

PV204 Security technologies LABS



Introduction to smart cards and JavaCard platform



Petr Švenda  svenda@fi.muni.cz  [@rngsec](https://twitter.com/rngsec)

Centre for Research on Cryptography and Security, Masaryk University

CRCS

Centre for Research on
Cryptography and Security

GLOBALPLATFROMPRO

The masterplan for this lab

1. Communicate with smart card (GPPro tool)
 - ATR, basic info, CPLC
2. Communicate with card programmatically
 - Java `java.smartcardio.*` or C/C++ PC/SC API
3. JavaCard programming

The masterplan for this lab

1. Communicate with smart card (GPPro tool)
 - ATR, basic info, CPLC
2. Communicate with card programmatically
 - Java `java.smartcardio.*` or C/C++ PC/SC API
 - CPLC data
 - Obtain list of supported instructions from unknown card

1. Communicate with smart card (GPPro)

- Contact PC/SC readers + cards
- GlobalPlatformPro tool
 - <https://github.com/martinpaljak/GlobalPlatformPro/releases>
 - Basic smart card commands, sending APDUs
 - Management of GlobalPlatform cards (JavaCard)
 - Type `gp --help` for all functionality
 - We will use basic functionality now, rest next week

gp --info

- Obtain information about smart card
 - gp --info
 - Obtain ATR (Answer To Reset)
 - Parse using <https://smartcard-atr.appspot.com/parse?ATR=xxx>
 - Consult cplc-20211110.pdf from IS
- Who is the probable manufacturer of card?
- What is the probable environment for this card?
- Is it an open JavaCard?
- What is this card's circuit serial number?
- When was the card produced?
- What is different if 'gp --info --debug' is executed?



gp --apdu APDU_in_hexa --debug

- Send APDU command from command line
- Try gp --info --debug
 - Can you spot APDU command to obtain CPLC info?
- Send get CPLC APDU separately
 - gp --apdu 80CA9F7F00 --debug
- Can you relate card's response data and gp --info?
- What is the response status word?



2. Communicate with card programmatically

- SimpleAPDU project (IS, NetBeans)
 - Uses Java's `javax.smartcardio.*` API
 - `CardMngr.java` – utility functions for card communication
- Obtain list of available readers
 - `List readers = TerminalFactory.getDefault().terminals().list();`
- Connect to card
 - `CardTerminal.isCardPresent(), CardTerminal.connect("*");`
- Obtain ATR: `Card.getATR().getBytes()`
- Send APDU:
 - `ResponseAPDU resp = CardChannel.transmit(apdu)`

3. Communicate with card programmatically

- Try to send get CPLC command
 - Pre-prepared in GetCPLCData() method
 - Necessary to set proper APDU
- Parse response buffer
- Can you relate card's response data and gp --info?
- What is value of response status word?



Supported commands

- Card responds to some APDU commands
 - Generic ones (e.g., get CPLC data)
 - Custom ones (what card's owner wants)
 - Usually CLA/INS/P1 only (P2 sometimes)
- How to get list of commands supported by a card?
 - Look into documentation / standard (e.g., SIM commands)
 - Try to probe card (limited number of possible commands)
 - Be careful – many failed attempts may block your card!
- Task: find at least one ADU supported by card



Optional: BF complete list of all supported commands

- Write code that will try all combination of CLA/INS
- Observe response codes
- Make list of CLA/INS which returns interesting code
- Analyse with curiosity!

IMPORTANT: use only with the provided blue card or card you can afford to brick (e.g., old unused banking card)

