Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-136227-73957

Bc. Peter Kaňuch

Zefektívnenie zabezpečenia komunikácie zariadení Internetu vecí pomocou OpenHip

Diplomová práca

Vedúci práce: Ing. Dominik Macko, PhD.

Apríl 2019

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-136227-73957

Bc. Peter Kaňuch

Zefektívnenie zabezpečenia komunikácie zariadení Internetu vecí pomocou OpenHip

Diplomová práca

Študijný program: Internetové technológie

Študijný odbor: 9.2.4 Počítačové inžinierstvo

Miesto vypracovania: Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU, Bratislava

Vedúci práce: Ing. Dominik Macko, PhD.

Apríl 2019

STU FIIT
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Zadanie diplomovej práce

Meno študenta: **Bc. Peter Kaňuch**

Študijný program: Internetové technológie

Študijný odbor: Počítačové inžinierstvo

Názov práce: **Zefektívnenie zabezpečenia komunikácie zariadení Internetu vecí pomocou OAuth 2.0**

Samostatnou výskumnou a vývojovou činnosťou v rámci predmetov Diplomový projekt I, II, III vypracujte diplomovú prácu na tému, vyjadrenú vyššie uvedeným názvom tak, aby ste dosiahli tieto ciele:

Všeobecný cieľ:

Vypracovaním diplomovej práce preukážte, ako ste si osvojili metódy a postupy riešenia relatívne rozsiahlych projektov, schopnosť samostatne a tvorivo riešiť zložité úlohy aj výskumného charakteru v súlade so súčasnými metódami a postupmi študovaného odboru využívanými v príslušnej oblasti a schopnosť samostatne, tvorivo a kriticky pristupovať k analýze možných riešení a k tvorbe modelov.

Špecifický cieľ:

Vytvorte riešenie, zodpovedajúce návrhu textu zadania, ktorý je prílohou tohto zadania. Návrh bližšie opisuje tému vyjadrenú názvom. Tento opis je záväzný, má však rámcový charakter, aby vznikol dostatočný priestor pre Vašu tvorivosť.

Riaďte sa pokynmi Vášho vedúceho.

Pokiaľ v priebehu riešenia, opierajúc sa o hlbšie poznanie súčasného stavu v príslušnej oblasti alebo o priebežné výsledky Vášho riešenia alebo o iné závažné skutočnosti, dospejete spoločne s Vaším vedúcim k presvedčeniu, že niečo v texte zadania a/alebo v názve by sa malo zmeniť, navrhnite zmenu. Zmena je spravidla možná len pri dosiahnutí kontrolného bodu.

Miesto vypracovania: Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU, Bratislava

Vedúci práce: **Ing. Dominik Macko, PhD.**

Termíny odovzdania:

podľa harmonogramu štúdia platného pre semester, v ktorom máte príslušný predmet (Diplomový projekt I, II, III) absolvovať podľa Vášho študijného plánu

Predmety odovzdania:

V každom predmete dokument podľa pokynov na www.fiit.stuba.sk v časti:
home > Informácie o > štúdiu > organizácia štúdia > diplomový projekt

V Bratislave dňa 12. februára 2018

SLOVENSKÁ TECHNICKÁ UNIVERZITA
V BRATISLAVE
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4
1

Ing. Katarína Jelemenská, PhD.
riaditeľka ústavu

STU FIIT

SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce [1]

Študent:

Meno, priezvisko, tituly:	Peter Kaňuch, Bc.
Študijný program:	Internetové technológie
Kontakt:	xkanuch@stuba.sk

Výskumník:

Meno, priezvisko, tituly:	Dominik Macko, Ing. PhD.

Projekt:

Názov:	Zefektívnenie zabezpečenia komunikácie zariadení Internetu vecí pomocou OAuth 2.0
Názov v angličtine:	Optimizing security of communication in Internet of Things by OAuth 2.0
Miesto vypracovania:	Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU, Bratislava
Oblasť problematiky:	Počítačové a komunikačné siete

Text návrhu zadania[2]

Pomocou Internetu je prepojených čoraz viacej zariadení, aj takých, ktoré donedávna na to neboli určené (domáce spotrebiče, automobily, elektromery, atď.). Tým vzniká tzv. Internet vecí (IoT - Internet of Things), v ktorom je v dôsledku množstva prepojených zariadení nutné neustále zvyšovať bezpečnosť a efektívnosť komunikácie. Jednou z oblastí, kde sa IoT zariadenia začínajú používať, je aj zdravotníctvo. Je dôležité, aby komunikácia bola dostatočne bezpečná vzhľadom na citlivosť prenášaných údajov v tejto oblasti. Vnášanie bezpečnostných prvkov do komunikácie ale zvyšuje jej energetickú náročnosť, pričom nízka energetická náročnosť je jeden z najdôležitejších parametrov v IoT. Analyzujte komunikačné protokoly a bezpečnostné mechanizmy používané v IoT. Na základe analýzy navrhnite zefektívnenie zabezpečovacieho mechanizmu komunikácie, napr. obmedzením nepotrebných funkcií alebo fixnou konfiguráciou niektorých parametrov komunikačného protokolu. Zamerajte sa na protokol Oauth 2.0 a navrhnite vhodné riešenie pre použitie v oblasti zdravotníctva. Navrhnuté riešenie implementujte a overte zvýšenie efektívnosti zabezpečenia komunikácie (zvýšenie úrovne zabezpečenia pri rovnakej energetickej náročnosti, príp. zníženie spotreby energie pri rovnakej úrovni zabezpečenia) analyticky a vo vhodnom testovacom prostredí alebo pomocou dostupných simulátorov. Svoje výsledky porovnajte s podobnými riešeniami.

---

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- SCIANCALEPORE, Savio, et al. OAuth-IoT: An access control framework for the Internet of Things based on open standards. In: Computers and Communications (ISCC), 2017 IEEE Symposium on. IEEE, 2017. p. 676-681.
- JUNG, Seung Wook; JUNG, Souhwan. Personal OAuth authorization server and push OAuth for Internet of Things. International Journal of Distributed Sensor Networks, 2017, 13.6: 1550147717712627.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Peter Kaňuch, konzultoval(a) a osvojil(a) si ho Ing. Dominik Macko, PhD. a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 18.1.2018

Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / nie[4]

Dňa: 16. 2. 2018

Podpis garanta predmetov

[3] 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

STU FIIT
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Návrh zadania diplomovej práce

Revízia č.: 1 [1]

Študent:

Meno, priezvisko, tituly:	Peter Kaňuch, Bc.
Študijný program:	Internetové technológie
Kontakt:	xkanuch@stuba.sk

Výskumník:

Meno, priezvisko, tituly:	Dominik Macko, Ing. PhD.

Projekt:

Názov:	Zefektívnenie zabezpečenia komunikácie zariadení Internetu vecí pomocou OpenHip
Názov v angličtine:	Optimizing security of communication in Internet of Things by OpenHip
Miesto vypracovania:	Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU, Bratislava
Oblasť problematiky:	Počítačové a komunikačné siete

Text návrhu zadania[2]

Pomocou Internetu je prepojených čoraz viacej zariadení, aj takých, ktoré donedávna na to neboli určené (domáce spotrebiče, automobily, elektromery, atď.). Tým vzniká tzv. Internet vecí (IoT - Internet of Things), v ktorom je v dôsledku množstva prepojených zariadení nutné neustále zvyšovať bezpečnosť a efektívnosť komunikácie. Jednou z oblastí, kde sa IoT zariadenia začínajú používať, je aj zdravotníctvo. Je dôležité, aby komunikácia bola dostatočne bezpečná vzhľadom na citlivosť prenášaných údajov v tejto oblasti. Vnášanie bezpečnostných prvkov do komunikácie ale zvyšuje jej energetickú náročnosť, pričom nízka energetická náročnosť je jeden z najdôležitejších parametrov v IoT. Analyzujte komunikačné protokoly a bezpečnostné mechanizmy používané v IoT. Na základe analýzy navrhnite zefektívnenie zabezpečovacieho mechanizmu komunikácie, napr. obmedzením nepotrebných funkcií alebo fixnou konfiguráciou niektorých parametrov komunikačného protokolu. Zamerajte sa na protokol OpenHip a navrhnite vhodné riešenie pre použitie napríklad v oblasti zdravotníctva. Navrhnuté riešenie implementujte a overte zvýšenie efektívnosti zabezpečenia komunikácie (zvýšenie úrovne zabezpečenia pri rovnakej energetickej náročnosti, príp. zníženie spotreby energie pri rovnakej úrovni zabezpečenia) analyticky a vo vhodnom testovacom prostredí alebo pomocou dostupných simulátorov. Svoje výsledky porovnajte s podobnými riešeniami.

---

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- HERNANDEZ-RAMOS, Jose L., et al. Toward a lightweight authentication and authorization framework for smart objects. IEEE Journal on Selected Areas in Communications, 2015, 33.4: 690-702.
- MOSKOWITZ, Robert, et al. Host identity protocol version 2 (HIPv2). 2015.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Peter Kaňuch, konzultoval(a) a osvojil(a) si ho Ing. Dominik Macko, PhD. a súhlasí, že bude takýto projekt viesť.

V Bratislave dňa 13.2.2019

Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / ~~nie~~[4]

Dňa: 13. 2. 2019

Podpis garanta predmetov

[3] 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

Anotácia

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Študijný program: Internetové technológie

Autor: Bc. Peter Kaňuch
Diplomová práca: Zefektívnenie zabezpečenia komunikácie zariadení Internetu vecí pomocou OpenHip
Vedúci diplomového projektu: Ing. Dominik Macko, PhD.
Apríl 2019

V súčastnosti je k Internetu pripojených čoraz viac vzájomne komunikujúcich zariadení označovaných ako Internet vecí. V práci sa venujeme optimalizácii bezpečnostných protokolov. Hlavným cieľom je zefektívniť niektorý z používaných protokolov v danej oblasti tak, aby sme znížili jeho energetickú náročnosť, a tým predĺžili životnosť zariadenia, no zároveň zachovali všetky bezpečnostné vlastnosti protokolu alebo práve naopak, zvýšili jeho bezpečnosť pri rovnakej energetickej náročnosti. Na základe analýzy viacerých existujúcich bezpečnostných protokolov a ich optimalizácií sme sa rozhodli zamerať na protokol OpenHip. V tomto protokole sme identifikovali možnosti optimalizácie pre efektívne použitie v oblasti IoT a navrhli jeho modifikáciu. Zo šiestich čiastkových návrhov sme implementovali tri (Odstránenie správy CloseAck a priebežného ukončovacieho stavu, Redukcia formátu parametrov protokolu OpenHIP, Odstránenie parametru HI-R - Manuálne riešenie). Takto modifikovaný protokol (E-HIP) sme otestovali v testovacom prostredí na reálnych zariadeniach s bezdrôtovou technológiou Bluetooth. Výsledkom práce je optimalizovaný protokol pre nízko-energetické zariadenia Internetu vecí. Implementované riešenie zníži energetickú náročnosť protokolu približne o 20 percent, čím sa predĺži životnosť zariadenia. Znížením počtu prenesených bajtov sa uvoľní prenosové pásmo pre inú komunikáciu, čo minimalizuje rušenie, či prípadné kolízie. Riešenie je možné rozšíriť spojením s inými existujúcimi optimalizáciami a otestovať s technológiou Bluetooth 5 alebo 5G mobilnou sieťou.

Annotation

Slovak University of Technology Bratislava
Faculty of Informatics and Information Technologies
Degree Course: Internet technology

Author: Bc. Peter Kaňuch
Master Thesis: Optimizing security of communication in the Internet of Things by OpenHip
Supervisor: Dr. Dominik Macko
2019, April

The swelling number of connected devices to the Internet is referred to as the Internet of Things. In our thesis, we focus on IoT, mainly the optimization of security protocols. The main goal of our work is to optimize one of the protocols used in this area in order to reduce its energy efficiency, thus extending the life of the device while maintaining all its security features. Based on the analysis of several existing security protocols, we decided to focus on protocol OpenHip. We identified six possible optimizations in the area of our focus and proposed its modifications. Three of them (Removal of CloseAck Message and Continuous State, Reduction of the OpenHIP Parameter Format, Removal of the HI-R Parameter - Manual Solution) have been implemented. Our modified protocol has been tested on devices with Bluetooth technology. The result is an optimized protocol for the low-energy IoT devices. Our solution reduces the protocol's power consumption by around 20 percent, extending the device life expectancy. Reduction of the transferred bytes releases the bandwidth for other communication, minimizes interference or collisions. Future work can bring an extension of our solution with other optimizations and test it with Bluetooth 5 or mobile network 5G.

Čestné prehlásenie

Čestne prehlasujem, že túto diplomovú prácu pod názvom Zefektívnenie zabezpečenia komunikácie zariadení Internetu vecí pomocou OpenHip som vypracoval sám, na základe konzultácií a uvedenej odbornej literatúry.

V Bratislave, 30.04.2019 .....

Bc. Peter Kaňuch

Poďakovanie

Touto cestou by som sa chcel v prvom rade poďakovať najmä vedúcemu práce Ing. Dominikovi Mackovi, PhD. za odbornú pomoc pri písaní, konzultácie a povzbudenie pri strate motivácie, ďalej mojej rodine za podporu pri štúdiu a v neposlednom rade ďakujem svojim priateľom, špeciálne Matúšovi, za spoločne strávené chvíle vo voľnom čase, odreagovanie sa od každodenných povinností a pod. Taktiež ďakujem OZ Ynet za udelenie prístupu do študovne Ybase a vybavenie kuchynky.

Obsah

1. Úvod

Čoraz viac inteligentných zariadení je pripojených k Internetu. Takéto zariadenia nazývame Internet vecí. Tie, medzi sebou komunikujú prostredníctvom daných komunikačných protokolov a technológií prispôsobených práve pre danú oblasť. V súčastnosti však netreba zabúdať na bezpečnosť, ktorá zohráva čoraz väčšiu rolu v oblasti informatiky a informačných technológií, pretože motivácia útočníkov rastie. údaje posielané medzi jednotlivými komunikujúcimi zariadeniami je potrebné chrániť proti pozmeneniu, či zneužitiu. Na to nám slúžia bezpečnostné protokoly.

Mnoho IoT zariadení je napájaných z batérie. Pre predĺženie životnosti batérie bez potreby jej výmeny, či dobitia, je potrebné optimalizovať procesy náročné na spotrebu elektrickej energie. Jedným z možných riešení je aj optimalizovať bezpečnostné protokoly, ktoré zabezpečujú vlastnosti ako zachovanie integrity, dôvernosti, súkromia, bezpečnosť dát a podobne.

Hlavným cieľom tejto diplomovej práce je optimalizovať jeden z bezpečnostných protokolov používaných v oblasti Internetu vecí a otestovať v niektorej oblasti života.

V prvej časti tejto práce, konkrétne v kapitole 2 sme definovali pojem Internetu vecí (IoT), využitie v jednotlivých oblastiach života, opísali komunikačné technológie, či uviedli bezpečnostné požiadavky pre IoT zariadenia a delenie bezpečnostných protokolov. V kapitole 3 uvádzame niekoľko bezpečnostných protokolov používaných v doméne IoT a ich existujúce riešenia optimalizácie. V kapitole 4 sme stanovili požiadavky na nami navrhované riešenie. Samotné návrhy riešenia optimalizácie bezpečnostného protokolu HIP uvádzame a opisujeme v kapitole 5. V časti 6 opisujeme implementáciu prototypu a finálneho riešenia. V posledných kapitolách 7 a 8 uvádzame testovanie navrhnutého riešenia a zhrnutie tejto práce.

2. Internet vecí

V tejto kapitole uvádzame definíciu pojmu Internetu vecí (v skr. *IoT - z angl. Internet of Things*), jednotlivé existujúce technológie, či využitie IoT v rôznych oblastiach života. V druhej časti tejto kapitoly rozoberáme klasifikáciu bezpečnostných riešení v tejto oblasti.

2.1 Čo je to Internet vecí

V posledných rokoch sa pojem Internet vecí objavuje čoraz viac [48]. Všetky informácie na internete boli donedávna vytvárané človekom. Ten však má obmedzený čas a presnosť. V dôsledku toho sa mohla rozvinúť oblasť Internetu vecí, kedy rôzne informácie, z jednotlivých oblastí života, budú vytvárané pomocou zariadení a k nim pripojených senzorov [32].

Existuje niekoľko foriem definície IoT. Mnoho ľudí si pod daným pojmom predstaví len akési prepojenie počítačov, tabletov, či mobilných zariadení. V skutočnosti je to množina všadeprítomných, vzájomne prepojených a komunikujúcich zariadení prostredníctvom Internetu. Prepája objekty skutočného a virtuálneho života, kde všetko môže komunikovať [68]. Termín IoT pozostáva z dvoch slov, kde *Internet* posúva tento pojem smerom k sieťovým technológiám a vecí *(z angl. things)*, ktorý zahŕňa všetky priamo adresovateľné a identifikovateľné objekty skutočného alebo digitálneho sveta so schopnosťou vzájomnej komunikácie, ktoré môžme klasifikovať do troch hlavných skupín: ľudia, prístroje a informácie [6] [42] [68].

Taktiež je potrebné povedať, že zariadenia Internetu vecí majú obmedzené výpočtové možnosti, častokrát určené len na zber a odosielanie surových dát, limitované kapacitou elektrického zdroja (napr. batérie) [40]. S tým súvisí aj sieťová konektivita. Pre dlhšiu výdrž batérie sú tieto zariadenia často v stave nečinnosti (*idle, power down* režim) a do

siete sa pripájajú len pri priamo prebiehajúcej komunikácii.

Požiadavky na zariadenia IoT [8] :

1. mobilita a bezdrôtovosť
2. použiteľnosť s/vo vnorených systémoch
3. rôznorodosť systémov pre jednotlivé možnosti využitia
4. škálovateľnosť systémov z hľadiska veľkého nárastu
5. energetická účinnosť

2.2 Využitie IoT v rôznych oblastiach života

V súčastnosti sa IoT vyskytuje pomaly v každej oblasti života, od priemyslu a moderných veľkomiest, až po agrokultúru, či zdravotníctvo [11].

- Priemysel a agrokultúra - jedna z najväčších oblastí využitia IoT technológií, kde sa postupne do automatizovaných procesov výroby zavádzajú prvky inteligentného prostredia Internetu vecí [25] [69]. Napríklad v potravinárskom priemysle umožňujú lepšie kontrolovať jednotlivé fázy spracovania potravín [11]. V oblasti poľnohospodárstva vieme tieto zariadenia využiť na kontrolovanie stavu plodín, ich uchovanie počas jednotlivých fáz spracovania a monitorovanie meteorologických parametrov [6].
- Logistika - IoT zariadenia v oblasti logistiky umožňujú jednoduchšie plánovanie, či kontrolovanie stavu zásob v jednotlivých skladoch [45]
- Inteligentné mestá, budovy a domácnosti - v tejto oblasti života si táto technológia taktiež našla uplatnenie. Pomocou jednotlivých senzorov dokážeme monitorovať veci ako kvalitu ovzdušia, hlučnosť prostredia, spotrebu energií, či riadiť parkovanie, osvetlenie miest, či vývoz odpadu [74]. Tým sa snažíme zvyšovať komfort občanov miest a jednotlivých domácností [11].
- Zdravotníctvo - v oblasti zdravotníctva najdú tieto zariadenia využitie najmä v oblasti monitorovanie zdravia pacienta pri rôznych ochoreniach ako diabetes, vysoký krvný tlak, EKG. Mnohé výskumy hovoria aj o blízkej možnosti integrovania IoT priamo do ľudského tela [67].

Tabuľka 2.1: Porovnanie technológií IoT [23]

	Frekvencia	**Dosah**	**Rýchlosť prenosu**
WiFi	900 MHz - 5 GHz	100 m	100 Mbit/s
Bluetooth	2,4 GHz	50 m	1 Mbit/s
6LoWPAN	-	-	20 - 250 kbit/s
RFID	125 kHz - 5,875 GHz	100 m	4 Mbit/s
NFC	13,56 MHz	10 cm	420 kbit/s
Sigfox	868 MHz	50 km	1 kbit/s
ZigBee	2,4 GHz	100 m	250 kbit/s
LoRaWAN	433 - 928 MHz	3 - 15 km	50 kbit/s
Thread	-	30 m	250 kbit/s
Z-Wave	900 MHz	30 m	100 kbit/s
5G	>28 GHz	n/a	10 Gbit/s

- Zábava - prostredníctvom zozbieraných údajov o jednotlivých používateľoch, dokážu spoločnosti lepšie odhadnúť a vyhovieť potrebám zákazníka [22].

2.3 Komunikácia v IoT

Ako bolo spomenuté, zariadenia Internetu vecí sa nachádzajú pomaly v každom odvetví života. Z dôvodu požiadaviek opísaných v časti 2.1, sa taktiež vyvinuli jednotlivé komunikačné protokoly, niektoré štandardizované *IEEE, ISO, RFC* [3][66]: low power WiFi (IEEE 802.11), 6LoWPAN (IEEE 802.15.4, RFC6282), RFID, NFC (ISO/IEC 18000-3), Sigfox, LoRaWAN, Zigbee (IEEE 802.15.4), Z-Wave, Thread (IEEE 802.15.4), Cellular, Neul. Štandard IEEE 802.15.4 definujúci fyzickú vrstvu a prístup k médiu tvorí základ pre technológie ním označené. V tabuľke 2.1 sú zobrazené jednotlivé parametre technológií.

2.4 Klasifikácia bezpečnostných riešení používaných v Internete vecí

V každej oblasti informatiky a informačných technológií je potrebné dbať na bezpečnosť. Najdôležitejšie bezpečnostné požiadavky pre IoT zariadenia sú [20, 47, 18]:

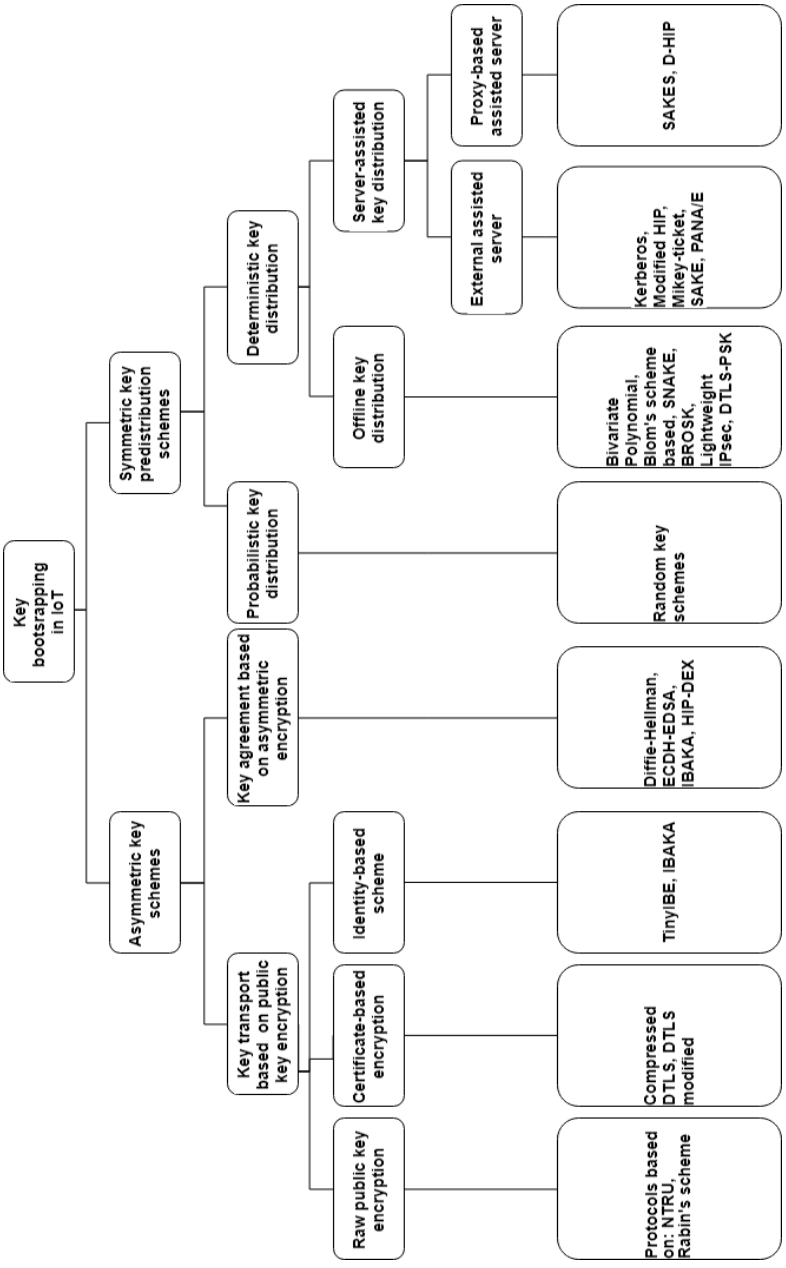
1. bezpečnosť dát, komunikácie a úložiska
2. autorizácia, autentifikácia
3. zachovanie integrity, dôvernosti, súkromia
4. šifrovanie
5. odolnosť voči útokom

2.4.1 Delenie protokolov

Jednotlivé požiadavky opísané v kapitole 2.4 vieme docieliť pomocou rôznych šifier alebo hašovacích funkcií. Tieto mechanizmy využívajú šifrovacie kľúče, a preto jednotlivé používané bezpečnostné protokoly v IoT môžme zatriediť do dvoch veľkých skupín na základe spôsobu výmeny kľúča (viď delenie na Obr. 2.1) [52]:

1. Symetrické
2. Asymetrické

V tabuľke 2.2 môžme vidieť splnenie niektorých požiadaviek daným protokolom.



