VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ŠIFROVÁNÍ NAD TEXTOVÝMI ZPRÁVAMI PRO ANDROID

MESSAGE ENCRYPTION OVER TEXT MESSENGERS FOR ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **DAVID BALVÍN**
AUTHOR

VEDOUCÍ PRÁCE — **Doc. Dr. Ing. DUŠAN KOLÁŘ**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19727/2016/xbalvi00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Balvín David**
Obor: Informační technologie
Téma: **Šifrování nad textovými zprávami pro Android**
Message Encryption over Text Messengers for Android
Kategorie: Informační systémy

Pokyny:

1. Prostudujte aplikační možnosti platformy Android, dále prostudujte standardní a populární aplikace pro zasílání zpráv a textových zpráv (SMS).
2. Na základě analýzy navrhněte aplikaci, která umožní přenos šifrovaných zpráv s využitím existujících aplikací/protokolů - aplikace a protokoly konzultujte s vedoucím.
3. Aplikaci z bodu 2 implementujte v prostředí Android - podporované verze konzultujte s vedoucím.
4. Aplikaci důkladně otestujte.
5. Zhodnoťte svůj přínos i celou práci, diskutujte možná rozšíření.

Literatura:

- Ryan Cohen, Tao Wang: GUI Design for Android Apps, Apress, pp. 147, August 2014
- Vybrané stati na https://developer.android.com
- Aplikace Messenger, Xabber, Skype, apod.
- Dále dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- První 2 body zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kolář Dušan, doc. Dr. Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem této práce je navrhnout a implementovat aplikaci, která bude zajišťovat zašifrovanou komunikaci pomocí několika protokolů či s využitím jiných aplikací. Aplikace zvládá automatické šifrování pomocí symetrické šifry při odesílání a dešifrování na požádání při přijmutí zprávy. Též podporuje asymetrickou výměnu klíčů pro následné symetrické šifrování. Výsledná aplikace má uživatelské rozhraní, inspirované defaultní SMS aplikací pro operační systém Android a populárními komunikátory Messenger a Whatsapp.

Abstract

The main purpose of this thesis is to design and implement an application that would handle encrypted communication using several protocols or existing applications. The application is able to handle automatic encryption using a symetric-key algorithm for sending messages and decryption on demand for receiving/reading messages. It also supports asymetric exchange of keys for symetric encryption. A simple user interface is also created for this application, taking ideas from default SMS application for Android and popular Messenger a Whatsapp.

Klíčová slova

Android, aplikace, kryptografie, SMS, GSM, textové komunikátory

Keywords

Android, application, cryptography, SMS, GSM, text communicators

Citace

BALVÍN, David. *Šifrování nad textovými zprávami pro Android.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Dr. Ing. Dušan Kolář

Šifrování nad textovými zprávami pro Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Dr. Ing. Dušana Koláře. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
David Balvín
31. července 2017

Poděkování

Poděkování patří především panu Doc. Dr. Ing. Dušanu Kolářovi za odbornou pomoc a cenné rady při vývoji této práce. Poděkování patří také uživatelům, kteří aplikaci během vývoje otestovali a poskytnuli cennou zpětnou vazbu.

Obsah

Kapitola 1

Úvod

Již od počátku historie lidstva existovala snaha o přenesení informace mezi dvěma stranami, bez přečtení stranou třetí. Ať už se jednalo o milostné psaní, diplomatické depeše či plány vojenského tažení, vždy bylo potřeba skrýt potřebný text tak, aby se jej bez určité znalosti nedalo přečíst.

Jak ale rostla snaha o zatajení zprávy, rostla též snaha o její rozluštění třetí stranou. Obě strany pro svůj úspěch musely (a musí) neustále vymýšlet nové postupy a způsoby, jak zatajit resp. odtajnit zprávu.

V dnešní informační době není přehnané říct, že správná data mají cenu zlata. Vzrůstající efektivita umělých inteligencí a neuronových sítí umožňují automatizovanější zpracování informací na internetu, mezi něž patří i komunikace uživatelů. Na druhé straně však vzniká stále větší potřeba pro bezpečnou komunikaci mezi dvěma subjekty, bez možnosti přečtení třetí stranou.

Při poslechu každodenních zpráv o napadení zařízení, prolomení zabezpečení či vyzrazení obchodních tajemství si člověk snadno uvědomí, jak i každodenní činnosti mohou být předmětem útoku na danou osobu. Že i tak obyčejný způsob komunikace, jako SMS, vyžaduje správnost HW zařízení, SW na tomto zařízení, protokolu a samotné sítě pro přenos. Proto je dobré tato zabezpečení ještě zvyšit pomocí zašifrování samotné zprávy. Pokud tedy útočník obejde zabezpečení GSM sítě a získá SMS zprávu, nezíská z ní žádná data, pouze zašifrovaný text.

Zároveň není možno pominout tendence mnoha státních zřízení o kontrolu a cenzuru komunikace uživatelů a přístupu k informacím. Současný stav například v Čínské lidové republice, kdy bez vhodných SW pomůcek není možné přistupovat k některým politickým otázkám svobodně [26], ukazuje nutnost šifrované komunikace. Takovouto komunikaci není možné snadno cenzurovat, jelikož není možné zjistit, co přesně daná osoba vyhledává.

Z těchto a dalších, níže popsaných důvodů, byla vytvořena aplikace, pracovně pojmenovaná jako Tombstone Talk.

Kapitola 2

Kryptografie a kódování

Tato kapitola popisuje dvě základní práce s textem v této práci, šifrování (kryptografii) a kódování. Tyto dva termíny se často pletou, nicméně znamenají dvě rozdílné operace.

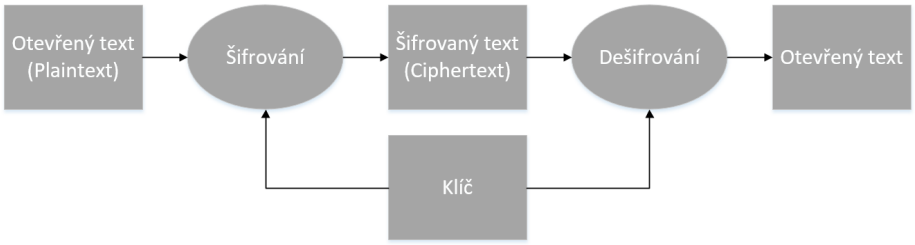
Šifrování informaci převádí z čitelné podoby (tzv. *otevřený text*) do takové reprezentace, kterou nelze snadno přečíst (tzv. *šifrový text*) [32]. Informaci z této podoby, narozdíl od kódování, lze získat pouze se speciální znalostí, například pomocí klíče. Detailněji rozepsáno v sekci 2.1.

Kódování je naopak proces vzájemně jednoznačného přiřazení symbolů jedné množiny symbolům druhé množiny. [28] Detailněji rozepsáno v sekci 2.2.

2.1 Kryptografie

Kryptografie je nauka o metodách utajování smyslu zpráv převodem do podoby, která je čitelná jen se speciální znalostí [32]. Spolu s kryptoanalýzou, vědou luštění zašifrovaných zpráv, se kryptografie pojmenovává jako kryptologie.

Utajování zpráv probíhá pomocí šifrování, což je proces, který převádí čitelnou zprávu, označenou jako otevřený text na její nečitelnou podobu (neboli šifrovaný text), typicky za pomoci klíče. Tento proces je znázorněn na obrázku 2.1, diagram je pro symetrické šifrování, popsané v sekci 2.1.3.



Obrázek 2.1: Obecný diagram kryptografie

Některé způsoby šifrování probíhají bez klíče, např. historicky známá *Caesarova šifra* viz níže sekce 2.1.1. Jednalo se o tzv. *„Security through obscurity“* (česky *Bezpečnost skrze utajení*), kdy utajení probíhá na základě pouhé znalosti šifrovacího algoritmu. Pokud třetí strana zjistí, jakým způsobem probíhá šifrování, může snadno dešifrovat všechy zprávy (například v tomto případě stačí útočníkovi pouze provézt posunutí každého znaku opačným směrem o 3 pozice). Opačný přístup je tzv. *„Secure by design“* (česky *Bezpečné díky návrhu*),

kdy útočníkovi znalost šifrovacího algoritmu nepomůže při jeho útoku a procesu rozšifrování zprávy.

Tyto přístupy jsou jedny z položek tzv. Kerckhoffsových principů, specificky řešeného pravidla *„Security through obscurity doesn't work“* [32].
Kryptografické algoritmy lze rozdělit na:

- **Klasická kryptografie** – Od prvního použití kryptografie do přibližně poloviny 20. století. Jejím úkolem bylo pouze utajení zprávy. Detailněji v následující kapitole 2.1.1.
- **Moderní kryptografie** – Od poloviny 20. století do současnosti. Detailněji v kapitole 2.1.2.

Dále se algoritmy dělí do dvou tříd podle způsobu dešifrování textu:

- **Symetrické šifrování** – K šifrování a dešifrování je využito stejného, tzv. tajného, klíče.
- **Asymetrické šifrování** – K šifrování a dešifrování je využita dvojice klíčů, veřejný a soukromý. Veřejný klíč slouží k šifrování a je možno jej distribuovat po nezabezpečeném médiu (např. internet). Soukromý klíč slouží k dešifrování zprávy a vlastní jej pouze příjemce zprávy. Detailněji v kapitole 2.1.5.

2.1.1 Klasická kryptografie

Klasická kryptografie má jediný úkol, a to utajení zprávy. K většině algoritmů je potřeba pouze tužka a papír k efektivnímu šifrování a dešifrování. Z mnoha historických algoritmů jsou pro práci důležité pouze dva. Jedná se o Caesarovu šifra, tedy šifru bez klíče, a Vernamova šifra, šifru s klíčem. [32].

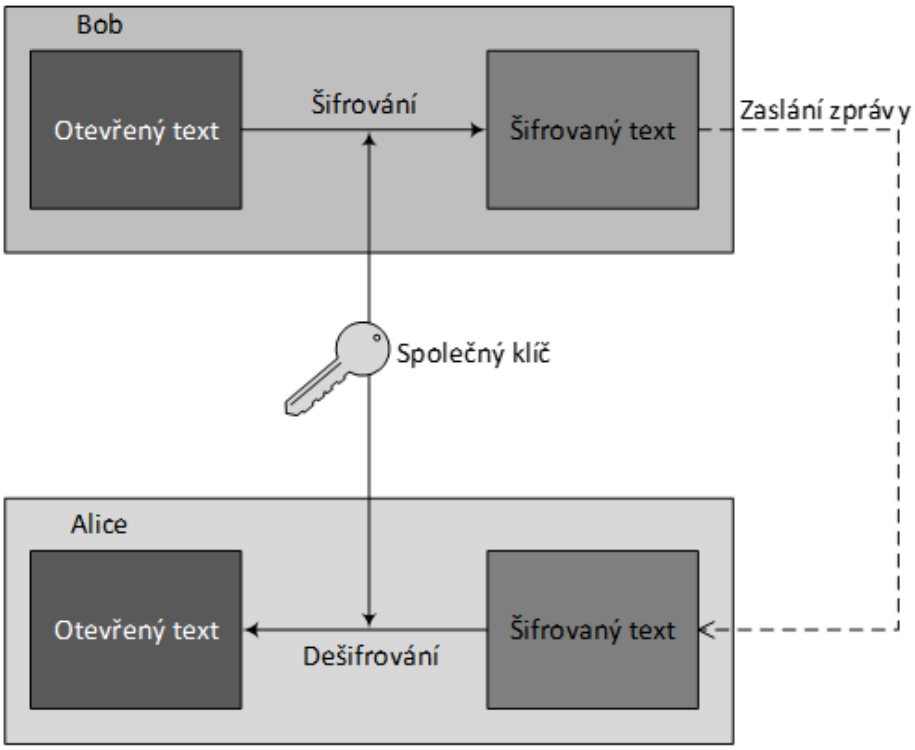
2.1.2 Moderní kryptografie

Moderní kryptografie má oproti klasické kryptografii 4 základní bezpečnostní cíle [36]:

- **Důvěrnost** – Data budou utajena před neoprávněnými uživateli. Zajišťuje se například řízením fyzického přístupu k médiu nebo kryptografickými metodami, tedy podobně jako v klasické kryptografii.
- **Integrita** – Data nebudou úmyslně nebo neúmyslně změněna či poškozena neoprávněným uživatelem. Může se jednat o počítání kontrolních výpočtu či podobné.
- **Autentizace** – Dělí se do dvou kategorií:
  - **Entit** – Ověřuje identitu dané entity (například uživatel, zařízení, program, proces atp.).
  - **Dat** – Ověřuje identitu dat (například její obsah, čas vzniku, původ). Vedlejším efektem je zajištění integrity těchto dat.
- **Nepopiratelnost** – Daný subjekt není schopen popřít svoje předchozí akce, prakticky může třetí strana rozhodnout, zda daný úkon proběhl nebo ne. Může se například jednat o nepopiratelnost odeslání či přijmutí zprávy, díky integritě navíc v určitém specifickém tvaru či textu.

2.1.3 Symetrické šifrování

Algoritmy na základě symetrické šifry využívají k šifrování i dešifrování stejný, tzv. tajný klíč. Pokud tedy chtějí dva modeloví účastníci komunikace, *Alice* (dále pouze *A*) a *Bob* (dále pouze *B*), zahájit šifrovanou komunikaci, musí si nejprve přes zabezpečený kanál či osobně předat/domluvit společný klíč, který bude známý pro obě strany. Následně *B* pomocí tohoto klíče zašifruje zprávu, kterou pošle *A*. Ta ji pomocí stejného klíče rozšifruje a může si přečíst původní text. Pokud bude chtít *A* odpovědět, zašifruje opět pomocí stejného klíče svoji odpověď, odešle ji *B* a ten za stále stejného klíče provede dešifrování a přečte si danou odpověď. Diagram této komunikace je vyobrazen na obrázku 2.2.



Obrázek 2.2: Diagram komunikace s využitím symetrické šifry

Tyto algoritmy se dělí do dvou tříd, na proudové a blokové, dle způsobu zpracování otevřeného textu.

Proudové šifry

Tento druh šifer převádí vstupní text po jednotlivých bitech (či bytech). Vstupní text (datový tok) je kombinován (typicky) pomocí funkce XOR s pseudonáhodným proudem bitů/bytů (anglicky tzv. *„keystream“*) vytvořeným z šifrovacího klíče a šifrovacího algoritmu. Jedná se o snahu o implementaci Vernamovy šifry, která je popsána v kapitole 2.1.1 Proudové šifry jsou obvykle méně náročné na výpočetní výkon a také často umožňují vygenerovat výstup stejně dlouhý jako vstup, nicméně ale jsou méně bezpečné a náchylnější k útokům skrze kryptoanalýzu.

Blokové šifry

Tento druh šifer rozděluje vstupní text na bloky o pevné velikosti (nejčastěji 128 nebo 256 bitů) a následné šifrování je prováděno právě v těchto blocích. Šifrování v blocích se následně liší podle zvoleného způsobu záměny výsledného šifrovaného textu. Některé způsoby vyžadují tzv. inicializační vektor, který je třeba odeslat spolu se zprávou, a který může být nešifrovaný. Slouží de facto jako „semínko“ pro znáhodnění výstupu šifrování. Způsob záměny textu závisí na zvoleném režimu šifrování, z nichž nejznámnější jsou:

- **ECB, Electronic Codebook** – Každý blok je (de)šifrován samostatně, bez ohledu na ostatní bloky. Výhodou je absence nutnosti inicializačního vektoru, nicméně za cenu snížené bezpečnosti. U tohoto režimu identické bloky se pokaždé zašifrují do identických výstupních bloků. Není tedy nutné využít inicializačního vektoru, ale je snažší provézt kryptoanalýtu a rozluštit zprávu.
- **CBC, Cipher Block Chaining** – U každého bloku je při šifrování provedena operace XOR s výstupem šifrování předchozího bloku a nově vzniklý blok je použit jako vstup pro šifrování. U prvního bloku je místo předchozího (neexistujícího) použit inicializační vektor.
- **CFB, Cipher Feedback** – Výstup šifrování je pomocí operace XOR spojen s plaintextem daného bloku a tímto vznikne výstupní šifrovaný text. Tento výstup je použit jako vstup pro další blok. U prvního bloku je místo předchozího (neexistujícího) použit inicializační vektor.
- **OFB, Output Feedback** – U tohoto režimu je použit jako vstup dalšího bloku rovnou výstup šifrování. Taktéž je tento výstup pomocí operace XOR spojen s plaintextem a vznikne výslený šifrovaný text. U prvního bloku je místo předchozího (neexistujícího) použit inicializační vektor.

