Accessing Secure Information using Export file Fraudulence

Guillaume Bouffard[1] Tom Khefif[1] Jean-Louis Lanet[1]
Ismael Kane[2] Sergio Casanova Salvia[2]

[1]Smart Secure Devices (SSD) Team – University of Limoges – Limoges, France
guillaume.bouffard@unilim.fr
http://secinfo.msi.unilim.fr

[2]Applus – LGAI Technological Center – Barcelona, Spain

CRiSIS 2013 – PhD Workshop

Outline

The Smart Card

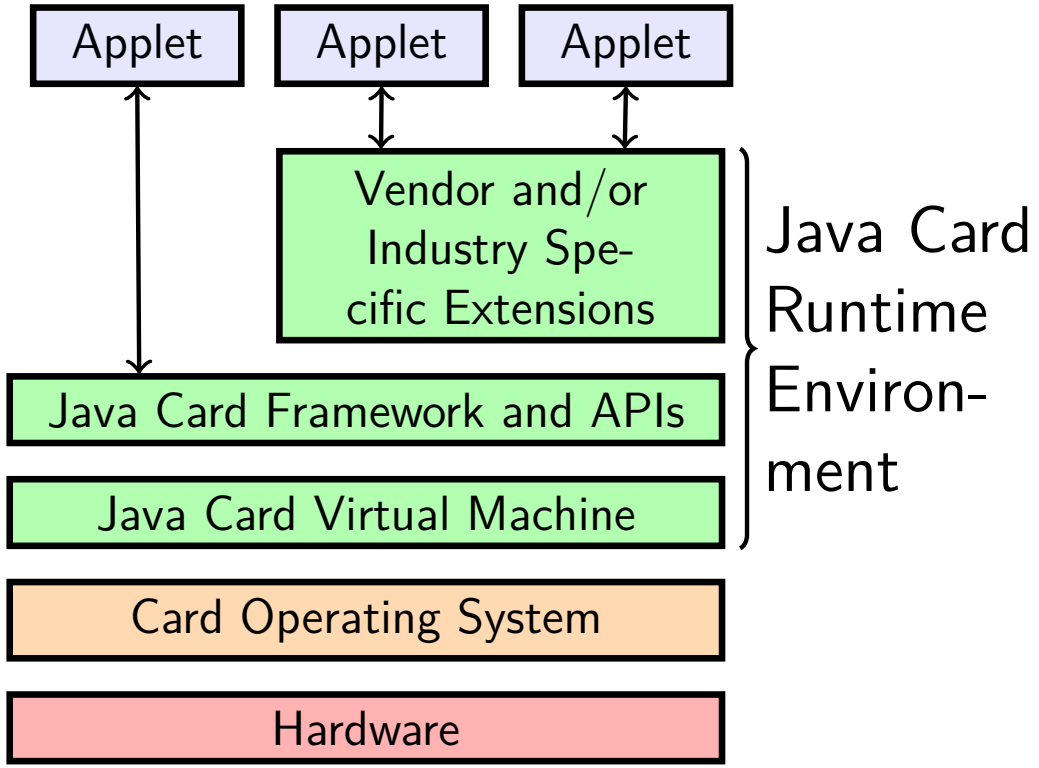
Widely used device

- Credit Card;
- (U)SIM Card;
- Health Card (french Vitale card);
- Pay TV;
- ...

