Mobilní aplikace pro správu dat na NFC kartách

Bc. Robert Jedek

Diplomová práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

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

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Robert Jedek**
Osobní číslo:	**A17284**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**

Téma práce:	**Mobilní aplikace pro správu dat na NFC kartách**
Téma anglicky:	**A Mobile Application for NFC Card Data Management**

Zásady pro vypracování:

1. **Prostudujte knihovny pro implementaci NFC v prostředí OS Android.**
2. **Analyzujte požadavky na aplikaci, která umožní spravovat data uložená na NFC kartě, jako např. konfiguraci WiFi připojení, kontaktní údaje uživatele, certifikáty pro šifrování komunikace atd.**
3. **Specifikujte jednotlivé funkcionality budoucí aplikace a navrhněte její uživatelské rozhraní.**
4. **Aplikaci implementujte dle specifikace.**
5. **Otestujte aplikaci v reálném nasazení.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **COSKUN, Vedat, Kerem OK a Busra OZDENIZCI. Professional NFC application development for Android. Chichester, West Sussex: Wrox, a Wiley brand, [2013]. ISBN 978-1118380093.**
2. **IGOE, Tom, Don COLEMAN a Brian JEPSON. Beginning NFC: near field communication with Arduino, Android, and Phonegap. Beijing: O'Reilly, [2014]. ISBN 14-493-7206-6.**
3. **SUBTIL, Victor. Near Field Communication with Android Cookbook. Birmingham: Packt Publishing, 2014. ISBN 9781783289653.**
4. **MILETTE, Greg a Adam STROUD. Professional Android sensor programming. Hoboken, N.J.: Wiley, 2012. ISBN 978-1-1181-8348-9.**
5. **MEIER, Reto. Professional Android 4e. 4rd ed. Indianapolis, IN: John Wiley, 2018. ISBN 978-1-118-94952-8.**

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

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

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

Ve Zlíně dne 7. prosince 2018

doc. Mgr. Milan Adámek, Ph.D.
děkan

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

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne 29. 4. 2019

Robert Jedlek v. r.

.....

podpis diplomanta

ABSTRAKT

Diplomová práce se zabývá technologií NFC a její podporou v operačním systému Android. Cílem je vytvářet NFC tagy s konfigurací pro připojení k Wi-Fi síti a navázání spojení se serverem. Důvodem tohoto řešení je potřeba urychlit provedení této konfigurace a její snadná změna bez nutnosti zásahu uživatele. Zaměřuje se proto především na aspekty čtení a zápisu NFC tagů, včetně zabezpečení uložených informací šifrováním.

Práce popisuje NDEF formát zapouzdřující data uložená v NFC tagu, včetně tvorby NDEF zpráv a záznamů, které poskytuje platforma Android, a dále seznamuje s použitými algoritmy pro šifrování citlivého obsahu tagu, způsobem ukládání šifrovacích klíčů, včetně způsobu jejich exportu pro potřeby aplikace, která obsah tagu zpracovává.

Klíčová slova: NFC, Android, NDEF, šifrování

ABSTRACT

The thesis is dedicated to NFC technology and especially to its support in Android OS. The main goal of this technology is connecting to Wi-Fi networks seamlessly using NFC tags containing configuration data, which can be used to communicate with the server. The reason for this solution is the need to speed up the creation and modification of this configuration without the need for user intervention. The thesis therefore deals with the aspects of reading and writing of NFC tags, including methods to protect the contained data through encryption.

The thesis describes the NDEF format which encapsulates the NFC tag data, corresponding Android API's for creation of NDEF messages and records. It also describes the encryption algorithms used to protect the sensitive content of NFC tags, as well as the procedures to store and export the encryption keys to be used in the target applications.

Keywords: NFC, Android, NDEF, encryption

Chtěl bych poděkovat svému vedoucímu bakalářské práce Ing. Tomáši Dulíkovi, Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce.

Dále bych chtěl poděkovat Ing. Ondřeji Fibichovi za vstřícnost a pomoc při získání potřebných informací a podkladů.

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

OBSAH

ÚVOD **8**
I TEORETICKÁ ČÁST **10**
1 SYSTÉM TEVOGS **11**
2 TECHNOLOGIE NFC **15**
2.1 TECHNICKÉ SPECIFIKACE 15
2.1.1 ISO / IEC 14443 16
2.1.2 ISO / IEC 18092 17
2.1.3 (JIS) X 6319-4 17
2.2 KOMUNIKAČNÍ REŽIMY 18
2.3 REŽIMY PŘENOSU 18
2.3.1 Čtení/zápis 18
2.3.2 Emulace karty 18
2.3.3 Peer-to-peer 18
2.4 PŘÍBUZNÉ TECHNOLOGIE 19
2.5 BEZPEČNOST 20
3 NFC ANDROID API **21**
3.1 SYSTÉM DISPEČINKU TAGŮ 22
3.2 ZPRACOVÁNÍ NFC ZÁMĚRU 23
3.3 NDEF ZPRÁVA 24
3.4 VYTVOŘENÍ NDEF ZÁZNAMU 26
3.5 POVOLENÍ NFC TECHNOLOGIE V MANIFESTU 27
3.6 FILTROVÁNÍ TAGŮ PODLE OBSAHU 27
3.7 FILTROVÁNÍ TAGŮ PODLE TECHNOLOGIE 28
3.8 ZPRACOVÁNÍ INFORMACÍ OBDRŽENÝCH V NFC ZÁMĚRU 29
3.9 ANDROID APPLICATION RECORD (AAR) 30
II PRAKTICKÁ ČÁST **32**
4 IMPLEMENTACE APLIKACE TVORBY TAGU **33**
4.1 POŽADAVKY NA APLIKACI 33
4.1.1 Nešifrované položky tagu 33
4.1.2 Šifrované položky tagu 34
4.1.3 Způsoby tvorby obsahu 34
4.1.4 Správa použitých kryptografických klíčů 34
4.2 NÁVRH FUNKCIONALIT VYCHÁZEJÍCÍCH Z POŽADAVKŮ NA APLIKACI 35
4.2.1 Návrh posloupnosti operací 35
4.3 NÁVRH FORMÁTU ZAPISOVANÉHO NA NFC TAG 37
4.4 STRUKTURA NDEF ZPRÁVY 38
4.4.1 NdefRecord 1 – nešifrovaný obsah tagu 38
4.4.2 NdefRecord 2 – šifrovaný obsah tagu 39
4.4.3 NdefRecord 3 – Android Application Record (AAR) 39
4.4.4 Zápis NDEF zprávy na tag 40

4.5 GUI, APLIKAČNÍ LOGIKA 42
4.5.1 Správa úložiště klíčů 42
4.5.2 Interní klíčové úložiště 42
4.5.3 Klíčové úložiště pro export 44
4.5.4 Volba způsobu tvorby tagu 45
4.5.5 Obrazovka zápisu na tag 49
5 IMPLEMENTACE KLIENTSKÉ APLIKACE 52
5.1 POŽADAVKY NA APLIKACI 52
5.2 NÁVRH FUNKCIONALIT VYCHÁZEJÍCÍCH Z POŽADAVKŮ 53
5.2.1 Návrh posloupnosti operací 53
5.3 GUI, APLIKAČNÍ LOGIKA 54
5.3.1 Volba režimu aplikace 54
5.3.2 Vložení klíčového úložiště 55
5.3.3 Mobility mód 56
5.3.4 Permanent mód 58
5.3.5 Zpracování obdržených dat 58
6 TESTOVÁNÍ APLIKACÍ 61
6.1 APLIKACE PRO TVORBU TAGŮ 61
6.2 KONFIGURACE KLIENTSKÉ APLIKACE 61
ZÁVĚR 63
SEZNAM POUŽITÉ LITERATURY 64
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 66
SEZNAM OBRÁZKŮ 67
SEZNAM TABULEK 68
SEZNAM PŘÍLOH 69

ÚVOD

NFC technologie, kterou se tato práce zabývá, je díky svým vlastnostem vhodným způsobem bezdrátové komunikace pro rychlý přenos malého objemu dat. Můžeme se s ní setkat stále častěji u mobilních telefonů, proto je vhodné se touto technologií zabývat. Výhodou této technologie je skutečnost, že není nutné komunikaci ručně navazovat, přičemž komunikace je zahájená pouhým přiblížením dvou zařízení.

Cílem této práce je využít NFC technologie pro prvotní konfiguraci mobilní aplikace způsobem, aby byla její změna prováděna automaticky bez nutnosti manuálního přestavování konfiguračních údajů uživatelem.

Aplikace je součástí specifického navigačního systému TE-VOGS, kde plní funkci klientské aplikace informující řidiče nejen o jeho aktuální poloze, ale také o poloze všech vozidel, které se nacházejí v jeho okolí a jsou do systému zahrnuty. Aplikace běží na zařízení Android, které pomocí Wi-Fi komunikuje s jednotkou, umístěnou na střeše automobilu.

Její prvotní konfigurace obsahuje mimo jiné i údaje potřebné pro spojení se zmiňovanou střešní jednotkou obsahující Wi-Fi Access Point, ke kterému má být zařízení po celou dobu běhu aplikace připojeno a permanentně vynucováno. Zařízení je ovšem přenášeno mezi různými vozidly, a proto je nutné tuto konfiguraci pokaždé manuálně nastavit, aby bylo připojeno ke konkrétní střešní jednotce automobilu, ve kterém se právě nachází.

Záměrem firmy TE-VOGS je umístit nezbytné konfigurační údaje do NFC tagu. Změna konfigurace mobilní aplikace by po přemístění do jiného vozidla probíhala pouhým přiložením zařízení k tomuto tagu.

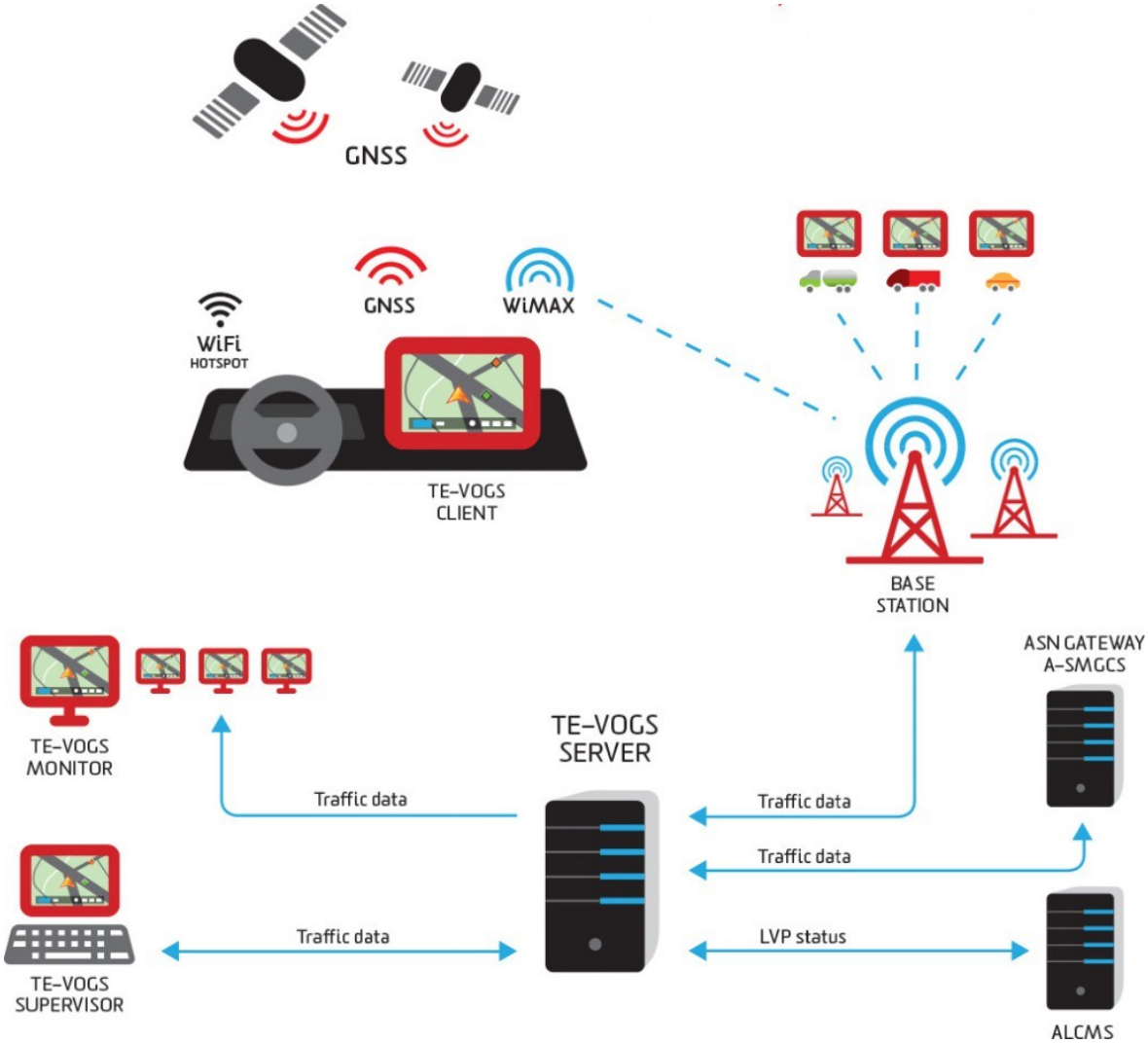
Řešení, které má tato práce poskytnout, je navrhnout a vytvořit rozšíření stávající mobilní aplikace, jež bude NFC tagy načítat. Z takto načtených údajů je provedena prvotní konfigurace a zajištěno permanentní vynucování připojení ke konkrétní Wi-Fi síti. Pro takové řešení je nutné navrhnout a realizovat samostatnou aplikaci pro tvorbu tagů, na které se bude zapisovat konfigurace specifická pro konkrétní vozidlo. Ukládaná konfigurace obsahuje i citlivé údaje, proto je požadováno, aby tyto údaje byly na tag ukládány v šifrované podobě a zároveň byla zajištěna jejich integrita.

Výběr předmětu této diplomové práce spočívá v aktuálnosti tématu, zejména v oblasti NFC technologie, kryptologie nebo platformy Android, a reálná problematika týkající se konkrétní aplikace.

I. TEORETICKÁ ČÁST

1 SYSTÉM TEVOGS

Systém TE-VOGS (*Vehicle Onboard Guidance System*) je řešení pro lokalizaci a navigaci vozidel pro zvýšení bezpečnosti pohybu ve vymezených oblastech. Systém byl původně vyvinutý speciálně pro využití na letištích, ale v současnosti je užíván pro různé aplikace řídící pohyb mobilních jednotek ve vymezeném prostoru. Centrální systém zpracovává pozice jednotlivých vozidel a zobrazuje jejich lokalizaci na mapě. Tato vizualizace je poté využívána dispečinkem a současně je také zobrazována v klientské aplikaci mobilních jednotek, které mají informaci o vlastní pozici, a také o pohybu ostatních mobilních jednotek ve svém okolí [1].



Obrázek 1: Schéma systému [2]

Systém ve spojení s ATC (*Air Traffic Control*) umožňuje kompletní přehled o pohybu na letišti, čímž poskytuje nástroj pro zvýšení bezpečnosti pohybu v jeho prostorech.

Kromě oblasti bezpečnosti systém napomáhá zvyšovat efektivitu řízení pohybu na letišti, protože umožňuje monitorovat pohyb desítek až stovek mobilních jednotek na ploše o velikosti desítek kilometrů. Uvedený systém je schopen predikovat kolize či přetížení provozu a účinně předcházet nežádoucím situacím [1].

TEVOGS Server

TEVOGS Server slouží ke zpracovávání všech obdržených dat vstupujících do systému. Zpracované informace zpětně poskytuje dispečinku a vnitřním mobilním jednotkám [3].

TEVOGS Dispatcher

Management TE-VOGS má různé úrovně autorizace. Autorizace je členěna na základě rolí a zaměření, které v systému vykonávají [3].

Role dispečerů:

- Administrátor
- Hlavní dispečer
- Dispečer profesní skupiny
- Dispečer pro dohled (*view only*)

Profesní skupiny

- Údržba
- Hasiči
- Dodávky paliva
- Ptáci

Obrázek 2: Operátorské centrum [3]

TEVOGS Mobile Client

Klient umožňuje sledovat polohu vozidla, zobrazit aktuální dopravní situaci řidiči a poskytovat další funkce systému. Skládá se ze dvou samostatných částí, jak je naznačeno na obrázku 3.

Obrázek 3: Mobilní klient [3]

První je mobilní jednotka (*roof unit*), která obsahuje přijímač GNSS, modem LTE, Wi-Fi hotspot, antény atd. Je umístěna na střeše pomocí magnetického držáku a pásu. Jedná se o autonomní zařízení, jež poskytuje informace o své poloze, přičemž připojení k síti je realizováno prostřednictvím sítě LTE, která slouží pro distribuci a příjem dat na/ze serveru [3].

Obrázek 4: Vnější mobilní jednotka [5]

Druhým samostatným zařízením mobilní jednotky je přenosné zařízení s operačním systémem Android (tablet, mobilní telefon). V tomto zařízení je nainstalovaná mobilní aplikace komunikující se střešní jednotkou, jejíž součástí je Wi-Fi. Střešní jednotka není nezbytná pro sledování aktuální pozice vozidla, ale poskytuje řidiči zpětnou vazbu prostřednictvím centrálního dispečinku. Dále poskytuje mapu letiště, jeho umístění včetně přehledu aktuální dopravní situace ve sledované oblasti. Umožňuje zasílat řidiči různé informace a varování. Tuto možnost má také řidič, který může posílat zprávy jiným klientům nebo klientským skupinám [3].

Obrázek 5: Vnitřní mobilní jednotka [6]

2 TECHNOLOGIE NFC

NFC technologie slouží k obousměrné komunikaci mezi dvěma elektronickými zařízeními na krátkou vzdálenost. To umožňuje uživatelům rychle přistupovat k informacím z NFC štítků a sdílet tak vizitky, řídit přístupová oprávnění, nebo provádět transakce.

NFC technologie je bezesporu neustále se rozšiřující technologií napříč mnohými obory a lidskými činnostmi. Tuto skutečnost ovlivňuje fakt, že se jedná o ověřené technologie RFID. Expanze spočívá v tom, že NFC technologie byla navržena jako bezpečná forma výměny dat, a také ve variabilitě režimu činností NFC zařízení, které může být jak čtecím zařízením, tak i značkou (tagem). Tato skutečnost umožňuje zařízením NFC komunikovat v režimu *peer-to-peer*.

Další výhodou této technologie je bezesporu nízká spotřeba, což z ní činí ideální technologii pro mobilní telefony, kde je právě nízká energetická náročnost jedním z hlavních kritérií. Nižší energetická náročnost souvisí s krátkou vzdáleností, se kterou mohou NFC zařízení komunikovat. Přestože tato skutečnost může být pro některé typy aplikací nevýhodou, snižuje se tím riziko rušení datového přenosu.