Blokové šifry též je potřeba vhodným způsobem zarovnat do celých bloků, pokud délka zprávy není násobkem velikosti bloku. Existují různů způsoby a algoritmy, v práci je ale použity dva, PKCS#7 a bitové zarovnání, které jsou popsány níže.

- **Bitové zarovnání** – Tento způsob přidá na konec binární zprávy znak `1` a tolik znaků `0` (i žádný), aby zpráva byla zarovnána do bloku. Pokud již zpráva byla zarovnána, přidá se blok nový ve formátu `100..00`, znaků `0` bude $N-1$, kde N je velikost bloku.
- **PKCS#7** – Tento způsob přidá N bytů o hodnotě N tak, aby byl blok zarovnaný. Pokud je již zpráva zarovnána, přidá se celý nový blok ve stejném formátu.

Mezi algoritmy symetrické šifry se řadí známé AES [5] (viz kapitola 2.1.4), ale také RC4 (využíván např. u protokolu HTTPS) či DES [25]. Algoritmus DES je dnes již historický, ale stále používaný, byť kvůli některým jeho vlastnostem (jako například kratší klíč) je nebezpečný a dnes již prolomený [24].

Výhody a nevýhody symetrických šifer

Kromě jednodušší implementace je hlavní výhodou symetrického šifrování především jeho rychlost.

Naopak hlavní nevýhodou je nutnost sdílení klíče. Vyjma problému s bezpečnou výměnou tohoto klíče pak nastává možnost jednoduššího odhalení klíče v případě slabší ochrany

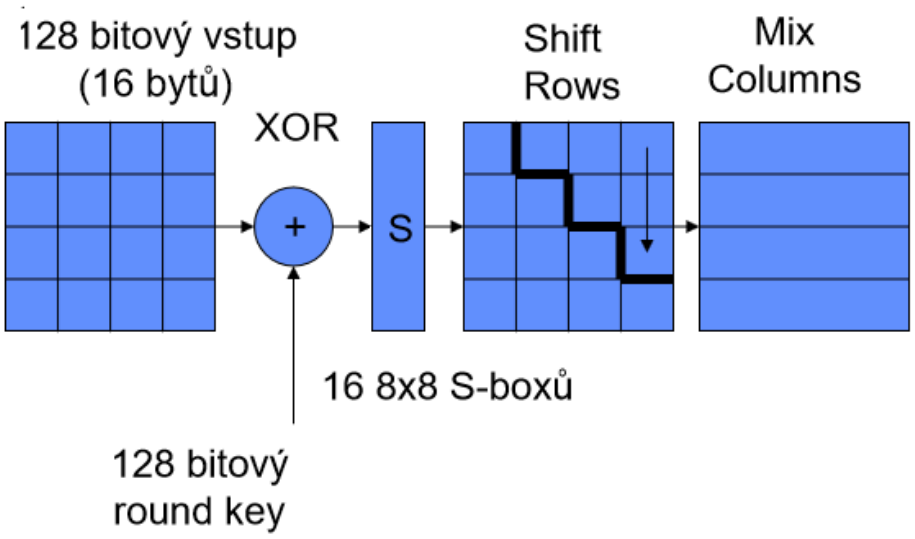
proti útočníkům na zařízení jednoho z účastníků komunikace (A či B), v tomto případě je možno komunikaci poslouchat obousměrně. Částečně se tomuto dá zabránit použitím dvou samostatných klíčů pro oba kanály (směr $A \rightarrow B$ bude (de)šifrován pomocí jiného klíče, než $B \rightarrow A$), nicméně v případě možnosti zjištění jednoho klíče je vysoká pravděpodobnost zjištění i toho druhého.

2.1.4 AES

V práci je využito algoritmu AES, který byl vytvořen jako nástupce zastaralého algoritmu DES. Jedná se o blokovou šifru, s bloky o pevné velikosti 128 bitů. Klíče mohou být 128, 192 nebo 256 bitů dlouhé. Její každodenní případ použití je v rámci zabezpečení Wi-Fi sítě, kde slouží k autentizaci uživatelů do sítě a šifrování dat během připojení. V této práci slouží k šifrování textu.

Samotné šifrování začíná operací XOR podklíče s šifrovaným blokem. Následně již probíhají operace, které mohou probíhat v několika kolech, dle velikosti klíče (například pro 128 bitů je to 10 kol). Tyto operace jsou 4 a to následující:

1. **Byte Sub** – Každý byte bloku je nahrazen jiným bytem podle S-boxu.
2. **Shift Row** – Byty jsou uspořádány do matice a posunuty.
3. **Mix Column** – Násobení matic
4. **Add round key** – Operace XOR subklíče.



Obrázek 2.3: Diagram principu šifrování algoritmu AES [32]

2.1.5 Asymetrické šifrování

Algoritmy na základě asymetrické šifry využívají k šifrování odlišný klíč než k dešifrování. Tyto klíče se nazývají soukromý klíč a veřejný klíč. Soukromý klíč vlastní pouze jedna strana a nesmí se dostat do rukou ostatních. Naopak veřejný klíč je možno distribuovat

po nezabezpečeném kanálu. Tento druh algoritmů je často založen na řešení některého matematického problému, například problém známý jako „Knapsack“[1].

Mějme dva modelové účastníky A a B, kteří spolu komunikují, a C, který se snaží naslouchat. Asymetrická šifra se využívá k dvěma základním principům:

- **Elektronický podpis** – A zašifruje text pomocí svého soukromého klíče a odešle jej B. Ten jej pomocí veřejného klíče A dešifruje a s jistotou může určit, že text skutečně pochází od A, jelikož A jako jediný mohl text zašifrovat. V tomto případě ale může C zprávu taktéž dešifrovat, jelikož též může vlastnit veřejný klíč A. Tento princip tedy neslouží k utajení zprávy.

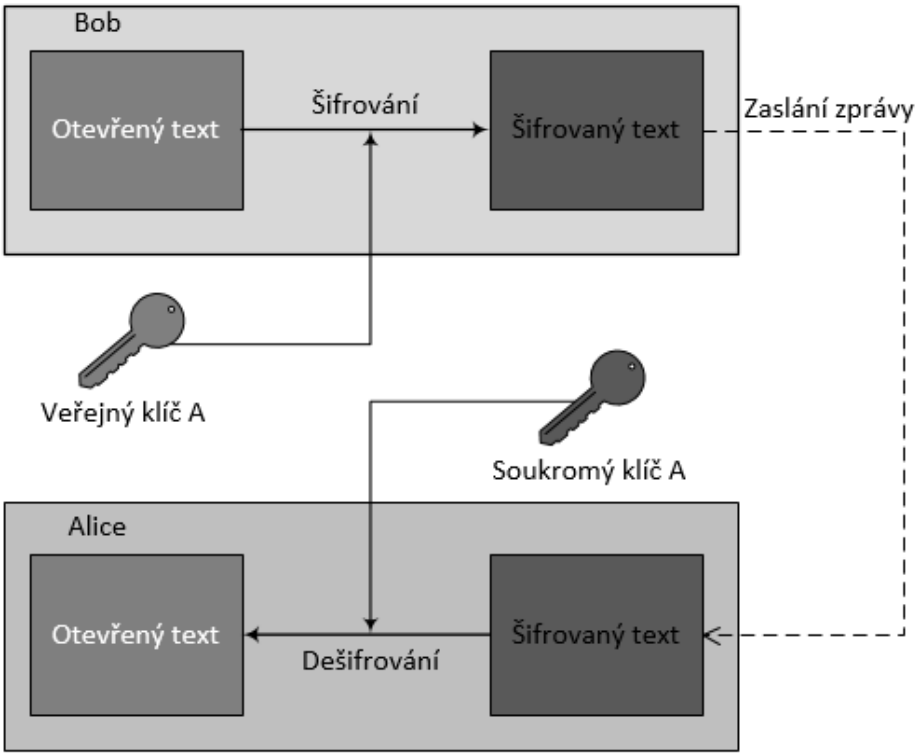
- **Utajení zprávy** – A zašifruje text pomocí veřejného klíče B a odešle jej. B tento text dešifruje pomocí svého soukromého klíče a jako jediný si může přečíst původní zprávu. V tomto případě ale neví, jestli zprávu skutečně odeslal A, jelikož C taktéž vlastní jeho veřejný klíč a nemůže tedy pouze s šifrou určit zdroj zprávy.

Z rozdělení je vidět, že ke skutečně bezpečné komunikaci, tedy že známe odesílatele a zprávu si může přečíst pouze příjemce, je nutno využít obou způsobů. V praxi se používá různých režimů šifrování, například populární *GCM*[2], který konkrétně kombinuje oba přístupy v jedné operaci. Tento režim je navíc mnoha procesory podporován na úrovni HW, což urychluje šifrování.

Dále taktéž platí, že v případě získání soukromého klíče B je možno odposlouchávat pouze směr $A \to B$, směr $B \to A$ zůstává zabezpečen. Diagram této komunikace je naznačen na obrázku 2.4.

[1] Problém Knapsack, česky známý jako *problém batohu* se zakládá na určení váhy těžítek v batohu. Známe váhy jednotlivých těžítek, některé z nich vložíme do batohu, který je neprůhledný. Následně se z celkové váhy batohu pokusíme určit, která závaží v batohu jsou.

[2] http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf



Obrázek 2.4: Diagram komunikace s využitím asymetrické šifry

Nejznámějším a nejpoužívanějším algoritmem asymetrické šifry je RSA [40], dalšími používanými jsou například Diffie-Helman [27] (asymetrická výměna klíče pro symetrické šifrování) či Merkle-Hellman Knapsack [38].

Výhody a nevýhody asymetrických šifer

Hlavní výhodou algoritmu je absence nutnosti sdílet jeden stejný klíč pro obě osoby. Klíč pro zašifrování zprávy lze též přenášet po nezabezpečené lince, což u symetrické šifry nelze (pokud třetí strana zachytí klíče během předávání, zprávu může číst také). S tím souvisí i případná snazší výměna dvojice šifrovacích klíčů v případě podezření/odhalení odposlechu.

Nevýhodou je především vyšší režie oproti symetrickým šifrám, z důvodu složitějších počtů. V případě asymetrických šifer totiž dochází k použití matematického problému, které útočníkovi znemožní rychlejší řešení, než-li pomocí tzv. *„brute-force“* útoku (útočník zde zkouší všechny možné klíče, což zabere mnoho času). Například u algoritmu RSA je to faktorizace na velká a dlouhá prvočísla, což je operace výpočetně složitá[3]. Tato nevýhoda se často neguje využitím asymetrické šifry pro výměnu klíče, který bude použit ke komunikaci se symetrickým šifrováním. Další možností je využití tzv. hybridní kryptografie[4]

V tomto případě oba dva skončí se stejnou hodnotou K, kterou lze použít jako klíč k symetrické šifře. Algoritmus se zakládá na faktu, že je velice složité z hodnot A a B vypočítat příslušné koeficienty a a b použité při jejich výpočtu.

[3] https://www.algoritmy.net/article/111/Faktorizace

[4] Hybridní kryptografie se zakladá na kombinaci symetrické a asymetrické šifry. Symetrická šifra se využije na zašifrování dat a následně asymetrická na zašifrování klíče symetrické šifry. tyto dvě informace se spojí a odešlou se.

Algoritmus ale není odolný vůči MITM[5] útoku, jelikož nijak neověřuje původ dat. Proto je potřeba tento algoritmus doplnit o další mechanismy.

2.1.6 Hashovací funkce

Hashovací funkce je matematická funkce, převádějící vstupní data o různé délce do výstupu o pevné délce. Jedná se o tzv. jednocestnou funkci (anglicky „one-time pad“), kdy z výstupu již zpětně nelze určit vstup. I změna jednoho bitu zprávy má drastický účinek na výstup. Pokud $F(x) = y$ značí, že y je výstup hashovací funkce F pro vstup x, pak platí:

- Je výpočetně nezvládnutelné pro dané y nalézt x takové, že platí $F(x) = y$. (Nelze zpětně z výstupu nalézt vstup)
- Je výpočetně nezvládnutelné pro dané x nalézt $x' \neq x$ takové, že platí $F(x') = F(x)$. (Nelze nalézt pro konkrétní vstup jiný vstup, který by měl stejný výstup)
- Je výpočetně nezvládnutelné nalézt x a x' takové, že $x \neq x'$ a $F(x') = F(x)$. (Nelze nalézt dvojici vstupů, která má stejné výstupy)

Hashovací funkce se využívá především v následujících případech:

- **Autentizace** – Při přihlašování se neodesílá přes nezabezpečený kanál heslo v tzv. „plaintext“ podobě (čistý text), ale jejich otisk v podobě výstupu hashovací funkce. Pokročilé možnosti nabízejí různé protokoly pracující s tzv. solí[6], která změní výstup hashovací funkce. Této funkcionality se využívá například v protokolu CHAP [42].
- **Bezpečné uložení přihlašovacích údajů** – Při ukládání přihlašovacích údajů se neukládají v plaintext podobě, ale uloží se jejich otisk pomocí hashovací funkce. Následné přihlašování poté probíhá pomocí porovnání obdrženého hashe od klienta a uloženého hashe. V případě průniku do databáze a odcizení údajů mají útočníci ztíženou práci ve zjišťování hesel.
- **Podepisování souborů** – Poskytovatel webového obsahu, v tomto případě souborů, před zveřejněním spočítá otisk daného souboru a vystaví jej veřejně spolu se souborem. Uživatel, který si takovýto soubor stáhne, si může tento otisk také vypočítat a porovnat jej s veřejně uvedeným otiskem souboru. Pokud se tyto hodnoty rovnají, může si být uživatel jist, že během přenosu nedošlo ke změně souboru.

Nejznámějšími hashovacími funkcemi jsou MD5 [39] (dnes již spíše historická), SHA-1 [30] a SHA-2 [29]. Taktéž používaná hashovací funkce je PBKDF2, jejíž přínos pro práci a samotný popis je rozebrán v kapitole níže, 2.1.6

[5] MITM je zkratka pro typ útoku „Man In The Middle“. Zpočívá ve způsobu zachycení komunikace uživatele se serverem a přesměrování obou směrů přes zařízení útočníka. Oběť se o útoku nemusí nijak dozvědět.

[6] Termín sůl se používá pro několik náhodných bitů, vhodně zkombinovaných se vstupem při hashování. Hash je poté pokaždé jiný. Takto je zamezeno tzv. *slovníkovému* útoku a použití tzv. *duhové* tabulky, která používá předvypočítané hashe pro snížení doby útoku.

PBKDF2

PBKDF2 [34], zkratka pro *Password Based Key Derivation Function 2* (česky volně *Funkce generující klíč na základě hesla 2*), je hashovací funkce, která převádí heslo ve formě textu na klíč ve formě otisku. Je speciálně vytvořena pro požadavky kryptografie, resp. zabezpečení a to tak, že její spočtění trvá mnohem déle, než klasické hashovací funkce jako SHA-1. Toho je dočiněno, že se výpočet otisku provádí na mnohonásobně více kol, specificky minimálně 1000 kol, oproti například SHA-1, která se počítá na 80 kol. Takováto zábrana ještě více znemožňuje útoky typu *„brute force“*, jelikož výpočet otisku pomocí těchto funkcí je velice pomalý.
Těchto vlastností je pro zvýšenou bezpečnost využito i v aplikaci pro bezpečnější práci s klíčem pro šifrování a dešifrování.

2.1.7 Steganografie

Mezi způsoby utajování zpráv patří také steganografie, což je věda zabývající se ukrýváním zprávy jako takové [32], například pomocí neviditelného inkoustu či v moderní době například do souborů s hudbou či obrázky. Jedná se též o snahu skrýt potenciálnímu odposlechu, že šifrovaná komunikace vůbec probíhá. Pro odposlouchávající stranu se může jednat o zdánlivě nevinné poslání fotky z dovolené, ale v tomto obrázku je ukryta tajná zpráva.

