DOKUMENTACE DO PŘEDMĚTU MPC-KRY, CLOUDOVÁ DATABÁZE PŘÍSTUPOVÝCH ÚDAJŮ

2022/23

JAKUB ASSZONYI, VOJTĚCH HYNEK, VOJTĚCH VÁŇA, MATĚJ VOPÁLKA

OBSAH

SEZNAM OBRÁZKŮ

1 ÚVOD A POPIS PROJEKTU

Našim společným projektem je problematika cloudové databáze přístupových údajů. Cílem tohoto projektu je vytvořit zabezpečenou databázi pro uchování uživatelských přístupových údajů. Tato databáze bude implementovat minimálně tři metody autentizace s využitím multifaktorové autentizace pro přístup do databáze. Aplikace bude také zajišťovat ověření důvěryhodnosti zařízení a šifrovanou komunikaci mezi uživatelem a databází. Přístup do databáze bude mít pouze uživatel a aplikace bude umožňovat systém pro obnovení přístupu do databáze.

Projekt byl naprogramován v Pythonu s využitím dostupných knihoven a bude zaměřen na bezpečnost a ochranu uživatelských dat. Cílem je vytvořit aplikaci, která bude schopna poskytnout uživatelům snadný přístup k jejich datům při zachování nejvyšší úrovně bezpečnosti. Tedy hlavním cílem je zajistit bezpečnost a ochranu uživatelských dat. Aplikace bude navržena tak, aby byla snadno použitelná a intuitivní pro uživatele.

Důležitým aspektem projektu je ověření důvěryhodnosti zařízení, ze kterého se uživatel připojuje do databáze. To pomůže zajistit, že pouze autorizované zařízení budou mít přístup k datům v databázi.

Komunikace mezi uživatelem a databází bude šifrovaná, aby byla zajištěna ochrana dat při přenosu. Přístup do databáze bude mít pouze uživatel bez možnosti nadřazeného administrátorského přístupu.

Aplikace bude také umožňovat systém pro obnovení přístupu do databáze v případě ztracených nebo zapomenutých údajů.

V rámci tohoto projektu se zabíráme cloudovou databází přístupových údajů. Proto si níže rozebereme některé důležité pojmy, které jsou potřeba pochopit pro správnou funkcionalitu a pochopení projektu.

Databáze

Jedná se o strukturovanou kolekci informací, která je uspořádána pro snadné prohledávání, vkládání, úpravy a odstraňování dat. Slouží k uchování a řízení velkého objemu informací v elektronické formě, například údaje o zákaznících, přístupové údaje a další. Databáze lze uložit na různém hardwaru a lze k nim přistupovat a spravovat je pomocí různých softwarových aplikací. Databáze jsou nezbytné pro mnoho oblastí informačních technologií, jako jsou webové aplikace, bezpečnostní systémy, ERP systémy, CRM systémy, Big Data a další.

Uživatelské přístupové údaje

Informace, které uživatel potřebuje k přihlášení do systému nebo k aplikaci, pro získání přístupu k určitým funkcím nebo datům. Údaje mohou zahrnovat uživatelské jméno, heslo, kódové slovo (PIN), otisk prstu nebo jiné biometrické údaje, tokeny, čipové karty a další. Přístupové údaje jsou používány ke kontrole a omezení přístupu k citlivým informacím, aplikacím a systémům, aby byla zajištěna bezpečnost a ochrana dat. Správné používání

uživatelských přístupových údajů je důležitou součástí zabezpečení informačních systémů a organizací a mělo by být chráněno před neoprávněným přístupem a zneužitím.

Metody autentizace

Proces ověřování totožnosti uživatele, který se pokouší získat přístup k systému nebo aplikaci. Metody autentizace jsou způsoby, jak ověřit totožnost uživatele. Některé z nejčastěji používaných metod autentizace jsou: [2]

- Uživatelské jméno a heslo - uživatel zadá své uživatelské jméno a heslo, které byly předem stanoveny a uloženy v databázi. Pokud údaje souhlasí s těmi uloženými, uživatel získá přístup.
- Biometrická autentizace - uživatel poskytuje biometrická data, jako například otisk prstu, sken duhovky nebo obličeje. Tyto údaje jsou porovnány s uloženými vzorky a pokud se shodují, uživatel získá přístup.
- Tokeny - uživatel získá token, což je zařízení nebo aplikace, které generuje náhodné číselné kódy, které slouží ke vstupu do systému.
- Certifikáty - uživatel poskytuje digitální certifikát, který byl vydaný důvěryhodnou autoritou. Tento certifikát ověřuje totožnost uživatele.
- SMS ověření - uživatel obdrží ověřovací kód na své telefonní číslo pomocí SMS zprávy a tento kód zadá do systému pro ověření totožnosti.

