Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Webovské rozhranie pre bezpečné zdieľanie dokumentov v cloude

bakalárska práca

2015

Peter Kovács

Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Webovské rozhranie pre bezpečné zdieľanie dokumentov v cloude

bakalárska práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Michal Rjaško, PhD.

Bratislava, 2015

Peter Kovács

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov:

Cieľ:

Literatúra:

Kľúčové slová:

Vedúci:
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania:

Dátum schválenia: doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie:

Abstrakt

Slovenský abstrakt 100-500 slov

Kľúčové slová: jedno, druhé, tretie (prípadne štvrté, piate)

Abstract

English abstract

Keywords:

Obsah

Zoznam obrázkov

Úvod

Úvod je prvou komplexnou informáciou o práci, jej cieli, obsahu a štruktúre. Úvod sa vzťahuje na spracovanú tému konkrétne, obsahuje stručný a výstižný opis problematiky, charakterizuje stav poznania alebo praxe v oblasti, ktorá je predmetom školského diela a oboznamuje s významom, cieľmi a zámermi školského diela. Autor v úvode zdôrazňuje, prečo je práca dôležitá a prečo sa rozhodol spracovať danú tému. Úvod ako názov kapitoly sa nečísluje a jeho rozsah je spravidla 1 až 2 strany.

Kapitola 1

Základné pojmy a označenia

Prvá kapitola slúži ako teoretický úvod do kryptografie. Vymenujeme jej ciele, zadefinujeme základné pojmy, vysvetlíme, čo je symetrické a asymetrické šifrovanie a ako fungujú. Budeme vychádzať z knihy Handbook of Applied Cryptography [1].

1.1 Kryptografia

Kryptografia sa zaoberá metódami ukladania a prenášania dát vo forme, ktorú dokáže spracovať iba taká entita, ktorej sú dáta určené.

1.2 Množiny a operácie

Zadefinujeme si základné množiny a operácie nad nimi, ktoré budeme používať.

- Množinu $\mathcal{A}$ budeme nazývať abecedou. Abecedou je napríklad slovenská abeceda alebo $\mathcal{A} = \{0, 1\}$ je binárnou abecedou.
- Množina $\mathcal{M}$ je množina všetkých možných správ nad danou abecedou $\mathcal{A}$. Napríklad nad abecedou $\mathcal{A} = \{0, 1\}$ pri správach maximálnej dĺžky 2 je $\mathcal{M} = \{00, 01, 10, 11\}$.

- Množina $\mathcal{C}$ obsahuje všetky šifrované správy nad danou abecedou $\mathcal{A}$.
- Množinu $\mathcal{K}$ nazveme množina kľúčov. Prvok $k \in \mathcal{K}$ nazveme kľúč.
- Každý prvok $e \in \mathcal{K}$ jednoznačne určuje bijekciu z $\mathcal{M}$ do $\mathcal{C}$. Túto transformáciu budeme značiť E_e a budeme ju nazývať šifrovacou funkciou.
- Nech D_d je bijektívna transformácia z $\mathcal{C}$ do $\mathcal{M}$ pomocou prvku $d \in \mathcal{K}$, potom D_d nazveme dešifrovacou funkciou.
- Keď aplikujeme transformáciu E_e na správu $m \in \mathcal{M}$ budeme hovoriť, že šifrujeme správu m. Ak aplikujeme D_d na $c \in \mathcal{C}$ budeme hovoriť o dešifrovaní.
- Šifra alebo aj šifrovacia schéma sa skladá z množiny $\{E_e : e \in K\}$ a množiny $\{D_d : d \in K\}$, kde platí, že pre kazdé $e \in K$ existuje $d \in K$ také, že $D_d = E_e^{-1}$ a teda platí aj $D_d(E_e(m)) = m$ pre všetky $m \in \mathcal{M}$.

1.3 Symetrické šifrovanie

Najčastejšie používame $e = d$. Symetrické šifry sú zväčša veľmi rýchle, takže dokážu zašifrovať veľa dát za krátky čas a kľúče sú relatívne krátke. Symetrické šifry môžu byť zapúzdrené, teda na jednu správu môže byť použitých viac šifier, vďaka čomu môžu dosahovať vačšiu mieru bezpečnosti. Na druhú stranu pri komunikácii dvoch entít býva dobrým zvykom meniť kľúče relatívne často a zároveň kľúč musí ostať v bezpečí počas celej komunikácie.

1.4 Asymetrické šifrovanie

Definícia nám hovorí, že keď máme $e \in \mathcal{K}$, tak je nemožné získat príslušný kľúč d taký aby platilo $D_d(E_e(m)) = m$. Aby bola táto vlastnosť zabezpečená, kryptografické funkcie sú založené na matematických problémoch ako je faktorizácia alebo výpočet diskrétneho logaritmu. Kľúč d budeme nazývať privátnym

a e verejným kľúčom. Pri využití asymetrickej kryptografie nám stačí uchovávať privátny kľúč a kľúče netreba meniť tak často ako pri symetrických šifrách. Nevýhodou oproti symetrickému šifrovaniu môže byť velkosť kľúčov a nižšia rýchlosť šifrovania .

1.5 Hašovacie funkcie

Vačšina kryptografických schém využívajúcich hašovacie funkcie využíva takzvané kryptografické hašovacie funkcie. Kryptografická hašovacia funkcia obvykle spĺňa nasledujúce vlastnosti:

- Jednosmernosť: k danému hašu $o \in O$ je výpočtovo "nemožné"získať akýkoľvek vstup $i \in I$ spĺňajúci $H(i) = o$.
- Odolnosť voči kolíziam: je výpočtovo "nemožné"nájsť dva rôzne vstupy s rovnakým hašom $i_1 \neq i_2$, kde $i_1, i_2 \in I$ pre ktoré platí, že $H(i_1) = H(i_2)$.

Kryptografické využitie je napríklad v digitálnych podpisoch, konštrukciách authentizačných kódov, pri ukladaní hesiel alebo môžu slúžiť na kontrolu integrity údajov.