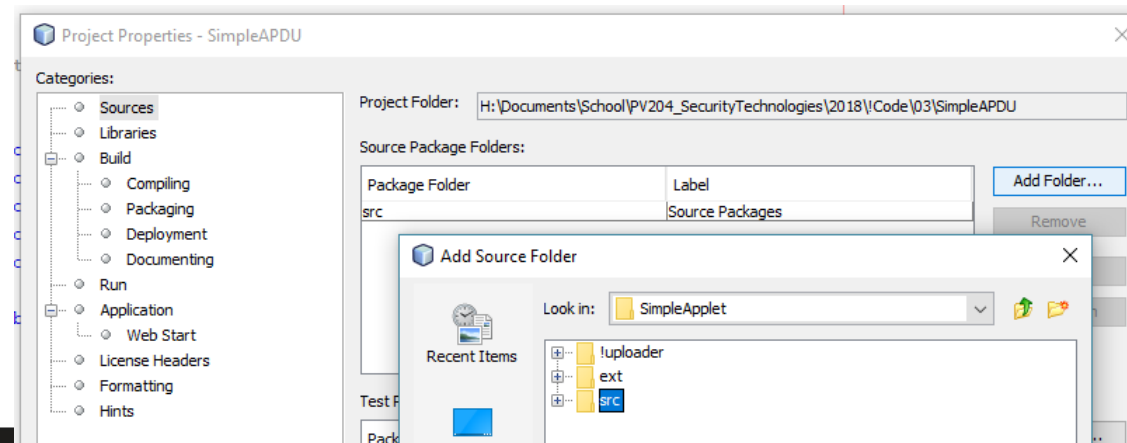
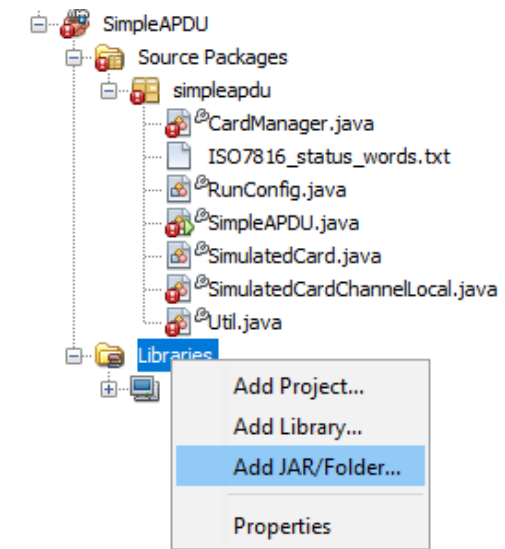
3. JAVACARD PROGRAMMING

Masterplan for today

1. Pre-prepared project for NetBeans
 - Debugging of applet with simulated card (jcardsim.org)
2. Compilation of applet from the command line
 - ant-javacard, GlobalPlatformPro for installation
 - Select applet and obtain random data
3. Quick switch from simulated to real card
4. Automation via gradle template
 - Use from command line
 - Import into IntelliJ IDEA

Setup SimpleAPDU [NetBeans]

- Run NetBeans and import existing project
 - File → Open project → SimpleAPDU
 - (Now contains only host application)
- Add jar library file with simulator
 - Libraries → Add JAR → \lib\jcardsim-3.0.5.5.jar
- Add link to folder with on-card applet
 - File → Project properties → Sources → Add folder
 - Add SimpleApplet\src



Setup SimpleAPDU [NetBeans]

- Project properties → Run → VM Options
 - `-enableassertions -noverify`
 - (required when custom compilation of jcardsim is used)
 - Not anymore, if you will use provided jcardsim-3.0.5.5.jar
 - Otherwise, you will get „*Exception in thread "main"*
java.lang.VerifyError: Expecting a stackmap frame at branch target 19“
- Project should now compile
- Try to run in debug mode
 - Place breakpoint inside applet's code

SIMULATED CARD (JCARDSIM)

Task: `demoGetRandomDataCommand()`

- Example code with single APDU command
- Simulated card is used
- Investigate the code, observe the values returned
- Place breakpoints and observe in debugger:
 - SimpleApplet constructor
 - `process()` method
 - `Random()` method

Tasks: demoEncryptDecrypt()

- Investigate method demoEncryptDecrypt
- **Task 1:** Prepare and send APDU with 32 bytes of data for encryption, observe output
- **Task 2:** Extract the encrypted data from the card's response. Send APDU with this data for decryption
 - Compare data for encryption with the decrypted data
- **Task 3:** What is the value of AES key used inside applet? Use debugger to figure this out
- **Task 4:** Prepare and send APDU for setting different AES key, then encrypt and verify

