FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	On-line správa klíčů pro vlakový zabezpečovací systém ETCS
Student:	Bc. Michal Kunert
Vedoucí:	Ing. Jiří Buček, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Počítačová bezpečnost
Katedra:	Katedra informační bezpečnosti
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

European Train Control System (ETCS) je vlakový zabezpečovací systém, který je nedílnou součástí evropského systému řízení železničního provozu (ERTMS).
V rámci ETCS probíhá komunikace mezi jeho jednotlivými zařízeními. Pro zajištění bezpečnosti této komunikace se využívá symetrická kryptografie a všichni účastníci dané komunikace tak musejí disponovat sdíleným tajným klíčem. Hlavním cílem této práce je vytvoření systému pro on-line správu těchto sdílených klíčů.

Úkoly:
1. Nastudujte a analyzujte dokumenty k on-line i off-line správě klíčů zabezpečovacího systému ETCS [1,2,3]
2. Definujte jednotlivé funkční a nefunkční požadavky kladené na systém pro on-line správu klíčů s ohledem na požadavky kladené dokumenty [1,2,3].
3. Navrhněte systém pro on-line správu kryptografických klíčů zabezpečovacího zařízení ETCS splňující definované funkční i nefunkční požadavky.
4. Implementujte prototyp systému na platformě typu PC, případně jednodeskovém počítači (Raspberry Pi apod.). V rámci prototypu implementuje SW pouze pro zařízení, která zajišťují distribuci klíčů.
5. Otestujte implementovaný prototyp.

[1] UNISIG. SUBSET-38 - Off-line Key Management FIS. 2015. Tech. zpr., verze: 3.1.0. European Union Agency for Railways. Dostupné také z https://www.era.europa.eu/sites/default/files/filesystem/ertms/ccs_tsi_annex_a_-_mandatory_specifications/

Elektronicky schválil/a prof. Ing. Róbert Lórencz, CSc. dne 4. ledna 2023 v Praze.

set_of_specifications_3_etcs_b3_r2_gsm-r_b1/index011_-_subset-038_v310.pdf
[2] UNISIG. SUBSET-114 - KMC-ETCS Entity Off-line KM FIS. 2015. Tech. zpr., verze: 1.1.0. European Union Agency for Railways. Dostupné také z https://www.era.europa.eu/sites/default/files/filesystem/ertms/ccs_tsi_annex_a_-_mandatory_specifications/set_of_specifications_3_etcs_b3_r2_gsm-r_b1/index079_-_subset-114_v110.pdf
[3] UNISIG. SUBSET-137 - On-line Key Management FFFIS. 2015. Tech. zpr., verze: 1.0.0. European Union Agency for Railways. Dostupné také z https://www.era.europa.eu/sites/default/files/filesystem/ertms/ccs_tsi_annex_a_-_mandatory_specifications/set_of_specifications_3_etcs_b3_r2_gsm-r_b1/index083_-_subset-137_v100.pdf

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

On-line správa klíčů pro vlakový zabezpečovací systém ETCS

Bc. Michal Kunert

Katedra informační bezpečnosti
Vedoucí práce: Ing. Jiří Buček, Ph.D.

3. května 2023

Poděkování

Rád bych poděkoval Ing. Jiřímu Bučkovi, Ph.D. za rady a pomoc při tvorbě této diplomové práce. Dále bych rád poděkoval rodině a přátelům za podporu během studia.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 3. května 2023 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Kunert, Michal. *On-line správa klíčů pro vlakový zabezpečovací systém ETCS.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Abstrakt

Tato diplomová práce se zabývá návrhem a implementací systému pro on-line správu klíčů vlakového zabezpečovacího zařízení ETCS. Práce začíná analýzou, ve které je uveden popis ETCS a možnosti správy klíčů tohoto zabezpečovacího zařízení. V analýze jsou taktéž definovány požadavky na výsledné řešení s ohledem na dokument ERTMS/ETCS SUBSET-137 a mimo to je v této kapitole také provedena rešerše technologií použitelných k implementaci řešení. Další kapitoly se věnují návrhu a implementaci celého řešení. Výsledný systém se skládá z několika komponent, kterými jsou centrum pro správu klíčů, komponenta pro uchovávání klíčů na koncových entitách a certfikační autorita. Pro přenos zpráv souvisejících se správou klíčů je využit TLS protokol. Pro komunikaci s certifikační autoritou je využit Certificate Management Protocol. Poslední část práce se zabývá testováním funkčnosti implementovaného systému.

Klíčová slova ETCS, kryptografické klíče, infrastruktura veřejného klíče, TLS, CMP

Abstract

The main aim of this diploma thesis is to design and implement an on-line key management system for the ETCS train control system. The thesis starts with the analysis, which contains a description of ETCS and the key management options for this train control system. The analysis also contains specification of requirements for the solution with regard to ERTMS/ETCS SUBSET-137 and this chapter also contains a description of suitable technologies for implementation. The next chapters deal with the design of the application and description of the implementation. The final system is composed of a few components, namely the key management centre, the component for storing keys on the end entity and the certification authority. The TLS protocol is used for the transmission of key management related messages. The Certificate Management Protocol is used to communicate with a certification authority. The last part of the thesis deals with testing the functionality of the implemented system.

Keywords ETCS, cryptographic keys, public key infrastructure, TLS, CMP

Obsah

Seznam obrázků

Seznam výpisů kódu

Úvod

Během několika posledních let bylo možné pozorovat zvyšující se zájem o železniční dopravu v segmentu osobní i nákladní dopravy, což mělo za následek zvýšení počtu vlaků na železničních tratích. Na některých z nich již dochází volná kapacita, což brání dalšímu rozvoji železniční dopravy. Vyčerpaná kapacita se projevuje problémy, které známe asi všichni, a jsou jimi neustálá zpoždění a celková nespolehlivost systému. Hustý provoz taktéž zvyšuje riziko nehod či jiných mimořádných událostí navzdory tomu, že se železnice řadí mezi nejbezpečnějším druhy dopravy. O bezpečný provoz na železnici se kromě traťových, respektive staničních zabezpečovacích zařízení starají rovněž vlaková zabezpečovací zařízení. Tato technologie dokáže zabránit projení návěsti stůj nebo také může dohlížet na dodržování nejvyšší dovolené rychlosti. V současné době existuje velké množství těchto zařízení, což snižuje interoperabilitu provozu z důvodu nekompatibility v rámci jednotlivých států. Dalším problémem je rovněž rozdílný stupeň zabezpečení jednotlivých systémů.

Výše zmíněné problémy má vyřešit evropský vlakový zabezpečovací systém ETCS. Tento systém má zvýšit bezpečnost železničního provozu a taktéž zvýšit propustnost tratí. ETCS potřebuje pro svou činnost komunikovat se svou traťovou částí. Tato komunikace je zabezpečena pomocí symetrické kryptografie. Tento typ kryptografie předpokládá znalost sdíleného tajného klíče oběma stranami komunikace, proto je nutné tento tajný klíč mezi obě zařízení distribuovat. Problematikou distribuce symetrických klíčů se zaměřením na on-line přenos se tato práce zabývá.

Text se bude skládat z analýzy požadavků na výsledné řešení, rešerše norem a použitelných technologií. Na tyto kapitoly navážou sekce věnující se návrhu řešení a jeho implementaci.

Kapitola 1

Cíl práce

Hlavním cílem této práce je navrhnout a implementovat systém, který umožní správu symetrických kryptografických klíčů používaných v rámci zabezpečovacího zařízení ETCS. Z důvodu zajištění interoperability je nutné, aby navržený systém plnil požadavky definované Agenturou Evropské unie pro železnice. V rámci tohoto systému musí existovat zařízení, která budou klíče generovat, spravovat a zajišťovat jejich distribuci. Tato distribuce bude směřovat na koncová zařízení, která přijaté klíče využijí k činnosti nesouvisející s touto prací. Přenos klíčů by měl být realizován on-line způsobem pomocí počítačové sítě.

Tento hlavní cíl by měl být realizován pomocí menších dílčích cílů. Prvním cílem je rešerše požadavků kladených na systém v rámci předpisů pro zajištění interoperability. Na základě toho mají být sestaveny a popsány jednotlivé funkční a nefunkční požadavky kladené na výsledné řešení. Náplní třetího cíle bude rešerše technologií, které bude možné využít při implementaci řešení. Součástí čtvrtého cíle bude návrh řešení v souladu se stanovenými požadavky. Předposledním krokem bude implementace řešení a nakonec bude následovat zhodnocení bezpečnosti výsledného řešení.

Kapitola 2

Analýza

2.1 ERTMS

European Rail Traffic Management System (ERTMS) je jednotný evropský systém pro řízení železniční dopravy. Cílem tohoto projektu je vytvořit interoperabilní železniční systém, který má být bezpečnější, propustnější a efektivnější. Vyšší automatizace železniční dopravy je také cílem tohoto projektu. [1, 2]

Jedním z důvodu vzniku tohoto standardu byla existence velkého množství různých vlakových zabezpečovacích zařízení napříč státy Evropské unie. Jedná se například o liniový vlakový zabezpečovač LS (LS 90, LS 06, MIREL), LZB, PZB, SCMT a další. Tento stav brání volnému pohybu železničních vozidel mezi různými státy, protože je na hranicích často nutná výměna hnacích vozidel nebo je potřeba mít na jednom vozidle mnoho vlakových zabezpečovačů, což zvyšuje náklady na provoz. Různá vlaková zabezpečovací zařízení zajišťují také různé úrovně ochrany.

ERTMS se skládá ze dvou hlavních částí, kterými jsou ETCS (detailně popsáno v kapitole 2.2) a GSM-R, které poskytuje bezdrátové spojení mezi traťovými zařízeními a vozidly. Jedná se o systém, který je založen na rádiové technologii GSM a který využívá výhradní frekvenční pásma pro komunikaci na železnici. GSM-R se používá jednak pro hlasovou komunikaci, například mezi strojvedoucím a výpravčím, ale také k přenosu dat potřebných pro systém ETCS. O GSM-R a jeho bezpečnosti detailněji pojednává kapitola 2.5. [3]

2.2 ETCS

European Train Control System (ETCS) je evropský vlakový zabezpečovací systém, jehož cílem je zvýšení bezpečnosti železniční dopravy. Vlakový zabezpečovač je zařízení, které umí v případě omylu nebo selhání strojvedoucího aktivně zasáhnout do řízení vlaku. ETCS přenáší kromě dalšího také informace

o povolení k jízdě a maximální povolené rychlosti z traťové části systému do palubní části. Tato data jsou strojvedoucímu zobrazována na stanovišti. Palubní část neustále porovnává maximální povolenou rychlost v aktuálním místě s aktuální rychlostí vlaku a v případě potřeby aktivuje brzdění. [4]

ETCS umožní vlaku, který jede pod jeho plným dohledem (Full Supervision Mode), jízdu pouze v případě, že daný vlak disponuje oprávněním k jízdě (movement authority). Toto oprávnění je vydáváno radioblokovou centrálou (RBC) nebo podobným zařízením na základě informace od traťového nebo staničního zabezpečovacího zařízení. Povolení k jízdě má časové nebo prostorové ohraničení s cílovou nulovou rychlostí. [4]

Mimo maximální dovolené rychlosti ETCS kontroluje dodržení směru jízdy vlaku, dodržení přechodných omezení a dodržení trasy vlaku.

2.2.1 Aplikační úrovně ETCS

ETCS lze provozovat v několika různých aplikačních úrovních, které se od sebe liší úrovní zabezpečení i zařízeními, která jsou použita. Popisované zabezpečovací zařízení je možné provozovat i na tratích vybavených národním vlakovým zabezpečovačem. [4]

2.2.1.1 Aplikační úroveň 0 – ETCS L0

Tato úroveň umožňuje vozidlům vybaveným mobilní částí systému ETCS používat tratě, které nejsou vybaveny traťovou částí systému. Vlakový zabezpečovač v takovém případě hlídá pouze maximální dovolenou rychlost. Maximální rychlost je myšlena ve smyslu rychlosti vlaku na celou trasu (definováno před výjezdem vlaku), ne ve smyslu maximální rychlosti pro nějaký úsek trati. [5]

2.2.1.2 ETCS Regional

Jedná se o zjednodušenou variantu ETCS, která by měla být využívána hlavně na tratích řízených předpisem D3 případně na méně vytížených tratích D1. Jedná se o ETCS L1 Limited Supervision a ETCS STOP. První z nich je bodový zabezpečovač postavený na ETCS L1, který zajišťuje **neprojetí** návěsti STŮJ, dále také **omezeně** dohlíží na nepřekročení maximální povolené rychlosti pro omezený počet rychlostních profilů. ETCS STOP zajistí nouzového brzdění vozidla v případě nedovoleného **projetí** návěstidla zakazujícího jízdu. Maximální povolená rychlost na trati s ETCS STOP je 100 km/h. Kontrolu nepřekročení maximální povolené rychlosti tato úroveň zabezpečovacího zařízení nepodporuje. [6]

Obě zmíněné varianty využívají přepínatelné a nepřepínatelné Eurobalízy u většiny návěstidel. Pro tuto aplikační úroveň je nutné zřídit konvenční návěstidla. [6]

2.2.1.3 Aplikační úroveň 1 – ETCS L1

Tato úroveň je určena pro tratě s klasickým traťovým a staničním zabezpečovacím zařízením. Tratě podporující tuto úroveň musejí být vybaveny přepínatelnými přenosovými prostředky (Eurobalízy, Eurosmyčky), které vlaku předávají oprávnění k jízdě a další provozní informace. Eurobalízy, případně Eurosmyčky jsou připojeny ke klasickému traťovému zabezpečovacímu zařízení, které mj. provádí kontrolu celistvosti vlaku. Balízy se využívají také jako referenční body pro určení polohy vlaku a směru jeho jízdy, v tomto případě se může jednat o jejich nepřepínatelnou variantu. Tato úroveň nevyžaduje síť GSM-R, návěstidla jsou naopak vyžadována. [4, 5]

2.2.1.4 Aplikační úroveň 2 – ETCS L2

Aplikační úroveň 2 je opět možné provozovat na tratích vybavených klasickým traťovým a staničním zabezpečovacím zařízením. Na rozdíl od úrovně L1 probíhá neustálý přenos dat mezi radioblokovou centrálou a vozidlem. Využití sítě GSM-R přináší možnost okamžité aktualizace oprávnění k jízdě vlaku, díky tomu se na rozdíl od úrovně L1 například strojvedoucí dozví o nově uděleném oprávnění k jízdě ještě předtím než spatří konkrétní návěstidlo. Z důvodu využití bezdrátové sítě odpadá nutnost pokládat kabely mezi přepínatelnými balízami a staničním respektive traťovým zabezpečovacím zařízením. Přepínatelné balízy nejsou použity, ale jsou využívány pouze jejich nepřepínatelné varianty určující pozici vlaku a směr jeho jízdy. Použití návěstidel je v této aplikační úrovni volitelné. [4]

2.2.1.5 Aplikační úroveň 3 – ETCS L3

Tato aplikační úroveň opět využívá síť GSM-R a nepřepínatelné balízy. Na rozdíl od úrovně L2 je potřeba, aby vozidlo hlásilo radioblokové centrále kromě své polohy i informace o celistvosti vlaku. V této úrovni je nutné, aby byla všechna vozidla na trati vybavena systémem pro bezpečné detekce celistvosti vlaku. Úroveň ETCS L3 nevyžaduje zřízení klasického zabezpečovacího zařízení, protože souhlas k přestavování výhybek dává RBC (může výhybky rovnou ovládat). Pokud je klasické zabezpečovací zařízení zřízeno, tak i v tomto případě musí dávat souhlas k přestavení výhybek RBC. [4]

V této úrovni není nutné zřizovat hlavní návěstidla, oddílová návěstidla a předvěsti, protože ETCS L3 dovoluje používat pohyblivé prostorové oddíly. To znamená, že následujícímu vlaku je uděleno oprávnění k jízdě (movement authority) až k místu, kde se nachází konec předešlého vlaku. Pohyblivé oddíly nevyžadují zřízení klasických zařízení detekce vlaků (kolejové obvody, počítače náprav), z tohoto důvodu je nutné hlásit celistvost vlaku. Pohyblivé prostorové oddíly dokáží výrazným způsobem zvýšit kapacitu trati. [4]

2.2.2 ETCS v České republice

První železniční tratí v České republice, která byla vybavena vlakovou částí zabezpečovacího zařízení ETCS byl úsek *Poříčany – Kolín*. Hlavním cílem tohoto projektu byla implementace systému ETCS do podmínek železnice v České republice. Tento projekt byl zahájen v roce 2005 a v roce 2011 byl ukončen. V rámci zmíněného projektu bylo vybaveno také několik málo vozidel mobilní částí systému ETCS. [7]

Podle Národního implementačního plánu ERTMS, který byl schválen vládou České republiky v roce 2017, se počítá se zavedením výhradního provozu zabezpečovacího zařízení ETCS k 1. 1. 2025 na vybraných koridorových tratích. Na trať s výhradním provozem tak nebude možné vypravit vlak, který není vybavený mobilní částí vlakového zabezpečovače ETCS. První úseky s výhradním provozem by měly být: *Děčín – Praha – Česká Třebová – Brno – Břeclav*, *Břeclav – Bohumín* a *Česká Třebová – Přerov*. [8]

Podle [6] jsou v době psaní této práce vybaveny systémem ETCS v České republice tratě: *Praha – Český Brod*, *Kolín – Česká Třebová – Brno*, *Brno – Břeclav – Kúty*, *Praha – Votice*, *Plzeň – Cheb*, *Česká Třebová – Brodek u Přerova*, *Břeclav – Přerov – Petrovice u Karviné* a další hraniční úseky. Ostatní koridorové i vedlejší tratě budou postupně tímto zabezpečovacím zařízením dovybavovány. Na koridorových tratích bude instalováno ETCS aplikační úrovně 2. Na lokálních tratích je počítáno se zjednodušenými verzemi ETCS L1 Limited Supervision a ETCS STOP.

Se zavedením výhradního provozu ETCS je také nutné doplnit nevybavená vozidla o mobilní částí tohoto zabezpečovacího zařízení. České dráhy plánují mimo jiné dosadit palubní část tohoto systému na jednotky 680 – Pendolino, 471 – CityElefant, ale také na elektrické lokomotivy řady 362. Druhou možností je pořízení zcela nových vozidel, které jsou vozidlovou částí ETCS vybavena již z výroby. [9]

2.3 Části ETCS

Zabezpečovací systém ETCS využívá různá zařízení, která jsou definována ve standardu ERTMS. Tato zařízení musí spolu vzájemně komunikovat.

2.3.1 Mobilní část

Jedná se o část systému ETCS, která je umístěna přímo ve vozidle. Mobilní část komunikuje s traťovou částí, ale v některých případech probíhá komunikace i s dalšími zařízeními mimo systém ETCS. Za tuto část ve většině případů zodpovídá provozovatel vozidla tedy většinou dopravce.

2.3.1.1 Euro Vital Computer (EVC)

Jedná se o jádro mobilní části ETCS obsahující logiku ochrany vlaku. Úkolem této logiky je vyhodnocování různých údajů, výpočet brzdné křivky a v případě potřeby aktivace brzdění. S EVC komunikují všechna ostatní zařízení, jako je odometr nebo přijímač dat z GSM-R. [10]

2.3.1.2 Driver Machine Interface (DMI)

DMI je rozhraní, které využívá strojvedoucí pro komunikaci se zabezpečovacím zařízením ETCS. Ve většině případů se jedná o dotykovou obrazovku, která zobrazuje aktuální rychlost, nejvyšší povolenou rychlost pro daný úsek, budoucí změny rychlosti a další. DMI se také používá pro zadávání dat o vlaku do zabezpečovacího zařízení. Jedná se například o délku vlaku, brzdicí procenta. [10]

2.3.1.3 Euroradio

Euroradio modul je využíván pro komunikaci s traťovou částí zabezpečovacího zařízení pomocí sítě GSM-R. Jedná se o zařízení, které zpracovává signál přijatý z antény a tak tedy přijímá datové zprávy z RBC nebo RIU. Euroradio modul samozřejmě také odesílá datové zprávy. [10]

2.3.1.4 Další zařízení mobilní části

Balise Transmission Module (BTM) respektive Loop Transmission Module (LTM) jsou zařízení, která zpracovávají data z antény určené pro přijímání dat z **Eurobalízy** respektive **Eurosmyčky**. Juridical Recording Unit (JRU) je modul poskytující funkčnost „černé skříňky“. Dále je využit odometr pro výpočet ujeté vzdálenosti. Train Interface Unit (TIU) se využívá pro komunikaci mezi ETCS a vozidlem. Pomocí tohoto rozhraní je přenášena například poloha páky určující směr jízdy. V neposlední řadě se využívá modul pro ukládání klíčů, kterému je věnována kapitola 2.6.4. [10]

2.3.2 Traťová část

S traťovou částí komunikuje mobilní část zabezpečovacího zařízení ETCS. Některé moduly této části se nacházejí přímo v kolejišti, jiné moduly jsou umístěny v budovách provozovatele železnice. Za traťovou část většinou zodpovídá provozovatel infrastruktury.

2.3.2.1 Eurobalíza

Jedná se o zařízení, které je umístěno na trati mezi kolejnicemi. Eurobalízy existují v aktivní a pasivní verzi. Pasivní Eurobalíza se využívá v aplikační úrovni 1, 2 a 3. Tato balíza vysílá vždy stejnou informaci (referenční údaj

o poloze). Aktivní balíza je využita v aplikační úrovni jedna a umožňuje vysílat proměnnou informaci (například je nebo není uděleno povolení k jízdě). Nepřepínatelné eurobalízy nepotřebují ke své činnosti napájení, protože jsou napajeny jedoucím vlakem pomocí antény.[10]

2.3.2.2 Eurosmyčka

Jedná se o volitelné zařízení, které se využívá u aplikační úrovně jedna. Jeho funkce je podobná přepínatelné Eurobalíze, ale oproti balíze umožňuje přenos na delším úseku tratě. Svým způsobem se tak jedná o kontinuální přenos. [10]

2.3.2.3 Lineside Electronic Unit (LEU)

LEU je zařízení představující rozhraní mezi přepínatelnými Eurobalízami (Eurosmyčkami) a traťovým zabezpečovacím respektive staničním zabezpečovacím zařízením. LEU tak získává data ze stavědla a odesílá příslušné informace (tzv. telegram) do Eurobalíz. Tento modul je využit u ETCS L1. [10]

2.3.2.4 Radio Infill Unit (RIU)

Opět se jedná o volitelnou část aplikační úrovně jedna. Tento modul umožňuje odeslat vozidlu informace z traťového respektive staničního zabezpečovacího zařízení ještě předtím než vozidlo přejede Eurobalízu nebo Eurosmyčku. Tato funkcionalita funguje na tratích, kde je dostupná síť GSM-R. Vozidlo tak dostává stále aktuální informace, aniž by muselo přejet Eurobalízu nebo Eurosmyčku. Oproti úrovni 2 a 3 se může jednat jen o malou oblast (velkou stanici, nepřehledný úsek). [10]

2.3.2.5 Radio Block Centre (RBC)

Toto zařízení, které se využívá v rámci aplikačních úrovni 2 a 3, funguje jako centrální jednotka traťové části zabezpečovacího zařízení ETCS. Jeho úkolem je příjem informací o poloze vlaku a odesílání povolení k jízdě (movement authority). RBC komunikuje se zabezpečovacím zařízením a získává od něj relevantní informace. Je také schopno komunikovat se sousedními RBC a umožňuje vydávat souhlas k přestavování výhybek. [10]

2.3.2.6 Key Management Centre (KMC)

KMC spravuje kryptografické klíče, které jsou potřeba pro zajištění bezpečné komunikace mezi traťovou a mobilní částí ETCS prostřednictvím protokolu EURORADIO. V rámci správy klíčů zajišťuje toto zařízení generování nových klíčů, generování požadavků pro práci s klíčem na cílovém zařízení (úložiště klíčů v RBC nebo v mobilní části) a úpravu existujících klíčů. K jednodušímu přenosu klíčů slouží zařízení Key distribution center (KDC). [11, 12] Detailnější popis KMC je uveden v kapitole 2.6.2.

2.3.2.7 ETCS módy

ETCS dokáže pracovat v několika různých režimech. Režim neboli mód se na rozdíl od ETCS úrovní (L1, L2, L3), které řeší komunikaci mezi vlakem a tratí, vztahuje k provozním okolnostem na trati nebo ke stavu palubního zařízení.

