MTAT.07.017
Applied Cryptography

Smart cards (JavaCard)

University of Tartu

Fall 2022
Smart card security model

Parties involved in a smart card–based system:
- Cardholder
- Data owner
- Terminal owner
- Card issuer
- Card (software)manufacturer

Possible attacks to consider:
- attacks by the terminal against the cardholder
- attacks by the cardholder against the terminal
- attacks by the cardholder against the data owner
- attacks by the cardholder against the issuer
- attacks by the terminal owner against the issuer
- attacks by the issuer against the cardholder
- attacks by the (software)manufacturer against the data owner

Estonian ID card

- Used to:
  - Store ECC private keys
  - Perform on-card signing/decryption
  - Authorize cryptographic operations (using PIN)
- Cardholder / Data owner / Terminal / Card issuer / Card (software)manufacturer
- Attacks:
  - by the terminal against the cardholder
  - by the cardholder against the terminal owner
  - by the cardholder against the data owner
  - by the issuer against the cardholder
  - by the (software)manufacturer against the data owner
Payment cards (EMV)

- Used to:
  - Store symmetric MAC key
  - Authentication of transactions (using PIN)

- Attacks:
  - by the terminal against the cardholder
    - Relay attack
  - by the terminal owner against the issuer
  - by the issuer against the cardholder
Mobile phone SIM cards

- **Used to:**
  - Store 128-bit symmetric subscriber authentication key
  - Perform RUN GSM ALGORITHM
  - Authorize operations (using PIN)
  - Store contacts and SMS messages
  - Store settings (operator information)
  - Mobile-ID

- **Attacks:**
  - by the cardholder against the data owner
  - by the terminal owner against the issuer
  - by the issuer against the cardholder
Pay TV

- Used to:
  - Decrypt TV signal
  - Store channel filters

- Attacks:
  - by the cardholder against the data owner/issuer
  - by the terminal owner against the issuer
Attacks against smart cards

• Side channel attacks:
  • Timing analysis
  • Power analysis

• Fault injection:
  • Voltage, clock rate, radiation

• Physical attacks:
  • Chemical etching
  • Chip re-wiring
  • Adding a track
  • Cutting a track

• Countermeasures:
  • Metal layers
  • Onboard sensors (temp, light, frequency)
  • ...
Common Criteria (CC) security certification

- Target of Evaluation (TOE) – a product that has to be evaluated
- Protection Profile (PP) – identifies security requirements for class of products
  - For example, PP for Secure Signature Creation Devices
- Security Target (ST) – identifies the security properties of TOE
- Evaluation Assurance Level (EAL) – level of verification (1 to 7 + augmentation)
- Evaluation facilities – certified IT security testing laboratories
- Certification Bodies – issue CC certificates (e.g., ANSSI and BSI)
JavaCard

- Card capable of running code written in Java
- Stripped-down version of Java
  - Data types: boolean, byte, short
  - Not supported: char, String, float, int
  - One dimensional arrays
  - No threads
- Rich cryptography API available
  - Employs cryptographic coprocessor
  - Algorithm support depends on card
  - Side-channel protection guaranteed only for crypto API calls
- Java .class files have to be converted to a .cap file
- Most of the smart cards today run JavaCard code
JavaCard applet

```
$ cat TestApplet.java
package appcrypto;
import javacard.framework.*;
import javacard.security.*;
import javacardx.crypto.*;

public class TestApplet extends Applet {
    private RandomData rnd;

    public static void install(byte[] ba, short ofs, byte len) {
        (new TestApplet()).register();
    }

    private TestApplet() {
        rnd = RandomData.getInstance(RandomData.ALG_SECURE_RANDOM);
    }

    public void process(APDU apdu) {
        byte[] buf = apdu.getBuffer(); // contains first 5 APDU bytes

        switch (buf[ISO7816.OFFSET_INS]) {
            case (byte)0x00:
                if (buf[ISO7816.OFFSET_LC] != (byte)1) {
                    ISOException.throwIt(ISO7816.SW_DATA_INVALID);
                }
                apdu.setIncomingAndReceive(); // read APDU data bytes
                short len = (short)(buf[ISO7816.OFFSET_CDATA] & (short)0xff); // get rid of sign
                rnd.generateData(buf, (short)0, len);
                apdu.setOutgoingAndSend((short)0, len); // return response data
                return;
            ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
        }
    }
}
```
Converting to CAP