This applies only if you have older/unused banking card and smartcard reader

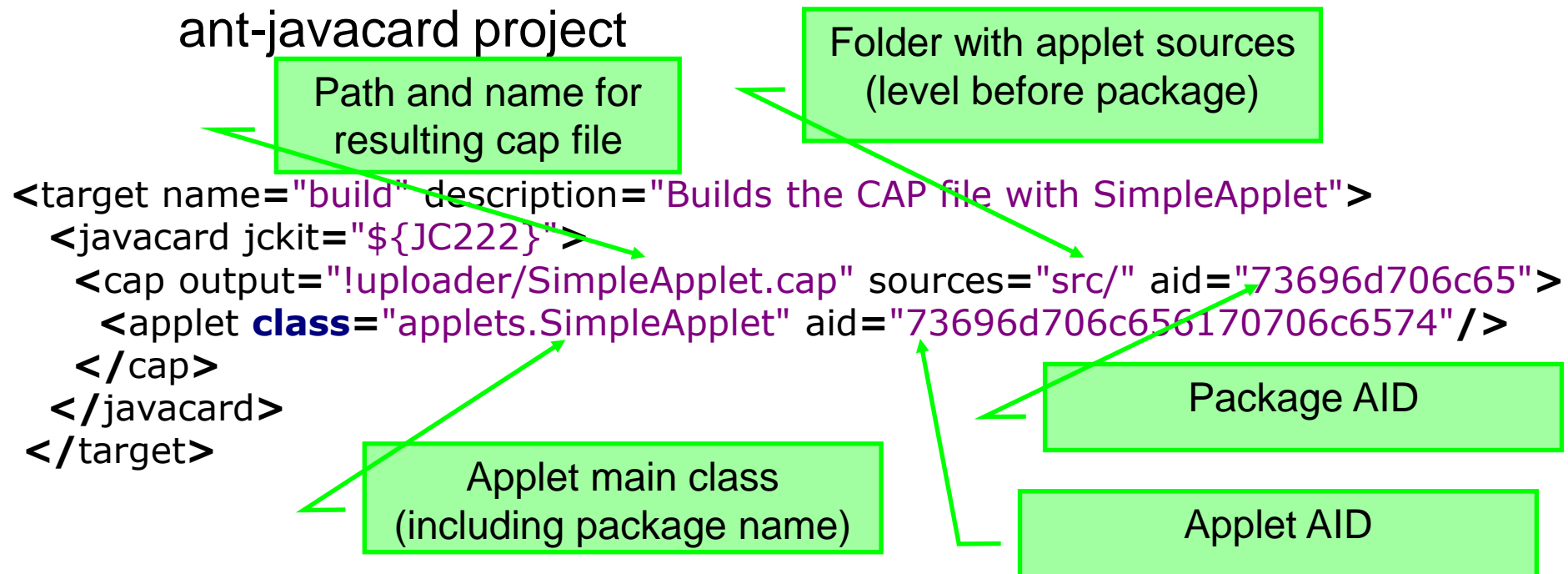
CONVERSION AND UPLOAD TO REAL CARD

Real cards –CPLC of your bank card

- Obtain ATR and CPLC info for your bank card
 - `java -jar gp.jar --info -d`
 - Don't worry, nothing is changed on card
- Can you figure out IC manufacturer and chip type?
- Can you figure out Operating System ID and name?
- Can you figure out fabrication date?
- Try to use `cplc.py` for reference
 - https://github.com/crocs-muni/JCAlgTest/blob/master/AlgTest_Process/cplc.py
- Can you locate CPLC in your project certificate?

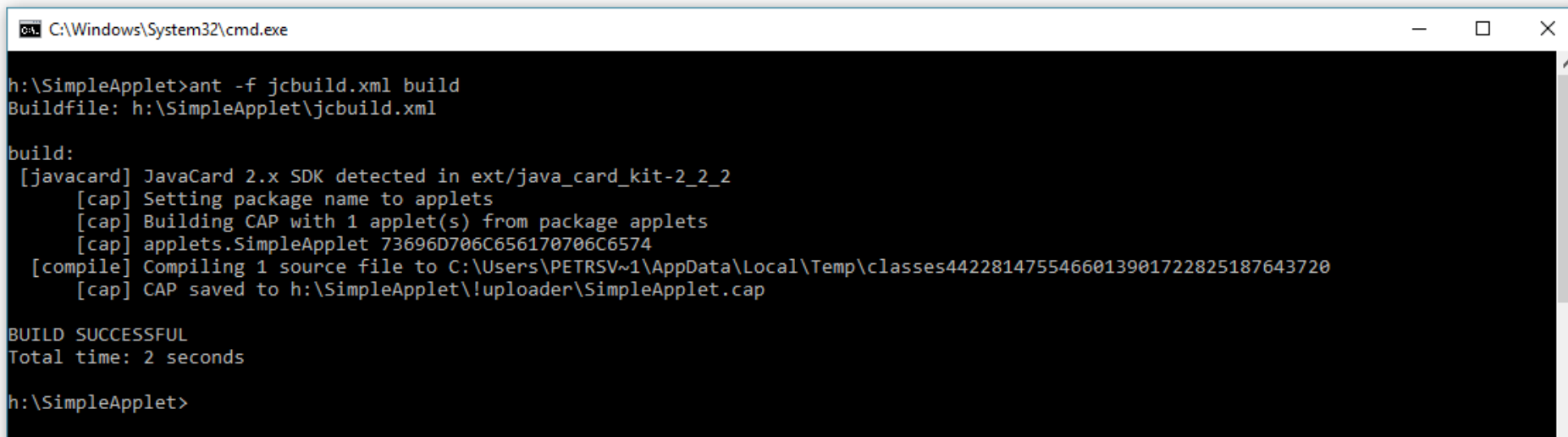
Task: Create cap file and upload to card

