Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Emulace čipové karty pomocí technologie NFC

Diplomová práce

Autor: Bc. Jan Brož, DiS
Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. Ing. Filip Malý, Ph.D.

Hradec Králové srpen 2016

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne

Poděkování:

Děkuji vedoucí bakalářské práce doc. Ing. Filipu Malému, Ph.D. za odborné vedení práce, za věcné připomínky a za cenné rady. Rovněž bych chtěl poděkovat mé rodině a mým přátelům za pomoc a jejich trpělivost v době, kdy jsem tuto práci psal.

Anotace

Diplomová práce se zabývá možnostmi emulace čipových karet pomocí rozhraní NFC na mobilní platformě Android. První část popisuje obecné principy z oblasti kryptografie a zároveň se zabývá problematikou kontaktních a bezkontaktních karet. V druhé části je provedena analýza bezpečného úložiště šifrovacích klíčů v systému Android. V rámci této práce byl vytvořen nezávislý platební systém, ve kterém mobilní telefon vystupuje v roli bezkontaktní platební karty. Součástí systému je platební terminál a autentizační server, přičemž pro autentizaci uživatele jsou implementovány dva nezávislé mechanismy. První na bázi symetrické kryptografie, druhý na bázi asymetrické kryptografie. V závěru práce je porovnání obou variant implementace s cílem vybrat takové řešení, které zajistí maximální bezpečnost a pohodlí uživatele.

Annotation

The thesis deals with the emulation of smart cards by using NFC on Android mobile devices. The first part describes the main principles of cryptography and focuses on contact and contactless smart card technology. This part also analyzes the secure storage of encryption keys in the Android operating system. The main goal was to create an independent payment system in which a mobile phone acts as a contactless smart card. The implemented payment system includes both a payment terminal and authentication server. For user authentication there are two independent mechanisms implemented. The first one is based on a symmetric cryptography, while the second one is based on an asymmetric cryptography. In conclusion there is a comparison of both implemented variants. The main objective is to choose a solution that ensures maximum security and provides the best possible experience for the user.

Obsah

1 Úvod

Čipové karty se za poslední desetiletí staly standardní součástí našich peněženek. Zaujímají čestné místo hned vedle osobních dokladů. Používají se téměř denně, od malých plateb za zboží v obchodě, přes výběry z bankomatu, až po platby na internetu.

Od roku 2007 se do popředí dostávají tzv. „chytré“ mobilní telefony, které již disponují vyspělými SIM kartami, případně integrovanými bezpečnostními moduly. V kombinaci s bezdrátovým rozhraním NFC (Near Field Communication) pak umožňují emulovat hned několik karet pomocí telefonu.

Cílem práce je vytvoření nezávislého platebního systému, ve kterém bude mobilní telefon vystupovat v roli bezkontaktní platební karty. Pro autentizaci uživatele a přenos transakčních dat bude využito rozhraní NFC v režimu Host-based Card Emulation, jehož API Google oficiálně uvolnil v roce 2013. V rámci zabezpečení aplikace budou navrhnuty a implementovány dva autentizační mechanizmy. První na bázi symetrické kryptografie, druhý na bázi asymetrické kryptografie. Cílem práce je implementovat takové řešení, které zajistí maximální bezpečnost a pohodlí uživatele. Požadavkem je provést celou autentizaci ze strany mobilního telefonu v režimu off-line. To znamená, že telefon není třeba mít připojen k internetu v okamžiku přiložení k platebnímu terminálu. Zároveň musí být zajištěna maximální bezpečnost mobilní aplikace. Interní úložiště citlivých dat musí být schopné zamezit vyjmutí bezpečnostních klíčů a jejich následné použití v jiném zařízení. Jednalo by se o útok, který by měl za následek vytvoření libovolného počtu kopií telefonu se stejnými platebními údaji uživatele.

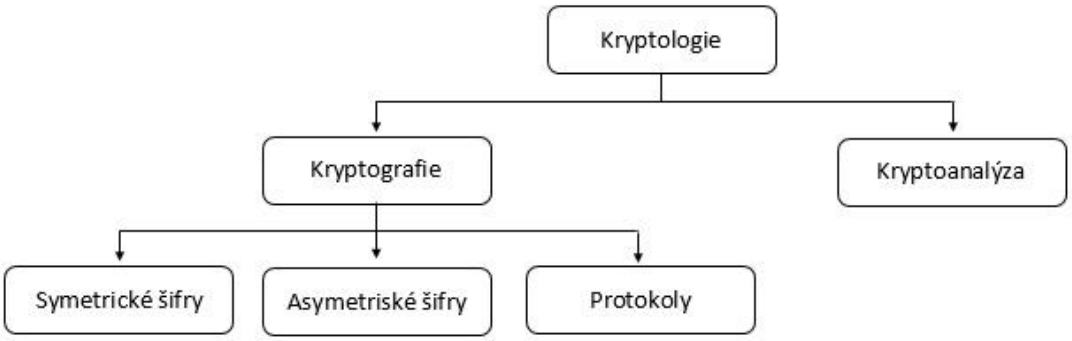
První část práce se obecně zabývá pojmy a principy z oblasti kryptografie. Vysvětleny jsou rozdíly mezi symetrickou a asymetrickou šifrou, na jejichž základech jsou pak postaveny dvě platební aplikace mobilního telefonu. Další kapitoly popisují funkčnost čipových karet, jejich dělení a principy komunikace s terminálem. Následuje popis technologie NFC společně s bezpečností systému Android. Důraz je kladen na analýzu dostupného úložného prostoru pro bezpečnostní klíče.

Ve druhé části je popsán návrh a implementace platebního systému pro variantu na bázi symetrické, následně pak asymetrické kryptografie. U obou

variant se práce podrobněji zabývá registrací uživatele a průběhem jeho ověřování při zahájení bezkontaktní platební transakce mobilním telefonem. V závěru této kapitoly je pak porovnání obou variant.

2 Základní pojmy a principy z oblasti kryptologie

Pojem kryptologie v sobě zahrnuje dva základní pojmy – kryptografii a kryptoanalýzu.



Obrázek 1: Přehled témat kryptologie [1]

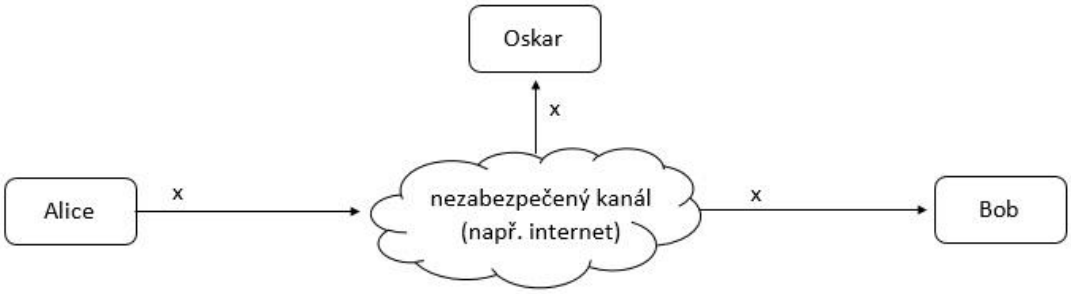
Kryptografie je věda, která se zabývá šifrováním, jehož cílem je zajistit utajení významu přenášené zprávy. Na začátku jsou data v běžně čitelné podobě, i když se v současné době již nemusí jednat pouze o text. Po jeho zakódování či zašifrovaní vzniknou další data, která by měla být nečitelná a splňovat jisté statistické požadavky. Pro jejich označení se používá pojem ciphertext, v české literatuře se překládá jako šifrovaný text. Hlavním úkolem kryptografie je zajištění důvěrnosti chráněných dat. Nikdo nepovolaný nesmí mít možnost přečíst data, která jsou chráněna kryptografickými prostředky, a to ani po vyvinutí jistého úsilí, například nasazením vysoce výkonných výpočetních systémů či týmu odborníků [2].

Kryptoanalýza je věda zabývající se získáváním původního významu zašifrované zprávy. Kryptoanalýza je dnes důležitou součástí moderních kryptosystémů a bez jejího uplatnění by nebylo možné prohlásit daný algoritmus za skutečně bezpečný. Vzhledem k zaměření této práce bude podrobněji popsána pouze kryptografie. Ta zahrnuje dva základní způsoby šifrování: symetrické a asymetrické. Jejich popisem se zabývají dvě následující podkapitoly.

2.1 Symetrické šifry

V symetrickém šifrování jsou klíče pro šifrování i dešifrování shodné. Často se tento klíč značí jako symetrický (tajný) klíč. Výhodou symetrické kryptografie je její obrovská rychlost ve srovnání s asymetrickými algoritmy. Nevýhodou pak

vyšší nároky na počet klíčů a jejich správu [2]. Funkcionalitu symetrického šifrování lze nejlépe předvést na dvou uživatelích Alici a Bobovi [1], kteří spolu chtějí navzájem komunikovat a zvolí k tomu nezabezpečený komunikační kanál. Tím může být internet, telefon, Wi-Fi nebo jakýkoliv jiný komunikační prostředek. Problém nastává ve chvíli, kdy třetí osoba (budeme jí říkat např. Oskar) získá přístup do komunikačního kanálu (hacking, odposlech sítě atd.). Oskar tak získá přístup k informacím, které měly původně zůstat pouze mezi Alicí a Bobem.



Obrázek 2: Komunikace přes nezabezpečený kanál [1]

Symetrická kryptografie nabízí pro zabezpečení toku informací následující řešení: Alice zašifruje svou zprávu *x* symetrickým algoritmem a vznikne tak zašifrovaná zpráva *y*. Bob obdrží tuto zprávu a znovu ji dešifruje. Dešifrování je v tomto případně opačný proces šifrování. Pro Oskara se takto zašifrovaná zpráva tváří jako náhodný shluk znaků, které nedávají smysl [1].

Obrázek 3: Symetrická kryptografie [1]

Celý proces je ilustrován na obrázku 3, kde význam proměnných je následující [1]:

- x je původní zpráva označovaná jako *plaintext* nebo *cleartext*
- y je zašifrovaná zpráva označovaná jako *ciphertext*
- k je klíč

Popisovaný princip vyžaduje, aby sdílený klíč mezi Alicí a Bobem byl distribuován zabezpečenou cestou. Podstatnou výhodou symetrických šifer je jejich nízká výpočetní náročnost, nevýhodou je však nutnost sdílet tajný klíč mezi oběma účastníky.

Jedním z nejrozšířenějších algoritmů byl dlouhou dobu DES, který používal šifrovací klíč délky 56 bitů. Dnes se však již považuje za nedostatečný a je nahrazen algoritmem 3DES s délkou klíče 112 nebo 168 bitů. Aktuálně doporučovaným algoritmem je však algoritmus AES s délkou klíče 128, 192 nebo 256 bitů [3].

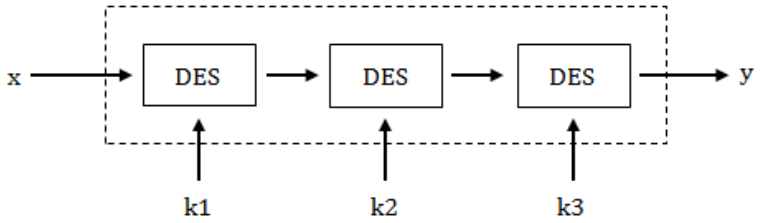
2.1.1 DES

Algoritmus DES (Data Encryption Standard) byla jedna z nejvyužívanějších blokových šifer posledních 30 let. V roce 1977 byl oficiálně zvolen za standard (FIPS PUB 46) pro šifrování dat v civilních státních organizacích USA a nedlouho na to se rozšířil i do soukromého sektoru [1]. Šifra pracuje s bloky o délce 64 bitů a klíčem velikosti 56 bitů. Každý blok je rozdělen na polovinu (L_0, R_0). Další postup šifry popisuje M. Ignjatovič takto: „*Vstupní 64bitová posloupnost projde vstupní permutací a poté se předá funkci, která používá statické tabulky permutací a substitucí. Pomocí klíče se vstupní data permutují na dvě posloupnosti o délce 48 bitů, které se pak předávají jako vstup statickým tabulkám permutací a substitucí. Tento postup se opakuje šestnáctkrát, vždy s jinými tabulkami a jinými bity klíče. Jejich výstupem jsou dvě 32bitová slova, která po finální permutaci dávají 64 bitů zašifrovaného textu*“ [4].

2.1.1.1 Bezpečnost DES

Krátce po svém zveřejnění čelil tento algoritmus hned několika námitkám. Největším nedostatkem se jevilo použití klíče délky pouze 56 bitů. Tím se šifra stává prolomitelnou za použití hrubé síly (tzv. brute force atack). V historii došlo k několika úspěšným pokusům o prolomení šifry. První zmínky o teoretickém útoku lze nalézt již v roce 1977. Konkrétní návrh zařízení však zveřejnil až Michael Wiener v roce 1993. Jednalo se o přístroj, který dokáže prolomit šifru za 1 a půl

dne a jehož náklady byly vyčísleny na 1 milión dolarů. V roce 1998 pak došlo k sestrojení podobného zařízení s názvem Deep Crack. Tento přístroj dokázal šifru prolomit za 56 hodin a jeho cena byla odhadnuta na 250 tisíc dolarů. V roce 2006 zařízení s názvem COPACOBANA dokázalo algoritmus DES prolomit za méně než 7 dní [1]. Vzhledem ke slabinám algoritmu byla tato šifra postupně nahrazena její novější variantou s názvem 3DES. Triple DES je vylepšenou verzí algoritmu DES, který se v tomto případě aplikuje třikrát za použití tří rozdílných klíčů, čímž zvyšuje odolnost proti útoku hrubou silou. Algoritmus 3DES je stále využíván ve finančních aplikacích, stejně tak lze nalézt využití v ochraně biometrických informací v elektronických pasech.



Obrázek 4: Algoritmus Triple DES (3DES) [1]

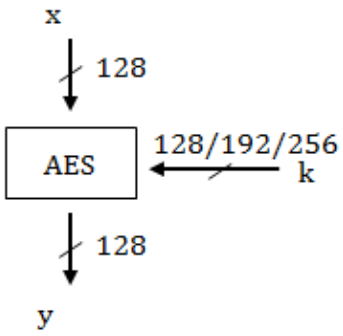
Po výpočetní stránce je 3DES považován za relativně náročný algoritmus. Oproti jeho předchozí variantě je navíc algoritmus 3DES ještě třikrát pomalejší. Z těchto důvodů se od něho pomalu ustupuje a je nahrazován modernější variantou – algoritmem AES.

2.1.2 AES

Advanced Encryption Standard je dnes jedna z nejvíce využívaných šifer. Používá ji k zabezpečení svých dat běžně i americká Národní bezpečnostní agentura NSA. Své uplatnění nalezne i v řadě komerčních standardů jako je např. IPsec, TLS, Wi-Fi šifrování IEEE 802.11i, SSH, Skype a další. Novou šifru schválil 26. 11. 2001 americký Národní úřad pro standardizaci (NIST) v publikaci FIPS PUB 197 jako federální standard USA s účinností od 26. 5. 2002. Na rozdíl od DES byl výběr této nové šifry otevřeným procesem pod administrací NIST[1]. Vítězem soutěže se stala bloková šifra Rijndael, která pochází od dvou belgických tvůrců

[1] US National Institute Of Standards and Technology

Joana Daemena a Vincenta Rijmena [1]. Šifra AES je téměř identická s šifrou Rijndael. Avšak hlavní rozdíl spočívá ve velikosti bloku a klíče. Rijndel podporuje rozpětí 128, 192 a 256 bitů. Naproti tomu standard AES podporuje bloky velikosti pouze 128 bitů s klíči o velikosti 128, 192 a 256 bitů.



Obrázek 5: Algoritmus AES s blokem o délce 128 bitů [1]

Podle délky klíče se mění počet iterací. Pro nejkratší klíč postačuje 10 kol, pro střední klíč 12 kol a pro 256 bitový klíč pak celých 14 kol. V jednotlivých kolech je provedena substituce (podobně jako u algoritmu DES), potom následují dva speciální transpoziční kroky. Blok je uspořádán do matice, nejprve jsou rotovány jednotlivé řádky. V následujícím kroku jsou promíchány sloupce pomocí vynásobení speciální maticí. Na závěr jsou data zkombinovaná s šifrovacím klíčem. Klíč se stejně jako u DESu, pro každé kolo mění [2].

2.1.2.1 Bezpečnost AES

Bezpečnostní experti považují za úspěšné prolomení šifry takovou metodu, která je rychlejší než útok hrubou silou. V současné době není známa žádná metoda, která by svou složitostí byla účinnější než útok hrubou silou [1]. V roce 2002 ohlásili Nicolas Courtois a Josef Pieprzyk teoretický útok pod názvem "XSL attack", ve kterém prohlašovali, že šifra je prolomitelná z důvodu své jednoduchosti. Další odborné publikace ovšem tuto možnost prolomení vyvrátily.

2.2 Asymetrické šifry

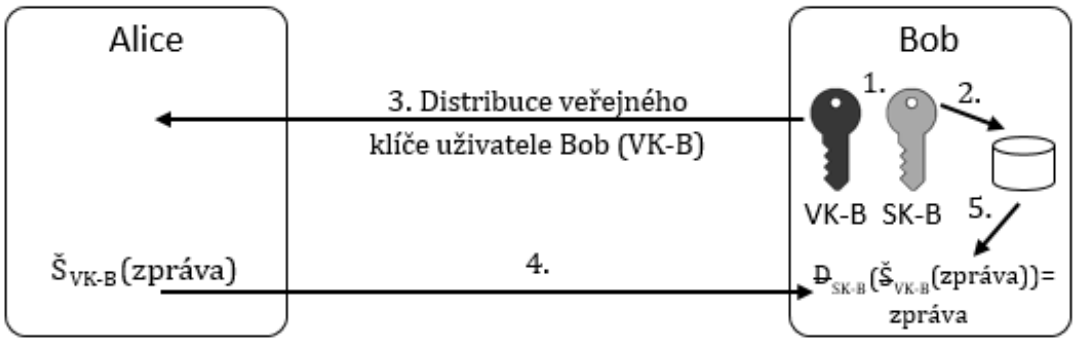
Základním znakem asymetrické kryptografie je existence dvou klíčů pro obě komunikující strany. Klíče navíc tvoří tzv. klíčový pár. Vlastností tohoto páru je skutečnost, že text zašifrovaným jedním klíčem z páru je možné dešifrovat pouze

druhým klíčem ze stejného páru. Zároveň text zašifrovaný jedním klíčem není možno tímto klíčem zpětně dešifrovat. Vžilo se označení jednoho klíče jako soukromého (private), druhého jako veřejného (public). Důležité je, aby uživatel uchovával soukromý klíč v dokonalé tajnosti. Veřejný klíč může libovolně rozšiřovat [2].

Asymetrická kryptografie se někdy označuje pojmem PKI (public-key infrastructure). Asi nejznámějším šifrovacím algoritmem je algoritmus RSA.

Pokud chce Alice šifrovat zprávu Bobovi asymetrickou šifrou, pak [3]:

1. Bob, tj. příjemce zprávy, si musí vygenerovat dvojici klíčů: veřejný klíč (VK-B) a soukromý klíč (SK-B).
2. Bob si uloží svůj soukromý klíč do důvěryhodného uložiště klíčů, např. na pevný disk, na čipovou kartu atd. Soukromý klíč je aktivem Boba, který si jej musí střežit.
3. Bob distribuuje svůj veřejný klíč do celého světa. Klidně může svůj veřejný klíč poslat Alici po Oskarovi.
4. Alice po obdržení veřejného klíče Boba šifruje zprávu Bobovi jeho veřejným klíčem (VK-B).
5. Bob (příjemce) dešifruje přijatou šifrovanou zprávu svým soukromým klíčem SK-B a získá původní zprávu.



Obrázek 6: Asymetrická šifra [3]

L. Dostálek a M. Vohnoutová dále uvádějí: *„Základní vlastností šifrování na bázi asymetrických algoritmů je skutečnost, že je relativně jednoduché šifrovat text za využití veřejného klíče. Takto zašifrovaný text již ale nelze rozšifrovat stejným klíčem.*

K rozšifrování je potřeba soukromý klíč“ [3]. Asymetrické šifrování se často používá pro zabezpečení přenosu symetrického klíče např. AES. V tomto případě Alice zašifruje vygenerovaný AES klíč veřejným klíčem Boba (VK-B), pošle jej Bobovi a Bob tento klíč dešifruje svým soukromým klíčem (SK-B). Následná komunikace mezi Bobem a Alicí může probíhat na základě symetrické šifry. Výhoda spočívá ve vyšší rychlosti symetrického šifrování oproti asymetrickému.

V asymetrické kryptografii existují 3 základní principy výpočtu algoritmů:

- *Princip faktorizace velkých čísel.* Tato metoda je založena na problému rozložení libovolného celého čísla na součin prvočísel. Tento rozklad je považován za velmi těžkou úlohu. Na jeho základě je postaven algoritmus RSA, kterým se dále zabývá kapitola 2.2.1 RSA.
- *Princip výpočtu diskrétního algoritmu.* Spočívá na principu rozložení přirozeného čísla na mocninu jiného čísla $a = g^x$ v modulární aritmetice dané vhodným modulem *m*. Nalezená mocnina *x*, kterou lze zapsat jako $x = \log_g a$, se pak nazývá diskrétní algoritmus. Na této metodě je postaven algoritmus Diffie-Hellman nebo DSA (Digital Signature Algorithm) [5].
- *Princip výpočtu na bázi eliptických křivek (EC).* Jedná se o analogii kryptografie s veřejným klíčem, ve kterých je modulární aritmetika nahrazena operacemi nad eliptickou křivkou. V. Klíma dále uvádí: „*V současné době pronikly eliptické kryptosystémy do řady světových standardů a staly se alternativou ke "klasickému" RSA i DSA. Mají své výhody zejména v rychlosti a menší náročnosti na hardware i software. (…) Jejich nasazení ovšem brání skutečnost, že "staré" kryptosystémy RSA, DSA, Diffie-Hellman, ElGamal atd. jsou používány, studovány a známy déle a mají vybudovánu infrastrukturu. Proto jsou vývojářům a technologům bližší*“ [6].

