complexity and inconvenience of

password management from the user.

While the use of a password manager helps to mitigate those problems, it creates another one. The credentials are only available on the device that runs the password manager. If the credentials are needed on other devices, it is necessary to synchronize them. To achieve this there are typically two options available. One possibility is to manually transfer the database of the password manager (e.g. via a USB storage device) to the target device. This method is inconvenient and it is usually not possible to merge changes made on two different devices. The second option is via an online service. By using a web service, usually provided by the developer of the password manager, the data can be stored and managed centrally and synced to all devices. However it is necessary to fully trust the service to appropriately encrypt and protect the data.

Therefore a different solution for password management across multiple devices is desired. A mobile device, probably a smartphone, should act as the single and central storage for login credentials. Subsequently a way to transfer the data from the mobile device to the target device is necessary. This has to be done in a secure manner and without requiring any modification to the target device, such as a software installation or the like. Additionally the process of searching the correct credentials currently needed by the user should be optimized.

1.2 Proposed Solution

The goal is to create an Android application that solves the problems described above and satisfies the requirements outlined in the previous section. For the secure transfer of the credential information, from the Android device to the target device, a Bluetooth human interface device (HID) connection should be utilized. Therefore a Bluetooth keyboard can be simulated, which requires no additional software installation on the target device. The target device only needs a working Bluetooth interface and corresponding driver. Using the Bluetooth HID profile provides security measures like encryption and authentication, which help to protect the information transfered wirelessly.

The application also supports the user in searching for the correct credentials for a specific site. For this the user can point the device's camera at a computer screen and snap an image of it. A user defined part of this image, typically the URL bar of a web browser, is then further processed to enhance the characteristics important to achieve good accuracy in automated text recognition. The result is converted to text using a state of the art optical character recognition (OCR) library. The user is then able to edit the result or directly use it to query the credential database. The complete information stored in the credential database

is searched for a possible match. If multiple possibilities are found, the user has to select the correct one manually from a list. Therefore the user does not have to manually type, possible long and complex, URLs or search terms into the application.

The primary goal of the application is to transmit credentials to another device. Credentials are sensitive data that must be stored in a secure manner that prevents access by unauthorized entities, even in case the device gets stolen. For exactly that problem a myriad of different applications already exist. They are usually summarized under the term password managers. Their purpose is to help the user manage credentials for different accounts and multiple services. This includes generating safe passwords for new accounts, searching the database, backup/restore functionality, and most importantly the safe and protected storage of the data. To achieve this, usually a combination of encryption and some form of master password is employed. Because of the wide availability of password managers, the task of storing sensitive information is delegated to one of them and is not handled within the application itself. To achieve good usability, it is necessary that the password manager can be integrated seamlessly into the application.

The proposed solution helps to manage all of a user's different credentials on a single device and transfer them to a different device when needed. This removes the necessity for the user to remember and type passwords, the most significant inconvenience factor when using password authentication. Therefore different, long and complex passwords can be used for each individual account. That significantly increases security. The application is designed in a way that no specialized software installation is required on the target device. Therefore it supports a wide variety of operating systems and web browsers. Additional using a wireless connection for password input, instead of actually typing the password on a physical keyboard, removes the possibility that other people observe the password via shoulder surfing.

The main security objectives are the user's credentials stored and processed by the application. On the Android device they are protected by the security mechanisms of the password manager. Protection against active and passive attacks, while transferring individual credentials, is achieved by utilizing security mechanisms defined in the Bluetooth specification. The biggest security uncertainty is the target device receiving the credentials via Bluetooth. Malicious software on the target device, e.g. a keylogger, cannot be detected and protected against by the application. Therefore it is necessary for the user to transfer information only to trustworthy target devices.

1.3 Alternative Solutions

Multiple different, less desirable, solutions are possible to solve the outlined problem. Those are considered fallback options, if the proposed solution is not feasible to implement. They concentrate on alternative ways to transfer the credential information from the Android device to the target device.

System service This solution employs a service installed operating system wide on the target device. Instead of using a Bluetooth HID connection a common Bluetooth data connection is used to transfer credentials from the Android application to the system service. The system service then emulates key-press events locally. Therefore this solutions works with every application on the target system. However a custom agent for every supported operating system is necessary. The major reason for not pursuing this approach as the main solution is the necessity to install software on the target system.

Browser plugin This solution works by creating a plugin/add-on for a specific web browser. A Bluetooth data connection is used to transfer the credential information to a web browser. Then the plugin is able to input the data into the correct form fields without the necessity to emulate key-press events. An additional advantage of this solution is the ability of the plugin to send the currently open URL to the Android application, where it can be used to search for the corresponding entry in the password database. The downside is that a special plugin for every web browser that should be supported is necessary. Additionally it may not be possible to access Bluetooth directly from a browser plugin, due to permission and sandbox restrictions. The major reason for not pursuing this approach as the main solution is the necessity to install software on the target system.

Flyfish Technologies hiDBLUE[16] The hiDBLUE dongle is a hardware device inserted into the USB port of the target device. It simulates a USB HID keyboard and can be addressed by connecting via a Bluetooth data connection to it. This solution does not require an additional software installation on the target device and is therefore very similar to the proposed solution. Major downside of this solution is the necessity of an additional hardware device and an USB port on the target device.

1.4 Related Work

There are already existing software solutions that emulate a human interface device for another target device. However there is no solution existing that implements a Bluetooth human interface device on a modern Android system. The following application built for Android already exist:

AndroHid[40] Simulates a Bluetooth HID keyboard on a rooted Android device. Does not work on Android version 4.2 and later.

BlueCtrl[55] Simulates a Bluetooth HID keyboard and/or mouse on a rooted Android device. Like AndroHid it does not work on Android version 4.2 and later.

Android Keyboard Gadget[53] Simulates a USB HID keyboard and/or mouse. An active USB connection to the target device is necessary.

The proposed solution employs optical character recognition (OCR) to detect plain text on an image. Albeit it is not in scope to develop an OCR algorithm, instead an existing solution should be used.

ABBYY Mobile OCR[1] A proprietary application for Android that provides an SDK that allows other applications to embed OCR functionality.

tesseract-ocr[62] An open source OCR engine originally developed for desktop operating systems. However a library for Android exists.

2 Bluetooth

2.1 Introduction to Bluetooth

Bluetooth is a wireless technology for short range communication. It consists of different specifications developed and maintained by the Bluetooth Special Interest Group (SIG). Its main application is in mobile devices, therefore it is designed with low power consumption in mind. For two devices to communicate no visual contact is necessary. The first version (1.0B) of Bluetooth was releases in 1999 and the current version (4.2) in 2014.[56]

Because of its generic design, Bluetooth has a wide variety of uses in different fields. A few examples where it is utilised:[7]

- Cars: Allows to use the car as a headset/speaker, stream music and relay navigation information from another Bluetooth device.
- Consumer Electronics: Uses cases range from wireless headphones to televisions and game controllers.
- Personal Computers: Bluetooth allows transferring data to other devices, stream audio/video or connect other arbitrary devices.
- Health and Fitness: Sensors (e.g. heat rate monitor) can transfer their measurements wirelessly to another device.
- Smartphone: Allows to connect a smartphone to a wide range of different devices like computers, audio/video equipment, cars and sensors.

Communication in Bluetooth is packet-based and segmented into piconets. A piconet consists of one master device and up to seven slave devices. The master is always the device that first initiates the connection. Every piconet communicates on one channel (1 MHz bandwidth) inside the 2.4 GHz Industrial, Scientific and Medical (ISM) frequency band. Multi-directional communication between the devices in one piconet is possible by using time

slots. To allow multiple piconets in parallel at one physical location the master determines viable channels (channel assessment) so that no local clash occurs.[56]

Each Bluetooth device shares some information with all other devices in range. The Class of Device (CoD) defines the basic type of the device in form of major device class/minor device class (e.g. Phone/Smartphone or Health/Blood Pressure Monitor). Additionally a device advertises the services (called profiles) it supports. With this information other devices can decide if they can/want to establish a connection.[7]

Because Bluetooth is mainly designed for mobile devices it defines different power classes, depending on how important it is to save power. Power class 3 sends with 1 mW (1 meter), power class 2 sends with 2.5 mW (10 meter), and power class 1 sends with 100 mW (100 meter).[56]

The Bluetooth architecture consists of multiple layers and protocols. Figure 2.1 show an overview and the listing gives a short description.[56]

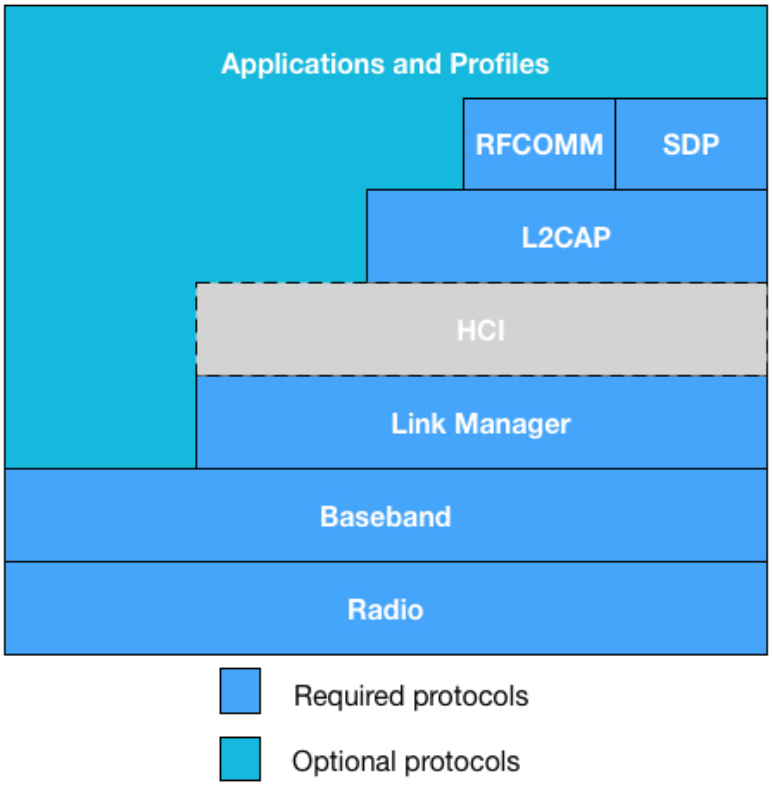


Figure 2.1: Bluetooth protocol stack[7]

- Radio: The radio layer is responsible for the actual transmission (frequency, power, modulation) via the antennas.

- Baseband: Puts the data inside frames. Adds information to identify the correct piconet and for error correction.
- Link Manager: The link manager is responsible for negotiating and controlling the connection between Bluetooth devices.
- HCI (Host Controller Interface): HCI is a communication interface between the actual Bluetooth chip and the end device. This means there is usually a physical separation at this point.
- L2CAP (Logical Link Control and Adaptation Protocol): Allows multiple logical connections to a device, which are all multiplexed before transmission. As a differentiating parameter for the different logical connections the Protocol Service Multiplexer (PSM) is used.
- SDP (Service Discovery Protocol): A Bluetooth device can provide different services. SDP is used to propagate them to other Bluetooth devices in reach. It additionally includes all necessary parameters for other devices to establish a connection to the specific service.
- RFCOMM (Radio Frequency Communication): Allows for emulation of serial RS-232 ports.

2.2 Human Interface Device

Bluetooth defines a profile that allows to provide and consume human interface device (HID) services via a wireless channel. The main steps for Bluetooth HID functionality can be grouped into four different stages at different levels (figure 2.2).

1. Discover Devices: At the baseband layer Bluetooth devices can discover each other and obtain basic information like the Bluetooth address and the class of device (CoD). The CoD gives general information about the device (e.g. Phone, Peripheral, TV). It can be used by the user to identify the target device and to restrict further scanning to devices that may provide the desired service.
2. Discover Services: After knowing the devices in reach further information can be obtained. The actual services provided by a device are obtained by utilizing the Service Discovery Protocol (see chapter 2.2.2). The result is a list of all available services of

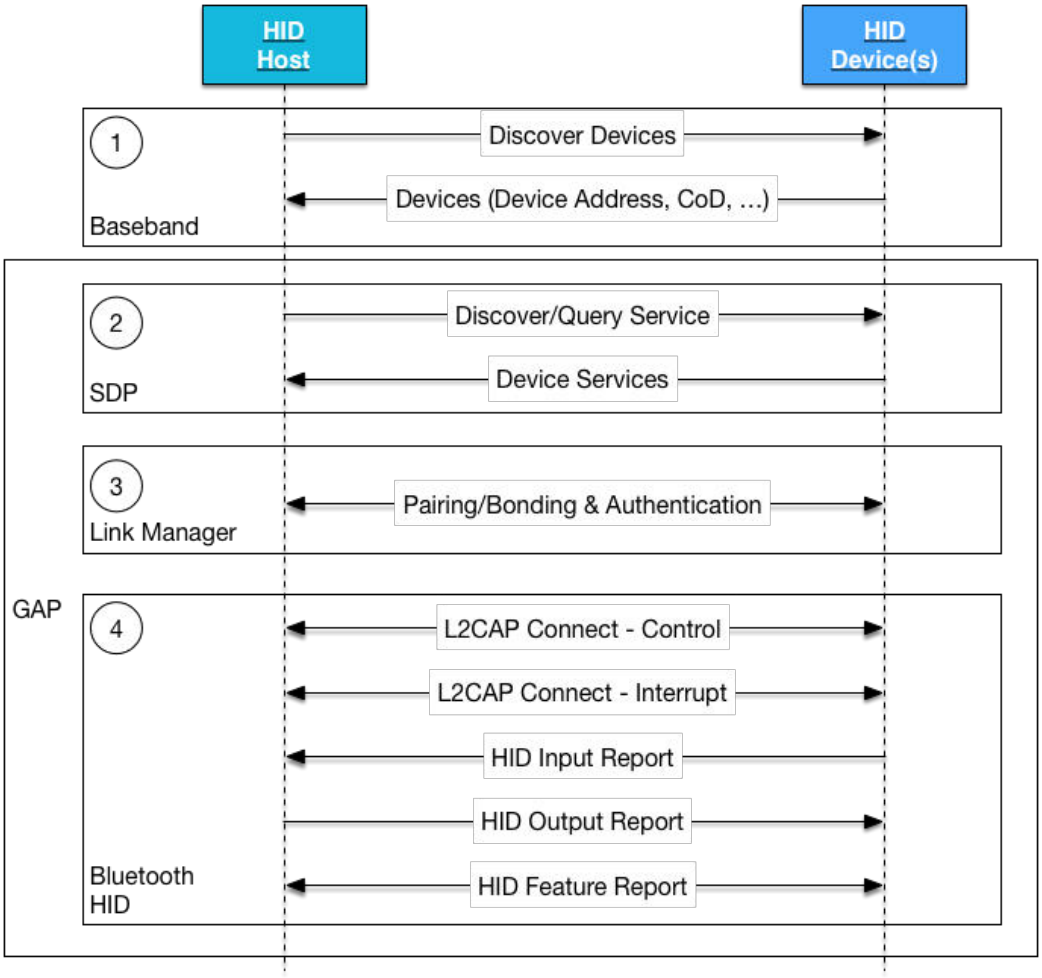


Figure 2.2: Bluetooth technologies used for HID

the device and information how to connect to a specific service. The Generic Access Profile (see chapter 2.2.3) serves as a base profile and defines which and how services are discovered.

3. Pairing/Bonding and Authentication: Before a Bluetooth device can access services from another device, trust between them must be established. This is done via a process called pairing and results in a "bond" between two devices. After this, the devices can authenticate each other on further connection attempts in the future (see chapter 2.2.3).

4. Bluetooth HID: A connection to a Bluetooth HID device is established by opening two L2CAP (see chapter 2.2.1) connections (as specified in the corresponding SDP entry) to the HID device. Actual data is transferred in the form of reports (see chapter 2.2.4).

The report type (Input, Output, Feature) depends on the direction and purpose of the sent data. The messages are modeled after the USB HID specification.

All information in this subchapter is taken directly form the Bluetooth Core specification[8] or the Bluetooth HID specification[9]. Information form different sources are explicitly marked in the text.

2.2.1 L2CAP

The Logical Link Control and Adaptation Protocol (L2CAP) allows higher level protocols to send and receive packets up to 64 kilobytes. It can act connection-oriented or connectionless. L2CAP provides logical channels which are multiplexed over on or more logical (lower level) links. Figure 2.3 gives an overview of the L2CAP layers and the communication to the surrounding layers.

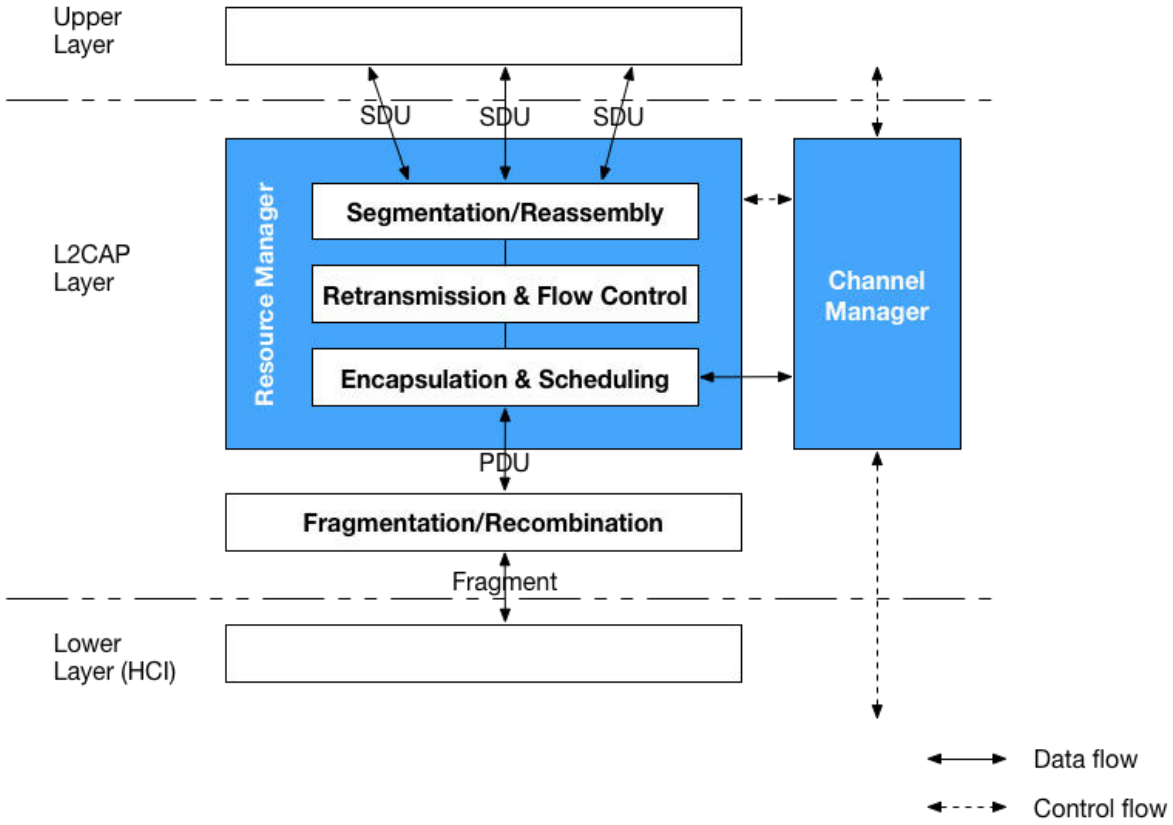


Figure 2.3: L2CAP[8]

Upper Layer The layer on top of L2CAP which uses it. The actual payload is sent in form of packages, with a maximum of 64 kilobytes, called Service Data Units (SDU). Communication from and to the L2CAP layer is done via those SDUs.

Channel Manager The L2CAP layer transfers data to other devices over logical channels. It is possible to have multiple channels to multiple devices (in the same piconet) at the same time. Each channel has a unique identifier called CID (channel identifier), which is used to identify the corresponding upper layer to which the data belongs.

L2CAP distinguishes between two different types of connections:

- Connection-Oriented: Allows full-duplex communication between both sides. The unique channel identifier alone is used to identify the correct upper layer.
- Connectionless: Allows communication in a single direction only. One fixed channel identifier is used for the whole connectionless traffic. To identify the correct upper layer an additional field is inserted into the L2CAP header, the Protocol/Service Multiplexer (PSM). With the PSM the correct upper layer can be determined, either through fixed PSM values defined in the specification (e.g. for RFCOMM) or dynamic values from a SDP entry (see chapter 2.2.2).

The channel manager communicates with the channel manager on the other device. Its task is to create, manage and destroy the channels, which is done via a state machine.

Resource Manager The resource manager is responsible for the actual transmission of data. It takes data in form of Service Data Units (SDU) from higher levels and transforms them to Protocol Data Units (PDU) for the layer below. It is responsible for the following tasks:

- Segmentation/Reassembly: Segmentation is the process of transforming a SDU into one or more segments suitable for transport via a L2CAP channel. Reassembly is the reverse procedure which creates a SDU from one or more segments.
- Retransmission and Flow Control: Data can get lost when transferring from one device to another. Therefore mechanisms for error checking and automatic retransmission are provided. They protect against lost packets and packets that have been accepted but contain an error.

  Flow control is done for each individual L2CAP channel using a window based mechanism.

- Encapsulation and Scheduling: Encapsulation transforms SDUs in PDUs and vice versa. It adds the necessary L2CAP protocol elements (header and checksum fields) to the incoming SDU.

  Scheduling is used for basic traffic management to ensure that there is no buffer overflow in the controller. Additionally it ensures that QoS (Quality of Service) parameters, agreed upon in the connection establishment process, are enforced.

Fragmentation/Recombination This layer allows a different fragmentation than the PDUs provided by the L2CAP segmentation. This may be required, if the controller (below the HCI layer) needs a different fragment size than the Bluetooth stack (above the HCI layer). It allows flexibility between Bluetooth hardware and Bluetooth software.

Lower Layer The L2CAP layer communicates with the layers below via the HCI (Host Controller Interface). The layers below are responsible for the actual wireless transfer.

2.2.2 SDP

The Service Discovery Protocol (SDP) allows for devices to discover each other's available services and their characteristics without consulting a third party. It allows for browsing without prior knowledge, but also searching based on specific attributes or service class. It must handle changes in available services due to changes in the radio frequency proximity.

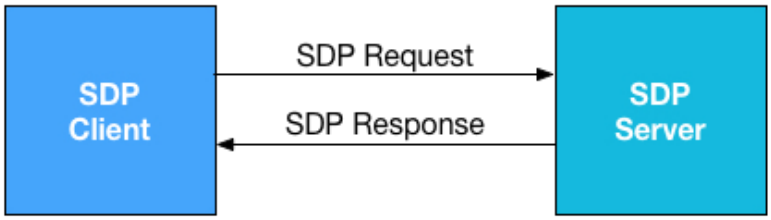


Figure 2.4: SDP interaction[8]

Communication is outlined as messages between a SDP client and a SDP server (seen in figure 2.4). The server knows all available services of the device it is managing. Each service is represented by a service record. The SDP client is the counterpart that queries for service records of other devices. SDP is only used for obtaining information about services and their attributes, it does not provide functionality to utilize those services.

All information about one service is contained in a single service record. A service record consists of multiple service attributes and is uniquely identified by a 32-bit number (record

handle) within a SDP server. A service record with the record handle 0x00000000 represents the SDP server itself and contains basic information like the SDP protocol version.

A single service attribute consists of two components. An attribute ID (16-bit unsigned integer) and an attribute value (variable length; the meaning is determined by the ID). The attribute value itself is a data element which consists of a header field (data type and data size) and a data field. The available attribute types, identified by the attribute ID, are split into two groups. There are attributes common to all service records and attributes defined by a specific service class.

Common Attributes

- ServiceRecordHandle (mandatory): A unique ID that identifies the SDP record within one SDP server.
- ServiceClassIDList (mandatory): A list of all classes this service is part of. The possible values are listed in the specification. It can contain multiple elements but must contain at least one.
- ServiceRecordState: This value is helpful when caching is used. When anything in the record changes this value must change too (usually incremented by one). Clients can check this single attribute and see if there local cached SDP entry is obsolete.
- ServiceID: Unique identification of this specific service described by the record. This ID is stable between different SDP servers.
- ProtocolDescriptorList: Specifies the protocol stack(s) used to gain access to the provided service.
- AdditionalProtocolDescriptorList: If a service needs more than one communication channel, defined in ProtocolDescriptorList, they can be defined in this attribute.
- BrowseGroupList: Can contain one ore multiple browse groups to which the entry belongs. Usually all entries are in the public browse root, but it is possible to create a group hierarchy.
- LanguageBaseAttributeIDList: With this attribute it is possible to provide the human readable attributes in different languages.

- ServiceInfoTimeToLive: The number of seconds since the record was received from the SDP server, for which it remains valid and unchanged. This value is no guarantee and it is used by clients to adapt their polling interval.
- ServiceAvailability: This value represents if the service is available at the moment, meaning that an additional client can connect.
- BluetoothProfileDescriptorList: Contains a list of all Bluetooth profiles the service provides. The specific profile IDs are documented in the specification.
- DocumentationURL: An URL pointing to the documentation of the service.
- ClientExecutableURL: An URL pointing to an executable that can be used to connect to this service.
- ProviderName: A textual name of the service creator.
- IconURL: An URL pointing to an icon that may be used to represent the service.
- ServiceName: A textual name for the service which should be human readable.
- ServiceDescription: A textual description of the service.

HID Attributes

- HIDParserVersion: Specifies the version of the USB HID specification that serves as a basis for the implementation.
- HIDDeviceSubclass: Specifies the type of input device implemented. It is split into two parts. The first two values specify if the device is a keyboard, a pointing device, both or none. The second parts allows a fine grade device specification (e.g. joystick, gamepad, remote control).
- HIDCountryCode: Specifies the country code for which the hardware is localized (e.g. the layout printed on a keyboard) or zero if not applicable.
- HIDVirtualCable: Specifies if the implementation supports the virtual cable feature. This is used to ensure a one to one link between the HID host and the HID device. If set, the HID device cannot be paired to any other HID host without interaction by the user (e.g. by pressing a button on the device).

- HIDReconnectInitiate: Specifies if the device initiates a reconnection process after a dropped connection or not.
- HIDDescriptorList: Defines the byte format of the messages according to the USB HID specification[30] (called input, output and feature reports).
- HIDLANGIDBaseList: Analogous to the LanguageBaseAttributeIDList attribute, it allows to provide human readable attributes, of USB HID related strings, in different languages.
- HIDBatteryPower: Specifies if the device is battery powered and therefore power management considerations should be taken into account.
- HIDRemoteWake: Specifies if the device is/should be capable of remotely waking the connected host device.
- HIDSupervisionTimeout: Allows to specify a recommended value for link supervision timeout. This value specifies the maximum allowed time with no received packet. After that the connection is considered lost.
- HIDNormallyConnectable: Specifies the connection mode of the device when no connection is active. Either a host can connect to the device or the connection has to be initiated from the device itself.
- HIDBootDevice: Specifies if the device supports the Bluetooth HID boot protocol. The boot protocol is a simplified version of the Bluetooth HID protocol. It defines a set of minimal and simplified methods for communication between a HID host and a HID device. This allows a simpler implementation, usually used on embedded devices or small systems (e.g. BIOS).
- HIDSSRHostMaxLatency, HIDSSRHostMinTimeout: Allows to specify values for sniff subrating. Sniff subrating allows to configure parameters for keep-alive messages, which can enhance battery power on devices with a lot of inactivity.