- Navigate to SimpleApplet folder
 - src folder contains applet's source code in SimpleApplet.java
 - `jcbuild.xml` contains configuration for conversion with ant-javacard project



Task: Create cap file and upload to card

- Compile & Convert
 - Execute on cmd line: `ant -f jcbuild.xml build`



```
C:\Windows\System32\cmd.exe
h:\SimpleApplet>ant -f jcbuild.xml build
Buildfile: h:\SimpleApplet\jcbuild.xml

build:
[javacard] JavaCard 2.x SDK detected in ext/java_card_kit-2_2_2
[cap] Setting package name to applets
[cap] Building CAP with 1 applet(s) from package applets
[cap] applets.SimpleApplet 73696D706C656170706C6574
[compile] Compiling 1 source file to C:\Users\PETRSV~1\AppData\Local\Temp\classes4422814755466013901722825187643720
[cap] CAP saved to h:\SimpleApplet\!uploader\SimpleApplet.cap

BUILD SUCCESSFUL
Total time: 2 seconds

h:\SimpleApplet>
```

- If OK, SimpleApplet.cap is created in !uploader folder

Task: Create cap file and upload to card

- <http://github.com/martinpaljak/GlobalPlatformPro>

1. List already loaded applets

```
- java -jar gp.jar -list -d
```

2. Uninstall previous version of SimpleApplet

```
- java -jar gp.jar -uninstall SimpleApplet.cap -d
```

3. Install SimpleApplet.cap

```
- java -jar gp.jar -install SimpleApplet.cap -d
```

4. Use applet (commands in SimpleAPDU code)

Problem: what with other applets on card?

1. List already loaded applets

- `java -jar gp.jar -list -d`

2. Find package_AID and run:

- `java -jar gp.jar -deleteddeps -delete package_aid`

- The `-deleteddeps` will also delete all applets from target package

- E.g., our SimpleApplet can be also removed by

- `gp -deleteddeps -delete 73696d706c65`

Be aware – real card can be blocked

- Too many unsuccessful authentication requests

```
>gp --list -debug
# Detected readers from SunPCSC
[*] Alcor Micro USB Smart Card Reader 0
SCardConnect("Alcor Micro USB Smart Card Reader 0", T=*) -> T=0, 3BF71800008031F
E45736674652D6E66C4
SCardBeginTransaction("Alcor Micro USB Smart Card Reader 0")
A>> T=0 (4+0000) 00A40400 00
A<< (0018+2) (56ms) 6F108408A000000003000000A5049F6501FF 9000
A>> T=0 (4+0008) 80500000 08 6265E168FB2639C1
A<< (0028+2) (118ms) 00003126960097543174010200103595AC1420213D2969EA8B8C41F3 90
00
```

```
openkms.gp.GPException: STRICT WARNING: Card cryptogram invalid!
Card: 3D2969EA8B8C41F3
Host: DB1E6E1E71958A15
!!! DO NOT RE-TRY THE SAME COMMAND/KEYS OR YOU MAY BRICK YOUR CARD !!!
    at openkms.gp.GlobalPlatform.printStrictWarning(GlobalPlatform.java:156)

    at openkms.gp.GlobalPlatform.openSecureChannel(GlobalPlatform.java:471)
    at openkms.gp.GPTool.main(GPTool.java:348)
```

Be aware – real card can be blocked

- Don't write script that executes many authentications at once (cycle, multiple commands)
- If unsuccessful one/two authentication is detected, then as for help, please!!!