Obr. 2.1: Rozdelenie protokolov na základe spôsobu výmeny kľúča [52]

Tabuľka 2.2: Sumarizácia bezpečnostných riešení [52]

	Confidentiality	Integrity	Authentication	Authorization	Freshness	Resilience	Computation complexity	Communication complexity	Memory	Scability	Privacy protection
NTRU, Rabin's scheme	+	n/a	n/a	n/a	n/a	+			+	+	n/a
Moustaine and Laurent	n/a	+	+			+	+	+	+	+	+
ZKP based on ECDLP	n/a	+	+		+	+		+	+	+	+
DTLS-modified	+	+	+		+	+			+	+	+
IKEv2-ECC based	+	+	+		+	+			+	+	+
TinyIBE	+		+			+	+		+		
IBAKA	+	+	+		+	+			+	+	+
ECDH-ECDSA	+	+	+		+	+		+	+	+	
HIP-DEX	+	+	+		+	+	+	+	+	+	
E-G	+						+				
DU et al.	+				+		+				
Chan et al.	+						+				
Ito et al.	+						+				
Blom's scheme based	+						+	+	+		
SNAKE	+	+	+		+		+	+	+		
BROSK	+	+	+		+		+	+	+		
Lightweight IPsec	+	+	+		+				+	+	+
DTLS-PSK	+	+	+		+	+			+	+	+
Diet-ESP	+	+	+		+	+	+		n/a		+
Mickey-ticket	+	+	+	+	+		+	+	+		
SAKE	+	+	+		+	+	+	+	+		
PANA/EAP-PSK	+	+	+		+	+	+	+	+		
SAKES	+	+	+		+	+		+			
D-HIP	+	+	+		+						

3. Bezpečnostné riešenia v IoT

V tejto kapitole opisujeme niekoľko existujúcich riešení používaných v Internete vecí. Tie pozostávajú z bezpečnostných a komunikačných protokolov a ich samotnej modifikácii pre zníženie energetickej náročnosti.

3.1 Datagram Transport Layer Security (*DTLS*)

Pri niektorých aplikáciách, používajúcich nepretržitý tok dát, kde nám nezáleží na spoľahlivosti ich doručenia, používame protokol UDP [61]. Protokol TCP by totiž mohol priniesť zbytočné réžijné náklady pri zariadeniach napájaných z batérie [38], a preto kvôli nepodporovaniu protokolu UDP protokolom TLS, bol vyvinutý protokol DTLS [70].

Pre zabezpečenie autentifikácie medzi zariadeniami IoT [70], vieme protokol DTLS použiť v troch rôznych formách [46]:

1. so zdieľaným tajomstvom (*PSK - z angl. pre-shared key*) nastaveného komunikujúcim zariadeniam pomocou bezpečného kanálu,
2. s použitím verejného a súkromného kľúča,
3. použitím certifikátov X509, založených na PKI *(z angl. public key infrastructure)*, podpísaných certifikačnou autoritou.

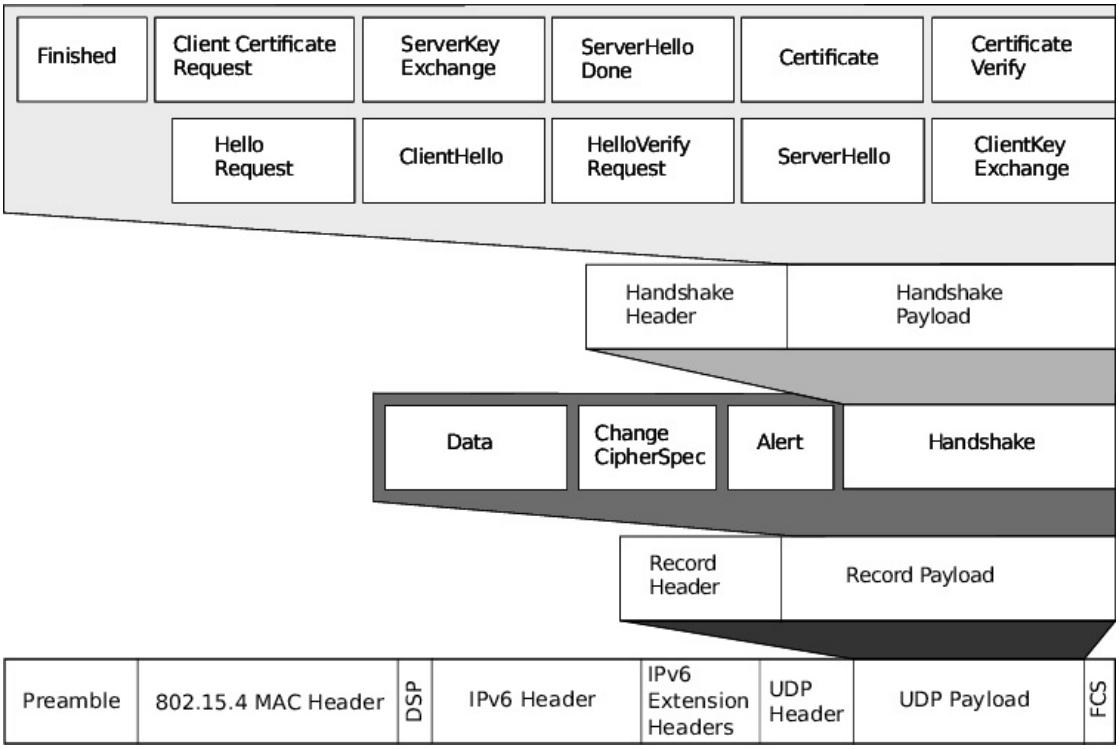
3.1.1 Štruktúra DTLS

Samotný protokol DTLS pozostáva z [9][58][70]:

- *Record protokol*

- *Handshake protokol*
- *Alert protokol*
- *ChangeCipherSpec protokol*
- Aplikačné dáta

Výsledný vyskladaný UDP paket zabezpečený pomocou DTLS protokolu môže potom vyzerať nasledovne (viď Obr. 3.1).



Obr. 3.1: UDP paket zabezpečený pomocou DTLS protokolu [58]

Record protokol - UDP vo svojom pakete ďalej nesie zabalený fragment pozostávajúci z jednoduchého protokolu *Record*, ktorého úlohou je spĺniť bezpečnostné požiadavky ako zachovanie diskrétnosti, integrity, autentifikáciu a šifrovanie dát [58]. Taktiež rozdeľuje tieto dáta do paketov. Hlavička protokolu pozostáva z nasledujúcich častí [61][70]:

1. Typ (1B) - nesie informáciu o ďalšom zabalenom fragmente protokolu.
2. Verzia použitého protokolu DTLS (2B).

3. Epocha (2B) - číslo, ktoré sa mení pri výmene nových šifrovacích kľúčov / tajomstiev.
4. Sequenčné číslo (6B) - identifikuje poradie paketov pre danú epochu.
5. Dĺžka fragmentu (2B).

Za ňou nasleduje jeden fragment typu *Handshake, Alert, ChangeCipherSpec alebo Data*

Handshake protokol - súbor asynchronných správ pre dohodnutie bezpečnostných pravidiel, a tým vytvorenie nového bezpečného kanálu pre komunikáciu [9].

Tento proces je vcelku zložitý. Pozostáva z niekoľkých typov správ, ktoré sú odosielané v jednotlivých fázach tohto procesu, v prísnom dohodnutom poradí. Pri strate paketov alebo nedodržaní poradia je potrebné začať tento proces odznova [61] (viď Obr. 3.2).

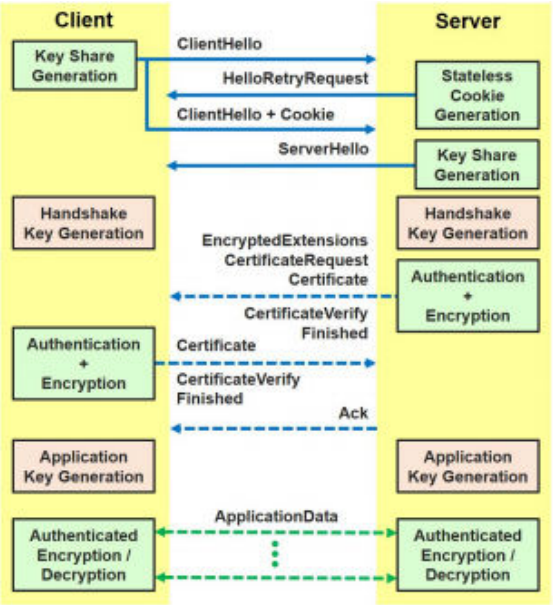
Hlavička protokolu pozostáva z nasledujúcich častí [61][70]:

1. Typ (1B) - nesie informáciu o type prenášanej správy.
2. Dĺžka (3b) - dĺžka celej správy
3. Sequenčné číslo správy (2B)
4. Začiatok / posunutie fragmentu (3B)
5. Dĺžka fragmentu (3B)

Následne nesie samotnú zabalenú správu. Jednotlivé typy správ sú podobné jeho predchodcovi, protokolu TLS [17], a to:

- Hello správy:
    1. HelloRequest - slúži na upovedomenie klienta serverom pre začatie nového procesu výmeny šifrovacieho tajomstva
    2. ClientHello - táto správa sa používa pri prvom kontaktovaní servera alebo taktiež ako odpoveď na predchádzajúcu správu
    3. ServerHello - slúži ako odpoveď na ClientHello správu
    4. HelloVerifyRequest
- Certificate - server odosiela pri používaní certifikátov

- ServerKeyExchange - určený na dokončenie výmeny kľúča pri niektorých typoch
- CertificateRequest - umožňuje serveru vypýtať si certifikát klienta
- ServerHelloDone - slúži na informovanie klienta o pripravenosti servera na výmenu kľúčov
- ClientKeyExchange - správa odoslaná klientom po ServerHelloDone
- CertificateVerify - používaná pre overenie certifikátov klienta
- Finished - správa overujúca úspešnosť dokončenia procesu výmeny kľúča prostredníctvom jeho použitia. Po tejto správe môže nasledovať výmena dát.



Obr. 3.2: Priebeh *Handshake* procesu [9]

Alert protokol - slúži na odosielanie chybových hlášok medzi komunikujúcimi uzlami, či na ukončenie spojenia [71]. Táto funkcionalita sa však už nepoužíva. Protokol DTLS sám zistí, že prichádzajúce správy sú neplatné a následne ukončí spojenie [61].

ChangeCipherSpec protokol - slúži na informovanie zariadenia, že sa použijú nové šifrovacie kľúče / tajomstvá pre nasledujúce správy [38]. Hoci daný fragment nezaraďujeme k správam protokolu *Handshake*, napriek tomu jeho súčasťou [61].

Aplikačné dáta - po dohodnutí jednotlivých bezpečnostných pravidiel pomocou spomínaných správ, protokol môže odosielať údaje prostredníctvom vytvoreného bezpečného kanála [61].

3.1.2 Energetická náročnosť protokolu

Existuje niekoľko prístupov pre vyjadrenie energetickej náročnosti protokolov:

1. odmeraním spotreby elektrickej energie,
2. odmeraním časov vykonávania jednotlivých procesov,
3. veľkosťou odosielaných rámcov (veľkosť hlavičiek),
4. počtom odosielaných správ pre dohodnutie bezpečného komunikačného kanálu.

Energetická náročnosť pri použití rôznych technológií, či parametrov môže byť odlišná. V nasledujúcej tabuľke 3.1 porovnávame energetickú náročnosť protokolu DTLS na základe dĺžky trvania procesu vytvorenia bezpečného kanála na komunikáciu a spotreby elektrickej energie.

Tabuľka 3.1: Porovnanie času a spotreby energie handshake procesu podľa jednotlivých analyzovaných zdrojov

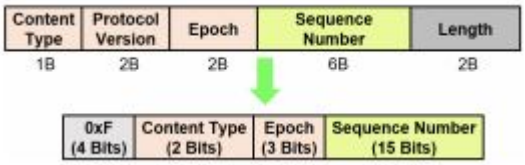
	Preamble Sampling Protocols [71]	**Beacon-Enabled IEEE 802.15.4 Networks [71]**	**802.15.4 RSA key (2048b)[39]**	**2048-bit key[38]**
DTLS Handshake Duration	<10 s	1.8 - 16.6 s	6,949 ms	-
DTLS Handshake Energy (mJ)	<30	4 - 5.5	579	487.8

3.1.3 Existujúce riešenia

Keďže v súčastnosti neexistujú dostatočne silné batérie na to, aby zariadenia v Internete vecí vydržali pracovať čo najdlhšie, je potrebné optimalizovať jednotlivé protokoly. V tejto časti opisujeme niekoľko existujúcich riešení optimalizácie protokolu DTLS.

3.1.3.1 eeDTLS (*z angl. Energy-Efficient DTLS*)

Riešenie eeDTLS je jednou z optimalizácii protokolu DTLS[9]. Svoje riešenie implementovali s použitím technológie Bluetooth. V tomto článku autori opisujú dve riešenia opti-



Obr. 3.3: Porovnanie pôvodnej a zmenšenej hlavičky DTLS

malizácie, a to pomocou:

- zmenšením hlavičiek jednotlivých protokolov v pakete (tu nás zaujíma iba hlavička protokolu DTLS),
- optimalizáciou procesu vytvorenia bezpečného kanála na komunikáciu (tzv. handshake).

Pôvodnú trinásť bajtovú hlavičku typu Record opísanú v 3.1.1 zmenšili na veľkosť troch bajtov, a to vynechaním poľa verzie a informácie o dĺžke fragmentu a zmenšením ostatných častí na bitové veľkosti namiesto bajtov. Napríklad, na typ zabalenej správy postačujú iba dva bity, nakoľko sú len tri možnosti správ. Pre zarovnanie dĺžky hlavičky a identifikáciu zmenšenej verzie DTLS použili 0xF na začiatku. Takouto optimalizáciou dokázali zmenšiť výsledný paket až o 91% a energetickú náročnosť znížili o 33%.

Pri optimalizácii spomínaného procesu handshake použili TLS-rozšírenia, kde obe komunikujúce zariadenie si môžu uložiť verejné kľuče pre ich následné overenie. Tým znížili počet správ *Certificate*, ale aj nutnosť overenia podpisov. Takýmto spôsobom dokázali znížiť náročnosť o 47%.

3.1.3.2 Optimalizovaná implementácia DTLS

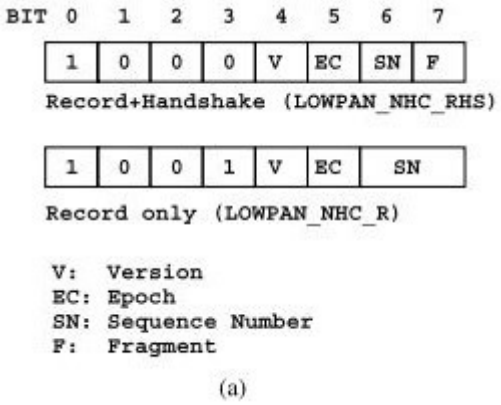
Z dôvodu vysokej pamäťovej a výpočtovej zložitosti bezpečnostných algoritmov, sa , autori [13] rozhodli znížiť energetickú náročnosť optimalizáciou implementácie protokolu DTLS. Vyvinuli vlastnú knižnicu ECC pre implementáciu Eliptických kriviek používaných v kryptografii. Vo svojom riešení použili jazyk asembler, čím dokázali preniesť niektoré pamäťové operácie do registrov. Taktiež optimalizovali náročné aritmeticko-logické oprácie. Pri takejto optimalizácii v článku uvádzajú 6,5-krát dlhšiu životnosť zariadenia.

3.1.3.3 Lithe: *Lightweight Secure CoAP for the Internet of Things*

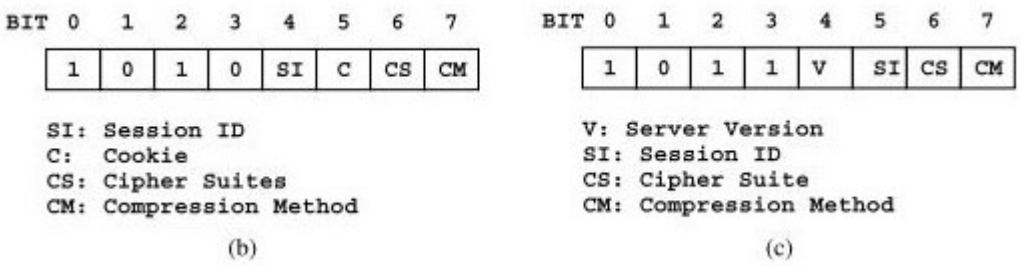
Autori [58] prinášajú ďalšie riešenie optimalizácie DTLS protokolu. Lithe je integrácia komunikačného protokolu CoAP a protokolu DTLS. Vo svojom riešení opisujú kompresiu hlavičiek protokolu na veľkosť 3 - 5 bajtov. Nezaručujú však bezpečnostné požiadavky protokolu. Na základe štandardu 6LoWPAN (IPV6 over Low power Wireless Personal Area Network) predstavili dve riešenia:

1. 6LoWPAN-NHC-RHS (NHC - next header compresion, RHS - record and handshake) - obe hlavičky sú komprimované,
2. 6LoWPAN-NHC-R (R - record) - record protokol prenáša aplikačné dáta, preto je komprimovaná len jeho hlavička

a riešenie pre kompresiu správ Client/Server Hello. Výsledné riešenie môžme vidieť na Obr. 3.4 a 3.5.



Obr. 3.4: a) Hlavička riešenia 6LoWPAN-NHC-RHS a 6LoWPAN-NHC-R



Obr. 3.5: b) Hlavička správy ClientHello c) ServerHello hlavička

Prvé štyri bity označujú typ správy. Obsah ostatných častí hlavičky kódujú jedným alebo dvoma bitmi. Napríklad pri určení verzie protokolu: ak je bit nastavený na 0, používa sa novšia verzia protokolu a pod.

3.2 Internet Protocol Security (*IPsec*) a Internet Key Exchange (*IKE*)

Z dôvodu nepodporovania autentifikácie, či šifrovania dát komunikačným protokolom COAP (z angl. *Constrained Application Protocol*), sa taktiež v súčastnosti v IoT používa protokol IPsec (*Internet Protocol Security*) [12]. Operuje nad sieťovou vrstvou RM OSI modelu a zabezpečuje jednotlivé bezpečnostné vlastnosti ako zachovanie integrity, dôvernosti, autentifikáciu [16]. Napriek tomu, že je najčastejšie spájaný v použití so sieťami VPN, zabezpečuje šifrovanie vyšších vrstiev medzi dvoma koncovými uzlami, a preto je považovaný za vhodné riešenie pre použitie v oblasti Internetu vecí [36] [27].

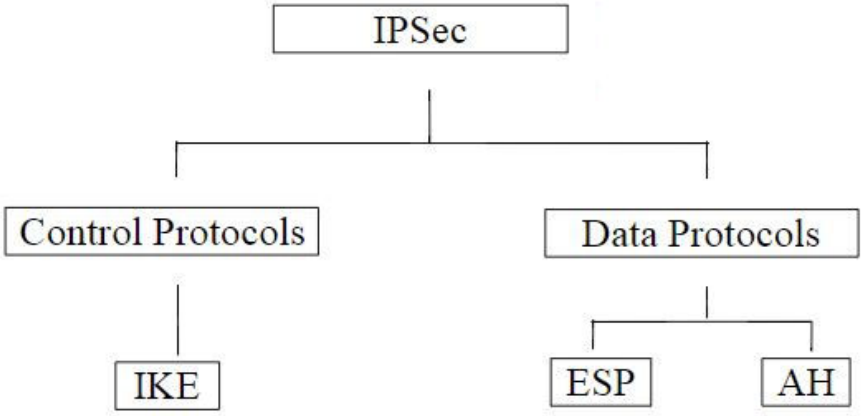
Dohodnutie atribútov potrebných pre vytvorenie bezpečnej komunikácie medzi dvoma zariadeniami ako šifrovacie algoritmy, zdieľané tajomstvo, kľúče sa nazýva Asociácia zabezpečenia (z angl. *SA - Security Association*). Jednotlivé parametre zadefinované SA môžu byť nastavené staticky alebo dynamicky. Pri statickom nastavení sa používa zdieľané tajomstvo PSK, pri dynamickom sú k dispozícii tri rôzne riešenia [59]:

1. Kerberized Internet Negotiation of Keys,
2. Internet Key Exchange (IKE), ktoré je najčastejšie používaným riešením [27],
3. riešenie založené na DNS.

3.2.1 Štruktúra protokolu IPsec

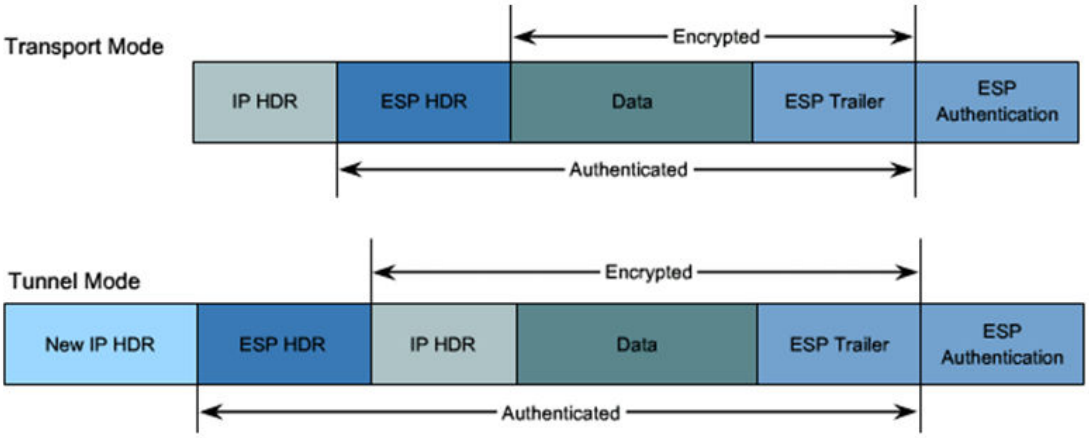
Architektúru protokolu IPsec môžme rozdeliť na dve časti (viď Obr. 3.6) pozostávajúce z riadiacej časti, najčastejšie zabezpečenej protokolom IKE a dátovej časti zabezpečenej protokolmi ESP a AH (*z angl. Encapsulating Security Payload Header, Authentication Header*) [27]. Protokol AH zabezpečuje integritu dát a autentifikáciu. ESP okrem vlastností poskytujúcich protokolom AH zabezpečuje aj dôvernosť [34].

IPsec rozdeľujeme do dvoch režimov (viď Obr. 3.7):

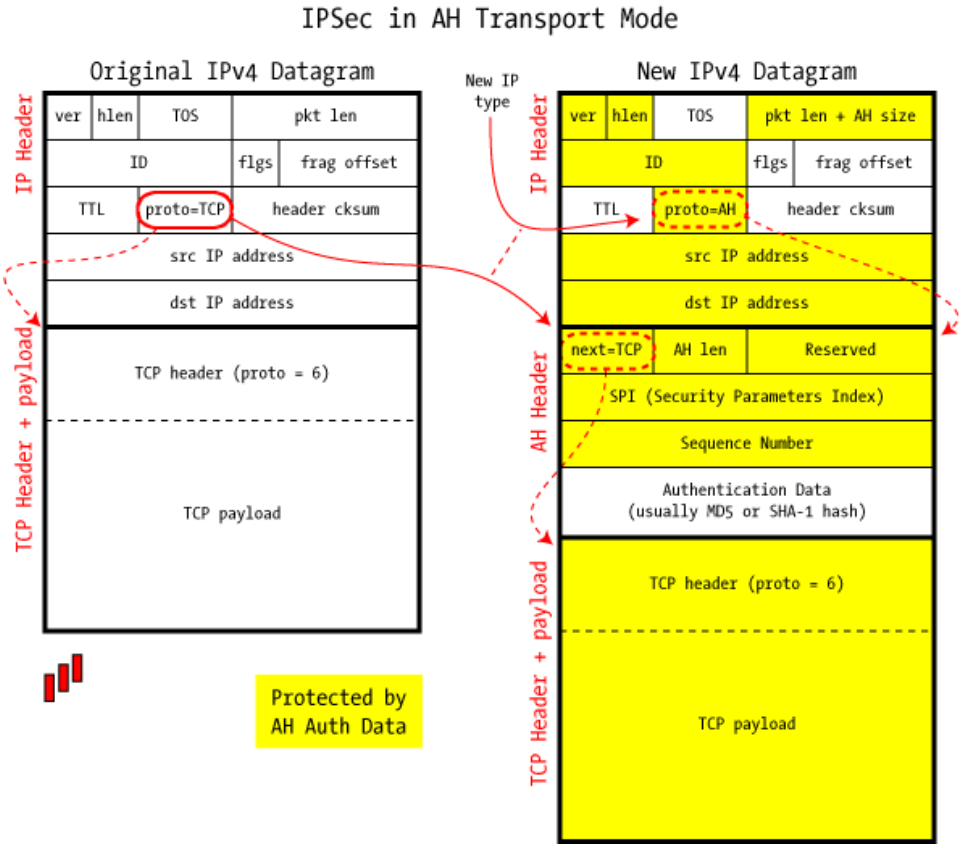


Obr. 3.6: Architektúra protokolu IPsec

- tunelový režim (zabaľuje IP paket do nového IP paketu),
- transportný režim (IP hlavička ostáva nezašifrovaná).



Obr. 3.7: Štruktúra paketu v jednotlivých módoch[33]



Obr. 3.8: Detailný pohľad na IPv4 packet chránený protokolom IPsec v transportnom móde [28]

3.2.2 Štruktúra protokolu IKEv2

Ako bolo spomenuté v úvode tejto časti 3.2, protokol IKE (konkrétne IKEv2) je jedným z možných riešení pre dohodnutie SA parametrov a vytvorenie tak bezpečného komunikačného kanálu pre prenos dát.

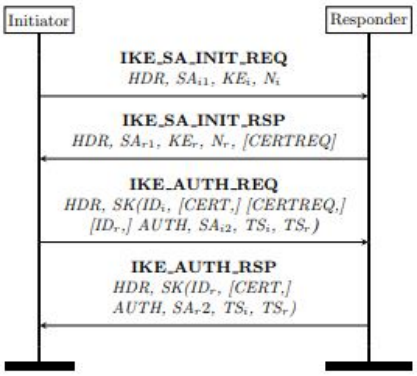
IKEv2 pozostáva z troch častí. Pre počiatočné nastavenia postačujú prvé dve fázy protokolu, zobrazené na Obr. 3.9[5][29][24]:

1. IKE_SA_INIT Exchange - pozostávajúca z dvoch správ:
   - žiadosť - obsahuje polia HDR (Hlavička), SAi (nesie informáciu o podporujúcich kryptografických algoritmoch), KEi (Key Exchange, hodnota algoritmu DH - Diffie Hellman), Ni (Nonce),
   - odpoveď - zložená z HDR, SAr (výber kryptografického algoritmu), KEr (hodnota pre dokončenie algoritmu DH), Nr (Nonce), [CERTREQ] (voliteľná správa používaná pri zabezpečení pomocou certifikátu)

     Pre zabezpečenie protokolu proti replikovaniu správ útočníkom, môže byť táto časť doplnená párom správ s obsahom cookies.
2. IKE_AUTH Exchange - slúži na vytvorenie IPsec SA. V tejto časti sú už správy šifrované. Prenášaný obsah sa však môže líšiť podľa typu autentifikácie (zdieľaný kľúč PSK, RSA certifikáty, EAP - Extensible Authentication Protocol). Vymenený pár správ medzi komunikujúcimi uzlami vyzerá rovnako: HDR, IDi (Identifikátor pre určenie identity), [CERT] (certifikát), [CERTREQ] (zoznam certifikačných autorít), [IDr] (pri nejasnom IDi, možnosť iného identifikátora pre dokončenie výmeny), AUTH (hodnota vypočítaná v predošlej časti), SA (návrh SA pre protokol IPsec), TSi, TSr (z angl. traffic selectors).
3. CREATE_CHILD_SA Exchange - táto časť pozostáva taktiež z páru správ, určených na dohodnutie nových šifrovací parametrov.

3.2.3 Energetická náročnosť protokolu

Na základe jednotlivých možností vyjadrenia energetickej náročnosti protokolov opísaných v časti 3.1.2, v nasledujúcej časti porovnávame energetickú náročnosť protokolu IPsec a jeho súčastí.

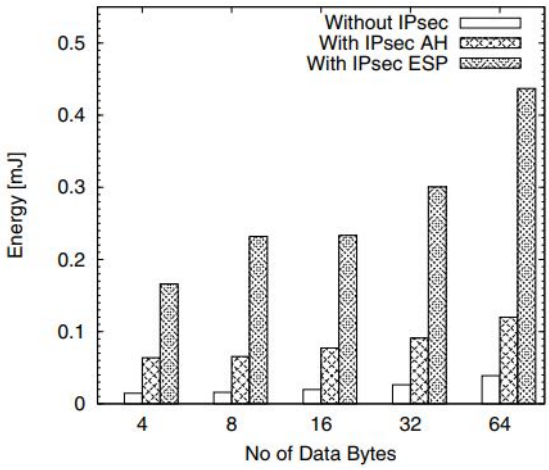


Obr. 3.9: Priebeh výmeny správ počas prvých dvoch častí

Tabuľka 3.2: Energetická náročnosť protokolu IKEv2 vyjadrená spotrebou CPU času [34]

IKEv2	**CPU Time consumed**
IKE_SA_INIT exchange	6.91 s
IKE_AUTH exchange	3.97 s
Total	10.87 s
Total with message transmission time delay	11.1 s

V tabuľke 3.2 sú zobrazené časy jednotlivých fáz protokolu IKEv2 [34] a taktiež celkový čas potrebný na dokončenie vytvorenia bezpečného kanálu s a bez času potrebného na prenos dát s komunikujúcim zariadením.



Obr. 3.10: Porovnanie náročnosti na základe použitého protokolu v IKEv2 [60]

Na Obr. 3.10 je zobrazená energetická náročnosť z pohľadu použitého protokolu AH alebo ESP (použitého na autentifikáciu a kryptovanie) vrámci protokolu IPsec [60]. V poslednej tabuľke 3.3 je porovnanie časovej náročnosti protokolu výmeny kľúča Diffie Hellman (DH) z hľadiska veľkosti kľúča, ktorý môže byť implementovaný v protokole IPsec/IKE [57].

Tabuľka 3.3: Energetická náročnosť protokolu Diffie Hellman [57]

	Key size (bits)	**Key generation (mJ)**	**Key exchange (mJ)**
DH	1024	875.96	1046.4
DH	512	202.56	159.6

3.2.4 Existujúce riešenia

V nasledujúcej časti opisujeme jednotlivé riešenia optimalizácie energetickej náročnosti protokolu IPsec.

3.2.4.1 LKA protokol

Jedným z existujúcich riešení optimalizácie je aj protokol LKA (z angl. *Proposed light weight key agreement*) [41]. Tento protokol bol vyvynutý za účelom optimalizácie a ako možná varianta minimálnej konfigurácie protokolu IKEv2, v ktorej sú podporované len základné možnosti ako fázy IKE_SA_INIT, IKE_AUTH. Pri vytváraní bezpečného komunikačného kanálu parameter SAi (zoznam podporujúcich kryptorgrafických algoritmov) môže obsahovať len jeden algoritmus. Taktiež, použitie certifikátov nie je podporované.

Protokol LKA, podobne ako minimálna verzia IKE, podporuje len informáciu o jednom kryptorgrafickom algoritme. Pre výmenu kľúčov a podpisov sa používaju algoritmi založené na eliptických krivkách, a to: ECDH a ECDSA (*Elliptic Curve Diffie Hellman & Elliptic Curve Digital Signature algorithm*). S certifikátmi je to rovnako ako u minimálnej verzie IKE.

Tabuľka 3.4 zobrazuje priemernú hodnotu z vykonaných štyroch meraní v určitých simulačných časoch (50, 100, 150, 200 s) jednotlivých protokolov.

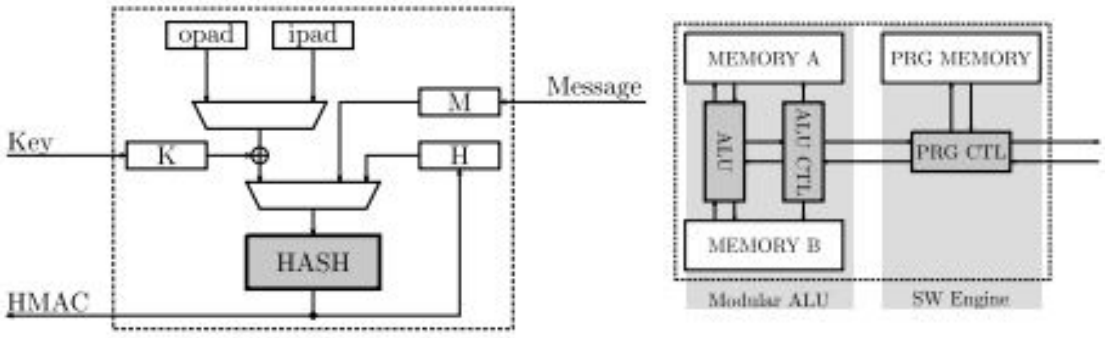
Tabuľka 3.4: Porovnanie protokolov IKE a LKA

	Computation cost	**End-to-end delay (ms)**	**Energy consumption (J)**
IKE	6,65	92,25	23,375
LKA	5,775	62,75	21,75

3.2.4.2 Vylepšenia pomocou hardvéru

Iné vylepšenia efektívnosti protokolu sú dosiahnuté pomocou hardvéru. Existuje niekoľko rôznych implementácií na programovateľných hardvéroch ako Xilinx Virtex-4, Spartan-3, Virtex-5, Virtex-2 Pro FPGA [44][54][64][72].

IPSecco: V danej práci sa zamerali na implementovanie jednotlivých súčasti protokolov (AH, ESP, IKE) používaných v protokole IPsec napr. konštrukcie HMAC, ktorá sa používa pre overenie integrity dát, alebo naprogramovaním jadra ECC (*elliptic curve cryptography*) používaného pri výmene kľúča (viď Obr. 3.11)[19].



(a) Schéma zapojenia HMAC

(b) Schéma pre ECC

Obr. 3.11: Schémy implementácie

V porovnaní s predošlými prácami dosiahli efektívnejšie riešenie čo sa týka využitia prostriedkov. Riešenie implementovali na zariadeniach Xilinx Spartan-3, Spartan-6.

3.3 EAP a PANA

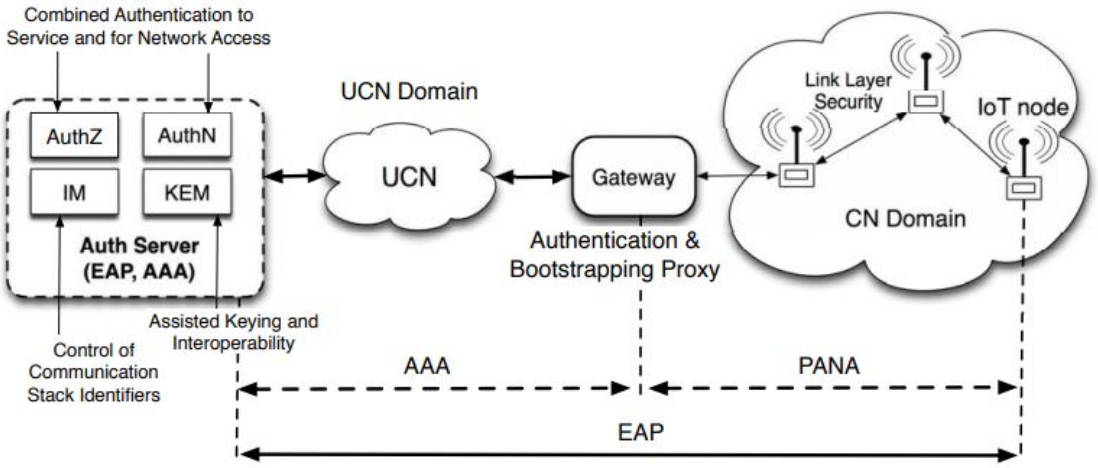
Jedným z možných používaných riešení pre zabezpečenie komunikácie v oblasti Internetu vecí je aj protokol PANA (*Protocol for Carrying Authentication for Network Access*) v

spojení s protokolom EAP (*Extensible Authentication Protocol*)[14]. EAP je používaný najmä v bezdrôtových sieťach Wi-Fi, ktorý zabezpečuje autentifikáciu zariadenia.

Architektúra pozostáva z troch častí [56] (viď Obr. 3.12):

1. autentifikačný server,
2. autentifikátor alebo taktiež brána (tzv. *Gateway*),
3. zariadenie IoT (IoT node).

Medzi bránou a autentifikačným serverom sa používa skupina protokolov označovaných ako AAA (*Authentication, authorization, accounting*) zvyčajne protokol Radius alebo Diameter [65]. Protokol PANA je založený na UDP protokole, používaný aj technológiou Zigbee, určený pre prenos správ protokolu EAP. Konkrétne v Internete vecí medzi zariadením IoT a bránou. To môžu zabezpečovať aj protokoly IKE a IPsec [31].



Obr. 3.12: Architektúra [12]

3.3.1 Štruktúra protokolu PANA

Komunikácia protokolu PANA pozostáva zo štyroch hlavných fáz [26].

- Prvotná fáza nazývaná discovery, ktorej účelom je pri zapojení zariadenia do siete nadviazať spojenie. Pozostáva z niekoľkých správ

```
IoT node      Gateway
        ----->      PANA-PAA-Discover(0,0)
```

```
        <-----      PANA-Start-Request(x,0)[Cookie]
        ----->      PANA-Start-Answer(y,x)[Cookie]
```

- Autentifikácia je zabezpečená protokolom EAP prostrdníctvom naledovných správ:

```
IoT node      Gateway
        <-----      PANA-Auth-Request(x+1,y)[EAP{Request}]
        ----->      PANA-Auth-Answer(y+1,x+1)[EAP{Response}]
        <-----      PANA-Auth-Request (x+2,y+1)[EAP{Request}]
        ----->      PANA-Auth-Answer (y+2,x+2)[EAP{Response}]
        <-----      PANA-Bind-Request(x+3,y+2)[EAP{Success}, Device-
            Id, Lifetime, Protection-Cap., MAC]
        ----->      PANA-Bind-Answer(y+3,x+3)[Device-Id, Protection-
            Cap., MAC]
```

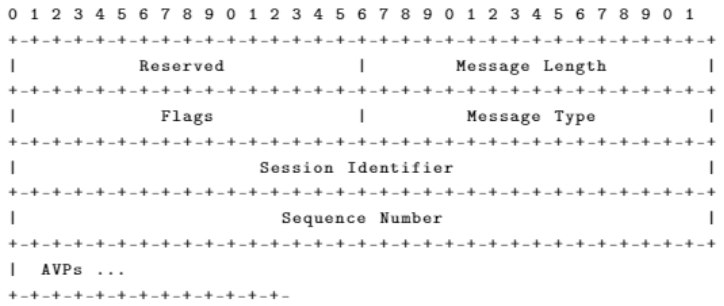
- Fáza reautentifikácie

```
IoT node      Gateway
        ----->        PANA-Reauth-Request(q,p)[MAC]
        <-----        PANA-Reauth-Answer(p+1,q)[MAC]
```

- Fáza ukončenia spojenia

```
IoT node      Gateway
        ----->      PANA-Termination-Request(q,p)[MAC]
        <-----      PANA-Termination-Answer(p+1,q)[MAC]
```

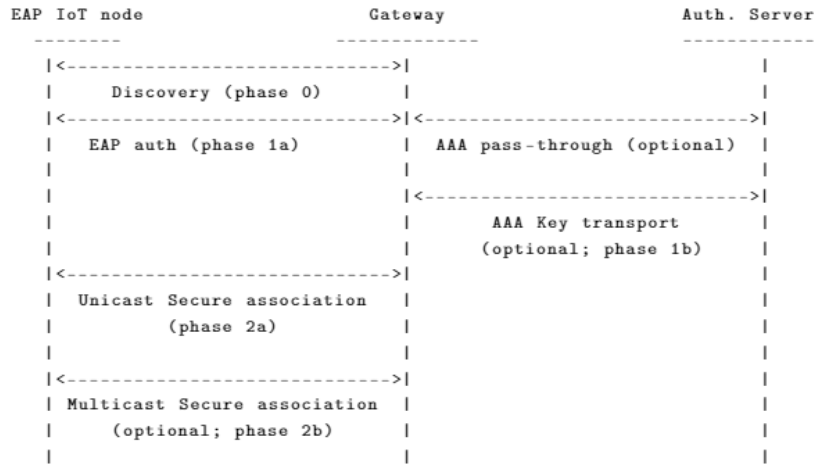
Na Obr. 3.13 môžeme vidieť štruktúru správy protokolu PANA.



Obr. 3.13: Štruktúra správy protokolu PANA

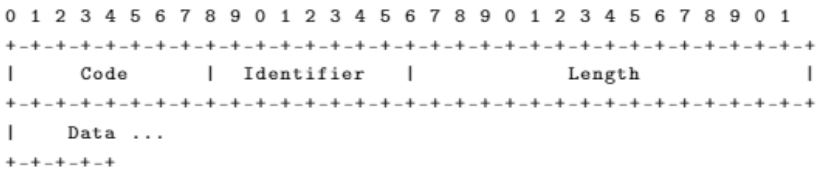
3.3.2 Štruktúra protokolu EAP

Protokol EAP, ktorý prebieha vo fáze autentifikácie protokolu PANA pozostáva z troch fáz (viď Obr. 3.14)[2]. EAP nedefinuje iba jedne spôsob autentifikácie, ale podporuje viacero metód ako EAP-PSK, ktorý je jeden z najvhodnejších kandidátov pre použitie v prostredí Internetu vecí, EAP-TLS/DTLS, EAP-AKA, EAP-SIM a iné.



Obr. 3.14: Fázy protokolu EAP

Štruktúra správy protokolu EAP, ktorý je prenášaný protokolom PANA, je zobrazený na Obr. 3.15.



Obr. 3.15: Štruktúra protokolu EAP

3.3.3 Energetická náročnosť

V tejto časti práce uvádzame energetickú náročnosť protokolov EAP a PANA. Jednotlivé zdroje uvádzajú nasledovné hodnoty (viď tabuľka 3.5 a 3.6).

V tabuľke 3.5 môžme vidieť energetickú náročnosť protokolu EAP/PANA uvedenú podľa jednotlivých kritérií [15]:

Tabuľka 3.5: Energetická náročnosť [15]

	EAP-PSK/PANA
Komunikačná náročnosť (bytes)	1380
Náročnosť kryptografických operácií (milliseconds)	49,90 / 22,75 (MAC/Kľúče)
Pamäťová náročnosť (bytes)	224
Celková náročnosť (microjoules)	10905

1. komunikačná náročnosť - určuje počet prenesených bajtov medzi zariadeniami PaC (*z angl. PANA client*) a PRE (*PANA Relay Element* - autentifikátor),

2. náročnosť kryptografických operácií - tu uvádzame generovanie a overovanie používaných kľúčov, či MAC (*z angl. Message Authentication Codes*), samotné šifrovanie a dešifrovanie je nenáročné (0 - 0.3 ms),

3. pamäťová náročnosť - určuje veľkosť využitej pamäte počas autentifikačných operácií,

4. celková energetická náročnosť vyjadrená v jednotkách microjoule.

V tabuľke 3.6 sú zobrazené energetické náročnosti dvoch metód poskytovaných protokolom EAP pre autentifikáciu. Jednotlivé stĺpce vyjadrujú počet odoslaných (TX) a prijatých (RX) packetov, či dát vyjadrených v bajtoch a celkový výsledný počet [56].

Tabuľka 3.6: Energetická náročnosť [56]

	TX packets	**TX data (B)**	**RX packets**	**RX data (B)**	**Total packets**	**Total data (B)**
EAP-MD5	3	66	3	59	6	125
EAP-PSK	5	181	4	160	9	341

3.3.4 Existujúce riešenia

V tejto kapitole opisujeme existujúce riešenia optimalizácie energetickej náročnosti protokolu EAP a PANA.

3.3.4.1 PANATIKI

V tomto riešení autori [65] [35] zjednodušili oba spomínané protokoly. Pri výbere metódy protokolu EAP zvolili metódu zdieľaného tajomstva (EAP-PSK) z dôvodu nižšej energetickej náročnosti v porovnaní s ostatnými. Svoje riešenie overili na reálnych zariadeniach, ako aj v simulátore Cooja.

V článku opisujú niekoľko optimalizácií:

1. optimalizácia stavového automatu protokolu EAP zahŕňa vymazanie premenných a stavov nepoužívaných pri IoT zariadeniach, vymazanie časového limitu (*timeout*) z dôvodu podporovania nižšou vrstvou protokolom PANA, vymazanie funkcionality ako notifikácie, ktoré nie je odporúčané používať s niektorými metódami protokolu EAP,
2. optimalizácia protokolu PANA vymazaním funkcionality súvisiacej s iniciovaním reautentifikácie bránou (možnosť neúspechu pri režimoch idle, power-down a pod.),
3. odstránením správ v protokole PANA, ktoré prenášajú len potvrdzujúce správy protokolu EAP,
4. využitím logiky pre oznamovanie o fungovaní klienta, odstránili funkcionalitu pre spracovanie správ odosielaných autentifikátorom,
5. vymazaním potvrdzovacích správ pri ukončení spojenia (využijú vypršanie časovača na autentifikátorovi s predpokladom, že je pripojený k zdroju napätia),
6. nahradením algoritmu *HMAC_SHA1* s algoritmom *AES-CMAC* podporovaným hardvérom vo väčšine IoT zariadeniach pre zachovanie integrity a odvodenie šifrovacích kľúčov,

3.3.4.2 Lightweight EAP-PK

Lightweight EAP-PK [43] je jednou z novo vytvorených metód. Autori sa však nezamerali na zníženie energetickej náročnosti, ale snažili sa vytvoriť riešenie, ktoré zabezpečí niektoré z bezpečnostných vlastností nepodporované doterajšími metódami protokolu EAP, ako jednoduchej implementácie protokolu, rýchlej autentifikácie, či odolnosť voči útokom DOS (*Deniel of Service* - Vyradenie služby) alebo MITM (*Man in the middle* - Muž v strede).

Samotný protokol neposkytuje všetky požiadavky definované vo forme RFC. Hlavné rozdiely s metódami používanými s protokolom EAP spočívajú v nasledovných bodoch:

1. zachovanie dôvernosti je zabezpečené používaním RSA, Pohlig-Hellman alebo iných menej známych algoritmov (A5/1, A5/2, A5/3),
2. protokol je založený na metóde autentifikácie výzva-odpoveď (*z angl. Challenge-response*) spočívajúcej v opýtaní sa otázky zariadenia a očakávania správnej odpovede, ktorá je implementovaná v tomto protokole s optimalizovaným, čo najmenším počtom správ,
3. protokol poskytuje možnosti pre odhalenie falošných prístupových bodov prostredníctvom porovnania MAC jednotlivých správ posielaných pri autentifikácii výzva-odpoveď.

3.4 HIP

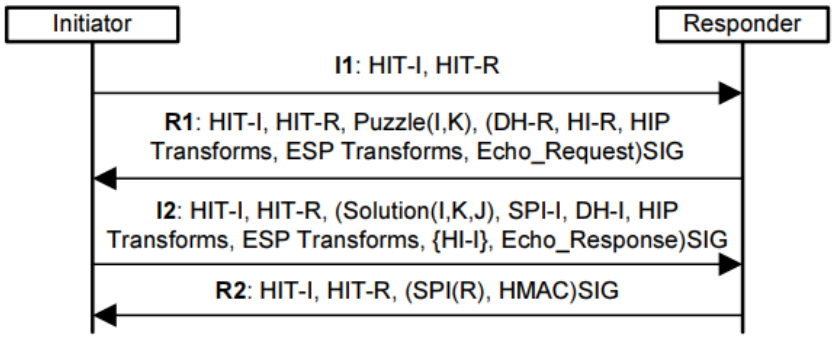
Ďalším z používaných protokolov v Internete vecí je aj protokol HIP (*z angl. Host Identity Protocol*) [50], ktorý je určený na výmenu kľúčov a pre použitie s inými protokolmi, ako napríklad s protokolom ESP/IPSec ako minimálnou možnosťou pre veľkosť paketu. Hlavičky protokolu HIP sa pre zmenšenie veľkosti odosielaných dát používajú len pri nadväzovaní spojenia a dohodnutí požadovaných parametrov. Ako riešenie môže byť porovnaný s protokolom IKE/IKEv2 [7].

3.4.1 Štruktúra protokolu HIP

Tento protokol pozostáva z dvoch komunikujúcich zariadení:

1. Iniciátor (v našom prípade IoT zariadenie)
2. Odpovedajúci / Reagujúci na odpoveď (server/brána)

Základná výmena správ pre dohodnutie parametrov a vytvorenie komunikačného kanála pozostáva zo štyroch základných správ zobrazených na Obr. 3.16 [7]. Po týchto správach nasleduje samotná výmena dát.

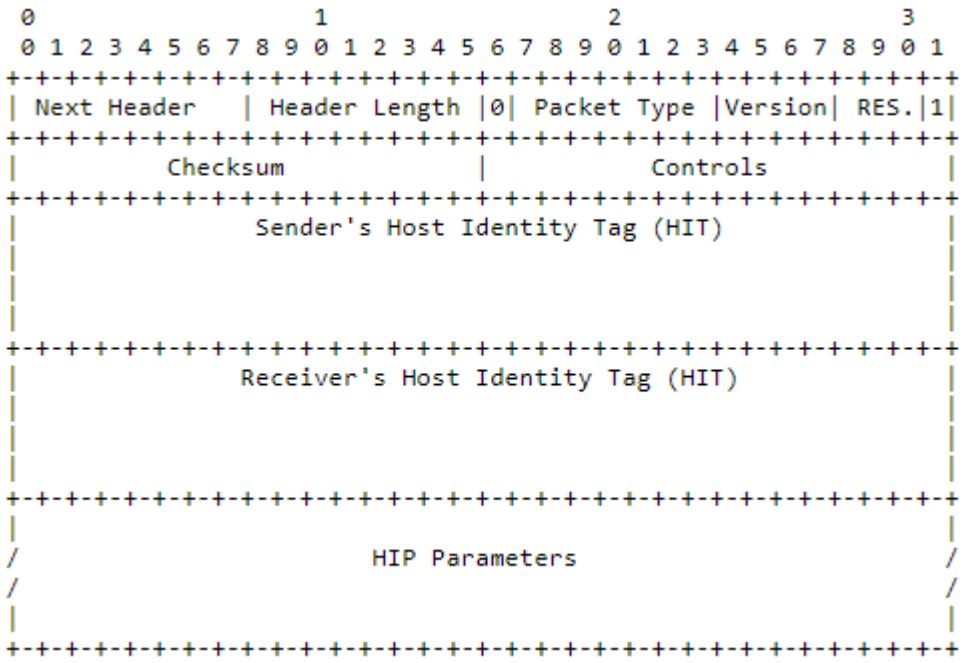


Obr. 3.16: Základná výmena správ protokolu HIP

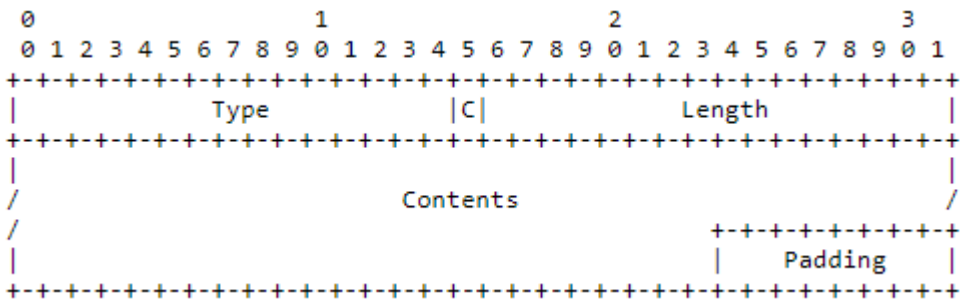
Správy protokolu sú označované podľa toho, kto ich odosiela. V prípade iniciátora ako I a v prípade odpovedajúceho ako R (*z angl. responder*) nasledované poradovým číslom (príklad: I1, R2). Význam jednotlivých prenášaných parametrov [7][53]:

- HIT (*z angl. host identity tag*, HIT-I, HIT-R) - 128-bitový hash verejného kľúča, ktorý slúži na overenie podpisov a identity zariadenia,
- Puzzle(I,K) - úloha pre iniciátora pre zdŕžanie odpovede I2,
- Solution(I,K,J) - odpoveď na vyriešenú úlohu puzzle,
- Diffie Hellman (DH-I, DH-R) - verejný kľúč,
- Host Id (HI-I, HI-R) - identita zariadenia (verejný kľúč),
- HIP Transforms - algoritmus používaný pre zvyšok výmeny protokolom HIP,
- ESP Transforms - algoritmus používaný protokolom IPSec,
- Echo_Request - dáta, ktoré majú byť odoslané späť správou Echo_Response, bez akejkoľvek modifikácie,
- Echo_Response
- SPI (*z angl. Security Parameter Index*) určený na identifikáciu inštancie protokolu HIP,
- HMAC - kontrolná suma správy,
- SIG - podpis správy.

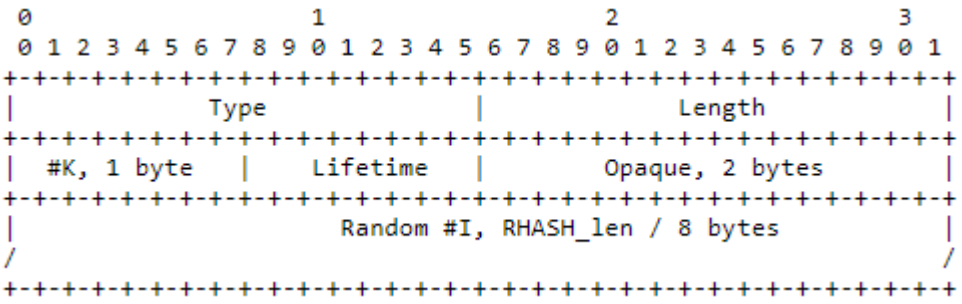
Formát jednotlivých správ a parametrov: Formát odosielaných paketov počas fázy dohadovania jednotlivých parametrov je zobrazený na Obr. 3.17. Okrem klasických polí väčšiny protokolov ako informácia o nasledujúcej hlavičke protokolu, veľkosť hlavičky, verzia protokolu, typ paketu, kontrolného súčtu, obsahuje dve veľké polia pre HIT-I a HIT-R. Za nimi už nasledujú jednotlivé potrebné parametre. Tie sú prenášané vo formáte TLV zobrazeného na Obr. 3.18. Každý parameter teda obsahuje informáciu o type a dĺžke. Jednotlivé typy označené číselne musia za sebou nasledovať vzostupne. Pri nedodržaní takejto podmienky sa paket považuje za modifikovaný. Pre zachovanie dĺžky, sa v prípade potreby pridáva na koniec zarovnanie. Pre príklad uvádzame formát parametru Puzzle (viď Obr. 3.19).



Obr. 3.17: Formát paketu protokolu HIP pri dohadovaní parametrov



Obr. 3.18: TLV formát



Obr. 3.19: Formát parametru Puzzle

Stavový diagram protokolou HIP: Proces spracovania jednotlivých správ počas vytvárania asociácie medzi iniciátorom a respondentom je opísaný pomocou stavového diagramu (viď Obr. 3.20). Ten zobrazuje jednotlivé stavy a prechody medzi nimi na oboch komunikujúcich zariadeniach.

Jednotlivé stavy:

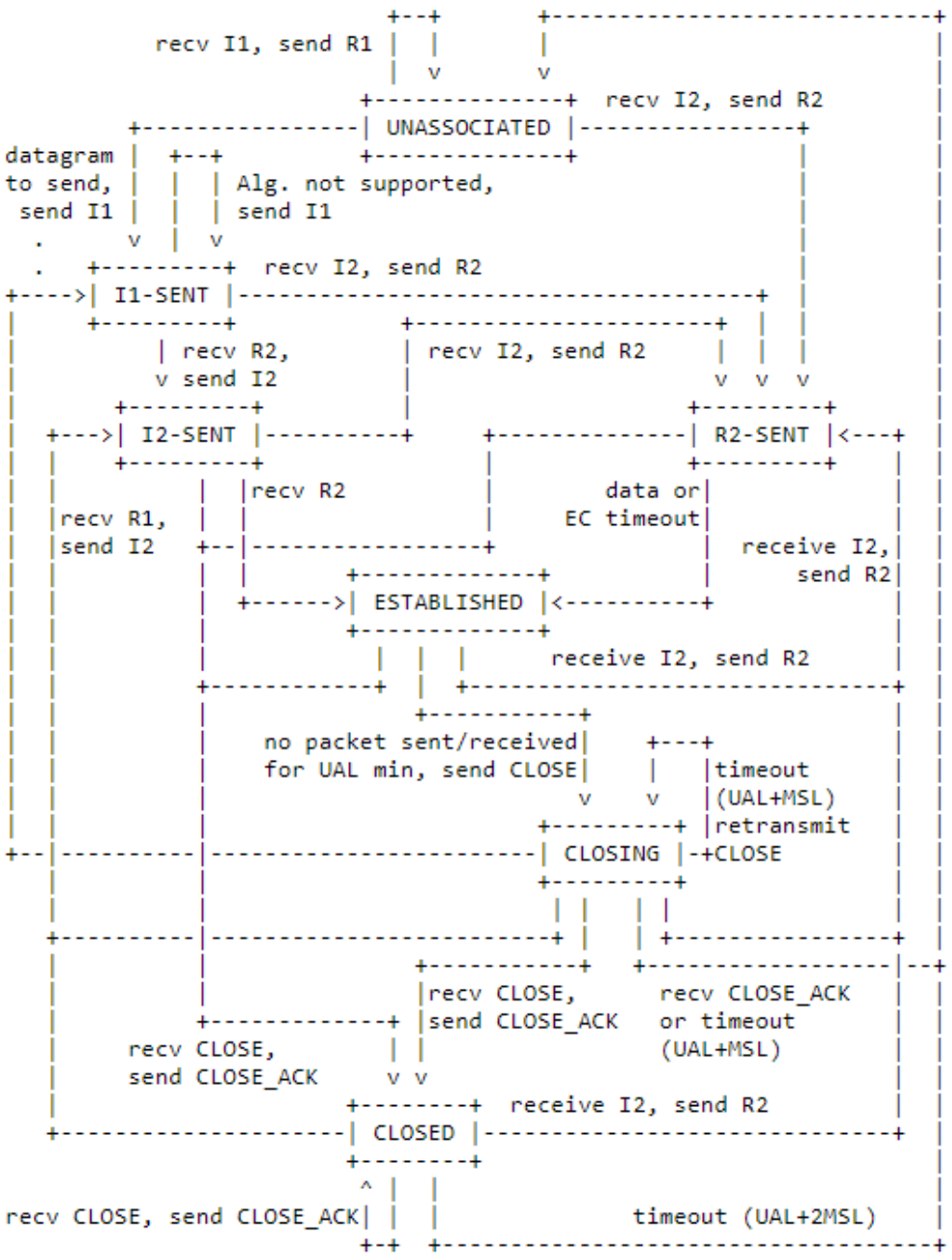
1. Unassociated - bez vytvoreného spojenia (dohodnutých parametrov) s komunikujúcou stranou,
2. I1-sent - v stave odoslania správy I1,
3. I2-sent - v stave odoslania správy I2,
4. R2-sent - v stave odoslania správy R2,
5. Established - vytvorené spojenie,
6. Closing - v stave ukončovania spojenia,
7. Closed - v stave ukončenia spojenia.

3.4.2 Energetická náročnosť protokolu HIP

V nasledujúcej tabuľke 3.7 uvádzame energetickú náročnosť jednotlivých modifikácii protokolu HIP z dostupnej uvedenej literatúry. Všetky hodnoty sú uvádzané v micro jouloch.

Tabuľka 3.7: Energetická náročnosť protokolu HIP (hodnoty uvedené v mJ) [62, 63]

Protokol	**Náročnosť na komunikáciu**	**Výpočtová náročnosť**	**Celková náročnosť**
Standard HIP	78,67	64,231	142,901
Ben-Saied et al.	94,55	3,43	97,98
C-HIP	47,186	63,622	110,808
D-HIP	86,59	15,17	101,76
D-HIP	48,14	0,76	48,9
CD-HIP	53,16	14,93	67,09
HIP-BEX	30,71	190,73	221,44



Obr. 3.20: Stavový diagram protokolu HIP

3.5 Zhodnotenie analyzovanej časti

Jednotlivé používané bezpečnostné protokoly, v oblasti Internetu vecí, sú za účelom znižovania energetickej náročnosti často-krát upravované a prispôsobené na konkrétne prípady použitia.
Existuje niekoľko rôznych spôsobov ako docieliť požadovaný výsledok. Niektoré zdroje nahrádzajú jednotlivé bezpečnostné mechanizmy za menej náročné, implementujú kryptografické algoritmy do hardvéru, znižujú náročnosť odstránením nepotrebných posielaných správ a parametrov, ktoré zaberajú väčšie množstvo miesta v pakete, zmenšujú hlavičky daných protokolov alebo znižujú náročnosť zmenou stavového diagramu, odstránením nadbytočnej, či duplikovanej funkcionality.

4. Špecifikácia požiadaviek

V tejto kapitole uvádzame zosumarizované požiadavky na modifikovaný algoritmus:

1. znížiť energetickú náročnosť protokolu HIP, inými slovami zvýšiť výdrž batérie a celkovú životnosť zariadenia IoT,
2. zachovanie bezpečnostných vlastností (integrity, diskrétnosti, autentifikácie, ...) protokolu HIP (D-HIP, HIP DEX) podľa tabuľky 2.2 opísanej v kapitole 2.4.1,
3. dosiahnutie zlepšenia oproti iným existujúcim riešeniam, porovnaním sa s ich dosiahnutými výsledkami v znížení energetickej náročnosti,
4. protokol musí fungovať správne, t.j. má fungovať bez chybových stavov,
5. protokol by sa mal jednoducho implementovať na IoT zariadenia,
6. v implementácii sa budú jednoducho vykonávať zmeny (pre jeho úpravu nebude potrebné zmeniť celú implementáciu, ale len vybranú časť kódu, implementácia bude čitateľná, bude obsahovať vhodné komentáre),
7. podpora fungovania protokolu s IPv4 alebo IPv6.