This device contains sensitive data

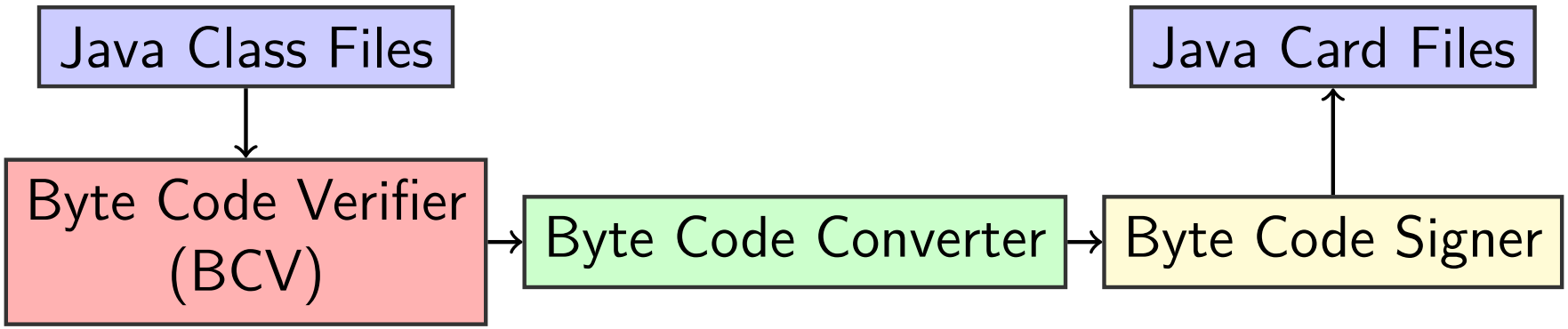
Java Card based Smart Card

- Created by Schlumberger in 1996.
- Specified by Oracle
- Provide a friendly environment to develop secured Java applications.