Významnou předností této technologie je její utilitární všestrannost a praktické využití. Perspektivně se jeví tento fakt především s ohledem na výrobce mobilních telefonů, kteří progresivně integrují NFC čipy do nových zařízení. Původní představa o tom, že nám mobilní telefon může nahradit klíče, peněženku či doklady, se v čase ukázala jako problematická s ohledem na možné potíže způsobené při ztrátě nebo krádeži telefonu. Je zřejmé, že v rámci oblasti bezpečnosti je důležitá koordinace všech zúčastněných stran, tedy nejen ze strany výrobců a vývojářů aplikací, ale i uživatele samotného [16].

2.1 Technické specifikace

NFC (*Near field communication*) je bezdrátová technologie určená pro přenos mezi dvěma elektronickými zařízeními na krátkou vzdálenost. Tato vzdálenost je zpravidla do deseti centimetrů. Jedná se o rádiový přenos na frekvenci 13,56 MH s rychlostí od 106 kbit/s do 424 kbit/s v závislostí na použitém standardu.

NFC technologie vychází z technologie RFID (*Radio Frequency Identification*). Technologie jsou popsány standardy definované neziskovou organizací NFC Forum založenou v roce 2004 firmami Nokia, Philips a Sony [7].

Obrázek 6: Logo organizace NFC Forum [8]

2.1.1 ISO / IEC 14443

ISO/IEC 14443, zavedená jako ČSN ISO/IEC 14443, Identifikační karty – Bezkontaktní karty s integrovanými obvody – Karty s vazbou nablízko. Norma se skládá celkem ze čtyř částí:

– Část 1: Fyzikální charakteristiky
– Část 2: Radiofrekvenční výkonové rozhraní a signálové rozhraní
– Část 3: Inicializace a antikolize
– Část 4: Protokol přenosu

Uvedené části tvoří soubor mezinárodních norem spravovaných společně s mezinárodními organizacemi ISO a IEC [9].

Typy NFC štítků jsou dále rozčleněny do následujících čtyř kategorií:

- Typ 1 je založen na standardu ISO/IEC 14443 A, jejich použití je možné v režimu čtení i zápisu včetně možnosti při zápisu štítek uzamknout pouze ke čtení. Dostupná kapacita tagu se pohybuje od 96 bajtů až do 2 kilobajtů, jejich přenosová rychlost je 106 Kb/s. Mezi zástupce patří například Topaz [10].
- Typ 2 je velmi podobný typu 1, má ovšem menší základní kapacitu o velikosti 48 bajtů. Tomuto typu vyhovuje například MIFARE Ultralight [10].
- Typ 3 odpovídá japonskému standardu Sony FeliCa, který poskytuje širokou škálu funkcí v relativně vyšší cenové relaci. Typ 3 umožňuje režim čtení/zápis nebo pouze režim čtení a nabízí různé varianty s kapacitou až do 9 kB. Rychlost je 212 Kb/s nebo 424 Kb/s [10].
- Typ 4 je založen na standardu ISO/IEC 14443 A i B, přičemž tag je obvykle konfigurován již ve výrobě. Režimy jsou čtení/zápis nebo pouze čtení. Kapacita paměti může být až 32 kB a rychlost komunikace je mezi 106 kB/s a 424 kB/s [10].

2.1.2 ISO / IEC 18092

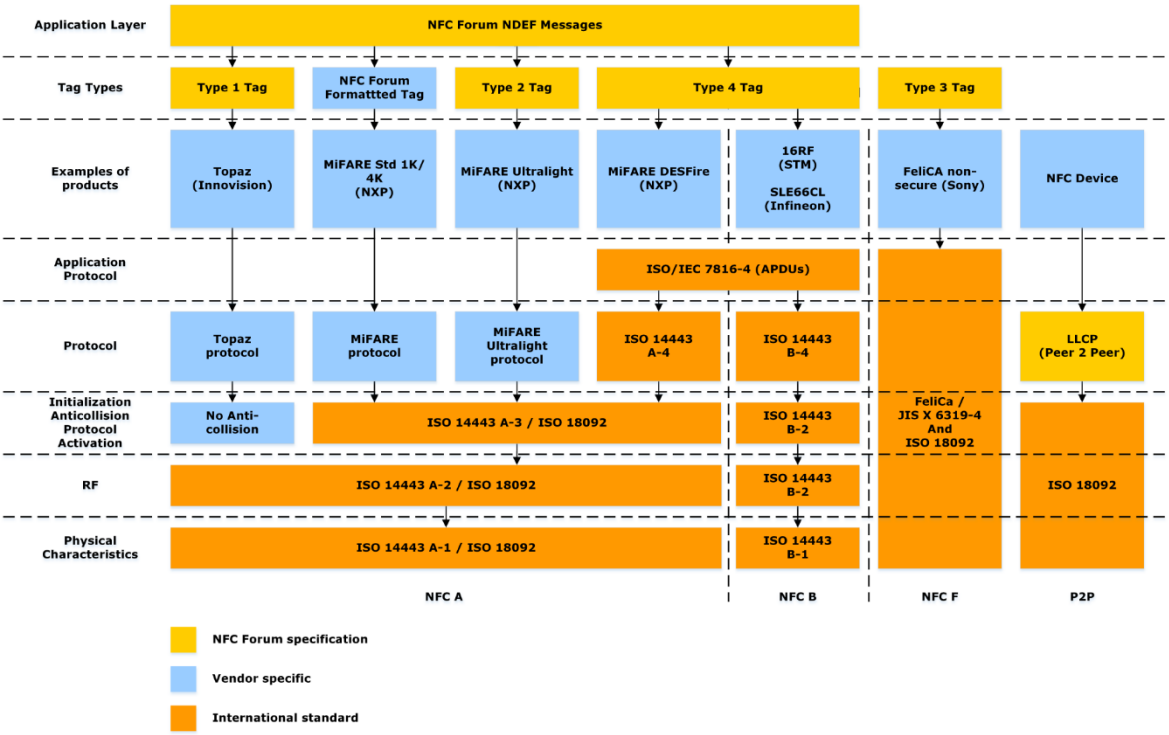
Mezinárodní norma ISO / IEC 18092 specifikuje rozhraní a protokol pro jednoduchou bezdrátovou komunikaci mezi zařízeními na krátkou vzdálenost. Tato zařízení pro bezdrátovou komunikaci (NFC) komunikují s přenosovými rychlostmi 106, 212 a 424 kb/s.

ISO 18092 (NFC) zavádí komunikační režim peer-to-peer pro libovolné výměny binárních dat. Tento režim chybí v normě ISO 14443.

ISO / IEC 18092 vychází z ISO / IEC 14443, ale má zásadní rozdíl. Používá jiný příkazový protokol, který nahrazuje část 4: Protokol přenosu. Obsahuje také dva komunikační režimy aktivní a pasivní umožňující zařízení NFC komunikovat s jinými zařízeními NFC v režimu peer-to-peer [10].

2.1.3 (JIS) X 6319-4

(JIS) X 6319-4 je japonský průmyslový standard známý také jako FeliCa (Felicity Card), jehož základem je ISO18092. Standard byl vytvořen firmou Sony. Standard se používá v různých komerčních platformách nejen pro karty, ale i pro spotřební elektroniku [11].



Obrázek 7: Přehled NFC protokolů [14]

2.2 Komunikační režimy

NFC standard definuje dva typy zařízení známé jako iniciátor komunikace a cíl komunikace. Iniciátorem komunikace je zařízení, které iniciuje komunikaci a řídí výměnu dat. Cílové zařízení, nazývané také jako transpondér, reaguje na požadavky iniciátora [12].

Komunikační režimy NFC zařízení se v základním členění rozlišují na aktivní a pasivní. V aktivním režimu komunikace iniciátor i cíl musí generovat své vlastní radiofrekvenční pole. To znamená, že musí mít obě zařízení vlastní zdroj elektrické energie [12]. V režimu pasivní komunikace vytváří radiofrekvenční signál pouze iniciátor komunikace. Druhé pasivní zařízení, které je cílovým zařízením, tentokrát žádný vlastní zdroj elektrické energie nemá. Zařízení používá techniku známou jako modulace, jejímž prostřednictvím získává energii potřebnou pro přenos dat zpět do iniciátoru z vlastního elektromagnetického pole pomocí elektromagnetické indukce [12].

2.3 Režimy přenosu

2.3.1 Čtení/zápis

Režim čtení/zápis umožňuje zařízením číst a zapisovat informace uložené na NFC štítcích. V tomto režimu je jedno zařízení aktivní, tedy s vlastním zdrojem napájení, a druhé zařízení je pasivní, a je napájeno elektromagnetickým polem aktivního zařízení, které je vždy iniciátorem komunikace [13].

2.3.2 Emulace karty

V režimu emulace karty spolu komunikují dvě aktivní zařízení. První zařízení, obvykle mobilní telefon, pracuje v pasivním režimu podobně jako běžná čipová karta. Druhým zařízením komunikace je iniciátor. Komunikace je zahájena vložením zařízení pracujícím v pasivním režimu (mobilní telefon) do rádiového pole čtečky [13].

2.3.3 Peer-to-peer

Peer-to-peer (doslova rovný s rovným) je režim umožňující obousměrnou komunikaci mezi NFC zařízeními. Je vhodný především pro vzájemnou výměnu dat, kontaktů či textových zpráv. Komunikující zařízení pracují v aktivním režimu [13].

2.4 Příbuzné technologie

Bezdrátová technologie Bluetooth nahradila tradiční kabelové připojení mezi mobilními telefony, notebooky a dalšími výpočetními a komunikačními zařízeními v rozmezí 10 metrů [17].

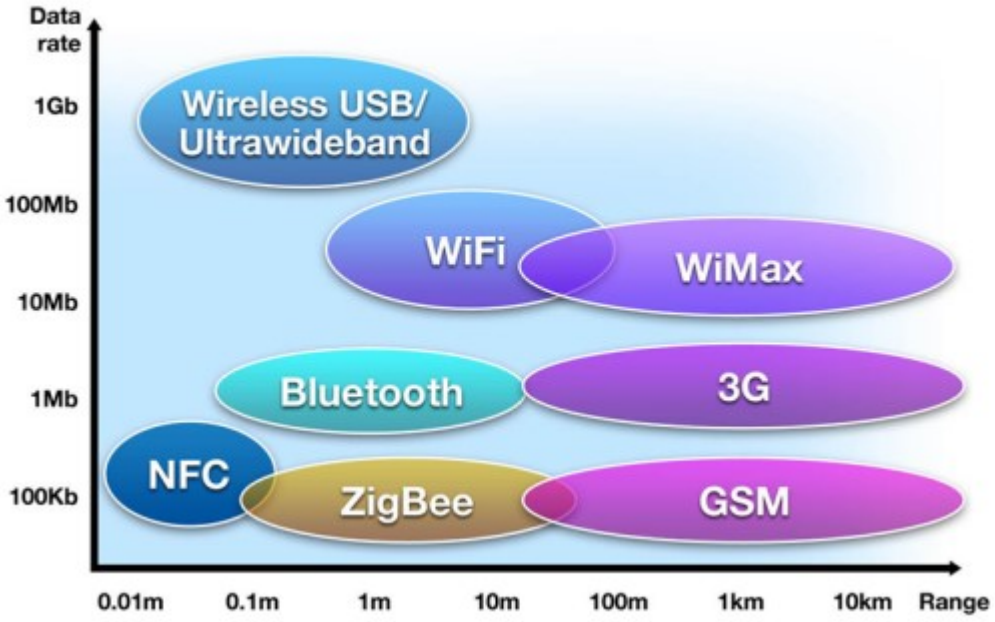
Technologie Wi-Fi byla navržena a optimalizována pro místní sítě LAN. Poskytuje rozšíření nebo výměnu kabelových sítí pro desítky výpočetních zařízení v rozmezí + 100 metrů [17].

Bezdrátová technologie ZigBee je standardní technologií umožňující ovládání a monitorování průmyslových a obytných aplikací v rozsahu + 100 metrů [17].

IrDA je komunikační standard krátkého dosahu (<1 metr), který umožňuje výměnu dat přes infračervené světlo. Rozhraní IrDA se často používají v počítačích a mobilních telefonech [17].

RFID (*Radio Frequency Identification*) je automatická identifikační metoda založená na ukládání a vzdáleném načítání dat pomocí zařízení nazývaných RFID tagy. RFID tag je malý objekt, který může být připojen k výrobku nebo je přímo jeho součástí. Značky RFID obsahují křemíkové čipy, které umožňují přijímat a reagovat na dotazy přicházející ze čtečky / zapisovače RFID [17].

Bezkontaktní čipové karty obsahují čip (mikroprocesor), který komunikuje s čtečkou karet pomocí technologie RFID. Příklady bezdotykové komunikace čipových karet jsou ISO / IEC 14443 a FeliCa umožňující komunikovat na vzdálenost až 10 cm [17].



Obrázek 8: Vlastnosti přenosu bezdrátových technologií [15]

2.5 Bezpečnost

NFC technologie s jejími standardy řeší pouze její funkcionalitu a kompatibilitu zařízení. Bezpečnostní oblasti zajišťuje subjekt, který technologii užívá v požadované aplikaci. Vzhledem k tomu, že jde především o zabezpečení informací přenášených pomocí NFC technologie, jsou k zabezpečení využívány postupy, které jsou aplikovány i u ostatních technologií zabývajících se přenosy informací. Velmi krátký dosah této technologie může sice odposlech komunikace znesnadnit, nikoli mu však zcela zabránit.

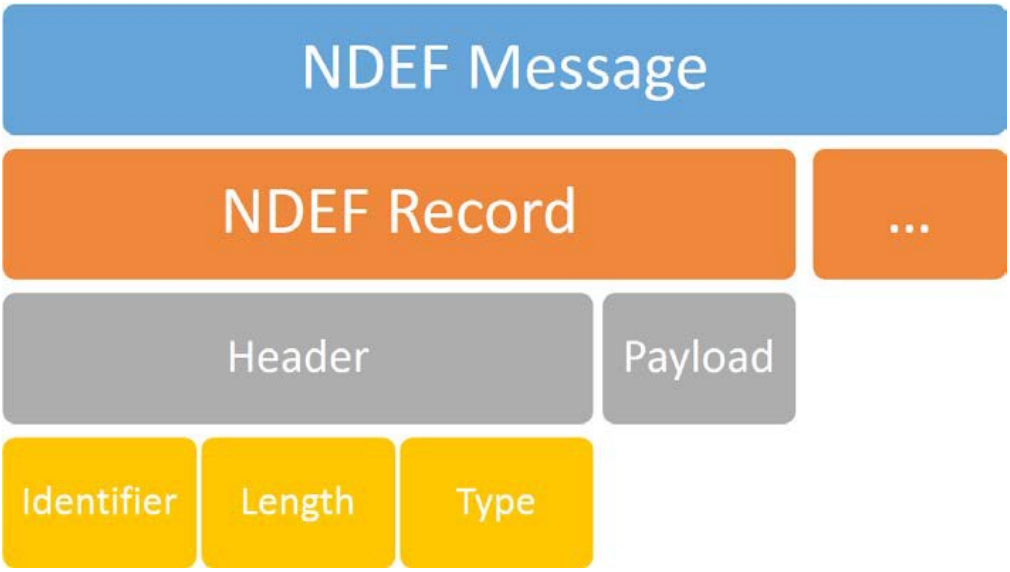
K tomuto účelu slouží kryptografie, která je společně s kryptoanalýzou součástí vědního oboru známého jako kryptologie. Smyslem kryptografie je šifrovat otevřený text přenášený informačním kanálem takovým způsobem, aby byl nečitelný pro nežádoucí a necílový subjekt. Příjemce zabezpečené komunikace má prostředky k dešifrování a může tak nazpět získat původní podobu otevřeného textu. Nemusí jít samozřejmě pouze o text, přenášené informace mohou být různého typu, jako například databázové soubory, programy, multimediální soubory, atd. [18].

3 NFC ANDROID API

V této kapitole čerpáme ze zdrojů uvedených v [10], [19] a [20]. Platforma Android poskytuje API pro práci s NFC tagy. Pomocí API máme k dispozici přístup k funkcím NFC, což umožňuje aplikacím číst a zapisovat NDEF zprávy do NFC tagů.

Mezi základní třídy API patří NfcAdapter, jehož prostřednictvím lze získat instanci třídy NfcAdapter, která je vstupním bodem pro provádění operací s NFC technologií. Instanci třídy NfcAdapter lze také získat pomocí její statické metody getDefaultAdapter(Context context), kde context je kontextem volající aplikace. Pokud se podaří vytvořit instanci adaptéru (návratová hodnota nebude null), je NFC technologie zařízením podporována. NfcAdapter dále poskytuje metody pro ověření, jestli není v nastavení NFC vypnuto, nebo metody pro povolení či zakázání odesílání NFC záměrů do aplikace. NFC záměry je možné povolit nebo zakázat, a také filtrovat. Lze tak řízením dosahovat stavu, že bude aplikace obsluhovat záměry pouze od tagů s konkrétním druhem obsahu, nebo na základě jejich technologie.

Další důležitou třídou tohoto API je NdefMessage, která reprezentuje datovou zprávu NDEF (NFC Data Exchange Format). Jedná se o standardní formát, ve kterém jsou mezi zařízeními a tagy přenášeny záznamy nesoucí data. NdefMessage je tvořena jedním nebo více záznamy NDEF (instance třídy NdefRecord). Každý NdefRecord obsahuje užitečné zatížení a záhlaví, ve kterém je uložena délka záznamu, jeho typ a identifikátor. Tuto strukturu zachycuje obrázek 9.



Obrázek 9: Struktura NdefMessage [19]

3.1 Systém dispečinku tagů

Z pohledu operačního systému Android je NFC záměr obsloužen podobně jako kterýkoli jiný implicitní záměr. To znamená, že vyhledá aktivitu, jež je schopna daný záměr obsloužit, a kterou následně spustí. Pokud existuje více takových aktivit, zobrazí jejich seznam a výběr ponechá na rozhodnutí uživatele. Záměr je datová struktura, která může nést informaci o operaci, jež se má vykonat, nebo může reprezentovat událost, která nastala v systému a je potřeba ji obsloužit. Touto událostí je i NFC záměr vyvolaný přiložením NFC tagu ke čtecímu zařízení. Pokud je v popředí spuštěna aplikace, která je schopna příchozí záměr obsloužit, je upřednostněna a k výběru nedochází.

