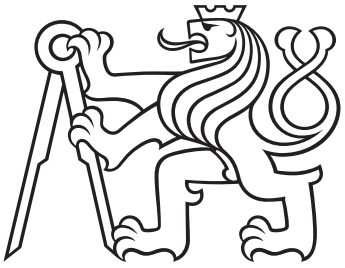
České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů
Studijní program: Otevřená informatika
Specializace: Kybernetická bezpečnost



Pseudonymizace dat pomocí HSM modulu

Data pseudonymization using the HSM module

DIPLOMOVÁ PRÁCE

Vypracoval: Bc. Pavel Novotný
Vedoucí práce: Ing. Pavel Náplava, Ph.D.
Rok: 2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Novotný** Jméno: **Pavel** Osobní číslo: **487037**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Kybernetická bezpečnost**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Pseudonymizace dat pomocí HSM modulu

Název diplomové práce anglicky:

Data pseudonymization using the HSM module

Pokyny pro vypracování:

- Proveďte rešerši zákonů vztahujících se k problematice osobních údajů na území Evropské unie, České republiky a Spolkové republiky Německo. Porovnejte je a vyhodnoťte jejich dopady do návrhu SW aplikací.
- Definujte důležité pojmy z oblasti kryptografie, popište HSM moduly a jejich výhody.
- Specifikujte způsoby komunikace s HSM moduly. Zaměřte se na Public-Key Cryptography Standards (PKCS), zejména na PKCS11 (Cryptoki).
- Navrhněte, a pomocí mikroslužeb vytvořte, testovací prostředí, které umožní testovat různé způsoby pseudonymizace dat. Toto prostředí bude fungovat jako webová aplikace, obsahující různé pseudonymizační moduly s možností volby jejich použití. V rámci práce je nutné vytvořit alespoň dva moduly. Architektura systému musí umožnit v budoucnu přidávat další.
- V rámci vytvořeného prostředí potvrďte, zamítněte, hypotézu o vhodnosti pseudonymizace dat v reálném prostředí pomocí využití HSM.
- Pro vyhodnocení hypotézy použijte, po dohodě s vedoucím práce, datovy vzorek vetsiho rozsahu.

Seznam doporučené literatury:

- TILBORG, Henk C.A. van. Encyclopedia Of Cryptography And Security. Springer Science+Business Media, Inc., 2005.
- NAŘÍZENÍ EVROPSKÉHO PARLAMENTU A RADY (EU) 2016/679 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů a o zrušení směrnice 95/46/ES (obecné nařízení o ochraně osobních údajů) [online]. Dostupné z: https://eur-lex.europa.eu/legal-content/CS/TXT/?uri=celex%3A32016R0679.
- NIST. FIPS PUB 140-2 Security Requirements for Cryptographic Modules [online]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Pavel Náplava, Ph.D. Centrum znalostního managementu FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **30.01.2023** Termín odevzdání diplomové práce: **26.05.2023**

Platnost zadání diplomové práce: **22.09.2024**

Ing. Pavel Náplava, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne .....

.....

Bc. Pavel Novotný

Poděkování

Rád bych poděkoval panu Ing. Pavlu Náplavovi, Ph.D. za vedení diplomové práce v letním semestru 2022/2023 na Fakultě elektrotechnické ČVUT v Praze, a to ve studijním programu Otevřená informatika. Děkuji za cenné rady a podněty, které mi poskytl.

Bc. Pavel Novotný

Název práce:
Pseudonymizace dat pomocí HSM modulu

Autor: Bc. Pavel Novotný

Studijní program: Otevřená informatika
Specializace: Kybernetická bezpečnost

Druh práce: Diplomová práce

Vedoucí práce: Ing. Pavel Náplava, Ph.D.

Abstrakt: V diplomové práci se zabýváme myšlenkou pseudonymizace dat většího rozsahu. Abychom byli schopni taková data efektivně zpracovávat, využijeme technologie z oblasti Big Data. Budeme předpokládat, že data, která chráníme, mají povahu osobních a citlivých údajů. Na ta se vztahují zvláštní legislativní požadavky dle nařízení EU. Je důležité taková data chránit před osobami, které ze své pracovní pozice nepotřebují s těmito daty pracovat. Vzhledem k důležitosti těchto dat volíme přístup pseudonimizace za využití HSM. V principu nám jde o návrh řešení, které bude jednoduše integrovatelné do reálného prostředí využívajícího Big Data technologie.

Klíčová slova: Pseudonymizace, GDPR, HSM, PKCS, Big Data

Title:
Data pseudonymization using the HSM module

Author: Bc. Pavel Novotný

Abstract: In this thesis, we explore the idea of pseudonymizing large-scale data. In order to be able to process such data efficiently, we will use technologies from the field of Big Data. We will assume that the data we protect is personal and sensitive in nature. These are subject to specific legislative requirements under EU regulations. It is important to protect such data from individuals who, by virtue of their job role, do not need to work with this data. Given the importance of this data, we choose a pseudonymisation approach using HSM. In principle, we want to design a solution that can be easily integrated into a real environment using Big Data technologies.

Key words: Pseudonymization, GDPR, HSM, PKCS, Big Data

Obsah

Seznam použitých zkratek **ix**

Seznam obrázků **xi**

Úvod **1**

1 Legislativní prostředí **2**
1.1 Anonymizace vs. pseudonymizace . . . 3
1.2 Obecné nařízení o ochraně osobních údajů (GDPR) . . . 3
1.2.1 Základní definice . . . 5
1.2.2 Zpracování osobních údajů . . . 6
1.2.3 Pseudonymizace . . . 7
1.3 Pověřenec pro ochranu osobních údajů . . . 8
1.4 Zákon 110/2019 Sb. . . . 9
1.4.1 Porušení povinností . . . 10
1.4.2 Přestupky . . . 10
1.5 Federal Data Protection Act (BDSG) . . . 11
1.5.1 Pověřenec pro ochranu osobních údajů . . . 11
1.5.2 Federální komisař pro ochranu dat a svobodu informací . . . 12
1.6 Požadavky na bezpečnost zpracování údajů . . . 12
1.7 Trestní ustanovení . . . 13
1.8 Souhlas s cookies . . . 13
1.9 Shrnutí legislativní části . . . 15

2 Základní pojmy kryptografie **16**
2.1 Terminologie k informační bezpečnosti . . . 17
2.2 Autentizace a autorizace . . . 17
2.3 Kryptosystém . . . 18
2.4 Symetrická kryptografie . . . 19
2.5 Asymetrická kryptografie . . . 20
2.6 Porovnání symetrické a asymetrické kryptografie . . . 20
2.7 Certifikát, elektronický a digitální podpis . . . 20
2.8 Hashovací funkce . . . 21
2.9 Shrnutí kapitoly . . . 22

3 Hardware Security Module (HSM) **23**
3.1 Obecná terminologie ke kryptografickým modulům . . . 24
3.2 Terminologie k obsluze kryptografických modulů . . . 24
3.3 Fyzická bezpečnost . . . 25
3.3.1 Terminologie k fyzické bezpečnosti . . . 25
3.4 Architektura . . . 27
3.5 Crypto API . . . 28
3.6 Shrnutí kapitoly . . . 28

4 Public Key Cryptography Standards (PKCS) **29**
4.1 PKCS #1: RSA Cryptography Specifications . . . 29
4.1.1 RSA . . . 30
4.1.2 Diffieho–Hellmanova výměna klíče . . . 31
4.2 PKCS #5: Password-Based Cryptography . . . 32

4.2.1 Key derivation function (KDF) . . . 32
4.2.2 PKCS #5 padding . . . 33
4.3 PKCS #11 . . . 34
4.4 Uzavření teoretické části práce . . . 37

5 Úvod do praktické části 38
5.1 Formulování hypotézy . . . 39
5.2 Technologie . . . 39
5.2.1 API pro komunikaci s HSM . . . 40
5.2.2 HDFS . . . 41
5.2.3 Spark . . . 42
5.2.4 Docker . . . 43
5.2.5 Cloudera CDP . . . 44
5.2.6 Formáty dat . . . 44
5.3 Shrnutí úvodu do praktické části . . . 44

6 Pseudonymizační prostředí 45
6.1 Plán realizace, motivace . . . 46
6.2 První větev / Testovací Docker prostředí / Aplikace . . . 47
6.3 Popis kontejnerů . . . 47
6.4 Kontejner HSM Utimaco simulátor . . . 48
6.5 Mikroslužba generování dat . . . 48
6.6 Mikroslužba ukládání souborů . . . 49
6.7 Backend . . . 49
6.7.1 Obecný úvod k modulům . . . 50
6.7.2 Integrované moduly . . . 51
6.7.3 Samostatné moduly . . . 52
6.8 Frontend . . . 52
6.9 Nasazení první větve / Aplikace . . . 53
6.10 Náčrt architektury aplikace . . . 54
6.11 Druhá větev / Reálné prostředí . . . 55
6.11.1 Použité HSM . . . 56
6.12 Praktické zkušenosti s HSM . . . 56
6.13 Nejslabší článek HSM . . . 57
6.14 Shrnutí pseudonymizačního prostředí . . . 57

7 Vyhodnocení výstupů práce 58
7.1 Provedené testy . . . 58
7.2 Vyhodnocení hypotézy . . . 59
7.3 Nezbytné konstatování . . . 59
7.4 Zpětné zhodnocení práce . . . 60
7.4.1 Důležité body vývoje . . . 60
7.5 Další možný rozvoj práce . . . 61
7.5.1 Z hlediska testovacího prostředí . . . 61
7.5.2 Z hlediska Big Data prostředí . . . 61
7.6 Shrnutí praktické části . . . 62

Závěr 63

Bibliografie 64

Přílohy 68
A Internetové odkazy . . . 68

Seznam použitých zkratek

EU European Union
GDPR General Data Protection Regulation
BDSG Federal Data Protection Act
BfDI Federal Commissioner for Data Protection and Freedom of Information
TTDSG Telecommunications-Telemedia Data Protection Act
HSM Hardware Security Module
PKCS Public Key Cryptography Standards
AES Advanced Encryption Standard
TLS Transport Layer Security
SHA Secure Hash Algorithm
ECB Electronic codebook
CBC Cipher block chaining
CTR Counter mode
CCM Counter with CBC-MAC
GCM Galois/Counter Mode
AEAD Authenticated Encryption with Associated Data
PKI Public Key Infrastructure
RSA Rivest–Shamir–Adleman
ECC Elliptic Curve Cryptography
QSCD Qualified Signature Creation Device
PIN Personal Identification Number
SO A Security Officer user
Cryptoki Cryptographic token interface
ANSDIT American National Standard Dictionary of Information Technology
NIST National Institute of Standards and Technology
FIPS Federal Information Processing Standard
OASIS Open Organization for the Advancement of Structured Information Standards
RFC Request for Comments
DH Diffieho–Hellmanova výměna klíče
DLP Discrete Logarithm Problem
IFP Integer Factorization Problem
SHS Secure Hash Standard

PRNG Pseudorandom number generator
TRNG True random number generator
CSP Critical security parameter
SPA Simple power analysis
EFP Environmental failure protection
EFT Environmental failure testing
SPIs Service Provider Interfaces
API Application programming interface
CXI Cryptographic eXtended services Interface
JCA Java Cryptography Architecture
JCE Java Cryptography Extension
HDFS Hadoop Distributed File System
UDF User Defined Functions
CSV Comma Separated Values
Parquet Apache Parquet
YARN Yet Another Resource Negotiator
RSAEP RSA Encryption Primitive
RSADP RSA Decryption Primitive
RSASP1 RSA Signature Primitive, version 1
RSAVP1 RSA Verification Primitive, version 1
A-CCA Adaptive Chosen Ciphertext Attack
OAEP Optimal Asymmetric Encryption Padding
PRF Pseudo-Random Functions
KDF Key derivation function
PBKDF1 Password-Based Key Derivation Function 1
PBKDF2 Password-Based Key Derivation Function 2
MAC Message Authentication Code
HMAC The Keyed - Hash Message Authentication Code
PoC Proof of Concept
ETL Extract, transform, load
NYC City of New York
MBK Master Backup Key
ESKM Enterprise Secure Key Manager
vESKM Virtual Enterprise Secure Key Manager
ARES Administrativní registr ekonomických subjektů
IČO Identifikační číslo osoby

Seznam obrázků

2.1 The conventional cryptosystem [24] 18

3.1 The internal elements of an HSM [24] 27
3.2 HSM design [35] 27
3.3 Logical software interfaces of an HSM [35] 28

4.1 How PKCS #11 works [46] 34
4.2 Object hierarchy [44] 35
4.3 Read-Only Session States [44] 36
4.4 Read/Write Session States [44] 36

5.1 HDFS Architecture [54] 41
5.2 Cluster Mode Overview [55] 42
5.3 Docker Architecture [57] 43

6.1 Architektura testovacího prostředí 54
6.2 Ilustrační foto použitého HSM [60] 56
6.3 Bezpečnostní karty [61] 56
6.4 PIN Pad [61] 56

7.1 Pseudonymizační aplikace 63

Úvod

Cílem diplomové práce je potvrdit nebo zamítnout hypotézu o vhodnosti pseudonymizace dat za využití HSM[1]. V práci jsme se rozhodli definovat výseč (viz kap. 5) z oblasti Big Data, dále jen Big Data. Potřebujeme tedy pracovat s technologiemi, které byly pro tyto účely navrženy.

Motivací pro výběr tématu je skutečnost, že lidé, společnosti a instituce při své činnosti v životě často volí drahé nástroje, služby a technologie, které nepotřebují, nebo je využívají neadekvátně či neefektivně. Tím využívají neefektivně i vynaložené finanční prostředky.

Příkladem nákladného zařízení je HSM. To lze považovat za bezpečné zařízení, které je možné integrovat do různých prostředí. Nabízí se otázka, zda-li jedním z těchto prostředí nemůže být to námi definované. A získáme vůbec za vynaložené finanční prostředky odpovídají přidanou hodnotu pro naše účely?

Pro práci máme k dispozici zařízení v ceně čtvrt milionu českých korun (viz kap. 6.11.1). Pokusíme se nasimulovat situaci vhodnou pro využití takto nákladného zařízení.

Diplomová práce je tématicky rozdělena na dvě hlavní části (teoretickou a praktickou). Teoretická část práce je věnována legislativě v oblasti ochrany osobních údajů na území EU[2] (viz kap. 1.2). Zaměříme se i na dva členské státy (Českou republiku a Německo). Následovat bude kapitola zabývající se základními pojmy z oblasti kryptografie (viz kap. 2). V předposlední kapitole teoretické části se detailněji zaměříme na teoretický náhled na HSM (viz kap. 3). V poslední kapitole zmíníme některé standardy PKCS[3] (viz kap. 4), které jsou v dnešní době již poněkud nekompletní, ale stále poskytují důležité informace.

Praktická část práce je uvedena již zmíněnou výsečí (viz kap. 5) z oblasti Big Data. Zahrnuje základní technologie z této oblasti (viz kap. 5.2) a dále pak kryptografická API[4] (viz kap. 5.2.1), které nám poslouží pro komunikaci s HSM. Teoretický pohled z úvodní části práce doplníme o pohled praktický. Také si představíme technické prostředí, které vybudujeme na běžném kancelářském počítači (viz kap. 6.2). Několik pseudonymizačních modulů v testovacím prostředí otestujeme. Z takto otestovaných modulů pak vybereme dva (viz kap. 6.7.3), které se pokusíme dále otestovat na reálném clustru za reálných podmínek na datech většího rozsahu. Formulujeme také hypotézu o vhodnosti pseudonymizace dat za využití HSM (viz kap. 5.1).

Závěr je věnován vyhodnocení výstupů práce a zamyšlení nad jejím možným budoucím rozvojem. Posledním bodem je pak samotné vyhodnocení hypotézy (viz kap. 7).

[1] Hardware Security Module
[2] European Union
[3] Public Key Cryptography Standards
[4] Application programming interface

Kapitola 1

Legislativní prostředí

Úvodní kapitolou začneme teoretickou část práce, kterou tvoří následujících kapitoly (viz kap. 1, 2, 3, 4). Jejich cílem je uvést čtenáře do legislativního prostředí, které se vztahuje k ochraně osobních údajů, a seznámit je se základními pojmy z oblasti kryptografie (viz kap. 2). Druhá polovina teoretické části je věnována obecnému pohledu na HSM (viz kap. 3). Závěr teoretické části je zaměřen na vybrané PKCS standardy, především na PKCS #11.

Cílem úvodní legislativní kapitoly není poskytnout komplexní přehled zákonů a jejich dopady. Spíše nám půjde o vytvoření jednoduchého průvodce, jenž nás uvede do problematiky, která by si zasloužila samostatné studium, resp. samostatnou práci.

Kapitola je rozdělena do tří základních bloků. V prvé řadě představíme pojmy anonymizace a pseudonymizace dat. Hlavním blokem je nařízení GDPR[1]. Popíšeme podstatné body odůvodnění nařízení a definujeme důležité pojmy vztahující se k uchovávání a zpracování osobních údajů (viz kap. 1.2.1). Mezi takové pojmy patří: subjekt údajů, osobní údaj, citlivý údaj, zpracování, správce osobních údajů, zpracovatel osobních údajů.

Dále zmíníme zákon 110/2019 Sb., který začleňuje GDPR do českého legislativního prostředí. Vedle tohoto zákona zmíníme zákon BDSG[2], který se vztahuje k legislativnímu prostředí Spolkové republiky Německo.

V závěru kapitoly (viz kap. 1.8) poukážeme na to, jak legislativa zasahuje do života běžného uživatele webových stránek v podobě vyjadřování souhlasu s *„cookies“*. Zde se podíváme i na konkrétní porovnání české a německé legislativy v této tzv. *„cookies“* oblasti. Konkrétně zmíníme paragrafy, které se této části věnují. Hlavní motivací je nastínit obecné formulace těchto paragrafů a poukázat na skutečnost, že jsou často obecné a není jasné, co spadá do jejich působnosti. Samotná *„cookies“* jsou samy generátorem dat, které lze dále zpracovávat.

Legislativní výčet uvažujme jako nejvyšší stupeň abstrakce (pseudonymizace), kterému je nutné v praktické části práce vyhovět. Nižším stupněm abstrakce jsou pak prostředky (šifrování), které využijeme, abychom naplnili abstrakci vyšší. V legislativě je uvedeno, že šifrování je jednou z doporučených forem provádění pseudonymizace. Právě k těmto účelům nám mohou HSM posloužit - jako tzv. technická opatření. Dále je důležité mít na paměti tzv. organizační opatření. Tím je např. využívání HSM oprávněnými osobami.

[1] General Data Protection Regulation
[2] Federal Data Protection Act

1.1 Anonymizace vs. pseudonymizace

Zatímco pojem pseudonymizace je definován v samotném nařízení GDPR (viz kap. 1.2.3), pojem anonymizace zde nenalezneme. Pro tyto účely využijeme parafrázi.

Anonymizace je nevratný proces, při kterém dojde ke ztrátě původní informace, a to takovým způsobem, že již nemůže být obnovena.

Jednoduchým příkladem je situace, kdy máme seznam jmen klientů. Každý řádek takového seznamu může v anonymizované podobě nabývat např. tří hodnot (muž, žena, neuvedeno). Každé jméno bude transformováno na jeho ekvivalent (např. na základě slovníku). Z výsledného seznamu již nelze původní data zpětně získat.

Pokud bychom si tuto transformaci poznamenali, resp. uzpůsobili, tak, aby ji bylo možné zpětně zrekonstruovat, již by se jednalo o pseudonymizaci.

Z výše uvedeného je patrné, že neexistuje, resp. není dobré používat, pojem *„deanonymizace“*. Jedná se o nevhodný termín. Data, která byla jednou anonymizována již nelze zpětně rekonstruovat. Neexistuje totiž mapování, jak bylo provedeno.

1.2 Obecné nařízení o ochraně osobních údajů (GDPR)

Nařízení je součástí sekundárního práva EU. Jedná se o právní předpis, který je právně závazný a platí v plném rozsahu pro všechny státy EU: *„NAŘÍZENÍ EVROPSKÉHO PARLAMENTU A RADY (EU) 2016/679 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů a o zrušení směrnice 95/46/ES (obecné nařízení o ochraně osobních údajů).“* [1]

Předpis chrání práva a svobody fyzických osob. Je tedy vhodné zdůraznit, že se toto nařízení nevztahuje na práva právnických osob. Dle odůvodnění v odstavci č. 10 nařízení vymezuje členským státům možnost blíže charakterizovat zvláštní kategorii citlivých údajů a dále stanovit vlastní pravidla. Nařízení se vztahuje jak na manuální, tak automatizované zpracování osobních dat.

Odůvodněni je nedílnou součástí nařízení. Zde namátkově uvádíme několik citací:

Dle odůvodnění odstavce č. 23 nařízení GDPR:

„Aby bylo zajištěno, že fyzickým osobám nebude odepřena ochrana, na niž mají podle tohoto nařízení nárok, mělo by se na zpracování osobních údajů subjektů údajů nacházejících se v Unii uskutečněné správcem nebo zpracovatelem, jenž není v Unii usazen, vztahovat toto nařízení, pokud činnosti zpracování souvisejí s nabídkou zboží nebo služeb těmto subjektům údajů bez ohledu na to, zda je spojena s platbou...“ [1]

Dle odůvodnění odstavce č. 26 nařízení GDPR:

„Zásady ochrany údajů by se měly uplatňovat na všechny informace týkající se identifikované nebo identifikovatelné fyzické osoby...“ [1]

Dle odůvodnění odstavce č. 39 nařízení GDPR:

„Zejména je zapotřebí, aby konkrétní účely, pro které jsou osobní údaje zpracovávány, byly jednoznačné a legitimní a aby byly stanoveny v okamžiku shromažďování osobních údajů...“ [1]

„Osobní údaje by měly být přiměřené, relevantní a omezené na to, co je nezbytné z hlediska účelů, pro které jsou zpracovávány. Je nezbytné zejména zajistit, aby byla doba, po kterou jsou osobní údaje uchovávány, omezena na nezbytné minimum. Osobní údaje by měly být zpracovány pouze tehdy, nemůže-li být účelu zpracování přiměřeně dosaženo jinými prostředky. Aby se zajistilo, že osobní údaje nebudou uchovávány déle, než je nezbytné, měl by správce stanovit lhůty pro výmaz nebo pravidelný přezkum...“ [1]

Při pročítání odůvodnění, které obsahuje 173 odstavců často narazíme na výraz dozorový úřad.

Dle článku č. 51 GDPR:

*„Každý členský stát stanoví, že jeden nebo více nezávislých orgánů veřejné moci jsou pověřeny monitorováním uplatňování tohoto nařízení s cílem chránit základní práva a svobody fyzických osob v souvislosti se zpracováním jejich osobních údajů a usnadnit volný pohyb osobních údajů uvnitř Unie (dále jen „**dozorový úřad**“).“* [1]

Dle článku č. 52 GDPR:

„Každý dozorový úřad jedná při plnění úkolů a při výkonu svých pravomocí podle tohoto nařízení zcela nezávisle.“ [1]

Dle článku č. 58 GDPR:

„Každý členský stát může v právních předpisech stanovit, že jeho dozorový úřad má další pravomoci než ty uvedené v odstavcích 1, 2 a 3. Výkon těchto pravomocí nesmí narušit účinné fungování kapitoly VII.“ [1]

Namátkově uvádíme následující pravomoci:

- *„nařídit správci a zpracovateli, případně zástupci správce nebo zpracovatele, aby mu poskytli veškeré informace, které potřebuje k plnění svých úkolů“* [1]
- *„upozornit správce či zpracovatele, že zamýšlené operace zpracování pravděpodobně porušují toto nařízení“* [1]

V rámci České republiky vykonává roli dozorujícího úřadu Úřad pro ochranu osobních údajů (viz kap. 1.4). V případě Spolková republiky Německo se jedná o Federálního komisaře pro ochranu dat a svobodu informací (tzv. BfDI[3] viz kap. 1.5.2).

