Reiner-SCT cyberJack wave Communication Sample

This sample shows how to communicate between Android devices and the cyberJack wave. This sample is made for everyone who wants to communicate with Reiner SCT Bluetooth readers, over Bluetooth LE 4.0 and above. Bluetooth 2.1 is not supported anymore. Please note that the communication over Bluetooth LE is always asynchronously, the implementation is not made to perform very fast, for a better understanding and debug purposes, we implement many byte arrays as strings.

Attention:

This sample only works with Android devices from Android 4.4.2 and above. Older devices are not supported. Please make sure that your Android device needs Bluetooth LE 4.0 hardware. And check your cyberJack wave software, which should be 1.8.8 or above.

This sample shows 3 different protocols to communicate with the cyberJack wave

1. communication via Secoder 3, (Secoder Spec DK Bluetooth Low Energy Service_v1.2 20150127),here we are just asking the reader for the Secoder Info and show the parsed Values. The Secoder Info describes the different functionality's and capabilities of the Secoder device.

2. The transmission of transparent card commandos over the DK Transport protocol, with a thin version of the USB CCID protocol customized for the Bluetooth LE communication

3. A Secoder application sample, which generates a TAN, here we generate a TAN using the Secoder TAN Application, to make this sample work you need a bank card which supports chipTAN

Developer devices suggestions:

Asus Nexus 7 2013(the Nexus 7 2012 does not provide Bluetooth LE capabilities), HTC One m8, Samsung Galaxy S4, Samsung Galaxy Tab 4.

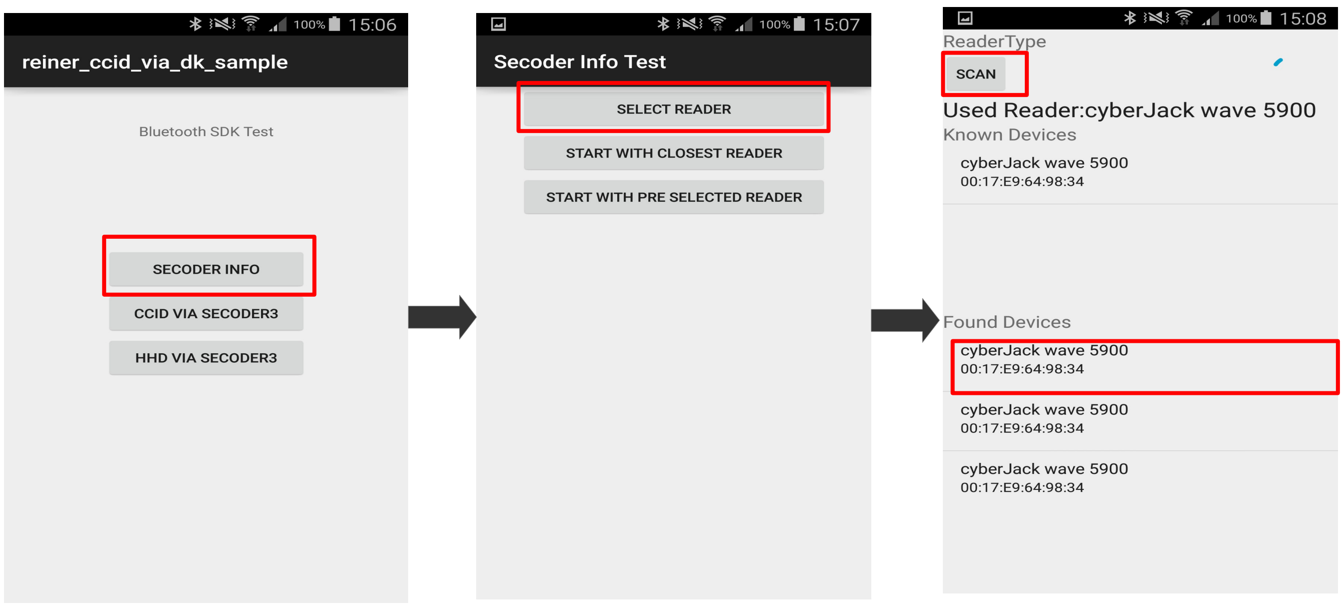
Order your personal cyberJack wave here:

https://www.chipkartenleser-shop.de/shop/rsct/article/5188

Please read the complete document carefully before you start developing.