USE OF REAL CARD

Tasks: demoUseRealCard()

- Change from simulator to real card
 - `runCfg.setTestCardType(RunConfig.CARD_TYPE.PHYSICAL);`
- **Task 5:** Obtain random data from real card
- **Task 6:** Set new key value and encrypt on card

ASSIGNMENT 2

Assignment 2: OpenFIPS201 analysis

- Analyze existing JavaCard applet
 - <https://github.com/makinako/OpenFIPS201>
- Answer the following questions:
 - What cryptographic algorithms are executed on card and with what parameterization? (use JavaCard constants like ALG_RSA_NOPAD)
 - For every algorithm, find the JavaCard specification version which introduced it first (e.g., JC API 2.1.1 for ALG_RSA_NOPAD)
 - Describe purpose of usage (achieved functionality) for a particular cryptographic algorithm (e.g., OwnerPIN cardPIN is used to protect RAM memory against fault induction)
 - Describe details of security mechanism used to protect APDUs going from and to applet
- Prepare applet to run in jcardsim simulator
 - Demonstrate working setup by description of process, issues encountered and screenshots
 - Comment out/modify code which is referencing org.globalplatform.* packages which are not available inside simulator
 - Change method in applet constructor just to register() with no parameters
 - Send at least one APDU command

Assignment 2: OpenFIPS201 analysis

- Bonus: upload and run with real card
 - Take reader and card from lab, write your name and reader serial number on the list
- Produce short (2xA4) text description of solution
 - Provide answers to questions asked
 - Provide demonstration of applet executed in simulator
- Submit **before 10.3.2020 23:59** into IS HW vault
 - Soft deadline: -1.5 points for every started 24 hours

Troubleshooting – jcardsim simulator

- Don't forget to add jcardsim-3.0.5.5.jar
- Use debugger – insert breakpoint directly into applet's method
- Local vs. remote simulator jcardsim
 - Only single card can be simulated as local one (`CAD.getCardInterface()`)
 - We will use and debug only one card (so local is fine)
 - Multiple cards can be used as remote simulators (sockets)

**SOLUTIONS (KIND OF 😊)
(VALID FOR BLUE CARDS FROM LAB)**

gp --info

- Who is probable manufacturer of card?
 - Gemplus/Gemalto
- What is probable environment for this card?
 - Possibly JavaCard with MPCOS applet
- Is it open JavaCard?
 - No (no CardManager with known keys)
- What is card's circuit serial number?
 - ICSerialNumber: 02006FC1 (Note: your card will be different)
 - Good to consider also other ICxxx values for uniqueness
- When was card produced?
 - ICFabricationDate: 1105
 - Probably 15th May 2011 (105th day of year ending with 1)

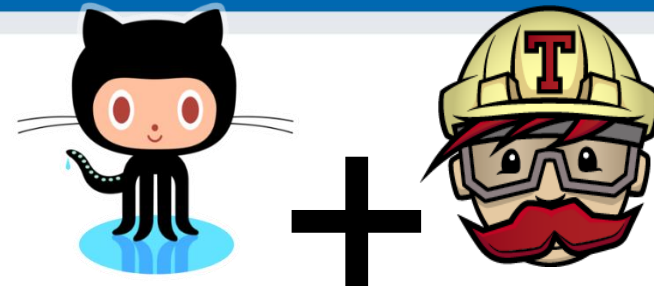
Probing unknown commands

- Probing possible because of:
 - limited space of command values
 - error message side channel
 - missing failed tries counter

GRADLE TEMPLATE

Usage of gradle template

- Intended for automatic conversion and tests
 - Used by TravisCI...
- IntelliJ IDEA can import project from gradle scripts
 - Interactive debugging, test coverage...
- Download complete zip
 - <https://github.com/crocs-muni/javacard-gradle-template-edu/releases/download/v0.2.0-edu/javacard-gradle-template-edu.zip>
- **Read README.md carefully!**
 - `./gradlew buildJavaCard --info --rerun-tasks`
 - `./gradlew test --info --rerun-tasks`



Integration with TravisCI

- .travis.yml

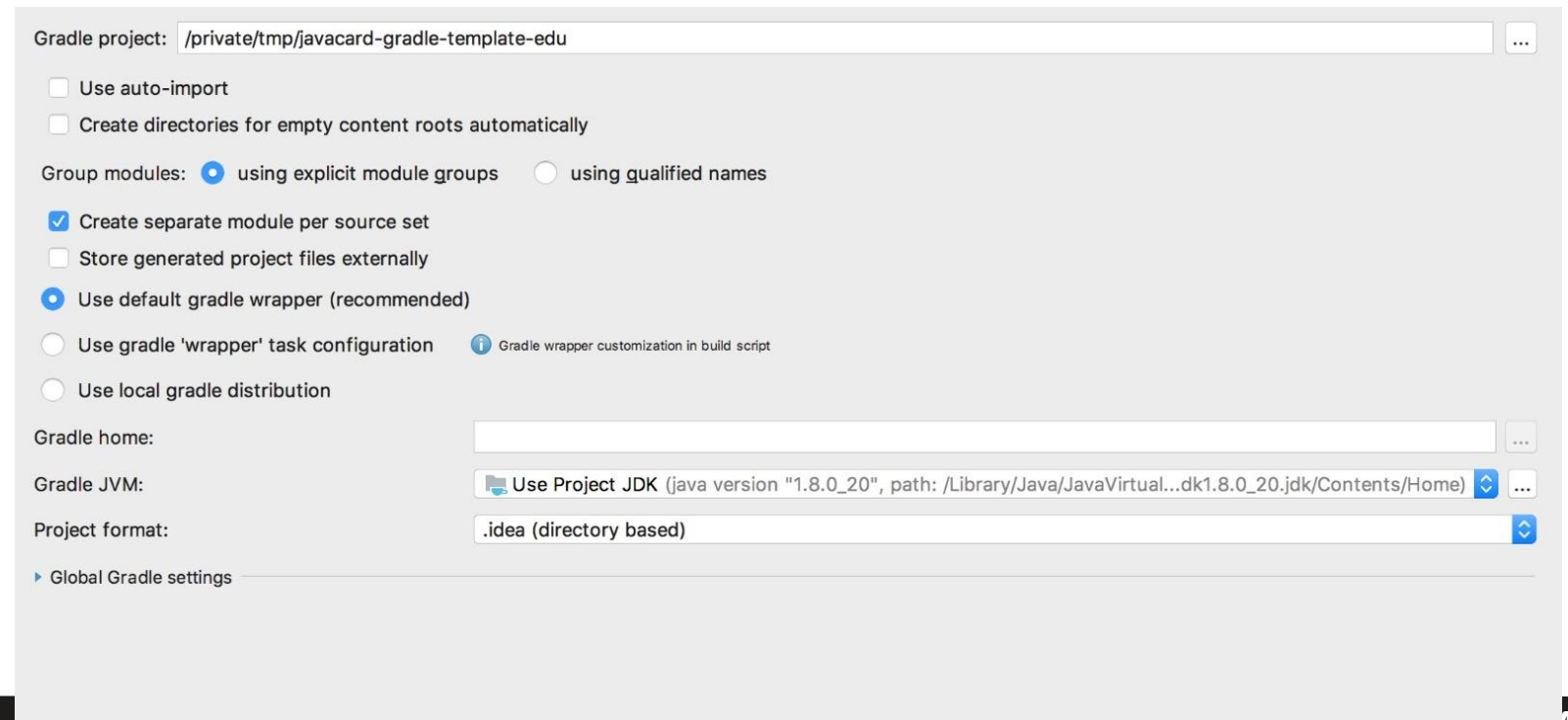
```
language: java

jdk:
- oraclejdk8

script:
- ./gradlew check --info
- ./gradlew buildJavaCard --info
```


Gradle + IntelliJ IDEA

- IntelliJ IDEA supports import of gradle –based projects (simpler IDE NetBeans not)
- Debugging with simulated card works as in NetBeans



Test coverage: Gradle + IntelliJ IDEA

- Go to Gradle plugin in IntelliJ IDEA
- Tasks → verification → test
- RClick → run with coverage

Coverage jcard:tester [test]

100% classes, 40% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
applet	100% (1/1)	60% (3/5)	89% (17/19)
cardTools	88% (8/9)	29% (18/61)	35% (89/250)

Gradle projects

- jcard (auto-import enabled)
 - jcard (root)
 - applet
 - Source Sets
 - Tasks
 - build
 - documentation
 - global platform
 - help
 - other
 - verification
 - check
 - test

Run 'javacard-gradle-temp...' with Coverage

IMPORTANT

- Due to distance learning situation, I do not assume that you have smartcard reader and (old) physical card
- If you do have both (e.g., old banking card), try to communicate with the card using GPPro tool as described by Tasks 1. and 2.
- If you don't have it, try Task 3. (EMV card and Android phone)
 - You may use your current banking card, application is reading out only public information