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

Kryptografia

Prvá kapitola slúži ako teoretický úvod do kryptografie. Vymenujeme jej ciele, zadefinujeme základné pojmy a 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.

Ciele kryptografie

Na úvod popíšeme základné kryptografické ciele ako dôvernosť, integritu a autentickosť. Z cieľov vynecháme dostupnosť, ktorú kryptografia ako taká nedokáže zabezpečiť.

- Dôvernosť je vlastnosť, ktorá nám zaručuje, že k dátam sa dostanú len také entity, ktorým bola správa určená a nikto iný. Dôvernosť dát budeme zabezpečovať šifrovaním.
- Integrita je vlastnosť, ktorá hovorí o modifikácii dát. Aby sme zaručili

integritu dát, musí byť zaručená možnosť detegovať manipuláciu s dátami neoprávnenými entitami.

- Autentickosť je vlastnosť, ktorá hovorí o pôvode entity alebo dát. Teda keď Anička pošle správu Bohušovi, Bohuš bude môcť overiť, že správa je skutočne od Aničky.

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. Pokiaľ 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

Nech šifrovacia schéma pozostáva z množín $\{E_e : e \in K\}$ a $\{D_d : d \in K\}$ kde $\mathcal{K}$ je množina všetkých kľúčov. Takúto schému nazveme symetrickou pokiaľ pre každý pár (e, d) platí, že je "ľahké", vypočítať d pomocou e, a opačne. Najčastejšie používame $e = d$. Symetrické šifry sú zväčša veľmi rýchle, takže dokážu zašifrovať veľa dát za krátky čas a taktiež kľúče sú relatívne krátke. Symetrické šifry môžu byť zapúzdrené, teda na jednu správu môže byť použitých viac šifier, vďaka čomu môžu dosahovať väčšiu mieru bezpečnosti. Na druhú stranu pri komunikácii dvoch entít býva dobrým zvykom meniť kľúče relatívne často a taktiež kľúč musí ostať v bezpečí počas celej komunikácie.

1.4 Asymetrické šifrovanie

Nech $\{E_e : e \in K\}$ je množina šifrovacích funkcií a nech $\{D_d : d \in K\}$ je množina príslušných dešifrovacích funkcií a $\mathcal{K}$ je množina všetkých kľúčov. Nech pre každý pár (E_e, D_d) platí, že je výpočtovo "nemožné"získať správu $m \in \mathcal{M}$ pomocou $c \in\ C$ a E_e, keď platí $E_e(m) = c$.
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 taktiež kľúče netreba meniť tak často ako pri symetrických šifrách. Nevýhodou oproti symetrickému šifrovaniu môže byť velkosť kľúčov a rýchlosť šifrovania ktorá býva často nižšia.

1.5 Hašhovacie funkcie

Hašovacou funkciou nazveme funkciu ktorá k vstupu ľubovolnej dĺžky vráti reťazec bitov pevnej dĺžky. Jej výstup budeme nazývať haš. Formálnejšie $H : I \to O$ hašovacou funkciou H nazveme funkciu zobrazujúcu z množiny vstupov I na množinu hašov O. Pre rovnaký vstup bude výstup vždy rovnaký.
Vačšina kryptografických schém využívajúcich hašovacie funkcie využíva takzvané kryptografické hašovacie funkcie. Kryptografická hašovacia funkcia obvykle spĺňa nasledujúce vlastnosti:

1. Jednosmernosť: k danému hašu $o \in O$ je výpočtovo "nemožné"získať akýkoľvek vstup $i \in I$ spĺňajúci $H(i) = o$.

2. 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 taktiež môžu slúžiť na kontrolu integrity údajov.

1.6 Použité šifry

AES

Ako moc to mam popisovat??

ECC

Co popisat k tomuto??

Kapitola 2

Súčasné riešenia

V nasledujúcej kapitole popíšeme čo sú vlastne cloudové úložiská, poskytneme prehľad existujúcich riešení šifrovaných úložísk. Uvedieme ich poskytovné služby a spôsob, akým fungujú. V závere kapitoly ich porovnáme s naším riešením.

