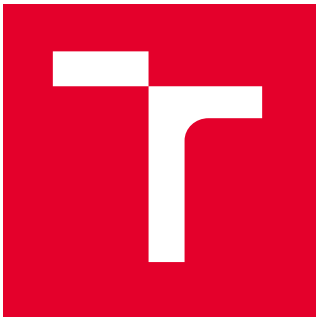


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

KRYPTOGRAFIE A BEZPEČNOST NA PLATFORMĚ MULTOS

CRYPTOGRAPHY AND SECURITY ON THE MULTOS PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Aleš Lidmila**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Dzurenda, Ph.D.**
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Aleš Lidmila ***ID:*** 221557

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Kryptografie a bezpečnost na platformě MultOS

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s platformou čipových karet MultOS se zaměřením na MultOS Trust Anchor technologii. Navrhněte a implementujte bezpečnostní IoT systém využívající tuto technologii. Analyzujte možnosti využití MultOS technologie jako TPM modulu pro IoT zařízení (např. Raspberry Pi, Arduino atp.). Analyzujte i možnosti vývoje vlastních kryptografických algoritmů založených na kryptografii eliptických křivek. Výstupem práce bude implementovaný zabezpečený IoT systém založený na technologii MultOS. Systém bude podporovat MultOS TPM modul a umožňovat komunikaci s okolím pomocí zabezpečeného spojení (s podporou autentizace, šifrování a integrity dat).

DOPORUČENÁ LITERATURA:

[1] MENEZES, Alfred, Paul C VAN OORSCHOT a Scott A VANSTONE. Handbook of applied cryptography. Boca Raton: CRC Press, c1997. Discrete mathematics and its applications. ISBN 0-8493-8523-7.

[2] TRUST ANCHOR TECHNICAL SUPPORT [online]. 2021 [cit. 2021-9-14]. Dostupné z: https://multos.com/support/multos-trust-anchor/

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 31.5.2022

Vedoucí práce: Ing. Petr Dzurenda, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Bakaláská práce se zabývá vyvinutím bezpečnostního IoT systému pro ověření uživatele s využitím platformy MultOS. Dále je zabývá vyvinutím a popisem vydavatelského systému, který spravuje koncové prvky MultOS.

KLÍČOVÁ SLOVA

MultOS, čipová karta, raspberry pi, Trust Core, IoT systém

ABSTRACT

The bachelor thesis deals with the development of a security IoT system for user authentication using the MultOS platform. It also deals with the development and description of a publishing system that manages MultOS endpoints.

KEYWORDS

MultOS, chip card, raspberry pi, Trust Core, IoT system

LIDMILA, Aleš. *Kryptografie a bezpečnost na platformě MultOS*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 59 s. Bakalářská práce. Vedoucí práce: Ing. Petr Dzurenda, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Aleš Lidmila
VUT ID autora:	221557
Typ práce:	Bakalářská práce
Akademický rok:	2021/2022
Téma závěrečné práce:	Kryptografie a bezpečnost na platformě MultOS

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Petru Dzurendouvi Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Internet věcí neboli IoT (Internet of Things) označuje miliardy fyzických zařízení po celém světě, která jsou nyní připojena k internetu a všechna shromažďují a sdílejí data. Díky příchodu levných počítačových čipů a přítomnosti bezdrátových sítí je možné proměnit cokoli, od něčeho malého jako mikročip až po něco velkého jako auto, na součást IoT. Propojením různých objektů a přidáním senzorů k zařízením vznikají různé Iot struktury, které každé součástce přidávají vlastní inteligenci a schopnost komunikovat s dalšími prvky systému v reálném čase bez nasazení člověka. Trh s IoT rychle roste. To znamená, že každý rok jsou vyvinuty miliardy zařízení s mnoha různými způsoby použití. IoT pomáhá zvýšit efektivitu procesů, ulehčit pracovní zátěž a rychle sbírat data. [1]

Iot je v posledních letech jedním z nejrychleji se rozvíjejících technologických trendů. S růstem Iot přichází problém s bezpečností, kde je obtížné celý systém zabezpečit tak, aby nebyl zranitelný. Zajištění bezpečnosti systémů IoT je složité, lze na něj pohlížet z mnoha hledisek. Běžným problémem může být omezení zdrojů. Mnoho chytrých zařízení je omezeno zdroji a mají nízký výpočetní výkon. Nemohou tedy spouštět výkonné bezpečnostní operace. Dalším problémem je přístup k jednotlivým zařízením přes internet, neaktualizování systému, slabé přihlašovací údaje apd. Problémů je mnoho a je nutné je řešit. Jedním z řešení jsou zabezpečené TMP moduly tvořící jádro celého systému.

Cílem této bakalářské práce je otestovat a navrhnout bezpečnostní IoT systém se zaměřením na TMP modul. Jako bezpečnostní prvek byla zvolena platforma MultOS, kvůli její obsáhlé specializaci na zabezpečení čipových karet a IoT systémů.

1 MultOS technologie

MultOS je systém pro čipové karty, který se používá pro zajištění bezpečnosti. Společnost MultOS je autorem multiaplikačního systému. Možnost ukládat více aplikací na kartě přinesla více funkcí spolupráce na vývoji a jednoduchosti v podání jednoho zařízení. Aplikace na kartě jsou od sebe odděleny a fungují nezávisle na sobě. Tímto způsobem lze kombinovat programy od různých výrobců a společně je nahrát na jeden mikročip MultOS [2]. [6].

1.1 Architektura

Skládá se ze dvou základních architektur. První architektura je navržena pro zabezpečení. Na čipu se nachází virtuální stroj, který spustí aplikaci a bezpečnostní schéma MultOS, přičemž bezpečnostní schéma chrání samotný chip, zkompilovaný kód a data aplikace před poškozením. Kódy jsou nejčastěji psané v jazyce C nebo Java. Ty jsou dále převáděny do bytecodů MultOS Executable Language (MEL), které jsou spouštěny virtuálním strojem. Jakákoliv neplatná reference k přístupu do paměti je odchycena virtuálním strojem a veškerý prováděný chod aplikace se zastaví. Kontrola času provádění instrukcí rovněž přináší úroveň zabezpečení, proto byla zhotovana jako druhá architektura. Aplikace nemůže přistupovat k datům jiné aplikace ani k přidělené paměti. Všechny implementace OS MultOS zahrnují snadardní sadu primitivních funkcí. Pro zaručení zpětné kompatibility mezi produkty je možné primitivní funkce volit [5].

1.2 Struktura paměti

Konzistence paměti je složena pomocí pamětí RAM (Random Access Memory), ROM (Read-Only-Memory) a EEPROM (Electrically Erasable Programmable ROM). RAM je přepisovatelnou pamětí využívající elektrický proud pro svůj chod a používá se na vykonání jednoduchých operací nutných pro dokončení programu jako celku. Data uchovávaná v paměti RAM jsou po zaniknutí elektrického proudu ztracena. Pamět EEPROM taktéž využívá pro zápis a aktualizaci dat elektrický proud. Její výhodou je, že po zániku elektrického proudu nedojde ke ztrátě informací. Poslední pamět ROM je určena pouze pro čtení a nelze ji měnit. Nachází se zde statická data a operační systém.

1.2.1 Veřejná paměť

Veřejná paměť slouží jako uložiště pro APDU zprávy a APDU odpovědi. APDU zprávy jsou realizovány čtením z globální proměnné, APDU odpovědi zápisem. Jedná se o paměť typu RAM a označuje se jako #pragma melpublic.

1.2.2 Statická paměť

Označována jako #pragma melstatic. Je přepisovatelnou pamětí, nezávislou na elektrickém proudu. Využívá se pro zapisání privátních dat do aplikace. Jsou jimi například privátní klíče nebo osobní informace. Využívá se paměť typu EEPROM.

1.2.3 Dynamická paměť

Používá se pro dynamické zpracování při běhu aplikace. Může se jednat o sérii výpočtů, přepisu proměnných. Je rozdělena na dvě části: Session Data (#pragma melsession) a zásobník (Stack). Velikost Session Dat je fixně stanovena při inicializaci proměnných. Velikost zásobníku se může měnit za běhu aplikace. Zpracovávaná data v dynamické paměti jsou po ukončený aplikace zapomenuta.

1.3 Bezpečnost systému

Bezpečnost je jedna z hlavních priorit, které je nutné řešit. Lze na ní nahlížet z pohledu zvolené kryptografie, kde řešíme přenos citlivých dat nebo z pohledu lokálního zabezpečení systému. MULTOS je více aplikační systém, tudíž je nutné zajistit, aby se aplikace vzájemně neomezovaly a nepřistupovaly do nepřidělené paměti.

1.3.1 Alokace paměti

Řešení lokálního zabezpečení aplikace je podstatné pro správný chod celého systému. Mezi nejčastější problémy patří alokace paměti, která je řešena maximální fixní velikostí aplikace. Maximální fixní velikost aplikace je stanovena po vytvoření aplikace a zařízení si ji přidělí při nahrávání. Nemůže se stát, že aplikace vezme paměťový prostor jiné aplikaci. Velikost potřebné paměti pro alokaci lze zjistit příkazem hls -t <program>.hzx.

1.3.2 Bezpečné nahrávání a mazání aplikací

Tento úkon je na platformě MultOS řešen certifikáty, které zaručují bezpečný přenos aplikací na zařízení nebo jejich vymazání. Při vytvoření aplikace vznikne soubor s příponou ALU (Application Load Unit) nesoucí zdrojový kód. Ze souboru ALU

je pak možné vytvořit certifikát pro bezpečné nahrání. Tento certifikát se nazývá ALC (Application Load Certificate) a nese AID (Application ID) aplikace. ALU je při pokusu o nahrání porovnán s certifikátem ALC. Pokud ALC je odvozeno od ALU, aplikace se přenese na zařízení. Podobným způsobem lze z ALU vygenerovat soubor pro mazání ADC (Application Delete Certificate) Postup je totožný s ALC certifikátorem s drobným rozdílem, že aplikace je poté ze zařízení odstraněna.

1.4 Komunikační protokol APDU

APDU protokol je aplikačním protokolem mezi zařízením MultOS a aplikací. Komunikační kanál zprostředkovaný protokolem APDU je pouze jednosměrný. První zařízení musí čekat po určitou dobu na vykonání procesu druhé strany, načež první zařízení odpovídá. Máme dva typy APDU zpráv. V případě odeslání komunikačním prvkem se nazývá APDU response. Odeslání zprávy APDU ze strany aplikace se označuje APDU command, který je tvořen hlavičkou a tělem [7].

Obsah hlavičky APDU command

- CLA: zvolení třídy instrukcí.
- INS: výběr instrukčního kódu.
- P1, P2: parametr 1,2 pro upřesnění instrukcí.

Obsah těla APDU command

- Lc: velikost přenášených dat v bitech.
- Data: data, která mají být přenesena.
- Le: velikost očekávaných dat v bitech.

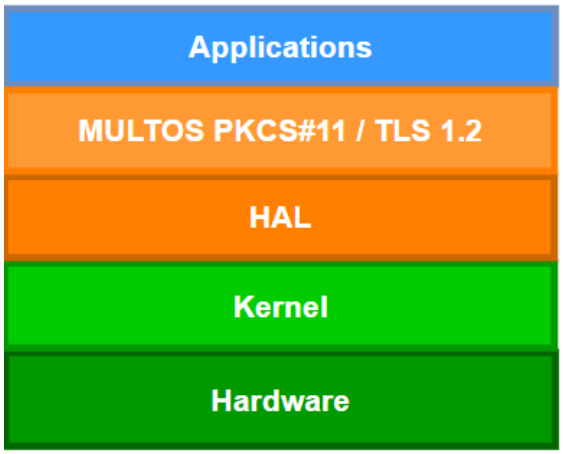
Obsah APDU response

- Datové pole: velikost odesílaných dat určených položkou APDU commandu Le.
- SW1: vrací typ chyby/úspěšné provedení.
- SW2: v případě neúspěchu chybu upřesní, pokud je operace úspěsná vrací 00.

1.5 Trust Core

MultOS Trust Core je vestavěný mikrokontrolér poskytující hardwarový bezpečnostní modul pro chytré a připojené zařízení spadající do rodiny Trust Anchor.

Trust Core je navržen pro Raspberry Pi™ Model 3B (včetně 3B+) a desky Arduino™ kompatibilní s rozložením záhlaví UNO R3. Nabízí hardwarovou root of trust ochranu (zdroj, které lze v rámci kryptografického systému vždy důvěřovat), kritickou pro mnoho IoT řešení. Komunikace s hardwarem Trust Core je řízena vrstvou HAL (Hardware Abstraction Layer). Vrstva HAL zajišťuje konzistentní rozhraní pro komunikaci s různorodými hostitelskými platformami zkompletovanými v různých programovacích jazycích. Na Raspberry Pi se používá podmnožina standardního PKCS 11 v2.40 API programovatelná v jazyku C/C++ nebo speciálně vyvinutého MULTOS TLS 1.2 API rozšířenou o programování v jazyce python. Hierarchie systému je znázorněna na Obr. 1.1: Hierarchie systému [8].



Obr. 1.1: Hierarchie systému

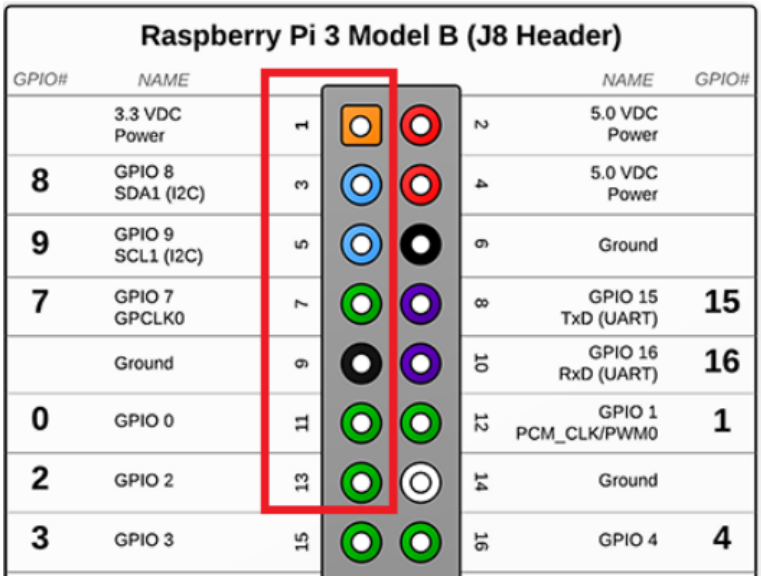
1.5.1 Využití

Trust Core byl navržen tak, aby podporoval AWS IoT Greengrass 1.x běžící na Raspbian. Greengrass je cloudová služba internetu věcí (IoT) pomáhající vytvářet, nasazovat a spravovat aplikace IoT na vašich zařízeních. Dále podporuje SAS (A shared access signature) token pro nakládání s osobními daty a ověřování X.509. Certifikát X.509 je digitální certifikát založený na široce uznávaném standardu X.509. Používají se ke správě identity a zabezpečení v internetové komunikaci a počítačových sítích. Setkáváme se s nimi každý den při používání webových stránek, mobilních aplikací, online dokumentů a připojených zařízení.