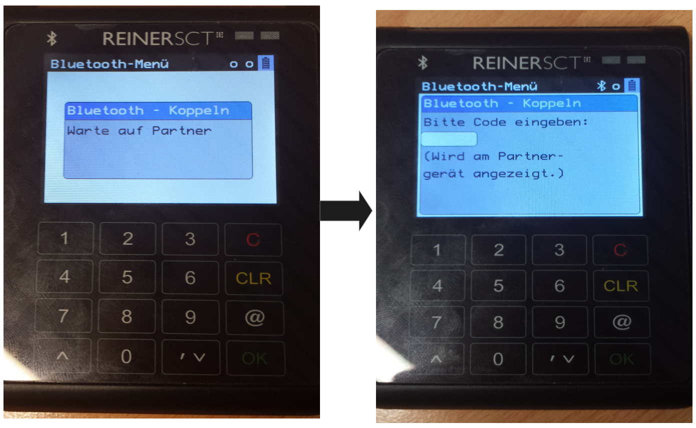
Function of the test application

Secoder Test



Here we are describing the first possibility to communicate with your cyberJack wave, please start your cyberJack wave , using the button on the back of the reader or inserting a smart card.

To bond the reader tap the @-symbol => Einstellungen => Bluetooth => Sichtbar machen. (please note attachment 1)



Now inside the App push the Button Select Reader, in the new Activity push Scan and wait till a reader appears in Found Devices, click this reader, now on the Android device as well as on the cyberJack wave, a bonding dialogue should appear. Please insert the PIN into the cyberJack wave and wait till the bonding dialogue on the Android device closes it self. The bonded reader is now saved as the default reader, and can be used at any time, without starting a scan for the device and you can guarantee that only one reader is used. Now you can start the reader after you have started the connect method of the SDK, which is helpful if the user started the process but he has no reader at hand.

Now get back to the Secoder Info test activity and choose “Start With Pre Selected Reader“, nun a command is send to the cyberJack wave and the answer is send back, the passed Answer is shown on the Android device.

Click the button “Start With Closest Reader“,now the next cyberJack wave in your near is Selected, please start the cyberJack so it can be found. now a bonding dialogue as mentioned above is opened. now a Secoder Info command is send to the reader and we wait for the answer. Next the answer is parsed and will be Displayed on your Android device. If more than one cyberJack wave is found in the area, the first reader found by your Android device is selected for the process, in this case it doesn’t matter if the reader was bonded previously.

An alternative to this method is the Android bonding menu, which can be found under settings Bluetooth, this is the same menu you can use to bond Bluetooth devices. Under the SelectReader Activity you can select a previously connected reader without bonding the reader again.

