VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Minh Tran

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

BEZPEČNOST AUTENTIZAČNÍCH PROTOKOLŮ
SECURITY OF AUTHENTICATION PROTOCOLS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Minh Tran
AUTHOR

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

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Minh Tran ***ID:*** 196013
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Bezpečnost autentizačních protokolů

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s vývojem Android mobilních aplikací a s komunikační technologií NFC. Dále analyzujte a popište současné bezpečnostní hrozby současných kartových systémů (přístupové systémy, platební systémy apod.) a možnosti využití platformy mobilních telefonů s podporou NFC/HCE pro realizaci těchto útoků. Na tomto základě implementujte demonstrátor bezpečnostních hrozeb kartových autentizačních systémů využívající Android chytrých telefonů.

DOPORUČENÁ LITERATURA:

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

[2] Android Developers [online]. Google [cit. 2020-09-14]. Dostupné z: https://developer.android.com/

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

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

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

ABSTRAKT

Tato práce řeší bezpečnost autentizačních protokolů. Cílem práce je analýza současných bezpečnostních hrozeb a útoků na současné kartové systémy především JavaCard a Mifare. A dále vytvořit mobilní aplikaci pomocí Android platformy s funkcemi NFC a HCE, která realizuje nalezené útoky. Těmito útoky jsou útok přeposláním a klonování karet. Mobilní část aplikace je psaná v programovacím jazyce Kotlin a serverová část v JavaScript, EJS a CSS. V závěru jsou předvedeny útoky na EMV a přístupový systém nejmenované univerzity.

KLÍČOVÁ SLOVA

autentizace, útok přeposláním, klonování, Android, Kotlin, NFC, HCE, EMV, Mifare, Java Card

ABSTRACT

This thesis deals with the security of authentication protocols. The aim of the thesis is to analyze current security threats and attacks on current card systems mainly JavaCard and Mifare. And then to create a mobile application using Android platform with NFC and HCE functions, which realises the founded attacks. These attacks are relay attack and card cloning. Mobile side of the application is written in Kotlin programming language and server side in JavaScript, EJS and CSS. In the end attacks on EMV and access system of an unnamed university are demonstrated.

KEYWORDS

authentication, relay attack, cloning, Android, Kotlin, NFC, HCE, EMV, Mifare, Java Card

TRAN, Minh. *Bezpečnost autentizačních protokolů*. Brno, 2021, 77 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Petr Dzurenda, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Bezpečnost autentizačních protokolů“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Petrovi Dzurendovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Technologie čipových též chytrých karet u autentizačních protokolů je v dnešní době bez pochyb jednou z nejpoužívanějších, se kterými se člověk každodenně setkává. Naprostá většina lidí může říci, že vlastní nějakou kartu či čip, kterými prokazuje svoji identitu vůči terminálu. Nejjednodušším příkladem jsou přístupové karty. Stačí vzít danou přístupovou kartu a přiblížit ji k terminálu a dveře se otevřou. Tento systém autentizace je hojně využíván u vstupů do větších budov, kde sídlí například firmy nebo školy. Tyto instituce musí chránit svoje fyzická aktiva, know-how a citlivá data. Nekontrolovaný přístup by mohl být katastrofální. Dále je třeba se zmínit o kartách, které přímo pracují s aktivy v podobě peněžních prostředků. Příkladem jsou platební karty bank jako Visa a MasterCard, některé cestovní karty, kam lze nahrát částku peněz nebo i jídelní čipy.

Dle článku [1] je celkově v oběhu kolem 50 miliard kusů chytrých karet. V roce 2019 se vydalo přes jednu miliardu těchto zařízení. Velmi podobná čísla následovala i v roce 2020. Využití této technologie se v mnoha oborech jen zvyšuje. Například státní systémy elektronických identifikací a zdravotnictví předpokládají do konce roku 2020 růst o 7,4 %. Ve finanční sféře za rok 2019 byl zaznamenám růst o 24,4 %. Do konce roku 2020 i přes celosvětovou krizi kvůli COVID-19 se předpokládá růst o dalších 6,3 %. Předpokládá se, že v roce 2023 bude trh s chytrými kartami a zařízeními s nimi blízce spojenými činit kolem 21 miliard dolarů. Lze tedy předpokládat, že technologie chytrých karet a systémy na nich založené budou stále velmi aktuálním tématem. Je jasné, že kvůli tak velké poptávce po chytrých kartách, musí být tyto technologie co nejlépe zabezpečené, téměř neprůstřelné, proti nekalým činnostem zločinců. Je zde nějaký prostor pro obavy ze zneužití těchto technologií?

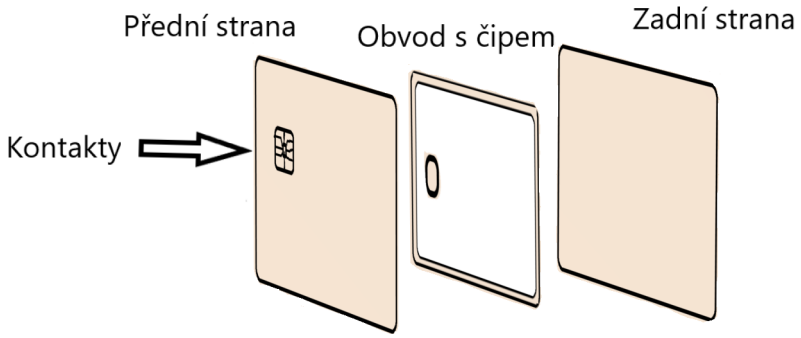
Technologie čipových karet je kompatibilní například s některými mobily s Android OS (Operating System). Technologiím zařizujícím tuto kompatibilitu se říká NFC (Near Field Communication) a HCE (Host-based Card Emulation). NFC dává telefonu schopnost komunikovat s kartou, číst ji a psát do ní. Může zastávat roli terminálu. A HCE dává telefonu schopnost emulovat kartu. Zde nastávají problémy s bezpečností čipových karet. Vývojáři kartových autentizačních systémů si jsou možností nekalých aktivit velmi dobře vědomi a snaží se data šifrovat, vylepšit stávající systémy novými bezpečnostními prostředky nebo vydávat nové bezpečnější karty. Uživatelé by si měli dávat pozor na existenci bezpečnostně nedostačujících karet a aktualizovat na bezpečnější systémy. V praxi tato myšlenka ale příliš nefunguje.

Tato práce se věnuje oblasti bezpečnosti těchto technologií, zejména možnostem útoků, které v dnešní době hrozí. Přesněji se bude práce zabývat funkcemi mobilů s Android OS podporující funkce NFC a HCE. Pomocí těchto funkcí bude snaha překonat bezpečnost například přístupových systémů, platebních systémů apod.

1 Čipové karty

Čipové karty jsou přenosná fyzická zařízení, ve kterých se nachází integrovaný obvod s čipem. Tento obvod chrání tělo karty, které je obvykle vyrobeno z plastového materiálu [2]. Stavební charakteristiky těchto karet, jako jsou například rozměry, materiál, odolnost vůči teplotě, chemikáliím, vlhkosti a dalším vnějším vlivům, jsou standardizované a popsané v ISO/IEC 7810:2019 [3].

Díky integrovanému obvodu má karta schopnost zpracovávat data. Do karty je možno psát data, číst z ní data, ukládat data nebo je posílat dále do jiného zařízení. S kartou lze komunikovat dvěma hlavními způsoby. Prvním způsobem je kontaktní komunikace pomocí kontaktů na jejím povrchu. Takovou kartu popisuje standard ISO/IEC 7816 od části 1 až po část 15, kde jsou uvedeny informace od fyzické charakteristiky přes komunikační protokoly až po kryptografické informace uložené na kartě. Druhá možnost je bezkontaktní komunikace přes standard připojení krátkého dosahu jako je NFC, která pracuje na frekvenci 13,56 MHz [4] nebo RFID (Radio Frequency Identification), u kterého se velmi často používají karty pracující na frekvenci 125 kHz [5]. Bezkontaktní karty jsou popsány ve standardu ISO/IEC 14443, který je rozdělen na čtyři části, kde jsou definované fyzické charakteristiky, radiová frekvence, signálové rozhraní, inicializace, antikolize a protokol přenosu. Z důvodu neshod výrobců, které nastaly v počátcích vývoje čipových karet, je ISO/IEC 14443 rozdělen na dva typy a to na typ A a na typ B. Jejich rozdílem jsou modulační metody popsané v [6]. Kombinací kontaktní a bezkontaktní technologie komunikace karet vzniká hybridní (duální) způsob komunikace. Karta má tedy schopnost komunikovat kontaktně i bezkontaktně [2]. Na obrázku 1.1 je možné vidět příklad stavby čipové karty schopné hybridní komunikace. Obrázek ukazuje integrovaný obvod s čipem, který umožňuje bezkontaktní komunikaci a který je z obou stran chráněn plastovou konstrukcí. Navíc obsahuje kontakty na povrchu pro schopnost kontaktní komunikace.



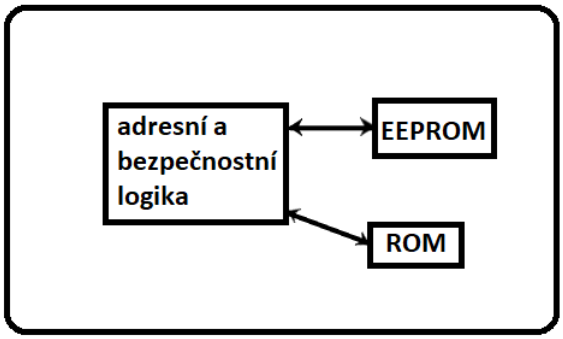
Obr. 1.1: Struktura hybridní čipové karty.

1.1 Typy čipových karet dle integrovaného obvodu

Čipové karty jsou kategorizované podle funkcí integrovaného čipu, které vyplývají ze stavby samotného čipu. [2].

1.1.1 Paměťové karty

Nejjednodušším typem čipových karet jsou karty paměťové. Obsahem karty je EEPROM (Electrically Erasable Programmable Read-Only Memory), ve které jsou uložená data, a ROM (Read-Only Memory), kde jsou uloženy komunikační protokoly karty. Data v ROM nelze již po výrobě změnit. S tímto druhem karty lze provádět jen lokální psaní dat do karty a čtení dat z ní. Data na kartě mohou být formátována jen pro čtení a tudíž s ní nelze provádět jiné operace než čtení. Na obrázku 1.2 je popsána paměťová stavba paměťové karty.



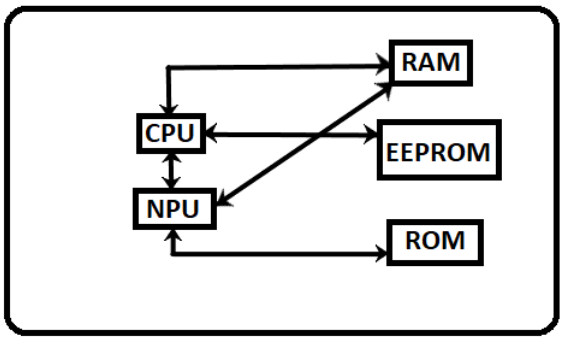
Obr. 1.2: Struktura paměťové karty.

Použitím těchto karet mohou být například věrnostní karty, kde se načítají a odečítají body nebo jednoduché přístupové karty, kde se načte hodnota na kartě pro ověření identity a nebo předem nabité karty, ze kterých se čerpají nahrané body a po vyčerpání všech bodů jsou karty vyhozeny nebo zlikvidovány.

1.1.2 Mikroprocesorové karty

Na rozdíl od paměťových karet mají mikroprocesorové karty navíc CPU (Central Processing Unit) a RAM (Random-Access memory). Paměťová stavba karty je popsána na obrázku 1.3 na následující straně. Díky těmto přídavným prvkům mohou nejen psát, číst a ukládat data, ale také je jakkoli zpracovávat. Zde jsou již karty označovány jako chytré karty (ang. Smart Cards). K základnímu procesoru může mít karta i koprocesor NPU (Numeric Processing Unit), který umožňuje další přídavné funkce. Například kryptografické koprocesory přináší kartám bezpečnostní technologie jako jsou kryptografické algoritmy AES (Advanced Encryption Standard), RSA

(Rivest–Shamir–Adleman) a ECDSA (Elliptic Curve Digital Signature Algorithm) pro šifrování a dešifrování dat.



Obr. 1.3: Paměťová stavba mikroprocesorové karty.

Užití mikroprocesorových karet je velmi široké. Příkladem pro použití těchto karet jsou platební karty, autentizační karty, identifikační karty či kryptografické karty. V dnešní době se hojně využívají programovatelné karty se svým vlastním OS, do kterých je možno nahrávat vlastní aplikace. Příkladem těchto technologií je Java Card, která je popsána v následující podkapitole 1.2. Další často používanou technologií je MULTOS nebo také Basic Card [2].

1.2 Programovatelné čipové karty Java Card

Jednou z nejvíce používaných technologií, která se využívá u čipových karet, je Java Card. Tato technologie používá podmnožinu jazyka Java pro vývoj aplikací čipových karet. V dnešní době se tato technologie často využívá v platebních kartách firem MasterCard nebo VISA. SIM karty jsou také obvykle implementovány pomocí Java Card technologie. Elektronické identifikační systémy jako například pasy, občanské průkazy a řidičské průkazy používají tuto technologii rovněž [2]. Na obrázku 1.4 na následující straně jsou vyfoceny příklady čipových karet s Java Card technologií. Vlevo na obrázku se nachází školní karta podporující více typů technologií, mezi kterými je i Java Card. Uprostřed na obrázku leží klasická debetní karta a vpravo na obrázku leží SIM karta standardní velikosti.

1.2.1 Struktura Java Card

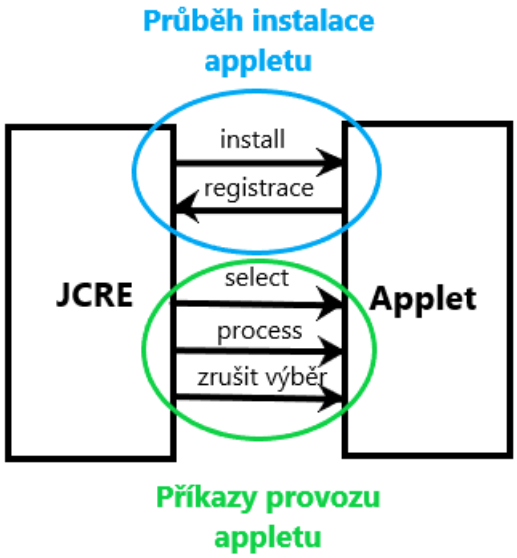
Technologie Java Card obsahuje tři části. První z nich je JCVM (Java card virtual machine), která umožňuje kartě pracovat se zmíněnou podmnožinou jazyka Java. Druhá část jsou API (Application Programming Interface) představující rozhraní

Obr. 1.4: Příklady Java Card.

Java balíčků. Třetí část se nazývá JCRE (Java Card Runtime Environment), která spojuje JCVM, API a možná přídavná rozšíření. Obvyklá Java karta je postavená na osmi nebo šestnácti bitovém CPU pracujícím rychlostí 3,7 MHz. Velikost RAM má 1 Kb a velikost EEPROM alespoň 16 Kb [7].

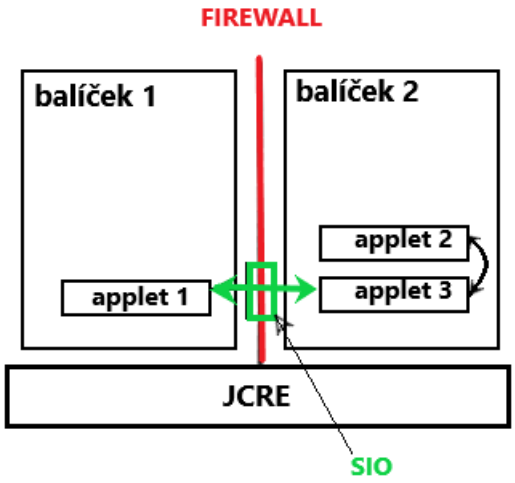
Applety se nazývají aplikace, které jsou nahrávány na čipové karty a spravuje je JCRE. Applet vzniká ze souborů se zdrojovým kódem naprogramovaných na JVC (Java virtual machine), ta obvykle běží na počítačích; soubory jsou dále konvertovány na CAP (Converted Applet) soubor. Tento soubor se nakonec nainstaluje do karty a v kartě je interpretován díky JCVM. Každý applet má svůj unikátní AID (Application Identifier), jak je definováno v ISO/IEC 7816-5 [8]. Ve zdrojovém kódu appletu se nacházejí funkce `install()` a `process()`. Funkce `install()` je volána JCRE při instalování appletu do karty. Funkce `process()` je volána, když je applet podle svého AID vybrán příchozím příkazem select APDU (Application Protocol Data Unit) a značí, že má applet provést úkony podle příchozího příkazu APDU [7]. Přidělení unikátního AID je klíčové pro chod systému z toho důvodu, aby applety mezi sebou nekolidovaly. Na obrázku 1.5 je popsán životní cyklus appletu od jeho nainstalování až po vlastní provoz.

Jedním z bezpečnostních mechanizmů Java Card je BCV (Bytecode Verification), jehož funkcí je ověřovat správnost Java kódu appletu (CAP). Kontrola je provedena před tím, než je applet nainstalován na kartu. Díky tomuto by měla být paměť karty chráněna před chybami, které by nesprávný kód mohl způsobit. Tato operace je ale velmi náročná na paměťovou velikost kódu. Z toho důvodu většina karet do verze Java Card 3 v sobě nemá BCV službu integrovanou [9]. Verifikace je v této situaci prováděna zvenčí karty. Jako kontrola může sloužit podepsaná CAP od důvě-



Obr. 1.5: Životní cyklus appletu.

ryhodné třetí strany. Další bezpečnostní mechanizmus zde použitý je firewall. Java Card má svůj vlastní firewall [2], jehož funkcí je zabraňovat vzájemnému vstupu appletů do jejich paměti. Z jednoho appletu tedy nelze číst ani psát na druhý, není-li to explicitně povoleno mezi applety. Sám applet tedy může vstoupit jen do své vlastní paměti nebo tam, kde má povoleno. JCRE, který se stará o přístup k appletům, může na druhou stranu vstupovat do jakékoli paměti na kartě. Na rozdíl od BCV firewall provádí svoje kontroly za běhu programu. Firewall navíc přidává další bezpečnostní prostředky, které BCV postrádá. V případě, že je chybně navržena ochrana dat v appletu - například tak, že má nastavené nějaké proměnné jako veřejné, tudíž ostatní applety mohou tyto data číst, tak i přes tuto chybu firewall nepovolí neoprávněný vstup. Výjimkou je vlastnost v Java Card API tzv. SIO (Shareable Interface Object). Díky tomuto rozhraní mohou applety mezi sebou bezpečně komunikovat. Applety mají přístup jen k objektům a funkcím, které jsou definovány v SIO. Lze tedy používat funkce jiných appletů, které jsou definovány v SIO. Všechna data SIO jsou, stejně jako samotné applety, chráněné systémem firewall. V souhrnu Java Card Firewall garantuje kontrolovaný přístup k appletům a jejich datům, ale za předpokladu, že jsou applety správně napsané. Strukturu zapouzdření appletů pomocí firewallu znázorňuje obrázek 1.6. Jenom applety, které se nacházejí ve stejném balíčku, mohou navzájem vstupovat do svých dat. Applety z různých balíčků mohou jen v případě, že jsou v rozhraní SIO.



Obr. 1.6: Java Firewall.

1.2.2 Komunikace Java Card

Součástí komunikace jsou dvě stanice [7]. Na jedné straně je zařízení CAD (Card Acceptance Device), na kterém běží aplikace snažící se aktivně navázat kontakt s appletem na kartě. Tímto zařízením může být platební terminál, čtečka karet nebo také mobilní telefon. Na druhé straně stojí čipová karta, na které je nainstalován jeden applet nebo více appletů. Karta je pasivní a čeká na kontakt od CAD, do té doby nic nedělá. Komunikaci zařizují příkazy a odpovědi APDU. Prvním krokem pro započetí komunikace je poslání APDU požadavku z CAD do karty pro výběr appletu. Applet se vybírá podle jeho unikátního AID. Po úspěšném výběru se již posílají příkazy, ve kterých je uvedeno, co se po appletu chce. Applet zpátky posílá APDU odpověď podle toho, jak příkaz zpracoval.

APDU [7] je klíčovým prvkem komunikace chytrých karet a CAD. Struktura APDU je definována v ISO/IEC 7816-4 [10]. APDU pracuje v hexadecimální soustavě. A dělí se na dvě kategorie. První je příkaz, druhý je odpověď - spolu tvoří neoddělitelný pár. Po odeslání APDU příkazu by měla vždy zpátky přijít APDU odpověď.

Příkaz APDU						
Hlavička (povinné)				*Tělo (volitelné)*		
CLA	**INS**	**P1**	**P2**	**Lc**	**Data**	**Le**

Tab. 1.1: Plný APDU příkaz

V tabulce 1.1 je ukázaný případ APDU příkazu, který obsahuje všechny povinné i nepovinné atributy. Obecně se příkaz skládá z hlavičky a z těla. Hlavička je povinná a

vždy musí být vyplněna. Tělo příkazu není povinné, nemusí být vyplněno. Hlavička APDU příkazu se skládá z CLA (Instruction Class) - třída instrukcí určující typ příkazu APDU, INS (Instruction Code) - kód instrukcí určující specifický příkaz, který se má provést a P1 a P2 - parametry příkazu APDU. Každý z prvků hlavičky má velikost 1 bajt. Tělo APDU příkazu se skládá z Lc (Length of Command Data) - délka příkazových dat v bajtech, Data - příkazová data a Le (Length of Expected Response) - délka očekávané odpovědi v bajtech. Velikost těchto prvků je dána tím, jaký komunikační protokol je použit. Na výběr je buď protokol T=0 nebo T=1. Při použití protokolu T=0 je velikost Lc rovna 0 až 1 bajt, velikost dat v rozmezí 0-255 bajtů a velikost Le 0 až 1 bajt. Při použití T=1 lze posílat mnohem větší objem dat. Lc může tedy obsahovat až 3 bajty, data až 65 536 bajtů a Le až 3 bajty [10].