1.6 Použité šifry

TODO =¿ kde ich budem používať

AES

Veľmi rýchla bloková šifra založená na permutáciach a substitúciach s veľkosťou bloku 128bitov. Dĺžka kľúča sa rôzni od 128 bitov cez 192 bitov až po 256 bitov. Algoritmus šifrovania je nasledovný:

1. Expanzia kľúča - podkľúče sú odvodené z kľúču

2. Inicializácia - xorujeme stav s podkľúčom
3. Cykly - prebiehajú zámeny bajtov, prehadzovanie riadkov, kombinovanie stĺpcov a xorovanie stavu s podkľúčom
4. Posledný cyklus - zámena bajtov, poprehadzovanie riadkov a xorovanie stavu s podkľúčom

Štandardizovaný americkým úradom pre štandardizáciu - NIST v štandarde FIPS - 197 [11]. Podľa už spomenutej organizácie by AES s dĺžkou kľúča 128 bitov mal byť bezpečný minimálne do roku 2030.

ECC

Co popisat k tomuto??

1.7 Cloudové úložiská

Cloudové úložisko je služba, ktorá nám umožnuje manipulovať s priestorom prenajatým od poskytovateľa služby. Táto služba by sa mala byť škálovateľná čo znamená, že vieme jednoducho zväčšiť priestor za ktorý platíme. Ďalej by sa mala starať o to aby naše dáta boli stále prístupné, čo zahŕňa ochranu voči strate a poškodeniu dát prípadne výpadkom siete.

Kapitola 2

Súčasné riešenia

V tejto kapitole si opíšeme riešenia podobné tomu nášmu. Rozdelíme si ich na také, ktoré využívajú vlastné cloudové riešenia, a také, ktoré používajú cloudové úložisko poskytované treťou stranou. Pozrieme sa na to ako akú kryptografiu používajú, koľko si účtujú a ako vedia zdieľať dáta. V závere si popíšeme, čo bude ponúkať naša služba, v čom sa odlišuje od ostatných a v čom je lepšia.

2.1 Existujúce riešenia na šifrované ukladanie dát

Súčasné služby môžeme rozdeliť do dvoch kategórií. Jedny sa rozhodli používať vlastný cloud a dalšie sa spoliehajú na cloudy tretej strany. Väčšina z nich poskytuje natívnu aplikáciu do počítača a mobilného zariadenia, tá menšia skupina sa zamerala na tvorbu webovej aplikácie, prostredníctvom ktorej užívateľ spravuje svoje dáta. Kryptografia prebieha na strane klienta pred prenosom dát do cloudu. Väčšina služieb sa snaží dodržiavať zásadu "zero-knowledge", ktorá zaručuje používateľovi, že poskytovateľ cloudového úložiska nebude mať o jeho dátach nijaké informácie.

Služby využívajúce vlastné cloudové riešenie

Jeden z najznámejších poskytovateľov šifrovaného cloudového úložiska je Mega [2]. Okrem webovskej aplikácie ponúka mobilné aj desktopové aplikácie, ktoré môžu niektorí užívatelia preferovať pred webovým rozhraním. Pri ukladaní súborov alebo zložiek Mega vygeneruje náhodný 128-bitový kľúč a následne dáta zašifruje šifrou AES-128. Všetky kľúče sú zašifrované pomocou univerzálneho hesla, ktoré sme si zvolili pri registrácii. To je zahašované kryptografickou hašovacou funkciou a uložené na serveroch. Celé šifrovanie prebieha na počítači klienta, takže Mega nemá žiadne informácie o obsahu uloženom v cloude a nepozná naše heslo. Pri zdieľaní súborov sa používa 2048 RSA kľúčový pár, pričom jeho privátna časť je zašifrovaná univerzálnym kľúčom. Služba ponúka 50 GB zdarma a za 500 GB používateľ zaplatí mesačne 9,99$. Keď sa rozhodneme zdieľať nejaký súbor Mega nám poskytne webovú adresu a kľúč ktorým je súbor zašifrovaný. Bohužiaľ distribúciu webovej adresy a hlavne hesla už nechá na nás.

Služby SpiderOak [6] a Wuala [5] tiež využívajú vlastný cloud, ale neposkytujú webové rozhranie a všetky operácie musíme robiť pomocou aplikácie na našom počítači. SpiderOak pracuje na podobných princípoch ako Mega ale namiesto AES-128 používa AES-256. Zadarmo sú dostupné 2 GB úložného priestoru a za 12$ mesačne je možné ho rozšíriť na 1 TB. Aplikácia SpiderOak ponúka takzvané zdieľacie miestnosti ktoré sú opäť definované nejakou linkou a heslom. Každý kto sa v tejto miestnosti nachádza má prístup ku kľúčom, ktoré šifrujú súbory dostupné v miestnosti.

Wuala využíva systém Cryptree [7] v ktorom používa AES-256 na šifrovanie, RSA 2048 na podpisy a zdieľanie dát a SHA-256 na zabezpečenie integrity. Cena sa pohybuje od 0,99 € za 5 GB do 159,90 € za 2 TB. Zdieľanie súborov v aplikácii Wuala funguje podobne ako pri službe Mega. Najprv si od aplikácie vypýtame link a následne ho pošleme aj s heslom k súboru tomu s kým chceme zdieľať.

Všetky tri služby sú kompatibilné s Windows, Mac, Linux, Android a iOS.

Využívanie dostupných cloudových riešení

Viivo [3] na rozdiel od služby Mega využíva už existujúce cloudové riešenia, ktoré poskytujú API na prácu so súbormi a vybudoval tak vrstvu medzi klientom a jeho obľúbeným cloudovými úložiskami Drive, Dropbox, Box alebo SkyDrive. Viivo vytvorí 2048 bitový kľúčový pár, ktorý sa používa pri zdieľaní dát. Kľúč je zabezpečený pomocou hesla, ktoré si užívateľ zvolí pri registrácii. Pre súkromné využitie je zadarmo, firmy si priplatia od 4,99$ do 9,99$ mesačne. Pri zdieľaní súborov pomocou Viiva musia mať obaja používatelia nainštalovanú aplikáciu a musia mať zdieľanú zložku v Dropboxe. Pri ostatných cloudoch som nenašiel informácie ako zdieľať.

