Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Mobilní aplikace pro vzdálenou autentizaci uživatele

Mobile Application for Remote User Authentication

2018 **Bc. Vladimíra Černá**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student:	**Bc. Vladimíra Černá**
Studijní program:	B2647 Informační a komunikační technologie
Studijní obor:	2612R025 Informatika a výpočetní technika
Téma:	Mobilní aplikace pro vzdálenou autentizaci uživatele Mobile Application for Remote User Authentication
Jazyk vypracování:	čeština

Zásady pro vypracování:

Aplikace bude sloužit pro autentizaci uživatele vůči počítači na základě rozpoznávání identity pomocí technologie Bluetooth. Aplikace vyhodnotí identitu uživatele a umožní mu autentizaci vůči počítači, a to jak v jednofázovém (detekce přiblížení kompatibilního zařízení), tak dvoufázovém režimu (mobilní zařízení vyzve uživatele k potvrzení akce). Součástí práce bude mobilní aplikace a desktopová komponenta zajišťující autentizace vůči OS. K uzamčení stanice dojde buď oddálením mobilního zařízení mimo dosah Bluetooth technologie nebo na základě vyhodnocení údajů ze senzorů pohybu.

1. Proveďte rešerši metod vzdálené autentizace, použitých technologií, autentizačních komponent operačního systému a zabezpečení.
2. Navrhněte vhodné autentizační schéma s využitím asymetrické kryptografie, analyzujte bezpečnostní rizika.
3. Naimplementujte modul operačního systému (Windows nebo Linux) realizující autentizaci uživatele.
4. Naimplementujte mobilní aplikaci pro operační systém Android, komunikující s vytvořeným autentizačním modulem a vyhodnocujícím údaje ze senzorů (akcelerometr, gyroskop).
5. Otestujte Vámi navržené řešení a porovnejte jej s jinými metodami autentizace.
6. Zhodnoťte dosažené výsledky a možnosti dalšího vývoje.

Seznam doporučené odborné literatury:

[1] John Horton, Android Programming for Beginners, Packt Publishing, 2015, ISBN 1785883267
[2] Ian F. Darwin, Android Cookbook: Problems and Solutions for Android Developers, O'Reilly Media; 2 edition, 2017, ISBN 1449374433
[3] Bruce Schneier, Applied Cryptography: Protocols, Algorithms and Source Code in C , Wiley; 1 edition, 2015, ISBN 1119096723
[4] Pavel Yosifovich et al., Windows Internals, Part 1: System architecture, processes, threads, memory management, and more (7th Edition), Microsoft Press, 2017, ISBN 0735684189

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Ing. Michal Krumnikl, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě dne: *29. dubna 2018*

Vladimíra Tirmá

.....

podpis studenta

Ráda bych poděkovala Mgr. Ing. Michalu Krumniklovi, Ph.D. za odbornou pomoc, konzultaci a velkou vstřícnost při vytváření této bakalářské práce.

Abstrakt

Cílem této práce je implementace aplikace pro mobilní telefon na platformě Android komunikující s aplikaci na osobním počítači a umožňující vzdálenou autentifikaci uživatele. Aplikace pro mobilní telefon bude navržena tak, aby detekovala pohyb uživatele a v tu chvíli umožnila zamčení počítače. V případě vzdálení mobilního telefonu z dosahu Bluetooth signálu počítače dojde k okamžitému uzamčení. Teoretická část je zaměřena na rozbor technologií, ze kterých jsem v práci vybírala nebo jsem je v práci přímo použila. Praktická část se věnuje přímé implementaci řešení.

Klíčová slova: Android, C#, vzdálená autentizace, mobilní aplikace, Windows služba

Abstract

This thesis is focused on implementation of application for Android mobile phone communicating with desktop application and providing user with remote authentication. Mobile application is designed to detect if user is walking and at this moment it offers him the choice to lock the computer. In the case the user is out of range of Bluetooth signal the computer locks itself immediatelly. The theoretical part is focused on research of technologies I would like to choose from or I have used. Practical part is focused on the implementation.

Key words: Android, C#, remote authentization, mobile application, Windows service

Obsah

Seznam použitých zkratek

Zkratka	Význam
API	Application Programming Interface
ASK	Amplitude Shifting Key
BR	Basic Rate
BSIG	Bluetooth Special Interest Group
DES	Data Encryption Standard
EDR	Enhanced Data Rate
FH-SS	Frequency Hopping-Spread Spectrum
FIPS	Federal Information Processing Standards
FSK	Frequency Shift Keying
GFSK	Gaussian Frequency Shift Keying
GSM	Global System for Mobile Communications
IBM	International Business Machine
ISM	Industrial, Scientific, Medical
LAN	Local Area Network
LTE	Long Term Evolution
MAC	Media Access Control
MANET	Mobile Ad Hoc Network
MIDL	Microsoft Interface Definition Language
NFC	Near-Field Communication
NIST	National Institut for Standards and Technology
OS	Operační systém
PAN	Personal Area Network
PIN	Personal Identification Number
RF	Radio Frequency
RFID	Radio Frequency Identification Device
Rijndael	Rijmen Daemen
RPC	Remote Procedure Call
RSA	Rivest, Shamir, Adleman
TCP/IP	Transmission Control Protocol/ Internet Protocol
TDD	Time Division Duplex
UMTS	Universal Mobile Telecommunication Systém

Zkratka	Význam
USA	United States of America
UUID	Universal Unique Identifier
Wifi	Wireless Fidelity
WPF	Windows Presentation Foundation

Seznam obrázků

Seznam tabulek

Úvod

Se vzrůstajícím životním a pracovním tempem roste také potřeba zrychlit či úplně omezit rutinní denní aktivity tak, aby se stávaly co nejméně časově náročnými a my mohli náš cenný čas věnovat činnostem, které jsou hodnoceny jako přínosnější. Bohužel dnes více než kdykoliv předtím pociťujeme na vlastní kůži význam rčení „čas jsou peníze“. V tomto ohledu se může jevit zamykání a odemykání plochy počítače na pracovišti či v pohodlí domova jako rutinní aktivita a přiznejme si, že v případě časté změny hesla až obtěžující. V případě zaměstnání v momentálně již neoblíbených velkých otevřených prostorách také roste riziko zahlédnutí zadávaného hesla a jeho pozdější zneužití. Každý z nás již jistě zažil situaci, kdy s myšlenkou „pouze si rychle odskočím, nemá smysl plochu zamykat“ odběhl k tiskárně umístěné tzv. za rohem a po svém návratu k pracovnímu místu zde našel, a to v tom lepším případě, humorný pozdrav svých kolegů v podobě nové často o dost progresivnější tapety na ploše. Za mou dosavadní praxi byl zatím nejpovedenější kousek vyfocení obsahu plochy, následné skrytí ikon a nastavení onoho obrazu plochy jako pozadí. Tento se stal velmi oblíbeným obzvláště pro přihlížející publikum, avšak nepříliš technicky zdatná oběť v tu chvíli jejich nadšení nesdílela a snažila se zběsilým klikáním a později dokonce servisním zásahem do myši celou věc rychle vyřešit. Myslím, že s podobnými situacemi se setkal již každý z nás a jejich popisování by možná vedlo přinejmenším k pousmání čtenáře, avšak vydalo by zřejmě na samostatnou slohovou práci.

Pokud bych se však měla vrátit k tomu, co pro mne byl onen impuls ke zpracování mého tématu, pak to byl nástup na projekt, kde spolupracuji s kolegy z protějšího konce naší republiky. Když jsem poprvé přišla do jejich velmi přátelského kolektivu, již několik málo dní po příjezdu mi byl na plochu naistalován portrét veselého čuníka v téměř živé velikosti. Byla jsem až šokovaná, s jakou obratností a rychlostí dokázali obraz na ploše modifikovat, neboť se jednalo snad o minutu, kdy jsem jen udělala pár kroků ke kolegovi vedle a zpět. Tento zážitek ve mně zanechal jisté ponaučení a touhu nepříjemné zamykání a odemykání počítače omezit či úplně odstranit.

Žijeme v době, kdy se mobilní telefon stal naší nedílnou součástí. Nosíme jej při sobě na ranních procházkách se psem, je tichým přítelem u oběda, odpoledne nám pomáhá měřit, jakou vzdálenost a za jaký čas jsme uběhli, fotí naše každodenní zážitky, které pak ihned sdílí s našimi přáteli, čteme smutné i veselé zprávy ze všech koutů světa. Často se také stává naším společníkem na místech, kam s námi jiní nechodí. Zkrátka je stále všude s námi. Této pro mnohé již zcela přirozené věci jsem se rozhodla využít a navrhnout aplikaci pro počítač, která by detekovala pohyb uživatele prostřednictvím mobilního telefonu. Mobilní telefony mimo jiné v dnešní době již nabízejí nepřeberné množství funkcí. Mezi nimi také pestrou nabídku senzorů, a to jak hardwarových, tak softwarových. Tyto jsou schopné vyhodnocovat rozmanité fyzikální veličiny od pohybu zařízení až po srdeční tep. A právě senzory pohybu, konkrétně tedy krokoměr, jsem zvolila pro určování okamžiku, kdy se uživatel začne vzdalovat od svého počítače. Aplikace nabízí prostřednictvím jednoduchého menu volbu zamčení, případně jeho odložení a v krajním okamžiku, kdy spolu obě zařízení ztratí kontakt, také okamžité zamčení bez dotazování.

1 Teoretický rozbor funkce

V následujících kapitolách bych se ráda věnovala technologiím, ze kterých bylo v řešení vybíráno, nebo byly nakonec použity. Také bych ráda nastínila, jakým způsobem lze zabezpečit komunikaci mezi mobilním telefonem a osobním počítačem a v neposlední řadě také rozeberu, jaké asynchronní vstupy nabízí platforma Android a jak je s nimi možné pracovat.

1.1 Technologie RFID

Jelikož je potřeba, aby mobilní telefon komunikoval s počítačem a navzájem si zasílali data, musela být vybrána technologie, která toto umožní. RFID (Radio Frequency Identification Device) technologie je dnes aktivně využívána pro vzdálenou autentizaci, a proto bych ráda popsala principy jejího fungování a využití v následujících odstavcích.

Technologie RFID, jak už její název napovídá, využívá ke své funkci radiofrekvenční vlny. Celý systém tvoří transpondéry, čtečky a podpůrné systémy. Transpondér je umístěn na objektu, který má být identifikován a jsou v něm uložena identifikační data. Čtečky mohou mít dva režimy zápis nebo čtení a zápis. Podpůrné systémy jsou například řídící počítače, databáze nebo telekomunikační sítě. Transpondéry se dále dělí na aktivní a pasivní. Pasivní využívají ke svému fungování energii vyslanou ze čtečky [4][5].

Princip fungování je stručně takový, jak jej v následujících řádcích přiblížím. V prvním kroku čtečka vyšle na svém nosném kmitočtu signál ve formě elektromagnetické vlny, ten je následně zachycen anténou transpondéru a indukované napětí vyvolá elektrický proud. V dalším kroku je elektrický proud usměrněn a pomocí něj dojde k nabití kondenzátoru v transpondéru. Výsledná energie je využívána pro napájení logických a radiových obvodů. Transpondér obsahuje také mikroprocesor nebo logický automat. Ten je spuštěn v okamžiku, kdy napětí na kondenzátoru dosáhne minimální hodnoty nutné pro jeho uvedení do provozu. V této chvíli vyšle transpondér odpověď čtečce. Při této komunikaci je zpravidla využíváno dvoustavové ASK modulace (Amplitude Shifting Key). Tato vzniká změnou zakončovací impedance antény transpondéru. Čtečka pak tyto odrazy změny impedance zachycuje a dekóduje na logickou 1 nebo 0. Čtečka také zpravidla obsahuje další interface, aby byla schopná přeposílat informace dalším systémům, jako jsou např. počítače. RFID systémy s frekvencemi od 100 kHz do 30 MHz využívají indukčního párování. Oproti tomu mikrovlnné systémy s frekvenčním pásmem 2,45-5,8 GHz využívají elektromagnetické pole. Nízkofrekvenční RF systémy jsou primárné využívány pro jejich lepší prostupnost objekty. Tohoto využívá například bolus – transpondér umístěný do bachoru či knihy přežvýkavců, který je detekovatelný zvenčí při frekvencích nižších než 135 kHz. Mikrovlnné systémy mají vyšší dosah, typicky 2-15 m, avšak na rozdíl od indukčních systémů vyžadují ke své funkci dodatečnou záložní baterii, neboť energie přenesená čtečkou je zpravidla nedostatečná. Druhy transpondérů a jejich použití můžeme vidět v tabulce 1.1[4][5].

Tabulka 1.1: *Druhy transpondéru a jejich použití*

Druh	Provedení	Použití
Skleněný transpondér	skleněná trubička	implantáty, označování zvířat
Klíče a klíčenky	zabudován do klíčů	zámky, imobilizéry
Bezkontaktní kartičky	karta	čipové karty
Inteligentní štítky	papírový štítek	označování zavazadel, zboží

Pro systém Android je technologie RFID využitelná v omezené míře, a to především z toho důvodu, že systém sám o sobě transceiver na 125 kHz neobsahuje. Nicméně kompatibilitu s RFID technologií lze zajistit připojením externí RFID čtečky, např. od společnosti Zebra Technologies [25].

1.2 Technologie NFC

Další technologií, která se aktivně využívá pro vzdálenou autentizaci či výměnu dat, je technologie NFC (Near-Field Communication), proto bych jí také ráda věnovala pár následujících odstavců a představila v krátkosti její princip fungování a použití.

Technologie NFC neboli komunikace na krátkou vzdálenost není RFID systém, nýbrž bezdrátové datové rozhraní mezí zařízeními podobné např. velmi známému Bluetooth. Nicméně NFC má několik vlastností, které jsou ve vztahu s RFID zajímavé. Přenos dat mezi dvěma NFC zařízeními využívá vysokofrekvenční magnetické střídavé pole o frekvenčním rozsahu 13,56 MHz. Maximální komunikační dosah typický pro NFC je 20 cm, neboť příslušný komunikační protějšek musí být umístěn v těsné blízkosti vysílací antény. To je také důvod, proč se tento druh komunikace nazývá komunikace na krátkou vzdálenost [5] [21] [22].

NFC rozhraní má 13,56 MHz vysílač a 13,56 MHz přijímač, alternativně připojené k anténě. Anténa bývá velkoplošná cívka či vodičová smyčka. Zařízení komunikující prostřednictvím technologie NFC mohou mít více rolí – NFC iniciátor (master), který komunikaci vždy začíná, nebo NFC cíl (slave). Mimo jiné rozeznáváme také dva módy komunikace, a to aktivní a pasivní. Aby dvě zařízení komunikovala v aktivním módu, musí jedno z nich aktivovat svůj vysílač, čímž se automaticky stává iniciátorem neboli přebírá roli master. Vysokofrekvenční proud, který teče anténou, indukuje střídavé magnetické pole, které se šíří okolím antény. Část indukovaného magnetického pole se přesune na anténu druhého zařízení, které je v těsné blízkosti. Poté je na této anténě indukováno napětí, které je zachyceno přijímačem. Pokud jedno zařízení zachytí signál a příslušné příkazy jiného zařízení neboli iniciátora, pak se z tohoto zařízení automaticky stává NFC přijímač neboli slave. Pro přenos dat mezi dvěma NFC zařízeními je využita amplitudová modulace generovaného střídavého magnetického pole. Podobně tomu je také u RFID čtečky a transpondéru. Nicméně na rozdíl od NFC u RFID technologie musí být transpondér z tohoto magnetického pole napájen. U NFC dochází k napájení přímo ze zařízení. Podobně jako u aktivního módu NFC Iniciátor indukuje střídavé magnetické pole pro přenos dat směrem k přijímacímu zařízení. Opět je využívána amplitudová modulace, avšak jakmile je přenesen datový blok, pole není přerušeno, nýbrž je dále vyzařováno, avšak v nemodulované formě. V této chvíli je přijímací zařízení taktéž schopno přenášet data prostřednictvím magnetického pole od inicializačního zařízení. Tento princip nám je již znám z technologie RFID. Využití pasivního módu v NFC komunikaci nabízí spoustu výhod. Také role jednotlivých zařízení komunikujících prostřednictvím pasivní NFC komunikace se mohou navzájem měnit. Tudíž zařízení, jež má například nízkokapacitní baterii, může v rámci úspory energie přejít do role přijímače. NFC zařízení, které figuruje jako podřízené zařízení, může také komunikaci iniciovat, což vede k tomu, že je kompatibilní s RFID transpondéry, kterým dodá energii a ty jsou poté schopny přenášet data k zařízení s NFC rozhraním. To umožňuje elektronickým zařízením vybaveným NFC technologií komunikovat například s inteligentními štítky. Pokud je NFC zařízení v blízkosti kompatibilní RFID čtečky, pak je s ní také schopné komunikovat. Zde přebírá roli přijímače a je schopno přenášet data směrem ke čtečce prostřednictvím energie od ní přijaté. Tato vlastnost umožňuje RFID čtečkám komunikovat s NFC zařízeními jako jsou například mobilní telefony. Z pohledu RFID čtečky se NFC zařízení chová stejně jako bezkontaktní čipová karta. Tento mód se nazývá kartový emulační mód [5] [21] [22].

Technologie RFID rozebraná v předchozí kapitole je technologie využívaná každým z nás denně, to samé by se dalo tvrdit o technologii NFC, která v této době opět doznává velkého rozmachu hlavně tedy v případě mobilních telefonů. Nicméně z důvodu jejich krátkého dosahu pro účely mého řešení nebyly zcela vhodné.

