Porovnání kryptografických algoritmů pro potřeby vývoje webových aplikací

Erik Faltynek

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Erik Faltynek**
Osobní číslo:	**A19020**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Porovnání kryptografických algoritmů pro potřeby vývoje webových aplikací**
Téma práce anglicky:	**Comparison of Cryptographic Algorithms for Web Application Development**

Zásady pro vypracování

1. Nastudujte a rozveďte problematiku vývoje webových aplikací v kontextu kryptografických algoritmů.
2. Sepište aktuálně využívané moderní algoritmy pro zabezpečení webových aplikací.
3. Rozepište výhody i nevýhody stávajících řešení a porovnejte je.
4. Zpracujte seznam doporučených kryptografických technologií a knihoven pro vývoj moderních webových aplikací.
5. Výsledky vhodně prezentujte.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. MOLHANEC, Martin. Webové metodiky. Praha: Alfa Nakladatelství, 2014, 210 s. Informatika. Monografie. ISBN 9788087197844.
2. LEK, Kamol a Naruemol RAJAPAKSE, ed. Cryptography: protocols, design, and applications. New York: Nova Science Publishers, c2012, ix, 242 s. Cryptography, steganography and data security. ISBN 9781621007791.
3. BURDA, Karel. Kryptografie okolo nás. Praha: CZ.NIC, z.s. p.o., 2019, 128 s. CZ.NIC. ISBN 978-80-88168-49-2. Dostupné také z: https://knihy.nic.cz/files/edice/Kryptografie_okolo_nas.pdf
4. MALEH, Yassine. Security and privacy management, techniques, and protocols. Hershey PA: IGI Global, [2018], 1 online zdroj. Advances in information security, privacy, and ethics (AISPE) book series. Dostupné z: doi:9781522555841.
5. TILBORG, Henk C. A. van a Sushil JAJODIA, ed. Encyclopedia of cryptography and security. 2nd ed. New York: Springer, c2011, xl, 1416 s. Springer reference. Dostupné z: doi:9781441959065.

Vedoucí bakalářské práce: **Ing. Petr Žáček, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne Erik Faltynek, v. r.

ABSTRAKT

Tato bakalářská práce se věnuje srovnáním technologií, určených pro vývoj bezpečných webových aplikací. V teoretické části jsou představeny základní pojmy a doporučení spjaté s vývojem webových aplikací. Dále jsou rozebrány kryptografické aspekty a doporučené algoritmy. Je i zmíněna post-kvantová kryptografie s jejími algoritmy ve fázi standardizace. Poté jsou představeny technologie pro bezpečný vývoj webových aplikací. V praktické části práce srovnává vybrané technologie, se zjišťováním výskytu, doporučených algoritmů v jejích knihovnách. Výstupem práce je seznam doporučených technologií a knihoven pro bezpečný vývoj webových aplikací.

Klíčová slova: webová aplikace, vývoj aplikací, kryptografie, bezpečnost, framework, back-end

ABSTRACT

This bachelor thesis deals with the comparison of technologies designed for the development of secure web applications. The theoretical part introduces the basic concepts and recommendations associated with the development of web applications. Cryptographic aspects and recommended algorithms are also discussed. Post-quantum cryptography with its algorithms in the standardization phase is also mentioned. Then, technologies for secure web application development are introduced. The practical part of the work compares selected technologies, with determining the occurrence of recommended algorithms in its libraries. The output of the work is a list of recommended technologies and libraries for secure web application development.

Keywords: web application, application development, cryptography, security, framework, back-end

Tímto bych chtěl poděkovat svému vedoucímu práce Ing. Petru Žáčkovi, Ph.D. za cenné rady a konzultace ohledně bakalářské práce.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

V dnešní době každý potřebuje mít zabezpečené osobní údaje a přenášená data. Jednou z hrozeb je jejich možná kompromitace či odcizení. Kryptografie je jeden z možných způsobů na minimalizaci tohoto rizika, a proto je důležité, aby vývojáři měli správné nástroje a základní znalosti v oboru kryptografie pro jejich bezpečnou implementaci. Avšak tyto nástroje mohou disponovat již zastaralými algoritmy anebo může dojít ke špatné implementaci, což by mělo za následek ohrožení chráněných dat.

Cílem této bakalářské práce je sestavit seznam doporučených technologií s ohledem na bezpečnost, který pomůže budoucím vývojářům webových aplikací ve výběru. Proto je ze začátku teoretické části seznámení se základními pojmy a doporučení pro vývoj webové aplikace. Dále jsou popsány kryptografické algoritmy a s nimi spjaté doporučení v kontextu správné implementace a výběru algoritmů. Jelikož v následujících letech mohou přijít kvantové počítače, čímž by byly ohroženy některé standardizované algoritmy, tak budou i vypsané algoritmy, které jsou ve výběru pro budoucí post-kvantový standard algoritmů. Na konci teoretické části bude sestaven seznam vývojových technologií, tedy frameworků a knihoven, určený pro bezpečný vývoj webových aplikací.

V praktické části budou tyto technologie srovnávané na základně kritérií sestavených na konci teoretické části. Tyto technologie budou porovnávané na základě bezpečnosti a bude se brát ohled na výskyt doporučených algoritmů v jejich knihovnách. V závěru praktické části budou shrnuty výhody a nevýhody zmiňovaných řešení a doporučení pro výběr technologií.

I. TEORETICKÁ ČÁST

1 ÚVOD DO VÝVOJE WEBOVÝCH APLIKACÍ

V této části budou popsány základní pojmy v oblasti vývoje webových aplikací. Prvně je třeba rozlišit rozdíl mezi webovou stránkou a webovou aplikací, dále je popsán rozdíl mezi front-end a back-end vývojem, poté jsou popsány důležité kroky pro vývoj úspěšné webové aplikace a posledním a nejdůležitějším bodem je bezpečnost.

1.1 Rozdíl mezi webovou stránkou a webovou aplikací

Rozdíl mezi webovou stránkou a webovou aplikací se může zdát neznatelný pro mnoho lidí, jelikož sdílí několik podobností, zejména v kontextu front end technologií například HTML, CSS a JavaScript. Avšak webové aplikace jsou mnohem komplexnější nežli webové stránky, je zde mnoho klíčových rozdílů, především v požadavcích na interaktivitu a autentizaci [1].

Webová stránka je soubor vzájemně propojených „web pages“, které jsou dostupné na internetu pod jednou doménou. Jejich účelem je uživatele informovat, takže není kladen tak velký požadavek na interaktivitu. Takovým příkladem může být webová stránka blogu, podporování byznysu či osobní portfolio [1].

Webová aplikace je program, ke kterému uživatelé přistupují pomocí webového prohlížeče, kde s ním mohou provádět specifické funkce. Mohou například kliknout na nějaké tlačítko, nebo vyplnit nějaký formulář po kterém dostanou odezvu od samotné stránky v podobě stáhnutého dokumentu, elektronické platby nebo online chatu. Příklad webových aplikacích jsou bankovní aplikace, e-shopy, sociální sítě anebo online mapy [1]. Pro reprezentaci dat, využívá kombinaci server-side scriptů a client-side skriptů [2].

1.2 Client-side a server-side skripty

Funkce dynamických webových stránek jsou rozdělené na client-side a server-side skripty. Občas jsou tyto skripty přirovnávané k front-end a back-end vývoji [3].

Client-side skript je označován jako program, který je stažený ze serveru a následně zkompilován v prohlížeči na straně uživatele. Jedná se o malé programy, které se vyvíjí ve skriptovacím jazyce JavaScript, jenž je značně využíván v dynamických webových stránkách [3]. V dnešní době se ale stále častěji vyvíjí větší JavaScriptové aplikace v populárních knihovnách a frameworcích jako je Angular, Vue a React [4].

Server-side skript je zpracován na straně serveru a provádí se, když například uživatel na client-side straně požádá o informace, tak je jeho požadavek zaslán na server, který tuto

informaci zpracuje a následně zašle zpět požadovaná data. Tyto skripty mohou být spuštěny i když není webová stránka plně načtena, to je například v případě načítaní dynamických dat. Tohle využívají například sociální sítě anebo vyhledávače, jelikož disponují velkým množstvím dat, kde by bylo neefektivní stahovat všechny tyto informace, což by vedlo ke snížení výkonu načítaní stránky [3].

1.3 Druhy vývoje

Webový vývoj se dělí na dva pojmy, a to front-end a back-end. Tyto pojmy se řadí mezi nejpopulárnějšími výrazy ve světe vývoje webů. Je mezi nimi značný rozdíl a je potřeba, aby jednotlivé strany komunikovaly a fungovaly efektivně, jakožto samostatná jednotka [5].

1.3.1 Front-end vývoj

Front-end vývoj se vztahuje k částí webové stránky, se kterou uživatel interaguje. Do toho se vztahuje vše, co uživatel vidí na obrazovce, jako je struktura, design například barva textu, obrázky, grafy, tlačítka, navigační menu a chování, když se webová stránka načte. Tyto věci jsou implementovány front-end vývojářem, kde jsou jeho hlavní cíle responzivita a výkon, protože je důležité, aby se webová stránka rychle a správně načetla na všech velikostech zařízení. Pro tento vývoj se zejména využívají jazyky HTML, CSS a JavaScript. Pro usnadnění vývoje se používají frameworky a knihovny, nejpopulárnější z nich jsou například Angular, React, jQuery a Vue [5].

1.3.2 Back-end vývoj

Back-end vývoj se na druhou stranu vztahuje k části aplikace, kterou uživatel nevidí. Do toho se vztahuje například ukládání a organizování dat a komunikace. Kde na straně front-endu například dojde k provedení nákupu nebo vyplnění kontaktního formuláře, tak je zaslán požadavek na back-end, ten ho zpracuje a zašle zpět odpověď, kterou front-end zobrazí.

Když se na webové stránce mění obsah, na základě uložených dat v databázi a je možno i měnit tyto informace uživatelem, tak se jedná o dynamickou webovou stránku. Mezi nejvýznamnější back-endové jazyky patří Ruby, PHP, Java, C#, JavaScript (Node.js) a Python. Stejně jako u front-endové části, tak i zde se využívají k vývoji back-endu různé frameworky, aby se usnadnil vývoj a zvýšila se tak efektivita [6].

Na straně back-endu se také řeší databáze, která je důležitá pro ukládání dat dané stránky. Data jsou ukládána ve struktuře, která usnadňuje čtení, úpravu, organizaci a ukládání dat.

Celý back-end i s databázi je spuštěný na vzdáleném počítači zvaný server. Na výběr je celá řada databázových systému, například MySQL, SQL Server, PostgresSQL a Oracle [6]. Tyto databáze se nadále dělí na relační a nerelační.

1.4 Vývoj webové aplikace

Ze začátku vývoje je nejdůležitější si vybral vhodný technologický stack. Tím je myšleno, vybrání si programovacího jazyka, front-end a back-end frameworku, vhodné knihovny pro implementaci, databázový systém a server. Vhodný technologický stack by měl umožňovat škálovat aplikaci v případě, kdy se návštěvnost stránky zvyšuje, a tak je zapotřebí provádět aktualizace. Pro snadnější aktualizace je zapotřebí udržovat kód konzistentní a čitelný, jelikož se tak předchází chybovostí staršího kódu a také je přehlednější pro vývojáře, kteří se nezapojovali ve vývoji v počátečních fázích. Další krokem je navrhnout kvalitní uživatelské rozhraní a uživatelskou zkušenost, jelikož atraktivní design přiláká více nových uživatelů a uživatelská zkušenost je klíčem k udržení spokojenosti a loajality. Jedním z nejsilnějších trendu v roce 2022 ve vývoji webových aplikací je kreativní vizualizace dat, protože webové animace jsou na první pohled přitažlivější. To se váže k popularitě, jelikož spokojení uživatelé budou kvalitní webovou aplikaci sdílet a doporučovat. Pro vybudování solidního zdroje nových zákazníků, je důležité dodržovat pokynů SEO, podle kterých se vyhledávací algoritmy řídí, když indexují stránky v internetových vyhledávačích. Dále je důležitě udělat webovou aplikaci responzivní na zařízení s menší obrazovkou. Důležitým aspektem je také výkon, jelikož když se stránka načítá 1 až 3 sekundy, tak se šance na opuštění stránky uživatelem zvyšuje o 32 %. Rychlost načítání stránky je spojený s výběrem technologického stacku, obsahu, architektury webu a dynamických prvků. Proto je důležité stránku neustále testovat a optimalizovat. Pokud webová aplikace potřebuje pravidelnou aktualizaci kontentu, tak je vhodné implementovat kvalitní CMS, který umožní efektivně spravovat obsah v aplikaci [188].

Dalším důležitým aspektem je zabezpečení, jelikož jsou celosvětově kybernetické útoky každým rokem častější a sofistikovanější, v některých případech jsou i vykonávaný útočníky, které podporuje stát. Kvůli vysokému počtu digitálních systému v osobním životě a podnicích mohou mít útoky za následek škody ve vysokých částkách a v některých případech mít katastrofální či fatální důsledky. Proto je důležité bezpečnost zahrnovat ve vývoji jíž od počátku plánování až po samotný provoz. Mnoho společnosti zahrnuje do vývoje praktiky DevOps a DevSecOps [8].

Pojem DevOps, je spojení dvou slov, a to „development“ a „operations“. Cílem této agilní praktiky vývoje je, aby bylo zákazníkům pravidelně doručováno kvalitních produktů a služeb. Tento vývoj se odlišuje od jiných vývojů tím, že jednotlivé role například vývoj, provoz IT, kontrola kvality a bezpečí, nejsou od sebe izolovány a namísto toho mezi sebou komunikují. To má za výsledek kvalitnějšího a bezpečnějšího produktu. Výhodou je tedy, že týmy dosahují vyšších výkonů a lepších produktů za kratší čas. Což má za důsledek, že to zkrátí dobu uvedení produktu na trh, čímž je dosaženo vyšší spokojenosti zákazníka [9]. DevOps staví na praktikách, jako je průběžná integrace a průběžné nasazování, agilní vývoj a průběžné monitorování produktu [8].

DevSecOps staví na DevOps a integruje bezpečností postupy do každé fáze procesu DevOps, jako je plánování, budování, poskytování a provoz softwaru [8]. Takže myšlenka tohoto modelu je, aby se integrovaly bezpečnostní prvky již od brzké fáze vývoje. Vývojáři, kteří již úspěšně aplikovali myšlenku DevOps, by měli DevSecOps implementovat, jakožto přirozenou součástí vývoje, místo toho, aby to byl pouze oddělený nápad. Tento postup pomáhá minimalizovat zranitelnosti softwaru, které by v pozdější fázi provozu, mohly být nákladné na opravu. Výhodou této metody vývoje je rychlejšího dodání softwaru, zlepšení bezpečnosti a snížení nákladů [10].

Dalším aspektem je kryptografie, je málo tak důležitých věci pro bezpečnost aplikaci, tak jako je kryptografie. Když se použijí správné algoritmy a jejich správná implementace, i při úniku informací, mohou tato data být pro útočníky nečitelná [8]. V další části bakalářské práce, jsou popsány kryptografické algoritmy a jejich použití v určitých oblastech zabezpečení.

2 KRYPTORGRAFICKÉ ALGORTIMY

Kryptografie je metoda, která chrání citlivé informace a komunikaci po internetu. Zajišťuje tedy to, že informace bude moct být viditelná a zpracována jen těm uživatelům, pro které byla poslána. Tuto ochranu zajišťuje pomocí matematických konceptů a sady pravidel pro výpočet, jež se nazývá algoritmus. Ten posílanou informací zašifruje tak, aby byla nečitelná pro případné útočníky. Tyto algoritmy jsou využívané pro generování klíčů, digitální podpisy, bezpečné procházení webu, posílání e-mailů a důvěrné komunikace, jako je transakce kreditní karty [7].

Algoritmus zašifruje normální čitelnou zprávu pomocí matematických pravidel za pomocí unikátního klíče, výstupem je šifrovaný text, který může být uložen anebo poslán přes internet, ke kterému útočník nemá přístup. Důležitým faktorem je použití silného klíče s dostatečnou délkou, aby se případný útok nestal jednoduchým. Pro dešifrování šifrovaného textu se použije stejný klíč a algoritmus, ze kterého následně je získány čitelný text, který může být nadále zpracován.

Pro šifrovanou komunikaci na internetu se využívají krypto systémy, což jsou sady šifer, pro šifrování a dešifrování zpráv, pro bezpečnou komunikaci mezi počítači, zařízeními a aplikacemi. Tato sada se skládá z algoritmů se symetrickými anebo asymetrickými klíči, autentizaci zpráv, digitálního podpisu a výměny klíčů. Tento proces bezpečné komunikace, je začleněn do protokolů, které běží v síťových a operačních systémech [7].

V dnešní době je již několik bezpečných algoritmů a doporučených implementací, díky zkušeným expertům v oblasti kryptografie, kteří vyvíjeli algoritmy a jejich správné a bezpečné použití několik let. Proto je důležité se vyvarovat vlastním algoritmům, či používání vlastní implementace na již existující algoritmy a místo toho se držet doporučených postupů a využívání standardních knihoven, či velmi známých a ověřených knihoven třetích stran daného programovacího jazyka [8].

Moderní kryptografické algoritmy se rozdělují na základě počtu klíčů do dvou základních skupin. Symetrická kryptografie pracuje s jedním klíčem a asymetrická kryptografie s dvěma. Symetrická kryptografie má tu výhodu, že šifrování je velice rychlé, ale problém nastává ve chvíli sdílení klíče. Asymetrická kryptografie tento problém sdílení klíče řeší, jelikož pracuje s veřejným klíčem pro šifrování a privátním klíčem pro dešifrování, ale nevýhodou je, že je tento způsob pomalý a při velkém množství dat to není ideální řešení. Proto vznikla hybridní kryptografie, která je kombinací těchto dvou kryptografií. Velké množství

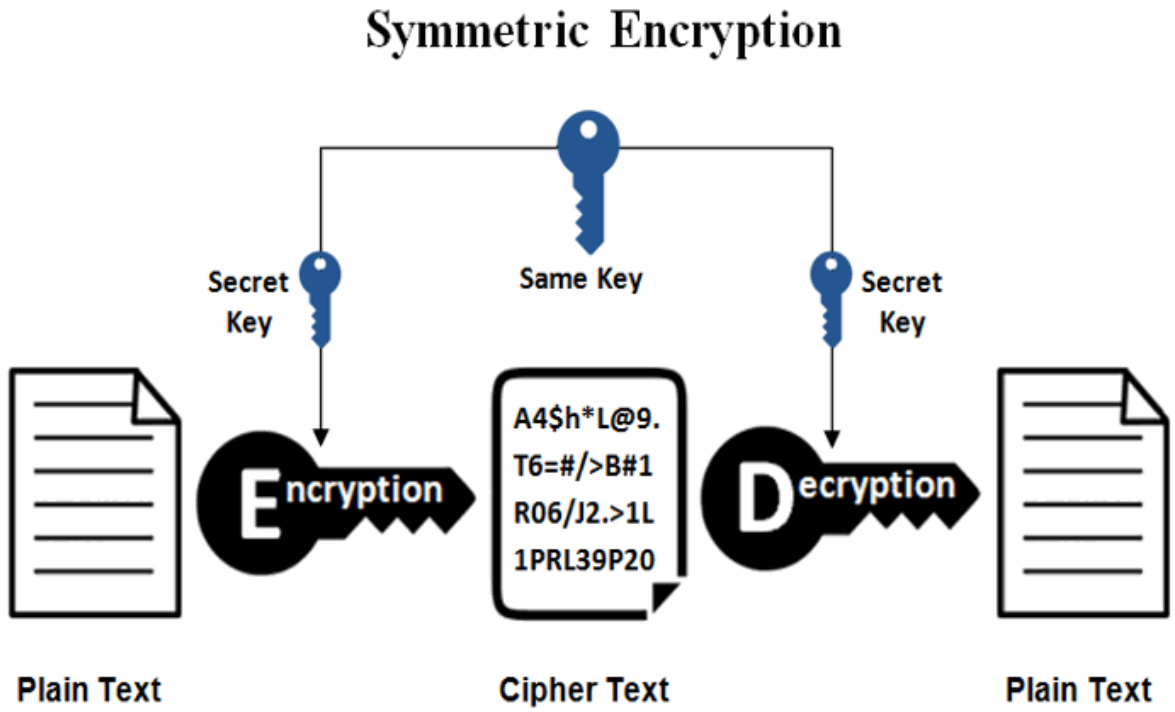
dat se zašifruje pomocí rychlé symetrické šifry a její klíč se zašifruje pomocí asymetrické šifry, následně jsou zašifrovaná data a klíč poslány příjemci.

Kryptografické klíče by měly být náhodné, či pseudonáhodné. K tomu slouží „Random Bit Generátor“ (RBG), který vygeneruje sekvenci náhodných bitů. RBG je potřebný k vygenerování klíčů a inicializačních vektorů, pro zajištění bezpečného šifrování dat. RBG se dále dělí na dvě odvětví, deterministický RBG a nedeterministický RBG. Deterministický RBG, někdy označován jako pseudonáhodný generátor, využívá inicializační hodnotu zvanou „seed“, která zajišťuje entropii neboli náhodnost [11]. Tento generátor nemůže být označován, jakožto úplně náhodný, jelikož za ním stojí nějaký algoritmus a vstupní hodnota. Proto je zde Nedeterministický RBG, protože je založený na nepředvídatelných fyzikálních jevech, které nejsou ovlivněny lidským faktorem, a tak je označovaný jako úplně náhodný generátor [11].

Je důležité, aby se klíče používali jen jednou na specifickou operaci, například šifrování, autentizace a zapouzdření klíče, jelikož použití stejného klíče na dvě kryptografické operace, může oslabit bezpečnost celého procesu, s tím se eliminuje možná škoda zapříčiněná prolomením jednoho klíče [11].

2.1 Symetrická kryptografie

Symetrická kryptografie pracuje s pouze jedním tajným klíčem, který je používán jak k šifrování, tak i k dešifrování elektronické informace. Entity, které šifrovaně komunikují se symetrickou šifrou, musí sdílet tento tajný klíč, aby se utajovaná informace mohla dešifrovat. Symetrické šifrování je sice starší metoda, avšak je mnohem rychlejší oproti asymetrickému šifrování, které má problémy s výkonem. Jelikož když šifruje velké množství dat, tak využívá velký výpočetní výkon procesoru. Díky této výhodě ve výkonu, se symetrické šifrování využívá pro šifrování velkého množství dat, například databází [12].



Obrázek 1 Znázornění symetrického šifrování [15]

Kromě potřebného sdílení stejného klíče s protější stranou, má symetrická kryptografie další nevýhody. Jednou z nich je, že při každém použití klíče odhalí informaci útočníkům, jak daná informace byla zašifrována. Obrana proti tomuhle spočívá v použití hierarchii klíčů, kde se nebudou opakovaně používat stejné klíče a namísto toho bude využívané střídání klíčů. Další nevýhoda oproti asymetrické kryptografii spočívá v metadatech, kterými nedisponuje. Metadata zaznamenávají informace, jako je datum vypršení platnosti klíče anebo označení, které indikuje, zda klíč slouží k šifrování anebo k dešifrování. Při menším počtu klíčů v provozu je tento problém možný řešit manuálně, avšak v případě reálného provozu, kde mohou být až tisíce klíčů, je potřeba využit zautomatizovaný manažer klíčů [12].

Symetrická kryptografie se nadále rozděluje na dvě odvětví, dle způsobu šifrování dat. Jedno z nich jsou blokové šifry, kde se otevřený text rozdělí na bloky o stejné velikosti a druhé jsou proudové šifry, kde se otevřený text šifruje po sekvenci bajtů [15].

2.1.1 Blokové šifry

Blokové šifry jsou navrženy tak, aby šifrovaly data o pevné délce. Velikost vstupního nešifrovanému bloku je obvykle roven tomu výstupnímu šifrovanému bloku, přičemž se může délka klíče lišit. Mezi moderní symetrické blokové šifry patří například AES, CAST, Twofish, IDEA, Serpent, RC5, RC6, Camellia a ARIA, tyto šifry jsou považovaný za

bezpečné [15]. Avšak dle organizace NIST jsou doporučené pouze AES a TDEA, přičemž TDES přestane být dne 31. prosince 2023 organizací uznávána jako bezpečná [16].

Blokovým šifrám, kterým by se mělo vyhnout kvůli jejich prolomení anebo spornému zabezpečení, jsou DES, RC2, Blowfish a GOST [15]. Nadále dle organizace NIST jsou za nebezpečné šifry označené 2TDES a SKIPJACK [16].

Většina populárních symetrických šifer jsou právě blokové, proto byly navržena různá schémata pro převedení blokových šifer do proudových, za účelem možnosti šifrování dat o libovolné délce. Tyto schémata jsou známá, jako režimy činnosti blokových šifer. Když je symetrická bloková šifra zkombinovaná s režimem činnosti, tak je šifra označená pomocí názvu šifry, velikosti použitého klíče a režimem činnosti, například AES-256-GCM, AES-128-CTR [15].

2.1.1.1 Režimy činností blokových šifer

Hlavní myšlenkou režimů činnosti blokových šifer je opakované přidávání bloků na vstup šifry, za účelem bezpečného šifrování většího množství dat, než je jeden blok o pevné délce. Některé režimy činnosti, jako je například CBC, potřebují do posledního rozděleného bloku, přidat takzvaný padding. Tento padding slouží k doplnění chybějících znaků do požadované délky bloku. Mezi padding algoritmy patří například PKCS7 a ANSI X.923. Avšak ostatní režimy činností, jako jsou CTR, CFB, OFB, CCM, EAX a GCM nepotřebují zmiňovaný padding, jelikož provádí XOR operace, mezi částmi otevřeného textu a vnitřním stavem šifry [15].

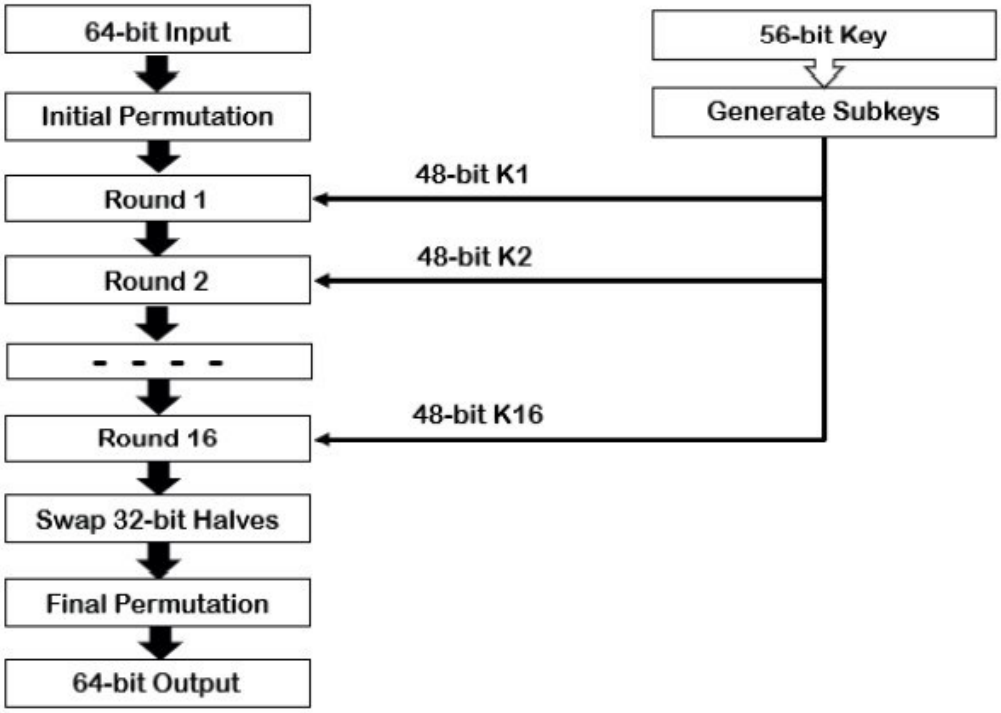
Mezi běžně používané režimy činnosti blokových šifer patří CBC, CTR a GCM. Tyto režimy vyžadují na začátku náhodný a nepředvídatelný inicializační vektor, označován zkratkou IV, taktéž známy jako „nonce“ či „salt“. Častou chybou bývá použití IV při šifrování více zpráv, tato chyba dává prostor pro kryptografické útoky. IV by měl mít stejnou velikost, jako je blok šifry [15].

CBC režim může být náchylný k takzvanému „padding oracle“ útoku, proto je vhodnější se mu vyhnout a namísto toho použít CTR, jelikož zde není potřeba použít padding. CTR je vhodnější ještě kvůli většímu zabezpečení a schopnostem paralelního zpracování, avšak neposkytuje autentizaci a integritu. Režim GCM má všechny výhody režimu CTR, a navíc je obohacen autentizaci. GCM je tak vyhovující pro rychlé a efektivní implementování autentizovaného šifrovaní. Proto se stává nejvíce doporučeným režimem činnosti. Výhodou těchto

režimu je podpora dešifrování náhodného přístupu, to se hodí při vyhledávání v libovolném časovém posunu ve videopřehrávači. Naopak režimu ECB je doporučené se vyhnout, protože je označený za nezabezpečený režim blokových šifer [15].

2.1.1.2 DES a TDES

Šifra Data Encryption Standard (DES), má velikost bloku 64 bitů a využívá délku klíče 56 bitů. Některé dokumentace uvádí délku 64 bitů, ale to je zapříčiněno tím, že každý osmý bit je paritní, sloužící k detekci chybného bitu v klíči. Šifra pracuje v šestnácti rundách, kde je v každé rundě použít jeden z šestnácti podklíčů. Tyto podklíče jsou vygenerovány z hlavního klíče [17].



Obrázek 2 Proces šifrování algoritmu DES [17]

Na výše uvedeném obrázku (Obr. 2) je znázorněný proces šifrování. Na vstup algoritmu je přiveden 64bitový blok dat, kde je provedená zahajovací permutace, která rozdělí blok na dvě 32bitové části. Obě části projdou šestnácti rundami, po kterých dojde k jejím sjednocení. Následně je provedena finální permutace s výsledkem 64bitovými zašifrovanými daty [17].

Avšak délka klíče je v dnešní době malá a má za důsledek potencionální útok hrubou silou, taktéž paritní bit nepřidává žádné vylepšení k procesu šifrování. Proto byl vyvinut lepší algoritmus TDES [17].

Triple Data Encryption Standard (TDES), také známy jako Triple Data Encryption Algorithm (TDEA) [16], je třikrát opakovaný DES algoritmus. Tím pádem se ztrojnásobila délka klíče na 168 bitů. Avšak kvůli trojnásobnému opakování se tento algoritmus stal pomalým [17].

TDES nabízí tři různé varianty použití klíčů, první je že se použijí tři různé klíče, což se docílí silného šifrování a odolnosti proti kryptoanalytickým útokům a útokům hrubou silou, avšak není tak silná, jelikož není odolná proti útoků zvaný „Meet-in-the-middle“. Tento útok však není považovaný za bezprostřední ohrožení, protože je potřeba velké množství času a informací o daném zašifrování. Druhá možnost je použití dvou různých klíčů a jednoho stejného a třetí varianta, že se použijí tři stejné klíče a je tak docílená zpětná kompatibilita se straším DES [18].

Kvůli ohrožení algoritmu DES útokem hrubou silou a následnému vývoji TDES, NIST rozhodlo, že je potřeba vyvinout nový standart šifrování. Cílem bylo vyvinout takový algoritmus, který by šifrování přenesl do nové generace, a tak byl odolný vůči útokům hrubou silou. NIST stanovila nároky na kandidáty, tak aby byl symetrický blokový algoritmus flexibilní, efektivní a ekonomicky výhodný. Dalšími požadavky byly podpora velikosti bloku 128 bitů a šířka klíče 128, 192 a 256 bitů. Z desítky kandidátu byl vybrán Rijndaelův algoritmus, který se stal novým americkým standardem pro šifrování dat. [17].

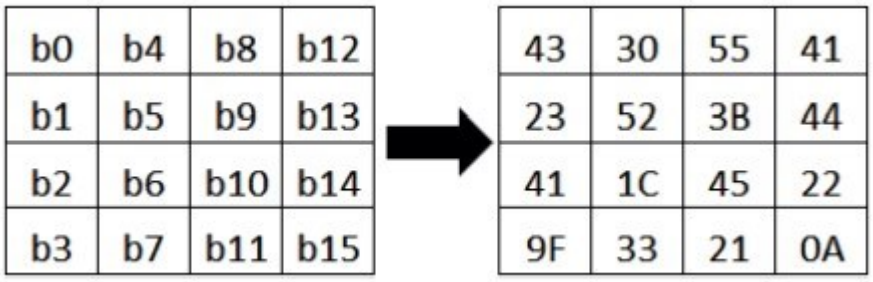
2.1.1.3 AES

Advanced Encryption Standard (AES), je znám také pod označením Rijndael. AES pracuje s blokem dat o velikosti 128 bitů a se symetrickým klíčem o velikostech 128, 160, 192, 224 anebo 256 bitů. Pro většinu aplikací je dostačující velikost klíče 128 bitů, ale je-li potřeba větší zabezpečení, tak se doporučuje použít velikost klíče 256 bitů. AES se taktéž využívá s kombinací režimem činností, pro šifrování většího množství dat, než je jeden blok. Většina vývojářů sahá po CTR, jelikož není potřeba řešit padding. Avšak s CTR je potřeba integrovat MAC, oproti GCM, kde je již integrován. AES je v dnešní době převážně používán v TLS a SSL, což je hlavní součástí protokolu HTTPS, které webové stránky a webové aplikace používají ke zabezpečené komunikaci se serverem. Kvůli popularitě v oblasti zabezpečené komunikace na internetu jsou v moderních procesorech implementované AES instrukce ke zvýšení rychlosti šifrování a dešifrování [15].

AES funguje na bázi čtyř transformací, které pracují s bajtama. Když data projdou všemi čtyřmi transformacemi, tak je to reprezentováno, jakožto jedna runda. Při délce klíče 128

bitů, algoritmus udělá 10 rund, u délky 191 bitů 12 rund a při poslední délce 256 bitů provede 14 rund. Podobně jak u algoritmu DES, tak i zde jsou z hlavního klíče vypočítaný podklíče, kde je každý jeden podklíč použít v jedné rundě [17].

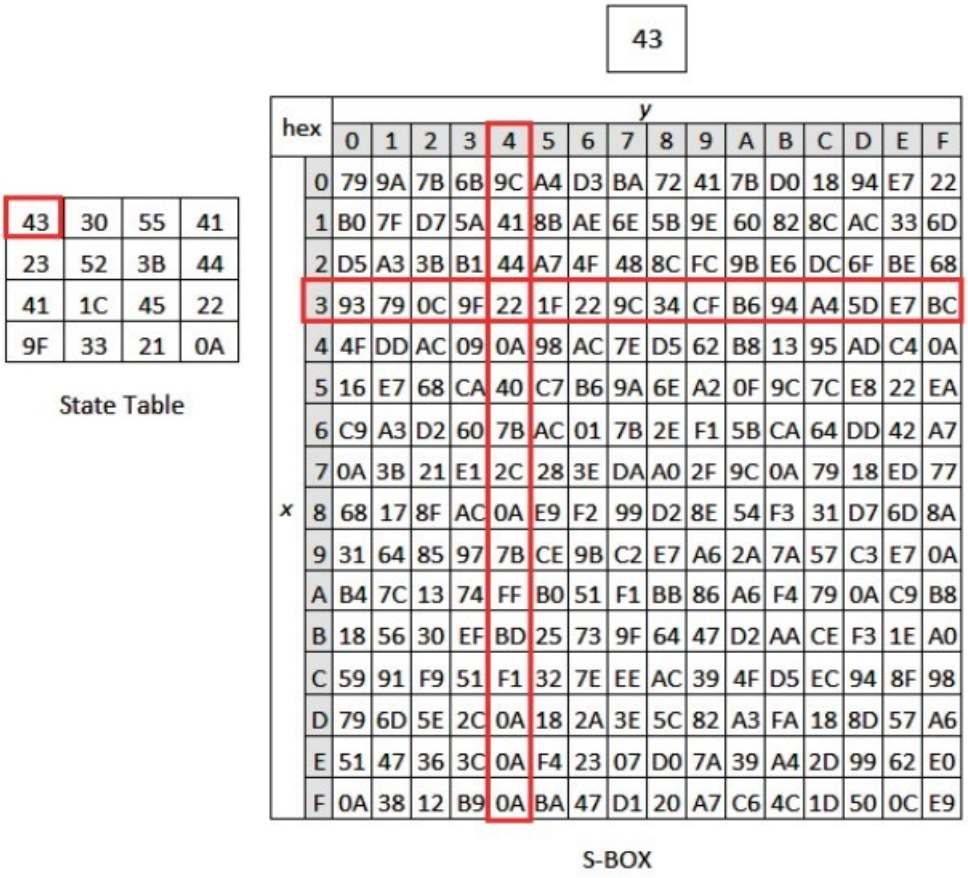
Algoritmus rozdělí 128bitový otevřený text do 16 bajtů, následné je každý bajt rozdělený do tabulky o velikosti 4x4. V obrázku (Obr. 3) lze vidět výsledná tabulka, kde je v každé buňce zapsaný jeden bajt v hexadecimální soustavě [17].



Obrázek 3 Vyplněna AES tabulka [17]

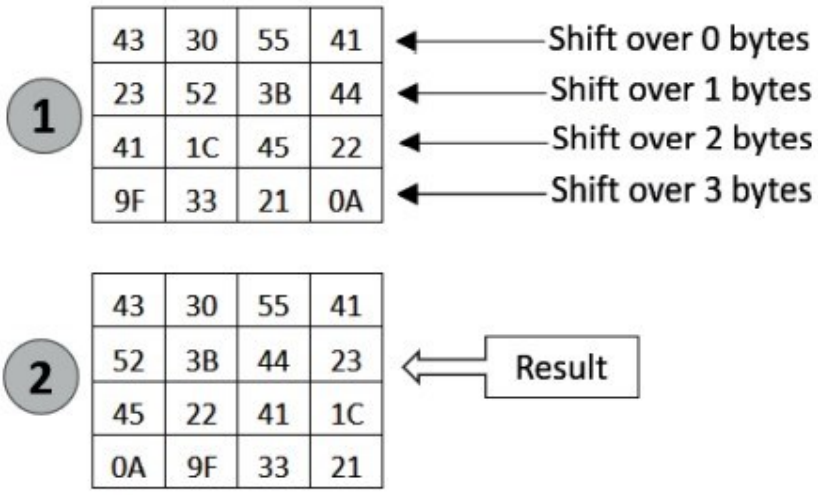
Každá runda se skládá ze čtyř transformací, a to Substitute bytes, Shift rows, Mix columns a Add round key. Ke každé rundě je přidělen jeden podklíč neboli Round key. Ze začátku je proveden XOR s round key a bajtovou tabulkou, což je následováno rundami [17].

V Substitu bytes, je použit k substituci takzvaný substituční box neboli S-box. Při používání S-boxu, je každý bajt substituovaný, dle jeho průsečíku souřadnic x a y. Souřadnice y je reprezentována pro sloupec S-boxu a je převzato z prvního hexadecimálního čísla bajtu a souřadnice x je pro řádek S-boxu a je převzata z druhého čísla bajtu [17]. Proces je vyobrazen v obrázku níže (Obr. 4).



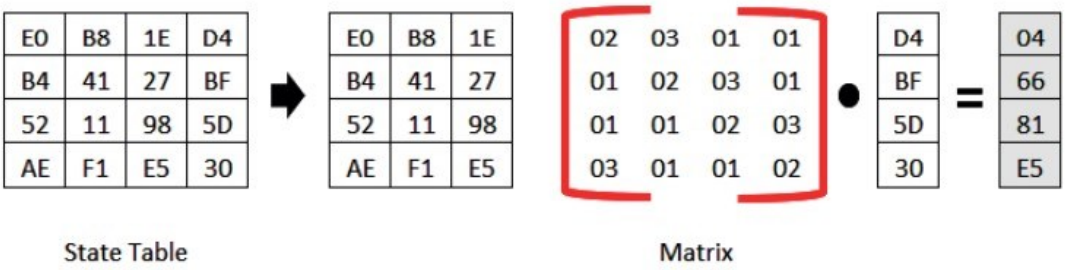
Obrázek 4 Substituce bajtů pomocí S-boxu [17]

Následná transformace je Shift rows, která provádí permutaci. Takže každý bajt v řádku tabulky je posunut doprava o číslo indexované od 0 až po 3. Z toho vyplívá, že první řádek není posunut [17]. Permutace je vyobrazena na obrázku (Obr. 5).



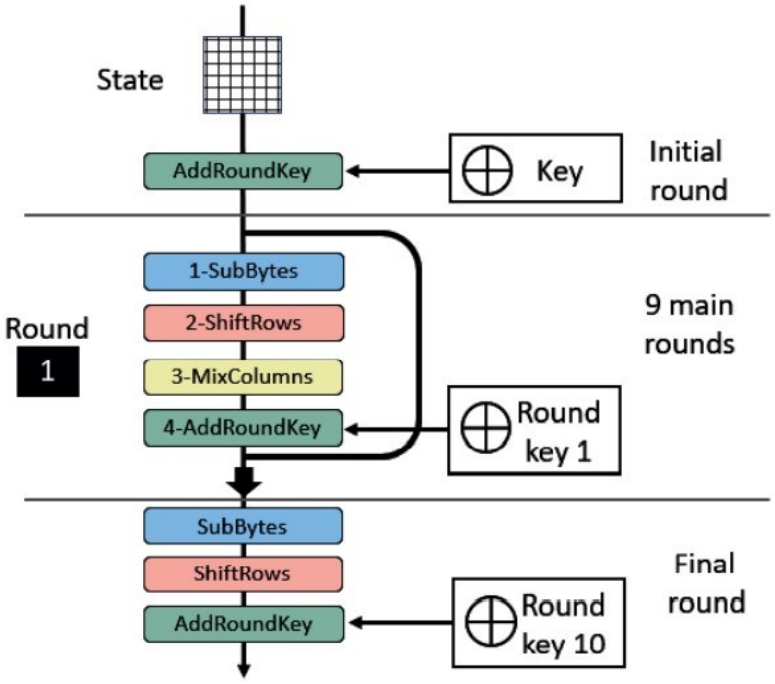
Obrázek 5 Vyobrazení Shift rows [17]

Poslední důležitá transformace je Mix columns, která také provádí permutaci. Tato transformace je zahájená tak, že se vezme pravý sloupec bajtové tabulky a ten je smíchán pomocí násobení s maticí, která má hodnoty 01, 02 a 03. Důvodem těchto menších čísel je, aby operace byla rychle dokončena, a tak poskytovala efektivní transformaci dat. Tento proces je proveden pro ostatní sloupce [17]. Vyobrazení této transformace je na obrázku (Obr. 6) níže.



Obrázek 6 Vyobrazení Mix columns [17]

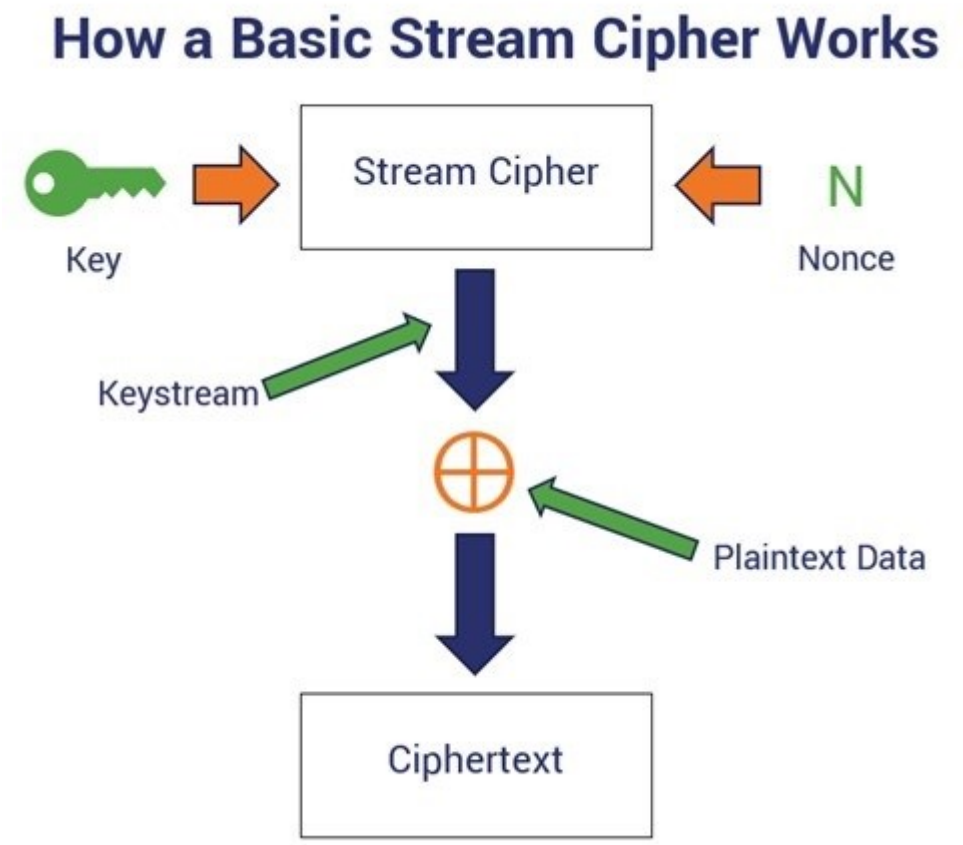
Na konci rundy je proveden XOR s bajtovou tabulkou a rundovým klíčem. Pokud už nenásleduje další runda, tak je výstupem šifrovaný text [17]. Celý proces šifrování je vidět na obrázku (Obr. 7) níže.



Obrázek 7 Šifrování algoritmu AES [17]

2.1.2 Proudové šifry

Proudové šifry šifrují data bit po bitu, namísto bloků, tak jak je to u blokových šifer. Hlavní části tohoto procesu je generování proudu pseudonáhodných bitů na základě šifrovacího klíče a seedu taktéž zvaný „nonce“. Tyto pseudonáhodné bity jsou zvané keystream, společně s otevřeným textem je proveden XOR a výsledkem této operace je již šifrovaný text. Proudové šifry se dělí na základě generování keystreamu a to na synchronní proudové šifry, kde keystream je generován nezávisle na otevřeném či předchozím šifrovaném textu a na samo-synchronní proudové šifry, kde se keystream generuje na základě předchozích bitů z šifrovaného textu [19].



Obrázek 8 Vyobrazení šifrování symetrické proudové šifry [19]

Symetrické proudové šifry jsou vhodnější pro zařízení s menším výkonem, jelikož šifrují data bit po bitu, místo celého bloku. Tento způsob šifrování je rychlejší a efektivnější, protože v případě chyby v jednom bitu není ovlivněný následující bit. Některé symetrické šifry jsou náchylné na útoky zvané „bit-flipping attack“ a „key reuse attack“. Proudové šifry nejsou tak dobře prozkoumaný oproti blokovým, které jsou vědci analyzované a testované, což má za důsledek menšího implementování. Avšak jsou proudové šifry využívaný v případě

šifrování dat při přenosu, například šifra ChaCha20 je využívaná v TLS 1.3 [19]. Další symetrické šifry jsou RC4, FISH, A5/1, A5/2, HELIX, CHAMELEON, OTP, Rabbit a Salsa20.

2.1.2.1 ChaCha

ChaCha patří do rodiny proudových šifer vytvořených Danielem J. Bernsteinem a je také vylepšená varianta šifry Salsa20. ChaCha pracuje v rundách, které může být 8, 12 a 20, kde ChaCha20 je rychlejší než šifra AES. Varianty ChaCha8 a ChaCha12 patří do skupiny nejrychlejších proudových o velikosti 256 bitů a jsou doporučeny aplikovat, pokud je rychlost důležitější než spolehlivost [20]. ChaCha20 pracuje s klíčem o délce 128 nebo 256 bitů a 64 bit dlouhým noncem se kterými zašifruje otevřený text. Výstupem šifry je stejně dlouhý zašifrovaný text [15].

Je důležité si uvědomit to, že symetrické šifry většinou v reálném provozu nejsou implementována samotná, ale jsou použitá v kombinaci s jiným kryptografickým algoritmem. Z toho vychází označení jako schéma symetrického šifrování či konstrukce symetrického šifrování. Ve většině šifrovacích schémat je využitá kombinace key derivation function (KDF) anebo s autentizaci zprávy [15].

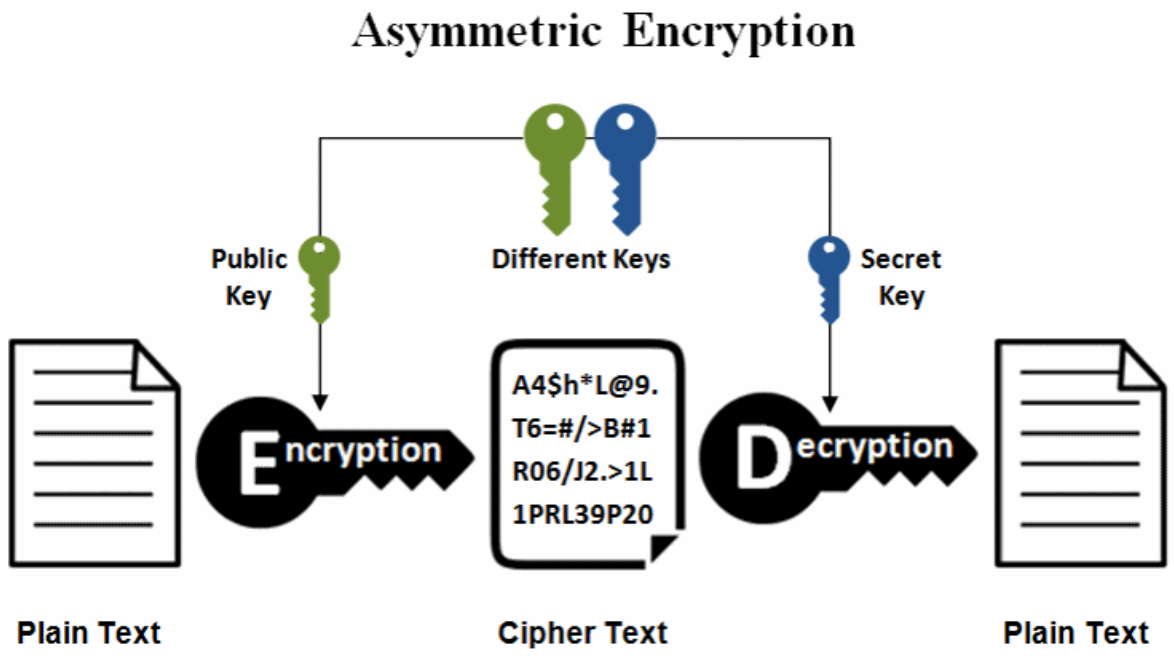
Jedním ze schémat je například autentizované šifrování s připojenými daty neboli AEAD. AEAD slouží k zajištění autentizaci a integrity dat a váže šifrovaná data se souvisejícími nešifrovanými daty společně s kontextem, kde se mají data objevit. Takže při pokusu vyjmout šifrovaná data a vložit je do jiného kontextu je zavčas detekováno. AEAD se používá tam kde jsou šifrovaná a nešifrovaná data společně posílána, například v síťových protokolech a zajišťuje, že celý datový tok je chráněn [15].

Mezi nejpopulárnějšími AEAD patří ChaCha20-Poly1305 a AES-256-GCM, které jsou implementovány ve většině moderních kryptografických knihovnách. Je doporučeno v dnešních aplikacích jejich implementaci použít, jelikož jsou velice bezpečná, osvědčená a dobře otestovaná [15].

2.2 Asymetrická kryptografie

Asymetrická kryptografie pracuje s dvěma klíči, které jsou matematicky svázaný. Pro šifrování je využíván takzvaný veřejný klíč, který může být přístupný pro všechny. K dešifrování se používá privátní klíč, který musí být držen v tajnosti. Výhodou je, že každý s veřejným klíčem, může zašifrovat informaci a případný majitel privátního klíče, tuto informaci

dešifrovat. Oproti symetrické šifře, kde je nutnosti, zaslat kopii tajného klíče, přičemž vzniká risk, že se ho může zmocnit útočník [13]. Nevýhodou je, že je oproti symetrické kryptografii pomalá, tak je neefektivní ji používat na velké množství dat. Mnoho protokolů, jako jsou například SSL a novější TLS, které jsou použitý v HTTPS, využívají asymetrickou kryptografii [14]. Asymetrické klíče uchovávají o sobě informace zvané metadata. Metadata obsahují záznamy jako jsou například typ klíče, jestli to je privátní či veřejný, datum vypršení platnosti, majitel daného klíče a k čemu se používá. Jak u symetrických šifer, tak i zde je vhodné použít zautomatizovaný manažer klíčů, který se stará o správu klíčů [13].



Obrázek 9 Znázornění asymetrického šifrování [15]

Asymetrická kryptografie je taktéž používaná k verifikaci autentičnosti zprávy, kde je zpráva nejdříve převedená pomocí hash funkce, do unikátního formátu zvaný otisk a poté zašifrována s použitím privátního klíče. Výsledkem toho procesu je elektronický podpis, který může být dešifrovaný veřejným klíčem, pro získání otisku a ověření autentičnosti zprávy, tak že příjemce zahashuje zprávu a srovná její otisk s otiskem získaným z elektronického podpisu [13]. Pro elektronický podpis jsou dle organizace NIST doporučený používat algoritmy ECDSA, EdDSA a RSA. Avšak samotný algoritmus DSA je nedoporučeno používat [16]. Asymetrická kryptografie se taktéž používá v protokolu pro výměnu klíčů, díky kterému lze bezpečně vyměnit symetrické klíče, mezi dvěma entitami [15]. Pro výměnu klíčů NIST doporučuje algoritmy Diffie−Hellman, MQV, Diffie-Hellman s využitím

eliptických křivek a RSA. Jelikož jde RSA použít jak na elektronický podpis a také na výměnu klíčů, je důležité v případě použití RSA na oba případy zároveň, nepoužít na tyto dvě operace stejný klíč [16].

Nejpopulárnější a nejbezpečnější asymetrické šifry jsou RSA a ECC. Mezi další populární asymetrické šifry se řadí ElGamal, DHKE, ECDH, DSA, ECDSA, EdDSA a Schnorrův podpis. Některé z těchto šifer provádí jednu či více funkcí, jako je generování veřejného a privátního klíče, šifrování a dešifrování, digitální podpis a algoritmus pro výměnu klíčů [15].

V případě, kdy uživatel vygeneruje ze svého privátního klíče veřejný klíč a ten digitálně podepíše, pro zajištění autentičnosti, který následně zašle protější straně pro zahájení komunikace. Zde nemá protější strana garanci, že tento veřejný klíč s digitálním podpisem, nevygeneroval útočník, ze svého privátního klíče. To lze vyřešit pomocí certifikátů, které v sobě obsahují metadata o asymetrických klíčích. Pro ověření těchto certifikátu se využívá Infrastruktura veřejných klíčů (PKI). PKI je centralizovaný systém, ve kterých lze důvěřovat certifikátům díky certifikačním autoritám (CA), které tyto certifikáty generují. Avšak tyto certifikáty jsou podepsané pomocí digitálního podpisu s takzvaným root certifikátem. Root certifikáty jsou „self-signed“, což znamená, že jejich důvěra pochází ze sebe samotných. Tyto root certifikáty jsou zahrnuty v operačních systémech a v bezpečném uložišti prohlížečů. Certifikáty se zejména využívají pro důvěrné a bezpečné navázaní komunikace v protokolu TLS [8]. Jeden z nejpoužívanějších certifikátu založený na PKI je standard X.509 [189].

2.2.1 RSA

RSA algoritmus je založený na matematických konceptech modulárního umocňování a faktorizace prvočísel. RSA nabízí generování klíčů velikostí 1024 až 4096 bitů, šifrování a dešifrování, digitální podpis a bezpečnou výměnu klíče. RSA může pracovat s klíči o velikosti 1024, 2048, 3072, 4096, 8129, 16384 bitů a výše. Za bezpečnou velikost klíčů se považují velikosti 3072 bitů a výše. Avšak vyšší velikosti klíčů, mají za důsledek spotřebování většího výpočetního času [15].

2.2.2 ECC

Eliptic curve cryptography (ECC) je považován za nástupce algoritmu RSA, protože ECC používá menší velikosti klíčů, velmi rychle generuje klíče, rychle se domlouvá na klíči a má rychlý digitální podpis. ECC je založen na algebraické struktuře eliptických křivek nad

konečnými tělesy a na problému diskrétního logaritmu eliptické křivky, který je považován za výpočetně neproveditelný pro velké klíče. ECC je nejvíce doporučená kryptografie, hlavně s moderními, velice optimalizovanými a zabezpečenými křivkami, jako jsou Curve2551, Curve448 a P-256. ECC nabízí algoritmy digitálního podpisu, jako jsou ECDSA (P256) a EdDSA (Ed25519, Ed448), šifrování a hybridní šifrovací schémata ECIES a EEECC a také algoritmy pro bezpečnou výměnu klíčů ECDH (X25519, X448) a FHMQV. Velikost klíče je ve většině knihoven 256 bitů, avšak můžou být i větší, kde velikost klíče ovlivňuje sílu zabezpečení a rychlost samotného algoritmu [15].

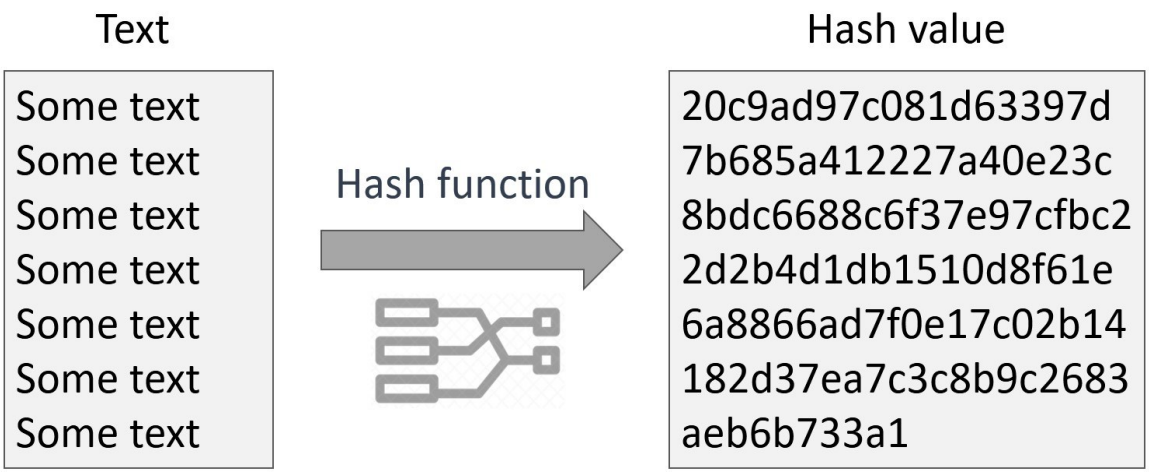
2.3 HASH, HMAC a KDF

Kryptografické hash funkce převádí vstupní data o libovolné velikosti do nečitelné hodnoty o pevné délce zvanou hash či otisk. Hash funkce jsou jedno cestné funkce, což znamená, že z jejich výsledku je velice obtížné získat vstupní hodnotu. Hash funkce jsou deterministické, což znamená že ze stejné vstupní hodnoty je vždy stejný hash. Jsou také velice rychlé a těžce analyzovatelné, jelikož jen při malé změně, například změnění jednoho písmena, vede k úplné změně výstupního hashe. U moderních hash algoritmů je prakticky nemožné získat z dvou různých vstupních dat stejnou hash hodnotu. Mezi nejbezpečnější hash algoritmy se řadí SHA-2, SHA-3 a BLAKE2 [15].

SHA-2 je rodina silných kryptografických hash funcí a je používána mezi vývojáři pro silné zabezpečení moderních komerčních aplikací. Hash může být ve třech různých velikostech a to SHA-256, SHA-384 a SHA-512, kde číslo v označení algoritmu značí velikost výstupního hashe v jednotkách bitů [15].

Rodina hash funkcí algoritmů SHA-3, známe tako pod označení Keccak, jsou mnohem bezpečnější, i při stejné velikosti hashe. Na rozdíl od SHA-2, jsou SHA-3 algoritmy odolné vůči útoků zvaný „length extension attack“. SHA-3 má několik variant a to SHA3-256, SHA3-384 a SHA3-512 [15].

Další bezpečné hash funkce jsou BLAKE2 a jeho vylepšené verze BLAKE2s a BLAKE2b a nadále RIPEMD-160. Je však doporučené vyhnout se algoritmům MD2, MD4, MD5, SHA-0, SHA-1, Panama, HAVAL a Tiger, jelikož jsou slabé anebo disponují bezpečnostní chybou [15].



Obrázek 10 Vyobrazení kryptografické hash funkce [15]

Message Authentication Code (MAC) slouží k zajištěni autentičnosti zprávy, tak jako u digitálního podpisu, ale s tím rozdílem, že se zde využívá symetrický klíč. Je přirovnávaný jako hash funkce s klíčem, protože malá změna ve vstupní hodnotě či v klíči vede k celkové změně na výstupu a taktéž je skoro nemožné zjistit původní vstupní hodnotu. Využití samotného hash algoritmu s klíčem, pro zaručení autentičnosti zprávy se považuje za nebezpečné. Proto je vhodnější využít HMAC anebo KMAC, což je MAC založený na algoritmu hash. Pro vypočítání MAC se využívají i symetrické šifry, tyto varianty jsou známe jako CMAC, GMAC anebo Poly1305. HMAC se používá pro ověření autentičnosti a integrity zprávy a také v některých případech pro KDF [15].

Key derivation function (KDF), se využívá pro ukládání hesel v databázi. Sice lze využít pro tyto účely hash, avšak toto rozhodnutí je velice nebezpečné a může vést k prolomení hesel. Pro více zabezpečené ukládání hesel je možné využít HMAC, společně s náhodnou hodnotou salt, avšak i tato možnost není natolik zabezpečená, jako speciálně vytvořené KDF algoritmy. Tyto algoritmy byly navrženy tak, aby se zpomalil proces prolamovaní hesel a to tak, že je použit salt s kombinací několika iterací. Pro moderní a zabezpečený KDF má útočník 5 až 10 pokusů za sekundu, oproti klasickým hash funkcím, kde má útočník tisíce až miliony pokusů za sekundu. Mezi KDF se řadí například PBKDF2, Bcrypt, Scrypt a Argon2 [15].

PBKDF2 je jednoduchá KDF, která je odolná proti útokům zvané “dictionary attack“ a „rainbow table attack“. Avšak tato KDF není odolná vůči útokům přes GPU, což je útok, kde se útočník snaží na grafické kartě paralelně prolomit heslo. Taktéž není odolná vůči ASIC útokům, kde ASIC je specializovaný hardware pro prolamování hesel [15].

Algoritmy Bcrypt, Scrypt a Argon2 jsou odolné vůči zmíněným útokům a jsou navrženy tak, že využívají salt a mnoho opakovaných iterací s kombinací velkého výpočetního výkonu CPU a velkého množství paměti RAM. Tato kombinace dělá velké komplikace pro vytvoření speciálního hardwaru a softwaru, pro jejich prolomení. Avšak Bcrypt využívá konstantní RAM paměti, což zjednodušuje útočníkům prolomení. Proto je více doporučeno využít Scrypt anebo Argon2 [15].

2.4 Post-kvantová kryptografie

V této práci byly popsány nynější bezpečné algoritmy, ale to se může v budoucnu změnit s nástupem kvantových počítačů, které mají potencionál prolomit například RSA, ECC a ostatní asymetrické algoritmy. Z tohoto důvodu, byly vyvinuty alternativní asymetrické algoritmy, zvané post-kvantové algoritmy. Tyto algoritmy se dělí na code-based, lattice-based, multivariate a hash-based. Kvantové počítače nejsou super počítače, které jsou schopny vyřešit to, co normální počítač nedokáže, například útok zvaný řešení hrubou silou anebo vyřešit NP-úplný problém. Normální počítače můžou mít stav 0 nebo 1 v jednom bitu, ale kvantový počítač může mít kvantový bit neboli „qubity“, který může být současně 0 a 1. Jeden z kvantových algoritmů, které můžou prolomit asymetrickou kryptografii je Shorův algoritmus, jelikož má exponenciální nárust v rychlosti řešení problému faktorizace prvočísel, diskrétního logaritmu a problém diskrétního logaritmu eliptické křivky. Na těchto problémech jsou stavěny asymetrické algoritmy, včetně RSA, ECC [21], ECDSA, ECDH a DHKE. Mezi kvantově bezpečné a jen lehce ovlivněné algoritmy se řadí hash algoritmy, jako jsou SHA2, SHA3 a BLAKE2. Také HMAC a CMAK společně s KDF, jako je Bcrypt, Scrypt a Argon2 nejsou natolik ovlivněny. Avšak pro ochranu před kvantovými počítači je třeba tyto algoritmy implementovat ve větší velikosti než 274 bitů [15].

Post-kvantové algoritmy ještě nejsou standardizované, avšak organizace NIST provádí standardizaci a vybírá tak vhodné kandidáty na doporučené post-kvantové algoritmy. Nyní se pro výběr uskutečnilo třetí kolo a finalisti tohoto kola jsou pro asymetrickou kryptografii Classic McEliece, CRYSTALS-KYBER, NTRU a SABER. Pro digitální podpis jsou to algoritmy CRYSTALS-DILITHIUM, FALCON a Rainbow. Mezi alternativní finalisty pro asymetrickou kryptografii se řadí algoritmy BIKE, FrodoKEM, HQC, NTRU Prime a SIKE. Pro alternativní finalisty digitálního podpisu to jsou algoritmy GeMSS, Picnic a SPHINCS+ [171].

3 VÝVOJOVÉ TECHNOLOGIE

Pro ulehčení vývoje webové aplikace, je nejlepší použít knihovny, souhrn doporučení anebo frameworky, protože je zbytečné vymýšlet a tvořit něco, co již bylo vytvořeno a otestováno. Při tvoření nového projektu, jsou již některé důležité komponenty, jako jsou například autentizace, správa hesel, „sessions“, platební brána a konfigurace databáze, hotové v knihovnách anebo ve frameworcích. Při používání knihoven anebo frameworků je jednoduché, v případě nějakého problému, vyhledat řešení na internetu. Díky frameworkům se taktéž zvyšuje zabezpečení, jelikož nabízí strukturu, která dodává zabezpečení vůči různým útokům, jako jsou například „Cross-Site Scripting“ a „SQL Injections“ [23]. V této kapitole budou popsány technologie, knihovny a frameworky v kontextu kryptografie.

3.1 JavaScript

JavaScript je programovací jazyk díky kterému je možno webovou stránku udělat interaktivní. Je možno jej použít jak na straně klienta (client-side), tak i na straně serveru (server-side). Vývojáři mohou použít JavaScriptové frameworky a knihovny, které obsahují již předepsané části kódů, pro rutinní funkcionalitu části webové aplikace. Mezi populární JavaScriptové front-end frameworky patří například React, Angular a Vue. Pro server-side část webové aplikace vývojáři využívají runtime prostředí Node.js [24]. JavaScript se řadí dle dotazníku již 9 let, mezi nejpoužívanější programovací jazyky na světě s 64.96 % odpovědí a jeho runtime prostředí Node.js se nachází na 6. místě s 33.91 % odpovědí [25].

3.1.1 Node.js

Node.js je open-source JavaScript runtime prostředí, které je postaveno na Google Chrome JavaScript V8 jádru. Využívá událostmi řízený a neblokující vstupně/výstupně model, který je díky tomu odlehčený, více efektivní a ideální pro náročné aplikace. Je důležité si uvědomit, že se nejedná o programovací jazyk a ani o framework, ale jak již bylo zmíněno, o runtime prostředí pro potřeby aplikace na straně serveru. Důležitým faktorem je, že je na Node.js postavena populární knihovna, pro stahování rozšiřujících balíčku zvaná NPM. NPM obsahuje miliony balíčku zdarma ke stažení, čímž se posiluje komunita okolo Node.js [26].

Node.js disponuje vestavěným stabilním kryptografickým modulem zvaný Crypto. Crypto modul je obalená knihovna OpenSSL, která disponuje kryptografickými operacemi, jako je šifrování, dešifrování, hash, elektronický podpis a HMAC [27]. Také má druhý

kryptografický modul zvaný Web Crypto API, avšak to je v experimentálním módu. Tento modul provádí implementaci standardizované knihovny Web Crypto API [27].

Modul Crypto je určen pro potřeby převzetí některých kryptografických operací anebo pro vlastní implementování protokolu TLS, ale to není doporučeno, jelikož je potřeba velká zkušenost v oboru kryptografie. Proto se doporučuje na místo toho využít moduly zvané TLS anebo HTTPS [43].

Modul TLS nabízí implementací protokolů TLS a SSL, které jsou převzatý z knihovny OpenSSL [44]. V tomto modulu jsou již implementované kryptografické operace potřebné pro šifrovací vrstvu. Avšak neobsahuje komunikační protokol HTTP. Modul HTTPS disponuje oběma funkcemi, jelikož rozšiřuje modul TLS o komunikaci přes HTTP [45].

Pro autorizaci uživatelů anebo pro zabezpečenou výměnu informací lze využít JSON Web Token (JWT). Informacím může být důvěřováno, jelikož JWT je digitálně podepsán pomocí algoritmů HMAC, RSA anebo ECDSA. JWT se skládá ze tří části, header, payload a signature. V headru se nachází informace o typu tokenu a o použitém algoritmu pro digitální podpis. V části payload se již nachází informace o daném uživateli. A v poslední části signature je již samotný podpis [48]. Pro Node.js existují tři JWT moduly a to Node-jsonwebtoken, Jose a Aws-jwt-verify.

3.1.2 Express.js

Mezi nejpoužívanější JavaScriptový back-end framework se řadí Express.js [25] Express.js je open-source framework pro Node.js a je určen pro rychlý vývoj webových aplikací. Express.js pomáhá organizovat strukturu webové aplikace na straně serveru do návrhového vzoru MVC. Express.js poskytuje jednoduché směrování požadavků ze strany klienta. A také middleware, který je zodpovědný za rozhodování o poskytnutí správných odpovědí na požadavky klienta. [28]. Express.js nedisponuje vlastní kryptografickým modulem, ale jelikož je stavěný na Node.js, tak může využít jeho moduly, nebo moduly třetích stran.

3.1.3 Nest.js

Nest.js patří taktéž mezi oblíbené frameworky pro programování server-side webových aplikací. Také staví na runtime prostředí Node.js a je velice škálovatelný. Nabízí objektové orientované programování, funkcionální programování a funkční reaktivní programování v jazyce TypeScript, anebo v JavaScriptu. Také nabízí robustní používání HTTP jako u

frameworku Express.js [69]. Tím že je stavěný na Node.js tak lze využít stejné kryptografické moduly, které jsou vestavěné v Node.js anebo importované ze třetích stran.

3.2 Python

Mezi nejpopulárnější programovací jazyky v dotazníku od Stackoverflow se Python řadí hned za JavaScriptem s počtem 48.24 % odpovědí [25]. Python se hodí pro programování back-end části webové aplikace, jelikož nabízí všestrannost. Což znamená, že Python může být použít například pro strojové učení, data science, zpracování obrazů a mnoho dalších odvětví. Další výhodou je lepší čitelnost kódu, díky zalamování pomocí mezery neboli bílého znaku. Python nabízí objektově orientované programování, díky čemuž lze kód rozdělit na jednotlivé části. Python taktéž nabízí mnoho frameworků, mezi nejpopulárnější se řadí Django a Flask [73].

Pro kryptografické účely se mezi nejlepší knihovny řadí PyCryptodome, Cryptography a PyNaCl [102]. Avšak tyto knihovny jsou třetích stran, Python také disponuje svými knihovnami. Jednou z nich je Hashlib, která slouží pro implementaci hash, HMAC a KDF algoritmů [119]. Druha vestavěná knihovna je Ssl pro implementaci protokolu TLS verze 1.3, která také využívá knihovnu OpenSSL [120]. Pro potřeby JWT lze využít knihoven třetích stran a to PyJWT, Python-jose a Authlib.

PyCryptodome nahrazuje starou PyCrypto knihovnu, proto je potřeba mít nainstalovanou jen jednu z nich, jinak může dojít ke kolizi, jelikož se jejich metody importují ze stejného balíčku s názvem „Crypto“. Tato instalace PyCryptodome se doporučuje pouze pokud se aplikace spouští ve virtuálním prostředí. Pro předejití kolizí se doporučuje naistalovat knihovnu přes název PyCryptodomex, kde se knihovna v kódu importuje z balíčku Cryptodome. Tato knihovna je implementována v čistém Pythonu, kromě některých části, které jsou implementovány v jazyce C, protože jsou kriticky důležité pro výkon, jako jsou například blokové šifry. Tato knihovna je určena pro nízkoúrovňovou kryptografii [103].

Knihovna Cryptography obaluje OpenSSL a rozděluje se na dvě úrovně. Jedna pro bezpečnou implementaci kryptografie, kde se vývojáři nemusí rozhodovat o výběru konfigurací zvaná vysokoúrovňová kryptografie. A druhá nízkoúrovňová kryptografie, kde je potřeba rozhodování o správné konfiguraci kryptografických algoritmu, kde je nutností znát kryptografické principy. Jelikož je tato úroveň hazardní, protože při špatné konfiguraci může dojít k ohrožení bezpečnosti, tak se jejich metody importují přes „hazmat“ [104].

PyNaCl staví na knihovně libsodium, která implementuje silnou knihovnu s názvem Networking and Cryptography library (NaCl). Nabízí výrazné zlepšení v použitelnosti, zabezpečení a rychlosti. PyNacl nabízí širokou škálu možností pro zabezpečenou komunikaci a ochranu dat [102]. Jedná se o vysokoúrovňovou kryptografickou knihovnu.

3.2.1 Flask

V dotazníku od stránky Stackoverflow se framework Flask řadí mezi nejpopulárnější frameworky jazyka Python s počtem 16.14 % odpovědí [25].

Flask je mikro-framework, což znamená, že je to normální framework, který nemá žádné anebo jen málo závislostí na externích knihovnách. Má to několik výhod, jednou z nich je, že framework je odlehčený, tak není třeba aktualizovat a hlídat výskyt bezpečnostních chyb v závislých knihovnách. Ale kvůli tomu je potřeba psát více kódu anebo navýšit list závislostí přidáním nějaké knihovny či pluginu [74].

Flask nedisponuje vlastní kryptografickou knihovnou, ale může využít ty, které jsou implementované v jazyce Python anebo importovat knihovnu třetí strany.

3.2.2 Django

Django se umístil na druhem místě mezi frameworky jazyka Python, s počtem odpovědí 14.99 % [25]. Django umožňuje rychlý vývoj bezpečných webových aplikací, jelikož se postará o většinu problému spojených s vývojem, a tak se vývojáři mohou soustředit na vlastní psaní kódu. Lze ho použít na jakýkoliv typ webové stránky, od systému pro správu obsahu až po sociální sítě a zpravodajské webové stránky. Django pomáhá vývojářům se vyhnout běžným bezpečnostním chybám, a to například bezpečného způsobu správy uživatelských účtu a hesel. Vyhýbá se běžným chybám, jako je vkládání informací o relaci do souborů cookies anebo nesprávného ukládání hesel v čitelném formátu bez použití KDF. Taktéž Django ve výchozím nastavení umožňuje ochranu proti mnoha zranitelnostem, jako je například „SQL injection“, „cross-site scripting“ a „clickjacking“. Pomocí návrhových principů a vzorů napomáhá psát udržovatelný a znovu použitelný kód [75]. Django pouze disponuje knihovnou Signer, která slouží pro hashování dat [136]. Pro ostatní kryptografické operace lze použít knihovny Pythonu anebo importovat knihovnu ze třetí strany.

3.3 Jazyk C#

Další populární programovací jazyk je C#, který má v dotazníku, zhotoveny webovou stránkou Stackoverflow, počet odpovědí 27.86 % [25]. C# je moderní objektově orientovaný programovací jazyk, jež napomáhá vytvořit robustní a bezpečné aplikace. C# běží v .NET, který nabízí runtime prostředí CLR a soubor knihoven. Kořeny tohoto jazyku sahají do populárního jazyku C, a tak pro vývojáře, kteří již programovali v jazyku C, C++, Java anebo JavaScript, bude jeho syntaxe povědomá. C# nabízí několik funkcí, které napomáhají ve vývoji robustní a odolné aplikace, jako je například automatické uvolňování paměti v případě její alokace. Nabízí také ošetření výjimek a jejích zpracování, asynchronní operace a mnoho dalšího [139].

.NET nabízí kryptografickou knihovnu zvanou Cryptography, ve které je mnoho standardizovaných algoritmů pro jednotlivé kryptografické operace jako je digitální podpis, výměna klíčů, hash, AEAD a KDF [141]. Avšak tato knihovna využívá kryptografických algoritmů, které podporuje daný operační systém, takže v případě použití například na Linuxu, je potřeba nainstalovat knihovnu OpenSSL [142]. Pro implementaci JWT lze využít knihoven třetích stran zvané Jwt a Jose-jwt.

3.3.1 ASP.NET Core

Mezi webovými frameworky v dotazníku stránky Stackoverflow se ASP.NET Core umístil na 6. místě s počtem odpovědí 18.10 %. Mezi frameworky zaměřený na back-end je na 2. místě [25]. ASP.NET Core je mezi platformní, open-source framework pro vytváření moderních, rychlých aplikací s připojením na internet. Disponuje návrhovým vzorem MVC a programovacím modelem „Razor Pages“, který napomáhá ve vytváření uživatelského prostředí. Pro renderování HTML kódu na straně serveru disponuje technologii „Tag Helpers“ a pro validaci jak na straně serveru a klienta modelem „Model validation“. ASP.NET Core bez problému integruje s populárními client-side frameworky a knihovnami, jako jsou například Angular, React a Bootstrap [140].

Pro využití kryptografických operací v tomto frameworku, se používá knihovna Cryptography. Pro šifrovaný přenos pomocí HTTPS využívajícího TLS, není potřeba konfigurovat nějakou knihovnu, jelikož ASP.NET CORE používá protokol HTTPS ve výchozím nastavení. [143]

3.4 Ostatní knihovny

3.4.1 OpenSSL

OpenSSL je open-source knihovna, napsaná v jazyce C. Nabízí robustní kryptografické nástroje a nástroje pro zabezpečenou komunikaci [172]. Jedná se o knihovnu, která je schválena ve standardu FIPS od společnosti NIST [42]. Je využívaná některými kryptografickými knihovnami, které jsou porovnávané v praktické části, a proto tato knihovna bude srovnána.

3.4.2 Open Quantum Safe

Open Quantum Safe (OQS), je open-source projekt pro implementování post-kvantových algoritmů. Tento projekt je napsaná v jazyce C a dělí se na dvě části. Jedna část je knihovna pro post-kvantové algoritmy zvaná Liboqs a druhá část je integrování těchto algoritmů do protokolů a aplikací, včetně knihovny OpenSSL [180]. OQS nabízí implementací Liboqs do různých programovacích jazyků, jako je C++, Go, Java, C#, Python a Rust [181].

3.5 Kryptografické potřeby pro vývoj webových aplikací

Pro vývoj webových aplikací máme následující kryptografické potřeby: digitální podpis, zabezpečenou komunikaci, správné ukládání hesel, integrita a autentičnost dat, generování klíčů a inicializačních vektorů, výměna klíčů, autorizaci uživatelů, certifikáty a protokoly.

3.6 Kritéria pro porovnání vývojových technologií

V praktické části budou tyto technologie srovnány na základě několika kritérií. Jedním základním faktorem je popularita, jelikož lze lehce dohledat na diskusních fórech řešení při nějakém problému a také se jeví, že daná technologie bude pravidelně aktualizovaná, což je úzce spojeno s bezpečností. Populární technologie má větší pravděpodobnost rychlejšího opravení chyb, což je dalším bodem kritérii pro porovnání. Pro porovnávání bude použito CVE, kde se bude popisovat v jaké verzi frameworku či runtime prostředí, se tato chyba nacházela a v jaké rychlosti byla vydaná aktualizace pro opravu. Pro rychlost opravy chyb je také důležitý faktor, zda je technologie označená open-source licencí, jelikož je kód dostupný na internetu, a tak mohou uživatelé ale také etičtí hackeři najít kritickou chybu a nahlásit jí [22]. Také bude srovnáváno, jakým návrhovým vzorem framework disponuje. Důležitým srovnávacím bodem bude, jakými kryptografickými knihovnami disponují a zda je nutností pro jednotlivé kryptografické operace importovat knihovnu třetí strany. Bude

srovnáno, zda disponují zmíněnými bezpečnými algoritmy, které jsou popsány v kapitole 2. Pro vývojáře je také důležitým aspektem dokumentace, kde se nachází podrobné vysvětlení jednotlivých komponentů s případným ukázkovým kódem.

II. PRAKTICKÁ ČÁST

4 ÚVOD DO PRAKTICKÉ ČÁSTI

V této kapitole budou srovnávány jednotlivé technologie pro vývoj bezpečné webové aplikace, v kontextu kryptografie. Porovnávaní bude zhotoveno pomocí tabulek, ve kterých se budou porovnávat informace na základě zmíněných kritérií, které jsou stanoveny na konci teoretické části. Technologie jsou rozdělené do kategorií dle programovacích jazyků, runtime prostředí či knihoven. V poslední kapitole praktické části je zhotoveno zhodnocení tohoto porovnání, kde se nachází doporučení pro výběr těchto technologií.

5 POROVNÁNÍ VÝVOJOVÝCH TECHNOLOGIÍ

5.1 JavaScript (Node.js)

Tabulka 1 Obecné informace Node.js

Runtime prostředí	**Node.js**
Popularita na StackOverflow	33.91 %
Licence	MIT
Vydání	27.05.2009
Nejnovější LTS verze	16.14.2
Vydání LTS verze	17.03.2022
Nejnovější verze	18.0.0
Vydání poslední aktualizace	19.04.2022
Hodnocení na GitHub	87 087
Počet otevřených dotazů na GitHub	1 248
Počet uzavřených dotazů na GitHub	13 211
Počet commitů na GitHub	36 131

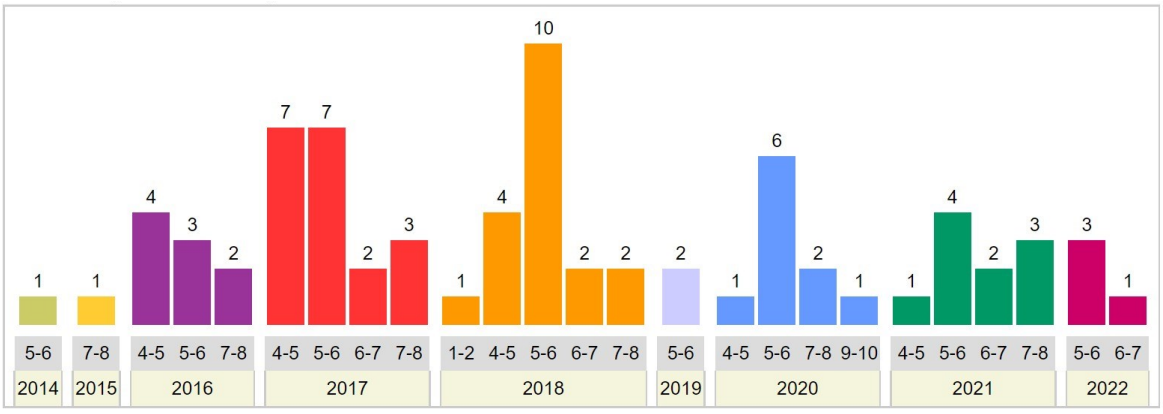
Výše uvedená tabulka (Tab. 1) vychází z následujících zdrojů [25][29][30][32]. Z této tabulky si lze všimnout, že je toto runtime prostředí velice populární, což je velký bonus, jelikož se zdá být ještě dlouhodobě podporován. Tato popularita vychází z dotazníku, který uskutečnila webová stránka StackOverflow, kde na tento dotazník zodpovědělo v květnu roku 2021 přes 80 000 lidí [25]. Na trhu je již několik let, z čehož se jeví, že zpracovaní je kvalitní a bezpečné. Na bezpečnost má podíl i open-source licence, z čehož se dá uvodit, že daná technologie je přístupná pro mnoho lidí, kteří dokážou příslušnou chybu odhalit. Node.js disponuje programem zvaný „Node.js bug bounty program“, kde na webové stránce zvanou HackerOne, lze nahlásit odhalenou chybu, která bude potvrzená do pěti dnů a do deseti dnu na ní bude odpovězeno s uvedením s instrukcemi pro zpracování daného příspěvku. Co se týče modulů třetích stran, tak je uvedeno, že by se měly nahlásit autorům těchto modulů [38]. Komunita je také přívětivá, jelikož na stránce GitHub, je velké množství

jak uzavřených, tak i otevřených dotazů v případě nějakého problému v kódu. Také hodnocení od jednotlivých uživatelů na GitHubu je vysoké, což potvrzuje fakt popularity.

5.1.1 Bezpečnost

Tabulka 2 Srovnání počtu CVSS pro Node.js v jednotlivých letech

CVSS / ROK	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10
2014	-	-	-	-	1	-	-	-	-
2015	-	-	-	-	-	-	1	-	-
2016	-	-	-	4	3	-	2	-	-
2017	-	-	-	4	4	2	3	-	-
2018	1	-	-	4	10	2	2	-	-
2019	-	-	-	-	2	-	-	-	-
2020	-	-	-	1	6	-	2	-	1
2021	-	-	-	1	4	2	3	-	-
2022	-	-	-	-	3	1	-	-	-



Obrázek 11 Počet CVSS pro Node.js dle jednotlivých let [33]

Výše uvedená tabulka (Tab. 2) vychází z následujícího zdroje [33]. Ze zjištěných dat lze usoudit, že Node.js měl v minulosti pouze jednu kritickou chybu a několik vyšších chyb. Tato kritická chyba byla v modulu API, kde některé funkce byly náchylné k přetečení

bufferu [68]. Ostatní nabyly střední hodnocení CVSS a jedna malého hodnocení. Z čehož se jeví, že daná technologie je bezpečná.

Tabulka 3 Srovnání CVE pro Node.js z roku 2022

CVE ID	**Datum**	**CVSS**	**Verze**	**Datum aktualizace**
CVE-2022-21824	20.12.2021	6.4	>= 17.3.1	10.01.2022
CVE-2021-44533	17.12.2021	5.0	>= 17.3.1	10.01.2022
CVE-2021-44532	17.12.2021	5.0	>= 17.3.1	10.01.2022
CVE-2021-44531	17.12.2021	5.8	>= 17.3.1	10.01.2022

Výše uvedená tabulka (Tab. 3) vychází z následujících zdrojů [34][35][36][37]. Z této tabulky lze vidět, že vývojáři vydali aktualizaci na odhalené chyby za necelý měsíc. Z čehož se jeví, že Node.js není zanedbaný a stále se na něm pracuje. Je to důležitý aspekt, protože uživatelé se nemusí obávat, že by jejich aplikace byla v budoucnu nezabezpečená.

5.1.2 Dokumentace Node.js

Dokumentace Node.js se nezdá pro nové vývojáře příliš přívětivá. V sekci „Usage and example“ je pouze ukázáno, jak založit nový projekt s jedním ukázkovým kódem pro „hello world“. Nadále není popsáno fungování celého runtime prostředí v samotné dokumentaci, ale na hlavní stránce v sekci „About“. Dokumentace je strukturovaná do jednotlivých modulů, kde je popsáno k čemu jsou určeny. Tyto moduly obsahují u jednotlivých metod ukázkové kódy. Popis jednotlivých parametrů metod se zdá být málo popsaný. Je zmíněno, jaké moduly jsou stabilní anebo experimentální, což dává přehled k vybírání příslušných modulů. Pro nové vývojáře se tato dokumentace může jevit jako nepřehledná a pro implementaci nějakého modulu musí vyhledat návod z jiného zdroje [129].

5.1.3 Kryptografické Node.js moduly

Tabulka 4 Porovnání kryptografických Node.js modulů

Modul	Součást Node.js	Stabilita	Implementuje knihovnu	Licence knihovny	FIPS schválení knihovny
Crypto	Ano	Stabilní	OpenSSL	Apache	Ano
Web Crypto API	Ano	Experimentální	Web Crypto API	MIT	Ne
TLS	Ano	Stabilní	OpenSSL	Apache	Ano
HTTPS	Ano	Stabilní	-	-	-

Výše uvedená tabulka (Tab. 4) vychází z následujících zdrojů [39][40][41][42][44][46]. Z tabulky si lze všimnout, že jsou tyto moduly vestavěné v Node.js a tak není třeba instalace, při které by mohla nastat špatná konfigurace, což by vedlo k ohrožení bezpečnosti. Další výhodou je kontrolovatelnost, jelikož je větší záruka správné implementace jejich algoritmů, samotnými vývojáři. Moduly Crypto a TLS obalují, knihovnu OpenSSL, která je schválená ve standardu FIPS 140-2, jenž provedla společnost NIST [42]. Z čehož lze vyvodit, že se jedná o bezpečnou implementaci, jejich algoritmů a kryptografických operací. Modul Web Crypto API implementuje knihovnu Web Crypto API. Tento modul je experimentální, proto se nejeví bezpečně. Moduly TLS a HTTPS, které slouží pro implementaci bezpečné komunikace, jsou taktéž stabilní. Stabilitou disponuje taktéž kryptografický modul Crypto. Z těchto poznatků, se tyto tři moduly jeví bezpečně.

Tabulka 5 Porovnání kryptografických operací Node.js modulů

	Crypto	**Web Crypto API**
RBG	Ano	Ano
AEAD	Ano	Ano
Digitální podpis	Ano	Ano
Protokol pro výměnu klíčů	Ano	Ano
Hash	Ano	Ano

HMAC	Ano	Ano
KDF	Ano	Ano
X.509	Ano	Ne
Post-kvantová kryptografie	Ne	Ne

Výše uvedená tabulka (Tab. 5) vychází z následujících zdrojů [27][39]. Z tabulky lze usoudit, že oba vestavěné moduly obsahují všechny potřebné algoritmy, pro základní kryptografické operace, avšak modul Web Crypto API nedisponuje pouze certifikátem X.509. Proto není potřeba pro potřeby kryptografie v dnešní době importovat knihovnu třetí strany, jelikož Crypto modul je pro tyto potřeby dostačující. Avšak pro budoucnost, kdy mohou přijít na trh kvantové počítače, nedisponují tyto moduly potřebnými post-kvantovými algoritmy. Populární a ani pravidelně aktualizovaný post-kvantový modul nebyl nalezen, jak pro čistý JavaScript, tak ani pro Node.js. Ale jelikož se jedná o nejpopulárnější jazyk a velice populární runtime prostředí, tak s nástupem kvantových počítačů bude velice rychle vytvořena knihovna pro potřeby implementace post-kvantových algoritmů. Z těchto důvodu budou popsány pouze tyto dva vybrané moduly, pro základní kryptografické operace.

Tabulka 6 Porovnání doporučených algoritmů v modulu Crypto

	Algoritmy
AEAD	Chacha20-poly1305, AES-256-GCM
Digitální podpis	ECDSA (P256), RSA, EdDSA (ed25519, ed448)
Protokol pro výměnu klíčů	DH, ECDH (x25519, x448)
Hash	SHA-2 (256,384,512), SHA-3 (256,384,512), BLAKE2b512, BLAKE2s256, RIPEMD-160
KDF	Scrypt

Výše uvedená tabulka (Tab. 6) vychází z následujícího zdroje [27]. Z tabulky lze vypozorovat, že tento modul disponuje s některými doporučenými algoritmy, pro jednotlivé kryptografické operace. HMAC v této tabulce nebyl porovnaný, protože využívá algoritmů hash. Z těchto výsledků se tento modul jeví jako bezpečný pro kryptografické účely. Avšak tento

modul podporuje i zastaralé a nebezpečné algoritmy a také umožňuje použití slabých délek klíčů. Proto je třeba znát bezpečnostní doporučení použití algoritmů a jejich konfigurací [27].

Tabulka 7 Porovnání doporučených algoritmů v modulu Web Crypto API

	Algoritmy
AEAD	AES-256-GCM
Digitální podpis	ECDSA (P256), RSA, EdDSA (node.ed25519, node.ed448)
Protokol pro výměnu klíčů	node.DH, ECDH (node.x25519, node.x448)
Hash	SHA-2 (256,384,512),
KDF	node.Scrypt

Výše uvedená tabulka (Tab. 7) vychází z následujícího zdroje [39]. Z tabulky si lze všimnout, že v modulu Web Crypto API, je méně doporučených algoritmů v kategorii AEAD a hash, oproti modulu Crypto. To je z toho důvodu, že chybějící algoritmy nejsou obsaženy v původní knihovně Web Crypto API [40]. Také jsou zde některé algoritmy s prefixem „node.“. Tyto algoritmy nejsou obsaženy v převzaté knihovně, a proto jsou přidané samotným Node.js. U těchto algoritmů je potřeba si dávat pozor, protože nejsou podporované ostatními WebCrypto implementacemi, což redukuje přenositelnost kódu do jiných prostředí [39].

Tabulka 8 Šifrovací sady pro TLS 1.3 v modulu TLS

Verze protokolu TLS	**Šifrovací sada**	**Výměna klíčů**	**Digitální podpis**
1.3	TLS_AES_256_GCM_SHA384	ECDH (x25519)	ECDSA (P256), EdDSA (ed25519)

1.3	TLS_CHACHA20_POLY1305_SHA256	ECDH (x25519)	ECDSA (P256), EdDSA (ed25519)
1.3	TLS_AES_128_GCM_SHA256	ECDH (x25519)	ECDSA (P256), EdDSA (ed25519)
1.3	TLS_AES_128_CCM_SHA256	ECDH (x25519)	ECDSA (P256), EdDSA (ed25519)
1.3	TLS_AES_128_CCM_8_SHA256	ECDH (x25519)	ECDSA (P256), EdDSA (ed25519)

Výše uvedená tabulka (Tab. 8) vychází z následujících zdrojů [44][47]. TLS modul disponuje nejnovější verzí TLS 1.3, je možné použití starších šifrovacích sad z verze 1.2, ale to se nedoporučuje, jelikož nejsou tak bezpečná [47]. Poslední dvě šifrovací sady nejsou v základním nastavení TLS, protože nabízí nižší zabezpečení. Důvodem implementace těchto dvou šifrovacích sad je, že se hodí pro systémy s nižším výkonem [44]. Z tabulky (Tab. 8) lze vyhodnotit, že šifrovací sady využívají doporučené algoritmy, a tak se jeví jako bezpečné. Jelikož modul HTTPS přebírá modul TLS, tak šifrovací sady jsou stejné, proto nadále nebude zhotovena tabulka pro modul HTTPS [45].

Tabulka 9 Porovnání JWT modulů pro Node.js

Modul	**Node-jsonwebtoken**	**Jose**	**Aws-jwt-verify**
Součást Node.js	Ne	Ne	Ne
Licence	MIT	MIT	Apache
Verze	0.0.1	4.8.0	3.0.0
Poslední update	31.03.2021	26.04.2022	30.03.2022
Hodnocení na GitHub	15 211	2 126	177
Počet stažení za týden	4 362	2 290 383	13 483

Výše uvedená tabulka (Tab. 9) vychází z následujících zdrojů [50][51][52][53][54][55]. Dle tabulky lze vidět, že tyto moduly jsou třetích stran, proto je potřeba je doinstalovat pomocí správce balíčku NPM. Tyto moduly mají open-source licenci. Pouze u modulů Jose a Aws-jwt-verify proběhla tento aktualizace. Modul Node-jsonwebtoken již nebyl skoro rok aktualizovaný, proto se nejeví bezpečně, přesto tento modul má největší popularitu na GitHubu. Ale má nejmenší počet stažení za týden. Modul Aws-jwt-verify má z těchto modulů nejnižší popularitu, a to jak v počtu stažení za týden, tak i v hodnocení na stránce GitHub. S nejvyšším počtem stažení a s nejaktuálnější aktualizací je modul Jose.

Tabulka 10 Srovnání kryptografických operací JWT pro Node.js

Modul	**Node-jsonwebtoken**	**Jose**	**Aws-jwt-verify**
Digitální podpis	Ano	Ano	Ne
Ověření	Ano	Ano	Ano
HMAC SHA (256, 384, 512)	Ano	Ano	Ne
ECDSA (P-256)	Ano	Ano	Ne
EdDSA (Ed25519, Ed448)	Ne	Ano	Ne
ECDH (X25519, X448)	Ne	Ano	Ne
RSA	Ano	Ano	Ano

Výše uvedená tabulka (Tab. 10) vychází z následujícího zdroje [49]. Z tabulky lze vypozorovat, že modul Jose podporuje všechny doporučené algoritmy pro potřeby kryptografických operací tokenu JWT, tím se jeví jako nejlepší volba ve výběru modulu pro tyto účely. Node-jsonwebtoken nedisponuje pouze eliptickými křivkami Curve2551 a Curve448. Modul Aws-jwt-verify má nejmíň podporovaných a doporučených algoritmů, nepodporuje digitální podpis, HMAC se SHA-256, SHA-384 a SHA-512 a ani žádnou z doporučených eliptických křivek.

5.1.3.1 Dokumentace kryptografických modulu Node.js

V dokumentaci Crypto modulu jsou popsány všechny metody, zdá se však, že je potřeba mít nějaké kryptografické znalosti, pro pochopení implementace. Některé metody nemají ukázkový kód, a ty které mají, tak je obtížné z nich pochopit implementaci do reálného provozu.

Pro vývoj bezpečné aplikace se tento modul hodí spíš pro zkušené vývojáře v oblasti kryptografie [27].

Dokumentace modulu Web Crypto API je přehlednější a také nabízí tabulkový přehled jednotlivých algoritmů a srovnání, jaké kryptografické operace podporují. Taktéž u jednotlivých metod je popsáno, jaké algoritmy lze použít. Jsou tam i ukázkové kódy pro některé kryptografické operace, ale u jednotlivých metod tyto ukázky chybí. Jelikož je tento modul experimentální, tak je i možné, že tato dokumentace ještě není zcela dokončená [39].

Modul TLS má podobnou dokumentaci jako předchozí dva moduly. Jsou zde vysvětleny jednotlivé metodiky protokolu TLS a také doporučené kryptografické sady pro použití. Avšak pro nové vývojáře může být problém správně implementovat tento modul [44].

Dokumentace modulu HTTPS neobsahuje podrobný popis funkčnosti HTTPS. Zdá se být neúplná, a tak je potřeba mít již znalosti ze správného implementování HTTPS. Pro nové a nezkušené vývojáře není tato dokumentace vhodná [46].

Na stránce GitHub je popsán modul Node-jsonwebtoken, kde jsou vysvětlené metody a parametry s ukázkovými kódy. Je tam dostupný i seznam podporovaných algoritmů. Tato dokumentace je přehledná a vhodná pro implementaci JWT [50].

Dokumentace modulu Jose je taktéž na stránce GitHub. Jednotlivé operace pro JWT jsou rozdělené v seznamu, kde po kliknutí na zvolenou operaci, dojde k přesměrování na samostatnou stránku s ukázkovým kódem a popisem metod. Toto provedení je přehledné a vhodné pro implementaci JWT [51].

Poslední dokumentace modulu Aws-jwt-verify je stejně jak u předešlých dvou modulů pro implementaci JWT na stránce GitHub. Zde se nacházejí ukázkové kódy pro jednotlivé operace, ke kterým je i přidáno vysvětlení jejich parametrů. Na konci dokumentace jsou i ukázkové kódy k implementaci k některým frameworkům. Jedním z nich je i ukázka implementace ve frameworku Express.js. Tato dokumentace působí zmatečně, stejně jako dokumentace k modulům od Node.js [52].

5.1.4 Srovnání Node.js frameworků

Tabulka 11 Srovnání frameworků Express.js a Nest.js

Framework	**Express.js**	**Nest.js**
Popularita na StackOverflow	23.82 %	-
Počet stažení za týden	23 986 955	1 352 258
Licence	MIT	MIT
Návrhový vzor	MVC	MVC
Vydání	03.01.2010	14.03.2017
Nejnovější LTS verze	4.18.1	8.4.4
Vydání LTS verze	29.04.2022	07.04.2022
Nejnovější verze	4.18.1	8.4.4
Vydání poslední aktualizace	29.4.2022	07.04.2022
Hodnocení na GitHub	56 798	46 559
Počet otevřených dotazů na GitHub	96	50
Počet uzavřených dotazů na GitHub	3 596	4 062
Počet commitů na GitHub	5 734	10 442

Výše uvedená tabulka (Tab. 11) vychází z následujících zdrojů [25][56][57][58][66][67]. Z tabulky si lze všimnout, že framework Express.js je velice populární, a to jak v dotazníku, který provedl StackOverflow, tak i v počtu hvězd na stránce GitHub, ale také v počtu stažení za týden. Framework Nest.js vůbec nebyl v dotazníku zmíněn, ale v počtu stažení na týden má obstojné výsledky a také v hodnocení na GitHubu má vysoký počet hvězd. Tyto vysoké hodnoty ve stažení mohou být zapříčiněny nedávnou aktualizací pro tyto frameworky. Komunita pro oba frameworky na stránce GitHub se zdá být aktivní, to vypovídá vysokým počtem uzavřených dotazu, takže v případě nějakého problému ve vývoji, se lze obrátit na komunitu na GitHubu anebo na StackOverflow. Framework Express.js existuje již od roku 2010 a těší se vysoké popularitě čímž se jeví jakožto bezpečnou volbou pro tvorbu webové aplikace. Nest.js je o něco mladší, jelikož vyšel o 7 let později, ale jeho popularita je taktéž

vysoká. Na bezpečnost se podílí i jejich open-source licence, kde jakoukoliv chybu ve zabezpečení může kdokoliv odhalit.

