VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PLATFORMA PRO BEZPEČNÝ PŘENOS SOUBORŮ MEZI MOBILNÍMI ZAŘÍZENÍMI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MICHAL PUSTKA
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PLATFORMA PRO BEZPEČNÝ PŘENOS SOUBORŮ MEZI MOBILNÍMI ZAŘÍZENÍMI

PLATFORM FOR SAFE FILE TRANSFER AMONG MOBILE DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL PUSTKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. MILOŠ MINAŘÍK
SUPERVISOR

BRNO 2015

Abstrakt

Práce se zabývá bezpečným přenosem souborů mezi mobilními zařízeními. V práci je také provedena analýza a zhodnocení současných řešení přenosu souborů mezi mobilními zařízeními a současných komplexních bezpečnostních řešení operačního systému Android. Dále také byly navrženy a zhodnoceny koncepty řešení pro přenos šifrovaných souborů mezi skupinami uživatelů a řešení problému přenosu šifrovaných souborů mimo firmu či instituci. Následně byl vybrán jeden z konceptů přenosu šifrovaných souborů mezi skupinami uživatelů a provedena jeho implementace na operačním systému Android. Na této implementaci byla popsána řešení a problémy, které se mohou vyskytnout při vývoji takového systému. Nakonec byla ověřena funkčnost a správnost celého konceptu.

Abstract

The thesis deals with safe files transfer among mobile devices. The first part of this thesis is the analysis and review of current solutions for transferring files among mobile devices and complex security solutions for Android operating system. The second part describes the design and evaluation of several concepts for encrypted files transfer among groups of regular users and outside the company or institution. One of the concepts for file transfer among users was chosen for implementation on the Android platform. The last part of the thesis describes the problems that can occur during the implementation, their solution, and finally the verification and validation of the concept itself.

Klíčová slova

Bezpečnost, Android, mobilní operační systémy, kryptografie, šifrování souborů, distribuce klíčů, bezpečné úložiště klíčů, koncept uživatelských skupin

Keywords

Security, Android, mobile operating systems, cryptography, files encryption , key distribution, secure key storage, user groups concept

Citace

Michal Pustka: Platforma pro bezpečný přenos souborů mezi mobilními zařízeními, bakalářská práce, Brno, FIT VUT v Brně, 2015

Platforma pro bezpečný přenos souborů mezi mobilními zařízeními

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana ing. Miloše Minaříka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Michal Pustka
18. května 2015

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářské práce ing. Miloši Minaříkovi za odborné vedení, za pomoc a rady při zpracování této práce. Děkuji také ing. Davidovi Lisoňkovi ze společnosti Safetica Technologies s.r.o., za jeho odborné konzultace, které mi pomohly lépe proniknout do dané problematiky.

© Michal Pustka, 2015.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod **3**

2 Současný stav **4**
2.1 Řešení bezpečného přenosu souborů mezi mobilními zařízeními 4
2.1.1 Secret Space Encryptor 4
2.1.2 Boxcryptor 4
2.1.3 Cloudové služby 5
2.2 Komplexní bezpečnostní řešení mobilních systémů 5
2.2.1 Samsung KNOX 5
2.2.2 IBM Lotus Mobile Connect 6
2.2.3 Citrix XenMobile 7

3 Šifrování **9**
3.1 Kryptografie 9
3.2 Symetrická kryptografie 9
3.3 Asymetrická kryptografie 10
3.4 Blokové šifry a jejich režimy 10

4 Přenos souborů mezi skupinami uživatelů **12**
4.1 Skupiny uživatelů definované na serveru 12
4.1.1 Případová studie 13
4.1.2 Zhodnocení 13
4.2 Skupiny uživatelů definované v souboru 13
4.2.1 Případová studie 14
4.2.2 Zhodnocení 14
4.3 Dvojí klíč skupiny (dočasný a perzistentní) 15
4.3.1 Případová studie 15
4.3.2 Zhodnocení 15

5 Přenos souborů mimo organizaci **17**
5.1 Ověřování uživatele přes SMS zprávu 17
5.1.1 Případová studie 18
5.1.2 Zhodnocení 18
5.2 Ověřování uživatele telefonicky 18
5.2.1 Případová studie 19
5.2.2 Zhodnocení 19
5.3 Ověření uživatele před zasláním souboru 19
5.3.1 Případová studie 19

5.3.2 Zhodnocení 20

6 Implementace **21**
6.1 Implementace klientské části 21
6.1.1 Bezpečné úložiště klíčů 21
6.1.2 Implementace vlastního správce klíčů 22
6.1.3 Implementace správce skupin 23
6.1.4 Třída `SymetricCipher` 24
6.1.5 Třída `AsymetricCipher` 25
6.1.6 Zabezpečení komunikace mezi klientem a serverem 25
6.1.7 Komunikace se serverovou části 25
6.1.8 Práce se soubory 27
6.1.9 Implementace uživatelských akcí 28
6.1.10 Implementace rozhraní pro přihlášení uživatele 29
6.2 Implementace serverové části 29
6.2.1 Server 29
6.2.2 Databáze 30

7 Ověření funkčnosti **31**
7.1 Případová studie 31
7.2 Postup ověření správnosti a funkčnosti 31

8 Závěr **32**

A Obsah CD **36**

Kapitola 1

Úvod

V dnešní době boomu chytrých mobilních zařízení, kdy takřka každý vlastní chytrý mobilní telefon či tablet, nastává otázka, jak řešit bezpečnost na mobilních platformách či bezpečný přenos souboru z/do mobilního zařízení. Hlavní oblast, kde je potřeba tuto otázku řešit, je především firemní sektor a sektor veřejné správy. Zaměstnanci dnes běžně používají chytrá mobilní zařízení pro komunikaci či sdílení souborů s kolegy v rámci firmy nebo mimo firmu. Zde může nastat nebezpečí úniku citlivých dat mimo firmu, které může způsobit ztrátu výhody nad konkurencí, únik důležitého know-how nebo sankce za nesprávné nakládaní s osobními údaji klientů či zaměstnanců. Důležité je si uvědomit, že k úniku citlivých dat nemusí docházet vědomě přičiněním zaměstnance, ale také napadením mobilního zařízení škodlivým softwarem nebo odposloucháváním linky útočníkem zvenčí. Proto je také důležité zabývat se částečně i komplexním bezpečnostním řešením mobilních platforem (např. dodržování bezpečnostních politik či odepření přístupu k citlivým souborům potenciálně nebezpečným aplikacím).

Tato práce se zabývá zejména řešením bezpečného sdílení citlivých souborů mezi mobilními zařízeními s operačním systémem Android uvnitř firemní sítě i mimo ni. Ačkoliv je tato práce zaměřena výhradně na mobilní zařízení s operačním systémem Android, veškeré koncepty či návrhy řešení budou moci být uplatněny i na ostatních mobilních platformách. Dalším cílem je analýza a průzkum současných komplexních řešení mobilní bezpečnosti na platformě Android. Výsledkem této práce je implementace nalezeného řešení pro sdílení souborů mezi mobilními zařízeními se zaměřením na přenos souborů více příjemcům.

Kapitola 2

Současný stav

V této kapitole jsou rozebrána a zhodnocena současná řešení problematiky bezpečného přenosu souborů mezi mobilními zařízeními a komplexní bezpečnostní řešení mobilních systémů.

2.1 Řešení bezpečného přenosu souborů mezi mobilními zařízeními

2.1.1 Secret Space Encryptor

Aplikace Secret Space Encryptor je jedna z nejstahovanějších nástrojů pro šifrování souborů na Google Play obchodu s aplikacemi. Je k dispozici i pro desktopové operační systémy Windows a Linux, ale nabízí také odlehčenou variantu pro mobilní operační systém iOS, kde je možno šifrovat nebo dešifrovat pouze textové soubory. Verze pro Android dokáže zašifrovat či dešifrovat soubor a chránit tak obsah souboru heslem.

Tato aplikace neřeší žádnou distribuci klíčů nebo šifrovaní pro skupinu uživatelů, řešení spoléhá na fakt, že uživatel si své heslo zapamatuje. Na druhou stranu to je dobrá alternativa především pro osobní použití, protože není potřeba serveru a internetového připojeni pro synchronizaci klíčů.

2.1.2 Boxcryptor

Služba Boxcryptor nabízí bezpečnostní řešení pro sdílení souborů napříč různými cloudovými službami. Aplikace je dostupná pro široké spektrum operačních systémů včetně těch mobilních (Android, iOS, Windows Phone). Uživatelé mohou prostřednictvím cloudové služby sdílet zašifrovaný soubor. Boxcryptor umožňuje nastavit na tento soubor přístup pouze jednomu uživateli, více uživatelům nebo nastavit uživatelskou skupinu. Mezi další funkcionality patří možnost dešifrování souborů při ztrátě hesla pomocí hesla administrátora, možnost šifrování názvu souborů, vynucování základních politik tvorby hesla či možnost integrace adresářových služeb.

Pro každého uživatele je vygenerován jeden pár RSA klíčů. Přičemž privátní RSA klíč je zašifrován uživatelským heslem. Dále je také vygenerován tzv. wrapping key, který je rovněž zašifrován uživatelským heslem. Tento wrapping key je pak použit na zašifrování ostatních AES klíčů patřících danému uživateli. Všechny tyto klíče, včetně informací o uživateli, jsou pak přeneseny na tzv. Boxcryptor Key server zabezpečenou komunikací. Boxcryptor

Key server potom slouží pro distribuci klíčů. Šifrování a dešifrování souborů pak probíhá následovně:

- Uživatel A vytvoří náhodný klíč pro šifrování souboru. Tímto klíčem je potom obsah souboru zašifrován.
- Uživatel A následně zašifruje tento klíč souboru veřejným klíčem uživatele B, který získá z Boxcryptor Key serveru. Takto zašifrovaný klíč je vložen do hlavičky souboru. V případě, že je soubor určen pro více uživatelů, je pro každého uživatele zašifrován klíč souboru jeho veřejným klíčem.
- Uživatel B svým privátním klíčem dešifruje klíč souboru a obsah souboru může být dešifrován.

Vytvoření uživatelské skupiny obnáší vygenerování páru RSA klíčů a členského klíče. Privátní RSA klíč je zašifrován členským klíčem a tento členský klíč je pro každého uživatele ze skupiny zašifrován jeho veřejným klíčem. Všechny tři klíče (pár RSA klíčů skupiny a členský klíč) jsou uloženy na Boxcryptor Key serveru. Pro dešifrování a šifrování klíče souboru se pak použije privátní klíč skupiny, tedy uživatel nejprve dešifruje svým privátním klíčem členský klíč a tento členský klíč použije na dešifrování privátního klíče skupiny [17].

Mezi výhody řešení od Boxcryptoru patří především široká podpora platforem, možnost spravovat uživatele přes administrátorskou konzoli či vytváření uživatelských skupin. Avšak řešení skupinového šifrování není úplně vhodné pro velké skupiny uživatelů či časté změny přidání nebo odebrání člena skupiny. Další velkou nevýhodou je fakt, že řešení se zaměřuje především na cloudové služby, nebere v potaz firemní NAS, SAMBA servery či další intranetové souborové služby. Také je nutno podotknout, že v současné době nenabízí základní free verze aplikace možnost šifrování souboru na mobilních zařízeních, nabízí pouze možnost vytvářet šifrované textové soubory či dešifrovat zašifrované soubory, které již jsou zašifrované aplikací na desktopu.

Je třeba zmínit, že předchozí dvě řešení nejsou jediná, např. podobné řešení jako Boxycryptor nabízí i služba GoCrypt. Rovněž Secret Space Encryptor není jediná aplikace pro šifrování souborů na mobilních zařízeních. Avšak všechny další alternativy se principiálně neliší od výše uvedených.

2.1.3 Cloudové služby

Samotní poskytovatelé cloudových služeb nabízejí pro firemní zákazníky produkty, kde mohou svá data na cloudovém úložišti šifrovat, případně nastavovat přístupová práva uživatelům. Toto řešení však nepokrývá požadavky zákazníků, kteří požadují mít svá data k dispozici lokálně v rámci intranetu v případě výpadku internetového připojení, či zákazníků, kteří využívají současně různé služby od různých poskytovatelů. Dalším argumentem proti je, že někteří zákazníci požadují mít nad svými daty plnou kontrolu a nepřejí si svá data ukládat do cloudových úložišť.

2.2 Komplexní bezpečnostní řešení mobilních systémů

2.2.1 Samsung KNOX

Samsung Knox je komplexní bezpečnostní řešení pro širokou škálu uživatelů od jednotlivých uživatelů po velké korporace. Technologie je implementována do více vrstev napříč zařízením a to jak do hardwaru, tak do softwaru.