Odpověď APDU		
Tělo(volitelné)	*Status(povinné)*	
Data	**SW1**	**SW2**

Tab. 1.2: Příklad APDU odpovědi

Odpověď APDU [10], viz tabulka 1.2, je ve srovnání s příkazem APDU velmi jednoduchá. V povinné části je pár SW1 a SW2 (Status Word) - návratový kód, který představuje status provedení příkazu. Každý ze SW má velikost 1 bajt. Ve volitelné části se nacházejí data, která applet vrací. Maximální velikost těchto dat určuje Le. Existují jen dvě možnosti, které mohou nastat při APDU odpovědi. Jako odpověď na příkaz se odpoví daty a SW1+SW2 nebo jen SW1+SW2. Návratový kód vrací svoji hodnotu podle toho, je-li proces dle APDU příkazu dokončen či ne. Při dokončeném procesu a normálním zpracování vrací kód 61XX nebo 9000. Kód 9000 znamená, že vše proběhlo úspěšně. Dokončený proces s varovným zpracováním reprezentuje kód 62XX nebo 63XX, kde informuje o změně nevolatilní nebo-li napěťově nezávislé paměti. U nedokončeného procesu s procesní chybou vrací kód 64XX nebo 65XX. A poslední možností je nedokončený proces s kontrolní chybou. Zde vrací kód 67XX až 6FXX.

1.2.3 Komunikace EMV

EMV neboli Europay MasterCard and Visa je celosvětový platební standard. EMV karty jsou vlastně Java Card čipové karty obsahující mikroprocesory schopné kryptografických procesů. Komunikační protokol pro Visa a MasterCard je rozdílný. Zde se popisuje protokol pro platební karty MasterCard.

Inicializace transakce začíná v momentě, kdy terminál pošle příkaz pro výběr PPSE (Proximity Payment System Environment). Zde terminál posílá kartě infor-

mace, kterou službu chce využít. Karta odpoví souborem služeb, které podporuje. Terminál si z těchto služeb vybere tu, kterou chce použít a pošle příkaz pro výběr AID appletu. Terminál pro autentizace transakce potřebuje různá data, proto karta odpoví data FCI (File Controll Information). FCI obsahuje například potvrzené AID, PDOL (Processing Data Options List), což jsou data parametrů potřebná pro průběh transakce, která musí terminál poslat kartě. Terminál dále pošle příkaz pro vrácení možností procesů s daty PDOL. Karta odpoví daty AIP (Application Interchange Profile) a AFL (Application File Locator). AIP jsou data, která říkají, jaké módy procesů karta podporuje. AFL data ukazují, kde jsou uložená statická data karty. Poté se čtyřikrát opakuje cyklus vyžádání dat a předání dat. Tímto končí proces inicializace [11]. Další fází je rozhodnutí terminálu, zda je nutné provést online transakci dle hodnocení bezpečnosti. Dle toho pošle výzvu na kartu. Karta také provede hodnocení bezpečnosti a rozhodne pro možnost online nebo offline transakce a pošle příslušný kryptogram. V případě offline verifikace terminál zkontroluje příchozí kryptogram. Je-li v pořádku, je transakce úspěšně dokončena, v opačném případě je transakce zamítnuta. V případě online verifikace pošle terminál všechna transakční data a kryptogram bance a banka tato data zkontroluje a rozhodne o přijetí nebo zamítnutí transakce [11].

1.3 Mifare karty

Velmi rozšířené jsou karty Mifare od firmy NXP. Postavení těchto čipů na trhu je velmi významné. To dokazuje přes 12 miliard prodaných kusů čipů. Porovná-li se toto číslo s článkem [1], tak se jedná o dvacetiprocentní až třicetiprocentní část z celkového objemu chytrých karet na světě. Dále s nimi pracuje více než 1000 partnerů a ohromné množství měst jejich technologii integruje do své infrastruktury. Mifare primárně pracuje s chytrými kartami, ale jejich čipy lze integrovat i do mobilních telefonů, náramků, chytrých hodinek nebo do obyčejných tagů [12].

Karty Mifare lze rozdělit na několik typů dle způsobu komunikace, paměti a bezpečnostních protokolů, které používají. Všechny typy ale spojuje jejich pracovní frekvence 13,56 MHz s krátkým dosahem a provoz v hexadecimální soustavě. V této kapitole se dále popisují nejvíce rozšířené typy Mifare karet.

1.3.1 Mifare Classic

Nejstarším zastupitelem Mifare karet je Mifare Classic, který je často popisován jako průkopník v chytrých bezkontaktních kartách operující na frekvenci 13,56 Mhz. První karty tohoto druhu se začaly vyrábět od roku 1994 [13], ale i přes svůj věk jsou dodnes velmi často používané kvůli jejich jednoduchému a levnému provozu.

Lze se s nimi setkat například ve školství u identifikačních karet studentů, v hromadné dopravě nebo v přístupových systémech. V Česku to jsou například některé studentské ISIC karty a nebo Hradecká či Pardubická karta. Dále se bude popisovat jen technologie nejnovějšího typu, tedy Mifare Classic EV1.

Paměť			
Typ	***EEPROM***	***Sektory***	***Bloky***
Mifare 1K	1024 bajtů	16	4
Mifare 4K	4096 bajtů	32 + 8	4 + 16

Tab. 1.3: Paměť Mifare Classic

Paměť EEPROM má dle typu Mifare Classic EV1 paměť buď 1024 bajtů nebo 4096 bajtů. Celek paměti je rozdělen na sektory a tyto sektory jsou dále rozděleny na bloky. Každý blok má velikost 16 bajtů. Kartu je možné přepisovat zhruba dvě stě tisíckrát se spolehlivou funkčností [14]. V tabulce 1.3 výše je popsána velikost paměti a počet sektorů a bloků pro jednotlivé typy Mifare Classic karet.

Sektory mají obecně za účel skladovat data sloužící k různým potřebám použití. Všechny bloky v takovémto klasickém sektoru kromě posledního se dají přepisovat dle potřeby ve smyslu ukládání dat. Poslední blok vždy slouží pro uložení zabezpečovacích dat. Důležitým sektorem je ale první sektor nebo-li nultý. Tento sektor je jediný, který se za normálních podmínek nemůže přepisovat. Je to z toho důvodu, že obsahuje náhodné UID (Unique Identifier) karty a tovární data, viz tabulka 1.4.

Sektor 0																
Bajt	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Blok 0	**UID**				**Tovární data**											
Blok 1	**Data**															
Blok 2	**Data**															
Blok 3	**Klíč A**						**Přístupové bity**				**Klíč B**					

Tab. 1.4: Sektor 0

Zabezpečení Mifare Classic

Bezpečnost karet Mifare Classic je postavená na proudové šifře CRYPTO1. Autentizace je prováděna pro každý sektor zvlášť pomocí jednoho až dvou klíčů. Šifrování a dešifrování uložených dat v blocích se řídí dle přístupových bitů v posledním bloku každého sektoru. Dle nastavení přístupových bitů lze určit, který z dvojice klíčů má jaká práva. Klíče mohou mít právo číst, psát, dekrementovat, inkrementovat,

obnovení a nebo převodu. Klíče i přístupové bity se dají libovolně měnit, jsou-li k dispozici správné klíče [14]. Kvůli prolomení šifry CRYPTO1 již tyto karty nesplňují dostatečné zabezpečení pro novodobé systémy [15].

1.3.2 Mifare DESFire

Nástupcem Mifare Classic jsou karty Mifare DESFire. První verze EV1 vyšla v roce 2002 [13]. V dnešní době je dle použití asi druhá nejrozšířenější karta Mifare. Technologie Mifare DESFire se používá ve stejných oblastech jako Mifare Classic jen s tím rozdílem, že je mnohem lépe zabezpečená. V Česku se tato technologie používá například ve slevové IN kartě od firmy České dráhy nebo v Pražské Lítačce. Dále se bude práce více zabývat jen nejnovější verzí Mifare DESFire EV3.

Narozdíl od Mifare Classic je Mifare DESFire multiaplikační karta, což znamená, že organizace paměti funguje na stejném principu jako Java karty. Velikost paměti je dle typu karty v rozmezí 2 kB až 8 kB s možností přepisování zhruba milionkrát se spolehlivou funkčností. Komunikace je definována dle ISO/IEC 7816 stejně jako u Java karet, tudíž se zde také posílají APDU zprávy mezi kartou a terminálem [16].

Zabezpečení Mifare DESFire

Mifare DESFire karty zabezpečují data pomocí 3DES a AES128. K přístupu k datům je zapotřebí správný klíč nebo více klíčů, které nesou přístupové právo. V nejnovější verzi EV3 je přidána zabezpečovací funkce časovače transakcí. Tento časovač má zabránit útoku mužem uprostřed, kdy útočník zpozdí transakci tím, že nechá kartu v provozu po tom, co jí odebere od legitimního terminálu. Dále je zde přítomná kontrola vzdálenosti, která dokáže měřit, je-li karta v jisté vzdálenosti od terminálu. Tato funkce slouží k zabezpečení proti útoku přeposláním [16]. Mifare DESFire EV3 se považuje za jednu z nejbezpečnějších technologií od Mifare a je doporučovaná k použití v místech, kde je vyžadována vysoká bezpečnost. V tabulce 1.5 jsou znázorněny některé bezpečnostní funkce pro různé verze Mifare DESFire.

Typ	**DESFire EV1**	**DESFire EV2**	**DESFire EV3**
Zabezpečení	**3DES, AES128**	**3DES, AES128**	**3DES, AES128**
Max. počet aplikací	**28**	**bez limitu**	**bez limitu**
Více klíčů k přístupovému právu	**ne**	**ano**	**ano**
Časovač transakce	**ne**	**ne**	**ano**
Kontrola vzdálenosti	**ne**	**ano**	**ano**

Tab. 1.5: Mifare DESFire generace

1.3.3 Další karty Mifare

Hlavními produkty z řady Mifare jsou Classic a DESFire, ale za zmínku stojí také Mifare Ultralight a nebo Mifare Plus. Použití těchto dvou technologií je poměrně nižší, ale přesto mají svoje místo na trhu s chytrými kartami.

Mifare Ultralight

Výhodou Mifare Ultralight je jeho jednoduchost, velmi nízká cena a tím možnost použití ve velkém rozsahu. Dále bude popsána nejnovější verze EV1. Paměťová stavba je podobná jako u Mifare Classic. Paměť EEPROM je rozdělená na stránky. Počet stránek je dán verzí karty, v 640bitové verzi je zde 20 stránek a v 1312bitové verzi je přítomno 41 stránek. Každá stránka má velikost 4 bajty, ve kterých jsou uložená data. Pro 640bitovou verzi je dostupno jen 384 bitů pro čtení a psaní do karty, zbytek paměti tvoří tovární data, mechanizmus pro nastavení read-only dat a jednorázově programovatelná část paměti. Pro 1312bitovou verzi je pro užití dostupno 1024 bitů. Zabezpečení přístupu k datům je provedeno pomocí 32bitového hesla s možností nastavení limitu pro nepovedené pokusy o autorizaci [17]. Nejčastěji se s Mifare Ultralight lze setkat v jednorázových potřebách. Příkladem jsou lístky na nějaké události nebo předem nabitá jízdenka, které se pak po použití zahodí.

Mifare Plus EV2

Další nástupce Mifare Classic je Mifare Plus. Její nejnovější verze funkčně připomíná Mifare DESFire, tedy komunikace je definována dle ISO/IEC 7816. Ale paměťová konfigurace je podobná Mifare Classic, paměť je rozdělená na sektory a bloky. Zabezpečení dat je provedeno pomocí AES128. Je zde přítomen i starý algoritmus CRYPTO1, díky kterému je možná zpětná kompatibilita s Mifare Classic. Zabezpečuje se každý sektor samostatně a nebo celá karta jako celek. Stejně jako Mifare DESFire má i Mifare Plus EV2 kontrolu vzdálenosti a časovače transakcí. Mifare Plus není funkčně tak flexibilní jako Mifare DESFire, ale dle firmy NXP jsou obě technologie na podobné bezpečnostní úrovni [18].

2 Android platforma a technologie NFC

Čipové karty a platforma Android OS na mobilních telefonech jsou velmi kompatibilní z hlediska komunikace. Systém Android je gigantem ve svém průmyslu a má obrovské zastoupení u mobilních telefonů, tudíž je zde velký potenciál užití. Dále je tato platforma obecně známá pro svoji otevřenost vůči vývojářům. Z těchto důvodů je dobrou volbou pro tuto práci. V této kapitole jsou popsány nejdůležitější funkce a služby, které jsou potřeba pro testování bezpečnosti kartových systémů.

První použití technologie NFC zaznamenává Android v roce 2010 od verze Gingerbread - Android 2.3 na mobilním telefonu Nexus S [19]. I přes tehdejší nepraktické využití předpovídali vývojáři velký posun v budoucnu. Jejich správnou předpověď dokazuje například platební systém Google Pay. U Android OS není nutno zůstávat jen u mobilních telefonů. U chytrých náramků Xiaomi Mi Band již standardně fungují autentizace a platby pomocí NFC funkce, které mají kořeny v Číně. Naštěstí se i tato technologie velmi rychle rozšiřuje dále do celého světa [20]. Tudíž lze čekat, že použití Android OS ve spolupráci s technologií NFC bude nadále stoupat.

Zkratka NFC označuje bezdrátový způsob komunikace na krátkou vzdálenost. Na rozdíl od svého staršího předchůdce RFID, kde je funkční vzdálenost počítána v řádech metrů, je u NFC vyžadovaná vzdálenost pro dosažení spojení maximálně okolo 10 centimetrů [4]. Hlavní účel NFC spočívá v přenášení poměrně malých množství dat mezi zařízeními podporujícími tuto technologii. Technické specifikace jsou popsány v ISO/IEC 18092 [21].

2.1 Typy zařízení

NFC zařízení se dělí na dva typy. Prvním typem je zařízení aktivní, jehož hlavní charakteristikou je vytváření radiofrekvenčního pole. Příkladem jsou mobilní telefony, autentizační terminály atd. Druhým typem je naopak zařízení pasivní, které nedokáže vytvářet žádné pole a je většinou napájeno polem aktivního zařízení. Taková zařízení jsou obvykle menší objekty, které se bez problémů vejdou do kapsy. Tato pasivní zařízení se zde představují jako tzv. tagy (česky štítek).

Tag je základní proměnná při vývoji aplikací využívajících NFC. Lze je kategorizovat dle složitosti funkcí, které přináší [22]. První kategorií jsou jednoduché tagy vykonávající jen ty nejjednodušší aktivity. Umožňují pouze čtení a zápis dat. Mohou být jednorázové nebo vícekrát použitelné. Data v nich je možné uzamknout jen pro čtení. V reálném světě tyto jednoduché tagy zastupují například NFC nálepky nebo jednoduché čipové tagy - viz obrázek 2.1.

Komplexnější tagy tvoří druhou skupinu. Na rozdíl od jednoduchých tagů disponují dalšími funkcemi, jako jsou různé matematické operace a nebo dokonce kryp-

Obr. 2.1: Čipový tag.

tografický hardware pro operace nad nimi. Tyto funkce lze použít pro ověřování přístupu, počítání kryptografických klíčů a mnoho dalších. Příkladem skutečného použití jsou vyšší verze Mifare karet.

Poslední kategorií jsou nejvíce propracované tagy využívající složitých funkcí. Obsahují v sobě operační prostředí pro komplexní interakci s kódem, který běží na samotném tagu [22]. Aplikace těchto tagů se využívá u služeb, kde je potřeba dosáhnout co nejvyšší možné bezpečnosti komunikace mezi zařízeními. Takové charakteristiky tagu hojně využívají banky u jejich platebních systémů, konkrétně platformu Java Card na platebních kartách.

2.2 Komunikační model

Zařízení NFC spolu komunikují v rámci prostředí Android OS stejně tak, jak je popsáno u komunikace Java Card. Na jedné straně stojí CAD, zde tedy aktivní zařízení, a na druhé straně pasivní zařízení, které se zde představuje jako tag. Inicializovat komunikaci může jen zařízení aktivní a pro komunikaci se používá radiofrekvenční signál 13,56 MHz [4]. Dalším předpokladem komunikace je mít aktivní NFC službu. V momentě, kdy je služba zapnutá, zařízení nepřetržitě hledá ve svém poli tagy, ke kterým by se mohlo připojit a začít komunikaci. Různé tagy podporují různé technologie, proto je nutno filtrovat jen takové tagy, se kterými aplikace chce komunikovat. Další důvod pro tento krok je situace, kdy jsou nainstalovány dvě aplikace, které mají odbavit různé karty, aby se navzájem nekřížily a nehledaly stejné tagy, i když pracují s odlišnými. Filtraci standardně zařizuje The tag dispatch systém [23]. Používá se ale také starší způsob, který využívá funkci ReaderCallback ze třídy NfcAdapter [24].

Android rozeznává tagy třídy IsoDep, MifareClassic, MifareUltralight, Ndef, NdefFormatable, NfcA, NfcB, NfcBarcode, NfcF a nebo NfcV. Důležitými třídami jsou pro tuto práci IsoDep a MifareClassic. Třída IsoDep zajišťuje komunikaci v rámci ISO/IEC 7816-4. Tato třída je tudíž velmi vhodná pro práci s JavaCard nebo s MifareDESFire a dalšími technologiemi používajícími APDU zprávy. Třída MifareClassic zajišťuje funkce pro komunikaci s Mifare Classic kartami.

Komunikaci s NFC tagem zajišťují:

1. **The tag dispatch systém** - Při nalezení NFC tagu systém zjistí příchozí intent (česky záměr). Intent je v podstatě objekt, ve kterém jsou dány parametry, které se dále předávají do aplikací. Hlavní činnost pro filtraci uskutečňuje intent-filter. V tomto objektu se definuje, jaké technologie budou aplikací přijímány. Nastavení filtrace se uskutečňuje v souboru `AndroidManifest.xml`. Tag dispatch systém má takovou vlastnost, že není nutné, aby byla aplikace otevřená, protože se automaticky otevře při nalezení hledaného tagu.
2. **ReaderCallback** - Na rozdíl od The tag dispatch systému je nutné mít spuštěnou a aktivní aplikaci pro funkčnost tohoto filtru. Což může být výhoda například při nechtěném přiložení tagu do aktivního pole NFC, kdy by se v minulém případě otevřela aplikace bez svolení uživatele. Tímto způsobem je tedy zařízená větší kontrola nad aplikací. Definování hledaných tagů je provedeno při inicializaci módu pro čtení ve funkci enableReaderMode.

2.3 Provozní módy NFC

Moderní zařízení s NFC se rozdělují na tři hlavní módy dle chování a funkce [22].

1. **Čtení/psaní** - Nejčastěji používaným módem je mód čtení/psaní. Snaha je navázat komunikaci s tagem a dostat z něho určitá data pro daný účel. Popřípadě lze na kartu určitá data zapisovat, přepisovat nebo mazat.
2. **P2P** - Přes P2P mód je možné zajistit přeposílání dat mezi dvěma zařízeními, která mají aktivní tento mód. Službu používá například Android Beam.
3. **Emulace karet** - Poslední módem je emulace karty nebo-li tagu samotného. Umožňuje NFC zařízení chovat se jako originální karta nebo tak, jak byla aplikace na emulaci vytvořena. S takovou emulovanou kartu lze pracovat úplně stejně jako s klasickým tagem. Je možno s ní komunikovat s jakýmkoli aktivním zařízením, jako je například mobilní telefon s NFC, externí NFC čtečka nebo klasický obchodní terminál. Tuto službu zařizuje funkce HCE, která bude v následující sekci popsána.

2.4 Služba HCE

Zkratka HCE označuje technologii s názvem Host-based card emulation [25]. Technicky je HCE podmnožinou NFC, kde se definuje jako služba. HCE umožňuje zařízení využívat mód emulace karet. Zařízení schopná NFC jsou standardně schopná i HCE. HCE byl vyvinut pro účely placení přes aplikaci Google Wallet. HCE podporuje standard ISO/IEC 7816-4 [10] a standard ISO/IEC 14443-2 [6]. Tudíž lze komunikovat zprávami APDU v bezdrátové komunikaci. Pro funkci emulace je vitální služba `HostApduService`. Zde je deklarována funkce `processCommandApdu()`, která zpracovává příchozí APDU zprávy. Komunikace mezi emulujícím zařízením a terminálem je úplně stejná jako komunikace čipové karty, která také používá APDU zprávy. To jest musí zde být prvně aplikace s určitým AID. AID se zde nastavuje prvně staticky v souboru pro službu HCE, kde se definují metadata. AID se nastavuje v položce aid-filter. V tomto souboru lze nastavit i více AID zároveň, na které by aplikace odpovídala. AID lze dynamicky měnit při běhu aplikace. Po výběru AID terminálem nastává část výměn zpráv. Přijde-li APDU zpráva, spustí se funkce `processCommandApdu()` a podle implementace vybere, jakou odpověď pošle terminálu.

Samotné Android zařízení emuluje svoje vlastní UID jako klasická čipová karta, ale s rozdílem, že se po každé inicializaci s terminálem změní. UID ale lze v root zařízení nastavit staticky v knihovnách libnfc-*, podporuje-li to NFC čip v zařízení [26].

3 Bezpečnostní hrozby kartových systémů

Kapitola se v první části zabývá rozdělením typů útoků na čipové karty. Dále se zde definují útoky na kartové systémy v obecném směru, například jaký je jejich proces, co je k provedení zapotřebí a jejich silné a slabé stránky. Nakonec jsou zde popsané vybrané kartové systémy, jak fungují a jaké konkrétní útoky na ně lze či nelze provést.