Boxcryptor [4] je ďalšia služba podobná Viivu, ktorá vytvorila vrstvu medzi cloudom a požívateľom. Podporuje rovnaké cloudy ako Viivo a naviac ešte SugarSync. Kryptografia funguje na rovnakých princípoch ako Mega, ale využíva silnejšie kľúče, t.j. AES-256 pri symetrickej a RSA s kľúčom dĺžky 4096 bitov pri asymetrickej kryptografii. Základné šifrovanie je poskytované zadarmo a neobmedzený firemný účet stojí 96$ ročne. Aby sme vedeli zdieľať stačí aby bol ten s kým chceme zdieľať zaregistrovaný na www.boxcryptor.com a mal aplikáciu BoxCryptor. Následne treba vybrať súbory ktoré chceme zdieľať a pomocou aplikácie vybrať s kým. Na výmenu kľúčov sa používa server ktorý ukladá všetky kľúče.

Viivo ani Boxcryptor neposkytujú webové rozhranie, takže používateľ je nútený inštalovať dodatočný softvér ktorý je kompatibilný s Windowsom, Macom ako aj s iOS a Androidom.

2.2 Naše riešenie

Cieľom našej práce bude implementovať službu, ktorá umožní používateľom bezpečne a jednoducho ukladať, sťahovať a zdieľať súbory v cloude. Rozhodli sme sa využívať cloudové úložisko poskytované treťou stranou, konkrétne Google-Drive. Všetko budeme ovládať cez webové rozhranie, ktoré má rôzne výhody, t.j. netreba inštalovať žiadny softvér okrem webového prehliadača,

ktorý je štandardne predinštalovaný, a zároveň pomerne jednoducho vieme zabezpečiť kompatibilitu aplikácie od mobilu až po desktop a tiež sa nemusíme starať o to aký operačný systém náš používateľ uprednostnuje. Riešenie sa pokúsime implementovať tak, aby bolo jednoduché ho rozšíriť o iné cloudové úložiská ako napríklad Dropbox, SkyDrive a iné. Našu službu sme sa rozhodli nazvať SecureCloud .

2.3 Porovnanie

V tejto časti vysvetlíme, v čom sa bude naše riešenie líšiť od ostatných služieb a aká je naša motivácia vytvoriť vlastné riešenie.

Mega vs SecureCloud

Mega patrí medzi najlepších poskytovateľov šifrovaných cloudových riešení na trhu. Okrem toho, že ponúka webovské rozhranie dáva nám aj dostatok priestoru zdarma. Avšak zmena cloudového úložiska implikuje problémy ako nedostupnosť starých súborov na novom úložisku poprípade iné prostredie. Medzi najpoužívanejšie a najznámejšie cloudové úložiská rozhodne patrí Google-Drive a Dropbox, ale ani jedno neponúka šifrovanie dát. Výhoda nášho riešenia oproti službe Mega spočíva v možnosti pokračovať vo využívaní služieb Googlu alebo Dropboxu a zároveň v zabezpečení šifrovania dát. Taktiež zdieľanie v našom riešení nevyžaduje nič iné ako registráciu užívateľa, s ktorým chceme zdieľať a pár klikov myškou. Zároveň pri zdieľaní zabezpečíme bezpečnú distribúciu kľúču k druhej strane a nedáme jej možnosť kompromitovať kľúč.

Viivo a Boxcryptor vs SecureCloud

Napriek tomu, že Viivo aj Boxcryptor ponúkajú využívanie vrstvy medzi obľúbenými poskytovateľmi úložísk a používateľom, nemajú nijaké webové rozhranie, čo zaťažuje používateľa okrem registrácie aj inštaláciami mobilných

a desktopových aplikácií na všetkých zariadeniach, na ktorých budú službu využívať. Naopak naše riešenie vyžaduje iba prihlásenie pomocou už existujúceho Dropbox alebo Google konta. Oproti Viivo máme aj veľkú výhodu v jednoduchšom zdieľaní súborov.

Kapitola 3

Návrh funkcionality

Ďalej opíšeme ako je naše riešenie navrhnuté. Popíšeme ako budeme postupovať v prípade nahrávania, sťahovania, zdieľania a čo spravíme keď už nechceme zdieľať dáta. Potom si popíšeme, čo treba spraviť predtým, ako budeme môcť tieto akcie vykonať.

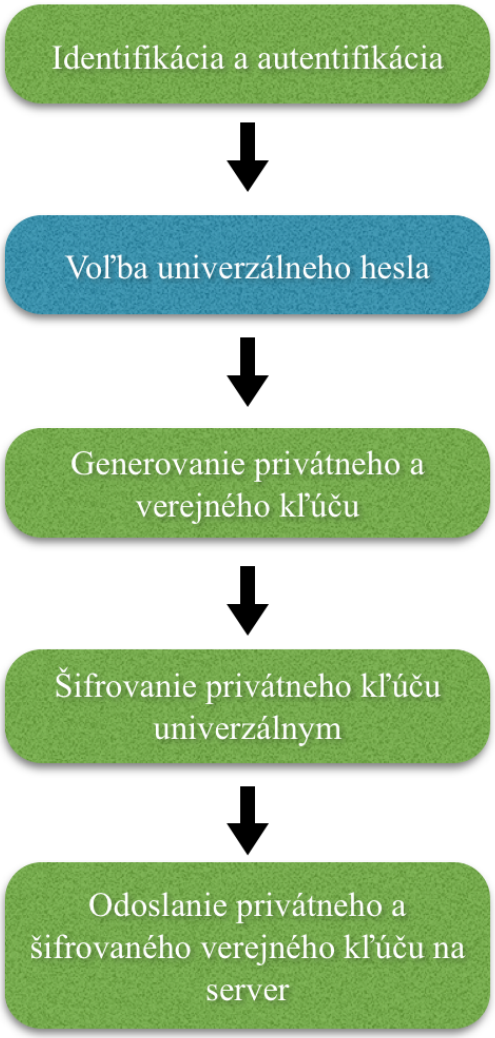
3.1 Prerekvizity

Aby naše riešenie fungovalo, budeme potrebovať server, ktorý bude poskytovať API na komunikáciu s databázou a bude hostovať naše webové rozhranie. Databázu kam budeme ukladať užívateľove kľúč a kľúče k súborom. Cloudové úložisko, ktoré bude zabezpečovať prácu so súbormi. Cloud musí poskytovať rozhranie pomocou ktorého vieme nahrávať a sťahovať dáta a taktiež musí byť schopné autorizovať rôznych užívateľov pre prístup k dátam iných užívateľov, aby sme mohli zdieľať súbory s inými používateľmi.

