FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Bezpečnostní analýza protokolu ZigBee Touchlink
Student:	Jakub Šatoplet
Vedoucí:	Ing. Jiří Dostál, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

ZigBee je jeden z nejrozšířenějších standardů pro komunikaci zařízení Internetu věcí (IoT) v tzv. chytrých domácnostech (smart home). V takové síti spolu komunikují zařízení od senzorů teploty, přes zámky dveří až po velké spotřebiče, např. lednice. S jejich snižující se cenou a dostupností o ně však roste i zájem útočníků. Proveďte analýzu tohoto protokolu se zaměřením na zabezpečení komunikace proti útokům, jako jsou například sniffing či replay útoky. Navrhněte a realizujte aplikaci pro příkazový řádek, která zjednoduší útoky na zařízení podporující připojení k síti pomocí ZigBee Touchlink. Rádiovou vrstvu implementujte pomocí technologie softwarově definovaného rádia (SDR). Výslednou aplikaci otestujte na reálné síti a zhodnoťte praktickou použitelnost.

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 2. ledna 2022 v Praze.

Bakalářská práce

BEZPEČNOSTNÍ ANALÝZA PROTOKOLU ZIGBEE TOUCHLINK

Jakub Šatoplet

Fakulta informačních technologií
Katedra informační bezpečnosti
Vedoucí: Ing. Jiří Dostál, Ph.D.
11. května 2022

Odkaz na tuto práci

Šatoplet Jakub. *Bezpečnostní analýza protokolu ZigBee Touchlink.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

V prvé řadě bych rád vyjádřil poděkování mému vedoucímu Ing. Jiřímu Dostálovi, Ph.D. za úvod do technologie softwarově definovaného rádia, věcné připomínky a nasměrování správným směrem ve chvílích, kdy to bylo potřeba. Zvláštní poděkování patří MUC. Kateřině Floriánové za výpomoc během testování výsledné aplikace. Nerad bych také opomenul a velice chtěl poděkovat mé rodině a blízkým za přečtení této práce, následnou pomoc s korekturou a převážně za trpělivost se mnou, která byla během tvorby potřeba.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít.

Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 11. května 2022 .....

Abstrakt

Obsahem bakalářské práce je teoretická analýza zabezpečení přenosu dat rádiovým protokolem ZigBee Touchlink a praktický vývoj aplikace využívající objevených zranitelností k útoku na tento protokol. Analytická část práce uvádí fungování rádiového přenosu normy IEEE 802.15.4, dále představuje strukturu dat přenášených pomocí ZigBee, dostupné prvky pro zajištění důvěrnosti a integrity přenosu a funkcionalitu Touchlink. Stěžejním výsledkem práce je aplikace pro příkazový řádek v jazyce Python, která využívá úniku statického šifrovacího klíče Touchlink pro dešifrování komunikace v ZigBee sítích. Aplikace poskytuje interaktivní i neinteraktivní režim, snadnou rozšiřitelnost pomocí knihovny Scapy a rádiovou vrstvu implementuje za využití technologie softwarově definovaného rádia (SDR). V závěru jsou shrnuty výsledky testování aplikace na reálné síti, které ukázaly vzdálenost možného odposlechu v nízkých desítkách metrů. Tyto výsledky činí praktickou použitelnost této aplikace omezenou.

Klíčová slova ZigBee, Touchlink, SDR, Python, Scapy, odposlech, IoT

Abstract

The content of the bachelor thesis is a theoretical analysis of the security of data transmission using the ZigBee Touchlink radio protocol and the practical development of an application exploiting discovered vulnerabilities to attack this protocol. The analytical part of the thesis introduces the operation of the radio transmission standard IEEE 802.15.4, then presents the structure of data transmitted using ZigBee, available elements to ensure confidentiality and integrity of the transmission and the functionality of Touchlink. The critical result of the thesis is a Python command-line application that uses the leakage of the Touchlink static encryption key to decrypt communications in ZigBee networks. The application provides interactive and non-interactive modes, easy extensibility using the Scapy library, and implements the radio layer using software-defined radio (SDR) technology. Finally, the results of testing the application on a real network are summarised, showing a possible eavesdropping distance in the low tens of meters. These results make the practical applicability of this application limited.

Keywords ZigBee, Touchlink, SDR, Python, Scapy, sniffing, IoT

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Seznam zkratek

ACK	acknowledgment
AEAD	authenticated encryption with associated data
AES	Advanced Encryption Standard
AOVD	ad-hoc on-demand distance vectoring
APDU	APS PDU
APS	application support sub-layer
ARPANET	Advanced Research Projects Agency Network
ASCII	American Standard Code for Information Interchange
AUX	auxiliary header
CBC-MAC	cipher block chaining message authentication code
CRC	cyclic redundancy check
CSMA/CA	carrier-sense multiple access with collision avoidance
CSV	comma-separated values
CTR	counter
DOS	denial-of-service
DSP	digital signal processing
DSSS	direct-sequence spread spectrum
ECB	electronic code book
ENC	encryption
EPID	extended PAN identifier
EUI-64	Extended Unique Identifier
FHSS	frequency-hopping spread spectrum
FPGA	field-programmable gate array
GSM	Groupe Spécial Mobile
GTS	guaranteed time slot
HA	Home Automation
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of things

ISM	industrial, scientific, medical
ISO	International Organization for Standardization
LAN	local area network
LLC	logical link control
LQI	link quality indicator
MAC	medium access control
MAN	metropolitan area network
MCPS	MAC common part sub-layer
MFR	MAC footer
MHR	MAC header
MIC	message integrity code
MLME	MAC layer management entity
MMO	Matyas-Meyer-Oseas
NPDU	NWK PDU
NWK	network layer
OSI	Open Systems Interconnection
OUI	organizationally unique identifier
P2P	peer-to-peer
PAN	personal area network
PDU	protocol data unit
PHR	PHY header
PHY	physical layer
PPDU	PHY PDU
PSDU	PHY service data unit
QR	quick response
RFC	Request for Comments
RID	response identifier
RX	radio receiver
SDR	software-defined radio
SHA	Secure Hash Algorithms
SHR	Synchronization header
SKKE	Symmetric Key Establishment Protocol
TC	trust center
TDM	time-division multiplexing
TID	transaction identifier
TX	radio transmitter
ZCL	ZigBee Cluster Library
ZDO	ZigBee device object
ZED	ZigBee end device

Kapitola 1

Úvod

Pravděpodobně jednou z nejvýznamnějších technologií 21. století je internet věcí (IoT). Internet věcí je nejčastěji popisován jako síť objektů, jež v sobě mají vestavěné senzory a technologie pro propojení a výměnu dat pomocí internetu či jiných síťových technologií. Součástí internetu věcí jsou každodenní objekty od senzorů tlaku v pneumatikách, přes domácí spotřebiče typu kávovar, ba dokonce i lidé nesoucí bionické protézy.

Přestože idea takovýchto systémů existovala dlouhou dobu, její praktické nasazení bylo umožněno až nedávnými technologickými pokroky. Mezi tyto pokroky patří všudypřítomná síťová konektivita, cloudové technologie umožňující správu a zpracování objemu dat generovaného takovýmto množstvím zařízení, a převážně rozvoj dostatečně přesných, malých a energeticky úsporných senzorů a mikrokontrolerů.

Tato technologie bezpochyby značně ulehčuje každodenní život každého z nás analýzou a využitím nasbíraných dat. Inteligentní budovy snižují energetickou náročnost optimalizací udržování vnitřní teploty a chytré domácnosti se umějí samy přizpůsobit našemu dennímu režimu. Nositelná zařízení, zejména ve formě hodinek či prstenů, sledují životní funkce a dokážou předpovědět blížící se infarkt. Průmyslová nasazení mohou kupříkladu zajišťovat bezpečnost pracovišť neustálým monitorováním škodlivin v ovzduší. Ze všech příkladů vyplývá, že se jedná o velmi citlivá data a oblasti nasazení. Přirozeným důsledkem je tudíž obrovský zájem útočníků o hledání a zneužití chyb v těchto systémech. Jejich zabezpečení je tak naprostou prioritou.

Hlavním cílem bakalářské práce je analýza zabezpečení komunikace protokolu ZigBee a jeho funkcionality Touchlink. Následným cílem je implementace aplikace pro příkazový řádek, která zjednoduší útoky na tento protokol s využitím zranitelností objevených v této analýze. Dílčím cílem této implementace je použití technologie softwarově definovaného rádia (SDR) v rádiové vrstvě aplikace. Posledním cílem je aplikaci otestovat na reálné síti a zhodnotit její praktickou použitelnost.

V kapitole 2 uvedeme krátký úvod do state-of-the-art technologií používaných v oblasti IoT. V navazující kapitole 3 provedeme podrobnou analýzu normy IEEE 802.15.4,

protokolu ZigBee a jeho funkcionality Touchlink, spolu s analýzou již dostupných nástrojů pro práci se ZigBee a Touchlink. Na základě této analýzy vymezíme a navrhneme v kapitole 4 aplikaci pro příkazový řádek, která pro útok na tento protokol využije zjištěných zranitelností. Tuto aplikaci implementujeme v kapitole 5 a spolu s tím otestujeme na reálné síti v kapitole 6. V závěru shrneme dosažené výsledky a představíme možný budoucí vývoj v kapitole 7.

Kapitola 2

State-of-the-Art

V této kapitole nejprve představíme krátký úvod do technologie internetu věcí, v něm užívaných komunikačních protokolů a obecný model zabezpečení. Navážeme představením jedné z technologií zvyšující dostupnost útoků na internet věcí, technologie softwarově definovaného rádia.

2.1 Referenční model OSI

První experimentální počítačové sítě vznikaly nezávisle na sobě v laboratořích vojenských výzkumných institucí a univerzit. Nejznámějšími příklady těchto sítí jsou projekty ARPANET a CYCLADES. Každý takový projekt vytvořil svou síťovou architekturu, která popisuje sadu vnitřních konvencí pro propojení síťových uzlů. Přirozeným důsledkem byla potřeba univerzálního propojení těchto heterogenních systémů. Tento problém dostal označení Open Systems Interconnection (OSI) a začala se jím zabývat Mezinárodní organizace pro normalizaci (ISO), která vypracovala referenční model OSI. [1]

2.1.1 Vrstvy modelu

Referenční model OSI popisuje teoretickou sedmivrstvou síťovou architekturu. Vrstvy modelu jsou záměrně oddělené, aby bylo možné je nezávisle vyvíjet a měnit. Každá vrstva N má svou specifickou funkci, kterou poskytuje vrstvě $N + 1$ pomocí rozhraní v rámci jednoho síťového uzlu. Tento model neobsahuje popis výměny dat vrstvy N mezi dvěma síťovými uzly. K tomu jsou definovány síťové protokoly vrstvy N v podobě norem či Request for Comments (RFC), které přenášejí protocol data units (PDU) [1]. Zdroj [2] uvádí popis jednotlivých vrstev:

Fyzická vrstva je první nebo také nejnižší vrstvou OSI modelu a je označována jako PHY. Funkcí vrstvy PHY je obsluha přenosového média a samotný bitový přenos.

PHY protokoly definují mechanické a elektrické parametry, mezi které patří napětí, přenosová rychlost, fyzické médium či maximální vzdálenost síťových uzlů. V neposlední řadě definují způsob kódování bitů pro konkrétní médium a duplex;

Spojová vrstva umožňuje přenos dat mezi dvěma fyzicky spojenými uzly. Navíc poskytuje detekci a korekci chyb vzniklých na fyzické vrstvě a řídí datový tok. Norma IEEE 802 [3] pro lokální sítě (LAN) a metropolitní sítě (MAN) tuto vrstvu rozděluje na dvě podvrstvy. První podvrstvou je medium access control (MAC) zajišťující výlučný přístup k přenosovému médiu. Druhou podvrstvou je logical link control (LLC), která zastřešuje kontrolu datového toku spolu s detekcí a korekcí chyb fyzické vrstvy;

Síťová vrstva obsluhuje doručení dat mezi dvěma síťovými uzly na základě logických adres, které mají uzly v síti přidělené. Toho je docíleno směrováním, které představuje hledání cesty mezi zdrojovým a cílovým uzlem skrze mezilehlé uzly. V případě, že je velikost PDU této vrstvy větší, než je maximální velikost určená spojovou vrstvou, může tato vrstva PDU rozdělit na několik menších, které pošle odděleně a opět sestaví do jednoho v cílovém uzlu;

Transportní vrstva zajišťuje doručení dat ke konkrétní službě (aplikaci) na cílovém uzlu. Toho je docíleno pomocí 16bitových identifikátorů, portů, které jsou další vrstvou logického adresování. Volitelně může tato vrstva zajišťovat spolehlivost přenosu. Toho může být docíleno potvrzováním obdržených PDU a opětovným přenosem těch nepotvrzených;

Relační vrstva obsluhuje zřízení, průběh a ukončení relace v rámci komunikace aplikace běžící na dvou síťových uzlech, včetně kontroly a případného zotavení při výpadku;

Prezenční vrstva má jako svou nejdůležitější funkci překlad přenášených dat z formátu přenášeného po síti do formátu, kterému rozumí cílová aplikace. Dále má tato vrstva na starost šifrovaní, dešifrování, kompresi a dekompresi přenášených dat;

Aplikační vrstva je sedmou, tedy nejvyšší vrstvou OSI modelu a zastřešuje interakci uživatele a vzdálených síťových služeb. Tuto interakci zprostředkovávají aplikační protokoly, které jsou implementovány v konkrétních aplikacích. [2]

2.2 Internet věcí

Samotný pojem „internet“ je výsledkem několika desetiletí vývoje propojení počítačových sítí od projektu ARPANET a odkazuje na široké spektrum sofistikovaných síťových protokolů a aplikací, které nepřetržitě slouží miliardám uživatelů na celém světě. Pojem „věc“ v tomto případě zahrnuje širší spektrum všedních objektů, než jsou pouze ty schopné být připojeny k internetu. Mezi takové objekty řadíme senzory, přes chytrá

zařízení a lidské bytosti, které jsou si vědomy své souvislosti v rámci většího celku a jsou kdykoliv a kdekoliv dostupné. [4]

Prostředí IoT roste velmi rychle. V roce 2021 bylo připojeno přes 12,3 miliardy zařízení a předpokladem je více než 27 miliard v roce 2025 [5], přičemž v roce 2020 byl tento předpoklad pouze 22 miliard [6]. Jde tak o značně heterogenní prostředí s širokým spektrem nasazení, do kterého patří například kontrola dopravy či zdravotnictví. Všechna tato nasazení mají své specifické požadavky a omezení na škálovatelnost, spotřebu energie či pokrytí, a proto vznikla řada protokolů pro uspokojení těchto potřeb. [7]

2.2.1 Komunikační protokoly

Stavebními bloky IoT jsou senzory a síťová komunikace. IoT má za snahu dosáhnout jejich univerzálního a všudypřítomného propojení. Právě toto vzájemné propojení je stěžejním požadavkem na systémové architektury IoT. Nedílnou součástí IoT je pohyblivost objektů a možnost je dynamicky v rámci systému přesouvat. Tato skutečnost má za důsledek vysokou míru použití rádiových komunikačních protokolů. [4]

Obecně lze bezdrátové komunikační protokoly užívané v IoT rozdělit do dvou kategorií. Tou první jsou protokoly s dlouhým dosahem, jejichž příkladem jsou LoRaWan, SigFox a NB-IoT. Zástupci této kategorie neumožňují vzájemnou komunikaci uzlů a vyžadují spojení s centrálním bodem. Jejich typickým využitím jsou inteligentní města. Příkladem zařízení využívající tyto protokoly může být pouliční lampa či měřič spotřeby vody. Očekává se tak komunikace zřídka, přibližně jednou denně. Druhou kategorií jsou protokoly s krátkým dosahem. Mezi ty patří například Bluetooth Low Energy či ZigBee. Ty mohou být typicky připojeny k jiné síti a umožňují velmi častou vzájemnou komunikaci uzlů. Takové sítě mnohdy integrují senzory a akční členy. [7]

2.2.2 Bezpečnost

Rostoucí rozšířenost a adopce IoT zvyšuje z několika důvodů zájem zlomyslných uživatelů a útočníků. Prvním důvodem je množství zařízení v IoT, protože to poskytuje útočníkům velkou plochu útoku. Navíc toto množství a rozptýlenost těchto zařízení značně ztěžuje schopnost vývoje a nasazení bezpečnostních aktualizací. V neposlední řadě mají útočníci velmi často fyzický přístup k těmto zařízením, což razantně zvyšuje šanci útočníka na průnik do zařízení a dělá jej jednodušším. [4]

Jak dále uvádí norma IEEE 802.15.4 [8], bezdrátová komunikace je zranitelná vůči pasivnímu odposlechu a manipulaci vysílaných dat, protože útočník nepotřebuje fyzický přístup k přenosovému médiu. Účelová podstata IoT sítí však pravděpodobně představuje nejobtížněji zabezpečitelné prostředí. Tato zařízení jsou tradičně levná, napájená z baterie, s omezeným výpočetním výkonem a úložištěm. U pohybujících se zařízení se mimoto nelze spolehnout na pevnou infrastrukturu a další nebezpečí pro ně představuje

jednoúčelová komunikace s neznámými zařízeními. Všechny tyto skutečnosti silně omezují a ovlivňují návrh bezpečnostní architektury, protože navazování a udržování vztahů důvěry mezi zařízeními je třeba řešit s extrémní opatrností.

Implementace standardních bezpečnostních mechanismů v těchto zařízeních je zásadně ovlivněna právě dostupnými zdroji. Nelze se spolehnout na to, že mají důvěryhodnou výpočetní základnu ani dostatečně kvalitní generátor náhodných čísel. Z těchto důvodů volí většina implementací symetrickou kryptografii, protože její výpočetní nároky jsou menší než u asymetrické kryptografie. [8, 4]

2.3 Softwarově definované rádio

Útoků na protokoly internetu věcí, a bezdrátové protokoly obecně, však nepřibývá pouze z důvodu rostoucího nasazení, ale také díky technologiím umožňujícím práci s nimi, jako je technologie SDR. Tato technologie obecně představuje systém, ve kterém jsou hardwarové prvky pro digitální zpracování signálu (DSP) implementovány pomocí softwaru, nejčastěji na programovatelných hradlových polích (FPGA). Softwarová implementace může být pouze částečná, v každém případě ale zajišťuje možnost snadné a rychlé rekonfigurace na různé rádiové protokoly. [9, 10]

Širší využití této technologie je mimoto způsobeno širokou nabídkou dostupného hardwaru. Produkty jako jsou RTL-SDR zajišťují snadný vstup pro radioamatéry, převážně díky svojí ceně [11]. Proti tomu zařízení podobné Ettus USRP B210 nabízí vysoký frekvenční rozsah a vzorkovací frekvenci. Tyto vlastnosti poskytují daleko vyšší citlivost, a taková zařízení je dokonce možné využít k vytvoření vlastní základnové stanice technologie Groupe Spécial Mobile (GSM) [12].

2.3.1 GNU Radio

GNU Radio je volně dostupný state-of-the-art soubor softwarových nástrojů pro implementaci softwarových rádií. Obsahuje již hotové bloky pro DSP, mezi které patří filtry, dekodéry či demodulátory. Umožňuje tyto bloky propojit a definovat pravidla pro tok dat mezi nimi. Primárním jazykem pro tvorbu aplikací za pomocí GNU Radio je jazyk Python, přičemž pro výkonnostně kritické aplikace je možné využít jazyk C++. GNU Radio také nabízí grafické rozhraní GNU Radio Companion, ve kterém je možné bloky propojit a nechat vygenerovat výslednou aplikaci v jazyce Python. Stěžejním blokem každé GNU Radio aplikace je takzvaný flowgraph, který představuje orientovaný graf propojení všech užitých bloků. [13]

Kapitola 3

Analýza

V této kapitole nejprve provedeme rozbor normy IEEE 802.15.4. Dále předložíme analýzu na ní postaveného protokolu ZigBee a jeho funkcionality Touchlink. Jako poslední budeme prezentovat rozbor již existujících nástrojů umožňujících analýzu přenosu a zabezepčení ZigBee.