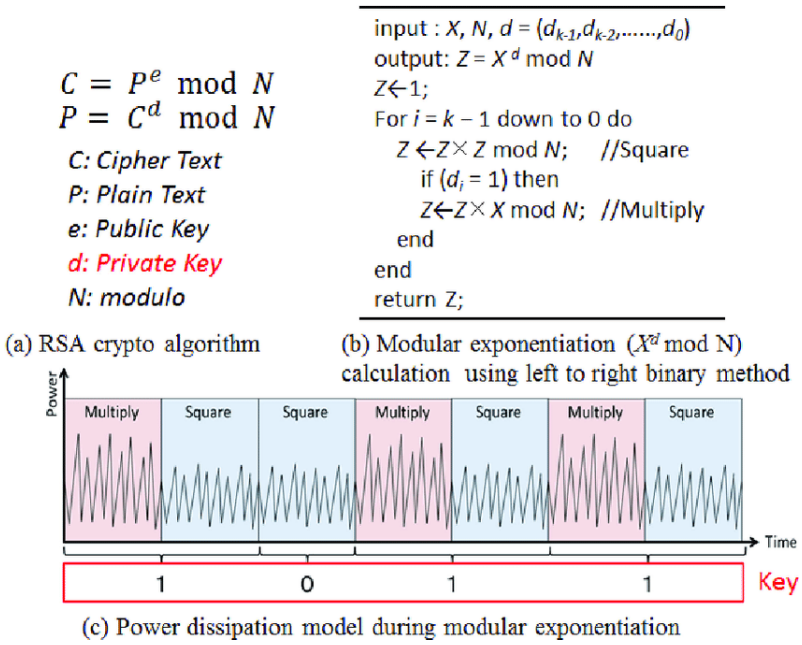
3.1 Útoky na čipové karty

Technologie čipových karet existuje již poměrně dlouho. Tudíž se tématem bezpečnosti a možnosti útoků na tento systém zabývalo již mnoho prací a výzkumů, jako například [9][27][28][29], kde autoři zkoumají možné útoky z mnoha směrů. Obecně lze ale říct, že existuje rozdělení do tří kategorií útoků.

3.1.1 Fyzické útoky

První kategorií útoků jsou fyzické útoky. Tento druh útoků používá specifické přístroje k fyzické manipulaci karty a sleduje důsledky po manipulaci nebo sleduje fyzikální jevy při běžné činnosti bez manipulace. Způsoby zkoumání lze rozdělit na invazivní a neinvazivní. K invazivním, tedy ničícím originální stavby karty, patří například použití chemických roztoků k pomalému rozložení vrstev nebo částí karty, kde může být cílem extrahovat čip karty, který se zkoumá se snahou zjistit citlivá data v pamětích. Dále se používají lasery pro manipulaci fyzické struktury karty, například ničení obvodu za cílem změnit funkcionalitu karty a získat data v pamětích. Stejné cíle má také měnění proudu elektřiny běžící v obvodu elektromagnetickými zařízeními [27]. Na druhou stranu neinvazivní nebo také útoky postranním kanálem neničí kartu, ale pouze sledují její fyzickou implementaci. Tedy postranní kanál je jakýkoliv proud informací o fyzickém chodu zařízení. Nejčastější typy postranních kanálů jsou kanály spotřeby energie, časové kanály, elektromagnetické kanály, chybové kanály a další. Na obrázku 3.1 na další straně lze vidět příkladový útok postranním kanálem spotřeby energie kryptosystému RSA [30].

Nevýhodou takových útoků je nutnost použití velmi specifických zařízení, jako jsou například zmíněné lasery, tudíž útok může vyžadovat nemalé finanční obnosy. Časově jsou fyzické útoky poměrně náročné, proces může trvat i několik dní.



Obr. 3.1: Získání klíče pomocí proudové analýzy.[30]

3.1.2 Logické útoky

Druhým typem útoků jsou útoky logické [28]. Nepoužívají se zde fyzické předměty ale specificky navržený kód, který využívá slabiny samotné architektury technologie chytrých karet. Slabin může existovat mnoho, příkladem mohou být limitované zdroje karty, nesprávná verifikace appletů a jejich instalace na kartu. Chyby nastávají také při lidské činnosti na kartě. Vývojář karty může vytvořit spoustu chyb při tvorbě softwaru. Jednou z nejzávažnějších chyb je špatně vytvořená bezpečnost dat na kartě. Tím se myslí nešifrovaná data kryptografických klíčů a citlivých dat nebo zapomenutí odebrání testovacích funkcí s kryptografickými klíči, které útočník pak může lehce získat. Praktické útoky lze najít v [31], kde autoři provedli záměnu typů a využili bug v implementaci transakčního mechanizmu pro přečtení celé paměti karty.

Na rozdíl od fyzických útoků není k logickým útokům potřeba žádná speciální technika, z tohoto důvodu jsou náklady na provedení mnohem nižší. Další výhodou je velmi krátký časový průběh útoku vůči fyzickým útokům. Doba trvání útoků může být i jen několik sekund.

3.1.3 Kombinované útoky

Posledním typem jsou kombinované útoky. Jedná se vlastně o kombinaci fyzických a logických útoků. Díky tomu se vyrovnávají nevýhody výhodami. Cena i čas se díky

logickým útokům sníží a fyzické přináší zase další rozměr pro širší možnosti operací nad kartou. Tudíž lze říci, že ve většině případů je kombinovaný útok nejúčinnější.

3.2 Útoky na kartové systémy obecně

Díky technologii NFC a HCE je možno sestavit útok, který je lehce uskutečnitelný bez potřeby speciálních zařízení a bez potřeby dlouhého kontaktu s kartou oběti. Jedná se o útoky opakováním (Replay Attack) a přeposíláním zpráv (Relay Attack). Důležité jsou hlavně u bezdrátové komunikace.

3.2.1 Relay Attack

Relay Attack (česky útok přeposláním) je založený na přeposílání zpráv mezi dvěma legitimními stranami přes nějaký komunikační kanál. Legitimní strany u kartových systémů představují čipová karta a terminál. Tato dvě zařízení si mezi sebou přeposílají data, kterými se autentizují. Útok nastává v momentě, kdy útočník přidá mezi dvě legitimní zařízení svůj komunikační kanál, který přeposílá data na větší vzdálenost. Z jedné strany kanálu se posílají data z terminálu na kartu a z druhé strany data z karty na terminál. Jako takový komunikační kanál se může zvolit standard Bluetooth, mobilní síť nebo Wi-Fi.

Komunikace začíná tím, že legitimní terminál pošle na zařízení s HCE APDU příkaz, ten jej přepošle zařízení s NFC, to dále přepošle příkaz na legitimní kartu. V tuto chvíli si karta myslí, že komunikuje s legitimním terminálem, protože dostává data, která očekává. Jako reakci na příchozí APDU příkaz odešle karta APDU odpověď, kterou pošle na zařízení s NFC, to dále pošle zařízení s HCE a to nakonec přepošle data legitimnímu terminálu. Tím, že terminál dostal jako odpověď originální, nijak nezměněná data od legitimní karty, tak také věří tomu, že komunikuje s legitimní kartou. Přeposílání dat probíhá tak dlouho, než se předají všechny potřebné zprávy nebo než se ztratí spojení mezi nějakými ze zúčastněných zařízení [32]. Na obrázku 3.2 na následující straně je zobrazeno schéma útoku přeposláním v případě útoku na debetní kartu.

Velkou výhodou tohoto útoku je nepotřebnost znát data, která proudí komunikačním kanálem. Z toho důvodu je možné úplně obejít jakákoliv kryptografická zabezpečení, která utajují obsah posílaných dat.[33]

Limity a nevýhody

Zařízení musí být v blízkosti oběti do 5 cm. Při nedostupnosti karty musí útočník použít sociální inženýrství, aby se ke kartě dostal. Je nutné zajistit co nejrychlejší spojení pro předávání zpráv [32]. Kvůli omezenosti funkce HCE lze útok provést

Obr. 3.2: Schéma útoku přeposláním

na systémech podporujících ISO/IEC 7816-4 a ISO/IEC 14443. Dalším případným problémem je nezbytná znalost AID. Nevýhodou může také být použití přídavného zabezpečení, jako je například PIN kód, který musí útočník znát k úspěšnému dokončení útoku.

3.2.2 Klonování karet

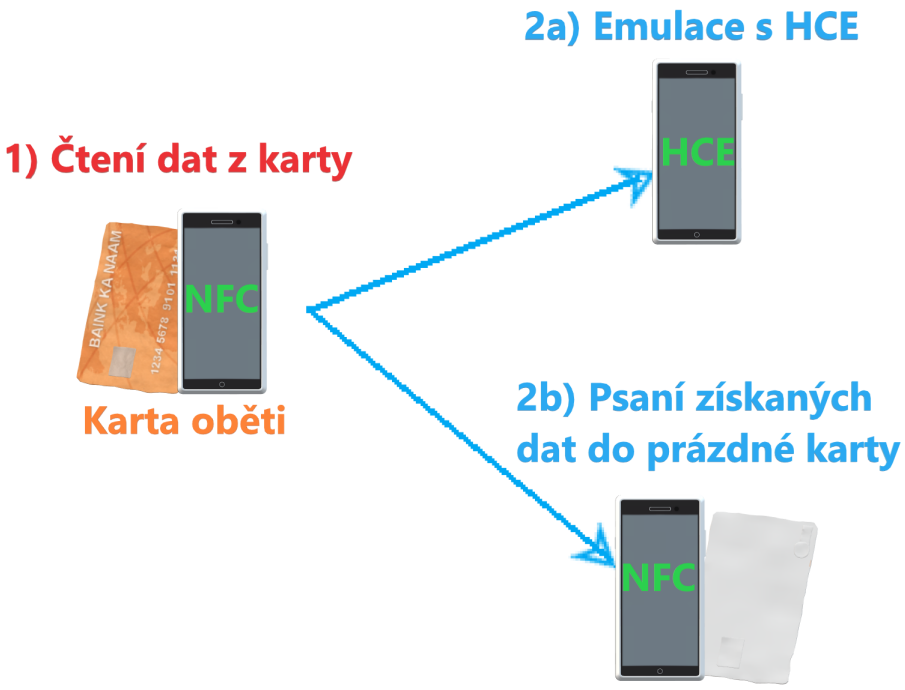
K tomuto útoku se použije jedno až dvě zařízení. První zařízení bude zastupovat funkci čtení dat z originální karty a tím bude zařízení s NFC. Druhé zařízení bude fungovat jako úložiště dat, která byla přečtena. Musí mít stejnou strukturu jako originální karta nebo se dokázat chovat jako ona. Tudíž tímto zařízením může být prázdná karta, která podporuje stejnou technologii nebo zařízení HCE se stejnou podmínkou. Schéma tohoto útoku, včetně dvou možných výsledků naklonovaných karet, je popsáno v obrázku 3.3 na následující straně.

Limity a nevýhody

V šifrovaném systému je nutné znát dešifrovací klíče. Při nedostatku znalostí o struktuře a systému není možné kartu klonovat ani napodobit její chování. A stejně jako u útoku přeposláním sdílí klonování problém s dodatečnou ochranou. Dalšími sdílenými problémy jsou dostupnost karty, vzdálenost pro provedení útoku a čas.

3.2.3 Modifikace dat

Pro tento útok je zapotřebí jen jedno zařízení, které umí číst a psát data do čipových karet. Použije se opět zařízení s NFC, které je ve čtecím/psacím módu. Postup



Obr. 3.3: Schéma Klonování karet

je následující. Prvním krokem je navázání kontaktu s cílenou kartou. V druhém kroku se v situaci prospěchu najdou a přečtou specifická data a poté modifikují. V situaci, kdy chce útočník uškodit držiteli karty, modifikuje různá data, aby poškodil strukturu karty a karta dále nemohla být použita. Nyní se již jedná o korupci dat a odepření služby neboli útok DOS (Denial Of Service).

Limity a nevýhody

Znalost nutná k provedení úspěšné modifikace dat s pozitivním důsledkem, tzn. prospěch držitele karty, je stejná nebo i vyšší než u klonování samotné karty. Při špatně změněné hodnotě může celý systém zkolabovat a karta se může stát nečitelnou a tudíž nepoužitelnou. Místo kladného důsledku je tedy důsledek negativní. Kryptograficky šifrovaná data nebo vstup do dat je zde samozřejmě jako u dvou předchozích útoků také problém.

3.3 Útoky na kartové systémy prakticky

Tato kapitola se zabývá technologiemi dnešních běžně využívaných služeb založených na čipových kartách komunikujících na frekvenci 13,56 MHz. Jsou zde zkoumány zabezpečovací prostředky a praktické útoky.

3.3.1 Platební systémy - EMV

Bezpečnost platebních systémů EMV je dáno:

1. Kryptografické mikroprocesory - Chrání data před čtením a zneužitím útočníkem.
2. Vlastní kryptografické protokoly - Data při autentizaci nejsou při každé transakci stejná a tedy chrání před útokem přehráním.
3. Fyzické ochrany úložiště - Nacházejí se zde kryptografické klíče. Tyto klíče jsou pro každou kartu jiné, takže i kdyby byl schopen útočník získat klíče na jedné kartě, nejsou dále použitelné na zbytek karet [34].

Platební karty jsou odolné vůči klonování karty a protokol EMV nelze snadno přehrát díky náhodně generovaným číslům při transakci. Z toho plyne, že nejjednodušším útokem na tento systém je přeposílání APDU příkazů a odpovědí. V pracích [33][32][35][11] se všem podařilo vytvořit útočící systém na základě přeposílání dat. Tyto útoky fungovaly na tehdejší implementaci platebních systémů. Dalším možným útokem je klonování a upravování dat předchozích transakcí. Těmito útoky se zabývaly práce [36][34]. V originální práci [36] našel Michael Roland způsob, jak využít starší mód MasterCard karet, který byl založen na magnetickém proužku. Využilo se zde degradování způsobu komunikace karty z EMV na mód pouze pro magnetický proužek a nevhodně vygenerované číslo pro bezpečnostní prvek s velikostí jen 2-3 míst. Šlo tedy vypočítat všechny možnosti bezpečnostního prvku.

3.3.2 Přístupové systémy

Podobně jako u platebních systémů se u kartových přístupových systémů může používat standard APDU příkaz a odpověď. Je zde tedy stejná možnost použití útoku přeposláním. Obecně jsou novější karty používající normu ISO/IEC 7816 považovány za bezpečnější kvůli horší dostupnosti dat na kartě, která nelze lehce přečíst [26].U starších typů karet, jako například Mifare Classic 1k/4k, lze číst data poměrně velmi lehce. Navíc mnoho institutů a firem nepoužívá ani základní kryptografické klíče pro šifrování dat na kartě nebo jsou tyto klíče v továrním nastavení [37]. Výchozí klíče jsou veřejně dostupné, tudíž není žádný problém je získat. Tím je jasně nejlepším výběrem útoku klonování karty. Dále je běžnou praktikou autentizovat přístup pouze UID, která by měla být pro každou kartu zcela unikátní. I kdyby útočník nedokázal získat jakákoli data na kartě nebo neznal její strukturu, stačí přepsat UID do jiné karty, která má schopnost přepisovat svoje UID nebo použít zařízení s HCE. Některé mobilní telefony s Android OS mají schopnost emulovat specifické UID, tudíž není nutné kopírovat všechna data na kartě ale jen bez obtíží dostupné UID.

3.3.3 Systém předplacených karet

Prostředky nabíjení peněžních aktiv na kartu jsou již zastaralé a v dnešní době jen výjimečně používané. Potíž je v tom, že data na předplacené kartě lze přepsat a útočník si tak může vygenerovat nekonečné množství prostředků. Příkladem může být článek z roku 2008 [38], kde bylo publikováno, že Oyster karta, která je dodnes používána jako hlavní systém placení pro dopravu v Londýně, byla zneužita. Naštěstí byly od té doby vyvinuty nové možnosti chránění takových typů dat, které se staly novým dnešním standardem a většinově nahradily staré nebezpečné karty. Možným útokem na takové techniky je jednoduchá modifikace dat. Modifikují se data, která představují stav prostředků na kartě [37]. Samozřejmě je-li možno číst a psát do paměti. Dalším proveditelným útokem je útok přehráním APDU příkazů. Rory Flynn ve své práci[39] provedl útok na kartu Mifare DESFire EV1. Využívá zde tzv. Value Files, tedy soubory s hodnotami, ve kterých jsou uloženy číselné hodnoty peněžních zůstatků. Při dobíjení karty se použije útok přeposláním pro přeposílání autentizačních příkazů a v momentě, kdy pošle terminál příkaz na nabití prostředků, tak zařízení posílající příkaz na kartu zopakuje n krát tento příkaz a tím se dobije n krát více než bylo zadáno.

3.4 Existující aplikace umožňující čtení/zápis/emulování či kopírování čipových karet

Jak již bylo popsáno v kapitole 2 „Android platforma a technologie NFC“, Android OS je téměř zcela kompatibilní s ISO/IEC 7816-4 a podporuje technologii Mifare Classic již od API 10 [40]. Lze tedy předpokládat, že jsou již vytvořené aplikace, které využívají tříd a funkcí dostupných na Android OS pro zacházení s chytrými kartami. V této kapitole budou dále popsány příklady takových aplikací.

3.4.1 NFC Smart Card Info

Na první pohled není většinou možné rozpoznat jakou technologií chytrá karta disponuje. Aplikace NFC Smart Card Info [41] dokáže zjistit technologie karty, její ID a další informace. V případě IsoDep technologie se zkouší, má-li karta nainstalované některé veřejně známé applety a v případě shody je vypíše. Na obrázku 3.4 (str. 36) je ukázán příkladový výpis čtené karty.

3.4.2 Credit Card Reader NFC (EMV)

Další aplikace, která dokáže číst data z chytré karty, se jmenuje Credit Card Reader NFC (EMV) [42]. Tato aplikace čte EMV karty a dokáže získat informace o číslu karty, expiraci karty, zbylých pokusech PIN, typu karty a appletech, které na kartě jsou. Ve starších EMV kartách bylo možné se dozvědět i jméno majitele karty a nebo minulé transakce kartou. Ukládání těchto dat je v některých nových kartách zrušeno kvůli soukromí. Tyto data dokáže aplikace získat tak, že posílá specifické APDU zprávy, které jsou veřejně dostupné a dostává odpovědi s informacemi o kartě. Na obrázku 3.5 (str. 36) je ukázán příkladový výpis čtené EMV karty.

Aplikace nezobrazuje ihned informace o čísle karty a její expiraci z důvodu zachování minimální úrovně bezpečnosti. Pro zobrazení těchto údajů je nutné zadat znalost posledních 4 číslic na kartě jako důkaz vlastnictví karty. Toto zabezpečení je nastaveno jen ze strany vývojáře aplikace. To tedy znamená, že jiné aplikace by mohly bez omezení citlivá data číst.

3.4.3 MIFARE Classic Tool

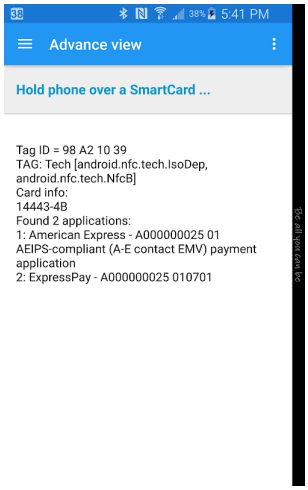
MIFARE Classic Tool [43] je velmi propracovaná aplikace umožňující různorodou práci s kartami Mifare Classic. V aplikaci lze kartu číst, psát do ní data, ukládat data z karty, klonovat kartu, přepisovat data a mnoho dalšího. Tyto funkce lze vidět na obrázku 3.6 (str. 36). Čtení dat na kartě a psaní dat na kartu je možné díky uloženým klíčům. V aplikaci jsou uloženy původní tovární klíče a prozrazené klíče, které lze najít v různých projektech zabývajících se bezpečností Mifare Classic a nebo na hackerských webových stránkách. Klíče je možné v případě potřeby přidat i odebrat. Klonování nebo spíše přepisování nultého sektoru je možné jen za pomocí tzv. GEN2 karty, která je vyrobená tak, aby mohla být celá přepisovatelná pomocí telefonního zařízení. Je-li karta používána jako předplacená karta, lze data vyjadřující reálné hodnoty změnit a například si tyto hodnoty navýšit. Aplikace jako celek je schopná úplně obejít bezpečnost systémů založených na kartách Mifare Classic za předpokladu, že jsou známé přístupové klíče a za předpokladu případného použití GEN2 karty .

3.4.4 NFC Card Emulator Pro (Root)

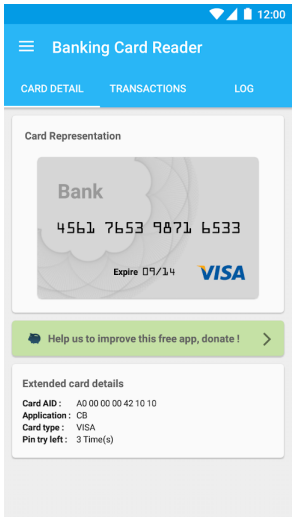
V aplikaci NFC Card Emulator Pro (Root) [44] se na rozdíl od předchozích aplikací používá nejen NFC pro čtení a zápis, ale také funkce HCE pro emulování. V nepřesném popisu aplikace je uvedena simulace různých typů karet, což není z větší části pravda. Pomocí HCE lze emulovat jen karty z třídy IsoDep. Aplikace ale dokáže měnit UID telefonu v režimu emulování na jakoukoliv hodnotu . Systémy, které fungují

jen na principu jedinečnosti UID karet, jsou touto aplikací bezpečnostně přemoženy. V obrázku 3.7 je ukázáno emulování UID jedné z uložených karet.

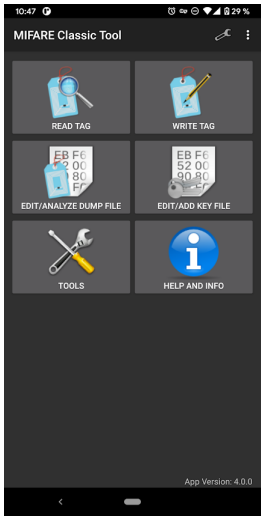
Mobilní zařízení většinou emuluje náhodné UID podle konfiguračního souboru libnfc-*. Tato aplikace tedy umí najít tento soubor a upravit v něm hodnoty tak, aby poté mobilní zařízení emulovalo chtěné UID. K přístupu do souboru libnfc-* je ale zapotřebí root telefonu. Ne všechny telefony ale mají konfigurační soubor napsaný tak, aby jej bylo možné upravit pro emulování specifického UID. Tyto soubory jsou různé dle NFC čipů. Příkladem nekompatibilního NFC čipu je Samsung S3NRN82 obsažený například v mobilním telefonu Samsung Galaxy S9+. U většiny mobilních zařízení s NXP nebo Broadcom NFC čipem bude aplikace funkční.



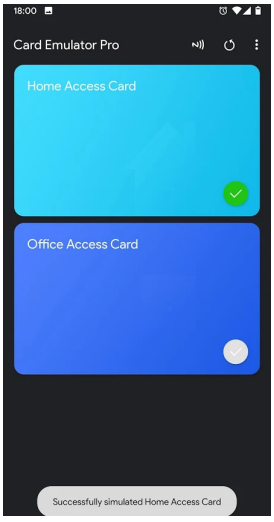
Obr. 3.4: NFC Smart Card Info



Obr. 3.5: Credit Card Reader NFC



Obr. 3.6: MIFARE Classic Tool



Obr. 3.7: NFC CardEmulator Pro

4 RR Demonstrátor - mobilní část

Cílem tohoto demonstrátoru je realizovat některé bezpečnostní slabiny a využít existujících způsobů útoků na čipové karty a tím poukázat na možnosti zneužití dnešních karetních systémů. Aplikace demonstrátoru se zabývá multiaplikačními kartami používajícími technologii **Java Card** nebo obecně kartami používajícími komunikační rozhraní definované v **ISO/IEC 7816-4** a kartami **Mifare Classic**. První oblast útoků je **klonování (replay)** a emulování přístupových karet a druhou oblastí je **útok přeposláním zpráv (relay)**.

U multiaplikačních karet technologií používajících komunikační rozhraní definované v ISO/IEC 7816-4 je obvykle použit nějaký kryptografický protokol, který zabraňuje přímému klonování karty. Klonování lze tedy provést pouze na jednoduchých bezpečnostních protokolech a k tomu je také zapotřebí znát některé informace o samotném protokolu. Z tohoto důvodu je u těchto karet brán hlavní zřetel na útok přeposláním.

Na Mifare Classic karty na rozdíl od multiaplikačních karet nelze použít útok přeposláním z důvodu, že Android funkce HCE zatím dokáže emulovat jen komunikaci využívající APDU zprávy. Naopak je lze poměrně lehce klonovat. Aplikace demonstrátoru má tedy velkou část věnovanou klonování Mifare Classic karet.

Aplikace je psaná v programu Android Studio v jazyce Kotlin s minimální verzí **API 22: Android 5.1 Lollipop**, která by měla spolehlivě fungovat na 92,3 % všech Android zařízení. V dalších kapitolách je popsána struktura aplikací, použité funkce a jejich scénář použití.

Technologie	***Replay (klonování karet)***	***Relay attack***
Multiaplikační karty (Java Card, Multos, Mifare DESFire, ...)	**ne**	**ano**
Mifare Classic	**ano**	**ne**
EMV	**ne**	**ano**

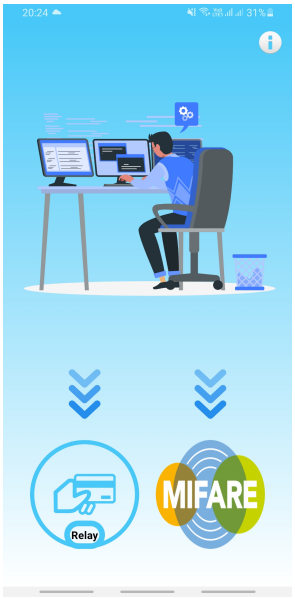
Tab. 4.1: Funkcionality RR demonstrátoru

Aplikace RR demonstrátor podporuje dva základní útoky (R)eplay a (R)elay, viz tabulka 4.1. Útok přehráním (replay) dokáže klonovat karty Mifare Classic na přepisovatelnou kartu podporující Miface Classic technologii. Útok přeposláním (relay) zpráv je funkční pro karty s komunikačním rozhraním definované v ISO/IEC 7816-4. V případě RR demonstrátoru byly testovány karty Java Card a EMV.

Hlavní aktivita aplikace je rozdělená na dvě části. První část s názvem „Relay“ provádí útok přeposláním zpráv na karty s komunikačním rozhraním dle ISO/IEC

7816-4. Druhá část se zabývá čtením a klonováním Mifare Classic karet. Tyto aktivity budou dále podrobně popsány. K plnému užití všech aktivit v aplikaci je nutné mít dvě mobilní zařízení se správným NFC hardwarem, internetové připojení, rootnuté zařízení a GEN2 kartu.

Uživatelská struktura aplikace je rozdělená na dvě hlavní aktivity. Každá hlavní aktivita obsahuje několik dalších aktivit, které poskytují podstatné funkce, nástroje a rozhraní. Na obrázku 4.1 lze vidět úvodní stránku včetně dvou hlavních aktivit a informační ikonku. Informační ikonky jsou přítomny ve všech aktivitách. Obsahují základní informace o aktivní aktivitě a pomáhají navést uživatele ke správnému postupu užití.



Obr. 4.1: Úvodní obrazovka RR aplikace pro Android

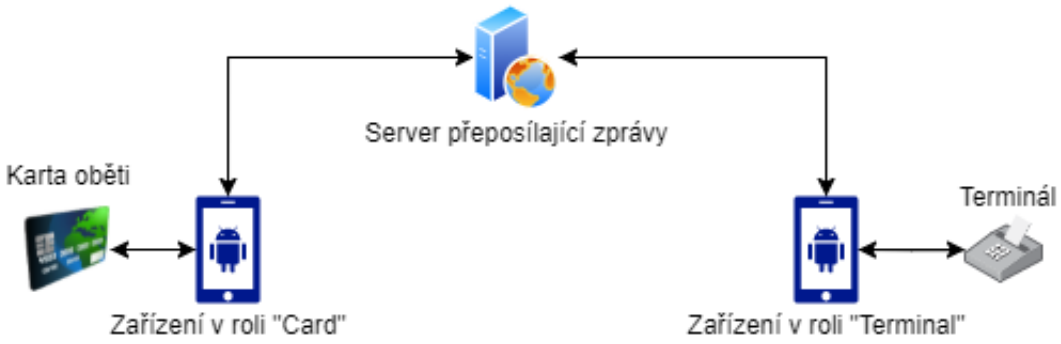
Struktura kódu aplikace je rozdělena na třídy, nebo-li aktivity, které obsluhují různé funkce v aplikaci. **Třídy:**

1. **RelayMainActivity** - Hlavní aktivita pro funkce části Relay. Volají se z ní všechny další aktivity.
2. **ScanAIDActivity** - Aktivita sloužící k získání AID z karty pomocí útoku hrubou silou.
3. **ScanTerminalActivity**- Aktivita pro čtení AID z terminálu.
4. **HCE_Service** - Služba obsluhuje emulaci karty a komunikuje s terminálem.
5. **CardEmulatorActivity** - Aktivita běžící na útočícím zařízení přiloženém na terminálu, která zpracovává příchozí APDU zprávy od terminálu a komunikuje se serverem.
6. **CardReaderActivity** - Aktivita běžící na útočícím zařízení přiloženém na kartě, která komunikuje s kartou a se serverem.

7. **Adapter** - Třída zařizující část grafického rozhraní pro aktivity CardEmulatorActivity a CardReaderActivity.
8. **MifareMainActivity**- Hlavní aktivita pro funkce v části Mifare. Volají se z ní všechny další aktivity potřebné ke klonování karty.
9. **MifareScanActivity** - Aktivita sloužící k přečtení dat z karty a jejich uložení.
10. **MifareSavedActivity** - Aktivita ukazující uložené karty.
11. **MifareCloneActivity** - Aktivita pro klonování Mifare Classic karet.
12. **Utils** - Soubor pomocných funkcí.

4.1 Relay

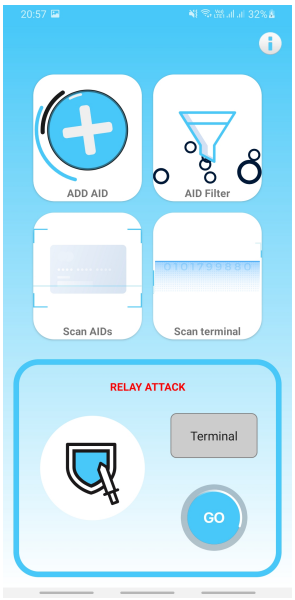
Při výběru ikony „Relay“ na úvodní obrazovce se uživatel dostane do této aktivity. Hlavním cílem této aktivity je úspěšný útok přeposláním na reálné autentizační systémy používající APDU zprávy. Na obrázku 4.2 je diagram popisující strukturu vytvořeného systému pro útok přeposláním zpráv.



Obr. 4.2: Úvodní obrazovka RR aplikace pro Android

Jedním z nejdůležitějších kroků je získat AID, se kterým bude terminál komunikovat. Tedy AID, který bude mobilní zařízení v HCE módu filtrovat. Na tento problém existují dvě řešení. Nejpraktičtějším řešením je nastavit AID filtrování pro všechny příchozí dotazy na výběr AID. K tomuto je nutné mít rootnuté zařízení a nainstalovaný modul od Xposed Framework s názvem **NFC HCE Catch-All-Routing**. Root mobilu a přidání modulu je popsáno v kapitole 6 „Testování systému“. Dále je nutné filtrovat speciální AID s hodnotou „F04E66E75C02D8“ pro filtrování všech AID. Tato hodnota je dána autorem modulu. Modul vytvořil Johannes Zweng a je veřejně dostupný zde [45]. Modul by měl oficiálně fungovat s Android zařízeními od **verze 4.4 do verze 9.0**. Při testování ale modul v pořádku běžel i na Android verzi 10.0. Druhým již méně praktickým řešením je manuálně přidat AID pro filtrování, které samozřejmě uživatel musí předem znát, což je ve většině případů velký problém, protože AID nemusí být veřejně známé.

Nelze předpokládat, že všichni uživatelé aplikace mají rootnuté zařízení, a proto jsou zde obsaženy nástroje, které mohou tito uživatelé použít pro připravení na samotný útok přeposláním. Tyto nástroje včetně aktivity pro útok přeposláním lze vidět na obrázku 4.3. Celou aktivitu řídí třída **RelayMainActivity**, ze které se volají funkce a další aktivity.



Obr. 4.3: Obrazovka aktivity Relay

Filtrování AID se děje při emulaci karty, z toho důvodu je nutné nastavit vlastnosti pro NFC, viz řádek 1 výpisu 4.1, a HCE v souboru **AndroidManifest.xml**, viz řádky 3-18 výpisu 4.1. Pro funkčnost HCE je zapotřebí vytvořit službu, která filtruje intent pro „android.nfc.cardemulation.action.HOST_APDU_SERVICE“, viz řádek 11 výpisu 4.1, a zdrojový kód, který jej bude obsluhovat, viz řádek 4 výpisu 4.1. AID filtr se běžně přidává staticky do metadat HCE služby v manifestu, viz řádek 17 výpisu 4.1. V této aplikaci se ale nebude tento filtr používat, protože je potřeba filtr dynamicky měnit. Způsob dynamického přidávání AID bude dále vysvětlen.

Výpis 4.1: AndroidManifest.xml: povolení NFC a HCE

```
1 <uses-permission android:name="android.permission.NFC" />
2  ...
3 <service
4     android:name=".HCE_Service"
5     android:enabled="true"
6     android:exported="true"
7     android:permission=
```

```
8           "android.permission.BIND_NFC_SERVICE">
9       <intent-filter>
10      <action android:name=
11  "android.nfc.cardemulation.action.HOST_APDU_SERVICE"
12      />
13       </intent-filter>
14      <meta-data
15          android:name=
16              "android.nfc.cardemulation.host_apdu_service"
17          android:resource="@xml/apduservice" />
18   </service>
```

Funkce ADD AID

První ikona na obrázku 4.3 na předchozí straně představuje funkci pro manuální přidávání AID do AID filtru aplikace. Při stisknutí ikony je funkce volána ze třídy RelayMainActivity a objeví se AlertDialog, který vyzve uživatele k vložení AID. Po potvrzení přidání se zkontroluje, zda je vložený AID v platném formátu. AID se přidá jen v případě, že je ve správném formátu.

Díky funkci `registerAidsForService()` je možné dynamické přidávání AID do filtru. Funkce bere v argumentu proměnlivý seznam obsahující jeden nebo více AID a registruje je do filtru. Pro testovací účely je již připravený **startovací seznam s AID** pro Mifare DESFire, EMV standard fungující v České Republice, tedy MasterCard, Visa a jejich startovací příkaz transakce PPSE a PSE. Pro rootnuté telefony je zde vložen také speciální AID pro filtraci všech AID, viz výpis 4.2. Tyto AID jsou pro případ chyby dynamického přidání vloženy také staticky v souboru **apduservice.xml**.

Výpis 4.2: Startovací seznam s AIDs

```
1 val startAIDList = mutableListOf<String>(
2     "325041592E5359532E4444463031",   // PPSE
3     "315041592E5359532E4444463031",   // PSE
4     "A0000000041010",                 // MasterCard
5     "A0000000031010",                 // VISA
6     "D2760000850100",                 // Mifare DESFire
7     "F04E66E75C02D8"     // speciální AID (Xposed modul)
8 )
```

Funkce AID Filter

Pro přehlednost je dobré mít možnost zjistit, jaké AID aplikace filtruje. K tomu je zde ikona AID Filter. Při stisknutí ikony je funkce volána ze třídy **RelayMainActivity** a objeví se AlertDialog se seznamem všech AID, které aplikace nyní filtruje. Funkcionalitu poskytuje funkce `getAidsForService()`, která při zavolání vrací dynamicky přidaná AID, viz řádek 1 výpisu 4.3. Tedy nevrací AID staticky vložené v manifestu. Všechny AID se postupně přidají do jedné pomocné proměnné, viz řádky 2-4, která se pak vypíše do AlertDialogu, viz řádek 6 výpisu 4.3.

Výpis 4.3: Získání AIDs a vypsání do AlertDialogu

```
1 cardEmulation.getAidsForService(componentName, "other")
2  .forEach {
3     aidString = aidString + "\n" + it
4  }
5 ...
6 builder.setMessage(aidString)
```

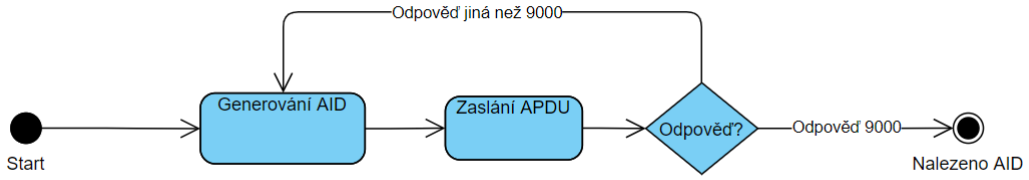
Funkce Scan AIDs

Je jen málo způsobů, jak získat neznámé AID z karty. Jedním z nich je útok hrubou silou. Po vybrání ikony Scan AID je volána ze třídy RelayMainActivity nová aktivita **ScanAIDActivity**. Způsob užití této aktivity je následující. Prvně se přiloží chytrá karta k mobilnímu zařízení, při nalezení se objeví upozornění a při stisku tlačítka „Start Scan“ začne zařízení skenovat kartu pro její AID. Logika skenování, stisknutí tlačítek a měnícího se grafického uživatelského rozhraní je zakomponována ve funkci `scanAIDs()`.

Skenování karty je prováděno tak, že mobilní zařízení, které je ve čtecím módu, posílá příkazy na výběr všech možných AID. K tomu je použita rekurzivní funkce `bruteForceRecursive()`, která generuje postupně všechny AID od minimální délky 10 znaků až po maximální délku 32 znaků v kombinaci s 16 znaky hexadecimální soustavy. S využitím pomocné funkce `createAPDUCommand()` se vytváří příkazy pro výběr, které jsou následně funkcí `transceive()` zasílány na kartu. Vráti-li se z karty odpověď 9000, tak je nalezen AID. Stavový diagram skenování AIDs je zobrazen na obrázku 4.4 na následující straně.

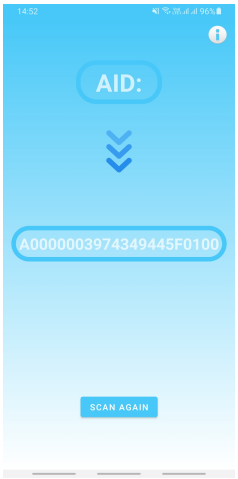
Funkce Scan Terminal

Posledním nástrojem pracujícím s AID je aktivita Scan Terminal. Účelem této aktivity je získat AID přímo z terminálu. Podmínkou použití této funkcionality je



Obr. 4.4: Scan AIDs - Diagram

rootnutý telefon. Terminály obvykle posílají jako první APDU zprávu výběr AID. Zařízení přiložené na terminálu musí tedy filtrovat dotazy na všechny AID, aby mohlo tuto zprávu zachytit a přijmout. Z toho důvodu je zde nutný modul NFC HCE Catch-All-Routing. Postup získání AID je jednoduchý. Po stisknutí ikony „Scan Terminal“ je volána ze třídy RelayMainActivity nová aktivita **ScanTerminalActivity**. Poté stačí přiložit zařízení k terminálu. V případě nalezení AID se data vypíší na obrazovku, viz obrázek 4.5.



Obr. 4.5: Scan Terminal - nalezené AID

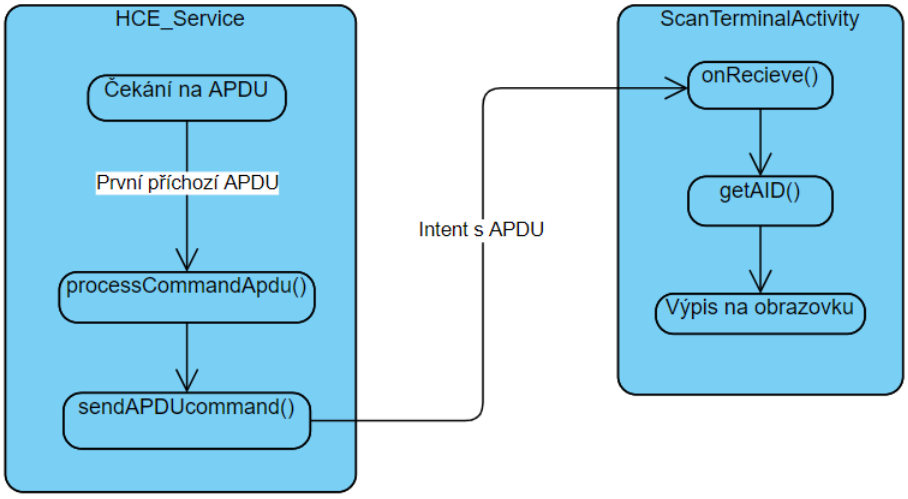
Z technického hlediska je tedy zařízení v HCE módu a čeká, než zaznamená příchozí APDU zprávu. HCE mód pro celou aplikaci zajišťuje třída **HCE_Service**, kde se zpracovávají příchozí APDU zprávy. Každá příchozí zpráva vyvolá funkci `processCommandApdu()`. Zde se posílá první zpráva pomocí funkce `sendAPDUcommand()`. Třída **ScanTerminalActivity** pomocí funkce `onReceive()` zachytí zaslanou zprávu a dále se ze zprávy extrahuje AID prostřednictvím pomocné funkce `getAID()` a vypíše se na obrazovku. Přeposílání dat mezi aktivitami zajišťuje třída **LocalBroadcastManager** a její funkce `sendBroadcast()`.

Funkce `sendAPDUcommand ()` má jako argument APDU příkaz, který vkládá

jako data do intentu s akcí „sendAPDUcommand“, viz řádek 2-5 výpisu 4.4. Intent s daty pak posílá funkce `sendBroadcast()` dále, viz řádek 6-7 výpisu 4.4. Celý proces zasílání a přijímání je popsán v diagramu na obrázku 4.6.

Výpis 4.4: Funkce sendAPDUcommand()

```
1 fun sendAPDUcommand (commandApdu: ByteArray) {
2     val intent = Intent().apply {
3         putExtra("data", byteArraytoHex(commandApdu))
4         action = "sendAPDUcommand"
5     }
6     LocalBroadcastManager.getInstance(this)
7         .sendBroadcast(intent) }
```



Obr. 4.6: Scan Terminal - diagram

4.1.1 Relay Attack

Po zajištění AID filtrování již následuje hlavní funkce této aktivity, a to útok přeposláním. Pro tento útok jsou nutné dvě mobilní zařízení a komunikační kanál mezi nimi, kterým si budou přeposílat zprávy. Každé zařízení má jednu roli, buď **„Card“** a nebo **„Terminal“**. Zařízení s rolí Card je přiloženo ke kartě oběti a zařízení s rolí Terminal je přikládáno k terminálu. Jako komunikační kanál je zde vybrán WebSocket server, který je připojený k zabezpečené webové stránce. Server i webová stránka budou blíže popsány dále v kapitole 5 „RR Demonstrátor - Serverová a webová část“.

Z toho důvodu, že je komunikační kanál veden přes internet, je potřeba v manifestu přidat navíc povolení, aby aplikace mohla používat síť Internet, viz výpis 4.5.

Výpis 4.5: AndroidManifest.xml: povolení Internetu

```
1 <uses-permission android:name=
2     "android.permission.INTERNET" />
```

Role „Card“

Předpokladem pro roli „Card“ je funkční NFC hardware schopný čtení karty a funkční internetové připojení. Uživatel je po výběru této role a stisknutí tlačítka „GO“ přesunut do aktivity **CardReaderActivity**. Na obrázku 4.7 (str. 47) je zobrazena tato aktivita a její komponenty.

Aktivita obsahuje 4 indikátory, které mění barvu dle stavu, který sledují. První indikátor sleduje konektivitu zařízení se serverem. Při vzniku aktivity se zavolá funkce `startSocketConnection()`, která připojí zařízení k serveru. Pro připojení je nutný nějaký klient. Zde je použit **OkHttpClient**. WebSocket se připojuje k webovému serveru, tudíž je vytvořena žádost na zadaný server, viz řádek 3-5 výpisu 4.6. Žádost je zaslána pomocí funkce `newWebSocket()` ze třídy OkHttpClient, viz řádek 7 výpisu 4.6. Je-li připojení úspěšné, zavolá se funkce `onOpen()` ze třídy SocketListener a nastaví se indikátor na zelenou barvu.

Výpis 4.6: Funkce startSocketConnection()

```
1 private fun startSocketConnection() {
2     val client = OkHttpClient()
3     val request = Request.Builder()
4         .url(SERVER_PATH)
5         .build()
6     val socketListener = SocketListener()
7     socket = client.newWebSocket(request, socketListener)
8 }
```

Druhý indikátor ukazuje připojení druhého zařízení v roli „Terminal“. Server udržuje informace o připojených zařízeních a v případě, že je zařízení v roli „Terminal“ připojeno, tak odesílá zařízení v roli „Card“ zprávu, že je přítomen a indikátor je zbarven zeleně. V případě, že se zařízení v roli „Terminal“ odpojí, server odešle zprávu, že se odpojil a indikátor je nastaven na červenou barvu. Příchozí zprávy zpracovává funkce `onMessage()` ze třídy **SocketListener**, která se zavolá vždy,

když server odešle zařízení zprávu. Zprávy jsou posílány jako **JSON objekty**, proto je nutné z nich extrahovat textový řetězec se zprávou, viz řádek 4-5 výpisu 4.7.

Výpis 4.7: Funkce onMessage()

```
1 override fun onMessage(webSocket: WebSocket, text: String)
2 {
3     ...
4     val jsonObject = JSONObject(text)
5     APDUCommand = jsonObject.getString("message")
6     ...
7 }
```

Třetí indikátor zobrazuje stav připojení karty oběti k zařízení. Při nalezení karty se zavolá funkce `onTagDiscovered()` ze třídy **NfcAdapter.ReaderCallback**. V této funkci se dále uskuteční připojení ke kartě pomocí funkce `get()`, která se volá ze třídy **IsoDep**, viz řádek 3 výpisu 4.8, protože karty na které se útočí jsou typu IsoDep. Je-li připojení zdárně uskutečněno, tak se indikátor zbarví na zelenou barvu, viz řádek 5-9 výpisu 4.8. Kontrola konektivity dále běží ve smyčce `loop()`. Funkcí `isConnected()` se zjišťuje, zda je karta připojena nebo ne. V okamžiku, kdy je karta odpojena, barva indikátoru se změní na červenou.

Výpis 4.8: Funkce onTagDiscovered()

```
1  override fun onTagDiscovered(tag: Tag?) {
2      ...
3      isoDep = IsoDep.get(tag)
4      ...
5      runOnUiThread {
6          b3.backgroundTintList =
7              ColorStateList.
8                  valueOf(Color.parseColor("#16ff16"))
9          }
10     ...
11     loop()
12 }
```

Poslední - čtvrtý indikátor slouží jako finální kontrola před samotným útokem. Pokud jsou všechny předchozí indikátory zeleně zbarvené, i tento se zbarví zeleně. V případě, že je jeden z indikátorů červený, tak je finální indikátor rovněž červený.

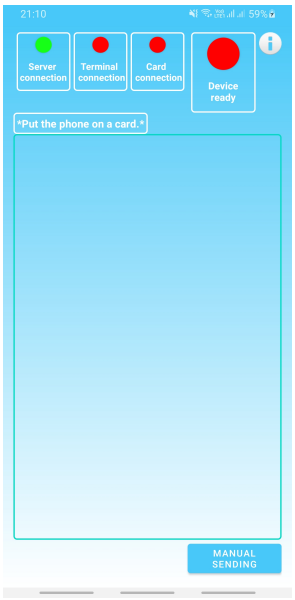
Uprostřed obrazovky je vloženo pole, kde se vypisují zprávy, které jsou zasílány mezi dvěma útočícími zařízeními. Pod tímto polem je tlačítko, při jehož stisku se

objeví možnost manuálního zasílaní zpráv. Pole pro vypisování i manuální zasílání zpráv jsou vytvořená na základě chatovací aplikace od Hey! Let's Code [46].

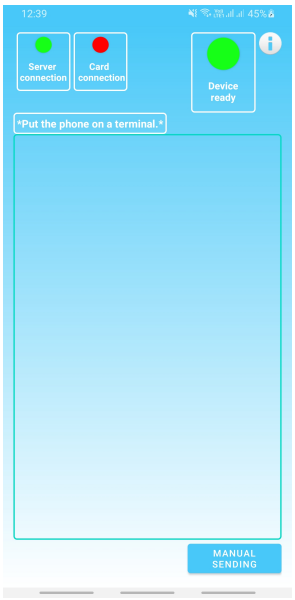
Role „Terminal“

Zařízení zastupující roli „Terminal“ by mělo ideálně mít root a NFC HCE Catch-All-Routing. Modul ani root není nutností. Útok lze provést i bez něho. Dále je, podobně jako u role „Card“, zapotřebí funkční NFC hardware schopný HCE, tedy emulování karty, a funkční internetové připojení.

Ve srovnání s aktivitou CardReader je tato aktivita identická, jen s rozdílem, že chybí indikátor připojení karty, protože zde není potřeba. Na obrázku 4.8 je zobrazena aktivita CardEmulator.



Obr. 4.7: Obrazovka aktivity CardReader



Obr. 4.8: Obrazovka aktivity CardEmulator

4.1.2 Relay Attack na EMV

Speciálním případem je útok na **EMV**. Jako první zpráva se zpravidla posílá **PPSE**, která na Android OS zařízeních automaticky vyvolává aplikaci **Google Pay** nad RR demonstrátorem a nelze kvůli tomu pokračovat v útoku. Je to z toho důvodu, že Android rozpoznává PPSE jako inicializační APDU příkaz pro platby a aplikace Google Pay je jediná legitimní aplikace pro platby pomocí HCE módu. Tento problém lze obejít tím, že se deaktivuje aplikace Google Pay. Až po úspěšné deaktivaci aplikace Google Pay na HCE zařízení je možné pokračovat v útoku.

4.1.3 Shrnutí logiky útoku

Útok je založen na přeposílání APDU zpráv mezi terminálem a kartou. První zprávu odesílá terminál, tu zachycuje zařízení v roli „Terminal“. K tomu je třeba mít znalost AID, na které se terminál ptá, a nebo filtrovat všechny AID pomocí modulu NFC HCE Catch-All-Routing. Je-li zachycení příkazu select APDU úspěšné, nejtěžší část je splněna.

Zpráva se posílá na server, který ji přepošle na druhé zařízení v roli „Card“. Toto zařízení je ve čtecím módu a posílá zprávu ze serveru na přiloženou kartu. Karta odpoví na zprávu a odpověď se posílá na server. Server ji přepošle na první zařízení a to odpověď posílá čekajícímu terminálu. Terminál dle odpovědi posílá další zprávu a nebo komunikaci ukončí. Počet cyklů přeposílání dotazu od terminálu a odpovědi od karty závisí na autentizačním protokolu terminálu, který vyhodnocuje autentizaci karty. Komunikace tedy končí až v momentě, kdy terminál přestane posílat zprávy. Celou logiku útoku lze vidět na obrázku 4.9 na následující straně.

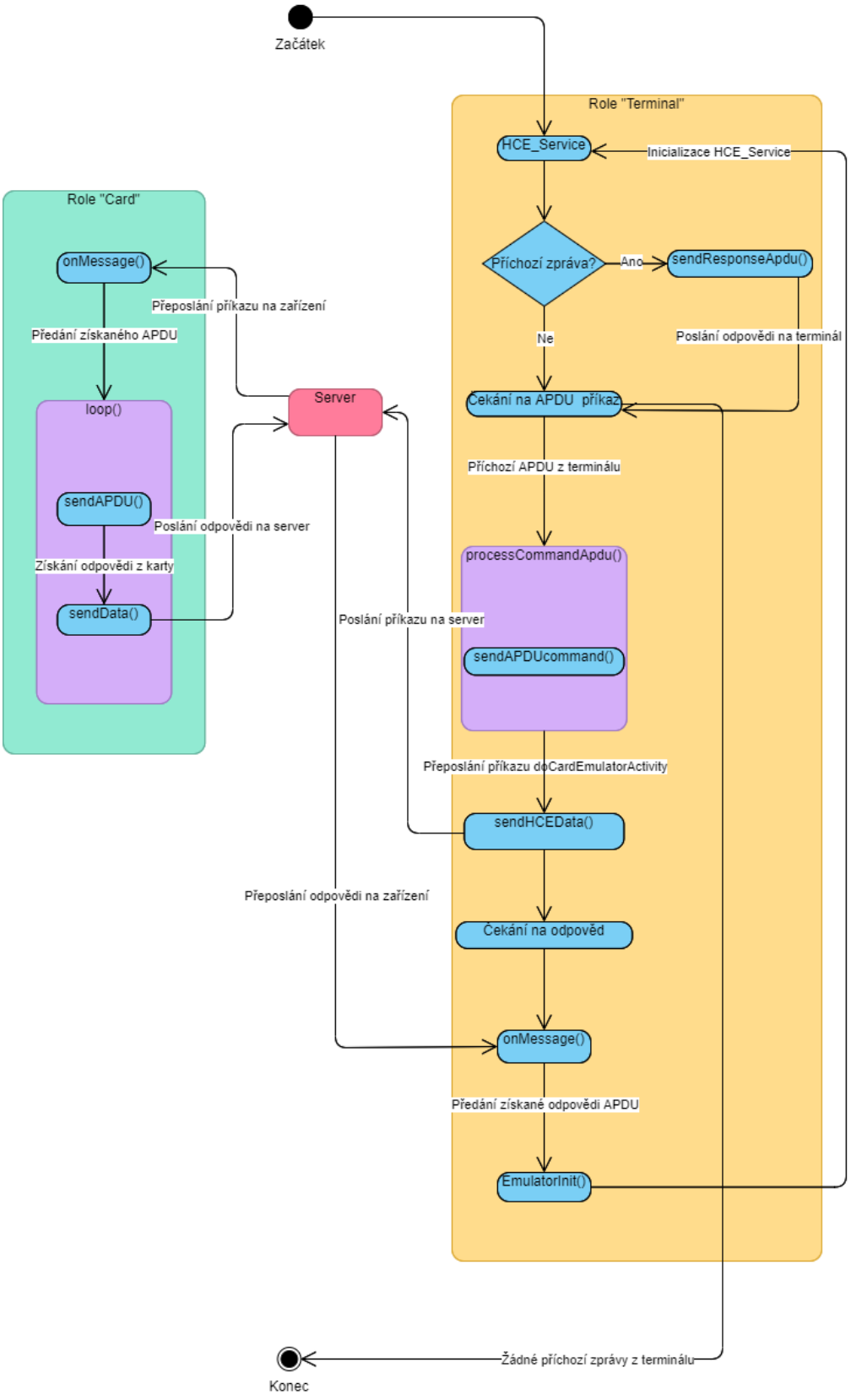
Jeden cyklus zpráv musí být co nejkratší, tzn. zpoždění je co nejmenší, aby legitimní terminál nebo karta nepoznala, že se jedná o útok přeposláním. Některé systémy a karty nemají nastavený žádný časový limit posílání zpráv, a proto nemusí být komunikační kanál přeposílající zprávy dokonalý. Naopak ale existují technologie, které by měly útok přeposláním zmírnit, například Mifare DESFire EV3.

V kódu roli „Terminal“ obstarává třída **HCE_Service** a **CardEmulatorActivity**. Účelem třídy HCE_Service je obsluhovat emulování karty. Jedna z hlavních funkcí je `processCommandApdu()`, která je volána při zachycení APDU zprávy. Tyto zprávy jsou poté přeposílány do třídy CardEmulatorActivity funkcí `sendAPDUcommand()` ze třídy **LocalBroadcastManager**. Zpráva se poté posílá na server pomocí funkce `sendHCEData()`. Ve výpisu 4.9 je na řádku 1-3 ukázáno vytvoření zprávy typu **JSONObject**. Do objektu jsou vkládány dva parametry. První je identifikace odesílatele a druhý je zpráva samotná. Na řádku 4 je již samotné odeslání zprávy pomocí funkce `send()`.

Výpis 4.9: Zasílání dat na server

```
1 val jsonObject =JSONObject()
2                  .put("name", name)
3                  .put("message", data)
4 socket.send(jsonObject.toString())
```

Po odeslání zprávy zařízení čeká na odpověď ze serveru. Příchozí odpovědi zachycuje funkce `onMessage()`, ty se přeposílají do třídy HCE_Service pomocí funkce `EmulatorInit()` využívající volání intentů. Po zavolání služby HCE_Service je volána funkce `onStartCommand()`, která extrahuje APDU zprávu z poslaného intentu



Obr. 4.9: Útok přeposláním - diagram

z funkce `EmulatorInit()`. Poté se použije funkce `sendResponseApdu()` pro zaslání odpovědi na terminál.

Roli „Card“ obsluhuje třída **CardReaderActivity**. Klíčová funkce je zde `loop()`, která se nachází ve funkci `onTagDiscovered()`. Jejím hlavním účelem je synchronizace příchozích zpráv a zasílaní odpovědí z karty. Zasílání zpráv na kartu zařizuje funkce `sendAPDU()`, která zároveň vrací z karty odpověď na APDU příkaz. Zasílání odpovědí z karty na server je provedeno pomocí funkce `sendData()`. Příchozí zprávy ze serveru jsou stejně jako u třídy CardEmulatorActivity zpracovávány ve funkci `onMessage()`. Při práci s IsoDep je nutné se na kartu připojit a po skončení komunikace je nutné se odpojit. Nelze mít kartu stále připojenou z důvodu, že by se karta mohla z nějakého vnějšího důvodu odpojit, a z tohoto důvodu by nastala chyba. Dále není jisté, jak dlouho bude útok trvat. Proto je při první příchozí zprávě nastaven časovač na 10 sekund. Po 10 sekundách se karta odpojí funkcí `close()` a zase připojí funkcí `connect()`. To znamená, že komunikace může trvat maximálně 10 sekund. Po nalezení karty běží funkce `loop()` v nekonečné smyčce kvůli kontrole indikátorů a příchozích zpráv. Při ztrátě spojení s kartou však smyčka zanikne, protože ji již není potřeba. Na obrázku 4.10 dále je zobrazen diagram funkce `loop()`.

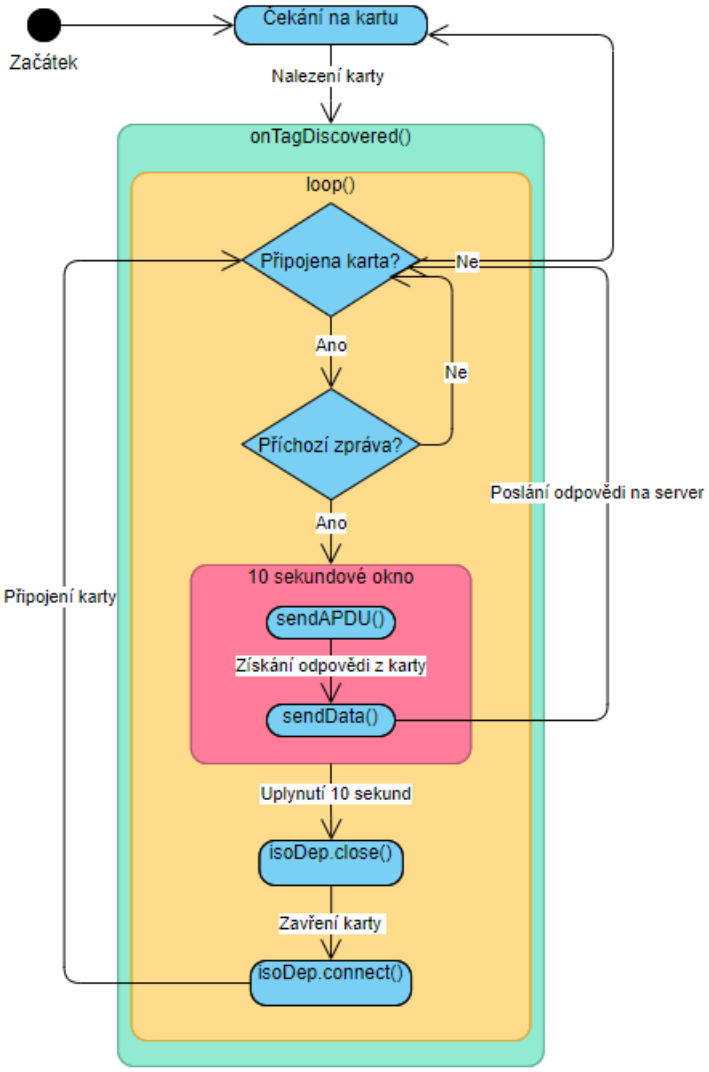
Při zasílaní zpráv ze zařízení s HCE se občas vyskytuje takový problém, že se zpráva odešle dvakrát rychle za sebou, i když terminál zasílá jen jednu zprávu. Zařízení ve čtecím módu ale zvládá odesílat odpověď z karty rychleji než přijde druhá duplikovaná zpráva a nastává chyba. Z tohoto důvodu je ve funkci `loop()` přidáno zpoždění 120 milisekund pomocí funkce `Thread.sleep()`. I kdyby HCE zařízení zasílalo duplikované zprávy, tak čtecí zařízení zaznamená jen jednu zprávu.

4.2 Mifare

K úplnému užití aktivity Mifare je zapotřebí přepisovatelná GEN2 karta a NFC hardware podporující Mifare Classic, což jsou například čipové sety **NXP** nebo **Samsung čipy**. Pro účel klonování byly naprogramovány nástroje, které lze vidět na obrázku 4.11 (str. 52). Aktivitu s nástroji obsluhuje třída **MifareMainActivity**, ze které se dále volají aktivity pro jednotlivé nástroje.

4.2.1 Scan Mifare

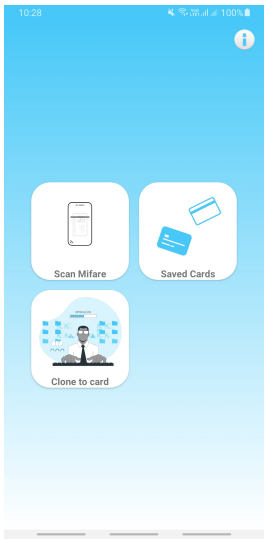
Nejdříve je potřeba získat data sektoru 0, aby bylo možné klonování. Proto je stěžejní částí pro aktivitu Mifare nástroj Scan Mifare. Ke čtení dat z karty jsou zapotřebí klíče, které autentizují přístup k datům sektorů. K tomuto účelu jsou v aplikaci pevně uloženy výchozí klíče pro Mifare Classic. V případě potřeby může uživatel přidat nové klíče pomocí funkce **Add Key**. Pro zobrazení všech přítomných klíčů



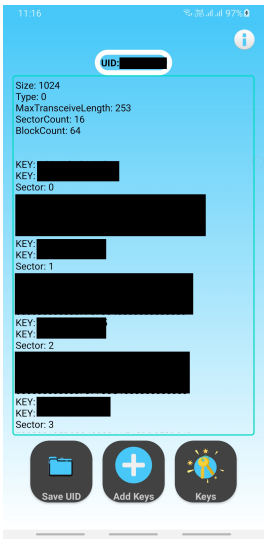
Obr. 4.10: Funkce loop() - diagram

stačí kliknout na ikonu **Keys**. Po definování klíčů stačí přiložit kartu k mobilnímu zařízení a počkat, než se objeví přečtená data. Z toho důvodu, že klonováním se zde rozumí nastavení GEN2 karty takovým způsobem, aby měla stejné UID jako originální karta, je zde vytvořená funkce **Save UID**. Tuto funkci uživatel může použít v případě, že si chce přečtenou kartu pojmenovat a uložit pro pozdější klonování. Na obrázku 4.12 lze vidět textové okno s přečtenými daty Mifare Classic karty včetně všech funkcí.

Nástroj obsluhuje třída **MifareScanActivity**. Přidávání klíčů zařizuje v kódu funkce `addMifareKey()`, která po vyplnění přidávaného klíče a potvrzení přidá klíč do souboru s klíči pomocí funkce `appendLineToFile()`. Zobrazení klíčů zajišťuje funkce `mifareKeys()`, jež čte soubor s klíči řádek po řádku funkcí `readFileByLines()` a zobrazí je na obrazovku.



Obr. 4.11: Obrazovka aktivity Mifare

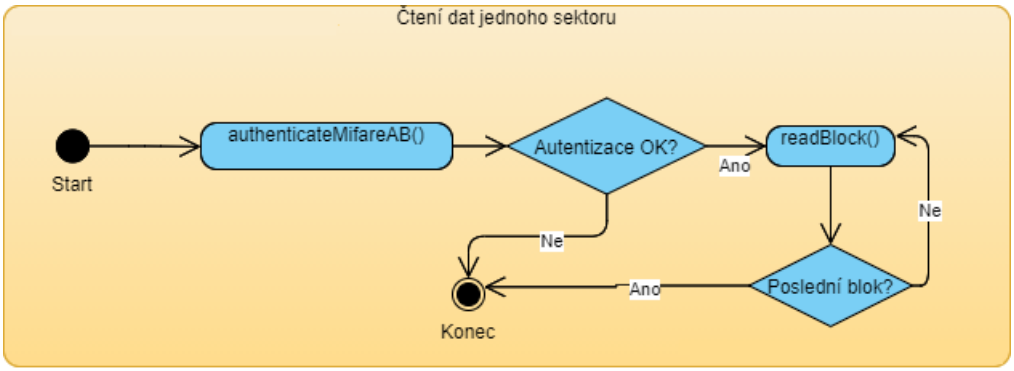


Obr. 4.12: Obrazovka aktivity MifareScan

Při čtení karty je v momentě zachycení karty volána funkce `onTagDiscovered()` a v ní je volána funkce `getMifareClassicinfo()`, která z karty postupně přečte a vypíše na obrazovku její velikost, typ, maximální délku posílaných dat, počet sektorů a počet bloků, viz výpis 4.10. Poté je volána klíčová funkce `readMifareClassic()`, která čte sektor za sektorem a blok po bloku. Čtení dat jako takové zařizuje funkce `readBlock()`. Data se po přečtení vypisují do textového okna. Před čtením bloků v sektorech je ještě volána funkce `authenticateMifareAB()` pro autentizaci. Autentizace probíhá tak, že se zkouší všechny klíče v souboru s klíči pro klíč A (funkce `authenticateSectorWithKeyA()`) i klíč B (funkce `authenticateSectorWithKeyB()`). Až po úspěšné autentizaci je možno data číst. Ukládání dat karty je provedeno pomocí **JSON objektů.** JSON objekt obsahuje dva záznamy a to data karty a její pojmenování. Tyto objekty se ukládají do jednoho souboru, ze kterého mohou být dále čteny. Na obrázku 4.13 je zobrazen cyklus čtení jednoho sektoru.

Výpis 4.10: Funkce getMifareClassicinfo()

```
1 private fun getMifareClassicinfo(tag: MifareClassic) {
2         textScrollMifareScan.append(
3 "Size:" + tag.size + "\n" + "Type:" + tag.type + "\n" +
4 "MaxTransceiveLength:" + tag.maxTransceiveLength + "\n" +
5 "SectorCount:" + tag.sectorCount + "\n" +
6 "BlockCount:" + tag.blockCount + "\n\n\n"
7         )}
```



Obr. 4.13: Čtení jednoho sektoru - diagram

4.2.2 Saved Cards

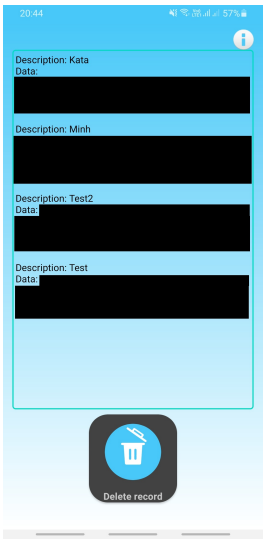
Nástroj Saved Cards slouží jako jednoduché vypsání všech uložených karet ze souboru pro ukládání karet. V této aktivitě je možné uložené karty si nejen prohlížet ale i mazat. Při otevření aktivity se zobrazí všechny uložené karty. Pro smazání stačí stisknout tlačítko **„Delete record“**, zadat jméno karty a potvrdit. Nástroj obsluhuje třída **MifareSavedActivity**. Vypisování zařizuje funkce `printAllSavedCards()`, která přečte v souboru pro ukládání karet všechny JSON objekty představující uložené karty a vypíše na obrazovku jméno karty a její data ze sektoru 0. Mazání zařizuje funkce `deleteRecord()`. Funkce vyhledá v souboru s kartami JSON objekt s jménem odpovídajícím zadanému jménu a objekt smaže. Na obrázku 4.14 na následující straně lze vidět grafické uživatelské rozhraní této aktivity.

4.2.3 Clone to card

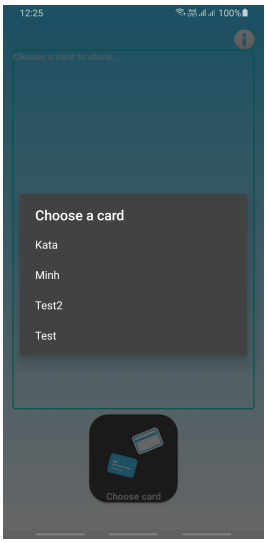
Klonování karty je v podstatě psaní originálních dat do druhé karty. Originální data si uživatel již předtím naskenoval pomocí nástroje Scan Card a uložil. V této fázi si uživatel vybírá, kterou kartu chce klonovat stisknutím ikony **„Choose Card“**. Po stisknutí se objeví seznam všech uložených karet. Vybírání karty pro klonování lze vidět na obrázku 4.15. Vybráním některé karty se na obrazovku vypíší data ze sektoru 0 a nyní je vše připraveno ke klonování. Posledním krokem je přiložení GEN2 karty pro zapsání dat.

Klonování obstarává třída **MifareCloneActivity**. Načtení karty ze souboru provádí funkce `loadZeroSector()`, která v souboru uložených karet hledá JSON objekt odpovídající vybrané kartě a extrahuje z ní data odpovídající sektoru 0. Pro psaní je shodně jako u čtení potřeba autentizace sektorů. Využívá se zde tedy stejná funkce `authenticateMifareAB()`. Zde není problém s nalezením klíčů, protože nové GEN2 karty jsou obvykle zabezpečené výchozími klíči. Psaní realizuje funkce `writeToMifareClassic()`, která načtená data píše blok po bloku do sektoru

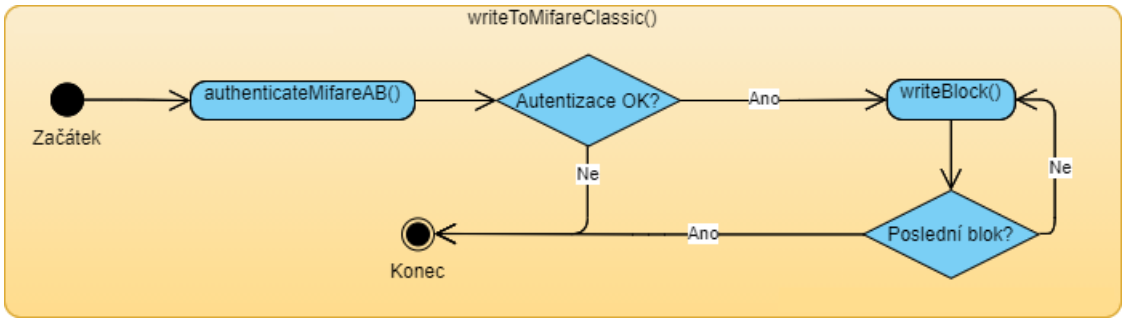
0. Psaní do bloků zařizuje funkce `writeBlock()`. Jak lze vidět na obrázku 4.16, psaní a čtení dat karty je téměř stejné.



Obr. 4.14: Obrazovka aktivity MifareSavedCards



Obr. 4.15: Obrazovka aktivity MifareClone

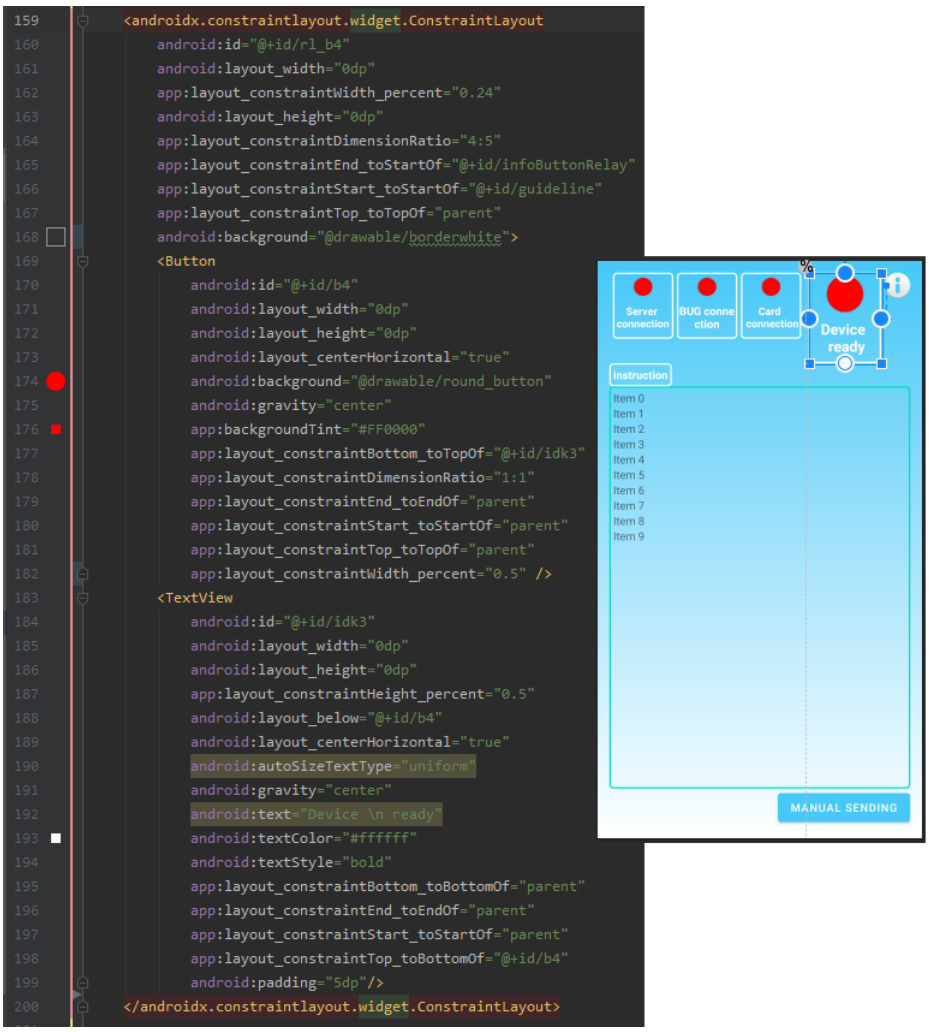


Obr. 4.16: Psaní do sektoru 0 - diagram

4.3 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní je vytvořeno pomocí **ConstraintLayout**. Tento layout byl zvolen kvůli své schopnosti přizpůsobovat se různým poměrům a velikostem obrazovky mobilních zařízení. Pro seskupení ikon nástrojů byl použit **GridLayout** v kombinaci s **CardView**. Zobrazování zpráv v aktivitě Relay Attack využívá textové pole **RecyclerView**, do kterého se přidávají data dle příchozích a odchozích. Přidávání dat do RecyclerView obsluhuje třída **Adapter**. Animace ikon, tlačítek, úspěchů a neúspěchů v aplikaci obstarává otevřený software **Lottie** [47]. Animace

použité v aplikaci nebyly vlastnoručně vyrobeny, byly vybrány z volně dostupných animací na oficiální stránce LottieFiles. Animace byly dále barevně upravovány pro estetiku.



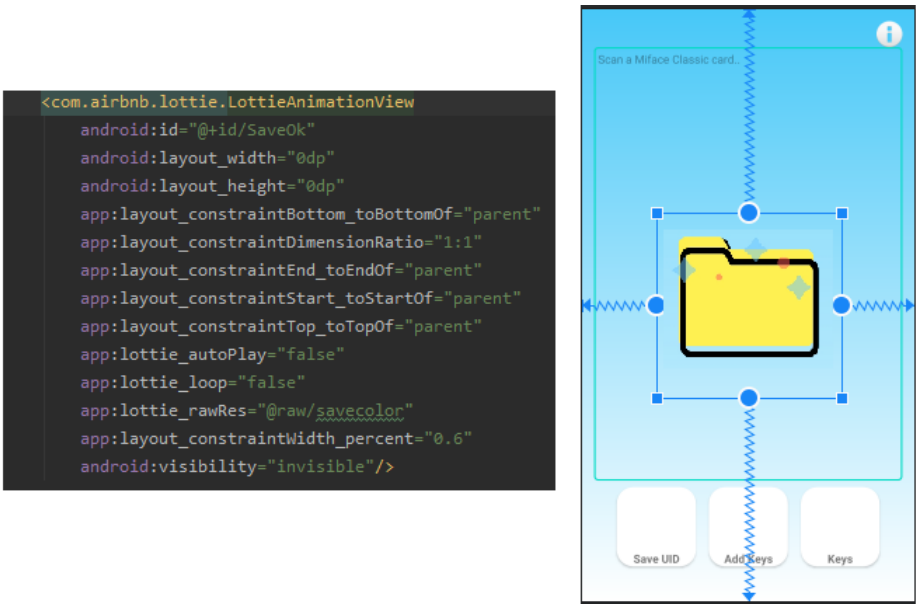

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/rl_b4"
    android:layout_width="0dp"
    app:layout_constraintWidth_percent="0.24"
    android:layout_height="0dp"
    app:layout_constraintDimensionRatio="4:5"
    app:layout_constraintEnd_toStartOf="@+id/infoButtonRelay"
    app:layout_constraintStart_toStartOf="@+id/guideline"
    app:layout_constraintTop_toTopOf="parent"
    android:background="@drawable/borderwhite">
    <Button
        android:id="@+id/b4"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_centerHorizontal="true"
        android:background="@drawable/round_button"
        android:gravity="center"
        app:backgroundTint="#FF0000"
        app:layout_constraintBottom_toTopOf="@+id/idk3"
        app:layout_constraintDimensionRatio="1:1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintWidth_percent="0.5" />
    <TextView
        android:id="@+id/idk3"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintHeight_percent="0.5"
        android:layout_below="@+id/b4"
        android:layout_centerHorizontal="true"
        android:autoSizeTextType="uniform"
        android:gravity="center"
        android:text="Device \n ready"
        android:textColor="#ffffff"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/b4"
        android:padding="5dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Obr. 4.17: Indikátor - kód X obrazovka

Každou aktivitu obsluhuje minimálně jeden layout, který tvoří grafické uživatelské rozhraní. Všechny layouty jsou umístěny v balíčku **layout**. Základním prvkem ConstraintLayout je **constraint** (česky omezení). Tyto omezení spojují stěny grafického objektu se stěnami jiných objektů a nemají pevnou velikost. A proto se dokáží přizpůsobit různým velikostem obrazovky. Grafický objekt by měl zpravidla mít všechny svoje stěny připojené k nějakému jinému objektu, minimální počet je však 2-3 spojení dle umístění. Výška a šířka se nastavuje na 0dp. Někdy je potřeba určit poměr objektu vůči jiným anebo poměr dimenzí objektu. K tomu lze využít procentuální definice šířky či výšky objektu a poměru stran objektu.

V jednom větším grafickém objektu mohou existovat další menší objekty, viz obrázek 4.17 výše, kde je zobrazen kód jednoho indikátoru a výsledný grafický objekt

indikátoru v celkovém grafickém objektu. V případě ukázky na obrázku představuje indikátor jeden vnořený ConstraintLayout s objekty tlačítka a textového pole.



Obr. 4.18: Animace - kód X obrazovka

Práce s animacemi má stejná pravidla jako všechny jiné grafické objekty v ConstraintLayout. Inicializace je velmi podobná jako v případě obrázků, a to je nutný zdroj grafického prvku. Animace jsou uloženy v balíčku **raw**. Časté využití animací v aplikaci spočívá v animaci úspěchů a neúspěchů po provedené funkci. Z tohoto důvodu je z počátku viditelnost nastavená na **invisible** a automatické spuštění na **false**. Nastavení parametrů v kódu a výslednou animaci lze vidět na obrázku 4.18.

Manipulace s animacemi se děje v kódu aktivity, ve které animace běží. Ve výpisu 4.11 lze vidět část funkce `animationSaveok()` obsaženou ve třídě MifareScanActivity. Pro spuštění je tedy nutné zviditelnit animaci a přehrát ji, viz řádek 1-2 ve výpisu 4.11, po skočení animace je automaticky volána funkce `onAnimationEnd()`, ve které je nastaveno zpětné zneviditelnění animace, viz řádek 5 výpisu 4.11.

Výpis 4.11: Funkce animationSaveok()

```
1 SaveOk.visibility = View.VISIBLE
2 SaveOk.playAnimation()
3 ...
4 override fun onAnimationEnd(animation: Animator?) {
5 SaveOk.visibility = View.INVISIBLE
6 ...
7 }
```

5 RR Demonstrátor - Serverová a webová část

Komunikační kanály mohou být vytvořené různými technologiemi. Na krátké vzdálenosti lze použít kupříkladu **Bluetooth**, na střední vzdálenosti mohou být útočící zařízení v jedné síti **Wi-Fi**, atd. Pro RR demonstrátor je ale zvolen **WebSocket server** dostupný v síti Internet. Díky tomu lze útok přeposláním provést odkudkoliv na světě za podmínky, že mají oba útočníci dostupnost k síti. Tento komunikační kanál by měl být tedy nejpraktičtější. Další výhodou je, že WebSocket je 5 až 7 krát rychlejší než normální https server [48].

Pro uživatelský přehled o dění v aktivitě Relay je vytvořená webová aplikace běžící na stejném serveru jako komunikační kanál.

Struktura kódu serveru je rozdělená na soubory JavaScript, Embedded JavaScript, Cascading Style Sheets a logy.

Soubory:

1. **server.js** - Představuje hlavní část kódu serveru. Vytváří se zde server, přeposílají zprávy, inicializují se webové stránky, zajišťuje se přihlašování a logování.
2. **passport-config.js** - Konfigurační soubor pro přihlašování.
3. **index.ejs** - Vytváří se zde hrubá html struktura webové stránky a skripty pro obsluhu funkcí na stránce.
4. **login.ejs** - Vytváří se zde hrubá html struktura přihlašovací stránky a skript pro animaci Lottie.
5. **index.css** - Soubor pro konfiguraci grafického rozhraní webové stránky.
6. **login.css** - Soubor pro konfiguraci grafického rozhraní přihlašovací stránky.
7. **app.log** - Log všech zpráv. Není veřejně dostupný.
8. **connections.log** - Dynamický soubor ukládající aktivní zařízení na serveru.

5.1 Server

Programovacím prostředím pro vývoj serveru byl zvolen Visual Studio Code a NodeJS. Server je psaný v jazyce JavaScript. Server je implementován v jediném souboru **server.js**. Cílem je zde vytvořit zabezpečený server pro komunikaci, co nejrychlejší přeposílání komunikace a ukládání logů komunikace pro případnou studii protokolu, na který bylo útočeno. Z důvodu dostupnosti je server virtuálně hostován v Nizozemí na operačním systému Ubuntu.

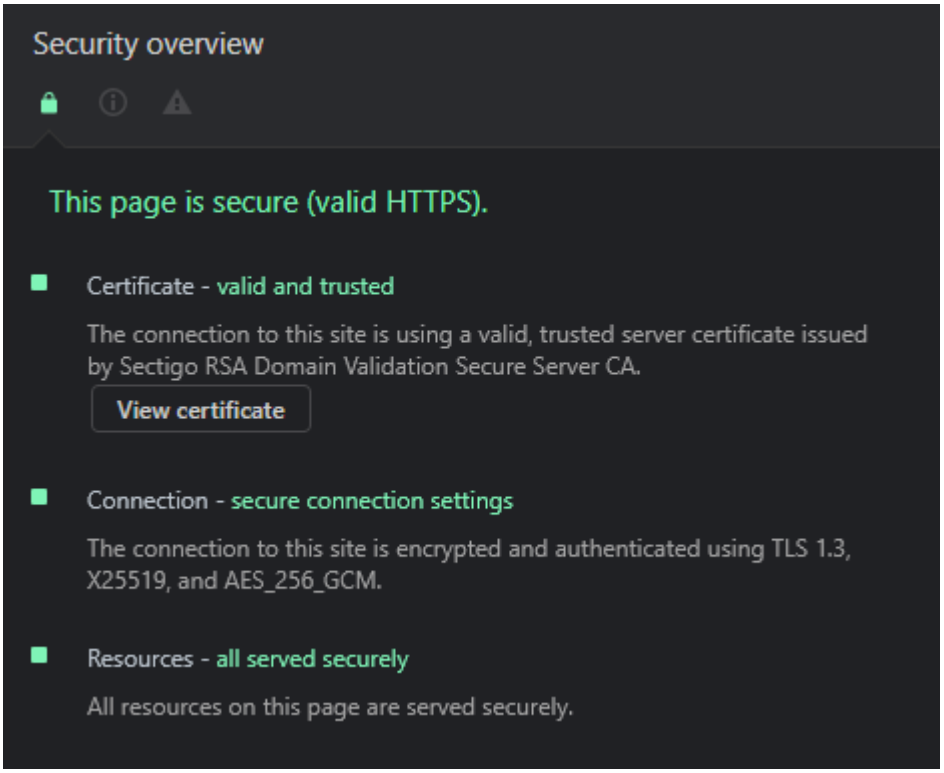
5.1.1 Zabezpečení serveru

Komunikačním kanálem mohou proudit velmi citlivá data, jako například číslo EMV karty, její expirace, jméno vlastníka a další, proto je nutné komunikaci nějakým způsobem zabezpečit. Z toho důvodu je server vytvořen pomocí modulů **express**, **https** a **websocket** spolu s klíči a certifikáty potřebnými pro zabezpečené spojení, viz řádek 1-5 výpisu 5.1. Po vytvoření serveru pak server naslouchá na portu 443, viz řádek 7 výpisu 5.1.

Výpis 5.1: Vytvoření serveru

```
1 const server = https.createServer({
2   cert: fs.readFileSync('certifikat'),
3   ca: fs.readFileSync('certifikacniautorita'),
4   key: fs.readFileSync('klíč')}, app, (req, res) => {
5   res.statusCode = 200;});
6
7 server.listen(port)
```

Na obrázku 5.1 lze vidět, že server má platný certifikát a komunikace je šifrovaná pomocí **TLS 1.3**.



Obr. 5.1: Zabezpečení serveru

5.1.2 Přeposílání zpráv

Po připojení mobilních zařízení čeká server na událost příchozích zpráv. To zde představují APDU příkazy z terminálu a APDU odpovědi z karty. Zachytí-li server tuto událost, zprávu přeposílá na druhé zařízení ve formě UTF-8 dat, která si přijímací zařízení samo zpracuje. Na výpisu 5.2 představuje „element“ zařízení, kterému se zasílá zpráva a funkce `sendUTF()` zařizuje zaslání zprávy ve formátu UTF-8.