3.2 Registrácia

Keď sa používateľ rozhodne používať našu službu musí sa nejakým spôsobom identifikovať a autentifikovať. Identifikácia vyžaduje aby si každý používateľ

zvolil nejaké meno a aby sme ho mohli authentifikovať potrebuje heslo. Druhou možnosťou je využiť tretiu službu ako napríklad cloud kde budeme ukladať súbory. Toto je možné napríklad pomocou protokolov oAuth alebo openID. Nech už vybereme jeden alebo druhý spôsob budeme jednoznačne vedieť s kým komunikujeme. S touto informáciou vyzveme užívateľa aby si zvolil svoje univerzálne heslo pomocou ktorého bude autorizovať akcie ako nahrávanie, sťahovanie, zdieľanie a zrušenie zdieľania dát.
Následne v užívateľovom prehliadači vygenerujeme verejný a privátny kľúč, ktorý symetricky zašifrujeme pomocou už zvoleného univerzálneho hesla. Verejný aj zašifrovaný privátny kľúč uložíme na serveri kde prístup k verejným kľúčom budú mať všetci používatelia no k privátnym len jeho vlastníci. Tým že privátny kľúč je zašifrovaný server nemá žiadnu informáciu o privátnom kľúči a teda schéma ostane bezpečná.
Ďalšou možnosťou bolo uložiť privátny kľúč na užívateľovom počítači. Toto sme sa rozhodli nevyužiť z dôvodu náročného manažmentu kľúčov na rôznych zariadeniach. Napríklad pri strate zariadenia by to vyžadovalo zmenu privátneho kľúču, čo by implikovalo nutnosť prešifrovania všetkých súborových kľúčov. Poprípade distribúcia kľúču do rôznych zariadení by bola v rukách používateľa čo by mohlo byť nepohodlnné.

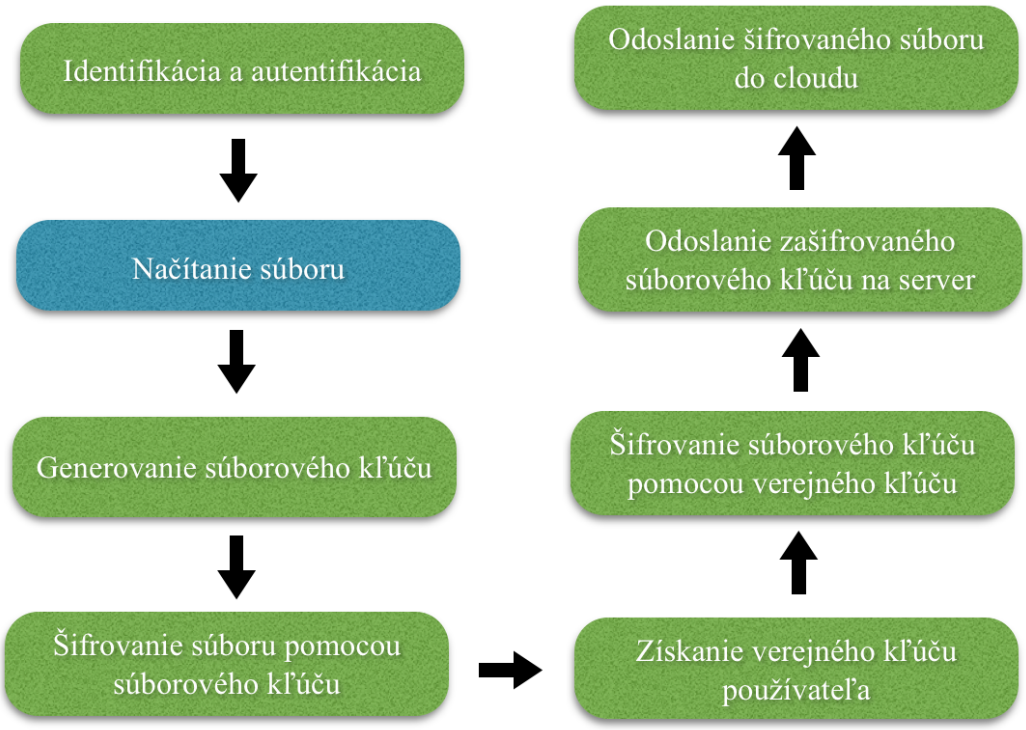


Obr. 3.1: Priebeh registracie

3.3 Nahrávanie dát

Užívateľ sa authentifikuje a v prehliadači a pomocou pseudonáhodného generátoru vygeneruje náhodný text. Ten využije ako heslo, s ktorým zašifruje súbor a vypýta si od servera svoj verejný kľúč. Pomocou verejného kľúču

zašifruje heslo k súboru a v taktomto tvare ho už môže poslať na server. Keďže heslo je zašifrované a server nemá žiadnu informáciu o privátnom kľúči, nevie získať dáta v otvorenom tvare. Následne už len stačí zašifrovaný súbor nahrať na cloudové úložisko.

