FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Inteligentní bezpečnostní systém – sekce zabezpečený vstup
Student:	Maroš Mačak
Vedoucí:	Ing. Martin Daňhel, Ph.D.
Studijní program:	Informatika
Studijní obor:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Pro funkční "Inteligentní bezpečnostní systém garáže" (dále jen IBSG), který byl obhájen na této fakultě v červnu 2018, vytvořte zabezpečení vstupních dveří pomocí čipové SMART karty, dle těchto pokynů:
1. Seznamte se s obhájenou BP M. Váni, který se zabýval tvorbou IBSG.
2. Analyzujte možnosti implementace autentizace pomocí symetrické kryptografie na platformě Java Card.
3. Navrhněte systém bezpečnostního vstupu založený na bezkontaktních čipových kartách Java Card s použitím jednoduché jednosměrné symetrické
autentizace čipové karty a zvolte vhodnou metodu ukládání klíče na kartě.
4. Navrhněte vnitřní logiku systému včetně komunikačního rozhraní s IBGS a programu na čipové kartě včetně způsobu ukládání klíče.
5. Navržený systém implementujte a otestujte.
6. Testováním odhalte a popište různé druhy možných útoků a vytvořte profil útočníka.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Pavel Tvrdík, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 14. února 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Inteligentný bezpečnostný systém – sekcia zabezpečený vstup

Maroš Mačak

Katedra počítačových systémů
Vedúci práce: Ing. Martin Daňhel, Ph.D.

13. mája 2019

Poďakovanie

Chcel by som sa poďakovať všetkým, ktorí mi pomohli pri tvorbe tejto práce. V prvom rade vedúcemu práce Ing. Martinovi Daňhelovi, Ph.D. za rady a tipy pri tvorbe. Ďalej Ing. Jiřímu Bučkovi, Ph.D. za vecné pripomienky k práci. Nakoniec mojej rodine za podporu počas štúdia a za pomoc pri tvorbe tejto práce.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 13. mája 2019 .....

Odkaz na túto prácu

Mačak, Maroš. *Inteligentný bezpečnostný systém – sekcia zabezpečený vstup.* Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Cieľom tejto práce je navrhnúť a implementovať zabezpečenie vstupných garážových dverí pomocou čipovej karty. Systém je navrhovaný pre už existujúci systém zabezpečenia garáže, ktorý bol implementovaný na platforme Arduino. Pre implementáciu bola zvolená jednoduchá symetrická autentizácia využívajúca šifrovací algoritmus Advanced Encryption Standard (AES) s dĺžkou kľúča 128 bitov v móde cipher block chaining (CBC) a platforma pre čipové karty bola zvolená Java Card. Prínosom tejto práce je zhrnutie a popis najpoužívanejších platforiem pre vývoj programov na čipové karty, analýza dostupných radio frequency identification (RFID) čítačiek pre Arduino a popísanie druhov útokov na čipové karty. Výsledkom je systém využívajúci čítačku Adafruit PN532, ktorý je schopný autentizovať karty a je stabilný a odolný voči jednoduchým útokom.

Klúčové slová Arduino Yún, IoT, Java Card, PN532, RFID, symetrická autentizácia

Abstract

The aim of this thesis is to design and implement security system for garage entrance doors using smart cards. The system is designed for already existing security system for garages which was implemented on Arduino platform. For the implementation a simple symmetric authentication was used with Advanced Encryption Standard (AES) using the key length of 128 bites in a cipher block chaining (CBC) mode. And for the smart cards Java Card was used. The main contributions of this thesis are the descriptions of the most used platforms for the development of the smart cards, analysis of available radio frequency identification (RFID) readers for Arduino and descriptions of attacks on smart cards. As a result this thesis offers a security system using Adafruid PN532 reader which is able to authenticate cards, is stable and resistant against simple attacks.

Keywords Arduino Yún, IoT, Java Card, PN532, RFID, symmetric authentication

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

V posledných rokoch sa využitie čipových kariet stalo metódou jednoduchej a efektívnej autentizácie. V najväčšej miere sa dnes využívajú v bankových kartách a SIM kartách. Vďaka ich nízkej cene môžu byť využité aj napríklad ako karty na obed, členské karty alebo karty na vstup do rôznych budov. S príchodom a stále väčším rozšírením internet of things (IoT) využitie čipových kariet bude rásť ešte viac. S tým samozrejme rastie hrozba ich jednoduchého zneužitia pri zlej implementácii.

Túto tému som si vybral preto, lebo téma čipových kariet a konkrétne Java Card ma na štúdiu odboru zaujala najviac. Zároveň to pre mňa bola príležitosť naučiť sa programovať pre platformu Arduino a získať skúsenosti v odbore internet of things.

Táto práca naväzuje na obhájenú prácu *Inteligentní bezpečnostní systém garáže* (ďalej už len IBSG), v ktorej bol vytvorený bezpečnostný systém garáže na platforme Arduino Yún schopný monitorovať stav garáže pomocou senzorov. V predchádzajúcej práci bolo pre jednoduchosť a nedostatok času odomykanie a zamykanie garáže vyriešené iba pomocou tlačidla. V tejto práci bude systém IBSG rozšírený o zabezpečený vstup pomocou čipových kariet založených na technológii Java Card 2.4.

Úvod tejto práce tvorí popis IBSG a je vymedzené, čo sa v tejto práci bude implementovať 2.1. Nasleduje analýza možností implementácie autentizácie pre čipové karty 2.2. Ďalej v tejto kapitole sú popísané dostupné neproprietárne operačné systémy pre čipové karty 2.3. Podľa tejto analýzy je zvolená najvhodnejšia platforma, ktorá je opísaná detailnejšie. Následne sú popísané druhy autenizácie a podľa toho je vybratá vhodná pre túto implementáciu 2.5. Záver tejto kapitoly obsahuje zoznam dostupný RFID čítačiek pre Arduino, ich opis a zvolenie tej, ktorá bude použitá pre implementáciu 2.6.

Kapitola Realizácia zabezpečenia vstupu sa delí na tri časti: Rozšírenie systému IBSG 3.1, Návrh logiky karty 3.2 a Komunikačné protokoly 3.3.

Prvá časť tejto kapitoly začína vysvetlením spôsobu rozšírenia systému IBSG a schémou zapojenia čítačky do Arduina. Potom nasleduje popis a funkcionalita pridaných súborov a u niektorých aj dôležité časti kódu. Záver tejto časti tvoria problémy, ktoré nastali pri implementácii a ich riešenie. Druhú časť tejto kapitoly tvorí opis implementácie appletu na karte, ukážka dôležitého kódu a komplikácie pri implementácii a ich riešenie. Poslednú časť tejto kapitoly tvorí zoznam REST príkazov, ktoré boli pridané do systému, zoznam APDU definovaných pre komunikáciu s kartou a zoznam status words, ktoré môžu nastať pri odpovedi na tieto APDU.

Úvod kapitoly Testovanie sa venuje testovaniu kúpených čítačiek 4.1. Potom sú vymedzené prípady použitia a na základe nich je systém otestovaný, či pridaná funkcionalita je funkčná a či nespôsobila nefunkčnosť 4.2.

Posledná kapitola začína profilom útočníka. Nasleduje spôsob delenia útokov na čipové karty 5.1 a opis častých útokov 5.2. Záver kapitoly tvoria príklady na konkrétne útoky a do akej miery je systém voči nim odolný 5.3.

Kapitola 1

Cieľ práce

Cieľom tejto práce bude rozšírenie systému IBSG o zabezpečenie vstupu pomocou čipovej karty. Autentizácia bude jednosmerná a symetrická. Prínosom tejto práce bude detailný opis zvolenej platformy, rozbor možností autentizácie na čipových kartách a opis a testovanie RFID čítačiek pre Arduino. Tým vznikne podrobný návod na implementáciu zabezpečenia vstupu pomocou čipových kariet.

Zvolená implementácia bude využívať najnovšie technológie a bude implementovaná s dôrazom na stabilitu a bezpečnosť. Stabilita bude zaručená dostatočným testovaním. Testy vzniknú na základe vymedzenia prípadov použitia. Na dosiahnutie bezpečnosti bude v práci vypracovaná kapitola venujúca sa útokom na čipové karty a spôsobe obrany voči nim. Na základe tejto kapitoly bude zrejmé, voči akým útokom a do akej miery je systém odolný.

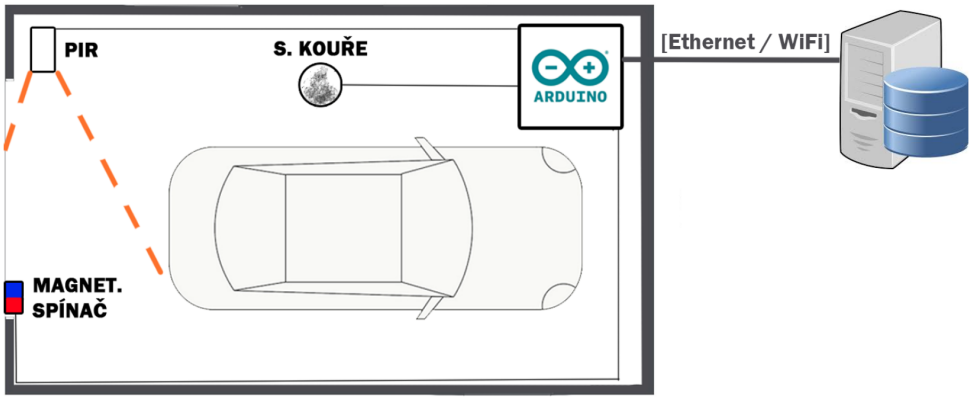
Kvôli rozsahu zadania práce určite nastanú problémy, ktorých riešenie je mimo rozsah tejto práce alebo ich implementácia je zložito realizovateľná. V takom prípade bude opísané, ako by sa daný problém dal vyriešiť.

Kapitola 2

Analýza možností implementácie

2.1 Inteligentný bezpečnostný systém garáže

Táto práca nadväzuje na bakalársku prácu Miroslava Váňu obhájenú v akademickom roku 2017/2018 s názvom *Inteligentní bezpečnostní systém garáže* [1]. V tejto práci bol vytvorený bezpečnostný systém garáže na platforme Arduino Yún schopný monitorovať stav garáže pomocou senzorov. Systém obsahoval detektor pohybu, detektor dymu a detektor otvorenia dverí. Každý z týchto senzorov bol v systéme dvakrát, aby sa zabránilo falošným poplachom. Získané dáta zo senzorov systém spracoval, vyhodnotil a na základe toho vykonal príslušnú akciu. V tejto práci bolo implementované rozhranie REST slúžiace na komunikáciu s Arduinom. Pre jednoduchosť a nedostatok času bolo v tejto práci odomykanie a zamykanie garáže vyriešené iba pomocou tlačidla. Pôvodný návrh systému je zobrazený na nasledujúcom obrázku:



Obr. 2.1: Pôvodný návrh systému v priestore garáže. Zdroj: [1]

Aby bolo možné tento systém rozšíriť o zabezpečenie vstupu, je nutné zvoliť platformu pre vývoj na čipových kartách, druh autentizácie a samotnú RFID čítačku pre Arduino. Takisto bude nutné vybrať metódu na odvádzanie špecifických kľúčov pre karty. Tieto témy sú spracované v samostatných podkapitolách tejto kapitoly.

V tejto práci nebude implementované žiadne grafické používateľské rozhranie a komunikácia s Arduinom bude len pomocou už implementovaného REST rozhrania. To bude musieť byť rozšírené o príkazy súvisiace s pridanou funkcionalitou.