2.2.3 GAP

The Generic Access Profile (GAP) is the base profile all Bluetooth devices have to implement. It describes device discovery, connection establishment and authentication. A device can support initiating a connection, accepting a connection or both. Crucial is that the other device has to support the complimentary functionality.

User Interface Aspects

GAP defines multiple aspects of the graphical user interface an application has to follow, when presenting Bluetooth information to the user.

- Bluetooth Device Address: The Bluetooth address from the baseband is a 48 bit number. If shown in the UI it should be presented as a hexadecimal number, in six groups of two digits, separated by colons.
- Bluetooth Device Name: Every device should transmit a user-friendly name which can be up to 248 bytes long.
- Bluetooth Passkey: If it is necessary that the user sees or types in a key/pin, it should be represented as a decimal number.
- Bluetooth Device Class: The class of device (CoD) is a bit value with a specific meaning defined in the specification. For known classes human-readable names should be presented to the user.
- Appearance Characteristic: A special element that can be mapped to an icon or string. It gives the user a visual hint to the corresponding device.

Discoverability

The discoverability determines if a Bluetooth device can be found by other devices. Special inquiry messages are sent and/or received between devices to transmit information about their existence. The inquire process exchanges information about the Bluetooth device address, Bluetooth device name, clock, class of device (CoD) and page scan mode (connectability). Three different discoverability modes are defined:

- Non discoverable: The device is not discoverable by others. It therefore does not answer to inquiry scans from other devices.
- Limited discoverable: The device is discoverable for a limited time, for a specific event or during temporary conditions.
- General discoverable: The device is discoverable without any constraint.

Connectability

A device can either be connectable or not connectable. If it is connectable is must listen for connection (paging) requests in periodic intervals and respond to them. It not the device can ignore those requests.

Bondable

A device can either be in bondable or non bondable mode. If it is bondable is must accept pairing requests which result in a bond. A device in non bondable mode can accept connections that do not end in a bond.

Bonding creates a relationship between two devices using a shared secret (link key). This process is called pairing or association model. There are four different models defined, which one is used depends on the I/O capabilities of the devices involved (keyboard, display):

- Numeric Comparison: Used when both devices are able to display a six digit number and accept a yes/no input from the user. The user just has to confirm if those numbers match.
- Just Works: Designed for use cases where at least one device does not have the possibility to display information or accept user input. This model uses the numeric comparison model, only the number is not presented to the user and a yes input is alway assumed.
- Out Of Band (OOB): Provides the possibility to use some kind of out of band mechanism (e.g. Near Field Communication) to discover devices and exchange cryptographic information.
- Passkey Entry: Used when one device has the ability to show the user a six digit number and the other device supports user input. A six digit number is shown on one device and must be typed into the other device.

Authentication and Security

Bluetooth authenticates a connecting device by the knowledge of a shared secret. The shared secret is called link key and is determined in the pairing process. If a link key exists but the authentication process fails, the pairing process must be repeated. The link key is stored on the devices and thus the pairing process is only necessary once.

Beyond authentication Bluetooth defines four different security modes. Which security mode is used is up to the specification of the Bluetooth profile and the actual implementation. Mode 1 - 3 are considered as legacy security modes in the current revision of the Bluetooth specification.

- Security mode 1 - non-secure: No security procedure will be initiated at all.
- Security mode 2 - service level enforced security: No security procedure will be initiated before the channel establishment. After that the service can define the necessary security requirements (authorization, authentication, encryption).
- Security mode 3 - link level enforced security: Security procedures will be initiated before the channel establishment is completed.
- Security mode 4 - service level enforced security: Similar to mode 2, but the possible security requirements are: authenticated link key required, unauthenticated link key required and security optional. Security optional is only allowed for specific services (e.g. SDP). An authenticated link key protects against man-in-the-middle (MITM) attacks, but needs some user interaction while pairing. When using the "just works" association model (where no user input is necessary at all) only an unauthenticated link key is created, which cannot prevent MITM attacks.

See chapter 2.3 for more information about security considerations specific to the Bluetooth HID profile.

2.2.4 HID

The Bluetooth human interface device (HID) profile is based upon the USB HID[64] specification and describes how it can be used in a Bluetooth context. There is a distinction between two different roles:

- Bluetooth HID device: The device providing data input to a host and/or data output from a HID host (e.g. keyboard).
- Bluetooth HID host: The device using the Bluetooth HID device (e.g. PC).

There is no definition which device acts as the Bluetooth master and which as the Bluetooth slave, but usually the HID host is also the master of the piconet.

The actual transfer of data is done via HID reports, whose format is defined in the HIDDescriptorList attribute of the corresponding SDP entry (see chapter 2.2.2). There are three different types of reports:

- Input Report: Low-latency information delivery from the HID device to the HID host.
- Output Report: Low-latency information delivery from the HID host to the HID device.
- Feature Report: Application information that is not time critical and can be sent in both directions.

Bluetooth HID defines two different protocol modes. Report protocol mode is the default and provides the complete functionality, whereas boot protocol mode is a simplified version targeted for small/embedded systems (e.g. the BIOS of a PC). The following description is limited to the report protocol mode.

Bluetooth HID Protocol

The Bluetooth HID protocol uses two L2CAP channels for communication between the HID host and the HID device. The channels are called control (PSM: 0x11) and interrupt (PSM: 0x13) and are used to transfer messages between the devices. Table 2.1 shows all defined messages, the used channel, flow direction and if the recipient has to send an acknowledge upon reception.

HANDSHAKE This message is used to acknowledge SET requests, report an error on GET requests or report an error if an unknown message was received. Sent only by the HID device over the control channel.

HID_CONTROL This message is sent to signal a major state change and can be sent by both devices. If the specified operation is not supported by the device it should be ignored. A state change is for example suspend/exit_suspend, where the device can enable a reduced power mode.

GET_REPORT Used by the HID host to request a specific type of report (input, output or feature) from the HID device. After receiving this message the HID device sends the requested report via a DATA message.

SET_REPORT Used by the HID host to set a specified type of report (input, output or feature) on the HID device. The report is embedded directly in the message.

GET_PROTOCOL With this message the HID host can query the current protocol mode (report or boot) of the device. The response is sent as a data message.

SET_PROTOCOL With this message the HID host can set the desired protocol mode (report or boot) the device should use. The device must support the requested protocol, which is indicated in the SDP entry.

DATA This message is used to transfer reports between the devices. The HID device sends input and receives output reports from the HID host, and vice versa.

Message	Channel	Sent by	Acknowledge
HANDSHAKE	Control	Device	None
HID_CONTROL	Control	Device and Host	None
GET_REPORT	Control	Host	DATA (or HANDSHAKE on error)
SET_REPORT	Control	Host	HANDSHAKE
GET_PROTOCOL	Control	Host	DATA (or HANDSHAKE on error)
SET_PROTOCOL	Control	Host	HANDSHAKE
DATA	Interrupt	Device and Host	None
	Control		Initiated by SET_REPORT or GET_REPORT

Table 2.1: HID protocol messages

2.3 Security of Bluetooth HID

Security of a Bluetooth HID connection is dependent on multiple factors, which can be influenced in part by the HID host and HID device. Depending on the I/O capabilities of the devices and the balance between usability and security, different modes can be used.

2.3.1 Pairing/Bonding

The Bluetooth HID profile requires hosts and devices to support secure simple pairing and legacy pairing (for communication with devices using Bluetooth versions prior to 2.1).

Legacy pairing uses algorithms based on SAFER+ and the input of a PIN by the user for the cryptographic procedure. It is considered weak, because PINs, usually consisting of only 4 bytes, are the only source of randomness for key generation. Such a PIN code can easily be attacked by brute force, if the pairing procedure and one authentication exchange is recorded. The encryption cipher is considered weak, the link key has no expiration and no protection against man-in-the-middle attacks exists.[47]

As a successor secure simple pairing (SSP) was introduced to simplify the pairing procedure for the user and improve security. The goal is to provide protection against passive

eavesdropping and man-in-the-middle attacks. For the available pairing types refer to the information about bonding in chapter 2.2.3.

- Passive eavesdropping protection: Requires a strong link key in combination with secure encryption algorithms. The quality of the link key is determined by the entropy in the generation process. Public key cryptography in form of the Elliptic Curve Diffie Hellman (ECDH) is used. As a result the protection is not dependent on a passkey, pin or any other user input. The passive eavesdropping protection is independent of the association model used, even when there is no user input at all.

- Man-in-the-middle protection: The usage of ECDH does not provide protection against man-in-the-middle (MITM) attacks. In a MITM attack a user, who wants to connect two devices, unknowingly connects both to a third device controlled by the attacker. This third device then relays the information back and forth between the devices of the user. Then the attacker can eavesdrop and modify the data transmitted. For this to work the attacker has to be within reach of both ends of the Bluetooth connection.

  To prevent MITM attacks the numeric comparison and passkey entry association models can be used. Both utilize a six digit number, which has to be confirmed on one or both devices. With this number the user is able to ensure which two devices are creating a connection. Unlike the legacy pairing, this number is not used as an input to the algorithm, it is just an artifact of it. Therefore knowing this number is of no use for a potential attacker.

The SSP process consists of five phases:

1. Public Key Exchange: Both devices exchange the public key from their ECDH public/private key pair and compute a Diffie Hellman key.

2. Authentication Stage 1: This is the only phase that differs depending on the chosen association model. Using the numeric comparison protocol both devices generate a pseudo-random nonce, used to prevent replay attacks. The non-initiating device additionally computes a commitment, using both public keys and its nonce (with HMAC-SHA-256). The commitment and nonce values are exchanged between the devices. The initiating device calculates the commitment on its own an checks if it matches with the received one. Based upon both public keys and both nonce values each device calculates a number (using SHA256) to show the user. This number has to be confirmed to be identical on both devices.

3. Authentication Stage 2: Both devices calculate a new confirmation value using the previously exchanged values. The confirmation value is exchanged between the devices and checked. This check ensures that the pairing was confirmed on both sides.

4. Link Key Calculation: After both sides have confirmed the pairing, the link key is calculated independently on both devices using HMAC-SHA-256. As parameters both Bluetooth addresses, both nonce values and the Diffie Hellman key are used.

5. Encryption: In the last step the encryption key gets generated based on the link key and a random value.

After a successful pairing process the devices can and usually do bond. Bonding means that the keys, created while pairing, are stored on the devices alongside the Bluetooth address. Those devices then form a trusted device pair. Those keys can be used later on to authenticate each other.

2.3.2 Authentication

On each connection attempt between bonded devices, both authenticate each other in a process called secure authentication. It is a challenge-response procedure with which both devices ensure that the other side knows a previously shared secret (link key).

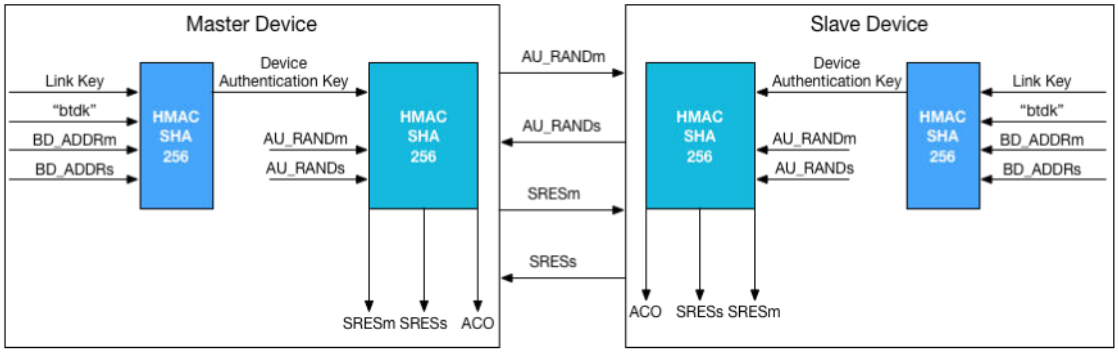


Figure 2.5: Secure authentication[8]

Figure 2.5 shows the secure authentication sequence.

1. Both devices generate a random input, the challenge, AU_RANDm and AU_RANDs and send it to each other.

2. Both devices generate the device authentication key using HMAC-SHA-256, which needs a key and a message as an input. The link key (shared secret) is used as the

key. The message is a concatenation of the Bluetooth device key (the string "btdk" in extended ASCII) and the Bluetooth addresses of both devices. The 128 most significant bits of the result are used further on as the device authentication key.

3. In this step HMAC-SHA-256 is used again. The device authentication key from the previous step is used as the key. The concatenation of the two challenges (AU_RANDm and AU_RANDs) is used as the message. The 128 most significant bits of the result are split into three values SRESm (32 bit), SRESs (32 bit) and ACO (64 bit). Both devices send their SRES value to their communication partner and compare the received value with the one calculated themselves. The ACO value is used later on for the actual encryption of the traffic.

4. If the values match both parties can be sure that the other side also knows the shared secret and is therefore indeed the expected communication partner.

2.3.3 Encryption

Data transferred between Bluetooth devices can be encrypted. Only the actual packet payload may be encrypted, the packet header and access codes are never encrypted. For encryption the stream cipher E0 is used, which is re-synchronized for every payload.

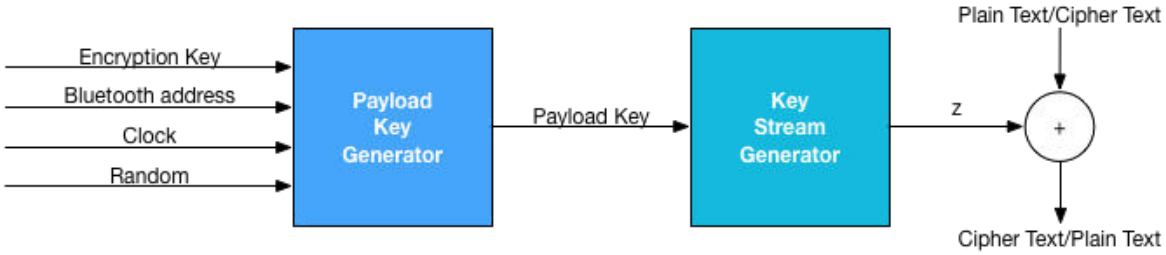


Figure 2.6: E0 stream cipher[8]

As seen in Figure 2.6 E0 consists of three different parts:

1. Payload key generator: Performs the initialization using the encryption key, the Bluetooth address of the master, master clock bits and a random value as an input. The encryption key is calculated using a random number and the ACO value from the authentication procedure. The random value has to be transferred to the slave device (in cleartext). The master's clock value is used, which is received by the baseband layer while negotiating frequencies.

2. Key stream generator: Generates the key stream bits used for encryption/decryption.

3. Encryption/decryption: Encrypts or decrypts the input to or from cipher text.

Unfortunately the E0 algorithm is considered weak and attackable. In a cryptanalysis of E0 in 2005 Lu, Meier and Vaudenay[39] were able to recover the encryption key using a correlation attack. It was necessary for them to analyze $2^{23.8}$ captured frames and it took 2^{38} computations to determine the key. To improve security, Bluetooth 4.1 introduced AES as a possible option for payload encryption. However this is not supported by most Bluetooth HID devices at the moment.

2.3.4 Security Mode 4

Bluetooth HID devices are required to support security mode 4 (service level enforced security - see "Authentication and Security" in chapter 2.2.3). Using this both devices can signal their I/O capabilities to their communication partner and therefore provide information if a keyboard, display or both is available. Then the HID host can signal if an authenticated or unauthenticated link key is required while pairing. Only the authenticated link key can protect against man-in-the-middle attacks.

2.4 Bluetooth for Android

2.4.1 Architecture

For the release of Android 4.2 (Jelly Bean) in 2012 Google switched the default Bluetooth stack and alongside the complete architecture of Bluetooth in Android[13]. The new architecture can be seen in figure 2.7. The biggest change was the introduction of a hardware abstraction layer (HAL), which sits between the Bluetooth stack and a higher level system process. The HAL defines a standardized set of functions a Bluetooth stack must provide. An actual implementation has to provide external access to those functions. Therefore the Bluetooth process is independent of the actual implementation and can work with every stack that provides the interface defined by the HAL. This makes it possible for other vendors to provide drop in replacements for the default Bluetooth stack.

The Bluetooth architecture of Android (figure 2.7) can be divided into four main layers. Every layer only communicates with directly adjacent layers:[2]

Implementation The lowest layer consisting of the implementation of the Bluetooth protocol stack and possibly some vendor extensions. It has to implement the generic Bluetooth HAL interface.

Hardware Abstraction Layer (HAL) The HAL defines a standard interface that is used by the layer on top of it. It ensures independence between the implementation layer and the Bluetooth process layer.

Bluetooth Process The Android system application implementing the Bluetooth service. It utilizes the functions of the Bluetooth stack as defined in the HAL. The methods are called using the Java Native Interface (JNI[46]), which allows to call native code from Java.

Application Framework Represents the application code using the Bluetooth functionality provided by the Android SDK. The application runs in a different thread than the other layers, therefore a binder is used for the necessary inter-process communication (IPC).

2.4.2 Bluetooth Stacks

For the Android platform two major Bluetooth stacks exist, namely BlueZ[10] and BlueDroid[11]. Prior to version 4.2 BlueZ was the default implementation shipped with Android. With version 4.2 and onwards the default was changed by Google to BlueDroid. There was no official reason given to why the change happened.[13]

Table 2.2 gives a short overview over the main differences between the BlueZ and BlueDroid Bluetooth stacks.

	BlueDroid	**BlueZ**
License	Apache	GPL
Documentation	None	Full
Unit Tests	None	Whole stack
Process	Single	Different for Bluetooth service/HAL and Daemon
Bluetooth certified	Up to vendor	Yes

Table 2.2: Comparison of BlueZ and BlueDroid[13,33]

A simple comparison suggests that BlueZ is feature richer and more mature than BlueDroid. Therefore Google's decision to use BlueDroid as the default stack in Android was received

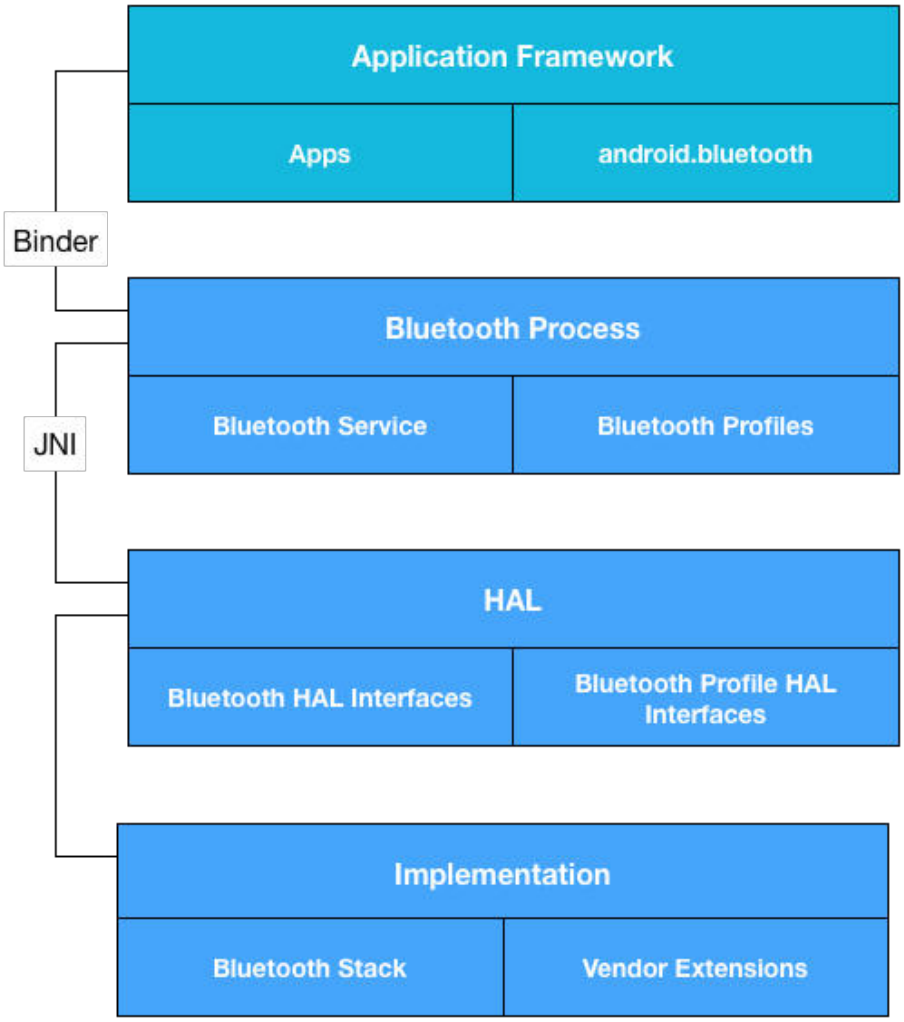


Figure 2.7: Android Bluetooth architecture[2]

with some critic[13,33] and work took place to create a version of BlueZ that is compatible with the Bluetooth architecture of Android 4.2 and above.[10,35]

Both Bluetooth stacks do not provide an implementation of the HID device profile. It is typical for smart-/computer like devices to support the HID host profile only. That means a Bluetooth HID device can be connected, but it cannot act as a Bluetooth HID device. The HID device profile is usually only implemented in simple input devices (e.g. mouse, keyboard, gamepad).

A Bluetooth stack must provide multiple functions so a HID device profile can be implemented. The following list gives a short overview.

L2CAP socket It must be possible to create listening L2CAP sockets on a specified PSM

value. This is necessary for other devices to connect and subsequently for the actual communication.

Alter SDP database A new SDP entry must be created to publish the information about the availability of the HID device profile to other Bluetooth devices. Additional it may be necessary to remove other SDP entries that could interfere with the HID device profile.

Control HID host profile Because the HID host and HID device profile both use the same PSM they cannot be active at the same time. Therefore before starting the HID device profile the HID host profile must be stopped and afterwards started again.

Alter class of device (CoD) Depending on the target device the Bluetooth class of device needs to be set to a HID peripheral value. Therefore it must be possible to change the CoD before activating the HID device profile.

Pairing The initial pairing process should be managed by the Bluetooth stack itself and should be completely invisible to the profile implementation.

Encryption Bluetooth HID for keyboards requires encryption of the data transferred. The necessary functionality for this should be provided by the Bluetooth stack and transparent for the profile implementation.

All those requirements are very low level features basically every Bluetooth stack has to support. Therefore both BlueZ and BlueDroid have support for them. Nevertheless with BlueDroid L2CAP sockets cannot be created with public methods.[4] Additionally the lack of documentation and source comments make it very hard to extend BlueDroid. Therefore BlueZ was chosen for the implementation of this project.

2.4.3 Building BlueZ

BlueZ is not integrated into Android (Lollipop) systems shipped with smartphones or into readily available third party builds. Therefore it is necessary to build a custom Android system image (also called a "build") that incorporates BlueZ as its Bluetooth stack.

Device Requirements

It is not possible to build a custom Android system image for every device that runs Android out of the box. Multiple requirements must be fulfilled:[5]

- Binary Driver: Most devices need several binary drivers, otherwise not all hardware is working properly or at all. For example on a Google LG Nexus 5 drivers for the following elements are necessary: NFC, Bluetooth, Wi-Fi, Camera, Sensors, Audio, Graphics, GSM, GPS, Media, DSP, USB.[29]
- Unlocked Bootloader: The bootloader of the device must be unlocked or it must be possible to unlock it. Otherwise a custom build cannot be flashed onto the device.

Build Machine Requirements

To build a complete Android system a computer running a 64-bit Linux or Mac OS environment is required. Google tests the build procedure regularly with a current Ubuntu LTS Linux distribution. Therefore Ubuntu LTS is the recommended system for building.[25]

Furthermore the following additional software has to be installed on the computer in order to be able to build a Android Lollipop system image.[25]

- Python 2.7: "Python is a programming language that lets you work more quickly and integrate your systems more effectively."[54]
- GNU Make 3.81: "GNU Make is a tool which controls the generation of executables and other non-source files of a program from the program's source files."[17]
- Java Development Kit (JDK) 7: "Java is a programming language and computing platform."[45]
- Git 1.7: "Git is a free and open source distributed version control system."[18]
- Repo: "A repository management tool that we built on top of Git."[24]

Create system image

The process of building consists of getting the necessary source code, compiling it and creating a flashable image for the target device. The precise steps are explained in the next paragraphs and follow the guide provided by the aosp-bluez project[5].

The first step is to get the complete source code of the Android Open Source (AOSP) project. Because the source is spread across multiple GIT repositories the "repo" tool is used. It reads a manifest file that contains all necessary GIT repositories and their location. Additionally it can initialize and download them all with one command. Here the manifest file provided by aosp-bluez for Android Lollipop is used.

Listing 2.1: Obtain Android source code

```
repo init -u https://code.google.com/p/aosp-bluez.platform-
   manifest -b lollipop
repo sync
```

Next all proprietary/binary drivers necessary for the target device, in this case for the Google LG Nexus 5 (codename "hammerhead"), must be downloaded and put into the "vendor" folder. The drivers are provided directly by Google in form of a single archive file for each hardware manufacturer. Each archive contains a single shell script that needs to be executed in the root of the AOSP checkout folder. No user input is required, only the license needs to be accepted by typing "I ACCEPT". The content automatically gets extracted into the appropriate folder.

Listing 2.2: Obtain binary drivers

```
# all Broadcom drivers
wget https://dl.google.com/dl/android/aosp/broadcom-hammerhead-
   lrx21o-01fad5db.tgz
tar xvf broadcom-hammerhead-lrx21o-01fad5db.tgz
./extract-broadcom-hammerhead.sh

# all LG drivers
wget https://dl.google.com/dl/android/aosp/lge-hammerhead-lrx21o
   -c6cf4582.tgz
tar xvf lge-hammerhead-lrx21o-c6cf4582.tgz
./extract-lge-hammerhead.sh

# all Qualcomm drivers
wget https://dl.google.com/dl/android/aosp/qcom-hammerhead-
   lrx21o-e0cd4949.tgz
tar xvf qcom-hammerhead-lrx21o-e0cd4949.tgz
./extract-qcom-hammerhead.sh
```

After the preparation steps the actual build process can take place. To do that, first a helper script (envsetup.sh) is executed. This adds some additional commands in the current shell, which assist in building the whole system or just specified modules. For a complete build the "lunch" command is used, which needs the platform (Google LG Nexus 5) and the buildtype (userdebug) as parameters. The buildtype "userdebug" is the preferred type for debugging. It provides debug functionality and root access.[20] Thereafter the actual build

can be started with "make", in this case with 8 parallel jobs. The build process can take some hours depending on the performance of the build machine.

Listing 2.3: Build the system

```
source build/envsetup.sh
lunch aosp_hammerhead-userdebug
make -j8
```

A successful build creates the kernel, all the necessary binaries and images for the specified target.

