PV204 Security Technologies

3th assignment – Signing with smartcard

Supervisor: Dr.Petr Švenda

Student: Pedro Gomes, 490830

Due date: 28th March

Design and code changes

The changes I did are in the *SimpleAPDU.java* file where the main logic relies. I deleted all the demos and added only one function called: *assignment_flow().* Briefly, this function initializes a communication channel with the smart and asks the user/attacker the pin of the card (The one provided by the installed Applet, more on that soon.). The user/attacker has a maximum number of 5 attempts, for each attempt a command with the byte: *0x55* is sent to the card with the input pin. If a successful code: *9000* is returned, the user may proceed with the signature of the desired data.

In order to sign the data, another command is sent to the card:

```
CommandAPDU command_to_sign = new CommandAPDU(0xB0, 0x58, 0x00, 0x00,
message_to_sign_bytes);
```

After this, the response with the signed data is extracted and printed on the terminal. I also added a variable to the class: *CardManager.java*, that variable is a time variable (*long*), after quickly analyzing the function:

```
private void log(ResponseAPDU response, long time)
```

I noticed that the time taken by the card to execute a command is measured here, hence it was trivial to equal the class variable to the result of the time variable in this function, and then access it in the main logic, by doing so extracting the time the signature took in the card:

```
System.out.format("[>]Time taken for encryption on card: %d ms.", cardMngr.time);
```

Finally, and to conclude the description of my changes I did not change the Applet. Initially I did change the Applet, so it stays just with the code concerned with setting the new pin and the generation of the RSA-2048bits keys on the card. Unfortunately, if I removed the part of the code used to generated the AES key I was able to successful install the Applet on the card, BUT NOT to communicate with it, I was constantly getting the error: 0x6A82, which according to this source[1] means: *File not Found.* I cannot explained this error, at least at the time of writing.

Relevant attacker models and impact on your applet

An attack vector that I MUST mention here is the hard-coded pin on my Applet. If the attacker is able to extract the Applet code that is installed on the card, it would be trivial to recuperate the PIN, and sign the necessary data with it. This would lead to the useless of the PIN security check.

What I really wanted to do was, generate some random pin, like in the vanilla Applet, and somehow warn the user of the generated pin during the installation of the Applet on the card. I was not able to do this, or if this information was provided during the installation of the Applet, I was not able to dissect it. Hence, the only option I had was to hard-code the pin, so I would know what the pin was going to be in the future. I would like to emphasize that **this cannot be done in a real-world scenario.** I did it, but I am stating here what I did and the dangers associated with it, furthermore I spent several hours trying to implement my plan A, with the generated random PIN.

Something else that should also be said is the fact the PIN is stored in a primitive data type: *byte[]*, this would be less critical if the pin was randomly generated, but it is not. Thus, this might lead to an easier extraction of the hard-coded Pin stored in the card, as stated in the lecture 5 of PV204[2].

References
[1]**:** eftlab; Complete list of APDU responses;
https://www.eftlab.com/index.php/site-map/knowledge-base/118-apdu-response-list
[2]: Dr.Svenda Petr, JavaCard platform, PV204_05_Javacard_2019.pdf