5.1.4.1 Bezpečnost frameworků Express.js a Nest.js

Tabulka 12 Srovnání CVE frameworku Express.js verze 4.x

CVE ID	**Datum**	**CVSS**	**Verze**	**Datum aktualizace**
CVE-2017-1000048	13.02.2017	5.0	< 4.15.2	06.03.2017
-	11.04.2017	3.7	< 4.15.3	16.05.2017
CVE-2017-16137	05.09.2017	5.0	< 4.15.5	24.09.2017
CVE-2017-16119	10.09.2017	5.0	< 4.15.5	24.09.2017
CVE-2017-16118	2017	5.0	< 4.16.0	28.09.2017
CVE-2017-16138	2017	5.0	< 4.16.0	28.09.2017

Výše uvedená tabulka (Tab. 12) vychází z následujících zdrojů [58][59][60][61] [62][63][64][65]. Z tabulky si lze povšimnout, že ve verzi 4.x většina chyb byla okolo CVSS 5.0, což se považuje za střední hodnotu. Pro tyto chyby byly vydány aktualizace skoro po měsíci od data odhalení. U posledních dvou chyb nebyl nalezen přesný datum, ale dle CVE ID byla tato chyba nahlášena někdy v roce 2017. A u jedné zranitelnosti nebylo nalezeno CVE ID. Je nutné si uvědomit, že chyby v zabezpečení obsažené v Node.js přímo ovlivňuje Express.js, proto je potřeba také sledovat zranitelnosti v Node.js.

Na framework Nest.js nebyly nalezeny žádné bezpečnostní chyby, dle databáze zranitelnosti, všechny verze tohoto frameworku jsou bez bezpečnostních chyb [70]. Ale tím nelze tvrdit, že je zcela bezpečný. Jelikož je stavěný na runtime prostředí Node.js, tak je taktéž ovlivňován jeho zranitelnostmi.

5.1.4.2 Kryptografické moduly frameworků Express.js a Nest.js

Jelikož jsou tyto frameworky postavené na Node.js, tak lze využít jeho vestavěné moduly či moduly třetích stran, popsané v kapitole 5.1.3. Nest.js se v dokumentaci odkazuje pro potřeby šifrování a hashování dat na modul Crypto [71]. Avšak pro autorizaci přes JWT odkazuje v dokumentaci na knihovnu nest/jwt, která je postavená na modulu Node-

jsonwebtoken [72], který byl srovnáván v kapitole 5.1.3 a jeho kryptografické operace v tabulce (Tab. 10). Z toho důvodu nebude dále popisována.

5.1.4.3 Dokumentace frameworků Express.js a Nest.js

Dokumentace frameworku Express.js nabízí pro nové vývojáře sekci „Getting started“, kde je vysvětlen postup instalace s ukázkou fungování komunikace se serverem. V této ukázce lze upravit a otestovat reakci serveru s připadnou úpravou kódu. V sekci „guide“ je dopodrobna vysvětleno, jak například funguje „routing“ anebo jak integrovat databázový systém. Jsou tam obsaženy jak ukázkové kódy, tak podrobný popis, jak vše funguje. V sekci „API reference“ je k nejnovější ale také ke starším verzím frameworku podrobně vysvětlena funkčnost každé metody a jejich parametrů s přiloženým ukázkovým kódem. Tato dokumentace je velice kvalitně zpracována a pomůže novým vývojářům vytvořit kvalitní a bezpečnou webovou aplikaci [130].

Framework Nest.js má po vizuální stránce lepší dokumentaci. Strukturování jednotlivých komponent je taktéž přívětivější. Na rozdíl od Express.js, kde je dokumentace strukturovaná dle komponent, je zde strukturovaná do jednotlivých operací, jako je například Autentizace, Eventy, HTTP a Cookies. Tohle je mnohem přívětivější pro nové vývojáře, jelikož si nedokážou pod názvem nějaké metody představit, k čemu může sloužit. Tyto operace disponují s krokovým vysvětlením, kde k jednotlivým krokům, jsou ukázkové kódy. Ale nejsou dopodrobna vysvětleny jednotlivé parametry. V sekci „Overview“, je pro nové vývojáře vysvětlená instalace i fungování jednotlivých komponent daného frameworku. Nabízí také video kurzy pro nováčky, které jsou ovšem placeny. Tato dokumentace je velice kvalitní a díky ní nový vývojáři dokáži vytvořit zabezpečenou webovou aplikaci [69].

5.2 Python

5.2.1 Kryptografické knihovny jazyka Python

Tabulka 13 Porovnání kryptografických Python knihoven třetích stran

Knihovna	**PyCryptodome**	**Cryptography**	**PyNaCl**
Součást Pythonu	Ne	Ne	Ne
Licence	BSD	BSD	Apache
Implementace z knihovny	-	OpenSSL	Libsodium

Verze	3.14.1	37.0.2	1.5.0
Poslední update	05.02.2022	27.04.2022	07.01.2022
Hodnocení na GitHub	1 986	4 734	882
Počet stažení za týden	5 810 674	26 038 325	7 748 371

Výše uvedená tabulka (Tab. 13) vychází z následujících zdrojů [103][105][106][107] [108][109][110][111][112]. Z tabulky lze vidět, že se jedná o knihovny třetí strany s open-source licencí. Výhodou knihovny PyCryptodome je, že neobaluje jinou kryptografickou knihovnu, a tak není ovlivněna jejími zranitelnostmi. Avšak není tak velká jistota ve správné implementaci jejich algoritmů, jak je to v případě knihoven Cryptography a PyNaCl. Nejvíce populární knihovna je Cryptography, a to jak v počtu stažení za týden, tak i v hodnocení na stránce GitHub. Klíčová výhoda Cryptography spočívá v její implementaci standardizované knihovny OpenSSL. U této knihovny nedávno vyšla aktualizace, oproti knihovnám PyCryptography a PyNaCl, kde aktualizace vyšly před několika měsíci. Přesto jsou podle počtu stažení za týden knihovny PyCryptodome a PyNaCl populární. Z těchto poznatku se knihovna Cryptography zdá jako nejvíce bezpečná.

Tabulka 14 Porovnání kryptografických operací knihoven Pythonu

Knihovna	**PyCryptodome**	**Cryptography**	**PyNaCl**
RBG	Ano	Ano	Ano
AEAD	Ano	Ano	Ano
Digitální podpis	Ano	Ano	Ano
Protokol pro výměnu klíčů	Ano	Ano	Ano
Hash	Ano	Ano	Ano
HMAC	Ano	Ano	Ano
KDF	Ano	Ano	Ano
X.509	Ne	Ano	Ano

Post-kvantová kryptografie	Ne	Ne	Ne

Výše uvedená tabulka (Tab. 14) byla sestavena na základě těchto zdrojů [103][104][110]. Tyto knihovny obsahují všechny algoritmy pro základní kryptografické operace, ale pouze knihovna PyCryptodome neimplementuje standard X.509. Ani jedna ze zmiňovaných knihoven neimplementuje post-kvantovou kryptografii. Pro potřeby implementace post-kvantové kryptografie je možno použít knihovnu Liboqs-python, která je popsaná v kapitole 5.5. Co se týče PyNaCl, tak se jedná o vysokoúrovňovou kryptografickou knihovnu a některé operace jsou již zakomponované v jejich metodách, proto jsou v tabulce označené jako „Ano“.

Tabulka 15 Porovnání doporučených algoritmů v knihovně PyCryptodome

	Algoritmy
AEAD	Chacha20-poly1305, AES-256-GCM
Digitální podpis	ECDSA (P256), RSA
Protokol pro výměnu klíčů	-
Hash	SHA-2 (256,384,512), SHA-3 (256,384,512), BLAKE2b, BLAKE2s
KDF	Scrypt

Výše uvedená tabulka (Tab. 15) vychází z následujících zdrojů [103][113]. Z tabulky si lze všimnout, že knihovna PyCryptodome disponuje pouze jedním ECC algoritmem, a to pro digitální podpis. Protokol pro výměnu klíčů nedisponuje žádným doporučeným algoritmem a ostatní kryptografické operace jimi disponují. HMAC, který není v tabulce zmíněný, využívá popsaných algoritmů hash. Avšak do budoucna mají v plánu přidat více ECC křivek a algoritmy pro výměnu klíčů [114]. Z těchto poznatku lze usoudit, že se momentálně tato knihovna nejeví vhodná pro bezpečnou implementaci kryptografie.

Cryptography staví na knihovně OpenSSL a tu také implementuje Node.js modul Crypto. Tento modul již byl popsaný v kapitole 5.1.3 a srovnání doporučených algoritmů již bylo

zhotovené v tabulce (Tab. 6). Z toho důvodu nebude nadále Cryptography srovnáván, jelikož implementují stejné algoritmy z knihovny OpenSSL.

Tabulka 16 Porovnání doporučených algoritmů v knihovně PyNaCl

	Algoritmy
AEAD	-
Digitální podpis	EdDSA (ed25519)
Protokol pro výměnu klíčů	ECDH (x25519)
Hash	SHA-2 (256,512), BLAKE2b
KDF	Scrypt, Argon2

Výše uvedená tabulka (Tab. 16) vychází z následujících zdrojů [115][116][117][118]. Jelikož se jedná o vysokoúrovňovou kryptografickou knihovnu, tak zde není potřeba konfigurace či výběru algoritmů. Výjimkou jsou hash, HMAC a KDF, kde si lze vybrat z doporučených algoritmů. Co se týče digitálního podpisu a protokolu o výměně klíčů, tak metody tyto knihovny pracují s křivkou Curve25519. V AEAD pracuje se schématem XChacha20-Poly130, které v teoretické části není mezi doporučenými. Z toho důvodu se tato knihovna nejeví příliš bezpečnou.

Tabulka 17 Srovnání vestavěných kryptografických knihoven jazyka Python

Knihovna	**Hashlib**	**Ssl**
Součásti Pythonu	Ano	Ano
Licence	BSD	BSD
Implementace z knihovny	-	OpenSSL
Verze	3.10.4	3.10.4
Poslední update	26.04.2022	26.04.2022

Výše uvedená tabulka (Tab. 17) vychází z následujících zdrojů [119][120]. Programovací jazyk Python disponuje open-source licenci, proto i jeho knihovny mají tuto licenci. Jejich verze a update se odvozují taktéž od programovacího jazyka Python. Ssl staví na knihovně

OpenSSL a tu také implementuje Node.js modul Tls. Tento modul již byl popsaný v kapitole 5.1.3 a jeho šifrovací sady pro TLS 1.3 byly srovnány v tabulce (Tab. 8). Proto bude pouze srovnávaná knihovna Hashlib.

Tabulka 18 Srovnání doporučených algoritmů v knihovně Hashlib

	Algoritmy
AEAD	-
Digitální podpis	-
Protokol pro výměnu klíčů	-
Hash	SHA-2 (256, 384, 512), SHA-3 (256, 384, 512), BLAKE2b, BLAKE2s
KDF	Scrypt

Výše uvedená tabulka (Tab. 18) vychází z následujícího zdroje [119]. Z tabulky si lze všimnout, že tento modul je pouze zaměřený na hash, HMAC a KDF funkce. Pro tyto účely disponuje doporučenými algoritmy, takže se jeví jakožto bezpečná pro účely implementace těchto funkcí.

Tabulka 19 Porovnání JWT knihoven pro jazyk Python

Knihovna	**PyJWT**	**Python-jose**	**Authlib**
Součást Pythonu	Ne	Ne	Ne
Licence	MIT	MIT	BSD
Verze	2.3.0	3.3.0	1.0.1
Poslední update	18.10.2021	5.6.2021	6.4.2022
Hodnocení na GitHub	4 190	1 088	3 083
Počet stažení za týden	20 865 164	1 313 890	1 162 527

Výše uvedená tabulka (Tab. 19) vychází z následujících zdrojů [121][122][123] [124][125][126]. Tyto knihovny třetí strany pro účely JWT disponují open-source licencí. V počtu stažení za týden a v hodnocení na stránce GitHub je nejpopulárnější knihovnou

PyJWT, i přestože poslední aktualizace byla vydána před více jak půl rokem. Knihovna Python-jose se také těší vysoké popularitě, i u této knihovny byla vydána aktualizace skoro před rokem. Nejbezpečnější, a také poměrně populární knihovnou se jeví Authlib, u které aktualizace proběhla před měsícem.

Tabulka 20 Srovnání kryptografických operací JWT pro Python

Knihovna	**PyJWT**	**Python-jose**	**Authlib**
Digitální podpis	Ano	Ano	Ano
Ověření	Ano	Ano	Ano
HMAC SHA (256, 384, 512)	Ano	Ano	Ano
ECDSA (P-256)	Ano	Ne	Ano
EdDSA (Ed25519, Ed448)	Ano	Ne	Ano
ECDH (X25519, X448)	Ne	Ne	Ano
RSA	Ano	Ano	Ano

Výše uvedená tabulka (Tab. 20) vychází z následujících zdrojů [49][127][128]. V této tabulky si lze všimnou, že jediná knihovna Authlib disponuje všemi doporučenými algoritmy pro kryptografické účely JWT. Knihovna PyJWT pouze nedisponuje ECDH křivkami a Python-jose neobsahuje žádnou z doporučených ECC algoritmů. Z tohoto porovnání lze usoudit, že knihovna Authlib je nejvhodnější pro bezpečné implementování JWT.

5.2.1.1 Dokumentace kryptografických knihoven jazyka Python

V dokumentaci knihovny PyCryptodome je návod pro dva různé způsoby instalace s vysvětlením pro rozhodnutí o výběru. V záložce „Feature“ je podrobný výpis jednotlivých kryptografických operací se seznamem podporovaných šifer. Dokumentace je rozdělená do balíčku dané knihovny, které obsahují teoretické vysvětlení a doporučení o výběru bezpečného algoritmu a vyhnutí se již nebezpečných algoritmu. Jednotlivé funkce disponují s vysvětlením parametrů s ukázkovým kódem. Více ukázkového kódu lze najít v sekci „Examples“. Tato dokumentace je přehledně strukturovaná a je napsaná na platformě Read the Docs, která slouží pro technické dokumentace. Je vhodná pro implementování a vhodný výběr kryptografických algoritmů [131].

Knihovna Cryptography má na úvodní stránce doporučení k výběr dvou způsobů pro implementaci kryptografie. Jeden z nich je vysokoúrovňový, který je doporučen pro vývojáře, kteří nemají zkušenosti se správnou konfigurací kryptografických algoritmů. Metody vysokoúrovňové kryptografie lze najít v záložce „The recipes layer“. Metody obsahují stručný popis s vysvětlením jednotlivých parametrů a ukázkové kódy. Druhý způsob je nízkoúrovňová kryptografie, která se nachází v záložce „The hazardous materials layer“. Je zde varování, že tato část je určená pouze pro vývojáře, kteří si jsou jistí se správnou implementací kryptografických operací. Což může zabránit v nebezpečné implementaci, která by vedla k úniku dat. Je zde vysvětlení jednotlivých funkcí a jejich parametrů s přiloženými ukázkovými kódy. Tato dokumentace je přehledně strukturovaná a je také napsána v platformě Read the Docs. Dokumentace je vhodná jak pro méně zkušené, tak i pro zkušené vývojáře v oblasti kryptografie [104].

Dokumentace knihovny PyNaCl je taktéž přehledná, jelikož je napsaná v Read the Docs. Dokumentace je strukturovaná do jednotlivých kryptografických operací, ve kterých se nachází metody, které jsou zaměřené na vysokoúrovňovou kryptografii s podrobným vysvětlením funkčnosti a jejich parametrů. U jednotlivých metod je několik ukázkových kódu, ze kterých vývojář pochopí implementaci kryptografických operací [132].

Dokumentace knihovny Hashlib nabízí výpis doporučených hash a KDF algoritmů. Nachází se tam i popis jednotlivých parametrů funkcí s ukázkovými kódy. Tato dokumentace je velice přehledná a jednoduchá, protože je tato knihovna zaměřená pouze na hash, KDF a HMAC. Vývojářům ulehčí bezpečnou implementací těchto kryptografických operací [119].

Knihovna Ssl disponuje obsáhlou dokumentací, kde je každá funkce popsaná i s jejími parametry. Nachází se zde ukázkové kódy s vysvětlením, jestli se využívá na straně klienta či serveru. Pro vývojáře je tato dokumentace užitečná, jelikož je kvalitně zpracovaná [120].

Dokumentace PyJWT je taktéž na platformě Read the Docs, díky které je poměrně přehledná. V sekci „Usage Examples“ jsou ukázkové kódy pro jednotlivé funkcionality JWT. U některých ukázkových kódu se nachází popis, pro které případy se taková funkcionalita využívá. Nadále v sekci „Digital Signature Algorithms“ je pospaný seznam podporovaných algoritmů. Díky této dokumentaci je možno správně implementovat JWT [133].

Knihovna Python-jose se zdá být nedokončená, jelikož se zde nachází čtyři záložky, které jsou úplně prázdné. Není zde popsaná funkcionalita a ani popis jednotlivých parametrů

funkcí. Nachází se zde málo ukázkových kódu, které se zdají být neúplné. Tato dokumentace je nevhodná, jelikož z ní nelze pochopit správná implementace JWT [134].

Dokumentace knihovny Authlib je nejrozsáhlejší a nejkvalitněji zpracována ze zmíněných JWT knihoven. Je konstruovaná do jednotlivých operací této knihovny, kde je návod pro implementaci na straně klienta a na straně serveru. Také disponuje těmito návody pro implementaci v případě používání frameworků Flask a Django. Tyto návody jsou podrobně popsány s ukázkovými kódy a vysvětlením, jak daný kód funguje. Pro implementování JWT je tato knihovna nejvhodnější, jelikož vývojářům pomůže k dosažení bezpečné komunikaci pomocí JWT i v případě používání zmiňovaných frameworků [135].

5.2.2 Srovnání frameworků jazyka Python

Tabulka 21 Srovnání obecných informací frameworků Flask a Django

Framework	**Flask**	**Django**
Popularita na StackOverflow	16.14 %	14.99 %
Počet stažení za týden	13 706 998	1 742 272
Licence	BSD	BSD
Návrhový vzor	Jakýkoliv	MTV
Vydání	16.04.2010	21.07.2005
Nejnovější LTS verze	2.1.2	3.2.13
Vydání LTS verze	28.04.2022	11.04.2022
Nejnovější verze	2.1.2	4.0.4
Vydání poslední aktualizace	28.04.2022	11.04.2022
Hodnocení na GitHub	58 809	63 810
Počet otevřených dotazů na GitHub	19	143
Počet uzavřených dotazů na GitHub	2 294	15 548
Počet commitů na GitHub	4 630	30 713

Výše uvedená tabulka (Tab. 21) vychází z následujících zdrojů [25][76][77][78][79] [80][81][82][83][84][85]. Z tabulky si lze všimnout, že oba frameworky disponují open-source licencí a jsou již na trhu několik let. V dotazníku od stránky Stackoverflow mají obstojné výsledky v popularitě a v počtu stažení za týden framework Flask dominuje. Django co se týče v počtu stažení zaostává, ale i tak byl stažen mnoho uživateli. Komunita na stránce GitHub je početná, což lze usoudit jak v počtu hodnocení, tak i v počtu dotazů. U obou frameworků proběhlo nedávné vydaní nové verze a u Djanga si lze povšimnout, že je stále podporovaná i jeho starší LTS verze, pro kterou byla vydaná bezpečnostní aktualizace. Jelikož je Flask mikro-framework, tak lze aplikovat jakýkoliv návrhový vzor, a to například MVC anebo Single Page [78]. Z těchto důvodů lze usoudit, že tyto frameworky jsou udržované, aktualizované a pod dohledem komunity, čímž se jeví jako bezpečné. Také při řešení nějakého problému vývojářem, tak lze s touto početnou komunitou dohledat řešení na internetových fórech.

5.2.2.1 Bezpečnost frameworků Flask a Django

Tabulka 22 Srovnání CVE frameworku Flask

CVE ID	**Datum**	**CVSS**	**Verze**	**Datum aktualizace**
CVE-2018-1000656	10.04.2018	5.0	> 0.12.3	26.04.2018
CVE-2019-1010083	-	5.0	> 1.0	26.04.1018

Výše uvedená tabulka (Tab. 22) vychází z následujících zdrojů [86][87][88][89]. K tomuto frameworku byly nalezeny pouze dvě zranitelnosti, to může být zapříčiněno tím, že je Flask mikro-framework, a tak nemá mnoho knihoven ve kterých by mohly být zranitelnosti. Dle tabulky byly u dříve dvou podporovaných verzí, dvě zranitelnosti se středním CVSS, jedna z nich byla opravená rychle, u druhé nebyl nalezený příslušný článek s nahlášením. Tím se Flask zdá být bezpečným frameworkem pro vývoj webové aplikace.

Tabulka 23 Srovnání počtu CVSS pro Django v jednotlivých letech

CVSS / ROK	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10
2009	-	-	-	-	1	-	-	-	-
2010	-	-	-	1	-	-	-	-	-
2011	-	-	-	2	5	2	-	-	-
2012	-	-	-	1	2	1	-	-	-
2013	-	-	-	3	3	-	-	-	-
2014	-	-	1	2	4	1	-	-	1
2015	-	-	-	4	8	-	-	2	-
2016	-	1	-	2	2	1	1	-	-
2017	-	-	-	1	2	-	-	-	-
2018	-	-	-	1	4	-	-	-	-
2019	-	-	-	3	6	-	1	-	-
2020	-	-	-	2	2	1	1	-	-
2021	-	-	-	2	4	-	2	-	-
2022	-	-	-	1	4	-	2	-	-

Obrázek 12 Počet CVSS pro Django dle jednotlivých let [101]

Výše uvedená tabulka (Tab. 23) vychází z následujícího zdroje [101]. Dle tabulky bylo zjištěno, že se ve frameworku Django od doby jeho existence objevovali zranitelnosti převážně se střední hodnotou CVSS a několik s vysokou hodnotou CVSS.

Tabulka 24 Srovnání CVE frameworku Django verzí 3.2.x a 4.0.x

CVE ID	Datum	CVSS	Verze	Datum aktualizace
CVE-2022-28346	-	7.5	3.2 - 3.2.12, 4 - 4.0.3	11.04.2022
CVE-2022-28347	-	7.5	3.2 - 3.2.12, 4 - 4.0.3	11.04.2022
CVE-2022-23833	-	5.0	3.2 - 3.2.11, 4 - 4.0.1	01.02.2022
CVE-2022-22818	-	4.3	3.2 - 3.2.11, 4 - 4.0.1	01.02.2022
CVE-2021-45452	-	5.0	3.2 - 3.2.10, 4.0.0	04.01.2022
CVE-2021-45115	-	5.0	3.2 > 3.2.11, 4.0.0	04.01.2022
CVE-2021-45116	-	5.0	3.2 - 3.2.10, 4.0.0	04.01.2022

Výše uvedená tabulka (Tab. 24) vychází z následujících zdrojů [90][91][92][93][94] [95][96][97][98][99]. Dle tabulky si lze všimnout, že dvě zranitelnosti dle CVSS byly vysoké a ostatní střední. Tyto zranitelnosti byly opraveny, ale nelze usoudit za jaké časové období, jelikož všechny bezpečnostní chyby jsou hlášeny na e-mail a jejich archív není zveřejněn [100]. Tímto se Django jeví velice bezpečně, jelikož nalezené zranitelnosti jsou oznámené až k vydané aktualizaci, takže těchto chyb nemůžou zneužít případní útočníci.

5.2.2.2 Kryptografické knihovny frameworků Flask a Django

Framework Flask nedisponuje vlastní kryptografickou knihovnou, ale je možno použít nějakou ze zmiňovaných knihoven třetích stran, které byly popsány v kapitole 5.2.1.

Framework Django disponuje pouze knihovnou pro hashování zvanou Signer, ta však využívá algoritmů knihovny Hashlib [136], která byla popsána v kapitole 5.2.1. a její doporučené algoritmy srovnány v tabulce (Tab. 18). Z toho důvodu již tato knihovna nebude

srovnávaná. Pro ostatní kryptografické operace je možno použít některou z popsaných knihoven třetích stran z kapitoly 5.2.1.

5.2.2.3 Dokumentace frameworků Flask a Django

Dokumentace frameworku Flask je velice obsáhlá a přehledná. Nachází se tam jak popis samotného frameworku, tak i návod k instalaci a prvnímu spuštění. Je tam vše, co vývojář potřebuje k bezpečné implementaci a pochopení fungování frameworku. Jednotlivé metody obsahují popis funkčnosti a parametrů s ukázkovými kódy. Vše je strukturované v přehledném seznamu, kde se vývojář lehce zorientuje [137].

Stejně jak u frameworku Flask, Django má obsáhlou a přehlednou dokumentaci strukturovanou do jednotlivých části. Prvním z nich je „First steps“, kde je podrobný návod pro uživatele, kteří jsou začátečníci v programování anebo v používání tohoto frameworku. Další část je „Topic guides“, kde lze najít vysvětlení hlavních funkcí a částí Djanga. Část „Reference guides“ obsahuje různé metody tohoto frameworku důležité k pochopení funkčnosti. A poslední „How-to guides“, kde jsou popsané řešení pokročilých problémů. Vše obsahuje podrobný popis s ukázkovými kódy. Tato dokumentace je vhodná pro pochopení a správné implementace frameworku Django [138].

5.3 Jazyk C#

5.3.1 Kryptografické knihovny .NET

Tabulka 25 porovnání kryptografických knihoven .NET

Knihovna	**Cryptography**	**Norgerman.Cryptography.Scrypt**
Součást .NET	Ano	Ne
Licence	MIT	MIT
Implementace z knihovny	-	-
Verze	6.0.4	2.3.0
Poslední update	12.04.2022	12.11.2021
Hodnocení na GitHub	-	3
Počet stažení aktuální verze	-	262

Výše uvedená tabulka (Tab. 25) vychází z následujících zdrojů [144][145][146][147]. Z Tabulky si lze všimnout, že knihovna Cryptography je implementovaná v balíčku .NET, která také nedávno byla aktualizovaná. Druhá knihovna pro KDF Scrypt, je knihovnou třetí strany a nedisponuje vysokou popularitou, čímž se nejeví jako bezpečná. Tato knihovna pro KDF, byla nalezena jako s nejvyšším počtem stažení a s nejnovějším datem aktualizace. Jelikož není vestavěná a neimplementuje nějakou ověřenou knihovnu, tak není jisté, že její implementace algoritmu Scrypt bezpečná.

Tabulka 26 Porovnání kryptografických operací knihoven .NET

Knihovna	**Cryptography**	**Norgerman.Cryptography.Scrypt**
RBG	Ano	Ne
AEAD	Ano	Ne
Digitální podpis	Ano	Ne
Protokol pro výměnu klíčů	Ano	Ne
Hash	Ano	Ne
HMAC	Ano	Ne
KDF	Ano	Ne
X.509	Ano	Ne
Post-kvantová kryptografie	Ne	Ne

Výše uvedená tabulka (Tab. 26) vychází z následujících zdrojů [146][148]. Dle tabulky si lze všimnout, že knihovna Cryptography disponuje všemi kryptografickými operacemi, až na post-kvantovou kryptografii. Pro implementaci post-kvantových algoritmů je možno použít knihovnu Liboqs-dotnet, která je popsaná v kapitole 5.5. Druhá knihovna Norgerman.Cryptography.Scrypt je zaměřená pouze na KDF a tak neobsahuje algoritmy pro ostatní kryptografické operace.

Tabulka 27 Srovnání doporučených algoritmů v knihovně .NET Cryptography

	Algoritmy
AEAD	Chacha20-poly1305, AES-256-GCM
Digitální podpis	ECDSA (P256), RSA
Protokol pro výměnu klíčů	ECDH (P256), RSA
Hash	SHA-2 (256,384,512)
KDF	-

Výše uvedená tabulka (Tab. 27) vychází z následujícího zdroje [148]. Z této tabulky si lze všimnout, že disponuje pro všechny kryptografické operace nějakým doporučeným algoritmem, až na KDF, kde je potřeba použít knihovnu třetí strany. Tato knihovna disponuje pouze jednou doporučenou eliptickou křivkou P256. Pro hashování souboru disponuje pouze jedním doporučeným algoritmem SHA-2. Což je nevýhoda, jelikož vývojáři nemají více možností, pro výběr bezpečného algoritmu. Jednou knihovnou třetí strany, pro použití KDF algoritmu je Norgerman.Cryptography.Scrypt, která poskytuje algoritmus Scrypt.

Tabulka 28 Multiplatformní použití .NET Cryptography knihovny

Operační systém	**Windows**	**Linux**	**macOS**
AES-256-GCM	Knihovna OS	Knihovna OS	Knihovna OpenSSL
RSA	Knihovna OS	Knihovna OS	Knihovna OpenSSL
ECDSA (P256)	Knihovna OS	Knihovna OpenSSL	Knihovna OpenSSL
ECDH (P256)	Knihovna OS	Knihovna OpenSSL	Knihovna OpenSSL
SHA-2	Knihovna OS	Knihovna OS	Knihovna OS

Výše uvedená tabulka (Tab. 28) vychází z následujícího zdroje [142]. V této tabulce si lze všimnout, že tato knihovna používá algoritmy, které jsou poskytnuté operačním systémem. To má výhodu, že tato knihovna čerpá ze spolehlivosti daného operačního systému, jelikož knihovny operačního systému, jsou většinou ověřené standardem FIPS. Avšak nevýhoda je

taková, že daný operační systém nemusí poskytovat dany algoritmus, a tak je zapotřebí nainstalovat knihovnu OpenSSL [142].

Tabulka 29 Porovnání JWT knihoven pro .NET

Modul	**Jwt**	**Jose-jwt**
Součást .NET	Ne	Ne
Licence	MIT	MIT
Verze	9.0.0	4.0.0
Poslední update	27.04.2021	06.04.2022
Hodnocení na GitHub	1 797	771
Počet stažení aktuální verze	7 000	12 900

Výše uvedená tabulka (Tab. 29) vychází z následujících zdrojů [149][150][151][152]. Z této tabulky si lze všimnout, že tyto knihovny nejsou součástí .NET. Disponují open-source licencí a jejich aktualizace proběhla nedávno, čímž se jeví jakožto udržované knihovny. Avšak jejich popularita není příliš vysoká, čímž se nezdají příliš bezpečně.

Tabulka 30 Srovnání kryptografických operací JWT pro .NET

Knihovna	**Jwt**	**Jose-jwt**
Digitální podpis	Ano	Ano
Ověření	Ano	Ano
HMAC SHA (256, 384, 512)	Ano	Ano
ECDSA (P-256)	Ne	Ano
EdDSA (Ed25519, Ed448)	Ne	Ne
ECDH (X25519, X448)	Ne	Ne
RSA	Ano	Ano

Výše uvedená tabulka (Tab. 30) vychází z následujících zdrojů [49][152]. Knihovna Jwt nedisponuje žádnou eliptickou křivkou a knihovna Jose-jwt nemá pouze křivky Curve448 a

Curve25519. Tyto knihovny disponuji ostatními doporučenými algoritmy pro bezpečnou implementaci JWT.

5.3.1.1 Dokumentace kryptografických knihoven .NET

Dokumentace .NET knihovny Cryptography je rozdělená na dvě části, jedná část obsahuje popis jednotlivých kryptografických metod s teoretickým vysvětlením a ukázkovými kódy. Následně se zde nachází vysvětlení, jak funguje šifrování na jiných operačních systémech s přehledem fungování této knihovny v .NET. Tato část je přehledná a dá se v ní snadno zorientovat [141]. Druhá část je již samotné knihovny Cryptography, kde je seznam její tříd. Každá třída obsahuje popis parametrů s ukázkovými kódy. Tato část je trochu nepřehledná, jelikož tyto třídy nejsou strukturované do kryptografických kategorií [148].

Knihovna Norgerman.Cryptography.Scrypt nedisponuje žádnou dokumentací, pouze na stránce GitHub je jeden řádek kódu bez žádného popisu [146].

Knihovna Jwt má dokumentaci na stránce GitHub. Zde se nachází pouze ukázkové kódy, pro různé implementace JWT. Není zde žádný popis funkcí, parametrů a ani se zde nenachází seznam podporovaných algoritmů, proto není příliš vhodná pro vývojáře [150].

Knihovna Jose-jwt je taktéž na stránce GitHub. Nachází se zde ukázkové kódy pro jednotlivé kryptografické operace JWT s použitím pro různé algoritmy. Tyto metody jsou lehce popsané teoreticky s občasným popisem parametrů. Nachází se zde podrobný seznam podporovaných algoritmů. Dokumentace této knihovny je pro vývojáře vhodnější než ty předchozí [152].

5.3.2 Srovnání frameworku ASP .NET Core

Tabulka 31 Srovnání obecných informací frameworku ASP .NET CORE

Framework	**ASP .NET Core**
Popularita na StackOverflow	18.10 %
Licence	MIT
Návrhový vzor	MVC
Vydání	27.06.2016
Nejnovější LTS verze	6.0.4

Vydání LTS verze	12.04.2022
Nejnovější verze	5.0.16
Vydání poslední aktualizace	12.04.2022
Hodnocení na GitHub	28 130
Počet otevřených dotazů na GitHub	2 242
Počet uzavřených dotazů na GitHub	23 309
Počet commitů na GitHub	49 283

Výše uvedená tabulka (Tab. 31) vychází z následujících zdrojů [25][140][145][153]. Tento framework je populární v dotazníku od stránky Stackoverflow. O jeho popularitě také vypovídá počet hvězd na stránce GitHub. Tento open-source framework je již na trhu několik let a je pravidelně aktualizovaný, to lze vyvodit z počtu commitů na stránce GitHub. Jeho komunita je taktéž početná, protože má několik tisíc uzavřených a otevřených dotazů. U obou verzí frameworku proběhla nedávno aktualizace, z čehož lze usoudit, že jsou stále podporované.

5.3.2.1 Bezpečnost ASP .NET Core

Tabulka 32 Srovnání počtu CVSS pro ASP .NET Core v jednotlivých letech

CVSS / ROK	**1-2**	**2-3**	**3-4**	**4-5**	**5-6**	**6-7**	**7-8**	**8-9**	**9-10**
2017	-	-	-	1	1	-	-	-	-
2018	-	1	-	2	5	2	-	-	-
2019	-	-	-	-	5	1	-	-	-
2020	-	-	-	-	4	-	-	-	1
2021	-	1	-	1	1	-	-	-	-
2022	-	-	-	-	-	-	-	-	-

Obrázek 13 Počet CVSS pro ASP .NET Core dle jednotlivých let [155]

Výše uvedená tabulka (Tab. 32) vychází z následujícího zdroje [155]. Z tabulky si lze všimnout, že v minulosti nebylo mnoho zranitelnosti. Většina těchto zranitelnosti se pohybuje ve střední hodnotě CVSS a některé v nízké hodnotě. Pouze jedna zranitelnost byla kritická. Z těchto údajů se tento framework jeví jakožto bezpečným.

Tabulka 33 Srovnání CVE pro ASP .NET Core v letech 2020 a 2021

CVE ID	**Datum**	**CVSS**	**Verze**	**Datum aktualizace**
CVE-2020-1597	-	5.0	> 2.1.2, >3.16	12.04.2022
CVE-2020-1161	-	5.0	> 3.1.3	12.04.2022
CVE-2020-1045	-	5.0	> 2.1.21, > 3.1.7	12.04.2022
CVE-2020-0603	-	9.3	2.1.x, 3.0.x, 3.1.x	17.08.2021
CVE-2020-0602	-	5.0	2.1.x, 3.0.x, 3.1.x	17.08.2021

CVE-2021-43877	-	4.6	3.1.x, 5.0.x, 6.0.x	12.04.2022
CVE-2021-34532	-	2.1	> 3.1.17, > 5.0.8	12.04.2022
CVE-2021-1723	-	5.0	> 3.1.10, > 5.0.1	12.04.2022

Výše uvedená tabulka (Tab. 33) vychází z následujících zdrojů [156][157][158][159][160][161][162][163][164][165][166][167][168]. Protože Microsoft disponuje vlastním dotazníkem na nahlašování chyb, nelze dohledat datum nahlášení dané zranitelnosti [169]. Což je výhoda, jelikož potencionální útočníci nemohou využít dané slabiny. Avšak lze odhadnou dle CVE ID, že některé aktualizace trvaly déle než rok. Čímž se nejeví moc bezpečně, ale jelikož tyto zranitelnosti nebyly zveřejněny, tak jich pravděpodobně nikdo nezneužil.

5.3.2.2 Kryptografické knihovny ASP .NET Core

ASP .NET Core implementuje knihovnu Cryptography anebo knihovny třetích stran popsané v kapitole 5.3.1.

5.3.2.3 Dokumentace ASP .NET Core

Dokumentace frameworku ASP .NET Core nabízí jednotlivé návody pro začátečníky. Tyto návody jsou členěny do důležitých komponent pro implementaci frameworku. Nabízí seznam scénářů, například server-side nebo client-side vývoj, ke kterým je odkaz, jenž vývojáře přesměruje na stránku s návodem. Tyto návody nabízí teoretické vysvětlení, ukázkové kódy a screenshoty, jak například přidat do projektu novou komponentu. Tato dokumentace je kvalitně zpracovaná a velice rozsáhlá. Dokumentace se jeví, jako nápomocná a může pomoct budoucím vývojářům vytvořit bezpečnou webovou aplikaci [170].

5.4 OpenSSL

Tabulka 34 Srovnání obecných informací knihovny OpenSSL

Knihovna	**OpenSSL**
Licence	Apache
Vydání	23.12.1998
Nejnovější LTS verze	3.0.3
Vydání LTS verze	03.05.2022
Nejnovější verze	1.1.1
Vydání poslední aktualizace	03.05.2022
Hodnocení na GitHub	18 340
Počet otevřených dotazů na GitHub	1 466
Počet uzavřených dotazů na GitHub	5 657
Počet commitů na GitHub	31 196
Počet stránek využívajících OpenSSL	2 704 839

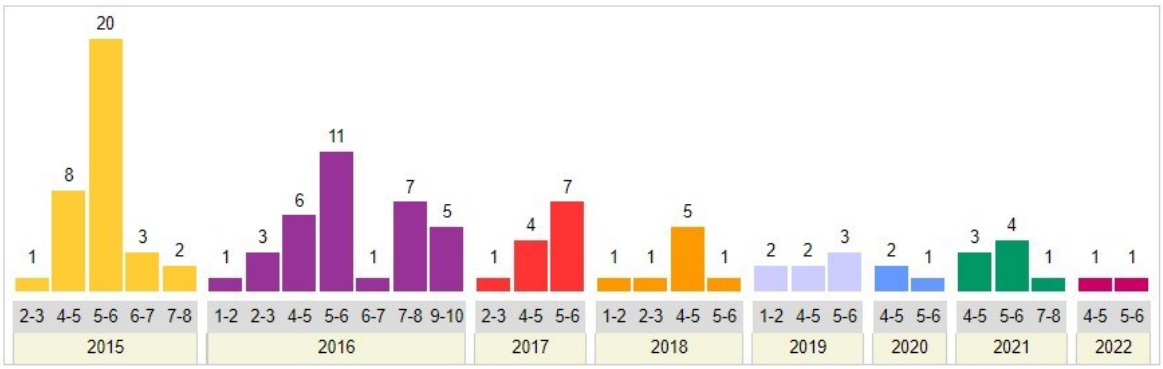
Výše uvedená tabulka (Tab. 34) vychází z následujících zdrojů [41][42][173][174]. Z tabulky si lze všimnou, že tato knihovna je na trhu přes 20 let a pyšní se poměrně vysokou popularitou, a to jak počtem využívajících stránek, tak i s komunitou na stránce GitHub. Tato knihovna podporuje dvě verze, u kterých nedávno proběhla aktualizace. Z toho lze usoudit, že se jeví jakožto bezpečnou kryptografickou knihovnou.

Jelikož modul Crypto od Node.js, implementuje tuto knihovnu, tak není potřeba již vypisovat doporučené algoritmy, protože byly pospané v kapitole 5.1.3 a srovnané v tabulce (Tab. 6). Taktéž podporuje nejnovější protokol pro zabezpečenou komunikaci a to TLS 1.3 [41], jehož doporučené šifrovací sady byly popsané v kapitole 5.1.3 a porovnané v tabulce (Tab. 8).

5.4.1 Bezpečnost knihovny OpenSSL

Tabulka 35 Srovnání CVE knihovny OpenSSL v jednotlivých letech

CVSS / ROK	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10
2015	-	1	-	8	20	3	2	-	-
2016	1	3	-	6	11	1	7	-	5
2017	-	1	-	4	7	-	-	-	-
2018	1	1	-	5	1	-	-	-	-
2019	2	-	-	2	3	-	-	-	-
2020	-	-	-	2	1	-	-	-	-
2021	-	-	-	3	4	-	1	-	-
2022	-	-	-	1	1	-	-	-	-



Obrázek 14 Počet CVSS pro OpenSSL dle jednotlivých let [175]

Výše uvedená tabulka (Tab. 35) vychází z následujícího zdroje [175]. Z tabulky si lze všimnout, že tato knihovna disponovala většinou zranitelnosti v CVSS okolo střední hodnoty. Některé se pohybují v nízké nebo vysoké hodnotě a 5 zranitelností z roku 2016 v kritické hodnotě.

Tabulka 36 Srovnání CVE pro OpenSLL v letech 2020 a 2021

CVE ID	**Datum**	**CVSS**	**Verze**	**Datum aktualizace**
CVE-2021-23839	-	4.3	1.0.2s - 1.0.2x	16.02.2021
CVE-2021-23840	-	5.0	1.1.1 - 1.1.1i, 1.0.2 - 1.0.2x	16.02.2021
CVE-2021-23841	-	4.3	1.1.1 - 1.1.1i, 1.0.2 - 1.0.2x	16.02.2021
CVE-2021-3449	-	4.3	1.1.1 - 1.1.1j	25.03.2021
CVE-2021-3450	-	5.8	1.1.1 - 1.1.1j	25.03.2021
CVE-2021-3711	-	7.5	1.1.1 - 1.1.1k	24.08.2021
CVE-2021-3712	-	5.8	1.1.1 - 1.1.1k, 1.0.2 - 1.0.y	24.08.2021
CVE-2021-4044	-	5.0	3.0.0	14.12.2021
CVE-2021-4160	-	4.3	3.0.0, 1.1.1 -1.1.1l, 1.0.2 – 1.0.2zb	28.01.2022
CVE-2022-0778	-	5.0	3.0.0 – 3.0.1 1.1.1 -1.1.1m, 1.0.2 – 1.0.2zc	15.03.2022

Výše uvedená tabulka (Tab. 36) vychází z následujících zdrojů [176][177]. Z tabulky lze vyvodit, že se vývojáři o tuto knihovnu starají a pravidelně aktualizují nahlášené zranitelnosti. Datum nahlášení nelze dohledat, jelikož bezpečnostní chyby jsou hlášeny na jejich emailovou adresu [178]. Tím se jeví tato knihovna jako bezpečná, jednak že je udržovaná a také zveřejnění bezpečnostních chyb je provedené až po jejich opravě.

5.4.2 Dokumentace knihovny OpenSSL

Dokumentace knihovny OpenSSL je rozdělená do několika sekcí. V sekci „Commands“ se nachází seznam příkazu, které jdou použít v terminálu při používání knihovny. Každý příkaz má popis funkčnosti. V další sekci s názvem „Libraries“ je seznam funkcí, které tato knihovna nabízí. Funkce obsahují popis funkčnosti a parametrů a obsahují i ukázkové kódy. V sekci „File Formats“ se nachází seznam konfiguračních souborů, ve kterých je podrobný popis s návodem pro jejich nastavení. A v poslední sekci „Overviews“ je seznam funkcí jednotlivých kryptografických šifer a jejich metod. Ty disponují s popisem funkčnosti s ukázkovým kódem. Tato dokumentace se zdá být nepřehledná, jelikož nedisponuje rozdělením funkcí pro jednotlivé kryptografické operace či nějakým návodem pro implementaci. Zdá se být spíše pro vývojáře, kteří mají již s implementaci kryptografických algoritmů zkušenosti [179].

5.5 Open Quantum Safe

Tabulka 37 Srovnání obecných informací knihoven OQS

Knihovna	**Liboqs**	**OQS-OpenSSL**	**Liboqs-dotnet**	**Liboqs-python**
Licence	MIT	Apache	MIT	MIT
Programovací jazyk	C	C	C#	Python
Verze	0.7.1	1.1.1	0.3.0	0.7.1
Poslední update	16.12.2021	06.01.2022	08.07.2020	05.01.2022
Hodnocení na GitHub	951	198	16	42

Výše uvedená tabulka (Tab. 37) vychází z následujících zdrojů [182][183][184][185]. Lze si z tabulky všimnout, že udržované knihovny jsou Liboqs, OQS-OpenSSL a Liboqs-python. Knihovna Liboqs-dotnet byla naposledy aktualizována více než před rokem. Nejpopulárnější je mateřská knihovna, ze kterých implementují ty ostatní. Tato nízká popularita a nepříliš častá aktualizace může být zapříčiněna tím, že post-kvantové algoritmy nejsou standardizované a není je v nynější době potřeba implementovat, avšak to se může nástupem kvantových počítačů změnit.

Tabulka 38 Porovnání algoritmů pro knihovnu Liboqs

Knihovna	**Liboqs**
Asymetrické šifry	BIKE, Classic McEliece, FrodoKEM, HQC, Kyber, NTRU, NTRU-Prime, SABER, SIKE
Digitální podpis	CRYSTALS-Dilithium, Falcon, Rainbow, SPHINCS+-Haraka, SPHINCS+-SHA256, SPHINCS+-SHAKE256

Výše uvedená tabulka (Tab. 38) vychází z následujícího zdroje [182]. Tato knihovna podporuje jak finalisty, tak i alternativní finalisty třetího kola pro výběr standardizovaných algoritmů od společnosti NIST, které byly vypsány v kapitole 2.4.

Tabulka 39 Porovnání algoritmů pro knihovnu OQS-OpenSSL

Knihovna	**OQS-OpenSSL**
Protokol pro výměnu klíčů	BIKE, CRYSTAL-Kyber, FrodoKEM, HQC, NTRU, NTRU-Prime, SABER, SIDH, SIKE
Autentizace	CRYSTALS-Dilithium, Falcon, Picnic, Rainbow, SPHINCS-Haraka, SPHINCS-SHA256, SPHINCS-SHAKE256

Výše uvedená tabulka (Tab. 38) vychází z následujícího zdroje [183]. Knihovna OQS-OpenSSL pro implementování prototypních post-kvantových algoritmů pro protokol TLS 1.3 obsahuje ty, které jsou popsány v kapitole 2.4.

Knihovny Liboqs-dotnet a Liboqs-python disponují stejnými algoritmy, které jsou popsány v tabulce (Tab. 38), proto nadále nebudou zhotoveny tabulky.

5.5.1 Bezpečnost projektu OQS

Zatím nejsou známe žádné zranitelnosti algoritmů používané v OQS projektu, avšak je doporučeno vyčkat s jejích implementací do doby dalších pokynů od normalizačních skupin jako je například společnost NIST. Tento projekt je určen pouze pro napomáhání ve výzkumu post-kvantových algoritmů. Pokud by chtěl nějaký vývojář implementovat některý post-kvantový algoritmus, tak je doporučeno použít hybridní kryptografii a využívat post-

kvantové algoritmy s tradičními asymetrickými šiframi, jako jsou například RSA anebo některé eliptické křivky [186].

5.5.2 Dokumentace knihoven OQS

Dokumentace Liboqs disponuje návodem k instalaci na různé operační systémy se seznamem podporovaných algoritmů, kde každý algoritmus obsahuje popis parametrů. Dokumentace této knihovny obsahuje pouze dva ukázkové kódy, a to pro asymetrickou kryptografii a digitální podpis. Nadále je tam seznam hlavičkových souboru, ve kterých jsou popsány jejich funkce s parametry, ale nedisponují ukázkovými kódy. Tato Dokumentace je přehledná, avšak zde chybí více ukázkových kódu. [187].

OQS-OpenSSL má dokumentaci na stránce GitHub, obsahuje seznam podporovaných algoritmu s návodem na instalování. Disponuje s návodem na spuštění a generování certifikátů s následným otestování certifikátů. Tato dokumentace je přehledná, ale je velmi zjednodušeně zpracována [183].

Dokumentace knihoven Liboqs-dotnet a Liboqs-python jsou rovněž na stránce GitHub. Obsahují pouze návod na instalaci a implementaci knihoven s následným otestováním. Ale nedisponují s ukázkovými kódy a vysvětlením funkcí. To je možná zapříčiněno tím, že celý projekt OQS je zaměřený hlavně na podpoření výzkumu post-kvantových algoritmů [184][185].

6 ZHODNOCENÍ POROVNÁNÍ

V praktické části byly porovnány vývojové technologie na základě kritérií, stanovené na konci teoretické části. Pro výběr těchto technologií byly vybrány 3 nejpopulárnější programovací jazyky, určené pro vývoj moderních webových aplikací, na základě dotazníku zhotoveným webovou stránku Stackoverflow. Následně byly z této webové stránky vybrány jejich populární back-end frameworky či runtime prostředí. Frameworku Nest.js byl vybrán na základě počtu stažení. K těmto technologiím byly sepsány jejich knihovny, či populární knihovny třetí strany, které jsou určený ke kryptografickým účelům anebo pro zabezpečenou komunikaci. V těchto knihovnách byl srovnán výskyt algoritmů, které byly doporučeny v teoretické části. Bylo zjištěno, že tyto technologie a jejich knihovny disponují open-source licencí, což je velká výhoda, jelikož jsou zdrojové kódy těchto technologií volně dostupné na internetu. Proto mohou být kontrolovaný početnou komunitou, ve které se mohou nacházet odborníci na bezpečnost, kteří můžou případnou bezpečnostní chybu odhalit a následně jí nahlásit na příslušnou podporu.

Prvním vybraným programovacím jazykem byl JavaScript, s jeho populárním runtime prostředí Node.js. Bylo zjištěno, že je tohle runtime prostředí na trhu několik let a těší se vysoké popularitě, z čehož se zdá usoudit, že bude ještě dlouhodobě podporován. Ze zranitelností, které byly nalezeny, se nejeví příliš náchylný na mnoho bezpečnostních chyb. Zranitelnosti z roku 2022 byly porovnány a bylo zjištěno, že jejich bezpečnostní aktualizace byla vydaná skoro za měsíc. Disponuje s „bounty“ programem na nahlašování chyb, což je výhoda, díky které finančně motivují komunitu, k hledání bezpečnostních chyb. Výraznou nevýhodou je, že se tyto chyby nahlašují na veřejném fóru, což by mohli zneužít případní útočníci. Nepřehledná a nepříliš přívětivá dokumentace je nevýhodou toho runtime prostředí pro nové vývojáře. Node.js disponuje svými kryptografickými moduly Crypto a Web Crypto API. Výhoda je, že jsou součásti Node.js a tak mohou být kontrolované vývojáři tohoto projektu. Další podstatnou výhodou Crypto modulu je jeho implementace, standardizované společností NIST, knihovny OpenSSL. Algoritmy jsou vytvořené odborníky v oblasti kryptografie, z čehož se dá usoudit, že jsou bezpečné. Nevýhodou se tak ale stává, že v případě objevení bezpečnostní zranitelnosti knihovny OpenSSL, by byl zasažen i modul Crypto. V porovnávání bylo také zjištěno, že tento modul disponuje některými doporučenými algoritmy, pro základní kryptografické operace. Web Crypto API modul implementuje knihovnu Web Crypto API, která však není standardizovaná společností NIST. U tohoto module je stejná nevýhoda, v případě implementování jiné knihovny, tak jako u modulu Crypto. Také bylo

zjištěno, že tento modul obsahuje některé z doporučených algoritmů, avšak PKI certifikátem X.509 nedisponuje. Další výraznou nevýhodou je, že je tento modul experimentální, čímž se nejeví bezpečně.

Modul Crypto také slouží pro vlastní implementaci protokolu TLS, avšak to může být nebezpečné, proto je vhodnější použít vestavěný modul TLS s kombinací modulu HTTPS, pro zajištění zabezpečené komunikace. U modulu TLS byla zjištěna podpora nejnovější verze protokolu TLS 1.3, který obsahuje šifrovací sady s doporučenými algoritmy. Co se týče dokumentací, tak jsou součástí Node.js, která není přívětivá pro nové vývojáře. Dokumentace modulů Web Crypto API a HTTPS se zdají být neúplné, avšak i přes tyto negativní stránky, není problém dohledat na internetu řešení implementace, jelikož se jedná o velice populární technologii se silnou komunitou.

Pro potřeby implementování JWT byly vybrané moduly Node-jsonwebtoken, Jose a Aws-jwt-verify, jejich nevýhoda spočívá v tom, že nejsou součástí Node.js a tak není záruka, že jsou implementované bezpečně. Avšak v případě modulu Jose, kde je jeho počet stažení za týden vysoký, tak se jeví jakožto bezpečný. Také nabízí nejvíce doporučených algoritmů s nejlépe vypracovanou dokumentací. Modul Node-jsonwebtoken je sice nejlépe hodnocený na stránce GitHub, ale jeho poslední aktualizace byla před rokem. Z čehož nelze usoudit, jestli je tento modul natolik bezpečný, že nebylo třeba ho aktualizovat anebo ho vývojáři zanedbávají. Modul Aws-jwt-verify měl nedávnou aktualizaci, ale nedisponuje velkou popularitou a také neobsahuje doporučené algoritmy.

Pro srovnávání byly vybrané back-end Node.js frameworky Express.js a Nest.js. Kde Express.js je na trhu déle a dominuje se svou popularitou, jak v hodnocení na stránce GitHub, v dotazníku od StrackOverflow tak i v počtu stažení za týden. Jeho zranitelnosti ve verzi 4.x byly opraveny během měsíce, ale jedná se o bezpečnostní chyby z roku 2017. Je možné, že tento framework je správně implementován, a tak již nemá vlastní zranitelnosti. U frameworku Nest.js nebyly nalezeny žádné zranitelnosti, čímž se jeví jako bezpečný. Je zapotřebí kontrolovat bezpečnostní zranitelnosti Node.js, protože jsou tyto frameworky na tomto runtime prostředí stavěny. Výhodou je, že mohou využívat jeho moduly, včetně těch kryptografických, protože nedisponují vlastní implementací. U frameworku Nest.js, lze použít pro účely JWT modul Nest/JWT, ten však implementuje nedoporučený Node-jsonwebtoken. Dokumentaci mají oba frameworky kvalitně zpracované, ale Nest.js jí má přívětivější pro nové vývojáře. Z důvodu vysoké popularity je doporučeno použít framework Express.js.

Z těchto poznatků je doporučeno používat Node.js a jeho kryptografický modul Crypto, pro implementování některých kryptografických operací. Avšak pro realizaci zabezpečené komunikace je bezpečnější sáhnout po modulu TLS s kombinaci HTTPS. Co se týče využití JWT pro Node.js, je jednoznačný favorit modul Jose. K ulehčení vývoje pomocí Node.js back-end frameworků, se doporučení více přiklání k Express.js, ale Nest.js je také vyhovující volba.

Druhým vybraným populárním programovacím jazykem byl zvolen Python. Ten disponuje vestavěnými knihovnami Hashlib a Ssl. Haslib je pouze určený pro hash HMAC a KDF funkce, kde pro tyto účely disponuje doporučenými algoritmy. Knihovna Ssl je určená pro použití protokolu TLS, kde je možno využít nové TLS verze 1.3. Tyto dvě knihovny mají přehlednou dokumentaci, pro jejich bezpečnou implementaci. U těchto dvou knihoven je šance že mohou být navrženy správně a bezpečně, jelikož jsou součástí jazyka Python.

Pro ostatní kryptografické operace je potřeba použít knihovnu třetí strany. Byly vybrány tři populární knihovny PyCryptodome, Cryptography a PyNaCl. Tyto knihovny mají vysokou popularitu, kde Cryptography jí má nejvyšší. Díky této popularitě budou tyto knihovny ještě nějakou dobu udržované a také se jeví jako bezpečné. Cryptography implementuje společností NIST, standardizovanou kryptografickou knihovnu OpenSSL a PyNaCl implementuje knihovnu Libsodium. Výhodou se stává, že algoritmy mohou být správně implementované, ale na druhou stranu v případě zranitelnosti knihovny, kterou implementují, to ovlivní i je samotné. Knihovna PyCryptodome disponuje všemi algoritmy pro základní kryptografické operace, avšak neobsahuje žádný z doporučených algoritmů pro výměnu klíčů a ani X.509. PyNaCl taktéž obsahuje všechny algoritmy pro kryptografické operace, ale nedisponuje doporučenými algoritmy pro AEAD. PyNaCl má vysokoúrovňovou kryptografii, což je výhoda pro vývojáře, kteří nemají zkušenosti s implementaci kryptografických algoritmů. Knihovna Cryptography disponuje jak s nízkoúrovňovou, tak i vysokoúrovňovou kryptografii, které mají doporučené algoritmy. Tyto knihovny mají vhodné dokumentace pro pochopení a správnou implementaci kryptografických algoritmů.

Pro účely JWT byly vybrány knihovny PyJWT, Python-jose a Authlib. Bylo zjištěno že knihovna PyJWT je nejpopulárnější v počtu stažení za týden a v hodnocení na GitHubu, ale její poslední aktualizace proběhla před půlrokem. Python-jose je taktéž populární, ale jeho aktualizace proběhla skoro před rokem, čímž se nejeví příliš bezpečně. Poslední knihovnou pro JWT je Authlib, která je také populární a její aktualizace vyšla před měsícem. Z jejich popularity lze vyvodit, že se jedná o kvalitní a bezpečné zpracování. Co se týče doporučených

algoritmů, tak Authlib disponuje všemi, pro bezpečnou implementaci JWT. PyJWT pouze neobsahuje eliptickou křivkou pro výměnu klíčů, ale pro tyto účely lze použít RSA. A Python-jose nemá žádnou z doporučených eliptických křivek a její dokumentace je neúplná, kvůli čemuž není vhodná pro správnou implementaci JWT. Knihovny PyJWT a Authlib mají přehlednou a vhodnou dokumentaci pro vývojáře, kteří by je chtěli použít.

Pro porovnání back-end frameworku jazyka Python byly zvoleny Flask a Django. Flask je již několik let na trhu a těší se velké popularitě v počtu stažení za týden a také v hodnocení na stránce GtiHub. Framework Django je taktéž populární v počtu stažení a hodnocení. Nalezené zranitelnosti u frameworku Flask byly pouze dvě, to může být tím, že je mikro-framework. Což je jeho výhoda, jelikož si sami vývojáři mohou navolit balíčky, které potřebují. Zranitelnosti u Djanga verzí 3.2.x a 4.0.x nebylo možné zjistit, za jakou časovou dobu byly opraveny, jelikož v případě nalezení bezpečnostní chyby se hlásí na e-mail podpory a jsou zveřejněné až při vydané aktualizaci. Což je velkou výhodou, jelikož tuto zranitelnost nemohou útočníci využít. Flask nedisponuje žádnou kryptografickou knihovnou, ale lze využít knihoven třetích stran anebo ty, které jsou implementované v jazyce Python. Django disponuje pouze knihovnou Signer, které implementuje knihovnu Hashlib. A pro ostatní kryptografické operace je třeba využít knihovny třetích stran. Oba frameworky disponují přehlednou a vhodnou dokumentaci pro správné použití.

Z těchto poznatků se Cryptography stává nejlepší volbou při výběru kryptografické knihovny. Pro implementaci pouze hash HMAC anebo KDF je doporučeno použít knihovnu Hashlib a pro implementaci protokolu TLS knihovnu Ssl. V případě implementace JWT je doporučeno využít knihovny Authlib. Co se týče back-end frameworků, tak je doporučen jak Flask, tak i Django s jeho knihovnou Signer, určenou pro hash, HMAC a KDF.

Posledním vybraným programovacím jazykem je C#. Ten disponuje balíčkem knihoven .NET, který obsahuje kryptografickou knihovnou Cryptography. Tato knihovna má některé z doporučených algoritmu, avšak tento výběr je malý. Co se týče KDF, tak nedisponuje žádným doporučeným algoritmem. Proto byla porovnána knihovna třetí strany Norgerman.Cryptography.Scrypt, která disponuje pouze doporučeným algoritmem Scrypt. Ta ale není populární, takže se nejeví příliš bezpečně. Také nemá žádnou dokumentaci, pouze na stránce GitHub je ukázaný jeden řádek kódu. Nevýhodou knihovny Cryptography je v implementaci na jiných platformách, než je Windows, jelikož tato knihovna využívá algoritmů, které poskytuje daný operační systém. V případě některých algoritmů je třeba na jiných

operačních systémech nainstalovat knihovnu OpenSSL. Pro správnou implementaci kryptografických algoritmů je její dokumentace vhodná.

Pro implementace JWT byly vybrány knihovny třetí strany Jwt a Jose-jwt, kde jejích popularita není příliš vysoká. U knihovny Jwt poslední aktualizace proběhla před rokem, čímž se nejeví příliš bezpečně. Disponuje některými doporučenými algoritmy, avšak neobsahuje žádnou z doporučených eliptických křivek. Její dokumentace se nezdá být vhodná pro správnou implementací JWT. Knihovna Jose-jwt byla aktualizovaná v nedávné době a pro implementaci JWT disponuje doporučenými algoritmy, avšak má pouze jednu eliptickou křivku. Její dokumentace se zdá být nejvhodnější z JWT knihoven jazyka C#, pro správnou a bezpečnou implementaci.

Pro jazyk C# byl porovnán framework ASP .NET Core, který je populární v dotazníku od Stackoverflow a v počtu hodnocení na stránce GitHub. Výhodou tohoto frameworku je používání protokolu HTTPS ve výchozím nastavení, a tak není potřeba ho implementovat. Jeho zranitelnosti byly porovnány v letech 2020 a 2021, ale rychlost opravy nebyl nalezen, jelikož disponuje s dotazníkem pro nahlašovaní chyb, který není zveřejněn. Jeho dokumentace je přehledná a jeví se jako vhodná pro správnou a bezpečnou implementaci.

Z těchto poznatků je doporučeno použít back-end framework ASP .NET Core, avšak kryptografická knihovna, z balíčků .NET, Cryptorgraphy se nezdá být vhodná, jelikož disponuje malou pestrostí pro výběr doporučených algoritmů, a hlavně neobsahuje doporučené KDF. Pro implementaci JWT také není příliš doporučeno použít některou ze srovnávaných knihoven, kvůli nízké popularitě, avšak v případě potřeby je nejvhodnější použít Jose-jwt.

Nadále byla také srovnána knihovna OpenSSL, jelikož jí implementují srovnávané knihovny Crypto, TLS, Cryptography a Ssl. Tato knihovna je velice populární, a hlavně standardizovaná společnosti NIST. Při srovnávání její zranitelnosti nebylo zjištěno časové období opravy, jelikož taktéž disponuje neveřejným nahlašovaní bezpečnostních chyb. Její dokumentace je určená spíše pro zkušené vývojáře v oblasti kryptografie. Z těchto zjištěných dat je doporučeno využívat tuto knihovnu, či jiných knihoven, které jí implementují.

Poslední srovnání bylo projektu Open Quantum Safe, kde byly porovnány jeho implementace do programovacích jazyků, které jsou obsaženy v této práci. Bylo zjištěno, že pro JavaScript anebo Node.js není vytvořená implementace. Tyto knihovny disponují algoritmy, které jsou ve fázi standardizace společnostní NIST, z čeho lze usoudit, že se jejich vývojáři snaží, aby tyto knihovny byly aktuální. Pro tyto knihovny neexistují žádné zranitelnosti, ale

není doporučeno je implementovat, jelikož tyto knihovny zatím slouží pro napomáhání ve výzkumu post-kvantových algoritmů. Dokumentace těchto knihoven není zpracovaná dopodrobna, ale to muže být zapříčiněno s jejími zaměřením na výzkum. Z těchto poznatků není doporučeno implementovat knihovny OQS projektu do reálného provozu, jelikož jeho algoritmy ještě nejsou standardizované.

Tabulka 40 Seznam doporučených technologií

Jazyk/Runtime prostředí	**JavaScript (Node.js)**	**Python**	**C#**	**C**
Framework	Express.js, Nest.js	Flask, Django	ASP .NET Core	-
Kryptografická knihovna/kryptografický modul	Crypto	Cryptography, Hashlib, Signer	-	OpenSSL
JWT	Jose	Authlib	Jose-jwt	-
TLS/HTTPS	TLS, HTTPS	Ssl	-	OpenSSL
Post-kvantová knihovna (pouze pro účely výzkumu)	-	Liboqs-dotnet	Liboqs-python	Liboqs, OQS-OpenSSL

ZÁVĚR

Cílem této bakalářské práce bylo sestavit seznam doporučených technologií s ohledem na bezpečnost, který pomůže budoucím vývojářům webových aplikací ve výběru.

Na začátku teoretické části byly představeny základní pojmy a doporučení spjaté s vývojem webových aplikací. Dále byly popsány vlastnosti kryptografických algoritmů a jejich operací. K těmto operacím byly popsané doporučené kryptografické algoritmy a šifrovací schémata.

Dále byla představená post-kvantová kryptografie s jejími algoritmy, které prochází standardizací. Následně byl sepsán seznam technologií, určených pro vývoj webových aplikací s důrazem na bezpečnost. Tyto technologie byly rozřazené dle programovacích jazyků kontextu webového vývoje. Na závěr teoretické části byla stanovena kritéria jako podklad pro další srovnání.

V praktické části byly vybrané technologie tabulkově srovnány na základě poznatků teoretické části. Na konci praktické části jsou shrnuty výhody a nevýhody. Výstupem práce je seznam doporučených technologií k vytvoření zabezpečené webové aplikace a usnadnění volby knihoven při použití kryptografických algoritmů. Tento seznam je zhotoven v tabulce (Tab. 40).

V budoucnu by bylo vhodné analyzovat výkon algoritmu, pro jednotlivé kryptografické operace, se závislosti na množství dat a velikosti klíčů. A následně z naměřených dat, zhotovit tabulkové a grafické porovnání. Nebo se podrobněji zaměřit na post-kvantové algoritmy.

SEZNAM POUŽITÉ LITERATURY

[1] Website vs. Web Application (App): What's the Difference?. *Indeed* [online]. 23.08.2021 [cit. 2022-03-30]. Dostupné z: https://www.indeed.com/career-advice/career-development/website-vs-web-application

[2] Difference Between Web application and Website. *Geeksforgeeks* [online]. 27.01.2022 [cit. 2022-03-30]. Dostupné z: https://www.geeksforgeeks.org/difference-between-web-application-and-website/

[3] Web pages and web apps. *Bbc* [online]. 2022 [cit. 2022-03-30]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/znkqn39/revision/3

[4] 2022 Stats on Top JS Frameworks: React, Vue, Svelte & Angular. *Tecla* [online]. 22.02.2022 [cit. 2022-03-30]. Dostupné z: https://www.tecla.io/blog/top-js-frameworks/

[5] Frontend vs Backend. *Geeksforgeek* [online]. 31.08.2021 [cit. 2022-03-30]. Dostupné z: https://www.geeksforgeeks.org/frontend-vs-backend/

[6] What's the Difference Between Frontend and Backend Web Development?. Careerfoundry [online]. 27.12.2021 [cit. 2022-03-30]. Dostupné z: https://www.careerfoundry.com/en/blog/web-development/whats-the-difference-between-frontend-and-backend/

[7] Cryptography. Techtarget [online]. 2022 [cit. 2022-03-30]. Dostupné z: https://www.techtarget.com/searchsecurity/definition/cryptography

[8] SEGALA, Alessandro. *Essential Cryptography for JavaScript Developers: A practical guide to leveraging common cryptographic operations in Node.js and the browser*. Birmingham: Packt Publishing, 2022. ISBN 978-1-80107-533-6.

[9] Co je DevOps?. *Azure* [online]. 2022 [cit. 2022-03-31]. Dostupné z: https://www.azure.microsoft.com/cs-cz/overview/what-is-devops/

[10] DevSecOps Overview. *Snyk* [online]. 2022 [cit. 2022-03-31]. Dostupné z: https://www.snyk.io/series/devsecops/

[11] NIST. *Special Publication 800-57: Recommendation for Key Management* [online]. NIST, 2016 [cit. 2022-04-05]. Dostupné z: http://dx.doi.org/10.6028/NIST.SP.800-57pt1r4

[12] Symmetric Key Encryption: why, where and how it's used in banking. *Cryptomathic* [online]. 18.01.2019 [cit. 2022-04-05]. Dostupné z: https://www.cryptomathic.com/news-events/blog/symmetric-key-encryption-why-where-and-how-its-used-in-banking

[13] Cryptographic Key Management Concepts: on Key Generation, Metadata, Life-cycles, Compromise and more. *Cryptomathic* [online]. 20.05.2019 [cit. 2022-04-06]. Dostupné z: https://www.cryptomathic.com/news-events/blog/cryptographic-key-management-concepts-on-key-generation-metadata-life-cycles-compromise-and-more

[14] Asymmetric cryptography (public key cryptography). *Techtarget* [online]. 2022 [cit. 2022-04-06]. Dostupné z: https://www.techtarget.com/searchsecurity/definition/asymmetric-cryptography

[15] NAKOV, Svetlin. *Practical Cryptography for Developers* [online]. Sofia: SoftUni Foundation, 2018 [cit. 2022-04-07]. ISBN 978-619-00-0870-5. Dostupné z: https://cryptobook.nakov.com/

[16] NIST. *Special Publication 800-175B: Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms* [online]. NIST, 2020 [cit. 2022-04-09]. Dostupné z: https://doi.org/10.6028/NIST.SP.800-175Br1

[17] BOCK, Lisa. *Modern Cryptography for Cybersecurity Professionals: Learn how you can leverage encryption to better secure your organization's data*. Birmingham: Packt Publishing, 2021. ISBN 978-1-83864-435-2.

[18] DE CANNIÈRE, Christophe a , ed. Triple DES. VAN TILBORG, Henk C. A a Sushil JAJODIA. *Encyclopedia of Cryptography and Security* [online]. 2nd ed. Boston: Springer, 2011 [cit. 2022-04-09]. ISBN 978-1-4419-5906-5. Dostupné z: https://link.springer.com/referenceworkentry/10.1007/978-1-4419-5906-5_621

[19] Block Cipher vs Stream Cipher: What They Are & How They Work. *Thesslstore* [online]. 04.01.2021 [cit. 2022-04-13]. Dostupné z: https://www.thesslstore.com/blog/block-cipher-vs-stream-cipher/

[20] ChaCha20. In: *Cryptopp* [online]. 2021 [cit. 2022-04-13]. Dostupné z: https://www.cryptopp.com/wiki/ChaCha20

[21] AUMASSON, Jean-Philippe. *Serious cryptography: a practical introduction to modern encryption*. San Francisco: No Starch Press, 2018. ISBN 978-1-59327-826-7.

[22] 3 Reasons Why Open Source Software is More Secure than Commercial Software. *Whitesourcesoftware* [online]. 07.04.2021 [cit. 2022-04-18]. Dostupné z: https://www.whitesourcesoftware.com/resources/blog/3-reasons-why-open-source-is-safer-than-commercial-software/

[23] Why would someone use a framework?: The importance of frameworks, libraries, and tools in software development. *Levelup* [online]. 05.01.2021 [cit. 2022-04-18]. Dostupné z: https://www.levelup.gitconnected.com/why-would-someone-use-a-framework-bd4706e4464f

[24] What is JavaScript used for?. *Hackreactor* [online]. 26.08.2021 [cit. 2022-04-19]. Dostupné z: https://www.hackreactor.com/blog/what-is-javascript-used-for

[25] Developer Survey. In: *Stackoverflow* [online]. 2021 [cit. 2022-04-19]. Dostupné z: https://www.insights.stackoverflow.com/survey/2021

[26] What is Node.js?: Where, When & How To Use It. *Simform* [online]. 04.02.2022 [cit. 2022-04-22]. Dostupné z: https://www.simform.com/blog/what-is-node-js/

[27] Crypto documentation. In: *Nodejs* [online]. [cit. 2022-04-22]. Dostupné z: https://www.nodejs.org/api/crypto.html

[28] What is Express.js?. *Besanttechnologies* [online]. 2022 [cit. 2022-04-22]. Dostupné z: https://www.besanttechnologies.com/what-is-expressjs

[29] Nodejs/node. In: *Github* [online]. 2022 [cit. 2022-04-26]. Dostupné z: https://www.github.com/nodejs/node

[30] Downloads. In: *Nodejs* [online]. [cit. 2022-04-26]. Dostupné z: https://www.nodejs.org/en/download/

[31] Node.js. In: *Wikipedia* [online]. 15.05.2022 [cit. 2022-04-26]. Dostupné z: https://en.wikipedia.org/wiki/Node.js

[32] RICHARDLAU. 2022-03-17, Version 16.14.2 'Gallium' (LTS). In: *Github* [online]. 18.03.2022 [cit. 2022-04-26]. Dostupné z: https://github.com/nodejs/node/releases/tag/v16.14.2

[33] CVSS Scores For Nodejs Node.js Between 2009-05-27 and 2022-04-26. In: *CVE Details* [online]. 26.04.2022 [cit. 2022-04-26]. Dostupné z:

https://www.cvedetails.com/cvss-score-charts.php?fromform=1&vendor_id=&product_id=30764&startdate=2009-05-27&enddate=2022-04-26&groupbyyear=1

[34] January 10th 2022 Security Releases. In: *Nodejs* [online]. 11.01.2022 [cit. 2022-04-26]. Dostupné z: https://nodejs.org/en/blog/vulnerability/jan-2022-security-releases/

[35] BENGL. Node.js Certificate Verification Bypass via String Injection. In: *Hackerone* [online]. 17.12.2021 [cit. 2022-04-26]. Dostupné z: https://hackerone.com/reports/1429694

[36] RUGVIP. Prototype pollution via console.table properties. In: *Hackerone* [online]. 20.12.2021 [cit. 2022-04-26]. Dostupné z: https://hackerone.com/reports/1431042

[37] Security Vulnerabilities Published In 2022. In: *CVE Details* [online]. 2022 [cit. 2022-04-26]. Dostupné z: https://www.cvedetails.com/vulnerability-list/vendor_id-12113/product_id-30764/year-2022/Nodejs-Node.js.html

[38] TROTT. Security. In: *GitHub* [online]. 01.03.2022 [cit. 2022-04-26]. Dostupné z: https://github.com/nodejs/node/blob/HEAD/SECURITY.md#security

[39] Web Crypto API. In: *Nodejs* [online]. 2022 [cit. 2022-04-26]. Dostupné z: https://nodejs.org/dist/v18.0.0/docs/api/webcrypto.html

[40] Web Crypto API: W3C Editor's Draft. In: *W3c* [online]. 31.03.2022 [cit. 2022-04-26]. Dostupné z: https://w3c.github.io/webcrypto/

[41] Openssl/openssl. In: *GitHub* [online]. 2022 [cit. 2022-04-26]. Dostupné z: https://github.com/openssl/openssl

[42] Comparison of cryptography libraries. In: *Wikipedia* [online]. 11.04.2022 [cit. 2022-04-26]. Dostupné z: https://en.wikipedia.org/wiki/Comparison_of_cryptography_libraries

[43] How to use the crypto module. *Nodejs* [online]. 26.08.2011 [cit. 2022-04-28]. Dostupné z: https://nodejs.org/en/knowledge/cryptography/how-to-use-crypto-module/

[44] TLS (SSL). In: *Nodejs* [online]. 2022 [cit. 2022-04-29]. Dostupné z: https://nodejs.org/api/tls.html

[45] How-to Improve Node.js HTTPS Server Performance. *Strongloop* [online]. 07.08.2013 [cit. 2022-04-29]. Dostupné z: https://strongloop.com/strongblog/improve-the-performance-of-the-node-js-https-server/

[46] HTTPS. In: *Nodejs* [online]. 2022 [cit. 2022-04-29]. Dostupné z: https://nodejs.org/api/https.html

[47] Security/Server Side TLS. In: *Mozilla wiki* [online]. 01.04.2022 [cit. 2022-04-29]. Dostupné z: https://wiki.mozilla.org/Security/Server_Side_TLS

[48] Introduction to JSON Web Tokens. *Jwt* [online]. [cit. 2022-04-29]. Dostupné z: https://jwt.io/introduction

[49] Libraries for Token Signing/Verification. In: *Jwt* [online]. [cit. 2022-04-29]. Dostupné z: https://jwt.io/libraries

[50] Auth0/node-jsonwebtoken. In: *GitHub* [online]. 2022 [cit. 2022-04-29]. Dostupné z: https://github.com/auth0/node-jsonwebtoken

[51] Panva/jose. In: *GitHub* [online]. 2022 [cit. 2022-04-29]. Dostupné z: https://github.com/panva/jose

[52] Awslabs/aws-jwt-verify. In: *GitHub* [online]. 2022 [cit. 2022-04-29]. Dostupné z: https://github.com/awslabs/aws-jwt-verify

[53] Node-jsonwebtoken. In: *Npmjs* [online]. [cit. 2022-04-29]. Dostupné z: https://www.npmjs.com/package/node-jsonwebtoken

[54] Jose. In: *Npmjs* [online]. [cit. 2022-04-29]. Dostupné z: https://www.npmjs.com/package/jose

[55] Aws-jwt-verify. In: *Npmjs* [online]. [cit. 2022-04-29]. Dostupné z: https://www.npmjs.com/package/aws-jwt-verify

[56] Express. In: *Npmjs* [online]. [cit. 2022-04-30]. Dostupné z: https://www.npmjs.com/package/express

[57] Expressjs/express. In: *GitHub* [online]. 2022 [cit. 2022-04-30]. Dostupné z: https://github.com/expressjs/express

[58] DOUGWILSON. History.md. In: *GitHub* [online]. 2022 [cit. 2022-04-30]. Dostupné z: https://github.com/expressjs/express/blob/master/History.md

[59] NATIONAL VULNERABILITY DATABASE: CVE-2017-1000048 Detail. In: *NIST* [online]. [cit. 2022-04-30]. Dostupné z: https://nvd.nist.gov/vuln/detail/CVE-2017-1000048

[60] Regular Expression Denial of Service (ReDoS). In: *Snyk* [online]. 2020 [cit. 2022-04-30]. Dostupné z: https://security.snyk.io/vuln/npm:ms:20170412

[61] NATIONAL VULNERABILITY DATABASE: CVE-2017-16137 Detail. In: *NIST* [online]. [cit. 2022-04-30]. Dostupné z: https://nvd.nist.gov/vuln/detail/CVE-2017-16137

[62] NATIONAL VULNERABILITY DATABASE: CVE-2017-16119 Detail. In: *NIST* [online]. [cit. 2022-04-30]. Dostupné z: https://nvd.nist.gov/vuln/detail/CVE-2017-16119

[63] NATIONAL VULNERABILITY DATABASE: CVE-2017-16118 Detail. In: *NIST* [online]. [cit. 2022-04-30]. Dostupné z: https://nvd.nist.gov/vuln/detail/CVE-2017-16118

[64] NATIONAL VULNERABILITY DATABASE: CVE-2017-16138 Detail. In: *NIST* [online]. [cit. 2022-04-30]. Dostupné z: https://nvd.nist.gov/vuln/detail/CVE-2017-16138

[65] Security updates. In: *Expressjs* [online]. [cit. 2022-04-30]. Dostupné z: https://expressjs.com/en/advanced/security-updates.html

[66] @nestjs/core. In: *Npmjs* [online]. [cit. 2022-04-30]. Dostupné z: https://www.npmjs.com/package/@nestjs/core

[67] Nestjs/nest. In: GitHub [online]. 2022 [cit. 2022-04-30]. Dostupné z: https://github.com/nestjs/nest

[68] TNIESSEN. Napi_get_value_string_X allow various kinds of memory corruption. In: *Hackerone* [online]. 27.01.2020 [cit. 2022-04-30]. Dostupné z: https://hackerone.com/reports/784186

[69] Introduction. Nestjs [online]. 2022 [cit. 2022-04-30]. Dostupné z: https://docs.nestjs.com/

[70] @nestjs/common vulnerabilities. In: *Snyk* [online]. 2022 [cit. 2022-04-30]. Dostupné z: https://snyk.io/vuln/npm:%40nestjs%2Fcommon

[71] Encryption and Hashing. *Nestjs* [online]. 2022 [cit. 2022-04-30]. Dostupné z: https://docs.nestjs.com/security/encryption-and-hashing

[72] Nestjs/jwt. In: *GitHub* [online]. 2022 [cit. 2022-04-30]. Dostupné z: https://github.com/nestjs/jwt

[73] Why Choose Python As Backend Development?. *Micropyramid* [online]. 2021 [cit. 2022-05-02]. Dostupné z: https://micropyramid.com/blog/why-choose-python-as-backend-development/

[74] Introduction to Flask. *Pymbook* [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://pymbook.readthedocs.io/en/latest/flask.html

[75] Django introduction. *Mozilla* [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

[76] Pallets/flask. In: *GitHub* [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://github.com/pallets/flask

[77] Flask. In: *Pypistats* [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://pypistats.org/packages/flask

[78] Flask Design Patterns And Best Practices For Web Applications. *Softwaretestinghelp* [online]. 04.03.2022 [cit. 2022-05-02]. Dostupné z: https://www.softwaretestinghelp.com/flask-design-patterns-for-web-applications/

[79] Changes. In: *Flask* [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://flask.palletsprojects.com/en/2.1.x/changes/

[80] Django/django. In: *GitHub* [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://github.com/django/django

[81] Project description. In: *Pypi* [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://pypi.org/project/Django/#description

[82] How to get Django: Supported Versions. In: *Djangoproject* [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://www.djangoproject.com/download/

[83] Django 3.2.13 release notes. In: *Djangoproject* [online]. 11.04.2022 [cit. 2022-05-02]. Dostupné z: https://docs.djangoproject.com/en/3.2/releases/3.2.13/

[84] Django. In: *Pypistats* [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://pypistats.org/packages/django

[85] FAQ: General. In: *Djangoproject* [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://docs.djangoproject.com/en/4.0/faq/general/

[86] CVE-2018-1000656. In: *Vulmon* [online]. [cit. 2022-05-02]. Dostupné z: https://vulmon.com/vulnerabilitydetails?qid=CVE-2018-1000656&scoretype=cvssv2

[87] DAVIDISM. Detect UTF encodings when loading json. In: *GitHub* [online]. 10.04.2018 [cit. 2022-05-02]. Dostupné z: https://github.com/pallets/flask/pull/2691

[88] DAVIDISM. 0.12.3. In: *GitHub* [online]. 26.04.2018 [cit. 2022-05-02]. Dostupné z: https://github.com/pallets/flask/releases/tag/0.12.3

[89] CVE-2019-1010083. In: *Vulmon* [online]. [cit. 2022-05-02]. Dostupné z: https://vulmon.com/vulnerabilitydetails?qid=CVE-2019-1010083&scoretype=cvssv2

[90] CVE-2022-28346. In: *Vulmon* [online]. [cit. 2022-05-02]. Dostupné z: https://vulmon.com/vulnerabilitydetails?qid=CVE-2022-28346&scoretype=cvssv2

[91] CVE-2022-28347. In: *Vulmon* [online]. [cit. 2022-05-02]. Dostupné z: https://vulmon.com/vulnerabilitydetails?qid=CVE-2022-28347&scoretype=cvssv2

[92] CVE-2022-23833. In: *Vulmon* [online]. [cit. 2022-05-02]. Dostupné z: https://vulmon.com/vulnerabilitydetails?qid=CVE-2022-23833&scoretype=cvssv2

[93] CVE-2022-22818. In: *Vulmon* [online]. [cit. 2022-05-02]. Dostupné z: https://vulmon.com/vulnerabilitydetails?qid=CVE-2022-22818&scoretype=cvssv2

[94] CVE-2022-22818. In: *Vulmon* [online]. [cit. 2022-05-02]. Dostupné z: https://vulmon.com/vulnerabilitydetails?qid=CVE-2021-45452&scoretype=cvssv2

[95] CVE-2021-45115. In: *Vulmon* [online]. [cit. 2022-05-02]. Dostupné z: https://vulmon.com/vulnerabilitydetails?qid=CVE-2021-45115&scoretype=cvssv2

[96] CVE-2021-45116. In: *Vulmon* [online]. [cit. 2022-05-02]. Dostupné z: https://vulmon.com/vulnerabilitydetails?qid=CVE-2021-45116&scoretype=cvssv2

[97] Django security releases issued: 4.0.4, 3.2.13, and 2.2.28. In: *Djangoproject* [online]. 11.04.2022 [cit. 2022-05-02]. Dostupné z: https://www.djangoproject.com/weblog/2022/apr/11/security-releases/

[98] Django security releases issued: 4.0.2, 3.2.12, and 2.2.27. In: *Djangoproject* [online]. 01.02.2022 [cit. 2022-05-02]. Dostupné z: https://www.djangoproject.com/weblog/2022/feb/01/security-releases/

[99] Django security releases issued: 4.0.1, 3.2.11, and 2.2.26. In: *Djangoproject* [online]. 04.01.2022 [cit. 2022-05-02]. Dostupné z: https://www.djangoproject.com/weblog/2022/jan/04/security-releases/

[100] Reporting bugs and requesting features. In: *Djangoproject* [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://docs.djangoproject.com/en/dev/internals/contributing/bugs-and-features/

[101] CVSS Scores For Djangoproject Django Between 2005-07-21 and 2022-05-03. In: *Cvedetails* [online]. 03.05.2022 [cit. 2022-05-03]. Dostupné z: https://www.cvedetails.com/cvss-score-charts.php?fromform=1&vendor_id=&product_id=18211&startdate=2005-07-21&enddate=2022-05-03&groupbyyear=1

[102] 3 Best Python Encryption Libraries in 2022. *Tleapps* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://tleapps.com/best-python-encryption-libraries

[103] Legrandin/pycryptodome. In: *GitHub* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://github.com/Legrandin/pycryptodome

[104] Welcome to pyca/cryptography. In: *Cryptography* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://cryptography.io/en/latest/

[105] Changelog. In: *Pycryptodome* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://www.pycryptodome.org/en/latest/src/changelog.html#february-2022

[106] Pycryptodomex. In: *Pypistats* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://pypistats.org/packages/pycryptodomex

[107] Pyca/cryptography. In: *GitHub* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://github.com/pyca/cryptography

[108] Changelog. In: *Cryptography* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://cryptography.io/en/latest/changelog/

[109] Cryptography. In: *Pypistats* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://pypistats.org/packages/cryptography

[110] Pyca/pynacl. In: *GitHub* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://github.com/pyca/pynacl

[111] Changelog. In: *Pynacl* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://pynacl.readthedocs.io/en/latest/changelog/

[112] Pynacl. In: *Pypistats* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://pypistats.org/packages/pynacl

[113] Crypto.Hash package. In: *Pycryptodome* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://pycryptodome.readthedocs.io/en/latest/src/hash/hash.html

[114] Future plans. In: *Pycryptodome* [online]. 2022 [cit. 2022-05-04]. Dostupné z: https://pycryptodome.readthedocs.io/en/latest/src/future.html

[115] Nacl.pwhash. In: *Pynacl* [online]. 2022 [cit. 2022-05-05]. Dostupné z: https://pynacl.readthedocs.io/en/latest/api/pwhash/

[116] Nacl.hash. In: *Pynacl* [online]. 2022 [cit. 2022-05-05]. Dostupné z: https://pynacl.readthedocs.io/en/latest/api/hash/

[117] Digital Signatures. In: *Pynacl* [online]. 2022 [cit. 2022-05-05]. Dostupné z: https://pynacl.readthedocs.io/en/latest/signing/

[118] Public Key Encryption. In: *Pynacl* [online]. 2022 [cit. 2022-05-05]. Dostupné z: https://pynacl.readthedocs.io/en/latest/signing/

[119] Secure hashes and message digests. In: *Python* [online]. 2022 [cit. 2022-05-05]. Dostupné z: https://docs.python.org/3/library/hashlib.html

[120] TLS/SSL wrapper for socket objects. In: *Python* [online]. 2022 [cit. 2022-05-05]. Dostupné z: https://docs.python.org/3/library/ssl.html

[121] Jpadilla/pyjwt. In: *GitHub* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://github.com/jpadilla/pyjwt

[122] Mpdavis/python-jose. In: *GitHub* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://github.com/mpdavis/python-jose

[123] Lepture/authlib. In: *GitHub* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://github.com/lepture/authlib

[124] Pyjwt. In: *Pypistats* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://pypistats.org/packages/pyjwt

[125] Python-jose. In: *Pypistats* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://pypistats.org/packages/python-jose

[126] Authlib. In: *Pypistats* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://pypistats.org/packages/authlib

[127] Digital Signature Algorithms. In: *Pyjwt* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://pyjwt.readthedocs.io/en/stable/algorithms.html

[128] RFC8037: CFRG Elliptic Curve Diffie-Hellman (ECDH) and Signatures in JSON Object Signing and Encryption (JOSE). In: *Authlib* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://docs.authlib.org/en/latest/specs/rfc8037.html?highlight=EdDSA

[129] Node.js v18.1.0 documentation. In: *Nodejs* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://nodejs.org/dist/latest-v18.x/docs/api/

[130] 4.x API. *Expressjs* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://expressjs.com/en/4x/api.html

[131] Welcome to PyCryptodome's documentation. *Pycryptodome* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://pycryptodome.readthedocs.io/en/latest/

[132] PyNaCl: Python binding to the libsodium library. *Pynacl* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://pynacl.readthedocs.io/en/latest/

[133] Welcome to PyJWT. *Pyjwt* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://pyjwt.readthedocs.io/en/stable/

[134] Python-jose. *Python-jose* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://python-jose.readthedocs.io/en/latest/

[135] Authlib: Python Authentication. *Authlib* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://docs.authlib.org/en/latest/

[136] Cryptographic signing. *Djangoproject* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://docs.djangoproject.com/en/4.0/topics/signing/

[137] User's Guide. *Flask* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://flask.palletsprojects.com/en/2.1.x/

[138] Django documentation. *Djangoproject* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://docs.djangoproject.com/en/4.0/

[139] A tour of the C# language. *Microsoft* [online]. 2022 [cit. 2022-05-08]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[140] Overview to ASP.NET Core. In: *Microsoft* [online]. 2022 [cit. 2022-05-08]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0

[141] .NET cryptography model. *Microsoft* [online]. 2022 [cit. 2022-05-08]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/standard/security/cryptography-model

[142] Cross-Platform Cryptography in .NET Core and .NET 5. In: *Microsoft* [online]. 2022 [cit. 2022-05-08]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/standard/security/cross-platform-cryptography

[143] Hosting ASP.NET Core images with Docker over HTTPS. *Microsoft* [online]. 2022 [cit. 2022-05-08]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/security/docker-https?view=aspnetcore-6.0

[144] .NET is open source. In: *Microsoft* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://dotnet.microsoft.com/en-us/platform/open-source

[145] .NET and .NET Core Support Policy. In: *Microsoft* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://dotnet.microsoft.com/en-us/platform/support/policy/dotnet-core

[146] Norgerman/Scrypt. In: *GitHub* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://github.com/Norgerman/Scrypt

[147] Norgerman.Cryptography.Scrypt. In: *Nuget* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://www.nuget.org/packages/Norgerman.Cryptography.Scrypt/

[148] System.Security.Cryptography Namespace. In: *Microsoft* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography?view=net-6.0

[149] JWT. In: *Nuget* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://www.nuget.org/packages/JWT

[150] Jwt-dotnet/jwt. In: *GitHub* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://github.com/jwt-dotnet/jwt

[151] Jose-jwt. In: *Nuget* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://www.nuget.org/packages/jose-jwt/

[152] Dvsekhvalnov/jose-jwt. In: *GitHub* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://github.com/dvsekhvalnov/jose-jwt

[153] Dotnet/aspnetcore. In: *GitHub* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://github.com/dotnet/aspnetcore

[154] ASP.NET Core. In: *Wikipedia* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://en.wikipedia.org/wiki/ASP.NET_Core

[155] CVSS Scores For Microsoft Asp.net Core Between 2016-06-27 and 2022-05-08. In: *Cve Details* [online]. 08.05.2022 [cit. 2022-05-09]. Dostupné z: https://www.cvedetails.com/cvss-score-charts.php?fromform=1&vendor_id=&product_id=42998&startdate=2016-06-27&enddate=2022-05-08&groupbyyear=1

[156] Asp.net Core: Security Vulnerabilities Published In 2020. In: *Cve Details* [online]. 2020 [cit. 2022-05-09]. Dostupné z: https://www.cvedetails.com/vulnerability-list/vendor_id-26/product_id-42998/year-2020/Microsoft-Asp.net-Core.html

[157] Asp.net Core: Security Vulnerabilities Published In 2021. In: *Cve Details* [online]. 2021 [cit. 2022-05-09]. Dostupné z: https://www.cvedetails.com/vulnerability-list/vendor_id-26/product_id-42998/year-2021/Microsoft-Asp.net-Core.html

[158] RBHANDA. Microsoft Security Advisory CVE-2020-1597 | ASP.NET Core Denial of Service Vulnerability. In: *GitHub* [online]. 11.08.2020 [cit. 2022-05-09]. Dostupné z: https://github.com/dotnet/announcements/issues/162

[159] Download .NET Core 3.1. In: *Microsoft* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://dotnet.microsoft.com/en-us/download/dotnet/3.1

[160] BLOWDART. Microsoft Security Advisory CVE-2020-1161 | ASP.NET Core Denial of Service Vulnerability. In: *GitHub* [online]. 13.05.2020 [cit. 2022-05-09]. Dostupné z: https://github.com/aspnet/Announcements/issues/416

[161] RBHANDA. Microsoft Security Advisory CVE-2020-1045 | Microsoft ASP.NET Core Security Feature Bypass Vulnerability. In: *GitHub* [online]. 08.09.2020 [cit. 2022-05-09]. Dostupné z: https://github.com/dotnet/announcements/issues/165

[162] BLOWDART. Microsoft Security Advisory CVE-2020-0603 : ASP.NET Core Remote Code Execution Vulnerability. In: *GitHub* [online]. 14.01.2020 [cit. 2022-05-09]. Dostupné z: https://github.com/dotnet/aspnetcore/issues/18337

[163] BLOWDART. Microsoft Security Advisory CVE-2020-0602 : ASP.NET Core Denial of Service Vulnerability. In: *GitHub* [online]. 14.01.2020 [cit. 2022-05-09]. Dostupné z: https://github.com/aspnet/Announcements/issues/402

[164] KALASKARSANKET. Microsoft Security Advisory CVE-2021-43877 | ASP.NET Core Elevation of privilege Vulnerability. In: *GitHub* [online]. 14.12.2021 [cit. 2022-05-09]. Dostupné z: https://github.com/dotnet/announcements/issues/206

[165] Download .NET 6.0. In: *Microsoft* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://dotnet.microsoft.com/en-us/download/dotnet/6.0

[166] RBHANDA. Microsoft Security Advisory CVE-2021-34532 | ASP.NET Core Information Disclosure Vulnerability. In: *GitHub* [online]. 10.08.2021 [cit. 2022-05-09]. Dostupné z: https://github.com/dotnet/announcements/issues/195

[167] Download .NET 5.0. In: *Microsoft* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://dotnet.microsoft.com/en-us/download/dotnet/5.0

[168] RBHANDA. Microsoft Security Advisory CVE-2021-1723 | ASP.NET Core Denial of Service Vulnerability. In: *GitHub* [online]. 01.12.2021 [cit. 2022-05-09]. Dostupné z: https://github.com/dotnet/announcements/issues/170

[169] What to Expect When Reporting Vulnerabilities to Microsoft. *Microsoft* [online]. 2020 [cit. 2022-05-09]. Dostupné z: https://msrc-blog.microsoft.com/2020/09/21/what-to-expect-when-reporting-vulnerabilities-to-microsoft/

[170] ASP.NET documentation. *Microsoft* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0

[171] Post-Quantum Cryptography. *NIST* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions

[172] Welcome to OpenSSL!. *OpenSSL* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://www.openssl.org/

[173] Newslog. In: *OpenSSL* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://www.openssl.org/news/newslog.html

[174] OpenSSL Usage Statistics. In: *Builtwith* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://trends.builtwith.com/Server/OpenSSL

[175] CVSS Scores For Openssl Openssl Between 2015-01-01 and 2022-05-09. In: *CVE Details* [online]. 09.05.2022 [cit. 2022-05-09]. Dostupné z: https://www.cvedetails.com/cvss-score-charts.php?fromform=1&vendor_id=&product_id=383&startdate=2015-01-01&enddate=2022-05-09&groupbyyear=1

[176] Vulnerabilities. In: *OpenSSL* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://www.openssl.org/news/vulnerabilities.html

[177] Openssl: Security Vulnerabilities. In: *CVE Details* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://www.cvedetails.com/vulnerability-list/vendor_id-217/product_id-383/Openssl-Openssl.html

[178] Community. *OpenSSL* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://www.openssl.org/community/

[179] Manpages for 3.0. *OpenSSL* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://www.openssl.org/docs/man3.0/

[180] Home. *OPEN QUANTUM SAFE* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://openquantumsafe.org/

[181] Language wrappers. *OPEN QUANTUM SAFE* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://openquantumsafe.org/liboqs/wrappers

[182] Open-quantum-safe/liboqs. In: *GitHub* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://github.com/open-quantum-safe/liboqs/

[183] Open-quantum-safe/openssl. In: *GitHub* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://github.com/open-quantum-safe/openssl

[184] Open-quantum-safe/liboqs-dotnet. In: *GitHub* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://github.com/open-quantum-safe/liboqs-dotnet

[185] Open-quantum-safe/liboqs-python. In: *GitHub* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://github.com/open-quantum-safe/liboqs-python

[186] About the Open Quantum Safe project. *OPEN QUANTUM SAFE* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://openquantumsafe.org/about/

[187] Liboqs. *OPEN QUANTUM SAFE* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://openquantumsafe.org/liboqs/

[188] Best Practices for Web Application Development in 2022. *Naturaily* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://naturaily.com/blog/best-practices-web-application-development

[189] What Is an X.509 Certificate & How Does It Work?. *Sectigo* [online]. 2021 [cit. 2022-05-10]. Dostupné z: https://sectigo.com/resource-library/what-is-x509-certificate

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

HTML	Hypertext Markup Language.
CSS	Cascading Style Sheets.
CMS	Content Management System
SSL	Secure Sockets layer
TLS	Transport Layer Security
HTTPS	Hypertext Transfer Protocol Secure
PKCS	Public Key Cryptography Standards
CBC	Cipher Block Chaining
CTR	Counter
CFB	Cipher Feedback
OFB	Output feedback
CCM	Counter with Cipher Block Chaining Message Authentication Code
EAX	Encrypt-then-authenticate-then-translate
GCM	Galois/counter mode
XOR	Exclusive Or
AES	Advanced Encryption Standard
IDEA	International Data Encryption Algorithm
RC	Rivest's Cipher / Ron's Code
DES	Data Encryption Standard
TDEA	Triple Data Encryption Algorithm
KDF	Key Derivation Function
AEAD	Authenticated Encryption with Associated Data
RSA	Rivest-Shamir-Adleman
ECC	Elliptic-curve Cryptography

ECDSA	Elliptic Curve Digital Signature Algorithm
EdDSA	Edwards-curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
EEECC	Elgamal Encryption using Elliptic Curve Cryptography
DH	Diffie-Hellman
ECDH	Elliptic-curve Diffie–Hellman
FHMQV	Fully Hashed Menezes-Qu-Vanstone
SHA	Secure Hash Algorithm
HMAC	Hash-based Message Authentication Code
KMAC	Keccak-based MAC
CMAC	Cipher-based MAC
GMAC	Galois MAC
GPU	Graphics Processing Unit
CPU	Central Processing Unit
RAM	Random Access Memory
SQL	Structured Query Language
NPM	Node.js Package Manager
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring Systém
MVC	Model-view-controller
MTV	Model-template-view
CLR	Common Language Runtime
OQS	Open Quantum Safe
PKI	Public Key Infrastructure
CA	Certificate Authority
FIPS	Federal Information Processing Standard

NIST National Institute of Standards and Technology

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH CD

Struktura obsahu přiloženého CD:

- Adresář **Text bakalářské práce** – obsahuje text bakalářské práce ve formátu PDF.