Operační systém MultOS Trust Core umožňuje využít komponentu jako podpůrného prvku v Iot systému ve formě co-processoru nebo jako hlavního zařízení zvané mikrokontrolér. Klíčové vlastnosti umožňují ochranu zařízení za běhu, zajištění identity koncových bodů a zabezpečení kritických dat.

1.5.2 Propojení Trust Core s Raspberry pi

Pro propojení Trust Core s Raspberry pi se využívá 7 GPIO pinů seřazených za sebou v druhém řádku na hlavní desce viz. Obr. 1.2: GPIO Trust Core. Mezi nejdůležitější GPIO piny patří pin pro napájení, vstup komunikace, výstup komunikace a uzemnění.



Obr. 1.2: GPIO Trust Core

1.5.3 Kryptografická podpora

- Generování klíčů RSA a jejich ukládání
- Generování podpisu pomocí RSA a ukládání
- Šifrování/dešifrování RSA
- Generování ECC klíčů a jejich ukládání
- Výměna klíčů ECC Diffie Hellman pro TLS
- Generování ECC DSA digitálního podpisu a ověření
- Autentizaci TLS / DTLS 1.2, výměnu klíčů a šifrování / dešifrování zprávy
- Generování klíčů AES
- Šifrování a dešifrování AES v CBC a GCM modu
- Generování náhodoných čísel na základě hardwaru
- SHA-1, SHA-256, SHA-384 a SHA-512 hash
- SHA-1 a SHA-256 založený na HMAC
- Management klíčů
- Management PINu

1.5.4 Čipové karty

Čipové karty jsou zhotoveny nejčastěji z plastové slitiny o velikosti platební karty. Obsahují vložený mikročip s integrovaným obvodem. Čip provádí veškeré operace. Mezi základní operace patří ověření oprávněné osoby, např. na základě znalostí, nebo samotná práce s daty. Čipové karty dále poskytují bezpečnou a spolehlivou službu stavěnou na základě kryptografických operací [3].

V tomto odvětví jsou zajímavé programovatelné mikroprocesorové karty s podporou kryptografie. Ty mají navíc vestavěný kryptografický procesor, tzv. co-processor umožňující provádět známé kryptografické standardy, např. AES (Advanced Encryption Standard) nebo RSA (Rivest, Shamir, Adleman). Navíc disponují vlastním operačním systémem, který je potřebný pro tvorbu programu. Mezi nejznámější programovatelné mikroprocesorové karty patří JavaCard, MultOS, BasicCard a .NET-Card.

Pro práci s čipovými kartami od firmy MultOS se používá bezplatný software SmartDeck přímo od výrobce. Software SmartDeck byl vyvinut pro nahrávání aplikací na kartu, vytváření hzx, alu, alc souborů nebo jako jednoduchý debugger. Pro zobrazení velikosti pamětí, id výrobce, úrovně zabezpečení a dalších informací dané čipové karty tento software disponuje příkazem *hterm*. Výpis příkazu hterm se provádí do příkazového řádku na Obr. 1.3: hterm pro čipovou kartu MC4.

```
C:\Users\xlidmi01>hterm -multos
MULTOS data
rom_ic_details 0x84 0x40
ic_manufacturer_id 0x02
multos_implementor_id 0x02
mcd_id.Mism_id 0xff 0xff
mcd_id.icc_serial_number 0xff 0xff 0xff 0xff
msm_mcd_permissions.mcd_issuer_product_id 0x00
msm_mcd_permissions.mcd_issuer_id 0x00 0x00 0x00 0x00
msm_mcd_permissions.set_msm_controls_data_date 0x00
msm_mcd_permissions.mcd_number 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
msm_mcd_permissions.RFU5 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01
msm_mcd_permissions.RFU2 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
msm_mcd_permissions.RFU6 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01
msm_mcd_permissions.RFU4 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
max_dynamic_size 0x1108
max_public_size 0x212
max_dir_file_record_size 0xff
max_fci_record_size 0xff
max_atr_hb_record_size 0xf
max_atr_file_record_size 0x20
multos_pk_certificate_length 0xc8
security_level 0x5a
certification_method_id 0xf2 0x00
app_signature_method_id 0x01 0x01
encipherment_descriptor 0x00 0x01
hash_method_id 0xf2 0x00
```

Obr. 1.3: hterm pro čipovou kartu MC4

2 Systémy řízení přístupu

Řízení přístupu je bezpečnostní technika regulující, kdo nebo co může přistupovat nebo používat zdroje ve výpočetním prostředí. Jedná se o základní koncept v oblasti bezpečnosti minimalizující riziko ohrožení či ztrátu aktiv. Existují dva typy řízení přístupu: fyzické a logické. Řízení fyzického přístupu omezuje přístup do budov, areálů nebo místností. Logická kontrola přístupu omezuje přístup k počítačovým sítím, systémovým souborům a datům. Zabývají se jimi systémy související s financemi, soukromím, bezpečností nebo obranou [9].

V některých systémech je úplný přístup udělen po úspěšné autentizaci uživatele, ale většina systémů vyžaduje sofistikovanější a komplexnější kontrolu. Kromě mechanismu autentizace (jako je heslo) se řízení přístupu zabývá tím, jak jsou strukturovány autorizace.

2.1 Autentizace

Autentizace je termín označující proces dokazování, že nějaká skutečnost nebo nějaký dokument je pravý. V informatice je tento termín obvykle spojován s prokazováním identity uživatele. Prokazování identity se provádí nejčastěji znalostmi, vlastnictvím nebo vlastnostmi uživatele [10] [11].

2.1.1 Autentizace znalostmi

Autentizace znalostmi je ověřovacím faktorem, kde uživatel dokazuje, že zná patřičné informace. Obvykle se jedná o heslo nebo osobní identifikační číslo (PIN), které sdílí uživatel a systém správy přístupu. K použití tohoto faktoru systém vyžaduje, aby uživatel poskytl sdílené informace.

2.1.2 Autentizace vlastnictvím

V tomto případě musí uživatel prokázat, že něco vlastní. Například chytrý telefon nebo čipovou kartu. Systém vytvoří výzvu pro uživatele, aby se ujistil, že vlastní požadovaný autentizační faktor. Například odesláním časově založeného jednorázového hesla přes SMS zprávu na zařízení vlastněného uživatelem. V případě čipové karty zkontroluje doloženou čipovou kartu. Autentizace vlastnictvím je úspěšná, je-li doloženo správné autentifikační zařízení.

2.1.3 Autentizace vlastnostmi

Autentizační faktor je založen na vlastnosti náležící konkrétnímu uživateli a je tomuto uživateli vlastní (faktor inherence). Tyto vlastnosti jsou obvykle biometrické charakteristiky – otisky prstů, rozpoznání obličeje nebo rozpoznání hlasu. Doložené informace jsou porovnány s kopiemi uloženými v databázi a jsou hledány shody. Jsou-li nalezeny dostačující shody, autentizace proběhla úspěšně.

2.2 Autorizace

Autorizace je bezpečnostní mechanismus navazující na provedení autentizace, který určuje úrovně přístupu nebo uživatelské oprávnění související se systémovými prostředky. Může se jednat o přístup k souborům, službám, počítačovým programům nebo funkcím aplikace. Jedná se o proces udělení nebo zamítnutí přístupu k síťovým prvkům, umožňující uživateli přístup k různým zdrojům na základě identity uživatele [12].

2.3 Integrita dat

Integrita dat je forma zajištění, že data nebudou poškozena a mohou k nim přistupovat nebo je upravovat výhradně osoby, které jsou k tomu oprávněny. Integrita zahrnuje udržování konzistence, přesnosti a důvěryhodnosti dat během celé existence. K zachování integrity dat nesmí být data měněna během přenosu a musí být podniknuty kroky k zajištění opatření, aby data nemohla být změněna neoprávněnou osobou nebo programem. V kryptografii se pro zajištění integrity používají hashovací funkce. Hashovací funkce jsou založeny na jednosměrné funkci, kde na vstupu jsou data o libovolné délce a na výstupu data pevně stanovené délky. Příkladem hashovacích funkcí jsou SHA-1, SHA-2 nebo SHA-3 [13].

2.4 TPM modul v IoT

TPM definuje hardwarové jádro v rámci bezpečnostního IoT systému a poskytuje např. integritu, autentizační služby nebo minimalizaci ztráty v případě napadení útočníkem. TPM se využívá ve spojení s dalšími bezpečnostními technologiemi jako jsou firewally, antivirové softwary, čipové karty a biometrická ověřování. Implementace TPM pro IoT je jiná než TPM pro PC, Iot TPM se vyznačuje zvýšenou rychlostí a jednoduchostí [14].

3 Kryptografické protokoly

3.1 Kryptografie

Kryptografie je vědeckým oborem zabývající se bezpečným a šifrovaným přenosem dat mezi dvěma nebo více stanicemi nebo jejich uchováváním. Jedná se o koncept, jehož cílem je udržet důležité informace, které mohou být předmětem potenciálního odcizení. Kryptografie umožňuje, aby data byla pseudonymní a bezpečná, a zajišťuje integritu bez nutnosti účasti třetích stran. Kryptografii dále rozdělujeme na symetrickou a asymetrickou [4].

3.1.1 Symetrická kryptografie

Je metodou šifrování/dešifrování využívající jeden klíč nazývajícím se tajným klíčem. Během tohoto procesu jsou data převedena do formátu, který nemůže přečíst ani zkontrolovat nikdo, kdo nemá tajný klíč, který byl použit k jejich zašifrování. Symetrická kryptografie se používá pro svou jednoduchost a rychlost. Symetrická kryptografie je dále dělena na proudovou a blokovou [15].

Proudové šifry

Proudové šifry se vyznačují tím, že šifrují zadaný řetězec bit po bitu nebo bajt po bajtu. Jako klíč se využívá pseudonáhodný generátor čísel. Klíč musí být jedinečný, aby zajistil bezpečnost zašifrovaného textu. Zašifrovaný řetězec je výsledkem operátoru XOR, kde jsou vstupními parametry klíč a nezašifrovaný text. Proudové šifry jsou rychlejší než blokové a využívají se pro malý obnos dat. Nejznámějšími proudovými šiframi jsou např. Salsa20, RC4 nebo A5.

Blokové šifry

Bloková šifra je způsob šifrování dat po blocích za účelem vytvoření šifrovaného textu pomocí kryptografického klíče a algoritmu. Data jsou zpracována po blocích fixně stanovené velikosti. Narozdíl od proudové šifry jsou bloky zpracovávány současně. Moderní šifry jsou navrženy tak, aby délka bloků byla 128, 192 nebo 256 bitů. Blokové šifry pro zabezpečení informací, jako je důvěrnost nebo autenticita, používají tzv. módy. Mezi používané módy paří ECB (Electronic Code Block), CBC (Cipher Block Chaining), CFB (Cipher Feedback), OFB (Output Feedback) a CTR (Counter). ECB je nejjednodušší, a proto se aplikuje jen zřídka. Využívá oddělené šifrování bloků o délce 64B, díky čemuž lze operace snadno prokouknout. Ostatní módy se

používají dle situace. Ke svému algoritmu využívají operaci XOR a IV (Initialization Vector) pro své znáhodnění. Zástupci blokových šifer jsou DES, 3DES nebo AES.

3.1.2 Asymetrická kryptografie

Je metodou šifrování/dešifrování využívající dva klíče. Klíče jsou děleny na soukromé a veřejné a jsou od sebe matematicky odvozeny. Soukromé a věřejné klíče se nazývají klíčovým párem a jsou schopny vzájemně šifrovat a dešifrovat stanovená data. Veřejný klíč je zpřístupněn komukoli, soukromý klíč je uchováván v tajnosti. Data se šifrují veřejným klíčem, výsledný zašifrovaný řetězec je schopný dešifrovat pouze majitel odpovídajího soukromého klíče. Šifrováním dat pomocí soukromého klíče se vytvoří digitální podpis. Tím je zajištěno, že zpráva přišla od uvedeného odesílatele. Při ztrátě nebo odhalení soukromého klíče je nutné vygenerovat nový pár.

Asymetrická kryptografie se nejčastěji používá k přenosu tajného klíče, k nasazení symetrické kryptografie. Pro přenos tajného klíče je nutné, aby jedna strana vygenerovala tajný klíč a zašifrovala ho pomocí veřejného klíče příjemce. Příjemce ho po přijetí dešifruje svým soukromým klíčem. Nadcházející komunikace poté probíhá prostřednictvím tajného klíče relace, který se stane šifrovacím klíčem. Mezi nejčastější zástupce patří RSA, DSA a Diffie-Hellman [16].

3.2 Kryptografické protokoly pro důvěrnost, autenticitu a soukromí na omezených zařízeních

Než dojde ke stanovení požadavků pro protokoly a algoritmy, je nutné provést zhodnocení a konkretizaci požadavků pro kryptografický systém. Kryptografické protokoly musí vyvinout způsob, jak splnit stanovené požadavky, které jim byly předem určeny. Následující popis požadavků odkazuje na Cryptographic Protocols for Confidentiality, Authenticity and Privacy on Constrained Devices [17].Mezi stanovené požadavky patří:

- **Integrita** - schéma podporuje ustanovení a vytvoření kanálu s ochranou integrity dat.
- **Utajení** - je realizováno vytvořením šifrovacího kanálu mezi zařízeními.
- **Diverzifikace klíče** - kryptografická pověření jsou pro každé zařízení unikátní, a proto získání jednoho klíče nezpůsobí získání druhého nebo dalších klíčů.
- **Autentičnost** - schéma umožní vytvoření šifrovaného a autentizovaného kanálu pouze za předpokladu, že dojde k oboustranné autentizaci.

- **Výpočetní složitost** - je využita pouze nízká výpočetní složitost, prvky symetrické kryptografie a základní operace pro optimalizaci výkonu.
- **Odolnost** - je zajištěno, aby data, která byla potencionálně odposlechnuta v předchozích relacích, nebylo možno využít pro napadení opakováním relace.

3.2.1 Entity a struktura protokolu

Protokol se skládá ze tří entit:

- **Vydavatel** - je důvěryhodnou osobou a působí jako správce klíčů, které vydává ověřovatelům a důvěryhodným uživatelům.
- **Uživatel** - vlastní unikátní identifikátor ID_u a disponuje speciálním klíčem $K_{v\text{-}u}$ pro každeho ověřovatele.
- **Ověřovatel** - přijímá žádosti uživatelů o ověření a na základě událostí je schvaluje nebo zamítá. Každý ověřovatel disponuje vlastním unikátním klíčem K_v, ten je párovatelný s klíčem uživatele, který je shledán legitimním.