Výpis 5.2: Zasílání zpráv

```
1 element.sendUTF(mes.utf8Data)
```

5.1.3 Logy

Logování zpráv, které prošly serverem, je uskutečněno za pomoci modulu **winston**. Data k ukládání jsou formátována tak, aby se uložil čas příchozích zpráv, události, jako například připojení a odpojení zařízení, a jaké zařízení zaslalo jaká data, viz řádek 1-2 výpisu 5.3. Data se ukládají do jednoho souboru, viz výpis 5.3 řádek 4.

Výpis 5.3: Logování pomocí modulu winston

```
1 winston.format.printf(info => {
2 return `${info.timestamp} ${info.level}: ${info.message}`;
3 })
4 new winston.transports.File({filename: 'private/app.log'})
```

5.2 Webová stránka

Pro účel uživatelského přehledu jsou zde vytvořeny 4 panely. Prvním panelem je jednoduché přivítání uživatele s možností odhlášení z webové stránky. V druhém panelu lze v reálném čase vidět připojená zařízení. Ve třetím panelu se v reálném čase zobrazují posílané zprávy přes komunikační kanál. A v posledním čtvrtém panelu je možnost prohlížet minulou činnost dle zadaného data.

Volný přístup k aktivní komunikaci a logům je přímým porušením bezpečnosti citlivých dat a proto je nutné zabezpečit i přístup k webové stránce. Celkem má tedy webová stránka 2 stránky a to přihlašovací a hlavní. Přihlašovací stránka je definována v souboru **login.ejs** a hlavní stránka je definována v souboru **index.ejs**.

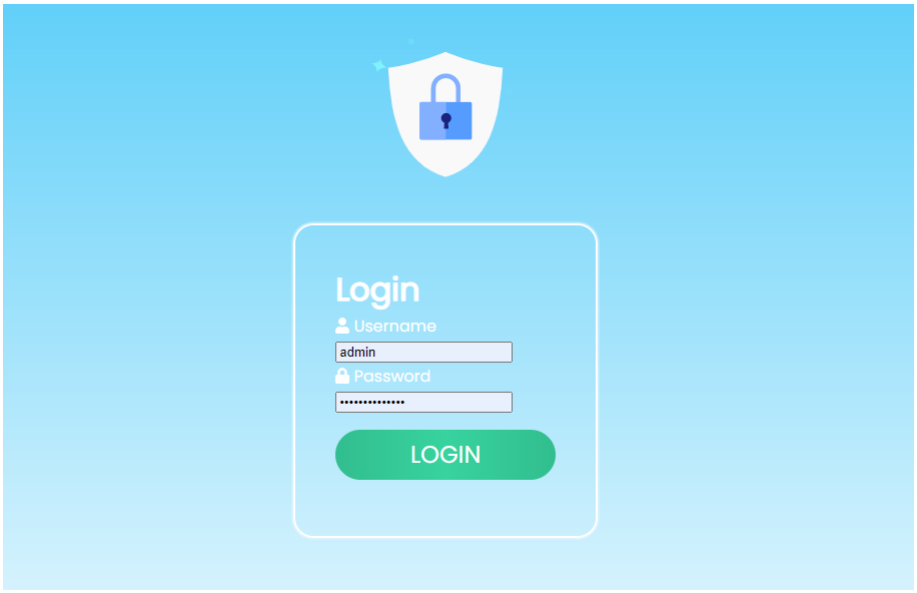
5.2.1 Přístup do webové stránky

Praktický způsob kontroly přístupu je například přihlášení přes uživatelské jméno a heslo. NodeJS nabízí jednoduchý modul pro registraci a přihlašování uživatelů

k webové stránce. Modul se nazývá **passport** a jeho logika je použita v souboru **server.js** a **passport-config.js**. Kód byl inspirován vzdělávacím projektem od Web-DevSimplified [49].

Prvním krokem k vytvoření přihlašovacího systému je registrace. Důležitým aspektem je správné uložení dat uživatele a zejména jeho hesla. A proto je zde použitá funkce `bcrypt.hash()`, která zadané heslo se solí zhashuje. Přihlašovací údaje se pro jednoduchost neukládají do databáze ale do veřejně nepřístupného souboru. Pro účely testování je předem vytvořen administrátorský uživatel a další registrace není aktivní.

Samotné přihlašování se děje klasicky v přihlašovacím okně, které lze vidět na obrázku 5.2.



Obr. 5.2: Přihlašovací okno

Po zadání přihlašovacích údajů a potvrzení formuláře zachytí server tuto událost a pustí se funkce `authenticate()`, která provádí autentizaci. Uložené heslo se porovnává se zadaným pomocí funkce `compare()`. V případě úspěšného přihlášení je uživatel přesměrován na hlavní stránku webu. V opačném případě je přesměrován na přihlašovací stránku s upozorněním na chybně zadaná data. Relace s úspěšným přihlášením je ukládána. Uživatel se tedy nemusí pokaždé přihlašovat, když obnoví stránku.

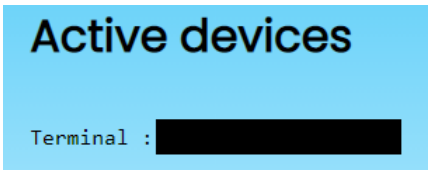
Active Devices

První panel s funkcionalitou je panel **Active Devices**, kde lze vidět připojená zařízení v momentu zobrazení panelu. Při připojení zařízení k serveru se ukládá role

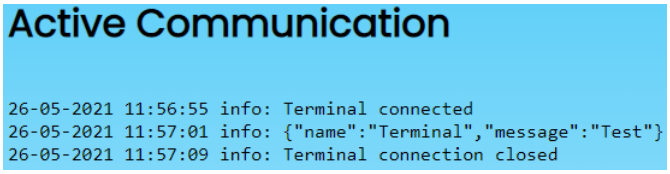
zařízení a její IPv6 adresa do souboru **connections.log**. A naopak při odpojení ze serveru se ze souboru připojení smaže. Funkcionalita pro zobrazení připojených zařízení funguje tak, že se každou sekundu čte soubor connections.log a zobrazují se jeho data na obrazovku, viz obrázek 5.3.

Active Communication

Druhá funkcionalita zobrazující aktivní komunikaci mezi zařízeními je postavená na podobném principu jako předchozí funkcionalita. Po kliknutí na panelu se spustí skript, který nastaví počáteční index řádku pro výpis jako poslední index řádku v souboru **app.log**. Poté se každou sekundu čte soubor a vypisují se řádky s indexem větším než nastavený počáteční index, tedy vypisují se data, která jsou v ten moment posílaná přes komunikační kanál, viz obrázek 5.4.



Obr. 5.3: Aktivní zařízení



Obr. 5.4: Aktivní komunikace

Logs

Poslední funkcionalitou je prohlížení si logu dle vybraného počátečního data a koncového data. Po potvrzení zadání dat se spustí skript, který vezme celý soubor **app.log** a vyfiltruje řádky, které odpovídají zadanému časovému rozmezí, a vypíše je na obrazovku. Na obrázku 5.5 je zobrazen filtr logů dle data.

Obr. 5.5: Filtrování logů

6 Testování systému

Poslední částí práce je testování RR demonstrátoru na testových systémech a na reálně běžících systémech. Pro útok na reálné systémy byl zvolen systém EMV a přístupový systém nejmenované univerzity.

6.1 Příprava zařízení na útok

Z hlediska klonování Mifare Classic karet, jak již bylo zmíněno, je nutné mít **GEN2 přepisovatelnou kartu** a mobilní zařízení s **NFC čipem podporujícím Mifare Classic** technologii. Při útoku přeposláním běží již server online, tudíž je zapotřebí pouze nakonfigurovat mobilní zařízení. Minimální předpoklad je zapnutá a funkční NFC služba a **dostupnost k síti Internet** na obou útočících zařízeních. Pro pokročilejší využívání je nutný **root telefonu**, modul **NFC HCE Catch-All-Routing** a **vypnutá aplikace Google Pay**.

6.1.1 Root telefonu

Postup rootování telefonu je pro většinu zařízení rozdílný. V této práci se rootoval mobil Samsung Galaxy S9+ a Sony Xperia Z1 Compact D5503.

U starších mobilů lze využít aplikace **KingRoot** [50] pro velmi jednoduché rootnutí zařízení. Aplikaci stačí stáhnout, spustit a pár kliknutími získat root. Kompatibilita verzí aplikace není jednotná pro všechny mobily. Pro zařízení Sony Xperia Z1 Compact D5503 se musela použít verze 5.1.2 z roku 2017 a ne nejaktuálnější verze 5.4.0 z roku 2021. U novějších zařízení je postup výrazně složitější.

Postup pro Samsung Galaxy S9+:

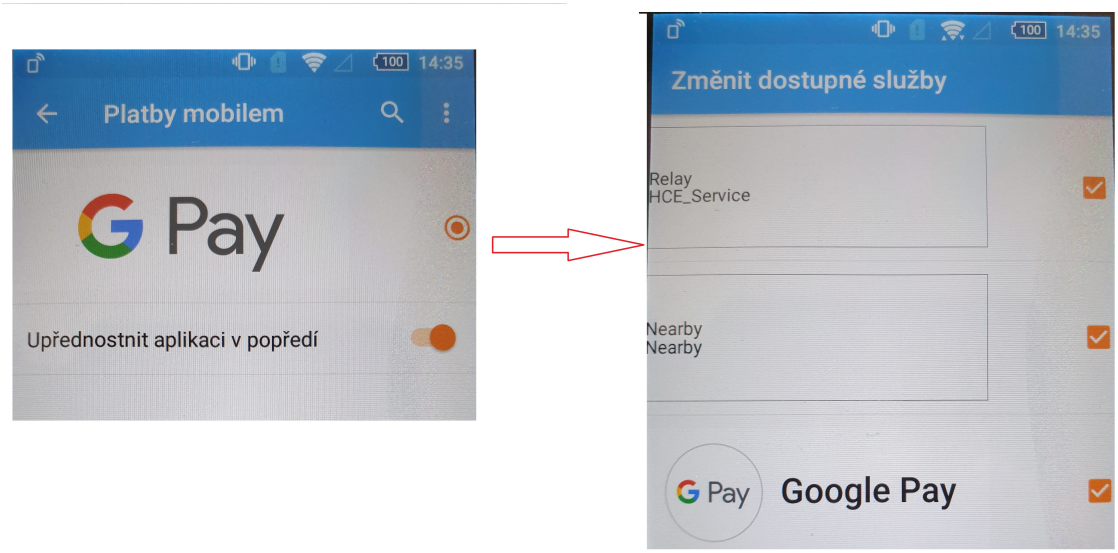
1. **Odemknutí bootloaderu** - Povolit OEM unlocking na mobilu.
2. **Stáhnout aplikaci Odin na počítač** [51] - Slouží k nahrání souborů nutných pro root.
3. **Stáhnout aplikace a soubory potřebné pro root** - Prvním nástrojem je aplikace pro obnovení systémů a instalování souborů **TWRP** (Team Win Recovery Project) [52]. Dále je nutná aplikace **Magisk** [53] pro spravování rootnutého zařízení. A nakonec soubor **Disable_Dm-Verity_ForceEncrypt** [54], který upravuje systémové soubory pro získání root práv.
4. Nahrát TWRP do zařízení pomocí aplikace Odin.
5. Restartovat zařízení, přejít do TWRP a smazat paměť telefonu.
6. Nahrát zbylé soubory přes USB kabel.
7. Přejít do TWRP a nainstalovat zbytek souborů.
8. Restartovat zařízení.

6.1.2 Instalace modulu NFC HCE Catch-All-Routing

Pro instalaci modulu je nutné mít Xposed framework přítomný v zařízení. Na zařízeních s aplikací Magisk je možné vyhledat a nainstalovat Xposed framework přímo v aplikaci. A poté najít modul v Xposed a nainstalovat. Pro zařízení bez Magisk je nutné stáhnout Xposed framework [55] samostatně. Instalace modulu je poté již stejná.

6.1.3 Deaktivace Google Pay

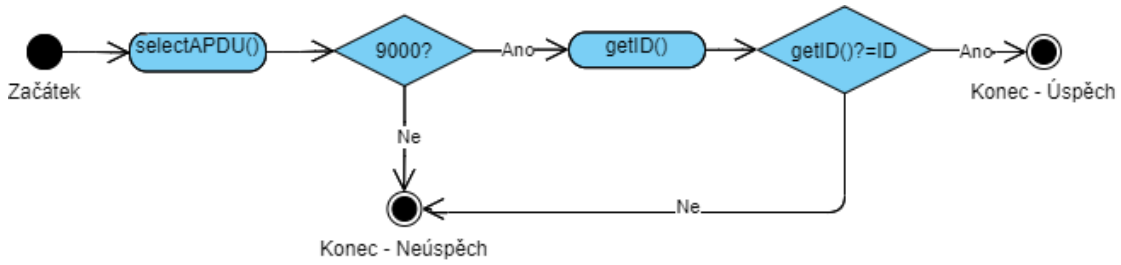
Deaktivace může být na každém zařízení jiná. Na některých zařízeních stačí aplikaci odinstalovat, na některých je nutné deaktivovat Google Pay v NFC nastavení zařízení a na některých zařízeních, například Samsung Galaxy S9+, je zapotřebí deaktivovat všechny Google služby a až tím se deaktivuje Google Pay. Pro Sony Xperia Z1 Compact D5503 stačí v nastavení vybrat možnost „Upřednostnit aplikaci v popředí“ a nebo vybrat službu HCE_Service jako primární službu pro platby, viz obrázek 6.1.



Obr. 6.1: Sony Xperia Z1 Compact D5503 - Nastavení Google Pay

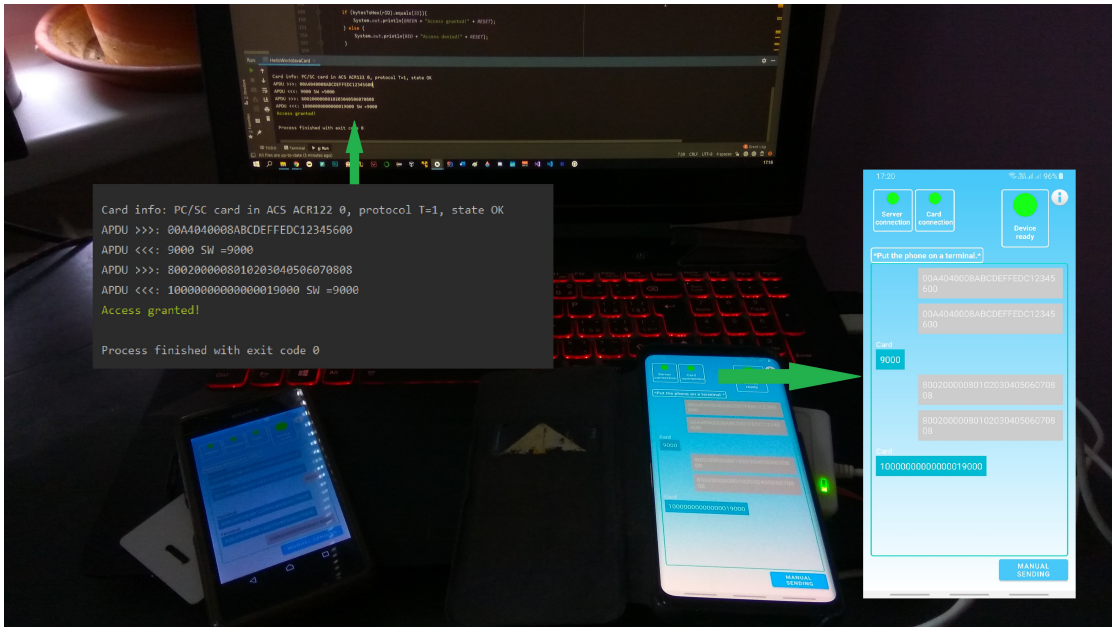
6.2 Útok přeposláním na testovací autentizační systém

Jako testovací autentizační systém zde poslouží jednoduchý autentizační systém, který vybírá applet a pomocí funkce `getID()` zasílá APDU příkaz pro získání ID z karty a tyto data porovnává s uloženým ID. Na obrázku 6.2 je zobrazen diagram tohoto systému.



Obr. 6.2: Autentizační systém - Diagram

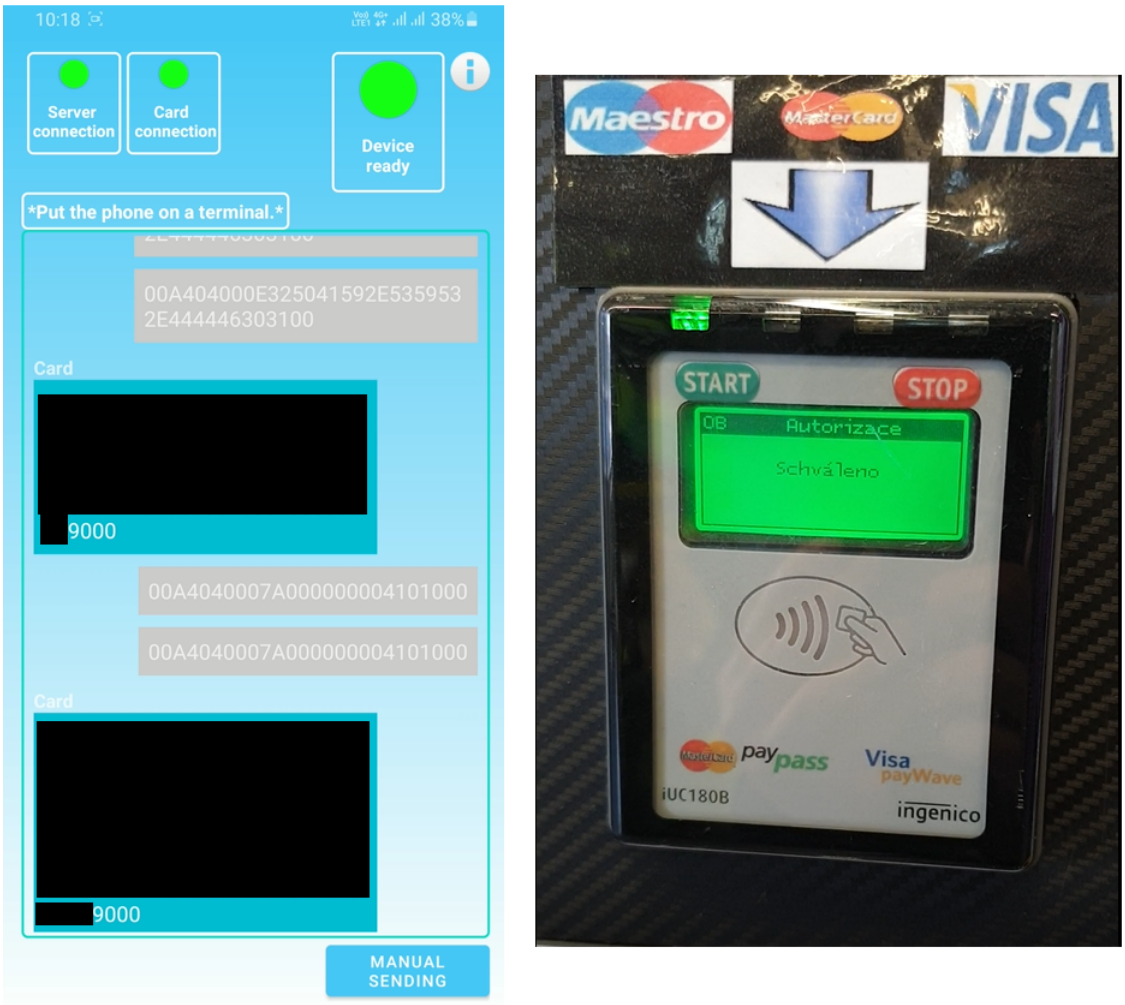
Na obrázku 6.3 je vyfocena testovací stanice. Prvním útočníkem je mobilní telefon Sony Xperia Z1 Compact D5503, který je v roli „Card“. Druhým útočníkem je rootnutý mobilní telefon Samsung Galaxy S9+ s modulem NFC HCE Catch-All-Routing, který je v roli „Terminal“. První krok útoku je přiložení mobilu Sony ke kartě. Druhým krokem je přiložení mobilu Samsung na aktivní terminál a v moment, kdy terminál zachytí jeho přítomnost, začne do něho posílat dotazy APDU a začíná se přeposílat komunikace. Přeposílání zpráv lze vidět na části obrázku se snímkem obrazovky mobilu. Terminál po útoku zobrazuje „Access granted!“, což znamená, že útok na testovací autentizační systém proběhl úspěšně.



Obr. 6.3: Útok přeposláním - testovací autentizační systém

6.3 Útok přeposláním na EMV

Útok na EMV je v podstatě úplně stejný jako předchozí útok na autentizační systém. Jediným rozdílem je nutnost vypnout aplikaci Google Pay pro zařízení, které je přiloženo k terminálu. Testování bylo provedeno na automatu na kávu. Na obrázku 6.4 vlevo lze vidět obrazovku zařízení v roli „Terminal“ a její komponenty. Indikátory jsou všechny zelené, vše je tedy v pořádku a je možné provést útok. V textovém poli lze vidět dvě prvotní zprávy z terminálu, a to PPSE a select APDU, na které bylo úspěšně (kód 9000) odpovězeno ze zařízení v roli „Card“ přiložené k platební kartě. Po předání všech zpráv terminál schválil transakci, viz obrázek 6.4 vpravo.



Obr. 6.4: Útok přeposláním - EMV

Při útoku byly použity mobilní telefony Samsung Galaxy S9+ v roli „Terminal“ a Xiaomi Redmi Note 8 Pro bez rootu v roli „Card“. Celkem se vyměnilo 14 zpráv, a to 7 příkazů a 7 odpovědí, což odpovídá počtu zpráv z teoretické části. Útok trval 4 sekundy.