2.2.1 RSA

Název RSA je tvořen počátečními písmeny jmen jeho tvůrců Rivest–Shamir–Adleman. Dnes se jedná o nejvíce rozšířenou asymetrickou šifru, která ještě do roku 2000 podléhala patentu v USA. S jejím využitím se nejvíce setkáme u:

- šifrování menšího objemu dat, zejména pak pro zabezpečenou výměnu klíčů
- digitálního podpisu, který je podrobněji popsán v kapitole 2.4 Digitální podpis

T. Doseděl popisuje princip šifrování následovně: „*Před šifrováním je třeba nejprve vygenerovat pár klíčů. Pro tyto účely jsou vytvořena náhodná čísla p a q, která musí být prvočísly. Jelikož generování velkých prvočísel (stovky míst) je velmi složitý problém, situace se řeší tak, že se vygeneruje náhodné číslo, které se následně podrobí testu prvočíselnosti. Tento test spočívá v postupném dělení čísla malými prvočísly, což by mělo ve většině případů odhalit jeho neprvočíselnost. Obě prvočísla jsou vynásobena, dostáváme číslo* $n = p * q$*. Veřejný klíč nyní generujeme jako náhodné číslo e, které nemá žádné společné součinitele s číslem* $(p-1)(q-1)$*. Soukromý klíč vzniká mnohem složitější matematickou operací. Získáme ho podle vztahu* $d = e^{-1} mod((p-1)(q-1))$*. Veřejný klíč je tvořen dvojicí {n, e}, soukromý klíč má dvojici {n, d}. Čísla p a q jsou nyní nepotřebná, můžeme je zničit. Pod slovem zničit je myšleno opravdové kryptograficky dokonalé zničení, lze z nich totiž snadno oba klíče vygenerovat*“ [2].

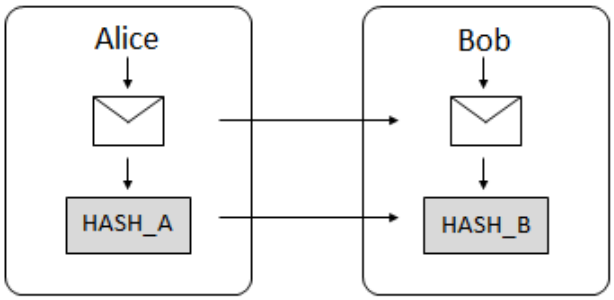
Algoritmus RSA dosud nebyl podroben žádnému úspěšnému útoku. Byl normalizován několika standardizačními organizacemi.

2.3 Digitální otisk

Hash, neboli otisk, je jednocestná funkce, která z libovolně dlouhého textu vytvoří krátký řetězec konstantní délky [3]. Jednocestná funkce znamená, že z vytvořeného řetězce nesmí být možné odvodit původní zprávu. Zároveň musí být splněn požadavek bezkoliznosti. Nesmí nastat situace, kdy dva různé řetězce vytvoří stejný otisk. Typická velikost výsledného textu je 16 bajtů (např. algoritmus MD-5) nebo 20 bajtů (algoritmus SHA-1). Dnes se již algoritmy MD-5 a SHA-1 vesměs považují za slabé, proto se stále častěji setkáváme s novými algoritmy produkujícími ale delší otisky: SHA-224 (otisk dlouhý 28 bajtů), SHA-256 (otisk 32 bajtů), SHA-384 (otisk 48 bajtů) a SHA-512 s otiskem dlouhým 64 bajtů [3].

Výhodou digitálního otisku je jeho malá velikost. Asymetrická kryptografie se tedy nemusí zabývat šifrováním velkých souborů (např. několik megabajtů), zcela stačí, pokud dojde k úspěšnému zašifrování několik stovek bitů [2].

Otisk lze použít i pro zajištění integrity přenášené zprávy. Slouží jako důkaz, že zpráva na cestě od Alice k Bobovi nebyla změněna. Alice v tomto případě neodešle pouze samotnou zprávu, ale data doplní o patu zprávy (trailer) obsahující otisk z textu [3]:



Obrázek 7: Využití otisku jako důkazu integrity zprávy [3]

Bob, poté co přijme zprávu, spočte otisk z přijaté zprávy a porovná svůj výsledek s otiskem ze zápatí přijaté zprávy (tj. s otiskem spočteným Alicí). Pokud se oba otisky shodují, zpráva nebyla cestou změněna. Tento typ důkazu integrity přenášených dat využívají linkové protokoly (např. Ethernet) pro detekci chyb vzniklých poruchami linek [3].

Autenticitu ale nelze v tomto případě zaručit, jelikož případný útočník může změnit původní zprávu a vypočítat z ní nový otisk. Následně ji přepošle příjemci, který otisk zkontroluje stejným způsobem. Otisky se budou i v tomto případě shodovat.

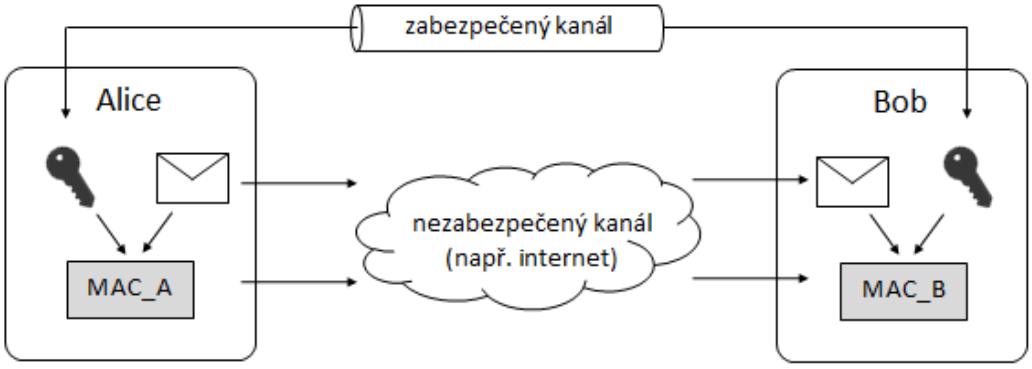
2.3.1 Algoritmus HMAC

HMAC[2] je konkrétní implementací algoritmu MAC[3]. Tyto algoritmy se obecně používají pro zajištění autenticity a datové integrity zasílané zprávy s použitím hashovací funkce v kombinaci s tajným šifrovacím klíčem. Lze jimi zaručit, že při

[2] Keyed-hash Message Authentication Code
[3] Message Authentication Code

přenosu zprávy nedošlo k její manipulaci nebo ke změně. Navíc lze důvěřovat odesílateli zprávy.

Tak jako každý MAC, může být i HMAC použit pro ověření datové integrity a autentizaci zprávy. Na výpočet je možné použít libovolnou iterativní kryptografickou hashovací funkci jako např. MD5, SHA-1 nebo SHA-2 (HMAC-MD5, HMAC-SHA-1, HMAC-SHA2). Bezpečnost výsledného kryptogramu závisí na síle algoritmu, velikosti a kvalitě klíče.



Obrázek 8: Zajištění integrity a autenticity přenášených dat pomocí algoritmu HMAC

Před zasláním zprávy je nutné, aby si Alice vyměnila s Bobem bezpečnostní klíč. Následně Alice vypočte kryptogram *MAC_A* z původní zprávy za použití sdíleného klíče. Alice zasílá Bobovi původní zprávu i vypočtený *MAC_A*. Bob vypočte svůj vlastní *MAC_B* z obdržené zprávy za použití sdíleného klíče. Pokud se výsledky rovnají (*MAC_A = MAC_B*), může Bob přijaté zprávě důvěřovat.

V některých případech lze o algoritmu HMAC uvažovat jako o „symetrickém podpisu“. Označení symetrický podpis není úplně přesné. Na rozdíl od digitálního podpisu totiž nelze zaručit pravost dokumentu („nepopiratelnost“ – non-repudiation), ale pouze jen integritu přenášených dat [3]. To znamená, že Bob by mohl přijatou zprávu sám změnit a spočetl by z ní znovu "symetrický podpis". V tomto případě by Bob disponoval zprávou, kterou Alice nikdy nevytvořila. Navíc by Alice nemohla dokázat, zdali "symetrický podpis" opravdu vytvořila ona nebo jej podvrhl Bob. Tuto problematiku řeší až digitální podpis.

2.4 Digitální podpis

Zatímco šifrování se používá k ochraně dat proti neoprávněnému prozrazení, digitální podpis zajišťuje integritu dat a jednoznačnou identifikaci toho, kdo podpis vytvořil. Digitální podpis tak v informačních a komunikačních technologiích nahrazuje klasický vlastnoruční podpis. Díky funkci integrity lze snadno zjistit, že nedošlo ke změně podepsaného dokumentu od okamžiku vytvoření digitálního podpisu.

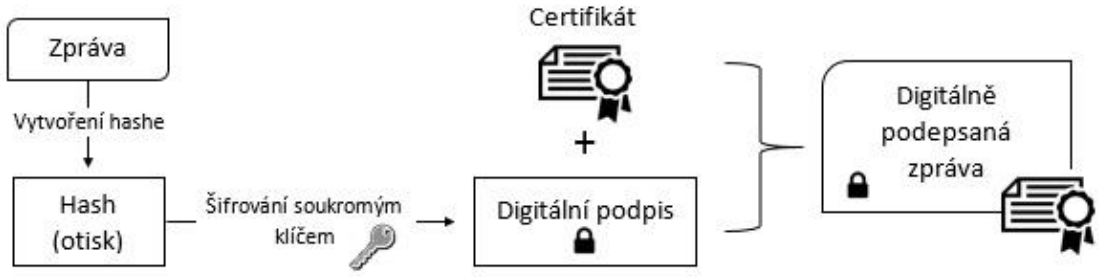
Digitální podpis se vytváří ve dvou krocích [3]:

1. Spočte se otisk (hash) z dokumentu.
2. Výsledný otisk se šifruje soukromým klíčem uživatele, který podpis vytváří. Soukromým klíčem šifrovaný otisk ze zprávy se nazývá digitální podpis zprávy.

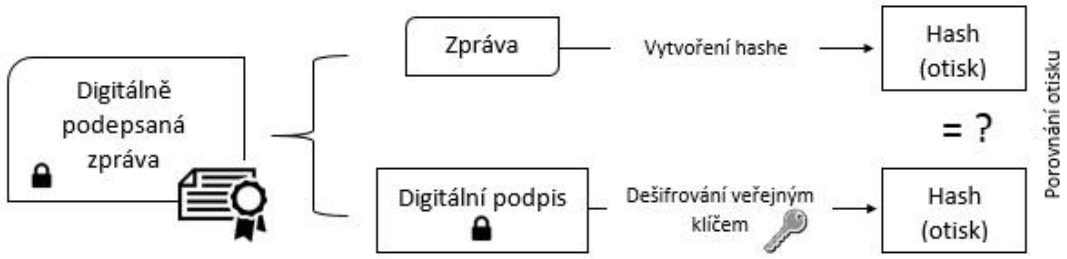
Ověření digitálního podpisu se provede ve třech krocích [3]:

1. Příjemce samostatně spočte otisk z přijaté zprávy.
2. Příjemce dešifruje přijatý digitální podpis veřejným klíčem odesílatele.
3. Příjemce porovná výsledek získaný z bodu 1 s výsledkem získaným z bodu 2. Pokud jsou stejné, pak mohl digitální podpis vytvořit pouze ten, kdo vlastní soukromý klíč odesílatele – tedy odesílatel. A navíc tato skutečnost prokazuje, že zpráva nebyla během přenosu pozměněna, tj. zajišťuje i integritu zprávy.

V tuto chvíli však není možné považovat platný elektronický podpis za důvěryhodný, protože není jisté, kdo je majitelem veřejného klíče, pomocí kterého došlo k matematickému ověření podpisu. V tomto smyslu se hovoří o tzv. přenosu důvěry z důvěryhodné třetí strany na údaj o majiteli veřejného klíče, pomocí kterého došlo k úspěšnému matematickému ověření platnosti elektronického podpisu. K tomu je využíván digitální certifikát (kapitola 2.4.1 Certifikát veřejného klíče), který vydává důvěryhodná certifikační autorita.



Obrázek 9: Tvorba digitálního podpisu [autor]



Obrázek 10: Ověření digitálního podpisu [autor]

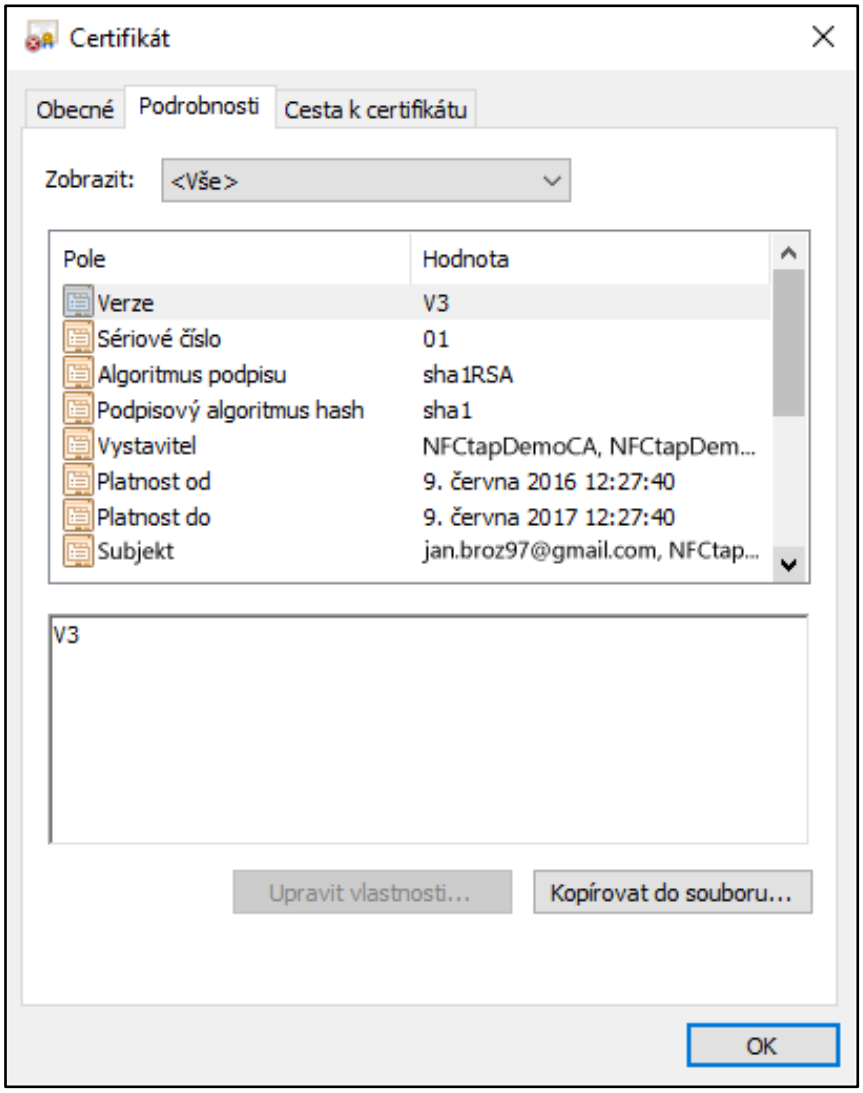
2.4.1 Certifikát veřejného klíče

V některých případech může dojít k situaci, kdy útočník infiltruje databázi veřejných klíčů (např. firemní server veřejných klíčů). V tomto případě je poměrně snadné vyměnit kterýkoliv veřejný klíč v databázi za nově vygenerovaný, k němuž má útočník i klíč soukromý. Následně může pod falešnou identitou tento veřejný klíč podstrčit další osobě, která jej využije k zašifrování zprávy. Útočník pak může tuto zprávu jednoduše dešifrovat, jelikož k danému veřejnému klíči má i klíč soukromý.

Řešením tohoto problému je certifikát veřejných klíčů, jehož princip podrobněji popisuje T. Doseděl: „*Někdo důvěryhodný (důvěryhodná třetí strana – Trusted Third Party, v tomto případě se označuje jako certifikační autorita) stvrdí svým podpisem (digitálním), že daný veřejný klíč patří konkrétní osobě. Nyní už není problém přikládat i verifikovaný veřejný klíč ke každé zprávě. Příjemce nejprve ověří podpis certifikátu. Pokud souhlasí, ověří osobní údaje uvedené o odesílateli v certifikátu. Pokud i tyto údaje souhlasí, může přiloženému veřejnému klíči důvěřovat a použít ho k ověření digitálního podpisu vlastní zprávy*“ [2].

Existuje několik norem definujících strukturu certifikátu (X.509, EDI, WAP apod.). V internetu se využívá norma X.509 popsaná v doporučení RFC-3280 – Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.

2.4.1.1 Obsah certifikátu



Obrázek 11: Zobrazení certifikátu ve Windows [autor]

Soubor certifikát obsahuje následující parametry [3]:

a) *Verze certifikátu* souvisí s tím, je-li certifikát odvozen od normy X.509 verze 1, 2 nebo 3. Dnes se zásadně používají pouze certifikáty verze 3.
b) *Sériové číslo* je definováno jako celé kladné číslo, které musí být jednoznačné v rámci konkrétní certifikační autority.

c) *Algoritmus podpisu* specifikuje algoritmy použité certifikační autoritou pro vytvoření elektronického podpisu certifikátu. Jedná se o dvojici algoritmů. První udává způsob tvorby otisku, zatímco druhý označuje použité šifrování pro otisk.
d) *Platnost* certifikátu od / do konkrétního data (Not Before / Not After).
e) *Vydavatel a předmět.* Vydavatel (Issuer) specifikuje toho, kdo certifikát vydal, tj. certifikační autoritu. Položka předmět (Subject) specifikuje držitele certifikátu.
f) *Veřejný klíč* je sekvencí dvou informací: identifikátorem algoritmu, pro nějž je veřejný klíč určen, a samotným veřejným klíčem.

2.4.1.2 Životní cyklus certifikátu

Certifikát v průběhu času prochází několika následujícími fázemi:

1. *Vytvoření žádosti o certifikát.* Vytvoření žádosti může, ale i nemusí předcházet generování párových dat [3]. Člověk, který chce vydat certifikát svého veřejného klíče, musí nejprve připravit žádost. V ní uvede všechny potřebné identifikační (osobní) údaje, přiloží veřejný klíč a vše podepíše příslušným soukromým klíčem. Žadatel tímto podpisem prokazuje, že je opravdu vlastníkem soukromého klíče k certifikovanému veřejnému klíči.
2. *Vydání certifikátu a jeho případná publikace*. Certifikační autorita zřizuje takzvané registrační autority. Jedná se o místa, kde pověřená osoba dané registrační autority přijímá od zákazníků žádosti o vydání certifikátů. Při této činnosti zároveň ověřuje identitu žadatelů.
3. *Platnost certifikátu*. Poté co byl certifikát vydán, nemusí být ještě automaticky platný. Platnost certifikátu začíná v době uvedené v položce „od“ (Not Before) a končí buď vypršením platnosti nebo odvoláním certifikátu [3].
4. *Vypršení platnosti* certifikátu nastane po uplynutí doby „do“ (Not After) uvedené v certifikátu.
5. *Odvoláním* certifikátu před uplynutím jeho původně deklarované doby platnosti. Certifikát odvolává certifikační autorita zpravidla tím, že identifikaci certifikátu zveřejní na seznamu odvolaných certifikátů (CRL). Odvolaný certifikát se uvádí na všech CRL po dobu jeho původní platnosti

[3]. Odvolání omezuje možné zneužití certifikátu a dochází k němu především ze dvou důvodů: Při změně údajů v certifikátu a při ohrožení soukromého klíče (odcizení atd.).

2.4.2 Certifikační autorita

Organizace, která vydává certifikáty veřejných klíčů, se nazývá certifikační autorita (CA). Musí se jednat o subjekt důvěryhodný, který má u svých uživatelů dostatečnou autoritu. Základním majetkem každé certifikační autority je pár klíčů. Veřejný klíč je zveřejněn (například na internetu), soukromý klíč je pak maximálně střežen. Je největším aktivem certifikační autority. Nesmí nastat situace, kdy by došlo ke ztrátě, případně konfrontaci soukromého klíče certifikační autority. Prakticky by to znamenalo nutnost zneplatnění všech vydaných certifikátů danou certifikační autoritou [2]. U certifikátů vydaných certifikační autoritou platí zásada přenosu důvěry. To znamená, že můžeme věřit informacím uvedeným v digitálních certifikátech, které vydala důvěryhodná certifikační autorita. Pokud je digitální podpis certifikátu platný a důvěřujeme certifikační autoritě, která klíč podepsala, přeneseme důvěru a věříme v důvěryhodnost neznámého veřejného klíče. V České republice působí takzvané kvalifikované (důvěryhodné) certifikační autority, které mohou vydávat kvalifikované certifikáty. Tyto certifikační autority jsou akreditovány Ministerstvem vnitra České republiky. Kvalifikované certifikáty jsou dle Zákona o elektronickém podpisu uznávány v rámci komunikace se státními institucemi České republiky. Na internetu také působí mnoho komerčních certifikačních autorit, které mají své veřejné klíče uloženy přímo v internetovém prohlížeči. Tím uživateli usnadňují rozhodování o důvěryhodnosti webových serverů, ke kterým se připojuje.

