VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

MIKROSKOPICKÁ ANALÝZA ČIPOVÝCH KARET

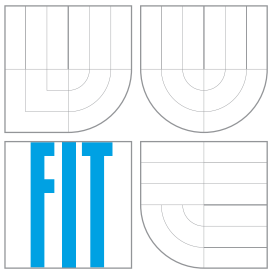
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN MICHÁLEK

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

MIKROSKOPICKÁ ANALÝZA ČIPOVÝCH KARET

MICROSCOPIC ANALYSIS OF SMART CARDS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN MICHÁLEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. DOMINIK MALČÍK
SUPERVISOR

BRNO 2012

Abstrakt

Bakalářská práce je zaměřená na popis ochraných prvků čipových karet. Obeznamuje s technologií karet a složením samotného čipu. Nalézt je zde možné také přehledný výčet konkrétních bezpečnostních prvků. V rámci práce byla také vytvořena knihovna pro anotaci logických elementů. Zpráva obsahuje návrh, implementaci a popis rozhraní této knihovny.

Abstract

The Bakchelor's thesis is aimed at description of protection features of smart cards. It introduces technology of card and anatomy of the chip itself. Further can be found transparent list of concrete security features. This works includes a library for anotation of logic elements. The thesis contains the design, implementation and interface of this library.

Klíčová slova

čipová karta, ochranné prvky, Qt, mikroskopická analýza, 2D kreslenie

Keywords

smart card, protection features, Qt, microscopic analysis, 2D drawing

Citace

Martin Michálek: Mikroskopická analýza čipových karet, bakalářská práce, Brno, FIT VUT v Brně, 2012

Mikroskopická analýza čipových karet

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Dominika Malčíka. Všechny literární prameny a publikace, ze kterých jsem čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

.....
Martin Michálek
13. května 2012

Poděkování

Chtěl bych se poděkovat vedoucímu této práce Ing. Dominiku Malčíkovi, který mi počas konzultací poskytoval své rady a připomínky, čím přispěl k skvalitnění této práce.

Obsah

Úvod **3**

1 Čipová karta **4**
1.1 História . . . 4
1.2 Typy kariet . . . 5
1.2.1 Kontaktné karty . . . 6
1.2.2 Bezkontaktné karty . . . 6
1.2.3 Pamäťové karty . . . 6
1.2.4 Mikroprocesorové karty . . . 7
1.3 ISO štandardy . . . 8
1.4 Anatómia čipu . . . 9
1.4.1 Kontakty čipovej karty . . . 9
1.4.2 Mikrokontrolér čipovej karty . . . 10

2 Ochranné prvky **14**
2.1 Ochrana fyzickej vrstvy . . . 15
2.1.1 Pasívne prvky . . . 15
2.1.2 Aktívne prvky . . . 17
2.2 Ochrana logickej vrstvy . . . 18
2.2.1 Prvky operačného systému . . . 18
2.2.2 Prvky aplikacií . . . 19

3 Nasnímanie čipu **20**
3.1 Vybavenia laboratória . . . 20
3.2 Vlastné snímanie čipu . . . 21

4 Návrh a implementácia knižnice **24**
4.1 Ciele . . . 24
4.2 Dátová reprezentácia . . . 25
4.2.1 Entity . . . 25
4.3 Vybrané riešenia vzniknutých problémov . . . 26
4.4 Rozhranie knižnice . . . 28
4.4.1 Všeobecné vlastnosti . . . 29
4.4.2 Špecifické vlastnosti a metódy . . . 30
4.4.3 Vytvorenie entít . . . 30
4.5 Implementačné prostriedky . . . 31
4.5.1 Qt . . . 31

5 Výsledky **32**
5.1 Demonštračná aplikácia 32

6 Záver **35**

A Obsah CD **37**

Úvod

Obrovský rozmach mikroelektroniky v sedemdesiatych rokoch zmenil fungovanie celej vyspelej spoločnosti. Pre svoje vlastnosti prenikli integrované obvody aj do plastových identifikačných kariet. Prvý výrazný nárast použitia nastal ich využitím v telefónnych automatoch vo Francúzsku a Nemecku. Využitie čipových kariet či už v oblasti zdravotníctva, alebo u mobilných operátorov, spôsobilo, že sa stali prirodzenou súčasťou moderného života.

Vďaka využitiu procesora sú aplikačné možnosti čipových kariet obrovské. Pokrývajú veľké množstvo rôznych oblastí ako napríklad kontorola prístupu do strážených priestorov, využívanie predplatených služieb, autorizácia pri finančných transakciách či ochrana pri verejnej komunikácii. Neustále zvyšovanie výkonnosti čipu prispieva k ešte výraznejšiu rozširovaniu do nových oblastí použitia.

Ruka v ruke s nárastom používania čipových kariet narastá aj snaha o hlbšie preskúmanie spôsobu ich fungovania. Zámerom výrobcov je samozrejme takéto počínanie čo najviac zťažiť a neohrozovať tak bezpečnosť informacií na karte uložených.

Táto práca je zameraná na získavanie informácií o čipových kartách pomocou mikroskopickej analýzy. V jednoduchosti by sa dalo povedať, že ide o nazeranie na čip pomocou mikroskopu s úmyslom pochopiť tak jeho fungovanie. Cieľom práce je zhromaždiť teoretické informácie o ochranných prvkoch čipu (viz kapitola 2) a zužitkovať ich pri získaní obrazových dát pomocou mikroskopu (kap. 3). Ďalším cieľom je vytvoriť programovú knižnicu pre anotáciu logických elementov a záujmových bodov využitelnú v budúcich aplikáciách. Jej návrhu a implementácii sa bude venovať kapitola 4.

Kapitola 1

Čipová karta

Čipová karta sa ukazuje byť ideálnym médiom. Poskytuje relatívne vysoký stupeň bezpečnosti, založenej na kryptografii. Vďaka procesoru môže bezpečne ukladať informácie a vykonávať rôzne algoritmy. Čipová karta je malá, ľahko použiteľná a môžeme ju so sebou nosiť kdekoľvek. Súčasné karty obsahujú čip, ktorý je v podstate zmenšeninou bežného stolového počítača. Spustený je tu operačný systém a v ňom aplikácie zabezpečujúce požadovanú funkčnosť.

V nasledujúcej kapitole bude predstavená história čipových kariet, ich výhody oproti iným druhom kariet, a vysvetlené bude aj delenie samotných čipových kariet. Ďalej budú v krátkosti popísané hlavné štandardy, zabezpečujúce univerzálnosť kariet naprieč rôznymi výrobcami. Druhá polovica kapitoly sa venuje hardvérovéj časti čipu. Objasnený je tu význam jednotlivých povrchových kontaktov karty a funkcie procesora, ako i ďalších významných komponentov čipu.

1.1 História

Prvé plastové karty sa vyskytli v Spojených štátoch na začiatku päťdesiatych rokov. Boli oveľa vhodnejšie na každodenné používanie ako dovtedy používané papierové, či lepenkové karty. Ich masovému rozšíreniu napomohla hlavne nízka cena výrobného meteriálu (PVC). Prvé celo-plastové platobné karty boli vydané spoločnosťou Diners Club v roku 1950. Boli znakom určitej spoločenskej prestíže, keďže držiteľ za služby v reštaurácii, alebo v hotely neplatil hotovosťou, ale svojim „dobrým menom“. Prvé karty slúžili ako médium pre uloženie informácií, chrániac ich tak proti falšovaniu a pozmeňovaniu. Všeobecné údaje ako meno vydavateľa karty, boli vytlačené na povrchu, zatiaľ čo osobné informácie, ako meno držiteľa, alebo číslo karty, boli embosované. Karty mali taktiež miesto pre vzorový podpis vlastníka. Ochrana proti zneužitiu, využívala vizuálne prvky a teda priamo záležala na svedomitosti osoby, ktorá karty akceptovala pri platení. Ako sa používanie kariet rozširovalo, narastali aj straty pre vydavateľov kariet, spôsobené nesolventnosťou zákazníkov.

Prvé výrazné vylepšenie spočívalo v zavedení magnetického prúžku na zadnej strane karty, ktoré umožňovalo uloženie informácií vo forme vhodnej pre strojové spracovanie, ako náhrada za vizuálnu kontrolu. Podpis bol nahradený tajným osobným identifikačným číslom (PIN - personal identification number), ktorého hodnotu by mal vedieť len majiteľ karty. Veľkou slabosťou ale je, že dáta uchovávané na magnetickom páse, môže ktokoľvek s vhodným vybavením čítať, mazať, alebo prepisovať. [8, 9]

Vznik čipových kariet umožnil veľký pokrok v oblasti mikroelektroniky v sedemdesia-

tych rokoch. Tie spájajú ukladanie dát, ako aj logiku pre spracovanie na jeden kremíkový čip s rozmermi len niekoľko milimetrov štvorcových. Myšlienka začlenenia integrovaného obvodu do identifikačnej karty sa prvýkrát objavila u dvojice nemeckých vynálezcov Jürgen Dethloff a Helmut Grötrupp v roku 1968. O dva roky neskôr prišiel s podobným patentom aj japonec Kunitaka Arimura, viac sa však zameriaval na samotné uloženiu čipu v karte, než na širšie možnosti využitia [4]. Za začiatok éry čipových kariet je považovaný rok 1974, kedy si francúz Roland Moreno patentoval koncept pamäťovej karty. Jeho licencie používajú v podstate všetci dnešní výrobcovia čipových kariet. Francúzska firma Honeywell Bull v spolupráci s americkou Motorolou vyrobila v roku 1976 prvú čipovú kartu (bez procesoru) a v roku 1979 prvú smart kartu - teda čipovú kartu s procesorom. [4]

Veľký zlom nastal v roku 1984, kedy francúzksky Poštový a telekomunikačný úrad úspešne testoval telefónne karty s čipmi. Tie splnili všetky očakávania na vysokú spoľahlivosť a ochranu dát proti manipulácii. Podobný test sa uskutočnil aj v Nemecku o rok neskôr. Porovnávané boli telefónne karty, založené na rôznych technológiách - s magnetickým pruhom, holografické a čipové karty. Jednoznačným víťazom sa stali čipové karty. Opäť potvrdili značnú mieru spoľahlivosti a zabezpečenia proti manipulácii. K prvému veľkému nasadeniu teda došlo v podobe telefónnych kariet. Klient si v obchode zakúpil kartu s vopred predplatenou finančnou čiastkou, ktorá sa mu pri telefonovaní postupne znižovala. Pre telefonovanie z verejnej búdky nemusel mať klient pri sebe hotovosť v minciach.

Čipové karty poskytujú vysoký stupeň flexibility. Pri zavádzaní novej funkčnosti čipu netreba meniť ostatné prvky systému ako čítačku, alebo prístupový terminál. Vďaka ich univerzálnosti sa rýchlo rozšírili aj do iných oblastí a v súčasnosti plnia mnohé funkcie. Okrem iného sa využívajú aj pre kontrolu prístupu, ako platobné karty, zdravotné karty a SIM karty mobilných operátorov. Od 1. decembra 2012 by mali elektronický čip obsahovať aj novovydané občianske preukazy na Slovensku, tak ako je to v súčasnosti napríklad v Nemecku či Estónsku [7]. Dalšie možnosti využitia tejto technológie sú obrovské.

Čipové karty disponujú, v porovnaní s kartami s magentickým pruhom, niekoľkonásobne väčšou kapacitou pamäte pre uloženie informácií. V súčasnosti až do 256kB. Každou novou generáciou sa tento parameter ešte znásobí. Ich hlavnou výhodou je ale ochrana uložených dát pred neautorizovaným prístupom. Čip implementuje rôzne šifrovacie algoritmy. Dáta sú prístupné len cez sériové rozhranie, ktoré je kontrolované operačným systémom a bezpečnostnou logikou. Na čipe sú uložené v podobe, ktorá zabraňuje ich násilnému vyčítaniu a následnému zneužitiu. Využitím softvérových a hardvérových prvkov zabezpečenia je možné vytvoriť nespočetné kombinácie ochranných mechanizmov. Významne to sťažuje prácu prípadného útočnika a predlžuje čas potrebný pre prelomenie karty. Životnosť kariet s magnetickým prúžkom sa pohybuje od jedného do maximálne dvoch rokov. Naproti tomu čipové karty majú dobu užívania stanovenú na tri roky. Po tomto období hrozí, že bezpečnostne zastarajú. [8]

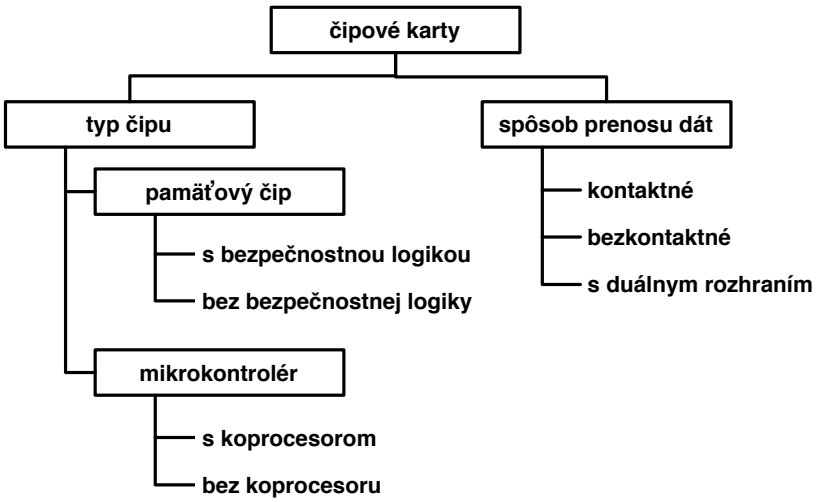
1.2 Typy kariet

Základným prvkom je integrovaný obvod, vsadený do karty, ktorý má zložky pre prenos, ukladanie a spracovanie dát. Dáta môžu byť prenášané s využitím kontaktov na povrchu karty, alebo pomocou elektromagnetického poľa, teda bez akéhokoľvek kontaktu. Vyšší výpočtový výkon, flexibilita a pamäť samozrejme zvyšujú aj cenu. Najčastejšie sú teda využívané len jednoúčelné karty.

Čipové karty sú definované podľa dvoch základných oblastí:

- ako sú dáta z karty čítané a zapisované
- aký typ čipu sa v karte nachádza a aké sú jeho funkcie

K dispozícii je teda široká škála možností pri výbere návrhu karty. Podrobnejšie rozdelenie je znázornené na obrázku 1.1.