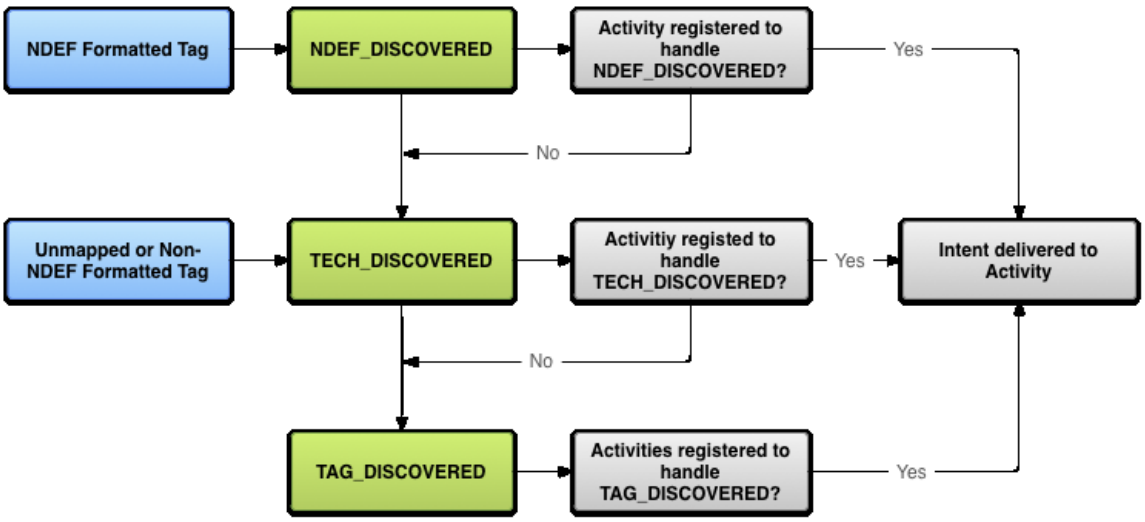
Operační systém Android NFC záměr ještě dále upřesňuje a dispečerský systém je poté schopen vyhledávat vhodnou aktivitu podle těchto doplňujících rozlišení v NFC záměrech. Je proto vhodné specifikovat druhy NFC záměrů, které chceme aplikací zpracovávat. Tak lze dosáhnout stavu, že nebude zobrazována volba pro výběr aktivit pro obsluhu záměru, přičemž bude spuštěna aktivita, která deklaruje, že přijímá blíže specifikovaný typ záměru.

Jestliže je aplikace v popředí, je do ní adaptérem odesílán jakýkoli NFC záměr, který je následně předán metodě onNewIntent(Intent intent). Protože je touto metodou přijímán jakýkoli záměr, je nutné ověřit, jestli jde o NFC záměr, který chceme aktivitou běžící v popředí zpracovávat. Metodu onNewIntent je nutné přepsat dle našich potřeb. Aktivita může být také kdykoli operačním systémem přesunuta na pozadí, a proto je vhodné přepsat i metody onPause() a onResume(), které patří do životního cyklu aktivity. V těchto metodách je třeba zakázat, respektive povolit, dispečerský systém popředí.

K tomuto účelu je vhodné využít třídu PendingIntent, která NFC intent zapouzdří. Objekt třídy PendingIntent je dále systémem Android naplněn podrobnostmi o tagu, který záměr vyvolal. Jestliže předáme uvedenému objektu instanci třídy IntentFilter, ve které definujeme, jaký typ záměru chceme zachytávat, bude tento záměr předán zpět do naší aplikace. V opačném případě, tedy jestliže definovaným filtrům neodpovídá, je vrácen zpět dispečerskému systému, který se pokusí nalézt aktivitu umožňující záměr obsloužit.

3.2 Zpracování NFC záměru

Android definuje tři základní typy záměrů při detekci NFC tagů. Záměry mají svou prioritu, podle které je dispečerský systém zařazuje. Schéma na obrázku 10 ilustruje, jak v systému odesílání tagů k zařazení dochází.



Obrázek 10: Systém odesílání tagů [20]

Rozlišujeme tedy následující události při detekci NFC značky:

NDEF_DISCOVERED: záměr je užíván ke spuštění aktivity za podmínky, jestliže je naskenován NFC tag, který obsahuje NdefRecord. Tento záměr má nejvyšší prioritu, a proto je systémem před ostatními záměry upřednostňován.

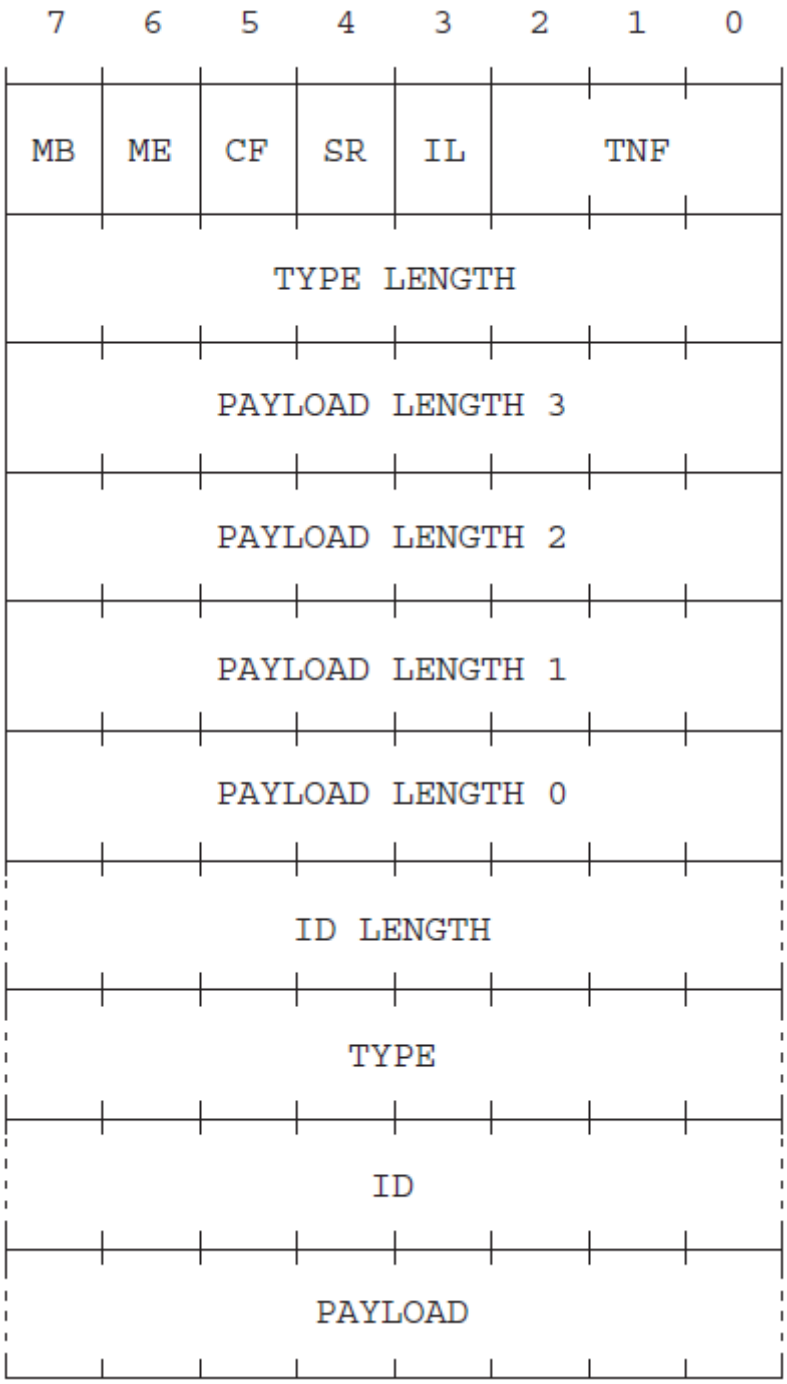
TECH_DISCOVERED: Pokud není nalezena aktivita, která je schopna obsloužit NDEF záměr, pokusí se dispečerský systém spustit záměr tohoto typu. Záměr je spuštěn také za předpokladu, že se jedná o známou technologii tagu, jež nebyl dosud formátován.

TAG_DISCOVERED: Záměr je spuštěn, jestliže se systému nepodaří spustit žádná aktivita na základě předchozích dvou událostí. Jedná se o záměr s nejnižší prioritou.

Z výše uvedeného plyne, že je výhodné využívat NDEF zprávy, čímž nebude naše aplikace znevýhodněna před jinou aplikací, která NDEF zprávy využívá.

3.3 NDEF zpráva

NDEF (*NFC Data Exchange Format*) data zapsaná na tagu jsou v operačním systému Android zapouzdřena objektem třídy NdefMessage. Struktura NDEF zprávy je vidět na obrázku 11.



Obrázek 11: Struktura zprávy NDEF [10]

NDEF zpráva obsahuje jeden nebo více záznamů reprezentovaných objekty třídy NdefRecord. Jestliže se podaří načíst formátovaný tag tímto způsobem, systém se pokusí analyzovat data podrobněji, a to na základě obsahu prvního záznamu.

Pro upřesnění záměru, díky kterému je možné detailněji specifikovat NFC záměr, jsou významné položky:

- TNF (*Type Name Format*) označuje strukturu položky TYPE. Jedná se o 3bitovou hodnotu, která může nabývat hodnot uvedených v tabulce 1.
- TYPE popisuje typ záznamu. Pokud má pole TNF hodnotu TNF_WELL_KNOWN, upřesňuje skutečné užitečné zatížení.

Tabulka 1: Podporované hodnoty TNF

TNF_ABSOLUTE_URI	Pole TYPE obsahuje hodnotu, která odpovídá konstruktu BNF absolutního URI definovaného v RFC 3986.
TNF_EMPTY	Záznam je prázdný, není přiřazen žádný typ ani užitečné zatížení.
TNF_EXTERNAL_TYPE	Pole TYPE obsahuje hodnotu, která odpovídá formátu názvu RTD definovaného ve specifikaci NFC Forum pro názvy externích typů.
TNF_MIME_MEDIA	Pole TYPE obsahuje konstrukt BNF typu média definovaný RFC 2046.
TNF_UNCHANGED	Určeno pro označení záznamů, které nejsou první, pokud je takto označen první záznam, přičemž je považován za neplatný a zpráva spadá zpět do ACTION_TECH_DISCOVERED.
TNF_UNKNOWN	Označuje, že typ užitečného zatížení není znám, zpráva spadá zpět do ACTION_TECH_DISCOVERED.
TNF_WELL_KNOWN	Pole TYPE obsahuje známý název typu RTD, např. RTD_TEXT nebo RTD_URI

Význam zbývajících položek je následující:

- MB: Příznak označující začátek NDEF zprávy.
- ME: Příznak označující konec NDEF zprávy.
- CF: Příznak zřetězeného záznamu.
- SR: Příznak označující krátký formát užitečného zatížení. PAYLOAD_LENGTH je v tomto případě pouze jeden octed(byte).
- IL: Příznak značí, že záznam obsahuje pole s délkou ID. Pokud je tento příznak nastaven, znamená to, že záznam obsahuje pole ID_LENGTH o velikosti jeden octed. Pokud příznak nastaven není, záznam pole ID_LENGTH a ID neobsahuje.
- TYPE_LENGTH: Je 8bitové celé číslo bez znaménka, které určuje délku pole TYPE.
- ID_LENGTH: Je 8bitové celé číslo bez znaménka, které udává délku pole ID. V záznamu je přítomno pouze v případě, pokud je nastaven (obsahuje hodnotu 1) příznak IL.
- PAYLOAD_LENGTH: Je celé číslo bez znaménka, které udává délku užitečného zatížení.
- ID: Volitelné pole identifikující užitečné zatížení.
- PAYLOAD: Obsahuje užitečné zatížení určené pro uživatelskou aplikaci.

3.4 Vytvoření NDEF záznamu

Struktura Ndef záznamu je popsána v předchozí kapitole v souvislosti s určením typu záměru, který k tomuto účelu využívá první záznam v obdržené zprávě.

NDEF (*NFC Data Exchange Format*) záznam je v operačním systému Android reprezentován třídou NdefRecord. Nejobecnější formou, jak lze vytvořit tento záznam je s využitím konstruktoru, který má následující signaturu:

```
public NdefRecord(short tnf, byte[] type, byte[] id, byte[] payload)
```

Použití konstruktoru k vytvoření záznamu je vhodné pouze v případech, kdy nelze záznam vytvořit pomocí statických metod třídy NdefRecord, které vytváří správně naformátované záznamy konkrétního typu dle standardu organizace NFC Forum. Jedná se o následující veřejné statické metody:

```
createTextRecord (String languageCode, String text)
```

```
createApplicationRecord (String packageName)
createExternal (String domain, String type, byte[] data)
createMime (String mimeType, byte[] mimeData)
createUri (Uri uri)
createUri (String uriString)
```

3.5 Povolení NFC technologie v manifestu

Operační systém Android vyžaduje, aby uživatel poskytl aplikaci oprávnění k citlivým údajům, nebo určitým službám, které poskytuje. Jedná se především o operace, které by mohly nepříznivě ovlivnit jiné aplikace nebo uživatele. Pokud aplikace potřebuje uvedené prostředky pro svou funkčnost, musí tuto skutečnost definovat v souboru AndroidManifest.xml a operační systém poté vyzve uživatele, jestli má zájem aplikaci povolení udělit.

Přestože není povolení od uživatele nutné pro všechny funkce, je doporučeno definovat tato oprávnění pro veškeré funkce, které bude aplikace využívat. Rovněž je vhodné definovat případná omezení pro jejich správnou funkčnost. Jinak tomu není ani při přístupu k NFC hardwaru telefonu. Jedná se o následující položky:

1. Element <uses-permission> pro přístup k hardwaru NFC.

```
<uses-permission android:name = "android.permission.NFC" />
```

2. Element <uses-sdk> pro definici minimální verze SDK.

```
<uses-sdk android:minSdkVersion = "10" />
```

3. Element <uses-feature> pro filtrování aplikací v Android Marketu.

```
<uses-feature android:name = "android.hardware.nfc"
    android:required = "true" />
```

3.6 Filtrování tagů podle obsahu

Pokud chceme, aby naše aplikace zpracovávala NFC záměry, které jsou vyvolány pouze tagy s určitým typem obsahu, můžeme vytvořit filtr, který bude tyto typy obsahu specifikovat.

Prvním způsobem, jak docílit filtrování tagu podle obsahu, je specifikace filtru v souboru AndroidManifest.xml. Operačnímu systému sdělíme, že aktivita přijímá pouze záměry odpovídající uvedeným filtrům.

Následující ukázka demonstruje, jak vytvořit filtr pro NDEF_DISCOVERED záměr, jehož MIME typ je text/plain.

```
<intent-filter>
    <action android:name = "android.nfc.action.NDEF_DISCOVERED" />
    <category android:name = "android.intent.category.DEFAULT" />
    <data android:mimeType = "text/plain" />
</intent-filter>
```

Druhý způsob využijeme, pokud chceme záměry filtrovat přímo v kódu aktivity, která NFC záměry zpracovává, abychom mohli řídit dispečerský systém popředí. Použití tohoto filtru pro systém v popředí zahrnuje několik datových struktur, které specifikují, jestli chceme zaslat příslušné záměry do naší aplikace.

Tyto záměry chceme filtrovat na základě toho, jestli je daná aktivita v popředí či nikoli. Uvedeného stavu docílíme tak, že kód pro filtrování záměrů budeme povolovat, respektive zakazovat, v metodách onResume() a onPause(). Povolení se provádí pomocí metody enableForegroundDispatch třídy NfcAdapter, která očekává čtyři argumenty. V třetím argumentu přijímá filtry tagů založených na jejich obsahu. Tímto argumentem tedy specifikujeme, že chceme přijímat pouze záměry vyvolané tagy s obsahem, který odpovídá některému z filtrů.

Pro zakázání odesílání záměrů (pokud přechází aktivita na pozadí) slouží metoda disableForegroundDispatch třídy NfcAdapter.

3.7 Filtrování tagů podle technologie

Filtrování podle technologie se provádí obdobně jako u filtrování podle obsahu. Pokud hodláme vytvořit aplikaci, která je spuštěna pouze tehdy, když se technologie tagu shoduje s námi požadovanou technologií, definujeme technologii v souboru AndroidManifest.xml.

Požadované technologie, které chceme aplikací zpracovávat, uvedeme v xml souboru s libovolným názvem. Soubor uložíme do nové složky s názvem xml v adresáři res projektu.

Následující kód níže ukazuje obsah tohoto souboru s definicí všech technologií.

```
<resources xmlns:xliff = "urn:oasis:names:tc:xliff:document:1.2">
    <tech-list>
        <tech>android.nfc.tech.IsoDep</tech>
```

```
        <tech>android.nfc.tech.NfcA</tech>
        <tech>android.nfc.tech.NfcB</tech>
        <tech>android.nfc.tech.NfcF</tech>
        <tech>android.nfc.tech.NfcV</tech>
        <tech>android.nfc.tech.Ndef</tech>
        <tech>android.nfc.tech.NdefFormatable</tech>
        <tech>android.nfc.tech.MifareClassic</tech>
        <tech>android.nfc.tech.MifareUltralight</tech>
    </tech-list>
</resources>
```

Takto vytvořený soubor poté uvedeme v <meta-data> tagu uvnitř definice aktivity, jak demonstruje ukázka níže.

```
<activity>
...
<intent-filter>
    <action android:name = "android.nfc.action.TECH_DISCOVERED" />
</intent-filter>
<meta-data android:name = "android.nfc.action.TECH_DISCOVERED"
    android:resources = "@xml/nfc_tech_filter" />
...
</activity>
```

Podobně jako u filtrování podle obsahu lze i filtry podle technologie specifikovat pro dispečink v popředí, který umožňuje aplikaci v popředí zpracovávat záměry, přestože jiné aplikace totožné záměry rovněž filtrují. Pro filtrování dle technologie se tentokrát využije čtvrtý argument metody enableForegroundDispatch třídy NfcAdapter. Pokud bychom chtěli akceptovat pouze technologii typu ISO-DEP, bude tento argument vytvořen následovně:

```
techList = new String[][]{new String[]{IsoDep.class.getName()}};
```

3.8 Zpracování informací obdržených v NFC záměru

Pokud je již aktivita spuštěna, a je vyvolán nový NFC záměr, tak je volána metoda onNewIntent(). Metoda onNewIntent() je ovšem spuštěna libovolným záměrem, proto musíme provést ověření, abychom se ujistili, že se jedná o NFC záměr.

Metoda onNewIntent má návratový typ void. Jestliže chceme v této metodě zpracovávat pouze NFC záměr, můžeme na začátek této metody vložit následující kód:

```
if (!intent.hasExtra(NfcAdapter.EXTRA_TAG)) {
      return;
}
```

Po ověření, že se jedná o NFC záměr, můžeme z tohoto záměru získat instanci reprezentující tag.

```
Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
```

Takto získanou reprezentaci tagu lze použít pro vytvoření instancí tříd Ndef a NdefFormatable potřebné pro zápis na tag nebo jeho formátování.

```
Ndef ndef = Ndef.get(tag);
```

```
NdefFormatable ndefFormat = NdefFormatable.get(tag);
```

Pokud je akce záměru typu ACTION_NDEF_DISCOVERED, získáme NdefMessage pomocí metody getParcelableArrayExtra následovně.

```
Parcelable[] messages =
   intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
NdefMessage message = (NdefMessage) messages[0];
```

3.9 Android Application Record (AAR)

Záznam typu AAR (Android Application Record) poskytuje silnější jistotu, že bude NFC záměr zpracován námi požadovanou aplikací. Přesněji řečeno, bude spuštěna aplikace s názvem balíčku (package), který je uveden v tomto druhu záznamu, přestože je v zařízení instalováno více aplikací, jež jsou schopné tento NFC záměr zpracovat.