3.1 IEEE 802.15.4

Norma IEEE 802.15.4 [8] definuje specifikace fyzické vrstvy a podvrstvy MAC, spojové vrstvy OSI modelu, pro bezdrátové připojení s nízkou rychlostí přenosu dat. Cílem této specifikace jsou stojící, přenosná a pohyblivá zařízení s požadavky na velmi omezenou spotřebu energie. Fyzická vrstva je definována pro zařízení pracující v různých frekvenčních pásmech z důvodu legislativních restrikcí odlišných zeměpisných oblastí. Kromě toho poskytuje norma režimy přesného měření vzdálenosti.

3.1.1 Fyzická vrstva

Provoz fyzické vrstvy je specifikován ve třech frekvenčních pásmech, dle již zmíněných legislativních restrikcí, za použití několika typů modulace signálu [14]. Tato tři frekvenční pásma jsou:

868–868,6 MHz k použití v Evropě. V tomto rozsahu je definován jeden kanál označený 1 u modulace ASK nebo 0 za podmínky využití ostatních typů modulace. Přehled propustnosti v závislosti na použité modulaci je vyobrazen v tabulce 3.1;

902–928 MHz k použití v USA. V tomto rozsahu je definováno deset kanálů označených 1–10. Přehled propustnosti v závislosti na použité modulaci je vyobrazen v tabulce 3.2;

2401–2480 MHz k použití celosvětově. V tomto rozsahu je definováno šestnáct kanálů označených 11-26. Jedinou modulací je O-QPSK s propustností 250 kbps. [14]

Tabulka 3.1 Propustnost pásma 868–868,6 MHz v závislosti na modulaci [14]

Modulace	Propustnost
BPSK	20 kbps
BPSK+O-QPSK	100 kbps
BPSK+ASK	250 kbps

Tabulka 3.2 Propustnost pásma 902–928 MHz v závislosti na modulaci [14]

Modulace	Propustnost
BPSK	40 kbps
BPSK+O-QPSK	250 kbps
ASK	250 kbps

Pásmo 2,4 GHz je téměř celosvětově dostupné bez licence pro průmyslové, vědecké a medicínské (ISM) účely. Přirozeným důsledkem je jeho použití v mnohých normách pro bezdrátovou komunikaci, například IEEE 802.11 (Wi-Fi), ale také v mikrovlnných troubách. Navíc má toto pásmo horší schopnost fyzické prostupnosti signálu, než je tomu u zbylých dvou pásem. Přes tyto skutečnosti se v implementacích normy IEEE 802.15.4 téměř výhradně využívá právě toto pásmo [14, 15]. Dle zdroje [15] tuto volbu nejvýznamněji ovlivňují tři faktory:

- nasazení zjednodušuje celosvětová dostupnost bez licence;
- větší množství kanálů a vysoká propustnost;
- menší spotřeba energie z důvodu větší propustnosti, tedy kratšího vysílacího času.

Obecný formát PDU vrstvy PHY (PPDU) je vyobrazen na obrázku 3.1.

Synchronizační hlavička (SHR)	PHY hlavička (PHR)	PHY payload (PSDU)

Obrázek 3.1 Formát obecného PDU fyzické vrstvy normy IEEE 802.15.4 (PPDU) [8]

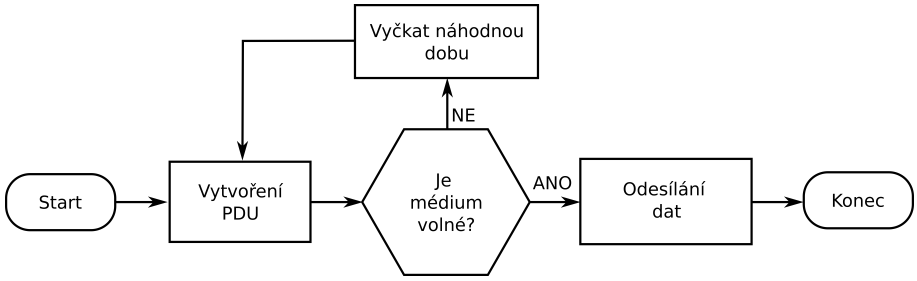
3.1.1.1 Funkce kvality přenosu

Aby se předešlo rušení technologií v pásmu 2,4 GHz, je definováno několik doporučení. IEEE 802.15.4 nevyužívá techniky frequency-hopping spread spectrum (FHSS), ve které je pásmo rozděleno na 79 kanálů po 1 MHz. Ty se používají v předem definované sekvenci, každý však maximálně 400 milisekund. Kolize s technologiemi využívající FHSS je díky tomu občasná a není nikterak řešena. V případě techniky direct-sequence spread spectrum (DSSS), která toto pásmo rozděluje na 11 kanálů po 16 MHz, je doporučením použití kanálů 15, 20, 25 a 26, které se nepřekrývají s nejpoužívanějšími kanály 1, 6 a 11 normy IEEE 802.11. Fyzická vrstva poskytuje aplikační vrstvě také funkci detekce energie, která změří úroveň energie každého kanálu, a dovoluje tak aplikacím implementovat automatickou volbu kanálu. [14]

Další funkcí kvality přenosu je link quality indicator (LQI). Tuto informaci je možné připojit za každé přijaté PDU, přičemž obsažené informace závisejí na konkrétní implementaci. Vyšší vrstvy, spolu s informací o počtu opakovaných přenosů a počtu ztracených PDU, mohou například rozhodnout o zvýšení vysílacího výkonu i změně kanálu. [14]

Jak uvádí zdroj [14], třetí funkcí kvality přenosu je použití carrier-sense multiple access with collision avoidance (CSMA/CA). CSMA/CA je metodou prevence kolizí vysílání na přenosovém médiu několika účastníky. Její funkce, která je znázorněna také na obrázku 3.2, spočívá v tom, že před vysíláním uzel nejprve naslouchá na přenosovém médiu, zda nevysílá někdo jiný. Pakliže nevysílá, začne vysílat on. V opačném případě vyčká náhodnou dobu a začne opět naslouchat. IEEE 802.15.4 přesně nedefinuje, kdy je kanál volný. Implementace si mohou zvolit jednu ze tří metod detekce volného kanálu:

- zhodnocení úrovně energie kanálu;
- detekci modulace;
- kombinaci obou předchozích. [14]



■ **Obrázek 3.2** Diagram metody CSMA/CA využíváné k prevenci kolizí přenosu na společném médiu více účastníky. Metoda spočívá v naslouchání na médiu, zda vysílá někdo jiný. Jestliže vysílá, vyčkáme náhodnou dobu a začneme opět naslouchat. V opačném případě vysíláme my [16]

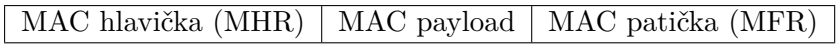
3.1.2 Podvrstva MAC

IEEE 802.15.4 rozděluje podvrstvu MAC na další dva objekty:

MAC common part sublayer (MCPS) zajišťuje samotný přenos PDU;

MAC layer management entity (MLME) zapouzdřuje konfiguraci vrstvy MAC. Ta zahrnuje adresy zařízení, počet opakování při zjištěné kolizi na společném médiu či atributy potvrzování PDU. [14]

Specifikace obecného PDU této vrstvy je vidět na obrázku 3.3. Norma dále specifikuje potvrzovací PDU ACK. ACK odesílá příjemce ihned po přijetí libovolného PDU určeného pro něj bez ohledu na CSMA/CA, což je umožněno díky tomu, že CSMA/CA umisťuje před každý přenos krátkou prodlevu. [14]



Obrázek 3.3 Formát obecného PDU podvrstvy MAC normy IEEE 802.15.4 [8]

3.1.2.1 Adresace

K adresaci síťových uzlů se používají dva typy adres:

Extended unique identifier (EUI-64) je 64bitová unikátní adresa definovaná v IEEE 802 [3], kterou má přidělenou každé zařízení. Tato adresa se skládá z 24bitového prefixu organizationally unique identifier (OUI), který má přidělen každý výrobce. Zbylých 40 bitů, rozšířený identifikátor, přiděluje výrobce svým zařízením sám;

Krátká adresa je 16bitový unikátní identifikátor v rámci osobní sítě (PAN), o kterou může zařízení požádat PAN koordinátora, který je vysvětlen v sekci 3.1.2.2. [14]

V obou případech je definována také všesměrová (broadcast) adresa, ve které jsou všechny bity nastaveny na hodnotu 1. [8]

3.1.2.2 Typy uzlů

Zdroj [14] uvádí, že ústředním uzlem každé IEEE 802.15.4 sítě je PAN koordinátor. Každá síť má právě jednoho koordinátora, kterým se stává zařízení, které jako první pošle beacon PDU. Beacon obsahuje informace o síti a je dále vysvětlen v sekci 3.1.2.4. Koordinátor má na starost několik klíčových funkcí, mezi které patří:

- skenování a výběr vhodného kanálu;
- volbu krátké adresy PAN (PAN ID);
- obsluhu připojení zařízení do sítě. [14]

Obecně se zařízení v IEEE 802.15.4 sítích dělí do dvou kategorií:

Plně funkční zařízení (FFD) mají jako svou hlavní vlastnost možnost stát se koordinátorem [8]. Zároveň mohou přeposílat PDU jiným uzlům, jinými slovy fungovat jako směrovače. Ať už je FFD koordinátorem či ne, může také být rodičem jiného zařízení v síti [14];

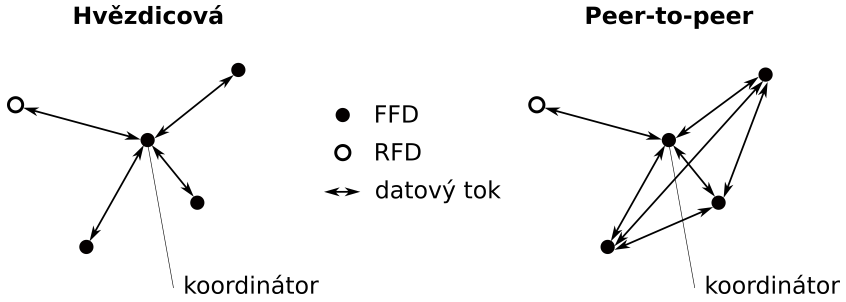
Zařízení s omezenou funkcí (RFD) se nemohou stát koordinátorem ani rodičem jiných zařízení. Hlavním důvodem je jejich neschopnost být směrovačem. Typickým příkladem takových zařízení jsou malé senzory teploty, jejichž přijímač i vysílač je většinu času vypnut. Probouzejí se pouze v krátkých časových intervalech, ve kterých mohou požádat svého rodiče o doručení zpráv, které jim uchoval v době jejich spánku, nebo sami odeslat data. [14, 8]

3.1.2.3 Síť

Z informací uvedených ve zdroji [8] vyplývá, že síť je tvořena alespoň dvěma zařízeními, která spolu komunikují na stejném kanálu, přičemž alespoň jedno z nich je koordinátor, a tedy nutně FFD. V závislosti na požadavcích konkrétní aplikace této normy operuje síť v jedné ze dvou možných topologií, které jsou následující a vyobrazené na obrázku 3.4:

Hvězdicová topologie směruje veškerou komunikaci skrze PAN koordinátora, ke kterému jsou připojeny všechny uzly. Koordinátor je v tomto případě velmi často napájen ze sítě, naproti tomu ostatní zařízení z baterie. Typickou ukázkou hvězdicové topologie je automatizace domácností;

Peer-to-peer (P2P) topologie dovoluje komunikaci jakýchkoli dvou FFD v dosahu. RFD se v tomto případě připojují pouze jako listy takové topologie, protože nemohou být rodiči. P2P topologie může být výhodná například pro aplikace inteligentního zemědělství či průmyslového dohledu. [8]



■ **Obrázek 3.4** Síťové topologie podporované normou IEEE 802.15.4. Ve hvězdicové topologii všechny zprávy v síti proudí skrze koordinátora sítě. V peer-to-peer topologii komunikují uzly s největším možným počtem sousedících uzlů [8]

Ve chvíli, kdy se chce zařízení připojit do sítě, pošle požadavek na připojení na adresu koordinátora dané sítě, kterou získá pomocí jednoho ze dvou typů skenování okolí:

Aktivní sken používá beacon request PDU, které vysílá na vybraných kanálech a očekává odpověď ve formě beacon PDU od koordinátora. Beacon PDU obsahuje informace potřebné k vyplnění požadavku na připojení k síti;

Pasivní sken pouze naslouchá a čeká na beacon PDU, které může koordinátor volitelně periodicky vysílat. [8]

Obsahem žádosti o připojení je pole informací o schopnostech daného zařízení, které je vyobrazeno v tabulce 3.3. Pokud zařízení žádá o přidělení krátké adresy, může koordinátor adresu přidělit, či speciální hodnotou 0xFFFE oznámit, aby zařízení používalo své EUI-64. Přirozeně může toto připojení selhat z důvodu nedostatečného oprávnění nebo dosažení maximálního počtu zařízení v síti. [14, 8]

Tabulka 3.3 Struktura pole schopností zařízení normy IEEE 802.15.4 [8]

Bit	Pole	Hodnota (výchozí 0)
0	Vyhrazeno	
1	Typ zařízení	1 pro FFD, 0 pro RFD
2	Zdroj napájení	1 pro napájení ze sítě
3	Přijímač zapnutý při nečinnosti	1 pokud zařízení naslouchá v nečinnosti
4	Typ připojení	1 pro rychlé připojení
5	Vyhrazeno	
6	Bezpečnostní schopnosti	1 při schopnosti zpracovat zabezpečené PDU
7	Přidělení adresy	1 při požadavku o přidělení krátké adresy

3.1.2.4 Přístup k médiu

Sítě mohou operovat ve dvou módech lišících se ve vysílání beacon PDU. Tyto dva operační módy jsou:

Beacon-enabled (slotted CSMA/CA) mód operuje v režimu periodického vysílání super-rámce koordinátorem, jež obsahuje instrukce k obsluze CSMA/CA. V tomto módu je možné přidělit garantovaná časová okna (GTS) pro vysílání určitým zařízením v síti. Jde tak o formu časového multiplexu (TDM). Těchto oken je dostupných patnáct, přičemž ta nevyužitá pro GTS implikují použití běžného CSMA/CA. Okna nevyužitá pro GTS vždy předcházejí všechna přidělená pro GTS. Jako poslední může koordinátor v této struktuře určit dobu svého spánku;

Nonbeacon-enabled (unslotted CSMA/CA) mód užívá běžného CSMA/CA, přičemž připojující se zařízení obdrží beacon PDU při aktivním skenování. [14, 8]

3.1.2.5 Bezpečnost

Norma IEEE 802.15.4 využívá k zabezpečení přenosu symetrickou kryptografii, která poskytuje důvěrnost a integritu přenášených dat spolu s ochranou před útoky přehráním. Implementace bezpečnosti je volitelná. Zřízení, správu a ochranu použitých klíčů mají na starost vyšší vrstvy a nejsou specifikovány touto normou. Klíč může být zřízen pro spojení právě dvou zařízení či skupinu. Požadovaná úroveň ochrany se dá měnit u každého PDU zvlášť. [8]

Zvoleným algoritmem je bloková šifra Advanced Encryption Standard (AES) v operačním módu CCM* [8]. Tento operační mód je představitelem authenticated encryption with associated data (AEAD) dle specifikace v RFC 5116 [17]. Vyjma důvěrnosti dat tedy poskytuje také jejich integritu a autentizaci na základě přidružených dat [18]. Operační mód CCM nejprve vygeneruje Message Integrity Code (MIC) pomocí AES v operačním módu cipher block chaining message authentication code (CBC-MAC), který slouží k zajištění integrity. Důvěrnost je zajištěna následnou aplikací AES v counter (CTR)

módu. Tento operační mód převádí AES na proudovou šifru, která potřebuje na vstupu nonce (number used once) a vstupní data, ke kterým se připojí vygenerovaný MIC. Operační mód CCM* je rozšířením operačního módu CCM. Toto rozšíření spočívá v možnosti využít pouze operace zajišťující důvěrnost či pouze operace pro zajištění integrity a zpřístupňuje MIC vygenerovaný módem CBC-MAC [19, 14]. Konkrétní využití tohoto algoritmu, sestavení nonce a správu klíčů si dále představíme v sekci 3.2.5 o bezpečnosti v ZigBee sítích.

3.2 ZigBee

Jak uvádí zdroj [14], IEEE 802.15.4 definuje nejnižší dvě vrstvy referenčního modelu OSI. Pro praktickou použitelnost není však tato norma sama o sobě dostatečná. Sítě obsahující malé senzory a podobná zařízení téměř nemohou fungovat bez topologie typu mesh, ve které uzly komunikují s největším možným počtem ostatních pro efektivní směrování zpráv sítí. Dále potřebují vymezit pevnou univerzální strukturu aplikačních dat. Proto v roce 2002 vznikla ZigBee Alliance s cílem definovat tyto zbylé struktury, jejíž počet členů dnes přesahuje dvě stovky. ZigBee je volně dostupný standard, přičemž jeho úpravy mohou navrhovat pouze členové této aliance. V důsledku toho, že je každý člen povinen souhlasit se zveřejněním všech patentů, které by se technologie ZigBee mohly týkat, je zajištěna vysoká míra kompatibility mezi různými výrobci.

3.2.1 Architektura

Protokol ZigBee je tvořen čtyřmi vrstvami. Fyzická a spojová vrstva jsou definovány normou IEEE 802.15.4. Na fyzické vrstvě jsou využita všechna frekvenční pásma definovaná touto normou, téměř výhradně se však v konkrétních implementacích využívá pásmo 2,4 GHz [20, 15]. Spojová vrstva je aplikována pouze v nonbeacon-enabled módu, jež implikuje použití CSMA/CA bez GTS pro přístup k médiu. Dále je omezen rozsah PAN ID z původní maximální hodnoty 0xFFFE na 0x3FFF [14]. Samotné ZigBee definuje síťovou vrstvu (NWK) a kostru pro aplikační vrstvu. Tato kostra je tvořena application support sub-layer (APS) a ZigBee device object (ZDO) [14].

3.2.2 Typy síťových uzlů

ZigBee rozšiřuje typy síťových uzlů IEEE 802.15.4 následovně:

Koncové zařízení ZigBee (ZED) je obdobou RFD. ZED jsou typicky senzory, které jsou většinu času uspány. Mají povinnost se k síti připojit skrze směrovač, který je dále jejich rodičem;

ZigBee směrovač (ZR) je obdobou FFD. Směrovače naslouchají nepřetržitě, směrují zprávy sítí a figurují jako rodiče ZED;

ZigBee koordinátor (ZC) je opět obdobou FFD. ZC může navíc oproti směrovačům figurovat jako PAN koordinátor. Má tedy pravomoc vytvářet nové sítě. [14, 20]

ZigBee mimoto definuje unikátní druh uzlu, centrum důvěry (TC). Tento uzel a jeho funkci vysvětlíme v sekci 3.2.5.1. Centrem důvěry je typicky koordinátor. [20]

3.2.3 Vrstva NWK

Hlavním cílem síťové vrstvy je poskytnout směrování v síti s topologií typu mesh. K tomuto se využívají krátké adresy uzlů, které vždy přiděluje koordinátor z rozsahu 0x0000–0xFFF7, přičemž on sám má vždy přidělenou adresu 0x0000 [14]. Obecný formát PDU vrstvy NWK (NPDU) je vyobrazen na obrázku 3.5.

Oktetů: 2	**2**	**2**	**1**	**1**	**0/8**	**0/8**	**0/1**	**N**	**N**
Frame control	Cílová adresa	Zdrojová adresa	Radius	Pořadové číslo	Cílové EUI-64	Zdrojové EUI-64	Multicast control	Source route	Payload
NWK hlavička									Payload

Obrázek 3.5 Formát obecného PDU vrstvy NWK protokolu ZigBee (NPDU) [20]

3.2.3.1 Vytvoření sítě

Prvním krokem vytvoření sítě koordinátorem je provedení IEEE 802.15.4 aktivního skenování, tedy vyslání beacon request PDU na všech nakonfigurovaných kanálech. Na základě odchycených PDU vybere koordinátor nejvolnější kanál. Shodnou vytíženost dvou a více kanálů řeší následným provedením pasivního skenu k výběru nejvolnějšího kanálu. Tento sken může trvat, v závislosti na implementaci, desítky milisekund až minuty. Z toho plyne, že vytvoření sítě může být značně náročnou operací na prostředky koordinátora. Následně koordinátor, z již dostupných dat, vybere PAN ID z rozsahu 0x0000–0x3FFF. Proti normě IEEE 802.15.4 má ZigBee síť navíc rozšířené PAN ID (EPID), obdobu EUI-64 uzlů. Koordinátor může jako tuto adresu určit své EUI-64 nebo vybere náhodnou. [14]

3.2.3.2 Připojení k síti

Podobně jako koordinátor při vytvoření sítě, začne každé zařízení svůj pokus o připojení k síti aktivním skenováním každého nakonfigurovaného kanálu. Každý uzel typu ZR a ZC odešle jako odpověď beacon PDU, ve kterém je uvedeno PAN ID, krátká adresa nebo EUI-64 koordinátora a volitelně EPID. Dále proces probíhá stejně jako je tomu u IEEE 802.15.4. Za podmínky, že je v síti zapnuto zabezpečení, následuje proces distribuce klíče vysvětlený v sekci 3.2.5.1. [14]

Pokud uzel ztratí spojení se svým rodičem, může se o znovu připojení pokusit pomocí PDU NWK rejoin. Právě zde využije EPID, protože nového rodiče si vybere z beacon PDU s EPID stejným jako jeho předchozí síť. [14]

3.2.3.3 Směrování

PDU síťové vrstvy může být jedním ze tří typů na základě cílové adresy:

Broadcast má cílovou adresu 0xFFFF a je směrován všem uzlům v síti. Adresa 0xFFFD znamená směrování všem zrovna nespícím uzlům a v případě adresy 0xFFFC je směrováno všem uzlům typu ZR a ZC. Každý uzel typu ZC a ZR si také drží tabulku již obdržených broadcastů, aby se předešlo jejich zacyklení;

Multicast je rozeslán všem členům skupiny určené krátkou adresou skupiny, kterou dále představíme v sekci 3.2.7;

Unicast je směrován uzlu danému krátkou cílovou adresou. [14]

Směrování probíhá jedním ze tří způsobů:

Meshed směrování je výchozím modelem, který zakládá na ad-hoc on-demand distance vectoring (AODV) algoritmu. Jestliže uzel nezná cestu k cílovému uzlu, vyšle route request PDU jako broadcast na adresu 0xFFFC. Každý uzel typu ZC a ZR po přijetí tohoto PDU přičte svůj odhad kvality linky, po které ho dostal a přepošle dál. Cílový uzel se pokusí odhadnout, kdy pravděpodobně přijal všechny zprávy a vybere z nich cestu s nejlepší kvalitou. Poté odešle unicastem route reply PDU vybranou cestou, přičemž tuto cestu si postupně směrovače na ní uloží do své směrovací tabulky. ZigBee předpokládá stejnou kvalitu každé linky oboustranně;

Stromové směrování využívá vlastnosti ZigBee, že každý směrovač zná adresy přidělené svým potomkům a všem potomkům svých podřazených směrovačů. Pokud tedy přijde PDU s jednou z těchto cílových adres, může směrovat přímo. V opačném případě směruje na svého rodiče;

Source směrování je využito v situaci, kdy se v síti nachází uzel, který hojně komunikuje se všemi uzly v síti jako zdroj PDU. V takovém případě by tento uzel a všechny jeho sousední uzly musely mít ve své směrovací tabulce každý uzel v síti a byla by tak příliš veliká. Aby se tomuto předešlo, tento uzel nejprve vyšle many-to-one route request PDU, které se zastaví nejdále na pátém směrovači od něj. Od všech těchto směrovačů dostane cesty ke všem jejich potomkům, přičemž tyto směrovače si uloží cestu k němu. On sám tedy musí mít velkou směrovací tabulku, ale všem těmto směrovačům a cílovým uzlům stačí znát cestu pouze k němu. [14]

3.2.4 Vrstva APS

Funkcí vrstvy APS je obsluha aplikací běžících na zařízení. Každé zařízení může implementovat libovolný počet aplikací a je potřeba je nějakým způsobem obsluhovat na jedné síťové adrese. Vrstva APS k tomuto zavádí endpoint, jež je obdobou portů transportní vrstvy OSI modelu. Každý tento endpoint dále obsahuje deskriptor, který obsahuje endpoint ID, který má přidělen každá aplikace, ID aplikačního profilu, ID aplikačního zařízení, který určuje o jaký typ zařízení se jedná, například senzor teploty, ID aplikační verze a jako poslední seznam vstupně-výstupních clusterů, které si dále představíme v sekci 3.2.6. [14]

3.2.4.1 Adresace

Obecný formát PDU vrstvy APS (APDU) je vyobrazen na obrázku 3.6. Aplikace mají při použití unicastu možnost uvést jako cílovou adresu EUI-64 či krátkou adresu cílového uzlu, přičemž vrstva APS vždy tuto adresu přeloží na krátkou adresu. Toho je docíleno adresní tabulkou APS, ve které si každé zařízení drží dvojici krátké adresy a k ní vázané EUI-64. Tuto tabulku si plní zachycením broadcast PDU device announce, které je rozesláno při každém připojení nového zařízení či změně jeho krátké adresy. Chybějící záznam v tabulce je doplněn procesem hledání cesty popsaným v sekci 3.2.3.3. [14]

Druhou možností je využití multicastu za použití krátké adresy skupiny. Každá aplikace může požádat o připojení do skupiny. Krátká adresa se skupině, na úrovni vrstvy APS, přiděluje z rozsahu 0x0000–0xFFFF a každý uzel může být součástí maximálně šestnácti skupin. Na úrovni vrstvy NWK se tato PDU posílají jako broadcast s adresou 0xFFFF. To má za důsledek, že skupinové PDU obdrží každé zařízení v síti a je na vrstvě APS, aby se postarala o doručení pouze členům skupiny. [14]

Oktetů: 1	**0/1**	**0/2**	**0/2**	**0/2**	**0/1**	**1**	**0/N**	**N**
Frame control	Cílový endpoint	Adresa skupiny	Cluster ID	Endpoint profilu	Zdrojový endpoint	APS čítač	Rozšířená hlavička	Payload
APS hlavička								Payload

Obrázek 3.6 Formát obecného PDU vrstvy APS protokolu ZigBee (APDU) [20]

3.2.5 Bezpečnost

Bezpečnost je v každé ZigBee síti volitelná a dostupná na obou vrstvách NWK i APS. Nepoužívá se zabezpečení vrstvy MAC poskytované normou IEEE 802.15.4, protože by tak mohla být zašifrována vrstva PDU potřebná ke směrování na úrovni vrstvy NWK. Přesto se uplatňuje stejný mechanismus založený na symetrické kryptografii s algoritmem AES-CCM* se 128bitovým klíčem, aby mohly být využity hardwarové akcelerátory tohoto algoritmu na zařízeních splňujících IEEE 802.15.4 [14]. ZigBee operuje v open trust

modelu zabezpečení, tedy každá vrstva věří všem ostatním vrstvám v rámci zařízení. Ustanovit vztah důvěry je nutné pouze mezi zařízeními. Z tohoto vyplývá, že je možno symetrické klíče využívat napříč vrstvami. Všechna zařízení v síti musejí používat stejnou úroveň zabezpečení. Dále platí, že za zabezpečení PDU je zodpovědná nejvyšší vrstva, ze které PDU pochází. Každé PDU je také opatřeno 32bitovým pořadovým číslem, aby se předešlo útokům přehráním. Každé zařízení si pamatuje poslední pořadové číslo, které od zdrojového uzlu přijalo a PDU s pořadovým číslem menším nebo stejným odmítne. Maximální hodnota tohoto čísla nesmí přetéct a jeho vynulování je možné pouze obměnou síťového klíče, který si dále představíme v sekci 3.2.5.1 [14, 21].

Jak uvádí zdroje [21, 20], bezpečnost závisí na několika předpokladech. Prvním je bezpečné uchování a správa symetrických klíčů na zařízení, které nejsou dostupné nezabezpečenou cestou. S ohledem na nízkou cenu a druh koncových zařízení však není předpokladem ochrana klíčů při fyzickém přístupu. Druhým předpokladem je podpora obou modelů zabezpečení popsaných v sekci 3.2.5.1 a adaptace na model vybraný a využívaný konkrétní sítí. Posledním předpokladem je korektní implementace kryptografických mechanismů a kvalitní generátor náhodných čísel.

3.2.5.1 Modely zabezpečení

Obecně využívá ZigBee dva druhy klíčů:

Linkový klíč využívaný k ochraně PDU mezi dvojicí zařízení;

Síťový klíč využívaný všemi zařízeními pro ochranu broadcast PDU. [21]

Distribuce a správa těchto klíčů je definována jedním ze dvou modelů:

Distribuovaný model, tradičně využívaný v domácích prostředích, nabízí méně bezpečný, ale jednoduchý systém zabezpečení. V tomto modelu může každý směrovač generovat síťové klíče a distribuovat je ostatním uzlům. Při připojení nového zařízení do sítě dostane tento klíč od svého rodiče zašifrován linkovým klíčem, který musejí mít všechna zařízení přednastavený v případě, že je daný síťový klíč označen jako high-security. Pokud je označen jako standardní, může být přenesen nešifrován. Jak je dále uvedeno ve zdroji [14], linkové klíče mohou uzly ustanovit pomocí schématu symmetric key establishment (SKKE);

Centralizovaný model, tradičně využívaný v průmyslových aplikacích, zavádí již zmíněné centrum důvěry. TC tvoří centralizovaný bod, jež distribuuje a pravidelně obměňuje síťový klíč, ověřuje a povoluje připojení nových uzlů a generuje linkové klíče pro dvojice zařízení. Stejně jako v předchozím případě musejí mít všechny uzly přednastavený linkový klíč, kterým se šifruje úvodní předání síťového klíče, jež v tomto modelu není volitelné. [21]

V obou těchto modelech platí, že vrstvy NWK a APS sdílejí síťový klíč. Při použití distribuovaného modelu sdílejí také linkové klíče. Klíče přednastavené z výroby v každém zařízení dále dělíme na dva podtypy [21]:

Globální klíč distribuovaného modelu používaný ke komunikaci zařízení právě jednoho výrobce;

Přednastavený linkový klíč, který se používá ke komunikaci zařízení různých výrobců. Tento klíč se dále dělí na tři podtypy v závislosti na fázi životního cyklu zařízení. Během vývoje zařízení se používá vývojový linkový klíč, který později v procesu certifikace přejde na certifikační linkový klíč a nakonec při nasazení zařízení do produkce na master linkový klíč. [21]

Centralizovaný model odlišuje specifikaci linkového klíče mezi vrstvou NWK a APS. Vrstva NWK používá dva podtypy:

Přednastavený globální linkový klíč, kterým TC šifruje úvodní předání síťového klíče. Tento klíč může být specifický pro výrobce nebo je využit klíč „ZigBeeAlliance09“;

Přednastavený unikátní linkový klíč, který má stejnou funkci jako předchozí typ a je volitelným rozšířením. Rozdíl oproti předchozímu typu je jeho unikátnost pro každou dvojici TC a uzlu. Tento klíč je zaveden do zařízení i TC před připojením do sítě sériovým portem či jinou metodou nevyužívající ZigBee. [21, 20]

V tuto chvíli je již zřejmé a ve zdroji [21] také uvedeno, že úvodní předání síťového klíče je velkým bezpečnostním rizikem, které je způsobeno kompromisem mezi bezpečností a jednoduchostí nasazení ZigBee. Při použití standardního klíče distribuovaného modelu není klíč šifrován vůbec. Pakliže je použit univerzální globální klíč, může útočník předaný síťový klíč dešifrovat. Pokud je využit specifický linkový klíč výrobce, spoléháme na zajištění jeho utajení výrobcem, jež je jistě špatnou bezpečnostní praktikou, protože jak uvedeme v sekci 3.2.8, tradičně jsou tato tajemství velmi rychle odhalena. Navíc je téměř nemožné jejich únik opravit kvůli zpětné kompatibilitě, protože jsou dána staticky. Z tohoto důvodu specifikace ZigBee 3.0 z roku 2015 zavádí povinnost každého zařízení nést instalační kód. Instalační kód je 128bitové náhodné číslo spolu s 16bitovým cyklickým redundantním součtem (CRC) uvedené na zařízení. Nejčastější formou je uložení v quick response (QR) kódu. Instalační kód se za podmínky využití centra důvěry před připojením zařízení opět zavede sériovým portem či jinou metodou nevyužívající ZigBee. Následně se pomocí kompresní funkce Matyas-Meyer-Oseas (MMO) odvodí unikátní linkový klíč, kterým se šifruje úvodní předání síťového klíče. Není však povinností tyto kódy používat.

Po zavedení linkových klíčů vrstvy NWK je možné distribuovat linkové klíče vrstvy APS. Tyto klíče jsou například nutné pro ovládání určitých typů zařízení jako jsou zámky

dveří či garážových vrat, protože jsou to zařízení podléhající vyšší úrovni zabezpečení a typicky nepovolují ovládání skrze broadcasty a požadují unicasty jen od předem definovaných zařízení či pouze kontroléru [21]. Globální a unikátní linkový klíč vrstvy APS jsou stejné jako v případě vrstvy NWK, ale využívají se k zabezpečení obecné komunikace s TC, nikoliv pouze k předání síťového klíče. Mimoto zavádí vrstva APS ještě jeden druh klíče [21]:

Aplikační linkový klíč, o který TC požádá jeden ze dvojice uzlů, kteří spolu chtějí komunikovat. Centrum důvěry vygeneruje nový linkový klíč, který si uloží pro danou dvojici identifikovanou jejich EUI-64. Tento klíč následně zašifruje síťovým klíčem a každému z dvojice doručí. Jestliže má pro daný uzel z dvojice ustanovený také unikátní klíč, zašifruje aplikační klíč před doručením ještě tímto klíčem. [21]

Jak je dále uvedeno ve zdroji [21], TC průběžně obměňuje síťové a aplikační klíče. Pokud má s daným uzlem sjednán unikátní klíč, šifruje nový klíč jím, v opačném případě pouze původním síťovým klíčem. Nový klíč nevchází v platnost po obdržení, protože centrum důvěry musí počkat na potvrzení od všech uzlů, že nový klíč obdržely, a až poté vydá pokyn k použití nového klíče. Mimoto může centrum důvěry úvést do sítě několik aktivních síťových klíčů, které celá síť používá v předem definované sekvenci.

3.2.5.2 Zabezpečení PDU

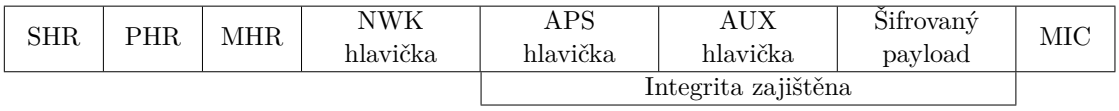
PDU může být opatřeno pomocnou (AUX) hlavičkou, kterou přidá nejvyšší vrstva, ze které PDU pochází, za svou hlavičku. Formát této hlavičky je vyobrazen na obrázku 3.7. Tato hlavička obsahuje pořadové číslo PDU, volitelně EUI-64 zdrojového uzlu, sekvenční číslo klíče pakliže je k zabezpečení PDU použit síťový klíč a pole řízení zabezpečení. Pole řízení zabezpečení je vyobrazeno na obrázku 3.8 a obsahuje pole úrovně zabezpečení, jehož hodnoty a důsledky pro zabezpečení PDU jsou vysvětleny v tabulce 3.4, identifikátor typu použitého klíče a příznak rozšířeného nonce, tedy zda je v AUX hlavičce obsaženo EUI-64 zdrojového uzlu. Zvolení úrovně zabezpečení využívající MIC navíc zajistí jeho připojení na konec PDU a zaručí integritu dané vrstvy, jak je vyobrazeno na obrázku 3.9 pro vrstvu APS. Stejný mechanismus funguje i pro vrstvu NWK. Jak již bylo zmíněno v sekci 3.1.2.5, AES v operačním módu CCM* potřebuje vstupní nonce, jehož stavba je v ZigBee uskutečněna pomocí informací, které mohou oba cílové uzly nezávisle získat. Nonce je vytvořen spojením bytů zdrojové EUI-64, pořadového čísla PDU a pole řízení zabezpečení za sebou. [14]

Oktetů: 1	**4**	**0/8**	**0/1**
Řízení zabezpečení	Sekvenční číslo	Zdrojové EUI-64	Sekvenční číslo klíče

Obrázek 3.7 Struktura AUX hlavičky PDU protokolu ZigBee [20]

Bit: 0–2	3–4	5	6–7
Úroveň zabezpečení	Identifikátor klíče	Rozšířené nonce	Vyhrazeno

Obrázek 3.8 Struktura pole řízení zabezpečení AUX hlavičky PDU protokolu ZigBee [20]



Obrázek 3.9 Struktura zabezpečeného PDU vrstvy APS protokolu ZigBee [21]

Tabulka 3.4 Hodnoty pole úrovně zabezpečení AUX hlavičky PDU protokolu ZigBee [20]

Úroveň zabezpečení	Bezpečnostní vlastnosti	Šifrování dat (ENC)	Integrita (M = oktetů MIC)
000	Žádné	NE	NE (M = 0)
001	MIC-32	NE	ANO (M = 4)
010	MIC-64	NE	ANO (M = 8)
011	MIC-128	NE	ANO (M = 16)
100	ENC	ANO	NE (M = 0)
101	ENC-MIC-32	ANO	ANO (M = 4)
110	ENC-MIC-64	ANO	ANO (M = 8)
111	ENC-MIC-128	ANO	ANO (M = 16)

3.2.6 ZigBee Cluster Library

ZigBee Cluster Library (ZCL) je podpůrná specifikace komunikace ZigBee aplikací. Obecně je cluster množina sdružených příkazů a atributů. Každý příkaz je opatřen identifikátorem z rozsahu 0x00–0xFF a každý atribut z rozsahu 0x0000–0xFFFF [14]. Cluster je rozdělen na dvě části:

Serverová část naslouchá na příkazy clusteru, vystavuje atributy, které je možné měnit a může podporovat jejich čtení zařízeními, jež implementují klientskou část;

Klientská část posílá příkazy podporované serverovou částí a čte hodnoty atributů serverové části. [14]

Každý cluster je definován pro konkrétní aplikační profil, jež jsou dále vysvětleny v sekci 3.2.7. Aby se předešlo redefinici společných clusterů, má každý takový cluster stejný identifikátor ve všech aplikačních profilech, jejichž je součástí. [14]

Zařízení definují své vstupní a výstupní clustery. Vstupní clustery určují, jaké příkazy umí zařízení provést a atributy, které je možné měnit. Výstupní clustery definují příkazy, jež je zařízení schopno poslat a atributy, které umí číst z jiných zařízení. [14]

3.2.7 ZigBee Application Profiles

Aplikační profily ZigBee definují komunikaci pro určitou formu nasazení konkrétní sítě. Každý aplikační profil má přiřazen identifikátor z rozsahu 0x0000–0xFFFF. Výrobci mohou specifikovat své proprietární profily, přičemž ty společné mají identifikátor z rozsahu 0x0000–0x7FFF. [14]

Každý aplikační profil definuje identifikátory obsažených zařízení. Pro každé takové zařízení dále definuje povinné a volitelné vstupní a výstupní clustery. Příkladem takového clusteru je Home Automation (HA), který definuje například světelný senzor s identifikátorem 0x0106, pro něj povinný serverový cluster Illuminance measurement s identifikátorem 0x0400 a volitelný klientský cluster Groups s identifikátorem 0x0004. [14]

3.2.8 Touchlink

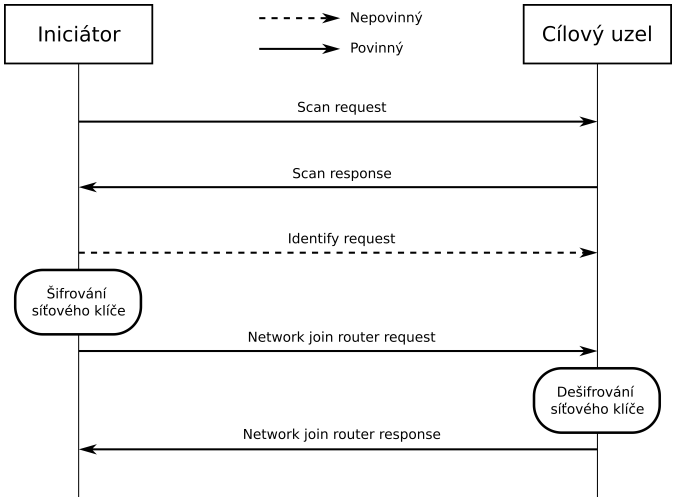
ZigBee definuje společný aplikační profil Light Link, který obstarává funkcionalitu spotřebitelských světelných zařízení. Běžné ZigBee sítě předpokládají existenci koordinátora, který je navíc kdykoliv dostupný. Pro jednoduché spotřebitelské nasazení ZigBee sítě, která obsahuje pouze jednotky světelných zařízení, se toto ukázalo jako silné omezení. Aplikační profil Light Link z tohoto důvodu definuje novou metodu vytvoření sítě a připojení zařízení do sítě nazvanou Touchlink [22]. Light Link je volitelnou součástí specifikace ZigBee 3.0, ale dle průzkumu trhu [23, 24] téměř veškeré světelné produkty podporující ZigBee 3.0 podporují také Light Link díky zpětné kompatibilitě se staršími produkty, které již spotřebitelé vlastní.

Obecně je Touchlink metoda nalezení zařízení v okolí na základě naměřené síly signálu, a proto tato zařízení musejí být velmi blízko u sebe. Vzdálenost není přesně definována a je na výrobcích, aby byla dostatečně krátká. Typickou implementovanou hodnotou je 1,5 metru. Touchlink využívá metody inter-PAN PDU, která obchází síťovou vrstvu ZigBee speciálním příznakem v PDU a dovoluje komunikovat dvěma zařízením na stejném kanálu, ale v jiných ZigBee sítích, pouze pomocí vrstvy MAC [22]. Typickým příkladem použití Touchlink je připojení dálkového ovladače ke světlu. Protože se v těchto případech očekává, že síť nemá koordinátora, poskytuje Touchlink základní příkazy pro správu zařízení, kupříkladu obnovu do továrního nastavení či změnu kanálu [25].

3.2.8.1 Proces komunikace

Proces komunikace začíná pomocí scan request PDU vyslaným iniciátorem Touchlink transakce, které je opatřeno náhodným 32bitovým identifikátorem transakce (TID). Každé zařízení v dostatečné blízkosti iniciátora vyšle zpět scan response PDU, jež obsahuje přijatý TID a 32bitový náhodný identifikátor odpovědi (RID). Od této chvíle obsahují PDU obou stran dané TID. Následně může iniciátor odeslat identify request PDU, jež instruuje zařízení k provedení nějaké akce, jež uživateli dovolí ho jasně identifi-

kovat. Typickým příkladem je zablikání žárovky na definovanou dobu [22]. Jako poslední odešle iniciátor PDU s příkazem. Tím může být například obnova do továrního nastavení či připojení k síti. Každý příkaz připojení k síti obsahuje EPID, PAN ID, krátkou adresu přidělenou zařízení, kanál sítě a šifrovaný síťový klíč. Celý proces je vyobrazen na obrázku 3.10. [22]



■ **Obrázek 3.10** Diagram průběhu připojení zařízení do ZigBee sítě využitím Touchlink. Iniciátor transakce nejprve vyšle scan request PDU, na které odpoví všechny uzly v dosahu scan response PDU. Následně může iniciátor vyslat identify request PDU k provedení akce, která umožní jasnou identifikaci cíle uživatelem. Následně odešle network join request PDU nesoucí informace potřebné pro připojení k síti, načež zařízení odpoví výsledkem v network join router response PDU [25]

3.2.8.2 Bezpečnost

Obecně je Touchlink komunikace zabezpečena pouze předpokladem, že jsou zařízení dostatečně blízko u sebe a nejsou využity žádné jiné vlastnosti zabezpečení PDU, které ZigBee nabízí. To samozřejmě otevírá prostor pro útoky typu denial-of-service (DOS) a příkladem je obnovení libovolného zařízení podporujícího Touchlink do továrního nastavení. Důsledkem je jeho odpojení od sítě nakonfigurované uživatelem, přestože nejsme součástí stejné sítě jako zařízení. Tento předpoklad navíc nedokáže zabránit odposlechnutí libovolné Touchlink transakce.

Kromě toho, transakce, které provádějí připojení k síti, přenášejí šifrovaný klíč sítě. Dle definice ve zdroji [22] je tento klíč šifrován pomocí AES v operačním módu electronic code book (ECB) s využitím Light Link master linkového klíče. Nejprve je vytvořen transportní klíč, který vznikne zašifrováním hodnoty vzniklé spojením bytů TID, TID, RID a RID za sebou tímto master klíčem. Síťový klíč je následně zašifrován vzniklým transportním klíčem. Master klíč je staticky definován ve specifikaci a je distribuován pouze mezi výrobce, kteří podepsali dohodu o mlčenlivosti. Jak je nicméně zmíněno ve

zdroji [25], v březnu 2015 se tento klíč objevil na sociální síti Twitter a dodnes se dá dohledat a ověřit pomocí otisku hašovací funkce z rodiny Secure Hash Algorithms (SHA) SHA-256 s hodnotou 57379035b23fed6aa2424e4791a21a13c7e9aa7881f0aa455a935f40ecbe41ff. Jak dále uvádí zdroj [25], tento objev byl oznámen a řešen s výrobci a ZigBee Alliance. Jak jsme však uvedli v sekci 3.2.5.1, z důvodu zpětné kompatibility nebylo možné tuto chybu nikterak opravit a všechna zařízení podporující Touchlink tak zůstávají rizikem pro ZigBee sítě.

3.3 Existující nástroje

K bezpečnostní analýze ZigBee lze využít řadu existujících nástrojů, které si krátce představíme. K samotnému rádiovému přenosu je jistě možné využít dříve představených technologií SDR a GNU Radio. Projekty gr-foo [26], gr-ieee802-15-4 [27] a gr-zigbee [28] poskytují bloky pro zpracování signálu normy IEEE 802.15.4 a protokolu ZigBee, přičemž gr-zigbee navíc také obsahuje ukázkový GNU Radio flowgraph pro příjem a odesílání ZigBee PDU.

3.3.1 Wireshark

Nejpokročilejším nástrojem pro pasivní analýzu PDU protokolu ZigBee je program Wireshark. Jedná se o protokolový analyzátor, který podporuje stovky protokolů, přičemž díky přispěvatelům do vývoje z řad odborné veřejnosti přibývají další. Jedná se o průmyslový standard v analýze síťové komunikace. Nejdůležitějším výběrem z jeho funkcionalit je analýza PDU v reálném čase i ze záznamu ze souborů ve formátu PCAP a široké možnosti filtrování. Wireshark také umožňuje dešifrovat řadu síťových protokolů za předpokladu známého klíče, včetně ZigBee. Přestože podporuje také aplikační profil Light Link, neumožňuje dešifrovat síťový klíč přenesený pomocí Touchlink. Mimoto pro ZigBee neposkytuje možnost odchycení a zobrazení PDU v reálném čase. [29]

3.3.2 Killerbee

Projekt KillerBee je sadou nástrojů pro testování bezpečnosti sítí IEEE 802.15.4 a ZigBee. K rádiovému přenosu podporuje řadu rádií, jako je například River Loop ApiMote. Většina jeho funkcionalit se soustředí na normu IEEE 802.15.4. Umí zachytávat PDU do souborů ve formátu PCAP, provést útoky přehráním či vysílat falešné beacon PDU. Umí také dešifrovat předání síťového klíče ZigBee, nikoliv však v případě Touchlinku. [30]

3.3.3 Scapy

Program a knihovna Scapy pro jazyk Python je převážně určena pro vytváření a dekódování PDU řady protokolů včetně ZigBee, a jako jeden z mála nástrojů umožňuje

vytvořit také neplatné PDU. Narozdíl od programu Wireshark umožňuje i jejich odesílání v reálném čase. Naproti tomu ale neumí dešifrovat PDU a u ZigBee úplně chybí podpora pro aplikační profil Light Link [31]. Přestože není možné odesílat ZigBee PDU pomocí SDR, je možné tohoto dosáhnout pomocí projektu scapy-radio [32]. Ten bohužel podporuje pouze verzi GNU Radio 3.7 a na distribuci Ubuntu 20.04 LTS operačního systému GNU/Linux není spustitelný.

3.3.4 Z3sec

Zdroj [25] uvádí projekt Z3sec [33], který se zabýval útoky na komunikaci Touchlink a uměl dešifrovat přenášený síťový klíč. Nicméně je závislý na zmíněné knihovně scapy-radio, jazyku Python verze 2 a GNU Radio 3.7, tudíž se nám nepovedlo ho spustit na Ubuntu 20.04 LTS. Nadto chybí funkcionality pro praktickou použitelnost, mezi které patří filtrování zařízení dle EUI-64 a vytváření vlastních ZigBee sítí.

Kapitola 4

Návrh aplikace

V této kapitole nejprve představíme zaměření naší aplikace, nazvané zigtoucher, spolu s motivací vedoucí k jeho výběru na základě předchozí analýzy. Dále uvedeme požadované vlastnosti, funkcionality a navrhovanou architekturu.

4.1 Požadavky

Jak vychází z předchozí analýzy, v tuto chvíli neexistuje funkční nástroj, který by se zabýval testováním bezpečnosti a útoky na Touchlink. Specifikujeme tedy aplikaci zigtoucher pro příkazový řádek, jejímž hlavním zaměřením bude odposlech, analýza a útok na komunikaci Touchlink s využitím uniklého master linkového klíče.

Rádiová vrstva musí být schopna využít technologii SDR. To nám nabídne velmi vysokou flexibilitu ve volbě konkrétního rádia uživatelem, dostupné implementační nástroje a přístup k diagnostickým datům během vývoje. Kromě toho musí být snadno rozšiřitelná o další typy rádií pro budoucí vývoj pomocí těch, která se specializují na normu IEEE 802.15.4.

První, a také nejdůležitější, funkcionalitou musí být odposlech a dešifrování síťového klíče přenášeného během připojení zařízení do sítě. Protože ale síť, jejíž klíč bychom chtěli odhalit, má již velmi pravděpodobně připojena všechna zařízení, budeme takovou transakci chtít vynutit. Z tohoto důvodu budeme požadovat schopnost obnovit zařízení v dosahu do továrního nastavení a donutit tak uživatele konkrétní sítě k připojení zařízení zpět pomocí Touchlink. Kromě toho nejspíše neznáme kanály, na kterých jsou v dosahu dostupná zařízení podporující Touchlink. Proto budeme vyžadovat schopnost skenování kanálů. Ze všech těchto událostí musíme být schopni ukládat detailní perzistentní záznamy k pozdější analýze.

Druhou zásadní funkcionalitou musí být schopnost takto získaná data využít, tedy odposlechnout libovolnou ZigBee komunikaci a dešifrovat ji. Tato funkcionalita je zásadní, protože nám umožní dešifrovat libovolnou následující obnovu síťového klíče dané

sítě za předpokladu, že komunikaci v této síti nepřestaneme sledovat. Abychom toto mohli provést, budeme požadovat perzistentní uložení dešifrovaných PDU.

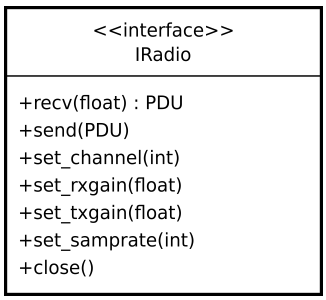
V obou těchto případech budeme požadovat schopnost nejen aktivního využití rádia, ale také analýzu již dříve zachycených PDU ze souboru.

Pro ukázku aktivního útoku budeme vyžadovat základní ovládání světelných zařízení. Navíc musí být dostupné ovládání jak zařízení z již odposlechnutých transakcí, tak vytváření vlastních ZigBee sítí a připojení zařízení do nich s využitím Touchlink.

Obecně budeme od aplikace požadovat jak interaktivní režim, například pro aktivní útoky, tak i běh v pozadí během odposlechu. Všechny funkcionality a perzistentní ukládání dat musejí být jemně konfigurovatelné, přičemž i tato konfigurace musí mít možnost perzistentního uložení. Aplikace musí být snadno rozšiřitelná o další funkcionality. Implementaci budeme vyžadovat pro operační systém GNU/Linux, distribuci Ubuntu 20.04 LTS, což nám zajistí velmi vysokou míru kompatibility i s jinými distribucemi tohoto operačního systému.

4.2 Rádiová vrstva

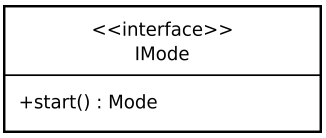
Pro zajištění snadné budoucí rozšiřitelnosti aplikace o další typy rádií, je definováno rozhraní pro jejich implementaci vyobrazené na obrázku 4.1. Každé rádio má navíc možnost nastavit zdrojovou EUI-64, a mimoto poskytuje schopnost uložení přijatých i odeslaných PDU v surové podobě do souborů ve formátu PCAP. Lze také přidružit okno programu Wireshark, ve kterém jsou veškerá PDU dostupná v reálném čase.



Obrázek 4.1 Rozhraní rádia aplikace zigtoucher, které je definováno pro snadnou rozšiřitelnost aplikace o další rádia

4.3 Operační módy

K dosažení požadovaných funkcionalit poskytuje aplikace několik operačních módů. Aby bylo dosaženo snadné rozšiřitelnosti, splňuje každý mód rozhraní vyobrazené na obrázku 4.2. Zároveň je možno specifikovat volitelné parametry, pro něž poskytujeme rozumné výchozí hodnoty. Navíc je možné určit časový limit ukončení libovolného módu. Operační módy dovolují cyklickou obsluhu libovolného výběru ZigBee kanálů.



Obrázek 4.2 Rozhraní operačního módu aplikace zigtoucher, které je definováno pro snadnou rozšiřitelnost aplikace o další operační módy

4.3.1 Sniff

Operační mód Sniff slouží k odposlechu libovolné ZigBee komunikace do souboru ve formátu PCAP. Za předpokladu poskytnutí dešifrovacího klíče jsou zachycená PDU dešifrována, AUX hlavičky odstraněny a do PCAP souboru uložena bez zabezpečení. Volitelně je možné tyto PDU zobrazovat na obrazovce. Z vlasnosti každého rádia mít možnost přidružené okno programu Wireshark vyplývá, že mód Sniff rozšiřuje Wireshark o chybějící schopnost zachytávat a zpracovávat ZigBee PDU v reálném čase.

4.3.2 Keylog

Operační mód KeyLog slouží k odposlechu Touchlink transakcí. U každé transakce sledujeme EUI-64 zapojených zařízení, počet zachycených PDU transakce a její typ, jež je určen jako jedna z hodnot Scan, Identify, Reset a KeyTransport na základě posledního zachyceného PDU. Poskytována je také možnost filtrovat transakce specifikací EUI-64 cílového uzlu.

U transakce typu KeyTransport, která připojuje zařízení do sítě, sledujeme také krátkou adresu přidělenou cílovému uzlu, kanál sítě, PAN ID a dešifrujeme síťový klíč. Tyto informace vždy zaznamenáváme do logu ihned po obdržení. Navíc je možné tato data ukládat perzistentně. První možností perzistentního uložení je soubor ve formátu comma-separated values (CSV). Struktura dat v tomto souboru je dále vysvětlena v sekci 4.6.2. Druhou možností je uložení do souboru konfigurace sítě, který využívá operační mód Control popsaný v sekci 4.3.6, jehož struktura je vysvětlena v sekci 4.6.1. V obou těchto případech však dochází k uložení pouze za předpokladu, že transakce skončila úspěchem a povedlo se klíč dešifrovat. Transakce může být označena jako neúspěšná cílovým uzlem anebo naší aplikací, jestliže nebyla zachycena všechna PDU transakce a nepovedlo se klíč dešifrovat.

Očekávané použití tohoto operačního módu je čekání na jednu transakci, a proto může automaticky přejít do operačního módu Sniff po první úspěšné transakci. Tento mód dále naslouchá na kanálu sítě, dešifruje pomocí zachyceného klíče a další parametry přebírá z výchozí konfigurace, jež je vysvětlena v sekci 4.5. Kombinace těchto operačních módů navíc splňuje požadavek na dešifrování obměn síťového klíče.

KeyLog umožňuje podrobný výpis výsledků, ve kterém jsou vypsány všechny transakce rozdělené podle kanálů, na kterých se uskutečnily. Každá transakce má přiřazen

identifikátor, jež může být použit v přidruženém módu Export, který umožňuje uložení PDU konkrétní transakce do souboru ve formátu PCAP.

4.3.3 Reset

Operační mód Reset umožňuje obnovu zařízení do továrního nastavení pomocí Touchlink s volitelnou zdrojovou EUI-64. Z toho vyplývá také možnost odeslání PDU identify request s volitelnou dobou trvání před obnovou zařízení. Stejně jako KeyLog má schopnost útoku pouze na konkrétní zařízení určené EUI-64.

Každé obnovené zařízení je ihned zaznamenáno do logu s informací o jeho EUI-64 a kanále, na kterém se transakce uskutečnila. Volitelně je možné tato data opět ukládat do CSV souboru, jehož formát je dále popsán v sekci 4.6.3. Aby nedocházelo k cyklickému obnovení jednoho zařízení během konkrétního běhu tohoto módu, již obnovená zařízení jsou vynechána.

4.3.4 Scan

Operační mód Scan umožňuje skenování okolí a nalezení zařízení podporujících Touchlink v dosahu. Obnovená zařízení jsou neprodleně uložena do logu, a stejně jako v předchozím případě je možné výsledky ukládat do souboru CSV, jehož struktura je totožná se strukturou souboru CSV operačního módu Reset. Každá EUI-64 je během jednoho běhu objevena nejvýše jednou.

4.3.5 Steal

Operační mód Steal umožňuje připojení zařízení do sítě pomocí metody Touchlink, opět s volitelnou zdrojovou EUI-64. Je umožněno vytvářet nové sítě či připojit zařízení do již existující sítě na základě existující konfigurace. Obdobně jako KeyLog a Reset umožňuje definovat EUI-64 cílového zařízení. Každé připojené zařízení je ihned zaznamenáno do logu a uloženo do konfiguračního souboru sítě.

4.3.6 Control

Operační mód Control slouží k interaktivnímu ovládání zařízení na základě konfigurace sítě s využitím běžných ZigBee PDU, nikoliv za použití Touchlink, avšak s plnou podporou všech úrovní zabezpečení ZigBee. Pro zjednodušení implementace jsou definovány pouze tři operace pro světelná zařízení. Operace identify s volitelnou dobou trvání, změna jasu s volitelnou velikostí změny a vypnutí a zapnutí zařízení. Protože jde o interaktivní mód, není dostupný v neinteraktivním režimu.

4.3.7 Transceiver

Operační mód Transceiver poskytuje možnost změny fyzických parametrů použitého rádia, tedy přijímacího a vysílacího zisku a vzorkovací frekvence. Tato změna je platná po zbytek trvání běhu aplikace. Spuštění tohoto módu bez parametrů nastaví zpět hodnoty definované výchozí konfigurací. Obsluhu změny kanálu mají na starost samotné operační módy za běhu. Tento mód není dostupný v neinteraktivním režimu, protože je v tomto případě nahrazen výchozí konfigurací.

4.4 Ovládání

Interaktivní režim poskytuje vlastní příkazový řádek, který poskytuje automatické doplňování operačních módů a parametrů pro vybraný mód po stisknutí klávesy Tab. Dále nabízí historii již použitých příkazů pomocí klávesových šipek nahoru a dolů. Běhové informační výpisy aplikace jsou zobrazeny barevně k oddělení od uživatelského rozhraní. K zjednodušení ovládání je navíc poskytnut krátký manuálový výpis pro každý operační mód, který obsahuje popis fungování, dostupné parametry a příklad použití.

4.5 Konfigurace

Konfigurace aplikace je umožněna na třech úrovních. První možností je konfigurační soubor pro uložení perzistentní konfigurace, druhou možností jsou přepínače při spuštění aplikace vhodné převážně pro neinteraktivní režim a poslední možností je specifikace parametrů při spuštění operačního módu v interaktivním režimu. Všechny tyto možnosti jsou volitelné, protože poskytujeme rozumné výchozí hodnoty. Priorita hodnot z těchto možností je uvedena od nejvýznamnější následovně, parametry interaktivního režimu, přepínače při spuštění a konfigurační soubor. Parametry v interaktivním režimu jsou navíc platné pouze pro konkrétní běh operačního módu. Naproti tomu zbylé možnosti konfigurace jsou platné po celou dobu běhu aplikace. Pojem výchozí konfigurace označuje kombinaci hodnot vzniklou z konfiguračního souboru a přepínačů.

Všechna potřebná nastavení již byla zmíněna v předchozích sekcích návrhu. Konkrétní implementace se řídí návrhem výchozího konfiguračního souboru z výpisu kódu 4.3. Pro plnou funkcionalitu však aplikace vyžaduje vyplnění dříve zmíněného Touchlink master klíče. Ten není z bezpečnostních důvodů distribuován a jeho správnost je ověřena pomocí otisku SHA-256 uvedeného v sekci 3.2.8.2.

4.5.1 Konfigurační soubor

Konfigurační soubor využívá jazyk YAML, protože jde o lidsky čitelnou formu serializace dat [34]. YAML je rozšířenou formou konfigurace aplikací a je tak k dispozici velké množství knihoven pro řadu programovacích jazyků, což velmi usnadňuje implementaci.

Návrh výchozího konfiguračního souboru je vidět ve výpisu kódu 4.3. Součástí tohoto výpisu jsou všechna dostupná nastavení a námi zvolené výchozí hodnoty, přičemž řádky začínající znakem „#“ představují nastavení bez výchozí hodnoty. Obecně shlukuje nastavení využitá více operačními módy pro jeho zjednodušení. Příkladem může být cesta k CSV souboru, kterou potřebuje operační mód KeyLog a Reset.

4.5.1.1 Umístění

Protože aplikace není distribuována v podobě balíčků pro konkrétní GNU/Linux distribuce a očekáváme instalaci naším instalačním skriptem, výchozí cesta globálního konfiguračního souboru je `/usr/local/etc/zigtoucher/zigtoucher.yml`. Umístění výchozí uživatelské konfigurace definuje specifikace XDG Base Directory [35]. Třetí možností je určení vlastního konfiguračního souboru. Přirozeně je pořadí priorit těchto souborů, seřazeno od nejvýznamnější, vlastní konfigurační soubor, uživatelský výchozí a globální výchozí konfigurační soubor.

4.5.2 Přepínače

Jazyk YAML nám dovoluje specifikovat datovou strukturu ve formě stromu, čehož v našem konfiguračním souboru využíváme k logickému členění konfiguračních hodnot. Protože ale požadujeme, aby každé nastavení v konfiguračním souboru bylo dostupné také jako přepínač při spuštění aplikace, musíme tuto strukturu mapovat. Vezměme v potaz ukázkový výňatek z našeho konfiguračního souboru ve výpisu kódu 4.1. K mapování využíváme dlouhé přepínače a zanoření vyjadřujeme spojovníkem. Tedy kupříkladu nastavení `timeout` vyjádříme jako přepínač pomocí `--modes-timeout=3600` a nastavení `follow` jako přepínač `--modes-keylog-follow=False`.

```
modes:
  timeout: 3600
  keylog:
    follow: False
```

Výpis kódu 4.1 Výňatek z výchozího konfiguračního souboru aplikace zigtoucher

4.5.3 Parametry interaktivního režimu

Parametry operačních módů v interaktivním režimu jsou specifikovány pomocí párů `key=value`. Konkrétní parametry dostupné pro každý operační mód je možné automaticky doplnit klávesou Tab. Mapování v této situaci probíhá jinak. Vezměme opět v potaz výpis kódu 4.1. Zanoření v tomto režimu již nemá význam. Ponecháváme si pouze nejnižší úroveň, protože se toto nastavení týká pouze tohoto konkrétního běhu módu. Tedy

kupříkladu nastavení `timeout` vyjádříme jako pár `timeout=3600` a nastavení `follow` jako pár `follow=False`.

4.6 Perzistence dat

Cesty k souborům pro perzistentní uložení dat mohou obsahovat speciální hodnotu `<random>`. V takovém případě je vytvořen nový soubor v umístění určeném specifikací XDG Base Directory. Název souboru je vytvořen spojením data a času ve formátu ISO 8601 a náhodného řetězce znakem „_“.

4.6.1 Konfigurační soubor sítě

K uložení konfigurace ZigBee sítě využíváme opět YAML. Soubor obsahuje dvě části. První sekce ukládá informace potřebné k přidávání zařízení do sítě a jejich ovládání. Pro odesílání PDU obsahuje tato sekce krátkou adresu, EUI-64 a poslední hodnotu pořadového čísla PDU, kterou od nás síť viděla. Pro připojení zařízení do sítě zde také ukládáme krátkou adresu, kterou bychom měli přidělit novému zařízení. Druhá sekce nese informace o samotné síti, tedy EUI-64 sítě, její PAN ID, kanál, šifrovací klíč a seznam zařízení určených dvojicí krátká adresa a EUI-64. Návrh struktury tohoto souboru je vidět ve výpisu kódu 4.2.

```
---
Control:
  IEEEAddr: "ff:ff:ff:ff:ff:ff:ff:fd"
  NwkAddr: "0x0001"
  NextNwkAddr: "0x0003"
  FrameCounter: "0"
Network:
  ExtPanID: "ff:ff:ff:ff:ff:ff:ff:fe"
  PanID: "0x1111"
  Channel: "11"
  Key: "0xD0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF"
  Devices:
  - NwkAddr: "0x0002"
    IEEEAddr: "ff:ff:ff:ff:ff:ff:ff:ff"
```

Výpis kódu 4.2 Struktura souboru konfigurace sítě aplikace zigtoucher

4.6.2 CSV operačního módu KeyLog

Operační mód KeyLog může ukládat výsledky zachycených transakcí do souboru ve formátu CSV. Uložen je vždy klíč, kanál sítě, její EUI-64 a PAN ID v tomto pořadí. Uloženy jsou pouze transakce označené jako úspěšné. Tento soubor je možné využít k následné analýze nebo ruční tvorbě souborů konfigurace sítě.

4.6.3 CSV operačního módu Reset

Operační mód Reset je schopen ukládat EUI-64 zařízení obnovených do továrního nastavení a kanál, na kterém se transakce uskutečnila do souboru ve formátu CSV v tomto pořadí. Tento soubor lze využít jako zdroj skutečných EUI-64 pro použití v konfiguračních souborech sítě a jiných módech umožňujících odesílání PDU.

```
---
mode: "interactive"

log:
  verbose: True
  #file: "/cesta/k/log/souboru"

keys:
  touchlink:
    #master: "<klic>"
    certification: "0xC0C1C2C3C4C5C6C7C8C9CACBCCCDCECF"
  classical:
    certification: "0xD0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF"

transceiver:
  type: "SDR"
  #address: "<nahodna EUI-64>"
  wireshark: False
  #pcap: "/cesta/k/pcap/souboru"
  rxgain: 30.0
  txgain: 89.0
  samprate: 4000000
  sdr:
    flowgraph: "zigtoucher"
    timeout: 20

modes:
  timeout: 60
  results: True
  #pcap: "/cesta/k/pcap/souboru"
  #csv: "/cesta/k/csv/souboru"
  #nwkfile: "/cesta/k/yaml/nwk/souboru"
  channels:
    - 11
  #target: "<EUI-64>"
  reset:
    identena: True
    identdur: 0x0003
  keylog:
    follow: False
    nwkcreate: True
  sniff:
    #key: "<klic>"
    show: False
```

Výpis kódu 4.3 Výchozí konfigurační soubor aplikace zigtoucher

Kapitola 5

Implementace

V této kapitole si nejprve představíme konkrétní zvolené implementační prvky a knihovny. Následně uvedeme detaily realizace rádiové vrstvy, operačních módů a konfigurace naší aplikace.

5.1 Zvolené prvky

Aplikace je implementovaná v jazyce Python verze 3. Tato volba je založená převážně na dostupnosti knihoven pro realizaci funkcionalit z návrhu naší aplikace, mezi které patří práce se soubory ve formátu YAML a CSV. Nejdůležitější knihovnou vytvořenou v jazyce Python je však knihovna Scapy, představená v sekci 3.3.3. Kromě možnosti ukládat a zpracovávat soubory ve formátu PCAP, nám Scapy poskytuje schopnost konstruovat libovolné ZigBee PDU. K nastavení a ovládání SDR je využita již dříve představená knihovna GNU Radio. Zvoleným SDR pro vývoj je Ettus USRP B210. Protože navíc knihovna Scapy neumožňuje příjem ani vysílání pomocí SDR, k jejich propojení je zvolena knihovna scapy-radio. V implementaci využíváme řadu modulů jazyka Python a pokud není uvedeno jinak, vždy pochází ze standardní knihovny jazyka. V případě uvedených datových typů se jedná o typy určené modulem `typing`.

5.2 Rádiová vrstva

Aby bylo dodrženo navržené rozhraní rádia z diagramu 4.1, je každé rádio implementováno pomocí třídy, která dědí z abstraktní třídy `_Transceiver(abc.ABC)`. Pro tvorbu abstraktních tříd v jazyce Python využíváme modul `abc`. Každá metoda z diagramu je označena jako abstraktní, konkrétní třídy jsou tak nuceny je implementovat.

Třída `_Transceiver` zároveň poskytuje konstruktor, který umí obsloužit vytvoření okna programu Wireshark a přípravu zápisu surových PDU do souboru PCAP. Obě tyto činnosti obsluhuje třída `_Writer(abc.ABC)`. Tato abstraktní třída slouží k defi-

nici tříd pro perzistentní ukládání dat do souborů dle specifikace XDG Base Directory. Ve svém rozhraní definuje dvě abstraktní metody, `write()` pro zápis libovolných dat a `close()` pro uzavření všech zdrojů potřebných pro zápis. Mimoto je dostupná třída `Dummy(_Writer)`, kterou je možné nahradit libovolnou instanci třídy dědící z `_Writer` a představuje prázdnou operaci.

Okno programu Wireshark implementuje třída `Wireshark(_Writer)`. Ta otevře program Wireshark s využitím modulu `subprocess` jako podproces naší aplikace a na jeho standardní vstup přesměruje výstup třídy `RawPcapWriter`. `RawPcapWriter` obsahuje knihovna Scapy a umožňuje zápis PDU do souborů PCAP. Třída je instanciována s příznakem `sync=True`, a tudíž je zajištěn zápis v reálném čase. Samotné okno je dostupné v atributu `_wireshark`.

Zápis PDU do souboru je umožněn třídou `PCAP(_Writer)`. Ta opět využívá třídu `RawPcapWriter` a je dostupná v atributu `_writer`.

Třídy implementující konkrétní rádia mají k dispozici oba tyto atributy a je jejich povinností použít oba ve svých metodách `send()` a `recv()`.

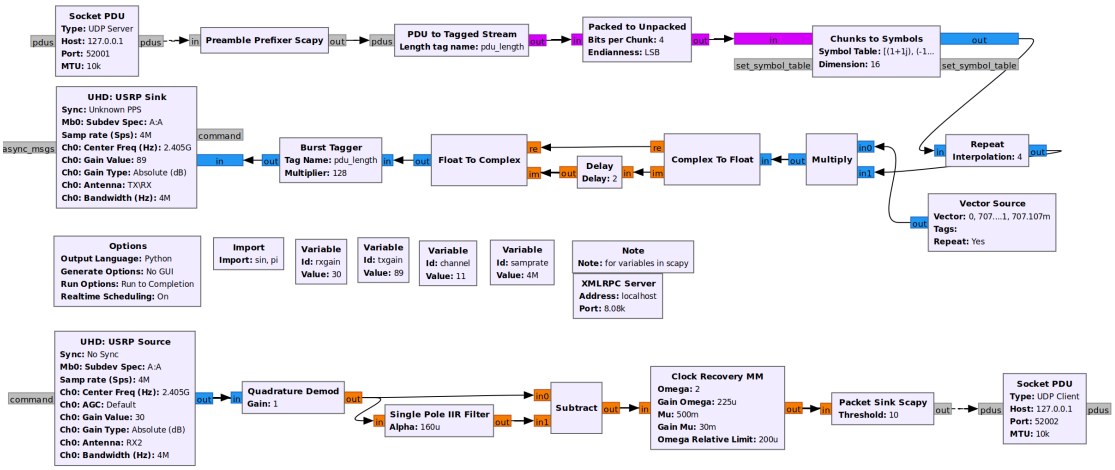
5.2.1 Rádio SDR

Stěžejním prvkem rádiové vrstvy SDR je knihovna scapy-radio. Scapy-radio rozšiřuje funkcionalitu knihovny Scapy o možnost odesílat PDU z SDR vestavěnou funkcí `send()` a přijímat PDU přidáním funkce `sniffradio()`, která je obdobou vestavěné funkce `sniff()`. Za předpokladu podpory ze strany GNU Radio flowgraphu ve formě spuštěného XMLRPC serveru, je možné flowgraph ovládat a měnit fyzické nastavení SDR úpravou hodnot proměnných ve flowgraphu.

Scapy-radio je šest let opuštěným projektem a k tomu obsahuje osm let starou fixní verzi knihovny Scapy. Navíc podporuje pouze GNU Radio verze 3.7, přičemž námi zvolená distribuce podporuje minimální verzi 3.8. Důsledkem je zásadní neaktuálnost těchto knihoven a chybějící opravy chyb. Nadto je realizace provedena v jazyce Python 2 a je nekompatibilní s naší aplikací. Z uvedených důvodů jsme provedli operaci fork a knihovnu aktualizovali pro GNU Radio 3.8 a Python 3. Více informací o aktualizaci je k dispozici v repozitáři Hipuranyhou/scapy-radio [36] na platformě GitHub.

Původní verze scapy-radio zahrnuje implementaci několika GNU Radio flowgraphů pro protokoly založené na IEEE 802.15.4. Jedním z nich je flowgraph pro ZigBee. Ten jsme použili jako základ naší implementace, avšak opět bylo nutné provést jeho aktualizaci pro GNU Radio 3.8. Flowgraph obsahuje bloky ze tří dříve představených rozšiřujících knihoven pro GNU Radio, gr-foo, gr-ieee802-15-4 a gr-zigbee. První dvě knihovny existují ve verzi pro GNU Radio 3.8, nicméně gr-zigbee ne. I v tomto případě jsme museli provést operaci fork a aktualizaci pro GNU Radio 3.8. Aktualizovaná verze gr-zigbee je dostupná v repozitáři Hipuranyhou/gr-zigbee [37] na platformě GitHub. Výsledný flowgraph je vyobrazen na obrázku 5.1.

Scapy-radio realizuje výměnu surových dat mezi GNU Radio a Scapy s využitím UDP socketu na síťovém rozhraní localhost. K ovládání vlastností SDR jsou v naší verzi scapy-radio dostupné dvě funkce využívající XMLRPC server standardní knihovny jazyka Python. Čtení proměnných flowgraphu zprostředkovává funkce `gnuradio_get_vars()` a změnu jejich hodnot dovoluje funkce `gnuradio_set_vars()`. GNU Radio 3.8 zahrnuje chybu, která v přeloženém souboru flowgraphu do jazyka Python, nazvaném `top_block.py`, generuje XMLRPC server pro Python 2. Instalační skript knihovny chybu v přeloženém souboru opravuje obměnou kódu pro Python 3.



■ **Obrázek 5.1** GNU Radio flowgraph aplikace zigtoucher, který umožňuje příjem a odesílání ZigBee PDU pomocí SDR Ettus USRP B210

5.2.2 Rádio PCAP

Analýzu již zachycených PDU realizujeme jako rádio, které simuluje příjem čtením souborů PCAP. Ke čtení souborů používáme třídu `RawPcapReader` knihovny Scapy, která je obdobou třídy `RawPcapWriter`. Zvolené řešení razantně zjednodušuje implementaci operačních módů, protože jejich chování se nemusí nikterak měnit. V případě použití tohoto typu rádia jsou vypnuty všechny operační módy vyžadující odesílání PDU, které toto rádio neposkytuje.

5.3 Operační módy

Operační módy implementujeme s využitím abstraktní třídy `Mode(abc.ABC)`, která splňuje rozhraní vyobrazené na obrázku 4.2. Metoda `start()` je označena jako abstraktní a poskytuje společné veřejné rozhraní, které startuje časovač pro automatické ukončení módu a volá abstraktní metodu `_act()`. Ta obsahuje logiku každého módu a kontroluje stav časovače.

5.3.1 Rozšíření knihovny Scapy

Knihovna Scapy obsahuje podporu pro normu IEEE 802.15.4 a ZigBee PDU, nicméně chybí podpora pro Light Link PDU. Tu bylo potřeba doplnit do fixní verze Scapy naší verze scapy-radio a částečně jsme ji přejali z jiné verze scapy-radio dostupné v repozitáři jkulskis/scapy-radio [38] na platformě GitHub.

5.3.2 Sniff

Operační mód Sniff realizujeme smyčkou naslouchající na PDU pomocí metody rádia `recv()`. V případě, že je poskytnut dešifrovací klíč, je nejprve každé PDU dešifrováno funkcí `decrypt()`. Funkce `decrypt()` a její protiklad `encrypt()` slouží k dešifrování a zašifrování libovolného ZigBee PDU s jakkoukoli úrovní ochrany. Jsou přejaty z projektu Killerbee a otestovány vektory dostupnými ve specifikaci [20]. Každé PDU je následně uloženo do souboru PCAP s využitím třídy `PCAP` obdobně jako u rádií. Pakliže je zapnuto zobrazení PDU, je využita metoda `show()` knihovny Scapy. Kanály cyklíme po každém přijatém PDU.

5.3.3 KeyLog

Operační mód KeyLog je opět implementován v jednoduché smyčce, která pouze naslouchá na příchozí PDU pomocí metody rádia `recv()`. Obecně drží Touchlink transakce v datovém typu `Dict`, kde klíčem je kanál, na kterém se transakce uskutečnila, a hodnotou je `Dict`. V tom je klíčem TID a hodnotou je třída `Transaction`. Každé PDU je Scapy metodou `haslayer()` otestováno, zda obsahuje Touchlink komunikaci. Toho je dosaženo hledáním vrstvy `ZigbeeZLLCommissioningCluster`. Jestliže neobsahuje Touchlink komunikaci nebo se nejedná o PDU související s filtrovanou EUI-64, je přeskočeno. V opačném případě je pro již existující TID zavolána metoda `update()` a pro neexistující vytvořena nová instance `Transaction`.

Třída `Transaction` má jako jeden ze svých atributů třídu `Network`, která nese veškeré informace o ZigBee síti popsané v sekci 4.6.1. Zařízení v síti jsou uložena v typu `Dict`, kde klíčem může být jak EUI-64, tak krátká adresa zařízení. `Transaction` používá jako klíč EUI-64, protože Touchlink nezná krátké adresy zařízení.

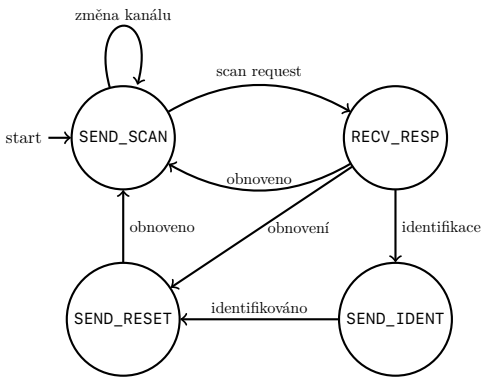
Metoda `update()` provede jednu z akcí na základě typu PDU. Pro PDU typu scan response si uloží RID pro dešifrování síťového klíče. Pakliže jde o PDU typu KeyTransport, učiní několik operací. První operací je uložení krátké adresy iniciátora 0x0001 a krátké adresy přidělené zařízení. Následně dešifruje síťový klíč funkcí `decrypt_nwk_key()` a jako poslední zapíše tuto událost do logu. V případě, že jde o odpověď na PDU typu KeyTransport a stav transakce byl úspěšný, zapíše tuto událost také do CSV souboru pomocí třídy `CSV(_Writer)` a volitelně vytvoří soubor konfigurace sítě pomocí třídy `NwkControl`.

Třída `NwkControl` nabízí statickou metodu `save_network()` pro serializaci libovolné instance třídy `Network` do souboru konfigurace sítě. Konkrétní proces serializace a uložení konfigurace sítě touto třídou je dále popsán v sekci 5.3.6.

Kanály cyklíme každých pět vteřin, přičemž tento časovač je obnoven při každé nově započaté transakci, abychom předešli změně kanálu během ní.

5.3.4 Reset

Operační mód Reset je uskutečněn jako konečný automat vyobrazený na obrázku 5.2. V úvodu odešleme PDU scan response a přejdeme do stavu `RECV_RESP`. Při nepřijetí odpovědi změníme stav zpět na `SEND_SCAN`. Jestliže nedostaneme odpověď na pět žádostí v řadě, přejdeme na další kanál a stav `SEND_SCAN`. U obdržené odpovědi zkontrolujeme zdrojovou EUI-64. Pro každý kanál si držíme typ `Dict`, kde kanál je klíčem a hodnotou je typ `Set` zdrojových EUI-64. Pro nevhodnou EUI-64 při využitém filtrování a známou EUI-64 změníme kanál a stav na `SEND_SCAN`. Pro známé adresy tuto akci provádíme, abychom předešli cyklickému obnovování jednoho zařízení. U neznámé EUI-64 přejdeme do stavu `SEND_RESET` nebo do stavu `SEND_IDENT` za podmínky, že je zapnuta funkce odesílání identify request PDU. Ze stavu `SEND_RESET` odešleme PDU pro obnovu do továrního nastavení a volitelně uložíme EUI-64 do CSV souboru pomocí třídy `CSV(_Writer)`. Jako poslední změníme kanál a stav na `SEND_SCAN`.



■ **Obrázek 5.2** Konečný automat operačního módu Reset aplikace zigtoucher, který definuje stavy využité pro příjem a odesílání PDU při obnově zařízení do továrního nastavení pomocí Touchlink

5.3.5 Scan

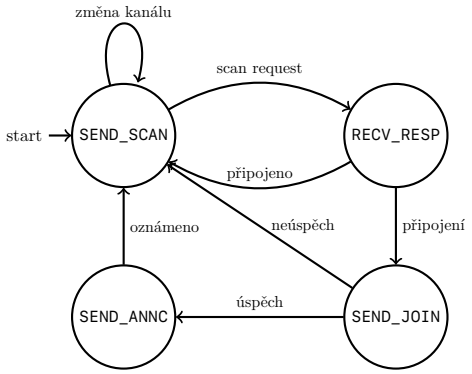
Operační mód Scan ve smyčce pouze vysílá scan request PDU a naslouchá na odpovědi typu scan response. Volitelně nalezená zařízení ukládá do CSV souboru třídou `CSV`.

5.3.6 Steal

Operační mód Steal je opět implementován pomocí konečného automatu, který je vyobrazen na obrázku 5.3. Nejprve je načten definovaný soubor konfigurace sítě. Pakliže soubor neexistuje, je vygenerována náhodná konfigurace. K načtení a deserializaci souborů YAML využívá třída `NwkControl` knihovnu PyYAML [39] a její funkci `safe_load()`. Ta převede soubor na základní datové typy jazyka Python. V našem případě očekáváme typ `Dict`, ve kterém pouze zkontrolujeme, že existují všechny očekávané klíče. Žádný z údajů uvedených v návrhu tohoto souboru není volitelný. Sekci nastavení sítě si třída `NwkControl` uloží v třídě `Network`, přičemž zařízení v tomto případě používají jako klíč krátkou adresu. Druhou sekci souboru si tato třída uloží do svých atributů.

Úvodní dva stavy konečného automatu se chovají stejně jako u módu Reset. Stav `RECV_RESP` přechází po přijetí PDU do stavu `SEND_JOIN`. Ten z třídy `NwkControl` získá krátkou adresu pro přidělení zařízení, odešle požadavek na připojení k síti a přejde do stavu `RECV_JOIN`. V tomto stavu čekáme na odpověď od zařízení pět vteřin. Pokud odpověď nepřijde nebo je neúspěšná, změníme kanál a stav na `SEND_SCAN`. V případě úspěchu posuneme počítadlo adres a přejdeme do stavu `SEND_ANNC`. V tomto stavu odešleme již běžné šifrované ZigBee PDU device announce, aby si nás nové zařízení vyplnilo do adresní tabulky. Nakonec změníme kanál a přejdeme do stavu `SEND_SCAN`.

Před ukončením módu je nutné konfiguraci sítě serializovat a uložit do souboru. K tomu je znovu využita knihovna PyYAML a funkce `safe_dump()`. Atributy tříd `NwkControl` a `Network` jsou jednoduše převedeny na datový typ `Dict` dle návrhu tohoto souboru a touto funkcí uloženy.



■ **Obrázek 5.3** Konečný automat operačního módu Steal aplikace zigtoucher, který definuje stavy využité pro příjem a odesílání PDU při připojení zařízení do ZigBee sítě pomocí Touchlink

5.3.7 Control

Operační mód Control začíná načtením definované konfigurace sítě. Jde o mód poskytující vlastní interaktivní uživatelské rozhraní obdobně jako samotná aplikace. Jeho im-

plementace je provedena pomocí třídy `Cmd` z modulu `cmd`, kterou si více představíme v sekci 5.4.

Jeho rozhraní nabízí pět dostupných příkazů:

List umožňuje zobrazit informace o načtené konfiguraci včetně všech zařízení;

Power umožňuje odeslat pokyn k vypnutí či zapnutí;

Level umožňuje snížit či zvýšit jas o libovolnou hodnotu;

Identify umožňuje odeslat pokyn na identifikační akci s volitelnou délkou trvání;

Transceiver implementuje ovládání rádia, jak je popsáno v sekci 5.3.8.

5.3.8 Transceiver

Operační mód Transceiver je realizován jako jednorázový příkaz, který pouze zavolá všechny metody měnící fyzické parametry rádia definované rozhraním 4.1. Všechny neuvedené parametry jsou nastaveny na hodnoty určené výchozí konfigurací.

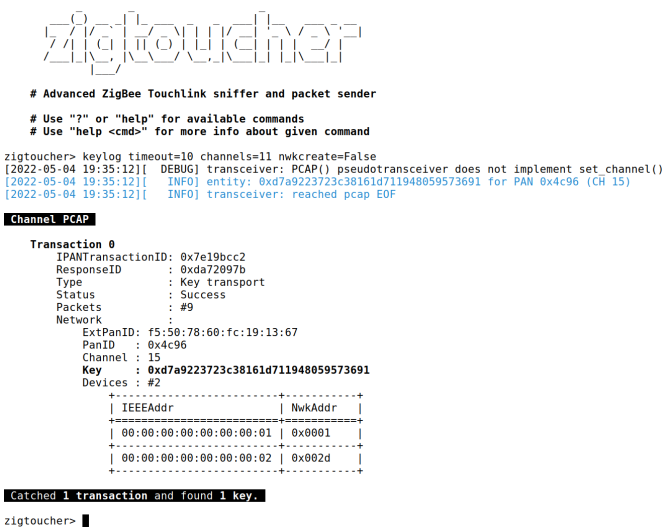
5.4 Ovládání

K implementaci interaktivního i neinteraktivního režimu ovládání aplikace využíváme třídu `CLI(cmd.Cmd)`. Modul `cmd` poskytuje jednoduchý framework pro tvorbu interaktivních uživatelských rozhraní pro příkazový řádek včetně podpory pro dokumentaci každého příkazu a automatické doplňování [40]. Ukázka uživatelského rozhraní aplikace je vyobrazena na obrázku 5.4.

5.4.1 Zpracování příkazu

Příkazy, které má třída `Cmd` zpracovávat, se definují přidáním metody `do_<cmd>()`, která je zavolána pro uživatelský vstup `<cmd>`. Zbylé znaky řádku jsou spojeny v jeden řetězec a předány této metodě jako argument. Tedy například vstup `keylog timeout=10` zavolá metodu `do_keylog("timeout=10")`. Zpracování těchto argumentů budeme dále řešit v sekci 5.5.1.

Každá implementace takové metody v našem případě volá metodu `__do_mode()`. Přestože tato metoda spouští konkrétní mód zavoláním metody `start()`, její hlavní funkcionalitou je možnost řetězení módů. Dle rozhraní módů z obrázku 4.2 vrací metoda `start()` další operační mód, kterým je v naší implementaci textový řetězec. Třída `Cmd` zpracovává uživatelský vstup z fronty řetězců v atributu `cmdqueue`. Jestliže je tato fronta prázdná, zeptá se na uživatelský vstup. V opačném případě zpracuje příkaz ve frontě. Řetězení módů tak simulujeme umístěním návratové hodnoty metody `start()` na konec této fronty.



■ **Obrázek 5.4** Ukázka grafického rozhraní výsledné aplikace zigtoucher vyobrazující dešifrování síťového klíče přeneseného metodou Touchlink ze souboru ve formátu PCAP

5.4.2 Manuálová stránka příkazu

Třída `Cmd` automaticky implementuje příkaz `help`, který vypíše všechny dostupné příkazy. Mimoto může každý příkaz implementovat metodu `help_<cmd>()`. Ta vypíše jeho manuálovou stránku a je zavolána pro uživatelský vstup `help <cmd>`.

5.4.3 Automatické doplnění příkazu

Třída `Cmd` samovolně implementuje automatické doplnění známých příkazů po stisknutí klávesy Tab. Tedy například pro vstup `key` automaticky doplní `keylog`. Každý příkaz ale může navíc implementovat metodu `complete_<cmd>()`. Tedy například pro vstup `keylog fol` zavolá metodu `complete_keylog("fol")`. Naše třída `CLI` obsahuje atribut držící množinu společných parametrů všech módů `__COMMON_OPTS` a množinu parametrů každého módu. Doplnění parametru realizujeme pomocí porovnání řětezce se začátkem každého parametru v konkrétní množině. V případě více možností nedojde k automatickému doplnění, ale k vypsání všech možností.

5.4.4 Neinteraktivní režim

Neinteraktivní režim je opět uskutečněn úpravou atributu `cmdqueue`. Zvolený mód vložíme do této fronty spolu s příkazem `exit` před spuštěním hlavní smyčky třídy `CLI`.

5.5 Konfigurace

Zpracování výchozí konfigurace, tedy konfiguračního souboru a přepínačů, probíhá společně při inicializaci aplikace pro dodržení definovaných priorit. Nejprve je vytvořen globální konfigurační typ `Dict` se strukturou definovanou výchozím konfiguračním souborem a hodnotou **`None`** každého klíče. Jako první jsou zpracovány přepínače. Ty jsou implementovány pomocí modulu `argparse` dle specifikace v sekci 4.5.2. Výsledkem je datový typ `Namespace`, ve kterém zkontrolujeme existenci každého klíče a v případě jeho existence a validní hodnoty přiřadíme do globální konfigurace. Následně je stejně jako u konfigurace sítě načten zvolený konfigurační soubor s využitím knihovny PyYAML. Výsledkem je znovu typ `Dict`, ve kterém zkontrolujeme existenci každého klíče. Pokud existuje a hodnota v globální konfiguraci je **`None`**, tuto hodnotu přiřadíme. Jako poslední krok provedeme průchod globální konfigurací a vyplníme prázdné klíče pomocí hodnot určených v návrhu výchozího konfiguračního souboru.

Konfigurace ve svém veřejném rozhraní neumožňuje nikterak měnit hodnoty v něm. Naproti tomu poskytuje funkci `get()` pro čtení libovolného nastavení. Vezmeme-li v potaz výpis kódu 4.1, hodnotu `timeout` bychom získali pomocí `get("modes.timeout")` a hodnotu `follow` pomocí `get("modes.keylog.follow")`.

5.5.1 Parametry interaktivního režimu

Pro zpracovaní parametrů interaktivního režimu implementujeme třídu `CLI._Parser`, která poskytuje metodu `parse()`. Ta na vstupu očekává řetězec párů `key=value`, který generuje třída `Cmd`, a názvy parametrů dostupných pro konkrétní mód, které dostane z množiny používané pro automatické doplňování. Při zpracování nejprve načte hodnoty těchto parametrů z výchozí konfigurace. Dále řádek pomocí modulu `shlex` rozdělí na jednotlivé páry a zpracuje jednotlivé klíče, jejichž hodnoty nahradí ty z výchozí konfigurace. Každý operační mód vždy dostane všechny jím používané parametry ve svém konstruktoru právě z této třídy. To je důležité, protože je tím jednak splněn požadavek na jednorázovost těchto parametrů v interaktivním režimu, dále požadavek na jejich prioritu a navíc žádný operační mód nemusí znát své výchozí hodnoty. Jediným místem jejich určení je zpracování výchozí konfigurace.

Kapitola 6

Testování

V této kapitole nejprve představíme prostředí, ve kterém byla aplikace testována. Dále uvedeme testované scénáře zaměřené na vzdálenost účinnosti a předložíme zjištěné výsledky spolu s jejich zhodnocením.

6.1 Prostředí

Testovacím strojem je notebook Apple MacBookPro11,4 s procesorem Intel i7-4770HQ, 16 GB paměti RAM a operačním systémem Arch Linux x86_64 s verzí jádra Linux 5.17.5-arch1-1. Aplikace běží ve virtualizovaném prostředí v operačním systému Ubuntu 20.04 LTS pomocí kombinace hypervisorů KVM a QEMU. Pro příjem a vysílání je zvoleno SDR Ettus USRP B210 spolu s dvojicí všesměrových antén určených pro frekvenci 2,4 GHz a ziskem 10 dBi. Toto SDR má frekvenční rozsah od 70 MHz do 6 GHz, plně duplexní provoz a propustnost 56 MHz v reálném čase [12]. Testovanou síť tvoří žárovka Philips LWB004 s verzí firmwaru 5.127.1.26420 a ovladač Philips 324131137411. Testovací zařízení jsou vyobrazena na obrázku 6.1.

6.2 Scénáře

Navrhujeme dva scénáře, jež testují efektivní vzdálenost pasivního odposlechu Touchlink transakcí a aktivního útoku v podobě obnovy zařízení do továrního nastavení naší aplikace. Kryptografické funkce, pro dešifrování síťového klíče předaného metodou Touchlink, byly otestovány během implementace využitím testovacích vektorů dostupných ve specifikaci [22]. Před každým měřením jsou žárovka i ovladač uvedeny do továrního nastavení metodou specifikovanou výrobcem v návodu každého z nich.

Obrázek 6.1 Testovací zařízení sestávající z žárovky Philips LWB004 s firmwarem verze 5.127.1.26420, dálkového ovladače Philips 324131137411 a SDR Ettus USRP B210 použitého při testování našeho nástroje zigtoucher pro bezpečnostní analýzu ZigBee Touchlink

6.2.1 Maximální vzdálenost

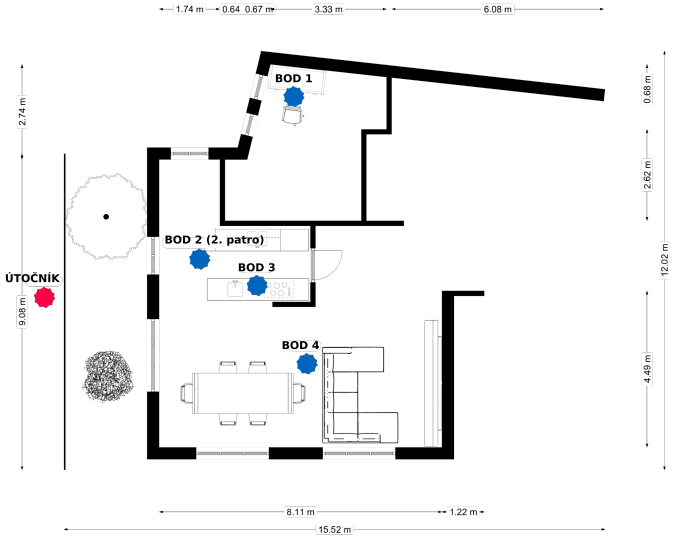
Cílem tohoto scénáře je otestovat maximální vzdálenost obou těchto útoků v laboratorních podmínkách, které představují volnou plochu ve venkovním prostředí bez jakýchkoli překážek mezi SDR a ZigBee sítí. Experiment je proveden postupnou úpravou zisku SDR. U testu odposlechu je to zisk přijímače. Test obnovy používá pro zisk přijímače nejlepší výsledek testu odposlechu a měněn je zisk vysílače. Vzorkovací frekvence je stanovena na čtyři miliony vzorků za vteřinu. Jediným testovaným ZigBee kanálem je kanál 11 frekvenčního pásma 2,4 GHz. Pro každou úroveň zisku je provedeno pět měření, přičemž maximální dosažená vzdálenost je započítána pouze při úspěchu všech pěti měření.

6.2.2 Reálné podmínky

Scénář reálných podmínek má za cíl otestovat praktickou použitelnost aplikace z pohledu útočníka stojícího před rodinným domem. Testována budou čtyři místa vyznačená na půdorysu vyobrazeném na obrázku 6.2. Budova využitá pro experiment má 350 milimetrů silné cihlové zdi a okna s trojsklem. Použitá nastavení SDR budou určena nejlepším výsledkem scénáře měření maximální vzdálenosti. V případě tohoto scénáře je opět provedeno pět měření a je zhodnocen podíl jejich úspěšnosti.

6.3 Výsledky

Na základě definovaných scénářů jsme provedli měření.



Obrázek 6.2 Půdorys skutečné testovací budovy má 350 milimetrů silné cihlové zdi a okna s trojsklem. Vyznačeny jsou body ZigBee sítě a útočníka. Tento scénář slouží k otestování praktické použitelnosti odposlechu transakcí Touchlink aplikací zigtoucher z pohledu útočníka

6.3.1 Maximální vzdálenost

Výsledky měření maximální vzdálenosti pro odposlech přenosu síťového klíče jsou zaneseny v tabulce 6.1. Nejlepšího výsledku jsme dosáhli při zisku přijímače 30 dB a výsledkem je vzdálenost 29 metrů. Stejné nastavení jsme následně použili pro přijímač při testu obnovení zařízení do továrního nastavení, který potřebuje jak přijímat, tak vysílat.

Výsledky testu obnovení do továrního nastavení jsou zaneseny v tabulce 6.2. Nejlepšího výsledku jsme dosáhli při maximálním zisku vysílače 89 dB a výsledkem je vzdálenost 1,7 metrů.

Tabulka 6.1 Výsledky měření vzdálenosti odposlechu Touchlink transakce v závislosti na zisku přijímače (RX)

Zisk RX (dB)	Vzdálenost (m)
0	0,1
15	13,0
30	29,0
45	21,0
60	10,0
76	0,3

Tabulka 6.2 Výsledky měření vzdálenosti obnovení zařízení pomocí Touchlink v závislosti na zisku vysílače (TX)

Zisk TX (dB)	Vzdálenost (m)
0	0,0
15	0,0
30	0,0
50	0,3
70	1,7
89	1,7

6.3.2 Reálné podmínky

Test reálných podmínek měl zisk nastaven na hodnotu 30 dB pro přijímač a 89 dB pro vysílač. Hodnoty jsme přejali z nejlepších výsledků testu maximální vzdálenosti.

Výsledky testu odposlechu jsou zaneseny v tabulce 6.3. V případě bodů 1 a 2 nebyl úspěšný žádný pokus o dešifrování klíče, naopak u bodů 3 a 4 byly úspěšné všechny.

Tabulka 6.3 Výsledky měření odposlechu Touchlink transakce v reálných podmínkách

Bod	Úspěch/Celkově
1	0/5
2	0/5
3	5/5
4	5/5

Žádný pokus o obnovení zařízení do továrního nastavení nebyl úspěšný.

6.4 Zhodnocení

Výsledky testů ukázaly malou praktickou použitelnost aplikace. Přestože je v laboratorních podmínkách možno úspěšně odposlechnout přenos síťového klíče na vzdálenost 29 metrů, testování v reálných podmínkách ukázalo sporadickou úspěšnost právě v polovině testů. Praktickou použitelnost navíc snižuje nemožnost vynutit provedení Touchlink transakce uživatelem obnovením zařízení do továrního nastavení. To má za důsledek dlouhé čekání na Touchlink transakci, protože je rozumným předpokladem, že cílová síť má již všechna zařízení připojena.

Velmi krátká vzdálenost pro možnost provedení obnovy do továrního nastavení je dle definice funkcionality Touchlink předpokládanou vlastností. Jistě je výsledkem správné implementace a použití funkcionalit pro přesné měření vzdálenosti poskytovaných normou IEEE 802.15.4 ze strany výrobce. Je rovněž nutno zmínit přínosy specifikace ZigBee 3.0, která přináší částečnou možnost obrany proti těmto útokům. Jak jsme uvedli v sekci 3.2.8, Touchlink je volitelným rozšířením této specifikace, a přestože ho výrobci stále implementují, zapříčinila tato změna úpravu chování sítí, které tuto metodu nepoužívají jako výchozí. Spotřebitelé a administrátoři sítí podporující ZigBee 3.0 mohou kromě toho tomuto útoku úplně předejít, jestliže využijí centralizovaného modelu zabezpečení a instalačních kódů představených v sekci 3.2.5.

Kapitola 7

Závěr

Cílem bakalářské práce byla analýza zabezpečení přenosu dat protokolem ZigBee, jeho funkcionality Touchlink a následné využití zjištěných zranitelností k vytvoření aplikace umožňující útok na tento protokol.

V analytické části práce jsme se v prvé řadě zaměřili na úvod do technologie internetu věcí, softwarově definovaného rádia a na výzkum rádiového přenosu protokolu ZigBee s využitím normy IEEE 802.15.4. Navázali jsme představením analýzy samotného protokolu ZigBee, která obsahovala rozbor struktury přenášených dat a převážně bezpečnostních vlastností poskytovaných k zajištění důvěrnosti a integrity těchto dat. Představili jsme řadu zranitelností týkajících se přenosu šifrovacích klíčů v protokolu ZigBee a jeho funkcionalitě Touchlink. Nejvýznamnější představenou zranitelností byl únik statického šifrovacího klíče, který Touchlink využívá.

V praktické části práce jsme definovali požadavky na aplikaci pro příkazový řádek, která využívá uniklého šifrovacího klíče k útoku na Touchlink a je stěžejním výsledkem této práce. Popis implementace zahrnoval vybrané stavební bloky, kterými je programovací jazyk Python, knihovna Scapy využitá k tvorbě PDU a knihovna GNU Radio, jež je využita v rádiové vrstvě aplikace pro ovládání SDR. Tím bylo splněno zadání bakalářské práce. Aplikace byla zveřejněna v repozitáři Hipuranyhou/zigtoucher [41] na platformě GitHub. Její vývoj plánujeme spolu s kolegou Ing. Jiřím Dostálem, Ph.D. zveřejnit ve vědeckém článku, jehož pracovní verze je k vidění v příloze B. Zároveň jsme uvedli vedlejší produkt bakalářské práce, kterým je aktualizace knihoven scapy-radio a gr-zigbee pro verzi 3.8 knihovny GNU Radio.

Testováním aplikace jsme odhalili její teoretickou efektivní vzdálenost v řádu nízkých desítek metrů v případě odposlechu komunikace využívající Touchlink. Dále jsme otestovali praktickou použitelnost aplikace v simulovaných reálných scénářích. Závěrečným zhodnocením všech těchto testů jsme došli k výsledku, že přestože je tento útok proveditelný, krátká efektivní vzdálenost činí jeho praktickou použitelnost omezenou.

Nejvýznamnějším plánem pro budoucí vývoj aplikace je implementace a otestování

méně obecného rádia, které se specializuje na technologii IEEE 802.15.4. Důvodem je menší komplexita takového řešení a uživatelská přívětivost. Druhým plánovaným rozšířením je odposlech a dešifrování klíčů přenesených pomocí jiných metod připojení zařízení do ZigBee sítě. Pro tuto problematiku již existují jiné nástroje, šlo by tak pouze o shromáždění těchto funkcionalit v jedné aplikaci. Díky zveřejnění aplikace na platformě GitHub může kdokoliv z řad obdorné veřejnosti aplikaci ověřit a přispět k jejímu vývoji.

Bibliografie

1. ZIMMERMANN, Hubert. OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications* [online]. 1980, roč. 28, č. 4, s. 425–432 [cit. 2022-04-09]. ISSN 0096-2244. Dostupné z DOI: 10.1109/TCOM.1980.1094702.
2. LI, Yadong; LI, Danlan; CUI, Wenqiang; ZHANG, Rui. Research based on OSI model. In: *2011 IEEE 3rd International Conference on Communication Software and Networks* [online]. IEEE, 2011, s. 554–557 [cit. 2022-04-09]. ISBN 978-1-61284-485-5. Dostupné z DOI: 10.1109/ICCSN.2011.6014631.
3. IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture. *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)* [online]. 2014, s. 1–74 [cit. 2022-04-10]. Dostupné z DOI: 10.1109/IEEESTD.2014.6847097.
4. BUYYA, Rajkumar; DASTJERDI, Amir Vahid. *Internet of Things: Principles and Paradigms* [online]. Cambridge (Massachuttes): Elsevier, 2016 [cit. 2022-04-28]. ISBN 9780128093474. Dostupné z: https://ebookcentral.proquest.com/lib/cvut/detail.action?docID=4530251.
5. SINHA, Satyajit. *State of IoT 2021: Number of connected IoT devices growing 9% to 12.3 B* [online]. IoT Analytics, 2021-09-22 [cit. 2022-04-29]. Dostupné z: https://iot-analytics.com/number-connected-iot-devices.
6. ORACLE. *What Is the Internet of Things (IoT)?* [online]. Oracle, 2022 [cit. 2022-04-29]. Dostupné z: https://www.oracle.com/internet-of-things/what-is-iot.
7. TOURNIER, Jonathan; LESUEUR, François; MOUËL, Frédéric Le; GUYON, Laurent; BEN-HASSINE, Hicham. A survey of IoT protocols and their security issues through the lens of a generic IoT stack. *Internet of Things*. 2021, roč. 16, s. 100264. ISSN 2542-6605. Dostupné z DOI: https://doi.org/10.1016/j.iot.2020.100264.

8. IEEE Standard for Low-Rate Wireless Networks. *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)* [online]. 2020, s. 1–800 [cit. 2022-04-10]. Dostupné z DOI: 10.1109/IEEESTD.2020.9144691.

9. O'MAHONY, George D.; HARRIS, Philip J.; MURPHY, Colin C. Analyzing using Software Defined Radios as Wireless Sensor Network Inspection and Testing Devices: An Internet of Things Penetration Testing Perspective. In: *2020 Global Internet of Things Summit (GIoTS)*. 2020, s. 1–6. Dostupné z DOI: 10.1109/GIOTS49054.2020.9119606.

10. CHAPPELL, William J.; NAGLICH, Eric J.; MAXEY, Christopher; GUYETTE, Andrew C. Putting the Radio in "Software-Defined Radio": Hardware Developments for Adaptable RF Systems. *Proceedings of the IEEE*. 2014, roč. 102, č. 3, s. 307–320. Dostupné z DOI: 10.1109/JPROC.2014.2298491.

11. RTL-SDR.COM. *About RTL-SDR* [online]. RTL-SDR.COM [cit. 2022-04-29]. Dostupné z: https://www.rtl-sdr.com/about-rtl-sdr.

12. NI. *USRP B210 USB Software Defined Radio (SDR)* [online]. NI, 2022 [cit. 2022-04-29]. Dostupné z: https://www.ettus.com/all-products/ub210-kit.

13. GNU RADIO PROJECT. *What is GNU Radio?* [online]. GNU Radio project, 2020-05-25 [cit. 2022-04-29]. Dostupné z: https://wiki.gnuradio.org/index.php?title=What_is_GNU_Radio%3F. Licence: CC BY-SA 3.0 dostupná z https://creativecommons.org/licenses/by-sa/3.0.

14. HERSENT, Olivier; BOSWARTHICK, David; ELLOUMI, Omar. *The internet of things: applications to the smart grid and building automation*. 2. vyd. Hoboken: John Wiley & Sons, 2012. ISBN 9781119994350.

15. NXP SEMICONDUCTORS. ZigBee 3.0 Stack User Guide [online]. 2018, č. 1.5, s. 1–508 [cit. 2022-04-11]. Dostupné z: https://www.nxp.com/docs/en/user-guide/JN-UG-3113.pdf.

16. JJGARCIA.TSC. *Český překlad diagramu CSMA-CA s použitím RTS/CTS Exchange, SVG verze* [online]. 2014-01-04. [cit. 2022-04-11]. Dostupné z: https://commons.wikimedia.org/wiki/File:CSMA_CA_cs.svg. Upraveno odstraněním RTS/CTS Exchange. Licence: CC BY 3.0 dostupná z https://creativecommons.org/licenses/by/3.0.

17. MCGREW, David. *An Interface and Algorithms for Authenticated Encryption* [online]. RFC Editor, 2008 [cit. 2022-04-10]. Request for Comments, č. 5116. Dostupné z DOI: 10.17487/RFC5116.

18. FENG, Bin; QI, De-yu; HAIWEN, Han. Parallel and Multiplex Architecture of AES-CCM Coprocessor Implementation for IEEE 802.15.4. In: *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies* [online]. IEEE, 2013, s. 149–153 [cit. 2022-04-10]. ISBN 978-1-4799-2141-6. Dostupné z DOI: 10.1109/EIDWT.2013.31.

19. LI, Hongwei; JIA, Zhongning; XUE, Xiaofeng. Application and Analysis of ZigBee Security Services Specification. In: *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing* [online]. IEEE, 2010, s. 494–497 [cit. 2022-04-10]. ISBN 978-1-4244-6597-2. Dostupné z DOI: 10.1109/NSWCTC.2010.261.

20. ZIGBEE ALLIANCE. zigbee Specification [online]. 2017, roč. 22, s. 1–574 [cit. 2022-04-11]. Dostupné z: https://csa-iot.org/developer-resource/specifications-download-request.

21. FAN, Xueqi; SUSAN, Fransisca; LONG, William; LI, Shangyan. *Security Analysis of Zigbee* [online]. Massachusetts Institute of Technology, 2017-05-18 [cit. 2022-04-11]. Dostupné z: https://courses.csail.mit.edu/6.857/2017/project/17.pdf.

22. ZIGBEE ALLIANCE. Cluster Library Specification [online]. 2019, roč. 8, s. 1–1213 [cit. 2022-04-30]. Dostupné z: https://csa-iot.org/developer-resource/specifications-download-request.

23. HOLDING, Signify. *Zigbee 3.0 support in Hue ecosystem* [online]. Signify Holding, 2019 [cit. 2022-04-29]. Dostupné z: https://developers.meethue.com/zigbee-3-0-support-in-hue-ecosystem.

24. SYSTEMS, Inter IKEA. *Getting started with TRÅDFRI smart lighting* [online]. Inter IKEA Systems, 2022 [cit. 2022-04-29]. Dostupné z: https://www.ikea.com/us/en/customer-service/product-support/smart-lighting/getting-started-with-smart-lighting-pub2721e8b9.

25. MORGNER, Philipp; MATTEJAT, Stephan; BENENSON, Zinaida; MÜLLER, Christian; ARMKNECHT, Frederik. Insecure to the touch. In: *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks* [online]. New York, NY, USA: ACM, 2017, s. 230–240 [cit. 2022-04-11]. ISBN 9781450350846. Dostupné z DOI: 10.1145/3098243.3098254.

26. BASTIBL. *gr-foo* [online]. GitHub, 2021 [cit. 2022-04-29]. Dostupné z: https://github.com/bastibl/gr-foo.

27. BASTIBL. *gr-ieee802-15-4* [online]. GitHub, 2021 [cit. 2022-04-29]. Dostupné z: https://github.com/bastibl/gr-ieee802-15-4.

28. GEFA. *gr-zigbee* [online]. GitHub, 2021 [cit. 2022-04-29]. Dostupné z: https://github.com/gefa/gr-zigbee.

29. WIRESHARK. *Wireshark* [online]. Wireshark, 2022 [cit. 2022-04-29]. Dostupné z: https://www.wireshark.org.

30. RIVERLOOPSEC. *killerbee* [online]. GitHub, 2022 [cit. 2022-04-29]. Dostupné z: https://github.com/riverloopsec/killerbee.

31. BIONDI, Philippe; SCAPY COMMUNITY. *Scapy* [online]. Scapy, 2021 [cit. 2022-04-29]. Dostupné z: https://scapy.net.

32. BASTILLERESEARCH. *scapy-radio* [online]. GitHub, 2016 [cit. 2022-04-29]. Dostupné z: https://github.com/BastilleResearch/scapy-radio.

33. IOTSEC. *Z3sec* [online]. GitHub, 2017 [cit. 2022-04-30]. Dostupné z: https://github.com/IoTsec/Z3sec.

34. YAML LANGUAGE DEVELOPMENT TEAM. *YAML Ain't Markup Language (YAML™) version 1.2* [online]. YAML Language Development Team, 2021-10-01 [cit. 2022-04-29]. Dostupné z: https://yaml.org/spec/1.2.2.

35. WALDO, Bastian. *XDG Base Directory Specification* [online]. [cit. 2022-04-22]. Dostupné z: https://specifications.freedesktop.org/basedir-spec/basedir-spec-0.6.html.

36. HIPURANYHOU. *scapy-radio* [online]. GitHub, 2022 [cit. 2022-05-05]. Dostupné z: https://github.com/Hipuranyhou/scapy-radio.

37. HIPURANYHOU. *gr-zigbee* [online]. GitHub, 2022 [cit. 2022-05-05]. Dostupné z: https://github.com/Hipuranyhou/gr-zigbee.

38. JKULSKIS. *scapy-radio* [online]. GitHub, 2019 [cit. 2022-04-29]. Dostupné z: https://github.com/jkulskis/scapy-radio.

39. YAML. *pyyaml* [online]. GitHub, 2021 [cit. 2022-04-30]. Dostupné z: https://github.com/yaml/pyyaml.

40. PYTHON SOFTWARE FOUNDATION. *cmd — Support for line-oriented command interpreters* [online]. Python Software Foundation, 2022-04-26 [cit. 2022-04-30]. Dostupné z: https://docs.python.org/3/library/cmd.html.

41. HIPURANYHOU. *zigtoucher* [online]. GitHub, 2022 [cit. 2022-05-09]. Dostupné z: https://github.com/Hipuranyhou/zigtoucher.

42. CANONICAL. *Install Ubuntu desktop | Ubuntu* [online]. Canonical, 2022 [cit. 2022-05-07]. Dostupné z: https://ubuntu.com/tutorials/install-ubuntu-desktop#1-overview.

Příloha A

Uživatelská příručka

A.1 Instalace

Jediným předpokladem pro instalaci aplikace zigtoucher je stroj s minimální instalací operačního systému Ubuntu 20.04 LTS. Postup instalace Ubuntu je popsán ve zdroji [42].

Následně je nutné získat zdrojové kódy aplikace. Ty můžeme zkopírovat ze složky `zigtoucher` na přiloženém DVD nebo naklonovat repozitář Hipuranyhou/zigtoucher [41] dostupný na platformě Github pomocí příkazu:

```
$ git clone https://github.com/Hipuranyhou/zigtoucher.git
```

Instalaci můžeme následně provést s využitím následujících dvou příkazů:

```
$ cd zigtoucher
$ ./setup-ubuntu-20.04.4-desktop-amd64.sh
```

A.2 Použití

Aplikaci je doporučeno spouštět spolu s přípravou SDR sledem příkazů:

```
$ uhd_usrp_probe
$ zigtoucher
```

Podrobné informace o konfiguraci aplikace jsou dostupné na wiki stránce, která je přístupná z repozitáře aplikace.

Pakliže je spuštěn interaktivní mód, který je výchozím nastavením, je možné využít interních manuálových stránek k získání informací o použití. Tyto stránky jsou dostupné věstavěným příkazem `help`. Navíc je možné obdobné informace zobrazit při spuštění přepínačem `--help`.

```
    ___(_) __ _| |_ ___   _  ___| |__   ___ _ __
   |_  / |/ _` | __/ _ \ | |/ __| '_ \ / _ \ '__|
    / /| | (_| | || (_) | |_| (__| | | |  __/ |
   /___|_|\__, |\__\___/ \__,_|\___|_| |_|\___|_|
          |___/

    # Advanced ZigBee Touchlink sniffer and packet sender

    # Use "?" or "help" for available commands
    # Use "help <cmd>" for more info about given command

zigtoucher> help

    # Commands:
        control      => Control lighting devices in specified network
        exit         => Exit zigtoucher
        export       => Export specified KeyLog transaction packets into PCAP file
        keylog       => Sniff Touchlink key transactions and decrypt them
        reset        => Reset Touchlink enabled devices in range to factory default
        scan         => Scan for Touchlink enabled devices
        sniff        => Sniff ZigBee packets and optionaly decrypt them
        steal        => Steal Touchlink enabled devices in range to specified network
        transceiver  => Set physical transceiver settings or reset to defaults

zigtoucher> help keylog

    # What:
        Sniff Touchlink key transactions and decrypt them

    # Options:
        [timeout   ] => Mode timeout in seconds
        [results   ] => Print detailed run results
        [nwkcreate ] => Create NWK files
        [csv       ] => Path to output CSV file or "<random>"
        [follow    ] => Continue in sniff mode after first catched key
        [channels  ] => List of channels to cycle or {all|primary|secondary}

    # Example:
        keylog timeout=20 follow=Off results=Yes

zigtoucher> 
```

■ **Obrázek A.1** Ukázka interních manuálových stránek výsledné aplikace zigtoucher

A.3 Provedení útoku

K provedení útoku potřebujeme síť podporující Touchlink, SDR Ettus USRP B210 a stroj s nainstalovanou aplikací zigtoucher. Síť může sestávat například z žárovky Philips LWB004 a ovladače Philips 324131137411 v továrním nastavení.

Prvním krokem je příprava SDR Ettus USRP B210. Antény pro pásmo 2,4 GHz zašroubujeme do zdířek `TX/RX` a `RX2` v sekci `RF A`. Následně připojíme SDR k počítači pomocí rozhraní USB. V konfiguračním souboru `zigtoucher.yml` je nutné vyplnit Touchlink master klíč, který není z bezpečnostních důvodů distribuován a je nutné ho nalézt například na platformě Twitter. Nyní spustíme aplikaci zigtoucher doporučeným způsobem popsaným v sekci A.2.

K nalezení zařízení podporující Touchlink zavoláme příkaz:

```
zigtoucher> scan channels=all timeout=10
[2022-05-09 22:17:10][   INFO] scan: <eui-64> on channel 11
```

■ **Výpis kódu A.1** Nalezení Touchlink zařízení v okolí pomocí aplikace zigtoucher

Následně můžeme odposlechnout připojení zařízení do sítě pomocí ovladače. Připojení vyvoláme podržením tlačítka se znakem „|“ na ovladači, který se nachází přibližně jeden metr od žárovky, a odposlechneme příkazem:

```
zigtoucher> keylog target=<eui-64> channels=11 timeout=15
[2022-05-09 22:21:11][   INFO] entity: <key> for PAN 0x6246 (CH 11)
[2022-05-09 22:21:11][   INFO] nwkcontrol: writing NWK /tmp/nwk.yml
```

Výpis kódu A.2 Odposlech Touchlink transakce pomocí aplikace zigtoucher

Zařízení můžeme ovládat příkazem:

```
zigtoucher> control nwkfile=/tmp/nwk.yml
zigtoucher|control> list
```

Výpis kódu A.3 Ovládání žárovky pomocí aplikace zigtoucher

Obnovu zařízení do továrního nastavení provedeme pomocí příkazu:

```
zigtoucher> reset channels=11 timeout=10 target=<eui-64>
[2022-05-09 22:24:56][   INFO] reset: <eui-64> on channel 11
```

Výpis kódu A.4 Obnovení zařízení pomocí aplikace zigtoucher

Pro odposlech komunikace sítě můžeme spustit příkaz `keylog` s přepínačem `follow`.

Příloha B

Připravená publikace

Práce byla také shrnuta jako vědecký článek připravený k publikaci. V příloze na následující straně naleznete pracovní verzi.

Zigtoucher: Revisiting Exploits of ZigBee Touchlink

Jakub Šatoplet
Faculty of Information Technology
Czech technical university in Prague
Prague, Czech Republic
satopja2@fit.cvut.cz

Jiří Dostál
Faculty of Information Technology
Czech technical university in Prague
Prague, Czech Republic
jiri.dostal@fit.cvut.cz

***Abstract*—This work explores the security of the ZigBee IoT protocol and its functionality, Touchlink. Touchlink is one of the ways new devices connect to a network. It is an optional extension of the latest ZigBee 3.0 specification, released in 2016 and developed as a convenience for users. Unfortunately, its security was breached in 2015 by a static keying material leak. This vulnerability is well documented, and the goal of this paper is to investigate its state in 2022. Since this feature is insecure by design and cannot be patched because of backwards compatibility, our findings show that its inclusion by manufacturers still poses a risk to ZigBee 3.0 networks. The critical result of this work is a command-line application written in Python, Scapy and GNU Radio to enable the decryption of network keys transmitted during communication using Touchlink. Although testing our application revealed little practical applicability of this attack and a theoretical maximum distance in the low tens of meters for eavesdropping, Touchlink still poses a risk and should not be deployed by manufacturers.**

***Index Terms*—ZigBee, Touchlink, SDR, Scapy, sniffing, IoT**

I. Introduction

Arguably one of the most critical technologies of the 21st century is the Internet of Things (IoT). IoT is often described as a network of objects with embedded sensors and technologies to connect and exchange data using the Internet or other network technologies. Everyday objects are part of the IoT, from tire pressure sensors to home appliances like coffee makers and even people wearing bionic prostheses or pacemakers. Although the idea of such systems has been around for a long time, their practical deployment has only been made possible by recent technological advances. These advances include ubiquitous network connectivity, cloud-based technologies, and, most notably, the development of sufficiently accurate, small and energy-efficient sensors and microcontrollers.

This ubiquitous technology undoubtedly makes everyone's daily life much easier by analysing and using the data collected. For example, intelligent door locks ensure we never forget to lock our homes. However, by this example, it is clear that the data collected are sensitive. Therefore, a natural consequence is the massive interest of attackers in finding and exploiting vulnerabilities in these systems. Securing these systems is thus an absolute priority. One of the most implemented short-range IoT protocols promising safety is an open standard, ZigBee, developed by the ZigBee Alliance.

A. Software defined radio

However, attacks on wireless protocols increase also thanks to enabling technologies such as software-defined radio (SDR). SDR generally represents a system in which the hardware elements for digital signal processing (DSP) are implemented using software, most often on field-programmable gate arrays (FPGAs). This fact allows for easy and fast reconfiguration of the SDR to different radio protocols. [1]

The broader use of this technology is also due to the wide range of available hardware. Products such as RTL-SDR provide an easy entry point for radio amateurs, primarily due to their price. In contrast, products such as the Ettus USRP B210 offer a high-frequency range and sampling rate. These properties provide much higher sensitivity and can be used, for example, to create a custom GSM base station [2].

II. ZigBee

A. Architecture

The ZigBee protocol consists of four layers. The IEEE 802.15.4 standard defines the physical and link layers, and nearly all implementations use its 2.4 GHz ISM frequency band. ZigBee itself defines the network layer (NWK) and a skeleton for the application layer. [3].

The most crucial feature implemented on the network layer is support for mesh networks with routing based on the AODV algorithm. To achieve this, ZigBee defines three node types. The first one is the ZigBee end device (ZE), typically a sensor which sleeps most of the time. These nodes can exist only as a leaf in the network topology and communicate with its parent. The second is the ZigBee router (ZR), which can route messages to other nodes. It is not expected to sleep and can become a parent of other nodes. The last node type is ZigBee coordinator (ZC). It behaves as ZR and can create new networks. Any network has precisely one ZC. [3]

On the application layer, the most notable feature is the definition of ZigBee Application Profiles. These profiles specify groups of related mandatory and optional clusters for specific applications, for example controlling lights. Clusters represent a set of commands and attributes used to perform some specific action. An example of such action can be changing the colour of an RGB LED light. [3]

B. Security features

Security is optional in any ZigBee network and available at both network and application layers. The chosen cryptographic algorithm is AES-CCM* with a 128-bit key. The network key is used to secure broadcast PDUs. Two nodes can also establish a link key for two-way communication utilising the SKKE schema. The choice of this algorithm provides the possibility to use hardware accelerators in IEEE 802.15.4 compliant devices. Furthermore, AES-CCM* provides confidentiality of the payload transmitted and integrity of the given layer by generating Message Integrity Code (MIC). It is possible to disable any or both of these features. In addition, each PDU gets assigned a 32-bit sequence number to prevent replay attacks if security is enabled. [3]

ZigBee defines two security models. The first one is the distributed model, typically used in residential applications. In this model, any ZR or ZC can connect new devices to a network and distribute the network key, which can even be transferred unencrypted. On the other hand, the centralised security model introduces a new node type, the trust centre (TC). ZC is typically TC, and only it can connect new nodes to a network and distribute the network key encrypted. TC also generates link keys for any two nodes. The initial key transfer can be secured by a preconfigured link key which the TC receives by some out-of-band method. [3]

A notable mandatory feature introduced in ZigBee 3.0 is the obligation of every device to carry an installation code. The installation code is a 128-bit random number with a CRC-16 specified on the device. TC can receive it using some out-of-band method before connecting the new device and derive a preconfigured link key using the Matyas-Meyer-Oseas (MMO) compression function. [3]

C. Touchlink

In general, Touchlink is a method of finding devices in the vicinity based on measured signal strength, and therefore these devices must be very close together. It allows essential control of devices, creating new ZigBee networks without a coordinator and connecting devices to existing networks. It was defined in the ZigBee Light Link application profile and is now an optional part of the ZigBee 3.0 specification. The main reason behind its existence is the need for manufacturers to sell introductory packages into their ecosystem. One such example is a package consisting of one light bulb and remote control. These devices cannot act as a ZC, which is needed in traditional ZigBee networks. [4]

The communication process begins with a scan request PDU sent by the initiator of a Touchlink transaction, which is tagged with a random 32-bit transaction identifier (TID). Next, each device close enough to the initiator sends back a scan response PDU that contains the received TID and response identifier (RID). From this point on, all messages from the initiator contain the given TID and RID, and messages from the target device contain the given TID. The initiator can then send an identify request PDU instructing the device to perform some action that allows the user to identify the specific target device. A typical example of light bulbs is flashing for a defined duration. Last, the initiator sends a PDU with the command. This command can be, for example, a factory reset or a key transport, which instructs the device to connect to or create a new network. This entire process is illustrated in figure 1. [5]

Touchlink does not use any standard security features provided by ZigBee. The only security feature is the close vicinity of the devices. This fact opens possibilities for attacks like DoS by resetting the device to factory default or eavesdropping on any Touchlink transaction. In the case of key transport transactions, the network key is transported in an encrypted form using AES-ECB and static key in the specification, which was leaked in 2015 on Twitter. Moreover, due to backwards compatibility, it cannot be changed. Even though it is optional in ZigBee 3.0, most major manufacturers like Philips [6] still implement Touchlink. [5]

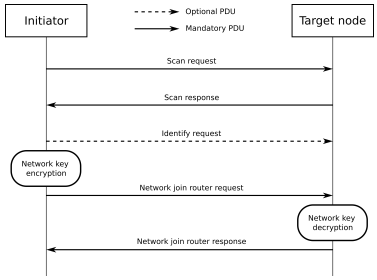


Fig. 1. Example of a Touchlink transaction connecting a new device to an existing ZigBee network. The transaction starts with a scan request PDU from the initiator, to which all network nodes in the Touchlink range respond with a scan response. The initiator then optionally sends the identify request PDU to one target used to identify it accurately. He then continues by encrypting the network key and sending it in the network join router request PDU, to which the target device responds with the result of the transaction in the network join response PDU [5]

III. Known exploits

In the past, one available tool, Z3sec, exploited this vulnerability and was presented by its original discoverers in [5]. Its main features included decryption of network keys transmitted using Touchlink and resetting devices to factory default. It is still available today, but it is written in Python 2 for Ubuntu 16.10 and depends on the scapy-radio project, which is not supported on the latest Ubuntu versions. These facts render it inoperable on the latest releases of GNU/Linux distributions.

IV. zigtoucher

For all of these reasons above, we introduce a new tool for exploiting this static keying material leak, zigtoucher. Zigtoucher is an interactive application and can be accessed on GitHub at Hipuranyhou/zigtoucher [7].

A. Features

The first and most crucial functionality, named KeyLog, is the interception and decryption of the network key transmitted using Touchlink. However, since the network whose key we would like to reveal very likely already has all devices connected, we provide a second functionality to force such

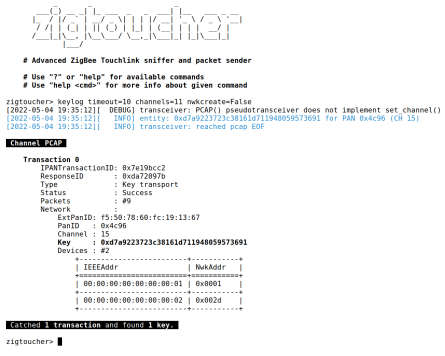


Fig. 2. Example screen of our tool zigtoucher used for security analysis of ZigBee Touchlink. It shows the decryption of a ZigBee network key transmitted during a Touchlink key transport transaction captured in a PCAP file

transaction. This functionality enables us to restore a device in range to its factory state using Touchlink. Furthermore, since we probably do not know which ZigBee channels contain Touchlink capable devices in range, one can scan all or some channels to find these. KeyLog can continue into another functionality, sniff, enabling us to intercept and decrypt any ZigBee traffic after the first decrypted key. We can use this to decrypt any following network key updates, provided that we never stop listening. Finally, Touchlink lets us connect any capable device to a new network even if it already belongs to one or create a new network, which we also implement. This functionality creates so-called network files, which KeyLog can also create, and we can use these to control any device in the provided ZigBee network.

Any mode sending packets can use filters to attack only concrete devices. All modes also provide detailed console logs and file logs, saving their results to CSV files and saving all packets to PCAP files. Even though zigtoucher is interactive, all sniffing modes can run in the background.

B. Chosen building blocks

Zigtoucher is implemented using Python 3. This choice is mainly based on the excellent availability of libraries for implementing functionalities from our application design, such as saving to YAML or CSV files. An essential library available in Python, however, is Scapy. Scapy allows us to construct arbitrary ZigBee PDUs in addition to being able to store and process files in PCAP format. The GNU Radio library is also used to set up and control SDR, Ettus USRP B210. Moreover, since the Scapy library cannot utilise SDRs, we use the scapy-radio library to connect them.

Scapy-radio is nonfunctional on Ubuntu 20.04 LTS because of its dependency on GNU Radio 3.7. Because of this, we had to update scapy-radio to GNU Radio 3.8 as a by-product of our work. What is more, Scapy does not implement support for Touchlink PDUs. We implemented this support based on the partial implementation in another fork of scapy-radio. Finally, to build a functioning GNU Radio flowgraph for ZigBee, we had to update project gr-zigbee to GNU Radio 3.8. More information about these updates can be found on the zigtoucher wiki page at GitHub.

C. Radio layer

Zigtoucher provides easy extensibility in all aspects and currently supports two radio technologies. First, all radios can have an associated Wireshark window open to show raw sent and received PDUs and save all of these PDUs to PCAP files. The first supported radio technology is SDR, Ettus USRP B210. The GNU Radio flowgraph variables controlling the SDR's physical settings can be changed in runtime. The second radio technology is the simulated reception of PDUs by reading PCAP files. This decision to simulate the reading of PCAP files as a radio makes the implementation effortless and lets us analyse past transactions.

D. Configuration

Zigtoucher allows fine-grained configurability using three methods. All of these methods are optional since we provide sensible defaults. One can use configuration file or command-line switches to provide settings for the whole application run or non-interactive mode. In interactive mode, parameters specific to only one run of a given mode can be specified, and autocompletion will help set them.

V. Testing

A. Testbed

The test machine is an Apple MacBookPro11.4 laptop with an Intel i7-4770HQ processor, 16 GB of RAM and Arch Linux x86_64 operating system with Linux kernel version 5.17.5-arch1-1. The application runs in a virtualised environment on the Ubuntu 20.04 LTS operating system using a combination of KVM and QEMU hypervisors. The SDR Ettus USRP B210 is chosen for receiving and transmitting together with a pair of omnidirectional antennas designed for 2.4 GHz frequency and 10 dBi gain. The tested network consists of a Philips LWB004 bulb with firmware version 5.127.1.26420 and a Philips 324131137411 remote control.

Fig. 3. Testbed consisting of SDR Ettus USRP B210, lightbulb Philips LWB004 with firmware version 5.127.1.26420 and Philips 324131137411 remote utilised in testing of our tool zigtoucher used for security analysis of ZigBee Touchlink

B. Test cases

1) Maximal distance: This scenario tests the maximum distance of both eavesdropping on Touchlink transactions and resetting devices to factory state under laboratory conditions. These conditions represent a free area outdoors without obstacles between the SDR and the network. The test is performed by gradually adjusting the sensitivity of the SDR, at a sampling rate of four million samples per second, on channel 11 of the 2.4 GHz band only. Five measurements are made for each sensitivity level, with the maximum distance achieved only counted if all five measurements are successful.

2) Real-life scenario: This scenario presents a test of these features when standing in front of a family house. The four locations marked on the floorplan depicted in figure 4 will be tested. The outside walls are built with 350 millimetres thick bricks and triple-glazed windows. The best result of the maximum distance scenario will determine the SDR settings used. Again, five measurements are taken for this scenario, and their success rate is evaluated.

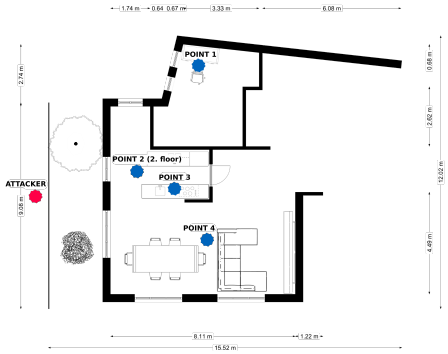


Fig. 4. The floorplan of the real-life test building has 350 millimetres thick brick walls and triple-glazed windows with ZigBee network points and attacker point marked. This scenario is used to test the practical applicability of eavesdropping on Touchlink transactions from the attacker's point of view

C. Results

1) Maximal distance: The results of the maximum distance measurement for eavesdropping on the network key transmission are plotted in table I. The best result was obtained with a receiver sensitivity of 30 dB, resulting in a distance of 29 meters. We then used the same settings to set the receiver during the factory reset test, which needs to both receive and transmit. The results for the factory reset test are plotted in the table II. The best result was obtained at a maximum sensitivity of 89 dB, resulting in a distance of 1.7 meters.

2) Real scenario: The real-life test used a sensitivity setting of 30 dB for the receiver and 89 dB for the transmitter based on the results of the maximum distance test. As a result, in the case of points 1 and 2, no attempt to decrypt the key transmitted using Touchlink was successful. On the other hand, all attempts were successful in the case of points 3 and 4. No attempts were successful in the reset to factory state test.

TABLE I
MAXIMUM DISTANCES OF EAVESDROPPING ON TOUCHLINK BASED ON SDR RX GAIN

RX gain (dB)	Distance (m)
0	0.1
15	13.0
30	29.0
45	21.0
60	10.0
76	0.3

TABLE II
MAXIMUM DISTANCES OF TOUCHLINK FACTORY RESET BASED ON SDR TX GAIN

TX gain (dB)	Distance (m)
0	0.0
15	0.0
30	0.0
50	0.3
70	1.7
89	1.7

VI. CONCLUSION

As we explored in this work, IoT protocols like ZigBee provide state-of-the-art security features. However, the problem with their safety often comes from user experience features implemented at the expense of security. In addition, features like Touchlink do not follow the best security practices, as was the choice to use static keying material in this case.

Test results of our application revealed little practical applicability of this attack and a theoretical maximum distance in the low tens of meters for eavesdropping on the transmission of a network key using Touchlink. Practical usability is further reduced by the practical inability of the attacker to force this transaction by resetting the device to its factory state, which is an assumed feature according to the Touchlink definition. It is also necessary to mention the benefits of the ZigBee 3.0 specification, which provides a partial defence against these attacks. Consumers and administrators of networks supporting ZigBee 3.0 can avoid this attack by taking advantage of the centralised security model and using installation codes.

Even though our testing showed little practical applicability, Touchlink still generally presents a high risk for ZigBee networks, and it is not advisable to continue its implementation by manufacturers.

REFERENCES

[1] W. J. Chappell, E. J. Naglich, C. Maxey and A. C. Guyette, "Putting the Radio in "Software-Defined Radio": Hardware Developments for Adaptable RF Systems," in Proceedings of the IEEE, vol. 102, no. 3, pp. 307-320, March 2014, doi: 10.1109/JPROC.2014.2298491.

[2] NI. "USRP B210 USB Software Defined Radio (SDR) - Ettus Research". ETTUS.com. https://www.ettus.com/all-products/ub210-kit (accessed May 5,2022).

[3] O. Hersent, D. Boswarthick, and O. Elloumi, "The internet of things: applications to the smart grid and building automation", 2nd ed. Hoboken: John Wiley, 2012.

[4] Cluster Library Specification, zigbee Document 05-3474-22, Apr 2019. [Online]. Available: https://csa-iot.org/developer-resource/specifications-download-request

[5] P. Morgner, S. Mattejat, Z. Benenson, C. Müller, and F. Armknecht, "Insecure to the touch", in Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2017, pp. 230-240.

[6] Signify Holding. "Zigbee 3.0 support in Hue Ecosystem Philips Hue Developer Program." MEETHUE.com. https://developers.meethue.com/zigbee-3-0-support-in-hue-ecosystem (accessed May, 5,2022).

[7] zigtoucher. (2022), GitHub. Accessed: May 5,2022. [Online]. Available: https://github.com/Hipuranyhou/zigtoucher.

Příloha C

Obsah přiloženého DVD

```
README ..................... stručný popis souborů obsažených na přiloženém DVD
thesis
    Satoplet-Jakub-BP-ZigBee.pdf .. text bakalářské práce ve formátu PDF
    src ....................... zdrojová forma bakalářské práce ve formátu LaTeX
zigtoucher ..................... zdrojové kódy výsledné aplikace zigtoucher
zigtoucher.pdf .... vědecký článek shrnující bakalářskou práci ve formátu PDF
```