Obr. 1.1: Rozdelenie typov kariet. Obrázok bol vytvorený podľa predlohy v [8].

1.2.1 Kontaktné karty

Sú najbežnejším typom čipových kariet. Elektrické kontakty sa nachádzaju na povrchu a sú spojené s čipom. Pre komunikáciu je potrebná čítačka kariet, do ktorej sa karta zasunie. Význam kontaktov je popísaný v sekcii 1.4.1.

1.2.2 Bezkontaktné karty

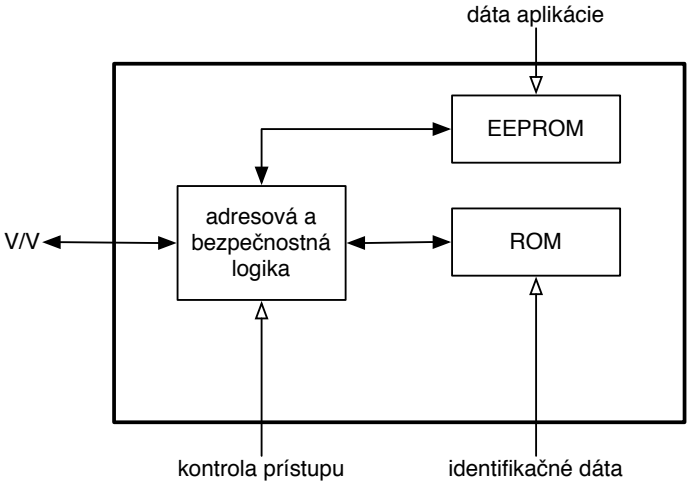
Jedná sa o čipové karty, ktoré pre spojenie a identifikáciu využívajú rádiové frekvencie (*RFID, angl. radio frequency identification*). Stačí teda s primeranou vzdialenosťou prejsť kartou okolo čítačky. Najčastejšie sú implementované ako read-only a využívajú sa pre kontrolu prístupu v budovách. V posledných rokoch sú využívané aj v obchodoch pri platení, keďže urýchľujú spracovanie transakcií oproti tradičným kontaktným kartám. [3]

1.2.3 Pamäťové karty

Pamäťové čipové karty si obľúbili telekomunikačné spoločnosti a boli to prvé čipové karty využívané naozaj vo veľkom. Tieto karty sú predplatené so sumou uloženou v čipe. Suma sa zníži o príslušnú čiastku, podľa ceny hovoru. Hlavnou výhodou je nemožnosť prepisovania dát útočníkom tak ako v prípade kariet s magnetickým pruhom. Pri kartách s magnetickým pruhom bolo možné si skopírovať dáta z magnetického pruhu ešte pred prvým použitím, kartu potom normálne používať a po určitom čase tieto dáta jednoducho prepísať predtým uloženými. Útočník tak získal kartu s „bezodným“ kreditom. Voči takémuto útoku je čipová

karta chránená vďaka bezpečnostnej logike, umiestnenej na čipe. Nie je možné zmazať určité pamäťové bunky potom, čo už boli raz zapísané, a takto upravenú kartu znovu použiť.

Tento typ kariet je samozrejme využívaný aj v iných službách, ktoré sú na predaj, na základe predchádzajúcej úhrady. Príkladom môže byť verejná doprava, parkovné automaty, samoobslužná jedáleň atď. Výhodou je jednoduchá technológia a nízka cena výroby. Nevýhodou je nemožnosť znovupoužitia karty po vyčerpaní kreditu. Použitá karta je jednoducho zničená a záujemcovi je vydaná nová. [3]



Obr. 1.2: Architektúra typickej kontaktnej pamäťovej karty s bezpečnostnou logikou. Obrázok bol vytvorený podľa predlohy v [8].

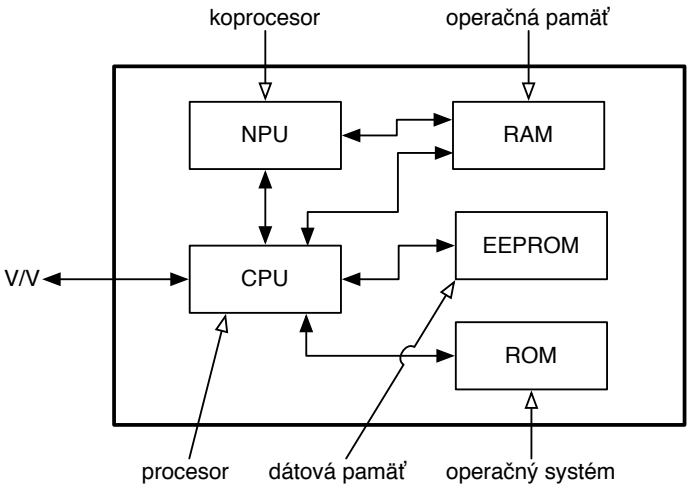
Na obrázku 1.2 sú zobrazené hlavné komponenty pamäťovej karty. Pamät typu EEPROM je úložisko pre dáta aplikácie. Prístup do pamäte je kontrolovaný bezpečnostnou logikou. V najjednoduchších kartách logiku tvorí iba ochrana proti zápisu. V pokročilejších pamäťových čipoch je prítomná komplexnejšia logika, ktorá zahrňuje aj jednoduché šifrovanie. Dáta sú prenášané cez V/V port, pomocou špeciálneho synchrónneho protokolu, vďaka ktorému je zachovaná jednoduchosť čipu.

1.2.4 Mikroprocesorové karty

Mikroprocesorové čipové karty boli prvýkrát nasadené vo forme bankových kariet vo Francúzsku. Vďaka ich schopnosti úschovy súkromných kľúčov a vykonávaniu moderných kryptografických algoritmov, bolo možné zaviesť vysoko bezpečný platobný systém. K ďalšiemu výraznému rozvoju tejto technológie, došlo po rozhodnutí využívať mikroprocesorové karty v mobilných telefónoch. Bez nich by bolo len s obtiažou možné, pre telefónnych operátorov, predávať svoje služby a telefóny oddelene.

V mikroprocesorových kartách môže byť spustených viacero aplikácií súčasne, pričom sú riadené operačným systémom. Okrem už spomínaných oblastí je ďalšie typické využitie kariet napríklad ako identifikácia osôb, kontrola prístupu, elektronický podpis a bezpečné uložisko dát. Moderné čipové karty s operačným systémom dovoľujú nahrať aplikáciu aj na kartu, ktorá už bola vydaná užívateľovi, a to pri zachovaní celkovej bezpečnosti čipu. Táto schopnosť otvára celkom nové možnosti vyžívania kariet. Okrem flexibility je výhodou mikroprocesorových kariet aj veľká úložná kapacita.

Ako ukazuje obrázok 1.3, srdcom mikroprocesorovej karty je procesor obklopený niekoľkými funkčnými blokmi. Ich význam a funkcie sú podrobne popísané v kapitole 1.4.2.



Obr. 1.3: Architektúra typickej kontaktnej mikroprocesorovej karty s koprocesorom. Obrázok bol vytvorený podľa predlohy v [8].

Mikroprocesorové čipové karty sa bežne označujú ako čipové karty. Táto práca je zameraná práve na tento druh kariet. V nasledujúcich častiach sa pri slovnom spojení čipová karta bude predpokladať čipová karta s mikroprocesorom. Informácie pre túto podkapitolu boli čerpané z [3, 5, 8].

1.3 ISO štandardy

Jednou z hlavných výhod technológie čipových kariet, je ich široká možnosť využitia. Každú kartu by bolo možné prispôsobiť pre potreby určitého systému, ktorý zabezpečuje služby s kartou spojené. Karta by sa však dala využiť len pre tento systém. S inými by bola nekompatibilná. V praxi je ale vhodné definovať určité spoločné rozhranie medzi kartou a zvyškom systému, nezávislé od jeho využitia. Medzinárodne uznávaný štandard pre kontaktné čipové karty je ISO-7816 [10]. Je to rodina noriem, popisujúcich všetko od fyzikálnych vlastností, cez komunikáciu s terminálom, až po zabezpečenie.

ISO 7816-1 Popisuje fyzikálne vlastnosti kariet ako rozmer, uloženie čipu, odolnosť proti statickej elektrine atď.

ISO 7816-2 Definuje umiestnenie a význam jednotlivých kovových kontaktov na karte. Jeho plnenie je dôležité pre kompatibilitu medzi jednotlivými terminálmi od rôznych výrobcov.

ISO 7816-3 Definuje elektrické signály a protokol prenosu.

ISO 7816-4 Signály pre komunikáciu ako zápis, čítanie, alebo aktualizáciu dát na čipe.

ISO 7816-11 Rozširuje osobnú identifikáciu uživateľa o možnosť využitia biometrických údajov.

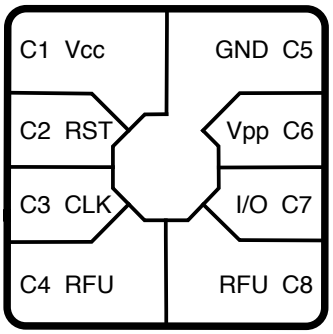
ISO 7816-15 Špecifikuje zabezpečenie karty - syntax a formát, šifrovaných infromácií na karte, popisuje rôzne techniky overovania užívateľa a viacero šifrovacích algoritmov.

1.4 Anatómia čipu

Mikroprocesor tvorí srdce čipovej karty. Skladá sa z množstva logických jednotiek sústredených na malú plochu kremíkového čipu. Komunikáciu s okolím zabezpečujú pre čip kontakty vyvedené na povrch karty.

1.4.1 Kontakty čipovej karty

Mikroprocesor čipovej karty potrebuje pre svoju činnosť elektrické napájanie, externý hodinový signál ako aj V/V kanál. Tieto a ďalšie funkcie zabezpečuje šesť, alebo osem pozlátených plôšok na povrchu plastovej karty. Umiestnenie a význam kontaktov je definovaný normou ISO 7816-2. Konkakty C4 a C8 nemajú zatiaľ v štandarde priradený význam, sú rezervované pre budúce použitie. [10]



Obr. 1.4: Kontakty čipovej karty. Obrázok bol vytvorený podľa predlohy v [11].

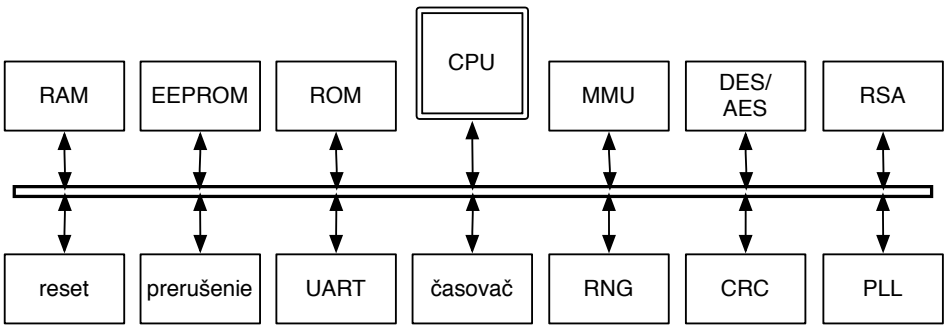
Význam kontaktov čipovej karty (viz obr. 1.4):

- Vcc - prívod napätia, ktorým je napájaný mikroprocesor.
- RST - kanál, ktorým môže terminál poslať procesoru signál reset.
- CLK - prívod hodinového signálu z terminálu.
- GND - uzemnenie (referenčné napätie) medzi terminálom a čipom.
- Vpp - programové napájanie používané pre programovanie EEPROM pamäte v prvej generácii čipov. Dnes sa už nepoužíva.
- I/O - vstupno/výstupný kanál pre sériovú komunikáciu medzi terminálom a čipovou kartou.

1.4.2 Mikrokontrolér čipovej karty

Mikrokontrolér, alebo čip je hlavný komponent čipovej karty. Je ukrytý pod kontaktmi. Spúšťa, kontroluje a monitoruje všetky procesy, ktoré na karte prebiehajú. Mikrokontrolér bol špeciálne vyvinutý pre svoj účel. Čip musí využívať napätie 3V, alebo 5V a sériové rozhranie. Musí obsahovať prepisovateľnú pamäť, ktorá ale na udržanie informácie nepotrebuje napájanie. Dôležitým parametrom je veľkosť čipu. Väčší čip zaberie pri výrobe viac miesta na kremíkovej doske a je teda drahší. Väčší čip je rovnako viac náchylný na poškodenie, spôsobené prehnutím plastovej karty, v ktorej je uložený. Čip má preto štvorcový pôdorys s hornou hranicou plochy 25 mm^2 [6]. V dôsledku nutnosti integrovať všetku funkcionalitu do pomerne malého čipu je výroba technicky náročná, a existuje teda len zopár špecializovaných výrobcov. Výrobcovia nedodávajú čipy na bežný trh, ale dodávajú ich odberateľom priamo. V opačnom prípade by výrazne uľahčili pokusy o analýzu hardvéru čipu a znížili tak jeho bezpečnosť.

Typické komponenty mikrokontroléru sú zobrazené na obrázku 1.5. Najdôležitejšími sú procesor, adresová a dátová zbernica a tri druhy pamätí. Procesor (CPU) sa stará o vykonávanie inštrukcií. Pamäť typu ROM je neprepisovateľná a je v nej uložený operačný systém. RAM pamäť je operačná pamäť, teda sú v nej uložené výsledky a medzivýsledky operácií, vykonávané procesorom. Po vypnutí napájania, čiže po vytiahnutí karty z čítačky, sa dáta stratia. Pre dlhodobé uchovanie dát je na čipe EEPROM pamäť. Čip má ďalej V/V rozhranie, ktoré zabezpečuje sériovú komunikáciu s okolím. V najjednoduchších čipoch je toto rozhranie oblasť adresovateľná procesorom a napojená na V/V kontakt. Súčasťou môže byť aj tzv. dodatočný hardvér, ktorý zabezpečuje na čipe funkcie, ktoré z rôznych dôvodov nemôžu byť implementované softvérovo. Jedným z hlavných dôvodov je rýchlosť spracovania operácie. Prítomnosť jednotlivých komponentov závisí od predpokladanej oblasti použitia čipovej karty.



Obr. 1.5: Typické komponenty mikroprocesoru čipovej karty. Obrázok bol vytvorený podľa predlohy v [8].