Install system image

A new build can only be flashed onto a device with an unlocked bootloader. The necessary steps to unlock the bootloader can be issued with the Android Debug Bridge (adb) tool, included in the Android SDK. On supported devices it is only necessary to reboot into the bootloader and issue the unlock command.[31] This step is necessary only once, the bootloader stays unlocked until it is locked again.

Listing 2.4: Unlock bootloader

```
adb reboot bootloader
fastboot oem unlock
```

The final step is to actually transfer and flash the new build onto the device. This is done again with the "fastboot" tool and the "flashall" subcommand. With this command the complete device is wiped and the new system is installed.[5]

Listing 2.5: Flash Android system image

```
fastboot flashall -w
```

2.4.4 Architecture of BlueZ

The implementation of the Bluetooth functionality in Android is scattered across multiple layers (see figure 2.8). Each layer adds another level of abstraction and has a specific purpose. BlueZ itself is only one part of the whole system and provides the necessary link between an application using Bluetooth and the kernel.

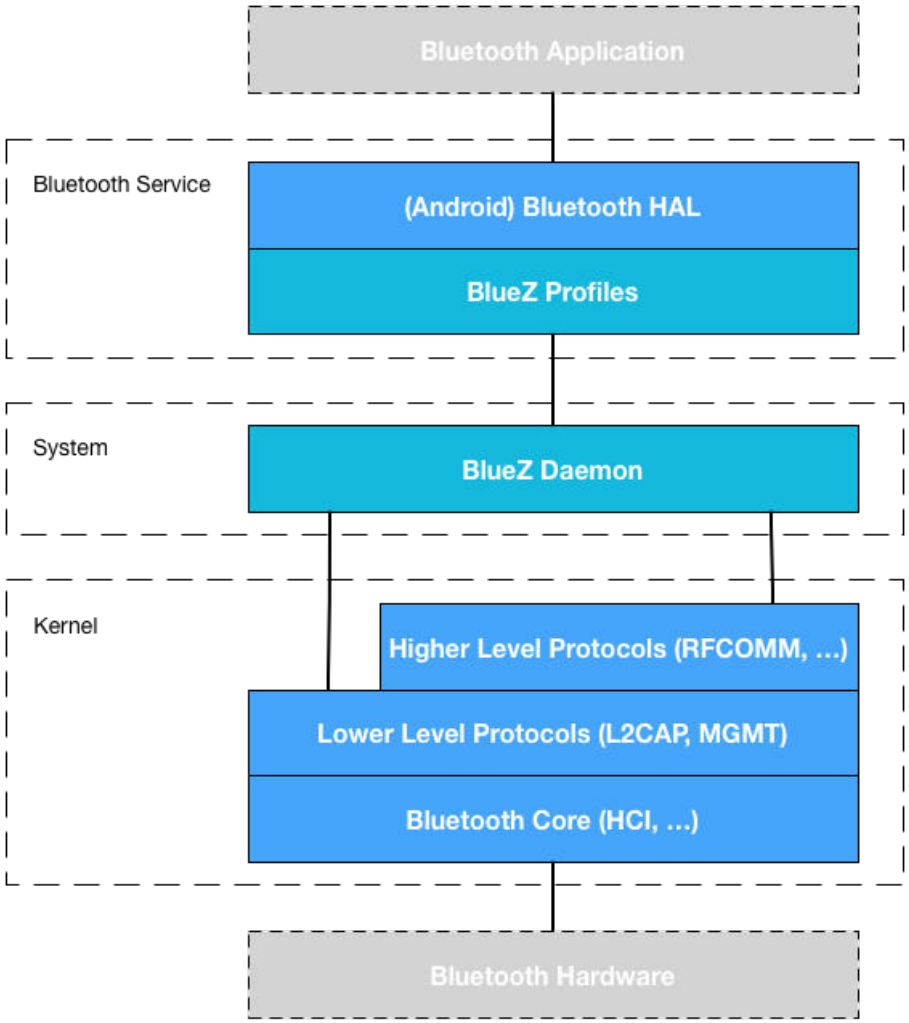


Figure 2.8: Architecture of BlueZ for Android[32][60]

Bluetooth Service The Bluetooth service can be consumed by an application that wants to utilize Bluetooth. Usually only the system's Bluetooth application uses this service directly. The hardware abstraction layer (HAL) defines an interface of standardized functions. This ensures that the application on top is completely independent from the actual implementation of the lower layers, as long as it provides the functions defined by the interface. The BlueZ profiles (BlueZ HAL part) provides the functions defined by the HAL interface. Its main purpose is to handle the interaction between the higher level application and the BlueZ daemon. For that every supported Bluetooth profile exposes the necessary functions and in turn invokes the respective logic in the BlueZ daemon.

System The middle layer represents the system service which contains the BlueZ daemon.

This is an independent service that runs in its own thread and not in the context of a specific application. The daemon hosts the actual functionality and implementation of the BlueZ profiles. It receives and handles commands from the Bluetooth service (BlueZ profiles).

Kernel The kernel provides the Linux Bluetooth subsystem. It implements the Bluetooth protocols that are used by the BlueZ daemon via sockets. There are higher level protocols and lower level protocols. Higher level protocols use lower level protocols to provide their functionality. The BlueZ daemon however can directly use both lower- and higher level protocols. Additionally a set of Bluetooth core functionalities needed by the Bluetooth protocols are provided. This also includes the host controller interface (HCI) which is used to communicate with the Bluetooth hardware (software/hardware boundary).

To add an additional profile it is necessary to add functionality to BlueZ, both in the profile and daemon part. It is not necessary to change or add anything in the Bluetooth subsystem of the Linux kernel. In the BlueZ profiles part it is necessary to add an interface with the functions necessary for the higher layer (system Bluetooth) application. This interface then can be retrieved via the Bluetooth HAL with the unique profile ID. The Bluetooth HAL is designed to be generic, therefore it is not necessary to change it in order to add a new profile. However it is necessary to provide a header file containing the definition of the new profile interface for the higher layer application.

The actual profile implementation is done inside the BlueZ daemon. It interacts with the BlueZ profiles part, manages the connection to another Bluetooth device (via sockets from the Bluetooth subsystem) and is responsible for all other Bluetooth related tasks (e.g. management of the SDP entry).

Because BlueZ is distributed between the Bluetooth service and a separate BlueZ daemon a mechanism for inter-process communication (IPC) is necessary. On BlueZ for Linux D-Bus is used for inter-process communication, which is not available on Android.[13] Instead a simple protocol for communication between the daemon and the HAL part (figure 2.9) is used. There are two different types of communication:[32]

- Command/Response: The HAL sends a command to the daemon. The daemon must answer with a response, else no other command can be issued. Commands are defined by the implementation of Bluetooth profiles as needed. The daemon itself cannot issue a command only answer to one.

- Notification: Notifications are a way for the daemon to signal events to the HAL. In contrast to commands notifications do not require any response from the HAL at all. Notifications can only be sent from the daemon to the HAL.

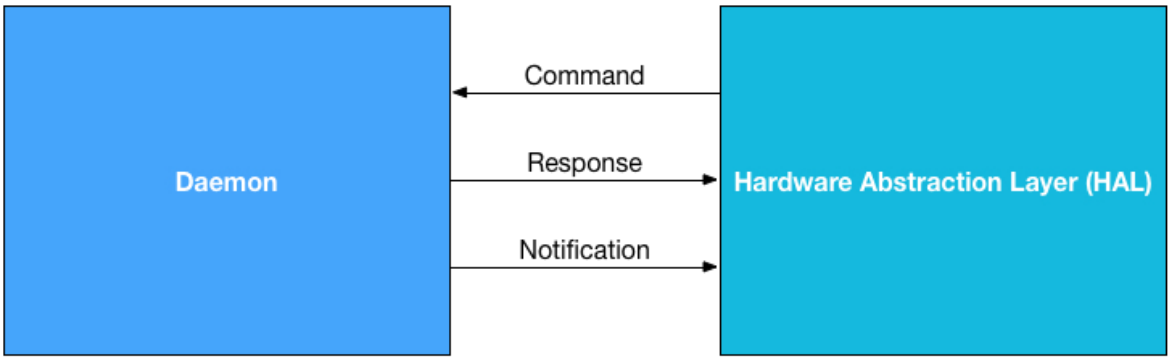


Figure 2.9: Communication in BlueZ for Android[33]

The protocol is implemented by utilizing Linux abstract sockets. A separate socket for the command/response and the notification channel is used, making it possible to run a separate thread for each type. All messages of one type are multiplexed and run over the same socket. To support multiplexing, and therefore allow multiple different profiles to send commands and notifications, every packet has a simple header (figure 2.10) explained in the listing below.[32]

- Service ID: A unique ID defined for every communication partnership/HAL and 0 for the basic control channel.
- Opcode: A ID unique for each Service ID. The opcode should match for each command and response pair. The opcode 0 is reserved for errors. All command and response messages have the least significant bit set whereas it is not set for notification messages.
- Data Length: Specifies the total length of the actual data in the message. The data itself follows directly after the header.
- Data: The actual payload.

2.4.5 Extending BlueZ

To add additional Bluetooth functionality to an Android system using BlueZ, it is necessary to make changes in multiple different modules. At the lowest layer changes in BlueZ, both in

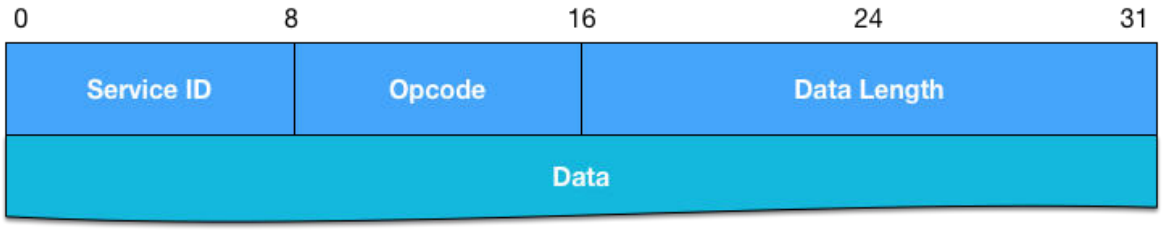


Figure 2.10: BlueZ message header[32]

the daemon and the HAL, are necessary. There the actual extension to the Bluetooth code is implemented. On top of that additions to the Bluetooth system application are necessary. There the new functionality gets invoked and a high level interface to them is provided for other applications. A third party application can access this interface via a service or API extension.

BlueZ

BlueZ provides the actual Bluetooth implementation and therefore the actual extension of the Bluetooth logic has to be implemented there. Only logic concerning the Bluetooth HID device protocols needs to be implemented. All other functionality like pairing, service discovery and L2CAP sockets are provided by BlueZ.

(A) HAL Interface

BlueZ exposes its functionality via a library called "bluetooth.default.so". This library contains functions to obtain interfaces, in the form of C structs, that represent different Bluetooth profiles (e.g. a2dp, handsfree, gatt). This library must contain a number of standardized functions, else it will not be compatible with the Bluetooth system application. But it is possible to add additional functions, for example to expose an additional interface (e.g. HID device profile).

Listing 2.6 shows the functions added to the top level header file "hal.h". This exposes an additional interface for the implemented HID device profile (HIDD). The actual interface is shown in listing 2.7. It contains all functions and callbacks defined for the higher level Bluetooth system application. This represents the sole access point to the the added functionality for other applications.

Listing 2.6: BlueZ - hal.h

```
bthd_interface_t *bt_get_hidd_interface(void);
```

Listing 2.7: BlueZ - bt_hidd.h

```
typedef void (* bthd_state_changed)(bt_bdaddr_t *bd_addr,
   uint8_t state);

typedef struct {
    size_t      size;
    bthd_state_changed  state_changed_cb;
} bthd_callbacks_t;

typedef struct {
 size_t size;
 bt_status_t (*init)( bthd_callbacks_t* callbacks );
 bt_status_t (*start_hidd)( bool sdp, bool bclass );
 bt_status_t (*stop_hidd)( void );
 bt_status_t (*send_keys)( int modifier, int keys[6] );
 void (*cleanup)( void );
} bthd_interface_t;
```

The "bthd_interface_t" structure defines the following functions:

- init: Registers the HID device functionality with the BlueZ daemon. This is required to allow communication between the HAL and the daemon, else no other commands can be issued. This function accepts a callback interface as a parameter. If provided, notifications will be sent on changes to the connection status (host connected, host disconnected).
- start_hidd: Starts the HIDD by registering a new SDP entry and listening for incoming connections. Additional it changes the Bluetooth device class and removes other SDP entries, if specified by the parameters.
- stop_hidd: Stops HIDD by closing the listening sockets, removing the SDP entry and changing the Bluetooth device class back to its original value.
- send_keys: Sends the specified keys, including the specified modifier keys, to the connected HID host. At most six simultaneous keys can be sent/pressed at once.
- cleanup: Unregisters the HID device functionality.

(B) Add Service

To add an additional service to BlueZ it has to provide methods to register and unregister

itself. Every service is registered with a unique Service ID, which is defined in "hal-msg.h". Then the functions service_register and unregister_service in the "main.c" file must be extended to register/unregister the newly created service. This adds/removes the specified service to the BlueZ daemon and allows communication between the HAL part and the daemon part.

(C) HAL Messages

Every service defines commands, responses and notifications to communicate between the HAL part and the daemon. This messages are used by the implementation of the HAL interface (described in section A). Listing 2.8 shows the messages defined for the HIDD service and the data structures used in those messages.

Listing 2.8: BlueZ - hal-msg.h

```
#define HAL_OP_HIDD_START                0x01
#define HAL_OP_HIDD_SEND                 0x02
#define HAL_OP_HIDD_STOP                 0x03

struct hal_cmd_hidd_start {
        bool sdp;
        bool bclass;
} __attribute__((packed));

struct hal_cmd_hidd_send {
        uint16_t modifier;
        uint16_t keys[6];
} __attribute__((packed));

struct hal_rsp_hidd {
        uint16_t data;
} __attribute__((packed));

#define HAL_HIDD_STATE_CONNECTED                0x02
#define HAL_HIDD_STATE_DISCONNECTED             0x00
#define HAL_EV_HIDD_STATE                       0x81

struct hal_ev_hidd_state {
        uint8_t bdaddr[6];
        uint8_t state;
```

```
} __attribute__((packed));
```

The defined messages and their purpose are:

- HAL_OP_HIDD_START: Command/Response sent for starting HIDD. The command contains a hal_cmd_hidd_start structure that defines if the Bluetooth device class should be changed and if all other SDP entries should be removed. The response returns a generic hal_rsp_hidd structure with an integer result code.
- HAL_OP_HIDD_STOP: Command/Response sent for stopping HIDD. The command will restore the Bluetooth device class and SDP entries to their original values. It does not have any parameters and the response returns the generic hal_rsp_hidd structure.
- HAL_OP_HIDD_SEND: Command/Response sent for transferring key-press events to a connected HID host. The command includes a hal_cmd_hidd_send structures, that defines the pressed keys and the pressed modifier keys.
- HAL_EV_HIDD_STATE: Notification sent on changes of the connection state to a HID host. This notification sends a hal_ev_hidd_state structure that contains the host's Bluetooth address together with the state (either connected or disconnected).

(D) Bluetooth CoD

The previous points all described how BlueZ can be extended to provide additional functionality. But for HIDD to work correctly on all target hosts, it is necessary to modify existing functionality. Specifically the Bluetooth class of device (CoD), which propagates the device type, needs changing. Usually an Android device advertises itself as a Phone/Smartphone. For HIDD it needs to be changed to Peripheral/Keyboard. Doing so may prevent other services from being discovered by different Bluetooth devices (depending on their individual behavior). Therefore the CoD is only changed if necessary and while actually using the Bluetooth HID functionality.

Changing the CoD can be done by adding an additional code to the "bluetooth.c" file. This file already contains the code that sets the initial Bluetooth CoD, therefore the new code can be modeled after that. To change the code a special management command is sent, which sets the device class (MGMT_OP_SET_DEV_CLASS) on a specific adapter. Listing 2.9 show how that is done.

Listing 2.9: BlueZ - bluetooth.c

```
#define ADAPTER_HIDD_MAJOR_CLASS 0x05 /* Peripheral */
#define ADAPTER_HIDD_MINOR_CLASS 0x40 /* Keyboard */

static void set_adapter_class_hidd(void)
{
 struct mgmt_cp_set_dev_class cp;
 memset(&cp, 0, sizeof(cp));

 cp.major = ADAPTER_HIDD_MAJOR_CLASS;
 cp.minor = ADAPTER_HIDD_MINOR_CLASS;

 if (mgmt_send(mgmt_if, MGMT_OP_SET_DEV_CLASS, adapter.index,
     sizeof(cp), &cp, NULL, NULL, NULL) > 0)
  return;

 error("Failed to set class of device");
}
```

(E) Bluetooth HID - Keyboard

The main logic that implements the Bluetooth HID device is spread across multiple files in the android folder of BlueZ. It is separated into multiple files all starting with "hidd", which is the file naming convention for BlueZ profiles. Those extend the BlueZ functionality without interfering with other Bluetooth profiles.

This code is either called directly by the HIDD handler, which receives the commands sent by the HAL part, or by a separate thread that handles the actual communication with an individual HID host. The following list gives an overview of the main parts.

hidd-sdp Functionality to manage the specific SDP entry to promote the HID device profile to other Bluetooth devices. Additional management of other SDP entries is possible. The BlueZ functions to add and remove a SDP entry are defined in "bluetooth.h".

- sdp_withdraw_entries: Removes all SDP entries from the SDP database. This is necessary to increase compatibility with different Bluetooth implementations, because some hosts only connect to Bluetooth HID devices which do

not expose any other service/SDP entry. All removed entries are saved so it is possible to restore them later on.

- sdp_restore_entries: Restores all previously saved SDP entries and adds them to the SDP database again.
- sdp_add_hidd_entry: Adds a new SDP entry to advertise the Bluetooth HID device functionality to other Bluetooth devices. The record specifies a keyboard device with no support for the boot protocol. It also contains the L2CAP PSM to which hosts can connect to.
- sdp_remove_hidd_entry: Removes the SDP entry related to the Bluetooth HID device implementation.

hidd-server The server creates sockets listening for incoming connections from HID hosts. Two sockets are necessary, which provide the control channel and the interrupt channel. A connection attempt on each channel is forwarded to the hidd-client implementation.

- hidd_server_start: Opens two listening sockets to register for connections on the control and interrupt channel. Additionally an extra thread is started that polls the sockets periodically to check if (new) data is available. In that case the connection information, including the socket, is passed to the client handler (hidd-client).
- hidd_server_stop: Stops the thread and closes the listening sockets.

hidd-client The client takes connection requests received by the hidd-server and manages the connection to a single Bluetooth HID host. It sends notifications on connection state changes (host connected, host disconnected) and sends/receives messages or reports according to the HID specification.

- hidd_client_connect: Called by the hidd-server to indicate a connection attempt of a host on either the control or interrupt channel. If a successful connection on the control and interrupt channel was created, a thread is started that polls for data periodically. Incoming messages are handled according to the HID specification.
- hidd_client_send_keys: Sends a key-press event to the connected host with the specified keys and modifier keys.
- hidd_client_stop: Disconnects the HID host, stops the thread and closes all sockets.

(F) Deploy

After changing the BlueZ source code multiple steps are necessary to compile the source and install the new binaries onto the Android device. Listing 2.10 shows the necessary commands which are explained in the following listing.

a) Compile: Changing the source of a module does not require a full recompile of the Android system. It is possible to recompile just a specific module. Before the recompile the environment must be initialized which is done by sourcing a shell script and defining the build type with the "lunch" command (for details see chapter 2.4.3). The actual build is started with the "mmm" command, which compiles all modules in the specified directory without their dependencies.[26] In case of BlueZ the necessary HAL libraries and the daemon binary are created. Any errors, warnings and success messages are shown on the terminal.

b) Stop Bluetooth: Before a new version can be installed onto an Android device the Bluetooth service has to be stopped. That can be done by using the "service" command provided by Android. To run it from the development machine the "adb shell"[19] command can be used.

c) Transfer binaries: To update the Android device two files must be replaced. The HAL library (bluetooth.default.so) and the BlueZ daemon (bluetoothd-main). The copy process can be done with the "adb" command.[19]

d) Start Bluetooth: Analogous to stopping the Bluetooth service it can be started again.

Listing 2.10: BlueZ - Compile and Deploy

```
# Compile
source build/envsetup.sh
lunch aosp_hammerhead-userdebug
mmm external/bluetooth/bluez/android

# Stop Bluetooth
adb shell service call bluetooth_manager 8

# Transfer binaries
adb push out/target/product/hammerhead/system/lib/hw/
   bluetooth.default.so /system/lib/hw
```

```
adb push out/target/product/hammerhead/system/bin/bluetoothd-
   main /system/bin

# Start bluetooth
adb shell service call bluetooth_manager 6
```

Bluetooth Application

The Bluetooth system application shipped with Android provides the necessary interface to use Bluetooth. It is responsible for actually starting and managing the Bluetooth stack. There are two different ways of interacting with it. First a graphical user interface for the user to manage basic Bluetooth tasks like enable Bluetooth, disable Bluetooth, search other Bluetooth devices in reach and initiate the pairing process. Second an API and service interface so other applications can integrate and call Bluetooth functionality.

(A) JNI Wrapper

At the lowest level of the Bluetooth application is the code that communicates with the Bluetooth stack (BlueZ). This is done by using the interfaces exposed by the HAL library. This code therefore needs to be written in C (native code). It encapsulates the calls to the Bluetooth stack and provides higher level functions. They can be called by the Java part of the Bluetooth app via Java Native Interface (JNI). Creating an application that uses both, native C-code and Java application code, is supported via the Native Development Kit (NDK).[28]

To retrieve the top level Bluetooth interface a method already exists in the native code of the Bluetooth app. With that the previously defined HIDD interface can be retrieved by name (see listing 2.11). With the retrieved interface all published functions can be called and callbacks can be registered.

Listing 2.11: Bluetooth Application - com_android_bluetooth_hidd.cpp (Interface)

```
const bt_interface_t* btInf = getBluetoothInterface();
btInf->get_profile_interface("hidd"));
```

The methods provided via the JNI interface to the Java code can be seen in listing 2.12. This methods call the functions provided by the HID interface defined in BlueZ (as described in the previous section). Initialize/cleanup (de)registers the HID device profile, start/stop actually adds/removes the HID device profile and determines if hosts can connect. Send allows to trigger key-press events on a connected HID host.

Listing 2.12: Bluetooth Application - com_android_bluetooth_hidd.cpp (JNI Methods)

```
static JNINativeMethod sMethods[] = {
 {"initializeHIDD", "()V", (void *) initializeHIDD},
 {"startHIDD", "(ZZ)V", (void *) startHIDD},
 {"sendKeys", "(I[I)V", (void *) sendKeys},
 {"stopHIDD", "()V", (void *) stopHIDD},
 {"cleanupHIDD", "()V", (void *) cleanupHIDD},
};
```

When a notification is received from the Bluetooth stack the information is relayed without modification to the Java part by calling the method "onStateChanged".

(B) Service Interface

To provide access to the new functionality to other applications a consumable service is necessary. Because the Bluetooth application and a 3rd party application run in different processes, a way for inter-process communication (IPC) is required. For that the Android Interface Definition Language (AIDL) exists. With AIDL only an interface, on which both parties have to agree, is necessary. Upon that the "aidl" tool included in the SDK generates the necessary server and client code.[27] Listing 2.13 shows the defined interface.

Listing 2.13: Bluetooth Application - IBluetoothHIDDBinder.aidl

```
interface IBluetoothHIDDBinder {
 boolean start(boolean withdrawSDP, boolean changeClass);
 boolean stop();
 boolean sendKey(int modifier, in int[] keys);
}
```

- start: Starts the HID device profile, adds the SDP record and starts accepting connections from a HID host. It can be specified if the Bluetooth class of device should be changed to peripheral/keyboard and if all other SDP entries should be removed.
- stop: Stops the HID device profile and restored the SDP database and Bluetooth class of device.
- sendKey: Sends the specified keys to a connected HID host.

To publish the service for other applications it is necessary to create a corresponding service entry in the application's manifest file (AndroidManifest.xml) shown in listing 2.14.

Listing 2.14: Bluetooth Application - AndroidManifest.xml

```
<service
 android:process="@string/process"
 android:name=".hidd.HIDDService"
 android:enabled="true"
 android:exported="true"
 android:permission="" />
```

To notify other applications on connection changes a broadcast message is sent on every state change. It can be received by any applications listening for it. This is done by creating an intent, which is an abstract description of an operation, and sending it via the "sendBroadcast" method provided by the context.[23] Listing 2.15 shows how that is done using the ACTION_CONNECTION_STATE_CHANGED event defined in the Android SDK.

Listing 2.15: Bluetooth Application - HIDDService.java

```
Intent intent = new Intent(BluetoothAdapter.
   ACTION_CONNECTION_STATE_CHANGED);
intent.putExtra(BluetoothAdapter.EXTRA_CONNECTION_STATE,
   newState);
intent.putExtra(BluetoothAdapter.
   EXTRA_PREVIOUS_CONNECTION_STATE, oldState);
sendBroadcast(intent, ProfileService.BLUETOOTH_PERM);
```

Application - Credential HID

The application that actual utilizes the Bluetooth HID device profile (Credential HID) can request an instance of the previously defined service with an intent. Listing 2.16 shows how the HIDDService (from the Bluetooth system application) can be used inside the application. An intent to bind the class "com.android.bluetooth.hidd.HIDDService" is provided to the "bindService" method of the application's context. The "ServiceConnection" will be notified a soon as the service is available and an actual instance can be created with the generated class from the "aidl" tool.

Listing 2.16: Bluetooth Application - HIDProvider.java

```
// Manage the state of the HIDD service
ServiceConnection mConnection = new ServiceConnection() {
 void onServiceConnected(ComponentName name, IBinder service) {
  service = IBluetoothHIDDBinder.Stub.asInterface(service);
 }
};

// Request binding to HIDD Service
Intent intent = new Intent();
intent.setComponent(new ComponentName("com.android.bluetooth", "
   com.android.bluetooth.hidd.HIDDService"));
bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
```

To listen for connection state change notifications from the HIDD implementation an intent is used. It must be the same event (ACTION_CONNECTION_STATE_CHANGED) which is sent by the Bluetooth application. The "registerReceiver" method, provided by the application's context, is used to register the handler. The receiver can use the "getIntExtra" method of the received intent to extract additional data. Listing 2.17 shows how to get the new connection state after a change event was received.

Listing 2.17: Bluetooth Application - BluetoothManager.java (1)

```
// Handle received broadcast messages
BroadcastReceiver broadcastReceiver = new BroadcastReceiver() {
 @Override
 public void onReceive(Context context, Intent idt) {
  String action = intent.getAction();
  if (BluetoothAdapter.ACTION_CONNECTION_STATE_CHANGED.equals(
     action)) {
   // Get the connection state information
   idt.getIntExtra(BluetoothAdapter.EXTRA_CONNECTION_STATE, 0);
  }
 }
}

// Register for connection state change event
IntentFilter filter = new IntentFilter(BluetoothAdapter.
   ACTION_CONNECTION_STATE_CHANGED);
```

```
registerReceiver(broadcastReceiver, filter);
```

Besides the newly implemented functionality it is also possible to interact with the default Bluetooth features. This is also done via intents and the "startActivity" method of the application's context. Listing 2.18 shows how the Android device can be set to be discoverable by other Bluetooth devices. With "startActivity" the target is allowed to open a dialog on top of the calling application. In this case a dialog is shown in which the user has to allow the action.