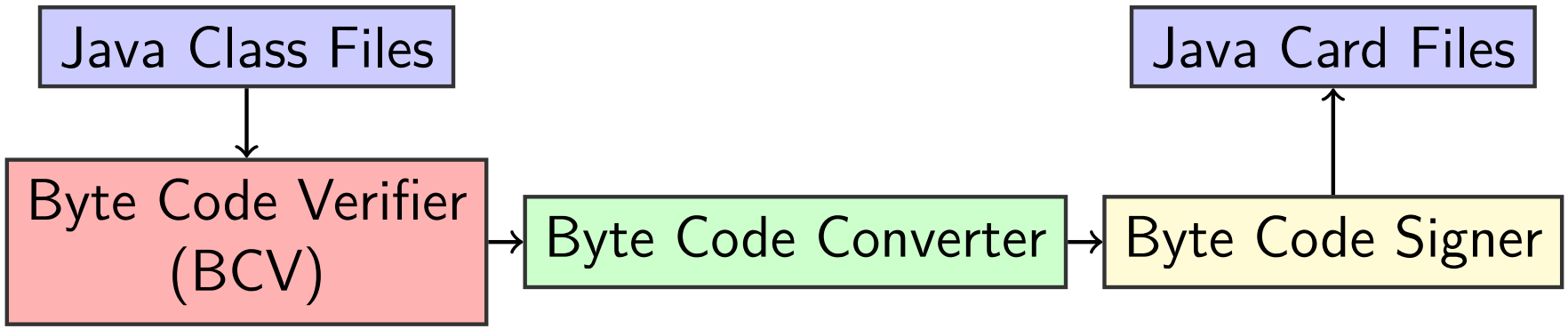
Java Card Security Model

- *Off-card* Security

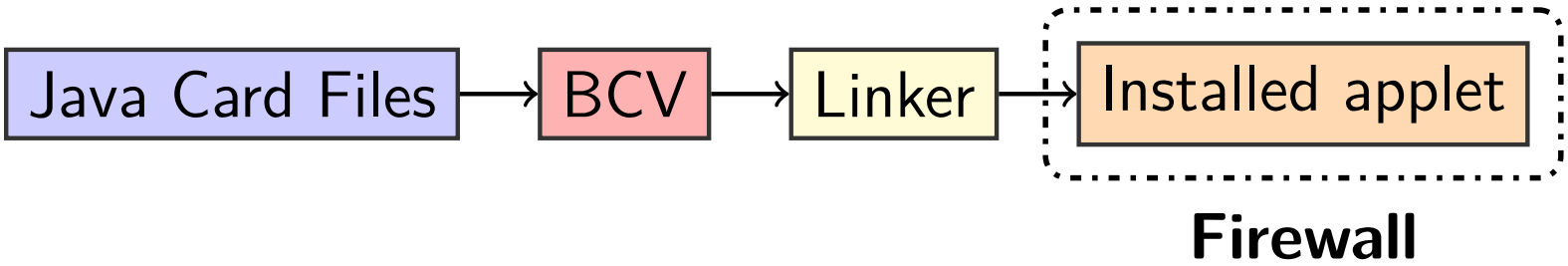


Java Card Security Model

- *Off-card* Security



- *On-card* Security



Off-card compilation

Java Archive

Application EXPORT File

Byte Code Converter, converter and signer

API EXPORT Files

Application CAP File

Off-card compilation

Java Archive

Application EXPORT File

Byte Code Converter, converter and signer

API EXPORT Files

Application CAP File

Off-card compilation

Java Archive

Byte
ver
verter

API EXPORT Files

*Java-*CLASS *Files*

- Non-optimized for embedded devices
- Itemized file
- Each item is an UTF8-String

#1 - Class Reference: name=#2

#2 - UTF8 Text: `fr/unilim/MyApplet`

#3 - UTF8 Text: `process`

#4 - UTF8 Text: `(Ljavacard/framework/APDU)V`

#5 - Method Reference: class=#1 signature=#6

#6 - Name/Type: name=#3 type=#4