2.2 Čipové karty

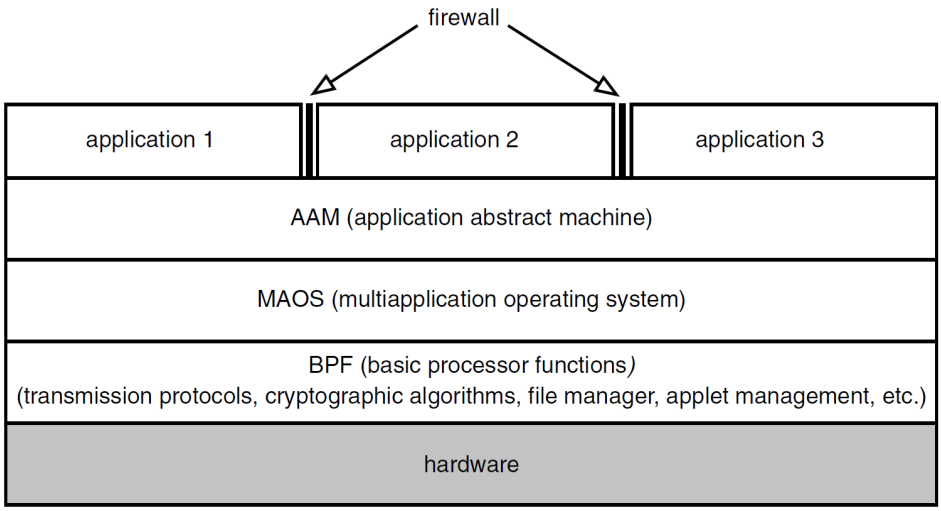
Pod pojmom čipová karta si môžeme predstaviť plastovú kartu obsahujúcu integrovaný obvod. Vo vnútri karty s mikročipom je procesor schopný spustenia jednoduchých operačných systémov a ko-procesor poskytujúci základné kryptografické funkcie. Vo väčšine prípadov sa čipové karty používajú na autentifikáciu. Čipové karty sa v niektorej literatúre nazývajú aj smart karty. Tento názov pochádza z angličtiny. V tejto práci budem používať zaužívanejší názov čipové karty.

2.3 Voľba platformy pre čipové karty

Na kartách obsahujúcich mikročip musí bežať operačný systém, ktorý bude spracovávať a vyhodnocovať požiadavky na kartu. Tieto platformy delíme na proprietárne a takzvané open platform. O proprietárnych platformách platí, že sú to riešenia istej firmy, ktorých špecifikácie nie sú známe, a teda ich modifikácia nie je možná. Na druhej strane open platformy dovoľujú beh aplikácií tretích strán bez nejakého zásahu vydavateľa operačného systému [2]. V tejto práci sa budem venovať iba open platformám. V tejto sekcii popíšem dve najpoužívanejšie platformy.

2.3.1 Platforma MULTOS

MULTOS je open platform operačný systém pre čipové karty. Programy na tejto platforme sú písané buď v C alebo v Jave a následne sú skompilované do MULTOS Executable Language, ktorý je spustený na virtuálnom stroji. To zabezpečí, že programy môžu byť spustené na kartách od rôznych výrobcov. Tento virtuálny stroj zaručuje, že každá spustená inštrukcia je správna. Nie je teda možné, aby aplikácia pristupovala k dátam inej aplikácie. Ak dôjde k spusteniu nesprávnej inštrukcie, beh programu je zastavený. MULTOS poskytuje aj takzvanú MULTOS Step/one, čo je lacnejšia verzia platformy. Tá sa používa na implementáciu aplikácií, ktoré nevyžadujú kryptografický koprocesor. Opis platformy je prevzatý zo stránky MULTOS [3].



Obr. 2.2: Architektúra platformy MULTOS. Zdroj: [2]

2.3.2 Platforma Java Card

Podľa webovej stránky Oracle [4] sa dá povedať, že táto platforma je v súčasnosti najpoužívanejšia. Používa sa v SIM kartách, platobných kartách, občianskych preukazoch, cestovných pasoch atď. Každý rok sa vyrobí a naprogramuje okolo dvoch miliárd kariet. V roku 2011 50 % SIM kariet používalo technológiu Java Card. Túto platformu som si zvolil pre implementáciu a viac sa jej budem venovať v samostatnej sekcii 2.4.

2.3.3 Rozdiely medzi MULTOS a Java Card

Čo sa týka rozšírenosti, JavaCard je rozšírenejšia a viac používaná platforma. Jedným z dôvodov je jej nízka cena a viacej výrobcov. Pre vývoj na Java Card stačí iba karta a čítačka kariet, všetky nástroje na vývoj sú open source a zadarmo. MULTOS je menej používaný, no podľa samotnej firmy je viac bezpečný a je preferovaný v sfére vlády. Pre platformu MULTOS je možné na rozdiel od Java Card vyvíjať aj v iných jazykoch než v Jave, no pre vývoj je nutné zakúpiť nástroje. Porovnanie je prevzaté zo stránky MULTOS [5].

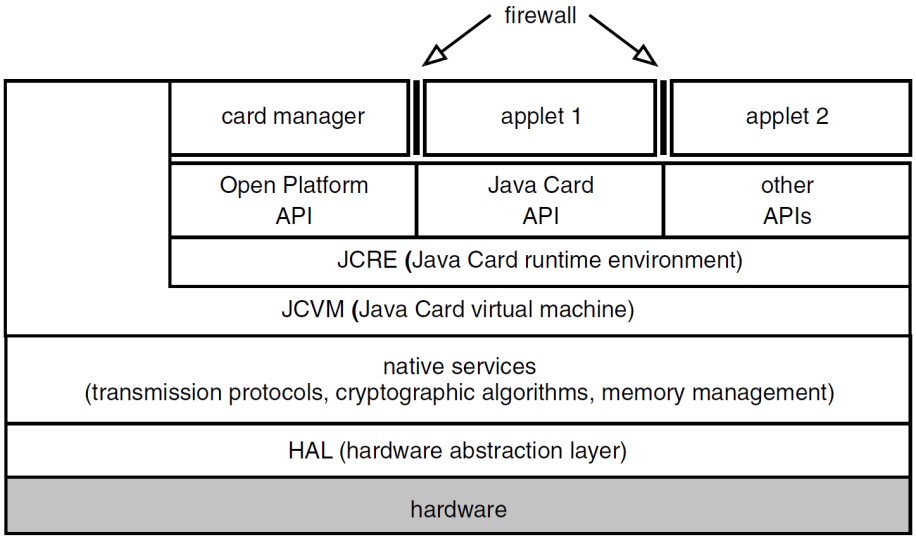
2.4 Platforma Java Card

Platforma Java Card umožňuje bezpečný beh Java aplikácií (takzvané applety) primárne na čipových kartách. Hlavnou výhodou platformy Java Card je prenosnosť a technológia je kompatibilná s už existujúcimi štandardami pre čipové karty. Applety vyvinuté Java Card technológiou je teda možné spustiť na akejkoľvek čipovej karte podporujúcej túto technológiu nezávisle

na výrobcovi alebo hardvéri. To je zaručené podobne ako pri iných Java technológiách pomocou virtual machine (Java Card Virtual Machine). Jediným limitom sú obmedzenia kariet v podpore komunikačných protokolov alebo šifrovacích algoritmov. Popis platformy je prevzatý zo stránky Oracle [6].

2.4.1 Architektúra

Architektúra Java Card sa skladá z viacerých častí. Na najvyššej úrovni sa nachádza applet, buď Java Card alebo z inej platformy. Tieto applety sú od seba oddelené firewallom. Java Card applet pristupuje cez Java Card API ku virtual machine, ktorá ďalej komunikuje s hardvérom [2].



Obr. 2.3: Architektúra platformy Java Card. Zdroj: [2]

2.4.2 Pamäť

Čipová karta má dva druhy pamäte, perzistentnú a transientnú. Dáta v perzistentnej pamäti su uložené v EEPROM, ktorá ma zvyčajne kolo 64 KB. Takisto sa sem ukladá halda a samozrejme aj dáta appletu. Transientná pamäť je uložená v operačnej pamäti a má zvyčajne okolo 1 KB. Premenné na perzistentnú pamäť sa vytvárajú statickou deklaráciou alebo cez *new* a na transientnú pomocou *JCSystem.makeTransientByteArray()*[1].

[1]Zdroj: Ing. Jiří Buček Ph. D., Prednáška 1 BI-HWB, 2019, [cit. 2019-03-09], dostupné z Moodle FIT ČVUT, po prilásení

2.4.3 APDU

Každý applet má svoj unikátny application identifier (ďalej už len AID), ktorý slúži na rozlíšenie appletov. Informácie medzi kartou a terminálom sa posielajú vo forme application data protocol unit (ďalej už len APDU). Štruktúra APDU je definovaná podľa ISO 7816 [7] štandardu nasledovne:

Hlavička (povinná)				Telo (nepovinné)		
CLA (1)	INS (1)	P1 (1)	P2 (1)	Lc (0 – 3)	Data	Le (0 – 3)

Tabuľka 2.1: APDU príkazu. Čísla v zátvorke znamenajú veľkosť v bajtoch[1].

CLA Inštrukčná trieda, slúži na identifikovanie typu príkazu, napríklad či APDU je podľa spomínaného štandardu alebo jeho štruktúra je proprietárna.

INS Inštrukčný kód, slúži na identifikovanie konkrétneho príkazu. Inštrukčné kódy pre základné operácie sú definované štandardom. Takisto je možné si zadefinovať aj vlastné inštrukčné kódy.

P1 a P2 Parametere príkazu, pomocou nich je možné príkazu nastaviť parametre a ovplyvniť jeho chovanie.

Lc Počet bajtov dát, určuje koľko bajtov budú mať dáta.

Data Obsah správy.

Le Počet bajtov očakávanej odpovedi, určuje koľko bajtov by mala mať korektná odpoveď.

Telo (voliteľné)	Zápätie (povinné)	
Data	SW1 (1)	SW2 (1)

Tabuľka 2.2: APDU odpovede. Čísla v zátvorke znamenajú veľkosť v bajtoch[1].

Data Obsah odpovede.

SW1 a SW2 Status kódy, slúžia na identifikáciu návratovej hodnoty, napríklad pri úspechu alebo neúspechu príkazu.

[1]Zdroj: Ing. Jiří Buček Ph. D., Prednáška 1 BI-HWB, 2019, [cit. 2019-03-09], dostupné z Moodle FIT ČVUT, po prihlásení

2.4.4 UID

Unique identifier (ďalej už len UID) je 4, 7 alebo 10 bajtový identifikátor kariet. Poskytnuté karty na vývoj *Gemalto GemCombiXpress R4 72K* majú UID dĺžky 4 B. Tento identifikátor bude použitý na identifikáciu kariet a bude z neho odvodený špecifický kľúč pre kartu.

2.4.5 Jazyk Java Card

Java Card je obmedzený Java jazyk s dôrazom na beh na zariadeniach s malým množstvom pamäte. Medzi základné rozdiely patrí, že jazyk Java Card[1]:

- nepodporuje základné dátové typy, napríklad: *int*, *char*, *string*, *float*, *double*,
- podporované su iba *byte* (8 b) a *short* (16 b),
- neobsahuje garbage collector,
- nepodporuje viacrozmerné polia, vlákna, klonovanie tried a objektov.

2.4.6 Štruktúra appletu

Každý applet (.java súbor) musí dodržiavať nasledujúcu štruktúru:

```
package example;

import javacard.framework.*;

public class example extends Applet {

  public static void install(byte[] bArray,
                             short bOffset,
                             byte bLength) {
    new example();
  }

  protected example() {
    register();
  }

  public void process(APDU apdu) {

  }
}
```

Funkcia *install()* sa zavolá vždy pri inštalácii appletu na kartu. Zvyčajne sa tu naalokujú premenné a zavolá sa konštruktor appletu. Funkcia *process()* sa zavolá, keď je karta napájaná a príde jej APDU.