Listing 2.18: Bluetooth Application - BluetoothManager.java (2)

```
// Set Bluetooth discoverable
Intent discoverableIntent = new Intent(BluetoothAdapter.
   ACTION_REQUEST_DISCOVERABLE);
discoverableIntent.putExtra(BluetoothAdapter.
   EXTRA_DISCOVERABLE_DURATION, duration);
startActivity(discoverableIntent);
```

3 Optical Character Recognition

Optical Character Recognition (OCR) is used to convert images containing text (handwritten or printed) into machine readable text. It is typically used on text printed on paper. This application however explores how that processes can be utilized to recognize text from a computer screen, using the camera of a smartphone. Particularly the detection of a URL from a web browser's address bar is most relevant. Figure 3.1 gives an overview of the complete URL recognition process.

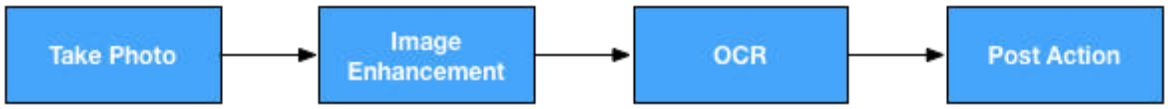


Figure 3.1: URL recognition process

- Take Photo: The first step is to take a photo of the computer screen using the mobile device. The area containing the URL has to be selected manually. It is important to choose suitable camera settings to produce the best possible image for further processing.
- Image Enhancement: Further enhancement of the photo can take place to increase the accuracy of the OCR process. Possible enhancements include noise reduction, background cleanup and brightness adjustment.
- OCR: The Optical Character Recognition (OCR) process itself converts the enhanced image into a machine readable text format.
- Post Action: As a result the extracted URL can be used for the desired action.

3.1 OCR

Optical Character Recognition (OCR) is the process of converting text on an image to editable, machine-processable text. The text image can contain either handwritten or printed

text. OCR faces multiple difficulties like the similarity between different letters/digits (e.g. the digit zero and the letter O). Another challenge is the quality of the input image. Dark backgrounds, askew text and heavy formatting can affect the process negatively. OCR has a wide variety of applications like automatic license plate readers and extraction of text from scanned documents (e.g. books). The accuracy of current OCR software usually lies somewhere between 71% and 98%, depending on the quality of the input image and the used OCR engine.[48]

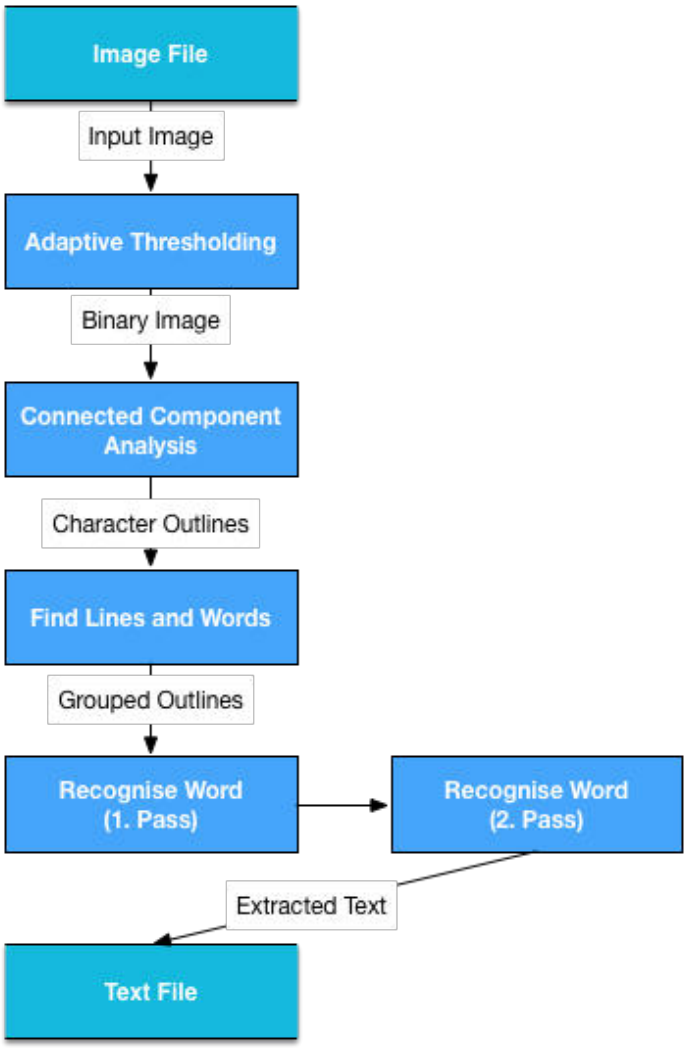


Figure 3.2: Tesseract processing steps[48]

Most available OCR software is neither free nor under an open-source license. One noteworthy exception is Tesseract[62], an open source (Apache license) OCR engine developed, among others, by Google. It is written in C and is available for different platforms, includ-

ing Android. The Tesseract OCR process consists of multiple steps run sequentially (figure 3.2).[48][58]

1. Adaptive Thresholding: Converts the input image into a binary image. That means each pixel is analyzed and based upon the neighbors and threshold parameters it is either set or not. This separates the text from the background and the resulting image can be represented as a monochrome picture.

2. Connected Component Analysis: In this step connected components (characters) are detected and outlined.

3. Find Lines and Words: The outlined components are further analyzed and grouped into text lines and words, based on the character spacing.

4. Recognize Word (1. pass/2. pass): The first pass of the recognition process attempts to recognize the word. If the confidence, that the word was detected correctly, is high enough it gets passed to the classifier as training data. The second pass runs again on all words not recognized initially. Because of the additional training data it may be possible to recognize them now.

3.2 Camera

To produce consistent and accurate results in the OCR process the input image is of high importance. Using a photo of a computer screen as an input for OCR poses multiple challenges. The image from the camera has to be sharp, properly exposed and should not show screen artifacts. Additionally the settings should be chosen to produce a similar result every time. Therefore it may be necessary to adapt exposure settings to accommodate for different screen brightnesses. This allows for better and more targeted image enhancement.

3.2.1 Available Settings

Using Android's full automatic mode to take photos produces non optimal images for OCR. Fortunately the Android SDK provides the possibility to control the camera settings manually. Various relevant parameters are available:

Focus The auto-focus system does not deliver fast and consistent results when trying to focus on a computer screen. Therefore better results can be achieved by using manual focus. Initially it is set to the minimal focus distance, therefore the camera can be

placed as near to the screen as possible. The user can change the focus distance from within the application if necessary.

Aperture Changing the aperture to reach an optimal exposure time would be beneficial. Unfortunately nearly no smartphone has a lens that supports variable aperture. Therefore this option is not further explored.

Exposure Time A long exposure time removes artifacts that arise when photographing a computer screen. Those artifacts arise, because a computer screen consists of many individual pixels. Taking a photo of it, especially from a short distance, can result in a picture that shows those individual pixels, which is a problem for the OCR process. On the other hand, a long exposure time makes it impossible to get a sharp, non blurry image when holding the camera in hand. Therefore it is necessary to strike a balance between a sharp image and a noisy image.

Flash The computer screen provides enough light, therefore using the flash is not necessary. On the contrary the reflection from the flash on the screen creates an unusable picture. Therefore flash is always disabled.

Face Detection Android can analyze the image an search for faces for e.g. focus. Because this information is not needed and only slows the operation down it is disabled.

Color Effect Different color effect modes (e.g. sepia) are available. This also includes the possibility to remove color altogether (grayscale, black and white). OCR engines usually work with black and white images, because the color does not provide any information, only the outline and structure does. Converting an image to black and white is called binarization and is done automatically by the OCR engine (see chapter 3.1). Therefore it is not necessary to create the image as black and white. However it makes sense to create the image as grayscale, because then it is easier to post process (see chapter 3.3).

ISO The ISO value defines the brightness sensitivity of the image sensor. A high ISO value reduces the necessary exposure time, but it also increases the noise in the image. The possible maximum and minimum ISO value depends on the image sensor used in the device.

For further testing aperture, flash, face detection and color effect settings are ignored, because they are not necessary, not available or done as part of image enhancement later on.

Therefore testing concentrates on exposure time and ISO. Focusing is alway done manually to get a sharp image.

3.2.2 Examples

The following examples show images shot with different camera settings and the result achieved running the OCR process on them. The applied camera settings are noted and explained for every example series. All images show text photographed from a computer screen and were shot hand-held with the camera of the Google LG Nexus 5. For optical character recognition Tesseract OCR[62] is used without any additional image enhancement.

Figure 3.3 and table 3.1 show images shot with different ISO values, but always the same exposure time. It can be seen that a higher ISO value increases the brightness of the resulting image. Unfortunately this also increases the noise, which can be seen especially by looking at individual letters. They contain bright spots and not full and strong lines. The OCR result shows that a higher brightness increases the accuracy of the result and the system can cope with the noise up to a high level.

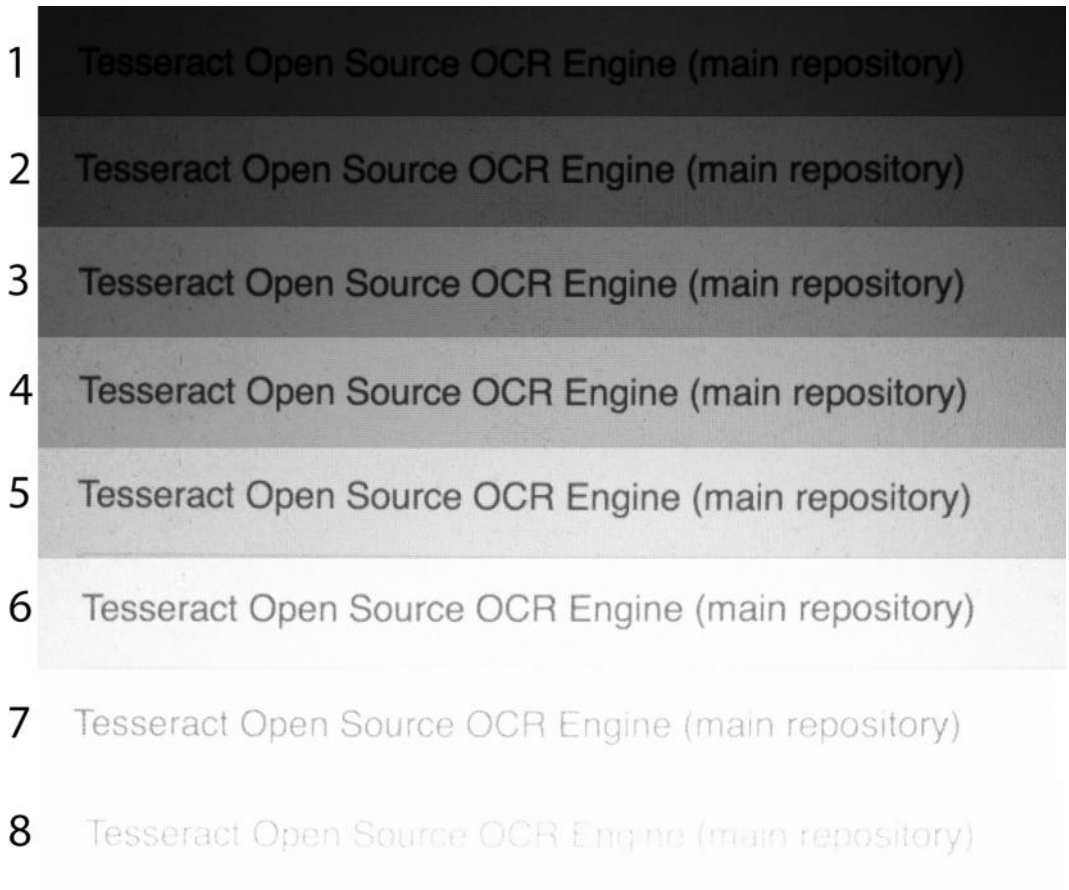


Figure 3.3: ISO series

#	ISO	Exposure	OCR
1	100	0.005	? 3:1?" So" urea 00? REngine (main "epon?t?owl?fi?m
2	200	0.005	-
3	400	0.005	féSSQEact Open Source OCR Engine (main repository) I '- ‘
4	800	0.005	H TesSeract Open Source OCR Engine (main repository)
5	1600	0.005	Tesseract Open Source OCR Engine (main repository) ' '
6	3200	0.005	Tesseract Open Source OCR Engine (main repository)
7	6400	0.005	Tesseract Open Source OCR Engine (mam repository)
8	10000	0.005	Tesseraci 0mm Sc?wrm Oilfi?f 5;; Hf; Maura rerx:5n?ctuy?

Table 3.1: ISO series

Figure 3.4 and table 3.2 show images shot with the same ISO level and different exposure times. The longer the exposure time the brighter the image is. But also the sharpness of the individual letters suffer, because it gets increasingly harder to hold the device steady. The

OCR results show that a certain brightness is necessary, else the background and foreground can not be distinguished. The loss of sharpness seems not to be a big problem, but if the image is too overexposed the result is unusable.

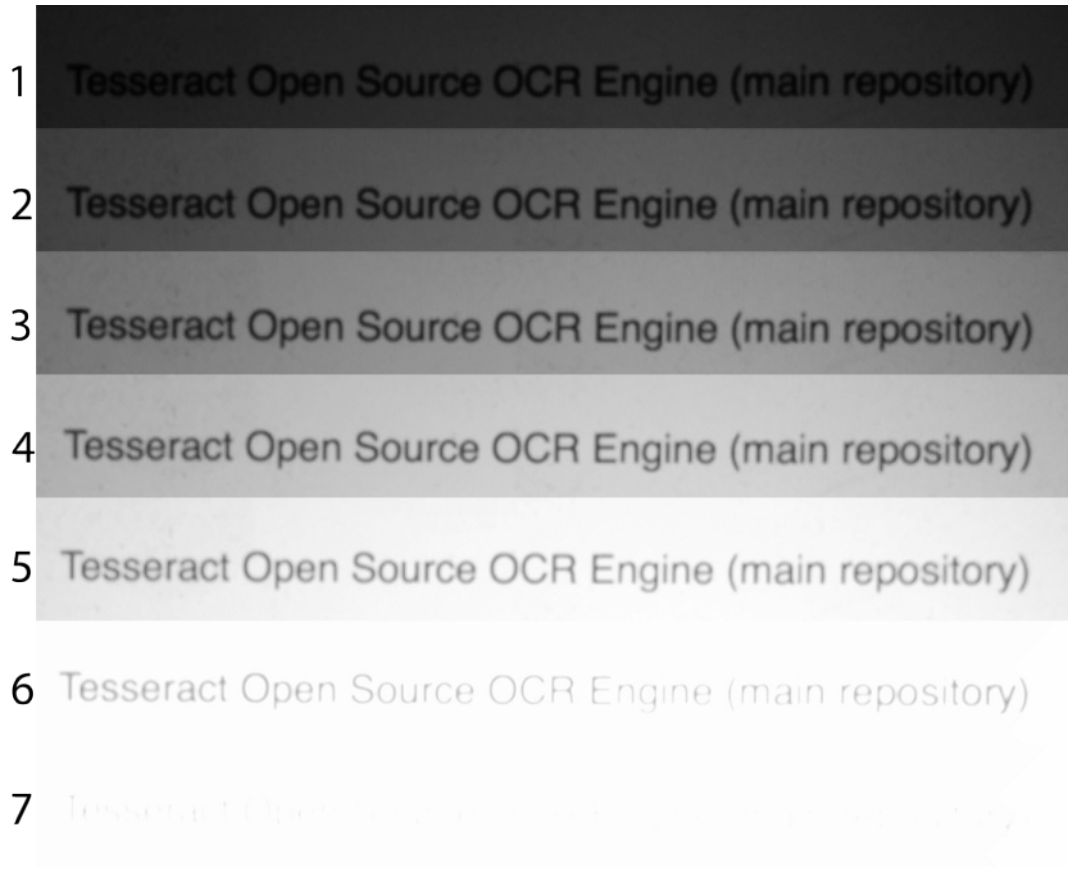


Figure 3.4: Exposure time series

#	ISO	Exposure	OCR
1	100	0.00677	-
2	100	0.01355	".0900 Source OCR Engine (main rapowory)¡ mg
3	100	0.02709	vm Open Source OCR Engine (main repository)
4	100	0.05419	Tmract Open Source OCR Engine (main repositmy)
5	100	0.10837	Tmracl Open Source OCR Engine (main repository)
6	100	0.21674	Tesseract Open Source OCR Enqmu (mam repOSItory)
7	100	0.43349	I.“ ‘ł" il.[‘;" ‘ ’ ‘ ’ ‘ ‘> ’ "

Table 3.2: Exposure time series

Figure 3.5 and table 3.3 show the interaction between ISO and exposure time. For each

sample both values are chosen, so that the resulting brightness is always roughly the same. The higher the ISO value gets the lower the necessary exposure time. This increases the sharpness, but also the noise level. The OCR result show that the noise in the image can be compensated up to a medium high level and the increased sharpness at that level produces a fairly good result.

Figure 3.5: ISO end exposure time series

#	ISO	Exposure	OCR
1	100	0.02709	' "Team ? 'ract Open Source OCR Engine (main rePOS?tO'Y) I
2	200	0.01355	. M'l'esse'ract Open Source OCR Engine (main repository) .
3	400	0.00677	7. Tesseract Open Source OCR Engine (main repository)
4	800	0.00339	Tesseract Open Source OCR Engine (main repositow) ?
5	1600	0.00170	?jmwd Open Source OCR Engine (main repository) I
6	3200	0.00085	-
7	6400	0.00042	-
8	10000	0.00021	-

Table 3.3: ISO end exposure time series

As a conclusion of all the test runs, the strategy to achieve the best pictures for further enhancement and OCR is to choose a ISO value of 400 or 800 and slightly overexpose the picture. To achieve consistent results, on different monitors with different brightness settings, it is useful to get the exposure time suggested by the automatic provided by Android and then set a manual exposure compensation.

3.3 Image enhancement

Image enhancement can help to achieve better results and higher accuracy when using OCR. Tesseract OCR always does some image processing using the Leptonica library.[62] Nevertheless better results can be achieved by doing manual image enhancement and use operations and parameters best suited for the specific use case.

For image enhancement the convert tool provided by ImageMagick[34] is used. ImageMagick is a free software for image manipulation and also available for the Android platform. All image enhancement examples are based on the image seen in figure 3.6. It was shot with the camera of the Google LG Nexus 5 (ISO 100, 0.01355 seconds exposure time) of a computer screen. The image was already converted to grey scale for easier processing. For the actual character recognition process the Tesseract OCR engine is used. Running the character recognition on this image produces the erroneous output "".0900 Source OCR Engine (main rapowory)¡ mg".

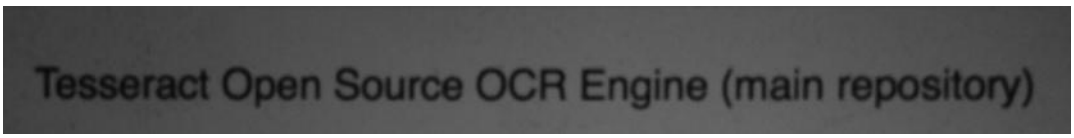


Figure 3.6: Base image

3.3.1 Contrast

By stretching the contrast it is possible to separate the text from the background. Optimization is done on an individual pixel level. If the white level of a pixel is over a specified threshold it is set to be completely white. Pixels over a specified black level are set to completely black. The subcommand used is "-contrast-stretch AxB%". The first value (A) specifies the threshold for the black point and the second (B) for the white point.[34]

Figure 3.7 and table 3.4 show different parameters for white point threshold. Figure 3.8 and table 3.5 show different parameters for black point threshold. The result shows that the white threshold can be set quite high for maximum background cleanup. The black adjustment should be low, so that the text gets stronger but the background noise does not get too strong.

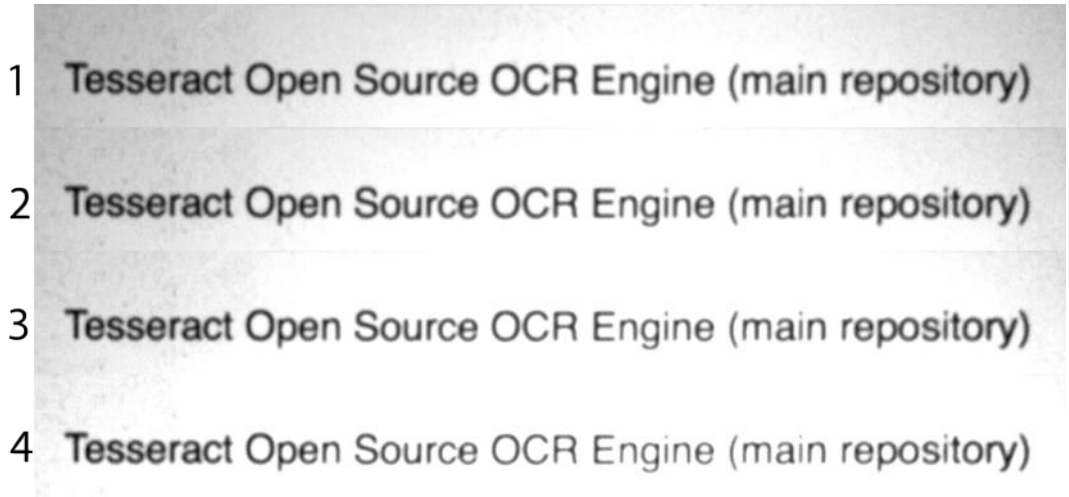


Figure 3.7: Contrast - White adjustment

#	Parameter	OCR
1	0x20	WW Open Source OCR Engine (main repository) ‘
2	0x40	W I ’* I Open Source OCR Engine (main repOSitO’Y)
3	0x60	rTmoracl Open Source OCR Engine (main repository)
4	0x80	Tosseract Open Source OCR Engine (main repository)

Table 3.4: Contrast - White adjustment

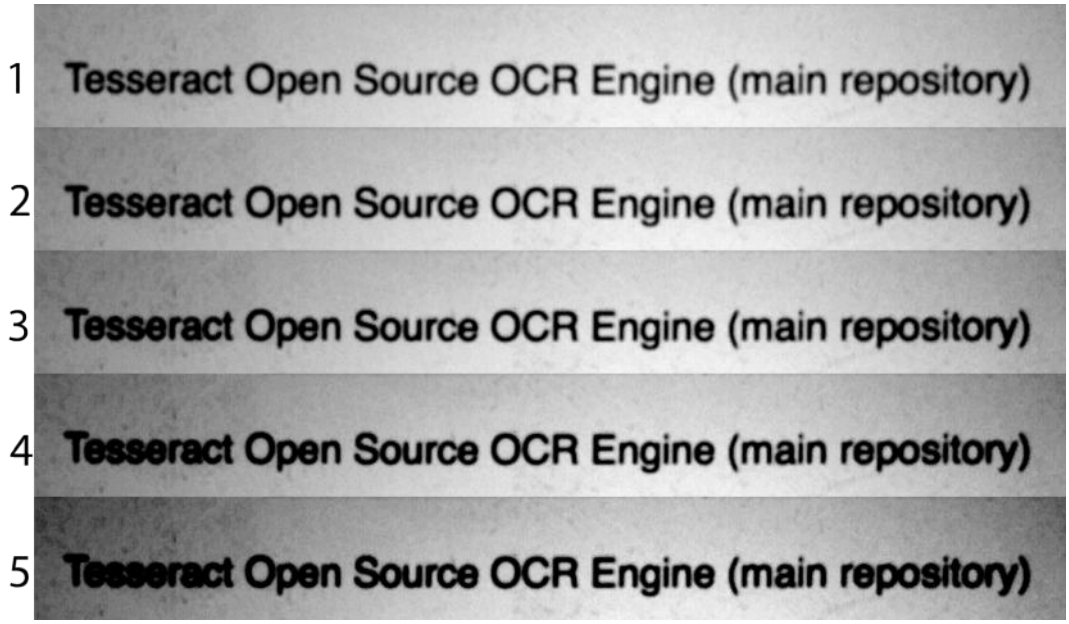


Figure 3.8: Contrast - Black adjustment

#	Parameter	OCR
1	1x0	Open Source OCR Engine (main mposthory)‘ W
2	3x0	Finn Sounoo OCR Engine (main
3	5x0	Opon V Samoa OCR Engine (main My *1 “a!
4	7x0	. ywwmmw’m‘ r ’m 00" Engine (’"al‘" [mm 1o
5	9x0	-

Table 3.5: Contrast - Black adjustment

Combining the best results determined for white (80%) and black (5%) threshold creates the image seen in figure 3.9. This allows Tesseract to extract the text without any error: “Tessaract Open Source OCR Engine (main repository)”.

Figure 3.9: Contrast - Black and white adjustment

3.3.2 Background clean-up

The goal of background clean-up is similar to contrast stretching. It allows to separate the text from the background. In contrast this is not done by looking at individual pixels, but by looking at the neighboring pixels. If a pixel is brighter than the average of the neighbors plus/minus a specified threshold it is made white, otherwise black. The subcommand used is "-lat XxY-Z". The value X and Y specify the window size at which pixels are considered neighbors. This value has to be smaller than the thickness of the font. The value Z specifies the threshold.[34]

Figure 3.10 and table 3.6 show this process with different threshold values. The window is specified as 13x13 which matches the font. It can be seen that low values create a noise pattern that make the image unusable for further OCR. In contrast higher values clean up the background quite good and deliver a perfect result. Too high values white out to much of the text itself and the accuracy drops again.

Figure 3.10: Background clean-up

#	Parameter	OCR
1	300	-
2	600	-
3	900	-
4	1200	Tesseract Open SourceOCR Engine(main repository) ‘
5	1500	Tesseract Open Source OCR Engine (main repository)
6	1800	Tessaract Open Source OCR Engine (main repository)
7	2100	”Ibsgeract Open Source OCR Engine (main repository)

Table 3.6: Background clean-up

3.3.3 Sharpening

Sharpening helps to define the outline of the text and removes blurriness. Because of the short exposure time the image is usually quite sharp. Therefore a high value for sharpening

has to be used to achieve any effect. The subcommand used is "-unsharp AxB" where A is the radius used to group pixels together. When set to 0 the optimal value is determined by the library itself. B specifies a standard deviation for the algorithm, this is the main value for influencing the result.

Figure 3.11 and table 3.7 show the result for different deviation values. The visual difference between the different values is hard to spot. But without any sharpening the text is not recognized correctly. Applying just a little amount of sharpening improves the accuracy. But if the value gets too high the accuracy deteriorates.