1.3 Technologie Bluetooth

Technologie Bluetooth je velmi využívaným způsobem bezdrátové komunikace a jelikož její dosah je v porovnání s předchozími dvěma technologiemi podstatně větší, byla vybrána jako technologie řešící tu část komunikace mezi mobilním telefonem a osobním počítačem. Z tohoto důvodu jsem se rozhodla v následujících řádcích představit její stručný vývoj, principy fungování a použití.

1.3.1 Historie Bluetooth

Vznik radiově založené komunikace s názvem Bluetooth se datuje zpět v čase již do roku 1994, kdy firma Ericsson hledala způsob, jak nahradit sériové kabelové spojení, bezdrátovým přenosem. K tomuto účelu využila radiové vlny krátkého dosahu. Postupně se připojovaly další společnosti, které se o rozvoj této technologie zasadily, což nakonec vyústilo v to, že v roce 1998 vzniklo sdružení 5 firem (IBM, Toshiba, Intel, Ericsson a NokiaBluetooth) nazvané Bluetooth Special Interest Group, zkráceně BSIG. Název tato technologie získala podle anglického překladu jména skandinávského krále Haralda Blaatanda (měl modře zbarvené zuby od borůvek a odtud tedy jeho přízvisko „modrý zub“). Jakožto velmi dobrý diplomat se zasloužil o sjednocení vikingských kmenů, a právě tato analogie byla využita pro pojmenování technologii, jejíž cílem je také spojovat [1][2].

1.3.2 Topologie a parametry

Technologie Bluetooth se řídí standardem IEEE802.15.1 a řadí se do kategorie osobních počítačových sítí (PAN – Personal Area Network). Systém Bluetooth tvoří malé síťové struktury, které se nazývají pikonety. Každé elektronické zařízení v této síti má v sobě obsažen přijímač a vysílač tzv. transceiver a procesor základního pásma. Procesor základního pásma řídí činnost rádiové části, komunikaci v síti i komunikaci s hostitelským zařízením. V jednom pikonetu může být až 8 navzájem komunikujících terminálů, které jsou obsaženy v samotných elektronických zařízeních. Pokud bychom uvažovali nominální výkon vysílačů zařízení 0 dBm, pak by jejich dosah byl až 10 m. Kdyby však bylo užito pomocných zesilovačů, tato vzdálenost by se prodloužila až na 100 m. Pokud je součástí také přístupový bod obsahující terminál, je možné navázat spojení se sítí LAN. Přenosové rychlosti můžeme vidět v tabulce 1.2 [2][3].

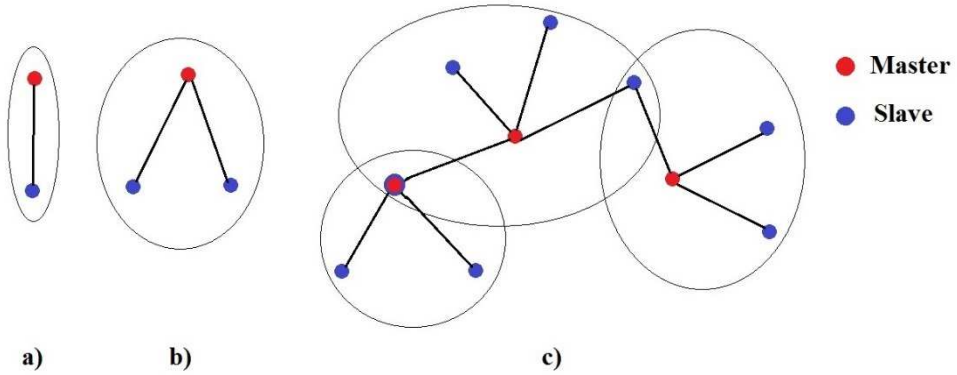
Tabulka 1.2: *Maximální přenosové rychlosti a propustnosti*

Verze	Přenosová rychlost [Mbit/s]	Maximální propustnost [Mbit/s]
Verze 1.2	1	0,7
Verze 2.0 + EDR	3	1,4
Verze 3.0 + HS	24	
Verze 4.0	24	

Systém Bluetooth se řadí mezí sítě s tzv. ad hoc topologií, konkrétněji tedy se jedná o mobilní sítě ad hoc neboli MANET (Mobile Ad Hoc Network), čímž se liší např. od sítí GSM majících topologii

celulární neboli buňkovou. Ad hoc topologie se vyznačují tím, že v nich neexistuje žádná centrální řídicí jednotka neboli základnová stanice, jak je tomu u buňkových topologií (GSM, UMTS, LTE) a ani žádný přístupový bod, jak je tomu u Wifi. Původně byla tato topologie zamýšlena pro zařízení, která byla ve vzájemném komunikačním dosahu a jejich poloha se nijak neměnila. Nemohlo tedy dojít ke ztrátě a následnému opětovnému navázání spojení. Později však rostly požadavky na mobilitu jednotlivých zařízení. Jednotlivé pikonet sítě mohou komunikovat stylem bod-bod nebo bod-více bodů, přičemž účastníci komunikace mezi sebou nemají žádnou hierarchii a všichni komunikují na stejné úrovni [2][3].

Terminály mohou zastávat dvě role, a to master, což je řídící jednotka, nebo slave, tedy podřízená jednotka. Master je terminál inicializující komunikaci, identifikuje účastníky a řídí jejich synchronizaci. Slave jsou ostatní účastníci komunikace. Jeden účastník může figurovat ve více sítích a může v nich zastávat rozdílné role. Tyto role jsou však dočasné a zanikají se zánikem spojení. Komunikace je možná pouze mezi zařízeními master a slave. Spojení slave-slave či master-master není podporované. Může se také stát, že v jedné síti pikonet se nachází další sítě pikonet, takovéto topologii potom říkáme rozptýlená ad hoc. Příklady topologií jsou znázorněny na obrázku 1.1 [2].



Obrázek 1.1: *Sítě pikonet: a), b) ad hoc topologie c) rozptýlená ad hoc topologie*

Technologie Bluetooth využívá kmitočtové pásmo ISM (Industrial, Scientific, Medical), které je vyhrazeno pro průmyslové, vědecké a zdravotnické aplikace a nevyžaduje žádné povolení. Rozsahy pro jednotlivé země jsou uvedeny v tabulce 1.3. Toto pásmo využívají rovněž radiové systémy, proto, aby nedocházelo k vzájemnému rušení, využívá technologie Bluetooth přenos s rozprostřeným spektrem, konkrétně pak tedy variantu s kmitočtovým skákáním nosné vlny FH – SS (Frequency Hopping-Spread Spectrum) [2].

Tabulka 1.3: *Kmitočtové rozsahy pásma ISM*

Stát	Kmitočtový rozsah [MHz]
Evropa a USA	2400 - 2483,5
Francie	2446,5 - 2483,5
Španělsko	2445 - 2475
Japonsko	2471 - 2497

Řídicí jednotka udává pseudonáhodnou sekvenci, kterou má každá síť unikátní a její fáze je dána hodinovým signálem řídicí jednotky. Radiový signál je pak tedy dělen na jednotlivé časové úseky tzv. time sloty a ty jsou poté dále číslovány dle hodinového signálu řídicí jednotky. Terminály spolu

komunikují prostřednictvím časového duplexu TDD. Každý sudý časový úsek je určen pro jednotku master a liché úseky pro jednotku slave. Data se přenášejí prostřednictvím paketů a většinou bývá obsažen jeden paket v jednom časovém úseku. Nemusí to však být pravidlem, v určitých případech může být jeden paket přenášen také v 3 nebo 5 časových úsecích. Formát paketu je graficky znázorněn v tabulce 1.4 [2] [3].

Tabulka 1.4: *Formát paketu*

72 bitů	54 bitů	0-2745 bitů
Přístupový kód	Záhlaví	Data

Bluetooth technologie využívá dvoustavové kmitočtové klíčování FSK s Gaussovskou předmodulační filtrací neboli GFSK se šířkou pásma $BT = 0{,}5$. Kladná odchylka kmitočtu od nosné je vyhodnocována jako logická jedna, a naopak záporná odchylka signalizuje logickou nulu. Komunikace může probíhat synchronně a taktéž asynchronně. Synchronní přenos je využíván zejména při telefonních hovorech a taková komunikace pak nabývá přenosové rychlosti 64 kbit/s v obou směrech, zatímco asynchronní přenos dat může mít přenosové rychlosti pro každý směr jiné. Maximální přenosová rychlost je v takovém případě v jednom směru 723,2 kbit/s a 57,6 kbit/s ve směru opačném. Pokud datový přenos probíhá symetricky, pak může v obou směrech přenosový rychlost dosahovat až 433,9 kbit/s [2].

1.3.3 **Profily Bluetooth**

Profily někdy též nazývané jako služby definují obecné chování a pravidla, která Bluetooth zařízení využívají pro komunikaci mezi sebou. Profily se zakládají na Bluetooth standardech a definují, jaké druhy dat mohou být Bluetooth modulem přenášeny. Oproti tomu pak aplikace definují, jaké profily musí dané zařízení podporovat. Aby byla dvě Bluetooth zařízená kompatibilní, musí podporovat stejné profily. Profily obsahují informace jako jsou závislosti na dalších profilech a doporučený formát uživatelského rozhraní. U Bluetooth BR/EDR neboli Basic Rate / Enhanced Data Rate je v profilech také specifikováno, jaké jsou parametry na každé vrstvě Bluetooth protokolu [6].

1.4 **Zabezpečení komunikace**

Protože v aplikaci bude manipulováno s citlivými daty jako jsou přihlašovací údaje uživatele, je nutné komunikaci mezi mobilním telefon a osobním počítačem patřičným způsobem zabezpečit. Pro tyto účely se nabízí využití kryptografie, která je pro zabezpečení zpráv nejrůznějšího druhu využívána již od dávných dob. V následujícím textu bych se kryptografii ráda věnovala více a nastínila, jaká je její stručná historie, jaké si klade základní cíle a jaké šifrovací algoritmy jsou v dnešní době stále používané a považované za bezpečné.

1.4.1 **Historie kryptografie**

Historie šifrování je dlouhá a zajímavá. Velký význam má ne zcela technická kniha The Codebreakers od Davida Khana vydaná v roce 1967. V ní nás seznamuje se šifrováním od jeho vzniku a prvního užití Egypťany již před více než 4000 lety až po 20. století, kde sehrálo šifrování významnou roli především během dvou světových válek. Khanova kniha zachycuje tato historická období, která byla pro kryptografii z hlediska jejího vývoje nejvýznamnější. Mezi odborníky převládali lidé, jejichž profese byla spjata s armádou, diplomacii či vládou obecně. Šifrování bylo užíváno jako nástroj k ochraně

státního tajemství a strategií. S velkým rozmachem počítačových věd v 60. letech 20. století se tato disciplína přemístila také do soukromé sféry, kde se zvýšila poptávka o ochranu dat v digitální podobě a nabídku zabezpečených služeb. Tato éra počala činností Horsta Feistela z IBM v ranných 70. letech minulého století a vyvrcholila v roce 1977, když byl standard DES (Data Encryption Standard) schválen jako federální standard pro zpracování civilních informací v USA. Jedná se o nejznámější šifrovací mechanismus v historii, který se stále standardně používá pro zabezpečování elektronického obchodu ve finančních institucích po celém světě. Nejstrmějšího vývoje však kryptografie doznala v roce 1976, kdy Bailey Whitfield Diffie a Martin Hellman vydali článek New Directions in Cryptography. Toto vydání představilo novou revoluční metodu výměny klíčů dnes známou jako Diffieho-Hellmanova výměna klíčů. Tato metoda vycházela z problému diskrétního logaritmu. Ačkoliv autoři nepředstavili žádnou praktickou realizaci tohoto šifrování, myšlenka byla zcela jasná a vedla k velmi aktivnímu zájmu kryptografické komunity. V roce 1978 Rivest, Shamir a Adleman jako první představili první praktický příklad šifrování veřejným klíčem a koncepci podepisování nyní známou jako RSA (pojmenování v sobě nese první písmena jmen autorů). Tato šifrovací metoda je založena zase na jiném matematickém problému, a to na obtížné faktorizaci velkého celého čísla. Další metoda byla představena Taher ElGamalem v roce 1985 a taktéž se zakládá na problému diskrétního algoritmu. Další vývoj a vylepšování již existujících šifrovacích technik stále pokračuje až do dnešních dní [7] [23].

1.4.2 Cíle kryptografie

Tak, jako každá nauka, i kryptografie si klade své cíle. Ty hlavní bych ráda přiblížila v následujícím textu. Jsou jimi [7] [23]:

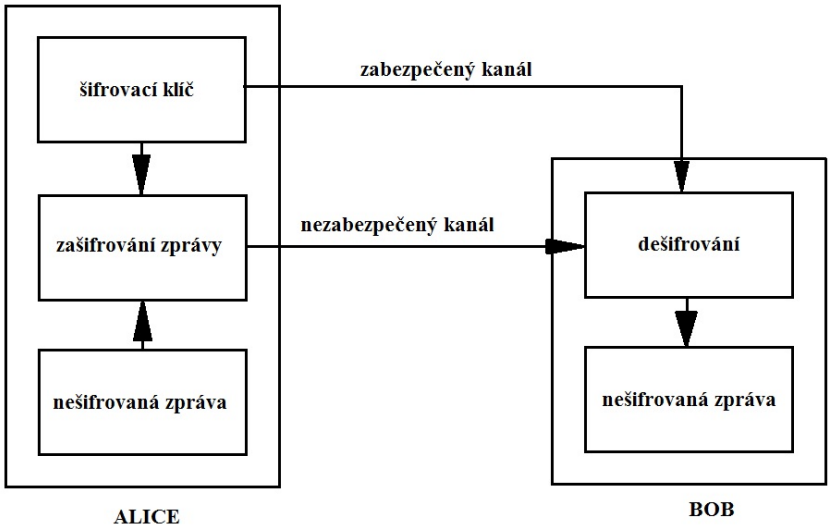
- ***Důvěrnost*** *(Confidentiality)* – Informace se musí dostat jen a pouze k těm, komu je určena a před zbytkem musí zůstat utajena.
- ***Integrita dat*** *(Data integrity)* – Odeslaná data musí být totožná s daty přijatými. Jejich obsah se nesmí změnit.
- ***Autentizace*** *(Authentization)* – Dvě strany, které vstupují do komunikace se musí navzájem identifikovat. Informace, která je skrz kanál doručena, musí být identifikována prostřednictvím svého původu, času, kdy vznikla, datového obsahu, času odeslání atd. Pro tyto účely rozlišujeme ověření totožnosti subjektu a ověření totožnosti dat.
- ***Nepopiratelnost*** *(Non-repudiation)* – Subjekt nemůže odmítnout provedení předchozích akcí či závazků. Konkrétněji, pokud jsou data podepsána určitým privátním klíčem, původce nemůže odmítnout, že data pochází od něj, neboť podepsání lze učinit pouze pomocí privátního klíče, který je pevně vázán na klíč veřejný a bez něj by nebylo možné zprávu dešifrovat.

1.4.3 Symetrická kryptografie

Způsob zabezpečení komunikace, který se v implementované aplikaci pro vzdálenou autentizaci nabízí, je symetrická kryptografie. Pro svůj výpočetní výkon a bezpečnost je stále velmi využívaná, proto v následujících řádcích rozvedu, na jakých principech funguje. V následující kapitole také představím velmi známý algoritmus, který je využíván dodnes, a to AES.

Symetrická kryptografie je taková kryptografie, kde je jak pro šifrování, tak dešifrování použit jeden klíč. Tento klíč mají obě strany. Komunikaci lze popsat způsobem, jak je schematicky znázorněno na obrázku 1.2. Jeden z největších problémů, který tato komunikace skýtá je nutnost využití efektivní

metody, jak přenést symetrický klíč, aniž by mohlo dojít k jeho získání třetí stranou. Tento problém je znám jako problém s výměnou klíčů [7] [24].



Obrázek 1.2: *Schématické znázornění komunikace při AES šifrování*

Blokové šifry

Blokové šifry bývají v kryptografických systémech ty nejdůležitější a nejvýznamnější. Jejich univerzálnost umožňuje sestavení generátorů pseudonáhodných čísel, proudových šifer a hašovacích funkcí. Dále mohou být využity jako hlavní komponenta v autentizačních technikách zpráv, mechanismech pro kontrolu integrity dat, autentizačních protokolech subjektů a schématech pro podepisování symetrickým klíčem. Princip této šifry je, že zpráva v nezašifrované podobě je rozdělena do řetězců o konstantní délce tzv. bloků a každý blok je poté šifrován najednou klíčem, který je fixní, tedy se nemění. Tuto metodu využívají nejznámější symetrické šifry. Dva nejdůležitější druhy symetrické kryptografie jsou substituční a transpoziční [7].

Proudové šifry

Proudové šifry jsou důležitou součástí šifrovacích algoritmů. Jejich princip je takový, že jsou šifrovány jednotlivé znaky (obvykle binární data) nezašifrované zprávy klíčem, který se v čase mění. Proudové šifry jsou zpravidla rychlejší než blokové. Bývají také často vhodnější a někdy zcela závazné (např. v telekomunikacích), zpravidla v případech, kdy je velikost vyrovnávací paměti omezena anebo když jednotlivé znaky musí být šifrovány tak, jak jsou postupně přijímány. Jelikož mají limitovanou nebo žádnou chybovost, mohou mít tyto šifry výhodu v případech, kdy je vysoce pravděpodobný přenos chyb [7].