2.5 Autentizace a autorizace

Autentizace a autorizace souvisí s ochranou logického přístupu k datům. Snaží se zabránit tomu, aby k datům měl přístup uživatel, který nemá dostatečná přístupová práva.

Autentizace znamená ověření identity uživatele nebo entity v systému za účelem řízení přístupu ke zdrojům a objektům v systému.

Po úspěšné autentizaci nastává proces autorizace, což je souhlas, schválení přístupu či provedení konkrétní operace daným subjektem.

Prvním krokem u autentizace je získání autentizační informace od uživatele. Pokud je tato informace předepsaným způsobem použita, prokazuje identitu uživatele, který ji předal informačnímu systému. Autentizaci uživatele lze obecně provést na základě prokázání:

1. *Uživatel něco zná.* V současné době je stále velice rozšířená metoda spočívající v zadávání údajů z klávesnice (heslo, PIN). Hlavní nevýhoda spočívá v nutnosti uživatele pamatovat si heslo a s tím spojené jeho zapomínání, zneužití útočníkem atd.
2. *Uživatel něco má*. Uživatel vlastní zařízení (čipovou kartu, autentizační kalkulátor nebo mobilní telefon), které využije při autentizaci.
3. *Uživatel něčím je*. Pro autentizaci se využívají biometrické vlastnosti, jako jsou otisky prstů, struktury oční sítnice či duhovky, tvar obličeje a jiné.

3 Analýza čipových karet

Pro zajištění spolehlivé a bezpečné autentizace uživatele se často využívají čipové karty. Ty zpravidla obsahují základní údaje o uživateli a některé typy karet poskytují i bezpečné úložiště soukromého klíče.

Existuje několik způsobů jak dělit čipové karty. Bankovní karty se často dělí na kreditní a debetní, SIM karty zase dle jejich velikosti (mini, mikro, nano), nebo se karty mohou naopak dělit dle způsobu využití. Nejprve je ovšem důležité zmínit konkrétní normy ISO / IEC, které, mimo jiné, předepisují vlastnosti karet.

3.1 Norma ISO / IEC

ISO (Mezinárodní organizace pro normalizaci) a IEC (Mezinárodní elektrotechnická komise) tvoří specializovaný systém celosvětové normalizace [7]. Národní orgány, které jsou členy ISO nebo IEC, se podílejí na vypracování mezinárodních norem prostřednictvím technických komisí zřízených příslušnou organizací.

Existují tři nejdůležitější normy popisující čipové karty:

- *ISO/IEC 7816 Karty s integrovanými obvody s kontakty*
- *ISO/IEC 14443 Bezkontaktní karty s integrovanými obvody - Karty s vazbou nablízko*
- *ISO/IEC 15693 Bezkontaktní karty s integrovanými obvody - Karty s vazbou na dálku*

3.2 Dělení karet

Čipové karty lze rozdělit dle způsobu práce s daty, komunikace s okolím a dle použitého operačního systému:

3.2.1 Způsob práce s daty

3.2.1.1 Paměťová karta

Její data jsou ukládána v paměti typu EEPROM a je používána převážně v aplikacích, kde není vyžadována vysoká bezpečnost. Karty obsahují paměť, ale žádnou inteligenci. Obsah je dán z výroby. Přístup do paměti je řízen vnitřní

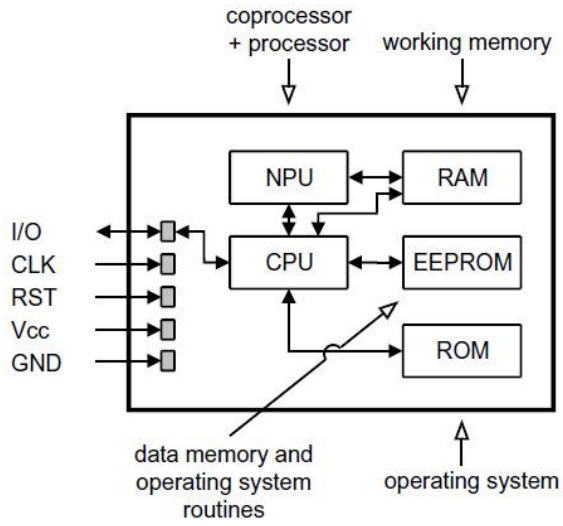
logikou a v některých případech obsahuje pouze ochranu proti zápisu nebo vymazání dat. Paměťové karty jsou většinou určeny pro specifické účely a jejich cena je velice nízká. V minulosti se často jednalo o předplacené telefonní karty.

3.2.1.2 Paměťová karta s autentizační logikou

Jedná se o vyspělejší paměťovou kartu, jejíž bezpečnost proti padělání je zvýšena požadavkem na vložení tajného kódu, který potvrzuje právo na přístup k datům uloženým v paměti [8].

3.2.1.3 Mikroprocesorová karta

Mikroprocesorová karta, jak již název napovídá, obsahuje aktivní mikroprocesor. Ten se skládá ze čtyř součástí: paměť ROM, EEPROM, RAM a I/O rozhraní. Paměť ROM obsahuje operační systém, který je nesmazatelně vložen již v průběhu výroby a obsah paměti již nemůže být změněn. Oproti tomu paměť EEPROM je elektricky mazatelná paměť, která slouží k ukládání a odkládání dat při běhu operačního systému. Paměť EEPROM je pouze dočasná, všechna data jsou proto smazána v případě přerušení napájení. Rozhraní I/O pak slouží pro komunikaci s okolím [9].



Obrázek 12: Architektura mikroprocesorové karty [9]

Do některých architektur je navíc přidán koprocesor, který zajišťuje matematické výpočty pro kryptografické algoritmy (např. RSA). Tím se celkový čas pro výpočet daného algoritmu snižuje na dobu několika stovek mikrosekund.

V minulosti bylo standardem využívat 8 bitový mikroprocesor pracující na frekvenci 5 MHz. Dnes se ovšem běžně osazují 16 bitové, někdy dokonce i 32 bitové mikroprocesory s architekturami jako je CISC nebo RISC.

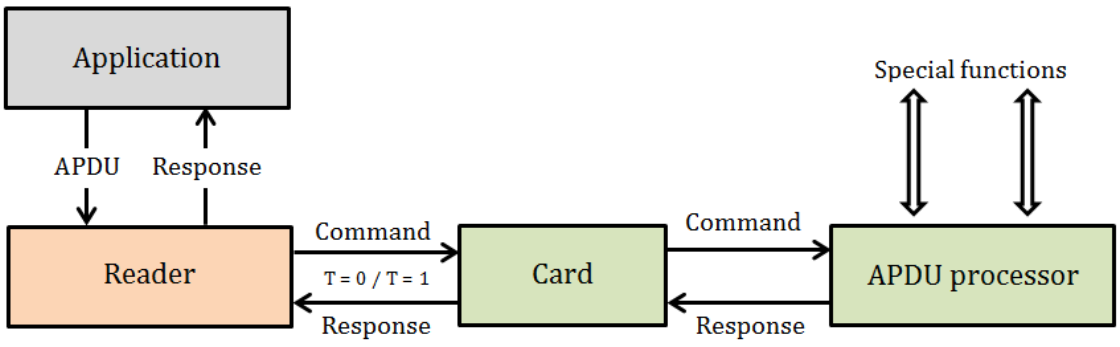
3.2.2 Komunikace s okolím

Při komunikaci s terminálem nebo čtečkou je třeba rozlišovat, zda se jedná o kontaktní či bezkontaktní kartu.

3.2.2.1 Kontaktní karty

Kontaktní karty jsou již od 80. let minulého století vyráběny dle normy ISO 7816, která, mimo jiné, předepisuje mechanické a elektrické provedení. Základním prvkem těchto karet jsou pozlacené kontaktní plošky, skrze které jsou karty napájeny a rovněž zajišťují komunikaci mezi kartou a čtečkou. Jedná se vždy o jednosměrnou komunikaci směřující z karty do terminálu nebo naopak z terminálu do karty. Jelikož zde existuje pouze jeden komunikační kanál, karta a čtečka se musí o tuto jednu linku dělit. Tento režim se proto nazývá polo-duplexní (half-duplex). V polo-duplexním režimu tedy komunikaci vždy začíná terminál a karta vzápětí reaguje. Jedná se o komunikaci příkaz – odpověď.

Komunikaci mezi kartou a terminálem si lze představit dle následujícího schématu:



Obrázek 13: Komunikace mezi čtečkou a kartou [10]

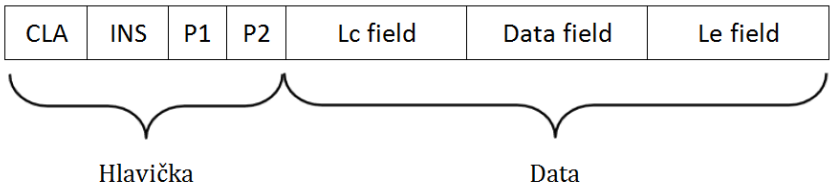
Komunikace vychází z OSI modelu a obsahuje fyzickou, datovou a aplikační vrstvu. Vzhledem k zaměření této práce bude podrobněji rozepsána pouze aplikační vrstva.

Aplikační vrstva

Pro komunikaci mezi čtečkou a kartou na této nejvyšší komunikační vrstvě se používají takzvané APDU příkazy (Application Protocol Data Unit). Rozlišujeme dva druhy ADPU příkazů:

a) Command APDU – příkazy zasílané ze čtečky do karty
b) Response APDU – odpovědi zasílané z karty do čtečky.

ad a) Command APDU. Příkaz APDU se skládá z hlavičky a dat:



Obrázek 14: Struktura příkazu APDU

Hlavička obsahuje čtyři elementy (4 x 1 bajt), konkrétně:

- *CLA* - (Class). Jednoznačně identifikuje aplikaci a její specifický soubor příkazů (command set). Např. „0X“ se používá pro příkazy vycházející ze standardu ISO/IEC 7816-4/7/8; „A0“ pro příkazy vycházející ze standardu GSM 11.11; „8X“ pro proprietární aplikace atd.
- *INS* - (Instruction byte). Určuje zvolený příkaz. Těch je např. dle ISO7816 definováno hned několik viz Tabulka 1: Základní příkazy obsažené v INS (APDU).
- *P1* - Podrobněji specifikuje příkaz INS.
- *P2* - Podrobněji specifikuje příkaz INS.

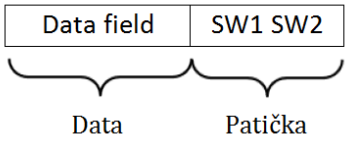
Následuje řetězec dat (0 – 255 bajtů):

- *Lc field* - (length command). Určuje počet bajtů v poli.
- *Data field* - Zasílaný řetězec bajtů proměnlivé délky.
- *Le field* - (length expected). Určuje předpokládaný počet bajtů v poli v odpovědi karty.

Tabulka 1: Základní příkazy obsažené v INS (APDU)

Příkaz	Funkce	Instrukce (INS)	Standard
SELECT (FILE)	Zvolí soubor nebo adresář	"A4"	ISO/IEC 7816-4
CREATE FILE	Vytvoří nový soubor	"E0"	ISO/IEC 7816-4
GET CHALLENGE	Žádost o vygenerování náhodného čísla	"84"	ISO/IEC 7816-4
VERIFY	Kontrola zaslaných dat, např. PINu	"20"	ISO/IEC 7816-4, EMV

ad b) Response APDU. Odpověď APDU se skládá z dat a patičky:



Obrázek 15: Struktura odpovědi APDU

Datová část, na rozdíl od patičky, není povinná. Délka datové části u odpovědi je předem definována v poli *Le* v předcházejícím příkazu APDU. Návratové kódy v patičce jsou rovněž předem definovány. Existuje celkem 6 různých variant odpovědí. Nejběžnější je „90 00“ značící úspěšné provedení příkazu (SW_SUCCESS). Z dalších je to např. „63 xx“, který znamená chybu [9].

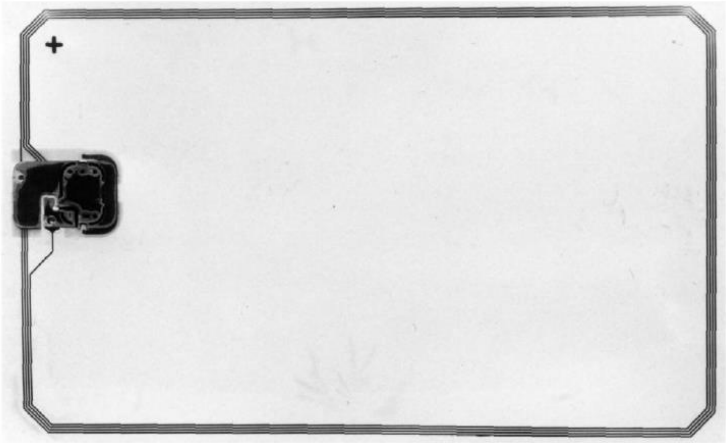
3.2.2.2 Bezkontaktní karty na bázi RFID

RFID[4] technologii je na trhu věnován stále větší prostor. V porovnání s kontaktní technologií přináší RFID hned několik výhod. Mezi ty nejvýraznější patří téměř nulové náklady na údržbu (nedochází zde k fyzickému kontaktu mezi kartou a terminálem) a z toho plynoucí delší životnost produktu. Jejich obliba v posledních letech neustále roste. O nárůstu obliby využívání bezkontaktních karet hovoří statistiky VISA a MasterCard, které nyní vydávají platební karty s bezkontaktním rozhraním. Nárůst bezkontaktních plateb je očekáván z dosavadních 4,32 miliard USD v roce 2013 na 9,88 miliard do roku 2018 [11].

Aktuálně je v oběhu přibližně 250 milionů bezkontaktních platebních karet [12].

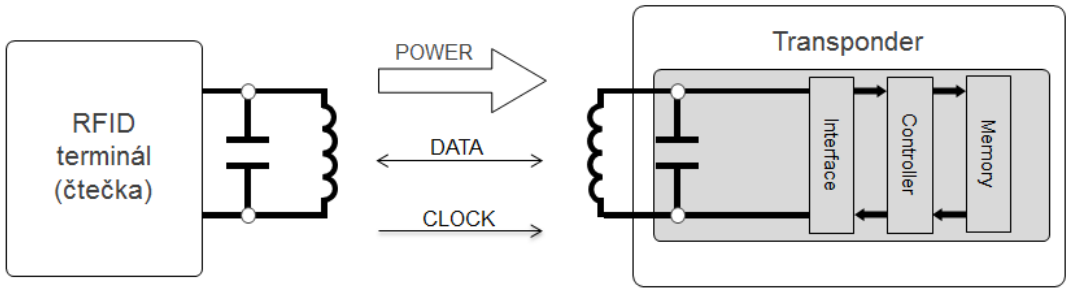
[4] Radio Frequency Identification - identifikace na rádiové frekvenci

Pod pojmem RFID se skrývá hned několik technologií sloužících primárně pro identifikační účely. Známé jsou systémy na 2,4 GHz a 5,8 GHz (sloužící převážně pro identifikaci předmětů na vzdálenosti 10 a více metrů), dále pak systémy na kmitočtech 433 MHz a 868 MHz (915 MHz) – k identifikaci předmětů na vzdálenosti do 5 metrů. Pro potřeby této práce budou předmětem zkoumání pouze identifikátory, které slouží pro identifikační účely s dosahem maximálně do 10 cm. Jedná se tedy o systémy pracující na frekvenci 125 kHz (LF) a 13,56 MHz (HF). Nejčastější podobou těchto identifikátorů je plastová karta se zalisovanou anténou a čipem.



Obrázek 16: Vnitřek bezkontaktní karty (čip + anténa) [9]

Přenos „dat“ probíhá bezdrátově pomocí elektromagnetického pole mezi kartou a terminálem. Jako nosná vlna je nejčastěji použita frekvence 125 kHz (LF) nebo 13,56 MHz (HF). Čtecí vzdálenost je závislá na provedení antén na obou stranách zařízení. Ve výsledku se běžně dosahuje čtecí vzdálenosti do 10 cm. Technologie RFID pracuje podle následujícího principu:



Obrázek 17: Princip funkce RFID [13]

Čtečka nejprve vysílá na svém nosném kmitočtu elektromagnetickou vlnu, která je přijata anténou transpondéru. Indukované napětí vyvolá elektrický proud, který nabíjí kondenzátor v transpondéru. Uložená energie je použita pro napájení logických a rádiových obvodů transpondéru. Tím je zajištěna dostatečná energie pro vyslání odpovědi čtečce.

Podobně jako u kontaktních karet, i zde lze technologii dělit dle způsobu práce s daty na karty paměťové, paměťové s autentizační logikou a karty mikroprocesorové.

Dalším dělením může být hledisko bezpečnosti:

a) *Minimální* – Kdokoliv může vyčítat data. Zápis může být zablokován.
b) *Nízká* – Čip v sobě implementuje určitou formu autentizačního mechanismu. Paměť tak bývá často chráněná heslem.
c) *Střední* – Čip v sobě implementuje šifrovací algoritmus využitý k autentizaci a šifrování dat. Algoritmus může být proprietární např. CRYPTO1 u karet Mifare Standard (1K a 4K) nebo např. obecně známý standard DES.
d) *Vysoká* – Čip v sobě implementuje několik symetrických a asymetrických obecně známých algoritmů pro autentizaci, šifrování, digitální podpis atd.

Velice oblíbeným a často používaným proprietárním systémem je bezkontaktní čip Mifare od výrobce NXP, který používá proprietární šifrovací algoritmus CRYPTO1 (dnes již bezpečnostně prolomený). Tato bezpečnostní slabina vede řadu vydavatelů karet k integraci její novější verze Mifare DESfire EV1.

Na standardech RFID je založena technologie NFC[5], kterou se podrobněji zabývá následující kapitola.

[5] Near Field Communication

4 Platforma Android a rozhraní NFC

Android vznikl v roce 2003 a po odkoupení firmou Google zaznamenal prudký nárůst podílu na trhu mezi operačními systémy. Dnes do vývoje přispívají milióny lidí a řada firem, které zajišťují hardwarovou kompatibilitu [14]. V roce 2007 byla založena nezisková skupina Open Handset Alliance s cílem definovat otevřené standardy pro mobilní zařízení. Android je open source projekt založený na jádře Linux a postaven na procesorové architektuře ARM. Dnes jsou ale již známé první modifikace na architekturu x86 (zejména pro procesor Intel Atom).

Vývoj aplikací probíhá převážně v programovacím jazyce Java. Aplikace jsou následně interpretovány pomocí Dalvik Virtual Machine.

Android využívá všechny bezpečnostní výhody nabízené Linuxovým jádrem. Linux je víceuživatelský systém, jehož jádro dokáže oddělit jednotlivé uživatelské zdroje a procesy. Uživatel tak nemá přístup k souborům jiného uživatele (pokud explicitně tento přístup nepovolí) a každému procesu je přiřazena identita (user a group ID, označené jako UID a GID) uživatele, který proces spustil. Android tento způsob izolace rovněž využívá, nicméně na uživatele nahlíží rozdílným způsobem než Linux. V Linuxu je číslem UID označen uživatel, který je zalogován v systému a může vykonávat jednotlivé příkazy nebo je jím označena systémová služba (daemon) běžící na pozadí. Android byl původně vyvíjen pouze pro mobilní telefony a ze začátku nebyl ještě požadavek na více uživatelů v rámci jednoho zařízení. Z tohoto důvodu se číslem UID označuje aplikace, nikoliv uživatel [15]. Každá aplikace tak může mít své vlastní oprávnění, což tvoří základ pro tzv. Sandboxing.

4.1 Aplikační sandbox

Smyslem sandboxu je izolovat proces a data aplikace od ostatních aplikací. Tento způsob izolace zajišťuje operační systém na úrovni jádra. Android automaticky přiřazuje aplikaci jedinečné UID (app ID) již v průběhu instalace a dále pod tímto UID spouští její proces. Navíc je každé aplikaci přiřazeno vlastní datové úložiště (adresář), kam může nahlížet, číst a zapisovat soubory pouze ona. Adresář se nachází v */data/data*, následuje jméno balíčku (např. *com.google.android.email*). Každá aplikace má v rámci svého sandboxu přístup

pouze k omezenému množství systémových zdrojů. Tento způsob omezení může být dále spravován pomocí systému oprávnění.

4.2 Ukládání citlivých dat

Android disponuje několika druhy úložišť. Zabezpečení ukládaných dat je však plně v kompetenci vývojáře aplikace.

4.2.1 Vnitřní (interní) úložiště

Každá aplikace používá své vlastní interní úložiště a ostatní aplikace do něho nemají přístup. To ovšem neplatí pro aplikace s root oprávněním. V tomto případě mohou přistupovat i do úložišť ostatních aplikací. Root oprávněním se podrobněji zabývá kapitola 4.3 Root zařízení.

4.2.2 Vnější (externí) úložiště

Soubory, které jsou vytvořeny a uloženy na externím úložišti (nejčastěji SD karta), jsou dostupné všem aplikacím. Protože SD karta může být z telefonu vyjmuta a data mohou být modifikovány libovolnou aplikací, externí úložiště proto nelze doporučit pro ukládání citlivých dat. Některé z posledních typů mobilních telefonů navíc slotem pro paměťovou kartu nedisponují.

