VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a
komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Ivan Chernikau

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

OCHRANA SOUKROMÍ V CLOUDU

PRIVACY PROTECTION IN CLOUD

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Ivan Chernikau
AUTHOR

VEDOUCÍ PRÁCE — Ing. Petr Dzurenda
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Ivan Chernikau ***ID:*** 146017

Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Ochrana soukromí v cloudu

POKYNY PRO VYPRACOVÁNÍ:

V rámci diplomové práce student provede základní srovnání současných technik pro zajištění ochrany soukromí uživatelských dat v prostředí cloudu, jako je plně homomorfní šifrování (full homomorphic encryption), dělení dat (data splitting) a vyhledávatelné šifrování (searchable encryption). Na základě této analýzy student vybere vhodný protokol poskytující ochranu soukromí uživatelských dat v cloudu a implementuje jej do funkčního systému.

DOPORUČENÁ LITERATURA:

[1] DZURENDA, P.; HAJNÝ, J. Techniky homomorfního šifrování a jejich praktické využití. Elektrorevue Internetový časopis (http://www.elektrorevue.cz), 2014, roč. 16, č. 2, s. 54-60. ISSN: 1213- 1539.

[2] CALVIÑO, Aida; RICCI, Sara; DOMINGO-FERRER, Josep. Privacy-preserving distributed statisticalcomputation to a semi-honest multi-cloud. In:Communications and Network Security (CNS), 2015 IEEE Conference on. IEEE, 2015. p. 506-514.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: Ing. Petr Dzurenda ***Konzultant:***

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

Abstrakt

V rámci diplomové práce byla popsána problematika ochrany soukromí dat uložených v cloudech, které v současnosti uživatele často využívají. Některé z popsaných problémů mohou být vyřešené pomocí homomorfního šifrování, dělení dat nebo vyhledávatelného šifrování. Výše uvedené techniky byly popsány a porovnány z pohledu bezpečnosti, efektivity a ochrany soukromí. Následně byla zvolena technika dělení dat a byla implementována knihovna v programovacím jazyce C. Výkonnost implementace byla porovnána s výkonností AES šifrování/dešifrování. Takže byla navržena a implementována aplikace realizující bezpečné ukládání uživatelských dat do cloudu. Aplikace využívá implementovanou metodu data splitting a aplikaci CloudCross. Navržená aplikace umožňuje práci jak přes konzolové (CLI), tak grafické rozhraní (GUI). GUI rozšiřuje možnosti CLI o registraci aplikace do cloudu a automatickou detekci typu použitého cloudu. Ukládání a nahrávání souborů do/z cloudu je zcela transparentní a nezatěžuje tak uživatele technickými detaily implementace techniky rozdělení dat.

Klíčová slova

Homomorfní šifrování, dělení dat, vyhledávatelné šifrování, cloud, bezpečnost, ochrana soukromí

Abstract

In the Master's thesis were described privacy protection problems while using cloud technologies. Some of the problems can be solved with help of homomorphic encryption, data splitting or searchable encryption. These techniques were described and compared by provided security, privacy protection and efficiency. The data splitting technique was chosen and implemented in the C language. Afterwards a performance of the implemented solution was compared to AES encryption/decryption performance. An application for secured data storing in cloud was designed and implemented. This application is using the implemented data splitting technique and third-party application CloudCross. The designed application provides command line interface (CLI) and graphical user interface (GUI). GUI extends the capabilities of CLI with an ability to register cloud and with an autodetection of registered clouds. The process of uploading/downloading the data to/from cloud storage is transparent and it does not overload the user with technical details of used data splitting technique.

Key words

Homomorphic encryption, Data splitting, Searchable encryption, Cloud, Security, Privacy protection

CHERNIKAU, I. *Ochrana soukromí v cloudu.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 62 s. Vedoucí diplomové práce Ing. Petr Dzurenda.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma „Ochrana soukromí v cloudu“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonu (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této diplomové práci byl realizovaný v laboratořích podpořených projektem Centrum senzorických, informačních a komunikačních systémů (SIX); registrační číslo CZ.1.05/2.1.00/03.0072, operačního programu Výzkum a vývoj pro inovace.

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Petru Dzurendovi za odborné vedení, konzultace a podnětné návrhy k práci.

V Brně dne ....

podpis autora

Obsah

SEZNAM OBRÁZKŮ

ÚVOD

Asi nás překvapí, jak dlouhou historii mají cloudové technologie. I když prudký vývoj nastal v posledních deseti letech, myšlenka se objevila mnohem dříve. Joseph Carl Robnett Licklider (J. C. R., 1915-1990) byl jedním z předních amerických odborníků v oblasti počítačových věd (computer science). V článku z r. 1960 [1] vyjádřil myšlenku, že budou existovat nějaká centra, ke kterým budou mít přes nějaký komunikační kanál přístup uživatelé, a uživatelé budou sdílet hardware a software v těchto centrech, proto si rozdělí i jejich cenu. Ale v té době to nebylo možné realizovat a vypadalo to jen jako science fiction. Až v roce 1989 Tim Berners-Lee [2] přišel s novou myšlenkou a vytvořil první webovou stránku. Tím se přiblížil okamžik, kdy se nápad J.C.R. mohl stát realitou. V devadesátých letech 20. století se začaly objevovat první služby přes internet, avšak neměly žádné společné jméno. To se změnilo kolem roku 2006, kdy se slovo *cloud* začalo používat i v literatuře.

Cloudové technologie se dají rozdělit na dvě logické části: *cloudové výpočty* (cloud computing) a *cloudová úložiště* (cloud storage).

Cloudové výpočty představují model vývoje a používání počítačových technologií založené na internetu. Jednodušeji řečeno je to poskytování služeb, aplikací či hardwaru uživatelům pomocí vzdálených serverů přístupných přes internet. Uživatel neplatí za vlastnictví žádného hardwaru ani softwaru, ale platí jenom za jeho využití. V dnešní době existuje mnoho poskytovatelů, například: Microsoft Azure, Google Cloud Platform, Amazon Elastic Compute Cloud (Amazon EC2) apod.

Cloudová úložiště se liší od cloudových výpočtů tím, že poskytují prostředky pro uložení uživatelských dat. Uživatelská data se uchovávají na velkém počtu distribuovaných serverů. Na rozdíl od uložení dat na vlastních serverech při užívání cloudového úložiště se uživatel nemusí starat o infrastrukturu a nemusí znát žádné technické detaily. Z pohledu uživatele jsou cloudová úložiště jeden vzdálený server, ke kterému má přístup přes internet. Poskytovatel cloudu přitom zaručí, že data jsou vždy přístupná a dostatečně zabezpečená. Mezi největší poskytovatele patří: Google Drive, Dropbox, OneDrive, Amazon Glacier.

Cloudové technologie přinášejí pro uživatele mnoho výhod, jako je nižší cena, vzdálená dostupnost všude, kde je internet, snadnost instalace, nebo možnost využití nejnáročnějších technologií bez nutnosti provozu vlastní drahé infrastruktury. Ale každá výhoda má i svoji cenu. V případě cloudových technologií uživatel platí svým soukromím a nižší bezpečností uložených dat.

Pro uživatele cloudu je jeho poskytovatel považován za nedůvěryhodnou třetí stranu. Pokud si uživatel nepřeje, aby měl poskytovatel cloudu přístup k jeho otevřeným citlivým datům, musí si je sám uživatel samostatně zašifrovat již na své klientské straně. S tím jsou následně spojeny další problémy, jako je správa klíčů, neschopnost aplikací v cloudu provést zpracování zašifrovaných dat, nebo nutnost dešifrovat data před jejich použitím.

Tato práce se zabývá studií současných technik zvyšujících ochranu soukromí uživatelských dat uložených v cloudu jak vůči případnému útočníkovi, tak vůči samotnému poskytovateli cloudové služby. Dále jsou rozebrány problémy, se kterými přijde uživatel do styku, a jejich existující řešení.

V rámci diplomové práce byla v programovacím jazyce C implementována technika rozdělení dat. Zvolená technika umožňuje bezpečné ukládání dat do cloudu způsobem splňujícím základní

požadavky na ochranu soukromí takto uložených dat. Byl proveden výkonnostní test, který porovnává rychlost techniky, rozdělení dat a šifrování.

Přímé použití techniky rozdělení dat by uživateli způsobilo komplikace při použití cloudových úložišť. Proto k tomu byla implementována konsolová aplikace (ve skriptovacím jazyce Bash) a aplikace s grafickým rozhraním (v programovacím jazyce C++).

1. Cloudové technologie

Cloudové technologie se dají rozdělit na dvě logické části: *cloudové výpočty* (cloud computing) a *cloudová úložiště* (cloud storage). V této kapitole jsou obecně popsány principy a výhody jejich využití.

Cloudové technologie sdílí následující vlastnosti:

- Hodně se používá virtualizace, aby přidělování/odebírání prostředků probíhalo co nejefektivnějším způsobem
- Prostředky jsou sdílené mezi různými uživateli
- Cloudové technologie jsou zpřístupněny přes internet
- Uživatel si může z nabídky poskytovatele cloudu vybrat prostředky, které potřebuje
- Nevyužité prostředky mohou být automaticky odebrány a v případě potřeby mohou být automaticky přiděleny uživateli
- Poskytovatel cloudu eviduje, jaké prostředky a jak dlouho byly využívány uživatelem, a na základě toho naúčtuje cenu
- Poskytovatel cloudu zajišťuje, aby veškerý software byl vždy aktualizován

1.1. Cloudová úložiště

Aby se nějaká technologie hodně rozšířila, musí poskytovat řešení problému, který trápí mnoho uživatelů. V dnešní době má celosvětově přístup k počítačům a na internet kolem 40 % lidí. A je naprosto neuvěřitelné, jaké množství digitálních dat vytvářejí každý den internetoví uživatelé. Každý z uživatelů si přeje svoje data někam bezpečně uložit a nepřijít o ně. Mnoho problémů s tím souvisejících řeší právě cloudová úložiště.

Pod pojmem *cloudové úložiště* chápeme virtuální prostor na serverech třetí strany, který se využívá především k uložení dat, přičemž přístup k datům je zajištěn přes internet.

Výhody využití cloudového úložiště:

- Cloud je schopen poskytnout možnost pro uložení většího objemu dat, než by si uživatel sám mohl dovolit ve svém počítači
- Přístup k datům v cloudu je uživateli umožněn odkudkoli, kde má přístup na internet
- Přes internet a cloud se dají data efektivně sdílet mezi velkým množstvím uživatelů, a navíc uživatelé mohou s takto uloženými daty spolupracovat
- Poskytovatel cloudu zajistí, aby v případě hardwarových problémů uživatel nepřišel o svá data
- Uživatel platí jenom za to místo na úložišti, které využívá, a ne za celkovou kapacitu serveru, kterou nevyužije
- Uživatel se nestará o koupi, podporu a údržbu infrastruktury na uložení dat. To dělá poskytovatel cloudu centralizovaně, což má za následek snížení ceny
- Poskytovatel cloudu zajistí redundanci a integritu uživatelských dat, a to bez jakékoliv účasti uživatele
- Cloud může poskytovat prostředky pro verzování uživatelských dat.

1.2. Cloudové výpočty

Pod pojmem *cloudové výpočty* rozumíme poskytování služeb, aplikací či hardwaru uživatelům pomocí vzdálených serverů přístupných přes internet. Uživatel neplatí za žádný hardware ani software, ale platí jenom za jeho využití.

Existuje několik různých modelů nasazení cloudu, ale nejpoužívanější jsou následující [3]:

- ***veřejný*** (*Public cloud computing*) – jedná se o schéma, v němž je služba poskytnuta široké veřejnosti
- ***soukromý*** *(Private cloud computing*) – služba je provozována soukromou organizací pouze pro své interní uživatele
- ***hybridní*** (*Hybrid cloud computing*) – vzniká vlastně pomocí kompozice několika různých typů cloudů

Poskytovatelé cloudu si mohou vybrat, jaký typ služby budou poskytovat, jestli budou poskytovat hardware, software, anebo jejich kombinaci. Cloudy se tedy rozlišují podle distribučního modelu. Nejrozšířenější modely jsou [3]:

- ***IaaS*** – *infrastruktura jako služba (Infrastructure as a Service)* – poskytovatel cloudu vlastní infrastrukturu (hardware) a poskytuje ji jako službu. Uživatelé vlastní a starají se o operační systémy, aplikace běžící na poskytované infrastruktuře apod.
- ***PaaS*** – *platforma jako služba (Platform as a Service)* – poskytovatel cloudu poskytuje platformu, na které může uživatel vyvinout, nasadit a udržovat vlastní aplikace
- ***SaaS*** – *software jako služba (Software as a Service)* – poskytovatel cloudu vlastní infrastrukturu, operační systém, aplikace a uživateli pronajímá jenom přístup k aplikacím (uživatelé si tedy nekupují aplikaci samotnou)

Výhody využití cloudových výpočtů:

- uživatel potřebuje menší znalosti na to, aby nasadil a použil aplikace, infrastrukturu či platformu
- uživatel má možnost rychlého zvýšení výkonu (stačí jen doplatit)
- přístup ke cloudu je uživateli umožněn odkudkoli, kde má přístup na internet
- poskytovatel cloudu zajistí, aby v případě problémů uživatel nepřišel o svá data
- uživatel platí jenom za ten výpočetní výkon, který využije, a ne za celkovou kapacitu serveru
- uživatel se nestará o koupi, podporu a údržbu infrastruktury/platformy/aplikace
- uživatel šetří náklady na provoz (například elektřinu)

2. Problémy cloudových technologií

Daná kapitola je plně věnována popisu problémů bezpečnosti v cloudu, jelikož bezpečnost a ochrana soukromí představují hlavní nevýhody cloudu. Některé firmy mají velice striktní bezpečnostní politiku, která jim vůbec neumožňuje používat cloudové technologie.

2.1. Bezpečnost uložených dat

Pod pojmem *prostor pro útok* rozumíme počet potenciálních možností styku dat s útočníky [20].

1. Když jsou data distribuována, zvětší se riziko neautorizovaného fyzického přístupu k datům. V cloudu jsou data neustále replikována a přenášena z jednoho fyzického místa na jiné, což vede k dramatickému zvýšení pravděpodobnosti neautorizovaného přístupu k datům (likvidace starého HW, znovuvyužití disku a další).
2. Počet lidí, kteří mají fyzický a elektronický přístup k datům a kteří mohou data zkompromitovat (mohou být podplaceni, nebo donuceni) se dramaticky zvýší.
3. Zvyšuje se počet sítí, přes které se data přesouvají, a tím se zvyšuje pravděpodobnost, že data budou nějakým způsobem zneužita.
4. Poskytovatel cloudu poskytuje svoje služby mnoha uživatelům, čímž se zvyšuje pravděpodobnost, že se k datům dostane někdo jiný než samotný uživatel.

2.2. Stabilita cloudové služby

Jelikož je pro firmu poskytovatel cloudu třetí stranou, nese pro ni přesun dat a služeb do cloudu následující rizika [4]:

1. poskytovatel cloudu může zkrachovat
2. poskytovatel cloudu se může rozšířit a změnit své zaměření a poskytované služby
3. poskytovatel cloudu může být koupen větší firmou
4. poskytovateli cloudu se může kompletně zhroutit infrastruktura

2.3. Dostupnost cloudové služby

Pro soukromé uživatele i organizace je v neposlední řadě důležité, jak rychle se dá přistupovat k datům či službám.

1. Kvalita přístupu k datům či službám velice záleží na rychlosti a kvalitě přístupu na internet.
2. Na poli se vyskytuje další hráč, poskytovatel internetového připojení, se všemi z toho vyplývajícími riziky.

2.4. Ochrana soukromí uživatelských dat

V dnešní době se velké množství citlivých a osobních dat zpracovává digitálně a jsou cílem útočníků. Proto je důležité si uvědomit, jaká rizika z pohledu ochrany soukromí cloud nese [4]:

1. větší riziko, že citlivá data budou zkompromitována, protože se zvětšil prostor pro útok
2. poskytovatel cloudu bude cílem útočníků, protože je bodem shromáždění dat od spousty spolu nesouvisejících uživatelů
3. poskytovatel cloudu může mít přístup k vašim citlivým datům (pokud jsou uloženy v nešifrované podobě)
4. problémy s legislativou (například v případě serverů s fyzickou lokací v jiném státě)
5. uživatel nemá pod kontrolou řízení bezpečnosti a je nucen slepě věřit poskytovateli cloudu, že jeho bezpečnostní politika je silná a že se dodržuje

3. Zabezpečení uživatelských dat v cloudu

Mnoho lidí a organizací využívá cloud bez ohledu na to, že je to spojeno s bezpečnostními riziky, a především s tím, že uživatel ztratí kontrolu nad vlastními daty. Proto se tématu zabezpečení těchto dat věnuje stále více lidí a snaží se najít řešení.

Současné přístupy k zabezpečení dat v cloudu jsou:

- ***šifrování na straně cloudu*** – ochrana dat před útočníkem
- ***šifrování na straně klienta*** – ochrana dat před útočníkem a poskytovatelem cloudu

Šifrování dat na straně cloudu znamená, že šifrovaná data a klíče jsou známy poskytovateli cloudu, takže má přístup k citlivým datům klienta. Časová náročnost šifrování a dešifrování na straně cloudu není pro uživatelé cloudu téměř viditelná. Například Dropbox ukládá data v šifrované podobě (používá symetrickou šifru AES s 256 bitovým klíčem) [5]. Google Drive ukládá data taky v šifrované podobě, ale na rozdil od Dropboxu, videa mohou být uložena v nešifrované podobě [6].

Šifrování dat na straně klienta by uspokojilo téměř všechny požadavky na bezpečnost a ochranu soukromí, toto řešení je však pro klienta časově náročné a je spojeno s komplikací při využití dat v cloudu. Takový přístup má čtyři velké nevýhody:

1. Operace šifrovaní a dešifrovaní jsou časově náročné
2. Uživatel musí nějakým způsobem uchovávat a spravovat klíče
3. V šifrovaných datech se špatně hledá, a proto bude uživatel nucen si pro jakoukoliv operaci stáhnout lokální kopii dat a dešifrovat je
4. Poskytovatel cloudových služeb není schopen pracovat s daty uloženými v cloudu (například umožnit uživateli v datech vyhledávat)

Jelikož šifrování dat na straně klienta není jednoduchý proces, už existují komerční řešení, jako Boxcryptor [7] anebo Tresorit [8], které udělají proces šifrování dat na straně klienta téměř neviditelným.

Pro soukromého uživatele je tento proces příliš složitý (i kdyby měl nějak bezpečnostně uchovávat jen pár klíčů, anebo využívat navíc ještě jeden další software pro šifrování) a pro firmy je to časově nevýhodné. Proto se hledá alternativní řešení. Existuje několik technik, které by mohly tím či oním způsobem pomoci řešit tyto problémy:

- techniky ***rozdělení dat*** (Data splitting)
- techniky využívající ***vyhledávatelné šifrování*** (Searchable encryption)
- techniky využívající ***homomorfní šifrování*** (Homomorphic encryption)

3.1. Rozdělení dat (Data splitting)

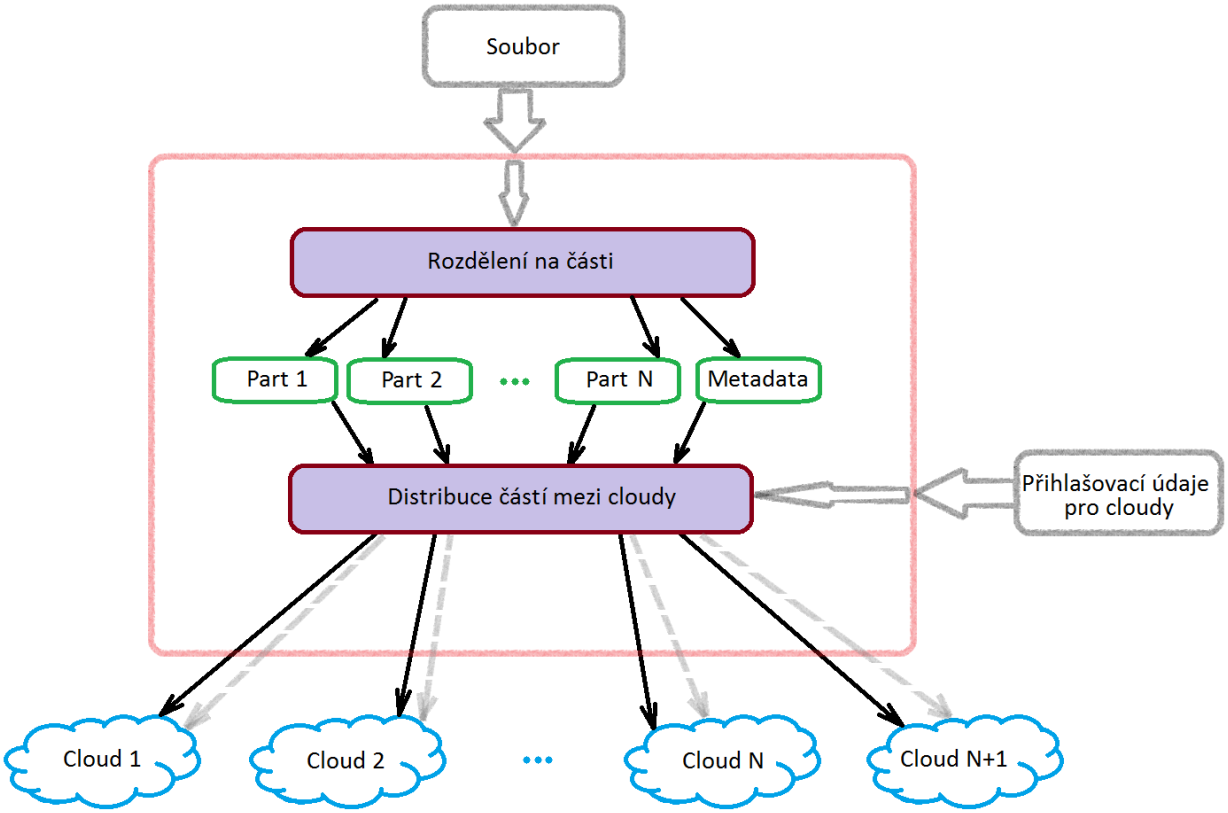
Z výše uvedených technik má jednodušší základní myšlenku technika rozdělení dat. Pomocí téhle techniky je možné zajistit ochranu soukromých a citlivých dat tím, že data jsou rozdělena a uložena do více cloudů. Základním předpokladem je, že jednotliví cloud poskytovatelé mají pouze část dat, a tudíž nemohou data zneužít, protože nejsou kompletní. Dalším podstatným předpokladem bezpečnosti je, že cloud poskytovatelé vzájemně nespolupracují a jsou čestní, případně neznají

jeden druhého. V opačném případě by byli schopni data uživatele zkompletovat a následně zneužít. Z tohoto důvodu nelze považovat techniku rozdělení dat za stejně bezpečnou jako šifrování, kdy klíč k šifrovaným datům vlastní pouze uživatel.