CCID Test

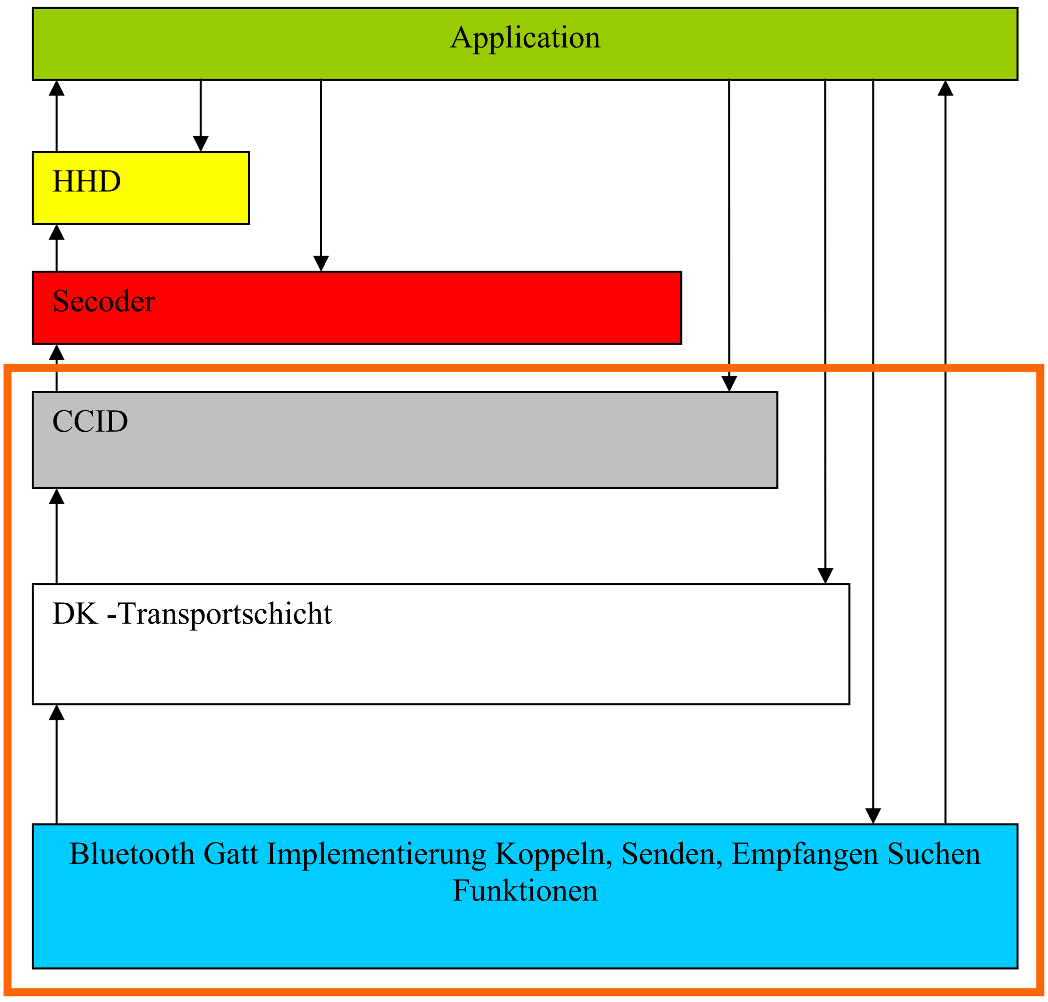
This sample demonstrates a simple transparent card command and is just a simple file select on a SmartCard.
At first we need to start the Card which is done by PowerOn which will return the ATR, then the SelectFile is send, and we wait for the answer which is Displayed on the Android device. At last we need to PowerOff the card.

You can use this sample according to Secoder Info scenario, if you have selected a reader in the Secoder Info Activity you don’t need to select one in this.

HHD Test

this sample demonstrates how to generate a TAN via a Secoder CTN application based on the HHD TAN protocol, to test this sample you will need a banking card which supports smart or chip TAN. We generate a HHD challenge command with hard coded parameters to the reader and wait for the answer block now the TAN is displayed in the Android device and a transmission success is send to the read either with a following command or final command flag. And the reader is disconnected. You can use this sample according to Secoder Info scenario, if you have selected a reader in the Secoder Info Activity you don’t need to select one in this.

Protocol Layers


Application
HHD
Secoder
CCID
DK -Transportschicht
Bluetooth Gatt Implementierung Koppeln, Senden, Empfangen Suchen Funktionen

Starting your own Implementation

First be sure about which, Layers you are working on, do you want to use a Secoder Application like CTN or AUT or would you like to send transparent card commands to the reader.

Crucial please give your android project the following permissions.

```
BLUETOOTH_ADMIN
BLUETOOTH
```

not necessary is

```
BLUETOOTH_PRIVILEGED
```

Secoder Application

To implement a Secoder application you can orientate on the HHD-TAN implementation code, found in the hhd packet. To successfully start a Secoder application your class should implement the SecoderReaderCallbacks and contain SecoderBluetoothReader object. After initializing the callback

initiated()

is called, Now you can use the SecoderBluetoothReader object, which represents a service on your Android device. Please note that this service has to be present in your manifest.xml like:

```
<service android:name="bluetooth.AndroidBluetoothService"></service>
```

After the successful start of the service you can connect to the reader, Therefore you need the address or id of the reader, used to connect the reader, You can obtain this information by starting a scan for appropriate devices by calling the function scanReaders(long timeout) of the SecoderBluetoothReader object.

```
_reader.scanReaders(5000);
```

this function starts a 5 second scan for a readers.