Jednotlivé komponenty mikroprecesoru čipovej karty (viz obr. 1.5):

- CPU - procesor.
- RAM, EEPROM, ROM - typy použitých pamätí.
- MMU - jednotka správy pamäte (*angl. memory management unit*).
- DES/AES, RSA - koprocesory pre kryptografické algoritmy.

- PLL - generátor fázového závesu (*angl. phase-locked loop*).
- CRC - výpočet kontrolného cyklického kódu (*angl. cyclic redundancy check*).
- UART - univerzálny asynchrónny prijímač-vysielač (*angl. universal asynchronus receiver-transmitter*).
- RNG - generátor náhodných čísel (*angl. random number generator*).

Procesory Procesory používané v čipových kartách nie sú špeciálne vyvíjané pre tento účel. Vývoj úplne nového procesora by bol extrémne nákladný a zdĺhavý, preto sa upravujú už existujúce a teda odskúšané typy procesorov. Za vzor sa väčšinou berú procesory zaostávajúce o dve generácie, oproti súčasnému stavu [8]. S ohľadom na požadovaný výkon však plne postačujú. Týmto postupom sa zachováva požadovaná vysoká spoľahlivosť procesoru. Pri výbere vhodného procesora sú kritériami veľkosť kódu, stratový výkon a odolnosť voči útokom.

Základom sú 8 bitové procesory. Obyčajne disponujú úplnou inštrukčnou sadou CISC (complex instruction set computer). Inštrukčná sada je založená na architektúre Motorola 6805, respektíve Intel 8051. Procesory môžu obsahovať aj pridané inštrukcie ako napríklad podpora pre 16bitové adresovanie pamäte. Rovnako môžu obsahovať rozšírenia, ktoré im umožňujú prekročit 64 kB limit pamäte. Prístup do takýchto pamäťových blokov je realizovaný cez špeciálny register, ktorý ich mapuje do špecifických oblastí pôvodnej pamäte. Takéto nelineárne mapovanie pamäte však výrazne sťažuje jej adresovanie a zvyšuje náchylnosť k chybám.

Vylepšením predchádzajúcej architektúry sú 16 bitové procesory. Využívajú redukovanú inštrukčnú sadu RISC (reduced instruction set computer). Dlhú dobu existoval len jediný takýto procesor - H8 od firmy Renesas.

Čoraz častejšie sa využívajú 32 bitové procesory. Poskytujú potrebný výkon pre moderné operačné systémy čipových kariet. Rovnako dokážu obsluhovať dostatočne veľkú operačnú pamäť. Príkladom je ARM 7 procesor, ktorý sa pôvodne využíval vo video kamerách. Pre svoje parametre sa dostal až do čipových kariet. Po optimalizácii dostal meno SC 100. Je napájaný 3V napätím a dosahuje frekvenciu až 20 MHz. Je vyrábaný $0,8\,\mu m$ technológiou a zaberá len 5,9 mm^2. Pridaním matematického koprocesora pre urýchlenie kryptografických operácií narastie plocha o ďalšie 2 mm^2. [5, 8]

Pamäte V mikrokontroléri slúžia pamäte na uchovávanie kódu programu a dát. Tri hlavné typy sú ROM, RAM a EEPROM, a spolu môžu zaberať až 2/3 plochy celého čipu. Kapacita jednotlivých typov pamätí záleží od predpokladanej oblasti využitia mikrokontroléru. V každom prípade sa ale výrobca snaží zredukovať EEPROM a RAM pamäť, pretože zaberajú najviac plochy pre uloženie jedného bitu informácie. V prípade čipových kariet, na ktorých môže byť spustených viacero aplikácií naraz, je kapacita ROM pamäte dvojnásobná oproti EEPROM. ROM pamät tu slúži na uloženie kódu operačného systému. V prípade karty s možnosťou spustenia len jedinej aplikácie, je využívaná prevažne EEPROM, ktorá poskytuje priestor pre uloženie kódu programu ako aj ostatných dát. Jej veľkosť je šitá na mieru konkrétnej aplikácii. Jednotlivé typy pamätí zaberajú pre ich rozdielne princípy uchovávania informácie a úlohy použitia na čipe rôzne veľkú plochu. Pamäť RAM zaberá 4x viacej miesta ako EEPROM, ktorá zase zeberá 4x viacej miesta ako ROM pamäť. Pri 4 kB RAM pamäte by teda na rovnakej ploche mohlo byť 16 kB EEPROM alebo dokonca 64 kB ROM pamäte [8]. Toto

je dôvod, prečo majú mikrokontroléry zdanlivo tak málo operačnej pamäte. V najnovších čipoch sa EEPROM pamäť nahrádza tzv. flash pamäťou, ktorá má rýchlosť zapisovania a mazania výrazne rýchlejšiu a zaberá len polovicu plochy.

RAM pamäť je operačná pamäť procesoru, je prepisovateľná a na udržanie zapísaných dát potrebuje elektrický prúd. Dáta sú v nej uložené len počas pripojenia karty k terminálu. RAM v čipových kartách je statická (SRAM), čo znamená, že dáta nemusia byť periodicky obnovované. Je dôležité, aby bola pamäť statická, a bolo tak možné zastaviť hodinový signál z čipovej karty v ľubovolnej chvíly. Pri dynamickej pamäti by hrozila strata informácií. Po vybratí karty z terminálu sa dáta stratia.

EEPROM pamät je prepisovateľná a pre dlhodobé uloženie dát nepotrebuje napájanie. EEPROM je využívaná pre všetky dáta a programy, ktoré musia byť v priebehu používania karty mazané, alebo zmenené. V analógii so stolným počítačom by EEPROM pamäť bola jeho pevným diskom.

Z ROM pamäťe môžu byť dáta iba čítané, nie zapisované. Keďže nepotrebuje napájanie pre udržanie informácií, dáta sú v nej uschované, aj keď nie je karta práve používaná. V ROM je teda uložený operačný systém a sada diagnostických nástrojov. Tieto programy sú do pamäte vypálené pri výrobe.

Jednotka správy pamäte Ako už bolo spomenuté, najnovšie operačné systémy umožňujú nahrať spustitelný kód do karty, aj po jej vydaní užívateľovi. Toto umožňuje napríklad pridať kryptografické algoritmy, ktoré pozná len užívateľ. Úlohou MMU je dozerať na to, aby pridaná aplikácia nepristupovala k tajným informáciám a nespôsobila tak bezpečnostnú hrozbu. Aplikácia sa spúšťa špeciálnym príkazom. Pred spustením aplikácie predá operačný systém MMU jej prístupové práva k pamäti. MMU následne dozerá, aby aplikácia pristupovala len do povolených miest pamäte. [8]

Ďalšou funkciou MMU je schopnosť premapovať fyzické adresové oblasti do ľubovoľných logických oblastí. Operačný systém potom pred spustením aplikácie nemusí, v prípade nutnosti, premapovávať spustitelný kód. Táto schopnosť sa dá využiť aj pre striktné odelenie adresových priestorov jednotlivých spustených aplikácii.

Koprocesor pre symetrické kryptografické algoritmy Symetrický algoritmus je štandardom v systéme finančných transakcií, ako aj v telekomunikáciach. V začiatkoch sa používal DES algoritmus a jeho časté využívanie teda viedlo k myšlienke vyčleneniť ho do samostatnej hardvérovej jednotky. Veľkosť tejto jednotky na čipe je rovnaká, ako by zabrala ROM pamäť pre jeho softvérovú implementáciu. DES algoritmus je však dnes prelomený a preto ho nahradil bezpečnejší AES.

Koprocesor pre asymetrické kryptografické algoritmy Koprocesor zabezpečuje, na rozdiel od koprocesoru pre symetrickú kryptografiu, len niekoľko základných operácií, ktoré sa v asymetrickej kryptografii využívajú. V podstate je optimalizovaný na urýchlenie len dvoch operácií, a to umocňovanie a modulo (zvyšok po celočíselnom delení) pracujúce nad veľkými číslami. Výpočet prebieta tak, že procesor uloží dáta do RAM, nad ktorými chce vykonať príslušné operácie. Následne zavolá aritmetickú jednotku, ktorej predá ukazovateľ na uložené dáta. Potom, čo je požiadavka spracovaná, je výsledok opäť uložený do RAM a kontrola nad čipom je predaná naspäť procesoru. V súčasnosti vedia tieto koprocesory spracovať dĺžky kľúčov do 1024 bitov pre RSA algortimus. [8]

Jednotka pre výpočet kontrolného cyklického súčtu Všetky ukladané dáta sú pre jednoduchšiu detekciu prípadných chýb chránené CRC súčtom. Pre množstvo potrebných bitových operácií je však softvérový výpočet dlhých vstupov príliš pomalý. Kvôli častému využívaniu tejto funkcie bolo teda nutné implementovať ju hardvérovo.

Univerzálny asynchrónny prijímač/vysielač Táto jednotka poskytuje spojenie mikrokontroléru s vonkajším svetom. Pri najlacnejších čipoch je toto V/V rozhranie kontrolované priamo procesorom. Problémom pri softvérovom riešení je rýchlosť komunikácie, ktorá záleží od rýchlosti samotného procesora. Nadväzujúcim problémom je vyťaženie procesora. UART zabezpečuje komunikáciu nezávislú od procesora. Najnovšie verzie zabezpečujú komunikáciu priamo s RAM, s využitím priameho prístupu do pamäte. [8]

Generátor náhodných čísel Náhodné čísla sú potrebné pri generovaní kľúčov, potrebných pre autentifikáciu karty a terminálu. Pri softvérovom generovaní je možné generovať len pseudo-nahodné čísla. Z bezpečnostných dôvodov je vhodnejšie, aby tieto čísla boli naozaj náhodné. Všetky nové čipy preto obsahujú hardvérový generátor. Takýto generátor by nemal využivať ku generovaniu fyzikálne vplyvy okolia ako sú teplota, alebo vstupné napätie, pretože by mohli byť útočníkom ľahko ovplyvniteľné. Generátor často využíva rôzne logické stavy procesora ako napríklad hodinový signál, alebo stav pamäte. Tieto údaje sú ukladané do lineárneho spätnoväzobného posuvného registra. Register je riadený hodinovým signálom, ktorý je opäť generovaný na základe niekoľkých iných parametrov. Vylepšenie kvality generovaných čísel možno dosiahnuť pridaním ďalších algoritmov.

Informácie boli čerpané zo zdrojov [5, 8]

Kapitola 2

Ochranné prvky

V tejto kapitole budú popísané v súčastnosti využívané ochranné prvky čipovej karty. Sú rozdelené podľa zamerania útokov na samotný hardvér čipu, teda jeho fyzickú úroveň, alebo programové vybavenie, teda logickú úroveň.

Nie je možné navrhnúť systém, ktorý bude dokonalý, a zároveň bude chrániť čip pred akýmkoľvek útokom. Naopak, počíta sa s tým, že útočníkovi sa pri maximalizovaní snahy, skôr či neskôr, podarí odhaliť istú bezpečnostnú chybu. Každý útočník útočí na kartu s nejakým zámerom. Môže to byť peňažný zisk, alebo osobné pohnútky ako zvyditeľnenie sa, či zvýšenie prestíže. Z pohľadu útočníka, by mal byť pomer predpokladanej odmeny za prelomenie systému a úsilia nutného pri hľadaní chyby, čo najpriaznivejší. Cieľom návrhu ochrany čipovej karty je práve tento pomer čo najviac zneatraktívniť. Len málo kto bude investovať svoj čas a peniaze potrebné k prelomeniu ochrany karty, ak mu to neprinesie náležitú odmenu.

Zabezpečenie ochrany čipovej karty má niekoľko špecifík. Pre samotnú kartu je ťažké, až nemožné rozoznať, že je práve pod útokom. Karta taktiež nemá vlastné napájanie. Kariet sú v obehu milióny, a nie je prakticky možné pri odhalení závažnej chyby požiadať uživateľov o ich rýchlu výmenu. Pri odhalení návrhovej chyby, alebo slabiny v hlavnom systéme čipovej karty, možno predpokladať jej rýchle šírenie pomocou internetu do celého sveta behom pár dní. Pomerne rýchlo po odhalení je možné zakúpiť vhodný softvér a hardvér pre opakovanie útoku. Často je dostupná aj podrobná dokumentácia, takže zopakovať prienik je pomerne ľahko zvládnuteľné takmer pre každého.

Ochrana čipovej karty sa skladá z dvoch zložiek. Prvou je samotné telo karty, kde je čip uložený. Tvorí ju gravírovanie, embosovanie, prípadne biometrické údaje ako napríklad fotka tváre. Slúžia na autentifikáciu užívateľa, teda overenie, že je ten, za koho sa vydáva. Ak sa karta využíva v prostredí, kde nie je nutná vizuálna kontrola človekom, táto časť ochrany odpadá. V tejto práci sa zameriam na druhú zložku ochrany, teda ochrana čipu a informácií uložených na ňom. Skladá sa z hardvéru čipu, operačného systému a aplikácií. Spoločne sa starajú o ochranu dát a programov uložených na čipe. Dôležitá je spolupráca všetkých prvkov, pri prelomení ktoréhokoľvek z nich sa ochrana ostatných prvkov výrazne oslabí.

Útoky sú rozdelené na tie, ktoré napádajú samotný hardvér čipu, a tie, ktorých cieľom je preniknutie do systému čipovej karty cez jeho logickú úroveň. Fyzické útoky a analytické metódy možno rozdeliť na statické a dynamické. Pri statickej analýze môže byť čip napájaný, ale je mimo prevádzky. Pri dynamicej analýze je čip sledovaný pri plnej prevádzke. [5, 8]

2.1 Ochrana fyzickej vrstvy

Útok na fyzickú vrstvu čipu vyžaduje široké technické zázemie. Medzi základné vybavenie patrí výkonný mikroskop, prostriedky pre chemické leptanie a rýchle počítače pre analýzu, zaznamenávanie a výpočty elektrických procesov na čipe. Vybavenie, časová náročnosť, ale hlavne znalosti potrebné k uskutočneniu takéhoto útoku ho obmedzujú zväčša len na špecialistov a organizácie, skúmajúce bezpečnosť čipových kariet. Spoločnosti vyrábajúce čipy však musia počítať aj s takýmto druhom útoku a zabezpečiť primeranú ochranu.