Kryptografie DES

Šifra DES neboli Data Encryption Standard patří k nejznámějším symetrickým kryptografiím a jedná se o předchůdce pozdější šifry AES. Světově známá byla DES šifra v polovině 70. let minulého století jako první zveřejněná včetně plně specifikovaných implementačních detailů. Byla přijata federálními standardy pro zpracování informací USA neboli NIST. Základem pro DES kryptografii je Feistelova šifra, která zahrnuje opakování běžné sekvence operací. Základní myšlenkou je sestavení komplexní

šifrovací funkce složením několika jednoduchých operací jako jsou výměna, posunutí, lineární zobrazení, aritmetické operace a jednoduchá nahrazení. DES šifruje text po blocích o velikosti 64 bit, poslední blok je třeba na tuto velikost doplnit. Velikost klíče je 56 bitů, existuje tedy 256 klíčů. Přesněji tedy velikost klíče je specifikována jako 64 bitů, přičemž nejméně signifikantní bit z každého bajtu lze využít jako paritní bit tak, aby v každém bajtu byl lichý počet logických jedniček. Hlasitě se spekuluje, že paritní bity byly představeny právě proto, aby efektivní délka klíče byla zkrácena z 64 bitů na 56 bitů, aby se zjednodušila operace hledání klíče [7].

Nicméně tato šifra již není považována za bezpečnou, a to z důvodu nízké délky klíče. Z tohoto důvodu se dnes již nevyužívá a byla nahrazena šifrou AES, kterou přiblížím v následujících řádcích.

Kryptografie AES

Jak se množily snahy o překonání kryptografie DES, rostla v USA potřeba ji nahradit bezpečnější variantou. Vyústění na sebe nenechalo dlouho čekat a NIST (National Institut for Standards and Technology) vyhlásil v lednu roce 1997 soutěž o vyvinutí nové šifrovací techniky s názvem AES (Advanced Encryption Standard). Původně měla soutěž 15 účastníků, kteří byli postupně zredukováni na 5 nejúspěšnějších a z nich poté vybrána v říjnu roku 2000 RijnDael – šifra dvou belgických matematiků Vincenta Rijmena a Joana Daemena. To, že byl Rijndael vybrát bylo překvapivé, neboť se údajně věřilo, že NIST nepřijme šifrovací algoritmus někoho, kdo není občan USA [9] [10].

Rijndael a AES nejsou zcela totožné, liší se však pouze v rozmezí podporované délky bloku a délky šifrovacího klíče. Rijndael je bloková šifra s proměnlivou délkou klíče i bloku. Délka bloku může být jakýkoliv násobek čísla 32 větší nebo rovný číslu 128 a zároveň menší nebo roven číslu 256. Pro AES jsou povoleny tři délky klíče - 128 bitů, 192 bitů a 256 bitů a délka jednoho šifrovaného bloku je stanovena na 128 bitů. S délkou klíče se také mění částečně šifrovací algoritmus, který má 10, 12 nebo 14 rund. Konkrétněji jsou počty rund vypočteny jako počet 32 bitových slov v klíči zvětšen o 6 [8] [9] [10].

Základní jednotkou šifrování je bajt tedy posloupnost 8 bitů, ke které se přistupuje jako k jednomu objektu. Základním vstupem a výstupem jsou pak pole bajtů. Vstupní údaje pro potřeby šifrování jsou šifrovací klíč a blok textu, který má být šifrován, výstupem je pak tedy zašifrovaný blok. Jedna runda algoritmu Rijndael a jeho jednotlivé kroky pracují s maticí bajtů, které se říká stav. Stav si můžeme představit jako dvourozměrné pole, které má v případě AES 4 řádky a 4 sloupce. Na konci šifrování je text sestaven zpět z bajtů stavu. Šifrovací klíč je taktéž rozložen do matice, konkrétnější tedy matice o 4 řádcích, jak tomu bylo u stavu, a počet sloupců se odvíjí od délky klíče neboli je roven délce klíče vydělené 32 bity. Pokud bychom počet rund označili jako Nr, pak před začátkem musí být vytvořeno 4 + Nr*4 tzv. rundovních klíčů. Na úplném začátku před provedením první rundy je provedeno zamíchání, což znamená, že nešifrovaný text je pomocí operace XOR změněn prvními čtyřmi klíči. Poté je provedeno Nr rund. Každá runda je sekvence čtyř transformací. Poslední runda je však odlišná – nedochází v ní k míchání sloupců. Základní kritéria, která si autoři stanovili jsou taková, že všechny kroky jedné rundy nesmí být proveditelné v obráceném pořadí a jsou preferovány jednoduché komponenty před složitými [9] [10] [24].

Krok č. 1: záměna bajtů

Jedná se o jedinou nelineární transformaci v šifře. Na bajt stavu je aplikována substituce prostřednictvím 8bitového substitučního boxu, tzv. Rijndael S-boxu. Celou operaci si můžeme představit jako nelineární substituční funkci, to zajišťuje, že šifra není lineální [8] [9] [10].

Při návrhu S-Boxu si autoři zvolili kritéria, že koeficient korelace vstupu a výstupu musí být nejmenší možný, pravděpodobnost rozšíření rozdílu musí být co nejmenší možná a algebraické vyjádření substituční matice musí být složité.

Krok č. 2: posunutí řádků
Výměna řádků je transpozice bajtů, která cyklicky přesune řádky stavu o rozdílné posunutí (0-3). První řádek je posunut doleva o 0, druhý o 1, třetí o 2 a čtvrtý o 3. V tomto kroku autoři usilovali o optimální rozptyl, tzn. čtyři posunutí musí být rozdílná, odolnost vůči útoku prostřednictvím zkrácené diferenciální kryptoanalýzy musí být co nejvyšší.

Diferenciální kryptoanalýza – Jedná se o metodu, která vychází ze zkoumání, jak změny vstupních dat ovlivňují změny výstupních dat [11].
Zkrácená diferenciální kryptoanalýza – jedná se o metodu, která zobecňuje diferenciální kryptoanalýzu pro blokové šifry, uvažuje pouze částečné diference, konkrétněji nezaměřuje se na celý blok, avšak soustředí se pouze na některé jeho bity [11].

Krok č. 3: zamíchání sloupců
Tato transformace je aplikována na všechny sloupce stavu, ke kterým přistupuje jako k čtyřčlenným polynomům. Všechny výstupní bity jsou ovlivněny všemi vstupními bity – jsou jejich lineární kombinací. Tímto krokem dojde k dostatečnému rozptylu v šifře [8] [10].

Autoři usilovali, aby transformace byly kostkové transformace na 4bajtových sloupcích, byly nejlépe lineární, měly odpovídající účinnost rozptýlení informace a také aby výkonnost na 8bitových procesorech byla co nejvyšší. Tento krok je jediný, který nebylo jednoduché optimalizovat pro 8bitové procesory [10].

Krok č. 4: Přidání klíče
Při této transformaci dochází k úpravě stavu tak, že je kombinován s rundovním klíčem operací XOR. Rundovní klíč má stejnou délku jako jeden šifrovaný blok a je odvozen z šifrovacího klíče. První přidání klíče proběhne ještě před první rundou

1.4.4 **Asymetrická kryptografie**

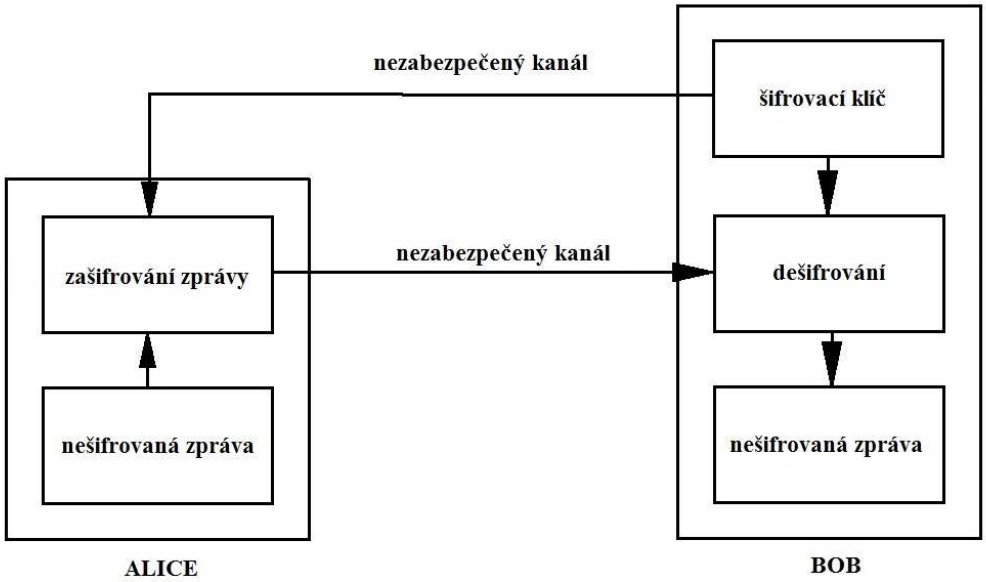
Přesto, že je symetrická kryptografie AES považována v dnešní době za stále bezpečnou a je stále vedena jako šifrovací standard v USA, pořád přetrvává problém bezpečné prvotní výměny šifrovacích klíčů. Aby mohlo být toto provedeno bezpečně, musí být komunikace vedena zabezpečeným kanálem. Tato možnost však v implementovaném řešení nebyla, a proto bylo přistoupeno ke zvolení jiné metody, a to využití asymetrické kryptografie. V této kapitole se tedy budu věnovat asymetrické kryptografii neboli šifrování veřejným klíčem, a to konkrétně metodě RSA.

Kryptografie RSA

Algoritmus pro asymetrickou kryptografii s názvem RSA vznikl v roce 1977 a byl pojmenován je dle prvních písmen příjmení svých autorů – Rivest Shamir a Adelman. V algoritmu RSA se využívá dvojice klíčů tedy veřejný a k němu příslušející soukromý klíč. Jejich generování probíhá následujícím způsobem [7]:

1) Vybereme dvě náhodná navzájem rozdílná prvočísla p, q.
2) Vypočítáme jejich součin $n = pq$ a hodnotu Eulerovy funkce $\phi = (p-1)(q-1)$.
3) Vybereme náhodné číslo e tak, aby platilo $1 < e < \phi$ a $\gcd(e, \phi) = 1$.
4) Použijeme výpočet rozšířeného Euklidova algoritmu pro získání jedinečného přirozeného čísla d, takového, že $1 < d < \phi$ a zároveň platí, že $ed \equiv 1 \pmod{\phi}$.
5) Poté veřejným klíčem nazýváme dvojici (n, e) a soukromým klíčem d.

Komunikace probíhá dle schématu na obrátku 1.3. Pro ilustraci použiji imaginární korespondenty Alici a Boba. Bob zvolí šifrovací pár veřejného a k němu příslušného soukromého klíče. Veřejný klíč pošle prostřednictvím nezabezpečeného kanálu Alici. Svůj dešifrovací neboli soukromý klíč si však ponechá v tajnosti. Nyní může Alice zaslat Bobovi zprávu zašifrovanou prostřednictvím veřejného klíče přijatého od Boba. Bob přijatou zprávu dešifruje inverzní transformací danou jeho soukromým klíčem [7].



Obrázek 1.3: *Schématické zobrazení komunikace pomocí šifrování RSA*

Zjednodušeně bychom šifrování a dešifrování zprávy mohli shrnout do následujících kroků [7]:
1) Obdržíme veřejný klíč jako dvojici (n, e).
2) Zvolme zprávu m jako přirozené číslo z intervalu $(0, n-1)$.
3) Vypočítáme $c = m^e \bmod n$.
4) Šifrovanou zprávu odešleme adresátovi.
5) Adresát získá původní zprávu m z c pomocí soukromého klíče $m = c^d \bmod n$.

Přirozené číslo e v kryptografii nazýváme šifrovací exponent a d pak dešifrovací exponent. Jejich součin n pak nazýváme modulem. Oproti symetrické kryptografii zde najdeme v komunikaci rozdíl v prvotní výměně klíče. Zde probíhá prostřednictvím nezabezpečeného kanálu, avšak u asymetrické kryptografie je třeba prvotní výměnu provést přes zabezpečený kanál, neboť by mohlo dojít k zneužití klíče a následné dešifrování komunikace [7].

1.4.5 Analýza bezpečnostních rizik

Tak jako u každé aplikace i u této je třeba se zabývat bezpečnostními riziky a zhodnotit jejich možný dopad. V následujících odstavcích bych ráda představila některá z rizik, která se přímo týkají implementovaného řešení.

Fyzické odcizení mobilního telefonu

Tomuto riziku podléhá téměř každá aplikace, avšak zvláště vysoké riziko s sebou nesou ty aplikace, jež manipulují s citlivými daty uživatele. Pokud ke zcizení dojde, může dojít ke zneužití přihlašovacích údajů uživatele a dat nacházejících se na osobním počítači, která se v tomto okamžiku ocitají zcela nechráněna. Způsob, jak tomu lze zabránit je případná budoucí implementace dvoufázového ověření, např. prostřednictvím otisku prstů pomocí čtečky mobilního telefonu nebo PIN kódu, který zná pouze uživatel aplikace.

Spoofing MAC adresy

Jedná se o situaci, kdy uživatel účelně zamění MAC adresu svého Bluetooth zařízení. Motivy mohou být různé, avšak v tomto případě by mohlo dojít k záměně za adresu uživatele využívajícího aplikaci pro vzdálenou autentizaci s cílem získání jeho přihlašovacích údajů, popřípadě zaslání pokynu k odemknutí počítače jeho adresou. Tyto případy jsou ošetřeny autentizací odesílatele prostřednictvím unikátního identifikátoru každého mobilního zařízení ve formě UUID, jež je uloženo v databázi osobního počítače a je ověřováno s každou přijatou zprávou.

Odposlech sekvence přihlašovacích údajů

Pro správnou funkci aplikace je nutné vybrané mobilní zařízení spolu s přihlašovacími údaji zadat do uživatelského rozhraní v podobě WPF aplikace a tyto jsou uloženy do databáze. Heslo je samozřejmě ukládáno v šifrované podobě. Přesto během ukládací sekvence může dojít ke zcizení přihlašovacích údajů a jejich zneužití.

Replay útok

Replay útok je praktika, kdy jsou data zachycena třetí stranou a následně pozdržena nebo zopakována později. Komunikace mobilního telefonu s osobním počítačem je koncipována na základě zasílání zpráv. Tyto zprávy, ačkoliv jsou šifrované, mohou být zachyceny a následně později přeposlány. Pokud by došlo k zachycení zprávy, která obsahuje požadavek na odemknutí počítače, mohlo by dojít k úniku dat. Tomuto útoku se v budoucnu dá zabránit například umístěním časových razítek nebo číselné posloupnosti do zpráv.

Možnost odcizení přihlašovacích údajů

K účelu vzdálené autentizaci uživatele je třeba, aby jeho přihlašovací údaje byly uloženy v databázi a serverová část aplikace na platformě OS Windows k nim měla přístup. V tomto okamžiku nastává problém s odcizením přihlašovacích údajů. Tomu je předcházeno tím, že jsou data uložena v šifrované

podobě. Šifrování je provedeno pomocí funkce `CryptProtectData()`, která je součástí Windows API.

1.5 Externí asynchronní vstupy aplikace

Pro zachycení okamžiku, kdy se uživatel vzdaluje s mobilním telefonem od počítače, bylo třeba zhodnotit, jakým způsobem by se dala úloha realizovat. Tato část mohla být implementována jak na straně mobilního telefonu, tak na straně počítače, avšak jelikož mobilní telefony dnes již nabízejí celou řadu senzorů, bylo rozhodnuto je v mobilní aplikaci využít a data z nich následně dále zpracovávat a vyhodnocovat. Senzory obecně lze na platformě Android rozdělit do následujících kategorií [12]:

- Pohybové senzory
- Senzory okolního prostředí
- Senzory polohy

Zpracování dat ze senzorů je na platformě Android umožněno senzorovým frameworkem. Ten poskytuje několik tříd a rozhraní, která umožňují velké množství úkonů ve spojitosti se senzory. Například zjištění, které senzory zařízení poskytuje, určení bližších informací o jednotlivých senzorech (maximální rozsah, výrobce, nároky na energii a rozlišení), získání surových data ze senzorů, registrování a odregistrování posluchače událostí snímače, který reaguje na změny senzorů. Obecně jsou k dispozici dva typy senzorů, a to hardwarové a softwarové. Hardwarové senzory jsou fyzické komponenty vestavěné do zařízení získávající data přímým měřením fyzikálních veličin, jako jsou zrychlení, síla geomagnetického pole a změna úhlu. Softwarové senzory nejsou přímo fyzické komponenty, ačkoliv napodobují hardwarové senzory. Softwarové senzory zpracovávají data z jednoho nebo více hardwarových senzorů, a proto se jim také někdy říká virtuální nebo složené senzory. Jak již bylo řečeno, k senzorům můžeme přistupovat prostřednictvím senzorového frameworku. Ten je součástí balíčku `android.hardware` a obsahuje třídy a rozhraní, kterým se budu věnovat v následujícím textu [12].

