Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Zabezpečení software a naměřených dat pro stanovená měřidla

2022
Vedoucí práce:
RNDr. Eduard Bartl, Ph.D.

Alena Svobodová
Studijní obor: Aplikovaná informatika,
prezenční forma

Bibliografické údaje

Autor:	Alena Svobodová
Název práce:	Zabezpečení software a naměřených dat pro stanovená měřidla
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2022
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	RNDr. Eduard Bartl, Ph.D.
Počet stran:	59
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Alena Svobodová
Title:	Title
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2022
Study field:	Applied Computer Science, full-time form
Supervisor:	RNDr. Eduard Bartl, Ph.D.
Page count:	59
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Cílem diplomové práce je vytvořit software pro zabezpečení dat z měřicího zařízení podle specifických požadavků daných příručkou WELMEC Guide 7.2. se záměrem integrace do stávajících softwarových řešení společnosti CROSS Zlín, a.s. V textu jsou podrobně popsány konkrétní metody zabezpečení a jejich implementace.

Synopsis

The thesis aim is to create software for securing data from a measuring device following the specific requirements given by the WELMEC Guide 7.2. intending to be integrated with the existing software solutions of the company CROSS Zlín, a.s. The text describes the theoretical background of the security features as well as the specific design and implementation of the used methods.

Klíčová slova: metrologie; zabezpečení; kryptografické podpisy; šifrování; Welmec Guide 7.2

Keywords: metrology; security; digital signatures; encryption; Welmec Guide 7.2

Chtěla bych poděkovat firmě CROSS Zlín, a.s., že mi umožnila realizovat moji diplomovou práci, zejména děkuji Ing. Václavu Machovi, Ph.D. za vedení práce v rámci firmy. Dále děkuji mému vedoucímu práce RNDr. Eduardu Bartlovi, Ph.D. z katedry infomatiky PřF UPOL za velmi cenné rady a připomínky.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Úvod

Bezpečnost hraje zásadní roli v každé aplikaci bez ohledu na její rozsah nebo použití, ale v případě legálně relevantních informací je ještě důležitější. Zabezpečení softwaru je velmi rychle se vyvíjející odvětví, které se stále přizpůsobuje novým a moderním hrozbám. Při vývoji softwaru se proto musíme neustále zabývat otázkami vhodného zabezpečení a pravidelně posuzovat úroveň zabezpečení implementovaných aplikací.

V případě stanovených měřidel, což jsou měřicí zařízení, která musí splňovat předepsané zákonné metrologické požadavky, je třeba při návrhu a implementaci softwaru provést taková opatření, aby naměřené údaje mohly být akceptovány jako platné v případech smluvních vztahů, při určování výše škody, stanovení poplatků, tarifů, daní, pro ochranu zdraví a životního prostředí a bezpečnost práce.

Metrologií a bezpečností softwaru se zabývá řada národních i mezinárodních institucí, které vydávají zákony a pokyny pro vytváření programů splňující aktuální nároky na zabezpečení. Naše národní instituce deklaruje, že software pro stanovená měřidla musí dodržovat příručku WELMEC Guide 7.2, která je vytvořena podle evropských předpisů. Dokument poskytuje základ, podle kterého jsem koncipovala tuto práci.

Cílem této práce je popsat proces návrhu a implementovat navržený software podle příslušných bezpečnostních předpisů za účelem vhodného zabezpečení softwaru i hodnot naměřených na vybraném měřidle. Diplomová práce je vytvořena ve spolupráci s firmou CROSS Zlín, a.s. pro navržení alternativy zabezpečení existujícího projektu se zobecněním na více typů stanovených měřidel. Podle relevantních požadavků jsem implementovala knihovnu pro zabezpečení naměřených dat z rychloměru.

Na začátku tohoto textu je uveden stručný přehled metrologie s vysvětlením národní a mezinárodní roviny. Následuje popis příručky WELMEC Guide 7.2, který vykládá strukturu a postup použití dokumentu. Dále jsou v části o kryptografii popsány nejběžnější algoritmy pro šifrování i hashování a je vysvětleno, jakým způsobem jsou kryptografické algoritmy použity pro vytváření digitálních podpisů. Kapitolu uzavírá přehled pojednávající o požadavcích různých zemí na délky klíčů algoritmů s ohledem na úroveň bezpečnosti, kterou poskytují.

Všechny tyto komplexní požadavky jsem zohlednila a vyvinula jsem nový software speciálně na míru pro firmu CROSS Zlín, a.s. podle zadání diplomové práce. Uvedený software je detailně popsán ve druhé části tohoto textu a je přílohou práce. Tato část je zaměřena na návrh a implementaci legálně relevantního softwaru používaného k zabezpečení dat z rychloměru. Nejprve je vysvětleno požadované použití softwaru se všemi nároky na funkčnost a strukturu dat. Využité technologie jsou uvedeny v následující části. Poté je podrobně popsán postup implementace knihovny i ukázkové aplikace. Poslední část obsahuje uživatelskou příručku, která poskytuje návod, jak pomocí implementovaných aplikací vyzkoušet funkcionalitu knihovny.

1 Metrologie

Metrologie označuje vědu o měření a jeho využití. Člení se na metrologii *fundamentální*, která představuje výzkum v doméně etalonů[1] a určování základních fyzikálních konstant, dále na metrologii *průmyslovou*, zahrnující správné fungování měřidel v průmyslu a ve výrobních procesech, a na metrologii *legální*, zajišťující přesnost měření z hlediska předepsaných technických a právních náležitostí, s cílem zajistit veřejnou záruku bezpečnosti a přesnosti měření. V příštích kapitolách je věnována pozornost právě legální metrologii, protože ta je pro tuto práci zásadní.

Význam metrologie spočívá nejen v prokazování dodržení požadavků daných zákonem a doložení shody s určenými parametry, ale také v diagnostice výrobních zařízení, zefektivňování výrobních procesů, snížení spotřeby energie, a ostatních optimalizací měření za účelem finančních úspor. V oblasti řízení kvality je významná metrologická konfirmace měřicího zařízení, pro kterou se dokládají zejména výsledky kalibrací. Každá země má vlastní systém organizací, které se starají o daná odvětví metrologie [12].

1.1 Národní metrologický systém ČR

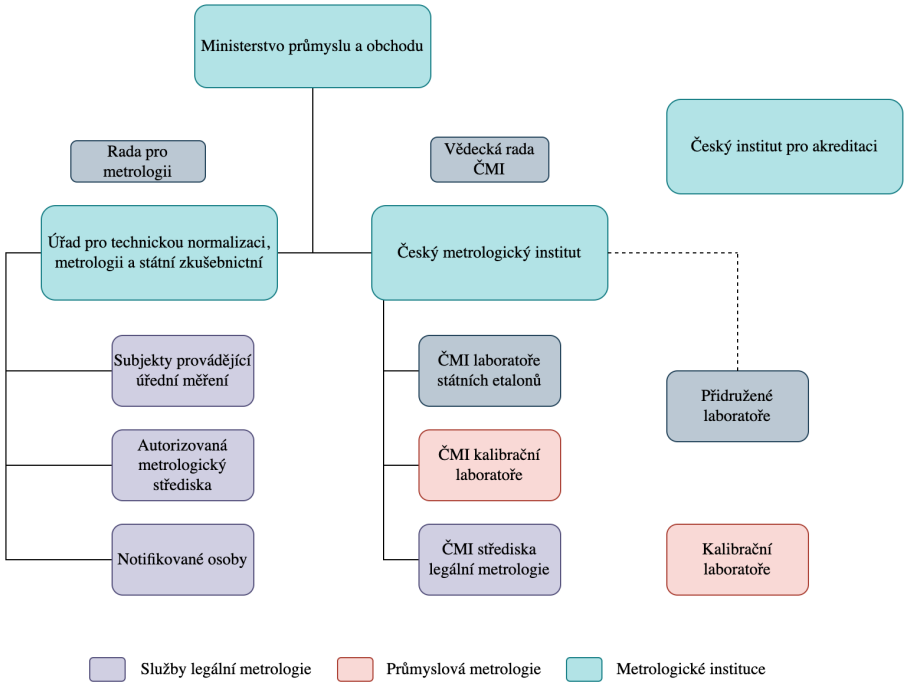
Národní metrologický systém České republiky zajišťuje správnost a jednotnost měřidel a měření pomocí technických předpisů, práv a povinností správních orgánů, právních osob i podnikajících fyzických osob. Zabývá se všemi částmi metrologie, a to fundamentální, legální i průmyslovou metrologií. Ve fundamentální metrologii jde především o vytváření a udržování standardů, naproti tomu průmyslová metrologie aplikuje měření do našich životů, průmyslu a výzkumu.

Celý systém funguje pro spotřebitele a podnikatelské subjekty, kteří tvoří základní elementy systému. Jednou z hlavních priorit je respektování oblasti zájmu spotřebitelů, pro které existuje infrastruktura výrobců a služeb. Do podnikatelských subjektů patří výrobci měřidel, opraváři i montážníci a také subjekty zajišťující služby. Stát je v tomto systému zodpovědný za projednávání a schvalování metrologické legislativy, podporu rozvoje metrologie v rámci České republiky a za mezinárodní spolupráci v metrologii [12].

Hlavní prvky Národního metrologického systému jsou Úřad pro technickou normalizaci, metrologii a státní zkušebnictví *ÚNMZ*, Český metrologický institut *ČMI* a Český institut pro akreditaci *ČIA*. Tyto úřady jsou znázorněny na obrázku 1.

Ministerstvo průmyslu a obchodu *MPO* je ústřední orgán státní správy pro technickou normalizaci, metrologii a státní zkušebnictví. Jak je znázorněno na obrázku 1, MPO řídí ÚNMZ a ČMI. Úřad pro technickou normalizaci, metrologii a státní zkušebnictví má na starost vytvoření státního programu metrologie a dohlíží na jeho realizaci. Dále zastupuje Českou republiku v mezinárodních metro-

[1] Etalon měřicí jednotky je měřidlo, sloužící k realizaci a uchovávání této jednotky a k jejímu přenosu na měřidla nižší přesnosti [5].



Obrázek 1: Národní metrologický systém [12]

logických organizacích, také autorizuje subjekty pro kontrolu měřidel, pověřuje subjekty k uchování státních etalonů, a kontroluje činnost ČMI. Autorizovaná metrologická střediska *AMS* jsou subjekty, které po předložení žádosti ÚNMZ schválil pro ověřování stanovených měřidel.

Český metrologický institut plní funkci národního metrologického institutu, zabývá se fundamentální metrologií, převodem jednotek a legální metrologií. Mezi jeho základní povinnosti patří metrologický výzkum, uchování státních etalonů, schvalování a ověřování *stanovených měřidel*,[2] provádí dozor pro státní metrologii, kontroluje hotově balené zboží, posuzuje dodržení technických požadavků na výrobu, a také vydává *opatření obecné povahy*.[3] Poslední důležitá entita, Český institut pro akreditaci, slouží především pro prokazování odborné způsobilosti kalibračních a zkušebních laboratoří, certifikačních a inspekčních orgánů. Akreditované kalibrační laboratoře *AKL* jsou subjekty, u nichž ČMI pravidelně kontroluje dodržování norem a uznává jejich technickou způsobilost a systém kvality.

Všechny zmíněné úřady a instituce slouží pro zajištění správnosti a jednotnosti měření, a tyto snahy jsou společné nejen v rámci našeho státu, ale také na mezinárodní úrovni. Pro rozvoj oblastí vědy, průmyslu i obchodu vzniklo několik mezinárodních organizací a konvencí, které slouží pro sjednocení měření mezi různými státy.

[2]Stanovená měřidla označují měřidla, která Ministerstvo průmyslu a obchodu stanoví vyhláškou k povinnému ověřování s ohledem na jejich význam [5].

[3]OOP je správní akt s konkrétně určeným předmětem a vymezeným okruhem adresátů stojící na pomezí mezi právním předpisem a individuálním rozhodnutím.

1.2 Mezinárodní spolupráce

Národní metrologický systém ČR je podobný systémům zemí Evropské unie a navazuje mezinárodní spolupráci s organizacemi působícími v metrologii v EU i ve světě. První dohoda, která měla dalekosáhlý dopad, je Metrická konvence. Jedná se o mezinárodní smlouvu, kterou uzavřelo 17 zemí dne 20.5.1875. Tento den je označován jako Mezinárodní den metrologie. Nyní, k lednu 2021 má 63 členských států a dalších 39 přidružených.

Hlavní význam Metrické konvence spočívá ve vytvoření univerzální dekadické soustavy jednotek, založené na jednotce jeden metr a jeho násobcích, která dala prostor pro rozvoj vědy, průmyslu i obchodu v druhé polovině 19. století. Smlouva také podnítila vznik několika orgánů jako Mezinárodní úřad pro váhy a míry *BIPM*, Všeobecné konference pro váhy a míry a Mezinárodní výbor pro váhy a míry. Dnes organizace s názvem BIPM spravuje mezinárodní systém jednotek měření SI (soustava SI), a tak přispívá ke sjednocování měřicích jednotek, vývoji etalonů a zajišťování jejich ekvivalence [5].

Další významnou organizací je Mezinárodní organizace pro legální metrologii *OIML*, která vznikla v polovině 20. století za účelem unifikace předpisů v oblasti legální metrologie na globální úrovni. Její klíčový přínos spočívá ve vytváření norem, které slouží jako doporučení pro tvorbu národních předpisů pro technické požadavky a metody zkoušení regulovaných měřidel (v ČR *stanovená měřidla*). Pro ověření, zda dané měřidlo splňuje OIML předpisy, je k dispozici certifikační systém OIML.

Evropská směrnice MID o měřicích přístrojích nahrazuje národní předpisy členských států a má za cíl harmonizovat požadavky na nová měřidla pro usnadnění mezinárodního obchodu. Sdružení WELMEC, které vzniklo uvnitř Evropské unie, má zásadní přínos ve vytváření návodů pro implementaci evropské legislativy danou směrnicemi NAWID a MID. V oblasti softwaru se WELMEC věnuje sjednocení postupů pro schvalování a posuzování shody software a jeho komponent.

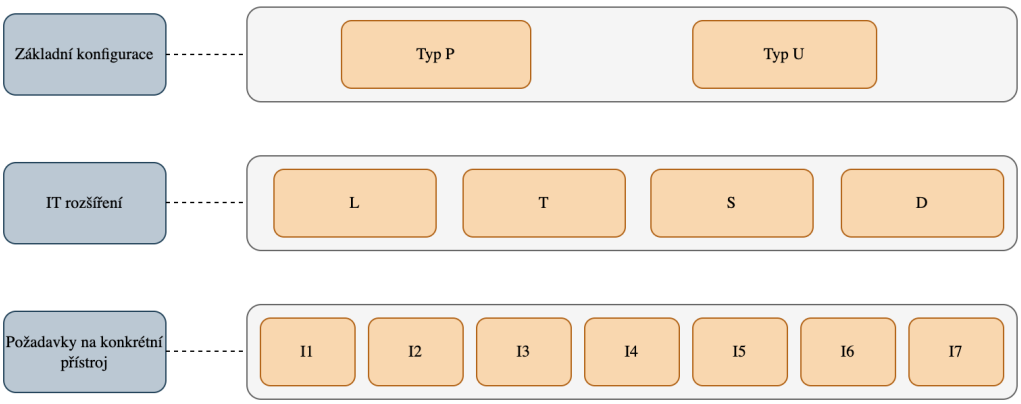
Díky mezinárodním organizacím a smlouvám, které mají všechny totožný cíl, sjednotit požadavky pro měřicí zařízení a tak podpořit mezinárodní obchod, jsou požadavky kladené na měřidla obdobné, a lze k nim vytvářet návody, jakým je WELMEC Guide 7.2, který slouží jako příručka pro zabezpečení softwaru podle evropské legislativy.

2 Zabezpečení SW podle Welmec Guide 7.2

WELMEC Guide 7.2 je jedna z mnoha příruček, které sdružení vydává pro výrobce měřicích přístrojů i pro notifikované osoby zodpovědné za posuzování shody výrobků. Ačkoliv má příručka pouze charakter doporučení, popisuje jak splnit požadavky Evropské unie v oblasti legální metrologie, protože je postavená na základě požadavků na software a jeho validaci vydanými v směrnici MID. Dokument je poměrně rozsáhlý a pro lepší orientaci jsou následující podkapitoly věnovány struktuře dokumentu a popisu, jak s ním pracovat. Tato kapitola je zpracována podle WELMEC Guide 7.2 verze z roku 2020 [18].

2.1 Skupiny požadavků

Celý dokument je koncipován jako strukturovaný soubor požadavků. Obrázek 2 znázorňuje tuto strukturu. Nejprve jsou zařízení rozdělena podle základní konfigurace na *Typ P* a *Typ U*. Dále jsou požadavky rozděleny podle IT konfigurace, tzv. rozšíření *L, T, S, D*, která jsou pojmenována jako zkratky anglických názvů. V poslední řadě jsou požadavky na konkrétní přístroje, rozšíření *I1–I7*.



Obrázek 2: Struktura dokumentu Welmec Guide 7.2 [18]

První třída požadavků je určena pro všechna zařízení: vždy nejprve určíme, o který typ zařízení se jedná a které požadavky z toho plynou. Určíme zda se jedná jednoúčelové zařízení nebo o měřicí přístroj používající univerzální počítač viz sekce 2.4. Druhá skupina zahrnuje požadavky pro možná rozšíření: dlouhodobé uložení naměřených dat (*L*ong-term Storage of Measurement Data), přenos naměřených dat (*T*ransmission of Measurement Data via Communication Networks), stahování softwaru (*D*ownload of Legally Relevant Software) a oddělení softwaru (*S*oftware Separation). Z této skupiny vybereme rozšíření pouze tehdy, když zařízení disponuje příslušnou funkcionalitou. Poslední je skupina požadavků pro konkrétní zařízení, které jsou číslovány shodně se směrnicí MID, vztahují se

pouze na jednotlivé určené měřicí přístroje, a tak nejsou relevatní pro praktickou část práce.

