

**HSM Project  
JavaCard Applet**

**Documentation**

# Table of Contents

1 Introduction.....	3
1.1 Design.....	3
1.2 Security.....	3
1.3 Applet.....	4
2 Implementation.....	5
2.1 Instructions.....	5
2.1.1 INS_SENDKEY (0x50).....	5
2.1.2 INS_CHANGEKEY (0x51).....	5
2.1.3 INS_SETPIN (0x52).....	5
2.1.4 INS_VERIFYPIN (0x53).....	5
2.1.5 INS_VERIFYPUK (0x54).....	6
2.1.6 INS_RUN (0x55).....	6
2.2 States.....	6
2.3 Summary.....	7
3 Installation and test.....	8
4 Security Domain and Secure channel.....	9

# 1 Introduction

For the purposes of the HSM project the JavaCard applet has to be implemented as the working key provisioner for the application. This document shall specify this part of the project in more detail as well as the implemented applet.

## 1.1 Design

The aim of the applet is to provide a cryptographic key that is used by the application. The key transmitting is made in a rather secure way, which consists of the following main features:

- Secure channel opened on JavaCard using three cryptographic keys.
- Key protected by a PIN (required from the application) and PUK (set during applet installation).
- Protected from running any applet functions by using automata-based programming approach.

All of these three concepts are introduced in the applet and are discussed in the following sections. The implementation of these features is described in the next chapter. The last two chapters present the usage of secure domain of Global Platform and scripts needed for an applet installation and testing.

## 1.2 Security

For the security of the overall project the secure channel must be initialized between the application and the applet. These are the goals of using such a secure channel:

- Mutual authentication of the JavaCard and the application.
- Integrity and confidentiality of APDUs.

All of the goals are reached by using the Security Domain that has been already installed inside the JavaCard. Using this it is only necessary to provide mandatory keys and open a secure channel between the card and the application. Keys can be sent into the card using GP Pro, while the secure channel can be open using the GP API. The channel is then open on the application side using functions from GP Pro. The communication is then sent through this channel to the Security Domain and then forwarded to the applet itself where it is unpacked.

More about the secure channel and how to provide it for the JavaCard is described later in this documentation. For the following text it is assumed that channel bridge already exists.

## 1.3 Applet

The applet provides a key for an application. For this reason it stores the key and provides functions for setting/getting this key out of the card. This key has to be protected from malicious users and the applet has to be protected from running unprivileged functions. PIN, PUK and automata-based programming are presented in the applet to reach the desired protection.

- PIN is used every time the user requires to send the key from card to the application. This PIN is firstly set by the user using the application. Then it can be changed only when verified with this PIN again.
- PUK is used when the user fails to verify by PIN. This can happen in a case that the user ran out of the available PIN tries. PUK is set during the installation of the applet and cannot be changed. It can also be used to change the card into “factory” state.
- Automata-based programming provides a way to call functions inside the applet in a secure way. There are several states inside the applet. Each state specifies what functions can be called. Example of such an approach is to have a factory state. The applet is in this state right after the applet has been installed. With this state set inside the card the user can send change, generate or send a key. But no other functionality is provided. For the PIN verification the other state has to be set. That is reached by calling a certain function. Using this approach the calling of functions is watched and thus is protected.

## 2 Implementation

Before describing the implementation details it is good to note that the entire project, as well as the applet was created in Java. The applet is built with JavaCard Kit 2.2.2 and Global Platform 2.1.1 library.

Before installing the applet the PUK has to be prepared. This PUK will be then set as a parameter of install command and then set to the PUK structure inside constructor of an applet. PUK is not changeable from the applet and thus has to be saved.

### 2.1 Instructions

The applet implements several instructions that can be run. To be able to run then it is also required to set the APDU class field to the value 0xB0, which specifies this applet.

#### 2.1.1 INS\_SENDKEY (0x50)

Instruction calls the function *sendKey* and is the response APDU will contain the key bytes. This function can be run only if the PIN was previous verified or the applet is in FACTORY state. If the applet is in the FACTORY state then the function does not change it. If the card is in AUTHORIZED state then the state is changed to NORMAL. The instruction does not accept any data or parameters.

#### 2.1.2 INS\_CHANGEKEY (0x51)

Instruction calls the function *changeKey* and does not returns any data. This function can be run only in FACTORY state and is performed when the key needs to be changed. The new key is either accepted in the APDU from the application – the parameter P1 is set to 0 – or is generated a new one using on-card RNG – the parameter P1 is set to 1.

#### 2.1.3 INS\_SETPIN (0x52)

Instruction calls the function *setPIN*. It requires data to represent the PIN that is to be set and can be run only when the applet is in the state SETUP or AUTHORIZED. If the state is SETUP, then there was no previous PIN set and thus needs to be set. If the state was AUTHORIZED then PIN was already set and has to be verified to change. The function changed the state of the applet to NORMAL.

