SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-91764

ZDIEĽANIE TAJOMSTVA PRE SPRÁVU SÚKROMNÝCH ÚDAJOV

DIPLOMOVÁ PRÁCA

2022 **Peter Čuřík**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-91764

ZDIEĽANIE TAJOMSTVA PRE SPRÁVU SÚKROMNÝCH ÚDAJOV

DIPLOMOVÁ PRÁCA

Študijný program:	aplikovaná informatika
Názov študijného odboru:	informatika
Školiace pracovisko:	Ústav informatiky a matematiky
Vedúci záverečnej práce:	prof. Ing. Pavol Zajac, PhD.

Bratislava 2022 **Peter Čuřík**

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	aplikovaná informatika
Autor:	Peter Čuřík
Diplomová práca:	Zdieľanie tajomstva pre správu súkromných údajov
Vedúci záverečnej práce:	prof. Ing. Pavol Zajac, PhD.
Miesto a rok predloženia práce:	Bratislava 2022

Táto práca sa venuje Shamirovej schéme zdieľania tajomstva a jej použitiu v praxi. Rieši problémy uchovávania citlivých údajov v komerčných, cloudových úložiskách. Prináša softvérové riešenie: aplikáciu Datachest. Aplikácia komunikuje s troma cloudovými úložiskami: Google Drive, Microsoft OneDrive a Dropbox. Používateľ cez aplikáciu nahráva a sťahuje svoje dôverné dáta. Aplikácia ich na cloud nahráva v zašifrovanom tvare. Kryptografický kľúč rozdelí na tri podiely a každému cloudu odovzdá jeden podiel. Riešenie zvyšuje úroveň bezpečnosti pri správe osobných údajov na cloudových úložiskách. Využíva Shamirovu schému na zdieľanie tajomstva. Riešenie je efektívne, jednoducho použiteľné, spĺňa požiadavky bezpečnosti.

Kľúčové slová: secret sharing, súkromné údaje, cloud, bezpečnosť

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Peter Čuřík
Master's thesis:	Secret sharing for privacy
Supervisor:	prof. Ing. Pavol Zajac, PhD.
Place and year of submission:	Bratislava 2022

The topic of this thesis is the Shamir's secret sharing scheme and its practical usage. We developed a solution for storing sensitive data in commercial cloud storage systems. The implemented software is an application called Datachest. The application communicates with three storage providers: Google Drive, Microsoft OneDrive and Dropbox. Users can upload and download their confidential files. The application uploads data in an encrypted form. Cryptographic keys are split into three shares. Each cloud receives one share. This solution improves the security of user's sensitive data in the cloud by using Shamir's scheme to share a secret. The solution is efficient, user-friendly and meets security requirements.

Keywords: secret sharing, personal data, cloud, security

Poďakovanie

Rád by som poďakoval vedúcemu tejto práce, prof. Ing. Pavlovi Zajacovi, PhD., za spoluprácu. Pán profesor, ďakujem za odborné vedenie, priateľský prístup a otvorený dialóg počas celej doby výskumu. Bolo pre mňa cťou s Vami spolupracovať.

Obsah

Úvod

V súčasnej dobe vysokého rozmachu technológií a nárastu ich používania, je bežný používateľ[1] vystavený výzve spravovať svoje digitálne údaje a dáta. Ak vlastní bežne používané zariadenia dnešnej doby, ako smartfón a počítač, môže ho zaujímať riešenie na otázku, ako synchronizovať a zálohovať osobné údaje (napríklad heslá), či dáta (napríklad fotografie) týchto zariadení, aby sa k nim mohol dostať odkiaľkoľvek.

Jedným z riešení je využívanie cloudu. Cloud predstavuje službu poskytovania úložiskovej kapacity. Obsah cloudu je dostupný autorizovanej osobe nezávisle od zariadenia, z ktorého na cloud pristupuje. Cloud vlastní a poskytuje externý poskytovateľ. Používateľovi je pridelená časť úložiska, ktorú môže užívať za dohodnutý poplatok. Cloud rieši otázku synchronizácie, aj zálohovania dát.

Cloud reprezentuje tretiu stranu, ktorá spravuje súkromné dáta používateľa. Vzniká tu preto problém dôvery. Má používateľ veriť cloud poskytovateľovi, že jeho dáta sú chránené? Je používateľ jedinou entitou, ktorá pozná ich obsah? Podujali sme sa poskytnúť nástroj na (čiastočné) riešenie tohto problému, využívajúc tzv. secret sharing[2].

Secret sharing je kryptografická schéma, pomocou ktorej je používateľ schopný rozdeliť svoje údaje na viacero podielov. Každý z nich zverí inej cloud entite. Tieto entity sú vzájomne nezávislé (rôzne spoločnosti). Jednotlivé podiely tajomstva nedávajú držiteľovi zmysel. Pôvodné tajomstvo sa dá zrekonštruovať iba spojením dostatočného počtu podielov dokopy.

Pre používateľa, ktorý má záujem využívať cloud úložisko od poskytovateľa tretej strany, je toto potenciálnym riešením na dosiahnutie vyššej bezpečnosti jeho údajov.

Cieľom našej práce bolo implementovať aplikáciu, ktorá je určená pre bežného používateľa. Aplikácia používa secret sharing na rozdelenie dát používateľa na viacero častí. Každú z nich uloží na iný cloud. V prípade potreby ich z cloudov získa a zrekonštruuje ich pôvodný obraz spojením jednotlivých častí dokopy. V práci popisujeme riešenie od analýzy, cez návrh aplikácie, až po výslednú softvérovú implementáciu. Na záver práce hodnotíme prínosy aplikácie z hľadiska bezpečnosti a použiteľnosti.

[1] pojmom „bežný používateľ" explicitne označujeme koncového používateľa softvéru, pri ktorom predpokladáme iba základné vedomosti v oblasti informačných technológií.

[2] v preklade tiež: zdieľanie tajomstva.

1 Základné pojmy

Pre lepšie porozumenie problematiky tejto práce bližšie popíšeme niektoré pojmy, s ktorými budeme pracovať.

1.1 Secret sharing

Secret sharing je schéma pôvodne určená pre správu kryptografických kľúčov [1]. Prevenciou proti strate kľúčov je vytvorenie viacerých kópií uložených na rôznych úložiskách. Tento prístup obnáša riziko odhalenia tajomstva. S nižším počtom kópií zase vzniká riziko straty tajomstva. Secret sharing sa venuje tejto problematike.

Základná myšlienka schémy secret sharing je rozdeliť tajomstvo S na viaceré časti, nazývané ako *shares*[3]. Jednotlivé shares sa rozdelia medzi definovaný zoznam účastníkov $\mathcal{P}$ tak, že pri spojení určitého počtu shares je možné zrekonštruovať pôvodné tajomstvo S. Súčasne platí, že spojenie nedostatočného počtu shares neumožňuje rekonštrukciu tajomstva a zároveň o ňom prezrádza nulovú informáciu.

Podmnožinou secret sharing schém sú tzv. *threshold schémy*. Neformálnou definíciou threshold schémy bližšie opíšeme vyššie uvedené úvodné myšlienky.

Definícia 1 *[2] Nech $t, n \in \mathbb{Z}^+$. Threshold schéma (t, n), $t \leq n$ je metóda, pri ktorej z pôvodného tajomstva S je vytvorených n častí S_i, $1 \leq i \leq n$ rozdelených medzi n účastníkov $P_i \in \mathcal{P}$. Platí, že akákoľvek skupina t a viac účastníkov dokáže zrekonštruovať S, ale skupina s počtom $t-1$ a menej účastníkov nie.*

Hodnota S je volená špeciálnym účastníkom. Označujeme ho ako *dealer*[4], D. Predpokladáme $D \notin \mathcal{P}$. D tajne doručí jednotlivé shares medzi P_i tak, aby žiadny z účastníkov nepoznal share iného účastníka.

1.2 Shamirova schéma na zdieľanie tajomstva

Shamirova schéma na zdieľanie tajomstva („Shamir's secret sharing scheme") je jednou z prvých threshold schém. Formuloval ju izraelský kryptograf Adi Shamir v roku 1979 [3]. Schéma je založená na interpolácii polynómov. Dealer zvolí vhodný polynóm z konečného poľa a na jeho základe získa shares, ktoré rozdistribuuje medzi účastníkov. Pre polynóm $y = f(x)$ stupňa $t-1$ platí, že je jednoznačne definovaný t bodmi (x_i, y_i), s odlišným x_i.

[3] v preklade tiež: podiely.
[4] v preklade tiež: distribútor.

Postup vytvárania tajomstva a vypočítania jednotlivých shares v Shamirovej schéme môžeme definovať v jednotlivých krokoch.

1. Dealer D vygeneruje tajomstvo S, pričom $S \geq 0$.
2. D volí prvočíslo p také, že $p > max(S, n)$, kde n je počet účastníkov.
3. D priradí $a_0 = S$.
4. D náhodne vygeneruje $t-1$ koeficientov $a_1, ..., a_{t-1}$, pričom $0 \leq a_j \leq p-1$, čím definuje náhodný polynóm nad poľom $\mathbb{Z}_p$, v tvare $f(x) = \sum_{j=0}^{t-1} a_j x^j$
5. D vypočíta shares ako $S_i = f(i) \bmod p$, pričom $1 \leq i \leq n$
6. D tajne pošle pár (i, S_i) účastníkovi P_i.

Pre rekonštrukciu pôvodného tajomstva je potrebných aspoň t bodov $(x, y) = (i, S_i)$. Z nich je možné vypočítať koeficienty a_j polynómu $f(x)$ pomocou Lagrange-ovej interpolácie, pričom $1 \leq j \leq t-1$. Pripomenieme, že v tejto fáze žiadny účastník P_i nepozná polynóm $f(x)$. Pre koeficienty neznámeho polynómu $f(x)$ stupňa nižšieho, než t, definovaného bodmi (x_i, y_i), pričom $1 \leq i \leq t$, platí vzorec Lagrange-ovej interpolácie:

$$f(x) = \sum_{i=1}^{t} y_i \prod_{1 \leq j \leq t, j \neq i} \frac{x - x_j}{x_i - x_j}$$

Využitím vyššie uvedeného vzťahu vieme zrekonštruovať pôvodný polynóm $f(x)$. Pre získanie tajomstva S využijeme znalosť, že $S = a_0 = f(0)$. Teda:

$$S = \sum_{i=1}^{t} c_i y_i, \qquad \text{kde } c_i = \prod_{1 \leq j \leq t, j \neq i} \frac{x_j}{x_j - x_i}$$

Príklad. Pre účely demonštrácie je nasledujúci príklad v poli $\mathbb{R}$. Nakoľko $\mathbb{R}$ nie je konečné pole, dealer nebude voliť prvočíslo p.

1. Dealer D vygeneruje tajomstvo $S = 1$.
2. D volí paremetre threshold schémy (t, n) nasledovne:
   $t = 3$ (počet potrebných účastníkov na získanie S),
   $n = 5$ (počet všetkých účastníkov schémy).
3. D priradí $a_0 = S$.

4. D náhodne generuje $t-1$ koeficientov $a_1 = -2, a_2 = 1$.
   $f(x) = a_0 \cdot x^0 + a_1 \cdot x^1 + a_2 \cdot x^2$ je vzniknutý polynóm stupňa $t-1$.
   $f(x) = 1 + (-2) \cdot x + 1 \cdot x^2$ je tvar polynómu po dosadení hodnôt.
   $f(x) = (x-1)^2$ je výsledný tvar polynómu po algebraickej úprave.

5. D vypočíta shares $S_i = f(k)$, pričom $1 \leq i \leq n$, $k \in D_f$ nasledovne:
   $S_1 = f(0.3) = 0.5$,
   $S_2 = f(1) = 0$,
   $S_3 = f(1.5) = 0.25$,
   $S_4 = f(2) = 1$,
   $S_5 = f(2.2) = 1.5$.

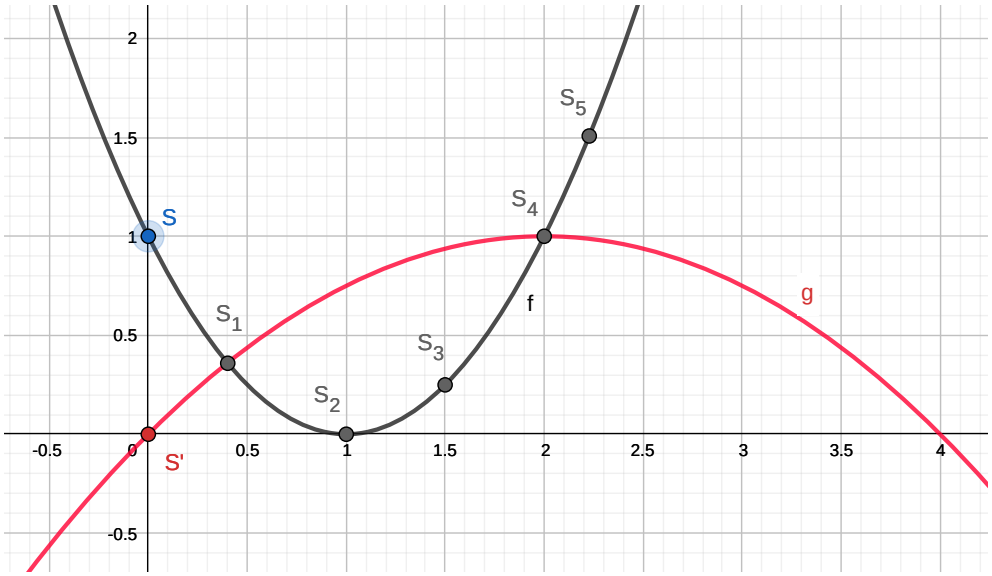
6. D odovzdá share S_i účastníkovi P_i.

Na obrázku 1 je znázornené, ako funkcia f definuje tajomstvo a jeho shares. Tajomstvo S je bod karteziánskej sústavy súradníc $f(0) = (0, 1)$, respektíve priesečník funkcie f s osou y. Jednotlivé shares S_i ležia na funkcii f, rovnako ako S. Táto skutočnosť implikuje fakt, že s dostatočným množstvom známych bodov S_i je možné funkciu f zostrojiť, za predpokladu nulovej informácie o f (Lagrange-ova interpolácia).

Pre jedného účastníka P_i je na rekonštrukciu tajomstva nedostatočný jeho S_i. Pre jeden bod existuje nekonečne veľa kvadratických funkcií $a_2x^2 + a_1x + a_0$, pre ktoré platí, že $S, S_i \in H_f$. Analogicky platí to isté pre dvoch účastníkov P_i, P_{i+1}, ktorí disponujú S_i, S_{i+1}.

Predpokladajme, že účastníci P_1, P_4 by chceli získať znalosť o S. P_1 začne zdieľať S_1 s P_4 a analogicky P_4 urobí to isté. Zostroja funkciu g, ktorá prechádza osou y a body S_1, S_4 na nej ležia. Funkcia g je potenciálnou funkciou na získanie správneho tajomstva. Na obrázku 1 vidíme, že funkcia g určuje $S' = 0$. Teda neplatí $S' = S$. Dá sa ukázať, že útočníci P_1, P_4 majú stále nulovú informáciu o S: akékoľvek S' by zvolili, mohli by bodmi S', S_1 a S_4 viesť parabolu. Teda, potenciálne tajomstvo S môže byť ľubovoľné.

Pre troch a viacerých účastníkov už platí, že existuje práve jedna kvadratická funkcia $a_2x^2 + a_1x + a_0$, pre ktorú platí, že $f(0) = S$. Pomocou Lagrange-ovej interpolácie sú schopní taký polynóm zostrojiť. Vidíme, že pre polynóm stupňa 2 platí $t = 3$, ako sme aj uviedli na začiatku podkapitoly, a teda: Pre polynóm $y = f(x)$ stupňa $t-1$ platí, že je jednoznačne definovaný t bodmi (x_i, y_i), s odlišným x_i.



Obr. 1: Secret sharing: Grafická reprezentácia príkladu v poli $\mathbb{R}$.

2 Bezpečnosť cloudov a secret sharing

V tejto kapitole sa venujeme bezpečnosti cloudov a rizikám, ktoré obnáša ich používanie. Ďalej analyzujeme bezpečnosť secret sharingu, jeho použitie a bezpečnostné vylepšenia voči cloudom. V závere kapitoly analyzujeme niektoré existujúce riešenia, ktoré implementujú secret sharing schému.

2.1 Bezpečnosť cloudov

V súčasnosti evidujeme vysokú mieru používania cloudu bežnými používateľmi na účely ukladania osobných údajov. Podľa [4] používalo cloud služby 3,6 miliárd internetových používateľov k roku 2018. To je oproti roku 2013 (2,4 miliárd používateľov) nárast o jednu tretinu. Prieskum z roku 2020 [5] sa na vzorke 648 respondentov venuje spôsobom používania cloudu. Najpoužívanejšie cloud služby na ukladanie dát sú Google Drive, Microsoft OneDrive a Dropbox. 65% z nich používa cloud ako primárne dátové úložisko, nie ako zálohu. 73% respondentov používa tieto služby na ukladanie svojich osobných údajov napriek faktu, že pre 78% je najväčším rizikom týchto služieb súkromie a bezpečnosť údajov.

Cloud služby poskytujú používateľovi mnoho funkcionalít, ktoré majú zvýšiť pohodlnosť používania. Napríklad, Dropbox poskytuje automatickú synchronizáciu lokálnych súborov, aplikácie pre viaceré zariadenia pre jednoduchú dostupnosť údajov, dátovú bezpečnosť, zdieľanie súborov s inými používateľmi a podobne [6].

Pri entitách spravujúce súkromné údaje používateľov je potrebné uvažovať problém dôvery. A to vo všetkých aspektoch CIA (z angl. Confidentiality, Integrity, Availability):

- dôvera v oblasti dôvernosti súkromných údajov

- dôvera v oblasti integrity súkromných údajov

- dôvera v oblasti dostupnosti súkromných údajov

Poskytovatelia cloud služieb riešia rôzne bezpečnostné problémy a hrozby. Niektoré z nich je možné riešiť. Napríklad Google Drive adresuje aspekt dôvernosti CIA pomocou šifrovania údajov šifrou AES s 256 bitovým kľúčom [7]. Aspekt dostupnosti CIA je adresovaný vybudovaním viacerých dátových centier [8]. Takýto krok má zabezpečiť životnosť dát a neustálu dostupnosť.

Napriek podloženým garanciám cloud poskytovateľov (certifikáty od auditorov a podobne) môže mať používateľ nepodložené obavy vo veci dôvery voči týmto entitám (viď rebríček najväčších obáv respondentov pri používaní cloudu pre účely ukladania osobných údajov v [5]). Pre každý aspekt CIA uvedieme pár príkladov.

Dôvernosť: spochybňovanie spôsobov spracovávania súkromných údajov, spochybňovanie využívania súkromných údajov na účely umelej inteligencie, obavy, že súkromné údaje sú využívané na obchodovanie, obavy potenciálnych zmien právnych predpisov, ktoré upravujú metódy ochrany údajov používateľov v digitálnom svete, v neprospech používateľa.

Integrita: spochybňovanie bezpečnosti prudko narastajúceho cloudového systému voči útokom na integritu dát, obavy o integritu osobných údajov počas nahrávania na cloud, pri ich spracovávaní na cloude, pri ich migrácii do úložísk tretích strán, s ktorými cloudový poskytovateľ môže spolupracovať.

Dostupnosť: DDoS útoky, fyzické ohrozenie dátových centier (požiar, zemetrasenie, povodeň, výpadok elektrickej energie), spochybňovanie prístupu cloudového poskytovateľa k zodpovednosti správy súkromných údajov klientov (nedostatočné zálohovanie údajov, nedostatočná údržba hardvéru v dátových centrách, iné neobnoviteľné straty dát).

Tieto obavy môžu podporiť verejne dostupné príklady zlyhania dátových centier ako napríklad [9], [10], či [11].

Cloudové služby poskytujú vysokú mieru pohodlnosti používania (ako napríklad automatické zálohovanie, synchronizácia medzi viacerými zariadeniami). Napriek vysokým bezpečnostným štandardom a prevenciám existuje riziko zlyhania. Vyššie uvedené zdroje dokladujú, že riziká vedia byť dosť vysoké na to, aby prišlo ku škodám na osobných úda-

joch. Používatelia môžu preto považovať cloud za nie dostatočne dôveryhodný priestor na správu ich súkromných údajov.