[1]Zdroj: Ing. Jiří Buček Ph. D., Prednáška 1 BI-HWB, 2019, [cit. 2019-03-09], dostupné z Moodle FIT ČVUT, po prilásení

2.4.7 Balíčky Java Card

Pre vývoj appletov na Java Card sú k dispozícii štyri balíčky obsahujúce API pre funkcie potrebné na činnosť čipových kariet.

java.lang

Keďže Java Card je jazyk Java, pre jeho vývoj je nutný balíček *java.lang* [8]. Tento balíček obsahuje základné triedy ako:

Názov triedy	Popis triedy
Object	definuje všetky triedy v Java Card
Exception	slúži na správu výnimiek
Throwable	obsahuje všetky výnimky a chyby pre Java Card

Tabuľka 2.3: Balíček java.lang. Zdroj: [8]

javacard.framework

Pre vývoj Java Card appletov je nutný balíček *javacard.framework* [9]. Tento framework poskytuje minimálnu základnú funkcionalitu, teda triedy a rozhranie pre vývoj a komunikáciu s appletmi. Medzi dôležité triedy balíčka patrí:

Názov triedy	Popis triedy
AID	obsahuje AID karty
APDU	poskytuje metódy na výmenu informácií medzi čipovou kartou a terminálom na úrovni APDU
Applet	definuje applet karty
ISO7816	obsahuje rôzne konštanty pre prácu s APDU, napríklad návratové hodnoty
JCSystem	obsahuje metódy slúžiace na vytváranie dočasných a perzistentných objektov a na povoľovanie viacerým appletom pristupovať k jednému objektu

Tabuľka 2.4: Balíček javacard.framework. Zdroj: [9]

javacard.security

Ďalší dôležitý balíček je *javacard.security* [10]. Ten obsahuje rozhrania a triedy slúžiace pre implementáciu kryptografickej funkcionality na platforme Java Card, ako napríklad generovanie kľúčov, hashovanie, generácia náhodných dát apod. Tento balíček obsahuje triedy ako:

Názov triedy	Popis triedy
Key	obsahuje základné rozhranie pre všetky druhy kľúčov
AESKey	poskytuje rozhranie ku AES
KeyBuilder	slúži na generovanie kľúčov
MessageDigest	trieda pre všetky hashovacie funkcie
RandomData	trieda na základné generovanie náhodných čísel

Tabuľka 2.5: Balíček javacard.security. Zdroj: [10]

javacardx.crypto

Posledný potrebný balíček je *javacardx.crypto* [11]. Ten v sebe obsahuje implementáciu podporovaných kryptografických funkcií. Tento balíček obsahuje triedy ako:

Názov triedy	popis triedy
KeyEncryption	rozhranie poskytujúce prístup ku šifrovaným kľúčom
Cipher	trieda pre všetky šifrovacie algoritmy

Tabuľka 2.6: Balíček javacardx.crypto. Zdroj: [11]

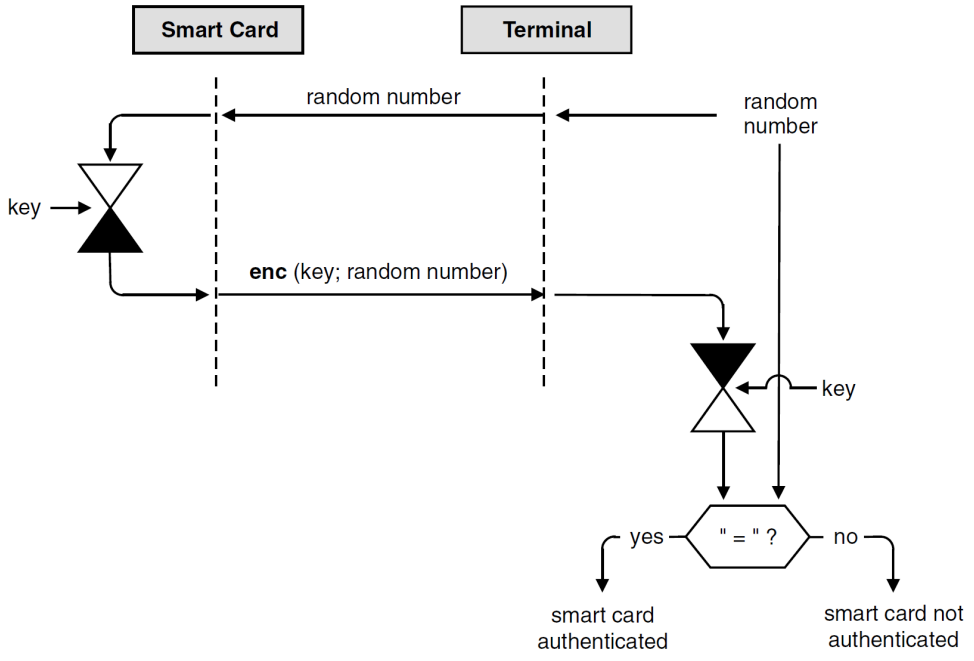
2.5 Protokol autentizácie

Hlavným zdrojom informácií pre túto podkapitolu je kniha *Smart Card Handbook* [2]. Autentizácia je definovaná ako spôsob overovania identifikácie a pravosti druhej strany pri komunikácii. Autentizácia vyžaduje, aby obe komunikujúce strany odoslali spoločný kľúč, ktorý môže byť overený autentizačnými metódami [2].

Pri autentizácii sú dôležité tri faktory: typ algoritmu, účastníci a metóda. Typy algoritmov sa delia na symetrické a asymetrické. Symetrický šifrovací algoritmus je taký, ktorý na šifrovanie a dešifrovanie používa rovnaký kľúč. Naopak asymetrický algoritmus je taký, ktorý používa rôzne kľúče na šifrovanie a dešifrovanie. Táto práca sa bude venovať iba symetrickým druhom autentizácie. Pri účastníkoch rozlišujeme, či je autentizácia jednosmerná alebo obojsmerná. Pri jednosmernej autentizácii sa vždy autentizuje len jedna strana druhej (napríklad karta čítačke) a pri obojsmernej sa autentizujú obe strany. Metódu autentizácie delíme na statickú a dynamickú. Pri statickej autentizácii sa vždy používajú rovnaké dáta, napríklad meno a heslo. Dynamická autentizácia sa odlišuje tým, že sa vždy pri nej používajú iné dáta, napríklad náhodné číslo [2].

2.5.1 Jednosmerná autentizácia

Jednosmerná symetrická autentizácia začína tým, že terminál vygeneruje náhodné číslo, ktoré pošle karte. Karta toto číslo zašifruje kľúčom, ktorý je známy karte aj terminálu a odošle ho naspäť terminálu. Terminál si túto správu rozšifruje a porovná obsah správy s vygenerovaným číslom. Ak sa rovnajú, karta pozná kľúč, teda je prehlásená za vierohodnú. V praxi sa často okrem náhodného čísla pridávajú do kryptogramu aj iné informácie, ako napríklad verzia protokolu. Táto metóda autentizácie je odolná voči replay útokom, pretože pri každej autentizácii sa šifruje/dešifruje náhodné číslo. Nevýhodou tejto autentizácie je, že kľúč pre kartu aj pre terminál je rovnaký. Ak teda bude prezradený, tak sú všetky karty ohrozené. Aby sa tomu zabránilo, terminál musí od karty vyžiadať jej unikátne číslo, z ktorého odvodí špecifický kľúč pre danú kartu. Také číslo je často odvodené od sériového čísla karty alebo od iného unikátneho čísla karty. Proces generovania špecifického kľúča je nejaká funkcia, ktorá vezme číslo karty a hlavný kľúč, ktorý pozná iba terminál. V konečnom dôsledku prezradenie hlavného kľúča spôsobí ohrozenie celého systému, a preto by mal byť uložený veľmi bezpečne. Ak je to možné, tak by mal byť aj zmazaný pri útoku [2]. Celý tento proces (okrem odvodzovania kľúča) je znázornený na nasledujúcom obrázku:



Obr. 2.4: Diagram jednosmernej symetrickej autentizácie. Poznámka: *enc* značí šifrovaciu funkciu. Zdroj: [2]

Odvodenie kľúča

Na odvodenie špecifického kľúča karty od hlavného je v tejto práci použitý Keyed-Hashing for Message Authentication (ďalej už len HMAC). Message Authentication (MAC) poskytuje spôsob, ako overiť integritu dát, ktoré boli prenášane cez nedôveryhodné médium. HMAC využíva kryptografické hashovacie funkcie spolu s tajným kľúčom, čím teda zaručuje nie iba integritu dát, ale aj autentizáciu. HMAC je podľa štandardu RFC 2014 [12] definovaný nasledovne:

$$HMAC(K, m) = H((K \oplus opad)||H((K \oplus ipad)||m))$$

K tajný kľúč

m správa

H hashovacia funkcia

opad padding, bajt 0x36 opakovaný B-krát

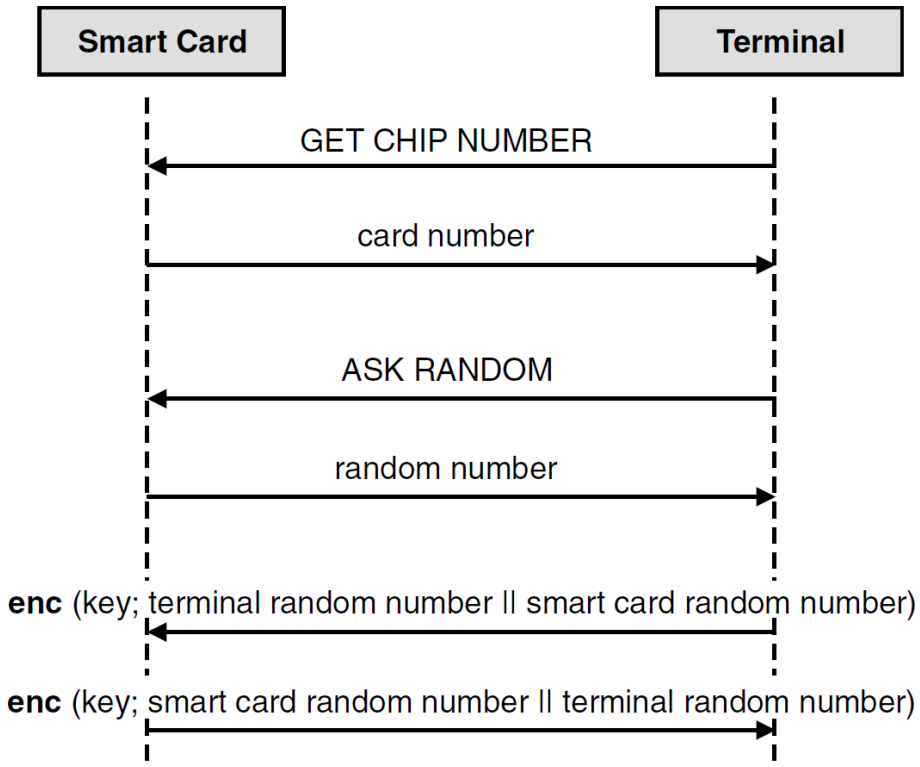
ipad padding, bajt 0x5C opakovaný B-krát

Poznámka: B značí dĺžku bloku v hashovacej funkcii.

Pre implementáciu v tejto práci je teda K hlavný kľúč uložený v IBSG, m je číslo karty a ako hashovacia funkcia bola zvolená SHA-256.

2.5.2 Obojsmerná autentizácia

Na začiatku obojsmernej symetrickej autentizácie terminál vyžiada od karty jej číslo. Toto číslo je unikátne pre kartu a terminál z neho odvodí autentizačný kľúč pre túto kartu. Následne od karty vyžiada náhodné číslo a sám terminál náhodné číslo vygeneruje. Terminál potom vymení poradie týchto čísel, zlúči ich dohromady, zašifruje odvodeným kľúčom a tento kryptogram pošle karte. Prijatú správu karta rozšifruje a skontroluje, či sa číslo z rozšifrovanej správy zhoduje s náhodným číslom, ktoré vygenerovala pri predchádzajúcej požiadavke. Ak sa zhoduje, karta vie, že terminál pozná kľúč, teda je pravý. Potom karta tieto čísla znova vymení, zašifruje a pošle terminálu. Terminál správu rozšifruje a skontroluje náhodné čísla. Ak sa zhodujú, terminál prehlási kartu za pravú a tým sa celý proces autentizácie ukončí. Dôvodom výmeny čísel pri posielaní správy je, aby sa rozlíšila výzva a odpoveď v autentizácii. Istá forma optimalizácie sa dá docieliť tým, že karta bude posielať náhodne vygenerované číslo spolu so svojím číslom karty v prvom kroku [2]. Celý tento proces je znázornený na nasledujúcom obrázku:



Obr. 2.5: Diagram obojsmernej symetrickej autentizácie. Poznámka: *enc* značí šifrovaciu funkciu. Zdroj: [2]

2.6 Voľba čítačky kariet

V bakalárskej práci, na ktorú táto práca naväzuje, bola použitá vývojová doska *Arduino Yún*. Táto doska bola vybratá preto, lebo poskytuje viac možností než iné Arduino dosky a nie je oproti ním až taká drahá. Čítačky kariet sa podľa frekvencie delia na dva druhy: tie, ktoré komunikujú na frekvencii 125 KHz a tie, ktoré komunikujú na 13,56 MHz. Opisy týchto frekvencií sú prevzaté zo stránky ICT [13].