SensorManager

Tato třída poskytuje různé metody přistupování k senzorům, pomocí ní je možné registrovat a odregistrovat posluchače událostí senzoru a poskytuje také několik konstant, které lze využít k zjištění přesnosti senzorů, nastavení rychlosti získávání dat a kalibrace senzorů.

Senzor

Prostřednictvím této třídy můžeme vytvořit instanci konkrétního senzoru a nastavit různé schopnosti senzoru.

SensorEvent

Systém tuto třídu využívá k vytvoření objektu události senzoru, který poskytuje informace o události snímače. Objekt událostí senzoru obsahuje data jako jsou surová data ze snímače, typ senzoru, který událost vyvolal, přesnost dat, a časový údaj, kdy událost nastala.

SensorEventListener

Pomocí tohoto rozhraní můžeme získat dvě metody zpětného volání, které jsou informovány, když se změní hodnota senzoru nebo jejich přesnost. Pro monitorování surových dat z událostí vyvolaných změnou přesnosti senzoru nebo změnou dat zachycených senzorem nabízí rozhraní dvě funkce, a to

`onAccuracyChanged()` a `onSensorChanged()`. Obě odkazují na objekt snímače, který událost vyvolal. V případě metody `onAccuracyChanged()` je pak přesnost prezentována čtyřmi stavovými konstantami:
`SENSOR_STATUS_ACCURACY_LOW`
`SENSOR_STATUS_ACCURACY_MEDIUM`
`SENSOR_STATUS_ACCURACY_HIGH`
`SENSOR_STATUS_UNRELIABLE`

Při registrování posluchače událostí senzoru také můžeme zvolit zpoždění, s jakým se mají data poskytovat. Jedná se o tzv. vzorkovací frekvenci. Opět je k výběru několik možností:
`SENSOR_DELAY_NORMAL` – Vzorkovací frekvence vhodná typicky pro monitorování orientace obrazovky a využívá zpoždění 200000 mikrosekund.
`SENSOR_DELAY_GAME` – zpoždění 20000 mikrosekund
`SENSOR_DELAY_UI` – zpoždění 60000 mikrosekund
`SENSOR_DELAY_FASTEST` – zpoždění 0 mikrosekund

Pro určení okamžiku, kdy se uživatel pohybuje, bylo rozhodnuto využití pohybových senzorů mobilního telefonu. V následujících řádcích jsem se rozhodla jejich funkce detailněji popsat u několika vybraných [12].

Senzor lineárního zrychlení

Tento senzor poskytuje trojrozměrný vektor reprezentující zrychlení v každé ose zařízení. Tato data se dají využít pro detekování gest. Tento senzor je vhodný, pokud chceme detekovat zrychlení bez vlivu gravitace. Před použitím je nutné odstranit přednastavenou kompenzaci např. tím, že je zařízení umístěno na stůl, je odečteno aktuální zrychlení ve všech osách, a to je posléze od měřených výsledků odečteno.

Senzor významného pohybu

Tento senzor vyvolá událost pokaždé, když dojde k detekování významného pohybu a poté se odpojí. Významný pohyb říkáme pohybu, který může vést ke změně polohy zařízení. Například chůze, jízda na kole nebo pohybující se automobil.

Senzor počítadla kroků

Tento senzor poskytuje počet kroků od posledního restartování zařízení. Senzor počítadla kroků má vyšší zpoždění ale také přesnost než senzor detekování kroku. Abychom předešli velké spotřebě energie, je dobré data z krokoměru vyčítat pomocí plánované úlohy, nikoliv nepřetržitě. A interval jednotlivých úloh se doporučuje nastavit co nejdelší možný.

Senzor detekce kroků

Tento detektor vyvolá událost vždy, když dojde k detekování kroku uživatele. Zpoždění je nastaveno tak, aby bylo menší než dvě sekundy.

1.6 Srovnání již existujících řešení

Pokud bychom se chvíli věnovali vyhledávání podobných řešení, najdeme jich hned několik. Proto jsem se jim rozhodla věnovat následující řádky svého textu a stručně je charakterizovat.

Windows 10 Dynamic Lock

Jedná se o funkci, která je vestavěnou součástí operačního systému Windows 10, jak již název napovídá. Nabízí funkci uzamčení plochy, jakmile se zařízení vzdálí z dosahu. Abychom mohli tuto funkcionalitu použít, je nutné zařízení spárovat s počítačem prostřednictvím Bluetooth a v nastavení zapnout funkci automatického zamykání plochy v případě, že zařízení odejde z dosahu Bluetooth. Funkcionalita umožňuje pouze zamykání. Odemknutí je nutné provést klasicky prostřednictvím zadání hesla na počítači. Zamknutí se provede až ve chvíli, kdy zařízení není v dosahu, dřívější uzamčení plochy aplikace nenabízí. Aplikace je umístěna pouze na počítači a není nutné ji instalovat také do telefonu, čímž není funkcionalita omezena platformou telefonu. Pokud máme spárovaných více zařízení, pak systém bere v potaz všechna.

Windows Hello

Jedná se o funkcionalitu, která umožňuje uživateli přihlášení bez nutnosti zadání hesla prostřednictvím detekce obličeje nebo otisku prstů. Tato funkce spolupracuje také s některými aplikacemi, kde je pak možné se pomocí Windows Hello autentizovat. Opět se jedná o jednostrannou funkcionalitu, která podporuje pouze přihlášení. Není závislá na signálu Bluetooth ani na jiných komunikačních kanálech a není tedy závislá na zařízení, které nosíme či nenosíme u sebe. Přihlášení je možné provést až v bezprostřední blízkosti počítače.

Funkce Knock

Použití je limitováno kombinací počítače značky Mac a telefonu značky iPhone nebo hodinek stejného výrobce. Tato kombinace totiž podporuje funkci Knock, která funguje na základě Bluetooth Low Energy technologie. Díky tomuto funkcionalita nemá vysoké nároky na energii a šetří tedy baterii. Jedná se opět o jednostrannou funkcionalitu, tedy odemykání. V případě, že je aplikace nainstalovaná v Apple hodinkách, pak je uživatel požádán o otisk prstu a odemykání probíhá na základě tohoto údaje. Jedná se o bezpečnější variantu, neboť ji nemůže k odemčení zneužít jiná osoba v případě krádeže. Pokud je aplikace instalovaná v iPhone telefonu, pak stačí pouze poklepat dvakrát na obrazovku telefonu, a to i v případě, že se zařízení nachází např. v kapse, a opět dojde k odemčení plochy. Kompatibilní jsou kombinace následujících telefonů: iPhone 6, 6 Plus, 5S, 5C a 4S s následujícími Mac počítači: 2011 MacBoon Air a novější, 2012 MacBook Pro nebo novější, 2012 iMac nebo novější, 2011 Macmini nebo novější, 2013 Mac Pro.

Aplikace Tether

Aplikace, která je opět určena pro dvojici Mac počítač a iPhone či Apple hodinky a taktéž využívá pro komunikaci technologii Bluetooth Low Eenergy. Tato aplikace podporuje dvojí funkci, a to jak zamykání, tak odemykání plochy počítače. Odemykání je detekování prostřednictvím Bluetooth, jakmile se zařízení přiblíží a vstoupí do dosahu, pak je plocha automaticky odemknuta. Totéž, avšak naopak se děje při zamčení, jakmile je zařízení z Bluetooth dosahu, je plocha počítače zamčena. Aplikace je volně ke stažení, avšak nabízí příplatkové rozšiřující funkce.

MacId

Opět se jedná o aplikaci pro zařízení z továrny Apple, která má jednostrannou funkci – odemykání plochy počítače. K odemknutí využívá čtečku otisku prstů na telefonu. Jedná se tedy o bezpečnější variantu, která nemůže být zneužita další osobou. Použití čtečky otisku prstů může být v případě zadávání dlouhého hesla pro přihlášení mnohem efektivnější a rychlejší řešení. Aplikace je volně ke stažení a je opět využitelná také pro iPhone hodinky.

Smart Lock

Funkcionalita, která je dostupná pro Chromebook počítač a jakýkoliv telefon na platformě Android. Hodinky s platformou Android však podporovány nejsou. Jedná se opět o jednostrannou funkci – odemykání. Aplikace komunikuje prostřednictvím technologie Bluetooth.

Řešení vzdálené implementace, kterým se zabývá tato bakalářská práce, nabízí výhodu ve spojení obou funkcionalit, a to jak zamykání počítače, tak jeho odemykání. Funkce zamykání a odemykání jsou navíc založeny na více vstupech. Těmi jsou pohybové senzory telefonu a současně vzdálenost mobilního telefonu od počítače. Aplikace je také dostupná pro nejrozšířenější platformu mobilních telefonů, kterou je Android. Srovnání jednotlivých řešení jsem provedla pro přehlednost v následující tabulce 1.5, do které jsem přidala také implementované řešení [13].

Tabulka 1.5: *Srovnání aplikací pro vzdálenou autentizaci.*

název aplikace	zamykání	odemykání	platforma počítače	platforma zařízení	podporuje i jiná zařízení než telefon	cena
Windows 10 Dynamic Lock	ano	ne	Windows	jakákoliv	ne	zdarma
Windows Hello	ne	ano	Windows	jakákoliv	ne	zdarma
Knock	ne	ano	Mac OS	iOS	ano	zdarma
Tether	ano	ano	Mac OS	iOS	ano	zdarma
MaxId	ne	ano	Mac OS	iOS	ano	zdarma
Smart Lock	ne	ano	Chrome OS	Android	ne	zdarma
Implementované řešení	ano	ano	Windows	Android	ne	zdarma

2 Praktická realizace

Tuto kapitolu jsem se rozhodla věnovat praktické realizaci celého řešení. Jedná se o řešení, které zahrnuje celkem čtyři komponenty: Službu pro systém Windows, WPF aplikaci, poskytovatele přihlašovacích údajů a aplikaci pro mobilní zařízení. Každou z nich vám přiblížím v následujících řádcích.

2.1 Výběr platforem aplikace a programovacího jazyka

Výběr platforem jednotlivých účastníků komunikace, ať už se jedná o mobilní telefon nebo osobní počítač, byl volen tak, aby nejlépe vyhovoval podmínkám použití mě a mých kolegů. Více výběr rozvedu v následujících kapitolách.

2.1.1 Aplikace pro mobilní telefon a programovací jazyk

Pro aplikaci na mobilní telefon jsem vybrala záměrně platformu Android, neboť dle statistik se jedná o nejprodávanější a nejrozšířenější platformu na světe. O tom se můžeme přesvědčit v tabulce 1.6 [13].

Tabulka 1.6: *Zastoupení operačních systémů mobilních telefonů na trhu*

Operační systém	2017 prodej [ks]	2017 zastoupení na trhu [%]	2016 prodej [ks]	2016 zastoupení na trhu [%]
Android	1 320 118	86	1 268 562	85
iOS	214 924	14	216 064	14
Ostatní OS	1 493	0	11 332	1
Celkem	1 536 535	100	1 495 958	100

Také můj osobní telefon je na platformě Android, tudíž volba pro mne byla jasná. Pro systém Android je možné vyvíjet například v jazycích jako jsou Kotlin, Xamarin nebo Java. Jelikož jsem již v minulosti několik softwarových řešení v jazyce Java psala, padla volba právě na ni. Z hlediska vývojového prostředí je také Java výhodná, neboť nabízí možnost vývoje v bezplatném vývojovém prostředí Android Studio.

2.1.2 Platforma osobního počítače a programovací jazyk

Zde byl můj výběr motivován myšlenkou, že aplikace bude sloužit především pro potřebu mou a mých kolegů, a proto jsem volila operační systém, který je u nás v zaměstnání používaný, a to je Windows 7 Professional x64. Služba pro operační systém Windows nabízí možnost využití nativních jazyků jako jsou C, C++ nebo jazyků postavených na platformě .NET, např. C#. Jelikož se s jazykem C# setkávám, jak v osobním životě, tak v zaměstnání, rozhodla jsem se právě pro tento. Mohla jsem jej využít taktéž pro vývoj WPF aplikace pro správu uživatelských účtů a přidružených instancí mobilních zařízení. U poskytovatele pověření byla nutnost využít k implementaci jazyk C++, neboť poskytovatel přihlašovacích údajů musí být řešen v nativním jazyce.

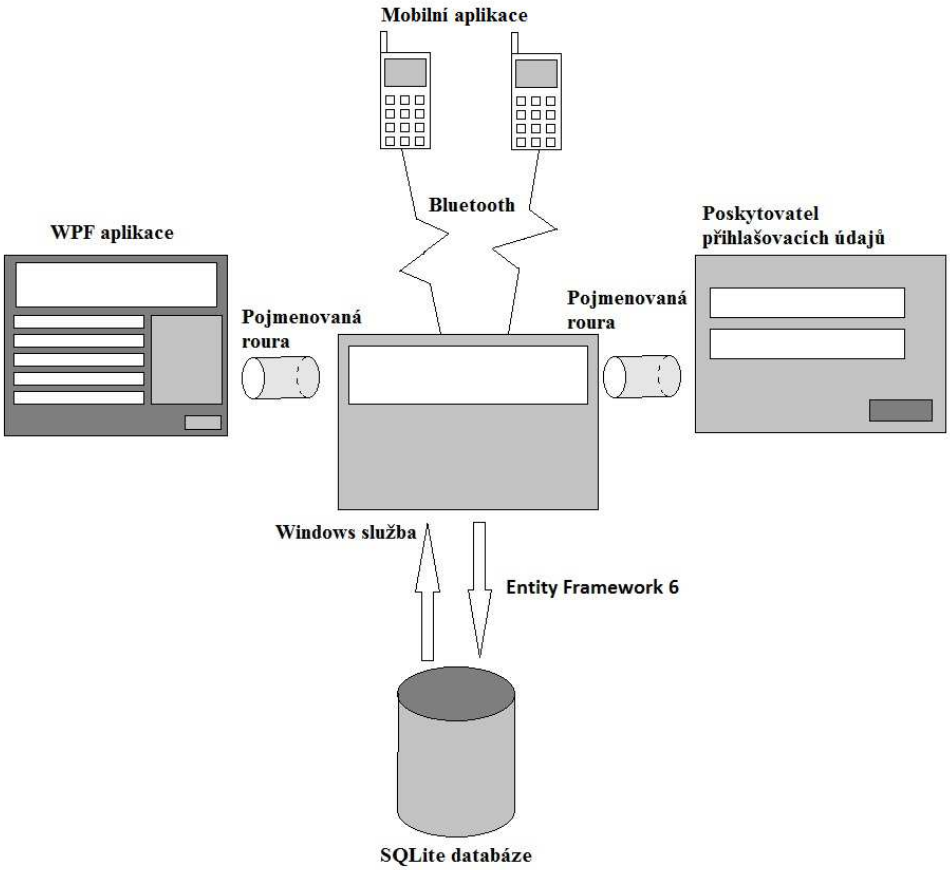
2.2 Schéma komponent

Systémové řešení je navrženo jako pět samostatných komponent, jež spolu navzájem komunikují. Jejich schematické zobrazení můžeme shlédnout na obrázku 1.4. Na platformě OS Windows jsou umístěny celkem čtyři komponenty, a to Windows služba, databáze typu SQLite, poskytovatel přihlašovacích údajů a WPF aplikace. Poslední pátá komponenta se nachází na platformě Android a jedná se o mobilní aplikaci. Způsoby komunikace mezi jednotlivými komponentami jsou popsány v tabulce 1.7.

Tabulka 1.7: *Způsob komunikace mezi komponentami*

komponenta	WPF aplikace	Poskytovatel přihlašovacích údajů	Databáze	Mobilní aplikace
Windows služba	pojmenovaná roura	pojmenovaná roura	Entity framework 6 (souborový systém)	Bluetooth

V následujících řádcích popíšu stručně funkce jednotlivých komponent. Detailněji budou tyto funkce popsány v následujících kapitolách.



Obrázek 1.4: *Schématické zobrazení komponent*

Mobilní aplikace

Mobilní aplikace zpracovává data ze senzorů mobilního telefonu a následně vyhodnocuje, zda se uživatel nevzdaluje od počítače. Tato komponenta také umožňuje uživateli okamžité zamknutí a odemknutí počítače. Tato část řešení funguje jako klientská část v socketovém spojení prostřednictvím technologie Bluetooth a řídí výměnu šifrovacích klíčů.

Windows služba

Tato komponenta funguje jako serverová část v komunikaci mezi mobilním telefonem a osobním počítačem a jako jediná komunikuje se všemi ostatními komponentami. Má za úkol zpracovávání a zasílání zpráv prostřednictvím socketového Bluetooth spojení mobilní aplikaci, vydává požadavky na zamykání a odemykání počítače a řídí ukládání konfiguračních dat mobilních telefonů do databáze. Windows služba také ověřuje funkčnost navázaného socketového Bluetooth spojení

WPF Aplikace

Tato komponenta poskytuje grafické uživatelské rozhraní a tímto umožňuje uživateli vybrat, který mobilní telefon bude schopen provádět vzdálenou autentizaci a konfigurační data těchto zařízení. Autentizační data uživatele zasílá Windows službě, aby je uložila do databáze.

Databáze

Databáze slouží k ukládání konfiguračních dat vybraných mobilních telefonů a přihlašovacích údajů jednotlivých uživatelů. Citlivá data jako je heslo jsou uložena v šifrované podobě, aby nemohlo dojít k jejich jednoduchému zneužití.

Poskytovatel přihlašovacích údajů

Tato komponenta poskytuje přihlašovací údaje uživatele OS Windows, aby mohlo proběhnout korektní odemčení počítače. Přihlašovací data jsou této komponentě předávána z databáze Windows službou. Poskytovatel přihlašovacích údajů předá Windows službě potvrzení, že přihlášení proběhlo korektně.