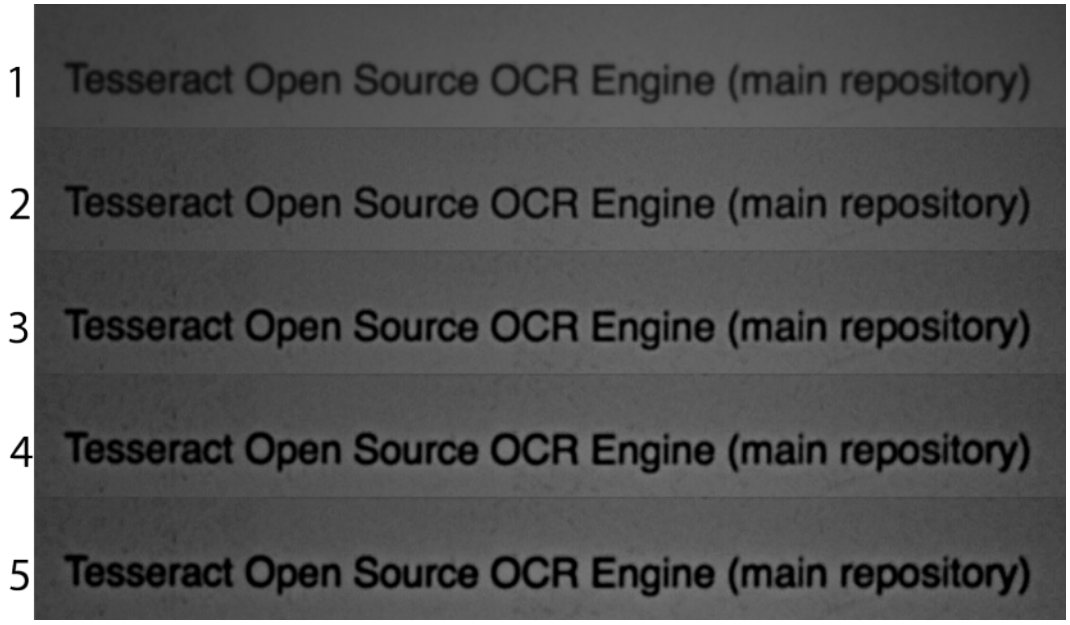


Figure 3.11: Sharpening

#	Parameter	OCR
1	0	".0990 Source OCR Engine (main ram; 1
2	5	Tesseract Open Source OCR Engine (main repository)
3	10	Tesseract Open Source OCR Engine (main repository)
4	15	Tesseract Open Source OCR Engine (main repository)
5	20	Tessoracl Open Source OCR Engine (main repository)

Table 3.7: Sharpening

3.4 Combining results

It is necessary to combine the results determined in the previous subchapters and consider how they influence each other. By doing so the best accuracy for detecting the URL in a browser's address bar from a photo can be achieved. A good picture can be taken with an ISO setting of 400 and an exposure time of 0,01354 seconds. The distance between the screen and the camera should not be too close (about 15 cm). A sample image can be seen in figure 3.12 and yields, without any further image processing, the OCR result "? https://|ogin.microsoftonlin=e.com".

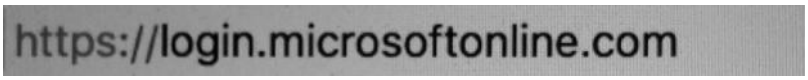


Figure 3.12: Image URL bar

To improve the result image enhancement with a contrast stretch of 1% for black level and 90% for white level, background clean-up with a threshold of 50 and a sharpening of 2 produces the image seen in figure 3.13. This delivers a perfect OCR readout of "https://login.microsoftonline.com".

Figure 3.13: Enhanced image URL bar

3.5 Android

3.5.1 Camera

The Android SDK allows third-party applications to access the camera of the device. To achieve this two different possibilities are provided. The easier way is via an intent that redirects the user to the camera application. For a possibility to integrate the camera functionality directly into the application and gain access to enhanced functionality the SDK methods can be used.

Intent An intent[23] is a message that can be broadcasted across the whole Android system. Other applications can listen for them and therefore propagate provided features and services to other application. To create still images with the camera the intent "ACTION_IMAGE_CAPTURE" is used. If an application starts this intent the camera

application is opened in the foreground, where the user can interact and shoot the image. After the user is finished the control is given back to the calling application, which is able to query for the created image file. This a very simple way to integrate camera functionality, but the camera is solely controlled by the camera application. Therefore the calling application has little control over the camera settings and parameters. Additionally because the user is redirected to a different application, no integration for a smooth user experience is possible.

SDK More control and flexibility is possible by directly using the camera specific API provided by the SDK (called camera2 API). This requires the application itself to show a user interface as well as start and handle the image capture process. The following steps outline the parts that need to be implemented:[22]

1. Detect Camera(s) and Features: The first step is to query the available cameras of the device (e.g. front facing and back facing) and determine the desired target camera. All available features and configuration parameters, of a specific camera, can be determined. This includes the possible values for exposure time, possible ISO values and available aperture values.
2. Preview: A preview session can route information from the camera to a predefined surface area of the application's window. This allows the user to see a preview of the image and all changes to the configuration can be seen immediately. Furthermore it is possible to add additional overlays and controls to allow the user to adjust the resulting image. Typical features include rectangles to crop the image and controls for manually adjusting focus and exposure time.
3. Capture Listener: The capture listener gets notified on every capture request sent to the camera. This listener needs to be attached before an actual capture request is triggered. The capture request is usually triggered by pressing a button or touching the preview area in the user interface.
4. Capture Image: The necessary logic in the listener that receives the byte stream from the camera for further processing. Usually the result is saved into an image file on the device.
5. Release Camera: As soon as the camera is not needed anymore it has to be released or else other applications cannot use it.

3.5.2 Tesseract OCR

The Tesseract[62] project does not provide a version of their OCR engine for Android. Fortunately two projects exists that provide an Android library. Tesseract-Android-Tools[65] and Tess-Two[63], the second being a fork which incorporates some enhancements. Because Tess-Two can be easier integrated into the project, it is used. It provides the necessary Android specific build files to compile the Tesseract source code into a library usable in an Android application.

Create Library

The library needs to be built and integrated into the Android application project. Then it is packaged into the application automatically and can be used from therein.

1. Retrieve source: Download the latest version of the library and extract it into the "Library" folder of the target Android application.

   Listing 3.1: Retrieve Tess-Two

   ```
   # Download latest release from https://github.com/rmtheis/
      tess-two/releases
   wget https://github.com/rmtheis/tess-two/archive/5.2.0.zip

   # Extract the archive
   unzip 5.2.0.zip
   ```

2. Build library: The library itself can be built with the tools provided by the Android SDK and NDK. The SDK and NDK tools have to be in the path of the current shell, else the commands "ndk-build" and "android" do not exist. With the "android update" command the target of the library project can be adjusted to match the application. This is necessary, else a a target version mismatch error can happen. The "ndk-build" command compiles the library for all CPU architectures the library is available.

   Listing 3.2: Build Tess-Two

   ```
   cd tess-two-5.2.0/tess-two
   android update project --path . --target android-21
   ndk-build
   ```

3. Integrate into application: After a successful build the library can be integrated into the application. After that the library is built automatically by the Android build system while building the application. Additionally it copies the library into the build result and packages it into the application package (apk). Therefore the library is shipped with the application and does not have to be installed manually on the Android device. This is achieved by adding the following information into the configuration file of the build system (build.gradle):

Listing 3.3: Integrate Tess-Two into the build

```
dependencies {
    compile project(':libraries:tess-two')
}
```

Use Library

The Tesseract library is written in C and compiles as a native library. Therefore it cannot be used directly from the Java code, it has to be wrapped with JNI[46]. Fortunately the JNI wrapper is provided alongside the library and is compiled automatically, so it can be used from the Java code of the application without any further ado.

Listing 3.4 shows how the library can be used from the Java code of the application. Important is the initialization line, that defines the detection language and the folder location of the training data. Training data is always required for Tesseract, else it cannot operate. It is provided by the Tesseract project for a wide variety of different languages. This data defines valid characters, their shape and how the can be detected. Additionally the recognition engine is set (OEM_TESSERACT_CUBE_COMBINED), which means Tesseract should determine the best available engine automatically. The page mode is set to single line, which defines that the input image only consists on a single line of text.

Listing 3.4: Using Tess-Two inside the application

```
TessBaseAPI tessBaseAPI = new TessBaseAPI();
# Initialize libary: Folder where the language data is stored,
   language, mode
tessBaseAPI.init("/sdcard/Download", "eng", TessBaseAPI.
   OEM_TESSERACT_CUBE_COMBINED);
# Configure that the input image only contains a single text
   line
```

```
tessBaseAPI.setPageSegMode(TessBaseAPI.PageSegMode.
   PSM_SINGLE_LINE);
# Set image for text recognition
tessBaseAPI.setImage(bmp);
# Run OCR and receive result
String result = tessBaseAPI.getUTF8Text();
# Close the library
tessBaseAPI.end();
```

3.5.3 ImageMagick

Using ImageMagick on an Android device is possible by using the library provided by Android-ImageMagick[6]. This project provides the source code and the necessary Android build files to create the library for different architectures. Unfortunately it does not provide access to the "convert" command from ImageMagick[34], therefore the build files needs to be changed to build a library with support for this command.

Create Library

To be able to use ImageMagick in the Android application project, it needs to be built and integrated.

1. Retrieve source: The project does not provide releases, therefore the most current development state is used. The archive needs to be extracted in the "Library" folder of the Android application.

   Listing 3.5: Retrieve Android-ImageMagick

   ```
   # Download current version from https://github.com/
      paulasiimwe/Android-ImageMagick
   wget https://github.com/paulasiimwe/Android-ImageMagick/
      archive/master.zip

   # Extract the archive
   unzip master.zip
   ```

2. Modify build file to include the "convert" command: Build information is located in the file "Android.mk". It is specific to the Android build system, but the syntax is

closely related to GNU make. To build an additional library, that contains the convert command, the following entry is necessary.

Listing 3.6: Buildfile for Android-ImageMagick

```
include $(CLEAR_VARS)

# Name of the module (libconvert)
LOCAL_MODULE := convert

# Necessary header files
LOCAL_C_INCLUDES  :=  \
 $(IMAGE_MAGICK) \
 $(IMAGE_MAGICK)magick \
 $(IMAGE_MAGICK)wand \
 ${PNG_SRC_PATH} \
 $(FREETYPE_SRC_PATH)include \
 $(FREETYPE_SRC_PATH)src \
 ${TIFF_SRC_PATH}libtiff/ \
 ${JPEG_SRC_PATH} \
 ${PHYSFS_SRC_PATH} \
 $(WEBP_SRC_PATH)src \
 ${JASPER_SRC_PATH}src/libjasper/include \

# The actual source compiled into the module
LOCAL_SRC_FILES := \
 $(IMAGE_MAGICK)wand/convert.c \
 $(IMAGE_MAGICK)wand/mogrify.c

# Dependencies
LOCAL_STATIC_LIBRARIES := \
    libpngo \
    libjpego \
    libwebpo \
    libfreetype \
    libtiff \
    libphysfs \
    libjasper \
    libmagick \
```

```
include $(BUILD_SHARED_LIBRARY)
```

3. Build library: Analogous to the build process of Tesseract library, the "android update" command is used to match the target version of the library with the one of the application. The "ndk-build" command compiles the source and creates the library files.

   Listing 3.7: Build Android-ImageMagick

   ```
   cd Android-ImageMagick-master]
   android update project --path . --target android-21
   ndk-build
   ```

4. Integrate into application: To include the libraries into the build process of the application, it is necessary to add it to the configuration file of the build system (build.gradle). The following entry needs to be added:

   Listing 3.8: Integrate into application

   ```
   dependencies {
       compile project(':libraries:Android-ImageMagick')
   }
   ```

Use Library

The newly created library (libconvert) can be loaded and used from the application's native code. All exposed methods defined in the header files can be consumed. In particular the command "ConvertImageCommand" is relevant. The following listing shows a sample usage. It takes an input file, runs the ConvertImageCommand (with the repage option) and produces an output file. The method uses JNI[46], so the command can be called from the Java code of the application.

Listing 3.9: Use Android-ImageMagick in native code

```
#include <jni.h>

#include <magick/api.h>
#include <wand/convert.h>
```

```
jint Java_me_praher_ocr_OCRTask_processImage( JNIEnv* env,
 jobject thiz, jstring inputName, jstring outputName)
{
  // Input and output files
  const char* input = (*env)->GetStringUTFChars( env, inputName
     , NULL ) ;
  const char* output = (*env)->GetStringUTFChars( env,
     outputName , NULL ) ;

  // Return information from ImageMagick
  ImageInfo *imageInfo = AcquireImageInfo();
  ExceptionInfo *exceptionInfo = AcquireExceptionInfo();

  char *command[] = { "convert", "(", input, ")", "+repage",
     output, NULL };
  ConvertImageCommand(imageInfo, 6, command, NULL, exceptionInfo
     );

  return 0;
}
```

It is necessary to create a build file to compile the native code. The build file is shown in the listing below. It takes the source file (imageprocess.c) and creates a library (chid) out of it. It is necessary to specify the location to the ImageMagick header files (LOCAL_C_INCLUDES) and the ImageMagick library (LOCAL_LDLIBS).

Listing 3.10: Buildfile for the native code

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)

LOCAL_MODULE    := chid
LOCAL_SRC_FILES := imageprocess.c

LOCAL_C_INCLUDES := ../../../../libraries/Android-ImageMagick/
   jni/ImageMagick-6.7.3-0
LOCAL_LDLIBS := ../../../../libraries/Android-ImageMagick/libs/
   armeabi-v7a/libconvert.so
```

```
include $(BUILD_SHARED_LIBRARY)
```

In the Java code the library (chid) can be loaded. This is usually done in a static code block. The existing native methods have to be defined as stubs with the native keyword. After that those methods can be called from anywhere in the Java code like regular Java methods.

Listing 3.11: Use the method defined in native code

```
// Define the native method
public static native int processImage(String input, String
   output);
// Load the native library
static {
 System.loadLibrary("chid");
}
```

4 Password Manager

4.1 Introduction

Despite the existence of authentication schemes that are more secure, authentication based on a combination of username and password is still prevalent. It is often the default and sometimes the only option a service provider offers. Unfortunately the use of passwords for authentication comes with multiple difficulties:[41]

- Entropy: Password strength is the most crucial factor that defines the security of password based authentication. A password should be resistant against dictionary- and brute-force attacks. That means it should not be based (solely) on existing words, have a sizable length and contain different character classes (uppercase letters, lowercase letters, digits, special characters). Yet passwords are often chosen with low entropy as to make them easier to remember and faster to type.
- Memorizing Passwords: Passwords with a high entropy are harder to remember, especially when a different password is used for every service/account.
- Phishing: Authentication with a combination of username and password is prone to phishing attacks. If the user enters the login information to a phishing site, an attacker can use this information directly to access the account of the user.

A password manager provides functionality to enhance the usability and security of authentication methods which utilize passwords. It stores login information, usually a combination of username and password, but possibly also other information like a credit card number, in a location controlled by the user (local device, cloud storage). If needed, the user can retrieve stored information and therefore does not have to remember it.[41]

This can reduce the cognitive load on the user, reduce log-in time and can therefore promote better password quality and eliminate password reuse. To protect the stored login information most password managers encrypt the data with some form of master password. This mitigates the consequences in case the data gets stolen. Additional features of password

managers usually include generators for strong passwords, backup and synchronization across devices. Password managers are also able to detect some phishing attacks. This is done by checking if the URL and source, of the site the user wants to insert credential information, matches the expected values.[41]

4.2 Password Manager for Android

A lot of different password managers exist for the Android platform. They all have a different feature set and provide different approaches, how to transfer data from the password manager to the target application. To chose a suitable password manager for the solution, the following criteria where defined as essential for the preselection process.

- License: The password manager has to be released under an open source compatible license. Being open source allows for independent security audits. This can help to ensure that there are no hidden backdoors included and cryptographic best practices are implemented.[57] Additionally using an open source password manager allows to extend and adapt it to specific needs.

- KeePass compatibility: For enhanced compatibility with password managers on different platforms (also on desktop systems) the KeePass database format should be supported. Preferably it should support KeePass database (KDB) version 1 as well as version 2, the latter being essential. Supporting both KDB versions ensures maximum compatibility with KeePass compatible password managers.

- Integration: It should be able to integrate the password manager into the actual application. That means there should be a way for the application to query login information, or at least to initiate this process. There a three different possibilities, how a password manager can support third party integration.

  Plugin Applications can provide a plugin infrastructure. With this other applications can request permissions to use all or only some of the provided functionality. The user has to grant the access request. Thereafter the application can access the stored data via provided access methods.

  GUI A password manager can allow an application to directly jump to a specified GUI activity. Therefore the user does not have to manually switch applications. The activity can also return data to the calling application.

Content Provider A content provider is a concept that enables applications to provide a interface to access data. It allows to exchange information between applications running in different processes. Basically a content provider encapsulates data and provides methods to ensure data security.[21]

The plugin option allows a very tight integration and to only request the least amount of permissions. Both plugin and content provider can show a custom GUI if necessary (e.g. to unlock the database). The GUI options does not allow such a transparent integration, the user always experiences a noticeable application switch.

With the defined criteria three different password managers exist, namely Keepass2Android[12], KeePassDroid[49] and KeepShare for KeePass[43]. Table 4.1 shows a comparison of some basic criteria. From the three options Keepass2Android got chosen for integration, because it supports all requirements. Furthermore it allows a very clean and unobtrusive integration. Due to the plugin architecture the user knows exactly what the application is allowed to do and can revoke the access if desired. Compared to the other two options it has the best documentation, especially on how it can be integrated.

	Keepass2Android	**KeePassDroid**	**KeepShare for KeePass**
License	GNU GPL v2	GNU GPL v2/v3	GNU GPL v2
KDB v1 Support	Yes	Yes	No
KDB v2 Support	Yes	Yes	Yes
Master Password Support	Yes	Yes	Yes
Integration	Plugin	GUI	Content Provider
Documentation	Good	None	None

Table 4.1: Android password manager

4.3 Integration of Keepass2Android

4.3.1 Security Considerations

A password manager is expected to protect the stored information and prevent any theft or misuse of the information. Yet the user needs to access the information and use it in another application (e.g. the web browser). Due to security and convenience it is not desirable that the information is shown to the user, memorized and then entered wherever needed manually. Therefore a password manager needs to be integrated into a system for best functioning. The different possibilities of integration can be classified into groups:

Copy/Paste The easiest way to transfer data from the password manager to another application is to use the clipboard of the system. A single data element (e.g. username, password) is copied and the user has to paste it in the correct target. This works with basically all applications without any modification or adjustment. Unfortunately this solutions has severe drawbacks. First it is very inconvenient, because the user has to constantly switch between applications and copy/paste the desired information. Second the password manager is not able to provide any security checks, like if a site is secure and if the user chose the correct information for the site. And last it is vulnerable[15], because other applications can monitor the clipboard and therefore steal the information.

Plug-In For popular applications (typically web browsers) password managers often provide plugins. They offer tight integration with the specific application. Therefore convenience and security can be enhanced. It is possible to automatically fill forms with the correct data for the site, check if the connection is secure. This approach does not use the system clipboard for data transfer.

Software Keyboard Password manager, especially those designed for touch based devices, have another interesting option. On those devices the keyboard is usually not physically available, but shown on the screen (software keyboard). These systems often allow third party applications to install alternate software keyboards. So a password manager can install an additional keyboard, where the user can select data from the password manager that should get typed. Compared to the copy/paste model this eliminates the necessity to utilize the system clipboard. Otherwise it possesses the same advantages and disadvantages. It works with every application, but cannot provide any security checks.

Service Interface A service interface allows other applications to gain access to the information from the password database, by providing well-defined access methods. This makes it possible for third-party developers to integrate the password manager into their product. It is crucial that the user has full control to allow/deny specific applications to access the stored information. For maximum security it is also best to let the user confirm every transfer of data beforehand. So the risk of rogue applications, trying to steal information, can be minimized. Of course it is always necessary to trust the target application with the transferred data.

For accessing data from different applications Keepass2Android offers a software keyboard and a service interface. The service interface allow a very tight integration into the application

and provides good security.

4.3.2 Request Permission

To protect the password database Keepass2Android only allows access, if the user explicitly gives the application permission for that. This is done by the application listening to predefined broadcast messages sent by Keepass2Android. To do so, it is necessary to define it in the "AndroidManifest.xml" file as shown in listing 4.1. This also specifies the class that handles those requests (PluginAAccessReceiver). There the actually required permissions are declared (listing 4.2). Only the minimum set of necessary permissions should be requested. The overall result of this can be seen in figure 4.1. This dialog is shown to the user the first time the application wants to access Keepass2Android. It shows the application requesting access and the defined permissions. The user has to accept this, else the application cannot interact with the password manager.

Listing 4.1: Keepass2Android - Intent

```
<receiver a:name=".PluginAAccessReceiver" a:exported="true">
 <intent-filter>
  <action android:name="k2a.ACTION_TRIGGER_REQUEST_ACCESS" />
  <action android:name="k2a.ACTION_RECEIVE_ACCESS" />
  <action android:name="k2a.ACTION_REVOKE_ACCESS" />
 </intent-filter>
</receiver>
```

Listing 4.2: Keepass2Android - Permissions

```
public class PluginAAccessReceiver extends
   PluginAccessBroadcastReceiver
{
 @Override
 public ArrayList<String> getScopes() {
  ArrayList<String> scopes = new ArrayList<String>();
  scopes.add(Strings.SCOPE_DATABASE_ACTIONS);
  scopes.add(Strings.SCOPE_CURRENT_ENTRY);
  scopes.add(Strings.SCOPE_QUERY_CREDENTIALS);
  scopes.add(Strings.SCOPE_QUERY_CREDENTIALS_FOR_OWN_PACKAGE);
  return scopes;
 }
```

```
}
```

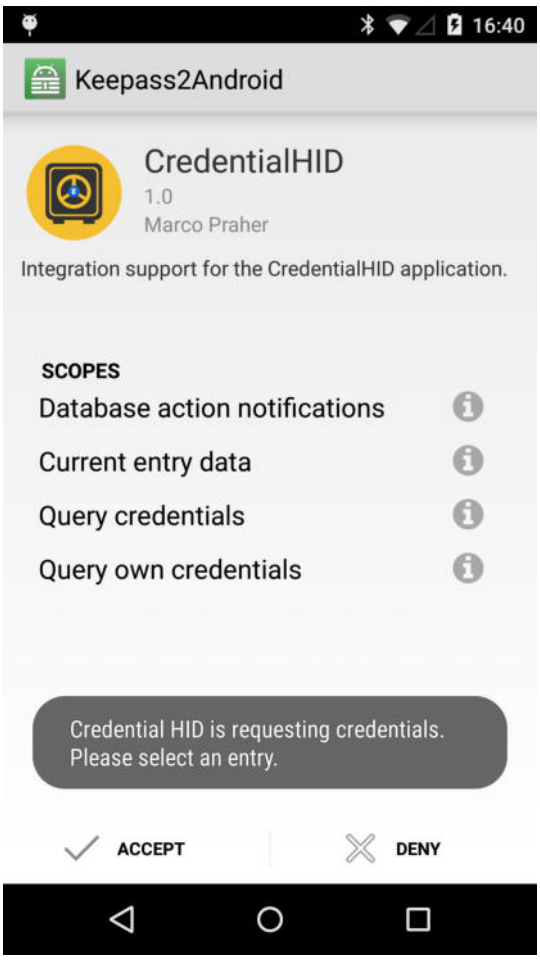


Figure 4.1: Keepass2Android - Request access

4.3.3 Unlock Database

The password database is protected by a master password, that needs to be entered by the user before any information can be accessed. After unlocking the database stays unlocked for a configurable amount of time. Leaving the database open for long times is convenient, but decreases security. Keepass2Android offers a feature called "Quick Unlock", where the user enters the complete password only once per session and on successive calls only the last few characters.

Unlock of the password database is completely transparent for the application integrating Keepass2Android. The unlock screen (figure 4.2) is shown to the user automatically on any request if the database is locked.

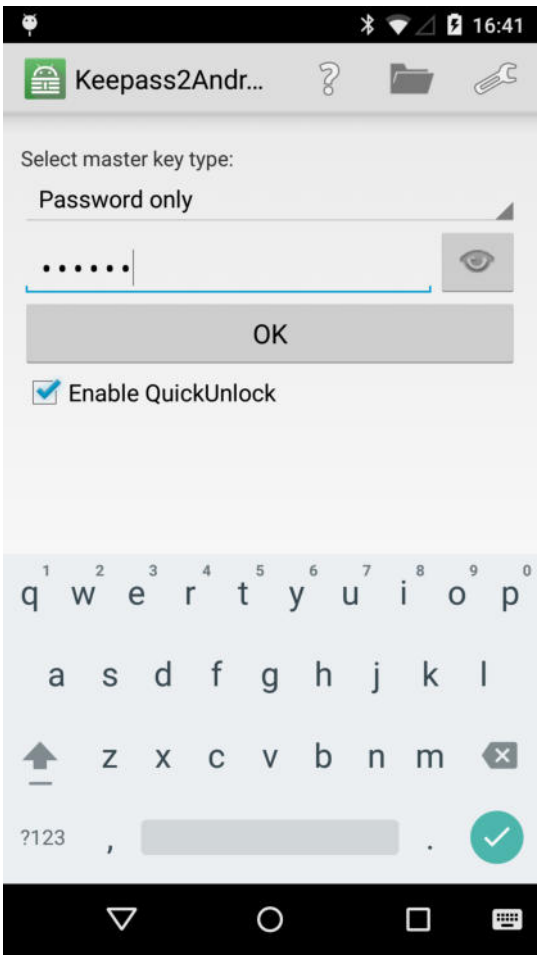


Figure 4.2: Keepass2Android - Unlock database

4.3.4 Query Entry

To query the password database it is necessary to start a single activity (listing 4.3). For Android applications an activity is the basic form for user interaction and is started using the current context (e.g. another activity/window). To search inside the database Keepass2Android offers a "QueryIntent" where a search term can be specified. If a single matching result is found, it is immediately returned (figure 4.3 right), if multiple possible results exist the user has to select the correct entry (figure 4.3 left). If an activity is finished a predefined method (onActivityResult) is called, where the result can be extracted (listing 4.4).

Listing 4.3: Keepass2Android - Search login data

```
context.startActivityForResult(
 Kp2aControl.getQueryEntryIntent(searchTerm),
 1
```