Application CAP File

Off-card compilation

Java Archive

Byte
ve
verte

API EXPORT Files

API EXPORT *Files*

- CLASS' item to Java Card token
- 1 EXPORT file/Java-Package
- The Java Card toolchain uses the *first find, first used* EXPORT file.

```
class_info { // javacard/framework/APDU
token #10
access_flags public final
name_index 172 // javacard/framework/APDU
export_supers_count 1
...
```

Application CAP File

Off-card compilation

Java Archive

Application EXPORT File

Byte Code Converter, converter and signer

API EXPORT Files

Application CAP File

Off-card compilation

Program EXPORT *File*

- Describe each public methods shared by the built application or API.

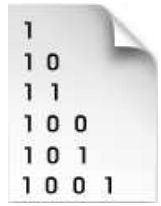
Application EXPORT File

Application CAP File

Off-card compilation

Program CAP File

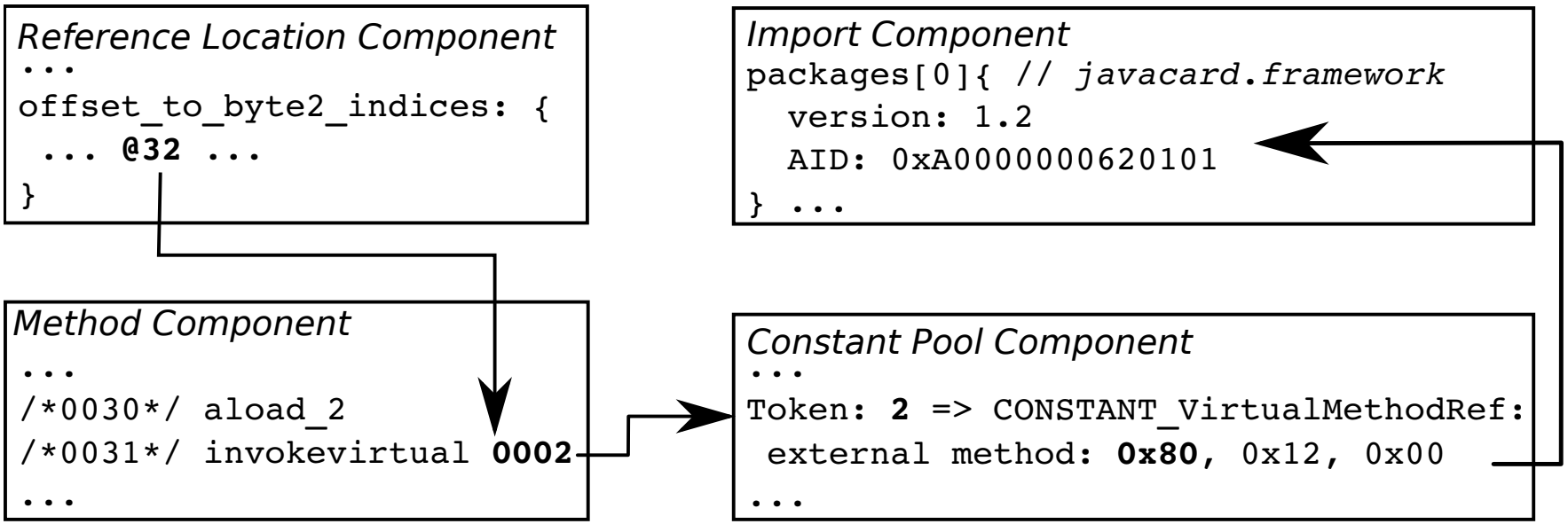
- Tokenized file
- Optimized for embedded devices