2.3 Výběr vhodných asynchronních vstupů a jejich zpracování

Pro detekci pohybu uživatele jsem se rozhodla po seznámení se s dokumentací vybrat senzor významného pohybu a senzor detekce kroků. Pro přijetí a zpracování dat zachycených senzory jsem využila v systému Android entitu typu `Service`, neboli služba. Služba v systému Android (v dalším textu v rámci této kapitoly název zkrátím na slovo služba) je aplikační komponenta, ve které mohou být prováděny dlouho trvající úkony na pozadí a neobsahuje uživatelské rozhraní. Jiná aplikační komponenta může službu spustit a ta bude pokračovat ve své práci na pozadí, i když uživatel přepne do jiné aplikace. S komponentou služba souvisí úzce také její speciální případ `IntentService`, která slouží k asynchronním úkonům vykonávaným na dotaz a sama se ukončí, jakmile splní všechny příkazy. Jelikož k vyhodnocování dat ze senzorů dochází plynule, tato aplikační komponenta není pro toto využití vhodná [12].

Služba tak jako aktivita obsahuje metodu `onCreate()`, ve které dochází k inicializaci dat a je volána pouze při samotném vytvoření služby. Pokud již služba běží, pak tato metoda volána není. Služba také obsahuje metodu `onStartCommand()`, která je volána ihned po jejím spuštění příkazem `startService()`. Pro inicializaci jsem využila tuto metodu neb obsahuje data, která potřebuji inicializovat po každém jejím spuštění. K zastavení služby může dojít dvěma způsoby, buď je zastavena

prostřednictvím jiné komponenty, k tomu slouží metoda `stopService()` nebo může služba zastavit sama sebe zavoláním metody `stopSelf()` [12].

Aby mohlo dojít ke zpracování dat přijatých senzory, je nutné, aby služba implementovala rozhraní `SensorEventListener`, které vyžaduje implementaci metod `onSensorChanged()` a `onAccuracyChanged()`. Rozdíl mezi nimi byl vysvětlen v kapitole 2.4. Pro své řešení jsem využila pouze první zmíněnou, neboť právě tato je vyvolána na základě zachycením dat senzory [12].
Zaměřila jsem se na detekci souvislé chůze uživatele, neboť se může stát, že uživatel učiní pouze dva kroky ke kolegovi a není potřeba počítač ihned zamykat. Také se může stát, že dojde k aktivitě, která svou dynamikou připomíná lidský krok (klepnutí telefonem, prudký pohyb nahoru a dolů) a tato je posléze příslušným způsobem vyhodnocena. Právě z tohoto důvodu jsem zvolila ukládání časového razítka jednotlivých kroků a počítání, kolik jich následuje bezprostředně za sebou. Vyvolání aktivity s dialogem pro uzamčení plochy je nastaveno na tři po sobě jdoucí kroky uživatele. Pokud jsou kroky uskutečněny v delším časovém sledu a jejich časová značka překročí stanovený limit, pak je počítadlo resetováno na nulu. Toto můžeme shlédnout v následující ukázce kódu:

```
Sensor sensor = sensorEvent.sensor;
if (sensor.getType() == Sensor.TYPE_STEP_DETECTOR) {
    long currentTime = System.nanoTime() / 1000000;
    if ((currentTime - timeStamp) < TWO_SECONDS) {
        steps++;
        if (steps > 1) {
            steps = 0;
            unregisterSensorListener();
            Intent mIntent = new Intent(this, NotificationActivity.class);
            startActivity(mIntent);
            stopSelf();
        }
    } else {
        steps = 0;
    }
    timeStamp = currentTime;
}
```

Mým původním záměrem bylo implementovat rozhodování k zamknutí plochy prostřednictvím obou z těchto senzorů, přičemž každý z nich by měl jinou váhu v rozhodování a bylo by nutné, aby oba detekovaly změnu. Avšak i přes veškerou snahu senzor významného pohybu na mém vlastním mobilním zařízení nefungoval zcela přesvědčivým způsobem a jeho detekce byla velmi nespolehlivá. Často docházelo k pozdním nebo žádným reakcím. Proto jsem se nakonec rozhodla využít pouze senzor detekce kroků, jehož výstup jsem podrobila dalšímu zpracování.

Jakmile tedy dojde k detekování souvislé chůze uživatele, je vyvolána aktivita zobrazující dialog pro uživatele a zároveň je ukončena činnost služby zachycující data ze senzorů, a to proto, aby nedocházelo ke stále novému spouštění zmíněné aktivity. Pro tuto aktivitu jsem vytvořila nový styl. Ten má pozadí nastaveno na průhledné, má odstraněn nadpis obrazovky, a je nastaven jako plovoucí okno. Tento styl způsobí, že se aktivita jeví jako průhledná. Bylo tak zvoleno čistě z důvodu estetického. Jediným úkolem této aktivity je zobrazení dialogu pro uživatele. Tento dialog je navržen jako instance typu `AlertDialog` obsahující tři tlačítka, zde uživatel může vybrat, zda plochu chce doopravdy zamknout, nechce zamknout či by tak rád učinil později.

Do budoucna by se dalo pro posouzení pohybu uživatele využít také senzoru pozice a pomocí něj detekovat případnou vzdálenost od osobního počítače. Po inicializačním spárování by došlo k uložení souřadnic počítače a na základě těchto údajů by poté docházelo k pravidelné kalkulaci vzdálenosti od zařízení.

2.4 Párování mobilní aplikace s desktopovou aplikací

Aby byl schopen mobilní telefon odemknout počítač a také jej zamknout, je nutné, aby byl tzv. spárován s WPF aplikací na počítači. Párování probíhá následujícím způsobem:

- Pro tyto kroky je nutné, aby Bluetooth adaptér na osobním počítači byl spuštěn a osobní počítač byl viditelný pro ostatní Bluetooth zařízení.
- Uživatel provede ve WPF aplikaci přidání nového uživatele, vyplní jeho uživatelské jméno a heslo a tyto údaje jsou uloženy do databáze zavoláním Windows služby.
- Uživatel prostřednictvím mobilní aplikace vyhledá osobní počítač.
- Výběrem zařízení ze seznamu je vyvolána výměna klíčů, jakmile výměna klíčů úspěšně proběhne, je zobrazena potvrzující zpráva
- Uživatel ve WPF aplikaci přidá mobilní zařízení (spárovaná mobilní zařízení jsou zobrazena v seznamu nalezených mobilních zařízení) tím, že jej vybere ze seznamu a uloží. Tímto dojde k uložení dat do databáze.
- Od této chvíle je mobilní telefon schopen zamykání a odemykání počítače

WPF aplikace je také opatřena funkcí zobrazování tzv. logů, tj. záznamů o všech provedených odemknutích a zamknutích prostřednictvím mobilní aplikace pro zvýšení bezpečnosti. Pro lepší přehlednost jsou snímky obrazovek při navigaci jednotlivých kroků uvedeny v příloze.

2.5 Komunikace Bluetooth a její zabezpečení

V následujících kapitolách bych se ráda věnovala komunikaci mezi mobilním zařízením a osobním počítačem. Ta je, jak již bylo zmíněno, řešena pomocí technologie Bluetooth a to především z důvodu její dostupnosti a ideálnímu dosahu.

2.5.1 Navázání komunikace

Aplikace je určena pro zařízení, která nejsou spárována, a to z především proto, že služba na platformě Windows je spuštěna pod účtem NT AUTHORITY/SYSTEM a spárování musí probíhat pod účtem uživatele, proto by tento krok nebyl možný. Komunikaci mezi osobním počítačem a mobilním zařízením probíhá prostřednictvím technologie Bluetooth, neboť technologie Bluetooth je lehce dostupná napříč spektrem mobilních zařízení a její podpora, jak v systému Android, tak v systému Windows, je velmi dobrá. Serverovou část komunikace byla delegována na osobní počítač a klientská část na mobilní telefon, neboť vize systému je taková, že jednomu osobnímu počítači může být připojeno větší množství klientských mobilních telefonů.

Serverová část je řešena následujícím způsobem. Při spuštění serverové části systému na platformě Windows je inicializovaná instance typu `BluetoothListener` a je vytvořeno nové vlákno, kde dochází k čekání na připojení klientské části běžící na platformě Android. Jakmile se klient připojí, spojení je jako instance typu `BluetoothClient` uloženo pro další zpracování. V této chvíli proběhne kontrola, zda již zařízení bylo v minulosti se serverovou částí spárováno a v databázi se tedy nachází

jeho konfigurační údaje, nebo se jedná o zcela nové zařízení. Dojde ke spuštění vlákna pro příjem zpráv a také vlákna, které má na starost vytváření pravidelných zpráv detekujících funkčnost spojení. K čekání na připojení klientských částí dochází nepřetržitě do té doby, dokud je serverová část aplikace spuštěna. Je tak voleno pro případ, že by mobilní zařízení ztratilo se službou kontakt. V této situaci se klientská část musí být schopna opětovně připojit k serverové části.

Klientská část na mobilním telefonu s platformou Android je řešena následovně. Proto, aby mohlo dojít k vytvoření spojení prostřednictvím technologie Bluetooth, je nutné vyžádat svolení uživatele. Tato povolení jsou uložena v manifestu aplikace. Jsou nutná následující dvě povolení sloužící pro umožnění vyhledávání Bluetooth zařízení a navázání komunikace:

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

K připojení k serverové části dojde při spuštění aplikace. Nejprve je získán a uložen Bluetooth adaptér přístroje jako instance typu `BluetoothAdapter`. Prostřednictvím něj je potom získáno vzdálené zařízení osobního počítače jako instance typu `BluetoothDevice`. Instance socketu Bluetooth typu `BluetoothSocket` je získána zavoláním metody `createInsecureRfcommSocketToServiceRecord()` na instanci vzdáleného zařízení. Tím dojde k inicializaci socketového spojení typu klient-server. Od této chvíle je po celou dobu spojení naslouchánо, zda nebyla prostřednictvím socketu obdržena zpráva. Pro spojení bylo vybráno nezabezpečené připojení, které je určeno pro nespárovaná zařízení.

2.5.2 Komunikace prostřednictvím zpráv

Aby mobilní zařízení a osobní počítač mohly vést komunikaci, byl sestaven systém rozhraní a modelů entit jednotlivých druhů zasílaných či přijímaných zpráv. Nejvýše v hierarchii stojí nejvíce abstraktní rozhraní `IPayload`, to je implementováno na nižší úrovni abstrakce rozhraními `IRequest` a `IResponse`. Jak název napovídá `IRequest` je určeno pro zprávy zasílané a `IResponse` pro zprávy přijímané. Tato dvě rozhraní jsou implementována jednotlivými druhy zpráv. Tyto slouží každá jako model pro určitý typ zprávy. Tato struktura zpráv byla vytvořena z důvodu usnadnění komunikace. Každá zpráva je po jejím sestavení serializovaná do objektu JSON řetězec, který je následně přenesen socketovým spojením a na straně druhé zpětně deserializován do instance objektu. Pro účely serializace a deserializace na platformě Windows byla vybrána knihovna `Newtonsoft.Json`, neboť umožňuje také serializaci a deserializaci abstraktních typů jako jsou rozhraní tím, že jejich konkrétní typ vloží do struktury řetězce ve formátu JSON. Na straně Android aplikace pak byla vybrána knihovna Flexjson. Bylo tak učiněno na základě faktu, že má také velmi dobrou podporu serializování a deserializování abstraktních typů. Serializéry na obou stranách musely být modifikovány tak, aby byly navzájem kompatibilní a umožňovaly vzájemnou serializaci a deserializaci napříč platformami .NET a Java.
Každá zpráva obsahuje unikátní identifikátor mobilního zařízení, který je datového typu UUID, jehož prostřednictvím dochází k ověřování uživatele. Jedná se o ochranu proti tzv. spoofing útoku, kdy je adresa zařízení útočníka záměrně zaměněna za adresu jednoho z klientů. První bajt zprávy je rezervován pro identifikátor šifrování, který je implementován jak na straně mobilního telefonu, tak na straně osobního počítače jako výčtový typ, který může nabývat následujících hodnot:

```
EncryptionAes   0xFF
EncryptionRsa   0xF0
PlainText       0x00
```

Typ zprávy je také řešen pomocí výčtového typu, avšak typů zpráv je poněkud více nežli způsobů šifrování, a proto jeho výčet zde uváděn nebudu.

2.5.3 Šifrování a zabezpečení komunikace

Jak bylo řečeno v kapitole 2.4.1, komunikace mezi mobilním telefonem a osobním počítačem probíhá prostřednictvím nezabezpečeného kanálu a je tedy potřebné tuto komunikaci odpovídajícím způsobem zabezpečit, neboť by mohlo dojít ke zneužití přihlašovacích údajů pro ovládnutí počítače. K tomuto účelu byla vybrána symetrická kryptografie AES. Jedná se o kryptografický standard, který je stále aktivně využíván, je považován za bezpečný a dostatečně výkonný, aby mohl být použit k šifrování komunikace v reálném čase. Zde však přetrvával problém s počáteční výměnou šifrovacích klíčů, pro kterou je v případě AES kryptografie potřebný zabezpečený kanál, který pro tuto aplikaci nebyl dostupný. Za účelem prvotní výměny klíčů byl tedy vybrán kryptografický systém RSA. Ten umožňuje provést výměnu symetrického AES klíče také prostřednictvím nezabezpečeného kanálu. Klíče jsou přenášeny a uchovávány ve formátu Base64. Formát Base64 je způsob uložení binární informace tak, aby výsledná data obsahovala pouze tisknutelné znaky. Výměna klíčů je vždy inicializována ze strany klientské části mobilního telefonu a obsahuje následující kroky [7] [8] [15]:

- **Mobilní telefon:** V tomto kroku je vytvořena mobilním telefonem zpráva typu `RsaRequest`, která není šifrovaná, tedy první bajt těla zprávy je nastaven na hodnotu `0x00`, což signalizuje nešifrovaný text.
- **Osobní počítač:** Osobní počítač má implementován vlastní certifikát, který obsahuje dvojici RSA klíčů tedy, jak klíč veřejný, tak klíč soukromý. Vytvoří tedy odpověď typu `RsaResponse` a do ní vloží veřejný klíč z certifikátu ve formátu řetězce zakódovaného do Base64 formátu. Zpráva není šifrovaná, proto první bajt těla zprávy nastaví na hodnotu `0x00`.
- **Mobilní telefon:** Mobilní telefon přijme zprávu, převezme z ní veřejný klíč, dekóduje jej z Base64 řetězce a převede na formát `AsymetricKeyParameter`. Zařízení vytvoří zprávu typu `AesKeyRequest`. Dále vygeneruje symetrický AES klíč, zakóduje jej opět do Base64 řetězce a zašifruje prostřednictvím přijatého veřejného klíče. První bajt těla zprávy nastaví na hodnotu `0xF0`, což signalizuje RSA šifrování.
- **Osobní počítač:** Osobní počítač zprávu přijme a podle prvního bajtu vyhodnotí, že se jedná o zprávu šifrovanou RSA šifrou. Na základě tohoto dešifruje zprávu svým soukromým klíčem a získaný AES klíč uloží ve formátu řetězce Base64. V tuto chvíli nevytváří žádnou odpověď ani potvrzení a vyčkává na další zprávu.
- **Mobilní telefon:** Mobilní telefon odešle ověřovací zprávu, aby zjistil, zda výměna klíčů proběhla úspěšně. Tato zpráva je typu `EchoRequest` a obsahuje náhodný řetězec šifrovaný symetrickým klíčem, který by v tuto chvíli měli mít oba účastníci komunikace stejný. První bajt zprávy nastaví na hodnotu `0xFF`, což signalizuje šifrování AES kryptografií a zprávu odešle.
- **Osobní počítač:** Osobní počítač zprávu přijme, zjistí, že je zašifrovaná pomocí AES klíče. Následně zprávu dešifruje klíčem přijatým v předchozí komunikaci a opět prostřednictvím

toho samého klíče zašifruje zpět, nastaví první bajt zprávy na hodnotu `0xFF` a odešle zpět mobilnímu telefonu.

- **Mobilní telefon:** Mobilní telefon zprávu přijme, na základě prvního bajtu zjistí, že je šifrovaná prostřednictvím AES klíče, dešifruje ji a porovná s řetězcem, který byl odeslán v předešlé zprávě. Pokud se tyto shodují je výměna klíčů považována za úspěšnou.

Jako náhodný řetězec byl zvolen náhodně generovaný unikátní identifikátor datového typu UUID převeden do řetězce, neboť je zde pravděpodobnost blížící se nule (konkrétně je tato pravděpodobnost 2^{-122} [14]), že dojde ke kolizi dvou stejných UUID hodnot na obou stranách komunikace. Pro jednu výměnu klíčů jsou zvoleny vždy celkem tři pokusy. Pokud dojde v průběhu popsané sekvence ke kolizi, která může být způsobená opožděním zprávy od osobního počítače nebo neshodnými náhodnými řetězci, pak vždy následuje další pokus, který prochází celou sekvenci od počátku.

V případě, že by došlo k velkému zpoždění komunikace, může dojít k tzv. dead locku neboli k situaci, kdy osobní počítač bude čekat na zprávu od mobilního telefonu, a naopak mobilní telefon bude čekat na odpověď od osobního počítače. Pro takovou možnost byla implementována funkce omezeného čekání na straně mobilního telefonu. Je nastavena na dobu 500 ms. Pokud v tomto čase nedojde k přijetí odpovědi, pokus výměny klíčů se opakuje, dokud nedosáhne celkového počtu tří pokusů.