Interakce entit probíhá za pomoci následujících protokolů:

- **Setup** - vygenerování systémového klíče K.
- **Issue** - derivace systémového klíče K a předání derivátů v podobě klíčů uživatele. $K_{v\text{-}u}$ a ověřovatele K_v.
- **Show <-> Verify** - ověřující fáze sloužící k ověřeření znalosti klíče uživatele a $K_{v\text{-}u}$ a ustanovení komunikačního kanálu s klíčem relace K_s.

3.2.2 Popis částí protokolu

Setup

Prvním protokolem je Setup a je vykonáván vydavatelem. Algoritmus vygeneruje systémový klíč K o délce 16B náhodné posloupnosti a uloží si ho. Systémový klíč je brán jako soukromý parametr a neopouští systém.

Issue

Protokol Issue je druhým algoritmem vydavatele. Algoritmus má jako vstupní parametr systémový klíč K. V prvním kroku je zderivován systémový klíč K pomocí standardu AES v režimu ECB, výstupem je klíč ověřovatele K_v. V druhém kroku je

zderivován stejným způsobem klíč ověřovatele K_v, výstupem je klíč uživatele $K_{v\text{-}u}$. V posledním kroku jsou vygenerované klíče předány uživateli a ověřovateli.

Show <-> Verify

Algoritmus Show je spuštěn na straně uživatele a současně interaguje s algoritmem Verify na straně ověřovatele. Entita uživatele vygeneruje výzvu *unonce* pomocí náhodné sekvence a společně s identifikátorem uživatelského ID_u zasílá ověřovateli. Ověřovatel si náhodnou sekvenci unonce uloží a odešle vlastní vygenerovaný vnonce se svým identifikátorem ID_v. Na straně uživatele je vypočítán tzv. ukey, který je výsledkem z řetězce znaků "000000000000User"a je zašifrován klíčem uživatele $K_{v\text{-}u}$. Klíč ukey je poté využit k zašifrování řetězce dat (ukey; ID_u, D_v, unonce, vnonce) a poslán ověřovateli.

Na straně ověřovatele nastane fáze ověřování v podobě výpočtu uživatelského klíče $K_{v\text{-}u}$ (klíč uživatele lze získat, pokud se jedná o vytvořený derivát během algoritmu Issue), pomocí kterého se dopočítá ukey. Získaným klíčem uživatele ukey poté dešifruje přijatý řetězec a zkontroluje jej. Pokud jsou přijatá data totožná, ověřovatel vytvoří vlastní klíč vkey (ten vzniká podobným způsobem jako ukey, ale je použit řetězec znaků "00000000Verifier") a vyšle zašifrovaný blok dat s výstupním klíčem relace K_s k uživateli, který dopočítá vkey, zskontroluje data a v případě totožnosti převezme klíč K_s.

4 Vývoj bezpečnostního IoT systému

V rámci praktické části bakalářské práce byly vytvořeny dvě aplikace. První aplikací je ověřovací systém. Ověřovací systém je navržen podle bezpečnostního protokolu v kapitole 3.2, konkrétněji jde o algoritmus Show <-> Verify. V jeho dalších podkapitolách je popsána příprava prostředí (jaké knihovny je potřeba doinstalovat, jak zprovoznit Trust Core a komunikaci s čipovou kartou). Dále se pak věnuje popisu komunikačního modulu systému a popisu jednotlivých aplikací na zařízeních. Druhou aplikací je vydavatelská aplikace. Vydavatelská aplikace vznikla za účelem správy klíčů systému, čipové karty a Trust Core. Odpovídá algoritmům Setup a Issue ve výše odkázaném protokolu. V dalších podkapitolách je blíže popsána aplikace, implementace a struktura systému, tabulky databáze a komunikace s databází, potřebné programy a Jar soubory. Poslední samostatná sekce se věnuje výkonovým testům, kde jsou zhodnoceny rychlosti konkrétních částí protokolu, celkové délky ověřovacího protokolu atd.

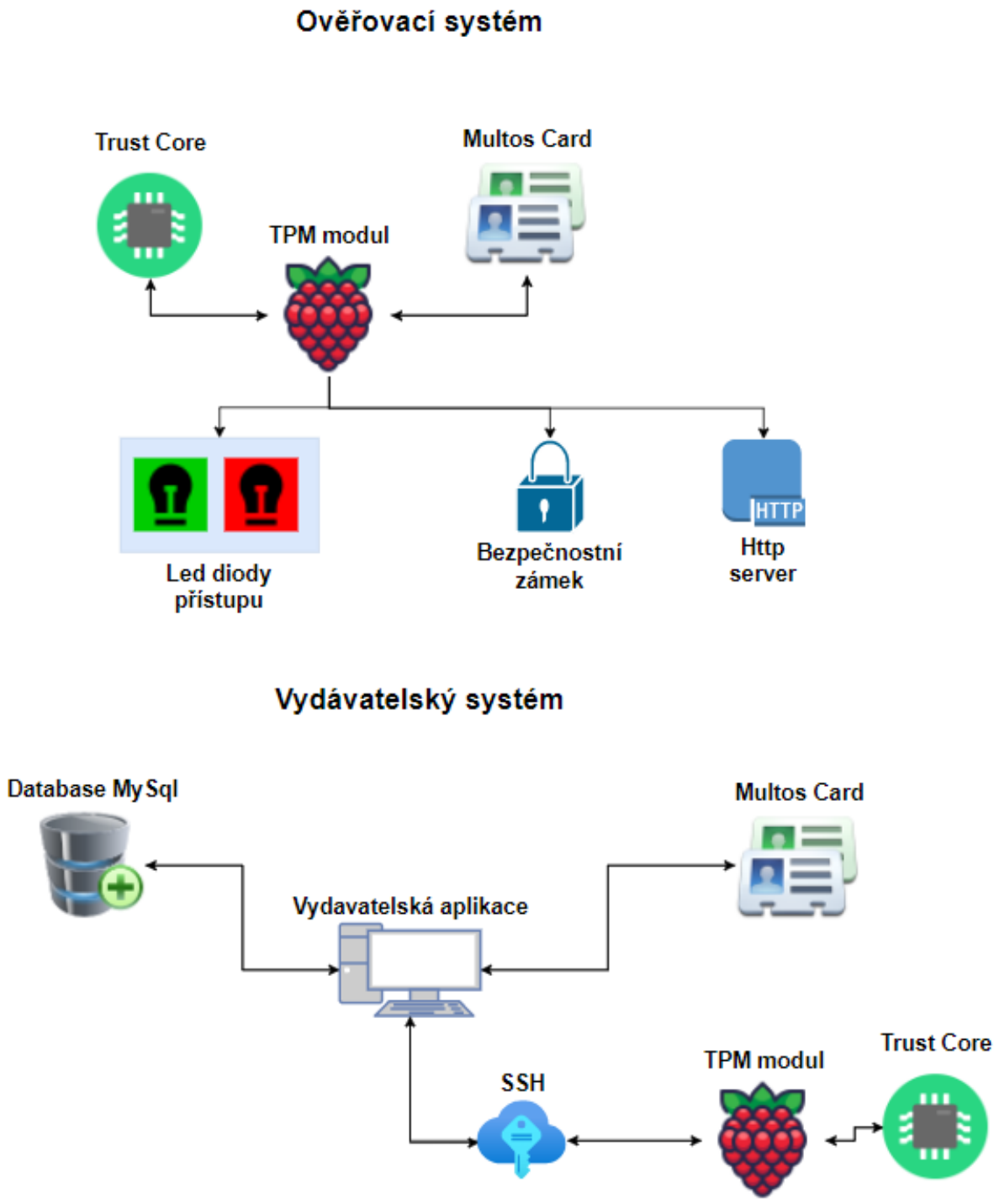
4.1 Ověřovací systém

Ověřovací systém je implementován mezi prvky Raspberry pi, čipovou kartou MultOS MC4 a Trust Core. Protokol je navržen tak, aby dodržoval základní prvky bezpečnosti a poukázal na výpočetní rychlost prvků MultOS.

Raspberry pi je použit jako TPM modul a přeposílá komunikaci mezi prvky. Pokud je uživatel shledán legitimním, tak Raspberry pi přivede napětí po dobu 5 sekund na dva GPIO piny a navíc otevře python Http server, kde se zobrazí výsledek se základními údaji ověřeného uživatele. Přivedením napětí na vybrané piny se rozsvítí zelená led dioda a otevřou se dveře. V opačném případě přivede napětí na jeden pin GPIO, čímž rozsvítí červenou diodu. Http server tentokrát zobrazí výsledek autentizace bez uživatelských dat.

Čipová karta MC4 zaujímá roli uživatele, snaží se tedy autentizovat proti ověřujícímu prvku srze TPM modul. Komunikace mezi TPM modulem a čipovou kartou probíhá přes USB čtečku (komunikace bude popsána níže v sekci Komunikace s Multos Kartou). Karta na vyzvání TPM modulu posílá zprávy, které jsou převzaty a dál předány na ověřující prvek. Výsledkem pro uživatele je otevření dvěří nebo odepření přístupu.

Jako ověřující prvek je použit mikrokontroler Trust Core. Trust Core ja naprogramován tak, aby přijímal výzvy od TPM modulu a vyhodnocoval uživatelem zaslané zprávy, kdy se výzvy vyhodnocují jedna za druhou podle autentizačního protokolu. Pokud jsou uživatelem zaslané výzvy zhodnoceny pozitivně, zašle uživateli klíč relace a předá zprávu TPM modulu, že se uživatel úspěšně autentizoval.



Obr. 4.1: Schéma systému

4.1.1 Příprava vývojového prostředí

Nedílnou součásti je zprovoznění všech IoT prvků. Prvky samy o sobě nejsou schopné vzájemné komunikace a funkčního stavu. V této podkapitole bude zpracováno zpro-

voznění technologie na Raspberry pi pro čipovou kartou a Trust core. Podrobnější instrukce naleznete v příloze C, kde je zaznamenán celý instalační postup a v příloze D, kde je znázorněn uživatelský manuál.

Potřebné knihovny pro Raspberry pi

Aby na Raspberry pi bylo možné zprovoznit Ověřovací systém, je nutné doinstalovat několik knihoven. Mezi tyto knihovny patří:

- Zlib - komprese dat.
- Openssl - podpora krytografie.
- Gmp - podpora krytografie.
- PCSC - komunikace s čipovou kartou.
- Libncurses5 - zprovoznění Trust Core.

Komunikace mezi Raspberry pi a Trust Core

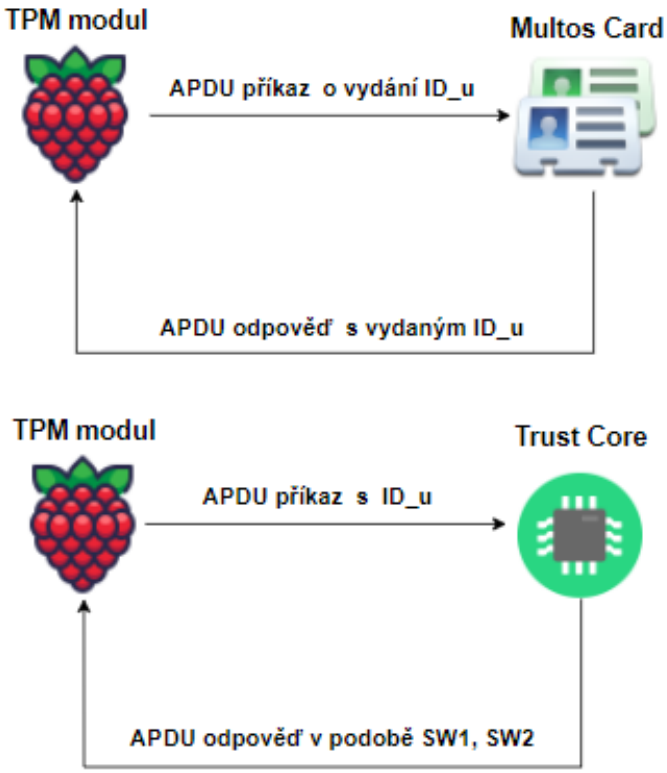
Trust Core vyžaduje povolení I2c rozhraní pro zpřístupnění GPIO pinů, jejichž prostřednictvím komunikuje. Dále je vyžadována knihovna lmultosio a stažení základních funkcí od výrobce. Funkčnost lze jednoduše ověřit příkazem *lsm*. Knihovna Multosio je využita pro komunikaci s Trust Core. Obsahuje základní strukturu APDU zpráv, inicializaci spojení, resetování zařízení nebo výběr aplikace.

Realizace Http serveru

Pro realizaci Http serveru je nutné Grafické zobrazení. Pro tuto problematiku je využit Program VNC Viewer, který podporuje vzdálené připojení (SSH) s grafickým zobrazením. VNC viewer je nutné opět povolit v nastavení systému pro Rasberry pi. Http server je realizován prostřednictvím programovací jazyk python. Rozšířené funkce pro realizaci python serveru jsou již předinstalovány v systému raspberian. Spuštění lokálního serveru probíhá přes samostatnou příkazovou řádku a příkazem python3 -m http.server 8000, kde argument -m představuje převzetí souborů v aktuální složce.

4.1.2 Komunikace mezi kartou a Trust Core

Komunikace mezi prvky probíhá pomocí APDU příkazů a APDU odpovědí viz. část Komunikační protokol APDU. APDU komunikace je vyvolávána v podobě APDU příkazů skrze TMP modul, který ji předává dál. Příklad o komunikaci je zachycen na Obr. 4.2: Ukázka komunikace. Na obrázku je zachycena komunikace o žádost ID uživatele, které je v následujícím kroku předanáno ověřujícímu prvku.



Obr. 4.2: Ukázka komunikace

4.1.3 Aplikace na Kartě MC4

První část aplikace běží na čipové kartě MC4. Čipová karta má za úkol posílat a odpovídat na výzvy ověřovacího prvku Trust Core a Vydávací aplikace. V téhle sekci budou rozebrány části kódu a důležité části blíže popsány. Alikace patří k implementaci autentizačního protokolu a Vydávacího protokolu.

AID: F0 00 00 01
CLA: 0x80

INS 0x01: ID uživatele
Karta se podívá do své statické paměti, překopíruje do globální proměnné své ID.

INS 0x10: Změna privátního klíče
Zkopírování nového klíče do statické proměnné, nový klíč je zaslán skrze vydavatelskou aplikaci.

INS 0x15: uNonce
Vygenerování náhodného řetězce o 16B.

INS 0x20: ID ověřovatele
Převzetí ID ověřovatele.

INS 0x25: vNonce
Převzetí náhodného řetězce o 16B zaslaného ověřovatelem.