6.4 Klonování Mifare Classic

K testování klonování byla zvolena nejmenovaná univerzita používající autentizaci pomocí Mifare Classic karet. Prvně byla vyzkoušena autentizace klonováním karty z jiné univerzity na GEN2 kartu. Jak je možno vidět z obrázku 6.5 vlevo, autentizace byla neúspěšná. Poté byla do GEN2 karty naklonována karta z útočené univerzity a autentizace byla úspěšná, viz obrázek 6.5 vpravo.

Obr. 6.5: Klonování Mifare Classic - přístupový systém

K útoku byl použit mobilní telefon Samsung Galaxy S9+. Útok byl úspěšný z toho důvodu, že autentizační systém autentizuje jen pomocí UID na kartě bez žádného přidaného bezpečnostního prvku. K tomu jsou všechny sektory karet na nejmenované univerzitě zabezpečené pomocí továrních klíčů. Přečtení dat sektoru 0 je tedy velmi jednoduché. I v případě, že by byly sektory lépe zabezpečené a nebyly známy klíče, bylo by možné i bez autentizace sektoru 0 získat UID karty například pomocí nativní funkce `getID()` ze třídy **Tag**. To lze z toho důvodu, že UID zapsané v sektoru 0 odpovídá UID karty, které lze jednoduše přečíst.

Závěr

Absolutní většina nalezených dobře proveditelných útoků na současné kartové autentizační systémy se děje na technologiích, které jsou již dávno prolomené a známé svými chybami. Příkladem jsou přístupové systémy. Oproti takovým zranitelným technologiím existují mnohem bezpečnější modernější alternativy, ale i přes tuto skutečnost mnohé instituty stále používají zastaralé a lehce zneužitelné technologie. V dnešní době není pravda, že neexistují bezpečné způsoby kartových autentizačních systémů, ale spíše lidé nemají potřebu zvyšovat bezpečnost u takových systémů. Tímto ale vznikají skvělé příležitosti testovat i starší technologie v praxi.

Výstupem práce je RR demonstrátor běžící na systémech Android, který implementuje nástroje potřebné k (R)elay (přeposlání) a (R)eplay (klonování) útokům. Stěžejní částí pro útok přeposláním bylo vytvoření komunikačního kanálu pro přeposílání zpráv. Důležitými prvky pro tento kanál jsou rychlost, zabezpečení, praktická dostupnost a dobrá kompatibilita s operačním systémem Android. Z tohoto důvodu byl vytvořen server využívající technologii WebSocket. Pro účel co nejnižšího zpoždění zpráv byl z dostupných hostovacích služeb vybrán hosting server v Nizozemí. Při zkoumání technologií čipových karet bylo zjištěno, že je nutné znát AID aplikace na kartě, aby bylo možné ji emulovat. Tento problém byl vyřešen pomocí instalace modulu NFC HCE Catch-All-Routing, který zařizuje, aby mobilní telefon mohl emulovat jakákoliv AID. K tomuto bylo nutno provést rovněž root mobilního telefonu. Z důvodu omezenosti funkce HCE bylo možné vytvořit funkčnost útoku přeposláním jen pro systémy používající APDU zprávy. Z hlediska klonování karet nebo útoku přehráním je výběr systému pro útok velmi malý, protože se v dnešní době používají z velké části bezpečnostní protokoly založené na kryptografii, které nelze lehce emulovat ani kopírovat. Naštěstí nebo bohužel, záleží na jaké straně člověk stojí, se hojně používají, již dávno prolomené, karty Mifare Classic. Tyto karty lze tedy aplikací klonovat. Je ale zapotřebí speciální GEN2 přepisovatelná karta.

Pro lepší uživatelský přehled o útocích přeposláním byla k serveru vytvořená webová stránka, kde uživatel může sledovat aktivní zařízení, komunikaci probíhající v reálném čase a výpis logů komunikace dle zadaného data. Webová stránka je samozřejmě zabezpečena přístupovým systémem z důvodu přítomnosti citlivých informací z karet, které se dají vyčíst z logů.

Při testování aplikace byl úspěšně vykonán útok přeposláním nejen na jednoduchý testovací systém ale také na terminálu používajícím protokol EMV, tedy platební systémy. Klonování karet Mifare Classic bylo testováno na kartách ISIC, které jsou často používané v systému autentizace na univerzitách. Naklonovaná karta z originální karty dokázala na nejmenované univerzitě úspěšně otevřít přístupové dveře.

RR demonstrátor by bylo možno rozšířit o útok přehráním pomocí již vytvoře-

ného útoku přeposláním, tedy přehrát uskutečněnou konverzaci zpátky terminálu v pozdější čas. Tento útok by ale fungoval jen na jednoduché autentizační protokoly bez zabezpečovacích prvků. K funkčnosti NFC je nutná velmi krátká vzdálenost, proto by mohlo být dobrým rozšířením práce tuto vzdálenost nějakým způsobem zvýšit. Dalším možným rozšířením by byla emulace specifického UID na mobilním telefonu. Telefon by mohl poté sloužit například místo Mifare Classic karet. V případě webové stránky lze přidat mnoho funkcí, jako například stahování logů, hezčí grafické rozhraní a výpisy a nebo databázi pro registraci uživatelů.

Literatura

[1] Thales Group. *Smart card basics – A short guide (2020).*[online]. poslední aktualizace 23. 8. 2020 [cit. 25. 11. 2020]. Dostupné z URL: <https://www.thalesgroup.com/en/markets/digital-identity-and-security/technology/smart-cards-basics>.

[2] RANKL, W. a W. EFFING. *Smart card handbook.* Chichester: Wiley, 2003. ISBN 0-470-85668-8.

[3] ISO/IEC. *ISO/IEC 7810:2019.* Prosinec, 2019. Dostupné z URL: <https://www.iso.org/standard/70483.html>.

[4] Ernst Haselsteiner, Klemens Breitfuß. Workshop on RFID security. *Security in Near Field Communication (NFC).*[online]. Austria. 2006. Dostupné z URL: <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=EF31F57598A8640A9EEEC16EF76303DA?doi=10.1.1.475.3812&rep=rep1&type=pdf>.

[5] HID Global Corporation. *HID Proximity.*[online]. poslední aktualizace 01. 07. 2012 [cit. 10. 12. 2020]. Dostupné z URL: <https://www.hidglobal.com/sites/default/files/resource_files/hid-prox-br-en.pdf>.

[6] ISO/IEC. *ISO/IEC 14443-2:2020.* Červenec, 2020. Dostupné z URL: <https://www.iso.org/standard/73597.html>.

[7] C. Enrique Ortiz. *An Introduction to Java Card Technology.*[online]. poslední aktualizace 29. 05. 2003 [cit. 10. 12. 2020]. Dostupné z URL: <https://www.oracle.com/java/technologies/java-card/javacard1.html>.

[8] ISO/IEC. *ISO/IEC 7816-5.* poslední aktualizace 01. 01. 2018 [cit. 10. 12. 2020]. Dostupné z URL: <https://www.iso.org/standard/34259.html>.

[9] Guillaume Barbu, Hugues Thiebeauld,Vincent Guerin. Oberthur Technologies, Telecom ParisTech, Dep. ComElec, Groupe SEN. *Attacks on Java Card 3.0 Combining Fault and Logical Attacks.*[online]. Francie. 2010. Dostupné z URL: <https://www.researchgate.net/publication/220962810_Attacks_on_Java_Card_30_Combining_Fault_and_Logical_Attacks>.

[10] ISO/IEC. *ISO/IEC 7816-4:2020.* poslední aktualizace 01. 05. 2020 [cit. 10. 12. 2020]. Dostupné z URL: <https://www.iso.org/standard/77180.html>.

[11] Jordi van den Breekel. KPMG. *Relaying EMV Contactless Transactions using Off-The-Shelf Android Devices.*[online]. Netherlands Březen.2015. Dostupné z URL: <https://www.blackhat.com/docs/asia-15/materials/asia-15-VandenBreekel-Relaying-EMV-Contactless-Transactions-Using-Off-The-Shelf-Android-Devices-wp.pdf>.

[12] NXP®. *Why MIFARE?.*[online]. poslední aktualizace 2021 [cit. 8. 5. 2021]. Dostupné z URL: <https://www.mifare.net/en/>.

[13] LIESKOVAN, T. *Bezpečnost autentizačních systémů založených na kartách Mifare Classic a ověřování pomocí UID.* [online]. 2019, roč. 21, č. 1, s. 16-20. ISSN:1213-1539. <http://www.elektrorevue.cz/cz/clanky/kybernetika--automatizace--merici-technika/0/bezpecnost-autentizacnich-systemu-zalozenych-na-kartach-mifare-classic-a-overovani-pomoci-uid--security-of-authentication-systems-based-on-mifare-classic-and-uid-verification-/>.

[14] NXP®. *MIFARE Classic EV1 4K - Mainstream contactless smart card IC for fast and easy solution development.*[online]. poslední aktualizace 23. 11. 2017 [cit. 9. 5. 2021]. Dostupné z URL: <https://www.nxp.com/docs/en/data-sheet/MF1S70YYX_V1.pdf>.

[15] NXP®. *NXP® MIFARE Classic® EV1.*[online]. poslední aktualizace 8. 2018 [cit. 9. 5. 2021]. Dostupné z URL: <https://www.mifare.net/wp-content/uploads/2018/10/MIFARE-Classic-EV1-Leaflet-web-102018.pdf>.

[16] NXP®. *MIFARE DESFire EV3 contactless multi-application IC.*[online]. poslední aktualizace 15. 5. 2020 [cit. 10. 5. 2021]. Dostupné z URL: <https://www.nxp.com/docs/en/data-sheet/MF3DHx3_SDS.pdf>.

[17] NXP®. *MIFARE Ultralight EV1 - Contactless ticket IC.*[online]. poslední aktualizace 9. 4. 2019 [cit. 11. 5. 2021]. Dostupné z URL: <https://www.nxp.com/docs/en/data-sheet/MF0ULX1.pdf>.

[18] NXP®. *MIFARE Plus® EV2.*[online]. poslední aktualizace 9. 4. 2019 [cit. 11. 5. 2021]. Dostupné z URL:

<https://www.mifare.net/wp-content/uploads/2020/06/NXP-MIFARE-Plus-EV2-product-fact-sheet-web.pdf>.

[19] Jerry Hildenbrand. *Gingerbread feature: Near Field Communication.*[online]. poslední aktualizace 21. 12. 2010 [cit. 04. 12. 2020]. Dostupné z URL: <https://www.androidcentral.com/gingerbread-feature-near-field-communication>.

[20] Marek Houser. *NFC platby z Xiaomi Mi Band 4 míří na západ do Evropy. Kde teď fungují?.*[online]. poslední aktualizace 27. 09. 2020 [cit. 04. 12. 2020]. Dostupné z URL: <https://www.svetandroida.cz/nfc-platby-mi-band-4-na-ukrajine-v-belorusku/>.

[21] ISO/IEC. *ISO/IEC 18092:2013.* poslední aktualizace 01. 03. 2013 [cit. 10. 12. 2020]. Dostupné z URL: <https://www.iso.org/standard/56692.html>.

[22] Google Developers. *Near field communication overview.*[online]. poslední aktualizace 27. 12. 2019 [cit. 23. 10. 2020]. Dostupné z URL: <https://developer.android.com/guide/topics/connectivity/nfc>.

[23] Google Developers. *NFC basics.*[online]. poslední aktualizace 7. 8. 2020 [cit. 7. 12. 2020]. Dostupné z URL: <https://developer.android.com/guide/topics/connectivity/nfc/nfc?authuser=3>.

[24] Google Developers. *ReaderCallback.*[online]. poslední aktualizace 4. 8. 2020 [cit. 7. 12. 2020]. Dostupné z URL: <https://developer.android.com/reference/kotlin/android/nfc/NfcAdapter.ReaderCallback>.

[25] Google Developers. *Host-based card emulation overview.*[online]. poslední aktualizace 27. 12. 2019 [cit. 7. 12. 2020]. Dostupné z URL: <https://developer.android.com/guide/topics/connectivity/nfc/hce?authuser=3>.

[26] Sławomir Jasek. *2018 practical guide to hacking NFC/RFID* .[online]. Confidence, Kraków. 4.06.2018 Dostupné z URL: <https://www.slideshare.net/wojdwo/a-2018-practical-guide-to-hacking-rfidnfc>.

[27] Michael Irauschek. *Java Card Attacks and Countermeasures against Malicious Applications in a User Centric Ownership Model.* [online]. Graz, Únor, 2013. Diplomová práce. University of Technology, Institute for Technical Informatics. Dostupné z URL: <https://diglib.tugraz.at/download.php?id=576a73fe652a8&location=browse>.

[28] Andrew Calafato, Information Security Group, Royal Holloway, University of London. *An analysis of the vulnerabilities introduced with Java Card 3 Connected Edition.*[online]. United Kingdom. 1. 5. 2013. Dostupné z URL: <https://www.ma.rhul.ac.uk/static/techrep/2013/MA-2013-04.pdf>.

[29] Sergei Volokitin. *Good, Bad and Ugly Design of Java Card Security.* Nijmegen, 2016. Diplomová práce. Radboud University, Software Science Faculty of Science, Computing Science.

[30] Takeshi Fujino1a, Takaya Kubota a Mitsuru Shiozaki. Department of Science and Engineering, Ritsumeikan University. Research Organization of Science and Engineering, Ritsumeikan University. Fig. 2. SPA (simple power analysis) attack exploiting a power traceduring modular exponentiation in RSA algorithm. [foto] *Tamper-resistantcryptographic hardware.*[online]. Japan. 01. 01. 2017. Dostupné z URL: <https://www.researchgate.net/publication/312873398_Tamper-resistant_cryptographic_hardware>.

[31] Jip Hogenboom, Wojciech Mostowski. Department of Computing Science, Radboud University Nijmegen. *Full Memory Read Attack on a Java Card.*[online]. Netherlands. 2009. Dostupné z URL: <https://www.researchgate.net/publication/228788902_Full_Memory_Read_Attack_on_a_Java_Card>.

[32] José Vila, Ricardo J. Rodríguez. Department of Computer Science and Systems Engineering, Research Institute of Applied Sciences in Cybersecurity University of León. University of Zaragoza. *Practical Experiences on NFC Relay Attacks with Android Virtual Pickpocketing Revisited.*[online]. Spain. 2015. Dostupné z URL: <https://vwzq.net/papers/rfidsec15-practical-nfc-relay.pdf>.

[33] Lishoy Francis, Gerhard Hancke, Keith Mayes, Konstantinos Markantonakis. Information Security Group, Smart Card Centre. Royal Holloway University of London. *Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones.*[online]. United Kingdom. 2012. Dostupné z URL:

<https://www.it.iitb.ac.in/frg/wiki/images/archive/3/3b/20120904124916!Arptit_paper05.pdf>.

[34] Peter Fillmore. *Crash and Pay: Owning and Cloning Payment Devices.*[online]. Germany. 2015. Dostupné z URL: <https://www.blackhat.com/docs/us-15/materials/us-15-Fillmore-Crash-Pay-How-To-Own-And-Clone-Contactless-Payment-Devices.pdf>.

[35] Dennis Giese, Kevin Liu, Michael Sun, Tahin Syed, Linda Zhang. *Security Analysis of Near-Field Communication (NFC) Payments.*[online]. 16.5.2018. Dostupné z URL: <https://courses.csail.mit.edu/6.857/2018/project/Giese-Liu-Sun-Syed-Zhang-NFC.pdf>.

[36] Michael Roland, Josef Langer. NFC Research Lab Hagenberg, University of Applied Sciences Upper. *Cloning Credit Cards: A Combined Pre-play and Downgrade Attack on EMV Contactless.*[online]. Austria. 2013. Dostupné z URL: <https://www.usenix.org/conference/woot13/workshop-program/presentation/roland>.

[37] Timo Kasper, Ingo von Maurich, David Oswald, Christof Paar. Horst Görtz Institute for IT Security, Ruhr-University Bochum. *Cloning Cryptographic RFID Cards for 25$.*[online]. Germany. 2010. Dostupné z URL: <http://proxmark.nl/files/Documents/13.56%20MHz%20-%20MIFARE%20DESFire/Cloning_Cryptographic_RFID_Cards_for_25USD-WISSEC_2010.pdf>.

[38] Peter Price. *Oyster card hack details revealed.*[online]. poslední aktualizace 6. 10. 2008 [cit. 29. 11. 2020]. Dostupné z URL: <http://news.bbc.co.uk/2/hi/programmes/click_online/7655292.stm>.

[39] Rory Flynn. School of Computer Science and Statistics, Trinity College. *An investigation of possible attacks on the MIFARE DESFire EV1 smartcard used in public transportation.*[online]. Dublin. 24.7.2019. Dostupné z URL: <http://www.proxmark.org/files/Documents/13.56%20MHz%20-%20MIFARE%20DESFire/FYP_Report_DESFireEV1.pdf>.

[40] Google Developers. *IsoDep.*[online]. poslední aktualizace 24. 2. 2021 [cit. 11. 5. 2021]. Dostupné z URL:

<https://developer.android.com/reference/kotlin/android/nfc/tech/IsoDep>.

[41] InoApp LLC. *NFC Smart Card Info.*[online]. poslední aktualizace 6. 10. 2015 [cit. 12. 5. 2021]. Dostupné z URL: <https://play.google.com/store/apps/details?id=com.inoapp.cardinfo>.

[42] Julien MILLAU. *Credit Card Reader NFC (EMV).*[online]. poslední aktualizace 18. 3. 2021 [cit. 12. 5. 2021]. Dostupné z URL: <https://play.google.com/store/apps/details?id=com.github.devnied.emvnfccard>.

[43] IKARUS Projects. *MIFARE Classic Tool.*[online]. poslední aktualizace 2. 12. 2020 [cit. 12. 5. 2021]. Dostupné z URL: <https://play.google.com/store/apps/details?id=de.syss.MifareClassicTool>.

[44] yuanwofei. *NFC Card Emulator Pro (Root).*[online]. poslední aktualizace 21. 3. 2021 [cit. 12. 5. 2021]. Dostupné z URL: <https://play.google.com/store/apps/details?id=de.syss.MifareClassicTool>.

[45] Johannes Zweng. *Xposed Module NFC HCE Catch-All-Routing.*[online]. poslední aktualizace 11. 8. 2018 [cit. 13. 5. 2021]. Dostupné z URL: <https://github.com/johnzweng/XposedModifyAidRouting>.

[46] Hey! Let's Code. *Chat-App-In-Android-And-NodeJS-Using-WebSockets.*[online]. poslední aktualizace 4. 5. 2020 [cit. 16. 5. 2021]. Dostupné z URL: <https://github.com/heyletscode/Chat-App-In-Android-And-NodeJS-Using-WebSockets>.

[47] Design Barn Inc. *LottieFiles.*[online]. poslední aktualizace 2021 [cit. 17. 5. 2021]. Dostupné z URL: <https://lottiefiles.com/>.

[48] Saurabh. *WebSocket vs HTTP Calls - Performance Study.*[online]. poslední aktualizace 13. 4. 2019 [cit. 18. 5. 2021]. Dostupné z URL: <https://browsee.io/blog/websocket-vs-http-calls-performance-study/>.

[49] WebDevSimplified. *Nodejs-Passport-Login.*[online]. poslední aktualizace 10. 7. 2019 [cit. 18. 5. 2021]. Dostupné z URL: <https://github.com/WebDevSimplified/Nodejs-Passport-Login>.

[50] KINGROOT STUDIO. *KingRoot.*[online]. poslední aktualizace 13. 1. 2021 [cit. 26. 5. 2021]. Dostupné z URL: <https://kingroot.en.uptodown.com/android/versions>.

[51] Odindownload.com. *Odin 3.13.1.*[online]. poslední aktualizace 2017 [cit. 26. 5. 2021]. Dostupné z URL: <https://odindownload.com/>.

[52] Team Win LLC. *Samsung - TWRP.*[online]. poslední aktualizace 2021 [cit. 26. 5. 2021]. Dostupné z URL: <https://twrp.me/Devices/Samsung/>.

[53] Magisk Manager. *Magisk Manager.*[online]. poslední aktualizace 2021 [cit. 26. 5. 2021]. Dostupné z URL: <https://magiskmanager.com/>.

[54] Zackptg5. *Disable_Dm-Verity_ForceEncrypt_12.16.2018.zip.*[online]. poslední aktualizace 16. 12. 2018 [cit. 26. 5. 2021]. Dostupné z URL: <https://androidfilehost.com/?fid=11410963190603877244>.

[55] Xposed. *Xposed Module Repository.*[online]. poslední aktualizace 2021 [cit. 26. 5. 2021]. Dostupné z URL: <https://repo.xposed.info/>.

Seznam symbolů, veličin a zkratek

ISO	mezinárodní organizace pro standardizaci – International Organization for Standardization
RFID	identifikace radiovými frekvencemi – Radio Frequency Identification
EEPROM	elektricky mazatelná programovatelná paměť jen pro čtení – Electrically Erasable Programmable Read-only Memory
ROM	paměť jen pro čtení – Read-Only Memory
CPU	centrální procesorová jednotka – central processing unit
RAM	paměť s náhodným přístupem – Random-access memory
AID	identifikátor aplikace – Application Identifier
JVM	virtuální přístroj Java – Java Virtual Machine
CAP	konvertovaný applet – Converted Applet
JCVM	virtuální přístroj Java karty – Java Card Virtual Machine
APDU	aplikační protokol datové jednotky – Application Protocol Data Unit
CAD	přístroj akceptující kartu – Card Acceptance Device
PPSE	blízké prostředí platebního systému – Proximity Payment System Environment
FCI	informace o kontrole souborů – File Controll Information
PDOL	zpracování dat seznamu možností – Processing Data Options List
AIP	profil výměny aplikací – Application Interchange Profile
AFL	vyhledávač aplikačních souborů – Application File Locator
UID	unikátní identifikátor – Unique Identifier
API	rozhraní pro programování aplikací – Application Programming Interface
OS	operační systém – Operating System
NFC	komunikace v blízkém poli – Near Field Communication

HCE emulace hostitelské karty – Host-based Card Emulation

DOS odepření služby – Denial Of Service

TWRP Team Win projekt obnovení – Team Win Recovery Project