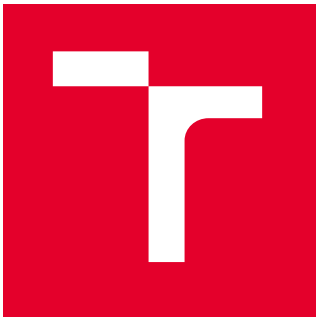


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

ZABEZPEČENÝ PŘENOS V RÁMCI PROTOKOLU CAN/CANOPEN

SECURED TRANSMISATION WITHIN PROTOCOL CAN/CANOPEN

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Josef Vlach**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Radek Fujdiak, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Josef Vlach ***ID:*** 221583

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Zabezpečený přenos v rámci protokolu CAN/CANOpen

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit demonstrátor pro zabezpečení komunikační systém s protokoly CAN/CANOpen, využívající protokolové nadstavby Secure CAN resp. Secure CANOpen. V první fázi bude vypracována analýza zdrojů, rozbor vektorů útoku a realizována rešerše vědecko-technického stavu v oboru, včetně dostupných open-source knihoven, ale i další nezbytných prvků pro řešení práce. Následně proběhne teoretické srovnání a výběr komponent pro uvažované zabezpečení a to na základě zvolených relevantních parametrů vůči hrozbám a možnostem protokolu. Tento teoretický výběr bude základem pro vysoko-úrovňový (architektonický) a nízko-úrovňový (detailní) návrh predikčního modelu. Poté bude provedena formální verifikace návrhu, otestování komponent a implementace. Dále bude vypracována sada testovacích scénářů s jasně definovanými cíli pro následné experimentální ověření pomocí převážně výkonnostních a bezpečnostních testů. V závěru bude provedena optimalizace funkčních parametrů a v neposlední řadě finální validace, umožňující zhodnocení dosažených výsledků oproti požadavkům.

DOPORUČENÁ LITERATURA:

[1] FARAG, Wael A. CANTrack: Enhancing automotive CAN bus security using intuitive encryption algorithms. In: 2017 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO). IEEE, 2017. p. 1-5.

[2] BOZDAL, Mehmet, et al. Evaluation of can bus security challenges. Sensors, 2020, 20.8: 2364.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: Ing. Radek Fujdiak, Ph.D.

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

ABSTRAKT

Tato bakalářská práce se zabývá problematikou využiti protokolu CAN a jeho vyšších nástaveb, konkrétně CANopen v dnešní době. Cílem této práce je analýza a průzkum bezpečnostní stránky tohoto protokolu. Jsou zde zpracovány analýzy aktuálně dostupných řešení, které je možné využít pro zajištění bezpečnosti protokolu, sběrnice, či zabezpečení periferii. Hlavním přínosem je analýza a návrh zabezpečení pro protokol CAN s využitím TLS s podporou otevřené krypto knihovny wolfSSL. Demonstrativní návrh představuje stanu klienta a serveru, kteří jsou navzájem ověřeni pomoci digitálního certifikátu a mají zajištěnou šifrovanou komunikaci.

KLÍČOVÁ SLOVA

CAN, CANbus, CANopen, TLS, wolfSSL, šifrování, autentizace.

ABSTRACT

This bachelor thesis deals with the use of the CAN protocol and its higher extensions, specifically CANopen in today's time. The aim of this thesis is to analyse and investigate the security aspects of this protocol. Analyses of currently available solutions that can be used to ensure the security of the protocol, bus or peripheral security are presented. The main contribution is the analysis and design of security for the CAN protocol using TLS with the support of the open crypto library wolfSSL. The demonstrative design represents a client and server state that are authenticated against each other using a digital certificate and have encrypted communication secured.

KEYWORDS

CAN, CANbus, CANopen, TLS, wolfSSL, encryption, authentication.

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

VLACH, Josef. *Zabezpečený přenos v rámci protokolu CAN/CANopen*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 62 s. Bakalářská práce. Vedoucí práce: Ing. Radek Fujdiak, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Josef Vlach
VUT ID autora:	221583
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	Zabezpečený přenos v rámci protokolu CAN/CANopen

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Radek Fujdiak, Ph.D za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Také bych chtěl poděkovat všem pracovníkům zahraničních firem, se kterými byla v průběhu práce navázána komunikace za cenné rady a informace k této problematice. Také bych chtěl poděkovat své drahé polovičce, Kláře Palánové za obrovskou psychickou podporu a pomoc při korektuře některých částí práce. V neposlední řadě bych chtěl také poděkovat všem učitelům, profesorům, rodině, zmámím a přátelům, díky kterým jsem se mohl dostat k tomu že vůbec píšu tyto řádky. Také bych chtěl zmínit poděkování OpenAI za vyvinutí produktu chatového modulu umělé inteligence, ChatGPT, který byl v rámci práce využit pro základní pochopení kontextů problematiky, ve které jsem se neorientoval, či při pomoci s připomenutím příkazů pro operační systém Linux.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Protokol CAN (Controller Area Network) je komunikační protokol vyvinutý před více než čtyřmi desetiletími na konci minulého tisíciletí, který své hlavní uplatnění nachází v automobilovém, leteckém a lodním průmyslu, pro vytváření vnitřní sítě propojující elektronické jednoty a kontrolní systémy. Ačkoliv byl původně navržen pro jednoduchou komunikaci mezi elektronickými jednotkami, v dnešní době je už možné nalézt tento protokol na více místech. Díky jeho vlastnostem roste jeho popularita a využití. V dnešní době nachází tento protokol a jeho nástavby uplatnění i v průmyslu, nemocnicích, letištích, odvětvích automatizace a v inteligentních budovách. S rostoucí popularitou rostou také rizika útoků vůči této sběrnici. Jelikož sběrnice v základu neimplementuje prakticky žádné prvky zabezpečení, je nutné se na tuto problematiku do budoucna zaměřit. V posledních letech se začaly objevovat rozšíření a nástavby, které mají za cíl posílit bezpečnost tohoto protokolu. A proto je v dnešní době více různých firem a uskupení, které se této problematice věnují. A tak mnoho firem přichází s různými vylepšeními a rozšířeními, které zlepšují fungování protokolu jako takového. Také přináší rozšíření či hardwarové komponenty, které umožňují lepší práci či zabezpečení připojených a rozšiřujících systémů. Jelikož v dnešní době existuje nepřeberné množství bezpečnostních řešení a tato problematika není nikterak řešena standardizovanou cestou. Je cílem této práce analyzovat současný stav bezpečnosti protokolu CAN, projít aktuálně řešené, poskytované či rozvíjené možnosti, projekty, implementace či komponenty, které se zaměřují na zvýšení bezpečnosti. Analyzovat aktuálně dostupná řešení a možnosti a na základě těchto poznatků navrhnout a implementovat bezpečnostní opatření, které zajistí důvěrnost, integritu a dostupnost v síti založené na komunikačním protokolu CAN.

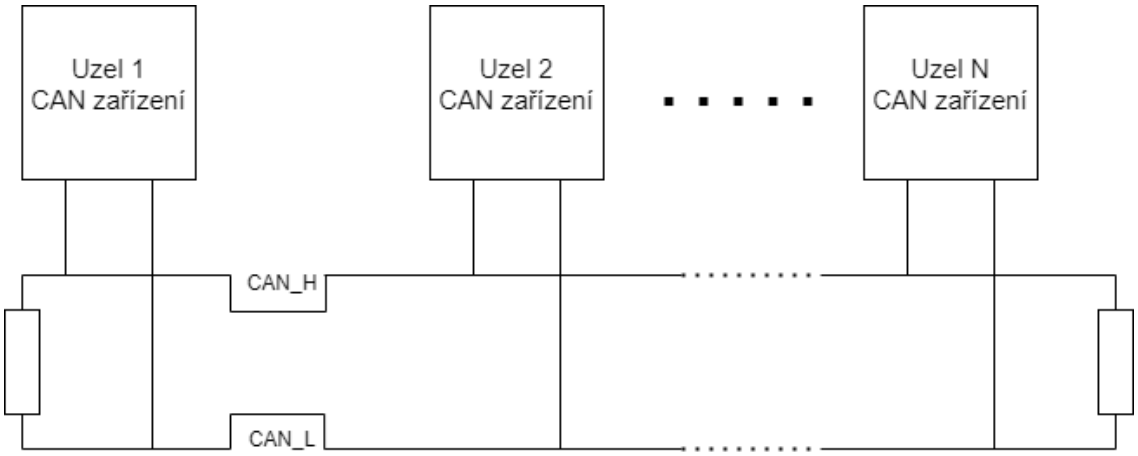
1 Teoretická část

1.1 O CANu

Název protokolu CAN je zkratkou anglického (Controller Area Network). Jedná se o sériový komunikační protokol který byl původně vyvinut tak, aby své hlavní uplatnění nacházel v oblasti automobilového průmyslu na propojení důležitých elektronických jednotek [1]. Jedná se o již poměrně starý komunikační protokol, který byl vyvinut německou společností Robert Bosch GmbH na počátku roku 1983. Původní myšlenkou k uvedení nového komunikačního protokolu byla potřeba jednoduchého a efektivního komunikačního protokolu pro rychle se rozvíjející automobilový průmysl, který by umožňoval propojení elektronických jednotek v automobilech a zároveň by mohl pracovat v prostředích s vysokým rizikem rušení. Postupný rozvoj tohoto protokolu vedl k jeho rozšíření a vytvoření nástaveb pro vyšší vrstvy komunikačního protokolu, které jsou využíván i pro ostatní průmyslová řešení s přesahem k automatizaci, či inteligentním budovám. Popularita CANu a jeho nástaveb stále roste. V dnešní době je možné nalézt implementace tohoto protokolu také například v nasazení pro lékařská zařízení. Konkrétně se využívá například CANopen v systémech pro rentgeny, díky jeho přívětivým přenosovým vlastnostem, ceně a spolehlivosti [2].

1.1.1 Architektura sběrnice CAN

Protokol CAN je založený tak, aby pro přenos informaci využíval sběrnici. Sběrnice je tvořena párem vodičů, které jsou označené CAN_H (CAN_High) a CAN_L (CAN_Low) (viz obr. 1.1. kde je vidět dva páry vodičů CAN_H a CAN_L a připojené CAN zařízení myšlenka obrázku převzatá z [3]) Data po této sběrnici jsou přenášena rozdílem v napětí na jednotlivých vodičích. Klasická CAN podporuje komunikační rychlost až 1 Mb/s, některé vyšší nástavby umožnují také vyšší přenosové rychlosti.



Obr. 1.1: Sběrnice CAN.

1.1.2 Komunikace na protokolu CAN

Komunikace po CAN sběrnici je založena na principu záplavového směrování, kdy jsou komunikační rámce posílána každému připojenému uzlu. Každý rámec obsahuje identifikátor daného rámce, data velikost tohoto bloku záleží na protokolu, který je využíván (u klasického CAN 0 - 64 bitů), následuje kontrolní součet a další parametry. Každý uzel, který je připojený ke sběrnici naslouchá přenášeným rámcům a přijaté zprávy zpracovává na základě identifikátoru. Na sběrnici je možné identifikovat více komunikačních rámců, které nesou různé informace, ať už informaci o přenášených datech, rámce identifikující chybu ve vysílání, či vyžadování čekání ve vysílání. [1]

1.1.3 Kontrola chyb

CAN obsahuje mechanismy, které zajišťují řešení chybových stavů. Například, pokud dochází k chybě na úrovni bitů, čímž se rozumí rozdílná data mezi daty odesílanými a mezi daty přenášenými, je odpovědnost za odhalení této chyby necháno na vysílajícím uzlu. Oproti tomu chyby na úrovni zpráv jsou řešené mechanismem, kdy při vysílání zpráv je nastaveno pole ACK na hodnotu 0, v případě, že dojde k úspěšnému přenosu, je toto pole přepsáno příjemcem na 1. V případě, pole ACK v rámci zprávy zůstává ve stavu 0, je indikována chyba přenosu. Mezi další mechanismy, které v rámci CANu slouží k odhalení chyb během přenosu je CRC, neboli kontrolní součet. Odesílací uzel vypočítá tuto hodnotu a zapíše ji do pole CRC, v případě, že jakýkoliv uzel, který tuto zprávu odchytil vypočítá jiné CRC, než které je zapsané, ohlásí chybu přenosu.[3]

1.1.4 Řešení konfliktů na sběrnici

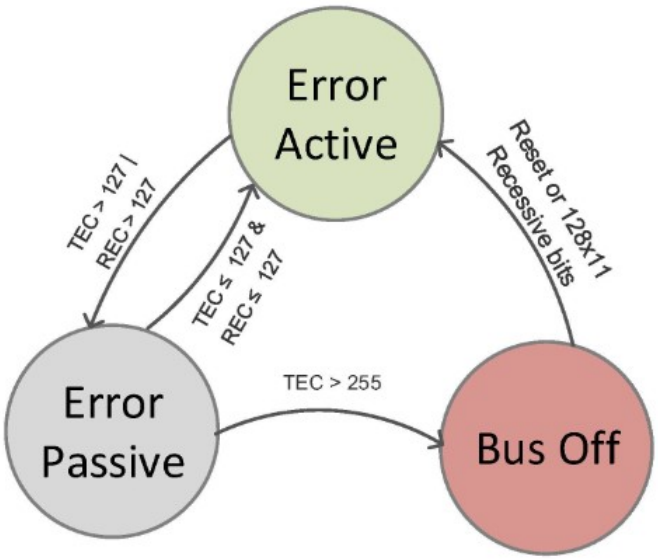
Princip vysílání na sběrnici se řídí modelem CSMA/CA, které stanovuje, jakým způsobem uzly kontrolují, zda je sběrnice volná a je možné započít vysílání. V případě, že nastane situace, při které začnou vysílat dva či více uzlů současně, je tento problém vyřešen tím, že ve vysílání pokračuje pouze uzel s nejvyšší prioritou. [3]

CSMA/CA

CSMA/CA je funkce, která umožňuje uzlům rozpoznat, zda je sběrnice připravená pro vysílání a umožňuje předcházení kolizí při vysílání. Před začátkem vysílání uzly kontrolují komunikační kanál, zda na něm neprobíhá žádná komunikace. Pokud je kanál volný, uzel začne vysílat data. Pokud komunikační kanál volný není, uzel počká náhodně dlouho dobu a následně provádí kontrolu komunikačního kanálu znovu.

Řešení chybných uzlů

CAN má také mechaniku zabraňující fyzickým chybám pomoci funkce vyřazení chybných uzlů pomoci mechanismu pro omezení chyb (ECM). Mechanismus omezení chyb je realizováno pomoci dvou chybových čítačů v každém uzlu, první s označením REC - „Received error counter“ a TEC - „transmitted error counter“. Výchozí hodnotou čítačů je nula a stav je „Error Active“. Jestliže dojde k chybě během přenosu zvýší se TEC o 8, pokud dojde k chybě během přijímaní zvýší se REC o 1. Při úspěšném přenosu či přijetí rámce se sníží odpovědný čítač o jeden bod. V případě, že hodnota chybového čítače uzlu překročí hodnotu 127, vstoupí do „Error passive“ stavu. Pokud TEC čítač překročí hodnotu 255, přepne se daný uzel do stavu „Bus Off“ [3]. Výše popsaný mechanismus demonstruje obr. 1.2, který byl převzat z [3].



Obr. 1.2: Omezení chyb (ECM).

1.2 CAN na modelu ISO/OSI

1.2.1 Fyzická vrstva

Na fyzické vrstvě pro protokol CAN se používá dvojice kroucených párů měděných kabelů se společným uzemněním. Na této vrstvě je realizován fyzický přenos dat. Uzly, které jsou připojené k jedné sběrnici, musí podporovat stejnou přenosovou rychlost. Připojení uzlů k fyzické sběrnici je většinou realizováno pomoci CAN transceiverů, které umožňují přenosovou rychlost až 1 Mbit/s a jsou standardizovány v ISO 11898-2:2016. Pro přenosové rychlosti vyšší než 1 Mbit/s, které například využívá CAN FD, je nutné využít vysokorychlostní transceivery CAN, které mají zlepšeny provozní funkce a to až 5Mbit/s. Výsledná přenosová rychlost samozřejmě závisí také na fyzickém návrhu sítě, topologii a výběru fyzického media, konektorů apod. [5]

1.2.2 Datová vrstva

Na datové vrstvě jsou definovány tři generace CAN protokolu:

1. Klasický CAN.
2. CAN FD.
3. CAN XL.

Klasický CAN

Jedná se o standardní a první verzi tohoto protokolu. Přenosová rychlost základního CANu je standardně 1 Mbit/s. Klasický CAN nachází využití hlavně co se týče přenosu informací v automobilovém průmyslu, automatizaci apod.

CAN FD

CAN FD (CAN Flexible data-rate) jedná se o rozšířenou verzi klasického CANu. Tento protokol umožňuje vyšší přenosové rychlosti a přenos většího objemu dat. Mezi hlavní výhody se dá počítat možnost přenosu dat až rychlostí 8 Mbit/s, oproti klasickému CANu, který podporuje pouze přenosovou rychlost 1Mbit/s. [6]

CAN XL

CAN XL je verze protokolu CAN, která nabízí vyšší přenosové rychlosti a větší prostor pro data, než klasická verze CANu a CAN FD. Tato verze protokolu umožňuje přenosovou rychlost až 10Mbit/s a poskytuje vyšší délku datového rámce (až 2048 B). Je také implementovaná zpětná kombatibilita s klasickým CANem a CAN FD. CAN XL také podporuje alternativní variantu kódovaní, což umožňuje přenosové rychlosti i 10Mbit/s. CAN XL by měl do budoucna mít i integrované zabezpečení datové vrstvy označené jako CADsec. CADsec je bezpečnostní rozšíření pro CAN XL, které vyvíjí společnost Vector Informatik GmbH. V rámci tohoto rozšíření by měly být implementovány kryptografické mechanismy pro zvýšení bezpečnosti tohoto protokolu. [7]

1.3 Protokoly pro vyšší vrstvy

1. CANopen.
2. CANopen FD.
3. J1939.
4. Divicenet.

1.3.1 CANopen

V roce 1994 uskupení CiA vydalo první verzi specifikaci pro protocal CANopen (CiA 301). Jedná se o komunikační systém, který je založený na klasickém CANu. CANopen byl vyvinut jako standardizovaná nástavba. Původně byl navržen pro řídicí systémy strojů, ale v dnešní době se využívá i v dalších oblastech (lékařská technika [2], autodoprava, námořní elektronika, železniční systémy, nebo automatizace budov). CANopen také umožňuje podporu funkce „plug and play“, toto je hlavní

výhodou hlavně pro výrobce zařízení pro CANopen sítě. Výrobci zařízení pro CANopen sítě mohou implementovat standardizovaná rozhraní, aby tak jejich zařízení podporovala funkci „plug and play“ v sítích CANopen. CANopen také poskytuje návrhářům zařízení možnost implementovat patřičné síťové chování přímo do zařízení. Díky této možnosti mohou upravené zařízení například indikovat chybové stavy, ovlivňovat chování sítě či předávat procesní data. [9]

CANopen je postaven na datové vrstvě modelu ISO/OSI podle normy ISO 11898-1. Bitové časování CANopen je specifikováno v normě CiA 301 a umožňuje úpravu datových rychlostí. Všechny CAN-ID adresní schémata jsou založena na 11bitové délce, přičemž CANopen také podporuje 29bitové CAN-ID [9].

1.3.2 CANopen FD

Stejně jako CAN FD je vylepšením klasického CANu, tak je CANopen FD vylepšením CANopen. Toto vylepšení je postavené na CAN FD a umožňuje vyšší propustnost. Díky tomu je vhodný pro datově náročné cloudové aplikace, či systémy, které mohou být modifikovány uživatelem za provozu.

1.4 Bezpečnost protokolu CAN

Tato část se zaměřuje na bezpečnost protokolu CAN. Soustředí se na jeho zranitelnosti, rizika či případné vektory útoků vůči této sběrnici.

1.4.1 Bezpečnostní koncepty

1. Důvěrnost.
2. Integrita.
3. Dostupnost.

1. Důvěrnost

Důvěrností se v problematice kybernetické bezpečnosti myslí to, že zprávy a údaje jsou poskytovány pouze oprávněným osobám. Jelikož CAN neimplementuje žádné kryptografické metody, které by zajišťovali ověření identity, je jasné že útočník tak má možnost přistupovat ke sběrnici a odposlouchávat přenášená data [11].

2. Integrita

Integritou se rozumí úplnost a nenarušenost dat přenášených na sběrnici. Sběrnice CAN má v základu pouze kontrolní součet, který zajišťuje ověření integrity dat

a zda jsou data úplná. Protokol nemá žádnou jinou funkcionalitu, která by zabránila vkládání dat útočníkem [11].

3. Dostupnost

Dostupností je myšleno, že oprávněný uživatel může využívat systém kdykoliv. Vzhledem k fungování CANu a zasílání zpráv na základě priority může nastat situace, kdy síť nebude dostupná pro uzly s nižší prioritou, jestliže bude některý uzel vysílat zprávy s nejvyšší prioritou [11]. Z této části je patrné, že protokol CAN nemá v základu žádná bezpečnostní opatření, která by zajišťovala bezpečnost této sběrnice proti případným útokům.

1.5 Bezpečnost pro CAN

Tato část by práce se zaměřuje na bezpečnostní opatření protokolu CAN. Ačkoliv je protokol CAN poměrně rozšířený a oblíbený pro jeho dobré přenosové vlastnosti, jeho bezpečnost není v základu moc řešena. Z tohoto důvodu je potřebné se na tuto problematiku trochu podívat a analyzovat, jaké jsou hlavní možné hrozby. Na základě stanovených hrozeb a pravděpodobných vektorů útoku je možné stanovit, jaké bezpečnostní prvky by měl zabezpečený systém CAN implementovat. Co se týče základního rozdělení, můžeme rizika rozlišit na dva hlavní vektory možného útoku vůči tomuto protokolu. Mezi základní rozdělení vektorů útoků je možné uvést útoky fyzického přístupu a útoky vzdáleného přístupu [3].

Útoky fyzického přístupu

Jak již název napovídá, tenhle vektor útoku předpokládá, že daný útočník má fyzický přístup ke sběrnici. Do této kategorie spadají útoky založené na odposlouchávání komunikace s využitím fyzického připojení škodlivého zařízení na sběrnici a následné odposlouchávání záplavově posílaných zpráv. Kompromitace přenosu, tímto se myslí možnost odesílání škodlivých dat a narušování klasického fungování sběrnice, u tohoto útoku může být rizikem i zneužití zranitelnosti využívaného CSMA/CA. V případě, že by útočník zasílal data s maximální prioritou a tím zamezil ostatním uzlům komunikovat (útok typu DoS). Útoky tohoto vektoru předpokládají přístup k fyzickému mediu. [11]

Útoky vzdáleného přístupu

Tento vektor útoku předpokládá, že existují zařízení připojená ke CAN síti, která jsou dostupná vzdáleně, nebo že jsou připojené k některému jinému zařízení, které

poskytuje například bezdrátový přenos informací z vnitřní CAN sítě. Cíle tohoto vektoru útoku jsou velmi podobné jako u útoků předpokládajícího fyzický přístup, obecně se dá říci, že opět je cílem útočníka kompromitovat komunikaci, či zachytit přenášená data. [3]

1.5.1 Příklady útoků vůči sběrnici CANbus

Tato část vychází z velké části z problematiky řešené v rámci práce [11]. Následující část obsahuje některé možné útoky zneužívající vlastností protokolu CAN.

Bus Flood Attack

Jedná se o klasický útok typu DoS (Denia of Service) s cílem poškodit CAN sběrnici tak, aby ji nebylo možné využívat. Tento útok spočívá v zaplavení sběrnice rámci s maximální prioritou, což zamezí jakoukoliv komunikaci po této sběrnici. Tento útok využívá funkce protokolu CAN, kdy každý uzel, který chce vysílat data nejdříve vyčkává a začne vysílat data až tehdy, kdy na dané sběrnici neprobíhá žádná komunikace. Posíláním rámců s maximální prioritou způsobí, že žádný uzel nemá možnost vysílat data [11].

Simple frame spoofing

Tento útok má za cíl kompromitovat komunikaci. Tento útok spočívá v tom, že útočník vysílá vlastní podvržené CAN rámce s cílem kompromitovat přenos. Hlavní nevýhodou Simple frame spoofing je, že na sběrnici se nachází jak podvržené rámce útočníka, tak legitimní rámce odesílatele se stejným označením zprávy. Problém duplicitních rámců může způsobit odpojení obou komunikačních uzlů (útočníka i legitimního uzlu), jelikož ostatní uzly budou indikovat chybu a oba komunikační uzly přejdou do stavu „bus off“. [11]

Adaptive spoofing

Tento útok je velmi podobný jako Simple frame spoofing. Rozdíl mezi těmito útoky je v jeho provedení oproti Simple frame spoofing útočník naslouchá sběrnici a podvržené data připojí k datům legitimním, aby nedocházelo k duplicitám a kolizím podvržených rámců s legitimními rámci. [11]

1.6 Bezpečnostní koncepty

V rámci této části se zaměříme na některé koncepty, které uvažují teoretické možnosti bezpečnostních implementaci, které snižují riziko daného útoku, či se snaží

minimalizovat jeho dopady.

1.6.1 Fyzické zabezpečení

Prvky fyzického zabezpečení se hlavně zaměřují na zamezení neoprávněného přístupu k fyzickému rozhraní CAN sítě či narušení provozu samotné sítě. Mezi prvky fyzického zabezpečení se dá například zahrnout ochrana proti elektromagnetickému rušení, ochrana před mechanickým porušením, ochrana před neoprávněným přístupem k fyzickému rozhraní CAN sítě. Jako ochrana proti těmto rizikům je možné využívat kryty či speciální pouzdra pro konektory, vypnutí nepoužívaných portů, segmentace sítě, detekce anomálií a podobně. Této problematice se věnují i specializované HW komponenty, jako jsou například bezpečnostní brány [10].

1.6.2 Softwarové zabezpečení

V rámci této problematiky se zaměřujeme na zamezení neoprávněného přístupu k datům, které jsou na dané CAN síti přenášené a případnému zabránění jejich kompromitaci. Mezi základní zabezpečení přenosu dat na sběrnici by se dalo zahrnout autentizace zpráv, kódování zpráv, šifrovaní přenosu zpráv, ochrana proti DoS, detekce anomálií. Tyto základně definované mechanismy slouží pro zvýšení bezpečnosti a důvěrnosti CAN komunikace. [3] V rámci této problematiky je opět možné využít specializovaného HW, jako jsou bezpečnostní brány, či jiné komponenty, které zlepšují zabezpečení pro protokol CAN. [10]

Autentizace zpráv

Autentizací zpráv se rozumí, že zprávy které jsou posílané po síti jsou opravdu od odesílatelů, kteří jsou k tomu oprávněni. k autentizaci zpráv se nejčastěji využívají digitální podpisy, díky kterým je možné ověřit identitu entity, která zprávu podepsala, či případně ověření komunikujících stran. Používání digitálního podpisu zvětšuje přenášená data nebo případně vyžaduje operace ověření komunikujících stran. Z tohoto důvodu je vždy nutné zvážit nutnost nasazení tohoto bezpečnostního mechanismu. [3]

Kódování zpráv

Kódování zpráv se využívá k ochraně zpráv před narušením či úpravou během přenosu. Nejčastěji se využívají různé metody pro kódování zpráv, například CRC. Díky tomuto algoritmu je vytvořen kontrolní součet zprávy, který je ke zprávě připojen a slouží k ověření integrity dat, jak bylo popsáno výše. Tyto metody CAN v základu

implementuje, ale případné další mechanismy kontroly je vždy nutné dobře promyslet, jelikož opět budou zvětšovat přenášená data a tak zvyšovat počet posílaných zpráv.[3]

Šifrování

Šifrováním přenosu dat je dosaženo zabezpečení obsahu zpráv před neoprávněným přístupem. k takovémuto typu zabezpečení je možné využít asymetrické či symetrické šifry podle konkrétních požadavků na bezpečnost a výkonnostní možnosti sítě. V problematice CAN se pro šifrování přenášených dat nejvíce řeší využívání symetrické kryptografie, kdy je komunikace ověřena šifrou se stejným klíčem. Symetrická kryptografie se využívá, jelikož ve většině případů je nutné šifrovat komunikaci v reálném čase.[3]

Ochrana proti DoS

Ochrana proti DoS útokům je u CANu zaměřena spíše na minimalizaci dopadů útoků na dostupnost sítě. Mezi základní bezpečností opatření se dají využít filtry zpráv, bezpečnostní brány, software pro detekci útoků či anomálii. [11]

Detekce anomálií

Detekce anomálií slouží k odhalení neobvyklých nebo podezřelých aktivit v síti. Díky detekci anomálií je možné identifikovat útoky či chyby, které se mohou na síti vyskytnout. Tuto detekci je možné realizovat prostřednictvím analýzy provozu sítě. k tomuto účelu je možné sehnat specializovaný hardware, který poskytuje některé detekční funkce, vždy je ale nutné reflektovat konkrétní požadavky konkrétního CAN nasazení. [11]

1.7 Shrnutí

Jak vyplývá z předešlých částí, protokol CAN pracuje se sběrnicí. Data jsou po sběrnici distribuována záplavově, což sebou nese některá rizika možných útoků. Jak již vyplývá z předchozích částí CAN v základu neimplementuje prakticky žádné bezpečnostní mechanismy, které by zajišťovaly jeho bezpečnost či ochranu před útoky. A jelikož CAN je využíván v průmyslu, automatizaci, automobilovém, lodním, leteckém sektoru apod., je nutné se zaměřit na problematiku zvýšení jeho bezpečnosti a vypracovat analýzu, která bude zkoumat možnosti jeho zabezpečení, ať už existující či prozkoumat a navrhnout některé nové.

2 Analýza a návrh zabezpečení

Jak je z popisu předchozí části patrné, protokol CAN nemá implementovány prakticky žádné způsoby zabezpečení. Proto bude v této části zaměřeno na možnosti zabezpečení a na způsoby, jak se na problematiku bezpečnosti u CAN dívat. Není tajemstvím, že v dnešní době je mnoho projektů, komponentů a rozšíření, které bezpečnost CANu vylepšují, ale každý na to jde trochu vlastní cestou. Z tohoto důvodu je naprosto klíčové stanovení toho, jak bude CAN síť využívána, provozována a jaké jsou potenciální rizika tohoto použití. V rámci této problematiky se rešerše zaměřila hlavně na zabezpečení klasického CANu, případně nalezení standardizované verze Secure CAN, případně nástavby CANopen. V rámci některých článků a popisů je možné narazit na zmínky o nově připravované nástavbě Secure CANopen či případně Secure CANopen FD, ale zatím nikde není možné nalézt standardizované nástavby s tímto označením. Z tohoto důvodu bylo kontaktováno uskupení CiA. CiA (CAN in Automation) jedná se o mezinárodní skupinu uživatelů a výrobců, kteří se zaměřují na CAN. Cílem tohoto uskupení je poskytnutí platformy pro budoucí specifikace a normy, které souvisí s CANem. Cílem bylo zjistit, jak jsou aktuálně nasazované implementace CANu v praxi a zda jsou již definovány zabezpečené nástavby pro CANopen.

Výsledkem komunikace s pracovníky CiA bylo zjištěno, že aktuálně zatím nejsou přesně definované oficiální bezpečnostní opatření pro CAN, CANopen a CANopen FD. Definice těchto opatření je aktuálně v procesu, ale je možné nalézt různá řešení od různých společností, které vydávají vlastní řešení. Jako aktuálně nejlepší možné bylo doporučeno řešení od společnosti EmSA, která vyvinula CANcrypt jako vlastní řešení kybernetické bezpečnosti pro CAN. Také bylo popsáno, jak vypadají některé praktické příklady využití CANu v praxi, které vysvětlovaly, proč kybernetická bezpečnost na CANu není hlavní prioritou pro toto uskupení. V každém případě je nutné rozhodovat o nasazovaném bezpečnostním řešení v kontextu hrozeb a příkladu používání.

První příklad využití CANu u zařízení v nemocnicích bere v potaz, že CANopen se používá pro vytvoření vnitřní sítě pro CT (Computer Tomograph), kdy útok s využitím fyzického přístup k médiu je velmi nepravděpodobný a tak se tento aspekt hrozby zanedbává. U útoku s využitím vzdáleného přístupu je to horší, ale zde se počítá s tím, že je CT je připojeno k ostatním zařízením skrz klasický Ethernet, ne přes CANopen. Na zařízení, které nám zajišťuje přenos dat mezi těmito dvěma doménami, můžeme implementovat firewall. Tímto zajišťujeme, že vnitřní síť založená na sběrnici CAN nevyžaduje speciální zabezpečení a další potenciální rizika se zanedbávají.

Další popsaný příklad útoku, je takzvaný GM hack, který provedl Charlie Miller

a Chris Valasek. V základu bylo zařízení v síti Vezion, které mělo otevřené všechny porty. Multimediální rozhraní mělo připojení pomoci telefonní linky k síti Verizon. QNX, které běželo na tomto multimediálním rozhraní mělo zbytečně otevřené porty a k této síti se dalo připojit z libovolného mobilního telefonu v síti Verizon. Následně se objevila chyba i v interním systému zpráv QNX. Díky této chybě bylo možné získat plný přístup k QNX běžícímu na multimediálním rozhraní pomocí již zmíněného otevřeného portu. Díky těmto dvěma problémům měli útočníci plný přístup k multimediálnímu rozhraní. Toto multimediální rozhraní bylo přes komunikační procesor připojeno k interní síti CAN. Komunikační procesor tvořil pomyslnou bránu do CAN sítě a měl také sloužit jako firewall. Ale firmware nebyl zabezpečen, tudíž bylo možné přeprogramovat tento komunikační procesor. Toto je třetí chyba která v návrhu byla a přes tento přeprogramovaný komunikační procesor měli útočníci plný přístup ke sběrnici CAN. A z tohoto vyplývá otázka, pomohlo by, když by byl CAN zabezpečený? Dost pravděpodobně ne, jelikož útočníci měli plný přístup ke komunikačnímu procesoru, takže by měli přístup i k potenciálním klíčům, které by zabezpečovaly CAN.

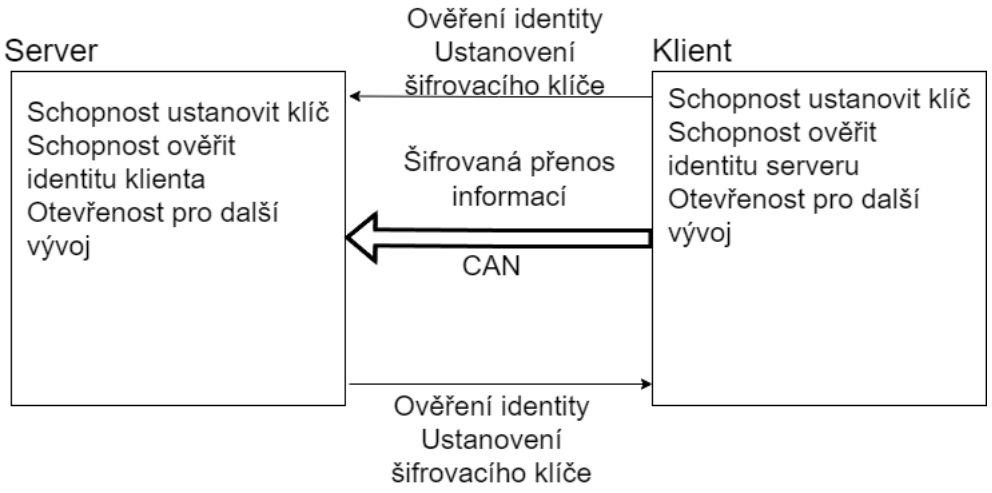
Tyto příklady nemají poukázat na to, že by zabezpečení sběrnice CAN bylo zbytečné, jen je potřeba brát v úvahu vždy kontext nasazovaného zabezpečení a uvážit potenciální hrozby. Také se povedlo zjistit, že aktuálně není možné nalézt standardizované prvky kybernetické bezpečnosti pro CAN a CANopen. Z tohoto důvodu je nutné se na tuto problematiku více zaměřit a projít neoficiální řešení, které je možné aktuálně sehnat. Informace které byly sděleny pracovníky CiA následně pomáhaly s volením parametrů pro hloubkovou analýzu nabízených bezpečnostních řešení.

2.1 Definování bezpečnostních požadavků

Díky komunikaci s uskupením CiA a předchozí analýze bylo rozhodnuto, že mezi nejdůležitější parametry, kterých se budeme snažit dosáhnout v rámci implementace bude autentizace komunikujících stran, šifrovaní přenášených zpráv. Celkový rámec uvažovaného zabezpečení je směřován na sektor inteligentních budov a průmyslovou implementaci. Bezpečnost pro automobilový, lodní a letecký průmysl není primárním cílem tohoto výzkumu.

Z tohoto důvodu by měla demonstrační implementace představovat dvě komunikující strany s možností ověření identity a možností ustanovení šifrovacích klíčů, díky kterým by bylo možné šifrovat komunikaci. Výsledné demonstrační řešení by mělo mít možnost využívání moderních kryptografických algoritmů, jako jsou například eliptické křivky, a další. Demonstrační návrh by měl představovat možnost jak řešit zabezpečenou komunikaci pro sítě s využitím například pro inteligentní budovy. Z tohoto důvodu by bylo ideální, když by řešení bylo optimalizované pro

fungování i s menšími systémovými zdroji, aby bylo případně možné implementovat na menší zařízení. Ideální by také byla případná možnost aktualizovat či upravovat využívané kryptografické funkce, aby bylo možné reflektovat aktuální bezpečnostní trendy. Mezi velké výhody je určitě třeba zmínit otevřenost řešení, aby bylo možné případně na návrhu dále pracovat a rozšiřovat toto řešení. Na obr 2.1 Návrh pro zabezpečenou komunikaci, můžeme vidět teoretickou představu demonstrativní, zabezpečené komunikace, který představuje model dvou komunikačních stran, stranu serveru a stranu klienta. Obě dvě strany by měli být schopné ověřit identitu a vytvořit společný šifrovací klíč, který se bude využívat pro šifrování komunikace.



Obr. 2.1: Návrh pro zabezpečenou komunikaci.

2.1.1 Analýza aktuálně dostupných řešení

Jak je již zmíněno výše, bylo rozhodnuto o nalezení bezpečnostní implementace či knihovny, která by umožňovala další práci na zvýšení bezpečnosti CANu. První nalezenou knihovnou, kterou dle popisu využívali lidé pro nasazování vlastních bezpečnostních algoritmů pro CAN byla knihovna s názvem CANfestival. Jedná se o open-source knihovnu, jež podporuje CANopen a další vývoj. Byla vydána v roce 2001 a je postavená na jazyce C. Mezi hlavní výhody této knihovny patří možnost vytvářet a implementovat kód pro specifické uzly a také vytvářet konfigurační skripty. Hlavní problém při pokusech o nasazení této knihovny byl v tom, že se jedná o opravdu starou knihovnu, která je do dnešních dní udržována jen komunitně. Mezi další problémy spojené se stářím této knihovny patří, že při využití na operačním systému Windows podporovala pouze Win32 a pro své fungování využívala Python 2, který je již také zastaralý. Nutnost využívání Pythonu 2, který již není podporovaný, vedla k hledání novější verze, která by byla kompatibilní s Python 3. Po dalším

průzkumu bylo nalezeno, že někteří uživatelé tento problém obcházeli. Problém ale byl, že těmito úpravami, které umožňovaly fungování i s Pythonem 3, docházelo k problémům s funkčností mnoha částí této knihovny. Z tohoto důvodu bylo od dalšího testování s CANfestival upuštěno. Problémy, které byly spojené s implementací knihovny CANfestival vedly k nutnosti hlubší analýzy nabízených bezpečnostních implementací a zařazení důležitých faktorů pro budoucí výběr potenciálního řešení pro bezpečnost na CAN.

V rámci hlubší analýzy bylo analyzováno velké množství bezpečnostních nástaveb, hardwarových komponent, knihoven a řešení, které se věnují obecné práci s CAN a jeho nástavbami. Parametry pro analýzu byly volené na základě dřívějších zkušeností a poznatků. U každého ze zmíněných prvků pro analýzu byl pokus o kontaktování dodavatele daného softwaru či hardwaru a dané informace a parametry, které zde jsou uvedené, byla snaha ověřit přímo u výrobců / dodavatelů. V následujícím obrázku jsou uvedené nejzajímavější řešení, které z analýzy vzešly. Parametry, které jsou uvedené v obr 2.2 Nástavby pro bezpečnost CANu porovnání, který vizualizuje užší vzorek, na kterém byla vedena analýza, zkoumají základní předpoklady pro možný zájem:

- Placené - Tento parametr říká, zda se jedná o placené rozšíření.
- Nezávislost na HW - Tento parametr určuje, zda je fungování tohoto produktu úzce spojeno se speciálním hardwarovým vybavením nebo zda se jedná o pouhé rozšíření či softwarovou implementaci.
- Open-source - určuje, zda je daná věc open-source a umožňuje zasahování do kódu, analýzu, další práci apod.
- Asymetrická kryptografie - zda dané řešení podporuje asymetrickou kryptografii.
- CAN... - identifikuje, pro jakou nástavbu CANu je dané rozšíření nasaditelné.
- Autentizace - poukazuje na to, zda daná věc podporuje autentizaci.
- Zabezpečení přenosu - toto určuje, zda dané rozšíření má přímý dopad na zabezpečení přenášených dat po sběrnici, nebo zda se jen jedná o podpůrnou funkcionalitu.
- Pouze podpůrné rozšíření - tento parametr říká, zda se jedná o bezpečnostní řešení, nebo zda se jedná o funkcionalitu, která jen umožňuje lepší práci s CAN / zabezpečení ostatních věcí ohledně CAN.

Implementace	CANcrypt	CANopen safety	CANpie FD	Cryptocan	TJA115x CAN	CANedge2
Placene	✓	✓	✗	✓	✓	✓
Nezávislost na HW	✓	?	✓	✗	✗	✗
Open-source	✗	✗	✓	✗	✗	✗
Asymetrická kryptografie	✗	✗	?	✗	✗	✗
CANbus	✓	✗	✓	✓	✓	✓
CAN FD	✓	✗	✓	✗	✓	✓
CAN XL	?	✗	✗	✗	✗	✗
CANopen	✓	✓	✗	✗	✗	✗
CANopen FD	✓	?	✗	✗	✗	✗
Autentizace	✓	✓	✗	✗	✓	✗
Zabezpečení přenosu	✓	✓	✗	✓	✓	✗
Pouze podpůrné rozšíření	✗	✗	✓	✓	✗	✓

Obr. 2.2: Nástavby pro bezpečnost CANu, porovnání.

CANcrypt

Jedná se o samostatný produkt vydaný společností ESAcademy [12]. CANcrypt podporuje šifrování a ověřování komunikujících uzlů a implementuje sady škálovatelných bezpečnostních rozšíření, které jsou založené na symetrické kryptografii, jelikož počítá s nasazením v reálném čase. Mezi hlavní výhody se dá počítat nezávislost na protokolu, díky čemuž je možné jej využívat i s protokoly jako je CANopen a CANopen FD. Také podporuje komunikační skupiny, které si mohou mezi sebou udržovat zabezpečené komunikační kanály. Také se jedná o bezpečnostní řešení, které bylo doporučené pracovníky CiA, jakožto aktuálně nejlepší a nejrelevantnější řešení, pro zabezpečení sběrnice CAN.[12]

2.1.2 Výsledky analýzy nabízených řešení

Z této analýzy vychází, že existují různá řešení, která rozšiřují různými způsoby používání CANu. Protože hlavním cílem této analýzy bylo nalezení zabezpečené verze pro CAN, pokračovala další analýza u CANcrypt. Důležité je taky neopomenout další řešení, která zajišťují bezpečnost jako jsou hardwarové moduly TJA115x CAN nebo prudukty od společnosti Canis Automotive Labs, které se hodně zaměřují na bezpečnost na fyzické vrstvě a poskytují řadu vlastních čipů a jejich rozšíření, které

řeší zabezpečení sběrnice CAN. Také produkty společnosti CSS Electronics se zaměřují na bezpečné uchovávání a práci s CAN daty. Existuje tedy velké množství užitečných rozšíření a implementací. Další výzkum byl zaměřen na implementaci CANcrypt. Jelikož se jedná o placené rozšíření, bylo pro další průzkum staženo demo této implementace, které je poskytováno zdarma. V rámci pokusu o rozchození a implementování tohoto produktu byl problém s prakticky neexistující volně dostupná dokumentací. Jelikož, neexistuje žádná volně dostupná dokumentace, která by popisovala jak provést nasazení tohoto produktu, bylo od dalších pokusů o nasazení upuštěno. Jak bylo zjištěno později demo má primárně sloužit pro demonstraci a testování příkladů z knihy která je možná separátně dokoupit. Testování demo verze CANcrypt nebylo možné bez zakoupené knihy.

Analýza CANcrypt pokračovala dále více teoretickou cestou. Mezi hlavní problémy, proč bylo rozhodnuto o nepokračování zkoumán CANcrypt patří, nemožnost případného vlastního dalšího rozvoje a absence možnosti využívat asymetrické kryptografie. Takže ač je CANcrypt velmi dobré řešení bezpečnosti na CANu, úplně neodpovídá požadavkům na navrhované bezpečnostní řešení, které by mělo býti více otevřené, umožňovat lepší práci a možnosti dalšího rozvíjení. Z těchto důvodů se další analýza zaměřila na více obecné kryptografické knihovny s cílem nalézt otevřenější řešení, které by umožňovalo volnější vývoj zabezpečeného řešení.

2.1.3 Zaměření na obecnější krypto knihovny

Tato nová analýza brala v potaz nově získané informace, které jsme získali z předchozího výzkumu, a proto se zaměřuje na porovnání obecnějších knihoven s konkrétními požadavky. Cílem této analýzy je nalezení lepší alternativy pro zabezpečení CANu s využitím volně dostupných obecných krypto knihoven, které by umožňovaly dosažení stanoveného cíle. Každá knihovna má obrovské množství věcí, které přináší, a které umožňuje, proto se zaměříme jen na parametry, které jsou relevantní a důležité pro tuto práci. Tyto parametry vychází z předchozí analýzy konkrétních nasazení. V rámci první analýzy byli kontaktováni dodavatelé bezpečnostních rozšíření pro CAN s otázkami ohledně jejich řešení. V rámci těchto komunikací bylo zjištěno, že některé firmy, zabývající se obecnou bezpečností pro komunikační protokoly, využívali pro zabezpečení TLS. Z tohoto důvodu je podpora TLS zařazena jako jeden z hlavních parametrů pro další analýzu obecných krypto knihoven, ostatní parametry jsou volené s ohledem na požadovanou zabezpečenou implementaci. Mezi hlavní parametry které je možné vidět na obr. 2.3 Otevřené krypto knihovny, který zobrazuje užší analýzu obecných krypto knihoven:

- Licence - označuje pod jakou licencí je daná knihovna.

- Podpora TLS verze 1.3 - jedná se o nejnovější verzi tohoto protokolu, který zajišťuje zabezpečenou komunikaci s využitím moderních šifer a protokolů, zvyšuje rychlost komunikace a snižuje náchylnost k útokům. Na tomto protokolu je postaveno velké množství bezpečnostních implementací i mimo problematiku CAN.
- Podpora DTLS - protokol zajišťující zabezpečenou komunikaci, nástavba na TLS.
- Podpora ECC - znamená, že kryptografická knihovna podporuje algoritmy pro šifrování a podepisování založené na eliptických křivkách.
- Podpora OCSP - protokol pro ověřování platnosti certifikátů.
- Velikost kódu (odhad) - popisuje velikost binárního souboru knihovny, jedná se o odhad, přesné určení závisí na instalovaných rozšířeních a funkcionalitách.
- Implementace pro CAN - identifikuje, zda daná knihovna již řeší implementaci na CAN.

Kategorie	OpenSSL	WolfSSL	GnuTLS	MatrixSS
Licence	Apache	GPL	LGPL	GPL
Podpora TLS 1.3	✓	✓	✓	✓
Podpora DTLS	✓	✓	✓	✓
Podpora ECC	✓	✓	✗	✓
Podpora OCSP	✓	✓	✓	✓
Velikost kódu (odhad) [kB]	4200	100	500	200
Implementace pro CAN	✗	✓	✗	✗

Obr. 2.3: Otevřené krypto knihovny, porovnání.

V rámci této analýzy byly porovnávané obecné krypto knihovny, které se zdály relevantní pro tuto analýzu. Knihovna, která nejlépe odpovídá analyzovaným požadavkům i kvůli již řešené implementaci pro CAN, byla knihovna wolfSSL od stejnojmenné společnosti.

WolfSSL

WolfSSL je americká firmá, jež byla založena v roce 2004 a vydala knihovnu wolfSSL. Zaměřuje se na poskytování jednoduchých vestavěných bezpečnostních řešení s důrazem na rychlost, malou velikost a přenositelnost. Tato knihovna je poskytována open-source s GPLv2 licencí, což umožňuje nahlížet do kódu a dále s touto knihovnou pracovat. Společnost wolfSSL poskytuje více produktů:

- wolfSSL - SSL/TLS knihovna postavená na jazyce C

- yaSSL - SSL knihovna postavená na jazyce C ++. (Pokud je to možné, doporučuje se využívat wolfSSL).
- wolfCrypt - vestavěný kryptografický engine postavený na jazyce C, který nabízí optimalizace pro rychlost, nižší spotřebu paměti a pod.
- wolfCrypt FIPS
- wolfSSH - malá implementace SSH

Mezi další přednosti patří dobře dostupná a detailní dokumentace k jednotlivým produktům, veřejně uvedené benchmarky, aktivní technická podpora a aktivně udržované, aktualizované a vylepšované produkty.

2.2 Zvolení wolfSSL

Nalézt specializované řešení pro bezpečnost na CANu, které by vyhovovalo požadavkům na navrhované zabezpečení není zcela možné i jak ukázala druhá analýza ohledně velkých krypto knihoven. Na základě informací, které se podařilo získat v rámci předchozích analýz, bylo zvoleno využití wolfSSL. Splňuje nejlépe hledané parametry, které byly nastaveny tak, aby odpovídaly bezpečnostnímu návrhu. Mezi parametry, které rozhodly ve prospěch zvolení wolfSSL je možnost využívání asymetrické kryptografie, možnost využívat eliptické křivky, podpora certifikátů, přívětivé licencování a otevřenost knihovny, velmi aktivní technická podpora, malá velikost zkompilovaného kódu, optimalizace i pro výkonnostně slabší zařízení, podpora TLS verze 1.3. Využívání této knihovny také zajišťuje možnost další práce a rozvíjení navrhovaného řešení. A velkou výhodou, proč bylo také zvoleno wolfSSL je již existující implementace pro CAN.

2.3 Jak funguje TLS 1.3

V rámci této části bude rozepsáno, jak funguje TLS verze 1.3, což je klíčové pro pochopení funkčnosti zabezpečení navrhovaného řešení a praktického výstupu této analýzy. Nápad na hledání bezpečnostní implementace postavené na TLS vychází z předchozí analýzy, v rámci které bylo zjištěno i po komunikaci s dodavateli, že některé zamýšlené zabezpečeni CAN uvažují využívání TLS nebo případně některá jiná bezpečnostní řešení z oblasti automatizace a inteligentních zařízení využívají tohoto protokolu a staví na něm své produkty.[14]

TLS (Transport Layer Security) jedná se o kryptografický protokol pro zabezpečenou komunikaci. Jedním z úkolů TLS je poskytování bezpečného kanálu mezi dvěma komunikujícími stranami. Mezi stěžejní vlastnosti, které by měl zabezpečený kanál poskytovat, a které jsou založené na obecném bezpečnostním modelu jsou:

- Autentizace: Možnost ověření strany serveru a klienta. Pro dosažení je možné využít asymetrické kryptografie, algoritmu podpisu s využitím eliptické křivky, atd.
- Důvěrnost: Data která jsou posílaná zabezpečeným kanálem jsou čitelná pouze pro koncové body. TLS ze základu neskrývá délku přenášených dat. Koncové body mají možnost skrýt délku přenášených dat, aby se tak předcházelo útokům založeným na analýze síťového provozu.
- Integrita: Data přenášená zabezpečeným kanálem není možné upravovat případným útočníkem bez toho, že by to bylo detekováno.

TLS není závislé na aplikačním protokolu, což znamená, že protokoly vyšší vrstvy mohou využívat TLS jakou součást svého zabezpečení. V případě CANu toto znamená, že by mělo být možné využít tuto knihovnu i pro případné zabezpečení CANopen apod.

2.3.1 Struktura TLS

TLS se skládá ze dvou primárních částí.

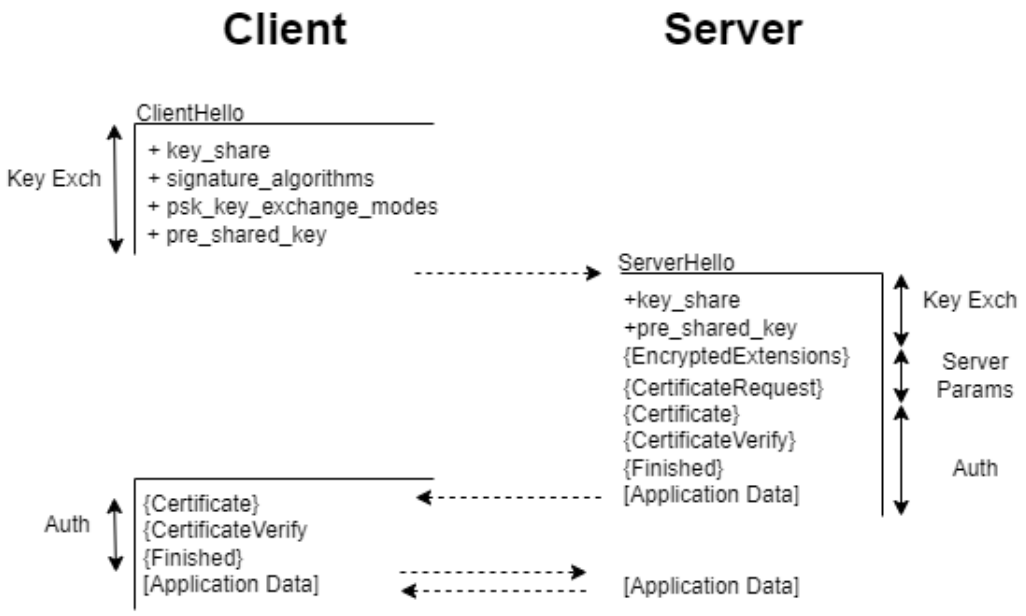
1. Protokol handshake
2. Protokol record

1. Protokol handshake

Jedná se o protokol, který zajišťuje ověření komunikujících stran a zajišťuje domluvení kryptografických modelů, které budou využívány. Měl by také zajistit distribuci klíčů pro šifrování přenášené komunikace a ustanovit potřebné parametry. Handshake protokol by měl být odolný proti útokům, které by se snažily vynutit novou výměnu parametrů. V případě, že dojde k selhání tohoto protokolu nebo k jiné chybě, spustí se ukončení spojení.

Protokol handshake obsahuje tři fáze, které následně budou patrné na následujícím diagramu viz obr. 2.4: Diagram TLS 1.3 handshake, tento obrázek vizualizuje diagram převzatý z dokumentace o TLS 1.3 [14].

1. Key Exchange: v rámci této části jsou vybrané kryptografické parametry a ustanoví se klíč. Všechny operace po této fázi jsou již šifrované.
2. Server Params: zde jsou stanoveny další parametry pro protokol handshaku.
3. Authentication: ověření, potvrzení handshake integrity.



Obr. 2.4: Diagram TLS 1.3 handshake.

2. Protokol record

Tento protokol využívá parametry které byly stanovené handshake protokolem k ochraně přenášených dat mezi komunikačními stranami. Tento protokol zajišťuje rozdělení zpráv na více částí, kdy každá z nich je nezávisle chráněna pomoci sestavených klíčů. Následně také zajišťuje ověření přijatých dat, dešifrování a znovu složení.

Větší podrobnosti je možné nalézt v dokumentaci k TLS 1.3. [14]

2.4 Implementace wolfSSL na CAN

V této části bylo zkoumáno, jak provést nasazení krypto knihovny wolfSSL (ta byla v předchozí části analýzy zvolena jako nejlepší) na sběrnici CAN, aby tak byla zajištěna zabezpečenou komunikaci po této sběrnici. Postup jak provést tuto implementaci krok za krokem je popsán v části chapter 3, zde je popis více teoretický.

Na základě stanoveného demonstračního návrhu, jak by měla vypadat zabezpečená komunikace, který předpokládá, dvě komunikační strany, Serveru a Klienta. Pro tento návrh je možné využít TLS, kdy by se měl klient ověřit proti serveru a následně by měla být ustanovená zabezpečená komunikace ze strany klienta na server s využitím ustanoveného klíče pro šifrování komunikace.

V rámci této části analýzy byl pro zprovoznění wolfSSL využit virtuální počítač s operačním systémem Debian. Pro tuto implementaci je také nutné nainstalovat balíčky, které nám umožní vytvářet a simulovat virtuální sběrnici CAN pro další

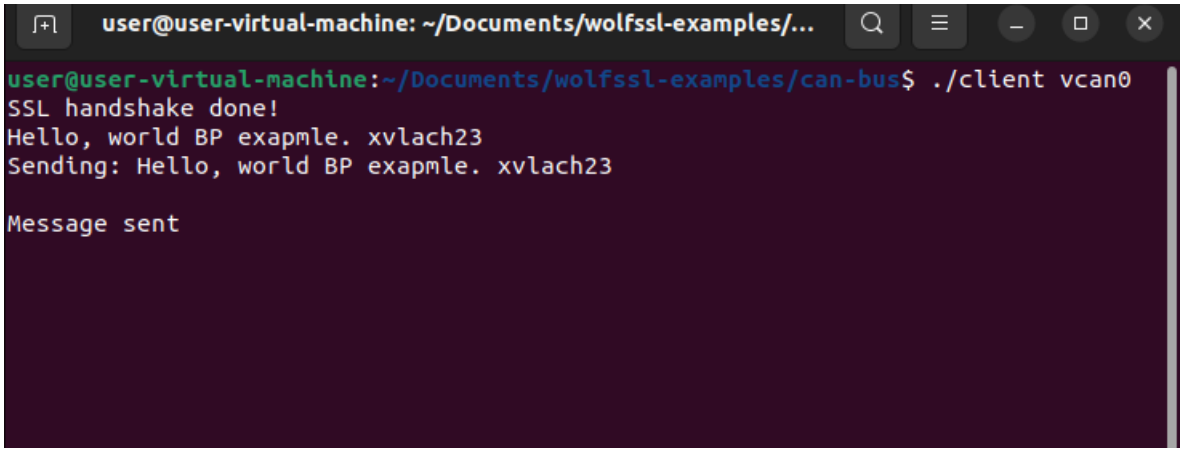
testování. Přesný popis je opět uveden v kapitole chapter 3. Další části je instalace samotné knihovny wolfSSL a jejího rozšíření, které by mělo podporovat připojení ke sběrnici CAN.

Nasazení čisté knihovny wolfSSL na virtuálním Debianu proběhlo bez problémů. Při nasazování rozšíření podporujícího připojení ke sběrnici CAN se problémy vyskytly a bylo nutné definovat specifickou funkci ISO-TP a provést sestavení tohoto rozšíření ručně. U řešení problémů s touto částí byl kontaktován dodavatel wolfSSL, se kterým bylo řešeno sestavování tohoto rozšíření.

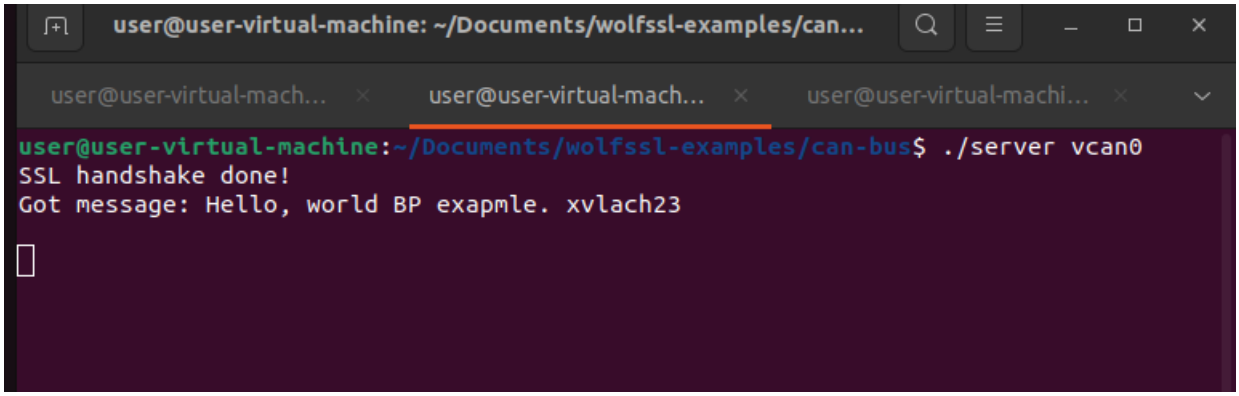
Poté, co se povedlo opravit a zprovoznit toto rozšíření, byl opět kontaktován dodavatel wolfSSL s připomínkou, že v README souboru na githubu, ve kterém byl definován popis a postup, jak je dané rozšíření nasaditelné. Je potřeba udělat úpravy pro sestavování tohoto rozšíření, aby bylo funkční a sestavitelné. Tato připomínka se potkala s velkým přijetím a na základě této připomínky došlo k upravení postupu pro sestavování tohoto rozšíření.

Komunikace s dodavateli wolfSSL přetrvávala i nadále, jelikož pracovníci wolfSSL projevili zájem o detaily o tomto výzkumu a popisu kontextu celé práce. Na tento popud proběhl i meeting na platformě Zoom s pracovníky wolfSSL, kde byly sdíleny zkušenosti a poznatky.

Praktické řešení implementuje TLS na CAN sběrnici, tudíž nám vzniká praktická demonstrace, která prezentuje stranu klienta a serveru. Ty se ověří na základě digitálních certifikátů, které jsou generované s využitím RSA 2048. Následně dojde k ustanovení symetrických šifrovacích klíčů, které zajišťují šifrování a dešifrování přenášených zpráv. Praktická implementace umožňuje spuštění stranu serveru a klienta na předem vytvořené virtuální CAN sběrnici vcan0. Po spuštění strany klienta a serveru, je zobrazena zpráva „SSL handshake done!“ která značí, že došlo k úspěšnému ověření stran a byly ustanoveny šifrovací klíče. Následně je patrné odeslání testovací zprávy ze strany klienta na server. U strany serveru je vidět obdržení zprávy. Na následujících obrázcích obr. 2.5: Klient strana připojená k virtuální sběrnici CAN a Obr. 2.6: Server strana připojená k virtuální sběrnici CAN je patrné zapnutí strany serveru a klienta a připojení k vytvořené virtuální sběrnici vcan0 s využitím příkazů „./server vcan0“ a „./client vcan0“. Následně už je jen vidět odeslání testovácí zprávy s využitím komunikačního rozhraní.

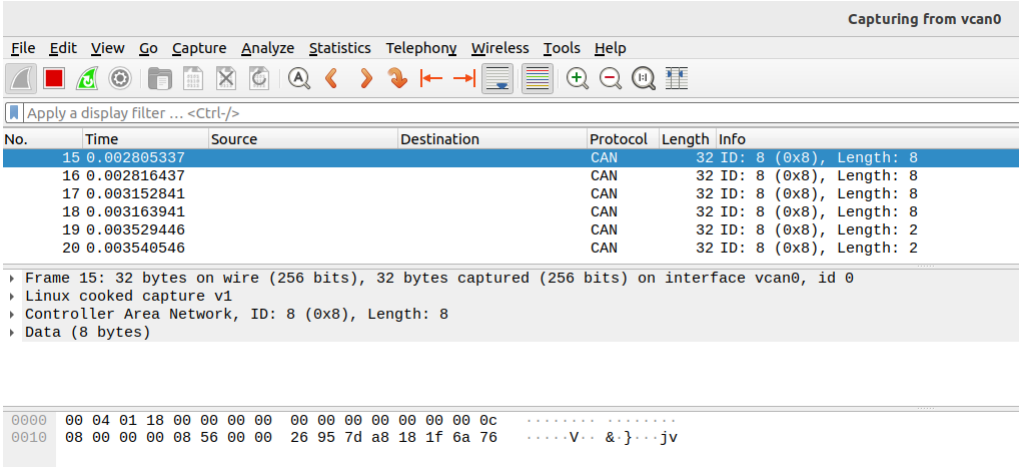


Obr. 2.5: Klient strana připojená k virtuální sběrnici CAN.



Obr. 2.6: Server strana připojená k virtuální sběrnici CAN.

Pro demonstraci, že je komunikace šifrována, je na následujícím obrázku zachycená komunikace s využitím Wireshark, což je program pro zachytávání komunikace a v tomto případě zobrazuje komunikaci, která probíhala na sběrnici vcan0, ke které jsou připojené komunikační strany. Jak je zřejmé ze spodní části obr.2.7: Wireshark zobrazující přenášená data z předchozí ukázky, jsou nesmyslná, což poukazuje na to, že daná data jsou šifrována.

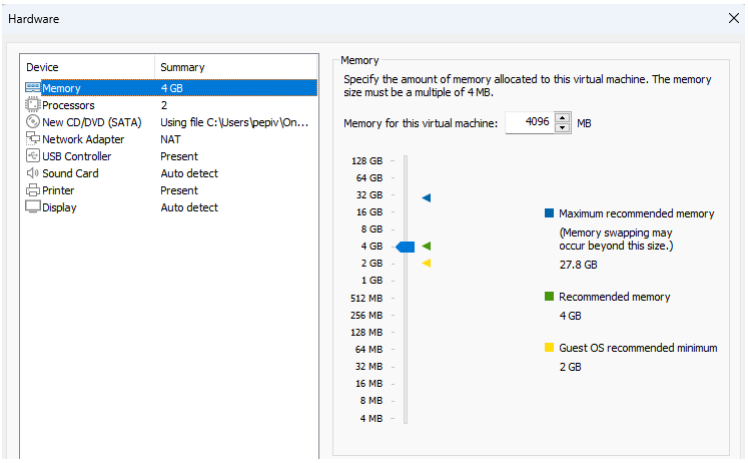


Obr. 2.7: Wireshark zobrazující přenášená data.

2.5 Testování

Tato část se zaměřuje na testování implementovaného zabezpečení, aby bylo zjištěno, jak moc ovlivňuje přenosové parametry. Testovat budeme implementované zabezpečení využívající wolfSSL pro CANBus s využitím TLS 1.3. Pro ověření stran je využíváno RSA 2048 a šifrování zpráv je zajištěno s využitím AES 128. Pro oba testované typy komunikace bude využívána virtuální CANbus sběrnice se standardní šířkou pásma.

Testovací výsledky je nutné brát orientačně. Jelikož je testování prováděno na virtuální CAN sběrnici, mohou se výsledky na fyzickém médiu mírně lišit a bylo by ideální provedení druhého kola kontrolního testování v případě nasazení na fyzické médium s konkrétním nasazením. Testování bylo prováděno na virtuálním PC Debian, které je již zmíněno výše v práci. Následující obrázek obsahuje přidělení systémových prostředků, které by mohly mít vliv na výsledky testování viz obr. 2.8: Systémové zdroje pro testování.



Obr. 2.8: Systémové zdroje pro testování.

2.5.1 Testovací scénář

V rámci testování byl měřen rozdíl v času přenášení stejné zprávy po nezabezpečené sběrnici a následně po zabezpečené sběrnici s šifrováním.

Pro toto měření bylo potřeba navrhnout jakým způsobem získat údaje o času pro již zmíněnou zabezpečenou a nezabezpečenou sběrnici. Návrh testovacího scénáře počítá se sadou X stejných zpráv, kdy by tyto zprávy byly posílané po CANBus sběrnici a následně by byl zaznamenán čas při odesílání a čas při přijetí zprávy. Tímto způsobem by byl zajištěn čas, který by prezentoval dobu, jak dlouho byla daná data přenášena. Pro zajištění, že data budou stejná pro testování obou verzí, bude potřeba mít tato data předgenerovaná, aby byly podmínky stejné. Dále je také ideální generovat data o stejné délce, aby byla doba přenášení zpráv co nejpodobnější a neměnila se délka přenášení kvůli velikostí dat. Je nutné mít také na paměti, že šifrování s využitím TLS 1.3 zvýší počet přenášených zpráv, ale jelikož testování je směřováno na dobu, jak dlouho jsou přenášena stejná data, je nutné u zabezpečené verze měřit čas započetí odesílání těchto dat až po jejich dokončení. V případě že jsou posílána data o maximální délce pro standardní CANbus (64 bitů), tak pro přenos těchto dat s využitím zabezpečené verze a všech operací souvisejících s tímto přenosem je nutné přenést 7 zpráv. Z tohoto důvodu je nutné, aby měřená data u zabezpečené verze započítaly čas od odeslání první zprávy, která souvisí s odesíláním 64 bitů informací, až po poslední zprávu, která zajišťuje doručení celé informace.

Pro testování byla napsána sada skriptu s využitím programovacího jazyka Python. Tyto skripty zajišťují vygenerování milionu zpráv v hexadecimálním zápisu, kdy prvních 8 znaků zprávy je vyhrazeno pro číselný identifikátor, který určuje číslo dané zprávy, následně je zpráva rozšířena o blok náhodných dat a při odesílání je

do zprávy přidán i čas v milisekundách. Díky tomu je případně možné kontrolovat, zda byly doručeny všechny zprávy. Následně byly vytvořené skripty, které zajišťují odesílání a zapisování času odeslané zprávy, pro oba dva typy komunikace. Následně byly vytvořené skripty pro zachycování zpráv a zápis času doručení. Takto zapsaná data jsou následně s využitím dalších skriptu zpracovaná do formy pro další práci, poslední skript zajišťuje nalezení maximální doby přenosu, minimální doby přenosu a vypočítání průměrné doby a mediánu pro všechny testované vzorky. Pro zabezpečenou komunikaci je ke každé zprávě připočítán čas potřebný pro šifrování. Všechny skripty, které byly využité jsou k nalezení v příloze.

První testovací scénář

Průběh testování byl navrhnut na minimálně 5 opakování, kdy při každém opakování by byly vygenerovány unikátní zprávy pro každé kolo testování, aby byl testovací vzorek co největší a počet těchto zpráv byl stanoven na 1 000 000. Tento počet byl zvolen, jelikož se jedná o poměrně velký vzorek dat, které je tak možné analyzovat a díky tomuto velkému množství dat by měla průměrná data být poměrně průkazná. Měření probíhalo formou „zaplavování“ sběrnice informacemi. Respektive, nebyla ponechána žádná časová prodleva mezi požadavkem na odeslání zpráv. Následující tabulky obsahují průměr, medián a maximální dobu přenosu dat. Testovací vzorek je přenos 1 000 000 zpráv bez jakéhokoli módu vstupního řízení či koordinování odesílání zpráv. Tyto data jsou průměrem 5 měření, tab. 2.1 a tab. 2.2.

Tab. 2.1: Zabezpečená komunikace, přenos milionu zpráv bez moderování odesílání.

Popisek	Čas v [ms]
`Medián:`	2652
`Průměr:`	2647
`Max:`	3702

Tab. 2.2: Nezabezpečená komunikace, přenos milionu zpráv bez moderování odesílání.

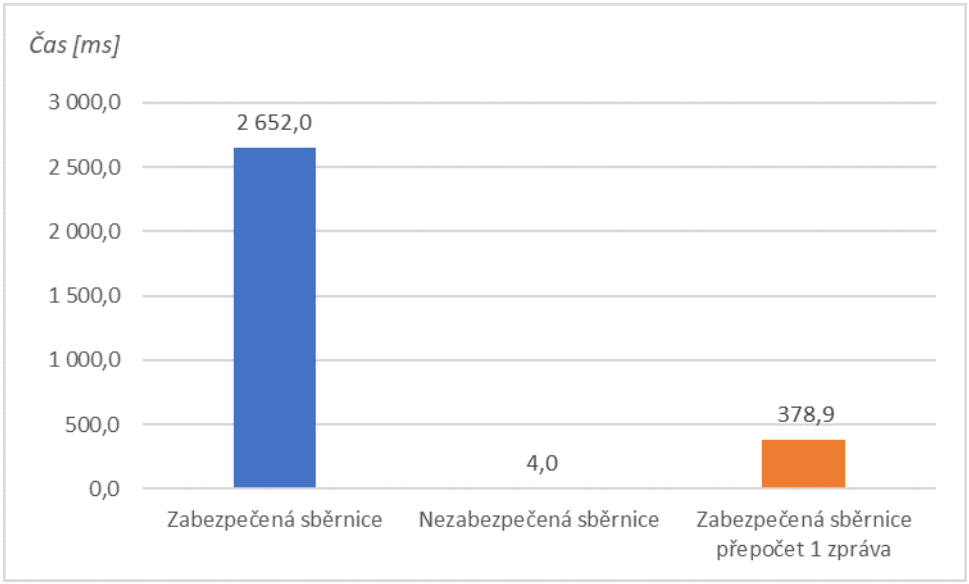
Popisek	Čas v [ms]
`Medián:`	4
`Průměr:`	4,28
`Max:`	85

Z naměřených dat je patrné, že přenos informací přes zabezpečenou sběrnici je delší než přes nezabezpečenou sběrnici, která neimplementuje žádné zabezpečení, což bylo předpokládáno. Pro úplnost je asi dobré uvést, jak dlouho by byla přenášena 1 zabezpečená zpráva, aby bylo jasné porovnání pro přenos celých dat a možnost porovnání času pro jednotlivé zprávy. Viz data v tab. 2.3.

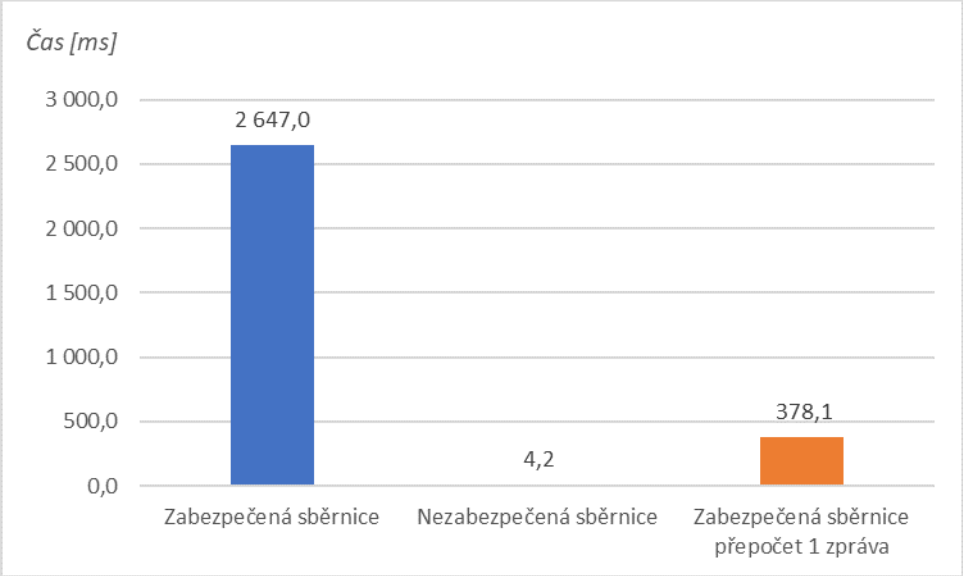
Tab. 2.3: Zabezpečená komunikace přepočet času pro jednu zprávu, přenos milionu zpráv, bez moderování odesílání

Popisek	Čas v [ms]
`Medián:`	378,9
`Průměr:`	278,1
`Max:`	528,9

Následující grafy, které jsou zobrazeny na obr. 2.9 a obr. 2.10, zobrazují porovnání mediánu a průměru pro milion přenášených z práv, bez moderace přenášení.



Obr. 2.9: Graf zobrazující medián pro přenos milionu stejně velkých zpráv, bez moderování odesílaní.



Obr. 2.10: Graf zobrazující průměr pro přenos milionu stejně velkých zpráv, bez moderování odesílaní.

Závěr pro první testovací scénář

Časy přenosu pro zabezpečenou sběrnici s těmito daty působí dost katastroficky. Přenosové rychlosti zabezpečené komunikace blížící se v průměru k 2,67s / data o délce 64 bitů, nejsou vůbec dobré. Problém s dobou přenosu v tomto případě dost možná vznikat na úrovni testování virtuální sběrnice a fungování testovacích skriptů. Při testování s nižšími vzorky dat, vychází výsledky výrazně lépe. Takto nekonzistentní výsledek není vůbec průkazný.

Druhý testovací scénář

Aby bylo možné validně zkontrolovat jakou skutečnou dobu jsou data skutečně přenášena a dostat tak adekvátní porovnání vlivu zabezpečené komunikace na CAN, bude potřeba mírně přepracovat testování. Pro zapracování úpravy, aby byl měřený skutečný čas vlivu šifrování na sběrnici aplikujeme moderovaní odesílání pro zabezpečenou i nezabezpečenou komunikaci. V tomto případě aplikujeme pravidlo, že mezi jednotlivými zprávami aplikujeme čekací dobu 100 ms, aby jsme zajistili skutečné časy, jak dlouho probíhá přenos šifrované a nešifrované zprávy.

Nový testovací scénář. Budeme odesílat 100 000 zpráv v módu moderovaného odesílání. Toto odesílání zopakujeme 5x a vypočítáme průměr doby odesílání. V tab. 2.4, jsou vidět časy, kterých bylo dosaženo na zabezpečené sběrnici. V následující tab. 2.5 je patrné jak dlouho byla data přenášena po nezabezpečené sběrnici.

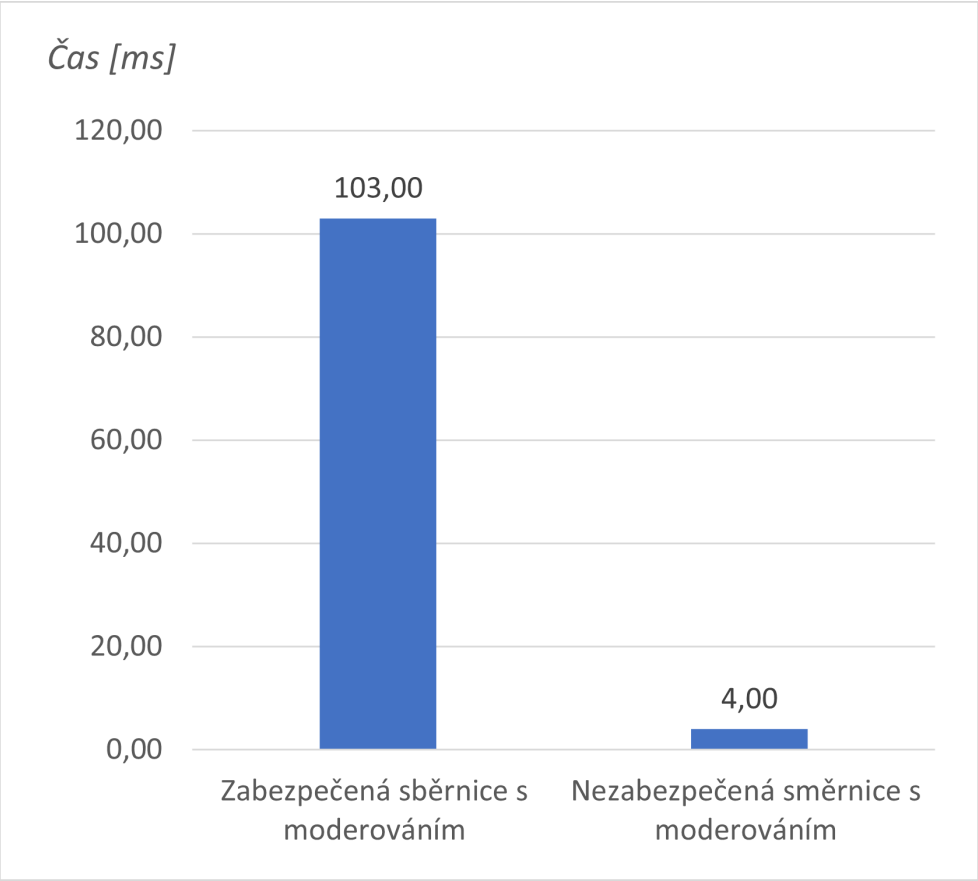
Tab. 2.4: Zabezpečená komunikace, přenos sto tisíc zpráv moderování odesílání.

Popisek	Čas v [ms]
`Medián:`	103
`Průměr:`	103,9
`Max:`	171

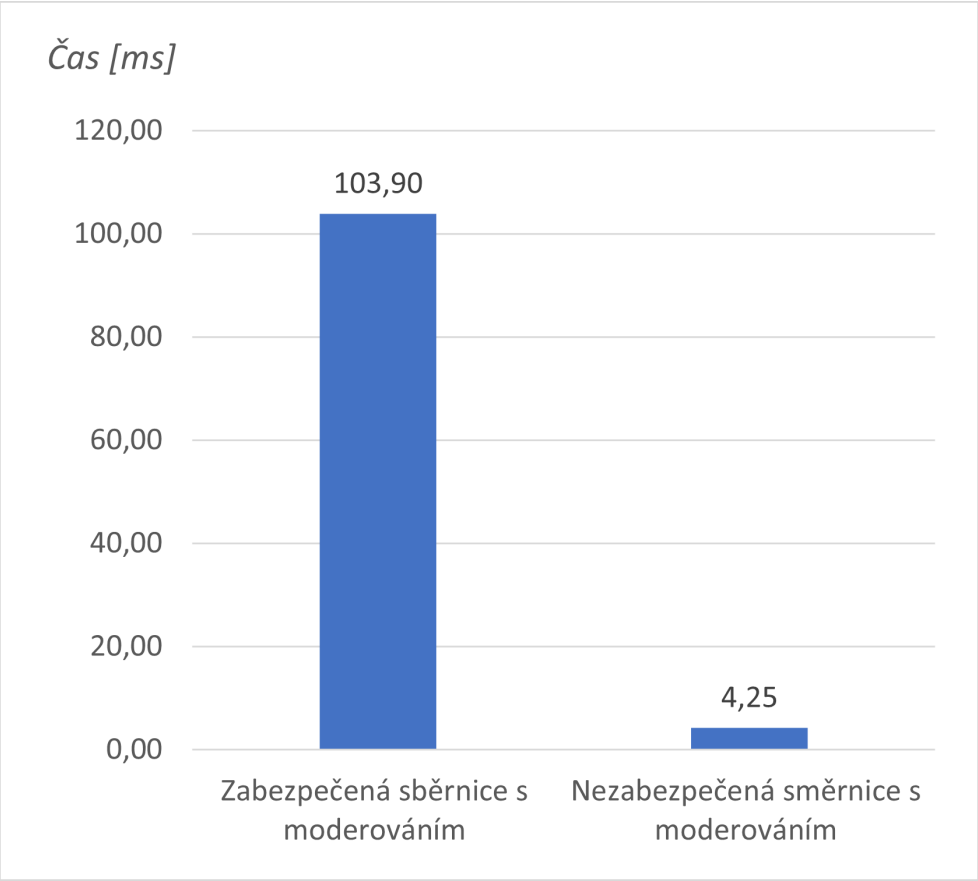
Tab. 2.5: Nezabezpečená komunikace, přenos sto tisíc zpráv moderování odesílání.

Popisek	Čas v [ms]
`Medián:`	4
`Průměr:`	4,25
`Max:`	85

Následující obrázky obr. 2.11 a obr. 2.12 grafu zobrazují porovnání zabezpečené a nezabezpečené verze, při odesílání 100 000 zpráv s moderování odesíláni.



Obr. 2.11: Graf zobrazující medián pro přenos sto tisíc zpráv, s moderování odesílaní.



Obr. 2.12: Graf zobrazující průměr pro přenos sto tisíc zpráv, s moderování odesílaní.

Závěr pro druhý testovací scénář

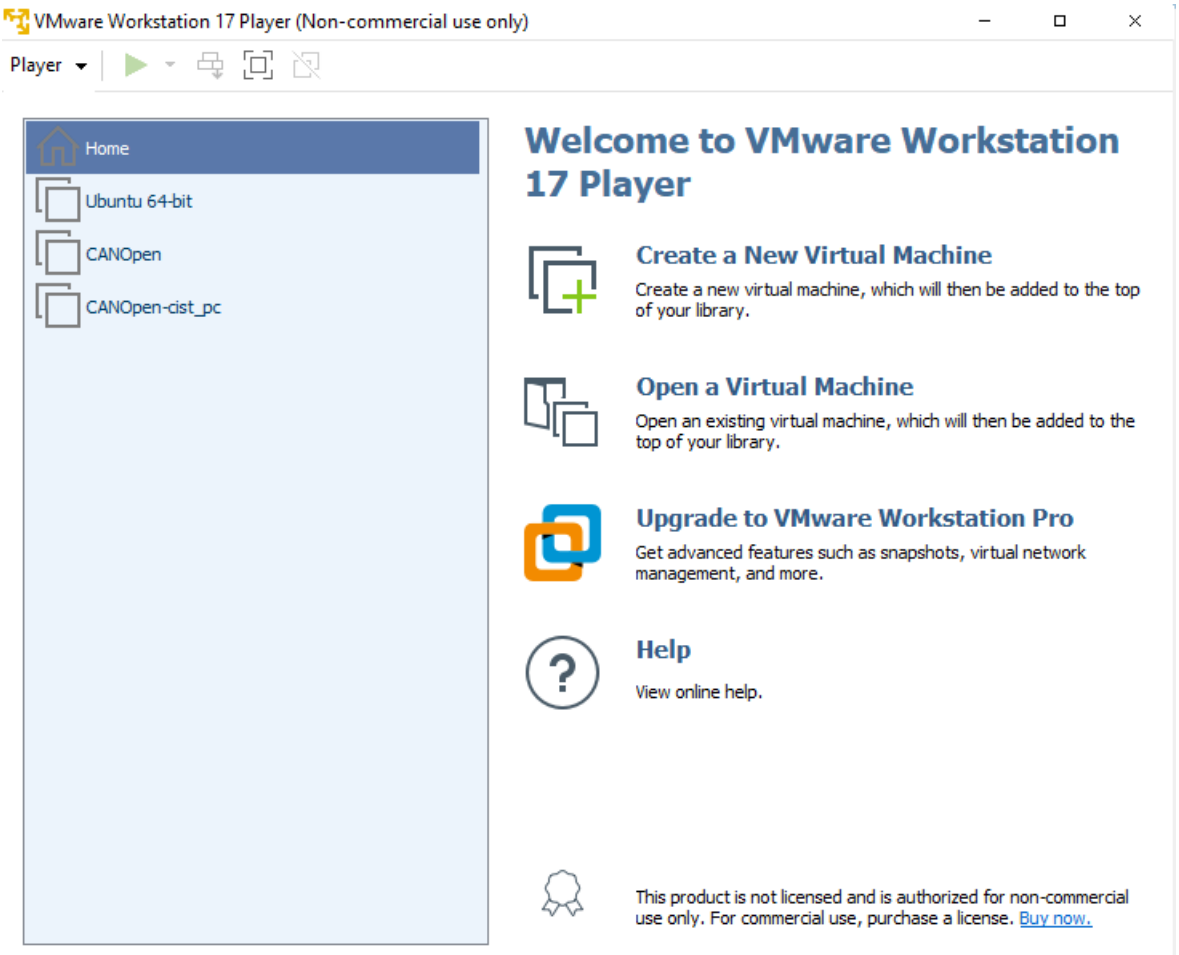
Časy v druhém scénáři jsou výrazně lepší a výrazně více konzistentní. Aplikováním moderování dostáváme výrazně přesnější data, které porovnávají skutečný vliv testování na přenášené zprávy. Dosahované rychlosti v průměru 103,9 ms, je velmi slušné a adekvátní vzhledem k operacím, které se s daty provádí pro jejich odeslání. Pro úplnost bylo toto měření provedeno i na vzorku 1 000 000 zpráv, aby bylo možné ověřit, zda budou výsledky konzistentní, toto měření potvrdilo časy z měření se 100 000 prvky.

3 Postupy

V této části jsou popsané postupy pro vytvoření a nastavení testovacího prostředí a implementace zabezpečení na CAN.

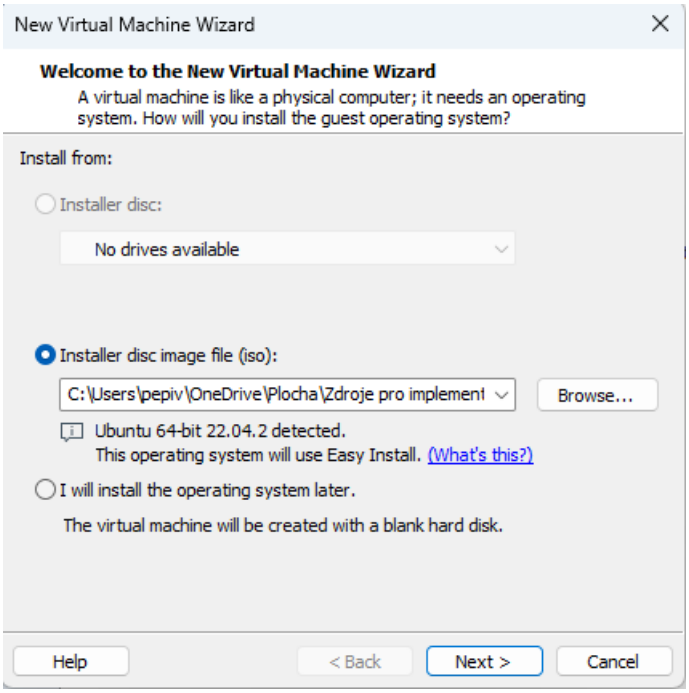
3.1 Vytvoření virtuálního počítače s operačním systémem Debian

Pro testování budeme využívat virtualizovaný operační systém Debian 22.04.02. Pro virtualizaci Debianu využijeme programu pro virtualizaci operačních systémů VMware Workstation Player 17.0.1. Po nainstalování VMware a spuštění vytvoříme nový virtuální počítač kliknutím na „Create a New Virtual Machine“. Viz obr. 3.1: Vytvoření nového virtuálního počítače.



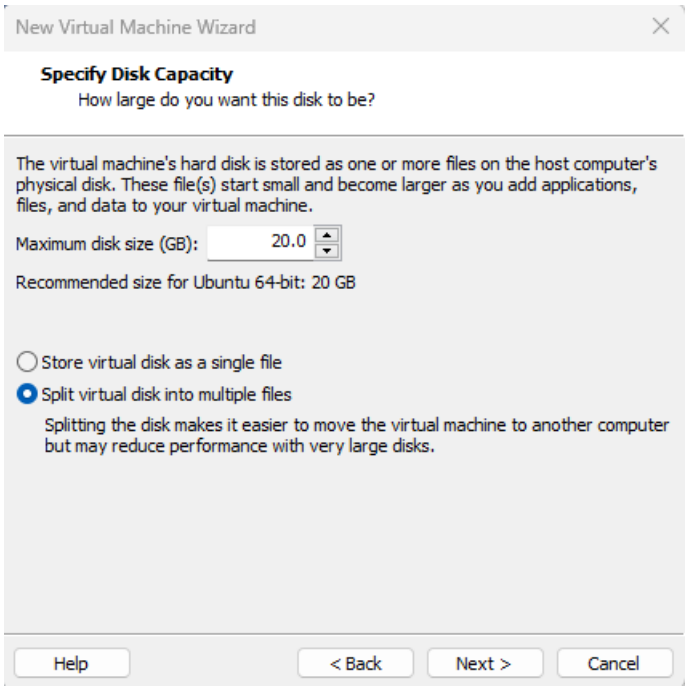
Obr. 3.1: Vytvoření nového virtuálního počítače.

Následně pokračujeme tím, že vybereme stažený iso soubor, který obsahuje kompletní instalační soubor operačního systému. V našem případě se jedná o instalační soubor pro operační systém Debian 22.04.02. Viz obr. 3.2: Zvolení instalačního disku.



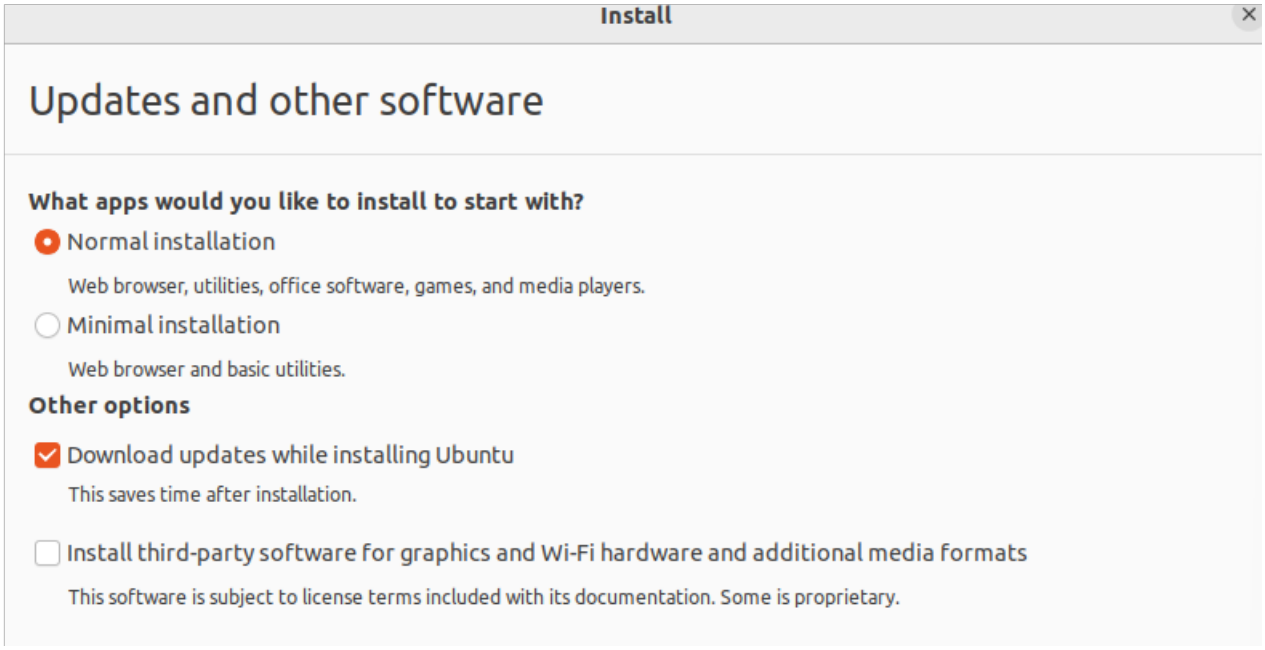
Obr. 3.2: Zvolení instalačního disku.

Následně je potřebné vyplnit obecné informace (jméno uživatele, heslo). Tyto parametry lze zvolit podle vlastního rozhodnutí, nezáleží na nich pro instalaci. V dalším kroku nastavíme pojmenování našeho virtuálního počítače a pozici, kam se má uložit na našem skutečném počítači. Následně je nutné nastavit velikost disku, který poskytneme virtuálnímu počítači. Velikost disku poskytnutého virtuálnímu počítači záleží na hardwarových možnostech našeho fyzického počítače. Pro zprovoznění této testovací verze stačí zvolit 20 GB, jak je patrné na obr. 3.3. Následně se dá pokračovat na obecný přehled připravované instalace, v rámci které se ujistíme, že je systému poskytnuto minimálně 2GB operační paměti, pokud je vše v pořádku je možné pokračovat.



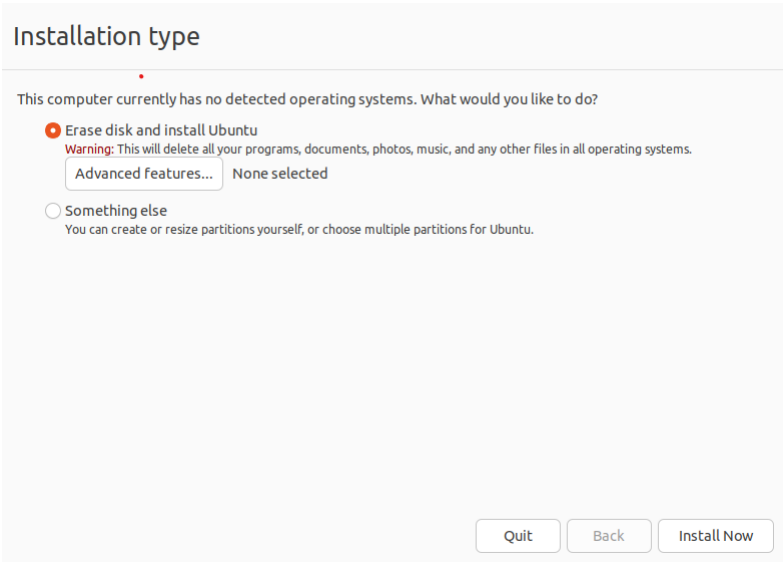
Obr. 3.3: Přidělení místa virtuálnímu disku.

Klepnutím na tlačítko „Finish“ posouvá tento proces do instalace samotného systému. Jako první věc kterou bude potřeba je vybrání klávesnice, která bude v systému využívána. K dalším krokům instalace pokračujeme tlačítkem „Continue“. Viz obr. 3.4: Typ instalace. Následující částí prochází beze změny tlačítkem „Continue“, s ponecháním „Normal installation“. Viz obr. 3.4: Typ instalace.



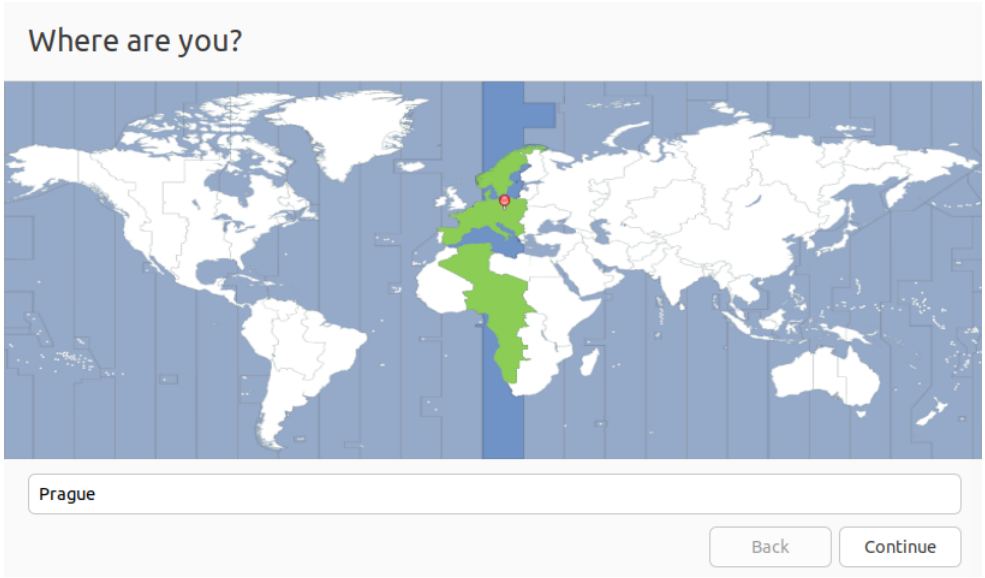
Obr. 3.4: Typ instalace.

V rámci části, kde se nastavuje disk, vybereme „Erase disk“ a pokračujeme na „Install Ubuntu“. Viz obr. 3.5: Nastavení pevného disku 56. Nastavení disku viz obr. 3.5: Nastavení pevného disku.



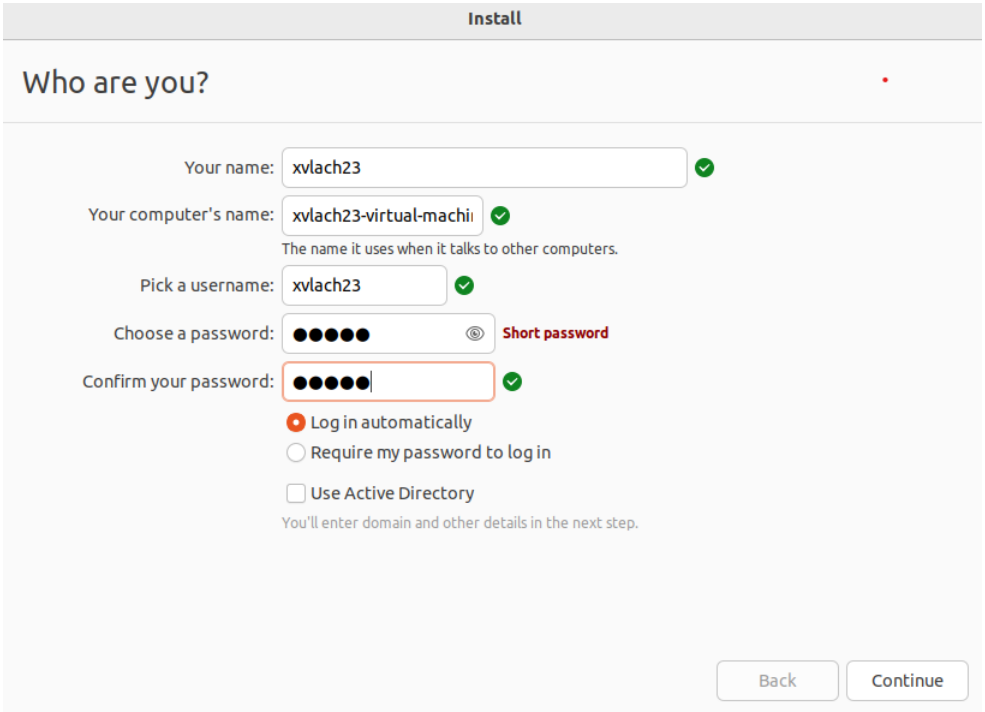
Obr. 3.5: Nastavení pevného disku.

Jednou z posledních věcí, kterou bude potřeba provést, je nastavení časového pásma. Opět pokračujeme stisknutím „Continue“. Viz obr. 3.6: Nastavení časového pásma.



Obr. 3.6: Nastavení časového pásma.

Posledním krokem instalace je vyplnění obecných parametrů o uživateli, který bude zároveň administrátor. Viz obr. 3.7: Nastavení systémového uživatele.



Obr. 3.7: Nastavení systémového uživatele.

Po dokončení instalace bude nově instalovaný operační systém vyžadovat restart, čímž finálně dokončíme instalaci celého virtuálního operačního systému.

3.2 Implementace wolfSSL

V rámci této části se zaměříme na to, jak instalovat knihovnu wolfSSL na virtuálním systému Debian 22.04.02, jehož instalace byla popsána v dřívějším kroku, jak instalovat rozšíření wolfSSL pro CAN sběrnici a jak vytvořit virtuální sběrnici CAN s využitím can-utils.

3.2.1 Instalace wolfSSL

Pro instalaci wolfSSL využijeme možnosti stažení kódů knihovny z oficiálního githubu společnosti wolfSSL (https://github.com/wolfSSL/wolfssl). Pro tento postup budeme využívat terminálu, který je ve virtuálním operačním systému už součástí původní instalace. Pro práci v terminálu slouží příkazy. K pohybu v adresářové struktuře slouží příkaz „cd <nazev adresare>“. Využitím lomítka je možné psát i delší cestu k souborům. Pro výstup z daného adresáře využijeme příkazu „cd ..“Ábychom se lépe orientovali, ve kterém se nacházíme adresáři je možné využít identifikátor „ /<adresář>$“, nejedná se o součást příkazu.

Otevřeme terminál a začneme s instalací potřebných baličků, které budou důležité pro další části instalace. Prvním z balíčků, který budeme potřebovat, bude balíček git. Bude využíván pro možnost práce s gitem přímo z terminálu. V tomto případě bude užíván pro stažení potřebných souborů a kódů pro knihovnu wolfSSL. Instalaci provedeme následujícím příkazem, po odeslání tohoto příkazu bude nutné zadat heslo pro administrátora, který byl vytvořen v průběhu instalace virtuálního systému. (Verze použita v této práci: git version 2.34.1.)

```
~\$ sudo apt-get install git
```

Dalším důležitým balíčkem, který budeme chtít instalovat, je balíček build-essential. Tento balíček obsahuje řadu funkcí, které jsou nutné pro konfiguraci a sestavení knihovny wolfSSL a její nástavby. Instalaci provedeme následujícím příkazem. Opět bude nutné zadat heslo pro administrátora systému. (Verze využitá v této práci: 12.9ubuntu3.)

```
~\$ sudo apt-get install build-essential
```

Po nainstalování těchto důležitých balíčků je možné přejít ke stažení samotné knihovny wolfSSL z oficiálného githubu společnosti wolfSSL. Pro tenhle krok bude využit dříve instalovaný balíček git. Stažení knihovny provedeme následujícím příkazem.

```
~\$ git clone https://github.com/wolfSSL/wolfssl.git
```

Po dokončení stahování zkontrolujeme, že je knihovna opravdu stažena pomocí příkazu „ls“, který vypíše obsah adresáře, ve kterém se aktuálně nacházíme. Mezi zobrazenými položkami by mělo být wolfSSL.

```
~\$ ls

Desktop    Downloads  Pictures  snap       Videos
Documents  Music      Public    Templates  wolfssl
```

V případě, že všechno proběhlo v pořádku, přejdeme do adresáře s knihovnou, aby bylo možné pokračovat v sestavení a instalaci. V tomto případě tedy zadáváme příkaz „cd“.

```
~\$ cd wolfssl
```

Obsah adresáře wolfssl je opět možné zkontrolovat příkazem „ls“. Adresář by neměl být prázdný. Pokud je vše v pořádku, můžeme přejít k sestavování a konfiguraci samotné knihovny. Konfiguraci knihovny zajistí příkaz „configure“ s dodatečnými parametry, které zajistí, že knihovna bude sestavená tak, aby efektivněji pracovala s pamětí a byla instalována jako statická knihovna. Následně ji sestavíme a instalujeme.

```
~/wolfssl\$ ./configure --enable-shared --enable-static

~/wolfssl\$ make

~/wolfssl\$ sudo make install
```

Po provedení těchto kroků by mělo být možné zobrazit verzi instalovaného wolfSSL s využitím příkazu „version“.

```
~\$ wolfssl-config --version

5.6.0
```

Pokud se zobrazila verze instalovaného wolfSSL, proběhlo všechno správně a čisté wolfSSL je již možné využívat.

3.2.2 Instalace rozšíření wolfSSL pro CAN

V rámci této části bude řešeno stažení a instalace rozšíření wolfSSL, které podporuje aplikaci na sběrnici CAN. Tato část postupu předpokládá instalovaný virtuální systém Debian a sestavenou a instalovanou knihovnu wolfSSL.

Pro stažení rozšíření pro wolfSSL bude opět využito githubu a stáhneme wolfssl-examples. Z tohoto důvodu je opět nutné mít instalovaný balíček git. Pro stažení kódu využijeme opět příkazu „git clone“.

```
~\$ git clone https://github.com/wolfSSL/wolfssl-examples.git
```

Po ukončení stažení je možné zkontrolovat správně stažené wolfssl-examples. Využijeme příkazu „ls“ pro zobrazení obsahu adresáře, který by měl obsahovat wolfssl-examples.

```
~\$ ls

Desktop    Downloads  Pictures  snap       Videos   wolfssl-examples
Documents  Music      Public    Templates  wolfssl
```

V rámci dalšího postupu je potřeba opět přejít do adresáře wolfssl a do instalace bude nutné přidat WOLFSSL_ISOTP pro umožnění ISO-TP funkcí, které budou potřeba. Toto provedeme pomocí příkazu „configure“ a opět provedeme sestavení knihovny a doinstalujeme toto rozšíření.

```
~/wolfssl\$ ./configure CFLAGS="-DWOLFSSL_ISOTP"

~/wolfssl\$ make all

~/wolfssl\$ sudo make install
```

Následně se přesuneme do staženého adresáře wolfssl-examples/can-bus, kde jsou připravené kódy pro fungování strany serveru, klienta a pro fungování implementovaného TLS. Také je zde připravený skript, který vygeneruje certifikáty a klíče pro stranu serveru a klienta s využitím RSA 2048.

```
~\$ cd wolfssl-examples/can-bus/

~/wolfssl-examples/canbus\$ ./generate_ssl.sh

~/wolfssl-examples/canbus\$ make
```

Tyto operace by měly vést k vytvoření certifikátu a klíče pro stranu serveru a stranu klienta.

3.2.3 Vytvoření virtuální CAN sběrnice

V této části postupu se zaměříme na vytvoření virtuální CAN sběrnice, kterou budeme využívat pro demonstraci zabezpečení s využitím wolfssl. Pro tyto operace využijeme balíček can-utils. (Verze využitá v této práci: 2020.11.0-1).

```
~\$ sudo apt-get install can-utils

~\$ sudo modprobe vcan

~\$ sudo ip link add dev vcan0 type vcan
```

3.2.4 Testování funkčnosti implementace

V rámci této části se zaměříme na poslední část a na to, jak připojit stranu serveru a stranu klienta na virtuální sběrnici CAN a jak ověřit, že zabezpečená komunikace funguje. Na začátek zapneme vytvořenou virtuální sběrnici CAN „vcan0“.

```
~\$ sudo ip link set vcan0 up
```

Na další část je třeba otevřít dvě okna terminálu, aby bylo možné zapnout stranu serveru v jednom terminálu a ve druhém stranu klienta. Pro tuto část je nutné mít zapnutou sběrnici vcan0 a v každém z terminálů přejít do wolfssl-examples/canbus v rámci kterého máme kód pro stranu serveru a stranu klienta.

Strana serveru:

```
~\$ cd wolfssl-examples/can-bus/

~/wolfssl-examples/canbus\$ ./server vcan0
```

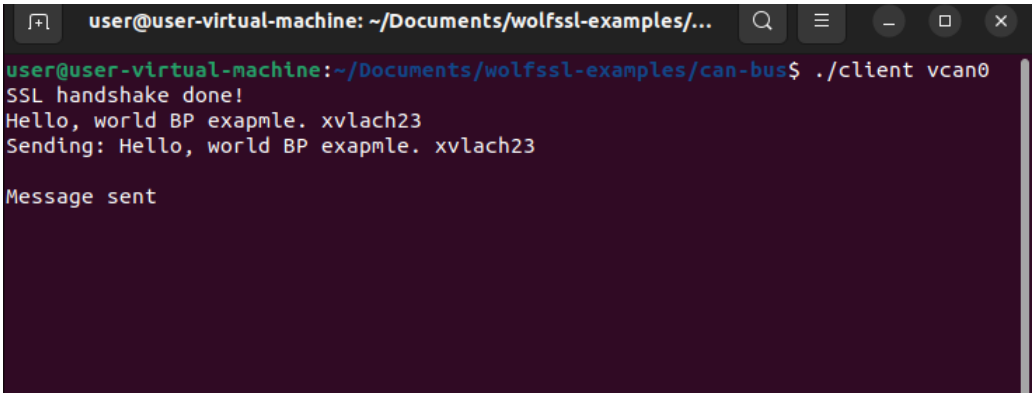
Strana klienta:

```
~\$ cd wolfssl-examples/can-bus/

~/wolfssl-examples/canbus\$ ./client vcan0
```

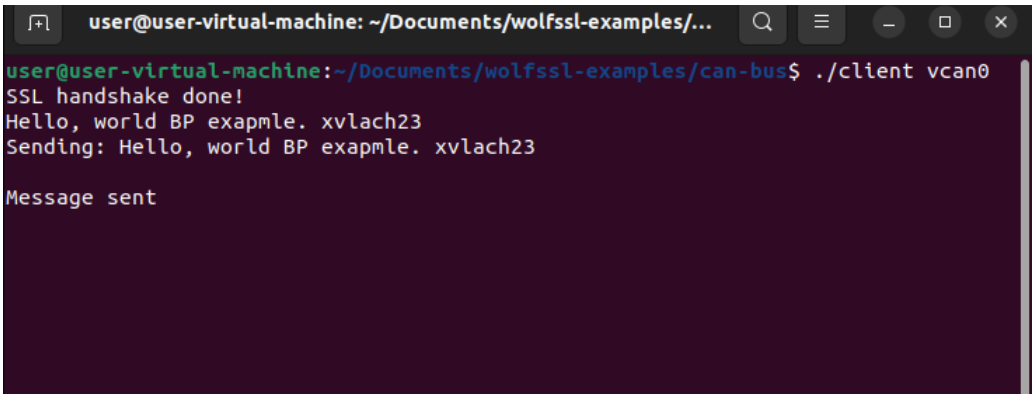
Pokud je vše správně a stranám se podaří správně navázat komunikaci a navzájem se ověřit, vypíše se do obou terminálů hláška o úspěšném navázání spojení.

```
SSL handshake done!
```

Pokud se objeví tato zpráva, je jasné, že všechno funguje správně a je vytvořený zabezpečený komunikační kanál mezi klientem a serverem. Zprávy je možné odesílat prostým napsáním do terminálu, kde je zapnutý klient a následným stisknutím klávesy „Enter“, což potvrdí odeslání zprávy. Odeslanou zprávu je následně možné vidět na straně serveru. Viz obr. 3.8: Strana klienta který zobrazuje spuštění strany klienta a následné odesílání zpráv a obr. 3.9: Strana serveru, kde je možné vidět spuštění strany serveru a ověření doručení zprávy od klienta.



Obr. 3.8: Strana klienta.



Obr. 3.9: Strana serveru.

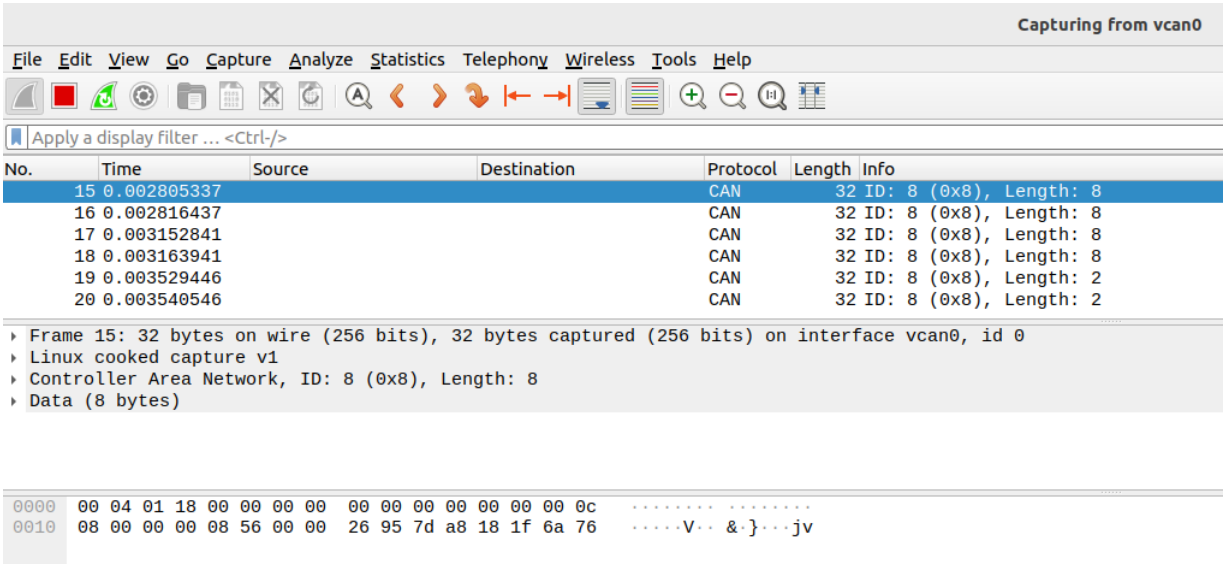
Pro ověření, že komunikace přes virtuální sběrnici probíhá, je možné v novém terminálu využít příkazu „candump“, který zobrazí posílanou komunikaci po této sběrnici a také díky přepínači „-tz“ zobrazí časové razítko posílaných CAN zpráv. Pokud je na sběrnici nějaký provoz, zobrazí se.

```
~\$ candump -tz vcan0

 (000.000000)  vcan0  008   [8]  10 18 17 03 03 00 13 3E
 (000.000173)  vcan0  007   [3]  30 00 F0
 (000.000326)  vcan0  008   [8]  21 28 5C D1 10 B3 7C 7B
 (000.000498)  vcan0  008   [8]  22 78 70 8F E0 73 14 AB
 (000.000660)  vcan0  008   [5]  23 3B 58 69 37
```

Pro demonstraci šifrování na sběrnici vcan0 využijeme programu wireshark, který nainstalujeme (vyskakovací hlášení při instalaci wiresharku odmítáme označením „No“) a spustíme tak, aby poslouchal na sběrnici vcan0. (Verze využitá v této práci: Wireshark 3.6.2).

```
~\$ sudo apt-get install wireshark

~\$ sudo wireshark -k -i vcan0
```

Následně pro demonstraci můžeme vidět nesmyslná data přenášená po sběrnici vcan0 ve spodní části snímku z předchozí ukázky fungování komunikace mezi stranami serveru a klienta. Viz obr. 3.10: Zobrazená data pomoci wireshark je možné vidět přenesená data a jejich prezentaci ve spodní části obrázku.



Obr. 3.10: Zobrazená data pomoci wireshark.

3.3 Testování zabezpečené komunikace

Tato část se věnuje hrubému popisu, jak případně provést testování s využitím skriptů, které jsou v příloze. Všechny skripty jsou pojmenované podle daného úkolu, který provádí. Všechny skripty mají poměrně detailně okomentovaný kód, který popisuje jejich fungování, proto se v tomto postupu zaměříme více na teoretické využívání. Před testováním je nutné ve skriptech nastavit správné cesty k souborům. Pro spouštění těchto skriptu je také nutné mít instalovaný Python 3. Tuto instalaci provedeme následujícím příkazem.

```
~\$ sudo apt install python3
```

Následně je také potřeba doinstalovat potřebné balíčky, které budou dále využívány.

```
~\$ sudo apt-get install socat moreutils
```

V rámci dalšího postupu je nejprve nutné spustit skript, který zajišťuje vygenerování daného počtu zpráv, který zadáváme jako druhý argument. Následující příkaz zajistí vygenerování 10 zpráv.

```
~\$ python3 generovani_zprav.py 10
```

Následně je potřeba využít skripty na odchytávání dat a jejich odesílání, tyto skripty jsou dle jména rozdělené pro zabezpečený a nezabezpečený CAN.

3.3.1 Nezabezpečený CAN testování

Pro nezabezpečený CAN stačí spustit skript na odposlouchávání sběrnice, který odchytává data a následně spustit skript na jejich odesílání v novém okně terminálu.

Terminál 1:

```
~\$ python3 odchytavani_zprav_2.0_cisty_can.py
```

Terminál 2:

```
~\$ python3 odchytavani_zprav_2.0_cisty_can.py
```

Po dokončení těchto operací ukončíme odchytávání zpráv klávesovou kombinací „Ctrl + C“. Naměřená data se uloží do samostatných souborů. Pro další zpracování dat je potřeba využít skriptu, který data připraví. Pro zpracování připravených dat na výsledek slouží další skript, který vypíše mix, max, medián a průměr.

```
~\$ python3 priprav_data_cisty_can.py

~\$ python3 vypocet.py
```

3.3.2 Zabezpečený CAN testování

Pro zabezpečený CAN je potřebné spustit stranu serveru, která je připojená k vytvořenému virtuálnímu rozhraní CAN. Následně je potřeba spustit skript na odposlouchávání sběrnice, který odchytává data. Poslední skript zajišťuje odesílání zpráv.

Terminál 1:

```
~/wolfssl-examples/can-bus\$ ./server vcan0
```

Terminál 2:

```
~/wolfssl-examples/can-bus\$ python3 odchytavani_zprav_2.0
   _zabezpeceny_can.py
```

Terminál 3:

```
~/wolfssl-examples/can-bus\$ python3 odesilani_zprav_2.0
   _zabezpeceny_CAN.py
```

Po dokončení těchto operací ukončíme odchytávání zpráv a server klávesovou kombinací „Ctrl + C“. Naměřená data se uloží do samostatných souborů. Pro další zpracování dat je potřeba využít dvou skriptu, které data připraví. Pro zpracování připravených dat na výsledek slouží další skript, který vypíše mix, max, medián a průměr.

```
~/wolfssl-examples/can-bus\$ python3 ocistit_soubor_zabezpeceny_can.py

~/wolfssl-examples/can-bus\$ python3 priprav_data_zabezpeceny_can.py

~/wolfssl-examples/can-bus\$ python3 vypocet.py
```

Závěr

Cílem práce bylo vytvořit demonstrátor pro zabezpečenou komunikaci s protokoly CAN a CANopen s využitím protokolových nástaveb secure CAN, secure CANopen. V rámci analýzy a průzkumu bylo této části dáno opravdu hodně úsilí, bohužel více než zmínky o tom, že by se mělo jednat o protokoly CAN s implementovaným zabezpečením se bohužel nepodařilo nalézt. Tuto informaci následně i potvrdili pracovníci CiA, se kterými byla tato problematika konzultována. Zabezpečené verze pro tento protokol nejsou aktuálně oficiálně standardizovány, ale jedná se o problematiku, která by měla být do budoucna řešena. Mezi další cíle práce byla analýza zdrojů. U této části byl ze začátku opravdu problém nalézt informace, které by se nevěnovali problematice CAN s nasazením pro automobilový průmysl, z tohoto důvodu byly časem využití i obecné informace z těchto zdrojů. Dalším z cílů práce byla analýza vektoru útoku. V rámci této části by bylo možné provést lepší analýzu, než která byla provedena, celkově této části nebylo věnováno tolik pozornosti a věnuje se jí pouze krátká pasáž na začátku práce a následně část popisující komunikaci s pracovníky CiA. Dalším z cílů této práce byla analýza aktuálně dostupných zdrojů. Této části byly hledány existující implementace, řešící bezpečnost pro protokol CAN. U každé nalezené implementace byl kontaktován dodavatel, aby bylo možné data ověřit, či případně zjistit některé podrobnosti. Je nutné podotknout, že ne vždy bylo kontaktování úspěšné, ale touto cestou se povedlo získat cenné informace. Z této části také vychází, že problematice bezpečnostních implementací se věnuje více různých řešení a vždy je nutné uvážit, zda není některé z řešení více vyhovující pro požadované fungování zabezpečeného CANu v konkrétním případě. V rámci druhé části této analýzy byly prozkoumávány větší krypto knihovny. Z této analýzy nejlépe vzešla knihovna wolfSS, která má podrobnou dokumentaci k produktům, veřejně vystavené benchmarky a jak bylo i v praxi zjištěno, velmi dobrou technickou podporu a již existující implementaci pro CAN. Teoretickému výzkumu měl následovat praktický návrh zabezpečené implementace. Tento návrh v rámci práce byl vytvořen, ale při kritickém pohledu je možné konstatovat, že se nejedná o ideální návrh, jelikož problematiky bezpečnosti je řešena poměrně povrchově. Teoretický návrh následuje praktická implementace bezpečnostních mechanismů pro protokol CAN. V rámci této části bylo využito již zmíněného wolfSSL, v rámci kterého byla na virtuálním testovacím prostředí nasazená takto knihovna společně s implementací pro CAN. V rámci řešení této části, byla také zahájena komunikace s pracovníky wolfSSL. V rámci této komunikace byl opraven postup pro nasazení implementace pro CAN, na oficiálním wolfSSL githubu tohoto rozšíření. Při řešení implementace tohoto zabezpečení také proběhl meeting přes platformu Zoom s pracovníky, kde byly sdíleny cenné zkušenosti a myšlenky tohoto výzkumu. Po praktické implementaci zabezpe-

čené komunikace pro CAN bylo prováděno testování. Testování se primárně zaměřilo na výkonnostní testy. V rámci kterých bylo zjištěno, jak dané bezpečnostní rozšíření ovlivňuje komunikaci, v případě nemoderovaného testování nebyla data moc dobrá. Moderované testování přineslo uspokojivější výsledky, jediný problém oproti původnímu plánování, je testování přenosových vastností s využitím jiných kryptografických algoritmů. Věcí na kterou se z časových důvodu také nepovedlo zaměřit dostatečně, jsou scénáře pro bezpečnostní testy, v rámci práce s implementovaným zabezpečením bylo testováno, že přenášená data po sběrnici jsou šifrována apod., ale na tuto část nevznikl, žádný testovací scénář.

Při celkovém ohlédnutí a zhodnocení práce je možné zhodnotit očekávání proti výsledkům. Podařilo se nalézt a implementovat otevřené zabezpečené řešení pro protokol CAN, které umožňuje využívání symetrické i asymetrické kryptografie, umožňuje další práci a rozvíjení, zajišťuje autentizaci a šifrování komunikace. Jak testování také ukázalo, vliv na přenosu zpráv je velmi uspokojivý. Věci kterou se v kontextu praktické implementace nepodařilo úplně řešit, je optimalizace implementovaného mechanismu, což se nejvíce projevilo u nemoderovaného testování a také se nepovedlo testovat implementace využívající jiné kryptografické funkce.

Literatura

[1] Robert Bosch GmbH, *SPECIFICATION, C. A. N. Bosch.*, Postfach 1991, 50.

[2] MERKEL, Martin; SCHLEGEL, Christian. *CANopen in X-ray systems*. IXXAT Automation GmbH, 2008.

[3] BOZDAL, Mehmet, et al. *Evaluation of can bus security challenges*. Sensors, 2020, 20.8: 2364.

[4] SIDDIQUI, Ali Shuja, et al. *Secure communication over CANBus.* In: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE, 2017. p. 1264-1267.

[5] CAN-CiA *Physical layer options*. CiA [online] . [cit. 2023-05-23]. Dostupné z: https://www.can-cia.org/can-knowledge/can/systemdesign-can-physicallayer/

[6] WOO, Samuel, et al. *A practical security architecture for in-vehicle CAN-FD*. IEEE Transactions on Intelligent Transportation Systems, 2016, 17.8: 2248-2261.

[7] HARTWICH, Florian; BOSCH, Robert. *Introducing CAN XL into CAN Networks. future*, 2015, 11898: 1.

[8] CAN-CiA – *CANopen The standardized embedded network*. CiA [online] . [cit. 2023-05-23]. Dostupné z: https://www.can-cia.org/canopen/

[9] PFEIFFER, Olaf; AYRE, Andrew; KEYDEL, Christian. *Embedded networking with CAN and CANopen*. Copperhill Media, 2008.

[10] TINDELL, K. *Can bus security attacks on can bus and their mitigation*. Canis automotive labs 2020.

[11] WOO, Samuel; JO, Hyo Jin; LEE, Dong Hoon. *A practical wireless attack on the connected car and security protocol for in-vehicle CAN*. IEEE Transactions on intelligent transportation systems, 2014, 16.2: 993-1006.

[12] PFEIFFER, Olaf. *Implementing scalable can security with cancrypt*. Embedded Systems Academy, 2017.

[13] FARAG, Wael A. *CANTrack: Enhancing automotive CAN bus security using intuitive encryption algorithms*. In: 2017 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO). IEEE, 2017. p. 1-5.

[14] RESCORLA, Eric. *The transport layer security (TLS) protocol version 1.3.* 2018.

[15] DAVIS, Robert I., et al. *Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised.* Real-Time Systems, 2007, 35: 239-272.

Seznam příloh

A Obsah elektronické přílohy

V této části je popsaný obsah elektronické přílohy k této práci. Příloha obsahuje instalační obraz pro operační systém Debian. A také testovací skripty pro testování praktické části.

- `/` ..... kořenový adresář přiloženého archivu
  - `scripty_na_testovani` ..... obsahuje skripty pro testování
    - `generovani_zprav.py`
    - `odchytavani_zprav_2.0_cisty_can.py`
    - `odesilani_zprav_2.0_cisty_can.py`
    - `priprav_data_cisty_can.py`
    - `odchytavani_zprav_2.0_zabezpeceny_can.py`
    - `odesilani_zprav_2.0_zabezpeceny_can.py`
    - `ocistit_soubor_zabezpeceny_can.py`
    - `priprav_data_zabezpeceny_can.py`
    - `vypocet.py`