VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Bronislav Strava

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

SYSTÉM ŘÍZENÍ PŘÍSTUPU ZALOŽENÝ NA ATRIBUTECH
ATTRIBUTE-BASED ACCESS CONTROL SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Bronislav Strava

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Petr Dzurenda, Ph.D.

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bronislav Strava ***ID:*** 211325

Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Systém řízení přístupu založený na atributech

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte problematiku atributových pověřovacích schémat a řízení přístupu na základě atributů. Výstupem bakalářské práce bude implementace přístupového systému využívající mikropočítač Raspberry Pi (přístupový terminál s webovým rozhraním) a autentizační token typu čipová karta, chytrý telefon nebo chytré hodinky. Přístupový systém bude založen na anonymních atributových pověřeních s revokací a bude implementovat entity vydavatele, revokační autoritu, uživatele a ověřovatele.

DOPORUČENÁ LITERATURA:

[1] MENEZES, Alfred, Paul C. VAN OORSCHOT a Scott A. VANSTONE. Handbook of applied cryptography. Boca Raton: CRC Press, c1997. Discrete mathematics and its applications. ISBN 0-8493-8523-7.

[2] HAJNÝ, J.; DZURENDA, P.; CAMENISCH, J.; DRIJVERS, M. Fast Keyed-Verification Anonymous Credentials on Standard Smart Cards. In ICT Systems Security and Privacy Protection. Springer Nature Switzerland, 2019. s. 1-13. ISBN: 978-3-030-22312-0.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: Ing. Petr Dzurenda, Ph.D.

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

ABSTRAKT

Tato práce se zaměřuje na implementaci webového rozhraní entity ověřovatele pro systém řízení přístupu založeného na RKVAC protokolu.

KLÍČOVÁ SLOVA

RKVAC, webová aplikace, systém řízení přístupu založený na atributech

ABSTRACT

This bachelor thesis is focused on implementing web application of verifier entity for attribute based access control system.

KEYWORDS

RKVAC, web application, attribute based access control system

STRAVA, Bronislav. *Systém řízení přístupu založený na atributech*. Brno, 2021, 51 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Petr Dzurenda, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Systém řízení přístupu založený na atributech“ jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské 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

PODĚKOVÁNÍ

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

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Systémy řízení přístupu jsou moderní, bezpečnostní mechanismy, se kterými se setkáváme prakticky každý den. Využívají různé způsoby autentizace uživatelů jako používání hesel, čipových karet, skenů obličeje apod. Je tak kladen stále větší důraz na bezpečnost a ochranu soukromí ze strany uživatelů i ze strany Evropské unie - Obecné nařízení o ochraně osobních údajů (GDPR), konkrétně požadavky „privacy-by-design“ a „privacy-by-default“. Tato skutečnost ovlivňuje i oblast řízení přístupu, kde je mimo jiné třeba zachovat anonymitu uživatelů. Tyto nároky realizuje oblast atributové autentizace, konkrétně pak technologie CDDH19 vyvíjená na Vysokém učení technickém v Brně, na jejíž částečnou, praktickou implementaci se tato práce zaměřuje - grafické rozhraní fyzického, přístupového terminálu entity ověřovatele, kompatibilní propojení s ostatními entitami přístupového systému (zejm. s revokační autoritou kolegy Ondřeje Malíka) a komunikace s modulem CKP.40 zajišťujícím otevírání fyzického zámku dveří.

V teoretické části se práce věnuje problematice řízení přístupu a rozdílu mezi anonymní atributovou autentizaci a autentizaci na základě identity.

Praktická část práce řeší problematiku vývoje webové aplikace pro systém řízení přístupu založeného na anonymních atributových pověřeních. Nejprve jsou popsány jednotlivé technologie použité při tvorbě aplikace, provedena analýza možností využití C kryptografických knihoven pro Android včetně nástroje Android NDK, dále je prezentován samotný vývoj, vytvořená webová aplikace a nakonec samotná implementace systému řízení přístupu.

1 Řízení přístupu

Řízení přístupu je bezpečnostní mechanismus, jehož smyslem je umožnit oprávněným uživatelům přístup k aktivům a opačně neoprávněným uživatelům v přístupu bránit. Cílem je minimalizovat riziko neautorizovaného přístupu. Aktiva reprezentují cokoliv, co je cenné. Mohou být fyzického charakteru (trezor, prostory, budovy), nebo logického charakteru (data, dokumenty, soubory). O přístup do systému usilují entity - nejčastěji uživatelé, fyzické osoby, servery atd. Pro umožnění přístupu k aktivům se využívají rozdílné autentizační mechanismy.

1.1 Autentizace

Autentizace je proces ověřování osoby nebo zařízení. Jde o prokázání skutečnosti, že entita (typicky osoba) je opravdu tím, za koho se vydává. Typicky se entity označují pomocí unikátních identifikátorů a prokazují se dokazovacím faktorem. Obecně se používají následující mechanismy autentizace.

1.1.1 Autentizace na základě znalosti

Uživatel disponuje znalostí, kterou si ukládá k sobě do paměti a následně uživatel znalost prokazuje u systému řízení přístupu. Používají se následující typy:

Login - heslo. Prokazuje se zde přímá znalost, tj. uživatel vyplňuje přístupové údaje do terminálu. Výhodou je, že dokazovací faktor nosí uživatel neustále s sebou. Nevýhodou pak je nutnost si dokazovací faktor zapamatovat a vybavit při ověřování, také zde není většinou možné volit komplexnější hesla.

Výzva - odpověď. Neprokazuje se zde přímá znalost, vrací se pouze odpověď na výzvu (Challenge-response).

Důkaz nulové znalosti. Dokazuje se zde znalost bez nutnosti odhalovat informace o entitě (Zero-Knowledge).

1.1.2 Autentizace na základě předmětu

Uživatelé se prokazují pomocí tokenu (předmětu). Dokazovací faktor je tedy uložen na paměťovém zařízení (např. čipová karta). Podstatnou výhodou je, že si uživatel nemusí pamatovat dokazovací faktor. Na druhou stranu si musí uživatel hlídat ztrátu nebo odcizení předmětu.

1.1.3 Autentizace na základě biometrické charakteristiky

V případě autentizace biometrické charakteristiky jsou dokazovacím faktorem biometrické údaje [1]. To jsou číselně reprezentované, biologické znaky osoby. Typicky se využívá struktura linií prstů nebo oční duhovka.

Znaky osoby jsou změřeny a uloženy do souboru, který slouží jako ověřovací faktor. Při autentizaci se pak biometrické údaje osoby změří znovu. Pokud jsou změřené údaje v dostatečné shodě s ověřovacím faktorem, tak je autentizace úspěšná. S biometrickou autentizací se lze setkat hlavně u přístupových systémů do místností, budov a areálů. Často se používá také při autentizaci osob vůči přenosným elektronickým zařízením, jako jsou notebooky.

1.2 Autorizace

Autorizace reprezentuje udělení oprávnění. Příkladem může být nový zaměstnanec ve firmě. Ta používá jako bezpečnostní opatření systém řízení přístupu, který kontroluje vstup do budovy. Při zahájení pracovního úvazku je nutné nového zaměstnance přidat na seznam zaměstnanců s oprávněním ke vstupu (autorizovaní uživatelé). V praxi se tak zastaví zaměstnanec u konkrétní autority, která provede zavedení přístupových údajů zaměstnance do databáze (autorizace). Pokud následně zaměstnanec přistoupí k ověřovacímu terminálu, zažádá o ověření a úspěšně se autentizuje, pak je mu udělen přístup do budovy.

1.3 Architektura

Systémy řízení přístupu regulují přístup k aktivům. Regulace spočívá v tom, že je obecně k aktivům přístup omezován. Splní-li však entita dané požadavky, je jí přístup udělen. Jeho funkce je tak být kontrolující překážkou mezi entitami a cennými aktivy [1]. V klasických přístupových systémech jsou práva vázána na identitu entit, a tak se v přístupových systémech sjednává:

- Identita - unikátní identifikátor konkrétní entity.
- Ověřovací faktor - data, jimiž systémem řízení přístupu entitu ověřuje (referenční data biometrických skenů, zahashované heslo...).
- Dokazovací faktor - prvek, jímž entita prokazuje svou identitu (a tím i přístupová práva).

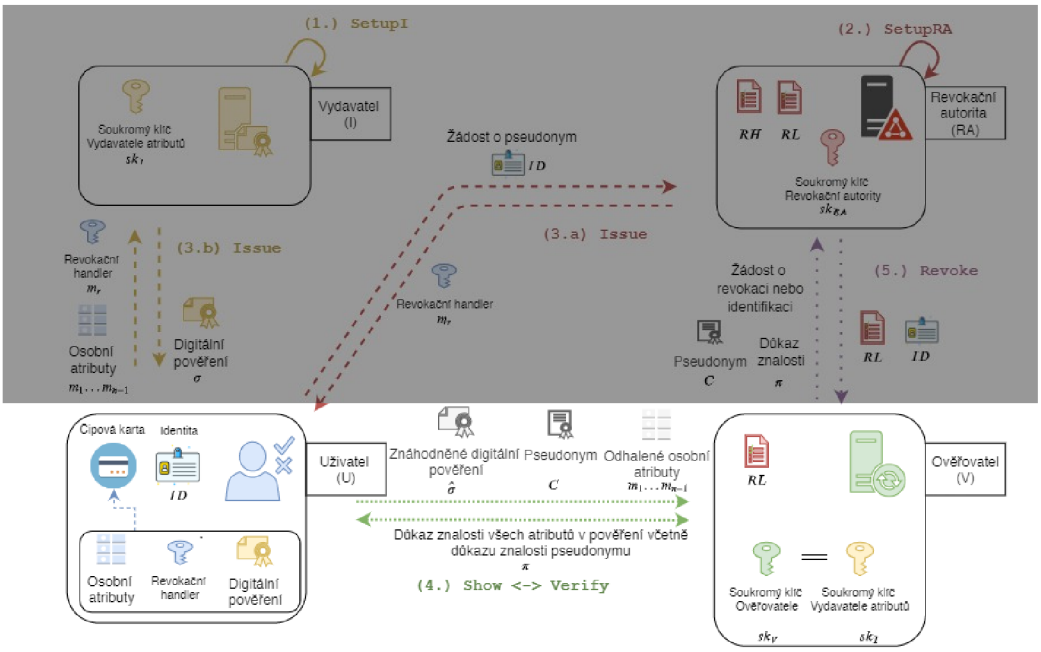
2 Autentizace pomocí anonymních atributových pověření

Při autentizaci uživatele pomocí identity dochází k ověření uživatele odhalením jeho identity. S tím se pojí různá rizika, např. riziko krádeže, kdy útočník oběti odcizí identitu a úmyslně ji využívá k získání nějaké výhody... Tento problém řeší autentizace pomocí atributů.

Atributová autentizace je forma autentizačního mechanismu, který provádí prokazování entity (typicky osoby) **bez nutnosti odhalení její identity**. Pro systém řízení přístupu není důležitá znalost uživatele, a proto údaje o uživateli nejsou třeba. Autentizace je realizována na základě konkrétních atributů, kterými osoba disponuje.

2.1 RKVAC protokol

Pro realizaci přístupového systému byla školitelem zadána C implementace RKVAC protokolu (obr. 2.1). Osobně jsem se zaměřil především na implementaci webového rozhraní entity ověřovatele a uživatele pro fyzický termínál řízení přístupu. Entity revokační autority a vydavatele tvořil kolega Malík. dohromady tak tvoří ucelený, plně kompatibilní systém řízení přístupu.