#### 2.1.4 INS\_VERIFYPIN (0x53)

Instruction calls the function *verifyPIN*. This instruction can be run only when the applet is in state NORMAL. It requires data to be sent in the APDU that represents the PIN being verified. The function can then change the state to either AUTHORIZED or FAILED. The former of the two is set when verification passed successfully, while the later one is set when PIN tries are exhausted. After successful verification the PIN counter is reseted.

### 2.1.5 INS\_VERIFYPUK (0x54)

Instruction calls the function *verifyPUK*. This instruction can be run only when the applet is in the state FAILED or NORMAL. In the case of former one it means that the verification of PIN has failed and user has to authorize with PUK. If the state of applet is NORMAL then user has to set PUK and if the verification is successful then the state is changed to FACTORY, otherwise the card is locked.

### 2.1.6 INS\_RUN (0x55)

Instruction calls the function *run*. This function is called whenever the card has been configured and is about to be used. This function changes the state from FACTORY to SETUP and prepares the card to receive and set the first PIN.

## 2.2 States

There are several states of the applet. These states are implemented as constant values which are assigned to a state variable during the applet operation. The following states are available:

- **FACTORY** – represents the state of the applet after the installation. In this state the user can change or generate a new key as well as send it out. It is also convenient to have an option to get into this state once user needs to change the key – after successful PUK verification, applet state is changed to FACTORY (this holds only when the previous state was NORMAL).
- **SETUP** – once the function *run* is called and successfully ended, the state changes to SETUP. This is the state when PIN needs to be set for the first time before applet can operate normally. This state can be reached only from the FACTORY state.
- **NORMAL** – after the PIN was set the state is changed from SETUP to NORMAL. This state represents a normal operating mode of the applet and during this state functions such as *verifyPIN* and *verifyPUK* can be run.
- **AUTHORIZED** – once the application successfully verifies with the PIN the applet changes to the this state and the function *sendKey* can be called to transmit the key to the application. After this operation the state changes back to NORMAL.
- **FAILED** – if the verification of PIN fails due to exhausted PIN tries the state FAILED is set. During this state the user needs to verify with PUK. If this verification is successful then the state is changed to AUTHORIZED and counter of the PIN and PUK are reseted.
- **LOCKED** – after user fails with PUK verification the state is changed to LOCKED. This state is not possible to change and is tested in the *select* function of applet – applet will not be selected.

The automaton with described states is described in the following figure.

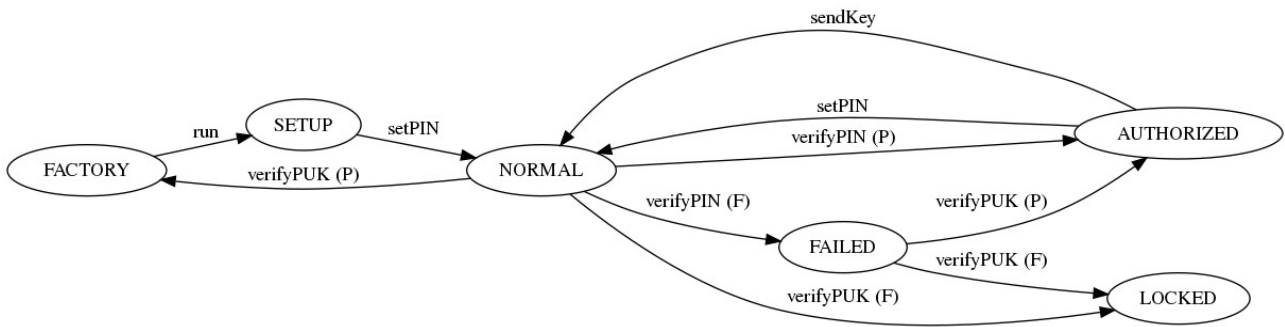


Figure 1: Automaton with states of the applet. Notice it does not display loops – only functions changing state.

## 2.3 Summary

The following table summarize all the instructions and their relation to the applet.

INS_	Code	P1	P2	INPUT	DATA IN	OUTPUT	DATA OUT	From state	To state
SENDKEY	0x50	-	-	N	-	Y	key	AUTH,FACT	NORM,FACT
CHANGEKEY	0x51	0/1	-	Y	-/key	N	-	FACT	FACT
SETPIN	0x52	-	-	Y	PIN	N	-	SETUP,AUTH	NORM
VERIFYPIN	0x53	-	-	Y	PIN	N	-	NORM	AUTH
VERIFYPUK	0x54	-	-	Y	PUK	N	-	NORM,FAIL	AUTH,FACT
RUN	0x55	-	-	N	-	N	-	FACT	SETUP

### **3 Installation and test**



## **4 Security Domain and Secure channel**