Implementace bezpečného přihlašování v embedded aplikacích

Filip Miškařík

Bakalářská práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Filip Miškařík**
Osobní číslo:	**A16592**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**prezenční**
Téma práce:	**Implementace bezpečného přihlašování v embedded aplikacích**
Téma anglicky:	**An Implementation of Secure Authentication in Embedded Applications**

Zásady pro vypracování:

1. **Proveďte průzkum aktuálního stavu knihoven pro bezpečné přihlašování v embedded aplikacích, např. se systémem embedded Linux.**
2. **Analyzujte požadavky na bezpečné přihlašování uživatelů v aplikaci výdejních stojanů pohonných hmot.**
3. **Vytvořte specifikaci systému pro bezpečné přihlašování a zaznamenávání provozních údajů ve výše uvedené aplikaci.**
4. **Implementujte bezpečné přihlašování dle specifikace.**
5. **Zhodnoťte možná bezpečnostní rizika vašeho řešení a pro případné slabiny navrhněte vhodná opatření.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **SCHNEIER, Bruce. Applied cryptography: protocols, algorithms, and source code in C. 20th anniversary edition. Indianapolis, IN: Wiley, [2015]. ISBN 978-1-119-09672-6.**
2. **STAPKO, Timothy John. Practical embedded security: building secure resource-constrained systems. Boston: Elsevier/Newnes, c2008. ISBN 9780750682152.**
3. **BURNETT, Steve a Stephen PAINE. RSA Security's official guide to cryptography. New York: Osborne/McGraw-Hill, 2001. ISBN 978-0072131390.**
4. **MITCHELL, Mark, Jeffrey OLDHAM a Alex SAMUEL. Advanced Linux programming. Indianapolis, Ind.: New Riders Pub., 2001. ISBN 9780735710436.**
5. **KLEIDERMACHER, David a Mike KLEIDERMACHER. Embedded systems security: practical methods for safe and secure software and systems development. Amsterdam: Elsevier, 2012. ISBN 978-0123868862.**

Vedoucí bakalářské práce: **Ing. Tomáš Dulík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **3. prosince 2018**

Termín odevzdání bakalářské práce: **15. května 2019**

Ve Zlíně dne 7. prosince 2018

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
garant oboru

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;

  beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 15. 5. 2019 Filip Miškařík, v. r.

ABSTRAKT

Tato bakalářská práce se zabývá zabezpečením přihlašování uživatelů do systému embedded Linux. Součástí jsou naprogramované aplikace pro vytvoření a ověřování hardwarového klíče pro přístup uživatele do chráněné části systému výdejního stojanu čerpací stanice. Bezpečnost celého systému je založena na použití čipové karty ("smart card"), která je součástí hardwarového klíče. Závěrečná kapitola zhodnocuje bezpečnostní rizika zvoleného řešení.

Klíčová slova: hardwarový klíč, čipová karta, autorizace

ABSTRACT

This bachelor thesis deals with the secure authentication of users of an embedded Linux system. It contains a programmed application for the initialization of a hardware key. This hardware key is used for the authentication and authorization of an user who wants to access a protected part of the embedded Linux system. Furthermore, the bachelor thesis includes the implementation of the verification program of a fuel dispenser machine. The security of whole system is based on using smart card, which is in the hardware key. At the end of the bachelor thesis, the evaluation of the security risks od the chosen solution are described.

Keywords: hardware key, Smart card, authorization

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Tímto chci poděkovat svému vedoucímu práce Ing. Tomáši Dulíkovi, Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce. Dále chci poděkovat Ing. Tomáši Juřenovi za pomoc při programování. Také chci poděkovat své přítelkyni Bc. Evě Pitrunové za pomoc při vypracování této práce a rodičům za podporu ve studiu.

OBSAH

ÚVOD

Tématem bakalářské práce je implementace bezpečného přihlašování v embedded aplikacích. Konkrétně se jedná o přihlašování servisního technika do systému výdejního stojanu pohonných hmot. Jedním z důvodů výběru tohoto tématu byla účast FAI UTB ve Zlíně na řešení grantového projektu OPPIK Aplikace v konsorciu s firmou Adast Systems, a.s., která se primárně zabývá výrobou těchto stojanů. V projektu jsem se účastnil procesu vývoje software nového typu stojanu, který by měl zvýšit konkurenceschopnost firmy Adast Systems. Zabýval jsem se zejména implementací zabezpečení přihlašování pomocí karet Smart card.

Moje práce začala průzkumem aktuálního stavu technologií pro bezpečné přihlašování v embedded aplikacích. Následně jsem analyzoval požadavky na celé řešení a poté navrhl a implementoval systém bezpečného přihlašování.

První kapitola práce se věnuje funkčním a nefunkčním požadavkům na zabezpečení přihlášení, které byly definovány firmou Adast Systems. Dále je popsáno samotné zařízení – hardwarový klíč, použitý pro přihlášení do systému. Podkapitola nefunkčních požadavků popisuje požadavky na softwarovou platformu, která zahrnuje jazyk C++, knihovnu Qt a embedded Linux.

Druhá kapitola práce popisuje použité kryptografické metody. Jednotlivé podkapitoly obecně popisují kryptografické algoritmy, které byly či by mohly být implementovány v zařízení. Patří zde např. RSA, které je použito pro vytvoření elektronického podpisu sériového čísla a datumu vydání. Dále zde řadíme i DES/3DES, který bude popsán pro případ využití chráněného přenosu dat.

Další kapitola obsahuje popis současného trhu se zařízeními, které slouží k bezpečnému přihlašování. Jsou zde charakterizovány čipové karty a jejich čtečky. Následovat bude výčet standardů, které určují vlastnosti těchto zařízení.

Čtvrtá kapitola je zaměřena na praktickou část této práce. Je zde charakterizována struktura programů – inicializačního a ověřovacího. Inicializační program slouží pro inicializaci karty ve firmě Adast Systems, zatímco ověřovací program běží přímo ve výdejním stojanu, kde ověřuje platnost karty a zabraňuje neoprávněnému vniknutí.

V páté kapitole je popsán průběh programu při inicializaci a ověřování platnosti karty. V kapitole budou nastíněny i úryvky kódu. Tato kapitola bude zahrnovat i konkrétní návod k použití inicializační aplikace, který bude v budoucnu firmou využíván.

Poslední kapitola obsahuje zhodnocení bezpečnostních rizik, které mohou vést k napadení systému. Toto zhodnocení může pomoci firmě Adast Systems při dalším vývoji jejich produktů.

I. TEORETICKÁ ČÁST

1 FUNKČNÍ A NEFUNKČNÍ POŽADAVKY NA ZABEZPEČENÍ PŘIHLAŠOVÁNÍ

Tato práce vzniká za účelem vývoje výdejního stojanu E-Line pro firmu Adast Systems a.s. Požadavky na řešení přihlašování tudíž byly konzultovány s firmou Adast Systems a jsou obsaženy v této kapitole.

1.1 Funkční požadavky

Funkčními požadavky na přihlašování servisního technika do výdejního stojanu čerpací stanice se rozumí všechny požadavky, které ovlivňují funkčnost tohoto zařízení a jak se bude zařízení chovat.

1.1.1 Tvar flashky

Hlavním požadavkem firmy bylo, aby přihlašovací zařízení bylo ve tvaru USB donglu, jelikož tento typ přihlašovacího zařízení bylo specifikováno při vytváření podnikatelského záměru. Proto byl vybrán USB dongle, který je popsán ve třetí kapitole této práce.

1.1.2 Přihlašování bez pinu

Firmou bylo specifikováno, že po vložení USB donglu není nutné ověřit uživatele pomocí pinu, avšak tyto USB dongly se nesmí nekontrolovatelně množit v případě ztráty či krádeže. USB dongle bude vydávat pouze firma Adast proškoleným servisním technikům.

1.1.3 Časově omezená platnost

USB dongle po vydání firmou Adast Systems musí být funkční po dobu jednoho roku. Poté musí servisní technik přijet opět do firmy Adast Systems, kde dostane nové školení k servisování výdejních stojanů a obdrží nový USB dongle, který bude mít platnost na další rok.

1.1.4 Vyčítání informací ze stojanu

USB dongle bude mít možnost vyčítat a ukládat do vlastní paměti informace z totalizérů. Totalizér je komponenta výdejního stojanu, která nese informaci o celkovém počtu vydaných litrů paliva. Jelikož je těchto totalizérů na stojanu více a paměť samotné čipové karty, která je popsána ve třetí kapitole této práce, by nestačila, zvolil jsem USB dongle, který má slot pro paměťovou SD kartu s dostatečnou kapacitou (až 8GB).

1.2 Nefunkční požadavky

1.2.1 Programovací jazyk C++

Programovací jazyk C++ byl zvolen z toho důvodu, že systém, který je implementován ve výdejním stojanu je programován v jazyce C++, a tudíž použití stejného jazyka přináší ulehčení práce při integraci přihlašovací aplikace od systému stojanu.

Programovací jazyk C++ vychází z programovacího jazyka C, vylepšení přišlo s tím, že nový jazyk C++ podporuje třídy. Proto byl v roce 1979, kdy vznikla první oficiální verze, nazván „C with classes“. Později v roce 1984 byl přejmenován na C++. Komerční uvedení na trh přišlo o rok později, tj. v roce 1985. Jazyk se postupně vyvíjel a v roce 1998 byl standardizován jako ISO C++. Jazyk, jak jej známý dnes, je dostupný od roku 2011, kdy vyšla poslední oficiální aktualizace standardu na ISO C++11. [1]

Když byl původně navržen jazyk C++, byly všechny identifikátory dostupné ve standardní knihovně C++ bez předpony std:: (včetně std::cin a std::cout). To však znamenalo, že jakýkoli identifikátor ve standardní knihovně mohl být v rozporu s identifikátorem, který si zvolil programátor. Dále by vznikl problém při kompilaci kódu napsaném ve starší verzi C++ v nové verzí překladače, protože nově zavedené identifikátory by mohly mít stejný název se starými. Proto byly identifikátory přesunuty do speciálních prostor nazvaných jmenné prostory (namespace). Autor využil ve své práci jmenný prostor std, který zejména slouží pro snazší vypisování informací na konzoli. [2]

Autor dále použil knihovnu iostream.h, která též souvisí s vypisováním na konzoli. Knihovna iostream je objektově orientovaná knihovna, která poskytuje vstupní a výstupní funkce.[3]

1.2.2 Embedded Linux

Operační systém Linux se používá jak v desktopových počítačích, tak v embedded zařízeních (vestavěné zařízení). Ve vestavěných zařízeních se používá jako operační systém v reálném čase. [4] Operační systém v reálném čase, známý též pod zkratkou RTOS (Real Time Operating System), je systémová komponenta, která rychle přepíná mezi úkoly, což vyvolává dojem, že je vykonáváno více úkolů najednou. Ve skutečnosti může jedno jádro procesoru vykonávat pouze jeden úkol v jednom okamžiku. [5] Vestavěné zařízení se od desktopových řešení liší v mnoha oblastech. Vestavěné systémy mají omezenou paměť, pevný disk zde většinou není, mají malý displej atd. Výhodou vestavěných systémů je jejich malá velikost čili kompaktnost, menší spotřeba elektrické energie a nižší pořizovací cena. [4]

Obrázek 1 – Raspberry Pi [23]

Typickým příkladem systému s embedded Linuxem jsou desky Raspberry Pi, viz obrázek č. 1. Jádro ve vestavěném systému běží často v reálném čase. Linuxové jádro běžící na stolním počítači ale nemá schopnost reagovat na události v reálném čase, tzn. reakce na události mohou trvat nedeterministicky dlouho. Běh v reálném čase však přináší omezení.

Linuxové jádro v embedded zařízení s jedno-jádrovým CPU je schopno provádět pouze jednu funkci v jeden čas. Což převedeno do praxe znamená, že například při běžícím VideoStreameru provádí funkci konverze videa do daného formátu a odesílá jej do sítě. Oproti tomu, desktopová verze linuxového jádra dokáže zpracovávat více úloh v jeden okamžik. Opět převedeno do praxe to znamená, že může běžet internetový prohlížeč, video, program Java i program C v jednom okamžiku. [4] Z výše uvedeného je zřejmé, že embedded Linux může dělat jen specifický úkol. Od toho se též odvíjí modul s komponentami je též omezen na požadavek dané aplikace vestavěného systému.

2 KRYPTOGRAFIE

Pro zabezpečení přístupu do výdejního stojanu autor použil šifrovací algoritmus RSA pro vytvoření digitálního podpisu. Konkrétní použití je popsáno v praktické části práce v kapitole 5.1.

2.1 RSA

RSA je šifrovací algoritmus, jenž byl vytvořen a publikován v roce 1978 Ronaldem Rivestem, Adim Shamirem a Leonardem Adlemanem. Zkratka RSA vychází z počátečních písmen příjmení zakladatelů. Je to asymetrická šifra, která je založena na Eulerově větě. Používá se pro šifrování a podepisování dokumentů. [6]

RSA je ze všech algoritmů, které mají veřejný klíč, nejjednodušší na pochopení a implementaci. Bezpečnost RSA algoritmu vychází z obtížnosti faktorizace velkých čísel. Používá se dvojice klíčů veřejného a soukromého klíče. Jedná se o čísla se 100-200 číslicemi (někdy i delšími), která jsou prvočísla. [7]

2.1.1 Princip asymetrické kryptografie

Symetrické šifry, jejichž příkladem můžou být Caesarova šifra nebo exponenciální šifra, mají zprávu šifrovanou pomocí jednoho klíče. Dešifrování probíhá opačným způsobem. U asymetrických šifer se používají dva klíče. Jeden soukromý, který si majitel důkladně uschová, a jeden veřejný, který je možné svobodně oznámit celému světu. Přitom též platí, že co je zašifrováno soukromým klíčem, půjde rozšifrovat pouze správným veřejným klíčem z klíčového páru. Totéž platí v opačném případě, kdy je zpráva zašifrována veřejným klíčem, lze ji správně rozšifrovat jen soukromým klíčem z klíčového páru. [6]

Pomocí algoritmu RSA je možné zprávu šifrovat nebo vytvořit pro zprávu digitální podpis. Při šifrování se použije pro zašifrování zprávy veřejný klíč. Zašifrovaná zpráva je odeslána příjemci, který si ji rozšifruje pomocí soukromého klíče. Útočník zprávu není schopen rozšifrovat, jelikož je zpráva rozluštitelná pouze při dešifrování soukromým klíčem, který má pouze příjemce. [6]

Druhé využití asymetrické šifry spočívá v elektronickém podpisu. To se využívá v případech, kdy je potřeba zajistit, aby dokument nemohl nikdo měnit a zároveň je potřeba zajistit, aby si uživatel mohl ověřit, to že s dokumentem nebylo manipulováno a jeho vydavatele. Pro zprávu se vytvoří tzv. hash (otisk zprávy), ten se zašifruje pomocí soukromého klíče a výsledek se přiloží ke zprávě, zpráva samotná však není nijak šifrována. Zprávu a její hash pak je možné odeslat příjemci. Ověření probíhá tak, že se hash zprávy dešifruje pomocí veřejného klíče a výsledek se porovná s originální zprávou. Jestliže jsou zprávy totožné, je jisté, že s podpisem nikdo nemanipuloval. [6]

2.1.2 Generování klíčů

Na začátku generování páru veřejného a soukromého klíče se vygenerují dvě, pro maximální bezpečnost stejně dlouhá, náhodná čísla. Vypočítá se součin *n* těchto dvou náhodných čísel:

$$n = p \cdot q$$

Poté se náhodně vybere šifrovací klíč *e* tak, že platí, že *e* a $(p-1)\cdot(q-1)$ jsou relativní prvočísla (jejich společný největší dělitel existuje jen jeden). Nakonec se použije rozšířený euklidovský algoritmus pro výpočet dešifrovacího klíče:

$$d = e^{-1} \cdot mod((p-1)\cdot(q-1))$$

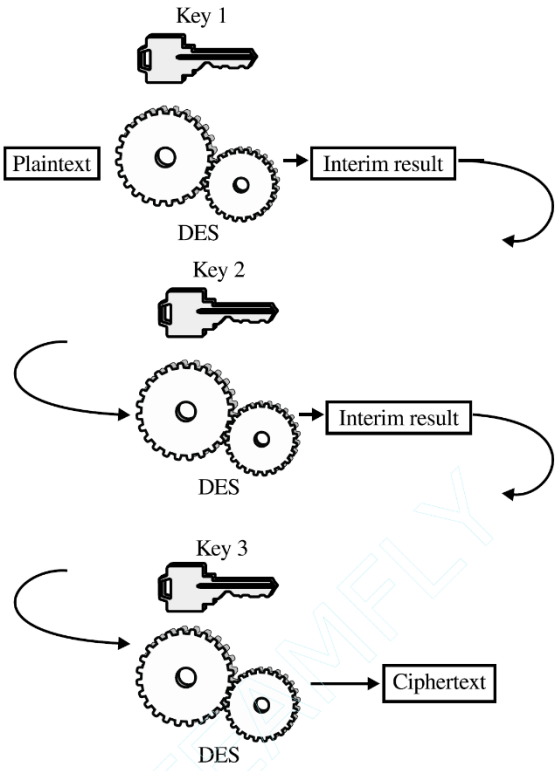
Číslo *e* je veřejným klíčem a číslo *d* je soukromým klíčem. Čísla *p* a *q* by měla zůstat utajena, pokud možno ztracena, aby nedošlo nechtěnému množení klíčů. [7]

2.2 DES/3DES

Šifrovací algoritmus DES je bloková šifra s 56-bitovým klíčem pro vytvoření tabulky klíčů. Šifrování probíhá tak, že pomocí klíčové tabulky provádí DES operace s prostým textem. Dešifrování je jednoduše opačný postup k šifrování. DES byl vyvinut v 70. letech minulého století výzkumníky ve firme IBM. Po zavedení se stal DES volně dostupným a hojně šířeným šifrovacím algoritmem. Z počátku byla délka klíče dostačující a byl mnohými kryptografy označen za bezpečný, dokonce tím, že nemá žádné slabiny. Jelikož hodnota klíče mohla dosáhnout až 15-místného čísla, byl tento klíč pro počítače 80. let neprolomitelný, což by trvalo při útoku hrubou silou několik let. [8]

V 90. letech však počítače začaly mít vyšší výpočetní výkon a prolomení 56-bitového klíče již bylo možné provést v rozumném čase. V roce 1999 byl DES prolomen za méně než 24 hodin. [8]

Jednou z široce používaných náhrada DES je 3DES (TripleDES). Již z názvu vyplývá, že se jedná o trojí provedení algoritmu DES. Používají se tři různé 56-bitové klíče. Prolomení třech různých klíčů je komplikované. I kdyby se podařilo prolomení prvního klíče, není to možné zjistit do doby kdy jsou prolomeny další dva klíče. Přitom uvažujme, že je délka klíče v řádech kvadrilionů (15-ti místné číslo). [8]



Obrázek 2 – TripleDES šifrování [9]

Obrázek č. 2 znázorňuje, jak probíhá šifrování pomocí šifrovacího algoritmu TripleDES. Stejný postup je využit při dešifrování, takže pokud by byl jeden z klíčů nesprávný, původní text nebude zřetelný.

TripleDES však s sebou nese dva základní problémy. První vyplývá z toho, že kryptoanalytici zefektivnili útok brutální silou. Zatím tento problém není aktuální, protože stále není výpočetní výkon počítačů natolik veliký, aby byl algoritmus prolomen v rozumném čase. Druhým problémem je rychlost šifrování. Již šifrování pomocí DES trvalo poměrně dlouhý čas, tripleDES trvá třikrát déle. Z tohoto důvodu nelze tripleDES aplikovat všude, například tam, kde je nutná vysokorychlostní propustnost dat. [8]

2.3 Open SSL

OpenSSL je univerzální kryptografická knihovna. OpenSSL je licencován pod licencí ve stylu Apache (licence požaduje zachování autorství), což znamená, že jej můžete zdarma získat a používat pro komerční a nekomerční účely v souladu s některými jednoduchými licenčními podmínkami. Autor knihovnu využil pro šifrování pomocí RSA algoritmu a 3DES algoritmu.

3 SOUČASNÝ STAV TECHNOLOGIÍ NA TRHU

Současný trh technologií pro přihlašování nabízí řešení pomocí Smart karet. Existují dva základní druhy čipových karet: kontaktní karty a bezkontaktní (NFC nebo RFID) karty. Každá karta komunikuje s cílovým zařízením pomocí čtečky. Na trhu jsou čtečky pro kontaktní i bezkontaktní karty. Existují čtečky se slotem pro SD kartu nebo přímo se zabudovanou Flash pamětí.