INS 0x30: Show

```
 //compute ukey
multosBlockEncipherECB(ALGORITHM_AES, AES_BLOCK_SIZE,
(BYTE *)"000000000000User",(BYTE *)ukey, 16,(BYTE *)k_vi_ui);
```

Blokové šifrování AES v modu ECB. Za pomoci řetězce "000000000000User"a klíče uživatele (k_vi_ui) vypočítáme ukey.

```
 //decrypted data
memcpy(&ue_data_output[0],&ue_identifier,AES_BLOCK_SIZE);
memcpy(&ue_data_output[16],ve_id,AES_BLOCK_SIZE);
memcpy(&ue_data_output[32],&ue_nonce,AES_BLOCK_SIZE);
memcpy(&ue_data_output[48],ve_nonce,AES_BLOCK_SIZE);
```

Zkopírování ID uživatele, ID vydavatele a náhodných řetězců uživatele a ověřovatele do bajtového pole ue_data_output.

```
 // AES (ukey; ID_ui,ID_vi, unonce, vnonce)
multosBlockEncipherCBC(ALGORITHM_AES, 4 * AES_BLOCK_SIZE,
(BYTE *)&ue_data_output,(BYTE *)&ue_cipher_output, AES_IV_LENGTH,
(BYTE *)"0000000000000000", AES_BLOCK_SIZE,(BYTE *)ukey);
```

Blokové šifrování AES v modu CBC. IV (inicializační vektor) je zvolen jako řetězec šestnácti nul. Výsledkem funkce je šifrování řetězce dat o délce 64B.

INS 0x35: Verify

```
 //decipher
multosBlockDecipherCBC(ALGORITHM_AES, 5 * AES_BLOCK_SIZE,
(BYTE *)&ve_cipher_input, (BYTE *)&ve_decipher_input,
AES_IV_LENGTH, (BYTE *)"0000000000000000", AES_BLOCK_SIZE,
(BYTE *)vkey);
```

Dešifrování příchozích dat včetně Session klíče. Data jsou sešifrována pomocí vkey klíče, který se vypočítá nad prezentovanou funkcí na dešifrování.

```
if(memcmp(&ve_decipher_input[0], ve_id, VERIFIER_ID_LENGTH) ||
    memcmp(&ve_decipher_input[16], &ue_identifier, USER_ID_LENGTH)
    memcmp(&ve_decipher_input[32], ve_nonce, NONCE_LENGTH) ||
    memcmp(&ve_decipher_input[48], &ue_nonce, NONCE_LENGTH)){

        ExitSW(ERR_SECURITY);
}

 // copy session key
memcpy(sessionKey, &ve_decipher_input[64], AES_BLOCK_SIZE);
```

Po dešifrování výsledného řetězce se porovnávají proti sobě data, které již karta dostala v předchozích krocích nebo jí náleží (ID ověřovatele, ID uživatele, Vnonce, Unonce). Pokud některý údaj není správný, karta vrací chybový kód ERR_SECURITY. V případě, že údaje sedí, bere si přiložený klíč relace na konci dešifrovaného řetězce. Zde končí ověřovací protokol.

INS 0x50: Uživatelská data
Tuto instanci využívá Vydavatelský systém pro nahrání uživatelských dat na kartu nebo jejich aktualizaci. Přenášenými daty jsou: jméno uživatele, emailová adresa, telefonní číslo, jméno ověřovacího prvku, možný čas přístupu a uživatelské heslo.

INS 0x66: Přenos uživatelských dat
Instance je využita pro vydání uživatelských dat na Raspberry pi. Data jsou předána http serveru pro zobrazení na webové stránce.

4.1.4 Aplikace Trust Core

Aplikace na Trust Core.

AID: F0 00 00 02 00 00 01
CLA: 0x80

INS 0x01: ID ověřovatele
Trust Core vydá své ID.

INS 0x10: Změna privátního klíče
Zkopírování nového klíče do statické proměnné. Nový klíč je zaslán skrze vydavatelskou aplikaci za pomoci vzdáleného připojení SSH.

INS 0x15: vNonce
Vygenerování náhodného řetězce o 16B.

INS 0x20: ID uživatele
Převzetí ID uživatele.

INS 0x25: uNonce
Převzetí náhodného řetězce o 16B zaslaného uživatelem.

INS 0x30: Show
Uložení zašifrovaného řetězce od uživatele o délce 64B (ID uživatele, ID vydavatele, uNonce, Vnonce).

INS 0x35: Verify

```
multosBlockEncipherECB(ALGORITHM_AES, AES_BLOCK_SIZE,
    (BYTE *)&ue_id,(BYTE *)k_vi_ui, 16,(BYTE *)ve_keys);
```

Výpočet klíče uživatele, který je následně použit pro výpočet ukey a vkey.

```
multosBlockEncipherECB(ALGORITHM_AES, AES_BLOCK_SIZE,
    (BYTE *)"000000000000User",(BYTE *)ukey, 16,(BYTE *)k_vi_ui);

multosBlockEncipherECB(ALGORITHM_AES, AES_BLOCK_SIZE,
    (BYTE *)"00000000Verifier",(BYTE *)vkey, 16,(BYTE *)k_vi_ui);
```

Výpočet klíčů ukey a vkey.

```
multosBlockDecipherCBC(ALGORITHM_AES, 4 * AES_BLOCK_SIZE,
    (BYTE *)&ue_cipher_output, (BYTE *)&ue_decipher_output,
    AES_IV_LENGTH, (BYTE *)"0000000000000000", AES_BLOCK_SIZE,
    (BYTE *)ukey);
```

Dešifrování řetězce dat z předchozí instance 0x30: Show. Zašifrování výsledného řetězce. Řetezec je poté zaslán kartě.

```
multosBlockEncipherCBC(ALGORITHM_AES, 5 * AES_BLOCK_SIZE,
    (BYTE *)&ve_cipher_data,(BYTE *)ve_cipher_output,
    AES_IV_LENGTH,(BYTE *)"0000000000000000", AES_BLOCK_SIZE,
    (BYTE *)vkey);
```

4.1.5 Aplikace Raspberry pi

Aplikace na Raspberry pi představuje dorozumívací rozhraní pro Multos kartu MC4, Trust Core. Aplikace zpracovává, vyhodnocuje a přeposílá požadavky dle naprogramovaných tříd. Třídy jsou strukturovány a členěny do jednotlivých souborů, aby se v nich lépe orientovalo a popřípadě dále vyvíjelo. Aplikaci je možné spouštět přes příkazovou řádku díky přepínačům a v neposlední řadě spouštět konečné vyhodnocování v podobě odepření přístupu závislém na čase, rozsvícení patřičné ledky dle výsledku protokolu, zobrazení výsledku na http server a otevření dveří. Jeden z přepínačů je také využit pro vydavatelskou aplikaci. Aplikace je zhotovena v jazyce C.

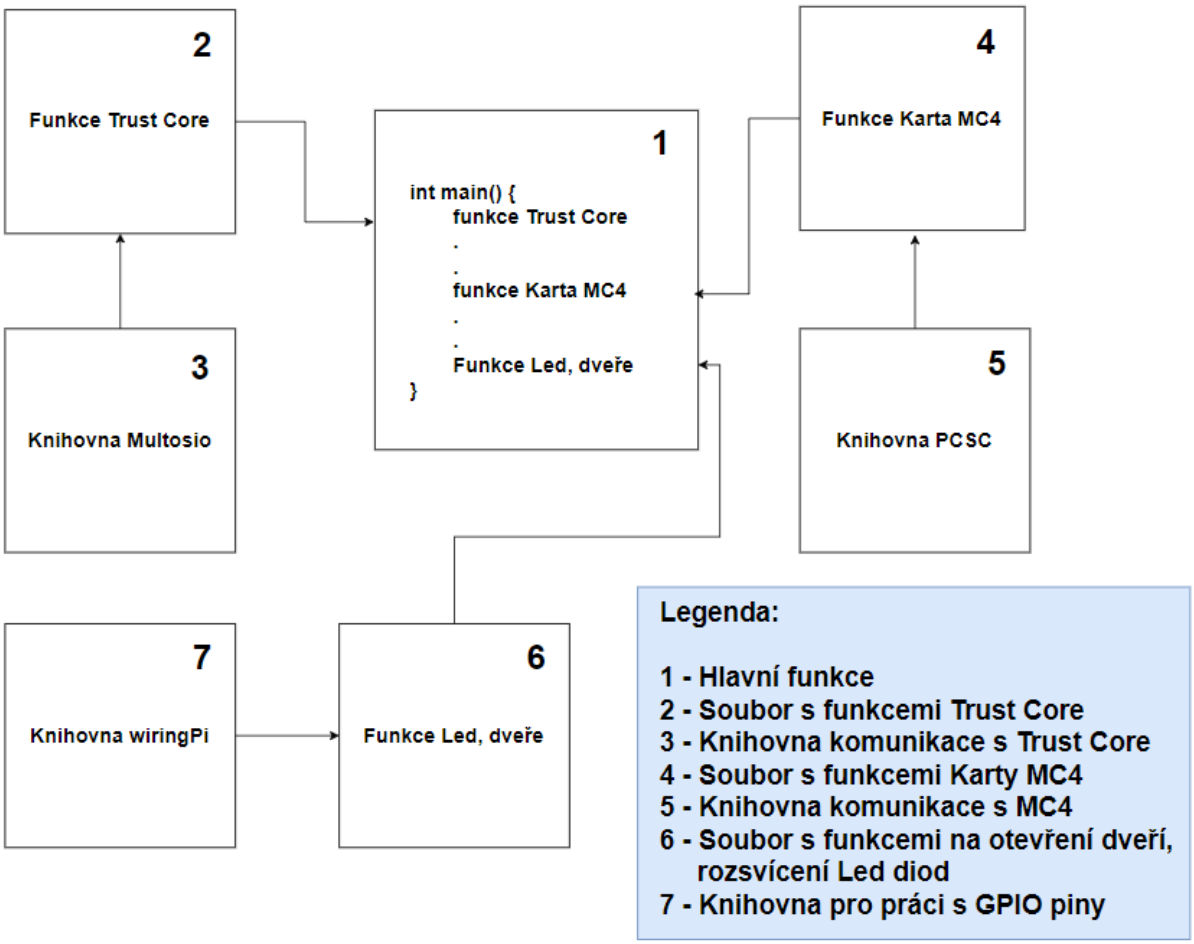
Struktura programu

Struktura programu je členěna a tvořena tak, aby byla přehlednou a dalo se v ní lehce orientovat nebo rozšiřovat. Funkce pro jednotlivé zařízení Iot jsou zvlášť v souborech, které obsahují oddělenou hlavičku a těla funkcí. Jednotlivé funkce jsou volány do hlavní (int main) funkce, přičemž voláním jednotlivých funkcí je sestaven autentizační protokol nebo jsou funkce využity pro dílčí řešení (např. předání ID ověřovatele pro vygenerování klíče). Strukura programu je zachycena na Obr. 4.3: Schéma program na Raspberry pi.

Rozdělení struktury dle přepínačů

Každý program napsaný v C musí být před spuštěním zkompilován. Zkompilování programu způsobí převedení do binárního kódu, který lze jednoduše volat z příkazové řádky. Přepínače jsou využity, protože operační systém běžící na Raspberry pi je typu linux, tudíž jsou jednoduše volány z příkazové řádky ve formě ./program -přepínač. Abstraktní znázornění přepínačů a jejich základní funkce jsou znázorněny na Obr. 4.4: Rozdělení programu do přepínačů.

Program je rozdělen do čtyř přepínačů provádějící posloupnost příkazů. První dva přepínače jsou využívány primárně pro SSH spojení, kde probíhá komunikace o vydání nebo změnu soukromého klíče pro Trust Core. První přepínač vyžádá apdu zprávou ID Trust Core, který je následně přenese do Vydavatelské aplikace, zde je zašifrováno vydavatelským klíčem a zasláno zpět SSH spojením jako argument pro



Obr. 4.3: Schéma programu na Raspberry pi

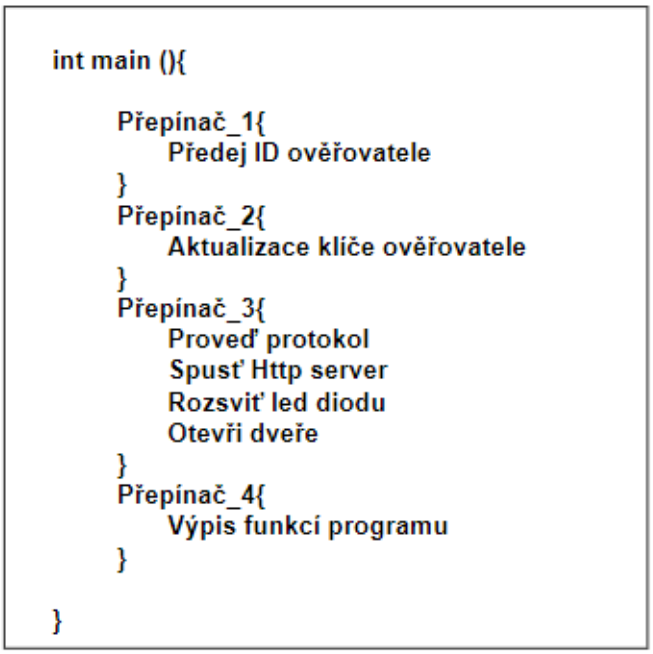
druhý přepínač. Přepínač_3 je pro provedení bezpečnostního protokolu a vypsání logu do příkazové řádky. Přepínač_4 vypíše funkcionalitu jednotlivých přepínačů a použití.

Komunikace s Multos Kartou

Multos Karta komunikuje přes USB čtečku. Aby Raspberry pi mohlo komunikovat se čtečkou karet, je nutné nainstalovat a výsledný program zkompilovat s knihovnou PCSC. Poté je možné komunikovat skrze APDU zprávy.

Komunikace s Trust Core

Trust Core vyžaduje povolení I2c rozhraní pro zpřístupnění GPIO pinů, přes které Trust Core komunikuje. Dále jsou vyžadovány knihovny libncurses5, lmultosio a stažení základních funkcí od výrobce. Instalační manuál lze nalézt v příloze. Funkčnost lze jednoduše ověřit příkazem *lsm*. Knihovna Multosio je využita pro komunikaci s



Obr. 4.4: Rozdělení programu do přepínačů

Trust Core a obsahuje základní strukturu APDU zpráv, inicializaci spojení, resetování zařízení nebo výběr aplikace.