Celá komunikace je dále opatřena funkcí autentizace mobilního telefonu. Každá zpráva, která je osobním počítačem přijata, obsahuje také unikátní identifikátor mobilního telefonu. Tento identifikátor je osobním počítačem uložen do databáze ihned při prvotním spárování. Před jakýmkoliv zpracováním zprávy dojde k ověření, zda se identifikátor ve zprávě a adresa, ze které byla zpráva přijata, shodují s údaji uloženými v databázi.

2.5.4 Volba knihoven a implementace šifrování

Pro symetrickou kryptografii AES nabízí framework .NET na straně osobního počítače vestavěnou knihovnu `System.Security.Cryptography`, která poskytuje veškeré potřebné funkce pro implementaci AES šifrování. Pro potřeby AES kryptografie na straně mobilní aplikace byla volena knihovna `javax.crypto`, která poskytuje veškeré potřebné metody a třídy pro implementaci.

Pro asymetrickou kryptografii RSA bylo využito knihovny `BouncyCastle`, která je dostupná jak v programovacím jazyce C#, tak v programovacím jazyce Java a v obou jazycích poskytuje celou řadu metod a tříd. Právě tento fakt byl hlavní motivací k jejímu využití.

AES - Generování klíčů

Generování klíče pro symetrickou kryptografii vždy probíhá jen a pouze na straně mobilního telefonu tedy v systému Android. K tomu Java poskytuje knihovnu `javax.crypto`, v níž se nachází třída `KeyGenerator`, pomocí níž lze generování klíčů realizovat. Instanci lze získat zavoláním statické metody `getInstance()` na třídě `KeyGenerator`. Je nutné provést jeho inicializaci zavoláním metody `init()` s parametrem velikosti šifrovacího klíče. Velikost šifrovacího klíče byla zvolena 128 bitů, neboť se jedná o klíč nejmenší délky, a tedy dochází k nejrychlejšímu zpracování textu. Ke generování klíče dojde při zavolání metody `generateKey()` a tento je pak typu `SecretKey`. Klíč je možné převést do pole bajtů zavoláním metody `getEncoded()`. Toto si lze prohlédnout v následující ukázce kódu:

```
public byte[] getNewKey(AesKeySize keySize) {
    KeyGenerator keyGenerator = null;
    try {
        keyGenerator = KeyGenerator.getInstance("AES");
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    keyGenerator.init(keySize.getId());
    SecretKey key = keyGenerator.generateKey();

    return key.getEncoded();
}
```

AES – šifrování

Symetrickou kryptografii AES využívá při komunikaci, jak mobilní aplikace, tak služba na straně osobního počítače. V mobilní aplikaci šifrování probíhá následujícím způsobem. K inicializaci šifrovacího nástroje je potřeba tří parametrů – mód, ve kterém má být objekt inicializován, šifrovací klíč a inicializační vektor. Módy jsou celkem dva – šifrování, prezentován konstantou `ENCRYPT_MODE` a dešifrování, prezentován konstantou `DECRYPT_MODE`. Šifrovací klíč musí být ve formátu `SecretKeySpec`. Inicializační vektor je náhodně generované pole bajtů. To lze získat prostřednictvím instance typu `SecureRandom` a metody, kterou nabízí `nextBytes()`.

Takto inicializovaný objekt typu `Cipher` lze použít k samotnému šifrování. To je realizováno metodou `doFinal()`, kterou třída `Cipher` nabízí. Jako parametr je nutno vložit zprávu, která má být šifrovaná, a to ve formátu pole bajtů. Celou sekvenci ilustruje následující ukázka kódu:

```
SecureRandom random = new SecureRandom();
SecretKeySpec keyAes = new SecretKeySpec(key, "AES");
byte[] initializationVectorByte = new byte[key.length];
random.nextBytes(initializationVectorByte);
IvParameterSpec initializationVector = new
        IvParameterSpec(initializationVectorByte);
Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, keyAes, initializationVector);

//encrypt message
byte[] encryptedMessage = cipher.doFinal(message.getBytes());

//copy initialization vector to message
messageCombInitVect = new byte[encryptedMessage.length +
initializationVectorByte.length];
System.arraycopy(initializationVectorByte, 0, messageCombInitVect,
        0, initializationVectorByte.length);
System.arraycopy(encryptedMessage, 0, messageCombInitVect,
        initializationVectorByte.length, encryptedMessage.length);
```

Na straně osobního počítače probíhá šifrování následovně. Šifrování je uskutečněno využitím instance typu `Aes`, která má vlastnosti `Key` – šifrovací klíč a `IV` – inicializační vektor. K vygenerování inicializačního vektoru dojde zavoláním metody `GenerateIV()`, kterou třída `Aes` implementuje. Samotné šifrování probíhá prostřednictvím několika vnořených proudů. Toto opět ilustruje následující ukázka kódu:

```
//initialization of cipherizer
aesCrypto.Key = key;
aesCrypto.GenerateIV();
initVector = aesCrypto.IV;
aesCrypto.Mode = CipherMode.CBC;
var encryptor = aesCrypto.CreateEncryptor(aesCrypto.Key,
        aesCrypto.IV);

//encryption
using (var msEncrypt = new MemoryStream())
{
   using (var csEncrypt = new CryptoStream(msEncrypt, encryptor,
      CryptoStreamMode.Write))
    {
 using (var swEncrypt = new StreamWriter(csEncrypt))
                              {
                                   swEncrypt.Write(textToEncrypt);
                              }
                              encrypted = msEncrypt.ToArray();
                         }
                    }
```

AES – dešifrování

Dešifrování zpráv na straně mobilní aplikace probíhá podobným způsobem jako šifrování, jen je dešifrovací nástroj inicializován v dešifrovacím módu.

Na straně osobního počítače je situace obdobná. Opět musí dojít k inicializaci ovšem tentokrát dešifrovacího nástroje.

RSA – Generování klíčů

Ke generování páru veřejného a soukromého klíče pro metodu RSA v aplikaci nedochází. Tento pár je již vygenerován a uložen prostřednictvím certifikátu na straně osobního počítače. Z tohoto certifikátu jsou oba klíče v případě potřeby získávány. Klíče je možné získávat opět v několika velikostech. Pro potřeby aplikace byla zvolena velikost klíče 2048 bitů, jedná se o velikost doporučenou Národním institutem pro standardy a technologie (NIST) v USA. Certifikát lze načíst do entity typu `X509Certificate2`, který se nachází v knihovně `System.Security.Cryptography`. Třída `X509Certificate2` obsahuje vlastnost `PrivateKey`. V následující ukázce kódu je demonstrován popsaný postup [16].

```
X509Certificate2 cert = new
          X509Certificate2(AppDomain.CurrentDomain.BaseDirectory +
          "VCDevelopmentRemote.pfx", "Password01",
          X509KeyStorageFlags.Exportable);
rsaKeyPair = DotNetUtilities.GetKeyPair(cert.PrivateKey);
```

RSA – šifrování veřejným klíčem

Šifrování veřejným klíčem je využito pouze na straně mobilní aplikace, která tímto způsobem bezpečně doručí AES klíč k osobnímu počítači. Nejprve je nutné opět provést inicializaci šifrovacího nástroje s parametrem veřejného klíče. Ukázku lze vidět v následujících řádcích:

```
RSAEngine rsaEngine = new RSAEngine();
PKCS1Encoding rsaEncoder = new PKCS1Encoding(rsaEngine);
rsaEncoder.init(true, publicKey);
```

Cele šifrování je provedeno po blocích, neboť velikost textu, která může být šifrována je omezena velikostí šifrovacího klíče. Při šifrování je nutné vždy zadat odkud kam se šifruje, tedy vymezení šifrovaných dat ve zprávě. Poslední blok je šifrován odlišným způsobem, neboť se může stát, že nemá stejnou délku jako bloky předchozí. Jednotlivé zašifrované bloky se následně ukládají do pole bajtů, které v tomto případě funguje jako zásobník.

RSA – dešifrování soukromým klíčem

Dešifrování probíhá jen a pouze na straně osobního počítače a opět probíhá po blocích analogickým způsobem jako šifrování. Opět zde platí, že na začátku je nutné inicializovat dešifrovací nástroj a poslední blok je dešifrován jiným způsobem z důvodu jeho nestandardní délky. Postupně dešifrované bloky jsou kopírovány do pole bajtů.

2.6 Princip funkce automatické autentizace

Celkové řešení úlohy nabízí, jak uzamčení, tak odemčení plochy osobního počítače. Tyto procedury mohou být provedeny zcela automaticky nebo na vyžádání uživatele. V následujících kapitolách bude popsáno, jak může k oběma případům dojít a jakým způsobem jsou metody implementované.

2.6.1 Automatické uzamknutí a odemknutí plochy

Jako impuls k automatickému zamčení plochy osobního počítače bez dotazování uživatele byl zvolen okamžik, kdy se uživatel vzdálí s mobilním telefonem do takové vzdálenosti, že přestane být v dosahu Bluetooth signálu osobního počítače. Tento okamžik je detekován prostřednictvím cyklicky zasílaných zpráv detekujících funkčnost spojení.

- Osobní počítač vyšle šifrovanou zprávu typu `EchoRequest` a očekává v časovém intervalu 2000 ms odpověď ve formátu `EchoResponse`. Tato zpráva obsahuje náhodně generovaný řetězec.
- Mobilní telefon zprávu přijme, dešifruje prostřednictvím symetrického klíče a vytvoří zprávu typu `EchoResponse`, zašifruje a pošle zpět.

- Osobní počítač zprávu dešifruje a provede kontrolu řetězců, pokud jsou shodné, čeká na zaslání další `EchoRequest` zprávy. Pokud se řetězce neshodují, odešle mobilnímu telefonu požadavek na vyvolání výměny klíčů.

Na každou odpověď čeká osobní počítač po dobu 2000 ms, pokud ve stanoveném čase odpověď nepřijde, opakuje pokus odesláním další zprávy `EchoRequest`. Pokud ani po druhém uskutečněném pokusu nepřijme osobní počítač očekávanou odpověď, je situace vyhodnocena tak, že se mobilní telefon nenachází v dosahu osobního počítače a ten je uzamčen. Přijímání zpráv je řešeno pomocí fronty. V případě, kdy mobilní telefon vyhodnotí, že došlo k odpojení ze socketu, snaží se cyklicky k socketu připojit. Takto se stane, jakmile se opět dostane do dosahu Bluetooth. V tento okamžik opět dojde k navázání spojení a osobní počítač začne cyklicky zasílat `EchoRequest` zprávy. Pokud vyhodnotí, že je počítač uzamčen a na dvě po sobě jdoucí `EchoRequest` zprávy obdrží od mobilního telefonu odpovídající `EchoResponse` odpovědi, je počítač automaticky odemčen.

2.6.2 Uzamknutí a odemknutí na vyžádání

Uživatel má také možnost uzamknout a odemknout plochu osobního počítače prostřednictvím notifikace aplikace v mobilním telefonu nebo dialogu, který je zobrazován na základě zpracování dat ze senzorů.

Notifikace

Při spuštění mobilní aplikace dojde k aktivací zobrazení notifikace. Tato notifikace obsahuje vždy aktivní tlačítko pro zamknutí či odemknutí plochy počítače podle toho, v jakém stavu se momentálně nachází. Tento stav mobilní telefon získá z cyklicky zasílaných `EchoRequest` zpráv, které tento údaj obsahují. Dle tohoto údaje je zobrazena buď notifikace zamknutí nebo notifikace odemknutí.

Pokud je počítač odemčen, je zobrazena notifikace, zda si uživatel přeje plochu zamknout. Při stisku tlačítka „Yes“ dojde k odeslání zprávy typu `LockRequest` osobnímu počítači, k jeho uzamčení a zastavení služby zpracovávající data ze senzorů. Pokud je počítač uzamčen, je zobrazena notifikace, zda si uživatel přeje počítač odemknout. Při stisku tlačítka „Yes“ dojde k odeslání zprávy typu `UnlockRequest` osobnímu počítači, k jeho odemčení a znovuspuštění služby zpracovávající data ze senzorů.

Dialog

Uživatel je schopen také zamknout osobní počítač na základě vyhodnocení dat ze senzorů pohybu. Jakmile dojde k vyhodnocení, že se uživatel pohybuje souvislou chůzí, pak je ukončena činnost služby zpracovávající data ze senzorů a je zobrazen dialog se třemi volbami. Volby a jejich funkce jsou následující:

Tlačítko „Yes“ – Při stisknutí tohoto tlačítka dojde k ukončení dialogu, odeslání zprávy typu `LockRequest` osobnímu počítači.

Tlačítko „No“ – Při stisknutí tohoto tlačítka dojde k ukončení dialogu a ke znovuspuštění služby zpracovávající data ze senzorů.

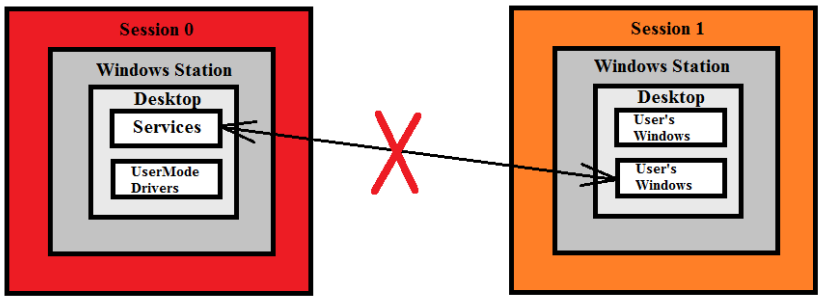
Tlačítko „Later“ – Při stisknutí tohoto tlačítka dojde k ukončení dialogu.

2.6.3 Detailní pohled na implementaci zamykání

Zamykání uživatelovy pracovní relace je v operačním systému Windows řešeno nativní funkcí `LockWorkstation()`. Tato funkce musí být vyvolána z kontextu uživatelovy relace, aby

k uzamčení došlo. Vzhledem k faktu, že serverová služba bude jako Windows služba spuštěna pod jinou relací, než je relace uživatele, je zapotřebí pro dosažení cíle překlenout problém s tím související. Tento problém je schematicky znázorněn na obrázku 1.5. Serverová služba v tomto případě bude muset pro uzamčení uživatelovy relace vyvolat funkci `LockWorkstation()` v kontextu této relace. Serverová služba může pro tento úkon využít těch možností, že je spuštěna pod oprávněním lokálního systémového účtu `NT AUTHORITY/SYSTEM`, který disponuje neomezenými právy pro operace s přihlašovacími uživatelskými tokeny v operačním systému Windows.
Postup po vyvolání funkce `LockWorkstation()` v relaci uživatele bude tedy následující:

- Nalezení vhodného spuštěného procesu, který běží v relaci uživatele a je tímto uživatelem spuštěn.
- Extrahování uživatelova přihlašovacího tokenu z procesu získaného v předchozím bodě pomocí funkce `OpenProcessToken()`.
- Získaný token z předchozího kroku je tzv. impersonizační token, který je nutno překonvertovat na token primární. K tomuto účelu je využita Windows API funkce `DuplicateTokenEx()`.
- Vytvoření nového procesu v relaci uživatele s využitím primárního tokenu uživatele pomocí funkce `CreateProcessAsUser()`.



Obrázek 1.5: *Znázornění problému komunikace mezi dvěma relacemi*

Vzhledem k existenci vhodného způsobu vyvolání funkce `LockWorkstation()` pomocí pomocného vestavěného programu ve všech verzích OS Windows s názvem RunDLL32.exe nebylo nutné vytvořit vlastní primitivní aplikaci, která by funkci `LockWorkstation()` vyvolala.

2.6.4 Detailní pohled na implementaci odemykání

OS Windows používal do verze Windows XP autentizační balíček MS Gina. Tento balíček umožňoval automatickou autentizaci uživatele pouze pomocí jeho přímé modifikace. Vzhledem k tomu, že OS Windows XP není již od roku 2014 podporován, bylo nutné pro účely automatické autentizace implementovat rozhraní nového způsobu přihlašování do OS Windows. Tento způsob je již od verze Windows Vista řešen pomocí tzv. poskytovatele přihlašovacích údajů (Credential Provider). K zajištění implementace automatické autentizace bylo tedy zapotřebí implementovat vlastního poskytovatele přihlašovacích údajů. Poskytovatel přihlašovacích údajů je v OS Windows realizován jako separátní dynamická knihovna implementující rozhraní `ICredentialProvider`. Tato dynamická knihovna je pak zaregistrována v registru OS Windows jako samostatný poskytovatel přihlašovacích údajů a současně musí být umístěna do některé ze známých systémových složek (typicky `C:\Windows\System32\`) [17] [18].

Rozhraní `ICredentialProvider` umožňuje definovat vlastní dlaždice na přihlašovací obrazovce OS Windows. Každý poskytovatel přihlašovacích údajů může těchto dlaždic definovat několik. Každá z těchto dlaždic je interně v poskytovateli přihlašovacích údajů realizována jako implementace rozhraní `ICredentialProviderCredential`. Toto rozhraní obsahuje primitivní metody pro nastavování a získávání dat z jednotlivých prvků grafického uživatelského rozhraní na přihlašovací obrazovce OS Windows [17] [18].