Hlavním módem je plný dohled (Full Supervision). V tomto režimu je ETCS plně zodpovědné za to, aby nebyla překročena maximální povolená rychlost a aby se vozidlo pohybovalo pouze v úseku, které mu vymezuje jeho oprávnění k jízdě. Zabezpečovací systém ETCS je v režimu plného dohledu, v případě že má mobilní část k dispozici všechna data o vlaku a trati. Podle situace existují i jiné režimy. Jedná se o posun, jízdu na zodpovědnost strojvedoucího, jízdu podle rozhledových poměrů, jízdu po trati nevybavené traťovou částí ETCS a další. [5]

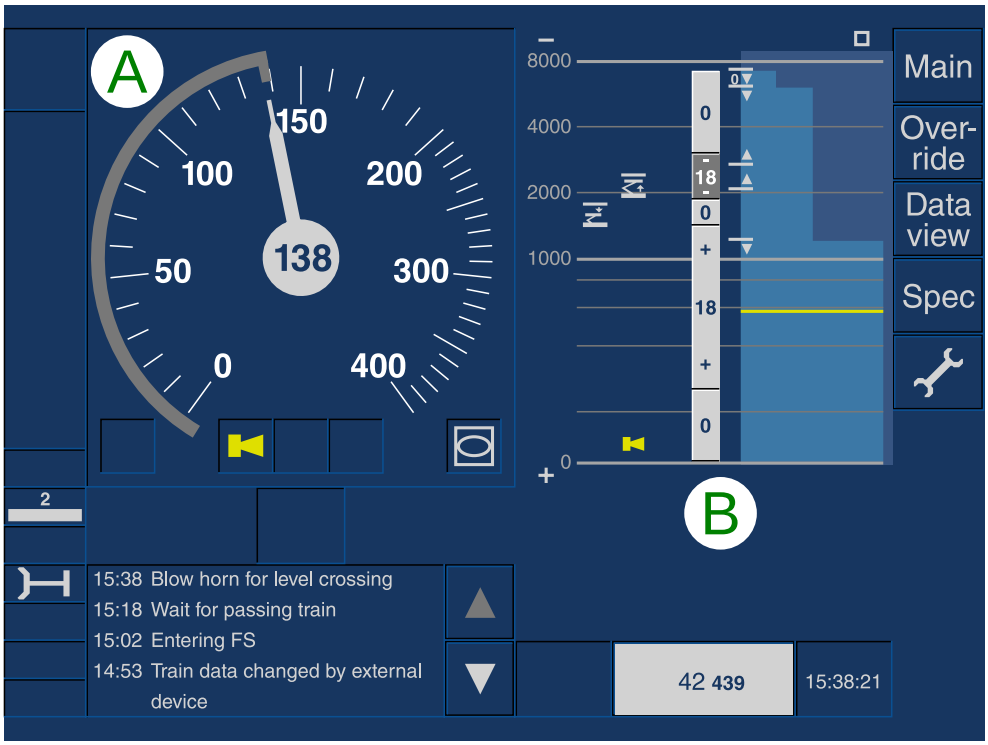
2.4 Princip zabezpečení jízdy vlaku

V této kapitole bude stručně a zjednodušeně uveden princip zabezpečení jízdy vlaku pomocí systému ETCS. K popisu bude využit obrázek 2.1, na kterém je znázorněna obrazovka mobilní části ETCS (DMI) sloužící jako rozhraní mezi strojvedoucím a zabezpečovacím zařízením. Při popisu bude uvažována úroveň L2 a režim plného dohledu ETCS.

Oblast obrázku 2.1 označená písmenem **A** (v bílém kruhu) slouží pro zobrazení aktuální rychlosti vozidla (světle šedivá ručička a číslo uvnitř) a nejvyšší povolené rychlosti pro místo, kde se vozidlo právě nachází (tmavě šedivý půlkruh po okraji ciferníku). Tato maximální rychlost je ovlivněna nejvyšší povolenou traťovou rychlostí, maximální rychlostí soupravy a také brzdnou křivkou. Tato křivka se generuje při očekávaném snížení povolené rychlosti a určuje jakou rychlostí se má vozidlo pohybovat v závislosti na daném snížením rychlosti. Z důvodu využití brzdné křivky bude zabezpečovač vyžadovat postupné snižování rychlosti ještě před vjetím vlaku do úseku se sníženou rychlostí. Místo, kam vlak již nemá povoleno vjet (tímto místem končí movement authority), je interpretováno zabezpečovačem jako místo s maximální povolenou rychlostí 0 km/h. Pokud se strojvedoucí neřídí nejvyšší povolenou rychlostí pro daný úsek (brzdnou křivkou), je aktivováno brzdění. Oblast obrázku 2.1 označená písmenem **B** je plánovací oblast, která zobrazuje změny rychlosti v dohledné vzdálenosti (na obrázku 8 km).

2.5 GSM-R

Zabezpečovací zařízení ETCS využívá ke své činnosti na aplikačních úrovních 2 a 3 síť GSM-R. Ta se liší od veřejné GSM sítě hlavně svojí spolehlivostí. Na rozdíl od veřejné GSM sítě se GSM-R buňky snaží pokrýt jen omezené území v těsném okolí trati, a to co nejdůsledněji bez hluchých míst. Z tohoto důvodu



Obrázek 2.1: Driver Machine Interface – Sekce označená písmenem **A** udává aktuální a nejvyšší povolenou rychlost, část určená písmenem **B** značí plánovací oblast. [13], upraveno

jsou pro GSM-R buňky typické velké vzájemné překryvy. Buňka je místo obsluhované jedním vysílačem. Síť je koncipována tak, aby buňky byly co nejdelší a nejužší. Způsob pokrytí také souvisí s adaptací této sítě na obsluhu vlaků jedoucích rychlostí přes 350 km/h. Běžná GSM síť je navržena pouze na rychlost do 250 km/h. Dalším rozdílem mezi veřejnou GSM sítí a GSM-R sítí je možnost nastavení priority při sestavování spojení, což znamená, že některé typy hovorů jsou důležitější než jiné a mají tedy přednost před těmi s nižší prioritou. GSM-R dále podporuje skupinové spojení, u kterého je možné spojení se skupinou uživatelů. [14]

GSM-R využívá bezpečnostní algoritmy A5, A3 a A8. První zmiňovaný algoritmus je zodpovědný za šifrování datového provozu mezi BTS (Base Transceiver Station) a koncovým zařízením. Algoritmy A3 a A8 jsou používány k ověření totožnosti a ke generování klíče pro šifrování hlasového a datového provozu. V praxi jsou algoritmy A3 a A8 obvykle implementovány společně pod názvem A3/A8. [15]

I přes využití principů popsaných v předchozím odstavci je přenos dat mezi zařízeními zabezpečovacího zařízení ETCS (například mezi RBC a OBU) pomocí GSM-R sítě považován za otevřený, a proto je nutné počítat s neopráv-

něným přístupem a s možností neoprávněné manipulace s daty procházejícími přes tento kanál. Ke komunikaci mezi zařízeními ETCS se využívá protokol Euroradio (ER), který disponuje bezpečnostní funkcí pro zajištění autenticity a integrity přenášených dat. K navázání bezpečného spojení se používá identifikační a autentizační (I&A) dialog, který by měl probíhat vždy při zahajování komunikace. [16]

Autenticita a integrita je v rámci protokolu Euroradio zajišťována pomocí autentizačního kódu MAC, který je vypočten podle standardu ISO/IEC 9797-1 s drobnou úpravou. K výpočtu je tak využita šifra DES s modifikovaným algoritmem MAC 3. Modifikace spočívá v tom, že poslední datový blok je vypočten zašifrováním pomocí K1, dešifrováním pomocí K2 a zašifrováním pomocí K3, kde K1, K2 a K3 jsou 64 bitové klíče DES včetně paritních bitů. Poslední krok je tedy **modifikován** oproti standardu ISO/IEC 9797-1, který využívá pouze dva klíče. Pro výpočet MAC kódu je využita také ISO 9797-1 Padding 1 metoda. Výsledkem je 64 bitová hodnota CBC-MAC, která je vypočtena na základě zprávy a tří 64 bitových klíčů. [16]

Samotná komunikace je zabezpečena tak, že odesílatel odesílá kromě samotné zprávy také MAC autentizační kód vypočtený ze zprávy (+ dalších údajů jako je délka zprávy, výplň, cílová adresa). Příjemce vypočte po přijetí zprávy MAC kód přijaté zprávy (+ dalších údajů) a porovná přijatý a vypočtený MAC kód. Pokud se tyto hodnoty nerovnají, tak hrozí porušení integrity a autenticity zprávy. V takovém případě příjemce vyvolá chybu. Autentizační kód MAC vyžaduje, aby oba účastníci komunikace disponovali sdíleným tajným klíčem (v rámci ETCS nazýván KMAC). Tento klíč se skládá ze tří 64 bitových DES klíčů (včetně paritních bitů). Pro zabezpečení komunikace pomocí autentizačního kódu MAC není využit přímo tento sdílený klíč, ale využívá se jednorázový klíč pro každou relaci (Session key). Tento klíč je účastníky odvozen na základě sdíleného klíče. [16]

Session klíč, označený jako KSMAC, je odvozen na základě algoritmu, který ze tří 64 bitových klíčů vygeneruje opět triple DES klíč. KSMAC se využívá z bezpečnostních důvodů, aby základní KMAC klíč nikdy nemusel opustit zařízení, ve kterém je uložen. Rovněž také v případě úniku použitého KSMAC klíče nebude možné kompromitovat komunikaci žádné jiné relace. Pro každé spojení by si měli oba účastníci komunikace vygenerovat nový KSMAC. Session klíč je odvozen na základě následujícího algoritmu (uvedeném v [16]):

- Oba účastníci (A, B) si vygenerují svá náhodná (pro každé spojení je nutné vygenerovat jiná) 64 bitová čísla R_A a R_B
- Každé náhodné číslo R_A R_B se dělí na dvě 32 bitové části, tedy $R_A = R_A^L|R_A^R$ a $R_B = R_B^L|R_B^R$
- Tři část KSMAC (K_{S1}, K_{S2} a K_{S3}) jsou vypočteny podle následujících vzorců:

$$K_{S1} := MAC(R_A^L|R_B^L, K_{AB}) =$$

$$= DES(K_3, DES^{-1}(K_2, DES(K_1, R_A^L|R_B^L)))$$

$$K_{S2} := MAC(R_A^R|R_B^R, K_{AB}) =$$
$$= DES(K_3, DES^{-1}(K_2, DES(K_1, R_A^R|R_B^R)))$$

$$K_{S3} := MAC(R_A^L|R_B^L, K'_{AB}) =$$
$$= DES(K_1, DES^{-1}(K_2, DES(K_3, R_A^L|R_B^L))),$$

kde | značí operátor spojení a DES^{-1} značí inverzní operaci k DES, tedy dešifrování.

Tyto tři 64 bitové klíče (včetně paritních bitů) tvoří klíč o velikosti 192 bitů. Tento klíč je již využit k zajištění integrity a autenticity přenášených dat pomocí MAC kódu. [16]

2.6 Správa ETCS klíčů

V kapitole 2.5 byla popsána nutnost existence sdílených klíčů. V rámci ETCS se definuje Key Management System (KMS), který má zajistit dostupnost sdílených kryptografických klíčů na vzájemně komunikujících zařízeních. KMS tedy zajišťuje tvorbu, bezpečné uložení, výměnu a celkovou správu používaných kryptografických klíčů. V rámci KMS se definují zařízení, mezi kterými jsou již zmiňované klíče vyměňovány. Jedná se buď o ETCS entitu, nebo o Key Management Centre (KMC). O druhém z nich pojednávají kapitoly 2.3.2.6 a detailněji také 2.6.2. Mezi ETCS entity (dále jen entity) se řadí Radiobloková centrála, Radio Infill Unit nebo On-Board Unit, což představuje mobilní část ETCS. Přenos klíčů je možné realizovat off-line nebo on-line způsobem. V rámci Key Management Systému se při off-line přenosu pracuje i s pomocnými klíči, které jsou využity pro realizaci bezpečného přenosu KMAC klíčů. Kromě již zmiňovaného KMAC klíče se tak pracuje i s KTRANS a K-KMAC klíči. Detailní popis těchto klíčů je zmíněn v dalších kapitolách. [11, 12]

2.6.1 Hierarchie kryptografických klíčů v ETCS

Dle [16] se klíče používané pro ochranu činnosti zabezpečovacího systému ETCS řadí do tří úrovní. V následujícím seznamu jsou uvedeny všechny používané úrovně. Klíče patřící do úrovně 3 se nepoužívají při on-line správě klíčů (přenos probíhá on-line). Pro off-line správu klíčů se využívají všechny tři úrovně.

- Úroveň 3 – Transportní klíče
  - **KTRANS** – Zabezpečení komunikace mezi KMC a ETCS entitami. Tento klíč se skládá ze dvou podklíčů. První část se využívá k zajištění autenticity a integrity. Druhá část je využívána pro šifrování přenášených dat.
  - **K-KMAC** – Zabezpečení komunikace mezi dvěma KMC. Tento klíč se opět skládá ze dvou podklíčů se stejnými účely jako v případě KTRANS klíče.
- Úroveň 2 – Autentizační klíče
  - **KMAC** – Zajištění autentizace mezi OBU a RBC (případně RIU) během zřizování bezpečného spojení protokolu Euroradio. Tento klíč slouží k odvození klíče pro relaci, který je součástí první úrovně.
- Úroveň 1 – Klíče pro relaci
  - **KSMAC** – Zajištění autenticity a integrity přenášených dat mezi OBU a RBC (případně RIU) v rámci jednoho spojení.

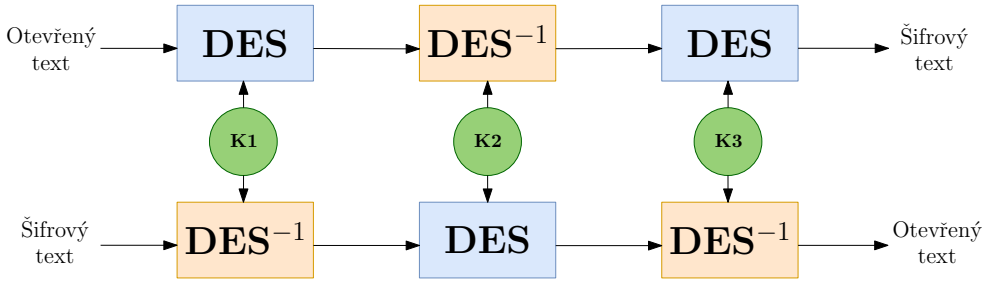
2.6.2 Detailnější popis Key Management Centre (KMC)

Jedná se o zařízení, které již bylo popisováno v kapitole 2.3.2.6, ale v této kapitole bude popis zpřesněn. Key Management Centre je zařízení spadající do systému správy klíčů zabezpečovacího systému ETCS. Na toto zařízení jsou kladeny různé požadavky z hlediska funkčnosti. Tyto požadavky se liší na základě toho, zdali se jedná o KMC podporující off-line nebo on-line přenos klíčů. V této kapitole budou uvedeny požadavky, které nejsou závislé na typu přenosu. Závislé funkcionality budou uvedeny v kapitolách 2.7.1 a 2.7.2.

KMC je mimo jiné zodpovědné za generování klíčů. Požadavky na generování kryptografických klíčů mohou zadávat pouze pověřené osoby. Každý vygenerovaný klíč musí být jednoznačně identifikovatelný a musí být generován náhodně, aby se zabránilo možnému předvídání. Všechny vygenerované klíče musí být přiřazeny entitám, které je budou využívat. Entity se dělí na *výchozí* a na tzv. *peery*, což jsou entity, se kterými bude umožněno výchozí entitě klíče komunikovat. Výchozí entita může být pro každý klíč pouze jedna, ale peerů může být více. Vygenerované klíče jsou uchovávány v KMC, které zodpovídá za jejich bezpečné uložení. ETCS klíče mají definovanou svoji časovou platnost, která je nastavována osobou generující daný klíč. Key Management Centre hlídá, aby nebylo možné vytvořit pro pár výchozí entita – peer dva klíče s překrývající se časovou platností. Každý klíč je identifikován svým sériovým číslem. Je možné přiřadit klíč se stejnou hodnotou více KMAC entitám, ale takto přiřazené klíče musejí mít odlišný identifikátor. Zmíněný identifikátor se skládá ze sériového čísla a také z čísla KMC, které ho vytvořilo. [11, 12]

V rámci ETCS standardu se pracuje s KM doménami, které jsou definovány pomocí Key Management Centra a pomocí ETCS entit. Každá entita může patřit právě do jedné domény, která se pro ni nazývá domácí. Zařízení Key Management Centre obsluhuje právě jednu KM doménu. Je například možné nastavit systém takový, že jedna doména odpovídá právě jednomu státu. [17, 12]

Mimo práce s klíči je KMC rovněž zodpovědné za správu ETCS entit v rámci KMS. Zmíněné entity je možné vytvářet, upravovat či mazat. Při vytvoření nové entity je nutné danou entitu přiřadit do domácí, nebo do cizí domény. Rovněž je také nutné zadat i unikátní ETCS ID, které ETCS entitu jednoznačně identifikuje. Přidání nové entity do KMC prakticky znamená vytvoření reference na danou entitu. Reference na entity cizí domény se využívá v mezi doménové komunikaci. KMC umožňuje také správu domén s tím, že vždy existuje jedna doména domácí a mnoho cizích domén, které slouží jako reference pro skutečné cizí domény. [11, 12]



Obrázek 2.2: Struktura šifry 3DES. Obrázek je dílem autora na základě [12].

2.6.3 Přenos klíčů mezi doménami

Pro zaručení interoperability je nutné zajistit, aby se mohla vozidla neomezeně pohybovat mezi mnoha státy. Z tohoto požadavku vyplývá skutečnost, že se vozidla musí pohybovat ve více KM doménách. Každé vozidlo ale může náležet pouze do jedné KM domény a tedy existuje pouze jedno KMC, kterému je umožněno vytvářet instalační a další příkazy pro danou KMAC entitu. Z tohoto důvodu je nutné zajistit komunikaci mezi KM doménami a tedy mezi KMC zařízeními. Tato komunikace může být opět prováděna on-line nebo off-line způsobem. [17, 11]

Pro lepší představu bude uveden příklad vysvětlující účel komunikace mezi doménami, spolu se způsobem provedení této výměny. Ve fiktivním a zjednodušeném příkladě bude předpokládáno, že existují dvě domény. Jedna z nich bude spravovat klíče pro území České republiky a druhá bude spravovat klíče na území Slovenské republiky. V české doméně se nachází RBC pro trať *Břeclav – Přerov*. Ve slovenské doméně se nachází hnací vozidlo s označením *363 999*, u kterého je požadavek na jízdu ve zmiňovaném úseku v České republice.

Z tohoto důvodu je nutné zajistit přítomnost stejného klíče v RBC traťového úseku a v OBU hnacího vozidla. Instalaci klíče do OBU vozidla může provádět pouze KMC Slovenské domény. Jedním z možných scénářů je tak vytvoření klíče prostřednictví české KMC, instalace tohoto klíče na RBC trať *Břeclav – Přerov* a odeslání tohoto klíče do KMC, které obsluhuje Slovenskou republiku. Zařízení slovenské KMC následně nainstaluje přijatý klíč na vozidlo *363 999*.

2.6.4 ETCS entita

ETCS entita (OBU, RBC/RIU) obsahuje zařízení, jehož účelem je přijímat příkazy od KMC, provádět přijaté příkazy a odesílat odpovědi na tyto příkazy. Toto zařízení nesmí provádět jiné operace, než ty nařízené od KMC. Kromě popsané komunikace také samozřejmě uchovává KMAC a KTRANS klíče. Kromě zmíněného také disponuje funkcionalitou pro odvození Session klíče (KSMAC). [11, 12]

2.7 Přenos ETCS klíčů

2.7.1 Off-line správa klíčů

V rámci off-line správy jsou příkazy a odpovědi na ně distribuovány pomocí přenosného úložiště (USB disk, pevný disk, CD, ...). Během tohoto přenosu musejí být dodržována opatření k zajištění bezpečnosti přenášených dat, protože v některých případech jsou přenášena citlivá data. [12, 17]

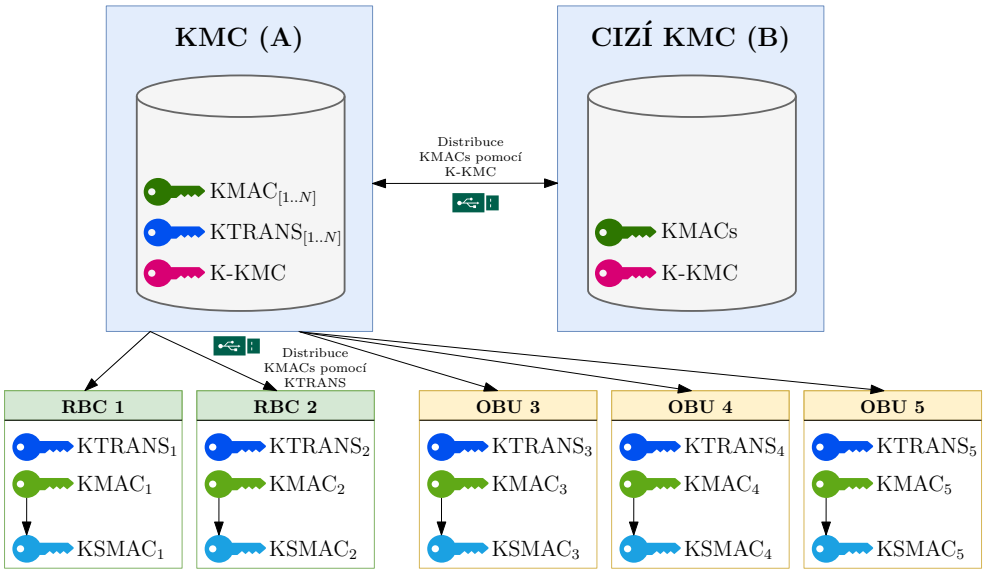
2.7.1.1 Instalace klíče na cílové zařízení

Po vytvoření entit a virtuálním přiřazení KMAC klíčů těmto entitám v rámci KMC je nutné daný klíč na fyzické zařízení skutečně přenést. Tento přenos se iniciuje v Key Management Centru, které vytvoří instalační příkaz ve formě binárního souboru. Tyto binární soubory se ukládají na přenosná úložiště do adresáře pro konkrétní zařízení. Cílové zařízení zná své označení a vyzvedne si pouze příkaz pro něj určený. Do stejného adresáře následně cílové zařízení uloží odpověď na prováděný příkaz. V souboru pro instalaci klíče se nachází samotný klíč (KMAC) v zašifrované podobě, výčet peerů, platnost klíče, MAC kód, ETCS ID příjemce a vydavatele klíče, číslo transakce a další. Z podoby instalační zprávy vyplývá, že je nutné vypočítat MAC autentizační kód zprávy a dále také to, že se klíč před přenosem šifruje. Z tohoto důvodu je nutná existence dalšího klíče, který se v rámci ETCS nazývá KTRANS. Tento klíč se skládá ze dvou Triple DES klíčů (KTRANS1 a KTRANS2), jedná se tak o 384 bitů dlouhý klíč. Klíč KTRANS1 je využit k zajištění integrity a autenticity přenášených dat. K tomuto účelu je využit algoritmus MAC, který je popsán v kapitole 2.5. Druhá část KTRANS klíče, tedy KTRANS2, se využívá k zašifrování přenášeného KMAC klíče. Jako šifrovací algoritmus je využit Triple

DES, který šifruje KMAC klíč v provozním režimu ECB. Instalace klíče není jedninou operací systému pro správu klíčů. Mezi podporované operace patří odstranění klíče, změna seznamu peerů a další. Platí, že všechny operace vždy iniciuje KMC. [12]

Zmíněný KTRANS klíč je opět generován pomocí KMC a je přiřazován konkrétní entitě (vždy se definuje pro vztah KMC – KMAC entita). Po vygenerování KTRANS klíče je nutné tento klíč přenést na fyzické zařízení, kterému bylo přiřazeno. Toto je opět provedeno instalačním příkazem ve formě binárního souboru, ale s tím rozdílem, že přenášený klíč již není zašifrován a ani u něj neprobíhá kontrola autenticity a integrity. Zabezpečení přenosu tohoto klíče se tak musí provést jiným opatřením. [12]

Na základě jakéhokoliv instalačního příkazu, který obsahuje v případě potřeby i data, vykoná cílová entita (OBU, RBC nebo RIU) požadovanou operaci. Podle výsledku této operace je entitou vygenerována odpověď, která musí být doručena zpět Key Management Centru. KMC si na základě této odpovědi zaznamená výsledek operace do své databáze. Odpověď je s příkazem spárována pomocí čísla transakce. [12]



Obrázek 2.3: Architektura off-line KMS z pohledu KMC „A“. Obrázek je dílem autora na základě [12].