Knihovna wiringPi a její využití

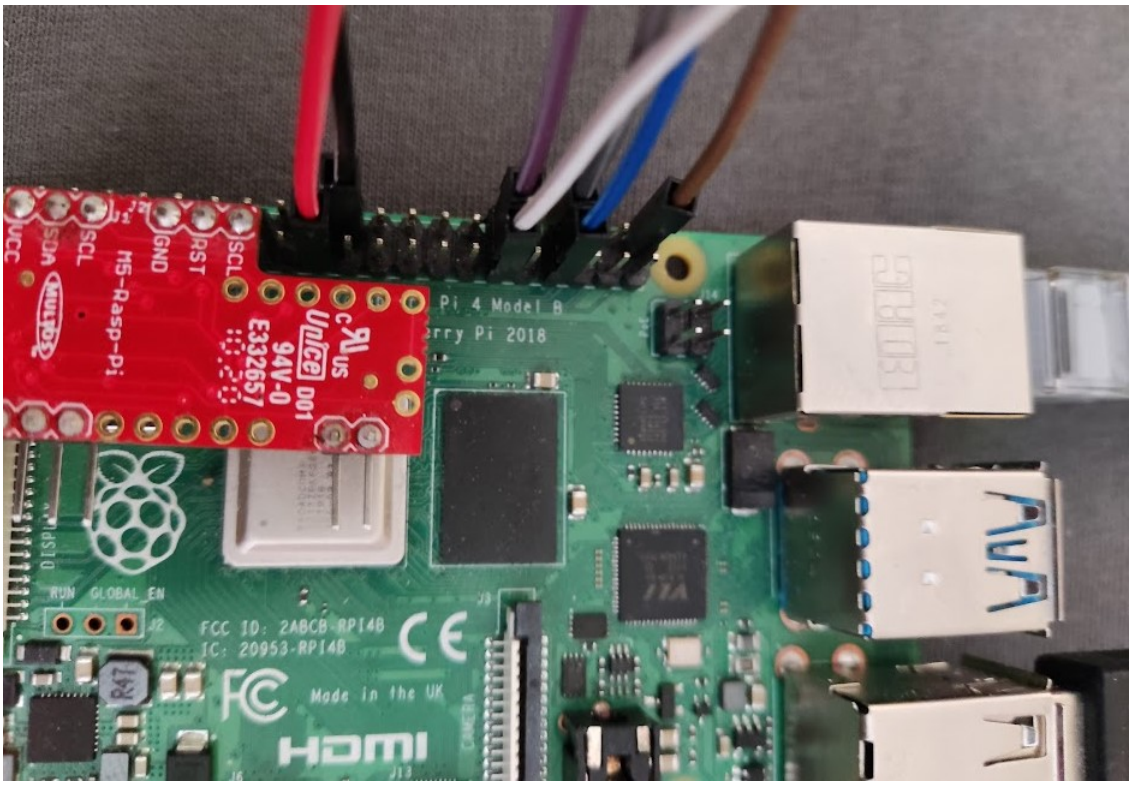
Knihovna WiringPi je nástrojem pro práci s GPIO piny napsaná v jazyce C. WiringPi je podporovaná ve všech verzích Raspberry pi. Zmíněná knihovna je využívána pro přivedení napětí na piny, které jsou zapojeny do obvodu s LED diodami nebo elektrickými dveřmi pomocí relé modulu. Na Obr. 4.5 je znázorněn příklad zapojení GPIO pinů. Ukázka kódu pro LED diody je níže:

```
#define LED 29

wiringPiSetup() ;
pinMode (LED, OUTPUT) ;
digitalWrite (LED, HIGH) ;  // On
delay (5000) ;    // mS
digitalWrite (LED, LOW) ; // Off
```

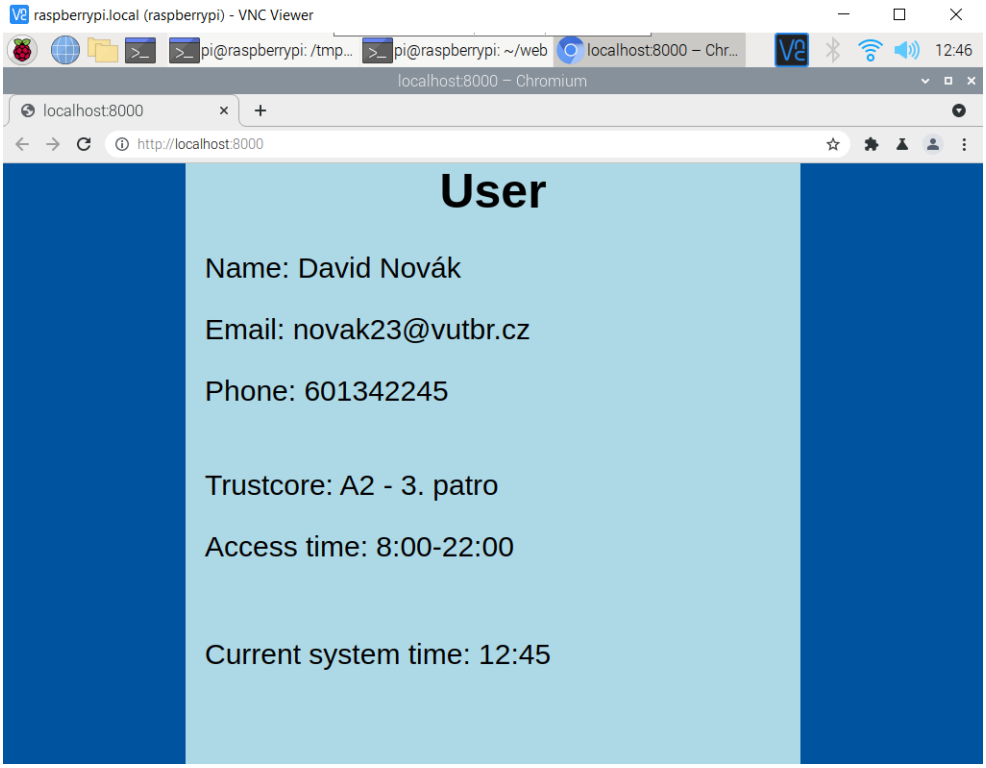
Výstup Aplikace

Jak bylo dříve zmíněno, výstup autentizačního protokolu je zapnutí patřičné ledky, otevření zámku a zobrazení výsledku na http server. V této části budou znázorněny

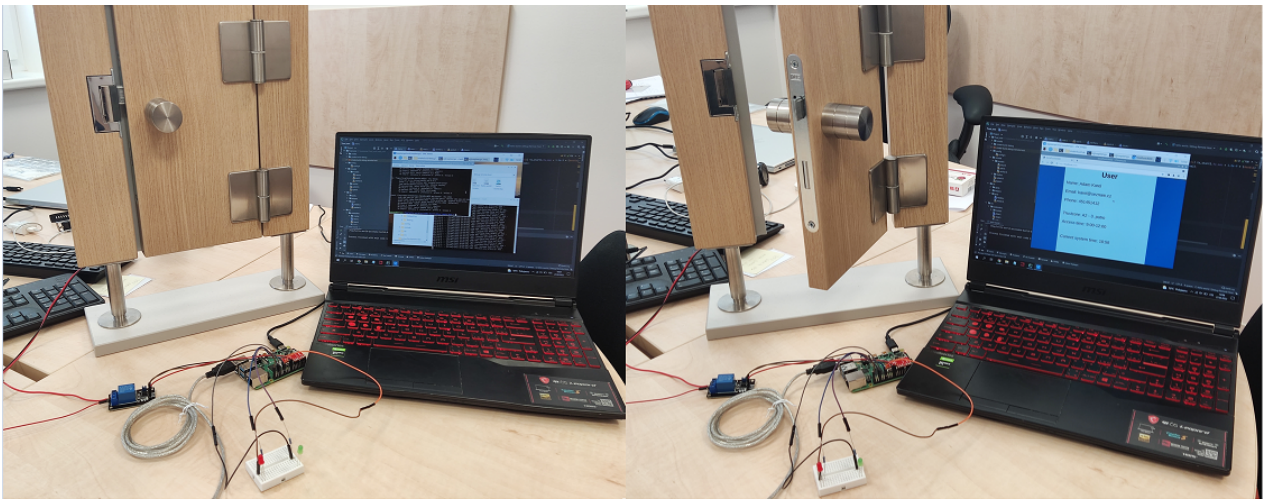


Obr. 4.5: Zapojení GPIO pinů

výstupy. Na Obr. 4.6 se nachází výsledek úspěšné autentizace v podobě výstupu Http serveru. Jsou zde vidět základní uživatelská data, která budou později zpracována a znázorněna ve vydavatelské části, a aktuální systémový čas Raspberry pi, který poukazuje na validní přístup na základě času. Celé znázornění systému je na Obr. 4.7, kde jsou zachyceny záběry před a po provedení ověřovacího protokolu. Na obrázku můžeme vidět Raspberry pi, na které je napojena čtečka karet s čipovou kartou, Trust Core a přes GPIO piny napojeny dva obvody. Obvod pro relé modul, který dále pokračuje v obvodu s elektronickými dveřmi, a obvod pro LED diody. Dále je na obrázku znázorněn průběh protokolu jako výstup na počítači.



Obr. 4.6: Zobrazení autentizovaného uživatele přes VCN Viewer



Obr. 4.7: Znázornění výstupu protokolu, před/po

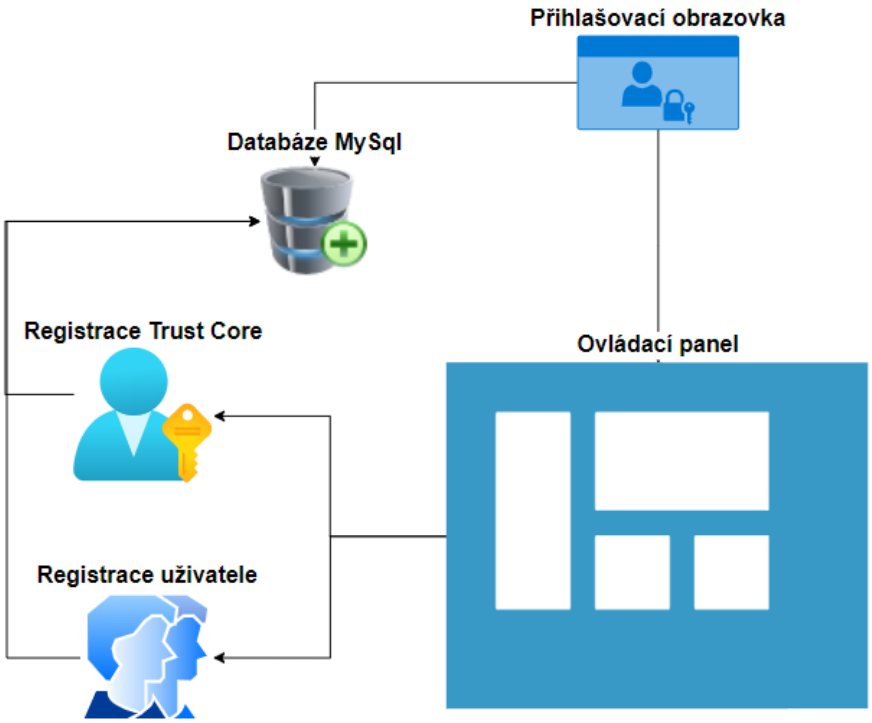
4.2 Vydavatelský systém

Vydavatelský systém byl zhotoven pro operace s kartou multos a Trust Core. Kartě lze předat osobní data jako jsou jméno a příjmení uživatele, email, telefonní číslo, přidělený ověřovací prvek a heslo. Systém rozpoznává novou kartu, ukládá ji do databáze, umožňuje aktualizaci dat na kartě nebo v databázi a derivuje ID uživatele

za pomoci osobního klíče Trust Core pro přidělení soukromého klíče zvolené kartě. Pro práci s Trust Core používá uložení do databáze, registrování prvku a předání soukromého klíče.

4.2.1 Struktura aplikace

Struktura aplikace je členěna do několika částí. Jeden z primárních důvodů je přehlednost kódu, který je rozdělen do tříd podle toho, co vykonává. Kód je tím pádem čitelnější a vývojář ho snadněji pochopí. Při nastudování konkrétní problematiky není těžké ho rozšířit. Druhým důvodem je uživatelská přehlednost, jelikož každá třída znázorňuje jedno okno a konkrétní prvky (tlačítka), které se v okně nacházejí, znázorňují oddělené funkce vykonávající posloupnost operací. Aplikace je tvořena tak, aby efektivně komunikovala s databází a s prvky bezpečnostního systému, které spravuje. Z uživatelského hlediska je aplikace snadno ovladatelná, jde pouze o doplňování patřičných údajů do buněk aktuálního okna a zmáčknutí patřičných tlačítek. Uživatel je rovněž informován o výsledcích, které provádí (např. aktualizace dat na kartě a v databázi), dostává kladné či záporné odpovědi. Po upravení konkrétního problému, které aplikace nahlásí nakonec dovede uživatele k pozitivní reakci ve formě provedení. Schéma provedené je znázorněno na Obr. 4.8.



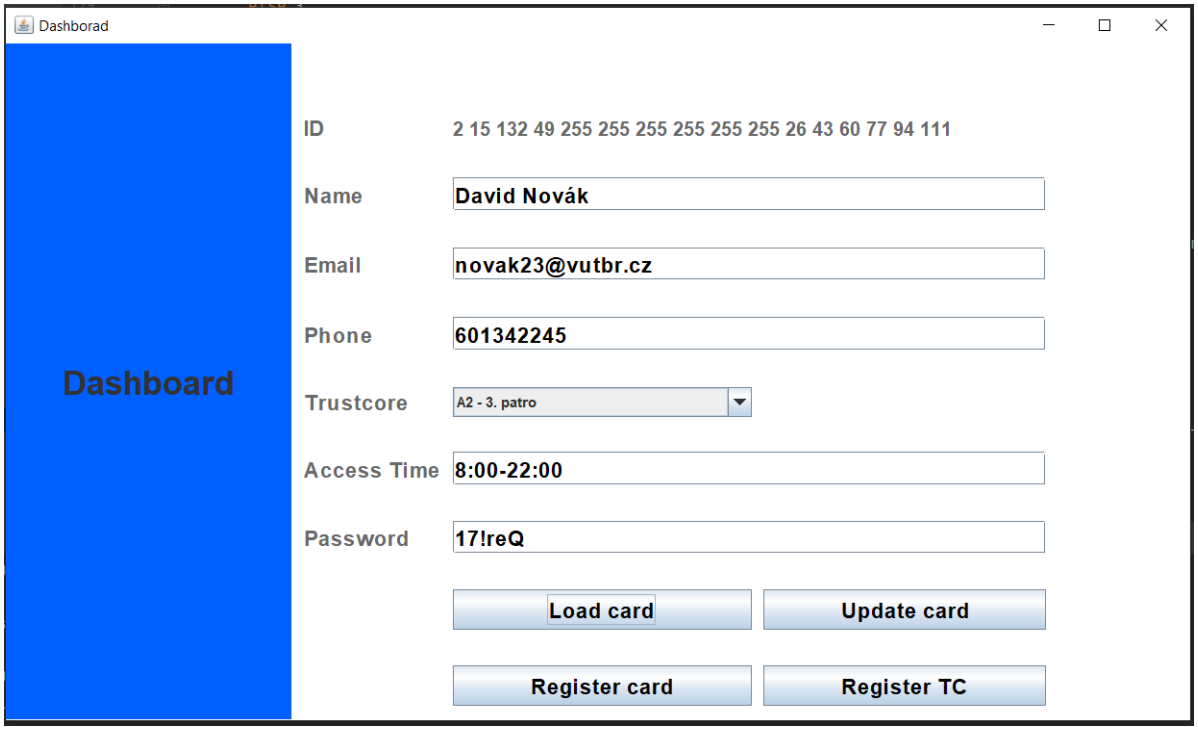
Obr. 4.8: Struktura vydavatelské aplikace

Přihlašovací obrazovka

Přihlašovací obrazovka je první věcí, která se zobrazí po spuštění aplikace. Aplikace je volána z hlavní třídy Dashbord. Tento snímek požaduje přihlašovací údaje, které se postisknutí tlačítka Login zpracují, načež se aplikace připojí do databáze a porovná údaje se zadanými daty. Pokud je nalezena shoda, je uživatel autentizován.

Ovládací panel

Ovládací panel je hlavní, ovládací třída celého programu. Zasahuje do ostatních naprogramovaných tříd a komunikuje s databází. Vyobrazení je na Obr.4.9, kde jsou znázorněna čtyři tlačítka. Po stisknutí tlačítka Load card se zobrazí data aktuálně připojené karty. Karta je požádána o své ID přes APDU žádost. Po získání ID karty se program podívá do databáze a pokusí se vyhledat ID. Pokud ID vyhledá, tak se zobrazí data uživatele. V opačném případě je uživatel informován, že karta není registrována. Tlačítko Update card po stisknutí aktualizuje data na kartě a v databázi. Data v databázi jsou aktualizována podle vyhledaného ID karty a aktualizovaná data jsou zaslána přes APDU žádost.Tlačítka Register card a Register TC vedou do samostatných registrovacích sekcí, proto budou rozebrány samostatně.



Obr. 4.9: Ovládací panel

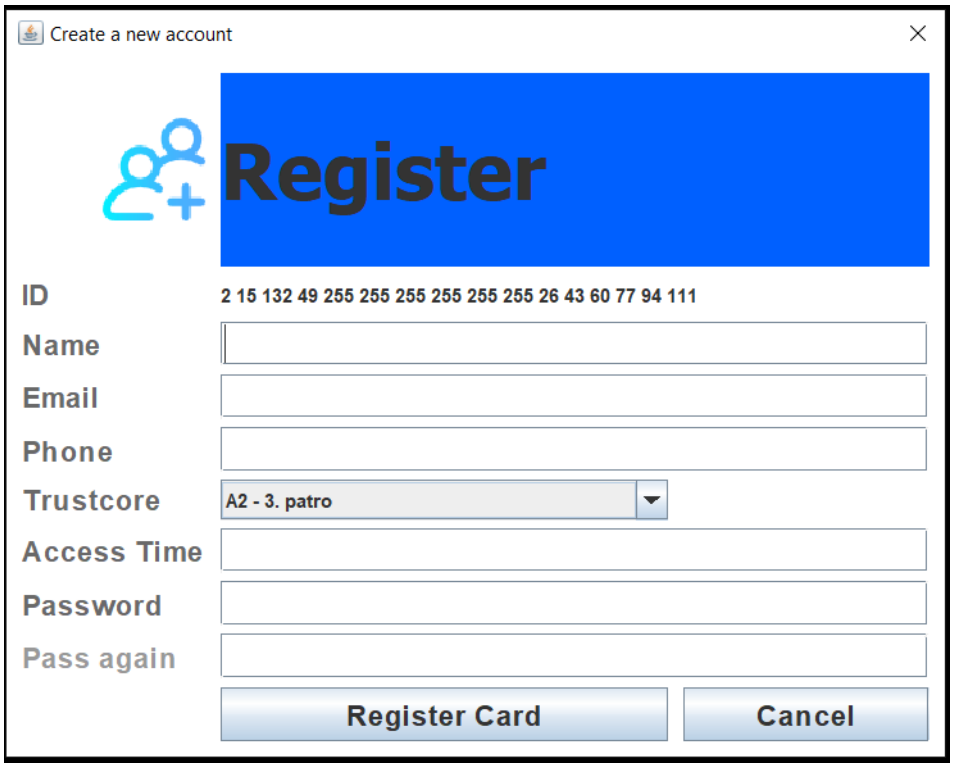
Registrace Trust Core

Registrace nového Trust Core probíhá po stisknutí tlačítka Register TC. Po tomto kroku se zobrazí nové okno pro registraci, kde je zadavatel vyzván k zadání IP adresy a přihlašovacích údajů ke konkrétnímu Raspberry pi. Je tomu tak, protože s Trust Corem lze komunikovat pouze přes Raspberry pi, tudíž je využito vzdáleného připojení SSH. SSH připojení má dvě fáze. První kontaktuje část C program, kde požádá o ID Trust Core, načež je vráceno zpět. Druhým krokem je zadání popisu (to slouží pouze pro lokalizaci Trust Core pro vydavatele, vydavatel také podle tohoto popisu vybírá patřičný Trust core pro kartu) a po stisknutí druhého tlačítka se vygeneruje nový soukromý klíč a pošle se opět SSH spojením na Raspberry pi, kde kontaktuje Trust Core opět přes C program a předá klíč, který se následovně nahraje na ověřovací prvek. Úspěšný výsledek je znázorněn zelenou fajfkou na obrázku 4.10.

Obr. 4.10: Registrace Trust Core

Registrace uživatele

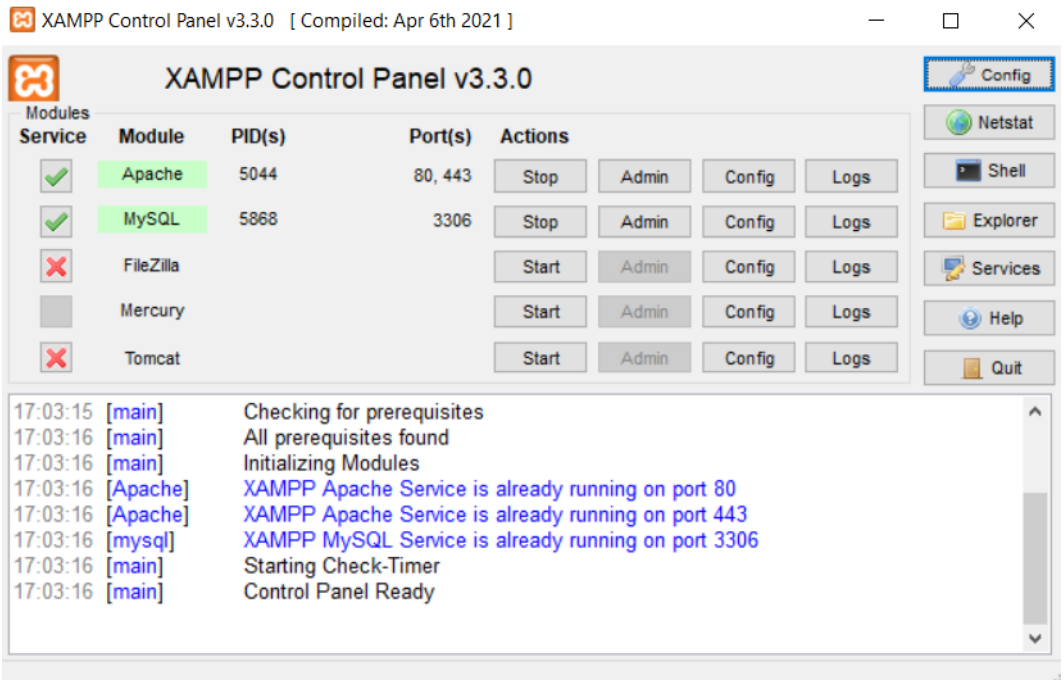
Registrace uživatele probíhá po stisknutí Register user na ovládacím panelu, zadavatel je přesměrován do Registraního okna viz. 4.11, které obsahuje několik kolonek pro zadání údajů. Údaje musí být zadány všechny, jinak není registrace uživatele povolena. Dále je hlídána délka datových řetězců, aby nepřeteklo bajtovové pole pro přenos, ale i pro privátní proměnné na kartě patřící konkrénímu údaji. Pole pro časové rozmezí přístupu (Access Time) je navíc kontrolováno formátem zadaného času. Zadavatel navíc musí vybrat patřičný Trust Core, proti kterému se bude uživatel autentizovat. Pokud jsou všechna data zadána správně, je nová karta zapsána do databáze. Také se vygeneruje nový klíč uživatele, který se pošle společně s ostatními osobními daty na kartu. Zadavatel bude upozorněn informační hláškou, že vše proběhlo v pořádku.



Obr. 4.11: Registrace Uživatele

4.2.2 XAMPP

XAMPP je open-source software vyvinutý společností Apache. Prohram XAMPP obsahuje distribuce Apache pro server Apache, MariaDB, PHP a Perl pro hostování lokálních serverů. XAMPP je využit k hostování databáze MySQL.



Obr. 4.12: Program XAMPP

4.2.3 Databáze MySQL

Jak bylo dříve zmíněno, ve Vydavatelské aplikaci se zpracuje více entit. Konkrétněji Karty - uživatelé a Trust Core - ověřovatelé. Ukládání do databáze bylo zvoleno kvůli archivaci a snadnější dostupnosti informací, co se nachází na konkrétní kartě nebo Trust Core, v případě karty, komu daný prvek patří, a kde se nalézá v případě Trust Core. Databáze je tvořena dvěma oddělenýma tabulkama, které znázorňují konkrétní entitu. Tabulka karet sdílí popis lokace Trust Core, aby bylo jednoznačně zřetelné, ke kterému ověřovacímu prvku může přistupovat.

4.2.4 Použité Jar soubory

MySQL konektor

MySQL konektor je využit pro proporojení databáze s Vydavatelskou aplikací. Komunikace probíhá v rámci kódu pomocí standardních SQL syntaxí. Níže můžete vidět příklad využití MySQL konektoru pro připojení do databáze:

```
String MYSQL_SERVER_URL = "jdbc:mysql://localhost/";
String DB_URL = "jdbc:mysql://localhost/CardDatabase?serverTimezone=
String USERNAME = "root";
String PASSWORD = "";
Connection conn = DriverManager.getConnection(MYSQL_SERVER_URL,
```

```
USERNAME, PASSWORD);
Statement statement = conn.createStatement();
```

Jsch-extension

Jar soubor jsch-extension je využit pro realizaci SSH spojení přímo z kódu aplikace. SSH je využito pro komunikaci s Trust Core, kde je sestaveno spojení a spuštěno části kódu Ověřovací aplikace pomocí přepínače pro předání informací. Příklad sestavení SSH spojení a zapnutí příkazu je zobrazeno níže:

```
    //sestavení SSH spojení
DefaultSessionFactory sessionFactory =
    new DefaultSessionFactory(name, IP, 22);
sessionFactory.setPassword(password);

    //uskutečnění spojení
CommandRunner runner = new CommandRunner(sessionFactory);

    //realizace příkazu
command = cd /home/pi && ./authentication_system -i;
result = runner.execute(command);
```

4.3 Výkonové testy bezpečnostních prvků a aplikací

Výkonové testy proběhly pro několik vybraných událostí, které jsou podstatné pro chod obou zkonstruovaných systémů. Rychlost nejnákladnějších událostí ovlivňuje odezvu celého systému, proto je dobré mít hlavní operace co nejrychlejší.

Tabulka 4.1 zaznamenává hodnoty z Ověřovatelského systému. Jsou zde proti sobě porovnány dvě hlavní části na kartě a Trust Core, kde je hlavním úkolem dešifrovat AES řetězec v módu CBC (karta 80B, Trust Core 64B) a v případě Trust Core navíc zašifrovat AES řetězec v módu CBC o 80B, které se karta v tomto porovnání bude snažit dešifrovat. Druhým vyhodnocením je celková doba průběhu protokolu (bez výstupu jako je otevření zámku). Třetím měřením je celková doba trvání ověřovacího protokolu a posledním je přesun uživatelských dat z karty na Raspberry pi, kde jsou poté zobrazeny jako výstup http serveru.

Měření vydavatelské aplikace je zachycen v tabulce 4.2. Mezi měřené veličiny patří základní operace jako vydání ID, které se hojně využívá, odeslání uživatelských

dat na kartu s propsáním do databáze, vygenerování soukromých klíčů a přeposlání soukromého klíče Trust Core přes SSH.

	1.	2.	3.	4.	5.
Trust Core	91 ms	81 ms	88 ms	101 ms	95 ms
Karta MC4	342 ms	360 ms	417 ms	330 ms	380 ms
Celkový čas protokolu	1719 ms	1403 ms	1623 ms	1578 ms	1807 ms
Přesun uživatelských dat	153 ms	133 ms	160 ms	174 ms	127 ms

Tab. 4.1: Výkonové měření ověřovatelského systému

	1.	2.	3.	4.	5.
Vydání ID uživatele	50 ms	33 ms	44 ms	53 ms	55 ms
Zpracování uživatelských dat	342 ms	360 ms	417 ms	330 ms	380 ms
Vygenerování soukromých klíčů	0.29 ms	0.26 ms	0.22 ms	0.27 ms	0.26 ms
SSH spojení s nahráním klíče TC	858 ms	988 ms	817 ms	905 ms	833 ms

Tab. 4.2: Výkonové měření vydavatelské aplikace

Závěr

V rámci této bakalářské práce bylo dosaženo vyvinutí vysoce sofistikovaného systému kontroly a zabezpečení pro uživatelský přístup. Tato bakalářská práce demonstruje systém kontroly a zabezpečení v rámci elektronických dveří. Součástí řešení je Vydavatelská aplikace, která generuje a schraňuje základní uživatelské údaje a vydává pro využití přístupu soukromé klíče pro uživatelskou kartu, které jsou unikátní. Vydavatelská aplikace zároveň generuje unikátní klíče pro zabezpečovací zařízení Trust Core společnosti Multos.

Systém využívá v rámci svého kryptografického jádra podstatu kryptografických protokolů pro důvěrnost, autenticitu a soukromí na omezených zařízeních a díky tomu umožňuje ověření uživatele. Systém se skládá z částí: Raspbery pi, Multos Trust Core a čipové karty MC4. Součástí práce je vydavatelský systém, který je uživatelsky přívětivý. Zároveň je potřeba implementovaný protokol, který je zanesen do ověřovacího systému. Vydavatelský systém navíc disponuje grafickým rozhraním dovolující uživateli zaregistrovat novou kartu, získat její údaje a upravit je, přičemž jsou všechny operace zaznamenány do databáze. Navíc spravuje Trust Core. Systémy jsou funkční a stabilní a navzájem spolu spolupracují.

Na konci bakalářské práce proběhlo měření, které zaznamenává dobu provedení nejdůležitějších a nejkritičtějších částí obou systémů. Při měření bylo dosaženo času ověření 81 ms pro Trust Core a 330 ms pro kartu MC4 v nejpodstatnější části implementovaného protokolu. Nejúspěšnější čas provádění celého protokolu činí 1403 ms. Ostatní naměřené hodnoty jsou zaznamenány v textu práce. Naměřené hodnoty byly pro funkci systému optimální.

Literatura

[1] *Current research on Internet of Things (IoT) security: A survey. Jalan Sultan Yahya Petra, Kampung Datuk Keramat, 54100 Kuala Lumpur, Wilayah Persekutuan Kuala Lumpur, Malajsie, 2019. Research. Universiti Teknologi Malaysia, Kuala Lumpur.*

[2] *CardWerk Technologies [online]. [cit. 2022-05-30].* Dostupné z URL: `https://cardwerk.com/smart-card-operating-system/`

[3] *What Is a Chip Card?. Chip Card Definition [online]. [cit. 2022-05-20].* Dostupné z URL: `https://www.investopedia.com/terms/c/chip-card.asp`

[4] *What is 'Cryptography'. The Econimics Times [online]. [cit. 2022-05-20].* Dostupné z URL: `https://economictimes.indiatimes.com/definition/cryptography`

[5] *WHAT IS MULTOS?. Multos [online]. [cit. 2022-05-20].* Dostupné z URL: `https://multos.com/`

[6] *Introduction to Multos. Multos international [online]. [cit. 2022-05-20].* Dostupné z URL: `https://www.multosinternational.com/en/products/multos-and-multos-stepone/intro-to-multos/`

[7] *Application Protocol Data Unit (APDU). CardLogix [online]. [cit. 2022-05-20].* Dostupné z URL: `https://www.cardlogix.com/glossary/apdu-application-protocol-data-unit-smart-card/`

[8] *TRUST CORE DETAILS. Multos [online]. [cit. 2022-05-20].* Dostupné z URL: `https://multos.com/support/multos-trust-anchor/developer-boards/trust-core-details/`

[9] *Access Control System. ScienceDirect [online]. [cit. 2022-05-20].* Dostupné z URL: `https://www.sciencedirect.com/topics/computer-science/access-control-system`

[10] *Authentication. ScienceDirect [online]. [cit. 2022-05-20].* Dostupné z URL: `https://www.techtarget.com/searchsecurity/definition/authentication`

[11] *5 User Authentication Methods that Can Prevent the Next Breach. IDRD [online]. [cit. 2022-05-20].* Dostupné z URL: `https://www.idrnd.ai/5-authentication-methods-that-can-prevent-the-next-breach/`

[12] *What is Authorization?. Auth0 [online]. [cit. 2022-05-20].* Dostupné z URL: https://auth0.com/intro-to-iam/what-is-authorization/

[13] *What is Data Integrity and Why Is It Important?. Talend [online]. [cit. 2022-05-20].* Dostupné z URL: https://www.talend.com/resources/what-is-data-integrity/

[14] *What is TPM?. Microsoft [online]. [cit. 2022-05-20].* Dostupné z URL: https://docs.microsoft.com/en-us/windows/iot-core/secure-your-device/tpm

[15] *Symmetric Cryptography. ScienceDirect [online]. [cit. 2022-05-20].* Dostupné z URL: https://www.sciencedirect.com/topics/computer-science/symmetric-cryptography

[16] *Asymmetric Cryptography. ScienceDirect [online]. [cit. 2022-05-20].* Dostupné z URL: https://www.sciencedirect.com/topics/computer-science/asymmetric-cryptography

[17] *HAJNÝ, Jan, Petr DZURENDA, Raúl CASANOVA-MARQUÉS a Lukáš MALINA. Cryptographic Protocols for Confidentiality, Authenticity and Privacy on Constrained Devices [online]. 2020 [cit. 2022-05-20].* Dostupné z URL: https://www.vut.cz/vav/vysledky/detail?vav_id=165663#vysledek-165663.Conferencepaper.

Seznam příloh

A Log ověřovacího protokolu

```
[!] Please insert a working reader...
[+] OK, using reader Gemalto PC Twin Reader (C73B97DF) 00 00
[!] Waiting for card insertion...
[+] OK, connected!

[-] Command: APDU_SCARD_SELECT_APPLICATION
[+] SCard request :
00 A4 04 00 04 F0 00 00 01 00
[+] SCard response:
90 00

[-] Command: APDU_TC_SELECT_APPLICATION
[+] TC request :
00 A4 04 00 04 F0 00 00 02 00 00 01
[+] TC response:
90 00

[+] SCard request :
80 01 00 00 10
[+] SCard response:
02 0F 84 31 FF FF FF FF FF FF 1A 2B 3C 4D 5E 6F 90 00

[+] SCard request :
80 15 00 00 10
[+] SCard response:
9E 38 07 7D 1B 4C E9 F5 A0 C6 E6 42 F3 A4 8A F6 90 00

[+] TC request :
80 20 00 00 10 02 0F 84 31 FF FF FF FF FF FF 1A 2B 3C 4D 5E 6F
[+] TC response :
90 00

[+] TC request :
80 25 00 00 10 9E 38 07 7D 1B 4C E9 F5 A0 C6 E6 42 F3 A4 8A F6
[+] TC response :
90 00
```

```
[+] TC request :
80 01 00 00 10
[+] TC response :
16 2D 79 36 AC 32 87 BD 7E 3B FA 3B A0 AB 00 A9 9000

[+] TC request :
80 15 00 00 10
[+] TC response :
A8 60 E2 4F BF 85 27 09 6E 13 BA 12 74 EB 47 DC 9000

[+] SCard request :
80 20 00 00 10 16 2D 79 36 AC 32 87 BD 7E 3B FA 3B A0 AB 00 A9
[+] SCard response:
90 00

[+] SCard request :
80 25 00 00 10 A8 60 E2 4F BF 85 27 09 6E 13 BA 12 74 EB 47 DC
[+] SCard response:
90 00

[+] SCard request :
80 30 00 00 40
[+] SCard response:
20 1B D2 BE 7A 07 AC 3B C8 1E 0D 8E BD F3 E3 C4 EB E4 2B 16 70
4A 06 BA BC F8 33 36 0A D3 75 76 22 92 61 C1 C8 17 8A 56 87 D3
50 C0 85 AB 95 7B C6 55 D5 50 4A 07 CB CD 0E F2 DE 68 DC C2 2A
5D 90 00

[+] TC request :
80 30 00 00 40 20 1B D2 BE 7A 07 AC 3B C8 1E 0D 8E BD F3 E3 C4
EB E4 2B 16 70 4A 06 BA BC F8 33 36 0A D3 75 76 22 92 61 C1 C8
17 8A 56 87 D3 50 C0 85 AB 95 7B C6 55 D5 50 4A 07 CB CD 0E F2
DE 68 DC C2 2A 5D
[+] TC response :
90 00

[+] TC request :
80 35 00 00 50
```

```
[+] TC response :
41 E4 00 D4 4B EA D4 7D A2 E7 99 1D 28 8B 1F FB BF 7C FC EE D1
47 5C E2 D2 09 0C 01 84 72 8D 95 1C A1 80 08 BE EB C6 B9 14 37
48 CD 04 53 A4 EA 25 FC BA 10 0C 4C 04 F9 02 B8 54 A5 30 99 11
DC 05 B0 63 D2 49 3A 1B 0D 4F 41 8B 46 F7 8B 39 7D 90 00

[+] SCard request :
80 35 00 00 50 41 E4 00 D4 4B EA D4 7D A2 E7 99 1D 28 8B 1F FB
BF 7C FC EE D1 47 5C E2 D2 09 0C 01 84 72 8D 95 1C A1 80 08 BE
EB C6 B9 14 37 48 CD 04 53 A4 EA 25 FC BA 10 0C 4C 04 F9 02 B8
54 A5 30 99 11 DC 05 B0 63 D2 49 3A 1B 0D 4F 41 8B 46 F7 8B 39
7D
[+] SCard response:
90 00

Elapsed time (Case verify Card MC4) = 0.342132 sec.
Elapsed time (Case verify Trust Core) = 0.090620 sec.

Elapsed time (Authentication protocol) = 1.719755 sec.

[+] SCard request :
80 66 00 00 8E
[+] SCard response:
44 61 76 69 64 20 4E 6F 76 C3 A1 6B 2A 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 6E 6F 76 61 6B 32
33 40 76 75 74 62 72 2E 63 7A 2A 2A 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 36 30 31 33 34 32 32 34 35 2A 00 00
41 32 20 2D 20 33 2E 20 70 61 74 72 6F 2A 00 00 00 00 00 00 00
38 3A 30 30 2D 32 32 3A 30 30 2A 2A 00 00 00 00 31 37 21 72 65
51 2A 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 90 00

Elapsed time (Transfer user data) = 0.156053 sec.
```

B Log vydavatelské aplikace

```
Logging to database...
Connected

Terminals: [PC/SC terminal Gemalto USB SmartCard Reader 0]
Selected Terminal: PC/SC terminal Gemalto USB SmartCard Reader 0
ATR: 3B6F00008031E06B844002020D555555555555
Card info: PC/SC card in Gemalto USB SmartCard Reader 0,
protocol T=0, state OK

Choose application:
APDU request: 00A4040004F0000001
APDU response: 9000
It took 55 milliseconds.

Load card:
APDU request: 8001000000
APDU response: 020F8431FFFFFFFFFFFF1A2B3C4D5E6F9000
It took 50 milliseconds.

Register TC:
Generates new Trust Core private key..
New trust Core private key was generated
It took 286201 nanoseconds.

Trust Core private key was updated successfully(SSH)
It took 858 milliseconds.

Terminals: [PC/SC terminal Gemalto USB SmartCard Reader 0]
Selected Terminal: PC/SC terminal Gemalto USB SmartCard Reader 0
ATR: 3B6F00008031E06B844002020D555555555555
Card info: PC/SC card in Gemalto USB SmartCard Reader 0,
protocol T=0, state OK

Choose application:
APDU request: 00A4040004F0000001
APDU response: 9000
It took 27 milliseconds.
```

```
Update Card Key:
APDU request: 80100000104D0FECBE2E69068D0CA9BABF1E54D1C5
APDU response: 9000
It took 51 milliseconds.

Send User data:
APDU request: 8050000048446176696420 4E6F76C3A16B2C6E6F76
616B32334076757462722E637A2C3630313334323234352C4132202D
20332E20706174726F2C383A30302D32323A30302C3137217265512A
APDU response: 9000
It took 871 milliseconds.
Registered new User!


Terminals: [PC/SC terminal Gemalto USB SmartCard Reader 0]
Selected Terminal: PC/SC terminal Gemalto USB SmartCard Reader 0
ATR: 3B6F00008031E06B844002020D555555555555
Card info: PC/SC card in Gemalto USB SmartCard Reader 0,
protocol T=0, state OK

Choose application:
APDU request: 00A4040004F0000001
APDU response: 9000
It took 28 milliseconds.

Update Card Key:
APDU request: 80100000104D0FECBE2E69068D0CA9BABF1E54D1C5
APDU response: 9000
It took 46 milliseconds.

Send User data:
APDU request: 8050000043416461 6D204B6172656C2C6B6172656C
4073657A6E616D2E637A2C34353134353134313 22C4132202D20332E
20706174726F2C393A30302D31323A30302C313232342A
APDU response: 9000
It took 951 milliseconds.
Updated User!
```

C Instalační manuál

C.1 Instalace knihoven Raspberry pi

zlib-1.2.11

wget https://www.zlib.net/zlib-1.2.11.tar.gz
tar xzf zlib-1.2.11.tar.gz
cd zlib-1.2.11
./configure –prefix=/usr/local/zlib-1.2.11 –static
make -j 4
make install

openssl-1.1.1l

wget https://www.openssl.org/source/openssl-1.1.1l.tar.gz
tar xzf openssl-1.1.1l.tar.gz
cd openssl-1.1.1l

./Configure threads zlib
–with-zlib-include=/usr/local/zlib-1.2.11/include
–with-zlib-lib=/usr/local/zlib-1.2.11/lib
–prefix=/usr/local/openssl-1.1.1l
–openssldir=/usr/local/openssl-1.1.1l/etc
linux-generic32
make -j 4
make install

gmp-6.2.1

sudo apt-get install m4
wget https://gmplib.org/download/gmp/gmp-6.2.1.tar.bz2
tar xjf gmp-6.2.1.tar.bz2
cd gmp-6.2.1
./configure –prefix=/usr/local/gmp-6.2.1 –enable-cxx
make -j 4
make install

PCSC

sudo apt-get install pcscd

Po zkompilování a instalaci knihoven upravte soubor 10-custom-libraries.conf

nano /etc/ld.so.conf.d/10-custom-libraries.conf

openssl-1.1.1l
/usr/local/openssl-1.1.1l/lib

gmp-6.2.1
/usr/local/gmp-6.2.1/lib

VNC Viewer

povolte v nastavení ovladač VNC:
    $ raspi-config

Trust Core

povolte v nastavení ovladač I2C:
    $ raspi-config

nainstalujte knihovnu libncurses5:
    $ sudo apt-get install libncurses5

stažení a instalace balíčků:
    $ wget https://www.multos.com/uploads/trustcore.gz
    $ tar -xf trustcore.gz
    $ sudo ./install
    $ reboot

C.2 Instalace na PC

je nutné nainstalovat programy:

XAMPP

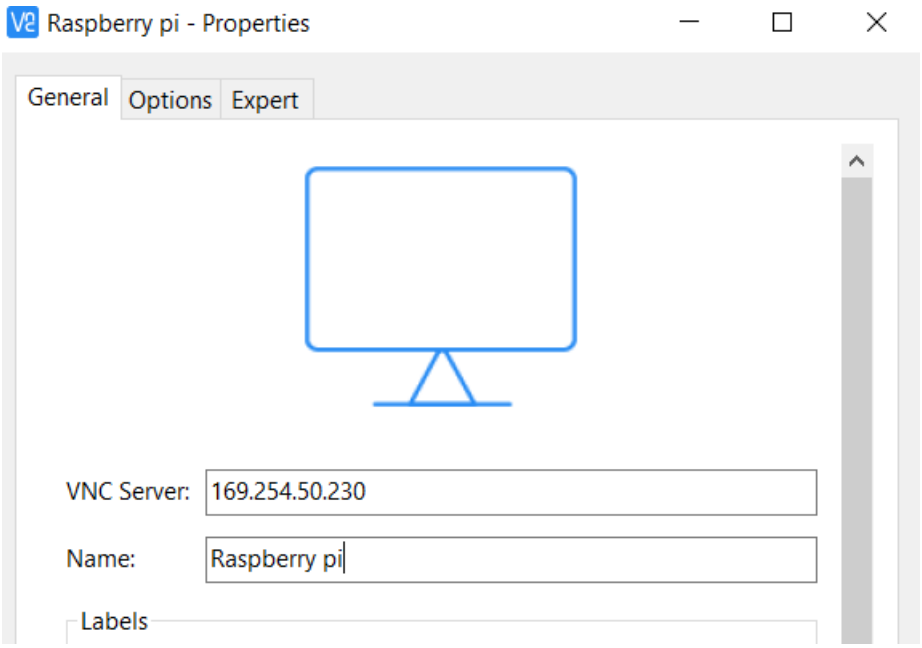
VNC Viewer

Java

D Uživatelský manuál

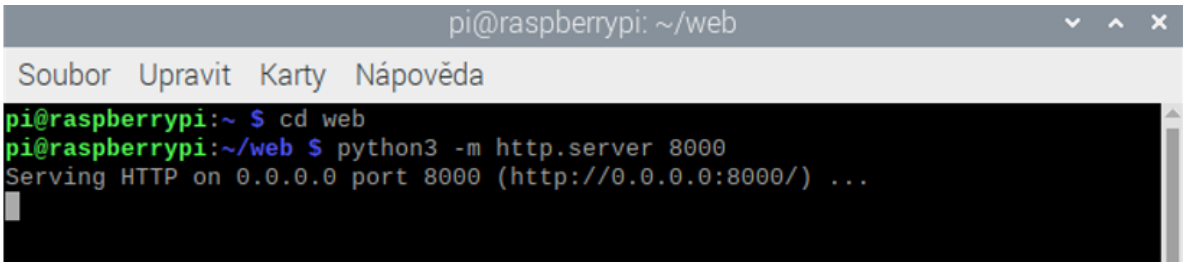
D.0.1 Přihlášení do systému

1. Připojte k Raspberry pi napájecí kabel a strčte ho do zásuvky.
2. Zapněte VNC Viewer.
   - je nutné ho mít nainstalovaný
3. Nastavte SSH spojení s Raspberry pi.
   - File -> New Connection...
   - zadejte ip adresu : 169.254.50.230 a potvrďte.