```
);
```

Listing 4.4: Keepass2Android - Retrieve result

```
protected void onActivityResult(int request, int result, Intent
    data) {
 HashMap<String, String> credentials = Kp2aControl.
     getEntryFieldsFromIntent(data);
 credentials.get(KeepassDefs.UserNameField);
 credentials.get(KeepassDefs.PasswordField);
 credentials.get(KeepassDefs.TitleField);
 credentials.get(KeepassDefs.UrlField);
}
```

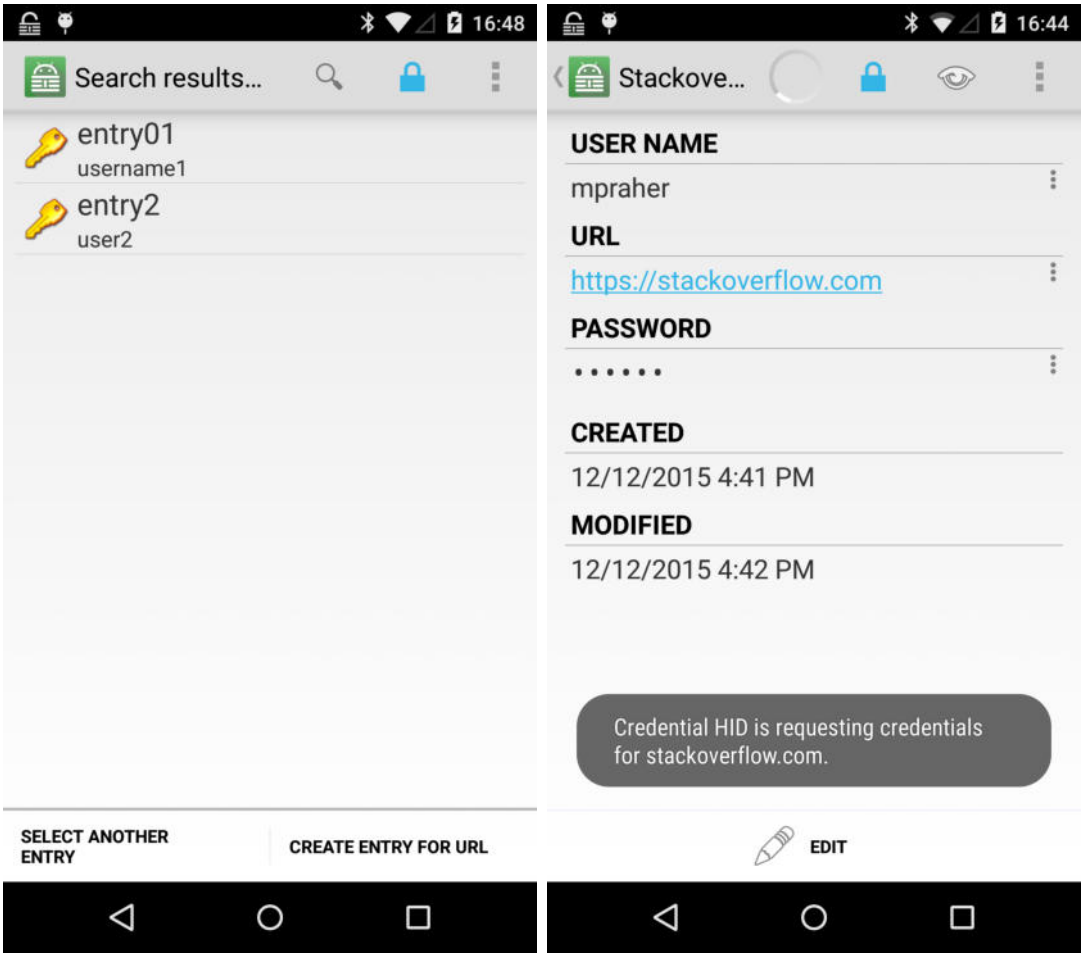


Figure 4.3: Keepass2Android - Search result

5 Credential HID

5.1 Architecture

Figure 5.1 shows an architectural overview of the implemented solution and a generic target device. The boxes outlined red represent parts that where added or changed as part of the implementation. It can be seen that no specific changes on the target device are necessary. The solution is implemented solely on the Android device. Interaction with the target device is based on standard Bluetooth functionality, usually available on every PC-like device.

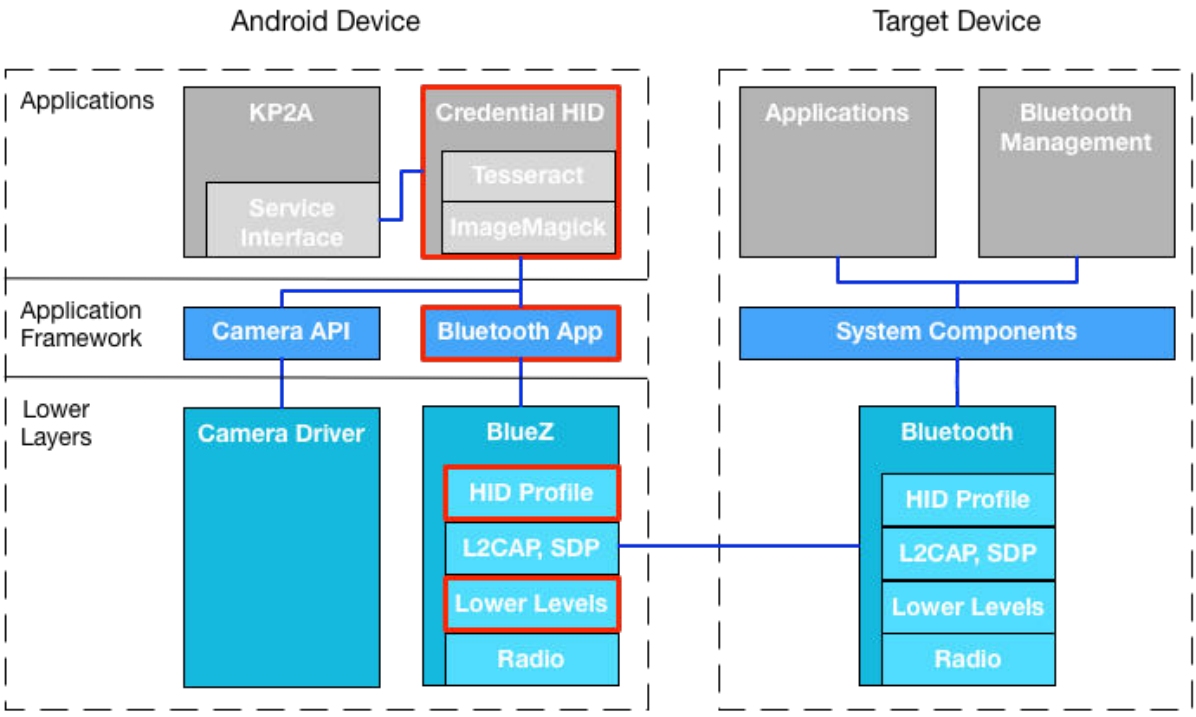


Figure 5.1: Credential HID - Architecture

The listing below describes the components that were created or adapted as part of the solution and how they interact with each other.

Credential HID The main application that controls the created functionality. It provides the graphical interface for the user to initiate all necessary actions. Within the application the required libraries for OCR are embedded: ImageMagick, for image processing and enhancement, and Tesseract for the actual OCR process. The application also communicates directly with the password manager Keepass2Android (KP2A) via the server interface it exposes. It also interacts with the Bluetooth application of the system, that allows other applications to control the Bluetooth functionality. With this it is possible to enable Bluetooth, set the Bluetooth discoverability and control the HID device profile.

Bluetooth Application The Bluetooth application is responsible for managing and controlling the Bluetooth stack. It provides a high level interface (API) for other applications to consume Bluetooth functionality. Additionally it interacts directly with the user to handle common Bluetooth tasks like starting/stoping Bluetooth, pairing and connecting other Bluetooth devices. It is necessary to extend the API to allow handling of functionality specific to the newly created HID device profile.

HID Profile The HID profile, to be more specific the HID device profile, represents the actual extension of the Bluetooth stack that adds new functionality. It extends BlueZ (the Bluetooth stack) by adding the logic necessary to provide the HID device profile. Existing functionality is utilized wherever possible (e.g. lower level protocols for actual communication, management of SDP entries). The pairing and bonding is managed entirely by existing functionality. In contrast the HID device profile is responsible for accepting and managing incoming connections from a target device, which acts as the HID host. Additionally HID device profile specific functions and notifications, that can be consumed by the Bluetooth application, are provided.

Lower Levels The Bluetooth architecture is divided into multiple layers. Therefore it is usually not necessary to modify lower levels in order to implement a new profile. However for this solution it was necessary to add the possibility to modify specific configuration values which are usually static. The values in question describe the Bluetooth class of device (CoD). Those are per default hardcoded to phone/smartphone, which must be changed to peripheral/keyboard while the HID device profile is active.

5.2 Interaction Sequences

5.2.1 Bluetooth

The sequence shown in figure 5.2 describes the interaction flow when using the Bluetooth HID device profile. It provides an overview of the communication between the Credential HID application, the Android Bluetooth application, the HID device profile inside BlueZ and the existing BlueZ core functionality. The messages can be grouped into five main procedures:

- Activate Bluetooth HID device mode (1-6): The sequence is initiated by the user from within the Credential HID application. By enabling the HID device mode the application sends the corresponding start command to the Bluetooth system application, which in turn calls the init and start command of the HID device profile. The init command registers the profile within BlueZ. The start command actually starts the HID device profile by adding the SDP entry and starting to accept incoming connections from a HID host. The Bluetooth options specify the desired class of device (CoD) and if non corresponding SDP entries should be removed. This allows to more accurately simulate a (hardware) Bluetooth keyboard.

- Set Bluetooth discoverability (7): The Bluetooth system application allows other applications to set the Bluetooth discoverability of the device. This is necessary if the user wants to connect a new Bluetooth device, e.g. if a HID host should be able to connect for the first time.

- HID host connected notification (8-9): A notification is sent by the HID device profile to the Bluetooth application every time a HID host connects or disconnects. The notification also includes the Bluetooth address of the host. With this information the Bluetooth application creates a system wide broadcast message that can be received by other applications.

- Send keys (10-11): After a HID host has connected it is possible to send characters to it. This can be done by creating key press events that consist of keys and modifier keys. This can be done multiple times, as long as the HID host and HID device have an active connection.

- Deactivate Bluetooth HID device mode (12-15): When the Bluetooth HID device profile is no longer needed it can be completely deactivated. Doing so disconnects the current HID host (if connected) and stops listening for new connections. Additionally the SDP entry is removed and the profile is unregistered from BlueZ.

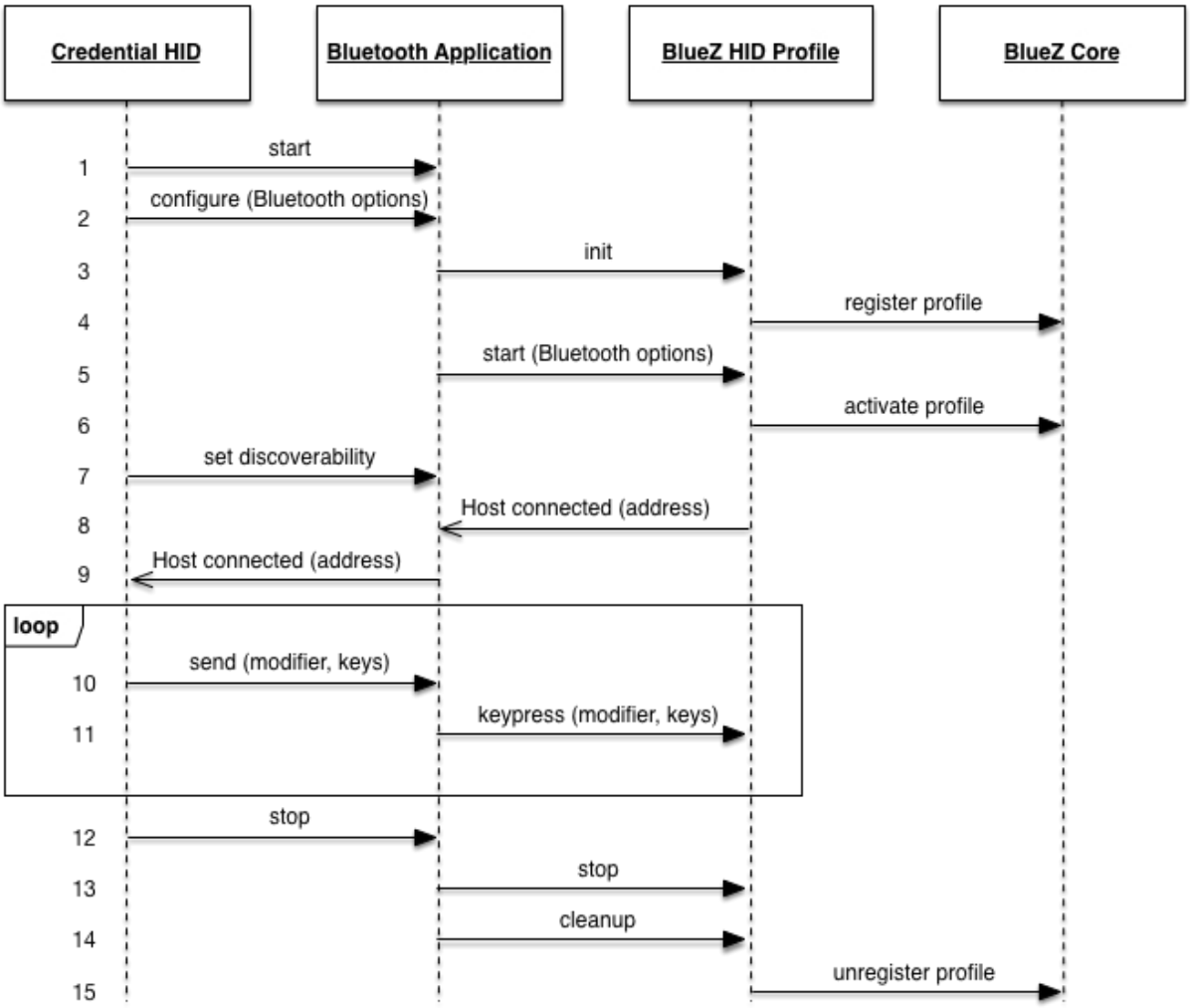


Figure 5.2: Credential HID - Bluetooth sequence

5.2.2 Optical Character Recognition

For the OCR process the Credential HID application has to interact with ImageMagick and Tesseract. These interactions can be seen in figure 5.3, which can be grouped into three steps:

- Enhance image (1-2): The ImageMagick build used in this solution contains the convert command. It allows to specify the desired transformation options similar to the CLI version of ImageMagick. Additionally the full path to the source image file is necessary. The library does not override the source image, instead it will create a new image file with the transformations applied.

- Initialize Tesseract (3-4): Tesseract needs to be initialized prior to use. It is necessary to specify training data for the language that should be recognized. Furthermore it is possible to specify a mode for layout analysis. Setting the mode (e.g. single line of text, single column of text, single word) that best fits the source image can improve the OCR result. For the Credential HID application the single block of text mode is used.
- Optical character recognition (5-6): The actual text recognition is initiated by submitting the bitmap image to the library, which in turn returns the recognized text in an UTF-8 encoding.

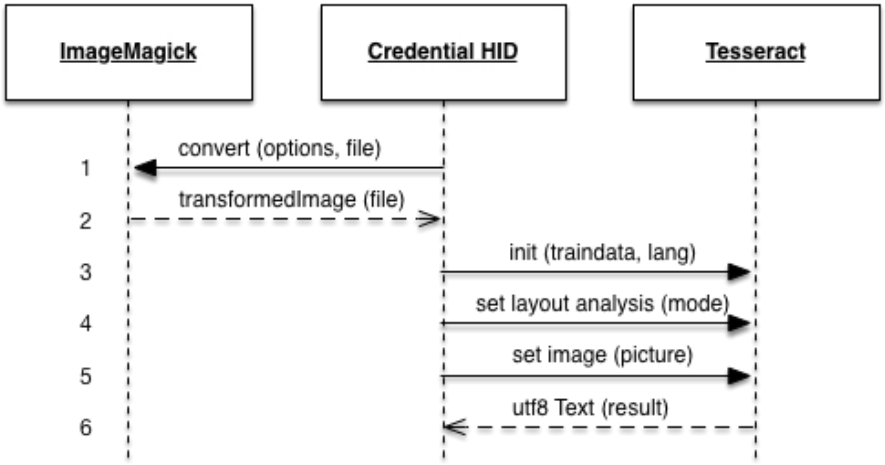


Figure 5.3: Credential HID - OCR sequence

5.2.3 Password Management

For managing user credentials the only interaction partner of the Credential HID application is Keepass2Android (KP2A), the password manager. Figure 5.4 shows the communication sequence for the two different workflows.

- Query for existing account (1-2): Searches for matching accounts in the password database by the supplied search term. Returns exactly one account or nothing. If multiple matching accounts were found the user has to select the correct one inside KP2A. An account consists of a username, the password, an URL and a title for the entry.

- Add a new account (3-4): Adds a new account to the password database. An URL field is provided for the new account. All other data has to be entered manually by the user directly into KP2A.

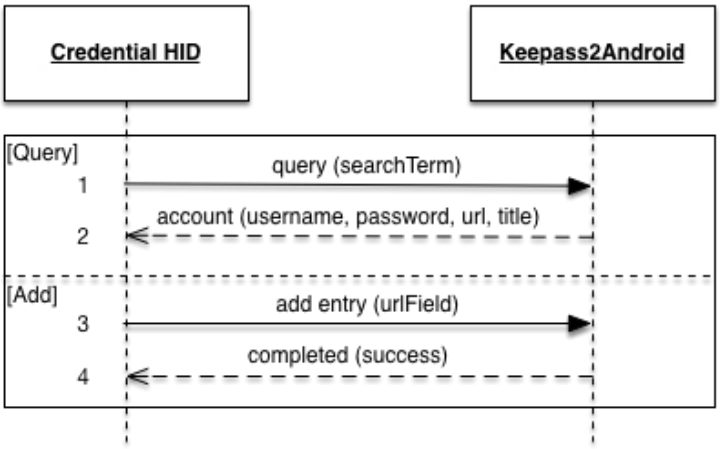


Figure 5.4: Credential HID - Password management sequence

6 User Guide

6.1 Installation

The Credential HID application can be installed and used only on supported Android devices. A device has to fulfill all of the criteria listed below to be able to successfully run the application:

- Support for Android 5.0 (Lollipop)
- Android system compiled with BlueZ
- Keepass2Android installed (e.g. via the Google Play Store)
- Root access ("rooted" device)

To install the application and use the full functionality it is necessary to install the Credential HID application, replace BlueZ and the Bluetooth system application. For a more detailed guide how to replace Bluetooth stack on an Android device consult chapter 2.4.

BlueZ To replace BlueZ it is necessary to stop Bluetooth first. This can be done via Android's Bluetooth settings dialog. Afterwards two files need to be replaced. The file "bluetooth.default.so" needs to be copied to "/system/lib/hw" and the file "bluetoothd-main" to "/system/bin".

Bluetooth application To replace the Bluetooth system application the file "libbluetooth_jni.so" has to be replaced in the folder "/system/lib". Additional the "Bluetooth.apk" file needs to be copied to "/system/app/Bluetooth" and installed by opening it (e.g. from a file browser).

Credential HID All libraries and components of the main application are packaged into a single Android application package (APK) file. It is not available via the Google Play application store. Therefore it is necessary to manually transfer the file "Credential-HID.apk" to the device and install it by opening it. After the installation it can be started via the icon seen in figure 6.1.

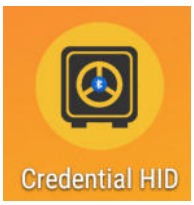


Figure 6.1: Credential HID - Icon

Opening the application shows the main screen from where all operations are initiated and controlled. Figure 6.2 shows an annotated screenshot of the main screen, segmenting it into the three main parts (A, B and C) and the settings (S). Section A controls and monitors the Bluetooth connectivity. Section B interacts with the password manager and allows using OCR technology to grab an URL from a computer screen. Section C is used to transfer the account information, via a previously established Bluetooth HID connection, to another device. The gear symbol at the bottom (S) can be used to open the settings dialog.

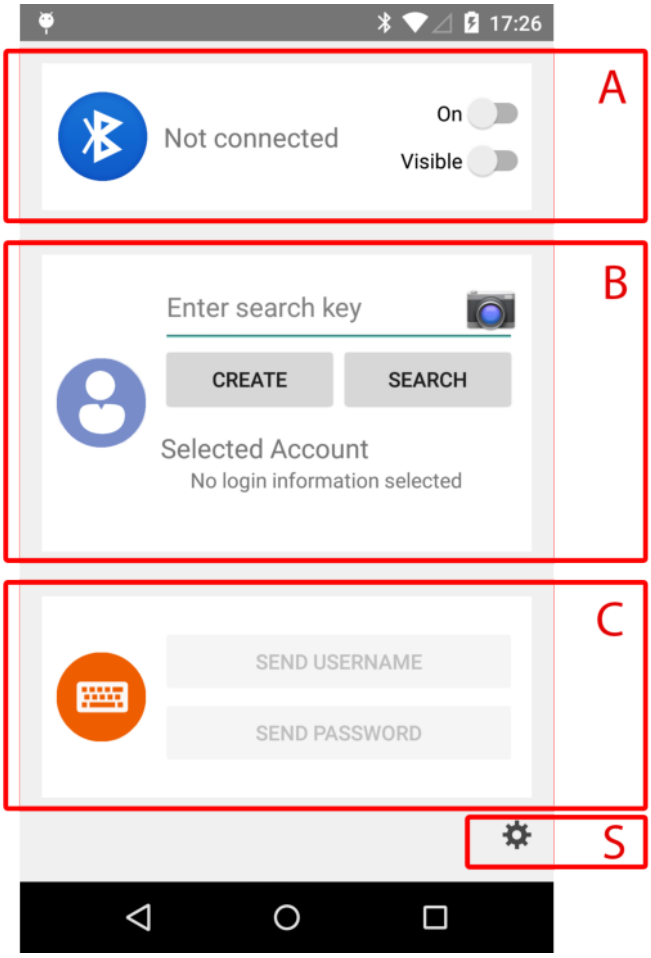


Figure 6.2: Credential HID - Main screen

6.2 Settings

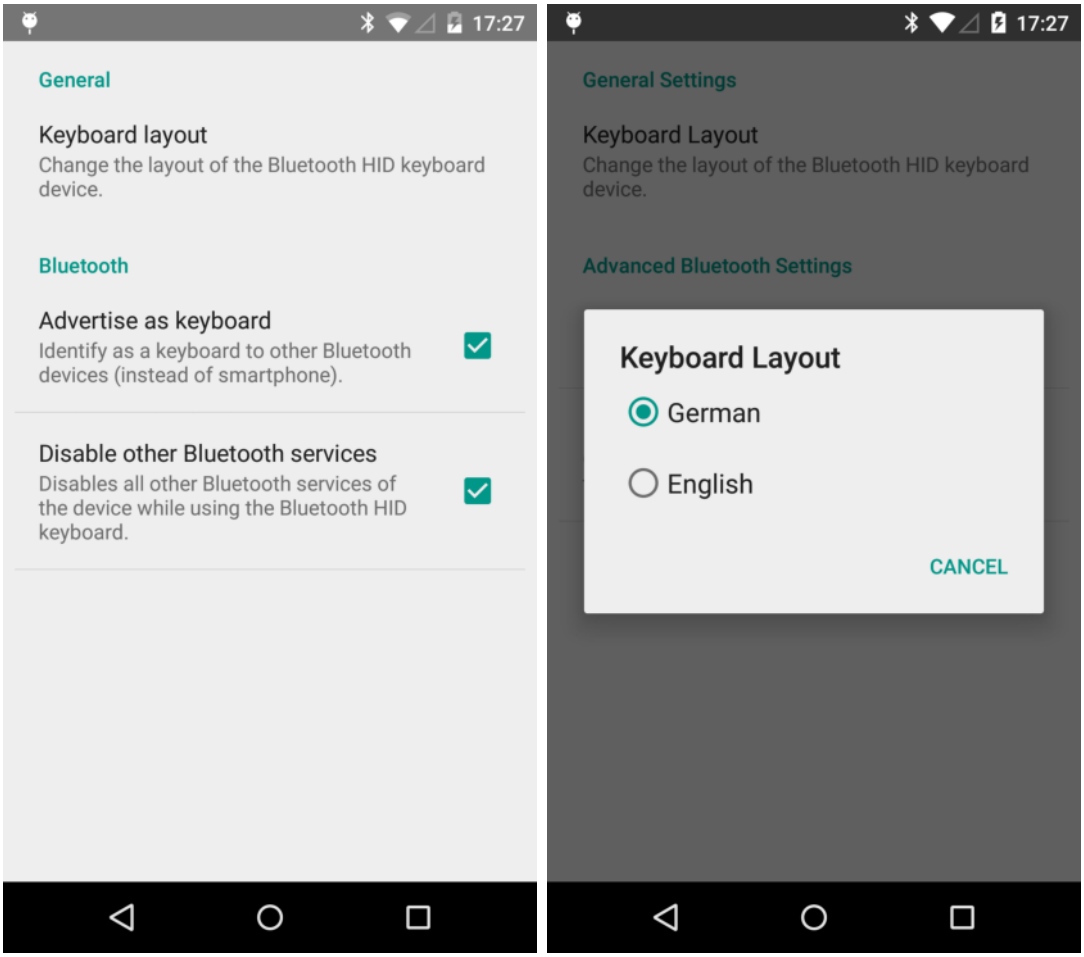


Figure 6.3: Credential HID - Settings

The settings dialog can be used to customize Bluetooth configuration if necessary. The following options are available:

Keyboard Layout The application can simulate different keyboard layouts. For correct functioning it must use the same option as the target device.

Advertise as keyboard This setting changes the Bluetooth device class to keyboard, prior to starting the Bluetooth HID functionality. This is necessary for some HID hosts to be able to discover and pair the device at all.

Disable other Bluetooth services This setting removes all other SDP entries, prior to starting the Bluetooth HID functionality. Therefore no other Bluetooth services can be

used while the Bluetooth HID functionality is activated. This increases compatibility with different Bluetooth HID hosts.

6.3 Bluetooth Connection

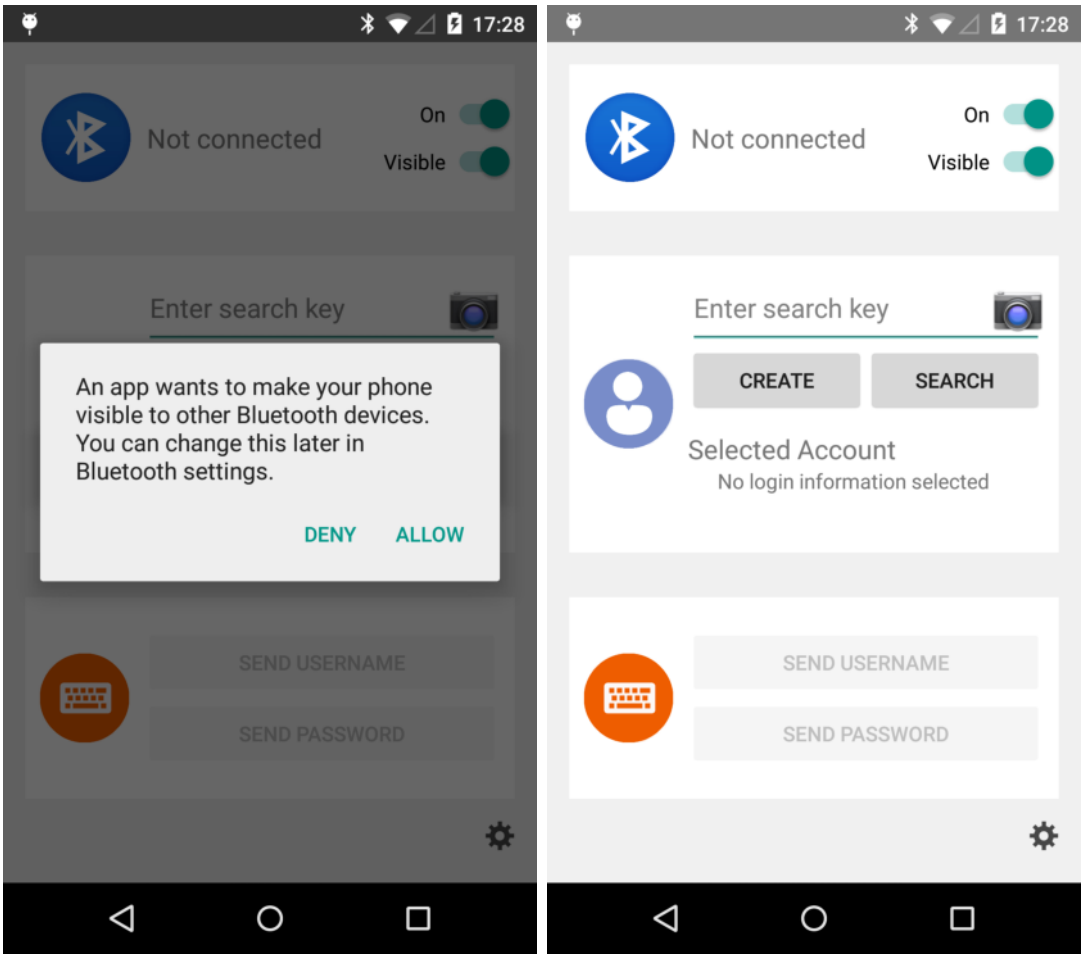


Figure 6.4: Credential HID - Activate Bluetooth

The Bluetooth connection is controlled via section A of the main screen (figure 6.2). The "On" switch activates the Bluetooth HID functionality. After that already paired devices can connect. If a new device should be able to pair, it is necessary to set the visibility to on. This is done via the switch "Visible". A message (seen in figure 6.4) is shown by the system asking for confirmation.

A pairing request has to be initiated by the HID host. Both devices will show a paring code consisting of six digits (figure 6.5). If the numbers match on both devices the pairing can be confirmed, otherwise it has to be canceled.

Pairing is only necessary once per HID host. After that the HID host can directly initiate the connection. No further user input is necessary. If a connection could be established the connected device is show by it's Bluetooth address (figure 6.5).

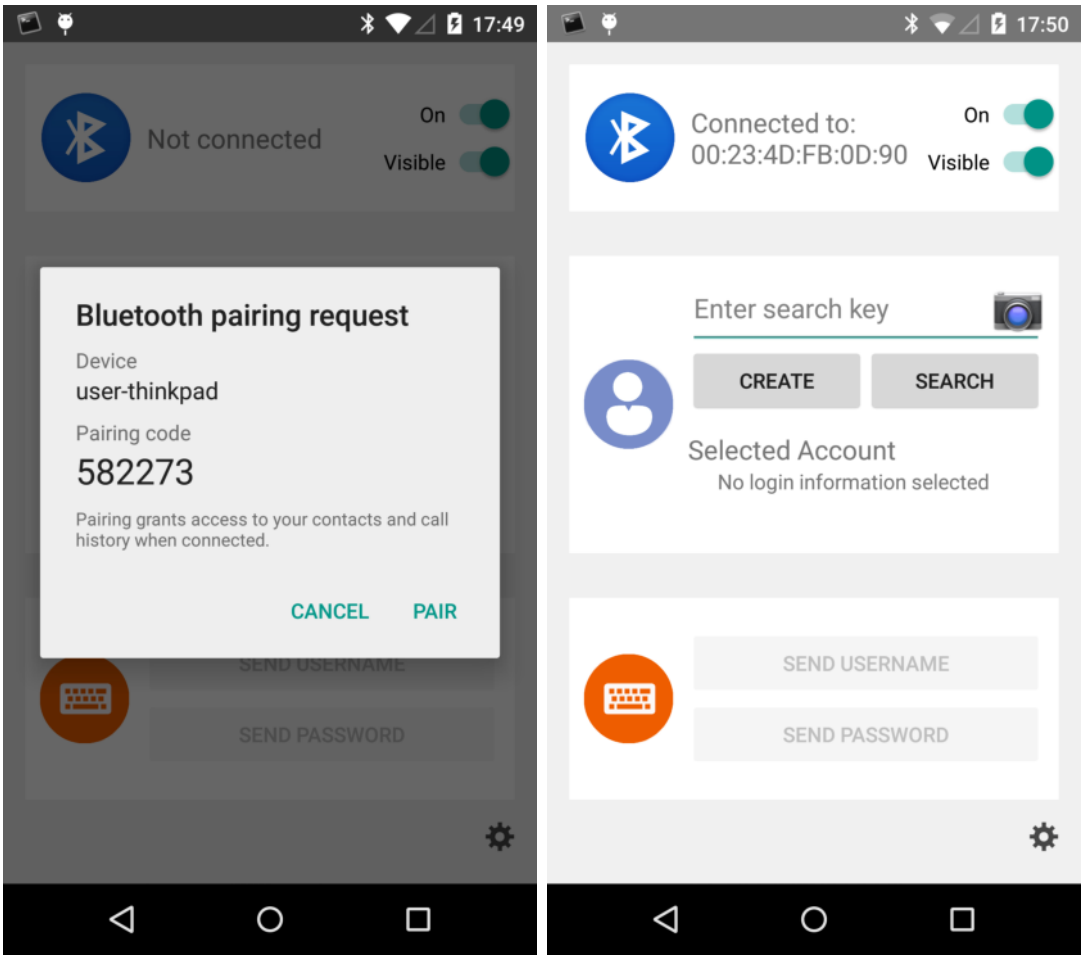


Figure 6.5: Credential HID - Bluetooth pairing

6.4 Enter URL (OCR)

The top of section B (figure 6.2) is used to input an URL, that is used for interacting with the password manager. An URL or keyword can be entered directly by clicking inside the text field labeled "Enter search key" or by using OCR. Using OCR allows that the camera of the device is used to create an image of a computer screen. That image is then further processed and the URL contained in the image gets extracted. This process can be initiated by pressing the camera icon on the right inside the text field. A new window (figure 6.6) is shown exposing the following controls to the user:

1. With the rectangle the target area (usually the URL bar) can be selected. The selected area should be as tight as possible around the text. The rectangle can be moved by touching and moving it around. The size can be changed with the dot on each corner.
2. With the exposure slider the brightness of the image can be controlled. It should be bright, but still readable.
3. The focus slider controls the sharpness of the image and needs to be moved to a position in which the text appears sharp and not blurry.
4. The shutter button takes the photo and starts the OCR process. While processing a info dialog is shown and the result is entered into the text box (figure 6.7). If the detected text is incorrect it can be edited or the process can be started again.

The text entered into the text field can be used to interact with the password manager. It is possible to start a search using this term or create a new entry. This is further explained in the following sections.

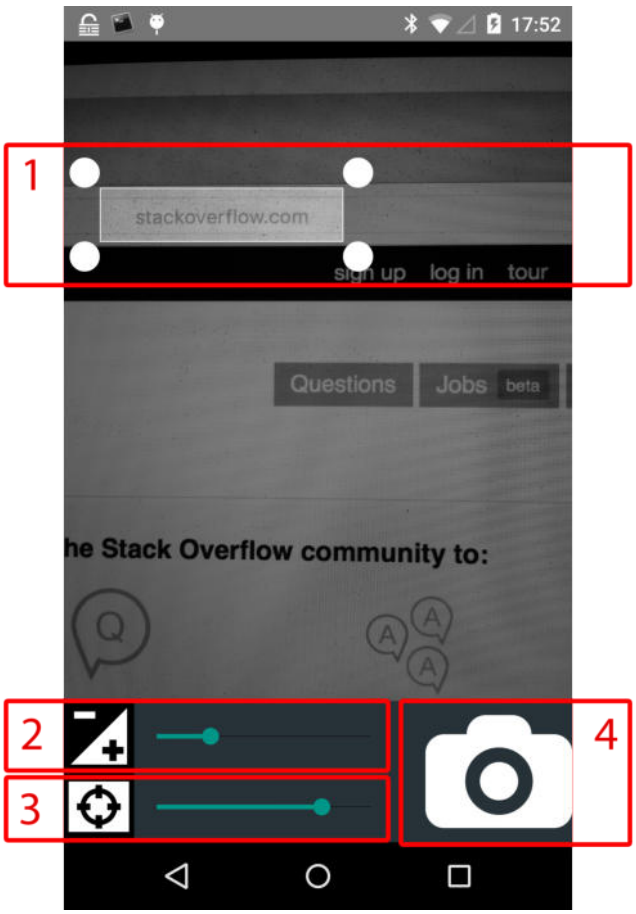


Figure 6.6: Credential HID - OCR interface

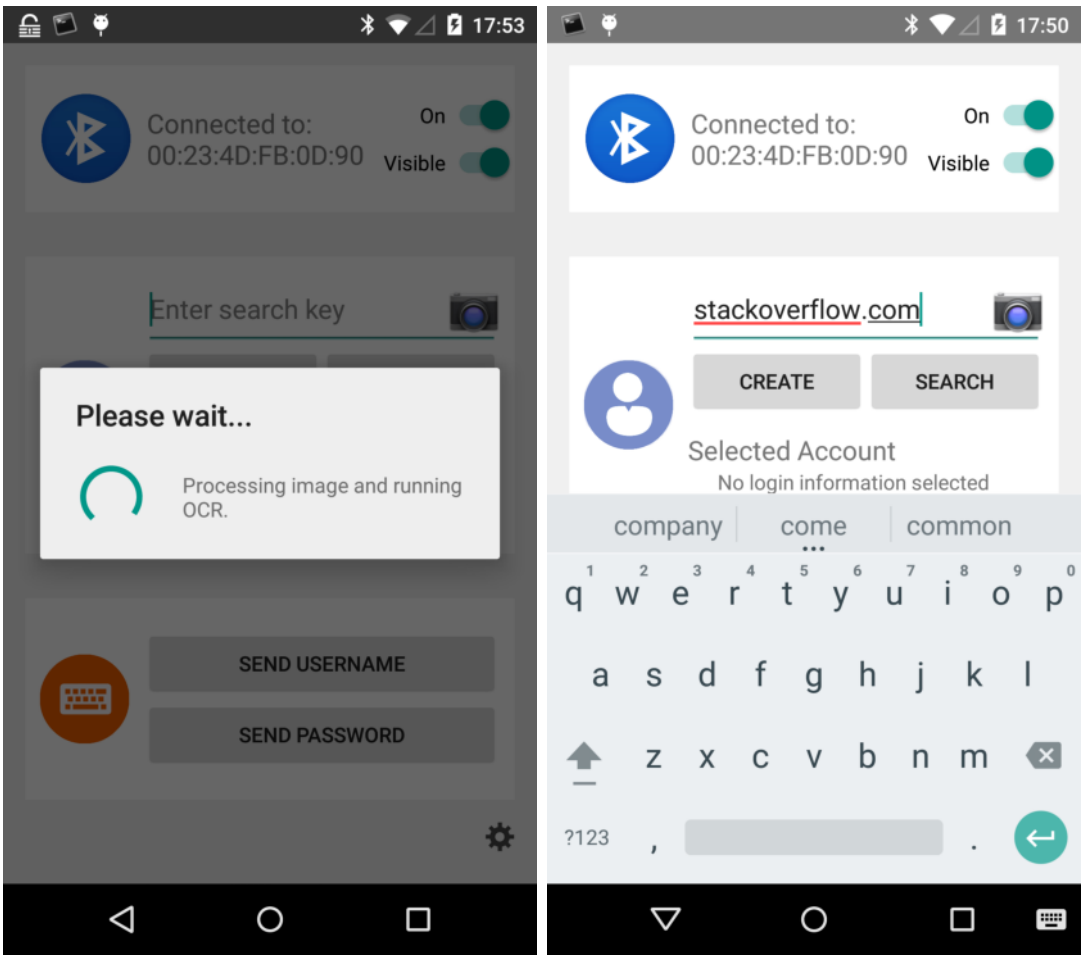


Figure 6.7: Credential HID - OCR processing and result

6.5 Create Account

To use the URL in the text field to create a new account in the password manager the button "Create" can be used. This will open the password manager and show the main screen (figure 6.8). Maybe it is necessary to unlock the password database first. From there a new account can be created with the following steps:

1. An entry is always stored inside a specific group. It is possible to create a new group or use one of the standard groups. Select the group where the new entry should be created (e.g Internet).

2. Inside a group press the "Add Entry" button. This will open a dialog where the necessary detail information can be entered (figure 6.9).

3. The URL field is already pre-populated with the data from the Credential HID application. It is only necessary to enter data for the fields "name" and "username". The password field is populated in the next step.

4. With the "..." button beside the password field a new password can be generated. The password criteria should be chosen long and complex, because it is not necessary to type it manually anywhere. The button "Generate Password" creates the new password which is applied by pressing "Accept".

5. The button "Save" stores the new account to the password database and returns the control to the main window of the Credential HID application.

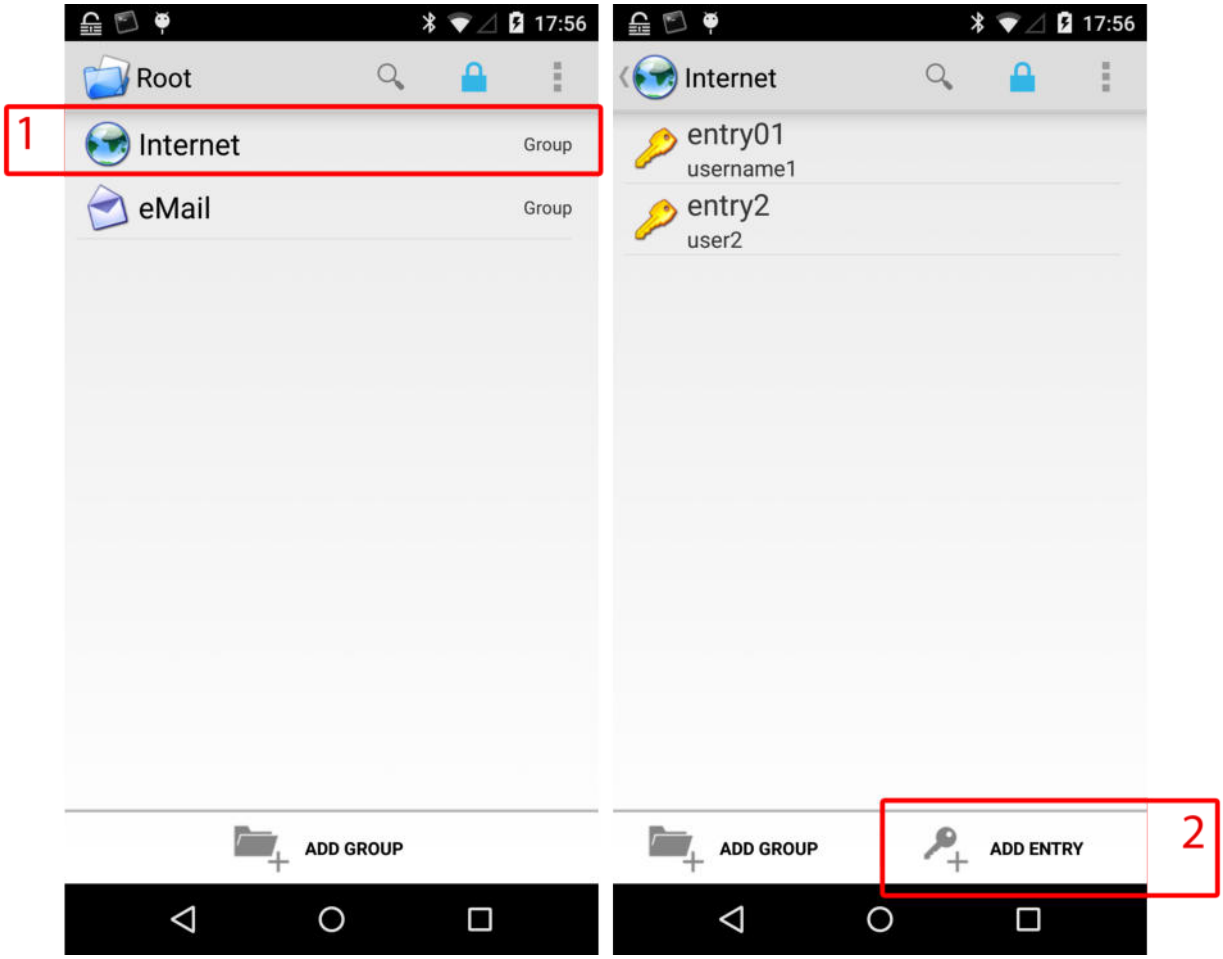


Figure 6.8: Credential HID - Create account

Figure 6.9: Credential HID - Account information

6.6 Search Account

The URL in the text field can also be used to search an existing entry in the password database. The fields username, URL and title are searched for the entered keyword. Login information obtained in that way can be used to transfer to the connected Bluetooth device. The search is initiated by pressing the button "Search" in the main window. This switches the focus to the password manager. It may be necessary to unlock the password database first.

If a single matching entry is found the details are shown briefly (figure 6.10 left) and then returned to the main window of Credential HID automatically. In case multiple results were found all possibilities are listed and the correct one must be selected (figure 6.10 right).

The account currently selected is shown in the main window under "Selected Account". The information title, username and URL are shown (figure 6.11).

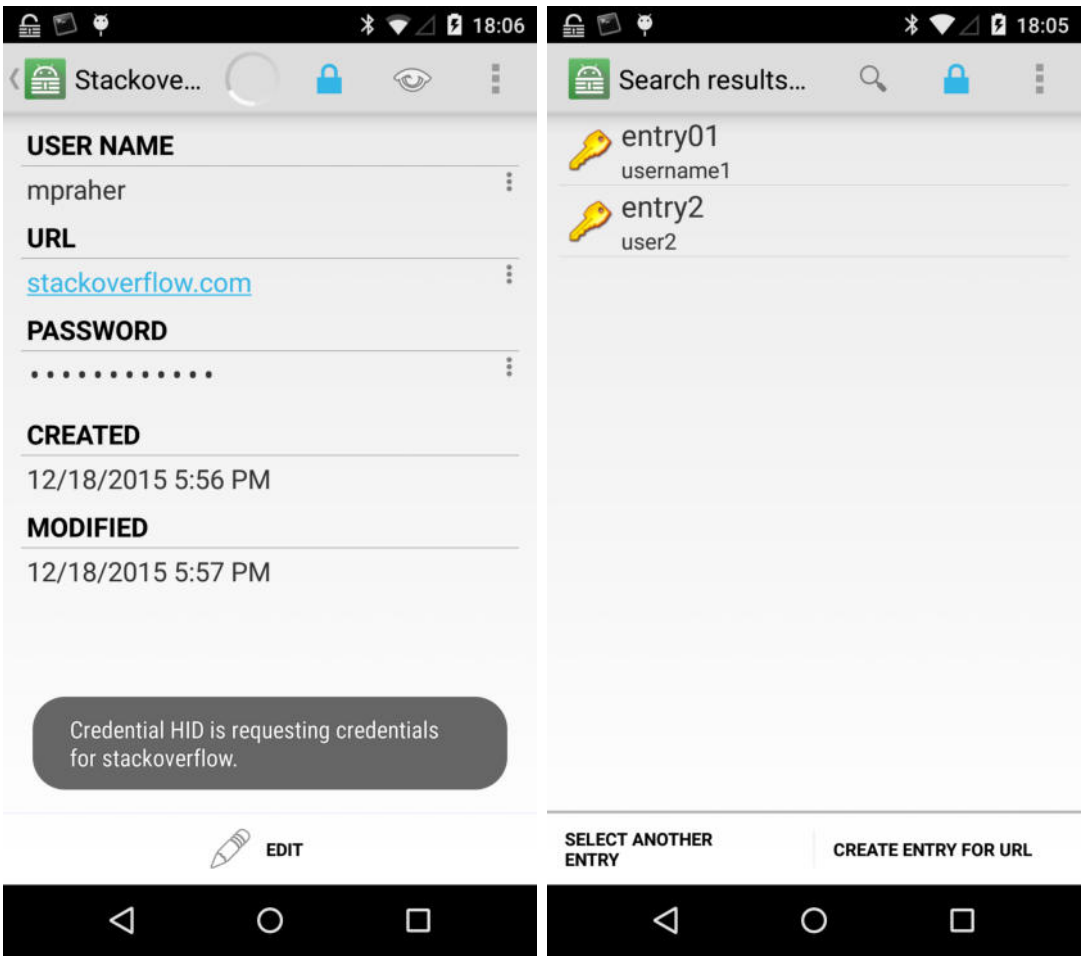


Figure 6.10: Credential HID - Search account

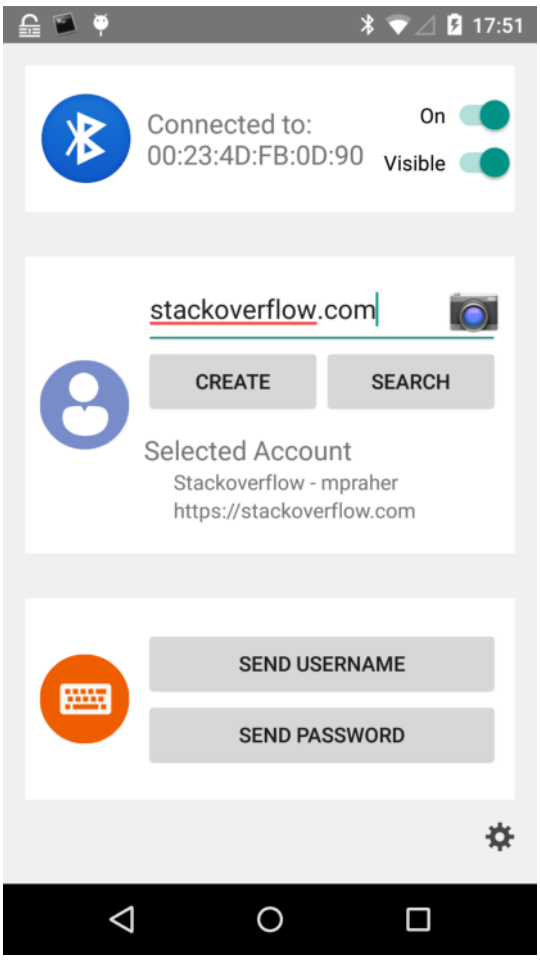


Figure 6.11: Credential HID - Selected account

6.7 Transfer Data

As soon as an active Bluetooth connection exists and an account from the password manager is selected the buttons to transfer the information are enabled (figure 6.12). Place the cursor of the target device in the username form field and press the "Send Username" button (A). The username is typed via the Bluetooth HID connection. As soon as the transfer is finished, place the cursor in the password form field and press the "Send Password" button (B).

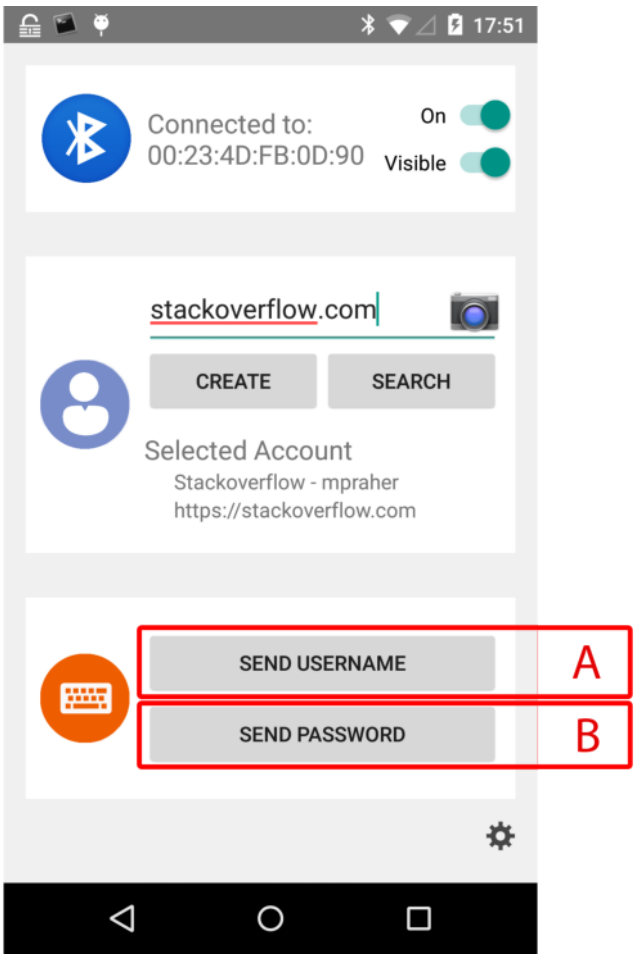


Figure 6.12: Credential HID - Transfer information

6.8 Delete/Modify Account

Modifying and deleting an existing account can be done directly in the Keepass2Android application. To do this, open the application, select the desired account and edit/delete it by following the instructions on the screen.

7 Test

Apart from normal application testing during development, the interfaces to other systems are particularly interesting. Important factors are the compatibility and reliability when interacting with these systems. In particular the functioning of the Bluetooth HID device profile and the accuracy of the OCR process are further considered.

7.1 Bluetooth HID

The application implements the Bluetooth HID device profile. That means Bluetooth HID hosts should be able to establish a connection, pair and receive keyboard events. Each individual test case has three different parameters:

- Target device: The operating system and Bluetooth stack of the device acting as the Bluetooth HID host. This device initiates the connection (initial and subsequent) to the application. Tested operating systems are Microsoft Windows 8.1, Linux with BlueZ 5 and Mac OS 10.10.
- Bluetooth CoD (Class of Device): The Bluetooth CoD broadcasted by the application to the HID host. Either peripheral/keyboard, the typical value for Bluetooth HID devices, or phone/smartphone, the typical value for Android phones, is used. Setting the value to keyboard should enhance compatibility with different HID host implementations.
- Bluetooth SDP (Service Discovery Protocol): Defines which services should be propagated via SDP to other devices. Either only the HID host SDP entry or all SDP entries of the device's services (including the HID host SDP entry) are distributed. Distributing only the HID host SDP entry should also enhance compatibility with different HID host implementations.

For each individual test case the complete Bluetooth HID functionality is tested. This includes the following main elements:

1. Initial pairing: Pairing the HID host with the application without an existing prior bond. Successful if a bond was created using the numeric comparison model (see chapter 2.3). Additionally the devices must be able to create and maintain a stable Bluetooth HID connection.

2. Subsequent Connect: Connecting a disconnected HID host and the application with an existing (valid) bond between the devices. Successful if the devices can create and maintain a stable Bluetooth HID connection.

3. Send Data: Send keypress events from the application to the HID host. This test is run after a connection has been established, both after the initial pairing and a subsequent established connection. Successful if the application can send multiple strings, each longer than 6 characters and containing uppercase letters, lowercase letters, digits and special characters. This test is run for every supported keyboard layout.

OS	CoD	SDP	Test Case	Result	Remark
Windows	Keyboard	All	1 - 2	OK	The warning "Driver is unavailable" is shown.
			3	OK	
	Keyboard	HID	1 - 3	OK	
	Smartphone	All	1 - 3	OK	
	Smartphone	HID	1 - 3	OK	
Linux	Keyboard	All	1 - 3	OK	
	Keyboard	HID	1 - 3	OK	
	Smartphone	All	1 - 3	OK	
	Smartphone	HID	1 - 3	OK	
Mac OS	Keyboard	All	1 - 3	OK	
	Keyboard	HID	1 - 3	OK	
	Smartphone	All	1 - 3	FAIL	The Bluetooth keyboard dialog only lists devices with a peripheral CoD.
	Smartphone	HID	1 - 3	FAIL	The Bluetooth keyboard dialog only lists devices with a peripheral CoD.

Table 7.1: Bluetooth HID test results

7.2 Optical Character Recognition

Optical character recognition (OCR), in combination with photo capture and image enhancement, is used to utilize the device's camera to determine text from a computer screen. Multiple factors affect the accuracy of the result:

- Computer screen: The photographed computer screen determines the image and the appearance of text on it. The influential factors are the screen resolution, scaling and screen brightness. The screen brightness only influences the exposure time and usually not the quality of the resulting image.
- Distance: The distance between the computer screen and the device influences the text size and artifacts on the resulting image. If the distance is too big the text is too small for recognition. If the distance is too small the individual pixels of the screen are discernible on the resulting image, which reduces the accuracy drastically.
- Text appearance: The representation of the text on the computer screen also plays an important role. Font type, font size, color and opacity have an affect on the quality of the OCR result.
- Human factor: Also the human using the system affects the accuracy of the OCR result. A steady hand and accurate selection of the text on the screen helps to improve the accuracy.
- Camera: The quality of the image, produced by the device's internal camera, also influences the OCR result.

The actual test is conducted by measuring the accuracy of the text recognition process, when scanning the URL bar of different web browsers displayed on a computer screen. Each individual test has three parameters:

1. Web Browser: The web browser displayed on the computer screen. Each browser's URL bar has a different text appearance. The tests are performed with the following web browsers in their default configuration: Mozilla Firefox 43, Google Chrome 47 and Microsoft Edge 25.10586.
2. Monitor: Testing is done using two different computer screens. The first screen (referred to as A) has a resolution of 2560×1600, a pixel density of 227 ppi and a scaling factor of 1,5. The second screen (referred to as B) has a resolution of 1920x1200, a pixel density of 94 ppi and no scaling.

3. URL: Testing is done using two different URLs. The URLs are chosen to reflect the representations browsers use for displaying different URL types. This is further explained in the result section after table 7.2.

Each test is run ten times and the result is averaged to minimize the impact of the human factor. The distance between the device and the screen is always chosen in a way, that the resulting image is sharp with the app's default focus setting. The camera itself is not considered, because the tests are all run on the same device. The accuracy is determined by calculating the Levenshtein distance, which is defined as "the minimal numbers of insertions, deletions and replacements to make two string equal"[42].

Figure 7.1: Address bar of different web browsers

Browser	URL	Monitor	Accuracy (%)
Mozilla Firefox	stackoverflow.com	A	95
		B	93
	https://login.microsoftonline.com	A	85
		B	69
Google Chrome	stackoverflow.com	A	98
		B	97
	https://login.microsoftonline.com	A	90
		B	86
Microsoft Edge	stackoverflow.com	A	100
		B	95
	https://login.microsoftonline.com	A	91
		B	75

Table 7.2: OCR test results

Figure 7.1 shows how the tested web browser display the URL (with the default configuration). Table 7.2 gives an overview of the test results. Considering both some interesting conclusions can be drawn:

- The screen with a higher pixel density, maintaining a normal font size, produces a better result across the board.
- The bigger a web browser display the actual URL the better the result. Overall accuracy is better for Microsoft Edge and Google Chrome, both displaying the URL slightly bigger than Mozilla Firefox.
- Text that is not black delivers a worse accuracy. This is especially problematic for URLs that contain a subdomain (e.g. `login.microsoftonline.com`). Edge and Firefox show the subdomain part in gray which produces a worse result. Additionally Firefox and Chrome show the protocol for secured sites (`https://`) in a different color than black, this also results in a lower accuracy.

7.3 Password Manager

The password manager (Keepass2Android) is used as the single storage for the user's credentials. The integration is done by using the predefined service interface provided by Keepass2Android. The password manager itself is not tested, only the integration into the Credential HID application. Two different extension points of the service interface are utilized by the application:

- Create Account: Used to initiate an action that allows the user to create a new account and store it into the password manager's database. Additionally it is possible to provide data to prepopulate individual fields.
- Query: Used to find accounts in the password manager's database. A query contains a search term which is used to search for matching entries.

The actual test is conducted by invoking the extension points of the service interface from the application. First multiple accounts are created and afterwards the password database is checked for correctness. Then those accounts are queried using different search terms that (should) match different fields (URL, title, username). The result delivered by the password manager is compared to the expected result.

7.3.1 Create Account

The create account action is invoked with an URL as a parameter. This URL is used to prepopulate the URL field in the create account UI of the password manager. The user then

has to complete the rest of the form and save or cancel the action. Each test case consists of the following parameters:

- URL: The URL for the new account. This value is provided by the Credential HID application via the service interface.
- Title, Username: The values for the title and username fields. These values are entered manually by the user into the create account UI of the password manager.
- Action: The final user action on the create account UI, either accept or cancel. Accept should create an persistent entry with the supplied information and cancel should discard all entered data.

Whether a test is successful depends on the action of the test case. After an accept action the new account should have been created with all the specified information. After an cancel action nothing about the new account should have been stored.

URL	Title	Username	Action	Result
`https://stackoverflow.com/users/login`	Stackoverflow	mpraher	Accept	Pass
`https://www.kusss.jku.at/kusss/index.action`	JKU	k0956126	Accept	Pass
`http://192.168.1.1`	OpenWRT LuCI	admin	Accept	Pass
`https://login.live.com/`	Outlook.com	mpraher	Cancel	Pass

Table 7.3: Password Manager - Create account tests

7.3.2 Query Account

The query account action is invoked with a single search term. The password manager than searches for matching entries by looking at the URL, title and username fields of each individual entry. A test case is successful if the password manager returns all accounts that match the provided search term.

Search term	Expected Results	Found Results
168	1	1
OpenWRT	1	1
mpraher	2	2
stackoverflow	1	1
Google	0	0

Table 7.4: Password Manager - Query account tests

7.4 Comparison with Requirements

At the beginning of this thesis (chapter 1) multiple requirements and desired properties for the solution were defined. The listing below outlines them and determines if they are met by the Credential HID application.

- Secure communication: The transfer of data from the Android device running the application to the target device should be secure. This requirement is achieved by utilizing the security mechanisms defined by the Bluetooth specification (see chapter 2.3). The devices create an initial connection in a way that is not susceptible to man-in-the-middle attacks. On subsequent connections the devices authentication each other and the data transfer takes place encrypted.

- Unnecessary to remember credentials: The solution should remove the user's necessity to remember usernames and passwords. This is possible due to the use of a password manager that stores this kind of information.

- Simplify credential input: Typing long and complex passwords on every login process is a slow and tedious task, that potentially leads to the use of weak passwords. Therefore the solution removes the necessity for the user to manually type usernames and passwords. This is done by simulating a Bluetooth HID keyboard that types the information automatically by pressing a single button in the application. Therefore the data input is fast, convenient and the use of long and complex passwords is possible without any drawbacks.

- Usage of strong passwords: To protect an account the user should choose a password that is long, complex and used only once. The solution enables the user to do this by generating a strong password for every individual account (via the password manager),

storing it, so the user does not have to remember it, and automatically typing it when necessary. This removes any burden that might arise by using strong passwords.

- Secure central storage: Unlike other solutions that sync credential information across multiple devices, the goal was to create a solution that stores this information in a secure way on a single device only. This is achieved by using a mobile Android device as the only storage and transferring the credentials only temporary when actually needed to another device via Bluetooth. To ensure the data is securely stored on the Android device a password manager, that protects all data, is used for actual storage.
- Utilize OCR to streamline credential search: The user should be supported in finding the correct credentials for a specific site. This is done by utilizing the camera of the Android device in combination with OCR technology to recognize the URL of a browser window on a computer screen. Therefore the user does not need to manually type a search term to find the right data.
- No installation on the target device: Another requirement was the ability to use the solution without the necessity to install anything on the target device. This is achieved by simulating a Bluetooth HID keyboard for the actual data transfer. Therefore the target device only needs a functioning Bluetooth setup. No additional software, specific to this solution, is necessary on the target device.

The implemented solution achieved all outlined requirements in the proposed way. It was not necessary to fallback to any of the alternative solutions. This means no system service or browser plugin was necessary for communication with the Android device. Likewise no further hardware, like the Flyfish Technologies hiDBLUE dongle, was necessary to provide a communication possibility.

8 Conclusion

This thesis explores how an Android device can act as a central storage and provider for login information. The device is used to store the data in a secure manner on the device and provide a way to transfer this information to other devices where it is needed. This transfer is designed to be both secure and easy to use. Additionally optical character recognition (OCR) technology is employed, which allows transferring textual data from a computer screen to the Android device.

The main objective was how to transfer login information from an Android device, where it is stored, to another device. Because this transfer should be done individually for every login process, Bluetooth seemed like a suitable choice. Particularly simulating a keyboard by employing the Bluetooth HID profile. To achieve this it was necessary to extend the Bluetooth stack by the HID device profile. Unfortunately this was only possible after switching the Bluetooth stack from the default BlueDroid to BlueZ. With this solution it is possible to communicate with any target device, provided it has a working Bluetooth setup. No additional software or special configuration is necessary. Additionally the connection establishment and data transfer is secured by mechanisms described in the Bluetooth specification.

Before the user is able to transfer credentials, it is necessary to find the correct ones first. To support the user in doing so, it should be possible to point the camera of the Android device to a browser window on a computer screen. The URL should then be detected automatically and used as a keyword for the search. To do this on the device itself OCR technology is necessary. Therefore the Tesseract OCR library was integrated. Because this alone provided non satisfying results some image enhancement prior to the OCR process is necessary. Therefore ImageMagick was integrated and suitable parameters for image enhancement determined.

Credential information is very sensitive and therefore has to be stored protected on the device. This job is usually done by a password manager. Because a lot of these already exist,

Keepass2Android was selected for storing the information on the device, instead of managing it directly in the Credential HID application. Keepass2Android was chosen because it is available under an open source license and can be tightly integrated into the application.

8.1 Problems

The following listing gives an overview over the biggest problems and challenges that emerged while creating the Credential HID application.

Extending Bluetooth for Android Android does not provide a way for application developers to extend the functionality of the Bluetooth stack. Therefore it is necessary to implement those features on a lower level, directly in the Bluetooth stack. Because BlueDroid, the default stack, is poorly documented and hard to extend, BlueZ was chosen for the implementation. To extend the Bluetooth stack on an Android device, it is necessary to have root access. Additionally to replace BlueDroid with BlueZ an unlocked bootloader is necessary.

Bluetooth HID hosts Different Bluetooth stacks behave differently when connecting to a Bluetooth HID device. Therefore it is necessary to analyze the differences and incorporate them in the implementation. Especially the requirements for HID hosts to connect to a keyboard vary. Some only connect to keyboards that identify themselves as "peripheral" and not to devices that identify themselves as "smartphone".

Password manager integration For the Credential HID application to be clean and easy to use, a good password manager integration is necessary. A lot of different solutions exist for the Android platform. However there a big differences in the way they can be integrated and sometimes it is not possible at all. The final solution uses Keepass2Android, because it provides a service interface for third party applications. Therefore it can be integrated in a way that feels native and is to some extend transparent to the user. Still the service interface does not provide the complete desired functionality and therefore the integration is not perfect.

OCR accuracy Optical character recognition is provided by an external library. The main purpose of the library is to detect text from a digitized version of paper. Therefore a lot of tweaking is necessary to achieve good accuracy when used for detecting text on a computer screen. First optimal settings for taking images have to be found. This does not only include camera parameters (like exposure time and ISO), but also the

optimal distance from the computer screen and selecting the text on the image. To further increase accuracy some image enhancements are necessary. This transforms the image in a way to make it look more like text on paper.

8.2 Future Work

The Credential HID application shows, how login information can be securely transfered from a safe storage on an Android device via Bluetooth to another device. Although the application does work multiple enhancements are possible:

1. BlueZ in stock image: The current solution uses a custom Android build that has BlueZ integrated. Because BlueZ can be used as a drop in replacement for BlueDroid, it should be possible to only build BlueZ and replace BlueDroid in the stock Android image. In that case no custom Android build is necessary, which reduces the complexity. Besides this way more devices can be supported, because an unlocked bootloader is not necessary. Additionally it can be tested risk free, because the original Bluetooth stack can always be restored easily.

2. BlueDroid: A further possibility for enhancement would be to implement the Bluetooth HID device functionality in BlueDroid, the default Bluetooth stack. This would remove the restriction to devices that run BlueZ.

3. OCR accuracy: Despite the optimizations already made to increase the OCR accuracy it can be further improved. The camera UI can be optimized to make it easier to use, e.g. allow text selection on the finished image and add a possibility to deskew the image. To further streamline the OCR process the user can be provided with an instant feedback. Instead of explicitly invoking the OCR process it can run in the background continually until the correct result is achieved. Furthermore the OCR engine can be trained with the URLs of the accounts stored in the password database. Also the actual search process should be moved from the password manager to the application itself. Then it would be possible to search for the best match and not rely on the built in search capabilities of the password manager. Additionally the parameters used for camera settings and image enhancement can be revisited, because it seems likely that better values can be found for more modern hardware. This is possible due to the fact that newer devices (usually) have better image sensors, which would allow higher ISO settings. Similar with increases in computing power a more sophisticated image enhancement is possible.

4. Password manager: The currently used password manager (Keeass2Android) allows a pretty good integration. Yet it could still be improved upon. First the service call used to create a new entry does not return any data. Therefore the application does not know if the user actually created a new entry and the attributes chosen by the user. Therefore it is necessary to query the database after the create, which delivers not the best user experience. Second (as mentioned above) the search is not perfect and not very forgiving on inaccurate input. It would be better to search for the best match and rank the results within the application.

5. Custom Fields: The current implementation is designed for a classical username and password workflow. For some uses it would be desirable for the user to be able to define additional custom fields.

Bibliography

[1] ABBYY. *ABBYY Mobile OCR Engine*. URL: http://www.abbyy.com/mobile-ocr/ (visited on 12/20/2015).

[2] Android Open Source Project. *Android Bluetooth*. URL: https://source.android.com/devices/bluetooth.html (visited on 10/05/2015).

[3] Anthem. *Anthem - Frequently Asked Questions*. URL: https://www.anthemfacts.com/faq (visited on 01/20/2016).

[4] AOSP Issue Tracker. *Support Bluetooth L2CAP*. URL: https://code.google.com/p/android/issues/detail?id=58164 (visited on 10/16/2015).

[5] AOSP-BlueZ. *Android Open Source Project with BlueZ 5*. URL: https://code.google.com/p/aosp-bluez/ (visited on 10/19/2015).

[6] Paul Asiimwe. *Android-ImageMagick*. URL: https://github.com/paulasiimwe/Android-ImageMagick (visited on 12/20/2015).

[7] Bluetooth SIG. *Bluetooth*. URL: https://www.bluetooth.org (visited on 10/30/2015).

[8] Bluetooth SIG. *Bluetooth Core Specification 4.2*. 2014. URL: https://www.bluetooth.org/en-us/specification/adopted-specifications (visited on 10/30/2015).

[9] Bluetooth SIG. *Bluetooth Human Interface Device Profile 1.1*. 2012. URL: https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc%7B%5C_%7Did=246761 (visited on 10/30/2015).

[10] BlueZ Project. *BlueZ*. URL: http://www.bluez.org/ (visited on 10/12/2015).

[11] Broadcom. *BlueDroid*. URL: https://android.googlesource.com/platform/external/bluetooth/bluedroid/ (visited on 10/12/2015).

[12] Philipp Crocoll. *Keepass2Android*. URL: https://keepass2android.codeplex.com/ (visited on 12/12/2015).

[13] Jake Edge. *Returning BlueZ to Android*. 2014. URL: https://lwn.net/Articles/597293/.

[14] Facebook. *Important Message from Facebook's White Hat Program*. 2013. URL: https://www.facebook.com/notes/facebook-security/important-message-from-facebooks-white-hat-program/10151437074840766 (visited on 01/20/2016).

[15] Sascha Fahl et al. "Hey, You, Get Off of My Clipboard." In: 2013, pp. 144–161. DOI: 10.1007/978-3-642-39884-1_12. URL: http://link.springer.com/10.1007/978-3-642-39884-1%7B%5C_%7D12.

[16] FlyFish Technologies. *hiDBLUE*. URL: http://www.flyfish-tech.com/hiDBLUE/ (visited on 12/21/2015).

[17] Free Software Foundation. *GNU Make*. URL: https://www.gnu.org/software/make/ (visited on 10/19/2015).

[18] Git. *Git*. URL: http://git-scm.com/ (visited on 10/19/2015).

[19] Google. *Android - Android Debug Bridge (adb)*. URL: https://developer.android.com/tools/help/adb.html (visited on 10/30/2015).

[20] Google. *Android - Building the System*. URL: https://source.android.com/source/building.html (visited on 10/23/2015).

[21] Google. *Android - Content Providers*. URL: http://developer.android.com/guide/topics/providers/content-providers.html (visited on 12/11/2015).

[22] Google. *Android Developers - Camera*. URL: http://developer.android.com/guide/topics/media/camera.html (visited on 11/25/2015).

[23] Google. *Android Developers - Intent*. URL: http://developer.android.com/reference/android/content/Intent.html (visited on 10/30/2015).

[24] Google. *Android - Developing*. URL: http://source.android.com/source/developing.html (visited on 10/19/2015).

[25] Google. *Android - Downloading and Building*. URL: https://source.android.com/source/initializing.html (visited on 10/19/2015).

[26] Google. *Android - envsetup.sh*. URL: https://android.googlesource.com/platform/build.git/+/master/envsetup.sh (visited on 10/30/2015).

[27] Google. *Android Interface Definition Language (AIDL)*. URL: http://developer.android.com/guide/components/aidl.html (visited on 10/30/2015).

[28] Google. *Android - Native Development Kit (NDK)*. URL: https://developer.android.com/tools/sdk/ndk/index.html (visited on 10/30/2015).

[29] Google. *Binaries for Nexus Devices*. URL: https://developers.google.com/android/nexus/drivers (visited on 10/19/2015).

[30] HID WG. *Human Interface Device Profile 1.1*. 2012.

[31] Jerry Hildenbrand. *How to unlock the Nexus 5 bootloader*. 2013. URL: http://www.androidcentral.com/unlocking-nexus-5-bootloader (visited on 10/23/2015).

[32] Marcel Holtmann. *Android HAL protocol for Bluetooth*. URL: http://git.kernel.org/cgit/bluetooth/bluez.git/tree/android/hal-ipc-api.txt (visited on 10/16/2015).

[33] Marcel Holtmann. *Bringing the BlueZ back to Android*. 2014.

[34] ImageMagick Studio LLC. *ImageMagick - Command-line-options*. 2015. URL: http://www.imagemagick.org/script/command-line-options.php (visited on 11/30/2015).

[35] Intel Corporation. *BlueZ for Android*. URL: https://01.org/bluez-android (visited on 10/16/2015).

[36] Rachel King. *Adobe admits 2.9M customer accounts have been compromised*. 2013. URL: http://www.zdnet.com/article/adobe-admits-2-9m-customer-accounts-have-been-compromised/ (visited on 01/20/2016).

[37] Eduard Kovacs. *Citi Exposes Details of 150,000 Individuals Who Went into Bankruptcy*. 2013. URL: http://news.softpedia.com/news/Citi-Exposes-Details-of-150-000-Individuals-Who-Went-into-Bankruptcy-369979.shtml (visited on 01/20/2016).

[38] Sinan Kubba. *Club Nintendo Japan hacked*. 2013. URL: http://www.engadget.com/2013/07/05/club-nintendo-japan-hacked/ (visited on 01/20/2016).

[39] Yi Lu, Willi Meier, and Serge Vaudenay. "The Conditional Correlation Attack: A Practical Attack on Bluetooth Encryption." In: 2005, pp. 97–117. DOI: 10.1007/11535218_7. URL: http://link.springer.com/10.1007/11535218%7B%5C_%7D7.

[40] Manuel Luitz. *AndroHid*. URL: https://code.google.com/archive/p/androhid/ (visited on 12/20/2015).

[41] Daniel McCarney. *Password Managers: Comparative Evaluation, Design, Implementation and Empirical Analysis*. 2013.

[42] Gonzalo Navarro. "A guided tour to approximate string matching." In: *ACM Computing Surveys* 33.1 (Mar. 2001), pp. 31–88. ISSN: 03600300. DOI: 10.1145/375360.375365. URL: http://portal.acm.org/citation.cfm?doid=375360.375365.

[43] Perry Nguyen. *KeepShare for KeePass*. URL: https://github.com/pfn/keepshare (visited on 12/12/2015).

[44] Carly Okyle. *Password Statistics: The Bad, the Worse and the Ugly*. 2016. URL: http://www.entrepreneur.com/article/246902 (visited on 01/20/2016).

[45] Oracle. *Java*. URL: https://www.java.com (visited on 10/19/2015).

[46] Oracle Corporation. *Java Native Interface*. URL: http://docs.oracle.com/javase/7/docs/technotes/guides/jni/ (visited on 11/10/2015).

[47] Gustavo Padovan. *Bluetooth Security*. 2011.

[48] Chirag Patel, Atul Patel, and Dharmendra Patel. "Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study." In: *International Journal of Computer Applications* 55.10 (Oct. 2012), pp. 50–56. ISSN: 09758887. DOI: 10.5120/8794-2784. URL: http://research.ijcaonline.org/volume55/number10/pxc3882784.pdf.

[49] Brian Pellin. *KeePassDroid*. URL: http://www.keepassdroid.com/ (visited on 12/12/2015).

[50] Julianne Pepitone. *Massive hack blows crater in Sony brand*. 2011. URL: http://money.cnn.com/2011/05/10/technology/sony%7B%5C_%7Dhack%7B%5C_%7Dfallout/ (visited on 01/20/2016).

[51] Andrea Peterson. *eBay asks 145 million users to change passwords after data breach*. 2014. URL: https://www.washingtonpost.com/news/the-switch/wp/2014/05/21/ebay-asks-145-million-users-to-change-passwords-after-data-breach/ (visited on 01/20/2016).

[52] Emil Protalinski. *4.93 million Gmail usernames and passwords published, Google says 'no evidence' its systems were compromised*. 2014. URL: http://thenextweb.com/google/2014/09/10/4-93-million-gmail-usernames-passwords-published-google-says-evidence-systems-compromised/ (visited on 01/20/2016).

[53] Sergii Pylypenko. *Android Keyboard Gadget*. URL: https://github.com/pelya/android-keyboard-gadget (visited on 12/20/2015).

[54] Python Software Foundation. *Python*. URL: https://www.python.org (visited on 10/19/2015).

[55] RonsDev. *BlueCtrl*. URL: https://github.com/RonsDev/BlueCtrl (visited on 12/20/2015).

[56] Martin Sauter. "Bluetooth." In: *Grundkurs Mobile Kommunikationssysteme*. Wiesbaden: Springer Fachmedien Wiesbaden, 2013. DOI: 10.1007/978-3-658-01461-2_6. URL: http://link.springer.com/10.1007/978-3-658-01461-2%7B%5C_%7D6.

[57] Bruce Schneier. *Open Source and Security*. 1999. URL: https://www.schneier.com/crypto-gram/archives/1999/0915.html%7B%5C#%7DOpenSourceandSecurity (visited on 11/10/2015).

[58] Ray Smith. "An Overview of the Tesseract OCR Engine." In: *Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)*. 2007, pp. 629–633.

[59] Statista. *Annual number of data breaches and exposed records in the United States from 2005 to 2015*. 2015. URL: http://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/ (visited on 01/20/2016).

[60] Janc Szymon. "Bluetooth on modern Linux." In: *Embedded Linux Conference*. San Diego, 2016. URL: http://events.linuxfoundation.org/sites/events/files/slides/Bluetooth%20on%20Modern%20Linux%7B%5C_%7D0.pdf.

[61] TeamsId. *Announcing Our Worst Passwords of 2015*. 2016. URL: https://www.teamsid.com/worst-passwords-2015/ (visited on 01/20/2016).

[62] Tesseract-OCR. *Tesseract Open Source OCR Engine*. URL: https://github.com/tesseract-ocr/tesseract (visited on 11/15/2015).

[63] Robert Theis. *Tess Two*. URL: https://github.com/rmtheis/tess-two (visited on 12/10/2015).

[64] USB Implementers' Forum. *Universal serial bus (USB) - Device Class Definition for Human Interface Device (HID)*. 2001.

[65] Alan Viverette. *Tesseract Android Tools*. URL: https://github.com/alanv/tesseract-android-tools (visited on 12/10/2015).

[66] Kim Zetter. *Hackers Finally Post Stolen Ashley Madison Data*. 2015. URL: http://www.wired.com/2015/08/happened-hackers-posted-stolen-ashley-madison-data/ (visited on 01/20/2016).

[67] Kim Zetter. *Twitter hacked: 250,000 accounts believed compromised*. 2013. URL: http://www.wired.co.uk/news/archive/2013-02/02/twitter-hacked (visited on 01/20/2016).

Lebenslauf

Marco Praher
Bahnhofstraße 35, 4230 Pregarten
(+43) 680 310 370 3
m@praher.me

Berufserfahrung

07.2009 - Heute	Softwareentwickler
	SecureGUARD GmbH, Linz (Österreich)
03.2010 - 11.2012	Unternehmensinhaber und Softwareentwickler
	Byteplex Solutions GmbH, Tragwein (Österreich)
08.2007 - 09.2007	Praktikum Softwareentwickler
	HC Solutions GesmbH, Linz (Österreich)
07.2006 - 08.2006	Praktikum IT Administration
	Industrie Logistik Linz GmbH & Co KG, Linz (Österreich)
07.2005 - 08.2005	Praktikum IT Administration
	Industrie Logistik Linz GmbH & Co KG, Linz (Österreich)

Schul- und Berufsbildung

09.2013 - Heute	Masterstudium Computer Science
	Johannes Kepler Universität Linz
09.2010 - 08.2013	Bachelorstudium Wirtschaftsinformatik (BSc)
	Johannes Kepler Universität Linz
09.2003 - 06.2008	Höhere Abteilung EDV und Organisation (Matura)
	Höhere Technische Bundeslehranstalt Perg

Persönliche Fähigkeiten

Muttersprache	Deutsch
Weitere Sprache	Englisch (Verstehen C1, Sprechen B2, Schreiben B2)
Berufliche Fähigkeiten	(Visual) C++
	PowerShell, C#, ASP.NET, .NET Technologien
	HTML, CSS, JavaScript
	Python, Django Framework
	Konfiguration und Betreuung von Microsoft Infrastrukturen und Firewalls

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Ort, Datum

Unterschrift