Operační systém Android se totiž kromě určení typu NFC zprávy na základě prvního záznamu pokouší najít ve zprávě i tento druh záznamu. Pokud ho nalezne, upřednostní aplikaci s tímto balíčkem. Jestliže takovou aplikaci nenalezne, pokusí se ji najít na Google Play a nabídnout ji uživateli ke stažení.

AAR záznamy je vhodné použít zejména v případě, pokud je NFC tag speciálně určen pro zpracovávání konkrétních aplikací. AAR záznam je podporován pouze na aplikační úrovni a nikoli na úrovni aktivity jako u filtrování záměrů, a to z důvodu omezení podle názvů balíčků.

K vytvoření tohoto záznamu poskytuje třída NdefRecord jednu ze svých statických metod pro tvorbu známých druhů záznamů. Záznam se tvoří následovně:

```
NdefRecord.createApplicationRecord("cz.tevogs.cnm ")}
```

II. PRAKTICKÁ ČÁST

4 IMPLEMENTACE APLIKACE TVORBY TAGU

Prvním úkolem je vytvořit Android aplikaci, která umožní vytvořit nebo načíst soubor s konfigurací, jež bude poté nahrán do NFC tagu.

4.1 Požadavky na aplikaci

Aplikace musí poskytnout grafické uživatelské rozhraní pro zadávání nezbytných údajů pro konfiguraci, které mají být zapsány do NFC tagu. Vložené údaje budou dále rozděleny do dvou skupin na základě potřeby jejich zabezpečení šifrováním.

4.1.1 Nešifrované položky tagu

Verze formátu slouží k identifikaci verze obsahu tagu. Položky "clientId" a "title" jsou identifikátory klientské aplikace, přičemž "clientId" bude užíván jako identifikátor pro server. Položka "title" je pro uživatele srozumitelnější forma, podle které budou v klientské aplikaci ukládány již načtené konfigurace. Dle tohoto identifikátoru bude umožněno uživateli zvolit konfiguraci bez nutnosti načítat konfiguraci z tagu. Dvojice položek "keyId" a "iv" slouží k dešifrování šifrované části obsahu tagu. Poslední položka z nešifrované části je "sign". Položka slouží k zajištění integrity šifrované části tagu. Obsahem položky je SHA-256 hash šifrovaného obsahu, který je podepsán privátním klíčem vydavatele tagu.

Tabulka 2: Položky nešifrovaného obsahu tagu.

název	**popis**
formatVersion	verze formátu konfigurace
clientId	identifikátor klienta
title	titulek tagu
keyId	identifikátor klíče pro dešifrování chráněného obsahu
iv	inicializační vektor pro dešifrování chráněného obsahu
sign	podpis zašifrované části tagu

4.1.2 Šifrované položky tagu

Šifrované položky jsou shrnuty v Tabulce 3. Položka "componentId" je jedinečný identifikátor klientské komponenty, který je generován při registraci vozidla do systému. Pro potřebu autentizace a šifrování komunikace mezi klientem a serverem tag obsahuje privátní klíč pod názvem "authkey". K navázání a konfiguraci spojení se serverem slouží položky "server", "distributionType" a "multicastGroup". Položky "wifiSSID" a "wifiPassword" slouží pro připojení ke konkrétní Wi-Fi vozidla, přičemž bude klientskou aplikací vynucováno stálé připojení pouze k této Wi-Fi . Šifrování bude provedeno pomocí symetrické kryptografické šifry AES-128.

Tabulka 3: Položky šifrovaného obsahu tagu.

název	**popis**
componentId	identifikátor klientské aplikace
authkey	PEM RSA 1024b private key
server	IP serveru
distributionType	způsob distribuce dat ze serveru
multicastGroup	multicast skupina, IP adresa typu D
wifiSSID	název sítě Wi-Fi
wifiPasword	heslo k Wi-Fi

4.1.3 Způsoby tvorby obsahu

Způsob zadávání položek je vyžadován třemi přístupy:

1. veškeré potřebné údaje jsou zadány ručně pomocí uživatelského rozhraní
2. konfigurace je načítána ze souboru
3. konfigurace je načítána z tagu, který již tuto konfiguraci obsahuje

4.1.4 Správa použitých kryptografických klíčů

AES klíče použité k šifrování zabezpečeného obsahu tagu a certifikát k ověření integrity zašifrovaného obsahu musí být exportovatelné pro použití v klientské aplikaci.

Tyto klíče budou předávány pomocí souboru obsahujícího klíčové úložiště vytvořeného pomocí Java KeyStore API. Toto klíčové úložiště bude chráněno heslem.

4.2 Návrh funkcionalit vycházejících z požadavků na aplikaci

Na základě požadavků je nutné navrhnout grafické uživatelské rozhraní umožňující různé způsoby zadávání vstupních položek, které se budou zapisovat do NFC tagu.

- Implementovat mechanismus generování kryptografických klíčů potřebných pro šifrování a umožnit export klíčů potřebných k dešifrování obsahu. S tím souvisí i správa hesla pro uzamčení exportovaného úložiště klíčů.
- Navrhnout formát, ve kterém budou jednotlivé položky zapsány na tag. Návrh musí být vytvořen s ohledem na jejich členění dle nutnosti šifrování.
- Zajistit posloupnost ověřovacích operací, která zajistí, že aplikace bude mít veškeré potřebné informace k provádění dalších kroků. Například heslo musí být zadáno před exportem úložiště klíčů, nebo musí být vygenerován alespoň jeden AES klíč před zápisem obsahu na NFC tag.
- Vytvořit validační pravidla, která budou kontrolovat vstupní data před jejich zapsáním na NFC tag.

4.2.1 Návrh posloupnosti operací

Posloupnost operací je důležitá, protože zajišťuje, aby se aplikace nenacházela v nekonzistentním stavu, který by mohl způsobovat selhání z důvodu absence potřebných vstupů k provedení jednotlivých operací. Navržená posloupnost operací je zobrazena pomocí diagramu na obrázku 12.

Prvním krokem je ověření přítomnosti alespoň jednoho AES klíče nutného pro šifrování obsahu. Součástí prvního kroku je zajištění přítomnosti klíčového páru k podpisu tagu a hesla, kterým bude uzamčen soubor s úložištěm klíčů. Ověření přítomnosti AES klíče a klíčového páru je provedeno v prvním kroku, což je nezbytné k uskutečnění zápisu na tag. Nutnou podmínkou pro vytvoření úložiště klíčů je zadání hesla, které bude sloužit v klientské aplikaci k dešifrování.

Druhým krokem je volba způsobu vytváření obsahu tagu uživatelem. V tomto kroku je také umožněna správa klíčů a hesla k úložišti určenému pro export.

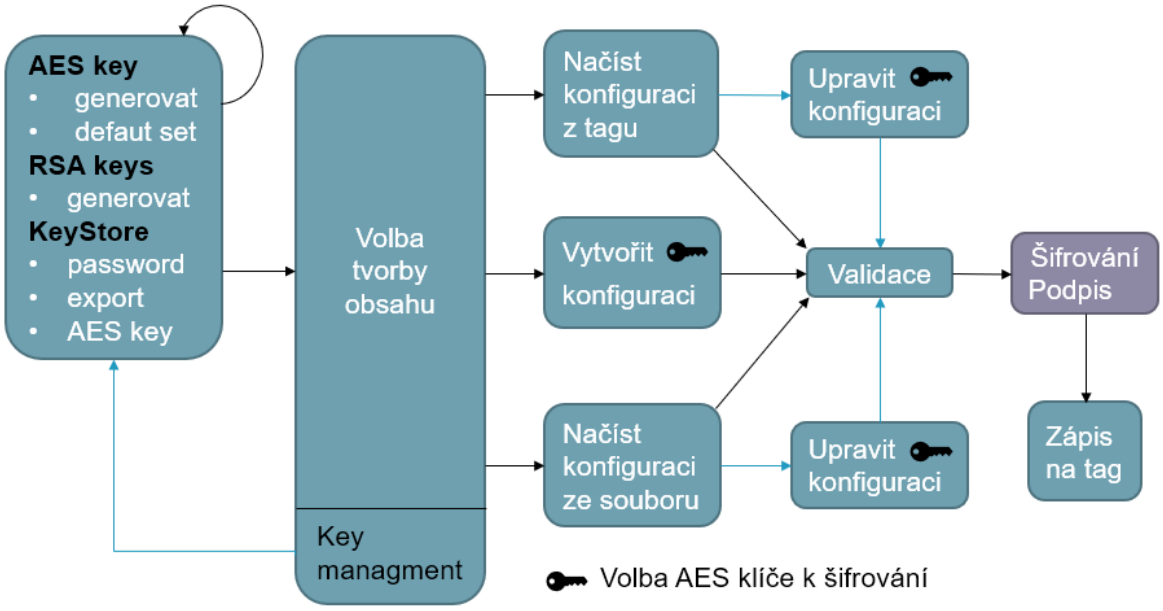
Na základě zvoleného způsobu bude uživatel přesměrován na patřičnou aktivitu, která bude tuto volbu obsluhovat. Jednotlivé volby budou obslouženy následujícím způsobem:

- Vytvořit konfiguraci: Při této volbě bude uživateli zobrazen formulář pro vložení všech potřebných údajů včetně výběru šifrovacího klíče, pokud úložiště obsahuje více klíčů.
- Načíst konfiguraci ze souboru: Po této volbě bude uživateli umožněno načíst konfiguraci pomocí procházení souborového systému. Obsah načtené konfigurace bude zobrazen uživateli, který se bude moci rozhodovat, jestli si přeje nezměněnou konfiguraci rovnou zapsat na tag, případně jestli ji chce ještě upravit. Mezi možné úpravy bude patřit i možnost změny šifrovacího klíče.
- Načíst Konfiguraci z tagu: Tato volba umožňuje k tvorbě tagu použít již dříve vytvořený tag. Zároveň může posloužit jako okamžitá kontrola o tom, že byl právě vytvořený tag správně zapsán. Opět bude po načtení zobrazen náhled obsahu s možností přímého zápisu bez změn, nebo nejdříve budou provedeny změny, jako v případě volby načítání ze souboru.

Pro všechny výše uvedené volby platí, že než je vytvořená konfigurace zapsána na tag, je nejdříve provedena validace této konfigurace, aby se tak předešlo zápisu obsahu tagu, který nesplňuje požadavky kladené na jednotlivé záznamy.

Pokud obsah validační pravidla splní, dojde k zašifrování citlivého obsahu tagu a k podpisu, který bude sloužit k ověření, že nedošlo k modifikaci obsahu.

Posledním krokem tohoto cyklu je samotný zápis obsahu do NFC tagu.



Obrázek 12: Životní cyklus tvorby tagu.

4.3 Návrh formátu zapisovaného na NFC tag

Jak při tvorbě tagu tak při jeho zpracování jsou jednotlivé hodnoty, které chceme pomocí tagu přenášet, uchovávány v objektech. Pro formát, ve kterém budou data zapsány na tag, byl zvolen Json, což usnadní načítání informací z tagu zpět do objektů, které tato data zapouzdřují. Další výhodou je skutečnost, že se jedná o textovou reprezentaci, kterou lze snadno zapsat v NDEF formátu jako TNF_WELL_KNOWN / RTD_TEXT. Celý obsah zapisovaný na tag bude reprezentován objektem, jehož obsahem jsou obálky formou objektů, které již budou reprezentovat jednotlivé NDEF záznamy tvořící NDEF zprávu. V rámci návrhu tohoto formátu bylo vytvořeno Json Schema exaktně popisující strukturu záznamu včetně validačních pravidel. Schéma je obsahem přílohy.

Základní struktura schématu je následující:

- MessageRecordWrapper
  - headerRecordWrapper
    - clientId
    - formatVersion
    - iv
    - keyId
    - sign
    - title

- bodyRecordWrapper
  - authkey
  - componentId
  - distributionType
  - multicastGroup
  - server
  - wifiPassword
  - wifiSSID
- aarPackageName

Význam jednotlivých položek je již popsán v kapitole 4.1. Z této struktury vyplývá, že bude nutné navrhnout NDEF zprávu se třemi NDEF záznamy. Rozdělení na části "header" a "body" je z důvodu rozdělení těchto položek na ty, které budou, respektive nebudou šifrovány. Každá tato sekce bude na NFC tag uložena v samostatném NDEF záznamu.

4.4 Struktura NDEF zprávy

Jak již z návrhu formátu vyplynulo, bude NdefMessage tvořena třemi NdefRecord.

4.4.1 NdefRecord 1 – nešifrovaný obsah tagu

Pro tento záznam byl zvolen formát typu TNF_WELL_KNOWN / RTD_TEXT. Jedná se o text ve formátu Json, který reprezentuje položky třídy HeaderRecordWrapper.

Záznam je vytvořen pomocí následujícího kódu:

```
private NdefRecord createTextRecord(String content) {
    byte[] language;
    language = Locale.ENGLISH.getLanguage().getBytes();
    final byte[] text = content.getBytes();
    final int languageSize = language.length;
    final int textLength = text.length;
    final ByteArrayOutputStream payload =
       new ByteArrayOutputStream(1 + languageSize + textLength);
    payload.write((byte) (languageSize & 0x1F));
    payload.write(language, 0, languageSize);
    payload.write(text, 0, textLength);
    return new NdefRecord(NdefRecord.TNF_WELL_KNOWN,
```

```
        NdefRecord.RTD_TEXT,
        new byte[0], payload.toByteArray());
    }
```

Užitečné zatížení přímo neodpovídá předanému řetězci, ale je upraveno v závislosti na zvoleném jazyku. Zde použitý anglický jazyk je požadavkem zadavatele. Protože se jedná o první záznam zprávy, je tento záznam využíván pro určení typu záměru pro dispečerský systém, jak bylo popsáno v kapitole 2.4.

4.4.2 NdefRecord 2 – šifrovaný obsah tagu

Obsah tohoto záznamu bude šifrován, proto pro něj byl zvolen formát typu TNF_UNKNOWN. Záznam bude obsahovat pole bajtů s šifrovaným obsahem Json reprezentace třídy BodyRecordWrapper.

Záznam je vytvořen níže uvedenou metodou:

```
private NdefRecord createUnknownRecord(byte[] content) {
    return new NdefRecord(NdefRecord.TNF_UNKNOWN, new byte[0],
        new byte[0], content);
}
```

Zde užitečné zatížení přímo odpovídá předanému poli bajtů, které obsahuje šifrovaný obsah části "body".

4.4.3 NdefRecord 3 – Android Application Record (AAR)

Tento záznam není nositelem informace potřebné pro vytvoření konfigurace, jak je tomu u dvou předešlých, ale upřednostní klientskou aplikaci jako příjemce záměru, který bude vyvolán načtením NFC tagu. Jeho obsahem je název balíčku (package) klientské aplikace. Operační systém Android má pro tento účel speciální formát záznamu, a pokud ho NDEF zpráva obsahuje, je záměr zaslán do aplikace s názvem balíčku uvedeného v tomto záznamu.

AAR záznam se vytváří pomocí statické metody třídy NdefRecord:

```
NdefRecord.createApplicationRecord(AAR_RECORD);
```

Kde AAR_RECORD je konstanta obsahující název balíčku cílové aplikace, pro kterou je NFC tag tvořen.

4.4.4 Zápis NDEF zprávy na tag

Poté co jsou vytvořeny jednotlivé záznamy tvořící zprávu, je možné z nich vytvořit NDEF zprávu pro zápis na NFC tag.

Prvním krokem je vytvoření instance NdefMessage:

```
NdefMessage ndefMessage = new NdefMessage(new NdefRecord[] {
        headNdefRecord, bodyNdefRecord, aaRecord
});
```

Vytvoření instance se provede způsobem, že se konstruktoru třídy NdefMessage předá pole dříve vytvořených instancí NdefRecord.
Před samotným zápisem je ještě nutné vytvořit instanci třídy Tag, kterou je možné získat ze záměru vyvolaného přiložením tagu, na který chceme zprávu zapsat.

```
Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
```

Pokud máme instanci zprávy i tagu, můžeme již provést samotný zápis na tag.

```
private boolean writeNdefMessage(Tag tag, NdefMessage ndefMessage)
{
    try {
        if (tag != null) {
            Ndef ndef = Ndef.get(tag);
            if (ndef == null) {
                return formatTag(tag, ndefMessage);
            } else {
                ndef.connect();
                if (ndef.isWritable()) {
                    ndef.writeNdefMessage(ndefMessage);
                    ndef.close();
                    return true;
                }
                ndef.close();
            }
        }
    } catch (Exception e) {
        if (e.getMessage() == null) {
            Toast.makeText(this, "Tag write failed!",
```

```
                Toast.LENGTH_LONG).show();
        } else {
            Log.e("formatTag", e.getMessage());
            Toast.makeText(this, e.getMessage(),
              Toast.LENGTH_LONG).show();
        }
        finish();
    }
    return false;
}
```

Po ověření, že se podařilo získat instanci tagu, bude provedeno vytvoření instance třídy Ndef. Tento objekt nám poskytne metody pro načtení a úpravu NdefMessage tagu. Zde dochází k členění kódu v závislosti na tom, jestli je tag formátovaný či nikoli.

Pokud je tag formátovaný, což lze ověřit podle úspěšnosti vytvoření instance třídy Ndef, pokusíme se otevřít spojení se značkou voláním metody connect. Po otevření spojení je nutné ověřit, jestli je možné na tag zapisovat. Tag může být uzamknut proti zápisu. Pokud tag není uzamknutý, nic nebrání provést zápis voláním metody writeNdefMessage, které jako argument předáme již dříve vytvořenou instanci třídy NdefMessage. Nakonec ještě uzavřeme spojení s tagem voláním metody close.

Jestliže však tag formátován není, je třeba jej před samotným zápisem naformátovat. V kódu výše jde o volání metody formatTag, jejíž implementace je následující.

```
private boolean formatTag(Tag tag, NdefMessage ndefMessage) {
    try {
        NdefFormatable ndefFormat = NdefFormatable.get(tag);
        if (ndefFormat != null) {
            ndefFormat.connect();
            ndefFormat.format(ndefMessage);
            ndefFormat.close();
            return true;
        }
    } catch (Exception e) {
        Log.e("formatTag", e.getMessage());
    }
    return false;
}
```

Jako první je vytvořen objekt typu NdefFormatable. Dále je postup podobný jako v případě již naformátovaného tagu. Hlavním rozdílem je volání metody format, které se také předá instance třídy NdefMessage. Tato metoda kromě zápisu zprávy na tag zajistí i jeho naformátování.
V obou případech je návratovou hodnotou o úspěšnosti tohoto zápisu hodnota boolean. Rovněž je nutné ošetřit výjimky, které mohou v průběhu zápisu nastat.