Obr. D.1: přihlášení přes VNC Viewer

   - nyní zadejte přihlašovací údaje (login: pi, heslo: raspberry237)
4. zapnutí python serveru viz obrázek D.2.
   - zapnětě příkazovou řádku v Raspberry pi a napiště příkaz: cd web
   - poté do příkazové řádky zadejte příkaz: python3 -m http.server 8000
5. spuštění ověřovacího protokolu
   - zapněte novou příkazovou řádku
   - zapnětě ověřovací protokol: ./aut_protocol -v

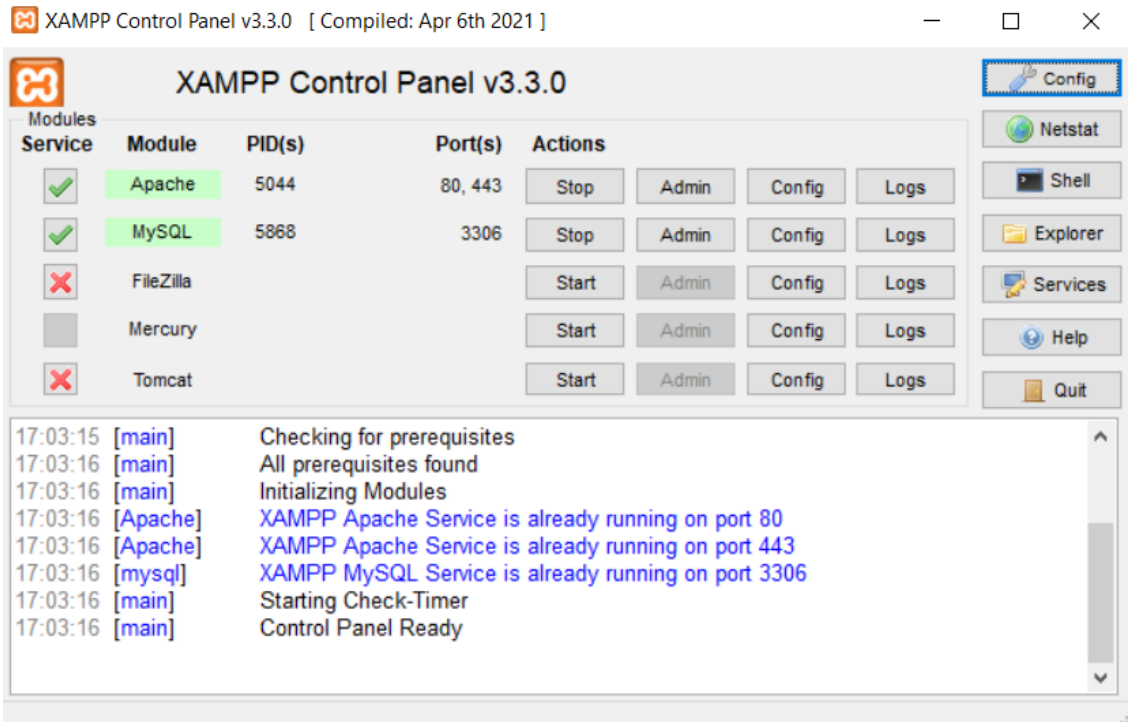


Obr. D.2: Zapnutí python serveru

D.1 Vydavatelský systém

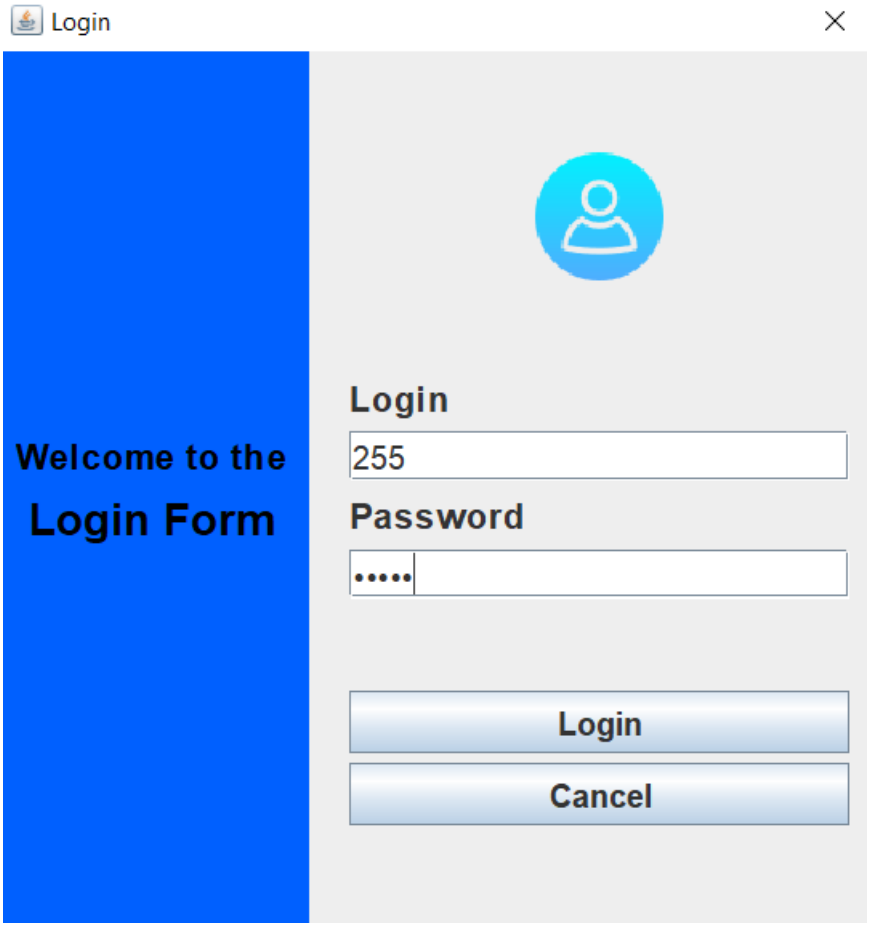
D.1.1 Přihlášení do Aplikace

1. Zapněte program XAMPP a na něm potvrďte dvě první služby (Apache a MYSQL viz. obrázek D.3



Obr. D.3: Program XAMPP

2. Zapněte vydavateslkou aplikaci, uvidíte obsah dle obrázku D.4
   - zadejte přihlašovací údaje (ID: 255, password: admin)



Obr. D.4: Vydavatelská aplikace, login

3. Pokud jste se úspěšně přihlásili, bude vám zobrazen hlavní ovládací panel.

D.1.2 Správa Vydavatelské aplikace

Registrace Trust Core

1. Připojte Trust Core na GPIO piny Raspberry pi.

2. Zapněte Raspberry pi.

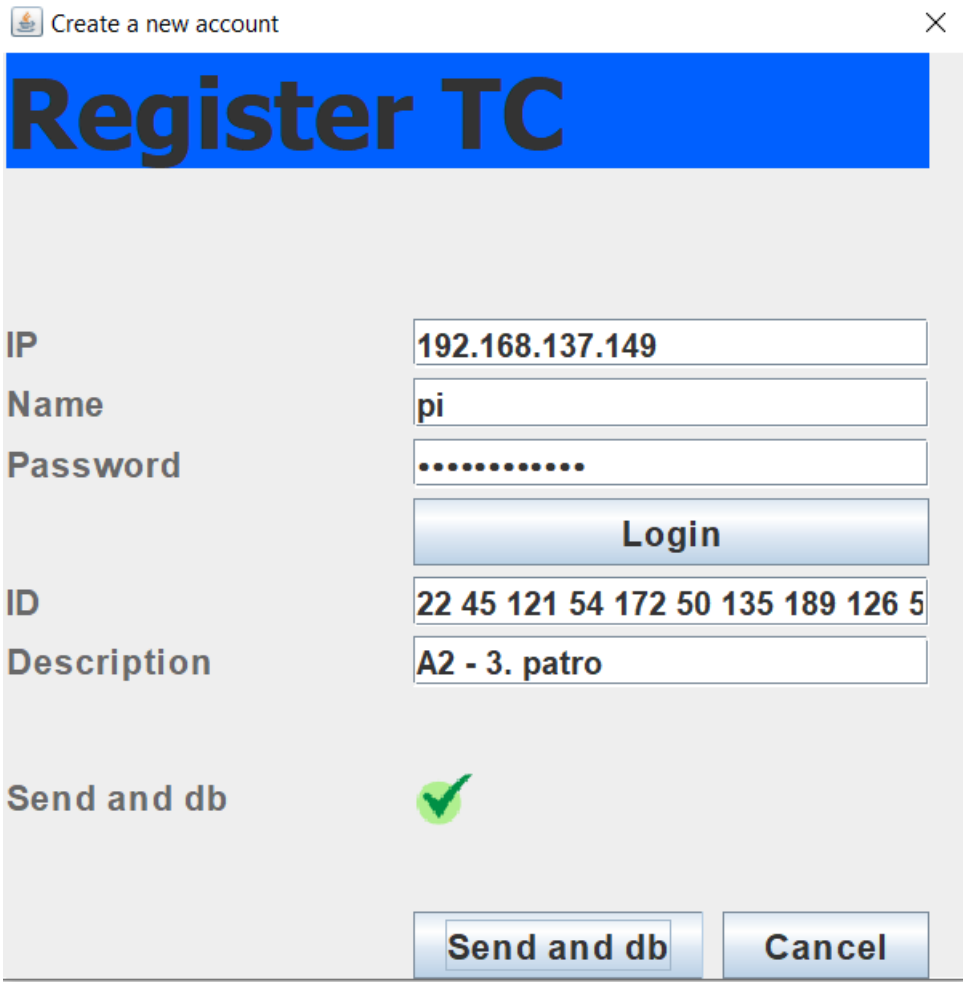
3. Stikněte na hlavním ovládacím panelu Vydavatelské aplikace tlačítko "Register TC".

4. Vyplňte tři políčko pro realizaci SSH spojení (IP, Name, Password) s raspberry pi.

5. Pokud bude spojení a provední úspěšné, předvyplní se Vám kolonka ID. Tuto kolonku nepřepisujte.

6. Vyplňte kolonku Description, dle lokace, kde se Trust Core nachází (např. A2 - 3. patro). patro).

7. Pokud Vše proběhlo v pořádku, zobrazí se Vám zelená fajfka viz. D.5.



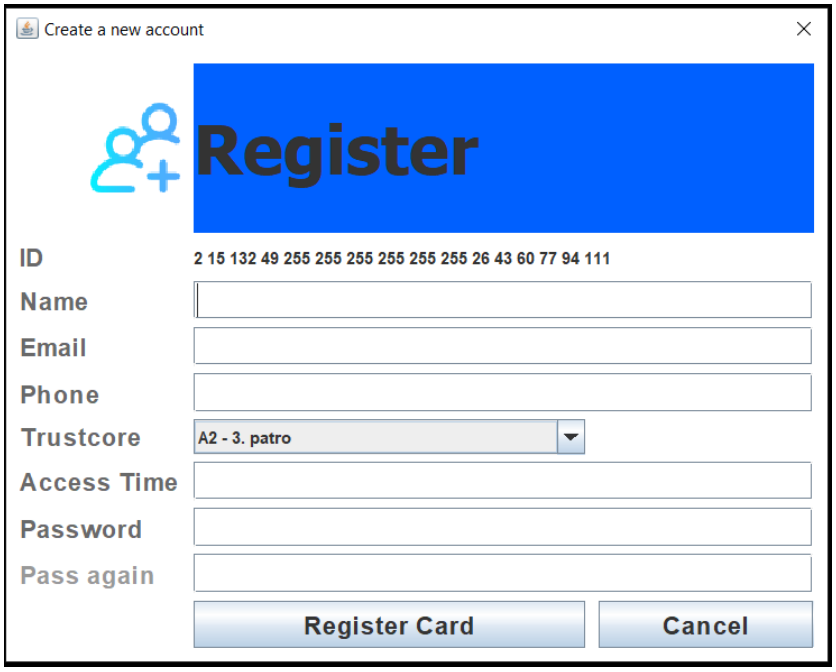
Obr. D.5: Úspěšná registrace Trust Core

Registrace karty

1. Připojte kartu společně se čtečkou do USB portu vašeho počítače.

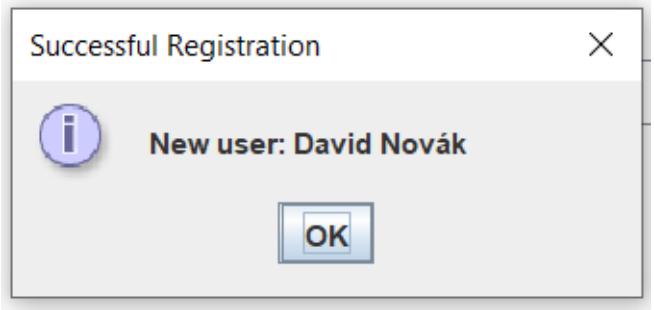
2. Stiskněte na hlavním ovládacím panelu tlačítko "Register card". Pokud vše proběhlo v pořádku, tak se Vám zobrazí nové okno s návzem "Create a new account"viz. .D.6

3. Pro registraci karty vyplňtě všechna políčka v daném okně. Pozor si dejte na



Obr. D.6: Registrace karty

vypnění času, je nutné dodržet formát "hh:mm-hh:mm"(např. 8:00-19:00). Pokud jste vše vypnil správně, tak se Vám zobrazí upozornění viz. D.7.



Obr. D.7: Úspěšná registrace

Načtení karty

1. Připojte kartu společně se čtečkou do USB portu vašeho počítače.

2. Stiskněte na hlavním ovládacím panelu tlačítko "Load card". Pokud vše proběhlo v pořádku, tak se Vám vyplní políčka na hlavním panelu. Zobrazená data jsou informace aktuálně připojené karty.

aktalizace dat karty

1. Opakujte instrukce z předchozí částí "Načtení karty".

2. Přepište libovolné políčko na hlavním ovládacím panelu.

3. Stiskněte "Update card".