2.2 Bezpečnosť secret sharingu

Bezpečnosť secret sharingu závisí od správneho nastavenia schémy a jej parametrov. Jedným z nich je výber účastníkov schémy. Musí ísť o také entity, pri ktorých nepredpokladáme spoluprácu dostatočne veľkého počtu účastníkov t. Taký prípad by znamenal odhalenie tajomstva S. Entity by medzi sebou nemali mať interné vzťahy. Výhodou je, ak si vzájomne konkurujú. V takom prípade by vzájomné spájanie nebolo výhodné a je vysoko nepravdepodobné. Navyše, takýto návrh od konkurenčnej strany môže znieť nedôveryhodne a nemusí byť preto akceptovaný druhou stranou. Strana, ktorá by takýto návrh predložila, riskuje odhalenie stranou, ktorej spojenie ponúka (nakoľko si vzájomne konkurujú) a teda môže čeliť vyradením zo schémy, prípadne sankciám, ak sa nejaké definovali v dohode schémy.

Nevhodné zvolenie parametrov (t, n) môže vytvárať bezpečnostné hrozby. Ľahko vieme ukázať, že parameter $n = 1$ nie je bezpečné nastavenie schémy. Pri voľbe $n = 2$ musí platiť rovnosť $n = t$, inak by každý účastník schémy vedel získať pôvodné S. Príliš vysoké n alebo t môže mať za následok problémy s dostupnosťou.

V porovnaní so symetrickým šifrovaním, kde je zodpovednosť koncentrovaná na jeden objekt (kľúč), poskytuje secret sharing distribúciu tejto zodpovednosti. Odcudzenie jednej časti tajomstva prezrádza o pôvodnom tajomstve nulovú informáciu a dealer D je stále schopný zrekonštruovať pôvodné tajomstvo zo zvyšných shares (ak $t \neq n$). Na druhej strane, odcudzenie kľúča by znamenalo úspech bezpečnostného útoku.

Z hľadiska CIA platia pre secret sharing nasledujúce tvrdenia:

Dôvernosť: Menšia ako šifrovanie. Prístup k dátam je udelený skupine entít. Pri šifrovaní je dôvera redukovaná na kľúč. Ak by dealer namiesto secret sharing schémy použil šifrovanie, dáta vložil na cloud a kľúč uschoval u seba, išlo by o bezpečnejšie riešenie. Treba však podotknúť, že secret sharing mení model dôvery: aj keď entity nie sú považované za dôveryhodné, nepredpokladáme, že budú pri útoku spolupracovať.

Integrita: Platí podobná úvaha ako pri analýze Dôvernosti.

Dostupnosť: Lepšia ako šifrovanie. Distribúciou shares medzi viaceré, nezávislé úložiská je možné tolerovať nedostupnosť až $n - t$ úložísk. V prípade šifrovania je kľúč iba jeden. Nedostupnosť úložiska, na ktorom sa tajomstvo (kľúč) nachádza, spôsobuje nedostupnosť kľúča samotného. Možným riešením tohto problému je dátová redundancia. To však spôsobuje riziko odcudzenia kľúča, keďže je uložený na viacerých úložiskách naraz.

Tento problém adresuje secret sharing (viď sekcia 1.1).

2.3 Prehľad existujúcich riešení

Pred návrhom a implementáciou nášho riešenia bol vykonaný prieskum existujúcich riešení. Cieľom prieskumu je vyvarovať sa duplicitnému riešeniu, nájsť nástroje, ktoré by uľahčili vývoj našej implementácie a získať prehľad o možných prístupoch k riešeniu problému. Vybrali sme tri existujúce riešenia. Shamirova schéma nie je v súčasnosti vo veľkom využívaná v praktickej rovine, preto majú spomínané riešenia skôr vedecký charakter.

2.3.1 Framework Archistar

Archistar je open-source framework, písaný v jazyku Java [12]. Ide o výskumný projekt. Kód je určený najmä pre middleware vrstvu: server prijme správu, zašifruje ju a jej shares distribuuje medzi nezávislé backend servery (viď Obr. 2). Repozitár zaznamenáva posledné zmeny v kóde pred ôsmimi rokmi. V súčasnosti je repozitár archivovaný a je prístupný iba v móde na čítanie (read-only). V repozitári je avšak uvedené, že sa momentálne pracuje na novej generácii softvéru. Jedna z vecí, ktorá ešte nie je implementovaná, je rozhranie pre koncového používateľa, pomocou ktorého by už bolo možné ukladať dáta do systému pomocou secret sharingu.

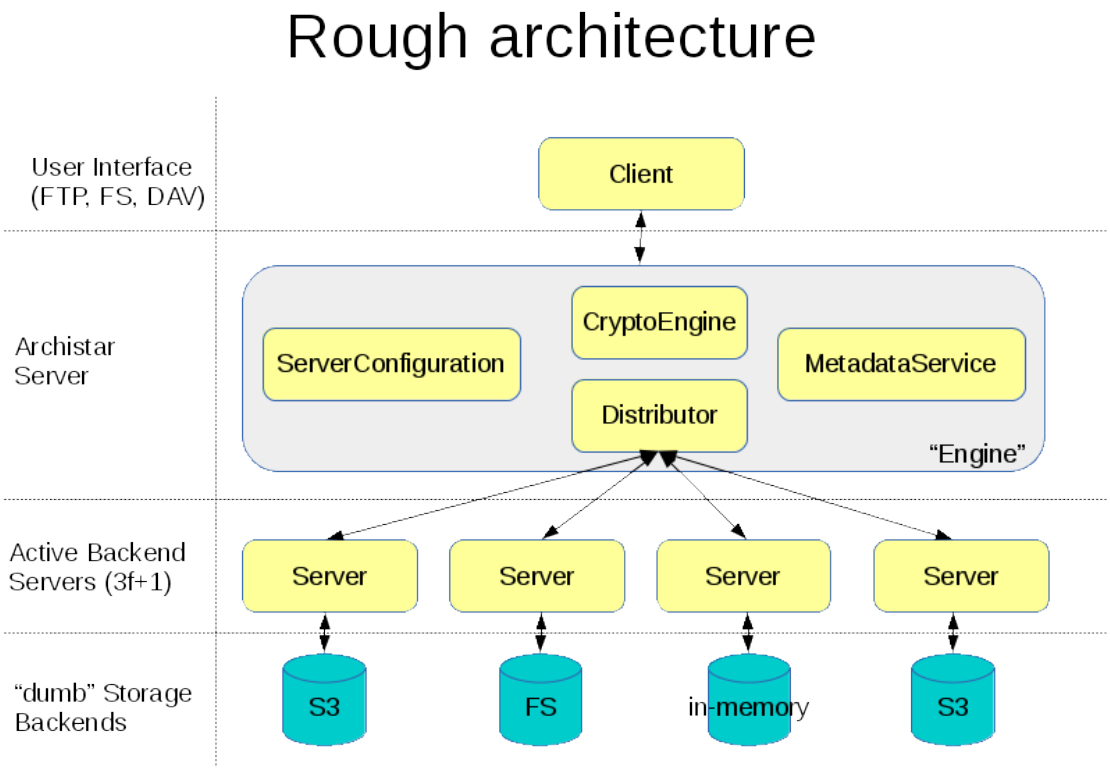
Práca [13] sa venuje vylepšeniu framework-u Archistar. Zámer práce je: *„Výskum uvedený v tejto publikácii sa zaoberá výzvou, ktorá spočíva v tom, aby konfigurácia systému Archistar bola použiteľná pre technicky zdatných používateľov vrátane správcov systému pri súčasnom splnení požiadaviek na súkromie, bezpečnosť, dôveru a ďalšie organizačné požiadavky."*

V skratke, navrhované riešenie spočíva v implementácii rozhrania, cez ktoré bude možné konfigurovať systém. V konfigurácii si používateľ môže nastaviť bezpečnostné parametre a ďalšie možnosti systému, ktoré sa následne naň aplikujú.

2.3.2 PRISMACLOUD

Príklad softvérového riešenia, ktoré implementovalo do svojej štruktúry framework Archistar, je PRISMACLOUD. Opäť ide o vedecký projekt. Primárnym cieľom je poskytnúť bezpečnosť pre používateľov cloudu pomocou kryptografie [14]. Nejde o konečný produkt, ale o sadu nástrojov, pomocou ktorých je možné konštruovať cloudové systémy.

Z tohoto usudzujeme, že spomínaný balík nástrojov je určený pre veľké systémy a komplexné softvérové riešenia. PRISMACLOUD na svojej stránke aj uvádza niekoľko príkladov ich cieľových, koncových používateľov; zdravotné systémy, poskytovatelia cloudu, ktorí majú záujem poskytovať dátam svojich klientov vyššiu bezpečnosť, bezpečné úlo-



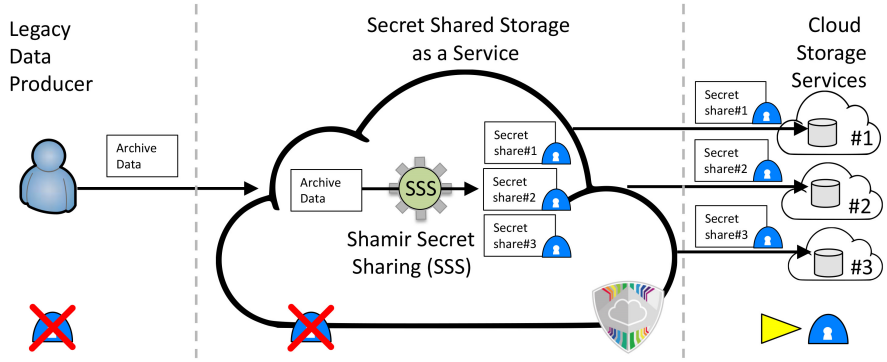
Obr. 2: Rola frameworku Archistar v komplexnom systéme.

žiská záznamov z bezpečnostných kamier v mestách a podobne.

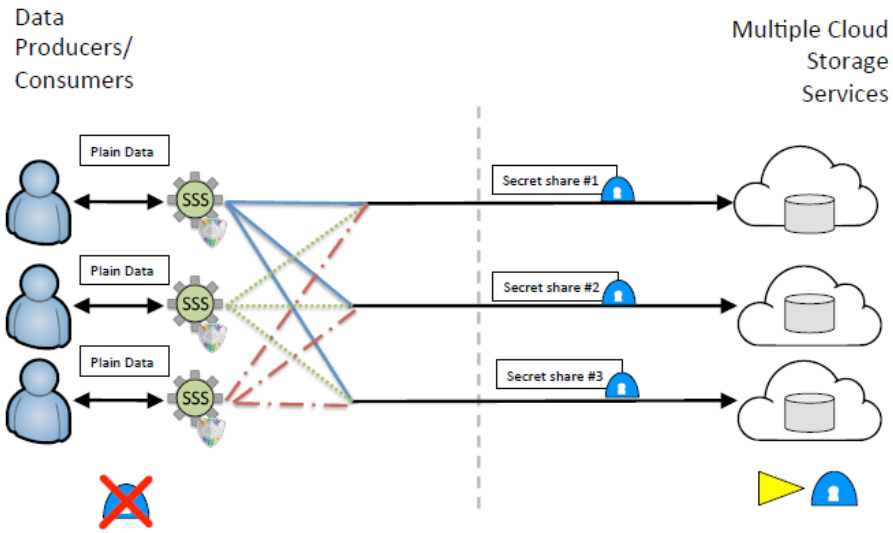
PRISMACLOUD definuje osem rôznych služieb (tzv. PRISMACLOUD services). Daná služba je kombináciou určitých kryptografických nástrojov a poskytuje takú sadu funkcionalít, ktoré sú aplikovateľné na mnoho praktických scenárov, na ktoré služba cieli. Na účely demonštrácie ukážeme niektoré služby, ich stručný opis a účel.

Secure Archiving Service: služba je aplikovateľná do scenárov zálohovania dát na cloude. Súbory sú rozdelené do viacerých shares na zariadení používateľa. Shares sú rozdistribuované medzi viacerých cloud poskytovateľov. Dostupnosť je možné zvýšiť, ak na rekonštrukciu pôvodného súboru nie sú potrebné všetky časti a poskytovateľovi cloud služieb už nie je potrebné dôverovať.

Data Sharing Service: služba umožňuje viacerým participantom ukladať dáta bezpečným spôsobom v sieti cloudu tak, že žiadne z úložísk nepozná formu dát v otvorenom tvare (plaintext). Týmto je umožnená bezpečná spolupráca medzi jednotlivými stranami bez nutnosti dôverovať jedinému cloud poskytovateľovi.



Obr. 3: Secure Archiving Service. PRISMACLOUD.



Obr. 4: Data Sharing Service. PRISMACLOUD.

2.3.3 Android aplikácia na ukladanie fotografií na cloud pomocou secret sharing mechanizmu

Existuje Android aplikácia ako výsledok vedeckej publikácie, ktorá dokáže pomocou techniky secret image sharing ukladať fotografie medzi tri nezávislé cloud úložiská [15]. Cieľom tohto riešenia je bežný používateľ. Cloud úložiská, ktoré sú k dispozícii, sú verejné, komerčné, určené pre bežných používateľov (v tomto príapde sa jedná o Dropbox, Firebase a Google Drive). Používateľ si nainštaluje aplikáciu do Android zariadenia. Po spustení sa prihlási do účtov všetkých cloud úložísk. Následne môže začať s ukladaním svojich fotografií. Aplikácia na pozadí rozdelí súbor na viacero častí a distribuuje ich medzi jednotlivé úložiská. Nerobí to pomocou Shamirovej secret sharing schémy, ale používa secret image sharing podľa Chih-Ching Thien a Ja-Chen Lin [16]. Ten pracuje s obrázkami ako s pixelmi a aplikuje na ne rôzne transformácie. Okrem secret image sharingu je

aplikovaná na obrázok šifrová schéma podľa autora menom Al-Husainy [17]. Aj v tomto prípade sa jedná o špecifickú kryptografiu, zameranú na obrázky (manipulácie s pixelmi).

2.4 Sumarizácia

U komerčných cloudových služieb zaznamenávame vysokú popularitu. Napriek tomu má mnoho používateľov obavy o bezpečie ich dôverných dát uložených na cloudoch. Bezpečnosť cloudov je po technickej aj teoretickej stránke na vysokej úrovni. Napriek tomu sme vedeli nájsť udalosti, ktoré ukazujú zostávajúce nedostatky v bezpečnosti cloudových úložísk a datacentier vo všeobecnosti.

Secret sharing má pri správnom zvolení parametrov schémy a správnom zvolení množiny účastníkov výhody oproti šifrovaniu. Secret sharing tak vie zvýšiť úroveň bezpečnosti osobných údajov používateľov v cloudových úložiskách.

Z podkapitoly 2.3 vyplýva, že v súčasnosti nemáme znalosť o implementácii, ktorá by riešila problém definovaný v tejto práci. Archistar a PRISMACLOUD sú technológie určené pre robustné systémy používané veľkými spoločnosťami, či dokonca štátnymi inštitúciami. Android aplikácia na správu fotografií taktiež nerieši nami definovaný problém. Nepoužíva Shamirovu schému na zdieľanie tajomstva. Je limitovaná na súbory rastrového charakteru. Používateľské rozhranie nespĺňa požiadavky.

Nami požadovaná implementácia by mala byť použiteľná a prehľadná pre bežného používateľa. Mala by podporovať najviac používané cloudové úložiská súčasnosti. Očakávame podporu všetkých známych typov súborov: dokumenty, obrázky, videá, hudba a iné. Implementáciu detailne opisujeme v nasledujúcich kapitolách.

3 Návrh riešenia

Na základe predstavených skutočností, faktov a úvah sme navrhli softvérové riešenie s názvom *Datachest*. Účelom je poskytnúť používateľovi správu nad jeho osobnými údajmi. Používateľ má byť cez aplikáciu schopný dáta bezpečne ukladať na komerčné cloudové úložiská a spravovať ich.

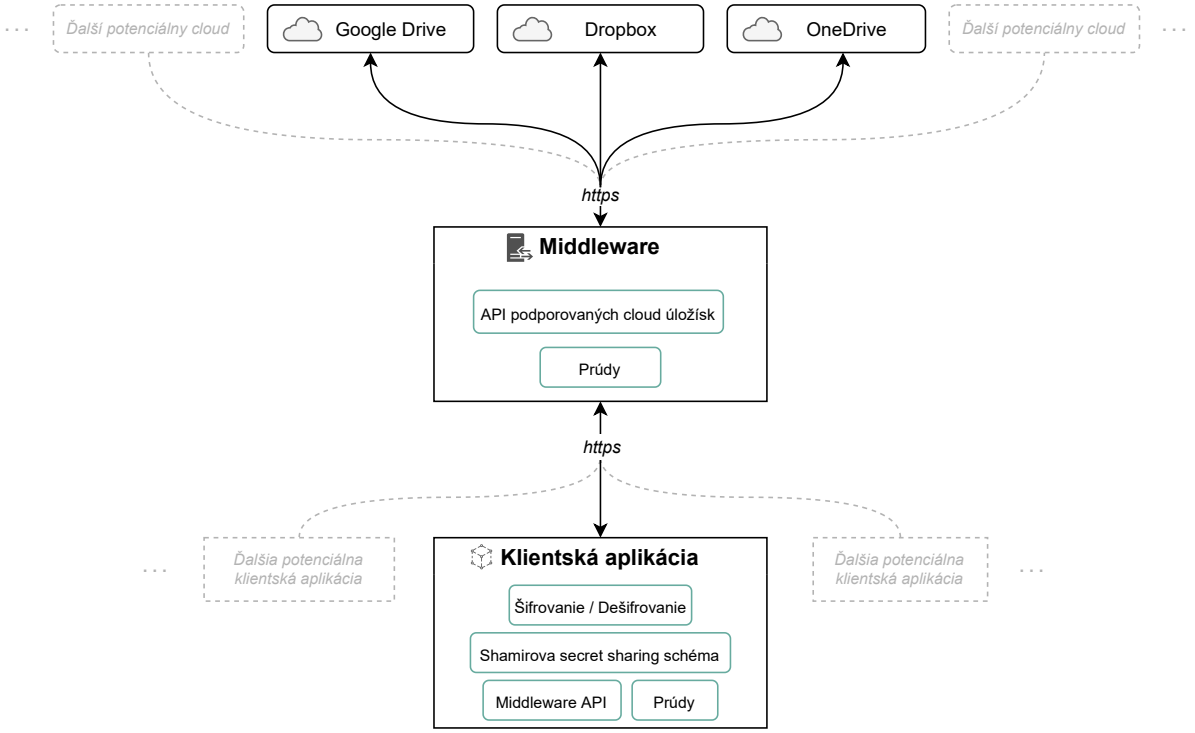
Boli vybrané tri úložiská, ktoré sú aplikáciou podporované: Google Drive, Microsoft OneDrive a Dropbox. Dôvodom tohoto výberu je ich vysoká popularita medzi používateľmi. Z prieskumu [5] vyplynulo, že patria medzi tri najpoužívanejšie cloudové úložiská: Google Drive používa 94.44% opýtaných, Dropbox používa 66.20% používateľov a Microsoft OneDrive je na tretej pozícii s popularitou 39.35%.

3.1 Konceptuálny návrh

Riešenie má implementovať dva komponenty: klientskú aplikáciu a middleware server (ukázané na obr. 5). Používateľ priamo interaguje s klientskou aplikáciou. V aplikácii má byť schopný vložiť svoje dáta pre účely ich bezpečnej distribúcie na cloudové úložiská. Aplikácia pri odosielaní dát alebo pri ich získavaní komunikuje na pozadí s middleware serverom. Ten je zodpovedný za komunikáciu s jednotlivými cloudami. Teda, aplikácia nikdy nekomunikuje s cloudami priamo. Namiesto toho komunikuje s middleware-om, ktorý preberie vedenie pri správe dát na cloudových úložiskách.

Dôvodom existencie middleware serveru v návrhu je snaha o ľahkú škálovateľnosť riešenia. S každou ďalšou pribudnutou klientskou aplikáciou (pre rôzne platformy ako PC, iOS, Android, Web a podobne) je ušetrená redundantná implementácia API komunikácie s každým cloud úložiskom. Namiesto toho stačí, aby každá aplikácia komunikovala s middleware-om, ktorý disponuje API logikou všetkých cloud úložísk. Množstvo ušetrenej redundancie v kóde sa zvyšuje aj pri zvyšovaní počtu podporovaných cloudov. Ak by v budúcnosti bolo toto softvérové riešenie rozšírené o ďalšie podporované úložiská, stačilo by API novo pridaných cloudov dodať iba middleware-u a nie každej klientskej aplikácii osobitne. Navyše, očakávame možné zmeny API špecifikácií jednotlivých cloudov v budúcnosti. V takom prípade by bolo nutné zmeniť iba kód middleware-u. Kódy klientských aplikácií by mohli ostať bez zmien.