4.2.3 Způsoby uložení citlivých dat

Pokud jsme nuceni uložit citlivá data v paměti telefonu, je vhodné data zašifrovat ještě před samotným uložením. K tomu lze využít třídu Cipher, která disponuje známými kryptografickými algoritmy jako je například AES nebo RSA. Otázkou ale vždy zůstává, kde bezpečně uložit klíče.

Ideální úložiště poskytuje hardwarový Secure Element (SE), který ale bohužel není pro vývojáře běžně dostupný na každém zařízení. Tímto úložištěm se podrobněji zabývá kapitola 4.4.1.3 Režim Card Emulation (emulace karty).

Jednou z možností je proto vložit klíče přímo do zdrojového kódu aplikace. Existuje ale řada utilit, které dokáží soubor *apk*[6] zpětně dekompilovat a získat tak přístup ke zdrojovému kódu [16]. Tuto metodu proto nelze doporučit.

[6] Android application package

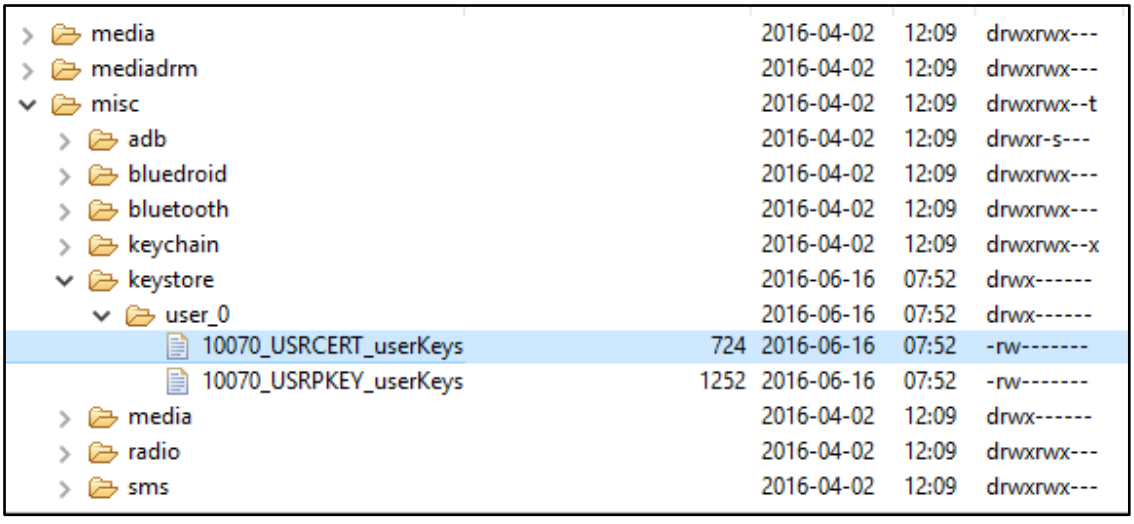
Druhou možností je nechat uživatele zadávat pin nebo heslo při každém otevření aplikace a z tohoto vstupu následně odvodit klíč pomocí vhodného algoritmu (např. PBKDF2 za využití náhodné soli). Nevýhoda spočívá v tom, že uživatel musí zadávat heslo pokaždé, dojde-li k využití klíče. Tento způsob může proto být pro některé uživatele nepohodlný.

Třetí možností je ukládat klíče zcela mimo mobilní zařízení skrze zabezpečený kanál (TLS/SSL) do vzdáleného úložiště, kde server může zastoupit úlohu Secure Elementu. Je bezpodmínečně nutné, aby server byl zajištěn dostatečnou ochranou proti neoprávněnému přístupu k uloženým datům. Nevýhodou tohoto řešení je nutnost mít v průběhu transakce dostupné internetové připojení.

Další možností je využít bezpečné softwarové úložiště s názvem Android Keystore. Toto úložiště klíčů bylo nakonec zvoleno i v samotné implementaci aplikace platebního systému.

4.2.3.1 Android Keystore

Android již od své verze Donut 1.6 poskytuje přístup k softwarovému uložišti bezpečnostních klíčů s názvem Android Keystore. Až do verze Ice Cream Sandwich (4.0) bylo toto úložiště využíváno pouze pro účely VPN a Wi-Fi konektivity, zejména pro ukládání soukromých klíčů a certifikátů. Přístup přes veřejné API nebyl možný. Tento přístup byl umožněn až od verze 4.0 a každá další verze pak toto API vylepšovala. Od verze Android 4.3 může mít každá aplikace své vlastní úložiště klíčů, které se nachází v adresáři*: /data/misc/keystore/*.



Obrázek 18: Úložiště Android Keystore [autor]

V tomto případě se každé jméno skládá z UID aplikace (1000 označuje systém), typu certifikátu (certifikát CA, uživatelský certifikát, soukromý klíč) a jména (alias). Data uložená v tomto úložišti jsou šifrována pomocí 128 bitového AES master klíče v modu CBC. Samotný master klíč je rovněž šifrován 128 bitovým AES klíčem, který je odvozen ze zámku obrazovky. Výpočet je prováděn pomocí PBKDF2 algoritmu s 8192 iteracemi a náhodně generovanou 128 bitovou solí. Tento způsob šifrování zajišťuje dostatečnou bezpečnost uložených klíčů [15]. Získá-li uživatel přístup k uloženým klíčům, např. pomocí root oprávnění, stále bude muset získat heslo zámku obrazovky. Tento způsob útoku je vzhledem k počtu 8192 iterací a využívání náhodné soli výpočetně i časově velice náročný, ale nikoliv nemožný. Náhodná sůl zamezuje použití slovníkových útoků.

I v tomto případě je bezpečnost master klíče zcela závislá na síle hesla nebo pinu. Požadovanou délku bohužel nejsme schopni ovlivnit při programování aplikace. Android například standardně vyžaduje minimální délku pinu čtyři numerické znaky. Vyžadování delšího pinu nebo hesla se speciálními znaky uživatele pouze odrazuje od užívání aplikace.

Při zadávání pinu obrazovky je vypočten hash (kombinace SHA-1 a MD5) s náhodnou 64 bitovou solí. Pokud se tato hodnota rovná uložené hodnotě, je telefon odemčen. Hash je většinou uložen jako hexadecimální hodnota v */data/misc/password.key.*

Získá-li případný útočník root oprávnění k telefonu, je schopen metodou brute force tento čtyřmístný pin prolomit za poměrně krátkou dobu. Útok navíc ulehčuje fakt, že zmíněná 64 bitová sůl je společně s dalšími informacemi uložena v souboru */data/system/locksettings.db.* K tomuto souboru sice mají přístup pouze systémové aplikace (přístup vyžaduje oprávnění *ACCESS_KEYGUARD_SECURE_ STORAGE*), nicméně v případě root oprávnění i tato ochrana může být prolomena.

Verze Android 4.3 přidala možnost skrze stejné API vložit klíče do Hardware-Backed úložiště, je-li součástí procesoru zařízení. Toto hardwarové úložiště klíčů se nazývá Trusted Execution Environment (TEE) a je standardizováno asociací GlobalPlatform. Všechny kryptografické operace jsou v tomto případě prováděny v chráněném prostředí mimo hlavní OS. Klíče jsou uloženy šifrovaně a nikdy toto úložiště neopustí. Tím získáváme mnohem vyšší bezpečnost než u softwarové varianty. Nicméně přístup ke klíčům je řešen na základě UID. Pokud tedy útočník

získá root oprávnění, je schopen dané klíče použít, nikoliv ale vyjmout a použít v jiném zařízení.

Je také třeba podotknout, že pro získání root oprávnění je nejprve nutné odemknout bootloader zařízení. Toto odemčení umožní vložit jinou verzi systému Android, případně jiný operační systém. Odemknutí bootloaderu ovšem často znamená smazání uživatelského oddílu, čímž je zajištěno, že případný škodlivý OS image nemůže získat přístup ke stávajícím datům uživatele.

4.3 Root zařízení

Jak již bylo mnohokráte zmíněno, systém Android je založen na jádře Linuxu. Z tohoto důvodu se rootování podobá získání oprávnění správce (superuser) v Linuxu nebo v jiném UNIXovém systému. Uživatel tím získá kompletní administrátorská práva k telefonu a může tak provádět operace, které byly před odemknutím telefonu blokovány (měnit vzhled UI, provádět přetaktování procesoru, instalovat kompletně nový OS atd). Na druhou stranu root oprávnění snižuje bezpečnost zařízení a zvyšuje riziko potenciálního útoku. Útočník s root oprávněním může teoreticky spustit odposlech klávesnice, sledovat emailovou nebo SMS komunikaci apod. U většiny výrobců mobilních telefonů nelze uplatnit nárok na záruční opravy, pokud byl proveden root zařízení. Většina uživatelů proto provádí tuto operaci na vlastní riziko.

4.4 Technologie NFC

NFC je bezdrátová bezkontaktní technologie propojující dvě zařízení na krátkou vzdálenost několika centimetrů. Pracuje na kmitočtu 13,56 MHz a vychází z technologie RFID zahrnující ISO/IEC 14443 a FeliCa. Tyto standardy jsou součástí normy ISO/IEC 18092 definovaném neziskovou organizací NFC Forum, jež byla založena v roce 2004 firmami Nokia, Philips a Sony.

Ze dvou komunikujících stran je vždy jedna v roli iniciátora a druhá vystupuje jako příjemce. Role iniciátora spočívá v zahájení komunikace a následné správě přenosu dat. Na fyzické vrstvě protokol dále rozlišuje mezi aktivním a pasivním komunikačním režimem. V aktivním režimu využívají obě komunikující strany svou energii pro generování radiofrekvenčního pole, zatímco v pasivním režimu

pouze iniciátor generuje radiofrekvenční pole a příjemce využívá jeho energii pro svou odpověď [17].

4.4.1 Režimy přenosu dat

Zařízení NFC může vystupovat ve třech základních rolích při režimu přenosu dat:

4.4.1.1 Režim Read/write (čtení zápis)

Režim čtení/zápis umožňuje mobilnímu telefonu nahrávat a číst data transpondérů (tagů). Transpondéry mohou být aktivní i pasivní. Tento režim komunikace vychází ze standardů ISO/IEC 14443 (typu A či B) a FeliCa (JIS X 6319) [14]. Celý proces komunikace spočívá pouze v zápisu nebo čtení dat z/do pasivního čipu, tzv. NFC tagu. NFC tag je v obou případech napájen elektromagnetickým polem iniciátora.

4.4.1.2 Režim peer-to-peer

Je definován ve standardu NFCIP-1 a je podobný režimu P2P z počítačových sítí založených na sadě protokolů TCP/IP. Tento režim je vhodný pro vzájemnou výměnu dat, kontaktů či textových zpráv. Na mobilních telefonech s operačním systémem Android je na tomto komunikačním protokolu založena funkcionalita přenosu zpráv Android Beam.

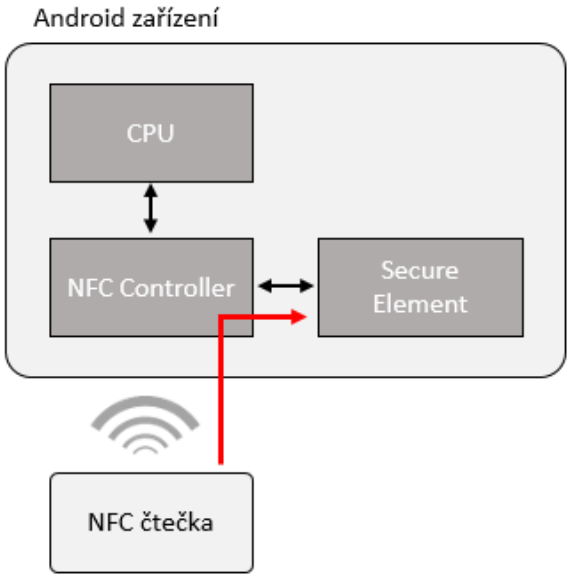
4.4.1.3 Režim Card Emulation (emulace karty)

Mobilní telefon s rozhraním NFC se v rámci tohoto režimu začne chovat jako běžná čipová karta a je proto plně kompatibilní se standardem bezkontaktních karet na bázi ISO/IEC 14443 typ A, B a FeliCa. Pro zahájení přenosu dat musí uživatel svůj telefon vložit do radiového pole čtečky. Telefon lze použít pro aplikace jako je jízdenka, vstupenka, debetní/kreditní karta nebo libovolná forma autentizačního faktoru ve formě pasivního čipu. V tomto režimu musí začít komunikaci čtecí zařízení.

Secure Element (SE)

Režim emulace karty předpokládá bezpečné uložení citlivých dat (např. bezpečnostní klíče, čísla kreditních karet) na straně mobilního telefonu. K tomu lze

využít úložiště zvané Secure Element (SE). Secure Element je čip splňující přísná bezpečnostní kritéria, má své CPU, RAM, ROM, EEPROM, I/O porty a často dodatečný kryptoprocesor pro zajištění kryptografických funkcí (RSA, AES, 3DES atd). Míra jeho bezpečnosti je proto srovnatelná s bezpečností čipových karet.



Obrázek 19: Režim emulace karty s využitím Secure Elementu [18]

Při zahájení komunikace směřuje NFC kontroler všechna obdržená data přímo do Secure Elementu, který nezávisle pokračuje v komunikaci s NFC čtečkou (terminálem). Samotná aplikace proto nemá žádný vliv na probíhající komunikaci, může však po skončení transakce informovat uživatele o jejím výsledku [18].

Ve spojitosti s mobilním telefonem se lze setkat hned s několika formami provedení SE:

1) *SE integrovaný na DPS telefonu.* Čip je na plošný spoj telefonu vložen již v procesu výroby. Prvním mobilním telefonem s integrovaným SE byl Nexus S od společnosti Google. Výrobcem použitého čipu PN65N je společnost NXP. Od roku 2013 však výrobci mobilních telefonů od integrování SE na DPS telefonu postupně upouští [15].
2) *Externí SE.* V telefonu se použije speciální SD nebo microSD karta, která má implementovaný SE. Na trhu se ale v poslední době objevují zařízení, která slot na SD kartu vůbec neobsahují. Navíc by aplikace mohla být určena pouze pro ty uživatele, kteří by měli zařízení se speciální SD kartou.

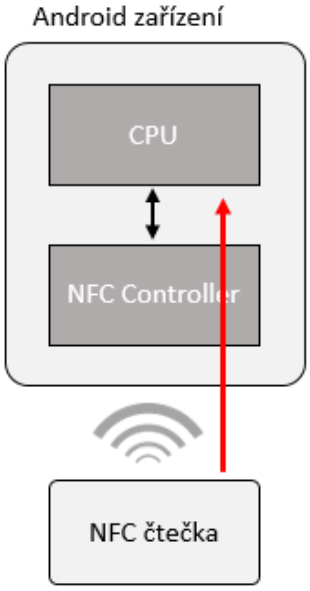
3) *SE na kartě SIM.* Každý telefon vyžaduje pro svou hlavní funkčnost SIM kartu, dnes častěji označovanou jako UICC (Universal Integrated Circuit Card), která je vhodným místem pro uložení Secure Eementu. Secure Element může být v tomto případě použit nejenom pro telekomunikační účely, ale zároveň může svou funkcionalitu nabídnout pro aplikace dalších poskytovatelů – bankovní, věrnostní, přístupové a jiné karty. Nevýhoda spočívá v tom, že SIM kartu vydává operátor a řídí tak přístup do jejího Secure Elementu. Uživatel, který chce platit mobilním telefonem, musí nejprve požádat svého operátora o SIM kartu s bankovní aplikací. Ta je navíc často svázána pouze s konkrétní bankou. Pro vývojáře je další nevýhodou absence API v systému Android, které by řídilo přímý přístup do Secure Elementu. Veškerá komunikace probíhá přes RIL (Radio Interface Layer), který zajišťuje komunikaci mezi GSM modulem telefonu a SIM kartou. Existují však pomocné API, které je třeba nejprve přidat do systému, a na jejich základě pak do SE přistupovat. Jednou z takových API je Smart Card API[7] z projektu SEEK for Android. I v tomto případě ale musí být umožněna přímá podpora ze strany výrobce zařízení.

Přístup do Secure Elementu je tedy obecně řešen buď pomocí proprietárního rozhraní plynoucího od výrobce čipu, nebo standardizovaným rozhraním jako je již zmíněný Smart Card API. Ve výsledku je však pro běžného vývojáře nahrání vlastní aplikace na SE téměř nemožné. Důvodem je zejména neznalost klíčů a snaha o zajištění maximální kompatibility mezi jednotlivými zařízeními.

4.4.1.4 Host-based Card Emulation

Host-based Card Emulation (HCE) je nová metoda emulace čipových karet, která ke své činnosti přímo nevyžaduje Secure Element [18].

[7] Smart Card API je implementací specifikace Open Mobile API, kterou ve svých telefonech využívá např. společnost Samsung.



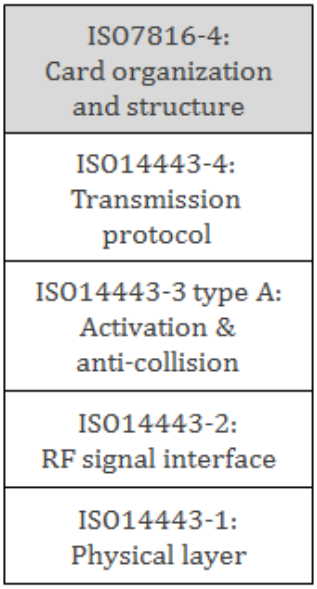
Obrázek 20: Režim Host-based Card Emulation (HCE) [18]

V režimu HCE jsou veškerá data a komunikace v pásmu ISO-DEP směrována přímo na procesor telefonu, na kterém běží aplikace emulující kartu. Do příchodu Androidu verze 4.4 byla emulace karty dostupná pouze na některých platformách BlackBerry 7 a na distribuci CyanogenMod [19].

V roce 2013 Google oficiálně uvádí novou verzi Android 4.4 Kitkat. Emulace karty je zde zpřístupněna pod API s názvem Host-based Card Emulation (HCE). Tato implementace umožňuje aplikaci být skrze rozhraní NFC v přímé interakci se čtečkou bez nutnosti využívat Secure Element [18]. Obcházení Secure Elementu sebou ovšem nese řadu bezpečnostních rizik. Karetní operace probíhající v mobilní aplikaci nemohou nadále těžit ze zabezpečeného úložiště, které poskytovalo SE. Vývojář je tak postaven před poměrně složitý úkol s cílem najít bezpečné úložiště pro citlivá data aplikace. Bezpečným úložištěm se podrobněji zabývala kapitola 4.2.3 Způsoby uložení citlivých dat. Jako nejvhodnější úložiště bezpečnostních klíčů byl nakonec vybrán Android Keystore.

V principu lze režimem HCE emulovat řadu existující typů karet, které podporují protokol ISO/IEC 14443-4. Přesněji řečeno – lze emulovat pouze ty karty, které využívají aplikační vrstvu dle normy ISO/IEC 7816-4. Tato norma definuje APDU příkazy sloužící pro komunikaci mezi terminálem a kartou. Z běžně dostupných karet, používaných např. pro městskou hromadnou dopravu, lze

v režimu HCE emulovat například typ karty Mifare DESfire. Tato karta musí navíc pracovat v protokolu dle ISO/IEC 7816-4, který dovoluje volit aplikaci dle AID[8]. Kartu Mifare DESfire není možné emulovat, pracuje-li ve svém nativním protokolu. V tomto případě příkazy APDU dle ISO/IEC 7816-4 nelze použít. Podobná situace je se staršími kartami Mifare Classic a Ultralight. Ty rovněž nepoužívají aplikační vrstvu dle normy ISO/IEC 7816-4.



Obrázek 21: Protokol HCE [18]

Jiná situace je u bankovních karet, které vycházejí ze standardu ISO/IEC 7816-4. Bankovní karty (jejich bezkontaktní části) mohou být emulovány mobilním telefonem, navíc lze použít stávající infrastrukturu platebních terminálů. Mobilní bankovní aplikace ještě donedávna spoléhaly na Secure Element nacházející se na SIM kartě. Situace se za poslední roky mění, stále více bank implementuje platební aplikaci využívající API Host-based Card Emulation a Secure Element umisťují na své servery. Tento způsob dává uživatelům větší volnost ve vybírání poskytovatele své platební aplikace.

Architektura Host-based Card Emulation je založena na službě s názvem „HCE services“, která běží na pozadí telefonu a uživatel ji tak nemusí manuálně spouštět při každém přiložení telefonu k terminálu. Naopak terminál zahajuje komunikaci

[8] Application ID je jedinečné 16 bajtové číslo aplikace dle normy ISO/IEC 7816-4.

a může vybudit příslušnou aktivitu v telefonu. Případně může celá transakce proběhnout na pozadí telefonu bez jakéhokoliv oznámení.

Uživatel může na svém zařízení využívat hned několik na sobě nezávislých platebních aplikací. Při zahájení transakce proto potřebuje terminál vědět, se kterou aplikací má navázat komunikaci. Již zmíněná norma ISO/IEC 7816-4 rozlišuje aplikace na základě jejich AID. V případě bankovních karet jsou tato čísla veřejně známá a podléhají mezinárodní registraci (Visa a MasterCard). U vývoje nových (nezávislých) aplikací, což je případ i ukázkové aplikace v rámci této diplomové práce, je potřeba zvolit číslo začínající znakem „F“. Pravidla tvorby AID jsou podrobněji specifikovány v ISO/IEC 7816-4. Zde je stručný přehled:

- *AID začínající znakem ‚A‘:* Mezinárodně registrované AID.
- *AID začínající znakem ‚D‘:* AID registrované na národní úrovni.
- *AID začínající znakem ‚F‘:* Proprietární AID nepodléhající registraci.