Obr. 2.1: RKVAC protokol [2]

Pro více informací o RKVAC protokolu, viz [2].

Příklady

Příklady atributové autentizace jsou:

- Přístup do kina na promítání věkově omezeného filmu.
  Atributem je zde věk. U výdeje lístků je osoba před vstupem do promítacího sálu požádána o předložení např. členského průkazu s vyznačeným datem narození, čímž dojde k ověření požadavku na minimální věkovou hranici (být starší než...). Pro zaměstnance kina není nutné znát identitu osoby (jméno a příjmení), osoba pouze ukazuje držený atribut věk, který buď plní podmínku věkového omezení, nebo ne. Při splnění této podmínky je následně návštěvníkovi kina poskytnut nebo zamítnut přístup do promítacího sálu.
- Sleva při nákupu v obchodu. Zákazník obchodu přistupuje k pokladně. U pokladny je požádán o zaplacení vybrané elektroniky. Na vybrané zboží je možné získat slevu, podmínkou je však platný studentský průkaz. Zákazník je informován o možnosti získání slevy. V případě, kdy zákazník je studentem a disponuje průkazem, průkaz ukáže a získá slevu. V případě, že zákazník studentem není, slevu nedostane. Zaměstnanec obchodu nepotřebuje vědět zákazníkovu národnost nebo rodné číslo pro poskytnutí slevy. Atributem je zde tedy platný, studentský průkaz.

3 Technologie

Pro vývoj webové aplikace je nutné zvolit vhodné technologie splňující patřičné požadavky. Hlavním problémem je zajištění komunikace mezi klientem a serverem, dále je třeba realizovat komunikaci mezi serverem a knihovnou RKVAC protokolu.

3.1 HTML

HTML (Hypertext Markup Language) je značkovací jazyk. Spolu s dalšími technologiemi jako CSS a JavaScript je používaný pro psaní webových stránek. Nejnovější verzí je aktuálně HTML 5.2, která je dostupná od prosince 2017. Webové prohlížeče obdrží HTML dokument od webového serveru a vyrenderují do multimediální webové stránky stránky. Dříve bylo nutné používat pro zobrazování multimediálního obsahu pluginy či technologii Flash, což již od vydání verze HTML5 není nutné.

Základní jednotkou je prvek (HTML element), ten může být párový či nepárový. Párový se skládá z počáteční a koncové značky, mezi které bývá vložen text. Samotné elementy lze rozřazovat podle identifikátorů nebo tříd, ke kterým lze pak přistupovat při stylizování webových stránek a při volání JavaScriptových funkcí.

3.2 CSS

CSS (Cascade Style Sheets) je jazyk pro stylizování HTML nebo XML dokumentů. Lze nastavovat fonty, velikosti písem, umístění, okraje, barvy, výplně, mezery... Stylizování přispívá k vizuálně lepší strukturovatelnosti a přehlednosti webových stránek. Aktuálně nejnovější verze je CSS3, která poskytuje možnost nastavení zaoblených rohů, stínů u elementů nebo např. průhlednost.

3.3 IntelliJ IDEA

IntelliJ IDEA je multiplatformní vývojové prostředí, které je spravováno českou softwarovou firmou JetBrains s.r.o sídlící v Praze. Jeho hlavní výhodou oproti konkurenčním vývojovým prostředím jako NetBeans a Eclipse je jeho větší podpora pro vývoj android aplikací – na platformě IntelliJ je postaveno vývojové prostředí Android Studio, které se stalo oficiálním vývojovým prostředím pro Android v roce 2013. IntelliJ je primárně zaměřené především ke tvorbě Java aplikací, dá se využít i například k vývoji v jazycích PHP, Kotlin, Groovy, Python, Ruby, SQL nebo JavaScript.

3.4 Spring Boot

Spring Boot je open source framework postavený na Javě. Poskytuje Java vývojářům platformu pro efektivnější konfiguraci Spring aplikací, což zlepšuje produktivitu a snižuje čas vývoje. Je spravován organizací Pivotal Software, Inc. Používá se široce ke tvorbě RESTful aplikací a mikroslužeb.

3.4.1 Thymeleaf

Thymeleaf je Java knihovna, která se používá při vývoji webových aplikací. Poskytuje platformu pro modifikaci HTML/XML stránek. V praxi tak řeší problém se statickými HTML stránkami, které dají na straně serveru dynamicky měnit přímo za běhu programu v závislosti na implementované aplikaci. To je výhodné, je-li třeba zobrazovat stránku s menšími obměnami v struktuře kódu. Ve výsledku není nutné tvořit dvacet podobných, statických stránek, stačí přistupovat k jedné a kód v případě potřeby modifikovat pomocí Thymeleafu.

3.4.2 Spring Web

Spring Web je knihovna, která se používá při tvorbě RESTful webových aplikací. Implementuje Spring MVC (Model View Controller), který umožňuje rozdělit webovou aplikaci na uživatelské rozhraní a řídící logiku do samostatných komponent, což je výhodné, protože modifikace jedné části neovlivní části ostatní.

3.4.3 Gradle

Gradle je multiplatformní, balíčkovací nástroj hojně využívaný pro automatické sestavování Java aplikací. Běží na JVM (Java Virtual Machine), proto je třeba mít pro použití nainstalovaný příslušný Java Development Kit.

3.5 Systemd

Systemd je systémový démon pro správu služeb především unixových systémů. Je to první proces, který je spuštěn při bootování systému a který spouští další uřivatelské procesy. V implementaci je použit pro konfiguraci vytvořené aplikace jako systémové služby.

3.6 Cron

Cron je unixový démon umožňující plánovat spouštění úloh, procesů a programů.

3.7 Analýza možnosti využití Android NDK

Programování android aplikací bývá většinou realizováno v programovacím jazyce Java, která umožňuje objektově orientovaný přístup. Někdy je ale potřeba aplikaci výkonnostně optimalizovat překonáním některých limitací Javy (správa paměti, překlad kódu), programováním přímo pro specifickou platformu, tzv. native development (v případě androidu vývoj v C a C++). Google poskytuje pro tento vývoj nástroj Android NDK (Native Development Kit). Typické použití Android NDK je pro mobilní zařízení, nebo pro ARM procesory na miniaturních počítačích, jako je Raspberry PI pro výkonnostně náročné sekce kódu.

Z hlediska vývoje webových aplikací není Android NDK optimální volba, neboť postrádá jistou flexibilitu pro tvorbu. Android NDK je možné použít pro tvorbu JNI knihoven, které lze využít pro výkonnostní optimalizaci částí kódu.

3.7.1 Analýza možnosti integrace C kryptografických knihoven

Při vývoji přístupového terminálu byla zvažována možnost realizovace pomocí integrace JNI knihoven do projektu. Náslůedující sekce se zabývá stručným přehledem možností využití C kryptografických knihoven pro vývoj Android aplikací a vzájemnému porovnávání implementovaných algoritmů daných knihoven.

3.7.2 Kryptografické knihovny

Pro samotný vývoj android aplikací lze využí zejména tyto C kryptografické knihovny: **cryptlib** [4], **Libgcrypt** [5], **libsodium** [6], **OpenSSL** [7], **wolfCrypt** [8]. Při vývoji aplikací užívajících kryptografické knihovny je nutné, aby knihovny splňovaly určité bezpečnostní požadavky.
Pro schvalování kryptografických knihoven vznikl standard FIPS 140-2 [9], který vydala americká vládní agentura NIST. V současnosti splňuje jeho požadavky knihovna wolfCrypt. Jeho nástupcem je standard FIPS 140-3, který nabyl účinnosti 22. září 2019.

Implementace operací s kryptografickými klíči

Tato část se věnuje porovnávání implementovaných kryptografických algoritmů.
V tab. 3.1 je zobrazeno srovnání knihoven z hlediska možností pro generování a výměny klíčů, šifrování dat a podepisování.

Tab. 3.1: Srovnání technologií pro generování a ustavování klíčů, šifrování a podepisování

Knihovna	ECDH	DH	DSA	RSA	ElGamal
`cryptlib`	Ano	Ano	Ano	Ano	Ano
`Libgcrypt`	Ano	Ano	Ano	Ano	Ano
`libsodium`	Ano	Ne	Ne	Ne	Ne
`OpenSSL`	Ano	Ano	Ano	Ano	Ne
`wolfCrypt`	Ano	Ano	Ano	Ano	Ne

Z porovnání vyplývá, že většina knihoven implementuje ECDH, DH, DSA i RSA, naopak málo implementovaným algoritmem je ElGamal, kerý pokrývají pouze knihovny cryptlib a Libgcrypt. Dále také v této kategorii výrazně zaostává knihovna libsodium.

Implementace hash funkcí

Hashe jsou jednosměrné funkce převádějící vstupní data do výsledného, unikátního otisku. V tab. 3.2 je zobrazen přehled implementovaných hash funkcí vybraných kryptografických knihoven.

Tab. 3.2: Srovnání implementací hash funkcí

Knihovna	MD5	SHA-2	SHA-3	Whirlpool
`cryptlib`	Ano	Ano	Ano	Ano
`Libgcrypt`	Ano	Ano	Ano	Ano
`libsodium`	Ne	Ano	Ne	Ne
`OpenSSL`	Ano	Ano	Ano	Ano
`wolfCrypt`	Ano	Ano	Ano	Ne

Většina knihoven implementuje algoritmy SHA-2 a SHA-3. Knihovny implementují bezpečnostně zastaralý algoritmu MD5. OpenSSL a Libgcrypt pokrývá všechny výše zmíněné technologie. V této kategorii opět zaostává knihovna libsodium.

Implementace MAC algoritmů

MAC algoritmy jsou technologie zajišťující autenticitu a integritu posílaných zpráv. Používá se kombinace hash funkcí a kryptografického klíčů. Srovnání implementací MAC algoritmů jednotlivých knihoven je zobrazeno v tab. 3.3.

Tab. 3.3: Srovnání implementací MAC algoritmů

Knihovna	HMAC-MD5	HMAC-SHA-2	Poly1305-AES
`cryptlib`	Ano	Ano	Ne
`Libgcrypt`	Ano	Ano	Ano
`libsodium`	Ne	Ano	Ano
`OpenSSL`	Ano	Ano	Ano
`wolfCrypt`	Ano	Ano	Ano

Technologii HMAC-MD5 neimplementuje knihovna libsodium, Poly1305-AES implementují všechny knihovny kromě cryptlib. Všechny knihovny také implementují technologii HMAC-SHA-2.

Implementace blokových šifer

Blokové šifry jsou symetrické šifry pracující na rozdíl od proudových šifer s bloky pevně stanovené délky, ty mohou být provozovány v různých operačních módech, viz doporučení NIST [10]. Implementace jednotlivých šifer danými knihovnami je porovnána v tab. 3.4.

Tab. 3.4: Srovnání implementací blokových šifer

Knihovna	Camellia	Blowfish	AES	3DES	Twofish
`cryptlib`	Ne	Ano	Ano	Ano	-
`Libgcrypt`	Ano	Ano	Ano	Ano	Ano
`libsodium`	Ne	Ne	Ano	Ne	Ne
`OpenSSL`	Ano	Ano	Ano	Ano	Ne
`wolfCrypt`	Ano	Ne	Ano	Ano	Ne

V této kategorii převahují knihovny OpenSSL a Libgcrypt, které implementuji všechny výše uvedené šifry. Naopak nejvíce zaostává knihovna libsodium, která podporuje pouze šifru AES-256.

Implementace proudových šifer.

Proudové šifry jsou symetrické šifry, které kombinují vstupní data s proudem bitů, daným šifrovacím klíčem a šifrovacím algoritmem. Porovnání implementovaných proudových šifer je zobrazeno v tab. 3.5.

Tab. 3.5: Srovnání implementací proudových šifer

Knihovna	RC4	Salsa20	ChaCha
`cryptlib`	Ano	Ne	Ne
`Libgcrypt`	Ano	Ano	Ano
`libsodium`	Ne	Ano	Ano
`OpenSSL`	Ano	Ne	Ano
`wolfCrypt`	Ano	Ano	Ano

V této kategorii opět vedou knihovny wolfCrypt a Libgcrypt.

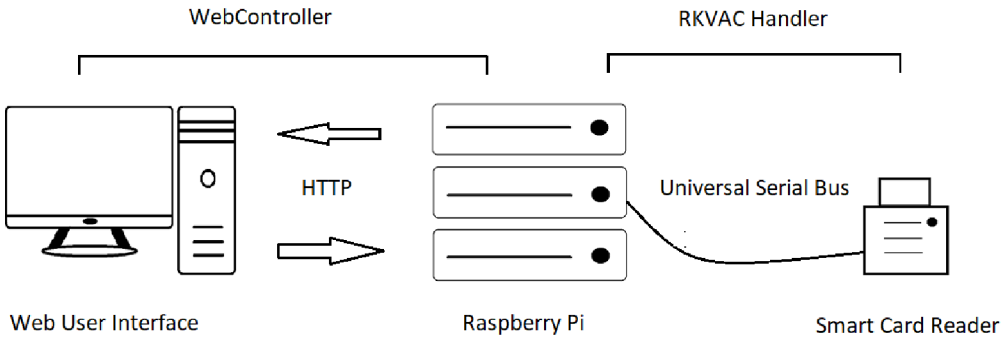
3.7.3 Závěr

Dle porovnání počtu implementovaných algoritmů vede knihovna Libgcrypt, která implementuje všechny porovnávané technologie, velmi praktickou knihovnou se ukazuje být i knihovna OpenSSL. Naopak nejméně vhodnou knihovnou pro vývoj android aplikací (z hlediska porovnání počtu implementovaných technologií) je libsodium.

Pro realizaci přístupového systému byla nakonec zvolena možnost kombinace Java frameworku pro tvorbu webových aplikací Spring Boot a lokálně nainstalovaná C verze již poskytnuté knihovny rkvac protokolu, odpadla tedy nutnost reimplementovat kryptografické jádro a ušetřený čas byl využit v dalších fázích vývoje aplikace.

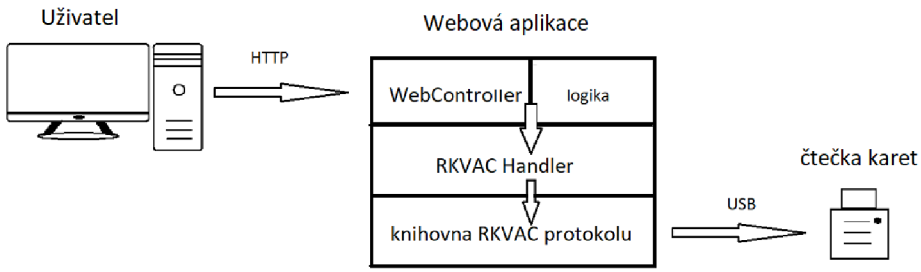
4 Architektura webové aplikace

Tato kapitola se zabývá vývojem webové aplikace. Jsou zde uvedeny a popsány stěžejní části aplikace, které jsou opatřeny ukázkami vybraných částí kódů. Samotný vývoj je realizován ve vývojovém prostředí IntelliJ IDEA. Jádro aplikace tvoří Java framework Spring Boot. Nejprve je rozebráno, jaký model komunikace je ve webové aplikaci použit, následně je vyobrazena samotná realizace komunikace.



Obr. 4.1: Schéma komunikace webové aplikace

Na obr. 4.1 je znázorněna komunikace, kterou implementuje webová aplikace. Nalevo je znázorněn uživatel. Prostřednictví webové uživatelského rozhraní uživatel interaguje s webovou aplikací spuštěnou na platformě Raspberry Pi (uprostřed). Interakce je realizována prostřednictvím HTTP dotazů, kterou zajišťuje třída WebController. Webový server dále přijatá data zpracovává a posílá je prostřednictvím třídy **RKVAC handler** na čtečku karet (vpravo). Poté je komunikace se čtečkou karet vyhodnocena a opět prostřednictvím třídy **WebController** vrácena uživateli HTTP protokolem.



Obr. 4.2: Vrstvové schéma komunikace webové aplikace

Na obrázku 4.2 je komunikační schéma znázorněno ve vrstvovém modelu. Uživatelem odeslaná data jsou zpracována třídou WebController. Zde se v závislosti na typu a formě přijatého požadavku realizuje určitá logika, poté jsou data předána třídě RKVAC handler, která obsluhuje knihovnu RKVAC protokolu, ta zajišťuje kryptografické operace nad čtečkou karet.

Následně komunikace probíhá opačným způsobem, tj. komunikace mezi knihovnou RKVAC protokolu a čtečkou karet je zpracovávána třídou RKVAC handler, která komunikaci předává zpět třídě WebController, kde danou logikou je komunikace vyhodnocena a prostřednictví HTTP protokolu obsloužen uživatel.

4.1 Třída RKVAC handler

Tato třída je vytvořena jako odpověď na problém komunikace mezi webovou aplikací a čtečkou karet. Komunikaci s čtečkou včetně kryptografických operací zajišťuje poskytnutá knihovna RKVAC protokolu. Ta má však zdrojové kódy napsané v jazyce C, a tak zde vyvstal problém přístupu webové aplikace ke knihovně RKVAC protokolu. Při vývoji jsem uvažoval dva způsoby, jak danou situaci řešit:

1. Přistupovat přímo k C knihovnám RKVAC protokolu.
2. Vytvořit pro implementaci RKVAC protokolu Java wrapper, který by danou knihovnu obsluhoval skrze unixový shell.

První bod se ukázal jako komplikovaný. Zdrojové kódy implementace RKVAC protokolu jsou primárně uzpůsobené pro CLI (terminál), analýza implementace by tak vyžadovala podstatné větší množství času, dále by zde vyvstal problém užití vhodné technologie, která by umožňovala volat C knihovny (zachovat tak výpočetní rychlost) a zároveň poskytovala flexibilitu pro tvorbu webové aplikace.

Řešením problému se tak stal bod druhý, spočívá v propojení webové aplikace a již hotové implementace rkvac protokolu skrze unixový shell, to zachovává výpočetní rychlost implementace a zároveň poskytuje flexibilitu při tvorbě webové aplikace. Samotné propojení zajišťuje implementovaná třída RKVAC handler, jejíž používání je zobrazeno ve výpisu 4.1.

Výpis 4.1: Volání objektů třídy RKVAC handler

```
1  RKVAC_handler rkvac = new RKVAC_handler();
2
3          rkvac.addAttributes("-i");
4          rkvac.addAttributes("-a");
5          rkvac.addAttributes(database);
6
7          rkvac.addUser_input("3");
8          rkvac.addUser_input(names);
9          rkvac.addUser_input(employee_id);
10         rkvac.addUser_input(employer);
11         rkvac.addUser_input(employee_position);
12         rkvac.addUser_input(write);
13
14         String container = rkvac.start();
```

Ve výpisu 4.1 je zobrazeno volání objektu vytvořené třídy RKVAC handler. Metoda addAttributes() nastavuje parametry, se kterými se má knihovna RKVAC protokolu skrze terminál spouštět. Třída RKVAC handler implementuje také metodu `addUser_input()`, která nastavuje uživatelský vstup, který je po spuštění zapsán do terminálu. Spouštění zajišťuje metoda start(), která následně vrací výpis z procesu při komunikaci se čtečkou karet (výpis 4.2).

Výpis 4.2: Obsluha terminálu

```
1  New thread created.
2  -----------WRITING------------
3  3
4  Bronislav Strava
5  1
6  VUT
7  student
8  Y
9  ---------DONE WRITING---------
10 [i] Issuer part started.
11 [+] Checking the ./data/Issuer/web_cache ... file does not
12 [+] Choose type of credentials (e.g. 1):
13  [1] eid
14  [2] ticket
15  [3] employee card
16  [4] user defined
17 > [+] Enter employee card credentials
18  [1] Name and surname>  [2] Employee id>  [3] Employer>
   [4] Employee position> [+] Writing the user attributes to
19  ...
20  [+] SCard response:
21 90 00
```

Ve výpisu 4.2 lze vidět na řádcích 3 až 8 uživatelské vstupy, které RKVAC handler předává knihovně RKVAC protokolu (metoda `addUser_Input()`), ty jsou poté postupně za jeden druhým zapsány jako normální, textový, uživatelský vstup implementace knihovny RKVAC protokolu. Tím je tedy realizována komunikace se čtečkou karet a také vyřešen problém propojení webové aplikace s knihovnou RKVAC protokolu.

4.1.1 Komplikace a problémy při vývoji

RKVAC handler zajišťuje jednak zapisování uživatelského vstupu a současně čtení dat z terminálu, to se z počátku vývoje nedařilo implementovat současně. Problém byl vyřešen paralelním vláknem, které umožňuje čtení i zápis dat najednou.

4.2 Třída WebController

Třída řeší problém komunikace mezi uživatelem a webovým serverem, zpracovává HTTP požadavky a také spojuje front-end s back-endem webové aplikace. Je zde implementována logika přístupu k jednotlivým stránkám a administrativní úkony.

4.2.1 Implementace logiky ověřovatele

K přístupu ke knihovně RKVAC protokolu je využita implementovaná třída RKVAC handler, ta umožňuje s malými obměnami v syntaxi kódu modifikovat logiku kontroleru a implementovat tak potřebnou logiku jednotlivých úkonů. Ve výpisu 4.3 je zobrazena ukázka offline změny epochy.

Výpis 4.3: Úspěšná autentizace a autorizace uživatele

```
1     @GetMapping("settings/offline-epoch-change")
2     public static String offlineEpochChange(Model model){
3         System.out.println("\n"+new Date()+" Offline Epoch Change Initialized");
4         RKVAC_handler offChangeEpoch = new RKVAC_handler();
5         offChangeEpoch.addAttributes("-v");
6         offChangeEpoch.addAttributes("-e");
7         String ep = null;
8         try {
9             String result = offChangeEpoch.start();
10            String epoch = new String(Files.readAllBytes(Path.of("/home/rkvac/
  rkvac_web/rkvac-protocol/build/data/Verifier/ve_epoch.dat")));
11            ep = epoch;
12            if (result.contains("Switch to the new epoch successful")){
13                System.out.println(new Date() +" New epoch: " + epoch);
14
15            }
16        }catch (InterruptedException e) {
17            e.printStackTrace();
18            model.addAttribute("message", "Epoch not changed.
19            Please check system logs." );
20            return "settings-err";
21        }catch (IOException e) {
22            e.printStackTrace();
23            model.addAttribute("message", "Epoch not changed.
24            Please check system logs." );
25            return "settings-err";
26        }
27
28        model.addAttribute("message", "Epoch successfully changed.
29        <br>New Epoch: " + ep);
30
31        return "settings-ok";
32    }
```

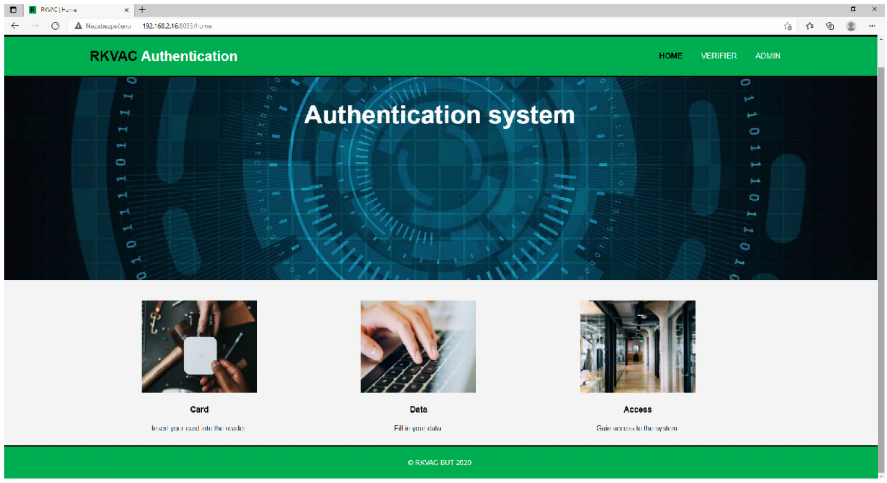
Na řádku 1 je vypsána anotace @GetMapping, která zajišťuje mapování příchozích HTTP GET požadavků na tuto metodu. Při odeslání požadavku typu GET na tuto adresu (přístup na stránku) je zavolána logika této metody a následně vrácena uživateli stránka definovaná klíčovým slovem *return*.

5 Webové rozhraní

V této kapitole je prezentováno vytvořené rozhraní webové aplikace.

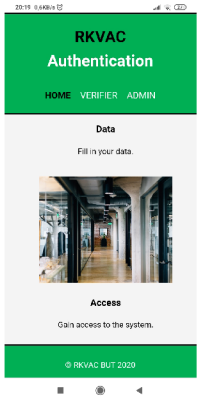
5.1 Home

Na obr. 5.1 je zobrazena úvodní stránka webové aplikace. Na tuto stránku se uživatel dostane, zadá-li adresu, na které je aplikace spuštěna. Na úvodní stránce jsou uživateli poskytnuty základní informace o používání přístupového systému.



Obr. 5.1: Zobrazení webové stránky Home

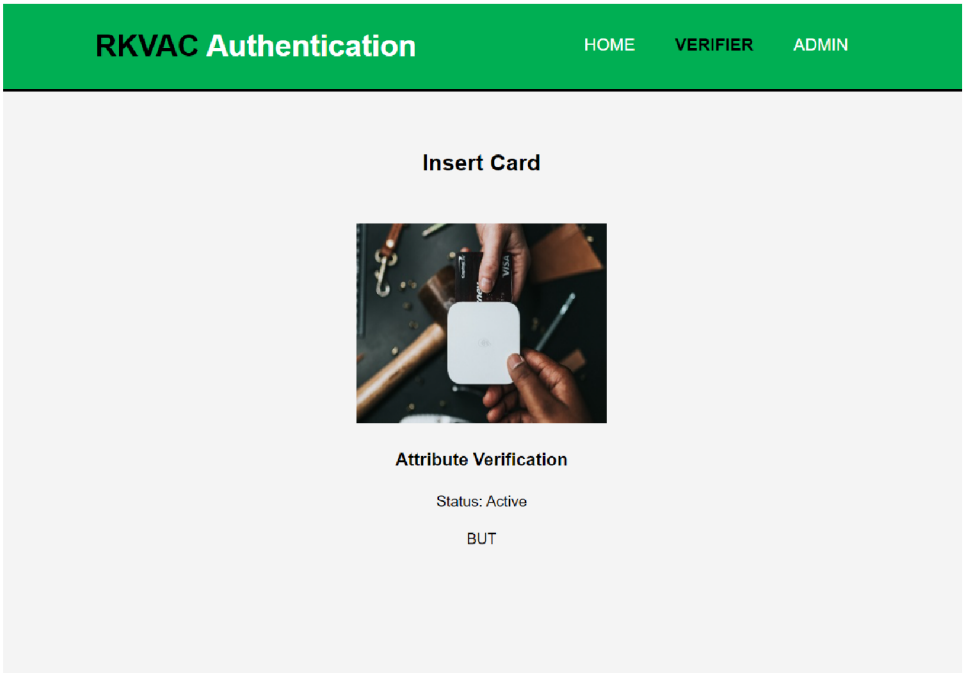
Webové rozhraní je také škálovatelné, tzn. při změně velikosti okna dojde k přestylizování HTML kódu. Optimalizované rozhraní pro mobilní telefony je zobrazeno na obr. 5.2.



Obr. 5.2: Mobilní zobrazení webové stránky Home

5.2 Verifier

Na obr. 5.3 je vyobrazena veřejná část aplikace ověřovatele. Má informativní charakter, vypisují se zde informace o stavu služby (Active/Disabled) a ověřované atributy (BUT). Ověřované atributy lze dynamicky měnit v nastavení aplikace (karta Credentials, kap. 5.3.1) včetně povolování a zakazování ověřování v kartě Settings (kap. 5.3.4).



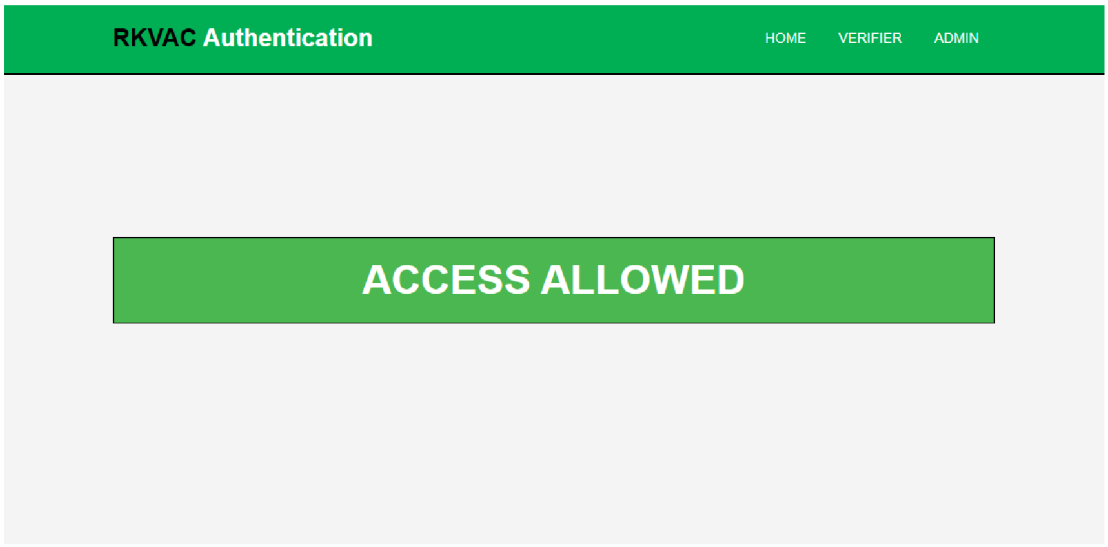
Obr. 5.3: Zobrazení webové stránky Verifier

Uživatel vloží kartu do čtečky karet a následně probíhá back-end ověření uživatele, úryvek z úspěšné autentizace a autorizace je vidět ve výpisu logu 5.1.

Výpis 5.1: Úspěšná autentizace a autorizace uživatele

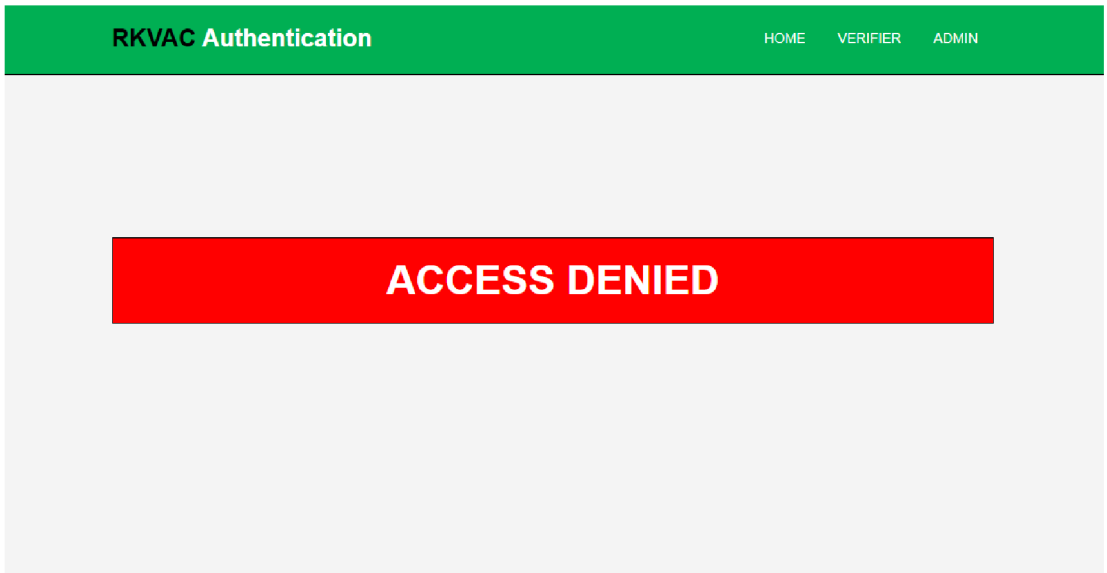
```
[i] Disclosed attributes:
 [2] 1E1220CC3773A7EFFB7609C6A535E451E8DD66A8730A4110C75437F1211A5194

ACCESS ALLOWED

[i] Verifier part complete.

Sun May 30 17:34:16 CEST 2021 Starting communication with CKP.40.
Sun May 30 17:34:16 CEST 2021 TLS server -> CKP.40 000007E17E37BB0B3020
Sun May 30 17:34:16 CEST 2021 TLS server <- CKP.40 00010006 61ms
Sun May 30 17:34:16 CEST 2021 TLS server <- CKP.40 01010006 19ms
Sun May 30 17:34:16 CEST 2021 TLS server -> CKP.40 01000106

Sun May 30 17:34:16 CEST 2021 CKP.40 ACCESS ALLOWED
```

Pokud je uživatel úspěšně autentizován a autorizován, je mu udělen přístup do místnosti. Na stránce dojde k zobrazení stavové hlášky o uděleném přístupu (obr. 5.4).



Obr. 5.4: Přístup povolen

V opačném případě je vrácena informativní stránka o zamítnutí přístupu (obr. 5.5).

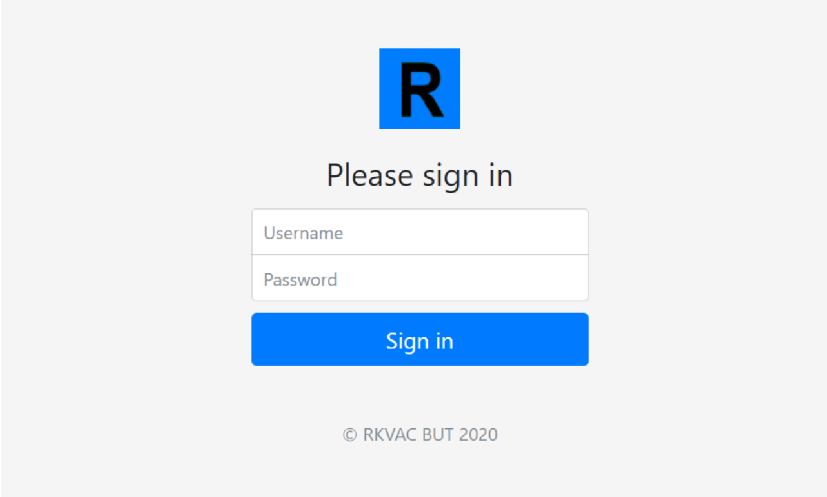


Obr. 5.5: Přístup zamítnut

Následně je zde opět zobrazena stránka s informacemi o stavu služby (obr. 5.3).

5.3 Admin

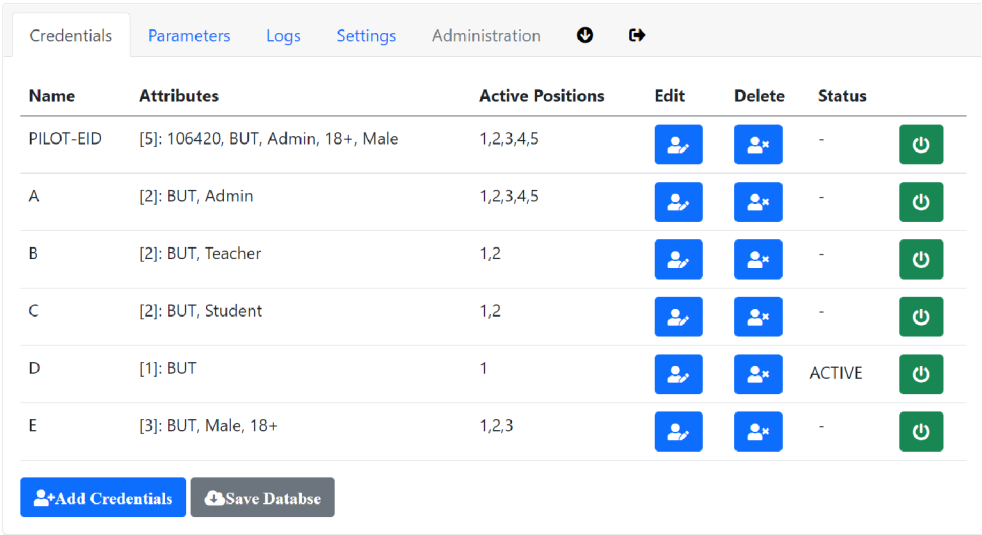
Sekce „Admin“ slouží k obsluze různých administračních úkonů entity ověřovatele, je neveřejná, tzn. uživatel se musí před vstupem přihlásit administrátorským účtem (obr. 5.6).



Obr. 5.6: Login

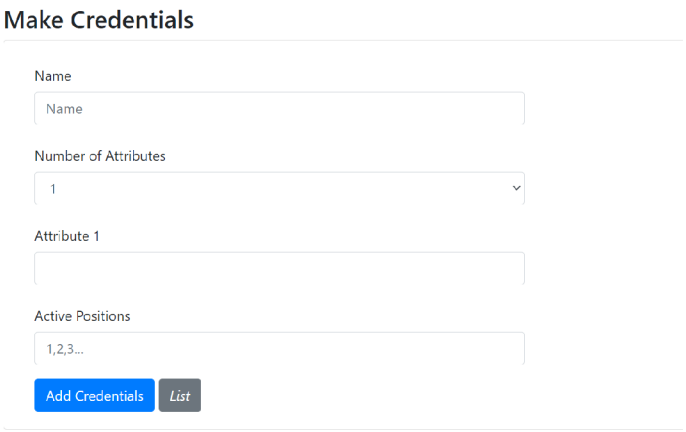
5.3.1 Credentials

Po úspěšném přihlášení se uživatel dostane do sekce „Credentials“, viz obr. 5.7.



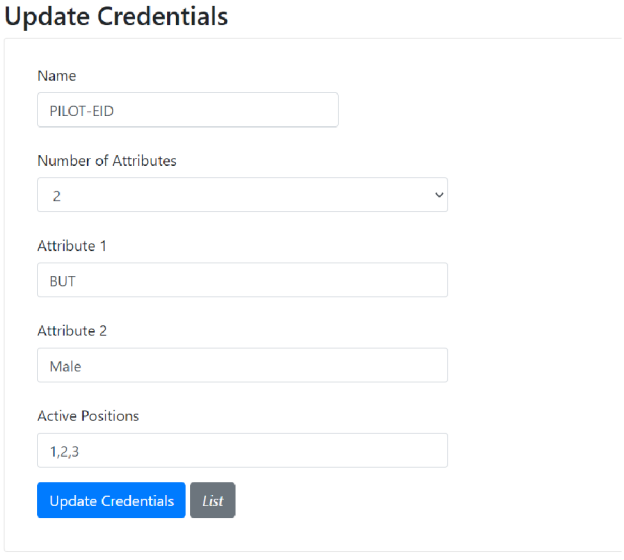
Obr. 5.7: Credentials

Tato karta slouží ke správě ověřovaných atributů. Je zde možné tvořit pověření (obr. 5.8), editovat vytvořené a mazat již nežádoucí. Celá databáze se dá v případě potřeby stáhnout kliknutím na tlačítko „Save Database“. Vytvořené pověření se aktivuje zeleným tlačítkem „power on“ a u aktivovanáho pověření se zobrazí status „ACTIVE“, který značí aktuálně ověřované atributy (obr. 5.8).



Obr. 5.8: Make Credentials

Již vytvořená pověření lze měnit (obr. 5.9) kliknutím na tlačítko „Edit“.

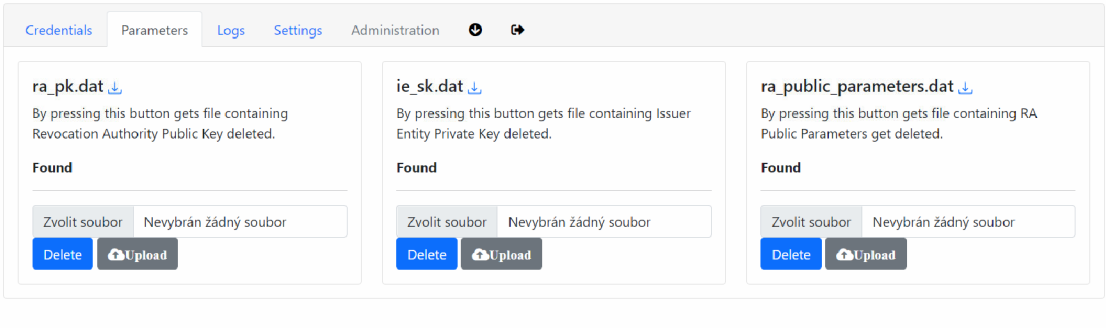


Obr. 5.9: Update Credentials

Stisknutím tlačítka „List“ je uživatel vrácen zpět na přehled pověření (obr. 5.7).

5.3.2 Parameters

Sekce slouží ke správě kryptgrafických klíčů, umožňuje importovat, exportovat a mazat kryptografické klíče ze systému. Indikuje také přítomnost kryptografických klíčů stavy „Found“ a „Not found“. Tato sekce je kritická pro součinnost entity ověřovatele s ostatními entitami systému řízení přístupu. Sekce je vyobrazena na obr. 5.10.

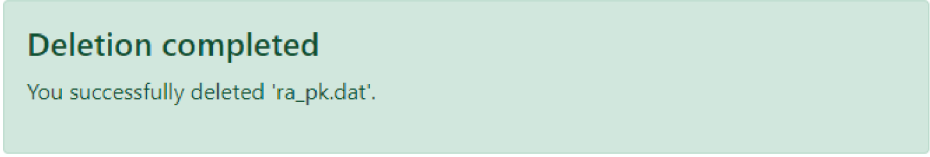


Obr. 5.10: Parameters

Pokud uživatel nevybere žádný soubor k nahrání, je vrácena informativní hláška „Empty file“, viz obr. 5.11.

Obr. 5.11: Empty file

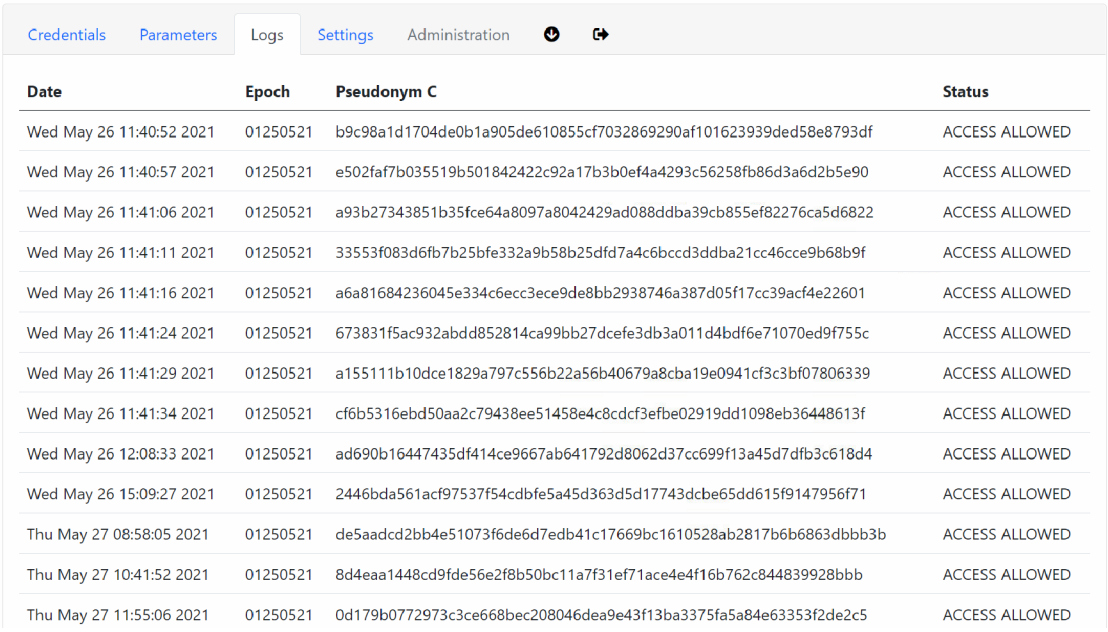
Pokud uživatel odstraní kryptografický klíč, je vrácena informativní hláška „Deletion completed“, viz obr. 5.12.



Obr. 5.12: Deletion completeted

5.3.3 Logs

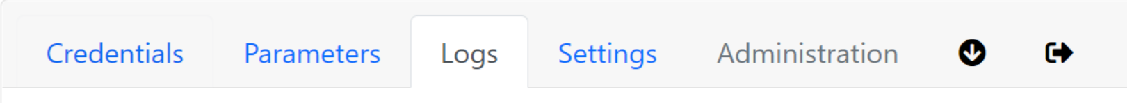
Na obr. 5.13 je zobrazena sekce „Logs“. Zaznamenávají se zde jednotlivé pokusy o autentizaci. Výpis v uživatelském rozhraní je omezen na 50 posledních pokusů. Vypisuje se zde datum, číslo epochy, pseudonym a výsledek autentizace.



Date	Epoch	Pseudonym C	Status
Wed May 26 11:40:52 2021	01250521	b9c98a1d1704de0b1a905de610855cf7032869290af101623939ded58e8793df	ACCESS ALLOWED
Wed May 26 11:40:57 2021	01250521	e502faf7b035519b501842422c92a17b3b0ef4a4293c56258fb86d3a6d2b5e90	ACCESS ALLOWED
Wed May 26 11:41:06 2021	01250521	a93b27343851b35fce64a8097a8042429ad088ddba39cb855ef82276ca5d6822	ACCESS ALLOWED
Wed May 26 11:41:11 2021	01250521	33553f083d6fb7b25bfe332a9b58b25dfd7a4c6bccd3ddba21cc46cce9b68b9f	ACCESS ALLOWED
Wed May 26 11:41:16 2021	01250521	a6a81684236045e334c6ecc3ece9de8bb2938746a387d05f17cc39acf4e22601	ACCESS ALLOWED
Wed May 26 11:41:24 2021	01250521	673831f5ac932abdd852814ca99bb27dcefe3db3a011d4bdf6e71070ed9f755c	ACCESS ALLOWED
Wed May 26 11:41:29 2021	01250521	a155111b10dce1829a797c556b22a56b40679a8cba19e0941cf3c3bf07806339	ACCESS ALLOWED
Wed May 26 11:41:34 2021	01250521	cf6b5316ebd50aa2c79438ee51458e4c8cdcf3efbe02919dd1098eb36448613f	ACCESS ALLOWED
Wed May 26 12:08:33 2021	01250521	ad690b16447435df414ce9667ab641792d8062d37cc699f13a45d7dfb3c618d4	ACCESS ALLOWED
Wed May 26 15:09:27 2021	01250521	2446bda561acf97537f54cdbfe5a45d363d5d17743dcbe65dd615f9147956f71	ACCESS ALLOWED
Thu May 27 08:58:05 2021	01250521	de5aadcd2bb4e51073f6de6d7edb41c17669bc1610528ab2817b6b6863dbbb3b	ACCESS ALLOWED
Thu May 27 10:41:52 2021	01250521	8d4eaa1448cd9fde56e2f8b50bc11a7f31ef71ace4e4f16b762c844839928bbb	ACCESS ALLOWED
Thu May 27 11:55:06 2021	01250521	0d179b0772973c3ce668bec208046dea9e43f13ba3375fa5a84e63353f2de2c5	ACCESS ALLOWED

Obr. 5.13: Logs

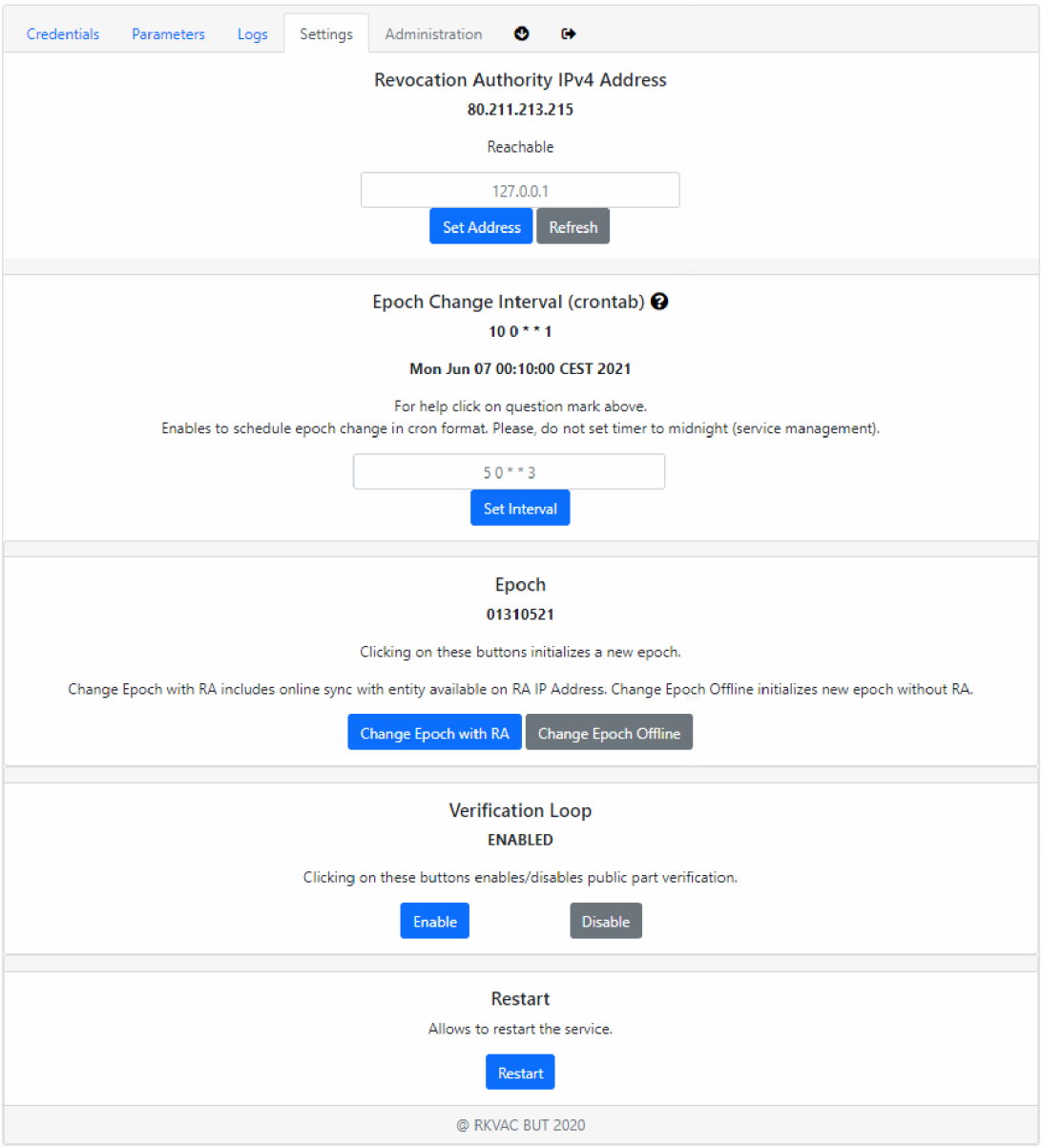
V menu (obr. 5.14) se nachází ikona šipky dolů, která umožňuje stáhnout aktuálně denní log a šipka značící odchod pro odhlášení z administrace.



Obr. 5.14: Menu

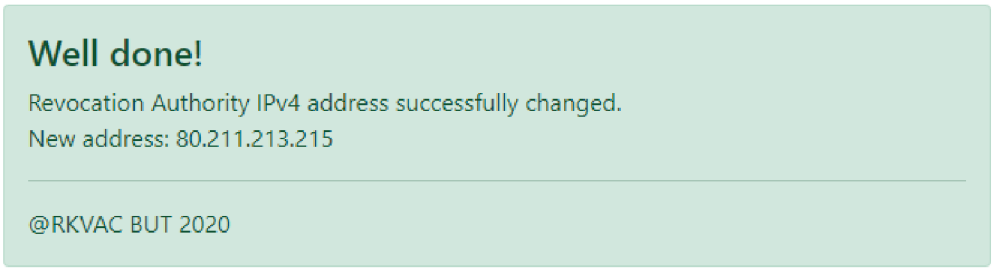
5.3.4 Settings

Na obr. 5.15 je vyobrazena sekce „Settings“. Sekce slouží k administrativní obsluze entity ověřovatele. Je zde možné nastavit IP adresu revokační autority, nastavit interval přechodu na novou epochu, dále je zde možné manuálně vynutit změnu epochy, spravovat ověřování a v případě potřeby restartovat službu.



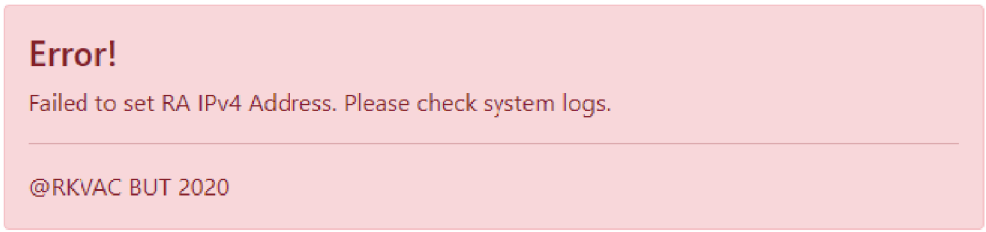
Obr. 5.15: Settings

Stiskem tlačítka „Set Address“ lze změnit IP adresu revokační autority, pokud je změna provedena úspěšně, je vrácena uživateli informativní hláška značící úspěch (obr. 5.16).



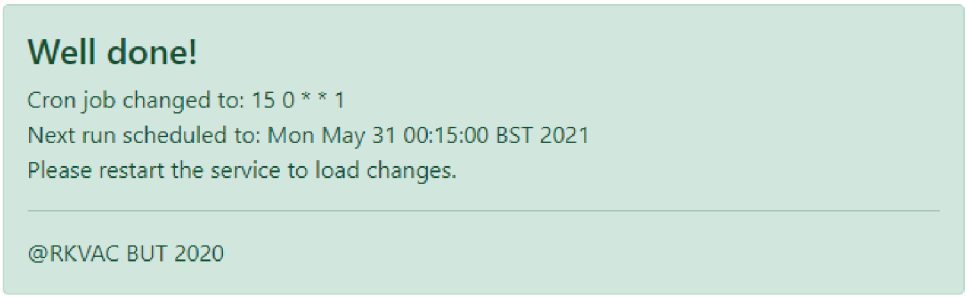
Obr. 5.16: Successful IP address change

V opačném případě je uživateli vrácena hláška informující o chybě, viz 5.17



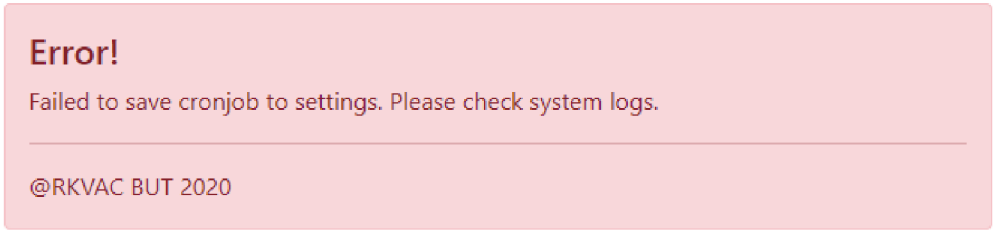
Obr. 5.17: Error IP address change

Stiskem tlačítka „Set Interval“ lze nastavit interval automatického přechodu na novou epochu. Pokud nastavení je provedeno úspěšně, je vrácena uživateli informativní hláška značící úspěch (obr. 5.18).



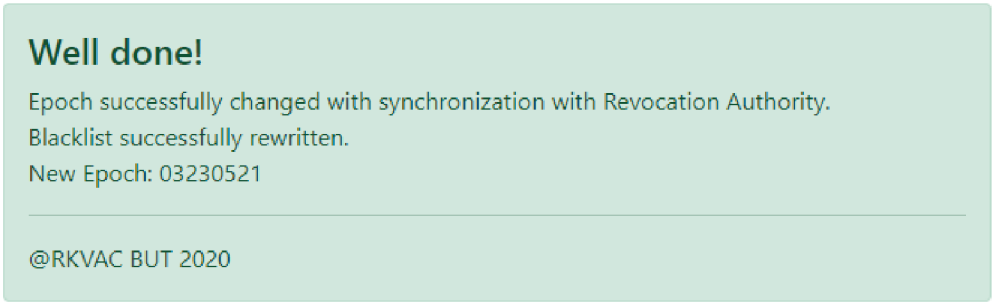
Obr. 5.18: Successful epoch change interval set

V opačném případě je vrácena uživateli hláška informující o chybě, viz obr. 5.19.



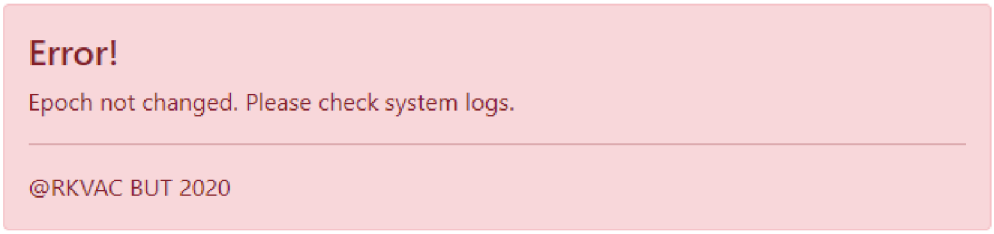
Obr. 5.19: Error epoch change interval set

Stiskem tlačítka „Change Epoch with RA“ nebo „Change Epoch Offline“ lze manuálně vynutit přechod na novou epochu. Pokud je přechod proveden úspěšně, je vrácena uživateli informativní hláška značící úspěch (obr. 5.20).



Obr. 5.20: Successful manual epoch change

V opačném případě je vrácena uživateli hláška informující o chybě, viz obr. 5.21.

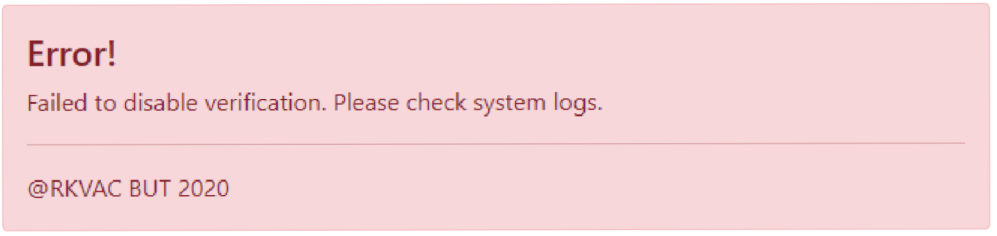


Obr. 5.21: Error manual epoch change

Stisknutím tlačítka „Enable“ nebo „Disable“ lze zapnout nebo vypnout ověřovací smyčku. Pokud je smyčka vypnutá, nedochází ke čtění ze čtečky karet. Pokud je nastavení provedeno úspěšně, je vrácena uživateli informativní hláška značící úspěch (obr. 5.22).

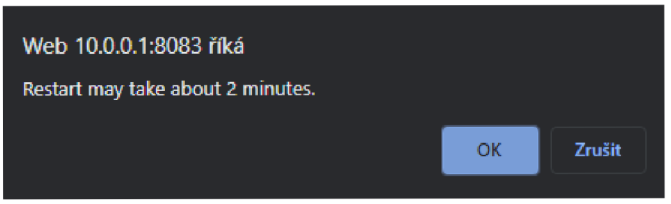
Obr. 5.22: Successfully disabled verification loop

V opačném případě je vrácena uživateli hláška informující o chybě, viz obr. 5.23.



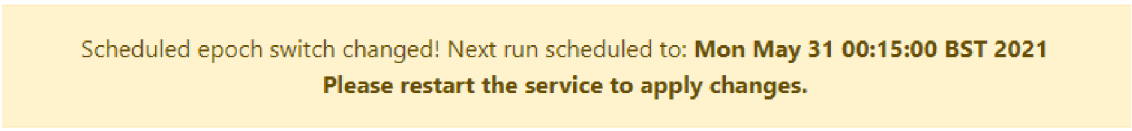
Obr. 5.23: Error disabling verification loop

Tlačítko „Restart“ umožňuje restartovat službu. Po stisknutí je uživateli zobrazeno dialogové okno, viz obr. 5.24.



Obr. 5.24: Restart dialog window

Restartovat službu je nutné, zobrazuje-li se v nastavení oranžový infobox, viz. obr. 5.25.



Obr. 5.25: Restart warning

Pokud uživatel potvrdí volbu dialogového okna, přejde se k restartování služby, viz obr. 5.26.

Obr. 5.26: Restart

Po úspěšném restartování služby je opět uživatel vyzván k přihlášení (obr. 5.6)

6 Implementace přístupového systému

Navržený přístupový systém je založen na platformě **Raspberry Pi**. V přístupovém systému je využita jako autentizační token **čipová karta MultOS**. Pro komunikaci mezi mikropočítačem a čipovou kartou je užita **čtečka karet HID OMNIKEY CARD Smart Card Reader** a **C implementace RKVAC protokolu** běžící na operačním systému GNU/Linux, kterou obsluhuje vytvořená **webová aplikace ověřovatele**.

6.1 Testování a pilotní provoz

Přístupový systém komunikuje s revokační autoritou vytvořenou kolegou O. Malíkem a také pomocí TLS spojení s modulem CKP.40, který zajišťuje mj. otevírání fyzického zámku dveří. Systémy jsou vzájem plně kompatibilní. Testování implementace je zobrazeno na obr. 6.1

Obr. 6.1: Testování přístupového systému

Po přijatelném testování byl přístupový terminál z iniciativy školitele spuštěn do pilotního provozu a namontován na zeď na FEKT VUT do 5. poschodí, viz obr. 6.2.

Obr. 6.2: Pilotní provoz přístupového systému

Udělení oprávnění ke vstupu do denní místnosti je vyobrazeno na obr. 6.3

Obr. 6.3: Povolení přístupu při pilotním provozu

Zamítnutí přístupu revokovanému uživateli při pilotním provozu je vyobrazeno na obr. 6.4

Obr. 6.4: Zamítnutí přístupu při pilotním provozu

6.2 Použitý hardware a software

V této kapitole je uveden a popsán HW a sofware použitý pro implementovaný systém řízení přístupu.

6.2.1 Raspberry Pi

Raspberry Pi je miniaturní počítač o velikosti kreditní karty, který byl vyvinut britskou nadací The Raspberry Pi Foundation. Cílem projektu bylo zpřístupnit výpočetní a digitální tvorbu s vysokým výkonem a svobodným softwarem především školám a rozvojovým zemím za minimální finanční náklady.

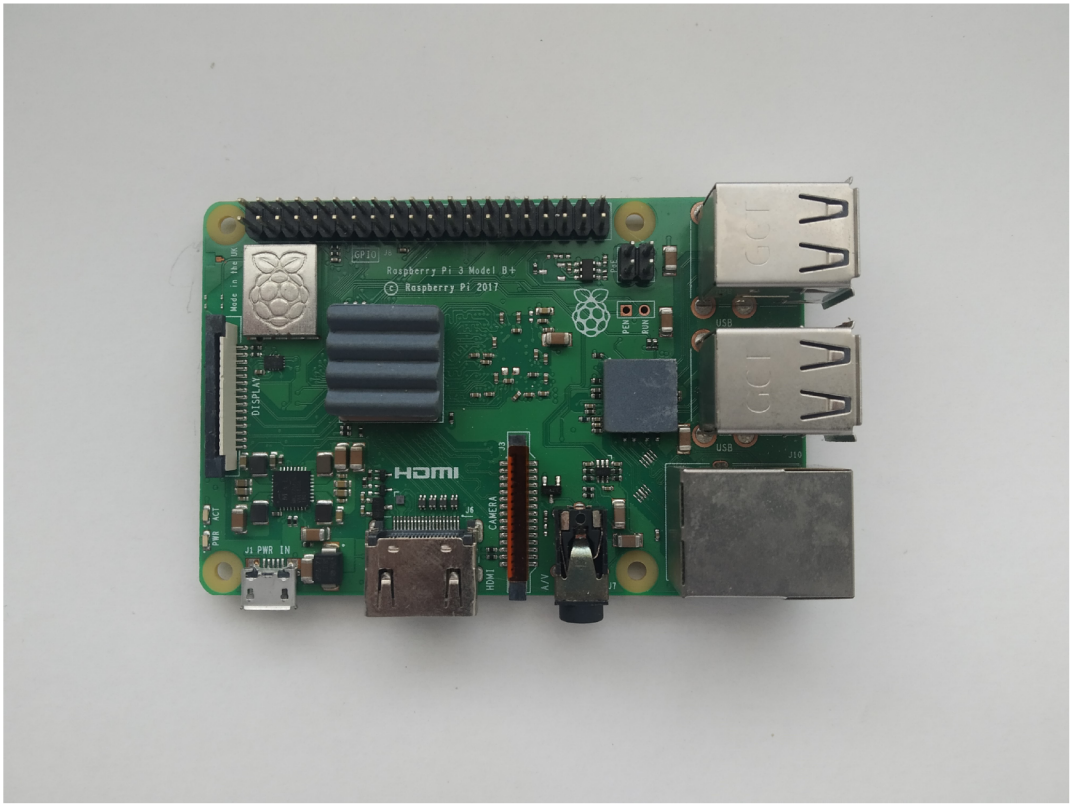
Momentálně jsou k dispozici 4 různé generace s rozdílným výkonem a hardwarovou modifikací. Modely A reprezentují odlehčenou verzi dané generace, modely B reprezentují standardní HW modifikaci a modely B+ reprezentují vylepšené verze. Systém řízení přístupu je realizován na modelu Raspberry Pi 3 Model B+ , jehož HW specifikace jsou uvedeny v tab. 6.1.

Tab. 6.1: HW specifikace Raspberry Pi 3 Model B+

Název	Parametr
`Procesor (CPU)`	1.4 GHz 64-bit quad-core ARM Cortex-A53
`Paměť (SDRAM)`	1 GB (sdílená s GPU)
`Video (GPU)`	Broadcom VideoCore IV @ 250 MHz
`Video výstup`	HDMI (rev 1.3, 1.4) až do 1920x1200
`Interní paměť`	MicroSDHC, USB Boot Mode
`Integrovaná síť`	Gigabit Ethernet + WiFi 802.11ac + Bluetooth 4.2 BLE
`Výkon`	1,5W bez zátěže

Z hlediska operačních systémů, které lze na raspberry provozovat, je díky poměrně velké skupině nadšenců hned celá řada možností. Nejoblíbenějším operačním systémem je Raspberry PI OS hlavně díky přímé podpoře od Raspberry Pi Foundation. OS je založený na distribuci Debianu a dříve byl známý pod názvem Raspbian. Dalšími možnostmi jsou např. Ubuntu server, Arch Linux ARM, Windows 10 IoT Core pro prototypy IoT zařízení, OSMC, OpenSUSE, Kali Linux, Gentoo Linux, FreeBSD a mnoho dalších [12].

Systém řízení přístupu je realizován z preference maximalizování HW prostředků na minimalistickém operačním systému Raspberry PI OS Lite, který disponuje CLI a minimálními sety balíčků. Tento operační systém je nainstalován na Raspberry Pi 3 Model B+ (obr. 6.5).



Obr. 6.5: Raspberry Pi 3 Model B+

6.2.2 Čipová karta MultOS

Čipové karty MultOS jsou hojně využívány např. v oblasti bankovnictví především díky své bezpečnosti. Jejich operační systém MultOS je považován za jeden z nejvíce bezpečných systémů. Umožňuje používat více aplikací, což zajišťuje spolehlivou platformu pro multifunčkní použití. Aplikace lze kdykoliv nahrát či mazat, což je výhodné z hlediska opětovného použití čipových karet. Aplikace mají také přístup pouze ke svým datům, přístup k datům jiných aplikací je zamítnut. Další výhodou těchto karet je, že lze využívat vysokoúrovňového programování při volání nízkoúrovňových kódů, což umožňuje jejich pohodlnější programování [13].

Čipová karta byla dodána již z předinstalovanou aplikací klientské části RKVAC protokolu. Komunikace s čipovou kartou je prostřednictvím APDU zpráv a dodané knihovny RKVAC protokolu.

6.2.3 Knihovna RKVAC protokolu

Pro vypracování webové aplikace byla poskytnuta vedoucím práce knihovna RKVAC protokolu, která implementuje technologii CDDH19 vyvíjenou na VUT v Brně [14].

Implementace je provedena na operačním systému GNU/Linux. Knihovna implementuje entity uživatele, vydavatele, revokační autority a ověřovatele.

6.2.4 Modul CKP.40

Modul CKP.40 je zařízení poskytující informace o zamítnutí/povolení autentizované karty. Má tedy funkci autorizační. Plní tak mj. kontrolující funkci otevírání fyzického zámku dveří. Při ověřování uživatele jsou odhalené atributy aplikací zřetězeny a zahashovány funkcí SHA-2. Výsledný hash je oříznut na 7 bytů a ty jsou posílány na schválení modulu CKP.40, viz výpis 5.1. Zpráva *1010006* značí kartu povolenou, zpráva *1010015* kartu zamítnutou.

6.2.5 Webová aplikace

Jedná se webovou aplikaci vytvořenou v rámci této práce. Aplikace je zkompilována v jar balíčku, který lze najít ve zdrojových kódech v příloze. Webovou aplikaci lze spustit kdekoliv, kde je nainstalován Java JDK 11. Spuštění se provádí příkazem, viz níže.

```
$ java -jar verifier.jar
```

Aplikace je defaultně spuštěna na portu *8083*. Spouštěcí lze v případě potřeby měnit při spuštění parametrem *–server.port* a tls port pro komunikaci s modulem CKP.40 parametrem *–tls.port*, viz níže.

```
$ java -jar verifier.jar --server.port=XXXX --tls.port=YYYY
```

V závislosti na knihovně RKVAC protokolu je nutné pro spolehlivý chod systému, aby se zkompilovaná knihovna RKVAC protokolu (spustitelný soubor rkvac-protocol-multos-1.0.0) nacházel v následujícím adresáři, viz níže.

```
/home/rkvac/rkvac_web/rkvac-protocol/build
```

Aplikace také přistupuje k souborům knihovny RKVAC protokolu, které jsou uvedeny níže.

- `data/Verifier/ve_requests.log`
- `data/Verifier/ve_epoch.dat`

Konfigurace aplikace jako systémové služby

Pro ukládání denních logů je možné konfigurovat aplikaci jako systémovou službu pomocí démonů systemd upravením souboru */etc/systemd/system/rkvac-verifier.service* následujícím způsobem (výpis 6.1)

Výpis 6.1: Konfigurace systemd

```
[Unit]
Description=RKVAC Verifier service

[Service]
WorkingDirectory=/home/rkvac/rkvac_web/
ExecStart=/bin/bash -c 'java -jar /home/rkvac/rkvac_web/verifier.jar >>
/home/rkvac/rkvac_web/logs/verifier_$$(date +%%d-%%m-%%Y).log 2>&1 '
ExecStop=/bin/bash -c 'pkill -9 rkvac-protocol'
KillMode=control-group
TimeoutStopSec=3
User=root
Type=simple
Restart=on-failure
RestartSec=3

[Install]
WantedBy=multi-user.target
```

V tomto případě je nutné mít aplikaci verifier.jar umístěnou v adresáři, viz níže.

```
/home/rkvac/rkvac_web/
```

A také je nutné vytvořit adresář *logs* pro samotné logy.

```
/home/rkvac/rkvac_web/logs
```

Poté je nutné povolit službu a restartovat démona pro aktualizaci změn, viz níže.

```
$ sudo systemctl enable rkvac-verifier.service
$ sudo systemctl daemon-reload
```

Nyní je vše připravené ke spuštění služby, viz níže.

```
$ sudo systemctl start rkvac-verifier.service
```

Spuštěnou službu lze vidět na obr. 6.6.

```
rkvac@raspberrypi:~ $ sudo systemctl status rkvac-verifier.service
● rkvac-verifier.service - RKVAC Verifier service
   Loaded: loaded (/etc/systemd/system/rkvac-verifier.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2021-05-31 00:00:04 CEST; 6h ago
 Main PID: 13670 (bash)
    Tasks: 41 (limit: 4915)
   CGroup: /system.slice/rkvac-verifier.service
           ├─13670 /bin/bash -c java -jar /home/rkvac/rkvac_web/verifier.jar >> /home/rkvac/rkvac
           ├─13671 java -jar /home/rkvac/rkvac_web/verifier.jar
           └─13731 ./rkvac-protocol-multos-1.0.0 -v

May 31 00:00:04 raspberrypi systemd[1]: Started RKVAC Verifier service.
lines 1-11/11 (END)
```

Obr. 6.6: Konfigurace systemd

Zbývá už jen nastavit denní restart služby pomocí démona cron, aby se aktualizoval soubor s denními logy. viz níže.

```
$ sudo crontab -e
```

A vložit záznam o půlnočním restartu služby, viz níže.

```
0 0 * * * /bin/systemctl restart rkvac-verifier.service
```

Nyní se bude konfigurovaná služba restartovat každou půlnoc, čímž bude docházet k vytvoření nového souboru s aplikačními záznamy.

Kompilace webového serveru

Webový server lze v případě potřeby zkompilovat přímo ze zdrojových kódů. Stačí vstoupit do složky s projektem, spustit příkazový řádek a zavolat příkaz:

```
C:\Users\User\Downloads\RKVAC_Verifier> .\gradlew clean build
```

Projekt se poté zkompiluje v adresáři `.\build\libs` .

Závěr

Cílem této práce byla implementace přístupového systému využívající mikropočítač Raspberry Pi a autentizační token typu čipová karta, chytrý telefon nebo chytré hodinky založeném na atributových pověřeních s revokací, který implementovatuje entity vydavatele, revokační autoritu, uživatele a ověřovatele.

Realizace byla úspěšná, výsledný přístupový systém využívá autentizační token typu čipová karta, implementuje entity uživatele, ověřovatele a komunikuje s revokační autoritou a entitou vydavatele, které byly vytvořené kolegou O. Malíkem. Během vývoje jsme spolu problematiku konzultovali a docílili navzájem plně kompatibilní implementace. Zpočátku jsme řešili obdobné problémy, proto jsme se v průběhu poloviny práce zaměřili každý na konkrétní část implementace systému. Kolega Malík řešil problematiku přístupového systému z hlediska připojení entity uživatele přes internet. Já jsem se zaměřil na tvorbu entity ověřovatele pro lokální, ověřovací terminál pro fyzické řízení přístupu (otevírání zámku dveří).

Největší výzvou bylo zpočátku získat přehled o možnostech využití různých technologií a získat představu o principu fungování webových aplikací. Dále bylo nutné zajistit komunikaci s uživatelem, poté také komunikaci se poskytnutou implementací RKVAC protokolu, komunikaci s modulem CKP.40 a nakonec odladit všechny chybové stavy, které v průběhu testování systému vznikaly. Nakonec byla implementace z iniciativy školitele nasazena v 5. podlaží FEKT VUT v Brně do pilotního provozu.

Literatura

[1] BURDA, Karel.: *Kryptografie okolo nás*. Praha: CZ.NIC, 2019. ISBN 978-80-88168-52-2.

[2] HAJNÝ, J.; DZURENDA, P.; CASANOVA MARQUÉS, R.; MALINA, L.: *Privacy ABCs: Now Ready for Your Wallets!* In Proceedings of The 19th International Conference on Pervasive Computing and Communications (IEEE PerCom 2021). 2021. s. 686-691. ISBN: 978-1-6654-0424-2.

[3] *RFC 2616* Hypertext Transfer Protocol – HTTP/1.1 [online]. [cit. 08.12.2020]. Dostupné z: <`https://tools.ietf.org/html/rfc2616`>

[4] *Cryptlib* Download [online]. [cit. 11. 12.2020]. Dostupné z: <`https://www.cs.auckland.ac.nz/~pgut001/cryptlib/download.html`>

[5] *Libgcrypt* dev.gnupg.org [online]. [cit. 11. 12. 2020]. Dostupné z: <`https://dev.gnupg.org/source/libgcrypt/browse/master/NEWS; libgcrypt-1.8.7`>.

[6] *Libsodium* libsodium-doc [online]. [cit. 11. 12. 2020]. Dostupné z: <`https://github.com/jedisct1/libsodium-doc`>.

[7] *OpenSSL* Downloads [online]. [cit. 11. 12. 2020]. Dostupné z: <`https://www.openssl.org/source/`>.

[8] *WolfCrypt* DOCS [online]. [cit. 11. 12. 2020]. Dostupné z: <`https://www.wolfssl.com/docs/`>.

[9] *FIPS PUB 140-2* Security Requirements for Cryptographic Modules [online]. [cit. 11. 12. 2020]. Dostupné z: <`https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf`>.

[10] *NIST SP 800-38A* Recommendation for Block Cipher Modes of Operation Methods and Techniques [online]. [cit. 11. 12. 2020]. Dostupné z: <`https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf`>.

[11] *Raspberry Pi Documentation: FAQs.* What is a Raspberry Pi? [online]. [cit. 06. 12. 2020]. Dostupné z: <`https:/www.raspberrypi.org/documentation/faqs/`>.

[12] *Operating systems for Raspberry Pi* 20 Best Operating Systems You Can Run on Raspberry Pi in 2020 [online]. [cit. 06. 12. 2020]. Dostupné z: <`https://bit.ly/2VHVA4j`>.

[13] Vančo, P.: *Kryptografická podpora současných programovatelných čipových karet* [online]. Brno, 2019 [cit. 2020-12-06]. Dostupné z: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=192563>.

[14] *Odborná zpráva o postupu prací a dosažených výsledcích za rok 2019* Příloha k závěrečné zprávě za rok 2019 [online] Evropská 1692/37 160 00 Praha 6, 2019 [cit. 2020-12-06]. Dostupné z: <https://vutbr-my.sharepoint.com/personal/dzurenda_vutbr_cz/>.

[15] *The Raspbberry Pi Foundation: Raspberry Pi OS Lite* [software]. 20. října 2020 [cit. 2020-12-06]. Dostupné z: <https://www.raspberrypi.org/software/operating-systems/>.

[16] *Cmake* Latest Release (3.19.1) [online]. [cit. 06. 12. 2020]. Dostupné z: <https://cmake.org/download/>.

[17] *MCL* A portable and fast pairing-based cryptography library [software]. 6. prosince 2020 [cit. 06. 12. 2020]. Dostupné z: <https://github.com/herumi/mcl>.

[18] VUT v Brně: *rkvac-protocol* README.md [online]. 19. 10. 2020 [cit. 06. 12. 2020]. Dostupné z: <https://vutbr-my.sharepoint.com/personal/dzurenda_vutbr_cz/>.

Seznam symbolů, veličin a zkratek

CLI	Command Line Interface
CSS	Cascade Style Sheets
DH	Diffie Hellman
DSA	Digital Signature Algorithm
ECDH	EclipticCurve Diffie Hellman
FEKT	Fakulta elektrotechniky a komunikačních technologií
FIPS	Federal Information Processing Standard Publication
GDPR	General Data Protection Regulation
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
JNI	Java Native Interface
MAC	Message Authentication Code
NDK	Native Development Kit
NIST	National Institute of Standards and Technology
RA	Revocation Atuhority
REST	Representational State Transfer
RKVAC	Revocable Keyed-Verification Anonymous Credentials
RSA	River Shamir Adleman
VUT	Vysoké učení technické v Brně

Seznam příloh

A Obsah přiloženého CD

A.1 RKVAC-Verifier.zip