První vrstva se zabývá ochranou proti nahrazení jádra operačního systému Android jiným infikovaným kernelem, v terminologii Samsung se tato vrstva označuje jako Secure Boot. Jedná se o vrstvu, která funguje na hardwarové úrovni, kdy se při zavádění jádra systému do operační paměti kontroluje, zda je v telefonu použité jádro kryptograficky podepsané ověřenými autoritami. Další částí je tzv. Trusted Boot, kde i tato část je řešena na hardwarové úrovni. Využívá sektoru TrustZone v ARM architektuře procesoru, kde se v průběhu bootu ukládají kryptografické otisky jádra a bootloaderů. Pokud na zařízení běží schválený Samsung firmware, klíče z TrustZone jsou poskytnuty ke kontrole. Režim kontroly klíčů je pak založen na unikátnosti páru soukromého a veřejného klíče. Výrobce přidělí každému zařízení unikátní pár klíčů a certifikát pro veřejný klíč, podepsaný privátním klíčem společností Samsung. Ověřovací servery zkouší zařízení a testují jejich integritu. Hodnoty z TrustZone se porovnávají s aktuálními hodnotami a výsledky se zašlou na posouzení ověřovacím serverům [20].

Další vrstva je již softwarová, je založena na nadstavbě Security Enhancements (SE) pro Android. Tato technologie pak vymezuje aplikacím přístupy k jednotlivým souborům a zdrojům. SE pro Android rozděluje operační systém do izolovaných domén. V každé této doméně jsou aplikace izolovány od zbytku systému. V rámci této domény je aplikacím dán pouze minimální přístup ke zdrojům. Pokud by pak došlo k nabourání do dané domény, útočník se nedostane ke všem zdrojům. Dále je systém rozdělen na dvě zcela separátní části: nezabezpečenou část a bezpečnou část, tj. KNOX kontejner. Aby aplikace mohla běžet v kontejneru, musí být obalena speciální vrstvou. V případě korporátního využití systému KNOX musí být cílová aplikace odeslána na server, kde je rozbalena, jsou z ní extrahovány vývojářské certifikáty a poté je zabalena s dodatkovými soubory do nového balíčku. Ten je pak odeslán na servery Samsung, kde se verifikují možná bezpečnostní rizika aplikace. Poté je aplikace odeslána zpět zákazníkovi. Tato aplikace pak může být firemním administrátorem umístěna do privátního obchodu s KNOX aplikacemi v dané firmě. KNOX rovněž zabraňuje potenciálnímu nebezpečí zcizení nešifrovaných dat či obnově dříve smazaných dat v interní paměti a na SD kartě. K dispozici je tzv. On-Device Encryption, který interní a externí paměť šifruje 256bitovou šifrou AES. Další možností, kterou KNOX poskytuje, je šifrování virtuální privátní sítě pro bezpečnou komunikaci s okolím. Dále nabízí KNOX možnost zařízení s firemním systémem KNOX na dálku spravovat, monitorovat pozici, zablokovat či uvést do továrního nastavení [20] [14].

Samsung KNOX nabízí opravdu dobré řešení bezpečnosti na mobilních zařízeních. Svědčí o tom fakt, že technologie prošla úspěšnou certifikací NSA (The National Security Agency), ale i to, že se Google části KNOX snažil implementovat do Androidu 5.X Lollipop. Mezi velkou nevýhodu patří omezenost tohoto řešení na zařízení vyrobená společností Samsung a operační systém Android. Jelikož ve firmách často zaměstnanci používají různé platformy a zařízení od různých výrobců, je toto řešení velmi omezeno.

2.2.2 IBM Lotus Mobile Connect

Společnost IBM pro firemní zákazníky nabízí řešení zabezpečeného připojení k firemním aplikacím, např. email, kontakty, kalendář a další. Toto řešení nabízí v produktu, které nazývá Lotus Mobile Connect. Technologie poskytuje dvě alternativy: alternativu založenou na plnohodnotném VPN klientovi a alternativu založenou na HTTP protokolu, přičemž obě alternativy by měly používat stejně silné autentizační a šifrovací algoritmy. HTTP alternativa pak používá kryptografické protokoly TLS (Transport Layer Security) a SSL (Secure Sockets Layer). Pro bezpečné navázání spojení pošle tzv. Connection manager vzdá-

lené aplikaci požadavek na přihlašovací údaje uživatele. Aplikace odešle přihlašovací údaje přes zabezpečené připojení HTTP metodou POST. Údaje jsou pak ověřeny a při úspěšném ověření uživatele je vytvořen a poslán token vzdálené aplikaci. Tento token je posléze uložen do cookies vzdálené aplikace (nejčastěji internetový prohlížeč). Token jednoznačně identifikuje daného uživatele pro budoucí požadavky na server [26].

Mezi výhody patří, že tato technologie nabízí řešení nezávislé na mobilní platformě. Bohužel toto řešení poskytuje pouze základní zabezpečenou práci na mobilních zařízeních v rámci firemní sítě. Problém, který toto řešení opomíjí, je fakt, že mobilní zařízení může být napadeno škodlivou aplikací.

2.2.3 Citrix XenMobile

Firma Citrix nabízí produkt XenMobile. Tento produkt zahrnuje základní MDM (Mobile device management) konfiguraci a zabezpečení mobilních zařízení, ale je také prorostlý do vyšších vrstev systému, kde poskytuje spuštění aplikací v režimu sandbox (např. internetový prohlížeč, emailový klient a aplikace pro sdílení dokumentů). Dále také nabízí víceúrovňovou autentizaci a aplikace pro práci a řízení.

První částí je XenMobile Device Manager. Toto řešení umožňuje nastavování rolí uživatelů, konfigurací zařízení, vynucování firemních politik, seznamy povolených resp. zakázaných aplikací na daném zařízení, detekci zásahu či úpravy operačního systému (rooting systém Android, jailbreak systém Apple iOS), wipe odcizených zařízení, geolokaci zařízení, nastavení hesel, konfiguraci VPN a mnoho dalšího. Všechno toto může administrátor spravovat přes webovou konzoli, kde může jednotlivé uživatele monitorovat a přiřazovat jim politiky. Pro přihlášení do různých síťových služeb lze využit centralizované certifikační autority jako například Microsoft Certificate Services či adresářových služeb LDAP [4].

Další částí je balík aplikací Citrix Worx Home. Ten se skládá ze sjednoceného obchodu s aplikacemi, kde jsou dostupné aplikace dle role uživatele, vynucování MDM a MAM (Mobile application management) politik a helpdesk služby. Následující část App Controller slouží jako poskytovatel obsahu pro nativní aplikace (např. uchovávání a distribuci přihlašovacích údajů). Aby bylo možno aplikace různě omezovat či vynucovat politiky, je potřeba, jako v případě Samsung KNOX, je zaobalit bezpečnostní vrstvou. Tento proces spočívá ve vložení kódu pro podporu úloh správy zařízení a kódu pro prosazování firemních politik. Pro vložení tohoto potřebného kódu lze využit dvou variant - vložení před kompilaci aplikace nebo vložení po kompilaci aplikace. Alternativa vložení kódu před kompilaci se používá především pro firemní aplikace, kde vývojáři aplikace vloží metody z Worx App SDK, které povolí App Controller a vynucování politik za běhu aplikace. Možnost vložení kódu po kompilaci je zase využívána pro aplikace třetích stran. Tento obal aplikace, resp. vložený kód, pak může sdílet informace např. o nastavených aplikačních politikách s Citrix Worx Home prostřednictvím tzv. StoreFront, což je sdílené autorizované datové úložiště, kde jsou data uložena perzistentně [4].

Poslední důležitou částí je aplikace ShareFile, která zajišťuje bezpečné ukládaní, sdílení a synchronizaci dat v rámci organizace i mimo. Tato část se skládá ze dvou komponent: Control system a Storage system. Control system je zodpovědný za udržování informací o uživatelských účtech a zprostředkovatelské funkce pro práci nad tímto systémem. Tyto informace jsou zašifrované a uložené v datacentrech firmy Citrix. Storage system je samotné úložiště dat. Zde je možno využít cloudového úložiště Amazon Web Services od společnosti Amazon nebo data ukládat lokálně, ať už zcela nebo částečně [4].

Řešení od společnosti Citrix je částečně podobné jako řešení KNOX od společnosti Sam-

sung, zaobalí aplikace vrstvou a odstíní aplikace od zbytku sytému. Na rozdíl od technologie KNOX však neposkytuje takové zabezpečení do hloubky na úrovni operačního systému či dokonce na hardwarové úrovni. Mezi výhody tohoto řešení patří především, že jej lze nasadit na různých mobilních platformách a na zařízeních od různých výrobců, ne pouze na zařízeních od jednoho výrobce, jako je to u technologie KNOX.

Kapitola 3

Šifrování

Aby bylo možné soubory bezpečně přenášet mezi dvěma čí více body, je třeba soubory šifrovat. V této kapitole jsou popsány různé kryptografické metody, které toto šifrování umožňují, a jejich typické použití.

3.1 Kryptografie

Dle definice z [28] je kryptografie obor zabývající se studiem matematických technik, vztahující se k aspektům bezpečnosti jako jsou: důvěrnost, integrita dat, autentizace entit, ověření pravosti a původu dat. Hlavní cíle kryptografie jsou pak:

- Důvěryhodnost - k datům mají přístup pouze ti, kteří mají autorizovaný přístup.
- Integrita dat - aby data nebyla neoprávněně změněna, resp. zahrnuje schopnost detekovat manipulaci dat neoprávněnou stranou.
- Autentizace - jednotlivé strany vstupující do komunikace mají možnost ověřit svou totožnost navzájem prostřednictvím komunikačního kanálu. Autentizaci lze rozdělit do dvou hlavních tříd: autentizace subjektu a ověřování původu dat, což implicitně zajišťuje integritu dat.
- Nepopiratelnost - jednotlivé entity nemohou popírat své předchozí akce a závazky. Např. pokud subjekty popírají svou vykonanou akci, musí mít třetí strana prostředky k vyřešení tohoto sporu.

3.2 Symetrická kryptografie

Symetrická kryptografie je založená na použití jednoho klíče k zašifrování dat mezi dvěma stranami komunikace. Symetrickou je nazývaná, protože k dešifrování a šifrování používá stejný klíč. Kdokoliv, kdo zná klíč, může dešifrovat resp. šifrovat zašifrovaný text. Klíč tedy musí být uchováván na bezpečném místě. Z toho plyne problém, že pokud budou dvě strany mezi sebou bezpečně komunikovat, je důležité tento klíč sdílet zabezpečenou komunikací. Proto je potřeba komunikaci zabezpečit asymetrickou kryptografii [21]. Mezi výhody symetrické kryptografie oproti asymetrické kryptografii patří především rychlejší šifrování větších dat a u některých zařízení taky možnost hardwarové akcelerace. Jeden z nepoužívanějších algoritmů symetrické kryptografie je algoritmus AES. Tento algoritmus

byl standardizován institutem NIST (National Institute of Standards and Technology). Tento algoritmus vychází z blokové šifry Rijndael, která se liší pouze tím, že nemá pevnou velikost bloku 128 bitů a velikost klíčů pouze 128, 192 a 256 bitů, ale velikost klíče určenou násobkem 32 bitů [21].

3.3 Asymetrická kryptografie

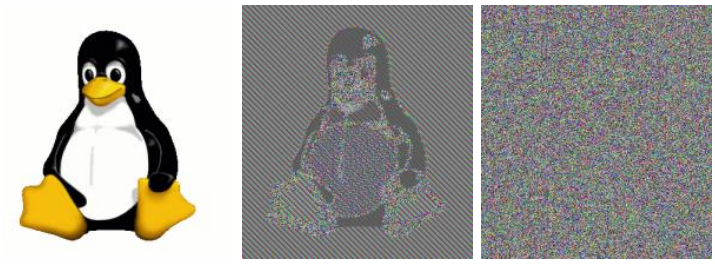
Asymetrická kryptografie používá pár klíčů, obvykle jsou označené jako veřejný klíč a soukromý klíč. V bezpečném systému je operace vypočtení soukromého klíče daného veřejným klíčem výpočetně nemožná. K šifrování slouží veřejný klíč a k dešifrování klíč soukromý. Veřejný klíč není na rozdíl od soukromého klíče uchováván v bezpečné paměti. [28].

Nejčastějším zástupcem je algoritmus RSA. Tento algoritmus je založen na problému známém již 250 let, kdy po vynásobení dvou velkých prvočísel, je tento výsledek obtížné faktorizovat [21]. Popis algoritmu generování klíčů dle [28]:

1. Vygeneruj dvě velká různá náhodná prvočísla p a q, každé zhruba stejné velikosti.
2. Vypočítej $n = pq$ a $\Phi = (p-1)(q-1)$.
3. Vyber náhodné celé číslo e, kde $1 < e < \Phi$ tak, že největší společný dělitel e a Φ je roven 1.
4. Použij rozšířený Eukleidův algoritmus k nalezení d, které je $1 < d < \Phi$, takže platí $ed \equiv 1 \pmod{\Phi}$.
5. Veřejný klíč je pak (n, e); Soukromý klíč je d.

3.4 Blokové šifry a jejich režimy

Blokové šifry mohou být jak šifry symetrické kryptografie, tak šifry asymetrické kryptografie. Bloková šifra funguje tak, že vezme klíč k délky r a zašifruje šifrou E bloky otevřeného textu m o fixní velikosti n tak, že vytvoří výsledné zašifrované bloky $c = E(k, m)$, kde n se nazývá velikost bloku. Někdy však otevřený text nemusí být násobkem velikosti bloku, tento problém řeší tzv. padding, který šifrovaná data zarovná na násobek velikosti bloku [21].



Obrázek 3.1: Nalevo originální obrázek, uprostřed obrázek zašifrovaný v režimu ECB, napravo obrázek zašifrovaný v režimu CBC. Obrázky převzaty z [3].

Každá bloková šifra funguje v nějakém režimu. Nejjednodušším režimem je režim ECB (Electronic Codebook), který šifruje bloky za sebou bez závislosti na předchozím bloku. Tento způsob však vytváří bezpečností riziko, že stejný otevřený text má stejnou zašifrovanou podobu. Dalším problémem, který z této metody vyplývá, je malá velikost bloku.

To se projeví zejména u obrázku, kdy se na zašifrovaném obrázku mohou objevit artefakty připomínající nezašifrovanou podobu obrázku, tento problém je naznačen na obr 3.1. Tyto problémy řeší režim CBC (Cipher Block Chaining), který na daný blok provede operaci *xor* s blokem předchozím předtím, než je blok zašifrován. Aby byl stejný text po každém zašifrování unikátní, je potřeba na první blok provést operaci *xor* s tzv. inicializačním vektorem, který musí být nepředvídatelný [3].

Kapitola 4

Přenos souborů mezi skupinami uživatelů

Potřeba věnovat se problému šifrování souborů pro uživatelskou skupinu vychází především z požadavku velkých společností, které potřebují sdílet soubory mezi různými skupinami uživatelů. Soubor je často ve firmě nebo jiné instituci nezbytné sdílet na úrovni konkrétního oddělení či uvnitř pracovní skupiny tohoto oddělení.

Při řešení tohoto požadavku se však vyskytne mnoho problémů. Prvním velkým problémem je distribuce šifrovacích klíčů v rámci uživatelské skupiny. Problémem je především to, že počet členů uživatelské skupiny nemusí být vždy pevný, resp. do skupiny mohou přibýt noví členové nebo být odebráni členové stávající. Toto implikuje problém častého generování či přegenerovaní šifrovacích klíčů a různé strategie generovaní klíčů se tento problém snaží eliminovat. Dalším čistě praktickým problémem je, že zařízení nemusí být vždy připojeno k internetové síti, a tak se změny v uživatelské skupině nepropagují okamžitě.

4.1 Skupiny uživatelů definované na serveru

Základem tohoto konceptu je myšlenka definovat uživatelskou skupinu na centralizovaném serveru. To znamená, že v databázi na serveru jsou uloženy všechny informace o uživatelské skupině, tj. identifikátory členů skupiny, samotný identifikátor skupiny a klíče skupiny. Pro každou skupinu je vygenerován pár klíčů, veřejný a soukromý klíč. Každému členu skupiny je vygenerován unikátní identifikátor. Tento identifikátor může být generován funkcí pro odvození klíče tzv. KDF (Key Derivation Function), která identifikátor odvodí z uživatelského hesla, který je uložen v bezpečné paměti zařízení, jež konkrétní uživatel používá. Tento identifikátor slouží k autentizaci na serveru, který autentizuje dle identifikátoru požadavek. Samotný soubor je šifrován dalším klíčem symetrické kryptografie, tj. klíčem souboru. Šifrování souboru funguje následovně: veřejný klíč skupiny je zaslán na koncová zařízení, kde je použit pro zašifrování klíče souboru, který má být přístupný dané uživatelské skupině. Dešifrování posléze pracuje tak, že klient, který daný soubor dešifruje, zašle na server klíč souboru zašifrovaný veřejným klíčem skupiny a identifikátor uživatele. Server dle identifikátoru autentizuje uživatele a podle toho, jestli patří uživatel do skupiny, pošle klientovi zpět dešifrovaný klíč souboru. Tímto klíčem je následně soubor dešifrován.

Důležité je si uvědomit, že soukromý klíč skupiny je vždy pouze na serveru a nikdy není distribuován ke klientům. Je také potřeba zmínit, že toto řešení spoléhá na prerekvizitu bezpečného spojení mezi klientem a serverem, které lze implementovat pomocí SSL protokolu.

4.1.1 Případová studie

1. Alice chce poskytnout soubor pouze konkrétnímu oddělení. Požádá server o veřejný klíč skupiny a identifikátor skupiny pro dané oddělení.

2. Alice vygeneruje klíč symetrické kryptografie, tj. klíč souboru, a tímto klíčem zašifruje obsah souboru. Klíč souboru je dále zašifrován veřejným klíčem skupiny, který Alice získala ze serveru.

3. Zašifrovaný klíč souboru a identifikátor skupiny pak vloží do hlavičky souboru.

4. Bob, který patří do skupiny, pro niž byl soubor určen, chce daný soubor dešifrovat. Zašle svůj identifikátor, identifikátor skupiny a zašifrovaný klíč souboru z hlavičky souboru na server.

5. Server autentizuje požadavek pomocí Bobova identifikátoru. Vyhledá v databázi podle identifikátoru skupiny informace o skupině. Zjistí, zda Bob má členství v dané skupině a dle toho rozšifruje klíč souboru soukromým klíčem skupiny. Pokud je Bob člen skupiny, je poslán dešifrovaný klíč souboru Bobovi.

6. Bob dešifruje klíčem souboru obsah souboru.

4.1.2 Zhodnocení

Tento koncept umožňuje velice snadno přidávat/odebírat uživatele ze skupiny, není třeba žádná složitá distribuce skupinových klíčů, stačí pouze přidat záznam do databáze na serveru a změny ve skupině se takřka ihned propagují. Toto řešení má za důsledek to, že nově přidaný člen má přístup ke starším zašifrovaným souborům pro danou skupinu oproti některým jiným řešením. Další výhodou je to, že při každém dešifrování souboru se musí zaslat požadavek na server o dešifrovaní klíče souboru. Lze tedy zaznamenávat, kdo a kdy otevřel daný soubor. Tato funkcionalita pak může sloužit k tomu, že lze dohledat, jestli člen skupiny soubor otevřel. Pokud ne, lze člena odebrat ze skupiny, aniž by se musel soubor znovu zašifrovat.

Mezi nevýhody tohoto konceptu patří nutnost připojení k síti při dešifrování souboru. Tento problém lze částečně řešit ukládáním dešifrovaných klíčů souboru do bezpečné paměti, tj. nejlépe do správce klíčů na daném zařízení. Poté není potřeba při opětovném otevření souboru připojení k internetu. Další nevýhodou jsou vyšší požadavky na server, kde je potřeba databáze pro uložení informací o skupinách a velký výkon při vysokém počtu požadavků na server.

4.2 Skupiny uživatelů definované v souboru

Toto řešení spočívá v tom, že členové skupiny jsou předem definováni v hlavičce souboru. Každý uživatel má pár klíčů, veřejný a soukromý klíč. Veřejné klíče a identifikátory uživatelů jsou pak uloženy na serveru, který slouží čistě k distribuci těchto klíčů a identifikátorů.

Soukromé klíče jsou uloženy ve správci klíčů na zařízení, kde je daný uživatel přihlášen. Při zašifrování souboru je potřeba získat veřejné klíče a identifikátory uživatelů ze serveru, pro které chceme daný soubor zpřístupnit. Poté je potřeba vygenerovat klíč souboru, který je klíčem symetrické kryptografie a tímto klíčem zašifrovat obsah souboru. Posléze je pro každého člena skupiny jeho veřejným klíčem zašifrován klíč souboru. Takto zašifrovaný klíč souboru je následně vložen do hlavičky souboru a je k němu přidán patřičný identifikátor uživatele. V hlavičce souboru je uloženo tolik zašifrovaných klíčů, kolik je členů skupiny. Identifikátor uživatele je zde proto, aby bylo možné v hlavičce souboru rychle najít správný klíč dle uživatele. Není pak třeba složitých mechanismů pro testování klíčů.

Pro distribuci veřejných klíčů, identifikátorů uživatelů a soukromých klíčů je třeba zabezpečeného spojení mezi zařízením a serverem.

4.2.1 Případová studie

1. Alice chce poslat soubor Bobovi a Evě. Požádá server o veřejné klíče a identifikátory Boba a Evy.

2. Alice vygeneruje klíč symetrické kryptografie, tj. klíč souboru, a tímto klíčem zašifruje obsah souboru. Klíč souboru je dále zašifrován veřejným klíčem Boba, následně je také klíč souboru zašifrován veřejným klíčem Evy.

3. Klíč souboru zašifrovaný veřejným klíčem Boba je vložen do hlavičky souboru, kde je k němu přidán identifikátor Boba. Totéž je analogicky provedeno pro zašifrovaný klíč souboru veřejným klíčem Evy.

4. Soubor je zaslán Bobovi a Evě.

5. Bob chce dešifrovat soubor. Nalezne dle svého identifikátoru správný zašifrovaný klíč souboru v hlavičce souboru. Tento klíč dešifruje svým soukromým klíčem a tím získá klíč souboru.

6. Klíčem souboru dešifruje obsah souboru.

7. Totéž jako v případě Boba platí analogicky pro Evu.

4.2.2 Zhodnocení

Výhodou řešení je absence serveru pro správu uživatelských skupin, uživatel se tedy nemusí při šifrování souboru připojovat na server, ale pouze stáhnout veřejné klíče členů skupiny. Pokud má tyto veřejné klíče už uloženy v zařízení, není se již potřeba při šifrování připojovat k internetu vůbec. Rovněž samotná tvorba uživatelských skupin není složitá, nejsou potřeba žádné skupinové klíče. Dešifrování funguje zcela bez připojení k internetu. Toto řešení je výhodné především pro malé uživatelské skupiny.

Nevýhodou tohoto konceptu je fakt, že při velkém počtu členů skupiny poroste přímou úměrností velikost hlavičky souboru. Další velkou nevýhodou je problematické přidávání nových členů do skupiny. Pro přidání nového člena je potřeba znát klíč souboru a ten zašifrovat veřejným klíčem nového člena a přidat do hlavičky souboru. I kdyby se tento problém vyřešil, pořád zůstává problém, že kopie souboru nebudou korespondovat s aktuálním stavem skupiny. Další problém je, pokud si některý z uživatelů vygeneruje nový pár klíčů (například kvůli kompromitaci stávajícího páru). Ostatním uživatelům se následně musí distribuovat nový veřejný klíč tohoto uživatele.

4.3 Dvojí klíč skupiny (dočasný a perzistentní)

Myšlenka tohoto konceptu spočívá v tom, že uživatelská skupina je definovaná dvěma druhy klíčů: dočasnými a perzistentními. Perzistentní klíč (dále označen zkratkou PKS - perzistentní klíč skupiny) je produktem symetrické kryptografie. Zde je potřeba dát pozor na known-plaintext attack, protože útočník může znát jak zašifrovaný tak dešifrovaný řetězec. Avšak dnešní moderní symetrické šifry jako AES jsou proti tomuto útoku rezistentní [9]. Tento klíč se nedistribuuje koncovým zařízením, zůstává vždy na serveru. Dalším klíčem je dočasný klíč skupiny resp. pár klíčů, které jsou založené na asymetrické kryptografii tj. veřejný (dále označen zkratkou DVKS - dočasný veřejný klíč skupiny) a soukromý klíč (dále označen zkratkou DSKS - dočasný soukromý klíč skupiny). DVKS slouží k zašifrování klíče souboru a DSKS k jeho dešifrování. Při šifrování souboru je potřeba získat ze serveru DSKS zašifrovaný klíčem PKS. Tento zašifrovaný fragment je pak vložen do hlavičky souboru. Dále je potřeba získat ze serveru DVKS, kterým je zašifrován klíč souboru, ten je také vložen do hlavičky souboru. Dešifrování souboru pak funguje tak, že pokud je v zařízení uložen v bezpečné paměti správný DSKS, je klíč souboru dešifrován DSKS klíčem. Pokud není nalezen správný klíč, je na server poslaná část hlavičky souboru (tj. DSKS, zašifrovaný klíčem PKS). Server rozšifruje zaslanou část hlavičky souboru klíčem PKS, tím získá DSKS, následně autentizuje požadavek uživatele a pokud je uživatel členem skupiny, je serverem zaslán na zařízení klíč DSKS. Klíčem DSKS se rozšifruje klíč souboru a DSKS se uloží do bezpečné paměti v zařízení pro další použití.

Tak jako u předcházejících konceptů i zde je potřeba mezi serverem a zařízením zabezpečené spojení. Kompletní řešení zabezpečeného spojení mezi klientem a serverem a autentizace uživatele je vysvětleno v části 6.1.6.

4.3.1 Případová studie

1. Alice chce zpřístupnit soubor uživatelské skupině, ve které je Bob. Alice získá ze serveru aktuální DVKS a DSKS zašifrovaný klíčem PKS, ty pak vloží do hlavičky souboru.

2. Alice vygeneruje klíč symetrické kryptografie tj. klíč souboru a tímto klíčem zašifruje obsah souboru.

3. Klíč souboru zašifruje klíčem DVKS a vloží do hlavičky souboru.

4. Soubor je zaslán k Bobovi.

5. Pokud má Bob správný DSKS, použije jej na dešifrování klíče souboru, jinak zašle část hlavičky souboru (tj. DSKS zašifrovaný klíčem PSK). Server autentizuje Bobův požadavek a zašle zpět Bobovi DSKS. Tímto Bob dešifruje klíč souboru.

6. Klíč souboru pak dešifruje obsah souboru.

4.3.2 Zhodnocení

Velkou výhodou tohoto konceptu je možnost generovat dočasné klíče skupiny. Doba pro generování nového klíče může být různá a to i v případě, kdy dochází k přegenerování klíče při každém šifrování souboru. To se zejména hodí, pokud bychom chtěli zaznamenávat, kdo a kdy daný soubor otevřel, protože dobou k přegenerování dočasného klíče lze určovat, jak často se zařízení budou při dešifrování připojovat na server a jakou mírou budou závislé na

síti. Zde je vidět, že koncept je celkem dobře škálovatelný. Současně doba k přegenerovaní dočasných klíčů nemá vliv na velikost databáze na serveru, jelikož veškerá historie dočasných soukromých klíčů skupiny je nesena v hlavičkách souborů a expirované veřejné klíče nemá smysl uchovávat. Další výhodou je, že nový člen skupiny může mít přístup ke starším souborům.

Nevýhodou toho konceptu mohou být vyšší nároky na server, kde vzniká potřeba ukládat informace o uživatelských skupinách. Další nevýhodou může být fakt, že všechny klíče na mobilním zařízení musí být uloženy do bezpečného úložiště zařízení.

Kapitola 5

Přenos souborů mimo organizaci

Dalším velkým problémem je distribuce citlivých souborů mimo instituci. Tento problém nastává, když zaměstnanec jedné firmy chce zaslat citlivý soubor zaměstnanci jiné firmy. Největším problémem je zde to, že obě firmy mohou fungovat na odlišných systémech šifrování souboru, nebo firma, která je příjemcem souboru, nemusí používat žádný systém šifrování svých souborů. K distribuci takových souborů, které je potřeba přenést jednorázově mezi dvěma společnostmi či společností a jednotlivcem, dochází nejčastěji emailovou korespondencí. V této práci je rozebráno zejména řešení tohoto problému z pohledu mobilních platforem, i když je potřeba tento problém řešit nejen na mobilních platformách.

Při řešení těchto problémů je potřeba vyřešit minimálně tyto otázky:

- Jak bude realizována bezpečná linka, přes kterou se budou distribuovat šifrovací klíče? Tato otázka je zásadní pro bezpečnou komunikaci mezi dvěma subjekty. Jako potenciálně zabezpečená linka může sloužit mobilní telefonní síť (ve formě SMS nebo telefonního hovoru), zabezpečená VOIP služba či protokol SSL.
- Je toto řešení z pohledu uživatele jednoduché a komfortní? Ačkoliv se tato otázka zdá být nedůležitá, v případě, kdy je řešení hlavně pro odesílající stranu komfortní, má uživatel mnohem větší motivaci používat zabezpečenou komunikaci. To implikuje menší selhání lidského faktoru, kdy např. uživatelé zasílají hesla k souborům nezabezpečenou komunikací.
- Na základě čeho lze příjemce souboru autentizovat? Autentizace uživatele mimo firmu lze řešit několika možnostmi: autentizace pomocí zaslání SMS, registrace uživatele resp. ověření emailem nebo ověření uživatele telefonicky.
- Jaké budou technické požadavky na systém? Poslední otázkou zůstává, jak moc toto řešení bude technicky náročné. Např. zda bude potřeba server s databází, či bude potřeba SMS brána atd.

5.1 Ověřování uživatele přes SMS zprávu

Tento koncept stojí na možnosti ověřovat uživatele mimo firmu pomocí SMS. Odesílatel souboru (uživatel uvnitř firmy) je před zašifrováním požádán o telefonní číslo příjemce (uživatel mimo firmu), soubor je zašifrován symetrickou kryptografií a vytvořen klíč souboru. Dále je vytvořen další klíč, který je vytvořen pomocí funkce na odvození klíče. Vstupem do této funkce je telefonní číslo příjemce a náhodně vygenerovaná kryptografická sůl resp. řetězec.

Tímto klíčem je symetrickou kryptografií zašifrován klíč souboru. Takto zašifrovaný klíč souboru a kryptografická sůl jsou zaslány pomocí SMS zprávy příjemci. Když je zašifrovaný soubor distribuován příjemci např. přes email, musí si příjemce stáhnout speciální aplikaci. Poté stačí, když příjemce otevře zašifrovaný soubor v zařízení a aplikace si okamžitě asociuje soubor a prohledá v telefonu SMS zprávy. Pokud nalezne správnou zprávu, vytvoří klíč pomocí funkce na odvození klíče, kde vstupem je kryptografická sůl z SMS zprávy a získané telefonní číslo pomocí API funkce operačního systému telefonu. Tímto klíčem je dešifrován klíč souboru z SMS zprávy. Klíčem souboru je pak dešifrován obsah souboru a je zobrazen příjemci.

5.1.1 Případová studie

1. Alice chce ve firmě X bezpečně poslat soubor Bobovi ve firmě Y. Alice vytvoří klíč souboru a zašifruje soubor symetrickou kryptografií.

2. Dále vezme telefonní číslo Boba a náhodě vygenerovanou kryptografickou sůl a vytvoří pomocí funkce na odvození klíče nový klíč. Tímto klíčem pomocí symetrické kryptografie zašifruje klíč souboru.

3. Zašifrovaný klíč souboru a kryptografickou sůl pošle Bobovi přes SMS zprávu. Zašifrovaný soubor zašle pomocí emailu.

4. Bob vezme číslo svého telefonu a kryptografickou sůl a vytvoří klíč. Tímto klíčem rozšifruje klíč souboru z SMS zprávy zaslané Alicí.

5. Klíčem souboru pak dešifruje obsah souboru.

5.1.2 Zhodnocení

Řešení není technicky náročné, nevyžaduje žádný centrální server. Jak pro odesílatele, tak pro příjemce je uživatelsky komfortní. Z hlediska bezpečnosti je řešení relativně bezpečné. Pokud by příjemce chtěl soubor sdílet s někým třetím a zaslal mu SMS zprávu se zašifrovaným klíčem, třetí strana soubor pomocí aplikace neotevře. V případě cíleného útoku však není problém údaje podvrhnout. Další zajímavou možnosti je to, že ověření uživatele umožňuje, aby aplikace zkontrolovala, jestli je správným příjemcem, a případně ho upozornila, že se odesílatel pravděpodobně spletl nebo nejlépe zaslala notifikaci na server, který upozorní odesílatele.

Velkou nevýhodou tohoto řešení je, že v praxi nelze vždy spolehlivě zjistit pomocí API operačního systému telefonní číslo ze SIM karty, jelikož někteří operátoři neukládají telefonní číslo na SIM kartu, např. O2 CZ. Dalším faktem je, že tento koncept opomíjí zařízení jako tablety bez 3G modulu.

5.2 Ověřování uživatele telefonicky

Další možností, jak ověřovat uživatele mimo firmu, je ověření pomocí prosté telefonní komunikace. Program vygeneruje PIN kód, tento kód a náhodně vygenerovaná kryptografická sůl jsou vstupem do funkce na odvození klíče, která vygeneruje klíč, jímž je zašifrován obsah souboru. Kryptografická sůl je posléze přidána do hlavičky souboru. Odesílatel pak pošle příjemci soubor a telefonicky kontaktuje příjemce a prozradí mu PIN kód. Příjemce si stáhne aplikaci do telefonu a zadá PIN kód. Obsah souboru je následně dešifrován.

5.2.1 Případová studie

1. Alice chce ve firmě X zaslat soubor Bobovi ve firmě Y. Alice vygeneruje PIN kód, ten je společně s kryptografickou solí vstupem do funkce pro odvození klíče. Tato funkce vygeneruje klíč souboru, kterým je zašifrován obsah souboru.

2. Alice zašle soubor emailem Bobovi a zároveň přes telefonní spojení Bobovi prozradí PIN kód.

3. Bob vezme kryptografickou sůl z hlavičky souboru a PIN kód vloží jako vstup do funkce pro odvození klíče.

4. Tato funkce vygeneruje klíč souboru, tímto klíčem dešifruje obsah souboru.

5.2.2 Zhodnocení

Hlavní výhodnou toho řešení je snadná implementace a malá technická náročnost. Mezi velké nevýhody tohoto konceptu je to, že spoléhá na autentizaci uživatele jiným uživatelem přes telefonní hovor. Jak pro příjemce tak odesílatele je toto řešení silně nekomfortní. Také samotný telefonní hovor může být odposloucháván třetí stranou.

5.3 Ověření uživatele před zasláním souboru

Jádrem tohoto konceptu je registrace uživatele mimo firmu ještě před zašifrováním souboru. Nejprve si příjemce stáhne aplikaci do mobilního telefonu. Aplikace příjemci vytvoří pár klíčů asymetrické kryptografie tj. veřejný a soukromý. Soukromý klíč je uložen do bezpečné paměti nejlépe do správce klíčů. Veřejný klíč a telefonní číslo zařízení se zašlou na server a jsou uloženy do databáze nefiremních uživatelů. Server při registraci nefiremního uživatele vygeneruje identifikátor a zašle ho na telefonní číslo SMS zprávou. Aplikace v zařízení na straně příjemce na tu to SMS zprávu zareaguje a uloží si identifikátor do paměti. Odesílatel při šifrování souboru vybere uživatele ze seznamu nefiremních uživatelů dle telefonního čísla. Následně aplikace na straně odesílatele vytvoří symetrickou kryptografií klíč souboru a zašifruje jeho obsah. Klíč souboru dále zašifruje veřejným klíčem příjemce. Celý tento fragment se následně zašifruje klíčem vytvořeným pomocí funkce na odvození klíče, jejímž vstupem bude identifikátor uživatele a náhodně vygenerovaná kryptografická sůl. Aplikace pak na straně příjemce použije identifikátor z SMS zprávy a kryptografickou sůl z hlavičky souboru a vytvoří klíč pomocí funkce na odvození klíče. Tímto klíčem dešifruje hlavičku souboru, kdy získá klíč souboru zašifrovaný veřejným klíčem příjemce. Tento fragment pak dešifruje soukromým klíčem příjemce a získá klíč souboru. Klíčem souboru pak dešifruje obsah souboru.

5.3.1 Případová studie

1. Bob chce ve firmě Y od Alice ve firmě X získat soubor. Vygeneruje si pár klíčů asymetrické kryptografie. Veřejný klíč a své telefonní číslo zašle na registrační server.

2. Registrační server si uloží Bobovy údaje a vygeneruje pro Boba identifikátor, který zašle Bobovi na jeho telefonní číslo.