Ochranné opatrenia na hardvéry možno rozdeliť medzi aktívne a pasívne komponenty. Pasívna ochrana je tvorená už pri výrobe čipu. Má za úlohu chrániť hlavne pamäť a funkčné časti čipu proti rôznym druhom analýz. Aktívne komponenty zahŕňaju rôzne senzory integrované do čipu. Senzory sú v prípade potreby využívané softvérom na čipe. Príkladom môže byť teplotné čidlo. Pri zvýšení teploty sa okamžite zmaže celá EEPROM pamäť.

Pri písaní tejto podkapitoly boli využité zdroje [5, 8, 6].

2.1.1 Pasívne prvky

Fiktívne štruktúry Sú to časti čipu, ktoré nemajú žiadnu praktickú funkciu. Ich úlohou je pomýliť útočníka. Pre zvýšenie bezpečnosti môžu byť vybavené senzormi, ktoré monitorujú dianie v okolý čipu (viz časť 2.1.2). Ich hlavnou nevýhodou je však zabratie miesta na čipe, ktoré by mohlo byť využité efektívnejšie. [6]

Zbernice čipu Interné zbernice, ktoré spájajú procesor s troma hlavnými typmi pamäte, nie sú vyvedené mimo čip. Nie je teda možné jednoduché napojenie na zbernicu, a umožniť tak útočníkovy odchytávať dáta, ktoré po nej prechádzajú. Pre prípad zasahovania útočníka priamo do čipu, sú zbernice vedené v jeho nižších vrstách, nie priamo po povrchu. Ďalším opatrením sú zmeny funkcie zberníc. Funkcia konkrétnej zbernice sa môže líšiť aj s reláciou karty a terminálu. Je v podstate nemožné, pozorovaním rozoznať, ktorá zbernica má aktuálne akú funkciu. Pravidlá pre zmenu funkcií zberníc sú dané maskou pri výrobnom procese.

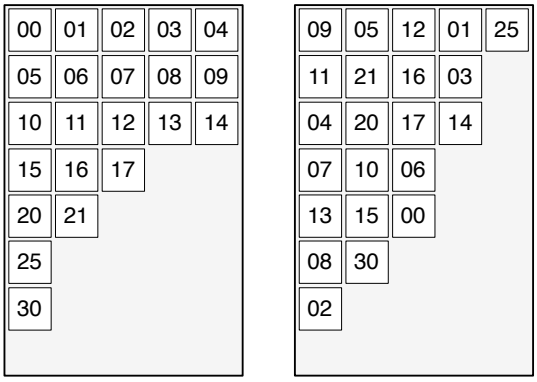
Dizajn pamäte Bežne používaná pamäť typu ROM, môže byť jednoducho čítaná bit po bite použitím optického mikroskopu. Nie je teda zložité zistiť kompletný kód programu, ktorý je v nej umiestnený. Pamäť je preto umiestnená v nižších vrstvách čipu. Pri odňatí vrchných vrstiev čipu by ale bolo možné sa k pamäti nakoniec dostať. Z tohto dôvodu je v mikroprocesoroch čipových kariet používaná ROM pamäť s iónovou implementáciou, pri ktorej už nie je možné rozoznávať jej obsah použitím viditeľného, alebo ultrafialového svetla.

Ochranné vrstvy Meranie elektromagnetického napätia na povrchu čipu predstavuje značnú hrozbu. Pri vhodnej technike možno merať jednotlivé časti čipu a odhadnúť tak obsah RAM pamäte, zatiaľ čo je čip v prevádzke. Táto technika môže byť veľmi ľahko eliminovaná pridaním vodivých vrstiev nad citlivé oblasti s pamätou. Vrstvy zároveň slúžia pre rozvod elektrickej energie pre čip. Ich odstránením by čip prestal fungovať správne. Každá dôležitá oblast čipu môže byť pokrytá niekoľkými takýmito vrstvami, ktoré môžu navyše obsahovať senzory. Pri ich poškodení prestane daná časť čipu okamžite pracovat.

Aj keď informácie z RAM pamätí sa stratia okamžite po odpojení napájania, existuje technika, ktorá umožní dáta z pamäte uchovať aj dlhý čas potom. Jednoducho stačí

pamäť schladiť na -60 °C. Pri využití sofistikovaných metód a elekrónového mikroskopu je potom možné vyčítať obsah jednotlivých buniek. Z tohto dôvodu sú citlivé dáta uložené v pamäti len nevyhnutnú dobu a potom sú okamžite mazané alebo prepisované inými. [8]

Kódovanie pamäte (angl. memory scrambling) Táto technika vychádza z obdoby používanej u zberníc spomenutej vyššie. Základom je, podľa určitého kľúča, poprehadzovať adresovanie pamäte tak, aby nasledujúce adresy neležali fyzicky vedla seba. Cieľom je utajiť schému pamäťových buniek a teda znemožniť útočníkovi rozpoznať adresovanie pamäte a jednoducho tak vyčítat jej obsah. Zakódovanie buniek je ľahko implementovateľné a väčšinou je riešené hardvérovo priamo obvodmi na čipe.



Obr. 2.1: Ukážka pamäte s lineárne vzrastajúcim adresovaním a pamäte, nad ktorou bolo použité kódovanie. Obrázok bol vytvorený podľa predlohy v [8].

Šifrovanie pamäte Moderné čipové karty poskytujú dávkové, alebo čipovo špecifické šifrovanie pamäte a niektorých procesorových registrov. Dáta musia byť šifrované a dešifrované pri operáciach zapisovania a čítania v reálnom čase. Pri niektorých čipoch je šifrovací kľúč odvodený od adresy pamäte. Rovnaké dáta nachádzajúce sa v iných častiach pamäte, majú po zašifrovaní inú hodnotu. Po uskutočnení úspešného vyčítania dát s pamäte, ostáva útočníkovi ešte zistiť šifrovací kľúč pre rozlúštenie dát. Útočník preto musí vedieť, v ktorých častiach pamäte tento kľúč hľadať, alebo systematicky vyčítať vsetky dáta z čipu.

Základnou myšlienkou obrany je, že ak je možné výčítať dáta z pamäte čipu, útočníkovi by sa mal tento proces sťažiť takou mierou, až by ho odradilo hľadať na karte PIN, alebo iný tajný kľúč. Napríklad hodnota PINu je uložená v zašifrovanej podobe ako referenčná hodnota pre porovnávanie. Pri jeho šifrovaní je použitý aj jedinečný kľúč karty, takže dve rovnaké hodnoty PINu sú zakódované inak na dvoch rôznych kartách. Ak má útočník prístup k celej pamäti, mohol by výčítať šifrovaciu funkciu a rovnako aj jedinečný kľúč karty. S týmito informáciami môže začat utočník hádať hodnotu PINu, ktorý sa skladá len zo štyroch číslic a existuje teda len 10 000 možností. Ako by sa mohlo zdať, použitie šifrovacej funkcie nemá príliš veľký význam. Jej cieľom je ale znásobiť úsilie potrebné pre odhalenie PINu. Vyčítanie celej funkcie je technicky a časovo neporovnateľne náročnejsie ako výčítať pár konkrétnych bitov

v ktorých je PIN uložený. Zakódovanie PINu pomocou jedinečného kľúča karty je preto pomerne rozšírené. [8]

2.1.2 Aktívne prvky

Monitorovanie pasivačnej vrstvy Pasivačná vrstva sa nachádza na vrchu mikrokontroléru. Je to pozostatok výrobného procesu a zabraňuje oxidácii. Jej ochranný význam spočíva v zakrytí celého čipu. Vrstva teda musí byť odstránená pred mikroskopickou analýzou čipu. Väčšinou je použité chemické leptanie. Vo vrstve sú taktiež zabudované snímače na meranie odporu, alebo kapacity. Pri zistení porušenia vrstvy je procesoru okamžite vyslaná žiadosť o prerušenie všetkých operácií a čip sa vypne.

Monitorovanie napätia Táto ochrana je prítomná na každej čipovej karte. V prípade vybočenia úrovne napätia mimo povolené limity, dôjde k okamžitému vypnutiu čipu. To zaručuje ukončenie operácii čipu pri hodnotách napätia, v ktorých by nemusel pracovať korektne. Bez tohto opatrenia by sa čítač inštrukcií mohol stať nestabilný, čím by dochádzalo ku vzniku množstva chýb v procesore. Takto vzniknuté chyby by mohli byť využité pri diferenciálnej chybovej analýze (*angl. DFA, differential fault analysis*) a odhaliť tak tajné kľúče. [8]

Monitorovanie frekvencie Mikrokontroler je vždy riadený externým hodinovým signálom, to znamená, že jeho frekvencia je generovaná mimo karty. Teoreticky by teda bolo možné signál spomaliť a spustiť čip v krokovacom móde. Tento mód by útočníkovi ponúkol rozsiahle možnosti pre skúmanie správania sa čipu, hlavne pomocou výkonovej analýzy. Pre zabránenie takýmto útokom je čip vybavený senzormi, ktoré zaznamenávajú vybočenie frekvencie z daných limitov. Vo väčšine špecifikácií je dolná medza frekvencie na úrovni 1 MHz a horná 5 MHz. Pre úspešné zabránenie krokovaniu je teda rovnako dôležité ochrániť aj samotné senzory frekvencie. Proti pokusu o ich vyradenie z činnosti sú pokryté niekoľkými ochrannými vrstvami.

Monitorovanie teploty Praktizuje sa len na niektorých typoch kariet, pretože jeho význam je diskutabilný. Pri náhlej zmene teploty mimo bežnú pracovnú teplotu čipu nedochádza k jeho okamžitému poškodeniu, alebo zmene chovania, ktoré by sa dalo využiť pre útok. Dôvod zavedenia teplotných senzorov je skôr snaha o nezvyšovanie miery poruchovosti, pri dlhodobej prevádzke čipu v nevyhovujúcich teplotách.

Kódovanie zbernice (*angl. bus scrambling*) Jedná sa o dôležitý ochranný prvok. Jednotlivé zbernice nie sú vedené jedna vedľa druhej, podľa dopredu známeho poradia, alebo čo najkratšou cestou. Dráha každej zbernice je vedená niekoľkými vrstavmi čipu. Funkcie jednotlivých zberníc sa taktiež náhodne menia. Útočníkovi sa teda výrazne znižuje šanca úspešne odpočúvať komunikáciu danej zbernice.

Táto technika bola najskôr predstavená len v, už spomenutej, statickej verzii. Schéma, podľa ktorej dochádza k náhodnej zmene funkcie zbernice, je tu rovnaká pre všetky čipy daného typu. Po dlhšom čase by teda pre útočníka nebolo ťažké postupne túto schému zistiť. Vylepšenie, ktoré sa dnes prevažne používa, je generovanie tejto schémy na základe obvodov v mikrokontroléri s využitím špecifického čísla čipu. Týmto postupom je kódovanie zbernice špecifické pre každy jednotlivý čip ako aj pre každú jeho reláciu. [8]

2.2 Ochrana logickej vrstvy

Útok na logickú vrstvu čipu predstavuje sledovanie komunikácie a dátového toku medzi terminálom a čipovou kartou. Nie je dôležité poznať hardvérovú vrstvu čipu, doležité je pochopiť, ako na karte prebiehajú jednotlivé softvérové operácie. Ochrana čipovej karty začína už pri správnom programovnom návrhu. Základnou chybou môže byť snaha o prílišnú optimalizáciu kódu. Príkladom je donedávna fungujúca časová analýza porovnávanie PINu. Pri jeho zadaní je táto hodnota porovnávaná na to navrhnutou funkciou. Programátor by túto funkciu mohol optimalizovať, aby už pri prvom rozdielnom znaku zadaného PINu a referenčnej hodnoty skončila a vrátila výsledok. Čas porovnania správnej hodnoty by bol teda dlhší ako porovnania zle zadanej hodnoty. Útočník by túto informáciu vedel využiť k rýchlemu uhádnutiu správnej hodnoty PINu. Stači merať čas medzi zadaním a odmietnutím PINu, podľa ktorého sa dá určiť, koľko prvých bitov bolo správnych. Pred pár rokmi bol tento typ útoku celkom efektívny, avšak na dnešné karty je už nepoužiteľný, kedže porovnávacia funkcia porovná vždy celý PIN. V nasledujúcom texte budú opísané najzákladnejšie ochranné prvky logickej vrstvy. Informácie boli získané z [6, 8].

Kryptografické algoritmy bez šumu

Niektoré kryptografické algoritmy potrebujú rôzne dlhý čas na zašifrovanie, alebo dešifrovanie rôznych vstupných hodnôt. To umožňuje útočníkovi, podľa času spracovania, odhadnúť vstupný reťazec a pokúsiť sa o jeho rozlúštenie hrubou silou. Ide o tzv. časový útok. Čas potrebný pre rozlúštenie tajného kľúča silno závisí od úrovne šumu použitého algoritmu. Ak je známy algoritmus použitý na karte, môžeme dopredu zostaviť referenčné tabuľky časov, potrebných pre rozlúštenie daných hodnôt, čím sa rýchlosť prelomenia kľúča ešte urýchli. Od objavenia tohto útoku, sú používané prevažne kryptografické algoritmy bez šumu. Ich čas potrebný pre krypografickú operáciu nezávisí od vstupnej hodnoty, čo zabraňuje tomuto typu útoku. Problémom ale je, že používajú dlhší kód a ich vykonávanie je vždy pomalšie.

2.2.1 Prvky operačného systému

Operačný systém poskytuje rozhranie medzi hardvérom čipu a aplikačnou vrstvou. Aplikácie sú tak ľahšie prenositeľné medzi rôznymi druhmi čipov. Z pohľadu bezpečnosti je zabezpečenie operačného systému rovnako dôležité, ako zabezpečenie aplikácií či hardvéru čipu. Operačný systém tvorí prostredie pre beh aplikácií, ktorých aktivity musí chrániť. Už pri inicializácii operačného systému je skontrolovaná funkčnosť základných komponentov čipu. V pamätiach sú prepočítané kontrolné súčty a overená platnosť uložených dát.

Oddelené vrstvy operačného systému Medzi jednotlivými vrstvami OS sú jasne definované parametre prechodu. Pri výskyte programovej, alebo návrhovej chyby v jednej vrste systému, sa znižuje šanca na jej prípadné využitie v iných vrstách. Je to znak stability systému.

Dohľad nad prenosom dát Všetka V/V komunikácia je sledovaná OS. Je to efektívny spôsob ako chrániť operačnú pamäť pred neautorizovaným prístupom. Prenosový protokol musí zachytiť všetky nekorektné vstupy a vylúčiť tak možnosť nedovoleného prenosu dát z pamäte do terminálu.