[3]Federal Commissioner for Data Protection and Freedom of Information

1.2.1 Základní definice

Subjektem údajů je dle článku 4 GDPR a pro účely tohoto nařízení se rozumí:

„identifikovatelná fyzická osoba, kterou lze přímo či nepřímo identifikovat, zejména odkazem na určitý identifikátor, například jméno, identifikační číslo, lokační údaje, síťový identifikátor nebo na jeden či více zvláštních prvků fyzické, fyziologické, genetické, psychické, ekonomické, kulturní nebo společenské identity této fyzické osoby;“ [1]

Osobními údaji dle článku č. 4 GDPR a pro účely tohoto nařízení se rozumí:

„veškeré informace o identifikované nebo identifikovatelné fyzické osobě (dále jen „subjekt údajů“);“ [1]

Osobními citlivými údaji jsou dle odstavce č. 10 odůvodnění GDPR:

„Toto nařízení rovněž poskytuje členským státům určitý prostor ke stanovení vlastních pravidel, včetně pravidel pro zpracování zvláštních kategorií osobních údajů („citlivé osobní údaje“). V tomto rozsahu nařízení nevylučuje, aby právo členského státu stanovilo okolnosti konkrétních situací, při nichž dochází ke zpracování, včetně přesnějšího určení podmínek, za nichž je zpracování osobních údajů zákonné.“ [1]

a dále dle zákona § 66 odst. 6 zákona č. 110/2019 Sb. o zpracování osobních údajů:

„Kde se v dosavadních právních předpisech používá pojem citlivý údaj nebo citlivý osobní údaj, rozumí se tím ode dne nabytí účinnosti tohoto zákona osobní údaj, který vypovídá o rasovém nebo etnickém původu, politických názorech, náboženském vyznání nebo filosofickém přesvědčení nebo členství v odborové organizaci, genetický údaj, biometrický údaj zpracovávaný za účelem jedinečné identifikace fyzické osoby, údaj o zdravotním stavu, o sexuálním chování, o sexuální orientaci a údaj týkající se rozsudků v trestních věcech a trestných činů nebo souvisejících bezpečnostních opatření.“ [2]

Zpracováním dle článku č. 4 GDPR a pro účely tohoto nařízení se rozumí:

„jakákoliv operace nebo soubor operací s osobními údaji nebo soubory osobních údajů, který je prováděn pomocí či bez pomoci automatizovaných postupů, jako je shromáždění, zaznamenání, uspořádání, strukturování, uložení, přizpůsobení nebo pozměnění, vyhledání, nahlédnutí, použití, zpřístupnění přenosem, šíření nebo jakékoliv jiné zpřístupnění, seřazení či zkombinování, omezení, výmaz nebo zničení;“ [1]

Správcem dle článku č. 4 GDPR a pro účely tohoto nařízení se rozumí:

„fyzická nebo právnická osoba, orgán veřejné moci, agentura nebo jiný subjekt, který sám nebo společně s jinými určuje účely a prostředky zpracování osobních údajů; jsou-li účely a prostředky tohoto zpracování určeny právem Unie či členského státu, může toto právo určit dotčeného správce nebo zvláštní kritéria pro jeho určení;“ [1]

Zpracovatelem dle článku č. 4 GDPR a pro účely tohoto nařízení se rozumí:

„fyzická nebo právnická osoba, orgán veřejné moci, agentura nebo jiný subjekt, který zpracovává osobní údaje pro správce“ [1]

1.2.2 Zpracování osobních údajů

Kapitola č. 2, článek č. 5, definuje zásady zpracování osobních údajů. Mezi hlavní zásady patří zákonnost, korektnost a transparentnost. Veškeré údaje musí být zpracovány pro účely, pro které byly shromážděny *(tzv. účelové omezení)*. Údaje musí být shromážděny v přiměřeném rozsahu těmto účelům *(tzv. minimalizace údajů)*. Dále by měly věrně zachycovat skutečnost. Mělo by tedy docházet k aktualizacím a případně i k opravám nashromážděných údajů *(tzv. přesnost)*. [1]

Pro účely diplomové práce je důležitý především bod f odstavce č. 1, který zní:

„Osobní údaje musí být zpracovávány způsobem, který zajistí náležité zabezpečení osobních údajů, včetně jejich ochrany pomocí vhodných technických nebo organizačních opatření před neoprávněným či protiprávním zpracováním a před náhodnou ztrátou, zničením nebo poškozením („integrita a důvěrnost“)“ [1]

„Správce odpovídá za dodržení odstavce č. 1 a musí být schopen toto dodržení souladu doložit („odpovědnost“).“ [1]

Zákonnost zpracování osobních údajů může být založena mimo jiné na jednom z následujících bodů :

- *„subjekt údajů udělil souhlas se zpracováním svých osobních údajů pro jeden či více konkrétních účelů;„* [1]
- *„zpracování je nezbytné pro splnění smlouvy„* [1]

GDPR dále v článku č. 7 zmiňuje podmínky, za kterých lze udělit, případně, odvolat souhlas se zpracováním osobních údajů. Článek č. 9 o *Zpracování zvláštních kategorií osobních údajů* zakazuje zpracovávat údaje, které výčtem odpovídají definici citlivých osobních údajů (viz kap. 1.2.1). Zároveň definuje řadu případů, kdy se tento zákaz neuplatní.

Kapitola č. 3 nařízení GDPR v článku č. 15 definuje *Práva subjektu na přístup k osobním údajům*. Mezi tato práva patří:

- Právo na opravu, na výmaz („právo být zapomenut“), na omezení zpracování
- Právo na přenositelnost údajů - *„Subjekt údajů má právo získat osobní údaje, které se ho týkají, jež poskytl správci, ve strukturovaném, běžně používaném a strojově čitelném formátu“* [1]
- Právo vznést námitku - *„Subjekt údajů má z důvodů týkajících se jeho konkrétní situace právo kdykoli vznést námitku proti zpracování osobních údajů, které se jej týkají.“* [1]
- Právo na přístup ke shromážděným osobním údajům které se ho týkají - *„,a měl by moci toto právo snadno a v přiměřených odstupech uplatňovat, aby byl o jejich zpracování informován a mohl si ověřit jeho zákonnost.“* - viz odůvodnění odstavce č. 63 nařízení GDPR [1]

1.2.3 Pseudonymizace

Pseudonymizací dle článku č. 4 GDPR a pro účely tohoto nařízení se rozumí:

„zpracování osobních údajů tak, že již nemohou být přiřazeny konkrétnímu subjektu údajů bez použití dodatečných informací, pokud jsou tyto dodatečné informace uchovávány odděleně a vztahují se na ně technická a organizační opatření, aby bylo zajištěno, že nebudou přiřazeny identifikované či identifikovatelné fyzické osobě;“ [1]

Dle odůvodnění odstavce č. 26 nařízení GDPR:

„Osobní údaje, na něž byla uplatněna pseudonymizace a jež by mohly být přiřazeny fyzické osobě na základě dodatečných informací, by měly být považovány za informace o identifikovatelné fyzické osobě“ [1]

Dle odůvodnění odstavce č. 28 nařízení GDPR:

„Použití pseudonymizace osobních údajů může omezit rizika pro dotčené subjekty údajů a napomoci správcům a zpracovatelům splnit jejich povinnosti týkající se ochrany údajů. Výslovné zavedení „pseudonymizace“ v tomto nařízení nemá za cíl předem vyloučit jakákoliv další opatření týkající se ochrany údajů.“ [1]

Kapitola č. 4 nařízení GDPR, která se zabývá odpovědností správce uvádí, že správce je povinen zavést *„vhodná technická opatření a organizační opatření jako je pseudonymizace, jejichž účelem je provádět zásady ochrany údajů, jako je minimalizace údajů, účinným způsobem a začlenit do zpracování nezbytné záruky, tak aby splnil požadavky tohoto nařízení a ochránil práva subjektů údajů …, aby se standardně zpracovávaly pouze osobní údaje, jež jsou pro každý konkrétní účel daného zpracování nezbytné. Tato povinnost se týká množství shromážděných osobních údajů, rozsahu jejich zpracování, doby jejich uložení a jejich dostupnosti. Tato opatření zejména zajistí, aby osobní údaje nebyly standardně bez zásahu člověka zpřístupněny neomezenému počtu fyzických osob.“* [1]

Zabezpečení osobních údajů je definováno v článku č. 32. V tomto článku je správci a případnému zpracovateli nařízeno zajistit *„úroveň zabezpečení odpovídající danému riziku„*[1]. Mezi doporučené požadavky na zabezpečení se začleňuje:

- *„pseudonymizace a šifrování osobních údajů“* [1]
- *„schopnost zajistit neustálou důvěrnost, integritu, dostupnost a odolnost systémů a služeb zpracování“* [1]
- *„schopnost obnovit dostupnost osobních údajů a přístup k nim včas v případě fyzických či technických incidentů“* [1]
- *„proces pravidelného testování, posuzování a hodnocení účinnosti zavedených technických a organizačních opatření pro zajištění bezpečnosti zpracování“* [1]

„Při posuzování vhodné úrovně bezpečnosti se zohlední zejména rizika, která představuje zpracování, zejména náhodné nebo protiprávní zničení, ztráta, pozměňování, neoprávněné zpřístupnění předávaných, uložených nebo jinak zpracovávaných osobních údajů, nebo neoprávněný přístup k nim.“ [1]

Dle odůvodnění odstavce č. 82 nařízení GDPR:

„Aby správce nebo zpracovatel doložil soulad s tímto nařízením, měl by vést záznamy o činnostech zpracování, za které odpovídá. Každý správce a zpracovatel by měl být povinen spolupracovat s dozorovým úřadem a na jeho žádost mu tyto záznamy zpřístupnit, aby na jejich základě mohly být tyto operace zpracování monitorovány.“ [1]

1.3 Pověřenec pro ochranu osobních údajů

Poslední částí, kterou v rámci GDPR zmíníme, je role pověřence pro ochranu osobních údajů. Role je definována v článku č. 37. *„Jmenování pověřence pro ochranu osobních údajů“*. Dle tohoto článku nastává jmenování v každém případě, kdy:

- *„zpracování provádí orgán veřejné moci či veřejný subjekt, s výjimkou soudů jednajících v rámci svých soudních pravomocí;“*
- *„hlavní činnosti správce nebo zpracovatele spočívají v operacích zpracování, které kvůli své povaze, svému rozsahu nebo svým účelům vyžadují rozsáhlé pravidelné a systematické monitorování subjektů údajů; nebo“*
- *„hlavní činnosti správce nebo zpracovatele spočívají v rozsáhlém zpracování zvláštních kategorií údajů uvedených v článku 9 nebo osobních údajů týkajících se odsouzení v trestních věcech a trestných činů uvedených v článku 10.“* [1]

V navazujícím článku č. 38 *„Postavení pověřence pro ochranu osobních údajů“* je zmíněno několik bodů, jako příklad uvádíme následující:

- *„Správce a zpracovatel zajistí, aby byl pověřenec pro ochranu osobních údajů náležitě a včas zapojen do veškerých záležitostí souvisejících s ochranou osobních údajů.“* [1]
- *„Správce a zpracovatel podporují pověřence pro ochranu osobních údajů při plnění úkolů uvedených v článku č. 39 tím, že mu poskytují zdroje nezbytné k plnění těchto úkolů, přístup k osobním údajům a operacím zpracování a zdroje nezbytné k udržování jeho odborných znalostí.“* [1]
- *„Pověřenec pro ochranu osobních údajů je v souvislosti s výkonem svých úkolů vázán tajemstvím nebo důvěrností, v souladu s právem Unie nebo členského státu.“* [1]

Článek č. 39 *„Úkoly pověřence pro ochranu osobních údajů“* pojednává o úkolech, které tomuto pověřenci náleží. Mezi hlavní úkoly patří:

- spolupráce s dozorovým úřadem,
- poskytování informací a poradenství správcům nebo zpracovatelům,
- monitorování souladu s nařízením a dalšími předpisy Unie nebo členských států v oblasti ochrany údajů [1]

K pověřenci pro ochranu osobních údajů se vrátíme v zákoně BDSG, který v rámci německé legislativy úkoly rozšiřuje (viz kap. 1.5.1). V rámci české legislativy tyto úkoly rozšířeny nejsou.

1.4 Zákon 110/2019 Sb.

Zákon č. 110/2019 Sb. implementuje nařízení EU do českého právního prostředí. Do působnosti tohoto zákona mimo jiné spadá postavení a pravomoc Úřadu pro ochranu osobních údajů (dále jen „Úřad“). [3]

Dle § 50 odst. 1 zákona č. 110/2019 Sb. o zpracování osobních údajů

„Úřad je ústředním správním úřadem pro oblast ochrany osobních údajů v rozsahu stanoveném tímto zákonem, jinými právními předpisy, mezinárodními smlouvami, které jsou součástí právního řádu, a přímo použitelnými předpisy Evropské unie.“ [4]

Dle § 51 odst. 1 zákona č. 110/2019 Sb. o zpracování osobních údajů

„Do činnosti Úřadu lze zasahovat jen na základě zákona. Při výkonu své působnosti v oblasti ochrany osobních údajů Úřad postupuje nezávisle a řídí se pouze právními předpisy a přímo použitelnými předpisy Evropské unie.“ [5]

Mezi činnosti tohoto úřadu patří (uvádíme pouze namátkově):

- *„sdělením upozorňuje správce nebo zpracovatele, že zamýšleným zpracováním osobních údajů zřejmě poruší své povinnosti“* [6]
- *„provádí dozor nad dodržováním povinností stanovených zákonem při zpracování osobních údajů“* [7]
- *„přijímá podněty a stížnosti na porušení povinností stanovených zákonem při zpracování osobních údajů a informuje o jejich vyřízení“* [8]
- *„projednává přestupky a ukládá pokuty“* [9]
- *„poskytuje konzultace v oblasti ochrany osobních údajů“* [10]

Úřad pro svoji činnost využívá informace z různých informačních systémů. Vždy využívá zejména údaje o jménu, příjmení, datu narození, rodném čísle a adrese místa pobytu. Informační systémy, které úřad využívá jsou např.:

- Základní registr obyvatel
- Informační systém evidence obyvatel
- Informační systém cizinců

Dle § 55 odst. 5 zákona č. 110/2019 Sb. o zpracování osobních údajů:

„Z poskytovaných údajů lze v konkrétním případě využít vždy jen takové údaje, které jsou nezbytné ke splnění daného úkolu.“ [11]

1.4.1 Porušení povinností

Dle § 60 zákona č. 110/2019 Sb. o zpracování osobních údajů:

„Dojde-li k porušení povinnosti stanovené zákonem nebo uložené na jeho základě při zpracování osobních údajů podle hlavy II nebo III nebo podle nařízení Evropského parlamentu a Rady (EU) 2016/679, může Úřad uložit opatření k odstranění zjištěných nedostatků a stanovit přiměřenou lhůtu pro jejich odstranění.“ [12]

1.4.2 Přestupky

Dle § 61 zákona č. 110/2019 Sb. o zpracování osobních údajů:

„Fyzická osoba, právnická osoba nebo podnikající fyzická osoba se dopustí přestupku tím, že poruší zákaz zveřejnění osobních údajů stanovený jiným právním předpisem.“ [13]

Za tento přestupek lze udělit pokutu jeden milion korun nebo pět milionu korun, *„jde-li o přestupek spáchaný tiskem, filmem, rozhlasem, televizí, veřejně přístupnou počítačovou sítí nebo jiným obdobně účinným způsobem.“* [14]

Namátkově uvádíme, co by musela spáchat právnická osoba, aby se dopustila přestupku:

- nestanoví účel zpracování osobních údajů
- údaje uchovává delší, než nezbytně nutnou dobu
- nevyhovuje žádostem subjektů
- nepřijme technická a organizační opatření nebo nevede jejich dokumentaci
- poruší omezení zpracování zvláštních kategorií osobních údajů
- neohlásí porušení zabezpečení osobních údajů Úřadu
- a další [15]

Za výše zmíněné přestupky lze uložit pokutu až do výše deseti milionů korun.

1.5 Federal Data Protection Act (BDSG)

Zákon, který začleňuje GDPR do německého právního prostředí, se nazývá BDSG. Zákon existoval již dříve, ale byl z důvodu GDPR po téměř čtyřiceti letech změněn. V prvním paragrafu tohoto zákona nalezneme jeho působnost. Zde je mimo jiné řečeno, že ustanovení zákona se neuplatní v případě, kdy je přímo použitelné nařízení GDPR. To vychází také ze skutečnosti, že nařízení má právní přednost. Část první a druhá tohoto zákona se dále uplatní pro činnosti veřejných institucí, na které se přímo nevztahuje GDPR, případně směrnice 2016/680, nestanoví-li tento, případně jiný zákon jinak. Pro další porozumění musíme zmínit definice, které jsou uvedeny ve druhém paragrafu tohoto zákona. [16]

- **Public bodies of the Federation** *„jsou orgány, soudní orgány a jiné veřejnoprávní instituce federace nebo přímé federální korporace, statutární orgány a nadace zřízené podle veřejného práva a jejich sdružení bez ohledu na jejich právní formu.“* [16]
- **Public bodies of the Länder** *„jsou orgány, soudní orgány a jiné veřejnoprávní instituce země, obce, svazku obcí nebo jiných veřejnoprávních právnických osob podléhajících pozemkovému dozoru a jejich sdružení bez ohledu na jejich právní formu.“* [16]
- **Private bodies** *„jsou fyzické a právnické osoby, společnosti a jiná soukromoprávní sdružení, pokud se na ně nevztahují odstavce 1 až 3.....“* [16]

1.5.1 Pověřenec pro ochranu osobních údajů

Oproti GDPR rozlišuje zákon dva typy pověřenců, a to v návaznosti na oblast sektorů, ke kterým se vztahují (*„public bodies“* a *„private bodies“*). U obou typů těchto pověřenců zákon blíže specifikuje jejich postavení, pravomoci a povinnosti. V případě zmocněnce v sektoru „*public bodies*“ se jeho jmenování řídí stejnými požadavky jako v GDPR.

V rámci sektoru „*private bodies*“ je nad rámec článku č. 37 GDPR zpracovateli nařízeno, aby jmenoval zmocněnce v situaci, kdy trvale zaměstnává alespoň 20 osob zabývajících se automatizovaným zpracováním osobních údajů. V případě typu zpracování, které se svoji charakteristikou blíží k článku č. 35 GDPR o *„Posouzení vlivu na ochranu osobních údajů“*, je zpracovatel povinen jmenovat správce bez ohledu na počet osob zabývajících se automatizovaným zpracováním osobních údajů. § 6 BDSG o postavení pověřence je převzata z článku č. 38 GDPR o *„Postavení pověřence pro ochranu osobních údajů“* s tím rozdílem, že jednotlivé povinnosti se vždy plně nevztahují na sektor „*private bodies*“. [16]

Úkoly pověřence v rámci „*public bodies*“ jsou prakticky identické s těmi zmíněnými v GDPR. V zákoně jsou ovšem obecná ustanovení nařízení doplněna o konkrétní odkazy na paragrafy BDSG. Dále tu je explicitně zmíněna *„SMĚRNICE EVROPSKÉHO PARLAMENTU A RADY (EU) 2016/680 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů příslušnými orgány za účelem prevence, vyšetřování, odhalování či stíhání trestných činů nebo výkonu trestů, o volném pohybu těchto údajů a o zrušení rámcového rozhodnutí Rady 2008/977/SVV.“* Pověřenec je povinen při své činnosti zajišťovat poradenství a shodu mimo jiné i s touto směrnicí.

1.5.2 Federální komisař pro ochranu dat a svobodu informací

Zákon ustanovuje federálního komisaře jako nejvyšší federální autoritu, která sídlí v Bonnu. Federálního komisaře jmenuje spolkový prezident po zvolení Německým spolkovým sněmem na návrh spolkové vlády. Funkční období komisaře je pětileté s možností jednoho opakování. Představíme § 14, který zmiňuje úkoly federálního komisaře, a § 16 o jeho pravomocích. Ostatní paragrafy k federálnímu komisaři vynecháme, protože jsou pro účely této práce nadbytečné. Úkoly zmíněné v nařízení jsou poupraveny tak, aby odpovídaly německé legislativě. Namátkově uvádíme pár bodů:

- *„sledovat a prosazovat uplatňování tohoto zákona a dalších právních předpisů na ochranu údajů, včetně právních předpisů přijatých k implementaci směrnice (EU) 2016/680;“* [16]
- *„podporovat povědomí veřejnosti o rizicích, pravidlech, zárukách a právech v souvislosti se zpracováním osobních údajů a jejich pochopení, přičemž zvláštní pozornost bude věnována opatřením týkajících se dětí;“* [16]
- *„sledovat příslušný vývoj, pokud má dopad na ochranu osobních údajů, zejména vývoj informačních a komunikačních technologií a obchodních praktik;“* [16]
- a nespočet dalších

V rámci svých pravomocí komisař spolupracuje s veřejnými orgány odpovědnými za kontrolu dodržování ustanovení o ochraně údajů ve spolkových zemích.

„Public bodies of the Federation“ jsou povinny poskytnout federálnímu komisaři a jeho asistentům následující:

- *„přístup do všech úředních prostor, včetně všech zařízení a prostředků pro zpracování údajů, a ke všem osobním údajům a všem informacím nezbytným k plnění jejich úkolů“* [16]

Subjekt údajů má právo vznést stížnost k federálnímu komisaři v případě, že se domnívá, že zpracováním jeho osobních údajů dochází k porušení jeho práva.

1.6 Požadavky na bezpečnost zpracování údajů

§ 64 *„Requirements for the security of data processing“* se zabývá požadavky, které se vztahují ke zpracovateli osobních údajů, Nad rámec nařízení článku č. 32 GDPR (viz kap. 1.2.3) ukládá zpracovateli povinnost zohlednit *„technické směrnice a doporučení Spolkového úřadu pro informační bezpečnost.“* Zjednodušeně lze říci, že tento paragraf se snaží položit silné základy pro dosažení informační bezpečnosti. V odstavci č. 3 jsou zmíněna opatření, která má správce a zpracovatel po vyhodnocení příslušných rizik zavést. Mezi tato opatření namátkově patří:

- Zajištění dostatečné autorizace a zabránění neautorizovanému přístupu k datům.
- Zajištění ochrany osobních údajů před ztrátou a zničením.
- Pro plný výčet odkazujeme na plné znění zákona. [16]

V ostatních paragrafech lze najít např. povinnosti správce a zpracovatele o zajištění dostatečného tzv. *„logování“* v rámci automatizovaného zpracování dat. Zde musí být uvedeno, jaká osoba a v jakém časovém období provedla příslušné operace. Zákon dále zmiňuje osoby a instituty, které k těmto záznamům (tzv. *„logs“*) mají přístup. Zejména se jedná o pověřence pro ochranu osobních údajů a federálního komisaře. *„Logy“* by měly být smazány ke konci následujícího roku, kdy byly vytvořeny.

1.7 Trestní ustanovení

Dle § 42 BDSG

Na tři roky nebo peněžitou pokutou bude potrestán ten, kdo úmyslně zveřejní osobní data velkého počtu osob pro komerční účely. [16]

Na dva roky nebo peněžitou pokutou bude potrestán ten, kdo s úmyslem obohatit sebe nebo osobu jinou zpracovává osobní data bez příslušného oprávnění, případně zpracovává data, která získal podvodným jednáním. [16]

Výše uvedené trestné činy budou stíhány pouze za předpokladu, že oprávněné subjekty podají stížnost. [16]

1.8 Souhlas s cookies

TTDSG[4] je zákon, který platí na území Německa od prvního prosince 2021. Tento zákon konsoliduje několik dřívějších zákonů. Působnost se vztahuje na poskytovatele telekomunikačních služeb a obecněji na poskytovatele služeb internetových. Jedná se o nakládání s tzv. *„cookies“*. Nutné je poznamenat, že termín *„cookies“* zde nenalezneme.

Přesnou formulaci nalezneme v § 25 s názvem *Schutz der Privatsphäre bei Endeinrichtungen*. Paragraf zmiňuje ochranu soukromí v koncových zařízeních. Jestliže mají být data uložena v koncovém zařízení (*„cookies“*), musí k tomu dát koncový uživatel souhlas. V zákoně se explicitně zmiňuje, že souhlas musí být v přímém souladu s GDPR. Výjimku z tohoto souhlasu mají tzv. *„technická cookies“* neboli data, která jsou nezbytná pro zprostředkování služby. [17]

Pokud bychom chtěli v české legislativě dohledat informace a náležitosti souhlasu s tzv. *„cookies“*, museli bychom nahlédnout do *zákona č. 127/2005 Sb. o elektronických komunikacích a o změně některých souvisejících zákonů*, konkrétně do ustanovení § 89.

Dle § 89 odst. 3 zákona č. 127/2005 Sb. o elektronických komunikacích:

„Každý, kdo hodlá používat nebo používá sítě elektronických komunikací k ukládání údajů nebo k získávání přístupu k údajům uloženým v koncových zařízeních účastníků nebo uživatelů, získá od těchto účastníků nebo uživatelů předem prokazatelný souhlas s rozsahem a účelem jejich zpracování. Tato povinnost neplatí pro technické ukládání nebo přístup výhradně pro potřeby přenosu zprávy prostřednictvím sítě elektronických komunikací nebo je-li to nezbytné pro potřeby poskytování služby informační společnosti, která je výslovně vyžádána účastníkem nebo uživatelem.“ [18]

[4] Telecommunications-Telemedia Data Protection Act

Je pozoruhodné, že žádný z výše zmíněných paragrafů nevyužívá pojmy z oblasti webových technologií. Terminologie je velice obecná a vyhovuje ji i např. chytré auto, které potřebuje komunikovat se vzdáleným serverem.

Podíváme li se na jeden ze zajímavých rozdílů TTDSG a *zákona č. 127/2005 Sb. o elektronických komunikacích*, jedná se o formu vyjádření souhlasu. Jak již bylo řečeno, v TTDSG je explicitně uvedeno, že se řídí platným nařízením GDPR. V případě *zákona č. 127/2005 Sb. o elektronických komunikacích* je situace odlišná. Zmíníme informace uvedené na stránkách Úřadu pro ochranu osobních údajů, konkrétně v kategorii *Často kladené otázky podle oblastí zpracování údajů.*

„Jaký je rozdíl mezi udělením souhlasu podle zákona o elektronických komunikacích a souhlasem podle obecného nařízení? Je třeba získávat dva samostatné souhlasy?“ [19]

„Zákon o elektronických komunikacích stanoví, že pro využití souborů cookies a podobných technologií je nutné předem získat prokazatelný souhlas uživatelů internetových stránek. Výjimku tvoří pouze tzv. technické cookies. Získaný souhlas tedy umožňuje využití netechnických cookies, neřeší však zpracování osobních údajů, ke kterému prostřednictvím cookies dochází. Souhlasem je pouze umožněno správci ukládání cookies do zařízení koncového uživatele. Pro zpracování osobních údajů prostřednictvím cookies třeba disponovat právním titulem podle obecného nařízení. ...“ Pro detailnější informace odkazujeme na plné znění často kladených dotazů. [19]

Závěrem obrátíme pozornost k odstavci č. 32 odůvodnění GDPR, který se zabývá udělováním souhlasu.

„Souhlas by měl být dán jednoznačným potvrzením, které je vyjádřením svobodného, konkrétního, informovaného a jednoznačného svolení subjektu údajů ke zpracování osobních údajů, které se jej týkají, a to v podobě písemného prohlášení, i učiněného elektronicky, nebo ústního prohlášení. Mohlo by se například jednat o zaškrtnutí políčka při návštěvě internetové stránky, volbu technického nastavení pro služby informační společnosti nebo jiné prohlášení či jednání, které v této souvislosti jasně signalizuje souhlas subjektu údajů s navrhovaným zpracováním jeho osobních údajů. ***Mlčení, předem zaškrtnutá políčka nebo nečinnost by tudíž neměly být považovány za souhlas. Souhlas by se měl vztahovat na veškeré činnosti zpracování prováděné pro stejný účel nebo stejné účely****. Jestliže má zpracování několik účelů, měl by být souhlas udělen pro všechny. Má-li subjekt údajů vyjádřit souhlas na základě žádosti podané elektronickými prostředky, musí být žádost jasná a stručná a nesmí zbytečně narušit využívání služby, pro kterou je souhlas dáván.“* [1]

1.9 Shrnutí legislativní části

Cílem úvodní kapitoly bylo přiblížit legislativní prostředí EU v oblasti zpracování osobních údajů. Stěžejním textem byly definice z nařízení GDPR. Nahlédli jsme do legislativy dvou členských zemí (České republiky a Německa). Zmínili jsme několik podstatných zákonů, které tuto legislativu integrují do právního prostředí dané země. Z výše představených zákonů je patrné, že s osobními údaji je nezbytné nakládat maximálně obezřetně. Z porušování těchto právních předpisů vyplývají nemalé finanční postihy, důsledkem může být i ztráta důvěryhodnosti společností, které by jednaly v rozporu s těmito předpisy. Zamýšlet se nad ochranou dat za využití HSM je opodstatněné.

GDPR je komplexním nařízením, které se dotýká celé řady zákonů. To je také jedním z důvodů, proč vzbuzuje všeobecný respekt. Dané téma by si zasloužilo další pozornost. Například by bylo možné diskutovat i *zákon č. 181/2014 Sb. o kybernetické bezpečnosti*. Mimo legislativní stránku je dále důležitá i konkrétní politika organizace, která má ochranu dat na starost. Dalším tématem jsou pak mezinárodní certifikace, kterými se ale již zabývat nebudeme.

Rozvíjet tématiku kapitoly dále by znamenalo také se zaměřit na samotnou strukturu zákonů v daných členských státech. Na příkladu německého zákona TTDSG a českého *zákona č. 127/2005 Sb. o elektronických komunikacích* lze poukázat na fakt, že oba zákony se věnují rozdílným oblastem, ale mají společný průsečík. Je jím již zmíněná problematika *„cookies“*. Té se vždy ovšem týká jen malý odstavec.

Následující kapitoly diplomové práce se již věnují technickým tématům, které jsou podstatou samotné práce.

Kapitola 2

Základní pojmy kryptografie

Tato kapitola slouží jako stručný přehled pojmů, které budeme v kontextu celé práce potřebovat. V první části se zaměříme na pojmy jako je kryptosystém, kryptoanalýza, symetrická kryptografie a její prostředky (tajný klíč). Dále nás bude zajímat pojem asymetrické kryptografie a příslušné pojmy (veřejný klíč, soukromý klíč). V závěru kapitoly se zaměříme na typy elektronických podpisů. Zmíníme rozdíl mezi elektronickým a digitálním podpisem. V neposlední řadě definujeme pojem certifikát a hashovací funkce.

Velice často budeme potřebovat terminologie z různých oblastí bezpečnosti. Pro tyto účely využijeme zejména následující zdroje:

- ANSDIT[1] [20]
- RFC[2] 4949 (Internet Security Glossary, Version 2) [21]

a dále příslušné tématické standardy:

- ISO/IEC 27000:2018 [22]
- NIST[3] FIPS[4] PUB 180-4 [23]

První pojem, který definujeme je samotná kryptografie:

Kryptografie *„je disciplína, která zahrnuje principy, prostředky a metody pro transformaci dat za účelem skrytí jejich sémantického obsahu, zabránění jejich neoprávněnému použití nebo zabránění jejich nezjištěné modifikaci.“* [20] (tzv. *„**cryptography**“*)

Je podstatné uvědomit si, že **bezpečnost je proces**. Tento proces má svůj začátek a cíl. Cílem procesu je udržet chráněné **aktivum** (viz kap. 2.1) v akceptovatelných mezích rizik. Jinými slovy: vždy se v rámci tohoto procesu snažíme rizika mitigovat. Jedním z takových aktiv je **informace** (viz kap. 2.1). Chceme-li porozumět pojmu informace v kontextu bezpečnosti, měli bychom nahlédnout do normy ISO/IEC 27000:2018. [22] Tuto normu využijeme i pro další pojmy, kterými jsou informační bezpečnost, utajení, dostupnost a integrita.

[1] American National Standard Dictionary of Information Technology
[2] Request for Comments
[3] National Institute of Standards and Technology
[4] Federal Information Processing Standard

2.1 Terminologie k informační bezpečnosti

Aktivum *„je systémový prostředek, který je*

- *chráněný bezpečnostní politiku informačního systému,*
- *určený k ochraně protiopatřením, nebo*
- *potřebný pro poslání systému.“* [21]

Informace *„je aktivem, které je, stejně jako ostatní, důležitá obchodní aktiva zásadní pro podnikání organizace, a proto je třeba je vhodně chránit ... Ať už má informace jakoukoli formu nebo způsob, jakým je přenášena, vždy potřebuje vhodnou ochranu.“* [22]

Informační bezpečnost *„zajišťuje důvěrnost, dostupnost a integritu informace.“* [22]

Důvěrnost *„informace nejsou zpřístupňovány nebo sdělovány neoprávněným jednotlivcům, subjektům nebo procesům“* [22] (tzv. *„**confidentiality**“*)

Dostupnost *„je vlastnost být přístupný a použitelný na vyžádání oprávněným subjektem.“* [22] (tzv. *„**availability**“*)

Integrita *„je vlastnost přesnosti a úplnosti.“* [22] (tzv. *„**integrity**“*)

Spolehlivost *„je vlastnost konzistentního zamýšleného chování a výsledků“* [22]

Proces *„je soubor vzájemně souvisejících nebo interagujících činností, které přeměňují vstupy na výstupy“* [22]

2.2 Autentizace a autorizace

Entita *„ je cokoli, jako je osoba, místo, proces, objekt, koncept, asociace nebo událost. ...“* [21]

Autentizace *„je proces ověřování tvrzení, že systémová entita nebo systémový prostředek má určitou hodnotu atributu.“* [21] (tzv. *„**authentication**“*)

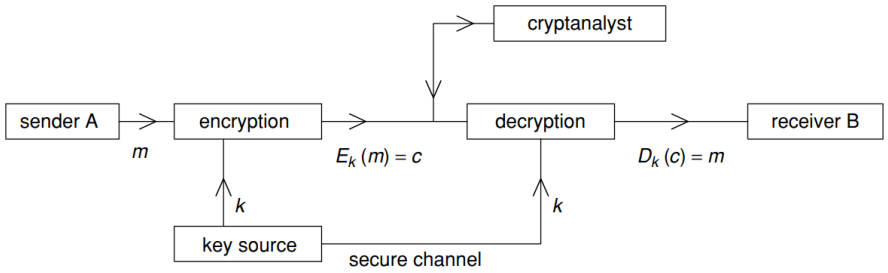
Autentizační proces se skládá ze dvou kroků, prvním krokem je identifikace a druhým je ověření. Hodnotou atributu může být například unikátní identifikátor uživatele.

- **Identifikace** *„je akt nebo proces, který předkládá identifikátor systému tak, že systém dokáže rozpoznat systémovou entitu a rozlišit ji od jiných subjektů.“* [21] (tzv. *„**identification**“*)
- **Verifikace** *„je proces zkoumání informace k zajištění pravdivosti tvrzené skutečnosti nebo hodnoty.“* [21] (tzv. *„**verification**“*)

Autorizace *„je proces pro udělení souhlasu systémové entitě k přístupu k systémovému prostředku.“* [21] (tzv. *„**authorization**“*)

2.3 Kryptosystém

Technickými prostředky, které může proces bezpečnosti využívat k dosažení svých cílů, jsou kryptosystémy. Formální popis kryptosystému (viz obr. 2.1) poskytl v roce 1949 matematik Claude Shannon. [24]



Obrázek 2.1: The conventional cryptosystem [24]

Na modelu je zachyceno několik komponent, které si postupně rozebereme a nastíníme jejich funkce. Prvními komponentami jsou dvě strany *„Sender A a Sender B“* (tzv. Alice a Bob), které si chtějí vyměnit zprávu *„m“* (tzv. otevřený text). Otevřený text představuje chráněné aktivum daným procesem. Dalšími komponentami jsou dva algoritmy, *„encryption“* a *„decryption“*, dále jen šifrovací, resp. dešifrovací algoritmus. Šifrovací funkce, která se označuje $E_k(m)$ obsahuje na vstupu otevřený text. Výstupem této funkce je *„c“* (tzv. zašifrovaný text). Dešifrovací funkce se označuje $D_k(c)$ a jejím argumentem je výstup z šifrovací funkce, tedy zašifrovaný text. Jejím výstupem je otevřený text. Do obou funkcí vstupuje tajná informace *„k“* (tzv. klíč). Klíč je nutné distribuovat mezi Alici a Boba přes *„secure channel“* (tzv. chráněný kanál). Pro ilustraci si lze představit poštovního holuba, který letí od Alice k Bobovi s daným klíčem. Klíč je jediná komponenta, kterou je potřeba chránit. Ostatní komponenty by se chránit neměly, viz Kerckhoffův princpip,

Kerckhoffův princpip zní následovně:

„A cryptosystem should be secure even if everything about the system, except the key, is public knowledge“

Snahou tohoto principu je uvědomit si skutečnost, že musíme předpokládat, že potenciální útočník má k dispozici veškerou znalost kromě již zmiňovaného klíče. Kerckhoffův princpip vychází z článku, který byl publikován v roce 1883 pod názvem *„La Cryptographie Militaire“* a jehož autorem je holandský kryptograf Auguste Kerckhoff. V článku popisuje šest principů praktického návrhu šifer, kde právě druhý z těchto bodů je podkladem pro Kerckhoffův princpip:

„The design of a system should not require secrecy and compromise of the system should not inconvenience the correspondents“

Vrátíme-li se k tzv. *„Shannonově modelu“*, objevíme zde poslední neprodiskutovanou komponentu a tou je *„cryptanalyst“*, dále jen osoba provádějící kryptoanalýzu. Kryptoanalýza je vědní obor, který se zabývá metodami, které se snaží získat otevřený text ze zašifrovaného textu a to bez znalosti klíče. Komponenta tedy symbolizuje skutečnost, že přenosové médium může být vystaveno různým metodám kryptoanalýzy a je nutné kryptosystémy konstruovat tak, aby byly vůči takovým útokům odolné.

2.4 Symetrická kryptografie

Mezi symetrickou kryptografii patří kryptosystémy, které využívají pro proces šifrování a dešifrování stejný tajný klíč. Z hlediska terminologie je klíč pro symetrickou kryptografii nazýván jako tzv. ***„secret key“***. Toto označení bude důležité v kapitole o standardu PKCS, konkrétně v podkapitole PKCS #11 (viz kap. 4.3). Symetrická kryptografie je řádově rychlejší než asymetrická. Mezi nejvýznamnější představitele se řadí bloková šifra AES[5], která vznikla na přelomu tisíciletí.

Parametrem AES je klíč o daném počtu bitů. AES podporuje klíče délky 128, 192 a 256 bitů. Délka použitého klíče je připsána za zkratkou algoritmu. Korektní označení použité šifry obsahuje použitý algoritmus a zvolenou délku klíče, např. AES-128, AES-192, případně AES-256. Vyšší počet použitých bitů v klíči znamená vyšší stupeň ochrany. Toto samotné označení ale nestačí. Aby bylo možné algoritmus využít, musí pracovat v tzv. režimu činnosti. Mezi tyto režimy namátkově patří: ECB[6], CBC[7], CTR[8] (tzv. základní režimy), CCM[9] a GCM[10] (tzv. AEAD[11] režimy). Jak název napovídá, jedná se o mechanismus, kdy režim činnosti nezajišťuje pouze utajení dat, ale zároveň integritu a autentizaci. Pro popis základních režimů odkazujeme na webovou stránku [25]. Pro popis GCM režimu pak odkazujeme na dokument [26].

Běžný uživatel internetu narazí na blokovou šifru AES na internetu. Je používána v kryptografických sadách pro TLS[12]. TLS využívá sadu protokolů, které pomáhají zajistit bezpečnost v internetovém prostředí. Pro běžného uživatele je pak stěžejním a jediným viditelným prvkem protokol HTTPS. Ten je indikován tzv. *„zeleným zámkem“* vedle vyhledávací lišty v příslušném prohlížeči. Právě dva poslední zmíněné režimy činnosti (CCM a GCM) jsou jako jediné využívány v poslední dostupné verzi TLS 1.3 v kombinaci s AES. Poslední kryptografická sada využívá proudovou šifru ChaCha20.

Kryptografické sady TLS 1.3 jsou následující:

- TLS_**AES_256_GCM**_SHA384
- TLS_**AES_128_GCM**_SHA256
- TLS_**AES_128_CCM**_SHA256
- TLS_**AES_128_CCM**_SHA256
- TLS_CHACHA20_POLY1305_SHA256

Všechny kryptografické sady v TLS 1.3 obsahují pouze algoritmy podporující AEAD. Detailnější informace o protokolu TLS 1.3 nalezneme např. zde [27], které tento protokol přehledně a detailně popisují. Obdobná stránka existuje i pro protokol TLS 1.2 [28].

[5] Advanced Encryption Standard
[6] Electronic codebook
[7] Cipher block chaining
[8] Counter mode
[9] Counter with CBC-MAC
[10] Galois/Counter Mode
[11] Authenticated Encryption with Associated Data
[12] Transport Layer Security

2.5 Asymetrická kryptografie

Asymetrická kryptografie se odlišuje od symetrické v použitém typu klíče. V asymetrické kryptografii rozlišujeme dva klíče. První je tzv. ***„public key“*** a druhý je tzv. ***„private key“***, dále jen veřejný klíč, resp. soukromý klíč. Veřejný klíč se používá pro účely šifrování. Soukromý klíč naopak pro účely dešifrování. Jak již samotné názvy klíčů napovídají, veřejný klíč není potřeba uchovávat v tajnosti. Soukromý klíč je zapotřebí chránit. Příkladem algoritmu, který využívá asymetrickou kryptografii je RSA[13], detailněji je popsán v kapitole PKCS, konkrétně v podkapitole PKCS #1 (viz kap. 4.1.1). Asymetrická kryptografie se často nazývá kryptografie veřejného klíče (tzv. ***„public key cryptography“***).

2.6 Porovnání symetrické a asymetrické kryptografie

Asymetrická kryptografie má oproti symetrické tu výhodu, že obě strany se nemusí předem dohodnout na daném klíči. Pokud Alice chce poslat Bobovi zašifrovanou zprávu, potřebuje k tomu znát Bobův veřejný klíč a naopak. Oproti symetrické kryptografii nepotřebujeme hledat tajnou cestu, jak si Alice s Bobem vymění veřejné klíče. Své veřejné klíče si mohou navzájem poslat skrze veřejný kanál. Nicméně asymetrická kryptografie je oproti symetrické extrémně pomalá. Jako příklad lze uvést velikost klíčů, které se používají. Chceme-li dosáhnout stejného zabezpečení jako v případě AES-128, museli bychom v případě protokolu RSA použít modul délky cca 3072 bitů. Definici pojmu modul nalezneme opět v kapitole PKCS (viz kap. 4.1.1). V případě AES-256 je rozdíl markantnější. Zde je odpovídající velikost modulu pro RSA cca 15 380 bitů.

2.7 Certifikát, elektronický a digitální podpis

Certifikát je technický prostředek, jak provázat veřejný klíč s entitou a tím vytvořit důvěru v tento klíč. Certifikát je součástí PKI[14], což je pojem pro nespočet činností, technických nástrojů, parametrů, konvencí apod., které se snaží zajišťovat důvěru ve sdílení informací. Certifikáty mají dvě hlavní oblasti využití. První oblastí je internetová komunikace, jejíž součástí jsou šifrované komunikace mezi klientem a serverem. Druhá oblast se vztahuje k elektronickým podpisům.

Ve fyzickém světě jsme zvyklí podepisovat celou řadu dokumentů. S příchodem digitalizace a online prostředí bylo ovšem nutné zavést obdobný prostředek (podpis) ve světě virtuálním. Hledáme tedy obdobu vlastnoručního podpisu. Touto obdobou je elektronický podpis, ale ne tak ledajaký. Konkrétně se jedná o kvalifikovaný elektronický podpis, který platí ve všech členských státech EU. Tyto pojmy je vhodné náležitě definovat. Definice nalezneme v dokumentu *NAŘÍZENÍ EVROPSKÉHO PARLAMENTU A RADY (EU) č. 910/2014 ze dne 23. července 2014 o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu a o zrušení směrnice 1999/93/ES* dále jen **eIDAS**. [29] Rozdíl mezi elektronickým a digitálním podpisem spočívá v tom, že elektronické podpisy definují zákony. Digitální podpisy poskytují příslušnou technologii.

[13] Rivest–Shamir–Adleman
[14] Public Key Infrastructure

Elektronický podpis *„jsou data v elektronické podobě, která jsou připojena k jiným datům v elektronické podobě nebo jsou s nimi logicky spojena a která podepisující osoba používá k podepsání“* [29]

Zaručený elektronický podpis *„je elektronický podpis, který splňuje požadavky stanovené v článku 26“* [29]. Je uvedeno následující:

- *„je jednoznačně spojen s podepisující osobou;“* [29]
- *„umožňuje identifikaci podepisující osoby;“* [29]
- *„je vytvořen pomocí dat pro vytváření elektronických podpisů, která podepisující osoba může s vysokou úrovní důvěry použít pod svou výhradní kontrolou; a“* [29]
- *„je k datům, která jsou tímto podpisem podepsána, připojen takovým způsobem, že je možné zjistit jakoukoliv následnou změnu dat.;“* [29]

Kvalifikovaný elektronický podpis *„je zaručený elektronický podpis, který je vytvořen kvalifikovaným prostředkem pro vytváření elektronických podpisů a který je založen na kvalifikovaném certifikátu pro elektronické podpisy“* [29]

Pro další informace k elektronickým podpisům odkazujeme na přílohy nařízení eIDAS [29], konkrétně přílohy jedna a dvě. Privátní klíče k veřejným klíčům od kvalifikovaných certifikátů by měly být uloženy na QSCD[15]. Těmto zařízením odpovídají HSM (viz kap. 3).

Digitální podpis využívá prostředků asymetrické kryptografie. Jedná se o technické řešení, jak zajistit integritu, identifikaci, autentizaci a nepopiratelnost dokumentů, které určitá entita podepsala a souhlasí s nimi. Digitální podpis nám poskytuje nástroj, jak podpisy vytvářet a zároveň je ověřovat. V principu proces podepisování funguje následovně: Entita podepisuje otisk dokumentu (viz kap. 2.8) svým **privátním klíčem**. Druhá strana tento podpis ověří **veřejným klíčem entity** a získá otisk dokumentu v době podpisu. Tento otisk ověří se současným otiskem dokumentu. Pokud při přenosu nedošlo k chybě, otisky by se měly shodovat.

2.8 Hashovací funkce

Nyní se dostáváme k tématu hashovací funkce, kterou budeme označovat hash(x). Úkolem hashovací funkce je převést libovolný vstup na tzv. ***„digest“*** neboli *„hash value“* neboli *„hash“* neboli otisk. Na funkci jsou kladeny různé nároky. Jednou z požadovaných vlastností je, že malá změna na vstupních bitech, způsobí velkou změnu na bitech výstupních.

Požadujeme, aby funkce vždy vracela stejný *„digest“* na stejné vstupy. Tato vlastnost je stěžejní v případě zajištění integrity dat. Ve zkratce tyto požadavky odpovídají hypotetickému zařízení nazývanému náhodné orákulum. Dále požadujeme, aby bylo výpočetně nemožné dohledat k danému *„digestu“* odpovídající vstup (x) do hashovací funkce, tato vlastnost se nazývá ***„preimage resistance“***. ***„Second preimage resistance“*** označuje vlastnost, kdy nemohu najít k **již zvolenému vstupu x** druhý **odlišný vstup y**, kde by oba tyto vstupy měly po aplikování hashovací funkce stejný *„digest“*. Poslední stěžejní vlastností je ***„collision resistance“***. Tato vlastnost popisuje situaci, kdy bychom byli schopni systematicky vytvářet kolize. Jinými slovy požadujeme, aby bylo výpočetně nemožné **libovolně volit vstupy x a y** a produkovat stejné *„digesty“* pro oba vstupy.

[15] Qualified Signature Creation Device

Nejznámější a zároveň bezpečné hashovací funkce specifikuje standard SHS[16] s označením NIST FIPS PUB 180-4 [23]. Jedná se převážně o hashovací funkce z rodiny SHA[17]-2 (SHA-224. SHA-256, SHA-512 apod.). Číslice za názvem hashovací funkce označují délku výstupního *„digestu“* v bitech.

Jak již bylo zmíněno, hashovací funkce jsou používány pro zajištění integrity dat a dále také pro zajištění autentizace zprávy, tedy zajištění, že zpráva pochází od entity, od které tvrdí, že přichází. Pro tyto účely vznikl MAC[18], což je skupina funkcí, která jako další parametr má klíč. MAC mechanismus založený na hashovací funkci se nazývá HMAC[19] (viz kap. 4.2.1) zmíněný ve standardu PKCS #5 (viz kap. 4.2).

Poslední role hashovacích funkcí, kterou si zde přiblížíme, je ukládání hesel v databázích různého typu aplikací. Potřebujeme-li uživatele autentizovat, využíváme toho, že uživatel nám jednou svoje heslo poskytl. Toto heslo ukládáme v databázi v podobě *„digestu“*. Pro tyto účely se ale nevyužívají výše zmíněné hashovací funkce z rodiny SHA-2, protože pro ně nejsou vhodné. Byly navrženy zejména pro účely podepisování (viz kap. 2.7) a kontroly integrity dat. Musí být rychlé. Hashovací funkce pro zpracování hesel by měly být naopak pomalejší, aby jejich zpracovávání zabralo případnému útočníkovi více času a nebylo snadné tvořit předem připravené databáze *„digestů“*. Mezi vhodné hashovací funkce pro ukládání hesel lze zařadit Bcrypt a ARGON2.

2.9 Shrnutí kapitoly

Cílem této kapitoly bylo přiblížit základní kryptografickou terminologii. Představili jsme základní algoritmy, které se v dnešní době nejčastěji využívají. V kontextu této práce je nejdůležitějším termínem symetrická kryptografie. Pro praktickou část této práce bude podstatná bloková šifra AES v režimu CBC. Dále již tedy budeme využívat pouze označení AES-CBC. Pro praktickou část práce implicitně předpokládáme velikost klíče 256.

Asymetrická kryptografie není pro naši práci stěžejní. V této kapitole jsme zmínili algoritmus RSA, který blíže popíšeme v kapitole zabývající se standardy PKCS (viz kap. 4.1.1).

Asymetrické kryptosystémy nad eliptickými křivkami tzv. ECC[20] vynecháváme, jelikož se jedná o robustní problematiku, která je pro diplomovou práci okrajová. Pro úvod do tématu eliptických křivek odkazujeme na materiály [30] Dr. Mgr. Aleny Gollové. Tyto materiály jsou výborně zpracované a poskytují kvalitní úvod do matematické kryptografie. Tyto materiály využijeme při popisu RSA (viz kap. 4.1.1) a DH[21] (viz kap. 4.1.2).

[16] Secure Hash Standard
[17] Secure Hash Algorithm
[18] Message Authentication Code
[19] The Keyed - Hash Message Authentication Code
[20] Elliptic Curve Cryptography
[21] Diffieho–Hellmanova výměna klíče

Kapitola 3

Hardware Security Module (HSM)

V následující kapitole si představíme obecnou terminologii, která se vztahuje ke kryptografickým modulům. Dále se budeme zabývat fyzickou bezpečností těchto zařízení. Představíme si standard, který stanovuje požadavky na úroveň zabezpečení tím, že specifikuje jednotlivé stupně zabezpečení. Na závěr si blíže představíme architekturu těchto zařízení, ze které nás pro diplomovou práci bude zajímat především komponenta, jejíž prostřednictvím budeme schopni s moduly komunikovat a tedy pseudonymizovat data. Zároveň definujeme veškeré použité pojmy.

V první kapitole teoretické části (viz kap. 1), jsme uvedli, že vhodnou formou pseudonymizace je šifrování. Chceme-li dosáhnout vysokého stupně zabezpečení vzhledem k prostředí, ve kterém jsou prováděny kryptografické operace, využijeme HSM.

HSM je hardwarové zařízení, které může nabývat různých forem. Vždy se jedná o zařízení, které poskytuje fyzicky bezpečné prostředí pro provádění kryptografických operací a uchovávání klíčů. HSM je schopno generovat náhodná čísla, která jsou nezbytnou součástí pro bezpečnost kryptografických operací.

Způsoby generování náhodných čísel jsou založeny na dvou principech tzv. deterministickém nebo nedeterministickém.

- Deterministický přístup využívá tzv. *„seedu“*, což je inicializační hodnota, od které se dále odvíjí generované sekvence bitů. Skupina algoritmů, využívající tento přístup se nazývá PRNG[1]. Tento způsob má ovšem své stinné stránky. Jednou z nich je skutečnost, že hodnoty, které jsou tímto způsobem generovány, jsou po určité době předvídatelné. Představitelem takového generátoru je *„Linear congruential generator“*.
- Nedeterministický princip generování náhodných čísel se nazývá TRNG[2] a vychází z fyzikálních vlastností a principů. Generování bitů probíhá na základě fyzikálního zdroje, který není předvídatelný.

Na následující straně je uveden přehled základní terminologie, která se vztahuje k zařízením typu HSM (viz kap. 3.1). Terminologie vychází ze standardu *Security Requirements for Cryptographic Modules* s označením NIST FIPS PUB 140-2 [31]. V dnešní době již existuje nástupce tohoto standardu, kterým je NIST FIPS PUB 140-3 [32].

[1] Pseudorandom number generator
[2] True random number generator

3.1 Obecná terminologie ke kryptografickým modulům

Kryptografický modul *„je sada hardwaru, softwaru a/nebo firmwaru, která implementuje schválené zabezpečení funkcí (včetně kryptografických algoritmů a generování klíčů) a je obsažen v tzv. „**cryptographic boundary**“.“* [31] (tzv. *„**cryptographic module**“*)

Cryptographic boundary *„je explicitně definovaný souvislý obvod, který stanoví fyzické hranice kryptografického modulu a obsahuje všechny hardwarové, softwarové a/nebo firmwarové komponenty kryptografického modulu.“* [31]

CSP[3] *„je informace související se zabezpečením (např. tajné (**secret**) a soukromé (**private**) kryptografické klíče a ověřovací data, jako jsou hesla a PIN), jejichž zveřejnění nebo úprava může ohrozit zabezpečení kryptografického modulu.“* [31]

Firmware *„jsou programy a datové komponenty kryptografického modulu, které jsou uloženy v hardwaru (např. ROM, PROM, EPROM, EEPROM nebo FLASH) v rámci kryptografické hranice a nelze je během provádění dynamicky zapisovat nebo upravovat.“* [31]

Fyzická ochrana *„je ochrana kryptografického modulu, kryptografických klíčů nebo CSP pomocí fyzických prostředků.“* [31] (tzv. *„**physical protection**“*)

3.2 Terminologie k obsluze kryptografických modulů

Operátor *„je jednotlivec přistupující ke kryptografickému modulu nebo proces (subjekt) fungující jménem jednotlivce, bez ohledu na předpokládanou roli.“* [31] (tzv. *„**operator**“*)

Crypto officer *„je operátor nebo proces (subjekt), jednající jménem operátora, provádějící kryptografickou inicializaci nebo funkce správy“* [31]

Uživatel *„je jednotlivec nebo proces (subjekt) jednající jménem jednotlivce, který přistupuje ke kryptografickému modulu za účelem získání kryptografických služeb.“* [31] (tzv. *„**user**“*)

PIN[4] *„je alfanumerický kód nebo heslo používané k ověření identity“* [31]

Heslo *„je řetězec znaků (písmena, čísla a další symboly) používané k ověření identity nebo k ověření oprávnění k přístupu.“* [31] (tzv. *„**password**“*)

[3] Critical security parameter
[4] Personal Identification Number

3.3 Fyzická bezpečnost

Chceme-li se blíže zaměřit na fyzickou bezpečnost, resp. na požadavky, které jsou kladeny na tato zařízení, je důležité prostudovat již zmíněný standard NIST FIPS PUB 140-2 [31]. Standard specifikuje požadavky na zabezpečení kryptografických modulů, které chrání citlivé informace. Nalezneme zde čtyři úrovně (tzv. *„Levely“*) ochrany. Úrovně jsou specifikovány od nejnižšího stupně zabezpečení (tzv. *„Level 1“*), až po nejvyšší stupeň ochrany (tzv. *„Level 4“*). Pro lepší orientaci v tomto standardu využijeme terminologii použitou v článku Tamper-Indicating Seals: Practices, Problems, and Standards [33] a dále terminologii použitou ve standardu, resp. v jeho glosáři. [31]

3.3.1 Terminologie k fyzické bezpečnosti

Tampering *„je získání neoprávněného přístupu nebo vstupu pro nekalé účely, jako je krádež, pašování, sabotáž, vandalismus, terorismus nebo špionáž“* [33]

Tamper evidence *„je vnější indikace, že byl učiněn pokus o kompromitaci fyzického zabezpečení kryptografického modulu“* [31]

Tamper detection *„je automatická detekce kryptografickým modulem, že byl učiněn pokus ohrozit fyzickou bezpečnost modulu“* [31]

Tamper response *„je automatická akce provedená kryptografickým modulem při detekci neoprávněné manipulace“* [31]

Tamper-indicating seal *„je TID (tamper-indicating device) navržené tak, aby zanechalo nesmazatelně jednoznačný důkaz neoprávněného přístupu nebo vstupu“*[33]

EFP[5] *„znamená použití funkcí na ochranu proti ohrožení bezpečnosti kryptografického modulu v důsledku podmínek prostředí nebo kolísání mimo normální provozní rozsah modulu.“* [31]

EFT[6] *„znamená použití testování k poskytnutí přiměřené jistoty, že bezpečnost kryptografického modulu nebude ohrožena podmínkami prostředí nebo výkyvy mimo normální provozní rozsah modulu.“* [31]

Zeroization *„je způsob vymazání elektronicky uložených dat, kryptografických klíčů a CSPs změnou nebo smazáním obsahu datového úložiště, aby se zabránilo obnově dat“* [31]

Simple power analysis *„je přímá (především vizuální) analýza vzorců prováděných instrukcí (nebo prováděných jednotlivých instrukcí), získaná sledováním změn ve spotřebě elektrické energie kryptografického modulu, za účelem odhalení funkcí a implementací kryptografických algoritmů a následně hodnoty kryptografických klíčů.“* [31]

Lock *„je zařízení, které zdržuje, komplikuje a/nebo odrazuje od neoprávněného vstupu.“*[33]

[5] Environmental failure protection
[6] Environmental failure testing

V následující části si přiblížíme jednotlivé stupně ochrany. Vycházíme z dokumentů zmíněných výše, zejména z NIST FIPS 140-2 [31]. Pro lepší orientaci v následujícím tématu lze použitou terminologii dohledat na předešlých stránkách (viz kap. 3.1 a dále 3.3.1).

Stupeň ochrany 1 (Security Level 1):

- Na kryptografický modul nejsou kladeny požadavky na fyzické mechanismy zabezpečení zařízení.

Stupeň ochrany 2 (Security Level 2):

- Jsou kladeny nároky na fyzické ochranné mechanismy kryptografického modulu. Jedná se o prvky zařazující tzv. *„tamper-evidence“*. Mezi tyto prvky lze zařadit např. ochranné pečetě (tzv. *„tamper-indicating seal“*). Tyto pečetě mohou být závislé na elektrické energii (tzv. *„active-seal“*) nebo být nemusí (tzv. *„passive-seal“*). Dále tato zařízení mohou být vybavena zámky (tzv. *„lock“*). Cílem těchto fyzických mechanismů je ochrana před neoprávněným přístupem. Prvky jsou umisťovány na různá místa zařízení.
- V této úrovni je vyžadována minimálně tzv. *„role-based authentication“* operátora zařízení tzv. *„operator“*).
- **Role-based authentication** *„je proces, který poskytuje ujištění o roli entity pomocí prostředků autentizačního mechanismu, který ověřuje roli entity.“* [34]

Stupeň ochrany 3 (Security Level 3):

- Zde jsou kladeny zvýšené nároky na fyzické zabezpečení v podobě *„tamper response“* a *„tamper detection“*. V případě neautorizovaného přístupu, který by mělo zařízení splňující tento stupeň ochrany s vysokou pravděpodobností detekovat, musí prvky zajišťující *„tamper response“* zajistit *„zeroization“* CSPs.
- V této úrovni je vyžadována tzv. *„identity-based authentication“*.
- **Identity-based authentication** *„je proces, který poskytuje ujištění o identitě entity prostředky autentizačního mechanismu, který ověřuje identitu entity.“* [34]

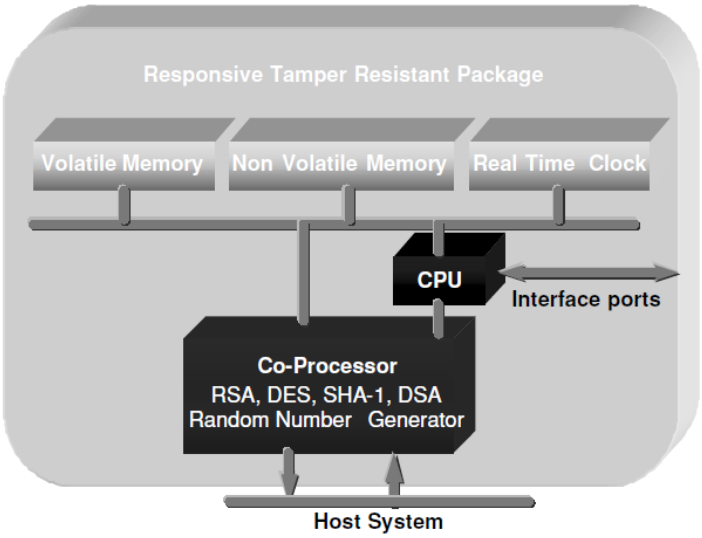
Stupeň ochrany 4 (Security Level 4):

- Nejvyšší stupeň ochrany, který zajišťuje detekci všech neoprávněných přístupu stejných jako ve stupni tři a dále EFP a EFT.
- Pokusíme-li se shrnout EFP a EFT, tak by modul měl být schopen reagovat na vnější podmínky, které mohou ovlivňovat jeho správnou činnost. Např. v případě detekce poklesu napětí v elektrické síti by měl přijmout vhodné bezpečnostní mechanismy, které opět zajistí případnou *„zeroization“* a nebo implementují mechanismy, které zajistí, že kryptografický modul nebude těmito výkyvy ovlivněn.
- Smyslem této ochrany je zabezpečit zařízeni proti sofistikovaným útokům (tzv. *„útoky postrannímí kanály“*). Mezi tyto útoky patří např. SPA[7].
- Tento stupeň ochrany, se hodí do prostředí, které není dobře fyzicky zabezpečené.

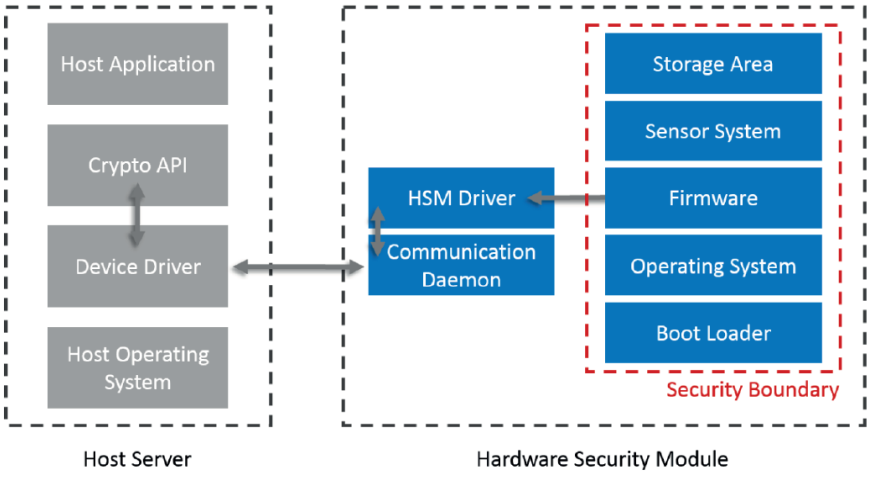
[7] Simple power analysis

3.4 Architektura

Na následujících diagramech jsou zobrazeny základní komponenty HSM. Zatímco první diagram vyobrazuje základní interní komponenty HSM (viz obr. 3.1), druhý diagram (viz obr. 3.2) je již pro naši práci zajímavější. Zachycuje základní komponenty, které jsou nezbytné pro zajištění komunikace s koncovým zařízením. Jednou ze základních komponent je firmware, který jako jediný smí přímo přistupovat k bezpečnostní zóně. Z dále vyobrazených komponent je pro nás důležitá zejména jedna. Jedná se o komponentu Crypto API, která nám bude sloužit pro komunikaci s HSM.



Obrázek 3.1: The internal elements of an HSM [24]



Obrázek 3.2: HSM design [35]

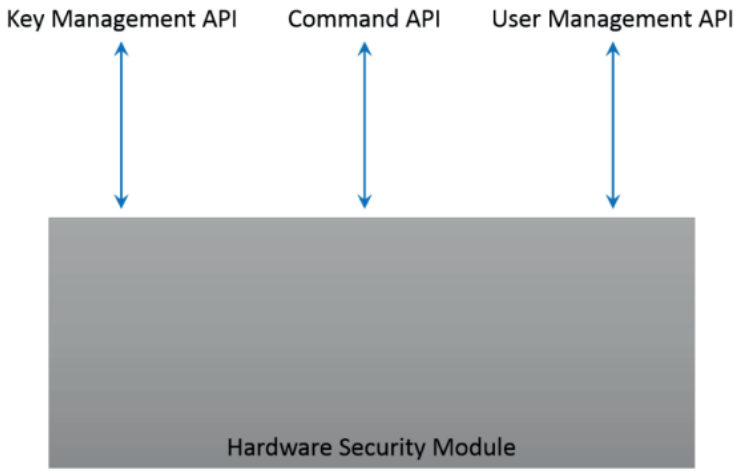
3.5 Crypto API

Chceme-li komunikovat s HSM, máme na výběr z několika možných (tzv. *„interfaců“* neboli API neboli rozhraní).

API *„je sada podprogramů, které mohou aplikační programy používat k vyžádání a provedení služeb nižší úrovně prováděných operačním systémem.“* [20]

Pravděpodobně nejznámějším představitelem je PKCS #11 (viz kap. 4.3). Dalšími jsou např. JCA[8]/JCE[9] (viz kap. 5.2.1), případně CXI[10] (viz kap. 5.2.1).

Posledním bodem, který je v souvislosti s rozhraními vhodné zmínit, je logické rozdělení rozhraní z pohledu HSM. Na následujícím diagramu (viz obr. 3.3) můžeme pozorovat tři rozhraní. Key Management API, jehož primární zodpovědností jsou administrativní funkce vztahující se ke klíčům. Druhým rozhraním je Command API, které zprostředkovává samotné kryptografické operace. Posledním rozhraním je User Management API, to zajišťuje vytváření uživatelů a jejich správu. [35]



Obrázek 3.3: Logical software interfaces of an HSM [35]

3.6 Shrnutí kapitoly

Tato kapitola nám posloužila jako tzv. *„high-level“* pohled na HSM, tedy jako teoretický a terminologický pohled. V praktické části (viz kap. 6.11.1) se již budeme věnovat konkrétnímu HSM od konkrétního výrobce. Jedná se o zařízení, které bylo využito pro praktickou část práce v kombinaci s různými Crypto API. Budeme uvažovat i o reálné bezpečnosti těchto zařízení v praxi (viz kap. 6.13).

[8] Java Cryptography Architecture
[9] Java Cryptography Extension
[10] Cryptographic eXtended services Interface

Kapitola 4

Public Key Cryptography Standards (PKCS)

PKCS je sada standardů, které vznikly v devadesátých letech minulého století. Za jejím vznikem stojí společnost RSA Laboratories. V průběhu let byly tyto standardy začleněny mimo jiné i do RFC dokumentů. V původní specifikaci bylo těchto standardů definováno 15. Některé z nich byly dále spojeny dohromady. V kapitole zmíníme několik z těchto standardů a přiblížíme si známe algoritmy, které jsou v nich použity. Vynecháme standardy, které se vztahují k certifikátům, protože jsou pro téma této práce nepodstatné.

Účelem RFC dokumentů je snaha zavést taková doporučení, aby všichni, kdo se rozhodnou implementovat dané protokoly, algoritmy apod. následovali stejná pravidla. Cílem tedy je unifikovat přístupy k implementacím.

Další dokumenty, které nám budou sloužit pro účely definic, byly publikovány pod (NIST). Tento institut spadá v dnešní době pod Ministerstvo obchodu USA.

4.1 PKCS #1: RSA Cryptography Specifications

Tento standard je aktuálně zakotven v RFC č. 8017 [36], Celým názvem *PKCS #1: RSA Cryptography Specifications Version 2.2* dále jen PKCS #1 z listopadu 2016. Zabývá se sadou doporučení, jak přistupovat k implementaci algoritmu RSA. Jak již bylo zmíněno dříve (viz kap. 2), RSA spadá mezi asymetrické šifry, a zde si ji blíže představíme. Pro účely seznámení budeme používat notaci ze standardu PKCS #1. Standard specifikuje čtyři kryptografická primitiva: šifrování, dešifrování, podepisování a ověřování. Mezi šifrovací primitivum spadá RSAEP[1], mezi dešifrovací pak RSADP[2]. Mezi podpisová, resp. ověřovací, patří RSASP1[3], resp. RSAVP1[4]. Standardy PKCS #2 a PKCS #4 byly v průběhu let začleněny do standardu PKCS #1.

[1] RSA Encryption Primitive
[2] RSA Decryption Primitive
[3] RSA Signature Primitive, version 1
[4] RSA Verification Primitive, version 1

4.1.1 RSA

Bezpečnost RSA algoritmus se opírá o problém, který se v anglické terminologii nazývá IFP[5]. Mějme složené číslo N, nalezněme čísla p a q, ze kterých se číslo skládá. Tento problém se uplatní při vytváření modulu (n), ze kterého se následně vytvoří RSA klíče. Třída složitosti je zatím neznámá.

K vytvoření RSA klíčů je nutné vygenerovat dvě různá a dostatečně velká prvočísla (p a q). Tato prvočísla se mezi sebou vynásobí a vznikne výše zmíněný modul (n). Při výpočtu modulu máme k dispozici prvočísla p a q, a tedy můžeme snadno dopočítat Eulerovu funkci φ(n). Eulerova funkce je definována jako počet čísel nesoudělných s čísly od nuly až do n - 1. Dále platí, že v případě, kdy jsou čísla p a q prvočísla, tak $\varphi(n) = (p-1)(q-1)$). Toto poslední tvrzení je podstatné pro výpočet privátního exponentu (d). Privátní exponent se počítá v okruhu Z_{φ}(n) jako inverze ke zvolenému veřejnému exponentu (e). Veřejný exponent (d) se volí tak, aby byl nesoudělný s φ(n). Musí tedy platit, že $\gcd(e, \varphi(n)) = 1$.

Výše zmíněným postupem nám vznikly dvě hlavní komponenty. První komponentou je veřejný klíč a druhou je soukromý klíč. Tyto dva klíče tvoří dvojici tzv. RSA key pair. Veřejný klíč tzv. (n ,e) je dvojice, která se skládá z modulu (n) a veřejného exponentu (e). Jak již samotný název napovídá, tento klíč je veřejný a tedy není nutné jej uchovávat v tajnosti. Soukromý klíč tzv. (n, d), je dvojice, kterou je vhodné uchovávat na bezpečném místě a nikomu jej neposkytovat.

Šifrování zpráv probíhá následujícím způsobem. Otevřená zpráva (m) se umocní na veřejný exponent (e) v okruhu Z_n, tedy $m^e = c$ v Z_n,. Povšimněme si, že pro účely šifrování potřebujeme pouze modul a veřejný exponent, tzv. veřejný klíč. Nicméně dešifrování výše zmínění zprávy již probíhá za využití modulu a privátního exponentu (e), tzv. soukromého klíče. A to následovně: Máme k dispozici zašifrovaný text (c), který umocníme na privátní exponent (e) a to opět v Z_n, tedy $c^d = a$ v Z_n. Šifrovací transformace se nazývá RSAEP a dešifrovací RSADP (viz kap. 4.1). Aby bylo šifrování a dešifrování bezpečné, je nutné vstupní text randomizovat, tedy jej náhodně doplnit znaky tak, aby byl odolný vůči A-CCA[6]. Tomuto doplnění se říká padding. V PKCS#1 je doporučeným paddingem pro nové aplikace OAEP[7].

[5] Integer Factorization Problem
[6] Adaptive Chosen Ciphertext Attack
[7] Optimal Asymmetric Encryption Padding

4.1.2 Diffieho–Hellmanova výměna klíče

Diffieho–Hellmanova výměna klíče (DH) byla zakotvena ve standardu PKCS #3, který byl nahrazen jinými standardy. Vzhledem k tomu, že šifra RSA je často používána v kombinaci s DH (například v protokolu TLS 1.2, resp. v nabízených kryptografických sadách pro tento protokol), přiblížíme si tuto výměnu. DH je protokol, kterým nelze šifrovat, ani jím nelze podepisovat. V případě, kdy dvě strany potřebují komunikovat a tedy si například předat sdílené tajemství, mohou využít tento protokol.

Bezpečnost DH je založena na matematickém problému, který se v anglické terminologii nazývá DLP[8]. Diskrétní logaritmus je založen na teorii grup. Konkrétně se jedná o cyklické grupy (G) daného řádu (r) a daným generátorem (a). Každý prvek $b \in G$, lze zapsat jako $b = a^x$ pro jediné $x \in Z_r$. *„Toto x se nazývá diskrétní logaritmus o základu a z prvku b v grupě G. $b \in G$. Značí se $dlog_b(a)$.“* [37] *„Předpokládá se, že pro většinu grup je výpočet diskrétního logaritmu exponenciální nebo subexponenciální problém“* [37]

Při výměně sdíleného tajemství si první strana zvolí cyklickou grupu řádu r a dále její generátor a. Provede volbu prvku **x**, tak aby $x \in Z_r$ a provede výpočet prvku $b = a^x$ v grupě G. Informaci o použité grupě G, použitém řádu r a generátoru zašle druhé straně. Tyto informace můžeme nazývat trojicí (G, r, a). Povšimněme si, že v této informaci není zanesen zvolený prvek x. Právě tento prvek (diskrétní logaritmus) nelze bez hrubé síly získat, alespoň prozatím.

Druhá strana zvolí vlastní tajný prvek **y**, tak aby $y \in Z_r$ a dopočítá prvek $c = a^y$, opět v grupě G. První straně zašle dopočítaný prvek c.

Poslední výpočet, který musí obě strany provést, je dopočítání sdíleného tajemství. První strana, resp. druhá strana, spočte $s_P = c^x$ resp. $s_D = b^y$ v grupě G.

Tímto výpočtem obě strany získaly stejnou tajnou informaci $\mathbf{s} = s_P = s_D$. Nyní je zajištěno, že obě strany aktuálně disponují stejným klíčem, který lze použít např. pro symetrické šifrování.

[8] Discrete Logarithm Problem

4.2 PKCS #5: Password-Based Cryptography

Standard je aktuálně zakotven v RFC č. 8018 [38], celým názvem *PKCS #5: Password-Based Cryptography Specification Version 2.1* dále jen PKCS #5, z ledna 2017. Ve standardu nalezneme způsoby, jak z hesel odvozovat klíče. Konkrétně se jedná o tzv. KDF[9].

Pomineme-li správce hesel, tak u většiny hesel se předpokládá, že si je uživatel zapamatuje. Hesla proto často obsahují známá slova a čísla (z hlediska bezpečnosti je toto špatně). Klíče naopak musí být konstruovány tak, aby měly vysokou míru entropie, tzn. aby byly jejich znaky čistě náhodné a nebyl v nich obsažen žádný vzor. Dále jsou na klíče kladeny nároky na přesnou délku. Například šifra AES-256 požaduje klíč o délce 256 bitů. Jinými slovy: potřebujeme z krátkých a zapamatovatelných řetězců tvořit tajné vstupy do kryptosystémů. Každý kryptosystém je tak silný, jak silný je jeho nejslabší článek. V moderních kryptosystémech je velice často nejslabším článkem právě použitý klíč. Z tohoto důvodu je nutné věnovat pozornost KDF. [39]

4.2.1 Key derivation function (KDF)

Jak již samotný název napovídá, funkce odvozuje klíče z předem dané znalosti. V případě tzv. *„password-based“* je touto znalostí heslo. Tyto funkce hrají roli zejména v tzv. *„encryption schemes“*. Standard PKCS #5 specifikuje funkce PBKDF1[10] a PBKDF2[11]. Doporučenou funkci PBKDF2 si přiblížíme.

Password-Based Key Derivation Function 2 (PBKDF2)

Vstupní parametry funkce jsou následující (dodržujeme notaci použitou ve standardu):

- P - heslo
- S - sůl tzv. *„salt“*
- c - počet iterací
- dkLen - požadovaná délka klíče

Mezi volitelné parametry patří pseudonáhodná funkce tzv. PRF[12]. Definici nalezneme v dokumentu *NIST Special Publication 800-135 Revision 1, Recommendation for Existing Application-Specific Key Derivation Functions*, dále jen NIST 800-135 Revision 1. [40]

Dle NIST 800-135 Revision 1 je **pseudonáhodná funkce**:

„funkce, kterou lze použít ke generování výstupu z tzv. random seed a datové proměnné, tak, že výstup je výpočetně k nerozeznání od skutečně náhodného výstupu.“[40]

Pseudonáhodná funkce, která je použita v PBKDF2, se nazývá HMAC. PKCS #5 uvádí následující příklady, kde název za HMAC označuje použitou hashovací funkci: HMAC-SHA-1, HMAC-SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 a SHA-512/256).

[9] Key derivation function
[10] Password-Based Key Derivation Function 1
[11] Password-Based Key Derivation Function 2
[12] Pseudo-Random Functions

HMAC

Pro potřeby kontroly (autentizace, integrity) přenášené informace vznikl tzv. MAC. HMAC je MAC mechanismus, založený na hashovací funkci. HMAC je popsán v RFC dokumentu č. 2104 [41]. Ten byl aktualizován RFC dokumentem č. 6151 [42]. Opět využijeme notaci použitou v RFC dokumentech. **HMAC** je definován následujícím vzorcem [41]:

H(K XOR opad, H(K XOR ipad, text)), kde:

- H - hashovací funkce
- K - tajný klíč
- B - označuje blok 64bajtů
- ipad - řetězec B bajtů *„0x36“*
- opad - řetězec B bajtů *„0x5C“*

rovnici lze přepsat následujícím způsobem:

H(K XOR opad || H(K XOR ipad || text)), kde:

- || - značí zřetězení

Bezpečnost HMAC se odvíjí od použité hashovací funkce.

Vrátíme-li se k funkci PBKDF2, tak počet iterací (c) určuje, kolikrát je vykonán HMAC. HMAC se aplikuje na zadané heslo a použitý salt. V dalších iteracích dochází k aplikaci HMAC na heslo a výstup z předchozí iterace. Doporučený počet iterací je v řádu statisíců. [43]

4.2.2 PKCS #5 padding

Posledním bodem, který je nezbytné zmínit, je tzv. padding. Ten je například nezbytný v AES-CBC. Šifrované bloky mají fixní velikost. Poslední blok je nutné doplnit tak, aby měl přesnou velikost vyžadovanou šifrovacím algoritmem. Takové doplnění lze učinit několik způsoby. Pravděpodobně tím nejznámějším je PKCS #5 padding, který je definován pro osmi bytové bloky. Zde si dovolíme odkázat přímo do standardu, konkrétně do podkapitoly *6.1.1. PBES1 Encryption Operation* [38]. Pokud bychom chtěli toto schéma zobecnit, stačilo by využít PKCS #7 padding, který je zobecněnou variantou PKCS #5 *„paddingu“*. Čistě teoreticky by tedy nemělo být možné aplikovat PKCS #5 padding na AES-CBC, protože zde jsou velikosti bloků šestnácti bytové.

Je vhodné poznamenat, že kryptografické knihovny tuto teorii respektovat nemusí. Příkladem je kryptografické API JCA/JCE. V případě, že bychom chtěli využít PKCS #7 padding v kombinaci s AES-CBC, program vyhodí výjimku (*„NoSuchPaddingException“*). Použijeme-li PKCS #5 padding, je vše v pořádku.

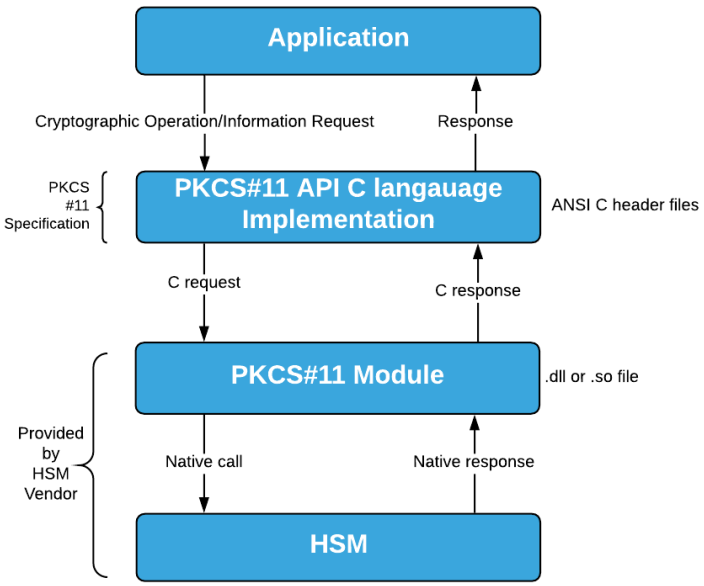
4.3 PKCS #11

PKCS #11 je nejdůležitějším standardem, kterým se v této práci budeme zabývat, protože nám poskytuje způsob, jak komunikovat z aplikace s kryptografickým zařízením. V kontextu této práce se jedná o HSM. Standard specifikuje interface, tzv. API, které se nazývá Cryptoki[13]. Zařízení, se kterým aplikace komunikuje pomocí Cryptoki, se nazývá *„cryptographic token“*. O vývoj tohoto standardu se v dnešní době stará nezisková organizace OASIS Open[14], která mimo jiné vyvíjí i další kryptografické standardy. PKCS #11 je k dispozici ve verzi 2.40 a je rozložen do několika dokumentů, z nichž definujeme pojmy klíčové pro další interakci s API.

Pro naše účely jsou nejdůležitější následující dokumenty:

- Cryptographic Token Interface Usage Guide Version 2.40 [44]
- PKCS 11 Cryptographic Token Interface Base Specification Version 2.40 [45]

Hlavičkové soubory pro API jsou specifikovány v jazyce C. Samotnou implementaci řeší výrobce kryptografického zařízení. Principem tohoto API je snaha unifikovat přístupy ke kryptografickému zařízení. Pro lepší ilustraci je k dispozici následující diagram (viz obr. 4.1).

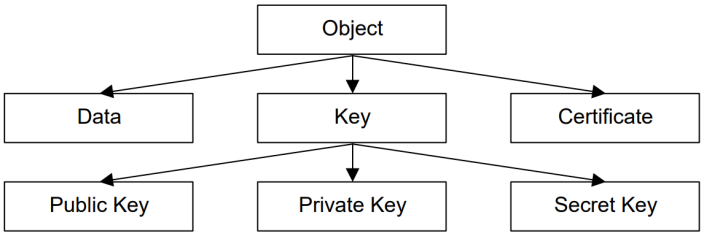


Obrázek 4.1: How PKCS #11 works [46]

[13] Cryptographic token interface
[14] Organization for the Advancement of Structured Information Standards

Logický pohled na kryptografické zařízení se nazývá *„Token“*. Ten je schopen ukládat objekty a provádět kryptografické operace. Logické spojení mezi aplikací a kryptografickým zařízením se nazývá *„Session“*. Definuje spojení mezi aplikací a *„tokenem“*. Aplikace následně přes toto spojení s *„tokenem“* komunikuje. Podle potřeby lze definovat, zdali má *„session“* sloužit pro čtení i zápis nebo pouze pro čtení (viz dále). *„Slot“* je logická čtečka, která obsahuje *„token“*. Pro lepší ilustraci terminologie si lze představit čtečku karet, kde čtečka je *„slot“* a karta je *„token“*.

Jak již bylo zmíněno, *„token“* je schopen ukládat objekty. Standard definuje tři typy objektů (Data, Key, Certificate) - viz následující *„class“* diagram ze standardu (viz obr. 4.2):



Obrázek 4.2: Object hierarchy [44]

S těmito objekty budeme dále pracovat. *„Data“* je objekt, který přestavuje zpracovávaná data poskytnutá aplikací. *„Public Key“* a *„Private Key“* reprezentují klíče k asymetrické kryptografii, oproti tomu *„Secret Key“* souvisí s kryptografií symetrickou.

Z pohledu životního cyklu objektu rozlišujeme:

- *„Token objects“* dále jen *„token objekty“* - jsou dostupné všem, kteří mají přístup k *„tokenu“*.
- *„Session objects“* dále jen*„session objekty“* - jsou vázány na danou *„session“*. Dojde-li k zániku *„session“*, objekty zanikají též.

V závislosti na implementaci může být pro určité operace prováděné na *„tokenu“* nutná autorizace. Jedním ze způsobů je **PIN**. Standard definuje dva typy uživatelů, tzv . **SO**[15] a *„normal user“*, dále jen **běžný uživatel**. SO uživatel inicializuje *„token“*. Další z jeho odpovědností je nastavit přístupový PIN pro běžného uživatele. Po přihlášení, smí běžný uživatel provádět veškeré kryptografické operace poskytované (implementované) na daném *„tokenu“*.

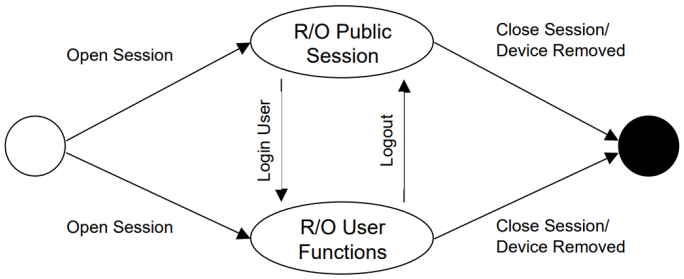
Abychom lépe pochopili způsob komunikace s *„tokenem“*, je vhodné detailněji popsat způsoby komunikace přes *„session“*, dále jen spojení. Aplikace a spojení je v relaci m:n. Tedy jedna aplikace smí navázat 1 až n spojení a spojení může být vytvořeno s 1 až m aplikacemi. Jak již bylo uvedeno výše, spojení může být typu, tzv. ***„R/W“***, které slouží pro čtení a zápis *„token objektů“* nebo pouze pro jejich čtení tzv. ***„R/O“***. Dále rozdělujeme spojení v závislosti na typu přihlášeného uživatele.

[15] A Security Officer user

V rámci *„R/O“*, kdy aplikace otevře spojení a uživatel není přihlášen, je spojení ve stavu ***„R/O Public Session“***. Po přihlášení běžného uživatele dojde ke změně spojení do stavu ***„R/O User Functions“***. Rozdíl mezi těmito dvěma spojeními je následující:

- *„R/O Public Session“* - přístup ke **čtení veřejných** *„token objektů“* a ke **čtení a zápisu veřejných** *„session objektů“*.
- *„R/O User Functions“* - přístup ke **čtení všech** *„token objektů“* a ke **čtení a zápisu soukromých a veřejných** *„session objektů“*.

Pro názornější ilustraci, přikládáme stavový diagram ze standardu (viz obr. 4.3) [44]:

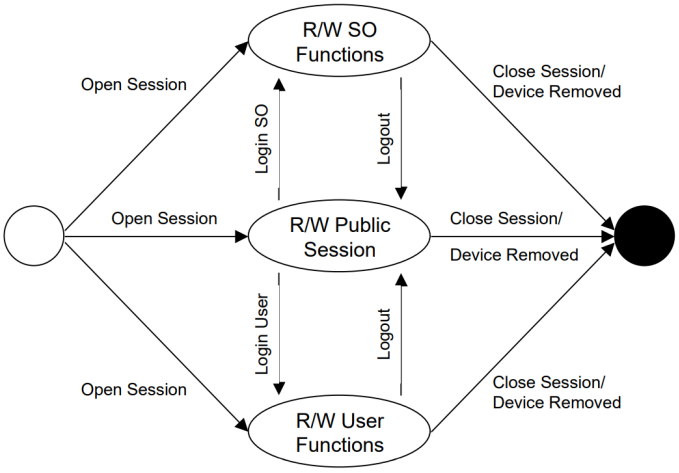


Obrázek 4.3: Read-Only Session States [44]

„R/W“ spojení obsahuje následující stavy:

- *„R/W Public Session“* - aplikace získá **čtecí a zápisové** právo na všechny **veřejné** objekty.
- *„R/W SO Functions“* - jedná se o obdobnou situaci jako v předchozím případě, ale SO uživatel je přihlášen a má schopnost **nastavit PIN běžnému uživateli**.
- *„R/W User Functions“* - běžný uživatel je přihlášen a aplikace získala **čtecí a zápisové** právo na **všechny** objekty.

Opět přikládáme stavový diagram ze standardu (viz obr. 4.4) [44]:



Obrázek 4.4: Read/Write Session States [44]

4.4 Uzavření teoretické části práce

Závěr teoretické části byl věnován standardům PKCS. Na ty je nutné nahlížet s určitým odstupem. Nejedná se o ucelené dokumenty, ale spíše o historické členění, které se rozpadlo. Dohledali jsme některé z částí, které obsahují/obsahovaly algoritmy, případně technologie, které jsou podstatné nebo zajímavé pro naše záměry.

Shrneme-li teoretickou část práce, jednalo se nám především o základní legislativní vhled do problematiky zpracování osobních údajů. Dále jsme pro účely pseudonymizace definovali několik podstatných pojmů z oblasti kryptografie. V předposlední kapitole jsme zmínili obecnou terminologii k HSM. Zakončení obsahovalo detailnější vhled do standardu PKCS #11.

Touto kapitolou jsme uzavřeli teoretickou část práce. Následující kapitoly se již budou věnovat praktické části, která bude uvozena použitými technologiemi a formulací našeho cíle (hypotézy).

Kapitola 5

Úvod do praktické části

Následující kapitola nás uvede do praktické části práce. Jak již ze zadání vyplývá, zabýváme se pseudonymizací dat většího rozsahu. Pro tento úkol si vypůjčíme pojem, resp. oblast/prostředí/svět Big Data. Vzhledem k tomu, že popsat tuto oblast překračuje rámec této práce, přiblížíme pouze nezbytné technologie. Následovat budou technologie, které se bezprostředně netýkají prostředí Big Data, ale jsou důležité pro pochopení testovacího prostředí (viz kap. 6.2)

Spíše než obecnou definici lze dohledat charakteristiky, vlastnosti, tzv. **8V**, kterými se toto prostředí vyznačuje. Pro jejich popis odkazujeme na článek [47].

Zde se budeme věnovat jen určité výseči, která je, dle mého názoru, reprezentativním příkladem reálného prostředí. Výseč je následující:

- Vzhledem k povaze dat nelze využít relační databáze.
- Pracujeme řádově se stovkami megabytů až jednotkami gigabytů strukturovaných dat (CSV[1], Parquet[2]), která jsou uložena na distribuovaném úložišti (viz kap. 5.2.2).
- Pro základní analytické zpracování dat je využit framework Spark (viz kap. 5.2.3). Zpracování probíhá na principu práce s tzv. *„DataFrame“*, který lze připodobnit k tabulce. Dle citlivosti je potřeba vybrané sloupce před analytickým zpracováním **pseudonymizovat**. Používané programovací jazyky ve frameworku Spark jsou: Java, Python a Scala.
- Uvedené nástroje jsou součástí systému Cloudera CDP.

Nebude-li uvedeno jinak, zmíníme-li dále v práci pojem Big Data, odkazujeme implicitně na výše popsanou výseč.

[1] Comma Separated Values
[2] Apache Parquet

5.1 Formulování hypotézy

Nyní již máme veškeré informace nezbytné pro formulování hypotézy. Budeme tedy předpokládat výše zmíněnou výseč a s tím související použité technologie.

Co zde ovšem nepředpokládáme, je forma použité pseudonymizace. Nabízí se tedy otázka, zda-li by nebylo možné umístit zde HSM pro zavedení pseudonymizace. Vzhledem k jejich vlastnostem by mělo dojít ke zlepšení bezpečnosti dat.

Datovému analytikovi by mělo být umožněno pracovat s daty obdobným způsobem, jako doposud. S tím rozdílem, že sloupce, které nepotřebuje a které mají být z legislativních a bezpečnostních důvodu nepřístupné, budou pseudonymizací skryty.

Zajišťujeme tedy přístup k datům na úrovni sloupců a pokusíme se integrovat způsob zabezpečení na úrovni, která je co nejblíže přístupu, který datový analytik zná. To v sobě zahrnuje i používané nástroje. Požadujeme, aby bylo možné pseudonymizovat pouze určité sloupce a další ponechat beze změny.

Hypotéza

Je za výše uvedených okolností pro účely pseudonymizace daných sloupců vhodné, resp. časově akceptovatelné, využít HSM pro provádění kryptografických operací?

Časové důvody nemusí být vždy hlavním měřítkem. Mějme ovšem na paměti skutečnost, že je nezbytné brát v potaz úměrnost ostatních operací (prováděných nad daty) vzhledem k pseudonymizačním. Pokud bychom tak neučinili, využití navrženého řešení by bylo nepraktické. Zejména je pak důležité poznamenat, že škálovatelnost je důležitá vlastnost Big Data prostředí.

Naším cílem bude tuto hypotézu potvrdit, případě zamítnout.

5.2 Technologie

V této podkapitole přiblížíme technologie, které byly zmíněny v úvodu kapitoly. První část bude věnována API pro komunikaci s HSM. API budeme muset integrovat do námi definované výseče, abychom byli schopni využít HSM.

Stručný popis distribuovaného úložiště HDFS[3] a technologie pro zpracování dat Spark nám mají přiblížit způsob, jak tyto technologie pracují.

Podkapitolu završíme poznámkou o Cloudera CDP a dále pak o formátu dat. Tím završíme technologické ohraničení našeho prostředí.

[3]Hadoop Distributed File System

5.2.1 API pro komunikaci s HSM

V práci jsme se rozhodli vyzkoušet tři CryptoAPI (viz kap. 3.5). Postupovali jsme podle jejich dostupnosti v jednotlivých programovacích jazycích, o které se zajímáme (viz kap. 5). V předchozí kapitole (viz kap. 4.3) jsme si detailněji popsali nejznámější a univerzální rozhraní Cryptoki (PKCS #11). Vedle toho existuje Java framework, který se nazývá JCA. Součástí tohoto frameworku je rozšíření JCE, které obsahuje silnější kryptografické algoritmy a historicky bylo od frameworku JCA odděleno [48]. Posledním rozhraním je proprietární rozhraní CXI, které je k dispozici pro HSM od společnosti Utimaco, jejíž fyzické zařízení a simulátor v této práci využíváme (viz kap. 6.11.1).

PKCS #11

Vhledem ke skutečnosti, že PKCS #11 je napsáno v programovacím jazyce C, potřebujeme tzv. *„wrappery“*. Jedná se o knihovny, které budou specifické pro daný jazyk. Pro programovací jazyk Java využijeme následující *„wrapper“* [49]. Pro jazyk Python pak tento [50]. Pro seznámení/práci s těmito *„wrappery“* a HSM, jsme využili sadu tutoriálů dostupných na internetu [51] [46] [52] [53]. Vybrané ukázkové kódy z tutoriálů jsou plně začleněny do samotných, námi vytvořených modulů (viz kap. 6.7.1).

JCA/JCE

Na JCA lze nahlížet jako na sadu *„interfaců“*, abstraktních tříd, apod. (tzv. SPIs[4]). Framework je navržen s důrazem na oddělení implementace od abstrakce s možností jednoduchého zaměnění použitých implementací daných algoritmů. Nejpodstatnější abstraktní třídou je tzv. *„Provider“*. Právě tato abstraktní třída umožňuje výrobci HSM poskytnout implementaci daných abstraktních tříd a algoritmů. Důsledkem tohoto návrhu je možnost operativně zaměnit jednotlivé poskytovatele s minimem úprav. Jediné, co potřebujeme vyřešit, je volba daného providera. Příkladem providera může být např. SunPKCS11, který nám umožňuje prostřednictví JCA využívat objekty vytvořené pomocí rozhraní PKCS #11. Výrobce HSM si zpravidla vytváří vlastního providera, jehož pomocí lze přistupovat k dalším objektům, které jsou např. od PKCS #11 odděleny. Vždy záleží na konkrétní implementaci abstraktní třídy *„Provider“*. V případě námi testovaného zřízení jde o *„CryptoServerJCE providera“*. Tento koncept je výhodný. V našem případě nám rozhraní umožnilo vyzkoušet prakticky identický kód. V jenom případě v pozadí provádělo kryptografické operace HSM. V druhém případě pak lokální počítač, resp. procesor.

CXI

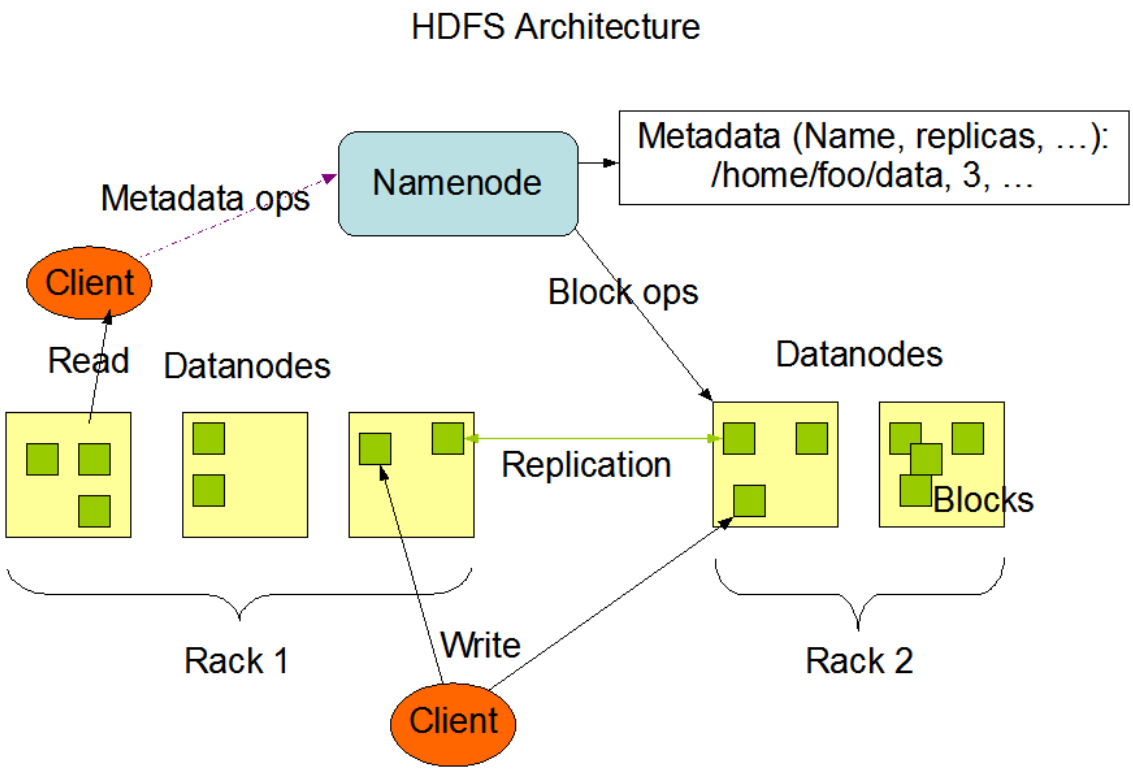
Jak již bylo řečeno, CXI je proprietární API společnosti Utimaco, jehož prostřednictvím lze spravovat i ostatní rozhraní. Obecně lze říci, že proprietární řešení mají tu výhodu, že jsou vytvořena na míru daným zařízením. Mohou např. obsahovat pokročilé funkce jako je Load Balancing. Naopak jejich nevýhoda je spojena s nemožností jednoduše přejít na jiná zařízení od jiných výrobců. Pro Javu je k dispozici *„CryptoServerCXI“* knihovna.

[4] Service Provider Interfaces

5.2.2 HDFS

HDFS je distribuovaný systém ukládání dat. Jeho hlavní výhodou je škálovatelnost. Byl navržen tak, aby jej bylo možné vybudovat z běžně dostupného hardwaru. V principu jej lze vybudovat na obyčejných počítačích. Distribuovaný cluster se může skládat z několika tisíc tzv *„nodů“*. Ze statistických vlastností distribuovaného systému musíme počítat s tím, že některé hardwarové komponenty budou v clusteru poruchové. U každého souboru je možné nastavit tzv. replikační faktor. Ten udává, v kolika kopiích je daný soubor v clusteru dostupný. V běžné konfiguraci je toto číslo rovno třem. HDFS byl primárně navržen pro velké datové sady v řádu gigabytů až terabytů dat. Zároveň je architektura postavena na tom, že budou převládat čtecí operace nad zápisovými, tzv. *„Write-once, read-many“*. Soubory jsou rozděleny na bloky. Obvykle se jejich velikosti rovná 64 MB nebo 128 MB. [54]

HDFS využívá master-slave architekturu. Master je tzv. *„NameNode“*. *„Slaves“* jsou tzv. *„DataNodes“*. Z pohledu uživatele práce se soubory připomíná standardní operace s linuxovým *„filesystémem“*. Mezi jednu z činností, kterou *„NameNode“* zajišťuje, je samotné zprostředkování interakce s *„filesystémem“*. *„DataNodes“* pak obsluhují čtecí a zápisové operace, protože sami slouží k ukládání bloků. Architektura je vyobrazena níže (viz obr. 5.1).



Obrázek 5.1: HDFS Architecture [54]

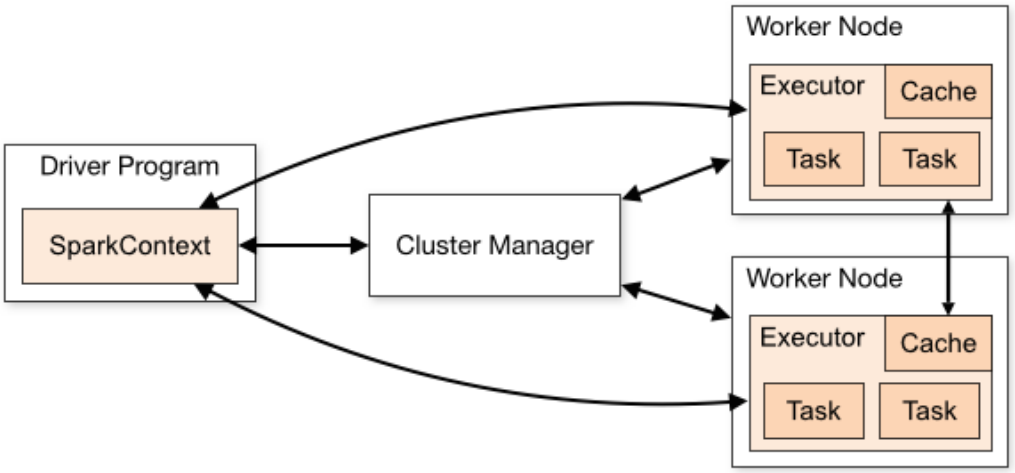
5.2.3 Spark

Spark je engine/framework navržený pro zpracování dat velkého rozsahu (až petabyty). Na nejvyšší úrovni abstrakce funguje následovně.

Na abstrakci/struktuře tzv. *„DataFrame“* (zjednodušeně tabulce) uživatel popisuje tzv. transformace a akce, které mají být na struktuře provedeny. Transformace jsou popisem operací, které mají být vykonány. V principu je to plán, který Sparku říká, jaké operace budeme požadovat s daty provést. Existuje komponenta Sparku tzv. *„Catalyst Optimizer“*, která takový plán dokáže optimalizovat. Akce pak startuje celý výpočet. Příkladem akce je funkce write, která zapíše *„DataFrame“* na disk. Dalším příkladem akce je *„collect“*. Ta v principu umožňuje iteraci na úrovni řádků. Její nevýhoda ovšem spočívá v tom, že iteraci musí provést tzv. *„driver“* a vše se musí nahrát do jeho operační paměti. To v principu vede k chybě. Pro nás nejpodstatnější a nejvíce využívanou transformací bude tzv. UDF[5]. Ta nám umožňuje provádět operace na úrovni řádků bez nutnosti přes ně iterovat. Je vhodné poznamenat, že samotná transformace UDF je náročná, protože ji *„Catalyst Optimizer“* vnímá jako tzv. *„black box“*, který neumí optimalizovat. Akce vytváří tzv. *„Spark Job“*, ten se dále rozpadá na tzv. *„stages“*, které jsou na sobě závislé a následně se rozpadají na tzv. *„tasks“*.

Z pohledu architektury se opět jedná o master-slave. Master je tzv. *„driver“*. Mezi jeho úkoly patří převést akce na *„Spark Joby“* a zajistit přidělení odpovídajících zdrojů u tzv. *„Cluster Manageru“*, kterým je např. YARN[6]. *„Executoři“* jsou zodpovědní za zpracovávání samostatných *„tasků“*. Pro lepší ilustraci (viz obr. 5.2) odkazujeme na oficiální dokumentaci [55]. V ní lze také dohledat dodatečné vysvětlení zmíněných pojmů.

Posledním důležitým pojmem je tzv. *„deploy mode“*, který odlišuje, kde jsou nasazeny komponenty z architektury. Rozlišujeme tři módy (local, client a cluster). V *local* módu běží vše na jednom stroji. V *client* módu je *driver* umístěn na stroji, který aplikaci odeslal. Odeslání aplikace probíhá prostřednictvím scriptu tzv. *„spark-submit“*. Samotní *„executoři“* jsou již dostupné na clusteru. V cluster módu běží na clusteru i samotný *driver*.

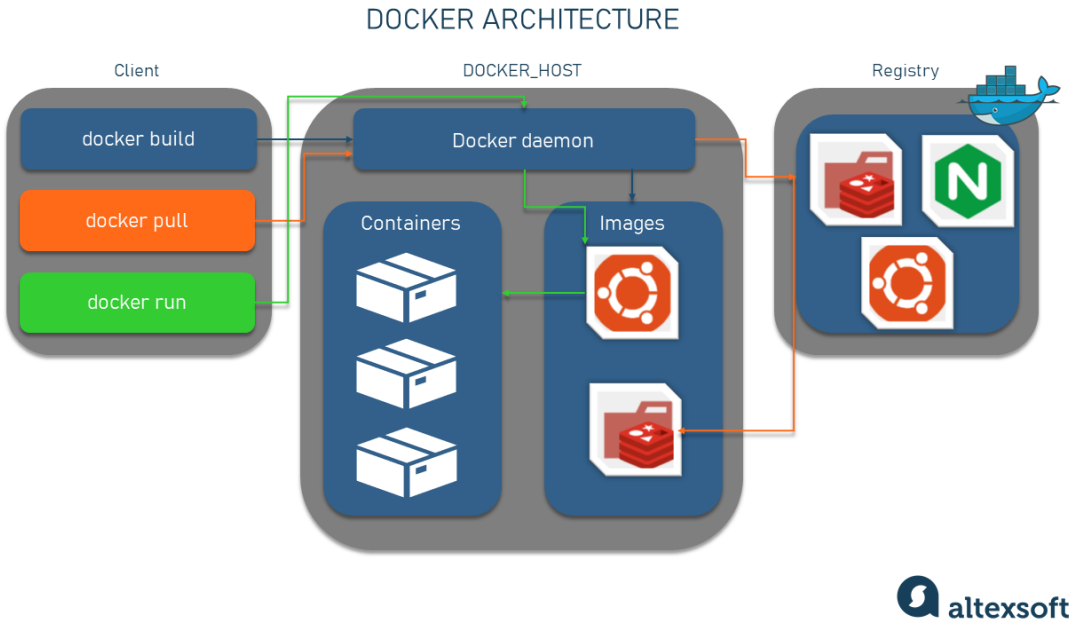


Obrázek 5.2: Cluster Mode Overview [55]

[5]User Defined Functions
[6]Yet Another Resource Negotiator

5.2.4 Docker

Docker je populární technologie, která připomíná standardní virtualizaci. Oproti standardní virtualizaci ale nedochází k virtualizaci samotného hardwaru, nýbrž operačního systému. Stěžejním pojmem jsou kontejnery. Jedná se o samostatné procesy, které sdílejí jádro operačního systému. Kontejnery běží na tzv. Docker Engine. To nám umožňuje vytvářet malé kontejnery (řádově desítky až stovky megabytů) oproti standardní virtualizaci (řádově gigabyty). *„Docker image“* je de fakto sada všech požadavků (technologie, nástroje, služby), které mají být v Docker kontejneru k dispozici. [56] Specifikace *„Docker imagů“* probíhá na základě tzv. *„Dockerfile“*. *„Dockerfile“* je v principu soubor, ve kterém lze jednoduše naše požadavky popsat. Pro detailnější informace doporučujeme článek [57], ze kterého pochází náčrt architektury (viz obr. 5.3).



Obrázek 5.3: Docker Architecture [57]

Kontejner

Kontejner je samostatná jednotka zahrnující veškeré programové vybavení, které je zapotřebí pro běh kódu. Mezi základní konfigurace patří:

- tzv. *„environment variables“* - vstupní proměnné
- tzv. *„networking“* - zajištění konektivity mezi kontejnery
- tzv. *„volumes“* - persistentní paměťové prostory, které lze vzájemně sdílet

Docker Compose

Pro usnadnění nasazování kontejnerů vznikl Docker Compose. Ten zajišťuje, mimo jiné, propojení kontejnerů na základě jednoho konfiguračního souboru, ve kterém lze definovat veškerou konfiguraci. Zjednodušeně řečeno: tvoří z několika kontejnerů jeden koherentní celek.

5.2.5 Cloudera CDP

Cloudera CDP je tzv. on-premise řešení, které obsahuje skupinu softwaru, která je pro svoji popularitu hojně využívána v reálném prostředí Big Data. Jejím hlavním přínosem je zajištění konfigurace a vzájemné integraci těchto nástrojů, včetně jejich správy a nasazení na cluster. Samotná správa nástrojů je pak pro administrátora výrazně snažší. Pro účely této práce je důležité zmínit, že mezi nástroje, které v základní konfiguraci obsahuje, patří HDFS (viz kap. 5.2.2), Spark (viz kap. 5.2.3) a YARN. Ten, resp. jeho komponenta, *„Resource Manager“*, zajišťuje přidělování zdrojů úlohám v clusteru. Clouderu lze tedy vnímat jako prostředí, které obsahuje vše potřebné pro základní zpracování dat ve světě Big Data.

5.2.6 Formáty dat

Formátů dat existuje nespočet. Pro účely této práce jsou nezbytné hlavně dva.

Tím první je velice známý CSV. Jedná se o tzv. *„row-oriented“* formát. To v praxi znamená, že záznamy v řádku jsou na disku uloženy pospolu. Oproti této variantě existují tzv. *„column-oriented“* formáty. Na disku jsou pospolu uloženy hodnoty sloupců. Jedním, velice oblíbeným je Parquet. Analytické dotazy v případě použití formátu Parquet jsou rychlejší. Formát je binární a velice efektivní. Efektivita spočívá především ve velikosti výsledných souborů [58]. Je to doporučený formát při zpracování dat ve Sparku.

5.3 Shrnutí úvodu do praktické části

Účelem této kapitoly bylo ohraničit praktickou část práce specifickými podmínkami, které popisují reálné zobecněné prostředí. Tyto zobecněné podmínky v sobě implicitně nesou nutnost jistého technického zázemí:

- přítomnost distribuovaného úložiště
- funkční a korektně fungující Spark
- dostatek výpočetního výkonu
- v neposlední řadě samotné HSM

Vše výše uvedené musíme v rámci diplomové práce zajistit. Tím se dostáváme k předposlední kapitole práce (viz kap. 6).

Kapitola 6

Pseudonymizační prostředí

V úvodu této rozsáhlejší kapitoly se zaměříme na motivaci navrženého řešení a popíšeme si jeho vlastnosti. Z motivace vyplynou přesné důvody, proč jsme v práci postupovali daným směrem. Nezapomeneme zmínit jednotlivé technologie, které jsme v práci využili a také hardwarové a softwarové zázemí. Představíme si reálné HSM, včetně jeho základní konfigurace a praktických rizik, které s využíváním takového zařízení souvisí. Kapitola plně navazuje na předešlou část práce, zabývající se použitými technologiemi. Technologie zde zmíněné, nejsou již dále popisovány, ale jsou plně začleněny do popisu vytvořeného prostředí. Pro jejich detailnější popis odkazujeme na předešlou kapitolu (viz kap. 5.2). Občas se v popisu setkáme s konstatováním, že jsme využili technologii/nástroj/službu, se kterou jsme měli zkušenost v rámci bakalářské práce [59]. Vzhledem ke skutečnosti, že dané nástroje jsou pro diplomovou práci důležité pouze okrajově, nepovažuji dále za podstatné jejich volbu důkladněji popisovat.

Před samotným úvodem do praktické části práce, je vhodné definovat **terminologii**, kterou na dalších stranách budeme používat.

- Kontejner - v kontextu této kapitoly jde o Docker kontejner (viz kap. 5.2).
- Mikroslužba - je nasazena v kontejneru a odpovídá vyvíjené softwarové komponentě pro účely diplomové práce. Pro svoji správnou funkci může využívat služeb několika kontejnerů.
- Modul začleněný do backendu - část kódu, kterou lze jednoduše vyčlenit z backendu a po drobných úpravách a v závislosti na použitém programovacím jazyce pustit samostatně přes tzv. *„spark-submit“*.
- Samostatný modul - část kódu, která již byla vyčleněna z backendu aplikace.
- Pseudonymizační modul/modul - v závislosti na kontextu je myšlen samostatný modul, případně modul začleněný do projektu. V závislosti na návrhu vždy provádí šifrovací nebo hashovací operaci. Šifrovací operace je vždy prováděna oproti HSM. V závislosti na prostředí se jedná o reálné HSM nebo simulátor.
- Cluster - sada čtyř až pěti počítačů (každý o dostupné kapacitě 64 GB paměti RAM), na kterých je dostupné Big Data prostředí.
- Big Data prostředí - nastavené Cloudera prostředí, pro naše účely je důležité HDFS, Spark a YARN.

6.1 Plán realizace, motivace

Na začátku září 2022 jsem byl postaven před rozhodnutí, jak k dané práci přistoupit. Přemýšlel jsem především o hardwarovém a softwarovém zázemí, které bych mohl pro práci využít. V té době jsem věděl, že budu mít k dispozici jeden testovací cluster o čtyřech počítačích, kde bych mohl teoreticky celou práci jednoduše odzkoušet. Tento cluster budeme dále nazývat jako cluster č. 2. Cluster č. 2 je prakticky nevyužívaný a slouží jenom pro účely tzv. PoC[1].

Správu clustru č. 2 jsem měl v té době pod svojí kontrolou. Na základě praktické zkušenosti jsem věděl, že podmínky k instalaci softwaru jsou ztížené. Jedná se o firemní infrastrukturu. Testovací HSM jsem měl přislíbené a předešlé zkušenosti s HSM jsem neměl. Než jsem přistoupil k reálnému HSM, potřeboval jsem si vše odzkoušet na simulátoru. Dále jsem si nebyl jist, v jakém stavu se daný cluster č. 2 bude nacházet v době letního semestru 2022/2023. Z tohoto důvodu jsem se rozhodl vše předem odzkoušet na dvou osobních počítačích. Jedná se o počítače z let 2013/2014 s již zastaralým hardwarem. Rozhodl jsem se využít architekturu mikroslužeb. Hlavním důvodem pro volbu byl odhad softwarových komponent, které bude třeba zprovoznit pro základní spuštění pseudonymizačních modulů. Přistoupil jsem k nasazení komponent do prostředí tzv. Dockeru. V této situaci se ze softwarových komponent stávají kontejnery. Konkrétně jsem se rozhodl nasadit tyto komponenty do Windows prostředí. Docker, resp. Docker Desktop, jsem nainstaloval na výkonnější z výše zmíněných počítačů (stolní počítač s 16 GB pamětí RAM). Samotný vývoj jsem pak prováděl na počítači s nižším výkonem (notebook s 8 GB pamětí RAM).

Tímto způsobem jsem práci fakticky rozdělil na dvě části, neboli větvě. První větev znamenala vybudovat vše v prostředí Dockeru, kde jsem měl jistotu, že jej budu schopen na konci letního semestru spustit na běžném počítači. Bude tedy možné prezentovat samotný výsledek práce, který by ovšem neběžel na reálném HSM a nereflektoval by reálné prostředí Big Data. Druhá větev byla postavena na tom, že se první větev překlopí na reálný cluster č. 2 v plném rozsahu a připravenost prostředí z první větvě umožní po drobných úpravách plnou konektivitu na reálné HSM. Volba Dockeru byla z mého pohledu ideální i z důvodu snadné převoditelnosti kontejnerů na jiného hostitele.

První větev je dostatečně flexibilní, stabilní a není příliš ovlivněná vnější konfigurací ostatních softwarových komponent přítomných na daném hostitelském stroji. Je zde přítomno několik pseudonymizačních modulů, které bylo možné vyzkoušet a ověřit tak jejich připravenost na nasazení v Big Data prostředí. Dále bylo možné určité moduly zamítnout v situaci, kdy bylo zřejmé, že se i při malé zátěži chovají nekorektně a neexistuje racionální důvod je nasazovat na cluster. První větev tak slouží jako síto modulů a dále poukazuje na řešení pseudonymizace dat v prostřední, které je různé od prostředí Big Data (viz kap. 7.5). Na první větev práce a její architekturu se budeme soustředit v následující podkapitole (viz kap. 6.2).

Předpokládal jsem, že práce na druhé větvi bude komplikovanější. Jak se později ukázalo, mé obavy se naplnili. Nakonec jsem byl schopen na clusteru č. 2 otestovat pouze jeden samostatně stojící modul (viz kap. 6.11).

[1] Proof of Concept

6.2 První větev / Testovací Docker prostředí / Aplikace

Na následujících stranách si přiblížíme architekturu testovacího prostředí, tzv. první větve, kterou jsme využili pro účely diplomové práce. Nastíníme jednotlivé kontejnery, které byly v rámci práce vytvořeny, abychom dosáhli našich cílů. Při popisu jednotlivých kontejnerů vždy zmíníme jejich funkce a způsob interakce s ostatními kontejnery. Rozebereme způsob nasazení testovacího prostředí v Dockeru a propojení jednotlivých kontejnerů za využití Docker Compose.

Našim cílem bylo vybudovat testovací prostředí, ve kterém jsme schopni testovat různé způsoby pseudonymizace. Mezi takové způsoby lze začlenit různé programovací jazyky, různá kryptografická rozhraní, případně různé formáty dat, apod. Abychom byli schopni docílit výše uvedeného, musíme navrhnout vhodnou architekturu systému. Vezmeme-li v úvahu jednotlivé architektury, máme v principu dvě základní volby. První volbou je vytvořit systém, který je monolitický a robustní, těžko je ale schopen reagovat na změny. Musíme vzít v úvahu, že jednotlivé moduly nejsou rozsáhlé a v principu mají právě jeden úkol a dále jsou na sobě prakticky nezávislé. Obdobně lze nahlížet na samotné kontejnery, až na tu skutečnost, že nejsou úplně nezávislé (viz dále). Jak již bylo řečeno v předešlé podkapitole vztahující se k plánu realizace, je zřejmé, že jako vhodnou architekturou se jeví forma mikroslužeb. Mikroslužby nám umožňují vytvářet systém, který se skládá z komponent. Každá komponenta je implementačně nezávislá na druhé.

Dále je pro náš systém podstatná variabilita v použitém jazyce, a to z důvodu jednotlivých kryptografických rozhraní (viz kap. 5.2.1). Důležitá je i možnost operativně přidávat další komponenty s minimem zásahů do zbytku systému. Podstatným aspektem je také způsob nasazení aplikace. Zvolená architektura nám také umožňuje nechat běžet jednotlivé komponenty na různých serverech.

6.3 Popis kontejnerů

Jak již bylo řečeno dříve, při tvorbě kontejnerů si můžeme zvolit operační systém, který v kontejneru poběží, i konkrétní programové vybavení nezávisle na ostatních. Jsme tedy schopni vytvářet prostředí přímo na míru konkrétnímu softwaru. Toto je mimořádně výhodné např. v situaci, kdy jeden modul vyžaduje konkrétní verzi určité knihovny a druhý modul naopak vyžaduje verzi odlišnou. Docker využijeme i v situaci, kdybychom potřebovali testovat různé verze simulátoru HSM; ať již od stejného nebo jiného výrobce. Vzhledem k časovým možnostem byl vyzkoušen jeden simulátor (viz dále).

Není-li uvedeno jinak, autorizace k mikroslužbě probíhá přes autorizační token pomocí frameworku OAuth 2.0. Token je vydán uživateli na základě autentizace vůči autentizačnímu serveru. V našem případě jde o řešení od společnosti Okta. Autentizace probíhá prostřednictvím jména a hesla. Vůči autentizačnímu servu se může libovolný uživatel registrovat na základě platné emailové adresy. Volba použitého frameworku byla opět pragmatická a vycházela ze zkušenosti získané při tvorbě bakalářské práce [59].

6.4 Kontejner HSM Utimaco simulátor

Utimaco, výrobce HSM, které jsme měli k dispozici (viz kap. 6.11.1), umožňuje na svých stránkách po registraci zdarma stáhnout simulátor. Simulátor je k dispozici na webové adrese: **https://support.hsm.utimaco.com/hsm-simulator**.

Společnost dále poskytuje sadu dokumentace, ve které lze např. dohledat návod, jak umístit simulátor do Docker kontejneru. Abychom mohli HSM simulátor plně využívat, potřebujeme jej nejprve inicializovat. Inicializaci lze provést přes příkazovou řádku. Seznam příkazů můžeme umístit do separátního kontejneru a zapnout ho libovolně v případě potřeby. V kombinaci s Docker Compose jsme schopni zajistit, aby byl kontejner vždy k dispozici v příslušné konfiguraci. Podle návodu, jsme tedy vytvořili kontejner a nakonfigurovali dle potřeby. Konfigurace obsahovala inicializaci kryptografického rozhraní PKCS #11 API a dále konfiguraci příslušných uživatelů dle potřeb jednotlivých rozhraní.

6.5 Mikroslužba generování dat

Abychom měli testovací data, nad kterými je možné spouštět jednotlivé pseudonymizační moduly, vytvořili jsme tuto mikroslužbu. Jedná se o jednoduchý generátor CSV souborů.

Mikroslužba generování dat vytváří CSV soubory, které simulují data zaměstnanců. Vygenerovaný soubor obsahuje jméno, příjmení, bydliště osob a IČO[2] zaměstnavatele. Při startu mikroslužby se tzv. *„pool“* naplní statickými záznamy, které jsou v dalším průběhu neměnné. Důvodem pro toto řešení je omezení počtu dotazů na příslušná API. V *„poolu“* jsou k dispozici reálná data ekonomických subjektů, která lze nalézt v ARES[3]. Konkrétně využíváme záznamy o školských zařízeních. Mezi údaje, které zde nalezneme se řadí jméno a příjmení ředitele, IČO organizace a název školského zařízení. Pro generování adres plníme *„pool“* adresami letišť, konkrétně se jedná o službu Rapid API, Airport info. Velikost *„poolu“* záleží na konfiguraci při spuštění kontejneru (tzv. *„environment variables“*), každé spuštění generuje jiné sety dat, protože volba reálných záznamů je čistě náhodná. Např. volba školských zařízení, která se dostanou do *„poolu“*, se odvíjí od náhodného výběru z předem připraveného seznamu IČO těchto zařízení. Následně je zařízení dohledáno v ARES.

Každý uživatelský požadavek na vygenerování souboru si následně vybírá náhodné záznamy z *„poolu“*, které zkombinuje. Je tedy zajištěno, že každý uživatel dostane k dispozici jiná data. Mikroslužba data obsažená v *„poolu“* zkombinuje dohromady a náhodné řádky nahradí údaji z offline souboru, který obsahuje názvy firem. Reálná jména ředitelů nahradí jmény náhodnými. Pro generování náhodných jmen využíváme Python knihovnu *„names“*. Mikroslužba je naprogramována v jazyce Python a svoje služby poskytuje prostřednictvím REST rozhraní. Po dokončení generování souboru mikroslužba nahraje výsledný CSV soubor do mikroslužby ukládání souborů. Z této mikroslužby je pak možné soubor stáhnout a následně jej dále zpracovat.

[2] Identifikační číslo osoby

[3] Administrativní registr ekonomických subjektů

6.6 Mikroslužba ukládání souborů

Úkolem mikroslužby je ukládat přijaté soubory a oproti autorizaci tento soubor umožnit stáhnout. Pro tyto účely byl zvolen Owncloud, který lze považovat za samotnou mikrolužbu. Přístup k Owncloudu je umožněn prostřednictvím mikroslužby ukládání dat, která je naprogramována v jazyce Java ve frameworku SpringBoot. Spojení s Owncloudem probíhá prostřednictvím OCS Share API. Mikroslužba ukládání souborů má své vlastní REST API, které slouží ostatním mikroslužbám pro získání uložených souborů a případnému ukládání dalších. Owncloud je nasazen opět formou Docker kontejnerů a pro svoji správnou funkčnost využívá MariaDB a Redis dostupné jako další kontejnery. Z principu zvolené architektury je možné celou mikroslužbu ukládání souborů zaměnit za jiné řešení. Konkrétně například HDFS ve smyslu zajištění dostupnosti dat všem zúčasněným mikroslužbám.

Zde je vhodné podotknout, že jsem mikroslužbu ukládání souborů netvořil pro účely diplomové práce. Byla vytvořena již v mé bakalářské práci [59]. Následně jsem mikroslužbu použil i v jednom z magisterských předmětů, který se věnoval softwarovým architekturám. Při plánování postupu na diplomové práci jsem si uvědomil, že je tato mikroslužba ideální i pro naše testovací prostředí, jelikož nám umožňuje jednoduchým způsobem zajistit sdílení souborů mezi mikroslužbami a následně je předat uživateli, např. prostřednictvím frontend vrstvy. Pro nasazení v diplomové práci byly provedeny pouze drobné úpravy oproti řešením z předešlých let. To je pozitivní informace, neboť to znamená, že mikroslužba splňuje vlastnost mnohonásobné použitelnosti.

6.7 Backend

První myšlenka byla vedena k tomu, aby každý pseudonymizační modul byl zároveň mikroslužbou a byl umístěn v samostatném kontejneru. Toto řešení bylo později zavrženo, neboť by vedlo k velkému množství duplicitních kódů. Tak vznikla jediná mikroslužba, kterou lze nazvat backendem. Backend má vystavené REST rozhraní, přes které lze posílat dotazy přímo, případně prostřednictvím frontendu.

Musíme vzít v úvahu též reálné Big Data prostředí, kde jsou jednotlivé scripty automatizovaně spouštěny v různých ETL[4] nástrojích, jako je např. NiFi. Těmto scriptům říkejme dále již jen moduly. Jak bylo uvedeno v úvodu kapitoly, pro naše účely rozlišujeme modul integrovaný do backendu, který by bylo možné po drobných úpravách použít s ETL nástrojem, a samostatně stojící modul přímo použitelný s ETL nástrojem. Pro účely popisu backendu je budeme nazývat integrované moduly (*„integrated modules“*), resp. samostatné moduly (*„standalone modules“*). Níže uvádíme seznam použitých knihoven:

- Flask==2.2.2, flask_restx==1.0.3, flask_sqlalchemy==3.0.2,
- python_pkcs11==0.7.0, okta_jwt_verifier==0.2.3, redis==4.5.1
- pandas==1.3.5, python-dotenv==0.21.0, requests==2.28.1, Werkzeug==2.2.2, flask-cors==3.0.10, pyspark==3.0.0

[4]Extract, transform, load

6.7.1 Obecný úvod k modulům

K pseudonymizačním modulům bylo možné přistoupit několika způsoby. Vezmeme-li v úvahu, že nás zajímá pseudonymizace za využití HSM, nabízejí se právě dvě možnosti, jak daná zařízení využívat. První možností je šifrování. Vzhledem k faktu, že nejpoužívanější symetrickou šifrou je AES, byla vždy zvolena tato šifra. Jediné, o čem bylo možné uvažovat, byl zvolený režim činnosti. Z důvodu zajištění široké kompatibility napříč různými kryptografickými API byl zvolen režim činnosti blokové šifry CBC (dále jen **AES-CBC**). Zajistíme tedy pouze **utajení** dat, nikoliv jejich integritu. To je pro naše testovací účely dostačující. Potřebujeme inicializační vektor, jehož primární rolí je randomizovat výstup šifrovací operace. V případě stejného klíče a stejných otevřených textů tak zajistí odlišnost výstupu. Máme-li např. seznam klientů, můžeme inicializační vektor zkonstruovat následovně:

- **sha256(statická tajná hodnota || id klienta || název sloupce)[16]**
- || - značí zřetězení; [16] - značí oříznutí výstupu hashovací funkce na prvních 16 bytů

K této volbě jsme přistoupili ve všech modulech kromě jednoho (Pandas modul). V Pandas modulu jsme inicializační vektory generovali čistě náhodně. To ovšem představuje značnou režii na ukládání těchto vektorů. Proto byl použit výše zmíněný postup s hashovací funkcí SHA-256. Ta je doposud považována za bezpečnou.

Všechny moduly jsou navrženy tak, aby bylo možné vždy získat původní hodnoty. Z tohoto důvodu nebyla HSM využita pro provádění hashovacích operací. Místo toho jsme se zabývali možností využití tzv. *„key-value“* databáze, za kterou lze považovat např. Redis. Bylo tedy nutné nasadit do prostředí Dockeru další kontejner, který Redis obsahoval, a zabezpečit jej základní autentizací. Moduly pracující s Redisem sloužily zejména pro porovnání s řešením, které využívá pouze šifrování.

V rámci vytvářených modulů je nutné definovat terminologii, která je stejná pro integrované i samostatné moduly. Ve všech modulech pracujeme s tzv. *„DataFrame“*. Ten si lze jednoduše představit jako běžnou tabulku, která obsahuje sloupce a řádky. vstupními daty do těchto modulů jsou formáty CSV a Parquet. Oba tyto formáty se do *„DataFrame“* načtou. CSV podporují všechny moduly. Parquet pouze moduly, které bylo možné otestovat na reálném clusteru.

Pro zjednodušení budeme pojem *„DataFrame“* a *„tabulka“* považovat za synonyma. Je důležité poznamenat, že toto zjednodušení použijeme pouze pro lepší nastínění vstupních dat, které zpracováváme.

Za synonyma nelze považovat pojem *„DataFrame“* používaný v Pandas a *„DataFrame“* v rámci Spark. Spark je samostatná technologie, ve které výpočet *„DataFrame“* probíhá odlišným způsobem.

Pro zpracování dat jsme v práci použili zejména Spark a v jednom případě Pandas. Spark je robustní distribuované řešení, které je standardní součástí Big Data prostředí. Z tohoto důvodu byly hlavní moduly implementovány za využití jeho knihoven.

Zmíníme-li v rámci popisu termín **data menšího rozsahu**, znamená to např. tabulky o velikosti tisíců řádků a několik málo jednotek sloupců. **Data většího rozsahu** naopak znamenají např. tabulky o velikosti statisíců řádků a několika málo desítek sloupců.

Z důvodu testovacího prostředí bylo také nutné dostat Spark do Docker prostředí. Po několika neúspěšně odzkoušených variantách byly využity Docker image od André Perez z GitHubu, které jsou popsány na této webové adrese:

https://github.com/cluster-apps-on-docker/spark-standalone-cluster-on-docker

6.7.2 Integrované moduly

V rámci práce byly vytvořeny dva integrované moduly. Oba moduly využívají kryptografické rozhraní PKCS #11 a jsou naprogramovány v jazyce Python. Aby bylo možné rozhraní využít, musí být v backendu dostupná knihovna *„python_pkcs11“*. Knihoven existuje více, pro účely práce byla vybrána tato. Její dokumentace je přímočará a ovládání knihovny intuitivní.

Jak již víme z předešlé kapitoly (viz kap. 4.3), ke komunikaci s HSM za využití rozhraní PKCS #11 potřebujeme název *„tokenu“* a PIN uživatele. Následně budeme schopni provádět kryptografické operace oproti vybranému HSM. Pro tyto úděly je k dispozici mikroslužba **HSM Utimaco simulátor**. Chceme-li využít fyzické zařízení, postačí pouze zaměnit IP adresu simulátoru za IP adresu fyzického zařízení. Jediné, co musíme zajistit, je konfigurace proměnných. Název *„tokenu“* a PINu kryptografického uživatele.

Pandas modul

Jedná se o první pseudonymizační integrovaný modul, který na svém vstupu přijme CSV soubor a uživatelem vybrané sloupce pseudonymizuje. Pro zpracování CSV souboru využívá knihovnu Pandas. Tento modul je nutné považovat za základní, jelikož výpočty nejsou distribuovány napříč clusterem. To je ovšem v určitém úhlu pohledu výhoda, jelikož řešení je rychlé na datech malého rozsahu. Nemusí totiž docházet k plánování samotné distribuce výpočtů. Výhoda tohoto modulu spočívá v jeho jednoduchosti, lze jej použít v libovolném Python prostředí. Pro naše účely se jedná o nejednodušší modul, který bylo možné přímočaře otestovat. Modul využívá kryptografické rozhraní PKCS #11, neboť v prostředí Python, nemáme mnoho možností na výběr.

Spark modul - *„akce collect“*

Druhý modul byl naprogramován tak, aby se co nejvíce podobal modulu předešlému. Toto řešení není praktické, protože při plné zátěži dojde k přetížení tzv. *„Spark driveru“*. Důvodem je právě použitá *„akce collect“*, která je vykonávaná na *„driveru“*. Jedná se o řešení pouze testovací, využívající možnosti Sparku. V průběhu vývoje byl tento modul podstatný pro formulování myšlenek a dalších postupů, které byly uplatněny u samostatných modulů (viz kap. 6.7.3). Modul se dále liší od předešlého vstupními parametry, které umožňují vybrat jednu ze dvou variant pseudonymizace.

První varianta pseudonymizace využívá postup šifrování, popsaný v podkapitole č. 6.7.1. Použité kryptografické rozhraní je opět PKCS #11.

Druhá varianta pseudonymizuje data za využití hashovaní a Redisu. Vstupní hodnoty do hashovací funkce jsou popsány níže:

- **sha256(statická tajná hodnota || id klienta || název sloupce || otevřená hodnota sloupce)**
- ||- značí zřetězení

6.7.3 Samostatné moduly

Oproti integrovaným modulům využívají, samostatné moduly vyšší rozmanitost z hlediska použitých kryptografických rozhraní. Jsou naprogramovány tak, aby využívali potenciál clusteru. První samostatný modul je naprogramován v jazyce Python a druhý je kombinací Jazyka Java a Scala. Aby bylo možné využít zmíněný potenciál clusteru, musí být na všech strojích v clusteru nainstalovány příslušné knihovny. To se týká řešení naprogramovaném v Python. V Java/Scala máme tu výhodu, že můžeme všechny knihovny zabalit do tzv. *„Fat Jar“*, který v praxi vypadá tak, že máme pouze jeden velký soubor, který můžeme na clusteru spustit. Verze knihoven musí být kompatibilní s komponentami na clusteru. Oba moduly využívají Spark transformaci UDF. V rámci odzkoušených variant se tato transformace jevila optimální v porovnání s ostatními.

Spark modul - transformace UDF

Tento modul se výrazně liší od posledního popisovaného modulu (viz kap. 6.7.2). Hlavní změna, která je odlišuje, je v použité transformaci oproti akci. Zde byla použita transformace UDF. Toto řešení je oproti předešlému výhodné proto, že plně využívá potenciál případného clusteru. Samotné UDF již mohou vykonávat tzv. *„workeři“*.

Java/Scala modul

Poslední modul je specifický v kombinaci dvou programovacích jazyků. V jazyce Java jsou k dispozici samotné knihovny, které zajišťují komunikaci s HSM. Samotný kód využívající kryptografická API je naprogramován v Java. Kód, který samotná data zpracovává, je naprogramován ve Scala. Využívá transformaci UDF. Právě tato transformace volá Java kód. Modul je specifický tím, že podporuje tři kryptografická API. Tím hlavním je proprietární CXI (viz kap. 5.2.1), dalšími pak JCA/JCE (viz kap. 5.2.1) a PKCS #11 (viz kap. 5.2.1).

6.8 Frontend

Poslední mikroslužbou zajištující frontend aplikace je grafická nadstavba aplikace naprogramovaná ve frameworku Angular. Výběr tohoto frameworku opět vychází ze zkušenosti z bakalářské práce [59]. Frontend komunikuje pouze s backendem a mikroslužbou generování dat.

6.9 Nasazení první větve / Aplikace

Na následující adrese: **https://diplomka.novascomp.synology.me/** (viz QR kód 7.1) je Docker prostředí vyvedeno z domácí sítě (viz dále). Testovací prostředí (aplikace) po registraci umožní uživateli vygenerovat testovací data (Mikroslužba generování dat) a následně je pseudonymizovat za využití HSM simulátoru (Mikroslužba HSM Utimaco simulátor), a modulů popsaných v podkapitole 6.7. Pro lepší názornost zde provážeme jednu z *„frontendových“* komponent (tzv. Pseudonymizační formulář) s backendem. Úkolem komponenty je zobrazit uživateli možnost volby pseudonymizačních modulů. Uvádíme možnosti volby v pseudonymizačním formuláři a v závorce referenci na popis modulu. U každé volby jsou dvě možnosti. Buď dojde k pseudonymizaci, nebo k opačnému procesu.

- Scala & Spark & (CXI / JCE / PKCS #11) (viz kap. 6.7.3)
- Python & Spark & (PKCS #11 / Redis) & UDF (viz kap. 6.7.3)
- Python & Spark & (PKCS #11 / Redis) & collect (viz kap. 6.7.2)
- Python & Pandas & PKCS #11 (viz kap. 6.7.2)

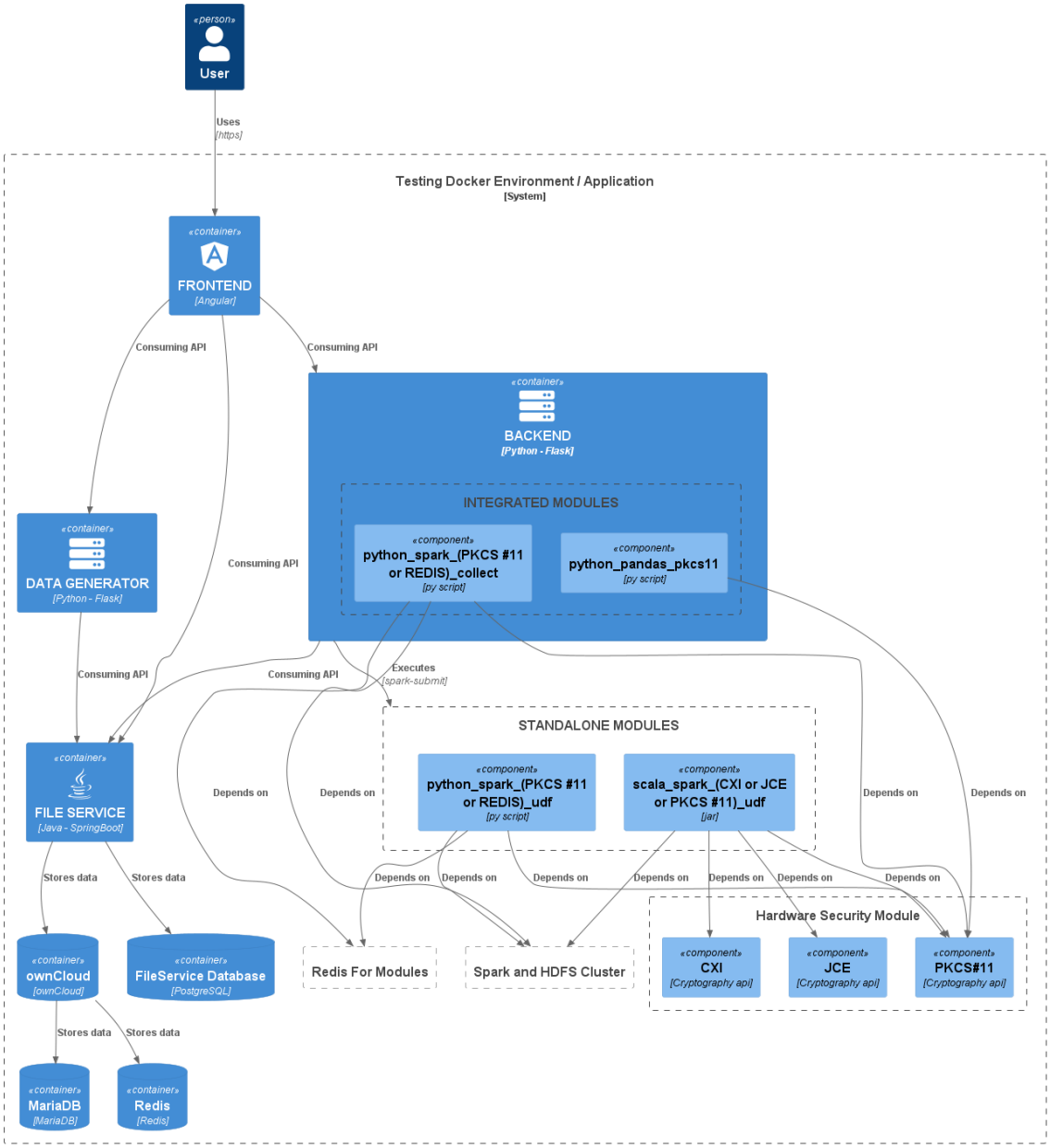
Aplikace je plně nakonfigurována pomocí Docker Compose. Jak bylo uvedeno dříve, Docker běží na běžném kancelářském počítači z roku 2014 s procesorem Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40GHz. Operační paměť byla z důvodu diplomové práce navýšena z 8 GB na 16 GB. Porty jsou vyvedeny z domácí sítě prostřednictvím reverzní proxy, která je k dispozici na zařízení Synology NAS. Mikroslužby jsou k dispozici ve formě *„Docker imagů“*, které byly vytvořeny a následně nahrány na Docker Hub za využití GitHub Actions a příslušných tzv. *„Dockerfiles“*. Samotné Docker image jsou k dispozici na Docker Hub a příslušné zdrojové kódy pak v samostatném GitHub repozitáři (viz příloha A).

Na další straně nalezneme náčrt architektury testovacího prostředí (viz obr. 6.1).

6.10 Náčrt architektury aplikace

Náčrt zachycuje všechny výše popisované mikroslužby a pseudonymizační moduly. Pro úplnost provážeme obrázek s dosavadním popisem.

- FRONTEND (viz kap. 6.8)
- BACKEND (viz kap. 6.7), INTEGRATED MODULES (viz kap. 6.7.2, 6.7.2)
- STANDALONE MODULES (viz kap. 6.7.3, 6.7.3)
- DATA GENERATOR (viz kap. 6.5)
- FILE SERVICE (viz kap. 6.6)



Obrázek 6.1: Architektura testovacího prostředí

6.11 Druhá větev / Reálné prostředí

Jak již bylo řečeno v plánu realizace (viz kap. 6.1), teoreticky by neměl být problém celé testovací prostředí v Dockeru překlopit na cluster č. 2. Realita je ovšem z důvod použité firemní infrastruktury jiná.

V prvé řadě zde nejde použít autentizaci a autorizaci k mikroslužbám prostřednictvím Okta. Mikroslužba generování dat také nefunguje, jelikož posílá požadavky na ARES. Vše z důvodu firemní bezpečnosti.

Z výše popsanými omezeními se lze snadno vypořádat tak, že využijeme pouze mikroslužbu backendu bez frontendu a bez mikroslužby generování dat. Mikroslužba generování dat, byla stejně určena primárně pro testovací prostředí, jelikož nebyla přizpůsobena na generování dat většího rozsahu. Pro reálné prostřední jsme využili otevřená data ve formátu Parquet. Mezi tato data patří např. data o taxi provozovaných v NYC[5]. Ta jsou k dispozici na adrese: **https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page** a lze je jednoduše škálovat, co do počtu testovaných řádků. Právě škálovatelnost a reprezentativnost prostředí Big Data byla důvodem pro volbu těchto dat. Ta byla následně nahrána na HDFS. Autentizace a autorizace byly z důvodu výše zmíněných podmínek vypnuty.

Backend byl tedy přizpůsoben těmto podmínkám. Bohužel se v průběhu testování objevil problém s konektivitou na reálné HSM. Pseudonymizační moduly běžely řádově pomaleji oproti běhu na simulátoru. Příčina je složitější. Určitou roli hraje fakt, že cluster č. 2 je umístěn v jiném data centru než HSM. Dalším problémem jsou dle mého názoru PKCS #11 *„wrappery“*. Na reálném HSM, kde je řádově několik stovek klíčů, trvá vyhledání klíče neúměrně dlouhou dobu (desítky sekund).

Tento problém byl neočekávaný a okamžitě nás postavil do jiné situace. Museli jsme tedy sehnat cluster, který bude blíže k fyzickému HSM. Cluster označme jako cluster č. 1.

Cluster č. 1 a č. 2 jsou si výkonnostně podobné. Rozdíl ovšem je v tom, že na clusteru č. 1 jsme naprosto limitováni možnostmi instalace. I z tohoto důvodu bylo rozhodnuto, že odzkoušíme pouze samostatně stojící moduly, resp. pouze poslední, ten za využití Java/Scala (viz kap. 6.7.3).

Samostatně stojící modul v Python (viz kap. 6.7.3) odzkoušet nelze. Problém spočívá právě v jazyce Python. Abychom zde mohli modul odzkoušet, znamenalo by to instalaci knihoven na všechny počítače v clusteru. Zejména podstatná je pro nás již výše zmiňovaná knihovna *„python_pkcs11“*. Tuto instalaci zde provést nemohu, protože bych zasahoval do infrastruktury, která je již důležitější než cluster č. 2.

Pseudonymizační modul napsaný v Java/Scala byl tedy plně odzkoušen na datech většího rozsahu oproti reálnému HSM. Zároveň na clusteru č. 1 máme jistotu, že Spark a HDFS jsou nakonfigurovány korektně. Samotné výsledky jsou popsány v následující kapitole (viz kap. 7).

[5]City of New York

6.11.1 Použité HSM

V oblasti hardwaru bylo k dispozici HSM Utimaco SecurityServer Se-Series (Se52) Gen2 CSLAN 4 (viz obr. 6.2). Toto zařízení lze zařadit mezi tzv. *„General Purpose“* nižší třídy. Pořizovací hodnota se pohybuje okolo 10 000 euro.

Zařízení je již poněkud starší. Bylo pořízeno před rokem 2020. Vezmeme-li v úvahu tehdejší kurz, můžeme konstatovat, že cena zařízení se pohybuje okolo čtvrt milionu českých korun. Konektivita zařízení je zajištěna pomocí ethernet portů. Zařízení má přiřazenou IP adresu. Konfiguraci zařízení lze provést přes dodaný software. Určité úkony lze provést prostřednictvím předního panelu zařízení.

Obrázek 6.2: Ilustrační foto použitého HSM [60]

6.12 Praktické zkušenosti s HSM

Na začátku práce jsme na HSM nahlíželi jako na tzv. *„black box“*. Vycházeli jsme z obecného popisu (viz kap. 3). Nyní je třeba nastínit praktické aspekty zařízení. Popíšeme základní konfiguraci a její praktické dopady. Je vhodné zdůraznit, že vycházíme ze zkušeností s výše zmíněným zařízením a jeho obdobou v podobě simulátoru.

Obrázek 6.3: Bezpečnostní karty [61]

Obrázek 6.4: PIN Pad [61]

Do zařízení se lze přihlásit výchozím klíčem. Klíč je k dispozici ve složce s dodávaným softwarem. K HSM je také dodána sada bezpečnostních karet (viz obr. 6.3) a tzv. PIN Pad (viz obr. 6.4) pro jejich čtení. Karty slouží k bezpečnému uchování klíčů.

Jedná se např. o klíče administrátorů (RSA-Key, ECC-Key) pro správu zařízení a tzv. **MBK**[6], resp. jeho část (viz dále). Klíče jsou chráněny příslušným PINem. Pro více informací uvádíme následující zdroj [61], ze kterého pocházejí fotografie tohoto příslušenství.

MBK je 256 bitový AES klíč rozdělený na několik částí. Jedna z doporučených variant je tzv. *„**m out of n**“* pravidlo. Toto pravidlo rozdělí MBK na **n** částí. V praxi to znamená, že je potřeba mít k dispozici n karet. Číslo **m** udává, kolik karet musíme mít v dané chvíli k dispozici, abychom MBK mohli složit. Jinými slovy tedy číslo **(m - n)** udává počet karet, které lze případně postrádat. Pro úplnost je vhodné dodat, že MBK lze exportovat i na pevný disk a zabezpečit ho heslem.

Nyní je třeba zmínit důležitost MBK. Nutné je připomenout roli kryptografického uživatele, neboli normální uživatel, neboli *„normal user“* (viz kap. 4.3). Tento uživatel má právo pracovat se všemi klíči, které má přidělené ve svém paměťovém prostoru. V rámci PKCS #11 je jím *„slot“*. To ovšem zahrnuje i jejich **export/import**. Právě pro ochranu klíčů v době exportu vznikl MBK.

Při exportu jsou klíče zašifrovány pomocí MBK. Následně lze klíče nahrát do jiného HSM stejné kategorie. Aby bylo možné klíče v takové situaci přečíst, musí zde být **naimportován** stejný MBK. Tento postup lze využít např. pro účely záloh klíčů, případně pro synchronizaci více HSM pro účely běhu v cluster módu.

6.13 Nejslabší článek HSM

Lze jednoduše dovodit, že máme-li k dispozici přihlašovací údaje kryptografického uživatele a zároveň MBK, tak máme k dispozici vše potřebné pro provádění kryptografických operací s těmito klíči. Zejména však máme naprostou kontrolu nad těmito klíči v podobě jejich exportu a importu do jiných zařízení. Skutečná hodnota klíčů je stále chráněna.

MBK je tedy extrémně důležitý klíč, který je nutné chránit, resp. jeho části. Pro tyto účely je vhodné použít výše zmíněné karty a neukládat je na disk. Karty je pak nutné na základě interní organizace společnosti rozdělit mezi zodpovědné osoby.

Osobně spatřuji největší slabinu v praktické ochraně přihlašovacích údajů kryptografického uživatele. Tyto údaje na kartu nahrát nelze. Bylo by to i velice nepraktické. V reálném prostředí tak může v praxi docházet k situaci, kdy přihlašovací údaje kryptografického uživatele budou ukládány spolu s příslušným kódem, který bude HSM využívat. V neposlední řadě je nutné zdůraznit, že se jedná o PIN. To znamená, že variabilita kombinací je omezená.

6.14 Shrnutí pseudonymizačního prostředí

Tato kapitola se zabývala motivaci a přístupem ke zvolenému postupu. Pokusili jsme se prezentovat jednotlivé kroky, které byly učiněny k tomu, abychom mohli svědomitě učinit závěry o formulované hypotéze (viz kap. 5.1). Námi vytvořené pseudonymizační prostředí jistě není optimální, ale snaží se především formulovat myšlenky, postupy a z nich pak produkuje jeden koherentní celek, který je možný otestovat na clustru v reálných podmínkách.

[6] Master Backup Key

Kapitola 7

Vyhodnocení výstupů práce

7.1 Provedené testy

Byly vyzkoušeny řádově desítky konfiguračních parametrů *„spark-submit scriptu“*. Na základě konfigurace, která vykazovala nejlepší vlastnosti, jsme modul doladili a finální verzi odzkoušeli na jedné datové sadě - datová sada NYC taxi. Konkrétně se jednalo o tři testy, každý test na jiném kryptografickém rozhraní (viz tab. 7.1).

Provedené testy - pseudonymizace, šifrování - HSM			
Velikost datové sady / kryptografické rozhraní	100 000 řádků x 18 sloupců	1 000 000 řádků x 18 sloupců	1 500 000 řádků x 18 sloupců
CXI	4.5 min	43 min	60 min
JCA/JCE	6 min	55 min	84 min

Tabulka 7.1: Pseudonymizace, šifrování za využití HSM

Názvy sloupců udávají počet záznamů v datové sadě. Z podstaty problému musíme šifrovat konkrétní záznam v daném řádků a příslušném sloupci (pole tabulky). To nás vede ke komplexitě dotazů do HSM o velikosti počtu řádků krát počet pseudonymizovaných sloupců. Modul vždy provádí pseudonymizaci sloupců na základě parametrů předaných při volání *„spark submit scriptu“*.

Z tabulky je patrné, že proprietární rozhraní CXI vykazuje nejlepší výsledky oproti rozhraní JCA/JCE. Rozhraní PKCS #11 nebylo na clusteru č. 1 testováno. Jak jsme již zmínili dříve, při testování *„wrapperů“* jsme objevili problém s počátečním vyhledáním klíče, který proceduru pseudonymizace neúměrně zpomaluje.

V tabulce je dále patrná lineární časová komplexita. Hlavním limitujícím faktorem se jeví spojení do HSM a dále pak jeho výkonost na symetrických kryptosystémech. Abychom dosáhli lepších výsledků, museli bychom vyměnit testované HSM za výkonnější model. Důvod, proč kryptografické rozhraní CXI je významně rychlejší než rozhraní JCA/JCE, je neznámý a aktuálně na něj nejsme schopni odpovědět. Kód, který data zpracovává, je identický.

Obdobné testy jsme provedli i bez využití HSM (viz tab. 7.2). Jedná se o identický kód a stejné kryptografické rozhraní JCA/JCE. Pouze jsme JCA/JCE neposkytli tzv. *„Providera“* a využili procesory clustru.

Provedené testy - pseudonymizace, šifrování - bez HSM			
Velikost datové sady / kryptografické rozhraní	100 000 řádků x 18 sloupců	1 000 000 řádků x 18 sloupců	1 500 000 řádků x 18 sloupců
JCA/JCE	40 s	60 s	72 s

Tabulka 7.2: Pseudonymizace, šifrování bez využití HSM

Rozdíl mezi těmito dvěma tabulkami je markantní. Výše uvedená tabulka nám potvrzuje naši teorii, že hlavním limitujícím faktorem je nedostatečná výkonnost HSM pro naše účely.

7.2 Vyhodnocení hypotézy

Formulovanou hypotézu (viz kap. 5.1) jsme nuceni **zamítnout**, a to na základě samostatně stojícího Java/Scala modulu (viz. kap. 6.7.3), který jsme vyzkoušeli na clusteru č. 1.

Hypotézu zamítáme z důvodu časově nákladných nároků na provádění pseudonymizace. Zjistili jsme, že jsme schopni zpracovat přibližně **27 miliónů záznamů** (polí v tabulce) za hodinu. To je evidentně nedostatečné v oblasti řešeného problému.

7.3 Nezbytné konstatování

Je důležité zdůraznit, že provedením pseudonymizace samotných dat nezajišťujeme jejich kompletní bezpečnost. Musíme zajistit jejich dostatečnou ochranu již před samotným začátkem pseudonymizace a dále pak ochranu před různými postranními kanály. Konkrétní podmínky v daném prostředí by determinovaly, jaká konkrétní technická a organizační opatření bychom byli nuceni přijmout.

Vždy je nezbytné myslet na skutečnost, že bezpečnost je proces a ne šifrovací algoritmus, technologie nebo HSM.

Představa, že bezpečnost zvýšíme tím, že pořídíme drahé zařízení, je lichá. Vždy musíme přispívat ke zlepšení všech článků procesu, a to úměrně a přiměřeně. Pokud pořídíme HSM a údaje ke kryptografickému uživateli budou přístupná všem, tak jsme naopak bezpečnost skokově snížili (viz kap. 6.13).

7.4 Zpětné zhodnocení práce

Na předloženou práci je nutné nahlížet jako na způsob ověření hypotézy o vhodnosti integrace HSM do prostředí Big Data pro účely ochrany zpracovávaných dat.

Samotným tématem by pak byla otázka, jak přesně má být pseudonymizace provedena. Chceme, aby se stejný otevřený text zobrazil na stejné zašifrované hodnoty? Pokud ano, tak pro jaké sloupce? Nebo je našim cílem ze zašifrovaných hodnot nevyčíst žádnou informaci?

Domnívám se, že toto je individuální záležitost dat, které zpracováváme. Záleží na jejich povaze a zejména na účelu zpracování. Pro účely této práce není nutné na tyto otázky odpovídat. Je ale třeba s nimi počítat v případě dalšího rozvoje práce.

Z hlediska volby režimu činnosti blokové šifry AES, bylo dle mého názoru, rozumné vybrat režim CBC. I s ohledem na naše tvrzení, že je nízká pravděpodobnost, že by tento režim nebyl někde podporován (viz kap. 6.7.1). Toto tvrzení si potvrdíme (viz kap. 7.5.2). Je evidentní, že zvolený režim činnosti významně neovlivňuje výše prezentované výsledky. O čem by ale bylo možné polemizovat a případně dále rozvíjet, je schopnost AES-CBC naplnit cíle informační bezpečnosti. Ty tento režim nenaplňuje, protože nezajišťuje integritu a autentizaci. V případném dalším rozvoji práce by bylo rozumné se na tuto skutečnost zaměřit.

7.4.1 Důležité body vývoje

Připomeňme seznam činností, které zabraly nejvíce času a bez kterých by se finální modul neobešel.

- Konfigurace testovacího prostředí: Časově nejnáročnější byl vývoj backendu, který z výkonnostních důvodů (nedostatek RAM paměti na notebooku) musel být vždy nahrán na GitHub a následně pomocí GitHub Actions nahrán na Docker Hub. Poté byl stažen na výkonnější z počítačů a spuštěn v Dockeru. Výše zmíněné trvalo v průměru deset minut. Počet takových stažení se blíží dvěma stovkám. Vývoj byl stížen i tím, že z technických důvodů bylo možné výkonnější počítač používat pouze prostřednictvím *„Remote Desktop Connection“*. Problematická byla také knihovna *„python-pkcs11“*, která se ve Windows prostředí nechovala vždy korektně.
- Konfigurace clustru č. 2: Mimo jiné bylo třeba zajistit, aby na každém stroji byly dostupné stejné verze programovacího jazyka Python a jeho příslušných knihoven. Dále pak zajistit náhradní řešení za tento cluster.
- Seznámení se s dokumentací HSM a příslušných kryptografických API.
- Dohledání kompatibilních knihoven, které je možné použít na clustru č. 1 a zároveň v testovacím prostředí.
- Odzkoušení a zprovoznění potřebných technologií v testovacím prostředí (zejména Spark a Redis) a následná konfigurace tzv. *„Dockerfiles“*
- Snaha o plné zprovoznění testovacího prostředí ve firemní infrastruktuře, včetně detekce komponent, které bylo možné odzkoušet a které nikoliv. Obecně se jednalo o řešení problémů (hledání alternativ) spojených s bezpečností a politiky firemní infrastruktury.

7.5 Další možný rozvoj práce

7.5.1 Z hlediska testovacího prostředí

Jak již bylo řečeno, hypotézu jsme byli nuceni zamítnout. To nám ovšem nebrání vzít v potaz zjištěné skutečnosti a zamyslet se nad možností praktického využití navrženého testovacího prostředí.

V prostředí firem často dochází k situacím, kdy si zaměstnanci přeposílají různé typy tabulek. V práci jsme ukázali, že takové tabulky lze jednoduše šifrovat za využití HSM, pokud jde o data malého rozsahu. V kancelářském prostředí jsou tyto tabulky malé, většinou menší, než tabulky, které generovala naše mikroslužba generování dat.

Do navrženého testovacího prostředí by bylo možné zakomponovat mikroslužbu, která by fungovala jako prostředník mezi odesílatelem a příjemcem. Odesílatel by odeslal email, jak je zvyklý. Rozdíl by ovšem spočíval v tom, že by do kolonky adresy místo příjemce uvedl výše zmíněnou mikroslužbu. Mikroslužba by se postarala o samotné zašifrování tabulky a zajistila by její bezpečné zpřístupnění zamýšlenému příjemci. To by mohlo probíhat formou odeslání odkazu. Po kliknutí na odkaz příjemcem by proběhlo dešifrování souboru a následné stažení.

Výše uvedený postup by v praxi vedl k určité ochraně. Samotné tabulky by nebyly v poštovních schránkách příjemců a nemohlo by tedy dojít k přeposlání takových emailů jiným adresátům. Tabulky by se internetem šířili čistě v zašifrované podobě, za předpokladu, že by stažení dešifrovaného souboru probíhalo za využití TLS.

Výše zmíněnou myšlenku by bylo možné rozvinout o autentizaci a autorizaci na straně příjemce. Na straně odesílatele by bylo vhodné zlepšit formu odeslání tabulky. To by v praxi mohlo znamenat nahrání tabulky do webového formuláře, který by se postaral o vše potřebné. Nutno dodat, že v praxi by mohlo docházet k tomu, že by se zaměstnanci snažili takový formulář obcházet. Vynutit vše výše popsané v praxi, by bylo velmi náročné.

7.5.2 Z hlediska Big Data prostředí

Pokud bychom se vrátili k pseudonymizaci dat v naší popsané výseči v úvodu kapitoly (viz kap. 5), mohli bychom provádět šifrování dat bez využití HSM, resp. využít samotné HSM pouze jako formu ochrany klíče a ne jako zařízení zajišťují bezpečné provádění kryptografických operací. Pro tyto účely lze ovšem využít jiné zařízení pro takové účely vytvořené, tzv. ESKM[1]. Zmíníme opět produkty společnosti Utimaco, obdobných produktů je více.

ESKM je navrženo především pro účely ochrany a správu velkého množství klíčů. V jeho hardwarové podobě se počet klíčů pohybuje v řádech nižších miliónů [62] a je navrženo pro práci v clusteru. Tato zařízení je možné integrovat s HSM. Existuje i virtualizovaná verze pro prostředí VMwaru tzv. vESKM[2]. To je užitečné v případech, kdy bychom rádi pracovali ve virtualizovaném prostředí.

[1] Enterprise Secure Key Manager
[2] Virtual Enterprise Secure Key Manager

Zajímavostí těchto řešení je např. podpora práce s klíči prostřednictvím REST rozhraní. To z nich dělá přívětivé a známe prostředí pro vývojáře. Prostřednictvím REST rozhraní lze např. získat vlastnosti daného klíče. Pokud má klíč příznak *„exportable“*, je možné získat bity klíče. Existuje i možnost provádět šifrovací a dešifrovací operace, ale to spíše symbolicky. K tomuto tvrzení mě vede osobní zkušenost (nastudovaná dokumentace rozhraní a vyzkoušená trial verze vESKM ve *„VMware Workstation Player“*).

To v praxi vypadá tak, že rozhraní podporuje právě dva režimy blokové šifry AES. ECB a námi vyzkoušený režim CBC. Při vytvoření klíče je režim činnosti explicitně zmíněn a je vytvořen tzv. *„default IV“* (výchozí inicializační vektor). Modifikovat *„IV“* (inicializační vektor) prostřednictvím REST rozhraní nelze. V praxi se pak celé šifrování stává nepraktické a z hlediska bezpečnosti je CBC režim výrazně degradován.

7.6 Shrnutí praktické části

V této chvíli jsme se dostali na konec praktické části práce a v principu i na její samotný konec. Provedli jsme vyhodnocení výstupů práce, kde jsme narazili na značný nepoměr. Tento nepoměr spočívá ve výkonosti pseudonymizace za využití konkrétního HSM oproti jednotlivým počítačům v clustru. Některé příčiny tohoto problému jsme již zmínili. Pokud bychom měli zmínit poslední z bodů, který by se věnoval dalšímu rozvoji této práce, jednalo by se o detailní prostudování sad protokolů, přes které komunikuje HSM s počítači v clustru.

Závěrem je vhodné poznamenat, že HSM jsou výborná zařízení pro určité spektrum oblastí. Jako příklad lze uvést asymetrickou kryptografii, např. oblast ukládání klíčů certifikačních autorit, dále pak pro účely podepisování dokumentů, apod.

Obdobně lze uvažovat v situacích, kdy máme zajištěnu fixní velikost dat a neklademe nároky na škálovatelnost, jejíž cena je zde vysoká (finanční náročnost pořízení dalších HSM pro vytvoření HSM clustru). Neklademe-li požadavky na rychlost zpracování, může se jednat o vhodné řešení. Vždy je nezbytné vše vztáhnout ke konkrétnímu očekávání, které od HSM máme. A dále pak zvážit konkrétní přínosy, které může přinést v daném prostředí. Těch je bezpochyby celá řada (viz kap. 3).

Závěr

Závěrem provedeme shrnutí práce. V úvodu jsme provedli rešerši českých a německých zákonů, které integrují nařízení GDPR do konkrétního odvětví. Provedli jsme také definici základních pojmů z oblasti kryptografie a detailněji jsme se zaměřili na terminologie vztahující se k HSM. Závěr teoretické práce byl věnován standardům Public Key Cryptography Standards neboli PKCS. Každou kapitolu teoretické části jsme shrnuli tak, aby byl zřejmý její přínos do diplomové práce.

Praktická část spočívala ve vybudování testovacího prostředí, které sloužilo k návrhu několika pseudonymizačních modulů. To, mimo jiné, zahrnovalo seznámení se s HSM a možnostmi komunikace s tímto zařízením, tzv. kryptografické API, a dále technologiemi, které jsme využili. Blíže jsme přiblížili technologie HDFS, Spark a Docker.

Vybrané pseudonymizační moduly byly otestovány na clusterech. Postupně vyplynulo, že je vhodné se zaměřit na jeden modul a ten otestovat na datech většího rozsahu. To nám poskytlo dostatek podkladů pro zamítnutí hypotézy o vhodnosti HSM pro provádění kryptografických operací v námi definované výseči Big Data. V samotném závěru jsme pak nastínili další možný rozvoj práce. Nezapomněli jsme také zdůraznit důležitost vnímání bezpečnosti jako procesu. **Tímto považuji cíle práce za naplněné**.

Mezi přínosy práce bych zařadil ověření hypotézy v podmínkách reálného clustru v kombinaci s HSM. Obecněji pak snahu integrovat HSM do prostředí Big Data. Konkrétně pak samotné provedené testy.

Pro mě osobně shledávám přínos práce v praktickém seznámení se s HSM a dále pak ve zprovoznění prostředí v běžných domácích podmínkách. Do těch nás zavede následující **QR kód** (viz obr. 7.1). Tento odkaz je platný pro květen a červen roku 2023.

Obrázek 7.1: Pseudonymizační aplikace

Bibliografie

1. *NAŘÍZENÍ EVROPSKÉHO PARLAMENTU A RADY (EU) 2016/679 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů a o zrušení směrnice 95/46/ES (obecné nařízení o ochraně osobních údajů)* [online]. [cit. 09. 11. 2022]. Dostupné z: `https://www.uoou.cz/uplne-zneni-gdpr/ds-6607/archiv=1&p1=3938`.
2. *§ 66 odst. 6 zákona č. 110/2019 Sb., o zpracování osobních údajů - znění od 24. 4. 2019.* [online]. [cit. 09. 11. 2022]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2019-110#p66-6`.
3. *§ 2 písm. e) zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 25. 09. 2022]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2019-110#p2-1-e`.
4. *§ 50 odst. 1 zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 25. 09. 2022]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2019-110#p50-1`.
5. *§ 51 odst. 1 zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 25. 09. 2022]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2019-110#p51-1`.
6. *§ 54 odst. 1 písm. c) zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 25. 09. 2022]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2019-110#p54-1-c`.
7. *§ 54 odst. 2 písm. a) zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 25. 09. 2022]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2019-110#p54-2-a`.
8. *§ 54 odst. 2 písm. c) zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 25. 09. 2022]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2019-110#p54-2-c`.
9. *§ 54 odst. 2 písm. d) zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 25. 09. 2022]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2019-110#p54-2-d`.
10. *§ 54 odst. 2 písm. e) zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 25. 09. 2022]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2019-110#p54-2-e`.
11. *§ 55 odst. 5 zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 25. 09. 2022]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2019-110#p55-5`.
12. *§ 60 odst. 1 zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 25. 09. 2022]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2019-110#p60-1`.

13. *§ 61 odst. 1 zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 03. 10. 2022]. Dostupné z: https://www.zakonyprolidi.cz/cs/2019-110#p61-1.

14. *§ 61 odst. 2 písm. b) zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 03. 10. 2022]. Dostupné z: https://www.zakonyprolidi.cz/cs/2019-110#p61-2-b.

15. *§ 63 zákona č. 110/2019 Sb. o zpracování osobních údajů - znění od 24.04.2019* [online]. [cit. 03. 10. 2022]. Dostupné z: https://www.zakonyprolidi.cz/cs/2019-110#p63.

16. *Federal Data Protection Act of 30 June 2017 (Federal Law Gazette I p. 2097), as last amended by Article 10 of the Act of 23 June 2021 (Federal Law Gazette I, p. 1858; 2022 I p. 1045).* [online]. [cit. 29. 12. 2022]. Dostupné z: https://www.gesetze-im-internet.de/englisch_bdsg/englisch_bdsg.html.

17. *Gesetz über den Datenschutz und den Schutz der Privatsphäre in der Telekommunikation und bei Telemedien* [online]. [cit. 02. 05. 2023]. Dostupné z: https://www.gesetze-im-internet.de/ttdsg.

18. *§ 89 odst. 3 zákona č. 127/2005 Sb. o elektronických komunikacích - znění od 01.07.2022* [online]. [cit. 02. 05. 2023]. Dostupné z: https://www.zakonyprolidi.cz/cs/2005-127#p89-3.

19. *Cookie lišty a udělování souhlasu* [online]. [cit. 02. 05. 2023]. Dostupné z: https://www.uoou.cz/casto-kladene-otazky-ohledne-souhlasu-s-cookies-udeleneho-prostrednictvim-tzv-cookie-listy/ds-6912/archiv=1&p1=2619.

20. *ANSDIT* [online]. [cit. 24. 10. 2022]. Dostupné z: https://www.incits.org/html/ext/ANSDIT/Ansdit.htm.

21. *Internet Security Glossary, Version 2* [online]. [cit. 24. 10. 2022]. Dostupné z: https://www.rfc-editor.org/rfc/pdfrfc/rfc4949.txt.pdf.

22. *INTERNATIONAL STANDARD* [online]. [cit. 10. 10. 2022]. Dostupné z: https://standards.iso.org/ittf/PubliclyAvailableStandards/c073906_ISO_IEC_27000_2018_E.zip.

23. NIST. *FIPS PUB 180-4 Secure Hash Standard (SHS)* [online]. [cit. 25. 10. 2022]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

24. TILBORG, Henk C.A. van. *Encyclopedia Of Cryptography And Security*. Springer Science+Business Media, Inc., 2005.

25. WANG, Shawn. *The difference in five modes in the AES encryption algorithm* [online]. [cit. 07. 05. 2023]. Dostupné z: https://www.highgo.ca/2019/08/08/the-difference-in-five-modes-in-the-aes-encryption-algorithm/.

26. MCGREW, David A.; VIEGA, John. *The Galois/Counter Mode of Operation (GCM)* [online]. [cit. 07. 05. 2023]. Dostupné z: https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf.

27. *The Illustrated TLS 1.3 Connection, Every byte explained and reproduced* [online]. [cit. 07. 05. 2023]. Dostupné z: https://tls13.xargs.org/.

28. *The Illustrated TLS 1.2 Connection, Every byte explained and reproduced* [online]. [cit. 07. 05. 2023]. Dostupné z: https://tls12.xargs.org/.

29. *NAŘÍZENÍ EVROPSKÉHO PARLAMENTU A RADY (EU) č. 910/2014 ze dne 23. července 2014 o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu a o zrušení směrnice 1999/93/ES* [online]. [cit. 25. 10. 2022]. Dostupné z: https://eur-lex.europa.eu/legal-content/CS/TXT/PDF/?uri=CELEX:32014R0910&from=CS.

30. GOLLOVÁ, Alena. *Matematická kryptografie* [online]. [cit. 07. 05. 2023]. Dostupné z: https://math.fel.cvut.cz/en/people/gollova/mkr.html.

31. NIST. *FIPS PUB 140-2 Security Requirements for Cryptographic Modules* [online]. [cit. 21. 10. 2022]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf.

32. NIST. *FIPS PUB 140-3 Security Requirements for Cryptographic Modules* [online]. [cit. 25. 10. 2022]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf.

33. JOHNSTON, Roger G. *Tamper-Indicating Seals: Practices, Problems, and Standards* [online]. [cit. 21. 10. 2022]. Dostupné z: https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-03-0269.

34. BARKER, Elaine; BRANSTAD, Dennis; SMID, Miles. *NIST Special Publication 800-152 A Profile for U. S. Federal Cryptographic Key Management Systems* [online]. [cit. 24. 10. 2022]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-152.pdf.

35. PHILIPP, Andreas. *Hardware Securit Modules (HSM) for Dummies* [online]. [cit. 24. 10. 2022]. Dostupné z: https://www.creaplus.si/images/Utimaco/HSM_for_Dummies.pdf.

36. *PKCS #1: RSA Cryptography Specifications Version 2.2* [online]. [cit. 16. 09. 2022]. Dostupné z: https://www.rfc-editor.org/rfc/pdfrfc/rfc8017.txt.pdf.

37. GOLLOVÁ, Alena. *Diskrétní logaritmus, 13. a 14. přednáška z kryptografie* [online]. [cit. 18. 09. 2022]. Dostupné z: https://math.fel.cvut.cz/en/people/gollova/mkr/mkr7.pdf.

38. *PKCS #5: RSA Password-Based Cryptography Specification Version 2.1* [online]. [cit. 18. 09. 2022]. Dostupné z: https://www.rfc-editor.org/rfc/pdfrfc/rfc8018.txt.pdf.

39. LAKE, Josh. *What is a key derivation function (KDF) and how do they work?* [online]. [cit. 19. 09. 2022]. Dostupné z: https://www.comparitech.com/blog/information-security/key-derivation-function-kdf/.

40. DANG, Quynh. *NIST Special Publication 800-135 Revision 1 Recommendation for Existing Application-Specific Key Derivation Functions* [online]. [cit. 19. 09. 2022]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf.

41. *HMAC: Keyed-Hashing for Message Authentication* [online]. [cit. 19. 09. 2022]. Dostupné z: https://www.rfc-editor.org/pdfrfc/rfc2104.txt.pdf.

42. *Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms* [online]. [cit. 19. 09. 2022]. Dostupné z: https://www.rfc-editor.org/pdfrfc/rfc6151.txt.pdf.

43. OWASP. *Password Storage Cheat Sheet* [online]. [cit. 19. 09. 2022]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html.

44. *PKCS 11 Cryptographic Token Interface Usage Guide Version 2.40* [online]. [cit. 10. 07. 2022]. Dostupné z: http://docs.oasis-open.org/pkcs11/pkcs11-ug/v2.40/cn02/pkcs11-ug-v2.40-cn02.html.

45. *PKCS 11 Cryptographic Token Interface Base Specification Version 2.40* [online]. [cit. 10. 07. 2022]. Dostupné z: http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html.

46. KARUNANAYAKE, Mevan. *Standard API for connecting HSMs with client applications* [online]. [cit. 10. 07. 2022]. Dostupné z: https://medium.com/@mevan.karu/standard-api-for-connecting-hsms-with-client-applications-6296eb187d89.

47. MATHUR, Shreyansh. *Big Data with 8v's and today's challenges* [online]. [cit. 22. 04. 2023]. Dostupné z: https://shreyanshmathur1998.medium.com/big-data-with-8vs-and-today-s-challenges-9938f00363c5.

48. *Java Cryptography Architecture (JCA) Reference Guide* [online]. [cit. 27. 12. 2022]. Dostupné z: https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html.

49. *PKCS11 Wrapper Core Crypto Toolkits* [online]. [cit. 23. 04. 2023]. Dostupné z: https://jce.iaik.tugraz.at/products/core-crypto-toolkits/pkcs11-wrapper/.

50. *Python PKCS11 - High Level Wrapper API* [online]. [cit. 23. 04. 2023]. Dostupné z: https://python-pkcs11.readthedocs.io/en/latest/.

51. KARUNANAYAKE, Mevan. *You don't need to buy a HSM to see how it works* [online]. [cit. 23. 04. 2022]. Dostupné z: https://medium.com/@mevan.karu/you-dont-need-to-buy-a-hsm-to-see-how-it-works-2bf201f39d83.

52. KARUNANAYAKE, Mevan. *Want to know how to talk to a HSM at code level?* [online]. [cit. 23. 04. 2022]. Dostupné z: https://medium.com/@mevan.karu/want-to-know-how-to-talk-to-a-hsm-at-code-level-69cb9ba7b392.

53. KARUNANAYAKE, Mevan. *Want to know how to talk to a HSM at code level? — Part 2* [online]. [cit. 23. 04. 2022]. Dostupné z: https://medium.com/@mevan.karu/want-to-know-how-to-talk-to-a-hsm-at-code-level-part-2-433086bed1ba.

54. *HDFS Architecture Guide* [online]. [cit. 23. 04. 2023]. Dostupné z: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

55. *Cluster Mode Overview* [online]. [cit. 23. 04. 2023]. Dostupné z: https://spark.apache.org/docs/latest/cluster-overview.html.

56. *Use containers to Build, Share and Run your applications* [online]. [cit. 23. 04. 2023]. Dostupné z: https://www.docker.com/resources/what-container/.

57. *The Good and the Bad of Docker Containers* [online]. [cit. 23. 04. 2023]. Dostupné z: https://www.altexsoft.com/blog/docker-pros-and-cons/.

58. *https://learncsv.com/csv-vs-parquet* [online]. [cit. 24. 04. 2023]. Dostupné z: https://learncsv.com/csv-vs-parquet.

59. NOVOTNÝ, Pavel. *Návrh systému pro podporu SVJ* [online]. Bakalářská práce, 2021 [cit. 24. 04. 2023]. Dostupné z: https://dspace.cvut.cz/handle/10467/94539.

60. *SecurityServer Se-Series Gen2* [online]. [cit. 16. 04. 2023]. Dostupné z: http://www.safesoft.hu/index.php?menu=utimaco/hsm_se_gen2&lng=en&caption=SecurityServer.

61. *Smart Card Handling* [online]. [cit. 16. 04. 2023]. Dostupné z: https://doc.primekey.com/appliance-eidas/operations/basic-hardware-operations/smart-card-handling.

62. *Secure Keys for Data at Rest, in Use, and in Motion – Fully FIPS Certified.* [online]. [cit. 27. 04. 2023]. Dostupné z: https://www.progreso.com.sg/newsite/wp-content/uploads/2019/05/UTIMACO_Product_Brief_ESKM_A4_20211203.pdf.

Přílohy

A Internetové odkazy

- GitHub: https://github.com/novascomp/MasterThesis
- Docker Hub: https://hub.docker.com/search?q=novascomp
- Obraz GitHub repozitáře je součástí elektronické přílohy