Například u karet MasterCard lze nalézt jejich AID (A0 00 00 00 04 10 10) na každém pokladním bloku.

Obrázek 22: AID na stvrzence [autor]

Při zahájení komunikace terminál posílá jako první APDU příkaz „SELECT AID“. Následuje konkrétní číslo AID. Na straně mobilního telefonu operační systém interně vyjme číslo AID z přijatého řetězce a dále jej zasílá službě Host card service, která adekvátně odpoví. Konkrétní ukázka implementace je popsána v kapitole 7.1 HostApduService.

5 Známé útoky na bezpečnost dat

V informatice existuje obor počítačová bezpečnost, který si klade za cíl odhalit a zmenšit rizika spojená s užíváním počítače, případně jiného komunikačního prostředku. Obecně má bezpečnost informačních systémů následující tři cíle:

- *Integrita* – informace a data zůstávají ve své původní podobě. Cílem je zajistit, aby nedocházelo ke změně dat náhodně (např. rušením), případně úmyslně (útokem).
- *Důvěrnost* – informace a data jsou dostupné pouze oprávněným osobám. Důvěrnost může být zajištěna stanovením interních pravidel v přístupu k daným informacím.
- *Dostupnost* – snahou je zajistit informace a data dostupné všem oprávněným uživatelům za každé situace.

Cíle útoku mohou být různé. Útočník se může snažit získat citlivá data, která následně odprodá za nemalou finanční odměnu. Případně se může pokusit o ovládnutí celého zařízení nebo o narušení jeho funkce. Typy útoků lze dělit na útoky na kryptografické algoritmy a na útoky na autentizační protokoly.

5.1 Útoky na kryptografické algoritmy

Luštěním zašifrovaného textu se zabývá vědní disciplína zvaná kryptoanalýza, která společně s kryptografií tvoří podmnožinu kryptologie. Člověk, který se o analýzu šifry nebo šifrovaného textu pokouší, bývá označován jako kryptoanalytik. Pokud se analýza šifry nebo šifrovaného textu povede, získá kryptoanalytik buď samotný otevřený text, nebo šifrovací klíč [2].

5.1.1 Útok hrubou silou

Útok hrubou silou (brute force attack) je pokus o rozluštění šifry bez znalosti klíče k jejímu dešifrování. Útočník se snaží rozšifrovat šifrovaný text postupně na základě všech možných kombinací z předem definovaných znaků. K tomu potřebuje výkonný výpočetní systém a dostatek času. Pro snížení tohoto rizika by proto měla být dostatečná délka šifrovacího klíče (minimálně 256 bitů). Tím se čas

a výkon potřebný pro útok hrubou silou dostane nad hranice dostupných možností.

Útok hrubou silou lze použít i pro uhádnutí dvojice *uživatel* a *heslo*. Útočník se snaží automaticky vygenerovat možná hesla a uhodnout správnou variantu. Existují i předem připravené seznamy hesel, které útočník postupně zkouší. V tomto případě se jedná o slovníkový útok. Pokud uživatel používá slabé heslo skládající se pouze z jednoho slova nebo několika číslic, je tato metoda velice úspěšná. Pro uživatele je proto nejlepší ochranou použít dostatečně silné heslo. To představuje kombinaci velkých a malých písmen společně s čísly a speciálními znaky.

Útokům hrubou silou mohou čelit také servery. K jejich obraně se často používá omezení počtu špatně zadaných hesel. Po předem definovaném počtu špatných pokusů server záměrně reaguje v delším čase. Tím se případný útok hrubou silou stává časově náročnější. Časté jsou také případy, kdy po několika špatně zadaných heslech dojde k časově omezenému uzamknutí účtu. Tato metoda může být ovšem útočníkem rovněž zneužita. Při zkoušení kombinací jednotlivých hesel může záměrně blokovat uživatelské účty.

5.1.2 Luštění se znalostí šifrovaného textu

Při tomto útoku je potřeba mít k dispozici velké množství textu šifrovaného stejným algoritmem a stejným klíčem (tyto předpoklady je nutné ověřit). Úkolem je přeložit tyto zprávy do otevřeného textu. V některých případech slabých šifer se podaří získat i šifrovací klíč [2].

5.1.3 Luštění se znalostí otevřeného textu

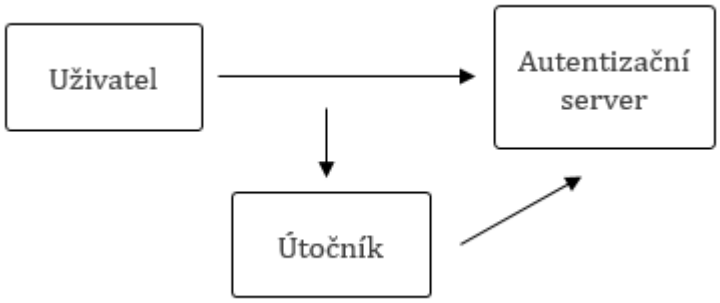
Situace je obdobná jako v předchozím případě. Útočník má k dispozici několik šifrovaných zpráv společně s otevřeným textem. Jeho úkolem je získat klíč [2].

5.2 Útoky na autentizační protokoly

Smyslem autentizačních protokolů je vzájemné prokázání identit komunikujících stran. I tyto protokoly jsou častým terčem útoků, které lze dělit do několika kategorií:

5.2.1 Útok opakováním

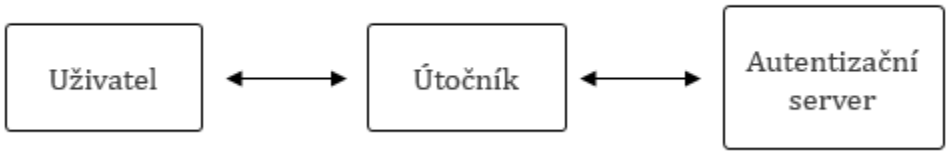
Spočívá v odposlechu (eavesdropping attack) části komunikace mezi dvěma autentizujícími se stranami a následném použití (replay) odposlechnutých dat k pozdější autentizaci útočníka. V nejjednodušším případě se může jednat o odposlechnutí hesla, které uživatel využívá ke své autentizaci. Jednoduchou obranou je použití časového razítkování či metody výzva – odpověď [2].



Obrázek 23: Útok opakováním [2]

5.2.2 Útok ze středu

Útok ze středu (man in the middle attack) je založen na přítomnosti útočníka mezi oběma komunikujícími stranami a jeho kontrole nad celou komunikací. Útočník odposlouchává jednotlivé zprávy autentizačního dialogu a postupně navazuje spojení s oběma stranami A a B tak, že pro stranu A se jeví být stranou B a naopak [2].



Obrázek 24: Útok ze středu [2]

Útok lze použít proti asymetrickým šifrám, kdy dochází k výměně klíčů. Alice chce Bobovi poslat zprávu. Pro šifrování potřebuje od Boba získat jeho veřejný klíč. Cyril ovšem tento veřejný klíč zachytí a nahradí ho svým vlastním klíčem (s falešnou informací, že pochází od Alice). Stejným způsobem Cyril odchytí veřejný klíč od Alice. Nyní si obě strany (Alice a Bob) myslí, že mají veřejný klíč toho druhého. To ale není pravda, mají klíč Cyrila. Cokoliv si nyní Alice s Bobem

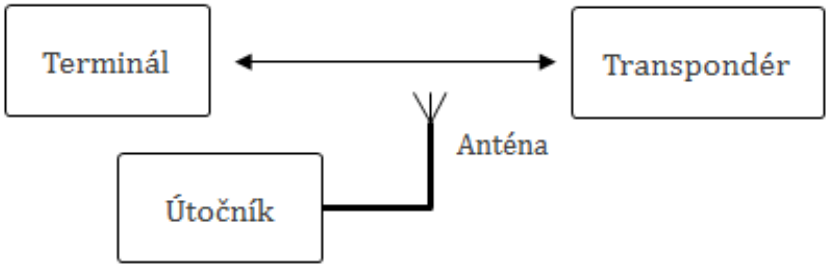
zašifrovaně pošlou, Cyril zachytí, dešifruje, přečte (případně pozmění), znovu zašifruje a odešle příjemci. Alice ani Bob nic nepoznají. Nejlepší možnou obranou proti tomuto útoku je ověření získaného veřejného klíče (certifikátu) pomocí třetí strany – certifikační autority. Ta zaručuje svým podpisem identitu majitelů veřejných klíčů.

5.3 Útoky na NFC

Technologie NFC vychází svoji podstatou z RFID. Je tedy náchylná na všechny útoky, které lze provést vůči zařízení komunikujícímu pomocí bezdrátového rozhraní. Čtecí vzdálenost technologie NFC je maximálně několik centimetrů. I přesto se na tak krátké vzdálenosti mohou vyskytnout bezpečnostní rizika. Technologie NFC nemá nijak zabezpečenou komunikaci, může proto teoreticky docházet k odposlechu přenášených dat. Tím se stává zranitelnou vůči modifikaci dat při jejich přenosu. Průběh komunikace si proto komunikující strany musí zabezpečit na vyšších vrstvách.

5.3.1 Odposlech

Při komunikaci dvou zařízení přes radiové rozhraní může útočník teoreticky použít anténu pro odposlech jejich komunikace. Otázkou vždy zůstává, na jakou vzdálenost se útočník musí dostat ke komunikujícím zařízením, aby byl schopen začít odposlouchávat přenos dat. Na tuto otázku neexistuje jednoznačná odpověď, působí zde hned několik faktorů, které ovlivňují kvalitu signálu (velikost antény, prostředí, kvalita útočníkova přijímače a dekodéru, síla vysílacího signálu NFC jednotky a další). Útočník musí také vědět jak získat potřebná data z přijatého signálu a jakým způsobem je dekódovat.



Obrázek 25: Odposlech NFC komunikace [2]

Dalším důležitým faktorem je způsob generování radiového pole. Zařízení může generovat své vlastní (aktivní mód) nebo může používat radiové pole vygenerované jiným zařízením (pasivní mód). Každý z režimů využívá jiný způsob vysílání dat. Obecně platí, že u pasivního režimu je možnost odposlechu náročnější. Vzdálenost pro odposlech je v tomto případě maximálně 1 metr. U aktivního vysílání může být dosah až do 10 metrů [20].

NFC technologie se nemůže bránit odposlechu dat. Nejefektivnějším řešením je proto přenášet data v šifrované podobě například na základě algoritmu RSA.

5.3.2 Modifikace dat

Útočník se snaží podstrčit příjemci falešná data tak, aby se mu jevila jako validní. Proveditelnost tohoto útoku závisí na síle amplitudové modulace. Útočník musí být schopen vysílat signál takovým způsobem, který dokonale překryje původní signál na anténě přijímače. To je prakticky neproveditelné [20].

Obrana před tímto útokem může být v použití lepšího způsobu kódování a větší hloubce amplitudové modulace. Dalším způsobem je neustálá kontrola radiofrekvenčního pole ze strany komunikujících stran. Jakmile dojde k rušení, strany přestanou komunikovat.

5.3.3 Vkládání dat

Útočník se snaží vkládat svá data do právě probíhající komunikace mezi dvěma zařízeními. Toto je možné pouze v případě, kdy odpovídající strana potřebuje dlouhý čas na odpověď. Útočník se snaží podstrčit svá data ještě před odpovědí oprávněné strany. Pokud by se data překrývala, příjemce je označí za vadná [20].

Zamezit tomuto útoku může komunikující strana tím, že minimalizuje čas pro svou odpověď. V tomto případě útočník nemůže být rychlejší než oprávněná strana. Zamezit případnému útoku může rovněž transpondér, který monitoruje radiofrekvenční pole ve svém okolí v době své aktivity. Pokud zjistí narušení, přeruší komunikaci.

5.3.4 Útok ze středu

Princip útoku byl již popsán v předchozí kapitole 5.2.2 Útok ze středu. V případě komunikace přes rozhraní NFC Alice musí vysílat data v aktivním modu

a Bob v pasivním. V případě, že je Cyril v dostatečné blízkosti, může odposlechnout data vysílaná Alicí. Navíc musí generovat rušivý elektromagnetický signál, který zamezí Bobovi přijmout původní data. Alice tento rušivý signál však může detekovat a přerušit spojení. Pokud by k přerušení ze strany Alice nedošlo, Cyril musí přeposlat data (již upravená) Bobovi. Zde nastává další problém, jelikož signál generovaný Alicí je stále aktivní, Cyril by proto musel vytvořit druhý radiový signál. Udržovat dva signály dat ve stejném radiovém pásmu je v praxi neproveditelné.

Další variantou je útok na zařízení, která spolu komunikují v aktivním režimu. Postup je zde podobný jako u aktivního zařízení, které komunikuje s pasivním. Pokud nedojde k přerušení spojení ze strany Alice, Cyril přepošle již upravená data Bobovi. Oproti předchozímu případu Alice v tomto momentě vypíná vysílání svého radiového pole, což může využít Cyril a začít vysílat. Alice však stále naslouchá a očekává odpověď od Boba. Místo toho obdrží data od Cyrila. I zde se ale objevuje problém se dvěma radiovými signály. Ty může Alice vyhodnotit jako kolizi a spojení přerušit.

Jak již bylo zmíněno v této kapitole, útok man in the middle je v případě NFC komunikace prakticky neproveditelný. I přesto je vhodné pro výměnu dat použít některou z metod šifrování.

6 Návrh platebního systému

Cílem této práce je vytvořit jednoduchý platební systém, ve kterém mobilní telefon vystupuje v roli bezkontaktní platební karty. Pro autentizaci uživatele a přenos transakčních dat mezi telefonem a terminálem je využito rozhraní NFC v režimu Host-based Card Emulation (HCE). Uživatel má k dispozici webový portál (www.nfctap.cz), kde spravuje svá mobilní zařízení a zároveň má přehled o platebních transakcích. Dále má k dispozici aplikaci pro mobilní telefon s operačním systémem Android, jejíž hlavní rolí je emulace platební karty. Aplikaci lze nainstalovat na mobilní telefony s minimální verzí Android 4.4. Důvodem je využívání režimu Host-based Card Emulation, který je podporován až od zmíněné verze. V době psaní této práce se jedná o 73,9 %[9] zařízení na trhu. Mobilní aplikace poskytuje uživateli několik základních aktivit, jako jsou informace o zařízení, historie plateb a základní nastavení. Její hlavní činnost však spočívá v podepisování platební transakce, na základě které lze ověřit autenticitu uživatele.

Další komponentou platebního systému je platební terminál. Pro účely této práce je jako platební terminál použita NFC čtečka, která je obsluhována desktopovou PC aplikací. Jedná se tak pouze o virtualizaci platebního terminálu. Pokud aplikaci nasadíme v reálném prostředí, musíme počítat s existující sítí platebních terminálů. Tato situace by vyžadovala úpravu aplikace na straně mobilního telefonu. Pravděpodobně by musel být upraven způsob komunikace na úrovni APDU příkazů. Stěžejní část funkcionality platebního systému, tedy autentizaci uživatele, lze však demonstrovat i s použitím virtuálního terminálu. Principy autentizace jsou implementovány na základě dvou mechanizmů – symetrické a asymetrické kryptografie. Každá z těchto variant představuje kompletně nezávislou ukázkovou aplikaci. Nazývají se NFCtapHMAC.apk a NFCtapPKI.apk. Jejich srovnání lze pak nalézt v 8. kapitole.

Při návrhu platebního systému byl brán zřetel na bezpečnost a pohodlí uživatele. Veškerá data, která jsou citlivější povahy, jsou nejprve šifrována a poté uložena v paměti mobilního telefonu. Šifrování je použito i v případě komunikace

[9] Údaj převzatý z Android Studia (červen 2016).

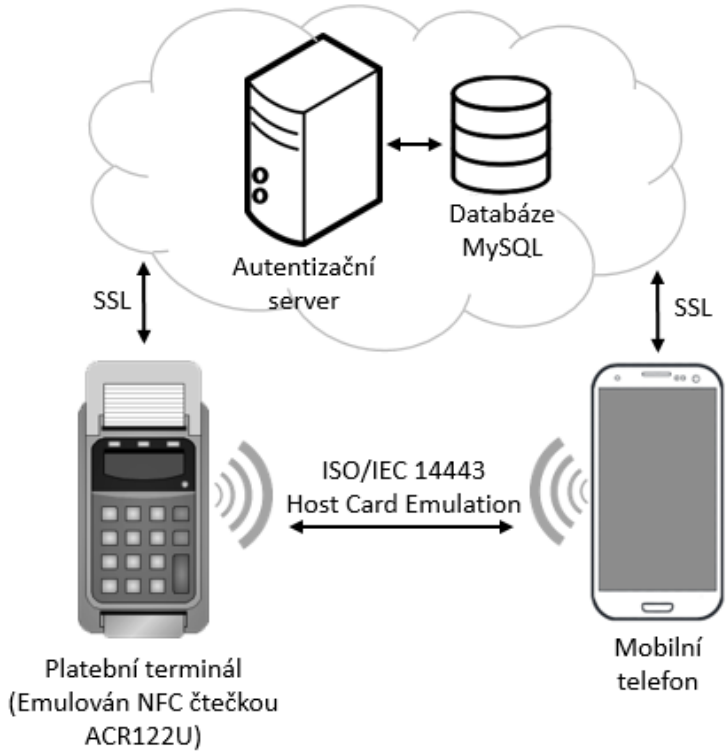
mobilní aplikace se serverem. Konkrétně je použit protokol SSL[10]. Bezpečnostní klíče jsou uloženy a šifrovány v úložišti Android Keystore. Jejich šifrovací master klíč je odvozen z hesla zámku obrazovky. Z tohoto důvodu lze aplikaci použít pouze tehdy, není-li telefon uzamčen. Nemá-li uživatel v průběhu instalace aplikace funkci zámku obrazovky aktivovanou, je na tuto skutečnost upozorněn. Pokud ani po upozornění zámek neaktivuje, aplikace se nespustí. Pokud dojde v průběhu užívání aplikace k deaktivování zámku obrazovky, případně ke změně hesla nebo pinu, jsou bezpečnostní klíče automaticky smazány. Tuto ochranu nelze nijak ovlivnit, vychází z vnitřní implementace úložiště Android Keystore. Dojde-li ke smazání klíčů, musí uživatel nainstalovat aplikaci znova.

Aby uživatel mohl začít používat svůj telefon jako platební kartu, musí nejprve projít registrací přes webový portál. Registrací se podrobněji zabývá následující kapitola 7 Implementace a testování. Jakmile je telefon registrován, může jej uživatel začít ihned používat. Po přiložení telefonu k platebnímu terminálu dojde k automatickému spuštění platební aplikace v mobilním telefonu. Na displeji telefonu je uživatel informován o úspěchu či neúspěchu transakce. Následně se aplikace sama zavře. V průběhu transakce není uživatel nijak obtěžován dodatečným zadáváním pinu nebo hesla. Jedinou podmínkou je přiložit telefon s odemčenou obrazovkou. O úspěšné autentizaci uživatele rozhoduje server, který je ve spojení s terminálem po celou dobu transakce. Je-li autentizace úspěšná, server provede zápis transakce.

6.1 Komponenty platebního systému

Platební systém se skládá ze tří hlavních částí: Autentizačního serveru, mobilního telefonu a platebního terminálu.

[10] Secure Sockets Layer



Obrázek 26: Komponenty platebního systému [autor]

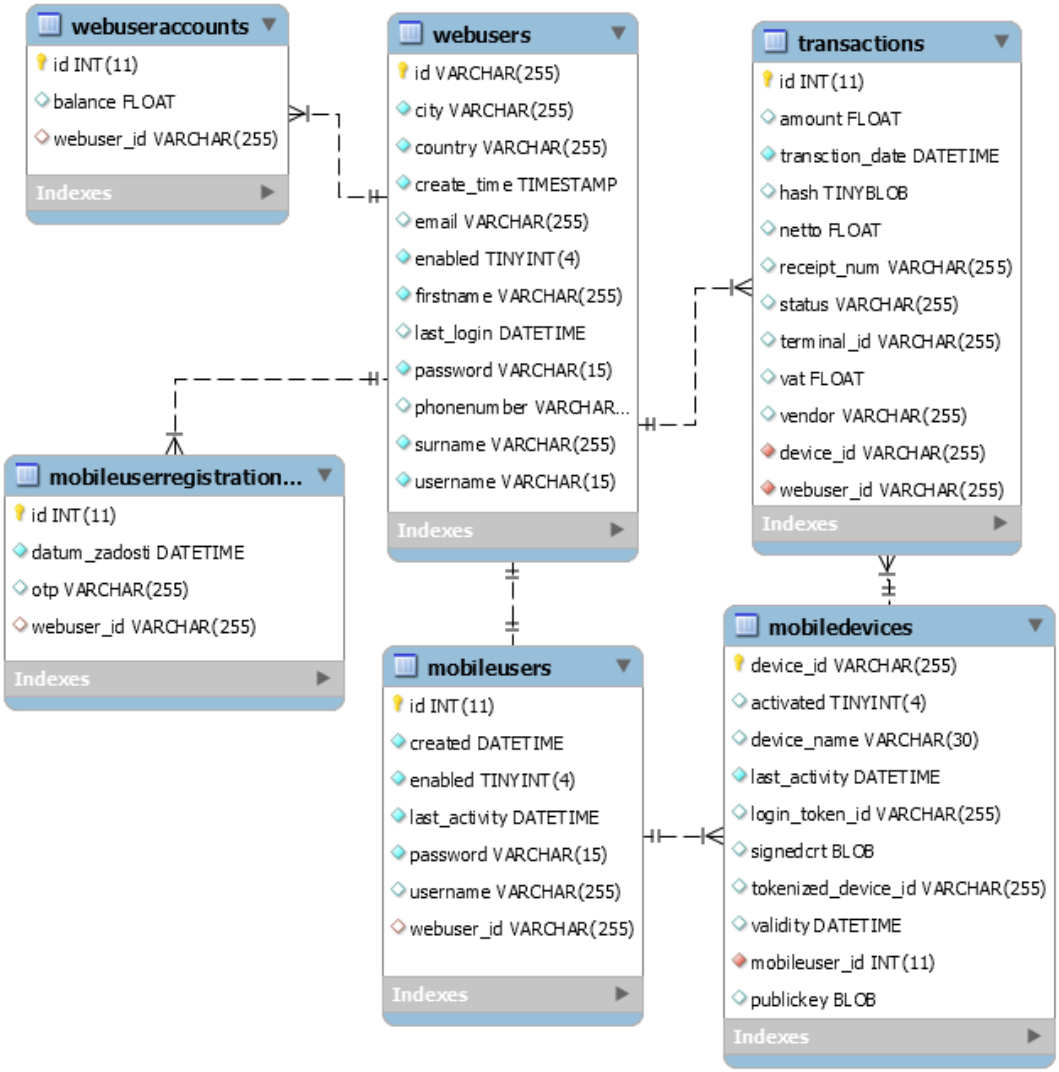
6.1.1 Autentizační server

Autentizační server je vyvinut v jazyce Java, konkrétně za využití frameworku Java Spring. Primární úlohou serveru je online autentizace a poté autorizace provedené platby. Probíhá-li autentizace mezi telefonem a terminálem na základě symetrické kryptografie, je na straně serveru vygenerován klíč, který je bezpečně uložen v databázi ke konkrétnímu telefonu. Zároveň je zaslán přes zabezpečenou komunikaci SSL mobilnímu telefonu hned v úvodu jeho registrace. V případě autentizace pomocí asymetrické kryptografie je na straně serveru uložen veřejný klíč uživatele a zároveň i jeho certifikát.

Server má i své webové rozhraní. Každý z uživatelů má na server přístup přes webové stránky, kde může vidět přehled transakcí a spravovat svá mobilní zařízení. Přístup k serveru je chráněn pomocí uživatelského jména a hesla. Server rozlišuje dvě skupiny uživatelů: *mobileusers a webusers*. Každému z nich je povolen přístup pouze k některým částem serveru. Z tohoto důvodu přístup k serveru ošetřují dva autentizační manažeři: *mobAuthenticationManager* kontrolující dotazy ze strany mobilního telefonu a *webAuthenticationManager*, který kontroluje

přístup uživatelů přes webové stránky. K administraci webové části byl vytvořen ještě uživatel *admin* patřící do skupiny *webusers*. Tento uživatel má zvláštní oprávnění editace běžných uživatelů ze skupiny *webusers* a jejich mobilních zařízení.

Server je napojen na databázi MySQL. Její hlavní entity znázorňuje následující obrázek:



Obrázek 27: Zjednodušené znázornění tabulek v databázi MySQL [autor]

Pro objektově-relační mapování je použit framework Hibernate, jehož hlavní funkcí je mapování Java objektů na entity v relační databázi. Mapování je provedeno na základě anotací u atributů jednotlivých objektů modelu.

6.1.2 Mobilní telefon

Důraz při vývoji aplikace byl kladen hlavně na bezpečnou autentizaci a přenos dat při zahájení platební transakce v režimu emulace karty. Aby uživatel mohl využívat mobilní telefon jako platební kartu, je třeba nejprve projít registrací. Smyslem registrace je jednak uložení údajů o mobilním telefonu na straně serveru, ale hlavně vygenerování bezpečnostních klíčů, které se následně používají při autentizaci uživatele, respektive k podpisu platební transakce.

Vzhledem k využívání služby Host-based Card Emulation, která byla implementovaná až ve verzi Android 4.4 (KitKat), je nutné stanovit minimální API na hodnotu 19. Pro komunikaci mobilní aplikace se serverem je použito rozhraní *REST*[11], které slouží pro jednotný a snadný přístup ke zdrojům serveru. Aplikace mobilního telefonu využívá framework „Spring-android-rest-template“. Ta zjednodušuje tvorbu *HTTPs* klienta a s ním spojené dotazy pro přístup ke zdrojům. Jako formát výměny dat mezi severem a mobilní aplikací je použit objektový zápis ve formátu *JSON*[12]. Přístup ke zdrojům na straně serveru je chráněn autentizačním mechanizmem (již zmíněný *mobAuthenticationManager)*, který při každém dotazu vyžaduje jméno *(deviceIdHashed)* a heslo uživatele. Aby nedocházelo k opětovnému zasílání jména a hesla, je použit systém časově omezeného tokenu. Token je tvořen na straně serveru po každém úspěšném přihlášení do mobilní aplikace. Časové omezení tokenu způsobí automatické odhlášení uživatele z mobilní aplikace při době nečinnosti delší jak 5 minut.

Mobilní aplikaci tvoří několik na sebe navazujících aktivit. Při instalaci je uživateli zobrazena aktivita obsluhující zadání a kontrolu jednorázového ověřovacího hesla *(OTP)* ze strany serveru. Po úspěšné registraci je uživateli zobrazena hlavní aktivita zobrazující základní přehled o účtu a nastavení mobilní aplikace. Při každém dalším spuštění aplikace je uživateli zobrazena úvodní obrazovka s výzvou zadání hesla. Heslo není v telefonu uloženo, naopak je kontrolováno online ze strany serveru. Toto heslo se nicméně vyžaduje pouze v případě manuálního spuštění aplikace. Do budoucna lze uvažovat i o možnosti

[11] Representational State Transfer
[12] JavaScript Object Notation

realizovat elektronické online platby přímo z mobilní aplikace. V tomto případě je proto dodatečná ochrana heslem na místě.

Avšak v okamžiku přiložení mobilního telefonu k platebnímu terminálu není potřeba zadávat heslo ani pin. Po uživateli se v tuto chvíli pouze vyžaduje mít odemčenou obrazovku, jinak není transakce zahájena. Nechybí ani aktivita potvrzující úspěšnou autentizaci nad platebním terminálem. Dále je v aplikaci několik pomocných tříd sloužících především pro vykonávání kryptografických operací, ověřování platnosti tokenu a dalších činností nutných pro provedení autentizace uživatele při platební transakci.

6.1.3 Platební terminál

Pro účely této práce je celý platební terminál virtualizován pomocí desktopové aplikace, která komunikuje s bezkontaktní čtečkou ACR122U NFC od výrobce Advanced Card Systems Ltd.

Obrázek 28: Virtualizace platebního terminálu (desktopová PC aplikace) [autor]

ACR122U je čtečka bezkontaktních čipových karet pracujících na frekvenci 13,56 MHz (HF) dle normy ISO14443-4 Typ A a B, Mifare, ISO 18092 a FeliCa. Čtečka podporuje framework PC/SC a proto je kompatibilní s již existujícími PC/SC aplikacemi. Navíc nevyžaduje instalaci vlastního ovladače, jelikož využívá standardní CCID ovladače od Microsoftu, které jsou součástí operačního systému Windows.

Obrázek 29: Bezkontaktní čtečka ACR122U [autor]

V následujících kapitolách bude podrobněji popsána implementace autentizace uživatele.

7 Implementace a testování

Pro vývoj serverové aplikace bylo použito vývojové prostředí Spring Tool Suite, které vychází ze základu studia Eclipse a je speciálně navrženo pro práci s frameworkem Java Spring. Pro vývoj mobilní aplikace byl využit nástroj Android Studio.

V této kapitole bude popsána implementace dvou autentizačních mechanizmů. Nejprve varianta autentizace využívající symetrickou kryptografii, poté varianta využívající asymetrickou kryptografii. Nejdříve je ovšem nutné seznámit se se službou HostApduService, která je využita v obou případech. Na jejím základě mobilní telefon emuluje bezkontaktní kartu.

7.1 HostApduService

Třída *HostApduService* implementuje dvě metody *processCommandApdu()* a *onDeactivated()*. Metoda *processCommandApdu()* se zavolá vždy, jakmile NFC terminál zašle APDU příkaz (dle specifikace ISO/IEC 7816-4). V případě aplikace NFCtap je metoda volána v okamžiku přiložení telefonu do pole čtečky. Jako první příkaz terminál zasílá "SELECT AID" APDU, který obsahuje AID požadované služby (aplikace). Návratová hodnota byte[] slouží jako odpověď pro čtečku. Její obsah je čistě v režii vývojáře.

```
public class MyHostApduService extends HostApduService {
    @Override
    public byte[] processCommandApdu(byte[] apdu, Bundle extras) {
        ...
    }
    @Override
    public void onDeactivated(int reason) {
        ...
    }
}
```

Zdrojový kód 1: Třída pro implementaci režimu Host-based Card Emulation [18]

Konkrétní AID číslo aplikace se registruje v souboru *apduservice.xml.* Pro účely aplikace se symetrickým šifrováním (*NFCtapHMAC.apk*) bylo zvoleno číslo „*F222222222*“, pro asymetrické šifrování (*NFCtapPKI.apk*) číslo „*F111111111*“.

Příkaz „SELECT AID“ posílaný čtečkou je pak následující:

```
// SELECT AID =00A4040005F222222222
public static final byte[] SELECT_NFCTAP_APPLET_CMD = { 0x00, (byte)
0xA4, 0x04, 0x00, 0x05, (byte) 0xF2, 0x22, 0x22, 0x22, 0x22 };
```

Zdrojový kód 2: Příkaz SELECT AID posílaný terminálem do mobilního telefonu (pro aplikaci NFCtapHMAC.apk) [autor]

Příkaz vychází z normy *ISO7816-4 (*podrobněji v kapitole 3.2.2 Komunikace karty s okolím), kde jednotlivé bajty znamenají:

- 0x00 (CLA) – Označení třídy příkazů.
- 0xA4 (INS) – SELECT FILE.
- 0x04 (P1) - Instruction parameter 1.
- 0x00 (P2) - Instruction parameter 2.
- 0x05 (Lc field) – Počet zasílaných bajtů.
- 0xF2, 0x22, 0x22, 0x22, 0x22 (Data field) – Představuje volanou aplikaci, resp. její AID dříve specifikovaný v souboru *apduservice.xml.*

Jak již bylo zmíněno, metoda *public byte[] processCommandApdu()* očekává návratovou hodnotu *byte[]*. Pokud by jejím obsahem bylo pouze jedinečné číslo (např. IMEI[13]), aplikace by pak byla velice snadno zneužitelná. Získá-li totiž útočník IMEI telefonu, může vytvořit svou vlastní aplikaci a toto číslo při žádosti čtečky libovolně zasílat v návratové hodnotě.

Z důvodu vyššího zabezpečení byly navrhnuty a implementovány dva autentizační mechanizmy.

7.2 Autentizace s využitím symetrické kryptografie

Svou identitu uživatel v tomto případě jednoznačně prokazuje vlastnictvím symetrického klíče, který ze serveru získá v průběhu své registrace. Mobilní telefon použije tento klíč k vytvoření jedinečného kryptogramu *(HMAC)* z přijaté výzvy od platebního terminálu. Po zpětném zaslání podepsané výzvy proběhne

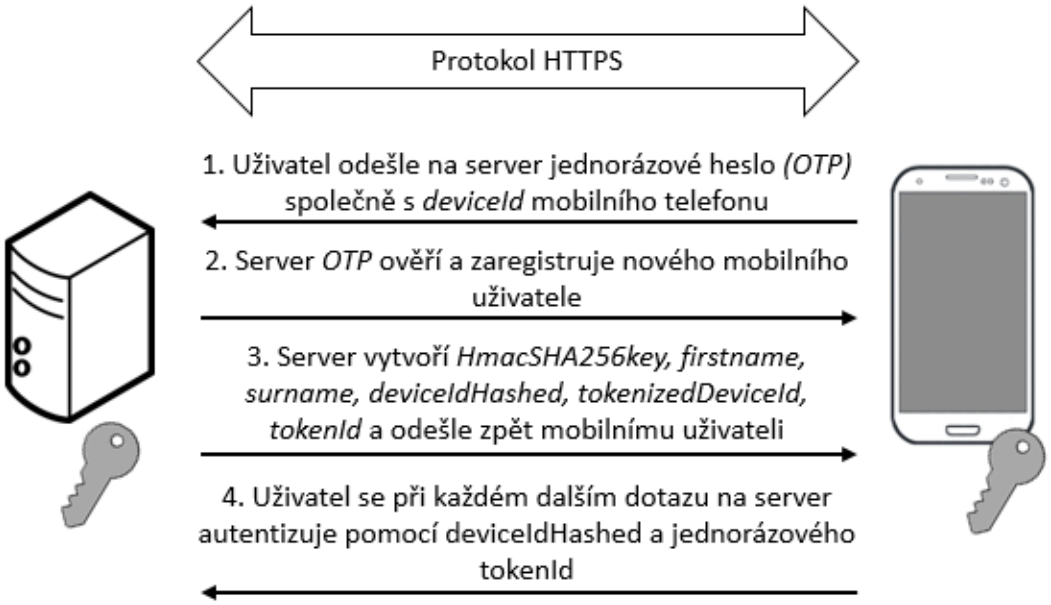
[13] International Mobile Equipment Identity. Jde o unikátní číslo přidělené výrobcem mobilnímu telefonu.

kontrola kryptogramu na straně serveru. Uživatel však nejprve musí projít registrací.

7.2.1 Registrace uživatele

Registrace uživatele je klíčový krok celého systému. Založit uživatelský účet je oprávněn pouze administrátor *(admin)*. Vzhledem k nakládání s citlivými daty uživatele je pro registraci nezbytné, aby se uživatel osobně dostavil na registrační místo a prokázal svojí totožnost např. občanským průkazem.

Následně je uživateli umožněn přístup do webového portálu. Ve svém profilu může uživatel sledovat transakční historii a spravovat svá mobilní zařízení. Aby mohl svůj telefon začít používat jako platební kartu, musí nejprve zvolit registraci nového zařízení.



Obrázek 30: Registrace nového zařízení s využitím symetrické kryptografie [autor]

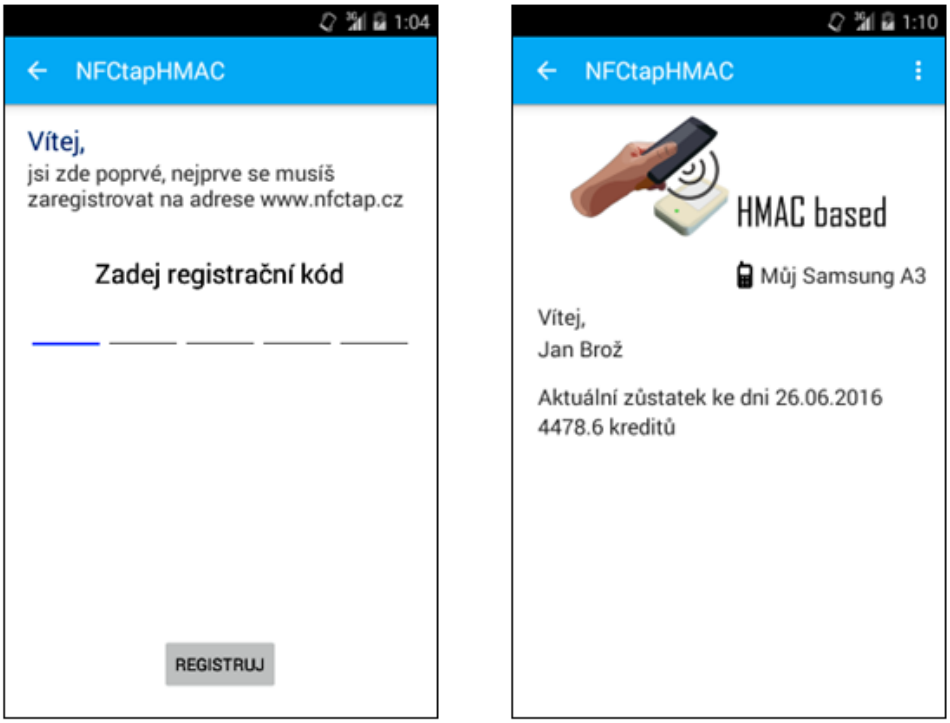
Po volbě „Zaregistrovat nové zařízení“ je v 1. kroku vygenerováno jednorázové 10 bajtové heslo *OTP* [14](registrační kód), které je uživateli znázorněno na obrazovce webového prohlížeče. Zároveň je vyzván, aby toto heslo vložil do mobilní aplikace NFCtapHMAC.apk ihned po jejím prvním spuštění.

[14] One Time Password

Server ve 2. kroku ověří, zda *OTP* odpovídá hodnotě, kterou sám vygeneroval. Pokud *OTP* souhlasí, vytvoří server nového mobilního uživatele (*mobileUser*). Znalostí *OTP* uživatel jasně prokazuje svou identitu, jelikož pro jeho získání musí být registrován a přihlášen ve svém webovém profilu.

```
@RequestMapping(value = "/verifyotp", method = RequestMethod.POST,
produces = "application/json")
public @ResponseBody Message getMessage(@RequestParam("otp") String
otp, @RequestParam("deviceId") String deviceId, Principal principal)
{   // Zjisti, zda již došlo ke generování OTP
    if (mobileUserRegService.getReqReqByOtp(otp) != null) {
        // Najdi na základě OTP celou žádost
        MobileUserRegistrationRequest mobUserRegReq =
        mobileUserRegService.getReqReqByOtp(otp)
        // Zjistím, ke kterému uživateli je dané otp vygenerováno
        WebUser webUser = mobUserRegReq.getWebUser();
        // Nalezni konkrétního mobilního uživatele
        MobileUser mobileUser = mobileUserService.
        getMobileUserByUsername(webUser.getUsername());
        // Vytvoř hash se solí(webuser.id) z obdrženého deviceid
        // Tímto se snažím předejít možnému odhadnutí deviceId
        // ze strany mobilního telefonu
        String deviceIdHashed = makeHashedDeviceId(deviceId,
        webUser.getId());
        // Vytvoř pro něj device
        MobileDevice mobileDevice = new MobileDevice();
        mobileDevice.setDeviceId(deviceIdHashed);
        mobileDevice.setMobileUser(mobileUser);
        mobileDevice.setLastActivity(new Date());
        mobileDevice.setActivated(true);
        // Vytvoř tokenizaci z deviceIdHash
        String tokenizedDeviceId =
        generateTokenizedDeviceId(deviceIdHashed));
        mobileDevice.setTokenizedDeviceId(tokenizedDeviceId);
        // Vygeneruj symetrický klíč
        String shaKey = generateHmacSHA256key();
        mobileDevice.setShaKey(shaKey);
        //Ulož tento mobilní device do db
        mobileDeviceService.createNewMobileDevice(mobileDevice);
        // Na závěr předej nové údaje uživateli, resp. mobilní
        // aplikaci
        Message messageBack = new Message();
        messageBack.setSubject("Success");
        messageBack.setText("Registrace uživatele proběhla úspěšně!
        \nVítejte " + mobileUser.getUsername());
        // Naplň zprávu zpět pro mobilní telefon
        String[] values = { shaKey, firstname, surname,
        deviceIdHashed, tokenizedDeviceId };
        // Pošli tyto údaje zpět do telefonu
        messageBack.setOther(values);
        return messageBack;
```

Zdrojový kód 3: Zjednodušená ukázka OTP kontroleru na straně serveru [autor]



Obrázek 31: Průběh registrace aplikace [autor]

Ve 3. kroku zároveň server vygeneruje a zašle zpět uživateli tyto nové hodnoty:

a) *HmacSHA256key* je symetrický klíč, který se bude využívat k autentizaci uživatele. Server tento klíč uloží k danému uživateli do databáze. Ve výsledku je proto klíč na straně serveru i mobilní aplikace stejný.

```
private byte[] generateHmacSHA256key() {
	KeyGenerator keyGen = KeyGenerator.getInstance("HmacSHA256");
	keyGen.init(256);
	SecretKey key = keyGen.generateKey();

	return key.getEncoded();
}
```

Zdrojový kód 4: Ukázka metody generující symetricky klíč (server) [autor]

b) *firstname* je křestní jméno, kterým je uživatel registrován na webovém portálu.

c) *surname* je příjmení, kterým je uživatel registrován na webovém portálu.

d) *deviceIdHashed* je upravené identifikační číslo mobilního telefonu deviceId, které uživatel zasílá v prvním kroku.

```
private byte[] createDeviceId() {
    TelephonyManager telephonyManager = (TelephonyManager)
getSystemService(Context.TELEPHONY_SERVICE);
       if (telephonyManager.getDeviceId() != null) {
         String imei = telephonyManager.getDeviceId();
         // Vytvoř nové pole z hashe (resp. zkraťjej)
         byte[] imeiBytes = myPki.doHashGetBytes(imei);
        // 8 bytes
         byte[] imeiBytes8 = Arrays.copyOf(imeiBytes, 8);

         return imeiBytes8;

    } else {

         return null;
    }
}
```