4.5 GUI, aplikační logika

4.5.1 Správa úložiště klíčů

Navržené obrazovky, které se podílejí na plnění prvního kroku návrhu, tedy ověření potřebných šifrovacích klíčů a hesla, jsou na obrázku 13.

Na levé straně je vidět obrazovku pro zadání hesla, které slouží k uzamčení úložiště klíčů pro export. Obrazovka bude po instalaci aplikace zobrazena vždy jako první.

Jakmile uživatel uloží zvolené heslo, bude mu zobrazena obrazovka z pravé části obrázku, která slouží ke správě a exportu klíčového úložiště. Je zde také možnost nastavení defaultního AES klíče. Defaultní AES klíč je automaticky přednastaven jako šifrovací klíč při volbě vytvoření nové konfigurace. Pokud není tímto způsobem defaultní klíč nastaven, stává se defaultním klíčem poslední vygenerovaný klíč.

Je zde umístěna i možnost provedení změny hesla, které bylo vytvořeno v předešlém kroku, což zároveň může být využito k exportu úložiště s různými hesly pro jejich odemknutí.

Pokud pomocí těchto dvou obrazovek vytvoříme nezbytná prvotní nastavení, můžeme přejít k výběru způsobu tvorby obsahu, který chceme zapsat do tagu.

V aplikaci jsou udržována dvě klíčová úložiště. První úložiště je určeno k uchovávání klíčů, které využívá interně pouze tato aplikace. Druhé úložiště obsahuje klíče, které bude nutné exportovat pro potřeby klientské aplikace, jež bude obsah tagů načítat.

4.5.2 Interní klíčové úložiště

Pro interní klíčové úložiště je využito Android keystore systému. Tento systém chrání klíčový materiál před neoprávněným zneužitím a zabraňuje neoprávněnému použití klíčů mimo aplikační proces a jejich extrakci mimo zařízení.

V tomto úložišti je uložen privátní klíč pro podepisování obsahu šifrované části tagu. Také je zde uložen AES klíč pro šifrování hesla, které slouží k uzamknutí druhého klíčového úložiště určeného pro export.

Generování klíčového páru je provedeno pomocí třídy KeyPairGenerator. Generátor je inicializován s algoritmem eliptických křivek a poskytovatelem zabezpečení androidKeyStore. Při vytváření klíčového páru je také specifikováno, které algoritmy budou akceptovány při podepisování a ověřování.

```
public static KeyPair generateRsaKeyPair(String identifier) {
  KeyPairGenerator kpg = eyPairGenerator.getInstance(
    KeyProperties.KEY_ALGORITHM_EC, ANDROID_KEY_STORE);
  kpg.initialize(new KeyGenParameterSpec.Builder(
    RSA_CIPHER_PREFIX + identifier,
    KeyProperties.PURPOSE_SIGN | KeyProperties.PURPOSE_VERIFY)
    .setDigests(KeyProperties.DIGEST_SHA256,
                KeyProperties.DIGEST_SHA512)
    .build())
  return kpg.generateKeyPair();
}
```

AES klíč pro šifrování hesla úložiště, které bude exportováno, je generován kódem níže.

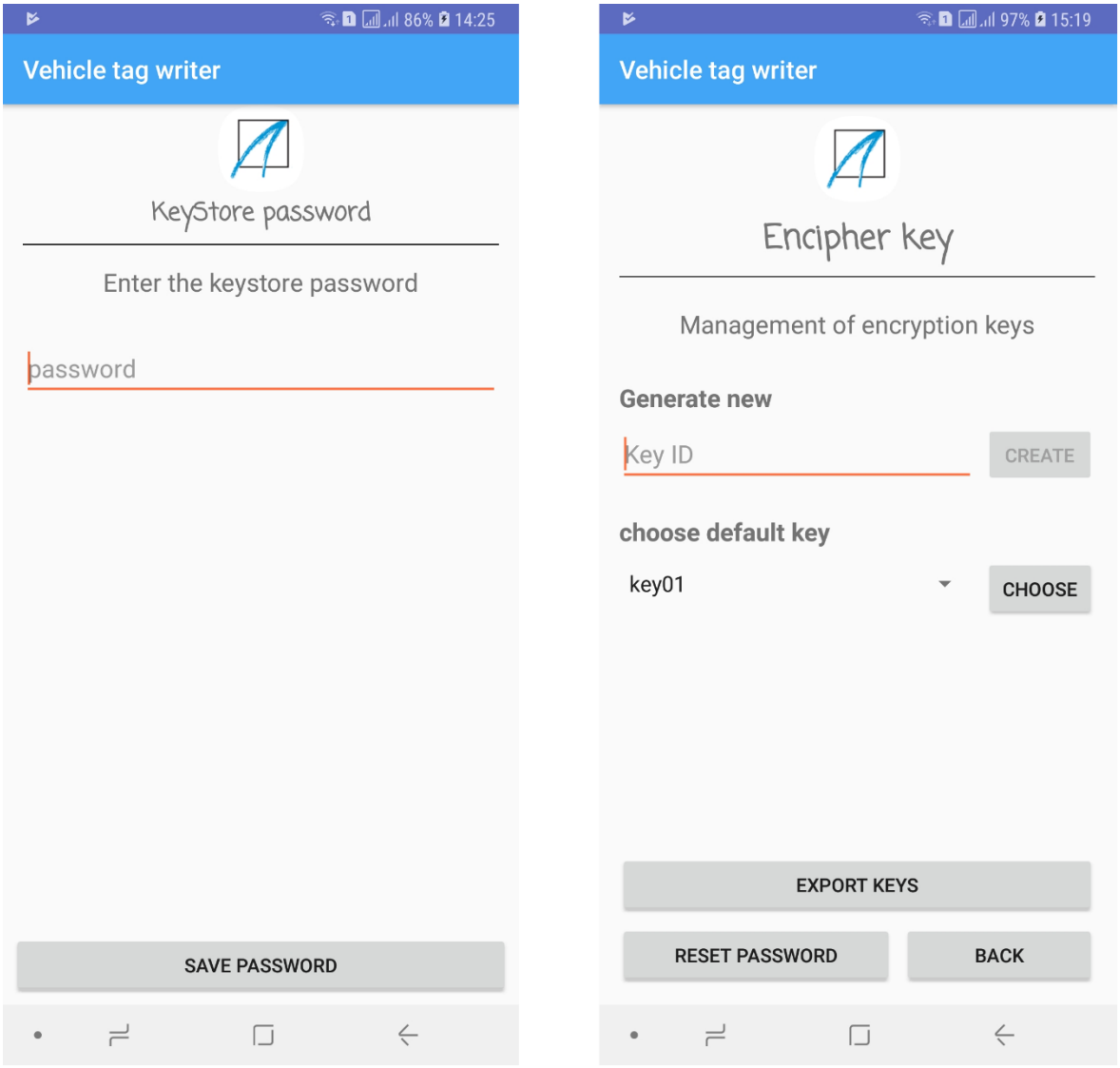
```
public static SecretKey generateAesKey(String identifier) {
  KeyGenerator keyGenerator = KeyGenerator.getInstance(
       KeyProperties.KEY_ALGORITHM_AES, ANDROID_KEY_STORE);
  keyGenerator.init(new KeyGenParameterSpec.Builder(
       AES_CIPHER_PREFIX +   identifier,
       KeyProperties.PURPOSE_ENCRYPT |
       KeyProperties.PURPOSE_DECRYPT)
    .setBlockModes(KeyProperties.BLOCK_MODE_GCM)
    .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_NONE)
    .setKeySize(128)
    .build());
  return keyGenerator.generateKey();
}
```

4.5.3 Klíčové úložiště pro export

V tomto úložišti jsou ukládány všechny AES klíče generované za účelem šifrování druhého záznamu zprávy zapisované na tag. Uložen je zde také certifikát pro verifikaci integrity tohoto zašifrovaného záznamu.

Klíčové úložiště pro export je spravováno třídou KeyStoreUtil. Konstruktor inicializuje vše potřebné pro správu tohoto úložiště. V konstruktoru třída obdrží soubor pro klíčové úložiště a přístupové heslo. Jestliže úložiště dosud neexistuje, je vytvořeno a následně je inicializován generátor AES klíčů.

```
public KeyStoreUtil(File keyStoreFile, String password) {
  this.keyStoreFile = keyStoreFile;
  this.password = password.toCharArray();
  try {
      keyStore = KeyStore.getInstance(KeyStore.getDefaultType());
      if (!keyStoreFile.exists()) {
          keyStoreFile.createNewFile();
          RsaUtil.generateRsaKeyPair();
          Certificate certificate = RsaUtil.getCertificate();
          keyStore.load(null);
          saveRsaCertificate(certificate, RSA_DEFAULT_ALIAS);
      } else {
        try (FileInputStream fis =
                              new FileInputStream(keyStoreFile)) {
          keyStore.load(fis, this.password);
        }
      }
  } catch (Exception e) {
      e.printStackTrace();
  }
  try {
    keygen = KeyGenerator.getInstance("AES");
      keygen.init(128);
  } catch (NoSuchAlgorithmException e) {
      e.printStackTrace();
  }
}
```



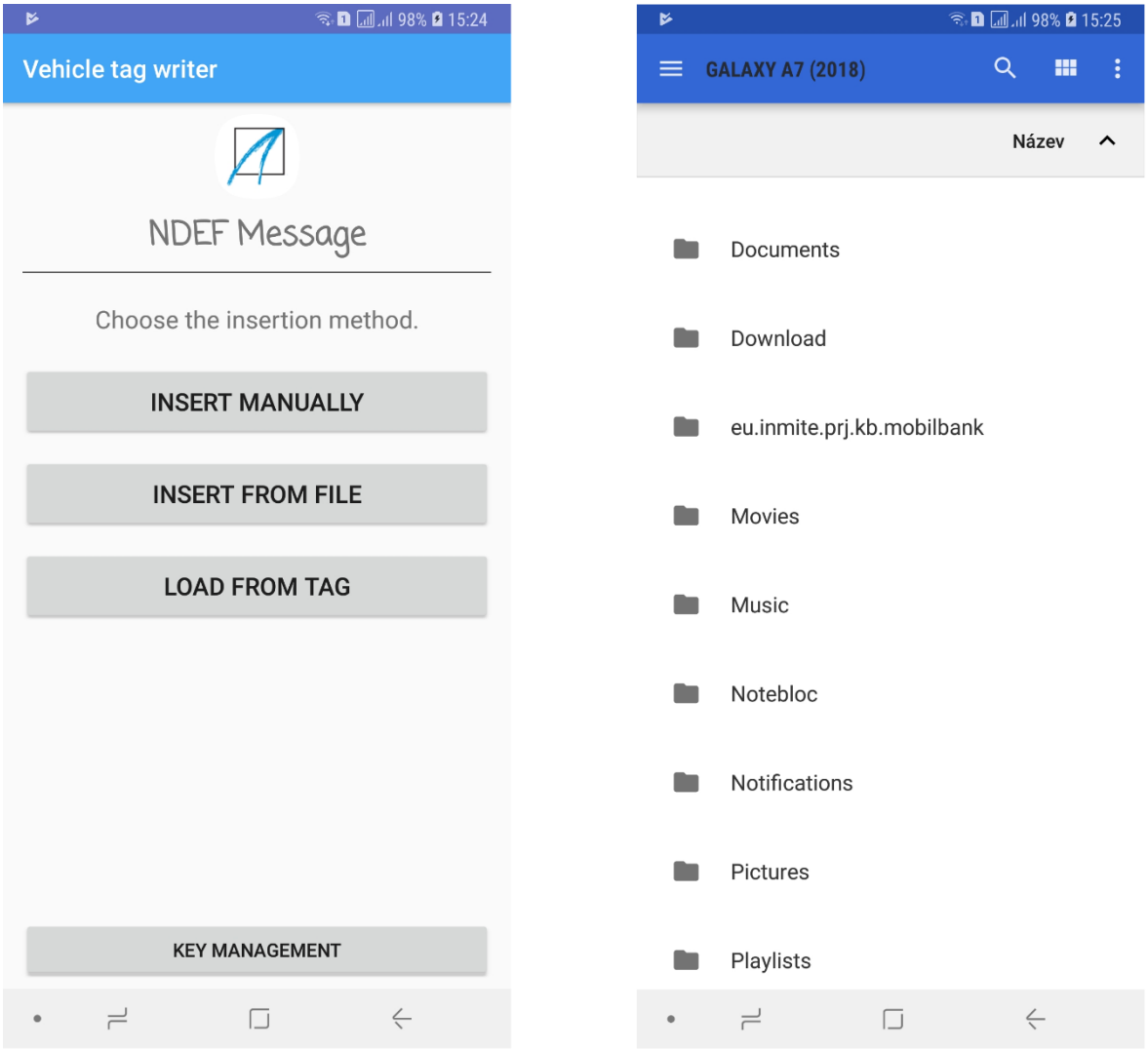
Obrázek 13: Obrazovky pro správu úložiště klíčů

4.5.4 Volba způsobu tvorby tagu

Hlavní nabídka pro rozhodování uživatele o způsobu vytvoření konfigurace je zobrazena v levé části obrázku 14. Kromě této volby je umožněno vrátit se ke správě klíčového úložiště. Obrazovka na pravé straně ukazuje souborový prohlížeč, který je uživateli zobrazen, pokud zvolí tvorbu konfigurace ze souboru. Po načtení souboru je uživateli zobrazen náhled načteného obsahu. Zde se bude uživatel moci rozhodnout mezi přímým zápisem na tag, nebo až po provedení úprav načtené konfigurace.

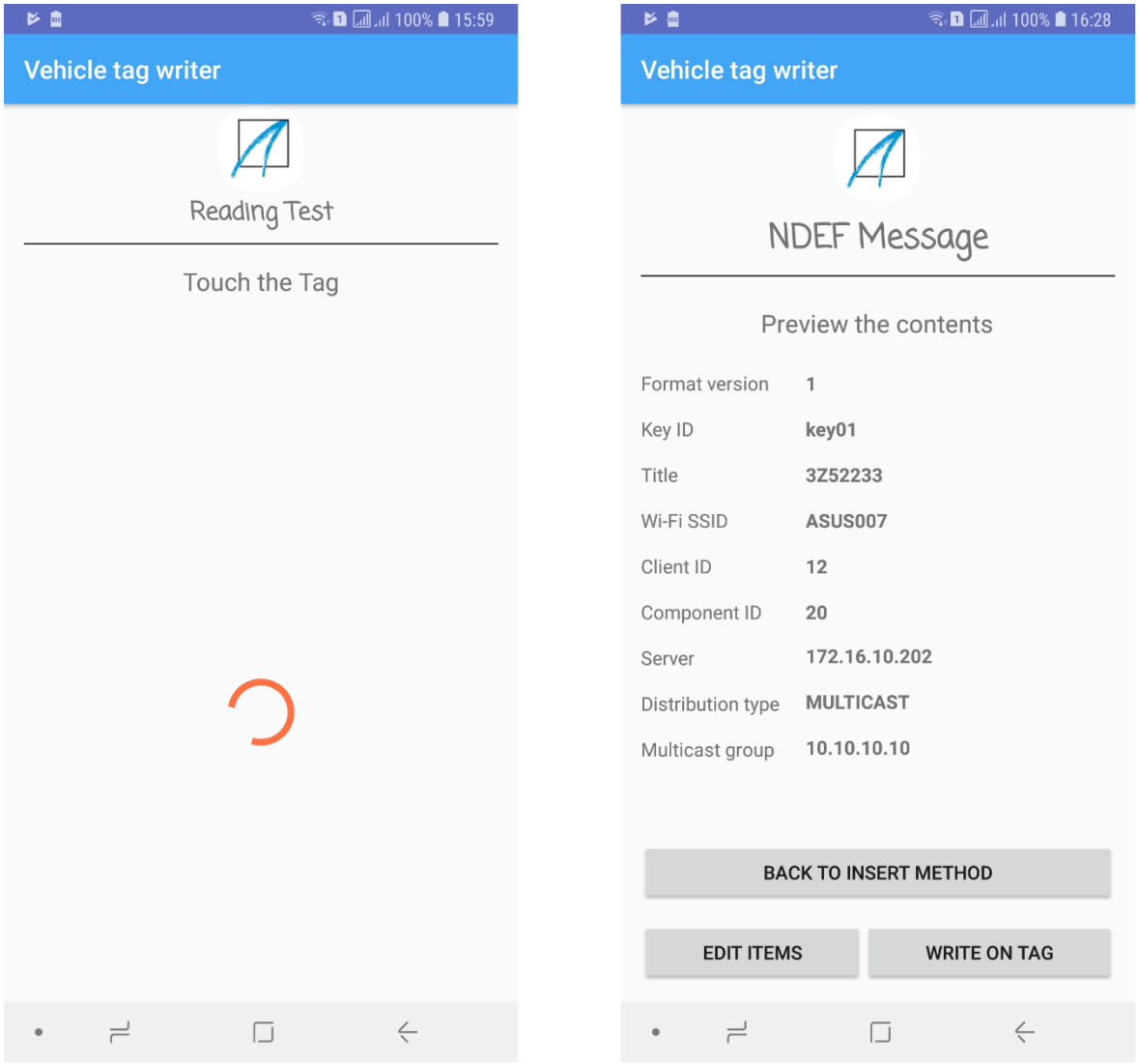
Obrazovka s náhledem načteného obsahu je zobrazena na pravé straně obrázku 15. Pokud se uživatel rozhodne před zápisem na tag provést úpravy této konfigurace, bude přesměrován na formulář, ve kterém mu bude umožněno změny realizovat.

Formulář je zobrazen na obrázku 16. Existuje zde také možnost opustit tento způsob zadávání a vrátit se zpět na hlavní stránku uvedených voleb s možností provést jinou volbu.



Obrázek 14: Volba tvorby tagu ze souboru a načtení z tagu

Na obrázku 15 je na levé straně vidět, jak vypadá návrh obrazovky pro načítání konfigurace z tagu. Uživatel prostřednictvím volby může vytvořit novou konfiguraci nebo ověřovat správnost předchozího zápisu na tag.



Obrázek 15: Načítání konfigurace z tagu a náhled načteného obsahu

Po obdržení NFC záměru je vytvořena instance třídy NfcIntentReader, které je v konstruktoru předán aktuální kontext a získaný Nfc záměr. Objekt této třídy se postará o všechny kroky potřebné k získání instance třídy MessageWrapper, která je přepravkou všech již dešifrovaných položek tagu.