Kontrolný súčet pre dôležité údaje v pamäti Súborová štruktúra, a zvlášť súborový deskriptor by mal byť chránený kontrolným súčtom. Tým je pamäť chránená proti

neautorizovaným zmenám uložených dát. Chránené sú aj všetky dôležité hodnoty, vrátane samotného kódu operácií. Pred ich použitím je znova vypočítaný a otestovaný kontrolný súčet.

Maskovanie aktivít operačného systému Vykonávanie jednotlivých príkazov na čipe má vplyv na aktuálnu spotrebu. Sledovaním spotreby by teda útočník mohol napríklad odhaliť presný čas zápisu, alebo mazania EEPROM pamäte. Je veľmi doležité, aby si útočník nemohol podľa aktuálnej spotreby odvodiť takéto informácie. Úlohou operačného systému je čo najviac sťažiť útok výkonovou analýzou, napríklad vkladaním náhodných inštrukcií.

Znefunkčnenie čipovej karty Táto funkcia je dôležitá v záverečnej fáze životného cyklu karty. Pri vyradení karty z obehu zostáva čip stále plne funkčný. Zozbieraním takýchto kariet a použitím štatistických metód by bolo pomerne presne možné analyzovať ich softvér. Operačný systém preto musí obsahovať dávku inštrukcií, ktoré ho nenávratne znefunkčnia.

2.2.2 Prvky aplikacií

Základom efektívnej ochrany je vytváranie aplikácií, ktoré sú čo najmenej komplikované. Zjednodušuje to ich implementáciu ako aj verifikáciu. Čím komplikovanejší kód, tým väčšia šanca pre nájdenie a využitie chyby útočníkom. Informácie sú čerpané z [8].

Konzervatívne prístupové práva Prístupové práva pre súbory a príkazy sú pridelované čo najkonzervatívnejšie je to možné. Prístup je všeobecne odopretý, len v nevyhnutných prípadoch je povolený. Toto opatrenie znižuje šancu k vyneseniu dát z čipu, cez útočníkom zneužitú aplikáciu.

Obmedzenie vykonania príkazov Útočiť na čipovú kartu je oveľa zložitejšie, ak nie je možné spúšťať ľubovoľný príkaz, v ľubovoľnom čase a k tomu nespočetne veľakrát. Takéto opatrenie je možné dosiahnuť pomocou stavového automatu, kde jednotlivé stavy vymedzujú prípustné sekvencie príkazov, ktoré možno vykonať.

Zotavenie po chybe Zotavanie nastáva po nečakanom prerušení relácie. Operačný systém udržuje súbor, v ktorom sú zaznamenané posledné činnosti čipu. Vďaka tomuto logovaciemu súboru je možné sa pokúsiť o obnovu predchádzajúceho stavu čipu, alebo uviesť čip do konzistentého stavu. Pri ďalšej relácii teda nedochádza k nekonzistencii dát, ktoré by útočník mohol využiť.

Autentifikácia Ide o vzájomnú autentifikáciu karty a terminálu. Terminál overí pravosť karty, ako aj karta overí vhodnosť terminálu a jeho pripojenie do požadovaného systému. Čipová karta odmietne akékoľvek pokusy terminálu o komunikaciu, kým sa neautentifikuje. Toto opatrenie zabraňuje analýzam chovania operačného systému v súkromí.

Kapitola 3

Nasnímanie čipu

Táto kapitola popisuje postup získania obrazových dát čipu, pre uskutočnenie mikroskopickej analýzy. V úvode je podané pracovné prostredie, teda vybavenie laboratória. Nasleduje popis samotného snímania a v závere je ukážka získaných snímkov.

3.1 Vybavenia laboratória

Mikroskopické laboratórium je umiestnené v Ústave inteligentných systémov Fakulty informačných technológií VUT v Brne. Nachádza sa tu mikroskop Olympus BX61 s pomocným vybavením, obslužný počítač a stabilizačný stôl. Celú zostavu zobrazuje obrázok 3.1. Jednotlivé časti sú bližšie popísané v nasledujúcom texte.

Mikroskop Jedná sa o typ Olympus BX61. Je to optický motorizovaný mikroskop, navrhnutý pre snímanie biologických a materiálových vzoriek. Je koncipovaný ako modulárny systém, ktorý je možné upraviť, a neskôr aktualizovať podľa potrieb konkrétneho zákazníka. Snímať je možné manuálne, po jednotlivých snímkoch, alebo automaticky, podľa dopredu nastavených parametrov v obslužnom programe.

Riadiaci počítač Hadrvérovú konfiguráciu tvorí procesor Intel Core2 Duo, 2 GB operačná pamäť, pevný disk s kapacitou 500 GB a LCD monitor. Operačným systémom je Windows XP Professional. Inštalovaný program Stream Motion od firmy Olympus poskytuje komplexnú sadu nástrojov pre automatizované spracovanie obrazu, a taktiež umožňuje plnú kontrolu nad motorizovanými časťami mikroskopu.

Ovládacie jednotky V našom prípade sú k dispozícii dve takéto zariadenie. Prvým je panel s bielymi tlačidlami - typ U-HSTR2, ktorým sa ovláda osvetľovacia sústava a revolver s objektívmi. Druhým je čierny panel s joystickom, ktorý slúži pre manuálny posun motorizovaného stolčeka.

Riadiaca jednotka Externá riadiaca jednotka BX-UCB ovláda motorizované časti mikroskopu (napr. XYZ stolček, revolver s objektívmi atď). Pomocou nej program obslužného počítača, ako aj manuálne ovládacie jednotky, riadia mikroskop.

Stabilizačný stôl Je to špeciálny stôl, ktorého prostredná časť je schopná tlmiť aj najmenšie záchvevy vznikajúce v budove. Táto časť je zreteľne vyznačená, a aby nedochádzalo k zbytočnému vzniku vybrácií priamo od obsluhy, je na nej umiestnený len mikroskop. Pri použití veľkého zväčšenia by prípadné vybrácie úplne znemožnili pozorovanie pripravenej vzorky.

Záložný zdroj Súčasťou zostavy je samozrejme aj záložný zdroj napájania. V prípade výpadku elektrického prúdu umožňuje korektne vypnúť zariadenie, a predísť tak strate rozpracovaných úloh, či možnému poškodeniu zariadenia.

Obr. 3.1: Fotka zachytáva vybavenie laboratória. Okrem mikroskopu Olympus BX61 a riadiaceho počítača, môžeme vidieť aj ovládacie jednotky (U-HSTR2 s bielymi tlačidlami a čiernu s joystickom) a samotný stabilizačný stôl, ktorého čierna vyznačená plocha tlmí aj najmenšie vybrácie.

3.2 Vlastné snímanie čipu

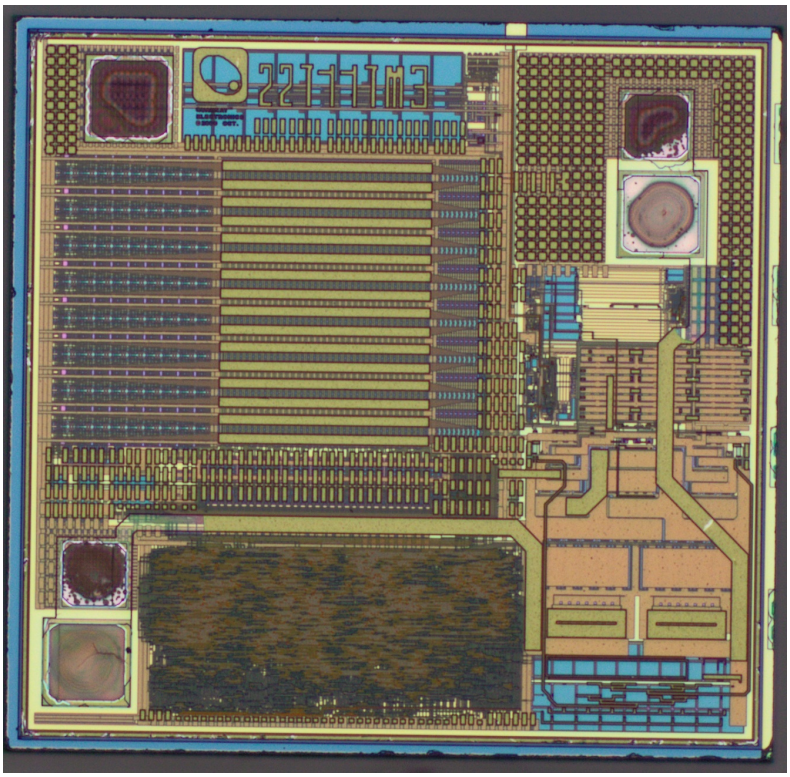
Získanie požadovaného snímku možno rozdeliť do niekoľkých fáz. Najdôležitejšou je príprava vzorky, najviac času zaberie samotné snímanie.

Príprava vzorky Veľkosť odpúzdrených čipov je vskutnu malá, niektoré vzorky boli menšie ako 1 mm^2. Čip sa môže ľahko „stratiť“ aj pri kýchnutí, maximálna opatrnosť je teda na mieste. Manipulovať s čipom je vhodné len v presvetlenej miestnosti prostredníctvom umelohmotnej, alebo ostrej kovovej pinzety. V opačnom prípade dôjde veľmi ľahko k jeho strate, alebo poškodeniu. Podložné sklíčko s čipom umiestnime na XYZ stolček pod objektívy. Snažíme sa ho umiestniť čo najpresnejšie do stredu zorného poľa. Presnú pozíciu nastavujeme pri zhruba 5 násobnom zväčšení. Nevykonávame ju ručne, ale pomocou ovládačov stolčeka. Ďalej nastavíme jas a v prípade potreby doostríme.

Snímanie Pri snímaní digitálneho predmetu volíme buď získavanie jednotlivých snímkov, alebo celej série. Zachytenie požadovaného predmetu len jedným snímkom je možné

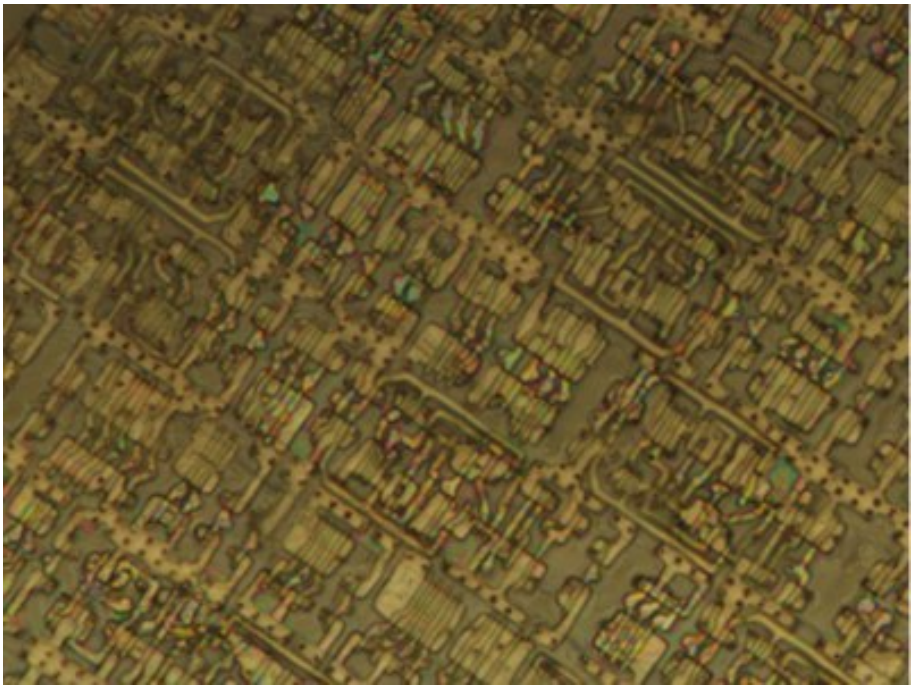
len pri malých čipoch a rozlíšeniach. Častokrát je ale nutné skladať výsledný obraz z množstva jednotlivých snímkov. Vtedy nastavíme dorazy pre pohyb XYZ stolčeka, a obslužný program sa už sám postará o nasnímanie vymedzenej oblasti. V prípade problémov s ostrosťou výsledného snímku je potrebné zvoliť použitie preostrenia z viacerých vrstiev. Nastavíme dolnú medzu, hornú medzu a krok. Program potom nasníma jeden obrázok z viacerých vrstiev, z každej vyberie najostrejšiu časť a spojí ich dohromady. Keďže ide vždy o sériu plnohodnotných snímkov, znásobuje sa čas potrebný pre vytvorenie požadovaného snímku.

Výsledok Dopracovanie sa k výsledku snímania je častokrát zdĺhavé a jeho úspech nie je zaručený. Potrebný čas závisí od zvolených parametrov. Hlavnými sú rozlíšenie snímku, počet preostrovacích vrstiev a počet snímkov v rovine x-y. Snímanie tak môže trvať rádovo hodiny, v extrémnych prípadoch až rádovo desiatky hodín. Výsledky softvéru Stream Motion sú však rozporuplné. Niektoré výsledné snímky sú v požadovanej kvalite, pri iných nastala chyba pri skladaní z jednotlivých snímkov a niekoľko hodinový proces bol tak zbytočný. Nasleduje ukážka výsledkov snímania (obrázky 3.2, 3.3 a 3.4).