Tyto metody se často kombinují do tzv. multifaktorové autentizace (MFA), což je proces ověření totožnosti uživatele pomocí více než jednoho faktoru. MFA tak zvyšuje bezpečnost přístupu k systému nebo aplikaci, protože útočník by musel získat nejen heslo nebo jiný typ prvního faktoru, ale také překonat další ověřovací faktor (například biometrickou identifikaci či ověřovací kód na mobilním zařízení). Nejčastějšími faktory používanými pro MFA jsou tedy heslo nebo PIN kód, biometrická data (například otisky prstů, rozpoznání obličeje nebo hlasu), ověřovací kód, který je zaslán na mobilní zařízení nebo email, fyzický token nebo smart karta, kterou uživatel musí mít u sebe.

Cloud

V informatice se označuje poskytování výpočetních zdrojů, jako jsou servery, úložiště dat, síťové prvky, prostřednictvím internetu. Cloudové služby umožňují uživatelům přístup k výpočetním zdrojům, aniž by musely vlastnit a spravovat vlastní hardwarovou a softwarovou infrastrukturu. To umožňuje firmám i jednotlivcům rychle a snadno získat potřebné výpočetní zdroje pro své aplikace a služby. Existují tři hlavní typy cloudových služeb: veřejný cloud, soukromý cloud a hybridní cloud.

Mezi nejznámější poskytovatele cloudových služeb patří Amazon Web Services (AWS), Microsoft Azure a Google Cloud Platform (GCP).

Šifrování

Proces přeměny informací nebo dat do tajné formy, kterou mohou přečíst pouze osoby, kterým je určena. Používá se k zabezpečení důvěrných informací při přenosu přes veřejné sítě nebo k ukládání na nezabezpečených úložištích. Existují různé typy, jako jsou symetrické a asymetrické šifrování. Symetrické šifrování je založeno na použití společného tajného klíče, který je znám pouze odesílateli a příjemci. Klíč je použit jak k šifrování tak i dešifrování. Příkladem symetrického šifrování je například AES (Advanced Encryption Standard). Asymetrické šifrování, známé také jako veřejné klíčové šifrování, používá dva klíče, veřejný a soukromý. Veřejný klíč může být sdílen s kýmkoli a používá se k šifrování zpráv. Soukromý klíč je znám pouze příjemci a používá se k dešifrování zpráv. Příkladem asymetrického šifrování je například RSA (Rivest-Shamir-Adleman). Existují také další šifrovací algoritmy a protokoly, včetně SSL (Secure Sockets Layer), TLS (Transport Layer Security) a VPN (Virtual Private Network). Tyto technologie používají kombinaci symetrického a asymetrického šifrování a dalších technik pro zabezpečení přenosu dat.

TOTP - Time-Based One-Time Password

Metoda, která vyžaduje, aby uživatel při přihlášení zadával kromě svého uživatelského jména a hesla také jednorázové heslo, které je platné pouze po omezenou dobu. TOTP funguje na základě algoritmu, který generuje jednorázová hesla podle aktuálního času a společného tajného klíče mezi uživatelem a serverem. Tyto jednorázové kódy jsou poté zobrazovány uživateli v podobě šesti- nebo osmimístného čísla nebo QR kódu. Uživatel si může vytvořit TOTP token pomocí aplikace pro generování jednorázových hesel, např. Google Authenticator, nebo použít hardwarový token, jako jsou např. YubiKey. TOTP je považováno za velmi bezpečnou autentizační metodu, protože jednorázová hesla jsou platná pouze po omezenou dobu a nelze je použít znovu. Zároveň je však důležité chránit společný tajný klíč a přístup k TOTP tokenu, aby se zabránilo jeho zneužití. [3]

Hašovací funkce

Hash je matematická funkce, která převádí vstupní řetězec libovolné délky na řetězec fixní délky (tzv. hash hodnotu). Hashování se používá při ukládání hesel, kdy je uživatelské heslo převedeno na hash hodnotu a ta je uložena v databázi. Pokud někdo získá přístup k databázi hesel, nedokáže heslo získat, protože nezná původní heslo, pouze jeho hash hodnotu. Funkce musí být odolná proti reverznímu inženýrství, což znamená, že není možné z hash hodnoty získat původní heslo.

Salt

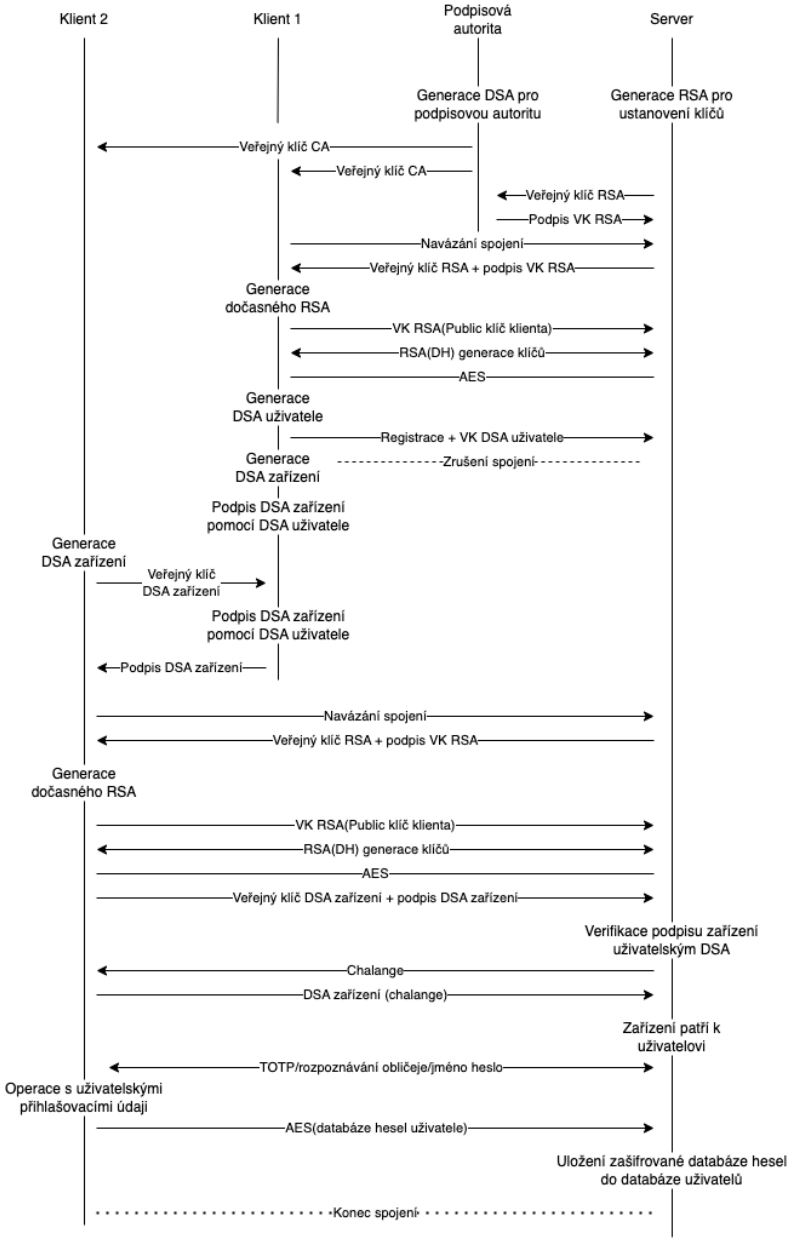
Náhodná hodnota, která je přidána k heslu před hashováním. Salt slouží k tomu, aby bylo složitější získat heslo pomocí útoku hrubou silou, kdy útočník zkouší různé kombinace hesel a ověřuje, zda se shodují s hash hodnotami v databázi. Pokud je pro každého uživatele

použit jiný salt, ztíží to práci útočníkovi, protože bude muset pro každou hash hodnotu použít jiný salt.

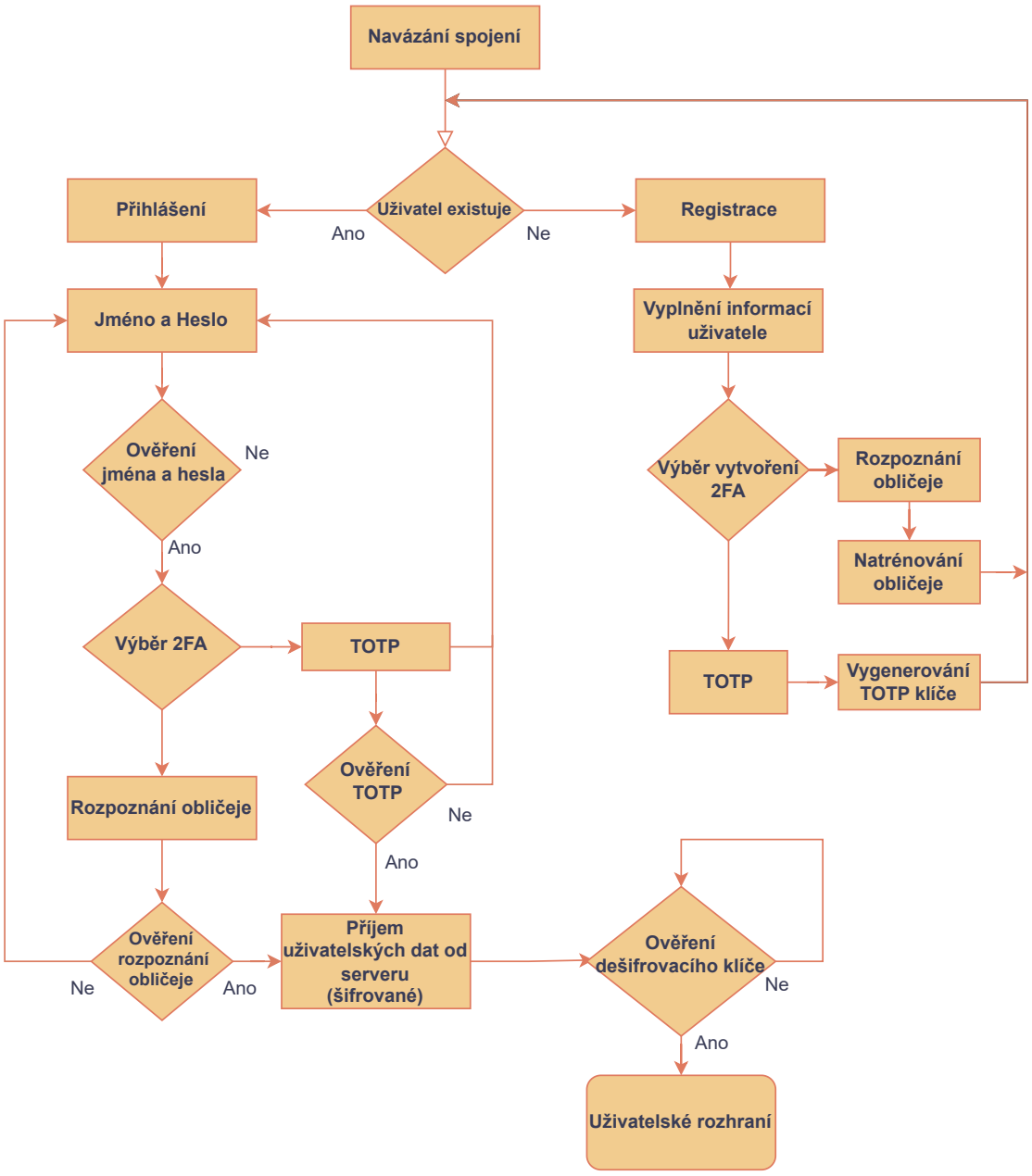
Iterace

Používá se zvýšení bezpečnosti hashování. Hashovací funkce se použije několikrát na vstupní řetězec (například 1000x). To znamená, že útočník musí provést více výpočtů, aby získal původní heslo. To ztíží útok hrubou silou a zvýší bezpečnost uložených hesel.

2 VÝVOJOVÝ DIAGRAM



Obr. 2.1: Diagram komunikace základních entit



Obr. 2.2: Vývojový diagram aplikace

3 POPIS KÓDU

V následující kapitole jsou stručně popsány základní části aplikace. Podrobnější dokumentace a komentáře týkající se funkčnosti kódu, vstupních a výstupních parametrů lze najít v dodaném kódu nebo na veřejně dostupném GitHub repozitáři viz [5] (větev `Final_KryPi`).

3.1 Klient

Nejdůležitější části kódu ze strany klienta se nachází ve třídách `Client.py` a `Client_session.py`. Třída `Client` slouží k vytvoření objektu klienta. Slouží ke spojení se serverem, odesílání a přijímání otevřeného textu nebo šifrovaných dat, a ověřování podpisů, generování ECDH. Dále obsahuje funkci pro vytvoření ověření pomocí obličeje. Podrobněji jsou tyto třídy rozebraný přímo na GitHubu.

3.2 Server

Nejdůležitější části kódu strany serveru se nachází ve třídě `Server_main.py`. Třída Server má inicializační metodu, která nastavuje hostitele a port serveru, a metodu start, která váže socket a naslouchá příchozím klientským spojením. Třída ClientThread dědí z modulu threading a definuje metody pro odesílání a přijímání dat a také pro import RSA klíčů, generování sdíleného klíče ECDH a uzavření spojení. Sdílený klíč ECDH se používá k šifrování a dešifrování dat pomocí šifrování AES. Veřejný klíč RSA a podepsaný hash jsou rovněž zasílány klientovi.

3.3 Podpisová autorita

Podpisová autorita, které klient věří, se nachází v souboru CA.py. Soubor funguje jednak jako knihovna a jednak jako samostatný nástroj pro generaci, podpis a ověření podpisů pomocí algoritmu DSA. Obsahuje funkce pro šifrování a dešifrování soukromého klíče pomocí uživatelského hesla, který se následně zahešuje a poté slouží jako klíč pro AES.

3.4 Autentizační metody

Pro autentizaci se primárně využívá metoda `DefaultLogin()`, která slouží pro základní autentizaci pomocí jména a hesla a následnou autentizaci pomocí 2FA. Server nabídne klientovi 2FA podle toho, jaké autentizační metody má klient k dispozici.

3.5 Jméno a heslo

Klient odešle jméno a heslo serveru, který ověří zda daný uživatel v databázi existuje a pokud ano, tak pomocí hashování a zasolení hesla ověří správnost hesla. Při registraci

v metodě `Register()` pošle jméno a heslo společně s dalšími informacemi z klienta na server. Na serveru se vytvoří uživatel pomocí daných informací a místo hesla se v databázi uloží jeho hash, sůl a počet iterací. Dalším krokem je ověření důvěryhodnosti klientského zařízení pomocí metody `credibility()`, více informací o této metodě je ve třídě `Server_functions.py` nebo `Client.py`. Poslední částí této metody je vybrání požadovaného 2FA.

3.6 TOTP

O autentizaci pomocí TOTP se u přihlášení opět stará metoda `DefaultLogin()`. V této metodě klient zadá jednorázový klíč (např. vygenerovaný pomocí Google Authenticatoru), který se odešle na server. Ten vypočítá pomocí seedu uloženého v databázi a aktuálního času svoje jednorázové heslo a porovná ho s tím, které mu pošle klient.

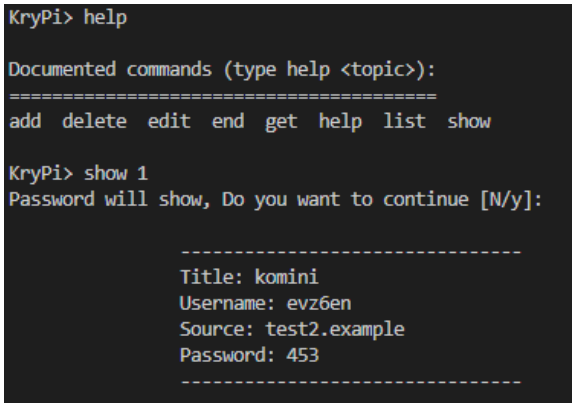
Při registraci autentizační metody TOTP se využívá funkcí `totp_registration()` a `create_TOTP()`. Server vygeneruje 16 znakový seed, který si uloží do databáze k danému uživateli a zároveň mu ho i pošle. Klient seed do konzole vypíše a zároveň s tím vygeneruje QR kód seedu.

3.7 Rozpoznaní obličeje

Pro autentizační metodu pomocí rozpoznání obličeje je použit python modul `OpenCV`. Při registraci se otevře okno s webkamerou, kde uživatel si naskenuje 70 obrázků svého obličeje. Pro to se využívá třída `Client_ReadFace.py`. Poté klient odešle data na server, kde je server přijme, dočasně uloží do adresáře, natrénuje data set, ve formátu `.yml` pro daného uživatele, a uložené fotky smaže. Pro to se využívá třída `Server_LearnFace.py`. Tím je dokončena registrace pro autentizaci pomocí rozpoznání obličeje uživatele. Při přihlašování, si uživatel naskenuje svůj obličej, program zachytí dva snímky obličeje uživatele. Poté jsou tyto snímky poslánu na server, kde server porovná snímky s naučeným data setem uživatele. Bere se v potaz malá kvalita webkamer, velká náchylnost ke změně osvětlení, výrazu ve tváři a úhlu , takže může docházet k false-pozitives (kdy aplikace rozpozná obličej jako jiného uživatele) a false-negative (Kdy to daného uživatele má rozpoznat ale nerozporná). Více informací a popis funkcí lze nalézt v dodaném kódu.

3.8 Uživatelské rozhraní

Uživatelské rozhraní je realizováno pomocí Python modulu `Cmd`. Je vytvořena vlastní příkazová řádka, ve které může uživatel provádět úpravy s jeho daty. Je implementováno šest příkazů, a to přidání, úprava, smazání, zobrazení hesla, vypsání seznamu údajů, zkopírování hesla do schránky, ukončení spojení a nápověda. Funkce jsou více popsané v dodaném kódu ve třídě `Shell.py`. Ukázku z uživatelského rozhraní lze vidět viz obrázek 3.1



Obr. 3.1: Ukázka uživatelského rozhraní po zadání příkazu help a show

4 POPIS INSTALACE

Projekt obsahuje soubor `README.md`, ve kterém je je také popsaný celý návod na spuštění projektu. Také se tam nachází údaje k testovacímu účtu, pomocí kterého lze snadno ověřit fungování aplikace bez nutnosti vytváření nového uživatele (neplatí pro FaceDetection). Doporučujeme tedy řídit se návodem v tomto souboru, případně pomocí obdobného návodu níže.

Instalace se skládá ze dvou bodů a to instalace serveru a instalace klienta. Popřípadě se dá rozšířit o nasazení podpisové autority.

4.1 Nasazení podpisové autority

Pro nasazení podpisové autority nám stačí pouze jeden soubor a to CA.py. Tento soubor je využit jako knihovna pro ověřování v klientské i serverové části a zároveň slouží jak nástroj pro vygenerování klíčů, podepisování souborů a ověření podpisů pro kryptografický protokol DSA (Digital Signature Algorithm).

Podpisová autorita může být samostatná entita, nebo může být součástí serveru, kdy uživatel věří vývojáři aplikace. Při obou případech se podpisová autorita nasadí příkazem python3 CA.py −−generate, který vygeneruje pár klíčů private_key.pem a public_key.pem. Privátní klíč je zašifrovaný heslem, které uživatel zadá při vygenerování a musí ho používat při podepisování.

4.2 Instalace serveru

Server slouží jako úschovna uživatelských údajů pro synchronizaci mezi více zařízeními a jako vzdálené úložiště. Pro jeho instalaci je zapotřebí provést následující kroky.

1. Nakopírovat všechny soubory aplikace na serverovou stranu, do adresáře ve kterém se bude server souštět.
2. Vygenerovat pár klíčů pro RSA příkazem python3 RSA_server.py, jehož výstupem jsou soubory private_key_RSA.pem sloužící jako soukromý klíč RSA a public_key_RSA.pem sloužící jako veřejný klíč RSA
3. Následně je třeba podepsat veřejný klíč RSA podpisovou autoritou. Záleží tedy na volbě zda je podpisová autorita samostatná, či je součástí serveru. Pokud je samostatná musí se veřejný klíč přenést k podpisové autoritě. Pokud je součástí serveru, už se veřejný klíč na místě kde má být. Podpis je realizován příkazem python3 CA.py −−sign public_key_RSA.pem a výstupem je soubor hash.sig který v případě samostatné podpisové autority musí být přesunut na server. Slouží totiž jako podpis, kterým klient verifikuje identitu serveru.
4. Posledním krokem je spuštění serverové aplikace příkazem python3 Server_main.py

4.3 Instalace klienta

Klientská aplikace je uživatelské rozhraní, které spouští klient při interakci s databází hesel. Aplikace má příkazové rozhraní a spouští se z příkazového řádku.

U klientské aplikace se tak jako u serveru musí nakopírovat všechny soubory aplikace na klientskou stanici do adresáře, kde se aplikace bude spouštět včetně předem vygenerovaného veřejného klíče podpisové autority

5 POPIS SPUŠTĚNÍ PROGRAMU

Při stažení kódu z GitHub repozitáře, nebudou fungovat některé podpisy a bude potřeba je opět vygenerovat Problém vzniká někde na straně GitHubu.

Spuštění programu se skládá ze dvou hlavní částí a to procesu registrace a procesu přihlášení. Oproti tomu registrovaný uživatel musí ověřit že se přihlašuje z důvěryhodného zařízení a tak musí vygenerovat podpisový pár DSA s ověřeným veřejným klíčem a dokázat vlastnictví privátního klíče podepsáním výzvy od serveru, kterou server ověří.

5.1 Registrace

Nový uživatel ještě nemá ověřené zařízení a tak se může registrovat bez dalších souborů.

1. Uživatel spustí aplikaci příkazem python3 Client.py
2. Registrace vytvoří pár klíčů DSA pro ověření uživatelského zařízení. Tento pár klíčů je pouze na zařízení využitém při registraci, ze kterého se stává podpisová autorita pro konkrétního uživatele a musí podepsat všechny ostatní podpisové veřejné klíče už pro konkrétní zařízení. Soukromý klíč je uložen jako private_key_<username>.pem.
3. Pro každé další první přihlášení z nového zařízení si musí uživatel vytvořit pár klíčů pomocí pomocného nástroje příkazem python3 App_client_gen_dsa_tool.py který vygeneruje pár klíčů private_key_DSA_<username>.pem, který je zašifrovaný heslem, které uživatel zadá a public_key_DSA_<username>.pem který uživatel musí podepsat soukromým klíčem private_key_<username>.pem pomocí pomocného nástroje python3 App_client_sign_user.py na prvním zařízení na kterém se registroval a podpis obsažený v souboru hash_<username>_verify.sig přenést zpět na zařízení, které registroval.
4. Po tomto procesu může začít už klasické přihlášení.

5.2 Ovládání aplikace

Ovládání aplikace, při registraci nového uživatele a přihlášení stávajícího uživatele, je pomocí následování pokynů, které se vypisují do konzole velmi jednoduché. Základní vstupy aplikace z terminálu jsou ošetřené tak, aby nedošlo k nechtěnému pádu aplikace.

Uživatelské rozhraní pro úpravu databáze je opět řešeno pomocí dialogu v terminálu a pomocí následování pokynů lze jednoduše ovládat. Případně je možné v uživatelském rozhraní použít příkaz `help`. Podrobný popis je v README našeho github repozitáře [5] (větev `Final_KryPi`).

6 POUŽITÉ EXTERNÍ KNIHOVNY A JEJICH VERZE

- `Socket` – Slouží k vytváření a komunikaci sítě prostřednictvím socketů. Umožňuje vytvářet síťové aplikace, jako jsou klienti a servery a komunikovat pomocí různých protokolů, jako jsou TCP a UDP. V Pythonu socket funguje jako objekt, který představuje koncový bod v síťové komunikaci. Knihovna umožňuje vytváření socketů pro různé typy síťových aplikací, včetně streamování dat (jako je TCP), přenosu dat bez spojení (jako je UDP) a přístupu k síťovým službám pomocí protokolu DNS. Knihovna je použita pro navázání spojení mezi serverovou a klientskou částí aplikace.
- `Cryptography.hazmat.primitives` – Poskytuje nízkoúrovňové funkce pro práci s kryptografií, jako jsou hašovací funkce, symetrické a asymetrické šifry, digitální podpisy a další. V našem projektu využíváme tuto knihovnu zejména pro vygenerování soukromých klíčů a ustanovení tajného klíče mezi klientem a serverem. [4]
- `Crypto.Cipher` – Knihovna poskytuje kryptografické funkce pro šifrování a dešifrování dat. Konkrétně využíváme třídu `AES`. Tuto knihovnu využíváme ve vlastní implementaci metod pro šifrování a dešifrování dat. Využíváme také knihovnu.
- `ecdsa` – Knihovna pro práci s digitálními podpisy a ověřování pravosti digitálních dokumentů. ECDSA používá algoritmy založené na eliptických křivkách pro výpočet digitálních podpisů, což umožňuje vytváření podpisů s menší velikostí klíčů a kratšími podpisy než u jiných algoritmů.
- `SQLAlchemy` – Knihovna slouží jako abstraktní vrstva nad relačními databázemi, umožňující programátorům pracovat s databázemi pomocí objektů a metod, místo psaní SQL dotazů přímo. SQLAlchemy podporuje mnoho typů databází, včetně SQLite, MySQL, PostgreSQL a Oracle, a umožňuje vytvářet a spravovat databázové schémata, dotazy a transakce. SQLAlchemy také obsahuje nástroje pro mapování objektů na databázové tabulky, provádění změn schématu bez ztráty dat, podporu pro různé typy datových polí a mnoho dalších funkcí, které usnadňují práci s relačními databázemi v Pythonu.
- `PyOTP` – Knihovna pro generování a ověřování jednorázových hesel používaných v autentizačních mechanismech, jako je TOTP. PyOTP umožňuje uživatelům generovat jednorázová hesla založená na čase nebo počítadlu, která jsou použitelná pro autentizaci uživatelů a přístup k různým aplikacím a službám. PyOTP implementuje standardní TOTP a HOTP algoritmy pro generování jednorázových hesel a podporuje použití společného tajného klíče nebo konfiguraci vlastního tajného klíče. Knihovna také poskytuje funkce pro ověřování jednorázových hesel a kontrolu jejich platnosti. PyOTP je open-source knihovna a lze ji nainstalovat pomocí balíčkovacího systému pip. Kromě toho, pyOTP také podporuje integraci s dalšími nástroji pro autentizaci, jako jsou Google Authenticator a FreeOTP.
- `OpenCV` – Python knihovna OpenCV poskytuje snadný přístup k mnoha funkcím OpenCV pomocí Python rozhraní, což umožňuje snadné a rychlé vytváření aplikací pro zpracování obrazu a strojové vidění v Pythonu. OpenCV se často používá v prů-