125 KHz

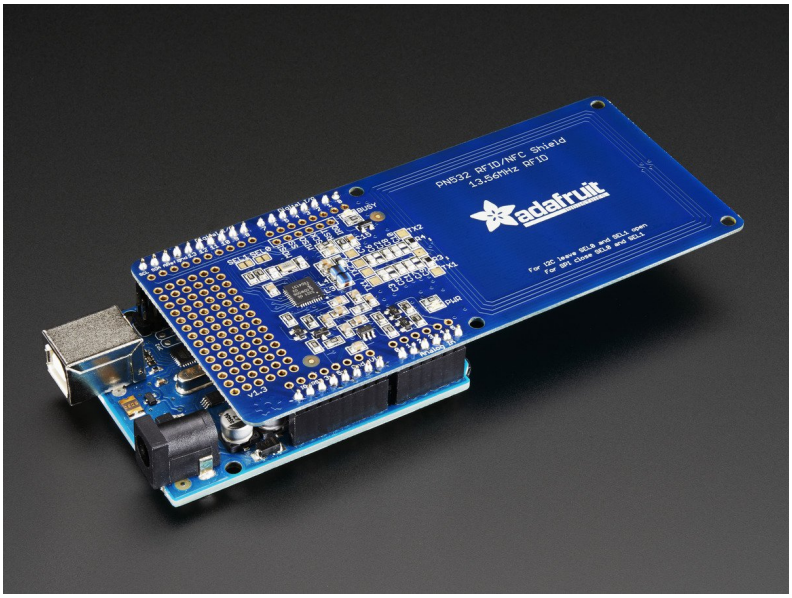
Prenos pomocou 125 KHz je starší štandard. Jeho výhodami sú malá náročnosť na energiu a nízky požadovaný prenos dát. Vďaka tomu dovoľuje prenos na veľkú vzdialenosť a krátky čas požadovaný na čítanie. Nevýhodami tejto frekvencie sú jednosmerná komunikácia a jednoduchá forma šifrovania [13].

13,56 MHz

13,56 MHz je novší štandard, ktorý podporuje obojsmernú komunikáciu, teda možnosť ukladania dát na karte a aj silnejšie šifrovanie. Tento štandard vyžaduje viac energie a viac dát na prenesenie. Maximálna prenosová vzdialenosť je menšia a čas prenosu dát o trochu dlhší. Na tomto štandarde funguje Mifare, napríklad jeho variant Mifare DESFire, a aj Java Card [13].

2.6.1 Adafruit PN532

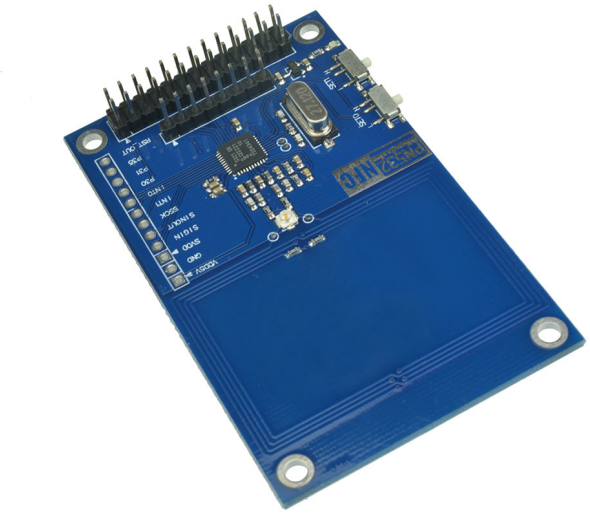
Jedná sa o shield čítačku kariet od známej americkej firmy Adafruit využívajúcu čip PN532. Podporuje NFC a RFID a pracuje na frekvencii 13,56 MHz. Je schopná komunikovať pomocou SPI, I2C aj UART. Môže byť pripojená až na 5,5 V. Jej hlavnou nevýhodou je vyššia cena ($39,95 podľa oficiálnej stránky Adafruit).



Obr. 2.6: Čítačka kariet Adafruit PN532. Zdroj: [14]

2.6.2 PN532

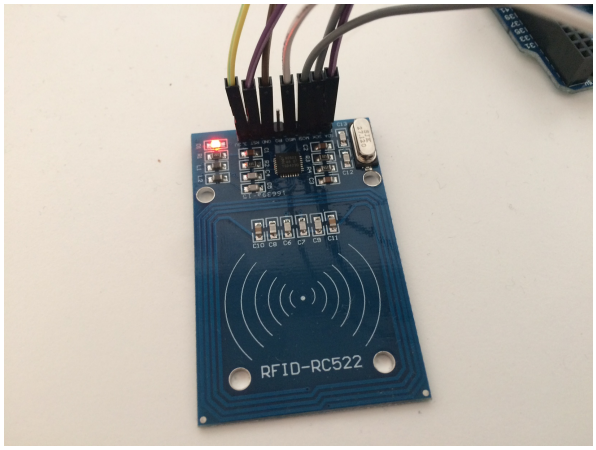
Táto čítačka je lacnejším variantom Adafruit PN532 2.6.1, ktorej slúžila ako predloha. Jej výhodou je, že stojí asi o polovicu menej. Pracuje na frekvencii 13,56 MHz a rovnako podporuje pripojenie cez SPI, I2C a UART. Takisto môže byť pripojená až na 5,5 V.



Obr. 2.7: Čítačka kariet PN532. Zdroj: [15]

2.6.3 MF RC522

Táto čítačka využívajúca čip RC522 pracuje na frekvencii 13,56 MHz. Je veľmi populárna a patrí medzi najlacnejšie čítačky, no podporuje iba RFID. Pripája sa pomocou SPI. Môže byť pripojená maximálne na 3,6 V.



Obr. 2.8: Čítačka kariet RC522. Zdroj: [16]

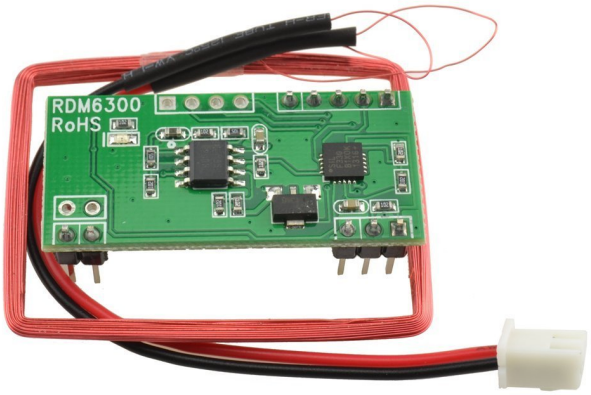
2.6.4 Wiegand RFID čítačka 13,56 MHz

Hlavnou výhodou tejto čítačky je, že je umiestnená v plastovom obale, ktorý je vodeodolný. Poskytuje tak istú ochranu, ak má byť čítačka umiestnená vonku. Takisto čítačka obsahuje indikačnú LED diódu a bzučiak.

Obr. 2.9: Čítačka kariet Wiegand RFID 13,56 MHz. Zdroj: [17]

2.6.5 RDM6300

Táto čítačka pracuje na frekvencii 125 KHz. Jej hlavnou nevýhodou je to, že nie je umiestnená v žiadnom obale a anténa je úplne odkrytá. Pripája sa cez UART a pracuje na napätí 5 V.



Obr. 2.10: Čítačka kariet RDM6300. Zdroj: [18]

2.6.6 Vodeodolná Wiegand RFID čtečka 125 KHz

Táto čítačka je rovnaká ako 2.6.4 s tým rozdielom, že pracuje na frekvencii 125 KHz.

Obr. 2.11: Čítačka kariet Wiegand RFID 125 KHz. Zdroj: [19]

2.6.7 Zvolenie čítačky

Hneď na začiatku sa ako nevhodné javia shield čítačky. Keďže sú v blízkosti Arduino vývojovej dosky, predstavuje to istú bezpečnostnú hrozbu. Čítačka sa musí nachádzať mimo zabezpečený objekt, kde by sa muselo nachádzať aj Arduino. Okrem iného čítačka Adafuit 2.6.1, ktorá je jediná shield čítačka v zozname, stojí oveľa viac ako ostatné spomínané čítačky. Spomedzi možných káblových čítačiek môžeme vylúčiť všetky tie, ktoré pracujú na 125 KHz frekvencii, teda RMD6300 2.6.5 a Wiegand 125 KHz 2.6.6. Tieto čítačky poskytujú starší a menej bezpečný štandard a stoja približne rovnako ako tie, ktoré pracujú na 13,56 MHz. Všetky zvyšné čítačky zo zoznamu sa zdali vhodné a tak boli objednané na testovanie. Výsledky tohoto testovania sa nachádzajú v kapitole Testovanie 4.1. Po zapojení a vyskúšaní čítačiek bolo zistené, že ani jedna nie je vhodná pre túto prácu a nakoniec bola použitá najdrahšia Adafruit PN532 2.6.1. Táto čítačka nebola pripojená ako shield, ale cez káble.

Kapitola 3

Realizácia zabezpečenia vstupu

3.1 Rozšírenie systému IBSG

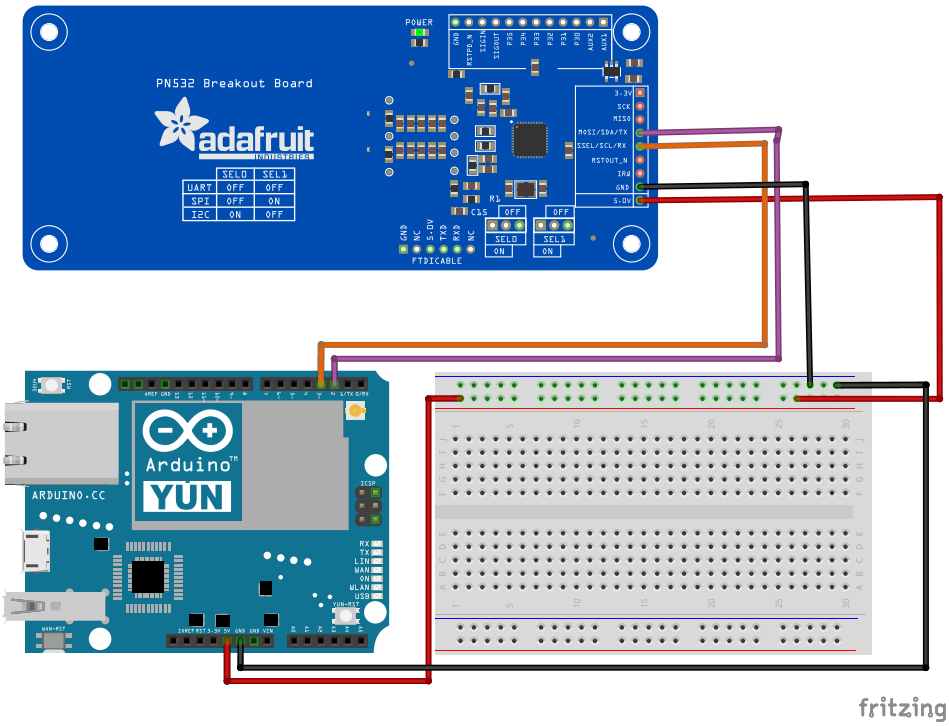
Vývoj pre Ardino bol realizovaný v Arduino IDE vo verzii 1.8.9. Zdrojové kódy od systému IBSG boli rozšírené o tri súbory:

- *cardAuth.ino* obsahujúci funkcie implementujúce logiku autentizácie,
- *cardCommunication.ino* obsahujúci funkcie slúžiace na komunikáciu s kartou,
- *cardStorage.ino* obsahujúci funkcie, pomocou ktorých systém manipuluje s EEPROM pamäťou.

Zoznam dôležitých funkcií v týchto súboroch a ich opis bude v samostatných sekciách. Súbor *Main.ino* bol rozšírený iba o inicializáciu PN532 čítačky. V súbore *REST.ino* boli pridané príkazy súvisiace s pridanou funkcionalitou. Opis REST rozhrania bude v samostatnej podkapitole 3.3. Do súboru *SystemManager.ino*, ktorý obsahuje logiku prepínania stavov, bola funkcia *isButtonPressed()* nahradená za *isCardAuthenticated()*. Takisto tam bolo pridané kontrolovanie maximálnych nesprávnych pokusov a prípadne zablokovanie odomykania. Súbor *Utility.ino* bol rozšírený o pomocné funkcie, ktoré sú používané z pridaného kódu. Nakoniec do súboru *constants.h* boli pridané definície ako napríklad štruktúra posielaných APDU alebo pozície dát v EEPROM.

3.1.1 Schéma zapojenia

Pôvodná schéma zapojenia je rovnaká ako v predošlej práci, ale bola rozšírená o zapojenie čítačky PN532. Na tejto schéme nie je zobrazené, ako zapojiť zvyšok systému. Komunikácia s čítačkou je naprogramovaná cez I2C, preto je nutné na čítačke prehodiť *SEL0* na *on*. Táto schéma bola vyrobená v open-source programe Fritzing v režime montážna doska.



Obr. 3.1: Schéma zapojenia čítačky PN532

3.1.2 Implementácia

Logika systému

Z pôvodného stavového diagramu sú pre rozšírenie dôležité iba stavy Odomknuté a Zamknuté. Po zapnutí systému je východiskový stav Odomknuté. V tomto stave systém čaká na priloženie karty, čím systém prejde do stavu Zamknuté a garáž sa zamkne. Opačne zo stavu Zamknuté sa priložením karty prejde do stavu Odomknuté a garáž bude odomknutá. Ak je systém v stave Zamknuté a bude 3-krát priložená nesprávna karta, systém zakáže odomykanie a zamykanie na minútu. Iba v stave Odomknuté je možné karty inicializovať.

cardAuth.ino

Tento súbor obsahuje implementáciu logiky autentizácie. V tomto súbore je implementovaná funkcia *isCardAuthenticated()*, ktorá bola nahradená namiesto *isButtonPressed()*. Tá zistí, či je v prítomnosti čítačky karta a nájde jej UID v databáze. Ak karta s daným UID ešte nie je inicializovaná, tak sa to zistí vo funkcii *isCardKnown()* a inicializuje sa. Ak všetko prebehne korektne, tak sa zavolá *authenticateCard()*. Jej implementácia je nasledovná:

```
bool isCardAuthenticated()
{
  uint8_t uid[4];

  if(!detectCard(uid))
    return false;

  if(!isCardKnown(uid))
  {
    if(state == LOCKED)
      attemptCounter++;
    return false;
  }

  return authenticateCard(uid);
}
```

Ďalšia dôležitá funkcia implementovaná v tomto súbore je *authenticateCard()*. V nej sa odohráva autentizácia z kapitoly Analýza možností implementácie 2.

```
bool authenticateCard(uint8_t uid[])
{
  uint8_t cardKey[16];
  deriveKey(uid, cardKey)

  if(!sendSelect())
  {
    if(state == LOCKED)
      attemptCounter++;
    return false;
  }

  uint8_t randNumber[16];
  generateRandom(randNumber);

  uint8_t encrypted[16];
  if(!sendEncrypt(randNumber, encrypted))
    return false;

  uint8_t decrypted[16];
  decryptRand(cardKey, encrypted, decrypted);

  if(!compareNumbers(randNumber, decrypted))
  {
    if(state == LOCKED)
      attemptCounter++;
    return false;
  }
  else
    return true;
}
```

cardCommunication.ino

V tomto súbore sú implementované funkcie slúžiace na komunikáciu s čítačkou PN532. Konkrétne sa jedná o funkcie: *detectCard()*, ktorá slúži na detekovanie karty, *sendAPDU()*, ktorá slúži na poslatie APDU karte a funkcie na generovanie APDU podľa jeho typu.

cardStorage.ino

Tento súbor obsahuje implementáciu funkcií slúžiacich na ukladanie a čítanie z EEPROM. Konkrétne sú to funkcie: *isCardKnown()*, ktorá nájde UID v databáze, *printUIDs*, ktorá po zavolaní z REST rozhrania vypíše známe UID z databázy a *addUID*, ktorá po zavolaní z REST rozhrania pridá UID do databázy.

3.1.3 Problémy pri rozširovaní systému

Rozšírenie knižnice pre PN532

Ako bolo spomenuté v 2.6.7, knižnica pre PN532 nepodporuje komunikáciu s Java Card. Pomocou programu *libnfc* skompilovaného so zapnutým prepínačom *–enable-debug* bol získaný výpis príkazov posielaných pri komunikácii s Mifare kartou a Java Card. Z tohto výpisu bolo vidno, že komunikácia prebieha rovnako, akurát pri výzve *InListPassiveTarget* je odpoveď rôzna. Knižnica pre PN532 obsahuje veľa debuggovacích výstupov, ktoré môžu byť zapnuté v *PN532_debug.h* pomocou odkomentovania riadku *#define DEBUG*. So zapnutými debuggovacími výstupmi bolo vidno, že pokus o nadviazanie komunikácie pri priložení karty zlyháva na počítaní checksumu. Kvôli nedostatku času nebolo toto počítanie opravené, iba zakomentované, čo môže predstavovať istú bezpečnostnú hrozbu. Pri reálnom nasadení by bolo nutné toto počítanie implementovať.

Problém s nedostatkom pamäte

Už v minulej práci bolo spomenuté, že pri vývoji nastal problém s nedostatkom pamäte. Na tento problém došlo znova aj pri rozširovaní systému. Arduino Yún má 32 KB flash pamäte, z ktorej ale 4 KB zaberá bootloader. Program nemôže aj tak zaberať 100 % tejto pamäte, lebo bude nestabilný. Prejaví sa to väčšinou tým, že Arduino odmietne komunikovať po sériovej linke a pomôže iba odpojenie od napájania a reštart. Ukázalo sa, že približne 88 % zabratej pamäte je maximum a v súčasnosti toľko program zaberá. Aby sa celý kód do Arduina vošiel, bolo nutné odstrániť istú funkcionalitu z predošlého programu. Jedná sa len o debuggovací výstup, ktorý bežnému používateľovi nič nehovorí a nie je až taký dôležitý. Kód bol iba zakomentovaný a v budúcnosti, ak by systém bežal na zariadení s väčšou pamäťou, tak by mohol byť znova použitý.

Bezpečné uloženie kľúčov

Keďže Arduino Yún neobsahuje žiaden Security Access Module (ďalej už len SAM), neexistuje nijaký bezpečný spôsob, ako ukladať kľúče. Celá bezpečnosť teda stojí na tom, že k Arduinu nebude mať nikto fyzický prístup. V riešení bolo rozhodnuté hlavný kľúč, inicializačný vektor a databázu UID uložiť do EEPROM. Tá nie je nijako chránená, ale aspoň po výpadku elektriny nehrozí strata týchto dát. Štruktúra EEPROM v IBSG je nasledujúca:

Adresa v pamäti	Obsah pamäti	Veľkosť
0 – 31	hlavný kľúč	32 B
32 – 47	inicializačný vektor	16 B
48	počet UID v databáze	1 B
59 – koniec pamäti	záznamy o UID	5 B každý záznam

Tabuľka 3.1: Štruktúra EEPROM v IBSG.

Pozícia týchto dát v pamäti je definovaná v *constans.h* makrami #define. Každý záznam o UID v EEPROM obsahuje 4 B UID a 1 B flagov. V implementácii je použitý iba najmenej významný bit tohto bajtu, ktorý ukazuje, či karta s daným UID má nastavený kľúč a zvyšných 7 bitov je stále voľných.

3.2 Návrh logiky karty

Pri autentizácii na šifrovanie náhodného čísla bola zvolená šifra AES, konkrétne v móde CBC s dĺžkou kľúča 128 b a bez paddingu. Kľúč od karty je uložený v objekte *AESKey* a môže byť inicializovaný iba raz. Pri zmene kľúča je nutné na kartu program nahrať ešte raz, čím sa prepíšu dáta perzistentnej pamäti, a teda sa tento objekt zmaže.

3.2.1 Použité vývojové nástroje

Pre vývoj na karte bol použitý *Java Development Kit* vo verzii 8, *Java Card Kit* vo verzii 2.1.2 a *NetBeans SE* vo verzii 8.2. Pre inštaláciu appletu na kartu a posielanie testovacích APDU bol použitý *GPShell* vo verzii 1.4.4.

3.2.2 Implementácia

Z implementácie je najdôležitejšia funkcia *process()*. Ako už bolo spomenuté v štruktúre appletu 2.4.6, táto funkcia sa zavolá vždy, keď je karta napájaná a obdrží APDU. V tejto funkcii sa skontroluje korektnosť prijatého APDU, teda či je podporovaná inštrukčná trieda a inštrukčný kód. Potom sa podľa inštrukcie vykoná daná operácia. Jej implementácia je nasledovná:

```
public void process(APDU apdu)
{
  if (selectingApplet())
    ISOException.throwIt(ISO7816.SW_NO_ERROR);

  byte [] buffer = apdu.getBuffer();
  if (buffer[ISO7816.OFFSET_CLA] != (byte) 0x80)
    ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);

  short len = apdu.setIncomingAndReceive();
  switch (buffer[ISO7816.OFFSET_INS])
  {
    case (byte) INSTR_ENCRYPT:
      byte encrypted[] = createBuffer();
      encryptData(buffer, ISO7816.OFFSET_CDATA, len, encrypted);
      sendData(apdu, len, encrypted);
      break;

    case (byte) INSTR_SETKEY:
      setKey(buffer, ISO7816.OFFSET_CDATA, len);
      break;

    default:
      ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
  }
}
```

3.2.3 Problémy pri implementácii appletu

Padding

AES je bloková šifra, teda šifruje dáta po tzv. blokoch. Veľkosť tohto bloku závisí od použitého algoritmu. Ak blok takú veľkosť nemá, zvyšok bloku sa vyplní paddingom, aby sa dosiahla daná veľkosť. Java karty podporujú AES iba bez paddingu. Ak teda niekto chce použiť AES na Java Card, musí si sám padding implementovať. V tejto práci nie je v AES padding použitý, pretože to nie je nutné. Pri implementácii bol použitý AES-128, teda dĺžka bloku je 16 B. Karta šifruje iba dáta dĺžky 16 B, a tým je zaručené, že padding nebude nutný. AES knižnica v Arduine padding automaticky dávala, a preto ho bolo nutné v knižnici zakázať.

Identifikovanie kariet

V implementácii sa na identifikovanie kariet a odvodzovanie ich špecifického kľúča používa UID. To je pre Java karty, ktoré sú používané, jedinečné pre každú kartu. Existujú však druhy kariet, ktoré majú UID rovnaké alebo také, ktoré pri každom použití majú UID náhodné. Na internete sa nepodarilo nájsť dokumentáciu ku používaným kartám, a teda nebolo možné zistiť,

akým APDU sa dá získať ich sériové číslo, ktoré je jedinečné pre každú kartu. Do budúcna by bolo nutné toto APDU zistiť, prípadne implementovať vlastné APDU a generovať vlastné čísla pre karty.

3.3 Komunikačné protokoly

3.3.1 Rozhranie k IBSG

Rozhranie REST slúžiace na komunikáciu s Arduinom bolo už implementované v práci, na ktorú táto naväzuje. Z REST rozhrania kvôli nedostatku pamäte boli odstránené nasledujúce príkazy: *values*, *debug*, *time*, *valuesHelp*, *commands*. Jediný príkaz, ktorý ostal je *state*, no aj ten bol rozšírený. Do rozhrania boli pridané príkazy zobrazené v nasledujúcej tabuľke a potom bližšie popísané:

Príkaz	Stručný popis
state	vypíše aktuálny stav systému
softreset	vykoná soft reset nad systémom
printuids	vypíše zoznam UID, ktoré IBSG pozná
adduid	slúži na pridanie UID do IBSG

Tabuľka 3.2: Zoznam príkazov v REST rozhraní.

state

Tento príkaz bol implementovaný už v minulej práci a slúžil na zobrazenie stavu garáže. Po implementácii bol rozšírený, aby ukazoval aj počet nesprávnych pokusov o autentizáciu. Po tretej nesprávnej autentizácii a zablokovaní odomykania systému zobrazí aj čas do povolenia odomykania.

softreset

Soft reset znamená: nastavenie stavu na UNLOCKED, resetovanie počtu pokusov o nesprávnu autentizáciu, resetovanie času do povolenia odomykania pri zablokovaní systému a vypnutie alarmu. Systém IBSG dovoľoval soft reset iba cez tlačidlo. To ale bolo zmenené a teraz sa soft reset vykonáva cez REST rozhranie.

printuids

Tento príkaz slúži na vypísanie UID známych pre IBSG. Každý záznam je vypísaný v tomto poradí: poradové číslo podľa pridania do systému, UID v hexadecimálnej sústave a flagy.

adduid

Na pridanie UID do systému slúži tento príkaz. UID musí byť zadané v hexadecimálnej sústave. Samotné UID v príkaze sa zadáva za „/“ po adduid, teda napríklad *adduid/BEEFCAFE*.

3.3.2 Rozhranie ku karte

Na komunikáciu s kartou boli definované APDU zobrazené v nasledujúcej tabuľke a potom bližšie popísané:

Názov APDU	Stručný popis
select	zvolí applet na karte
encryptData	zašifruje blok dlhý 16 B
setKey	slúži na inicializáciu kľúča na karte

Tabuľka 3.3: Zoznam APDU.

select

Toto APDU je definované štandardom a slúži na zvolenie appletu na karte podľa jeho AID. AID pre implementovaný applet je *0x0102030405060809*. Formát select APDU pre zvolenie appletu teda vyzerá nasledovne:

CLA	INS	P1	P2	Lc	Data	Le
0x00	0xA4	0x04	0x00	0x08	0x0102030405060809	0x00

Tabuľka 3.4: Formát select APDU.

encryptData

Toto APDU slúži na poslanie dát karte, ktoré zašifruje a vráti zašifrované v odpovedi. Vstup musí byť vždy 16 B a očakávaná odpoveď tiež 16 B. Formát encryptData APDU je nasledovný:

CLA	INS	P1	P2	Lc	Data	Le
0x80	0x01	0x00	0x00	0x10	16B dát na zašifrovanie	0x10

Tabuľka 3.5: Formát encryptData APDU.

setKey

Týmto APDU vie IBSG inicializovať kľúč karte, ktorý musí byť dĺžky 16 B. Formát setKey APDU je nasledovný:

CLA	INS	P1	P2	Lc	Data	Le
0x80	0x02	0x00	0x00	0x10	kľúč dĺžky 16B	0x00

Tabuľka 3.6: Formát setKey APDU.

Definované status words

Pri korektnej odpovedi karta podľa štandardu vráti status word 0x9000. Ak sú dáta posielané v zlom formáte alebo pri operácii nastala chyba, karta pošle buď status word definovaný štandardom alebo jeden z nasledujúcich status words:

Status Word	Popis	Môže nastať pri APDU
0x9101	kľúč je už nastavený	setKey
0x9102	zlá dĺžka kľúča	setKey
0x9103	kľúč nie je nastavený	encryptData
0x9104	zlá dĺžka šifrovaných dát	encryptData
0x9105	nepodporovaná verzia protokolu	encryptData
0x9106	zlá očakávaná dĺžka odpovede	encryptData

Tabuľka 3.7: Zoznam definovaných status words.

Kapitola 4

Testovanie

4.1 Testovanie kúpených čítačiek

4.1.1 Čítačka Wiegand

Ako prvá bola vyskúšaná čítačka Wiegand 2.6.4. Prvá jej nevýhoda spočíva v tom, že musí byť napájaná na 8 až 15 V. To je napätie, ktoré Arduino Yún nezvládne. Preto bolo nutné kúpiť napájací zdroj. Takisto jej inštalácia bola mierne zložitá, keďže čítačka nemala žiadne konektory, len vyvedené káble a všetko bolo nutné pájkovať. Po zapojení do Arduina a testovaní sa ukázalo, že priložená knižnica nevie posielať dáta smerom ku karte, iba čítať z karty a ostatné knižnice tretích strán to tiež nepodporovali. Táto čítačka nie je veľmi rozšírená, a preto na internete nie je až toľko návodov ako k ostatným čítačkám. Táto čítačka je teda pre túto prácu nevhodná.

4.1.2 Čítačka MF RC522

Ako ďalšia čítačka bola testovaná MF RC522 2.6.3. Tej stačilo pripájkovať pribalený konektor a z toho bolo možné pripojiť káble rovno do Arduina.

Prvý problém nastal pri snahe detekovať priloženým programom rôzne karty. Dve karty (Mifare Classic) pribalené k čítačke bola čítačka schopná detekovať. Java karty, ktoré boli na programovanie zapožičané, už detekovať nedokázala. Je to pravdepodobne preto, že čítačka je napájaná 3,3 V a to možno nestačí pre všetky karty. Čítačka podľa dokumentácie môže byť napájaná maximálne 3,6 V, no taký výstup z Arduina nie je dostupný. Arduino ma prítomný ešte 5 V výstup a po zapojení čítačky do 5 V bol program nakoniec schopný Java karty detekovať.

Tu ale nastal druhý problém. Použitá knižnica nemá implementovanú komunikáciu s Java Card. Na internete nebola nájdená knižnica, ktorá by komunikáciu podporovala. Nakoniec bolo rozhodnuté túto čítačku nepoužiť vôbec, keďže neexistuje knižnica podporujúca tento prenos a čítačka by musela byť napájaná na väčšom napätí než podporuje.

4.1.3 Čítačka PN532

Ako tretia bola vyskúšaná čítačka PN532 2.6.2. Tá vie pracovať pri napätí 5 V, no rovnako nebola schopná detekovať kartu ani po pripojení čítačky k počítaču a použití programu *libnfc*.

4.1.4 Adafruit PN532

Ako posledná bola objednaná Adafruit PN532 2.6.1. Tá po pripojení do počítača a cez *libnfc* kartu našla, no priložené knižnice stále odmietali komunikovať. Po malej zmene (spomínanej v 3.1.3) knižničnej funkcie na komunikáciu sa podarilo komunikáciu cez túto čítačku spojazdniť.

4.2 Testovanie implementovaného systému

4.2.1 Prípady využitia

UC1: odomknutie/zamknutie garáže

Majiteľ garáže alebo iné osoby, ktoré vlastnia kartu od garáže, budú schopné priložením karty garáž odomknúť alebo zamknúť (v závislosti na aktuálnom stave). To, či je garáž zamknutá alebo odomknutá, bude daná osoba vedieť pomocou LED diódy. To, že garáž zamkla alebo odomkla, zistí dvojitým zapípaním bzučiaka.

UC2: pridanie UID do systému

Majiteľ garáže bude môcť pridať nové UID kariet do systému pomocou REST rozhrania. Na pridanie mu postačí obyčajný webový prehliadač.

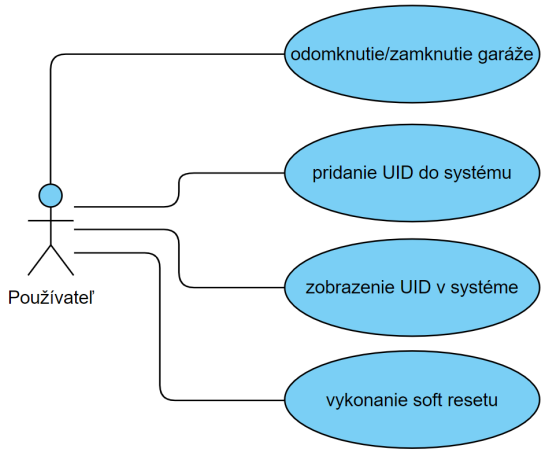
UC3: zobrazenie UID v systéme

Majiteľ garáže si bude môcť zobraziť UID kariet v systéme. Na zobrazenie UID v systéme mu postačí obyčajný webový prehliadač. Po vykonaní príkazu sa na obrazovke zobrazí tabuľka UID kariet v systéme a ich flagy.

UC4: vykonanie soft resetu

Ak sa odomykanie a zamykanie garáže zablokuje na minútu po treťom priložení zlej karty alebo sa spustí poplach, používateľ bude môcť nad systémom vykonať soft reset. Tým sa timeout na odomykanie a poplach zruší a garáž sa dostane do stavu Odomknuté.

4.2.2 Diagram prípadov využitia



Obr. 4.1: Diagram prípadov využitia. Obrázok vygenerovaný pomocou stránky VisualParadigm Online.

4.2.3 Test priloženia karty

Tento test bol vykonaný na finálnej verzii systému. Čítačka kariet bola vyvedená vonku a prilepená lepiacou páskou na okraj krabice. Test bol vykonaný 10-krát priložením inicializovanej karty so správnym kľúčom k systému. Tým sa otestovala funkčnosť prípadu využitia 1. Výsledky sú v nasledujúcej tabuľke. OK značí, že karta bola detekovaná a garáž sa odomkla alebo zamkla.

Meranie č.	1	2	3	4	5	6	7	8	9	10
Výsledok	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK

Tabuľka 4.1: Test priloženia karty.

Všetky testovacie merania boli v poriadku.

4.2.4 Test pridania karty do systému

V tomto teste sa začalo s prázdnou databázou UID. Následne sa pridalo UID karty do databázy, vypísal sa obsah databázy, aby sa skontrolovalo, či sa dané UID pridalo a garáž sa skúsila odomknúť a zamknúť. Tento test bol vykonaný iba 3-krát, pretože na testovanie boli zapožičané iba tri rôzne karty. Tým sa otestovala funkčnosť prípadu vyžitia 1, 2 a 3. Výsledky sú v nasledujúcej tabuľke:

Meranie č.	1	2	3
Výsledok	OK	OK	OK

Tabuľka 4.2: Test pridania karty do systému.

4.2.5 Test magnetických spínačov

Tento test bol vykonaný tak, že v každom meraní sa garáž kartou zamkla. Potom sa otvorilo veko krabice, čím sa simulovalo narušenie objektu. Následne sa garáž pomocou *softreset* REST príkazu resetovala. Tým sa otestovala funkčnosť prípadu využitia 1 a 4. Test bol vykonaný 10-krát. Výsledky sú v nasledujúcej tabuľke:

Meranie č.	1	2	3	4	5	6	7	8	9	10
Výsledok	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK

Tabuľka 4.3: Test magnetických spínačov.

Všetky testovacie merania boli v poriadku.

4.2.6 Meranie rýchlosti funkcie isCardAuthenticated()

Tento test bol vykonaný tak, že sa uložil čas pomocou funkcie *millis()* na začiatku a konci funkcie *isCardAuthenticated()*. Rozdielom týchto hodnôt je dĺžka trvania autentizácie. Test bol vykonaný 10-krát a v každom meraní bola priložená inicializovaná karta so správnym kľúčom. Čísla v riadku Dĺžka trvania sú uvedené v milisekundách.

Meranie č.	1	2	3	4	5	6	7	8	9	10
Dĺžka trvania	232	232	217	216	214	268	220	227	245	217

Tabuľka 4.4: Dĺžka trvania funkcie isCardAuthenticated().

Priemerná hodnota dĺžky trvania funkcie *isCardAuthenticated()* je 228,8.

Kapitola 5

Profil útočníka

Záver práce tvorí klasifikácia druhov útokov na čipové karty a popis najčastejšie využívaných. V závere tejto kapitoly je navrhnutý systém otestovaný voči týmto útokom. Hlavným zdrojom pre túto kapitolu je prednáška BI-HWB[2].

5.1 Druhy útokov

Čipová karta je len počítač, na ktorom beží softvér, teda ako všetky ostatné počítače je tiež náchylná na rôzne útoky. Tieto útoky sa kategorizujú podľa rôznych delení. Opisy delenia a definície typov útokov sú prevzaté z prednášky[2]. Prvé delenie je podľa fyzického zásahu do karty. Tie sa delia na:

- invazivné, pri ktorých dochádza ku rozobraniu karty, odpuzdreniu čipu alebo pripojeniu vlastného obvodu. Často dochádza k zničeniu čipu, a teda aj karty,
- semiinvazivné, pri ktorých dochádza takisto k odpuzdreniu čipu, no nenarušuje sa pasívna vrstva a maximálne sa opticky zavádzajú poruchy,
- neinvazivné, pri ktorých sa karta nerozoberá a možný vplyv je len zvonku.

Ďalším kritériom delenia je delenie podľa druhu zbierania dát. Tie delíme na:

- aktívne, pri ktorých sa karte posiela vstup a na základe toho sa kontroluje výstup,
- pasívne, kde sa karta len odpočúva.

[2]Zdroj: Ing. Jiří Buček Ph. D., Prednáška 5 BI-HWB, 2019, [cit. 2019-03-09], dostupné z Moodle FIT ČVUT, po prihlásení

5.2 Známe útoky

5.2.1 Invazivné útoky

Medzi najčastejšie invazivné útoky patrí istá forma reverse engineeringu čipu na karte. Ten sa po odobratí z karty preskúma silným mikroskopom. Následne je možné zistiť kľúč z pamäte. Ďalšou metódou je manipulácia s elektrickými vodičmi na karte bez toho, aby boli poškodené, čím sa uhádne kľúč. Tieto metódy sú veľmi obtiažne, finančne náročné a vedia ich vykonať len špeciálne laboratóriá.

5.2.2 Replay útok

Útok typu replay je neinvazívny a pasívny. Pri tomto útoku útočník by odpočúval komunikáciu a uložil by si správu použitú na autentizáciu, kľudne aj zašifrovanú. Tú by potom znova poslal strane, ktorá už túto správu prijala predtým a tým pádom by nevedela rozlíšiť, že vlastne prijala správu od niekoho iného než pôvodne.

5.2.3 Preplay útok

Útok typu pre-play je neinvazívny a aktívny. Pri tomto útoky si útočník vybuduje databázu dotazov a ich odpovedí tým, že bude obom stranám posielať dotazy a ukladať si odpovede. Čím väčšiu databázu si útočník vybuduje, tým má väčšiu šancu, že na náhodný dotaz bude vedieť správne odpovedať.

5.2.4 Relay útok

Útok typu relay je neinvazívny a pasívny. Jedná sa typ útoku *man in the middle*. Zmysel tohto útoku spočíva v tom, že sa vytvorí komunikačný tunel medzi dvoma zariadeniami a tie spolu začnú komunikovať bez vedomia ich majiteľov.

5.2.5 Útoky postranným kanálom

Medzi dosť časté neinvazivné a pasívne útoky patrí útok postranným kanálom. Postranný kanál je definovaný ako nevyžiadaný únik informácii medzi modulom a jeho okolím. Môže sa jednať napríklad o spotrebu modulu, čas trvania šifrovania alebo vyžarovanie elektromagnetických vĺn. Touto metódou je možné uhádnuť kľúč samotný, no ale aj iné informácie, napríklad typ šifrovacieho algoritmu[2]. Tieto postranné kanály sa delia podľa typu sledovaných dát na:

[2]Zdroj: Ing. Jiří Buček Ph. D., Prednáška 5 BI-HWB, 2019, [cit. 2019-03-09], dostupné z Moodle FIT ČVUT, po prihlásení

- časový postranný kanál, kde sa sleduje dĺžka trvania operácií, napríklad ako dĺžka operácií závisí od kľúča,
- odberový postranný kanál, kde sa sleduje spotreba pri rôznych šifrovacích operáciách,
- chybový postranný kanál, kde sa v module naschvál vyvolávajú chyby a sleduje sa reakcia modulu,
- elektromagnetický postranný kanál, kde sa meria elektromagnetické vyžarovanie modulu pri šifrovacích operáciách.

5.2.6 Útok postranným kanálom spotreby

Pri útoku postranným kanálom spotreby existujú dve metódy: simple power analysis (ďalej už len SPA) a differential power analysis (ďalej už len DPA). Pri SPA sa meria spotreba iba raz. Výsledkom tohto merania je graf, kde spotreba závisí na čase. Tým sa dá napríklad určiť, ktorá časť šifrovacieho algoritmu sa vykonávala v akom čase, pretože v tej dobe bude modul spotrebovávať viac energie. Zložitejšia, no prínosnejšia je metóda DPA. Pri nej sa meranie musí vykonať viackrát a následne analyzovať stav spotreby v rovnakom čase počas všetkých meraní[2].

5.3 Test navrhnutého systému

5.3.1 Bezpečnosť Arduina

Ako už bolo spomenuté v kapitole Realizácia zabezpečenia vstupu 3.1.3, najväčšiu bezpečnostnú hrozbu predstavuje samotné Arduino. To neumožňuje nijako bezpečne ukladať kľúče. V reálnych systémoch je toto riešené inštaláciou SAM a ukladaním kľúčov tam. V implementácii je kľúč uložený v EEPROM, z ktorej sa dá čítať. Arduino takisto nepodporuje žiadnu ochranu operačnej pamäte. Takže aj jednoduchým útokom, napríklad buffer overflow, by bolo možné pamäť prečítať. Ak by aj tak mal byť tento systém nasadený, musí sa zaručiť to, že bude fyzický prístup úplne obmedzený. Ďalej pre zaručenie bezpečnosti by malo byť určite nastavené heslo na Wi-Fi, prípadne ho vôbec nepoužívať a využiť pripojenie cez ethernet.

5.3.2 Odolnosť voči replay útoku

Systém je sčasti odolný proti útokom replay. Príkladom pre tento útok v IBSG by bolo to, ak by sa nejaká karta autentizovala stále rovnakou správou

[2]Zdroj: Ing. Jiří Buček Ph. D., Prednáška 5 BI-HWB, 2019, [cit. 2019-03-09], dostupné z Moodle FIT ČVUT, po prihlásení

(náhodným číslom). Túto správu by útočník zachytil. Ak by prišiel k terminálu a na výzvu by odpovedal touto správou, tak by to terminál vyhodnotil, že je to správna karta. To je ale ošetrené tým, že pri každej autentizácii sa používa náhodné číslo, a teda aj keby útočník túto správu zachytil pri ďalšej výzve, odpoveď by bola iná. V IBSG sa používa ako náhodné číslo pole pätnástich pseudonáhodných bajtov (čísla od 0 do 255). Každý náhodný bajt je generovaný funkciou *random(255)*, ktorá generuje len pseudonáhodné čísla a seed pre túto funkciu je analógový šum na pine *A5*. Ak by útočník dosiahol to, že na danom pine by bol dvakrát po sebe rovnaký šum, dostal by dvakrát rovnakú postupnosť generovaných čisel. To je nebezpečné a mohol by to zneužiť. Bez skutočne náhodného generátora čísel sa tomuto útoku úplne vyhnúť nedá.

5.3.3 Odolnosť voči preplay útoku

Systém je sčasti odolný proti preplay útokom. Problém je úplne rovnaký ako pri útoku typu replay. Ak by útočník bol schopný ovplyvniť generovanie náhodných čísel, vedel by útok vykonať. Odolnosť proti tomuto útoku sa dá zaručiť len vysokou entropiou náhodného čísla, ktoré sa šifruje. Použitých 15 B je celkom dostatočná entropia a predpočítať všetky kombinácie by chvíľu trvalo.

5.3.4 Odolnosť voči relay útoku

Systém je sčasti odolný voči relay útokom. Príkladom pre tento útok v IBSG je priloženie vysielača k terminálu. Ten by vedel všetko preposlať nejakému prijímaču a odosielaču týchto dát, ktorý by bol v blízkosti skutočnej karty. Tento útok by mohol byť vykonaný v miestach, kde je možné priblížiť sa k obeti na takú vzdialenosť, ktorá by umožnila komunikáciu s kartou obete.

Timeout na komunikáciu s kartou je prednastavený na jednu sekundu. Ak by teda celá komunikácia a preposielanie trvalo viac ako sekundu, spojenie sa preruší. Ak by ale celá komunikácia trvala menej než jednu sekundu, tento útok by bolo možné vykonať. V reálnych systémoch je toto riešené tak, že sa meraním zistí priemerná dĺžka komunikácie a tá sa nastaví ako maximálna. Tým pádom pridanie hocijakého preposielania predĺži komunikáciu a spojenie sa preruší.

5.3.5 Odolnosť voči útoku postranným kanálom

Vykonať útok postranným kanálom je relatívne zložité a jeho vykonanie na poskytnutých kartách je mimo rozsah tejto práce. Výrobcovia kariet s týmito útokmi počítajú a väčšinou sú ošetrené aspoň pre základné útoky postranným kanálom.

Záver

Táto práca nadväzovala na obhájenú bakalársku prácu *Inteligentní bezpečnostní systém garáže*, v ktorej bol navrhnutý a implementovaný jednoduchý bezpečnostný systém na platforme Arduino. V tejto práci nebolo nijako riešené zabezpečenie vstupu, keďže to nebolo v rozsahu zadania a odomykanie/zamykanie bolo riešené iba pomocou stlačenia tlačidla. Hlavným zadaním bolo vymyslieť jednoduchý systém autentizácie využívajúci bezkontaktné čipové karty s využitím dnešných technológií.

Práca začína opisom IBSG a vymedzením, čo sa bude implementovať. Nasleduje analýza možností implementácie, kde boli popísané neproprietárne platformy pre čipové karty. Ako najvhodnejšia sa javila platforma Java Card, ktorá bola podrobne opísaná. Ďalej bola popísaná metóda využitej symetrickej autentizácie a spôsob použitého odvodzovania špecifických kľúčov pre karty. Ako šifrovací algoritmus bol zvolený AES v operačnom móde CBC s dĺžkou kľúča 128 bajtov. Na odvodzovanie kľúča bol zvolený HMAC spolu s SHA-256. V poslednej časti tejto kapitoly je opis dostupných RFID čítačiek pre Arduino. Po zakúpení troch čítačiek, ktoré sa javili ako vhodné pre túto prácu, sa testovaním zistilo, že každá je niečim nevhodná pre tento projekt. Nakoniec bola kúpená najdrahšia a najkvalitnejšia Adafruit PN532.

Kapitola Realizácia zabezpečenia vstupu začína popisom implementácie rozšírenia systému IBSG. Obsahuje schému zapojenia a rozbor pridaných súborov a funkcií, ktoré súbory obsahujú. Záver tejto časti tvorí zoznam problémov, na ktoré bolo narazené počas rozširovania a spôsob ich riešenia. Prvý problém bol v nefunkčnosti knižnice k čítačke *PN532* pre Java Card. Knižnica opravená nebola, keďže je to mimo rozsah tejto práce. Kód, ktorý problém spôsoboval, bol zakomentovaný. Do budúcna by bolo určite vhodné toto opraviť, pretože to môže predstavovať bezpečnostnú zraniteľnosť. Ďalší problém bol v nedostatku pamäte a bolo nutné istú funkcionalitu predošlého programu odstrániť. Nejednalo sa o nič dôležité, len o debuggovacie výstupy. Pri ďalších prípadných rozšíreniach bude nutné zvoliť Arduino s väčšou flash pamäťou alebo inú platformu. Tretí problém spočíval v neprítomnosti SAM

na Arduine, teda nie je možné nijako bezpečne uložiť kľúče. Do budúcna by bolo vhodné k systému SAM pripojiť alebo sa presunúť na platformu, ktorá SAM obsahuje. Ďalšia časť tejto kapitoly sa venuje opisu implementácie appletu pre Java Card. V závere tejto kapitoly je popísaný komunikačný protokol, teda opis pridaných príkazov do REST rozhrania a definovaná APDU a status wordy.

Začiatok kapitoly Testovanie sa venuje testovaniu zakúpených čítačiek. V tejto kapitole bol systém otestovaný na funkčnosť a rýchlosť. Na základe vymedzených prípadov užitia boli navrhnuté testy, čím je zaručené, že systém je stabilný.

Posledná kapitola obsahuje kategorizáciu útokov na karty a príklady najčastejších útokov, ktoré sú vysvetlené. V závere je opísané, do akej miery je systém voči týmto útokom odolný. Samotné Arduino veľmi bezpečnou platformou nie je a jej zabezpečenie spočíva len v zaručení obmedzenia fyzického prístupu.

Systém je sčasti odolný voči útoku typu replay, pretože pri každej autentizácii sa používa náhodné číslo. Toto číslo je v skutočnosti pseudonáhodné. Ak by útočník dokázal zreprodukovať seed funkcie generujúcej náhodné čísla, dostal by rovnakú postupnosť, a tým by bol systém ohrozený. Systém je sčasti odolný voči preplay útokom, pretože sa šifruje náhodné číslo zložené z pätnástich bajtov pseudonáhodných čísel, čo poskytuje vysokú entropiu. Do budúcna by bolo vhodné rozšíriť systém o generátor skutočných náhodných čísel alebo prejsť na platformu, ktorá taký generátor obsahuje. Systém je sčasti odolný voči relay útokom. Timeout pre komunikáciu s kartou je nastavený na jednu sekundu. Ak by teda presmerovanie a samotná komunikácia trvala kratšie než jednu sekundu, útok by bolo možné zrealizovať. Ak by sme sa chceli tomuto útoku vyhnúť, bolo by nutné vykonať merania, z ktorých by sa zistila priemerná dĺžka komunikácie a tú nastaviť ako timeout. Test na útok postranným kanálom nebol vykonaný, pretože je zložitý a mimo rozsah práce.

Garáž je po implementácii rozhodne bezpečnejšia už len preto, že predchádzajúca implementácia mala odomykanie riešené bez autentizácie. Všetky spomínané útoky by sa síce za istých podmienok dali vykonať, no to by znamenalo, že útočník má fyzický prístup k Arduinu, a teda je v garáži. To bude vyhodnotené ako poplach, a teda systém svoj účel splní.

Pridávanie UID kariet a zobrazenie UID kariet v databáze nie je veľmi užívateľsky prívetivé. Do budúcna by bolo vhodné implementovať jednoduchú webovú stránku na tieto prípady využitia. Na tejto stránke by bolo možné vidieť aj stav garáže.

Literatúra

[1] Váňa, M.: *Inteligentní bezpečnostní systém garáže*. České vysoké učení technické v Praze, Fakulta informačních technologií, Katedra počítačových systémů, 2018, bakalárska práca, Vedúci práce: Martin Daňhel.

[2] Wolfgang Rankl, W. E.: *Smart Card Handbook*. Wiley, John & Sons, Incorporated, tretie vydanie, 2004, ISBN 0-470-85668-8, preklad: autor bakalárskej práce.

[3] MULTOS Technology [online]. 2019, [cit. 2019-03-13]. Dostupné z: `https://www.multos.com/technology`

[4] ORACLE JavaCard [online]. 2019, [cit. 2019-03-04]. Dostupné z: `https://www.oracle.com/java/java-card.html`

[5] MULTOS FAQ [online]. 2019, [cit. 2019-03-13]. Dostupné z: `https://www.multos.com/faq`

[6] JavaCard Technology Overview [online]. 2019, [cit. 2019-03-09]. Dostupné z: `https://www.oracle.com/technetwork/java/embedded/javacard/overview/overview-jsp-135353.html`

[7] ISO 7816 part 4, section 5 [online]. 2019, [cit. 2019-03-22]. Dostupné z: `https://cardwerk.com/smart-card-standard-iso7816-4-section-5-basic-organizations/`

[8] java.lang package summary [online]. 2019, [cit. 2019-03-16]. Dostupné z: `https://docs.oracle.com/javacard/3.0.5/api/java/lang/package-summary.html`

[9] javacard.framework summary [online]. 2019, [cit. 2019-03-16]. Dostupné z: `https://docs.oracle.com/javacard/3.0.5/api/javacard/framework/package-summary.html`

[10] javacard.security summary [online]. 2019, [cit. 2019-03-16]. Dostupné z: https://docs.oracle.com/javacard/3.0.5/api/javacard/security/package-summary.html

[11] javacardx.crypto summary [online]. 2019, [cit. 2019-03-16]. Dostupné z: https://docs.oracle.com/javacard/3.0.5/api/javacardx/crypto/package-summary.html

[12] RFC 2104, HMAC: Keyed-Hashing for Message Authentication [online]. 2019, [cit. 2019-03-23]. Dostupné z: https://www.ietf.org/rfc/rfc2104.txt

[13] ICT, Choosing Card Technology [online]. 2019, [cit. 2019-03-17]. Dostupné z: https://www.ict.co/Choosing-Card-Technology

[14] Adafruit PN532 NFC/RFID Controller Shield for Arduino [online]. 2019, [cit. 2019-05-05]. Dostupné z: https://www.adafruit.com/product/789

[15] RFID IC čtečka karet 13.56MHz [online]. 2019, [cit. 2019-05-05]. Dostupné z: https://arduino-shop.cz/arduino/2005-rfid-ic-ctecka-karet-13.56mhz-modul-pro-arduino-pn532-nfc.html

[16] RFID / ARDUINO : COPY A CARD WITH KNOWN KEYS [online]. 2019, [cit. 2019-05-05]. Dostupné z: https://ebc81.wordpress.com/2014/11/16/rfid-arduino-copy-a-card-with-known-keys/

[17] RFID čtečka 13,56 MHz [online]. 2019, [cit. 2019-05-05]. Dostupné z: https://arduino-shop.cz/arduino/5423-rfid-ctecka-13-56-mhz.html

[18] HiLetgo 125 KHZ EM4100 RFID Card Read Module RDM6300 RF Module [online]. 2019, [cit. 2019-05-05]. Dostupné z: https://www.amazon.com/HiLetgo-EM4100-Module-Compatible-Arduino/dp/B00HGB9TOE

[19] Voděodolná RFID čtečka 125 KHz [online]. 2019, [cit. 2019-05-05]. Dostupné z: https://arduino-shop.cz/arduino/5424-rfid-ctecka-125-khz.html

Dodatok **A**

Inštalačná príručka

V tejto inštalačnej príručke sa počíta s tým, že používateľ už má systém IBSG nainštalovaný, funkčný a chce ho len rozšíriť o zabezpečenie vstupu. Obsahom tejto príručky je: rozšírenie systému IBSG, inštalácia kariet a pridávanie kariet do systému.

A.1 Rozšírenie systému IBSG

- Vypojte systém z napájania.
- Pripojte čítačku podľa schémy z časti 3.1.1.
- Nainštalujte vývojové prostredie Arduino IDE z oficiálnych stránok `https://www.arduino.cc/en/Guide/HomePage`.
- Pripojte systém IBSG k počítaču.
- Otvorte zdrojový kód *media/impl/arduino/init_eeprom/init_eeprom.ino* v Arduino IDE a vyplňte pole *masterKey* s kľúčom pre šifrovanie a pole *iv* s inicializačným vektorom.
- Nahrajte program na Arduino, tým sa uloží kľúč a inicializačný vektor do pamäte.
- Otvorte zdrojový kód *media/impl/arduino/Main/Main.ino* v Arduino IDE.
- Pridajte všetky knižnice z priečinka *media/impl/arduino/libraries* pomocou Arduino IDE.
- Nahrajte program na Arduino.
- Systém je rozšírený o zabezpečenie vstupu a je pripravený k použitiu.

A.2 Inštalácia kariet

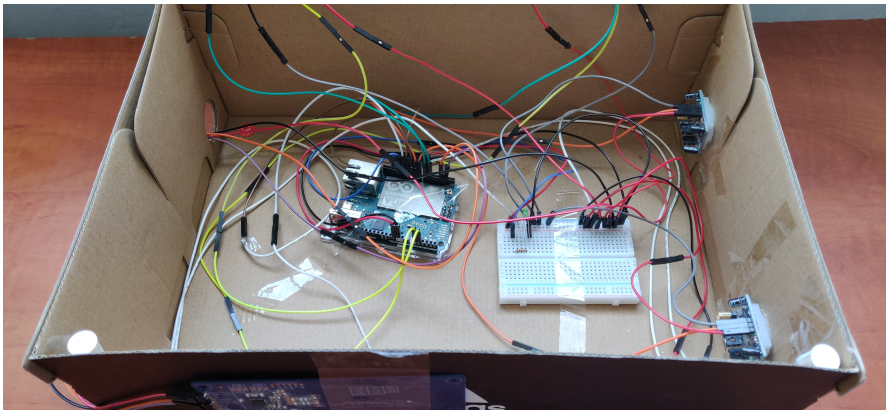
- V súbore *media/impl/javacard/src/IBSG_auth/IBSG_auth.java* upravte pole *iv* s príslušným inicializačným vektorom.
- Nainštalujte *Java Development Kit 8*, inštalácia bola testovaná vo verzii 8u202.
- Nainštalujte *Java Card Kit 2.1.2*.
- Nainštalujte *GPShell 1.4.4*.
- V súbore *media/impl/javacard/scripts/e221.cmd* nastavte správne cesty ku nainštalovaným *Java Development Kit 8, Java Card Kit 2.1.2 GPShell 1.4.4*, ak ste ich inštalovali inde ako na prednastavené miesto.
- V príkazovom riadku spustite *media/impl/javacard/scripts/e221.cmd* a *media/impl/javacard/scripts/c221.cmd*.
- Pripojte k počítaču čítačku kariet a vložte kartu, na ktorú chcete program nainštalovať.
- Následne spustite príkaz *gpshell install.txt*
- Karta je pripravená k inicializácii.
- Pred každou inicializáciou karty v IBSG musí byť jej UID pridané cez REST rozhranie a systém musí byť v stave *Odomknutné*.

A.3 Pridávanie kariet do systému

- Pripojte sa na Wi-Fi, ktoré vytvorí Arduino alebo sa pripojte pomocou ethernetu k Arduinu.
- Zadajte adresu *arduino.local/arduino/adduid/UID* vo webovom prehliadači, kde namiesto *UID* zadajte UID karty v hexadecimálnej sústave. Na zistenie UID karty môžete použiť priložený Arduino program umiestnený v *media/impl/arduino/get_uid/get_uid.ino*.
- Vo webovom prehliadači zadajte adresu *arduino.local/arduino/printuids* a skontrolujte, či pridané UID sa zobrazilo v tabuľke.
- Odomknite garáž už funkčnou kartou alebo vykonajte soft reset zadaním adresy *arduino.local/arduino/softreset*.
- Priložte kartu k čítačke.
- Karta sa inicializuje a rovno zamkne garáž. Ak systém 2-krát zapípa, karta je inicializovaná.

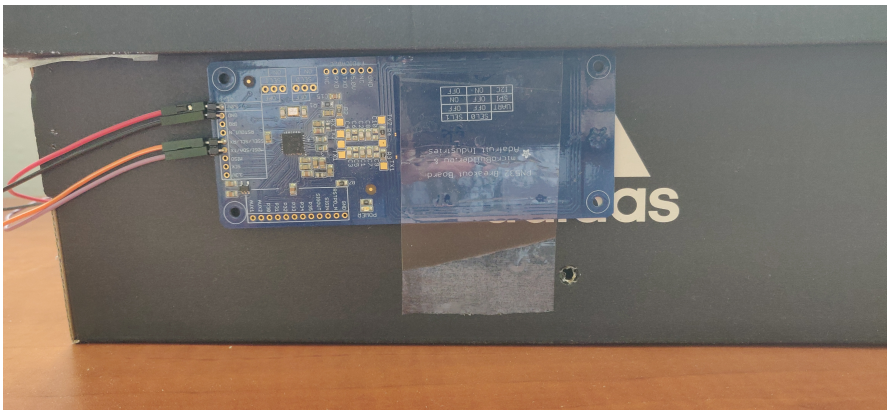
Dodatok B

Fotografie systému



Obr. B.1: Fotografia zapojenia systému v krabici.

Obr. B.2: Detail na vyvedenie káblov pre čítačku kariet z krabice.



Obr. B.3: Detail na čítačku kariet prilepenú o bok krabice.

Dodatok C

Zoznam použitých skratiek

AES Advanced Encryption Standard

AID Application Identifier

APDU Application Data Protocol Unit

API Application Programming Interface

BI-HWB Bakalársky predmet Hardwarová Bezpečnost na FIT ČVUT

CBC Cipher Block Chaining

DPA Differential Power Analysis

EEPROM Electrically Erasable Programmable Read-Only Memory

HMAC Keyed-Hashing for Message Authentication

I2C Inter-Integrated Circuit

IBSG Inteligentní Bezpečnostní Systém Garáže

IDE Integrated Development Environment

IOT Internet of Things

ISO International Organization for Standardization

LED Light Emitting Diode

MAC Message Authentication

NFC Near Field Communication

PCB Printed Circuit Board

REST Representational State Transfer

RFC Request for Comments

RFID Radio Frequency Identification

SAM Security Access Module

SHA Secure Hash Algorithm

SIM Subscriber Identity Module

SPA Simple Power Analysis

SPI Serial Peripheral Interface

UART Universal Asynchronous Receiver-Transmitter

UC Use Case

Dodatok D

Obsah priloženej SD karty

- `readme.txt` ..... stručný popis obsahu SD karty
- `impl` ..... zdrojové kódy implementácie
  - `arduino` ..... implementácia pre Arduino
    - `get_uid` ..... program na zistenie UID karty
    - `init_eeprom` ..... program na inicializáciu EEPROM
    - `libraries` ..... knižnice potrebné pre implementáciu
    - `Main` ..... zdrojové kódy pre IBSG
  - `javacard` ..... implementácia pre Java Card
    - `scripts` ..... inštalačné skripty
    - `src` ..... zdrojový kód pre Java Card
- `thesis` ..... text práce a jej zdroje
  - `images` ..... obrázky použité v práci
  - `latex` ..... pomocné súbory pre LaTeX
  - `BP_Macak_Maros_2019.pdf` ..... text práce vo formáte PDF
  - `BP_Macak_Maros_2019.tex` .... zdrojová forma práce vo formáte LaTeX