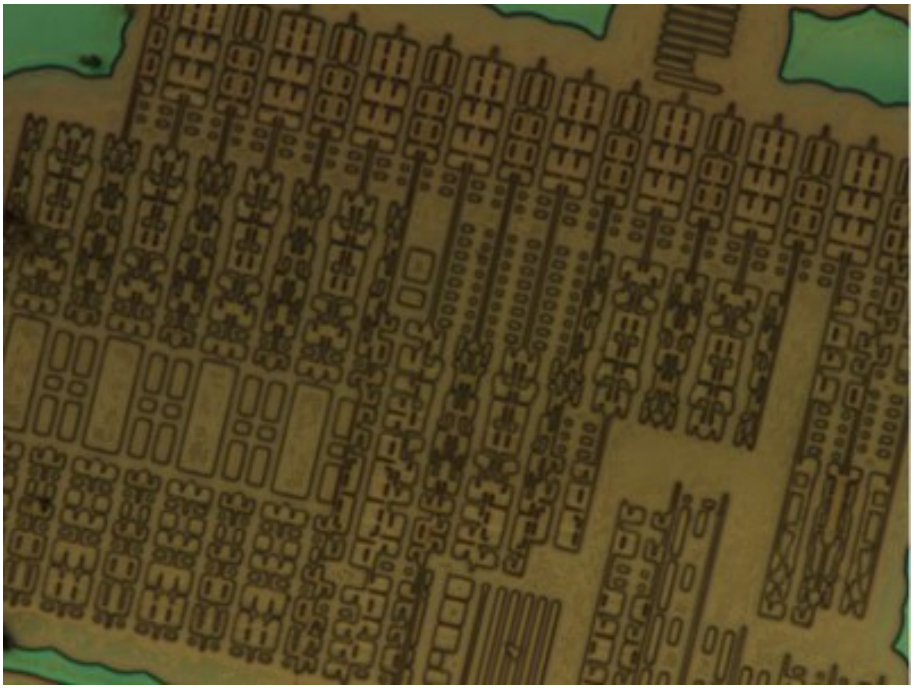


Obr. 3.2: Prehľadový snímok RFID čipu po odstránení pasivačnej vrstvy.

Ako je vidieť na obrázkoch 3.3 a 3.4, aj pri maximálnom sto násobnom zväčšení sa pohybujeme na samej hranici použiteľnosti snímkov pre mikroskopickú analýzu.



Obr. 3.3: RFID čip po odstránení vrchných vrsiev, stokrát zväčšený, auto fokus, svetlé pole.



Obr. 3.4: Obnažené tranzistory na MF(*angl. mifare*) čipe po odstránení vrchných vrstiev, storkát zväčšený, auto fokus, svetlé pole.

Kapitola 4

Návrh a implementácia knižnice

4.1 Ciele

Dôležitým momentom pri vytváraní aplikácie je stanovenie si cieľov, teda požadovaných funkcií, ktoré budú vo výsledku k dispozícii. Rovnako tomu bolo aj v prípade tejto knižnice. Základným cieľom bolo vytvorenie statickej knižnice pre anotáciu logických elementov a záujmových bodov, tak ako to umožňuje aplikácia projektu Degate [1]. Ciele boli určené nasledovne:

- Vytvorenie grafických entít, ktorými sa budú označovať väčšie oblasti čipu - elipsa, úsečka, obdĺžnik a mnohouholník.
- Vytvorenie grafických entít, ktorými sa budú označovať záujmové body - minimálne základné tvary ako trojuholník, štvorec, kruh. Okolo entít bude možné vykresliť ich obrys, pre ľahšiu identifikáciu v scéne.
- K entitám bude možné vpisovať textovú poznámku.
- Možnosť posunu entity po scéne, tzv. uchopovací mód.
- Možnosť zmeny tvaru entity ako celku, alebo pomocou jedného z jej významných bodov.
- Možnosť zmeny grafických vlastností entity po jej vytvorení.
- Možnosť určovať prekrívanie entít pri vykreslovaní.
- Vytvorenie statickej knižnice.
- Vytvorenie demonštračnej aplikácie pre predvedenie funkčnosti knižnice.

Návrh aplikácie vychádza z grafického frameworku Qt. V jednoduchosti by sa podstata fungovania každej grafickej aplikacie dala zhrnúť nasledovne: Programátor vytvorí niekoľko grafických entít (či už vlastných, alebo použije preddefinované), ktoré umiestni do grafickej scény. Tá sa stará o ich vykreslovanie kedykoľvek je to potrebné.

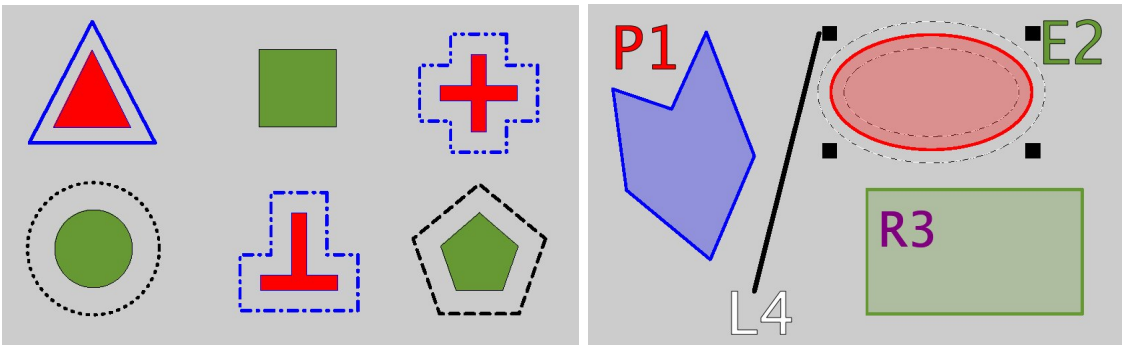
4.2 Dátová reprezentácia

Základnou triedou pre grafické entity v Qt je `QGraphicsItem`. Že ide o triedu Qt, je poznať podľa začiatočného písmena. Sú od nej odvodené preddefinované entity ako elipsa, mnohouhoľník, text atď. Taktiež poskytuje odľahčený podklad pre vytváranie vlastných grafických entít. Všetky geometrické informácie objektu sú vedené v lokálnom súradnicovom systéme. Iba metóda `pos()` vracia polohu entity v súradnicovom systéme rodiča. Pri vlastnej entite musíme vytvoriť podtriedu odvodenú z `QGraphicsItem` a minimálne implementovať jej dve čisto virtuálne metódy. Metóda `boundingRect()` vracia odhadovanú plochu, v ktorej je entita vykreslená. Metóda `paint()` vykresluje samotnú entitu. Často implementovanou metódu je `shape()`. Určuje plochu, v ktorej bude entita reagovať na kurzor myši. V prípade, že táto metóda nie je implementovaná, entita reaguje plochou danou `boundingRect()`. Ďalšie zaujímavé vlastnosti entít sa dajú nastaviť príznakmi. Napríklad `setFlags(QGraphicsItem::ItemIsMovable)` zabezpečí požadované presúvania entity po scéne pomocou myši. Programátor už nemusí nič iné doprogramovávať.

4.2.1 Entity

Entity sú rozdelené na dve skupiny. Prvou skupinou sú tzv. figúry, ktorými sa budú označovať väčšie celky v scéne. Každú entitu reprezentuje vlastná trieda, ktorá je odvodená od triedy `AbstractFigure`. Jedná sa o triedy `FigureElipse`, `FigureLine`, `FigurePolygon` a `FigureRectangle`. ity pre označenie záujmových bodov, tzv. bodové entity. Rodičovskou triedou je pre nich trieda `AbstractPoint`. Vtomto prípade sú odvodené triedy iba dve. Jednoduchý kruh reprezentuje trieda `PointCircle`. Mnohouhoľníkové tvary trieda `PointPolygon`. Entity sú zobrazené na obrázku 4.1.

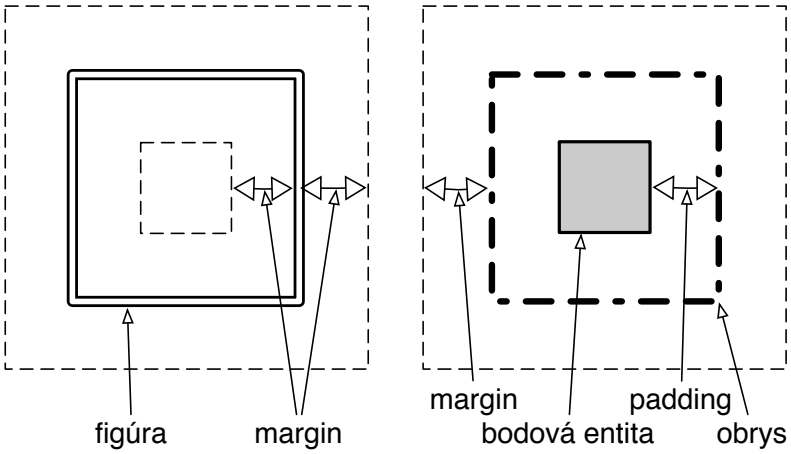
Obe abstraktné triedy sú odvodené od `AbstractItem`,ktorá je potomkom triedy `QGraphicsItem` Trieda deklaruje a definuje metódy, ktorými sa menia vlastnosti jednotlivých entít v scéne. Tieto metódy tvoria základ vonkajšieho rozhrania knižnice.



Obr. 4.1: Vľavo sú zobrazené entity pre anotáciu záujmových bodov. Každá z nich má iný štýl obrysového pera. Vpravo je ukážka figúr s textovým popisom. Pri elipse sú zobrazené aj záujmové body a hranice, v ktorých reaguje na kurzor myši.

Každá grafická figúra, ako aj záujmový bod, má okolo vykreslenej časti plochu, v ktorej je aktívny na udalosti vyvolané myšou (vybratie, uchopenie). Kurzorom myši teda nie je nutné nachádzať sa priamo nad entitou, čo by v prípade tenkých čiar spôsobovalo problémy. Pri vybratí entity sa na hraniciach plochy vykreslí čierno-biela čiara. Šírka plochy je u figúry dáná hodnotou vlastnosti *margin*. U tzv. bodovej entity je situácia komplikovanejšia. Je totiž

nutné vykresliť aj obrysy tvaru entity. Prítomná je preto ďalšia vlastnosť *padding*, ktorá udáva vzdialenosť v ktorej sa obrys vykreslí. V tejto oblasti entita reaguje na kurzor myši. Vlastnosť *margin* sa počíta až od obrysu. Názorná ukážka je na obrázku 4.2.



Obr. 4.2: Ukážka významu vlastností *padding* a *margin*. Vľavo je zobrazená figúra, vpravo bodová entita.

Špecifikácia jednotlivých entít:

Úsečka Úsečka je daná dvoma koncovými bodmi. Nachádzajú sa tu tzv. významné body. Ich uchopením a premiestňovaním sa mení dĺžka a smer úsečky.

Elipsa Definuje ju stredový bod a dĺžka hlavnej a vedľajšej poloosy. Významné body neležia priamo na nej, ale v rohoch opísaného obdĺžnika. Pri potiahnutí tohto bodu sa mení tvar elipsy, stred je však statický.

Obdĺžnik Je určený ľavým horným a pravým dolným bodom. Významné body ležia v strede každej strany. Pri potiahnutí sa posunie celá strana, ostatné ostávajú nehybné. Kedže ide o obdĺžnik, možný je pohyb len v smere osí X a Y.

Mnohouholník Významné body ležia priamo na bodoch, ktorými je mnohouholník určený, a pri ich potiahnutí sa mení len tento konkrétny bod.

Kruhový bod Ide o kruhovú bodovú entitu. Je daná vlastnosťou *size*, podľa ktorej sa určí stred vykreslenia. Bodové entity nemajú záujmové body.

Mnohouholníkový bod Ide o rôzne bodové entity, ktoré je možné vykresliť ako mnohouholník. Implemetované boli trojuholník, štvorec, päťuholník, obrátené T a kríž.

4.3 Vybrané riešenia vzniknutých problémov

Prekrývanie entít V grafických programoch je často nutné sa vysporiadať s problémom prekrývania zobrazovaných entít. Riešenie sa ponúka v zavedení vrstiev, kde entity

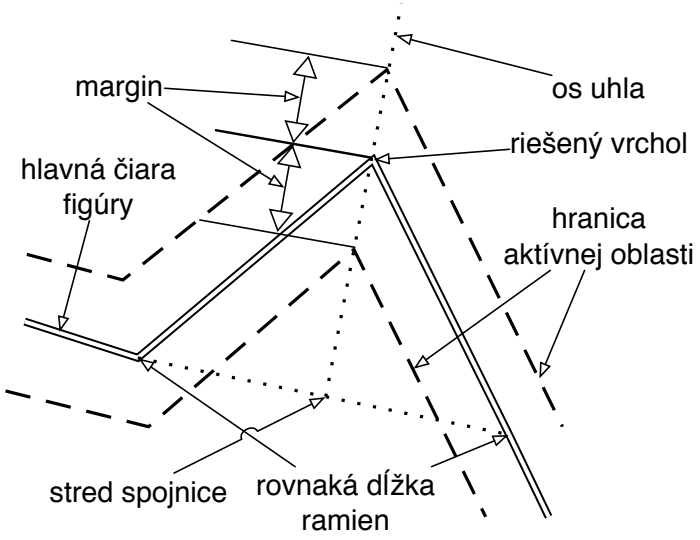
s vyššou vrstou sú vykreslované neskôr, teda prekrývajú tie s nižšou vrstou. Tento postup je samozrejme použitý aj v tejto knižnici. Pre prehľadnejšiu manipuláciu s grafickými entitami v scéne majú navyše figúry zmenenú oblasť, v ktorej reagujú na udalosti myši. Pri klasickej entite je táto oblasť daná plochou vymedzenou hlavnými čiarami. Figúry tejto knižnice však reagujú na udalosti myši len v okolí hlavných čiar, a nie aj v ich vnútornom priestore. Umožňuje to jednoduchšiu manipuláciu s bodovými entitami, ktoré sa tu budú zväčša nachádzať. Šírku tejto aktívnej oblasti určuje vlastnosť *margin* a samotný výpočet je osobitý problém popísaný v nasledujúcom texte.

Určenie hraníc aktívnej oblasti V pripade elipsy, alebo štvorca je výpočet jednoduchý, pretože je dopredu známy ich tvar. U elipsy sa len zmenší/zväčší polomer, u štvorca je známy počet strán a sú na seba kolmé, čo zjednodušuje výpočet. U všeobecného mnohouhoľníka je situácia komplikovanejšia. Vonkajšia a vnútorná hranica by mala byť len zväčšeným/zmenšeným tvarom entity. Problémom je určit práve polohu bodov definujúcich hranicu. Pri riešení sa vychádzalo z faktu, že každý hraničný bod vznike pri odpovedajúcom vrchole entity. Tento bod leží na osi uhla, ktorý zvierajú dve ramnená vrcholu mnohouhoľníka. Os, ako každá iná priamka, je určená dvoma bodmi. Prvým je samotný vrchol, druhý sa nachádza niekde na spojnici koncových bodov ramien. Ak sú ramená vrcholu rovnako dlhé, os uhla prechádza presne stredom tejto spojnice. Najjednoduchšie teda je upraviť dĺžku ramien. Pri konvexnom mnohouhoľníku potom bod vonkajšej hranice leží na polpriamke určenej bodom stredu spojnice a bodom vrcholu (smeruje z vnútra telesa von). Vnútorný bod hranice zase leží na opačnej polpriamke. Vzdialenosť hraničného bodu od vrcolu je daná vlastnosťou *margin*. Pri prejdení všetkých vrcholov mnohouhoľníka tak získame dva zoznamy bodov - pre vonkajšiu a vnútornú hranicu. Ilustrácia k tomuto postupu je znázornená na obr. 4.3. Problém vznikajúci pri nekonvexnom telese je ten, že síce určíme oba hraničné body prislúchajúce k danému vrcholu, ale nevieme určiť, do ktorého zoznamu bodov patria. Kvôli tomu je implementovaná metóda `isInside(QPointF point)`, ktorá určí, či bod leží v mnohouhoľníku, alebo nie.

Vnútorný bod telesa Ako už bolo spomenuté vyššie, pri určovaní hranic akívnej oblasti je využívaná metóda `isInside(QPointF point)`. Jej úlohou je zistiť, či dotazovaný bod leží vnútri telesa, alebo nie. Implementácia je realizovaná na základe algoritmu semienkového vyplňovania, známeho z počítačovej grafiky. Pri vyšetrovaní bodu sa prechádza scéna z ľavej strany (alebo z pravej, na tom nezáleží) a počíta sa, koľko hranic telesa sme pri tomto pohybe prešli, kým sme dorazili k vyšetrovanému bodu. Párny počet signalizuje, že bod leží mimo telesa, nepárny zase, že bod v telese leží.

Záujmové body figúry Pre pridanie figúry do scény je nutné ju nejakým spôsobom editovať. Pre zmenenie zobrazovaných vlastností sa použijú na to určené metódy. Zmena geometrických vlastností týmto spôsobom by bola nepraktická. Každá figúra má teda záujmové body. Pri ich ťahaní sa mení jej tvar. Pre záujmové body je daná figúra rodičovský prvok, a preto jej odovzdávajú všetky udalosti od myši. Nutné bolo teda jednotlivým bodom nainštalovať `sceneEventFilter()`. Tento filter rozhoduje, ktoré udalosti spracuje rodičovská entita a ktoré záujmový bod. Pri stlačení a ťahaní myši v záujmovom bode je táto udalosť spracovaná bodom, čím sa mení jeho pozícia, od ktorej sa mení tvar entity. Pri stlačení a ťahaní mimo záujmového bodu je táto udalosť spracovaná figúrou a mení sa celá jej poloha.

Textová poznámka Z vytýčených cieľov vyplýva, že každá entita musí mať možnosť vpi-



Obr. 4.3: Ilustrácia k výpočtu hraníc aktívnej oblasti entity.

sovania textovej poznámky. Do úvahy pripadali dve možnosti. Prvou je, že by textová entita bola samostatným objektom a nejakým spôsobom by sa zaistila jej komunikácia s grafickou entitou, ku ktorej patrí. Druhou je pokúsiť sa začleniť text priamo do entity, teda, že by text bol potomkom grafickej entity. Pri prvej možnosti vzniká problém so vzájomným pohybom entít. Pri premiestňovaní entity by sa text, kedže ide o samostatný objekt, nepremiestňoval. Zvolená bola preto druhá možnosť. Pri vzniku každej grafickej entity sa priamo v konštruktore vytvorí aj inštancia triedy `CustomTextItem`. Definované sú metódy pre prácu s týmto objektom, takže pre užívateľa je práca s textom rovnaká ako práca s hociktorou inou vlastnosťou grafickej entity.

Zmena tvaru kurzoru myši Manipulácia s entitami pri oddialení pohľadu na scénu sa ukázala byť pri testovaní problemtatická. Pri malých útvaroch, alebo tenkých čiarach je ťažko rozoznateľné, či sa kurzor nachádza práve nad ním, alebo nie. Pri snahe o trafenie pozície dochádza k zbytočnému klikaniu. Riešením sa ukázala zmena kurzoru myši. Poloha nad aktívnou oblasťou je signalizovaná kurzorom znázorňujúcim otvorenú ruku. Pri stlačení myši a pohybovaní entitou sa mení na zatvorenú ruku. Polohu nad významným bodom figúry oznamuje vystrčený ukazovák.

4.4 Rozhranie knižnice

Metódy rozhrania knižnice využívajú ako parametre a návratové hodnoty objekty z tried Qt. Pre lepšie pochopenie práce metód, budú preto najskôr popísané tieto triedy.

`QColor` Trieda stanovuje farbu založenú zväčša na modely RGB (červená, zelená modrá). Priesvitnosť sa riadi tzv. alfa zložkou.

`QFont` Z názvu triedy je zrejmé, že určuje typ písma. Typ je určený množstvom vlastností ako rodina písma, veľkosť, rez, váha atď.

`QPen` Pero je používané pri vykreslovaní základných (hraničných) čiar grafických entít. Medzi základné vlastnosti patrí farba, hrúbka a štýl vykreslovanej čiary. Hrúbka čiary je daná v pixeloch originálnej scény. Pri približovaní/oddaľovaní scény sa zväčšujú/zmenšujú aj jednotlivé pixely, čím sa mení hrúbka čiary. Špeciálnou hodnotou je nula, ktorá indikukuje tzv. kozmetické pero. Jeho čiara má vždy hrúbku 1 pixel rozlíšenia monitoru. Bez ohľadu na zväčšenie, alebo zmenšenie priblíženia scény je teda konštantná.

`QBrush` Štetec definuje výplň oblasti, ktorej hranice boli vykreslené perom. Základné vlastnosti sú farba a vzor.

`QString` Trieda reprezentujúca textový reťazec.

`qreal` V Qt ide o premennú typu double.

`QPointF` Trieda reprezentujúca všeobecný bod v dvoj-dimenzionálnej súradnicovej sieti. Koncové F v názve symbolizuje, že hodnoty súradníc reprezentuje desatinné číslo. [2]

4.4.1 Všeobecné vlastnosti

Tieto vlastnosti majú všetky vytvorené grafické entity. Za popisom vlastnosti sú uvedené súvisiace metódy. Predpona get označuje typ metódy, ktorá vracia danú vlastnosť. Naopak, predponou set sa táto vlastnosť nastavuje.

Text Vlastný text poznámky.

`QString getText()`, `void setText(QString text)`

TextFont Určuje typ písma textu.

`QFont getTextFont()`, `void setTextFont(QFont font)`

TextPen Týmto perom je vykreslovaný obrys písma.

`QPen getTextPen()`, `void setTextPen(QPen pen)`

TextBrush Určuje štetec, teda výplň písmen textu.

`QBrush getTextBrush()`, `void setTextBrush(QBrush brush)`

Pen U figúr je týmto perom vykreslovaná základná čiara a jeho farbou je vyplňovaná vnútorná plocha. Intenzita výplne je daná vlastnosťou *alpha*. V prípade entít pre anotáciu záujmových bodov, toto pero vykresluje obrys bodu. Vzdialenosť obrysu je daná vlastnosťou *padding*. Farbou z tohto pera je vykreslovaná aj hranica samotného bodu, jej hrúbka je nulová (tzv. kozmetické pero).

`QPen getPen()`, `void setPen(QPen)`

Margin Pri figúrach táto vlastnosť udáva šírku plochy okolo základnej čiary, v ktorej entita reaguje na udalosti myšou. Hranica tejto plochy je v prípade vybratia entity lemovaná čierno-bielou čiarou. U bodových entít táto plocha nereaguje na udalosti myši. Len obaľuje plochu danou vlastnosťou *padding*. Pre názornú ukážku viz obr. 4.2.

`int getMargin()`, `void setMargin(int)`.

Layer Určuje vrstvu vykreslovania. V prípade kolízie, entita s vyššou vrstvou, prekrýva tú s nižšou. Bodové entity by mali mať vyššiu vrstvu ako figúry.

`unsigned int getLayer()`, `void setLayer(unsigned int)`

Type Ide o číselný typ grafickej entity. Kedže všetky triedy odvodené zo spoločného predka majú rovnaký ukazovateľ, je dotaz na typ jednoduchou možnosťou, ako zistiť, s ktorým objektom práve pracujeme. Hodnota je pevne definovaná v triede reprezentujúcu danú entitu.

`int type()`

4.4.2 Špecifické vlastnosti a metódy

Bodové entity:

Padding Udáva vzdialenosť od hraníc vlastnej entity, v ktorej je vykreslený obrys pre lepšiu identifikáciu entity v scéne (viz obr. 4.2). Tento obrys zachováva tvar entity a jeho grafická podoba je určené vlastnosťou *PaddingPen*.

`qreal getPadding()`, `void setPadding(qreal)`]

PaddingPen Pero, ktorým je vykreslený obrys okolo entity.

`QPen getPaddingPen()`, `void setPaddingPen(QPen)`

Brush Udáva štetec, ktorým je vyfarbená entita.

`QBrush getBrush()`, `void setBrush(QBrush)`

Size Nastavuje veľkosť strany štvorca, v ktorom je bodová entita vykreslená.

`qreal getSize()`, `void setSize(qreal)`

Figúry:

BrushAlpha Určuje tzv. alfa zložku farby štetca u figúr. Táto vlastnosť určuje stupeň priesvitnosti pozadia. Môže nadobúdať hodnoty od 0 po 255. Čím vyššia hodnota, tým menej priesvitné pozadie.

`int getBrushAlpha()`, `void setBrushAlpha(int)`]

`addPoint()` Pridá záujmový bod do figúry typu mnohouhoľník.

`deletePoint()` Odoberie záujmový bod z figúry typu mnohouhoľník.

4.4.3 Vytvorenie entít

V tejto časti sú popísané konštruktory pre vytvorenie grafických entít. Hodnoty vlastností sú implicitne nastavené a nie je nutné ich zadávať v okamihu vytvárania entity.

`FigureElipse(QPointF center, qreal rx, qreal ry)` Stred figury je v bode `center`, `rx` a `ry` sú rozmery hlavnej a vedľajšej poloosy.

`FigureLine(QPointF lineStart, QPointF lineEnd, int margin)` Úsečka je definovaná začiatočným bodom, koncovým bodom a okrajom.

`FigurePolygon(QPolygonF polygon, int margin)` Pre vytvorenie grafickej entity typu mnohouhoľník, je najskôr potrebné určiť jeho vrcholy typom `QPolygonF`.

`QPointF topLeft, QPointF bottomRight, int margin` Figúra typu obdĺžnik je určená ľavým horným a pravým dolným rohom.

`PointCircle(qreal size, int margin, int padding)` Bodová figúra typu kruh je určená vlastnosťou *size*. Podľa nej sa určuje stred kruhu.

`PointPolygon(int shapeType, qreal size, int margin, int padding)` Týmto konštruktorom sa vytvárajú mnohouhoľníkové bodové entity. V konštruktore stačí určiť požadovaný tvar. K dispozícii sú trojuhoľník, štvorec, päťuhoľník, obrátené T a kríž. Tvary špecifikujú položky typu s vymenovaním hodnôt - `ShapeTriangle`, `ShapeSquare`, `ShapePentagon`, `ShapeUpsideDownT` a `ShapeCross`.

Okrem tried pre grafické entity bola vytvorená aj trieda `CustomGraphicsView`. Ide o reimplementovanie triedy `QGraphicsView`, ktorá poskytuje widget pre vykreslovanie obsahu grafickej scény. Doplnené boli signály indikujúce polohu kurzora myši v scéne.

Ako už bolo spomenuté, základnou triedou pre grafické objekty je `QGraphicsItem`. Táto trieda definuje množstvo metód a niektoré z nich boli použité pri implementácii tejto knižnice. Pre prácu s grafickými entitami plne postačujú vytvorené metódy, ktoré poskytujú všetku predpokladanú funkčnosť. Pre prípadné doplnenie funkčnosti sú ale ponechané aj všetky metódy materskej triedy.

4.5 Implementačné prostriedky

Pri výbere implementačných nástrojov a následne programovacieho jazyka bol braný ohľad na prenositeľnosť knižnice a predošlé skúsenosti. Knižnica bola vyvíjaná v nástroji Qt a ako programovací jazyk bol zvolený objektovo orientovaný jazyk C++.

4.5.1 Qt

Pre tvorbu knižnice bol zvolený nástroj Qt. Ide o multi-platformový aplikačný framework pre vývoj aplikácií s grafickým uživateľským rozhraním. Taktiež umožňuje tvorbu konzolových aplikácií, a čo je dôležitejšie, aj statických a dynamicky spojených knižníc. Qt disponuje výbornou podporou pre prácu s 2D a 3D grafikou. Umožňuje spravovanie a zobrazovanie veľkého počtu grafických objektov v dvoj-dimenzionálnej scéne. Pridávať je možné buď preddefinované objekty, alebo si vytvoriť vlastné. Qt je open-source projekt a pre nekomerčné využitie je šírený pod LGPLv2 licenciou. Je to vlastne knižnica pre programovací jazyk C++, ale v súčastnosti existuje aj pre množstvo iných jazykov. Veľkou výhodou je prehľadne spracovaná dokumentácia spolu s množstvom podrobne komentovaných zdrojových kódov dostupných priamo na webe [2]. Pre vývoj aplikácií je možné použiť vývojové prostredie Qt Creator. Pre rýchlu a intuitívnu tvorbu grafického rozhrania je vhodný Qt Designer. Aplikácie s grafickým rozhraním zachovávajú pri zobrazení natívny vzhľad operačného systému, takže do prostredia zapadnú.

Pri tvorbe knižnice ako aj demonštračnej aplikácie bolo použité vývojové prostredie Qt Creator 2.4.1 a SDK s knižnicami verzie 4.8.0.

Kapitola 5

Výsledky

Teoretická časť práce mala dva hlavné ciele. Prvým bolo oboznámiť čitateľa s technológiou čipových kariet, čím sa zaoberá kapitola 1. Druhým bolo zhodnotiť stav ochranných prvkov čipovej karty, čomu sa venuje kapitola 2.

V praktickej časti došlo k nasnímaniu obrazových dát z čipu, pomocou optického mikroskopu. Výsledné fotografie čipu sú však, aj pri využití maximálnych približovacích možností mikroskopu, na samej hranici použiteľnosti pre mikroskopickú analýzu. Snímaniu sa podrobne venuje kapitola 3.

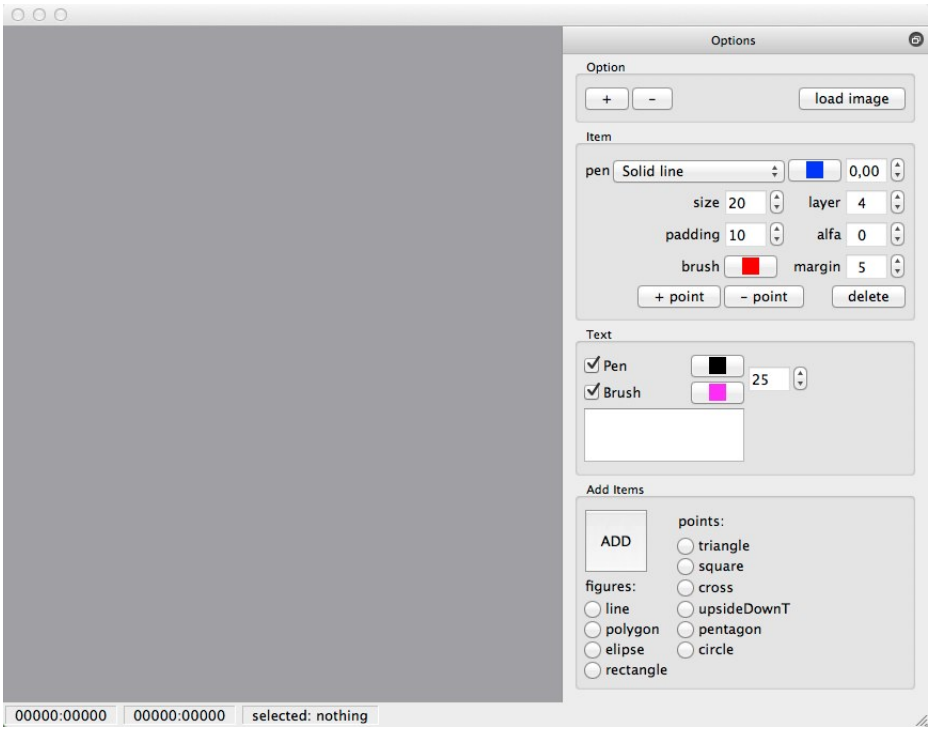
Hlavnou prioritou tejto práce však bolo vytvoriť statickú knižnicu pre anotáciu logických elementov. V tejto kapitole preto bude na demonštračnej aplikácii ukázané, aké funkcie knižnica poskytuje.

5.1 Demonštračná aplikácia

Aplikácia pre anotáciu logických elementov, využívajúca vyvinutú knižnicu, je implementovaná ako oknová aplikácia s grafickým uživateľským rozhraním. Aj keď ide len o demonštračný program, rozhranie bolo navrhnuté s ohľadom na jednoduchosť vytvárania nových a editáciu už vytvorených grafických entít. Všetky funkcie, poskytované knižnicou, sú ovládané z bočného panela. Panel možno presunúť, alebo od pracovnej plochy úplne oddeliť. Tlačidlá na ňom sú logicky zoskupené podľa funkcií, ktoré poskytujú. Jedná sa o skupiny pre všeobecné nastavenia, prácu s entitou, prácu s textom entity a pre pridanie vybratej entity. V dolnom stavovom riadku sa zobrazuje aktuálna poloha kurzoru v scéne, poloha posledného kliknutia a informácia, či je vybraná entita. Vzhľad aplikácie ukazuje obr. 5.1.

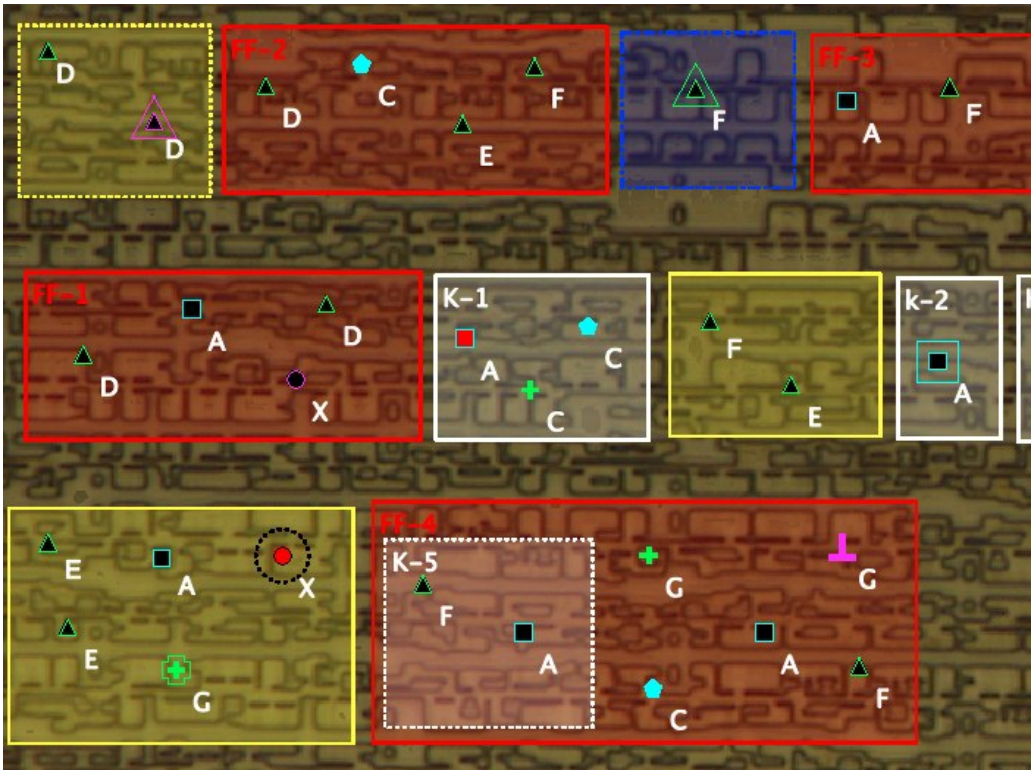
Pri vybraní entity sa na bočnom panely zobrazia jej aktuálne nastavenia, ktoré možno upravovať. Ak nie je vybraná žiadna entita, zobrazené sú implicitné hodnoty, ktoré sa použijú pri vytváraní novej entity. Tieto hodnoty možno samozrejme upravovať.

Na obrázku 5.3 je ukážka mikroskopickej analyzy čipu, pripravená demonštračnou aplikáciou. Na vyznačenie väčších oblastí čipu boli použité figúry typu obdĺžnik, s rôznym farebným prevedením. U niektorých je v ľavom hornom rohu vpísané aj konkrétne označenie. Záujmové body sú vyznačené bodovými entitami pri kombinovaní vlastností *pen* a *brush*. Ako je vidieť, farebých kombinácií je možné vytvoriť naozaj nespočetne veľa. Obrázok rovnako demonštruje prácu s vrstvami. Bodové entity majú nastavenú vyššiu úroveň vrstvy ako figúry, a preto sú vždy vykreslované na popredí. Podobnú anotáciu vidieť aj na obrázku 5.2. Ide o ukážku práce v aplikácii projektu Degate [1]. Využitím implementovanej knižnice je teda možné dosiahnúť rovnakých výsledkov ako v tomto projekte.

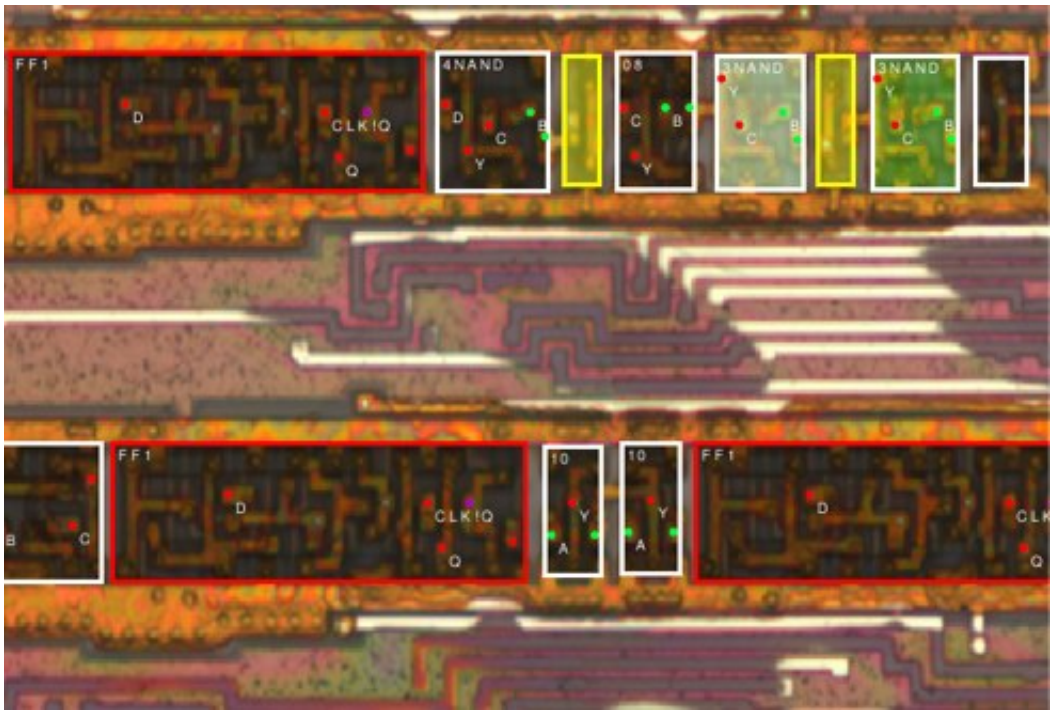


Obr. 5.1: Demonštračná aplikácia. Vpravo od pracovnej plochy sa nachádza ovládací panel prehľadne rozdelený do sekcií. V dolnej časti je stavový riadok, v ktorom sa zobrazujú súradnice kurzora myši v scéne ako aj informácia, či je vybratá nejaká položka alebo nie.

Demonštračne vytvorená aplikácia, využívajúca implementovanú knižnicu pre anotáciu logických elementov, ukazuje splnenie hlavného cieľa. Knižnica poskytuje základné grafické entity ako úsečka, elipsa, obdĺžnik a všeobecný mnohouhoľník pre vyznačenie väčších oblastí čipu. Pre označenie záujmových bodov poskytuje entity tvaru trojuhoľník, štvorec, mnohouhoľník, obrátené T a kríž (viz obr. 5.2). Zmenou ich vlastností možno vytvoriť nepreberné množstvo rôznych typov (viz obr. 4.1). Každému druhu logického prvku tak možno priradiť inú charakteristickú značku. Samozrejmosťou je vpisovanie poznámok. Vytvorená knižnica dosahuje po grafickej stránke minimálne rovnakú úroveň ako iné projekty s podobým zameraním.



Obr. 5.2: Ukážka vyznačenia logických obvodov v demonštračnej aplikácii.



Obr. 5.3: Ukážka vyznačenia logických obvodov v aplikácii projektu Degate. Obrázok prevzatý z [1].

Kapitola 6

Záver

Deklarované ciele tejto bakalárskej práce boli úspešne splnené. Práca oboznamuje čitateľa s technológiou čipových kariet. Podáva mu stručný prehľad vývoja kariet a ich možností využitia v rôznych sférach života spoločnosti. Spracovaná je anatómia čipu, a za prínos osobitne považujem vytvorenie prehľadného súhrnu ochranných prvkov čipu, s ktorými je nutné počítať pri vykonávaní rôznych druhov analýz.

V praktickej časti sa podarilo získať obrazové dáta čipu s pomocou mikroskopu. Hlavný cieľ, teda vytvorenie knižnice využiteľnej pri anotácii logických elementov čipu, bol rovnako dosiahnutý. Knižnica umožňuje vytváranie a manipuláciu s požadovanými grafickými entitami. Splňuje všetky na začiatku stanovené požiadavky a bude prínosom pri vytváraní aplikácií, zaoberajúcich sa mikroskopickou analýzou čipu.

Za osobný prínos práce považujem získanie vedomostí z diskutovanej problematiky, ako aj nových skúseností pri práci s nástrojom Qt.

Literatúra

[1] Project Degate. [cit. 2012-05-06], [online].
URL http://degate.org/

[2] Qt Online Reference Documentation. [cit. 2012-03-02], [online].
URL http://doc.qt.nokia.com/

[3] CardLogic: Smart Card & Security Basics. 2010, [cit. 2011-11-10], [online].
URL http://www.smartcardbasics.com/

[4] Chiedozie, A.: The History of Smart Cards. [cit. 2012-01-21], [online].
URL http://www.smartcardsupply.com/Content/Cards/7816standard.htm

[5] Haghiri, Y.; Tarantino, T.: *Smart Card Manufacturing : a practical guide*. John Wiley & Sons, 2001, ISBN 0-471-49767-3, 221 s.

[6] Hendry, M.: *Smart Card Security and Applications*. Artech House, druhé vydání, 2001, ISBN 978-1-58053-156-3, 328 s.

[7] MVSR: Slovensko začne tento rok vydávať občianske preukazy s elektronickým čipom. [cit. 2012-04-13], [online].
URL http://www.minv.sk/?tlacove-spravy&sprava=slovensko-zacne-tento-rok-vydavat-obcianske-preukazy-s-elektronickym-cipom

[8] Rankl, W.; Effing, W.: *Smart Card Handbook*. John Wiley & Sons, třetí vydání, 2003, ISBN 978-0-470-85668-8, 1088 s.

[9] Rosa, T.: Když se řekne SmartCards. *Chip*, ročník 7, č. 5, 1997, ISSN 1210-0684.

[10] SmartCardSupply: Smart Card Standards Overview. [cit. 2012-05-05], [online].
URL http://www.ehow.com/about_5468406_history-smart-cards.html

[11] Tech-FAQ: ISO 7816. [cit. 2012-01-27], [online].
URL http://www.tech-faq.com/iso-7816.html

Dodatok A

Obsah CD

\lib	statická knižnica
\bin	spustitelné súbory demonštračnej aplikácie
\doc	dokumentácia zdrojových kódov
\src-lib	zdrojové súbory knižnice
\src-app	zdrojové súbory demonštračnej aplikácie
\src-doc	zdrojové súbory technickej správy
\sup	inštalačný súbor vývojového prostredia
manual.pdf	stručný manuál k demonštračnej aplikácii
xmicha44.pdf	technická správa
readme.txt	nápoveda k obsahu CD