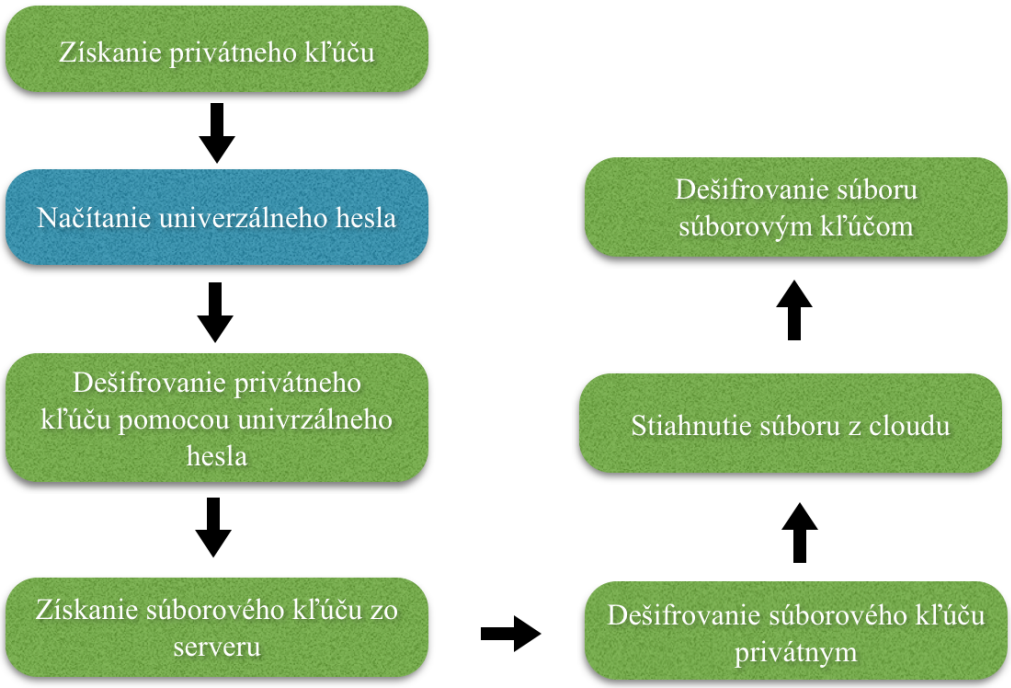


Obr. 3.2: Nahrávanie dát

3.4 Sťahovanie dát

V prípade, že privátny kľúč je uložený na serveri, tak si ho vypýtame a požiadame užívateľa o jeho univerzálne heslo, aby sme mohli privátny kľúč dešifrovať. S privátnym kľúčom môžeme následne dešifrovať heslo k súboru, ktoré si opäť vypýtame od servera. V tejto chvíli máme k dispozícii kľúč k

súboru a teda nám stačí stiahnuť súbor z cloudu a rozšifrovať ho pomocou zmieneného kľúča.

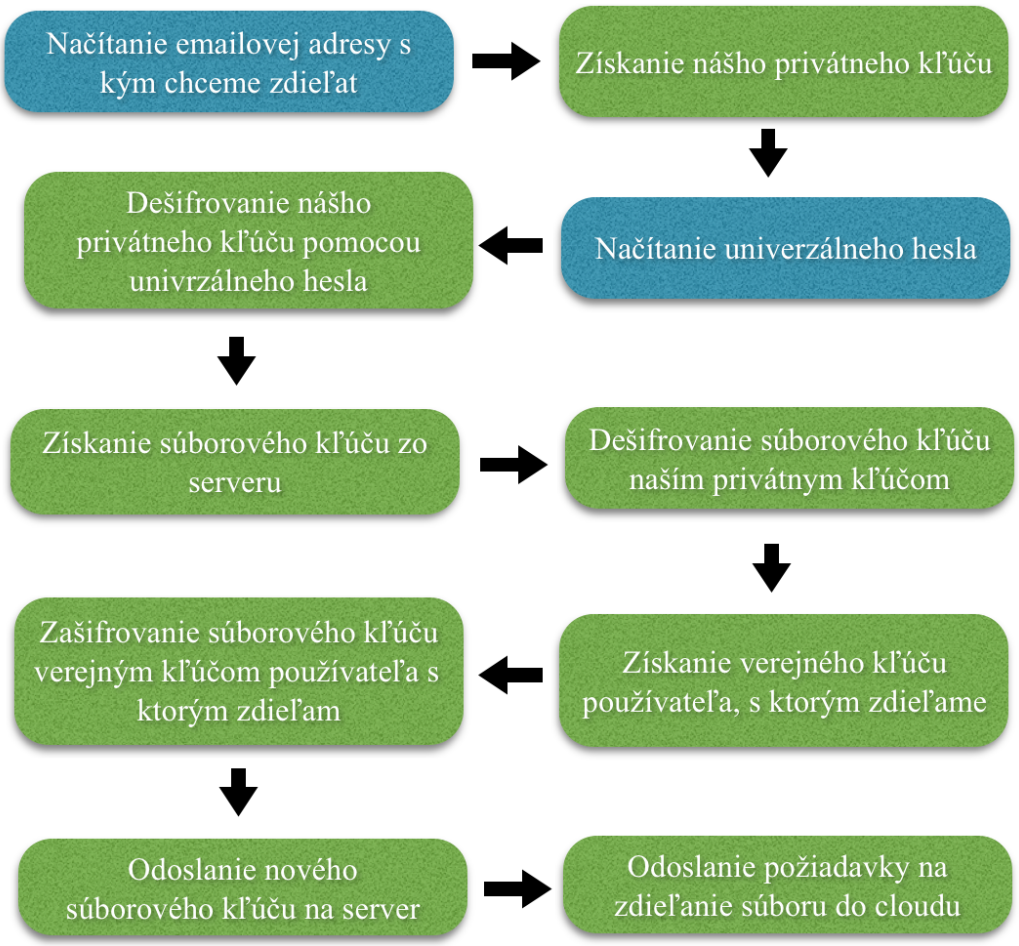


Obr. 3.3: Sťahovanie dát

3.5 Zdieľanie

Chceme zdieľať súbor ktorý je uložený v zašifrovanej forme na cloude. Na zdieľanie je nutné, aby cieľový príjmatel bol zaregistrovaný na našom serveri a mal vygenerovaný privátny a verejný kľúč. Zdieľanie bude prebiehať tak, že používateľ si od serevra vypýta privátny kľúč, ktorý dešifruje pomocou svojho univerzálneho kľúču. Následne požiada server o zašifrovaný kľúč k súboru, ktorý dešifruje pomocou privátneho kľúču. Dešifrovaný kľúč k súboru potom zašifruje verejným kľúčom používateľa, s ktorým chce dáta zdieľať a takýto

kľúč k súboru uloží na serveri. Následne pošle požiadavku na cloud, aby povolil prístup k súboru používateľovi, s ktorým zdieľame. Príjemca nášho zdieľaného súboru má teraz prístup ku kľúču na dešifrovanie súboru a tiež si môže stiahnuť zašifrovaný súbor z cloudu.