myslu, výzkumu a akademickém prostředí a je velmi užitečný nástroj pro zpracování a analýzu obrazů.

7 ZÁVĚR

V projektu jsme úspěšně dokončili realizaci zabezpečené cloudové databáze přístupových údajů, která úspěšně uchovává uživatelské přístupové údaje (zdroj, uživatel, heslo). Byly vytvořeny tři strany komunikace a to strana serveru, klienta a certifikační autority. V rámci projektu byly implementovány tři metody autentizace a to pomocí jména a hesla, jednorázových časových kódů (TOTP) a rozpoznání obličeje. Přihlášení je vždy dvoufaktorové a to tak, že první metoda autentizace je vždy pomocí jména a hesla a druhá pomocí TOTP, nebo rozpoznání obličeje.

Dále byl implementován systém pro ověřování důvěryhodnosti serveru, na který se klient přihlašuje, a také ověření důvěryhodnosti zařízení, ze kterého se uživatel připojuje do databáze. S tím že uživatel se může připojit z více zařízení, na kterých je potřeba mít náležité kryptografické primitiva. A také ověření důvěryhodnosti serveru, na který se uživatel připojuje pomocí certifikační autority. Komunikace mezi uživatelem a serverem je šifrovaná pomocí symetrické blokové šifry AES použité v CBC.

Přístup do databáze má pouze uživatel bez možnosti nadřazeného administrátorského přístupu. Součástí aplikace je také úspěšně fungující systém pro obnovu přístupu do databáze, který využívá záložního klíče. Pomocí něhož se uživatel dozví klíč k jeho datům. Celkově je tedy zabezpečená databáze připravena pro použití a poskytuje vysokou úroveň bezpečnosti dat.

Výsledná databáze tedy splňuje požadavky zadané v počátečním zadání a dokáže úspěšně zabezpečit uživatelské přístupové údaje. Snažili jsme se minimalizovat rizika spojená s únikem dat a navrhnout co nejbezpečnější řešení. Po dokončení projektu byly provedeny testy, které prokázaly správnou funkčnost celé aplikace a její schopnost ochránit data před neoprávněným přístupem.

Díky úspěšné realizaci projektu se podařilo zvýšit úroveň zabezpečení ukládání dat a minimalizovat rizika spojená s únikem informací. Vytvořená databáze tak umožňuje bezpečné ukládání citlivých dat, což je pro mnoho podniků a organizací velmi důležité. Tým je spokojen s výsledkem a věří, že vytvořená aplikace by teoreticky mohla být úspěšně využívána v praxi.

LITERATURA

[1] S. Ibrokhimov, K. L. Hui, A. Abdulhakim Al-Absi, h. j. lee and M. Sain, "Multi-Factor Authentication in Cyber Physical System: A State of Art Survey,"2019 21st International Conference on Advanced Communication Technology (ICACT), PyeongChang, Korea (South), 2019, pp. 279-284, doi: 10.23919/ICACT.2019.8701960.

[2] IDrnd. 5 Authentication Methods That Can Prevent the Next Breach [online]. [cit. 09.03.2023]. Dostupné z: https://www.idrnd.ai/5-authentication-methods-that-can-prevent-the-next-breach/

[3] OneLogin. OTP vs. TOTP vs. HOTP: What's the Difference? [online]. [cit. 09.03.2023]. Dostupné z: https://www.onelogin.com/learn/otp-totp-hotp

[4] BOWER, Steve. Python Cryptography and Hazmat. Medium [online]. 2019-02-05 [cit. 09.03.2023]. Dostupné z: https://medium.com/asecuritysite-when-bob-met-alice/python-cryptography-and-hazmat-105aaafc05a9

[5] GitHub: Let's build from here · GitHub [online]. Dostupné z: https://github.com/MPC-KRY/KryPi/tree/Final_KryPi