3. Bob přijme SMS a uloží si svůj identifikátor.

4. Alice zašle na registrační server požadavek s Bobovým telefonním číslem. Server dle telefonního čísla najde Boba v databázi a zašle Alici Bobův veřejný klíč a identifikátor Boba.

5. Alice vytvoří symetrickou kryptografií klíč souboru. Tento klíč zašifruje Bobovým veřejným klíčem. Dále ten to fragment zašifruje klíčem získaným z funkce na odvození klíče, jejíž vstupem je náhodně vygenerovaná kryptografická sůl a Bobův identifikátor. Výsledek posléze vloží do hlavičky souboru.

6. Zašifrovaný soubor pošle Bobovi přes email.

7. Bob vygeneruje klíč pomocí funkce na odvození klíčů, jejíž vstupem je kryptografická sůl z hlavičky souboru a identifikátor z SMS. Tímto dešifruje fragment hlavičky souboru a získá zašifrovaný klíč souboru jeho veřejným klíčem. Ten dešifruje svým soukromým klíčem.

8. Klíčem souboru dešifruje obsah souboru.

5.3.2 Zhodnocení

Výhodou tohoto řešení je především to, že soubor lze otevřít pouze na zařízení, kde byl soukromý klíč příjemce vygenerován. (pokud však soukromý klíč neunikne k útočníkovi). Další výhodou je, že řešení provádí ověření příjemcova telefonního čísla, tudíž lze příjemce snadno autentizovat podle jeho telefonního čísla. To se obzvláště hodí, pokud se odesílatel a příjemce neznají, odesílatel tak může proklamovanou identitu příjemce ověřit dle telefonního čísla dostupného např. na webových stránkách či v databázi kontaktů. Jak pro odesílatele, tak příjemce je tento koncept z uživatelského hlediska relativně komfortní.

Řešení je vlastně variací řešení ověřování uživatele přes SMS, z toho plyne, že zde nastává stejný praktický problém a to získávání příjemcova telefonního čísla z API operačního systému. Zde však není problém nechat příjemce telefonní číslo vyplnit, jelikož je pak ověřeno SMS zprávou. Možnou další nevýhodou jsou větší technické nároky na systém.

Kapitola 6

Implementace

K implementaci navrženého byl zvolen koncept: Skupiny uživatelů definované na serveru s dvojicí klíčů skupiny (dočasný a perzistentní). Důvodem, proč zvolit tento koncept, byla především jeho široká škálovatelnost, výhoda správy uživatelských účtů na serveru a ukládání historie skupinových klíčů do souboru, z něhož plyne možnost častého přegenerování skupinového klíče.

Pro implementaci byla zvolena platforma Google Android verze 4.3 a výše. Důvod, proč byla zvolena právě tato verze operačního systému, je popsán v následující kapitole. Hlavním argumentem, proč se zabývat platformou Google Android, je to, že v současné době je nejrozšířenějším mobilním operačním systémem s podílem přibližně 81 % v roce 2014 [13].

6.1 Implementace klientské části

6.1.1 Bezpečné úložiště klíčů

Už od verze 1.6 má operační systém Android k dispozici úložiště klíčů přístupné přes Android Keystore Provider pro ukládání VPN a WiFi přihlašovacích údajů. Bohužel toto úložiště je součástí privátního API a nemohou ho využívat aplikace třetích stran. Ve verzi 4.0 bylo představeno KeyChain API [8], toto API umožňuje vývojářům aplikací třetích stran ukládání klíčů a certifikátu do bezpečného úložiště. Avšak k tomuto úložišti vlastní přístup všechny ostatní nainstalované aplikace a systém Android poskytuje ochranu zvenčí. Použití tohoto úložiště by útočníkovi nezabránilo pomocí jeho nainstalované aplikace získat klíče a certifikáty ostatních nainstalovaných aplikací. Proto byl ve verzi 4.3 zpřístupněn Android Keystore Provider [2] pro aplikace třetích stran. Do tohoto úložiště klíčů lze ukládat pouze RSA 2048 bitové klíče. Tyto klíče pak systém zpřístupňuje pouze aplikaci, která do úložiště klíče vložila. Přesněji, každý klíč je vázán na UID aplikace, které je přiděleno při instalaci aplikace systémem. Toto UID je unikátní a pouze aplikace podepsaná stejným vývojářským certifikátem může sdílet toto UID a tedy i mít přístup k stejnému úložišti klíčů [16] [24].

Další věcí, kterou je potřeba zmínit, je to, že s příchodem verze 4.3 je Android Keystore implementován hardwarově, pokud to daný hardware zařízení podporuje. To, jestli zařízení podporuje hardwarovou implementaci Keystoru, lze zjistit buď v menu Nastavení - položka Zabezpečení nebo voláním API funkce [7]. Celá implementace stojí na použití technologie TrustZone v ARM procesoru. Aplikace přistupují ke Keystoru, ke kterému se přistupuje v zabezpečeném běhovém prostředí, které je separováno od operačního sytému. To zajišťuje vyšší bezpečnost a to i v případě infikování operačního systému, příp. rootu zařízení [22] [1].

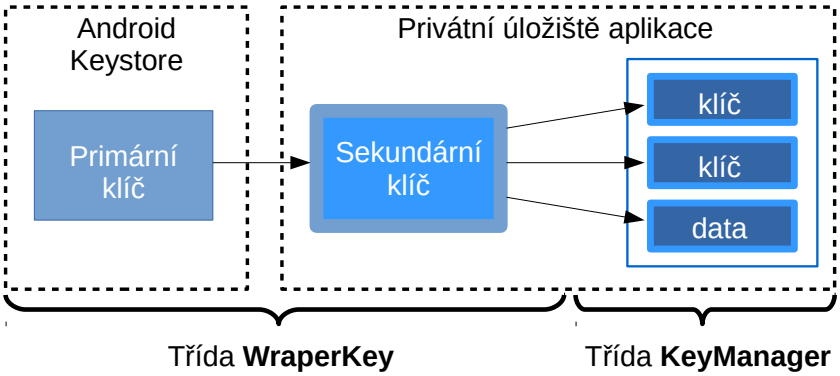
Pro verze 4.3 a nižší existuje možnost jak obejít systém a zpřístupnit tak Android

Keystore. Řešení spočívá v tom, že samotný Keystore je ve starších verzích operačního systému Android implementován pomocí Linuxové služby (daemon), která komunikuje na lokálním rozhraní, se kterým lze pomocí socketů komunikovat. Je třeba zmínit, že se jedná pouze o experimentální řešení a není doporučeno ho používat v produkční verzi aplikací [23].

6.1.2 Implementace vlastního správce klíčů

Android Keystore Provider umožňuje ukládat pouze RSA 2048 bitové klíče, z toho plyne omezení, že šifrovaná data tímto klíčem nemohou být vetší než 2048 bitů. Dalším problémem je, že bude potřeba ukládat do bezpečné paměti AES klíče a fragmenty z hlavičky souboru. Z tohoto důvodu je nutné implementovat vlastního správce klíčů. Základní princip bude spočívat v tzv. wrap key tj. zašifrování klíče jiný klíčem. Jako primární klíč resp. pár klíčů označíme RSA 2048 bitový klíč, který uložíme do uložiště klíčů spravované operačním systémem, tj. Android Keystore. Další AES klíč o velikosti 256 bitů označíme jako sekundární klíč. Sekundární klíč je pak zašifrován veřejným primárním klíčem a takto zašifrovaný sekundární klíč se ukládá do souboru v privátním úložišti aplikace, zakódován Base64 kódováním. Sekundárním klíčem se šifrují ostatní klíče a jelikož se jedná o klíč symetrické kryptografie AES, nejsme omezeni velikostí klíčů. Šifrování ostatních klíčů bude provedeno AES-256 šifrou konkrétně v režimu CBC.

Výše uvedený postup implementuje třída `WrapperKey`, která při své konstrukci načte primární klíč z Android Keystore. Pokud klíč neexistuje, vygeneruje primární klíč a uloží ho do Android Keystore. Analogický postup je proveden i při generování sekundárního klíče s tím rozdílem, že v případě ukládání se zašifruje sekundární klíč veřejným primárním klíčem a naopak při nahrávaní klíče ze souboru dešifruje sekundární klíč soukromým primárním klíčem. Třída `WrapperKey` obsahuje metody wrap/unwrap a wrapEntry/unwrapEntry, kde první dvojice metod slouží k šifrování resp. dešifrování klíčů a druhá dvojice k šifrování resp. dešifrování obecných. Rozdíl mezi těmito dvojicemi je především jejich výsledek, kde první dvojice sestaví z dešifrovaných dat přímo objekt daného typu klíče.



Obrázek 6.1: Schéma ukládání klíčů.

Samotný správce klíčů pak implementuje třída `KeyManager`, která ukládá klíče do souboru v privátní paměti aplikace. Celé toto řešení je postaveno na třídě `Properties` ze standardní knihovny, tato třída implementuje hash tabulku, která dokáže její prvky exportovat či načíst ze souboru, z tohoto důvodu je vhodná pro tento účel. Klíče jsou v tabulce ukládány dle jména klíče, formát záznamu hodnoty v tabulce je pak naznačen na obr. 6.2. Na obrázku je vidět, že některá data jsou kódovaná Base64, a to především z důvodu dekódování, aby se v datech nevyskytoval separátor označený na obrázku prázdným obdélníkem. Typ klíče je pak pouze výčtový typ resp. celočíselná hodnota, proto není potřeba ho nikterak kódovat. Zkratkou IV je pak označen inicializační vektor pro šifru AES-256 v modu CBC, kterou je zašifrován daný klíč.

Obrázek 6.2: Formát záznamu klíče v souboru.

Třída `KeyManager` obsahuje metody `getKey/setKey`, `getEntry/setEntry` a `clean`. Pár metod `getKey/setKey` slouží k získávání resp. ukládaní klíčů. Na rozdíl od následující dvojice metod, `setKey` zjistí typ klíče dle typu objektu klíče předaného parametrem metody a `getKey` pak vytvoří požadovaný typ instance klíče. Poslední zajímavou metodou je metoda `clean`, která smaže soubory, v nichž jsou uloženy klíče a vygeneruje nový privátní klíč, který uloží do Android Keystore.

Smazané soubory však mohou ležet v paměti dál a operační systém Android nenabízí žádné API pro bezpečné smazání souborů z paměti telefonu. Bohužel ani samotné přepsání souboru nezajistí bezpečné smazání souboru, protože dnešní mobilní telefony mají flash paměť, která podporuje technologii Wear leveling. Tato technologie zvyšuje životnost paměti tak, že se snaží nezapisovat do stále stejných bloků paměti [27]. Avšak v tomto případě není zranitelnost nikterak kritická, jelikož data jsou zašifrována a primární klíč je přegenerován. Útočník by se tedy k nezašifrovaným datům neměl dostat.

6.1.3 Implementace správce skupin

Další částí implementace je správce skupin. Tato část je implementovaná pomocí tříd `Group` a `GroupManager`. Třída `Group` slouží pouze k uchovávání dat o uživatelských skupinách. Její metody `getPublicKey/setPublicKey` a `getPrivateKey/setPrivateKey` mají za úkol získaní a uložení dočasného veřejného klíče skupiny resp. dočasného soukromého klíče skupiny. Následující metody `getFragment/setFragment` jsou určeny k získání a uložení dat, která budou později vložena do hlavičky souboru. Všechny tyto metody pracují se třídou `KeyManager`, odkud se data na základě jména skupiny buď získávají či ukládají. Poslední metodou je `getAlias`, která vrací jméno skupiny.

Třída `GroupManager` implementuje návrhový vzor Jedináček, tedy lze vždy získat pouze jednu instanci této třídy. `GroupManager` zajišťuje ukládání a získávaní skupin do/z paměti. Tato funkcionalita je implementována způsobem, že třída `GroupManager` používá tzv. `SharedPreferences` [18] pro ukládání názvů uživatelských skupin. Třída pak obsahuje kontejner instancí třídy `Group`, který při konstrukci naplní instancemi třídy `Group`,

které vytvoří předání jména uživatelské skupiny jako parametr konstruktoru. Metoda `updateOrCreateGroup` třídy `GroupManager` pak vytvoří nebo aktualizuje záznam o uživatelské skupině a k tomu, aby zjistila, jestli daná uživatelská skupina existuje, použije metodu `isExist`, která na základě jména uživatelské skupiny prohledá kontejner skupiny téže třídy. Další metodou `getGroup` je možno na základě jména uživatelské skupiny předaného parametrem získat konkrétní instanci skupiny. Poslední zajímanou metodou je `clean`, která smaže všechny záznamy z `SharedPreferences`.