Čipová karta je vyrobená z plastu a má v sobě zabudovaný čip a paměť pro zpracování a uložení dat. Data jsou většinou spojena s hodnotou, informacemi či obojím a jsou uložena a zpracovávána v čipu karty. První čipové karty představené v Evropě téměř před třemi dekádami, sloužily jako nástroj pro uložení hodnoty kreditu pro telefonní automaty. Jak pokročil vývoj čipových karet, začaly se používat jako platební karty. [10] Dnes se využívají v mnoha odvětvích jako je zdravotnictví, doprava, zábava a jak již bylo výše zmíněno, tak hlavně v bankovnictví. Každoročně počet karet na světě roste o jednotky procent. V roce 2017 bylo v oběhu přes 9,9 mld. kusů smart karet a v roce to bylo již přes 10,2 mld. kusů. Z toho vyplývá, že meziroční nárůst pro roky 2017 a 2018 činil 2,61%. Významný je též přechod z kontaktních karet na bezkontaktní, kde meziroční růst v období 2017-2018 činil 13,14%. [11]

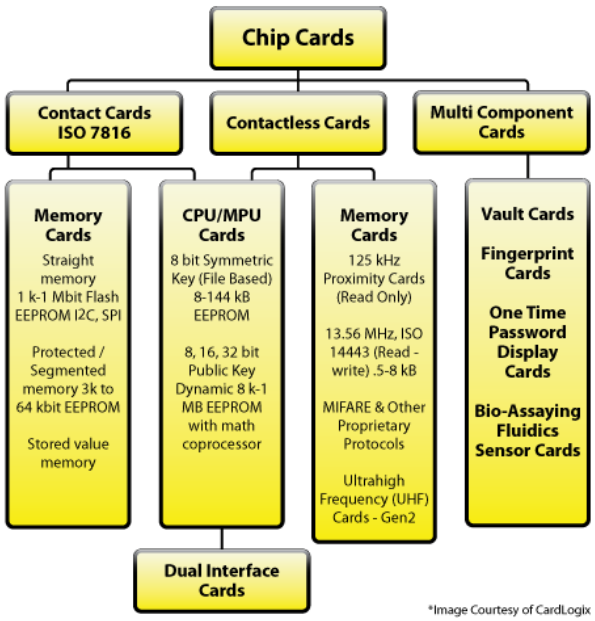
Obrázek 3 – Vrstvy karty [14]

Většina dnešních čipových karet je vyrobena z PVC, polyesteru či polykarbonátu. Tyto různé materiály jsou po vrstvách nanášeny, čímž kartě dodávají specifickou životnost a funkčnost. Vrstvy (viz obrázek č. 3) se nejprve potisknou a následně se laminují ve velkém lisu. Následuje vysekávání a vložení čipu. Sestavování karty může mít až 30 kroků a může být použito až 12 různých položek. Ve výsledku se karta uživateli však jeví jako jednoduché zařízení. [11]

3.1 Dělení čipových karet

Smart karty se rozdělují podle toho, jak jsou na kartu data ukládána a jak jsou zpracovávána a podle toho, jak je do karty implementován čip. Při návrhu systému je k dispozici celá řada možnosti, jak to vyjadřuje obrázek č. 4.

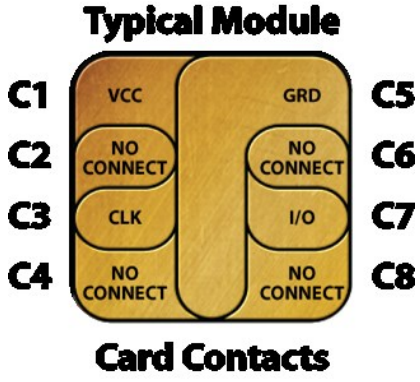
Čipové karty se dělí na tři hlavní kategorie: kontaktní karty, bezkontaktní karty a multikomponentové karty. [12]



Obrázek 4 – Dělení čipových karet [14]

3.1.1 Kontaktní karty

Kontaktní karty jsou nejběžnějším typem čipové karty. Elektrické kontakty jsou umístěny na vnější straně karty, přes které se karta připojí ke čtečce, ve které je vložena. Konektor je připojen k zapouzdřenému čipu v kartě. Na obrázku č. 5 jsou znázorněny a popsány jednotlivé kontakty. Se zvýšením úrovně výkonu zpracování, flexibility a pamětí se zvyšují náklady. Proto je karta s jednou funkcí obvykle nákladově efektivnějším řešením. Je důležité správně vybrat typ čipové karty pro dané řešení. Je potřeba určit požadované úrovně zabezpečení a vyhodnotit funkčnost ve vztahu k nákladům. [12]



Obrázek 5 – Konektor kontaktní čipové karty [16]

3.1.2 Bezkontaktní karty

Jedná se o karty, které využívají rádiové frekvence mezi kartou a čtečkou pro přenos dat. K přenosu dat tedy dochází bez fyzického vložení karty do čtečky. Bezkontaktní karty byly poprvé využity v dopravě pro rychlou aktualizaci hodnoty jízdného, kde nižší zabezpečení nebylo problémem. Tyto karty fungují s chráněnými typy pamětí a komunikují na frekvenci 13,56 MHz. [12]

3.1.3 Duální karty

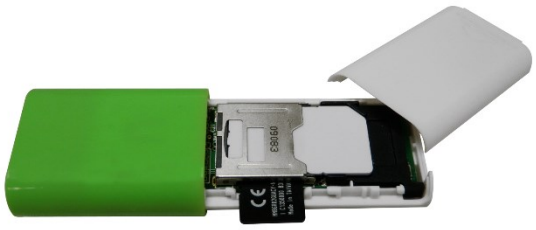
Tento typ karet využívá z předchozích dvou typů obě vlastnosti, tzn. že na kartě se nachází kontakty a zároveň lze pro komunikaci využívat rádiové frekvence. Tyto karty našly využití v bankovnictví. [12]

3.2 Čtečky

Čtečky a terminály pracují s čipovými kartami, aby získaly informace o kartách a provedly transakci. Čtečky i terminály čtou a zapisují na čipové karty. Čtečky komunikují s PC pro většinu požadavku na zpracování. Terminály se, na rozdíl od čteček, podobají samostatnému PC. Konektivita terminálů je obvykle přes TCP/IP nebo GSM síť. Komunikace karty s čtečkou pomocí kontaktů je považována za bezpečnější řečení než u bezkontaktního řešení. Kontaktní řešení má i větší rychlost přenosu dat mezi kartou a čtečkou. Pro řešení přihlašování ve výdejním stojanu bylo zvoleno řešení pomocí kontaktní čtečky ASC ACR 101l.

3.2.1 ACS ACR 101l

Čtečka ACS ACR 101l (viz obrázek č. 6) kombinuje čtečku čipových karet se slotem pro SD kartu. Čtečka vyhovuje standardu ISO 7816. Čtečka je vhodná pro práci s kartami o velikosti klasické SIM karty. Čtečka je dokonce vybavena čipem pro čtení i bezkontaktních karet. Dále karta vyhovuje standardu PC/SC, kterého jsem využil a s kartou komunikuji pomocí knihovny pcsclite.h. Čtečka podporuje nejrozšířenější operační systémy jako jsou Windows, Linux, MAC OS a Android. [17]



Obrázek 6 – Čtečka ACS ACR101l [18]

3.3 Standardy

Standardy čipových karet specifikují jejich fyzikální vlastnosti, komunikační vlastnosti a identifikátory aplikací vloženého čipu a dat. Téměř všechny standardy odkazují na ISO 7816 a ISO 14443. Prodej kopií těchto norem zajišťuje American National Standards Institute. [19]

Vlastnosti specifikací jsou diskutovány mnoha velkými organizacemi a skupinami, které navrhují své standardy. Interoperabilita karty by měla platit na několika úrovních. Zaprvé na kartu samotnou, zadruhé pro přístupové terminály karty, zatřetí pro sítě a za čtvrté pro vlastní systémy vydavatelů karet. [19]

3.3.1 ISO 7816

ISO 7816 je mezinárodní standard jak pro karty s integrovanými obvody, které využívají elektrické kontakty na kartě, tak i pro karty, které komunikují se čtečkami pomocí radiových frekvencí. Samotný standard ISO 7816 má celkem 14 částí. Části 1, 2 a 3 se zabývají pouze kontaktními čipovými kartami. Definují různé aspekty karty jako jsou její fyzické rozměry, elektrické rozhraní a komunikační protokoly. Části 4, 5, 6, 7, 8, 9, 11, 13 a 15 řeší všechny typy čipových karet (kontaktní i bezkontaktní). Definují logickou strukturu karty (soubory a datové prvky), příkazy používané aplikačním programovacím rozhraním pro základní použití, správu aplikací, biometrické ověřování, kryptografické služby a pojmenování aplikací. Část 10 je věnována paměťovým kartám, které se využívají jako telefonní karty nebo jako karty pro prodejní automaty. Část 7 definuje zabezpečený přístup do relační databáze založené na rozhraní SQL. [19]

ISO 7816 článek 4 definuje Application Protocol Data Unit (APDU), čímž se rozumí komunikační protokol mezi čtečkou karet a čipovou kartou. APDU se dělí na dvě kategorie: příkaz a odpověď. Každý APDU příkaz má 5 bytovou hlavičku této struktury: CLA, INS, P1, P2 a P3. Vzhledem k tomu, že se jedná o bytovou hlavičku, znamená to, že jednotlivé části hlavičky můžou nabývat hodnot 0-255 bytů. Význam jednotlivých částí hlavičky popisuji podrobněji dále v textu. [19]

3.3.2 ISO 14443

ISO 14443 je mezinárodní norma definující rozhraní pro bezkontaktní karty, včetně vysokofrekvenčního rozhraní, elektrického rozhraní a komunikačních a protikolizních protokolů. Karty, které vyhovují této normě, fungují na frekvenci 13,56 MHz a mají provozní dosah až 10 cm. Karty dle normy ISO 14443 se využívají v aplikacích v oblastech dopravy, finanční a v řízení přístupu. Využívají se též elektronických pasech. [19]

3.3.3 PC/SC

PC/SC je globálně implementovaný standard pro rozhraní počítačů s čipovými kartami, který je dostupný na většině operačních systémů, včetně Windows a Linux. Při vývoji aplikace využívající tohoto rozhraní jsem využil operační systém Ubuntu Linux ve verzi 18.04 Bionic Beaver a knihovnu PCSClite. [19]

3.3.4 GlobalPlatform

GlobalPlatform je mezinárodní nezisková organizace, jejíž posláním je zavádět, udržovat a řídit přijetí standardů, které přinesou otevřenou infrastrukturu pro čipové karty, zařízení a systémy. Standard GlobalPlatform byl přijat prakticky všemi bankami po celém světě pro načítání kryptografických dat založených na technologii JavaCard. Standard zavádí mechanismy a zásady, které umožňují bezpečnou komunikaci s pověřeními. [19]

Java Card je průmyslová platforma, vyvinutá společností Sun Microsystems (nyní Oracle), umožňující aplikacím na bázi Java appletů běžet na čipových kartách podporující tento standard. JavaCard pomáhá vývojářům rychle a efektivně vytvářet, testovat a nasazovat aplikace založené na čipových kartách. [19]

3.4 ACOS 3

Obrázek 7 – Čipová karta ACOS3[20]

Pro řešení zabezpečeného přístupu do výdejního stojanu byla vybrána kontaktní karta ACOS 3 od firmy ACS (viz obrázek č. 7). Sám výrobce tuto kartu doporučuje pro využití v aplikacích řízení přístupu. Karta ACOS 3 je nabízena ve dvou paměťových verzích. Velikosti paměti EEPROM jsou 32 KB a 72 KB. Autor pracoval s verzí 32 KB. Karta vyhovuje standardu ISO 7816 a to v částech 1 až 3. Karta podporuje šifrování DES/3DES a MAC. [21]

3.4.1 Struktura souborového systému karty

Karta ACOS3 má svůj operační systém. Tento operační systém má souborový systém, který obsahuje různé soubory. Tyto soubory jsou čteny či inicializovány během personalizace karty.

Soubory na kartě jsou dvou typů. První typ je typ záznam (record) a druhý typ je soubor (file). Rozdíl mezi nimi je takový, že záznam může mít maximální délku 255 bytů. Proto se využívá hlavně pro vnitřní soubory karty, které jsou výhradně typu záznam. Druhý typ má neomezenou délku (omezena velikostí karty). K souboru se přistupuje pomocí offsetu. Tento typ mohou mít uživatelské soubory. [22]

Tabulka 1 – Vnitřní soubory karty [22]

		File Security Attributes			Record Organization
Memory Area	**Internal File ID**	**Manufacturing Stage**	**Personalization Stage**	**User Stage**	
MCU-ID File	FF 00_H	R: FREE W: NO ACCESS	R: FREE W: NO ACCESS	R: FREE W: NO ACCESS	2 x 8 bytes
Manufacturer File	FF 01_H	R: FREE W: IC	R: FREE W: NO ACCESS	R: FREE W: NO ACCESS	2 x 8 bytes
Personalization File	FF 02_H	R: FREE W: IC	R: FREE W: IC	R: FREE W: NO ACCESS	3 x 4 bytes
Security File	FF 03_H	R: IC W: IC	R: IC W: IC	R: NO ACCESS W: IC	12 x 8 bytes
User File Management File	FF 04_H	R: FREE W: IC	R: FREE W: IC	R: FREE W: IC	N_OF_FILE x 7 bytes
Account File	FF 05_H	R: FREE W: IC	R: FREE W: IC	R: IC W: IC	8 x 4 bytes
Account Security File	FF 06_H	R: FREE W: IC	R: FREE W: IC	R: NO ACCESS W: IC	4 x 8 bytes
ATR File	FF 07_H	R: FREE W: IC	R: FREE W: IC	R: FREE W: NO ACCESS	1 X 36 bytes
User File Data Area	File IDs: xx yy_H xx ≠ FF_H	According to the file definitions			

Strukturu souborů, které se v souborovém systému karty nacházejí, ukazuje tabulka č. 1. Každý soubor má svoje ID, které má formát xx yy. Všechny vnitřní soubory karty mají ID začínající FF, následující byte „yy“ slouží pro jednoznačné rozlišení. Uživatelské soubory jsou označeny libovolnými dvěma byty, kromě hodnoty prvního bytu FF. Vnitřní soubory karty jsou rozděleny do několika záznamů nazvané „Records“. [22]

Velikosti souborů a počet záznamů v souborech je zřetelný ve sloupci „Record Organization“ tabulky č. 1. Tabulka též ukazuje možnosti přístupů k jednotlivým souborům. „R“ znamená čtení a „W“ zápis. Lze rozlišit tři typy přístupů: FREE (volný přístup), NO ACCESS (přístup odepřen) a IC (přístup možný po zadání Issuer klíče). Issuer klíč je v kartě a má výchozí hodnotu nastavenou od výrobce. Přístupy k jednotlivým souborům se během jednotlivých stavů karty mění, což vyplývá z tabulky č. 1. [22]

Prvním souborem na kartě je MCU ID, jehož ID je FF 00. Tento soubor je rozdělen do dvou záznamů, kde první záznam nese informace o unikátním osmi bytovém sériovém čísle. Druhý záznam označuje revizní číslo karty, je též v délce 8 bytů. Tento soubor lze číst, nikoliv však upravovat. [22]

Dalším souborem „Manufacturer File“ s ID FF 01, který se skládá ze dvou osmi bytových záznamů. Úpravy probíhají pouze v prvním bytu prvního záznamu. Tento byte je rozdělen do osmi bitů, přičemž osmý bit charakterizuje „Manufacturer Fuse“, což znamená, že pokud je v tomto bitu zapsaná jednička, je ukončen první stav. Tato změna se projeví až po odpojení nebo resetování připojení karty od počítače, skrz který je karta programována. Šestý bit charakterizuje přepínač, který ovládá, jestli je na kartě povolen mód účtu. Pátý bit charakterizuje mód počítání záznamů. Pokud je zapsána jednička, je povolen mód počítání od jedničky, v opačném případě se počítá od nuly. [22]

Personalization file, který následuje s ID FF 03, má velikost 12 bytů rozdělenou do třech záznamů po 4 bytech. Tento soubor je upravován během Personalization stage. V tomto souboru je uložen Personalization bit, který slouží pro ukončení Personalization stage. Personalization bit je uložen ve čtvrtek bytu prvního záznamu tohoto souboru. Tato změna se projeví až po resetování připojení karty nebo po odpojení karty od počítače, skrz který je programována. V prvních byte prvního záznamu se nazývá „Option register“. Osm bitů tohoto záznamu slouží pro definování vlastností karty, mezi které patří povolení módu účtu na kartě, povolení šifrování 3DES apod. Druhý záznam ve svých osmi bitech definuje, které základní klíče budou používány pro přístup k souborů, proto je taky nazván Security register. Třetím záznamem je uložena informace o počtu uživatelských souborů na kartě. [22]

V souboru s názvem Security file jsou uloženy klíče, kterými se přistupuje k souborům, v našem případě Issuer klíč, dále je zde uložen klíčový pár pro autentizaci karty vůči výdejnímu stojanu, náhodné číslo, které se používá ke stejnému účelu a v neposlední řadě jsou zde uloženy čítače neúspěšných pokusů o zadání nějakého klíče. [22]

Dalším souborem, který je součástí souborového systému karty je User File Management File. Tento soubor má tolik záznamů, kolik je na kartě inicializováno uživatelských souborů. Každý záznam má 7 bytů, reprezentuje daný uživatelský soubor a určuje jeho vlastnosti. Tomuto záznamu se říká User File Definition Block, který znázorňuje tabulka č. 2. Ukládá informace o tom, jakého typu je soubor na kartě uložen, jaké pro daný soubor platí přístupová práva (tzn. zda je na daný soubor vyžadován Secure messaging – autentizace karty), dále velikost souboru, pokud se jedná o soubor nebo pokud se jedná o záznam, tak počet jednotek záznamů v tomto záznamu. Dále určuje identifikátor daného souboru. [22]

Tabulka 2 – User File Definition Block [22]

byte 1	byte 2	byte 3	byte 4	byte 5 / 6	byte 7
Record Length	Number of Records	Read Security Attribute	Write Security Attribute	File Identifier	File Access Flags
OR					
File Length (High Byte)	File Length (Low Byte)	Read Security Attribute	Write Security Attribute	File Identifier	File Access Flags

Soubory Account file a Account Security File jsem v tomto projektu nevyužil, proto je také nepopisuji.

Dalším souborem, který jsem využil, je ATR File. Soubor má jeden záznam o velikosti 36 bytů a ukládá informace o stavu karty. Jsou zde uloženy informace o tom, v jakém stavu se karta momentálně nachází, o tom, jaká verze karty je připojena a údaje z registrů popsaných výše v této kapitole. Další informace, které obsahuje tento soubor jsou zřetelné v tabulce č . 3. [22]

Po tomto souboru v souborovém systému karty již následují uživatelské soubory. Jejich počet a velikost se odvíjí od provedení procesu personalizace, tudíž záleží na programátorovi, jak kartu naprogramuje. [22]

Tabulka 3 – ATR File [22]

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
41_H	01_H	10_H 20_H 38_H	Option registers			Personalization File bytes 4 – 8					Life-cycle Stage	90_H	00_H

3.4.2 Proces personalizace karty

Programování karty ke zvolenému účelu prochází třemi základními stavy: Manufacturing stage, Personalization stage, User stage. Tyto stavy musí karta projít, aby byla připravena pro nasazení. V posledním stavu již karta zůstává po celou dobu své životnosti. Pokud se karta dostane do posledního stavu, je návrat do přechozích stavů nemožný, tudíž další úpravy souborů, které během předchozích stavů proběhly, již nejsou možné. Celý proces programování karty se nazývá proces personalizace. [22]

Na začátku procesu personalizace se musí programátor autorizovat pomocí Issuer klíče. Poté má oprávnění upravovat obsah jednotlivých souborů. Během procesu personalizace se Issuer klíč zadává několikrát, jelikož karta musí být během přechodu ze stavu do stavu odpojena od programovacího zařízení. Toto odpojení probíhá softwarově. Na začátku procesu personalizace se karta nachází ve stavu Manufacturer. Zde proběhnou úpravy v Manufacturer file, kde se nastaví požadované vlastnosti karty. Po provedení úprav se zapíše bit, který ukončuje Manufacturer stage a proběhne resetování připojení karty. Resetováním připojení se uloží změněné informace v kartě. [22]