2.1 Cloudové úložiská

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

2.2 Existujúce riešenia na šifrované ukladanie dát

Súčasné služby môžeme rozdeliť do dvoch kategórií. Väčšina z nich poskytuje natívnu aplikáciu do počítača alebo mobilného zariadenia, tá menšia skupina sa zamerala na tvorbu webovej aplikácie, prostredníctvom ktorej užívateľ

spravuje svoje dáta. Šifrovanie prebieha na strane klienta pred prenosom dát do cloud. Väčšina služieb sa snaží dodržiavať zásadu "zero-knowledge", ktorá zaručuje užívateľovi, že poskytovateľ cloudového úložiska nebude mať o jeho dátach nijaké informácie.

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 dostupné 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 univerzálna časť je zašifrovaná univerzálnym kľúčom. Služba ponúka 50 GB zdarma a za 500 GB používateľ zaplatí 9,99$.

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.

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.

Všetky tri služby podporujú Windows, Mac, Linux, Android a iOS.

Využívanie dostupných cloudových riešení

Viivo [3] na rozdiel od Megy 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.

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, neobmedzený firemný účet stojí 96$ ročne.

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

2.3 Cieľ práce

Pre naše riešenie sme sa rozhodli skombinovať dva prístupy. Rozhodli sme sa používať už existujúce cloudové úložiská ku ktorým vytvoríme webovské rozhranie. Netreba ho inštalovať, čo zvyšuje pouziteľnosť a podporuje okrem počítačov aj mobilné zariadenia. Keby sme sa rozhodli pre natívnu aplikáciu, nielen že by sme potrebovali naprogramovať aj mobilný variant, ale aj by sme zaťažovali cieľového užívateľa sťahovaním a inštalovaním. Preto sme vytvorili jednoduché a prehľadné prostredie, z ktorého bude možné využívať viacero úložísk. Pre testovanie a koncept návrhu budeme využívať cloudové úložisko firmy Google, Drive. Našu službu sme sa rozhodli nazvať SecureCloud.

2.4 Porovnanie

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

Mega vs SecureCloud

Mega patrí medzi najlepších poskytovateľov šifrovaných cloudových riešení na trhu. Bohužial veľa ľudí nechce začať využívať iné riešenie ako to, na aké boli doteraz zvyknutí. Medzi najpoužívanejšie a najznámejšie rozhodne patrí Google-Drive a Dropbox, ale ani jedno neponúka šifrovanie dát. Výhoda SecureCloudoproti Mega spočíva v možnosti pokračovať vo využívaní služieb Googlu alebo Dropboxu a zároveň v zabezpečení šifrovania dát. Modelový užívateľ, ktorý uprednostní naše riešenie oproti riešeniu Megy, je taký, ktorý má napríklad zaplatený poplatok za priestor u jedného zo spomenutých prevádzkovateľov.

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.

Kapitola 3

Návrh funkcionality

Ďalej opíšeme ako sme navrhli naše riešenie. Popíšeme ako budeme postupovať v prípade nahrávania, sťahovania, zdieľania a čo spravíme keď už nechceme zdieľať dáta. A taktiež 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, ktorá bude ukladať užívateľove dáta a 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 boli schopný zdieľať súbory s inými používateľmi.

3.2 Registrácia

Keď sa používateľ rozhodne pužívať našu službu musí sa nejakým spôsobom identifikovať a autentifikovať. K tomuto poslúži nejaká forma registrácie pri

ktorej si užívateľ zvolí 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 servri 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.

3.3 Nahrávanie dát

Užívateľ si v prehliadači vygeneruje, pomocou dostatočne náhodného generátoru, náhodné heslo ktorým zašifruje súbor a vypýta si od servru svoj verejný kľúč, ktorým heslo zašifruje. V taktomto tvare ho už môže poslať na server. Keďže heslo je zašifrované a server nemá informáciu o privátnom kľúči užívateľove dáta by mali byť stále v bezpečí. Následne už len stačí zašifrovaný súbor nahrať na cloudové úložisko.

3.4 Sťahovanie dát

Úplne na začiatku si vypýtame od servru privátny kľúč užívateľa a zároveň si od užívateľa vypýtame univerzálne heslo aby sme mohli privátny kľúč dešifrovať. S privátnym kľúčom môžme následne dešifrovať heslo k súboru, ktoré si opäť vypýtame od servru. V tejto chvíli máme k dispozícii kľúč k súboru a teda nám stačí stiahnuť súbor z cloudu a rozšifrovať ho pomocou zmieneného kľúču.