6.1.4 Třída `SymetricCipher`

Implementace využívá šifrovacích algoritmů ze standardní knihovny `Crypto` [11] jazyka Java. Obal nad touto knihovnou poskytují třídy `SymetricCipher` a `AsymetricCipher`. Jak je z názvu třídy `SymetricCipher` patrné, je tato třída určena pro symetrickou kryptografii. Konkrétně používá algoritmus AES s velikostí klíče 256 bitů v režimu šifrování CBC a zarovnáváním `PKCS5Padding`.

Třída `SymetricCipher` vlastní metodu `generateKey` ve dvou provedeních pro generování AES-256 klíče. První varianta bez parametrů vygeneruje nový klíč z kryptograficky bezpečného pseudonáhodného generátoru čísel. Tento generátor je implementován standardní třídou `SecureRandom` z jazyka Java. Zde se však může vyskytnout problém s kompatibilitou různých verzí operačního systémů Android. Při výchozím nastavení před verzí Android 4.2 byla interně použitá jiná implementace, konkrétně z knihovny `Crypto`. V novějších verzích sytému Android je použitá implementace z knihovny OpenSSL, tudíž při použití metody `setSeed` se stejným parametrem se mohou vygenerovat dva různé výstupy. Tento problém lze vyřešit explicitním nastavením použité knihovny v generátoru. Více o této problematice se lze dočíst z [25]. Avšak dle [15] je metoda `setSeed` využitelná pouze k testovacím účelům a pro produkční verzi aplikace je doporučeno spolehnout se na systém, který vygeneruje dostatečně nepředvídatelnou inicializaci generátoru. Druhou variantou metody `generateKey` je předání uživatelského hesla a kryptografické soli. Tato metoda použije pro odvození klíče uživatelské heslo a kryptografickou sůl. Konkrétně se použije standardní funkce na odvozování klíčů PBKDF2 [19], která na základě těchto parametrů dokáže bezpečně klíč odvodit. Krom těchto parametrů je třeba předat ještě parametr o množství iterací, se kterým bude tato operace odvozování hesla prováděná. Tento parametr slouží jako ochrana proti útoku hrubou silou, kdy při dostatečném množství iterací lze útočníkovi podstatně zvýšit časovou náročnost na uhádnutí hesla. Dle standardu RFC [19] je doporučeno nechat tento parametr minimálně na 1000 iterací. Omezení na počet iteraci není, avšak při vysokých hodnotách se ztrácí uživatelský komfort, jelikož uživatel je nucen čekat delší dobu na přihlášení do aplikace. Funkce PBKDF2 je implementovaná ve standardní třídě `SecretKeyFactory` z knihovny Crypto. Konkrétně pro generování klíčů v metodě `generateKey` je použit algoritmus `PBKDF2WithHmacSHA1` a nastavení na 4096 iterací.

Další sada metod `dercypt/encrypt` slouží k šifrování resp. dešifrování dat pomocí symetrické kryptografie. Zde jsou dvě varianty těchto metod, varianta šifrování/dešifrování bajtového pole nebo varianta šifrování/dešifrování souborů. Obě varianty se liší především svou implementací. Varianta pracující ze souborem používá standardní třídy `CipherOutputStream` a `CipherInputStream`. Jelikož operace čtení/zápis do souboru a šifrování/dešifrování mohou mít různou rychlost, je potřeba vstup či výstup do souboru obalit třídami `BufferedInputStream` a `BufferedOutputStream`. Toto řešení pomáhá částečně se vyrovnat s tímto problémem tak, aby se jednotlivé operace navzájem nebrzdily.

Pro generování inicializačního vektoru pak slouží metoda `generateIV`, která má jako parametr klíč. Avšak inicializační vektor není generován v závislosti na klíči, klíč je použit pouze jako parametr do třídy `Cipher`, která generuje inicializační vektor implicitně nezávisle na klíči. Poslední metoda je `decodeKey`, která slouží k sestavení objektu klíče z binárních dat.

6.1.5 Třída `AsymetricCipher`

Tato třída poskytuje rozhraní nad algoritmy asymetrického šifrování z knihovny `Crypto`. Použitý šifrovací algoritmus je RSA s 1024bitovými klíči v režimu ECB se zarovnáním `PKCS1Padding`.

Pro generování soukromého a veřejného klíče slouží metoda `generateKey`. Další metodou `generateWrappingKey` je generován primární RSA 2048bitový klíč a zároveň je uložen do Android Keystore. Metody `decodePublicKey` a `decodePrivateKey` mají za úkol sestavit objekty veřejného a privátního klíče z binárních dat. Další důležitá dvojice metod je `decrypt` a `encrypt`, kde metoda `encrypt` používá k zašifrování dat veřejný klíč a metoda `decrypt` soukromý klíč k dešifrování dat.

6.1.6 Zabezpečení komunikace mezi klientem a serverem

Pro zabezpečení komunikace slouží klíč uživatele, který je se sestaven pomocí metody `generateKey` třídy `SymetricCipher`. Této metodě se předá vygenerovaná kryptografická sůl a uživatelovo heslo. Metoda vygeneruje AES 256bitový klíč tj. klíč uživatele. Pokud klient chce poslat na server citlivá data, zašifruje tato data klíčem uživatele. Server poté sestaví tento klíč, kde heslo uživatele získá z databáze uživatelů a data tímto klíčem dešifruje. Stejný postup platí pro zaslání citlivých dat ze serveru na klienta. Takto zabezpečená komunikace zajišťuje, že data dešifruje pouze strana, která zná uživatelské heslo.

6.1.7 Komunikace se serverovou části

Síťovou komunikaci obstarává třída `Network`, která implementuje komunikaci HTTP pomocí standardní třídy `HttpURLConnection`. Komunikace se serverem funguje metodou POST. Metoda `prepareData` serializuje parametry předané metodu `addParam`. Komunikace se zahájí metodou `sendRequest`, která interně volá metodu `request`, která naváže komunikaci a zašle na server parametry dotazu. Poté se již pouze čeká na odpověď ze strany serveru, který zašle výsledky ve formátu JSON. Výsledky jsou deserializovány ve formě hash mapy, která je vrácena jako návratová hodnota metody `sendRequest`.

Řízení komunikace implementuje třída `Server`. Některé metody z třídy `Server` jsou asynchronní, resp. při jejich zavolání se vytvoří a spustí nové vlákno, ve kterém se začne vykonávat daná funkcionalita. Zde byla použita knihovna `Guava` vyvíjená společností Google. Tato knihovna obsahuje třídu `ListeningExecutorService`, při jejíž instanciaci se parametrem předá nově vytvořené vlákno pomocí třídy `Executors` [5]. Parametrem lze předat i sérii vytvořených vláken a vytvořit tzv. pool vláken, která budou danou operaci vykonávat paralelně. Z instance třídy `ListeningExecutorService` se zavolá metoda `submit`, jejíž parametr je třída `Callable<T>`, kde `T` je typ návratové hodnoty vracené vykonávanou operací. V této třídě je nutné přepsat metodu `call`, která se zavolá při spuštění vlákna. Instance třídy `ListeningExecutorService` navrátí objekt typu `ListenableFuture<T>`. Zde jsou dvě možnosti jak s tímto objektem pracovat. První možnost je zavolání metody `get`,

v tomto případě se po zavolání této metody program zastaví a vyčkává na dokončení operace v metodě `call`. Po jejím ukončení vrací návratovou hodnotu operace. Další možností je vytvoření tzv. callbacku, který se vykoná po dokončení operace. Zde lze vytvořit dva typy callbacku: `onSuccess` zavolaný při úspěšném dokončení operace a `onFailure` zavolaný při neúspěšném dokončení operace. Více o této problematice se lze dočíst na [10].

Přihlašování uživatele implementuje metoda `loginAsync`, která je asynchronní. Nejprve je nutné vygenerovat klíč uživatele pomocí metody `generateKey` třídy `SymetricCipher`. Tímto klíčem se zašifruje login uživatele a spolu s nezašifrovaným loginem, inicializačním vektorem a kryptografickou solí, z níž byl uživatelský klíč sestaven, je zaslán na server. Server pak sestaví uživatelský klíč a dešifruje zašifrovaný login uživatele. Tento login je potom porovnán s nezašifrovaným loginem, pokud jsou oba loginy shodné, server vrátí klientovi kladnou odpověď. Tento mechanismus umožní sdělit uživateli, jestli zadal správné uživatelské údaje. Jestliže uživatel zadal přihlašovací údaje správně, uloží metoda `loginAsync` uživatelský klíč do správce klíčů pomocí metody `setKey` třídy `KeyManager`.

Další potřebnou akcí je synchronizace klíčů. Tuto funkci obstarává metoda `synchronizeKeysAsync`, která je rovněž jako předchozí metoda asynchronní. Na server pošle požadavek, kde předá parametry: login a kryptografické soli uživatelského klíče. Server pak vrátí všechny veřejné dočasné klíče skupin, dočasné soukromé klíče skupiny zašifrované perzistentním klíčem skupiny a u skupin, kde má uživatel členství, nešifrované dočasné soukromé klíče skupiny. Celý tento výstup je zašifrován uživatelským klíčem. Metoda `synchronizeKeysAsync` tento výstup dešifruje a uloží všechny tyto klíče do správce klíčů.

Poslední důležitou metodou je metoda `getKeyFromFragmentSync`. Tato metoda slouží k dešifrování dočasného soukromého klíče skupiny z hlavičky souboru. Toto se uplatní zejména v případě, kdy uživatel pracuje se starým souborem, resp. aktuální dočasný soukromý klíč skupiny je jiný než při šifrování tohoto souboru. Metoda předá na server požadavek s parametry: login, jméno uživatelské skupiny, pro kterou byl soubor zašifrován a zašifrovaný dočasný soukromý klíč skupiny. Server ověří členství uživatele ve skupině a v případě úspěchu vrátí dočasný soukromý klíč skupiny zašifrovaný uživatelským klíčem. Celý jednoduchý aplikační protokol je naznačen v tabulce 6.1.

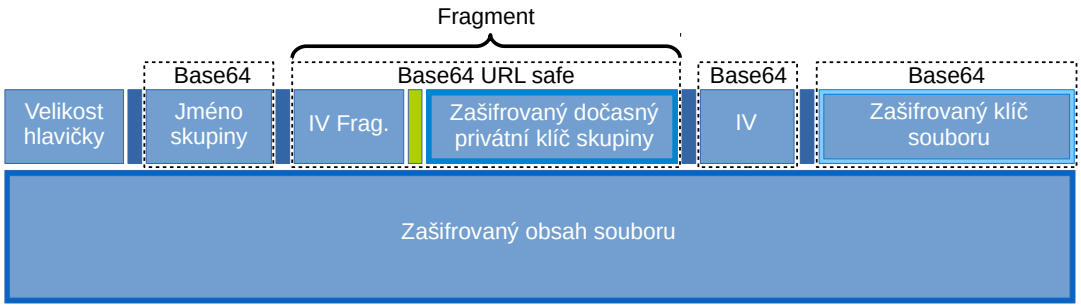
Název akce	**Parametry**	**Návratové hodnoty ze serveru**
Přihlášení uživatele	Login, zašifrovaný login, inicializační vektor a kryptografická sůl	Při úspěšném přihlášení vrací hodnotu true.
Synchronizace klíčů	Login, kryptografická sůl	Pro skupiny, kde má daný uživatel členství, vrátí server jak dočasný soukromý klíč, tak dočasný veřejný klíč skupiny, jinak vrací pouze dočasný veřejný klíč skupiny.
Dešifrování dočasného soukromého klíče skupiny z hlavičky souboru	Název skupiny, klíč z hlavičky souboru a kryptografická sůl	Vrací dešifrovaný dočasný soukromý klíč skupiny.

Tabulka 6.1: Ukázka protokolu.

6.1.8 Práce se soubory

Samotnou práci ze soubory zajišťuje třída `FileManager`. Tato třída má především na starosti správné dekódování hlavičky souboru a její správné sestavení.