V první řadě se uskuteční kontrola obsahu, tedy že se jedná o NDEF zprávu obsahující právě tři záznamy. Poté následuje postupné zpracování jednotlivých záznamů. Během celého procesu zpracování je při jakémkoli neúspěchu informován uživatel pomocí zpráv, což je provedeno pomocí kontextu volající aktivity, která pověřila třídu NfcIntentReader zpracováním záměru.

Nejdříve je zpracován záznam "header" obsahující nešifrovanou část obsahu zprávy. Po ověření správnosti typu zprávy (TNF_WELL_KNOWN / RTD_TEXT) je na základě položky "keyId" vyhledán odpovídající klíč k dešifrování druhého záznamu. Celý záznam je poté převeden na objekt třídy HeaderRecordWrapper.

Proces pokračuje zpracováním druhého šifrovaného záznamu. Jako první je provedena kontrola integrity obsahu voláním metody verify, které je předáno užitečné zatížení záznamu, obsah položky "sign" z prvního záznamu a identifikátor certifikátu.

```
public boolean verify(byte[] data, byte[] signature,
                              String identifier) throws Exception {
    Signature verifySignature =
                        Signature.getInstance(SIGNATURE_ALGORITHM);
    verifySignature.initVerify(getCertificate(identifier));
    verifySignature.update(data);
    return verifySignature.verify(signature);
}
```

Metoda vrátí boolean hodnotu informující o úspěšnosti ověření. Jestliže je vrácena hodnota true, bylo ověření úspěšné a pokračuje se dešifrováním samotného obsahu záznamu. Ověření přítomnosti potřebného klíče proběhlo již při zpracování prvního záznamu, proto se přistoupí přímo k dešifrování. Tuto činnost má na starosti třída CipherUtil a její metoda decipher. Jako argument je jí předán šifrovaný obsah, inicializační vektor a šifrovací klíč.

```
public byte[] decipher(byte[] cipherMessage, byte[] iv,
                              SecretKey key) throws Exception {
    final Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
    cipher.init(Cipher.DECRYPT_MODE, key,
                                       new GCMParameterSpec(128, iv));
    byte[] plainBytes= cipher.doFinal(cipherMessage);
    return plainBytes;
}
```

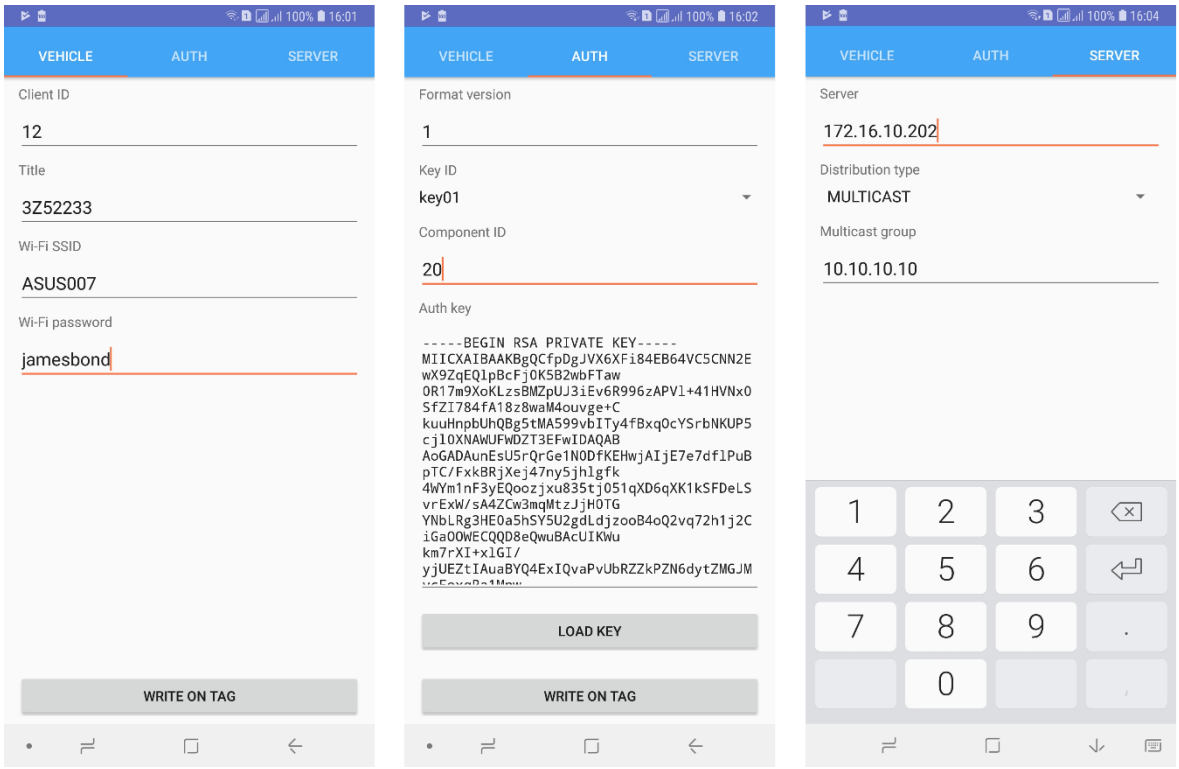
Její návratovou hodnotou je pole bajtů s dešifrovaným obsahem. Po jeho převedení do textové podoby je z něj vytvořena instance třídy BodyRecordWrapper, která je obálkou tohoto záznamu.

Následuje zpracování posledního záznamu, z nějž získáme textovou hodnotu názvu balíčku cílové aplikace, pro kterou jsou NFC tagy vytvářeny.

Z takto zpracovaných záznamů (HeaderRecordWrapper, BodyRecordWrapper, aarPackageName) je vytvořen objekt třídy MessageWrapper, který je vrácen aktivitě ReadTagActivity, jež o zpracování záměru požádala.

Po úspěšném načtení je opět zobrazen náhled načteného obsahu se stejnými volbami dalšího postupu, jak je tomu v případě načítáni konfigurace ze souboru.

Poslední volbou je vytvoření konfigurace pomocí formuláře. Návrh obrazovek pro volbu je vidět na obrázku 16. Formulář je zde rozdělen do tří obrazovek, které seskupují související položky. Formulář bude vytvořen tak, aby se klávesnice přepínala kontextově v závislosti na obsahu vkládaného obsahu. Na pravé straně obrázku 16 je například vidět obrazovka, kde klávesnice umožňuje zadávat pouze číslice a tečku.



Obrázek 16: Zadávání konfiguračních údajů pomocí formuláře

4.5.5 Obrazovka zápisu na tag

Posledním krokem po vytvoření konfigurace je zápis na tag. Návrh obrazovky je zobrazen na obrázku 17.

Popis postupu zápisu NDEF zprávy a způsob vytváření jednotlivých záznamů, ze kterých se zpráva skládá, je již popsán v kapitole 4.4. Je ovšem ještě potřeba z předaných údajů vytvořit obsah těchto zpráv.

Jako první je zašifrován Json formát objektu BodyRecordWrapper. Zašifrování se provede pomocí volání metody encipher třídy CipherUtil. Metodě je předán řetězec a šifrovací klíč, kterým má být obsah zprávy zašifrován.

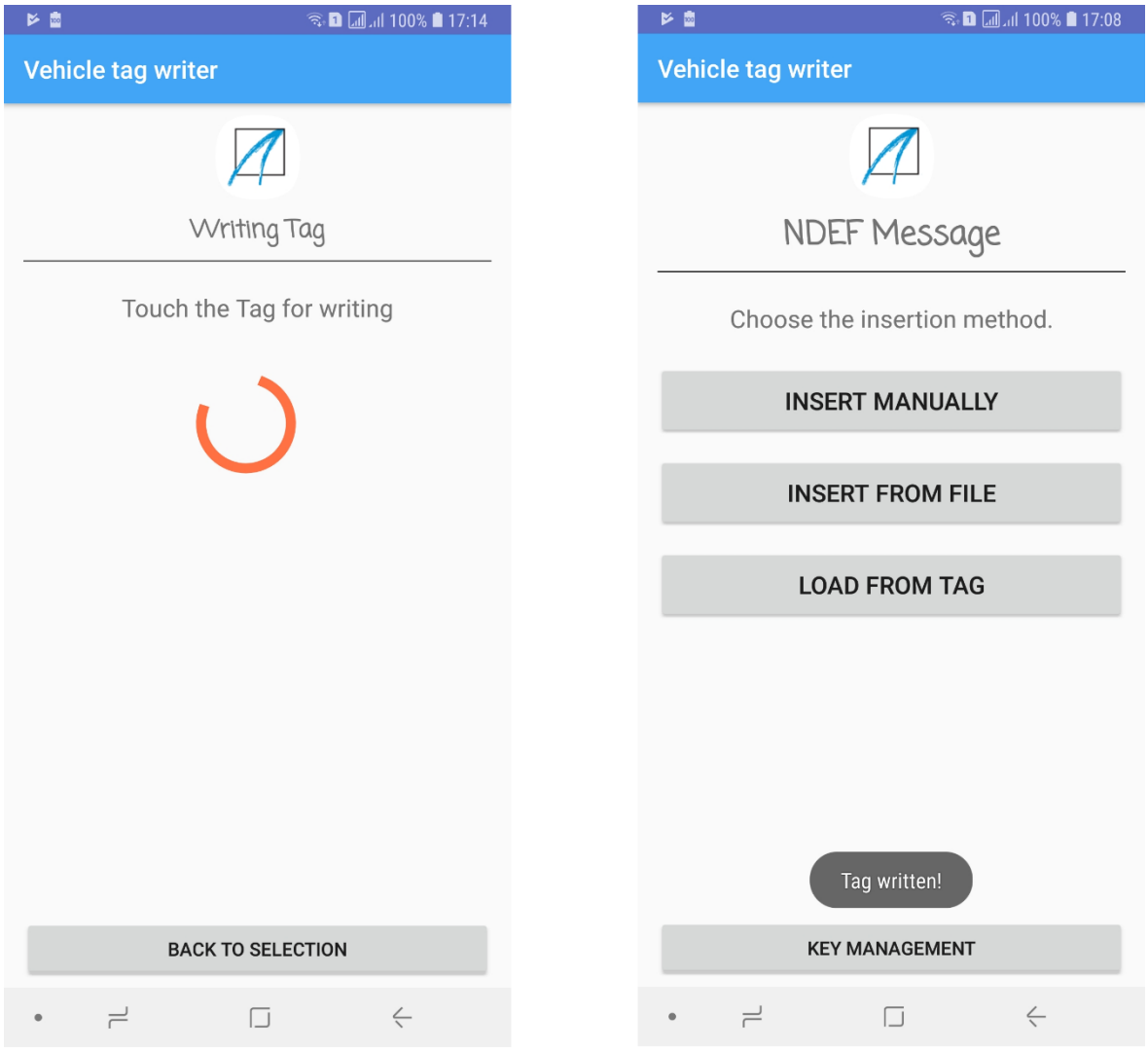
```
public byte[] encipher(byte[] plainText, SecretKey key) {
    Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
    cipher.init(Cipher.ENCRYPT_MODE, key);
    byte[] ciphertext = cipher.doFinal(plainText);
    iv = cipher.getIV();
    return ciphertext;
}
```

Následně je do proměnné "iv" objektu HeaderRecordWrapper vložena hodnota inicializačního vektoru, který byl použit při šifrování. Dále je vložena hodnota do proměnné "sign", jež obsahuje digitální podpis šifrovaného obsahu. Podpis se provede voláním metody sign třídy RsaUtil pomocí privátního klíče z interního úložiště, jehož generování je popsáno v kapitole 4.4.2.

```
public static byte[] sign(byte[] data, String identifier) {
    KeyStore ks = KeyStore.getInstance(RsaUtil.ANDROID_KEY_STORE);
    ks.load(null);
    KeyStore.Entry entry = ks.getEntry(
       RsaUtil.RSA_CIPHER_PREFIX + identifier, null);
    if (!(entry instanceof KeyStore.PrivateKeyEntry)) {
        Log.w(LOG_CIPHER, "Not an instance of a PrivateKeyEntry");
    }
    Signature signSignature =
       Signature.getInstance(SIGNATURE_ALGORITHM);
    signSignature.initSign(((KeyStore.PrivateKeyEntry) entry)
       .getPrivateKey());
    signSignature.update(data);
    return signSignature.sign();
}
```

Položky "iv" i "sign" jsou před uložením převedeny do formátu Base64, ve kterém jsou zapisovány na tag, což je posledním krokem přípravy dat pro tvorbu záznamů. Následuje tedy jejich vytvoření, vložení do NDEF zprávy a její zápis na NFC tag.

Pokud je zápis úspěšně proveden, je o této skutečnosti uživatel informován prostřednictvím zprávy o úspěšném zápisu. Uživateli je zobrazena obrazovka s volbou způsobu tvorby další konfigurace a možností správy klíčového úložiště.



Obrázek 17: Zápis na tag

Úspěšným zápisem konfigurace na tag úkol aplikace končí.

5 IMPLEMENTACE KLIENTSKÉ APLIKACE

Klientská aplikace je druhou aplikací, kterou se tato práce zabývá. Klientská aplikace bude číst obsah NFC tagů vytvořených první aplikací, která tento obsah vytváří a zapisuje na NFC tagy.

5.1 Požadavky na aplikaci

Systém TEVOGS musí umožnit, aby vnitřní mobilní klientská stanice byla přenositelná mezi jednotlivými vozy s vnějším mobilním vybavením. Konfigurace bude probíhat automaticky podle aktuálního umístění. Jednotliví uživatelé budou mít každý vlastní mobilní telefon nebo tablet, který si budou do vozu přinášet (případně přenášet mezi různými vozy). Jednotlivé vnitřní mobilní jednotky automaticky detekují umístění v konkrétním voze a upraví podle toho umístění svoji konfiguraci. Jednotky se tedy ve stejném voze chovají shodně (mají stejnou konfiguraci).

Aplikace bude fungovat ve dvou režimech (Permanent a Mobility). Při prvním spuštění se určí, kterým režimem má proběhnout konfigurace. Aplikace bude volbu ukládat a při dalších spuštěních již začne pracovat ve dříve zvoleném režimu.

Rozdíl mezi režimy je následující:

PERMANENT: Uživatel vloží nezbytné údaje pro konfiguraci prostřednictvím formuláře a aplikace se na jejich základě pokusí provést konfiguraci. Konfigurace je opět uložena, aby při dalším spuštění byl přeskočen krok vkládání konfiguračních údajů a aplikace začala rovnou provádět konfiguraci z uložených údajů.

MOBILITY: Při volbě tohoto režimu bude konfigurace vytvářena na základě údajů načtených z tagu. Obsah načteného tagu bude ukládán, aby bylo možné místo načítání konfigurace z tagu zvolit i jakoukoli již dříve načtenou konfiguraci.

Dále aplikace musí poskytnout uživateli možnost vložení úložiště klíčů, které mu poskytne vydavatel tagů. Úložiště klíčů je nezbytné při načítání konfigurace z tagu z důvodu dešifrování záznamu a pro ověření integrity jeho obsahu.

5.2 Návrh funkcionalit vycházejících z požadavků

Aplikaci bude možné spouštět přiložením tagu, kdy pomocí AAR záznamu bude aplikace upřednostněna před ostatními aplikacemi, které jsou z pohledu operačního systému Android schopné tento záměr také zpracovávat.

5.2.1 Návrh posloupnosti operací

Po nainstalování aplikace nejsou uloženy žádné údaje, na jejichž základě by byl zvolen režim (Mode), ve kterém se má aplikace spouštět. Při prvním spuštění bude tedy nutné nejdříve zvolit a uložit režim, ve kterém bude aplikace startovat při dalším spuštění.

Pokud bude zvolen režim MOBILITY, proběhne kontrola přítomnosti klíčového úložiště a příslušného hesla. Jestliže dosud nebylo klíčové úložiště přiřazeno, bude nutné uživateli umožnit jeho vložení včetně zadání hesla, jehož prostřednictvím získá k úložišti přístup.

Poté bude uživatel vyzván k přiložení tagu s konfigurací. Zároveň bude uživateli nabídnuta volba k výběru z předchozích konfigurací, které byly již dříve z tagů načteny. Jakmile uživatel zvolí jednu z nabízených možností, provede aplikace verifikaci načteného obsahu a jeho dešifrování. Pokud se tyto operace uskuteční úspěšně, proběhne ještě validace načteného a dešifrovaného obsahu. Jestliže je obsah validní, načtená konfigurace se uloží pro možnost jejího výběru při dalším spuštění vedle volby načítání z tagu. Posledním krokem bude sestavení konfigurace, po jehož úspěšném provedení úloha aplikace končí.

Při volbě režimu (Mode) PERMANENT se existence klíčového úložiště nevyžaduje, protože v tomto režimu nedochází k načítání konfigurace z tagu, tedy nedochází ani k žádnému dešifrování, a proto není toto úložiště potřebné. Je ovšem nutné zadat nezbytné údaje pro konfiguraci, a proto bude uživateli při prvním spuštění aplikace zobrazen formulář, ve kterém nezbytné údaje vyplní. Opět proběhne validace údajů a následná konfigurace. Vložené údaje budou uloženy pro potřebu při dalším spuštění, kdy již nebude nutné tyto údaje opětovně zadávat.

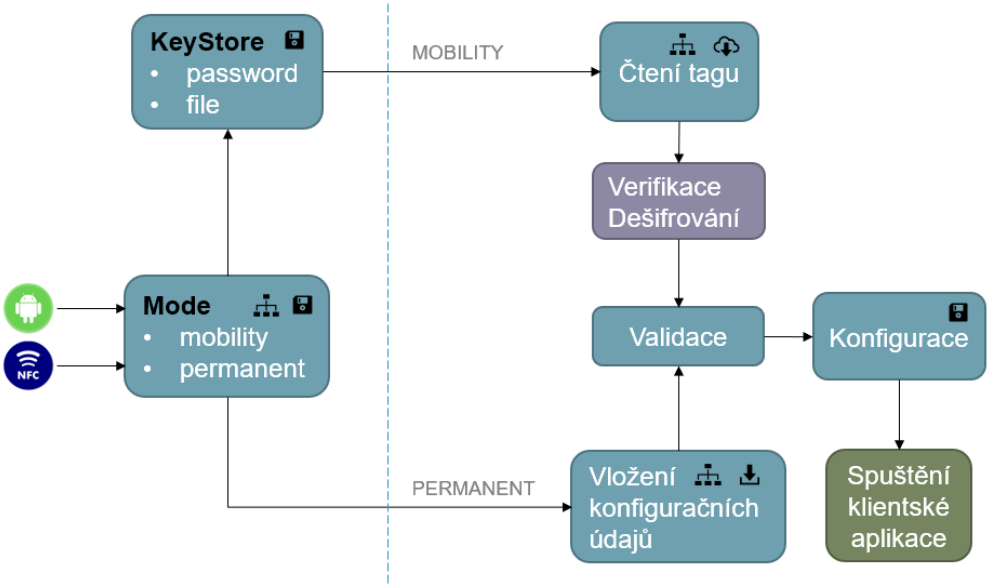
Druhé a každé další spuštění aplikace již bude vycházet z voleb uložených při prvním spuštění.