Například pokud k zobrazení obrázku využíváme pro každý pixel 24bitové barevné rozložení RGB, tedy pro každou ze tří základních barev 8 bitů, je možno u každé složky odebrat 2 nejméně důležité bity a tyto využít k vložení naší zprávy či obrázku. U zdrojového obrázku sice dojde ke snížení kvality a počtu barev, nicméně pro lidské oko nepoznatelné.

Jelikož se opět jedná o přístup tzv. *„Security through Obscurity“*, který navíc není pro tuto práci příliš využitelný (používá se především pro média ve formě vodoznaků, anglicky *watermarking*), nebude steganografie využito.

2.2 Kódování

Kódování je proces vzájemně jednoznačného přiřazení symbolů jedné množiny symbolům druhé množiny. Jedná se například o převod znaku abecedy do jeho binární reprezentace pomocí tzv. *kódu*. Termín kód může znamenat buď předpis pro vzájemně jednoznačné přiřazení mezi kódovanými symboly a kódovými kombinacemi a nebo označení vlastní kódové kombinace. Pro jednotky provádějící kódování a dekódování se používají termíny *kodér*, resp. *dekodér*.
Kódování se dělí do dvou tříd dle způsobu zpracování vstupních dat:

- **Blokové** – Vstupní data (nebo také *vstupní tok*) se rozdělí na bloky nastavitelné velikosti a kódují se jednotlivé bloky odděleně.
- **Proudové** – Vstupní data jsou zpracována způsobem, že kodér zakóduje postupně data po jednotlivých bitech (či bytech) a takto pokračuje až do konce souboru.

Důležitým faktorem během kódování je tzv. *kompresní poměr*, který udává poměr výstupního toku a vstupního toku. Je tedy možné jej spočítat pomocí vzorce:

$$x = \frac{\text{velikost výstupního toku}}{\text{velikost vstupního toku}}$$

Pro hodnoty $x \in (0, 1)$ se jedná o tzv. *kompresi*, pro hodnoty $x \in (1, \infty)$ se jedná o tzv. *expanzi* (výsledný kód je delší, než vstupní, například u níže popsaného (kapitola 2.2.1) Base64).
Samotné metody kódování se mohou rozdělit do tří druhů na:

- **Statistické** – Znaky se kódují podle pravděpodobnosti výskytu kódem s proměnnou délkou kódového slova.
- **Slovníkové** – Vytvářejí se slovníky omezené velikosti z předcházejících slov nebo slabik
- **Kontextové** – Z předchozích a sousedních hodnot lze odhadnout skutečnou hodnotu s malou chybou a tato chyba se kóduje.

2.2.1 Base64

Base64 [7] je kódování, primárně určené pro použití v e-mailových zprávách. Jeho princip se zakládá na překódování 3 binárních oktetů na 4 ASCII znaky. Jedná se o znaky malé (`a-z` a velké (`A-Z`) anglické abecedy, číslic (`0-9`), znak `+` a znak `/`. Zarovnání je do 3 oktetů, v případě, že původní zpráva není zarovnaná, použije se znak `=` jednou či dvakrát pro případné zarovnání. Toto kódování průměrně rozšíří zprávu o 33%. Je to cena za převedení ninárního vstupu do tisknutelných znaků.

2.2.2 Huffmanovo kódování

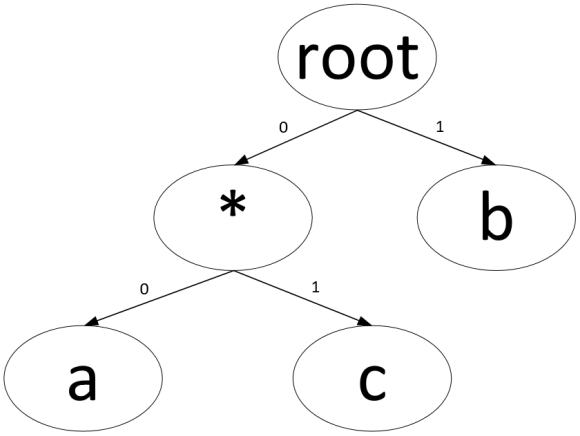
Huffmanovo kódování je metoda, která vytváří kód, kdy délka kódu pro jednotlivé znaky je určena jejich pravděpodobností. Tento kód se určí za použití Huffmanova stromu. Tento strom obsahuje symboly jakožto listy a je tvořen zdola nahoru. Tento strom má vlastnost, že váhový součet listů, kde váhy jsou vzdálenosti listů od kořene, se rovná součtu vnitřních uzlů [28]. Pseudokód vytvoření takového stromu by mohl být zapsán jako (pro statické kódování):

```
vytvoř seznam abecedních symbolů (v sestupném pořadí dle
 pravděpodobností);
while délka seznamu není 1 do
    vyber dva prvky seznamu s nejnižší pravděpodobností;
    prvky sečti do vrcholu místního sbíhání větví;
    prvky vyškrtni ze seznamu;
    přidej do seznamu symbol reprezentující vyškrtlé prvky;
end
```

Algorithm 1: Pseudokód vytvoření Huffmanova stromu

Následným systematickým procházením stromu se zjistí kód pro všechny symboly. Například pro binární strom (vyobrazený na obrázku 2.5) při průchodu stromem by vybrání levé větve znamenal zapsání znaku `0`, vybrání pravé větve znaku `1`. Použití binárního stromu (resp. binárních hodnot) však není nutné, algoritmus je možné použít i na tři a více hodnotovou logiku (například levý `0`, prostřední `1` a pravý `2`).
Huffmanovo kódování je statistické (kódování znaku záleží na pravděpodobnosti jeho výskytu). Výsledný kód je prefixový, znamenající že žádné kódové slovo není prefixem jiného (symboly jsou ve stromu pouze jako listy, nelistové větve jsou pouze zástupné symboly)

Dále je možno též Huffmanovo kódování rozdělit na tři druhy, dle způsobu a času vytvoření stromu, na: statické, adaptivní a semi-adaptivní.



Obrázek 2.5: Příklad binárního stromu použitelného pro Huffmanovo kódování

Statické Huffmanovo kódování

Tento způsob kódování počítá se znalostí četnosti symbolů předem. Strom je v tomto případě vytvořen před kódováním samostatně a stejný strom je proužit pro různá data. Z této vlastnosti vyplývá, že dekodér musí mít k těmto informacím (resp. stromu) přístup, aby mohl provést operaci dekódování. Řešením je trvale uložený strom (resp. jeho reprezentace) v samostatném souboru, či ke každému zakódovanému souboru je k výstupu přiložen i samostatný strom.

Semi-adaptivní (hybridní) Huffmanovo kódování

V tomto případě probíhá kódování ve dvou fázích. V první fázi se prochází data a vytváří se tabulka četnosti (resp pravděpodobnosti) jednotlivých symbolů. Z této tabulky se vytvoří strom Huffmanova kódování, který se použije v druhé fázi stejně jako u statického k zakódování samotných dat.

Adaptivní Huffmanovo kódování

Zde začíná kódování a dekódování z počátku s prázdným stromem. Teprve v průběhu (de)kódování se podle vstupu vytváří samotný strom, který se často během procesu mění na základě postupného přijímání dalších symbolů. Každý znak se při prvním výskytu umístí na výstup v nekomprimované podobě a přidá se symbol do stromu. Pokud se již nejedná o Huffmanův strom, je strom přetvořen a další výstupní kód již může být jiný, i u předchozích vstupů. Dekodér pracuje podobně, pouze v opačném směru. Též je potřeba mu naznačit, jaký symbol je vložen v nekomprimované podoby, většinou pomocí tzv. *„escape znaku“*. Tento znak se během dekomprese sám může měnit, podobně jako ostatní znaky.

Kapitola 3

Technologie pro odesílání textových zpráv

Tato kapitola se věnuje rozboru technologií použitých pro přenos textových zpráv a jejich požadavcích. Z těchto požadavků vychází i návrh pro výslednou aplikaci Tombstone Talk. Též zde jsou prozkoumány možnosti již existujících aplikací pro odesílání textových zpráv a využity k návrhu aplikace Tombstone Talk.

3.1 SMS

SMS, zkratka pro *Short Message Service* (česky *Služba krátkých textových zpráv*), je celosvětově dostupná služba pro posílání krátkých zpráv, většinou pomocí mobilních telefonů. Označení SMS se také často používá pro samotnou zprávu poslanou skrz tuto službu.

Služba SMS využívá nejčastěji ke svému provozu sít GSM, viz kapitola 3.2. Samotná SMS zpráva se dle standardu [19] kóduje do maximálně 1120 bitů na jednu SMS. Standard umožňuje na kompatibilním zařízení využívat tzv. „řetězení zpráv“, pomocí odebrání určité kapacity zprávy pro definování hlavičky, viz kapitola 3.2.2. Dle použitého kódování se může jednat o následující maximální velikost textu pro jednu SMS zprávu (Druhé číslo udává hodnotu při zřetězené zprávě):

- **160 / 153 znaků** – 7bitové kódování ve znakové sadě **GSM 03.38**
- **70 / 67 znaků** – 16bitové kódování ve znakové sadě **UCS-2**

7bitové kódování je implicitní, 16bitového se využije pouze v případě, že zpráva obsahuje alespoň jeden znak mimo 7bitové kódování (například znaky tzv. *emoji*, definované v Unicode, jsou psány pomocí UCS-2). Standard ještě popisuje velikost 140 znaků pro kódovaná 8bitová data, nicméně tato možnost se v práci neuvažuje. Některá zařízení možnost řetězení SMS nevyužívají či nepodporují a zprávy SMS delší než 1120 bitů odesílají jako zprávu MMS (zkratka pro *Multimedia Message Service*).

Služba je populární především pro její dostupnost. Téměř každý dnes vlastní telefon schopný přijímat a odesílat SMS. Dále je to také absence jakékoliv registrace, vyjma potřeby vlastnit SIM kartu (viz kapitola 3.2.3) od operátora a paušál či kredit pro odeslání. V neposlední řadě je důvodem popularity též minimální nutnost kvality signálu, kdy i při velmi špatném signálu je možno odeslat SMS. Naopak u komunikátorů postavených na technologii IP často dochází k nemožnosti odeslat zprávu při slabém signálu.

Maximální počet znaků na zprávu značně omezuje možnosti šifrování, a jelikož při použití mnohých algoritmů dojde ke zvětšení velikosti oproti původnímu textu, může nastat situace, že je třeba odeslat více SMS zpráv, než-li bylo původně nutno. Je tedy třeba zvolit vhodný šifrovací algoritmus a taktéž použít kompresní algoritmus, analýza vhodných je v sekci 5.4.6.

3.2 GSM

GSM[1], zkratka pro *Global System for Mobile Communications* (česky *Globální Systém pro Mobilní komunikaci*), je označení pro nejpoužívanější standard pro mobilní komunikaci a zároveň označení pro síť umožňující takovouto komunikaci. Byl navrhnut nejen pro umožnění telefonování, ale i používání další služeb, jako zprávy SMS, datové přenosy, hlasová schránka, přesměrování hovorů a jiné. Jednalo se také, na rozdíl od předchozích systémů, o digitální přenos, který umožňuje šifrování, lepší využití jednotlivých buněk, menší spotřebu při hovoru a snadný přenos čísla na jiné zařízení pomocí SIM karty (viz kapitola 3.2.3). GSM síť je buňková, což znamená, že zařízení se do sítě připojuje pomocí nejbližší buňky, skrz kterou pak dále komunikuje se zbytkem sítě. Označuje se jako tzv. *„síť druhé generace“* (2G).

3.2.1 GSM 03.38

GSM 03.38 [18] je znaková sada využívaná v SMS zprávách, definovaná pracovní skupinou GSM. Od dob, kdy se o GSM stará projekt 3GPP, se znaková sada označuje jako 3GPP 23.038. Tabulka znaků je vyobrazena v příloze A.1. Tato tabulka je rozdělena do dvou částí. V první části se znaky píši jejich standardním zakódováním. Druhá část vyžaduje zapsání znaku `ESC` a následné zapsání znaku dle tabulky. Tato druhá část tabulky se využívá k zapsání některých národních znaků, například pro české znaky s diakritikou, např. ř, š, ž apod. Specifikace nevynucuje podporu pro tuto druhou část tabulky. V tomto případě se znak `ESC` při přijetí interpretuje jako mezera a následuje znak, který by pocházel z první části tabulky, či zakóduje do USC-2 během odesílání zprávy. Při použití rozšířené tabulky je potřeba definovat použité kódování rozšířené tabulky v hlavičce SMS zprávy.

3.2.2 GSM 03.40

GSM 03.40 [19] je standard popisující formát hlavičky SMS zprávy, sloužící pro nastavení příznaků zprávy a kontrolu přenosu zprávy. Standard popisuje nejen vynucené části standardu, ale i nevynucené části, kde není zaručena 100% funkcionalita a zachování při přenosu sítí, například výše uvedené určení kódování rozšířené tabulky GSM 03.38. Od převzetí správy GSM projektem 3GPP se standard označuje jako 3GPP TS 23.040, nicméně často se stále používá toto starší označení.

3.2.3 SIM

SIM [20], zkratka pro *Subscriber Identity Module* (česky volně *Účastnický identifikační modul*), je speciální čip, známý jako SIM karta, která umožňuje autentifikaci uživatele do sítě GSM. Jako standard jej popisuje 3GPP TS 51.011 (dříve GSM 11.11). Pomocí této karty lze

[1] http://www.3gpp.org/

snadno měnit zařízení při zachování svého telefonního čísla. Některé telefony jsou prodávány jako tzv. *„SIM locked“*, což znamená že je možné v nich využít pouze SIM od jednoho operátora nebo skupiny operátorů. Jejich odblokování často není operátorem poskytováno, či je zpoplatněno.

V současné době se začíná zavádět nová technologie, tzv. *„eSIM“*. Jedná se o vestavěný čip přímo do zařízení, který na rozdíl od klasické SIM karty nelze vyjmout, zato jej ale lze přepisovat. Proces přenosu čísla mezi zařízeními A a B by tedy znamenal smazání identity na zařízení A a zapsání této identity na zařízení B.

3.2.4 Zabezpečení GSM sítě

Původní specifikace GSM počítá pouze s ověřením uživatele u buňky, nikoliv naopak. Zároveň komunikace mezi těmito subjekty může (ale nemusí) být šifrována, nejčastěji pomocí šifrovacích algoritmů A5/1 nebo A5/2 [32]. Oba tyto algoritmy jsou již ale dnes brány za prolomené, a tedy nebezpečné.[2]

Jelikož je ale i samotné šifrování výše uvedenými algoritmy specifikací GSM nevynuceno, ještě více se snižuje celkové zabezpečení sítě. Taktéž se jedná i o možnost využití falešné GSM buňky. Technicky je to sice náročná možnost útoku, nicméně v případě dostatečné blízkosti k oběti může útočník využít MITM útoku a odposlouchávat komunikaci. Vyřešení tohoto problému přináší rozšíření UMTS[3] konkrétně USIM, která kromě jiného vynucuje autentifikaci nejenom směrem uživatel → buňka, ale i buňka → uživatel.

Nevynucené zabezpečení a bezpečnostní úniky jsou důležitou motivací pro vypracování tohoto projektu. Je zřejmé, že přenos je jednoduché zachytit a nepozorovaně odposlechnout. Tomuto zabrání šifrování zpráv.

3.3 Komunikátory a protokoly ke komunikaci

V této kapitole jsou rozebírány další možnosti přenosu zašifrovaných zpráv, za použití jiných než GSM technologií. Jedná se především o komunikátory pracující na protokolu IP[4], které je možno využít jako prostředníka pro přenos zprávy, bez znalosti daných komunikátorů o obsahu dané zprávy. Tyto komunikátory se dělí do několika druhů, především dle otevřenosti jejich API[5].

3.3.1 Otevřené protokoly

Tato podsekce je věnována otevřeným protokolům. Tímto termínem jsou v této práci označovány takové protokoly, které umožňují vytvoření vlastního klienta bez nutnosti připojení k centrální autoritě či serveru. Takový klienti mohou pracovat mimo internet, například v zabezpečené lokální síti firmy. Mezi nejznáměnší příklady takovýchto protokolů lze zařadit XMPP a IRC, oba popsané níže.

[2] https://eprint.iacr.org/2008/147.pdf
[3] http://www.umtsworld.com/umts/faq.htm
[4] https://tools.ietf.org/html/rfc791
[5] API je zkratka pro *Application Programming Interface*. Jedná se o rozhraní obsahující sadu metod, funkcí či protokolů, které standardizují programátorovu práci.

XMPP

XMPP [16], zkratka pro *Extensible Messaging and Presence Protocol* (česky *Rozšiřitelný protokol pro posílání zpráv a zjištění stavu*), je otevřený protokol pro textovou komunikaci. Jedná se o protokol pracující na principu klient-server, tedy klient se připojí k serveru, skrz který komunikuje s ostatními klienty. Jeho primární využití je pro takzvaný IM, což je zkratka pro *Instant Messaging* (česky volně *Okamžité komunikace skrz zprávy*). Rozdíl oproti IRC (viz níže) je fakt, že uživatel není omezen pouze na jednotlivé servery, ale komunikace může probíhat i mezi uživateli z jiných serverů.

Základem komunikace skrz XMPP je registrace na vybraný XMPP server. Po registraci dostane uživatel „přezdívku“ ve formě speciálního URI nazvaného Jabber ID, zkráceně JID. Toho ID má formát podobný e-mailové adrese *jmeno@server* a pomocí tohoto ID můžou ostatní uživatelé kontaktovat uživatele *jmeno* na serveru *server*.

XMPP též umožňuje zjištění stavu, tedy klient A může zjistit, zdali je klient B připojen na server a nebo aktuálně není dostupný.

Tento protokol byl dříve známý jako *Jabber*, pod tímto názvem vznikl v roce 1999 [12].

IRC

IRC [21], zkratka pro *Internet Relay Chat* (česky volně *Chat přenášený internetem*), je otevřený protokol pro textovou komunikaci. Jedná se o protokol pracující na principu klient-server, tedy klient se připojí k serveru, skrz který komunikuje s ostatními klienty. Jeho primární využití je pro zakládání tzv. *„chatovacích místností“*, ve kterých více uživatelů zaráz komunikuje. Rozdíl oproti XMPP (viz výše) je fakt, že jednotlivé IRC servery mezi sebou nekomunikují, je možno tedy komunikovat pouze s uživately na stejném serveru (bez použití nadstandardních rozšíření).

Pro IRC existuje i nadstavba, označená jako DCC [41], což je zkratka pro *Direct Client-to-Client* (česky *Přímá [komunikace] klienta s klientem*). Tato nadstavba umožňuje existenci soukromých zpráv (mezi pouze dvěma uživateli), ale také například přenos souborů.

Tento protokol byl jedním z prvních možností chatovat s uživateli přes internet, vznikl ve Finsku již v roce 1988, odkud se později s rozšířením internetové sítě přesunul i do světa.

3.3.2 Otevřené komunikátory

Tato podsekce je věnována otevřeným komunikátorům. Tímto termínem jsou v této práci označovány takové komunikátory, které sice umožňují vytvoření vlastní aplikace, nicméně ale vyžadují použití centrální autority či v nějakém smyslu neumožňují uzavřenou komunikaci mimo internet. Je to umožněno například existencí vlastního API, které umožní komunikaci se servery služby, či jako doplněk pro existující aplikaci. Mezi nejznáměnší příklad takovýchto komunikátorů lze říct Tox, popsaný níže.

Tox

Tox [14] je distribuovaný, od základu šifrovaný IM klient, umožňující bezpečnou komunikaci pomocí internetu. Kromě psaní zpráv umožňuje volání, přenos videa, sdílení souborů a další. Tox vznikl především jako reakce na zvyšující se obavy z kontroly a špehování občanů státními orgány, kolem roku 2013 [1].

Tox dává vývojáři možnost použít knihovnu, která umožní vytvořit vlastní aplikaci pro šifrované zprávy a to i bez velkých předešlých znalostí problematiky kryptografie, viz 2.1.

Aplikace je stále ve vývoji, objevují se i negativní komentáře na některé fungování celé aplikace, především na samotnou distribuovanost a decentralizovanost celého systému[6].

3.3.3 Ostatní komunikátory

Termínem „ostatní komunikátory“ jsou v této práci označovány takové komunikátory, které neumožňují ani vytvoření vlastní aplikace díky neexistenci API, ale také neumožňují lokální provoz a vyžadují neustálou komunikaci s centrálními servery služby. Tento fakt značně přidává na složitosti efektivně přidat šifrovanou komunikaci do konverzací s uživateli na těchto komunikátorech. Mezi takové komunikátory patří například Skype[7], WhatsApp[8] a mnoho dalších.

3.4 Existující řešení

Při vypracování nové aplikace je velice vhodné se podívat na již existující produkty a porovnat cílenou funkcionalitu s funkcionalitou těchto produktů. Všechny aplikace byly nalezeny na Google Play[9], při hledání pod tagy *„sms“* a *„encrypted“*, uvedená hodnocení jsou hodnoty z Google Play. Následuje analýza vybraných aplikací a výběr možných prvků pro naší aplikaci.

Výchozí SMS aplikace

Výchozí SMS aplikace v Android 5.1 se skládá z rozhraní „tří oken“. V prvním okně „seznam konverzací“ aplikace standardně začíná. Při poklepnutí na libovolnou konverzaci se daná konverzace otevře v novém okně, kde se zobrazí historie konverzace s daným kontaktem ve formě okna „detail konverzace“. Poslední možné „okno“ je „nová konverzace“, která umožňuje zasílání nové zprávy specifickému kontaktu a založení nové konverzace. Po odeslání zprávy je uživatel přesměrován právě do okna „detail konverzace“.

Při podržení ukazatele/prstu na určitém prvku (ať již konverzace či zpráva přímo v konverzaci) dojde k zobrazení nabídky dalších akcí. Jedná se například o smazání zprávy či konverzace, ale také kopírování textu zprávy či označení jako přečtené.

Aplikace též upozorní na chyby při odesílání, ať již nevyžadující zásah uživatele (například opožděné odeslání z důvodu ztráty signálu) či naopak takové, u kterých je potřeba SMS znovu odeslat (například došlo ke kritické chybě při odesílání).
Snímek aplikace je vyobrazen na obrázku 3.1a.

Encrypted SMS

Hodnocení: 4,7 / 5
Encrypted SMS[10] je druhá nejlepší aplikace z výběru. Pro každou konverzaci nabízí možnost výměny klíčů pro asymetrickou šifru, symetrickou nepodporuje. Po výměně si pamatuje, že k ní došlo a automaticky šifruje odchozí SMS. Aplikace nefunguje jako náhrada výchozí SMS aplikace, práce s ní je tedy problematická, viz nevýhody níže.

[6] https://github.com/irungentoo/toxcore/issues/1398
[7] https://www.skype.com/
[8] https://www.whatsapp.com/
[9] https://play.google.com/store
[10] https://play.google.com/store/apps/details?id=pl.ipage.encryptedsms

Mezi výhody jistě patří příjemné uživatelské rozhraní a přítomnost nápověd, například upozorní, že odchozí SMS nebude šifrovaná před výměnou klíčů. Aplikace také umožňuje nastavení tzv. *„Master Password“*, tedy hesla které je nutné zadat při každém spuštění aplikace, jinak není uživatel do aplikace puštěn. Další vrstvou bezpečnosti je i blokování vytváření snímku plochy a tedy dalším znesnadněním odcizení dat.

Nevýhodou je plýtvání místa v podobě pevně dané minimální velikosti textu na 6 řádků, krátké texty tedy zbytečně zabírají místo (na testovacím telefonu se pod sebe vešlo maximálně 5 zpráv). Taktéž je nepříjemné vynucování šifrování, při pokusu o nezašifrovanou SMS se objeví upozornění, není tedy možné mít pohodlně nezašifrované některé konverzace.
Snímek aplikace je vyobrazen na obrázku 3.1b.

SMS Encrypt

Hodnocení: 3,9 / 5
SMS Encrypt[11] se při testování na referenčním telefonu sice spustí, ale nelze přijímat, odesílat ani číst zprávy. Dle snímku uživatelského prostředí se jednalo nejspíše o aplikaci pro Android 4.4, proto mohly nastat problémy z důvodu kompatibility.
Snímek aplikace je vyobrazen na obrázku 3.1c.

Silence

Hodnocení: 4,6 / 5
Silence[12] je subjektivně nejlepší aplikace z výběru. Ihned při startu nabídne nastavení sebe samé jako výchozí aplikaci pro SMS. Následně nabídne přesun všech SMS do chráněné a zašifrované části uložiště aplikace, což zvyšuje bezpečnost, ale i omezí práci s SMS ostatními aplikacemi. Umožňuje stejně jako *Encrypted SMS* pouze asymetrickou šifru, nevyžaduje ale pro každou zprávu zašifrovaný stav.

Výhodou je moderně vypadající uživatelské rozhraní, jednoduchost ovládání a rychlost běhu. Též jako uživatelsky přívětivou vlastnost lze označit možnost změny barvy jednotlivých konverzací, je tedy možné pro různé kontakty mít různé barvy konverzací. Pro jednotlivé zprávy v konverzacích je i elegantně zobrazen stav přijetí zprávy, podobně jako je tomu například v aplikaci WhatsApp[13].

Nevýhodou je opět pouze asymetrická šifra, která taktéž potřebuje vlastnění aplikace na druhé straně. Není tedy možné poslat někomu zašifrovanou zprávu pomocí symetrické šifry pokud například nevlastní OS Android, či obecně chytrý telefon.
Snímek aplikace je vyobrazen na obrázku 3.1d.

Text Encryption

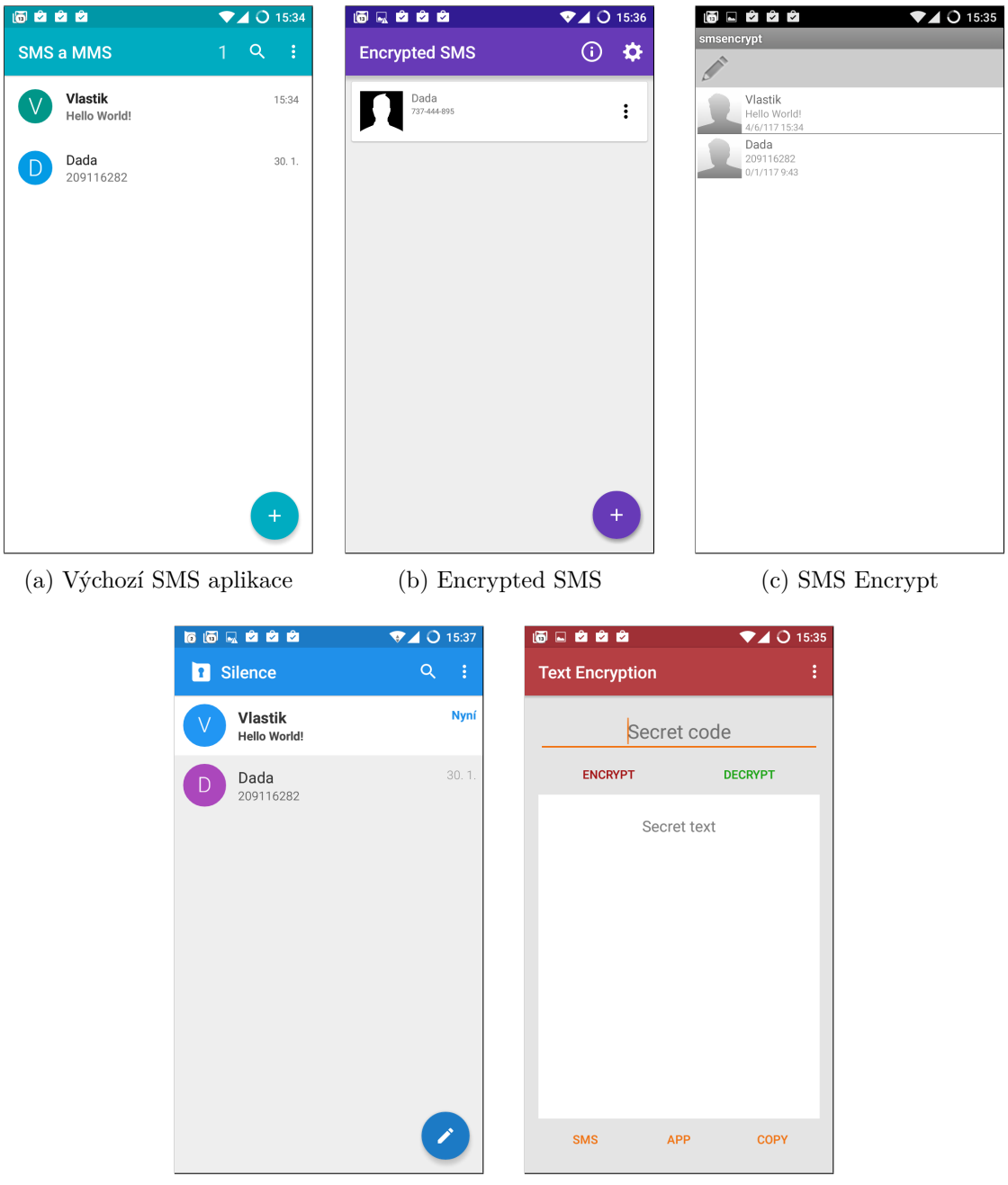
Hodnocení: 3,7 / 5
Text Encryption[14] je aplikace, která sice neumožňuje přímo odesílat šifrované SMS, pouze umožňuje šifrování vybraného textu a následné sdílení, kde již lze vybrat SMS aplikace pro odeslání. Jedná se o jednoduchou, „jednooknovou“ aplikaci, která obsahuje pouze místo pro text, místo pro heslo a dvě tlačítka „encrypt“ a „decrypt“. Tato dvě tlačítka zašifrují, resp. dešifrují, zadaný text pomocí hesla, v případě zadání nesprávného hesla nedešifrují.

[11] https://play.google.com/store/apps/details?id=edu.udc.smsencrypt
[12] https://play.google.com/store/apps/details?id=org.smssecure.smssecure
[13] https://play.google.com/store/apps/details?id=com.whatsapp&hl=cs
[14] https://play.google.com/store/apps/details?id=dk.venja.crypto

Výhoda aplikace zpočívá v její jednoduchosti, nejsou zde zbytečné rušivé prvky a aplikace vykonává činnost, ke které byla stvořena.

Nevýhodou je neoznámení šifrovacího algoritmu, opět je tedy nutné vlastnit tuto aplikaci oběma stranami pro šifrovanou komunikaci. Při testování na některých webových utilitách pro šifrování bylo při každém pokusu o dešifrování textu za použití stejného hesla oznámena chyba.

Snímek aplikace je vyobrazen na obrázku 3.1e.



(a) Výchozí SMS aplikace (b) Encrypted SMS (c) SMS Encrypt

(d) Silence (e) Text Encryption

Obrázek 3.1: Snímky hlavních oken jednotlivých testovaných aplikací

Kapitola 4

Tvorba aplikací pro systém Android

4.1 Android OS

Operační systém Android [4] je v aktivním vývoji již od roku 2003, důležitá událost ale nastala v roce 2005, kdy společnost Google [2] odkoupila společnost Android Inc. a zakomponovala ji jako svoji dceřinou společnost [35]. V následujících letech si Google vydobyl vedoucí pozici na trhu, v současné době je 80 procent nových telefonů vybaveno OS Android [31].

Android je tzv. *„open source“*[1] operační systém, postavený na linuxovém jádru s drobnými úpravami pro potřeby mobilního nasazení (v prvních verzích Linux 2.3, v současné verzi Android 7.1 na Linux 4.4.1). Otevřenost Androidu umožnila vznik neoficiálních ROM pro telefony, například CyanogenMod (nyní již bez aktivního vývoje, pokračovatelem je LineageOS [13]) či MIUI [23].

Jednotlivé verze jsou značeny čísly, příslušnou verzí API (například verze 19 pro Android 4.4) a také sladkostí v anglickém jazyce podle abecedy.

Z možností Linuxového jádra Android využívá například nastavování odlišných práv různým uživatelům. Každá aplikace spouštěna s unikátním UID (User ID), umožňující spouštět aplikace v tzv. „sandboxu“[2], kde bez implicitního nastavení a požadavku nemá aplikace přístup k prostředkům aplikace jiné.

Jedním z problémů, které musí vývojář často řešit, je množství jednotlivých verzí Androidu na trhu. O vývoj nových aktualizací pro jádro Androidu se stará společnost Google, nicméně distribuci musí povolit a zahájit výrobce telefonu. Z důvodu nutnosti složitého testování kompatibility a s tím spojených nákladů je pro výrobce snadnější a ekonomičtější telefon vydat a pak na něj „zapomenout“. Tento fakt ukazuje následující tabulka.
Z tabulky je patrné, že téměř 27 % zařízení stále používá verzi 4.4 a nižší, pro kterou již dnes Google neposkytuje podporu. Tento fakt snižuje celkovou bezpečnost systému, jelikož se do těchto telefonů nedostanou nejnovější bezpečnostní aktualizace a zařízení jsou tedy nechráněná vůči různým druhům útoků.

[1] open source znamená *„otevřený zdroj“*. Jedná se o styl vydávání SW, kdy každý má přístup ke zdrojovým kódům daného SW.

[2] sandbox je princip, který si lze přeložit jako „pískoviště“. Aplikaci zde reprezentuje dítko, které si hraje na pískovišti a pouze tam. Mimo toto pískoviště nesmí vystoupit.

Android	2.3	4.0	4.1-4.3	4.4	5.0-5.1	6.0	7.0	7.1
API	10	15	16-18	19	21-22	23	24	25
Procento	0,7	0,7	8,1	17,1	30,1	31,8	10,6	0,9

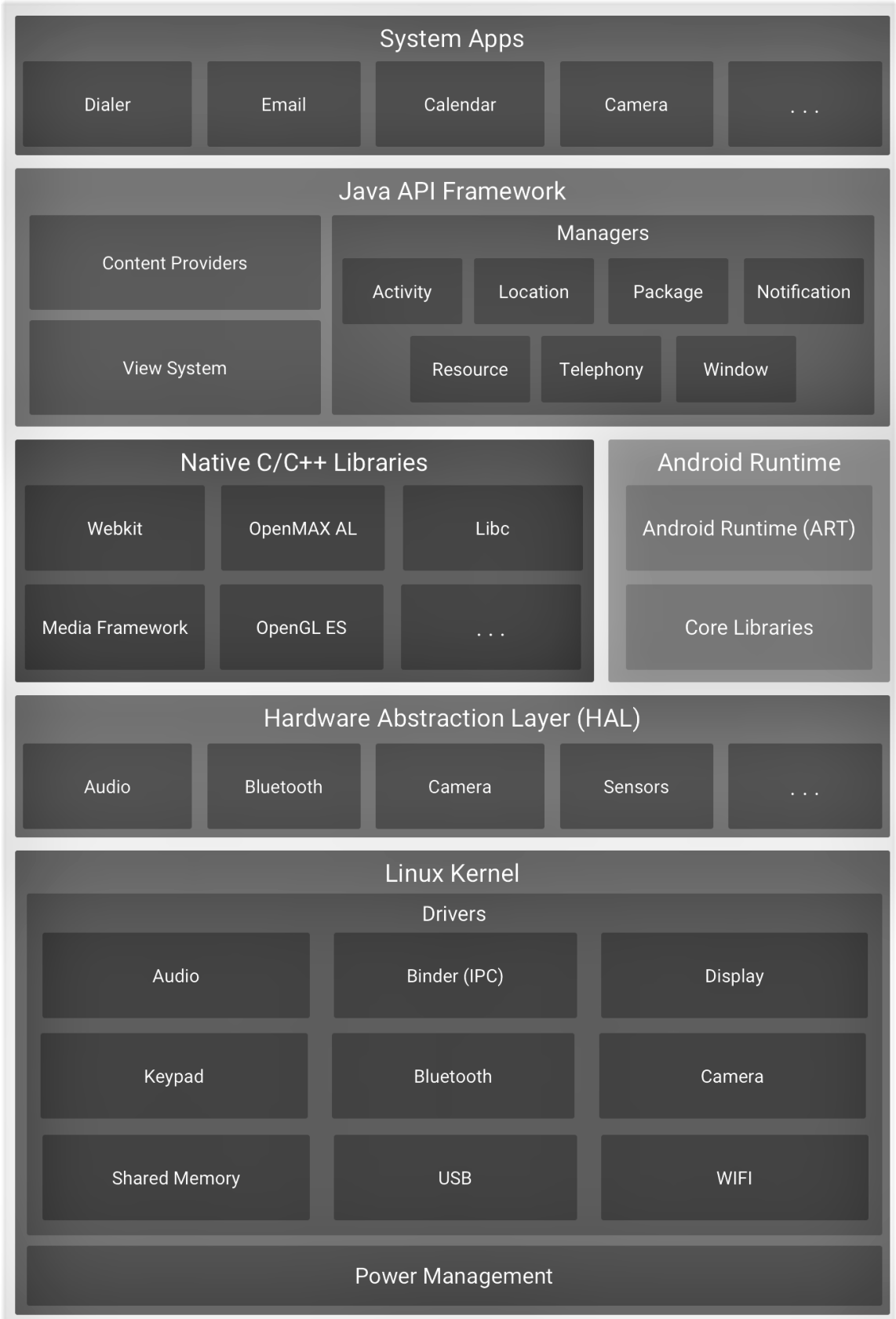
Tabulka 4.1: Procentuální rozložení verzí OS Android, stav červenec 2017 [8]

4.2 Architektura Androidu

V popisu architektury je popsán spíše model Android 5.0 (API 21 a vyšší) [17], který výrazně urychlil běh aplikací oproti staršímu modelu. Tento model je rozložen do následujících částí a vyobrazen na obrázku 4.1:

- **Linuxové jádro** – Upravené pro potřeby mobilních zařízení (vyšší výdrž, menší nároky na paměť RAM). Umožňuje například pouštět každou aplikaci v tzv. *„Sandbox mode“*, který aplikaci zamezí přístup k prostředkům jiné aplikace.
- **HAL** – Hardware Abstraction Layer, rozhraní pro abstrakci Hardware (dále jen HW). Modelovou situací je práce se specifickou HW komponentou skrz volání API, například použití fotoaparátu na zařízení. V tomto případě Android načte příslušnou knihovnu k obsluze této komponenty a umožňuje vývojáři abstrahovat práci a komunikaci Aplikace ↔ HW.
- **ART** – Android Runtime, slouží pro dopředný (AOT – Ahead-Of-Time) překlad aplikace při instalaci na zařízení z DEX[3] bytecode formátu do nativního jazyka procesoru. Tímto se model liší od předchozích verzí, které využívali překlad „za běhu“ (JIT – Just-In-Time) a využívaly interpret bytecode nazvaný Dalvik[4].
- **Nativní C/C++ knihovny** – nejsou dostupné přímo, ale pomocí Java API Framework (viz níže). Jsou zde obsaženy některé knihovny, například pro práci s grafikou (OpenGL [15], Vulkan [22]), umožňují nízko úrovňový přístup k některým částem telefonu, často HW. Knihovny psané v nativním jazyce jsou v aplikaci použity pomocí NDK (aglicky *Native Development Kit*).
- **Java API Framework** – Souhrn všech API, která jsou dostupná pro aplikací běžící v OS Android. Umožňují například práci s uživatelským rozhraním, notifikacemi a dalšími. Ulehčují práci vývojáře, abstrahují HW a standardizují komunikaci aplikací. Systémové aplikace (viz níže) využívají stejné API jako každá jiná aplikace.
- **Systémové aplikace** – Předinstalované aplikace, často bez možnosti je odinstalovat. Zajišťují základní funkcionalitu OS a HW. Patří mezi ně například SMS komunikátor, fotoaparát, webový prohlížeč a další. Od Android verze 4.4 lze většina systémových aplikací nahradit vlastní zvolenou aplikací, například již zmíněný SMS komunikátor aplikací Tombstone Talk a zastoupit tedy její funkcionalitu.

[3] DEX je mezikód, který vzniká překladem Java kódu.
[4] Dalvik je virtuální stroj pro interpretaci bytekódu DEX



Obrázek 4.1: Model architektury OS Android [17]

Kapitola 5

Návrh aplikace

5.1 Inspirace

Hlavní inspirací byla výchozí SMS aplikace v Android 5.1, a to především po stránce uživatelského rozhraní. Z této inspirace vychází návrh aplikace Tombstone Talk, popsáný v této kapitole.

5.2 Cílové požadavky na aplikaci

Aplikace by měla splňovat některé kritické požadavky na chování a uživatelské pohodlí, především:

- jednoduchá práce s aplikací
- snadná správa několika komunikačních kanálů u jednoho kontaktu
- minimální interakce uživatele s aplikací během šifrování či dešifrování
- snadné rozpoznání aktivního i neaktivního režimu šifrování
- možnost volby větší bezpečnosti na úkor pohodlnosti aplikace a naopak

5.3 Návrh

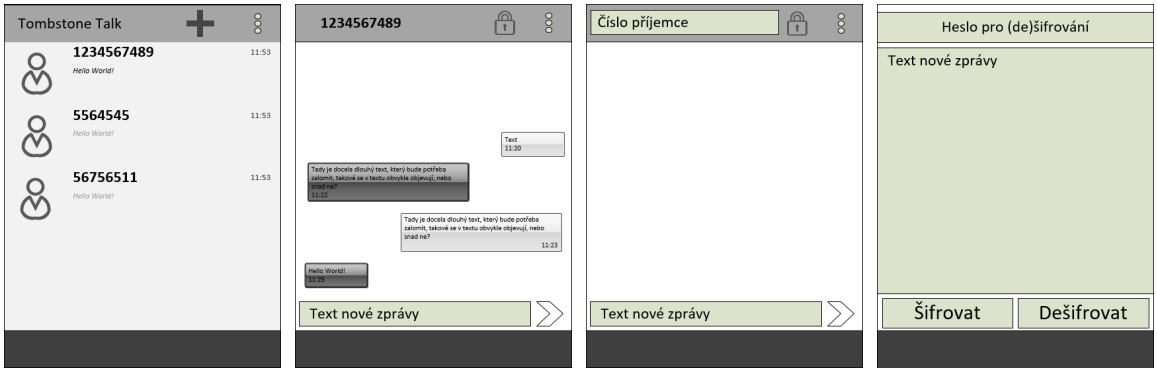
Návrh počítá s rozšířením schopnopností výchozí SMS aplikace za ponechání většiny známého prostředí. Tento přístup usnadní zažití uživatele a jeho orientaci v aplikaci, bude se tedy moci soustředit na naučení práce s novými elementy, jako je například přepínač aktivního šifrování [10].

(a) Šifrování neaktivní (b) Šifrování aktivní (c) Nastavení hesla

Obrázek 5.1: Návrhy rozhraní pro přepínání šifrování

5.3.1 Uživatelské prostředí

Jak již bylo zmíněno, uživatelské prostředí je inspirováno prostředím výchozí SMS aplikace Androidu 5.1, viz obrázek 3.1a. Jedná se tedy o „čtyřoknové“ prostředí, jednotlivá okna mohou být definována jako „seznam konverzací“, „detail konverzace“, „nová konverzace“ a „šifrovací utilita“. Obrázky 5.2a až 5.2d ukazují původní návrh jednotlivých oken.



(a) Seznam konverzací (b) Detail konverzace (c) Nová konverzace (d) Šifrovací utilita

Obrázek 5.2: Návrhy rozhraní jednotlivých oken

Seznam konverzací

Návrh seznamu konverzací počítá s jednoduchým zobrazením všech konverzací, které při klepnutí prstem zavedou uživatele do okna detailu konkrétní konverzace. Okno též bude obsahovat ikonu pro přechod na vytvoření nové konverzace, stejně jako panel s dalšími akcemi (jako je Nastavení apod.).

Návrh vyobrazen na obrázku 5.2a.

Detail konverzace

Toto okno by mělo obsahovat zobrazení všech SMS zpráv, pole pro text nové zprávy, tlačítko pro odeslání zprávy a také tlačítko pro nastavení aktivního šifrování zprávy. Implicitně bude šifrování vypnuto z důvodu zátěže na omezený počet znaků SMS.

Návrh vyobrazen na obrázku 5.2b.

Nová konverzace

Okno nové konverzace je velice podobné detailu konverzace, pouze neobsahuje seznam předchozích zpráv s kontaktem a místo označení okna číslem kontaktu se nachází na jeho místě pole pro zadání tohoto čísla.

Návrh vyobrazen na obrázku 5.2c.

Šifrovací utilita

Jednoduché okno s polem pro zadání hesla, otevřeného textu pro zašifrování a dvěma tlačítky pro zašifrování, resp. dešifrování textu.

Návrh vyobrazen na obrázku 5.2d.

5.3.2 Kódování

Při odesílání šifrované zprávy SMS bude potřeba zajistit převod z 8bitového výstupu šifrování na znaky 7bitové abecedy a to pomocí vhodného zakódování. Během návrhu byla zkoumáno několik možných řešení, ale jako nejefektivnější vyšel algoritmus Base64 (kapitola 2.2.1).

5.4 Použité technologie

5.4.1 Vývojové prostředí a programovací jazyk

Pro Android aplikaci je v současné době možno využít dvou (resp. tří) programovacích jazyků. Jedná se o kombinaci jazyku Java a volání Android API funkcí, či využití jazyku C++ a použití Android NDK (popsáno v kapitole 4.2). Jelikož aplikace nevyužívá žádnou přímou práci s HW, k čemuž by bylo třeba využít jazyk C++, či nevyžaduje nějakou C++ knihovnu, bylo zvolen jako programovací jazyk Java. Tento jazyk navíc oproti C++ ušetřuje práci programátora tím, že používá tzv. *„Garbage Collector“*[1] a navíc pro tento jazyk existuje celá řádka příruček specificky pro použití na OS Android.

Jakožto vývojové prostředí bylo opět využito standardního prostředí Android Studio[2]. Toto prostředí umožňuje rychlou práci a vývoj aplikace, navíc obsahuje integrované emulační a debugovací prostředí pro různé verze OS Android. Z použitých emulátoru to byly (všechny x86[3]) verze Android 4.4, 5.0 a 6.0. Tyto verze byly vybrány především z důvodu několika notných změn, jmenovitě například změna celkové koncepce práv aplikací ve verzi 6.0, kdy aplikace nemůže počítat s vlastněním všech potřebných práv za svého běhu a musí zjišťovat, zda-li je má k dispozici.

V současné době je i populární vývojové prostředí Xamarin[4]. Toto prostředí umožňuje psát aplikace v jazyce C# a psát aplikace pro různá platformy zařízení na trhu, například Android, iOS či Windows Phone. Vývojář (častěji ale již vývojářský tým) nemusí tedy řešit každou platformu zvlášť a sníží tím tedy celkovou práci a cenu podpory více platforem. V této práci ale přenositelnost (tedy schopnost přenést program z jedné platformy na druhou) nebyla důležitým prvkem a proto této možnosti nebylo využito.

5.4.2 Testovací zařízení

Pro testování na fyzickém zařízení bylo využito telefonu Lenovo A6000[5] s OS CyanogenMod s verzí, odpovídající Android 5.1. Jedná se telefon s procesorem Qualcomm Snapdragon 410, čtyřjádrovým CPU s frekvencí 1.2 GHz. Operační paměť telefonu je 1 GB RAM. Dodáván je s upravenou verzí Android 4.4. Jedná se tedy o telefon střední třídy, což bylo důležité pro testování výkonosti a svižnosti aplikace i na takovémto zařízení, jelikož emulátor pracoval s mnohem výkonějším HW osobního počítače (Intel Core i5 3350P 3.3 GHz, 8GB RAM).

[1] Garbage Collector je funkcionalita některých jazyků, která automaticky na pozadí kontroluje ukazatele a pokud již nejsou používány, ukazovaný prostor automaticky uvolní. Programátor tedy nemusí myslet na uvolňování zdrojů, jazyk se o to postará sám.

[2] https://developer.android.com/studio/index.html

[3] http://www.x86-guide.com/

[4] https://www.xamarin.com/

[5] https://mobilni-telefony.heureka.cz/lenovo-a6000/specifikace/#section

5.4.3 Android API

Nejvíce využívanými částmi tohoto API jsou v aplikaci Tombstone Talk `android.telephony` pro práci s SMS a také `android.support`, obsahující většinu tříd a metod pro práci s uživatelským rozhraním.

Pro práci s SMS jsou využívány především třídy `SmsManager` a `SmsMessage`, umožňující práci s příchozími a odchozími SMS.

Níže jsou vyjmenovány klíčové části Android API, které tato aplikace využívá ke své funkcionalitě.

Soubor Manifest

Soubor Manifest je důležitým prvkem každé aplikace pro Android. Definuje potřebná práva aplikace pro práci, jednotlivá „okna“ v aplikaci a třídy jím odpovídající a taktéž služby, které běží na pozadí systému i mimo běh aplikace.

Aktivity a Fragmenty

Aktivity, specificky tedy instance třídy `Activity`[6], jsou hlavní možností interakce uživatele s aplikací. Každá aplikace má jednu hlavní aktivitu, která se spouští při startu aplikace (kromě výjimek, například reakce na notifikaci), a dále neomezeně mnoho vedlejších aktivit, které vykonávají další činnosti. Každá aktivita prochází svým vlastním životním cyklem, zobrazeným na obrázku v příloze B.1. Je vhodné metody aktivity přetížit a dosáhnout tím své vlastní funkcionality (například předávání parametrů jiné aktivitě).

Součástí aktivity také mohou být části zvané fragmenty, resp. instance třídy `Fragment`[7]. Těchto částí se využívá například ke sdílení některých úseků uživatelského rozhraní, snižují tedy redundanci kódu. Každý fragment má svůj vlastní životní cyklus, je ale spjatý s cyklem rodičovské aktivity [9].

Broadcast Receiver

`BroadcastReceiver` je třída sloužící k zachycení celosystémových hlášení Broadcast (česky *všesměrová vysílání*). Tato hlášení jsou různých typů a k zachycení mnohých je potřeba speciálních pravomocí. K aplikaci využívaného hlášení `SMS_DELIVER` je potřeba deklarovat v souboru Manifest a následně od systému obdržet pravomoc `BROADCAST_SMS`.

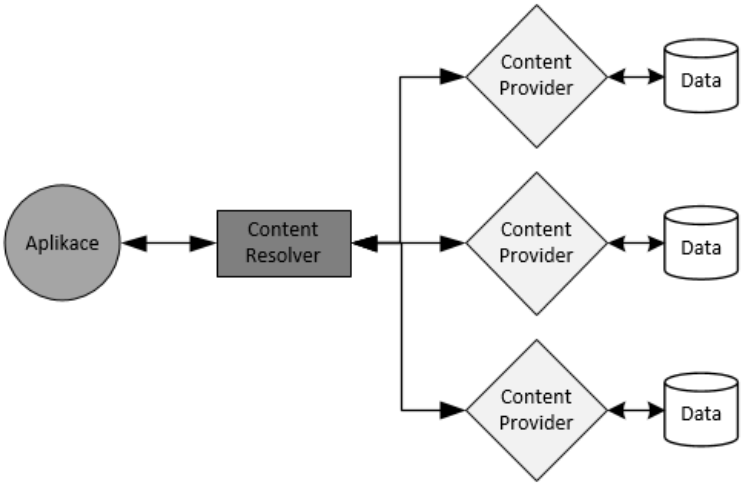
Práce s takovým hlášením probíhá pomocí třídy `Intent`, která v sobě ukrývá i detaily, v tomto případě například jednotlivé příchozí SMS.

Content Resolver / Content Provider

V každé aplikaci je dostupná jedna globální instance třídy `ContentResolver`. Tato třída se stará přijímání požadavků od uživatele a přesměrování je do patřičné instance třídy `ContentProvider`. Slouží tedy na propojení jednotlivých aplikací a sdílení jejich dat ve standardizované podobě.

`ContentResolver` tedy abstrahuje komunikaci mezi databázemi, `ContentProvider` abstrahuje práci s daty v těchto databázích, viz obrázek 5.3.

[6] https://developer.android.com/reference/android/app/Activity.html
[7] https://developer.android.com/reference/android/app/Fragment.html



Obrázek 5.3: Diagram komunikace Content Provider a Content Resolver [37]

AsyncTask

Třída `AsyncTask` slouží pro zjednodušení výpočtu jednoduchých operací na pozadí hlavního vlákna pro uživatelské rozhraní bez přímého použití vláken. Po skončení výpočtu vrátí instance třídy hodnotu zpět hlavnímu vláknu pro změnu rozhraní. Využívá se například pro načtení dat, které může trvat i několik vteřin, což by mohlo zamrznout hlavní vlákno a aplikace by se jevila jako neresponzivní.

Výhodná je především pro krátké operace. Pro delší operace (například v řádu minut) je lepší využít jiných tříd, mj. `Executor`, `ThreadPoolExecutor` či `FutureTask` [6].

5.4.4 Knihovny

Velkou část funkcionality je nevhodné programovat znovu, ale lepší by bylo využít již existujících knihoven dodávající tyto funkcionality. V této podsekci jsou projednávány knihovny v aplikaci použité.

SmsLib

SmsLib[8] je knihovna je použita pro pokročilejší práci s PDU hlavičkou SMS zprávy, viz 3.2.2. V této hlavičce se může ukrývat spousta užitečných hodnot, například informace o řetězení, které standardní funkce API nedokážou jednoduše detekovat. Hlavička je vývojáři dostupná pouze v binární podobě, je potřeba z ní hodnoty vyextrahovat ručně. Tato knihovna pomáhá tento krok zjednodušit pomocí parsování.

Knihovna je implementována v jazyce Java a dodávána jako zdrojový kód.

JCharset

JCharset[9] je pomocná knihovna pro konverzi pole bytů na znaky sady GSM 03.38, které Java knihovny v základu neumožňují. Slouží především pro zjednodušení práce. Pomocí této knihovny je především kontrolována aktuální znaková sada v nové SMS zprávě.

Knihovna je implementována v jazyce Java a dodávána jako *.jar* knihovna.

[8] https://github.com/tdelenikas/smslib-v3
[9] https://www.freeutils.net/source/jcharset/

Facebook Conceal

K samotnému šifrování bylo z důvodu jednoduchosti a pravidla *„Špatné šifrování je horší, než žádné šifrování“* [33] plánováno využit knihovny Facebook Conceal[10]. Tato knihovna si dává jako hlavní prioritu jednoduchost používání a rychlost oproti standardním možnostem. Byla vytvořena především pro potřeby aplikace Facebook, pro šifrování dat na externí SD kartě v telefonu, nicméně je možné ji použít i na šifrování obecných dat, tedy i textu. Jedná se o symetrickou šifru AES v režimu GCM, neumožňuje ale implicitní výběr z několika různých šifer.

Bohužel možnosti této knihovny byly přeceněny a opakoval se tedy známý případ „Stagefright“, rozebírán například v předmětu ITS[11], kdy knihovna byla využita k funkcionalitě, pro kterou nebyla připravena a chovala se tedy nespecifikovaně. Osobně tedy byla ověřena teorie předmětu v praxi.

Knihovna je implementována v jazyce C++ a dodávána jako nativní NDK knihovna.

5.4.5 Využitá existující řešení

Z existujících aplikací, jmenovaných v sekci 3.4, bude jistě hlavní inspirace a vzor ve výchozí SMS aplikaci. Taktéž inspirací je uživatelská jednoduchost aplikace Silence, v neposlední řadě možnost samostatného šifrování textu bez odeslání SMS z aplikace Text Encryption.

Pro komunikaci skrz jinou, než GSM síť, bude využita šifrovací utilita, viz 6.4.3. Ta umožní zprávu zašifrovat a přenést i přes komunikátory, které to v základu neumožňují. Takové komunikátory jsou rozebírány například v sekci 3.3.3.

5.4.6 Kódování a šifrování

Během vybírání vhodného šifrovacího algoritmu bylo vždy potřeba uvažovat nad dvěmi protikladnými vlastnostmi odesílání šifrovaných zpráv pomocí služby SMS. Na jednu stranu je nutné zajistit maximální bezpečnost před, během a po přenosu zprávy, a to pomocí co největšího počtu bezpečnostních opatřeních, na stranu druhou tato opatření ale často zvyšují délku původního textu. Je tedy potřeba se rozhodnout, který z těchto principů bude více upřednostněn, případně dát možnost uživateli vybrat. Taktéž je potřeba zohlednit rychlost předpokládaných koncových zařízení, kdy se často bude jednat o mobilní čipy ne příliš velkého výkonu.

Po zvážení bylo rozhodnuto využít šifrovacího algoritmu AES pro šifrování pro jeho rychlost a bezpečnost. Jako výchozí režim byl vybrán ECB (Electronic Codebook, vysvětlen v kapitole 2.1.3). Tento režim je kompromisem mezi vyšší bezpečnostní jiných režimů (jako například CBC či OFB) a zároveň jejich navýšením velikosti textu. Též je pro zprávu použito Huffmanova kódování, jehož teorie je popsána v kapitole 2.1.3 a speciální úprava pro použití v telefonech v kapitole 6.4.1. Jelikož je výstup šifrovací metody 8bitové, je pro přenos GSM zprávy využito Base64 (popsané v kapitole 2.2.1), tedy kódování, které 8bitovou zprávu převede do znaků zapsatelných 7 bity.

[10] http://facebook.github.io/conceal/

[11] Přednáška ze dne 11. 2. 2016, záznam dostupný online z https://video1.fit.vutbr.cz/av/records-categ.php?id=1283, časová značka 0:55:10.

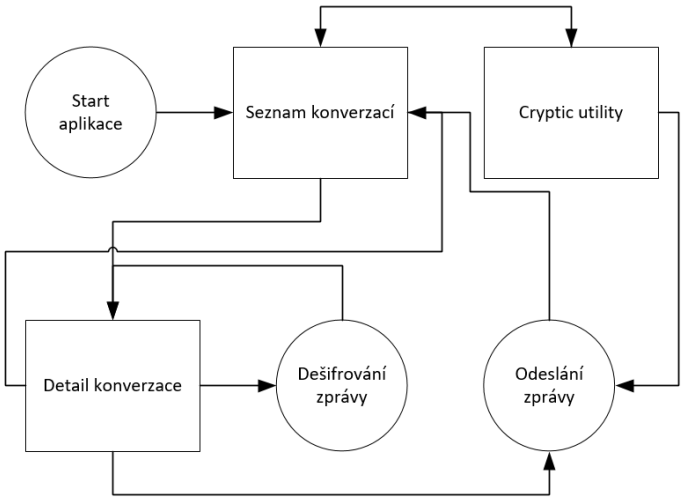
Kapitola 6

Implementace

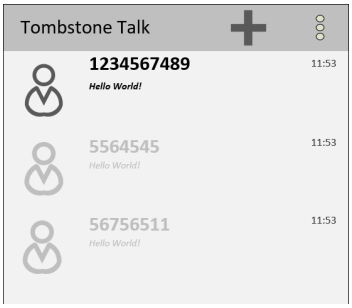
V této kapitole je popsána samotné implementace návrhu, dle kapitoly 5. Snímky implenentace jednotlivých oken jsou znázorněny na obrázku v příloze C.1.

6.1 Start aplikace

Aplikace zahajuje svoji činnost ve dvou možných stavech. Jednou z nich je otevření aplikace z nabídky všech aplikací, jejíž diagram je naznačen na obrázku 6.1. V tomto případě se tedy otevře aplikace v okně „seznam konverzací“ (viz obrázek C.1a). Na pozadí se kontaktuje Content Provider pro SMS a MMS zprávy a načtou se z každé konverzace poslední zprávy spolu s informací, zda-li byla poslední zpráva této konverzace již přečtená. V případě že nebyla, je zpráva označena modrou barvou pro lepší přehlednost (viz obrázek 6.2).



Obrázek 6.1: Naznačení principu stavového automatu aplikace



Obrázek 6.2: Naznačení principu zvýraznění nepřečtené zprávy (první)

Druhou možností startu aplikace je reakce uživatele na notifikaci oznamující novou zprávu. V tomto případě aplikace zahajuje svoji činnost přímo v okně „detail konverzace“ (viz 5.3.1) s konverzací, která obsahuje zprávu z notifikace. Zde se opět na pozadí kontaktuje Content Provider pro SMS a MMS zprávy, nicméně se ale načítají všechny zprávy pouze pro danou konverzaci a zobrazí se na obrazovce.

V obou případech je kontaktován Content Provider pomocí třídy `ListFragment`, rozšířenou o `CursorAdapter`, která zajišťuje aktuálnost hodnot, automatické posouvání a „recyklaci“[1] pohledů.

6.2 Příchozí SMS

Aplikace pro příchozí SMS využívá třídu `ReceiveSMS` odvozenou od `BroadcastReceiver`, která poslouchá Broadcast typu `SMS_DELIVER` a následně ze zprávy `Intent` pomocí standardní funkce `getMessagesFromIntent()`, která rozdělí veškeré zprávy do samostatných objektů `SMSMessage`. Tato funkce rozdělí i zřetězené SMS do samostatných, zde je využito funkcí z knihovny `PduUtils`. Tyto funkce jsou použity pro pokročilou práci s hlavičkou SMS, v tomto případě pro získání referenčního čísla, které je pro všechny členské zřetězené SMS stejné, a identifikátoru SMS, které jednoznačně určuje danou SMS v řetezi.

Následně proběhne základní kontrola a řazení příchozích SMS, které putují jako parametr do `AsyncTask` pro další kontrolu, především konzistence, a následně k uložení do databáze SMS a MMS, což zajistí Content Resolver. Ukládání pomocí API zajistí kompabilitu mezi různými zařízeními (databáze může být na telefonu různých výrobců fyzicky uložena na různých místech). V `AsyncTask` probíhá také spojování zřetězených SMS do jedné, včetně náhrady nedostupných částí.

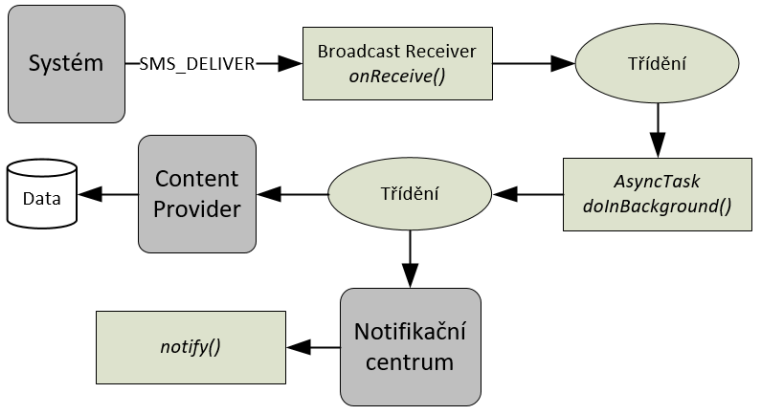
Případné dešifrování je inicializováno uživatelem při podržení prstu nad libovolnou zprávou v okně „detail konverzace“. V tomto případě se uživateli zobrazí dialog žádající o zadání klíče, který bude použit pro dešifrování.

Při potvrzení se zadaný klíč použije k zavolání statické metody `decrypt()` třídy `Cryptic` (viz sekce 6.4.3. Tato metoda inicializuje novou instanci třídy `Cipher` pomocí dodaného šifrovacího klíče. Před šifrováním je ještě text převeden z formátu Base64 (viz sekce 2.2.1) na pole bytů.

Následně proběhne samotné dešifrování. Pokud je zpráva v kódování GSM 03.38, je ještě nutné zprávu dekódovat z Huffmanova kódování, voláním metody `decode()` třídy `HuffmanCoding` (viz sekce 6.4.1).

[1] Tato *recyklace* pohledů zajišťuje, že při posouvání obrazovky se zmizelé a již naalokované pohledy využijí pro nově načtené a potřebné pohledy.

Diagram zpracování příchozí SMS je naznačen na obrázku 6.3.



Obrázek 6.3: Diagram zpracování příchozí SMS

6.3 Odchozí SMS

Pro odchozí SMS je využito standardních funkcí třídy `SmsManager`, opět především pro zajištění maximální kompatibility na všech zařízeních. `SmsManager` obsahuje funkce `sendTextMessage()` a `sendMultipartTextMessage()`, které zajistí odeslání regulérní, resp. zřetězené SMS, včetně vytvoření správné hlavičky. Z těchto funkcí je v obou případech využita pouze `sendMultipartTextMessage()`, pro kterou dodáváme data rozdělená funkcí `divideMessage()`, která text rozdělí do patřičných částí včetně kontroly kódování. V případě UCS-2 tedy SMS zprávy správně rozdělí po 70, resp. 67 znacích.

Dále je též možno k SMS přibalit objekt třídy `Intent` (resp. dědící objekty `PendingIntent`) pro oznámení o odeslání a přijetí na druhém zařízení, které je v této aplikaci využito pro označení zprávy v chatu (například odeslaná, ale ještě nepřijatá SMS je označena žlutě).

Jelikož tato funkce ale neukládá odeslané SMS do databáze, je potřeba je stejně jako u příchozích SMS manuálně vložit skrz Content Resolver, opět pro zajištění maximální kompatibility.

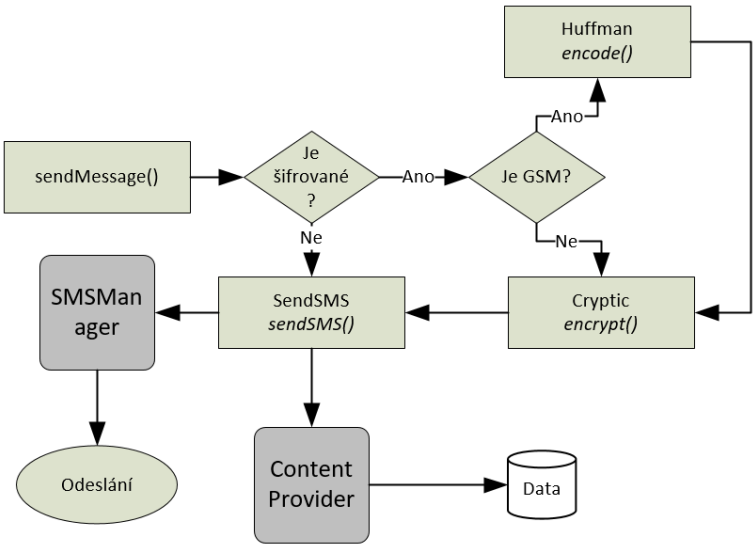
Pokud je zapnuté šifrování, tak před samotným voláním šifrovacích a odesílacích funkcí je ještě na základě aktuálního kódování (GSM či UCS-2) provedena komprimace Huffmanovým kódováním, metodou `encode()` třídy `HuffmanCoding` (viz kapitola 6.4.1). Toto kódování je použito pouze u znakové sady GSM 03.38, neb u UCS-2 by bylo naprosto nereálné a neefektivní jej použít. Kódování zajistí i zarovnání, vysvětlené v části 2.1.3.

Šifrování, pokud je nastavené, probíhá pomocí statické metody `encrypt()` pomocné třídy `Cryptic`. Tato metoda inicializuje novou instanci třídy `Cipher` pomocí dodaného šifrovacího klíče. Následně proběhne samotné šifrování. Výstup šifrování je převeden do formátu Base64, pro kompatibilitu přenosu.

V případě neúspěchu zobrazí *Toast Notifikaci*[2] s chybovou hláškou.

Diagram zpracování odchozí SMS je naznačen na obrázku 6.4.

[2]Toast Notifikace je drobný panel zobrazující se ve spodní části obrazovky. Používá se pro oznámení informací, které nevyžadují žádnou další interakci s uživatelem.



Obrázek 6.4: Diagram zpracování odchozí SMS

6.4 Implementace rozšíření

6.4.1 HuffmanCoding

Instance třídy `HuffmanCoding` je při spuštění pokaždé inicializována a vytváří Huffmanův strom (viz kapitola 2.2.2). Tento strom je pokaždé stejný, jelikož se vytváří podle stejně serializovaného[3] stromu, jež vznikl během testování na základě analýzy reálných textových zpráv v anglickém jazyce. Jedná se tedy o statickou variantu Huffmanova kódování. Toto provádí statická metoda `initStatic()`, která pouze vytvoří novou instanci třídy `AsyncTask` (viz kapitola 5.4.3) a tato již provádí vytváření stromu a kódovací tabulky na pozadí. Po dokončení je po dobu běhu aplikace dostupný globální (v rámci aplikace) objekt `globalHuffman`, pomocí kterého se provádí kódování.

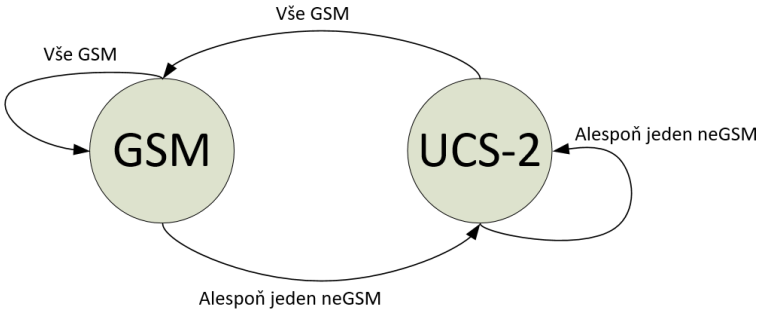
Samotné kódování je voláno metodou `encode()`, která vstupní bytové pole převede na výstupní zakódované bytové pole. V této metodě se vstupný bytové pole prochází jedno po druhém a reprezentované znaky se naleznou v kódovací tabulce, jejich kód se vloží do výstupního řetězce. Tento řetězec je na konci rozdělen zpět po osmi znacích na byte a dorovnán pomocí bitového zarovnání, viz kapitola 2.1.3.

Operace dekódování probíhá opačně, pomocí metody `decode()`. V tomto procesu se nejdříve odstraní předchozí zarovnání a pole bytů převede na binární řetězec. Tento řetězec se prochází znak od znaku (`0` nebo `1`) a systematicky se prochází Huffmanův strom.

6.4.2 Encoding Watcher

`EncodingWatcher` je pomocná třída, která zpracovává kontrolu změny kódování při psaní nové zprávy. V případě změny na toto upozorní příslušný prvek. Následně je možno upravit uživatelské rozhraní a případné kódování dle aktuální znakové sady. V aplikaci je tato třída využita pro predikci počtu zpráv. Třída zareaguje na přítomnost nepovolených znaků v sadě GSM 03.38 a vlákno uživatelského rozhraní na tento fakt upozorní. Uživatelské rozhraní

[3]Serializace objektu se nazývá proces, který instanci objektu zapíše do některého textového formátu. Z tohoto formátu lze pri tzv. *deserializaci* opět vytvořit stejná instance objektu, jako před serializací.



Obrázek 6.5: Diagram stavového automatu třídy `EncodingWatcher`

reaguje pomocí změny maximálního počtu znaků na zprávu a přepočtem potřebných zpráv pro odeslání textu. Inverzní akce se provádí v případě, že celý text obsahuje pouze znaky sady GSM 03.38.

Chování této třídy je ukázáno stavovým automatem na obrázku 6.5. Stav `GSM` znamená, že naslouchající třída přijímá informaci, že text je stále pouze ve znakové sadě GSM 03.38. Pokud se tento stav změní a vytvářený text obsahuje alespoň jeden znak mimo tuto sadu, je automati přepnut do stavu `UCS-2` a adekvátně o této změně informuje naslouchající objekt. V případě, že text opět neobsahuje ani jeden znak mimo GSM sadu, je opět automat vrácen do stavu `GSM` a informuje o této změně naslouchající objekt.

6.4.3 Cryptic

`Cryptic` je třída, zapouzdřující veškeré šifrovací a dešifrovací operace pro potřeby aplikace. Obsahuje dvě statické metody, `encrypt()` a `decrypt()`, zpracující samotné šifrování, resp. dešifrování.

V metodách se nejdříve zpracuje vstupní klíč, ve formě `String`. Tento klíč se převede pomocí algoritmu `PBKDF2` (viz kapitola 2.1.6) do formátu, který dokáže zpracovat algoritmus AES, který probíhá v následující fázi. Šifrování, resp. dešifrování, probíhá v režimu ECB, se zarovnáním typu PKCS#7, obě vysvětlené v kapitole 2.1.3.

Šifrovací utilita

Tato šifrovací utilita šifruje stejným způsobem, jako odchozí zprávy SMS. Šifruje se zde pomocí šifry AES ve třídě Cryptic (viz 6.4.3), nedochází ke kompresi Huffmanovým kódováním (viz 2.2.2), ale dochází k zakódování pomocí Base64 (který je rozebrán v sekci 2.2.1).

Takovouto zašifrovanou zprávu je poté možno vložit do schránky zařízení a vložit do konverzace jiného komunikátoru, například zmiňovaný WhatsApp. Protistrana si může přijatou zprávu téže zkopírovat, otevřít aplikaci Tombstone Talk a pomocí tlačítka *DECRYPT* získat zprávu zpět. V obou případech potřebují obě strany použít stejný klíč.

6.5 Omezení aplikace

Každá aplikace, která má nahrazovat výchozí SMS aplikaci, musí také ve svém souboru Manifest (viz kapitola 5.4.3) deklarovat podporu a třídy pro zpracování MMS, `respond-via-message` (sloužící pro odeslání odpovědi na hovor pomocí SMS) a `sendto`

(sloužící pro odeslání SMS či MMS na základě požadavku jiné aplikace). Tato deklarace musí být přítomna, i když aplikace de facto tyto funkcionality nepodporuje.

Aplikace v základním stavu z výše uvedených podporuje pouze odesílání SMS, pro ostatní funkcionality jsou připraveny zatím neaktivní třídy. V současné verzi je též omezeno aktivní šifrování pouze na symetrickou šifru AES, viz kapitola 2.1.4.

Zároveň je zde funkcionalita omezena na odesílání SMS zpráv, aplikace neumí nativně pracovat s jinými protokoly/komunikátory. Pro takovéto případy je implementována Šifrovací utilita, popsaná v 5.3.1.

V současné verzi také nefunguje dodatečná kontrola konzistence zřetězených SMS, pokud tedy části SMS přijdou se zpožděním, budou zobrazeny jako více SMS s chybějícími částmi.

Kapitola 7

Testování aplikace

Tato kapitola pojednává o procesu testování a získaných informací ze zpětné vazby uživatelů. Samotné testování probíhalo ve dvou fázích.

První fáze probíhala již během vývoje a jednalo se o testování funkcionality s použitím prostředí *Android Studio*, které umožňuje využít debugger[1]. Zde se jednalo především o zkoušení různých uživatelských pohybů v aplikaci. Dále byla aplikace nasazena na fyzickém telefon *Lenovo A6000*, které ukázalo především výkon a optimalizaci pro dané zařízení.

7.1 Dotazník a zpětná vazba

Druhá fáze celkového testování započla ke konci vývojového cyklu, kdy již aplikace byla možná nainstalovat na jiná zařízení bez problému. Aplikace byla zaslána několika příbuzným s různou technickou znalostí na otestování, zpětnou vazbu a možné návrhy na zlepšení. Před osobním testováním se někteří uživatelé nedozvěděli žádné informace, z testů tedy bylo možné i usoudit, jak aplikace působí pro naprosto cizí osobu, které nebude možnost sdělit instrukce k používání aplikace. K testování byl taktéž připojen jednoduchý dotazník, který zúčastnění po otestování aplikace vyplnili.

Dotazník obsahoval následující otázky:

1. Jak dlouho Vám trvalo se v aplikaci zorientovat?
2. Jak hodnotíte uživatelské prostředí?
3. Jak snadné bylo zahájit šifrovanou komunikaci?
4. Dokážete si představit každodenní používání této aplikace?
5. Máte nějaké návrhy ke zlepšení aplikace, co vám v aplikaci chybí?

Celkem proběhlo testování u 5 lidí, zde jsou zpracovány jejich vyhodnocení.

Dana Balvínová, pracovnice školní družiny
1. „Orientace byla rychlá. Po vysvětlení je aplikace velice jednoduchá a přístupná i mé věkové kategorii a technickým znalostem.“

[1] Debugger je zvláštní program, který umožňuje programátorovi krokovat chování programu za běhu a hledat v něm chyby.

2. „Uživatelské prostředí je intuitivní, vzhled příjemný. Ikona je nápadná a zároveň nápaditá. Pochopila jsem její význam a spojení se jménem aplikace (Tombstone - Náhrobí kámen) (tři tečky, SMS).“

3. „Snadné, pouze jsem musela dávat pozor na automatickou korekci textu telefonu. Při první zprávě automaticky doplnil háček u hesla a poté jsem se rozčilovala, že to nefunguje.“

4. „Ano, určitě ne při běžné konverzaci ale například při potřebě sdělit osobní údaje je to vhodné a pro tento případ je to skvělé.“

5. „Bylo by dobré takto posílat i obrázky či fotky, měla bych lepší pocit, že moje soukromé fotky jsou opravdu soukromé.“

Anonym L, studentka
1. „Na základní orientaci mi stačilo pár minut. Program má podobné rozložení ovládacích prvků jako jiné aplikace, které ke komunikaci běžně využívám, takže zorientování se v prostředí je snadné.“

2. „Aplikace je přehledně koncipovaná a snadno ovladatelná. Po grafické stránce jsem také spokojena.“

3. „Zahájení komunikace bylo bezproblémové, ovládání je intuitivní a zvládnou ho i méně technicky zdatní jedinci.“

4. „Pro svou běžnou komunikaci aplikaci nepotřebuji, nicméně si dovedu představit její využití při posílání „tajných“ zpráv, které nejsou určeny cizím očím. Tuto vymoženost by mohli ocenit především teenageři, protože nabízí něco nového, atraktivního.“

5. „Ocenila bych rozšíření v podobě kódování „smajlíků“, které k dnešní komunikaci neodmyslitelně patří, a také možnost posílání MMS zpráv.“

Anonym H, studentka
1. „Velice rychle. Okamžitě lze vidět přijaté zprávy, ikonku pro zprávu novou i ikonku pro šifrování.“

2. „Příjemné. Barvy nejsou nijak křiklavé a s textem tvoří správný kontrast, ovšem dělení mezi jednotlivými zprávami různých odesilatelů je poněkud nevýrazné.“

3. „Bylo snadné zprávu zašifrovat a poslat, ale rozšifrovat přijatou už ne. Tedy nikde není psáno, že uživatel musí ťuknout na danou zprávu, aby ji mohl rozšifrovat.“

4. „Ano dokážu. Po pár menších úpravách by mohla alespoň pro mě být i velice užitečná.“

5. „Text zpráv je moc malý. Uživatel s vadou očí by tedy mohl mít problém. Uvítala bych ikonku nápovědy právě kvůli problému s dešifrováním popsanému výše. Dále také určení hranice textu ukázky zpráv. Pokud je zpráva moc dlouhá, je vidět jen horní půlka písmen a to nepůsobí esteticky dobře.“

Anonym P, studentka
1. „Protože jsem zvyklá s mobilními aplikacemi pracovat, zorientovat se nebyl žádný velký problém. Navíc se aplikace nijak zásadně neliší od běžných SMS aplikací. Symbol zámečku pro šifrování je také dostatečně výmluvný.“

2. „Uživatelské prostředí je přizpůsobeno účelu aplikace, a jak jsem již psala, je podobné, jako jsme u běžných SMS aplikací zvyklí. Proto mi přijde vhodné. Mohlo by být barevně o trochu více lahodící oku, to se však dá spravit v rámci dalších aktualizací.“
3. „Zahájení šifrované komunikace mi nepřipadalo těžké, zejména proto, že jsem od autora dostala pár rad, jak s aplikací nakládat. Ale předpokládám, že díky použitému symbolu zámečku by mi netrvalo dlouho na postup přijít. Přesto bych ale přidala do aplikace stručnou nápovědu.“

4. „Pokud bych chtěla své zprávy posílat šifrovaně, umím si představit, že bych aplikaci používala každodenně. Musela by ovšem ještě projít pár aktualizacemi. Co mi v aplikaci chybí, popíšu v následujícím bodu.“

5. „Jako zásadní nedostatek vnímám fakt, že aplikace není propojená s kontakty v telefonu, proto máme možnost pracovat pouze s telefonními čísly, které u většiny lidí neznáme. Také by z mého pohledu bylo vhodné aplikaci rozšířit, aby podporovala rozlišení sim karet u DualSim telefonů.“

Lenka Hrubá, asistent pedagoga
„Osobně jsem měla problém se zahájením šifrované konverzace. Ikona zámečku byla malá a přehlédla jsem ji.“[2]

7.2 Vyhodnocení odpovědí uživatelů

Prakticky každý se v aplikaci rychle zorientoval, což připisuji blízkosti uživatelského prostředí této aplikace a výchozí SMS aplikace. Se samotným vzhledem problém nebyl, jednalo se spíše o snadno upravitelné drobnosti, nicméně bude jistě vhodné pokračovat v jeho vývoji.