O sestavení hlavičky souboru a správné zašifrování souboru se stará metoda `encryptFile`. Metoda `encryptFile` pracuje asynchronně. Jejím parametrem je URI resp. cesta k souboru, který má být zašifrován. Nejprve zavolá metodu `openFile` třídy `FileManager`, která na základě URI souboru otevře patřičně soubor. Pokud je například soubor sdílen přes HTTP, protokol naváže HTTP komunikaci. Posléze vytvoří nový soubor ve složce `EncryptionData` na SD kartě mobilního zařízení. Do tohoto souboru se budou zapisovat zašifrovaná data. Poté je potřeba sestavit hlavičku souboru. Tento proces probíhá tak, že se nejprve vygeneruje klíč souboru a inicializační vektor pomocí třídy `SymetricCipher`. Poté vypočítá celkovou velikost hlavičky, zapíše tuto velikost do souboru a oddělí separátorem. Pak zapíše do souboru název skupiny, pro kterou je daný soubor sdílen a rovněž oddělí separátorem. Dále je zapsán zašifrovaný dočasný soukromý klíč dané skupiny, který je označen jako fragment. Nakonec jsou zapsány inicializační vektor a klíč souboru zašifrovaný veřejným dočasným klíčem skupiny, kde obě tyto části jsou navíc kódovány Base64 pro jednodušší pozdější zpracování. Obsah souboru je šifrován metodou `encrypt` třídy `SymetricCipher`. Formát souboru je naznačen na obr. 6.3, kde modré prázdné obdélníky jsou znázorněny separátory hlavičky.



Obrázek 6.3: Formát zašifrovaného souboru.

Dekódování hlavičky a dešifrování souboru zajišťuje metoda `decryptFile`, která je asynchronní. Dešifrovaný soubor je uložen v privátním úložišti aplikace. Soubor je zpřístupněn ostatním aplikacím pomocí třídy PrivateStorageProvider. Metoda `createFile` třídy `PrivateStorageProvider` vytvoří soubor v privátním úložišti aplikace. Další metoda `generateURI` vygeneruje URI odkazující se na soubor v privátním úložišti. Poslední důležitá metoda `clean` vymaže všechny dešifrované soubory z privátního úložiště.

Tento přístup má velkou výhodu v tom, že je možno kontrolovat, jaká aplikace má přístup k dešifrovanému souboru. To umožňuje definovat filtr aplikací, který nepovolí přístup potenciálně nebezpečným aplikacím. Nevýhodou tohoto řešení je omezení velikostí této paměti, tudíž pro velké soubory řádově stovky megabajtů nebude toto řešení fungovat.

Samotné dekódování hlavičky a dešifrování zašifrovaného souboru funguje tak, že parametrem metody `decryptFile` je předáno URI referující na zašifrovaný soubor. Nejprve metoda `decryptFile` vytvoří soubor v privátním úložišti aplikace a smaže všechny předchozí

dešifrované soubory v privátním úložišti pomocí metody `createFile`. Poté je z hlavičky zašifrovaného souboru přečtena informace o celkové velikosti hlavičky. Následně je načtena celá hlavička a předána do objektu standardní třídy `Scanner`. Tato třída má za úkol rozdělit jednotlivé části hlavičky oddělené separátorem. Po zpracování hlavičky souboru je dle jména skupiny z hlavičky vyhledán dočasný soukromý klíč skupiny pomocí metody `getGroup` třídy `GroupManager`. Pokud není nalezen tento klíč, je na server zaslána část z hlavičky, označena jako fragment na obr 6.3 pomocí metody `getKeyFromFragmentSync` třídy `Server`, jejíž návratová hodnota je starý dočasný soukromý klíč skupiny získaný z fragmentu (pokud daný uživatel má členství v skupině). Tímto klíčem je pak dešifrován klíč souboru. Klíč souboru je společně s inicializačním vektorem předán metodě `decrypt` třídy `SymetricCipher`, která dešifruje zbytek souboru.

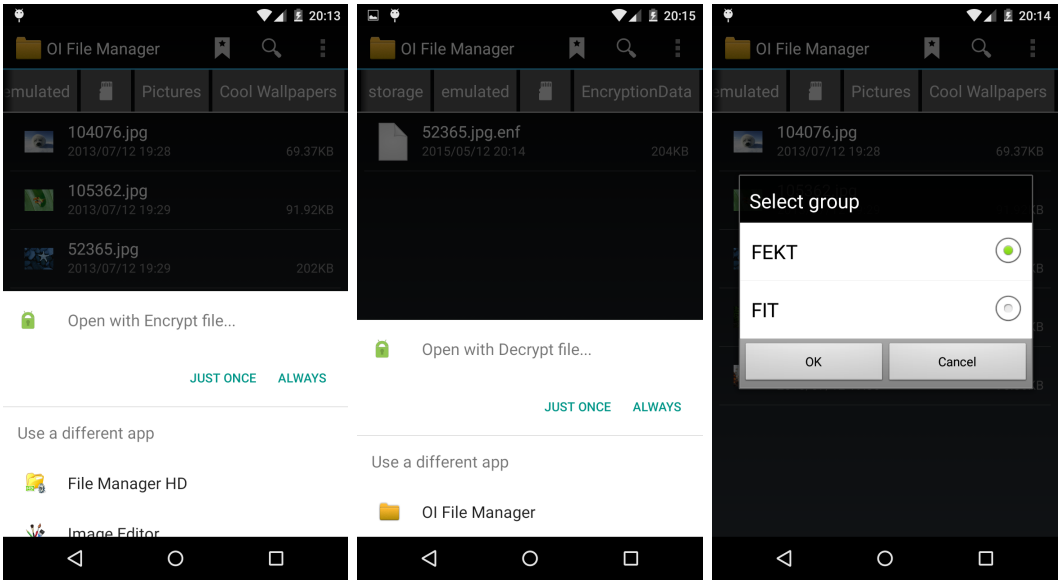
6.1.9 Implementace uživatelských akcí

Pro lepší uživatelský komfort byla aplikace implementovaná pomocí uživatelských akcí. Akcí se myslí děj, kdy uživatel či systém vyvolá nějakou interakci, na kterou jsou aplikace schopné zareagovat, např. kliknutí na soubor. Tato akce ve formě tzv. intentu (česky záměru) notifikuje všechny citlivé tzv. intent filtry, které mají dané aplikace zaregistrované. Tyto jednotlivé intent filtry mají zaregistrované tzv. aktivity, které na jednotlivé intenty reagují vykonáním dané akce [6]. Aplikace implementuje dvě aktivity `EncryptAction` a `DecryptAction`, obě citlivé na akci `ACTION_VIEW`. Tato akce se vyvolá v případě, že uživatel klikne na jakýkoliv soubor či odkaz na soubor. Aktivity `EncryptAction` a `DecryptAction` mají rozdílně citlivý intent filtr, zatímco `EncryptAction` má intent filtr nastavený soubory typu `MIME */*` resp. na všechny typy souboru, intent filtr aktivity `DecryptAction` je nastaven na přípony souboru `*.enf`. Tento způsob implementace má výhodu, že šifrování a dešifrování souboru umožní používat souborové manažery třetích stran. Tedy, na rozdíl od některých jiných konkurenčních řešení, uživatel může používat aplikaci pro správu souborů, na kterou je zvyklý a není nucen šifrovat a dešifrovat soubory přes souborový manažer šifrovací aplikace. Na obr 6.4 vlevo a uprostřed jsou zobrazeny uživatelské akce šifrování a dešifrování.

Aktivita `EncryptAction` implementuje uživatelskou akci šifrování. Nejprve je uživateli zobrazen dialog s výběrem uživatelské skupiny, pro kterou bude soubor zašifrován, viz obr. 6.4 vpravo. Poté se provede synchronizace klíčů pomocí metody `synchronizeKeysAsync` třídy `Server`, pokud je uživatel připojený k internetu. To proto, aby byl klíč zašifrován aktuálním skupinovým klíčem. Následně je soubor zašifrován pomocí metody `encryptFile` třídy `FileManager` a uložen do složky `EncryptionData` na SD kartě zařízení.

Dešifrování souboru probíhá pomocí aktivity `DecryptAction`, která rovněž provede nejprve synchronizaci klíčů a poté zavolá metodu `decryptFile` třídy `FileManager`. Po dešifrování souboru je soubor uložen v privátním úložišti aplikace. Následně je vytvořen intent pro zobrazení seznamu aplikací, které daný typ dešifrovaného souboru mohou zobrazit. Zde je důležité vyfiltrovat ze seznamů aplikací aktivitu `EncryptAction`.

Všechny výše zmíněné aktivity pracují ve vlákně, které obsluhuje uživatelské prostředí. Avšak při různých operacích asynchronních metod se spouštějí další vlákna. Aby vlákno vykonávající danou operaci mohlo předat informaci o stavu operace, je použito mechanismu zasílání zpráv do vlákna obsluhujícího uživatelské rozhraní tak, aby stav mohl být zobrazen uživateli.



Obrázek 6.4: Ukázky uživatelských akcí z aplikace.

6.1.10 Implementace rozhraní pro přihlášení uživatele

Poslední dvě aktivity `LoginActivity` a `MainActivity` implementují přihlašovací a odhlašovací formulář. Přihlašování uživatele probíhá tak, že uživatel vyplní jméno a heslo v polích v `LoginActivity`. Ta zavolá metodu `loginAsync` třídy `Server` a pokud tato metoda vrátí pozitivní hodnotu, zobrazí aktivitu `MainActivity`, jinak vyvolá chybové hlášení. Komplementární aktivita `MainActivity` implementuje odhlášení uživatele tím, že vymaže všechny klíče ze správce souboru a vymaže dešifrované soubory v privátním úložišti aplikace.

6.2 Implementace serverové části

Serverová část je implementovaná v jazyce PHP ve verzi 5.4.38. Databázová část je řešena MySQL databázi.

6.2.1 Server

Serverová část aplikace využívá implementaci kryptografických algoritmů z opensource knihoven: phpseclib pro algoritmus AES a část knihovny PHP-CryptLib pro PBKDF2 funkci na odvození klíče. Parametry algoritmu jsou stejné jako u klientské části aplikace.

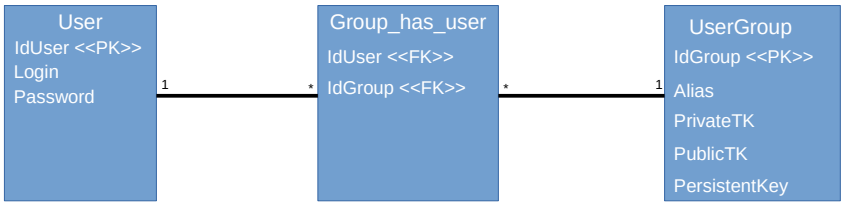
Všechny důležité metody a funkce jsou implementovány ve třídě `User` a ve skriptu `functions.php`. Jednou ze zajímavých funkcí je `GetUserKey`, která vygeneruje uživatelský klíč na základě hesla a kryptografické soli pomocí funkce PBKDF2. Další funkce `EncryptData` a `DecryptData` šifrují a dešifrují data prostřednictvím třídy `Crypt_AES` z knihovny phpseclib. Poslední zajímavou funkcí je `GenerateIV`, která vygeneruje inicializační vektor za použití standardní funkce `mcrypt_create_iv`. Třída `User` implementuje operace, se kterými pracují skripty na obsluhu komunikace. Metoda `checkPassword` třídy `User` kontroluje správnost uživatelského hesla tak, že nejprve načte heslo z databáze a sestaví z něho uživatelský klíč. Tímto klíčem dešifruje zašifrovaný login předaný parametrem a porovná ho s nezašifrovaným loginem uživatele. Pokud se oba loginy shodují, je heslo

správné. Další metoda `getGroupsAllKeys` vrací všechny přístupné skupinové klíče pro daného uživatele, tedy všechny dočasné veřejné skupinové klíče a dočasné soukromé skupinové klíče zašifrované perzistentním klíčem skupiny. Dále také vrací nezašifrované soukromé skupinové klíče patřící skupinám, ve kterých má daný uživatel členství. Všechny tyto klíče jsou serializovány do formátu JSON a zašifrovány uživatelským klíčem. K dešifrování dočasného soukromého klíče skupiny zašifrovaného perzistentním klíčem skupiny slouží metoda `getPrivateKeyFromFragment`. Tato metoda nejprve ověří členství uživatele v dané skupině a následně získá perzistentní klíč skupiny. Tímto klíčem dešifruje dočasný soukromý klíč skupiny z fragmentu hlavičky souboru a zašifruje ho uživatelským klíčem.

O všechny dotazovací operace nad databází se starají třídy `MySQL_User` a `MySQL_Group`, které obsahují jednotlivé dotazy v jazyce SQL. Všechny SQL dotazy jsou implementovány přes tzv. prepared statements, které dle [12] zajišťují ochranu proti případnému SQL injection útoku. Skripty `login.php`, `synckeys.php` a `decrypt.php` obsluhují komunikaci s klientem. Skript `login.php` ověřuje správnost uživatelských údajů, skript `synckeys.php` slouží pro distribuci klíčů a skript `decrypt.php` dešifruje dočasný soukromý klíč skupiny z fragmentu hlavičky souboru.

6.2.2 Databáze

Jak již bylo uvedeno, databázová část běží na MySQL serveru, konkrétně na InnoDB enginu. Databáze se skládá ze tří entit resp. tabulek. Tabulka `User` ukládá informace o uživatelích, zejména login, který musí být unikátní a uživatelské heslo. Tabulka `UserGroup` vlastní data o uživatelských skupinách. Sloupec `Alias` reprezentuje jméno skupiny, `PrivateTK` dočasný soukromý klíč skupiny, `PublicTK` dočasný veřejný klíč skupiny a `PersistentKey` perzistentní klíč skupiny. Všechny uvedené sloupce musí podléhat integritnímu omezení `Unique`. Návrh celé relační databáze vidíte na obr 6.5. Je třeba uvést, že data v databázi nejsou šifrovaná, což je bezpečnostní riziko, avšak zde je vytvořena databáze pouze k testovacím účelům a ověření funkčnosti celého konceptu. Ideální by bylo, pokud by klíč skupiny nebyl uložen v `UserGroup`, ale byl by zvlášť v tabulce pro každého uživatele, přičemž pro každého uživatele by byl zašifrován jeho klíčem. V případě, že se útočník zvenčí nějakým způsobem (např. z důvodu bezpečnostní chyby v MySQL) dostane ke všem datům, nebyl schopen zjistit klíče skupin.



Obrázek 6.5: Návrh relační databáze.

Kapitola 7

Ověření funkčnosti

7.1 Případová studie

Pro ověření základní správnosti a funkčnosti konceptu použijeme následující případovou studii: Alice pracující pro oddělení X chce zaslat soubor Bobovi, který pracuje pro téže oddělení a oddělení Y. Eva pracující pro oddělení Y chce zaslat soubor Bobovi. Alice i Eva by se rády dostaly k souborům pro oddělení, kde nemají členství.

7.2 Postup ověření správnosti a funkčnosti

1. Nejprve se přihlásíme do aplikace jako Alice. Poté otevřeme správce souborů a zašifrujeme soubor pro oddělení X.

2. Posléze se z aplikace odhlásíme a přihlásíme se jako uživatel Eva. Otevřeme správce souborů a zobrazíme složku EncryptionData. V této složce je soubor zašifrovaný pro Boba. Zkusíme daný soubor dešifrovat a otevřít. Tato akce je však zamítnuta a aplikace vypíše hlášení o zamítnutí přístupu k souboru.

3. Dále zašifrujeme soubor pro Boba a odhlásíme se z aplikace.

4. Přihlásíme se do aplikace jako Bob. Otevřeme správce souborů ve složce EncrytpionData a dešifrujeme soubor od Alice. Soubor je dešifrován a zobrazen. Totéž provedeme pro soubor od Evy se stejným výsledkem.

5. Odhlásíme se z aplikace a přihlásíme se jako Alice. Otevřeme správce souborů ve složce EncrytpionData, zkusíme dešifrovat a otevřít soubor od Evy pro Boba. Tato akce je však zamítnuta a aplikace vypíše hlášení o zamítnutí přístupu k souboru.

Takto uvedený postup ověřuje základní správnost a funkčnost celého konceptu. Spravovat uživatele, skupiny a klíče je možno přihlášením do phpMyAdmin administrace databáze.

Kapitola 8

Závěr

V této práci je analyzováno a zhodnoceno současné řešení přenosu souborů mezi mobilními zařízeními. Další analýza se věnuje současným komplexním řešením bezpečností na mobilním operačním systému Android, kde jsou podány argumenty, proč je dané řešení výhodné či nevýhodné a porovnání s existující konkurencí.

Dále je v této práci letmý úvod do pojmů z oblastí kryptografie tak, aby i čtenář nepohybující se v oblasti bezpečnosti a kryptografie rozuměl všem problémům, řešením a bezpečnostním rizikům vyskytujícím se v tomto textu. Současně byly v rámci této práce vytvořeny koncepty přenosu šifrovaných souborů mezi skupinami uživatelů a koncepty přenosu souborů mimo firmu či instituci se zaměřením na mobilní zařízení, avšak tyto koncepty lze vztahovat obecně na všechny typy zařízení. Všechny koncepty byly důkladně zhodnoceny a nastíněny jejich bezpečnostní slabiny a přednosti.

Z konceptů šifrovaných souborů mezi skupinami uživatelů je po zhodnocení vybrán koncept s nejlepšími parametry, konkrétně koncept s názvem: Dvojí klíč skupiny (dočasný a perzistentní). Tento koncept je implementován ve formě aplikace pro operační systém Android 4.3. Na této implementaci jsou představeny problémy a jejich řešení, které se při implementaci tohoto konceptu vyskytly, zejména problém a řešení bezpečného ukládání klíčů v operačním systému Android či problém synchronizace klíčů mezi klientem a severem. Celá tato implementace ověřila, že lze vyvinout relativně bezpečnou platformu pro přenos souborů na operačním systému Android. Aby bylo řešení opravdu bezpečné, je potřeba se orientovat na novější verze systému Android, nejlépe 4.3 a vyšší. Rovněž je důležité, aby zařízení mělo hardwarovou podporu TrustZone-enabled procesoru, aby mohla být použita hardwarová implementace správce klíčů, kterou nabízí systém Android.

Pro otestování a ověření správnosti celého konceptu byl implementován i testovací server včetně databáze, kde se nachází testovací data. Správnost konceptu je ověřena testovací případovou studii. Celá implementace by šla rozšířit hlavně z hlediska serverové implementace, kde by bylo možné vytvořit propracovaný informační systém pro správu celého systému. Taktéž nad testovací databází, ve které jsou data v otevřené podobě z důvodu testování, by bylo možné vybudovat systém správy uživatelů, který by přiděloval práva jednotlivým uživatelům pouze nad určitými částmi databáze a data v databázi šifroval.

Protože je koncept dobře škálovatelný a variabilní, co se týče hlediska generování a přidělovaní skupinových klíčů, bylo by možné empiricky ověřit různé strategie generování a přidělovaní skupinových klíčů a určit, která strategie má nejlepší parametry pro daný systém.

Literatura

[1] Android 4.3 APIs: Security [online]. 2013 [cit. 2015-5-9]. Dostupné z: http://developer.android.com/about/versions/android-4.3.html#Security.

[2] Android Keystore System [online]. 2013 [cit. 2015-5-1]. Dostupné z: http://developer.android.com/training/articles/keystore.html.

[3] Block cipher mode of operation In: *Wikipedia: the free encyclopedia* [online]. Last modified on 5.4.2015 [cit. 2015-5-13]. Dostupné z: http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation.

[4] Citrix XenMobile technology overview [online]. 2014 [cit. 2015-2-1]. Dostupné z: http://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/citrix-xenmobile-technology-overview.pdf?accessmode=direct.

[5] Class Executors [online]. [cit. 2015-5-11]. Dostupné z: http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Executors.html.

[6] Intents and Intent Filters [online]. [cit. 2015-5-12]. Dostupné z: http://developer.android.com/guide/components/intents-filters.html.

[7] KeyChain: isBoundKeyAlgorithm [online]. 2013 [cit. 2015-5-8]. Dostupné z: http://developer.android.com/reference/android/security/KeyChain.html#isBoundKeyAlgorithm(java.lang.String).

[8] KeyChain [online]. 2013 [cit. 2015-5-1]. Dostupné z: http://developer.android.com/reference/android/security/KeyChain.html.

[9] Known-plaintext attack In: *Wikipedia: the free encyclopedia* [online]. Last modified on 5.4.2015 [cit. 2015-5-18]. Dostupné z: http://en.wikipedia.org/wiki/Known-plaintext_attack.

[10] ListenableFutureExplained [online]. [cit. 2015-5-11]. Dostupné z: https://code.google.com/p/guava-libraries/wiki/ListenableFutureExplained.

[11] Package javax.crypto [online]. [cit. 2015-5-11]. Dostupné z: http://docs.oracle.com/javase/7/docs/api/javax/crypto/package-summary.html.

[12] Prepared statements and stored procedures [online]. [cit. 2015-5-13]. Dostupné z: http://php.net/manual/en/pdo.prepared-statements.php.

[13] Press Release: *International Data Corporation* [online]. 2015 [cit. 2015-5-9]. Dostupné z: http://www.idc.com/getdoc.jsp?containerId=prUS25450615.

[14] Samsung KNOX [online]. 2015 [cit. 2015-1-15]. Dostupné z: https://www.samsungknox.com.

[15] SecureRandom: Seeding SecureRandom may be insecure [online]. [cit. 2015-5-11]. Dostupné z: http://developer.android.com/reference/java/security/SecureRandom.html#insecure_seed.

[16] System Permissions: User IDs and File Access [online]. 2013 [cit. 2015-5-1]. Dostupné z: http://developer.android.com/guide/topics/security/permissions.html#userid.

[17] Technical Overview In: *Boxcryptor*[online]. 2015 [cit. 2015-1-20]. Dostupné z: https://www.boxcryptor.com/en/technical-overview.

[18] Storage Options: Using Shared Preferences [online]. [cit. 2015-5-11]. Dostupné z: http://developer.android.com/guide/topics/data/data-storage.html#pref, [cit. 2015-5-11].

[19] B.Kalavski: Password-Based Cryptography Specification Version 2.0 In: *The Internet Engineering Task Force*[online]. 2000 [cit. 2015-5-11]. Dostupné z: http://tools.ietf.org/html/rfc2898.

[20] Chroust, M.: Samsung KNOX bezpečnost především In: *Mobilmania.cz* [online]. 2014 [cit. 2015-1-14]. Dostupné z: http://samsungmania.mobilmania.cz/clanky/samsung-knox-bezpecnost-predevsim-recenze/uvod-bezpecnost-androidu-popis-knoxu/sc-309-a-1325928-ch-1061166#articleStart.

[21] DELFS, H. a. H. K.: *Introduction to cryptography.* New York: Springer, druhé vydání, 2007, ISBN 3540492437, s. 11 – 41.

[22] Elenkov, N.: Credential storage enhancements in Android 4.3[online]. 2013 [cit. 2015-5-8]. Dostupné z: http://nelenkov.blogspot.cz/2013/08/credential-storage-enhancements-android-43.html.

[23] Elenkov, N.: Storing application secrets in Android's credential storage [online]. 2012 [cit. 2015-5-9]. Dostupné z: http://nelenkov.blogspot.fr/2012/05/storing-application-secrets-in-androids.html.

[24] Elenkov, N.: *Android security internals.* San Francisco: No Starch Press, Inc., první vydání, 2015, ISBN 978-1-59327-581-5, s. 173 – 178.

[25] Johns, T.: Using Cryptography to Store Credentials Safely In: *Android Developers Blog* [online]. 2013 [cit. 2015-5-10]. Dostupné z: http://android-developers.blogspot.cz/2013/02/using-cryptography-to-store-credentials.html.

[26] Kari, J.: Enabling secure, remote access to IBM Lotus iNotes using IBM Lotus Mobile Connect In: *IBM Developerworks* [online]. 2008 [cit. 2015-1-20]. Dostupné z: `http://www.ibm.com/developerworks/lotus/library/notes-mobile-connect/index.html?S_TACT=105AGX13&S_CMP=EDU`.

[27] Lofgren, K.; Norman, R.; Thelin, G.; aj.: Wear leveling techniques for flash EEPROM systems In: *Google Patent Search* [online]. 2005 [cit. 2015-5-9]. Dostupné z: `http://www.google.com/patents/US6850443`, uS Patent 6,850,443.

[28] MENEZES, A. J.: *Handbook of applied cryptography.* Boca Raton: CRC Press, vyd. 1. vydání, 1997, ISBN 0-8493-8523-7, s. 4 – 286.

Příloha A

Obsah CD

Obsah přiloženého elektronického média:

- `doc/` - technická zpráva, pdf verze a zdrojové soubory LaTeX.
- `lic/` - seznam licencí použitých knihoven.
- `src/` - zdrojové soubory implementace.
  - `app/` - zdrojové soubory Android aplikace
  - `server/` - zdrojové soubory testovacího serveru
  - `database/` - SQL skript pro vytvoření MySQL databáze a vytvoření testovacích dat.
  - `man/` - uživatelská příručka včetně postupu, jak zprovoznit testovací server a databázi.
- `bin/` - přeložená aplikace.