V režimu PERMANENT dojde ke konfiguraci bez nutnosti jakéhokoli zásahu uživatele, protože vše potřebné je již uloženo po prvním a úspěšném spuštění.

V režimu MOBILITY závisí průběh na tom, v jakém okamžiku byl tag přiložen. Jestliže není aplikace v zařízení spuštěna a tag je přiložen, pak vše opět proběhne bez nutnosti zásahu uživatele. V druhém případě, kdy je aplikace nejdříve spuštěna, je uživatel vyzván k přiložení tagu nebo k výběru konfigurace z dříve načtených tagů.

Spuštění aplikace přiložením tagu je funkční i při uloženém režimu PERMANENT, přičemž konfigurace proběhne na základě již dříve vytvořené konfigurace v tomto režimu. Samotné přiložení tagu způsobí pouze spuštění aplikace, ale jeho obsah není již ke konfiguraci použit.

Uvedený životní cyklus aplikace je zobrazen na obrázku 18.



Obrázek 18: Životní cyklus aplikace

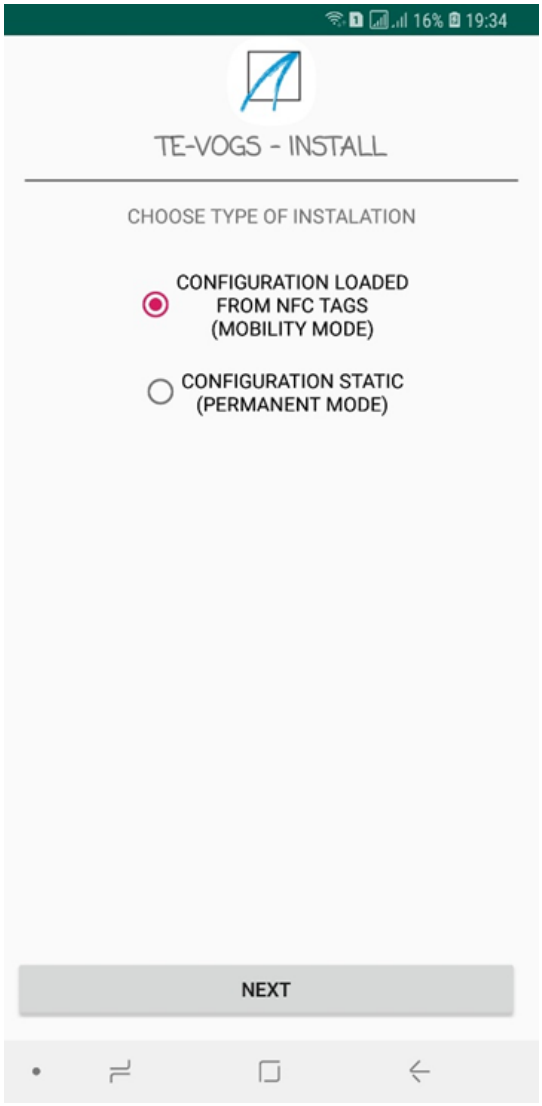
5.3 GUI, aplikační logika

Jednotlivým krokům z předchozí kapitoly je nutné poskytnout grafické uživatelské prostředí.

5.3.1 Volba režimu aplikace

Volba režimu aplikace je realizována pomocí přepínače, jak je vidět na obrázku 19. Obrazovka bude při prvním spuštění (nebo po resetu nastavení) vždy první zobrazenou obrazovkou.

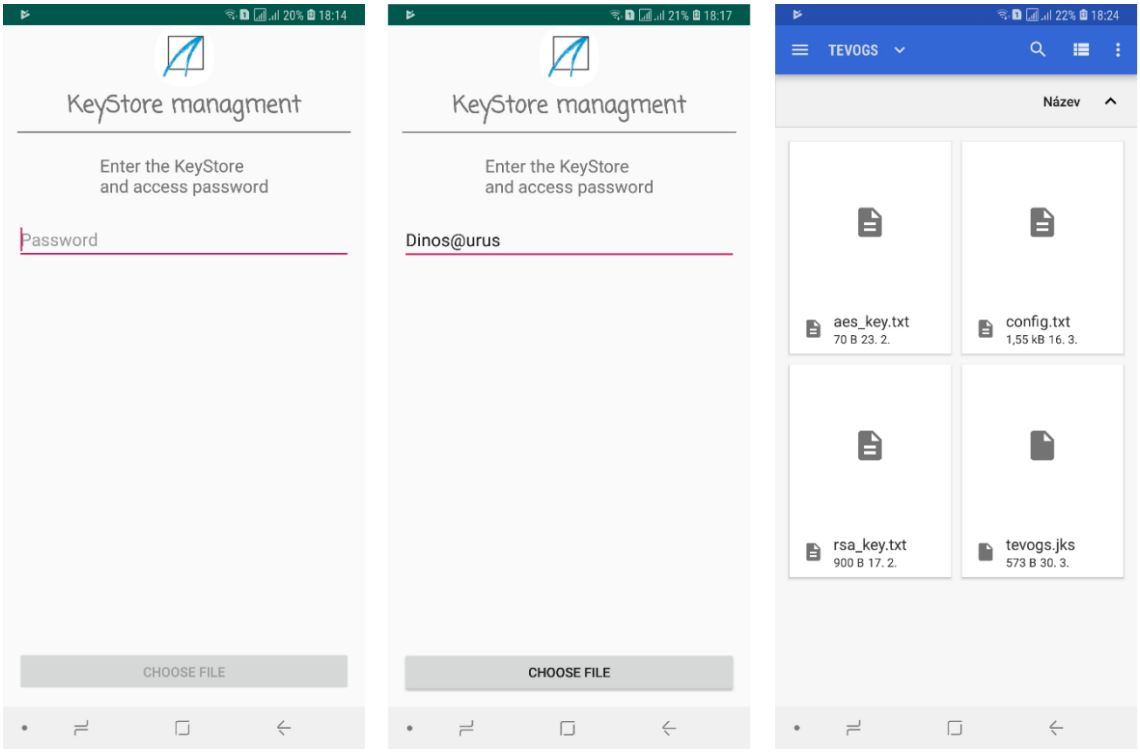
Zvolený režim je následně uložen do souboru cnm.properties pod klíčem cnm.configMode. Aplikace při každém spuštění zjišťuje, zda je tato volba uložena. Jestliže je volba uložena, aplikace přeskočí tento krok a nastaví režim dle uložené hodnoty.



Obrázek 19: Obrazovka s volbou režimu

5.3.2 Vložení klíčového úložiště

Obrazovky na obrázku 20 zobrazují postup při vkládání klíčového úložiště a přístupového hesla.



Obrázek 20: Postup vložení klíčového úložiště

Pokud je zvolen režim mobility, a není vloženo klíčové úložiště, zobrazí se obrazovka na pravé straně, kde je uživatel vyzván k vložení hesla. Tlačítko pro výběr souboru s klíčovým úložištěm není přístupné, dokud heslo nevyhoví validačnímu pravidlu na délku hesla obsahující osm znaků. Pokud je heslo vyhovující, tlačítko se zpřístupní a po jeho stisknutí se uživateli zobrazí souborový prohlížeč pro výběr souboru s klíčovým úložištěm.

Pro zadané heslo je vygenerován AES klíč, kterým je heslo zašifrováno a uloženo v šifrované podobě do SharedPreferences. Způsob generování a uložení klíče je řešeno Android keystore systémem stejně jako v aplikaci pro tvorbu tagů popisované v kapitole 4.5.2. Klíčové úložiště je zkopírováno do interní paměti vyhrazené aplikaci, čímž je zaručeno, že bude stále k dispozici. Dispoziční přístup by nemusel být zaručen, kdyby se klíčové úložiště načítalo z umístění při jeho výběru, kdy se do zařízení klíčové úložiště přeneslo například na SD kartě.

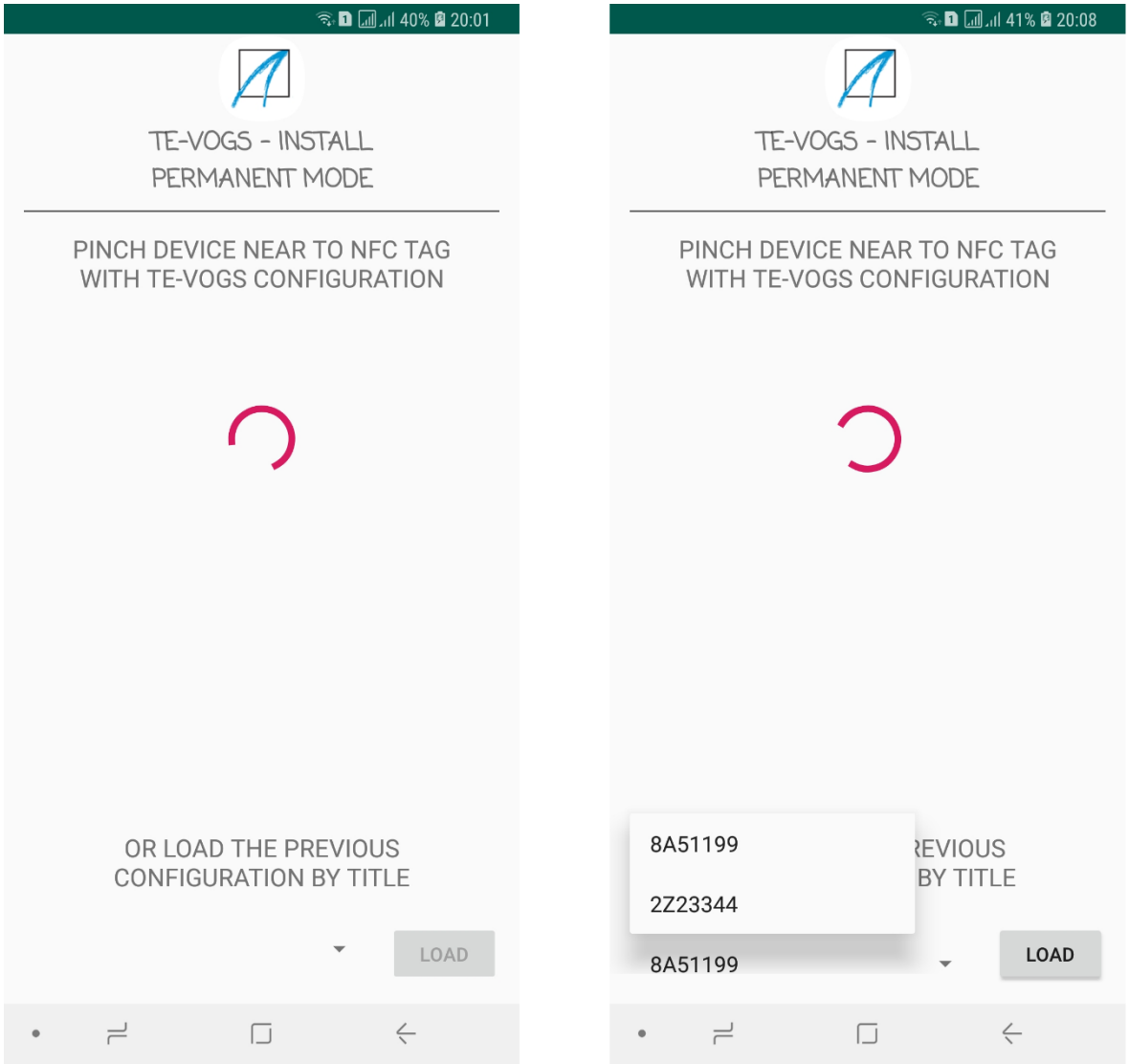
5.3.3 Mobility mód

Při tomto režimu je nutné nabídnout obrazovku s výzvou na přiložení tagu z důvodu jeho načtení. Zároveň musí obrazovka nabídnout uživateli k výběru seznam s předchozími úspěšně načtenými konfiguracemi. Použité řešení je na obrázku 21.

Obrázek na levé straně ilustruje, jak obrazovka vypadá při prvním spuštění, kdy ještě žádné předchozí konfigurace neexistují, a proto je tlačítko pro jejich načítání nepřístupné. Na pravé straně je vidět, že již proběhly dvě úspěšně načtené konfigurace a je umožněno jednu z nich použít jako alternativu k načítání z tagu.

Předchozí načtené konfigurace jsou ukládány pouze tehdy, pokud projdou verifikací a validací, aby nebyly do výběru z předchozích uložených konfigurací nabízeny nepoužitelné konfigurace vykazující chyby. Ověření je možné provést až po dešifrování, přičemž konfigurace se ukládají v šifrované podobě ve formátu, v jakém jsou zapsány na tagu.

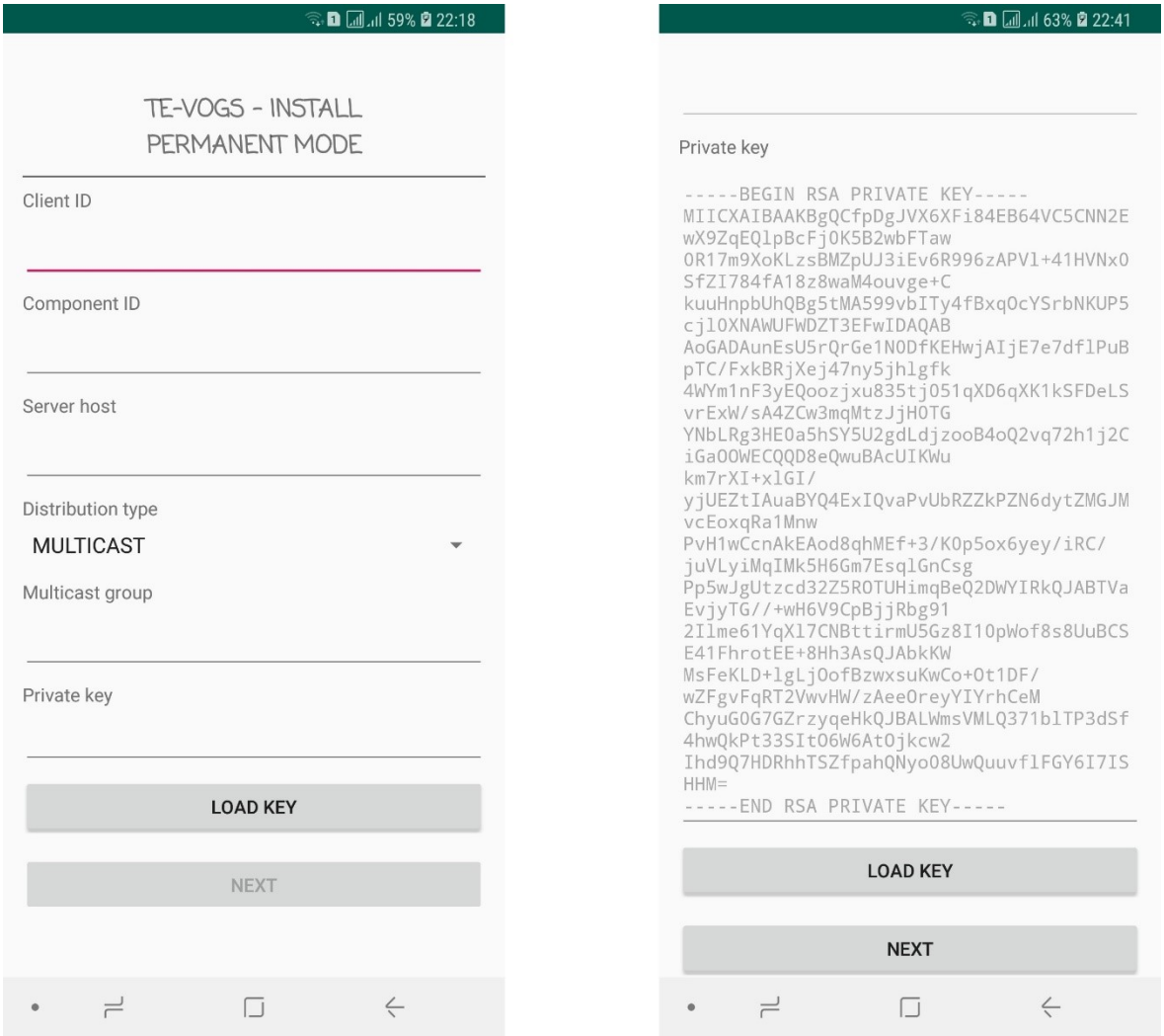
Načítání, verifikace a dešifrování probíhá stejným způsobem jako v aplikaci pro tvorbu tagů, když je vybrána volba tvorby načtením konfigurace z tagu (kapitola 4.5.4).



Obrázek 21: Mobility mód

5.3.4 Permanent mód

Tento režim nevyužívá konfiguraci z tagu, ale potřebné údaje jsou zadány do aplikace pomocí formuláře. Tento formulář je zobrazen na obrázku 22. Formulář všechny zadané údaje validuje již v průběhu jeho vyplňování. Dokud nejsou vloženy veškeré údaje dle validačních pravidel, není možné tato data odeslat, protože tlačítko pro jejich odeslání zůstává nepřístupné.



Obrázek 22: Permanent mód

5.3.5 Zpracování obdržených dat

Provedení konfigurace se liší v závislosti na spuštěném režimu aplikace.

Při režimu PERMANENT nejsou prostřednictvím formuláře zadávány údaje o připojení ke konkrétní Wi-Fi. Připojení k požadované Wi-Fi provádí uživatel sám v nastavení

systému Android. Údaje obdržené v tomto módu slouží pouze k navázání spojení a nastavení komunikace se serverem.