Zdrojový kód 5: Tvorba deviceId (mobilní aplikace) [autor]

Z bezpečnostních důvodů je pak toto *deviceId* ještě změněno na straně serveru na novou hodnotu d*eviceIdHashed.* Touto hodnotou se pak aplikace automaticky hlásí při vznesení dotazu na server.

```
private byte[] makeHashedDeviceId(String deviceId, String salt) {
    MessageDigest digest = MessageDigest.getInstance("SHA-256");
    digest.reset();
    digest.update(Converter.hexStringToByteArray(salt));
    byte[] hashedBytes256 =
          digest.digest(Converter.hexStringToByteArray(deviceId));
    // Zkraťdeviceid na 10 bajtů
    byte[] hashedBytes = Arrays.copyOf(hashedBytes256, 10);

    return hashedBytes;
}
```

Zdrojový kód 6: Tvorba deviceIdHashed [autor]

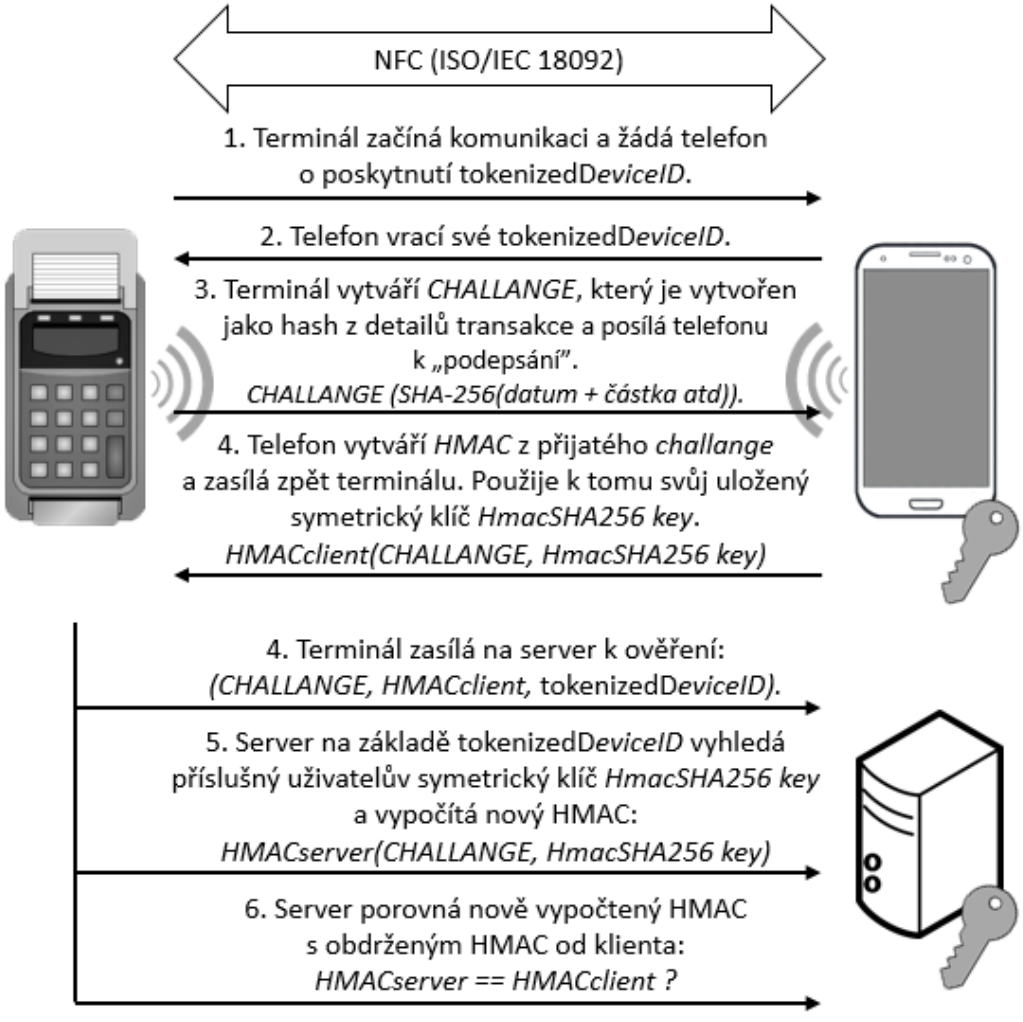
e) *tokenizedDeviceId* je číslo, které se skládá z náhodně vygenerovaných 6 bajtů. K nim jsou připojeny poslední 4 bajty z původního *deviceIdHashed*. Číslo *tokenizedDeviceId* se používá při autentizaci uživatele u platebního terminálu, zatímco deviceIdHashed se využívá při autentizaci uživatele vždy, jakmile mobilní aplikace kontaktuje server.

f) *tokenId* je náhodné číslo generované serverem, jehož platnost je časově omezena. Aplikace toto číslo automaticky používá jako heslo při kontaktování serveru z mobilní aplikace.

7.2.2 Platební transakce a autentizace uživatele

Po úspěšné registraci je telefon připraven být v roli platební karty skrze své rozhraní NFC, resp. mód Host-based Card Emulation (HCE).

Po zadání částky do platebního terminálu je uživatel vyzván k přiložení svého mobilního telefonu. Princip bezkontaktního přenosu dat mezi terminálem a mobilním telefonem probíhá na kmitočtu 13,56 MHz, čtecí vzdálenost lze proto očekávat maximálně do 10 cm. Po uživateli se nevyžaduje manuální spuštění aplikace, pouze musí mít odemčenou obrazovku v okamžiku přiložení telefonu k terminálu. O úspěšném ukončení přenosu dat je uživatel informován na displeji mobilního telefonu. Postup autentizace je následující:



Obrázek 32: Průběh autentizace s využitím symetrické kryptografie [autor]

Terminál nejprve požádá telefon o zaslání svého identifikačního čísla *tokenizedDeviceId* (krok 1 a 2). Terminál následně vypočte *CHALLANGE* (krok 3), který představuje hash (SHA256) ze zadané transakce. Do výpočtu použije datum, id, číslo terminálu, částku, datum a další údaje spojené s danou transakcí. Tím vznikne jedinečný hash, který terminál zasílá k podpisu mobilnímu telefonu.

Telefon ve 4. kroku vytvoří HMAC z přijatého hashe za použití symetrického klíče *HmacSHA256key.* Tím vzniká jedinečný kryptogram *HMACclient*, který telefon posílá zpět terminálu ke kontrole.

Terminál nyní musí kontaktovat server s požadavkem o ověření přijatého kryptogramu *HMACclient.* Na server zasílá konkrétně *CHALLANGE, HMACclient,* tokenizedD*eviceID* (krok 5).

V 6. kroku server pomocí přijatého tokenizedD*eviceID* vyhledá příslušeného mobilního uživatele. U něho nalezne klíč *HmacSHA256key,* který klient použil k tvorbě kryptogramu. Server nyní použije přijatý *CHALLANGE* a vypočte za použití klíče *HmacSHA256key* znovu kryptogram. Vzniká tak kryptogram s názvem *HMACserver.*

Na závěr server porovná oba kryptogramy *HMACclient* a *HMACserver.* Pokud se rovnají, znamená to, že uživatel disponuje správným klíčem a má k němu přístup. V tomto případě server terminálu vrací: Autentizace úspěšná. V opačném případě server autentizaci zamítá.

7.2.3 Ukázka komunikace mezi terminálem a mobilním telefonem

Komunikace mezi terminálem a mobilním telefonem probíhá na základě příkazů APDU. Příkazy ze strany čtečky a návratové hodnoty telefonu lze vidět ve zdrojovém kódu č.Zdrojový kód 7. Implementace návratových hodnot na straně aplikace je znázorněna v další ukázce zdrojového kódu č.Zdrojový kód 8.

```
COMMAND   > 00A4040005F222222222 // terminál zasílá SELECT_AID
RESPONSE  < 9000 // telefon odpovídá v případě úspěchu selektování aplikace
COMMAND   > 80040000 // terminál žáda o tokenizedDeviceId(INS_GETDEVICEID)
RESPONSE  < 33D590068D9BB52898859000 // telefon vrací své tokenizedDeviceId
// Terminál posílá telefonu challange k podepsání (CHALLANGE):
> 800300002087489CB1458D7887BFA0BB15EE82CA4161B89ABE4F53A49B3BAD6BCCD01119A4
// Telefon vrací zpět podepsanou výzvu (HMACclient)
< 9F68603AADE96F64F5E9C19D6540279EE5ED34818D4BD67675363BF1079FF3DD9000
```

Zdrojový kód 7: Ukázka výměny APDU mezi terminálem a mobilním telefonem [autor]

```
public byte[] processCommandApdu(byte[] commandApdu, Bundle extras) {
    // V případě, že terminál zasílá SELECT_AID
    if (Arrays.equals(commandApdu, SELECT_NFCTAP_APPLET_CMD)) {
       return toBytes(SW_SUCCESS); // Pošli pouze 9000
    }
    // ...
    // ****************************************
    // * Část po úspěšném „vyselektování" aplikace
    // ****************************************/
    // Ano, obdržel jsem APDU 80xxxx (NFC_APPLET_CLA)
    // (CLA, INS, formát pole dle ISO 7816)
    // např 8004xx
    byte ins = commandApdu[OFFSET_INS]; // OFFSET_INS == 1
    switch (ins) {
        // V případě, že terminál zasílá žádost o tokenizedDeviceId
        case INS_GETDEVICEID: // INS_GETDEVICEID == 0x04
            // Vrať tokenizedDeviceId a řetězec zakonči dvěma bajty
            // označující úspěch 0x90 a 0x00
            return ConcatArrays(tokenizedDeviceId, toBytes(SW_SUCCESS));
        // V případě, že terminál zasílá žádost o podpepsání výzvy
        case INS_SIGNCHALLANGE: // INS_SIGNCHALLANGE == 0x03
            // Zjisti délku přijatého challange (lze najít na pozici 5)
            int challangeSize = commandApdu[4];
            byte[] challange = new byte[challangeSize];
            // Odděl část s challenge
            System.arraycopy(commandApdu, 5, challange, 0, challangeSize);
            // Podepiš challange, resp. vytvoř z něj HMAC
            byte[] challangeHmac = createHmac(challange);
            // Zobraz uživateli potvrzovací okno
            showTappingScreen(Convertor.byteArrayToHexString
            (tokenizedDeviceId), Convertor.byteArrayToHexString(challange)
            // vrať terminálu podepsaný challange + úspěch 0x90 a 0x00
            return ConcatArrays(challangeHmac, SELECT_OK_SW);
    }
    // V případě neúspěchu vrať chybu
    return toBytes(SW_DATA_INVALID);
}
```

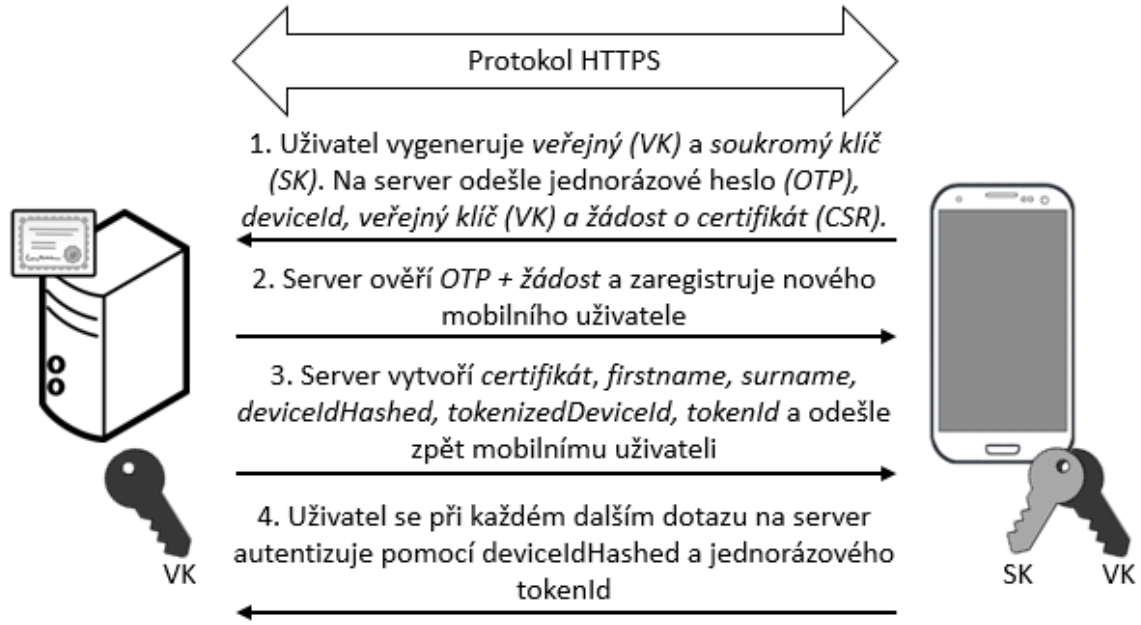
Zdrojový kód 8: Zjednodušená ukázka zpracování APDU ze strany mobilního telefonu [autor]

7.3 Autentizace s využitím asymetrické kryptografie

Svou identitu uživatel v tomto případě jednoznačně prokazuje vlastnictvím soukromého klíče. Autentizace je založena na principu digitálního podpisu. Uživatel svým soukromým klíčem podepíše přijatou výzvu od platebního terminálu a zasílá ji zpět ke kontrole. Ta proběhne na straně serveru pomocí uživatelova veřejného klíče, který se zde nachází od okamžiku jeho registrace. Průběh registrace i autentizace uživatele je v některých částech totožný s postupy popsanými u autentizace s využitím symetrické kryptografie. Z tohoto důvodu budou podrobněji popsány pouze hlavní rozdíly při autentizaci za využití asymetrické kryptografie.

7.3.1 Registrace uživatele

Průběh registrace je podobný jako u předchozí varianty. Po úspěšném vytvoření účtu na webovém portálu musí uživatel zvolit registraci nového zařízení.



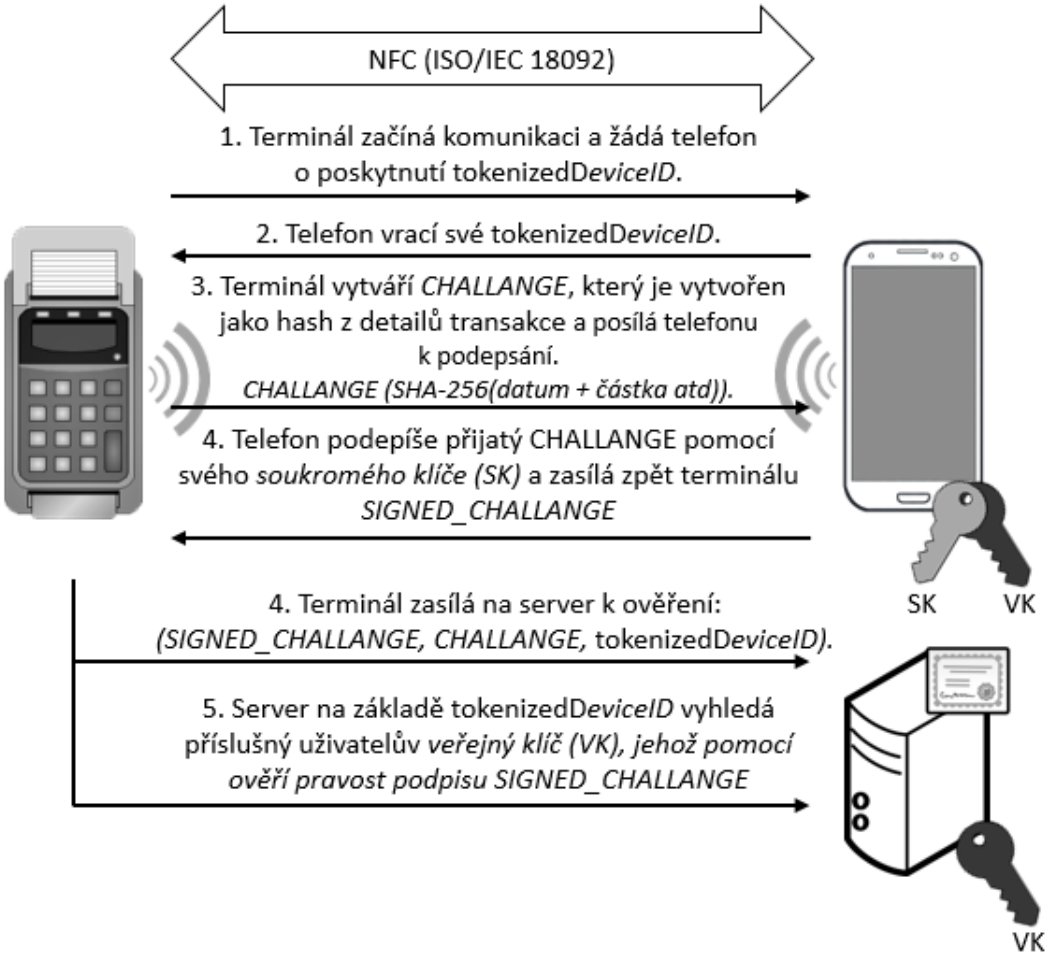
Obrázek 33: Registrace nového zařízení s využitím asymetrické kryptografie [autor]

Součástí serveru je v případě asymetrické kryptografie ještě *certifikační autorita CA.* Ta rovněž disponuje svým soukromým *(SKCA)* a veřejným klíčem *(VKCA)*. Soukromý klíč musí být maximálně střežen, nesmí nastat situace, kdy by došlo ke ztrátě, případně jeho konfrontaci.

V prvním kroku registrace je vytvořen *veřejný (VK)* a *soukromý klíč (SK)* v mobilním telefonu uživatele. *Soukromý klíč (SK)* bude uživatel používat ke své autentizaci při platební transakci. *Veřejný klíč* je společně s *deviceId* odeslán na server k uložení do databáze. V tomto kroku se ještě zasílá *žádost o certifikát (CSR)*, která je na straně serveru zkontrolována a následně je z této žádosti vytvořen *certifikát (CRT).* Certifikační autorita zaručí svým podpisem, že daný certifikát, resp. *veřejný klíč (VK)* patří dané osobě. Certifikát společně s dalšími údaji je ve 3. kroku zaslán zpět mobilnímu telefonu.

7.3.2 Platební transakce a autentizace uživatele

Po úspěšné registraci je telefon připraven být v roli platební karty skrze své rozhraní NFC, resp. mód Host-based Card Emulation (HCE). Na bázi asymetrické kryptografie je postup autentizace následující:



Obrázek 34: Průběh autentizace s využitím asymetrické kryptografie [autor]

Krok 1 až 3 je stejný jako v případě autentizace s využitím symetrické kryptografie. Změna přichází až ve 4. kroku, kdy telefon přijímá *CHALLENGE* od terminálu. Telefon tuto výzvu podepíše svým soukromým klíčem. Vzniká tak *SIGNED_CHALLANGE*, který telefon posílá zpět terminálu.

Terminál nyní musí kontaktovat server s požadavkem o ověření přijatého kryptogramu (krok 4).

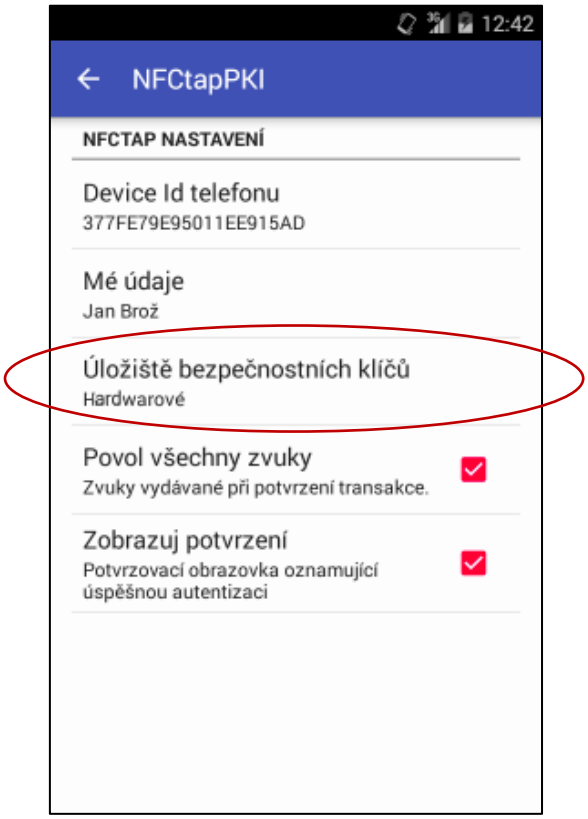
Server pomocí přijatého *tokenizedDeviceId* vyhledá příslušného mobilního uživatele. U něho nalezne jeho veřejný klíč *(VK)*, který použije k ověření podpisu *SIGNED_CHALLANGE.* Pro zajištění vyšší bezpečnosti by server mohl v tomto kroku ještě ověřit platnost přijatého veřejného klíče *(VK)*, resp. certifikátu uživatele *(CRT).* K ověření by použil veřejný klíč certifikační autority (*VKCA*). Pokud dojde k úspěšnému ověření podpisu i pravosti certifikátu, lze autentizaci uživatele považovat za úspěšnou.

7.3.2.1 Možnost off-line autentizace

V rámci autentizace s využitím asymetrické kryptografie lze uvažovat i o variantě s off-line ověřením identity uživatele. V tomto případě by musel být na straně každého platebního terminálu uložen veřejný klíč certifikační autority *(VKCA)*. Mobilní telefon by v okamžiku přenášení podepsané výzvy S*IGNED_CHALLANGE* zasílal zpět terminálu ještě svůj veřejný klíč, resp. svůj certifikát *(CRT).* Ten byl již dříve v průběhu registrace uživatele podepsán soukromým klíčem certifikační autority *(SKCA).* Terminál by tak mohl použít *veřejný klíč* (*VKCA*) certifikační autority k ověření podpisu přijatého uživatelova certifikátu. Pokud by byl podpis ověřen, může důvěřovat danému uživateli. Kontrola by proto proběhla „uvnitř“ terminálu, nikoliv online na serveru.

7.4 Uložení citlivých dat v mobilní aplikaci

Hodnoty, které uživatel obdrží ze serveru je třeba na straně mobilní aplikace uložit do interní paměti. To platí jak u varianty symetrické autentizace, tak u varianty asymetrické autentizace. Uložit tyto hodnoty jako obyčejný plaintext je z hlediska bezpečnosti nevhodné (viz kapitola 4.2 Ukládání citlivých dat). V průběhu návrhu aplikace byly zváženy všechny možnosti ukládání citlivých dat. Jako nejdostupnější a nejvhodnější byla pro implementaci nakonec zvolena varianta interního úložiště mobilního telefonu. Takto uložená data podléhají ochraně, kterou poskytuje aplikační Sandbox. Žádná další aplikace k nim standardně nemá přístup. Navíc před samotným uložením dat probíhá jejich šifrování na základě algoritmu RSA. Tato metoda vyžaduje dvojici klíčů (soukromý a veřejný), které jsou generovány v průběhu registrace uživatele na straně mobilního telefonu. Jako úložiště klíčů je využit Android Keystore (viz kapitola 4.2.3.1 Způsoby uložení citlivých dat).



Obrázek 35: Informace o hardwarovém úložišti kryptografických klíčů [autor]

Na testovaném telefonu Samsung Galaxy (A4) s operačním systémem Android 5.0.2 navíc dochází k uložení klíčů do hardwarového úložiště „Hardware Backed“, které poskytuje mnohem vyšší bezpečnost než jeho softwarová varianta.

8 Shrnutí výsledků

Bezpečnost celého systému spočívá v míře zabezpečení úložiště klíčů u obou variant. Jejich únik nebo prozrazení by ohrozilo bezpečnost celé aplikace. Pro obě varianty bylo nakonec použito interní úložiště mobilního telefonu s využitím šifrování.

Z hlediska bezpečnosti vítězí varianta autentizace založená na asymetrické kryptografii s využitím hardwarového úložiště klíčů. Pro autentizaci je použit soukromý klíč uživatele, který je společně s veřejným klíčem bezpečně uložen v Android Keystore. V případě, že by se útočníkovi podařilo získat přístup k Android Keystore, může použít dvojici klíčů, nikoliv je však exportovat a zařízení tím klonovat nebo jinak zneužít. Tato metoda tedy splňuje všechny požadavky práce, navíc je vhodná i pro provedení off-line autentizace. Nevýhoda této varianty spočívá v náročnější implementaci aplikace a nutnosti kontrolovat a zajišťovat životní cyklus všech certifikátů. Pro případ rozšíření této metody i pro autentizaci vůči třetím stranám je nutné, aby všechny certifikáty pocházely skutečně od důvěryhodné certifikační autority.

Autentizační metoda s využitím symetrického klíče používá pro autentizaci uživatele jediný klíč *HmacSHA256key*. Ukládání symetrického klíče do úložiště Android Keystore umožňuje až API 23, které odpovídá verzi Android 6.0 (Marshmallow) uveřejněné koncem roku 2015. V době psaní této práce bylo na trhu dostupných pouze 4,7 %[15] zařízení se zmíněnou verzí Androidu. Z tohoto důvodu byla použita varianta uložení symetrického klíče do interní paměti (mimo oblast Keystore). Ještě před samotným uložením je klíč šifrován pomocí algoritmu RSA. Získá-li však útočník přístup k šifrovacím klíčům, může tyto klíče použít a získat tak plný přístup k symetrickému klíči *HmacSHA256key.* Symetrická varianta je sice jednodušší z hlediska implementace, neposkytuje ale dodatečnou ochranu proti vyjmutí symetrického klíče.

Bezpečnost obou variant lze dále hodnotit nalezením odpovědí na tři otázky týkajících se integrity, autentizace a nepopiratelnosti.

[15] Údaj převzatý z Android Studia (červen 2016).

- *Integrita* - Může si být příjemce jistý, že zpráva nebyla cestou změněna?
- *Důvěrnost* (autenticita) - Může si být příjemce jistý, že zpráva pochází od odesílatele, který se za odesílatele označuje a kterému příjemce důvěřuje?
- *Nepopiratelnost (non-repudiation)* - Pokud příjemce pošle zprávu třetí osobě, může si být tato osoba jista, že pochází skutečně od původního odesílatele?

Odpovědi na tyto otázky poskytuje již samotná podstata symetrické a asymetrické kryptografie. Lze je znázornit v následující tabulce:

Tabulka 2: Porovnání autentizačních metod [autor]

	Symetrická kryptografie (HMAC)	Asymetrická kryptografie (PKI)
Integrita	+	+
Důvernost	+	+
Nepopiratelnost	-	+

Šifrování zajišťuje důvěrnost (autenticitu) přenášených dat. Vedle toho podepisování zajišťuje nepopiratelnost a integritu. Integrita je u obou variant splněna. Aplikace skutečně ověřuje, zda nedošlo ke změně zprávy na cestě k příjemci (platebnímu terminálu). Integrita přijaté zprávy je nicméně ověřena až online na straně serveru, který disponuje stejným symetrickým, resp. veřejným klíčem uživatele. Stejně tak obě varianty splňují požadavek autentizace. Každý uživatel se prokazuje znalostí klíče, kterým následně podepisuje výzvu zaslanou platebním terminálem.

Jak již ale bylo zmíněno v kapitole 2.3.1 Algoritmus HMAC, u symetrické kryptografie nelze zcela hovořit o digitálním podpisu, jelikož tato metoda nesplňuje nepopiratelnost a tento podpis proto nelze důvěryhodně použít vůči třetí straně. Z celkového hodnocení opět vychází lépe varianta asymetrické kryptografie, která splňuje všechny tři požadavky.

Kapitola 5.3 Útoky na NFC se zabývala možnými útoky na technologii NFC. Jak již bylo v této kapitole zmíněno, technologie NFC má ve své podstatě nezabezpečenou komunikaci a nemůže se proto bránit odposlechu přenášených

dat. U obou variant implementovaných autentizačních mechanizmů dochází po celou dobu transakce k měnícímu se řetězci binárních dat. Ty jsou navíc výsledkem „podpisu“ transakce pomocí algoritmu HMAC u symetrické varianty, respektive digitálního podpisu u asymetrické varianty. Pro případného útočníka je proto nemožné z nich i přes opakovaný odposlech sestavit původní hodnoty. Bezpečnost je navíc ještě zvýšena tím, že tajné klíče nikdy neopustí své úložiště v telefonu a při bezkontaktním přenosu nejsou přenášeny.

Zajímavé jsou také výsledky měření rychlosti provedení autentizace. Začátek měření byl stanoven na okamžik přiložení telefonu k terminálu, přesněji řečeno na okamžik zaslání prvního APDU příkazu „SELECT AID“ ze strany terminálu. Konec měření byl stanoven na ukončení přenosu dat mezi terminálem a mobilním telefonem, čemuž u obou variant odpovídá krok 4. Do času měření se dále již nepočítala doba, kterou terminál potřebuje pro kontaktování serveru se žádostí o ověření kryptogramů. Ta by výsledky mohla výrazně ovlivňovat v závislosti na kvalitě připojení. Navíc v tomto okamžiku uživatel již nemusí telefon držet v radiovém poli terminálu. Následující hodnoty proto odpovídají době potřebné pro přenos dat mezi terminálem a mobilním telefonem:

Tabulka 3: Naměřené hodnoty rychlosti autentizace [autor]

	Symetrická kryptografie (HMAC)	Asymetrická kryptografie (PKI)
Rychlost autentizace	427,9ms	579,8ms

V tomto bodě je také důležité zmínit, že práce se zabývala autentizací pouze ze strany mobilního telefonu vůči platebnímu terminálu. V reálném nasazení by bylo nutné použít stejné principy i pro autentizaci ze strany platebního terminálu vůči mobilnímu telefonu a také autentizaci platebního terminálu vůči platebnímu serveru.

9 Závěr

Za posledních pět let můžeme zaznamenat výrazný nárůst platebních karet, které pro bezkontaktní přenos dat využívají pásmo 13,56 MHz. Tento trend lze spatřit i u mobilních telefonů, které pro bezkontaktní přenos využívají rozhraní NFC (Near Field Comunication) rovněž v pásmu 13,56 MHz. Technologie NFC slouží obecně pro komunikaci mezi dvěma zařízeními na krátkou vzdálenost čítající jednotky centimetrů. Jednou z možností využití NFC rozhraní je emulace čipové karty mobilním telefonem.

U karetních aplikací hraje důležitou roli bezpečnost jednotlivých komponent systému. Čip osazený na plastové kartě je ve své podstatě mikroprocesor, který vykonává kryptografické operace a zároveň slouží jako bezpečné úložiště klíčů. Čipové karty jsou regulovány mezinárodními předpisy a normami prokazující vysokou míru bezpečnosti (ISO / IEC 15408 známé jako Common Criteria, FIPS-140[16], EMV norma pro Europay, MasterCard a VISA). U mobilních telefonů je situace složitější. Bezpečné (hardwarové) úložiště, srovnatelné s bezpečností čipových karet, lze nalézt pouze v podobě Secure Elementu. Ten v dnešní době můžeme nalézt prakticky pouze v SIM kartě, kam běžný vývojář bez znalosti klíčů nemá přístup.

Nejdále v implementaci rozhraní NFC je mobilní platforma Android, která od verze 4.3 poskytuje API k softwarovému úložišti klíčů s názvem KeyStore. Data uložená v tomto úložišti jsou šifrována pomocí AES master klíče, který je dále šifrován klíčem odvozeného ze zámku obrazovky. Android poskytuje ochranu proti náhodnému zadávání pinu nebo hesla pro odemknutí telefonu v podobě maximálního počtu pokusů. Po překročení limitu dojde ke smazání obsahu telefonu. Tímto je zajištěna dostačující bezpečnost.

Problém však nastává v případě, kdy útočník získá root oprávnění k telefonu. Časté jsou i případy, kdy root oprávnění povolí uživatel na základě vlastního rozhodnutí. Nepoužije-li vlastník telefonu složitější heslo, pak útočník je schopen metodou brute force prolomit zamknutí obrazovky za poměrně krátkou dobu.

[16] Federální norma pro zpracování informací

Hlavním cílem práce proto bylo nalézt optimální rovnováhu mezi bezpečností a pohodlím uživatele při užívání mobilní aplikace. V rámci zabezpečení aplikace byly navrhnuty a implementovány dva autentizační mechanizmy. První na bázi symetrické kryptografie, druhý na bázi asymetrické kryptografie.

V první části práce byla analyzována bezpečnost systému Android se zaměřením na poskytované možnosti úložného místa pro bezpečnostní klíče. Ve druhé části již byly popsány dvě konkrétní implementace platebního systému s cílem zvolit vhodnější variantu. Bez ohledu na typ použitého šifrování, celá bezpečnost aplikace je v prvé řadě závislá na již zmíněném úložišti klíčů. Jako nejvhodnější varianta úložiště byla zvolena softwarová varianta s názvem Android Keystore, která navíc od verze Android 4.3 ještě přidala možnost skrze stejné API vložit klíče do Hardware-Backed úložiště. Toto úložiště je součástí procesoru mobilního zařízení a zaručuje, že veškeré kryptografické operace probíhají v chráněném prostředí mimo hlavní OS. Klíče jsou uloženy šifrovaně a nikdy toto úložiště neopustí. Pokud útočník použije root oprávnění, je schopen dané klíče použít, ale nikoliv získat a použít na jiném zařízení. Tento způsob útoku by měl za následek vytvoření libovolného počtu kopií telefonu se stejnými platebními údaji uživatele. Z hlediska bezpečnosti proto zvítězila varianta založená na asymetrické kryptografii s využitím hardwarového úložiště klíčů. Ta jako jediná používá pro autentizaci soukromý klíč uživatele, který je společně s veřejným klíčem bezpečně uložen v Android Keystore. V případě, že by se útočníkovi podařilo získat přístup k Android Keystore, může použít dvojici klíčů, nemůže je však exportovat a zařízení klonovat nebo jinak zneužít.

Autentizační metoda s využitím symetrického klíče nesplnila požadavek bezpečnosti. Tato metoda používá pro autentizaci uživatele jediný symetrický klíč *HmacSHA256key*, který na rozdíl od předchozí varianty nelze uložit do bezpečného úložiště Android Keystore. Ukládání symetrických klíčů do úložiště Android Keystore umožňuje až API 23, které odpovídá verzi Android 6.0 (Marshmallow) uveřejněné koncem roku 2015. Z tohoto důvodu byla použita varianta uložení symetrického klíče do interní paměti (mimo oblast Keystore). Získá-li však útočník přístup k šifrovacím klíčům, může tyto klíče použít a získat tak plný přístup k symetrickému klíči HmacSHA256key. Symetrická varianta je sice jednodušší

z hlediska implementace, neposkytuje ale dodatečnou ochranu proti vyjmutí symetrického klíče.

Existuje i možnost neukládat bezpečnostní klíče v mobilním zařízení vůbec a uchovávat je na vzdáleném serveru. Být v kontaktu se vzdáleným úložištěm ale vyžaduje mít telefon trvale připojen k internetu i v době provedení autentizace nebo těsně před ní. V případě výpadku signálu nebo nedostupného internetového připojení by platební transakce nemohla být realizována. Tento způsob autentizace lze považovat za uživatelsky nepřívětivý a jeho implementace nebyla proto v práci realizována.

Představa, že v mobilním telefonu budeme brzy nosit hned několik platebních a věrnostních karet, je lákavá. Nese sebou ovšem bezpečnostní rizika, kterými se tato práce zabývala. Ve výsledku je proto při vývoji aplikace potřeba najít kompromis mezi pohodlím uživatele a úrovní zabezpečení. Bude-li uživatel dodržovat několik bezpečnostních pokynů jako je např. zachování originální operačního systému, neinstalovat aplikace z neznámých zdrojů, pak nemusí mít o svá data obavy. Navrhnutý systém rovněž umožňuje telefon kdykoliv zablokovat v případě ztráty nebo odcizení stejným způsobem, na jaký je uživatel zvyklý při užívání bankovní karty. Při dodržení všech těchto podmínek a při zodpovědném nakládání s aplikacemi mobilního telefonu, lze emulaci karty mobilním telefonem považovat za bezpečnou.

10 Seznam použité literatury

[1] **PAAR, Christof a PELZL, Jan.** *Understanding Cryptography.* Berlin : Springer-Verlag, 1998. ISBN 978-3-642-44649-8.

[2] **DOSEDĚL, Tomáš.** *Počítačová bezpečnost a ochrana dat.* Brno : Computer Press, 2004. ISBN 80-251-0106-1.

[3] **DOSTÁLEK, Libor a VOHNOUTOVÁ, Marta.** *Velký průvodce infrastrukturou PKI a technlogií elektronického podpisu.* Brno : Computer Press, a.s., 2006. ISBN 80-251-0828-7.

[4] **IGNJATOVIČ, Martin.** Šifry kolem nás - Díl první - přehled algoritmů. *pcworld.cz.* [Online] IDG Czech Republic, a. s. [Citace: 6.5.2016] http://pcworld.cz/internet/sifry-kolen-mas-dil-prvni-prehled-algoritmu-12983.

[5] **MENEZES, Alfred J., OORSCHOT, Paul C. van a VANSTONE, Scott A.** *Handbook of Applied Cryptography.* Boca Raton : CRC Press, 1996. ISBN 0-8493-8523-7.

[6] **KLÍMA, Vlastimil.** Eliptické křivky a šifrování. *crypto-world.info.* [Online] [Citace: 4.4.2016.] http://crypto-world.info/klima/2002/chip-2002-09-134-136.pdf.

[7] **Český normalizační institut.** Identifikační karty - Fyzikální charakteristiky. *nahlay.normy.biz.* [Online] Český normalizační institut, 1.8.2004 [Citace: 5.3.2014] http://nahledy.normy.biz/nahled.php?i=70891.

[8] **JUŘÍK, Pavel.** *Svět platebních karet.* Praha : RADIX, sopl. s r.o., 1995. ISBN 80-901853-1-2.

[9] **RANKL, Wolfgang a EFFING, Wolfgang.** *Smart Card Handbook.* Munich/FRG : John Wiley & Sons, Ltd, 2010. ISBN 978-0-470-74367-6.

[10] **SILNÝ, Martin.** *Čipové karty - demonstrační aplikace: Bakalářská práce.* Praha : České vysoké učení technické v Praze, 2007.

[11] **Research and Markets.** Contactless Payments Market by Solutions & Services. *Businesswire.com.* [Online] 22.1.2014 [Citace: 7.3.2016] http://www.businesswire.com/news/home/20140122005678/en/Research-Markets-Contactless-Payments-Market-Solutions-Services%20-%20.UxG5dWePK70#.UxmpvrCPIdU.

[12] **KOETSIER, John.** 'Contactless' payment to reach 250M cards next year (U.S.A. is last, as usual). *Venturebeat.com.* [Online] 19.11.2013 [Citace: 7.3.2016]

http://venturebeat.com/2013/11/19/contactless-payment-to-reach-250m-cards-next-year-u-s-a-is-last-as-usual/.

[13] **VOJTĚCH, Lukáš.** RFID - technologie pro internet věcí. *Pandatron.cz.* [Online] 14.4.2009 [Citace: 16.8.2013] http://pandatron.cz/?733&rfid_-_technologie_pro_internet_veci.

[14] **ROSENBERG, Martin a MERTLÍK, Tomáš.** *Technologie NFC - popis, bezpečnost a využití.* Brno : Elektrorevue, 2013, Sv. 15. ISSN 1213-1539.

[15] **ELENKOV, Nikolay.** *Android Security Internals: an in-depth guide to android's security architecture.* San Francisco : No Starch Press, Inc., 2015. ISBN-10: 1-59327-581-1.

[16] **BOWN, Scott Alexander.** Android Security: Adding Tampering Detection to Your App. *Airpair.* [Online] [Citace: 7.5.2016] https://www.airpair.com/android/posts/adding-tampering-detection-to-your-android-app.

[17] **COSKUN, Vedat, OZDENIZCI, Busra a OK, Kerem.** *A Survey on Near Field Communication (NFC) Technology.* New York : Springer Science + Business Media, 2012. 71:2259–2294.

[18] **Google, Inc.** Host-based Card Emulation. *Developers Android.* [Online] Google, Inc. [Citace: 9. 5. 2016] http://developer.android.com/guide/topics/connectivity/nfc/hce.html.

[19] **ROLAND, Michael a LANGER, Josef.** Comparison of the usability and security of NFC's different operating modes in mobile devices. *Elektrotechnik & Informationstechnik.* 2013, 130/7.

[20] **HASELSTEINER, Ernst a BREITFUß, Klemens.** Security in Near Field Communication (NFC). [Online] [Citace: 12. 4. 2016] http://events.iaik.tugraz.at/RFIDSec06/Program/papers/002%20-%20Security%20in%20NFC.pdf.

11 Zadání práce

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2015/2016

Studijní program: Aplikovaná informatika
Forma: Kombinovaná
Obor/komb.: Aplikovaná informatika (ai2-k)

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Bc. Brož Jan DiS.	Jeronýmova 539, Vysoké Mýto - Litomyšlské Předměstí	I14279

TÉMA ČESKY:

Emulace čipové karty pomocí technologie NFC

TÉMA ANGLICKY:

Emulation of smart cards by using NFC

VEDOUCÍ PRÁCE:

doc. Ing. Filip Malý, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cílem práce je vytvoření nezávislého platebního systému, ve kterém bude mobilní telefon vystupovat v roli bezkontaktní platební karty. Pro autentizaci uživatele a přenos transakčních dat bude využito rozhraní NFC v režimu Host Card Emulation (HCE). V rámci zabezpečení aplikace bude navrhnuto a implementováno několik autentizačních mechanizmů. Cílem je vybrat nejvhodnější variantu z hlediska bezpečnosti a pohodlí uživatele.

Osnova:
1. Úvod
2. Základní pojmy a principy z oblasti kryptologie
3. Analýza čipových karet
4. Platforma Android a rozhraní NFC
5. Známé útoky na bezpečnost dat
6. Návrh platebního systému
7. Implementace a testování
8. Závěr

SEZNAM DOPORUČENÉ LITERATURY:

MAYES, Keith a MARKANTONAKIS, Konstantinos. Smart Card, Tokens, Security and Applications.
JUŘÍK, Pavel. Svět platebních karet.
RANKL, Wolfgang a EFFING, Wolfgang. Smart Card Handbook
PIPER Fred, MURPHY Sean. Kryptografie, Průvodce pro každého
PAAR Christof, PELZL Jan. Understanding Cryptography
DOSTÁLEK Libor, VOHNOUTOVÁ Marta. Velký průvodce infrastrukturou PKI
GUNASEKERA Seran A. Android Apps Security
SCOTT Alexander-Bown. Android Security: Adding Tampering Detection to Your App.

Podpis studenta: Jan Brož ..... **Datum:** 10.10.2015

Podpis vedoucího práce: Malý ..... **Datum:** 10.10.2015