Please end all scans before start communicating with the reader or using the Android Bluetooth module in a different way. The scan blocks all operations on the Android Bluetooth module as long as it is scanning, this is an intense process for the Bluetooth module.

if you want to wait for the timeout of the scan wait for the callback

onScanningFinished()

if you want to stop the scan call

stopScaning()

and wait for the

onScanningFinished()

callback. Before starting another function of the SecoderBluetoothReader.

the found readers are reported in the

didFindReaders(List<Bluetooth_ReaderInfo> devices)

in every object

Bluetooth_ReaderInfo

you can find the reader ID
getReaderID()

used to connect to the reader, you can save this information for further use to connect to the reader without starting a scan before the connect. to connect just call the function

connect(String id)

of your SecoderBluetoothReader object and insert the Bluetooth_ReaderInfo ID and wait for the

readyToSend()

callback which is called after the Bluetooth communication of the devices (Android und cyberJack wave) are established. No we can start sending commands to the cyberJack wave, if you want to communicate with a Secoder application read the corresponding Secoder 3 spec, and orient yourself on the HHD sample.

It is recommended to query the Secoder Info of the reader and the supported Secoder 3 applications of the cyberJack wave and compare it with the ID for your application . Is your application listed in the Secoder Info applications we can send commands via the function sendCommand(String data, **boolean** transparent)

with the transparent flag to false because we are using an application and not a transparent card command.

wait for the answer in the

didRecieveApdu(String answer)

callback, If you are done with your communication please disconnect the reader for energy saving purposes.

disConnect(**boolean** justDisconnectDevice)

The flag justDisconnectDevice controls if you want to end the service on your Android device as well as the Bluetooth connection or just disconnect your device.
(if you are not calling the Secoder Info in your implementation and the connected reader does not support the Secoder Application you need, the Callback didRecieveResponseError(BluetoothErrors errorMessage, String respCode) in the respCode a error code Specified in the Secoder 3 spec is returned),

Transparent Card Commands

To send transparent card commands to the cyberJack wave use an object of the type CCIDBluetoothReader and implement the CCIDReaderCallbacks interface in your class which will provide the answers of the service. First initiate the CCIDBluetoothReader object, which will hook the service to the system and wait for the

initiated()

callback, called if the service is ready to be used. Please note that this service has to be present in your manifest.xml like.

```
<service android:name="bluetooth.AndroidBluetoothService"></service>
```

After the successful start of the service you can connect to the reader, Therefore you need the address or id of the reader, used to connect the reader, You can obtain this information by starting a scan for appropriate devices by calling the function scanReaders(long timeout) of the CCIDBluetoothReader object.

```
_reader.scanReaders(5000);
```

this function starts a 5 second scan for a readers.

Please end all scans before start communicating with the reader or using the Android Bluetooth module in a different way. The scan blocks all operations on the Android Bluetooth module as long as it is scanning, this is an intense process for the Bluetooth module.

if you want to wait for the timeout of the scan wait for the callback

onScanningFinished()

if you want to stop the scan call

stopScaning()

and wait for the

onScanningFinished()

callback. Before starting another function of the CCIDBluetoothReader .

the found readers are reported in the

didFindReaders(List<Bluetooth_ReaderInfo> devices) in every object

Bluetooth_ReaderInfo

you can find the reader ID

getReaderID()

used to connect to the reader, you can save this information for further use to connect to the reader without starting a scan before the connect. to connect just call the function

connect(String id)

of your CCIDBluetoothReader object and insert the Bluetooth_ReaderInfo ID and wait for the

readyToSend()

callback which is called after the Bluetooth communication of the devices (Android and cyberJack wave) are established. No we can start sending commands to the cyberJack wave, But first we need to power the card via the

powerCardOn(CardVoltage voltage)

command. Select a voltage to power the card and send the command to the reader. Then wait for the

didRecieveCCID_DataBlock(CCID_AnswerBlock ccid_AnswerBlock)

callback, In which are different Parameters of the connection and answer data are represented. in which you find after a successful powerOn the ATR.
Now the card is powered we can check the card status or send a command to the reader.