Rozdělení zařízení podle základní konfigurace je podrobněji popsáno v podkapitole 2.4. Někdy není na první pohled zřejmé, do které kategorie měřicí zařízení spadá, a proto je přiloženo několik otázek, podle jejichž odpovědí určíme jednoznačně typ základní konfigurace zařízení.

Obrázek 3: Postup vybírání požadavků [18]

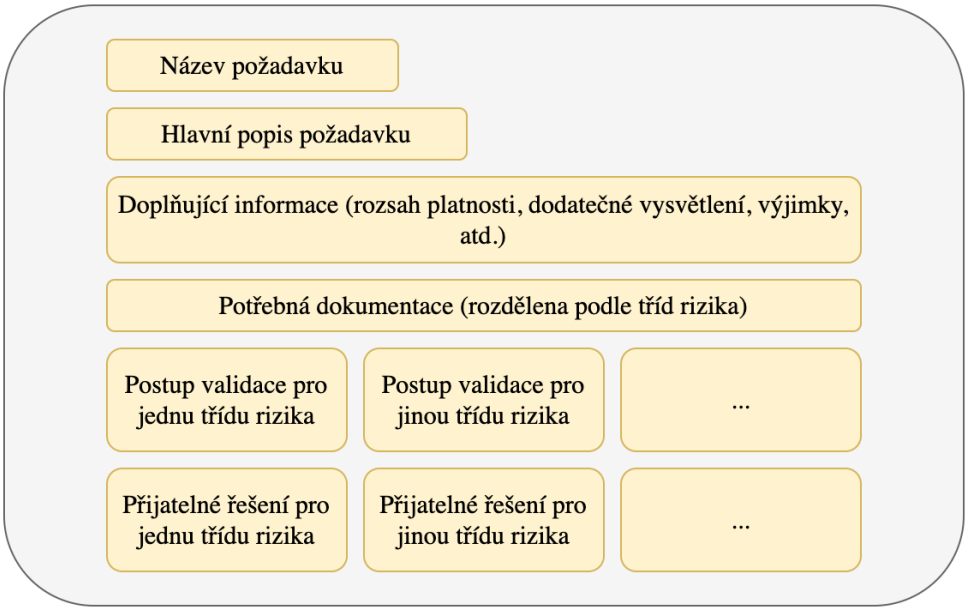
Jak tedy vybrat, které požadavky jsou pro konkrétní instanci relevatní? Obrázek 3 ukazuje postup při výběru tříd požadavků, popsaných výše na obrázku 2, která se vztahují na konkrétní případ. Každá ze tří kategorií obsahuje seznam číslovaných požadavků, které je nutno všechny splnit v rámci dané třídy. Dále jsou jednotlivé požadavky rozděleny do šesti rizikových tříd *A*–*F*, kde třída *A* představuje nejnižší stupeň rizika. V současné době se třídy *A*, *E* a *F* nevztahují na zařízení regulované směrnicí MID, ale slouží pro budoucí případy použití. Každý konkrétní měřicí přístroj má přiřazenou jednu třídu rizika, která určuje míru zabezpečení všech požadavků. To znamená, že každý jednotlivý požadavek je dále ještě rozdělen podle stupně rizika a jemu příslušnému zabezpečení. Například, silniční rychloměry spadají běžně do třídy D, a tak u každého požadavku relevatního k silničnímu rychloměru musíme splnit úroveň zabezpečení podle této třídy.

Při práci s příručkou Welmec 7.2 je tedy doporučeno postupovat podle obrázku 3: Nejprve vybrat, zda se jedná o zařízení *Typu P* nebo *Typu U*; dále určit, která z konfiguračních rozšíření *L, T, S, D* se na dané zařízení vztahují; pak zjistit, zda zařízení podléhá požadavkům konkrétních tříd *I*, a v poslední řadě zvolit, kterou třídu rizika aplikovat.

Přesně tak je postupováno v části textu, která se věnuje výběru relevantních požadavků k silničním rychloměrům, mezi něž patří i pro tuto práci zvolené měřicí zařízení. Nyní je zřejmá globální struktura dokumentu a skupin požadavků, a v následující podkapitole je popsán formát jednotlivého požadavku.

2.2 Struktura požadavku

Každý požadavek podléhá struktuře znázorněné na obrázku 4. Začíná názvem a popisem jednotlivého požadavku, dále obsahuje dodatečné informace vysvětlující výjimky a rozsah použití, nutnou dokumentaci, postup validace a možné přípustné řešení. Obsah jednotlivého bloku může být rozdělen podle jednotlivých tříd rizika.



Obrázek 4: Struktura požadavku [18]

Účelem takto definovaného požadavku je oznámit výrobci i notifikované osobě, jaké jsou technické informace a postup validace. Požadavek představuje nejen minimální předpoklad, ale také by neměly být kladeny nároky nad jeho rámec.

2.3 Třídy rizika

Jednotlivé požadavky se dělí podle stupně rizika do tříd pro software měřicího přístroje. Každému měřidlu je přiřazena třída rizika, od které se dále odvíjejí požadavky na zabezpečení softwaru. Posuzují se tři faktory nebezpečí ve třech stupních *nízký*, *střední*, *vysoký*, ze všech možných kombinací je vybráno šest a ty jsou pojmenovány *A*–*F*. Faktory nebezpečí jsou *nedostatečné zabezpečení softwaru*, *nedostatečné přezkoušení softwaru* a *neshoda s typem*, a jim odpovídají nutná bezpečnostní opatření [18].

V tabulce 1 jsou znázorněny třídy rizika *A*–*F* a jim odpovídající stupně *zabezpečení softwaru*, *přezkoušení softwaru* a *shoda softwaru*. Pro zabezpečení softwaru na nízkém stupni nejsou žádné speciální požadavky proti úmyslným změnám, dále pro střední stupeň je třeba software chránit proti úmyslným změnám za použití volně dostupných nástrojů, jakým je například textový editor. Pro vysoký stupeň je nutné zabezpečení proti úmyslným změnám za použití pokročilejších nástrojů, například ladících programů nebo nástrojů pro vývoj softwaru. Při přezkoušení softwaru s nízkým stupněm nejsou kladeny další požadavky nad obvyklé testy funkcí zařízení. Pro střední stupeň se provádí přezkoušení podle dokumentace, která obsahuje popis funkcí i parametrů, ze kterých se náhodně vyberou některé pro kontrolu přesnosti dokumentace a efektivity implementova-

Tabulka 1: Třídy rizika [18]

Třída rizika	Zabezpečení SW	Přezkoušení SW	Shoda SW
A	nízká	nízká	nízká
B	střední	střední	nízká
C	střední	střední	střední
D	vysoká	střední	střední
E	vysoká	vysoká	střední
F	vysoká	vysoká	vysoká

ných prostředků pro ochranu. Požadavky na shodu softwaru při nízkém stupni jsou takové, že *legálně relevantní software*[4] konkrétního přístroje musí odpovídat technické dokumentaci legálně relevatního softwaru daného typu. Binární kód se nezkoumá. Naproti tomu při středním stupni musí být binární kód legálně relevantní části totožný se softwarem daného typu, v případě, že zařízení splňuje požadavky rozšíření S, je zde možné rozdělení softwaru na legálně relevantní a legálně nerelevantní část. Vysoký stupeň takové rozdělení nedovoluje a binární kód celého softwaru musí být totožný se softwarem konkrétního typu [18].

Ze všech možných kombinací stupňů jsou relevantní právě ty uvedené v tabulce 1, pokrývající všechny třídy, které spadají pod směrnice MID. Každé zařízení musí mít danou třídu zabezpečení, protože ta specifikuje podmínky pro dané požadavky. Nejčastěji jsou používané třídy B, C, D. Třída E se zabývá kontrolou zdrojového kódu, a třída F navíc nedovoluje rozdělení softwaru. Pro zařízení *typu* U je nejnižší možná třída rizika C, protože je zde větší riziko.

2.4 Jednoúčelové a univerzální měřicí přístroje

První krok při určování požadavků je rozlišit, zda se jedná o jednoúčelové zařízení, nebo o měřicí přístroj používající univerzální počítač. Pro zjednodušení je zde pět otázek, které pomohou určit, o který typ zařízení se jedná.

1. Je celý software zařízení určený pro účely měření?
2. Jsou splněné požadavky pro začlenění operačního systému (nebo podsystému)?
3. Je možné zařízení přepnout do režimu, který nepodléhá legální kontrole, je přístup do OS zamezený?
4. Jsou implementované programy a softwarové prostředí neměnné (kromě aktualizací)?
5. Je zařízení vybavené programovacími nástroji?

[4] viz kapitola 2.5.3

Pokud je odpověď na otázky 1–4 *ano*, a odpověď na otázku 5 *ne*, tak se jedná o jednoúčelové zařízení, tedy zaříení *typu P*. V jakémkoliv jiném případě se jedná o zařízení *typu U*. V další části textu jsou popsány základní vlastnosti obou typů zařízení.

2.4.1 Zařízení typu P

Měřicí zařízení *typu P* má vestavěný IT systém, například na základě mikroprocesoru, a všechny komponenty systému jsou přístupné k vyhodnocení. Daný software je implemetovaný výhradně pro účely měření, včetně funkcionality pro zabezpečení, přenos i stahování. Uživatelské rozhraní slouží pouze pro měření a má dva režimy, základní režim podléhající legální kontrole a pomocný režim, který legální kontrole nepodléhá. Může obsahovat operační systém, pokud je legálně relevatní software zodpovědný za veškerou komunikaci mezi ním a OS, dále neumožní změnit programy, parametry, data a běžící programy, a neumožní ani změny prostředí legálně relevantního softwaru. Ochrana přístupu musí být předem daná, nikoliv výsledkem pozdějšího nastavení komponent. Softwarové prostředí není možné měnit, neexistují vnitřní nebo vnější nástroje na změnu SW, který je definovaný jako vestavěný. Stahování softwaru je možné za předpokladu, že jsou splněny všechny podmínky rozšíření *D* [9].

2.4.2 Zařízení typu U

Požadavky pro zařízení *typu U* jsou určené pro veškerá měřicí zařízení, která používají univerzální počítač pro své měření a příslušnou funkcionalitu. Počítač může fungovat samostatně, nebo být součástí uzavřené či otevřené sítě (například ethernet, LAN, internet). Měřicí zařízení může být k univerzálnímu počítači připojeno přes komunikační rozhraní. Funkcionalita, která nepodléhá legální kontrole, může být součástí uživatelského rozhraní. Paměť počítače může být pevná (HDD), vyjímatelná (USB), nebo vzdálená. Operační systém je obvykle součástí zařízení a další spuštěné programy, mimo software určený pro měření, mohou být spuštěny. Operační systém a ovladače, které nejsou určené pro úlohu měření, nejsou považovány za legálně relevatní software [9].

2.5 Rozšířená IT konfigurace

Po obecných požadavcích pro každé měřicí zařízení budou dále popsána volitelná rozšíření, která nemusí být vždy součástí konkrétní instance. Jedná se o dlouhodobé uložení naměřených hodnot (L), přenos naměřených dat pomocí komunikačního rozhaní (T), rozdělení software na legálně relevantní a legálně nerelevatní části (S) a stažení legálně relevatního software (D).

2.5.1 Dlouhodobé uložení naměřených hodnot (L)

Dlouhodobé uložení je ve Welmec Guide 7.2 definováno jako uložení hodnot od okamžiku dokončení procesu měření až po dokončení všech legálně relevantních procesů. Termín také označuje uložení dat po zmíněném časovém úseku. V tabulce 2 jsou popsány tři různé technické konfigurace pro dlouhodobé uložení dat. Tato rozšíření se aplikuje pouze když je navrženo dlouhodobé ukládání naměřených dat, tedy pokud zpětně využíváme naměřené údaje pro legálně relevantní účely.

Pro jednoúčelová zařízení je typické využití integrované paměti, která je součástí základního vybavení měřicího zařízení. Příkladem takové paměti je RAM, FLASH, HDD a EEPROM. V případě univerzálního počítače je obvykle použita vnitřní pamět počítače, například HDD. Poslední variantou je vyjímatelná paměť, kterou lze použít jak pro jednoúčelové zařízení, tak pro měřicí přístroje používající univerzální počítač. Příkladem mohou být USB paměť, SD karta, nebo vzdálená databáze připojená přes internetovou síť. Přístroj, který načte a zpracovává data z vyjímatelné paměti nebo externího uložiště, musí být také předmětem legální kontroly [9].

Tabulka 2: Typy paměti pro dlouhodobé uložení naměřených dat [9]

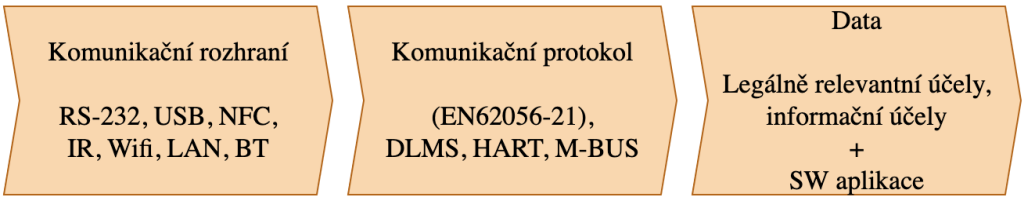
Typ paměti	Obvyklé použití	Příklady
Integrovaná paměť	Typ P	RAM, FLASH, HDD, EEPROM
Paměť univerzálního počítače	Typ U	Paměť kterou lze vyjmou a kopírovat
Vyjímatelná paměť, externí uložiště	Typ P, Typ U	USB, SD karta, vzdálená databáze

2.5.2 Přenos naměřených dat komunikačními sítěmi (T)

Požadavky v tomto rozšíření se vztahují na všechna zařízení, která přenáší naměřená data z měřicího přístroje do jiného zařízení, které je následně zpracovává pro legálně relevantní účely. Pokud data příjemce nepoužívá pro legálně relevantní účely, pak není toto rozšíření nutné. Přenos dat je možný po uzavřené či otevřené síti.

Pro uzavřenou síť platí, že počet účastníků je pevný, každý z nich má jednoznačnou identifikaci, a všechna zařízení v síti podléhají legální kontrole. Pro každé zařízení musí být známa jeho identita, funkce a umístění. Naproti tomu v otevřené síti může být libovolný počet účastníků, a nemusí být známa identita, funkce ani umístění jednotlivých zařízení.

Pokud je potřeba použít přenášená data, pak celá komunikace zaznačena na obrázku 5 spadá do legálně relevantního software. Toto rozšíření se zabývá pouze přenosem a neřeší, zda výsledek zobrazení na zařízení příjemce je legálně



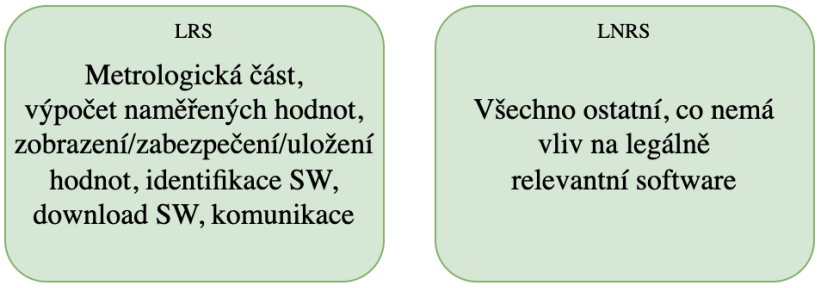
Obrázek 5: Přenos naměřených dat komunikačními sítěmi [9]

relevantní. Pokud příjemce data používá pro legálně relevantní účely, pak i to zařízení musí být validované.

Fyzická realizace přenosu se dá rozdělit na drátovou, například LAN, USB, RS-485, a bezdrátovou, jako je Wifi a BT. Pokud se jedná o přenos drátový, s fyzicky zapečetěnými konektory, pak není potřeba rozšíření T, protože kabely jsou uzavřené a tvoří součást měřicího zařízení. V opačném případě, pokud je možné konektory přepojit nebo se jedná o bezdrátové připojení, pak je rozšíření T nutné [9].

2.5.3 Rozdělení softwaru (S)

Volitelné rozšíření je rozdělení softwaru na legálně relevatní (LRS) a legálně nerelevantní (LNRS) části. Rozdělení umožňuje funkce, které nespadají pod proces měření dat, oddělit od legálně relevantních. Příkladem legálně nerelevantní funkcionality softwaru měřicího zařízení může být nastavení jasu displeje, jazykové menu a jiné informační funkce. Značná výhoda použití rozdělení softwaru je možnost legálně nerelevantní část kdykoliv aktualizovat bez potřeby schválení od *notifikované osoby*.[5]



Obrázek 6: Rozdělení softwaru na LRS a LNRS [9]

Rozdělení je reprezentováno na obrázku 6, kde je v levé části vypsáno, co patří do legálně relevantního softwaru. Příklady jsou údaje na displeji, zobrazování

[5]Notifikované osoby (také *oznámené subjekty*) jsou subjekty schopné zkoušek, kontrolovat a certifikovat výrobky.

identifikace či přenášení legálně relevantních údajů přes komunikační rozhraní. Vše ostatní spadá pod legálně nerelevantní software, například jazykové menu, podsvícení displeje a zobrazení údajů.

2.5.4 Stahování legálně relevantního softwaru (D)

Poslední rozšíření dané kategorie je stahování legálně relevantního softwaru. Rozšíření *D* se týká pouze zařízení umožňujících download softwaru bez porušení plomby. V případě rozdělení softwaru na legálně relevantní a nerelevantní, podle předchozího rozšíření *S*, se toto rozšíření týká jen legálně relevantní části. Legálně nerelevantní software je možné aktualizovat kdykoliv. Výhoda je zjevná při aktualizace softwaru na mnoha zařízeních, bez porušení plomby, což vede ke snažší distribuci. Je třeba dodat, že ne všechny země EU dovolují aktualizovat software bez porušení plomby.

Příklad realizace může být takový, že měřicí zařízení obsahuje legálně relevantní software *D*, který má za úkol zabezpečit proces stažení legálně relevantního softwaru *N*. Zařízení také obsahuje záznamník událostí, který nelze bez porušení plomby smazat, obsahující počet možných stažení legálně relevantního softwaru *N*. Při zaplnění záznamníku neumožní zařízení další aktualizaci. Software *N* by měl obsahovat klíč pro ověření důvěryhodnosti zdroje [9].

3 Kryptografické podpisy a šifrování

V této kapitole jsou předpokládány základní znalosti kryptografie a šifrování. Podrobněji jsou popsány algoritmy AES a RSA, protože jsou jedny z nejpoužívanějších v reálných aplikacích, a také jsou použity v praktické části práce. Dále budou připomenuty hashovací funkce a jejich využití při digitálním podepisování. Nakonec budou uváženy různé stupně bezpečnosti běžných algoritmů v závislosti na délce klíče. Pro porovnání doporučení různých zemí vezmeme požadavky americké NIST (*National Institute of Standards and Technology*), francouzské ANSSI a německé BSI a vyhodnotíme, které délky klíče jsou v dnešní době považovány za bezpečné.

3.1 Algoritmus AES

Nejpoužívanější bloková šifra, dokonce nejpoužívanější šifra vůbec, zvaná *Advanced Encryption Standard* neboli AES, je používaná od roku 2000 kdy nahradila jejího předchůdce DES (*Data Encryption Standard*). Algoritmus byl vyvinut v Belgii pod původním jménem Rijndael podle autorů Rijmen a Daemen a dnes jej lze nalézt ve většině elektronických zařízeních. Algoritmus AES byl navržen, aby překonal dvě nevýhody algoritmu DES. Za prvé nízká efektivnost, protože pro dosažení 112 bitové bezpečnosti je potřeba klíč délky 168 bitů, a dále nízká rychlost na běžných procesorech, protože byl optimalizován na integrované obvody [1].

Blokové šifry se skládají z šifrovacího algoritmu a dešifrovacího algoritmu. Šifrovací algoritmus (E) vezme klíč (K) a blok otevřeného textu (P) a vytvoří blok šifrovaného textu (C). Matematicky značíme

$$C = E(K, P).$$

Dešifrovací algoritmus (D) je inverzní k šifrovacímu a dešifruje zprávu do původního otevřeného textu (P). Tuto operaci značíme

$$P = D(K, C).$$

Jelikož jsou algoritmy k sobě inverzní, tak většinou obsahují podobné operace.

Pro bezpečnost takové šifry je nutné, aby byla pseudo-náhodná permutace, neboli dokud je klíč uchován v tajnosti, pak by útočník neměl být schopen vypočítat výstup blokové šifry pro jakýkoliv vstup. Obecně by útočník neměl být schopen odhalit žádný vzor ze vstupních a výstupních bloků.

Dva hlavní parametry bezpečnosti blokové šifry jsou velikost bloku a velikost klíče. Pro AES jsou typické bloky velikosti 128 bitů. Bloky nemohou být moc veliké, aby se minimalizovala *paměťová stopa*. Pro zpracování 128 bitového bloku je potřeba 128 bitů paměti. Tato velikost je dostatečně malá, aby se vešla na registry většiny CPU, nebo aby byla implementovaná na hardwarových obvodech. Délky bloků 64, 128, i 512 jsou všechny dostatečně krátké pro efektivní

implementaci ve většině případů. Pokud bychom chtěli bloky delší, mělo by to viditelný dopad na cenu i výkon implementací [1].

Bloková šifra probíhá v sekvencích kol. Jednotlivá kola se skládají z jednoduché transformace, která není náročná ani na výpočet, ani na implementaci.

Například bloková šifra se třemi koly zašifruje otevřený text následujícím výpočtem

$$C = R_3(R_2(R_1(P))),$$

kde R_1, R_2 a R_3 představují jednotlivá kola a P je otevřený text. Každé kolo má k sobě inverzní operaci, aby bylo možné na straně příjemce zprávu dešifrovat do původní podoby. Přesněji

$$P = iR_1(iR_2(iR_3(C))),$$

kde iR_1 je inverzní operace k R_1, a obdobně pro iR_2 a iR_3. Operace jednotlivých kol jsou totožné, liší se pouze v *podklíči*. Dvě kola s dvěma podklíči budou mít různé výsledky při stejném vstupním řetězci. Podklíče jsou odvozeny z hlavního klíče v procesu *expanze*, a jednotlivá kola používají příslušný podklíč. Například kolo R_1 používá klíč K_1, a tak dále.

Algoritmus AES používá bloky 128 bitů a tajný klíč délky 128, 192, nebo 256 bitů. Nejpoužívanější varianta zůstáva klíč délky 128, z prostého důvodu, že je šifrování o něco rychlejší, a rozdíly v bezpečnosti mezi 128 a 256 bitovými klíči nejsou z pohledu většiny aplikací významné. Na rozdíl od jiných šifer pracuje AES s jednotlivými byty namísto jednotlivých bitů. 16 bytový otevřený text zpracovává jako dvojrozměrné pole bytů. Počet kol transformací je 10 kol pro 128 bitový klíč, 12 pro 192 bitový klíč a 14 pro 156 bitový klíč. Algoritmus lze rozdělit do několika kroků:

1. expanze klíče;
2. inicializační část
   - přidání podklíče;
3. iterační část
   - záměna bytů,
   - prohození řádků,
   - kombinování sloupců,
   - přidání podklíče;
4. závěrečná část
   - záměna bytů,
   - prohození řádků,
   - přidání podklíče.

V prvním kroce je provedena expanze klíče, kde jsou z hlavního klíče odvozeny podklíče. Dále se opakují operace *záměna bytů*, *prohození řádků*, *kombinování sloupců* a *přidání podklíče*. V inicializační části se provede *přidání podklíčů* (Add Round Key), neboli každý byte je zkombinován s podklíčem za pomocí operace XOR nad všemi bity. Dále proběhne iterační část (9, 11 nebo 12 iterací v závislosti na délce klíče), která se skládá ze *záměny bytů* (Sub Bytes), *prohození řádků* (Shift Rows), *kombinování sloupců* (Mix Columns) a *přidání podklíče* (Add Round Key). *Záměna bytů* je nelineární krok, při kterém je každý byte nahrazen jiným podle vyhledávací tabulky. *Prohození řádků* představuje posunutí každého řádku o určitý počet kroků. Procedura *kombinování sloupců* zkombinuje čtyři byty v každém sloupci. Poslední krok v iterační části je *přidání podklíče*. Závěrečná část se skládá ze *záměny bytů*, *prohození řádků* a *přidání podklíče*.

Různé implementace algoritmu AES nevyžadují výše zmíněné funkce v této podobě. Pro zefektivnění výpočtu jsou zavedeny vyhledávací tabulky nebo nativní instrukce. Implementace AES, které používají vyhledávací tabulky, nahrazují sekvenci procedur *záměna bytů–prohození řádků–kombinování sloupců* kombinací operací XOR a vyhledávání v tabulkách, které jsou napevno uloženy v programu a načteny do paměti v průběhu šifrování. Veliká výhoda je mnohem rychlejší výpočet, nevýhoda je však zranitelnost při *cache-timing útocích*, které zneužívají rozdíly v čase, kdy program čte a zapisuje v cache paměti. Rychlost přístupu k elementům je závislá na jejich relativní poloze v cache paměti. Nativní instrukce AES-NI tuto zranitelnost eliminují. Jedná se o rozšíření assembleru, jazyku symbolických adres, kde jediná instrukce assembleru AESENC provede jedno kolo algoritmu AES. Díky nativním instrukcím jsou tyto implementace deset krát rychlejší [1].

V dnešní době je AES považován za velmi bezpečnou blokovou šifru. Bezpečnost spočívá v závislosti všech výstupních bitů na všech vstupních bitech komplexním pseudonáhodným způsobem. Procedura *kombinování sloupců* zajistí maximální rozptyl a *záměna bytů* přidává nelineární rozměr, což zabezpečuje algoritmus proti mnohým třídám běžných útoků.

3.2 Algoritmus RSA

RSA je dnes jedna z nejúspěšnějších asymetrických šifer dnes. Nese název podle příjmení Rivest, Shamir a Adleman, kteří v roce 1977 šifru publikovali. Na rozdíl od dřívějšího symetrického šifrování používá asymetrické šifrování dva klíče: *veřejný klíč* pro šifrování zpráv a *soukromý klíč* potřebný pro dešifrování zpráv šifrovaných veřejným klíčem. O rok dříve přišli s myšlenkou veřejných klíčů Diffie a Hellman, ale jejich řešení bylo pouze pro výměnu soukromého klíče přes nezabezpečené rozhraní, nikoliv pro šifrování.

Základem RSA je aritmetický trik. Jedná se o jednosměrnou funkci s padacími dvířky, pro kterou platí, že je jednoduché ji vyčíslit, ale velmi obtížné z výsledku odvodit vstup, pokud neznáme informaci navíc, v našem případě soukromý klíč. Myšlenka RSA spočívá ve faktu, že rozložit velké číslo na součin prvočísel, neboli

faktorizace, je velmi náročná úloha. Z čísla $n = pq$ nelze v rozumném čase zjistit činitele p a q, neboť známé algoritmy pro faktorizaci nepracují v polynomiálním čase (vzhledem k velikosti binárního zápisu čísla n).

Při modulu n a číslu e, zvaným veřejný exponent, transformuje algoritmus číslo x, které patří do zbytkové třídy $\mathbb{Z}_n^*$ na číslo $y = x^e \bmod n$. Dvojice čísel n a e představují RSA veřejný klíč. Pro zpětné získání hodnoty x z hodnoty y použijeme číslo d pro výpočet

$$y^d \bmod n = (x^e)^d \bmod n = x^{ed} \bmod n = x.$$

Číslo d jsou zadní vrátka, která nám umožňují zprávy dešifrovat, nazývá se soukromý exponent, a je součástí soukromého RSA klíče. Vztah čísel e a d je takový, že jejich součin je 1, a proto $x^{ed} \mod n = x$ pro každé x. Přesněji potřebujeme $ed = 1 \mod \varphi(n)$, kde $\varphi(n)$ značí Eulerovu funkci, pro získání $x^{ed} = x^1 = x$. Používáme modulo $\varphi(n)$ místo modulo n, protože se exponenty chovají jako indexy elementů $\mathbb{Z}_n^*$, nikoliv jako jednotlivé elementy. Protože $\mathbb{Z}_n^*$ má $\varphi(n)$ elementů, index musí být menší než $\varphi(n)$.

Pro bezpečnost algoritmu RSA je hodnota $\varphi(n)$ zásadní. Naleznutí hodnoty $\varphi(n)$ pro RSA modulo n znamená prolomení šifry, protože z $\varphi(n)$ a e lze jednoduše odvodit soukromý exponent d tím, že se spočítá prvek inverzní k prvku e.

Proces generování dvojice klíčů spočívá ve zvolení dvou náhodných prvočísel p a q, pro která spočítáme hodnotu $\varphi(n)$. Dále zvolíme prvočíslo e ze třídy $\varphi(n)$ a číslo d získáme jako inverzní k e. Takto vygenerujeme veřejný klíč (modulo n a veřejný exponent e) a jeho soukromý klíč (soukromý exponent d). Hodnoty p a q by měly být také uchovány v tajnosti, protože z nich lze odvodit $\varphi(n)$ a přeneseně pak i hodnotu d.

Bezpečnost algoritmu závisí na třech faktorech: na velikosti čísla n, na volbě prvočísel p a q, a na způsobu použití zadních vrátek. V případě zvolení malého čísla n je faktorizace proveditelná v reálném čase a soukromý klíč je vyzrazen. Nejmenší doporučená velikost n je 2048 bitů, ale preferuje se velikost 4096 bitů. Hodnoty p a q by měly být nezávislá, přibližně stejně velká prvočísla. Když jsou malá, nebo blízko u sebe, pak zvyšujeme riziko prolomení. V poslední řade by se RSA jednosměrná funkce s vrátky neměla používat v této podobě pro šifrování [1].

Pro šifrování se RSA používá v kombinaci se symetrickým šifrováním, kde RSA použijeme pro šifrování symetrického klíče, který je použit pro skutečné šifrování zprávy (například AES). Základní RSA šifrování, kdy otevřený text obsahuje pouze zprávu, kterou chceme šifrovat, obsahuje několik bezpečnostních rizik. Například pro zašifrování textového řetězce *RSA* jej nejprve převedeme na číslo tím že spojíme ASCII kódy jednotlivých písmen jako byty (R byte 52, S byte 53, A byte 410). Získáme byte string 525341, který převedeme do desítkové soustavy na 5395265 a dále šifrujeme výpočtem $5395265^e \bmod n$. Bez soukromého klíče nemáme možnost zjistit původní zprávu.

Základní RSA, jak je výše popsán, je deterministický, což znamená, že pokud zašifrujeme stejný otevřený text drakrát, v obou případech dostaneme stejnou šifru jako výstup. Další problém spočívá v možnosti násobení dvou šifer.

Uvážíme-li dvě šifry $y_1 = x_1^e \mod n$ a $y_2 = x_2^e \mod n$, tak lze získat násobek $x_1 \cdot x_2$ vynásobením šifer následujícím způsobem:

$$y_1 \cdot y_2 \mod n = x_1^e \cdot x_2^e \mod n = (x_1 \cdot x_2)^e \mod n.$$

Výsledek je $(x_1 \cdot x_2)^e \mod n$, neboli šifrování zprávy $x_1 \cdot x_2 \mod n$. Takže útočník může vytvořit validní šifru ze dvou různých šifer, což narušuje bezpečnost RSA algoritmu.

Jako řešení tohoto problému se používá RSA spolu s náhodnými daty zvanými *padding*. Rozšíření RSA s paddingem se nazývá RSA-OAEP (Optimal Asymmetric Encryption Padding) a díky němu se algoritmus stává bezpečnější tím, že není deterministický, ale pravděpodobnostní.

3.3 Hashovací funkce

Jedná se o jednosměrnou funkci, ale na rozdíl od jednosměrných funkcí z předchozí podkapitoly nemá zadní vrátka, což znamená, že je výpočetně nemožné zjistit původní vstup z výstupu hashovací funkce. Hashovací funkce mají široké využití od identifikace totožných souborů na cloudových uložištích, identifikace souborů v Gitovském[6] repozitáři, analýzy zda jsou diginální dokumenty beze změny, po využití Bitcoinem a *proof of work*[7] systémy. Hashovací funkce bere vstup libovolné délky a vrací krátký řetězec konstantní délky (typicky 256 nebo 512 bitů) jako výstup, který nazýváme *hash* (nebo *hash hodnota*). Pro bezpečnost hashovací funkce jsou důležité odolnost vůči *kolizím* (dva různé vstupní řetězce by neměly mít stejný hash) a odolnost vůči získání *předlohy* (hashovací funkce by měla být jednosměrná). Celá tato podkapitola se věnuje kryptografickým hashovacím funkcím a netýká se jiných hashovacích funkcí, jakými mohou být například hashovací tabulky nebo cyklický redundantní součet (CRC), neboť tyto žádné metody pro zabezpečení neobsahují [1].

Zabezpečení hashovací funkce, na rozdíl od zabezpečení šifry, nepožaduje utajení dat, ale zajištění integrity, neporušenosti dat. Pro bezpečnou hashovací funkci platí, že dva různé vstupy dat mají vždy různý hash. Díky této vlastnosti může hash sloužit jako identifikátor souboru. Nejrozšířenější aplikací hashovacích funkcí jsou digitální podpisy. V případě digitálních podpisů se nepodepisuje celý dokument, ale pouze jeho hash, který slouží jako identifikátor dokumentu a zajišťuje jeho neporušenost. Pokud by se změnil pouze jeden bit dokumentu, pak výsledný hash bude jiný a při ověření podpisu budou dokument i podpis nevalidní. Podepisováním hashe dokumentu získáme výstup mnohem rychleji, než podepisováním celého dokumentu, a zároveň zachováme stejnou úrovně bezpečnosti.

Pro bezpečnost hashovací funkce je rozhodující nepravděpodobnost výstupu. Uveďme příklad použití standardní hashovací funkce SHA-256 a ASCII kódováním písmen *a* (97), *b* (98) a *c* (99) v binárním formátu:

[6] Distribuovaný systém správy verzí vytvořený Linusem Torvaldsem pro vývoj jádra Linuxu.
[7] Způsob pro dokázání hodnoty platby pomocí vynaložené práce.

SHA-256(”01100001”) =
$6da21f8975f53c826ca0cd92588799a8dbda30d89a445605f0b6cc62d070b2df$

SHA-256(”01100010”) =
$2f7e6d83a2cbbc3bf4dabd8ff8104eaacb9eb87993ebee17813ba8355b3f28e1$

SHA-256(”01100011”) =
$2ac052b445f5a9f7ec4d03aeeb381301bca8b641b66936fe2908594bb5447f52$

Jak lze na první pohled vidět, pro velmi podobné vstupní řetězce jsou hodnoty hashe extrémně rozdílné. Hashovací funkce je jako černá skříňka, která nám vrací nepředvídatelný výsledek. Pokud bychom chtěli určit hodnotu hashe obdobným způsobem pro písmeno *d*, pak ji nelze nijak odvodit ze tří předchozích hashů. Opravdu bezpečná hashovací funkce by neměla mít žádné schéma (*pattern*), ale měla by se chovat jako čistě náhodná.

Hashovací funkce jsou jednosměrné funkce a tak k nim neexistují inverzní funkce, které by nám daly původní zprávu z hashe. Vzor hashe H je jakákoliv zpráva M, která splňuje $\text{Hash}(M) = H$. Odolností proti zjištění předlohy (vzoru) rozumíme vlastnost, že z libovolného hashe nelze zjistit jeho vzor. Dokonce i pokud bychom měli neomezený výpočetní výkon, pak nelze najít právě tu zprávu, která vedla k danému hashi, protože zpráv, které vedou na stejný hash existuje mnoho. V praxi požadujeme, aby nebylo možné zjistit žádnou z takových zpráv (vzorů).

Druhý požadavek na bezpečné hashovací funkce je odolnost proti kolizím. Z Dirichletova principu však víme, že pokud existuje daleko více zpráv než hodnot hashe, pak jsou kolize nevyhnutelné. Pro připomenutí Dirichletův princip říká, že pokud umístíme m předmětů do n přihrádek, kde $m, n \in \mathbb{N}$ a zároveň $m > n$, pak existuje alespoň jedna přihrádka se dvěmi předměty, v našem případě jsou předměty zprávy a přihrádky jsou hodnoty hashe.

Víme, že kolize jsou nevyhnutelné a odolností proti kolizím rozumíme, že je velmi obtížné kolizi zjistit a útočník by neměl být schopen najít dvě různé zprávy se stejnými hodnotami hashe. Pokud je hashovací funkce odolná proti kolizím, pak je také odolná proti nalezení různých vzorů k jedné zprávě (odolnost proti zjištění předlohy).

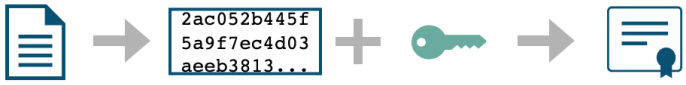
Jedna z nejpoužívanějších hashovacích funkcí je SHA-256, která spadá do skupiny SHA hashovacích funkcí. SHA, neboli *Secure Hash Algorithm*, jsou algoritmy definované americkým *National Institute of Standards and Technology* (NIST), avšak jsou používané celosvětově, kromě některých zemí používajících vlastní hashovací algoritmy (například Čína – SM3, Rusko – Streebog). SHA-0

byl první algoritmus zveřejněný v roce 1993, který byl nahrazen novým SHA-1 o dva roky později, kvůli nezveřejněnému bezpečnostnímu problému. Přestože je SHA-1 mnohem bezpečnější než jeho předchůdce, tak dnes například prohlížeč Google Chrome označuje webové stránky, které jej používají v rámci HTTPS, za nebezpečné. V dnešní době jsou algoritmy SHA-2, BLAKE2 a SHA-3 označeny za bezpečné organizací NIST [1].

SHA-2 je rodina čtyř hashovacích funkcí: SHA-224, SHA-256, SHA-384 a SHA-512, kde čísla označují délku hashe v bitech. Nejpoužívanější v této skupině jsou SHA-256 a SHA-512. Na rozdíl od SHA-1, která má 160 bitové hodnoty, má SHA-256 hashe délky 256 bitů. Z bezpečnostního pohledu jsou hashovací funkce považovány za dostatečně spolehlivé, ale pro budoucí použití připravil NIST soutěž na nový SHA-3, který vyhrál tým čtyř kryptografů z Belgicko-Italské firmy a jejich algoritmus *Keccak*.

3.4 Digitální podpis

Digitální podpisy jsou kryptografický nástroj pro podepisování a ověřování dat za účelem doložení pravosti digitálních zpráv nebo dokumentů. Poskytují tři funkce: ověření pravosti dat, ověření integrity, autor podpisu jej nemůže popřít či odejmout po podepsání. V dnešní době se používají zejména při autorizaci bankovních transakcí, výměně podepsaných elektronických dokumentů, podepisování transakcí veřejných blockchainů a pro podepisování digitálních smluv. Pro důvěryhodnost klíčů existuje *Public Key Infrastructure* (PKI), což je infrastruktura správy a distribuce veřejných klíčů, která zaručuje pravost klíčů.

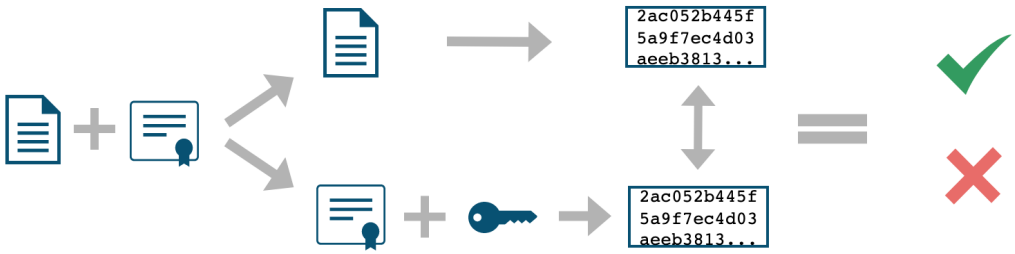


Obrázek 7: Podepsání dat soukromým klíčem.

V podkapitole 3.2 o šifrování asymetrickou kryptografií byl popsán systém veřejného a soukromého klíče. Digitální podpisy také využívají dvojici veřejného a soukromého klíče, avšak jiným způsobem. Data jsou podepsána pomocí soukromého klíče a pomocí veřejného klíče je podpis ověřen. Jak lze vidět na obrázku 7, při podepisování se data zahashují, a pomocí soukromého klíče použitého na hash se vygeneruje podpis, který se k datům připojí. Data zůstávají nepozměněná a v čitelné podobě. Po podepsání není možné data ani podpis změnit a tak je zaručena autenticita a integrita.

Obrázek 8 ukazuje postup ověření podepsaných dat. Při ověřování podpisu vezmeme data a zahashujeme je, dále z podpisu a veřejného klíče získáme další hash. Tyto dva hashe porovnáme, pokud jsou shodné, pak jsou data i podpis validní, v opačném případě došlo ke změně podpisu či dat a data nejsou validní.

Nejpoužívanější algoritmy pro digitální podpisy jsou RSA, DSA (*Digital Signature Algorithm*), ECDSA (*Elliptic Curve Digital Signature Algorithm*). V prak-



Obrázek 8: Ověření podepsaných dat pomocí veřejného klíče.

tické části je pro digitální podpisy použit algoritmus RSA s SHA-256 jako hashovacím algoritmem, proto zde detailněji popíšeme podepisování pomocí RSA.

3.4.1 Digitální podpis pomocí RSA

Šifrování pomocí algoritmu RSA bylo podrobněji popsáno v podkapitole 3.2. Podepisování pracuje na stejných principech jako šifrování, avšak s podstatnými rozdíly zejména v prohození klíčů a také v použití algoritmu. Cílem šifrování je zabezpečit informace tak, aby nebyly čitelné. Naopak při digitálním podepisování jsou data zachována v čitelné podobě a cílem je zabránit falšování. Další rozdíl mezi šifrováním a podepisováním je použití algoritmu na celá data, přesněji na jejich hash, oproti šifrování pouze *session* klíče, jak bylo popsané v podkapitole 3.2.

Digitální podpis zaručuje, že entita vlastnící soukromý klíč vázaný k určitému digitálnímu podpisu dokument podepsala a tento podpis je autentický, pravý. Jenom vlastník soukromého klíče d může vypočítat podpis $y = x^d \mod n$ pro nějakou hodnotu x, naproti tomu všichni mohou ověřit $y^e \mod n = x$ za pomocí veřejného klíče e. Na první pohled lze digitální podpis vidět jako šifrování se soukromým klíčem, avšak rozdíl oproti šifrování je podstatný.

Algoritmus RSA umožnuje klíče délek 1024, 2048, 4096, ..., 16384 bitů (podporuje i delší klíče, ale z praktického hlediska jsou operace s delšími klíči neefektivní). Při podepisování nejprve vypočítáme hash zprávy, $h = \text{hash}(m)$, dále hash šifrujeme pomocí veřejného RSA klíče:

$$s = h^d \mod n,$$

kde s je vzniklý podpis. Zprávu m posíláme společně s podpisem s. Pro ověření podpisu nejprve vypočítáme hash zprávy:

$$h_1 = \text{hash}(m),$$

dále dešifrujeme podpis s pomocí veřejného klíče e:

$$h_2 = s^e \mod n,$$

a nakonec porovnáme hodnoty h_1 a h_2. Pokud je podpis validní, pak platí následující rovnice:

$$h_2 = s^e \mod n = (h^d)^e \mod n = h_1,$$

kde h_2 je hash získaný dešifrováním digitálního podpisu s, a h_1 je hash získaný použitím hashovací funkce na text zprávy. Pokud rovnice platí, pak je podpis validní a data jsou nezměněná.

Existují i jiné algoritmy pro digitální podpisy podle standardů W3C, ale podepisování pomocí RSA stále zůstává jedním z nejpoužívanějších algoritmů. Ostatní algoritmy se liší ve způsobu zpracování hashe, ale princip podepisování a ověřování zůstává stejný.

3.4.2 Public Key Infrastructure

Veřejné klíče nemusí být nijak zabezpečené a je možné je sdílet otevřeně, jenže kdo nám zajistí, že konkrétní veřejný klíč je právě dotyčné entity, jako v následujícím příkladu. Bob[8] si musí být jistý, že veřejný klíč patří Alici.[8] Pokud by Marvin[9] přesvědčil Boba, že jeho klíč patří Alici, pak se může před Bobem vydávat za Alici. Tento problém řeší *Public Key Infrastructure* (PKI).

PKI je soubor zásad, procedur a technologie potřebné pro správu digitálních certifikátů v systému veřejných klíčů. Digitální certifikát je elektronická struktura, která spojuje entitu (například instituce, osoba, program, webová adresa, atd.) s veřejným klíčem. Nejčastější použití digitálních certifikátů je pro bezpečnou komunikaci, asymetrickou kryptografii a digitální podpisy. Cílem PKI je zaručit věruhodnost digitálního certifikátu [14].

V asymetrické kryptografii je soukromý klíč dostupný pouze majiteli a veřejný klíč je k dispozici pro veřejnost. Toto schéma lze použít dvěma způsoby: zabezpečená komunikace a digitální podpisy. Zabezpečená komunikace počítá s tím, že veřejný klíč je použit pro šifrování dat tak, aby jej mohl dešifrovat jen majitel soukromého klíče. Při digitálních podpisech jej majitel soukromého klíče jej použije pro šifrování dat, pro prokázání totožnosti, při dešifrování příslušným veřejným klíčem.

Digitální cerfitikáty zaručují spolehlivost veřejných klíčů, jedná se o digitální podpis jedné nebo více důvěryhodných stran, které dokládají validitu a autenticitu veřejného klíče. Certifikát slouží pro identifikaci a dokázání pravosti entity, podobně jako občanské průkazy identifikují občany. V praxi jsou používané dva modely: *Síť důvěry* a *Certifikační autority*.

Síť důvěry označuje model, který je použitelný, pokud jsou certifikované entity osoby. Jednotlivé osoby podepíší certifikáty dalších osob, které znají, nebo je ověřili podle oficiálních dokumentů při osobním setkání. Vznikne graf důvěry, podle kterého si účastníci mohou určit, že budou například důvěřovat certifikátu, kterému důvěřují další dvě osoby, kterým účastníci důvěřují. Tento model je používaný pro PGP šifrování při zabezpečené emailové komunikaci. Výhody sítě důvěry jsou v jednoduchosti a odolnosti proti samostatnému útočníkovi. Nevýhody jsou závislost na dodržování správných principů a absence centrálního řízení [14].

[8] Alice a Bob jsou fiktivní jména tradičně používaná pro příklady v kryptografii.
[9] Fiktivní jméno používané pro aktivního nepřátelského útočníka.

Certifikační autorita (CA) je důvěryhodná třetí strana, která se specializuje na vydávání a spravování digitálních certifikátů. CA může vydat certifikát přímo entitě, nebo také autorizovat další stranu pro vydávání certifikátů. Výhoda centrální správy certifikátů je ve snížení počtu stran potřebných pro verifikaci certifikátů a zaručení použití správných postupů. Nevýhoda je však v udržování dat jen na jednom místě, které může být snáze napadeno [14].

3.5 Doporučení národních organizací pro délky klíčů

WELMEC Guide 7.2 uvádí jak zabezpečit data, avšak pro bezpečné algoritmy a doporučené délky klíčů se odkazuje na národní organizace, například na americkou NIST, francouzskou ANSSI a německou BSI.

V následujících kapitolách uvádíme doporučené algoritmy a délky klíčů podle dokumentu organizace NIST z roku 2020 [2] a porovnáme je s doporučeními přijatými ve vybraných zemích. Bezpečnost dat zabezpečených kryptografickými algoritmy i efektivita kryptografických algoritmů přímo závisí na síle klíčů. Soukromé klíče musí být chráněny proti zveřejnění a všechny klíče proti změnám.

3.5.1 Doporučení NIST

Kryptografický systém může být skvěle zabezpečen, avšak pokud je nedostatečné zabezpečení a správa klíčů, pak je celý systém zranitelný. Pro většinu klíčů je doporučená doba používání méně než dva roky. Po uplynutí období dvou let by měly být klíče zničeny a vygenerovány nové. Některá základní pravidla při práci s klíči:

- omezit období, kdy jsou symetrické a asymetrické klíče v čitelné (*plaintext*) podobě,
- znemožnit čtení klíčů v podobě otevřeného textu,
- používat kontejnery pro uložení klíčů,
- používat *timestamp* (datum a čas) spolu s podepisováním dat,
- zrušit klíče, jakmile nejsou potřebné,
- zavést kontroly integrity pro detekci změny klíče.

Některé doporučené kryptografické algoritmy jsou specifikovány pro různé délky klíčů, které nabízí různé síly bezpečnosti. Pro porovnávání bezpečnosti uvažujeme číslo síly zabezpečení představující kolik práce je potřeba pro prolomení kryptografického algoritmu. Některé běžně používané algoritmy mají síly zabezpečení 80, 112, 128, 192, 256 bitů. Algoritmy se silou zabezpečení 80 bitů již nepovažujeme v dnešní době za bezpečné. Je pravděpodobné, že nové nebo vylepšené útoky a technologie, které budou v budoucnu vyvinuty, vyřadí některé z algoritmů, které dnes používáme.

Tabulka 3: Doporučení NIST pro délky klíčů v bitech [2]

Síla zabezpečení	Symetrický algoritmus	FFC	IFC
112	3TDEA	$L = 2048$ $N = 224$	$k = 2048$
128	AES-128	$L = 3072$ $N = 256$	$k = 3072$
192	AES-192	$L = 7680$ $N = 384$	$k = 7680$
256	AES-256	$L = 15360$ $N = 512$	$k = 15360$

Až budou běžně dostupné velké kvantové počítače, pak bude potřeba přizpůsobit většinu kryptografických algoritmů, zejména digitální podpisy, výměny klíčů pomocí Diffie-Hellman i RSA algoritmů. NIST nyní pracuje na tak zvané *post-quantum* kryptografii.

Bezpečnost kryptografických algoritmů je mimo jiné ovlivněna implementací, některé implementace prozrazují malé množství informace o klíči. Při zvolení delšího klíče je pravděpodobnost vyzrazení relevantní části klíče menší. Odhady bezpečnosti jsou počítány za předpokladu korektního generování a použití klíčů. Pokud nejsou zásady pro práci s klíči dodrženy, pak je skutečná bezpečnost nižší. Když má algoritmus X bitů bezpečnosti a uvažujeme útok hrubou silou, pak by průměrný útok trval $2^{X-1}T$ jednotek času, kde T je doba potřebná pro zašifrování jedné *plaintextové* (otevřený text) hodnoty a porovnání s odpovídající šifrovanou hodnotou.

V tabulce 3 jsou uvedeny srovnatelné odhady maximální bezpečnosti pro symetrické blokové šifry AES a asymetrické šifry a délky klíčů. V prvním sloupci je uvedena maximální bezpečnost v bitech pro algoritmy a délky klíče v daném řádku. Délka klíče není vždy totožná se sílou bezpečnosti kvůli útokům, které využívají znalosti algoritmů a dokáží jej prolomit rychleji než útokem hrubou silou. Ve druhém sloupci jsou uvedeny symetrické algoritmy, které nabízejí bezpečnost uvedenou v prvním sloupci. Třetí sloupec označuje minimální délku parametrů spojeného s algoritmy používajícími kryptografii na konečných tělesech *(Finite field cryptography)* jako jsou DSA a Diffie-Hellman, kde L je velikost veřejného klíče a N je velikost soukromého klíče. Ve čtvrtém sloupci je hodnota k (velikost klíče) pro algoritmy založené na faktorizaci *(Integer factorization cryptography)* jako RSA.

Hashovací algoritmy musí splňovat dvě podmínky zmíněné v kapitole 3.3: odolnost vůči nalezení vzoru a odolnost vůči kolizím. V tabulce 4 lze vidět sílu zabezpečení v bitech jednotlivých variant hashovacích funkcí pro aplikace vyžadující odolnost vůči kolizím. Dnes se síla bezpečnosti menší než 112 bitů nedoporučuje. SHA-224 a výše jsou tedy bezpečné pro dnešní použití [2].

Tabulka 4: Doporučení NIST pro hashovací funkce [2]

Síla zabezpečení	Aplikace vyžadující odolnost vůči kolizím
≤ 80	SHA-1
112	SHA-224, SHA3-224
128	SHA-256, SHA3-256
192	SHA-384, SHA3-384
≥ 256	SHA-512, SHA3-512

Tabulka 5: Srovnání minimálních národních doporučení pro délky klíčů v bitech

Organizace	Rok	Sym. alg.	Modulus	Souk. k.	Veřejný k.	Hash
NIST	2019-2030	112	2048	224	2048	224
ANSSI	2021-2030	128	2048	200	2048	256
BSI	2020-2022	128	2000	250	2000	256

3.5.2 Porovnání národních doporučení

Pro srovnání různých národních minimálních bezpečnostních požadavků byla vybrána doporučení americké NIST z roku 2020 podle dokumentu [2], francouzské ANSII podle dokumentu [17] a německé BSI podle dokumentu [4].

Srovnání lze vidět v tabulce 5, kde NIST uvádí doporučení pro minimální délky klíčů do roku 2030. V druhém řádku lze vyčíst doporučení podle organizace ANSSI vydaných v roce 2014, kde uvádějí minimální délku klíče pro symetrické algoritmy 128 bitů, pro asymetrické algoritmy uvádějí minimální doporučenou délku veřejného klíče a modulu 2048 bitů, minimální délku soukromého klíče 200 bitů a pro hashovací funkce uvádějí doporučení použití minimálně SHA-256. Podobná doporučení nalezneme od BSI z roku 2020, která jsou platná pouze do konce roku 2022. Od roku 2023 požaduje BSI místo 2000 bitů minimální délku 3000 bitů pro veřejné klíče.

4 Návrh knihovny

Pro praktickou část práce byla zvolena implementace knihovny sloužící pro zabezpečení naměřených hodnot při úsekovém měření rychlosti. Knihovna je navržena pro nasazení na existující projekt firmy CROSS Zlín, a.s., se kterou byla ve spolupráci diplomová práce napsána. Návrh i metody zabezpečení jsou v souladu s WELMEC Guide 7.2, a tak splňují požadavky dané evropskou legislativou. Jak bylo zmíněno v kapitole 2, WELMEC Guide 7.2 má pouze charakter doporučení, ve vzniklém software je však nutné splnit jeho požadavky podle dokumentu 0111-OOP-C005-09 [13], vydaného Českým metrologickým institutem. Pro práci je vybráno úsekové měření rychlosti, ale vzniklý software lze použít například i pro vážicí systémy.

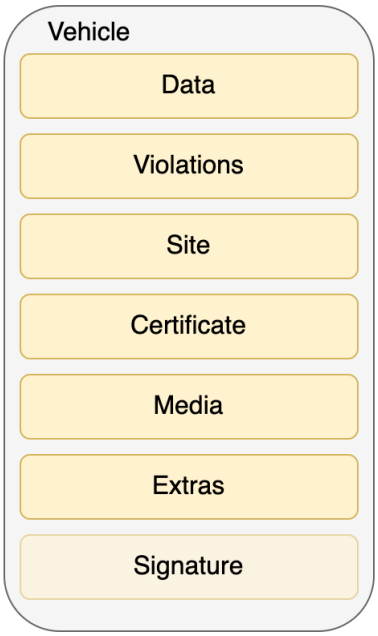
Při naměření přestupkové hodnoty na měřicím zařízení, je potřeba naměřené údaje spolu se snímky zabezpečit, aby mohly sloužit jako podklad pro přestupkové řízení. Nejprve se na zařízení ukončí proces měření a data jsou dále poslána pro vyhodnocení. Tato data jsou legálně relevatní, a proto je nutné je zabezpečit proti úmyslnému i neúmyslnému poškození. Příkladem neúmyslného poškození může být chyba při přenose, která by vedla k neúplným, nebo nesprávným údajům. Na straně příjemce, který data vyhodnocuje pro legální účely se musí příchozí balíček dat verifikovat, a v případě, že je detekována chyba, se nesmí data zpracovat pro vyhodnocení přestupků. Naměřené údaje o vozidle, spolu se snímky vozidla, musí být od okamžiku naměření zabezpečena, pro jakékoliv další zpracování, aby byla zajištěna autenticita i integrita.

Knihovna, která zprostředkovává zabezpečení je zde rozdělena podle dvou funkcionalit. První funkcionalita slouží pro zabezpečení naměřených hodnot na straně měřicího zařízení a druhá slouží pro verifikaci dat na straně příjemce dat. Pro datovou strukturu je zvolen XML soubor, obsahující údaje o naměřeném vozidle, o přestupcích, o stanici na které měření proběhlo, certifikát stanice, snímky vozidla a dodatečné informace. XML soubor vzniklý na měřidle je obohacen o digitální podpis, který umožňuje ověření totožnosti odesílatele souboru. Součástí digitálního podpisu je hash, který slouží pro potvrzení neporušení záznamu. Veřejný klíč pro verifikaci digitálního podpisu je schválen certifikační autoritou, která dokládá jeho důvěryhodnost. Dále je podepsaný XML soubor šifrován pro přenos přes otevřená komunikační rozhraní. Na straně příjemce se šifrovaný soubor dešifruje, a ověří se digitální podpis. Pokud je verifikace úspěšná, pak lze s daty dále pracovat, naopak, pokud verfikace selže, nelze data ani zobrazit na aplikaci příjemce.

4.1 XML Vozidlo

Struktura XML souboru je pevně daná a obsahuje jeden hlavní element *vehicle*, který obsahuje další elementy *data*, *violations*, *site*, *certificate*, *media*, *extras* a *signature*. Elementy pod hlavním *vehicle* obsahují bloky informací, které se vážou k vozidlu, přestupkům, ke stanici na které měření proběhlo, certifikátu stanice, mediálním souborům připojených k měření a dalším volitelným položkám. Tato

struktura je zaznačena na obrázku 9. Pokud je dokument navíc podepsaný, tak obsahuje element *signature*, ve kterém je digitální podpis.



Obrázek 9: XML struktura vozidla

První blok informací rozepsáný v tabulce 6 obsahuje veškeré údaje o vozidle. Vozidlo musí obsahovat jednoznačný identifikátor průjezdu, datum a čas měření, rychlost vozidla, jednotku měření, informace o jízdním pruhu, flagy a registrační značky. Flagy jsou hodnoty, které obecně nesou označení o stavu, v našem případě jednotlivé flagy mohou nabývat stavů *info*, *warning* a *error* a nesou informaci o varování nebo chybě při měření. Klasifikace je v podobě kolekce, protože existují různé systémy pro klasifikování vozidla, a tak jich může být uvedeno více. Interně je používám obsáhlý systém klasifikace, který se mapuje na jiné jednodušší podle potřeb zákazníka. Jedním z používaných klasifikačním systému je ŘSD8 definovaný v dokumentu [15]. Registrační značka nemusí být jen jedna, protože se ukládají přední i zadní podle dostupných snímků.

V tabulce 7 jsou vypsány údaje, které obsahuje element přestupků. Každý přestupek obsahuje naměřenou hodnotu, jednotku měření, zákonem povolenou hodnotu místa (pro osobní auto i nákladní automobil), zákonem povolenou hodnotu rychloměru (pro osobní auto i nákladní automobil). Pro rychloměry uvažujeme povolenou toleranci 3 km/h, pro váhy 5 %. Limit místa měření zavádíme pro možnost omezení rychlosti nebo hmotnosti v daném místě měření, limit uvádíme v kilometrech za hodinu pro rychloměry a v kilogramech pro váhy.

Informace o stanici jsou vidět v tabulce 8. Pro stanici uvádíme umístění, město i zeměpisné souřadnice, dále infomace o silnici, její číslo, a také sériové číslo zařízení spolu s verzí softwaru. V případě měření rychlosti na dlouhém

Tabulka 6: XML vozidlo: blok data

Id	Jednoznačný identifikátor průjezdu vozidla	
Timestamp	Datum a čas měření	
Direction	Údaj o směru jízdy	
Speed	Rychlost vozidla	
Measuring unit	Jednotka měření	
Lane	Name	Název jízdního pruhu
	Description	Popis jízdního pruhu
Classification	Kolekce klasifikace	
Flags	Kolekce flagů	
Plates	Number	Číslo registrační značky
	Confidence	Procento úspěšnosti přečtení RZ
	Source	Zdroj RZ (Front / Rear)
	Country	Země vydání registrační značky
	State	Stát vydání registrační značky
	Color	Barva textu a pozadí

Tabulka 7: XML vozidlo: blok informací o přestupku

Measured	Naměřená hodnota
Measuring unit	Jednotka měření
Site limit passenger	Limitní hodnota pro dané místo pro osobní automobily
Site limit truck	Limitní hodnota pro dané místo pro nákladní automobily
Device limit passenger	Limitní hodnota pro rychloměr pro osobní automobily
Device limit truck	Limitní hodnota pro rychloměr pro nákladní automobily
Allowance	Povolený rozdíl mezi naměřenou h. a limitní h.

Tabulka 8: XML vozidlo: blok informací o stanici

Location	Údaje o umístění stanice	
	City	Město kde se stanice nachází
	Description	Popis místa
	Longtitude	Zeměpisná šířka
	Latitude	Zeměpisná délka
Road	Údaje o silnici	
	Number	Číslo silnice
	Owner	Majitel silnice
Device	Údaje o rychloměru	
	Serial number	Sériové číslo
	Version	Verze softwaru

Tabulka 9: XML vozidlo: blok informací o přiložených datech

Medium	Typ (zpráva o přestupku, obrázek), číslo, typ obsahu (text/html, image/jpeg)

úseku je nutná synchronizace času pro validní výpočet.

Blok *media*, v tabulce 9, musí obsahovat informace o mediálních datech měření, typ dat (například zpráva o přestupku nebo obrázek). Při úsekovém měření na dlouhou vzdálenost musí být nutně přítomen snímek (s okem čitelnou RZ) ze začátku i z konce úseku. Naproti tomu při vážení vozidla stačí přední snímek (s okem čitelnou RZ). Další volitelné údaje mohou být surové signály ze senzorů.

4.2 Struktura projektu

Na začátku kapitoly 4 je stručně popsána požadovaná funkcionalita softwaru. Tato funkcionalita je implementovaná do knihovny, která vznikla jako součást této diplomové práce a která obsahuje metody pro vygenerování RSA klíčů, načtení XML souboru, načtení RSA klíče, podepsání XML souboru podle W3C xmldsig standardu [3], zašifrování XML souboru podle W3C xmlenc standardu [8], dále metody pro verifikaci podpisu XML souboru a dešifrování XML souboru.

4.2.1 Správa klíčů

Tato knihovna bude nasazena na stranu stanice, na které probíhá měření, pro šifrování a podpisování, a také na stanici příjemce dat, kde bude probíhat verifikace. Z tohoto předpokladu vyplývá nutnost uvážit určitou správu klíčů. V případě digitálního podpisu by měl být soukromý klíč uložen například v externím paměťovém zařízení, které bude zaplombováno, a SW na stanici by neměl obsahovat žádné nástroje pro zobrazení ani editaci klíče. Veřejný klíč pro ověření podpisu musí být pak uložen v příslušném akreditovaném centru, nebo uložen přímo v zařízení [18].

RSA klíče pro šifrování a dešifrování musí být také systematicky spravovány. Jedno z možných řešení je mít jeden pár (soukromý a veřejný) univerzálních klíčů, což znamená šifrovat veřejným klíčem, který nemusí být nijak zabezpečený, a všem, kdo potřebují s daty pracovat distribuovat soukromý klíč pro dešifrování. Výhoda tohoto řešení je v omezeném množství klíčů. V případě, kdy chceme ale rozdělit data nebo stanice pro různé příjemce, například pro různé skupiny stanic, nebo potřebujeme pro každého zákazníka jiný klíč, se musíme zabývat vytvořením systému pro uchování i distribuci klíčů. Výhoda tohoto řešení je, že pokud máme různé skupiny příjemců, pak by každý měl mít svůj klíč, aby bylo zajištěno, že data zobrazí jen dotyčný, komu jsou určena.

4.2.2 Zabalení dat

Na straně stanice se po dokončení procesu měření vytvoří XML dokument a podle obrázku 10 se dokument podepíše a zašifruje. Procedura, která vytváří digitální podpis, používá hashovací algoritmus SHA-2 s 256 bitovým klíčem a vzniklý hash šifruje pomocí RSA s 2048 bitovým klíčem. Podpis je zhotoven nad celým dokumentem a je s XML dokumentem spojen transformací *enveloped*, neboli zabalený. To znamená, že je hodnota podpisu vložena do XML dokumentu, pod hlavní element, v našem případě *vehicle*, do elementu *signature* spolu s informacemi o hashovací funkci i transformaci.

Obrázek 10: Proces zabalení dat

Poté se podepsaný dokument šifruje. Šifrování probíhá na úrovni jednotlivých bloků dat symetricky pomocí algoritmu AES s náhodně vygenerovaným 256 bitovým klíčem. Tento AES klíč se šifruje pomocí RSA veřejného klíče a zašifrovaný se přiloží k datům. Zašifrovaný dokument obsahuje informace o použité šifrovací metodě, a šifrované bloky zahrnují zašifrovaný AES klíč daného bloku spolu se zašifrovaným blokem dat.

Takto zabezpečený balíček je možné poslat dál i otevřenou sítí pro další zpracování. Kromě hlavní výhody podepsaných a šifrovaných dat, kterou je nutné zabezpečení pro legálně relevantní účely, je další výhoda například při dlouhodobém uložení dat. Pokud máme uložená data spolu s přestupky a potřebujeme se k nim později vrátit, zajišťuje nám digitální podpis neporušenost dat. Například když by byla data uložena v databázi, tak v případě přepsání jakékoliv hodnoty se při validaci označí celý balíček dat za neplatný.

4.2.3 Verifikace dat

Na straně příjemce dat se pomocí knihovny příchozí balíček verifikuje, dříve než se s daty vykonají jakékoliv jiné procedury. Proces je demonstrován na obrázku 11.

Obrázek 11: Proces verifikace dat

Nejprve se najde každý zašifrovaný element šifrovaného dokumentu a nahradí se původním elementem s otevřeným textem. Každý šifrovaný element s sebou nese AES klíč, kterým byl symetricky šifrován. Tento symetrický klíč je šifrován pomocí asymetrického RSA klíče. Příjemce použije svůj soukromý RSA klíč pro dešifrování jednotlivých AES klíčů, které pak slouží pro dešifrování jim odpovídajících bloků dat. Výhody tohoto přístupu jsou podrobně popsány ke konci části 3.2.

Po předchozím kroku je XML dokument v čitelné podobě, a použijeme veřejný RSA klíč pro ověření neporušení dat. V dokumentu se najde *signature* element, který obsahuje informace o transformaci (v našem případě *enveloped signature*), použité hashovací funkci i hodnotu podpisu. Z podpisu za pomoci veřejného RSA klíče získáme hodnotu hashe, kterou porovnáme s hodnotou hashe, který vypočítáme z dokumentu. Pokud jsou obě hodnoty stejné, pak nedošlo k narušení dat a XML soubor je validní.

Krok verifikace dat je nutno vykonat vždy před čtením nebo zpracováním XML balíčku dat. Pokud jsou data vyhodnocena za nevalidní, nelze je ani zobrazit, ani s nimi dále pracovat. Autenticitu i integritu balíčku zaručíme použitím procedury verifikace dat při nasazení knihovny na stranu příjemce, kde příjemcem je myšlena jakákoliv funkce pracující s daty.

4.3 Návrh aplikace

Na straně měřicího zařízení je pro použití knihovny postačující konzolová aplikace, která z XML souboru vytvoří podepsaný a šifrovaný XML balíček. Pro nasazení na existující řešení firmy bude potřebné přidat zabalení dat použitím knihovny jako mezikrok stávajícího procesu.

Na straně uživatele je možné data ověřit pomocí desktopové aplikace. Pro ověření balíčku bylo možné navrhnout konzolovou, desktopovou, či webovou aplikaci. Použití konzolové aplikace by bylo nejjednodušší pro implementaci, avšak

pro prohlížení informací o vozidle to není příliš praktické. Naproti tomu webová aplikace nabízí daleko více možností prezentace dat a bohatou javascriptovou funkcionalitu, nevýhodou je však velikost a náročnost na zdroje. V našem případě se ukázalo jako nejvhodnější řešení zvolit desktopovou aplikaci, která přináší možnosti grafického zobrazení dat, spolu s nízkými nároky na systémové zdroje. Nevýhodou desktopové aplikace je omezení uživatelů na operační systém. Podle interního průzkumu používá většina uživatelů ve státní správě, v obcích s rozšířenou působností, kteří vyhodnocují přestupky operační systém Windows, a proto je desktopová aplikace vyvinuta pro použití na Windows 10. Do budoucna by bylo vhodné řešení rozšířit i na další operační systémy.

Desktopová aplikace musí načíst XML soubor, dešifrovat data o vozidle a ověřit, zda nedošlo k úmyslnému nebo neúmyslnému poškození dat. Při úspěšném načtení zobrazí informace o vozidle, stanici a přestupcích uživateli. Při neúspěšném načtení zobrazí aplikace hlášku, upozorňující uživatele na nekorektnost dat, aniž by dále umožnila prohlížení informací o vozidle. Aplikace umožnuje načítat a ukládat omezené množství vozidel, postačující pro běžné použití. V případě nového požadavku na práci s velikým množství vozidel by bylo vhodnější řešení zvolit ukládání vozidel do databáze místo do paměti programu. Aplikace má interně uloženy klíče pro dešifrování i ověření podpisu. Veřejný klíč pro ověření podpisu může být uložen bez nutnosti dalšího zabezpečení. Soukromý klíč pro dešifrování by měl být zabezpečen proti manipulaci nebo vyzrazení a uložen v *key container*, podle doporučení Microsoft .NET dokumentace. Protože aplikace pracuje s legálně relevatní knihovnou, musí být implementované metody pro ověření jejího kontrolního součtu. Kontrolní součet by se měl ověřit při spuštění a také by uživatel měl mít možnost jej zobrazit na vyžádání.

5 Použité technologie

Knihovna byla napsána v jazyce C# na platformě .NET pomocí Microsoft Visual Studio Code. Implementované funkce knihovny *System.Security.Cryptography* byly použity podle standardů W3C. Pro přenos dat byl zvolen formát XML souborů s přesně definovanou strukturou představující data vozidel a naměřených přestupků. Řešení bylo navrženo pro integraci se stávajícím systémem WIM[10] firmy Cross Zlín, a.s. s cílem splnit požadavky softwaru pro schválení typu měřidel. Pro validaci a zobrazování příchozích XML balíčků byla implementována jednoduchá grafická aplikace.

Desktopová aplikace byla psána v jazyce C# a vyvinuta pomocí Microsoft Visual Studia 2022. Pro grafické rozhraní byla použita knihovna WinUI verze 3, která nabízí nejnovější možnosti pro vzhled a návrh desktopových aplikací pro Windows.

C# je vysokoúrovňový, objektově orientovaný, silně typovaný jazyk. Patří do skupiny jazyků C a je velmi blízký jazykům C, C++ a Java. Přestože je jazyk objektově orientovaný, tak obsahuje i nástroje pro vývoj orientovaný na komponenty [7].

C# byl vyvinut firmou Microsoft a představen spolu s prostředím Visual Studio a platformou .NET Framework. Dnes se používá pro vývoj webových aplikací, desktopových aplikací, mobilních aplikací, her i databázových programů.

Microsoft .NET je prostředí pro běh a spouštění aplikací, jehož jádrem je *Common Language Runtime*. Common Language Runtime je konkrétní implementace *Common Language Infrastructure* (CLI), která je otevřenou specifikací popisující vlastnosti proveditelného kódu a prostředí běhu. C# je jeden z možných jazyků pro programování pomocí .NETu. Ze začátku byla vyvinuta platforma .NET Framework pouze pro operační systém Windows, ale .NET, dříve .NET Core, je nyní používán pro operační systémy Windows, MacOS i Linux. Pro knihovnu i aplikaci byla použita verze .NET 6.

NuGet Package Manager je systém pro vytváření a sdílení open–source balíčků. Pro implementaci knihovny byl použit balíček *System.Security.Cryptography*, který obsahuje kryptografické funkcionality jako například šifrování, dešifrování, hashování a generování náhodných čísel. Pro použití knihovny WinUI pro desktopovou aplikaci byl instalován balíček *Project Reunion*.

Balíček *System.Security.Cryptography* obsahuje podporu pro většinu běžných symetrických (DES, 3DES, RC2, Rijndael), asymetrických (RSA, DSA) a hashovacích (MD5, SHA-1, SHA-256, SHA-384, SHA-512) kryptografických algoritmů. Balíček je distribuovaný pod licencí MIT.

Balíček *Project Reunion* obsahuje vzory (*templates*) pro tvorbu WinUI aplikací. Dříve byl balíček pojmenáván *WinUI 3 Project Templates*, ale tento balíček je nyní označen za zastaralý a Project Reunion je doporučená verze. Nabízí Packaged WinUI app projekt, který obsahuje funkcionalitu pro vytvoření MSIX balíčku, který je doporučeným způsobem pro instalaci a odinstalaci aplikací distri-

[10] WIM je zkratka pro *weigh-in-motion*, což je označení pro vážící systémy.

buovaných pomocí Microsoft Store. Balíček je distribuovaný s Microsoft software licencí.

XML neboli *eXtensible Markup Language* je značkovací jazyk, obdobně jako HTML. Slouží pro uchovávání a přenos dat, nese důležité informace o datech a je lehce čitelný. Není potřeba dalších komentářů k datům, protože značky definují strukturu i parametry dat. XML není určen pro zobrazení dat a značky nejsou předem definované, na rozdíl od HTML. Schema lze pevně definovat pomocí *XML Schema Definition* (XSD), což je dokument definující strukturu, neboli elementy a jejich atributy, počet a pořadí potomků elementu, a také datové typy elementů a atributů [6].

WinUI 3.0 je nejnovější rozhraní pro návrh a implementaci Windows desktopových aplikací pomocí .NETu i C++ (dřívější WPF a Windows Forms pracovaly jen s .NETem). Oproti starší WinUI 2.x verzi obsahuje nejen nové XAML ovládací prvky, ale také celé XAML UI rozhraní. Starší WPF a WinForms aplikace mohou používat nové WinUI prvky pomocí XAML *Islands*, a tak postupně přidávat novou funkcionalitu do fungující aplikace. Microsoft Application Store nabízí aplikaci *WinUI 3 Controls Gallery*, která ukazuje použití všech moderních ovládacích prvků, kde si programátor může vyzkoušet nejnovější ovládací prvky. Project Reunion je název projektu, který zpřístupňuje moderní Windows prvky jako NuGet balíčky.

Model–View–ViewModel je návrhový vzor pro software, který usnadňuje oddělení vývoje grafického uživatelského rozhraní (*view*) od vývoje programové funkcionality (*model*) a také umožňuje nezávislost grafického *view* na konkrétní platformě *modelu*. *ViewModel* slouží jako konvertor hodnot a je zodpovědný za převedení datových objektů na objekty, které lze snadno prezentovat. Vzor byl začleněn do WPF aplikací a nyní je také používán v WinUI 3.0 aplikacích. Tato architektura je znázorněna na obrázku 12.

Obrázek 12: MVVM architektura

Extensible Application Markup Language (XAML) je značkovací jazyk založený na XML, vyvinutý Microsoftem pro návrh grafického rozhraní aplikací. XAML je používán v WPF, WinUI i UWP aplikacích. V souboru XAML jsou třídy a vlastnosti odkazovány pomocí elementů XML a atributů a jsou vytvořeny odkazy mezi značkami a kódem [11].

6 Programátorská dokumentace

Hlavním cílem při návrhu a implementaci bylo vytvořit software, který bude splňovat všechna doporučení WELMEC Guide 7.2 a bude jej možné integrovat se stávajícím řešením firmy Cross Zlín, a.s. za účelem schválení typu měřicích zařízení.

Po podrobném nastudování dokumentu WELMEC Guide 7.2 jsem vytvořila návrh softwaru, popis potřebných funkcí a požadavky nutné pro zabezpečení softwaru i dat. Hlavním požadavkem bylo zabezpečit data na měřicím zařízení před dalším zpracováním a přenosem. Pro tuto práci jsem vybrala úsekové měření rychlosti, a tak jsem zvolila požadavky relevantní pro silniční rychloměry. Pro třídu rizika i zabezpečení jsem zvolila třídu D jako kompromis mezi zamezením reálných bezpečnostních hrozeb a složitostí validace SW při procesu schválení typu stanoveného měřidla.

6.1 Požadavky podle Welmec Guide 7.2

Při návrhu i implementaci jsem vycházela z jednotlivých požadavků. Nejprve jsem určila, které z kategorií požadavků jsou potřebné pro můj úkol, a dále jsem navrhla knihovnu, která obsahuje funkce nutné pro zabezpečení dat při přenosu.

Měřicí zařízení pro úsekové měření rychlosti je zařízení *typu U*, jedná se o počítač s linuxovým operačním systémem, který nabízí další funkce mimo legálně kontrolované. Zařízení je připojeno do otevřené sítě a senzor je připojen k výpočetní jednotce pomocí komunikačního rozhraní. Uživatelské rozhraní obsahuje funkcionalitu nad rámec měření a paměť je pevná, nejčastěji HDD.

6.1.1 Požadavky pro univerzální měřicí přístroje

Z požadavků pro univerzální měřicí přístroje jsou pro konkrétní software důležité zejména body U2, U5, U6 a U7 z textu [18], podle kterých musí být pro legálně relevantní software vypočten kontrolní součet, který je možno specifickým příkazem zobrazit. Bezpečnosti algoritmů a vhodných délek klíčů byla věnována kapitola 3.5. Program musí kontrolovat, zda je kontrolní součet správný, a v případě změny se musí zablokovat. Chráněné parametry přístroje, které je nutné zabezpečit, by měly být uloženy na jednoúčelovou přídavnou jednotku, která je zajištěna plombou, nebo přímo v jednotce snímače.

6.1.2 Požadavky pro přenos naměřených dat komunikačními sítěmi

Naměřená data jsou přenášena přes otevřenou síť pro další legálně relevantní účely. Musí tedy splňovat požadavky pro přenos komunikačními sítěmi. Požadavky se vztahují na úplnost dat, úmyslné a neúmyslné změny, zabezpečení klíčů a verifikaci dat. Tyto požadavky byly pro tvorbu knihovny zásadní, protože od nich se odvíjel návrh struktury XML dokumentu a také zabezpečení dat. Jedná se o požadavky T1–T6 z dokumentu [18].

Požadavky pro přenos naměřených dat komunikačními sítěmi uvádějí nutné zabezpečení pro úplnost dat, jinými slovy, co vše by měla legálně relevantní data obsahovat. Dále uvádějí, jak zajistit data proti změnám, přičemž proti neúmyslným změnám stačí zabezpečení kontrolními součty, naproti tomu proti úmyslným změnám je potřeba digitální podpis celých dat. Je zde nutné zabezpečení a ověření klíčů. Obecně je třeba klíče certifikovat v akreditovaném Trust centru, výjimkou je pouze pokud se obě strany předem dohodnou, a klíč lze přečíst z přístroje. Úkolem softwaru, který přijímá data, je ověřit digitální podpis, zda nedošlo k chybě při přenosu, nebo k záměrné změně dat.

6.1.3 Požadavky pro dlouhodobé uložení naměřených dat

Požadavky z rozšíření pro dlouhodobé uložení naměřených dat nejsou v rozsahu této práce, avšak pro nasazení na existující řešení firmy je bude třeba splnit. Data jsou ukládána od okamžiku dokončení procesu měření a zpracována i poté, proto pro jejich dlouhodobé uložení je vhodné data ukládat externě v databázi.

Data uložena v databázi by měla být zabezpečena proti neúmyslným i úmyslným změnám, aby se s nimi mohlo dále pracovat jako s legálně relevantními údaji. V našem řešení bude vhodné data ukládat spolu s digitálním podpisem. Pokud by data nebyla zabezpečena jako legálně relevantní, pak se k nim sice můžeme vrátit, ale nemůžeme si být jisti, zda jsou data validní. Data musí obsahovat některé minimální údaje a musí být možné vyčíst z jakého měřicího přístroje byla vygenerována. V případě použití asymetrické kryptografie je nutné vyřešit zabezpečení klíčů. Při práci s uloženými daty je nutné je nejprve validovat a data se musí ukládat automaticky na úložiště s dostatečnou kapacitou. Při zaplnění kapacity úložiště zařízení, pak je operátor informován prostřednictvím výstravy a zařízení nespouští nové měření.

6.1.4 Požadavky pro oddělení softwaru

Oddělení softwaru je volitelný přístup pro návrh softwaru pro oddělení legálně relevantního od legálně nerelevantního. Tyto části spolu komunikují pomocí řízeného rozhraní. Při implementaci odděleného softwaru není potřeba kontrolovat legálně nerelevantní části při aktualizaci nebo změně. Je vhodné software rozdělit například pro usnadnění aktualizací.

Rozdělení nám umožňuje uzavřít pouze část pracující s legálně relevantními daty a ponechává ostatní části otevřené změnám a úpravám. V našem řešení je celá knihovna legálně relevantní, avšak aplikace, které ji používají, mohou obsahovat i legálně nerelevantní části. Požadavky na oddělení softwaru specifikují zejména, která data a funkcionalita jsou legálně relevantní.

6.1.5 Požadavky pro stahování legálně relevantního softwaru

Stahování legálně relevantního softwaru je mimo rozsah práce, ale bude jej potřebné v budoucnu splnit pokud zařízení umožňuje stahování bez porušení plomby.

Stahování se skládá ze dvou fází: přenos a instalace. Před instalací musí být ověřen podpis softwaru pro zajištění autenticity i integrity. Měl by být přítomen logger, který zaznamená každý pokus o download softwaru.

6.2 Implementace knihovny

Knihovna *XMLPackageLibrary* byla navržena, aby fungovala nezávisle na operačním systému, proto používá pouze takové funkce z knihovny *System.Security.Cryptography.Xml*, které jsou nezávislé na operačním systému. Knihovna obshuje následující metody:

- SignXmlDocument,
- VerifyXmlSignature,
- EncryptXmlDocument,
- DecryptXmlDocument,
- LoadXmlFromFile,
- LoadRsaKeyFromFile.

6.2.1 SignXmlDocument

Statická metoda pro podepsání XML dokumentu přijímá dva argumenty: XML dokument a RSA klíč, a následně vrací podepsaný XML dokument. Metoda implementuje standard W3C zvaný XMLDSIG [3] pro digitální podepisování XML dokumentů. Metoda podepíše celý dokument a připojí podpis ve formě XML elementu *Signature*.

Zdrojový kód 1 obsahuje kód jádra metody. Nejprve vytvoříme objekt *Reference*, v němž jsou části dokumentu, které podepisujeme. Pro podepsání celého dokumentu nastavíme vlastnost *Uri* na prázdný řetězec. Dále vytvoříme objekt *XmlDsigEnvelopedSignatureTransform*, který představuje transformaci použitou pro reprezentaci XML dat. Transformace je typu *Enveloped Signature*, což znamená, že element *Signature* je potomek hlavního root elementu a při počítání podpisu je odstraněn z výpočtu. Objekt přidáme do *Reference* jako použitou transformaci. Nyní vytvoříme objekt *SignedXml*, který představuje obal pro XML dokument. Nastavíme mu podepisující RSA klíč, přidáme dříve vytvořenou *Reference* a vypočítáme podpis. Nakonec vytvoříme *XmlElement*, který obsahuje podpis, a vložíme jej do původního XML dokumentu.

```
Reference reference = new Reference();
reference.Uri = "";
XmlDsigEnvelopedSignatureTransform env = new
   XmlDsigEnvelopedSignatureTransform();
```

```
reference.AddTransform(env);

SignedXml signedXml = new SignedXml(xmlDoc);
signedXml.SigningKey = rsaKey;
signedXml.AddReference(reference);
signedXml.ComputeSignature();
XmlElement xmlDigitalSignature = signedXml.GetXml();
XmlDocument signed = xmlDoc;
signed.DocumentElement.AppendChild(
                signed.ImportNode(xmlDigitalSignature,
                   true));
```

Zdrojový kód 1: Metoda SignXmlDocument

Metoda vyvolává výjimku v případě neúspěšného vytvoření podpisu, kterou informuje o nezdaření operace. Struktura podpisu, která je vidět ve zdrojovém kódu 2, ukazuje v úvodním elementu odkaz na standard XMLDSIG [3]. Dále je v elementu *SignatureMethod* uvedena metoda podepisování, v našem případě se jedná o metodu RSA na SHA-256. V elementu *Reference* lze vidět, že je podepsán celý dokument a součástí *Transforms* je uvedena transformace *Enveloped signature*. *DigestMethod* uvádí metodu pro výpočet hashe a *DigestValue* obsahuje vypočtený hash. Element *SignatureValue* obsahuje hodnotu podpisu.

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
        <CanonicalizationMethod Algorithm="..." />
        <SignatureMethod Algorithm="..." />
        <Reference URI="">
            <Transforms>
               <Transform Algorithm="..." />
            </Transforms>
            <DigestMethod Algorithm="..." />
            <DigestValue>...</DigestValue>
        </Reference>
    </SignedInfo>
    <SignatureValue>...</SignatureValue>
</Signature>
```

Zdrojový kód 2: Struktura XML elementu Signature

6.2.2 VerifyXmlSignature

Statická metoda pro ověřování podepsaného XML dokumentu přijímá dva argumenty: XML dokument a RSA klíč. Metoda vrací hodnotu *true*, pokud je podpis ověřen, a v opačném případě vrací *false*. Hlavní část metody je uvedena ve zdrojovém kódu 3. Nejprve je vyhledán element *Signature*, pokud není nalezen, metoda vyvolá výjimku informující o tom, že XML element *Signature* nebyl

nalezen. Pokud je element přítomen, pak proběhne ověření a hodnota výsledku je vrácena metodou.

```
SignedXml signedXml = new SignedXml(xmlDoc);
XmlNodeList nodeList =
   xmlDoc.GetElementsByTagName("Signature");

XmlElement signature = (XmlElement)nodeList[0];
signedXml.LoadXml(signature);
bool verified = signedXml.CheckSignature(rsaKey);
```

Zdrojový kód 3: Metoda VerifyXmlSignature

6.2.3 EncryptXmlDocument

Statická metoda pro šifrování celého XML dokumentu přijímá tři vstupní argumenty: XML dokument, RSA klíč a název klíče a jako výsledek vrací zašifrovaný XML dokument. Logiku metody lze vidět ve zdrojovém kódu 4. Nejprve je nalezen element, který má být šifrován, v našem případě se jedná o root element *vehicle*. Dále je vytvořen symetrický AES klíč, který je použit pro zašifrování elementu *vehicle*. Poté je vytvořen nový *EncryptedData* objekt, kterému je nastavena metoda šifrování na AES-256. Nyní je AES klíč zašifrován pomocí RSA klíče. Nový objekt *EncryptedKey* obsahuje objekt *CipherData*, který je tvořen šifrovaným AES klíčem a objektem *EncryptionMethod*, který uvádí metodu šifrování RSA. K šifrovaným datům je připojena informace o klíči a metodě šifrování. *KeyInfoName* objekt pomáhá při dešifrování identifikovat správný asymetrický klíč pro dešifrování symetrického klíče. Nakonec je k *EncryptedData* objektu připojena hodnota šifry.

```
XmlDocument encryptedXmlDoc = xmlDoc;
XmlElement elementToEncrypt =
   encryptedXmlDoc.GetElementsByTagName("vehicle")[0] as
   XmlElement;
Aes sessionKey = Aes.Create();

EncryptedXml encryptedXml = new EncryptedXml();
byte[] encryptedElement =
   encryptedXml.EncryptData(elementToEncrypt,
   sessionKey, false);
EncryptedData edElement = new EncryptedData();
edElement.EncryptionMethod = new
   EncryptionMethod(EncryptedXml.XmlEncAES256Url);

EncryptedKey ek = new EncryptedKey();
```

```
byte[] encryptedKey =
   EncryptedXml.EncryptKey(sessionKey.Key, rsaKey,
   false);
ek.CipherData = new CipherData(encryptedKey);
ek.EncryptionMethod = new
   EncryptionMethod(EncryptedXml.XmlEncRSA15Url);

edElement.KeyInfo.AddClause(new
   KeyInfoEncryptedKey(ek));
KeyInfoName kin = new KeyInfoName();
kin.Value = keyName;
ek.KeyInfo.AddClause(kin);

edElement.CipherData.CipherValue = encryptedElement;

EncryptedXml.ReplaceElement(elementToEncrypt,
   edElement, false);
```

Zdrojový kód 4: Metoda EncryptXmlDocument

Metoda vrací výsledný šifrovaný XML dokument. Struktura šifrovaného XML dokumentu je definovaná standardem XMLENC [8] a je ukázána ve zdrojovém kódu 5. Hlavní element se nazývá *EncryptedData* a obsahuje odkaz na šifrovací standard. Hlavní element obsahuje informace, že použitá metoda šifrování je AES-256. První element *KeyInfo* obsahuje zašifrovaný AES klíč (*CipherValue*) a metodu šifrování RSA (*EncryptionMethod*) použitou pro šifrování AES klíče. Druhý element *KeyInfo* obsahuje název použitého RSA klíče (*KeyName*). Poslední potomek rootu *CipherData* má jediného potomka *CipherValue*, který obsahuje šifrovaný XML dokument.

```
<EncryptedData
   xmlns="http://www.w3.org/2001/04/xmlenc#">
    <EncryptionMethod Algorithm="..." />
    <KeyInfo xmlns="...">
        <EncryptedKey xmlns="...">
            <EncryptionMethod Algorithm="..." />
            <KeyInfo xmlns="...">
                <KeyName>...</KeyName>
            </KeyInfo>
            <CipherData>
                <CipherValue>...</CipherValue>
            </CipherData>
        </EncryptedKey>
    </KeyInfo>
    <CipherData>
        <CipherValue>...</CipherValue>
```

```
    </CipherData>
</EncryptedData>
```

Zdrojový kód 5: Struktura šifrovaného XML dokumentu

6.2.4 DecryptXmlDocument

Statická metoda pro dešifrování XML dokumentu přijímá tři vstupní argumenty: šifrovaný XML dokument, RSA klíč a název klíče. Metoda vrací dešifrovaný XML dokument. V prvním kroku metoda nalezne element *EncryptedData* a nahradí jej původním elementem obsahujícím původní obsah jako otevřený text. RSA klíč je použit pro dešifrování symetrického AES klíče, který je použit pro dešifrování XML elementu. Metoda je ilustrována ve zdrojovém kódu 6. Je zde použit název RSA klíče, podobně jako v šifrovací metodě, pro nalezení správného klíče.