3.5 Zdieľanie

Chceme zdieľať súbor ktorý je uložený v zašifrovanej forme na cloude. Keď chceme zdieľať náš súbor je podmienkou aby bol ten s kým chceme zdieľať

zaregistrovaný na našom servri a mal vygenerovaný privátny a verejný kľúč. Zdieľanie bude prebiehať tak, že používateľ si od servru 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 následne zašifruje verejným kľúčom používateľa s ktorým chce zdieľať a takýto kľúč k súboru uloží na servri. Potom pošle požiadavku na cloud aby povolil prístup k súboru používateľovi s ktorým zdieľame. Príjemnca nášho zdieľaného súboru má teraz prístup ku kľúču na dešifrovanie súborou a taktiež si môže stiahnuť zašifrovaný súbor z cloudu.

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 človek s ktorým ten súbor zdieľame rozhodol ho zverejniť šifrovanie nám nepomôže, pretože už si mohol spraviť kópiu nešifrovanej verzie. Preto stačí revokovať prístup na cloude a nemusíme prešifrovať súbor.

Kapitola 4

Implementacia

V tejto kapitole budem popisovat hlavne implementacne detaily TODO

4.1 Použité technologie

Pozrieme sa aké technológie naša implementácia využíva na strane klienta a aké na strane servru a v jednoduchosti popíšeme prečo sme ich vybrali.

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 mongoose a ako cloudové riešenie využívame služby Google-Drive.
Písať v jednom jazyku front-end aj back-end už dnes vďaka node.js nie je problém. S pomocou webového frameworku Express vieme veľmi ľahko vytvoriť REST API a pridávanie middleware-u taktiež nie je žiadny problém. MongoDB sme sa rozhodli použiť koli dynamickej schéme, škálovateľnosti ale taktiež aj kôli možnosti pracovať s moongose, ktorý nám dáva možnosť pracovať s databázou ako s obyčajnými objektami čo výrazne uľahčuje vývoj. Google-Drive sme sa rozhodli využiť z dôvodu ktívneho využívania a taktiež

kôli dobrej dokumentácii API.

Front-end

Za front-end budeme považovať všetky veci ktoré sa dostanú k užívateľovi a budú zobrazované alebo vykonávane v jeho prehliadači. 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, napríklad na prekreslenie bez toho aby sme museli kontaktovať server a žiadať od neho novú verziu HTML , používame knižnicu jQuery ktorá ja jedna z najznámejších. Jej dokumentácia a developerská podpora je velmi rozsiahla. Knižnicu jQuery budeme taktiež využívať na AJAX volania pre ktoré nám poskytuje jednoduché API.
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 aj symetrickú tak aj asymetrickú kryptografiu a taktiež rôzne hašovacie funkcie.

4.2 Stanford Javascript Crypto Library

Opíšeme si bližšie aké šifry knižnica ponúka a predvedieme jednoduché ukážky funkčného kódu využívajúc SJCL.

Prečo SJCL?

Okrem toho, že nám poskytuje všetky kryptografické primitívy ktoré naše riešenie vyžaduje jej hlavnou výhodou je efektívna implementácia ktorá bola jej primárnym cieľom. V rýchlosti šifrovania je v priemere 4x rýchlejšia ako existujúce riešenia [8]. Kompatibilita medzi všetkými modernými prehliadačmi ako Chrom, Firefox, Safari a Internet Explorer v kombinácii s

rýchlostou a jednoduchým používatelským rozhraním boli kritickými prvkami pri výbere.

4.2.1 Ukážky

V tejto časti si spravíme krátke ukážky symetrického šifrovania a asymetrického šifrovania. Neskôr sa pozrieme ako sa dá pomocou SJCL hašovať alebo podpisovať. Ukážky sú dostupné aj vo wiki knižnice SJCL [10].

4.2.2 Symetrické šifrovanie

Symetrické šifrovanie je velmi jednoduché vďaka tomu ako je zaobalené. V štandardnom nastavení sa použije AES-128 v CCM móde. Pokiaľ je ako prvý parameter vložené heslo typu reťazec, knižnica predpokladá, že heslo nie je dostatočne dobré a použije naňho PBKDF2 s náhodnou soľou.

```
1  // sifrujeme retazec "text"
2  var sifrovany_text = sjcl.encrypt("heslo", "text");
3
4  // desifrujeme
5  var desifrovany_text = sjcl.decrypt("heslo",
       sifrovany_text);
6
7  console.log(sifrovany_text);
8  // ---> {"iv":"A0DOQPxWAlJ5LHjoyhGWcw==","v":1,"iter
       ":1000,"ks":128,"ts":64,"mode":"ccm","adata":"","
       cipher":"aes","salt":"Kk+ws1Xj0Xo=","ct":"
       text_v_zasifrovanom_tvate_bude_tu"}
9  console.log(desifrovany_text);
10  // ---> text
```

Listing 4.1: Symetrické šifrovanie

Je možnosť použiť rôznu dĺžku kľúčov, napríklad 128, 192 alebo 256 bitov. Mód šifrovania je tiež parametrizovaný a môžme sa rozhodnúť medzi štandardným CCM alebo volitelnými OCB2 a GCM.

4.2.3 Asymetrické šifrovanie

Asymetrická kryptografia je implementovaná pomocou eliptických kriviek a je nutné knižnicu skompilovať s parametrom $--with-ecc$ aby sme ju mohli využívať.

```
1  // vygenerujeme kluce
2  var kluce = sjcl.ecc.elGamal.generateKeys(256);
3  //popripade sjcl.ecc.elGamal.generateKeys(sjcl.ecc.
       curves.krivka) kde krivka je jedna z dostupnych
       kriviek
4
5  // sifrujeme retazec "text" pomocou verejneho kluca
6  var sifrovany_text = sjcl.encrypt(kluce.pub, "text");
7
8  // desifrujeme pomocou privatneho kluca
9  var desifrovany_text = sjcl.decrypt(kluce.sec,
       sifrovany_text);
10
11 console.log(desifrovany_text);
12  // ---> text
```

Listing 4.2: Asymetrické šifrovanie

Ak kľúče potrebujeme niekam posielať alebo ukladať tak občas sa stane, že ich treba serializovať. Našťastie toto už je implementované v SJCL takže to bez problémov zvládneme na pár riadkov.

```
1  var kluce = sjcl.ecc.elGamal.generateKeys(256);
2  var verejny = pair.pub.get();
3  var privatny = pair.sec.get();
4
5  // Serializujeme verejny kluc
6  verejny = sjcl.codec.base64.fromBits(verejny.x.concat
       (verejny.y));
7  console.log(verejny);
8  // ---> uQuXH/yeIpQq8hCWiwCTIMKdsaX...
9
10 // Deserializujeme verejny kluc:
```

```
11          verejny = new sjcl.ecc.elGamal.publicKey(
12            sjcl.ecc.curves.c256,
13            sjcl.codec.base64.toBits(verejny)
14          );
15
16
17          // Serializujeme privatny kluc
18          privatny = sjcl.codec.base64.fromBits(privatny);
19          console.log(verejny);
20          // ---> IXkJSpYK3RHRaVrd...
21
22          // Deserializujeme privatny kluc:
23          privatny = new sjcl.ecc.elGamal.secretKey(
24                sjcl.ecc.curves.c256,
25                sjcl.ecc.curves.c256.field.fromBits(sjcl.codec.
                     base64.toBits(privatny))
26          );
```

Listing 4.3: Serializácia

4.2.4 Hašovanie

Vytvorenie hašu je otázkou jedného riadku.

```
1           //zahashujeme
2           var hash = sjcl.hash.sha256.hash("zahasuj ma");
```

Listing 4.4: Hašovanie

Okrem SHA-256 je možné použiť aj SHA-1, SHA-512 alebo RIPEMD-160. V balíčku je štandardne iba prvá spomenutá ostatné treba pridať pri kompilácii pomocou parametru $--with-sha512$ respektíve $--with-ripemd160$.

4.2.5 Podpisovanie

```
1         var podpis_kluce = sjcl.ecc.ecdsa.generateKeys(256);
2
3         var podpis = pair.sec.sign(sjcl.hash.sha256.hash("
              dolezite data"));
4         console.log(podpis);
5         // [ 799253862, -791427911, -170134622, ...
6
7         var ok = pair.pub.verify(sjcl.hash.sha256.hash("Hello
              World!"), sig);
8         // vrati true alebo vyhodi error
```

Listing 4.5: Podpisovanie

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