Moduly vymenované v komponentoch na obr. 5 bližšie opíšeme v nasledujúcej podkapitole.



Obr. 5: Konceptuálny návrh softvérového riešenia Datachest.

3.2 Architektonický návrh

Architektonický návrh definuje, aké nástroje a techniky majú byť v riešení použité a zodpovedá otázky, prečo sú tieto elementy potrebné a aké problémy riešia.

Kľúčovým prvkom aplikácie je secret sharing a jeho použitie za účelom správy osobných údajov používateľa. Používateľ má byť schopný do aplikácie vložiť súbory. Tie majú byť bezpečne uložené na cloudové úložiská s použitím secret sharing schémy podľa Shamira.

Jedna z otázok, ktorú je potrebné zodpovedať pri vytváraní architektonického návrhu, je voľba vhodného programovacieho jazyka, knižníc, prípadne framework-ov na naplnenie požiadaviek riešenia. Pre klientskú aplikáciu závisel výber programovacieho jazyka od jeho niektorých možností a schopností. Prvou z nich je tvorba mobilných, webových, či desktopových aplikácií. Ďalšou je disponovanie knižnicami, ktoré implementujú Shamirov secret sharing algoritmus a niektoré kryptografické nástroje. Na základe týchto podmienok boli zvolené jazyky Swift (mobilné aplikácie), TypeScript (webové aplikácie) a Java (desktopové aplikácie).

Pre vyššie uvedené jazyky boli vybrané knižnice poskytujúce implementáciu Shamirovho secret sharing algoritmu. Implementácie boli podrobené testom. Cieľom bolo

pozorovanie správnej funkčnosti a efektivity implementácií. Pre žiadny z jazykov nebolo možné dosiahnuť prijateľnú časovú efektivitu. Tabuľka 1 zobrazuje namerané hodnoty v jednotkách času potrebných na rozdelenie súboru o veľkosti 1MB na 3 shares. Výsledky nie sú vyhovujúce. Dôvodom je najmä fakt, že rozsah možných veľkostí súborov vkladaných používateľom do aplikácie nie je presne určený. Platí, že časová náročnosť stúpa s veľkosťou súboru. Faktor času neumožňuje použitie secret sharing algoritmu pre všetky veľkosti súborov. S rastúcim časom dochádza ku značnému zásahu do praktickej použiteľnosti softvéru - pre každú platformu v inom bode.

Programovací jazyk	**Rozdelenie tajomstva**	**Použitá knižnica**
Java	0.217 sekundy	com.codahale.shamir
Swift	33.488 sekúnd	SwiftySSS
TypeScript	3.283 hodín	shamirs-secret-sharing-ts

Tabuľka 1: Časové merania rôznych implementácií rozdelenia tajomstva o veľkosti 1MB na 3 shares pomocou Shamirovho secret sharing algoritmu.

Z toho dôvodu bolo do architektonického návrhu pridané šifrovanie a dešifrovanie, ako znázorňuje obr. 6. Rola tajomstva bola prenesená z osobného údaju na 128 bitový kľúč, ktorým je údaj šifrovaný. Týmto krokom je možné zúžiť množinu všetkých rôznych veľkostí tajomstiev na množinu M[5], ktorej prvky sú časovo realizovateľné Shamirovým algoritmom. Z definovaného rozsahu vieme vyhodnotiť, že časová náročnosť je akceptovateľná. Samotný súbor už nie je nutné rozdeľovať, lebo je šifrovaný. Namiesto jednotlivých súborov budú rozdeľované kryptografické kľúče.

Čitateľ si môže položiť otázku, či pridaný prvok symetrickej kryptografie nezhorší celkovú časovú náročnosť do opäť neprijateľných hodnôt. V súvislosti s touto problematikou sme podrobili rovnakým testom aj operáciu šifrovania tajomstva a výsledky zapísali do tabuľky 2. Výsledky sú uspokojivé a prijateľné. Opäť je nutné brať do úvahy rôzne veľké vstupy dát. Rovnako ako pri secret sharingu, aj pri šifrovaní šifrou AES platí, že časová zložitosť narastá so zväčšujúcim sa vstupom. Avšak tu platí, že aj pri veľkých vstupoch sú časové hodnoty prijateľné a spĺňajú požiadavky architektonického návrhu riešenia Datachest.

[5]v našom prípade $M = \{128\}$ s mohutnosťou 1. V prípade algoritmu AES sa dá uvažovať aj o $M = \{128, 192, 256\}$ (hodnoty v bitoch).

Programovací jazyk	Šifrovanie tajomstva	Použitá knižnica
Java	0.001 sekundy	org.apache.commons.crypto
Swift	0.004 sekundy	CryptoKit
TypeScript	0.064 sekundy	crypto-js

Tabuľka 2: Časové merania rôznych implementácií šifrovania tajomstva o veľkosti 1MB pomocou algoritmu AES-GCM-128.

3.2.1 Klientská aplikácia

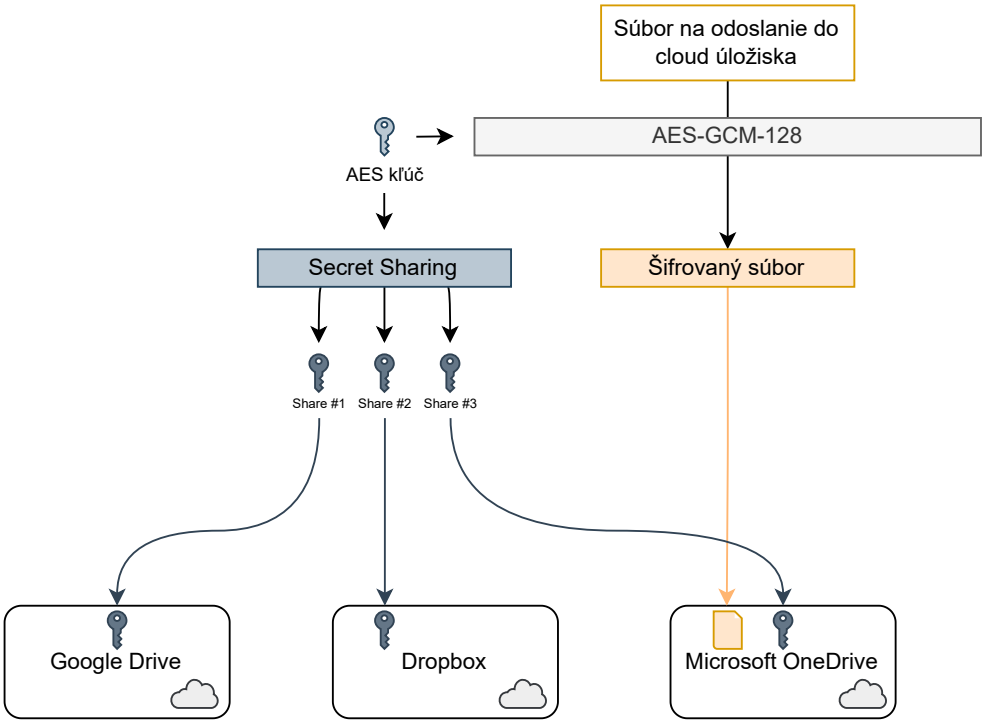
Klientská aplikácia zastupuje v Shamirovej schéme rolu dealera, resp. distribútora. Nakoľko táto práca hľadá riešenie, ktoré je pochopiteľné a ľahko použiteľné bežným používateľom, klientská aplikácia má dosiahnuť požiadavky UI a UX. Rozhranie má byť prehľadné a intuitívne. Malo by rovnako informovať používateľa o dôležitých procesoch, ako napríklad: stav sťahovania, stav nahrávania, výskyt zlyhaní a chýb, stav prihlásenia do účtov a podobne.

Klientská aplikácia musí spĺňať požiadavky bezpečnosti, pretože pracuje s citlivými dátami používateľa. Mala by implementovať bezpečné kryptografické algoritmy, vhodne spravovať kľúče, zabezpečene komunikovať s middleware serverom, ochrániť aplikáciu zámkom (heslo, PIN, biometria a iné) a podobne.

Pri nahrávaní a sťahovaní súboru sa vstup načíta do vyrovnávacej pamäte zariadenia. Tento fakt vyžaduje riešenie problému príliš veľkých vstupov, ktoré dokážu spôsobiť pretečenie pamäte. Preto má byť práca s dátami tohto typu realizovaná pomocou prúdov. Vstupom do prúdu má byť teda chunk[6]. Týmto spôsobom je po chunkoch sekvenčne spracovávaný súbor, ktorý je nahrávaný alebo sťahovaný. Každý chunk je spracovaný aplikáciou (šifrovanie, resp. dešifrovanie, ošetrenie vstupu a ďalšie súvisiace úkony). Po spracovaní je odoslaný buď na middleware server (v prípade nahrávania súboru) alebo na pevný disk zariadenia (v prípade sťahovania súboru).

Na základe schémy na obr. 6 vieme ukázať postup spracovania súboru a jeho následnú distribúciu na cloudové servery. Zvolený súbor má byť najskôr zašifrovaný šifrou AES v GCM móde. Výsledný súbor v zašifrovanom tvare má byť odoslaný na úložisko (prostredníctvom middleware servera), ktoré má dostatočné množstvo priestoru na uchovanie takéhoto súboru. Na kľúč, použitý na šifrovanie súboru má byť aplikovaný Shamirov secret sharing algoritmus, ktorého výstupom sú tri shares pôvodného kľúča. Jednotlivé shares majú byť odoslané všetkým cloudovým úložiskám (opäť prostredníctvom middle-

[6]v preklade aj: kus, resp. dávka informácií



Obr. 6: Schéma ukladania súboru na cloud v aplikácii Datachest.

ware servera). Po tomto procese je potrebné bezpečne zničiť pôvodný kľúč na zariadení.

Získanie súboru z cloudu prebieha analogicky. Klientská aplikácia má vyslať podnet middleware serveru na získanie zašifrovaného súboru z úložiska, na ktorom sa nachádza. Ďalej má prevziať zo všetkých úložísk aspoň t shares kľúča, ktorý prislúcha k danému súboru. Shares má vložiť do Shamirovho secret sharing algoritmu, ktorého výstupom má byť pôvodný kľúč. Použitím rovnakej šifry na dešifrovanie má klientská aplikácia získať pôvodný súbor v otvorenom tvare. Na záver má spracovaný súbor zapísať do filesystému zariadenia.

3.2.2 Middleware server

Serverová aplikácia má byť postavená na princípe REST API [18]. Pomocou HTTP requestov má komunikovať s klientskou (resp. klientskými aplikáciami) a spracovávať prijaté dáta.

Základným prvkom je autentifikácia. Server má poskytovať autentifikačné služby pre klientské aplikácie, ktoré umožnia prihlasovanie do cloudových služieb Google Drive, Microsoft OneDrive a Dropbox. Pomocou protokolu OAuth 2.0 [19] má middleware server získať tzv. access token a predať ho klientskej aplikácii. Všetky cloudy, s ktorými má Datachest pracovať, protokol OAuth 2.0 podporujú a poskytujú API, pomocou ktorej je možné token získať. Access token je reťazec, ktorý autorizuje klientské aplikácie k vykoná-

vaniu operácií na danom cloude [20]. V prípade aplikácie Datachest je potrebné umožnenie prístupu k čítaniu a zápisu na cloudových úložiskách používateľa. Klientská aplikácia má vyžiadať middleware o spustenie autentifikačného procesu. Po jeho dokončení má middleware vrátiť klientskej aplikácii access token ako výsledok tohto procesu.

Serverová aplikácia má byť zároveň schopná implementovať API všetkých cloudov a použiť ich na potrebné úkony. Medzi ne patrí: nahrávanie súborov, sťahovanie súborov, zobrazenie súborov a iné.

Nahrávaný súbor má middleware prijímať v chunkoch fixnej veľkosti od klientskej aplikácie. Každý chunk sekvenčne prepošle na cieľový cloud. Na konci nahrávacieho procesu získa od klientskej aplikácie všetky shares kľúča, ktorým bol súbor šifrovaný a rozdistribuuje ich na jednotlivé cloudy. Analogicky má prebiehať získavanie súborov z cloudu. Middleware má na podnet HTTP requestu začať proces sťahovania vyžiadaného súboru po častiach (chunks). Každý chunk sekvenčne posiela klientskej aplikácii, ktorá si daný súbor vyžiadala. Po ukončení sťahovacieho procesu middleware získa od všetkých cloudov shares kľúča, ktorý je potrebný na dešifrovanie sťahovaného súboru a prepošle ich klientskej aplikácii.

Architektúra serverovej aplikácie má byť implementovaná s možnosťou efektívneho škálovania. Pre každý ďalší cloud, ktorý bude v budúcnosti zaradený do aplikácie, má existovať efektívne začlenenie do súčasnej architektúry. Po zaradení musia byť efektivita, bezpečnosť a použiteľnosť klientskej aj serverovej aplikácie v rámci akceptovateľných hodnôt.

4 API špecifikácia cloudov

V tejto kapitole podrobne opisujeme API cloudov Google Drive, Microsoft OneDrive a Dropbox. Pri uvádzaní jednotlivých HTTP requestov jednotlivých API používame tabuľky, popisujúce konfiguráciu requestov a štruktúru následných odpovedí (HTTP response). Z dôvodu robustnosti jednotlivých špecifikácií sme vybrali iba časť, ktorá je používaná v aplikácii (kapitola 5).

4.1 Google Drive API

4.1.1 Vytváranie priečinkov

V cloudovom úložisku Google Drive je možné vytvárať súbory [21]. Pri vytváraní priečinku je potrebné do tela HTTP requestu dodať objekt v JSON formáte s niekoľkými atribútmi. Priradením `"application/vnd.google-apps.folder"` atribútu `mimeType` špecifikujeme, že vytváraný súbor je priečinok. Ďalšími atribútmi vieme ešte špecifikovať jeho názov a identifikátor jeho rodiča. Rodič je priečinok, ktorý je v hierarchii o jednu úroveň vyššie ako vytváraný priečinok. Pomocou operácie vytvárania priečinkov je možné vytvoriť priečinkovú hierarchiu v cloude a na základe nej organizovať odosielané súbory z aplikácie Datachest. HTTP request je definovaný v tabuľke 3.

Typ HTTP requestu	`POST`
Základná URL	`https://www.googleapis.com`
Endpoint	`/drive/v3/files`
Telo requestu	Atribúty vytváraného súboru v JSON formáte (voliteľné).
HTTP hlavičky	`Authorization: "Bearer <access-token>"` [7] Autorizácia. Uvádza sa access token. `Content-Type: application/json` Uvádza sa v prípade poskytnutia metadát súboru.
Odpoveď	**Kód** `HTTP 200 OK` **Telo** Atribúty vytvoreného súboru v JSON formáte.

Tabuľka 3: Google Drive API: vytváranie súboru typu: priečinok. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

[7] Pri opise HTTP requestov v tejto práci označujeme syntaxou <> ohraničenie literálu v reťazci (angl. výraz: string interpolation).

Typ HTTP requestu	`POST`
Základná URL	`https://www.googleapis.com`
Endpoint	`/upload/drive/v3/files?uploadType=media`
Telo requestu	Súbor.

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Type`
	Uvádza sa typ nahrávaného súboru.
	`Content-Length`
	Uvádza sa počet nahrávaných bajtov (veľkosť súboru).

Odpoveď	**Kód** `HTTP 200 OK` **Telo** Atribúty nahratého súboru v JSON formáte.

Tabuľka 4: Google Drive API: nahrávanie súboru metódou simple upload. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

4.1.2 Nahrávanie súborov

Google Drive API umožňuje nahrávanie súborov na cloud viacerými spôsobmi. V závislosti od veľkosti súboru je možné nahrať súbor v celku (simple upload), alebo po častiach (resumable upload) [22]. Konfigurácia HTTP requestu a štruktúra následnej odpovede pri metóde simple upload sú zapísané v tabuľke 4.

Metódu resumable upload je nutné použiť pri súboroch väčších ako 5 MB. Okrem flexibility nahrávať veľké súbory poskytuje aj ďalšie vlastnosti. Tým, že sa súbor nahráva po častiach, je proces odolný voči sieťovým výpadkom. Po obnovení spojenia je možné pokračovať v nahrávaní súboru tam, kde nastal výpadok, namiesto reštartovania procesu od úplného začiatku. Prehľad o stave nahrávania pomocou nahratých častí poskytuje možnosť zobrazovať používateľovi priebežný stav procesu, ktorý indikuje, aké množstvo z celkového objemu súboru je už nahratých na cloud.

Postup pri metóde resumable upload možno rozdeliť do dvoch krokov. V prvom kroku je poslaný úvodný HTTP request (tabuľka 5). Ten je potrebný na získanie URL, ktorá je adresou nahrávania daného súboru po častiach (v angl. nazývaná ako resumable session URL). URL má limitovanú platnosť a je rezervovaná pre súbor s vlastnosťami opísanými v tomto requeste. Prichádza v odpovedi requestu v `Location` hlavičke.

Typ HTTP requestu	`POST`
Základná URL	`https://www.googleapis.com`
Endpoint	`/upload/drive/v3/files?uploadType=resumable`
Telo requestu	Metadáta súboru, ktorý má byť nahratý (voliteľné).

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Type: application/json`
	Uvádza sa v prípade poskytnutia metadát súboru.

Odpoveď	**Kód**
	`HTTP 200 OK`
	HTTP hlavičky
	`Location`

Tabuľka 5: Google Drive API: nahrávanie súboru metódou resumable upload. Úvodný request. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

V druhom kroku používame resumable session URL na nahrávanie súboru. Nahrávanie je vykonávané vo viacerých requestoch sekvenčne (tabuľka 6). V prípade paralelného posielania viacerých častí súboru proces zlyhá. Dáta súboru sú posielané v tele requestu. Hlavičky `Content-Length` a `Content-Range` používame na uvádzanie informácií o bajtoch súboru: koľko a ktoré bajty súboru sú práve posielané. Google Drive API vyžaduje posielanie chunkov s veľkosťou, ktorá je násobkom 256 kilobajtov. Toto pravidlo nemusí platiť pre posledný chunk, ktorý ukončuje nahrávanie. Napríklad, `Content-Range: bytes 0-524287/2000000` znamená, že je odosielaných prvých 524 288 bajtov ($2 \cdot 256$ kB) súboru s celkovou veľkosťou 2 000 000 bajtov. Odpoveď requestu obsahuje hlavičku `Range`, v ktorej je uvedené, v akom bajtovom rozmedzí je očakávaný ďalší chunk. Táto hlavička by mala byť smerodajná pri špecifikovaní ďalšieho bajtového rozpätia a nie to, koľko bajtov bolo odoslaných v predchádzajúcom requeste. Môže sa stať, že server neprijal všetky bajty v predchádzajúcom HTTP requeste. Hlavička `Range` preto poskytuje informáciu, koľko bajtov bolo serverom skutočne prijatých.

4.1.3 Získanie zoznamu nahratých súborov

Vykonaním tejto operácie je možné používateľovi v aplikácii zobraziť súbory, ktoré má nahraté na cloudovom úložisku [23]. API vráti zoznam súborov v JSON formáte. Každý súbor obsahuje informácie ako: dátum nahratia, názov, id, typ a podobne. HTTP request je zobrazený v tabuľke 7.

Typ HTTP requestu	`PUT`
URL	Resumable session URL.
Telo requestu	Chunk odosielaného súboru.

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Length`
	Uvádza sa počet odosielaných bajtov.
	`Content-Range: "bytes <byte-range>/<total-bytes>"`
	Uvádza sa rozpätie odosielaných bajtov z celkového počtu bajtov súboru.

Odpoveď	**Kód**
	`HTTP 200 OK` alebo `HTTP 201 Created`
	HTTP hlavičky
	`Range`

Tabuľka 6: Google Drive API: nahrávanie súboru metódou resumable upload. Nahrávanie na resumable session URL. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

Typ HTTP requestu	`GET`
Základná URL	`https://www.googleapis.com`
Endpoint	`/drive/v3/files?<query>`
HTTP hlavičky	`Authorization: "Bearer <access-token>"` Autorizácia. Uvádza sa access token.
Odpoveď	**Kód** `HTTP 200 OK` **Telo** Zoznam súborov, ktoré spĺňajú query požiadavku v JSON formáte.

Tabuľka 7: Google Drive API: získanie zoznamu nahratých súborov. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

Typ HTTP requestu	`GET`
Základná URL	`https://www.googleapis.com`
Endpoint	`/drive/v3/files/<file-id>?alt=media`
HTTP hlavičky	`Authorization: "Bearer <access-token>"` Autorizácia. Uvádza sa access token.
Odpoveď	**Kód** `HTTP 200 OK` **Telo** Obsah súboru.

Tabuľka 8: Google Drive API: sťahovanie súboru. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

4.1.4 Sťahovanie súborov

Sťahovanie súborov pomocou Google Drive API prebieha špecifikovaním identifikátora súboru[8] a URL parametra `alt=media` [24]. Parameter špecifikuje, že request žiada API o obsah súboru. Bez parametra by v odpovedi boli iba informácie o danom súbore, nie jeho obsah. Ak HTTP request prebehne úspešne, API vráti súbor v plnej veľkosti. HTTP request je uvedený v tabuľke 8.

4.2 Microsoft OneDrive API

4.2.1 Vytváranie priečinkov

Vytváranie priečinkov je potrebné z rovnakých dôvodov ako je uvedené v predchádzajúcej podkapitole. Podobne ako pri Google Drive API, aj v tomto prípade je možné špecifikovať metadáta vytváraného priečinka v tele requestu ako JSON objekt [25]. Vieme vyplniť atribúty ako: názov priečinka, správanie v prípade už existujúceho priečinka s rovnakým názvom a podobne. Cestu k priečinku definujeme priamo v URL. Koreňový priečinok značíme ako `/root`. Ak je potrebné priečinok vytvoriť hlbšie ako v koreňovom priečinku, zvyšok cesty k priečinku pridáme k `/root` s ohraničením dvojbodkami. Napríklad, pri vytváraní priečinka `Files` s cestou `/root/Datachest`, skonštruujeme URL nasledovne: `https://graph.microsoft.com/v1.0/me/drive/root:/Datachest:/children`. Celá štruktúra HTTP requestu je uvedená v tabuľke 9.

[8] Súvisí so získaním zoznamu nahratých súborov. Viď tabuľka 7.

Typ HTTP requestu	`POST`
Základná URL	`https://graph.microsoft.com/v1.0/me/drive`
Endpoint	`/root<path>/children`
Telo requestu	Atribúty vytváraného priečinka v JSON formáte.

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Type: application/json`
	Uvádza sa formát tela HTTP requestu.

Odpoveď	**Kód** `HTTP 201 Created` **Telo** Atribúty vytvoreného priečinka v JSON formáte.

Tabuľka 9: Microsoft OneDrive API: vytváranie priečinka. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

4.2.2 Nahrávanie súborov

Microsoft OneDrive API umožňuje nahrávať súbory po častiach alebo v celku, v závislosti od veľkosti súboru. Nahrávanie súboru v celku je povolené pre súbory menšie ako 4 MB [26]. HTTP requesty tohto typu sú opísané v tabuľke 10.

Väčšie súbory je nutné nahrávať v sekvenčných HTTP requestoch po častiach (chunkoch). V prvom kroku je potrebné vytvoriť tzv. upload session [27]. Vytvorenie upload session spočíva v poslaní requestu (tabuľka 11) a získaní dočasnej URL, ktorá slúži na sekvenčné nahrávanie obsahu súboru. Ide o adresu dočasného úložiska, ktoré je používané na nahrávanie súboru, kým nie je proces dokončený. Cesta k súboru je špecifikovaná rovnakým spôsobom, ako pri vytváraní priečinka pomocou Microsoft OneDrive API. V prípade nahrávania nového súboru je súčasťou cesty aj názov samotného súboru. Napríklad, pri nahrávaní súboru `encryptedFile.dat` s cieľovým priečinkom `Files`, skonštruujeme URL nasledovne: `https://graph.microsoft.com/v1.0/me/drive/items/root:/Datachest/Files/encryptedFile.dat:/createUploadSession`. Telo HTTP requestu je voliteľné. V ňom sú špecifikované metadáta nahrávaného súboru ako JSON objekt. Objekt má rovnaké atribúty, ako objekt tela requestu pri vytváraní priečinku pomocou Microsoft OneDrive API. V prípade poskytnutia JSON objektu musí byť atribút názvu súboru totožný s názvom súboru uvedenom v URL. V prípade nezhody mena súboru v JSON objekte a mena súboru v URL vráti API chybný kód a proces zlyhá. V prípade úspešného requestu pri-

Typ HTTP requestu	`PUT`
Základná URL	`https://graph.microsoft.com/v1.0/me/drive`
Endpoint	`/items/root<path>/content`
Telo requestu	Súbor.

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Type`
	Uvádza sa typ súboru v tele HTTP requestu.

Odpoveď	**Kód** `HTTP 201 Created` **Telo** Atribúty vytvoreného súboru v JSON formáte.

Tabuľka 10: Microsoft OneDrive API: nahrávanie súboru do 4 MB v celku. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

chádza v odpovedi JSON objekt, v ktorom je špecifikovaná dočasná URL a dátum jej platnosti.

Po vytvorení upload session je možné začať s nahrávaním súboru. Súbor je nutné nahrávať v chunkoch vo veľkostiach predpísaných v API. Maximálna veľkosť chunku je 60 MiB. Teda, ak celková veľkosť súboru nepresahuje 60 MiB, súbor možno nahrať v celku pomocou upload session. Veľkosť chunku musí byť násobkom 320 KiB, t.j., minimálna veľkosť chunku je 320 KiB. HTTP hlavičky `Content-Length` a `Content-Range` používame rovnako, ako v prípade Google Drive API. Pre každý úspešne nahratý chunk API vracia odpoveď `202 Accepted` a JSON objekt špecifikujúci nasledujúci očakávaný rozsah bajtov, ktoré majú byť odoslané. Po odoslaní posledného chunku je odpoveď API uvedená v tabuľke 12.

4.2.3 Získanie zoznamu nahratých súborov

Špecifikácia HTTP requestu je rovnaká ako v tabuľke 9 (vytváranie priečinku), s dvomi rozdielmi:

- Typ requestu nie je `POST`, ale `GET` [28].
- Telo requestu sa neposiela.

Microsoft OneDrive API vracia zoznam súborov, ktoré sú umiestnené pod špecifikovanou cestou. Cestu špecifikujeme v URL, rovnakou logikou ako pri vytváraní upload session,

Typ HTTP requestu	`POST`
Základná URL	`https://graph.microsoft.com/v1.0/me/drive`
Endpoint	`/items/root<path>/createUploadSession`
Telo requestu	Metadáta súboru, ktorý má byť nahratý (voliteľné).

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Type: application/json`
	Uvádza sa v prípade poskytnutia metadát súboru.

Odpoveď	**Kód** `HTTP 200 OK` **Telo** Atribúty dočasnej URL v JSON formáte.

Tabuľka 11: Microsoft OneDrive API: vytváranie upload session. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

Typ HTTP requestu	`PUT`
URL	Dočasná upload session URL.
Telo requestu	Chunk odosielaného súboru.

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Length`
	Uvádza sa počet odosielaných bajtov.
	`Content-Range: "bytes <byte-range>/<total-bytes>"`
	Uvádza sa rozpätie odosielaných bajtov z celkového počtu bajtov súboru.

Odpoveď	**Kód** `HTTP 200 OK` alebo `HTTP 201 Created` **Telo** Atribúty nahratého súboru v JSON formáte.

Tabuľka 12: Microsoft OneDrive API: nahrávanie súboru väčšieho ako 4 MB pomocou upload session. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

Typ HTTP requestu	`GET`
Základná URL	`https://graph.microsoft.com/v1.0/me/drive`
Endpoint	`/items/<file-id>/content`
HTTP hlavičky	`Authorization: "Bearer <access-token>"` Autorizácia. Uvádza sa access token.
Odpoveď	**Kód** `HTTP 200 OK` **Telo** Obsah súboru.

Tabuľka 13: Microsoft OneDrive API: sťahovanie súboru. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

nahrávaní súborov do 4 MB alebo vytváraní priečinka.

4.2.4 Sťahovanie súborov

Súbor sťahujeme špecifikovaním identifikátora súboru v URL requeste [29]. API vracia súbor v plnej veľkosti. HTTP request je uvedený v tabuľke 13.

4.3 Dropbox API

4.3.1 Vytváranie priečinkov

Vytváranie priečinka pomocou Dropbox API prebieha podobne ako pri ostatných API, ktoré sme uviedli. V requeste špecifikujeme telo v podobe JSON objektu [30]. Objekt pozostáva z dvoch atribútov: cesta k priečinku a indikácia, či má byť meno zmenené, ak už také existuje. Koreňový priečinok sa označuje ako `/`. Cesta k priečinku obsahuje aj samotný názov vytváraného priečinku. Napríklad, pri vytváraní priečinka `Files` s cestou `/Datachest` by sme atribútu priradili hodnotu `/Datachest/Files`. Týmto spôsobom nie je potrebné špecifikovať názov priečinka v dodatočnom atribúte, alebo v URL, prípadne v HTTP hlavičke. HTTP request je uvedený v tabuľke 14.

4.3.2 Nahrávanie súborov

Dropbox API umožňuje nahrávať súbory v celku alebo po častiach pomocou upload session (terminológia rovnaká ako pri Microsoft OneDrive API).

Súbory v celku možno posielať až do veľkosti 150 MB [31]. Súbor je potrebné vložiť do tela HTTP requestu. Jeho metadáta sa vkladajú ako špeciálna HTTP hlavička, ktorá je špecifická pre Dropbox API. Ide o JSON objekt vo forme reťazca. Uvádzajú sa parametre ako: cesta k súboru, mód vkladania, spôsob riešenia konfliktu v prípade zhody v mene a

Typ HTTP requestu	`POST`
Základná URL	`https://api.dropboxapi.com/2/files`
Endpoint	`/create_folder_v2`
Telo requestu	Atribúty vytváraného priečinka v JSON formáte.

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Type: application/json`
	Uvádza sa formát tela HTTP requestu.

Odpoveď	**Kód** `HTTP 200 OK` **Telo** Atribúty vytvoreného priečinka v JSON formáte.

Tabuľka 14: Dropbox API: vytváranie priečinka. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

iné. Celá konfigurácia HTTP requestu je uvedená v tabuľke 15.

Súbory väčšie ako 150 MB je nutné nahrávať pomocou upload session. Nahrať možno súbory s veľkosťou najviac 350 GB. Dropbox API pre túto operáciu definoval tri URL: inicializácia upload session, nahrávanie súboru, ukončenie upload session [32]. Úvodný request inicializuje nahrávací proces spolu s prvým chunkom dát (tabuľka 16). V odpovedi requestu je uvedený identifikátor novej upload session. Táto informácia je potrebná v ďalších krokoch.

Nahrávanie ďalších chunkov prebieha v sekvenčných requestoch, ako je definované v tabuľke 17. Platnosť upload session je stanovená na 7 dní. Konfiguráciu upload session možno upraviť tak, aby bolo možné nahrávať chunky aj konkurentne, s cieľom zvýšenia efektivity. Chunk je odosielaný v tele HTTP requestu. Keďže URL nie je špecifická pre konkrétnu nahrávaciu reláciu, je potrebné uviesť údaj, o akú reláciu ide. Na to je určený identifikátor upload session, získaný v inicializačnom requeste. Ten sa pri nahrávaní ďalších chunkov uvádza v hlavičke `Dropbox-API-Arg` [33]. Ide o JSON objekt vo formáte reťazca. Ďalším atribútom JSON objektu je `offset`. Je to číselná hodnota, ktorá indikuje, ku ktorému bajtu nahrávaného súboru sa má priložiť ďalší chunk.

Na ukončenie nahrávania sa používa posledná z troch uvedených URL (tabuľka 18). Súčasťou `Dropbox-API-Arg` hlavičky sú okrem `session_id` a `offset` aj informácie o nahratom súbore [34]. Ide o rovnaký typ atribútov ako pri vytváraní priečinka. Veľkosť

Typ HTTP requestu	`POST`
Základná URL	`https://content.dropboxapi.com/2/files`
Endpoint	`/upload`
Telo requestu	Súbor.

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Type`
	Uvádza sa typ súboru v tele HTTP requestu.
	`Dropbox-API-Arg`
	Uvádzajú sa metadáta súboru: JSON objekt formátovaný do reťazca.

Odpoveď	**Kód** `HTTP 200 OK` **Telo** Atribúty vytvoreného súboru v JSON formáte.

Tabuľka 15: Dropbox API: nahrávanie súboru do 150 MB v celku. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

Typ HTTP requestu	`POST`
Základná URL	`https://content.dropboxapi.com/2/files`
Endpoint	`/upload_session/start`
Telo requestu	Prvý chunk súboru.

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Type`
	Uvádza sa typ nahrávaného súboru.

Odpoveď	**Kód** `HTTP 200 OK` **Telo** Identifikátor novej upload session v JSON formáte.

Tabuľka 16: Dropbox API: Inicializácia upload session. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

Typ HTTP requestu	`POST`
Základná URL	`https://content.dropboxapi.com/2/files`
Endpoint	`/upload_session/append_v2`
Telo requestu	Chunk súboru.

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Type`
	Uvádza sa typ nahrávaného súboru.
	`Dropbox-API-Arg`
	Uvádzajú sa ďalšie informácie k upload session: JSON objekt formátovaný do reťazca.

Odpoveď	**Kód** `HTTP 200 OK`

Tabuľka 17: Dropbox API: Nahrávanie súboru po častiach pomocou upload session. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

posledného chunku je rovná počtu zvyšných, doteraz nenahratých bajtov súboru. V tele odpovede sú informácie o nahratom súbore v JSON formáte. Pre všetky tri kroky nahrávania platí, že veľkosť odosielaného chunku je variabilná. Avšak platí, že nemôže presiahnuť veľkosť 150 MB.

4.3.3 Získanie zoznamu nahratých súborov

Získanie zoznamu súborov v špecifikovanom priečinku možno vykonať pomocou HTTP requestu, ktorý je špecifikovaný v tabuľke 19. Súbory, ktoré majú byť serverom vrátené, možno filtrovať špecifikovaním atribútov [35]. Atribúty uvádzame v tele HTTP requestu, ako JSON objekt. Niektoré z nich bližšie opíšeme. Atribútom `path` možno špecifikovať cestu k priečinku, ktorého zoznam súborov treba zobraziť. Počet výsledkov je možné limitovať atribútom `limit`. Zahrnutie vymazaných súborov možno prepínať Booleovským atribútom `include_deleted`. Dropbox API umožňuje získať zoznam aj rekurzívne. T.j., od špecifikovanej cesty prechádza stromom súborovej štruktúry až po list a vráti všetky nájdené položky. Atribút pre toto správanie je uvedený ako `recursive`. Predvolená hodnota je `False`. API vracia zoznam súborov na základe uvedených požiadaviek ako JSON objekt.

Typ HTTP requestu	`POST`
Základná URL	`https://content.dropboxapi.com/2/files`
Endpoint	`/upload_session/finish`
Telo requestu	Posledný chunk súboru.

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Type`
	Uvádza sa typ nahrávaného súboru.
	`Dropbox-API-Arg`
	Uvádzajú sa ďalšie informácie k upload session:
	JSON objekt formátovaný do reťazca.

Odpoveď	**Kód** `HTTP 200 OK` **Telo** Atribúty nahratého súboru v JSON formáte.

Tabuľka 18: Dropbox API: Finalizácia upload session. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

Typ HTTP requestu	`POST`
Základná URL	`https://api.dropboxapi.com/2/files`
Endpoint	`/list_folder`
Telo requestu	Špecifikácia žiadaných súborov v JSON formáte.

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Content-Type: application/json`
	Uvádza sa formát tela HTTP requestu.

Odpoveď	**Kód** `HTTP 200 OK` **Telo** Zoznam súborov v JSON formáte.

Tabuľka 19: Dropbox API: Získavanie zoznamu súborov. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

4.3.4 Sťahovanie súborov

Na stiahnutie súboru sa uvádza jeho identifikátor alebo cesta. Hodnota sa priradí k atribútu `path` [36]. Atribút sa posiela ako JSON objekt formátovaný do reťazca v HTTP hlavičke `Dropbox-API-Arg`. Poslaním requestu sa začne sťahovanie súboru v plnej veľkosti. Request je bližšie opísaný v tabuľke 20.

Typ HTTP requestu	`POST`
Základná URL	`https://content.dropboxapi.com/2/files`
Endpoint	`/download`

HTTP hlavičky	`Authorization: "Bearer <access-token>"`
	Autorizácia. Uvádza sa access token.
	`Dropbox-API-Arg`
	Uvádza sa identifikátor alebo cesta k súboru:
	JSON objekt formátovaný do reťazca.

Odpoveď	**Kód** `HTTP 200 OK` **Telo** Zoznam súborov v JSON formáte.

Tabuľka 20: Dropbox API: Sťahovanie súboru. Konfigurácia HTTP requestu a štruktúra následnej odpovede.

5 Softvérová implementácia

Na základe predstaveného problému, návrhu riešenia a špecifikácie API bol implementovaný softvér, ktorý pracuje s dôvernými údajmi používateľa a spravuje ich na princípe secret sharing algoritmu. Vzhľadom na novo vzniknuté problémy a časové obmedzenia sa výsledné riešenie líši od návrhu v niektorých bodoch.

Middleware aplikácia nebola implementovaná. Hlavným dôvodom je časová náročnosť implementácie. Požiadavka autentifikácie na strane middleware aplikácie vyžadovala manuálnu implementáciu protokolu OAuth 2.0 a znemožnila použitie klientských OAuth knižníc, ktoré umožňujú jednoduché použitie. Tento proces vyžadoval príliš vysokú komplexitu a nespĺňal časový rámec vyhradený na implementáciu riešenia autentifikácie a autorizácie aplikácie Datachest. Naplnenie bezpečnostných požiadaviek serverovej aplikácie je ďalší dôvod, prečo nebola implementovaná. Zabránenie rôznym útokom na server (napríklad DDoS), riešenie práv a prístupov, či filtrovanie prichádzajúcich požiadaviek sú problémy, ktorých kopmlexita znemožňuje ich riešenie vo vyhradenom čase. Zároveň ide o problémy, ktoré priamo nesúvisia so zadaním práce. Nakoľko hlavnou úlohou serverovej aplikácie malo byť nahrávanie a sťahovanie súborov, už s malým počtom paralelne prebiehajúcich spojení vzniká hustý tok dát. Riešenie tohto problému vyžaduje robustné hardvérové riešenie, ale aj pokročilé metódy pri nasadzovaní aplikácie a samotnej architektúry softvéru. Tento komponent je vo všeobecnosti vhodným prvkom architektúry aplikácie s požiadavkami a cieľmi, ktoré sme v práci opísali. Poskytuje efektívnu škálovateľnosť a zlučuje duplicitný kód vo všetkých klientských aplikáciách na jedno miesto. Avšak pre účely tejto práce je možné vytvoriť funkčné riešenie aj bez tohto komponentu.

Jedinou implementovanou aplikáciou ostáva klientská, mobilná aplikácia Datachest vytvorená pre platformu iOS. Keďže middleware aplikácia nebola implementovaná, klientská aplikácia vykonáva autentifikačné, nahrávacie aj sťahovacie procesy. Komunikuje pritom priamo s API jednotlivých cloudov, prípadne knižnicami, ktoré boli navrhnuté na tieto úkony.

V tejto kapitole podrobne opisujeme mobilnú aplikáciu Datachest. Technicky analyzujeme komponenty, architektúru, štruktúru, logiku a zdrojový kód tejto aplikácie.

5.1 Architektúra

5.1.1 MVVM

Základom architektúry aplikácie je návrhový vzor MVVM, často používaný pri vývoji moderných iOS aplikácií [37]. Názov je skráteným výrazom troch pojmov: Model, View, ViewModel.

Model je reprezentácia skupiny dát. V jazyku Swift sa pod modelom myslí štruktúra alebo enumerácia. Model neobsahuje konkrétne dáta, iba všeobecne definuje, ako majú byť štrukturované. Modely reprezentujú interné dáta aplikácie, ale aj prichádzajúce a odchádzajúce dáta (komunikácia s API). Napríklad, pre zobrazenie zoznamu súborov uložených v Google Drive podľa tab. 7 je vhodné definovať dva nasledujúce dátové modely:

```
struct GoogleDriveListFilesResponse: Codable {
    let files : [GoogleDriveFileResponse]
}

struct GoogleDriveFileResponse: Codable, Identifiable {
    let kind: String
    let id: String
    let name: String
    let mimeType: String
}
```

Štruktúra horného modelu zodpovedá JSON objektu, ktorý vracia Google Drive API. Obsahuje jediný atribút, ktorý obsahuje zoznam objektov (súborov). Jednotlivé objekty zoznamu majú štruktúru definovanú v dolnom modeli. Protokol `Codable`[9] [38] jazyka Swift umožňuje konvertovať:

- štruktúru na JSON objekt pri odosielaní dát
- JSON objekt na štruktúru pri prijímaní dát

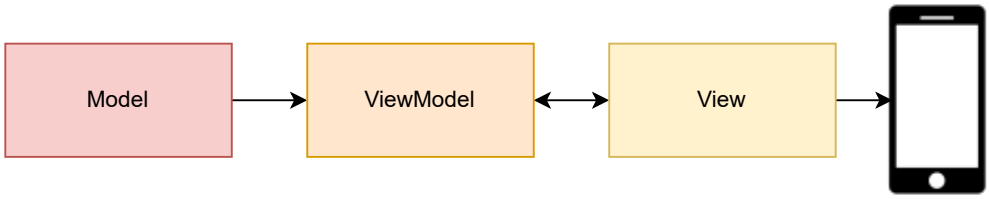
Podľa MVVM by dáta aplikácie mali byť štrukturované do modelov pre jednoduchosť práce s týmito dátami. Pre naplnenie modelu konkrétnymi dátami vytvárame inštancie modelu v logických častiach aplikácie.

View a ViewModel sú dvojice komponentov, ktoré spolu úzko súvisia. ViewModel je typ triedy, ktorá pracuje s modelmi a vykonáva nad nimi logické operácie, validácie a podobne. Poskytuje sadu metód a atribútov, s ktorými pracuje View. View je špeciálny typ štruktúry, ktorý používateľovi vykresľuje grafické rozhranie na obrazovku. View môže byť tlačidlo, karta, ale aj komplexné zobrazenie, pozostávajúce s viacerých, menších View štruktúr. Podľa MVVM má View iba vykresľovať grafické elementy a napĺňať ich dátami z ViewModelu. View by s dátami nemal pracovať (logické operácie, validácia a podobne).

[9] Pre jednoduchosť uvedené ako protokol. V skutočnosti je `Codable` združeným pomenovaním (`typealias`) pre protokoly `Encodable` a `Decodable`. Viď [38].

Preto by každý View mal mať priradený ViewModel. ViewModel možno vidieť ako logické rozšírenie jedinečne priradené k danému View.

Diagram 7 graficky znázorňuje vzájomné vzťahy medzi M, V a VM návrhového vzoru MVVM. Medzi View a ViewModelom je obojsmerná komunikácia. View dodáva ViewModelu vstupy používateľa. ViewModel vráti View výstup týchto vstupov, ktorý View zobrazí používateľovi ako odozvu jeho vstupov. Modely sú pevne definované a pracuje s nimi ViewModel. View v prípade potreby číta referencie na modely spracovávané ViewModelom a zobrazuje atribúty modelov na obrazovke.



Obr. 7: Datachest: Návrhový vzor MVVM.

Aplikácia Datachest nenasleduje MVVM vo všetkých aspektoch. Disponuje ďalšími celkami, ktoré nie sú súčasťou návrhového vzoru. Venujeme sa im nižšie.

5.1.2 Application store

Application store (úložisko aplikácie) je prístup reaktívneho riadenia stavu aplikácie. Nástroje ako Ngrx [39] poskytujú prístup jediného, globálneho zdroja všetkých dát aplikácie (angl. výraz „single source of truth"). Store je reaktívny. To znamená, že akákoľvek zmena v store je okamžite reflektovaná v grafickom rozhraní. Toto správanie je vhodné pre asynchrónne udalosti ako napríklad hlásenie chýb, či získavanie dát z cloudov. Application store bol do aplikácie Datachest implementovaný v primitívnejšej podobe. Neriadi stav dát celej aplikácie. Namiesto toho spravuje iba tie dáta, ktoré sú používané v aplikácii na mnohých miestach a ich umiestnenie vo ViewModeloch by prinieslo limitácie. Obsahuje dáta ako: access tokeny, stav prihlásenia, chybové hlášky, prebiehajúce sťahovania a nahrávania. Uvedené dáta majú v aplikácii globálny charakter. Zároveň platí, že v jednom čase pre nich platí iba jedna „pravda", preto komponenty čerpajúce zo storu môžu mať istotu, že čerpajú správne dáta.

5.1.3 Stateless singleton services

V aplikácii sa vyskytuje mnoho komplexných logických štruktúr, ktoré sú zaradené do jadra: v hierarchii hlbšie (low-level), než ViewModely. Ide o triedy, ktoré sú typu singleton [40]. Singleton triedy môžu mať v aplikácii iba jedinú inštanciu. Očakáva sa, že nebudú ob-

sahovať dáta, resp. stav (angl. výraz „stateless"), iba sadu metód na vykonávanie rôznych operácií. Napríklad, v aplikácii Datachest to sú triedy, ktoré:

- poskytujú komunikáciu s API cez HTTP requesty a vracajú odpoveď volajúcemu
- vyberajú a zapisujú dáta do Apple Keychain (spomíname nižšie)
- komunikujú s filesystémom mobilného zariadenia
- vykonávajú secret sharing operácie
- prepájajú ViewModely s low-level service triedami (tzv. fasády)

Singleton triedu možno v jazyku Swift definovať nasledujúcim spôsobom:

```
class DropboxService {
    static let shared = DropboxService()
    private init() {}

    func metoda() {
        // ...
    }
}
```

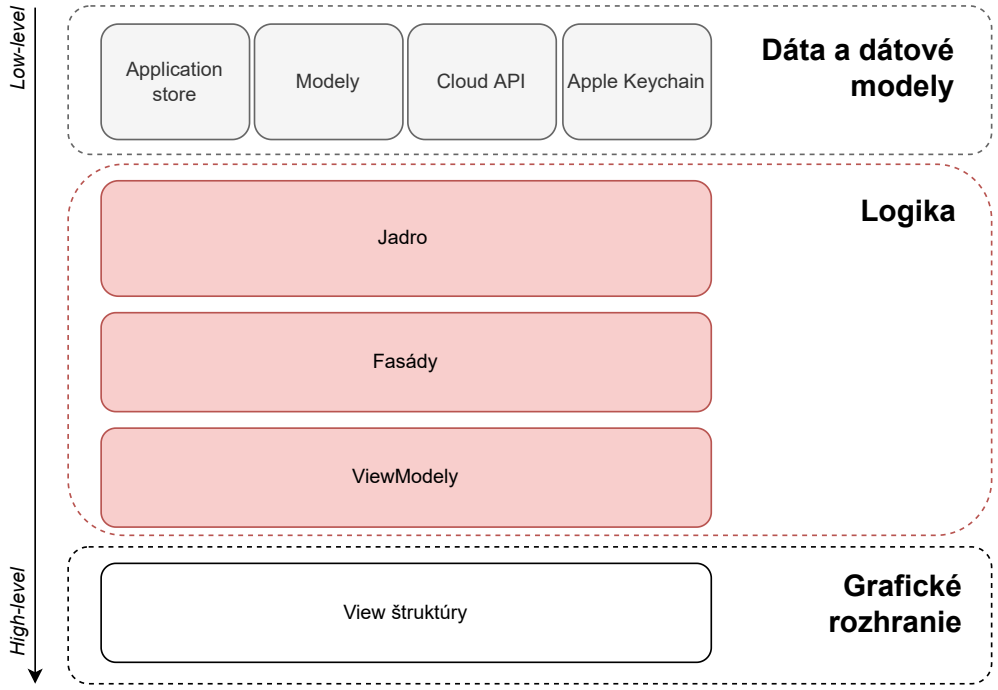
Trieda `DropboxService` je singleton, pretože má neprístupný konštruktor (vďaka kľúčovému slovu `private`). V kóde teda nie je možné vytvoriť inštanciu tejto triedy. Namiesto toho pristupujeme k jej jedinej, už vytvorenej inštancii s názvom `shared`. Ak je potrebné v kóde volať metódu tejto service triedy, volanie je zapísané nasledovne: `DropboxService.shared.metoda()`.

Singleton triedy by mali byť stateless. Ak potrebujú pracovať s dátami, čerpajú ich zo storu alebo ich vyžadujú vo vstupných parametroch jednotlivých metód.

Architektúra aplikácie Datachest pozostáva z viacerých návrhových vzorov, ktoré sa vzájomne dopĺňajú. Pre zhrnutie uvedených princípov a konštrukcií je pre čitateľa poskytnutý diagram. Obr. 8 zobrazuje hierarchiu komponentov aplikácie. Najprednejšia vrstva sú View štruktúry, zobrazujúce grafické rozhranie používateľovi. Hlbšie sú logické komponenty, čerpajúce s najhlbšej, dátovej vrstvy.

5.2 Autentifikácia a správa access tokenov

V podkapitolách 4.1, 4.2 a 4.3 sme uviedli niektoré API operácie cloudových úložísk, ktoré sú potrebné na zostrojenie aplikácie. Na ich použitie je potrebná autorizácia

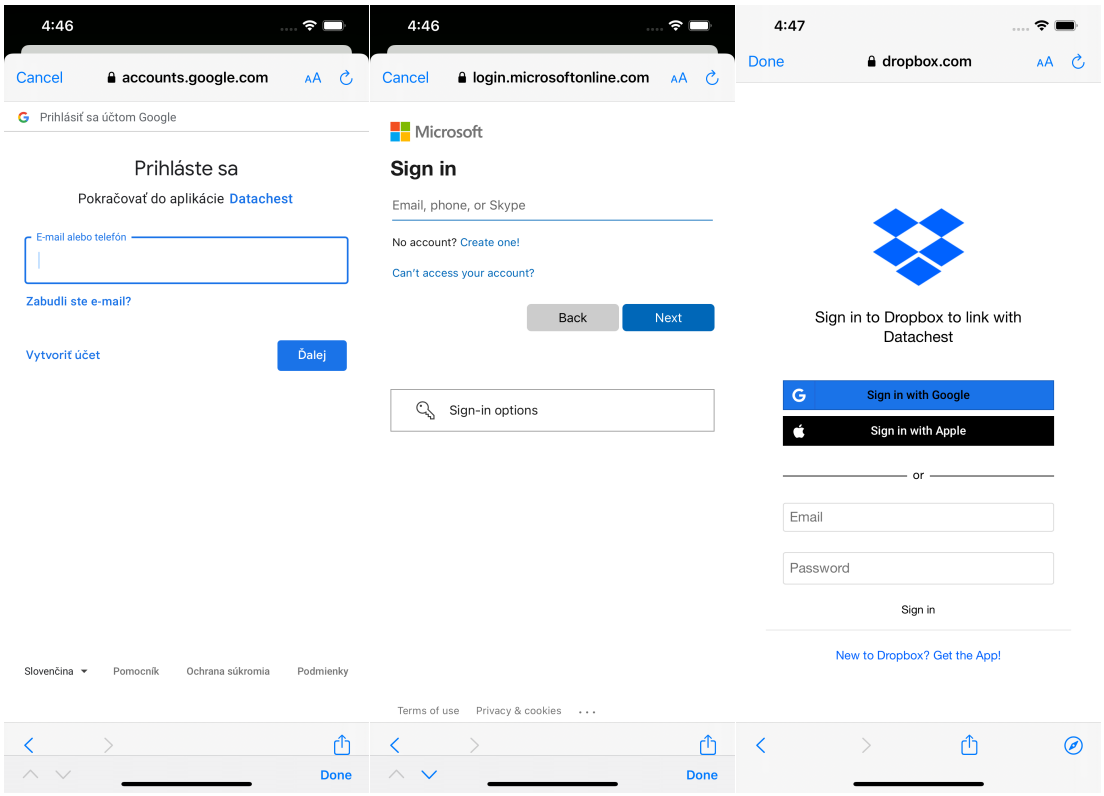


Obr. 8: Datachest: Hierarchia komponentov aplikácie.

pomocou access tokenu. Aplikácia teda umožňuje používateľom prihlásenie do Google, Microsoft a Dropbox účtov na základe protokolu OAuth 2.0. Každý z poskytovateľov disponuje knižnicami pre klientské aplikácie, určené na zjednodušený vývoj autentifikácie a autorizácie účtov [41], [42], [43] (ďalej iba knižnica/knižnice). Knižnice patria medzi závislosti v projekte. Spustenie autentifikačného a autorizačného procesu daného cloud poskytovateľa je v grafickom rozhraní naviazené na tlačidlo. Po stlačení tlačidla je používateľ presmerovaný na prihlasovacie rozhranie daného poskytovateľa (obr. 9). Rozhranie je rovnaké ako v aplikáciach samotných poskytovateľov. Pre používateľa, ktorý tieto služby používa, by mali byť rozhrania známe.

Po úspešnom prihlásení je používateľ presmerovaný späť do aplikácie. V kóde sú tieto procesy implementované v jadre. Pracuje sa s metódami a modelmi knižníc. Pre každého poskytovateľa bola v aplikácii vytvorená jedna singleton, stateless service trieda zodpovedná za správu účtov, autnetifikáciu a autorizáciu: `GoogleAuthService`, `MicrosoftAuthService` a `DropboxAuthService`. Výstupom prihlasovacieho procesu je objekt: účet autorizovaného používateľa. Obsahuje atribúty ako: používateľské meno, email, access token, id token, refresh token, dátum platnosti tokenov a iné. Aplikácia pracuje s access tokenmi poskytovateľov a ich dátumami platnosti.

Tokeny sú uložené do storu, kde prebývajú počas životného cyklu aplikácie. Service



Obr. 9: Datachest: Autentifikačné rozhrania. Google, Microsoft a Dropbox.

triedy zodpovedné za komunikáciu s cloudovými API tokeny zo storu čerpajú pri vytváraní HTTP requestov. Platnosť tokenov je uložená do bezpečne šifrovanej databázy zariadenia, tzv. Apple Keychain [44]. Tieto dáta ostávajú uložené aj po vypnutí aplikácie. Pred použitím access tokenu je skontrolovaná jeho platnosť v Keychaine[10]. Ak je platný, môže sa použiť. Ak jeho platnosť vypršala alebo je tesne pred vypršaním, pomocou autorizačných service tried sa vyžiada na pozadí nový token, bez účasti používateľa.

Hneď po naštartovaní aplikácie prebehne tzv. „silent sign-in" - tiché prihlásenie. Teda, prihlásenie bez interakcie používateľa. Silent sign-in je možný, ak je z minulosti v danom zariadení evidovaný prihlásený používateľ, ktorý sa explicitne neodhlásil. Taký používateľ sa nemusí prihlasovať pri každom spustení aplikácie. Namiesto toho pre neho aplikácia zabezpečí čerstvý access token vždy, keď začne aplikáciu používať. Tento proces je riadený knižnicami jednotlivých cloudov. V kóde sú po naštartovaní aplikácie volané niektoré metódy týchto knižníc, ktoré zabezpečia silent sign-in. Knižnice používajú na ukladanie informácií o prihlásenom účte taktiež Apple Keychain. Odtiaľ tokeny čerpajú pri opätovnom spustení aplikácie. Aplikácia ich odchytí a vloží do storu pre jednoduché používanie aplikáciou počas behu. Pokiaľ tokeny v Keychaine nie sú nájdené, silent sign-in zlyhá a

[10]Dĺžka platnosti access tokenov: Google: 1 hodina, Microsoft: 1 hodina, Dropbox: 4 hodiny.

používateľ sa musí prihlásiť explicitne. Tento stav môže nastať, ak sa používateľ v minulosti explicitne odhlásil, alebo aplikáciu používa prvýkrát. Pri odhlasovaní sú dáta o prihlásenom používateľovi z Keychainu zmazané (taktiež v réžii knižníc). Pokiaľ tokeny v Keychaine sú nájdené, ale sú expirované, automaticky sa získa nový access token s použitím refresh tokenu. Niektoré knižnice (Dropbox) nerozlišujú expirované a čerstvé access tokeny. Poskytnú ho bez ohľadu na jeho platnosť. Preto spravujeme dátumy platnosti tokenov manuálne a používame ich na kontrolu získaného tokenu z Keychainu.

5.3 Firestore

Keďže sa nahrávané súbory v aplikácii Datachest šifrujú a šifrovací kľúč sa rozdelí na 3 shares pomocou secret sharing algoritmu, musí existovať informácia o tom, ktoré shares patria ktorému šifrovanému súboru. Bez tejto informácie by bolo veľmi nepraktické súbory rekonštruovať do pôvodnej formy. Bolo by nutné vyskúšať všetky možné trojice shares z celej množiny shares uložených na cloudových úložiskách používateľa.

Riešenie tejto úlohy vyžaduje použitie centrálnej databázy. Teda, nie je vhodné použiť databázu zariadenia, ktorá je limitovaná samotným zariadením. Služba Firebase od spoločnosti Google toto riešenie umožňuje. Firebase poskytuje klientským aplikáciam služby back-end servera, bez nutnosti ho implementovať [45]. Pomocou Firebase je možné obohatiť klientské aplikácie o funkcionality ako: autentifikácia a správa registrovaných/prihlásených používateľov, centrálna databáza (nazývaná aj ako Firestore [46]), hosting webových klientských aplikácií a podobne. K týmto službám je poskytnutá Firebase API, ktorú používa klientská aplikácia na ovládanie tejto služby.

Na účely klientskej aplikácie Datachest bola vo Firestore vytvorená centrálna Firestore databáza. Databáza je typu NoSQL. Ide o databázu, ktorá uchováva tzv. „dokumenty". Nami vytvorená Firestore databáza je určená výhradne pre aplikáciu Datachest. V iOS aplikácii používame Firestore na ukladanie informácií, ktoré shares patria ku ktorému šifrovanému súboru na cloude a pracujeme s nimi pri nahrávaní a sťahovaní súborov (podkapitoly 5.5, 5.6).

5.4 Organizácia súborov do priečinkov

Na organizáciu súborov sme v aplikácii definovali štruktúru priečinkov, ktorá je rovnaká pre cloudové úložiská, aj úložisko mobilného zariadenia. Vyzerá nasledovne:

Datachest

- Files
- Keys

V koreňovom priečinku úložiska je vytvorený priečinok Datachest. Týmto priečinkom izolujeme dáta aplikácie od dát, ktoré s ňou nesúvisia. V rámci priečinka Datachest súbory ďalej triedime do priečinkov Files a Keys. Priečinok Files obsahuje všetky šifrované súbory (cloud), resp. dešifrované súbory (úložisko mobilného zariadenia) používateľa. V priečinku Keys sú shares kľúčov, ktorými sa šifruje, resp. dešifruje. V úložisku mobilného zariadenia sa priečinok Keys nenachádza. Kľúče, ktoré používa Datachest, nie sú ukladané v mobilnom zariadení.

Aplikácia generuje uvedenú priečinkovú štruktúru na všetkých cloudových úložiskách a vo filesystéme mobilného zariadenia automaticky, pokiaľ štruktúra neexistuje.

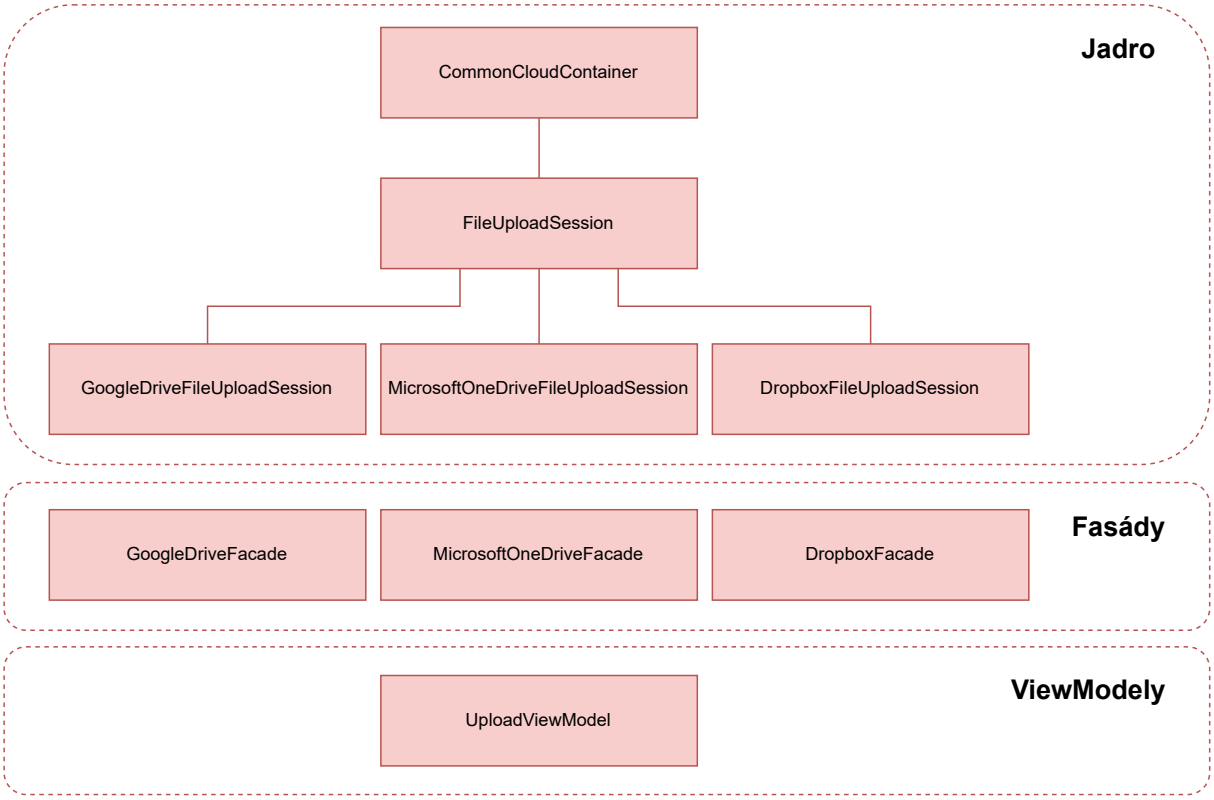
5.5 Nahrávanie súborov

Proces nahrávania súborov je v architektúre štrukturované tak, ako zobrazuje diagram na obr. 10. Skupiny tried sú zaobalené do hierarchických celkov (Jadro, Fasády, ViewModely), uvedených na obr. 8. Medzi triedami, ktoré sú spojené čiarou, existuje dedičný vzťah. Pre každý cloud existuje trieda, ktorá manažuje reláciu nahrávania súboru. Sú to triedy: `GoogleDriveFileUploadSession`, `MicrosoftOneDriveFileUploadSession` a `DropboxFileUploadSession`. Dedia zo spoločnej triedy `FileUploadSession`, ktorá združuje rovnaké metódy a atribúty vyššie spomenutých troch tried. Táto trieda dedí z triedy `CommonCloudContainer`, ktorá je globálnym združením spoločných metód a atribútov všetkých cloudov. Teraz opíšeme podrobný postup nahrávania súboru.

Z obrazoviek uvedených na obr. 11 je používateľ schopný nahrávať súbory na cloud. Môže si vybrať, na ktoré cloudové úložisko bude súbor nahratý (obrazovka vľavo). Táto obrazovka nie je prístupná, ak používateľ nie je prihlásený ku všetkým cloud poskytovateľom. Pre jednoduchosť uvažujme, že používateľ zvolil Google Drive ako cieľový cloud súboru. Po výbere sa mu zobrazí okno, v ktorom môže prehľadávať súbory uložené na zariadení. Po výbere súboru začína proces nahrávania.

`UploadViewModel` zachytáva udalosť výberu súboru a cloudového úložiska. Keďže nahrávanie súboru je komplexná operácia, ViewModel použije fasádu na delegovanie žiadosti jadru architektúry. Použije fasádovú triedu `GoogleDriveFacade` a zavolá metódu `uploadFile()`. Ako vstupný parameter do metódy uvedie adresu súboru vo filesystéme zariadenia. Fasáda inicializuje nahrávaciu reláciu vytvorením inštancie triedy `GoogleDriveFileUploadSession`, ktorá manažuje reláciu nahrávania súboru.

Konštruktor `GoogleDriveFileUploadSession` triedy nastaví do parametrov adresu súboru a veľkosť chunku odosielaných dát. Ďalej skontroluje, či je na príslušnom cloude vytvorená priečinková štruktúra aplikácie. Ak nie, vytvorí ju. Na vytvorenie jednotlivých

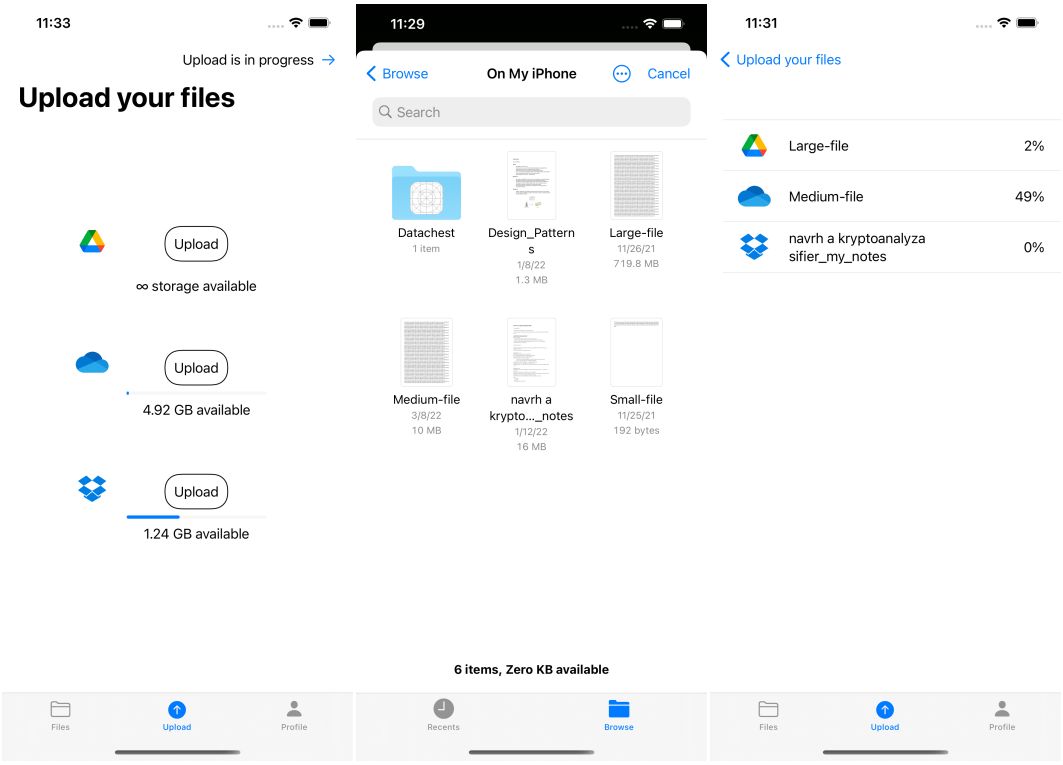


Obr. 10: Datachest: Nahrávanie súborov. Podrobnejší diagram.

priečinkov volá metódy triedy `GoogleDriveService`. Táto trieda riadi komunikáciu s Google Drive API. Pri vytváraní priečinkov používa volanie zobrazené v tabuľke 3.

Ďalej volá fasáda metódu inštancie triedy `GoogleDriveFileUploadSession` s názvom `createNewUploadSession()`. Metóda cez `GoogleDriveService` získa od Google Drive API dočasnú URL, ktorá slúži na nahratie súboru pomocou resumable upload (tabuľka 5). HTTP requestu zároveň poskytne informácie o súbore, ktorý má byť nahratý: meno, priečinok, priečinok, do ktorého má byť súbor nahratý a podobne. Resumable upload URL je po získaní nastavená ako atribút do `FileUploadSession`.

Posledný krok fasády je zavolanie metódy `uploadFile()` na vytvorenom objekte nahrávacej relácie. Metóda spustí nahrávanie súboru po jednotlivých chunkoch sekvenčne. Otvorí vstupný prúd. Z prúdu v cykle číta súbor z disku zariadenia po častiach predom definovanej veľkosti do vyrovnávacej pamäte. Chunk zašifruje šifrou AES, v GCM móde, so 128 bitovým kľúčom. Inicializácia prúdu, generovanie symetrického kľúča a hodnoty nonce vykonal pri spustení relácie rodič triedy, `FileUploadSession`. Výstupom šifrovania je zašifrovaný chunk a authentication tag, ktorý slúži na overenie integrity pri dešifrovaní. Tag je pre každý chunk iný. Vzniká pole tagov, ktoré počas nahrávania uchovávame v

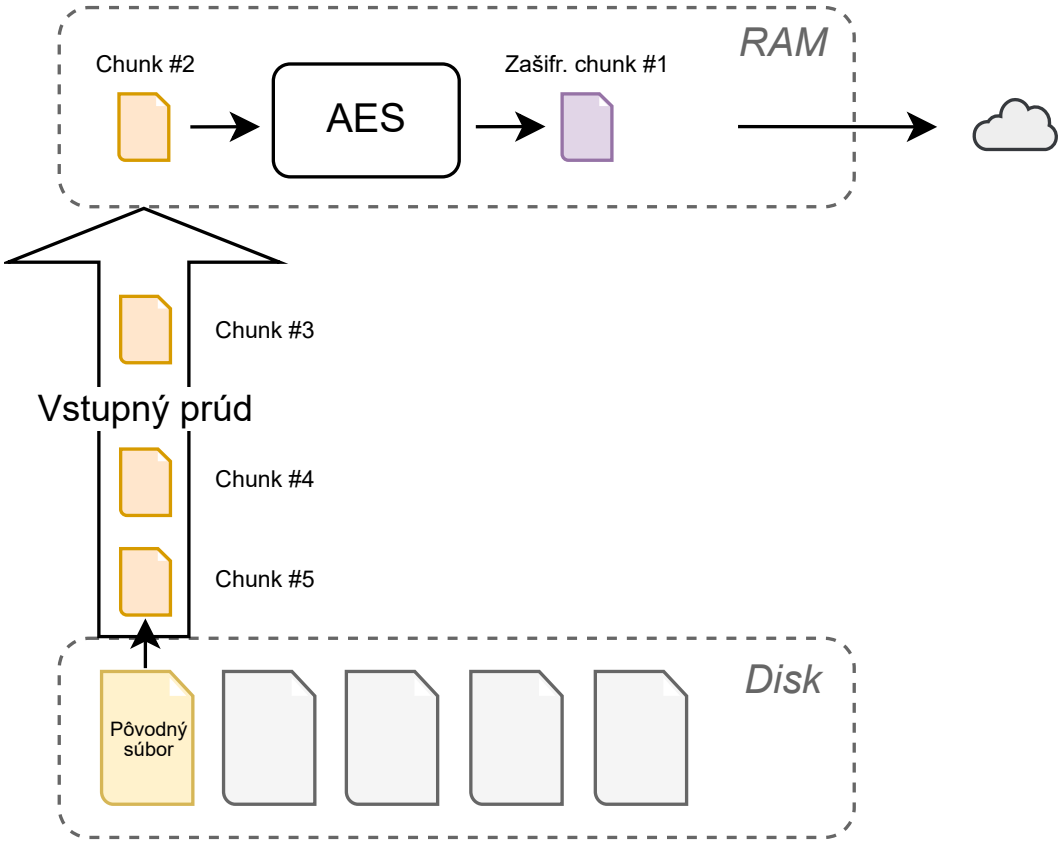


Obr. 11: Datachest: Výber cloudu na nahratie súboru (vľavo), výber súboru na nahratie (stred), zoznam práve nahrávaných súborov (vpravo). Grafické rozhrania.

pamäti. Keďže ide o GCM mód šifry AES, je potrebné uchovať nielen šifrovací kľúč, ale aj nonce a všetky tagy všetkých chunkov. Bez týchto parametrov nie je možné súbor spätne dešifrovať. Pomocou `GoogleDriveService` odosielame šifrovaný chunk na cloud (tabuľka 6). Operáciu opakujeme až vtedy, keď príde odpoveď zo servera o úspešnom prijatí chunku. Vtedy sa rekurzívne volá `uploadFile()`. Opísaný proces graficky znázorňuje obr. 12.

Ukončenie nahrávania indikuje návratový HTTP kód z API. Ak je kód rovný hodnote 200 alebo 201, nahrávanie je ukončené. Spolu s týmto kódom je spracované telo HTTP odpovede. Obsahuje atribúty nahratého súboru. Z atribútov vyberieme identifikátor súboru a uchovávame na neskoršie použitie. Po ukončení nahrávania zavolá `uploadFile()` metódu `distributeKeyShares()`. Metóda rozdelí použitý kľúč na tri shares a rozdistribuuje ich medzi všetky cloudy v podobe troch súborov (ďalej iba pomocné súbory). Keďže tento proces je pre všetky cloudy totožný, metóda je umiestnená v `FileUploadSession`.

Každý pomocný súbor obsahuje JSON objekt, ktorý okrem samotného share môže obsahovať aj ďalšie informácie, potrebné na rekonštrukciu pôvodného súboru. Zodpovedajúci model vyzerá nasledovne:



Obr. 12: Datachest: Nahrávanie súborov. Vizualizácia spracovania nahrávaného súboru.

```
struct DatachestKeyShareFile: Codable {
    let keyShare: String
    let mappedFileData: DatachestMappedFileData?
}

struct DatachestMappedFileData: Codable {
    let fileId : String
    let aesTag: [Data]
    let aesNonce: String
    let fileType: String
}
```

Atribút `keyShare` je share kľúča. Tento atribút je prítomný v každom pomocnom súbore. Hodnota `mappedFileData` je optional (syntax `?` označuje, že atribút môže byť `nil`, viď [47]). Obsahuje ďalšie informácie: identifikátor šifrovaného súboru, nonce a pole tagov (potrebné parametre na dešifrovanie) a typ súboru (potrebné pri ukladaní dešifrovaného

súboru späť do mobilného zariadenia používateľa). Hodnotu `mappedFileData` obsahuje iba jeden pomocný súbor. Je to súbor, ktorý má byť nahratý na rovnaké úložisko, na ktorom je uložený aj používateľov pôvodný, zašifrovaný súbor (ďalej iba hlavný súbor). Ak by `mappedFileData` boli na inom cloude, než hlavný súbor, zvyšuje sa hrozba nemožnosti obnovenia zašifrovaného súboru. Pretože v prípade, že by nastal výpadok alebo strata dát na cloude, na ktorom sú prítomné `mappedFileData`, by nebolo možné dešifrovať hlavný súbor. Avšak, ak by `mappedFileData` boli prítomné na rovnakom cloude ako hlavný súbor, zlyhať môže ľubovoľný z dvoch zvyšných cloudov. Na zostrojenie kľúča stačia dva z troch shares kľúča (vyplýva z konfigurácie Shamirovej secret sharing schémy). Zvyšné kryptografické parametre a hlavný súbor sú taktiež k dispozícii. Takýmto prístupom minimalizujeme riziko straty alebo nemožnosti obnovenia dôležitých dát používateľa. Jediná zraniteľnosť ostáva v zlyhaní cloudu, kde je uložený hlavný súbor spolu s `mappedFileData`. To je riziko, ktoré naše riešenie akceptuje. Keďže pomocné súbory sú malej veľkosti (rádovo desiatky bajtov), odosielajú sa v celku. Odosielanie súborov v celku je pre každú cloud API definované v kapitole 4. Po úspešnom nahratí pomocných súborov obdržíme od API objekt s atribútmi novo nahratých súborov. Identifikátory pomocných súborov uchováme pre neskoršie použitie.

Po dokončení nahrávania hlavného súboru a pomocných súborov je potrebné zapísať do Firestore databázy vzájomné mapovanie zašifrovaného súboru a troch, k nemu prislúchajúcich súborov.

Zápis dodržuje dohodnutú štruktúru. Jednotlivé dokumenty existujú pod jedinou kolekciou `files`. Názov dokumentu je identifikátor nahratého, zašifrovaného súboru. Tento dokument obsahuje tri polia: `googleDriveShare`, `microsoftOneDriveShare` a `dropboxShare`. Každému poľu priradíme príslušný identifikátor jedného z troch pomocných súborov podľa toho, na akom cloude sú nahraté.

Príklad: Hlavný súbor má identifikátor `ABC`. K nemu prislúchajúce pomocné súbory majú identifikátory `123` (uložený na Google Drive), `B3F` (uložený na Microsoft OneDrive) a `JJL` (uložený na Dropboxe). Dokument pre tento scenár vyzerá nasledovne:

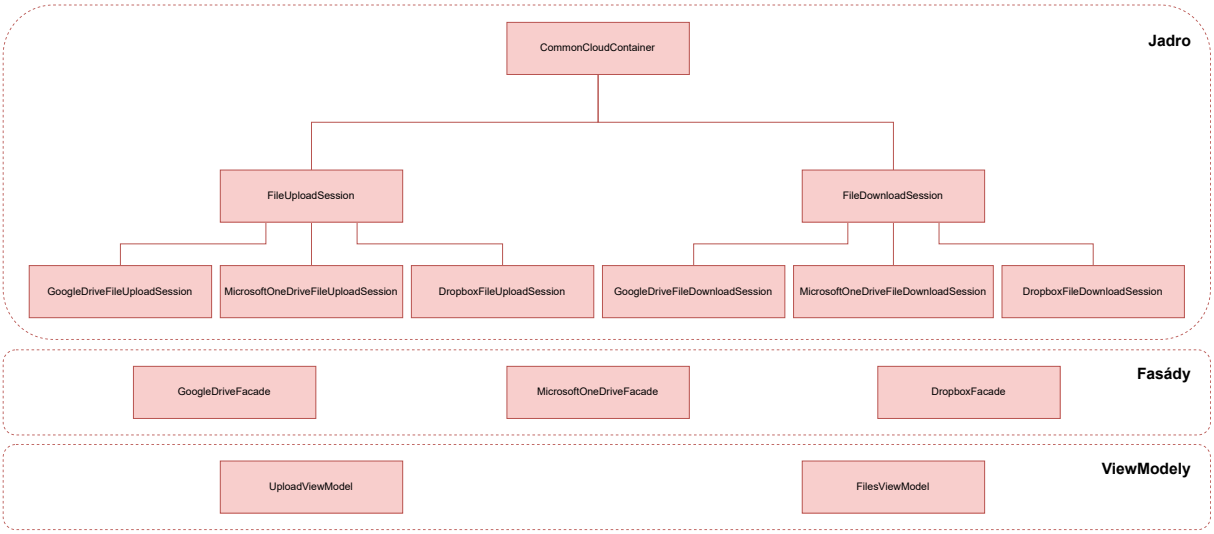
Názov dokumentu	ABC
Polia	dropboxShare: `"JJL"` googleDriveShare: `"123"` microsoftOneDriveShare: `"B3F"`

Týmto je proces nahrávania súboru zvoleného používateľom dokončený a nahrávacia relácia zaniká spolu s dočasnými dátami viazanými k nej (kľúč, nonce, tagy, prúd, resu-

mable upload URL a podobne). Opísaný postup platí pre súbory nahrávané na Google Drive. Presný postup nahrávania súboru sa u ostatných cloudov môže v malých detailoch líšiť, kvôli rozdielom v API jednotlivých cloudov.

5.6 Sťahovanie súborov

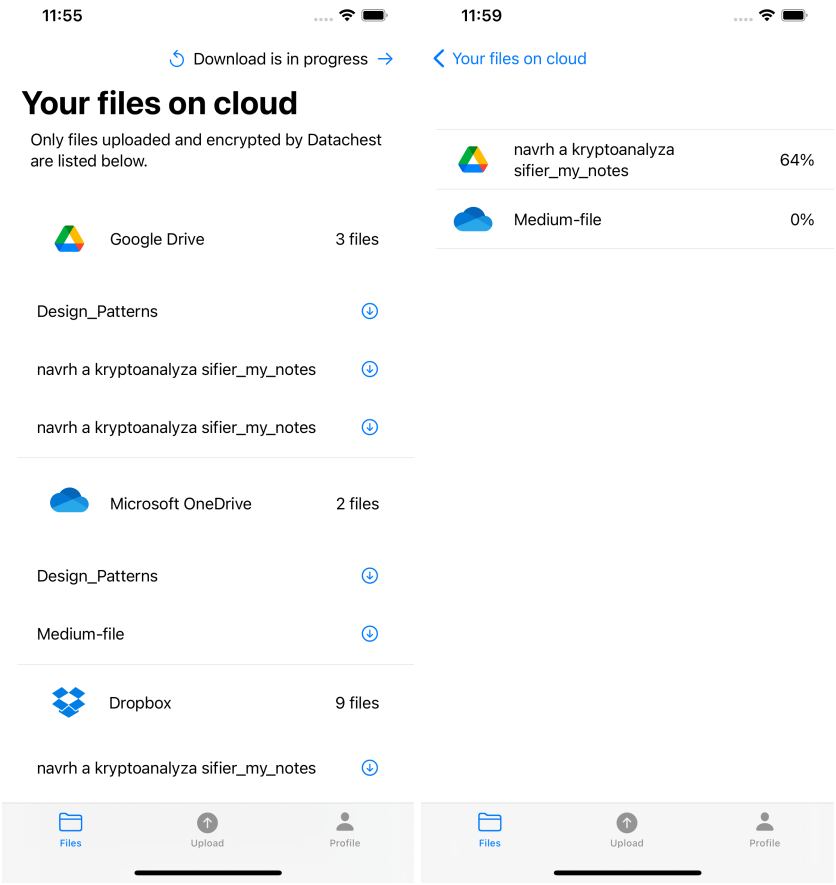
Sťahovanie súborov je komplexnejší proces ako nahrávanie, pretože prichádzajúci súbor musí byť spracovaný klientskou aplikáciou, nie cloudom. Vo väčšine krokov je postup analogický ako pri nahrávaní. Diagram z obr. 10 obohatíme o komponenty zodpovedné za sťahovanie súborov. Výsledný diagram je zobrazený na obr. 13. Z diagramu vidno, že architektúra je analogická ako pri nahrávaní. Pre každý cloud je definovaná trieda reprezentujúca reláciu sťahovania. Dedenie a význam dedenia je analogický ako pri nahrávaní. Teraz opíšeme postup sťahovania súboru.



Obr. 13: Datachest: Sťahovanie súborov. Podrobnejší diagram.

Z obrazoviek uvedených na obr. 14 je používateľ schopný sťahovať súbory z cloudu. Obrazovka vľavo zobrazuje zoznam súborov, ktoré používateľ nahral na cloudové úložiská pomocou aplikácie Datachest. Aplikácia zobrazuje používateľovi iba súbory prítomné v priečinku `Files` (viď podkapitola 5.4). Používateľovi je sprístupnená obrazovka aj v prípade, keď je prihlásený iba k dvom z troch poskytovateľov. Pre účet, z ktorého je odhlásený, neuvidí svoje súbory. Zo zvyšných cloudov avšak vie sťahovať súbory, napriek absencii tretieho cloud účtu. Z našej konfigurácie Shamirovej schémy vyplýva, že k rekonštrukcii súboru stačia dva shares z troch. Preto je sťahovanie aj v tomto prípade stále možné. V prípade menšieho počtu prihlásených cloud účtov, než dvoch, je obrazovka na obr. 14 neprístupná. Pre jednoduchosť opäť predpokladajme, že používateľ chce stiah-

nuť súbor z úložiska Google Drive. Používateľ kliknutím na tlačidlo sťahovania vyžiada aplikáciu o stiahnutie súboru do zariadenia.



Obr. 14: Datachest: Zoznam súborov s ich možnosťou stiahnutia (vľavo), zoznam aktuálne sťahovaných súborov (vpravo). Grafické rozhrania.

`FilesViewModel` získava z cloudov zoznamy nahratých súborov podľa tabuliek uvedených v kapitole 4. Tieto dáta čerpá View a zobrazuje ich na obrazovku (obr. 14, vľavo). Udalosť kliknutia na tlačidlo sťahovania daného súboru zachytáva ViewModel. ViewModel deleguje žiadosť fasáde `GoogleDriveFacade`. Fasáda inicializuje sťahovaciu reláciu vytvorením inštancie triedy `GoogleDriveFileDownloadSession`.

Konštruktor triedy `GoogleDriveFileDownloadSession` nastaví do parametrov identifikátor a názov hlavného súboru a veľkosť chunku sťahovaného súboru. Fasáda v ďalšom kroku spúšťa sťahovací proces volaním metódy `downloadFile()` nad `GoogleDriveFileDownloadSession`.

Prvým krokom metódy `downloadFile()` je získanie pomocných súborov, obsahujúcich údaje k dešifrovaniu hlavného súboru. Za túto operáciu je zodpovedná metóda `collectKeyShares()`, spoločná pre všetky cloudy. Preto je definovaná v rodičovskej triede

`FileDownloadSession`. Metóda stiahne príslušný share z daného cloudu iba vtedy, ak je doň používateľ prihlásený. Z UI/UX vyplýva, že `collectKeyShares()` získa vždy dva alebo všetky tri shares.

Získanie pomocných súborov začína získaním príslušného dokumentu z Firestore. Keďže názov dokumentu je identifikátor hlavného súboru, získanie správneho dokumentu je triviálna úloha. Po získaní dokumentu má aplikácia k dispozícii informácie, pod akými identifikátormi sú uložené pomocné súbory patriace k hlavnému súboru. Pomocné súbory sťahujeme volaním metódy service triedy `GoogleDriveService`, podľa tabuliek opisujúcich sťahovanie súborov v kapitole 4. Veľkosť pomocných súborov je zanedbateľne malá. Preto je bezpečné ich udržiavať vo vyrovnávacej pamäti zariadenia používateľa. Pomocné súbory sú prevedené do štruktúr podľa modelov definovaných v predchádzajúcej podkapitole. Zo štruktúry je možné pracovať s jednotlivými údajmi podľa potreby.

Po získaní a spracovaní pomocných súborov možno začať so sťahovaním hlavného súboru. Použijeme rovnakú metódu service triedy `GoogleDriveSerivce` ako pri sťahovaní pomocných súborov. Sťahovanie súborov ľubovoľnej veľkosti (nie väčšej, ako je zostávajúca kapacita disku zariadenia používateľa) je v réžii natívnej knižnice jazyku Swift, Foundation. Metóda `downloadTask()`, ktorú na tieto účely používame, je definovaná a bližšie opísaná v odkaze [48]. Hlavný súbor je stiahnutý na pevný disk. Cesta k súboru je definovaná Foundation knižnicou. Ide o špeciálnu cestu pre súbory s dočasnou existenciou na disku. Referencia na dočasne uložený súbor je z metódy vrátená ako výsledok procesu. Ak nie je využitá, súbor je odstránený z dočasného úložiska. V opačnom prípade možno so súborom pracovať, definovať mu cestu, kde bude trvalo uchovaný, vykonávať nad ním operácie a podobne.

Rekonštrukcia kľúča, dešifrovanie a uloženie hlavného súboru prebieha v metóde triedy `FileDownloadSession`, `decryptAndSaveFile()`. Najskôr nastavíme všetky parametre, potrebné na dešifrovanie šifrou AES v GCM móde, so 128 bitovým kľúčom. Shares sú spojené do kľúča secret sharing algoritmom[11]. Pripravia sa autentizačné tagy a nonce. Definuje sa cesta trvalého uchovania hlavného súboru a jeho typ. Následne je vytvorená dvojica prúdov: prvý, vstupný prúd, číta hlavný súbor, uložený v dočasnom úložisku pevného disku zariadenia po chunkoch s veľkosťou definovanou pri inicializácii relácie sťahovania.

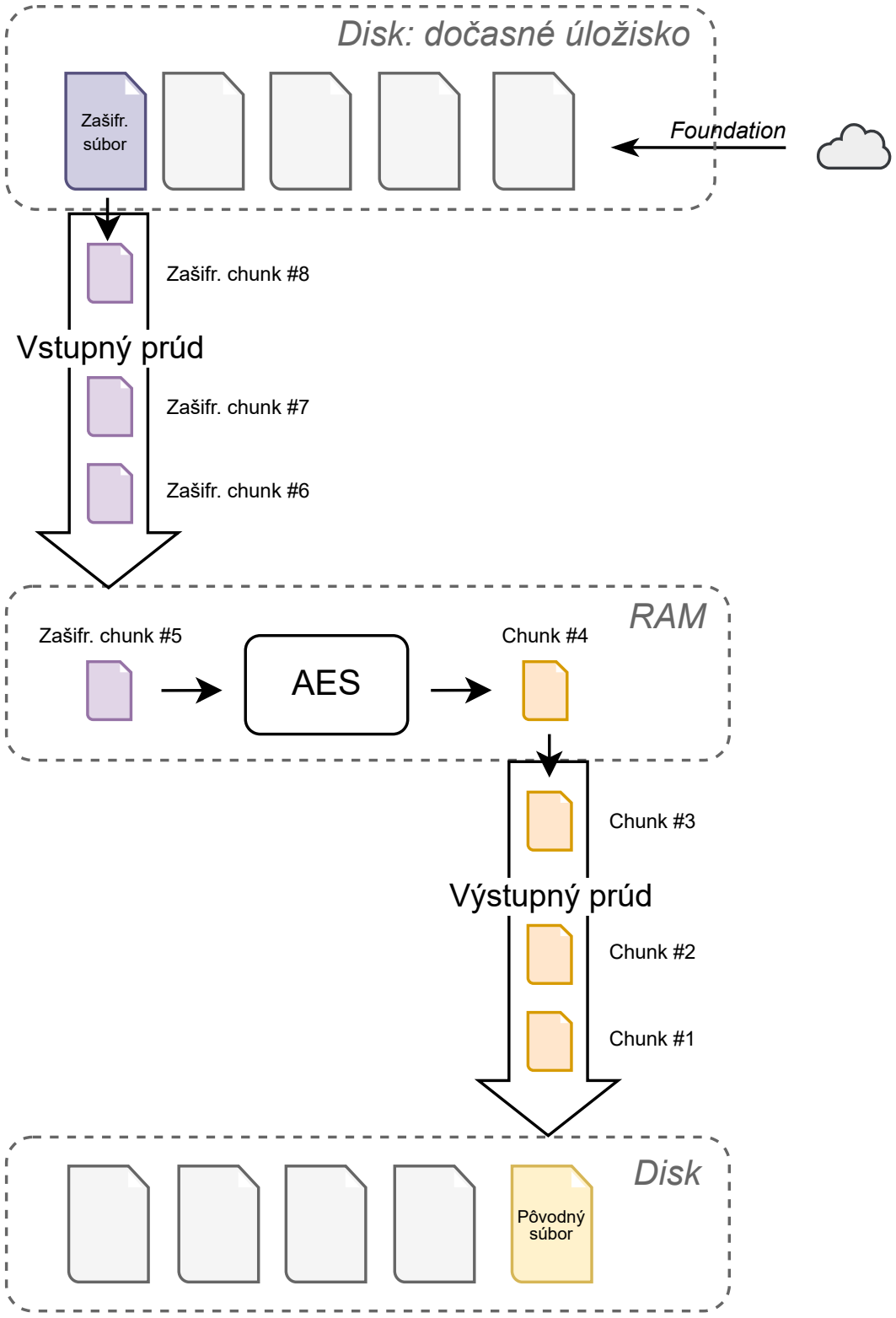
Poznámka: Je zrejmé, že veľkosť chunku musí byť rovnaká, ako pri nahrávaní súboru. V opačnom prípade by bol súbor rozdelený na iný počet chunkov ako pri šifrovaní a

[11] vstupom do algoritmu sú vždy iba dva shares, tretí v algoritme nepoužívame (z dôvodu neprítomnosti tretieho share v prípadoch, kedy používateľ nie je prihlásený do všetkých cloud účtov)

autentizácia šifrovaných blokov by zlyhala.

Chunky sú v sekvenčnom poradí overované príslušným tagom a dešifrované symetrickým kľúčom. Druhý, výstupný prúd, odchytáva spracované chunky a ukladá ich na definovanú cestu vo filesystéme mobilného zariadenia. Po zápise posledného chunku vzniká pôvodný súbor, v otvorenom tvare. Opísaný postup znázorňuje diagram na obr. 15.

Týmto je proces sťahovania súboru zvoleného používateľom dokončený a relácia sťahovania zaniká spolu s dočasnými dátami viazanými k nej (kľúč, nonce, tagy, prúdy a podobne). Opísaný postup platí pre súbory sťahované z Google Drive. Presný postup sťahovania súboru sa u ostatných cloudov môže v malých detailoch líšiť, kvôli rozdielom v API jednotlivých cloudov.



Obr. 15: Datachest: Sťahovanie súborov. Vizualizácia spracovania stiahnutého súboru.

6 Vyhodnotenie riešenia

V tejto kapitole sú predmetom diskusie prínosy implementovaného riešenia. Definujeme zostávajúce hrozby, problémy a možné implementačné vylepšenia. Zároveň vyhodnocujeme, či aplikácia vyriešila problémy definované v zadaní tejto práce.

6.1 Použiteľnosť

Aplikácia disponuje funkcionalitami, ktoré používateľovi umožňujú spravovať dôverné dáta. Docieľuje to mechanizmami ako:

- autentifikácia u všetkých cloud providerov
- komunikácia so všetkými cloud providermi pomocou API
- práca s kryptografickými nástrojmi ako secret sharing, šifrovanie, dešifrovanie
- spracovávanie súborov rôznych veľkostí pomocou prúdov

Aplikácia poskytuje grafické rozhranie, ktoré je intuitívne a smerodajné pre naplnenie potrieb používateľa. Je určená pre mobilné zariadenia, čo značí vysokú dostupnosť pre bežného používateľa. Týmito charakteristikami aplikácie sa podarilo splniť cieľ implementovania riešenia, ktoré bude dostupné, ľahko použiteľné a pochopiteľné pre bežného používateľa.

V našom riešení od používateľa očakávame, že mimo aplikácie Datachest rovnako používa natívne cloudové aplikácie od samotných cloudových poskytovateľov (aplikácie Google Drive [49], Microsoft OneDrive [50] a Dropbox [51]). V týchto aplikáciach avšak nebude schopný čítať súbory uložené prostredníctvom aplikácie Datachest v otvorenom tvare. Tento fakt prináša limitácie v použiteľnosti. Používateľ je teda nútený používať štyri aplikácie, podľa potreby. Rovnako platí, že v aplikácii Datachest používateľ nie je schopný čítať súbory iné, než v priečinku `/Datachest/Files`. V prípade, že by cez natívne aplikácie do tohto priečinka nahral ďalšie súbory v otvorenom tvare, ich stiahnutie by v Datacheste zlyhalo. A to preto, lebo pre tieto súbory neexistuje záznam vo Firestore.

K týmto úvahám doplníme, že používateľ s pokročilými znalosťami v informačných technológiách nie je viazaný k aplikácii Datachest, ak chce čítať ňou spracované súbory. Pomocou manuálne zostrojených API volaní vie získať hlavný súbor a pomocné súbory a zrekonštruovať pôvodný súbor. Tento scenár je možný za predpokladu znalosti:

- veľkosti chunku, akou má byť súbor sťahovaný

- identifikátorov pomocných súborov patriacich k hlavnému súboru, ktorý chce používateľ stianuť

6.2 Bezpečnosť

Citlivé dáta používateľa môžu byť v tomto riešení prítomné na cloudoch, ale aj v zariadení používateľa. Zariadenie používateľa disponuje operačným systémom, ktorý šifruje dáta uložené na pevnom disku. Kľúče spravuje v šifrovanej databáze zariadenia (Apple Keychain), alebo v hardvérovo samostatnej jednotke (Secure Enclave) [52]. Počas nahrávania súborov na cloud sa do otvoreného tvaru dostávajú dočasne jednotlivé chunky súborov, nahrávané do RAM zariadenia. Pre súbory uložené na cloudoch platia bezpečnostné záruky poskytnuté cloudovými poskytovateľmi. Servery poskytovateľov sú chránené proti známym útokom. Implementovanie middleware servera, ako bolo v pôvodonom návrhu, by z hľadiska bezpečnosti muselo zahŕňať podobnú ochranu. Komunikácia medzi klientom a cloudom je zabezpečený protokolom TLS. Poskytuje úvodnú autentizáciu a šifruje prenášané dáta. Šifrovanie je potrebné pri odosielaní shares kľúčov, pretože sú, narozdiel od súborov používateľa, odosielané v otvorenom tvare.

6.2.1 CIA

V tejto sekcii sa venujeme analýze bezpečnosti riešenia z hľadiska CIA. Naše riešenie, ktoré používa zdieľanie tajomstva, porovnávame s metódou správy osobných údajov bez zdieľania tajomstva. Analyzované sú v poradí: dôvernosť, integrita a dostupnosť.

Úroveň dôvernosti osobných údajov bola prostredníctvom zdieľania tajomstva zvýšená. Ak by sme nepoužili šifrovanie a dáta ukladáme na cloud v otvorenom tvare, spoliehame sa na bezpečnosť poskytovanú cloudom. Akonáhle sa útočník dostane do cloudového úložiska, vidí všetky citlivé dáta používateľa. Ak sú dáta šifrované a rolu tajomstva prenesieme na šifrovací kľúč, ktorý je zdieľaný, útočník vidí šifrované súbory a share kľúča. Z definície Shamirovej schémy zdieľania tajomstva platí, že útočník nezíska informáciu o plnom kľúči. Pri úspešnom útoku na dva cloudy by bola bezpečnosť prelomená.

Úroveň integrity ostáva nezmenená. Za celistvosť a neporušenosť dát na cloude zodpovedá cloudový poskytovateľ.

Úroveň dostupnosti bola prostredníctvom zdieľania tajomstva znížená. Ak tajomstvo nie je zdieľané medzi viaceré cloudy, k dostupnosti je vždy potrebný len jeden cloud. Ak je tajomstvo zdieľané medzi viaceré cloudy, k dostupnosti tajomstva potrebujeme prístup vždy k aspoň dvom cloudom (podľa nastavenia secret sharing schémy). Cloudy podporované aplikáciou Datachest patria medzi najpoužívanejšie (prieskum v sekcii 2.1). Preto má z praktického hľadiska scenár súčasného výpadku dvoch a viac cloudov zanedbateľne

malú pravdepodobnosť.

6.3 Implementácia a škálovateľnosť

Kvôli rozdielnostiam v API špecifikáciách jednotlivých cloudov bolo pri implementácii potrebné pristupovať ku každému cloudu osobitne. Každý cloud má v kóde vlastné triedy a modely. Združovanie operácií pod spoločné bloky kódu je možné v minime prípadov.

Z týchto dôvodov spôsobí pridanie nového cloudu do aplikácie značný zásah do kódu. Pre nový cloud by bolo potrebné vytvoriť: fasádu, service triedy, atribúty v store, autentifikačný mechanizmus a správu tokenov, modely, grafické rozhranie a podobne. Tento proces by vedel zjednodušiť middleware server. Oddelil by značnú časť logiky od klientskej aplikácie.

Počas implementácie nebolo možné použiť knižnice jednotlivých cloudov, ktoré uľahčujú komunikáciu s API. Knižnice neumožňujú so súborom pracovať počas jeho nahrávania. Pre naše účely by súbor musel byť šifrovaný ešte pred začatím nahrávania. Bolo by nutné vytvoriť kópiu pôvodného súboru, ktorá by bola šifrovaná. Na to je potrebný dodatočný úložný priestor a čas. Preto nahrávanie súborov spravujeme v kóde bez použitia knižníc. To nám umožňuje súbor šifrovať v priebehu nahrávania, bez potreby vytvárania šifrovanej kópie. Počas vykonávania aplikácie je efektívne využívaný čas aj úložný priestor. Nevýhodou je veľké množstvo kódu a robustná architektúra.

Záver

Hlavnou témou tejto práce je Shamirova schéma zdieľania tajomstva. Hľadali sme riešenie, ktoré by túto schému využilo, za účelom vyššej ochrany osobných údajov používateľa. Venovali sme sa oblasti cloudových úložísk a ich bezpečnosti.

Implementovali sme mobilnú aplikáciu Datachest, ktorá komunikuje s troma cloudovými úložiskami. Používateľ do nich prostredníctvom aplikácie môže nahrávať svoje súbory. Aplikácia ich počas nahrávania šifruje a kľúč rozdelí ako tajomstvo Shamirovej schémy. Jednotlivé shares kľúča distribuuje medzi cloudy. Zároveň vedie evidenciu, ktoré tajomstvá patria ku ktorým šifrovaným súborom. Používateľ môže cez aplikáciu šifrované súbory spätne stiahnuť do úložiska mobilného zariadenia. Aplikácia na pozadí prevezme dostupné shares kľúča, a ak ich je dostatočný počet, rekonštruuje kľúč, dešifruje súbor a uloží ho do zariadenia.

Nami implementované riešenie spĺňa požiadavky bezpečnosti. Poskytuje vyššiu úroveň ochrany dôverných dát, pokiaľ majú byť uchovávané na cloude. Rieši problém ukladania a správy kľúčov (vysvetľujeme v 1.1). Riešenie poskytuje grafické rozhranie, ktoré je pre používateľa prehľadné a intuitívne. Je dostupné pre bežného používateľa. Nevyžaduje pokročilé technické znalosti.

Aplikácia má potenciál pre ďalšie vylepšenia. Implementácia middleware servera by oddelila low-level časť architektúry aplikácie a najmä komunikáciu s cloudami. Existenciou middleware servera by bolo možné efektívne implementovať ďalšie klientské aplikácie (Android, PC, web). Do aplikácie je možné integrovať viac cloudov a poskytnúť tak väčší výber pre používateľa. Zaujímavé by bolo aj rozšírenie grafického rozhrania o živé ukážky súborov (dokumenty, obrázky, videá, hudba a iné) bez potreby ich sťahovania do úložiska telefónu. Navrhnuté úpravy by však vyžadovali väčší tím programátorov a UI/UX dizajnérov, čo by bolo možné realizovať napríklad vo forme start-up firmy.

Zoznam použitej literatúry

1. MENEZES, A., OORSCHOT, P. van a VANSTONE, S. *Handbook of Applied Cryptography*. CRC Press, 1996. ISBN 0-8493-8523-7.
2. STINSON, Douglas R. *Cryptography: Theory and Practice, Third Edition (Discrete Mathematics and Its Applications)*. University of Waterloo, Ontario, Canada: Chapman & Hall/CRC Taylor & Francis Group, 2006. ISBN 978-1-58488-508-5.
3. SHAMIR, Adi. How to share a secret. *Communications of the ACM*. 1979, roč. 22, č. 11, s. 612–613.
4. *Number of consumer cloud-based service users worldwide in 2013 and 2018* [online]. Statista Research Department, 2014 [cit. 2021-11-09]. Dostupné z: `https://www.statista.com/statistics/321215/global-consumer-cloud-computing-users/`.
5. *Usage & Trends of Personal Cloud Storage: GoodFirms Research* [online]. GoodFirms, 2021 [cit. 2021-11-09]. Dostupné z: `https://www.goodfirms.co/resources/personal-cloud-storage-trends`.
6. *What is Dropbox?* [Online]. Dropbox, 2021 [cit. 2021-11-09]. Dostupné z: `https://www.dropbox.com/features`.
7. *Get started with encrypted files in Drive, Docs, Sheets & Slides* [online]. Google, 2021 [cit. 2021-11-09]. Dostupné z: `https://support.google.com/docs/answer/10519333?hl=en`.
8. *Cloud locations* [online]. Google, 2021 [cit. 2021-11-09]. Dostupné z: `https://cloud.google.com/about/locations#network`.
9. *Dropbox Security Issues 2021: The Good, the Bad & the Ugly* [online]. Cloudwards, 2021 [cit. 2021-11-09]. Dostupné z: `https://www.cloudwards.net/dropbox-security/`.
10. *Millions of websites offline after fire at French cloud services firm* [online]. Reuters, 2021 [cit. 2021-11-09]. Dostupné z: `https://www.reuters.com/article/us-france-ovh-fire-idUSKBN2B20NU`.
11. *Apple toughens iCloud security after celebrity breach* [online]. BBC News, 2021 [cit. 2021-11-09]. Dostupné z: `https://www.bbc.com/news/technology-29237469`.
12. *Archistar core* [online]. GitHub, 2015-08 [cit. 2021-10-13]. Dostupné z: `https://github.com/archistar/archistar-core`.

13. *Making secret sharing based cloud storage usable* [online]. Emerald Publishing, 2019-06 [cit. 2021-10-13]. Dostupné z: https://www.emerald.com/insight/content/doi/10.1108/ICS-01-2019-0016/full/pdf?title=making-secret-sharing-based-cloud-storage-usable.

14. *PRISMACLOUD: PRIvacy & Security MAintaining Services in the CLOUD* [online]. PRISMACLOUD, 2017 [cit. 2021-10-13]. Dostupné z: https://prismacloud.eu.

15. PRAVEEN, K., INDU, G., SANTHYA, R. a SETHUMADHAVAN, M. An Android Application for Secret Image Sharing with Cloud Storage. 2017. Dostupné z DOI: 10.1007/978-981-10-6898-0_33.

16. THIEN, Chih-Ching a LIN, Ja-Chen. Secret image sharing. *Computers & Graphics*. 2002, roč. 26, s. 765–770. Dostupné z DOI: 10.1016/S0097-8493(02)00131-0.

17. AL-HUSAINY, Mohammed. A novel image encryption algorithm based on the extracted map of overlapping paths from the secret key. *RAIRO - Theoretical Informatics and Applications*. 2016, roč. 50. Dostupné z DOI: 10.1051/ita/2016023.

18. *What is a REST API?* [Online]. Red Hat, 2020 [cit. 2022-04-09]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api.

19. *OAuth 2.0* [online]. OAuth, 2012 [cit. 2022-04-09]. Dostupné z: https://oauth.net/2/.

20. *OAuth Access Tokens* [online]. OAuth, 2012 [cit. 2022-04-09]. Dostupné z: https://oauth.net/2/access-tokens/.

21. *Files: create* [online]. Google, 2022 [cit. 2022-04-13]. Dostupné z: https://developers.google.com/drive/api/v3/reference/files/create.

22. *Upload file data* [online]. Google, 2022 [cit. 2022-04-10]. Dostupné z: https://developers.google.com/drive/api/guides/manage-uploads.

23. *Files: get* [online]. Google, 2022 [cit. 2022-04-13]. Dostupné z: https://developers.google.com/drive/api/v3/reference/files/get.

24. *Download Files* [online]. Google, 2022 [cit. 2022-04-13]. Dostupné z: https://developers.google.com/drive/api/guides/manage-downloads.

25. *Create a new folder in a drive* [online]. Microsoft, 2021 [cit. 2022-04-15]. Dostupné z: https://docs.microsoft.com/en-us/onedrive/developer/rest-api/api/driveitem_post_children?view=odsp-graph-online.

26. *Upload or replace the contents of a DriveItem* [online]. Microsoft, 2021 [cit. 2022-04-15]. Dostupné z: https://docs.microsoft.com/en-us/onedrive/developer/rest-api/api/driveitem_put_content?view=odsp-graph-online.

27. *Upload large files with an upload session* [online]. Microsoft, 2021 [cit. 2022-04-15]. Dostupné z: https://docs.microsoft.com/en-us/onedrive/developer/rest-api/api/driveitem_createuploadsession?view=odsp-graph-online.

28. *List children of a driveItem* [online]. Microsoft, 2021 [cit. 2022-04-16]. Dostupné z: https://docs.microsoft.com/en-us/onedrive/developer/rest-api/api/driveitem_list_children?view=odsp-graph-online.

29. *Download the contents of a DriveItem* [online]. Microsoft, 2021 [cit. 2022-04-16]. Dostupné z: https://docs.microsoft.com/en-us/onedrive/developer/rest-api/api/driveitem_get_content?view=odsp-graph-online.

30. *Dropbox for HTTP Developers* [online]. Dropbox, 2022 [cit. 2022-04-17]. Dostupné z: https://www.dropbox.com/developers/documentation/http/documentation#files-create_folder.

31. *Dropbox for HTTP Developers* [online]. Dropbox, 2022 [cit. 2022-04-17]. Dostupné z: https://www.dropbox.com/developers/documentation/http/documentation#files-upload.

32. *Dropbox for HTTP Developers* [online]. Dropbox, 2022 [cit. 2022-04-18]. Dostupné z: https://www.dropbox.com/developers/documentation/http/documentation#files-upload_session-start.

33. *Dropbox for HTTP Developers* [online]. Dropbox, 2022 [cit. 2022-04-18]. Dostupné z: https://www.dropbox.com/developers/documentation/http/documentation#files-upload_session-append.

34. *Dropbox for HTTP Developers* [online]. Dropbox, 2022 [cit. 2022-04-18]. Dostupné z: https://www.dropbox.com/developers/documentation/http/documentation#files-upload_session-finish.

35. *Dropbox for HTTP Developers* [online]. Dropbox, 2022 [cit. 2022-04-19]. Dostupné z: https://www.dropbox.com/developers/documentation/http/documentation#files-list_folder.

36. *Dropbox for HTTP Developers* [online]. Dropbox, 2022 [cit. 2022-04-19]. Dostupné z: https://www.dropbox.com/developers/documentation/http/documentation#files-download.

37. *iOS Design Patterns: MVVM* [online]. SWIFTLY RUSH, 2022 [cit. 2022-04-23]. Dostupné z: https://www.swiftlyrush.com/ios-design-patterns-mvvm/.

38. *Codable* [online]. Apple Developer, 2022 [cit. 2022-04-23]. Dostupné z: https://developer.apple.com/documentation/swift/codable.

39. *Reactive state for Angular* [online]. NgRx, 2022 [cit. 2022-04-23]. Dostupné z: https://ngrx.io/.

40. *Singleton pattern* [online]. Wikimedia Foundation, 2022 [cit. 2022-04-23]. Dostupné z: https://en.wikipedia.org/wiki/Singleton_pattern.

41. *Google Sign-In for iOS* [online]. GitHub, 2022 [cit. 2022-04-19]. Dostupné z: https://github.com/google/GoogleSignIn-iOS.

42. *Microsoft Authentication Library for iOS and macOS* [online]. GitHub, 2022 [cit. 2022-04-19]. Dostupné z: https://github.com/AzureAD/microsoft-authentication-library-for-objc.

43. *Dropbox for Swift* [online]. GitHub, 2022 [cit. 2022-04-19]. Dostupné z: https://github.com/dropbox/SwiftyDropbox.

44. *Keychain Services* [online]. Apple Developer, 2022 [cit. 2022-04-19]. Dostupné z: https://developer.apple.com/documentation/security/keychain_services.

45. *Firebase helps you build and run successful apps* [online]. Firebase, 2022 [cit. 2022-04-24]. Dostupné z: https://firebase.google.com.

46. *Cloud Firestore: Store and sync app data at global scale* [online]. Firebase, 2022 [cit. 2022-04-24]. Dostupné z: https://firebase.google.com/products/firestore.

47. *Optional* [online]. Apple Developer Documentation, 2022 [cit. 2022-04-25]. Dostupné z: https://developer.apple.com/documentation/swift/optional.

48. *Optional* [online]. Apple Developer Documentation, 2022 [cit. 2022-04-25]. Dostupné z: https://developer.apple.com/documentation/foundation/urlsession/1411511-downloadtask.

49. *Google Drive on the App Store* [online]. Apple Inc., 2022 [cit. 2022-04-30]. Dostupné z: https://apps.apple.com/us/app/google-drive/id507874739.

50. *Microsoft OneDrive on the App Store* [online]. Apple Inc., 2022 [cit. 2022-04-30]. Dostupné z: https://apps.apple.com/us/app/microsoft-onedrive/id477537958.

51. *Dropbox: Cloud Photo Storage on the App Store* [online]. Apple Inc., 2022 [cit. 2022-04-30]. Dostupné z: `https://apps.apple.com/us/app/dropbox-cloud-photo-storage/id327630330`.

52. *Secure Enclave* [online]. Apple Inc., 2022 [cit. 2022-05-04]. Dostupné z: `https://support.apple.com/en-gb/guide/security/sec59b0b31ff/web`.

Prílohy

A Zdrojový kód

/datachest-ios

· priečinok obsahujúci zdrojový kód softvérovej implementácie aplikácie Datachest

B Inštalačná a používateľská príručka

Zdrojový kód k aplikácii Datachest je dostupný z Prílohy A alebo z odkazu: `https://github.com/petercurikjr/datachest-ios`. Na otestovanie riešenia je potrebné:

- mať nainštalovaný operačný systém macOS Catalina, alebo vyššie
- mať nainštalované vývojové prostredie Xcode a jeho najnovšiu verziu

Poznámky: Riešenie bolo vyvíjané na operačnom systéme macOS Catalina a testované na emulovanom zariadení iPhone 12 Pro Max. Kompatibilita projektu bola testovaná aj na macOS Monterey, s úspešným výsledkom. V prípade chybného správania aplikácie je v niektorých prípadoch vhodné aplikáciu vypnúť a znovu skompilovať.

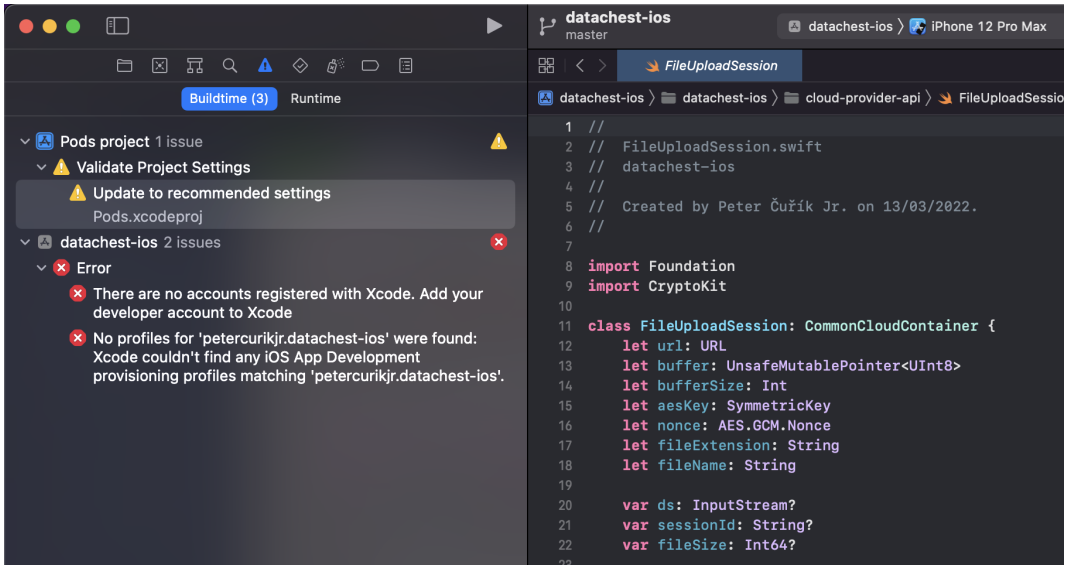
Postup:

- Súbor `datachest-ios.xcworkspace` sa nachádza v koreňovom priečinku zdrojového kódu aplikácie. Otvorte ho. Projekt by sa mal načítať do prostredia Xcode.
- Vyberte emulované zariadenie, na ktorom sa má spustiť aplikácia. Kliknite v hornej strednej časti na zariadenie (Obr. B.1). Odporúčame iPhone 12 Pro Max, na ktorom bola aplikácia testovaná.
- Spustite aplikáciu pomocou tlačidla s bielym trojuholníkom (všeobecne známe ako tlačidlo „play"). Tlačidlo sa nachádza v ľavej hornej sekcii vývojového prostredia (Obr. B.2). Spustenie je neúspešné. V ľavom bočnom menu by ste mali vidieť jeden warning a dva errory (Obr. B.2).
- Kliknite na warning s názvom *Update to recommended settings*. Zobrazí sa vám pop-up okno (Obr. B.3). Kliknite na *Perform Changes*. Warning by mal zmiznúť.
- Na vyriešenie uvedených errorov je potrebné v Xcode nastaviť Apple ID. V *Xcode -> Preferences...*, v sekcii *Accounts* kliknite na tlačidlo so znakom „+", vľavo dole. Zvoľte *Apple ID* (Obr. B.4) ako typ účtu a prihláste sa[12].
- Spustite aplikáciu znova. Aplikáciu sa podarilo spustiť. V aplikácii Datachest kliknite na sekciu *Upload* (dole), prihláste sa do všetkých cloud účtov a kliknite na jeden z *Upload* tlačidiel, patriacim k jednotlivým cloudom.

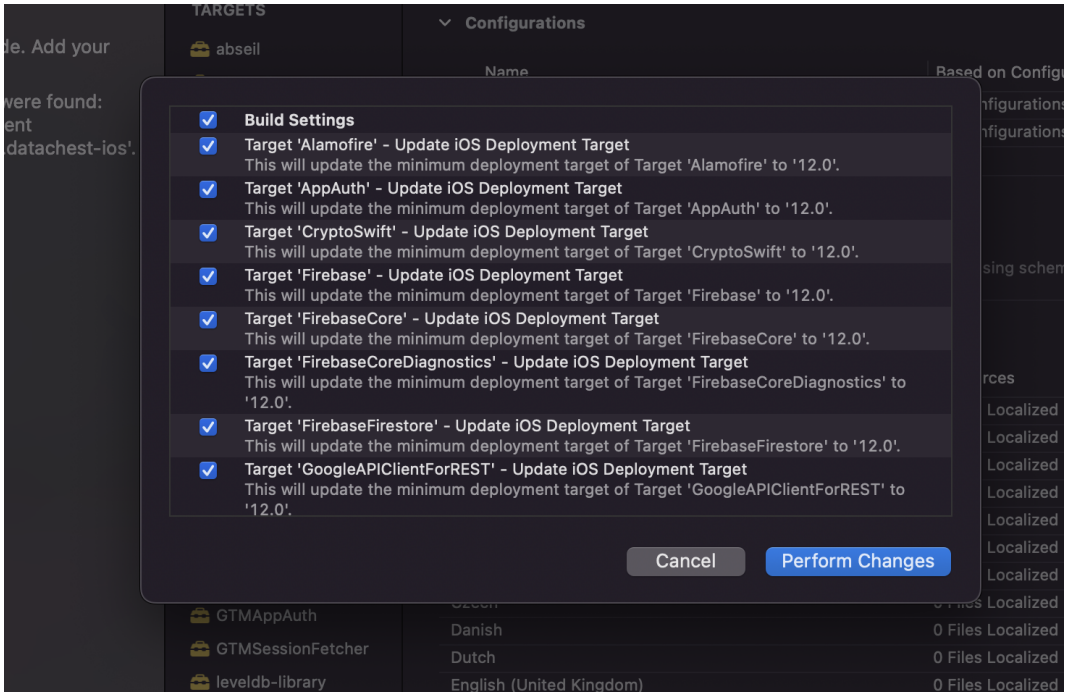
[12]Účet Apple ID musí mať aktivovaný bezplatný developer program. Viac na odkaze `https://developer.apple.com`.

- Zobrazí sa modálne okno, kde je možné zvoliť súbor k nahratiu. V prípade, že emulované zariadenie nemá žiadne súbory, pretiahnite zo systému počítača ľubovoľný súbor do modálneho okna virtuálneho iPhone zariadenia pomocou metódy drag and drop. Súbor sa uloží do filesystému mobilného zariadenia a následne ho možno použiť na nahrávanie z aplikácie Datachest.
- Aplikáciu možno v tomto bode ľubovoľne používať.

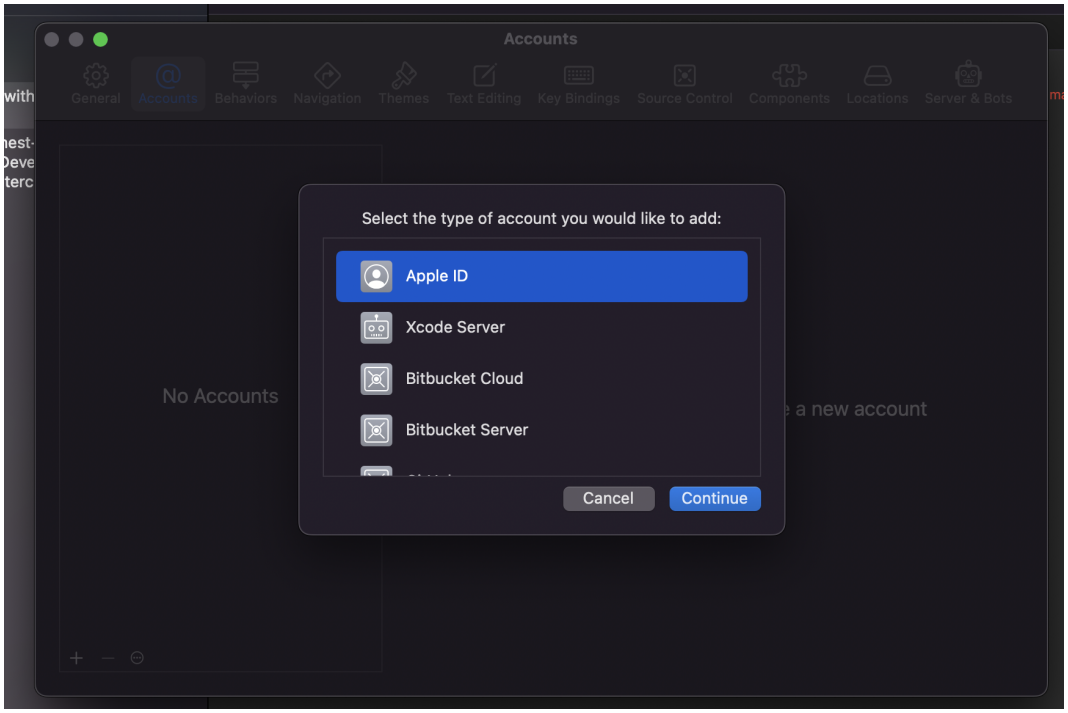
Obr. B.1: Výber emulovaného iOS zariadenia v Xcode.



Obr. B.2: Rozhranie vývojového prostredia Xcode.



Obr. B.3: Okno warning hlášky *Update to recommended settings* v Xcode.



Obr. B.4: Nastavenie účtu v Xcode.