Jednotlivé funkce poskytovatele přihlašovacích údajů, které jsou implementovány v rozhraní `ICredentialProvider` případně `ICredentialProviderCredential`, jsou vyvolány samotným operačním systém pomocí procesu přihlašovací obrazovky (`LogonUI.exe`). Pro komunikaci mezi poskytovatelem přihlašovacích údajů a serverovou službou byla využita pojmenovaná roura. Samotný proces automatické autentizace tedy vypadá přibližně následovně:

- Po uzamčení relace uživatele zajistí OS spuštění nové instance procesu `LogonUI.exe`. Toto způsobí i načtení všech poskytovatelů přihlašovacích údajů a tím i načtení námi vytvořeného poskytovatele přihlašovacích údajů.
- Poskytovatel přihlašovacích údajů vytvoří serverovou část pojmenované roury a vyčkává na připojení od klienta. Připojení od klienta je realizováno pouze tehdy, jeli v serverové službě zaregistrován požadavek na odemčení počítače.
- Serverová služba zašle poskytovateli přihlašovacích údajů požadavek na přihlášení konkrétního uživatele a předá mu jeho přihlašovací údaje.
- Poskytovatel přihlašovacích údajů zpracuje vstupní data a vyvolá na rozhraní `ICredentialProviderEvents` funkci `CredentialsChanged()`. Tím dojde k předání požadavku na reenumerizaci jednotlivých dlaždic zobrazených na přihlašovací obrazovce.
- V průběhu reenumerizace dojde k přeposlání příznaku charakterizujícího požadavek na automatické přihlášení do procesu přihlašovací obrazovky LogonUI.exe.
- Proces přihlašovací obrazovky LogonUI.exe po zjištění příznaku charakterizujícího požadavek na přihlášení zastaví proces reenumerizace a ihned využije na dlaždici reprezentované implementací rozhraní `ICredentialProviderCredential` funkci `GetSerialization()`.
- Proces přihlašovací obrazovky získané přihlašovací údaje použije pro přihlášení do systému.
- Proces přihlašovací obrazovky předá informace o úspěšnosti či neúspěšnosti pokusu přihlášení zpět do poskytovatele přihlašovacích údajů a to konkrétně v jeho implementaci rozhraní `ICredentialProvider` vyvoláním funkce `ReportResult()`. Tímto je celý proces ukončen.

2.7 Řešení meziprocesové komunikace

Vzhledem ke skutečnosti, že rozdílné procesy na platformě operačního systému Windows nesdílejí adresní prostor, tak je nutné vhodným individuálním způsobem zajistit komunikaci mezi nimi. Taktéž další faktor, který je nutno vzít v úvahu, je nemožnost přímé komunikace serverové služby s administrativní řídící aplikací. Na platformě OS Windows je totiž již od OS Windows Vista zavedena striktní izolace mezi službami operačního systému, které běží pod vysoce privilegovaným uživatelským účtem a ostatními uživatelskými aplikacemi. Z tohoto důvodu nelze pro tuto komunikaci využít například dobře zavedených Windows Messages pro komunikaci mezi okny. Je nutno využít služeb

meziprocesové komunikace. Na platformě operačního systému Windows je implementováno několik způsobů, jak meziprocesovou komunikaci řešit. V následujících odstavcích si popíšeme několik z nich.

2.7.1 Rešerše vhodných technologií meziprocesové komunikace

V následujících odstavcích se budu věnovat rozborům technologií, které by případně byly vhodné pro využití v meziprocesové komunikaci.

Microsoft RPC

Implementace vzdáleného volání procedur od společnosti Microsoft umožňuje vyvolat funkci a předat jí parametry z jiného procesu, než ve kterém bude funkce vyvolána. Díky této technologii tak lze vliv rozdílného adresního prostoru jednotlivých procesů zcela eliminovat. Implementace Microsoft RPC technologie využívá nativních funkcí dostupných přes Windows API a umožňuje aplikacím psaným v nativním jazyce komunikovat mezi procesy. Funkce, které mají být vyvolány, jsou definovány v jazyce MIDL, který vytváří potřebná rozhraní mezi klientem a serverem. Vzhledem ke skutečnosti, že serverová část na platformě operačního systému Windows je implementována ve frameworku .NET, je však použití technologie Microsoft RPC nevhodné [19].

Síťové spojení přes sockety

Síťové spojení přes sockety umožňuje meziprocesovou komunikaci díky vytvoření socketového spojení přes protokol TCP/IP. Jednotlivé aplikace pak spolu dokáží komunikovat stejným způsobem, jako by se nacházely na místní síti případně v rámci celého internetu. Toto řešení tedy vytváří robustní způsob, jak lze meziprocesovou komunikaci řešit. Vzhledem k tomu, že zadání úlohy nespecifikuje konkrétní hardwarové vybavení serverové části řešení, je však nutno vzít v úvahu také možnost nekompatibility hardwaru stroje, na kterém serverová část poběží. Konkrétně tato nekompatibilita může existovat v tom, že serverová stanice teoreticky nemusí obsahovat žádné síťové adaptéry, v takovém případě by komunikace pomocí socketového spojení nebyla možná.

Pojmenovaná roura

Pojmenovaná roura představuje jednu z dalších možností meziprocesové komunikace na platformě operačního systému Windows. Pojmenovaná roura může pracovat v jednocestném nebo duplexním režimu a umožňuje podobně jako socketové síťové spojení posílat a přijímat data. Na rozdíl od anonymní roury, která umožňuje meziprocesovou komunikaci pouze mezi procesy ve vztahu rodič potomek, umožňuje pojmenovaná roura komunikaci mezi libovolnými procesy, neboť k navázání spojení pomocí pojmenované roury stačí znát její jméno. Komunikace pomocí pojmenované roury je řešena na straně jednoho procesu vytvořením pojmenované roury jako server a na straně procesu druhého připojení se k pojmenované rouře jako klient. Pojmenovaná roura na platformě OS Windows umožňuje komunikaci pomocí dvou módů. V prvním módu dochází k zasílání jednotlivých bajtů, které v závislosti na velikosti vyrovnávací paměti nemusí představovat celkovou zprávu. V tomto módu je zodpovědností aplikace, aby data zpracovala a rozpoznala, kdy jsou všechny bajty zprávy přijaty. Druhým přenosovým módem je komunikace po zprávách. Operační systém zajistí, aby koncovému klientovi dorazila vždy zpráva celá. Tedy jeden zápis dat do pojmenované roury je vždy přijat jako celek na výstupní straně, a to zcela nezávisle na velikosti vyrovnávací paměti [20].

2.7.2 **Použité řešení meziprocesové komunikace**

Jako nejvhodnější technologické řešení pro meziprocesovou komunikaci mezi serverovou službou a poskytovatelem přihlašovacích údajů byla zvolena pojmenovaná roura. Ta se v konečném řešení používá, jak při komunikaci mezi serverovou službou a poskytovatelem přihlašovacích údajů, tak pro komunikaci mezi serverovou službou a administrativní řídící aplikací. Jako nejvhodnější způsob pojmenování pojmenované roury se jevil již výše popsaný UUID identifikátor, který by měl zamezit jisté kolizi jmen. Pro komunikaci mezi serverovou službou a poskytovatelem přihlašovacích údajů byly vytvořeny dvě pojmenované roury. První z nich slouží pro případ použití poskytovatele přihlašovacích údajů pro přihlášení nového uživatele a druhá z nich slouží pro odemčení uzamčeného počítače již přihlášeného uživatele. Toto řešení bylo nezbytné pro dosažení stability komunikace během přepínání uživatelů při scénáři, kdy jeden uživatel je přihlášen, ale počítač je uzamčen a druhý jiný uživatel se pokouší k počítači přihlásit. V takovém případě by při použití pojmenované roury jediné, mohlo dojít k předání dat sloužících pro přihlášení nevhodnému poskytovateli přihlašovacích údajů. Další využití pojmenované roury je při komunikaci mezi serverovou službou a administrativní řídící aplikací. V tomto případě je již použita pouze jedna instance pojmenované roury.

2.8 **Testování**

Řešení bylo otestováno na několika platformách mobilních telefonů a operačních systému. Cílem testování bylo odhalit celkovou spolehlivost aplikace včetně možnosti konkurenčního využití při spojení více mobilních aplikací s jedním serverem. Situace je nejlépe zachycena v následujících tabulkách 1.8 a 1.9. V rámci této aktivity byla vyzkoušena celá řada testovacích scénářů. Některé z nich znázorňuje následující seznam:

- Testování spolehlivosti párovací metodiky při výměně klíčů pro šifrování mezi klientskou stanicí a serverem.
- Testování spolehlivosti při navázání spojení klientské části s částí serverovou.
- Testování odolnosti serverové části aplikace na odebírání spárovaných zařízení, taktéž na katastrofální selhání přenosové vrstvy.
- Testování spolehlivosti při iniciaci zamykací a odemykací sekvence prostřednictvím notifikace v mobilním telefonu.
- Testování spolehlivosti registrace události z asynchronního vstupu ve formě detektoru kroku prostřednictvím vyvolaného modálního dialogu v mobilním telefonu.
- Testování spolehlivosti při odchodu uživatele z aktivní zóny dosahu signálu Bluetooth a jeho opětovném návratu.

Řešení bylo testováno na počítačích s platformami Windows 7 a Windows Vista při využití virtualizačního nástroje VMware Player. Na straně klientské části byly využity mobilní telefony vybaveny operačním systémem Android v několika verzích a sestaveních. Tímto by mělo dojít k úplnému pokrytí všech testovacích případů a případů použití mobilní aplikace se serverovou částí. Navržené řešení bylo otestováno na následujících mobilních zařízeních uvedených v tabulce 1.8.

Tabulka 1.8: *Přehled testovaných mobilních zařízení*

Typ telefonu	Operační systém	Průběh
Xiaomi Redmi Note 4	Android 7.0 NRD90M	OK
Samsung Galaxy J3	Android 5.1.1	OK

V průběhu testování se ukázalo, že operační systém Windows Vista vzdoruje instalaci pod virtualizačním nástrojem Vmware Player. Při hlubší investigaci bylo zjištěno, že instalace selhává s kódem modré obrazovky 0x3B. Z tohoto důvodu nebylo možné řešení na platformě operačního systému Windows Vista řádně otestovat. Ve všech definovaných testovacích případech vykazovala aplikace maximální spolehlivost a stabilitu, tudíž lze předpokládat, že v případě komerčního nasazení vyvinutého řešení bude aplikace dostatečně konkurenceschopná. Jednotlivé průběhy a výsledky testování charakterizují tabulky 1.8 a 1.9.

Tabulka 1.9: *Přehled testovaných operačních systémů*

Operační systém	Číslo sestavení	Průběh
Microsoft Windows 7 Home Basic	6.1.7601 Service Pack 1 Build 7601 x86	OK
Microsoft Windows 7 Home Premium	6.1.7601 Service Pack 1 Build 7601 x86	OK
Microsoft Windows 7 Professional	6.1.7601 Service Pack 1 Build 7601 x86	OK
Microsoft Windows 7 Ultimate	6.1.7601 Service Pack 1 Build 7601 x86	OK
Microsoft Windows 7 Home Basic	6.1.7601 Service Pack 1 Build 7601 x64	OK
Microsoft Windows 7 Home Premium	6.1.7601 Service Pack 1 Build 7601 x64	OK
Microsoft Windows 7 Professional	6.1.7601 Service Pack 1 Build 7601 x64	OK
Microsoft Windows 7 Ultimate	6.1.7601 Service Pack 1 Build 7601 x64	OK

Závěr

Aplikací pro usnadnění vzdálené autentizace existuje na trhu celá řada, avšak pouze velmi málo z nich umožňuje jak odemykání počítače, tak jeho zamykání. Implementované řešení spojuje obě funkce a ve srovnání s jinými aplikacemi podobného určení nabízí navíc několik způsobů, jak odemčení či zamčení počítače dosáhnout. Tímto aplikace zcela splnila zamýšlený účel, velmi zjednodušila vzdálenou autentizaci uživatele a umožnila mu uzamknutí plochy také v případech, kdy se již nenachází v bezprostřední blízkosti svého počítače. To vše s důrazem na maximální bezpečnost, která je zde zcela na místě.

Celé řešení se sestává z pěti komponent, a to Windows služby, WPF aplikace, poskytovatele přihlašovacích údajů, SQLite databáze a mobilní aplikace. První čtyři komponenty jsou určeny pro osobní počítač na platformě Windows 7 a poslední z nich je cílená pro mobilní telefon na platformě Android. K výběru platformy Windows 7 mne vedla myšlenka, že aplikace bude sloužit především pro účely mé a mých kolegů a tato platforma je na našich počítačích velmi zastoupená. Mobilní aplikace komunikuje s Windows službou prostřednictvím technologie Bluetooth a v případě, že uživatel opouští své stanoviště, jí také předává požadavky na zamčení počítače. Zamknutí či odemknutí lze docílit také využitím notifikace, která je v mobilní aplikaci zobrazena od doby jejího spuštění a navázání spojení se serverovou částí. V případě, že se mobilní zařízení vzdálí natolik, že se dostane z dosahu signálu Bluetooth počítače, je počítač zamčen okamžitě. K této detekci je využíváno detekčních zpráv, kterými spolu zařízení komunikují. Řešení je také v souladu se stanovenými bezpečnostními kritérii, kterými byla například komunikace s využitím asymetrického a symetrického šifrování. K tomuto účelu byla využita symetrická šifrovací metoda AES a asymetrická šifrovací metoda RSA. Implementované řešení má v mém okolí velký úspěch a je aktivně využíváno, čímž došlo k nezanedbatelné časové úspoře.

Původně bylo zamýšleno bohatší využití senzorů pohybu mobilního telefonu pro posouzení, kdy se uživatel vzdaluje od osobního počítače a jednotlivé výsledky zahrnout v celkovém rozhodnutí dle jejich váhy. Nicméně toto řešení se v průběhu realizace ukázalo jako velmi nespolehlivé a časově náročné, díky čemuž docházelo k opožděným a často rozlišným reakcím. Do budoucna by se dalo řešení rozšířit také na jiné platformy, ať už bereme v potaz platformu počítače nebo platformu mobilního telefonu. Další možnosti rozšíření je spojení aplikace s doménovým řadičem v rámci podnikové domény. Také požadavky na vzdálenou autentizaci mohou být později vyhodnocovány na základě údajů o okamžité poloze uživatele.

Použitá literatura

[1] ŠIMEK, Martin. Bluetooth. Fakulta aplikovaných věd Západočeské univerzity v Plzni [online]. Plzeň: Západočeská univerzita, 2004 [cit. 2018-04-25].
Dostupné z: http://www.kiv.zcu.cz/~simekm/vyuka/pd/zapocty-2004/bluetooth-colakov/

[2] HANUS, Stanislav. Bezdrátové a mobilní komunikace. Brno: Vysoké učení technické, 2001. ISBN 80-214-1833-8.

[3] BEČVÁŘ, Zdeněk, Pavel MACH a Ivan PRAVDA. Mobilní sítě [online]. V Praze: České vysoké učení technické, 2013 [cit. 2018-04-25]. ISBN 978-80-01-05305-8.

[4] VOJTĚCH, L. RFID - technologie pro internet věcí. Access server [online]. Praha, 12.2.2009 [cit. 2018-04-25]. Dostupné z: http://access.fel.cvut.cz/view.php?cisloclanku=2009020001

[5] FINKENZELLER, Klaus. RFID handbook: fundamentals and applications in contactless smart cards and identification. 2nd ed. Hoboken, N.J.: Wiley, c2003. ISBN 04-708-4402-7.

[6] Bluetooth. Bluetooth [online]. [cit. 2018-04-25].
Dostupné z: https://www.bluetooth.com/specifications/profiles-overview

[7] MENEZES, A. J., Paul C. VAN OORSCHOT a Scott A. VANSTONE. Handbook of applied cryptography. Boca Raton: CRC Press, c1997. ISBN 08-493-8523-7.

[8] NIST. ADVANCED ENCRYPTION STANDARD (AES) [online]. 2001 [cit. 2018-04-25]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf

[9] KLÍMA, Vlastimil. Základy moderní kryptologie: Symetrická kryptografie II. [online]. 2005, 35 [cit. 2018-04-25].
Dostupné z: http://crypto-world.info/klima/mffuk/Symetricka_kryptografie_II_2006.pdf

[10] DAEMEN, Joan a Vincent RIJMEN. The design of Rijndael: AES--the Advanced Encryption Standard. New York: Springer, c2002. ISBN 35-404-2580-2.

[11] MÜLLER, Zdeněk. Bezpečné kryptografické algoritmy [online]. Olomouc, 2012 [cit. 2018-04-25]. Dostupné z: https://theses.cz/id/b638au/
Diplomová práce. Univerzita Palackého v Olomouci, Přírodovědecká fakulta. Vedoucí práce RNDr. Miroslav Kolařík, Ph.D.