Now you can send your APDU via the

sendXfrBlock(String data)

function and wait for the answer in the

didRecieveCCID_DataBlock(CCID_AnswerBlock ccid_AnswerBlock)

callback. The answer is inside the block reachable via the getter

getCommandoData()

At any time you can call a

getSlotStatus()

and receive a CCID_AnswerBlock.

If your communication with the reader is done please Power of the card via

powerCardOff()

and you wil get a CCID_Answerblock with a fresh slot status powerdOff or unknown state as return now you can disconnect the reader via

disConnect(**boolean** justDisconnectDevice)

to save energy.

The flag justDisconnectDevice controls if you want to end the service on your Android device as well as the Bluetooth connection or just disconnect your device.

Packets

bluetooth (Bluetooth functions)

- AndroidBluetoothService (Bluetooth Service)
- Bluetooth_ReaderInfo (Bluetooth device container)
- BluetoothConnectionState(connection states)
- BluetoothErrors(error enum)
- BluetoothReader(Bluetooth reader functions)
- BluetoothReaderBondedReciever(Broadcast Receiver for bonding)
- BluetoothReaderCallbacks(Callbacks for the communication)
- BluetoothReaderService(the AndroidBluetoothService)
- BluetoothReaderType(reader types depending on the wave Software)
- BluetoothUUIDS(GattService and Bluetooth charakteristik UUID'S)

ccid (CCID implementation)

- CCIDBluetoothReader (BluetoothReader CCID layer)
- CCIDProtocoll(CCID protocol functions)
- CCIDReader(functions of the reader)
- CCIDReaderCallback(Callbacks of the CCID reader)

hhd (HHD implementation)

- HHDAnswer (parsed HHD answer data)
- HHDBluetoothReader(SecoderBluetoothReader with HHD layer)
- HHDGenerator(generates hhd challenges)
- HHDProtocoll(hhd protocol functions)
- HHDReader(HHDBluetoothReader functions)
- HHDReaderCallbacks(HHDBluetoothReader Callbacks)

secoder3 (Secoder3 implementation)

- SecoderBluetoothReader(BluetoothReaderService via Secoder transport protocol
- SecoderProtocoll(protocol functions)
- SecoderReader(SecoderBluetoothReader functions)
- SecoderReaderCallbacks(SecoderBluetoothReader callbacks)

secoderInfo (Secoder Info Container)

- SecoderAplications(Secoder applications container)
- SecoderAplicationCapabilities(Description of the Secoder application)
- SecocoderAplicationIDs(Strings of the Secoder applications)
- SecoderInfo(SecoderInfo class)
- SecoderInfoData(SecoderInfo answer data)
- SecoderInfoNumericReaderID(numeric reader id)
- SecoderInfoReaderProperties(reader properties)
- SecoderReaderQuallifiers(reader qualifiers)
- TLV(TLV parser)
- TLVINFO(TLV object)

userinterface

- BluetoothReaderSelection(the reader page)
- BluetoothTestTestActivity(test page activity)
- CCIDTestActivity(ccid sample)
- HHDTestActivity(hhd sample)
- Secoder3TestActivity(Secoder3 sample)
- TestActivity(start page)

utilitis(tools)

- BluetoothReaderInfoAdapter (Array adapter forSecoder_ReaderInfo objects)
- ByteOperations(working with bytes)
- SetList(list which does not allow duplets)

References

Secoder 3

Version:
Secoder Spec DK Bluetooth Low Energy Service_v1.2 20150127

Order here:

Bank-Verlag GmbH
Wendelinstraße 1
50933 Köln

Tel.:	+49-221-5490-0
Fax:	+49-221-5490-315
E-Mail:	medien@bank-verlag.de
Internet:	http://www.bank-verlag.de

CCID Spec

referenced under the following URL:

http://www.usb.org/developers/docs/devclass_docs/DWG_Smart-Card_CCID_Rev110.pdf

but changed partly to work with Bluetooth LE.

Chip TAN

Version:

Secoder3G_chipTAN_v010000_draft20150129

Bluetooth

The Current Bluetooth spec is found under:

https://www.bluetooth.org/en-us/specification/adopted-specifications