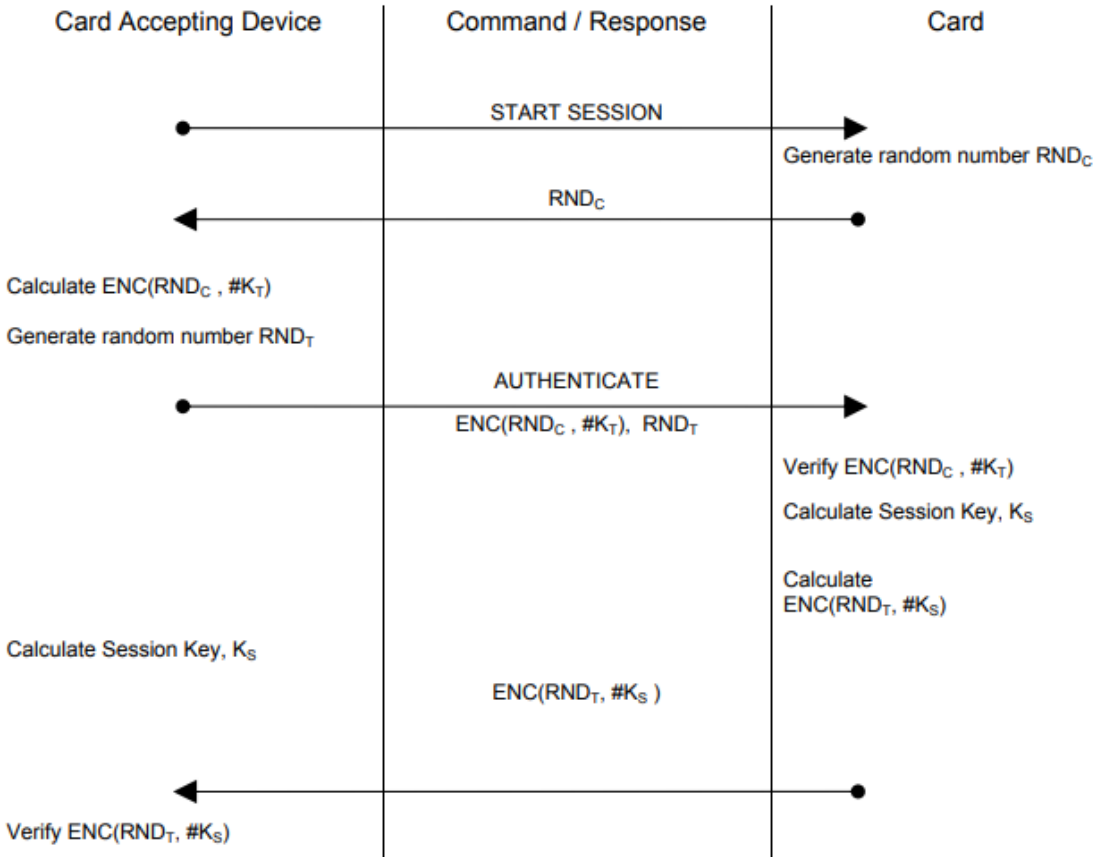
Karta přejde do druhého stavu Personalization stage. Nyní musí být opět programátor autentizován pomocí Issuer klíče. Následuje výběr Personalization File, ve kterém se nastaví počet uživatelských souborů a hodnoty klíčů. Po dokončení úprav musí dojít k dalšímu přerušení čili resetování připojení karty od zařízení, kterým je karta programována. [22]

Po nastavení počtu uživatelských souborů je nutné každému souboru nastavit jeho atributy. Tyto atributy jsou uloženy v User File Management File. Tento soubor je složen z File Definition Blocks, jak autor popisuje výše v této kapitole. Po ukončení inicializace uživatelských souborů se v případě potřeby inicializují další klíče, které se využívají v případě komunikace skrz secure messaging. [22]

Nakonec se zapíše Personalization bit, který je v souboru Personalization file. Následuje resetování připojení karty od programovacího zařízení. Tímto krokem je karta uzavřena a již nejsou možné další úpravy. Karta je ve stavu User stage. [22]

3.4.3 Secure Messaging

Jedná se o chráněný přenos dat. Pokud je secure messaging aktivní, data, která jsou do karty přiváděna, i data, která karta odesílá, jsou chráněna šifrováním pomocí DES nebo 3DES. Při použití secure messagingu musí být karta autorizována vůči zařízení, se kterým komunikuje. Tato autorizace probíhá pomocí výpočtu Session klíče. Proces autorizace je znázorněn na obrázku č. 8.



Obrázek 8 – Proces autorizace [22]

V prvním kroku zařízení, k němuž je karta připojena, odešle do karty požadavek START SESSION pomocí daného APDU příkazu. Na tento požadavek karta odpoví, tím, že do zařízení odešle náhodné číslo. Toto náhodné číslo je poté zařízením zašifrováno pomocí zvoleného šifrovacího algoritmu v OPTION registru. Zařízení si vygeneruje svoje náhodné číslo a spolu se zašifrovaným náhodným číslem karty, jej odešle zpět do karty. Nyní na obou stranách proběhne výpočet SESSION klíče. Karta do zařízení odešle zašifrované náhodné číslo, které si vygeneroval stojan. Pokud stojan rozšifruje zašifrované náhodné číslo pomocí

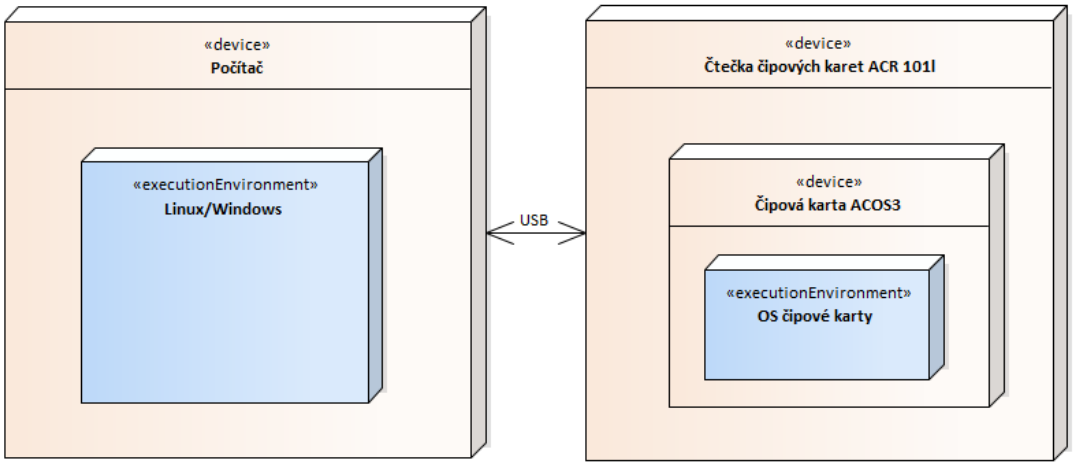
vypočteného SESSION klíče a dojde k závěru, že výsledkem je jeho vygenerovaného číslo, proběhla autorizace správně. [22]

Nyní se pomocí vypočteného SESSION klíče šifrují veškerá data odesílána i přijímána kartou. Rozložení dat v APDU příkazu je upraveno. Secure messaging slouží k tomu, aby data nemohla být odposlouchávána třetí stranou. [22]

II. PRAKTICKÁ ČÁST

4 POPIS ARCHITEKTURY

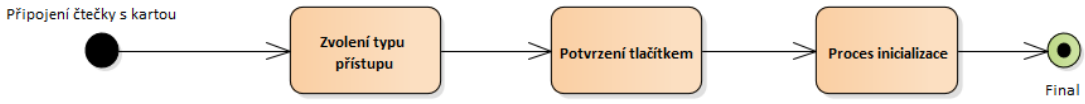
4.1 Inicializační program karty



Obrázek 9 – Deployment diagram inicializačního programu karty

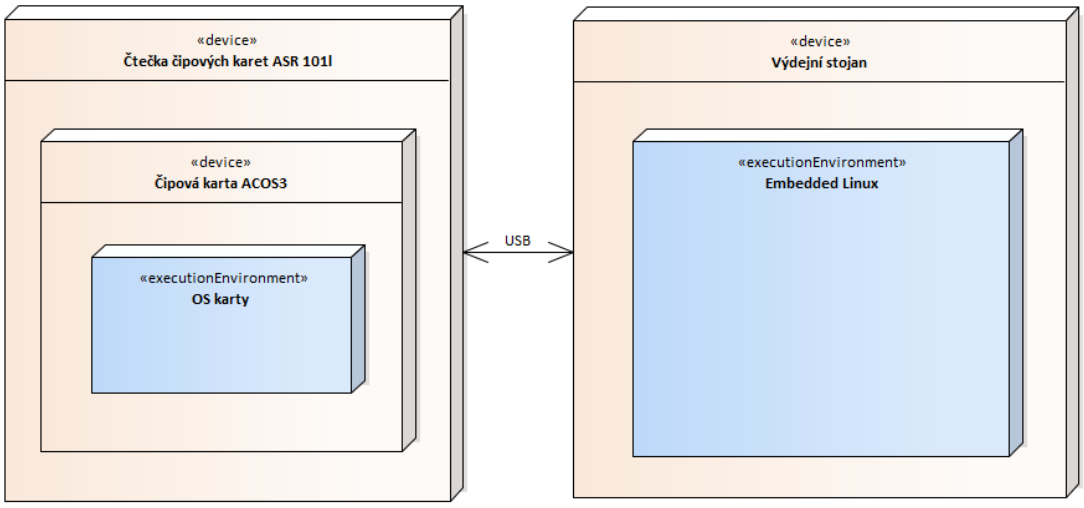
Deployment diagram ukazuje zobrazuje dvě zařízení, které mezi sebou komunikují. Zobrazuje jejich hardwarové součásti a jejich operační systém. Konkrétně se v tomto případě (viz obrázek č. 9) se jedná o programovací zařízení (počítač) s operačním systémem Linux nebo Windows a naproti tomu čtečku čipových karet ACR 101l. Součástí čtečky je další zařízení, které je znázorněné v diagramu – čipová karta ACOS3, která má svůj operační systém.

Aktivitní diagram na obrázku č. 10 vyjadřuje proces inicializace karty z pohledu uživatele. Diagram ukazuje, jak bude uživatel prostupovat programem.



Obrázek 10 – Aktivitní diagram inicializačního programu

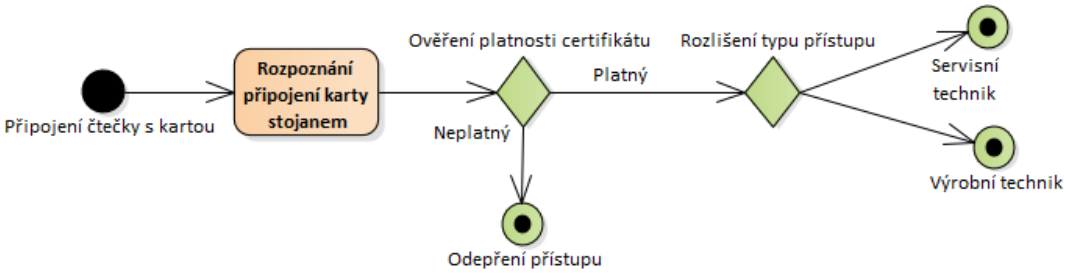
4.2 Ověřovací program ve výdejním stojanu



Obrázek 11 – Deployment diagram ověřovacího programu ve výdejním stojanu

Deployment diagram ověřovacího programu (viz obrázek č. 11) ve výdejním stojanu znázorňuje zařízení čtečky čipových karet ACR 101l a zařízení výdejního stojanu. Součástí čtečky je další zařízení, čipová karta ACOS3, která má svůj operační systém. Na druhé straně je znázorněn výdejní stojan s operačním systémem Embedded Linux.

V aktivitním diagramu ověřovacího programu je znázorněno, jak probíhá autentizace servisního či výrobního technika. Nejprve stojan rozezná, že k USB portu bylo připojeno zařízení. Poté se ověří platnost certifikátu uloženého v čipové kartě. Následně se rozliší, zda jde o servisního technika či výrobního technika (mají odlišné možnosti přístupu k nastavení).Tento diagram je znázorněn na obrázku č. 12.



Obrázek 12 – Aktivitní diagram ověřovacího programu

5 VYTVOŘENÉ PROGRAMY

Autor při vypracovávání práce vytvořil programy, jež popisuje v této kapitole.

5.1 Inicializační program

Pro připojení čtečky a karty k počítači slouží metoda SCardConnect, která je součástí knihovny pcsclite.h. Tato metoda má návratové hodnoty, dle kterých lze odvodit, jaký problém brání připojení karty k počítači. Těmi jsou, jak z ukázky kódu vyplývá, například „Reader unavailable“, který značí, že se nedaří připojit ke čtečce. Další návratové hodnoty této i ostatních metod jsou popsány v oficiální dokumentaci knihovny pcsclite.h. Důležitým vstupním parametrem této metody je název čtečky, ve které je připojena čipová karta.

```
rValue = SCardConnect(hSC, "ACS ACR101 ICC Reader 00 00", SCARD_SHARE_SHA-
RED, SCARD_PROTOCOL_T0, &hCard, &dwActiveProtocol);
        if (rValue == SCARD_S_SUCCESS)
          std::cout << "Card succesfully connected" << std::endl;
        else if (rValue == SCARD_E_READER_UNAVAILABLE)
          std::cout << "Reader unavailable" << std::endl;
        else if (rValue == SCARD_E_INVALID_PARAMETER)
          std::cout << "INVALID_PARAMETER" << std::endl;
        else if (rValue == SCARD_E_INVALID_HANDLE)
          std::cout << "INVALID_handle" << std::endl;
        else if (rValue == SCARD_E_INVALID_PARAMETER)
          std::cout << "INVALID_PARAMETER" << std::endl;
        else if (rValue == SCARD_E_INVALID_VALUE)
          std::cout << "INVALID_VALUE" << std::endl;
        else if (rValue == SCARD_F_COMM_ERROR)
          std::cout << "COMM_ERROR" << std::endl;
        else if (rValue == SCARD_E_NO_SERVICE)
          std::cout << "NO_SERVICE" << std::endl;
        else if (rValue == SCARD_E_NO_SMARTCARD)
          std::cout << "NO_SMARTCARD" << std::endl;
        else if (rValue == SCARD_E_READER_UNAVAILABLE)
          std::cout << "READER_UNAVAILABLE" << std::endl;
        else if (rValue == SCARD_E_UNKNOWN_READER)
          std::cout << "UNKNOWN_READER" << std::endl;
        else
          std::cout << "Other error" << std::endl;
```

Po připojení karty k programovacímu zařízení začíná proces personalizace karty. Prvním krokem je autorizace programátora pomocí Issuer klíče voláním metody ICSubmit, jež je popsána v kapitole 5.3. Následuje zvolení Manufacturer file, který se vybere pomocí metody selectFile. Vstupním parametrem bude ID Manufaturer souboru, který je „0xFF 0x01“. Po úspěšném zvolení souboru, se do souboru zapíše „Manufacturer fuse“, a to tak, že do prvního bytu prvního záznamu tohoto souboru se uloží hodnota „0xA0“. To značí, že se ukončí Manufacturer stage a číslování souborů bude počítáno od nuly (viz podkapitola 3.4.1).

Nyní se karta nachází ve druhém stavu svého životního cyklu – Personalization stage. V tomto stavu bude nastavena většina parametrů karty. Začíná se tím, že se zvolí Personalization file, ve kterém se nastaví obsah Option registru a počet uživatelských souborů. V tomto konkrétním případě se uloží do prvního bytu prvního záznamu hodnota „0x02“, což znamená, že je aktivní 3DES šifrování , při využití Secure Messagingu. V této verzi programu není secure messaging požadován. Dále se v prvním záznamu upraví třetí byte, a to na hodnotu „0x01“, což značí, že bude na kartě jeden uživatelský soubor.

Pro uložení provedených změn se provede reset připojení karty od počítače. Po resetu připojení je nutné, aby se programátor opět autorizoval pomocí Issuer klíče, což provede pomocí metody ICSubmit.

Nyní se zvolí User File Management File. V tomto souboru se zapisuje do prvního záznamu, který představuje první uživatelský záznam. Do tohoto záznamu se uloží blok „0x08, 0x00, 0x00, 0x00, 0x01, 0x01, 0x80“. Pro tento soubor je požadováno, aby byl typu File (viz kapitola 3.4.1), proto se uloží do posledního bytu (File Access Flag, viz tabulka č. 4) příznak „0x80“, jež zaručí, že bude soubor potřebného typu. Další příznaky, které tento byte ovlivňuje jsou znázorněny v následující tabulce.

Tabulka 4 – File Access Flag [22]

b7	b6	b5	b4	b3	b2	b1	b0	Meaning
0								Record File type – byte 1 and 2 of User File Definition Block specifies Record Length and Record Number respectively
1								Binary File type – byte 1 and 2 of User File Definition Block specifies File Length
	1							Read requires Secure Messaging
		1						Write requires Secure Messaging
			0	0	0	0	0	Reserved for future use

Dále je nutné souboru nastavit jeho požadovanou délku. O to se starají první dva byty. V tomto případě bude mít soubor velikost 2048 bytů (to vychází z toho, že je rezervováno 8-krát 256 bytů).

V dalším kroku se provede vyčtení sériového čísla karty. Sériové číslo je uloženo v MCU File a má 8 bytů. Pro výběr souboru se použije metoda selectFile, do které se předá ID souboru 0xFF 0x00, jako parametr. Po zvolení je nutné přečíst ze souboru první záznam, který má 8 bytů. Vyčtená data ze souboru se uloží do proměnné serialNumber, která je později součástí certifikátu.

```
BYTE MCUID[] = {0xFF, 0x00};
selectFile(MCUID, "selection of MCUID file");
readRecord(0x01,0x00,0x08, "serial number read");
BYTE serialNumber[8];
memcpy(serialNumber, pbRecvBuffer, 8);

uint32_t currentDateTime = QDateTime::currentSecsSinceEpoch();
BYTE timeOfSet[sizeof(currentDateTime)+261];
memcpy(timeOfSet,&currentDateTime,sizeof(currentDateTime));
BYTE modeS = {0x00};
BYTE modeA = {0x01};
```

Dále pomocí třídy QDateTime se zjistí aktuální čas pomocí metody currentSecsSinceEpoch, která vrátí číslo značící počet sekund od 1.1.1970. Tento formát zápisu času je zakotven v ISO 8601. Tento čas se uloží do proměnné currentDateTime, která se později stane součástí certifikátu.

Proměnné serialNumber a currentDateTime se spojí dohromady společně příznakem typu přístupu. Tento příznak je ve formě 0 a 1, kdy 0 představuje typ přístupu pro servisního technika a 0 představuje přístup pro výrobního technika. Tato data se poté podepíšou pomocí metody singMessage. Vytvořený certifikát je následně uložen do připraveného uživatelského souboru. Vytvoření certifikátu je podpořeno knihovnou openssl.h a jejími dalšími knihovnami. Ukládání probíhá ve dvou krocích, jelikož má certifikát délku 256 bytů, a za něj je ještě potřeba uložit původní data pro ověření podpisu, tudíž ve výsledku se jedná o data o délce 261 (sériové číslo se neukládá – je uloženo v MCUID souboru). Karta může na jednou uložit 256 bytů. Proto v prvním kroku je uloženo prvních 256 bytů a ve druhém kroku zbylých 5 bytů.

Dále program inicializuje klíče pro secure messaging, který v této verzi programu není implementován. V posledním kroku programu se vybere soubor Personalization File, aby se do čtvrtého bytu jeho druhého záznamu uložil Personalization bit, který ukončí Personalization stage karty. Po provedení tohoto kroku je karta v User stage a v tomto stavu je až do konce své životnosti.

5.2 Použité metody

Tato podkapitola popisuje metody, které autor vytvořil, pro zjednodušení při procesu inicializace karty. Jedná se o metody, které slouží pro čtení či zápis dat, nebo pro autorizaci programátora.

5.2.1 ICSubmit

```
void ICSubmit(void) {
  BYTE pbSendBufferSC[] = {0x80, 0x20, 0x07, 0x00, 0x08, 0x41, 0x43,
                           0x4F, 0x53, 0x54, 0x45, 0x53, 0x54};
  dwSendLength = sizeof(pbSendBufferSC);
  dwRecvLength = sizeof(pbRecvBuffer);
  printf("\nSending: ");
  for (int i = 0; i < (int)dwSendLength; i++)
    printf("0x%02X ", pbSendBufferSC[i]);
  std::cout << "Code submit" << std::endl;

  rValue = SCardTransmit(hCard, pioSendPci, pbSendBufferSC, dwSendLength,
                         &pioRecvPci, pbRecvBuffer, &dwRecvLength);
  if (rValue == SCARD_S_SUCCESS)
    std::cout << "Transmit success" << std::endl;
  printf("Receiving: ");
  for (int i = 0; i < (int)dwRecvLength; i++)
    printf("\x1B[32m0x%02X\033[0m ", pbRecvBuffer[i]);
  std::cout << "" << std::endl;
}
```

Pomocí této metody se do karty odesílá Issuer key, který slouží pro autorizaci programátora. Prvních pět bytů proměnné pbSendBufferSC jsou příkazy APDU příkazu. Dalších 8 bytů představuje samotný Issuer klíč. V této ukázce je použitý předdefinovaný klíč, který byl na kartě již od výroby. Další důležitou částí této ukázky kódu je metoda SCardTrasmit, která je součástí knihovny pcsclite.h, a která slouží pro odesílání APDU příkazu do karty. Spolu s APDU příkazem do této metody vstupují proměnné jako je délka vstupního příkazu, který se ukládá do proměnné dwSendLength.

Pak zde vstupuje proměnná, která bude sloužit pro příjem odpovědi karty – pbRecvBuffer. V tomto případě jsou možnými odpověďmi „0x90 0x00“ v případě správnosti všech parametrů, „0x63 0xCn“ v případě použití špatného klíče (písmenko n značí počet pokusů, které na autorizaci zbývají, maximální počet pokusů je 7). Existují i další typy odpovědí, ale pro jejich nedůležitost je zde neuvádím.

5.2.2 selectFile

```
void selectFile(BYTE idFile[], char msg[]) {
  BYTE pbSendBufferS[7];
  BYTE tmp[] = {0x80, 0xA4, 0x00, 0x00, 0x02};
  memcpy(pbSendBufferS, tmp, 5);
  memcpy(pbSendBufferS + 5, idFile, 2);
  dwSendLength = sizeof(pbSendBufferS);
  dwRecvLength = sizeof(pbRecvBuffer);
  printf("Sending: ");
  for (int i = 0; i < (int)dwSendLength; i++)
    printf("0x%02X ", pbSendBufferS[i]);
  std::cout << msg << std::endl;

  rValue = SCardTransmit(hCard, pioSendPci, pbSendBufferS, dwSendLength,
                         &pioRecvPci, pbRecvBuffer, &dwRecvLength);
  if (rValue == SCARD_S_SUCCESS)
    std::cout << "Transmit success" << std::endl;
  printf("Receiving: ");
  for (int i = 0; i < (int)dwRecvLength; i++)
    printf("\x1B[32m0x%02X\033[0m ", pbRecvBuffer[i]);
  std::cout << "" << std::endl;
}
```

Metoda selectFile slouží pro zvolení souboru, se kterým se v danou chvíli má pracovat. Do této metody vstupuje ID souboru v proměnné typu BYTE s názvem idFile. Obsah této proměnné se poté předává pbSendBufferS, do které se předtím uloží APDU příkaz, který je příznačný pro zvolení souboru. V tomto případě je to obsah proměnné tmp. Poté se s ostatními proměnnými, které vstupují do metody SCardTransmit, odešlou do karty, a odpovědí je proměnná pbRecvBuffer. Tyto odpovědi můžou nabývat hodnot „0x90 0x00“ v případě správnosti všech vstupních parametrů, „0x91 0xnn“ v případě, kdy je volen uživatelský soubor (nn značí pořadní číslo souboru) a „0x6A 0x82“ v případě, že soubor s daným ID neexistuje.

5.2.3 readRecord

Metoda readRecord slouží pro přečtení obsahu souboru, který byl vybrán. Metoda má vstupní parametry recNum, jenž slouží k určení záznamu, který se bude v souboru číst, offset určuje posun v bytech, odkud se bude číst a length značí počet bytů, které se budou číst. Spolu s dalšími parametry se odešle přes metodu SCardTransmit do karty. V proměnné pbRecvBuffer, která do metody SCardTransmit též vstupuje, se vrací odpověď karty. Pokud odpověď začíná byty „0x90 0x00“, následuje za těmito byty obsah přečteného souboru. Pokud se vrátí jiná odpověď, znamená to buď, že je některý z parametrů chybný, nebo jsou

parametry offset a length delší, než je skutečná délka souboru, nebo se jedná o jinou strukturu souboru, která se čte pomocí metody readBinary, jež je popsána dále v této práci.

```
void readRecord(BYTE recNum, BYTE offset, BYTE length, char msg[]) {
  BYTE pbSendBufferRFg[5];
  pbSendBufferRFg[0] = {0x80};
  pbSendBufferRFg[1] = {0xB2};
  pbSendBufferRFg[2] = recNum;
  pbSendBufferRFg[3] = offset;
  pbSendBufferRFg[4] = length;

  dwSendLength = sizeof(pbSendBufferRFg);
  dwRecvLength = sizeof(pbRecvBuffer);
  printf("Sending: ");
  for (int i = 0; i < (int)dwSendLength; i++)
    printf("0x%02X ", pbSendBufferRFg[i]);
  std::cout << msg << std::endl;

  rValue = SCardTransmit(hCard, pioSendPci, pbSendBufferRFg, dwSendLength,
                         &pioRecvPci, pbRecvBuffer, &dwRecvLength);
  if (rValue == SCARD_S_SUCCESS)
    std::cout << "Transmit success" << std::endl;
  printf("Receiving: ");
  for (int i = 0; i < (int)dwRecvLength; i++)
    printf("\x1B[32m0x%02X\033[0m ", pbRecvBuffer[i]);
  std::cout << "" << std::endl;
}
```

5.2.4 writeRecord

Pomocí metody writeRecord se do předem zvoleného souboru zapisují data. Do metody vstupují parametry jako číslo záznamu, do kterého se data mají zapsat, posun, jenž představuje počet bytů, které od začátku souboru zůstanou prázdné. Dále do metody vstupuje délka vstupních dat a vstupní data samotné. Tyto vstupní parametry se uloží do jedné proměnné spolu s APDU příkazy a celý tento APDU příkaz se odešle do karty. Odpovědí je 0x90 0x00 při úspěšném zápisu dat. Pokud karta odešle jakoukoliv jinou odpověď, znamená to, že data se na kartu nezapsala a pokus musí být opakován.

5.2.5 readBinary

Metoda readBinary funguje obdobně jako metoda readRecord, ale jelikož pro zápis dat typu record a typu file musí být použitý rozdílný APDU příkaz, používám dvě metody.

5.2.6 readBinary

Metoda readBinary funguje obdobně jako metoda writeRecord, ale jelikož pro zápis dat typu record a typu file musí být použitý rozdílný APDU příkaz, používám dvě metody.

5.2.7 clearCard

Tato metoda se volá na začátku každé inicializace. Má za úkol vyčistit kartu do výrobního stavu. Tento proces je však možný pouze tehdy, kdy je karta v Personalization nebo Manufacturer stage. Pokud se karta nachází již ve stavu User stage, není tento proces možná a kartu již nejde upravovat v plném rozsahu (je možné upravovat pouze obsah uživatelských souborů).

5.2.8 signMessage

Pro vytvoření certifikátu se používá metoda signMessage, která pomocí dalších metod vytvoří certifikát. Jejími vstupními parametry je soukromý klíč a text, jenž má být podepsán. Výsledkem je 256 bytů dlouhý podpis.

5.2.9 verifySignature

Slouží pro ověření správnosti certifikátu. Vstupními parametry jsou veřejný klíč a certifikát k ověření. Výstupním parametrem je pravdivostní výraz o platnosti certifikátu.

5.3 Ověřovací program

V prvním kroku programu je připojení karty k výdejním stojanu pomocí metody SCardConnect, která je součástí knihovny pcsclite.h. Poté se vybere soubor MCUID, ze kterého se vyčte sériové číslo karty pro ověření platnosti certifikátu.

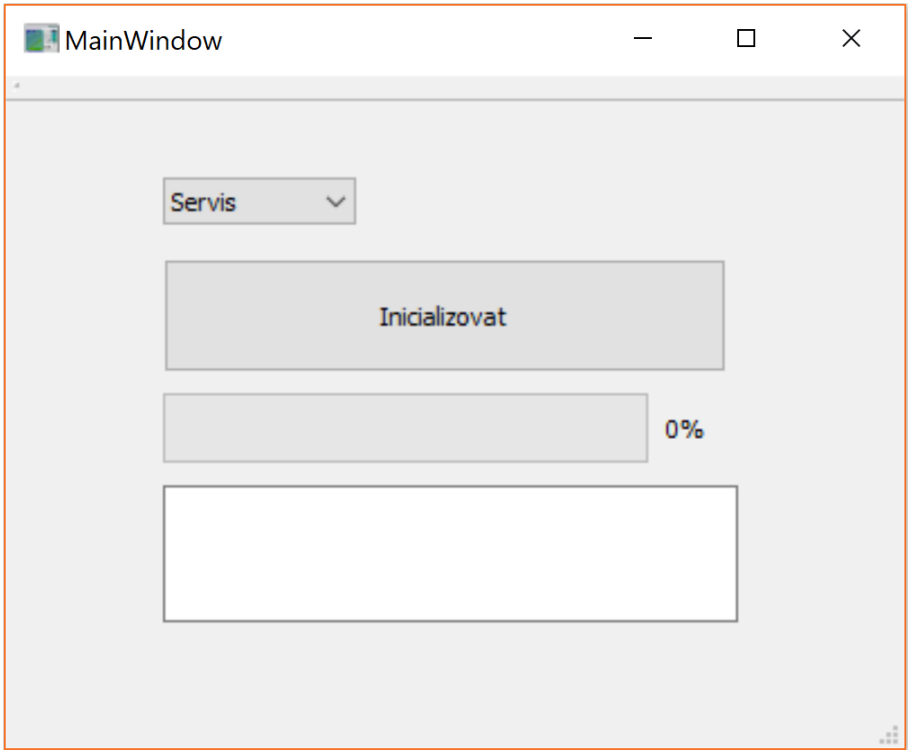
V dalším kroku se zvolí a přečtou z uživatelského souboru data, která se do karty uložila při inicializaci. Tato data se rozdělí na certifikát a data, se kterými se bude certifikát ověřovat. K datům, která slouží ověření certifikátu, se připojí sériové číslo karty. Pomocí metody verifySignature se ověří platnost certifikátu.

Celý běh programu je zpracován do jedné metody getCredentials. Její návratovou hodnotou je pole proměnných uint32_t, které obsahují čas vytvoření karty, typ přístupu a příznak ověřenosti certifikátu. Časovou platnost karty se ověřuje ve stojanu podle aktuálního času. Pokud je aktuální čas více než rok po vydání karty, je karta neplatná a přístup je odepřen.

5.4 Návod k použití aplikace

Aplikace obsahuje rozbalovací seznam, ve kterém se volí, na jaký typ přístupu má být karta připravena. Dále obsahuje tlačítko Inicializovat, které spustí inicializaci. Pod tlačítkem je ukazatel průběhu inicializace. Po dokončení inicializace se zobrazí text „Hotovo“ v textovém poli, které je umístěné pod ukazatelem průběhu.

Na začátku inicializace je potřeba připojit k počítači čtečku čipových karet ASC ACR 1011, v níž je vložena nová čipová karta ACOS3. Poté je potřeba zvolit typ přístupu přes rozbalovací seznam. Dalším krokem je stisknutí tlačítka Inicializovat. Po doběhnutí ukazatele průběhu na konec a zobrazení textu „Hotovo“, je možné čtečku odpojit. Karta s je připravena k používání. Obrázek č. 13 zobrazuje grafické zpracování aplikace.



Obrázek 13 – GUI inicializační aplikace

6 ZHODNOCENÍ BEZPEČNOSTNÍCH RIZIK

Zvolené řešení s sebou nese možná bezpečnostní rizika. Jedním z hlavních bezpečnostních rizik je krádež karty. Pokud dojde k odcizení karty, zloděj dále překoná zámek stojanu, může se dostat nekontrolovaně až do nastavení stojanu. Tento problém by šel ošetřit zavedením PINu při vložení karty do stojanu, avšak požadavky firmy byly PIN kód nepoužívat.

Dále může být bezpečnostním rizikem to, že přenášena data mezi kartou a stojanem nejsou chráněna. To znamená, že se dají odposlouchávat třetí stranou. To může vést k tomu, že hacker může zjistit strukturu dat a může se pokusit tyto data napodobit. To však je ošetřeno tím, že se používá pro ověření správnosti dat elektronický podpis šifrovacího algoritmu RSA. To je bez znalosti soukromého klíče jen velice obtížné, až skoro nereálné. Data je možné chránit pomocí secure messagingu, který nabízí přímo čipová karta. Ale vzhledem k použití zastaralých šifrovacích algoritmů, které karta pro secure messaging používá a také vzhledem k časové náročnosti implementace této funkcionality není secure messaging v této verzi programu implementován. Lepším řešením by mohla být novější verze karty ACOS5, která již podporuje šifrování pomocí RSA.

ZÁVĚR

Cílem práce bylo provést průzkum aktuálního stavu knihoven pro bezpečné přihlašování v embedded aplikacích, analyzovat požadavky a následně navrhnout a implementovat systém bezpečného přihlašování.

V teoretické části jsme společně s firmou Adast Systems a.s. nadefinovali hlavní požadavky pro přihlašování do výdejního stojanu čerpací stanice. Tyto požadavky však bylo nutné doplňovat a upravovat i během procesu vývoje nového systému pro výdejní stojan. Následně jsem v teoretické části popsal šifrovací algoritmy, se kterými jsem se dostal do kontaktu během vývoje systému bezpečného přihlašování.

Teoretická část dále obsahuje stručný popis technologií, které je možné použít jako alternativu. Původně jsme pro systém bezpečného přihlašování chtěli použít Java karty od firmy Oracle, které splňují standardy vydávané neziskovou organizací GlobalPlatform. Avšak pro nedostatek informací a ukázkových projektů nebylo v mých silách vytvořit applet pro platformu JavaCard.

Proto jsem zvolil na první pohled snadnější variantu pomocí procesorových karet ACOS3, které však nedosahují takové bezpečnosti, jako Java karty. V teoretické a v praktické části práce jsem shrnul postupy programování karet ACOS3. Tato práce může sloužit čtenářům pro porozumění programování čipových karet.

Zvolené a vypracované řešení má slabiny, které jsou shrnuty v závěru praktické části práce. Velkou překážkou při vývoji řešení s kartami ACOS je slabá přímá podpora ze strany výrobce a nedostatečná dokumentace. Na tuto překážku jsem tvrdě narazil například v případě hardwarového omezení počtu neúspěšných pokusů o autorizace programátora pomocí Issuer klíče. Zde jakákoli neznalost nebo chyba v kódu inicializace karty, která způsobí překročení limitu 7-mi neúspěšných pokusů o autorizaci, danou kartu trvale zablokuje. Ta je pak pro další vývoj i jakékoli jiné účely nepoužitelná. Cena jedné karty je přibližně 100 Kč, což velmi limituje možnost ladění a hledání takovýchto chyb.

Veškerá omezení mohou být samozřejmě překonána vynaložením většího úsilí a finančních nákladů, což platí i v případě zlepšování bezpečnosti informačních systémů. V našem

případě je v dalších etapách vývoje nutné získat přímou podporu výrobce nebo použít jiný typ karty s lepší dokumentací.

Tato práce mi byla velkým přínosem jak při rozšiřování mých znalostí programování v jazyce C++, tak v oblasti principu funkčnosti a programování čipových karet. Také jsem získal zkušenosti pro práci ve vývojovém týmu.

SEZNAM POUŽITÉ LITERTURY

[1] STROUSTRUP, Bjarne. *The C++ programming language.* Fourth edition. Upper Saddle River, NJ: Addison-Wesley, [2013]. ISBN 978-0321563842.

[2] ALEX. Naming conflicts and the std namespace. *LearnCpp.com* [online]. 2016, 6.11.2016, 1.2.2019 [cit. 2019-4-15]. Dostupné z: https://www.learncpp.com/cpp-tutorial/naming-conflicts-and-the-std-namespace/

[3] Input/Output. *Cplusplus.com* [online]. 2019 [cit. 2019-04-15]. Dostupné z: http://www.cplusplus.com/reference/iolibrary/

[4] EMBEDDED LINUX AND DESKTOP LINUX: DIFFERENCE AT A GLANCE. *EmbeddedCraft: Crafting on intelligent systems* [online]. 1.2.2019 [cit. 2019-4-15]. Dostupné z: http://www.embeddedcraft.org/embedlinuxdesktoplinux.html#top

[5] What is an RTOS? *Wittenstein: HighIntegritySystems* [online]. 2019, 1.2.2019 [cit. 2019-4-15]. Dostupné z: https://www.highintegritysystems.com/rtos/what-is-an-rtos/

[6] Algoritmus RSA. *Algoritmy.net* [online]. 2019, 1.2.2019 [cit. 2019-4-15]. Dostupné z: https://www.algoritmy.net/article/4033/RSA

[7] SCHNEIER, Bruce. *Applied cryptography: protocols, algorithms, and source code in C*. 2nd ed. New York: Wiley, c1996. ISBN 0471128457.

[8] BURNETT, Steve a Stephen PAINE. *RSA Security's Official Guide to Cryptography*. Kniha. USA: RSAPress, 2001. ISBN 0-07-219225-9.

[9] OpenSSL: Cryptography and SSL/TLS Toolkit [online]. *OpenSSL Software Foundation*, 2018 [cit. 2019-04-15]. Dostupné z: https://www.openssl.org/

[10] Smart card overview. *CardLogix* [online]. CardLogix Corporation, 2019 [cit. 2019-04-15]. Dostupné z: https://www.cardlogix.com/smart-card-overview/

[11] Facts & figures. EuroSmart: *The voice of the smart security industry* [online]. Belgie, 2018 [cit. 2019-04-15]. Dostupné z: http://www.eurosmart.com/facts-figures.html

[12] Types of smart cards. *CardLogix* [online]. CardLogix Corporation, 2019 [cit. 2019-4-15]. Dostupné z: https://www.cardlogix.com/types-of-smart-cards/

[13] Types of smart cards. *CardLogix* [online]. CardLogix Corporation, 2019 [cit. 2019-4-15]. Dostupné z: http://www.smartcardbasics.com/smart_card_images/smart-card-construction.jpg

[14] Types of smart cards. *CardLogix* [online]. CardLogix Corporation, 2019 [cit. 2019-4-15]. Dostupné z: http://www.smartcardbasics.com/smart_card_images/types-of-smart-cards.gif

[15] Types of smart cards. CardLogix [online]. CardLogix Corporation, 2019 [cit. 2019-4-15]. Dostupné z: http://www.smartcardbasics.com/smart_card_images/smart-card-module.gif

[16] *Advanced Card Systems Ltd.: Card & Reader Technologies* [online]. Hongkong: Advanced Card Systems, 2019 [cit. 2019-05-14]. Dostupné z: https://www.acs.com.hk/

[17] ACR101l: MicroSIM(CCID). *Advanced Card Systems Ltd.: Card & Reader Technologies* [online]. Advanced Card Systems, 2019 [cit. 2019-4-16]. Dostupné z: https://www.acs.com.hk/en/products/141/acr101i-simicro-ccid/

[18] ACR101l: MicroSIM(CCID). *Advanced Card Systems Ltd.: Card & Reader Technologies* [online]. Advanced Card Systems, 2019 [cit. 2019-4-16]. Dostupné z: https://www.acs.com.hk/en/download-product-image-library/1775/20121229161913lib_acr101_3.png

[19] Smart Card Standards. *CardLogix* [online]. CardLogix Corporation, 2019 [cit. 2019-4-16]. Dostupné z: https://www.cardlogix.com/smart-card-standards/

[20] ACOS3: Microprocessor Card (Contact). *Advanced Card Systems Ltd.: Card & Reader Technologies* [online]. Advanced Card Systems, 2019 [cit. 2019-4-16]. Dostupné z: https://www.acs.com.hk/en/download-product-image-library/2814/2814-images-acos3-contact.png

[21] ACOS3: Microprocessor Card (Contact). *Advanced Card Systems Ltd.: Card & Reader Technologies* [online]. Advanced Card Systems, 2019 [cit. 2019-4-16]. Dostupné z: https://www.acs.com.hk/en/products/306/acos3-microprocessor-card-contact/

[22] Reference manual: *ACOS3* [online]. 3.11. Hongkong: Advanced Card Systems, 2008 [cit. 2019-04-16]. Dostupné z: https://www.smartcardsreaders.shop/index.php?dispatch=attachments.getfile&attachment_id=155

[23] Raspberry Pi. *AlternativeTo* [online]. [cit. 2019-05-15]. Dostupné z: https://d2.alternativeto.net/dist/s/5d674ef5-ce34-e311-b64a-002590a05f5f_1_full.jpg?format=jpg&width=1600&height=1600&mode=min&upscale=false

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

a.s.	Akciová společnost
USB	Universal seriál bus
SD	Secure digital
ISO	International Standardization Organization
RTOS	Real Time Operating Systém
RSA	iniciály autorů Rivest, Shamir, Adleman
tzn.	To znamená
DES	Data Encryption Standard
IBM	International Business Machines
NFC	Near Field Communication
RFID	Radio Frequency Identification
mld.	miliarda
PVC	Polyvinylchlorid
č.	číslo
MHz	Megahertz
PC	Personal computer
TCP/IP	Transmission Control Protocol/Internet Protocol
ACS	Advanced Card Systems
SIM	Subscriber Identity Module
PC/SC	Personal Computer/Smart Card

MAC OS	Macintosh operating systems
SQL	Structured Query Language
cm	centimetr
APDU	Application Protocol Data Unit
KB	kilobyte
ID	Identifier
R	Read
W	Write
MAC	Message Authentication Code
PIN	Personal identification number

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH CD

Přiložené CD obsahuje:

- Text:
  - Implementace_bezpecneho_prihlasovani_v_embedded_aplikacich_Filip_Miskarik.docx – bakalářská práce ve formátu docx
  - Implementace_bezpecneho_prihlasovani_v_embedded_aplikacich_Filip_Miskarik.pdf – bakalářská práce ve formátu pdf
- Zdrojové kódy:
  - prilohy.zip – Zdrojové kódy pro inicializační a ověřovací aplikaci