Application EXPORT File

Application CAP File

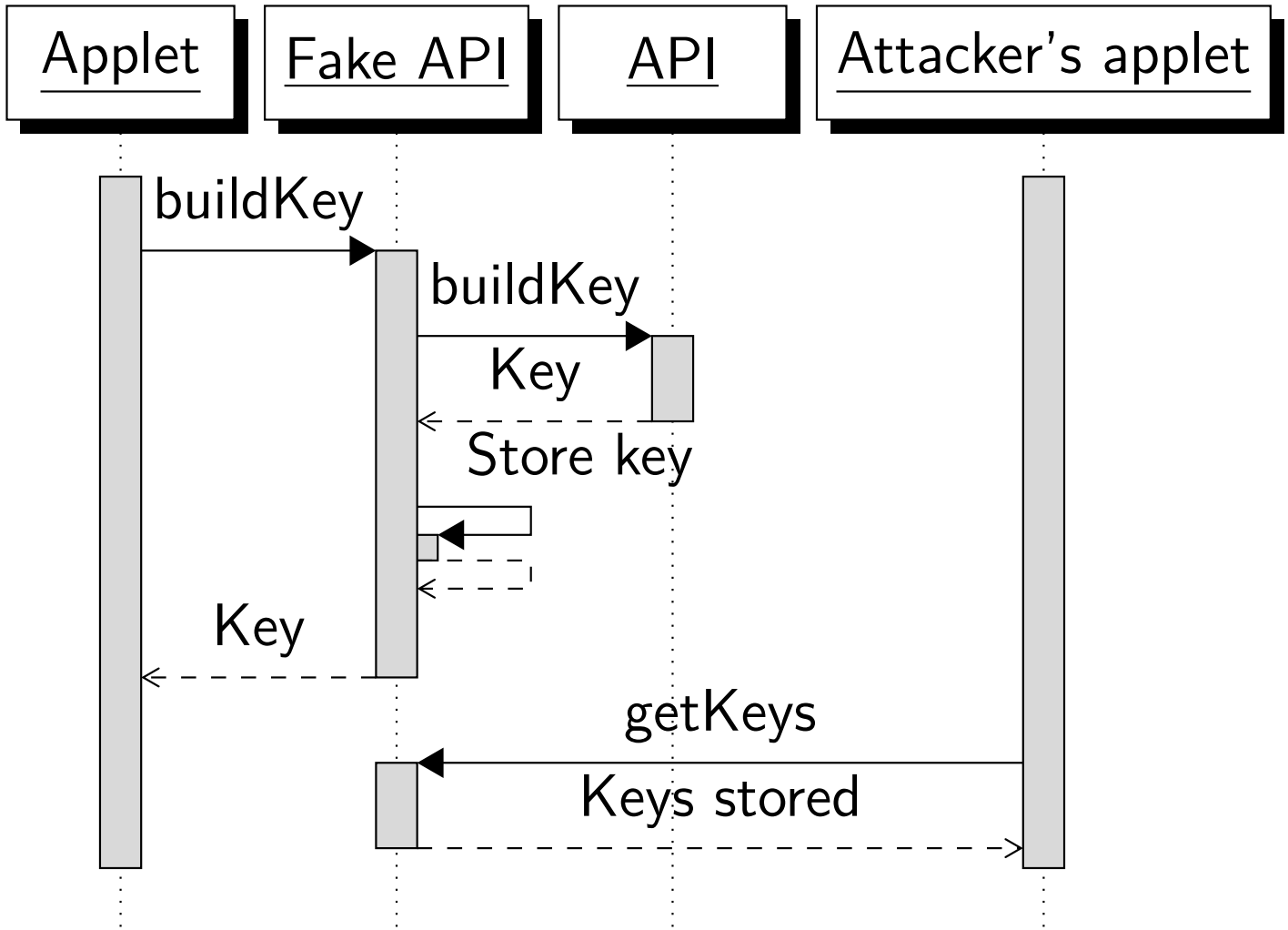
On-Card Linking Step



Man-In-The-Middle

- Attacks aims to:
  - Abuse the *Off-card* Java Card toolchain;
  - Link a malicious library instead of the legitimate one.
- Hypothesis:
  - The Java Card EXPORT folder can be corrupted;
  - The Smart Card's loading keys are known.