Funkcionalitou splnila aplikace očekávání uživatelů, objevily se občas drobné chyby, které byly nalezeny v kódu a opraveny. Uživatelé též podali hodnotné návrhy pro zlepšení aplikace, některé z nich jsou vyčteny v sekci 8.1. Některé z nich již byly zakomponovány do aplikace (například nápověda).

[2]Pozn: Uživatelka neodpověděla na celý dotazník, pouze okomentovala problémy s aplikací.

Kapitola 8

Závěr

Cílem této bakalářské práce bylo navrhnout, implementovat a otestovat aplikaci pro Android, která umožňuje komunikaci šifrovanými zprávami. Ze studia požadavků a specifikací byl vytvořen prvotní návrh na funkcionalitu a rozhraní aplikace, který byl neustále upravován a zlepšován, stejně jako samotná analýza požadavků a použité technologie.

Na základě návrhu a studia již existujících projektů bylo vytvořeno uživatelské prostředí značně připomínající výchozí SMS aplikaci, což umožňuje snadnou orientaci uživatele v nové aplikaci a snižuje čas potřebný na zaučení se. Aplikace je schopná šifrovat a dešifrovat příchozí SMS nativně, ostatní protokoly či způsoby komunikace podporuje externě pomocí samostatné šifrovací utility. Šifrování a dešifrování probíhá pomocí symetrické šifry AES, viz sekce 2.1.4, implementace popsána v kapitole 6.4.3.

Pro snížení délky zprávy a tedy i potenciální snížení ceny zprávy bylo též využito Huffmanova kódování. Pro kompatibilitu přenosu bylo využito algoritmu Base64, které převádí 8bitový výstup kódování na 7bitové znaky tisknutelné v sadě GSM 03.38, viz sekce 5.3.2.

Oproti původnímu plánu podpory asymetrické šifry a uvedení na Obchod Play bylo upuštěno z důvodu časové tísně.

Aplikace byla též otestována několika uživateli a na základě jejich zpětné vazby byl upraven návrh, implementace, ale i možné budoucí vlastnosti a funkcionality aplikace.

8.1 Budoucnost aplikace

Aplikace jistě není ve své finální podobě, je potřeba na ní ještě dále zapracovat, což je také plánováno. Po přidání několika následujících funkcionalit je možné aplikaci nahrát i na Google Play a aplikaci monetizovat. Plánované a možné funkcionality aplikace jsou:

- podpora pro kontakty
- podpora asymetrické šifry, možnost volby šifrovacího algoritmu
- možnost odesílat šifrované SMS jako datové SMS
- nativní podpora více protokolů, například XMPP
- analýza textu uživatele a návrhy pro zlepšení Huffmanova kódování
- větší kontrola konzistence dat, například v případě zřetězených zpráv

Literatura

[1] About - Tox. [Online; navštíveno 28.07.2017].
URL https://tox.chat/about.html

[2] About Us | Google. [Online; navštíveno 28.07.2017].
URL https://www.google.com/intl/en/about/

[3] The Activity Lifecycle. [Online; navštíveno 12.05.2017].
URL https://developer.android.com/guide/components/activities/activity-lifecycle.html

[4] Android. [Online; navštíveno 28.07.2017].
URL https://www.android.com/

[5] Announcing the ADVANCED ENCRYPTION STANDARD (AES). [Online; navštíveno 26.07.2017].
URL http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[6] AsyncTask. [Online; navštíveno 13.05.2017].
URL https://developer.android.com/reference/android/os/AsyncTask.html

[7] The Base16, Base32, and Base64 Data Encodings. [Online; navštíveno 28.07.2017].
URL https://tools.ietf.org/html/rfc4648

[8] Dashboards. [Online; navštíveno 12.05.2017].
URL https://developer.android.com/about/dashboards/index.html#Platform

[9] Fragment. [Online; navštíveno 12.05.2017].
URL https://developer.android.com/reference/android/app/Fragment.html#Lifecycle

[10] Google Design. [Online; navštíveno 12.05.2017].
URL https://design.google.com/

[11] GSM 03.38. [Online; navštíveno 12.05.2017].
URL https://en.wikipedia.org/wiki/GSM_03.38#GSM_7-bit_default_alphabet_and_extension_table_of_3GPP_TS_23.038_.2F_GSM_03.38

[12] History of XMPP. [Online; navštíveno 28.07.2017].
URL https://xmpp.org/about/history.html

[13] LineageOS - LineageOS Android Distribution. [Online; navštíveno 28.07.2017].
URL https://lineageos.org/

[14] A New Kind of Instant Messaging. [Online; navštíveno 28.07.2017].
URL https://tox.chat/

[15] OpenGL - The Industry Standard for High Performance Graphics. [Online; navštíveno 28.07.2017].
URL https://www.opengl.org/

[16] An Overview of XMPP. [Online; navštíveno 28.07.2017].
URL https://xmpp.org/about/technology-overview.html

[17] Platform Architecture. [Online; navštíveno 12.05.2017].
URL https://developer.android.com/guide/platform/index.html

[18] Specification # 03.38. [Online; navštíveno 12.05.2017].
URL http://www.3gpp.org/ftp/Specs/archive/03_series/03.38/0338-720.zip

[19] Specification # 03.40. [Online; navštíveno 12.05.2017].
URL
http://www.3gpp.org/ftp//Specs/archive/23_series/23.040/23040-e00.zip

[20] Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface. [Online; navštíveno 26.07.2017].
URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2793

[21] Technical Documents. [Online; navštíveno 28.07.2017].
URL http://www.irc.org/techie.html

[22] Vulkan Overview - The Khronos Group Inc. [Online; navštíveno 28.07.2017].
URL https://www.khronos.org/vulkan/

[23] Xiaomi MIUI Official Global Site. [Online; navštíveno 28.07.2017].
URL http://en.miui.com/

[24] Cracking DES : secrets of encryption research, wiretap politics & chip design. 1998.

[25] DATA ENCRYPTION STANDARD (DES). Říjen 1999, [Online; navštíveno 26.07.2017].
URL http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[26] Březen 2015, [Online; navštíveno 28.07.2017].
URL http://www.ceskatelevize.cz/ct24/svet/1513920-cina-v-cenzure-internetu-opet-pritvrdila-problemy-maji-i-zahranicni-firmy

[27] Diffie, W.; Hellman, M.: Diffie-Hellman Key Agreement Method. 1999, [Online; navštíveno 26.07.2017].
URL https://tools.ietf.org/html/rfc2631

[28] Drábek, V.: Kódování a komprese dat. [Online; navštíveno 28.07.2017].
URL http://www.fit.vutbr.cz/study/course-l.php.cs?id=12228

[29] Eastlake, D.; Hansen, T.: US Secure Hash Algorithms (SHA and HMAC-SHA). Červenec 2006, [Online; navštíveno 28.07.2017].
URL https://tools.ietf.org/html/rfc4634

[30] Eastlake, D.; Jones, P.: US Secure Hash Algorithm 1 (SHA1). Září 2001, [Online; navštíveno 28.07.2017].
URL https://tools.ietf.org/html/rfc3174

[31] Goasduff, L.; Forni, A. A.: Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016. Únor 2017, [Online; navštíveno 12.05.2017].
URL http://www.gartner.com/newsroom/id/3609817

[32] Hanáček, P.: Kryptografie. [Online; navštíveno 28.07.2017].
URL http://www.fit.vutbr.cz/study/course-l.php.cs?id=12232

[33] Jones, D.: *The Definitive Guide to Building Code Quality*. Realtimepublishers.com, 2009, ISBN 1931491909.

[34] Kaliski, B.: PKCS #5: Password-Based Cryptography Specification Version 2.0. Září 2000, [Online; navštíveno 28.07.2017].
URL https://tools.ietf.org/html/rfc2898#section-5.2

[35] Kilián, K.: Historie Androidu v kostce aneb Od verze 1.0 až po Android M. Červen 2015, [Online; navštíveno 12.05.2017].
URL https://www.svetandroida.cz/historie-androidu-201506

[36] Klíma, V.: Základy moderní kryptologie – Symetrická kryptografie I. Duben 2005, [Online; navštíveno 16.06.2017].
URL
http://crypto-world.info/klima/mffuk/Symetricka_kryptografie_I_2005.pdf

[37] Lockwood, A.: Content Providers & Content Resolvers. Červen 2012, [Online; navštíveno 13.05.2017].
URL http://www.androiddesignpatterns.com/2012/06/content-resolvers-and-content-providers.html

[38] Merkle, R.; Hellman, M.: Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, ročník 24, č. 5, Sep 1978: s. 525–530, ISSN 0018-9448, doi:10.1109/TIT.1978.1055927.

[39] Rivest, R.: The MD5 Message-Digest Algorithm. Duben 1992, [Online; navštíveno 26.07.2017].
URL https://tools.ietf.org/html/rfc1321

[40] Rivest, R.; Shamir, A.; Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. 1978, [Online; navštíveno 26.07.2017].
URL http://people.csail.mit.edu/rivest/Rsapaper.pdf

[41] Rollo, T.: A description of the DCC protocol. [Online; navštíveno 28.07.2017].
URL http://www.irchelp.org/protocol/dccspec.html

[42] Simpson, W.: PPP Challenge Handshake Authentication Protocol (CHAP). Srpen 1996, [Online; navštíveno 28.07.2017].
URL https://tools.ietf.org/html/rfc1994

Příloha A

Tabulka kódování znaků sady GSM 03.38

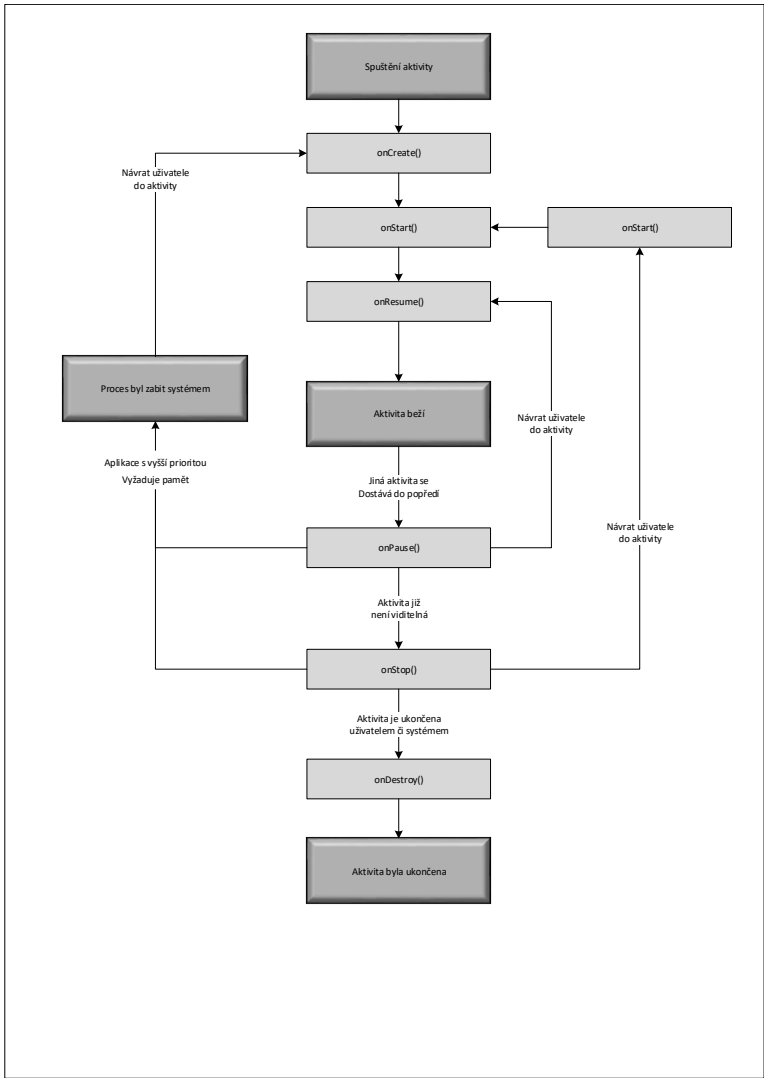
	0x00	0x10	0x20	0x30	0x40	0x50	0x60	0x70
0x00	@	Δ	SP	0	¡	P	¿	p
0x01	£	_	!	1	A	Q	a	q
0x02	$	Φ	"	2	B	R	b	r
0x03	¥	Γ	#	3	C	S	c	s
0x04	è	Λ	¤	4	D	T	d	t
0x05	é	Ω	%	5	E	U	e	u
0x06	ù	Π	&	6	F	V	f	v
0x07	ì	Ψ	'	7	G	W	g	w
0x08	ò	Σ	(	8	H	X	h	x
0x09	Ç	Θ	)	9	I	Y	i	y
0x0A	LF	Ξ	*	:	J	Z	j	z
0x0B	Ø	ESC	+	;	K	Ä	k	ä
0x0C	ø	Æ	,	<	L	Ö	l	ö
0x0D	CR	æ	-	=	M	Ñ	m	ñ
0x0E	Å	ß	.	>	N	Ü	n	ü
0x0F	å	É	/	?	O	§	o	à

	0x00	0x10	0x20	0x30	0x40	0x50	0x60	0x70
0x00					|			
0x01								
0x02								
0x03								
0x04		^						
0x05							€	
0x06								
0x07								
0x08			{					
0x09			}					
0x0A	FF							
0x0B		SS2						
0x0C				[				
0x0D	CR2			~				
0x0E				]				
0x0F			\					

Obrázek A.1: Tabulka kódování znaků sady GSM 03.38 [11]

Příloha B

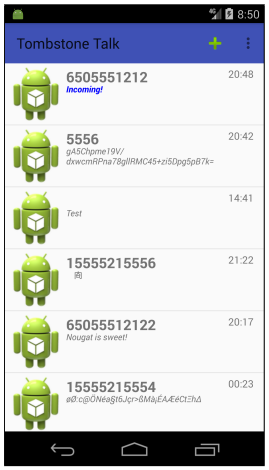
Obrázek životního cyklu Aktivity



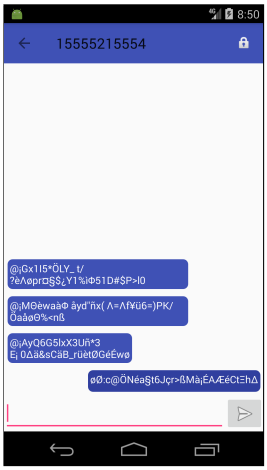
Obrázek B.1: Životní cyklus aktivity [3]

Příloha C

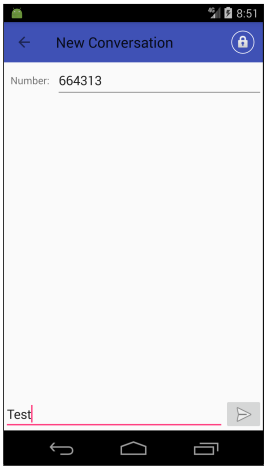
Snímky jednotlivých oken aplikace



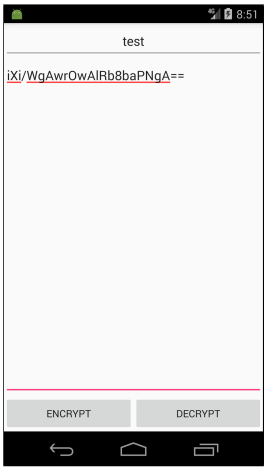
(a) Seznam konverzací



(b) Detail konverzací



(c) Nová konverzace



(d) Šifrovací utilita

Obrázek C.1: Snímky implementace jednotlivých oken

Příloha D

Obsah CD

Na přiloženém CD se nachází adresáře:

- src – zdrojové kódy aplikace
- bin – spustitelná aplikace ve formátu APK
- pdf – text této práce ve formátu PDF
- tex – text této práce ve formátu LaTeX