2.7.1.2 Off-line komunikace mezi doménami

Přenos klíčů mezi doménami v off-line režimu probíhá opět za pomoci binárních souborů. Jedná se vlastně o stejný princip jako při přenosu klíče z KMC na KMAC entitu. Mezi doménová komunikace je také zabezpečena za účelem zajištění autenticity a integrity pomocí MAC kódu. Pro utajení přenášeného

klíče se používá symetrická šifra Triple DES. Zabezpečení přenášených dat je totožné jako v případě přenosu klíče mezi KMAC entitou a KMC (viz 2.7.1.1). Klíč používaný pro komunikaci mezi doménami se nenazývá KTRANS, ale nazývá se K-KMAC. Skládá se opět ze dvou částí o velikosti 192 bitů, které jsou pojmenovány K-KMAC1 a K-KMAC2. [17]

Zmíněná mezi doménová komunikace zajistí přenos KMAC klíče z jedné KM domény do druhé KM domény. Přenesený klíč je následně nutné nainstalovat do traťové, respektive mobilní části zabezpečovacího zařízení podle stejného principu, který je popsán v kapitole 2.7.1.1. [17]

2.7.2 On-line správa klíčů

V rámci systému on-line správy jsou KMAC klíče přenášeny pomocí počítačové sítě, do které tak musí být připojena všechna zařízení podporující daný způsob výměny klíčů. Pro komunikaci v rámci této sítě se využívá TCP/IP protokol. Tento protokol sám o sobě nezajišťuje důvěrnost, integritu (úmyslná změna) a ani autenticity komunikace, proto je nutné komunikaci zabezpečit pomocí TLS protokolu. [11]

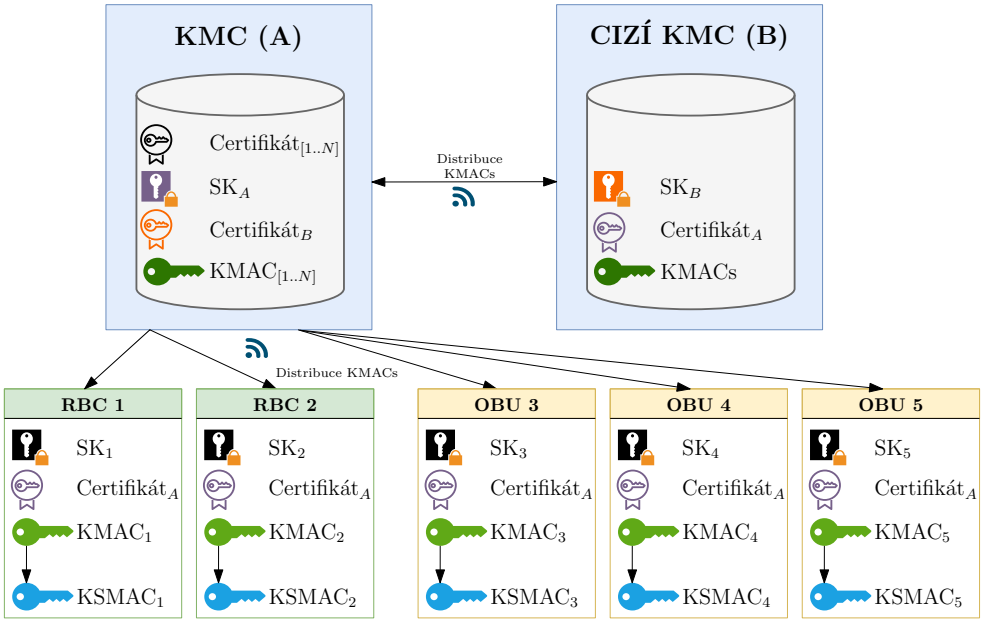
Pro ověření účastníků komunikace se využívají digitální certifikáty infrastruktury veřejného klíče (PKI). Další možností je využití sdíleného Preshared klíče (PSK). Každá KMS entita (KMC nebo ETCS entita) musí podporovat TLS-PKI. Jako alternativu k TLS-PKI je možné využít TLS-PSK, ale pouze pro komunikaci uvnitř domény. Pro mezi doménovou komunikaci je nutné využít TLS-PKI. Při použití PSK se musí na komunikující zařízení přenést sdílený tajný klíč a v případě PKI se na tato zařízení musí přenést certifikát kořenové certifikační autority. Tento přenos není součástí on-line správy klíčů a musí se provést v rámci jiného procesu. Z důvodu odlišného způsobu zabezpečení integrity a autenticity není v rámci on-line správy klíčů potřeba pracovat s KTRANS a K-KMAC klíči. [11]

2.7.2.1 Přenos zpráv

Zprávy přenášené mezi KMS entitami v on-line režimu lze rozdělit do tří kategorií. Jedná se o příkazy, dotazy a oznámení. Příkazy vyžadují změnu v databázi klíčů na straně příjemce. Dotazy vyžadují pouze odpověď od příjemce, ale neprobíhá úprava databáze klíčů na jeho straně. Mezi oznámení patří odpovědi na příkazy a dotazy, ale jsou to také zprávy týkající se TLS a jiné. [11]

Před přenosem jakékoliv zprávy je nutné vytvořit spojení mezi odesílatelem a příjemce dané zprávy. Platí, že za zřízení spojení mezi Key Management Centrem a traťovou částí ETCS je zodpovědné KMC. Za zřízení spojení mezi KMC a mobilní částí ETCS (OBU) je naopak zodpovědná mobilní část. Je to z toho důvodu, že OBU nemusí být vždy dostupné. OBU tak kontaktuje KMC vždy po zapnutí, pokud uběhla určitá doba od posledního kontaktu, pokud

si to obsluha vyžádá nebo pokud OBU zjistí neplatný či poškozený klíč. Po navázání spojení si pošlou obě komunikující strany zprávu, která indikuje zahájení komunikace. Během on-line komunikace může nastat mnoho chyb, které by v rámci off-line komunikace nastat nemohly. Jedná se například o přerušení spojení nebo o ztrátu některé ze zpráv. Dále může dojít k neočekávanému uvolnění spojení hned po odeslání příkazu na instalaci klíče v situaci, kdy ještě cílové zařízení nestihlo odpovědět. V takovémto případě nemá KMC informace o tom, zdali již došlo k zápisu klíče do databáze a nebo ne. Mohlo by tak dojít k nekonzistenci informací mezi KMC a ETCS entitou. Pro zajištění konzistence se definuje dotaz *INQ_REQUEST_KEY_DB_CHECKSUM*, který odesílá KMC svým entitám. Zařízení, které obdrží tento dotaz, vypočte kontrolní součet nad informacemi o klíčích bez samotných klíčů pomocí hašovací funkce MD4. KMC si tak může po obdržení odpovědi porovnat otisk svých dat s otiskem dat přítomných na entitě a na základě tohoto porovnání může případně provést novou instalaci klíče. [11]



SK_X označuje soukromý klíč vázaný k $Certifikátu_X$
$Certifikát_X$ obsahuje veřejný klíč entity X

Obrázek 2.4: Architektura on-line KMS z pohledu KMC „A“. Obrázek je dílem autora na základě [11].

2.7.2.2 Rozhraní

Zařízení podporující on-line správu kryptografických klíčů musí podporovat několik různých rozhraní pro komunikaci s okolím. Za prvé se jedná o **TLS rozhraní**, které umožňuje zřídit TLS spojení pro bezpečnou výměnu přenáše-

ných informací. Toho se využívá ke komunikaci mezi ETCS entitami a KMC, případně mezi KMC zařízeními navzájem. Přenášejí se přes něj hlavně příkazy pracující s klíči. Dále existuje **rozhraní pro doručování certifikátů**, které se využívá ke komunikaci s certifikační autoritou. Toto rozhraní umožňuje získání nebo obnovení certifikátu pomocí Certificate Management Protokolu. Ke kontrole platnosti certifikátů pomocí Online Certificate Status Protokolu je využito rozhraní **certificate status management**, které se definuje mezi KMS entitou a zařízením OCSP Responder (většinou certifikační autorita). Pro zajištění kompatibility a zvýšení spolehlivosti systému je také možné implementovat rozhraní pro off-line přenos klíče, které je popsáno v kapitole 2.7.1. [11]

2.7.3 Zhodnocení možností správy klíčů

Jak již bylo uvedeno v předchozích kapitolách, existují dvě možnosti správy klíčů podle typu přenosu klíčů mezi zařízeními. Jedná se o off-line a on-line přenos. Oba způsoby je možné spolu vzájemně kombinovat. Na základě zkušenosti z praxe může autor této práce konstatovat, že v současné době (rok 2022) je v České republice využíván pouze off-line způsob správy klíčů, i když je výrazně pomalejší a náročnější na obsluhu. Složitější způsob přenosu klíčů patří k hlavním nevýhodám off-line správy klíčů, protože například při přenosu klíče z KMC do vlaku je nutné tento klíč fyzicky zanést na dané vozidlo, což při velkém množství vozidel samotný proces výrazně zpomaluje. Při on-line komunikaci je tento způsob práce s klíči výrazně rychlejší a pohodlnější. Jednou z nevýhod on-line způsobu správy klíčů je nutnost mít ETCS entity a KMC připojené do počítačové sítě. Do jisté míry jsou tak tato zařízení náchylnější k různým útokům.

2.8 Požadavky na aplikaci

Jak již bylo zmíněno, cílem této práce je navrhnout a implementovat systém pro on-line správu klíčů zabezpečovacího zařízení ETCS. Na základě analýzy v předchozích kapitolách byly identifikovány jednotlivé funkční i nefunkční požadavky, které jsou kladeny na výsledné řešení. Tyto požadavky jsou zformulovány v kapitolách 2.8.1 a 2.8.2. Základ pro výše uvedenou analýzu tvoří standardy – Set of specifications 3 (ETCS B3 R2 GSM-R B1), které zaštiťuje Evropská železniční agentura (ERA). Hlavním dokumentem definující požadavky pro zpracovávané téma je On-line Key Management FFFIS (SUBSET-137) [11]. Kapitola analýza obsahuje i části pojednávající o off-line správě klíčů a ETCS samotném. Tyto kapitoly nesloužily jako podklad k formulaci požadavků, ale slouží k zasvěcení čtenáře do problému. Požadavky uvedené v této kapitole budou zohledněny při pozdějším návrhu řešení.

2.8.1 Funkční požadavky

V této kapitole jsou uvedeny a popsány všechny požadavky, které stanovují funkcionalitu výsledného systému.

F1 – Správa klíčů

V rámci Key Management Systému musí existovat Key Management Centre, které umožní generování respektive ruční zadávání nových KMAC klíčů (Triple DES). Rovněž musí umožnit ke klíčům přidávat jejich identifikující údaje, časovou platnost a seznam peerů. Kromě generování klíčů zodpovídá KMC za jejich odebírání, úpravu a virtuální přiřazení entitám.

F2 – Správa entit

KMC musí umožňovat správu ETCS entit své KM domény. Entity je nutné vytvářet, modifikovat a také mazat. Informace evidované o každé entitě odpovídají požadavkům definovaným v příslušných dokumentech. Kromě entit v domácí doméně bude KMC umožňovat vytvoření reference na entity cizí KM domény. Definice entit je nutná pro práci s KMAC klíči.

F3 – Správa domén

Další z podporovaných funkcionalit KMC je správa ETCS domén, které musí být možné přidat, upravit a odebrat. Key Management Centre musí rozlišovat mezi doménou domácí a doménami cizími. Domácí doména je pro každé KMC pouze jedna. Definice domén je nutná pro práci s ETCS entitami.

F4 – Uložení klíčů

KMAC klíče musejí být uloženy na zařízeních, která je budou aktivně využívat a kterým jsou přiřazena. Jedná se tedy o ETCS entity (RBC, RIU nebo OBU). Samotné uložení klíčů musí zajišťovat modul k tomu určený. Funkcí zmíněného modulu není pouze uložení klíčů, ale také vydávání session klíče (KSMAC), který je odvozen z uloženého KMAC klíče. Podpora pro vydávání KSMAC klíčů nemusí být součástí řešení, ale musí existovat způsob pro jednoduchou implementaci dané funkcionality. Uložený KMAC klíč nesmí modul pro uložení klíče nikdy opustit.

F5 – On-line distribuce klíčů

Vytvořené KMAC klíče je nutné přenést z KMC na KMAC entitu, která ho bude využívat pro činnost, která nesouvisí s touto prací. Samotný přenos musí být realizován on-line způsobem pomocí počítačové sítě. Tento typ přenosu musí být realizován pomocí aplikačního protokolu, který je definován

dokumentem ERTMS/ETCS SUBSET 137 [11]. Musejí být implementovány následující funkce:

- **Přidání klíčů** – spouští ji KMC za účelem instalace jednoho nebo více klíčů (KMAC) na cílovou KMAC entitu. Tato funkce se taktéž využívá pro odeslání jednoho nebo více klíčů na cizí KMC za účelem instalace těchto klíčů na zařízení cizí KM domény.
- **Odebrání klíčů** – spouští ji KMC pro odstranění jednoho nebo více instalovaných klíčů z KMAC entity. Tato funkce je rovněž využívá v rámci mezi doménové komunikace.
- **Odebrání všech klíčů** – funkce odebírá **všechny** nainstalované klíče z RBC, RIU nebo OBU. Tento proces inicializuje KMC.
- **Aktualizace časové platnosti klíčů** – využívána KMC k aktualizaci časové platnosti KMAC klíčů ve své nebo cizí doméně.
- **Aktualizace entit vázaných ke klíčům** – upravuje seznam peerů, které jsou vázány ke KMAC klíčům na entitách. Tuto funkci je rovněž možné využít pro komunikaci mezi dvěma doménami.
- **Kontrola databáze klíčů** – využíváno ke kontrole stavu databáze na KMAC entitě. Samotná kontrola je zajištěna pomocí výpočtu kontrolní sumy. Tato kontrola je vyžadována od KMC.
- **Nahlášení změny stavu klíče** – Jedná se o funkci využívanou KMC k informování cizí KMC (tak která daný klíč vydala), že došlo ke změně stavu klíče v doméně odesílatele. Může se jednat o odpověď na příkaz nařízený cizí doménou.
- **Žádost o provedení operace s klíčem** – tuto funkci využívá KMC k tomu, aby si od KMC spravující cizí KM domény vyžádalo provedení operace pro zařízené své domény. Tato funkce se využije například v případě, kdy je nutné provozovat nové vozidlo v cizí doméně.

Komunikace mezi jednotlivými zařízeními musí být provedena způsobem, aby byla zajištěna integrita, důvěrnost a autenticita přenášených dat. K zajištění kompatibility je pro tyto účely nutné využívat TLS protokol.

F6 – Aktualizace certifikátů

Z důvodu využívání certifikátů pro distribuci veřejných klíčů, musí všechna zařízení podporující on-line správu klíčů podporovat komunikaci s certifikačními autoritami. Tato komunikace se využije při nutnosti generování, aktualizace nebo zneplatnění certifikátů pomocí Certificate Management Protokolu.

F7 – Ověření platnosti certifikátů

Zařízení podporující on-line správu kryptografických klíčů musí mít implementované rozhraní pro on-line kontrolu stavu certifikátů. Tato kontrola musí být prováděna pomocí *Online Certificate Status Protokolu*, který je popsán v dokumentech *RFC-6277* [18] a *RFC-2560* [19]. Dané zařízení by tak mělo mít možnost zkontrolovat si, zdali nedošlo ke kompromitaci certifikátu protistrany.

2.8.2 Nefunkční požadavky

Jedná se o požadavky, které se nevztahují přímo k funkcionalitě výsledného řešení, ale definují omezení na provedení systému. Bezpečnostní omezení budou důležitou součástí této kapitoly.

N1 – Plnění požadavků ERTMS

Pro zajištění kompatibility musí celý systém pro správu klíčů (Key Management System) plnit požadavky definované Evropskou železniční agenturou (ERA). Jedná se o standardy – Set of specifications 3 (ETCS B3 R2 GSM-R B1). Hlavním dokumentem definující požadavky na on-line správu klíčů je On-line Key Management FFFIS (SUBSET-137) [11].

N2 – Kryptograficky bezpečný generátor náhodných čísel

Používaný generátor náhodných čísel musí být kryptograficky bezpečný. Generátor pseudonáhodných čísel může být použit, ale pouze v kryptograficky bezpečné variantě. Požadavky na tento generátor jsou kladeny vzhledem k tomu, že vygenerovaná náhodná čísla budou použita mimo jiné pro vytváření nových kryptografických klíčů.

N3 – Přenositelnost kódu

Aplikace Key Management Centre má být uživatelům poskytována ve formě webové aplikace. S tímto požadavkem souvisí nutnost instalace aplikace na webový server, který bude danou aplikaci poskytovat. Z tohoto důvodu by měla aplikace podporovat běžně používané operační systémy.

N4 – Integrita dat

Musí být zajištěna integrita dat během jejich přenášení mezi KMS zařízeními, tedy mezi dvěma KMC a rovněž mezi KMC a KMAC entitou. Key Management Centre musí mít rovněž přehled o stavu klíčů na spravovaných entitách.

2.8.3 Zhodnocení požadavků

V předchozích kapitolách byly definovány funkční i nefunkční požadavky, které jsou kladeny na výsledné řešení. Ze stanovených funkčních požadavků jsou nejdůležitější požadavky **F5, F6** a **F7**, protože se plní hlavní cíl této práce. Ostatní požadavky slouží jako podpůrné pro hlavní funkcionalitu. Požadavek **N1** patří mezi nejdůležitější nefunkční požadavky.

2.9 Rešerše použitelných technologií

Tato kapitola obsahuje popis technologií, které bude možné nebo nutné využít pro realizaci finálního řešení. Detailní použití konkrétní technologie bude uvedeno v kapitole zabývající se návrhem. Vybrané technologie vycházejí z definic funkčních a nefunkčních požadavků.

V této kapitole zmiňované koncové entity a klíče nijak nesouvisí s entitami (OBU, RBC/RIU, KMC) a KMAC klíči zabezpečovacího zařízení ETCS.

2.9.1 Transmission Control Protocol – TCP

Jedná se o spojový protokol transportní vrstvy, který se nejčastěji využívá nad IP protokolem pro zajištění spolehlivého přenosu paketů. TCP protokol obsahuje mechanismy pro řešení situací, kdy se některý z paketů ztratí, nebo naopak zduplikuje. Prohození paketů je tímto protokolem taktéž řešeno, proto může být vyšším vrstvám garantované pořadí doručených dat. Zmíněné vyšší vrstvy se taktéž mohou spolehnou na bezchybné přenesení svých dat. Je nutné zmínit, že se jedná o ochranu před neúmyslnou změnou dat. Transmission Control Protocol umožňuje řídit objem odesílaných dat a rovněž také dokáže eliminovat zahlcení linky. Prvního je dosaženo pomocí klouzavého okna a druhého pomocí metody Congestion Window. Šifrování přenášených dat TCP protokol nezajišťuje a musí si ho tak zajistit vyšší vrstvy. Adresování v rámci TCP komunikace je provedeno pomocí síťových portů. Port je číslo z rozsahu 0 – 65535 určující aplikaci, ke které mají být doručená data předána. [20, 21]

Data jsou přenášena pomocí TCP segmentů. Každý z těchto segmentů se skládá z hlavičky a samotných dat. Součásti hlavičky jsou čísla zdrojových a cílových portů, sekvenční číslo, potvrzovací číslo, velikost hlavičky, bity pro řízení příznaků, velikost okna, kontrolní součet a další volitelné položky hlavičky. [20]

Zahájení komunikace probíhá pomocí třícestného ověření tak, že počítač zahajující komunikaci odesílá paket s nastaveným příznakem SYN, který se nachází v hlavičce. Poté co druhá strana komunikace tuto zprávu přijme, odpovídá paketem s nastavenými příznaky SYN a ACK. V okamžiku, kdy první počítač přijme od druhého tuto odpověď, tak odesílá druhému počítači paket s nastaveným příznakem ACK. Po dokončení této komunikace se má za to, že TCP komunikace byla ustanovena. Pakety v této ustanovující komunikaci

většinou nenesou žádná data. Ukončení komunikace probíhá stejným způsobem jako při jejím zahajování. Opět se jedná o třícestné ověření, ale místo paketů s nastaveným příznakem SYN se posílají pakety s příznakem FIN. Jak již bylo zmíněno, tak aplikační data jsou přenášena pomocí TCP paketů obsahující sekvenční číslo odesílaného a sekvenčním číslem očekávaného paketu. Tato čísla řeší správné pořadí při doručování paketů a detekci duplicitních paketů. Příjemce odpovídá na doručené TCP pakety odpovědmi, které mají nastavený příznak ACK. Pokud se odesílatel nedočká potvrzovací odpovědi do určitého času, tak odesílá nepotvrzený paket znovu. [20]

Pro řízení objemu posílaných dat se využívá metoda klouzavého okénka, pomocí kterého si **příjemce** reguluje množství dat posílaných odesílatelem. Délka okna definuje počet odeslaných paketů bez potvrzení. Tuto délku si stanovuje **příjemce** tak, aby nedošlo k jeho přetížení. Další možností řízení množství odesílaných dat je regulace objemu odesílaných dat přímo **odesílatelem** za účelem zabránění zahlcení datové linky. Velikost okna v rámci odesílání se nazývá Congestion Window Length (CWL). Tato hodnota určuje počet paketů, které je možné odeslat najednou, aby nedošlo k zahlcení linky. Na rozdíl od regulace toku dat řízené příjemcem, zde existuje mnoho různých algoritmů řešící tento problém. Jedná se například o Tahoe nebo Reno. [21]

2.9.2 Transport Layer Security - TLS

Transport Layer Security je kryptografický protokol, který zajišťuje ochranu dat posílaných pomocí počítačové sítě. Tento protokol je obvykle implementován nad protokolem TCP a jeho úkolem je zajistit integritu, autenticitu a důvěrnost dat vyšších vrstev komunikace (HTTP, IMAP, ...). Existuje také verze s označením Datagram Transport Layer Security (DTLS), která funguje nad protokolem UDP. [22]

Pro zajištění bezpečnosti je využita symetrická a asymetrická kryptografie. Kombinace těchto dvou mechanizmů zajišťuje odpovídající bezpečnost a rozumnou rychlost. Symetrická kryptografie se používá k šifrování a dešifrování přenášených aplikačních dat. Asymetrická kryptografie se používá ke zřizování symetrických klíčů relace během fáze *TLS handshake*. Klíče zřízené během této fáze se využijí následně pro šifrování aplikačních dat pomocí symetrického šifrovacího algoritmu. Pro zřízení společného klíče dovoluje TLS využít algoritmy jako je Diffie-Hellman nebo Diffie-Hellman s využitím eliptických křivek. Pro bezpečnou komunikaci v rámci počítačové sítě je nutné zajistit, aby si účastníci komunikace mohli ověřit identitu protistrany, se kterou komunikují. V rámci TLS protokolu se identita ověřuje v rámci *TLS handshake* pomocí digitálních certifikátů infrastruktury veřejného klíče. Zmíněné certifikáty vydává důvěryhodná certifikační autorita. Tento způsob ověřování se nazývá TLS-PKI, ale existuje i možnost TLS-PSK s využitím předem sdíleného klíče. [22, 11]

Podle [23] se algoritmy používané v rámci TLS protokolu nazývají šifrovací sada. Řetězec *TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256*

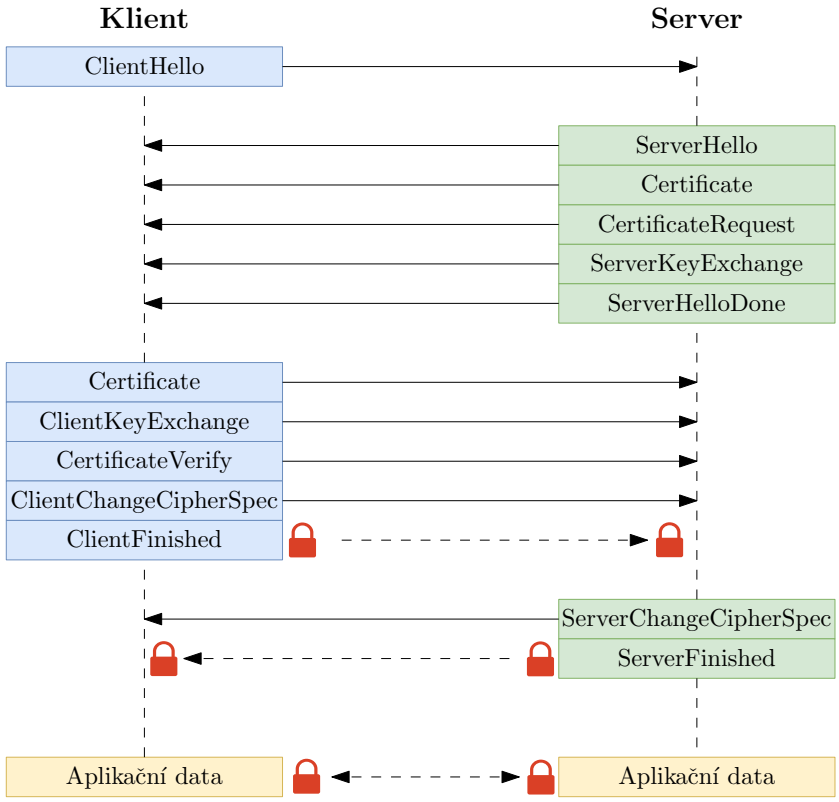
označuje jednu z podporovaných šifrovacích sad TLS ve verzi 1.2. Interpretace tohoto řetězce je následující:

- Ustanovení klíčů – Protokol Diffie-Hellman s využitím eliptických křivek, podpis zpráv je realizován pomocí šifry RSA.
- Šifrování dat – Využití šifry AES s klíčem o velikosti 128 bitů v módu GCM.
- MAC – SHA256

Podle [24] se před započetím každé TLS komunikace provádí již zmíněný *TLS handshake*, jehož cílem je dohoda komunikujících stran na parametrech komunikace. Tato část komunikace je tvořena několika zprávami, které budou nyní popsány na protokolu ve verzi 1.2. Novější verze TLS 1.3 přináší zrychlení počáteční fáze. Na obrázku 2.5 se nachází diagram, který obsahuje zprávy přenášené během *TLS handshake*.

- **ClientHello** – Klient touto zprávou informuje server, že chce započít zabezpečenou komunikaci. Současně touto zprávou oznamuje nejvyšší podporovanou verzi TLS protokolu spolu s podporovanou šifrovací sadou.
- **ServerHello** – Server vybere nejvyšší verzi TLS protokolu podporovanou oběma stranami komunikace a rovněž vybere nejvýhodnější šifrovací sadu, která je podporována oběma zařízeními.
- **Certificate** – Server zašle klientovi svůj certifikát, případně řetězec certifikátů. Tato zpráva je nepovinná.
- **CertificateRequest** – V případě nutnosti ověření identity klienta, posílá serveru žádost o jeho certifikát. Ověření klienta se ale používá pouze velmi zřídka.
- **ServerKeyExchange** – Nepovinná zpráva obsahující informace používané v rámci protokolu, který zajišťuje ustanovení společného klíče (např. Diffie-Hellman).
- **ServerHelloDone** – Server oznamuje ukončení počáteční fáze komunikace.
- **Certificate** – Klient odesílá svůj certifikát, pokud byl serverem požadován.
- **ClientKeyExchange** – Klient odesílá informace, které slouží k ustanovení společného klíče pro symetrické šifrování. V případě využití RSA obsahuje tato zpráva některé informace podepsané veřejným klíčem serveru pro jeho ověření.

- **CertificateVerify** – Zpráva sloužící pro ověření klienta, pokud je jeho ověření požadováno.
- **ClientChangeCipherSpec** – Poslední zpráva od klienta, která je nešifrovaná.
- **ClientFinished** – Jedná se o první zašifrovanou zprávu od klienta, která obsahuje hash všech předchozích zpráv (TLS handshake) + řetězec „client finished“.
- **ServerChangeCipherSpec** – Obdobná zpráva jako v případě ClientChangeCipherSpec
- **ServerFinished** – Obdobná zpráva jako v případě ClientFinished



Obrázek 2.5: TLS 1.2 handshake. Obrázek je dílem autora na základě [24].

TLS protokol vznikl z protokolu SSL (Secure Socket Layers), který byl vyvinut společností Netscape Communications Corporation v roce 1994 a jeho účelem bylo zabezpečení webových relací. SSL 1.0 byla první verzí tohoto protokolu, která nebyla nikdy publikována z důvodu vysokého počtu chyb. Následovala již verze 2.0, která byla publikována roku 1995. Tato verze obsahovala

rovněž velké množství chyb, a proto byla rychle nahrazena verzí 3.0, u které došlo ke kompletnímu přepracování. Tato verze byla publikována roku 1996 a jednalo se o poslední verzi protokolu SSL. Následující verze jsou již označovány jako TLS, jeho první verze 1.0 byla publikovaná v roce 1999. Jednalo se o aktualizaci poslední verze protokolu SSL. Následovaly verze 1.1 a 1.2. Druhá z nich byla publikována roku 2008 a přinášela podporu šifry AES a opouštěla hašovací funkci MD5. TLS 1.3 je zatím poslední verzí, která přináší podporu nových šifer a odstraňuje slabé algoritmy a nepoužívané funkce. [23]

2.9.3 Public key infrastructure – PKI

Public key infrastructure slouží pro distribuci a správu veřejných klíčů asymetrické kryptografie. Tento typ kryptografie je možné využít k šifrování dat, ale také k ověřování identity. Pro využití asymetrické kryptografie je nutné, aby každá z komunikujících stran znala veřejný klíč své protistrany. Soukromé klíče drží všechny strany komunikace v tajnosti. V souvislosti s používáním veřejných klíčů asymetrické kryptografie hrozí možnost podvržení těchto klíčů útočníkem. Pokud by útočník podvrhl veřejný klíč některé z oprávněných stran komunikace, tak by se za danou stranu mohl vydávat a taktéž by mohl dešifrovat a upravovat zprávy patřící oběti. Podvržení veřejného klíče znamená, že útočník nahradí veřejný klíč oběti svým veřejným klíčem, ke kterému vlastní odpovídající soukromý klíč. Jiní uživatelé nemají bez využití technik bezpečné distribuce veřejných klíčů možnost ověřit, že používají nesprávný veřejný klíč. Celá infrastruktura veřejného klíče je popsána normami, která vycházejí z norem ITU-T X.500. Hlavním účelem těchto norem je zajištění kompatibility. [25]

Podle [25] existují pro zajištění bezpečné distribuce veřejných klíčů následující techniky:

- **Public announcement (zveřejnění veřejných klíčů)** – distribuce veřejného klíče v rámci nějaké komunity (vystavení na webu, příloha k emailu, ...). Jedná se o jednoduchý způsob s nízkou bezpečností.
- **Public available directory (veřejně dostupný adresář)** – veřejné klíče jsou uloženy ve veřejném adresáři, který má svého správce. Záznam se skládá z položek jako je {jméno, veřejný_klíč}. Veřejně dostupný adresář je bezpečnější než pouhé zveřejnění klíčů, ale stále hrozí možnost modifikace adresáře útočníkem.
- **Public-key authority (autorita pro veřejné klíče)** – způsob podobný adresáři, ale zvýšení bezpečnosti zpřísněním kontroly nad vkládanými veřejnými klíči. Vyžaduje, aby uživatelé znali veřejný klíč adresáře. K tomuto veřejnému klíči existuje soukromý klíč, který vlastní autorita pro správu klíčů.

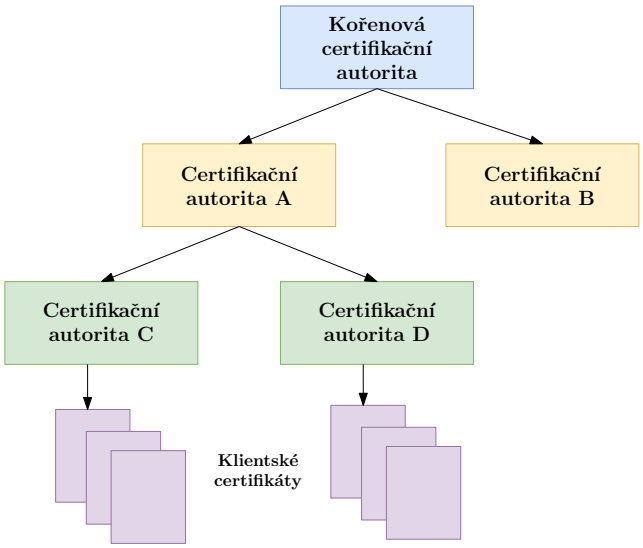
- **Public-key certification (certifikace veřejných klíčů)** – založeno na principu existence certifikátů, které umožňují kontrolu identity vlastníků veřejných klíčů bez nutnosti komunikace s jakoukoliv autoritou. Tato komunikace tvořila úzké hrdlo v předchozích případech, ale při použití této metody je nutná pouze během vytváření nových certifikátů. Certifikát je struktura obsahující veřejný klíč vlastníka, údaje identifikující vlastníka, dobu jeho platnosti, a také údaje o certifikační autoritě (CA), která daný certifikát vydala. Celý tento obsah je podepsán soukromým klíčem CA a může jej ověřit kdokoli, kdo disponuje veřejným klíčem této autority. Důvěra v certifikát je tedy založena na důvěře v CA, která daný certifikát vydala. Pokud chce klient důvěřovat certifikátům vydaným konkrétní CA, tak musí mít uložený certifikát konkrétní autority. Princip této techniky je založen na tom, že odesílatel i příjemce nejprve požádají CA o vytvoření certifikátů, poté si tyto certifikáty vymění a na základě toho tak mohou zahájit komunikaci. Pokud by některý z účastníků komunikace již certifikát vlastnil, využije jej a nemusí tak s CA komunikovat. Formát vydaných certifikátů je stanoven doporučením ITU-T X.509.

  Pro lepší efektivitu celého systému se předpokládá vzájemné propojení mezi certifikačními autoritami. Toto propojení lze realizovat například formou certifikačního stromu, který reprezentuje kořenová certifikační autorita. Samotné propojení se realizuje pomocí certifikátů CA, které vytvářejí řetězec propojený podpisy. Uživatelé tak nemusejí mít uložené velké množství certifikátů různých certifikačních autorit, kterým důvěřují. Těmto uživatelům stačí důvěřovat pouze kořenové certifikační autoritě a tím tedy vyjadřují důvěru všem jejím podřazeným autoritám. Řetězec CA spolu s kořenovou certifikační autoritou je naznačen na obrázku 2.6. [25]

2.9.4 Certificate Management Protocol – CMP

Certificate Management Protocol se využívá pro správu certifikátů infrastruktury veřejného klíče. Využívá se tedy ke komunikaci mezi koncovými entitami (těm je vydáván certifikát) a certifikačními respektive registračními autoritami. Tento protokol je rovněž možné využít k zajištění komunikaci mezi dvěma certifikačními autoritami. CMP klienti mohou využít CMP protokol za účelem vyžádání, odvolání nebo obnovení certifikátů. Certifikační autority mohou CMP protokol využít ke komunikaci související s křížovými certifikáty. Požadavky jsou přenášeny ve formátu CRMF (Certificate Request Message Format). Samotné zprávy je možné přenášet pomocí protokolu HTTP(S). [26, 27]

Pokud se koncová entita připojuje pomocí popisovaného protokolu k certifikační autoritě poprvé, tak musí projít inicializační (registrační) fází. Jako



Obrázek 2.6: Řetězec certifikátů. Obrázek je dílem autora na základě [11].

výsledek inicializačního procesu by mělo být vydání a odeslání prvního certifikátu koncové entitě. V rámci inicializační fáze je taktéž možné certifikát vydaný zveřejnit ve veřejném úložišti. V rámci CMP protokolu je nutné zajistit autenticitu zpráv posílaných mezi koncovou entitou a certifikační autoritou. Existuje několik schémat zajišťující autenticitu. Ověření CA/RA je vždy vyžadováno a lze jej zajistit instalací certifikátu autority na koncovou entitu. Pro prokázání identity koncové entity je možné využít proces využívající tajnou hodnotu (počáteční ověřovací klíč) a referenční hodnotu (sloužící k identifikaci tajného klíče). Přenos těchto sdílených informací musí být realizován jiným způsobem než pomocí CMP protokolu. Takto nastavený počáteční ověřovací klíč lze poté použít k ochraně příslušných zpráv protokolu. Existuje taktéž několik schémat ke kontrole vazby mezi koncovou entitou a párem klíčů. Koncová entita může například podepsat nějaké náhodné číslo svým soukromým klíčem a CA/RA může následně daný podpis ověřit veřejným klíčem daného koncového subjektu. Je možné také provést kontrolu například dešifrováním certifikátu. Způsob ověření se vybírá na základě toho, jakým způsobem budou certifikované klíče využívány. [26]

Podle [26] se všechny zprávy používané za účel správy certifikátů skládají z položek uvedených ve výpisu 2.1. Struktura *header* obsahuje informace používané pro směrování a identifikaci transakce. Část *body* uchovává data specifická pro každý typ zprávy. *Protection* obsahuje například informace pro kontrolu integrity. Pole *extraCerts* může obsahovat certifikáty, které mohou být užitečné pro příjemce. Toto pole může být využito, pokud certifikát CA/RA, která vydala koncovému subjektu certifikát, není kořenovou certifikační autoritou pro tento subjekt.

Pokud si koncová entita přeje vygenerovat nový pár klíčů, může tak uči-

```
PKIMessage ::= SEQUENCE {
    header           PKIHeader,
    body             PKIBody,
    protection   [0] PKIProtection OPTIONAL,
    extraCerts   [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate
                     OPTIONAL
}
PKIMessages ::= SEQUENCE SIZE (1..MAX) OF PKIMessage
```

Výpis kódu 2.1: Struktura PKI zprávy

nit sama nebo může požádat certifikační respektive registrační autoritu [26]. Podpora pro Certificate Management Protocol je implementována například v knihovně OpenSSL. [28]

2.9.5 Online Certificate Status Protocol – OCSP

Online Certificate Status Protocol se opět váže k certifikátům infrastruktury veřejného klíče. V případě kompromitace soukromého klíče je nebezpečné dále využívat certifikát příslušející k takto kompromitovanému klíči, proto musejí existovat mechanismy pro zneplatnění certifikátů. Každý certifikát má v rámci PKI stanovenou určitou platnost, která definuje časový rámec jeho použitelnosti. K nutnosti zneplatnění certifikátu může dojít ještě před koncem jeho časové platnosti. Z tohoto důvodu existují další techniky revokace. Jedná se o Certificate Revocation List (CRL), což je seznam zneplatněných certifikátů. Tento seznam je zveřejňován certifikačními autoritami. V současné době existuje velké množství záznamů v CRL seznamech, a tudíž je nutné stahovat velké množství dat. Z tohoto důvodu začínají být tyto seznamy nahrazovány protokolem OCSP. [29]

Online Certificate Status Protocol je tedy on-line metoda sloužící k ověření stavu certifikátů. Jeho hlavní využití spočívá v kontrole toho, zda nebyl některý z certifikátů zneplatněn. Komunikace probíhá přes HTTP protokol. Servery nabízejícím OCSP službu se nazývají OCSP Respondery a jejich adresy jsou běžně součástí certifikátů. Službu OCSP Responderu většinou provozují certifikační autority. S využitím tohoto protokolu odpadá nutnost stahování velkého množství informací, které nebyly z většiny ani potřebné. Vyměňují se pouze informace ohledně certifikátů, které klienta zajímají. Na druhou stranu je nutná komunikace s OCSP Responderem, což může působit problémy v případě jeho nedostupnosti. V takovýchto případech může docházet k odmítání platných certifikátů. Zpráva odesílaná OCSP serverem se skládá z identifikátoru vázajícímu se k dotazovaného certifikátu, stavu certifikátu, platnosti odesílané zprávy a případně dalších volitelných částí. Stav certifikátu může

nabývat tří hodnot, kterými jsou *good*, *revoked* a *unknown*. Server může označovat jako neplatné všechny certifikáty vydané certifikační autoritou, u které došlo ke kompromitaci soukromého klíče. Zprávy odesílané OCSP Respondery jsou digitálně podepsané. Klíče používané k podpisu těchto zpráv mohou patřit certifikačním autoritám, které vydaly právě ověřovaný certifikát. Dále mohou tyto klíče patřit autorizovaným Responderům nebo Responderům, kterým klienti důvěřují. Samotné zprávy jsou kódovány pomocí ASN.1. Součástí odpovědi může být taktéž odkaz na CRL seznam, kde se nachází zneplatněný certifikát, který se vztahuje k dané odpovědi. [30]

Tento protokol může být zranitelný vůči replay útokům. Útočník zachytí od OCSP Responderu podepsanou odpověď s příznakem *good*. Tuto odpověď si ponechá a odešle ji klientovi v době, kdy již tato odpověď obsahuje neplatný stav certifikátu v důsledku změny jeho stavu. Pokud se toto stane v rámci platnosti zachycené odpovědi, tak klient dostává neplatné údaje. Pro zamezení těchto útoků umožňuje OCSP protokol vkládat do požadavků nonce, které umožní spárovat požadavek a odpověď. Server se nicméně může rozhodnout do své odpovědi nonce rozšíření neumístit, i když bylo součástí požadavku. [30]

2.10 Použitelné architektury aplikací

Na základě funkčních požadavků je zřejmé, že bude nutné vytvořit webovou aplikaci. Z tohoto důvodu je nutné se zabývat architekturou, která bude využita při návrhu a implementaci výsledné webové aplikace.

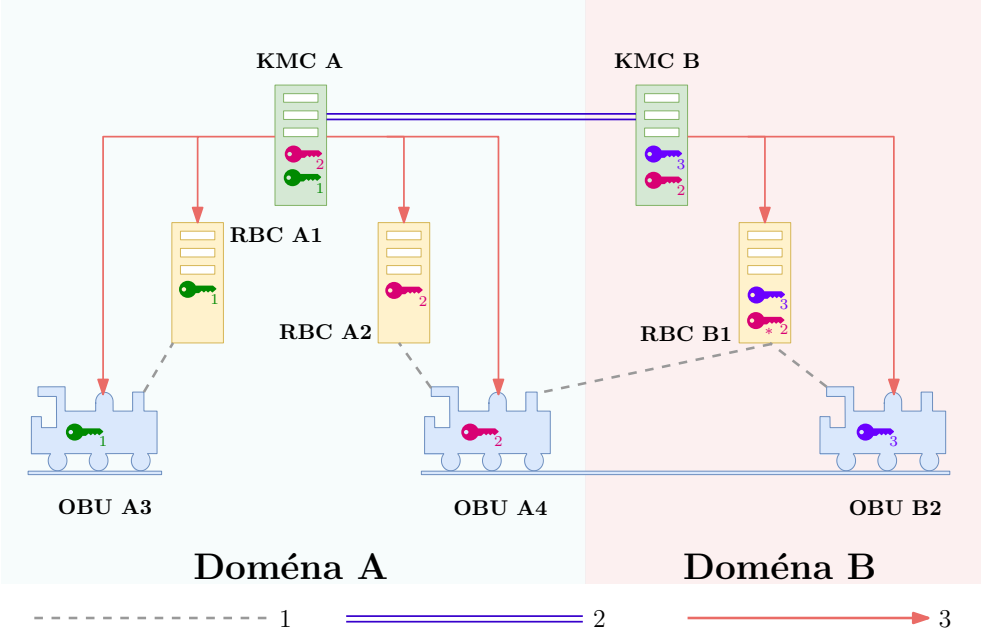
2.10.1 Třívrstvá architektura

Vícevrstvá logická architektura rozděluje celou aplikaci do vzájemně komunikující vrstev. Právě popisovaná třívrstvá architektura definuje tři vrstvy, kterými jsou (od nejvyšší po nejnižší): **prezentační**, **doménová (business) a datová**. Samotnou třívrstvou architekturu lze ještě rozdělit na striktní a relaxovanou podle vzájemné závislosti jednotlivých částí. Vždy ale platí, že pouze vyšší vrstva může být závislá na nižší, ale nikdy ne obráceně. Striktní architektura dovoluje závislosti pouze o jednu úroveň níže, zatímco relaxovaná architektura dovoluje závislosti přes libovolný počet úrovní níže. Úkolem prezentační vrstvy je zpracování uživatelských požadavků a zajištění výstupu pro uživatele. V případě webových aplikací je složená z HTML stránek nebo šablon. Doménová vrstva zajišťuje různé výpočty a validace. Poslední datová vrstva řeší persistenci dat. [31]

Pravě popisovanou architekturu je vhodné využít pro složitější aplikace. Její výhody spočívají v oddělení jednotlivých vrstev, což umožňuje snadnou výměnu jedné vrstvy za druhou a také jednodušší testovatelnost celé aplikace. Mezi další výhody popisované architektury patří znovupoužitelnost jednotlivých částí. [31]

2.11 Zhodnocení analýzy

V této kapitole byly zanalyzovány systémy pro on-line i off-line správu klíčů používaný v rámci vlakového zabezpečovacího systému ETCS. Z této analýzy vyplynulo, že systém pro on-line správu klíčů není v době psaní této práce rozšířen, proto bude v této práci navržen a implementován na základě definovaných funkčních a nefunkčních požadavků. Pro implementaci budou využity technologie, které byly popsány v rešerši použitelných technologií.



Obrázek 2.7: Vybraná komunikace v rámci systému ETCS. Spojení **1** označuje komunikaci zabezpečenou pomocí klíče KMAC prostřednictvím sítě GSM-R. V rámci této komunikace například oznamují vozidla svoji polohu traťové části nebo naopak dostává od traťové části povolení k jízdě. Jedná se o komunikaci **mimo** rámec této práce. Šipka **2** označuje distribuci klíčů mezi doménami on-line nebo off-line způsobem. A spojení **3** znázorňuje distribuci klíčů uvnitř domény opět on-line nebo off-line způsobem. Práce se zabývá pouze **on-line** způsobem přenosu klíčů. Na obrázku jsou znázorněny dvě domény, které jsou spravovány KMC „A“ a KMC „B“. Pod každou doménu spadá několik entit (RBC a OBU). Z obrázku plyne (podle přítomnosti klíčů na jednotlivých entitách), že OBU „A-3“ může využívat pouze úseky obsluhované RBC „A-1“. OBU „A-4“ může využívat tratě obsluhované RBC „A-2“ a „B-1“, což je z jeho pohledu úsek v cizí doméně. Klíč OBU „A-4“ (červená barva) se tak nachází kromě své domovské domény i v cizí doméně a musel tedy projít mezi doménovou komunikací, proto je v cizí doméně označen hvězdičkou. Obrázek je dílem autora na základě [11, 12].

Kapitola 3

Návrh

Tato kapitola se věnuje návrhu systému pro on-line správu kryptografických klíčů zabezpečovacího zařízení ETCS. Samotný návrh vznikl na základě funkčních a nefunkčních požadavků specifikovaných v kapitole 2.8.

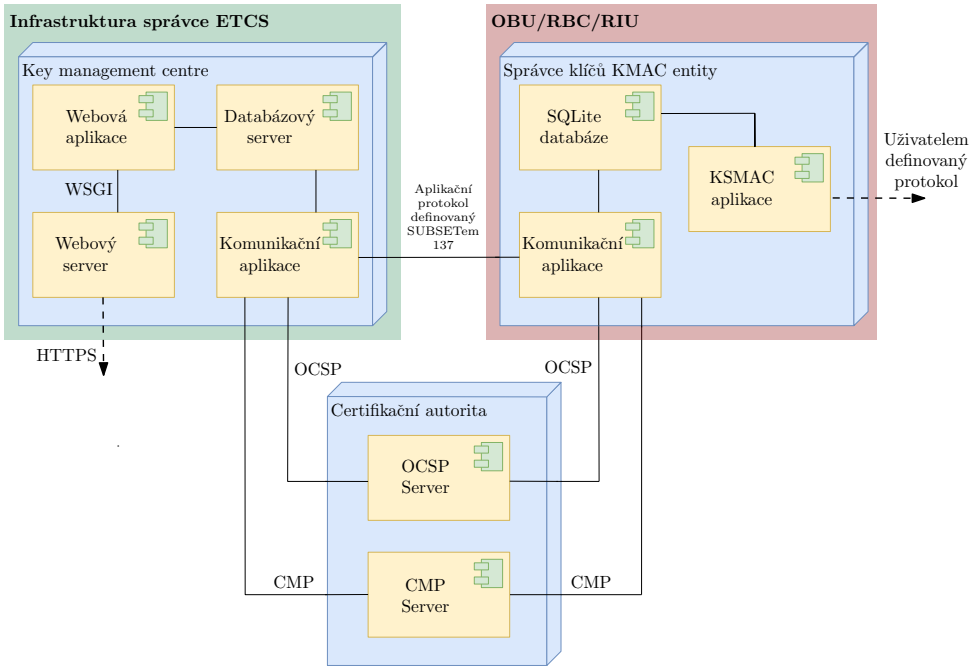
3.1 Komponenty

Výsledný systém pro správu klíčů se bude skládat z několika komponent, které mezi sebou budou vzájemně propojeny. Každá komponenta bude plnit předem definovanou funkcionalitu a komunikace mezi nimi bude probíhat předem určeným způsobem. V rámci reálného nasazení systému se každá komponenta nemusí vyskytovat pouze jednou, ale zpravidla se každá z nich může v daném systému vyskytovat mnohokrát s tím, že probíhá jejich seskupování do logických celků tak, jak je to popsané v kapitole 2. Níže uvedené komponenty je možné ještě dále rozdělit na menších celky, které jsou v této práci označeny jako moduly.

Zmíněné komponenty budou v závislosti na svém typu a situaci plnit roli serveru (nabízí službu) nebo roli klienta (využívá službu). Některé z nich jsou striktně svázány se zabezpečovacím systémem ETCS. Další z nich jsou obecné a jedná se tak o běžně dostupná řešení, na která mohou být kladeny další požadavky. Detailní popis každé komponenty bude uveden v dalších kapitolách.

Seznam komponent:

- Key management centre
- Správce klíčů KMAC entity
- CMP server
- OCSP Responder



Obrázek 3.1: Model nasazení

Na obrázku 3.1 je znázorněn zjednodušený model nasazení. Zjednodušení spočívá v tom, že se zde nachází od každé komponenty pouze jeden zástupce a je tak učiněno z důvodu zajištění přehlednosti. Každá komponenta je znázorněna pomocí modrého kvádru a jednotlivé moduly (aplikace) každé komponenty jsou znázorněny pomocí žlutého obdélníku. Diagram taktéž znázorňuje komunikaci mezi jednotlivými komponentami respektive moduly. Na obrázku došlo ke sloučení aplikace obsluhující OCSP a CMP požadavky pod jednu komponentu se dvěma moduly. Toto je jeden z možných a nejčastějších případů použití. V rámci této práce nebudou všechny znázorněné moduly implementovány, ale v některých případech dojde k využití již stávajících řešení.

3.2 Key management centre (KMC)

Komponenta Key management centre řeší funkční požadavky *F1*, *F2* a *F3* definované kapitolou 2.8. Umožňuje tedy správu domén, entit a samotných klíčů. Do funkcionality správy klíčů rovněž spadá možnost jejich generování. Samotné generování bude zajištěno kryptograficky bezpečným generátorem pseudonáhodných čísel. Tato komponenta by měla být umístěna v rámci infrastruktury organizace, která bude spravovat KMAC klíče. V následujících podkapitolách budou uvedeny a detailněji popsány jednotlivé moduly KMC komponenty.

3.2.1 Webová aplikace KMC komponenty

Jedná se o aplikaci, která bude uživatelům poskytovat možnosti správy domén, entit a samozřejmě také samotných klíčů. S touto aplikací budou moci pracovat pouze pověřené osoby, proto bude požadováno přihlášení. Kromě práce s klíči bude také umožněna registrace nových uživatelů a správa již existujících účtů. Mimo zmíněného bude webová aplikace rovněž zobrazovat seznam vytvořených příkazů spolu s jejich stavem. Příkazy, které skončily s chybou, bude možné odeslat cílovým zařízením znovu. Příkazy, u kterých ještě nedošlo k zahájení instalace (stále se nacházejí ve frontě příkazů čekajících na zpracování), bude možné přerušit, čímž se z popisované fronty vyjmou. Webová aplikace bude pro každý klíč poskytovat jednoduchou diagnostiku, která bude sloužit pro kontrolu stavu klíče na jednotlivých zařízeních. Kromě diagnostiky klíčů bude WA poskytovat také diagnostiku jednotlivých zařízení.

Z požadavků vyplývá, že se bude jednat o webovou aplikaci, tedy aplikaci poskytovanou webovým serverem a využívanou uživateli pomocí webových prohlížečů. Samotné propojení této aplikace s webovým serverem bude moci být realizováno pomocí rozhraní Web Server Gateway Interface (WSGI).

3.2.1.1 Architektura webové aplikace KMC komponenty

Při návrhu webové aplikace je počítáno s vícevrstvou architekturou, konkrétně se bude jednat o třívrstvou architekturu s využitím návrhového vzoru MVC. Tato volba zajistí nízkou provázanost jednotlivých dílčích částí celé aplikace a umožní tak jejich jednodušší udržovatelnost a přehlednost celé aplikace. Výměna jakékoliv části aplikace bude taktéž jednoduchá. Návrh tedy bude například umožňovat jednoduchou výměnu části starající se o zobrazování výstupu uživatelům bez nutnosti zásahu do části zabývající se prací s databází.

Při tvorbě této aplikace bude v závislosti na zvoleném programovacím jazyce využit některý z dostupných webových frameworků. Tato softwarová struktura bude sloužit jako podpora při vývoji aplikace a taktéž dokáže zajistit zmiňovanou třívrstvou architekturu pomocí návrhového vzoru MVC.

Prezentační vrstva aplikace bude realizována pomocí Hypertext Markup Language dokumentů (HTML) s využitím kaskádových stylů (CSS) pro definici způsobu zobrazení jednotlivých HTML elementů. Z důvodu toho, že se bude jednat o dynamickou webovou aplikaci, tak zmíněné dokumenty budou psány ve formě HTML šablon pro konkrétní zvolený webový framework a programovací jazyk. Doménová vrstva aplikace se bude starat o různé výpočty, zpracování dat a kontrolu vstupů zadaných uživateli. Datová vrstva bude zajišťovat komunikaci s databází popsanou v kapitole 3.2.2. Bude se jednat o komunikaci s relační databází za pomoci objektově relačního mapování, které zajistí odstínění webové aplikace od dotazovacího jazyka konkrétní databáze a rovněž pomůže zvýšit bezpečnost. Relační mapování rovněž zajistí příslušný webový framework.

3.2.1.2 Komunikace webové aplikace KMC komponenty s databází

Jak již bylo nastíněno v předchozí kapitole, tak webová aplikace bude komunikovat s relační databází. V rámci této komunikace se budou přenášet samotné klíče, informace o nich a informace o entitách i doménách. Pokud nastane potřeba komunikovat s fyzickou entitou (OBU, RBC/RIU), tak webová aplikace uloží do databáze instalační příkaz. Jedná se například o situaci, kdy je nutné nainstalovat nějaký klíč na fyzické zařízení. S relační databází bude rovněž komunikováno za účelem správy uživatelských účtů a kontroly oprávnění uživatelů. V tomto odstavci popsaná komunikace bude realizována za pomocí SQL příkazů.

3.2.2 Databáze KMC komponenty

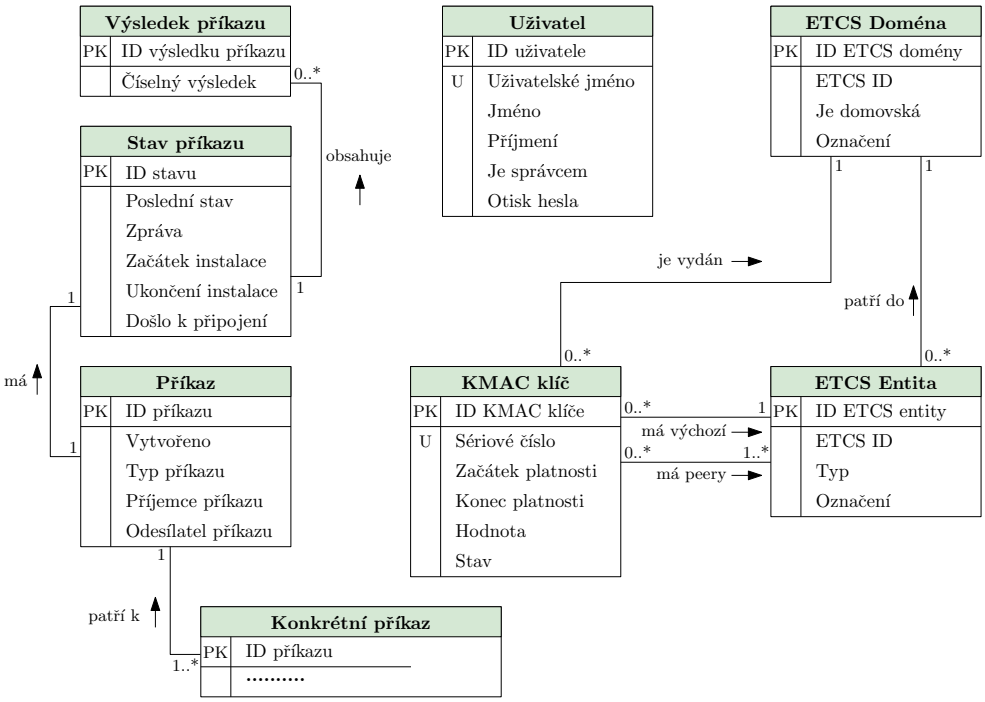
Účelem této komponenty je uchovávání všech informací týkajících se jedné domény systému pro správu klíčů zabezpečovacího zařízení ETCS. K tomuto účelu bude využita relační databáze. Tento typ databáze byl zvolen, protože se jedná o osvědčený, spolehlivý a často používaný způsob pro uchovávání dat. Relační databáze taktéž zajistí konzistenci dat a umožňuje jejich jednoduché zálohování. Dalším důvodem pro výběr relační databáze je zkušenost autora s tímto způsobem ukládání dat.

V databázi budou existovat záznamy vztahující se k hlavním požadavkům na systém pro on-line správu KMAC klíčů, ale budou zde taktéž uloženy podpůrné záznamy sloužící pro provoz webové aplikace. V databázi tedy budou uloženy informace o doménách, entitách a KMAC klíčích. Musí se zde rovněž vyskytovat záznamy popisující instalační příkazy, které budou využívány aplikací sloužící pro komunikaci s fyzickými entitami. Výčet obsahující nejdůležitější tabulky spolu s jejich atributy a vzájemnými vazbami, je uveden na obrázku 3.2.

K této databázi bude mít přístup pouze webová aplikace a komunikační aplikace KMC komponenty (viz. 3.2.4) zajišťující spojení s ostatními fyzickými entitami (OBU, RBC/RIU nebo jiné KMC). Některá data v databázi jsou citlivá (samotné klíče), a proto je nutné věnovat zvýšenou pozornost jejímu zabezpečení. Zabezpečení databáze a dalších prvků celého on-line systému pro správu klíčů je popsáno v kapitole 3.7.

3.2.3 Kryptograficky bezpečný generátor náhodných čísel

Na základě nefunkčního požadavku *N2* je nutné, aby KMC komponenta disponovala kryptograficky bezpečným generátorem náhodných čísel. Hodnoty ze zmíněného generátoru budou použity při generování KMAC klíčů v případě, že uživatel nezadá klíč ručně. Mimo to se náhodná čísla taktéž využijí pro zajištění zabezpečené komunikaci s ostatními entitami v rámci systému pro správu klíčů. Pokud by klíče byly generovány generátorem čísel, který není



Obrázek 3.2: Model databáze

kryptograficky bezpečný, tak by mohli potenciální útočníci například předpovídat klíče, což by mohlo vést k ohrožení bezpečnosti. Pro účely generování náhodných čísel bude využit pseudonáhodný generátor náhodných čísel, který poskytuje knihovna OpenSSL [32].

3.2.4 Komunikační aplikace KMC komponenty

Jedná se o další modul KMC komponenty, který má zajistit samotnou komunikaci s fyzickými zařízeními (OBU, RBC/RIU, jiné KMC) v rámci systému pro správu klíčů. Tato aplikace bude vycházet ze své generické verze a bude tak pouze přizpůsobena pro použití na KMC. Dané přizpůsobení bude spočívat ve změně konfiguračního souboru a vytvoření třídy implementující funkcionalitu KMC s využitím společné části. Z důvodu toho, že detailní popis generické komunikační aplikace je obsažen v kapitole 3.4, tak zde je popsán pouze způsob propojení právě popisované aplikace s jinými moduly v rámci KMC. Dále jsou v této kapitole zmíněny odlišnosti oproti generické verzi aplikace.

Komunikační aplikace bude využívat dvě hlavní komunikačními rozhraní. *Rozhraní pro komunikaci s databází klíčů* propojí komunikační aplikaci s relační databází. V rámci tohoto rozhraní budou získávány jednotlivé příkazy vytvořené webovou aplikací pro fyzické entity. Kromě nich mají být tímto rozhraním rovněž posílána data, která se k samotným příkazům vztahují. Po vyzvednutí příkazu se pomocí *rozhraní pro komunikaci s fyzickými entitami*

odešle příkaz spolu s daty cílové fyzické entitě, které je příkaz směřován. Odesílané informace musejí být ve formátu, kterému druhá strana rozumí (bude využit aplikační protokol definovaný dokumentem ERTMS/ETCS SUBSET 137 [11]). Právě popisovaná komunikace bude probíhat s aplikací vycházejí rovněž z generické verze komunikační aplikace, ale pouze umístěné na cílové fyzické entitě (viz. 3.3.2). Poté co cílová entita provede požadovanou operaci, tak vrátí výsledek prováděné operace zpět odesílateli (právě popisované komunikační aplikaci). Na základě odpovědi zapíše komunikační aplikace do databáze výsledek operace bez interakce s webovou aplikací. Druhé ze zmiňovaných rozhraní bude rovněž poslouchat, zdali se s danou entitou nepotřebuje spojit entita jiná. Bude se tak v zásadě jednat o klientskou i serverovou aplikaci. V kapitole 3.4.5 je pro přehled znázorněno, v jakých případech se komunikační aplikace chová jako server a jakých jako klient. Mimo těchto hlavních komunikační rozhraní se budou využívat rozhraní pro komunikaci v rámci infrastruktury veřejného klíče (PKI).

Pokud je vytvořen příkaz mající efekt na fyzickou entitu, tak se o tomto příkazu musí komunikační aplikace dozvědět, aby mohla dané entitě příkaz odeslat spolu s potřebnými daty. Webová aplikace, pomocí které je nový příkaz vytvořen, pouze vkládá záznam do databáze a neposílá žádné notifikace. Způsob, jakým se o novém příkazu komunikační aplikace dozví, závisí na typu cílové entity. Pokud se jedná o entitu typu OBU, tak se komunikační aplikace KMC chová jako server. OBU tak vždy vytváří nové spojení s KMC, pomocí kterého mu může KMC případně poslat příkaz. Komunikační aplikaci tedy v tomto případě není nutné informovat o novém příkazu, protože se čeká až na dotaz od OBU. V případě, že tento dotaz nastane, tak komunikační aplikace aktivně zkontroluje příkazy v databázi. V případě, že je instalační příkaz určen pro RBC, RIU nebo jiné KMC, tak TLS spojení vytváří KMC. Z tohoto důvodu musí komunikační aplikace, která dané spojení vytváří, vědět o vzniku nového příkazu, aby mohla otevřít nové spojení s cílovou entitou. Informaci o tomto nově vzniklém příkazu poskytne databázový server pomocí notifikace. Z tohoto důvodu musí být využit databázový server s podporou notifikací. Tento způsob je efektivnější, než kdyby probíhalo neustále aktivní dotazování, zdali do databáze nebyl vložen nový příkaz.

3.3 Správce klíčů KMAC entity

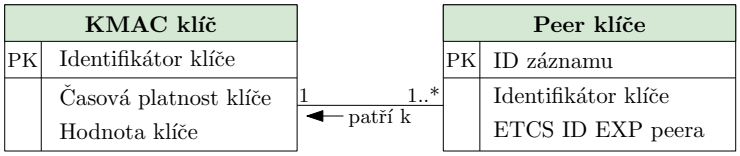
Správce klíčů KMAC entity řeší požadavek *F4* definovaný v kapitole 2.8.1. Tato komponenta bude součástí KMAC entit (OBU, RBC/RIU), které využívají protokol Euroradio pro svou komunikaci. Cílem této komponenty je získání KMAC klíče od centra pro správu klíčů (KMC) pomocí on-line komunikace, uschování přijatého klíče a v případě potřeby vydání z něj odvozeného klíče modulu obstarávajícímu komunikaci pomocí protokolu Euroradio. Modul pro Euroradio komunikaci není součástí systému pro správu klíčů (KMS)

a tudíž není ani součástí této práce. Kromě přijímání požadavků na instalaci klíčů bude tato komponenta řešit i další požadavky související se správou klíčů. Samotná komponenta se bude ještě dělit na více modulů.

3.3.1 Databáze klíčů KMAC entity

Jedná se o úložiště, které bude na KMAC entitě uchovávat KMAC klíče a informace o nich. Tato data budou ukládána do zjednodušené relační databáze. Zjednodušení bude spočívat v tom, že se nebude jednat o plnohodnotnou databázi s databázovým serverem, ale bude se jednat o SQLite databázi. Tato databáze je reprezentována jedním souborem, který obsahuje všechna data uchovávaná databází. Pro práci s daným souborem slouží knihovny vytvořené za tímto účelem. Tyto knihovny budou použity v rámci komunikační aplikace na správci klíčů KMAC entity (popsáno v kapitole 3.3.2). Zjednodušený typ databáze byl zvolen, protože se jedná o jednoduché řešení, které je plně dostačující pro použití v rámci KMAC entity. SQLite databázi je zde možné použít, protože zde není potřeba ukládat velké množství dat a rovněž zde není nutné řešit problémy zápisu dvou a více aplikací (vláken) do databáze v jeden okamžik.

Na obrázku 3.3 je znázorněn model, který ukazuje databázi pro ukládání dat na komponentě pro správu klíčů KMAC entity. V rámci konfiguračního souboru komunikační aplikace pro KMAC entitu (viz. kapitola 3.3.2) bude moci být nastaven limit pro maximální počet uložených klíčů. Toto omezení je zavedeno v rámci ochrany před případným zaplnění paměti v případě využití zařízení s nízkou kapacitou úložiště.



Obrázek 3.3: Model databáze KMAC entity

3.3.2 Komunikační aplikace KMAC entity

Hlavním úkolem této aplikace je zajištění TLS spojení mezi správcem klíčů KMAC entity (právě popisovaná komponenta) a centrem pro správu klíčů (KMC). Stejně jako v případě komunikační aplikace pro KMC (viz. 3.2.4) vychází i tato aplikace ze své generické verze, které se věnuje kapitola 3.4. Z tohoto důvodu jsou v této kapitole opět popsány pouze odlišnosti oproti generické verzi. Dále se zde nachází zmínka o využití aplikace v rámci celého systému pro správu klíčů a opět je zde popsána komunikace s okolními moduly v rámci komponenty KMAC entity.

Komunikační aplikace správce klíčů KMAC entity tedy bude komunikovat se svým protějškem na straně KMC v rámci on-line systému správy klíčů. Tato komunikace bude probíhat pomocí TLS rozhraní pro *komunikaci s fyzickými entitami*, které bude generické. Popisovaná aplikace se bude chovat jako TLS klient, i jako TLS server v závislosti na typu entity, na které bude nainstalovaná. V rámci tohoto rozhraní tedy mají být doručovány požadavky správy klíčů vydané KMC. V případě, že si typ doručeného požadavek bude žádat přístup k úložišti klíčů, tak bude využito rozhraní pro *komunikaci s SQLite databází klíčů* pro vyřízení konkrétního požadavku. Tato komunikace bude prováděna pomocí klasických SQL příkazů. Stejně jako v případě komunikační aplikace pro KMC bude i zde nutné upravit konfigurační soubor.

3.3.3 KSMAC modul KMAC entity

Jedná se o další modul na komponentě pro správu klíčů KMAC entity, který bude poskytovat KSMAC klíče modulu pro Euroradio komunikaci. KSMAC klíč je odvozený z KMAC klíče pro jednotlivé relace a je to jediný možný typ klíče, který je možné ze správce klíčů získat. Tento modul nesouvisí s primární funkčností systému pro on-line správu klíčů a proto nebude v rámci této práce implementován. Výsledný návrh správce klíčů však s tímto modulem počítá.

3.4 Generická verze komunikační aplikace

Jedná se o stěžejní aplikace celého systému pro on-line správu kryptografických klíčů zabezpečovacího zařízení ETCS. Jejím úkolem bude zajištění komunikace mezi jednotlivými komponentami celého systému (OBU – KMC, RBC/RIU – KMC, KMC – KMC). Právě popisovaná generická verze komunikační aplikace nebude v reálu nasazena na žádné z komponent, protože se vlastně bude jednat o sadu knihoven. Pro nasazení komunikační aplikace na fyzické komponenty bude nutné implementovat chování jednotlivých entit. Toto je nutné, protože se každá z komponent chová jinak a jsou na ně rovněž kladeny různé požadavky. Rovněž také bude nutné upravit konfigurační soubor.

V následujících kapitolách budou popsána všechna komunikační rozhraní jak pro komunikaci v rámci jedné komponenty (tedy mezi moduly), tak pro komunikaci mezi komponentami (TLS komunikace). Úpravy, které budou muset být provedeny pro použití na jednotlivých entitách, již nebudou v této kapitole diskutovány. Tato diskuze byla provedena u jednotlivých komponent, tedy v kapitolách 3.2 a 3.3.

3.4.1 Rozhraní pro komunikaci s databází klíčů

Toto komunikační rozhraní bude využíváno jak pro čtení dat z databází entit, tak i pro zápis dat do databází. Samotné komunikační rozhraní bude poskytováno pomocí podpůrného objektu v závislosti na druhu databáze. Přenášená

data na aplikační úrovni nebudou zabezpečena za účelem zajištění integrity, autenticity a důvěrnosti. Více informací ohledně zabezpečení přenášených dat se nachází v kapitole 3.7.

3.4.2 Rozhraní pro komunikaci s jinými entitami

Rozhraní pro komunikaci s jinými entitami je jedním z nejdůležitějších rozhraní celé komunikační aplikace, protože se jedná o hlavní funkcionalitu systému pro on-line správu kryptografických KMAC klíčů. V rámci tohoto rozhraní bude využit protokol TCP, který garantuje integritu přenášených dat. Pro zajištění bezpečnosti (autenticita, důvěrnost) přenášených dat se využije TLS protokol. K ověření identity jednotlivých stran v rámci TLS protokolu budou využity certifikáty infrastruktury veřejného klíče (TLS-PKI). Všechny strany komunikace budou vždy ověřovat certifikát svého protějšku. Na základě požadavků ze strany předpisů se bude využívat TLS protokol verze 1.2, který je již v době psaní této práce v rámci webové komunikace nahrazován novějším protokolem verze 1.3. Spojení, která budou požadovat využití starších verzí TLS protokolu (TLS 1.0 nebo TLS 1.1), budou automaticky odmítána. Podporu pro novější TLS protokoly (TLS 1.3, ...) bude možné přidat jednoduchou úpravou zdrojových souborů komunikační aplikace. Popisované TLS rozhraní se v některých situacích bude chovat jako klientské a v jiných situacích jako serverové. Konkrétní situace se bude odvíjet o toho, na jakém zařízení je komunikační aplikace nainstalována a rovněž také od toho s jakým zařízením se má v daný okamžik komunikovat. Přehled konkrétních situací spolu s chováním komunikační aplikace je uveden v kapitole 3.4.5. Právě popisované komunikační rozhraní se bude chovat tak, jak je popisováno v dokumentu [11]. Řešení případných chyb (neočekávané uvolnění spojení, ...) bude taktéž implementováno v souladu se zmíněným dokumentem.

V rámci TLS je možné definovat šifrovací sadu, která bude využita pro zabezpečení komunikace. Podporované a výchozí šifrovací sady bude možné měnit. TLS rozhraní bude v základu podporovat (z požadavků dokumentu [11]):

- TLS-PKI: *TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384*

Pokud se využívá ověřování komunikujících stran pomocí certifikátů infrastruktury veřejného klíče, tak je nutné kontrolovat platnost certifikátů. Tato kontrola bude zahrnovat ověření časové platnosti a také bude ověřováno, zdali již nebyl daný certifikát revokován nebo zda opravdu pochází od důvěryhodné certifikační autority. Tuto kontrolu bude provádět jak server, tak i klient. TLS klient připojující se k serveru bude pomocí doménového jména z certifikátu ověřovat, zda se připojit k serveru, ke kterému se připojit chtěl.

Na aplikační úrovni budou pomocí tohoto rozhraní přenášeny zprávy protokolem definovaným v dokumentu [11]. Přenášené zprávy se skládají z hlaviček a samotného obsahu. V rámci hlavičky je přenášena délka zprávy, verze

rozhraní, typ zprávy (jaká operace se má na cílovém zařízení provést), identifikátor příjemce a odesílatele, sekvenční číslo a číslo transakce. Některé zprávy mohou mít pouze hlavičku. Všechny zprávy musejí být na aplikační úrovni přenášeny v binárním formátu s BigEndian pořadím bajtů. Komunikační aplikace bude podporovat všechny zprávy definované SUBSETem 137 [11]. Žádné jiné zprávy (příkazy) podporovány nebudou. Komunikační aplikace bude podporovat kontrolu stavu KMAC entity. V rámci této operace se bude KMC snažit kontaktovat KMAC entitu za účel ověření jejího stavu (zdali dokáže KMAC entita reagovat na požadavky). Dokument popisující aplikační protokol nedefinuje žádný příkaz k tomuto účelu, proto bude využit příkaz, který požaduje pro KMAC entitě výpočet kontrolního součtu její databáze. Tento příkaz byl zvolen, protože nijak nemodifikuje databázi KMAC entity.

3.4.3 OCSP rozhraní

Toto komunikační rozhraní bude sloužit pro ověřování platnosti certifikátů infrastruktury veřejného klíče při TLS komunikaci. K tomuto účelu bude využit Online Certificate Status Protocol (OCSP), který je detailněji popisován v kapitole 2.9.5. Jako OCSP responder (odpovídá na dotazy ohledně platnosti certifikátů) bude sloužit zařízení popisované v kapitole 3.6. Komunikační aplikace bude vždy v roli OCSP klienta.

Právě popisované komunikační rozhraní bude součástí generické verze komunikační aplikace a nebude jej nutné upravovat pro použití na jednotlivých zařízení. Podpora pro Online Certificate Status Protocol nebude v komunikační aplikaci implementována od začátku, ale bude využito existující řešení v podobně knihovny OpenSSL. V rámci komunikační aplikace dojde pouze ke konfiguraci zmíněného protokolu pro správnou funkčnost.

3.4.4 CMP rozhraní

Toto komunikační rozhraní bude sloužit pro správu a distribuci certifikátů infrastruktury veřejného klíče. Pro tyto účely bude využit Certificate Management Protocol (CMP), který je detailněji popisován v kapitole 2.9.4. Na nižší vrstvě síťové komunikace bude využit TCP protokol s tím, že celá komunikace mezi CMP klientem a CMP serverem proběhne v rámci jednoho TCP spojení. Komunikační aplikace se vždy bude chovat jako CMP klient. CMP server bude zařízení propojené s certifikační (registrační) autoritou. Tento server bude detailněji popsán v kapitole 3.6.

CMP rozhraní bude součástí komunikační aplikace a využije se v případě žádosti o nový certifikát (pro nové zařízení) a také v případě nutnosti výměny již existujícího certifikátu z důvodu blížícího se data konce platnosti starého certifikátu. CMP server bude součástí konfigurace komunikační aplikace. Pro ověření identity komunikačních stran, tedy CMP klienta a CMP serveru se při žádosti o první certifikát využije sdílené tajné heslo. Podle požadavků ze strany

dokumentu [11] nesmí být toto heslo součástí počáteční konfigurace a musí být zadáno až v případě jeho potřeby. Toto bude vyřešeno tím způsobem, že se během instalační fáze zařízení toto heslo zadá a rovnou v ten samý okamžik operátor požádá certifikační (registrační) autoritu o certifikát. Po obdržení zmíněného certifikátu již nebude zmíněné heslo potřebné, protože CMP server již bude k ověření CMP klienta využívat jeho certifikát.

V rámci komunikační aplikace nebude Certificate Management Protocol implementovaný od začátku, ale bude využito existující řešení v podobně knihovny. V rámci komunikační aplikace dojde stejně jako v případě OCSP pouze ke konfiguraci knihovny pro správnou funkčnost.

3.4.5 Chování komunikační aplikace v různých situacích

V následujícím seznamu je uvedeno, jaký typ funkcionality (TLS klient nebo TLS server) musí být podporován při jaké komunikaci. Tento seznam vychází z dokumentu [11].

1. KMC musí implementovat funkci serveru pro komunikace:
   a) KMC–KMC
   b) KMC–OBU
2. KMC musí implementovat funkci klienta pro komunikace:
   a) KMC–KMC
   b) KMC–RBC/RIU
3. OBU musí implementovat funkci klienta pro komunikace:
   a) KMC–OBU
4. RBC/RIU musí implementovat funkci serveru pro komunikace:
   a) KMC–RBC/RIU

3.5 Proces od vytvoření až po využití KMAC klíče

Zjednodušené schéma komunikace na obrázku 3.4 zachycuje proces od vytvoření KMAC klíče (na KMC) až do vyřízení požadavku o vygenerování jednorázového KSMAC klíče (na KMAC entitě) pro Euroradio modul. Jedná se tak o jeden z možných scénářů komunikace. Pro zjednodušení zachycuje obrázek pouze komunikaci v rámci jedné domény (KMC a KMAC entita patří do stejné domény). Šipky na obrázku označené číslem značí jakým směrem a v jakém pořadí se přenášejí příkazy nebo data. Na obrázku je z důvodu přehlednosti zachycena pouze stěžejní komunikace. Znázorněny tedy nejsou

například některé odpovědi potvrzující provedení operace. Nezakreslená komunikace je rovněž důležitá, ale již nepřispívá k pochopení celkového konceptu komunikace. Obrázek dále předpokládá, že jsou již vytvořeny příslušné KMAC entity a ETCS domény. Komunikace v rámci PKI infrastruktury zde rovněž není zakreslena. Postup při vytváření a instalaci klíče je tedy následující:

1. **2.** uživatel pomocí *webové aplikace* vytvoří KMAC klíč, přiřadí ho výchozí entitě a přidá mu peery. Rovněž nařídí instalaci klíče na KMAC entitu.

   3. *webová aplikace* uloží KMAC klíč a informace o něm do *databáze*. Do zmíněné *databáze* je rovněž uložen instalační příkaz pro cílovou fyzickou KMAC entitu.

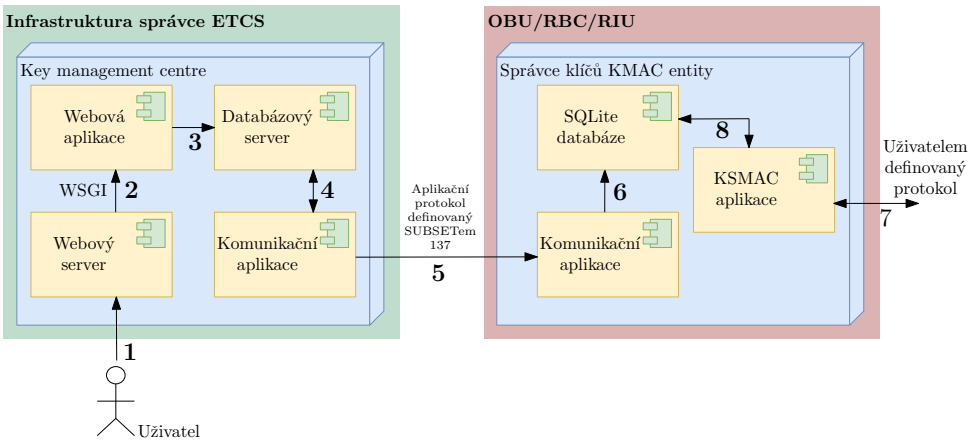
   4. značí přenos instalačního příkazu a s ním souvisejících dat z *databáze* do *komunikační aplikace*. Tento přenos iniciuje buď sama *komunikační aplikace* dotazem do databáze a nebo *databázový server* notifikací.

   5. spojení mezi *komunikační aplikací* centra pro správu klíčů a tou samou aplikací cílové KMAC entitě. Cílová entita je myšlena ve smyslu toho, kam se má klíč fyzicky nainstalovat. Pomocí této komunikace se přenese klíč z KMC na cílovou entitu.

   6. *komunikační aplikace* KMAC entity uloží přijatý KMAC klíč do své *databáze*. Od tohoto okamžiku tak může entita nainstalovaný klíč používat.

   7. Pokud modul pro Euroradio komunikaci (není součástí obrázku) potřebuje pro svou činnost KSMAC klíč, tak o něj požádá *KSMAC aplikaci*.

   8. K tomu aby KSMAC aplikace mohla KSMAC klíč vydat, tak musí proběhnout komunikace s databází klíčů KMAC entity.



Obrázek 3.4: Schéma komunikace komponent KMS systému

3.6 Certifikační autorita

Další komponentou on-line systému pro správu klíčů je certifikační autorita. Jejím účelem je správa certifikátů infrastruktury veřejného klíče. Kromě vydávání certifikátů a jejich obnovování bude rovněž v případě potřeby odvolávat platnost již vydaných certifikátů. Tato certifikační autorita bude s okolními zařízeními systému pro správu klíčů automaticky komunikovat pomocí *Certificate Management protokolu* a *Online Certificate Status protokolu*.

V rámci návrhu se počítá s tím, že bude využito již existující řešení implementující certifikační autoritu i s podporou služeb CMP serveru a OCSP responderu. Pro tyto účely bude využit softwarový balíček EJBCA, který splňuje všechny definované požadavky.

3.7 Bezpečnost a řešení chyb

Tato kapitola se věnuje bezpečnosti celého systému on-line správy klíčů zabezpečovacího zařízení ETCS. Řešeno je zde zajištění integrity, autenticity a důvěrnosti přenášených i uložených dat. Kromě zmíněného se tato kapitola věnuje spolehlivosti celého systému a to hlavně z pohledu zotavení se z chyb při případném neplánovaném ukončení některé z aplikací. Chyby při přenosu jsou řešeny v rámci příslušných protokolů. U komunikace mezi moduly jedné komponenty nebude řešen problém autenticity a důvěrnosti přenášených dat. Na druhou stranu integrita přenášených dat bude řešena i u komunikace uvnitř jedné komponenty.

Bezpečnost a spolehlivost celého systému bude popisována na schématu komunikace, který se nachází na obrázku 3.4. Samotný popis bude postupovat popořadě tak, jak je znázorněno na zmíněném obrázku.

3.7.1 Webová aplikace KMC komponenty

Uživatelé budou k webové aplikaci přistupovat pomocí HTTP protokolu. K zajištění bezpečnosti přenášených dat mezi webovým prohlížečem a HTTP serverem se musí využít zmíněný protokol s TLS nástavbou, tedy HTTPS. HTTPS protokol funguje nad TCP protokolem, čímž je zajištěna integrita přenášených dat. Webová aplikace není využívána žádnou službou (není na ní závislá jiná aplikace), ale využívají ji pouze uživatelé. Z tohoto důvodu není nutné řešit automatické zotavování webové aplikace z chyb. V případě havárie aplikace (serveru) chybu nahlásí samotní uživatelé. Pro zajištění komunikace mezi webovou aplikací a databází centra pro správu klíčů (KMC) bude rovněž využit TCP protokol, který zajistí integritu přenášených dat.

3.7.2 Databázový server KMC komponenty

Pokud nastane závažná chyba na straně databázového serveru, který v jejím důsledku přestane poskytovat své služby, tak tuto skutečnost detekují aplikace využívající služby databáze. Tyto aplikace následně budou muset nahlásit danou chybu. Komunikace mezi databázovým serverem a komunikační aplikací bude za účelem zajištění integrity realizována pomocí TCP protokolu.

V právě popisované databázi budou uložena citlivá data (KMAC klíče v otevřené podobně, otisky hesel uživatelů, ...). Z tohoto důvodu musí být zajištěna ochrana uložených dat. Databázový server by měl být nastaven tak, aby neakceptoval připojení realizovaná z vnější sítě. Pro přístup k databázi bude zapotřebí zadat uživatelské jméno a heslo. Server bude evidovat dva uživatelské účty určené pro potřeby aplikací. První z nich bude sloužit webové aplikaci. Tento účet bude mít práva pro práci s databází v rozsahu, který je pro činnost webové aplikace potřeba. Jedná se ale o celou databázi. Druhý uživatelský účet bude sloužit komunikační aplikaci a tento účet již bude mít výrazněji omezená oprávnění pro práci s databází. Toto omezení bude spočívat v umožnění a zápisu a čtení pouze těch dat, která jsou pro komunikační aplikaci potřebná.

3.7.3 Komunikační aplikace KMC komponenty

Přenos dat mezi komunikačními aplikace je zabezpečen jak z pohledu integrity (TCP protokol), tak z pohledu autenticity a důvěrnosti (TLS). V rámci tohoto protokolu se bude kontrolovat platnost certifikátů obou komunikujících stran.

V případě havárie komunikační aplikace se o této skutečnosti dozví druhá strana komunikace, protože dojde k rozvázání TCP spojení. Systém musí být nastaven tak, aby se v případě havárie komunikační aplikace znovu sama spustila. Chování po restartu dané aplikace se bude lišit na základě typu dané aplikace (zdali se jedná o aplikaci pro KMC, RBC/RIU nebo OBU).

V případě KMC verze jsou po každém novém spuštění zkontrolovány nevyřízené požadavky, které jsou uchovávány v databázovém serveru KMC. Nalezené požadavky jsou následně vyřízeny. Pokud se v databázi bude vyskytovat požadavek, který se již začal zpracovávat (došlo k chybě na jedné nebo druhé straně během vyřizování), tak komunikační aplikace ověří, zda již byla instalace fakticky provedena pomocí příslušné zprávy druhé entitě.

Pokud se restartovaná komunikační aplikace nachází na entitě typu OBU, tak pouze dojde ke kontaktování KMC s dotazem na nové požadavky. V případě RBC/RIU není po spuštění prováděna žádná akce.

Rovněž je nutné rozebrat případy, kdy není druhá strana komunikace dostupná, nebo dojde k chybě během komunikace. Chování v takovémto případě bude záviset na tom, zdali se komunikační aplikace chová jako klient nebo server. Pokud se komunikační aplikace bude chovat jako server, tak tato aplikace nebude po svém restartu aktivně vykonávat žádnou činnost a bude pouze na-

slouchat, aby případně mohla přijmout nového spojení. Obnova komunikace je tak povinností klienta. Pokud nebude moci klient otevřít spojení, tak tuto akci zopakuje několikrát po sobě v definovaných časových odstupech. Pokud se ani po několikáté nepodaří se serverem spojit, tak dojde k nahlášení daného problému.

Pokud nastane chyba v komunikaci mezi komunikační aplikací a úložištěm, respektive databází klíčů (entit, domén, příkazů), tak nebudou moci být vyřizovány příkazy směřující na danou entitu. Z tohoto důvodu budou tyto příkazy odmítány. Odmítnutí vyřízení požadavku bude realizováno ukončením TLS spojení.

3.7.4 Databáze KMAC entity

Jedná se o databázi, která je součástí správce klíčů KMAC entity. Tato databáze bude obsahovat citlivá data (hodnoty klíčů). Z tohoto důvodu je nutné zabývat se bezpečností této databáze. Jak již bylo zmíněno v návrhu, tak se bude jednat o SQLite databázi. Tento typ úložiště dat neposkytuje podporu pro řízení přístupu uživatelů. Z tohoto důvodu bude nutné řídit přístup k databázi na úrovni operačního systému. K souboru s databází tak budou mít přístup pouze někteří uživatelé systému.

3.7.5 KSMAC modul KMAC entity

Správce klíčů KMAC entity může poskytovat rozhraní pro vydávání KSMAC klíčů Euroradio modulu, proto je nutné zabývat se bezpečností přenášených dat ven z popisované komponenty. K účelu poskytování KSMAC klíče nebude využita komunikační aplikace, protože tato aplikace bude využita pro TLS komunikaci (tedy pro komunikaci směřovanou do vnější sítě). Z tohoto důvodu může být komunikační aplikace náchylná například k (D)DoS útokům. Pokud by takový útok nastal mohl by tuto aplikaci zpomalit nebo úplně znemožnit její používání, což by rovněž znemožnilo vydávání KSMAC klíčů. Modul pro Euroradio komunikaci tedy bude využívat nástroj určený pro poskytování KSMAC klíčů, který je popsán v kapitole 3.3.3. Tímto krokem se modul pro Euroradio komunikaci odstíní od databáze klíčů, čímž se zvýší bezpečnost. KSMAC modul bude přímo spojen s úložištěm klíčů. Komunikace mezi úložištěm a KSMAC modulem bude opět zajištěna na úrovni operačního systému, který bude řešit bezpečný přenos.

Komunikace mezi Euroradio modulem a KSMAC modulem bude záviset na implementaci Euroradio modulu, proto zde tato komunikace nebude popisována.

Kapitola 4

Implementace

Tato kapitola se zabývá realizací prototypu systému pro on-line správu klíčů (KMS) zabezpečovacího zařízení ETCS. Výsledná implementace systému vychází z návrhu, který je uveden v kapitole 3. Popis implementace je rozdělen do několika kapitol podle popisovaných komponent.

4.1 Komunikační aplikace

Komunikační aplikace primárně zajišťuje komunikaci mezi jednotlivými entitami (zařízeními) celého systému. Z tohoto důvodu se tedy jedná o nejdůležitější aplikací celé práce. Návrh popisované aplikace je uveden v kapitolách 3.4, 3.2.4 a 3.3.2. V těchto kapitolách je zmíněno, že existuje několik verzí komunikační aplikace. Všechny tyto verze vychází z jedné generické verze. Generická verze komunikační aplikace je vlastně sada modulů psaných v rámci této práce. Každý z těchto modulů řeší jeden konkrétní problém. Nejdůležitějších moduly budou diskutovány v následujících kapitolách. Implementace komunikační aplikace pro konkrétní entitu je použití vhodných modulů spolu s jejich správným nastavením. Následující části se budou zabývat jednotlivými moduly generické verze.

Pro implementaci komunikační aplikace byl zvolen programovací jazyk C++. Dále byly využity některé externí volně dostupné knihovny. Popis a použití těchto knihoven bude uvedeno v následujících kapitolách. Při implementaci aplikace byl kladen důraz na využití moderních konstrukcí jazyka C++, proto je doporučené kompilovat zdrojový kód kompilátorem podporujícím standard C++20. Třídy a metody jsou opatřeny dokumentačními komentáři pro nástroj *Doxygen*, který umožní vytvoření dokumentace generované ze zdrojových kódů. Komunikační aplikace rovněž podporuje zaznamenávání své činnosti (logování) s možností nastavení úrovně. Jako podpora pro sestavení binárních souborů aplikace slouží nástroj CMake.

4.1.1 TLS Support

TLS Support poskytuje podporu pro komunikaci pomocí protokolu TLS. Při implementaci popisovaného modulu je využita knihovna **OpenSSL**. Mezi nejdůležitější části modulu se řadí třídy implementující *TLS server* a *TLS klient*. Další důležitou částí modulu je třída *SSLSupport*. Instance této třídy konfigurují SSL kontext OpenSSL knihovny podle požadavků SUBSETu 137 [11]. V rámci konfigurace kontextu je nastavován certifikát, soukromý klíč a úložiště důvěryhodných certifikátů majitele kontextu. Rovněž je nastavena minimální a maximální podporovaná verze TLS protokolu (oboje na TLS 1.2). Dále je zakázána komprese a nastaveno je rovněž ověřování protistrany pomocí certifikátů. Přímo ve třídách *TLS server* a *TLS klient* je v části věnující se navazování spojení nastaven požadavek na ověření doménového jména protistrany a také jsou zde uvedeny podporované šifrovací sady. Po provedení operace TLS handshake se zkontroluje platnost certifikátu. Pokud všechny operace proběhnou úspěšně, tak je navázáno TLS spojení a řízení je předáno obsluze spojení, v opačném případě k otevření TLS spojení nedojde.

4.1.2 SQLSupport

Tento modul slouží jako podpora pro interakci s relačními databázemi. V rámci komunikační aplikace je implementována podpora pro SQLite a PostgreSQL databáze. Instance tříd právě popisovaného modulu umožňují otevřít nové spojení s databází, uzavřít dané spojení a vrátit objekt reprezentující spojení pro další práci s databází. Pro implementaci třídy podporující komunikaci s PostgreSQL databází je využita knihovna **libpqxx**. Naopak knihovna **sqlite3** je využita pro implementaci komunikace s SQLite databází. Zmíněné knihovny mimo jiné zajišťují odeslání dotazů databázi. V případě, že je nutné vložit do SQL příkazu argumenty, tak je využita konstrukce Prepared statement. Tato konstrukce zabraňuje útokům typu SQL Injection. Na výpisu kódu 4.1 je ukázána funkce pro kontrolu stavu klíče. Tato ukázka demonstruje způsob provedení dotazu do databáze s využitím knihovny **sqlite3** a konstrukce Prepared statement. Původně byla celá komunikace s databázemi řešena pomocí knihovny **Qt** podporující obě výše zmíněné databáze, ale také mnoho jiných databázových systémů. Tato knihovna nakonec použita nebyla, z důvodu její vysoké komplexnosti.

Právě popisovaný modul rovněž obsahuje podporu pro databáze konkrétních entit. Jedná se o třídu *KmacEntityDbSupport*, která rozšiřuje třídu *SQLiteDbSupport*. Instance této třídy umožní komunikaci s databází KMAC entity (OBU, RBC/RIU) a její metody například umožní zkontrolovat, zda již nedošlo k zaplnění kapacity úložiště. Dále také vypočtou kontrolní součet databáze nebo zkontrolují, zda se klíč daného identifikátoru v úložišti již nenachází. Pro KMC entity existuje třída *KmcDbSupport*, která rozšiřuje *PostgreSQLDbSupport*. Její instance dokáží zkontrolovat, zdali neexistuje nový příkaz pro některé

```
bool isInstalled(const KIdentifier &key) {
  sqlite3_stmt *stmt;
  const char *q = "SELECT key FROM KMAC_KEY WHERE key=?";
  sqlite3_prepare_v2(db, q, -1, &stmt, NULL);
  sqlite3_bind_text(stmt, 1, key, -1, SQLITE_TRANSIENT);
  int result = sqlite3_step(stmt);
  if (result != SQLITE_ROW && result != SQLITE_DONE) {
      throw DatabaseException("Cannot execute SQL query.");
  }
  if (result == SQLITE_ROW) {
      return true;
  }
  return false;
}
```

Výpis kódu 4.1: Ukázka SQL dotazu pomocí sqlite3 (upraveno, zkráceno)

ze spravovaných zařízení. Rovněž je možné získat frontu příkazů pro zadané zařízení atd.

Pokud nastane potřeba zpracovat na některém ze zařízení přijatou zprávu (například na KMAC entitu přišel příkaz AddKeyCommand a je tedy potřeba vložit do databáze nový klíč), tak je nutné pomocí vhodného SQL příkazu provést úpravu databáze. Z tohoto důvodu existují dvě abstraktní třídy *ProcessableOnKmc* a *ProcessableOnKmacEntity*. Každá z těchto tříd obsahuje abstraktní metodu určující, co se má s databází stát po zavolání dané metody (myšleno po zavolání implementované metody na potomcích). Konkrétní KMS příkaz (například AddKeyCommand) pouze implementuje popisovanou abstraktní metodu podle svého typu. Pokud přijde na některou z entit příkaz, tak daná entita pouze zavolá implementovaná metodu, čímž se příkaz na dané entitě provede.

4.1.3 KMS support

Tento modul se vztahuje přímo k systému pro on-line správu klíčů. Jedná se o implementaci aplikačního protokolu, který definuje SUBSETem 137 [11]. Modul KMS support obsahuje struktury používané v rámci KMS zpráv, které jsou spolu se svým chováním rovněž implementovány právě popisovaným modulem. Všechny třídy reprezentující KMS zprávy (například AddKeyCommand, CmdDeleteAllKeys, ...) jsou potomci abstraktní třídy *Command*. Tato třída je dále potomkem abstraktních tříd *Serializable*, *ProcessableOnKmc* a *ProcessableOnKmacEntity*. Tyto třídy ovšem neimplementuje, ale nechává implementaci na konkrétních příkazech (svých potomcích). Poslední dvě zmiňované abstraktní třídy jsou popsány v kapitole 4.1.2. Abstraktní třída *Serializable*

obsahuje abstraktní metodu *toBytes*, která předepisuje převedení zprávy na pole bajtů pro přenos zprávy pomocí TLS protokolu. Implementace převodu na pole bajtů respektuje SUBSET 137 [11]. KMS Support obsahuje třídy, jejichž implementace dokáží převést pole bajtů zpět na KMS příkazy. Metody těchto tříd rovněž kontrolují platnost takovýchto převáděných dat. Převod objektu (KMS zprávu) na pole bajtů se využívá při požadavku na odeslání dané zprávy druhé straně komunikace. Opačný postup, tedy převod pole bajtů na KMS objekt, se využívají při příjmu zprávy od protějška.

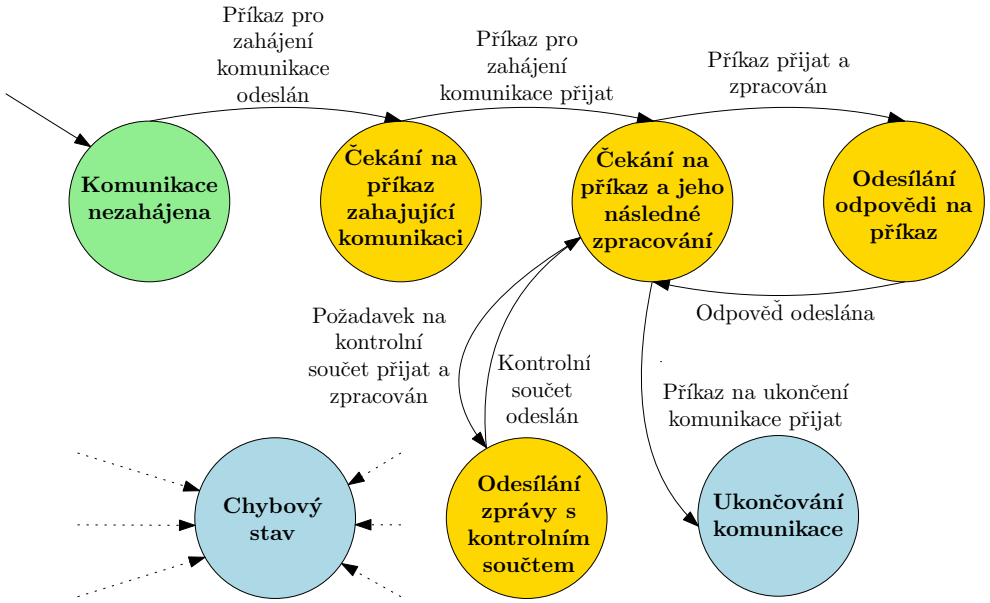
Další důležitou částí KMS support modulu je třída *CommunicationSupport*, jejíž instance zajišťují komunikaci na aplikační úrovni podle protokolu definovaného dokumentem SUBSET 137 [11]. Instance této třídy tak zajišťují přijímání a odeslání zpráv aplikačního KMS protokolu. Taktéž udržují sekvenční i transakční čísla a mimo to provádí kontrolu přijatých zpráv. Aby mohla být jakákoliv zpráva korektně přijata, tak musí být zkontrolováno, že daná zpráva má správného odesílatele i příjemce. Hodnoty sekvenčních a transakčních čísel jsou taktéž kontrolovány.

Jak již bylo zmíněno, tak metody třídy *CommunicationSupport* umožňují sestavit na základě parametrů příkazy KMS protokolu a tyto příkazy následně odeslat svému protějšku. Samotné odeslání probíhá pomocí TLS rozhraní, které je reprezentováno ukazatelem na objekt *SSL* knihovny OpenSSL. Metody pro přijímání a odesílání zpráv počítají s tím, že čtení a zápis do TLS rozhraní je blokující. Přijímání zpráv probíhá tím způsobem, že se nejdříve načte hlavička mající fixní velikost a následně se čte počet bajtů specifikovaný v hlavičce. V případě, že je přečteno méně bajtů, než je očekáváno, tak se čeká na zbylé bajty až do doby než nastane timeout. Hodnota timeoutu je nastavována taktéž instancemi této třídy na základě informací ze zprávy zahajující komunikaci na aplikační úrovni.

4.1.4 RequestHandlers

Účelem tohoto modulu je zajištění podpory pro obsluhu požadavků aplikačního protokolu KMS dle dokumentu SUBSET 137 [11]. Instance tříd v tomto modulu provádějí zmíněnou obsluhu na vyšší úrovni, než je zajišťováno instancemi třídy *CommunicationSupport*, která je popsaná v kapitole 4.1.3. Na nižší úrovni je zajišťováno samotné odeslání a přijímání zpráv (včetně kontroly daných zpráv). Vyšší úroveň naopak definuje chování entity v případě, že přijde konkrétní zpráva. Tato obsluha je implementována pomocí konečného automatu, jehož stavy udávají operaci, která se má nyní provést. Operace zahrnují odeslání konkrétní zprávy nebo vyčkávání na některou ze zpráv. Úspěšné i neúspěšné odeslání nebo obdržení zprávy znamená přesun do jiného stavu konečného automatu. V rámci obsluhy požadavků je nutné komunikovat i s databázemi entit. Tento modul tedy využívá spojení s TLS rozhraním i rozhraní pro komunikaci s databází.

Modul RequestHandlers obsahuje několik tříd pro obsluhu požadavků v závislosti na entitě, pro kterou jsou určeny. Není možné vytvořit pouze jedu logiku pro obsluhu požadavků, protože se každá entita chová jinak. KMAC entita (OBU, RBC/RIU) například neodesílá žádné požadavky na změnu databáze své protistraně (odesílá pouze odpovědi). Instance třídy *KmacEntityHandler* slouží pro obsluhu požadavků KMAC entit. Třídy *KmcRecipientHandler* a *KmcSenderHandler* slouží pro obsluhu požadavků na straně centra pro správu klíčů (KMC). Instance první z uvedených tříd se využije v případě, že KMC je v režimu odesílatele a zasílá tak příkazy do KMAC entit nebo do jiného KMC. Instance druhé z uvedených tříd slouží pro obsluhu požadavků na straně KMC v situaci, kdy je dané KMC v režimu příjemce a získává tedy příkazy od jiného KMC. Na obrázku 4.1 je graficky znázorněn automat pro obsluhu požadavků KMAC entity.



Obrázek 4.1: Automat zajišťující obsluhu požadavků

Důležitou součástí logiky obsluhy požadavků je funkcionalita pro ověřování stavu odeslaného příkazu v případě, že nebyla doručena potvrzující odpověď od protější strany. Tohoto je využito při komunikaci mezi KMC a KMAC entitou. Při této komunikaci může dojít k následujícímu scénáři. KMC úspěšně odešle KMAC entitě (například OBU) příkaz pro instalaci klíče. Ve své databázi si tak změní stav příkazu z *nového* na *v procesu instalace*. Poté co KMAC entita úspěšně daný příkaz přijme, tak dojde k chybě znamenající rozpad TLS spojení. V tomto okamžiku KMC neví, zdali byl příkaz na KMAC entitě úspěšně proveden. Z tohoto důvodu definuje aplikační protokol zprávy pro ověření stavu databáze. Stav databáze je ověřován pomocí kontrolního součtu (MD4) některých jejích částí. V případě, že KMC nemá potvrzený jeden pří-

kaz (v případě konzistentní databáze nemůže být pro jednu entitu více nepotvrzených příkazů), tak odešle KMAC entitě žádost o výpočet kontrolní sumy databáze. Výsledek poté porovná s kontrolní sumou předpokládající provedení příkazu a rovněž s kontrolní sumou předpokládající neprovedení příkazu. Oba předpoklady jsou následně porovnány s reálným stavem. Na základě výsledku porovnání je upraven stav příkazu. Důležité je zmínit, že do vyřešení nepotvrzeného příkazu nejsou dané entitě odesílány žádné další příkazy. Popisovaná funkcionalita je implementována ve třídě *KmcSenderHandler*.

4.1.5 Komunikační aplikace entit

Tato podkapitola se zabývá implementací konkrétních komunikačních aplikací, které jsou určeny pro jednotlivé entity. Dosud se kapitoly pojednávající o komunikační aplikaci zabývaly moduly (generickou verzí KA), které jsou využity v právě popisovaných konkrétních verzích. Jednotlivé verze se liší svými implementacemi, ale také rozdílnou konfigurací. Ta je uložena v textovém konfiguračním souboru a jeho záznamy mají podobu *klíč=hodnota*. Parser konfiguračních souborů je založen na ukázce [33]. Tento parser podporuje komentáře, ale také bílé znaky. Výňatek z konfiguračního souboru je ukázán na výpisu 4.2.

```
# Application timeout
applicationTimeout=45
# Defines ETCS_ID of this entity (HEX value)
entityEtcsId=AABBCC
# Defines the port on which KMS applications listen
port=7912
```

Výpis kódu 4.2: Ukázka části konfiguračního souboru

Ve výsledku existují tři typy komunikačních aplikací pro různé typy entit. První z nich je určena pro OBU (OnBoard Unit), druhá z nich je určena pro entity spojené s tratí (RBC a RIU). Posledním typem je aplikace centra pro správu klíčů (KMC). Verze pro OBU využívá modul TLS klienta pro komunikaci s jinými entitami. K zajištění přístupu do databáze entity se využívá podpora pro SQLite databáze. Druhá verze pro traťové části (RBC, RIU) se oproti verzi pro OBU liší tím, že místo TLS klienta je využit TLS server. Podpora pro SQLite databázi se využívá i zde. Obě právě popsané verze využívají instanci třídy *KmacEntityHandler* pro zajištění obsluhy požadavků. Komunikační aplikace využívá modul pro TLS klienta i pro TLS server. Pro komunikaci s databází se používají instance třídy *PostgreSQLDbSupport*. Instance tříd *KmcSenderHandler* a *KmcRecipientHandler* jsou využity pro obsluhu spojení.

4.2 Databáze centra pro správu klíčů (KMC)

Jak již bylo nastíněno v návrhu, tak úložiště na straně KMC musí být realizováno pomocí SQL databáze s podporou notifikací. Pro implementovaný prototyp byl zvolen databázový systém PostgreSQL, který poskytuje podporu notifikací pro své klienty. Dalším důvodem pro volbu tohoto databázového systému je to, že se jedná o open-source řešení.

V databázi se musí nacházet všechny tabulky uvedené v návrhu. Cílem této kapitoly je rozbor přístupových práv k uvedeným tabulkám. Databázový systém musí evidovat pro potřeby aplikací dva uživatelské účty se jmény *kmcWa* a *ComApp*. První z nich slouží pro potřeby webové aplikace centra pro správu klíčů. Tento uživatelský účet musí mít přístup k celé databázi pro zápis i pro čtení. Druhý uživatelský účet má sloužit pro komunikační aplikaci, která je popisována v kapitole 4.1. Tento účet musí mít povolený přístup k tabulkám *obecný příkaz*, *stav příkazu*, *výsledek příkazu* a rovněž ke *konkrétním příkazům (přidat klíč, odebrat klíč, změnit časovou platnost klíče, ...)*. Konfigurace databáze se skládá z několika kroků. Jednotlivé kroky i s jejich vysvětlením jsou součástí návodu na instalaci, kterým se zabývá kapitola 6.

4.2.1 CMP klient

Klient zajišťující komunikaci pomocí *Certificate Management Protokolu* je reprezentován třídou *CmpClient*. Při implementaci této třídy byla použita knihovna **generic CMP client**, která se nachází na [34]. Hlavičkové soubory zmíněné knihovny byly integrovány přímo do projektu komunikační aplikace. Pro jejich umístění byl zvolen adresář *libs* tak, aby bylo zřejmé, že se nejedná o soubory vytvořené v rámci této práce. Při sestavování projektu je nutné přilinkovat dynamickou knihovnu s implementací popisovaných hlavičkových souborů. Instance třídy *CmpClient* poskytují metody pro odeslání žádosti o nový certifikát. V závislosti na způsobu ověření se jedná o žádost o první certifikát (initial request) nebo o žádost o vytvoření certifikátu na základě již existujícího certifikátu (Key Update Request). Další podporovanou funkcionalitou popisovaného klienta je odesílání žádostí o revokaci existujících certifikátů. Klient se může vůči certifikační autoritě (zařízení poskytujíc CMP službu) ověřovat pomocí sdíleného hesla nebo pomocí certifikátů získaných od dané certifikační autority. Klient ověřuje CMP server s využitím jeho certifikátu. Popisovaný CMP klient umožňuje přenášet zprávy Certificate Management Protokolu pomocí TLS protokolu.

4.2.2 OCSP klient

Úkolem tohoto klienta je zajištění komunikace pomocí *Online Certificate Status Protokolu*, který slouží pro ověřování platnosti certifikátů. Popisovaný klient je reprezentován třídou *OcspClient*. Pro implementaci této třídy byla vy-

užita knihovna **OpenSSL**. Instance popisované třídy zajistí vytvoření nových OCSP požadavků, odeslání daných požadavků OCSP responderu a zpracování výsledku přijatého od responderu. Samotná OCSP komunikace je realizována nad HTTP protokolem. Adresa OCSP responderu je získávána přímo z ověřovaného certifikátu. Ověřování celé posloupnosti certifikátů pomocí OCSP protokolu je realizováno vždy před zahájením TLS komunikace, poté co komunikující protistrana poskytne svůj certifikát. Pokud byl kontrolovaný certifikát revokován nebo došlo k chybě během komunikace (OCSP responder je nedostupný), tak nedojde k zahájení TLS spojení.

4.3 Webová aplikace centra pro správu klíčů (KMC)

Implementace webové aplikace vychází z návrhu, který je uveden v kapitole 3.2. Pro realizaci prototypu byl zvolen programovací jazyk Python a jako podpora při tvorbě byl využit open-source webový aplikační framework Django. Tato softwarová podpora splňuje všechny požadavky, které jsou uvedeny v návrhu. Samotná webová aplikace je napsaná s ohledem na MVC (Model-View-Controller) architekturu. S dodržováním zmíněné architektury pomáhá framework Django, který jednotlivé komponenty pouze označuje odlišněji. Označení *Modelu* zůstává nezměněné. MVC *View* je v Djangu označeno jako *Template* a MVC *Controller* je v Djangu označen jako *View*. V následujících kapitolách bude využíváno označení používané frameworkem. Při tvorbě frontend části aplikace byl využit skriptovací jazyk JavaScript a pro definici způsobu vykreslení HTML elementů byly využity kaskádové styly (CSS).

4.3.1 Template

Jedná se o část aplikace, která je zodpovědná za generování obsahu pro uživatele. Tato vrstva je reprezentována HTML šablonami frameworku Django, kaskádovými styly a zdrojovými soubory jazyka JavaScript. Některé soubory této vrstvy jsou umístěny v adresáři *templates*, zbylé části, které nebyly vytvořeny v rámci této práce, jsou při načítání stránek stahovány. Mezi stahované soubory patří různé knihovny jazyka JavaScript nebo soubory kaskádových stylů. V adresáři *templates* se nacházejí HTML šablony pro stránky spravující klíče, domény nebo ETCS entity. Rovněž se zde nacházejí šablony pro stránky zajišťující registraci a další správu uživatelských účtů.

Pro definici způsobu zobrazení HTML stránek je využita CSS šablona [35]. Způsob použití zmíněné šablony je založen na příkladu, který je uveden na [36]. Pokud uživatel vytváří nový klíč nebo některý z již existujících klíčů upravuje, tak je nutné vybírat peery pro daný klíč ze seznamu. Pro tyto účely je využita knihovna **Select2** [37], protože kromě možnosti výběru několika prvků z listu také podporuje vyhledávání v daném seznamu. V některých pří-

padech je nutné zobrazit uživatelům zprávu (klíč byl vytvořen, klíč by odstraněn). Pro tyto účely je využit *The messages framework* softwarové podpory Django. Uživatelům je nutné tyto notifikace zobrazovat způsobem, který je pro ně přehledný. K tomuto účelu byly využity *Alerts* sady nástrojů **Bootstrap** [38]. Způsob použití notifikací je inspirován ukázkou [39]. Pro zobrazování velkých seznamů jsou využity tabulky umožňující stránkování nebo různá řazení svých sloupců. Pro tyto účely je využita knihovna **django-tables2**. Knihovna **django_filters** je využita pro vytvoření podpory pro filtrování takovýchto velkých seznamů.

4.3.2 View

Tato část zajišťuje propojení mezi výstupem pro uživatele a databází. V této části také probíhá část kontrol uživatelského vstupu, protože některé kontroly probíhají až na úrovni samotné databáze.

4.3.3 Model

Tato vrstva se stará o komunikaci s databází, ve které jsou uloženy domény, entity, KMAC klíče, příkazy a informace o registrovaných uživatelích. Model obsahuje třídy reprezentující uživatelské tabulky, které se nacházejí v databázi. Tyto třídy jsou potomky třídy *Model*, která je součástí frameworku Django. Pokud některá část aplikace potřebuje pracovat s databází, tak nepoužívá přímo SQL příkazy, ale objekty reprezentující jednotlivé tabulky. Ukázka práce s databází je uvedena na 4.3.

```
def create_add_key_command(receiver, kmac, cmd):
    add_key_command = AddKeyCommand.objects.create(
        command=cmd,
        recipient_ext_etcs_id=receiver,
        kmac_id_exp_identifier=kmac.kmac_identifier(),
        kmac_key_value=kmac.key_value,
        validity_period=kmac.key_valid_period(),
    )
    add_key_cmd.save()
    create_peer_records(receiver, kmac, add_key_cmd)
    return None
```

Výpis kódu 4.3: Vytváření nového příkazu pro fyzickou entitu (upraveno)

4.3.4 Důležité moduly

Součásti webové aplikace jsou Python moduly psané v rámci této práce. Jedná se například o moduly balíčku *command_management* pro práci s příkazy. Po vytvoření klíče (do databáze se vloží informace o daném klíči) se také musí vytvořit instalační příkaz pro fyzické zařízení. Právě tuto funkcionalitu provádějí instance třídy *CommandCreator* ze stejnojmenného modulu. Výpis kódu na 4.3 ukazuje vytvoření nového příkazu pro fyzickou entitu. Druhým zástupcem balíčku *command_management* je *command_importer*. Úkolem tohoto modulu je poskytovat možnosti pro import příkazů mezi doménové komunikace. V případě, že centru pro správu klíčů (KMC) přijde příkaz od KMC cizí domény, tak je potřeba z přijatého příkazu vytvořit příkaz pro fyzickou KMAC entitu vlastní domény. Právě o tuto funkcionalitu se stará třída *CommandImporter*.

Instance třídy *ValidatorHelpers* poskytují funkcionalitu pro kontrolu platnosti uživatelských vstupů. Konkrétně je možné ověřit, zdali není zadaný 3DES klíč *weak* nebo *semi-weak* a mimo to ověří zdali je na vstupu opravdu požadovaný 3DES klíč. Umožňují také ověřit správnost různých identifikátorů podle SUBSETu 137 [11]. Třída *KeyStatusInspector* stejnojmenného modulu umožňuje zkontrolovat aktuální stav klíče na fyzických entitách. Stav klíče se kontroluje, pokud si uživatel zobrazí detail klíče. Tento stav je vždy odvozen od posledního příkazu, který byl s daným klíčem prováděn.

Při vytváření nových klíčů je nutné, pokud uživatel nezadá klíč ručně, vygenerovat kryptograficky bezpečným náhodným generátorem samotnou hodnotu klíče. Tohoto je docíleno využitím knihovny **OpenSSL**, která poskytuje kryptograficky bezpečný generátor náhodných čísel. Z tohoto důvodu plyne skutečnost, že je nutné mít v operačním systému, na kterém bude spuštěna webová aplikace KMC, nainstalovanou knihovnu OpenSSL a správně nastavenou cestu k této knihovně. Funkcionalita generování náhodných čísel je poskytována instancemi tříd umístěných v modulu *random_number_providers*.

4.4 Certifikační autorita

V rámci vytvářeného prototypu systému on-line správy klíčů a hlavně pro účely testování byla využita open-source certifikační autorita EJBCA, která se nachází na [40]. Komunikační aplikace, která využívá služeb certifikační autority, není vázána pouze na jednu uvedenou autoritu, ale je možné použít jakoukoliv CA. Je samozřejmě nutné, aby použitá certifikační autorita splňovala požadavky uvedené v návrhu.

Certifikační autorita EJBCA podporuje CMP protokol pro správu certifikátů a rovněž také poskytuje služby OCSP responderu. Adresa responderu je vkládána touto certifikační autoritou přímo do vydávaných certifikátů, tudíž není nutné udržovat v rámci konfigurace komunikační aplikace adresu OCSP responderu. Popisovaná CA umožňuje pomocí registrační autority definovat

entity, které určující koncová zařízení (KMC, RBC/RIU, OBU). V rámci definice entity je nutné zadat i heslo, které se využije při první žádosti o certifikát pomocí CMP protokolu. Samotným koncovým entitám může být přiřazen profil, který usnadní jejich tvorbu.

Kapitola 5

Testování

Tato kapitola se zabývá testováním implementovaného systému on-line správy kryptografických KMAC klíčů vlakového zabezpečovacího zařízení ETCS. Testování je nezbytnou součástí vývoje každého softwaru, protože se snaží eliminovat chyby vniklé při implementaci aplikací. Důležité je poznamenat, že pokud aplikace nebo systém projde všemi testy bez chyb, tak z toho nevyplývá, že se v dané aplikaci nebo systému žádné chyby nevyskytují.

5.1 Průběh testování

Implementovaný systém se skládá z několika komponent, proto byl kladen důraz na otestování každé z nich. Testována byla rovněž i komunikace mezi jednotlivými komponentami. tato část zahrnovala také ověření chování aplikací v případě, že nastane chyba během komunikace. V implementovaném systému jsou použity některé aplikace nebo knihovy, které nebyly vytvářeny v rámci práce. U těchto externích řešení funkčnost testována nebyla, ale komunikace s těmito řešeními testována byla.

5.2 Automatické testování

Testování bylo zahájeno již v průběhu vývoje vytvořením automatizovaných testů pro komunikační aplikaci. Jako podpora při vytváření těchto testů byl použit testovací framework *GoogleTest*. Právě popisované automatické testy se zabývají ověřováním toho, jak se bude komunikační aplikace chovat v závislosti na jednotlivých zprávách protokolu dle SUBSETu 137 [11]. Pro každou zprávu je testována serializace a deserializace pro platný i neplatný vstup. Prováděné testy jsou součástí zdrojových kódů komunikační aplikace. Při jejím sestavování je možné v rámci CMake nástroje zvolit, zdali se budou dané automatické testy kompilovat a zda se bude vytvářet cíl reprezentující testo-

vací aplikaci. Komunikační aplikace, která je součástí příloh této práce projde všemi automatickými testy.

5.3 Manuální testování

Celý systém on-line správy klíčů byl dále testován manuálně podle různých testovacích scénářů. Tyto scénáře byly rozděleny do dvou hlavních kategorií. První kategorie se zabývala funkčností všech komponent a komunikací mezi nimi za předpokladu validního chování uživatelů, aplikací i sítě. Druhá kategorie testovacích scénářů se zabývá testem komponent a komunikací mezi nimi za předpokladu nevalidního chování uživatelů, sítě nebo aplikací. Pro testování bylo nutné vytvořit prostředí simulující celý systém obsahující několik zařízení. Toto prostředí je popisováno v kapitole 5.3.1. **První** kategorie, která obsahuje scénáře **validního** chování, zahrnuje následující:

- Vytvoření nového klíče spolu se seznamem peerů.
- Aktualizace časové platnosti klíče.
- Aktualizace seznamu peerů.
- Odstranění klíče.
- Odstranění všech klíčů z entity.
  - Entita, na kterou je příkaz směřován, je výchozí pro některý z klíčů.
  - Entita, na kterou je příkaz směřován, je peer pro některý z klíčů.
- Odeslání žádosti o vydání nebo úpravu klíče cizí doménou.
- Odeslání potvrzení o provedení operace s klíčem vydavatelské doméně.
- Zpracování příkazu na úpravu klíče od cizí domény.

Výše uvedené operace byly prováděny pomocí webové aplikace centra pro správu klíčů (KMC). V rámci této aplikace bylo sledováno, zdali se požadovaná změna opravdu provedla a také to, zda se vytvořily příslušné instalační příkazy pro koncové KMAC entity nebo pro jiná KMC. V rámci komunikačních aplikací byl ověřován přenos zpráv pomocí kontroly záznamů chování těchto aplikací. Po dokončení přenosu došlo k ověření změny přímo na koncové entitě. **Druhá** kategorie scénářů, která zahrnuje **nevalidní** chování uživatelů, sítě nebo aplikací, obsahuje následující:

- Vytvoření nového klíče s prázdným seznamem peerů.
- Nastavení chybné platnosti klíče.

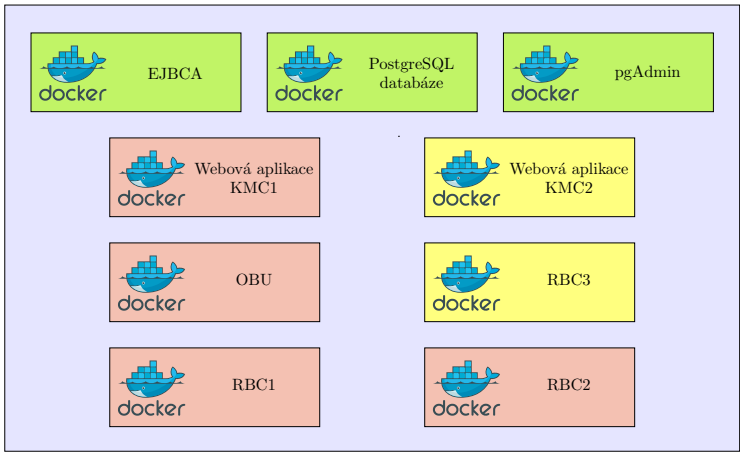
- Nepovolené odstranění všech klíčů z entity.
- – Entita, na kterou je příkaz směřován, je jediným peerem pro daný klíč.
  – Entita, na kterou je příkaz směřován, obsahuje klíč, který byl vydán cizí entitou.
- Snaha o druhou instalaci klíče se stejným sériovým číslem.
- Snaha o úpravu neznámého klíče na KMAC entitě.
- Ověření chování komunikační aplikace v případě, že nebylo doručeno potvrzení příkazu.
- Ověření chování komunikační aplikace v případě, že není dostupný CMP server.
- Ověření chování komunikační aplikace v případě nedostupnosti OCSP serveru.
- Ověření chování komunikační aplikace v případě, že není dostupná druhá strana komunikace.
- Ověření chování komunikační aplikace v případě, že se není možné spojit s databází.

U druhé kategorie scénářů bylo testováno chování komunikačních a webových aplikací v závislosti na simulované situaci. Výše uvedené scénáře (první i druhé skupiny) nejsou kompletní, ale slouží pro vytvoření přehledu o testování.

5.3.1 Testovací prostředí

Zmiňované testovací prostředí bylo vytvořeno pomocí *Docker* kontejnerů s tím, že jeden kontejner většinou odpovídal jednomu modulu (popisováno kapitolou 3). Použití kontejnerů zajistilo pro testovací prostředí síť nezávislou od sítě hostujícího počítače. V rámci testovacího prostředí byly využity dva kontejnery pro dvě webové aplikace centra pro správu klíčů (KMC). Bylo nutné nutné vytvořit dvě KMC, protože byla testována i komunikace mezi doménami. Další kontejnery sloužily pro PostgreSQL databázi KMC a pro administrační nástroj *pgAdmin*. V testovacím prostředí se rovněž nacházelo pět kontejnerů s operačním systémem Ubuntu 23.04. Tyto kontejnery sloužily pro komunikační aplikace (dvě KMC, dvě RBC a jedno OBU). Poslední kontejner sloužil pro potřeby certifikační autority EJBCA. Z důvodu velkého množství kontejnerů byl využit nástroj na spouštění a konfiguraci kontejnerů *docker-compose*. Konfigurační soubor zmíněného nástroje na správu kontejnerů je uložen v adresáři *Installation*, který je součástí příloh práce. V tomto adresáři se

rovněž nachází konfigurace některých kontejnerů a je zde taktéž archiv s daty nutnými pro první spuštění všech aplikací systému. Archiv s daty byl vytvořen proto, aby nebylo nutné celý systém neustále konfigurovat po jeho vytvoření. Na obrázku 5.1 je graficky znázorněno prostředí, které bylo využíváno pro testování systému. Ve fialovém obdélníku se nacházejí všechny využívané Docker kontejnery. Červeně označené kontejnery se vztahují k první doméně a žlutě označené kontejnery slouží pro druhou doménu. Kontejnery označené zelenou barvou poskytují služby pro aplikace obou zmíněných domén.



Obrázek 5.1: Testovací prostředí

5.4 Výsledky

Během testování aplikace bylo nalezeno několik chyb napříč různými komponentami. Tyto chyby byly následně odstraněny a dané testy byly zopakovány. Výsledné řešení, které je součástí této práce, prošlo všemi prováděnými testy.

Kapitola 6

Návod k instalaci a použití

Tato kapitola obsahuje kroky, které je nutné učinit k úspěšné instalaci systému pro on-line správy KMAC klíčů zabezpečovacího zařízení ETCS. Celý systém se skládá z několika aplikací, které jsou mezi sebou vzájemně propojeny. V následujících kapitolách bude uveden popis instalace podle komponent (viz. obrázek 3.1), v rámci kterých je dále nutné nainstalovat i jejich jednotlivé moduly. Ve většině případů platí, že se jeden modul rovná jedné aplikaci. Instalace samotných komponent by měla probíhat ve stejném pořadí, jako je uvedeno v tomto návodě. Tato kapitola zmiňuje instalaci každé komponenty pouze jednou, ale pro plně použitelný systém bude nutné nainstalovat některé komponenty vícekrát (typicky komponentu pro správu klíčů na KMAC entitě). Instalační manuál předpokládá, že každá komponenta bude běžet na samostatném zařízení s operačním systémem. Rovněž je předpokládáno, že všechny moduly (aplikace) jedné komponenty budou běžet na stejném zařízení.

6.1 Instalace centra pro správu klíčů (KMC)

V rámci centra pro správu klíčů bude popsán postup pro instalaci webové a komunikační aplikace a rovněž zde bude uveden postup pro konfiguraci databáze.

1. Požadavky

Pro instalaci KMC je nutné využít zařízení, na kterém je nainstalovaný operační systém (návod předpokládá systém založený na Debian GNU/Linux), webový server s podporou WSGI (například Apache HTTP Server) a interpret jazyka Python s podporou verze 3.10. Kromě zmíněného je nutné mít nainstalovaný databázový systém PostgreSQL v nejnovější verzi.

2. Instalace

Webovou aplikaci, která se nachází v adresáři *KeyManagementCentre* (příloha práce), není nutné nijak instalovat, ale pro její spuštění je nutné vytvořit nové virtuální prostředí jazyka Python, nainstalovat do něj potřebné balíčky a nastavit databázový server PostgreSQL. Rovněž je nutné propojit webový server pomocí WSGI rozhraní s webovou aplikací.

Následující seznam ukazuje kroky, pomocí kterých je možné vytvořit nové virtuální prostředí a nakonfigurovat databázi. Všechny níže uvedené cesty v rámci instalace **webové aplikace** jsou relativní k adresáři *KeyManagementCentre/DjangoWebApp* přílohy práce.

- Nové virtuální Python prostředí (jménem *kmcEnv*) je možné v aktuálním adresáři vytvořit pomocí:

  ```
  python -m venv kmcEnv
  ```

- Vytvořené prostřední se aktivuje pomocí příkazu (nutné spouště ve stejném adresáři jako předchozí příkaz):

  ```
  source kmcEnv/bin/activate
  ```

- Potřebné balíčky pro webovou aplikaci KMC lze do aktivovaného Python prostřední nainstalovat pomocí:

  ```
  pip install -r requirements.txt
  ```

- V konfiguračním souboru *DjangoWebApp/settings.py* je nutné v sekci *DATABASES/default* nastavit informace o používané PostgreSQL databázi. Je potřeba upravit *NAME*, *HOST*, *PORT*, *USER* a *PASSWORD*. Je doporučeno použít uživatele *webApp*.

- Po přihlášení k databázovému serveru je nutné vytvořit dva uživatele (jednoho pro webovou aplikaci, druhého pro komunikační aplikaci). Je doporučeno ponechat názvy účtů stejné, jako je uvedeno v následujícím příkladě. Hesla zmíněných uživatelů je ale **nutné** změnit.

  ```
  CREATE USER "webApp" WITH PASSWORD 'heslo';

  CREATE USER "comApp" WITH PASSWORD 'heslo';
  ```

- V databázovém serveru je potřeba vytvořit novou databázi pomocí:

  ```
  CREATE DATABASE "KMC";
  ```

- Další kroky je nutné provádět v rámci právě vytvořené databáze.

- Pomocí následujícího příkazu se přidá uživateli oprávnění na vytváření nových tabulek v databázi:

  ```
  GRANT CREATE ON SCHEMA public TO "webApp";
  ```

- Pomocí níže uvedeného příkazu se vytvoří v databázi všechny potřebné tabulky. Příkaz je nutné spouštět v rámci operačního systému (není to tedy SQL příkaz pro databázový server). Rovněž je nutné využívat aktivované Python virtuální prostředí z prvního bodu.

  ```
  python manage.py migrate
  ```

- Dále je nutné vytvořit superuživatele, který bude prvním uživatelem webové aplikace. Tento uživatel bude moci následně již pomocí webové aplikace vytvářet další účty. Superuživatele je možné vytvořit pomocí (opět nutné spouštět v prostřední operačního systému):

  ```
  python manage.py createsuperuser
  ```

- SQL příkazy ve výpisu kódu 6.1 nastavují oprávnění jednotlivým uživatelům pro přístup k databázi a jejím tabulkám.

- Dále je nutné v rámci databázového serveru pomocí příkazů z výpisu kód 6.2 přidat triggery, které bude posílat notifikace při změněně dat.

```
REVOKE CREATE ON SCHEMA public FROM "webApp";
REVOKE ALL ON SCHEMA public FROM "comApp";
GRANT CONNECT ON DATABASE "KMC" TO "comApp";
GRANT SELECT, INSERT, UPDATE ON "command_status",
↪  "command_result" TO "comApp";
GRANT SELECT, INSERT, UPDATE ON "command",
↪  "add_key_command", "peers_of_add_key_command",
↪  "remove_key_command", "update_key_validity_command",
↪  "update_key_entities_command",
↪  "peers_of_update_entities_command",
↪  "requests_key_operation_command",
↪  "notif_key_update_status_command" TO "comApp";
```

Výpis kódu 6.1: Úprava oprávnění pro uživatele databáze.

V rámci nastavení webového serveru (například Apache) je nutné vytvořit nový *VirtualHost*, který je následně nutné povolit. U takto vytvořeného virtuálního hosta je nutné definovat cestu k *DjangoWebApp/wsgi.py* souboru a také se musí nastavit cesta k virtuálnímu Python prostředí. V případě webového serveru Apache je nutné nainstalovat modul mod_wsgi pro podporu WSGI.

Další důležitou aplikací komponenty KMC je komunikační aplikace, tedy **KMSCommunicator**. Instalační proces této aplikace bude uveden v kapitole 6.3. Bez instalace správného typu tohoto nástroje nebude systém KMC plně funkční. Je nutné poznamenat, že je nutné nastavit **KMSCommunicator**

```
CREATE OR REPLACE FUNCTION notify_command_update()
↪  RETURNS TRIGGER AS $$
    BEGIN

    PERFORM pg_notify('command_updated', 'notify');
    RETURN NULL;

    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_command_updated
  AFTER INSERT OR UPDATE
  ON command
  FOR EACH ROW
  EXECUTE PROCEDURE notify_command_update();

CREATE TRIGGER trigger_status_updated
  AFTER UPDATE OF last_status
  ON command_status
  FOR EACH ROW
  WHEN ((OLD.last_status = 'FAILED' AND NEW.last_status
  ↪  = 'NEW') OR
        (OLD.last_status = 'CANCELED' AND
        ↪  NEW.last_status = 'NEW') OR
        (OLD.last_status = 'SUCCESSFUL' AND
        ↪  NEW.last_status = 'NEW'))
  EXECUTE FUNCTION notify_command_update();
```

Výpis kódu 6.2: Přidání triggeru pro zasílání notifikací.

pro KMC tak, aby tato aplikace využívala PostgreSQL databázi, která byla nastavena v rámci předchozího kroku.

6.2 Instalace správce klíčů KMAC entity

V rámci komponenty správce klíčů KMAC entity bude popsána instalace komunikační aplikace a rovněž zde bude uvedena konfigurace SQLite databáze pro ukládání samotných klíčů.

1. Požadavky

Pro instalaci správce klíčů KMAC entity je nutné využít zařízení, na kterém je nainstalovaný operační systém (návod předpokládá systém založený na Debian GNU/Linux). Na tomto systému je nutné mít nainstalované dynamické knihovny požadované komunikační aplikací.

2. Instalace

Pro správnou funkčnost této komponenty je nutné nainstalovat komunikační aplikaci. Návod na její instalaci je uveden v kapitole 6.3. Komunikační aplikace potřebuje pro svou činnost databázi, ve které budou uchovávány přijaté klíče. K tomuto účelu slouží SQLite databáze, která je reprezentována klasickým souborem. Je důležité nastavit oprávnění v rámci operačního systému tak, aby k tomuto souboru měl přístup pouze uživatel, pod kterým daná komunikační aplikace běží. Pro vytvoření databázového souboru je možné využít přiložený SQL dump, který se nachází ve složce *Installation* pod jménem *KmacEntityDbDump.sql*. V rámci konfigurační souboru komunikační aplikace je nutné nastavit cestu k databází. Je důležité zmínit, že instalace SQLite databáze se týká pouze komponenty *Správce klíčů KMAC entity*. Pro potřeby KMC se využívá PostgreSQL databáze.

6.3 Instalace komunikační aplikace

Tato kapitola obsahuje pokyny pro instalaci komunikační aplikace, kterou je nutné použít v rámci centra pro správu klíčů (KMC) a v rámci správce klíčů KMAC entity.

1. Požadavky

V rámci operačního systému, na kterém bude komunikační aplikace běžet, je nutné mít nainstalované dynamické knihovny **OpenSSL** alespoň ve verzi 3.0, **libpqxx** a **sqlite3**. Tyto knihovny jsou často poskytovány správci balíčků Linuxových distribucí. Rovněž se předpokládá, že jsou na daném systému nainstalovány nástroje *CMake*, *make* a kompilátor jazyka C++ s podporou standardu 20 (kompilace testována na g++ (GCC) 12.2.1). Tyto tři zmíněné nástroje jsou nutné pouze pro překlad aplikace. Dále je nutné pro sestavení a i samotný běh aplikace zkompilovat knihovny **libsecutils**, **libgencmp** a **libcmp**. Tyto knihovny jsou dostupné z [34].

2. Instalace

V rámci instalace komunikační aplikace je nutné tuto aplikaci přeložit pro cílovou platformu. Zdrojový kód komunikační aplikace se nachází v adresáři

KMSCommunicator. Před sestavováním je nutné definovat, zdali se bude sestavovat aplikace pro KMC, OBU nebo RBC/RIU. Toto lze provést pomocí voleb *BUILD_OBU*, *BUILD_KMC*, *BUILD_TRACKSITE*. Pro kompilaci a sestavení komunikační aplikace je možné použít nástroj CMake. Rovněž je nutné při linkování přidat dynamické knihovny **libsecutils**, **libgencmp** a **libcmp**, které je možné získat z [34]. V případě použití nástroje Cmake pro kompilaci komunikační aplikace musí být tyto dynamické knihovny umístěny v adresáři *libs*. Z důvodu toho, že dané dynamické knihovny nejsou součástí operačního systém a ani nebyly instalovány v rámci správce balíčků, je nutné pro spuštění komunikační aplikace zmíněné knihovny vložit do umístění, ve kterém operační systém hledá dynamické knihovny.

Po dokončení kompilace komunikační aplikace je nutné provést úpravu konfiguračního souboru (je nutné vybrat si správný typ konfiguračního souboru v závislosti na druhu entity). Význam jednotlivých bodů konfiguračního souboru je vysvětlen komentáři nad každým z bodů přímo v souboru. Komunikační aplikace předpokládá, že se konfigurační soubor nachází v adresáři, ze kterého je aplikace spouštěna.

6.4 Instalace certifikační autority

Jedná se o komponentu, která bude poskytovat služby pro správu certifikátů infrastruktury veřejného klíče. Certifikační autorita je povinnou komponentu on-line systému pro správu klíčů (PKI). Tato aplikace není součástí příloh práce, protože je možné využít jakékoliv dostupné řešení. Je však nutné využít certifikační autoritu, která podporuje *Certificate Management Protocol* a *Online Certificate Status Protocol*. Je doporučeno využít certifikační autoritu EJBCA, protože implementovaný systém byl s touto autoritou testován.

6.5 Práce s testovacím prostředím

Jak již bylo uvedeno v kapitole 5.3.1, tak součástí příloh této práce je konfigurační soubor nástroje *docker-compose*, který spolu s přiloženými daty umožňuje vytvořit testovací prostředí systému pro on-line správu KMAC klíčů. Tato kapitola si klade za cíl popsat způsob práce s dodaným testovacím prostředím.

Všechny potřebné soubory testovacího prostředí se nacházejí v adresáři *Installation/DockerContainers*. Potřebná data pro provoz systému se nachází v archivu *data.zip*, který je nutné rozbalit do stejného adresáře, ve kterém se archiv nachází. Následně je nutné spustit všechny kontejnery pomocí:

```
docker-compose up --build
```

Spouštění všech kontejnerů může trvat i několik minut. Pokud jsou všechny kontejnery spuštěné, tak je možné začít využívat jejich služeb. Detailnější informace o jednotlivých službách jsou uvedeny v příloze této práce.

Z hosta je možné využívat:

- Webovou aplikaci KMC1.
- Webové aplikaci KMC2.
- Webové rozhraní CA EJBCA.
- Administračnímu rozhraní PostgreSQL
- Komunikačním aplikace KMS entit

Je nutné zmínit, že se jedná pouze o testovací prostředí, které není určené pro reálné nasazení. Komunikace s webovou aplikací KMC probíhá pomocí HTTP protokolu, ač by tato komunikace měla probíhat pomocí HTTPS protokolu. Rovněž jsou použita jednoduchá hesla, která nejsou bezpečná pro běžné použití.

6.6 Používání aplikací systému

Uživatelé budou primárně využívat webovou aplikaci centra pro správu klíčů (KMC). Pomocí této aplikace budou moci generovat klíče, odesílat je jednotlivým zařízení a upravovat metadata o klíčích. Detailnější informace o používání webové aplikaci KMC se nacházejí v kapitole 2.

Předchozí odstavec předpokládá, že jsou všechny entity systému již nainstalovány a správně nastaveny. Instalace již byla probrána výše v této kapitole. Tento odstavec se bude zabývat činnostmi, které je nutné učinit po instalaci všech komponent před zahájením jejich komunikace. Před vytvořením spojení s entitami je pro každou nově vytvořenou entitu nutné vytvořit DNS záznam. Doménové jméno entity je ve tvaru *idxxxxx.tyzz.etcs*, kde *xxxxx* je unikátní ETCS ID a *zz* je typ entity (00, 01, 02, 05). Po vytvoření DNS záznamu je nutné koncovou entitu přidat do záznamů certifikační autority. Zde je důležité vyplnit doménové jméno entity a heslo, kterým se bude koncová entita prokazovat CA. Bez záznamu o entitě nebude moci komunikační aplikace požádat pomocí CMP protokolu o certifikát. V důsledku čehož dojde k jejímu ukončení.

Závěr

Cílem této práce bylo navrhnout a implementovat prototyp systému pro on-line správu kryptografických klíčů vlakového zabezpečovacího zařízení ETCS. Bylo tak tedy nutné vytvořit systém, který umožní přenos symetrických klíčů ze zařízení zajišťující správu klíčů (umožňuje generování, ...) do koncových entit, které využívají přijaté klíče pro svou činnosti. Řešení mělo být v souladu s dokumentem ERTMS/ETCS SUBSET 137 pro zajištění kompatibility.

Před návrhem a implementací celého systému byla provedena analýza, ve které bylo popsáno samotné vlakové zabezpečovací zařízení ETCS s důrazem na využití symetrických klíčů v rámci činnosti celého systému. Tato kapitola rovněž popisovala i alternativu k on-line správě klíčů, tedy off-line správu klíčů. Na základě analýzy dokumentu SUBSET 137 byly vytyčeny funkční a nefunkční požadavky kladené na výsledné řešení.

Práce dále pokračovaly návrhem řešení. Celý systém byl rozdělen do několika vzájemně propojených komponent. Jednou z důležitých komponent bylo centrum pro správu klíčů, které má zajistit generování klíčů, odesílání příkazů a rovněž má také zajistit úpravu metadat o klíčích. Další komponentou je správce klíčů na koncové entitě. Tato komponenta má uchovávat klíče na entitách, které je budou vyžívat ke své činnosti. Další důležitou komponentou je certifikační autorita, která má poskytovat certifikáty infrastruktury veřejného klíče pro ověřování jednotlivých zařízení. Právě popisované komponenty byly ještě rozděleny do několika menších modulů. Až na některé výjimky platí, že jeden modul odpovídá jedné aplikaci.

Na základě návrhu byla provedena implementace systému, v rámci které byl v závislosti na aplikaci využit programovací jazyk C++ a skriptovací jazyk Python. Pro ukládání dat byly využity databáze PostgreSQL a SQLite. Takto implementovaný systém byl nakonec otestován.

Během psaní této práce se autor setkal s mnoha problémy. Za zmínku například stojí počáteční problémy při tvorbě TLS klienta a serveru. Další výzvou při implementaci byl *Certificate Management Protocol*, protože se jedná o velmi málo rozšířený protokol. Většina z problémů byla úspěšně vyřešena.

Bibliografie

1. ERTMS/ETCS. In: [online]. AŽD Praha s.r.o., 2022 [cit. 2022-09-09]. Dostupné z: https://www.azd.cz/cs/o-spolecnosti/ertmsetcs.
2. ERTMS. In: *What is ERTMS about?* [online]. European Commission, 2022 [cit. 2022-09-13]. Dostupné z: https://transport.ec.europa.eu/transport-modes/rail/ertms_en.
3. ERTMS in brief. In: [online]. UNIFE, 2022 [cit. 2022-09-09]. Dostupné z: https://www.ertms.net/about-ertms/ertms-in-brief/.
4. GAŠPAŘÍK, Jozef; KOLÁŘ, Jiří. *Železniční doprava: technologie, řízení, grafikony a dalších 100 zajímavostí*. Grada Publishing, 2017. ISBN 978-80-271-0058-3.
5. Mobility and Transport. In: *ETCS Levels and Modes* [online]. European Commission, 2022 [cit. 2022-09-09]. Dostupné z: https://transport.ec.europa.eu/transport-modes/rail/ertms/how-does-it-work/etcs-levels-and-modes_en.
6. KONOPÁČ, Ing. Tomáš. Zavádění ETCS v ČR – aktuální stav. In: [online]. ZČU Plzeň, 2022 [cit. 2022-09-10]. Dostupné z: https://www.fel.zcu.cz/rest/cmis/document/workspace://SpacesStore/27106a3e-a50d-404d-9b5b-fc960229df8b;1.0/content.
7. DLABAJA, Jiří. Pilotní projekt ETCS v České republice dokončen. In: [online]. AŽD Praha s.r.o., 2022 [cit. 2022-09-10]. Dostupné z: https://www.azd.cz/cs/historie-aktualit/pilotni-projekt-etcs-v-ceske-republice-dokoncen.
8. Národní implementační plán ERTMS. In: [online]. Ministerstvo dopravy, 2017 [cit. 2022-09-10]. Dostupné z: https://www.mdcr.cz/getattachment/Dokumenty/Drazni-doprava/Evropska-unie-na-zeleznici/Evropska-unie-na-zeleznici/NIP-ERTMS-2017.pdf.aspx?lang=cs-CZ.

9. Důležitý krok k bezpečnější železnici: Do provozu může první lokomotiva zpětně vybavená ETCS. In: [online]. Ministerstvo dopravy, 2022 [cit. 2022-09-10]. Dostupné z: https://www.mdcr.cz/Media/Media-a-tiskove-zpravy/Dalsi-milnik-v-zavadeni-evropskeho-zabezpecovace.
10. Subsystems and Constituents of the ERTMS. In: [online]. European Commission, 2022 [cit. 2022-09-11]. Dostupné z: https://transport.ec.europa.eu/transport-modes/rail/ertms/how-does-it-work/subsystems-and-constituents-ertms_en.
11. UNISIG. *SUBSET-137 - On-line Key Management FFFIS*. 2015. Tech. zpr., verze: 1.0.0. European Union Agency for Railways. Dostupné také z: https://web.archive.org/web/20230429045058/https://www.era.europa.eu/system/files/2023-01/sos3_index083_-_subset-137_v100.pdf.
12. UNISIG. *SUBSET-114 - KMC-ETCS Entity Off-line KM FIS*. 2015. Tech. zpr., verze: 1.1.0. European Union Agency for Railways. Dostupné také z: https://web.archive.org/web/20230429060338/https://www.era.europa.eu/system/files/2023-01/sos3_index079_-_subset-114_v110.pdf.
13. VRIES, Marten de. ETCS driver machine interface (DMI) with screen areas. In: [online]. 2016 [cit. 2022-09-14]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/0/08/DMI_ETCS-areas.svg.
14. SNÁŠEL, Jaroslav. Technologie GSM-R: mobilní sítě ve službách železnice. In: [online]. MobilMania.cz, 2006 [cit. 2022-09-16]. Dostupné z: https://mobilmania.zive.cz/clanky/technologie-gsm-r-mobilni-site-ve-sluzbach-zeleznice/sc-3-a-1112292/default.aspx.
15. EIRENE SRN. GSM-R System Requirements Specification. In: 2015. Č. verze: 16.0.0. Dostupné také z: https://web.archive.org/web/20230429062329/https://www.era.europa.eu/system/files/2023-01/sos3_index033_-_eirene_srs_v1600.pdf.
16. UNISIG. *SUBSET-037 - EuroRadio FIS*. 2015. Tech. zpr., verze: 3.2.0. European Union Agency for Railways. Dostupné také z: https://web.archive.org/web/20230429061156/https://www.era.europa.eu/system/files/2023-01/sos3_index010_-_subset-037_v320.pdf.
17. UNISIG. *SUBSET-38 - Off-line Key Management FIS*. 2015. Tech. zpr., verze: 3.1.0. European Union Agency for Railways. Dostupné také z: https://web.archive.org/web/20230429061005/https://www.era.europa.eu/system/files/2023-01/sos3_index011_-_subset-038_v310.pdf.
18. SANTESSON, Stefan; HALLAM-BAKER, Phillip. *Online Certificate Status Protocol Algorithm Agility* [RFC 6277]. RFC Editor, 2011. Request for Comments, č. 6277. Dostupné z DOI: 10.17487/RFC6277.

19. GALPERIN, Slava; ADAMS, Dr. Carlisle; MYERS, Michael; ANKNEY, Rich; MALPANI, Ambarish N. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP* [RFC 2560]. RFC Editor, 1999. Request for Comments, č. 2560. Dostupné z DOI: 10.17487/RFC2560.

20. FOX, Pamela. Transmission Control Protocol (TCP). In: [online]. Khan Academy, 2022 [cit. 2022-10-06]. Dostupné z: https://en.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/transmission-control-protocol--tcp.

21. FESL, Jan. Počítačové sítě. In: *6. přednáška - Transportní vrstva, protokoly TCP a UDP* [online]. 2021 [cit. 2022-10-06]. Dostupné z: https://courses.fit.cvut.cz/BI-PSI/media/lectures/P_6.pdf.

22. TLS Basics. In: [online]. Internet Society, 2022 [cit. 2022-10-06]. Dostupné z: https://www.internetsociety.org/deploy360/tls/basics/.

23. DOSTÁL, Jiří. Síťová a systémová bezpečnost. In: *Šifrované síťové protokoly (TLS)* [online]. 2021 [cit. 2022-10-06]. Dostupné z: https://courses.fit.cvut.cz/BI-SSB/media/lectures/new/BI-SSB_02_Sifrovane_protokoly_TLS.pdf.

24. Transport Layer Security (TLS) Protocol Overview. In: [online]. Oracle, 2022 [cit. 2022-10-06]. Dostupné z: https://docs.oracle.com/en/java/javase/17/security/transport-layer-security-tls-protocol-overview.html#GUID-69ECD56C-3B20-47F4-AEF0-A06EFA13A61D.

25. LÓRENCZ, Róbert. Bezpečnost. In: *11. Infrastruktura veřejného klíče* [online]. 2020 [cit. 2022-10-02]. Dostupné z: https://courses.fit.cvut.cz/BI-BEZ/media/lectures/bez11.pdf.

26. MONONEN, Tero; KAUSE, Tomi; FARRELL, Stephen; ADAMS, Dr. Carlisle. *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)* [RFC 4210]. RFC Editor, 2005. Request for Comments, č. 4210. Dostupné z DOI: 10.17487/RFC4210.

27. KAUSE, Tomi; PEYLO, Martin. *Internet X.509 Public Key Infrastructure – HTTP Transfer for the Certificate Management Protocol (CMP)* [RFC 6712]. RFC Editor, 2012. Request for Comments, č. 6712. Dostupné z DOI: 10.17487/RFC6712.

28. THE OPENSSL PROJECT AUTHORS. openssl-cmp. In: [online]. 2022 [cit. 2022-10-21]. Dostupné z: https://www.openssl.org/docs/man3.0/man1/openssl-cmp.html.

29. AWATI, Rahul; COBB, Michael. certificate revocation list (CRL). In: [online]. 2021 [cit. 2022-10-04]. Dostupné z: https://www.techtarget.com/searchsecurity/definition/Certificate-Revocation-List.

30. SANTESSON, Stefan; MYERS, Michael; ANKNEY, Rich; MALPANI, Ambarish; GALPERIN, Slava; ADAMS, Dr. Carlisle. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP* [RFC 6960]. RFC Editor, 2013. Request for Comments, č. 6960. Dostupné z DOI: 10.17487/RFC6960.

31. MLEJNEK, Jiří. Před. 6 - Architektonické vzory. In: *BI-SI1* [online]. 2022 [cit. 2022-11-25]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/532111/mod_resource/content/3/06.prednaska.pdf.

32. THE OPENSSL PROJECT AUTHORS. *OpenSSL - RAND_bytes* [online]. 2021. [cit. 2022-11-18]. Dostupné z: https://www.openssl.org/docs/man3.0/man3/RAND_bytes.html.

33. How to parse a simple config file. In: [online]. Walletfox.com, 2022 [cit. 2023-03-26]. Dostupné z: https://www.walletfox.com/course/parseconfigfile.php.

34. DAVID VON OHEIMB; ANDREAS KRETSCHMER. *generic CMP client*. [B.r.]. Dostupné také z: https://github.com/siemens/gencmpclient.

35. EGIL, Jan; REFSNES, Borge. W3.CSS 4.15. In: [online]. 2020 [cit. 2023-03-28]. Dostupné z: https://www.w3schools.com/w3css/4/w3.css.

36. W3Schools Tryit Editor. In: [online]. W3Schools, 2023 [cit. 2023-03-28]. Dostupné z: https://www.w3schools.com/w3css/tryit.asp?filename=tryw3css_templates_fifty&stacked=h.

37. BROWN, Kevin. Select2. In: [online]. 2023 [cit. 2023-03-28]. Dostupné z: https://select2.org/.

38. Bootstrap: Alerts. In: [online]. Bootstrap team, 2023 [cit. 2023-03-28]. Dostupné z: https://getbootstrap.com/docs/4.5/components/alerts/.

39. JAYSHA. How to use Django Messages Framework. In: [online]. 2020 [cit. 2023-03-28]. Dostupné z: https://ordinarycoders.com/blog/article/django-messages-framework.

40. PRIMEKEY SOLUTIONS AB. *EJBCA*. [B.r.]. CE 7.11.0. Dostupné také z: https://www.ejbca.org/.

Příloha A

Seznam použitých zkratek

CA Certifikační autorita

CRL Certificate Revocation List

CMP Certificate Management Protocol

DMI Driver Machine Interface

ECB Electronic codebook

ERTMS European Rail Traffic Management System

ETCS European Train Control System

EVC Euro Vital Compute

KA Komunikační aplikace

KM Key Management

KMC Key Management Centre

KMS Key Management System

MAC Message authentication code

OBU On-Board Unit

OCSP Online Certificate Status Protocol

PKI Public key infrastructure

RBC Radio Block Centre

TCP Transmission Control Protocol

TLS Transport Layer Security

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF