VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 David Kolaja

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

ZABEZPEČENÍ ÚZKOPÁSMOVÉ LICENČNÍ BEZDRÁTOVÉ KOMUNIKACE
SECURING NARROWBAND WIRELESS COMMUNICATION IN LICENSED BAND

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **David Kolaja**
AUTHOR

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

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Informační bezpečnost**
Ústav telekomunikací

Student: David Kolaja ***ID:*** 195157
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Zabezpečení úzkopásmové licenční bezdrátové komunikace

POKYNY PRO VYPRACOVÁNÍ:

Student provede analýzu nových technologií LPWAN (Low Power Wide Area Network) s bližším zaměřením na celulární technologii NB-IoT (Narrow Band - Internet of Things). Bude provedena komplexní analýza komunikace, společně s jednotlivými procesy v síti, hrozbami, nedostatky a známými útoky. Teoretickým výstupem práce bude posouzení bezpečnostních aspektů technologie NB-IoT. Na tomto základě bude postaven vlastní návrh zabezpečení v aplikační vrstvě, které zajistí tzv. E2E bezpečnost (konec-konec). V neposlední řadě student provede výběr, návrh, formální ověření bezpečnosti a realizaci navrženého zabezpečení. Důraz bude kladen na aspekty použitelnosti, bezpečnosti a energetické efektivity návrhu. Praktickým výstupem bude softwarová knihovna využita pro navržené a zrealizované zabezpečení typu E2E u technologie NB-IoT zohledňující bezpečnostní aspekty i technologické omezení.

DOPORUČENÁ LITERATURA:

[1] „NB-IoT Deployment Guide to Basic Feature set Requirements.“ GSMA. Version 2.0, 2018.

[2] „CAT-M & NB-IoT Design and Conformance Test“ Keysight Technologies. 2017.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 27.5.2019

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

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

V současné době se poptávka po mobilních zařízeních a vývojem zařízení v Internetu věcí neustále navyšuje. S touto skutečností také narůstají obavy o bezpečnost a soukromí probíhající komunikace. Pro nově příchozí komunikační technologií NB-IoT (Narrowband Internet of Things), která spadá pod zařízení s nízkým odběrem LPWAN (Low-Powered Wide Area Network), toto není vyjímkou a zákazník se musí při nasazení těchto zařízení spoléhat na zabezpečení sítě operátora, která je efektivní pouze uvnitř jeho sítě. Tato práce se tedy zabývá analýzou komunikace NB-IoT a jejím možném zabezpečení typu end-to-end, které poskytuje také prvky post-kvantových algoritmů. V neposlední řadě se práce zabývá aplikací návrhu na výpočetně omezeném zařízení a zhodnocení její efektivnosti z pohledu časových, energetických a paměťových náročností.

KLÍČOVÁ SLOVA

Úzkopásmová komunikace, Internet věcí, post-kvantová kryptografie, bezpečnost, 5G, NewHope, AES, omezená zařízení

ABSTRACT

Contemporary demand for mobile devices and development of devices in the Internet of Things is constantly increasing. This reality also raises concerns about security and privacy of ongoing communication. This is no exception for expanding scale of Low Powered Wide Area Network (LPWAN) devices which communicate over Narrowband IoT and the customer of such devices who has to rely on security of provider's network to secure customer's data. This security is effective only in operator's network while there is no end-to-end encryption enabled. Therefore, this thesis deals with the analysis of NB-IoT communication and its possible end-to-end security proposal, which also provides elements of post-quantum algorithms. Last but not least, this thesis deals with application of this proposal on constrained device and evaluation of its effectivness of time, energy and memory demands.

KEYWORDS

Narrowband, Internet of Things, post-quantum cryptography, security, 5G, NewHope, AES, constrained devices

KOLAJA, David. *Zabezpečení úzkopásmové licenční bezdrátové komunikace*. Brno, Rok, 60 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Radek Fujdiak, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Zabezpečení úzkopásmové licenční bezdrátové komunikace“ jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské 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

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Radku Fujdiakovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych také chtěl poděkovat členům týmu wislab, konkrétně Ing. Pavlovi Maškovi, Ph.D. a Ing. Martinu Štůskovi, za zapůjčení výpočetně omezeného zařízení, zajištění přístupu k LTE serveru pro vývoj koncové aplikace a jejich podporu při řešení problémů.

Brno ....

podpis autora

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

podpis autora

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

Obsah

Úvod **11**

1 Úvod do LPWAN **12**
1.1 Přiblížení zástupců LPWAN technologií 12
1.1.1 LoRaWAN 12
1.1.2 SigFox 14
1.1.3 Narrowband Internet of Things – NB-IoT 15
1.2 Porovnání LPWAN technologií 16

2 Rozbor vlastností NB-IoT **18**
2.1 Velmi nízka spotřeba 18
2.2 Mód přenosu 18
2.2.1 Pracující režim NB-IoT 19
2.3 Síťové prvky NB-IoT 20
2.4 Aplikace NB-IoT v praxi 20
2.5 Protokoly NB-IoT 21
2.5.1 NAS/AS – Non Access Stratum/Access Stratum 22
2.5.2 RRC – Radio Resource Control 22
2.5.3 PDCP – Packet Data Convergence Protocol 23
2.5.4 RLC – Radio Link Control 24
2.6 Bezpečnost NB-IoT 25
2.6.1 Hrozby Perception Layer 25
2.6.2 Hrozby Přenosové vrstvy 26
2.6.3 Hrozby v Aplikační vrstvě 26
2.7 Shrnutí 27

3 Návrh End-to-End zabezpečení NB-IoT **28**
3.1 Kryptografie na omezených zařízeních 28
3.2 Výběr kryptografických algoritmů 28
3.2.1 NewHope 30
3.2.2 AES – Advanced Encryption Standard 32
3.3 Zprovoznění komunikace NB-IoT 33
3.4 Návrh zabezpečení 35

4 Implementace návrhu zabezpečení **37**
4.1 Používaná zařízení 37
4.2 Příprava prostředí 38
4.2.1 Arduino IDE 38

4.2.2 Linux Ubuntu 39
4.3 Používané knihovny 41
4.3.1 Arduino Cryptography Library 41
4.4 Popis realizace návrhu zabezpečení 44
4.4.1 Implementace Arduino a NB-IoT strany 45
4.4.2 Implementace na straně serveru 47
4.5 Shrnutí realizace návrhu zabezpečení 48

5 Měření implementace zabezpečení **49**
5.1 Časová měření a měření spotřeby implementace 49
5.2 Využití operační paměti Arduina 53
5.3 Shrnutí měření implementace 54

6 Závěr **55**

Literatura **56**

Seznam symbolů, veličin a zkratek **58**

7 Obsah přiloženého CD **60**

Seznam obrázků

2.1 Funkce PSM a eDRX. 18
2.2 Režimy nasazení NB-IoT. 19
2.3 Struktura NB-IoT sítě. 20
2.4 Skupina protokolů NB-IoT. 21
2.5 Typy RRC zpráv. 23
2.6 Proces šifrování a ochrana integrity PDCP. 24
3.1 Série příkazů pro stanovení spojení a odeslání dat na VUT server. . . 34
3.2 Návrh zabezpečení NB-IoT komunikace. 36
4.1 Prostředí Arduino IDE. 39
4.2 Prostředí Linux - Sublime Text. 40
5.1 Zapojení laboratoře měření. 49
5.2 Detail proudového průběhu ustanovení klíče a šifrování. 51
5.3 Detail proudového odběru při nastavení modulu SARA NB-IoT. . . . 51

Seznam tabulek

1.1 Tabulka používaných klíčů v LoRaWAN komunikaci. 13
1.2 Struktura rámce - Uplink. 15
1.3 Struktura rámce - Downlink. 15
1.4 Srovnání technologií LPWAN. 17
3.1 Srovnání post-kvantových algoritmů. 29
5.1 Detailní rozbor časového měření a proudového odběru zařízení. 52

Úvod

S neustále narůstajícím vývojem zařízení v Internetu věcí také souvisí problém udržení úrovně bezpečnosti probíhající komunikace. Toto není vyjímkou ani pro kontinuálně se rozšiřující škálu nízko-odběrových technologií LPWAN (Low-Powered Wide Area Network) zařízení s komunikační technologií NB-IoT (Narrowband Internet of Things) [1]. Taková zařízení většinou disponují podstatně menším výpočetním výkonem v porovnání s běžnými zařízeními připojenými do počítačových nebo mobilních sítí. Díky této skutečnosti není tak jednoduché použití běžných zabezpečovacích algoritmů a protokolů, tudíž jsou taková zařízení tedy lehce napadnutelná a případný únik dat je jedním z příkladů hrozeb pro zařízení s nízkým odběrem energie.

Narrowband IoT je bezdrátová úzkopásmová technologie založená na komunikačních technologiích GSM a LTE [2], vyvinutou partnerským projektem 3GPP. Tento fakt ji dělá zajímavou volbou pro všechny, kteří se chtějí zapojit do světa Internetu věcí. Operátoři mají snadnou implementaci takového řešení do již stávající infrastruktury mobilních sítí právě kvůli tomu, že spolupracuje s technologiemi GSM a LTE. Zájemci o koncová zařízení s NB-IoT moduly jistě ocení jejich výborné charakteristiky jako například nasazení ve velkém měřítku v rámci relativně malého prostoru, dlouhou výdrží baterie, velkého rozsahu pokrytí a propustnostní signálu pevnými překážkami [2]. Jak můžeme vidět v bakalářské práci, tato řešení však nejsou koncipována tak, aby zajistila důvěryhodnost nebo integritu uživatelských dat, která se v síti přenáší. Pokud se zákazník rozhodne pro nasazení NB-IoT modulů přes službu, kterou by poskytoval operátor, tak v tomto případě data chráněna budou, jenže za cenu, že tato data spravuje třetí strana. Pokud na druhou stranu zákazník zvolí implementovat vlastní řešení jen za pomocí nákupu NB-IoT modulů a příslušných senzorů, vysílaná uživatelská data nejsou nijak chráněna.

Cílem práce tedy je se věnovat rozdělení LPWAN zařízení, analýzou komunikace úzkopásmové technologie NB-IoT, následný návrh zabezpečení na aplikační vrstvě a jeho realizace na omezeném zařízení. Cílem bylo také navrhnout zabezpečení, které by bylo efektivní z pohledu využitelnosti v Internetu věcí. Proto byla zvolena varianta implementace na mikrokontroléru, který splňuje tyto podmínky a je dostatečně výkonný pro vykonání daných úloh. Posledním aspektem, na který byl také kladen důraz, byla využitelnost v budoucnosti ohledně bezpečnosti z pohledu post-kvantových systémů. Práce je rozdělena do příslušných kapitol dle rozebíraného tématu, konkrétně s úvodem do LPWAN, kde jsou popsány a porovnány tři nejznámější LPWAN řešení, rozboru vlastností samotného NB-IoT, příslušný návrh End-to-End zabezpečení pro NB-IoT komunikaci a v neposlední řadě realizace návrhu a jeho zhodnocení.

1 Úvod do LPWAN

Současné technologie mobilních sítí, jako GSM, UMTS a LTE, poskytují velmi dobré pokrytí po celém světě. Nicméně většina potenciálně připojitelných zařízení se nachází ve vzdálených místech, daleko od nejbližší mobilní vysílací stanice. Pokud je v této oblasti tedy připojení, není většinou tak silné, což nutí zařízení pracovat na vysokém výkonu a tudíž plýtvá baterií.

Problémem, se kterým se také potýkáme v současných mobilních sítích, je nárůst počtu mobilních zařízení. Předpověď je taková, že v roce 2020 by mohlo být až 31 bilionů připojených zařízení [1], všechny přispívající do takzvané „Internet of Things“ sítě. Většina zařízení mají různá využití, například provoz bezpečnostních systémů, řízení zemědělství, sledování polohy aktiv, chytré odečty nebo chytré domovy [2]. Jsou tu tedy určité nároky a specifika k používání takových zařízení tak, aby nevyužívaly zbytečně veliké množství elektrické energie, nezatěžovaly mobilní síťovou infrastrukturu a měly dostatečné pokrytí. K tomuto účelu se rozšiřuje vývoj a použití LPWAN zařízení.

Z perspektivy přenosových rychlostí mohou být IoT zařízení klasifikovány do dvou kategorií: služby vyžadující rychlý přenos dat (jako například CCTV kamery), a služby, kterým postačuje malá přenosová rychlost (meřící meteorologické přístroje).

Podle [2] definujeme několik charakteristik, které LPWAN technologie sdílí:

- velice malá spotřeba – až 15 let se standardní AA baterií (dle aktivity zařízení),
- mnoho připojených zařízení – až 50 tisíc na jednu základnovou stanici,
- rozsáhlé pokrytí – od 1 km až po 40 km,
- nízká cena zařízení – v jednotkách euro za zařízení,
- nízké přenosové rychlosti – 100 bit/s až 300 kbit/s.

1.1 Přiblížení zástupců LPWAN technologií

1.1.1 LoRaWAN

LoRaWAN (Long Range Wide Area Network) je bezdrátová technologie pro nízkonapěťové aplikace s malým výkonem umožňující přenášet data při rozsáhlém pokrytí, vyvíjená aliancí LoRa. Tyto sítě jsou většinou zapojené do hvězdicové topologie, kde výchozí brána slouží k přenosu dat mezi konečnými zařízeními a centrálním síťovým serverem. Výchozí brána je připojena k síti přes IP linky, zatímco koncová zařízení používají single-hop LoRaWAN komunikaci, která může být přijata jednou či více výchozími branami [3]. Výchozí bráně se také říká koncentrátor.

[1] Dle statistiky uvedené na stránkách společnosti Statista – `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`

Pro přenos se využívá ISM pásem (Industrial, Scientific and Medical), konkrétně 868 MHz v Evropě a 915 MHz ve Spojených Státech Amerických. Pracovní cyklus zařízení je limitován možnostmi vysílání v ISM pásmu. Charakteristiky rychlosti přenosu a velikosti rámce jsou mezi Evropou a Spojenými Státy odlišné. Zatímco v Evropě se v pásmu 868 MHz mohou data vysílat minimální rychlostí 250 bit/s o velikosti rámce minimálně 59 oktetů, ve Spojených Státech je to 980 bit/s a 19 oktetů [3]. Tyto nastavení se nevztahují na přenos rámců pro registraci zařízení do sítě. Koncová zařízení jsou povinna použít takové parametry, které dokáží poslat 23 bytovou Join-request zprávu. Používá se konceptu portů (8 bitová, volitelná hodnota) pro ovládání různých aplikací na koncovém zařízení. Pokud je port nastaven na 0, je rezervován pro specifické LoRaWAN zprávy pro konfiguraci zařízení a změnu parametrů. K tomuto se používají MAC (Media Access Control) příkazy.

K tomu, aby mohlo zařízení pracovat v LoRaWAN síti, potřebuje 32 bitovou adresu zařízení, která je přiřazována při připojení zařízení do sítě nebo je tato adresa již dodávána se zařízením. V případě pohybujících se zařízení je takové zařízení opatřeno adresou založenou na 24 bitovém síťovém identifikátoru (*NetID*) [3], které je přiřazeno do sítě aliancí LoRa.

Koncová zařízení v síti pracují s jednou, či několika málo aplikacemi, které souhlasí s LoRaWAN použitími. Aplikace se identifikují pomocí 64 bitové *AppEUI* hodnoty, která by měla být zaregistrovaná jako IEEE EUI64 hodnota. Navíc musí mít zařízení uložené další dva symetrické relační klíče, jeden pro ochranu proti síťovým artefaktům (*NwkSKey*) a druhý pro ochranu provozu na aplikační vrstvě (*AppSKey*) [3]. Tyto klíče jsou použity pro kryptografické operace na algoritmu AES-128, viz Tab. 1.1.

Tab. 1.1: Tabulka používaných klíčů v LoRaWAN komunikaci.

Hodnota	**Popis**
DevAddr	32 bitová adresa zařízení generovaná z NetID
AppEUI	IEEE EUI64 hodnota identifikující aplikaci
NwkSKey	128 bitový klíč relace používaný s AES-CMAC
AppSKey	128 bitový klíč relace pro aplikační vrstvu používaný s AES-CTR
AppKey	128 bitový klíč relace používaný s AES-ECB

Alternativně může koncové zařízení využít připojovací proceduru přímo přes servery LoRaWAN k nastavení hodnot výše zmíněných, a tak dynamicky získat přístup k síti. K tomuto je potřeba, aby zařízení znalo *AppEUI* klíč, a navíc i odlišný symetrický klíč, který je vázaný na *AppEUI – AppKey*. Každé koncové zařízení má jedinečný *AppKey*. Posledním identifikátorem, které každé zařízení potřebuje, je *DevEUI*, který určuje název zařízení a je globálně jedinečný.

Veškerý datový obsah je šifrovaný a je zajištěna integrita dat. MAC příkazy, které jsou posílány jako datový obsah jsou taktéž chráněny. MAC příkazy jdou ale poslat i jako možnosti zprávy, proto pokud jsou tyto příkazy posílány tímto způsobem, tak nejsou součástí datové části, a mohou být viděny jakýmkoliv příjemcem, jelikož nejsou šifrované. Pro LoRaWAN verzi 1.0.x je *NwkSKey* používán k zajištění integrity dat mezi konečným zařízením a síťovým serverem [3]. *AppSKey* je použit pro důvěrnost dat mezi koncovým zařízením a serverem, nebo aplikací za tímto serverem. Všechny zprávy vrstvy MAC mají vnější 32 bitovou Message Integrity Code vypočítanou ze šifrované datové části, jiných hlaviček a NwkSKey s použitím AES-128 CMAC.

1.1.2 SigFox

SigFox funguje na základě modelu „jedna smlouva, jedna síť“, což poskytuje zařízením se připojit v jakékoliv zemi bez ohledu na roaming nebo předávky spojení. Architektura sítě spočívá v jediné cloudové síti, která umožňuje globální přístup s minimálním dopadem na koncová zařízení a rádiovou přístupovou síť.

Hlavními prvky této sítě jsou Servisní Středisko a Registrační Autorita. Servisní středisko má za úkol starat se o připojení základnových stanic k Internetu, dále o správu a ovládání základnových stanic (neboli také výchozí brána, jak ji známe z LoRaWAN) a zařízení připojených přímo k základnové stanici. Registrační Autorita má za úkol autentizaci zařízení do přístupové sítě. Koncová zařízení mohou být statická nebo v pohybu, jelikož se asociují k Servisnímu Středisku, a ne k určité základnové stanici.

Rádiové rozhraní je založeno na komunikaci přes Ultra Narrowband, stejně jako LoRaWAN na ISM pásmu. V Evropě tedy využívá frekvenci 868 MHz a ve Spojených Státech 902 MHz [4]. Tento signál dokáže procházet skrz pevné objekty. Spojení je pro zařízení umožněno v obou směrech čili jak pro downlink, tak i uplink. Nicméně systém je optimalizován spíše pro komunikaci přes uplink (od zařízení k Servisnímu Středisku). Kvůli optimalizaci spektra jsou potřeba různé uplink, downlink rámce, ale také metody synchronizace časování.

Fyzická vrstva uplinku má dle [4] následující charakteristiky:

- šířka pásma 100 Hz / 600 Hz (záleží na státě),
- rychlost modulace 100 Bd / 600 Bd (záleží na státě),
- modulační schéma – DBPSK,
- link budget – 155 dBm.

Formát rámce linkové vrstvy je definován v Tab. 1.2. Jak už bylo napsáno výše, je kvůli optimalizaci spektra definovány jiné struktury pro uplink a downlink. Charakteristiky fyzické vrstvy pro downlink se liší hlavně v šířce pásma, která se zvýšila

Tab. 1.2: Struktura rámce - Uplink.

Preambule	Hlavička rámce	ID zařízení	Data	Autentizační zpráva	FCS
19 bitů	29 bitů	32 bitů	0–96 bitů	16–40 bitů	16 bitů

na 1500 Hz, zatímco rychlost modulace zůstává stejná, na 600 Bd. Změna je i v modulačním schématu, které je na downlinku zajištěno pomocí GFSK [4].

Formát rámce linkové vrstvy (Tab. 1.3) na downlinku je podobný formátu rámce na uplinku, akorát délky jednotlivých polí se liší. Rádiové rozhraní je optimalizováno

Tab. 1.3: Struktura rámce - Downlink.

Preambule	Hlavička rámce	Kontrola a oprava chyb	Data	Autentizační zpráva	FCS
91 bitů	13 bitů	32 bitů	0–64 bitů	16 bitů	8 bitů

pro uplink komunikaci, která je asynchronní. Downlink komunikace probíhá tak, že zařízení po odeslání uplink přenosu otevře fixní okno pro přijetí vyžádaných dat. Zpoždění a trvání tohoto okna mají fixní hodnotu. Uplink a downlink na těchto zařízeních jsou tedy nevyvážené, kvůli regulačnímu omezení ISM pásem.

Co se týče zabezpečení komunikace v SigFox síti, zde se předpokládá, že kvůli malé složitosti zařízení, budou zařízení obhospodařovat jednu, maximálně několik málo aplikací, které komunikují jednou za delší čas s ostatními síťovými prvky. Rádiový protokol tedy umožňuje autentizaci a zajišťuje také integritu přenášených zpráv, dosaženou pomocí unikátního ID zařízení a autentizačního kódu zprávy. Zabezpečovací klíče jsou nezávislé pro každé zařízení. Tyto klíče jsou přiřazovány k ID zařízení a jsou předem poskytované. Data aplikací mohou být tedy šifrována dle kritickosti případu použití, což umožňuje vyvážení nákladů a také na výběr mezi zatížením zařízení vs. riziko.

1.1.3 Narrowband Internet of Things – NB-IoT

Narrowband IoT bylo vyvinuto a standardizováno v 3GPP vydání 13 v červnu roku 2016 [2]. Ostatní vylepšení a charakteristiky NB-IoT jsou specifikovány i ve 14. vydání v roce 2017, například podpora multicast vysílaní. Na rozdíl od SigFox

[2]Více informací na `http://www.3gpp.org/news-events/3gpp-news/1785-nb_iot_complete`

a LoRaWAN je tedy NB-IoT vyvíjeno prostředky společnosti 3GPP a operuje na licencovaných pásmech již nasazeného LTE, zatímco ostatní technologie využívají nelicencované (ISM). Konkrétně se v Evropě využívají frekvenční pásma 800 MHz, 900 MHz a 1800 MHz. Celá komunikace NB-IoT je postavena na sadě protokolů LTE [2], jen upravených tak, aby splňovala požadavky zařízení s omezeným výkonem, paměťovými nároky a nízkou spotřebou, které jsou charakteristické pro LPWAN zařízení.

NB-IoT operuje ve třech různých módech nasazení, jmenovitě Stand-alone mód, který využívá nosnou GSM pásma kolem 900 MHz [5]. In-Band mód znamená, že je tato úzkopármová technologie nasazená společně s LTE pásmem, kde je flexibilně sdíleno mezi NB-IoT a klasickou LTE nosnou. Posledním je Guard-Band, kdy NB-IoT pracuje v nevyužitých blocích prostředků mezi dvěma LTE nosnými. NB-IoT podporuje poloviční duplex s frekvenčním dělením (FDD) s maximálně 60 kbit/s na uplinku, 30 kbit/s na downlinku a maximální velikostí jednotky 1600 bajtů, což je limitováno PDCP vrstvou (popsáno v kapitole 2.5.3).

Cílovým parametrem NB-IoT je také maximální ztráta na spojení, která by měla být 164 dB [5]. S takovýmto parametrem se výkon downlinku pohybuje někde mezi 200 bit/s až 2-3 kbit/s, záležící na módu nasazení. Za účelem prodloužení životnosti baterie jsou využity technologie úsporného režimu (PSM) a rozšířeného nespojitého vysílání (eDRX). NB-IoT infrastruktura je rozdělena do dvou hlavních oblastí – řídící vrstva a uživatelská vrstva. Řídící vrstva obsahuje protokoly, které se starají o funkce přístupu rádiových nosičů a spojení mezi zařízeními klienta a sítí. Nejvyšší vrstva řídící vrstvy je nazývána Non-Access Stratum a je blíže popsána v kapitole 2.5.1.

Hlavní výhodou NB-IoT oproti zmiňovaným technologiím je větší pokrytí a velmi dobrá schopnost penetrace objektů, právě kvůli nízké frekvenci, na které NB-IoT pracuje. Slibuje se také velmi malá pořizovací cena NB-IoT modulů, která by neměla v roce 2020 překračovat US$5 za modul, se životností baterie až 10 let.

1.2 Porovnání LPWAN technologií

Tab. 1.4 poskytuje srovnání technologií LPWAN popsaných v předešlých kapitolách [2]. Každá technologie má také různou terminologii pro prvky používané v jejich sítích. Například pro používané senzory v každé technologii jsou v LoRaWAN, NB-IoT a SigFox používané termíny, resp., End Device, User Equipment a Leaf Node [5]. Co se týče technických rozdílů, tak se technologie hlavně liší kvalitou služeb (kdy je na tom NB-IoT nejlépe kvůli tomu, že je udržováno LTE vývojem), zpožděním dat, velikostí přenášené jednotky, dosahu (kdy SigFox je na tom nejlépe, na pokrytí města by stačila jedna základnová stanice, tedy na celé území Belgie o rozměru 30

500 km^2 stačí sedm základnových stanic [2]). Zato technologie NB-IoT má výhodu oproti ostatním technologiím jelikož umožňuje nasazení zařízení v místech, kde není dosah mobilních sítí tak silný, kvůli frekvenci na které pracuje. Příkladem mohou být domovní sklepy nebo hluboko pod vodní hladinou.

Tab. 1.4: Srovnání technologií LPWAN.

	NB-IoT	**SigFox**	**LoRaWAN**
Frekvence (pro Evropu)	LTE (800/900 MHz)	ISM (868 MHz)	ISM (868 MHz)
Šířka pásma	180 kHz	100 Hz	125 kHz
Max. rychlost přenosu	250 kb/s	100 b/s	50 kb/s
Dosah signálu	<10 km	<40 km	<20 km
Bezpečnost	LTE PDCP	Nepodporováno	Ano (AES128)
Max. velikost PDU	1600 bajtů	12 bajtů	243 bajtů

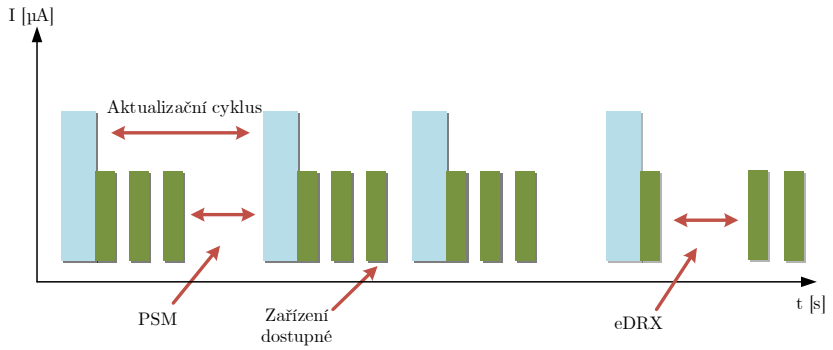
Vzhledem k převažujícím výhodám technologie NB-IoT oproti ostatním LPWAN technologiím, zejména kvůli jednoduchému nasazení a vysoké využitelnosti, se v následujících kapitolách na tuto technologii více zaměříme. Popíšeme jednotlivé funkce a rozebereme protokoly, které využívá ke komunikaci s okolními zařízeními.

2 Rozbor vlastností NB-IoT

Z porovnání technologií uvedených dříve jsme zjistili, že komunikace přes NB-IoT bude pravděpodobně mít největší osazení a užitečnost v zemích, kde již je implementována mobilní síť LTE, jelikož je to pro operátory mobilních sítí jen otázkou aktualizace software, aby se NB-IoT dalo jednoduše nasadit. V několika dalších kapitolách si tedy přiblížíme NB-IoT charakteristiky.

2.1 Velmi nízka spotřeba

Pomocí úsporného režimu (PSM) a rozšířenému nespojitému vysílání (eDRX) může být uskutečněna dlouhá pohotovostní doba (Obr. 2.1). PSM technologie byla nově přidána do 3GPP 12. vydání, kde v tomto úsporném režimu je zařízení stále registrováno, ale nemůže přijímat žádný signál kvůli jeho „hlubokému spánku“ pro co největší úsporu baterie [1]. Na druhou stranu eDRX prodlužuje dobu úsporného cyklu ještě dále, čímž se zařízení vyhne zbytečnému probouzení z hlubokého spánku.



Obr. 2.1: Funkce PSM a eDRX.

2.2 Mód přenosu

Celý vývoj NB-IoT je založený na základním LTE přenosu, který je upravený podle charakteristik a jedinečných vlastnostech NB-IoT. Šířka pásma na fyzické vrstvě je 200 kHz. V Evropě využívá frekvenčních pásem 800/900/1800 MHz pásma LTE. Na downlinku (komunikace směrem ke koncovému zařízení) je použito modulace QPSK a technologie OFDM se subnosnými, kde každá má šířku 15 kHz. Na uplinku (komunikace směrem od koncového zařízení) je podobně jako na downlinku použita modulace QPSK nebo BPSK a technologie SC-FDMA s aplikováním jediné, či několika subnosných, kde v případě jediné subnosné je šířka 3,75 kHz a u několika je šířka subnosné 15 kHz, jako u downlinku [1].

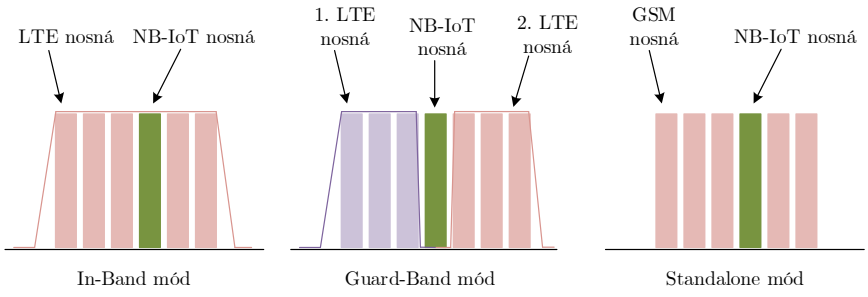
Pro šířku subnosné 15 kHz je definováno 12 kontinuálních subnosných, což dává dohromady 180 kHz na šířce pásma. Obdobně je pro 3,75 kHz subnosných definováno 48 kontinuálních subnosných. Schopnost vyššího pokrytí pro šířku subnosné 3,75 kHz je větší, než u 15 kHz, kvůli vyšší výkonové spektrální hustotě.

Na vyšších vrstvách je protokol NB-IoT formulován skrz modifikace funkcí LTE, jako například mnohočetné připojení, nízká spotřeba a přenosu několika málo dat. Hlavní část NB-IoT sítě je připojena přes S1 rozhraní. Maximální definované délky transportních bloků jsou pro komunikaci na uplink (NPUSCH – Narrowband Physical Uplink Shared Channel) 1000 bitů a pro downlink (NPDSCH – Narrowband Physical Downlink Shared Channel) 680 bitů [1]. Minimální délka pro obě spojení je 16 bitů.

2.2.1 Pracující režim NB-IoT

V současné době podporuje NB-IoT jen polo-duplexní FDD se šířkou pásma 180 kHz, umožňující 3 typy nasazení (Obr. 2.2):

- Stand-alone mód – Zužitkovává nezávislé frekvenční pásmo mimo LTE pásma,
- Ochranné pásmo – Zužitkovává ochranné pásmo LTE, čemuž se rozumí čas, kdy je LTE pásmo v klidu,
- Přímo v pásmu LTE – Pracuje kolektivně v pásmu LTE a zabírá jeden fyzický zdrojový blok frekvence pásma LTE, aby mohlo pracovat.



Obr. 2.2: Režimy nasazení NB-IoT.

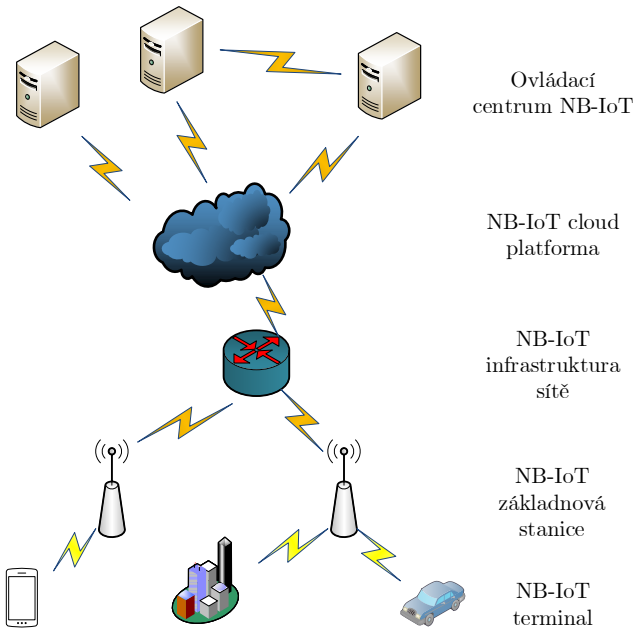
Přenosové rychlosti jsou na downlinku a uplinku různé. Z koncového zařízení se do infrastruktury sítě může přenášet rychlostí maximálně 67 kbit/s, na druhou stranu pokud jsou nějaká data posílána na zařízení, může je přijímat s rychlostí maximálně poloviční, jako na uplinku – 30 kbit/s [1]. Maximální velikost přenášené jednotky je stanovena na 1600 bajtů.

Při nasazení v nezávislém režimu, NB-IoT má až 164 dB sílu pokrytí. Pokud je přenos spolehlivý, potom se této síly pokrytí dosáhne jedině s větším zpožděním. V současnosti je největší tolerované zpoždění při přenosu dat 10 sekund.

2.3 Síťové prvky NB-IoT

Obr. 2.3 nám popisuje obecnou architekturu sítě NB-IoT:

- NB-IoT terminal (koncové zařízení) – Všechna zařízení mají přístup k síti, dokud mají nainstalovanou odpovídající SIM kartu
- NB-IoT Base Station (rádiový vysílač/přijímač) – Anténa, která byla již předem nainstalovaná operátory mobilních sítí, a podporuje všechny tři módy nasazení zmíněné výše
- Infrastruktura sítě NB-IoT – S pomocí infrastruktury sítě se může rádiový vysílač/přijímač připojit ke cloudové službě NB-IoT
- Cloudová platforma NB-IoT – Zpracovává mnoho služeb NB-IoT zařízení, výsledky jsou přeposílány dále do ovládacího centra NB-IoT
- Ovládací centrum NB-IoT – Může ukládat data služeb NB-IoT, a také ovládat koncová zařízení



Obr. 2.3: Struktura NB-IoT sítě.

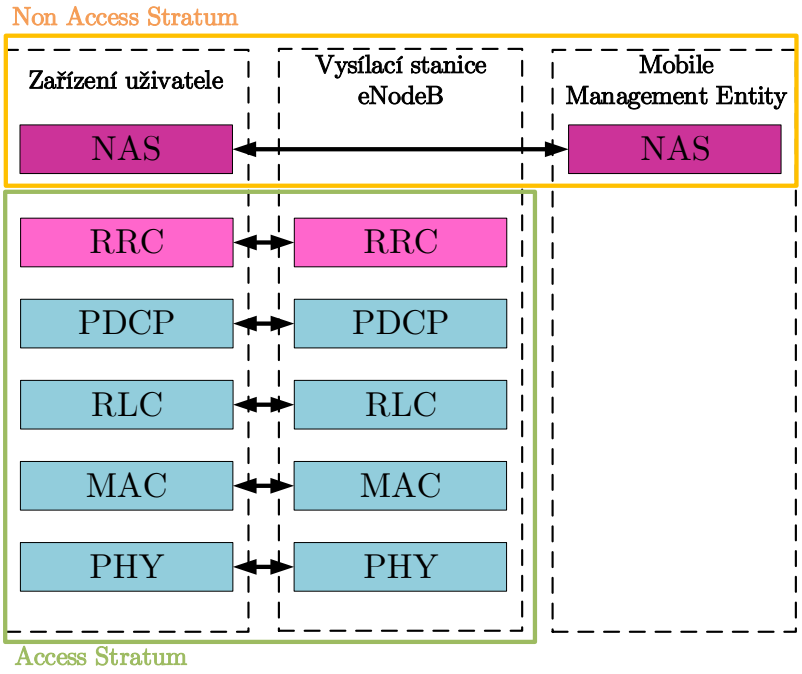
2.4 Aplikace NB-IoT v praxi

Kvůli charakteristikám NB-IoT tato technologie splňuje požadavky aplikací pro přenos malých objemů dat, služeb s malými přenosovými rychlostmi a nízkým proudovým odběrem nebo dlouhou pohotovostní dobou.

Příkladem užití NB-IoT mohou být autonomní senzory zabezpečovacích systémů, jako kouřové detektory, které nemají skoro žádné nároky na velikost přenášených dat (v řádech několika bajtů), a jejich přenosový cyklus je většinou jednou za měsíc nebo rok. Mohou tím být i chytré odečty z měřících zařízení na elektřinu, vodu nebo plyn. Tímto se eliminuje nutnosti použití lidské činnosti, která je nadbytečná. Dalším užitím mohou být i chytrá parkovací místa, kde senzory kontrolují, zda je místo volné. Díky takovým informacím si také můžete udělat rezervaci pro příslušné parkovací místo apod. Senzory NB-IoT mohou být používány také na sledování aktiv, jako jsou nákladní vozy, jedoucí veliké vzdálenosti, kde by nevadilo, že jezdí i do rozsáhlých míst, kde nemusí být tak dobrý signál mobilní sítě. S více se rozšiřujícími půjčovny kol online, kdy jen přijdete k volnému kolu ve městě a půjčíte si jej, tak zde by bylo nasazení NB-IoT senzorů také vhodné.

2.5 Protokoly NB-IoT

Protokolová sada NB-IoT (Obr. 2.4) z větší části přejímá protokolovou sadu LTE, jen trochu zjednodušeně, aby splňovaly podmínky vlastností zařízení NB-IoT. Směrem od nejnižší po nejvyšší vrstvu jsou poté protokoly následující: PHY – Fyzická vrstva, MAC – Media Access Control, RLC – Radio Link Control, PDCP – Packet Data Convergence Protocol, RRC – Radio Resource Control a v nejvyšší vrstvě je poté NAS – Non Access Stratum.



Obr. 2.4: Skupina protokolů NB-IoT.

2.5.1 NAS/AS – Non Access Stratum/Access Stratum

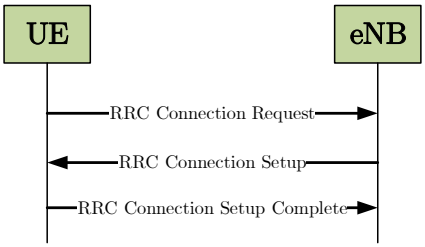
NAS – Non Access Stratum – je nejvyšší vrstva skupiny protokolů NB-IoT a je používaný pro navázání IP spojení mezi koncovým zařízením a infrastrukturou sítě. NAS je tedy signalizační vrstva komunikace mezi koncovým zařízením (UE – User Equipment) a MME (Mobile Management Entity – server zpracovávající požadavky koncového zařízení ohledně komunikace v síti), zajišťující autentizaci zařízení do sítě, ustanovení klíčů používaných v komunikaci a výměnu parametrů potřebných pro komunikaci se sítí Internet.

AS – Access Stratum – popisuje komunikaci mezi koncovým zařízením a eNodeB (vysílací/přijímací rádiová stanice, ke které jsou koncová zařízení připojena) stanicí. Je to sada protokolů pracující jak na uživatelské vrstvě, tak i na řídící vrstvě komunikace NB-IoT, resp. LTE celkově. Těmito protokoly jsou RRC, RLC a PDCP, které budou popsány v dalších kapitolách.

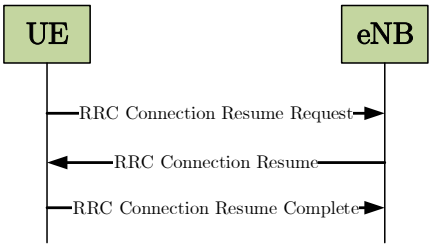
2.5.2 RRC – Radio Resource Control

Předtím, než začne zařízení vysílat jakákoliv uživatelská data, musí zařízení pracovat v módu RRC – Connected, jak lze vidět na Obr. 2.5a. Navázání spojení s RRC je provedeno pomocí 3-way handshake mezi koncovým zařízením a eNodeB, což zajistí, že se zařízení přepne ze stavu RRC-Idle do RRC-Connected módu. Tuto proceduru zahajuje koncové zařízení, které se chce připojit do sítě a skládá se ze tří kroků. Koncové zařízení pošle RRC požadavek na eNodeB stanici, ustanovení připojení mezi koncovým zařízení a eNodeB, a potvrzení spojení mezi oběma prvky. Tyto zprávy jsou velice důležité pro celé spojení ustanovené mezi koncovými zařízeními. Kvůli účinnosti a šetření výpočetních zdrojů zařízení nejsou tyto zprávy nijak chráněny, zajištění šifrování pomocí šifrovacího klíče a klíče integrity se používá až ve vyšších vrstvách přenosu NB-IoT [6]. Ustanovení klíčů zahrnující i klíče pro protokol RRC zajišťuje NAS pomocí tzv. LTE Attach procedury, která běží současně s RRC protokolem, a proto jsou některé zprávy nechráněny. Všechny módy a procedury lze vidět v Obr. 2.5, kde UE je koncové zařízení a eNB je základnová stanice.

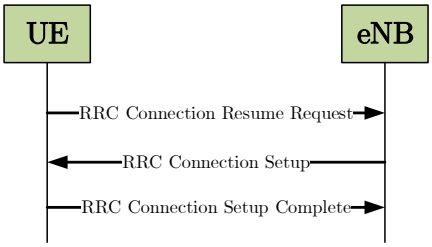
Různé RRC zprávy také stanovují různé signalizační rádiové nosiče – Signaling Radio Bearer (SRB). SRB popisuje způsob, jakým jsou data přenášeny z UE do Internetu. Jsou 3 typy SRB, lišící se kvalitou služeb, typu přenášených informací a prioritou zpracování – SRB0, SRB1 a SRB2. Pod nosičem si můžeme také představit pipeline, která slouží jako virtuální tunel pro data komunikace.



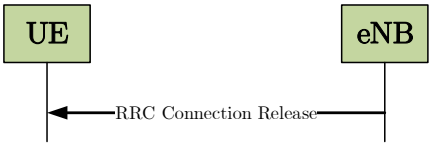
(a) RRC navázání spojení.



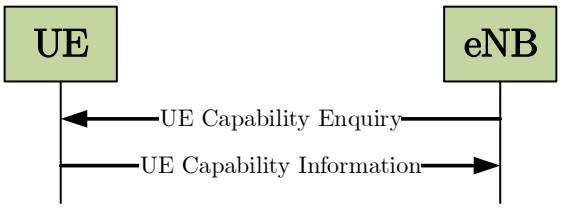
(b) RRC pokračování přenosu.



(c) RRC zamítnutí pokračování přenosu.



(d) RRC ukončení přenosu.



(e) RRC stanovení parametrů schopnosti zařízení.

Obr. 2.5: Typy RRC zpráv.

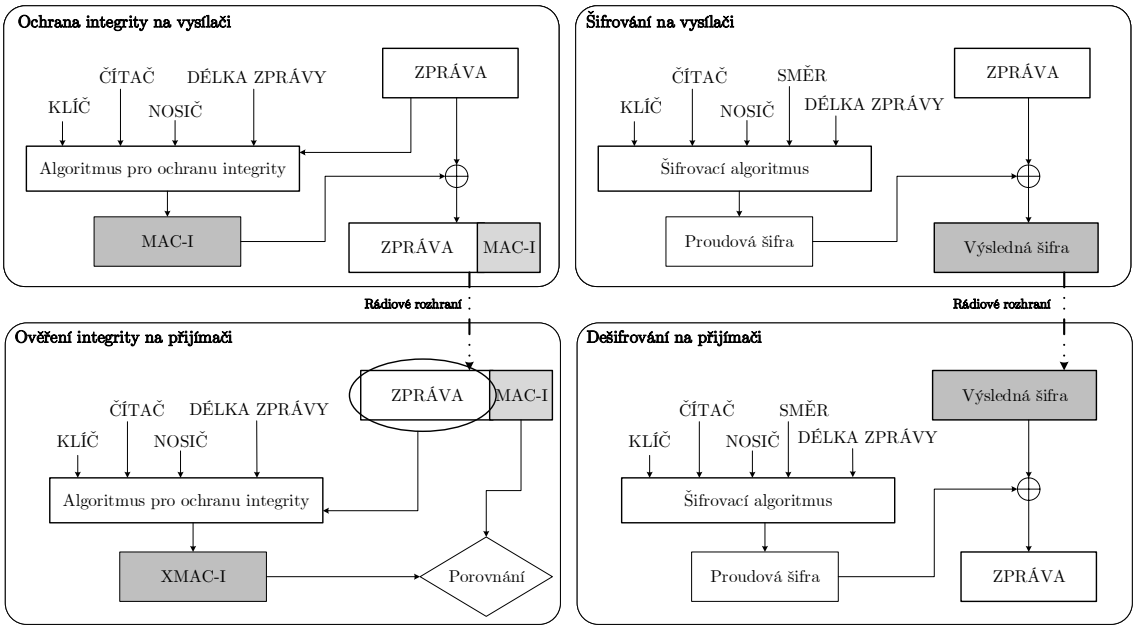
2.5.3 PDCP – Packet Data Convergence Protocol

Z hlediska základních operací je to, co dělá PDCP, velmi jednoduchý protokol – jen přidává PDCP hlavičku k příchozím datům a předá je RLC na downlinku, nebo právě naopak odstraňuje PDCP hlavičku od příchozího paketu a předává ji dále vrstvě IP v případu uplinku [7]. Základní operace, co tedy PDCP provádí jsou:

- komprese/dekomprese hlavičky pomocí RoHC (Robust Header Compression) – jen na úrovni přenosu uživatelských dat,
- šifrování a dešifrování uživatelských/řídících dat,
- přenesení dat a údržba sekvenčního čísla PDCP,
- ochrana integrity a ověření dat řídící roviny – jen na úrovni přenosu řídících dat.

Ochrana integrity (Obr. 2.6a) je prováděna pomocí MAC-I (Message Code Authentication Integrity), která má délku 4 bajty a je generována na vysílači pomocí algoritmu na ochranu integrity, do které vstupuje pět parametrů.

Šifrování dat (Obr. 2.6b) je zajištěno pro každou protokolovou datovou jednotku vysílající ze zařízení. Originální zprávy jsou maskovány (pomocí XOR operace) šifro-



(a) Algoritmus ochrany integrity. (b) Šifrovací algoritmus PDCP.

Obr. 2.6: Proces šifrování a ochrana integrity PDCP.

vacím klíčem, který je výstupem šifrovacího algoritmu EEA (Evolved Packet Systém Encryption Algorithm) [8], do kterého vstupuje, stejně jako u ochrany integrity, pět parametrů:

- Klíč – 128 bitový klíč uložený na SIM
- Čítač – 32 bitové sekvenční číslo PDCP jednotky
- Identifikátor nosiče – 5 bitové číslo
- Směr – 1 bitové číslo (downlink/uplink)
- Délka vstupní zprávy – 32 bitové číslo

Samotný šifrovací algoritmus, do kterého vstupují tyto parametry se už liší od verze EAA algoritmu. Ve verzi EAA1 byla použita šifra SNOW 3G, ve verzi EAA2 AES-128 s operačním módem CTR, a v nejnovější verzi EAA3 je použita šifra ZUC.

2.5.4 RLC – Radio Link Control

Radio Link Control je protokol druhé vrstvy a jeho prací je zajistit korektnost vysílaných dat a kvalitu rádiového kanálu. Stanovuje parametry komunikace přes rádiové rozhraní jako např. nastavení velikosti okna přenosu, časovač dotazování nebo maximální počet opakovaného poslání protokolové datové jednotky [7]. Protokol RLC je velice podobný protokolu SR-ARQ (Selective Repeat Automatic Request). RLC má na starosti tedy několik funkcí:

- Přenos protokolových datových jednotek vyšších vrstev
- Korekce chyb přes ARQ (Automatic Repeat Request)

- Segmentace, zřetězení a opětovné sestavení RLC SDU (Služební Datová Jednotka)
- Potvrzované, Nepotvrzované a Transparentní operační módy

Různé módy RLC protokolu zajišťují různé operační módy přenosu datových jednotek. Transparentní mód RLC – Neposkytuje žádnou segmentaci a sestavování SDU, RLC hlavička není přidávána a doručení datové jednotky není garantováno. Nepotvrzovaný mód RLC – Poskytuje segmentaci a opětovné sestavení SDU, RLC hlavičky jsou přidávány, není garantováno doručení datové jednotky. Potvrzovaný mód RLC – Stejné, jako nepotvrzovaný mód, jen navíc garantuje zaručení datové jednotky, což je vhodné pro TCP spojení.

2.6 Bezpečnost NB-IoT

Bezpečnostní požadavky NB-IoT jsou podobné, jako bezpečnost tradičních IoT zařízení. Nicméně jsou zde určité odlišnosti, které se vztahují k aspektům zařízení s nízkým odběrem, módu síťové komunikace a skutečnými požadavky služeb. Například, zařízení v obyčejném IoT mají obecně větší výpočetní výkon, komplikované síťové protokoly pro přenos a zavádí striktnější bezpečnost, ale na druhou stranu mají vysokou spotřebu energie. To pro NB-IoT zařízení znamená, že není tolik prostředků pro implementaci bezpečnostních opatření, což představuje hrozbu, kde i ta nejmenší chyba může mít mnohem větší následky, protože zabudovaný systém koncových zařízení je jednodušší a lehčí. Jelikož komunikace NB-IoT prvků sdílí výpočetní zdroje a komunikační kanály klasického LTE, valná většina útoků, které se dají provést v LTE komunikaci, by se dala aplikovat i na komunikaci NB-IoT.

IoT architektura se skládá ze tří různých vrstev a na každé definujeme různá bezpečnostní rizika. Tyto vrstvy se nazývají Vnímací vrstva (Perception Layer) – nejnižší vrstva, do které spadají fyzická zařízení; Přenosová vrstva (Transmission Layer) – nebo také síťová vrstva, kam spadají všechna síťová zařízení starající se o přenos dat k cíli; a vrstva Aplikační (Application Layer), kde můžeme najít například servery služeb nebo přístup k uživatelům.

2.6.1 Hrozby Perception Layer

Podobně jako perception layer obyčjného IoT má tato vrstva NB-IoT tendenci být jak pod pasivním, tak i pod aktivním útokem. Pasivní znamená, že útočník jen ukradne informaci, ale nijak ji neupravuje, například jen analyzuje síť nebo provádí eavesdropping. Jelikož se přenos dat v NB-IoT spoléhá na otevřenou bezdrátovou síť, útočník může jednoduše získat informace přenášené pomocí naslouchání datového spojení za účelem analýzy funkcí sítě, a pozdějšího provedení série útoků. Na rozdíl

od pasivních útočníků se aktivní útoky využívají hlavně k falsifikaci informace a poškození integrity, tudíž mají aktivní útoky mnohem větší dopad a představují větší hrozbu NB-IoT koncovým zařízením než pasivní. Metody aktivních útoků zastupují například útok klonováním uzlů, útok získáním uzlů, upravování zpráv vysílaných z koncových zařízení atd. Typickým aktivním útokem by mohlo být na chytré odečty z měřících zařízení, kdy útočník upraví svévolně obsah odeslané zprávy, které přímo ovlivňují životně důležité zájmy uživatele. Proti těmto útokům by se tedy dalo chránit užitím kryptografických mechanismů, jako šifrování dat, autentizace a ověřování integrity zpráv.

Co se týče autentizace koncových zařízení k základnové stanici, tato bezpečnost by měla být obousměrná. To znamená, že přístup autentizací koncových zařízení k základnové stanici by mělo být bráno v potaz, ale i v opačném směru, aby se nemohly vytvářet pseudo-základnové stanice, které by se tvářily jako ty správné [9], ke kterým se koncová zařízení mohou připojit.

2.6.2 Hrozby Přenosové vrstvy

Mnoho problémů tradičních IoT systémů (více sítí v jedné síti, vysoká cena a vysoké kapacity baterií) řeší NB-IoT. S tím ale jsou přineseny další bezpečnostní rizika. Jedním z útoků představuje přístup k vysokokapacitním NB-IoT terminálům – Jeden sektor NB-IoT dokáže pojmout až 100 000 koncových zařízení [1]. Hlavní výzvou je tedy přinést systém, kterému bude umožněna efektivní ověřování identity a kontrola přístupu v reálném čase pro vyhnutí se útokům jako podstrčení falešných informací škodlivým uzlem. Dalším příkladem by mohlo být znepřístupnění komunikace mezi koncovými zařízeními a základnovými stanicemi pomocí rušiček signálu nebo by se také mohlo způsobit masivní odepření služeb (DDoS) pomocí uzlů, které by ovládal.

Řešením těchto problémů je představení efektivního end-to-end mechanismu pro autentizaci a správy klíčů k poskytnutí důvěrnosti a zajištění integrity přenášených dat. Takovéto protokoly již existují, jak v LTE, tak v počítačových sítích (IPSec, SSL, a mnoho dalších). Jenže je problém tyto protokoly implementovat v NB-IoT zařízeních tak, aby dodržovaly stejné charakteristiky, které byly představeny na začátku. Bylo by také dobré implementovat ochranu na způsobu behaviorálních dat každého uzlu v síti. Pokud by tedy zařízení zvýšilo z ničeho nic svoji aktivitu porovnáním oproti jeho normálnímu chování, tak by se vyvolal bezpečnostní incident.

2.6.3 Hrozby v Aplikační vrstvě

Cílem aplikační vrstvy v NB-IoT sítích je ukládat, analyzovat a spravovat efektivně data. Po vnímací vrstvě a přenosové se tedy nahromadí spousty dat v aplikační vrstvě, které musí být určitým způsobem zpracovávány pro další užití. Největším

problémem je tedy identifikace, klasifikace a následné zpracovávání takovýchto dat. Jelikož mohou být příchozí data heterogenního typu, komplexita zpracování těchto dat se zvyšuje. Otázkou tedy je, zda je vhodné tyto data nějak zálohovat nebo také zavést určitou toleranci chyb.

Hlavními požadavky na bezpečnost podle [1] jsou následující: identifikace a zpracování masivních heterogenních dat; integrita a autentizace těchto dat; kontrola přístupu dat – v NB-IoT je velké množství uživatelských skupin, přístupů a operačních akcí. Korespondující správy pro odlišné úrovně uživatelů by měly být zavedeny.

2.7 Shrnutí

Jak již bylo popsáno v předešlých kapitolách, NB-IoT je jedinečná technologie přenosu, zjevně se lišící od ostatních například velmi skvělou penetrací objektů, což nám umožňuje instalovat senzory do míst, kde není pokrytí mobilními sítěmi tak silné, jako sklepy, podzemní garážové komplexy, ale také vzdálené místa od nejbližší eNodeB stanice.

Je zde ale nutnost chránění obsahu, což již dostatečně poskytuje LTE technologie, na které je NB-IoT založené. Jenže jsou zde určité spekulace o soukromí přenášeného obsahu přes cizí kanál, ještě k tomu, když správa šifrovacích klíčů není pod naší ochranou, nýbrž pod ochranou třetí strany, udržující infrastrukturu mobilních sítí. V případech, kdy využíváme senzory NB-IoT pro chytré odečty energií nebo sledování aktiv, bychom mohli mít podezření, že třetí strana si kdykoliv může posílaná data přečíst. Již dnes má Vodafone pokryté celé území České Republiky a poskytuje tedy služby NB-IoT. Pokud by tedy například Vodafone chtěl, neměl by být problém si z MME (Mobile Management Entity) databáze vzít příslušné klíče a zpracovávat data o uživatelích z důvodu profilování a následné cílené reklamy. Jistě jsou tyto popsané praktiky extrémy, jenže jsou tu jisté hrozby, které jsou uskutečnitelné. Řeč je hlavně o hrozbách Man-In-The-Middle, které jsou založeny, jak již bylo zmíněno v kapitole 2.6.1, například na falešné základnové stanici (FBS – False Base Station).

Proto se tedy budeme snažit v následujících kapitolách zajistit další vrstvu zabezpečení na aplikační vrstvě, kde budeme uživatelská data generovaná senzorem šifrovat pomocí šifrovacího algoritmu AES-256 a pro ustanovení šifrovacího klíče využijeme jeden z dostupných post-kvantových algoritmů pro ustanovení klíče.

3 Návrh End-to-End zabezpečení NB-IoT

3.1 Kryptografie na omezených zařízeních

Omezená zařízení jsou taková, která mají výpočetně slabé možnosti. Skládají se většinou z jednoho čipu na integrovaném obvodu s integrovanou pamětí a vstupními/výstupními rozhraními. Takovými přístroji jsou například čipové karty RFID, mikrokontrolery apod. Slouží většinou k jednomu účelu, ke kterému byly vyvinuty. Programovací jazyky pro vývoj programů na takováto zařízení jsou obvykle nižší úrovně. Oblíbeným způsobem je tedy použití programovacího jazyka C.

Jelikož jsou tato zařízení výpočetně omezená, při implementaci kryptografických mechanismů s tímto musíme počítat. To řeší tzv. lehká kryptografie. Ta poskytuje takový stupeň bezpečnosti, která má co nejvyšší úroveň, ale je také zároveň uskutečnitelná na takovýchto zařízeních.

Je zde tedy několik řešení pro bezpečnost v IoT zařízeních, a to i pro taková, která musí splňovat dlouhodobou výdrž baterie a nemají velký výpočetní výkon. Ze symetrické kryptografie máme hned několik zástupců, především z blokových šifer je zde například CLEFIA vyvíjená společností Sony. CLEFIA používá Feistelovy sítě, kde je velikost bloku 128 bitů a klíče mohou být o velikosti 128, 192 nebo 256 bitů. Další možností je z blokových šifer algoritmus PRESENT, který má oproti CLEFIA dvakrát menší velikost bloku čili 64 bitů, a je založený na substitučně-permutační síti. Velikosti klíčů mohou být 80 nebo 128 bitů [10].

Zatímco lehká kryptografická primitiva veřejných klíčů vyžadují protokoly pro správu klíčů v sítích inteligentních objektů, požadovaný výpočetní výkon je mnohem větší než pro primitivní prvky symetrických klíčů. V současné době bohužel neexistují žádná slibná asymetrická lehká kryptografická primitiva, které by vyhovovaly dostatečnými bezpečnostními a nenáročnými vlastnostmi ve srovnání s konvenčními, které jsou založeny na eliptických křivkách nebo problému faktorizace.

3.2 Výběr kryptografických algoritmů

V dnešní době se kryptografie zaměřuje hlavně na dva matematické problémy, jmenovitě problém faktorizace a problém diskrétního logaritmu. Faktorizace je problém, kde pokud máme čísla $p, r, q \in \mathbb{Z}$, známe číslo p, poté najít rozklad tohoto čísla p, na prvočísla r a q, je velmi těžké dokázat, když platí:

$$p = r * q. \tag{3.1}$$

Druhým, již zmiňovaným, matematickým problémem v současnosné kryptografii je problém diskrétního logaritmu. Mějme tedy $a, b, C \in \mathbb{Z}$ a prvočíslo n, poté najděte

číslo b, pokud je zadána rovnice následovně:

$$C \equiv a^b \, (mod \, n) \, . \tag{3.2}$$

Výše popsané matematické problémy (3.1) a (3.2) jsou hojně využívány v dnešních kryptografických algoritmech, ať už je to RSA, DSA, či ECDH. V roce 1994 ale Peter Shor, americký profesor aplikované matematiky na MIT, přišel na způsob, jakým lze tyto problémy vyřešit pomocí kvantových počítačů, kvantové Fourierovy transformace, modulárního a binárního umocňování [11]. Tím se dnešní algoritmy stávají doslova nepoužitelnými v budoucnosti. Na řadu tedy přišly algoritmy, které využívají jiných matematických problémů, u kterých není zatím známo, že by byly řešitelné v dostačujícím čase. Post-kvantová kryptografie tedy implementuje řadu jiných matematických problémů, jako například mřížky, hashovací funkce, supersingulární isogenní eliptické křivky nebo také polynomiální rovnice.

V Tab. 3.1 můžeme najít srovnání několika málo post-kvantových algoritmů pro ustanovení klíčů. Dle této tabulky jsem vybral algoritmus NewHope, jelikož na výstupu dokáže generovat klíč o délce 256 bitů, který bude poté jednoduše přiveden na vstup algoritmu AES-256. Tento algoritmus také nemá velikou náročnost na paměť zařízení.

Tab. 3.1: Srovnání post-kvantových algoritmů.

Název protokolu	**Typ**	**Délka privátního klíče**	**Délka sdíleného klíče**
Frodo	Mřížky	39774 bajtů	256 bitů
SIDH	Křivky	48 bajtů	1504 bitů
Newhope	Mřížky	4096 bajtů	256 bitů
BCNS	Mřížky	4480 bajtů	128 bitů

V další kapitole si představíme zástupce post-kvantové kryptografie, zejména algoritmus pro ustanovení klíčů – NewHope. Poté, až budeme mít stanovené klíče na obou stranách komunikace, můžeme tento klíč využít v jednom z dostupných symetrických algoritmů pro šifrování citlivých dat, v našem případě půjde konkrétně o algoritmus AES. NewHope je také velmi rychlý a efektivní algoritmus kvůli jeho jednoduchým operacím nad mřížkami, což je pro nás příhodné k implementaci na zařízení s nízkým výpočetním výkonem.

3.2.1 NewHope

Mřížky

Pod pojmem mřížka si můžeme představit množinu bodů v n-rozměrném prostoru, která je generována linerárně nezávislými vektory $(\mathbf{v_1}, \ldots, \mathbf{v_n}) \in \mathbb{R}^n$ a definujeme ji následovně:

$$\mathcal{L} = \{a_1 v_1 + \cdots + a_n v_n; a_i \in \mathbb{Z}\}, \tag{3.3}$$

kde n je prvočíslo, někdy se také značí q. Mřížky se opírají o dva důležité matematické problémy, které právě využívají šifrovací algoritmy. Těmi jsou problém nejkratšího vektoru (Shortest Vector Problem – SVP), a problém nejbližšího vektoru (Closest Vector Problem – CVP). Oba tyto problémy se řadí mezi NP-těžké.

SVP je takový problém, kde zkoušíme najít nejkratší vektor báze. Pokud máme tedy mřížku $\mathcal{L}$ a jeho libovolnou bázi, tak je cílem najít takový nenulový vektor, který je co nejblíže nule [12]. Druhý problém se poté zabývá nalezením nejbližšího vektoru vůči jakémukoliv vektoru mřížky. Tento vektor může být i nulový.

Popis algoritmu NewHope

Algoritmus NewHope je jedním z nových post-kvantových algoritmů, který byl predložen na soutěži post-kvantových algoritmů vyhlášené organizací NIST v roce 2016. Vychází z protokolu BCNS, který je založený na problému R-LWE (Ring-learning-with-errors) a autoři NewHope se snažili napravit chyby tohoto algoritmu a zefektivnit jej. Například používá mřížku D_4, která umožňuje snížit modulo na $q = 12289 < 2^{14}$, parametr $\mathbf{a}$ se teď generuje nově při každém ustanovení klíčů nebo při distribuci chyb nahrazuje diskrétní Gaussovo rozložení za binomické rozložení, jehož střední hodnota je 0 a rozptyl $k/2$, které je mnohem efektivnější [13].

Autoři protokolu NewHope definují parametry, které by měly být dle nich dlouhodobě dostatečně bezpečné, s poskytnutím 128 bitové post-kvantové bezpečnosti. Všechny polynomy, kromě $\mathbf{r} \in \mathscr{R}_4$, jsou definovány v okruhu $\mathscr{R}_q = \mathbb{Z}_q[\mathrm{X}]/(\mathrm{X}^n + 1)$, dimenze $n = 1024$, modulo $q = 12289$ a binomické rozložení ψ_k, kde $k = 16$.

Nejprve se vygeneruje veřejný parametr $\mathbf{a}$ pro každou výměnu klíču. Tento parametr se generuje z náhodného 32 bitového *seed* – (3.4), který je rozšířený v algoritmu SHAKE-128 ze standardu FIPS-202 [14] podle (3.5), resp.

$$seed \leftarrow \{0,1\}^{256} \text{ a} \tag{3.4}$$

$$\mathbf{a} \leftarrow \text{Parse}(\text{SHAKE-128}(seed)). \tag{3.5}$$

Obě dvě strany si následně vygenerují soukromý klíč a chybové stavy pomocí binomického rozložení ψ_{16}^n a proudové šifry ChaCha20. Alice si tedy vygeneruje privátní

klíč a chybovou mřížku $\mathbf{s}, \mathbf{e} \leftarrow \psi_{16}^n$, Bob potom $\mathbf{s}', \mathbf{e}', \mathbf{e}'' \leftarrow \psi_{16}^n$. Alice vypočítá $\mathbf{b}$ z právě vygenerovaných parametrů pomocí (3.6)

$$\mathbf{b} \leftarrow \mathbf{as} + \mathbf{e}. \tag{3.6}$$

Alice potom pošle parametr $\mathbf{b}$ společně se *seed* přes tzv. NTT (Number-Theoretic Transform). Polynomy jsou zakódovány jako pole o velikosti 1792 bajtů a poté komprimovány do menšího formátu. *Seed* je pole o velikosti 32 bajtů a je kódován společně s $\mathbf{b}$ [13]. První zpráva od Alice k Bobovi tedy má celkově 1824 bajtů. Bob na druhé straně vygeneruje také svoje $\mathbf{a}$, stejným způsobem jako Alice – (3.5). Ze získaných parametrů od Alice si Bob vypočítá pomocí (3.7) parametry $\mathbf{u}$ a $\mathbf{v}$ následovným způsobem:

$$\mathbf{u} \leftarrow \mathbf{as}' + \mathbf{e}'; \mathbf{v} \leftarrow \mathbf{bs}' + \mathbf{e}''. \tag{3.7}$$

Dále na řadu přichází dvě funkce pro obnovení z chybových stavů – `HelpRec` a `Rec`. Tyto funkce jsou založené na hledání nejbližšího vektoru ve 4-rozměrné mřížce s bází

$$B_4 = \begin{pmatrix} 1 & 0 & 0 & 0{,}5 \\ 0 & 1 & 0 & 0{,}5 \\ 0 & 0 & 1 & 0{,}5 \\ 0 & 0 & 0 & 0{,}5 \end{pmatrix}.$$

`HelpRec` nejprve rozdělí 1024 koeficientů vloženého polynomu $\mathbf{v}$ do 256 čtyř rozměrných vektorů $\mathbf{x}_i = (\mathbf{v}_i, \mathbf{v}_{i+256}, \mathbf{v}_{i+512}, \mathbf{v}_{i+768})^t$ pro $i = 0, \ldots, 255$. Poté vypočítá nápravnou informaci $\mathbf{r}_i$ z $\mathbf{x}_i$ za pomoci

$$\mathbf{r}_i = \texttt{HelpRec}(\mathbf{x}_i, b) = \mathrm{CVP}_{D_4}\left(\frac{2^r}{q}(\mathbf{x}_i + b\mathbf{g})\right) \bmod 2^r, \tag{3.8}$$

$$\mathbf{r} \leftarrow \texttt{HelpRec}(\mathbf{v}), \tag{3.9}$$

kde b je náhodný bit a $\mathbf{g} = (0.5, 0.5, 0.5, 0.5)^t$. Bob tedy použije funkci `HelpRec` pro vytvoření parametru $\mathbf{r}$ pomocí (3.9). Parametry $\mathbf{r}$ a $\mathbf{u}$ Bob pošle Alici, což je druhá zpráva, a poslední, vyměněná mezi účastníky komunikace, s celkovou délkou 2048 bajtů. Alice si dopočítá vlastní $\mathbf{v}'$ pomocí (3.10)

$$\mathbf{v}' \leftarrow \mathbf{us}. \tag{3.10}$$

Obě strany poté provedou rekonciliaci parametru v funkcí `Rec`, která pak funguje obdobně jako `HelpRec`, také pracuje s 4-rozměrným vektorem, jen je definována jako $\texttt{Rec}(\mathbf{x},\mathbf{r}) = \texttt{LDDecode}(\frac{1}{q}\mathbf{x} - \frac{1}{2^r}\mathbf{Br})$. Alice na vstup této funkce zadá parametry $\mathbf{v}', \mathbf{r}$ a Bob $\mathbf{v}, \mathbf{r}$:

$$v \leftarrow \texttt{Rec}(\mathbf{v}', \mathbf{r}), v \leftarrow \texttt{Rec}(\mathbf{v}, \mathbf{r}). \tag{3.11}$$

Posledním krokem k ustanovení konečného klíče μ je přivedení vypočítaného parametru v na vstup funkce SHA3-256 na obou stranách a je vše hotovo.

$$\mu \leftarrow \text{SHA3-256}(v) \tag{3.12}$$

3.2.2 AES – Advanced Encryption Standard

Dnes standardně používaná symetrická bloková šifra, která disponuje stejnými klíči pro šifrování i dešifrování. AES nahrazuje starší šifrovací algoritmus DES, který byl chybný v tom, že implementoval malou délku klíčů, což by na tu dobu stačilo. Jenže jak výpočetní výkon zařízení roste, 64 bitový klíč (56 bitů efektivních) již nestačil a šifra byla prolomena hrubou silou. Jeho variantou byl také 3DES, který je i dnes poměrně bezpečný, jen je oproti AES pomalý.

AES umožňuje 3 typy šifrovacích klíčů – AES-128 se 128 bitovým klíčem, AES-192 se 192 bitovým klíčem, a AES-256 s 256 bitovým klíčem. Pro každý tento typ je různý počet rund čili 10 rund pro 128 bitový klíč, 12 pro 192 bitový, resp. 14 pro 256 bitový. Všechny typy AES šifrují bloky textu o 128 bitech. Šifrovací algoritmus je založený na substitučně-permutační sítí, která prochází následujícími operacemi:

- ByteSub – substituční tabulka,
- ShiftRow – bitový posun řádků,
- MixColumn – záměna sloupců,
- AddRoundKey – přidání rundového klíče.

Blokové šifry také implementují různé operační módy pro zvýšení náhodnosti a vyhnutí se kolizím. Operační módy umožňují šifrovat data ne blok po bloku, jak by normálně AES fungoval (mód ECB – Electronic Codebook), ale s určitou vazbou mezi každým blokem. Jedním z operačních módů je CBC (Cipher Block Chaining), který zavádí před první rundou šifrování ještě Inicializační Vektor, a nad každým následujícím otevřeným textem je provedena operace XOR se šifrovaným textem předchozí rundy. Dalšími módy mohou být například OFB (Output Feedback), CTR (Counter), CFB (Cipher Feedback), GCM (Galois/Counter Mode), a jiné. AES má také hlavní výhodu, že zatím nebyla nalezena slabina ani s použitím kvantových počítačů. Obdobně jako algoritmus NewHope poskytuje 128-bitovou post-kvantovou bezpečnost při použití varianty s 256-bitovým klíčem [15].

Pomocí AES tedy budeme šifrovat uživatelská data, která vytvoří senzor. Použijeme AES v operačním módu GCM s 256 bitovým klíčem. Tyto data poté budeme přenášet přes NB-IoT síť až k uživateli, který musí mít tedy pochopitelně stejný šifrovací klíč, který byl vložen do blokové šifry, aby mohl tyto data dešifrovat. Pro zajištění ustanovení klíčů musíme implementovat jeden z dostupných kryptografických algoritmů pro ustanovení klíčů mezi oběma stranami, v našem případě půjde o post-kvantový algoritmus NewHope.

GCM – Galois/Counter Mode

Jediný mód, který je pro nás zajímavý, je GCM – Galois/Counter Mode, jelikož dokáže nejen zašifrovat otevřený text, ale také zajištění autentičnosti dat. V roce 2007

byl publikován americkou agenturou NIST jako doporučený mód pro AES. Mohli bychom sice implementovat AES v jiném módu, například CBC, ale to by pro nás nebylo již tak zajímavé, jelikož bychom s tímto módem museli implementovat také další algoritmus pro ochranu integrity nebo autentičnosti. S módem GCM máme jistotu, že data, která jsme zašifrovali, nebyla nikým pozměněna. GCM se také nazývá jako mód autentizovaného šifrování [15].

GCM zastřešuje 2 módy, CM – Counter Mode, který se stará o šifrování otevřených dat a také Galois mód, který vypočítá tzv. Autentizační tag, čímž zajiští autentičnost šifrovaných dat. Na vstup tohoto módu musí být přivedeny 2 povinné parametry a jeden nepovinný. Povinnými parametry jsou Inicializační Vektor IV a otevřený text, který chceme šifrovat. Nepovinným parametrem jsou doplňující údaje. Otevřený text a doplňující údaje jsou 2 různé kategorie dat, které GCM chrání. Zatímco u otevřeného textu zajišťuje jak autentičnost, tak i důvěrnost dat, u doplňujících údajů GCM mód zajišťuje jen autentičnost. Pokud totiž chceme implementovat GCM mód v internetovém protokolu, doplňující údaje mohou zahrnovat například internetové adresy, porty, pořadová čísla packetů, verze používaných protokolů a jiné.

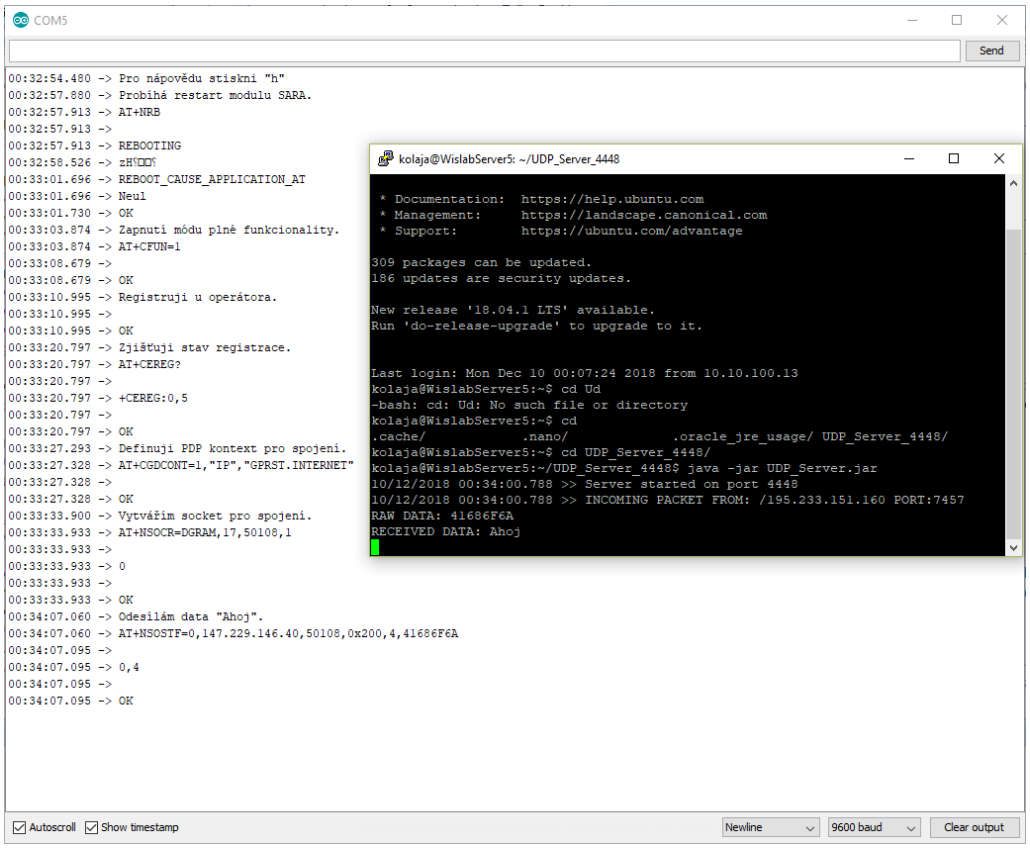
3.3 Zprovoznění komunikace NB-IoT

V rámci praktické části bylo stanovena komunikace mezi VUT serverem a NB-IoT modulem umístěným na desce ArduinoMega 2560. Na VUT serveru běží jednoduchá aplikace, napsaná v programovacím jazyce Java, která vytvoří soket na určitém portu a IP adrese pro následné přijmutí dat. K tomu, abychom mohli komunikovat se serverem, je potřeba se nejdříve zaregistrovat do sítě operátora pomocí AT příkazů. AT příkazy, také známé jako Hayesovy příkazy, jsou série krátkých textových příkazů, pomocí nichž se ovládá zařízení komunikující v mobilních sítích. Pomocí mikrokontroléru a jeho sériové linky jsou modulu SARA-N210 předávány příkazy se syntaxí `"AT+<název_příkazu><parametry><CR>"`, kde:

- `AT+` značí prefix každého příkazu,
- `<název_příkazu>` je řetězec znaků se jménem příkazu,
- `<CR>` jsou poté znaky označující konec příkazu, v tomto případě `"\r\n"`.

V práci se používaly příkazy pro restart modulu SARA-N210, přepínání módu funkcionality, registraci zařízení do sítě operátora, zjištění stavu registrace u operátora, přiřazení PDP (Packet Data Protocol) kontextu, vytvoření socketu na daném portu a příkazu pro odeslání dat na server VUT. Návratové hodnoty většiny příkazů mohou být jen dvě – `"OK"` nebo `"ERROR"`. V případě příkazu pro zjištění registrace očekáváme návratovou hodnotu `+CEREG:0,5` a u odeslání dat na server `0,4` a následně `"OK"`. Těmito příkazy jsou poté konkrétně:

- `AT+NRB\r\n` pro restart modulu,
- `AT+CFUN=1\r\n`, který přepne funkcionalitu modulu do módu plné rádiové komunikace,
- `AT+COPS=1,2,"23003"\r\n` vynucuje pokus o registrování SIM u operátora,
- `AT+CEREG?\r\n` pro zjištění stavu registrace u operátora,
- `AT+CGDCONT=1,"IP","GPRST.INTERNET"\r\n`, který definuje hodnoty parametrů připojení pro PDP kontext,
- `AT+NSOCR=DGRAM,17,50108,1\r\n` vytvoří lokální soket pro komunikaci se serverem a
- `AT+NSOSTF=0,147.229.146.40,50108,0x200,4,41686F6A\r\n` pro odeslání dat `41686F6A` na server s IP adresou 147.229.146.40.



Obr. 3.1: Série příkazů pro stanovení spojení a odeslání dat na VUT server.

Na Obr. 3.1 je možné vidět sérii příkazů vykonaných pro registraci zařízení do sítě operátora a následné odeslání dat na server VUT. Je nutné podotknout, že na straně serveru VUT v koncové aplikaci není zajištěno žádné šifrování, ani ustanovení klíčů mezi koncovými zařízeními. Je to aplikace čistě na vypsání přijatých dat na určité IP adrese a portu. NB-IoT modul komunikuje přes nespolehlivý a nezabezpečený protokol UDP a tento fakt znázorňuje, že data, která jsme vyslali z NB-IoT modulu na server nebyla žádným způsobem chráněna.

3.4 Návrh zabezpečení

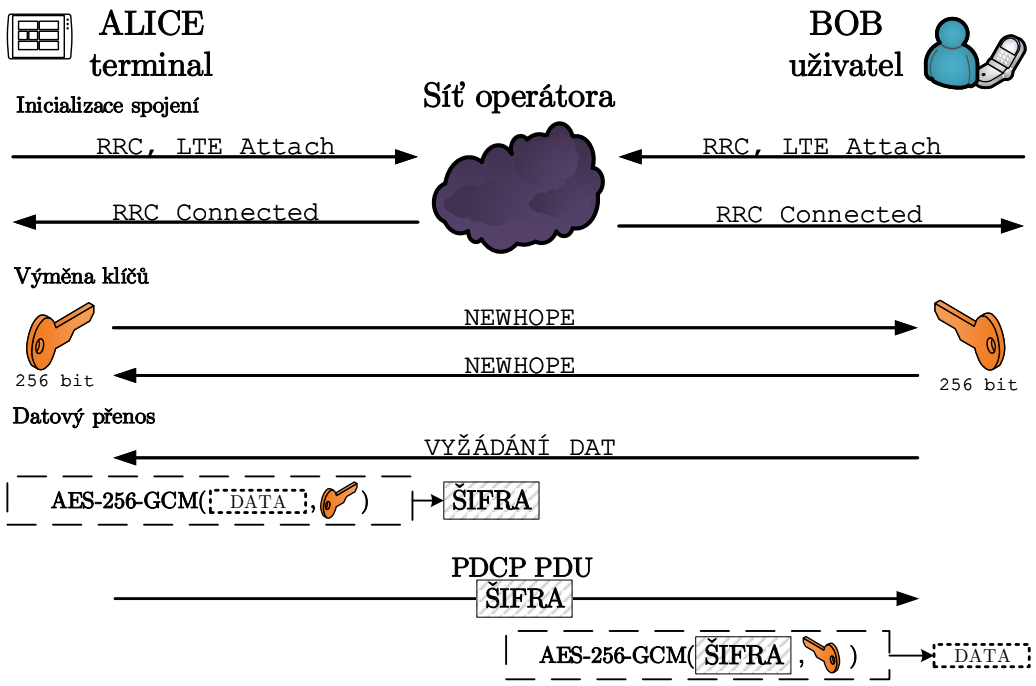
Dle kapitoly 2.5.3 můžeme vidět, že v celé komunikaci je PDCP jediný protokol, který se stará o šifrování a integritu přenášených dat v sítích LTE. Celý tento návrh staví na tom, že potencionální útočník nezná klíč a ani parametry, které jsou dodávané do šifrovacího algoritmu (Obr. 2.6b), resp. do algoritmu pro ochranu integrity (Obr. 2.6a). Jelikož jsou všechny klíče uložené u operátora v MME, viz kapitola 2.5.1, můžeme mít jistotu, že jsou data šifrovaná na cestě ze senzoru do sítě operátora. Dále ale musí jít data přes Internet až k našemu zařízení, kde tyto předešle vyžádaná data přijeme [1]. Jak můžeme vidět v kapitole 3.3, data putující internetem směrem od operátora do našeho zařízení jsou přenášena protokolem UDP. Tento protokol je ideální pro využití v NB-IoT sítích, jelikož zařízení nemusí navazovat spojení pokaždé když se přenáší data, což šetří spotřebu energie. Nicméně jakmile UDP paket opustí síť operátora, jeho data jsou v otevřeném formátu a viditelná třetím stranám.

Návrh na Obr. 3.2 přejímá odlišný přístup k zabezpečení komunikace NB-IoT. Prerekvizitou celého návrhu je připojení zařízení do sítě operátora. Jakmile se daná zařízení připojí do sítě operátora a budou zaregistrována, mohou začít přenášet data. Jelikož nechceme, aby data vysílaná z NB-IoT modulu nebyla přenášena Internetem otevřeně, budeme implementovat šifrovací algoritmus na aplikační vrstvě, který zašifruje data generovaná senzorem a pošle tyto data přes síť až k uživateli. Na druhé straně uživatelská aplikace v mobilním zařízení tyto data dešifruje zpět do otevřeného textu, čitelného uživatelem.

K realizaci šifrovacího algoritmu je nutnost implementovat i algoritmus pro ustanovení klíčů mezi stranami, které spolu komunikují přes nezabezpečené médium – Internet. Ustanovení klíčů mezi oběma stranami obstarává post-kvantový algoritmus založený na mřížkách, NewHope, poskytující bezpečnost přes 128-bitů. Jeho výstupem je poté 256-bitový sdílený klíč, který bude uložen v obou koncových zařízení. Jelikož máme stanovený klíč, můžeme pomocí mikrokontroléru šifrovat data generovaná senzorem. K tomuto použijeme šifrovací algoritmus AES pracující v operačním módu GCM, který zajišťuje nejen důvěrnost dat šifrováním, ale poskytuje také autentizaci dat. Uživatel na druhé straně tedy bude vědět, že originální data vyslaná modulem NB-IoT nebyla po cestě nijak pozměněna. Na mikrokontroléru budou na vstup algoritmu AES-256-GCM přivedena data a sdílený klíč, dříve stanovený pomocí NewHope. Výstupem bude šifra s autentizačním tagem, která se poté předá NB-IoT modulu a pošle se Internetem směrem k uživateli. Uživatel po přijmutí dat vypočítá autentizační tag pomocí sdíleného klíče a obdržené šifry a porovná s pří-

[1] Informace poskytnuta společností Accent systems zabývající se vývojem a výrobou NB-IoT řešení, dostupná na adrese `https://accent-systems.com/blog/security-of-nb-iot-devices/`

chozím tagem. Pokud jsou autentizační tagy shodné, ví, že data nebyla pozměněna a může šifrový text dešifrovat pomocí stejného algoritmu a originální data přečíst.



Obr. 3.2: Návrh zabezpečení NB-IoT komunikace.

4 Implementace návrhu zabezpečení

V této části se bakalářská práce zabývá praktickou implementací na zařízeních představených v kapitole 4.1. V rámci této kapitoly bude také dále představeno programové vybavení a softwarové knihovny, za pomocí nichž byla realizována implementace návrhu zabezpečení. V neposlední řadě zde bude také uvedeno jakým postupem byla implementace dosáhnuta a celé řešení představeno.

4.1 Používaná zařízení

K uskutečnění našeho návrhu budeme používat zařízení s omezeným výkonem, konkrétně mikrokontrolér ATmega2560, společně se shieldem, na kterém je umístěn NB-IoT modul SARA-N210. Arduino Mega 2560 je vývojová deska, která umožňuje vymýšlet nové prototypy na mikrokontroléru ATmega2560. Disponuje také kolíkovými lištami, které umožňují přístup ke vstupním a výstupním periferiím mikrokontroléru. Dá se na ně také připojit velká škála tzv. shieldů, které dodávají různou funkcionalitu vývojové desce od senzorů teploty až po měřiče kvality ovzduší, aj. Na kontrolér je připojený shield s NB-IoT modulem SARA-N210 vyráběný firmou ublox, konkrétně verze 01B-00. Z pohledu hardwarového vybavení se jedná o verzi Arduina, která je jedna z výkonnějších. Technické parametry[1] ATmega2560 mikrokontroléru jsou tedy následující: CPU – ATmega2560 pracující na frekvenci 16 MHz, 8 KB SRAM operační paměti, 256 KB Flash paměti, z čehož 8 KB zabírá bootloader, 4 KB EEPROM paměti a rozhraní USB typu B. Jediná EEPROM paměť má omezení, a tím je životnost, která je udávaná na sto tisíc zapisovacích cyklů. S EEPROM pamětí se ale v naší implementaci nepracuje.

Komunikace NB-IoT vždy probíhala mezi modulem SARA umístěným na shieldu Arduina a serverem, který je umístěn v laboratořích Wislab týmu na fakultě FEKT Vysokého Učení Technického v Brně. Tento server tedy sloužil jako simulace koncové aplikace uživatele, která by měla zpracovávat data generované a poslané NB-IoT modulem. Na serveru běží operační systém Linux s distribucí Ubuntu ve verzi 16.04.2 LTS. Verze jádra je 4.4.0-72-generic. Na server se přistupovalo jen vzdáleně, skrz aplikaci PuTTy, a tudíž chyběla nástavba grafického rozhraní, pomocí níž by uživatel mohl pracovat na implementaci návrhu. Vlastní skripty byly proto vyvíjeny lokálně ve virtuálním prostředí Oracle VM VirtualBox v operačním systému Linux, který byl také distribuce Ubuntu, jen s vyšší verzí 18.10 a také vyšší verzí jádra – 4.18.0-18-generic. Vyvinuté skripty byly potom nahrány přes protokol SFTP na úložiště serveru pomocí programu WinSCP.

[1] Více informací na `https://store.arduino.cc/arduino-mega-2560-rev3`

4.2 Příprava prostředí

Pro to, abychom mohli začít vyvíjet skripty, ať už pro Arduino zařízení nebo pro serverovou implementaci v Linuxu, musíme mít určité softwarové vybavení. Na obou stranách budeme programovat v programovacím jazyce `C++`.

Na straně serveru je to jednoduché. Tam potřebujeme jakýkoliv textový editor, který umožňuje zvýraznění syntaxe programovacího jazyka `C++` pro zjednodušení práce a dále terminál, pomocí kterého si budeme moct kompilovat výsledné zdrojové kódy. Samotná kompilace poté probíhá pomocí kompilátoru `g++`, který by měl být automaticky součástí každého jádra Linuxových distribucí.

Co se týče vyvíjení skriptů, resp. skic (v anglickém jazyce sketch) – tak se nazývají samotné skripty pro Arduino, zde budeme potřebovat mít nainstalované vývojové prostředí Arduino IDE. Arduino IDE je dostupné také jako webový editor, ale ten není v našem případě využívaný. Arduino IDE je dostupný pro všechny známé operační systémy – Windows, Mac OS X, i Linux. Při vývoji byla použita verze Arduino IDE 1.8.9.

4.2.1 Arduino IDE

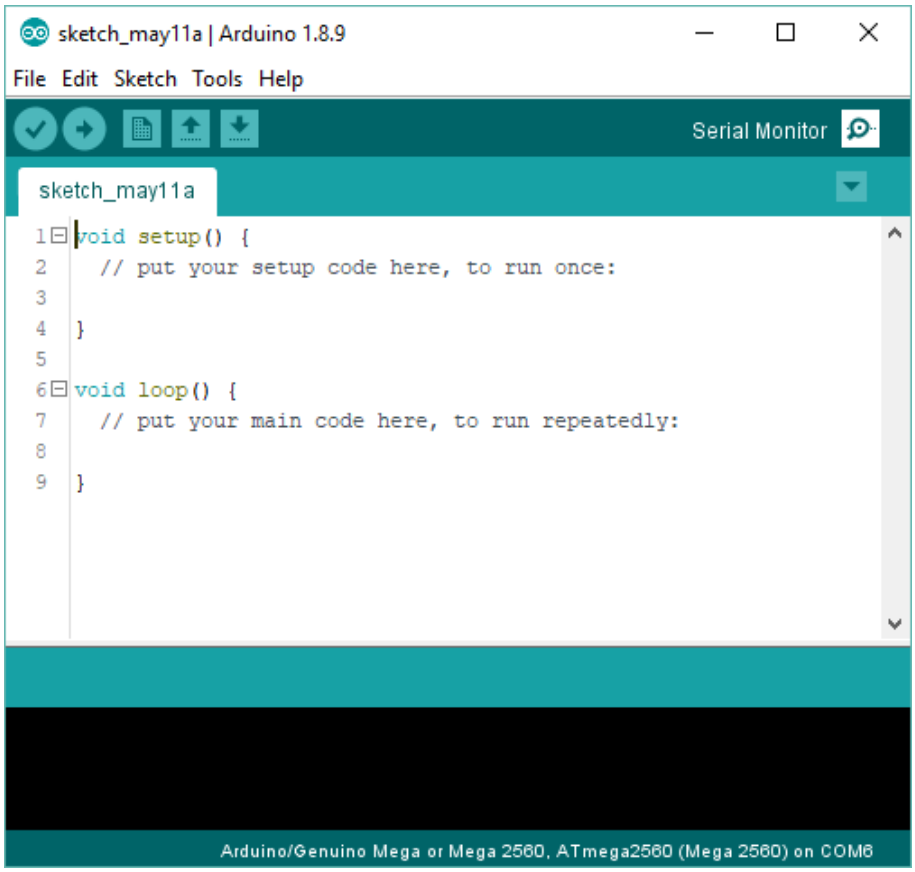
Jak již bylo naznačeno, Arduino IDE je vývojový program sloužící pro psaní zdrojových kódů, resp. skic, v programovém jazyce `C++`. Součástí tohoto programu jsou i kompilátory `avr-gcc` a `avr-g++`, které se starají o kompilaci zdrojových kódů do `.hex` souboru, který se dále nahraje do paměti zařízení, ze které se spouští.

Samotné skicy jsou nahrávány do zařízení pomocí `avrdude`, což je integrovaný nástroj Arduino IDE, který uchovává technické parametry různých desek Arduino a podle nichž dělá iniciální nastavení zařízení před nahrátím a spuštěním skic.

Pokud připojíme vývojovou desku k počítači, operační systém si automaticky vyhledá ovladače pro takové zařízení. Pokud se tak nestane, na oficiálních stránkách Arduina jsou dále návody jak postupovat při nesprávném rozpoznání zařízení. Jakmile je správně rozpoznáno zařízení na PC, můžeme spustit aplikaci Arduino IDE. Na obrázku 4.1 můžeme vidět její obrazovku po spuštění. Nejdůležitější věcí je mít aplikaci nastavenou přímo pro naše zařízení, což můžeme vidět ve stavovém řádku dole nalevo. První část nám říká pro které zařízení je aplikace nastavena, druhou informací je poté mikrokontrolér a ve třetí části můžeme najít na kterém seriovém portu máme zařízení připojení.

Pokud tyto informace nejsou dostupné ve stavovém řádku, můžeme je ručně v aplikaci nastavit. Toto je možné provést pomocí hlavního menu programu v záložce *Tools*, která je rozdělena do tří oblastí oddělených čarou. Právě ve třetí části

můžeme najít tři sub-menu, ve kterých manuálně nastavíme naši desku (*Board*), mikrokontrolér (*CPU*) a na kterém seriovém portu je deska připojena (*Port*).



Obr. 4.1: Prostředí Arduino IDE.

Jakmile je deska připravena k nahrávání skic, můžeme vlastní skicy začít vyvíjet. V hlavním okně jsou po spuštění programu vidět dvě metody, které jsou povinné při vývoji skic. Jedna z nich je `setup()`, do které se umisťují iniciální nastavení zařízení nebo proměnných. Metoda `setup()` se provede vždy při startu zařízení nebo restartování. Druhou metodou je `loop()`, ve které je poté hlavní zdrojový kód, který se vykonává ve smyčce neustále do nekonečna.

4.2.2 Linux Ubuntu

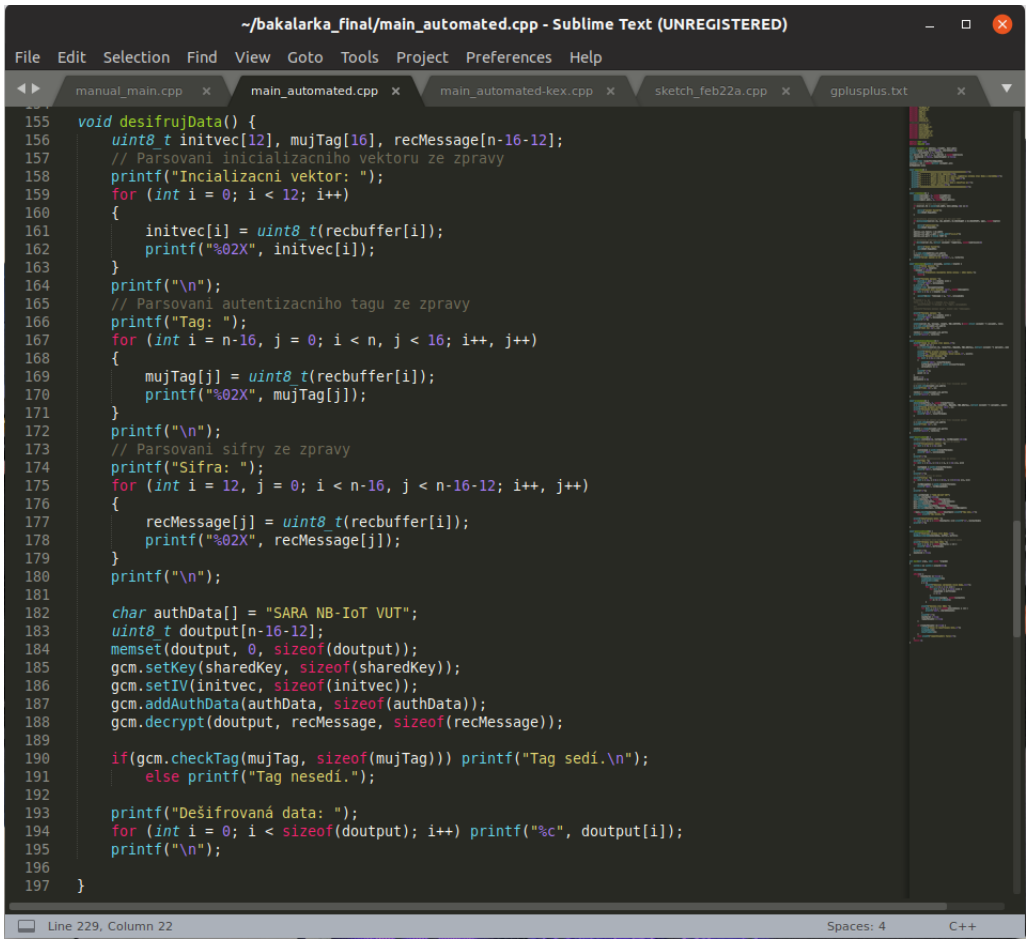
Jak již bylo nastíněno v kapitole 4.2, skript běžící na serveru budeme vyvíjet lokálně v operačním systému Linux v patřičné distribuci, která umožňuje zdrojový kód kompilovat pomocí kompilátoru `g++`.

Nastavení prostředí v Linuxu je docela jednoduché a záleží spíše na každém jedinci jaké má preference pro vývoj programů na platformě Linux. V našem případě

jsme ale zvolili textový editor *Sublime Text 3*, který umožňuje automatické zvýraznění syntaxe v podstatě jakéhokoliv programovacího jazyka a také má funkci pro automatické doplnění klíčových slov.

Pro kompilaci zdrojového kódu je za potřebí jen a pouze terminál a příkazu `g++` s přepínači:

- `-I` – sloužící pro určení zdrojových souborů knihoven,
- `-o` – sloužící ke specifikaci jména výsledného souboru.



Obr. 4.2: Prostředí Linux - Sublime Text.

Mezi těmito přepínači je potřeba uvést všechny zdrojové kódy, které jsou potřebné pro překlad do výsledného skriptu. Překladač `g++` si totiž nejdříve z každého souboru s příponou `.cpp` vytvoří objektový soubor s příponou `.o` a dále z těchto objektů poskládá výsledný skript.

4.3 Používané knihovny

Pro to, abychom mohli začít vyvíjet skicy na Arduino a také skripty pro serverové řešení, musí být zvoleny knihovny pro lehčí implementaci těchto programů. Na straně Arduina byla využívána rozsáhlá knihovna Arduino Cryptography Library, které je dedikována kapitola 4.3.1, a dále standardní knihovny pro práci s textovými řetězci, jako například `string.h` nebo `stdio.h`.

S Linuxovým řešením se k používaným knihovnám na Arduinu, Arduino Cryptography Library a standardním knihovnám pro práci s textovými řetězci, přidávají ještě knihovny pro práci se síťovými sokety. Těmito knihovnami jsou `sys/socket.h` a `arpa/inet.h`. Knihovna `sys/socket.h` slouží k definicím jednotlivých datových typů a struktur proměnných pro tvorbu socketu a `arpa/inet.h` nám umožní práci s proměnnými typu `in_port_t` nebo `in_addr_t`.

4.3.1 Arduino Cryptography Library

Arduino Cryptography Library je knihovna dostupná všem, kteří chtějí provádět kryptografické operace a implementovat kryptografické algoritmy na omezená zařízení jako je Arduino. Knihovna je distribuována pod licenčními podmínkami MIT licence. Dělí se na čtyři oblasti – jádro, odlehčené verze, originální a ostatní algoritmy [16]. Jelikož náš návrh zahrnuje algoritmy NewHope a AES256-GCM, budeme využívat oblast jádra, kam spadá AES256-GCM a dále ostatní algoritmy, kde můžeme najít NewHope.

AES256-GCM v Arduino Cryptography Library

Co se týče algoritmu AES256-GCM, zde implementujeme dvě knihovny do našeho projektu. Těmi jsou knihovny `GCM.h` a `AES.h`. Knihovna `GCM.h` obsahuje implementaci operačního módu AES, Galois Counter Mode, samostatného. Z této knihovny budeme pro šifrování využívat funkce vypsané ve výpise 4.1. V následující části si představíme jednotlivé funkce a jejich úlohu.

Výpis 4.1: Využívané funkce GCM.h knihovny – šifrování

```
GCM<AES256> gcm;
gcm.setKey(key, sizeof(key));
gcm.setIV(iv, sizeof(iv));
gcm.addAuthData(adata, sizeof(adata));
gcm.encrypt(ciphertext, plaintext, sizeof(plaintext));
gcm.computeTag(tag, sizeof(tag));
```

Pro to, abychom mohli začít využívat funkce této knihovny, musíme mít definovanou proměnnou typu *GCM<T>*, kde *T* je typ blokové šifry, v našem případě AES256. Metoda `setKey(key, sizeof(key))` nastaví šifrovací klíč pro algoritmus, který se bude do šifrovacího algoritmu přivádět. Šifrovací klíč bude předem stanovený pomocí algoritmu NewHope. Na vstupu má dva parametry – *key*, což je ukazatel na proměnnou klíče, který se použije, a *sizeof(key)*, což je délka klíče vkládaného do algoritmu. Jelikož využíváme algoritmus AES256, náš klíč bude mít velikost 32 bajtů, respektive 256 bitů. Jejím výstupem je poté hodnota ***true*** pokud je klíč přijat. Je také možný výstup této funkce ***false*** pokud není vstupní délka klíče správná nebo pokud je klíč slabý z pohledu bezpečnosti a nedá se šifrou využít [16].

Metoda `setIV(iv, sizeof(iv))` nastavuje inicializační vektor pro následné použití v GCM šifrovacím algoritmu. Obdobně jako u metody `setKey()` jsou jejími vstupy parametry *iv*, což je ukazatel na proměnnou inicializačního vektoru, a *sizeof(iv)*, který určuje délku použitého inicializačního vektoru. Samotný inicializační vektor je generovaný náhodně pomocí knihovny `RNG.h`, která je také součástí ACL. Z této knihovny jsou pro generaci inicializačního vektoru použity dvě funkce. Těmi jsou `begin(const char * tag)`, která inicializuje náhodný generátor čísel přidáním jejího parametru *tag* ze vstupu pro zvýraznění náhodnosti generovaného výstupu. Následné volání metody `rand(data, sizeof(data))` vygeneruje náhodná data do ukazatele proměnné *data*, která je předávána na vstupu metody. Tuto metodu tedy použijeme pro generaci náhodného inicializačního vektoru na začátku každého šifrování. Výstupem metody `setIV()` je poté buď ***true*** pokud je inicializační vektor přijat nebo ***false*** v případě, že není délka inicializačního vektoru podporována. Používaná délka inicializačního vektoru je v našem případě 12 bajtů, resp. 96 bitů. Tato délka je doporučována organizací NIST [15].

Další metodou je `addAuthData(adata, sizeof(adata))`, která se využívá pro autentizaci šifrovacích dat, které se posílají společně se šifrou. Autentizační data jsou posílána v otevřeném textu a v algoritmu je pomocí nich a samotné šifry počítán výsledný autentizační tag. Vstupem této metody jsou ukazatel na autentizační data a délka autentizačních dat.

Samotným jádrem celého šifrovacího algoritmu je metoda `encrypt(ciphertext, plaintext, sizeof(plaintext))`. Metoda samotná je velmi jednoduchá a zašifruje data předávané parametrem *plaintext* do šifry *ciphertext*. Třetí parametr, *sizeof(plaintext)*, slouží ke specifikaci délky vstupního otevřeného textu. Dle [15] lze do AES256-GCM předat až 60 GiB dat na výkonnějších výpočetních strojích.

Finálním krokem k zakončení celého procesu šifrování je výpočet autentizačního tagu ze šifrovaných a autentizačních dat. Výstupem metody `computeTag(tag, sizeof(Tag))` je tedy ukazatel na *tag*, který je také přiveden na vstup.

Poté, co druhá strana, se kterou komunikujeme, přijala data, musí nejprve ověřit,

zdali s daty nebylo manipulováno a následně může dešifrovat přijatá data. K tomuto úkonu potřebuje provést metody uvedené ve výpise 4.2.

Výpis 4.2: Využívané funkce GCM.h knihovny – dešifrování

```
GCM<AES256> gcm;
gcm.setKey(key, sizeof(key));
gcm.setIV(iv, sizeof(iv));
gcm.addAuthData(adata, sizeof(adata));
gcm.checkTag(tag, sizeof(tag)));
gcm.decrypt(ciphertext, plaintext, sizeof(ciphertext));
```

První tři uvedené metody probíhají stejně jako při procesu šifrování. Při procesu dešifrování jsou jedinými změnami akorát metody `checkTag(tag, sizeof(tag))` a `decrypt(ciphertext, plaintext, sizeof(ciphertext))`. Tyto metody jsou opakem `computeTag()`, resp. `encrypt()`.

Jakým způsobem se implementuje metoda `checkTag()` je poté jen na vývojáři, ale měla by být vždy zajištěná zpětná vazba uživateli, pokud se zjistí pomocí této metody, že bylo s daty někým neautorizovaným zacházeno. Pokud by k takovým úpravám došlo, mělo by být zajištěno, že data nebudou k ničemu využívána a budou zahozena.

NewHope v Arduino Cryptography Library

S knihovnou NewHope je to jednodušší, než s AES256-GCM knihovnou, kdy nám stačí importovat jediný soubor – `Newhope.h`. Zde máme pro implementaci k dispozici jen a pouze tři metody, které se využívají pro výměnu klíčů mezi účastníky komunikace. Výpis 4.3 ukazuje, co vše je potřeba pro implementaci algoritmu NewHope pro ustanovení sdíleného klíče.

Pokud začneme s proměnnými, které pro implementaci potřebujeme, pro úsporu

Výpis 4.3: Využívané funkce a proměnné NewHope.h knihovny

```
uint8_t alice_public[1824];
uint8_t bob_public[2048];
uint8_t shared_secret[32];
NewHopePrivateKey alice_private;
keygen(alice_public, alice_private);
sharedb(shared_secret, bob_public, alice_public);
shareda(shared_secret, alice_private, bob_public);
```

místa se pro proměnné *alice_public* a *bob_public* může alokovat jediné pole o velikosti 2048 bajtů. Struktura *NewHopePrivateKey* se poté skládá ze seedu o velikosti 32 bajtů nebo koeficientu o velikosti 1024 bajtů dle toho, zdali je deska založena na AVR (Alf, Vegard a Risc) architektuře. V našem případě je, takže je používán jen seed ze struktury *NewHopePrivateKey*.

Zařízení, které začíná komunikaci, si musí vygenerovat nejdříve pár veřejného a soukromého klíče pomocí metody `keygen(alice_public, alice_private)`. V našem případě vždy začíná komunikaci zařízení Arduino, tudíž *Alice_public* je posláno na zařízení se kterým komunikujeme, se serverem.

Koncové zařízení, se kterým komunikujeme, provede pouze metodu `sharedb( shared_secret, bob_public, alice_public)`, kde veřejným klíčem, který se pošle iniciátorovi konverzace, je *bob_public* o velikosti 2048 bajtů. Pomocí této metody se rovnou vygeneruje i sdílený klíč, *shared_secret*.

Jakmile strana, která počínala s komunikací, obdrží veřejný klíč *bob_public*, provede metodu `shareda()`, kde na vstupu přivede svůj předešle vytvořený privátní klíč *alice_private* a přijatý veřejný klíč *bob_public*. Jejím výstupem je sdílený klíč *shared_secret*, který by měl být shodný se sdíleným klíčem vytvořeným na druhém koncovém zařízení. Touto metodou končí ustanovení šifrovacího klíče, který je následně využit na obou stranách pro vstup algoritmu šifrování, resp. dešifrování.

4.4 Popis realizace návrhu zabezpečení

V následující části bakalářské práce bude popsána realizace návrhu zabezpečení dle kapitoly 3.4. Jelikož se vyvíjely implementace pro různá zařízení, popíšeme si také v této kapitole jejich odlišnosti.

Na straně Arduina je potřeba více dbát na nároky implementace na využívánou operační paměť zařízení, jelikož máme k dispozici pouze 8 KB SRAM. Co se týče implementace na Linuxu, tam tolik omezeni nejsme, ale potřebujeme zajistit, aby byly obě dvě strany synchronizované a při přijetí určité zprávy serverem, v případě ustanovení klíče, aby patřičně zareagoval a zpět poslal jeho veřejný klíč na NB-IoT modul.

Co se týče rozdílností zdrojových kódů na obou stranách, není zde až tak markantní rozdíl kvůli tomu, že implementuje stejné knihovny kvůli kompatibilitě. Jediné, co se liší, je implementace soketu na straně serveru a zacházení s NB-IoT modulem v případě Arduina. V obou případech byly vyvinuty tři skripty, resp. skicy, kdy:

- první verze je kompletně bez šifrování, jen poslání 50 bajtů dat přes NB-IoT síť – můžeme vidět i v kapitole 3.3,

- druhá verze je pouze jedna výměna klíčů a tímto klíčem šifrujeme všechny následné zprávy a
- třetí verze je s ustanovení klíče před každým šifrování zprávy.

Tyto verze se od sebe liší pouze interní logikou o to, jak se využívají deklarované funkce a metody. První a třetí verze jsou používány pro měření a zhodnocení efektivnosti realizace návrhu koncového zabezpečení.

4.4.1 Implementace Arduino a NB-IoT strany

Při inicializaci zařízení Arduino jsou tedy nastaveny některé proměnné a také inicializované seriové porty Arduina, kde `Serial3` je sériové rozhraní mezi shieldem NB-IoT a vývojovou desou a `Serial` značí sériovou linku, pomocí které komunikujeme z počítače do Arduina.

V prvé řadě je za potřebí zajistit nastavení NB-IoT modulu a jeho připojení do sítě operátora. K tomuto se používají AT příkazy popsané v kapitole 3.3, i ve stejném pořadí, jak jsou zde vypsány. Vše je ale nyní zajištěno tak, aby to probíhalo automaticky. Ve zdrojovém kódu Arduina můžeme tedy vidět řadu přepínačů, které se spouští v závislosti na přepínaných proměnných. Jelikož kód běží na Arduinu stále dokola (hlavní kód je totiž napsaný v metodě `loop()`), musíme zajistit, aby při každé smyčce přešel kód do správné větvě přepínače, což zajistíme kondicionály před celým kódem přepínače.

Pro každou vykonávanou funkci, ať už je to nastavení modulu NB-IoT, výměna klíčů s koncovým zařízením, se kterým komunikujeme nebo zašifrováním dat a jejich následným posláním, jsou kontrolovány určité proměnné, které zajistí, aby se při každé smyčce vykonala správná funkce. Těmito proměnnými jsou *moduleSet*, *keySent*, *keyShared* a *readyToCipher*, které jsou datového typu *bool*.

Poté je zde zmíněný přepínač, který slouží pro nastavení modulu NB-IoT a připraví jej na posílaní dat na server. Tento přepínač je ovládán proměnnou *saraFunction*. Přepínač poté funguje tak, že pokud je proměnná *moduleSet* nastavena na ***false***, přepíná se číselná proměnná *saraFunction* a provádí se následující funkce:

- `restartSARA()` – restartuje modul SARA,
- `modFunkcionality()` – přepne mód funkcionality na 1,
- `registraceUOperatora()` – registruje zařízení v síti operátora,
- `pdpKontext()` – přiřadí PDP kontext,
- `vytvoreniSocketu()` – vytvoří socket pro odesílání dat,
- `vypisIPAdresu()` – funkce sloužící pro kontrolu přiřazené IP adresy uživatelem ve výpisu konzole.

Mezi vykonáním všech kroků je zde určitá časová mezera, aby se stihly všechny metody provést v pořádku na modulu v určitém časovém intervalu. Například při

metodě `registraceUOperatora()` musíme čekat, než se modul spojí se sítí operátora pomocí RRC protokolu (popsaném v kapitole 2.5.2) a než je mu přidělena IP adresa, pět sekund.

S verzí skicy, u které se vyměňují klíče pomocí protokolu NewHope, jsou v posledním kroku tohoto přepínače přepnuty proměnné *moduleSet* na ***true*** a *keySent* na ***false***. Po úspěšném vykonání celého přepínače sloužícího k nastavení modulu SARA NB-IoT je pro komunikaci se serverem vyžadováno ustanovení šifrovacího klíče. K tomuto účelu slouží proměnná *keySent*, která pokud je nastavena na ***false***, se při další iteraci celého programu vykonají následující metody:

- `generaceKliceA()` – vygeneruje pár veřejného a soukromého klíče Alice,
- `odesliKlicAlice()` – odešle veřejný klíč Alice na server pro zpracování,
- `ctiVerejnyKlicB()` – přijme a zpracuje veřejný klíč Boba,
- `generaceSdileneho()` – vygeneruje sdílený klíč.

Metoda `odesliKlicAlice()` je unikátní v tom způsobu, že zde jsme poprvé narazili na nevýhodu v omezených vlastností první verze modulu SARA NB-IoT. Na uplinku (tudíž komunikace směrem z NB-IoT modulu do sítě operátora) jsme totiž omezeni jen 512 bajty, které můžeme poslat. Na downlinku (komunikace směrem ze sítě operátora do NB-IoT modulu) jsme také omezeni, ale již o něco méně, a to 1024 bajty možnými k přijetí zpráv.

Kvůli těmto omezení jsme byli nuceni veřejný klíč alice *alice_public* rozdělit na čtyři části po 456 bajtech a poslat každou část klíče zvlášť, o což se tedy stará metoda `odesliKlicAlice()`. Poté, co se úspěšně odešle veřejný klíč Alice, zařízení čeká na odpověď ze strany serveru – na veřejný klíč Boba. Tuto odpověď značí indikátor `+NSONMI:<číslo_socketu>, <délka_zprávy>`, který NB-IoT modul přijme ze sítě operátora.

Jakmile je tedy na straně serveru vygenerován veřejný klíč Boba, je rozdělen na dvě poloviny po 1024 bajtech a po částech poslán na modul NB-IoT. Kvůli dalšímu omezení, a to je omezení z pohledu operační paměti vývojové desky Arduino, musí být tento veřejný klíč Boba zpracováván po částech o velikosti 256 bajtů. Jakmile je tedy ze sítě přijat indikátor `+NSONMI`, je provedena metoda `ctiVerejnyKlicB()` a do modulu vyslán AT příkaz pro stáhnutí dat ze sítě operátora – `AT+NSORF=0,256`, kde první číslo značí číslo soketu na kterém byla data přijata a druhé číslo značí jakou délku dat chceme přijat. Tento příkaz je zopakován čtyřikrát a tímto je první polovina veřejného klíče Boba zpracována. Po pěti sekundách se ze serveru pošle druhá část Bobova veřejného klíče, který se na NB-IoT modulu zpracuje stejným způsobem, jako první část. V neposlední řadě se k ustanovení klíčů mezi oběma stranami provede metoda `generaceSdileneho()`, která vypočítá 32 bajtový klíč, který je uložen do proměnné *key*. Posledním krokem je přepnutí proměnných *keyShared*, *keySent* a *readyToCipher* na hodnotu ***true***.

Jakmile máme hodnoty proměnných *keyShared*, *moduleSet* a *readyToCipher* na hodnotě ***true***, můžeme začít se šifrováním. Celý proces šifrování byl popsán v předchozí kapitole 4.3.1. Naše implementace generuje náhodný plaintext o délce padesáti bajtů, který se následně šifruje s autentizačními daty, které byly nastaveny na textový řetězec *„SARA NB-IoT VUT“*. Posledním krokem celého návrhu na straně Arduina a NB-IoT modulu je šifra, která se následně odešle na server, společně s inicializačním vektorem a vypočítaným autentizačním tagem, pomocí AT příkazu `AT+NSOSTF`.

4.4.2 Implementace na straně serveru

Jak již bylo zmíněno, na straně Linuxového serveru nejsme tak omezeni výpočetním výkonem jako u omezeného zařízení Arduino, ale i přesto musíme využít knihovny Arduino Cryptography Library, uvedené v kapitole 4.3.1, kvůli kompatibilitě výpočtů dat na zařízeních. Zde je naštěstí nastavení připojení do sítě operátora již vyřešeno a nachystáno. Jediné, co na nás zbývá, je otevřít UDP soket pro naslouchání na předem stanoveném portu.

Jelikož zde skript po spuštění běží jen jednou, museli jsme část zdrojového kódu vložit do nekonečné `while` smyčky, která bude pracovat zároveň s implementací na Arduinu. Zdrojový kód na Linuxu je poté velmi jednoduchý a nijak výrazně se od realizace na Arduinu neliší:

- `prijmiPublicKeyAlice()` – přijme a zpracuje všechny čtyři části veřejného klíče Alice,
- `generaceKliceB()` – vygeneruje a odešle veřejný klíč Boba a zároveň také vygeneruje sdílený klíč,
- `prijmiData()` – přijme a zpracuje jakákoliv přijatá data na soketu,
- `desifrujData()` – dešifruje přijatá data.

Jakmile skript spustíme, nejdříve se otevře soket na IP adrese 0.0.0.0, což znamená, že naslouchá na všech svých síťových rozhraních a portu 4448. Poté kód přejde do vytvořené nekonečné smyčky a dokola provádí funkce uvedené výše. Stejně jako na Arduino jsme ale zvolili, aby se jen při určitém nastavení proměnných datového typu *bool* provedly patřičné metody. Příkladem jsou metody `prijmiPublicKeyAlice()` a `generaceKliceB()`, což jsou metody, které se vykonávají pouze pokud je proměnná *keyShared* nastavena na hodnotu ***false***.

Poté, co se otevře UDP soket, server čeká na příjem částí veřejného klíče Alice, které zpracuje a následně vygeneruje svůj veřejný klíč Boba, který pošle ihned sítí zpět na modul NB-IoT, jak již bylo naznačeno v kapitole 4.4.1. Po úspěšné generaci sdíleného klíče se nastaví proměnné *keyShared* a *readyToCipher* na hodnotu ***true***.

Dále již jen server čeká na přijetí vygenerovaných a zašifrovaných dat Arduinem,

které pomocí funkce `prijmiData()` zpracuje a dešifruje metodou `desifrujData()`. Výsledek procesu dešifrování je vypsán do konzole, kde je uživatel informován, zdali bylo s daty manipulováno při přenosu, či ne.

4.5 Shrnutí realizace návrhu zabezpečení

Návrh byl zhotoven tak, aby uživateli stačilo jen spustit skript na serveru a dále zapojil koncové zařízení Arduino s modulem NB-IoT ke zdroji napájení a žádný jiný vstup uživatele nebyl za potřebí. Každopádně návrh není připravený pro žádný případ použití v reálném světě, ale jen poukazuje na skutečnost, že je teoreticky i prakticky možné implementovat určitou úroveň zabezpečení i na výpočetně omezená zařízení jako je Arduino a nízko-odběrový modul SARA NB-IoT.

Implementace má svoje určitá pozitiva, ale i negativa. Pozitivní na implementaci je ukázka, že je možné implementovat nejmodernější kryptografické algoritmy i na tak výpočetně slabá zařízení. Uživateli tedy stačí například jen připojit příslušný senzor k desce Arduino, upravit kód tak, aby využíval data ze senzoru pro šifrování a má teoreticky zajištěné, že jeho data budou v bezpečí.

Negativa tohoto návrhu jsou ta, že koncová zařízení neví, s kým si reálně stanovila šifrovací klíč. Pokud by tedy na síti naslouchal útočník, může se vydávat za Alici nebo Boba a stanovit si tak klíč s oběma stranami komunikace, aniž by se něco dozvěděli. Tento problém by se dal vyřešit implementací digitálního podpisu, kdy bychom s každou odeslanou zprávou poslali i digitální podpis ze zprávy, který by si strany komunikace ověřovaly, ale bylo by za potřebí přivést do řešení asymetrickou kryptografii.

Dalším negativem návrhu je v čekání obou stran na příslušný klíč v průběhu ustanovení šifrovacího klíče. Nejen, že nevíme z jakého zařízení tato zpráva přišla, takže by se tento návrh nedal nasadit v reálné aplikaci, kdy se serverem komunikuje několik senzorů, ale je zde i jistá zranitelnost při útoku odepření služby. Pokud totiž pošleme jakákoliv data na server nebo modul SARA v době, kdy čekají na veřejné klíče, tak se tato data sice zpracují, ale nevznikne na obou stranách stejný šifrovací klíč nebo je také možnost, že zařízení již nebudou synchronizována a budou se provádět úplně jiné metody na obou stranách, čehož výsledkem by bylo zahlcení paměti Arduina a jeho následné selhání. Tato skutečnost by se dala vyřešit identifikací každé posílané zprávy jednoduchým tagem, znázorňující co se ve zprávě nachází za data.

5 Měření implementace zabezpečení

V rámci praktické části bakalářské práce je provedeno měření implementace návrhu zabezpečení. Jsou zde popsány postupy měření jednotlivých verzí skic představených v kapitole 4.4. Měření vždy probíhalo na straně omezeného zařízení Arduino z důvodu zjištění nákladnosti a změny vlastností LPWAN zařízení před a po implementaci návrhu zabezpečení.

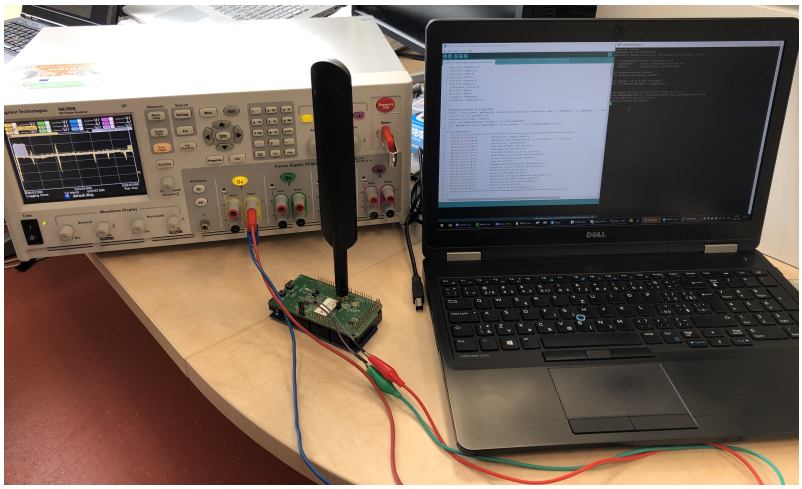
Při měření byl kladen důraz na náročnost návrhu z pohledu spotřebovávané energie, využité paměti a v neposlední řadě i časové náročnosti.

5.1 Časová měření a měření spotřeby implementace

V této části bylo za účel změřit spotřebovávanou energii při nasazení návrhu implementace zabezpečení, změřit energetickou spotřebu jednotlivých funkcí zajištěných v návrhu a poté tyto data porovnat s nezabezpečenou komunikací NB-IoT.

Měření bylo prováděno pomocí stejnosměrného analyzátoru elektrické energie N6705B značky Agilent Technologies (nyní Keysight Technologies), který napájel zařízení Arduino společně s modulem SARA NB-IoT a analyzoval proudový odběr tohoto zařízení. Skripty na serveru byly spouštěny uživatelem pomocí notebooku, který byl připojený k serveru protokolem SSH. Notebook také sloužil pro nahrávání měřených verzí skic na zařízení Arduino.

Na analyzátoru byly nastaveny následující hodnoty pro měření – 5 V zdrojové napětí pro desku Arduino a maximální hodnota odebíraného proudu 1 A. Obnovovací frekvence měření proudové spotřeby byla nastavena na 1 ms. Zapojení laboratoře můžeme vidět na obrázku 5.1.



Obr. 5.1: Zapojení laboratoře měření.

Jak již bylo zmíněno v kapitole 4.4, měřily se dvě vyvíjené verze komunikace NB-IoT – nezabezpečená jako referenční a poté s implementací našeho návrhu s tím, že proběhlo ustanovení klíčů před každou poslanou zprávou. Samotné měření trvalo vždy deset minut, kde všechny měření mají společnou jednu funkci, a to nastavení modulu SARA NB-IoT, popsaném v kapitole 4.4.1. Při měření a celkovém odhadu spotřeby zařízení ale nebude toto nastavení zahrnuto, jelikož při reálné aplikaci je modul SARA nakonfigurovaný těmito hodnotami již při rozmístění zařízení.

Grafy 5.2 a 5.3 nám ukazují detailní proudové průběhy jednotlivých funkcí vykonávaných modulem SARA NB-IoT, i vývojovou deskou Arduino. Oba dva detaily jsou výřezy z celkového měření a znázorňují hlavní funkce skicy. Přesněji, výřez 5.2 ukazuje detaily průběhu ustanovení klíče, následné šifrování a odeslání zašifrované zprávy. Druhý graf, 5.3, nám pak pro představu umožňuje vidět proudový průběh konfiguraci komunikačního modulu.

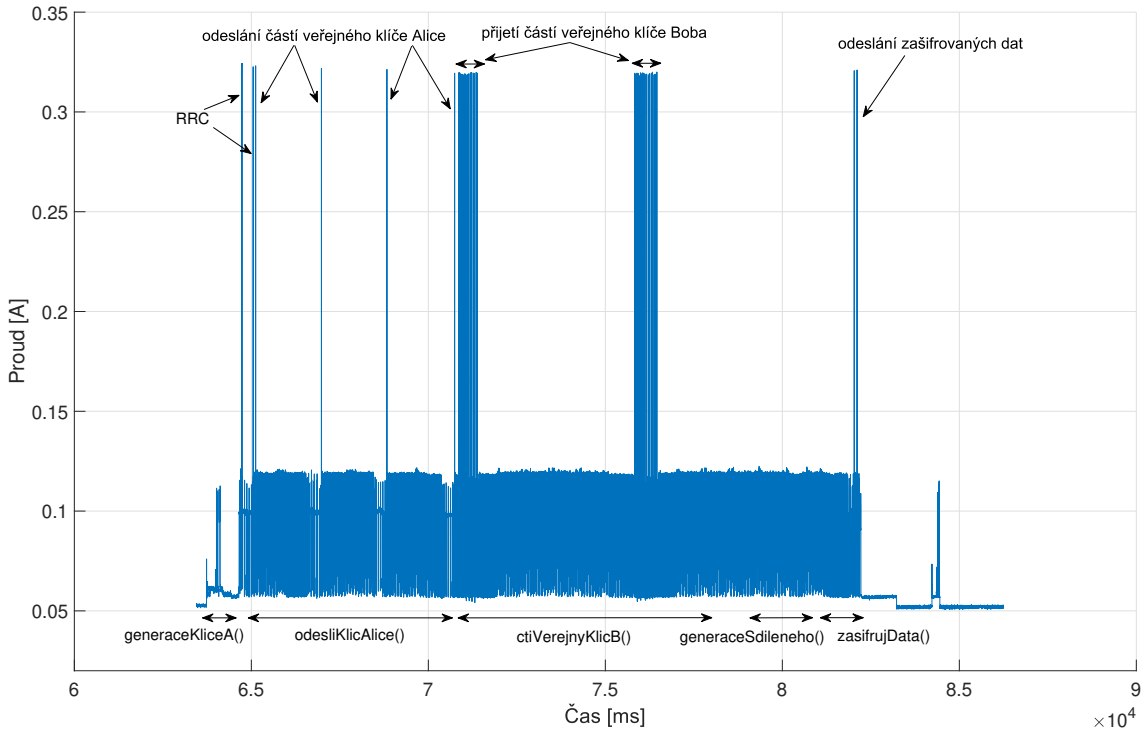
Jelikož je po každém úspěšném odeslání zprávy s uživatelskými daty komunikační modul uveden do režimu spánku, tak pokud chce znovu komunikovat se sítí operátora (tedy tehdy, když chce odeslat či přijmout zprávu), musí se pomocí RRC zpráv znovu spojit se sítí (RRC protokol v kapitole 2.5.2). Tyto RRC zprávy můžeme vidět na obou grafech, kdy v grafu 5.2 jsou zprávy posílány před započetím odesílání veřejného klíče Alice a v případě grafu 5.3 jde o celkovou registraci komunikačního modulu v síti.

Tabulka 5.1 ukazuje detailní rozbor měření času a také průměrného spotřebovávaného proudu za čas vykonávané funkce. Hodnoty v této tabulce vyplývají z měření celkové proudové spotřeby, kdy se vždy udělal průměr proudového odběru za časový úsek, ve kterém probíhal výpočet dané funkce, viditelné v grafu 5.2. Následovně můžeme v tabulce vidět i spotřebu v jednotce *mAh*, která se vypočítala dle vzorce 5.1

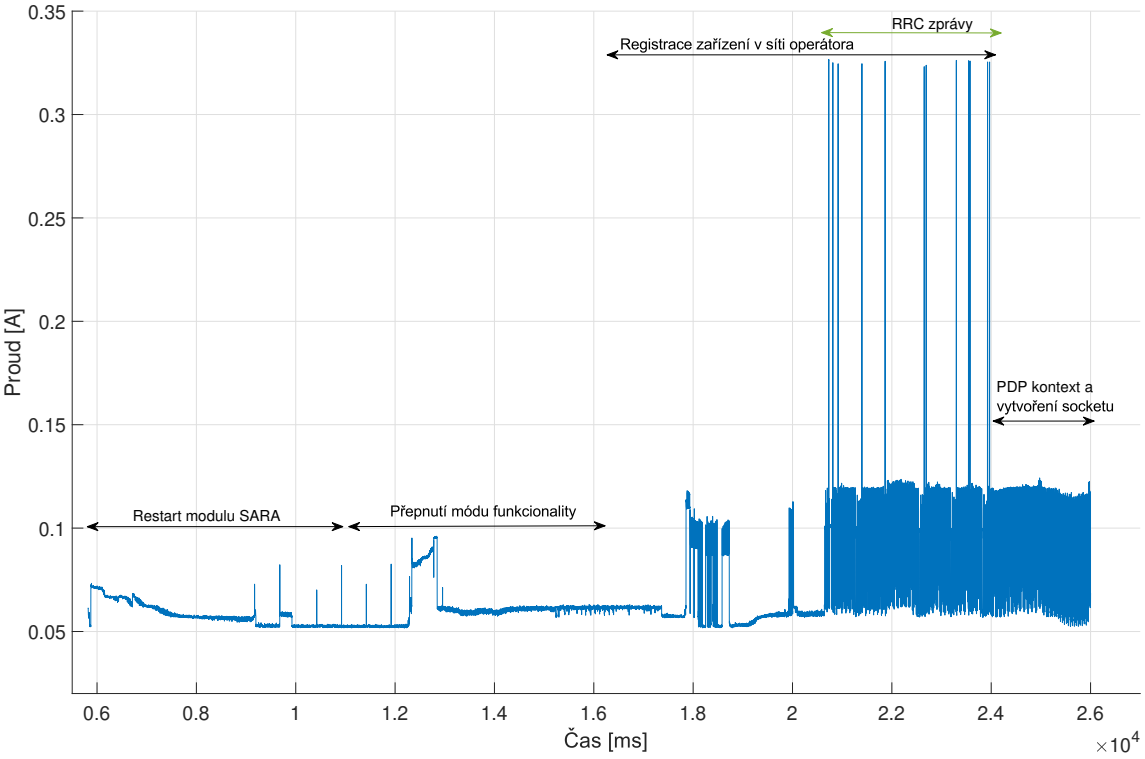
$$(t * 2{,}7778 * 10^{-}7) * I, \tag{5.1}$$

kde t je vždy čas, za který byla funkce vykonána v jednotkách *ms* (sloupec „Čas [ms]“ z tabulky 5.1), $2{,}7778 * 10^{-}7$ je konstanta pro převod času t na hodiny a I je proudový odběr zařízení po dobu vykonávané funkce v jednotkách *mA* (sloupec „Proud [mA]“ z tabulky 5.1).

Bylo zjištěno, že je zařízení při nasazení zabezpečené verze až 74 krát delší dobu aktivní při zaslání jediné zprávy. Zatímco při zaslání nezabezpečené zprávy je nutné zařízení udržet aktivní po dobu 137 ms, při zabezpečené verzi musí být zařízení dostupné po dobu něco málo přes deset sekund, na čemž se také projeví spotřeba celého zařízení a jeho schopnost být dlouhou dobu napájené pouze baterií.



Obr. 5.2: Detail proudového průběhu ustanovení klíče a šifrování.



Obr. 5.3: Detail proudového odběru při nastavení modulu SARA NB-IoT.

Tab. 5.1: Detailní rozbor časového měření a proudového odběru zařízení.

	Celkové hodnoty			Hodnoty na 1 zprávu		
Vykonávaná funkce	Čas [ms]	Proud [mA]	Spotřeba [mAh]	Čas [ms]	Proud [mA]	Spotřeba [mAh]
Generace páru klíčů Alice	640,333	62,325	0,01109	–	–	–
Odeslání veřejného klíče Alice	3800,313	97	0,10240	950,078	97,133	0,02563
Přijetí veřejného klíče Boba	4946,688	101,833	0,13993	618,336	107,9	0,01853
Generace sdíleného klíče	477,188	97,3	0,01290	–	–	–
Zašifrování a poslání	224,875	95,2	0,00595	144,438	97,8	0,003924
Celkem – zabezpečeno	10089,396	90,92	0,25481	–	–	–
Nezabezpečeno - odeslání	–	–	–	136,895	53,3	0,002027

5.2 Využití operační paměti Arduina

Jak jsme již řekli, podíváme se i na náročnost implementace zabezpečení komunikace z pohledu využívané operační paměti obou využívaných algoritmů. Jelikož jsme na Arduinu omezeni 8 KB operační paměti SRAM, musíme při implementaci takových algoritmů být opatrní s prací a využívání jednotlivých proměnných.

Co je na platformě Arduino nejdůležitější, je ukládat si statické textové řetězce do paměti flash pomocí integrované funkce `F()`, která nám toto při kompilaci skicy zajistí. Jejím jediným parametrem je textový řetězec napsaný mezi úvozovky. Tato funkce se obzvlášť hodí, pokud potřebujeme informační výpisy do konzole z důvodu odstraňování chyb ve skici, jelikož Arduino IDE nemá nástroj pro ladění programu.

Pokud se podíváme do dokumentace knihovny Arduino Cryptography Library [16], zjistíme, že při běhu skicy je za potřebí mít alespoň 2080 byte volného místa pro uložení vygenerovaných veřejných klíčů a při samotném průběhu algoritmu NewHope alespoň dalších 4500 bajtů dočasného místa pro práci s potřebnými lokálními proměnnými. Pro ušetření místa se ale dá deklarovat jediný globální buffer pro uložení veřejného klíče Alice a posléze i veřejného klíče Boba, kterým přepíšeme již nepotřebný veřejný klíč Alice při běhu algoritmu. Musíme tedy počítat s celkovými nároky jen algoritmu NewHope na zhruba 6500 bajtů z celkových 8 KB operační paměti.

S algoritmem AES256-GCM to již není takový problém, ale stále i při samotné deklaraci nám zabere zhruba dalších 350 bajtů místa. Proto jej definujeme jen za pomocí lokální proměnné až jej budeme potřebovat, v tomto případě až poté, co se ustanoví sdílený šifrovací klíč, ve funkci `zasifrujData()`.

Na interní logiku celého návrhu jak vše bude pracovat dohromady nám tedy zbylo teoreticky 2500 KB operační paměti k dispozici. Pro ulehčení logiky některých funkcí bylo nutné stále nutné deklarovat některé globální proměnné jako jsou čítače, proměnné datových typů *bool* pro operaci s přepínači, proměnné do kterých si budeme ukládat například šifrovací klíč, inicializační vektor nebo také výstup z algoritmu AES256-GCM. Tyto proměnné zabírají zhruba dalších 200 bajtů.

Jak již bylo zmíněno v kapitole 4.4.1, ze strany komunikačního modulu SARA NB-IoT jsme byli omezeni při odeslání či přijetí veřejných klíčů obou stran, a proto jsme byli nuceni rozdělit tyto veřejné klíče na menší části, abychom s nimi mohli pracovat. I přesto ale pro zpracování těchto klíčů pro odeslání či přijetí potřebujeme celkově ve funkcích k těmto úkonům alokovat dohromady dalších 1000 bajtů lokálních proměnných, a to přesněji ve funkcích `odesliKlicAlice()`, `ctivVerejnyKlicB()` a `odesliData()`.

Při kompilaci skicy nám také pomáhá integrovaný nástroj `avrdude`, zmíněný v kapitole 4.2.1, kdy do konzole vypíše kolik paměti zabírá celá skica. Výpis tohoto

nástroje můžeme vidět ve výpise 5.1. Celkově poté realizace zabezpečení v čase kompilace zabírá 40% (3358 bajtů) celkové operační paměti a zbylých 60% (4834 bajtů) nám poté zbývá pro deklaraci dočasných, respektive lokálních, proměnných.

Výpis 5.1: Výpis využívané paměti nástrojem `avrdude`

```
Sketch uses 38172 bytes (15%) of program storage space.
Maximum is 253952 bytes.
Global variables use 3358 bytes (40%) of dynamic memory,
leaving 4834 bytes for local variables.
Maximum is 8192 bytes.
```

5.3 Shrnutí měření implementace

Předešlé kapitoly se věnovaly energetickým měřením, tak i měřením časové a paměťové náročnosti celé realizace návrhu na zabezpečení komunikace NB-IoT. Z výsledků měření jsme mohli zjistit, že je zde velký prostor pro optimalizaci samotné implementace.

Při měření spotřeby a samotném porovnání odebíraného proudu zařízení Arduino společně s modulem NB-IoT jsme naměřili hodnotu prakticky o dva řády vyšší při zabezpečené komunikaci oproti nezabezpečené. Tato skutečnost je totiž také dána tím, že zařízení musí být pro správné vykonání všech funkcí pro zabezpečení jedné zprávy být až 74 krát déle aktivní, než při pouhém vyslání jednné nezabezpečené zprávy. Tímto můžeme říct, že je implementace, co se týče energetické spotřeby, velmi neefektivní na úkor zabezpečení uživatelských dat.

Ze stránky paměťové náročnosti vybraných algoritmů jsme několikrát naráželi na omezení výpočetního výkonu omezeného zařízení. Museli jsme proto vytvářet opatření, abychom zprovoznili návrh – například rozdělení veřejného klíče Boba na čtyři části, jinak by zařízení nefungovalo správně. Kvůli této skutečnosti se také prodloužila aktivní doba zařízení, což mělo dopad i na celkovou spotřebu zařízení.

Pro optimalizaci implementace bychom tedy mohli využít i dostupné paměti EEPROM, ale s tím, že bychom byli omezeni zapisovacími cykly této paměti (uvedené v kapitole 4.1). Je zde také možnost rozšíření operační paměti zařízení Arduino o rozšiřující modul dostupný pro takové zařízení. Tímto by se ale navýšila cena koncového zařízení. Třetí možností by byl návrh zařízení, které by bylo uzpůsobené pro potřeby implementace zabezpečení, čímž bychom mohli, podle potřeby, regulovat výpočetní výkon zařízení dle uvážení a tímto také omezit celkovou energetickou náročnost.

6 Závěr

Cílem bakalářské práce bylo provést analýzu komunikace a funkce nových nízkoodběrových technologií NB-IoT a na základě těchto informací navrhnout zabezpečení komunikace na aplikační vrstvě ISO/OSI modelu s důrazem na bezpečnost, co nejvěrnější zachování vlastností využívaných zařízení a efektivitu ve využití v reálném nasazení. Nezbytnou částí práce je i realizace tohoto návrhu na výpočetně omezeném zařízení.

Teoretická část popisuje rozdělení LPWAN technologií s nejběžněji využívanými telekomunikačními technologiemi. Dále se zaměřuje na komunikační technologii NB-IoT a zabývá se rozborem vlastností s popisem využívaných protokolů. Posledním probíraným tématem teoretické části je rozbor bezpečnostních hrozeb, které tuto technologii mohou činit rizikovou. Celá bezpečnost uživatelských dat stojí na typu zabezpečení poskytované službami operátora. V současné době jsou uživatelská data zabezpečena pouze v rámci sítě operátora, přičemž po jejím opuštění jsou dále posílána v otevřeném formátu přes nezabezpečenou síť Internet až ke koncové aplikaci uživatele.

Praktická část se zaobírá výběrem kryptografických algoritmů, které by bylo vhodné implementovat, společně se zařízením, které bude dostačovat výpočetním výkonem k provádění kryptografických výpočtů těchto algoritmů. Byla zprovozněna komunikace NB-IoT modulu s VUT serverem, ze které bylo zjištěno, že data nejsou žádným způsobem chráněna a případný útočník má přístup k přenášeným zprávám. Cílem návrhu bylo zajistit úroveň bezpečnosti, která bude dostačující i v budoucnu, ve smyslu kvantových počítačů. Oba vybrané algoritmy, jak NewHope, tak AES256 v operačním módu GCM, poskytují úroveň post-kvantové bezpečnosti 128 bitů. Protokol NewHope nám umožňuje výměnu sdíleného klíče přes nezabezpečené médium a následně AES256-GCM zajišťující důvěrnost a integritu přenášených dat.

Výstupem bakalářské práce je realizace návrhu zabezpečení na výpočetně omezeném zařízení Arduino, která byla následně zhodnocena měřeními. Výsledkem bylo tedy navázání zabezpečené úzkopásmové komunikace mezi komunikačním modulem a koncovou aplikací. Z měření však vyplývá, že je zde nutná optimalizace návrhu i samotné realizace. Bylo zjištěno, že z důvodu velmi dlouhého času, po který bývá zařízení aktivní, není energeticky efektivní. Na vzdory energetické neefektivitě je návrh vhodný pro použití v situacích, kdy by po zařízení bylo požadováno jen několik jednotek zpráv denně, aby zachovalo svoji vlastnost dlouhé výdrže baterie.

Literatura

[1] CHEN, Min, Yiming MIAO, Yixue HAO a Kai HWANG. *Narrow Band Internet of Things*. IEEE Access [online]. 2017, 5, 20557-20577 [cit. 27. 9. 2018]. DOI: 10.1109/ACCESS.2017.2751586. ISSN 2169-3536. Dostupné z: <`https://ieeexplore.ieee.org/document/8038776`>.

[2] MEKKI, Kais, Eddy BAJIC, Frederic CHAXEL a Fernand MEYER. *A comparative study of LPWAN technologies for large-scale IoT deployment*. ICT Express [online]. 2018 [cit. 28. 10. 2018]. DOI: 10.1016/j.icte.2017.12.005. ISSN 24059595. Dostupné z: <`https://linkinghub.elsevier.com/retrieve/pii/S2405959517302953`>

[3] S. Farrell *LoRaWAN Overview* Online: `https://tools.ietf.org/html/draft-farrell-lpwan-lora-overview-01` publikováno 26. 10. 2016, citováno 25. 9. 2018.

[4] JC. Zuniga *SIGFOX System Description* Online: `https://tools.ietf.org/html/draft-zuniga-lpwan-sigfox-system-description-04` publikováno 4. 12. 2017, citováno 26. 9. 2018.

[5] S. Farrell *LPWAN Overview* Online: `https://tools.ietf.org/id/draft-ietf-lpwan-overview-07.html` publikováno 3. 10. 2017, citováno 25. 9. 2018.

[6] Da Yu, Wushao Wen *Non-access-stratum request attack in E-UTRAN* Online: `https://ieeexplore.ieee.org/document/6154001` publikováno 21. 2. 2012, citováno 27. 10. 2018.

[7] Manoj Pradhan, Nex-G *Narrow Band Internet of Things (NB IoT)* Online: `https://www.slideshare.net/Manji620/nb-iot-presentation` publikováno 1. 9. 2017, citováno 27. 10. 2018.

[8] SeungJune Yi, SungDuck Chun, YoungDae Lee, SungJun Park, SungHoon Jung. Packet Data Convergence Protocol (PDCP) in *Radio Protocols for LTE and LTE-Advanced*, John Wiley & Sons, 2012. ISBN 978-1-118-18856-9.

[9] Mohamed Amine Ferrag, Leandros Maglaras, Antonios Argyriou, Dimitrios Kosmanos, Helge Janicke *Security for 4G and 5G Cellular Networks: A Survey of Existing Authentication and Privacy-preserving Schemes* Online: `https://arxiv.org/abs/1708.04027v1` publikováno 14. 8. 2017, citováno 11. 11. 2018.

[10] Masanobu Katagi, Shiho Moriai *Lightweight Cryptography for the Internet of Things* Online: https://iab.org/wp-content/IAB-uploads/2011/03/Kaftan.pdf publikováno květen 2012, citováno 1. 10. 2018.

[11] NAGAICH, Shweta a Y.C. GOSWAMI. *Shor's Algorithm for Quantum Numbers Using MATLAB Simulator*. In: 2015 Fifth International Conference on Advanced Computing & Communication Technologies [online]. IEEE, 2015, 2015, s. 165-168 [cit. 18. 11. 2018]. DOI: 10.1109/ACCT.2015.16. ISBN 978-1-4799-8488-6. Dostupné z: <https://ieeexplore.ieee.org/document/7079073>.

[12] NEJATOLLAHI, Hamid, Nikil DUTT a Rosario CAMMAROTA. *Trends, challenges and needs for lattice-based cryptography implementations*. In: Proceedings of the Twelfth IEEE/ACM/IFIP International Conference on Hardware/-Software Codesign and System Synthesis Companion - CODES '17 [online]. New York, New York, USA: ACM Press, 2017, 2017, s. 1-3 [cit. 18. 11. 2018]. DOI: 10.1145/3125502.3125559. ISBN 9781450351850. Dostupné z: <http://dl.acm.org/citation.cfm?doid=3125502.3125559>

[13] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, Peter Schwabe *Post-quantum key exchange - a new hope* [online]. Dostupné z URL: <https://eprint.iacr.org/2015/1092>, publikováno 10. 11. 2015, citováno 12. 11. 2018.

[14] Erdem Alkim, Philipp Jakubeit, Peter Schwabe *A new hope on ARM Cortex-M* [online]. Dostupné z URL: <https://eprint.iacr.org/2016/758>, publikováno 5. 8. 2016, citováno 12. 11. 2018.

[15] Morris Dworkin (NIST). *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC* [online]. NIST, 2007, [cit. 2. 12. 2018]. DOI: SP 800-38D. Dostupné z: <https://csrc.nist.gov/publications/detail/sp/800-38d/final>

[16] R. WEATHERLEY. *Arduino Cryptography Library* [online]. [cit. 11. 5. 2019]. Dostupné z: <https://rweather.github.io/arduinolibs/crypto.html>

Seznam symbolů, veličin a zkratek

LPWAN	Low Powered Wide Area Network
NB-IoT	Narrowband – Internet of Things
IoT	Internet of Things – Internet Věcí
GSM	Groupe Spécial Mobile
LTE	Long Term Evolution
UMTS	Universal Mobile Telecommunication System
CCTV	Closed-Circuit Television
LoRaWan	Long Range Wide Area Network
ISM	Industrial, Scientific and Medical
MAC	Media Access Control
FDD	Frekvenčně Dělený Duplex
eDRX	Extended Discontinuous Reception – Rozšířené nespojité vysílání
PSM	Power Saving Mode – Úsporný režim
PDCP	Packet Data Convergence Protocol
QPSK	Quadruple Phase Shift Keying – Čtyřstavové klíčování fázovým zdvihem
OFDM	Orthogonal Frequency Division Multiplexing – Ortogonální multiplex s frekvenčním dělením
BPSK	Binary-Phase Shift Keying – Binární klíčování fázovým zdvihem
SC-FDMA	Single-Carrier Frequency Division Multiple Access
NPUSCH	Narrowband Physical Uplink Shared Channel
NPDSCH	Narrowband Physical Downlink Shared Channel
RLC	Radio Link Control – Ovládání radiového spojení
RRC	Radio Resource Control – Ovládání radiových zdrojů
NAS	Non Access Stratum
AS	Access Stratum
IP	Internet Protocol
UE	User Equipment – Koncové zařízení
SRB	Signaling Radio Bearer – Signálový radiový nosič
RoHC	Robust Header Compression
EEA	Evolved Packet System Encryption Algorithm – Šifrovací algoritmy pro vyvinutý paketový systém
SIM	Subscriber Identity Module – Účastnická identifikační karta
CTR	Counter Mode – Čítačový mód
SDU	Service Data Unit – Služební datová jednotka
DDoS	Distributed Denial of Service – Distribuované Odepření Služby
SSL	Socket Layer Security – Vrstva bezpečných soketů

MME	Mobile Management Entity – Entita spravující mobilní připojení
FBS	False Base Station – Falešná základnová stanice
AES	Advanced Encryption Standard
RFID	Radio Frequency Identificator – Identifikace na rádiové frekvenci
RSA	Rivest Shamir Adleman
DSA	Digital Signature Algorithm – Algoritmus pro digitální podpis
ECDH	Elliptic-Curve Diffie Hellman – Diffie Hellman založený na eliptických křivkách
SVP	Shortest Vector Problem – Problém nejkratšího vektoru
CVP	Closest Vector Problem – Problém nejbližšího vektoru
NTT	Number-Theoretic Transform
DES	Data Encryption Standard
ECB	Electronic Codebook – Režim kódové knihy
CBC	Cipher Block Chaining – Řetězení šifrových bloků
OFB	Output Feedback
CFB	Cipher Feedback – Šifrová zpětná vazba
GCM	Galois/Counter Mode
SRAM	Static Random Access Memory – Statická paměť
EEPROM	Electronically Erasable Programmable Read-Only Memory – Elektronicky vymazatelná paměť pouze pro čtení
USB	Universal Serial Bus – Univerzální seriová sběrnice
IDE	Integrated Development Environment – Vývojové prostředí

7 Obsah přiloženého CD

Přiložené CD s bakalářskou práci obsahuje:

- zdrojové kódy pro systém Linux,
  nezabezpečená verze ve složce *\LINUX_unsecured\source_code*,
  zabezpečená verze ve složce *\LINUX_secured\source_code*,
- zdrojové kódy pro vývojovou desku Arduino,
  nezabezpečená verze ve složce *\NBIOT_com_unsecured*,
  zabezpečená verze ve složce *\NBIOT_com_secured*,
- `.pdf` soubor s bakalářskou prací,
- `.xlsx` soubor s naměřenými hodnotami v tabulkách.

Přiložený soubor *README.txt* popisuje jak postupovat pro vyzkoušení samotné implementace. Soubory *compilation_command.txt* v příslušných složkách obsahují příkaz pro kompilaci zdrojového kódu v operačním systému Linux.

```
/ ..... kořenový adresář přiloženého CD
├── LINUX_secured
│   ├── compilation_command.txt
│   ├── automated-KEX_server-side_implementation_Kolaja
│   └── source_code
│       └── main_automated-kex.cpp.....zdrojový kód
├── LINUX_unsecured
│   ├── compilation_command.txt
│   ├── manual_server-side_implementation_Kolaja
│   └── source_code
│       └── manual_main.cpp.....zdrojový kód
├── NBIOT_com_secured
│   └── NBIOT_com_secured.ino.....zdrojový kód
├── NBIOT_com_unsecured
│   └── NBIOT_com_unsecured.ino.....zdrojový kód
├── Kolaja_David-bakalarska_prace.pdf
├── mereni_NewHope-AES-50B.xlsx
└── README.txt
```