$ sudo apt install opensc openjdk-8-jdk ant
$ wget https://github.com/martinpaljak/ant-javacard/releases/download/v20.03.25/ant-javacard.jar
$ git clone https://github.com/martinpaljak/oracle_javacard_sdks

$ cat build.xml
<?xml version="1.0" encoding="UTF-8"?>
<project default="applet" basedir=""/>
<taskdef name="javacard" classname="pro.javacard.ant.JavaCard" classpath="ant-javacard.jar"/>
<target name="applet">
    <javacard jckit="oracle_javacard_sdks/jc222_kit/">
        <cap output="applet.cap" sources="/home/user/eclipse-workspace/appcrypto/src/">
            <applet class="appcrypto.TestApplet" aid="0102030405060708"/>
        </cap>
    </javacard>
</target>
</project>

$ ant
applet:
    [cap] INFO: using JavaCard 2.2.2 SDK in oracle_javacard_sdks/jc222_kit
    [cap] INFO: Setting package name to appcrypto
    [cap] Building CAP with 1 applet from package appcrypto (AID: 0102030405)
    [cap] appcrypto.TestApplet 0102030405060708
    [compile] Compiling files from /home/user/eclipse-workspace/appcrypto/src
    [compile] Compiling 1 source file to /tmp/jccpro2232880529567474390
    [javacard] NB! Please use JavaCard SDK 3.0.5u3 or later for verifying!
    [verify] Verification passed
    [cap] CAP saved to /tmp/jc/applet.cap
BUILD SUCCESSFUL
Total time: 1 second
GlobalPlatform

- Standard for applet management on JavaCards
- Multiple applets can be installed
  - An applet is SELECT’ed using Application Identifier (AID)
  - An applet can be set as the default applet (selected by default)
  - Applets are isolated (with exceptions – Shareable Interface)
- Applet can be deleted (usually), but never downloaded
- Security Domain (SD)
  - Every applet belongs to a SD
  - Card Issuer Security Domain (ISD)
  - Supplementary Security Domains (SSDs)
  - Secure Channel Protocol for communication with SD
Installing CAP file (GlobalPlatformPro)

$ wget https://github.com/martinpaljak/GlobalPlatformPro/releases/download/v0.3.5/gp.jar

$ java -jar gp.jar --install applet.cap --default
CAP loaded

$ java -jar gp.jar --list
[..]
AID: 0102030405060708 (........)
  App SELECTABLE: Default selected

AID: 0102030405 (.....)
  ExM LOADED: (none)
  0102030405060708 (........)

$ opensc-tool -s 00:00:00:00:01:05:00
Received (SW1=0x90, SW2=0x00):
  47 62 C6 A1 E3 Gb...

$ opensc-tool -s 00:00:00:00:01:a0:00
Received (SW1=0x90, SW2=0x00):
  CD 12 03 41 BA 22 69 32 1D 43 1E 46 21 26 76 8C ...A."i2.C.F!&v.
  1B D5 E1 F5 A6 6C 65 7E 87 68 B7 36 D9 6B 61 B0 ......le~.h.6.ka.
  07 E5 CF E6 D0 CE E2 28 9A 53 F4 6C 3B CB 3C 0F .......(S.1;.<.
  C7 E4 5D 9C EC FE 94 7D 07 6A 90 20 A1 F6 2E E4 ..].}..}'.j.
  26 D1 23 79 3F D2 F1 93 0E 1C 5E 11 8E 60 3E FB &.#y?......">.n
  1D 23 C4 E7 04 CB E2 67 96 77 48 F0 AF E2 30 00 .#.....g.wH...0.
  F9 E9 C4 63 10 D5 C0 6E 6F A4 4C 3B C6 54 75 99 ...c..no.L;..Tu.
  0B 42 6D C8 2F 40 D5 FD 2D CB 70 8C 1F 3C 30 4D .Bm./@..-.p..<0M
  D2 88 5F 68 3B 43 1C 85 D9 AD C0 B7 EC 01 3C 0E .._h;C.........<.
  D4 EE 01 F0 B5 EB 5E 8D E2 5C F4 D1 2E 09 77 BC ......^..