```
XmlDocument xmlDoc = encryptedXmlDocument;
EncryptedXml exml = new EncryptedXml(xmlDoc);
exml.AddKeyNameMapping(keyName, rsaKey);
exml.DecryptDocument();
```

Zdrojový kód 6: Metoda DecryptXmlDocument

6.2.5 LoadXmlFromFile

Statická metoda pro načtení XML dokumentu z cesty, která je zadaná jediným argumentem, vrací objekt představující XML dokument. Metoda načte soubor a v případě neúspěchu vrací výjimku, která oznamuje chybu při načítání XML dokumentu z dané adresy.

6.2.6 LoadRsaKeyFromFile

Statická metoda pro načtení RSA klíče ze souboru na cestě zadané argumentem *filename* vrací objekt *System.Security.Cryptography.RSACryptoServiceProvider*. Metoda načte string bytů zvaný *blob* ze souboru a dále jej převede na RSA klíč, který vrací jako výsledek. Při chybě načítání vyvolá metoda výjimku, která informuje o chybě při operaci načítání RSA klíče.

6.3 Implementace grafické aplikace

Pro ukázku použití knihovny jsem navrhla a implementovala grafickou aplikaci *VehicleBrowser* pro operační systém Windows 10 sloužící pro verifikaci a zobrazování dat o průjezdu vozidel. Aplikace slouží pro demonstrativní použití knihovny, funkcionalita je omezená zejména pro validaci XML balíčků. Do budoucna by aplikace mohla být rozšířena pro použití při zobrazování, filtrování i exportování průjezdů vozidel a přestupků.

Vzhled grafické aplikace jsem definovala pomocí značkovacího jazyka XAML. Celý vzhled jsem navrhla v souladu s nejnovějšími doporučeními pro Windows desktopové aplikace a pomocí nejnovější grafické knihovny WinUI 3.0 sloužící pro implementaci moderních desktopových aplikací. Aplikace slouží pro načtení XML souboru reprezentující data o průjezdu vozidla. XML soubor obsahující data o průjezdu může být podepsán, nebo podepsán a také šifrován. Po načtení XML souboru se nejprve průjezd vozidla, pokud byl šifrován, dešifruje pomocí mé *XMLPackageLibrary*. Dále se za pomoci knihovny *XMLPackageLibrary* ověří digitální podpis a pokud jsou data o průjezdu vozidla validní, pak se deserializují do příslušných tříd představujících bloky dat o vozidle. Pokud není digitální podpis validní, pak je uživatel upozorněn, že data o průjezdu nebyla validní a není s nimi dále možno pracovat. Pokud je průjezd vozidla ověřen, pak je přidán do seznamu a je možné prohlížet detaily.

Struktura kódu grafické aplikace je jednoduchá, protože aplikace implementuje MVVM architekturu a vozidla jsou převáděna na objekty *VehicleViewModel*, které mají jednodušší strukturu a obsahují pouze hodnoty, které jsou zobrazeny v okenní aplikaci. Aplikace obsahuje soubory pro jednotlivé třídy objektů, sloužící pro deserializaci XML souboru. Dále obsahuje aplikace třídu *Application* a jí odpovídající XAML soubor *App.xaml.cs*, kde je definované použití WinUI knihovny a kde je spuštěno základní okno aplikace. Pro návrh grafického rozhraní jsem použila zejména ovládací prvky *ListView* pro zobrazení seznamu vozidel a *GridView* pro zobrazení detailu vozidla. Ovládací prvky nabízí možnost provázání s vlastnostmi *ViewModel*, což zaručuje zobrazování měnících se údajů. Knihovna WinUI 3.0 nabízí rozšířené možnosti *data bindings*, takže je hlídána syntaxe na rozdíl od použití *data bindings* například v WPF aplikacích. Vzhled okna aplikace je definován v souboru *MainWindow.xaml* a třída obsahující veškerou funkcionalitu okna je definována v souboru *MainWindow.xaml.cs*.

Hlavní okno obsahuje odkaz na *MainViewModel*, což je třída reprezentující model pro zobrazení dat. *MainViewModel* obsahuje kolekci objektů *VehicleViewModel*, ze které je naplněn seznam v okně aplikace. Na pozadí aplikace jsou vozidla (přesněji průjezdy vozidel) uložena v kolekci, pro perzistentní uložení při opakovaném spuštění jsou průjezdy uloženy jako XML soubory v lokálním úložišti aplikace, avšak jak bylo zmíněno v podkapitole návrhu aplikace 4.3, pro větší množství uložených vozidel by bylo vhodnější je ukládat do databáze. V tomto případě by podle požadavků o dlouhodobém uložení uvedených v sekci 6.1.3 musela databáze obsahovat sloupec s digitálními podpisy. Dále obsahuje *MainViewModel* také vlastnost *SelectedVehicle*, která slouží pro zobrazení detailu vybraného vozidla.

Třída *VehicleViewModel* slouží pro definování modelu pro zobrazení jednotlivého průjezdu. Třída obsahuje vlastnosti pro všechna zobrazovaná data například *Id*, *Timestamp*, *Speed* atd. Třídy *MainViewModel* a *VehicleViewModel* implementují rozhraní *INotifyPropertyChanged*, které zaručuje, že pokud jsou data změněna na pozadí v kódu, pak se změna promítne do uživatelského rozhraní.

Veškerá funkcionalita okna je implementována ve třídě *MainWindow*, která

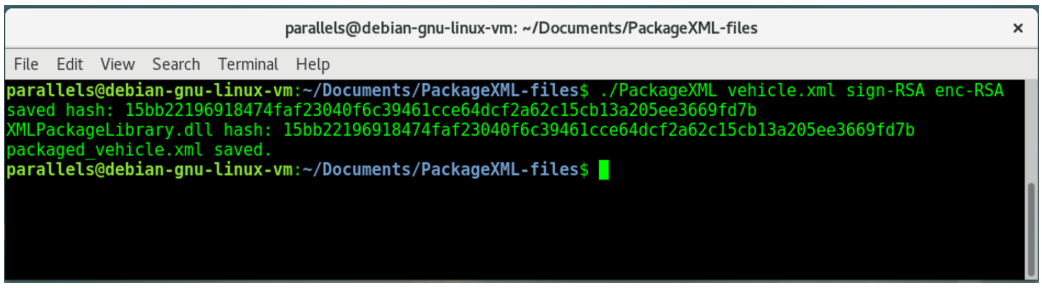
obsahuje metody pro ověření kontrolního součtu knihovny, obsluhu tlačítek, zpracování vybraného souboru a zobrazení hlášek o chybách uživateli.

7 Uživatelská dokumentace

Knihovna byla vyvinuta pro zabezpečení naměřených dat před náhodným nebo úmyslným poškozením. Zabezpečená data mohou být přenášena otevřenou sítí bez rizika vyzrazení nebo poškození informací. Knihovna je legálně relevantní, a tak musí být vypočítán její kontrolní součet, který zaručuje její pravost při použití. Knihovna bude v praxi využita na straně serveru pro šifrování a podepisování dat a na straně uživatele bude použita pro ověření úplnosti a nepoškození dat. Pro ukázku použití byla připravena data, v podobě XML souborů, představující fiktivní průjezdy vozidel přes měřicí stanici. Struktura XML dat je pevně daná a byla popsána v podkapitole 4.1, obsah elementů je zvolený tak, aby ilustroval reálné případy, ale není důležitý. Pro šifrování a podepisování byly vygenerovány dva páry RSA klíčů. Uživatel si pomocí RSA klíčů a konzolové aplikace data podepíše a zašifruje. Poté si uživatel pomocí grafické aplikace data dešifruje, ověří a zobrazí.

7.1 Konzolová aplikace pro Linux

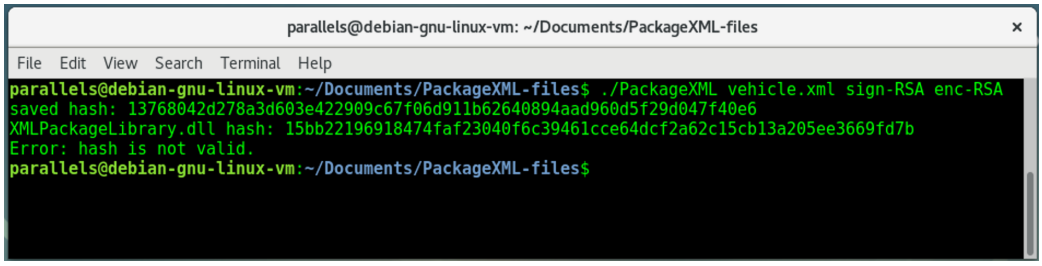
Pro použití knihovny byla implementována jednoduchá konzolová aplikace *PackageXML* vyvinuta pro linuxové operační systémy. Aplikace byla testována na Debian GNU/Linux 9 na 64 bitové architektuře.



Obrázek 13: Příklad použití konzolové aplikace

Spuštění aplikace nevyžaduje instalování žádných programů ani balíčků. Uživatel si nejprve zkopíruje celou složku s názvem *PackageXML-files* na místní uložiště. Složka obsahuje veškeré binární soubory potřebné pro spuštění a také spustitelný soubor *PackageXML*. Aplikaci lze spustit se dvěma nebo třemi argumenty. První argument udává cestu k XML souboru, druhý argument udává cestu k RSA klíči pro podepsání dokumentu a třetí volitelný argument udává cestu k RSA klíči pro šifrování XML souboru. Při použití se dvěma argumenty vytváří program podepsaný XML soubor a při použití se třemi argumenty vytváří podepsaný i šifrovaný XML soubor.

Na obrázku 13 je ukázkán příklad použití se třemi argumenty: *vehicle.xml*, *sign-RSA* a *enc-RSA*. Po spuštění aplikace se ověří kontrolní součet knihovny a pokud není validní, pak se vypíše hláška informující uživatele o chybě a program se ukončí, jak lze vidět na obrázku 14. Při úspěšném ověření se vypíše hodnota

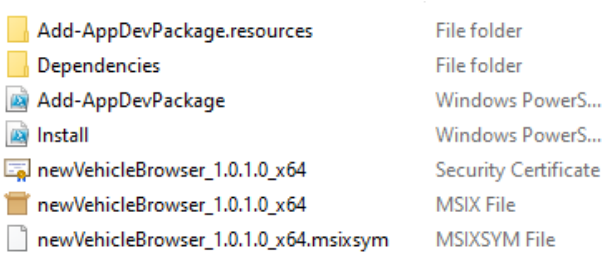


Obrázek 14: Příklad nevalidního hashe knihovny

součtu na konzoli a vstupní XML soubor je podepsán a zašifrován. Na konec je na konzoli vypsána zpráva o úspěšném vytvoření souboru s názevem původního souboru spolu s označením *packaged*, v našem případě *packaged_vehicle.xml*.

7.2 Grafická aplikace pro Windows

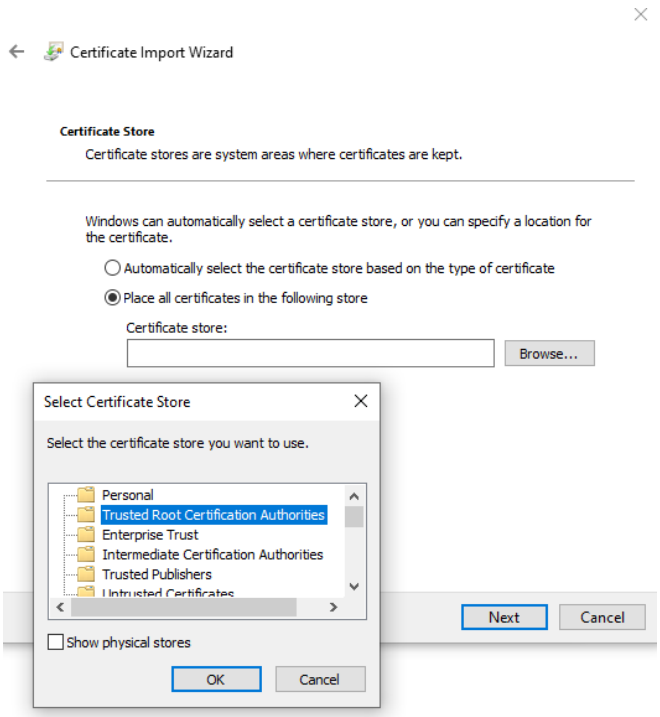
Pro ilustraci použití knihovny na straně uživatele byla implementována grafická aplikace *Vehicle Browser* pro operační systém Windows 10 s nejstarší podporovanou verzí 10.0.17763.0 a cílenou na verzi 10.0.19041.0. Aplikace byla testována na Windows 10 verzi 19044.1645.



Obrázek 15: Struktura složky pro instalaci grafické aplikace

Pro spuštění aplikace je třeba si nejprve zkopírovat složku *VehicleBrowser* na lokální uložiště. Struktura složky je vidět na obrázku 15. MSIX soubor s názvem *newVehicleBrowser_1.01.0_x64* je instalátor aplikace. Před spuštěním instalátoru je však potřebné označit certifikát, kterým je aplikace podepsána, jako důvěryhodný. Otevřením souboru typu *Security Ceritificate* zobrazíme dialogové okno pro instalaci certifikátu. Kliknutím na tlačítko *Install Certificate* se otevře průvodce pro instalaci certifikátu. Máme na výběr, jestli certifikát povolíme s platností pro současného uživatele nebo celý počítač. Zvolíme celý počítač a klikneme na tlačítko *Next* pro pokračování. Vybereme vlastní umístění certifikátu do složky *Trusted Certification Authorities*, jak je vidět na obrázku 16 a dokončíme instalaci.

Nyní můžeme spustit MSIX soubor *newVehicleBrowser_1.01.0_x64* pro otevření instalátoru aplikace *Vehicle Browser*, který lze vidět na obrázku 17. Po úspěšné instalaci je aplikace automaticky spuštěna.



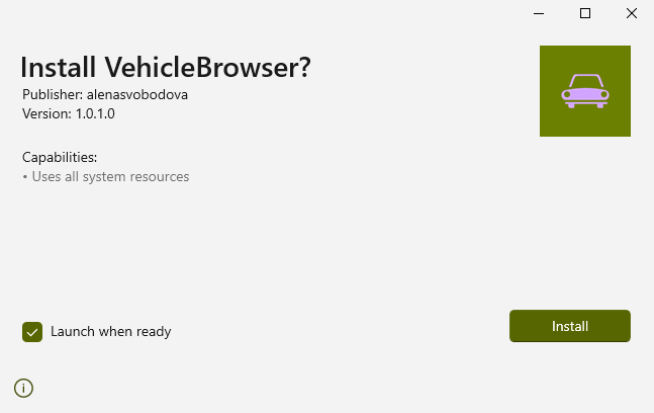
Obrázek 16: Průvodce instalací certifikátu

Grafická aplikace *Vehicle Browser* obsahuje jediné hlavní okno, ve kterém se nachází horní lišta s tlačítky pro přidání a odstranění vozidla. Dále obsahuje okno v levé části seznam vozidel, ze kterého lze vybrat vozidlo pro zobrazení detailního náhledu v pravé části okna. Vzhled aplikace je vidět na obrázku 18.

Při spuštění aplikace je uživateli ukázáno dialogové okno s informací o tom, zda byl ověřen kontrolní součet legálně relevantní knihovny. Dialogové okno obsahuje pouze jediné tlačítko pro pokračování. V případě, že kontrolní součet není ověřen, pak se aplikace po kliknutí na tlačítko dialogového okna ukončí. V opačném případě je kontrolní součet úspěšně ověřen a uživatel může pokračovat v používání aplikace.

Horní lišta aplikace obsahuje tři tlačítka, první tlačítko je pro přidání vozidla, druhé slouží k odstranění vozidla a třetí je pro zobrazení dodatečné nabídky, která obsahuje tlačítko pro ověření kontrolního součtu legálně relevantní knihovny a tlačítka pro nahrání souborů obsahující RSA klíče. Uživatel má možnost kdykoliv během práce znovu ověřit kontrolní součet, a v případě, že ověření selže, je aplikace ukončena.

Při kliknutí na tlačítko pro přidání průjezdu vozidla je otevřeno dialogové okno pro výběr souboru. Uživatel zvolí XML soubor obsahující data pro vložení. Při otevření dialogového okna se v hlavním okně zobrazí ukazatel průběhu, který je ukončen úspěšným přidáním průjezdu vozidla nebo vyskakovacím upozorněním o neúspěšném přidání. Operace přidání vozidla může selhat například v případě chyby v dešifrování, chyby při ověření podpisu, nekorektní struktury XML souboru, nebo přítomnosti jiného průjezdu se stejným identifikátorem. Při



Obrázek 17: Instalátor aplikace Vehicle Browser

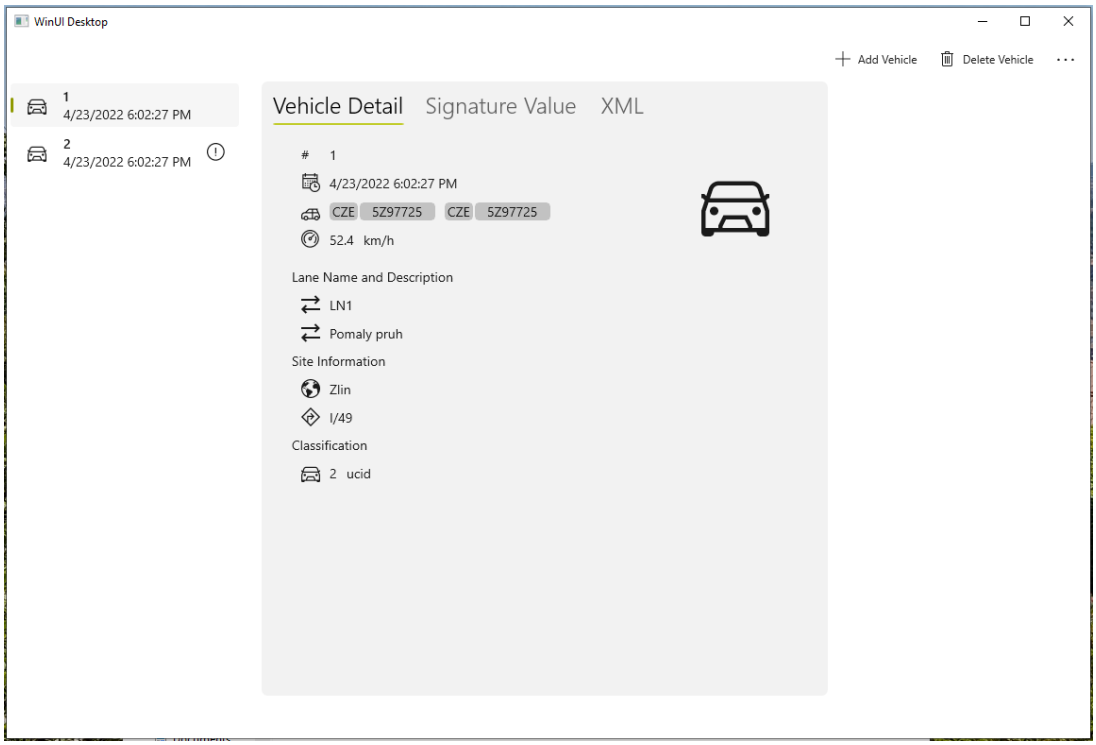
úspěšném přidání je průjezd vozidla zobrazen v seznamu.

Pro odstranění záznamu o průjezdu vozidla ze seznamu je třeba kliknout na tlačítko pro odstranění. Při úspěšném odstranění je vozidlo odebráno ze seznamu a v opačném případě je uživatel upozorněn na chybu při odstraňování pomocí vyskakovacího upozornění. Odstranit lze pouze vybrané vozidlo.

Levá část okna obsahuje seznam uložených průjezdů vozidel. Ze seznamu lze zvolit jednotlivé položky, není povolena volba více průjezdů zároveň. Každý průjezd obsahuje ikonku vozidla, jeho jednoznačný identifikátor a datum a čas pořízení záznamu. Pokud obsahuje průjezd vozidla přestupek, pak je v položce seznamu uvedena ikonka vykřičníku. Při zvolení záznamu v seznamu se jeho detail zobrazí v pravé části okna.

Pravá část okna slouží pro zobrazení detailů o zvoleném průjezdu. Informace jsou rozděleny do tří záložek: základní informace, hodnota digitálního podpisu a text celého XML souboru představující konkrétní průjezd. Záložka se základními informacemi obsahuje ikonku vozidla, jednoznačný identifikátor průjezdu, datum a čas měření, volitelně přední a zadní registrační značky, naměřenou rychlost spolu s jednotkou měření. Dále jsou uvedeny informace o jízdním pruhu, o stanici, na které měření probíhalo, a pokud došlo k přestupku, pak také informaci o typu přestupku. Druhá záložka obsahuje hodnotu digitálního podpisu záznamu. Poslední záložka obsahuje text celého dešifrovaného XML dokumentu spolu s digitálním podpisem.

Při prvním spuštění obsahuje aplikace dva ukázkové záznamy průjezdů vozidel a při vložení dalších záznamů jsou záznamy uchovány v paměti aplikace pro příští spuštění.



Obrázek 18: Okno aplikace Vehicle Browser

Závěr

Bezpečnost softwaru je jedním z nejdůležitějších aspektů aplikací pracujících s legálně relevantními informacemi. Stanovená měřidla musí splňovat požadavky a kritéria předložená národními a mezinárodními organizacemi. Hlavním referenčním bodem této práce je příručka WELMEC Guide 7.2, která poskytuje požadavky nezbytné pro splnění národních právních požadavků.

Návrh legálně relevantního softwaru určeného k zabezpečení dat z rychloměru přímo vyplynul z požadavků prezentovaných příručkou WELMEC Guide 7.2 a z konfigurace měřicího zařízení. Vzhledem k nutnosti přenášet legálně relevantní data z měřicího zařízení k dalšímu vyhodnocení na jiných počítačích bylo nezbytné definovat metody zabezpečení dat proti úmyslnému nebo neúmyslnému poškození. Byla definována přesná struktura formou souboru XML, který obsahuje všechny informace potřebné pro legální použití. Zvolenou metodou zabezpečení přenášených dat proti úmyslnému a neúmyslnému poškození byla sekvence digitálního podpisu a šifrování. Digitální podpisy poskytují prostředky pro zajištění autentizace a integrity dat. Byly použity kombinované symetrické a asymetrické šifrovací algoritmy, které zabezpečily data před zachycením a zároveň umožnily použití veřejných sítí pro přenos. Knihovna implementující digitální podpisy a šifrování je legálně relevantním softwarem a jako taková musí být také odpovídajícím způsobem zabezpečena a identifikována.

Pro ilustraci zamýšleného použití knihovny byly navrženy dvě aplikace, jedna pro zabalení dat na měřicím zařízení, druhá pro zobrazení a ověření zabalených dat uživatelem. Tyto aplikace nejprve ověří kontrolní součet knihovny a poté pokračují v zabalení a ověřování legálně relevantních údajů. Konzolová aplikace s názvem *PackageXML* je určena pro použití na počítači s linuxovým operačním systémem, protože to je obvyklá konfigurace běžných měřicích zařízení. Aplikace převezme soubor XML obsahující údaje vygenerované měřicím zařízením při průjezdu vozidla a pomocí šifrovacího algoritmu RSA tento XML soubor podepíše a zašifruje. Uživatel pak vezme zabalený XML soubor a pomocí grafické aplikace *Vehicle Browser* tyto údaje dešifruje a ověří před analýzou jakýchkoli právních aspektů, jakým je například porušení rychlosti.

Návrhem a implementací softwaru pro úsekové měření rychlosti jsem popsala a demonstrovala proces tvorby softwaru pro stanovená měřidla. Probrala jsem a implementovala metody potřebné k zabezpečení softwaru a naměřených dat proti úmyslnému nebo neúmyslnému poškození. Výsledný software je určen pro využití firmou CROSS Zlín, a.s., se kterou byla tato práce vytvořena, k zajištění alternativního zabezpečení jejich stávajícího projektu zabývajícího se legálními daty ze stanovených měřidel.

Conclusions

Software security is one of the most important aspects of applications working with legally relevant information. Legally controlled measuring devices must comply with the requirements and criteria presented by national and international organizations. The WELMEC Guide 7.2 is the main reference point of this thesis, as it provides the requirements necessary to meet the national legal demands.

The design of the legally relevant software intended to secure data from a speedometer resulted directly from the requirements presented by the WELMEC Guide 7.2 and the configuration of the measuring device. Due to the necessity of transporting legally relevant data from the measuring device to be further evaluated on other computers, it was essential to define methods securing it against intentional or unintentional damage. A precise structure for the data was defined as an XML file containing all the information needed for legal use. The chosen method for securing the transferred data against intentional and unintentional damage was a sequence of digital signatures and encryption. Digital signatures provide a means for ensuring data authentication and integrity. Combined symmetrical and asymmetrical encryption algorithms were used to secure the data from being intercepted while enabling the use of public networks for transportation. The library implementing the digital signatures and encryption is legally relevant software and as such must also be secured and identified accordingly.

In order to illustrate the intended use of the library, two applications were designed, one to package data on the measuring device, the other to display and verify the packaged data by the user. These applications first verify the check sum of the library and then continue to package and verify the legally relevant data. The console application named *PackageXML* is used on a computer with a Linux operating system, as this is the usual configuration of the measuring devices used. The application takes an XML file containing the data generated on the measuring device when a vehicle passes through and using the RSA encryption algorithm signs and encrypts this XML file. The user then takes the packaged XML file and using the graphical application *Vehicle Browser* decrypts and verifies this data before analyzing any legal aspects such as speed violations.

By designing and implementing the software for speedometers I have described and demonstrated the process of creating software for legally controlled measuring devices. I have discussed and implemented the methods needed to secure the software and measured data against intentional or unintentional damage. The resulting software is intended to be used by the company CROSS Zlin, a.s. with whom this thesis was created, in order to provide required security for their existing project dealing with legal data from legally controlled devices.

A Obsah přiloženého CD

K práci je přiloženo CD, které obsahuje celý text práce, zdrojové kódy softwaru, soubor s instrukcemi pro spuštění softwaru a ukázková data pro testování softwaru.

bin/
: Program PackageXML a instalátor programu Vehicle Browser, spustitelné přímo z CD. Adresář obsahuje i všechny runtime knihovny a další soubory potřebné pro bezproblémový běh instalátoru a programu z CD.

doc/
: Text práce ve formátu PDF, vytvořený s použitím stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu.

src/
: Zdrojový kód knihovny XMLPackageLibrary, kompletní zdrojové kódy konzolové aplikace PackageXML a grafické aplikace Vehicle Browser se všemi potřebnými zdrojovými kódy, knihovnami a dalšími soubory pro bezproblémové vytvoření spustitelných verzí programu.

readme.txt
: Instrukce pro instalaci a spuštění programů XMLPackageLibrary a Vehicle Browser.

data/
: Ukázková a testovací data v podobě XML souborů představující průjezdy vozidel a souborů obsahující RSA klíče použitých v práci a pro potřeby testování práce při tvorbě posudků a obhajoby práce.

Literatura

[1] AUMASSON, Jean-Philippe. *Serious cryptography: a practical introduction to modern encryption*. No Starch Press, 2017.

[2] BARKER, Elaine, et al. *Recommendation for key management: Part 1: General (Revision 5)*. Gaithersburg, MD, USA: National Institute of Standards and Technology, Technology Administration, 2020.

[3] BARTEL, Mark, et al. *XML Signature Syntax and Processing Version 1.1* [online]. 11 April 2013 [cit. 2022-03-15]. Dostupné z: https://www.w3.org/TR/xmldsig-core1/

[4] BSI, Cryptographic Mechanisms. Recommendations and Key Lengths. *Technical Guideline*, 2020.

[5] Český metrologický institut. *Základní pojmy* [online]. [cit. 2022-01-20]. Dostupné z: https://www.cmi.cz/node/537

[6] HAROLD, Elliotte Rusty; MEANS, W. Scott. *XML in a nutshell: a desktop quick reference*. "O'Reilly Media, Inc.", 2004.

[7] HEJLSBERG, Anders, et al. *The C# programming language*. Pearson Education, 2008.

[8] IMAMURA, Takeshi, et al. *XML Encryption Syntax and Processing Version 1.1* [online]. 11 April 2013 [cit. 2022-03-15]. Dostupné z: https://www.w3.org/TR/xmlenc-core1/

[9] KOVAL, Martin. *Validácia SW podľa WELMEC 7.2, 2015 – Časť 1*. Metrologie [online]. Praha, 2018, 2018(3), 14-17 [cit. 2022-1-15]. ISSN 1210-3543. Dostupné z: https://www.unmz.cz/files/metrologie/casopis/Metrologie%203-18%20%20-%20%20web.pdf

[10] KOVAL, Martin. *Validácia SW podľa WELMEC 7.2, 2015 – Časť 2*. Metrologie [online]. Praha, 2018, 2018(4), 10-18 [cit. 2022-1-15]. ISSN 1210-3543. Dostupné z: https://www.unmz.cz/files/metrologie/casopis/Metrologie%204-18%20-%20small%20-%20www.pdf

[11] MACVITTIE, Lori A. *XAML in a Nutshell*. "O'Reilly Media, Inc.", 2006.

[12] Národní metrologický systém České republiky. *ÚNMZ – Úřad pro technickou normalizaci, metrologii a státní zkušebnictví* [online]. [cit. 2022-01-15]. Dostupné z: https://www.unmz.cz/metrologie/metrologicky-system/narodni-metrologicky-system-ceske-republiky/

[13] *Opatření obecné povahy: Silniční rychloměry*. Brno, 2010, 0111-OOP-C005-09.

[14] PERLMAN, Radio. An overview of PKI trust models. *IEEE network*, 1999, 13.6: 38-43.

[15] *Požadavky na provedení a kvalitu inteligentních dopravních systémů na dálnicích a silnicích ve správě Ředitelství silnic a dálnic ČR*. Praha, 2016. Dostupné z: https://www.rsd.cz/documents/20125/46605/PPK_ITS_06-16.pdf?t=1639569601439

[16] Public Key Infrastructure (PKI). *ENISA* [online]. [cit. 2022-03-10]. Dostupné z: https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/public-key-infrastructure-pki

[17] Le, Règles et recommandations concernant. Guide des mécanismes cryptographiques. 2020. Dostupné z: https://perso.univ-rennes1.fr/sylvain.duquesne/master/documents/Anssi-mecanismes.pdf

[18] WELMEC 7.2. *Software Guide: Measuring Instruments Directive 2014/32/EU*. Braunschweig, Germany: WELMEC Secretariat, 2020. Dostupné z: https://www.welmec.org/welmec/documents/guides/7.2/2020/WELMEC_Guide_7.2_v2020.pdf