[12] Android P Developer: Sensors Overview [online]. [cit. 2018-04-25]. Dostupné z: https://developer.android.com/guide/topics/sensors/sensors_overview.html

[13] Gartner [online]. 2018 [cit. 2018-04-25].
Dostupné z: https://www.gartner.com/newsroom/id/3859963

[14] CHEN, Raymond. The Old New Thinkg [online]. 14 January 2016 [cit. 2018-04-25].
Dostupné z: https://blogs.msdn.microsoft.com/oldnewthing/20160114-00/?p=92851

[15] RICHTER, Miloslav. Kódování a dekódování Base64 [online]. 2017 [cit. 2018-04-25]. Dostupné z: http://www.uamt.feec.vutbr.cz/~richter/vyuka/PC2A/cviceni/02_base64.html.cs

[16] BARKER, Elaine a Quynh DANG. Recommendation for Key Management: Application - Specific Key Management Guidance [online]. January 2015, 102 [cit. 2018-04-25].
Dostupné z: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-57pt3r1.pdf

[17] Microsoft: Winlogon and Credential Providers [online]. [cit. 2018-04-25].
Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/bb648647(v=vs.85).aspx

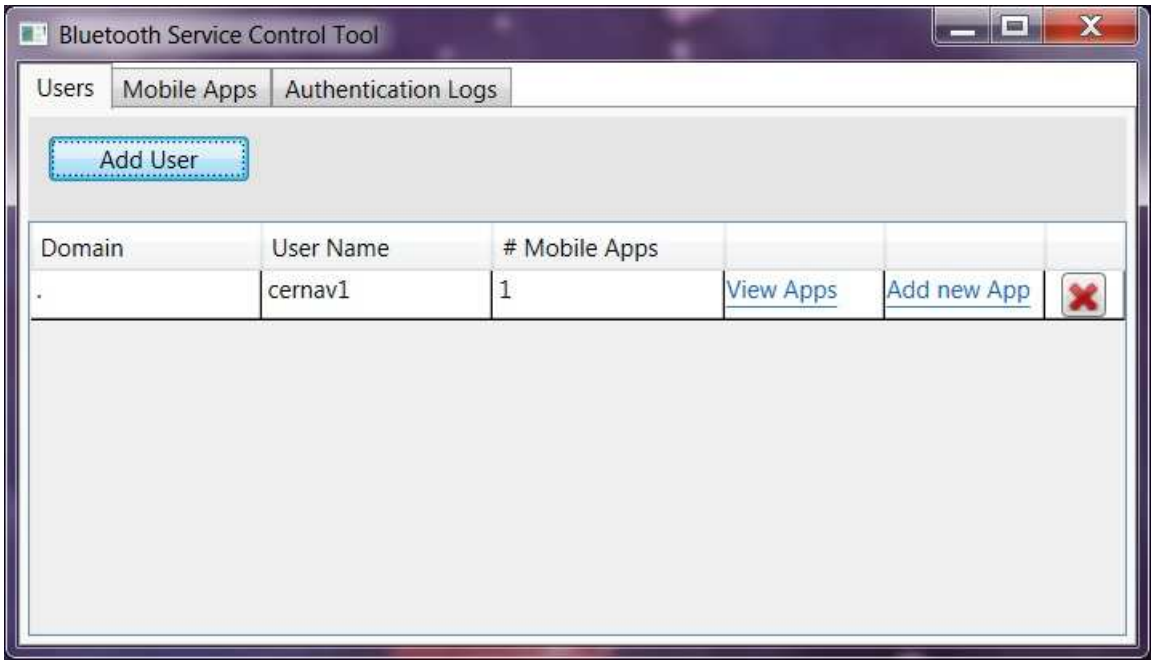
[18] MICROSOFT CORPORATION. Credential Provider Technical Reference [online]. 2016, 35 [cit. 2018-04-25].
Dostupné z: https://www.microsoft.com/en-us/download/confirmation.aspx?id=53556

[19] MICROSOFT CORPORATION. Microsoft RPC Model [online]. [cit. 2018-04-25]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/aa373935(v=vs.85).aspx
[20] MICROSOFT CORPORATION. Named Pipes [online]. [cit. 2018-04-25]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/aa365590(v=vs.85).aspx
[21] COSKUN, Vedat., Kerem. OK a Busra. OZDENIZCI. *Near field communication: from theory to practice*. Hoboken, NJ: Wiley, 2012. ISBN 978-1-119-97109-2.
[22] SABELLA, Robert. *NFC for dummies*. Indianapolis, IN: John Wiley and Sons., 2016. ISBN ISBN:978-1-119-18292-4.
[23] PIPER, F. C. a Sean MURPHY. *Cryptography: a very short introduction*. New York: Oxford University Press, 2002. ISBN 978-0192803153.
[24] CID, Carlos., Sean. MURPHY a Matthew. ROBSHAW. *Algebraic aspects of the Advanced Encryption Standard*. New York: Springer, c2006. ISBN 978-0-387-36842-9.
[25] SWEDBERG, Claire. *RFID Journal: Zebra's Sled Reader Enables UHF RFID Tag Reads Via Smartphone* [online]. 22 April 2015 [cit. 2018-04-27]. Dostupné z: https://www.rfidjournal.com/articles/view?12971

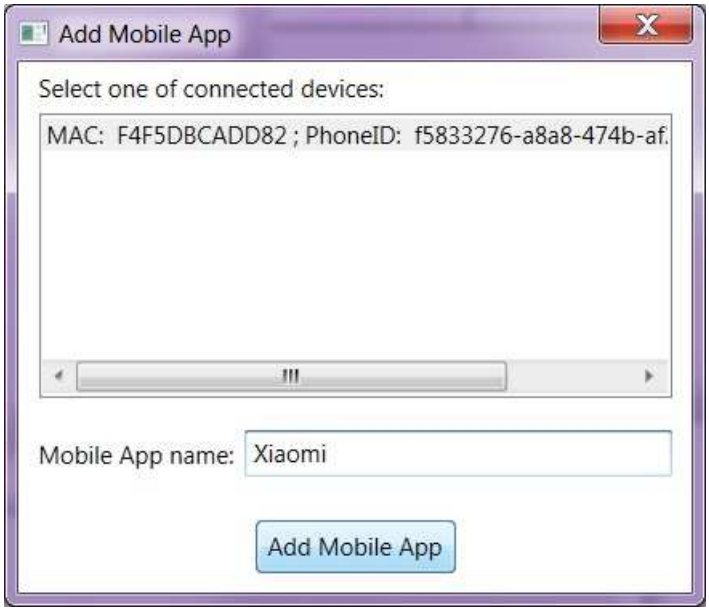
Seznam příloh

Součástí BP je DVD.
Adresářová struktura přiloženého DVD:
BtInstall – složka s instalací celkového řešení obsahující automatizovaný instalátor serverové části
Source – složka obsahující zdrojové kódy celého řešení
Doc – složka obsahující textovou část bakalářské práce ve formátu PDFA

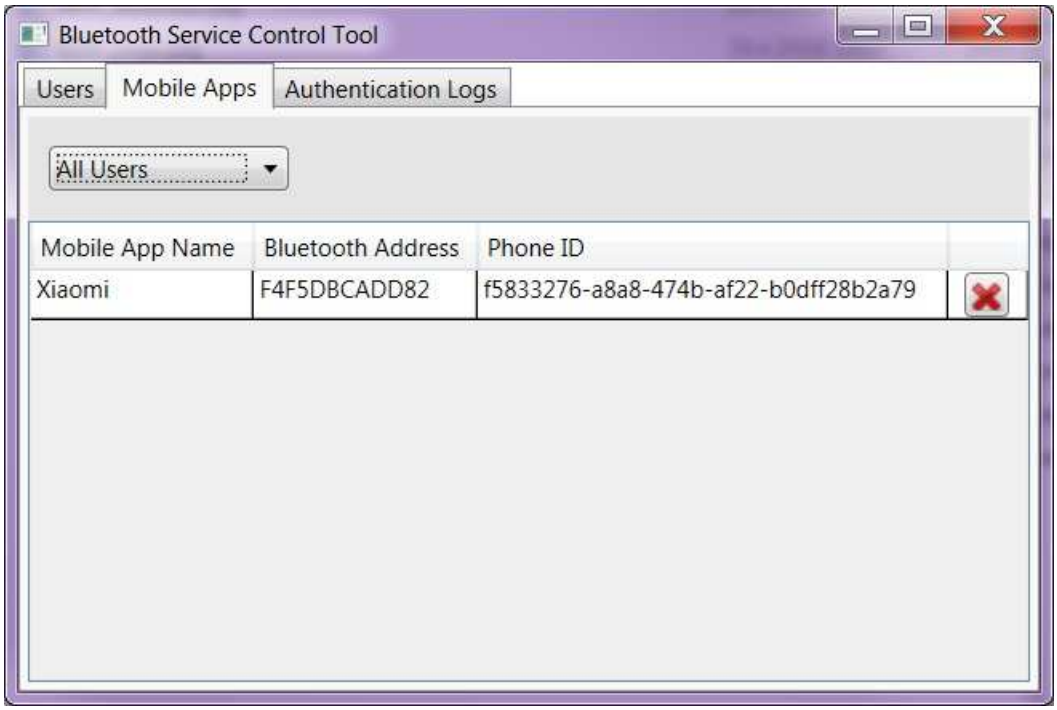
Příloha A: *Ukázky z aplikace*



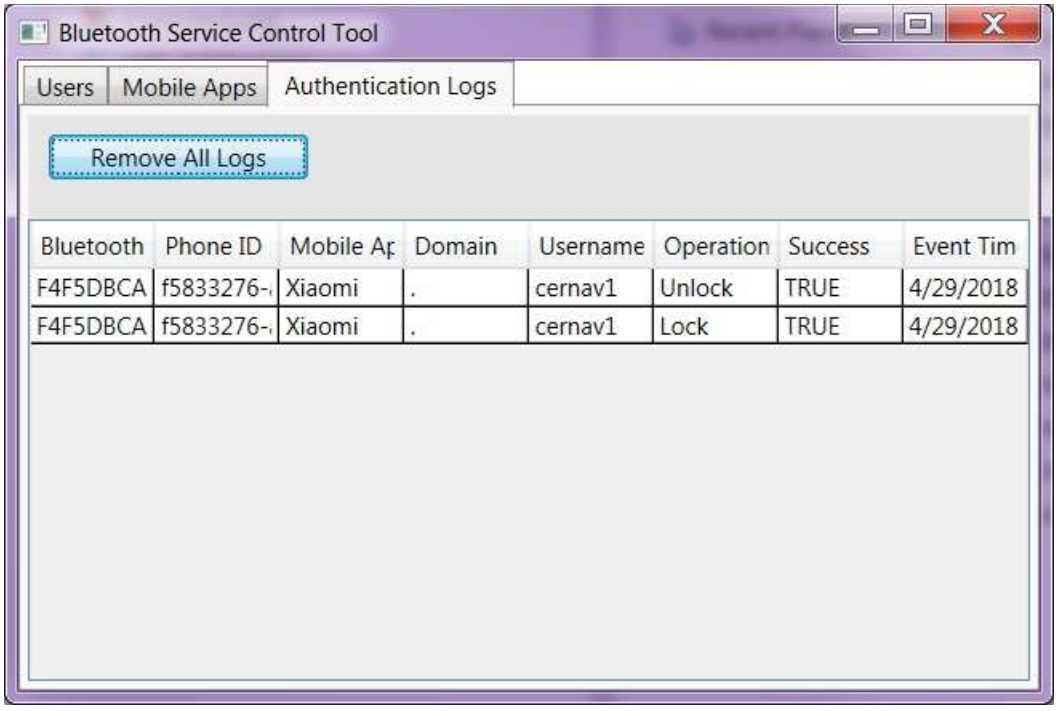
Obrázek 1: *WPF Aplikace – přidání uživatele*



Obrázek 2: WPF *Aplikace – přidání mobilního telefonu*

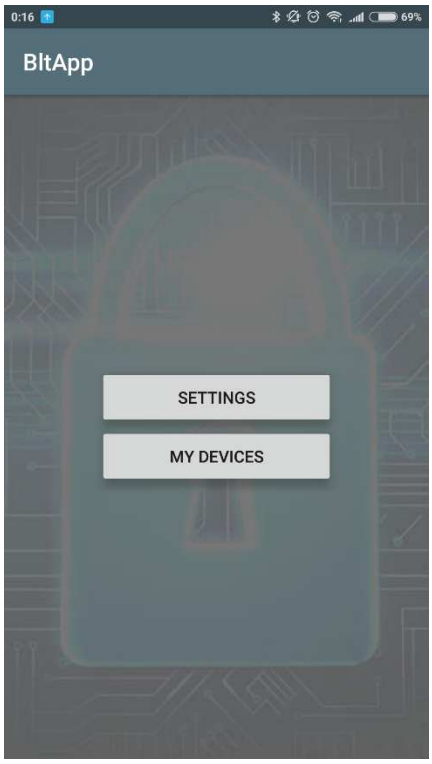


Obrázek 3: *WPF Aplikace – seznam spárovaných mobilních telefonů*

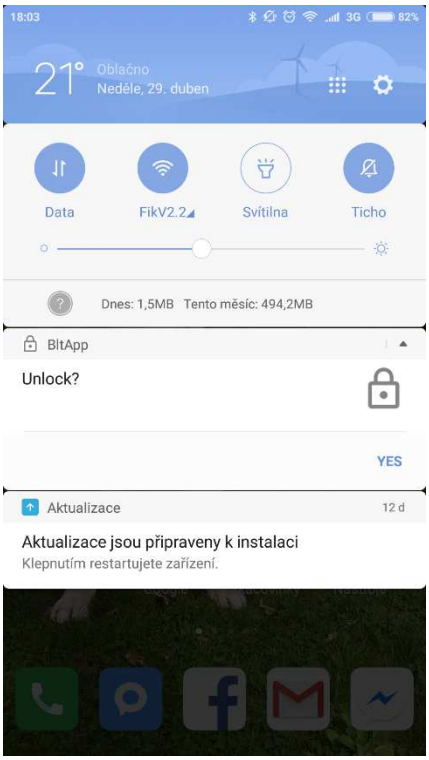


Obrázek 4: *WPF Aplikace – seznam událostí*

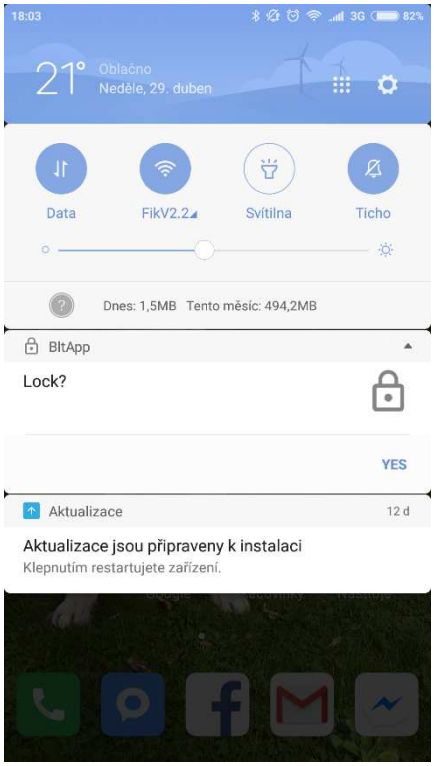
Obrázek 5: *Poskytovatel přihlašovacích údajů*



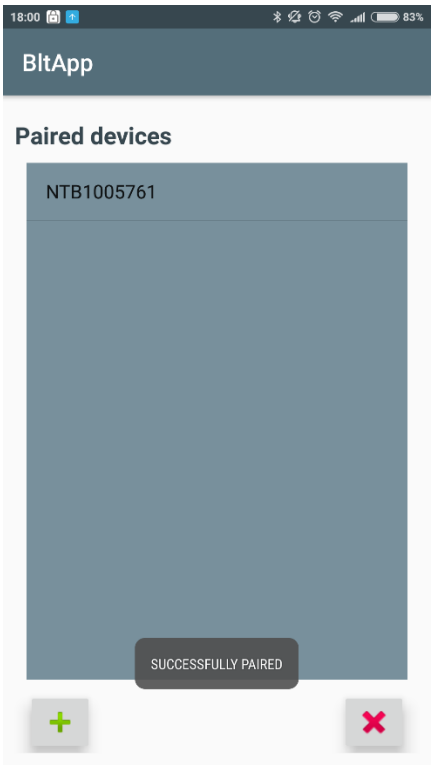
Obrázek 6: *Mobilní aplikace titulní obrazovka*



Obrázek 7: *Mobilní aplikace – Unlock notifikace*

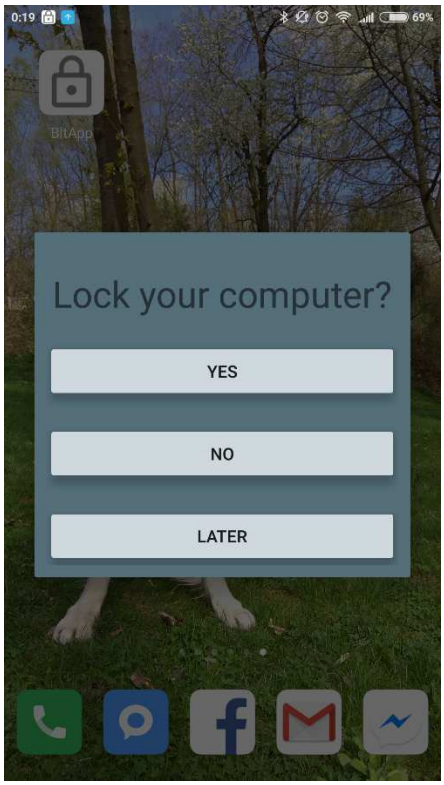


Obrázek 8: *Mobilní aplikace - Lock notifikace*



Obrázek 9: *Mobilní aplikace – Spárovaná zařízení*

Obrázek 10: *Mobilní aplikace – zařízení v okolí*



Obrázek 11: *Mobilní aplikace – Uzamykací dialog*