Obr. 3.4: Zdieľanie dát

3.6 Zrušenie zdieľania

V prípade, že sa rozhodneme zrušiť zdieľanie súboru, stačí, aby sme požiadali cloudové úložisko o revokovanie prístupu k súborom. Keby sa používateľ, s ktorým ten súbor zdieľame, rozhodol ho zverejniť alebo inak distribuovať, šifrovanie nám nepomôže, pretože už si mohol spraviť kópiu nešifrovanej verzie. Preto stačí revokovať prístup na cloude a nemusíme súbor prešifrovať. V prípade, že sa súbor zmení a budeme ho znova uploadovať prebehne opäť procedúra ako pri nahrávaní dát. Nové informácie teda nebudú kompromitované.

Kapitola 4

Implementácia

V tejto kapitole si popíšeme hlavné implementačné detaily. Pozrieme sa s akými technológiami sme sa rozhodli pracovať. Ukážeme si čo všetko dokáže knižnica ktorú sme si vybrali na implementáciu kryptografie. A taktiež sa pozrieme na reálnu architektúru riešenia.

4.1 Použité technologie

Zhrnieme si aké technológie naša implementácia využíva na strane klienta a aké na strane serveru a v krátkosti si ich popíšeme.

4.1.1 Back-end

Pod pojmom back-end máme na mysli implementáciu riešenia mimo užívateľovho počítaču. Na implementáciu serverovej časti sme sa rozhodli použiť Node.js s frameworkom Express, databázový systém MongoDB s pomocou objektového modelovacieho nástroju mongoose.

Node.js

Je platforma postavená na Googlovskom V8 javascript engine ktorý je napísaný v C++. Písať v jednom jazyku front-end aj back-end už dnes vďaka Node.js nie je problém. Celý model tejto platformy je postavený na udalostiach a neblokujúcich operáciach takže je vhodný pre real-timeové aplikácie.

Express

Express je webovy framework postavený na Node.js, ktorý nám zjednodušuje implementáciu backendu. S pomocou Expressu vieme veľmi ľahko vytvoriť REST API. Napríklad napísať čo sa stane keď užívateľ pristúpi k webovej adrese http://mojserver.sk/ahoj by vyzeralo nasledovne:

```
1        app.get('/ahoj', function (req, res) {
2          res.send('Hello World!');
3        });
```

Zdrojový kód 4.1: Express routing

MongoDB

MongoDB sme sa rozhodli použiť kvôli dynamickej schéme, škálovateľnosti, ale aj kvôli možnosti pracovať s moongose. Moongose je nástroj, ktorý nám dáva možnosť pracovať s databázou ako s obyčajnými objektami, čo výrazne urýchľuje vývoj. Pre ilustráciu vytvorenie nového dokumentu v databáze bude vyzerať nasledovne:

```
1        var Cat = mongoose.model('Cat', { name: String });
2
3        var kitty = new Cat({ name: 'Murko' });
4        kitty.save(function (err) {
5          if (err) // napriklad chyba v databaze
6          console.log('meow');
7        });
```

Zdrojový kód 4.2: Vytvorenie mongoose modelu a jeho použitie

4.1.2 Front-end

Za front-end budeme považovať všetky programy a scripty, ktoré sa dostanú k užívateľovi a budú zobrazované alebo vykonávane v jeho prehliadači.

jQuery

Pre vykreslenie HTML stránky väčšina moderných prehliadačov vnútorne používa nejakú reprezentáciu, ktorú budeme nazývať objektový model dokumentu, tzv. DOM, z anglického Document Object Model. Pre prácu s DOM, používame knižnicu jQuery. Jej dokumentácia a developerská podpora je velmi rozsiahla čo uľahčuje jej používanie. Knižnicu jQuery budeme tiež využívať na AJAX volania, pre ktoré nám poskytuje jednoduché API. AJAX slúži na výmenu dát medzi serverom a clientom. Napríklad keď si chceme vypýtať od servru privátny kľúč pošleme GET požiadavku na http://server.com/getPrivKey a dostaneme od neho s privátnym kľúčom používateľa.

SJCL

Kryptografiu nám bude zaobstarávať knižnica Stanford Javascript Crypto Library, ktorej zdrojový kód a dokumentácia je online [9]. Táto open-source knižnica ponúka jednoduché rozhranie, pomocou ktorého vieme šifrovať a dešifrovať dáta. Podporuje ako symetrickú, tak aj asymetrickú kryptografiu rôzne hašovacie funkcie a tiež umožnuje vytvárať a overovať podpisy. Okrem toho, že nám poskytuje všetky kryptografické primitíva, ktoré naše riešenie vyžaduje, jej hlavnou výhodou je efektívna implementácia. V rýchlosti šifrovania je v priemere 4x rýchlejšia ako existujúce riešenia [8]. Kompatibilita medzi všetkými modernými prehliadačmi ako Chrome, Firefox, Safari a Internet Explorer v kombinácii s rýchlostou a jednoduchým používatelským rozhraním boli kritickými prvkami pri výbere.

4.2 Implementáciu návrhu

Túto časť venujeme prevažne zdrojovému kód nášho riešenia. Prejdeme cez kód registrácie, nahrávania a sťahovania dát a nakoniec sa pozrieme ako funguje zdieľanie. Niektoré časti kódu pre jednoduchosť vynechám a pokúsim sa uviesť iba tie najdôležitejšie časti.

4.2.1 Registrácia

Aby sme vedeli s kým komunikujeme potrebujeme najprv používateľa overiť. Na toto sme sa rozhodli použiť oAuth protokol. S jeho pomocou dostaneme informácie o tom s kým komunikujeme a zároveň nás ním aj autorizuje k používaniu jeho Google-Drive účtu. Následne potrebujeme vygenerovať privátny a verejný kľúč, vypýtať si od používateľa jeho univerzálne heslo a zašifrovať privátny kľúč univerzálnym heslom. Potom ho už len stačí poslať na server kde sa uloží.

```
1  function generateKeyPair(callback) {
2      var keys = sjcl.ecc.elGamal.generateKeys(256);
3
4      //SERIALIZING KEYPAIR
5      var pub = keys.pub.get();
6      var sec = keys.sec.get();
7      var serializedKeyPair = {
8          pub: sjcl.codec.base64.fromBits(pub.x.concat(pub.y)),
9          priv: sjcl.codec.base64.fromBits(sec)
10     };
11
12     // ASKING USER FOR HIS PASSWORD
13     var passwd = prompt('Enter your new universal password.
           You will need this password to decrypt and share files
           ', 'Password here');
14     // ENCRYPTING USERS PRIVATE KEY
15     var enc = sjcl.encrypt(passwd, serializedKeyPair.priv);
16     serializedKeyPair.priv = enc;
17
```

```
18      // SENDING KEYPAIR TO SERVER
19      saveKeyPair(serializedKeyPair, callback);
20  }
21
22  function saveKeyPair(keyPair, callback) {
23      // SEND KEYPAIR TO SERVER
24      $.post('/saveKeypair', keyPair, function (res2) {
25          callback(res2);
26      });
27  }
```

Zdrojový kód 4.3: Generovanie kľúčového páru a odoslanie na server

4.2.2 Nahrávanie dát

Keď chceme uložiť ľubovolné dáta v cloude v šifrovanej forme najprv potrebujeme načítať súbor.

```
1  function uploadFile() {
2      if ($('input#uplFile')[0].files[0] != undefined){
3          updateFileResumableGoogle({'title': $('input#uplFile'
               )[0].files[0].name + '.sc'}, $('input#uplFile')
               [0].files[0], function (uplResp) {// possible to
               add callback});
4      }
5  }
```

Zdrojový kód 4.4: Načítanie súboru

Po načítaní celého súboru vygenerujeme náhodný kľúč, ktorý požijeme na zašifrovanie dát súboru. Zo serveru si vypýtame náš verejný kľúč a ním zašifrujeme kľúč použitý na šifrovanie súboru. Nakoniec odošleme súbor do cloudu a zašifrované súborové heslo na server.
Vačšina našich funkcií používa tzv. callback, čo je vlastne funkcia, ktorá sa zavolá po skončení predchádzajúcej funkcie. Napríklad, keď zavoláme getPubKey(email, callback), tak najprv sa vykoná funkcia getPubKey, a keď

skončí svoju prácu zavolá funkciu callback(pubKey). Ako parameter je vždy návratová hodnota predchádzajúcej funkcie.

```
6  function updateFileResumableGoogle(metadata, fileData,
       callback) {
7      var accessToken = gapi.auth.getToken().access_token;
8      var email = $.cookie('email');
9
10     //GET PUBLIC KEY OF LOGGED USER
11     getPubKey(email, function(pubKey) {
12
13         //GENERATE PASSWORD FOR FILE ENCRYPTION
14         createRandomString(function (passString) {
15
16             // FILE UPLOAD
17             var reader = new FileReader();
18             reader.readAsArrayBuffer(fileData);
19             reader.onload = function() {
20
21                 //FILE ENCRYPTION
22                 var bits = sjcl.codec.bytes.toBits(new
                       Uint8Array(reader.result));
23                 var crypt = sjcl.encrypt(passString, bits);
24                 var blob = new Blob([crypt]);
25
26                 //ACTUAL UPLOADING
27                 var uploader = new MediaUploader({
28                     metadata: metadata,
29                     file: blob,
30                     token: accessToken,
31                     onComplete: function(gResp) {
32
33                         //FILE PASSWORD ENCRYPTION
34                         var encPass = sjcl.encrypt(pubKey,
                               passString);
35                         gResp = JSON.parse(gResp);
36
```

```
37                          //SENDING ENCRYPTED FILE PASSWORD TO
                                SERVER
38                          saveFileKey(email, encPass, gResp.id,
                                callback);
39                      },
40                      onError: function(err) {
41                          log(err);
42                      }
43                  });
44                  uploader.upload();
45              };
46          });
47      });
48  }
```

Zdrojový kód 4.5: Upload súboru

4.2.3 Sťahovanie dát

V prípade sťahovania súborov si klikneme na súbor vo webovom rozhraní a potom na tlačítko "Download", ktoré vykoná nikoľko akcií. V prvom rade získame zo serveru privátny kľúč a rozšifrujeme ho našim univerzálnym heslom.

```
1  function getPrivKey(callback) {
2      $.get('/getPrivKey', function(res) {
3          // DECRYPTS PRIVATE KEY WITH USERS KEY
4          var userKey = prompt('Please enter your password', '
               Enter your password here');
5          var privKey = sjcl.decrypt(userKey, res);
6
7          // DESERIALIZING PRIVATE KEY
8          var sec = new sjcl.ecc.elGamal.secretKey(
9              sjcl.ecc.curves.c256,
10             sjcl.ecc.curves.c256.field.fromBits(sjcl.codec.
                   base64.toBits(privKey))
11         );
```

```
12
13          callback(sec);
14      });
15  }
```

Zdrojový kód 4.6: Získanie privátneho kľúču

Keď už ho máme k dispozícii požiadame server o zašifrovaný kľúč k súboru a rozšifrujeme ho pomocou privátneho kľúču.

```
16  function getFileKey(fileId, callback) {
17      getPrivKey(function (privKey) {
18          $.post('/getFileKey', {'fileId': fileId}, function(
                encFileKey) {
19
20              // DECRYPTS FILE KEY WITH USERS PRIVATE KEY
21              var fileKey = sjcl.decrypt(privKey, encFileKey);
22
23              callback(fileKey);
24          });
25      })
26  }
```