Princip zabezpečení dat pomocí techniky rozdělení dat je následující. Předpokládáme, že chceme zabezpečit nějaký soubor. Pomocí zvolené metody rozdělíme soubor na N nečitelných částí (nějaké metody navíc by vyžadovaly ještě jednu, kterou tvoří metadata). Pod pojmem metadata rozumíme informaci potřebnou ke složení souboru z částí, aby si uživatel nemusel pamatovat, který soubor reprezentuje, kterou část a kde je uložen. Soubor s metadaty by nebyl moc velký a dal by se zabezpečit pomocí šifrování.

Každá samotná část sama o sobě nedává žádný smysl, proto i když ji útočník získá, nemá pro něho žádný přínos, pokud nebude mít všechny ostatní části (a popřípadě i metadata). Síla techniky rozdělení dat je schovaná v metodě rozdělení dat na části viz [9], [10].

Dalším krokem je vzít dané části a uložit je do cloudů. Když už máme soubor rozdělený na několik částí a máme možnost vybírat mezi několika poskytovateli cloudu, dá se toho využít a uložit kusy do cloudu od různých cloud poskytovatelů. Jediné, co by si měl uživatel pamatovat, je to, kde se nachází metadata, aby pak mohl soubor složit zpátky.



Obr. 3.1: Použití techniky rozdělení dat k zabezpečení souborů v cloudu

Podíváme-li se z pohledu uživatele, zjistíme, že tento proces není úplně jednoduchý a potřebuje hodně manipulace. Aby se proces rozšířil, je potřeba ho celý automatizovat, viz *obr. 3.1*. Například by mohla existovat aplikace, která by běžela lokálně u uživatele (když aplikace běží lokálně, otevřená citlivá/osobní data nejsou přenášena po síti, a proto útočník nemá téměř žádnou šanci útok provést). Aplikace by uměla rozdělit soubor na části, připojit se k různým

poskytovatelům cloudu a uložit tam části souboru (uživatel by musel zadat přihlašovací údaje k jednotlivým cloudovým úložištím) a byla by open-source (aby bylo možné mít přístup ke zdrojovému kódu). Také by se dala zajistit redundance uložení dat (na *obr. 3.1* je to znázorněno šedými šipkami).

Technika rozdělení dat je vhodná pro cloudová úložiště a pro cloudové výpočty. Při využití v cloudovém úložišti je cílem rozdělit data na části tak, aby jedna samotná část pro cloud poskytovatele nebo útočníka neměla žádný význam. V cloudových výpočtech by se rozdělení dat používalo především k rozdělení dat na smysluplné části pro další nezávislé zpracování různými cloudy [11].

3.2. Vyhledávatelné šifrování (Searchable encryption)

Ve většině případů, jsou data v cloudu uložena zašifrovaně. Uživatel může svá data zašifrovat sám před uložením, anebo může využít služby cloudu a nechat poskytovatele cloudu zašifrovat data po uložení.

Šifrováním dat se zvýšila úroveň bezpečnosti, ale tím uživatel přišel o jednoduchou, ale velice užitečnou operaci s daty. Jak teď uživatel může ve svých datech na serveru vyhledávat pomocí klíčového slova, aby mohl efektivně pracovat se svými daty uloženými v cloudu? Server nemůže provádět hledání podle klíčových slov nad zašifrovanými daty. Proto uživatel bude nucen stáhnout všechny soubory lokálně, dešifrovat je a pak jen provést hledání podle klíčového slova. Takový způsob není efektivní, protože uživatel opravdu nepotřebuje všechny soubory, ale jen malý počet z nich, kde se nachází hledané slovo.

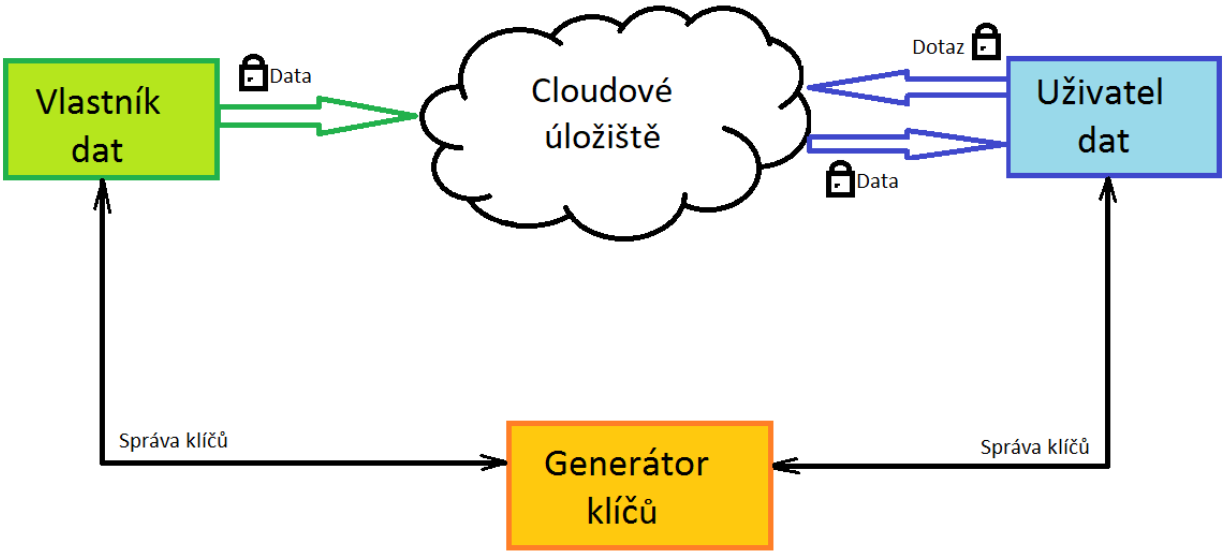
Technika, která bude umožňovat bezpečný a efektivní přístup k datům, má zajistit [12], že:

- uživatel může vyhledávat podle klíčového slova v zašifrovaných datech
- takové hledání nesmí odhalit serveru obsah zašifrovaných dat
- takové hledání nesmí odhalit serveru hledané klíčové slovo
- takové hledání nesmí odhalit serveru jakoukoliv vedlejší informaci o datech (například velikost souboru)

Šifrování, které umožňuje splnění výše uvedených požadavků, se nazývá *vyhledávatelným šifrováním (searchable encryption – SE).*

3.2.1. Architektura vyhledávatelného šifrování

SE dovoluje uživatelům vygenerovat z hledaného slova hledací token, pomocí kterého je server v cloudu schopen najít a vrátit uživateli data, která splňují jeho požadavek. Hledací token reprezentuje zašifrovaný dotaz na zašifrovaná data. Základní myšlenka SE je znázorněna na následujícím schématu *obr. 3.2*.



Obr. 3.2: Architektura vyhledávatelného šifrování

Architektura SE se skládá ze čtyř hlavních částí [12]:

- ***Vlastník dat*** generuje data, šifruje je pomocí kryptografického schématu, které podporuje SE, a ukládá je do cloudu.
- ***Uživatel dat*** generuje zašifrované požadavky na hledání v šifrovaných datech uložených v cloudu. Jako odpověď na svůj požadavek dostává zašifrovaná data, která odpovídají požadavku. Aby uživatel dat mohl vlastní data využít, potřebuje je dešifrovat. V systému může být více než jeden uživatel dat. Jeden člověk může mít roli vlastníka dat a zároveň uživatele dat.
- ***Generátor klíčů*** je důvěryhodná entita, která je zodpovědná za generování, správu a distribuci klíčů mezi vlastníkem dat a uživateli dat.
- ***Cloudové úložiště*** poskytuje možnost uložit a poslat data zpátky svým zákazníkům. Umí zpracovávat zašifrované vyhledávací požadavky na zašifrovaná data a vracet hledaná data uživateli dat. Cloud poskytovatel nesmí po provedení téhle operace získat žádnou informaci.

3.2.2. Bezpečnostní požadavky na vyhledávatelné šifrování

Šifrování dat má za cíl transformovat data tak, aby pouze oprávněná osoba vlastnící dešifrovací klíč byla schopna data přečíst. Když se do kryptografického schématu zahrne požadavek na to, že nějaká neautorizovaná třetí strana (v našem případě poskytovatel cloudového úložiště) umí v šifrovaných datech vyhledávat, můžeme tím prozradit nějakou informaci o samotných datech. Proto se při návrhu SE má dát pozor na následující požadavky [12]:

- ***získaná data*** – server nesmí rozlišit mezi samotnými dokumenty a určit vyhledávané položky
- ***hledací požadavek (token)*** – server nesmí získat z hledacího požadavku žádná data o klíčovém slově. Podle tokenu může server získat jenom ukazatele na zašifrovaná data, která obsahují hledané klíčové slovo
- ***generování tokenů*** – server nesmí být schopen vygenerovat token. Token je schopen vygenerovat jenom uživatel dat

- ***výsledek zpracovaní tokenu*** – server nesmí získat žádnou informaci o obsahu dat, která jsou výsledkem zpracování požadavku
- ***přístupové šablony*** – server se nesmí dozvědět o posloupnosti a frekvenci přístupu k dokumentům
- ***šablony tokenů*** – server se nesmí dozvědět, jestli dva různé tokeny představují hledání podle stejného klíčového slova

3.2.3. Druhy vyhledávatelného šifrování

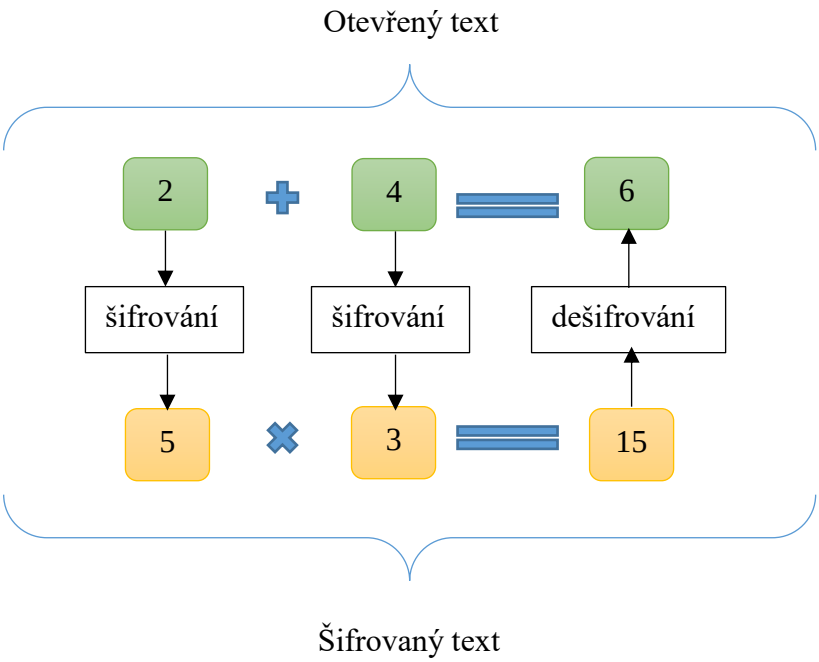
Jestliže uživatel chce zachovat svoje soukromí, tak bude chtít zašifrovat data před jejich uložením do cloudu. V tenhle okamžik uživatel má na výběr dvě možnosti a každá z nich má své výhody a nevýhody:

- ***Asymetrické vyhledávatelné šifrování*** — *Asymmetric searchable encryption (ASE).* Asymetrické schéma se zvolí, když je potřeba oddělit roli *vlastníka dat* od role *uživatele dat*. V asymetrickém schématu se využívá asymetrické šifrování. Při tom existují dva scénáře využití. První případ umožňuje vice uživatelů v roli *uživatele dat* (tito dostanou public key), ale jenom jeden uživatel s rolí *vlastníka dat* (ten dostane private key). Druhý případ je právě naopak, uživatelů s rolí *vlastníka dat* je hodně (tito dostanou public key) a uživatel v roli *uživatele dat* je jenom jeden (ten dostane private key). Ale za možnost mít v systému více než jednoho uživatele platíme vysokou cenu. Asymetrické šifrovaní je velice pomalé. To ovlivní proces samotného uložení dat do cloudu a protáhne čekání mezi okamžikem posílání tokenu na hledání a okamžikem, kdy dokumenty budou stažené a připravené pro používání (dešifrované). A tak dlouhé časy čekání nedovolují tomuto schématu se široce rozšířit. [12]
- ***Symetrické vyhledávatelné šifrování*** – *Symmetric searchable encryption (SSE).* Symetrické schéma se zvolí, když není potřeba oddělovat roli *uživatele dat* od role *vlastníka dat*. V SSE se využívá symetrické šifrování. Protože symetrické šifrování je rychlé, tenhle přístup má širší využití než ASE, ale pro ještě větší rozšíření je omezující fakt, že umožňuje vytvořit bezpečný systém jenom pro jednoho uživatele. [12]

3.3. Homomorfní šifrování (Homomorphic encryption)

Šifrování samo o sebe umožňuje utajit citlivou informaci. Ale v dnešní době už se objevily i další požadavky a homomorfní šifrování (Homomorphic encryption – HE) může uspokojit jeden z nich. HE umožňuje pracovat nad zašifrovanými daty bez nutnosti jejich dešifrování. Tahle technika by byla vhodná pro cloudová úložiště i pro cloudové výpočty. Pomoci HE schématu se dají zašifrovat data tak, že existuje možnost provádění operací nad šifrovanými daty bez znalosti dešifrovacího klíče.

Hlavní myšlenka je znázorněna následujícím příkladem na *obr. 3.3*. Řekneme, že v našem případě operaci sčítaní nad otevřenými daty odpovídá operace násobení nad zašifrovanými daty. Takže pro šifrovaná data (známe jenom kryptogramy „5“ a „3“) díky vlastnosti homomorfního šifrování můžeme spočítat kryptogram, který by odpovídal součtu otevřených textů, aniž bychom museli zašifrována data dešifrovat.



Obr. 3.3: Znázornění hlavního principu FHE

Částečně homomorfní šifrování – *Partially homomorphic encryption (PHE)* – podporuje jenom jednu operaci nad šifrovanými daty, buď sčítání nebo násobení a jejich počet může být neomezený.

Například široce známé schéma RSA je homomorfní vůči operace × (násobení). Další částečně homomorfní šifrovací schéma je: ElGamal, Paillier [14].

Poněkud homomorfní šifrování – *Somewhat homomorphic encryption (SHE)* – podporuje obě operace nad šifrovanými daty – sčítaní i násobení, ale jenom jejich omezený počet. FHE na rozdíl od SHE podporuje neomezený počet operací sčítání a násobení [14].

Šifrovaný text, který produkuje dosud známé SHE, obsahuje šum. Čím více homomorfních operací se provádí, tím je šum silnější a nastane okamžik, kdy je šum tak velký, že zašifrovaný text se nedá dešifrovat. Bootstrapping je mechanismus, který dovoluje kontrolovat šum vzniklý provedením jedné operace.

Plně homomorfní šifrování – *Fully Homomorphic Encryption (FHE)* – je šifrovací schéma, které podporuje libovolné výpočty nad šifrovanými daty. Takové schéma umožňuje vytvoření programu s libovolnou požadovanou funkcionalitou, který by na vstup dostával šifrovaná data a na výstup produkoval také smysluplná šifrovaná data. Takový program by mohl běžet v nedůvěryhodném prostředí, protože by nikdy nedešifroval vstupní data. Existence časově efektivního FHE by vyřešila některé problémy spojené s cloudovými technologiemi a posunula by je na vyšší úroveň.

V roce 2005 Dan Bonh, Eu-Jin Goh a Kobbi Nissim dokázali sestrojit částečné homomorfní schéma [15], které podporovalo libovolný počet sčítaní, ale jenom jedno násobení. Zdálo se, že myšlenka o existenci plně homomorfního šifrování bude brzy uskutečněna. To se stalo v roce 2009, kdy Craig Genty ve své dizertační práci „A Fully Homomorphic Encryption Scheme“ [16] popsal konstrukci plně homomorfního schématu pomocí ideální bodové mříže (ideal lattice) a představil ji na konferenci v [17].

V roce 2010 se objevila nová práce Graiga Gentryho vytvořená v kolektivu spolu s Martenem van Dijkem, Shaiem Halevim a Vinodem Vaikuntanathanem [18], kde autoři představili další způsob, jak zkonstruovat FHE. Důležité je, že nová konstrukce byla jednodušší než předchozí, protože už nebyla založena na ideální bodové mříži, ale na číslech. Dále v letech 2011-2012 Zvika Brakerski, Craig Gentry a Vinod Vaikuntanathan prezentovali další konstrukce FHE, které jsou více efektivnější, ale stále jako základní kámen využívají přístup popsaný v dizertační práci Craiga Gentryho. Na bázi teoretických výsledků začaly vznikat implementace výše uvedených schémat. Ale testy ukázaly, že ještě nejsou tak efektivní, aby se daly prakticky využít.

Homomorfní vlastnost šifrovacích schémat je velmi zásadní, jestliže jsou data uložena a zpracovávána mimo infrastrukturu vlastníka (například cloudové technologie, systémy elektronických voleb, systémy elektronických peněženek).

SHE, které zvládne vypočítat svůj vlastní dešifrovací algoritmus, se nazývá ***bootstrappable SHE***. Algoritmus, který převede bootstappable SHE na FHE se nazývá ***bootstrapping***.

Jestliže narazíme na překážku, vždycky se ji snažíme eliminovat. Jedinou překážkou pro SHE schéma je narůstající šum. Úplně eliminovat šum nesmíme, protože jinak by šifrovací schéma nebylo bezpečné, proto je potřeba tento šum čas od času pouze zmenšovat. Takže hlavním úkolem algoritmu bootstrapping je periodické snížení šumu.

V praxi se ukázalo, že zkonstruovat bootstrappable SHE není jednoduché. Craig Gentry ve své disertační práci k tomu využíval velice složité techniky. Zároveň se Gentry začal věnovat tomu, jak sestrojit bootstrappable SHE schéma jednodušším způsobem.

Craig Gentry a Shai Halevi představili popis implementace FHE (STOC 2009) a použitých optimalizací [19]. Ale výsledky testů ukázaly, že představenou implementaci v dnešní podobě nemá smysl nasazovat v produkci. Čas provedení jedné bootstrapping operace pro vysokou úroveň bezpečnosti může dosahovat až k 30 minutám. Dalším problémem je velikost klíčů. Malé klíče mají velikost kolem 70 MB, ale velké mohou přesáhnout i 2 GB.

Jan Hajný a Petr Dzurenda představili další implementace FHE [14]. Časová náročnost využití této implementace pro vysokou úroveň bezpečnosti je také překážkou pro použití v produkci (generování klíčů dosahovalo 45 minut a doba rešifrování byla 35 minut).

4. Návrh řešení

V předchozích kapitolách 3.1, 3.2 a 3.3 byly popsané techniky, které přispívají k ochraně soukromí uživatele v cloudu. Homomorfní šifrování a vyhledávatelné šifrování z důvodů složitosti a náročnosti výpočtů nemohou být v dnešní době široce použitelné. Proto byl k implementaci vybrán mechanismus rozdělení dat.

Každá aplikace se vytváří k tomu, aby vyřešila nějaký problém, se kterým se setkávají uživatelé. Naše řešení umožní uživateli používat techniku rozdělení dat k bezpečnému uložení dat v cloudu. Použití techniky rozdělení dat nese v sobě následné celkem komplikované použití. Navržené řešení automatizuje složité věci anebo poskytne uživateli jednoduché rozhraní. Tahle kapitola popisuje obecnou architekturu a dílčí části, ze kterých sestává implementované řešení. Řešení bylo implementováno pro Linux.

4.1. Dílčí části řešení

Naším cílem je, aby uživatel byl schopen zabezpečit data na klientské straně pomocí techniky rozdělení dat. Proto první část implementovaného řešení je tvořena implementací jedné varianty techniky rozdělení dat.

Při operacích nahrávání do cloudu a stahování souborů z cloudu chceme odstranit to, aby uživatel ručně manipuloval na začátku s první částí a pak s druhou částí souboru. Proto je druhá část řešení tvořena skripty, které od uživatele skryjí informaci, že soubor byl rozdělen na části a vlastně se ukládá do několika cloudů.

Používání aplikace v příkazové řádce omezuje potenciální počet uživatelů. Proto třetí část řešení je grafické uživatelské rozhraní, které navíc poskytuje vizualizaci pro seznamy souborů, které má uživatel lokálně a které má v cloudu.

4.1.1. Rozdělení souboru na části

Rozdělení souboru na části bylo implementováno jako konzolová aplikace Doit. Samotný algoritmus rozdělení souboru na části a jeho implementace jsou popsané v kapitole 5. Konzolová aplikace, která se řídí vstupními argumenty (viz kapitola 5.4), umí rozdělit soubor na dvě části i složit soubor ze dvou částí. Byl proveden výkonnostní test a jeho výsledky jsou popsané v kapitole 5.5.

4.1.2. Konzolová aplikace

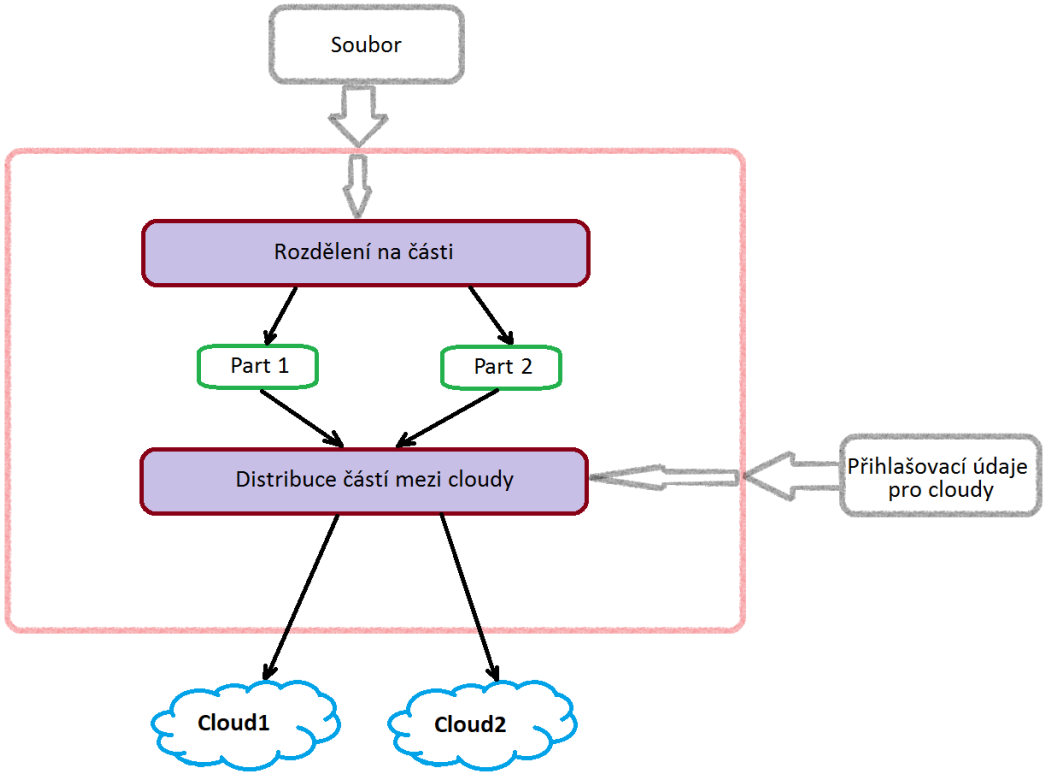
Druhá část řešení je tvořena skripty ***combine_pull.sh*** a ***split_add.sh*** (viz kapitola 6). Tato skripta poskytují konzolový interface pro synchronizaci adresáře (nahrávání a stahování souborů) s cloudem. Interně skripty používají konzolovou aplikaci *Doit* k rozdělení/složení souboru a program *CloudCross* po přímou komunikaci s cloudem (více o programu *CloudCross* viz kapitola 6.1.1). Skripty mohou být použity uživateli, kteří nechtějí nebo nemohou používat GUI (grafické uživatelské rozhraní). Skript ***combine_pull.sh*** se má používat pro stahování souborů z cloudu a ***split_add.sh*** pro nahrávání. Registrace cloudu (viz kapitola 6.2.2) a vytvoření souborové struktury (viz kapitola 6.2.1) při použití jenom skriptů bez GUI se má provádět ručně.

4.1.3. Grafické uživatelské rozhrání (GUI)

Grafické rozhraní (viz kapitola 7) poskytuje vizualizaci pro proces synchronizace vybraného adresáře s cloudem. GUI je nadstavbou nad CLI s rozšířením možností. V GUI je automatizovaná registrace cloudů a vytvoření souborové struktury. V grafickém rozhraní je kladen důraz na zjednodušení použití a vizualizaci souborů v lokálním adresáři a v cloudu (ukazuje soubory, které byly uložené správně a soubory, které jsou rozbité). Interně využívá skripty ***combine_pull.sh*** a ***split_add.sh*** k synchronizaci s cloudem.

4.2. Obecná architektura implementovaného řešení

Jak bylo uvedeno v kapitole 3.1 teoreticky architektura aplikace by měla vypadat jako na *obr. 3.1*. Ale v rámci diplomové práce byla implementována zjednodušená verze architektury, která je znázorněna na následujícím *obr. 4.1*.



Obr. 4.1: Architektura implementovaného řešení

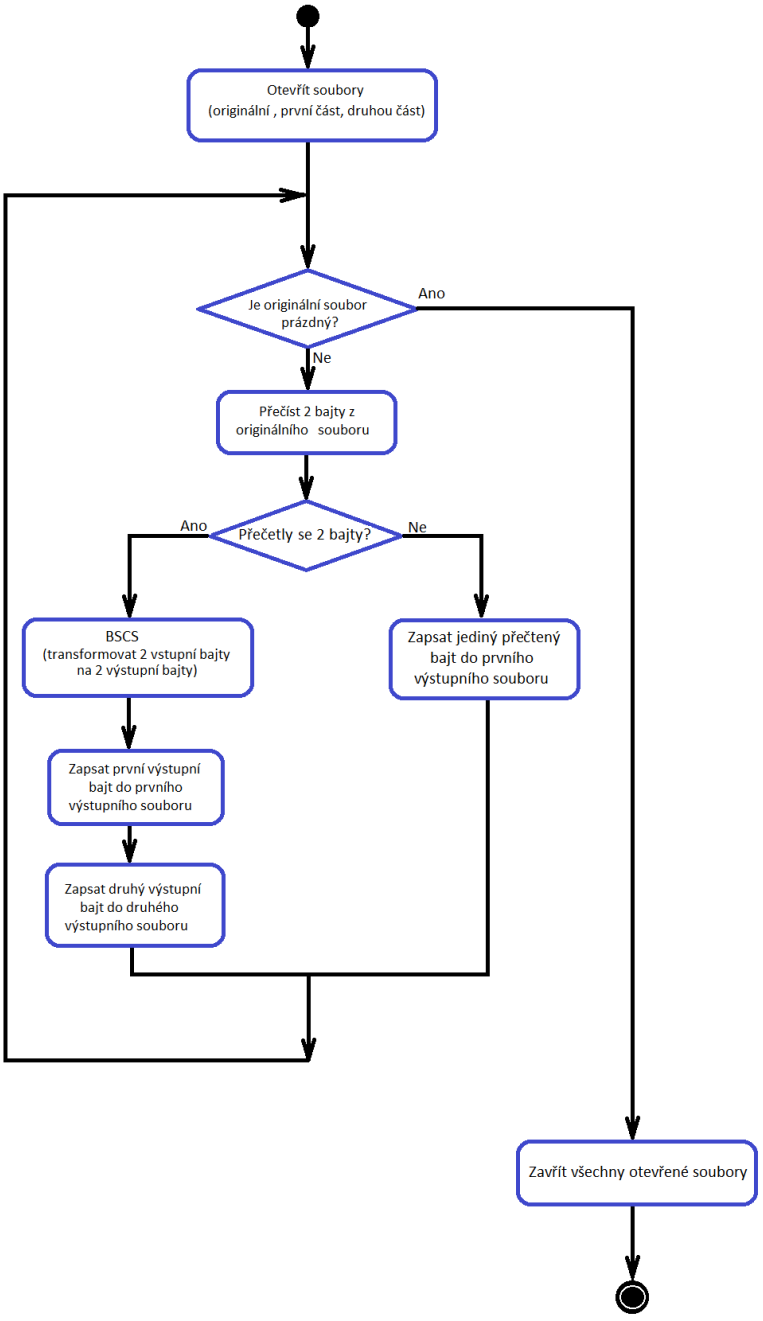
V rámci diplomové práce bylo implementováno rozdělení souboru na dvě části. Všechny první části se ukládají do jednoho cloudu, všechny druhé části se ukládají do druhého cloudu. Jelikož z jednoho souboru, který vidí uživatel interně, se stanou dva soubory, je potřeba zvolit přístup, jak pojmenovat jednotlivé části rozděleného souboru. Takže pro soubor s názvem – ***jmeno_souboru.rozliseni*** se první část bude jmenovat - ***jmeno_souboru.rozliseni.part1*** a druhá část bude mít název velmi podobný pouze s rozlišením číslem na jeho konci – ***jmeno_souboru.rozliseni.part2.*** Taková konvence nám pomůže spárovat první část souboru s jeho druhou částí.

5. Implementace techniky rozdělení dat

V rámci diplomové práce byla implementována aplikace *Doit* v programovacím jazyce C, která obsahuje implementaci techniky rozdělení dat pod názvem Bit Split Bit Combine (BSBC). Technika BSBC byla publikována ve článku [9]. Navržený algoritmus umožňuje bezpečně rozdělit soubor na dvě části a následně ho opět složit do původního stavu.

5.1. Rozdělení souboru na dvě části

Logika aplikace při rozděleni dat na dvě části je znázorněna na *obr. 5.1*.



Obr. 5.1: Logika rozdělení souboru na části

Funkce dělení souboru na části přijímá tři parametry. První parametr je název souboru (včetně cesty), který chceme rozdělit. Pro zjednodušení popisu řekněme, že chceme rozdělit soubor *example.txt*. Zbylé dva parametry jsou názvy souborů (včetně cesty), kam se zapíší výsledné části po dělení. Zase pro zjednodušení popisu vybereme dva názvy, například *example.txt.part1* a *example.txt.part2*.

Technika BSBC není závislá na typu zpracovávaných dat a pracuje s daty jako s proudem bajtů. Proto na začátku soubor *example.txt* otevřeme pro čtení v binárním modu. Aby se dalo pracovat i s velkými soubory, nesmíme načítat celý soubor *example.txt* do paměti, ale musíme ho číst postupně. Ale operace čteni souboru z disku jsou samy o sobě pomalé, proto při čteni souboru implementujeme buffer o rozměru 1 MB. V následujícím kroku otevřeme pro zápis v binárním modu soubory *example.txt.part1* a *example.txt.part2*. Ale operace zápisu souboru z disku jsou samy o sobě rovněž pomalé, proto implementujeme buffer o rozměru 1 MB pro zápis souboru.

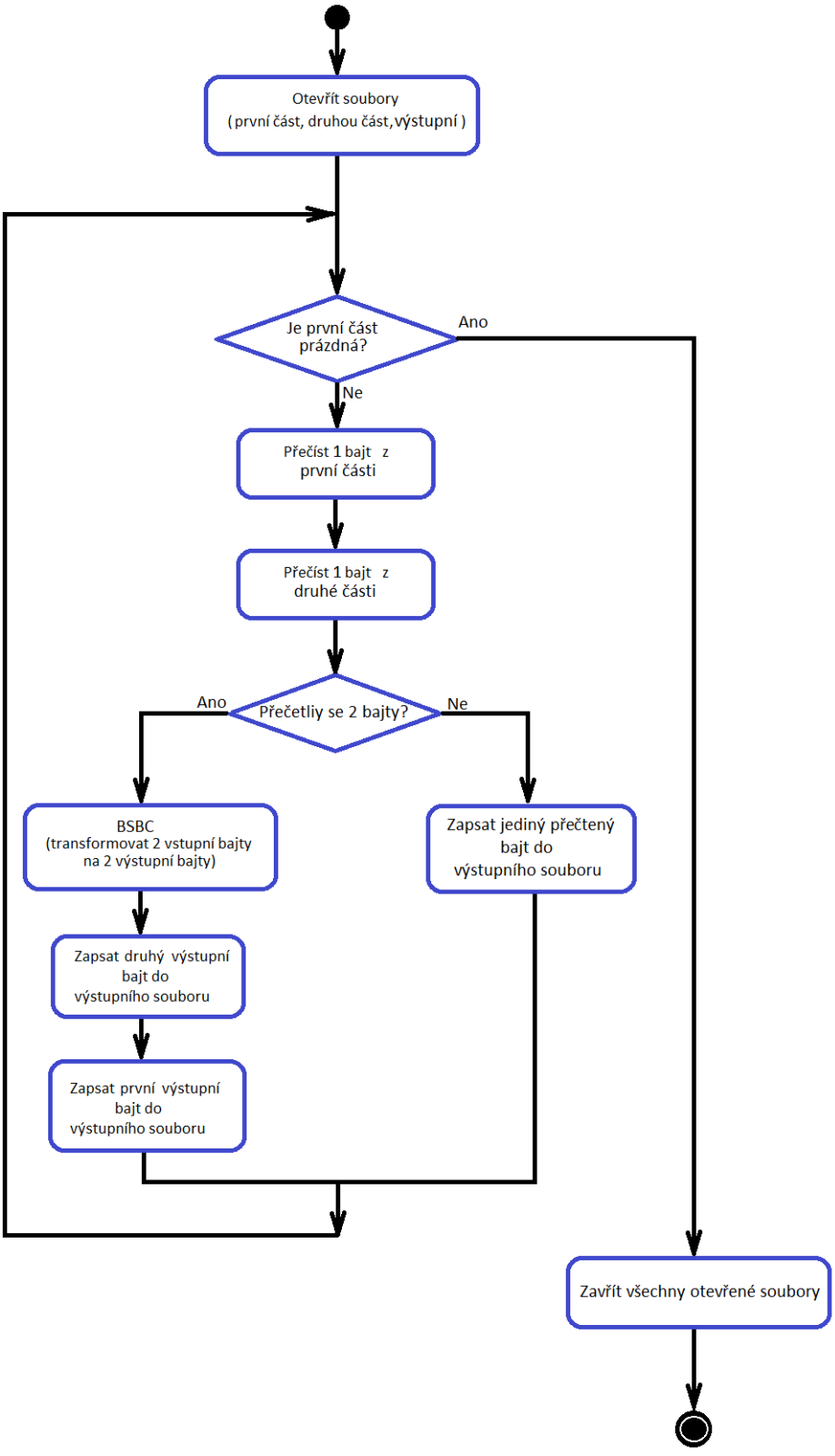
Po otevření všech potřebných souborů se můžeme posunout na hlavní smyčku programu. V téhle smyčce zpracováváme soubor *example.txt*, pokud není prázdný. Pokaždé se pokusíme načíst další dva bajty z už načteného bufferu vstupního souboru a při tom mohou nastat tři následující situace. Přečteme 0, 1 nebo 2 bajty. Jestliže se načetlo 0 bajtu, znamená to, že nastala chybová situace a kvůli zjednodušení schématu není zobrazena na diagramu, viz *obr. 5.1*. Jestliže se přečetly 2 bajty, tehdy pomocí BSBC techniky (samotná technika bude podrobně popsána dále) tyto dva bajty přetransformujeme na dva výstupní bajty. První výsledný bajt zapíšeme do *example.txt.part1* souboru a druhy výstupní bajt zapíšeme do *example.txt.part2* souboru (na začátku se bajty zapíší do bufferu, a když se buffer zaplní, tak ho zapíšeme na disk a začneme ho zaplňovat od začátku). Jestliže nastala třetí situace a byl přečten jenom 1 bajt, tehdy ho bez transformace zapíšeme rovnou do *example.txt.part1* souboru. Když byl vstupní soubor *example.txt* přečten celý, tak se ukončí hlavní smyčka programu. Pak musíme uzavřít všechny otevřené soubory a tím program končí.

5.2. Složení souboru ze dvou částí

Pro složení souboru ze dvou částí diagram na obr. 5.2 vypadá velice podobně, proto zdůrazníme jenom důležité rozdíly.

Funkce složení souboru z částí přijímá tři parametry. První parametr je název první částí, druhý parametr je název druhé části (všechno včetně cesty). Název výstupního souboru (včetně cesty) je třetím parametrem. Je důležité říct, že operace složení souboru je teď závislá na pořadí vstupních parametrů. To znamená, že jestli jako první parametr dáme název druhé části a jako druhý parametr dáme název první části, na výstupu dostaneme soubor, který neodpovídá realitě.

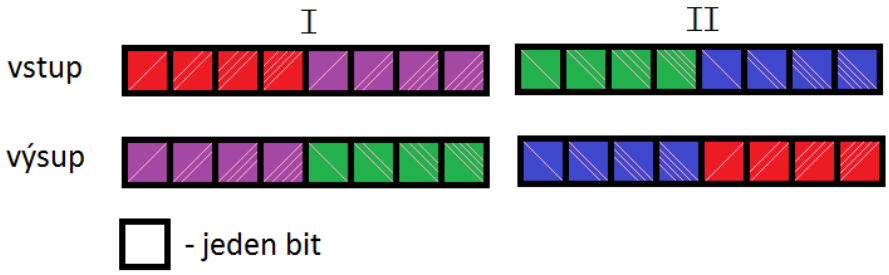
Operace BSBC, která přijímá dva bajty na vstup a přetransformovává je na dva bajty na výstupu, je stejná pro složení a rozdělení souborů. Ale při složení souboru operace BSBC vrátí bajty v opačném pořadí. Čemuž na schématu odpovídá otočené pořadí zápisů výstupních bajtů do výstupního souboru (na začátku zapíšeme druhý bajt a až pak první).



Obr. 5.2: Logika složení souboru z částí

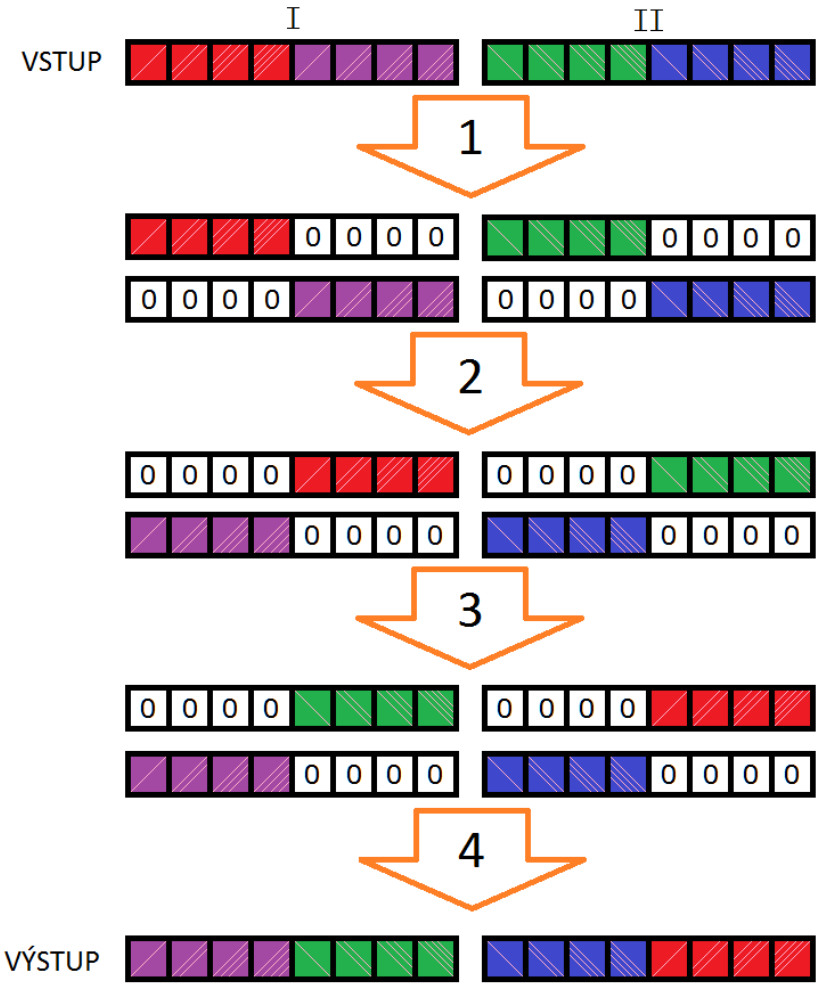
5.3. Znázornění techniky BSBC

Následující *obr. 5.3* ukazuje BSBC techniku z pohledu vstupu a výstupu. Na vstup přijdou dva bajty a na výstup dostaneme dva jiné bajty. Každý čtvereček odpovídá jednomu bitu. Ve skupině bitů stejné barvy nedojde k žádné změně pořadí.



Obr. 5.3: Technika BSBC zjednodušeně

Následující diagram na *obr. 5.4* po krocích ukazuje, jakým způsobem se vstupní bajty transformují na výstupní.



Obr. 5.4: Technika BSBC podrobně

Na vstup dostaneme dva bajty. Jako první akci rozdělíme každý bajt na horní a dolní polovinu, zbylou část doplníme nulami. V našem případě mají horní bity červenou barvu u prvního bajtu a zelenou u druhého. V druhém kroku posuneme horní bity (červené a zelené) doprava a dolní bity (fialové a modré) doleva. Jako třetí krok bity zamícháme, tak že červené a zelené bity přehodíme. Následně ve čtvrtém kroku je sečteme a dostaneme výsledné bajty.

5.4. Použití aplikace Doit

Konzolová aplikace *Doit* implementuje techniku BSBC rozdělení souboru na dvě části. Jako každá konzolová aplikace *Doit* je řízena vstupními argumenty (přepínači).

- Přepínač ***-h / --help*** vypíše na obrazovku seznam podporovaných přepínačů (viz *obr. 5.5*)

```
[test@localhost doit_src]$ ./doit -h
./doit [options]
        --help / -h            this information
        --split / -s [original_file_name] split original file into 2 parts
        --combine / -c [part_1_name] [part_2_name] [new_file_name] combine 2 files into one
```

Obr. 5.5: Použití aplikace Doit

- Přepínač ***-s/ --split [filename]*** (kde ***filename*** je jméno souboru s cestou) rozdělí soubor na části pomoci techniky BSBC
- Přepínač ***-c/ --combine [name_part1] [name_part2] [name_new_file]*** (kde ***name_part1*** je jméno souboru první části s cestou, kde ***name_part2*** je jméno souboru druhé části s cestou a ***name_new_file*** jméno výstupního souboru) složí soubor ze dvou částí pomoci techniky BSBC

5.5. Výkonnostní test

V rámci diplomové práce bylo provedeno výkonnostní porovnávání implementované techniky rozdělení dat se symetrickým šifrováním pomocí algoritmu AES s 256 bitovým klíčem. Jako vzorové implementace algoritmu šifrování byly vybrané knihovny OpenSSL a AEScrypt. Knihovna OpenSSL podporuje šifrování a dešifrování AES na hardwarové úrovni (instrukce AES-NI).

Testovací prostředí:

- CPU: Intel i7 7700k 4.5 GHz
- Memory: 16 GB DDR4 3200 MHz
- Hard drive: SSD Samsung 960 PRO
- OS: CentOS 7 64bit
- OpenSSL: 1.0.1i-fips
- AEScrypt: 3.11
- Doit: (implementovaná technika BSBC)

Sada testovacích dat byla tvořena soubory velikosti $1\,MB$, $5\,MB$, $20\,MB$, $50\,MB$, $100\,MB$, $250\,MB$, $500\,MB$, $1000\,\mathrm{MB}$, $1500\,MB$, $2000\,\mathrm{M}B$ a $2300\,\mathrm{M}B$. Výsledkem je průměr z 10 opakování stejného testu.

5.5.1. Použité příkazy

Při testování byly využity následující příkazy, kde *JMENO_VSTUPNIHO_SOUBORU* je jméno vstupního souboru, a *HESLO* je libovolné heslo, které zadává uživatel:

- Šifrování bez HW podpory pomocí AEScrypt:

```
$ aescrypt -e -p HESLO JMENO_VSTUPNIHO_SOUBORU
```

- Šifrovaní s HW podporou pomocí OpenSSL:

```
$ openssl enc -aes-256-cbc -salt -in JMENO_VSTUPNIHO_SOUBORU -out
JMENO_VSTUPNIHO_SOUBORU.enc
```

- Rozdělení souboru na 2 části pomocí implementované BSBC techniky:

```
$ /doit -s JMENO_VSTUPNIHO_SOUBORU
```

- Dešifrovaní bez HW podpory pomocí AEScrypt:

```
$ aescrypt -d -p HESLO JMENO_VSTUPNIHO_SOUBORU.aes
```

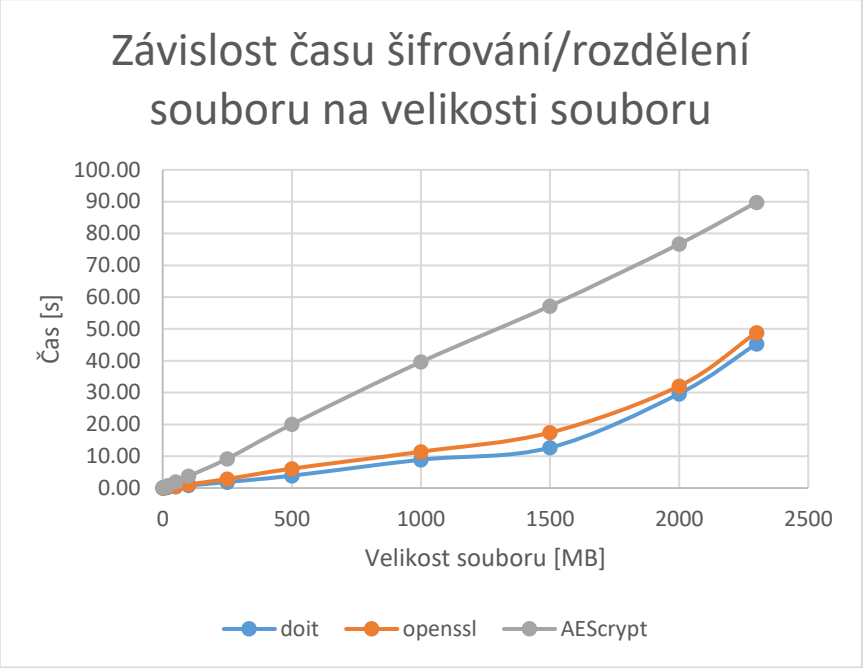
- Dešifrovaní s HW podporou pomocí OpenSSL:

```
$ openssl enc -d -aes-256-cbc -in JMENO_VSTUPNIHO_SOUBORU.enc -
out JMENO_VSTUPNIHO_SOUBORU.ssl
```

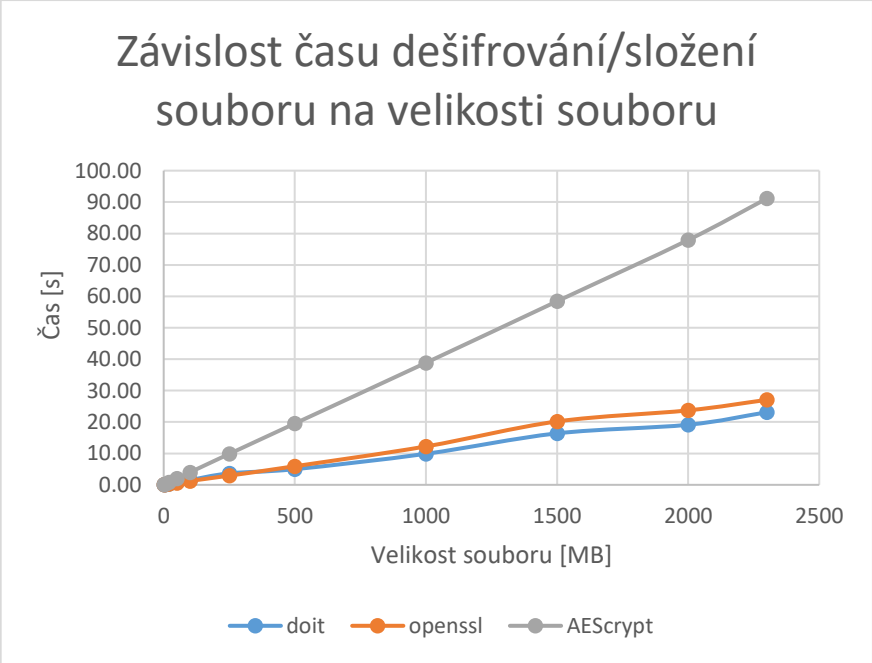
- Složení souboru ze 2 částí pomocí implementované BSBC techniky:

```
$ ./doit -c JMENO_VSTUPNIHO_SOUBORU.part1
JMENO_VSTUPNIHO_SOUBORU.part2 JMENO_VSTUPNIHO_SOUBORU.new
```

5.5.2. Porovnávaní šifrování/dešifrování souboru s rozdělením/složením souboru

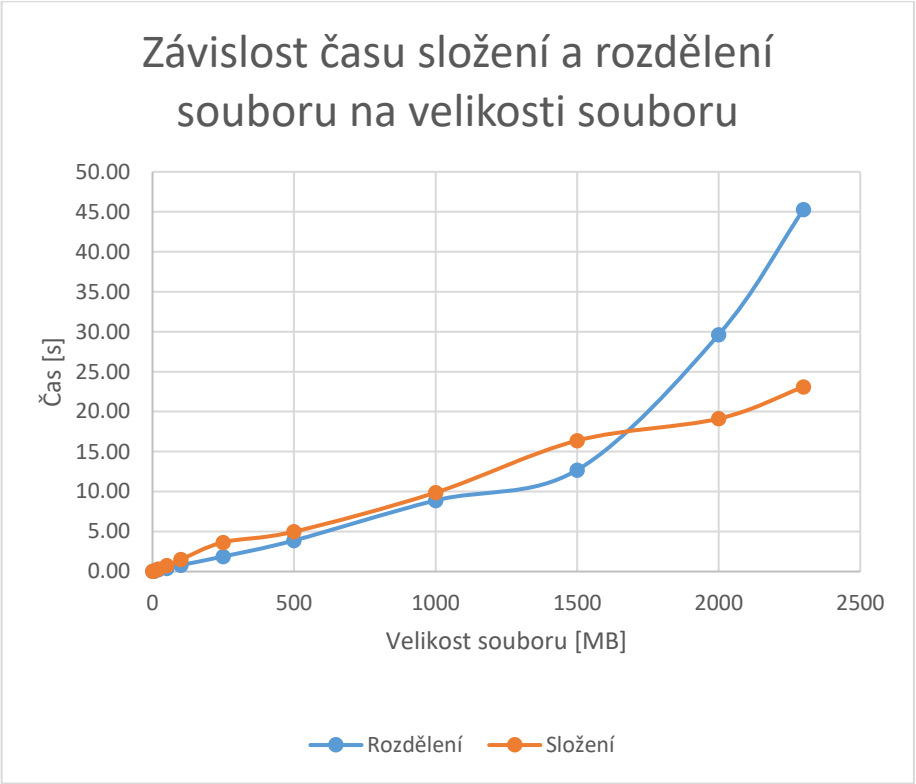


Obr. 5.6: Porovnávání šifrování souboru s rozdělením souboru na dvě části



Obr. 5.7: Porovnávání dešifrování souboru se složením souboru ze dvou částí

Grafy na *obr. 5.6* a *obr. 5.7* ukazují závislost času zpracování souboru na velikosti zpracovaného souboru při šifrování/rozdělení a dešifrování/složení dat. Z grafů je patrné, že aplikace *Doit* (implementující techniku BSBC rozdělení souboru na části) má nejlepší výsledky. Rozdíl mezi *Doit* a *AEScrypt* je o hodně větší než rozdíl mezi *Doit* a *OpenSSL*. *OpenSSL* šifruje pomocí stejného principu jako *AESCrypt*, ale vzhledem k tomu, že podporuje instrukce AES-NI, je výrazně rychlejší.



Obr. 5.8: Porovnání výkonnosti složení a rozdělení souboru

Z grafu srovnání rychlosti rozdělení a složení souborů (*obr. 5.8*) plyne, že složení velkých souborů probíhá rychleji než rozdělení.

6. Konzolová aplikace (CLI)

Hlavní rozdíl z hlediska uživatele při práci s cloudovým úložištím bez techniky rozdělení dat a s technikou rozdělení dat je, že při použití techniky rozdělení dat uživatel vlastně musí do cloudu ukládat místo jednoho souboru minimálně dva soubory (anebo i více, jestliže soubor byl rozdělen na více než na dvě částí), což udělá práci s cloudem minimálně dvakrát těžší (uživatel má provést minimálně dvakrát více akci). Proto byly vyvinuty dva skripty, které tuto nevýhodu odstraní tak, že uživatel pro operaci nahrávání/stažení souborů do/z cloudu bude muset udělat pouze jednu akci. Implementované skripty vlastně tvoří konzolové rozhraní našeho řešení.

V dané kapitole na začátku popíšeme, jakým způsobem se budeme připojovat přímo ke cloudu. Pak popíšeme, jak se používá konzolové rozhraní a na závěr uvedeme detaily implementace.

6.1. Problém připojování k různým cloudům

V dnešní době si uživatel může vybrat mezi různými poskytovateli cloudových úložišť. Například Google Drive, Dropbox, OneDrive, Yandex Disk a další. Každé úložiště nabízí uživateli různé služby. Z druhé strany každý cloud poskytovatel si přeje, aby jeho služby byly co nejvíce používané, a proto pro vývojáře různých aplikací poskytuje veřejné rozhraní (API) přes http protokol.

Například podíváme-li se blíž do Dropbox, najdeme na jeho stránkách [21] detailní dokumentaci pro propojení aplikace s Dropbox. Mimo jiné jsou pro vývojáře připravena SDK pro nejvíce využívané jazyky, aby developeři měli co nejméně problémů. SDK je v podstatě knihovna, která poskytuje abstrakci nad základním API (přes http protokol) jednoho konkrétního cloud poskytovatele.

Čistě teoreticky, kdybychom chtěli místo Dropbox využít Google Drive, tak bychom museli SDK (popřípadě http API) Dropboxu jednoduše vyměnit za SDK (popřípadě http API) Google Drive. Ale v praxi to není tak jednoduché, protože základní API přes http protokol je navrženo různými způsoby a dost se liší. Pro definici API pro Google Drive viz [22].

Tenhle problém (různým způsobem navřené API) se týká jakýchkoliv dalších cloud poskytovatelů. Proto vznikají různé projekty, které poskytují určitou abstrakci pro jednotnou práci s různými cloudy.

V diplomové práci budeme využívat aplikaci CloudCross (viz [25]). Ta už má v sobě implementované přímé připojení na různé cloudy a poskytuje konzolové rozhraní pro použití.

6.1.1. Aplikace CloudCross

CloudCross je open source software pro synchronizaci lokálních souborů a adresářů s různými cloudovými úložišti. Momentálně je podporovaná synchronizace s Yandex Disk, Google Drive, OneDrive a Dropbox. Pro naše účely jsou důležité následující vlastnosti:

- Synchronizace je možná podle bílého nebo černého listu. Takové chování bylo implementováno pomocí využití ***.exclude*** a ***.include*** souborů. Synchronizace podle bílého listu je prováděna jenom pro soubory uvedené v bílém listu (***.include*** soubor).

Synchronizace podle černého listu je prováděna pro všechny soubory, které nejsou uvedené v černém listu (***.exclude*** soubor).

- Možnost nastavení priority (lokálně či vzdáleně) při synchronizaci. Jestliže priorita je nastavena na „***local***“, znamená to, že lokální soubory mají přednost, a proto budou tyto soubory přehrány na cloudové úložiště. Takže lokální priorita se využívá, když chceme nějaké změny nahrát na cloudové úložiště. Jestliže potřebujeme něco stáhnout z cloudového úložiště na lokální počítač, je potřeba nastavit prioritu na „***remote***“, aby lokální soubory byly v případě vzniku rozdílů přehrané.
- Poskytování stejného jednoduchého konsolového rozhraní pro synchronizaci adresáře mezi různými cloudy.

Zkompilovaná verze aplikace je k dispozici na následujících operačních systémech CentOS, Debian, Fedora, OpenSUSE, Ubuntu (instrukce k instalace je na stránkách [24]). Protože aplikace je open source, tak existuje možnost ručně zkompilovat knihovnu ze zdrojových kódů, které naleznete na GitHub [25] a tam také najdete postup pro kompilaci.

6.1.2. Obecná autorizace aplikaci na cloudovém úložiště

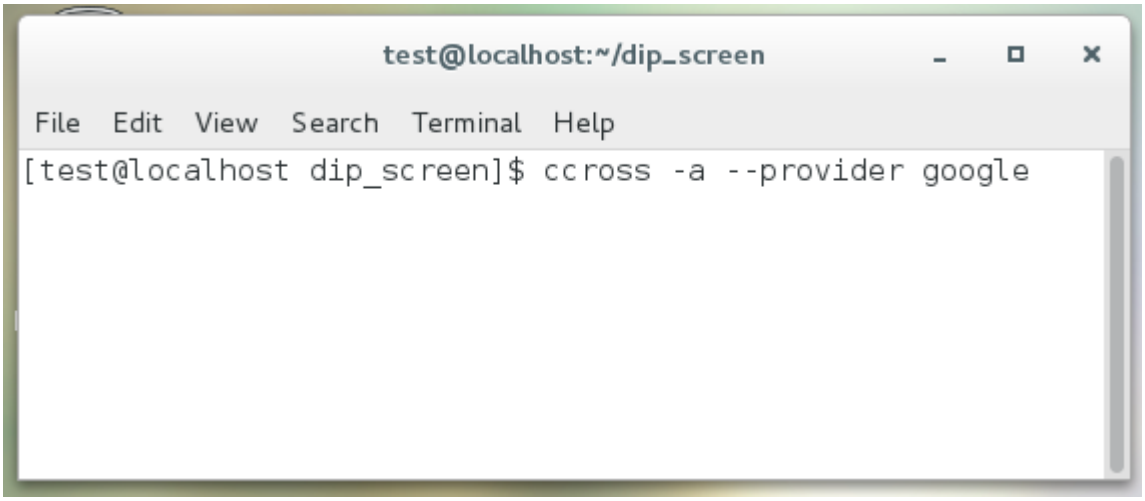
I když různá cloudová úložiště poskytují odlišné rozhraní pro použití jinými aplikacemi, musí používat stejně principy. Jedním ze společných přístupů je způsob autorizace aplikace na cloudovém úložišti. Bylo by uživatelsky nepřívětivé, kdyby všechny aplikace pro jakoukoliv operaci s cloudem požadovaly autorizaci (buďto login a heslo anebo otisk prstu, případně jakoukoliv jinou metodu). Rovněž by bylo nebezpečné, kdyby uživatel musel poskytovat autentifikační údaje různým aplikacím s cílem uložení a jejich následujícího automatického použití. Proto byl vymyšlen jiný mechanismus, který netrpí výše uvedenými vadami.

Mechanismus je založen na následující myšlence. Jelikož v každém případě pro každou operaci je potřeba autentifikace, tak na rozdíl od uživatele budou aplikace využívat autentifikační tokeny. Po prvním pokusu o přístup do cloudu aplikace požádá uživatele, aby ji autentizoval. Aplikace na začátku vygeneruje odkaz na stránky cloud poskytovatele. Na těchto stránkách uživatel projde autentizací a povolí aplikaci přístup ke svému učtu v cloudu. Je důležité si uvědomit, že v tento okamžik aplikace nemá přístup k autentizačním údajům uživatele na cloudovém úložišti. Jakmile uživatel provede úspěšné odsouhlasení možnosti přístupu do cloudu, aplikace dostane token od cloud poskytovatele, který bude i nadále využívat pro autorizovaný přístup ke cloudovému úložišti. Bezpečnost mechanismu přístupu přes tokeny není součástí dané diplomové práce, a proto budeme předpokládat, že je bezpečný.

6.1.3. Autorizace aplikaci CloudCross na cloudovém úložiště

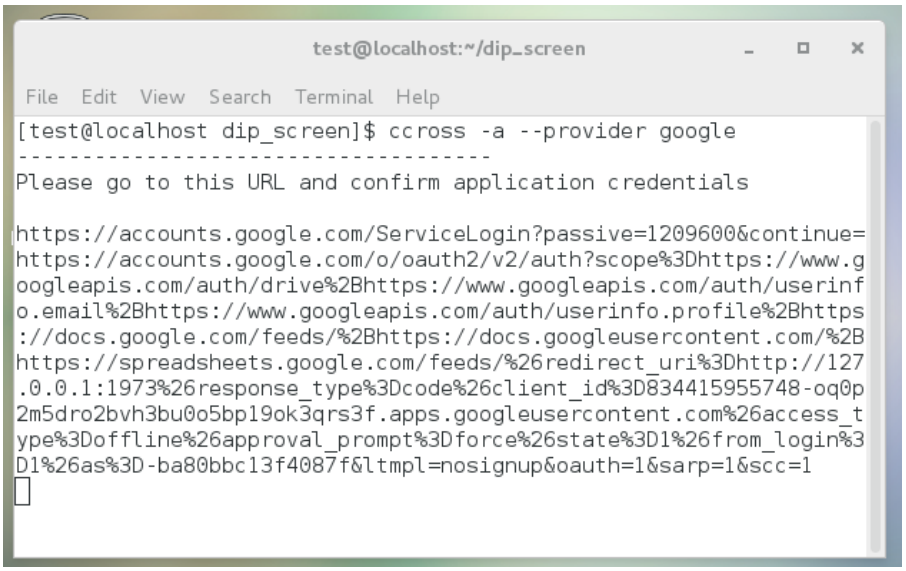
V našem případě budeme přistupovat přímo k datům v cloudu přes aplikaci *CloudCross*. Proto níže popíšeme, jaké tam má rozhraní autorizační mechanismus. Aplikace *CloudCross* se spouští na příkazové řádce pomocí příkazu *ccross* s různými argumenty.

Jak bylo uvedeno výše, aplikace *CloudCross* poskytuje stejné rozhraní pro přístup k různým cloudům. Ale nemá informaci, ke kterému z výše uvedených cloudů bychom chtěli přistupovat, proto je potřeba při každé operaci explicitně uvádět typ cloud poskytovatele. K tomu slouží přepínač ***--provider***. Hodnota přepínače může nabývat následujících hodnot „*yandex*“, „*onedrive*“, „*google*“, „*dropbox*“ a „*mailru*“. Na následujícím *obr. 6.1* je příklad použití.



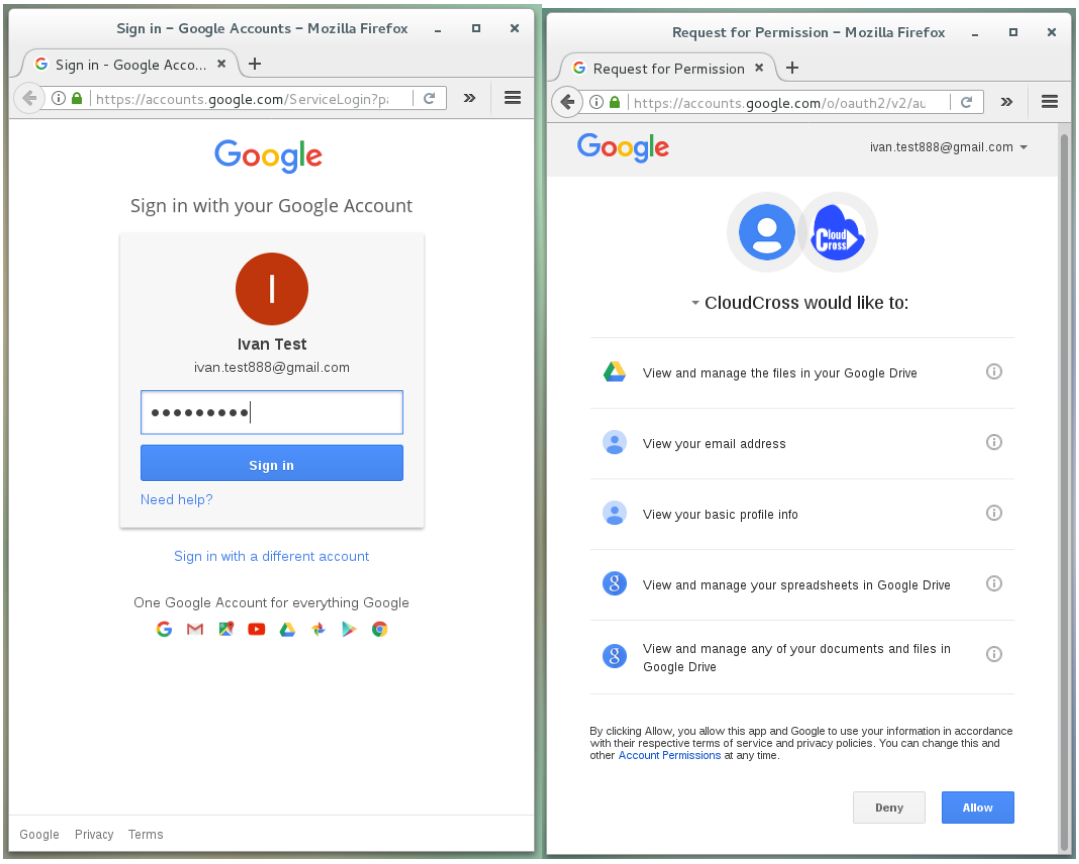
Obr. 6.1: Registrace cloudu pomocí příkazu ccross

Zahájit autentizaci aplikace *CloudCross* je možné pomocí přepínače ***-a***. Spuštěním příkazu uvedeného na *obr. 6.2* zahájíme registraci pro synchronizace adresáře ***dip_screen*** s cloudem Google Drive.

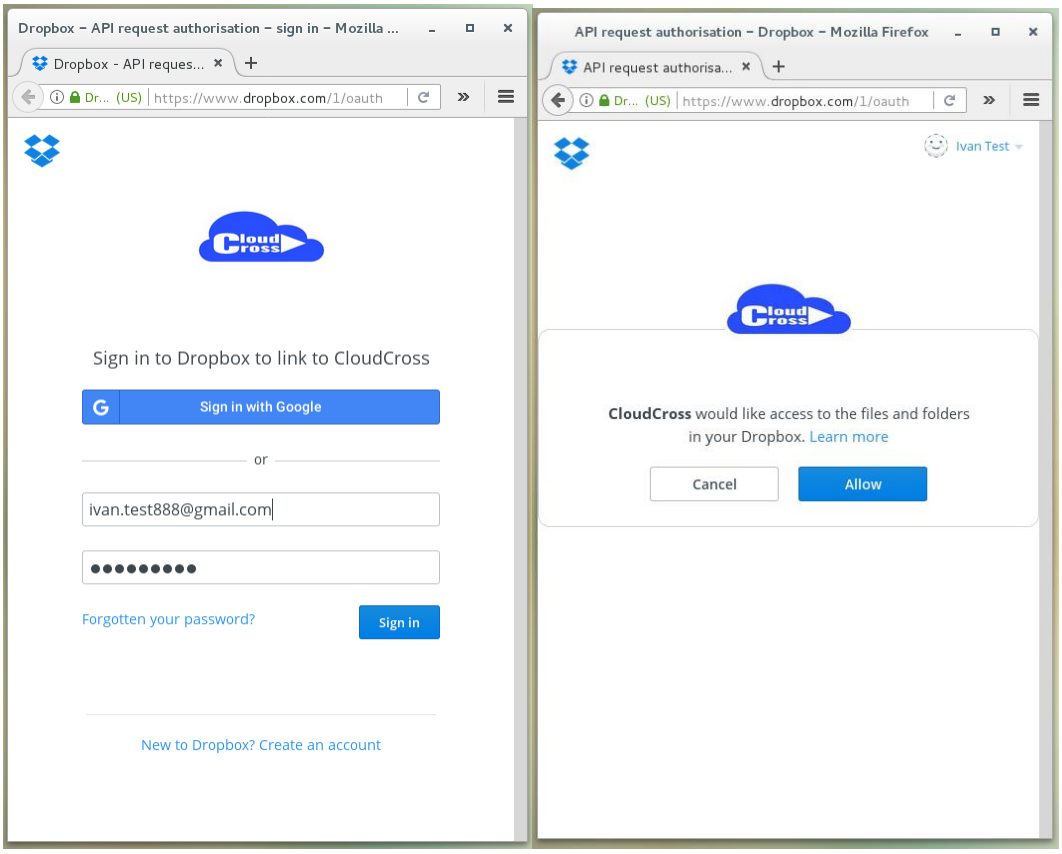


Obr. 6.2: Aplikace CloudCross vygeneruje URL pro registraci v cloudu

Po spuštění příkazu se na příkazové řádce vypíše URL, který uživatel musí zkopírovat do jakéhokoliv prohlížeče, přihlásit se na svůj účet v cloudu a pak autorizovat aplikaci *CloudCross* k přístupu na cloudové úložiště. Příklad autentizace na Google Drive je zobrazen na *obr. 6.3* a na Dropbox na *obr. 6.4.*



Obr. 6.3: Autorizace aplikace CloudCross na Google Drive (vlevo) Přihlášení do Google účtu (vpravo), přidělení práv aplikaci CloudCross na manipulaci s Google Drive



Obr. 6.4: Autorizace aplikace CloudCross na Dropbox (vlevo) Přihlášení do Dropbox účtu (vpravo), přidělení práv aplikaci CloudCross na manipulaci s Dropbox

Pro jiné cloudy by bylo vygenerované jiné URL a uživatel by musel postupovat podle procesů nastavených v daném cloudovém úložišti.

Při implementaci CLI nebyla automatizovaná autorizace na cloudových úložištích, proto uživatel musí spustit aplikaci *CloudCross* ručně na příkazové řádce. Při implementaci GUI bylo automatizováno spuštění *CloudCross* aplikace na pozadí externího procesu. Proto uživatel nemusí zadávat žádné příkazy na příkazové řádce a bude schopen vybrat typ cloud poskytovatele přímo v GUI a přímo v GUI dostane odkaz pro autentifikaci.

6.1.4. Popis použitých přepínačů v aplikaci CloudCross

Aplikace *CloudCross* má velké množství různých přepínačů. Zde ale uvedeme jen jejich zkrácený seznam (jenom těch, které byly přímo využity při implementaci grafického rozhraní). Celý seznam se dá najít buď na stránkách [24] nebo ve zdrojovém kódu [25] nebo vypsat na obrazovku seznam možných přepínačů příkazem `ccross --help`.

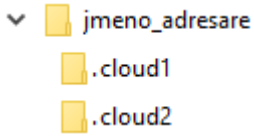
- ***-s*** [***--list***] vypíše seznam všech souborů, které jsou uložené v příslušném cloudu
- ***-h*** [***--help***] vypíše na obrazovku seznam podporovaných přepínačů
- ***-v*** [***--version***] vypíše verzi aplikací *CloudCross*
- ***-a*** [***--auth***] slouží k autorizaci aplikací *CloudCross* na účtu uživatele u cloud poskytovatele
- ***--prefer ARGUMENT*** nastaví směr synchronizací. ***ARGUMENT*** může nabývat dvou hodnot „*local*“ (lokální soubory mají přednost, ve většině případů se používá při nahrávání souborů do cloudu) a „*remote*“ (soubory v cloudovém úložišti mají přednost, ve většině případů se používá pro stahování souborů z cloudového úložiště na lokální počítač)
- ***--use-include*** zapíná využití bílého listu. To znamená, že při synchronizaci libovolným směrem budou synchronizované jen soubory, které jsou explicitně uvedené v bílém listu. Bílý list je soubor ***.include***, kde jsou vyjmenovány soubory vždy jeden na každém řádku. Jestliže tento přepínač není explicitně uveden, znamená to, že obsah souboru ***.include*** (když takový bude existovat) nebude na synchronizaci mít žádný vliv a zároveň to znamená, že synchronizace probíhá s použitím černého listu (soubor ***.exclude***), a to rovněž znamená, že budou synchronizované všechny soubory, které nejsou explicitně vypsané v černém listu
- ***--provider ARGUMENT*** uvádí, se kterým cloud poskytovatelem bychom chtěli synchronizovat. ***ARGUMENT*** může nabývat následujících hodnot: „*yandex*“, „*onedrive*“, „*google*“, „*dropbox*“ nebo „*mailru*“

6.2. Použití CLI

Implementované konzolové rozhraní je postaveno nad použití aplikace *CloudCross*. Hlavním cílem bylo zjednodušení stahování/nahrávání souborů z/do cloudu. Použití CLI vyžaduje jednorázové manuální přednastavení adresáře, který si uživatel přeje synchronizovat s cloudem. Pod pojmem přednastavení adresáře rozumíme vytvoření správné souborové struktury v adresáři a registraci cloudů. Pro zjednodušení následujícího vysvětlení dodejme, že uživatel chce synchronizovat adresář se jménem „jmeno_adresare“.

6.2.1. Vytvoření souborové struktury

Vytvoření souborové struktury má uživatel udělat jenom jednou, a to před první synchronizaci. Tehdy musí uživatel ručně vytvořit strukturu adresářů tak, jak je uvedeno na následujícím *obr. 6.5*.



Obr. 6.5: Souborová struktura

Jak bylo uvedeno v kapitole 4.2, všechny první části se ukládají do jednoho cloudu a všechny druhé části se ukládají do druhého cloudu. Takže první části budou synchronizované s cloudem, který bude zaregistrován v adresáři „jmeno_adresare/.cloud1“, a druhé části budou synchronizované s cloudem, který bude zaregistrován v adresáři „jmeno_adresare/.cloud2“. Adresář „jmeno_adresare/.cloud1“ budeme nazývat vlastní adresář prvního cloudu a adresář „jmeno_adresare/.cloud2“ budeme nazývat vlastní adresář druhého cloudu. Takové řešení je dobře škálovatelné, protože při použití algoritmu rozdělení souborů na více než dvě části, bude pouze potřeba přidat další skryté podadresáře. Čísla v názvu nám pomohou při vyhledávání částí.

6.2.2. Registrace cloudů

Dále je potřeba zaregistrovat cloudová úložiště. Tohle uživatel má udělat jenom jednou, a to před první synchronizaci a až po vytvoření souborové struktury. Registraci prvního cloudu provedeme tak, že přímo z vlastního adresáře prvního cloudu zadáme příkaz `ccross -a --provider NAZEV_PROVIDERU` a registraci druhého cloudu provedeme tak, že přímo z vlastního adresáře prvního cloudu zadáme také příkaz `ccross -a --provider NAZEV_PROVIDERU`. Detailně je proces manuální registrace cloudu pomocí aplikace *CloudCross* popsán v kapitole 6.1.3.

6.2.3. Nahrávání souborů do cloudu

Operaci nahrávání souborů do cloudu uživatel může provádět jen po vytvoření souborové struktury a registrace cloudů. Pro nahrávání souborů do cloudu byl vyvinut skript split_add.sh. Tento skript podporuje nahrávání ve dvou variantách. V první variantě se do cloudu nahraji všechny lokální soubory a v druhé variantě se do cloudu nahrají jenom vybrané soubory. Podle vstupních parametrů skriptu je schopen sám detekovat, jaká varianta byla použita.

Skript má tři povinné parametry. První parametr je jméno (včetně cesty) adresáře, který si uživatel přeje synchronizovat s cloudem. Tomuto adresáři budeme říkat pracovní adresář. Druhý parametr je typ prvního cloud poskytovatele, kam podle kapitoly 4.2 se ukládají první části. A třetí parametr je typ druhého cloud poskytovatele, kam podle kapitoly 4.2 se ukládají druhé části. Všechny následující parametry jsou považované za jména souborů v pracovním adresáři, které si uživatel přeje nahrát do cloudu.

```
[test@localhost binaries]$ ./split_add.sh /home/test/CLOUD dropbox google
argc = 3
Split mode=ALL
entry=2010.doc.part1
entry=DSCN7441.JPG.part1
entry=Manual-SAMSUNG.pdf.part1
entry=Motherboard.JPG.part1
entry=Picture.bmp.part1
entry=Processor.JPG.part1
entry=split_add.sh.part1
entry=Xbox_Elite.JPG.part1
entry=2010.doc.part2
entry=DSCN7441.JPG.part2
entry=Manual-SAMSUNG.pdf.part2
entry=Motherboard.JPG.part2
entry=Picture.bmp.part2
entry=Processor.JPG.part2
entry=split_add.sh.part2
entry=Xbox_Elite.JPG.part2
Reading remote files
Reading local files and folders
Checking folder structure on remote
Start syncronization
/split_add.sh.part1 Changed local. Uploading.
/2010.doc.part1 Changed local. Uploading.
/DSCN7441.JPG.part1 Changed local. Uploading.
/Motherboard.JPG.part1 Changed local. Uploading.
/Picture.bmp.part1 Changed local. Uploading.
/Manual-SAMSUNG.pdf.part1 Changed local. Uploading.
/Processor.JPG.part1 Changed local. Uploading.
/Xbox_Elite.JPG.part1 Changed local. Uploading.
Syncronization end
Reading remote files
Reading local files and folders
Checking folder structure on remote
Start syncronization
Syncronization end
```

Obr. 6.6: Vzorový výstup skriptu split_add.sh pro nahrání všech souborů

Například uživatel chce nahrát do cloudu všechny soubory z adresáře `/home/test/CLOUD`, první cloud je Dropbox a druhý cloud je Google Drive, tehdy má zadat na konzoli následující příkaz

```
./split_add.sh /home/test/CLOUD dropbox google
```

Na následujícím *obr. 6.6* je znázorněno, jak by mohl vypadat výstup takového příkazu.

Kdyby uživatel chtěl nahrát do cloudu z adresáře */home/test/CLOUD* jenom jeden soubor *2010.doc*, tehdy by měl uživatel zadat na konzoli následující příkaz

```
./split_add.sh /home/test/CLOUD dropbox google 2010.doc
```

Na následujícím *obr. 6.7* je znázorněno, jak by mohl vypadat výstup takového příkazu.

```
[test@localhost binaries]$ ./split_add.sh /home/test/CLOUD dropbox google 2010.doc
argc = 4
Split mode=SELECTED
Element=2010.doc
entry=2010.doc.part1
entry=2010.doc.part2
Reading remote files
Reading local files and folders
Checking folder structure on remote
Start syncronization
/2010.doc.part1 Changed local. Uploading.
Syncronization end
Reading remote files
Reading local files and folders
Checking folder structure on remote
Start syncronization
Syncronization end
```

Obr. 6.7: Vzorový výstup skriptu split_add.sh pro nahrání jen vybraných souborů

6.2.4. Stažení souborů z cloudu

Operaci stažení souborů z cloudu uživatel může provádět jen po vytvoření souborové struktury a registraci cloudů. Pro stažení souborů z cloudu byl vyvinut skript combine_pull.sh. Tento skript podporuje stahování ve dvou variantách. V první variantě se na lokální počítač z cloudu stáhnou všechny soubory a v druhé variantě se z cloudu stáhnou jenom vybrané soubory. Podle vstupních parametrů je skript schopen sám detekovat, jaká varianta byla použita.

Skript má tři povinné parametry. První parametr je jméno (včetně cesty) adresáře, který si uživatel přeje synchronizovat s cloudem. Tomuto adresáři říkáme pracovní adresář (stejně jak to bylo v kapitole 6.2.3). Druhý parametr je typ prvního cloud poskytovatele, kam se ukládají první části. A třetí parametr je typ druhého cloud poskytovatele, kam se ukládají druhé části. Všechny následující parametry jsou považované za jména souborů v pracovním adresáři, které si uživatel přeje stáhnout z cloudu.

Například uživatel chce stáhnout z cloudu všechny soubory do adresáře */home/test/CLOUD*, první cloud je Dropbox a druhý cloud je Google Drive, tehdy má zadat na konzoli následující příkaz

```
./combine_pull.sh /home/test/CLOUD dropbox google
```

Na následujícím *obr. 6.8* je znázorněno, jak by mohl vypadat výstup takového příkazu.

```
[test@localhost binaries]$ ./combine_pull.sh /home/test/CLOUD dropbox google
argc = 3
WORK_DIR=/home/test/CLOUD
path_to_doit=/home/test/Documents/diplom/diplom_test/binaries/doit
Combine mode=ALL
Download of first parts started
Reading remote files
Reading local files and folders
Checking folder structure on local
Start syncronization
/Motherboard.JPG.part1 New remote. Downloading.
/Picture.bmp.part1 New remote. Downloading.
/2010.doc.part1 New remote. Downloading.
/Xbox_Elite.JPG.part1 New remote. Downloading.
/Processor.JPG.part1 New remote. Downloading.
/Manual-SAMSUNG.pdf.part1 New remote. Downloading.
/split_add.sh.part1 New remote. Downloading.
/DSCN7441.JPG.part1 New remote. Downloading.
Syncronization end
Download of first parts ended
list_cloud1
2010.doc DSCN7441.JPG Manual-SAMSUNG.pdf Motherboard.JPG Picture.bmp Processor.JPG split_add.sh Xbox_Elite.JPG
Download of second parts started
Reading remote files
Reading local files and folders
Checking folder structure on local
Start syncronization
Syncronization end
Download of second parts ended
list_cloud2
2010.doc DSCN7441.JPG Manual-SAMSUNG.pdf Motherboard.JPG Picture.bmp Processor.JPG split_add.sh Xbox_Elite.JPG
Unique names
Motherboard.JPG DSCN7441.JPG Xbox_Elite.JPG Manual-SAMSUNG.pdf Picture.bmp split_add.sh 2010.doc Processor.JPG
Combining: Motherboard.JPG
Combining: DSCN7441.JPG
Combining: Xbox_Elite.JPG
Combining: Manual-SAMSUNG.pdf
Combining: Picture.bmp
Combining: split_add.sh
Combining: 2010.doc
Combining: Processor.JPG
```

Obr. 6.8: Vzorový výstup skriptu combine_pull.sh pro stažení všech souborů

Kdyby uživatel chtěl stáhnout z cloudu do adresáře */home/test/CLOUD* jenom jeden soubor *2010.doc*, tehdy by měl uživatel zadat na konzoli následující příkaz

```
./combine_pull.sh /home/test/CLOUD dropbox google 2010.doc
```

Na následujícím *obr. 6.9* je znázorněno, jak by mohl vypadat vystup takového příkazu.

```
[test@localhost binaries]$ ./combine_pull.sh /home/test/CLOUD dropbox google 2010.doc
argc = 4
WORK_DIR=/home/test/CLOUD
path_to_doit=/home/test/Documents/diplom/diplom_test/binaries/doit
Combine mode=SELECTED
Download of first parts started
Reading remote files
Reading local files and folders
Checking folder structure on local
Start syncronization
Syncronization end
Download of first parts ended
list_cloud1
2010.doc
Download of second parts started
Reading remote files
Reading local files and folders
Checking folder structure on local
Start syncronization
Syncronization end
Download of second parts ended
list_cloud2
2010.doc
Unique names
2010.doc
Combining: 2010.doc
```

Obr. 6.9: Vzorový výstup skriptu combine_pull.sh pro stažení jen vybraných souborů

6.3. Implementace CLI

Pro zjednodušení operací stahování/nahrávání souborů z/do cloudu byly vyvinuté v jazyce Bash dva skripty ***combine_pull.sh*** a ***split_add.sh***. Oba skripty tvoří nadstavbu nad aplikacemi *CloudCross* a *Doit*.

6.3.1. Implementace nahrávání souborů do cloudu

Skript ***split_add.sh*** je řízen vstupními parametry. Jestliže počet parametrů je menší, než počet povinných argumentů (máme vlastně tří povinné argumenty), skript skončí s chybou. Jestliže parametry byly předané správně, skript pokračuje dále. Protože budeme aktivně používat synchronizaci s bílým listem, je potřeba starý bílý list smazat a vytvořit nový, ale na začátku je potřeba detekovat, jaká varianta nahrávání byla spuštěna (všechny nebo jen vybrané soubory). Detekujeme to podle počtu předaných parametrů. Při synchronizaci všech souborů přečteme jmena všech souborů v lokálním adresáři a vytvoříme bílé listy (jeden pro první cloud další pro druhý cloud). Při synchronizaci jenom vybraných souborů vytvoříme bílé listy jednoduše z předaných parametrů. Pak každý soubor v bílém listu rozdělíme na části tak, že pustíme aplikaci *Doit* se správnými parametry (viz kapitola 5.5), tím se nám vytvoří soubory prvních a druhých částí. Přesuneme první části souborů do vlastního adresáře prvního cloudu a druhé části souborů do vlastního adresáře druhého cloudu. Pak pomocí aplikace *CloudCross* synchronizujeme vlastní adresáře prvního a druhého cloudu (nahrajeme soubory do cloudů).

6.3.2. Implementace stažení souborů z cloudu

Skript ***combine_pull.sh*** je řízen vstupními parametry stejným způsobem jako skript ***split_add.sh***. Jestliže počet parametrů je menší, než počet povinných argumentů (máme vlastně tří povinné argumenty), skript skončí s chybou. Jestliže byly parametry předány správně, skript pokračuje dále. Protože budeme aktivně používat synchronizaci s bílým listem, je potřeba starý bílý list smazat a vytvořit nový, ale na začátku je potřeba detekovat, jaká varianta nahrávání byla spuštěna (všechny nebo jen vybrané soubory). Detekujeme to podle počtu předaných parametrů. Při synchronizaci všech souborů, přečteme jmena všech souborů v lokálním adresáři a vytvoříme bílé listy (jeden pro první cloud, a další pro druhý cloud). Při synchronizaci jenom vybraných souborů vytvoříme bílé listy jednoduše z předaných parametrů. Pak pomoci aplikace *CloudCross* synchronizujeme vlastní adresáře prvního a druhého cloudu (stáhneme soubory z cloudu). Nyní máme stažené soubory prvních a druhých částí, které byly v cloudu. Může se stát, že nějaká část chybí, anebo pro nějaký požadovaný soubor se v žádném cloudu nenašly žádné dílčí části. Před tím, než budeme moci zkombinovat první část souboru s druhou části souboru, je potřeba vygenerovat seznam těch souborů, které jdou složit. To uděláme tak, že najdeme soubory, pro které byly stažené obě části – první i druhá. Následně projedeme celý tento seznam a pomocí aplikace *Doit* složíme soubor.

7. Grafické rozhraní (GUI)

V předchozí kapitole byly popsány implementované skripty pro zjednodušení použití techniky rozdělení dat pro ochranu soukromí v cloudu. Ale používání příkazové řádky pro více uživatelů je velmi komplikovaná cesta.

Tato kapitola je věnovaná popisu grafického rozhraní aplikace, které zjednoduší práci uživatele při synchronizaci adresáře s cloudy takovým způsobem, že uživatel jen zaregistruje cloudy, a aplikace sama obstará operace rozdělení souborů na několik částí, složení souborů z několika částí, synchronizace správných částí do správných cloudů, pro první část nalezení odpovídající druhé části apod.

7.1. Použití aplikace

Grafické uživatelské rozhraní poskytuje vizualizaci stavu adresáře, který si uživatel přeje bezpečně uložit do cloudu.

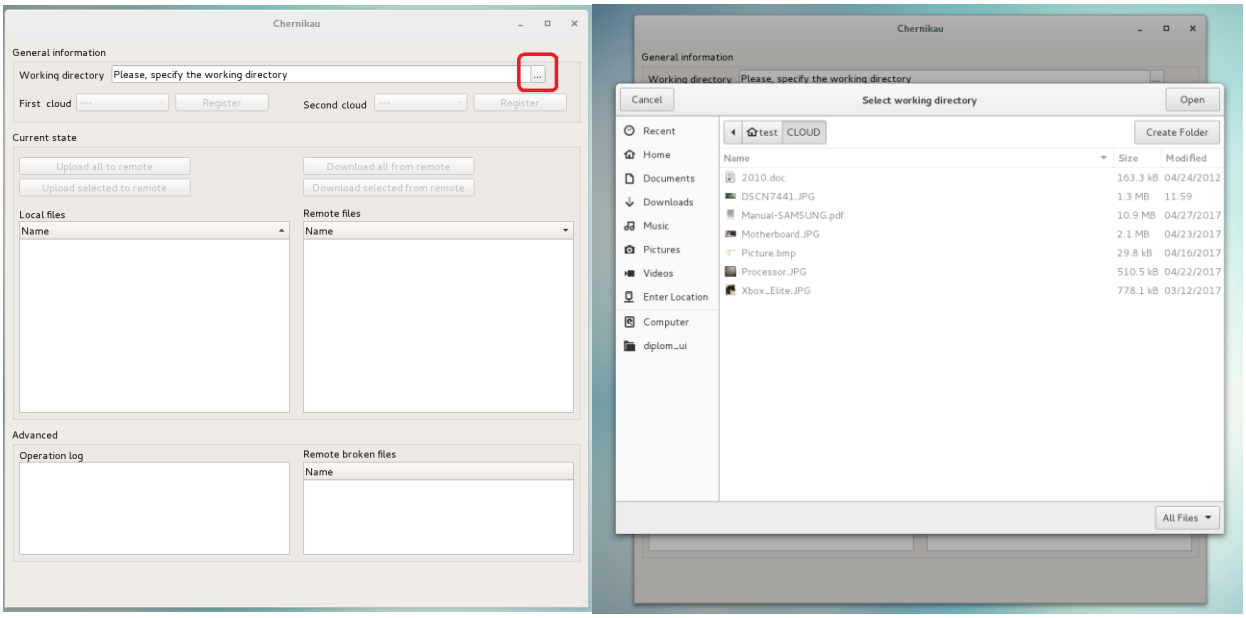
Aplikace je vizuálně rozdělena na tři oblasti:

- Základní nastavení (výběr pracovního adresáře a registrace cloudu)
- Vizualizace obsahu (lokálního adresáře a cloudu) a možné akce
- Dodatečná funkcionalita (log všech operací a seznam rozbitých souborů v cloudu)

Pod použitím aplikace rozumíme její nastavení a provádění běžných akcí stahování/nahrávaní souborů z/do cloudu.

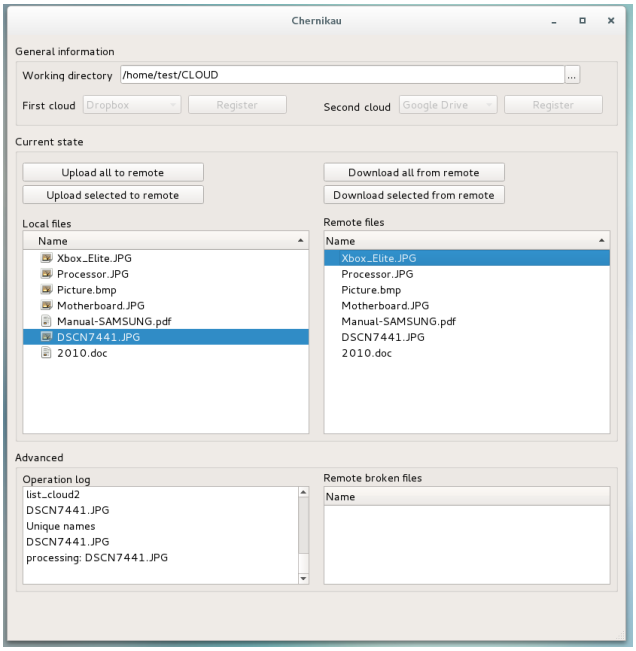
7.1.1. Základní nastavení

Po otevření aplikace musí uživatel před zahájením práce vždy na začátku vybrat adresář pro synchronizaci (viz *obr. 7.1*).



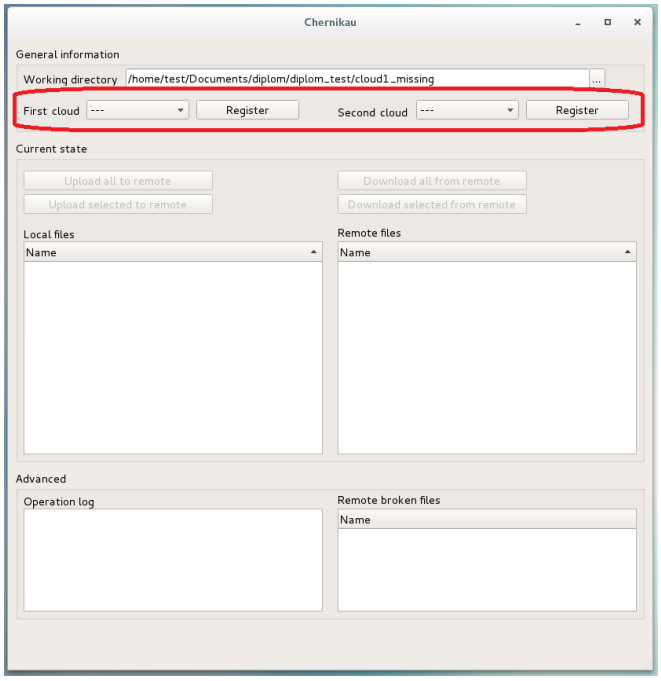
Obr. 7.1: Výběr pracovního adresáře v GUI

Po výběru pracovního adresáře v dialogovém okně automaticky proběhne detekce zaregistrovaných cloudů, kam by se měly ukládat jednotlivé části. Jestliže oba cloudy byly zaregistrované, tak se uživateli zobrazí typy zaregistrovaných cloud poskytovatelů a povolí se další akce (uložení/nahrávání souborů do/z cloudu), načtou se soubory v lokálním adresáři a soubory v cloudu (viz o*br. 7.2*).



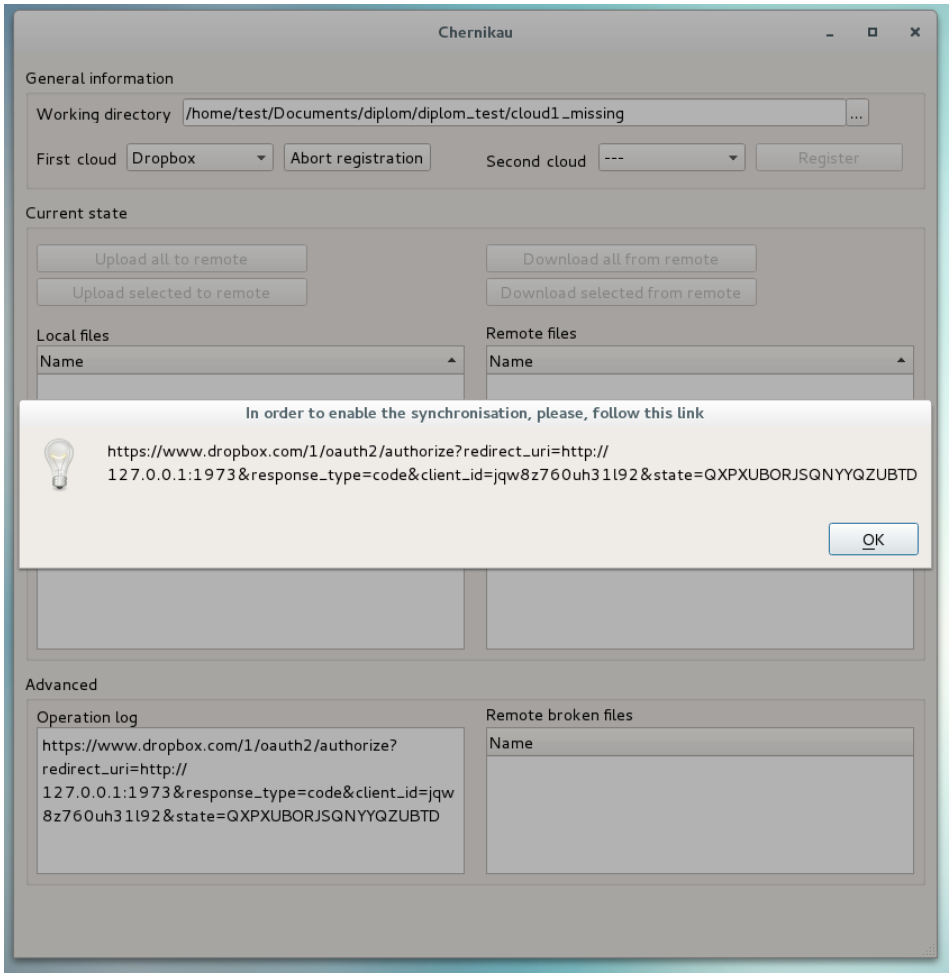
Obr. 7.2: Stav GUI po úspěšném výběru pracovního adresáře

Jestliže ani jeden cloud nebyl detekován, tak aplikace informuje uživatele, že nějaký (nebo oba) z cloudů nebyl zaregistrován. V tom okamžiku může uživatel buď nastavit správně vybraný adresář anebo vybrat nový adresář pro synchronizaci. Aby uživatel mohl zaregistrovat cloud, v GUI se povolí možnost registrace. Uživatel to může učinit zmáčknutím tlačítka „*Register*“, viz *obr. 7.3*, ale předtím musí vybrat ze seznamu cloud poskytovatele, kterého chce zaregistrovat.



Obr. 7.3: Možnost registrace cloudu v GUI

Po zmáčknutí tlačítka „*Register*“ se spustí proces registrace cloudu a uživateli se v informačním okénku zobrazí odkaz na stránky cloud poskytovatele. Jestliže uživatel udělal chybu při výběru cloud poskytovatele, tak může zavřít informační okénko stisknutím tlačítka „*OK*“ a zrušit registraci stisknutím tlačítka „*Abort registration*“ (viz *obr. 7.4*).



Obr. 7.4: GUI generuje URL pro registrace cloudu

Jak je popsáno v kapitole 6.1.3, uživatel musí tento odkaz zkopírovat do prohlížeče. Přes odkaz se dostane na stránky cloud poskytovatele, kde se uživatel přihlásí do svého účtu. Po přihlášení do účtu se uživateli zobrazí stránka (každý cloud poskytovatel ji má implementovanou jinak), kde uživatel povolí programu *CloudCross* (protože ho používáme pro přímou komunikaci s cloudem) přístup ke cloudovému úložišti. V okamžiku, kdy uživatel toto učiní, cloud bude v jeho aplikaci zaregistrován. Je důležité dodat, že registrace cloudu se provádí jenom jednou pro každý cloud. Po zaregistrování obou cloudů se uživateli zpřístupní další akce.

7.1.2. Synchronizace dat s cloudovými úložišti

Navržená a implementovaná aplikace dále umožňuje synchronizovat lokální data s daty uloženými v registrovaných cloudových úložištích. Za tímto účelem jsou uživateli zpřístupněny následující funkcionality:

- Stáhnout všechny soubory z cloudu na lokální počítač
- Stáhnout jen označené soubory z cloudu na lokální počítač
- Nahrát všechny soubory z lokálního počítače do cloudu

- Nahrát jen označené soubory z lokálního počítače do cloudu

Pro stažení všech souborů z cloudu uživateli stačí pouze zmáčknout tlačítko „*Download all*“. Operace stahování se uživateli jeví jenom jako jedna atomická operace stahování, ale ve skutečnosti probíhají 3 operace:

1) Stáhnutí první části souboru z cloudu
2) Stáhnutí druhé části souboru z cloudu
3) Složení celého souboru z obou částí

Pro nahrávání všech souborů do cloudu uživateli stačí jenom zmáčknout tlačítko „*Upload all*“. I tady se operace nahrávání uživateli jeví jako jedna atomická operace nahrávání, přestože opět probíhají 3 operace:

1) Rozdělení souboru na dvě části
2) Nahrání první části souboru do cloudu
3) Nahrání druhé části souboru do cloudu

Operace nahrávání a stahování jsou implementované ve dvou variantách. První varianta manipuluje se všemi soubory (operace jsou navázané na tlačítka „*Download all*“ a „*Upload all*“), druhá varianta manipuluje jenom se soubory, které uživatel ručně vyznačil v GUI (operace jsou navázané na tlačítka „*Download selected*“ a „*Upload selected*“).

7.2. Implementace grafického rozhraní

Funkční a přívětivé grafické uživatelské rozhraní dělá aplikaci přístupnou pro větší množinu potenciálních uživatelů. Následující část je věnována detailnímu popisu implementace grafického rozhraní pro synchronizaci adresáře s cloudy, kde je technika rozdělení dat používána k zabezpečení dat na straně klienta. Na začátku je v kapitole krátce popsána knihovna Qt v5.8 použitá k vývoji grafického rozhraní, pak je detailně popsána implementace.

7.2.1. Popis knihovny Qt

Pro napsání GUI jako nástroje byla vybrána knihovna Qt. Tato knihovna je často využívána nejen pro vývoj desktopových grafických a konzolových aplikací, ale i pro vývoj embedded aplikací. Mezi výhody použití knihovny patří:

- jednoduchost používání
- velká sada už implementovaných pomocných tříd
- dobrá veřejně přístupná dokumentace
- Qt Creator (nativní IDE pro vývoj aplikací pomocí Qt knihovny)
- multiplatformovost (možnost tvorby aplikací pro Windows, Linux a Mac OS současně)

Multiplatformovost

Knihovna Qt zaručuje, že všechny funkce (třídy, objekty apod.) jsou přenositelné mezi platformami. To znamená, že při využití aplikací Qt funkcí může zadavatel bez problémů přeložit kód i pro Linux nebo Windows. Jestliže používá funkce mimo knihovnu Qt, tak by zřejmě musel kód nějak přizpůsobit.

Signály a sloty

Signály a sloty jsou nejužitečnějším mechanismem, kterým Qt rozšiřuje možnosti jazyka C++. Vývojáři se často setkávají s problémem, že je potřeba asynchronně provést akci, když nastane nějaká událost. Signály a sloty výborně rozdělují funkcionalitu (akce od událostí) a poskytují možnost navázání určitých akcí na dané události. Toto zvládají velice jednoduše za běhu programu, čímž dosáhnou lepší čitelnosti a rozšiřovatelnosti kódu.

Každý objekt v knihovně Qt může emitovat signály (jednak jsou přednastavené signály, ale dá se i naimplementovat vlastní). V okamžiku, kdy objekt emituje signál, se nezajímá o to, kdo na tento signál bude reagovat, ani o to, jestli ho bude někdo zpracovávat. Jediné, co je důležité, je skutečnost, že objekt oznámil systému, že se něco stalo. Signál vlastně oznamuje, že objekt změnil svůj stav takovým způsobem, že by to mohlo být zajímavé pro jiné objekty.

Z druhé strany každý objekt knihovny Qt může mít nějaké sloty. Slot je vlastně takový speciální druh metody ve třídě, který slouží k tomu, aby mohl být navázán na nějaký signál. Objekt, který má sloty, se vlastně nezabývá tím, jak, kdy a s čím bude propojen. Je důležité zdůraznit, že provázání signálů a slotů je dynamické (provádí se až za běhu programu), proto se propojováním může zabývat i jiný objekt, což vede ke zlepšení kvality kódu.

Z těchto důvodů máme možnost vytvářet zcela nezávislé komponenty a rozdělit funkcionalitu do tří kategorií:

- signál – informace, že v systému nastala nějaká důležitá událost
- slot – nějaká akce, která se má uskutečnit jako odpověď na nějakou událost v systému
- logika – jaké akce (sloty) a jak jsou propojené s událostmi (signály)

Je důležité upozornit na skutečnost, že signály fungují bezpečně i mezi vlákny, protože knihovna Qt předává data přes frontu událostí a není tudíž potřeba používat jiné prostředky pro mezivláknovou synchronizaci.

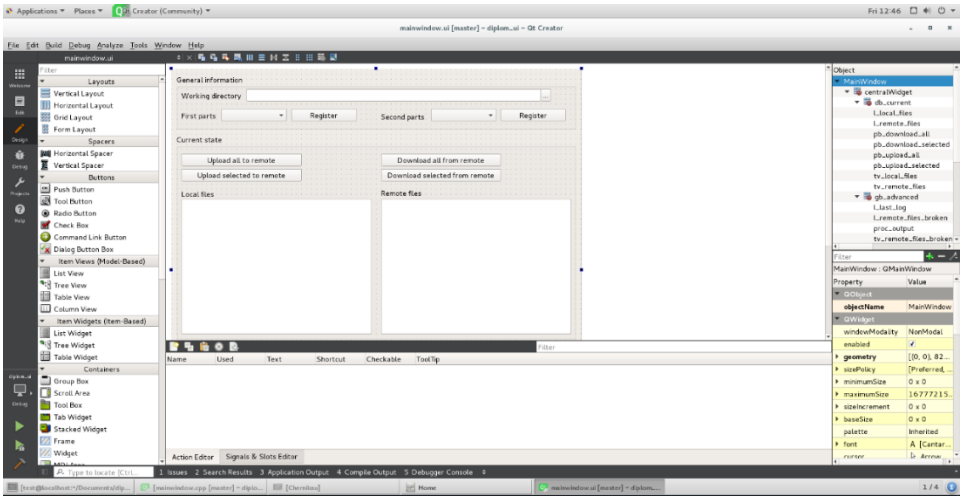
Řízení událostí

Mechanizmus „Signály a sloty“ přidává událostní (asynchronní) charakter celé knihovně Qt.

Modularita

Knihovna Qt je modulární. Sama o sobě je knihovna rozsáhlá, ale není potřeba používat celou její kapacitu. Uživatel může použít pouze nějakou podmnožinu modulů. Níže je uveden seznam základních modulů:

- QtCore – základní modul, který obsahuje třídu QObject a podporu pro vlákna, externí procesy, soubory, kontejnery, textové streamy, časovače, MVC a další
- QtGui – grafické elementy
- QtMultimedia – podpora pro video a audio
- QtNetwork – podpora různých síťových protokolů
- QtTest – modul pro testování a další



Obr. 7.5: Qt Creator ukázka

Dokumentace

Knihovna Qt je velice podrobně zdokumentovaná. Dokumentace je přístupná na stránkách [23] a obsahuje nejen popis jednotlivých tříd, metod, maker, signálů, slotů atd., ale i množství různých příkladů.

Qt Creator

Qt Creator je IDE, které pomáhá s vývojem aplikací. Má v sobě wizardy pro vytváření aplikací. Zároveň má integrovanou dokumentaci a vestavěný Qt Designer (nastroj pro zjednodušení vývoje GUI aplikace, který poskytuje možnosti drag-and-drop vizuálních elementů, viz *obr. 7.5*).

7.2.2. Správa externích procesů pomoci knihovny Qt

Naše GUI aplikace je nadstavbou nad CLI s dodatečně zabudovanou možností registrace cloudů, proto se skripty ***split_add.sh*** (nahrávání souborů) a ***combine_pull.sh*** (stažení souborů) zavolají jako externí procesy se správně nastavenými parametry.

Knihovna Qt poskytuje třídu *QProcess*, která je používána pro práci s externími procesy a komunikací s nimi. Detailní dokumentace k této třídě se dá najít na stránkách [26]. Na přikladu procesu nahrávaní ukážeme, jak se používá třída *QProcess*.

Vytvoření objektu třídy *QProcess* vypadá následovně:

```
process_upload = new QProcess(this);
```

Pak je potřeba nastavit proces, který budeme spouštět (v našem případě je to ***split_add.sh***):

```
process_upload->setProgram(helper_scripts + „split_add.sh“);
```

Zde je proměnná ***helper_scripts***, která obsahuje absolutní cestu k souboru „*split_add.sh*“.

Dále je potřeba nastavit argumenty, se kterými budeme spouštět externí proces. Protože argumenty pro každé spuštění se mohou lišit, budeme je nastavovat pokaždé znovu.

Po nastavení argumentů nám už nic nebrání spustit externí proces pomocí příkazu:

```
process_upload->start();
```

Proces nahrávání se spustí asynchronně, to znamená, že poběží někde na pozadí a hlavní okénko s GUI prvky uživateli nezamrzne.

Po ukončení procesu potřebujeme nějakým způsobem zareagovat na výsledek. K tomu využijeme mechanismus *Signály a sloty*. Ukončení procesu je oznámeno systému tím, že proces, který byl ukončen, emituje signál *finished*. Na tento signál můžeme navázat slot následujícím způsobem:

```
connect(process_upload, SIGNAL(finished(int , QProcess::ExitStatus )), this,
SLOT(process_upload_finished(int , QProcess::ExitStatus )));
```

Tímto způsobem budou použity všechny externí procesy, které budeme pouštět.

7.2.3. Detailní popis prováděných akcí

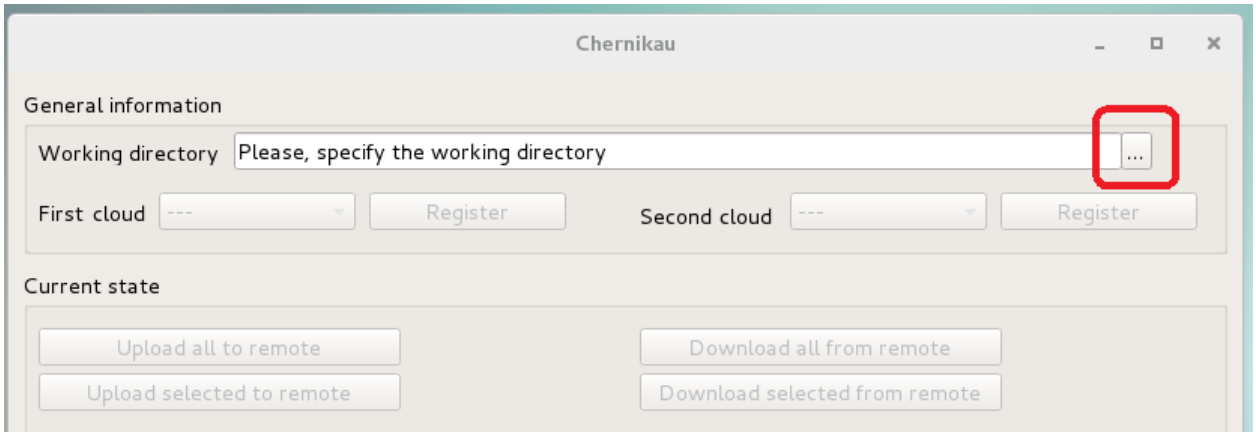
Grafické rozhraní poskytuje možnost provedení následujících akcí buď automaticky nebo na přání uživatele:

- Výběr pracovního adresáře
- Detekce typu cloudu
- Registrace cloudu
- Zobrazení seznamu souboru v lokálním adresáři
- Zobrazení seznamu souborů v cloudu
- Stažení všech (nebo vybraných) souborů z cloudu
- Nahrání všech (nebo vybraných) souborů do cloudu

Výběr pracovního adresáře

Pod pojmem pracovní adresář rozumíme adresář, jehož obsah chce uživatel bezpečnostně uložit do cloudu s použitím metody rozdělení dat k zabezpečení dat na straně klienta. Výběr pracovního adresáře je jediná povolená akce při spuštění aplikace.

Po výběru pracovního adresáře se buď načtou informace o lokálních a vzdálených souborech (jestliže byl adresář správně nastaven), nebo je uživateli poskytnuta možnost registrace cloudů.



Obr. 7.6: GUI výběr pracovního adresáře

Po kliknutí na tlačítko (viz *obr. 7.6*) by se měl uživateli zobrazit dialog výběru adresáře. Proto potřebujeme na signál *clicked* u tlačítka navázat slot vlastního zpracování *setup_working_directory*.

Výhled dialogového okénka pro výběr adresáře závisí na operačním systému, kde běží aplikace. Nastavíme dialogové okénko tak, aby uživatel mohl vybrat jenom adresář, ale nemohl vybrat soubor (tím se vyhneme zpracování zbytečné chyby, kdy uživatel vybere soubor místo adresáře).

Jestliže uživatel adresář nevybral (místo jména adresáře bude prázdný řetězec) tak by neměl provádět žádné změny, aby nedošlo ke zbytečným chybám.

Jestliže uživatel adresář vybral, potřebuje zakázat všechny akce, což provede tak, že zablokuje (zakáže) všechna tlačítka. Taková pojistka je nutná, protože proces zpracování požadavku na výběr pracovního adresáře probíhá asynchronně a může trvat i déle a uživatel by mohl v průběhu procesu rekonfigurace spustit něco jiného v okamžiku, kdy konfigurace není ukončena.

Pak potřebujeme zjistit, zda pro adresář vybraný uživatelem už byly zaregistrované cloudy a tyto zobrazit v GUI cloud poskytovatelů (jestliže byly detekované). Následně povolíme registraci cloudů (povolíme tlačítka „*Register*“) tam, kde je to potřeba.

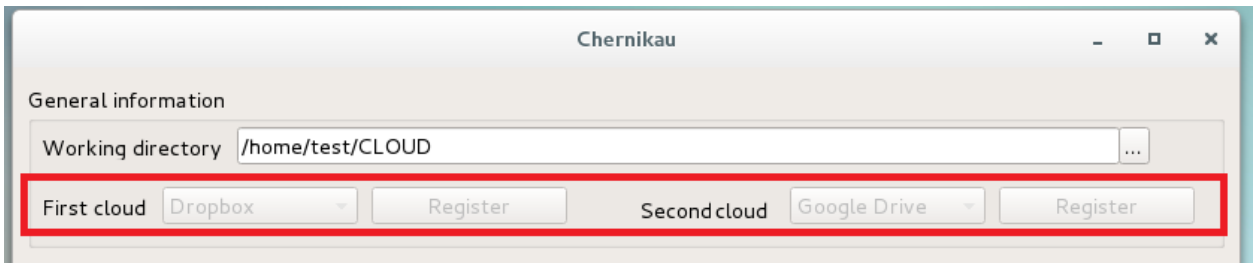
Jestliže byly oba cloudy zaregistrované, zobrazí se uživateli seznam lokálních souborů, seznam souborů správně uložených v cloudu a seznam rozbitých souborů v cloudu.

Detekce typu cloudu

Pro synchronizaci s cloudem používáme vlastní skripty přes externí program *CloudCross*. Jeho rozhraní je navrženo tak, že při každé synchronizaci adresáře je potřeba explicitně uvádět typ poskytovatele cloudu. Na druhou stranu není uživatelsky přívětivé pokaždé nutit uživatele, aby ručně vybíral typ cloud poskytovatele (navíc by to mohlo vést i ke zbytečným uživatelským chybám), a proto je možné typ cloud poskytovatele detekovat automaticky. Víme, že program *CloudCross* pro své účely ukládá tuto informaci tak, že v adresáři, který synchronizuje, vytvoří dodatečný skrytý adresář (například „.dbox“ pro Dropbox a „.grive“ pro Google Drive). Na základě přítomnosti jednoho či druhého adresáře detekuje, který typ cloudu byl zaregistrován.

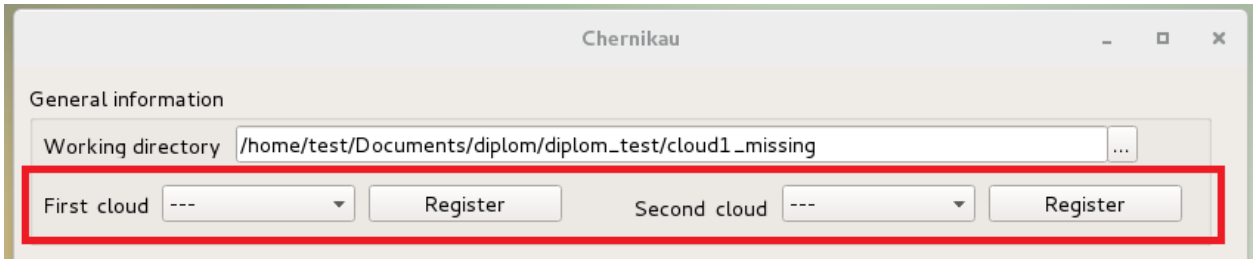
Interní souborová struktura (popsána v kapitole 6.2.1) není pro uživatele viditelná, ale záleží na ní, jestli uživatel může provádět další synchronizaci, nebo musí ještě vyčkat. Proto je potřeba ji kontrolovat. Jestliže vlastní adresář některého z cloudů chybí, znamená to, že registrace tohoto cloudu neproběhla a nejsme tudíž schopni detekovat typ cloud poskytovatele, protože nebyl ještě nastaven. V případě, že vlastní adresář cloudu existuje, tak na základě znalosti toho, že adresář cloudu obsahuje nějaký specifický podadresář (například „.dbox“ nebo „.grive“) detekujeme typ cloud poskytovatele.

Po úspěšném detekování typ cloud poskytovatele zobrazíme tuto skutečnost v GUI tak, že v seznamu podporovaných cloud poskytovatelů vybereme správnou hodnotu a zakážeme uživateli ji měnit (viz *obr. 7.7*).



Obr. 7.7: GUI automaticky úspěšně detekovalo zaregistrované cloudy

Situaci, kdy typ cloud poskytovatele nebyl automaticky detekován a registrace cloudů ještě neproběhla, vyřešíme tak, že v GUI v seznamu podporovaných cloud poskytovatelů vybereme hodnotu „---“ a následně bude uživateli povoleno tuto hodnotu měnit (viz *obr. 7.8*). Pak je možno cloudy zaregistrovat.

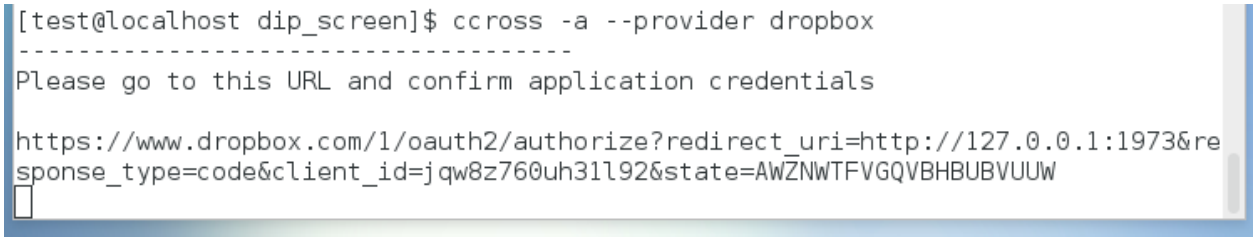


Obr. 7.8: GUI automaticky neúspěšně detekovalo zaregistrované cloudy

Registrace cloudového úložiště

Proces registrace cloudu bude uživateli povolen tehdy, jestliže program nebyl schopen automaticky detekovat typ alespoň jednoho cloud poskytovatele (to vlastně znamená, že cloud ještě nebyl zaregistrován). Proces registrace cloudu můžeme rozdělit na několik na sebe navazujících částí.

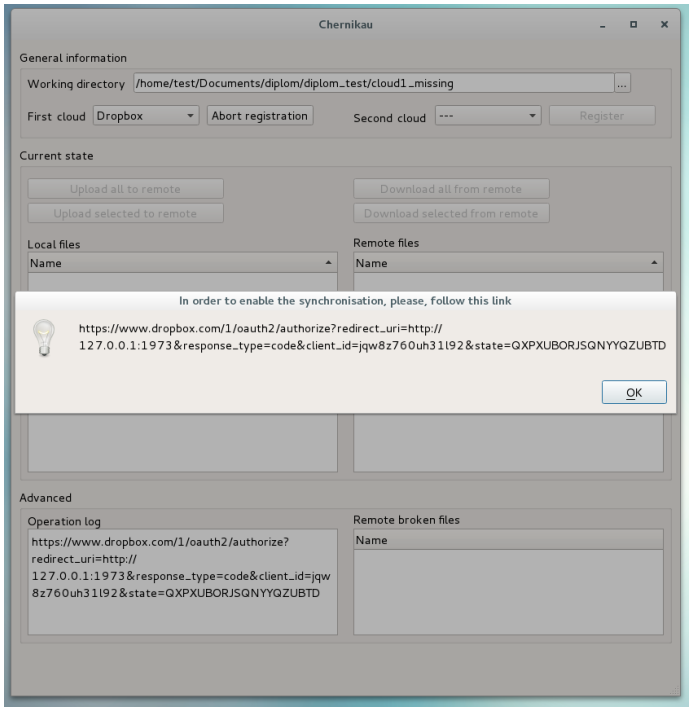
Zaprvé potřebujeme nastavit parametry, se kterými budeme chtít pustit externí program *CloudCross*. Přepínač ***-a*** říká, že chceme požádat o autentifikaci, přepínačem ***--provider NAZEV_CLOUD_POSKYTOVATELU*** předáme informaci, se kterým cloud poskytovatelem budeme chtít synchronizovat. Registrace cloud poskytovatele může probíhat pouze v případě, že cloud poskytovatel ještě nebyl zaregistrován. Proto musí uživatel při registraci ručně vybrat typ cloud poskytovatele ze seznamu. Dalším krokem je nutnost nastavit pracovní adresář pro program *CloudCross* na vlastním adresáři cloudu (jestliže neexistuje, vytvoříme ho).



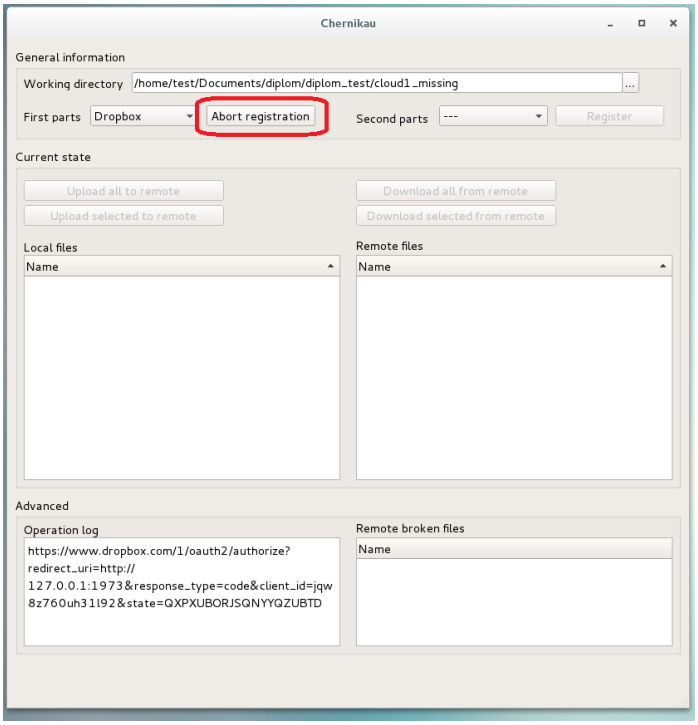
Obr. 7.9: Aplikace CloudCross vypisuje URL a nějaký dodatečný text na obrazovku

Jakmile jsme předchozí kroky provedli, asynchronně pustíme program *CloudCross* jako externí proces. Program *CloudCross* vygeneruje odkaz na stránky cloud poskytovatele, vypíše ho na standartní výstup a bude čekat, dokud uživatel neprojde odkazem a povolí (popřípadě nepovolí) autentifikaci. Protože nejde jednoduše poznat, jestli odkaz byl už vypsán celý, nebo je tam jenom jeho část, potřebujeme znát způsob, jak to detekovat. Počkáme, dokud se alespoň něco nevypíše

na standartním výstupu a pak zkontrolujeme, jestli už byl odkaz vypsán celý. To poznáme podle toho, že poslední znak na výstupu je konec řádky (který symbolizuje konec odkazu), a jestli ve výstupním textu je podřetězec „http“ (který symbolizuje začátek odkazu). Jestliže nebyl vypsán celý, opět počkáme a provedeme kontrolu ještě jednou. Budeme čekat tak dlouho, dokud nebude vypsán celý odkaz. Je důležité si uvědomit, že na standartní výstup program *CloudCross* při registraci mimo potřebný odkaz vypisuje ještě dodatečně nějaký text (viz *obr. 7.9*). Tento text potřebujeme odseknout, protože uživateli chceme zobrazit jenom odkaz.



Obr. 7.10: Vygenerovaný URL pro registraci ve zvláštním okně



Obr. 7.11: Uživatel má možnost zrušit registraci cloudu

Pak ukážeme vygenerovaný odkaz ve zvláštním okénku (viz *obr. 7.10*) a následné změníme název tlačítka z „*Register*“ na „*Abort Registration*“, abychom poskytli uživateli možnost zrušit proces registrace cloudu (viz *Obr. 7.11*) pro případ, když si registraci rozmyslí.

V předchozím popisu byl proces registrace pouze spuštěn. Ale je potřeba zpracovat i jeho ukončení. Když proces registrace cloudu skončí, je potřeba aktualizovat data na obrazovce (jaké akce budou pro uživatele povolené/zakázané a budou-li zobrazeny seznamy souborů uložených lokálně a vzdáleně v cloudech). Na signál *finished* navážeme vlastní slot *process_registration_finished*.

Nyní probereme, jak vlastně vypadá slot *process_registration_finished*. Zaprvé potřebujeme zjistit, u kterého cloudu (prvního nebo druhého) byla ukončena registrace a až pak provést úpravu zobrazených grafických elementů (povolit nebo zakázat registraci cloudu, stahování a nahrávání souboru). Jestliže máme všechny cloudy správně zaregistrované, je potřeba zobrazit všechny lokální soubory, které má uživatel v adresáři pro synchronizaci, a všechny soubory, které má uživatel v cloudu.

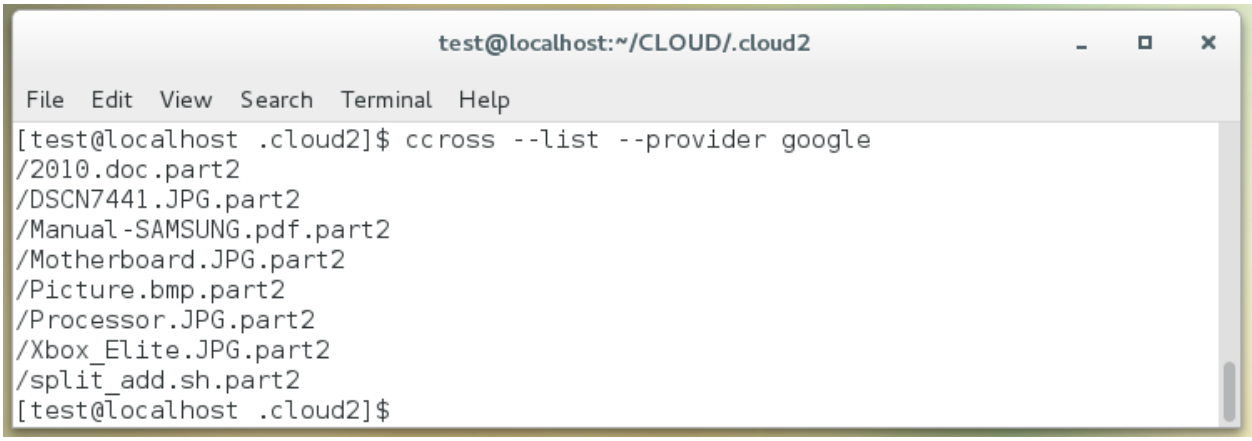
Zobrazení seznamu souborů v lokálním adresáři

Uživatel potřebuje mít přehled o lokálních souborech v synchronizovaném adresáři. Knihovna Qt poskytuje třídu *QFileSystemModel* pro práci se souborovým systémem. Tato třída umí přečíst seznam souborů a jejich atributy, umí sledovat, jaký soubor byl smazán, přidán nebo přejmenován apod., a v každý okamžik obsahuje data a souborech, která odpovídají skutečnosti. Kvůli tomu, že v knihovně Qt se rozlišují datové modely a způsoby reprezentace (grafické prvky, kde se mohou data zobrazovat) je potřeba data z instance datového modelu třídy *QFileSystemModel* nahrát do nějakého grafického prvku, například do objektu třídy *QTreeView*.

Zobrazení seznamu souborů v cloudu

Zobrazení seznamu souborů, které má uživatel uložené v cloudu, není tak triviální, jak se to může jevit. Pomocí programu *CloudCross* můžeme získat seznam souborů uložených v každém z cloudů zvlášť. Ale pro uživatele seznam prvních částí a seznam druhých částí není příliš užitečný (musel by sám spárovat první část souboru s její druhou částí, což chceme dělat automaticky) a navíc tenhle přístup poskytuje navenek spoustu interních technických informací (například jaké části jsou uložené v kterém cloudu, jak se jmenují části apod.). Proto je potřeba ze dvou seznamů souborů (seznamu prvních částí a seznamu druhých částí) vygenerovat seznam souborů uložených v cloudu. Ale může se stát, že pro první část se nenašla odpovídající druhá část (či naopak), anebo je v cloudu uložen soubor, který vlastně není ani první ani druhou částí. Z tohoto důvodu pro uživatele vygenerujeme dva seznamy. První seznam – seznam správně uložených souborů, které si může stáhnout z cloudu, a seznam souborů v cloudu, které jsou nějakým způsobem rozbité. Dále následuje detailní popis generování seznamů správných a rozbitých souborů uložených v cloudu.

Zaprvé je potřeba získat seznam souborů v prvním cloudu a seznam souborů v druhém cloudu. Jelikož na komunikaci s cloudem používáme externí program *CloudCross*, využijeme jeho funkcionalitu. Přepínačem ***--list*** řekneme, že nechceme provádět synchronizaci, ale jenom potřebujeme načíst seznam souborů uložených v cloudu a poté přepínačem ***--provider JMENO_PROVIDERU*** uvedeme typ cloud poskytovatele. V konzoli by to vypadalo jako na *obr. 7.12*



Obr. 7.12: Získání seznamu souborů uložených v cloudu pomoci CloudCross

Je vidět, že seznam souborů se vypisuje na standartní vystup, každý soubor začíná symbolem „/“ a vlastně každý řádek na výstupu reprezentuje jméno právě jednoho souboru. Takže po spuštění externího programu *CloudCross* se správnými parametry dostaneme na standardním výstupu seznam souborů, který je ještě potřeba naparsovat. Proces načtení a parsování je implementován ve funkci *get_remote_lists***.** Tak pro každý cloud máme seznam souborů správně a nesprávně uložených. Celkově je seznam správně uložených souborů tvořen soubory, které mají správně uloženou první i druhou část. Seznam rozbitých souborů se sestává ze čtyř částí:

- Soubory, které mají první část, ale nemají druhou
- Soubory, které mají druhou část, ale nemají první
- Soubory nesprávně uložené v prvním cloudu
- Soubory nesprávně uložené v druhém cloudu

Zobrazíme dva vygenerované seznamy v GUI uživateli.

Stažení všech (nebo vybraných) souborů z cloudu

Na signál *clicked* tlačítka „*Download all*“ je navázáno zpracování stažení všech souborů z cloudu. Logické je, že současně může probíhat jenom jeden proces stahování. Proto na začátku uživateli zakážeme možnost nastartovat další proces stahování souborů z cloudu. Nastavíme správně argumenty pro stahování tak, ze první argument je absolutní cesta k pracovnímu adresáři, druhy argument je typ cloud poskytovatele, kam ukládáme první části, třetí argument je typ cloud poskytovatele, kam ukládáme druhé části. Pak asynchronně spustíme proces stahování (skript ***combine_pull.sh***). Po ukončení procesu stahování je potřeba udělat jen dvě věci. Nejdříve je potřeba zobrazit uživateli log z průběhu stahování a poté povolit spuštění dalších procesů stahování (tlačítkem „*Download all*“, „*Download selected*“). Protože po ukončení procesu stahování bude emitován signál *finished*, tak na něho navážeme slot *proces_download_finished*, který provede i dvě výše uvedené operace.

Stahování vybraných souborů z cloudu se liší od stahování všech souborů jen ve dvou místech. Nejdříve je potřeba zkontrolovat, jestli uživatel označil alespoň jeden soubor ke stažení. V případě, že uživatel nevybral žádný soubor, tak nelze pokračovat. Pokud je soubor (soubory) vybrán, budeme muset zadat více než tři argumenty pro externí proces stahování. Jako v předchozím případě mají první tři argumenty stejný význam, ale všechny další argumenty jsou názvy souborů, které je potřeba stáhnout.

Nahrání všech (nebo vybraných) souborů do cloudu

Implementace procesu nahrávání všech (anebo vybraných) lokálních souborů do cloudu je téměř stejná jako implementace stažení souborů z cloudu. Jediným rozdílem je, že jako externí proces se spouští skript ***split_add.sh*** místo skriptu ***combine_pull.sh*** a všechno je navázáno na tlačítka „*Upload all*“ a „*Upload selected*“.

8. ZÁVĚR

V rámci diplomové práce byly popsány výhody a nevýhody využití cloudových úložišť a cloudových výpočtů. Mezi největší výhody patří dostupnost přes internet a zjednodušení využití pro uživatele (protože potřebují méně znalostí a peněz, budou mít méně starostí s provozem, zálohováním a redundancí dat). Zároveň byly popsány problémy a rizika spojená s využitím cloudu. Největšími problémy cloudových technologií jsou případná nedostupnost, a hlavně ochrana soukromí umístěných dat. Poskytovatel cloudu je pro jednotlivé uživatele i pro organizace nedůvěryhodnou třetí stranou. Bez vhodného zabezpečení dat uživatelé dobrovolně poskytují přístup k osobním, a to často i velmi citlivým datům. Pro organizace to také znamená ztrátu kontroly bezpečnosti nad svými vlastními daty, službami a procesy.

Šifrování dat na straně klienta před použitím cloudu vyřeší problémy spojené s ochranou soukromí a bezpečnosti. Přestože to vypadá jednoduše, není pro klienta cloudu využití tohoto procesu tak snadné. Proto na trhu existují komerční řešení jako Boxcryptor nebo Tresorit, které zjednoduší proces šifrování dat na straně klienta před použitím cloudu (a také zjednodušují i dešifrování). Šifrování dat na straně klienta znemožňuje provádět cloudu výpočty nad šifrovanými daty. Toto se jeví velkou překážkou, a proto se rozvíjejí techniky využívající homomorfní šifrování (Homomorphic encryption), techniky rozdělení dat (Data splitting) a techniky využívající vyhledávatelné šifrování (Searchable encryption).

V současnosti uživatelé ukládají velká množství dat a mají potřebu efektivně se svými daty zacházet, ale zároveň je mít zabezpečená. Vyhledávatelné šifrování poskytuje šifrování dat a zároveň zvyšuje efektivitu práce s daty (poskytuje možnost hledat v šifrovaných datech podle tokenu bez nutnosti dešifrovat samotná data). Ale mnohonásobné hledání v datech po malých kouscích uvolňuje informaci o datech a teoreticky by poskytovatel cloudu mohl poskládat a odvodit některé citlivé informace.

Homomorfní šifrování je rovněž vhodné jak pro cloudová úložiště, tak i pro cloudové výpočty. Umožňuje práci s šifrovanými daty (mohlo by se využit pro elektronické volby nebo peněženky). Plně homomorfní šifrování se jeví jako nejefektivnější možnost jak ukládat data do cloudu. Cloud je schopen pracovat s daty, aniž by znal jejich obsah, a tudíž soukromí uživatelských dat zůstává zachováno. V současnosti však neexistuje efektivní řešení FHE techniky, které by bylo možné implementovat do reálných systémů. Problémem je počet operací, které je možné provádět, a zároveň efektivita výpočtů, která je v současnosti neúměrně nízká. Dalším problémem jsou velikosti klíčů a kryptogramů.

Technika rozdělení dat je vhodná jak pro cloudová úložiště, tak i pro cloudové výpočty. Při práci s cloudovým úložištěm je cílem rozdělit data na části tak, aby jedna část pro cloud nebo pro potenciálního útočníka neměla žádný význam. Při použití v cloudových výpočtech by se rozdělení dat používalo především k rozdělení na smysluplné části pro další nezávislé zpracování různými cloudy.

Pro implementaci byla zvolena technika rozdělení dat pod názvem Bit Split Bit Combine. Výsledkem implementace je aplikace v programovacím jazyce C. Kromě toho byl proveden výkonnostní test, ve kterém byla technika BSBC porovnávána s již existujícími produkčními implementacemi AES šifrování, v knihovnách OpenSSL a AEScrypt. Ukázalo se, že BSBC má výkonnost odpovídající výkonnosti AES šifrování v knihovně OpenSSL a má výkonnost o dva krát lepší než šifrování v knihovně AEScrypt.

V závěru práce byla navržena a implementována aplikace realizující bezpečné ukládání uživatelských dat do cloudu. Aplikace využívá implementovanou metodu data splitting BSBC a aplikaci CloudCross. Navržená aplikace umožňuje práci jak přes konzolové, tak grafické rozhraní. GUI rozhraní rozšiřuje možnosti CLI o registraci aplikace do cloudu a automatickou detekci typu použitého cloudu. Ukládání a nahrávání souborů do/z cloudu je zcela transparentní a nezatěžuje tak uživatele technickými detaily implementace techniky rozdělení dat.

SEZNAM ZKRATEK

AES – Advanced Encryption Standard – Standard pokročilého šifrování (je standardizovaný algoritmus používaný k šifrování dat v informatice)

ASE – Asymmetric Searchable Encryption – Asymetrické vyhledávatelné šifrování

API – Application Programming Interface – Rozhraní pro programování aplikací

BSBC – Bit Split Bit Combine – Název techniky rozdělení dat

CLI – Command Line Interface – Řádkové rozhraní

ENISA – The European Union Agency for Network and Information Security

FHE – Fully Homomorphic Encryption – Plně homomorfní šifrování

GUI – Graphical User Interface – Grafické uživatelské rozhraní

HE – Homomorphic Encryption – Homomorfní šifrování

IaaS – Infrastructure as a Service – Infrastruktura jako služba

IDE – Integrated Development Environment – Vývojové prostředí

PaaS – Platform as a Service – Platforma jako služba

PHE – Partially Homomorphic Encryption – Částečně homomorfní šifrování

RSA – Initial letters of the author's surnames (Rivest, Shamir, Adleman) – je asymetrické kryptografické schéma využívané pro šifrovací a podpisové účely v informatice.

SaaS – Software as a Service – Software jako služba

SE – Searchable Encryption – Vyhledávatelné šifrování

SDK – Software Development Kit

SSE – Symmetric Searchable Encryption – Symetrické vyhledávatelné šifrování

SHE – Somewhat Homomorphic Encryption – Poněkud homomorfní šifrování

URL – Uniform Resource Locator (or shortly named web address) – Jednotná adresa zdroje

LITERATURA

[1] LICKLIDER J.C.R. Man-Computer Symbiosis. In: *Transactions on Human Factors in Electronics*, volume HFE-1, s. 4–11, March 1960.

[2] BERNERS-LEE T. (CERN) *Information Management: A Proposal*. [online] March 1989, May 1990 https://www.w3.org/History/1989/proposal.html

[3] What is cloud computing. In: Microsoft Azure [online]. [cit. 2016-11-12]. Dostupné z: https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/

[4] Cloud Computing Risk Assessment. [online]. The European Union Agency for Network and Information Security, last modified on 09.11.2009 [cit. 2016-11-12]. Dostupné z: https://www.enisa.europa.eu/publications/cloud-computing-risk-assessment

[5] Is Dropbox safe to use? In: Dropbox Help. [online] [cit. 2016-11-12]. Dostupné z: https://www.dropbox.com/help/27

[6] Security. In: Google Cloud Help. [online] [cit. 2016-11-12]. Dostupné z:

https://support.google.com/work/answer/6056693?hl=en

[7] Boxcryptor. *Highest Security for your Files in the Cloud* [online]. [cit. 2016-11-12]. Dostupné z: https://www.boxcryptor.com

[8] Tresorit. *Tresorit*. [online] [cit. 2016-11-12]. Dostupné z: https://tresorit.com/

[9] ZHANG W., SUN X., XU T. Data Privacy Protection Using Multiple Cloud Storages. In: *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, Shengyang, 2013, pp. 1768-1772.

[10] DANWEI C., YANJUN H. A Study on Secure Data Storage Strategy in Cloud Computing. In: *Journal of Convergence Information Technology* 5(7):175-179· September 2010.

[11] CALVIÑO, A.; RICCI, S.; DOMINGO-FERRER, J. Privacy-preserving distributed statisticalcomputation to a semi-honest multi-cloud. In: *Communications and Network Security (CNS), 2015 IEEE Conference on*. IEEE, 2015. p. 506-514.

[12] SALAM, M.I., YAU, WC., CHIN, JJ. aj. *Implementation of searchable symmetric encryption for privacy-preserving keyword search on cloud storage*. Human-centric Computing and Information Scientists. [online] (2015) 5: 19. doi:10.1186/s13673-015-0039-9.

[13] SONG X. D.; WAGNER D.; PERRIG A. Practical Techniques for Searches on Encrypted Data. In *Proceedings of the IEEE Symposium on Security and Privacy – SP 2000*, IEEE Computer Society, 2000.

[14] DZURENDA, P.; HAJNÝ, J. *Techniky homomorfního šifrování a jejich praktické využití*. *Elektrorevue* [online]. 2014-4-20 [cit. 2016-4-24]. ISSN 1213-15391213-1539.

[15] BONEH D.; GOH E.; NISSIM K. Evaluating 2-DNF Formulas on Ciphertexts. In: *Proceedings of Theory of Cryptography (TCC) '05*, LNCS 3378, pp. 325-341, 2005.

[16] GENTRY, C. *A fully homomorphic encryption scheme*. Dizertační práce, Stanford University, 2009.

[17] GENTRY, C. Fully homomorphic encryption using ideal lattices. In: *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing -STOC '09*, ACM Press, 2009, s. 169-178.

[18] DIJK, M.; GENTRY, C.; HALEVI, S. aj. *Fully Homomorphic Encryption over the Integers*. In *Advances in cryptology - EUROCRYPT 2010 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30- June 3, 2010*, Springer, 2010, s. 24-43.

[19] GENTRY, C.; HALEVI, S. Implementing Gentry's Fully-Homomorphic Encryption Scheme. In: Advances in Cryptology - EUROCRYPT 2011 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings, Springer Berlin Heidelberg, 2011, s. 129-148.

[20] GRUSCHKA N.; JENSEN M. Attack Surfaces: A Taxonomy for Attacks on Cloud Services. In: *2010 IEEE 3rd International Conference on Cloud Computing*. [online] Dostupné z: http://barbie.uta.edu/~hdfeng/CloudComputing/cloud/cloud25.pdf

[21] Dropbox API v2. In: Dropbox for HTTP Developers. [online] [cit. 2017-04-28]. Dostupné z: https://www.dropbox.com/developers/documentation/http/documentation

[22] API Reference. In: Google Drive APIs. [online] [cit. 2017-04-28]. Dostupné z: https://developers.google.com/drive/v3/reference/

[23] Qt 5.8. In: Qt Documentation. [online] [cit. 2017-04-28]. Dostupné z: http://doc.qt.io/qt-5/index.html

[24] CloudCross. [online] [cit 2017-03-31]. Dostupné z: https://cloudcross.mastersoft24.ru/

[25] CloudCross. In: Github. [online] [cit 2017-03-31]. Dostupné z: https://github.com/MasterSoft24/CloudCross

[26] QProcess Class. In: Qt Documentation. [online] [cit. 2017-04-28]. Dostupné z: http://doc.qt.io/qt-5/qprocess.html

OBSAH ELEKTRONICKÉ PŘÍLOHY

Přiložené CD obsahuje následující soubory a adresáře:

- diplomova_prace.pdf(text práce)
- src_datasplitting (zdrojové kódy)
- src_cli (zdrojové kódy)
- src_gui (zdrojové kódy)