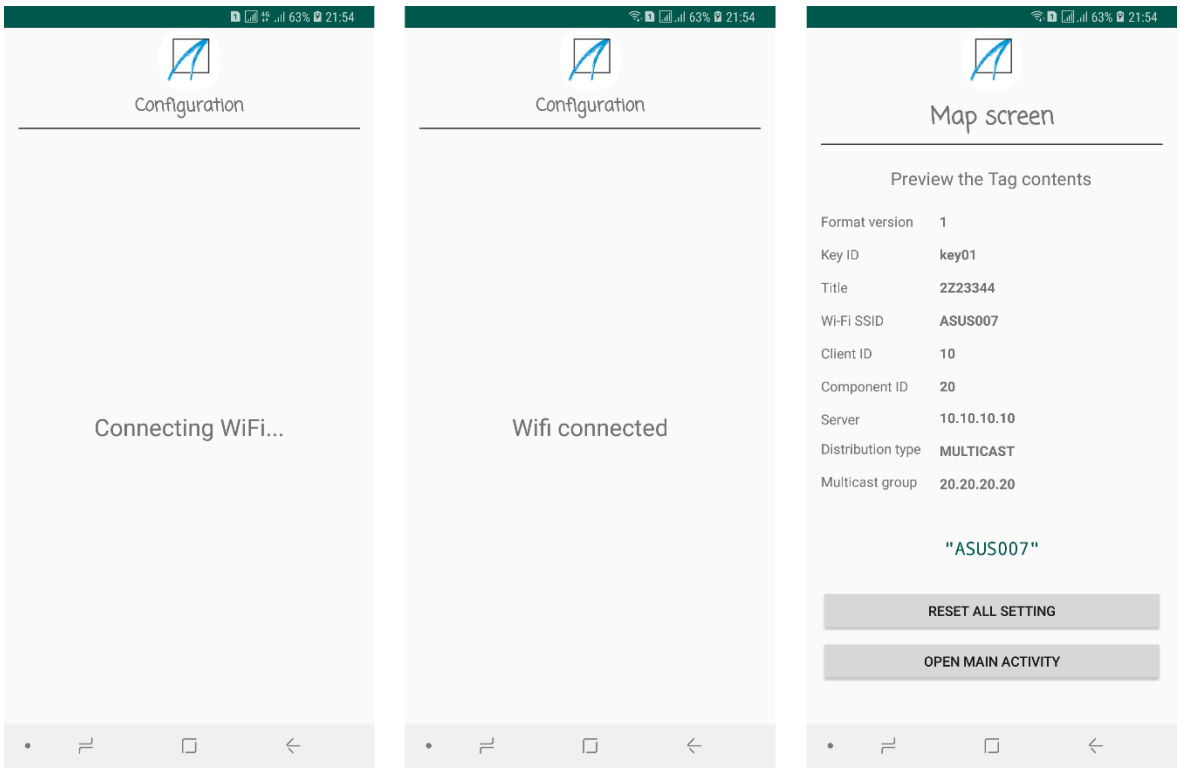
V režimu MOBILITY je definována i konkrétní Wi-Fi síť, ke které má být zařízení během běhu aplikace připojeno. Připojení musí být po celou dobu vynucováno, a pokud se připojení nepodaří navázat, nebo je z nějakého důvodu přerušeno, nesmí se zařízení připojit k žádné jiné Wi-Fi síti.

K tomuto účelu slouží v aplikaci třída WifiReceiver, která je potomkem třídy BroadcastReceiver. Tento přijímač je inicializován v aktivitě, která vytváří konfiguraci na základě obdržených konfiguračních údajů. Mezi těmito údaji je i SSID a heslo preferované Wi-Fi. Přijímač je následně zaregistrován s filtrem záměrů, které jsou vyvolány při změnách na Wi-Fi připojení. Detailní informace o stavu, respektive změně v připojení, lze získat ze záměru v podobě instance třídy NetworkInfo. Obdržené informace využívá objekt třídy WifiReceiver ke kontrole preferovaného připojení udržovaného v třídě EnforcedWifi. Pokud WifiReceiver zjistí, že se o preferované připojení nejedná, je volána metoda enforcedWifi této třídy, jejíž implementace se postará o připojení k Wi-Fi definované v konfiguraci. Po úspěšném připojení k preferované Wi-Fi je o této skutečnosti informována aktivita starající se o konfiguraci, protože její další činnost je prováděna až poté co tak nastane. Další činností je navázání komunikace se serverem a nastavení komunikace. Činnost v této aplikaci není zahrnuta, přičemž je již implementována ve stávající aplikaci, které vyvíjené rozšíření poskytuje potřebné údaje. Dle požadavků zadavatele je úspěšné a trvale vynucované připojení k preferované síti posledním krokem, kterým úkol této aplikace končí.

Aby však bylo možné aplikaci reálně otestovat, je samotné obdržení nezbytných a validních údajů potřebných pro toto spojení považováno za připojení k serveru a aplikace dále pokračuje na aktivitu, která představuje spuštění klientské aplikace s navigační mapou.

Místo mapy jsou zde zobrazeny údaje, jež byly aplikací načteny a dešifrovány, včetně SSID Wi-Fi, ke které je zařízení aktuálně připojeno. Z této aktivity je také možné resetovat všechna uložená nastavení a přesměrování zpět na hlavní aktivitu. Patrný je prostor pro testování, zdali je aktivně vynucováno permanentní připojení k preferované Wi-Fi, nebo zdali a jakým způsobem proběhne konfigurace, pokud již běžící nakonfigurovaná aplikace obdrží nový NFC záměr.

Grafické uživatelské rozhraní probíhající konfigurace a obrazovka zobrazující její výsledky jsou ilustrovány na obrázku 23.



Obrázek 23: Konfigurace

6 TESTOVÁNÍ APLIKACÍ

6.1 Aplikace pro tvorbu tagů

Ověření spočívalo v testování posloupnosti operací, jestli na sebe navazují v pořadí, jak bylo navrženo, aby se aplikace neocitla v nekonzistentním stavu. Při nepřítomnosti klíčového úložiště aplikace správně startovala na aktivitě, kde je uživatel vyzván k zadání hesla a k vygenerování AES klíče pro šifrování obsahu druhého záznamu. Generování klíčů, validace délky jejich identifikátoru, volba defaultního klíče, validace minimální délky hesla, jeho změna i export úložiště do externího souboru probíhalo v souladu s návrhem a bezchybně.

Poté byly testovány jednotlivé režimy tvorby obsahu tagu, validace jednotlivých položek a umožnění zápisu pouze pokud validačním pravidlům vyhovovaly všechny položky. U voleb tvorby načítání z tagu nebo ze souboru bylo také ověřeno správné předvyplnění položek formuláře, který je zobrazen, jestliže si uživatel přeje provést úpravy položek před jejich zápisem. Ve všech případech bylo ověřeno, že komponenta výběru šifrovacího klíče obsahuje všechny klíče, jež byly vygenerovány. Samotný zápis pro všechny přístupy proběhl úspěšně, včetně zápisu na tagy, které nebyly před zápisem naformátované.

Díky možnosti tvorby tagu (z již existujícího tagu) proběhla zároveň kontrola načítání aplikací vytvořených tagů a jejich dešifrování, které bylo ve všech případech úspěšné.

V rámci těchto testů byly vytvořeny rovněž tagy, u kterých byl těsně před zápisem modifikován obsah druhého šifrovaného záznamu z důvodu ověření správné funkce algoritmu zajišťujícího kontrolu integrity. U všech modifikovaných tagů algoritmus správně vyhodnotil nesoulad s podpisem uloženým v prvním záznamu v položce "sign".

6.2 Konfigurace klientské aplikace

Při testování této aplikace byl kladen důraz na ověření životního cyklu v závislosti na tom, zda se jedná o první spuštění, nebo následné spuštění s již uloženými údaji o módu aplikace. První spuštění, nebo spuštění po resetu všech nastavení, správně zobrazovalo formulář s volbou módu, ve kterém má být aplikace spuštěna.

Po volbě permanent byla ověřena správná funkce validačních pravidel a uložení zadané konfigurace, která byla při dalším spuštění použita již bez nutnosti zásahu uživatele.

U volby módu mobility při prvním spuštění aplikace správně detekovala nepřítomnost klíčového úložiště a uživatel byl vyzván k jeho vložení včetně zadání přístupového hesla. Kromě validace délky hesla bylo testováno, zdali aplikace upozorní uživatele na skutečnosti, které by znemožnily další správné fungování aplikace. Jedná se o správnost hesla, přítomnost certifikátu pro verifikaci obsahu tagu, a zdali klíčové úložiště obsahuje minimálně jeden AES klíč nezbytný pro dešifrování obsahu druhého záznamu. Všechny testy spojené s klíčovým úložištěm proběhly úspěšně.

Dalším krokem bylo ověření načítání, verifikace a dešifrování obsahu načteného tagu. Zde došlo pouze k zopakování testů provedených na první aplikaci s tím rozdílem, že byla ověřena správná funkce možnosti načíst konfiguraci z již dříve načtených a uložených konfigurací. První bylo tedy kontrolováno, zdali k ukládání vůbec dochází a zda jsou všechny uložené konfigurace uživateli nabídnuty k výběru. Poté bylo ověřeno, jestli správně probíhá verifikace a dešifrování i z těchto souborů, protože jsou ukládány do zařízení v šifrované podobě. Všechny testy spojené s načítáním byly úspěšné.

Následně bylo nutné ověřit, zda dochází k připojení k příslušné Wi-Fi síti, která je definována v načtené konfiguraci, a jestli je toto připojení permanentně vynucováno. Za tímto účelem byly vytvořeny tagy s konfigurací pro různé Wi-Fi sítě. Aplikace vždy zajistila připojení zařízení k Wi-Fi síti definované v konfiguraci. Pokud tato síť nebyla právě k dispozici, bylo zařízení odpojeno od aktuálně připojené sítě. Také bylo ověřeno, že tak učiní pokaždé, když se operační systém pokusí zařízení připojit k síti, která není definována v načtené konfiguraci. Po obnovení signálu preferované sítě aplikace zajistila, že k ní bylo zařízení připojeno. Stejné chování aplikace vykazovala i při ztrátě a obnovení preferovaného připojení.

Poslední kontrolou byla možnost resetu nastavení módu aplikace a odstranění údajů pro konfiguraci v režimu permanent. Ověřením bylo zjištěno, že je reset předvoleb funkční a aplikace po něm nabíhá na aktivitě volby módu, jako by se jednalo o první spuštění.

ZÁVĚR

Cílem diplomové práce bylo vytvořit nové řešení mobilní aplikace na platformě Android. Aplikace ke svému správnému fungování potřebuje při spuštění provést konfigurační nastavení, které je vždy rozdílné pro jednotlivá spuštění. Požadavkem zadavatele bylo využití NFC technologie jako zdroj konfiguračních údajů a nástroj pro jejich přenos do mobilní aplikace.

Nejdříve bylo nutné analyzovat požadavky vyvíjeného rozšíření, a poté na jejich základě navrhnout vhodné řešení. Práce je ve své teoretické části zaměřená na možnosti, které nabízí platforma Android v souvislosti s NFC technologií. Jsou zde uvedeny zejména preferované oblasti použité v praktické části práce. Pro dosažení primárního cíle práce, tedy načítání konfigurace z NFC tagu, bylo nutné první vytvořit aplikaci, která tyto tagy vytváří. Dále pak definovat obsah tagu a zvolit jeho formát s ohledem na požadavek šifrování citlivých položek. Do této aplikace bylo nutné zahrnout i správu šifrovacích klíčů a umožnit jejich export pro potřeby aplikace načítající jejich obsah.

V praktické části je popsáno uživatelské rozhraní aplikace včetně postupů, jakými lze konfiguraci vytvářet a zapisovat na NFC tag. Součásti návodu je správa klíčového úložiště a způsob exportu.

Po dokončení aplikace zapisující konfigurace se práce zabývala rozšířením stávající aplikace mobilního klienta, která umožňuje provádět prvotní konfiguraci nezbytnou pro její funkci. Načítání obsahu tagu, ověření integrity a dešifrování obsahu bylo vyřešeno a popsáno již v aplikaci pro jejich tvorbu. Práce se dále zabývala návrhem posloupnosti operací, jako je ukládání uživatelských voleb, které jsou zadávány při prvním spuštění, a dále způsobem vložení klíčového úložiště a ostatními aspekty, jež vyplývají z požadavků. Návrh a realizace včetně popisu uživatelského rozhraní je i v případě této aplikace součástí praktické části práce. Preferované detaily popisují plnění požadavků na aplikaci včetně popisu režimů, ve kterých může prvotní konfigurace probíhat.

Závěr práce popisuje testování obou aplikací, a jejich ověření, zdali plní veškeré cíle vyplývající z požadavků, a zdali pracují v souladu s návrhem, který byl na základě požadavků vytvořen. Veškeré provedené testy prokázaly, že aplikace pracují v souladu se stanovenými požadavky a na základě navrženého a zadavatelem odsouhlaseného řešení.

SEZNAM POUŽITÉ LITERATURY

[1] Aero4TE s.r.o. [online]. Praha: Aero4TE, 2015 [cit. 2019-04-07]. Dostupné z: http://www.aero4te.com/te-vogs/

[2] Scheme-1024x929. In: Aero4TE s.r.o. [online]. Praha: Aero4TE, 2015 [cit. 2019-04-07]. Dostupné z: http://www.aero4te.com/wp-content/uploads/2015/05/scheme-1024x929.jpg

[3] TEVOGS_032017 - TEVOGS_032017.pdf [online]. Praha: aero4te.com, 2017 [cit. 2019-04-07]. Dostupné z: http://www.aero4te.com/wp-content/uploads/2015/05/TEVOGS_032017.pdf

[4] Operator_center-300x179. In: Aero4te.com [online]. Praha: aero4te.com, 2015 [cit. 2019-04-07]. Dostupné z: http://www.aero4te.com/wp-content/uploads/2015/05/operator_center-300x179.jpg

[5] DSC05543.jpg. In: Aero4TE s.r.o. [online]. Praha: Aero4TE, 2016 [cit. 2019-04-07]. Dostupné z: http://www.aero4te.com/wp-content/uploads/2016/11/DSC05543.jpg

[6] DSC05606.jpg. In: Aero4TE s.r.o. [online]. Praha: Aero4TE, 2016 [cit. 2019-04-07]. Dostupné z: http://www.aero4te.com/wp-content/uploads/2016/11/DSC05606.jpg

[7] Near Field Communication. Wikipedie [online]. Wikimedia Foundation, 2019 [cit. 2019-04-13]. Dostupné z: https://cs.wikipedia.org/wiki/Near_Field_Communication

[8] NFC Forum. In: NFC Forum [online]. [cit. 2019-04-13]. Dostupné z: https://nfc-forum.org/wp-content/uploads/2016/01/nfc-logo1.png

[9] ČSN online [online]. Praha: Česká agentura pro standardizaci, 2017 [cit. 2019-04-14]. Dostupné z: https://csnonline.agentura-cas.cz/default.aspx

[20] COSKUN, Vedat, Kerem OK a Busra OZDENIZCI. Professional NFC application development for Android. Chichester, West Sussex: Wrox, a Wiley brand, [2013]. ISBN 978-1118380093.

[11] FeliCa – NFC Forum | NFC Forum . Home – NFC Forum | NFC Forum [online]. Copyright © 2018 NFC Forum [cit. 2019-04-07]. Dostupné z: https://nfc-forum.org/glossary/felica/

[12] IGOE, Tom, Don COLEMAN a Brian JEPSON. Beginning NFC: near field communication with Arduino, Android, and Phonegap. Beijing: O'Reilly, [2014]. ISBN 14-493-7206-6.

[13] NFC Forum [online]. NFC Forum, 2019 [cit. 2019-04-14]. Dostupné z: https://nfc-forum.org/what-is-nfc/what-it-does/

[14] NFC Protocol Stack – Near-field communication [online]. In: . Wikimedia Foundation [cit. 2019-04-14]. Dostupné z: https://en.wikipedia.org/wiki/Near-field_communication#/media/File:NFC_Protocol_Stack.png

[15] NFC – bezdrátová komunikace blízké budoucnosti? In: Notebook.cz [online]. Praha: Viktor Péder, 2011 [cit. 2019-04-14]. Dostupné z: https://notebook.cz/clanky/technologie/2011/nfc-bezdratova-komunikace-blizke-budoucnosti

[16] Co je NFC a k čemu ho můžete použít? Svět Androida [online]. Karel Kilián, 2019 [cit. 2019-04-14]. Dostupné z: https://www.svetandroida.cz/co-je-nfc-k-cemu-je-dobre-ho-pouzit/

[17] NFC Forum [online]. 2018 [cit. 2018-11-22]. Dostupné z: https://nfc-forum.org/what-is-nfc/about-the-technology/

[18] MERTLÍK, Tomáš Popis technologie NFC a jejího zabezpečení: semestrální projekt. BRNO: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012. 98 s. Vedoucí práce byl Ing. Martin Rosenberg

[19] SUBTIL, Victor. Near Field Communication with Android Cookbook. Birmingham: Packt Publishing, 2014. ISBN 9781783289653.

[20] Android Developers [online]. [cit. 2019-04-14]. Dostupné z: https://developer.android.com/guide/topics/connectivity/nfc/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ATC	Air Traffic Control
GNSS	Global Navigation Satellite Systém
LTE	Long Term Evolution
NFC	Near field communication
RFID	Radio frequency identification
ISO	International organization for standardization
IEC	International electrotechnical commission
PCD	Proximity coupling device
PICC	Proximity inductive coupling card
ASK	Amplitude shift keying
OOK	On-Of Keying
NRZ-L	Non Return to Zero Level
BPSK	Binary-Phase Shift Keying
FeliCa	Felicity card
NFCIP-1	Near-Field Communication Interface and Protocol 1
API	Application Programming Interface
NDEF	NFC Data Exchange Format
AAR	Android Application Record
SSID	Service Set Identifier
Wi-Fi	Wireless Fidelity
IrDA	Infrared data association

SEZNAM OBRÁZKŮ

Obrázek 1: Schéma systému [2] 11
Obrázek 2: Operátorské centrum [3] 12
Obrázek 3: Mobilní klient [3] 13
Obrázek 4: Vnější mobilní jednotka [5] 13
Obrázek 5: Vnitřní mobilní jednotka [6] 14
Obrázek 6: Logo organizace NFC Forum [8] 16
Obrázek 7: Přehled NFC protokolů [14] 17
Obrázek 8: Vlastnosti přenosu bezdrátových technologií [15] 20
Obrázek 9: Struktura NdefMessage [19] 21
Obrázek 10: Systém odesílání tagů [20] 23
Obrázek 11: Struktura zprávy NDEF [10] 24
Obrázek 12: Životní cyklus tvorby tagu. 37
Obrázek 13: Obrazovky pro správu úložiště klíčů 45
Obrázek 14: Volba tvorby tagu ze souboru a načtení z tagu 46
Obrázek 15: Načítání konfigurace z tagu a náhled načteného obsahu 47
Obrázek 16: Zadávání konfiguračních údajů pomocí formuláře 49
Obrázek 17: Zápis na tag 51
Obrázek 18: Životní cyklus aplikace 54
Obrázek 19: Obrazovka s volbou režimu 55
Obrázek 20: Postup vložení klíčového úložiště 56
Obrázek 21: Mobility mód 57
Obrázek 22: Permanent mód 58
Obrázek 23: Konfigurace 60

SEZNAM TABULEK

Tabulka 1: Podporované hodnoty TNF 25
Tabulka 2: Položky nešifrovaného obsahu tagu. 33
Tabulka 3: Položky šifrovaného obsahu tagu. 34

SEZNAM PŘÍLOH

P I CD

PŘÍLOHA P I: CD

Obsah přiloženého CD:

- Diplomová práce v elektronické podobě
- Json schéma obsahu tagu
- Zdrojové kódy aplikace pro tvorbu tagů
- Zdrojové kódy klientské aplikace
- APK obou aplikací