Zdrojový kód 4.7: Získanie súborového kľúču

Teraz už môžme prejsť k samotnému stiahnutiu súboru a jeho rozšifrovaniu.

```
27  function downloadFileGoogle(file, callback) {
28      if (file.downloadUrl) {
29          var accessToken = gapi.auth.getToken().access_token;
30          var xhr = new XMLHttpRequest();
31          xhr.open('GET', file.downloadUrl);
32          xhr.setRequestHeader('Authorization', 'Bearer ' +
                accessToken);
33
34          // ITS ENCRYPTED FILE
35          xhr.onload = function() {
36              getFileKey(file.id, function(fileKey) {
37
38                  // DECRYPT FILE
```

```
39                  var base64decrypt = sjcl.decrypt(fileKey, xhr
                        .response, {'raw': 1});
40                  var byteArray = new Uint8Array(sjcl.codec.
                        bytes.fromBits(base64decrypt));
41
42                  // PASS DOWNLOADED FILE TO CALLBACK
43                  callback({
44                      'title': title,
45                      'data': byteArray
46                  });
47              });
48          };
49          xhr.onerror = function() {
50              console.log("ERROR!");
51          };
52          xhr.send();
53      } else {
54          console.log("ERROR MISSING DOWNLOAD URL");
55      }
56  }
```

Zdrojový kód 4.8: Stiahnutie súboru

Keď všetko prebehne úspešne zavoláme callback funkciu ktorej ako parametre dáme meno súboru a jeho dáta.

4.2.4 Zdieľanie dát

To čo robí naše riešenie zaujímavým je zdieľanie súborov. Pre zdieľanie súboru nám stači zavolať funkciu ktorej podstrčíme štyri parametre. Id súboru, email užívateľa s ktorým chceme zdieľať, rolu a funkciu ktorá sa má po skončení funkcie vykonať. Parameter role nám hovorí aké právomoci má užívateľ k tomu súboru. Môže byť čitatel, čo mu umožní súbor iba čítať, zapisovateľ môže meniť dáta súboru alebo vlastník, čo znamená, že môže robiť všetko ako čitateľ a zapisovateľ ale naviac môže súbor aj zmazať. Celé zdieľanie vyžaduje niekoľko krokov. V prvom rade musíme získať kľúč k súboru rovnako ako pri sťahovaní dát [4.7]. Zatým verejný kľúč používateľa

s ktorým plánujeme zdieľať súbor. Následne zašifrujeme súborový kľúč verejným kľúčom, ktorý sme získali a uložíme ho na server. A napokon už len stačí požiadať cloud aby povolil prístup k súboru používateľovi, s ktorým sme sa rozhodli podeliť o naše dáta.

```
1  function shareFileGoogle(fileId, email, role, callback) {
2
3      getFileKey(fileId, function(fileKey) {
4          getPubKey(email, function (pubKey) {
5
6              //ENCRYPT FILEKEY WITH PUB-KEY OF USER I AM
                   SHARING WITH
7              var shareFileKey = sjcl.encrypt(pubKey, fileKey);
8
9              //SAVE NEW ENCRYPTED KEY
10             saveFileKey(email, shareFileKey, fileId, function
                   (saveFileKeyResponse) {
11
12                 //SHARE ON GOOGLE DRIVE (ADD PERMISSION FOR
                       USER)
13                 insertPermissionGoogle(fileId, email, 'user',
                       role, function(resp) {
14                     callback(resp);
15                 });
16             });
17         });
18     });
19 }
```

Zdrojový kód 4.9: Zdieľanie súboru

4.2.5 Zrušenie zdieľania

Aby sme naše dáta prestali zdieľať stačí požiadať cloud o zrušenie prístupu k súboru a aby sme zbytočne nezaplnovali databázu vymažeme aj súborový kľúč zo servru.

```
1  function stopShareingFileGoogle(fileId, email, callback) {
```

```
2       retrievePermissions(fileId, function(permissions) {
3           for(var permission in permissions) {
4               if (permissions[permission].emailAddress == email
                    ) {
5                   var permId = permissions[permission].id;
6                   removePermissionGoogle(fileId, permId,
                        function(resp) {
7                       callback(resp);
8                   });
9               }
10          }
11      });
12  }
```

Zdrojový kód 4.10: Zrušenie zdieľanie súboru

Záver

V závere je potrebné v stručnosti zhrnúť dosiahnuté výsledky vo vzťahu k stanoveným cieľom. Rozsah záveru je minimálne dve strany. Záver ako kapitola sa nečísluje.

Literatúra

[1] Alfred J. Menezes - Paul C. van Oorschot - Scott A. Vanstone, 1996, Handbook of Applied Cryptography, CRC Press

[2] Mega, Februar 2015, [online] Dostupné na internete: https://mega.co.nz/#doc

[3] Viivo, Februar 2015, [online] Dostupné na internete: https://viivo.com/

[4] BoxCryptor, Februar 2015, [online] Dostupné na internete: https://www.boxcryptor.com/en

[5] Wuala, Februar 2015, [online] Dostupné na internete: https://www.wuala.com/en/learn/technology

[6] SpiderOak, Februar 2015, [online] Dostupné na internete: https://spideroak.com/

[7] Grolimund D. - Meisser L. - Schmid S. - Wattenhofer R., Cryptree: A Folder Tree Structure for Cryptographic File Systems, [online] Dostupné na internete: http://dcg.ethz.ch/publications/srds06.pdf

[8] Stark E. - Hamburg M. - Boneh D., Symetric Cryptography in Javascript, 2009, Annual Computer Security Applications Conference, [online] Dostupné na internete: https://bitwiseshiftleft.github.io/sjcl/acsac.pdf

[9] SJCL, Stanford Javascript Crypto Library, April 2015, [online] Dostupné na internete: https://github.com/bitwiseshiftleft/sjcl/

[10] SJCL, Stanford Javascript Crypto Library, April 2015, [online] Dostupné na internete: https://github.com/bitwiseshiftleft/sjcl/wiki

[11] NIST, Processing Standards Publication 197 - AES, April 2015, [online] Dostupné na internete: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf