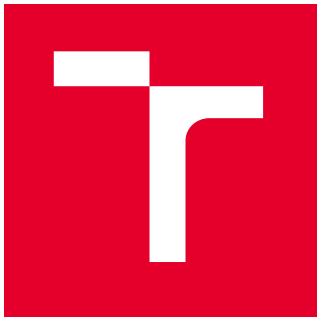


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

ARCHIVACE DAT S VYUŽITÍM TECHNOLOGIE BLOCKCHAIN

DATA ARCHIVING USING BLOCKCHAIN TECHNOLOGY

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Tobiáš Řihánek**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Vlastimil Člupek, Ph.D.**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Tobiáš Řihánek ***ID:*** 191814

Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Archivace dat s využitím technologie blockchain

POKYNY PRO VYPRACOVÁNÍ:

V diplomové práci analyzujte možnosti využití technologie blockchain pro archivaci elektronických dat. Navrhněte webový archivační systém využívající technologii blockchain k zabezpečení dat. Archivační systém bude zajišťovat autentičnost, integritu a volitelně důvěrnost archivovaných dat. Pro přístup do systému bude vyžadováno ověření identity uživatele. Implementujte navržený webový archivační systém využívající technologii blockchain a ověřte jeho funkčnost. Určete hardwarové a softwarové požadavky vytvořeného systému a úroveň zabezpečení archivovaných dat z hlediska dlouhodobé archivace. Získané výsledky přehledně prezentujte.

DOPORUČENÁ LITERATURA:

[1] HWANG, Hyun Cheon; SHON, Jin Gon; PARK, Ji Su. Design of an Enhanced Web Archiving System for Preserving Content Integrity with Blockchain. Electronics, 2020, 9.8: 1255.

[2] BRALIĆ, Vladimir; STANČIĆ, Hrvoje; STENGÅRD, Mats. A blockchain approach to digital archiving: digital signature certification chain preservation. Records Management Journal, 2020.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 24.5.2022

Vedoucí práce: Ing. Vlastimil Člupek, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato práce se zabývala problematikou archivace dat. Cílem bylo prozkoumat možnosti využití technologie blockchain k řešení bezpečnostních aspektů digitální archivace, a poté navrhnout a implementovat aplikaci využívající blockchain k zabezpečení integrity dat. Nejdříve byl proveden teoretický rozbor jak archivace, tak technologie blockchain. Dále došlo k návrhu archivačního systému, který tuto technologii využívá k detekci porušení integrity. Na základě návrhu byl implementován systém sestávající z webové aplikace napsané v jazyce JavaScript a archivační aplikace a validační aplikace, napsané v jazyce Java. Tyto aplikace mezi sebou komunikují a vytvářejí bezpečný archivační systém využívající blockchain. Došlo k demonstraci funkčnosti systému a byla otestována jeho bezpečnost.

KLÍČOVÁ SLOVA

blockchain, archivace dat, integrita dat, autentizace, digitální archiv, webový archiv

ABSTRACT

This thesis dealt with the issue of data archiving. The aim was to explore the possibilities of using blockchain technology to address the security aspects of digital archiving, and then to design and implement an application that uses blockchain to secure data integrity. First, a theoretical analysis of both archiving and blockchain technology was conducted. Next, a web application was designed that uses this technology to detect integrity breaches. Based on the design, a system consisting of a web application written in JavaScript and an archiving application and a validation application written in Java was implemented. These applications communicate with each other and create a secure archiving system using blockchain. The functionality of the system was demonstrated and its security was tested.

KEYWORDS

blockchain, data archiving, data integrity, authentication, digital archive, web archive

ŘIHÁNEK, Tobiáš. *Archivace dat s využitím technologie blockchain*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 90 s. Diplomová práce. Vedoucí práce: Ing. Vlastimil Člupek, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Tobiáš Řihánek
VUT ID autora:	191814
Typ práce:	Diplomová práce
Akademický rok:	2021/22
Téma závěrečné práce:	Archivace dat s využitím technologie blockchain

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné 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 závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné 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 následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (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.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Dnešní digitalizující se společnost je založena na informacích a datech. Odhaduje se, že v roce 2020 bylo každý den vytvořeno 2,5 trilionů bytů informací. [1] Toto číslo raketově roste s tím, jak se větší a větší části lidských životů, státních institucí a ekonomiky přesouvá do digitální sféry. Významná část těchto dat pak vyžaduje dlouhodobou archivaci, a tím pádem je naprosto klíčové zajistit jejich spolehlivé ukládání.

Ve spojitosti s tím tak vyvstávají důležité otázky. Jak například zajistíme, že archivovaná data zůstávají neměněna? Jak případnou změnu detekujeme? Jak si můžeme být jistí původem archivovaných dat?

Technologie řešící tyto problémy sice existují, ale žádná metoda není perfektní. Hledání alternativních a elegantnějších řešení tak stále pokračuje. Jedním takovým řešením by mohla být, poslední dobou často diskutovaná, technologie blockchain. I když je blockchain nejčastěji spojován s oblastí kryptoměn, jeho využití může být mnohem širší - od elektronických voleb po archivaci dat.

Tato diplomová práce se bude věnovat využitím technologie blockchain v procesu archivace dat. Práce je rozdělena na dvě části - teoretickou a praktickou. V první kapitole dojde k teoretickému rozboru nejdůležitějších aspektů ukládání a archivace dat, bezpečnosti digitálních archivů a využívaných kryptografických primitiv. Druhá kapitola se bude zabývat technologií blockchain - dojde k vysvětlení základních principů, na kterých funguje a budou popsány různá využití včetně již existujících návrhů spojených s archivací dat.

Praktická část obsahuje dvě kapitoly. V první dojde k návrhu archivačního systému využívajícího technologii blockchain. Součástí návrhu je popis základních funkcí a dále zabezpečení systému. V poslední části dojde k implementaci navrženého systému, v textu budou popsány principy implementace včetně výsledků. Dojde k demonstraci implementovaného systému a jeho otestování včetně posouzení bezpečnost. Dojde taky ke krátké diskuzi nad výsledky práce.

1 Archivace dat

Pod pojmem archivace dat se obecně myslí dlouhodobé uchování dat na základě regulačních požadavků, či za účelem pozdějšího dohledání a prohlížení. Může se jednat o finanční záznamy, faktury, obsahy databází, smlouvy, ale archivovat lze v podstatě jakákoliv data. Typickým případem užití jsou i historické archivy. Archivací dat není myšleno pouhé uložení dat, na archivy jsou kladeny mnohé specifické požadavky a důraz je na dlouhodobé uchování. [2]

1.1 Digitální archiv

Digitální archivy jsou elektronickým ekvivalentem klasických archivů. Mohou obsahovat buď digitální dokumenty, nebo digitalizované fyzické dokumenty. Jejich hlavním účelem je dlouhodobé zachování dat. [3]

Digitální archivy mohou být otevřené veřejnosti [3], jako příklad může posloužit Vojenský ústřední archiv. Mimo jiné obsahuje databázi padlých vojáků a legionářů, která je přístupná široké veřejnosti. [4]

Druhou možností jsou interní digitální archivy, které jsou přístupné pouze v rámci nějaké organizace, firmy či skupiny uživatelů. Může se jednat o archivy finančních dokumentů nějaké firmy, archivy tajných informací, archivy obsahující osobní informace apod. Zde musí být kladen větší důraz na řízení přístupů a autentizaci.

1.2 Požadované vlastnosti

Základní vlastností digitálních archivů je možnost dlouhodobého ukládání dat. Následně mohou k těmto datům přistupovat uživatelé a pracovat s nimi. Archiv by tak měl mít schopnost vyhledávání a zobrazování dokumentů. Dokumenty v archivu mají většinou formu obsahu a metadat. [5] Uložení obsahu samotného totiž není dostatečné k dalšímu používání archivu. Údaje jako datum přidání do archivu, jméno archiváře, který dokument přidal, se tak ukládají zvlášť ve formě metadat. Metadat může být velké množství a vždy záleží na specifikách daného archivu. Dělí se mimo jiné na primární a sekundární. [6]

Primární metadata jsou spojená se samotným souborem - jeho název, formát, velikost, rozlišení (v případě vizuálních dat), čas vytvoření a modifikace, autor a další. Tato metadata jsou většinou vytvářena automaticky při manipulaci se souborem ale modifikace může být umožněna. [6]

Sekundární metadata jsou přidávána manuálně během manipulace s dokumenty. Jedná se o metadata spíše spojená se samotným archivem - popis dokumentu, lokace souboru v souborovém systému, identifikační čísla, údaje umožňující audity (původ

dat, jejich historie, apod.) a spousta dalších. Většinou jsou změny metadat povoleny jen autorizovaným uživatelům. [6]

Dalším typem metadat jsou odvozená metadata, která vznikají automaticky během analýzy pomocí programů a algoritmů. Typicky jde například o vyhledávače, které prochází soubory a indexují je na základě svých potřeb. [6]

1.3 Zabezpečení digitálních archivů

Jelikož mohou archivní dokumenty být důvěrné povahy, mají archivy specifické nároky na bezpečnost. Stěžejní je zajistit jejich integritu - tedy zabránit tomu aby nějaký útočník změnil již archivovaná data. Archiv, který nedokáže zajistit integritu jeho obsahu, nesplňuje svůj základní účel. [7]

Důležité je také řízení přístupu pro jednotlivé skupiny uživatelů. Přidávání nových dokumentů je většinou omezeno jen na specifickou skupinu uživatelů - archivářů. Manipulace s dokumenty tak může být omezena na základě toho, jaký přístup má daný uživatel. [7]

Dalším aspektem bezpečnosti je důvěrnost dat, obzvláště pokud se jedná o citlivá či osobní data. [7] V případě, že se potenciálnímu útočníkovi nebo neautorizovanému uživateli podaří dostat k takovýmto datům, neměli by být schopni je přečíst. Toho lze dosáhnout pomocí šifrování. Data nejsou ukládána v otevřené podobě, ale předtím jsou zašifrována. Dešifrována jsou až v případě nutnosti, a to pouze autorizovanými uživateli.

1.3.1 Integrita dat

Zajištění integrity dat znamená zabezpečení dat proti neautorizované modifikaci. Může se týkat jak uložených dat, tak dat právě přenášených. V prvním případě jde tedy o to zajistit stav, kdy uložená data nelze modifikovat bez vědomosti autorizovaných uživatelů. [8] Pokud k takovéto modifikaci dojde, měl by systém být schopen toto porušení integrity minimálně zachytit. Ve druhém případě jde o to zajistit, aby data, která jeden uživatel posílá druhému, nebyla po cestě modifikována.

Porušení integrity může být způsobeno buď důsledkem nějakého nežádoucího jevu, jako je šum, nebo v důsledku úmyslného pozměnění. Prostředky, kterými je integrita většinou zajištěna, jsou kontrolní součty, hashovací funkce a elektronické (digitální) podpisy. [9]

Kontrolní součty

Kontrolní součet je informace, která se přidá k datům a slouží k ověření integrity těchto dat. Na vstupu jsou data, která prochází algoritmem, jehož výstupem je

kontrolní součet. Při validaci se tak kontrolní součet přepočítá, a pokud se kontrolní součty shodují, data zůstala neporušena.[10]

Algoritmů, které jsou pro kontrolní součty používány, je celá řada. Asi nejjednodušším algoritmem je parita. Jako kontrolní součet se vezme takzvaný paritní bit. Ten má hodnotu buď 0 nebo 1, v závislosti na počtu bitů s hodnotou 1 v poli dat. Pokud je používána sudá parita a v datech je sudý počet jedniček, je kontrolní součet také 1. Pokud by počet jedniček byl lichý, kontrolní součet by byl 0. [11]

Mezi další klasické algoritmy pro kontrolní součty patří Hammingův kód, Cyklické Redundantní Kódy (CRC) ale i hashovací fuknce.

Hashovací funkce

Hashovací funkce jsou kryptografické funkce, sloužící k vytváření otisku nějakých dat. Tento otisk se nazývá hash. [12] Matematicky je lze popsat následovně:

$$H = f_h(M)$$

kde
H – výsledný hash
f_h - hashovací funkce
M – vstupní data

Hashovací funkce pak musí splňovat následující vlastnosti:

- Determinismus - výsledný hash má vždy stejnou délku, nezávisle na délce vstupních dat. Například hash jedné věty bude mít stejnou délku jako hash celé knihy [12]
- Jednocestnost - Zatímco vypočítat hash z M je relativně jednoduché, vypočítat M z hashe musí být extrémně náročné
- Bezkoliznost 1. řádu – Je velmi těžké najít M a M', takové aby $f_h(M)=f_h(M')$
- Bezkoliznost 2. řádu – Pro zadané M je velmi těžké najít M', takové aby $f_h(M)=f_h(M')$
- Malá změna vstupních dat vyústí ve velkou změnu hashe – nelze tedy odvodit původní data
- Rychlost – Výpočet hashe by měl být co nejrychlejší

Díky těmto vlastnostem jsou hashovací funkce naprosto stěžejními kryptografickými prostředky. Jejich využití je široké, ale mezi hlavní patří ochrana integrity dat, kde jsou používány jako obdoba kontrolního součtu. Při validaci integrity je porovnáván hash aktuálních dat s uloženým hashem. [12]

Pomocí hashovacích funkcím lze také bezpečně ukládat citlivá data, jako jsou například hesla. Místo ukládání samotných hesel, což je nebezpečné, se ukládají

pouze jejich hashe. Z vlastností hashovacích funkcí vyplývá, že vypočítat heslo z jeho hashe je velmi náročné. Při autentizaci pomocí hesla se pak pouze porovnávají jejich hashe. Díky bezkoliznosti je extrémně vzácné, aby dvě hesla měla stejný hash. [12]

Pokud je hashovací funkce dobře navržena, jediný způsob, jakým zjistit z hashe původní heslo, je útok hrubou silou – zkoušet hashovat různá hesla, dokud není nalezen shodný hash. V případě vhodně zvolených hesel je však tento způsob časově neuskutečnitelný. [12]

Dnes nejpoužívanějšími hashovacími funkcemi jsou SHA-1 a hashe z rodiny SHA-2, jako SHA-256 nebo SHA-512. Poměrně novým algoritmem je SHA-3. Dříve používaná funkce MD5 je dnes již považována za prolomenou a její používání není doporučeno.

1.3.2 Řízení přístupu, autentizace a autorizace

Řízení přístupu je metoda garance toho, že uživatelé, kteří interagují s nějakým systémem jsou, kdo tvrdí, že jsou, a že mají pouze taková práva, jaká jim přísluší. Skládá se ze dvou komponent – autentizace a autorizace. [13]

Autentizace je způsob ověření identity daného uživatele. Autorizace je pak ověření oprávnění pro určitou činnost. Jako příklad může posloužit databáze, kde někteří uživatelé mohou pouze prohlížet data a jiní je i upravovat. Všichni uživatelé se sice dokáží k databázi autentizovat, ale autorizaci mají odlišnou.

Způsobů udělování práv různým uživatelům je několik, mezi hlavní patří:

- Discretionary Access Control (DAC)
- Mandatory Access Control (MAC)
- Role Based Access Control (RBAC)
- Attribute Based Access Control (ABAC)

DAC je nejméně centralizovaný a spočívá v tom, že jakmile dostane uživatel určitá práva, může je podle vlastního úsudku předávat ostatním uživatelům. [13] [14]

MAC je nejstriktnější a spočívá v tom, že administrátor dává každému uživateli práva na základě nějakých pravidel. Tato pravidla jsou určována nějakou centrální autoritou. [13] [14]

RBAC třídí práva do různých rolí od nejméně privilegovaných k nejvíce privilegovaným. Každému uživateli je pak přidělena určitá role a má práva pouze této role. [13] [14]

ABAC rozděluje práva na základě nějakých atributů. Tyto atributy mohou být i externí, jako například čas, lokace apod. [13]

Metod autentizace uživatelů existuje celá řada, ale mezi nejčastější patří následující [15]:

- Autentizace heslem – uživatelé znají svoje tajné heslo, kterým se autentizují u služby

- Dvoufaktorová autentizace (2FA) - kombinace hesla a další autentizační metody – například zaslání SMS zprávy nebo e-mailu s PIN kódemkódem
- Autentizace tokenem – většinou se jedná o fyzický token, např. USB dongle, RFID kartu nebo NFC čip
- Biometrické údaje – uživatel se autentizuje pomocí unikátních biologických charakteristik svého těla - např. otisky prstů nebo tváře

Každá metoda má své výhody a nevýhody. Hesla jsou jednoduchá na implementaci, na druhou stranu jsou náchylnější na odcizení oproti jiným metodám. Tokeny a biometrické údaje jsou bezpečnější, protože je těžší je odcizit, ale vyžadují specializovaný hardware (např. čtečky otisků prstů).

Dvoufaktorová autentizace je často používaný kompromis – pro uživatele je náročnější než pouhé zadání hesla, ale autentizace je mnohem bezpečnější a není vyžadován speciální hardware.

Implementace bezpečné autentizace a následná správa identifikačních údajů může být velmi náročná a proto se často nabízí a vyplácí využití autentizační platformy třetí strany. Ve své podstatě je autentizace delegována důvěryhodné třetí straně, která již zajistí co nejvyšší míru zabezpečení. Platforma pak umožňuje správu uživatelů, autorizaci, udělování práv a další funkce. Vše bez nutnosti vlastní implementace.

Identifikační údaje jsou u těchto služeb ukládány v cloudovém úložišti dané služby, není tedy potřeba řešit jejich bezpečné uchovávání. Jedná se tak o velmi jednoduchý a elegantní způsob jak zajistit řízení přístupu. [16] Mezi takovéto platformy patří auth0, Firebase nebo Amazon Cognito. Velmi často lze tyto služby nabízí i tzv. Single Sign-On (neboli SSO).

Single Sign-On

SSO dává uživatelům možnost použít stejné přihlašovací údaje do různých aplikací bez nutnosti vytvářet účty pro každý z nich. Největší výhodou je uživatelská přívětivost ale SSO také zjednodušuje správu uživatelů, především v rámci velkých organizací. Z hlediska bezpečnosti pak může být výhoda v tom, že si uživatel musí pamatovat pouze jedno heslo, které pak může být komplexnější. Na druhou stranu, pro útočníka stačí najít právě toto heslo a dostane přístup k vícero aplikacím. Proto dává smysl vždy kombinovat SSO s dvou-faktorovou autentizací. [17]

1.3.3 Důvěrnost dat

Zabezpečení důvěrnosti dat především v kontextu datových úložišť a archivů znamená zamezení neautorizovanému čtení informací. Řeší tak případy, kdy útočník dokáže překonat autorizaci a dostane se k uloženým informacím. Pokud je zajištěna jejich důvěrnost, neměl by být schopen je přečíst. [18, kap. 1.4]

Kryptografická služba, která tento problém řeší, je šifrování. Šifrování je proces, při kterém jsou vstupní data transformována za použití klíče. Dešifrování je proces opačný. Šifrování se dělí na 2 základní typy:

- Symetrické – klíč pro šifrování a dešifrování je stejný, dělí se na proudové a blokové
- Asymetrické – klíč pro šifrování je rozdílný od klíče pro dešifrování

Symetrické proudové šifry

Data jsou u proudových šifer šifrována po jednotlivých symbolech (tedy bit po bitu, popřípadě bajt po bajtu). Kromě proudu dat je na základě algoritmu a vstupního klíče vygenerován proud klíčů. Tyto dva proudy následně projdou operací XOR a je vytvořen zašifrovaný proud. Na druhé straně je stejným způsobem generován identický proud klíčů a opět se provede operace XOR. Díky vlastnostem operace XOR je výsledkem původní proud dat. [18, kap. 8.4]

Tento typ proudové šifry se nazývá synchronní, protože vyžaduje perfektní synchronizaci zdroje a příjemce. V momentě, kdy se během přenosu ztratí jediný symbol, šifra se desynchronizuje a příjemce je nebude schopný dešifrovat. Řešení nabízí asynchronní proudové šifry, které proud klíčů generují i na základě N předešlých šifrovaných symbolů. Tím pádem se každých N symbolů obě strany znovu synchronizují.

Největší výhodou těchto šifer je rychlost šifrování, proto mají využití v situacích, kde je potřeba co nejnižší možné zpoždění. Nevýhodou je nízká míra takzvané difúze – tedy jeden symbol na vstupních datech je šifrován do jednoho symbolu v šifrovaných datech. Jeho změna se tedy v šifrovaném proudu promítne pouze pro daný symbol. Příklad takovéto šifry je RC4. [18, kap. 8.5]

Symetrické blokové šifry

Tyto šifry šifrují data po blocích různých velikostí. Data jsou rozdělena do bloků a každý blok projde šifrovacím algoritmem, na jehož vstupu je klíč. Výstupem jsou pak zašifrované bloky. K dešifrování se pak použije inverzní funkce.

Výhodou symetrických šifer je velká míra difúze – změna jednoho symbolu v bloku se promítne do změny celého šifrovaného bloku. Oproti proudovým šifrám mají blokové větší zpoždění právě kvůli nutnosti šifrování po blocích fixní délky.

Většina blokových šifer jsou kaskádové šifry, což znamená, že jeden blok šifrují iterativně v rundách. Pro každou rundu je použit jiný, tzv. rundovní klíč, který vychází ze vstupního klíče. Příkladem takovéto struktury je Feistelova síť, která byla používána mimo jiné šifrou DES (Data Encryption Standard). [18, kap. 4.1]

Asi dnes nejpoužívanější blokovou šifrou je AES (Advanced Encryption Standard). Velikost bloků je 128 bitů, velikost klíčů může být 128, 192 nebo 256 bitů. Počet

rund závisí na velikosti klíče - 10, 12 nebo 14. Každý blok je převeden do matice o velikosti 4 x 4, kde každý prvek je jeden byte. [18, kap. 6.2] V každé rundě jsou provedeny čtyři operace:

- Sub Bytes– nahradí každý byte za jiný podle vyhledávací tabulky S-box
- Shift Rows – první řádek matice zůstane stejný, druhý řádek se posune o 1 byte doleva, třetí o 2 byty doleva a poslední o 3 byty doleva
- Mix Columns - každý sloupec je vynásoben předem danou maticí
- Key Addition – je proveden XOR s maticí rundovního klíče

Operace Key Addition je provedena ještě před první rundou a v poslední rundě je vynechána operace Mix Columns. Rundovní klíče jsou pak vytvářeny z klíče předchozí rundy (první klíč je vytvářen z počátečního klíče) pomocí speciální funkce g a operace XOR, prováděné mezi jednotlivými čtveřicemi klíče. [18, kap. 6.3]

Šifra AES má velmi široké využití, díky své rychlosti a malé výkonové náročnosti. Lze ji používat i při šifrování uložených dat, jak je tomu například u aplikace BitLocker. [19]

Bloky u těchto šifer jsou mezi sebou často řetězené, aby nedošlo k tomu, že dva shodné bloky vyústí ve stejný šifrovaný blok, což umožňuje kryptoanalýzu. Po zřetězení tak záleží nejenom na datech v bloku, ale i na jeho pozici v rámci všech šifrovaných dat. [18, kap. 7.3] Existuje několik módů, mimo jiné:

- ECB – bloky nejsou řetězeny
- CBC – před šifrováním je pro každý blok nejdříve provedena operace XOR se zašifrovaným předchozím blokem, u prvního bloku je použit IV (Inicializační Vektor)
- CFB – nejdříve se zašifruje IV, který je následně XOR-ován s blokem dat, čímž vznikne šifrovaný blok. Tento blok je znovu zašifrován a XOR-ován s dalším blokem dat. Funguje tedy podobně jako proudové šifry.
- CTR – tzv. Counter mód. Převádí blokovou šifru na proudovou tak, že na vstupu je náhodné číslo, které je zašifrováno čímž vznikne proud klíčů. Ten je XOR-ován s blokem dat. Číslo je následně iterováno pro další blok dat.

Asymetrické šifrování a elektronický podpis

Asymetrické šifry jsou specifické tím, že k šifrování a dešifrování jsou použity různé klíče. Klíči používanému k šifrování se říká veřejný klíč, klíči k dešifrování se říká soukromý klíč. Jak již název napovídá veřejný klíč může být veřejně známý, protože by z něj nemělo být možné vypočítat soukromý klíč. [18, kap. 9.1]

Asi nejznámější a nejpoužívanější asymetrickou šifrou je RSA (Rivest–Shamir–-Adleman). Matematický problém, který zajišťuje bezpečnost tohoto algoritmu, je faktorizace prvočísel – tedy rozdělení součinu na činitele v případě, že se jedná

o prvočísla. U dostatečně velkých čísel se jedná o extrémně výpočetně náročný problém. [18, kap. 9.2]

Vytváření klíčů probíhá následovně:

- 1. Vygenerují se velká, rozdílná prvočísla p a q.
- 2. Vypočítá se jejich součin n = pq
- 3. Vypočítá se Eulerova funkce n. Díky tomu, že se jedná o prvočísla, ji lze vypočítat jako $\varphi(n) = (p-1)(q-1)$.
- 4. Zvolí se číslo e, které je nesoudělné s $\varphi(n)$
- 5. Vypočítá se číslo d, takové že $d=e^{-1} \bmod \varphi(n)$

Veřejný klíč je pak dvojice (e,n). Soukromý klíč je dvojice (d,n). Bezpečnost spočívá v tom, že bez znalosti p a q je velmi náročné získat $\varphi(n)$, a tím pádem i d.

Šifrování zprávy m pak probíhá podle vzorce:

$m^e \equiv c \,(mod\, n)$

Dešifrování probíhá obdobně:

$c^d \equiv m \,(mod\, n)$

kde m je původní zpráva a c je zašifrovaná zpráva.

Velikost m je omezena velikostí n.

Algoritmus RSA lze jednoduše použít i k elektronickému podepsání zprávy. Podpis je vytvořen zašifrováním zprávy pomocí soukromého klíče. Během ověření je pak podpis dešifrován veřejným klíčem, a pokud dešifrovaný podpis souhlasí s původní zprávou, je podpis validní. [18, kap. 9.2] Předtím, než se nějaká data podepíší většinou projdou nějakou hashovací funkcí, protože podpis má při použití RSA stejnou délku jako podepisovaná data.

2 Blockchain

Blockchain (volně přeloženo jako "řetězec bloků") je v dnešní době velmi populární technologií s širokým využitím. V zásadě se jedná o způsob záznamu dat ve struktuře složené z bloků. Při vytváření nových záznamů jsou do série přidávány další a další bloky. Ty na sebe vzájemně odkazují a využitím různých kryptografických metod je dosaženo toho, že starší záznamy trvale zůstávají zaznamenány a je téměř nemožné je změnit. [20, kap. 2.1]

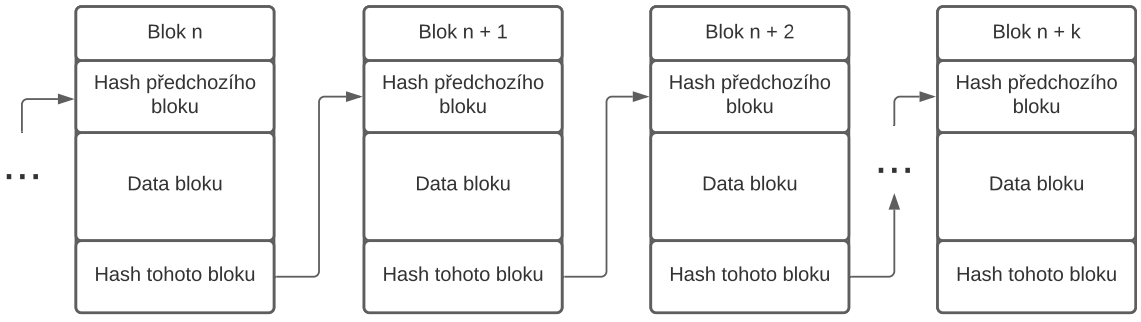
Implementace blockchainu se liší v mnoha ohledech. Typickým rysem je například decentralizace a distribuovanost, ale blockchain lze bez problémů využít i centralizovaně. Typickým a asi nejvýznamnějším využitím této technologie jsou kryptoměny. V tomto případě blockchain hraje roli účetní knihy, do které jsou zaznamenávány jednotlivé transakce. V případě elektronických voleb může blockchain představovat záznamy volebních lístků. A případů je mnohem víc.

2.1 Blok

Základním stavebním kamenem blockchainu jsou samotné bloky. Každý blok musí povinně obsahovat alespoň následující tři informace:

- Samotný záznam - např. informace o transakci
- Hash předchozího bloku
- Hash samotného bloku

Řetězení bloků lze vidět na obrázku 2.1. Hashe slouží především k validaci blockchainu. Pokud je záznam nějakým způsobem změněn, změní se i jeho hash. Toto lze velmi jednoduše použít k ověřování jednotlivých záznamů v každém bloku. Samozřejmě je možné změnit jak obsah, tak i hash bloku. V tom případě však dojde k porušení blockchainu, protože navazující blok neobsahuje správný hash předchozího bloku. [20, kap. 2.1.1]



Obr. 2.1: Řetězení bloků v blockchainu

V závislosti na užití a implementaci však mohou bloky obsahovat mnoho dalších údajů. Jsou jimi například:

- Časové razítko - např. kdy byl blok vytvořen
- Číslo bloku - o kolikátý blok v pořadí se jedná

Tyto údaje mohou sloužit k jednoduššímu procházení a validaci. V blocích se však mohou nacházet i složitější údaje. Dobrý příklad lze najít u blockchainu kryptoměny Ethereum. Bloky zde obsahují i údaj state root. Jedná se o kořen Hashovacího stromu uchovávajícího stav celého systému. [21]

Hashovací stromy

Hashovací strom, také **Merkleův strom** je speciální datovou strukturou ve tvaru stromu. Listy tohoto stromu jsou samotná data - v případě Etherea například zůstatek na účtu některého z uživatelů. Každý vrchol stromu pak obsahuje hash jeho dětí. Přidáváním dalších dat vznikají nové listy, a tím pádem i nové vrcholy obsahující hashe těchto dat. Výsledkem je, že kořen stromu obsahuje hash, který je výsledkem všech předchozích hashů, a tím pádem i všech dat. [20, kap. 2.1.1]

Jedná se o velmi efektivní způsob dokázání integrity velkého množství dat. Lze si například představit, že je potřeba zajistit integritu 1000 různých transakcí bez toho, aniž by bylo nutné ukládat 1000 hashů. Transakce jsou uloženy do stromové struktury jako jednotlivé listy. Následně jsou dopočítány hashe jednotlivých uzlů, až se dospěje ke kořenu, který má finální hash. [20, kap. 2.1.1]

Uložit pak stačí hash kořenu stromu a jeho strukturu (kde ve stromu se nachází která data). Je však možné uložit celou strukturu včetně jednotlivých hashů, což přináší další výhody. Přidání nového listu potom teoreticky vyžaduje pouze přepočet hashů na jeho větvi bez nutnosti přepočítávání celého stromu. Získání nového hashe kořenu tak může být podstatně rychlejší. [20, kap. 2.1.1]

V případě bloku Ethereum jsou do stromové struktury ukládány tři údaje - zůstatky na účtu, jednotlivé transakce a stav celého systému. Je nutno podotknout, že samotné Ethereum používá speciální strukturu Merkle Patricia Trie, její princip je ale velmi podobný. Každý blok obsahuje kořen této struktury. [21]

2.1.1 Přidávání bloků

U centralizovaných systémů je přidávání bloků velmi jednoduché - vlastník blokchainu vytvoří nový blok, vypočítá správné hashe a následně ho zařadí na konec série. V případě distribuovaných blockchainů je situace mnohem složitější. Jelikož neexistuje jedna autorita, která by blockchain spravovala, musí existovat mechanismus, pomocí kterého jsou nové bloky kolektivně přidávány do sdíleného blockchainu. [20, kap. 2.1.3] [22, kap. 4.1.3]

Naivním řešením by mohlo být přidání všech nově vytvořených bloků bez jakékoliv kontroly. Jednoduše by uživatelé vytvářeli nové bloky, které by pak sdíleli s ostatními uživateli a kolektivně by je přidávali do blockchainu. Toto řešení je však nedostatečné z jednoho hlavního důvodu - takto vytvořený blockchain neobsahuje důkaz o tom, že je validní. [22, kap. 4.1.3]

Například by mohlo dojít k tomu, že dva různí uživatelé vytvoří nové bloky, zařadí si je do své kopie blockchainu a ve stejnou dobu je pošlou všem ostatním. Na poslední blok blockchainu tak najednou ukazují dva nové, a tím pádem došlo k jeho rozvětvení. Neexistuje způsob, jakým určit, která větev je validní. Různí uživatelé mohou dostat oba bloky v různém pořadí, takže se nelze spolehnout jen na čas. Mezi uživateli chybí konsenzus na tom, které bloky do blockchainu patří a které ne. Způsobem, jakým tento konsenzus najít, jsou důkazy. [22, kap. 4.1.3]

2.2 Důkazy a těžení

Vytváření nových bloků je přenecháno speciálním uživatelům - těžařům, kteří do bloků přidají nějaký důkaz o tom, že jim lze věřit, a tím pádem i jimi vytvořenému bloku. Ostatní uživatelé pak na základě důkazů posuzují, které bloky do blockchainu patří a které ne. Tím pádem pak může nastat shoda. Vytváření tohoto důkazu a nových bloků tímto způsobem se říká těžení. Těžařem se může stát jakýkoliv uživatel, a proto těžení nesmí být triviální proces, aby se zvýšila hodnota důkazu. Důkaz tedy označuje, že těžař vynaložil určité úsilí k tomu, aby vytvořil nový blok. [20, kap. 2.1.4]

Dalším aspektem těžení jsou odměny. Jelikož těžení může být náročný proces (například na výkon či čas), je potřeba vytvořit pobídku ve formě odměn za vytěžení nového bloku. V případě kryptoměn je touto odměnou určité množství měny, která je těžaři připsána. Tím je také do oběhu kontinuálně přidávána nová měna bez přítomnosti centrální autority.

Uživatelé pak za správný blockchain považují ten nejdelší, protože obsahuje nejvíce důkazů o jeho validitě. Tím je bez centrální autority možné dosáhnout konsenzu.

2.2.1 Typy důkazů

Typů důkazu existuje několik, různé implementace blockchainu používají různé důkazy. Hlavním účelem je prokázat, že tvůrce bloku je důvěryhodný.

2.2.2 Proof of Authority

Nejjednodušším typem důkazu je Proof of Authority (PoA), neboli důkaz autoritou. V těchto systémech jsou bloky vytvářeny a validovány jen uživateli, kteří k tomu mají autorizaci. Tu mohou získat například členstvím v nějaké důvěryhodné organizaci nebo pověřením nějakou ještě vyšší autoritou. Již z podstaty se tento typ důkazu spíše používá u centralizovaných či soukromých blockchainů.

Důkaz je vytvořen například elektronickým podpisem celého bloku. Výhodou je zde absence složitějších výpočtů. Na druhou stranu je potřeba dobrého zabezpečení ze strany autorizovaných uživatelů. [23]

2.2.3 Proof of Stake

Proof of Stake (PoS), neboli důkaz vlastnictví či podílu, je dalším používaným typem důkazu. Za důvěryhodné jsou zde považováni těžaři, kteří sami vlastní nejvíce měny. Konkrétněji se uživatelé dočasně vzdají určitého množství své měny, čímž se stanou validátory. Periodicky je pak pomocí algoritmu vybrán náhodný validátor, který dostane určitý čas (řádově sekundy) na vytvoření jednoho bloku. Čím větší množství měny se validátor vzdá, tím má větší šanci, že je vybrán. Velkou výhodou je, že validátoři mají pobídku chránit integritu blockchainu, protože tím zároveň chrání svoje vlastnictví.

Proof of Stake zatím není tak rozšířený, ale jeho popularita roste vzhledem k nevýhodám spojeným s nejrozšířenějším Proof of Work se k němu uchylují další a další platformy. [20, kap. 2.1.4]

2.2.4 Proof of Work

Asi nejznámějším typem důkazu je Proof of Work (PoW), neboli důkaz prací. Princip spočívá v tom, že důvěra v těžaře je podmíněna nějakou prací vynaloženou pro vytvoření nějakého bloku. V reálné implementaci se tak jako důkaz používá výsledek nějaké velmi obtížné matematické operace, který je přidán do bloku. [22, kap. 4.1.3]

Touto operací je ve většině implementací hashování. Těžaři vezmou blok a k němu přidají náhodné číslo označované jako nonce. Následně vypočítají hash celého bloku. Pointou je najít takové číslo nonce, které způsobí, že hash bloku splňuje určité podmínky – například začíná několika nulami. [22, kap. 4.1.3]

Z povahy hashovacích funkcí vyplývá, že jediný způsob, jak zjistit který nonce vyústí v hash splňující podmínky, je vyzkoušet vypočítání hashe. Těžaři tedy musí znovu a znovu hashovat blok s různými hodnotami čísla nonce, dokud hash neodpovídá podmínkám. Pokud takové číslo najdou, přidají jej do nově vytvořeného bloku, pošlou ho ostatním a za odměnu si připíšou určité množství měny. [22, kap. 4.1.3]

Jelikož zároveň může jeden blok zkoušet těžit (tedy zkoušet různé nonce a počítat hashe) ohromné množství těžařů, dá se těžení přirovnat k loterii, kde se těžaři předbíhají v tom, kdo jako první nalezne odpovídající hash. U loterií platí, že čím více losů si účastník koupí, tím větší pravděpodobnost výhry má, u těžení je princip podobný – čím rychleji dokáže těžař počítat hashe, tím větší má pravděpodobnost, že najde ten správný. V závislosti na zvolených podmínkách může být nalezení takového hashe extrémně náročné, a aby těžař měl reálnou šanci nějaký najít, potřebuje co největší výpočetní výkon. [22, kap. 4.1.3]

Výhodami PoW jsou vyšší odměny oproti PoA a PoS, kde těžaři většinou vydělávají pouze na poplatcích za transakce. Ale i když PoW používají i velmi známé kryptoměny jako Bitcoin či Ethereum, má tento systém velké nevýhody. PoW je zranitelný vůči situaci, kdy jeden těžař ovládá více jak 51 % veškerého výpočetního výkonu. Jelikož se za důvěryhodný považuje ten nejdelší blockchain, může těžař vytvořit bloky s naprosto fiktivními daty, a protože má více výkonu než ostatní, bude jím vytvořený blockchain ten nejdelší.

PoW také vytváří zpoždění mezi provedením a potvrzením transakce. Blok se stává důvěryhodným až poté, co se za něj zařadí několik dalších důvěryhodných bloků.

Ale asi největší a nejdiskutovanější nevýhodou Proof of Work jsou jeho požadavky na elektrickou energii. Čím více je v systému těžařů, tím větší výkon každý z nich potřebuje, aby měl šanci vytěžit nějaký blok. Výsledkem je, že na těžení nejpoužívanějších kryptoměn, jako je Bitcoin, je používáno více a více výkonu, a tím pádem roste i celková spotřeba elektrické energie.

Podle některých odhadů byla spotřeba energie pro těžení Bitcoinu v lednu 2021 zhruba 90 TWh za rok. V prosinci je tato hodnota více než dvojnásobná – zhruba 200 TWh za rok. [24] Pro lepší představu, těžení Bitcoinu tímto tempem by za rok spotřebovalo více energie než Polsko a více než dvakrát více než Česko.[24] I z ekologických důvodů se tím pádem z těžení stalo velmi kontroverzní téma. Některé kryptoměny tak začaly přecházet na udržitelnější důkazy – například Ethereum ve verzi Ethereum 2.0 přejde na Proof of Stake. [20, kap. 2.1.4]

2.3 Využití technologie blockchain

Blockchain jako takový je velmi flexibilní technologií a v principu se jedná pouze o datovou strukturu, takže teoreticky jej lze využít všude, kde je potřeba pracovat s daty. Ve velkém se však zatím používá jen v některých odvětvích.

2.3.1 Kryptoměny

Blockchain je v dnešní době asi nejvíce spojován s kryptoměnami. První kryptoměnou je Bitcoin (zkratka BTC), který vznikl v roce 2009 a během další dekády získával větší a větší popularitu a dodnes se jedná o nejrozšířenější kryptoměnu. Jeho hodnota se postupem času vyšplhala až do desítek tisíc dolarů za jeden BTC. Na konci května 2022 se jeho cena pohybovala kolem 30 000 dolarů. [25] Další významnou kryptoměnou je například Ethereum.

Výhodou kryptoměn oproti klasickým měnám je především decentralizace a relativní soukromí. Kryptoměny jsou také zatím víceméně neregulované. Mezi nevýhody může patřit velmi nestálá cena, malé rozšíření mimo online prostor a nejasná budoucnost.

2.3.2 Chytré kontrakty

Jak již název napovídá, chytré kontrakty jsou obdobou klasických kontraktů. V principu se jedná o software, který obsahuje smluvní podmínky a na jejich základě například provede určité transakce. Tento software je v podobě zkompilovaného kódu přidán do blockchainu v podobě transakce. Během těžení a validace bloku obsahující tuto transakci je kód spuštěn a kontrakt je inicializován. Uživatelé s ním dále interagují skrze další transakce. Kontrakt pak podle smluvních podmínek reaguje a simuluje tak klasický kontrakt. [22, kap. 5.1.2]

Chytré kontrakty se vkládají i do blockchainů klasických kryptoměn, pokud jsou podporovány. Například u kryptoměny Ethereum je k tomuto používán virtuální stroj Ethereum Virtual Machine. [22, kap. 4.1.3]

2.3.3 NFT

Poměrně novým fenoménem jsou NFT (Non-fungible Token), neboli nezaměnitelné tokeny. V podstatě se jedná se o doklad o vlastnictví nějakých dat. Nejčastěji jsou NFT spojovány s digitálním uměním a princip je poměrně jednoduchý. Prodává se zde doklad o tom, že novým vlastníkem je kupující. K tomu jsou využívány veřejné blockchainy, do kterých jsou tyto doklady zapsány. [26]

Problémem digitálního umění a obecně digitálních dat je, že jsou z podstaty snadno replikovatelná. Kupující tak v podstatě nekupuje nic víc než záznam v blockchainu, který říká, že nyní vlastní nějaká data - token zvaný NFT. Data samotná se však internetem mohou šířit dál. Nákup NFT totiž nemusí znamenat převedení vlastnických práv na nového vlastníka, jedná se primárně o záznam v blockchainu. I proto kolem NFT panuje určitá skepse. Přesto se prodej NFT stal nesmírně výdělečný a kupující

jsou za tokeny ochotni dát tisíce dolarů. Například NFT virálního videa Nyan Cat bylo prodáno za 600 000 dolarů. [27]

2.3.4 Blockchain a archivace dat

Problém archivace dat je spojen s problémem online datových úložišť a služeb ke sdílení souborů. Příkladem takové služby v českém prostředí může být server Ulož.to. I v této oblasti si blockchain našel využití.

Konkrétněji je blockchain často využíván u peer-to-peer decentralizovaných služeb. Jedná se o služby pro sdílení souborů za absence centrálních serverů.

Příkladem může být Filecoin. Uživatelé zde prodávají nevyužité místo na vlastních discích. Jiní uživatelé pak mohou toto místo využít ke svým účelům. Za využívané místo platí kryptoměnou zvanou filecoin (zkráceně FIL). Data jsou v systému rozeseta k většímu množství uživatelů, čímž je chráněn jejich obsah. Když chce uživatel data znovu vidět, jsou složena zpět dohromady. Blockchain Filecoinu obsahuje údaje o transakcích. [28]

Filecoin samotný nenabízí službu, pouze infrastrukturu pro sdílení místa. Samotné služby tedy řeší aplikace existující nad ním. Jedná se například o Chainsafe, který zajišťuje end-to-end šifrování, nebo Estuary, který funguje jako aplikace v prohlížeči.

Filecoin využívá i služba Starling, která je primárně určená pro archivní účely – dává tedy větší důraz na integritu ukládaných dat. Dokumenty jsou ukládány v několika různých kopiích a průběžně je kontrolována jejich integrita. Bezpečnost je zajištěna tím, že samotnou distribuci dat řeší Filecoin, který je považován za bezpečný. [28]

TrustChain

Archivací dat se zabývala i práce „A blockchain approach to digital archiving: digital signature certification chain preservation“. [29] Konkrétně se zabývala problémem dočasné platnosti digitálních podpisů a navrhla způsob jak tento problém řešit pomocí blockchainu.

Moderní digitální podpisy často využívají certifikáty s časovými razítky, které mají jen omezenou časovou platnost - většinou mezi 2 a 5 roky. Takto podepsané dokumenty mohou být následně archivovány, avšak archivace má za úkol uchovávat data v mnohem delších časových horizontech, než je platnost těchto podpisů. Pokud podpisy slouží k udržení integrity dokumentů, je potřeba, aby byly dál validní. Nabízí se tak dvě možnosti - kontinuální podepisování nebo ukládání podpisů v metadatech.

Hlavní problém kontinuálního podepisování si lze jednoduše představit u archivu, který obsahuje 10 000 podepsaných dokumentů, přičemž každému z nich vyprší

platnost podpisu v různou dobu. Management takového systému je sice možný pomocí automatizace, avšak není to příliš elegantní řešení.

Ukládání podpisů v metadatech naráží na to, že podpisům vyprší platnost, a tím pádem integrita dokumentu začne záviset pouze na archivačním systému. Není to tedy úplně ideální řešení.

Autoři práce tak navrhují systém, který problémy spojené s podpisy a časovými razítky řeší pomocí blockchainu. Tento systém nazývají TrustChain, v práci popisují jeho druhou iteraci. TrustChain ukládá certifikáty do blockchainové struktury s tím, že jej zajímají pouze certifikáty, s daty samotnými nijak neoperuje.

Při přidávání nových certifikátů TrustChain nejdříve zkontroluje, zda se již v blockchainu nenachází a zda je validní. Jeden blok obsahuje několik certifkátů. Dále je v bloku prostor pro informace o revokaci existujících certifikátů. Konsenzu se u TrustChainu dosahuje hlasováním - každý blok je předáván mezi uzly, které hlasují zda je validní nebo ne, a pokud většina souhlasí, je přidán do blockchainu. [29]

BCLinked Web Archiving System

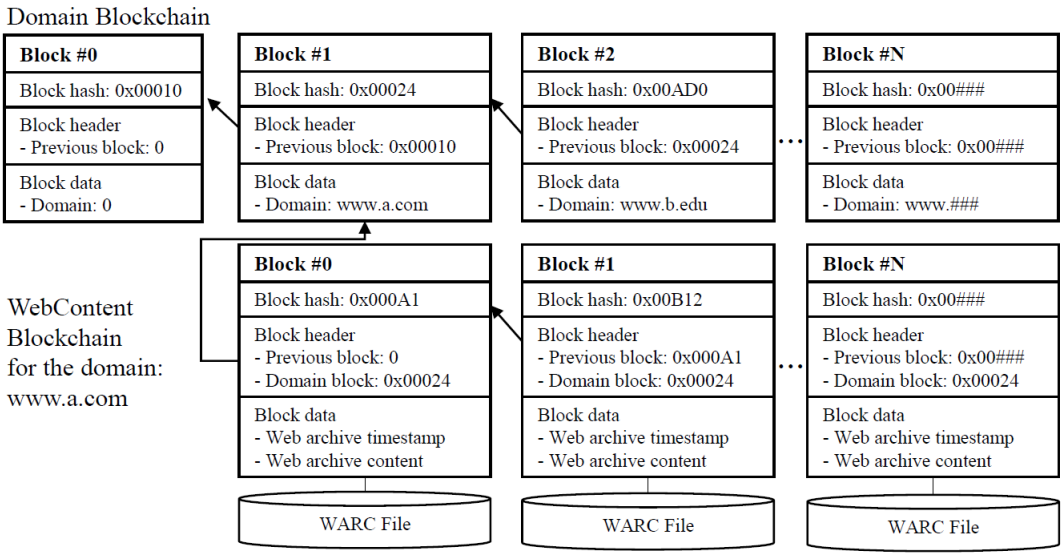
Práce „Design of an Enhanced Web Archiving System for Preserving Content Integrity with Blockchain“ se zase zabývala využitím blockchainu ve webových archivech. [30] Webové archivy jsou specializované digitální archivy, které uchovávají webový obsah - např. webové stránky. Webové stránky jsou velmi často archivovány periodicky, aby bylo vidět, jak se vyvíjely v čase. Toto je důležité kvůli tomu, jak triviální je obsah stránek upravovat. Nejznámějším webovým archivem je asi Internet Archive, který uživatelům umožňuje prohlížet archivované verze stránek.

Pro archivaci webových stránek existuje standardizovaný formát WARC. Soubory tohoto typu obsahují metadata spojená s danou žádostí a odpovědí na stáhnutí určité webové stránky. Dále jsou obsažena samotná data. Podobně jako u normálních digitálních archivů, i zde existuje potřeba zachovat integritu těchto dat. Práce navrhuje systém, nazvaný BCLinked (Blockchain Linked) Web Archiving System, který archivované stránky ukládá do blockchainové struktury.

Nejdříve je vytvořen blockchain, který obsahuje názvy jednotlivých domén - tento blockchain autoři nazývají doménový. Když je pak potřeba přidat nový archivní záznam, není blok přidán na konec blockchainu, ale odkazuje se na blok obsahující název domény, ke které stránka patří. Tento blok odkazuje informace o dané archivované instanci stránky a odkazuje se na WARC soubor s archivovaným obsahem. Integritu zajišťuje přítomnost hashe tohoto souboru. Pokud je stránka archivována znovu, je vytvořen další blok, který se zařadí do toho sub-blockchainu.

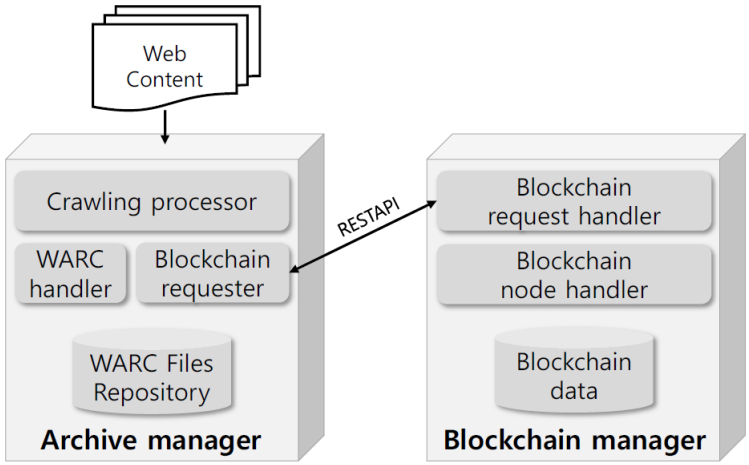
Navržený systém lze vidět na obrázku 2.2.

BCLinked je rozdělen do dvou částí - Archive manager a Blockchain manager.



Obr. 2.2: Blockchain archivu BCLinked [30]

Archive manager stahuje webové stránky, zpracovává je do WARC souborů a spravuje repositář archivovaného obsahu. Blockchain manager spravuje samotný blockchain. Rozdělení lze vidět na obrázku 2.3



Obr. 2.3: Struktura archivu BCLinked [30]

Blockchain je zde plně privátní a není distribuovaný. To znamená, že není třeba řešit důkazy a konsenzus, protože jediný kdo do blockchainu dokáže zasáhnout je jeho vlastník.

3 Návrh archivu využívajícího blockchain

Prvním praktickým výstupem práce je návrh archivu využívajícího blockchain. Nejdříve bude služba navržena obecněji a následně budou představeny konkrétnější řešení některých problémů archivace, tak jak byly nastíněny v předchozích kapitolách.

3.1 Obecný návrh

Z širšího hlediska se bude jednat o databázovou aplikaci zaměřenou na archivaci dat. Z pohledu uživatele tak nebude příliš odlišná od již existujících řešení. Jedním z hlavních cílů bude co největší bezpečnost, a proto bude aplikace cílená i na archivy uchovávající citlivá data.

Typickým případem užití tak mohou být firmy potřebující archivovat staré finanční záznamy. Nebo se může jednat o historické archivy, kde některé archiválie mohou být tajné. Stejně tak bude aplikace potenciálně využitelná i pro orgány typu policie.

Blockchain bude implementován v podobě externí aplikace, která bude kontrolovat a validovat integritu archivu. Tato aplikace bude přístupná pouze pro správce archivu. Diagram obecného návrhu lze vidět na obrázku 3.1

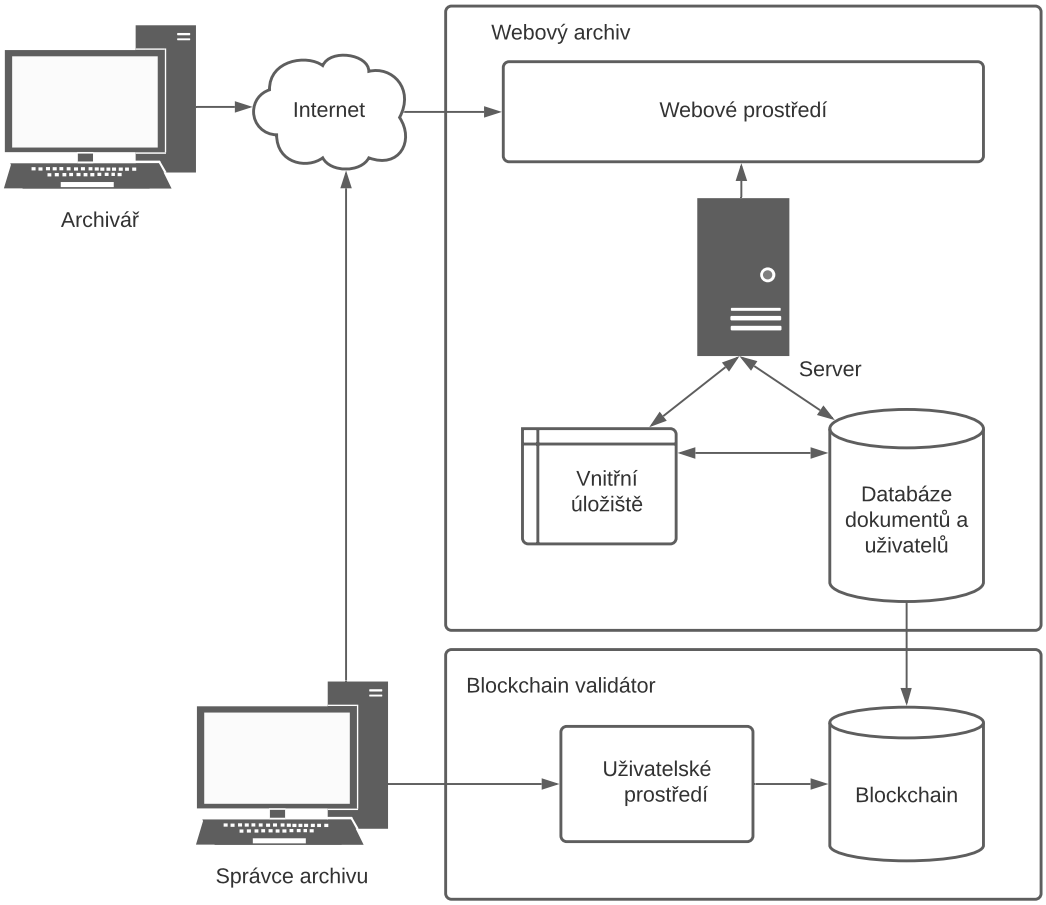
3.2 Základní funkcionalita

Předně bude zapotřebí implementovat veškeré základní funkce digitálního archivu. Samozřejmostí je možnost dlouhodobého ukládání dat v podobě dokumentů. K tomuto poslouží databáze.

Zde se naskýtá možnost centralizované databáze nebo distribuované databáze. Vzhledem k tomu, že jedním z případů užití může u archivů být i archivace tajných či citlivých dokumentů, se jeví centralizovaná databáze jako lepší možnost. Data tak budou uchovávána na jednom centrálním místě, které bude plně pod kontrolou správců archivu.

Celá aplikace bude mít uživatelsky přívětivé rozhraní, bude přístupná skrze prohlížeč. Administrace pak bude mít větší možnosti interakce s databází a dokumenty, jelikož bude mít přístup k samotnému serveru. Archiv bude nabízet základní procházení archivovaných dokumentů. Záviset bude na typu souboru, nastavení ze strany administrace a autorizace uživatele.

Obsahem dokumentů mohou být soubory, s tím, že správci budou mít možnost omezit obsah jen na určité typy souborů. V základu však na typu nezáleží. Archivovat půjdou i internetové stránky, kde uživatelé pouze zadají jako obsah URL a aplikace



Obr. 3.1: Návrh webového archivu a blockchainu

ji automaticky stáhne a archivuje. Možností by mohlo být i periodická archivace stránky v určitém intervalu (například měsíčně).

V neposlední řadě bude archiv podporovat verzování dokumentů. Dokumenty mohou být postupně upravovány, včetně obsahu, čímž dochází k vytváření nových verzí. Starší verze v archivu budou zůstávat, protože odstraňování archivovaných dokumentů bude možné jen ve výjimečných případech. Ve výsledku tak budou uživatelé mít možnost zobrazovat všechny verze dokumentů a sledovat historii jejich úprav.

3.3 Bezpečnost archivu

Z bezpečnostního hlediska je nutné se zaměřit na tři oblasti - řízení přístupu, důvěrnost dat a integrita dat.

3.3.1 Řízení přístupu

Řízení přístupu bude řešeno na základě principu Role Based Access Control - tedy na základě přidělování rolí. Nejvyšší úroveň bude mít **systémový administrátor**, ten bude mít kontrolu nad všemi technickými aspekty, včetně přístupu k samotným uloženým datům a blockchainu. V ideálním případě bude tuto roli mít jeden uživatel. Další rolí bude **správce archivu**. Ten bude mít možnosti přímého přidávání dokumentů, bude moci potvrzovat požadavky nižších rolí a přidělovat ostatním uživatelům role. Těchto správců bude co nejmenší množství.

Další rolí budou **archiváři**. Tito uživatelé budou mít možnost práce s dokumenty, avšak většina jejich akcí bude muset být potvrzena správcem archivu. Například vyberou obsah, který je potřeba přidat do archivu, vytvoří pro něj dokument, vyplní metadata a poté pošlou správci požadavek na přidání do archivu. Správce ověří správnost údajů, přidá svůj elektronický podpis a přidá dokument do archivu a blockchainu. Stejný postup bude fungovat i při vytváření nové verze dokumentu.

Poslední rolí bude **běžný uživatel**. Ten bude mít práva pouze na základě úvahy správců. Může to být například možnost zobrazování či stahování určitých dokumentů.

Archiv bude podporovat vytváření nových uživatelů s tím, že vždy musí být potvrzeni správcem, který jim přidělí vhodná práva. Noví správci mohou být jmenováni, jen pokud se shodne většina existujících správců.

3.3.2 Důvěrnost dat a šifrování

Šifrování ukládaných dat nebude automaticky povinný prvek, ale správci budou mít možnost jej vyžadovat. Tak či tak, bude vždy existovat možnost jakýkoliv dokument zašifrovat. K tomu bude využita symetrická šifra AES v módu CBC. Politika hesel bude záležet na správcích. Při prohlížení daného souboru tak bude muset archivář či správce zadat správné heslo. Důvěrnost přenášených dat během nahrávání a stahování souborů do archivu bude zajištěna použitím protokolu TLS.

3.3.3 Integrita dat a blockchain

Integritu dat bude především zajišťovat technologie blockchain. Na rozdíl od databáze bude blockchain distribuován, k čemuž bude sloužit samostatná aplikace. Jako typ důkazu se bude používat Proof of Authority za použití elektronických podpisů. Autoritu přidávat nové bloky budou mít správci archivu. Kopii blockchainu budou uchovávat všichni správci, systémový administrátor a samotný server.

Přidání nového bloku tak bude fungovat následovně.

- 1. Archivář vytvoří nový dokument - přidá obsah a metadata
- 2. Archivář pošle dokument k validaci
- 3. Správce potvrdí správnost údajů, popřípadě je opraví
- 4. Správce vypočítá hash souboru a elektronicky jej podepíše
- 5. Správce vytvoří nový blok a rozešle jej ostatním správcům
- 6. Ostatní správci ověří platnost bloku a přidají jej do blockchainu

Ke správě blockchainu budou tito uživatelé používat speciální aplikaci, která bude mít možnost komunikace s archivem. Tato aplikace bude provádět i validace blockchainu. Ty pak mohou probíhat automaticky v periodických časových intervalech nebo na vyžádání správce. Probíhá následovně:

- 1. Validátor načte blockchain a po blocích jej začne procházet
- 2. V každém bloku najde hash souboru a odkaz na daný soubor
- 3. Validátor přepočítá hash uloženého souboru a porovná jej s hashem v bloku
- 4. Validátor ověří elektronický podpis (včetně platnosti klíče)
- 5. Validátor ověří, že souhlasí hash předchozího bloku s údajem v následujícím bloku

Pokud jsou všechny bloky validní, je validní celý blockchain a integrita tak nebyla porušena. Pokud validace jakéhokoliv bloku selže, je systém považován za porušený. Detailní výpis validace je rozeslán správcům a systémovému administrátorovi, který bude mít za úkol incident vyšetřit.

Specifickým problémem je pak odstraňování existujících dokumentů a oprava dokumentů, jejichž integrita byla porušena. Blockchain se může zdát jako velmi rigidní strukturou, protože jedna z jeho point je zajistit neměnnost uložených informací. Avšak jeden z principů, na kterých je blockchain postavený je konsenzus mezi uživateli.

To znamená, že k úpravám již zaznamenaného bloku stačí, aby se polovina správců rozhodla, že jej určitým způsobem změní a nový blockchain se tak stane legitimním. K tomuto by však mělo docházet jen ve velmi vzácných případech - například pokud se do archivu nedopatřením dostala citlivá data a je potřeba je vymazat. V tom případě by pak proces vymazání probíhal zhruba následovně:

- 1. Některý z uživatelů vznese požadavek na odstranění některého z dokumentů
- 2. Správci tuto žádost buď potvrdí nebo zamítnou
- 3. Jeden ze správců vytvoří nový speciální blok, který pouze informuje o tom, že nahrazuje již odstraněný soubor - při validaci je tento blok přeskočen
- 4. Správce přepočítá hashe všech následujících bloků (hashe souborů zůstávají nezměněné), čímž vytvoří nový validní blockchain
- 5. Ostatní správci jej ověří a přijmou jako hlavní blockchain systému
- 6. Nové bloky se od této chvíle přidávají do upraveného blockchainu

V případě, že je dokument pouze upravován, stačí přepočítat hash souboru, znovu

jej podepsat a následně přepočítat hashe následujících bloků. Jakékoliv změny by měly být zaznamenány a archivovány s jasným zdůvodněním, proč k nim došlo.

3.4 Hardware a software

Bude se jednat o webovou aplikaci s rozsáhlou databází, čemuž musí odpovídat i hardwarové a softwarové nároky. Nejkritičtějším hardwarovým omezením je volné místo. Zde se budou různé implementace lišit v závislosti na počtu dokumentů a jejich typu. Nelze proto tedy určit ideální velikost disku.

Aplikace by neměla být příliš výkonově náročná, ale opět bude silně záležet na velikosti databáze a počtu dokumentů. Větší archivy budou potřebovat výkonnější servery.

Z hlediska softwaru je potřeba vybrat operační systém, server a databázi.

3.4.1 Operační systém

Archiv nebude mít speciální požadavky na operační systém. Proto si stačí vybrat z operačních systémů, které jsou často používané pro vývoj a hosting webových aplikací. Z tohoto pohledu se jeví jako nejideálnější linuxové distribuce jako například CentOS nebo Ubuntu. [31]

3.4.2 Server

I v případě serveru dává smysl výběr ozkoušeného softwaru, jako je například Apache Tomcat. Jedná se o volně dostupný, open-source server s širokým využitím.

3.4.3 Databáze

U databáze může být výběr o něco náročnější. Klasickými řešeními jsou databáze typu MySQL nebo PostgreSQL. Pro potřeby archivu se však vybízí využití databáze MongoDB, která místo klasických záznamů používá dokumenty.

4 Implementace archivu využívajícího blockchain

Hlavním praktickým výstupem práce je implementace popsaného návrhu. Implementovány budou všechny důležité aspekty, kde největší důraz je kladen na blockchain a integritu dat. Nejdřív budou vybrány vhodné nástroje, následně bude popsána funkčnost systému, budou probrány technické aspekty a principy implementace. V neposlední řadě dojde k otestování funkčnosti a finálnímu zhodnocení.

4.1 Návrh aplikace

Nejdříve je potřeba si logicky rozvrhnout jednotlivé součásti celého systému. Vzhledem k tomu, že některé části mají formu webového prostředí, zatímco jiné jsou spíše lokálního charakteru, nebude se jednat o jedinou aplikaci, ale několik, navzájem interagujících aplikací. S ohledem na praktičnost implementace a průběh vývoje tak ve výsledku existují tři hlavní prvky:

- Webová aplikace - tvořená front endem a back endem
- Archivační aplikace - bude běžet na serveru
- Validační aplikace - bude běžet lokálně u uživatelů

Systém lze vidět i na diagrammu 4.1

Cestu jednoho dokumentu skrze systém pak lze vidět na diagramu 4.2

Z tohoto rozvržení začínají vyplývat určité požadavky na vybrané nástroje. Část aplikace má webový charakter, část formu API, část formu desktopové aplikace. Lze tedy přejít k samotnému výběru.

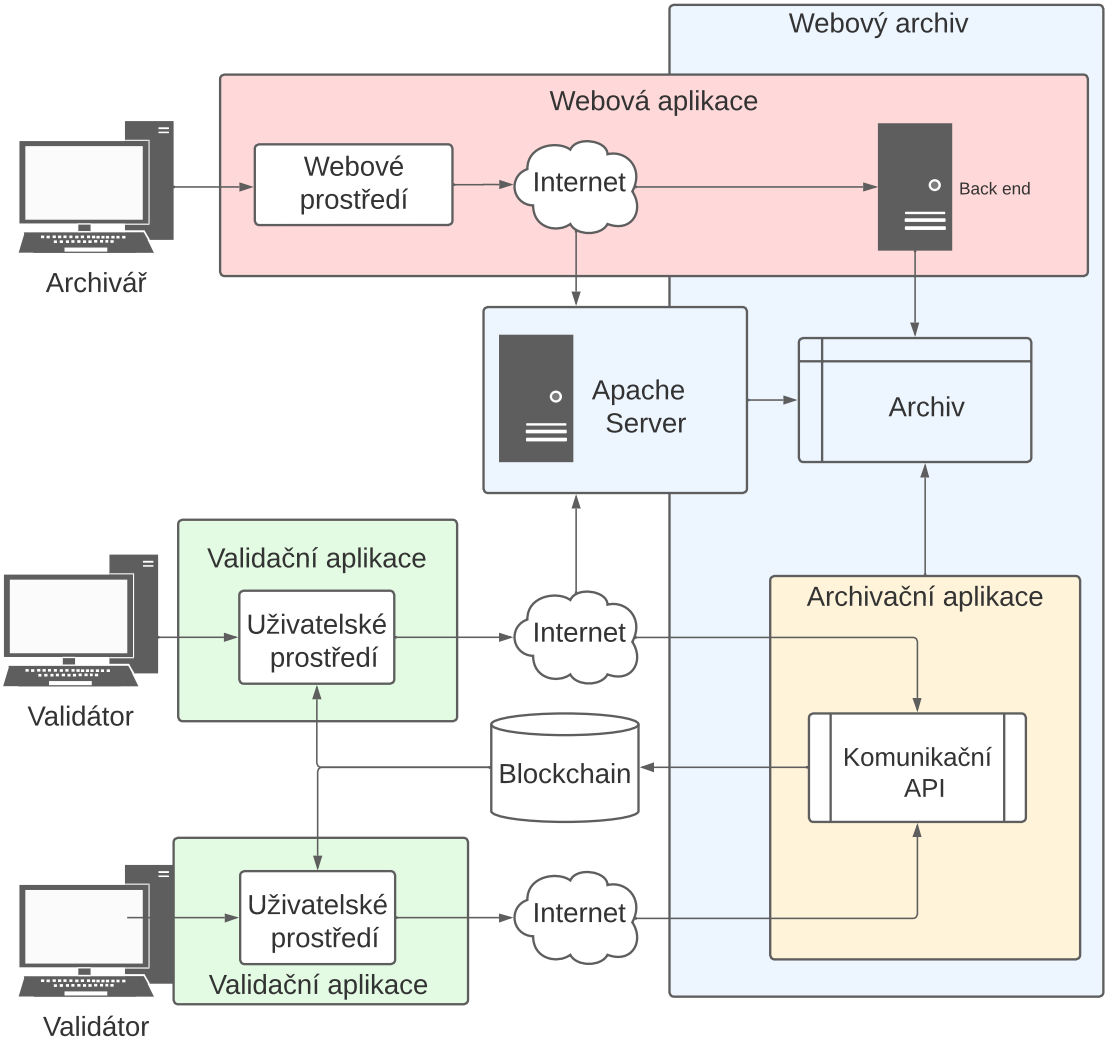
4.1.1 Výběr nástrojů

Především jde o výběr vhodného programovacího jazyka, vývojového prostředí a testovacího prostředí.

Programovací jazyky

Programovací jazyk ve kterém bude systém naprogramován by měl splňovat následující podmínky:

- Podpora desktopových aplikací
- Podpora webových aplikací
- Implementace GUI
- Schopnost komunikace
- Zkušenosti s vývojem v tomto jazyce

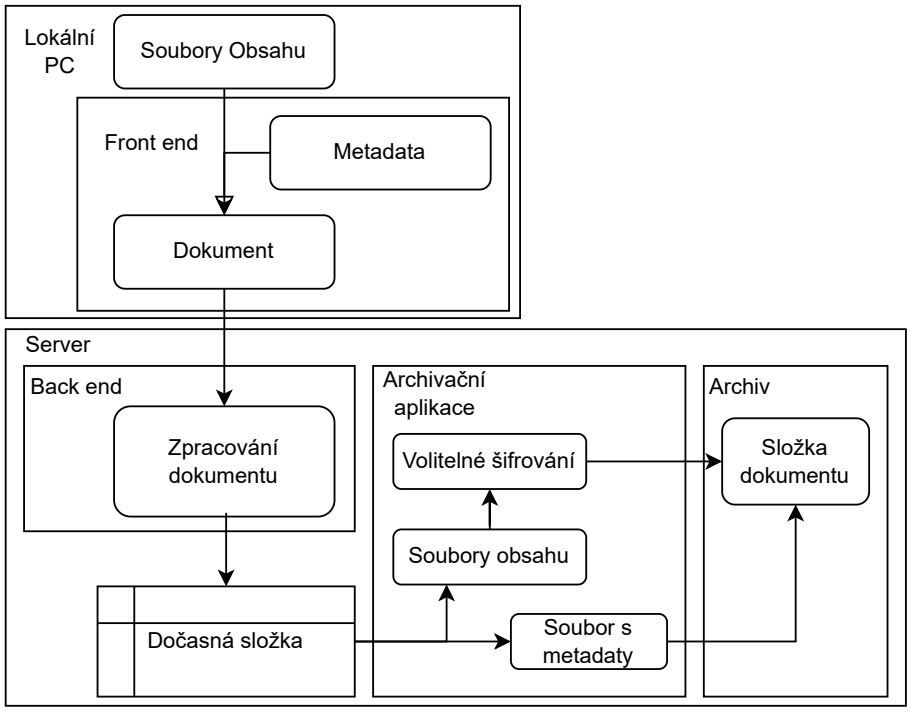


Obr. 4.1: Implementace webového archivu a blockchainu

Nejlepší řešení je použít jazyk JavaScript pro webovou část a jazyk Java pro desktopovou část systému. Jedná se o hojně využívané programovací jazyky, které nabízí veškeré potřebné vlastnosti.

JavaScript je skriptovací a interpretovaný jazyk, který původně sloužil pro vytváření dynamických webových stránek. V posledních letech však nachází využití i na straně serveru a stal se tak naprosto dominantním jazykem v oblasti webových aplikací.

JavaScript je málokdy používán ve své základní verzi, místo toho jsou používány různé aplikační rámce - tzv. frameworky. Mezi nejpopulárnější patří jQuery, AngularJS, React nebo Vue.JS. Některé frameworky jsou určené spíše pro front end, zatímco jiné spíše pro back end aplikací. Často se tak během implementací kombinují. Pro implementaci této aplikace byl zvolen Vue.JS pro jeho relativní jednoduchost při vytváření front endů. Pro back end je pak použit framework Express.



Obr. 4.2: Diagram uploadu dokumentu

Java je široce používaný, objektově orientovaný programovací jazyk. I když se v dnešní době používá hlavně v serverových aplikacích, dlouhou dobu byl používán i pro vývoj desktopových aplikací. Java je také nejlepší variantou z hlediska zkušeností. Pro vývoj a testování je použita verze JDK 15.

Vývojové a testovací prostředí

Dále je potřeba vybrat vývojové prostředí (zkráceně IDE – Integrated Development Environment). Většina IDE podporuje více jazyků, avšak jsou často zaměřené právě na jeden. Ve výsledku však téměř vždy záleží spíše na preferencích vývojáře.

Pro vývoj v JavaScriptu se nejvíce používají IDE Visual Studio Code (také VSCode), React či Atom. Pro vývoj této aplikace byl zvolen VSCode.

Mezi nejpoužívanější IDE pro Javu patří Eclipse, NetBeans a IntelliJ. Mezi těmito prostředími nejsou příliš velké rozdíly a záleží tak spíše na preferencích vývojáře. Pro vývoj této aplikace je používáno prostředí IntelliJ.

V neposlední řadě je potřeba vybrat knihovny a nástavby, které bude program využívat. Jak již bylo zmíněno, pro JavaScript je to framework Vue.JS, který umožňuje vytváření jednotlivých komponent. Konkrétně pak jde o verzi Vue 3. Pro potřeby komunikace je využita knihovna Axios. Jako CSS šablonu je použita volně dostupná, open source šablona Bulma. [32]

V rámci Java aplikace nejsou využity žádné nestandardní knihovny. I blockchain samotný je naprogramován pouze za použití standardních knihoven. Jako JDK byl použit Java SE Development Kit 15.0.2. Jedinou větší volbou je pak výběr knihovny pro vytváření GUI. Dvě hlavní knihovny sloužící tomuto účelu pro Javu jsou Swing a JavaFX. Jelikož je GUI aplikace poměrně jednoduché, na výběru až tak nezáleží. Pro potřeby této aplikaci tak byla vybrána knihovna Swing.

Celý vývoj a testování jsou provedeny na desktopovém počítači s operačním systémem Windows 10.

Po výběru veškerých nástrojů je možné přejít k implementaci. V následujících částech budou popsány jednotlivé komponenty, jejich funkčnost, principy fungování a výsledky.

4.2 Webová aplikace

Webová aplikace je asi nejviditelnější součástí systému, proto bude nejprve popsána ona. Z pohledu uživatele jde o rozhraní, přes které mohou nahrávat a následně prohlížet archiválie. Z hlediska implementace ji lze rozdělit na front end a back end.

4.2.1 Front end

Front end je ta část aplikace, která běží přímo v uživatelově prohlížeči, jeho hlavní funkcí tedy je prezentovat uživateli informace a reagovat na jeho požadavky. K přístupu k jednotlivým funkcím je vytvořeno grafické uživatelské prostředí (GUI - Graphical User Interface), pomocí kterého můžou uživatelé navigovat v rámci stránky. Základním prvkem je navigační panel, přes který lze přecházet k jednotlivým komponentám. Vzhled bude mimo jiné demonstrován v kapitole 4.7 Demonstrace funkčnosti aplikace.

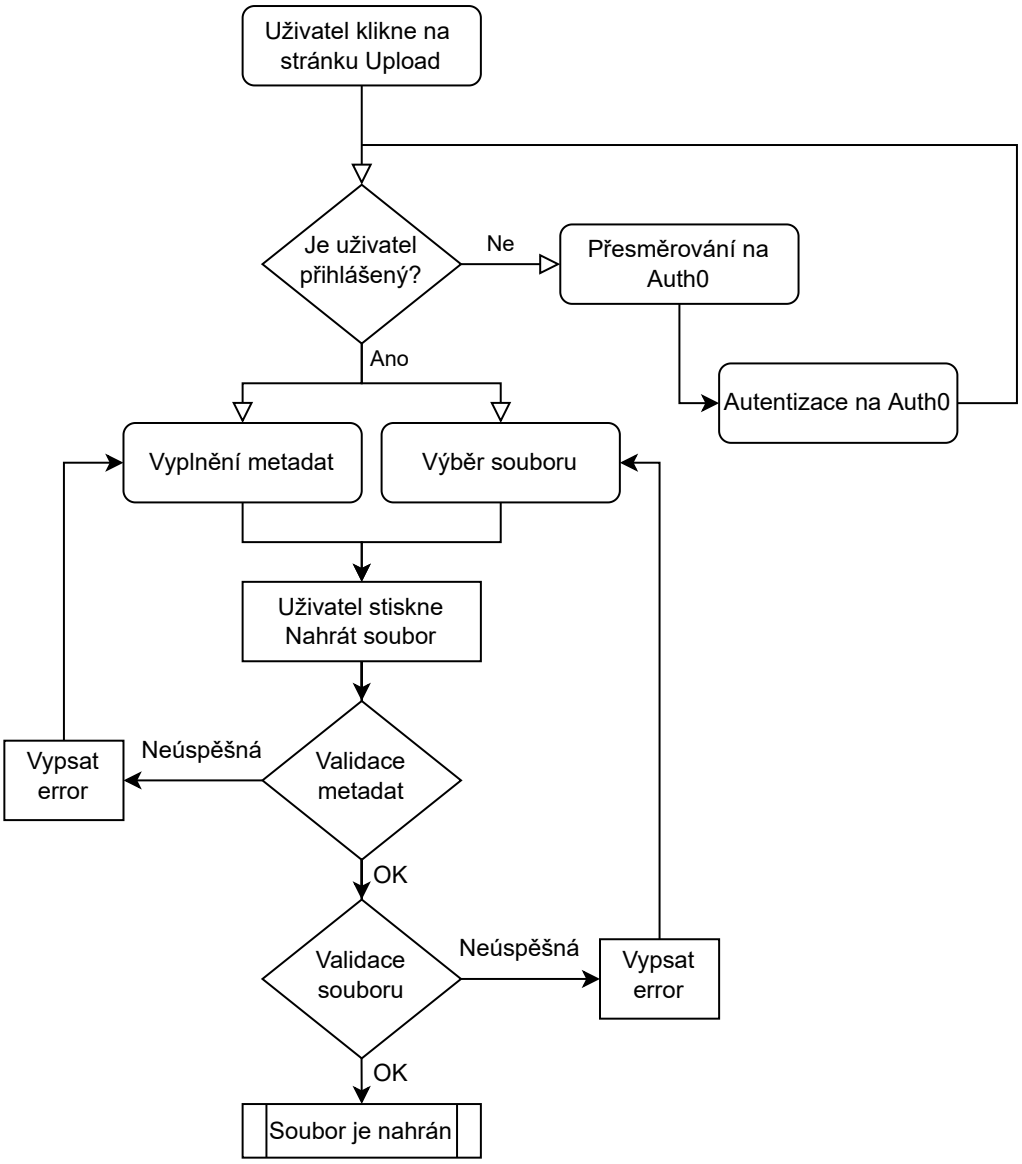
Jelikož se jedná o velmi základní funkcionalitu, byla k implementaci využita šablona, která zároveň implementuje autentizaci. [33] Byly provedeny jenom menší úpravy vzhledu a obsahu.

Z návrhu vyplývají tři hlavní funkce, které by měl front end umožňovat:

- Nahrávání dokumentů s metadaty
- Procházení archivovaných dat
- Zajistit autentizaci uživatelů

Nahrávání dokumentů

Tuto funkci zajišťuje v kódu Vue komponenta MultiUpload. Opět se jedná o přejatý kód [34], ovšem výrazně upravený pro potřeby aplikace. Modularita frameworku Vue umožňuje poměrně jednoduché přidávání nových komponent. Ty stačí definovat



Obr. 4.3: Front end část uploadu souborů

a poté se na ně pouze jednoduše odkazovat. Přibližný diagram této funkce lze vidět na obrázku 4.3.

Dokumentem se v rámci archivu rozumí kombinace metadat a obsahy souboru. Nahrávání dokumentů tedy probíhá skrze jednoduchý formulář. Uživatel zadá název dokumentu, vybere, zda chce uložená data šifrovat a popřípadě jakým heslem, a následně vybere samotné soubory obsahu. Dokument se tedy dá rozdělit následovně:

- Název dokumentu - zadán uživatelem
- Autor - uživatelské jméno, odesláno automaticky
- Šifrování - má hodnotu true, nebo false, zadáno uživatelem
- Heslo - heslo, pomocí kterého je dokument šifrován, volitelně zadáno uživatelem

- Obsah - libovolné množství souborů, které tvoří obsah dokumentu

Ve spodní části formuláře se objevují vybrané dokumenty, které uživatel může ještě před odesláním odebrat. Dále komponenta vypisuje hlášky, které jsou relevantní pro uživatele - především chybové hlášky validace a status nahrávání. Zobrazí také odpovědi přicházející z back endu.

Jak již bylo naznačeno, důležitým prvkem této komponenty je validace dokumentu. Archiv musí mít určitá omezení, co se týče ukládaných dat, a dokumenty tak musí splňovat určité podmínky. Proto dává smysl kontrolovat jejich splnění již na vstupu. Podmínky jsou následující:

- Formát souborů podle jeho MIME typu
- Soubor nesmí být větší než stanovené maximum
- Musí být vyplněn název dokumentu
- Název nesmí být delší než stanovené maximum
- Název nesmí obsahovat určité znaky
- Pokud je zaškrtnuto šifrování, nesmí chybět vyplněné heslo
- Dokument se zadaným názvem nesmí existovat

Některé validace obsahují údaje, které lze v případě potřeby administrátory měnit. Např. lze přidávat či odebírat povolené formáty, nebo zvětšit či zmenšit maximální velikost souboru. V základní implementaci jsou povoleny formáty jpeg, png, tiff, gif, bmp, pdf, docx, odt, txt, mpeg, mp4 a avi. Pro samotnou aplikaci na formátu nezáleží, nahrání a následovná manipulace není nijak omezená.

Jedná se tedy pouze o omezení k lepší správě archivu. Pokud by se například jednalo o archiv faktur, které jsou všechny ukládány ve formátu pdf, nemá smysl, aby se do archivu mohly dostat i jiné formáty.

Podobně je na tom i validace velikosti. V základní implementaci je maximální velikost jednoho souboru 5 MB. Opět záleží na dané implementaci, v archivu, který je určen pro uchování textových dokumentů, je toto dostačující velikost, pokud by se jednalo o archiv fotografií či videí, byla by velikost nedostatečná. Samozřejmě tato validace má význam i z hlediska požadovaného místa. Příliš velké soubory by také mohly způsobovat problémy se sítí a následným zpracováním.

Validace probíhají pro každý soubor v dokumentu zvlášť a uživatel je informován o výsledku pro každý soubor. Pokud některý z nich validací neprojde, vedle jeho názvu se objeví důvod a soubor ve výpisu zčervená. Uživatel jej následně může odstranit z výběru.

Samozřejmě všechny tyto validace mají spíše informační charakter pro uživatele, pro nečestného uživatele by nebyl problém je obejít. Stačilo by vyplnit legitimní dokument, před odesláním požadavek zastavit a upravit jej tak, aby obsahoval i nevalidní hodnoty. Proto na back endu všechny validace proběhnou znovu.

Procházení dokumentů

Původní návrh počítal s vytvářením databáze, avšak nakonec je archiv tvořen pouze složkou uloženou na serveru. Sice tím archiv ztrácí některé funkce, které by databáze umožnila, ale výrazně se tím zjednoduší celá implementace. Procházení archivu je tak ve výsledku umožněno pomocí vypsání obsahu této složky do prohlížeče.

Každý dokument je uložen ve vlastní složce pojmenované podle jeho názvu. V této složce se pak nachází soubory obsahu, které si ponechaly svůj původní název, a soubor „metadata.json“, který byl vygenerován Archivační aplikací. Tento soubor obsahuje všechna uložená metadata. V prohlížeči si jej lze zobrazit rozkliknutím.

Na jednotlivé obsahy lze kliknout a v závislosti na jejich formátu je lze rovnou prohlížet nebo jsou staženy. Toto závisí především na prohlížeči a jeho možnostech. Pokud je obsah zašifrován, prohlížeč je skoro jistě nebude schopen zobrazit a je tak automaticky stažen.

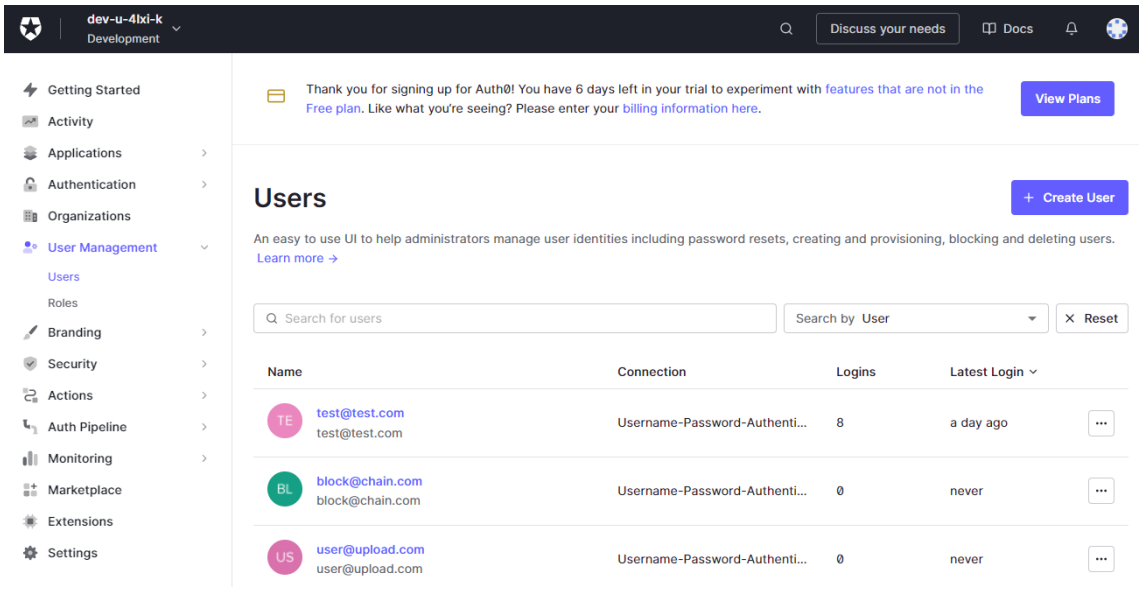
Zde vyvstal jeden z problémů a to je fakt, že šifrování probíhá v procesu nahrávání mnohem později a má je v gesci Archivační aplikace. Automatické dešifrování by tak bylo velmi složité implementovat. Z tohoto důvodu je prohlížení zašifrovaných souborů vyřešeno pomocí speciální Java aplikace Decryptor. Uživatel si stáhne zašifrovanou verzi souboru a uloží si ji na vlastní počítač. Dále spustí aplikaci Decryptor, kde zadá stáhnutý soubor, název dokumentu a heslo a spustí dešifrování. Pokud bylo heslo správné, aplikace vytvoří dešifrovanou verzi, kterou si již může klasicky zobrazit. Popis samotné aplikace Decryptor je v kapitole 4.5.

Autentizace a řízení přístupu

Uživatelé se do webového prostředí budou autentizovat pomocí uživatelského jména a hesla. Jak bylo popsáno v kapitole 1.3.2, elegantním řešením je využít autentizační platformu třetí strany. Konkrétněji je pak pro implementaci zvolena platforma Auth0.

Auth0 nabízí širokou škálu možností, i když pro implementaci budou stačit jen ty nejzákladnější. K administraci se správci archivu dostanou přihlášením na stránce „auth0.com“, správa autentizace je tedy kompletně oddělena od samotné aplikace. Administrace Auth0 má velmi uživatelsky přívětivé prostředí a správa uživatelů a řízení přístupu je tak relativně jednoduchá. Na obrázku 4.4 lze vidět správu uživatelských účtů na platformě.

Ve své podstatě Auth0 (a podobné autentizační platformy) funguje tak, že během autentizace je uživatel přesměrován na vygenerovanou stránku Auth0, kde se přihlásí svými údaji. Pomocí API pak front end a back end kontrolují, zda je daný uživatel opravdu autentizovaný. Toho je dosaženo pomocí přístupových tokenů (access token), které jsou předávány v rámci jednotlivých HTTP požadavků. Zjednodušeně - front end si vyžádá token od Auth0 API, přidá ho k HTTP požadavku a back end ověří



Obr. 4.4: Administrátorská stránka na platformě Auth0

zda je token platný. V administraci pak lze nastavit dobu, po kterou zůstává token validní.

4.2.2 Back end

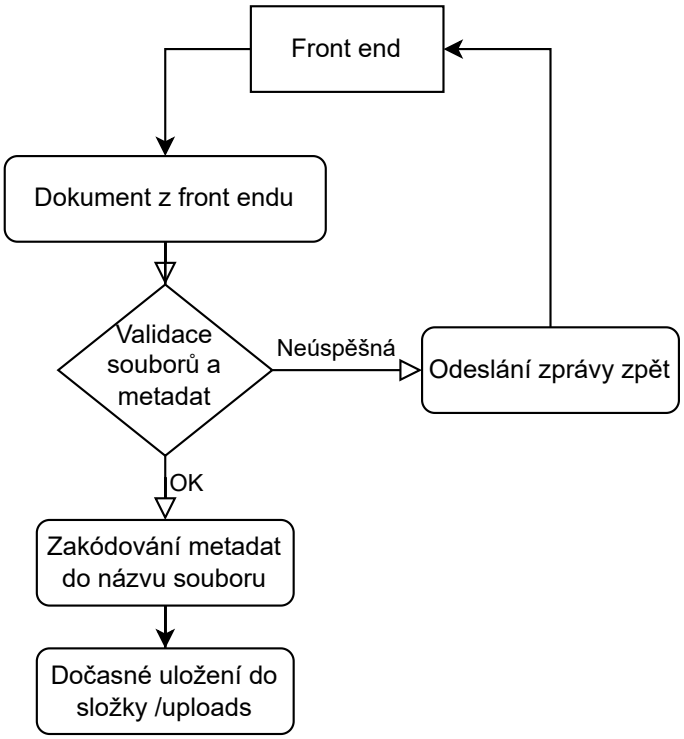
Back end je ta část aplikace, která běží na serveru. Většinou je jeho úloha zpracování požadavků předané front endem, kterých může být v závislosti na dané aplikaci široké spektrum. V této implementaci back end zajišťuje následující služby:

- Zpracování a uložení nahrávaných souborů
- Validace souborů
- Zpracování poslaných metadat

Komunikace mezi front endem a back endem probíhá pomocí HTTPS požadavků. Je tedy šifrována pomocí protokolu TLS v1.3 a nahrávané soubory tedy nelze získat odposloucháváním komunikace, čímž je zajištěna důvěrnost přenášených dat. Soubory jsou nahrávány pomocí metody POST s tím, že jako html kódování je použita metoda „multipart/form-data“. Požadavek sestává z metadat, uložených ve formátu JSON a samotných souborů obsahu.

Jak bylo zmíněno v kapitole 4.1.1, jako framework je použit Express.js. Back end není příliš složitý, ve své podstatě jde pouze o definice toho, co má server dělat s určitým HTTPS požadavkem. V tomto případě zpracovává pouze jediný, a to požadavek na upload souboru.

Back end tedy vezme tento požadavek a jeden po druhém začne zpracovávat soubory v něm obsažené. K tomu je využit middleware Multer určený právě ke



Obr. 4.5: Back end část uploadu dokumentu

zpracování přijatých souborů. Znovu proběhnou všechny validace, a pokud jsou soubory v pořádku, jsou uloženy.

Předem je vytvořena složka „uploads“, ve které je nově vytvořena podsložka nazvaná podle názvu dokumentu. Do ní jsou pak uloženy jednotlivé soubory. Metadata jsou zakódována do názvu těchto souborů pro další jednoduší zpracování. Název každého souboru pak tedy má následující formu:

název_dokumentu.autor.šifrování.původní_název_souboru.formát

V případě, že je hodnota šifrování kladná, pak je název následovný:

název_dokumentu.autor.šifrování.heslo.původní_název_souboru.formát

Všechny požadavky musí mít validní autentizační token, což back end vždy ověřuje. Diagram uploadu souborů lze vidět na obrázku 4.5.

Prohlížení archivu pomocí Apache serveru

Jedním z problémů během implementace bylo právě prohlížení archivu. Uživatelská část byla popsána v kapitole 4.2.1, zde bude popsán back end.

Asi nejlepším řešením by bylo implementovat prohlížení do back endu. Front end by poslal požadavek na výpis archivu, back end by pak zpracoval obsah archivu a v nějaké podobě jej poslal zpět. Uživateli by se pak například zobrazila tabulka,

kde by byly vypsány všechny relevantní metadata. Uživatel by pak mohl rozkliknout jednotlivé dokumenty a stáhnout jejich obsah.

Nakonec se však ukázalo jako nejjednodušší způsob vypsání obsahu samotné složky, tak jak bylo popsáno v kapitole 4.2.1. Nejedná se o nejlepší řešení z hlediska uživatelské přívětivosti, avšak z hlediska funkčnosti zde nic nechybí.

Samozřejmě pro vypsání složky je také na severu nutno zpracovat požadavek, a právě proto na archivačním serveru navíc běží Apache server, jehož úkolem je pouze na vyžádání vypsání složky a jednotlivých dokumentů. Pomocí aplikace Xampp je jeho zprovoznění velmi jednoduché. V podstatě jej stačí pouze zapnout, nastavit šifrování pomocí TLS a zajistit aby se obsah archivu nacházel v jeho kořenové složce.

Ve front endu pak stačí namapovat tlačítko „Archive“ tak, aby se odkázalo na adresu:

https://{url_archivu}/archive/

Důvodem pro toto řešení je, že podobně byl archiv řešen během testování validační aplikace, která vznikla jako první. Byla tak již ověřena jeho funkčnost, což umožnilo rychlejší vývoj.

4.2.3 Shrnutí webové aplikace

Pro shrnutí tedy webová část aplikace (front end a back end) dosáhne toho, že vezme soubory a metadata zadané uživatelem a uloží je do složky „uploads“. Uživatelé jsou autentizováni pomocí identifikačních údajů a mají schopnost prohlížet obsah archivu. Další zpracování už pak zajišťuje Archivační aplikace. Validaci pomocí blockchainu pak zajišťuje Validační aplikace.

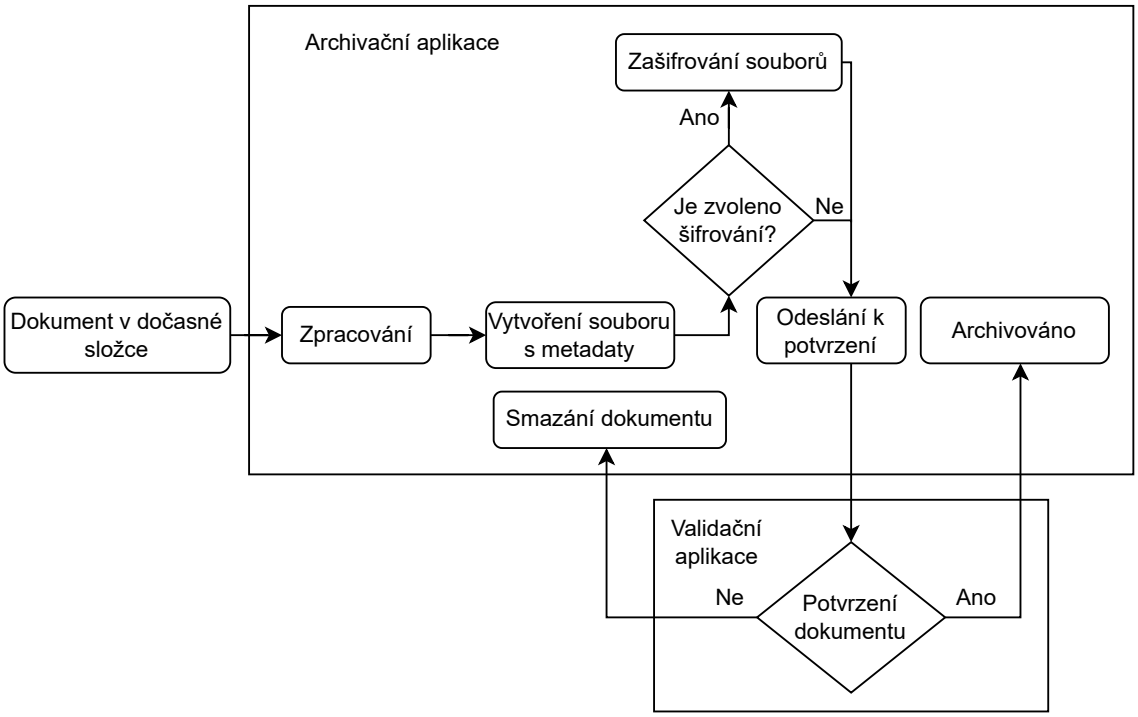
4.3 Archivační aplikace

Tato část sytému je napsána v programovacím jazyce Java a zajišťuje velké množství funkcí, především co se týče bezpečnosti. Jedná se opět o back end, takže běží na serveru, na kterém se nachází archivované soubory. Kód je v tomto případě vlastní. Funkce této aplikace se dají rozdělit do dvou oblastí:

- Zpracování dokumentů
- Komunikace s validačními aplikacemi

Aplikace je vícevláknová, s tím že různá vlákna mají na starosti různé funkce.

Kód aplikace je rozdělen do dvou tříd - Server a ClientHandler, s tím že Server obsahuje metody spojené s celou infrastrukturou a ClientHandler metody týkající se jednotlivých připojených klientů. Pod pojmem klient je myšleno samotné připojení, pod pojmem uživatel pak klient spolu s identifikačními údaji (uživatelské jméno a heslo).



Obr. 4.6: Zpracování archiválií archivační aplikací

Po spuštění se aplikace pokusí načíst konfigurační soubor „config.json“, který obsahuje definice údajů jako například cesta do složky archivu nebo cesta ke složce „uploads“. Také načte údaj o tom, zda je povolená volitelná funkce potvrzování nových dokumentů. Pokud tento soubor nenajde nebo ho nedokáže přečíst použije výchozí hodnoty.

4.3.1 Zpracování archiválií

První úlohou archivační aplikace je zpracování nových archiválií, které je znázorněno na diagramu 4.6. Zpracování má několik kroků:

- Přesun souborů do archivu
- Vytvoření souboru metadata.json
- Volitelné zašifrování souboru
- Odeslání souborů k validaci

Pro tento účel je po spuštění aplikace spuštěno vlákno **newDocumentThread**, které postupně spustí jednotlivé kroky. Jak bylo popsáno v kapitole 4.2.2, po uploadu dokumentu z webu skončí soubory s metadaty ve složce „uploads“, resp. v podsložkách nazvaných podle názvu dokumentu. Archivační aplikace tedy kontinuálně nahlíží do složky „uploads“, která je ve výchozím stavu prázdná, a pokud se v ní objeví nová podsložka, začne ji zpracovávat.

Přesun souborů do archivu

Nejprve je v archivu vytvořena nová složka s názvem dokumentu. Archivem se myslí složka „archive“, nacházející se v kořenové složce Apache serveru. Jelikož na back endu webové aplikace došlo k validaci dokumentu, nemělo by se stát, že by se v archivu nacházel dokument se stejným jménem. V této složce se pak nachází všechny dokumenty archivu a pomocí Apache je pak její obsah vypisován v prohlížeči.

Vytvoření souboru metadat

Metadata jsou zakódována v názvech souborů, Archivační aplikace tedy postupně projde všechny soubory a přejmenuje je tak, aby zůstaly pouze původní názvy souborů. Zároveň s tím si však zapamatuje metadata - název dokumentu, autor, šifrování a volitelné heslo.

Aplikace poté vytvoří soubor „metadata.json“, ve kterém zapíše následující údaje:

- author - autor dokumentu, tedy uživatelské jméno uživatele, který jej vytvořil
- added - přesný čas ve kterém byl dokument přidán do archivu
- docName - název dokumentu
- id - identifikační číslo dokumentu, které aplikace vygeneruje
- files - seznam názvů souborů obsahu
- encryption - hodnota zda jsou obsahy zašifrovány

Tento soubor následně lze v archivu prohlížet a je použit ve validační aplikaci k zobrazení nového dokumentu.

Dále záleží na tom, zda je povolené šifrování nebo ne. Pokud není, tak jsou jednoduše soubory obsahu přesunuty do nové složky. Pokud je povolené, přejde se k dalšímu kroku - šifrování. V obou případech jsou zpracované soubory vymazány z původní složky „uploads“, která by tak měla zůstat prázdná.

Šifrování souborů

Pokud bylo u dokumentu zvoleno šifrování souborů, obsahy dokumentů jsou aplikací zašifrovány. Povinným údajem bylo také heslo, které bylo aplikaci předáno. Zašifrování provede aplikace pomocí metody **encrypt** ve vlastní třídě **Crypto**.

Tato metoda používá standardní Java knihovny specializované na bezpečnost jako „javax.crypto“ a „java.security“. Na vstupu je tak původní soubor a heslo které bylo předáno z front endu. Metoda nejdřív musí z hesla vytvořit tajný klíč, který bude kompatibilní s algoritmem AES. Pro tento účel je použita funkce PBKDF2. Ta vyžaduje na vstupu heslo, hashovací funkci a sůl. Jako hashovací funkce je použita SHA-256. Sůl je nejdříve vygenerována, a to tak, že funkce vezme název souboru, délku hesla a tento řetězec zahashuje také pomocí SHA-256. Tyto údaje jsou použity

proto, aby byly jednoduše znovupoužitelné během dešifrování, ale zároveň byly unikátní pro daný soubor.

Pro šifrování je použit algoritumus AES v módu CBC, délka klíčů je 256 bytů a pro padding je použita funkce PKCS#5. Jako inicializační vektor je opět použit hash názvu souboru a délky hesla, který projde hashovací funkcí ještě jednou a následně se použije jeho prvních 16 bytů. Odbourává se tím nutnost uložení IV spolu se souborem, jedinou podmínkou je, že se soubor před dešifrování musí jmenovat tak, jak se jmenoval během šifrování.

Poté co jsou soubory zašifrovány, jsou uloženy do složky dokumentu. Následně se přejde na odeslání dokumentu k validaci.

Odeslání souboru k validaci

Tento krok už je velmi jednoduchý. Aplikace vezme název dokumentu a následně jednotlivé názvy souborů obsahu a pošle je náhodnému dostupnému administrátorovi ve zprávě s kódem „files“. Tím je jeden průběh vlákna hotov a aplikace se vrátí zpátky na začátek, kde monitrouje složku „uploads“.

4.3.2 Komunikace s validačními aplikacemi

Archivační aplikace slouží jako centrální bod pro validátory, kteří k němu jsou připojeni v hvězdicové topologii, viz obrázek 4.7. Jedná se tedy o formu API (a v této sekci tak je i pojmenována), která pak zajišťuje následující služby:

- Přeposílání zpráv mezi uživateli za pomocí TLS spojení
- Autentizace a řízení přístupu uživatelů
- Přidávání nových uživatelů
- Generování klíčů pro elektronické podpisy
- Posílání žádostí o potvrzení nových dokumentů a zpracování odpovědí

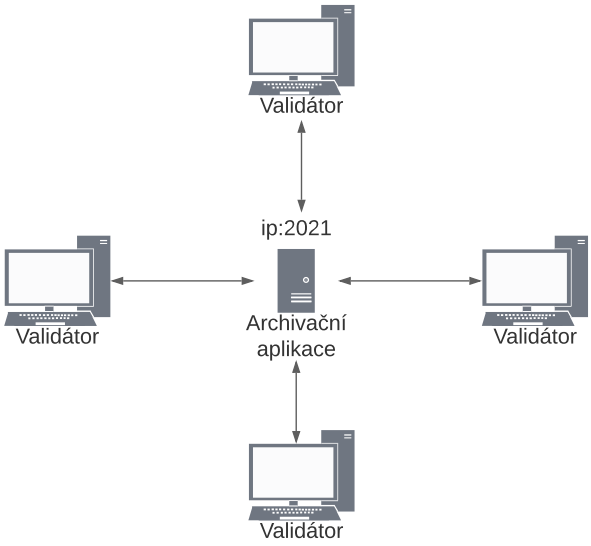
Každé spojení s validační aplikací probíhá ve vlastním vlákně, takže archivační aplikace dokáže asynchronně zpracovávat více požadavků zároveň.

Přeposílání zpráv

Na začátku se ve třídě Server vytvoří síťový socket, tedy koncový bod pro připojení. Ten poté naslouchá na portu 2021 a čeká na připojení uživatelů. Když se nějaký pokusí o připojení, je vytvořeno nové vlákno, které zpracovává veškeré zprávy od tohoto uživatele.

Socket je opatřen protokolem TLS v1.2, a tím pádem je komunikace mezi API a uživateli zašifrována a nelze ji jednoduše odposlouchávat. K vytvoření takového spojení je potřeba certifikát, ten je v této implementaci vygenerován předem pomocí

příkazu „keygen“. Jedná se o certifikát formátu X.509 uložený v souboru typu PKCS 12. Pro potřeby testování a lokální implementace stačí certifikát podepsaný sám sebou, v reálném provozu by bylo lepší použít certifikát ověřený nějakou důvěryhodnou certifikační autoritou.



Obr. 4.7: Topologie komunikace mezi validátory

Zprávy samotné se přenáší v TCP streamu a mají následující formu:

code;msg#recipient

,kde

- code - označení typu zprávy (např. „newblock“, pokud je přenášen nově vytvořený blok)
- msg - obsah zprávy (např. serializovaný nový blok)
- recipient - příjemce (buď „server“ nebo „broadcast“)

Když API přijme nějakou zprávu, zjistí zda je určena jí (hodnota recipient je „server“) nebo ostatním uživatelům (hodnota recipient je „broadcast“). Pokud je příjemce API, tak dále začne v metodě **parse** zpracovávat zprávu. Nejdřív zjistí hodnotu „code“ a v závislosti na ní zpracuje obsah zprávy. Kódy, které API může přijmout jsou následující:

- auth - obsahem zprávy jsou přihlašovací údaje, API se tedy pokusí uživatele autentizovat
- keys - uživatel žádá o podpisové klíče, API je pošle, jen pokud má práva admina
- confirmed - konfirmační zpráva o přijetí nového dokumentu
- rejected - admin odmítl nový dokument, API ho tedy z archivu smaže

- newuser - požadavek na vytvoření nového uživatele, obsahem zprávy je jméno a heslo

Na tyto zprávy následně zareaguje v závislosti na jejich typu a na tom, jak dopadne jejich zpracování.

Autentizace

Poté co se připojí některý z uživatelů, začne proces autentizace. Na rozdíl od webové aplikace, kde je využita externí platforma, archivační API řeší autentizaci uživatelů sama. Identifikační údaje jsou uloženy v souboru, který si API načte a následně použije při autentizaci.

Hesla jsou uložena v podobě hashů, takže je nelze jednoduše přečíst a vzhledem k tomu, že jsou přenášena v zašifrované podobě, nelze je ani získat odposlechem komunikace mezi API a uživateli. Ukládání v souboru místo databáze sice není ideální, ale zjednodušuje implementaci. Souboru se mohou nastavit přístupová práva jen pro uživatele, pod kterým je spuštěno API.

Každý uživatel pak má různá přístupová práva. Existují dvě úrovně - „admin“ a „user“. Admin má veškerá práva - tedy může přidávat nové bloky do blockchainu a účastní se na vytváření konsenzu pro blockchain, dá se tedy nazvat validátorem. User pouze dostává od ostatních zprávy a sám nemůže nic posílat. Může tedy pouze pozorovat vývoj blockchainu a validovat integritu archivu.

Klient tedy po připojení pošle své identifikační údaje, které API porovná s načtenými údaji, a pokud sedí, je uživatel považován za autentizovaného. Jeden uživatel může být připojen pouze jednou, jinak je autentizace odmítnuta. Výsledek autentizace je poslán zpět uživateli, kde je vypsán.

Důvody pro neúspěšnou autentizaci jsou tři:

- Uživatelské jméno není v seznamu uživatelů
- Heslo nesedí k uživatelskému jménu
- Uživatel je již přihlášen (nachází se v seznamu přihlášených uživatelů)

Každý uživatel je reprezentován třídou ClientHandler. Třída Server si uchovává několik seznamů uživatelů - za prvé je to seznam všech aktivních tříd ClientHandler - tedy seznam klientů připojených k socketu. Za druhé je to seznam přihlášených uživatelů, který slouží k ověření toho, aby se jeden uživatel nedokázal přihlásit vícekrát. A za třetí je to seznam dostupných administrátorů. Do něj jsou dynamicky přidávání a odebíráni administrátoři v závislosti na tom, jestli zrovna zpracovávají nějaký požadavek. Tento seznam slouží k tomu, aby nedošlo k zahlcení administrátorů více požadavky na potvrzení nových dokumentů.

Přidávání nových uživatelů

Přidání nových uživatelů je velmi jednoduché. Uživatel pošle požadavek (jak jej vytvoří, je popsáno v kapitole 4.4.8) na vytvoření nového uživatele v podobě zprávy s kódem „newuser“ a obsahem „uživatelské_jméno:heslo“. API vypočítá hash hesla a následně do souboru s identifikačními údaji přidá nový pár - uživatel/hash. Automaticky mu pak přiřadí prává „user“. Validátoři tedy mohou pouze vytvářet uživatele typu „user“, jediný, kdo dokáže povýšit uživatele na práva „admin“, je správce serveru, který má přístup k souboru s údaji a změní hodnotu „user“ na „admin“.

Generování klíčů pro elektronické podpisy

Každý blok je elektronicky podepsán některým z validátorů. API tedy slouží jako autorita, která tyto klíče dokáže vytvářet a distribuovat. Zároveň je potřeba ukládat všechny použité klíče, a to z toho důvodu, že jednou z validací integrity je zjištění, zda byl k podpisu použit legitimní klíč. Na začátku komunikace s uživatelem mu tak API předá seznam všech doposud použitých klíčů, které se pravidelně ukládají v souboru.

API vytvoří nový pár klíčů na vyžádání uživatele za předpokladu, že je autentizovaný a má práva „admin“. K podpisům se používá algoritmus RSA, délka klíčů je 2048b. Po vygenerování nového páru soukromý/veřejný klíč jej pošle uživateli a veřejný klíč si přidá do seznamu legitimních klíčů, který pak pošle všem ostatním uživatelům. Když na něj pak narazí při validaci bloků, považují jej za legitimní.

Posílání žádostí o potvrzení nových dokumentů

Volitelnou funkcí, kterou lze zapnout v konfiguračním souboru, je potvrzování nových dokumentů. Jak bude popsáno v další kapitole, když API narazí na nově vytvořený dokument, pošle jej některému z validátorů, aby jej ověřil a přidal do blockchainu. Pokud je zapnutá funkce potvrzování, dostane validátor možnost nový dokument odmítnout. To se může hodit například v případě, že uživatel na front end-u zadá špatné údaje, nebo omylem nahraje špatný soubor.

Když API odešle nějaký dokument k potvrzení, odebere daného uživatele do seznamu dostupných administrátorů. Zpátky ho přidá poté, co dojde odpověď o potvrzení nebo zamítnutí dokumentu. Díky tomu se nestane, že by nějakého uživatele zahltil více žádostmi.

Pokud uživatel potvrdí dokument, odešle zprávu s kódem „confirmed“, API pouze odpoví zprávou OK a přidá uživatele zpět do seznamu dostupných administrátorů. Pokud uživatel nový dokument odmítne, odešle zprávu s kódem „rejected“ obsahující název daného dokumentu. API pak daný dokument smaže z archivu včetně všech obsahů a opět vrátí uživatele zpět do seznamu dostupných.

4.3.3 Uživatelské rozhraní

Archivační aplikace není opatřena uživatelským rozhraním, veškeré důležité informace jsou vypisovány do konzole. Jediný, kdo k ní má přístup, je správce serveru, a to především z kontrolních důvodů. V ideálním případě se pouze zapne a následně bude běžet na pozadí.

4.3.4 Shrnutí archivační aplikace

Archivační aplikace, která běží na serveru spolu s back ende-m webové aplikace, plní dvě důležité funkce. Za prvé je to zpracování nově nahraných dokumentů. Konkrétně zpracuje jejich metadata a přesune obsahy do archivační složky. Pokud je zvolena možnost zašifrování, tak aplikace každý soubor zašifruje.

Druhou důležitou funkcí je komunikace s validačními aplikacemi. V tomto kontextu má archivační aplikace formu API. Zprostředkovává veškerou komunikaci mezi validátory, která je šifrována pomocí protokolu TLS. Dále zajišťuje autentizaci uživatelů, přidávání nových uživatelů a řízení přístupových práv. Validátorům také posílá nově nahrané dokumenty k validaci a přidání do blockchainu. Navíc také generuje a distribuuje klíče k elektronickým podpisům.

Jedná se tedy o jakousi páteř, která propojuje všechny ostatní aspekty systému.

4.4 Validační aplikace

Poslední částí celého archivačního systému je validační aplikace. Jedná se o hlavní komponentu zajišťující integritu archivu a implementaci blockchainu. Jak již bylo zmíněno, je to desktopová aplikace napsaná v jazyce Java, kód je vlastní, a to včetně úplné implementace blockchainu, kde není využito jakékoliv externí knihovny. Aplikace rozsáhle využívá multithreading. Její vzhled bude demonstrován v kapitole 4.7 Demonstrace funkčnosti aplikace.

Zajišťovat bude následující funkce:

- Připojení k archivu
- Autentizace uživatelů pomocí přihlašování
- Přidávání nových uživatelů
- Komunikace s archivační aplikací a s ostatními validátory
- Potvrzování nových dokumentů
- Vytváření a uchovávání blockchainu
- Validace integrity archivovaných dat

To vše bude uživatelům prezentováno skrze uživatelské rozhraní (GUI).

4.4.1 Připojení k archivu

Když uživatel zapne validační aplikace, jeho prvním (a jediným možným) krokem je připojení k archivu. Připojení probíhá zadáním URL archivu do pole a stisknutím tlačítka **Připojit**. Pokud je URL správná a na daném serveru běží Archivační aplikace, dojde k vytvoření spojení přes socket, šifrovaného pomocí TLS v1.2, tak jak bylo popsáno v kapitole 4.3.2.

Stav připojení je vypsán do části GUI nazvané **Log komunikace**. Pokud je tedy neúspěšné, uživatel se to dozví.

4.4.2 Přihlášení

Po připojení se může uživatel přihlásit. Do polí **Uživatel** a **Heslo** zadá svoje údaje a ty jsou poslány archivační aplikaci. Ta provede autentizaci tak, jak byla popsána v kapitole 4.3.2. Výsledek autentizace je předán validační aplikaci, která jej vypíše do **logu kommunikace**. Zároveň se také uživatel dozví, jaká má práva - „admin“ nebo „user“.Po přihlášení už může probíhat veškerá komunikace s archivem a s ostatními validátory.

4.4.3 Komunikace s API

Jak bylo popsáno v kapitole4.3.2, komunikace probíhá pomocí TCP streamu, zabezpečena je pomocí TLS. Spojení je dosaženo pomocí síťových socketů. Všechny zprávy mají formu:

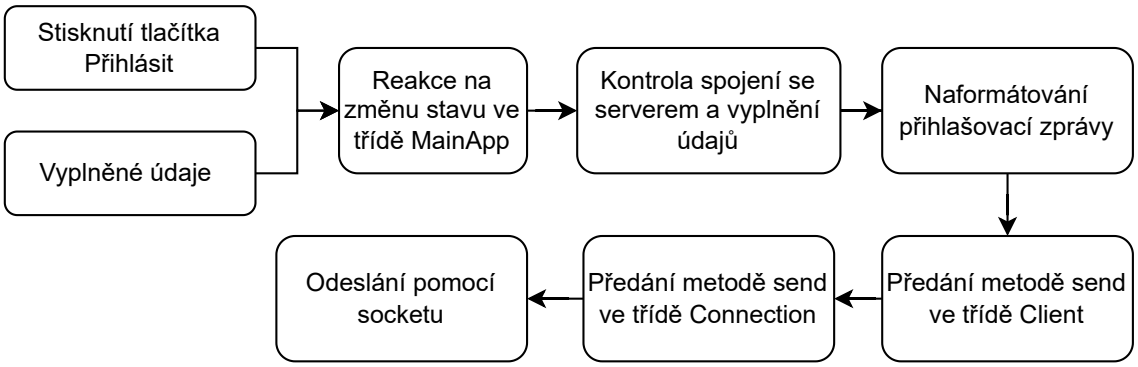
code;msg#recipient

s tím, že možné kódy, které validační aplikace přijímá od archivační aplikace, jsou následující:

- authSucc - autentizace byla úspěšná
- authFail - autentizace byla neúspěšná, obsahem zprávy je přesný důvod
- pubkey - obsah je serializovaný seznam validních veřejných klíčů
- keypair - obsah je serializovaný veřejný a soukromý klíč
- keys - obsah indikuje, zda je povolená funkce potvrzování nových dokumentů
- files - obsah je serializovaný seznam nových dokumentů a souborů poslaný k validaci a přidání do blockchainu

Od ostatních validátorů pak může dostávat následující zprávy:

- blockrequest - žádost o poslání blockchainu
- blockresponse - obsahem je serializovaný blockchain
- newblock - obsahem je nově vytvořený blok



Obr. 4.8: Zpracování odeslání požadavku na přihlášení

Serializace a deserializace objektů je řešena pomocí základní Java knihovny Serializable. Komunikace je řešena hierarchicky ve třech třídách - Connection, Client a MainApp.

Třída MainApp řeší především uživatelské rozhraní, v kontextu komunikace tak například reaguje na situaci, když uživatel stisikne tlačítko **Připojit**. Zároveň také vypisuje důležité informace do **Logu komunikace**. Obsahuje také vlákno ClientThread, které kontroluje vlastnosti uvnitř třídy Client a reaguje na jejich změny.

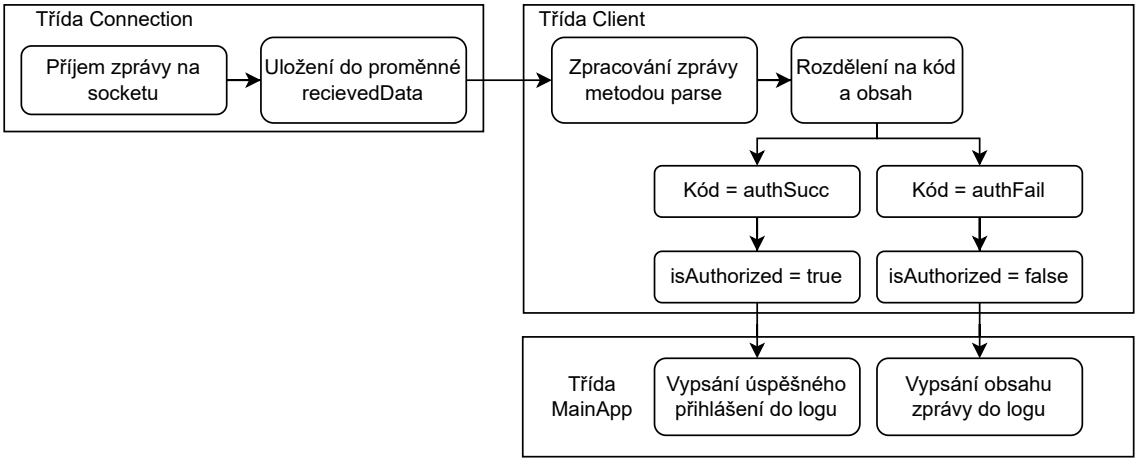
Třída Client řeší obsahy jednotlivých zpráv a reakce na ně. Právě zde probíhá zpracování zpráv podle jejich kódů. Třída také obsahuje množství vlastností, které se průběžně mění v reakci na přijaté zprávy. Třída dále obsahuje vlákno createConnection, která kontroluje třídu Connection.

Třída Connection už pak řeší samotné spojení. Obsahuje socket, na kterém poslouchá a přijímá zprávy. Zároveň pak dokáže odesílat zprávy. Neřeší se zde obsah, jde jen a pouze o data.

Nejlépe lze komunikaci mezi třídami vysvětlit na příkladu přihlášení. Uživatel zadá přihlašovací údaje a stiskne tlačítko **Přihlásit**. To zachytí třída MainApp. Ta vezme zadané údaje a řekne třídě Client, aby odeslala zprávu obsahující tyto údaje. Třída Client naformátuje zprávu do správného formátu a předá ji třídě Connection k odeslání. Tento proces je znázorněn na diagramu 4.8.

Dejme tomu, že je autentizace úspěšná a archiv odešle zpět zprávu „authSucc“. Třída Connection ji zachytí a uloží do proměnné **receivedData**. Vlákno **createConnection** ve třídě Client zjistí, že se hodnota proměnné změnila, a převezme její obsah. Spustí se funkce **parse**, která zprávu rozdělí a zjistí její kód. Ten je v tomto případě „authSucc“. Třída Client změní proměnnou **isAuthorized** na **true**, na což zareaguje vlákno ClientThread v třídě MainApp a do **Logu komunikace** vypíše, že autentizace byla úspěšná. Zpracování přijaté zprávy je znázorněno na obrázku 4.9.

Po úspěšném přihlášení si validační aplikace od serveru vyžádá vygenerování



Obr. 4.9: Zpracování odpovědi na požadavek na přihlášení

klíčů k elektronickému podpisu. Archivační aplikace této žádosti vyhoví a zároveň ještě pošle seznam legitimních klíčů, který si validační aplikace uloží pro potřeby validace blockchainu.

V neposlední řadě pak načte svůj uložený blockchain a následně vyžádá od ostatních připojených uživatelů o jejich verzi blockchainu.

Celý začátek úspěšné komunikace tak lze vidět na obrázku 4.10.

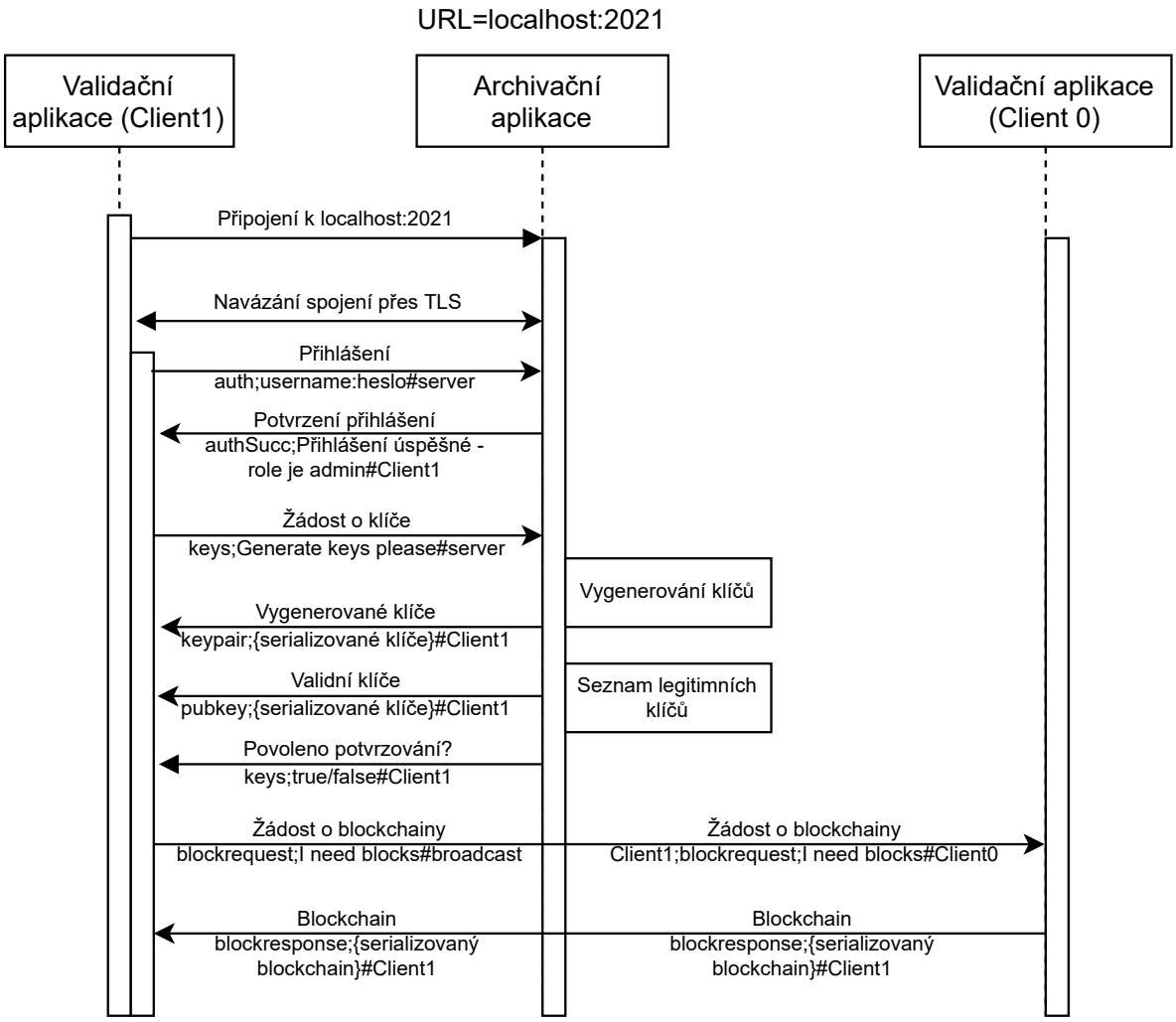
4.4.4 Blockchain

Pokud je hlavním účelem aplikace validovat integritu archivovaných dat, pak blockchain je způsob, jakým je tohoto dosaženo. Jeho hlavní funkcí je uchovávat záznamy o jednotlivých dokumentech v archivu. Na základě toho pak bude probíhat validace souborů v dokumentech. Implementovaný blockchain je znázorněn na obrázku 4.11

Blockchain je uchováván aplikací ve formě objektu skládajícího se z bloků. Každý blok obsahuje následující atributy:

- Název dokumentu
- Názvy souborů obsahu
- Název souboru s metadaty
- Časové razítko – kdy byl blok přidán do blockchainu
- ID bloku – automaticky vygenerované identifikační číslo
- Hash souborů obsahu
- Digitální podpis – elektronický podpis souboru, včetně veřejného klíče
- Hash předchozího bloku

Název dokumentu a názvy souborů slouží k nalezení správné cesty pro validaci souborů. Časové razítko označuje čas vytvoření daného bloku a slouží pouze pro lepší identifikaci bloku. ID bloku je vygenerované automaticky.



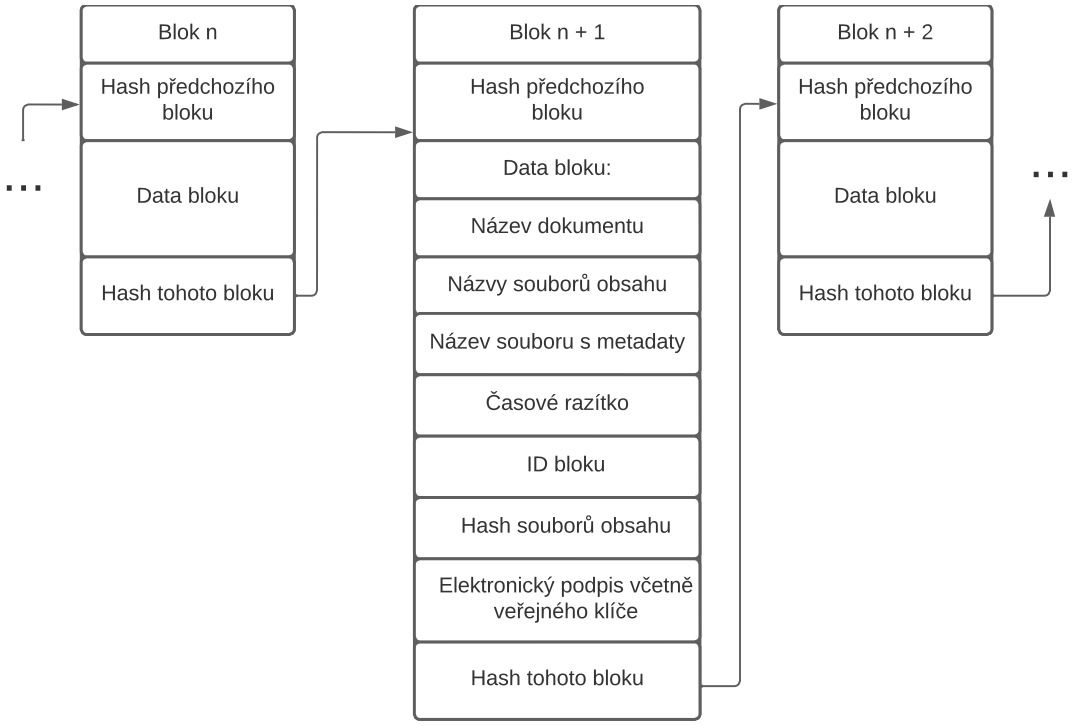
Obr. 4.10: Začátek komunikace mezi validační aplikací a archivem

Stěžejním údajem je hash souboru obsahu. Při vytváření nového bloku vypočítá aplikace tento hash. Pokud je obsahů více než jeden, tak aplikace vypočítá hashe pro všechny a následně je spojí do jednoho delšího řetězce. Na základě něj pak bude validovat celý blockchain a integritu dat.

Kromě výpočtu hashe dojde během vytváření bloku k podepsání souboru. Do bloku je pak přiložen podpis a jeho veřejný klíč. Během validace bloku je podpis ověřen, což dále chrání integritu dat a jedná se o důkaz autorství tohoto bloku.

Distribuce těchto klíčů byla popsána v předchozích kapitolách a společně s autentizací tak pro blockchain vytváří Proof of Authority – důkaz autoritou. Jediný způsob, jakým lze získat klíč, který budou ostatní považovat za validní, je autentizace u archivační aplikace, která je všemi považovaná za důvěryhodnou. Tím pádem, pokud je blok podepsaný validním klíčem, je důvěryhodný.

Jak bylo popsáno v kapitole 2.1, hashe bloku slouží k vytváření samotného



Obr. 4.11: Implementovaný blockchain

blokchainu. Každý blok obsahuje odkaz na předchozí blok v podobě jeho hashe. Dále obsahuje blok sebe samého. Při validaci jsou tyto hashe porovnávány a tím je zajištěna integrita blockchainu.

Konsenzus

Velmi důležité je, aby nad blockchainem panoval konsenzus, tedy aby se všichni validátoři shodli na stejném validním blockchainu. Toho je dosaženo pomocí žádostí o blockchain a následnými odpověďmi. Tyto žádosti aplikace vysílá po přihlášení, dále pokaždé když vytvoří nový blok a na vyžádání při výpisu blockchainu.

Žádost je rozeslána všem ostatním validátorům a ti odpoví svým serializovaným blockchainem. Aplikace tedy chvíli čeká a poté vezme všechny blockchainy, které dostala (společně se svým), pro každý provede validaci hashů bloků a validitu klíčů a následně z validních vybere ten, který se vyskytuje nejčastěji.

Zde je důležité při posuzování nevalidovat integritu dat. To by totiž bylo extrémně nebezpečné. Mohla by nastat následující situace – útočník změní některý z uložených souborů a vytvoří si vlastní blockchain, který změnu reflektuje. Poté se připojí nový uživatel. Zažádá ostatní o jejich blockchain, jako odpověď dostane několik původních blockchainů, před porušením integrity a útočníkův blockchain. Pokud by následně kontroloval validitu blockchainu včetně integrity souborů, jako jediný validní by vyšel

ten útočníkův a nový uživatel by jej přijal.

Je potřeba si uvědomit, že na blockchainu, který neprojde validací integrity dat není nic špatného, jen plní svoji funkci. Proto je v pořádku pokud se vytvoří konsenzus i nad takovým blockchainem.

Pokud validátor nedostane žádnou odpověď nebo dostane pouze jeden blockchain, ponechá si ten svůj. Pokud dostane žádost v momentě, kdy je jeho blockchain prázdný, tak na žádost neodpoví. Pokud jsou během výběru dva blockchainy stejně časté, vybere náhodně jeden z nich.

V momentě, kdy jsou uživatelé čestní, by nemělo docházet k situacím, kdy dva takoví uživatelé mají různý blockchain. Dá se tedy předpokládat, že aby si nečestný uživatel prosadil vlastní blockchain, potřeboval by poslat více blockchainů než ostatní uživatelé dohromady. Z toho tedy lze vyvodit, že pokud je přihlášeno více čestných uživatelů než nečestných, blockchain by měl být bezpečný.

Při zavření validační aplikace je blockchain uložen v serializované podobě a při otevření jej aplikace pokusí načíst. Tím je zajištěna kontinuita blockchainu. Pokud by se blockchainy neukládaly, tak by byl v momentě, kdy není připojený ani jeden validátor, blockchain ztracen a celý systém by se musel resetovat jelikož přidávání nových dokumentů do blockchainu je možné jen v momentě jejich uploadu. Je tedy důležité aby v každém momentu existovala alespoň jedna kopie validního blockchainu.

4.4.5 Potvrzování nových dokumentů

Jednou z funkcí validační aplikace je potvrzování nově nahraných dokumentů. Jak již bylo několikrát zmíněno, jedná se o volitelnou funkci. Validační aplikace se dozví, zda je zapnutá na začátku komunikace s archivem.

Když Archivační aplikace zpracuje nový dokument a uloží jej do archivu, pošle náhodnému validátorovi zprávu typu „files“, která obsahuje název nového dokumentu a seznam všech souborů obsahu. Validační aplikace tuto zprávu zachytí, a pokud je potvrzování povoleno, otevře potvrzovací okno.

V tomto potvrzovacím okně jsou vypsána všechna metadata, které aplikace získá ze souboru „metadata.json“ v archivu. Okno pak má dvě tlačítka – **Potvrdit dokument** a **Odmítnout dokument**. Validátor, který má toto potvrzení na starosti, má teď možnost posoudit, zda chce dokument potvrdit nebo ne. Důvody pro odmítnutí mohou být například překlepy v metadatech, špatné vyplnění metadat, nezatrhnutí šifrování apod. Mimo metadata má také možnost se přes webovou aplikaci podívat do archivu a posoudit obsah, například může validátor zjistit, že uživatel omylem nahrál špatný obsah.

Rozhodnutí je tedy na validátorovi. Je nutno říct, že pokud je dokument potvrzen, je považován za archivovaný a nelze jej smazat bez porušení blockchainu. Pokud jej

přijme přejde se k přidávání bloku.

Pokud je potvrzování nových dokumentů vypnuté, žádné okno se validátorovi nezobrazí a rovnou se přejde na vytváření nových bloků. Toto se může hodit například v případech, kdy je nahráváno velké množství dokumentů, které by validátoři nestíhali zpracovat.

4.4.6 Přidávání bloků

Přidávání nových bloků je poměrně jednoduché. Aplikace má k dispozici seznam souborů v novém dokumentu a jeho metadata. Postupně tak vypočítá hash každého ze souborů, a to přímým přístupem pomocí Apache serveru. Toto spojení je chráněno protokolem TLSv1.3, takže data proudí v zašifrované podobě. Spolu s url, na které běží archiv a názvem nového dokumentu si vytvoří adresu pro každý dokument ve formě:

https://{url}/archive/{název_dokumentu}/{název_souboru}

Pro každý dokument si pak vytvoří InputStream, který následně zashashuje pomocí algoritmu SHA–256. K tomu je použita Java knihovna „java.security“ a její třída MessageDigest. Výsledkem je tedy textový řetězec o délce 256 bitů. Pokud je souborů, a tím pádem i hashů, více, jsou zařazeny za sebe, čímž se vytvoří delší řetězec. Daný řetězec je pak považován za hodnotu „filehash“.

Dalším krokem je elektronický podpis. Pokud vše proběhlo správně, měl by validátor mít k dispozici pár elektronických klíčů, které ostatní validátoři považují za důvěryhodný. Aplikace tedy vezme řetězec filehash a podepíše jej pomocí svého soukromého klíče. K samotnému podpisu je využita funkce RSA.

Posledním krokem je výpočet hashe předchozího bloku. Zde se vezmou hodnoty jednotlivých atributů v posledním bloku blockchainu, přidají se do jednoho řetězce a následně je vypočítán hash tohoto řetězce. Opět je použita hashovací funkce SHA-256

Pak už aplikace pouze poskládá dohromady nový blok. Přidá název dokumentu a názvy všech souborů, hash předchozího bloku, jako časové razítko použije naformátovaný aktuální čas, přidá hodnotu „filehash“, podpis a veřejný klíč podpisu.

Následně si blok přidá ke svému blockchainu a pošle broadcast zprávu s kódem „newblock“, kde je obsahem serializovaný blok. Tato zpráva by pak měla dorazít všem ostatním validátorům.

Než si validátor přidá nově vytvořený blok, který mu právě dorazil, ověří nejdříve jeho platnost. To znamená, že přepočítá hodnotu „filehash“, zkontroluje zda je elektronický podpis validní, zda je veřejný klíč legitimní a zda hodnota předchozího hashe v novém bloku odpovídá hodnotě posledního bloku v blockchainu. Pokud nový

blok projde touto kontrolou, validátor si ho přidá do blockchainu. Tímhle způsobem si tedy kolektivně validátoři přidávají nové a nové bloky do svých blockchainů.

4.4.7 Validace blockchainu

Pokud vše probíhá správně, měli by všichni validátoři vlastnit stejný blockchain obsahující záznamy všech dokumenů v archivu. Lze tedy přejít k samotné validaci. Ta v implementaci probíhá na vyžádání stisknutím tlačítka **Validace Blockchainu**, ale nebyl by problém kód upravit tak, aby probíhala automaticky v určitém časovém intervalu.

Samotná validace je pak poměrně přímočará. Blockchain je předán třídě **BlockchainValidator**, která provede sérii kontrol a zjistí tak, jestli došlo k porušení integrity.

BlockchainValidátor načte blockchain a skrze něj začne postupně procházet jednotlivé bloky. Stejným způsobem, jakým je vytvořen hash při vytváření bloků (viz předchozí kapitola), je vytvořena hodnota „filehash“. Poté jsou provedeny následující kontroly:

- 1. Porovnání vypočítané hodnoty „filehash“ s hodnotou v blokou
- 2. Ověření zda je veřejný klíč podpisu v seznamu legitimních klíčů
- 3. Ověření platnosti elektronického podpisu
- 4. Porovnání hashe předchozího bloku s hodnotou uloženou v daném bloku

Pokud jediný z těchto čtyř kroků selže, je blok považován za nevalidní a integrita archivu byla porušena. Validátor si výsledky ukládá pro každý blok a následně vygeneruje výpis, který přesně popíše průběh validace. Tento výpis je ukázán uživateli.

Díky tomuto systému je tedy zaručena integrita dat, respektive pokud je porušena, validátoři se to dozví a situaci lze řešit. V reálném nasazení by proto docházelo k periodickému zálohování archivu a blockchainů tak, aby se v případě porušení šlo vrátit do stavu před porušením.

Dále je zajištěna autentičnost bloků, a tím pádem i souborů, protože za každý musel ručit některý z autorizovaných validátorů. Vždy je tedy jasné, zda je blok legitimní nebo ne.

4.4.8 Přidání nových uživatelů

Poslední funkcí validační aplikace je přidávání nových uživatelů. Stiskem tlačítka **Nový uživatel** je zobrazen dialog, kde validátor zadá uživatelské jméno a heslo nového uživatele. Po potvrzení je archivační aplikaci odeslána zpráva s kódem „newuser“. Ten ji zpracuje, pouze pokud má validátor práva „admin“, jak bylo vysvětleno v kapitole 4.3.2.

Nový uživatel má vždy pouze práva „user“, pokud chce být povýšen, je potřeba zásah administrátora serveru, který má přístup k souboru s údaji, kde stačí upravit

záznam uživatele tak, aby měl práva „admin“. Tímto omezením se zamezí tomu, aby útočník, který získá kontrolu nad jedním účtem s právy „admin“, nemohl vytvořit velké množství nových účtů se stejnými právy. V takové situaci by pak jednoduše získal kontrolu nad blockchainem, protože by jeho verze vždy převažovala nad ostatními.

4.4.9 Uživatelské rozhraní

Aplikace je osazena grafickým uživatelským rozhraním (zkráceně GUI – graphical user interface), pomocí kterého uživatel interaguje s uživatelskými funkcemi aplikace. Většina z nich se nachází v hlavním okně aplikace, které lze vidět na 4.12. To se skládá z následujících tlačítek, polí a výpisů:

- Pole na zadání URL archivu
- Tlačítko Připojit k archivu (viz kapitola 4.4.1)
- Pole na zadání uživatelského jména
- Pole na zadání hesla
- Tlačítko Přihlásit (viz kapitola 4.4.2)
- Tlačítko Výpis Blockchainu
- Tlačítko Validovat Blockchain (viz kapitola 4.4.7)
- Výpis Blockchainu
- Log Komunikace
- Tlačítko Nový uživatel (viz kapitola 4.4.8)

Funkce většiny těchto prvků byla vysvětlena v předchozích kapitolách. Zbývá tedy krátce popsat Výpis Blockchainu a Log komunikace.

Po stisknutí tlačítka **Výpis Blockchainu** aplikace nejdříve synchronizuje blockchain s ostatními (pošle žádost o blockchainy a vybere nejčastější). Následně vypíše obsah blockchainu v podobě textového řetězce. Postupně jsou vypsány jednotlivé bloky v pořadí, v jakém byly přidány do blockchainu. Pro každý blok jsou vypsána všechna data v něm uložená.

Log komunikace pak vypisuje nejdůležitější události, které se udávají na pozadí - přihlášení nebo třeba přidání nového bloku. Slouží k hrubé orientaci v tom, které funkce aplikace jsou právě používány.

4.4.10 Struktura aplikace

V neposlední řadě bude krátce popsána struktura validační aplikace. Aplikace je složena z několika hlavních tříd podporovaných pomocnými třídami.

Obr. 4.12: Hlavní okno validační aplikace

Hlavní třídy

Hlavní třídou, ze které je celá aplikace spuštěna, je třída **App**. Ta obsahuje v podstatě pouze definice jednotlivých oken, která jsou inicializována. Hlavní okno je pak definováno ve třídě **MainApp**. Ta obsahuje mimo jiné uložený blockchain, metody reagující na stisknutí jednotlivých tlačítek, metody na vytváření nových bloků a na zjištění nejčastějšího blockchainu.

Jak již bylo popsáno v kapitole 4.4.3, **MainApp** také obsahuje objekt třídy **Clien**t, který slouží ke zpracování zpráv. Samotné spojení s archivem pak obstaraává třída **Connection**.

Vlastní třídu mají také okna na vytvoření nového uživatele a potvrzení nového dokumentu - **NewUser** a **ShowDocument**.

Pomocné třídy

Mimo tyto hlavní třídy, které ovládají fungování celé aplikace existuje několik pomocných tříd. Těmi nejdůležitějšími jsou Blockchain, Block, Crypto a BlockchainValidator.

Třída **Blockchain** reprezentuje celý blockchain a především obsahuje seznam bloků uložených ve struktuře LinkedList. Každý blok je pak reprezentován třídou

Block, která obsahuje všechny atributy, které jsou v blocích zaznamenány (viz kap. 4.4.4).

Třída **Crypto** pak obsahuje všechny kryptografické metody použité v rámci celé aplikace. Tuto třídu používá i **Archivační aplikace** a aplikace **Decryptor**. Metody, které se zde nachází, jsou následující:

- generateKeyPair - Vygeneruje náhodný pár RSA klíčů (soukromý a veřejný) o velikosti 2048 b
- sign - Vezme data v podobě pole bytů a soukromý klíč a vytvoří podpis v podobě textu
- verify - Vezme data v podobě pole bytů, podpis a veřejný klíč a ověří platnost podpisu
- blockHash - Vezme objekt typu Block a pomocí funkce SHA-256 vypočítá jeho hash. Do funkce jsou vkládány všechny atributy bloku.
- getFileHash - Vypočítá hash zadaného souboru
- getStringHash - Vypočítá hash textového řetězce
- getHashOfUrls - Vezme seznam názvů souborů, url archivu a název dokumemntu a vypočítá hash (viz kap. 4.4.6)
- encrypt - Zašifruje soubor pomocí hesla
- decrypt - Dešifruje soubor na základě hesla
- cryptoFile - Funkce, která provede samotné šifrování
- getKeyFromPassword - Pomocí funkce PBKDF2 vypočítá z hesla a soli, šifrovací klíč
- generateIv - Vygeneruje inicializační vektor na základě vstupní soli
- serialize - Serializuje objekt
- deserialize - Deserializuje objekt

Třída **BlockchainValidator** obsahuje metody určené k validaci blockchainu a nových bloků, tak jak bylo popsáno v kapitole 4.4.7.

4.4.11 Shrnutí validační aplikace

Validační aplikace tedy především zajišťuje integritu archivovaných dat. Komunikuje s archivační aplikací a s ostatními validačními aplikacemi. Spolu pak vytváří blockchain, který uchovává záznamy o archivovaných dokumentech. Pomocí hashovacích funkcí pak validační aplikace má schopnost detekovat změny v archivovaných datech, a tím pádem dokáží kontrolovat jejich integritu.

4.5 Aplikace Decryptor

Jak bylo naznačeno v kapitole 4.2.1, dešifrování archivovaných dokumentů bylo nakonec delegováno pro zvláštní aplikaci Decryptor. Jedná se o poměrně přímočarou aplikaci, která využívá třídu **Crypto** k dešifrování souborů.

Je osazena jednoduchým GUI, pomocí kterého uživatel zadá zašifrovaný soubor (buď zadáním cesty nebo výběrem v průzkumníku), a heslo, které bylo použito pro šifrování. Stisknutím tlačítka **Dešifrovat** poté dojde k dešifrování zadaného souboru. Ve stejné složce, jako se nachází původní soubor, se vytvoří nový soubor s názvem ve formátu „Decrypted-původní název“, jeho obsah by pak měl být stejný jako soubor před zašifrováním.

Jednou z nutných podmínek je, aby se zadaný soubor jmenoval stejně, jako se jmenoval v archivu. Důvodem je to, že název souboru je používán k odvození Inicializačního vektoru, který musí být stejný při šifrování a dešifrování.

Pokud je heslo špatné nebo se soubor nejmenuje stejně, dešifrování neproběhne správně. V metodě decrypt pak dojde k výjimce a uživateli se objeví hláška „Špatné heslo!“. Dešifrovaný soubor by v tomto případě byl stále nečitelný.

4.6 Shrnutí implementovaných bezpečnostních mechanismů

V návrhu byly vytyčeny tři hlavní oblasti bezpečnosti archivu, kterých by měl systém dosáhnout. Těmi byla integrita archivovaných dat, řízení přístupu a důvěrnost dat. Zde tedy budou shrnuty mechanismy, které byly implementovány k zajištění těchto bezpečnostních oblastí a do jaké míry jsou tyto mechanismy dostačující.

4.6.1 Integrita dat

Asi stěžejním bezpečnostním aspektem, na který je tento archiv zaměřen, je zajištění integrity dat. Především proto je implementován distribuovaný blockchain. Integrita dat může být nejčastěji porušena ze dvou důvodu - kvůli chybě (např. se zkorumpuje disk na kterém jsou uloženy) nebo záměrnou změnou dat.

Blockchain v tomto archivu zajistí, že ať k porušení došlo z jakéhokoliv důvodu, vždy bude detekováno. Chybám lze apriori předcházet jen těžko, proto je detekce důležitá.

V případě útoku je první překážkou fakt, že aby útočník data vůbec dokázal nějakým způsobem změnit, musí mít přístup k server, na kterém jsou archivovány. Při reálných implementacích je tak velmi důležité správně volit přístupová hesla a správně řídit přístup ke složkám. V ideálním případě by přímý přístup k samotným

datům měl pouze jeden člověk - administrátor serveru. Kdyby se však útočníkovi podařilo data nějakým způsobem porušit, systém tuto změnu opět zachytí.

Jediný způsob, jakým by se útočník vyhnul detekci, je ovládnutí blockchainu. Asi nejpřímočařejším způsobem jak tohoto dosáhnout by bylo vytvoření falešných uživatelských účtů s admin právy, pomocí kterých by se pak přihlásil do systému a následně mohl mezi ostatní šířit blockchain, který by bral v potaz upravená data. Pokud by ovládal dostatečné množství účtů, ostatní by přejali jeho verzi jako tu pravou. Poté co by zkusili validovat změněný archiv, validace by prošla a na porušení integrity by se tak nepřišlo.

Je však potřeba si uvědomit, že by k tomuto útoku opět potřeboval přístup do serveru pod právy root. Takový scénář sice není nemožný, ale jedná se o poměrně extrémní narušení bezpečnosti a opět by muselo nejdříve selhat řízení přístupu. Útočník by si v takovém stavu mohl dělat téměř cokoliv, včetně man-in-the-middle útoků, kde by v podstatě nahradil archivační aplikaci jako centrální bod komunikace.

Dalším prvkem jsou elektronické podpisy. V podstatě zajišťují to stejné co hashe, avšak navíc přidávají autentičnost jednotlivých bloků a blockchainu přidávají důkaz autoritou - Proof of Authority. Elektronické podpisy jsou často používány v reálných digitálních archivech.

Pokud tedy archiv má schopnost reagovat na porušení integrity, vyvstává otázka, jestli neexistuje způsob, zda takovému porušení nelze apriori zabránit. Přece jenom detekce je důležitá, ale pokud dojde k narušení dat, bylo by dobré mít způsob, jak vše vrátit do původního stavu. Asi nejrozumnějším řešením je periodické zálohování archivu, nejlépe na nějaké externí úložiště. Opět se jedná o reálně užívanou metodu a dala by se dobře kombinovat s možností rané detekce, kterou nabízí blockchain. Pokud by došlo k detekci, administrátor serveru by měl schopnost obnovy do stavu před narušením.

4.6.2 Řízení přístupu

Řízení přístupu je implementováno na dvou, respektive na třech místech. První dvě jsou přímo v popsané implementaci - řízení přístupu k webové aplikaci a řízení přístupu u validační aplikace. Třetím místem je pak přístup k samotnému prostředí kde běží webová aplikace, Apache server a archivační aplikace. To v této implementaci nebylo řešeno, protože vše bylo implementováno a testováno na vývojovém počítači. Při reálném nasazení by však šlo o neméně důležitý aspekt bezpečnosti.

Řízení přístupu k webové aplikaci je řešeno externě pomocí platformy Auth0. Uživatelé, kteří chtějí do archivu nahrávat nové dokumenty, se musí u této služby autentizovat. Jedná se o velmi bezpečnou implementaci řízení přístupu, jediným slabým místem jsou v podstatě jen špatně volená hesla.

U validační aplikace byl zvolen jiný přístup. Uživatelé, validátoři, se autentizují přímo u archivační aplikace, ke které se připojují. K tomu jsou využity přístupové údaje, které jsou posílány v zašifrované podobě a porovnávány s hashem uloženým v souboru, ke kterému má přístup pouze archivační aplikace.

V neposlední řadě jsou řešeny role jednotlivých uživatelů. Za prvé v systému existuje alespoň jeden administrátor serveru, který má přístup k samotnému serveru, kde aplikace běží. Pak existuje správce webové aplikace, který má přístup do správy Auth0 autentizace. Reálně by se nejspíš jednalo o stejnou osobu.

V rámci webové aplikace pak existují uživatelé/archiváři, kteří mají možnost nahrávat nové dokumenty. V rámci validační aplikace pak existují dvě různé úrovně práv. Za prvé jsou to práva „user“, kde mají uživatelé možnost pouze pozorovat vývoj blockchainu a validovat pomocí něj archiv. Za druhé jsou to práva „admin“, kde mají uživatelé právo vytvářet a přidávat nové bloky a odesílat je ostatním.

4.6.3 Důvěrnost dat

Důvěrnost přenášených dat je řešena pomocí protokolu TLS. Veškerá komunikace mezi jednotlivými prvky využívá tento protokol, který zajistí šifrování přenášených dat a nemožnost jejich odposlechu.

Důvěrnost uložených dat je volitelnou funkcí. Během nahrávání nového dokumentu ji můžou uživatelé povolit. Data jsou poté uložená pouze v zašifrované podobě. Dešifrování pro potřeby čtení pak zajišťuje aplikace Decryptor.

4.7 Demonstrace funkčnosti aplikace

V této kapitole bude demonstrována funkčnost systému. Nejlepším způsobem bude simulace následujícího scénáře:

- Validátor zapne validační aplikaci, připojí se k archivu a autentizuje se svými údaji, poté čeká na příjem nového dokumentu
- Pro kontrolu validátor vypíše existující blockchain
- Jiný uživatel se mezitím přes webový prohlížeč připojí k webové aplikaci.
- Pokusí se přejít na záložku upload ale je přesměrován k autentizaci
- Autentizuje se správnými údaji a přejde k uploadu
- Zadá název dokumentu, povolí šifrování a napíše heslo
- Pokusí se nahrát soubor, ale ten neprojde některou z validací
- Nahradí ho jiným souborem a odešle jej
- Validátorovi vyskočí okno pro potvrzení nového dokumentu
- Validátor potvrdí dokument
- Validátor vypíše nový blockchain, který bude obsahovat nový blok

- Validátor spustí validaci archivu
- Uživatel přejde na obsah archivu, nalezne nový dokument a stáhne si obsah
- Uživatel dešifruje obsah souboru pomocí aplikace Decryptor

Tento scénář obsahuje téměř všechny funkce archivu a dobře tak demonstruje jeho funkčnost. Všechny aplikace běží na stejném stroji, komunikace tak probíhá přes loopback adresu. Systém však bezproblémově funguje v rámci lokální sítě a přenesení do internetu by teoreticky nemělo dělat potíže.

Spuštění jednotlivých komponent je tedy v této implementaci následující:

- Front end webové aplikace - spuštění pomocí npm, příkaz „npm run serve“
- Back end webové aplikace - spuštění pomocí npm, příkaz „npm run start“
- Apache server - pomocí aplikace Xampp, stačí spustit Apache
- Archivační aplikace - přes jar soubor
- Validační aplikace - přes jar soubor
- Decryptor - přes jar soubor

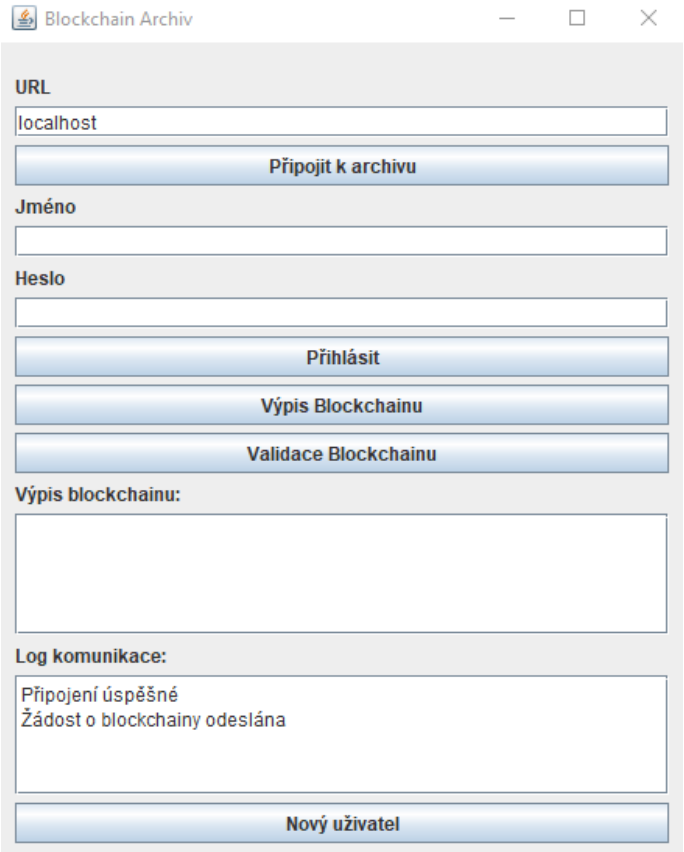
Po spuštění webové aplikace, Apache serveru a archivační aplikace je možné přejít k demonstračnímu scénáři. Prvním krokem je tedy otevření Validační aplikace. V hlavním okně, které se ukáže, pak validátor zadá základní údaj archivu - jeho URL. V tomto případě, stačí zadat „localhost“, protože vše běží v rámci jednoho počítače. Po stisknutí tlačítka **Připojit k archivu** začne proces navázání spojení. Po krátké chvíli se v **logu komunikace** objeví informace o úspěšném připojení. Tuto situaci lze vidět na obrázku 4.13

Na pozadí si aplikace zažádá o blockchainy ostatních připojených validátorů tak, aby měla nejaktuálnější verzi blockchainu. V tomto případě je jediná připojená, a tak si ponechá svůj vlastní blockchain, který získala deserializací uloženého.

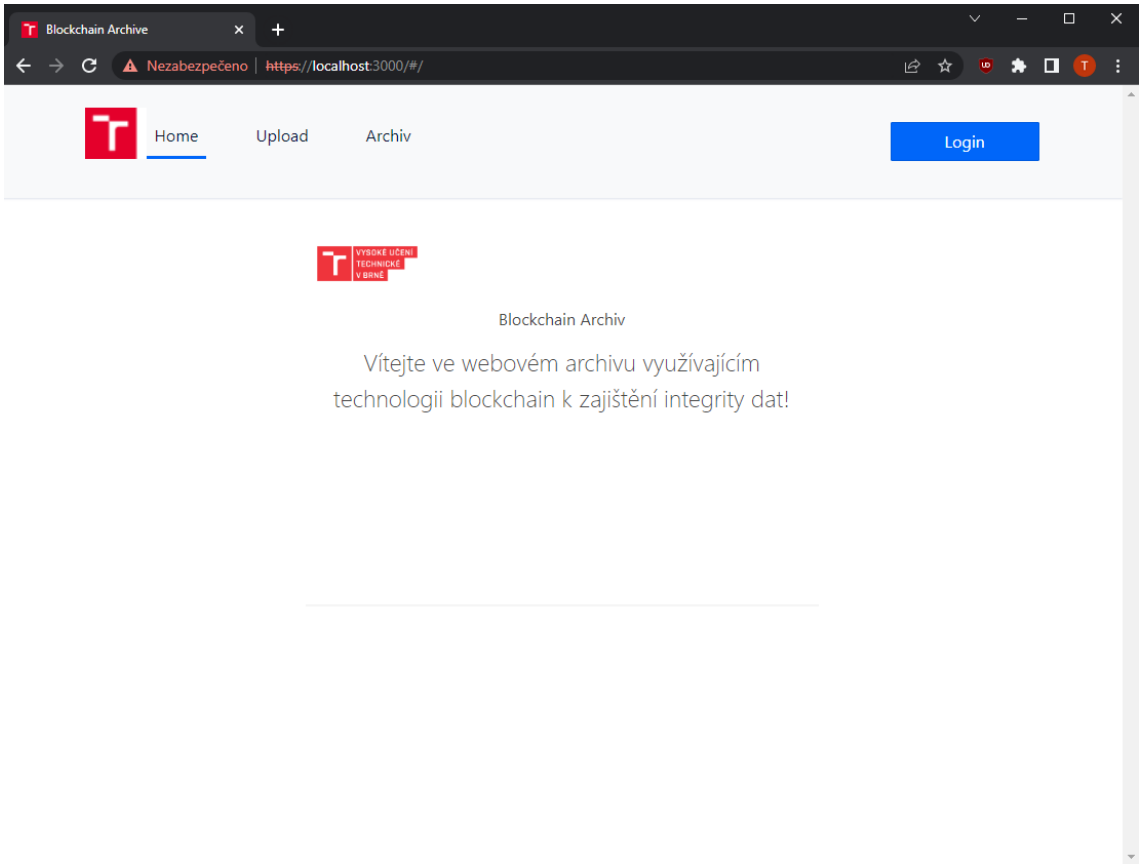
Pro kontrolu si teď validátor vypíše blockchain stisknutím tlačítka **Výpis blockchainu**. Ve výpisu 4.1 lze vidět textovou verzi blockchainu, která momentálně obsahuje jeden blok. Dokument v tomto bloku mimo jiné obsahuje dva soubory obsahu.

Mezitím si jiný uživatel otevře prohlížeč, do kterého zadá adresu archivu „https://localhost:3000“. Je nutné zadat jak https, tak port 3000. Jak jde vidět na obrázku 4.14 Dostane se na hlavní stránku webové aplikace. Je vidět, že není přihlášený, protože v pravém horním rohu vidí tlačítko login.

V navigační liště teď stiskne tlačítko **Upload**, ale jelikož není přihlášen, je webovou aplikací přesměrován na přihlášení pomocí Auth0, viz obrázek 4.15. Zde zadá svoje přihlašovací údaje a je úspěšně přihlášen. Automaticky se teď otevře stránka **Upload**, viz 4.16.



Obr. 4.13: Hlavní okno validační aplikace po připojení



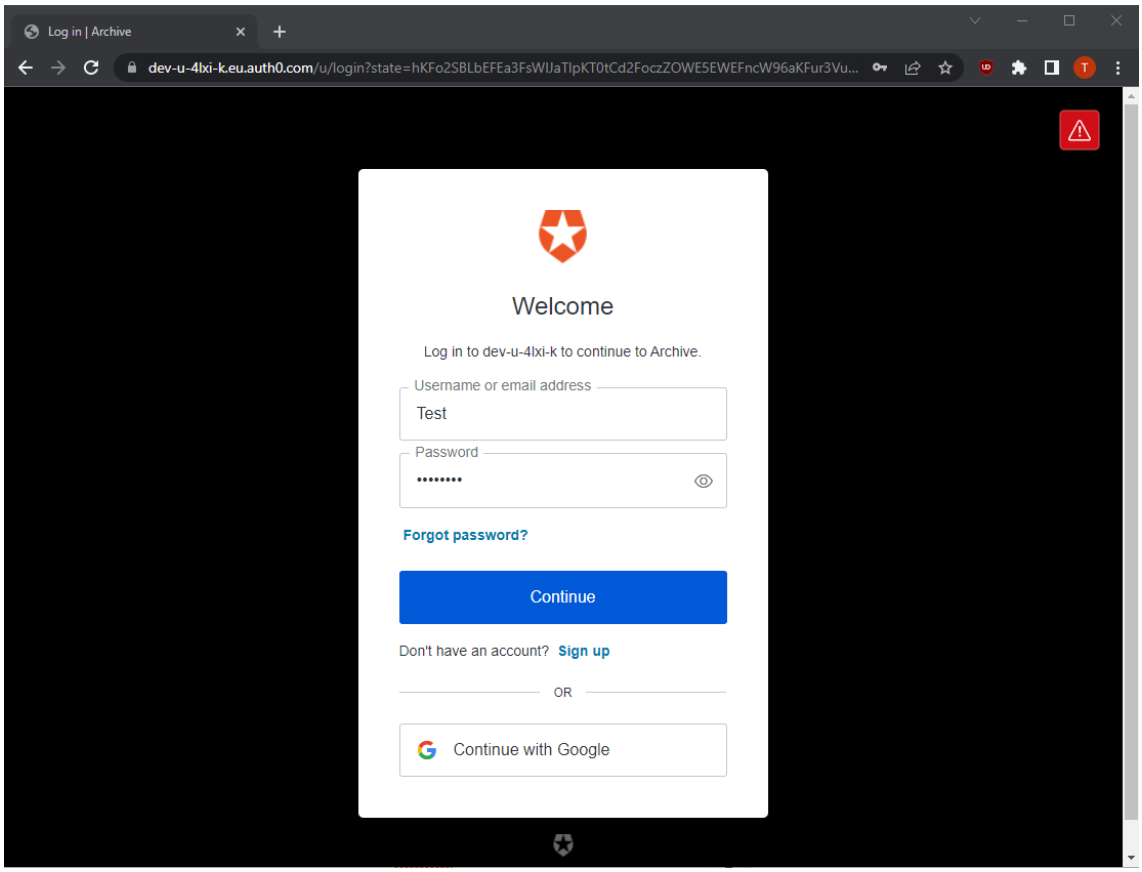
Obr. 4.14: Úvodní stránka webového archivu

Výpis 4.1: První výpis blockchainu

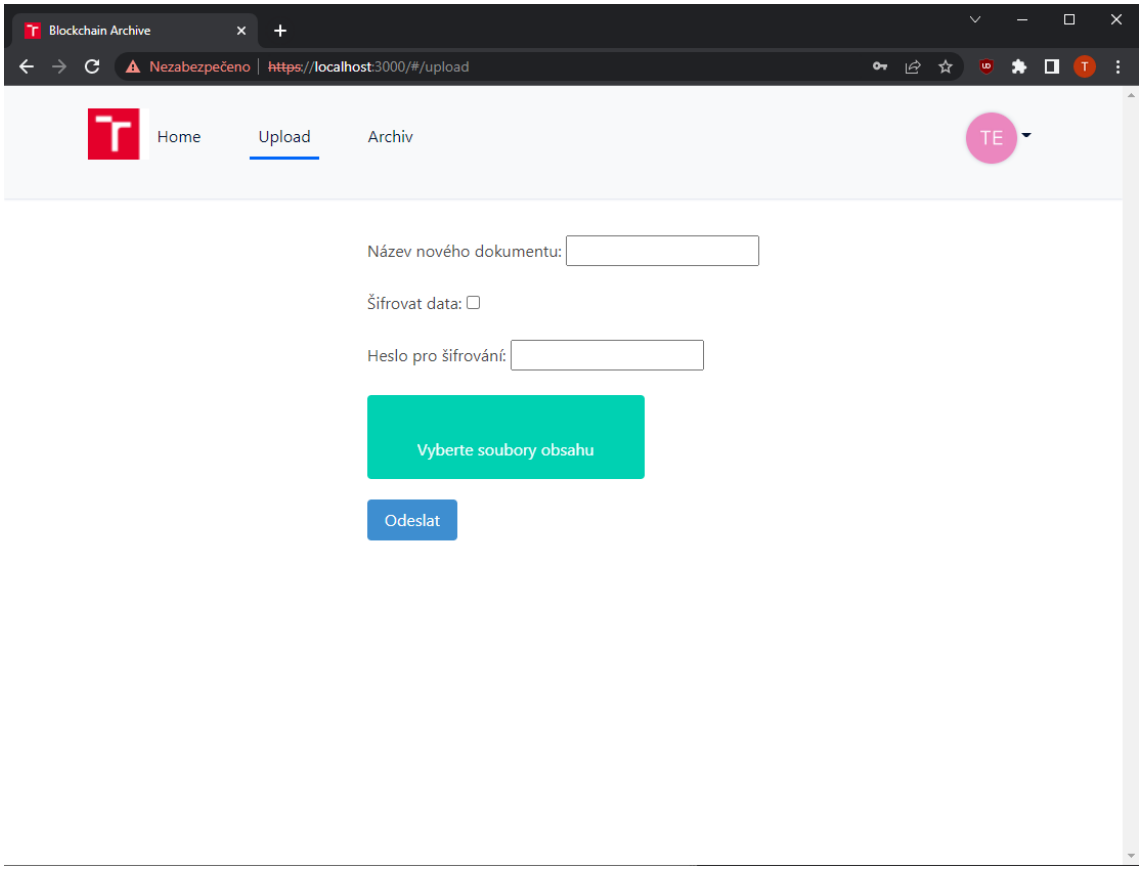
```
--------Blockchain---------

-------------------------
Block 0:
Předchozí hash: FIRST BLOCK
Název dokument: FirstDoc
Soubory obsahu:
FirstDocumentFile1.txt
FirstDocumentFile2.txt
Metadata: metadata.json
Hash souboru: 8
   a95a4bc83f7400b811fb66747238afc422f0703cce1c81b2058d434132f7daa
Digitální podpis: N8nY0nN4V+nDHGe3opFZ/91KrgmxNTgNw3m6mwQz4U/
   t7mqkYUdnZ68x/Bn9GUv+W0Fcbzm497p3uVl4TK4v6OVZt3jo34nG0h7+
   y56FKWNwn1bFyADOfihEHg0Y1xiJQhul/sphnMQbK1cJJYs+
   LgIMfHc1O2OGLx0pJcASlfXjr+
   DuLvngdbhDDF9KmRzP79i4TxNWs2q4uubhNR8YYjcC5RAjcTq7Yq4MS/
   O6N1mGO8mjrd5NOuzoqNQPrLgEdJ1DcoN2gmWdF74JwCn19jslSvumAW/3l+2
   MJWi0Z6YfkUPlDVxJ9CFPhXNlP11mQQO6I8jTDB/+sBJsHEHt+Q==
Veřejný klíč: Sun RSA public key, 2048 bits
  params: null
  modulus:
     18579903563192109986483987447592197997714587024706703479271836...
      public exponent: 65537
Časové razítko: 14:05:09 20. 05. 2022
ID bloku: 0
Hash tohoto bloku:
   dc260039fbe82ea4b979dff07b70e1c86438ea9a6c514ecce96f34065182cf10
----------End of Block------------
```

Lze vidět základní formulář pro upload nového dokumentu. Uživatel zadá jméno dokumentu a zatrhne políčko **Šifrovat**. Dále klikne na tlačítko **Vyberte soubor obsahu**. Otevře se okno průzkumníka souborů a uživatel vybere soubory, který chce nahrát. Ty se objeví v seznamu vybraných souborů, avšak jméno jednoho z nich je začervenalé a vedle je napsáno, že je jeho velikost příliš veliká, viz obrázek 4.17a. Soubor je tedy odstraněn z výběru kliknutím na křížek a nahrán je pouze jeden soubor.

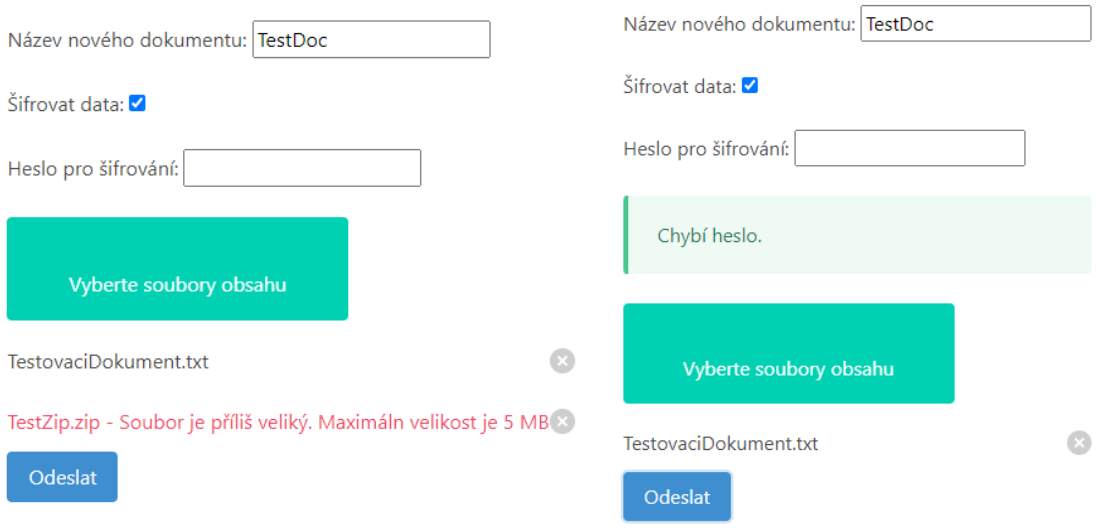
Uživatel klikne na tlačítko **Odeslat**, ale objeví se hláška „Chybí heslo“, protože zapomněl vyplnit heslo k šifrování, viz obrázek 4.18. Po zadání hesla, znovu stiskne



Obr. 4.15: Přihlašovací stránka Auth0



Obr. 4.16: Formulář na upload nového dokumentu

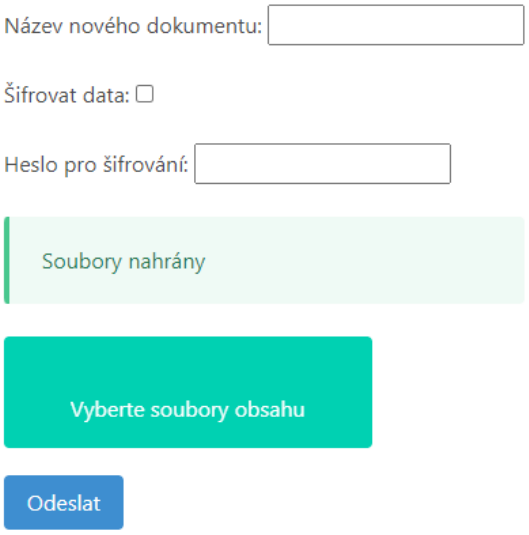


(a) Soubor je příliš velký

(b) Heslo nebylo vyplněno

Obr. 4.17: Výřez formuláře za určitých podmínek

Odeslat a tentokrát je nahrání dokumentu úspěšné, objeví se hláška „Soubory nahrány“ a formulář je resetován.

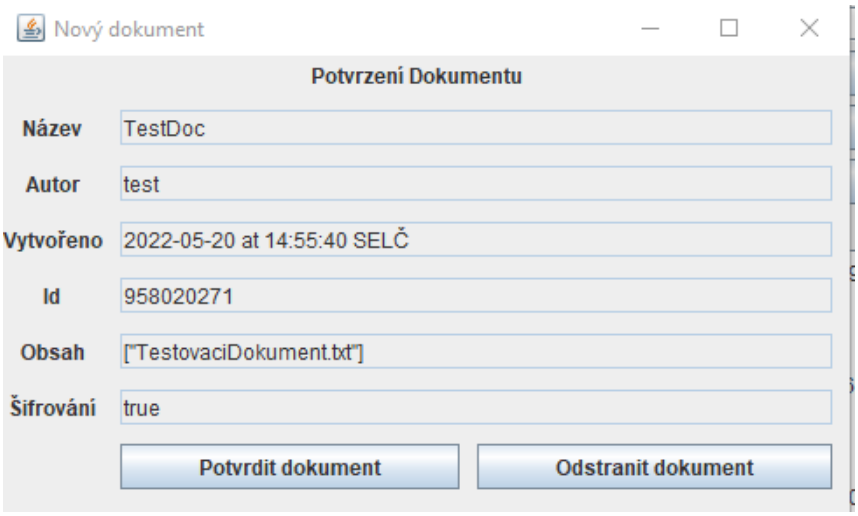


Obr. 4.18: Výřez formuláře po úspěšném odeslání souboru

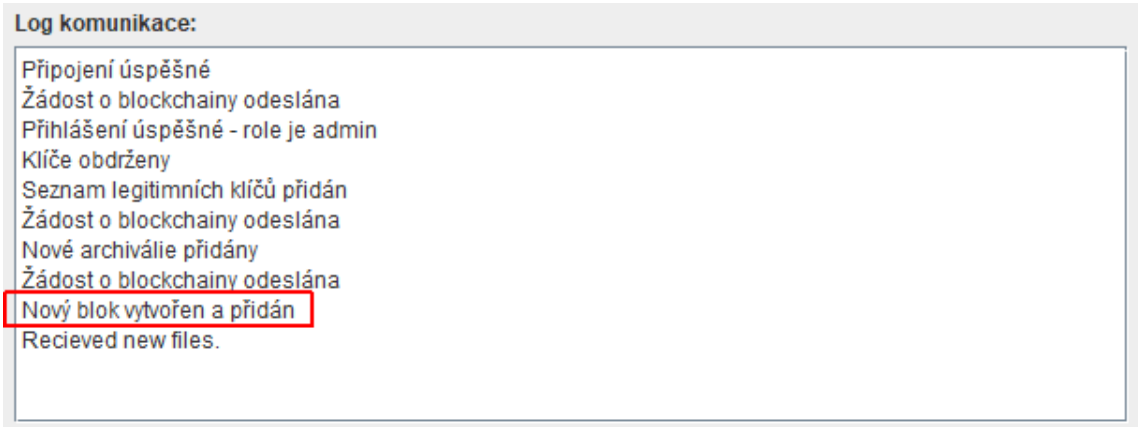
Téměř okamžitě po nahrání souborů k validátorovi dojde žádost o potvrzení a vyskočí na něj nové okno s názvem **Nový dokument**, viz obrázen 4.19. V něm jsou vypsána metadata souboru a seznam souborů. Validátor tento dokument potvrdí stiskem tlačítka **Potvrdit dokument**.

V logu komunikace lze vidět, že aplikace vytvořila nový blok obsahující tento dokument, viz obr. 4.20. Znovu spustí výpis blockchainu, viz výpis 4.2, kde je tento blok vidět jako poslední. Celkově tak teď blockchain má dva bloky.

Validátor ještě pro jistotu spustí validaci blockchainu stisknutím tlačítka **Validace Blockchainu**. Na pozadí aplikace provede veškerou validaci a zobrazí ji ve výpisu, viz 4.3. Je vidět, že integrita ani jednoho z dokumentů není porušena, a celý archiv je tak považován za neporušený.



Obr. 4.19: Validační aplikace - potvrzení nového dokumentu



Obr. 4.20: Log komunikace

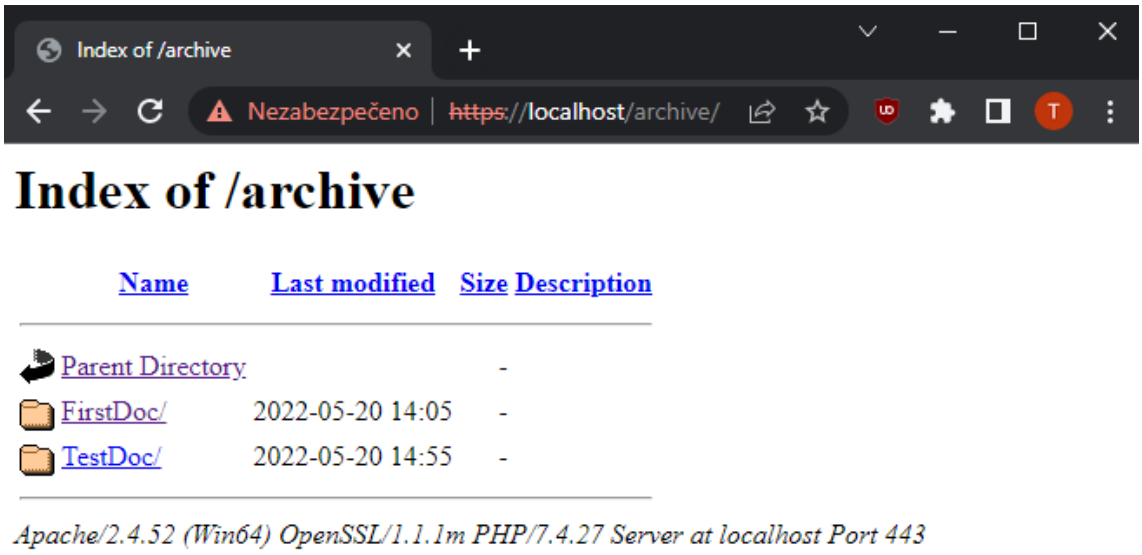
Uživatel webové aplikace v navigační liště klikne na tlačítko **Archiv**, které ho přesměruje na stránku výpisu archivu, viz obrázek 4.21. Vidí zde jednotlivé dokumenty reprezentované složkami. Najde nově nahraný dokument a klikne na něj, viz obrázek 4.22. Uvnitř vidí dva soubory - obsah a soubor s názvem „metadata.json“. Nejdříve si může prohlídnout soubor s metadaty, který mu prohlížeč vypíše, viz obrázek 4.23.

Výpis 4.2: Druhý výpis blockchainu (zredukovaný výpis)

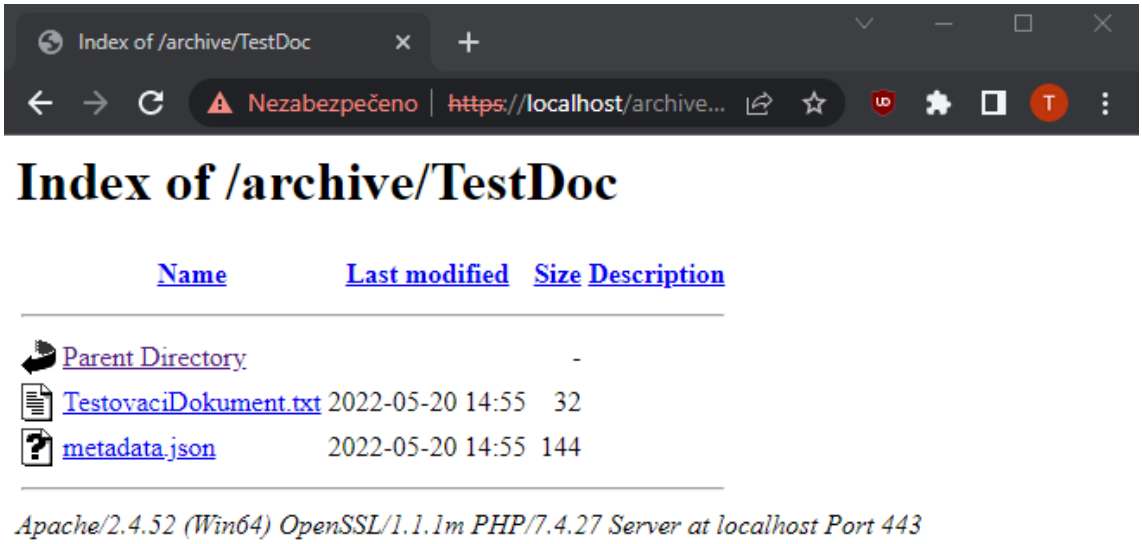
```
---------Blockchain---------

-------------------------
Block 0:
Předchozí hash: FIRST BLOCK
Název dokument: FirstDoc
[...]
Hash tohoto bloku:
   dc260039fbe82ea4b979dff07b70e1c86438ea9a6c514ecce96f34065182cf10
----------End of Block------------

-------------------------
Block 1:
Předchozí hash:
   dc260039fbe82ea4b979dff07b70e1c86438ea9a6c514ecce96f34065182cf10
Název dokument: TestDoc
Soubory obsahu:
TestovaciDokument.txt
Metadata: metadata.json
Hash souboru: 096
   e7cf46f666212010d0b54fbbc741e65b3b8c7745258506a9bc7e31782ad84
Digitální podpis: [...]
Časové razítko: 15:02:26 20. 05. 2022
ID bloku: 1
Hash tohoto bloku: 4
   ad9f15eaea69781c43765ddc1e30c175072fa51d83c5061682f0a134cf15ddd
----------End of Block------------
```

Výpis 4.3: Výpis validace blockchianu v případě neporušeného archivu

```
VYPIS VALIDACE BLOCKCHAINU
--------------------
Dokument FirstDoc
Hash je validní.
Podpis souboru FirstDocumentFile1.txt je validní.
------------------------
Dokument TestDoc
Hash je validní.
Podpis souboru TestovaciDokument.txt je validní.
------------------------
VÝSLEDEK: Blockchain validní, integrita archivu neporušena
```



Obr. 4.21: Výpis archivu ve webovém prohlížeči

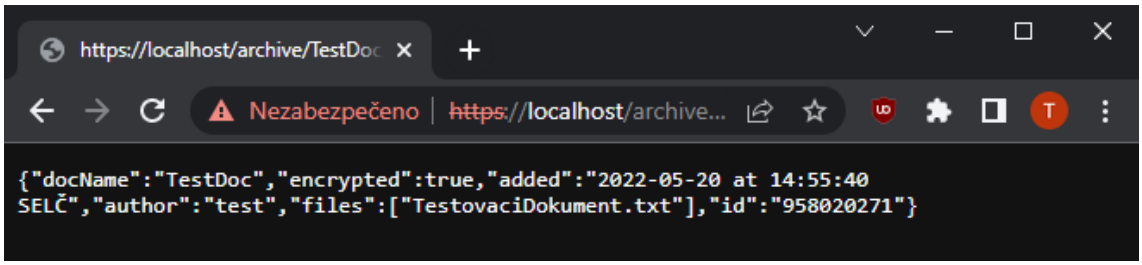


Obr. 4.22: Detail dokumentu ve výpisu archivu

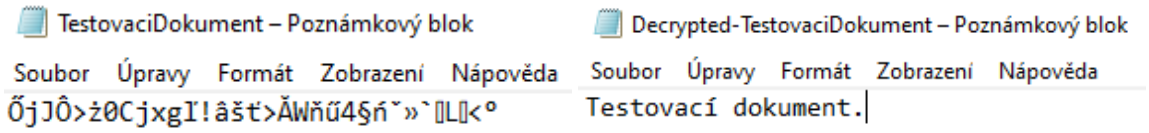
Když se pokusí otevřít soubor s obsahem, prohlížeč to nedokáže, protože je soubor zašifrovaný. Místo toho je stažen na místní disk. Když si jej uživatel otevře teď, lze vidět že obsah je zašifrovaný, viz obrázek 4.24a. Spustí tedy aplikaci **Decryptor**, viz obr. 4.25. Stisknutím na tlačítko **Načíst soubor** se otevře dialog pro výběr souboru. Uživatel nalezne stáhnutý zašifrovaný soubor a vybere ho. Následně zadá heslo a stiskne tlačítko **Dešifrovat**. Aplikace oznámí úspěšné dešifrování.

Uživatel se podívá do složky, kde byl uložen zašifrovaný soubor a uvidí vytvořený dešifrovaný soubor. Po jeho otevření již vidí původní obsah, viz 4.24b.

Tento scénář pokrývá naprostou většinu interakcí s archivačním systémem a je



Obr. 4.23: Výpis souboru metadata.json ve webovém archivu



(a) Zašifrovaný obsah

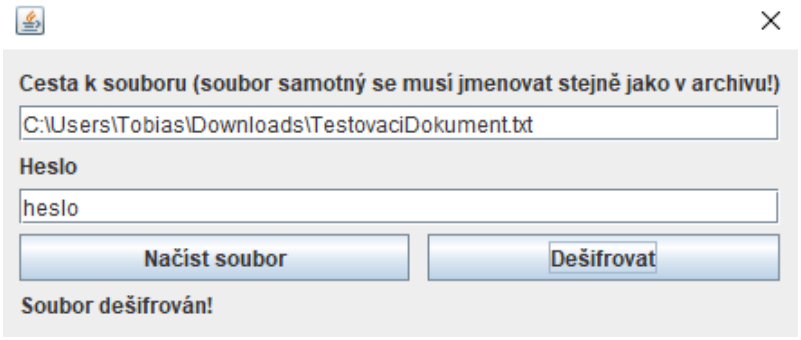
(b) Dešifrovaný obsah

Obr. 4.24: Stažený soubor před a po dešifrování

vidět, že systém tento typ interakcí umí zpracovat. V neposlední řadě je však ještě potřeba otestovat některé pokročilejší funkce a ověřit bezpečnost tohoto archivu.

4.8 Testování bezpečnosti aplikace

V této sekci budou provedeny tři testy zaměřené na bezpečnostní aspekty aplikací. Prvním bude test detekce porušení integrity, druhý test demonstruje šíření blockchainu mezi více validátorů za přítomnosti nečestného uživatele a třetí test dokáže důvěrnost dat během jejich přenosu.



Obr. 4.25: Aplikace Decryptor

4.8.1 Detekce porušení integrity

Test bude velmi jednoduchý. Výchozí stav bude situace po dokončení demonstrace - tedy v archivu jsou dva dokumenty. K archivačnímu systému je připojen jeden validátor. Na začátku testu provede validaci. Jelikož se nic neměnilo, výpis je identický s výpisem 4.3.

Z výpisu lze vidět, že integrita nebyla porušena. Následně bude simulováno porušení integrity. V podstatě nezáleží na tom, jestli by bylo způsobeno v důsledku chyby nebo v důsledku nějakého útoku. Obsahem souboru FirstDocumentFile1, který se nachází v dokumentu FirstDoc, je text „This is the first document“. Ten je změněn na „his is the first document“. Došlo tedy k odstranění prvního symbolu.

Validátoři znovu provedou validaci archivu a porušení integrity je okamžitě zachyceno, viz výpis 4.4. Zároveň aplikace vypíše, které dokumenty byly porušeny.

Výpis 4.4: Výpis validace blockchianu v případě porušení integrity

```
VYPIS VALIDACE BLOCKCHAINU
--------------------
Dokument FirstDoc
Hash souboru:
 8a95a4bc83f7400b811fb66747238afc422f0703cce1c81b2058d434132f7daa
měla by být:
 076556e0b3b8090e38e722b919610e2613f18c6a60edc283c20bb27b804510a7
HASH NENÍ VALIDNÍ - INTEGRITA PORUŠENA!
Podpis: N8nY0nN4V+nDHGe3opFZ/91KrgmxNTgNw3m6mwQz4U/t7mqkYUdnZ68x/
   Bn9GUv+W0Fcbzm497p3uVl4TK4v6OVZt3jo34nG0h7+
   y56FKWNwn1bFyADOfihEHg0Y1xiJQhul/sphnMQbK1cJJYs+
   LgIMfHc1O2OGLx0pJcASlfXjr+
   DuLvngdbhDDF9KmRzP79i4TxNWs2q4uubhNR8YYjcC5RAjcTq7Yq4MS/
   O6N1mGO8mjrd5NOuzoqNQPrLgEdJ1DcoN2gmWdF74JwCn19jslSvumAW/3l+2
   MJWi0Z6YfkUPlDVxJ9CFPhXNlP11mQQO6I8jTDB/+sBJsHEHt+Q==
PODPIS NENÍ VALIDNÍ - INTEGRITA PORUŠENA!
------------------------
Dokument TestDoc
Hash je validní.
Podpis souboru TestovaciDokument.txt je validní.
------------------------
VÝSLEDEK: Blockchain není validní, integrita archivu porušena!
Porušené dokumenty:
FirstDoc
```

4.8.2 Blockchain a nalezení konsenzu

Druhý test bude složitější. Půjde o simulaci stavu, kdy útočník získá kontrolu nad jedním účtem validátora s právy admin. Zároveň se mu podaří upravit některý z archivovaných dokumentů. Vytvoří si vlastní blockchain, který bere tuto změnu v potaz. Následně se připojí nový, čtvrtý, validátor a zažádá ostatní o jejich blockchainy. Pokud vše funguje tak, jak má, měl by přijmout původní blockchain, a ne ten útočníkův.

Pro jednoduchost bude archiv obsahovat jeden dokument. Předem byl vytvořen serializovaný blockchain, který obsahuje hash upraveného (tedy porušeného) souboru. Ten si načte jeden z validátorů - útočník. Výchozí stav je tedy - archiv s jedním dokumentem, dva validátoři s validním blockchainem, jeden validátor s nevalidním blokckchainem. K porovnání samotných blockchainů lze vzít hodnotu posledního (v tomto případě jediného) bloku. Ta je vždy unikátní pro daný blockchain. Stav lze tedy popsat následnově:

- Obsah souboru - „Testovací dokument.“
- Validátor #1 - Hash: „b3eaa7ce4bd34fd0f1b5e10dcb41d3b6fc662520a3bc0c5c4398044011292a46“ Blockchain validní.
- Validátor #2 - Hash: „b3eaa7ce4bd34fd0f1b5e10dcb41d3b6fc662520a3bc0c5c4398044011292a46“ Blockchain validní.
- Útočník - Hash: „0111cc6e14350606e1e5ad3c3a62929cb58bbfc9c81ba172184717766435faa7“ Blockchain nevalidní.

První krok je úprava souboru v archivu. Podobně jako u předchozího testu je upraven obsah změnou jednoho symbolu. Pro jistotu je provedena validace. U čestných uživatelů se ukáže, že integrita byla porušena. Útočník však má upravený blockchain a z jeho pohledu je integrita v pořádku. Stav je tedy následující

- Obsah souboru - „Restovací dokument.“
- Validátor #1 - Hash: „b3eaa7ce4bd34fd0f1b5e10dcb41d3b6fc662520a3bc0c5c4398044011292a46“ Blockchain nevalidní.
- Validátor #2 - Hash: „b3eaa7ce4bd34fd0f1b5e10dcb41d3b6fc662520a3bc0c5c4398044011292a46“ Blockchain nevalidní.
- Útočník - Hash: „0111cc6e14350606e1e5ad3c3a62929cb58bbfc9c81ba172184717766435faa7“ Blockchain validní.

Následně se připojí nový validátor a přihlásí se. V tom momentu zažádá o blockchain od ostatních. Ti mu pošlou svoji verzi, a pokud systém funguje tak, jak má, měl by přijmout ten, který vlastní čestní validátoři, protože jich je více. Po výpisu lze vidět, že je tomu tak, a i nový validátor tak ví, že archiv má porušenou integritu.

- Obsah souboru - „Restovací dokument.“
- Validátor #1 - Hash: „b3eaa7ce4bd34fd0f1b5e10dcb41d3b6fc662520a3bc0c5c

4398044011292a46“ Blockchain nevalidní.

- Validátor #2 - Hash: „b3eaa7ce4bd34fd0f1b5e10dcb41d3b6fc662520a3bc0c5c 4398044011292a46“ Blockchain nevalidní.
- Útočník - Hash: „0111cc6e14350606e1e5ad3c3a62929cb58bbfc9c81ba17218471 7766435faa7“ Blockchain validní.
- Nový validátor - Hash: „b3eaa7ce4bd34fd0f1b5e10dcb41d3b6fc662520a3bc0c5c 4398044011292a46“ Blockchain nevalidní.

Bylo tedy dokázáno, že i pokud se útočníkovi podařilo porušit integritu některého z dokumentů a vytvořit si vlastní blockchain, nedokáže se mu ho rozšířit mezi většinu ostatních validátorů. Průnik je tedy vždy detekován a validátoři vždy dokáží najít konsenzus nad jedním blockchainem.

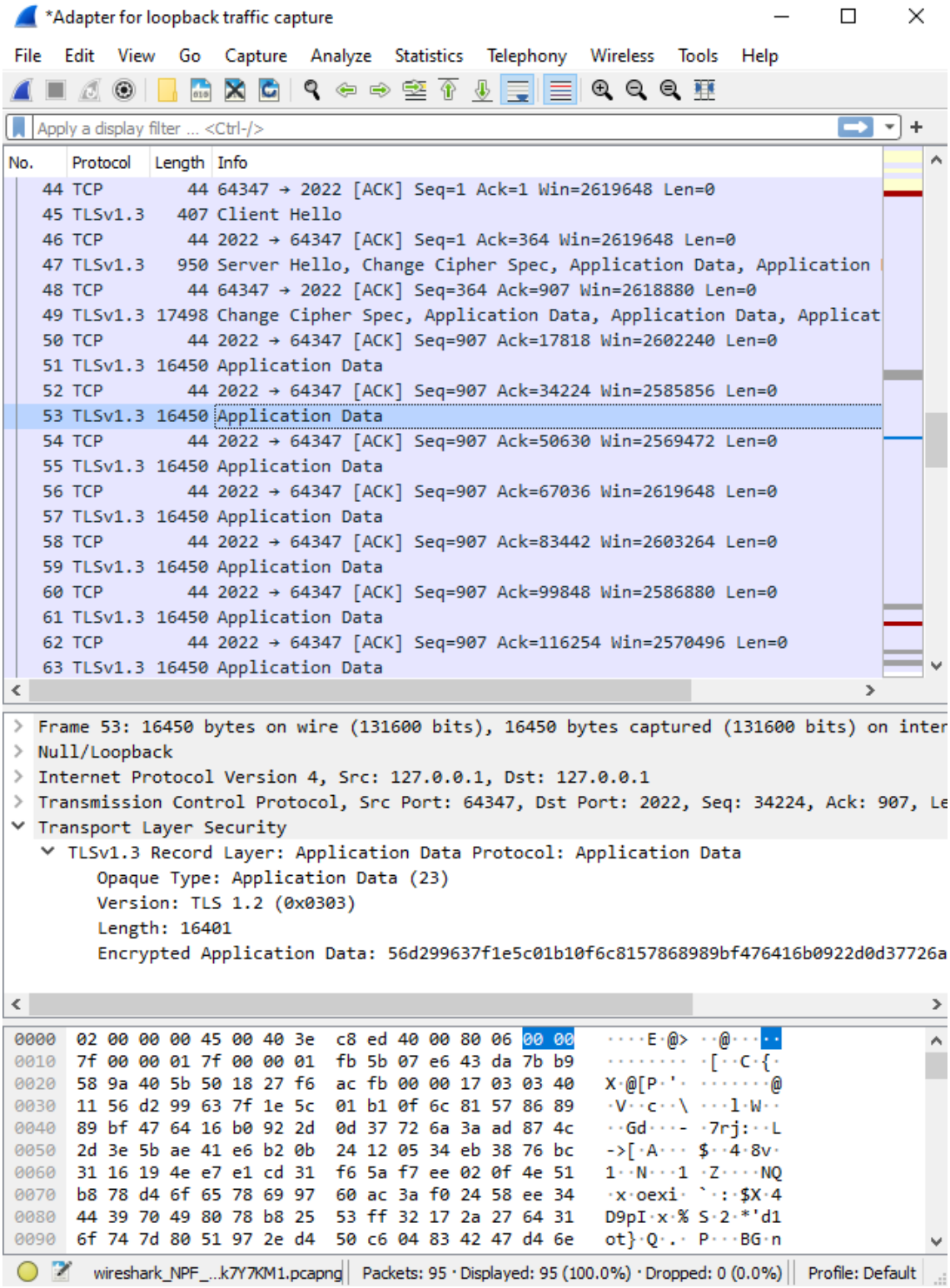
4.8.3 Důvěrnost dat během přenosu

K poslednímu testu bude potřeba odposlouchávat komunikaci, k čemuž poslouží program Wireshark. Pomocí něj bude zachytáván provoz na rozhraní loopback. Pointou je prokázat důvěrnost přenášených dat. Záchyty budou čtyři - při uploadu souboru, při procházení archivu, dále při komunikaci mezi Validační a Archivační aplikací a při validaci blockchainu. Všechny čtyři typy komunikace by měly být chráněny protokolem TLS a nemělo by být možné číst obsah této komunikace. V záchytu tak pouze bude vidět komunikace mezi porty avšak obsah bude skryt.

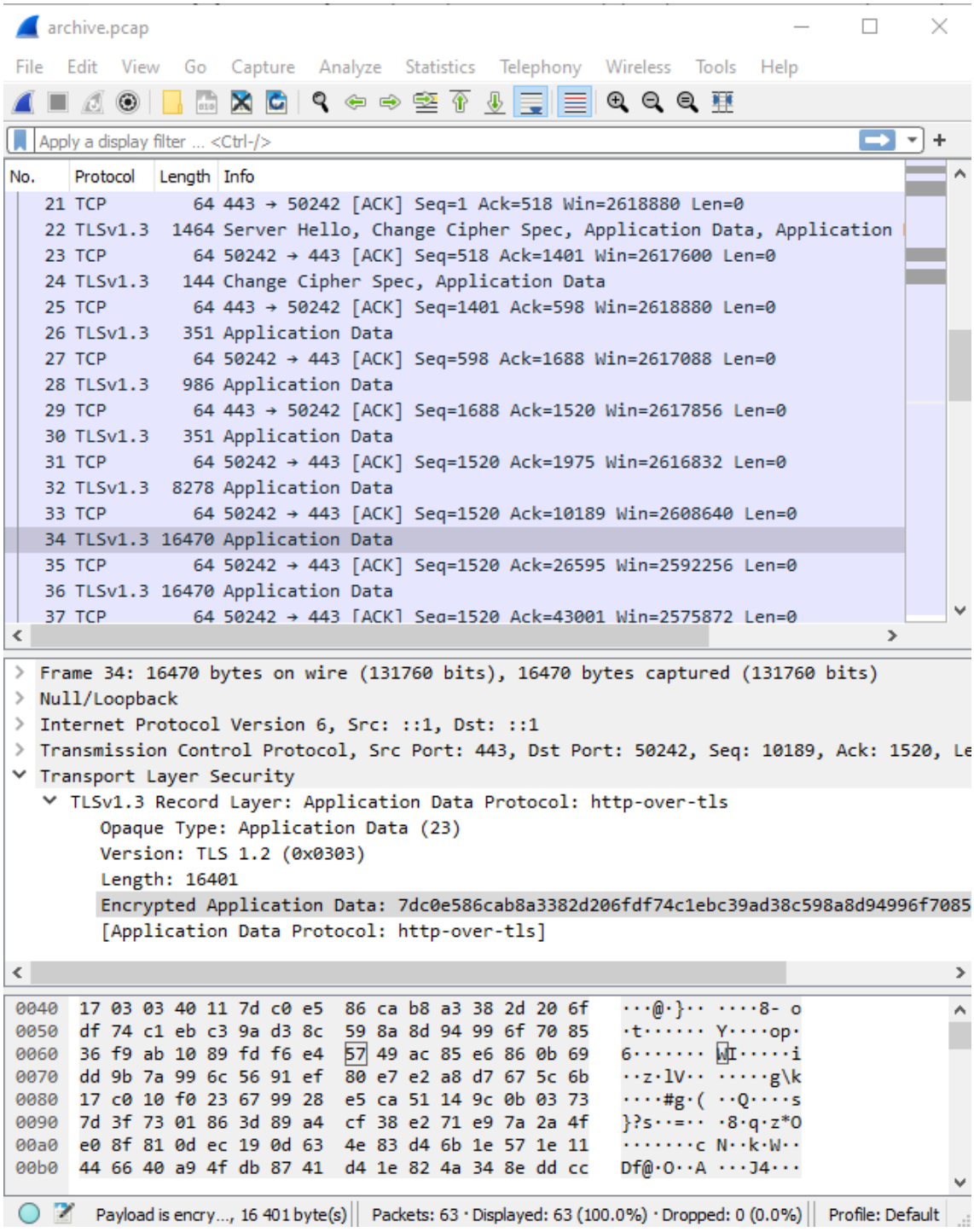
Jako první bude zachycena komunikace mezi uživatelem a back endem webové aplikace při uploadu nějakého souboru. Na obrázku 4.26 lze vidět výpis komunikace po spuštění uploadu. Hned na začátku dojde k navázání Uživatel je připojen portem 64347, zatímco back end běží na portu 2022. Lze vidět, že směrem od uživatele (src. port je 64347 a dst. port je 2022) proudí pakety o velikost 16450 bytů zašifrované protokolem TLS. Ve spodní části obrázku lze vidět detail jednoho z paketů.

Druhý záchyt, který lze vidět na obrázku 4.27, zobrazuje situaci uživatele procházejícího archiv a následné otevření jednoho ze souborů. Apache server, přes který uživatel prochází archiv, běží na portu 443, zatímco uživatelův port je v tomto případě 50242. Lze vidět pakety proudící z Apache serveru k uživateli (src. port je 443 a dst. port je 50242). Obsah je opět šifrován.

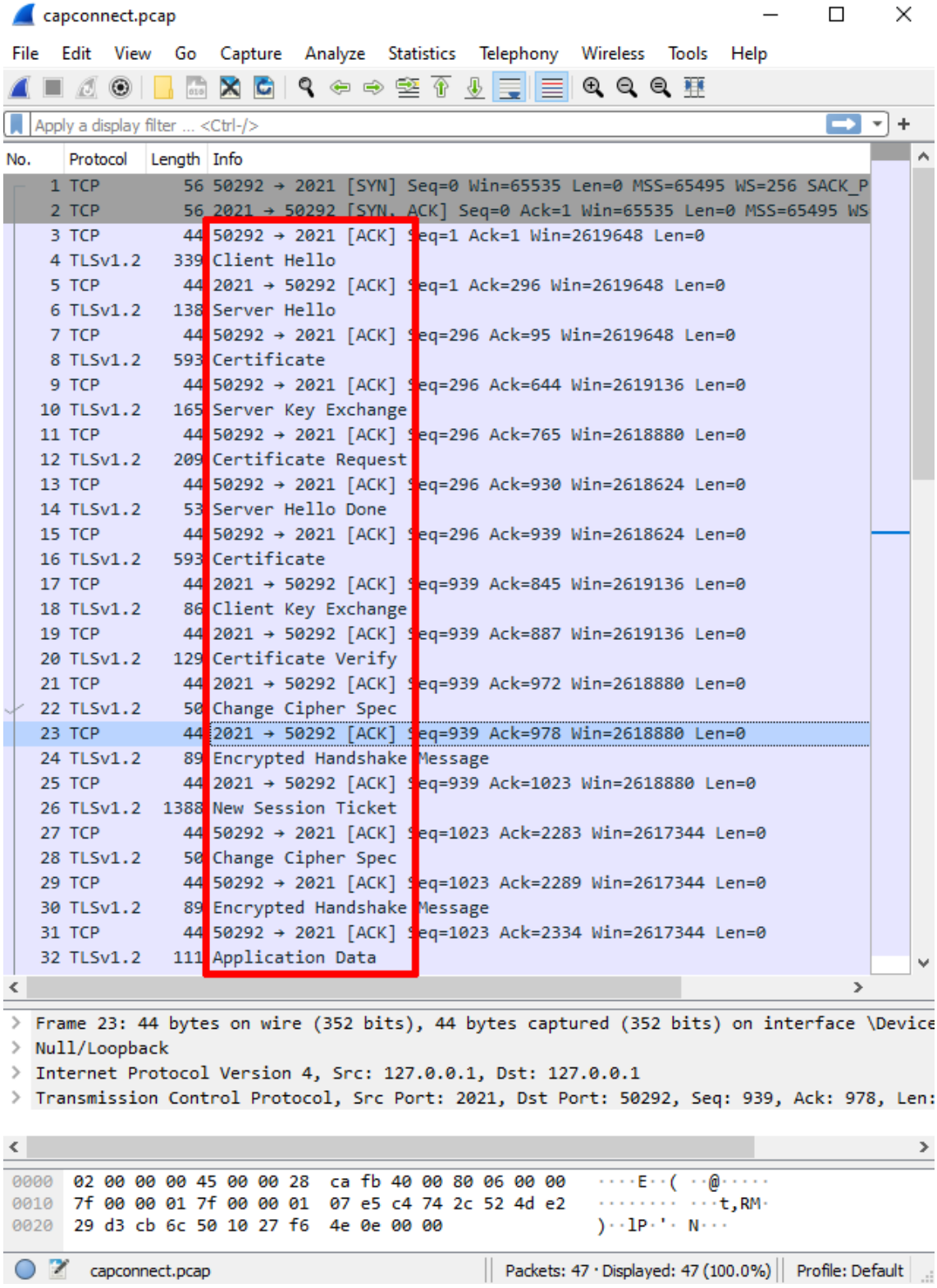
Třetí záchyt zobrazuje komunikaci mezi Validační aplikací a Archivační aplikací. Port validátora je 50292, Archivační aplikace běží na portu 2021. Na prvním obrázku 4.28 lze vidět situaci po stisknutí tlačítka **Připojit k serveru** ze strany validátora. Prvních zhruba 30 paketů ukazují navázání TLS spojení (a příslušné TCP ACK zprávy). Strany si vymění certifikáty, šifrovací klíče a provedou šifrovaný handshake. Na obrázku 4.29 pak lze vidět již zašifrovanou komunikaci, konkrétně pak šlo o přihlášení validátora k archivní aplikaci.



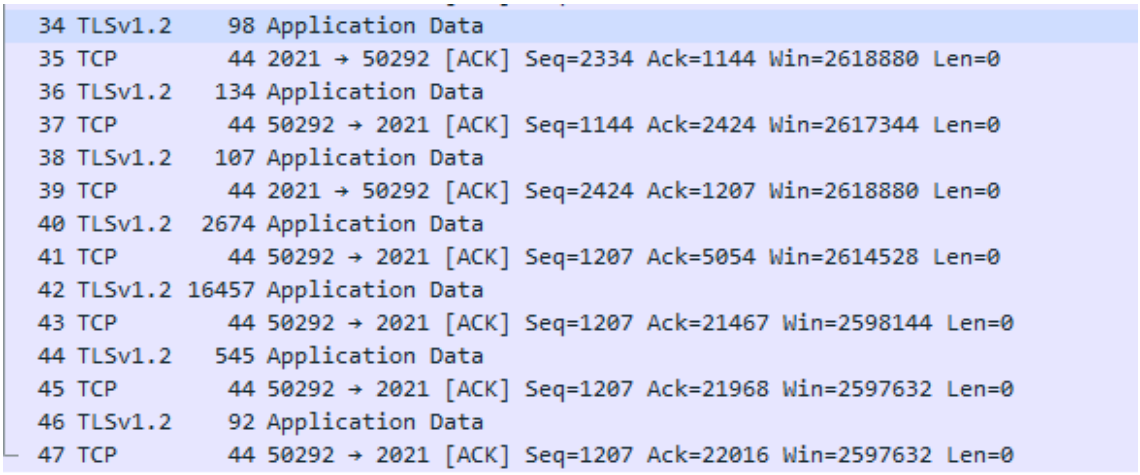
Obr. 4.26: Záchyt uploadu souborů



Obr. 4.27: Záchyt procházení archivu



Obr. 4.28: Záchyt navázání spojení mezi Validační aplikací a archivem



Obr. 4.29: Záchyt přihlášení šifrovaného pomocí TLS

Poslední záchyt ukazuje validaci blockchainu. Během té si validátor vytváří spojení s Apache serverem, ze kterého získá jednotlivé soubory aby mohl zjistit jejich integritu. Validátor má v tomto případě port 50372, zatímco Apache stále běží na portu 443. Na obrázku lze vidět navázání TLS spojení a následnou šifrovanou komunikaci. Nelze tedy číst obsah paketů.

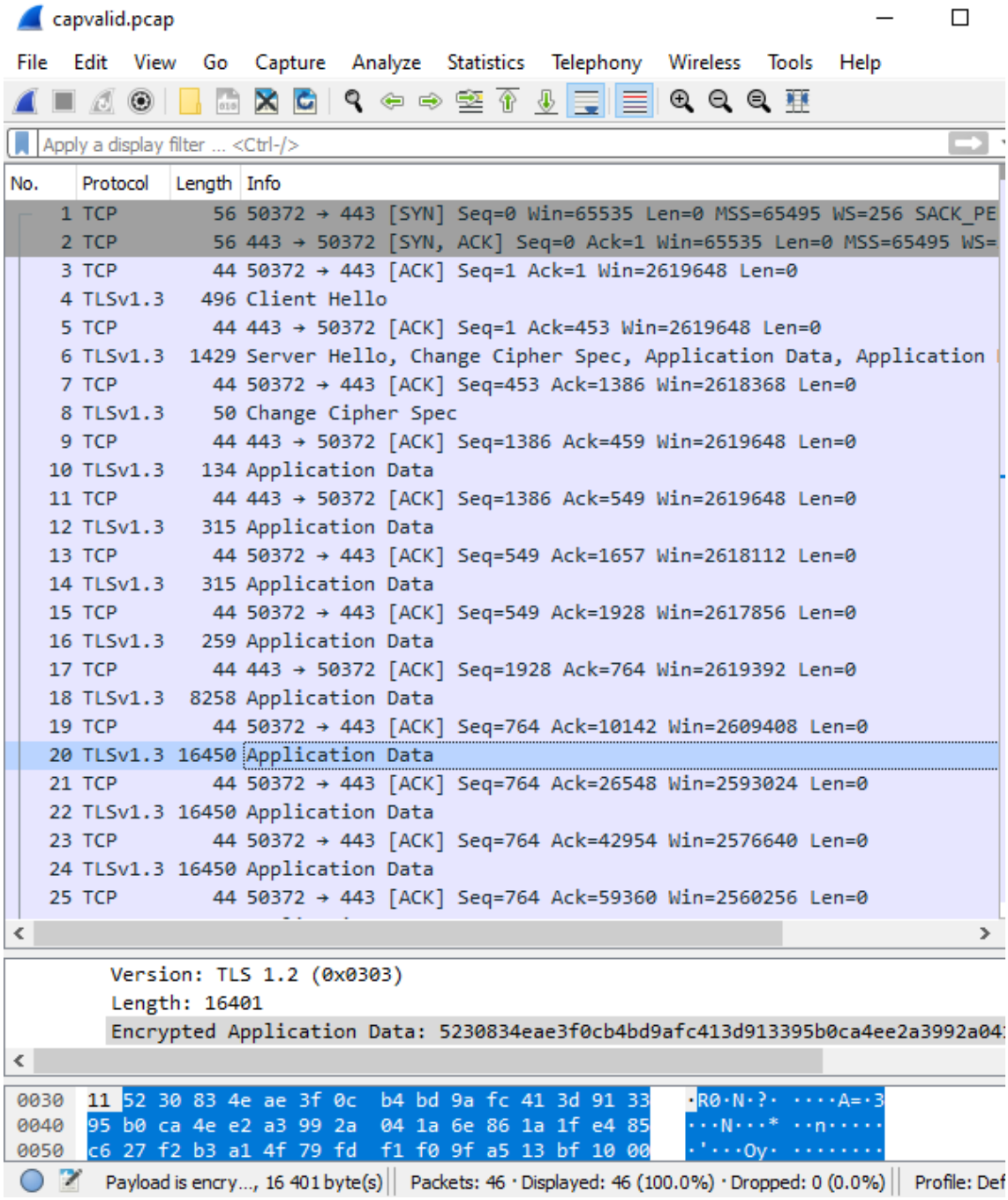
Tyto testy tedy dokázaly, že archiv chrání důvěrnost přenášených dat pomocí šifrování na všech úrovních komunikace.

4.9 Hardwarové a softwarové požadavky

Hardwarově není aplikace příliš náročná. Kód samotný je ve své podstatě jednoduchý a nepotřebuje vysoký výpočetní výkon. Jediným aspektem, který je potřeba zohlednit, je volné místo, které archiv bude potřebovat. Zde se budou požadavky výrazně lišit v závislosti na dané implementaci, respektive účelu, pro který je archiv určen. Pokud je archiv určen pro pouze textové dokumenty a je využíván malou firmou, dá se předpokládat, že nebude vyžadovat více než jednotky, maximálně desítky, gigabytů místa. Pokud by se jednalo o archiv multimédií v nějaké státní instituci, požadavky na místo by se jistě dostaly řádově do terabytů. Zvláště pokud by navíc bylo implementováno zálohování.

Ani uživatelé webové aplikace a validátoři nemusí počítat s vysokými nároky na hardware, avšak jsou zde některá softwarová omezení. Konkrétně jde o požadavek na nainstalovaný software Java, konkrétně pak ve verzi 15, ve které byla aplikace naprogramována. Java je potřeba pro spuštění Validační aplikace a aplikace Decryptor. K přístupu k webové aplikaci je pak potřeba webový prohlížeč.

Na serveru, kde bude běžet back end a archivační aplikace je softwarových



Obr. 4.30: Záchyt validace blockchainu

požadavků více. Implementace byla provedena a testována v operačním systému Windows 10, ale neměl by být problém zprovoznit aplikace i na Linuxovém prostředí. Opět je vyžadována Java 15, dále software Xampp, pomocí kterého lze jednoduše nastavit a spustit Apache server, a dále musí mít prostředí nainstalovaný manager „npm“, pomocí kterého se spouští back end a front end webové aplikace.

4.9.1 Úroveň bezpečnostni a dlouhodobá archivace

Dá se říct, že úroveň zabezpečení archivovaných dat je na poměrně vysoké úrovni, protože jsou implementovány všechny důležité bezpečnostní aspekty ukládání dat. Archivace však navíc specificky vyžaduje dlouhodobou bezpečnost, a je tedy otázka do jaké míry tato implementace dlouhodobé zabezpečení zajišťuje.

Na jednu stranu neexistuje důvod, proč by celý systém nemohl běžet donekonečna a konstantně tak zajišťovat bezpečnost. Na druhou stranu se nelze spoléhat na to, že po desetiletí nenastane moment, kdy nebude připojený jediný validátor nebo kdy na delší dobu nespadne Archivační aplikace. Zvláště u menších firem, které by implementaci využívaly, k takovým situacím jistě dojde.

Pravdou však je, že i kdyby se nějakou dobu blockchain nevalidoval, archiv by bezpečnostně spadl na úroveň klasických archivů. A vždy by existovala možnost se vrátit k validaci blockchainem, za předpokladu, že by někde existovala kopie uloženého blockchainu.

4.10 Výsledky implementace archivu

Návrh webového archivu využívajícího technologii blockchain byl implementován v téměř celém rozsahu. Sestává z několika částí, které spolu komunikují a celkově vytváří komplexní archivační systém. Ten splňuje všechny základní náležitosti digitálního archivu, zvláštní pozornost pak byla věnována zajištění integrity archivovaných dat. K tomu je využita technologie blockchain, která je zde implementována v míře v jaké byla navržena.

Všechny prvky implementace byly popsány v předchozích kapitolách včetně jejich funkčnosti a principu, na kterých fungují. Dále byla provedena demonstrace, která ukazuje všechny nejdůležitější aspekty fungování celého systému. Navíc bylo provedeno testování některých bezpečnostních prvků. Implementace zajišťuje zejména integritu dat ale neopomíjí jejich důvěrnost a zajišťuje řízení přístupu pro uživatele.

Z implementace bylo vypuštěno několik volitelných funkcí. Konkrétně je to verzování a odstraňování archivovaných dokumentů, dále archivace webových stránek zadáním jejich URL. Žádná z těchto funkcí není zásadní, jednalo se spíše o návrhy,

které by zvýšily uživatelskou přívětivost, jejich absence tedy příliš neovlivňuje výsledek práce.

4.10.1 Diskuze nad výsledky

Celkově je implementace spíše proof-of-concept než hotový produkt. Splňuje všechny náležitosti digitálního archivu a zadání, avšak se zpětným pohledem by určitě šly řešit některé aspekty lépe. Například by asi dávalo smysl přenést Validační aplikaci do webového prostředí a nejlépe ji sloučit s existující webovou aplikací. K tomu nedošlo z několika důvodů. Mimo jiné by byla implementace validace do podoby webové aplikace poměrně složitá vzhledem k menším zkušenostem s vývojem webových aplikací oproti desktopovým. Jedná se tak spíše o otázku uživatelské přívětivosti, z hlediska funkčnosti by neexistovaly rozdíly.

Poslední otázkou tak je, zda má technologie blockchain v této oblasti smysl. Na základě rešerše, návrhu i implementace se dá poměrně jasně odpovědět, že ano. Blockchain přináší velmi efektivní způsobny jakým dlouhodobě uchovávat informace, například o integritě dat. Práce TrustChain [29] navrhla způsob jak pomocí blockchainu řešit dočasnou platnost elektronických podpisů, práce BCLinked [30] navrhla způsob jakým použit privátní blockchain k archivací webových stránek, dále existují služby jako Filecoin, které nabízí decentralizované úložiště souborů.

Návrh v této práci pak kombinuje některé aspekty již existujících implementací a vytváří systém, kde distribuovaný blockchain slouží především k detekci porušení integrity archivu, což je asi jeho největším přínosem. Určitě se tak jedná o oblast, kterou lze dále rozvíjet.

Závěr

Tato diplomová práce se zabývala problematikou archivace dat a bylo prozkoumáno využití technologie blockchain k tomuto procesu. Hlavními cíli bylo provést teoretickou analýzu, navrhnout webový archivační systém a následně jej implementovat včetně všech důležitých náležitostí.

Nejprve došlo k teoretickému rozboru, kde byla analyzována problematika práce. Byly vysvětleny základní principy a technologie, které jsou využívány v oblasti bezpečnosti digitálních archivů. Dále byla teoreticky zanalyzována technologie blockchain.

Archivace dat je proces, který zajišťuje dlouhodobé ukládání nějakých digitálních dat. Každý digitální archiv musí splňovat určité podmínky a zajišťovat některé bezpečnostní aspekty. Těmito aspekty jsou důvěrnost dat, řízení přístupu, autenticita a integrita dat.

Blockchain je datová struktura založená na řetězení bloků s využitím hashovacích funkcí. V oblasti archivace ji lze využít mnoha způsoby, předchozí práce jej například použily k řešení problému vypršení platnosti elektronických podpisů. V této práci se pak pomocí blockchainu vytváří systém sloužící k detekci porušení integrity archivovaných dat.

Praktická část práce pak měla dvě hlavní části - návrh a implementaci. Nejdříve tedy došlo k navržení archivačního systému, který splňuje dané podmínky a implementuje technologii blockchain. Návrh dále bere v potaz i ostatní aspekty bezpečnosti - řízení přístupu a autentizace uživatelů je řešena několika způsoby včetně využití externích autentizačních platforem. Důvěrnost dat během přenosu je řešena implementací protokolu TLS, důvěrnost uložených dat pak umožňuje volitelná funkce šifrování.

Implementace sestává z několika částí. Za prvé je to Webová aplikace, která uživatelům nabízí jednoduché webové prostředí, skrze které mohou do archivu nahrávat nové dokumenty a poté tento archiv prohlížet. Zajišťuje také autentizaci uživatelů.

Druhou částí je takzvaná Archivační aplikace. Tato aplikace běží na serveru, kde se nachází archivovaná data a má několik úkolů. Za prvé tyto archivovaná data zpracovává poté co jsou uživateli nahrána přes webové prostředí. Mimo jiné tak zajišťuje důvěrnost v podobě volitelného šifrování uložených dat.

Dalším úkolem Archivační aplikace je komunikace s Validačními aplikacemi, třetí komponentou celého systému. Tyto Validační aplikace jsou využívány tzv. validátory, tedy uživateli, kteří mají na starosti blockchain a validaci archivovaných dat. Ti se musí u Archivační aplikace autentizovat a dále zajišťují autentičnost blockchianu pomocí elektronických podpisů.

Ve své podstatě je blockchain implementován tak, že po uploadu nového dokumentu do archivu, dostane některý z validátorů žádost o potvrzení daného dokumentu. Pokud tuto žádost potvrdí, dokument je definitivně archivován. Zároveň si validátor vytvoří nový blok, který obsahuje hash daného dokumentu. Tento blok si pak všichni validátoři přidají do svých blockchainů. Ty si mezi sebou validátoři kontinuálně posílají a dosahují tak konsenzu.

Validátoři pak mají možnost validovat integritu archivu hashováním uložených dat porovnáváním hodnot s těmi v blockchainu. Pokud pak dojde ke změně archivovaných dat, hashe se nebudou shodovat a změna je detekována. Poslední menší součástí práce je pomocná aplikace Decryptor, která slouží k dešifrování uložených dat.

Celý systém byl dopodrobna popsán a prezentován, dále byla demonstrována jeho funkčnost a otestovány bezpečnostní mechanismy. Ke konci práce pak došlo ke zhodnocení a krátké diskuzi. Vytyčené cíle tedy byly splněny.

Návrh obsahoval i další rozšiřující funkce, které nakonec implementovány nebyly, popřípadě došlo k jejich zjednodušení. Šlo například o možnosti přidávání nových verzí archivovaných dokumentů nebo jejich odstraňování. Aplikace tak má potenciál pro další rozvoj.

Problematika, kterou se práce zabývá je velmi aktuální a zároveň extrémně důležitá. Technologie blockchain se postupně dostává do všech odvětví IT včetně digitální archivace dat, kde nabízí velmi zajímavé možnosti. Tato práce prozkoumala jednu takovou možnost ale je téměř jisté že se tímto tématem bude zabývat mnoho dalších prací a téma se bude dále rozvíjet.

Literatura

[1] *How Much Data Is Created Every Day in 2022?* [online]. Techjury.net. May 02, 2022 [cit. 2022-05-10]. Dostupné z URL: https://techjury.net/blog/how-much-data-is-created-every-day/

[2] *Zálohování nebo archivace?* [online]. Acronis. 28.7.2016 [cit. 2021-11-4]. Dostupné z URL: https://www.acronis.cz/zalohovani-nebo-archivace/

[3] *What are Archives & Digital Archives?* [online]. Newton Gresham Library. Jun 30, 2021 [cit. 2021-11-4]. Dostupné z URL: https://shsulibraryguides.org/c.php?g=86819&p=558330

[4] *Databáze VHA* [online]. Vojenský ústřední archiv. 2010 [cit. 2021-11-4]. Dostupné z URL: http://www.vuapraha.cz/fallensoldierdatabase

[5] GLADNEY, H. M. *Principles for digital preservation* [online]. Communications of the ACM. 49 (2): 111–116. 15 December 2004 [cit. 2021-11-10]. Dostupné z URL: https://arxiv.org/ftp/cs/papers/0411/0411091.pdf

[6] HILLYARD, M. *Digital archiving: the seven pillars of metadata* [online]. The National Archives. 14 March 2018 [cit. 2021-10-28]. Dostupné z URL: https://blog.nationalarchives.gov.uk/digital-archiving-seven-pillars-metadata/

[7] DONALDSON, D., BELL, L. *Security, Archivists, and Digital Collections* [online]. Journal of Archival Organization 15(1-2):1-19. May 2019[cit. 2021-11-20]. Dostupné z URL: https://www.researchgate.net/publication/332972857_Security_Archivists_and_Digital_Collections

[8] COTE, C. *What is data integrity and why does it matter?* [online]. Harvard Business School Online. 04 FEB 2021 [cit. 2021-11-15]. Dostupné z URL: https://online.hbs.edu/blog/post/what-is-data-integrity

[9] SARAFIANOU, E., MOGYOROSI, M. *How Secure Are Encryption, Hashing, Encoding and Obfuscation?* [online]. Auth0 . August 21, 2019 [cit. 2021-11-15]. Dostupné z URL: https://auth0.com/blog/how-secure-are-encryption-hashing-encoding-and-obfuscation/

[10] HOLEČEK, O. *CRC (kontrolní součet)* [online]. root.cz . 30. 1. 2003 [cit. 2021-11-20]. Dostupné z URL:
https://www.root.cz/clanky/crc-kontrolni-soucet/

[11] *Parity bit* [online]. Computer Hope. 10/17/2017 [cit. 2021-11-20]. Dostupné z URL:
https://www.computerhope.com/jargon/p/paritybi.htm

[12] CRANE, C. *What Is a Hash Function in Cryptography? A Beginner's Guide* [online]. hashedout by the SSLStore. January 25, 2021. [cit. 2021-11-20]. Dostupné z URL:
https://www.thesslstore.com/blog/what-is-a-hash-function-in-cryptography-a-beginners-guide/

[13] MARTIN, J. *What is access control? A key component of data security* [online]. CSO | Security news, features and analysis about prevention, protection and business innovation. J AUG 21, 2019. [cit. 2021-11-20]. Dostupné z URL:
https://www.csoonline.com/article/3251714/what-is-access-control-a-key-component-of-data-security.html

[14] *What is access control?* [online]. Cloudflare. 2021. [cit. 2021-11-20]. Dostupné z URL:
https://www.cloudflare.com/learning/access-management/what-is-access-control/

[15] *Common Network Authentication Methods* [online]. N-able. 24th April, 2019. [cit. 2021-11-25]. Dostupné z URL:
https://www.n-able.com/blog/network-authentication-methods

[16] Synopsys Editorial Team. *5 reasons to use third-party authentication instead of creating your own* [online]. Synopsys, Monday, November 28, 2016. [cit. 2022-05-15]. Dostupné z URL:
https://www.synopsys.com/blogs/software-security/5-reasons-third-party-authentication

[17] OneLogin. *How Does Single Sign-On Work?* [online]. 2022. [cit. 2022-05-15]. Dostupné z URL:
https://www.onelogin.com/learn/how-single-sign-on-works

[18] STALLINGS, William, *Cryptography and network security: principles and practice.* Pearson Education, 2016. ISBN 1292158581.

[19] *Referenční informace k nastavení BitLockeru* [online]. Microsoft Docs. 15. 08. 2021. [cit. 2021-11-25]. Dostupné z URL: https://docs.microsoft.com/cs-cz/mem/configmgr/protect/tech-ref/bitlocker/settings

[20] SHEN, Meng, Liehuang ZHU a Ke XU, *Blockchain: Empowering Secure Data Sharing*. Singapore: Springer Singapore Pte. Limited, 2020. ISBN 9789811559389.

[21] *BLOCKS* [online]. Ethereum.org 2021. [cit. 2021-11-30]. Dostupné z URL: https://ethereum.org/en/developers/docs/blocks/

[22] GAYVORONSKAYA, Tatiana a MEINEL, Christoph. *Blockchain: Hype or Innovation*. Cham: Springer International Publishing, 2021. ISBN 9783030615581.

[23] *Proof of Authority Explained* [online]. Binance Academy Dec 10, 2020. [cit. 2021-11-30]. Dostupné z URL: https://academy.binance.com/en/articles/proof-of-authority-explained

[24] *Bitcoin Energy Consumption Index* [online]. Digiconomist. [cit. 2021-12-12]. Dostupné z URL: https://digiconomist.net/bitcoin-energy-consumption/

[25] *BITCOIN - Kurz BTC/Bitcoin* [online]. Kurzy.cz. [cit. 2022-05-20]. Dostupné z URL: https://www.kurzy.cz/bitcoin/

[26] *NFT* [online]. Ethereum.org 2021. [cit. 2021-11-30]. Dostupné z URL: https://ethereum.org/en/nft/

[27] GRIFFITH, E. *Why an Animated Flying Cat With a Pop-Tart Body Sold for Almost $600,000* [online]. The New York Times, May 27, 2021. [cit. 2021-11-30]. Dostupné z URL: https://www.nytimes.com/2021/02/22/business/nft-nba-top-shot-crypto.html

[28] *Filecoin Documentation* [online]. Filecoin.io, 2021. [cit. 2021-11-30]. Dostupné z URL: https://docs.filecoin.io

[29] BRALIĆ, V., STANČIĆ, H., STENGÅRD, M., "A blockchain approach to digital archiving: digital signature certification chain preservation", Records Management Journal, Vol. 30 No. 3, pp. 345-362. 2020 [cit. 2021-11-15]. Dostupné z URL: https://doi.org/10.1108/RMJ-08-2019-0043

[30] HWANG, H. C., SHON, G. J., PARK, S. J., "Design of an Enhanced Web Archiving System for Preserving Content Integrity with Blockchain", Electronics 2020, 9, 1255 [cit. 2021-11-16]. Dostupné z URL: https://doi.org/10.3390/electronics9081255

[31] SIMIC, S. *CentOS vs Ubuntu: Choose the Best OS for Your Web Server* [online]. phoenixNAP, DECEMBER 23, 2019. [cit. 2021-12-08]. Dostupné z URL: https://phoenixnap.com/blog/centos-vs-ubuntu

[32] THOMAS, J. *Bulma: Free, open source, and modern CSS framework based on Flexbox* [online]. 2022. [cit. 2022-04-20] Dostupné z URL: https://bulma.io/

[33] Auth0. *Auth0 Vue.js Samples* [online]. 27 Sep 2019. [cit. 2022-04-25] Dostupné z URL: https://github.com/auth0-samples/auth0-vue-samples

[34] ZOLOTYKH, R. *Upload Files with Vue and Express*. 2018. Youtube série. [cit. 2022-04-15] Dostupné z URL: https://www.youtube.com/watch?v=GXe$_JpBQLTQlist = PLuNEz8XtB51LJ - 3DuKypUlvYlC1tF - BT5$

A Obsah elektronické přílohy

V elektronické příloze se nachází 3 adresáře:

- Java
- JavaScript
- Extra

Dále se zde nachází soubor README.txt, který obsahuje detailní popis zprovoznění a spuštění jednotlivých komponent.

Ve složce Java jsou zkompilované „.jar“ soubory tří Java aplikací - Archivační aplikace, Validační aplikace a aplikace Decryptor. Každá z těchto aplikací se nachází ve svojí složce a obsahuje i pomocné soubory - jako například konfigurační soubor Archivační aplikace nebo serializovaný blockchain. Navíc je zde složka BlockchainArchiveKod, která obsahuje okomentovaný kód všech tří aplikací dohromady, jelikož byly vyvíjeny jako jeden projekt.

Ve složce JavaScript jsou tři podsložky - „Front End“, „Back End“ a „Front End - dist“. Jak název napovídá, složka Front End obsahuje kód front endu webové aplikace, Back End pak back endu aplikace. Složka „Front End - dist“ pak obsahuje produkční verzi front endu.

Složka Extra pak obsahuje soubory, které lze využít pro lokální testování. Jedná se o dva dokumenty, v takovém stavu v jakém jsou zpracovány back endem. Lze je využít v případě, že je webová aplikace nefunkční.