$ java -jar gp.jar --deletedeps --delete 0102030405
Blank JavaCard (Infineon)

- Chip: Infineon SLE78
- EEPROM: 150K
- RAM: ?
- JavaCard 3.0.4
- GlobalPlatform 2.2.1
- DES/3DES/AES256
- MD5/SHA1/SHA256/SHA512
- RSA-2048 (on-card generation)
- ECC-384 (on-card generation)
- CC EAL5+ high certification

Warning: Infineon RSA key generation flaw!
Blank JavaCard (Feitian)

- Chip: ST31
- EEPROM: 50K
- RAM: 5K
- JavaCard 2.2.2
- GlobalPlatform 2.1.1
- DES/3DES/AES128
- MD5/SHA1/SHA224/SHA256
- RSA-2048 (on-card generation)
- ECC-256 (on-card generation)
- Contactless Interface
- Garbage collector
- No security certifications

*Warning: On-card RNG flawed!*
Write a JavaCard applet that performs on-card RSA 2048-bit key generation and decryption:

$ ./test_applet.py
[+] Selected reader: Gemalto PC Twin Reader (E0660B9A) 00 00
[+] Infineon jTOP SLE78 (SLJ52GCA150)
[+] Generating a 2048-bit RSA key...
[+] Key generated in 15.01943 seconds!
[+] Retrieving the public key...
[+] N=190794771674569780198429666830627291334929824552257611784041727252007138393597039149
[+] e=65537
[?] Enter a message to encrypt: secret message!
[+] Encrypted message: 912d7285385d51c5511d3c108bd3b2361f7895d8f5f1b32192e3194afed22002225
[+] Sending ciphertext to card...
[+] Message decrypted in 0.742395 seconds!
[+] Decrypted message (15 bytes): secret message!

Commit TestApplet.java to your repository
Task: JavaCard applet

- Find out the communication protocol from test_applet.py
- JavaCard API calls to use:

```java
keypair = new KeyPair(KeyPair.ALG_RSA, KeyBuilder.LENGTH_RSA_*);
keypair.genKeyPair();

pub = (RSAPublicKey) keypair.getPublic();
pub.getExponent(byte[] buffer, short offset);
pub.getModulus(byte[] buffer, short offset);

rsa = Cipher.getInstance(Cipher.ALG_RSA_PKCS1, false);
rsa.init(keypair.getPrivate(), Cipher.MODE_DECRYPT);
rsa.doFinal(byte[] inBuff, short inOffset, short inLength,
        byte[] outBuff, short outOffset);
```
Task: JavaCard applet

- Size limit for data APDU body is 255 bytes
  - The first two bytes of ciphertext are embedded in P1 and P2
  - Make the ciphertext continuous using:

    ```java
    Util.arrayCopyNonAtomic(byte[] src, short srcOff, byte[] dest, short destOff, short length);
    ```

- Avoid memory leaks and EEPROM wear:
  - Initialize EEPROM objects only once!
  - Don’t write into EEPROM without a need!

- Make sure that the key pair is generated only once
  - Ignore (without error) repeated keygen requests

- Java has signed data types – cast byte to short using 0xff mask

- Debugging is possible only via the data or SW returned!
JavaCard memory management

EEPROM (or flash):
• Slow writes, subject to wear
• Preserves data on power loss

RAM:
• Fast writes (1000x faster)
• Loses data on power loss
• Small storage space

Persistent Objects:
• Class-member variables
• Static variables
• Array data

Transient Objects:
• Local variables
• Method parameters
• Transient array data (makeTransientByteArray())
• APDU buffer

A garbage collector is not present in JCRE (space of unreferenced objects is not reclaimed).
JavaCard applet development under Eclipse

- `sudo snap install eclipse --classic`
- Create a project: “File → New → Java Project → Project name: appcrypto”
- Right-click on the project “New → Class → Name: TestApplet, Package: appcrypto”
- Under the project “src” delete `module-info.java` file (if present)
JavaCard applet development under Eclipse

- Right-click on your project “Build Path → Configure Build Path... → Libraries → Add External JARs” and add oracle_javacard_sdks/jc222_kit/lib/api.jar
  This will enable JavaCard code validation and completion
JavaCard applet development under Eclipse

- Right-click on `api.jar` → “Properties → Javadoc Location → Javadoc location path:” and specify `http://www.win.tue.nl/pinpasjc/docs/apis/jc222/`

This will enable javadoc for JavaCard API calls