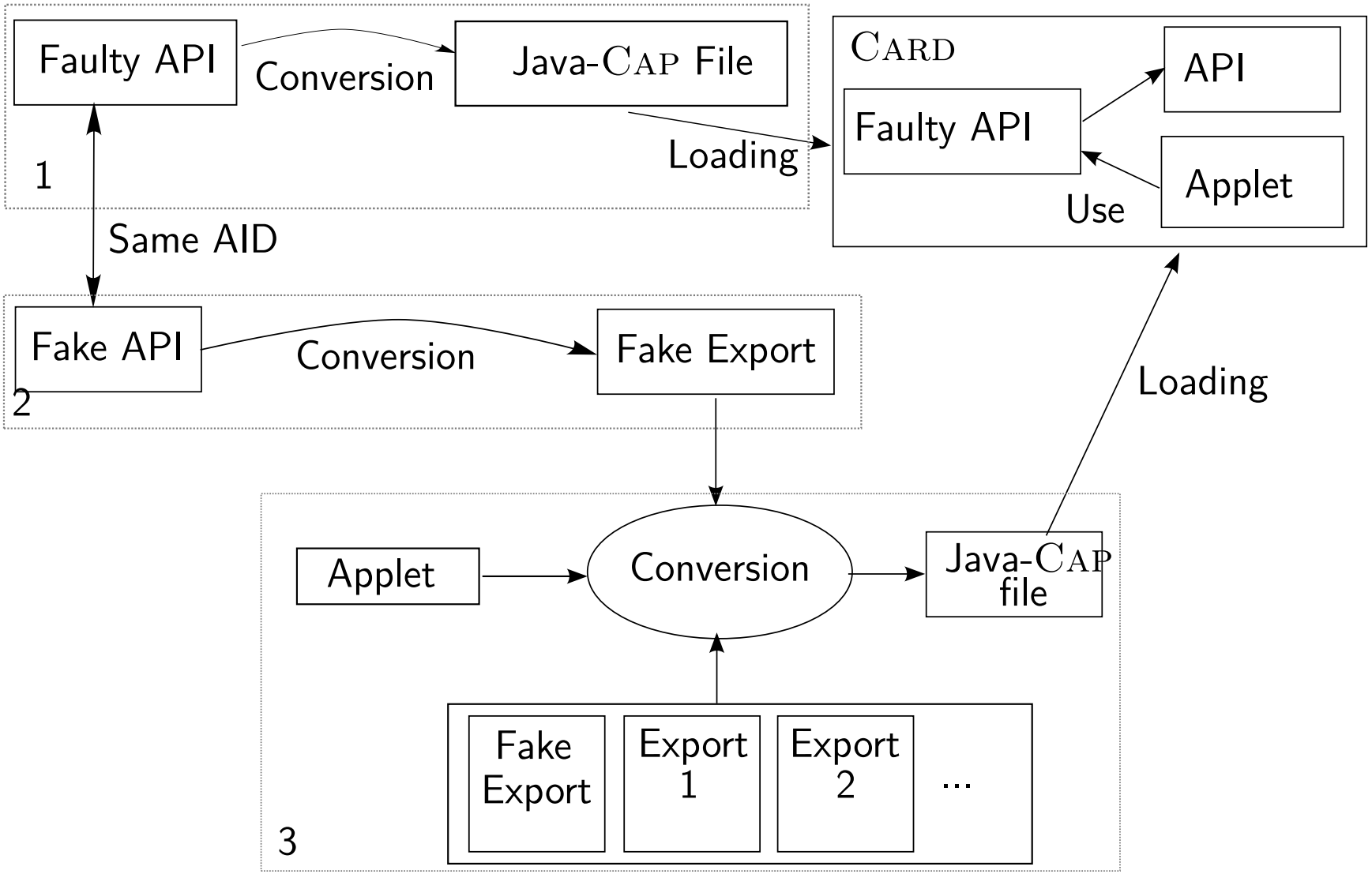
Principle


Applet
Fake API
API
Attacker's applet
buildKey
buildKey
Key
Store key
Key
getKeys
Keys stored

Modus Operandi I

1. A copy of API to confuse is developed:
   - Same classes' prototype;
   - Same methods' prototype;
   - Package's name?/AID?
2. The developer downloaded the fake EXPORT file:
   - The Java Card uses the *first find, first used* policy.
3. The Java-CLASS file to be converted is linked with our malicious EXPORT file
4. The Applet is linked with the malicious Java Card API.

Modus Operandi II



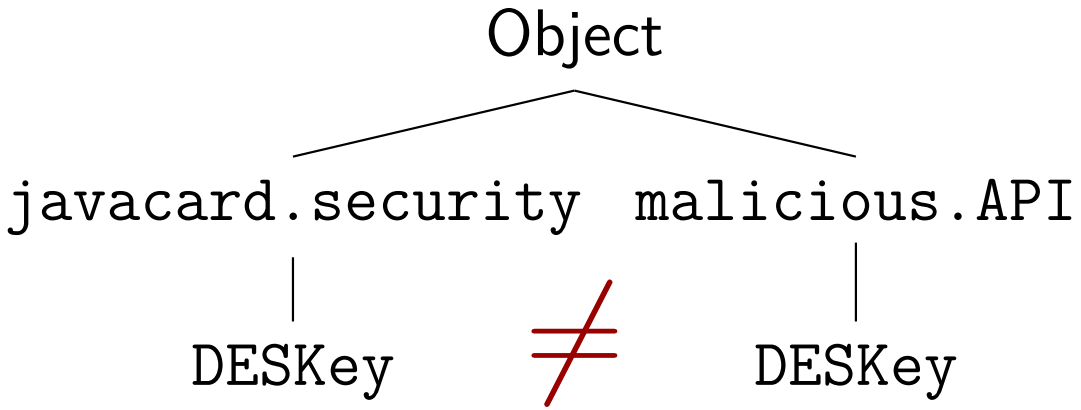
A Piece of a Java Card Crypto. Application

```
this.desKey = (javacard.security.DESKey)
  javacard.security.KeyBuilder.buildKey
    (KeyBuilder.TYPE_DES,   // key's type
     KeyBuilder.LENGTH_DES, // key's length
     true);                 // key value is encrypted
// DES Key initialization
this.desKey.setKey(DES_KEY_VALUE, //PIN code init.
                   OFFSET_DES_KEY_VALUE);
```

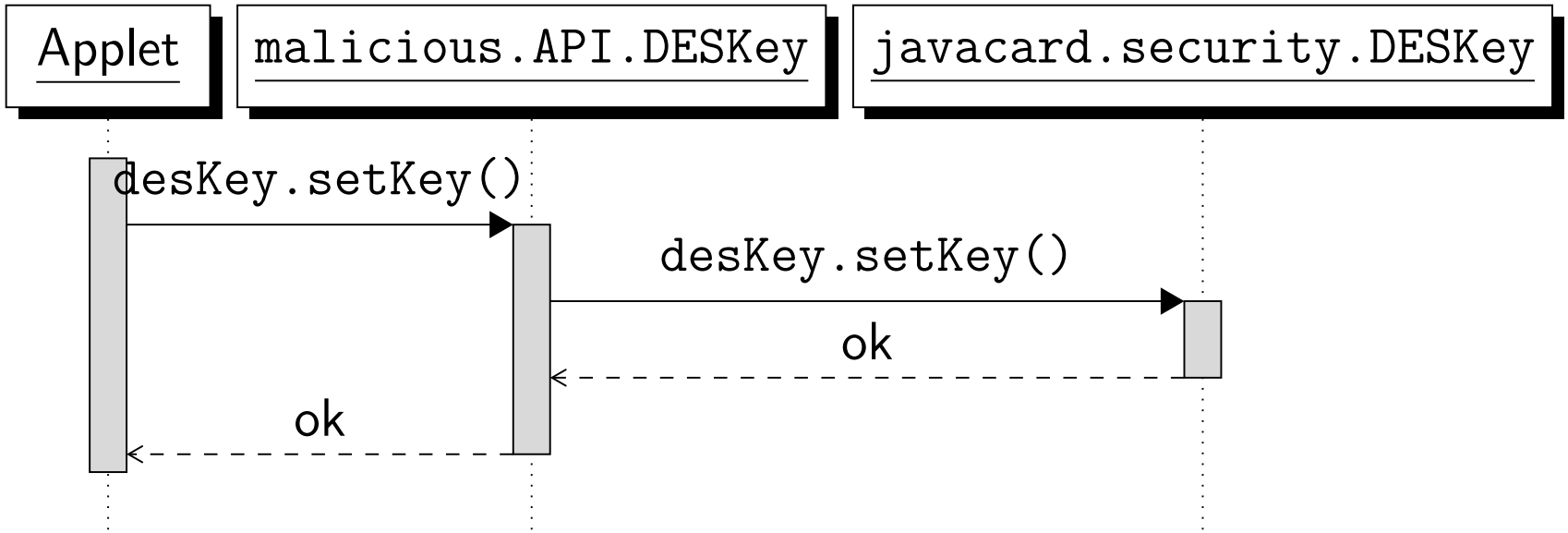
- Exploitation:
  - Develop a fake `javacard.security` EXPORT file;
  - The fragment of Crypto Application is linked with our malicious EXPORT file
  - Problem: Can the application be correctly executed?

A Piece of a Java Card Crypto. Application

```
this.desKey = (javacard.security.DESKey)
  javacard.security.KeyBuilder.buildKey
    (KeyBuilder.TYPE_DES,   // key's type
     KeyBuilder.LENGTH_DES, // key's length
     true);                 // key value is encrypted
// DES Key initialization
this.desKey.setKey(DES_KEY_VALUE, //PIN code init.
                   OFFSET_DES_KEY_VALUE);
```



How to Execute an Ill-Linked Applet? I



How to Execute an Ill-Linked Applet? II

```
public class DESKey extends malicious.API.DESKey {
 private javacard.security.DESKey desKey;

 // Default constructor
 public MyDESKey (javacard.security.DESKey desKey)
 { this.desKey = desKey; }

 // Implementation of the setKey function
 public void setKey(byte[] keyData, short kOff) {
    this.desKey.setKey(keyData, kOff);
 }

 // ...
}
```

The End

- What we did?
  - A Man-in-the-Middle attack on Java Card was presented;
  - The `javacard.security` API was exploited;
- How to prevent that?
  - Sign the EXPORT file!

Thank you for your attention!
Do you have any questions?

guillaume.bouffard@unilim.fr
http://secinfo.msi.unilim.fr