5. Návrh riešenia

V tejto časti diplomovej práce navrhneme spôsob úpravy protokolu HIP tak, aby sme splnili všetky stanovené požiadavky v zadaní, opísané v kapitole 4.

Možné návrhy:

1. Odstránenie potvrdzovacej správy pri ukončovaní spojenia medzi danými zariadeniami za určitých predpokladov.
   Odstránenie stavov - odstrániť stavy ako Closing, Closed (v prípade Closed použiť stav Unassociated), čo by mohlo znížiť celkový čas priebehu protokolu, čím by sme dokázali znížiť čas vykonania protokolu, a tým aj jeho celkovú energetickú náročnosť.

2. Redukcia formátu parametrov protokolu HIP - odstránenie polí typ a dĺžka pri formáte parametrov zadefinovaním ich pevnej dĺžky a poradia v jednotlivých prenášaných správach. Touto implementáciou by sme mohli zmenšiť celkovú veľkosť prenášaných správ, a tým znížiť náročnosť protokolu na komunikáciu.

3. Odstránenie parametru HI-R - v prípade, že paket obsahuje rovnakú informáciu v oboch poliach, a to DH-R (verejný kľúč) a HI-R, mohli by sme jeden z nich vynechať, čím by sme taktiež znížili veľkosť odosielanej správy.

4. Nahradenie parametru SPI - prostredníctvom IP adresy alebo parametra HIT-I, čo taktiež zredukuje veľkosť správy.

5. Odstránenie parametra HIT - odstránenie parametra príjemcu v pakete by znížilo celkovú veľkosť správy (závisí od použitia tohto parametra v predošlých návrhoch).

6. Výpočtové úlohy protokolu - pre jednotlivé úlohy ako Puzzle(), HMAC a podobne použiť hardvérovú podporu algoritmov alebo naprogramovať efektívne riešenie s použitím programovacieho jazyka assembler. Jednotlivé optimalizácie by mohli znížiť

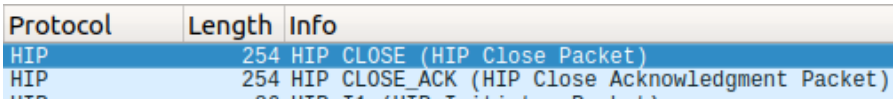
čas a výpočtovú náročnosť protokolu, a tým predĺžiť životnosť zariadenia.

Jednotlivé spomenuté navrhované a možné riešenia sú podrobne rozpracované a vysvetlené v nasledujúcich podkapitolách.

5.1 Odstránenie správy *CloseAck* a priebežného ukončovacieho stavu

Analýzou protokolu HIP sme zistili, že pri ukončovaní spojenia medzi komunikujúcimi zariadeniami sa odosielajú dve správy (viď Obr. 5.1), a to:

- Close - ktorá signalizuje zariadeniu ukončenie spojenia,
- CloseAck - ktorá slúži pre potvrdenie o ukončení spojenia medzi komunikujúcimi uzlami.



Obr. 5.1: Ukážka ukončenia spojenia

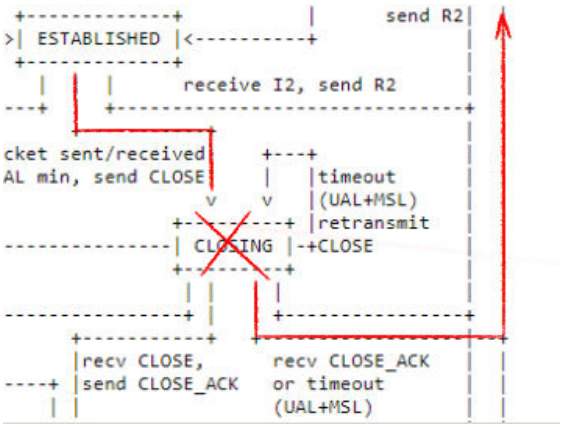
Navrhujeme odstrániť jednu z týchto správ, konkrétne správu CloseAck, čím ušetríme:

1. procesorový čas na vykonanie inštrukcií pre spracovanie prijatej správy CloseAck,
2. čas, kedy zariadenie čaká na potvrdzujúcu správu CloseAck od odoslania správy Close, a to prechodom do stavu Ukončený (*Closed*), prípadne stavu Neasociovaný (*Unassociated*).

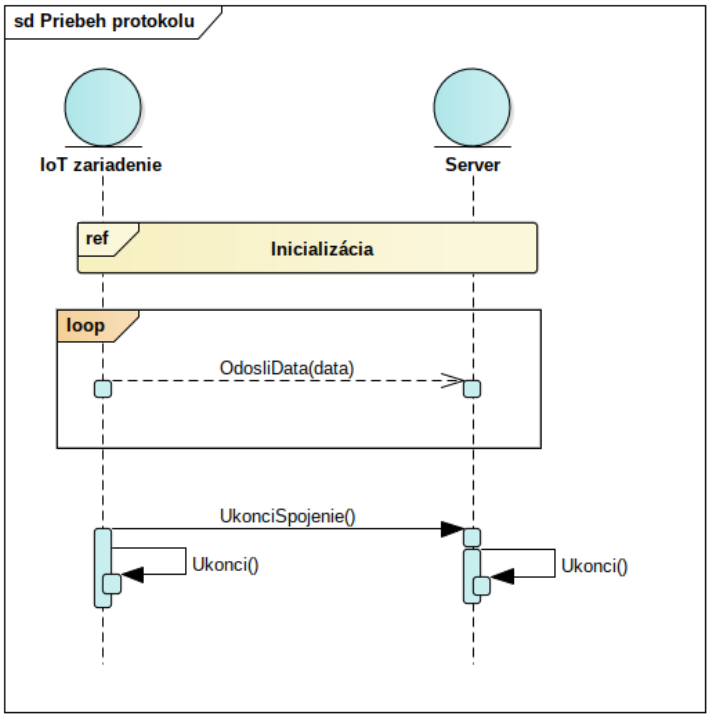
S bodom 2, zobrazeným na Obr. 5.2, súvisí aj návrh Odstránenia stavov spomenutých v úvode tejto kapitoly 5.

Opísané riešenie navrhujeme pre prostredie Internetu vecí, kde predpokladáme komunikáciu servera priamo pripojeného k zdroju a IoT zariadenia napájaného batérie. Počas priebehu ukončovania spojenia môžu nastať určité situácie:

1. IoT zariadenie ukončí spojenie odoslaním správy Close, server ukončí spojenie prijatím tejto správy. A naopak.



Obr. 5.2: Zobrazenie prechodu odstránením stavu CLOSING



Obr. 5.3: Priebeh protokolu

2. IoT zariadenie ukončí spojenie odoslaním správy Close, avšak server nedostane prijatú správu a neukončí spojenie. Počká na vypršanie časovača neprijatím žiadnej správy od IoT zariadenia.

Tým, že predpokladáme napájanie servera priamo zo zdroja napájania, nám nemusí

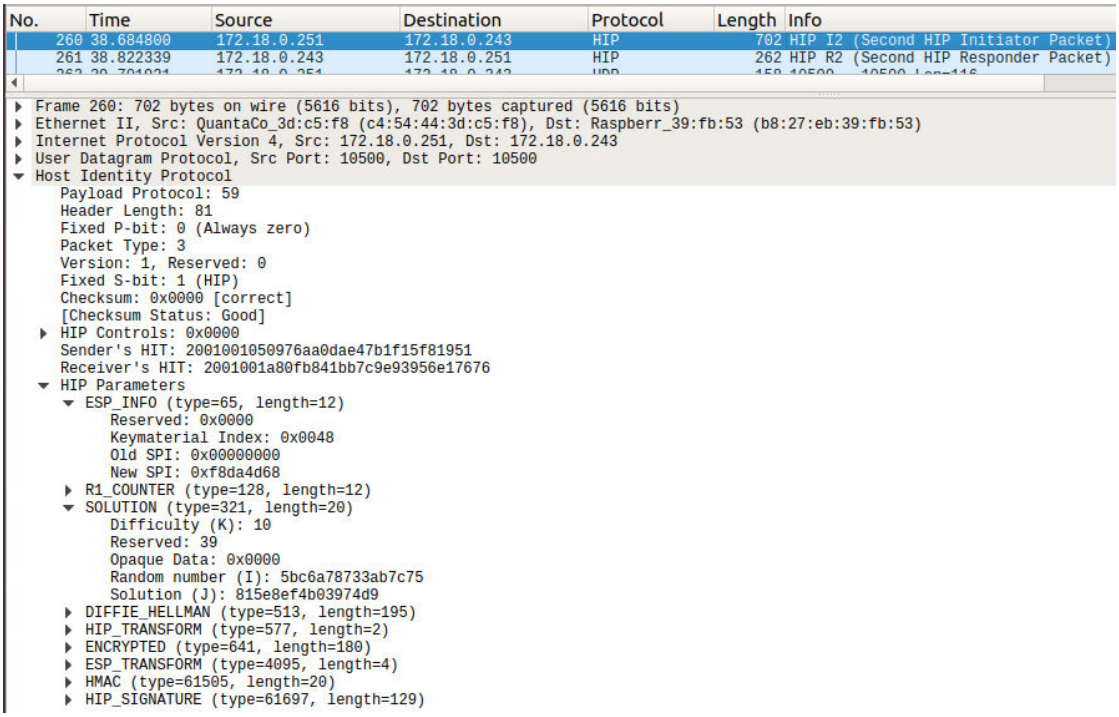
vadiť čakanie na ukončenie spojenia na serveri až do vypršania časovača. Nevýhodou však môže byť opačná situácia, kedy by musel časovač vypršať práve na danom IoT zariadení. Z hľadiska Internetu vecí je vhodné, ak dané zariadenie sa pripojí k serveru, odošle dané údaje a následne ukončí spojenie. Je preto nutné uviesť ďalší predpoklad, a to iniciovanie ukončenia spojenia len IoT zariadením (viď Obr. 5.3).

5.2 Redukcia formátu parametrov protokolu HIP

Počas procesu vytvárania asociácie sa odosielajú štyri správy: I1, R1, I2, R2.

Zaujímavejšie sú pre nás práve R1 a I2, ktoré prenášajú viacej parametrov. R1: Puzzle(), DH-R, HI-R, HIP, Transform, ESP Transforms, Echo_Request, SIG. Všetky HIP parametre so sebou nesú informáciu o aktuálnom type a jeho dĺžke.

Príklad zobrazenia odchytenej správy typu I2 protokolu HIP nástrojom *Wireshark* môžme vidieť na Obr. 5.4, kde nás podrobnejšie zaujíma časť paketu HIP Parameters.



Obr. 5.4: Zobrazenie správy typu I2.

V riešení navrhujeme odstrániť polia typ a dĺžka pri niektorých parametroch s pevným

zadefinovaním ich poradia alebo zadefinovaním ich dĺžky. Odosielaný paket by teda mohol vyzerať nasledovne:

```
Sender's HIT: 2001001e2c023cbf8e65ee2b05a42820
Receiver's HIT: 2001001c009d1d347d57bd541d10a393
HIP Parameters:

ESP_INFO
    Reserved: 0x0000
    Keymaterial Index: 0x0048
    Old SPI: 0x00000000
    New SPI: 0xf8da4d68
R1_COUNTER
    Reserved: 0x00000000
    R1 Counter: 0000000000001045
SOLUTION
    Difficulty (K): 10
    Reserved: 39
    Opaque Data: 0x0000
    Random number (I): 23c8b08466518471
    Solution (J): 847123c8b0846651
DIFFIE_HELLMAN
    3 (1536-bit MODP group) Public Value Length: 192 Public Value:
        c6d90a4e31a12b297b00162e7ce87d4eac71f53e032a7088...
...
```

Podobne by sme spracovali aj ostatné typy správ.Navrhovaným riešením by sme mohli zmenšiť prenášaný paket pri uvedenom príklade o 32B.

Pri prototypovom riešení zefektívnenia protokolu sa nám podarilo daným riešením zmenšiť typ správy I2 o 16B.

5.3 Odstránenie parametru HI-R

V návrhu riešenia sme uviedli Odstránenie parametru HI-R - v prípade, že paket obsahuje rovnakú informáciu v oboch poliach, a to DH-R (verejný kľúč) a HI-R (*z angl. Host identity - Identita hosta*), mohli by sme jeden z nich vynechať, čím by sme taktiež znížili veľkosť odosielanej správy.

Tento návrh sme však po detailnejšej analýze týchto parametrov upravili, pretože HI-R

nesie informáciu, konkrétne parametre verejného kľúča, pre použité asymetrické šifrovania (napr. RSA, DSA, ...). Parametre:

1. informácia o veľosti exponenta,
2. exponent,
3. modulus.

Bezpečnosť algoritmu RSA spočíva najmä vo veľkosti parametra modulus. Počas prototypovania riešenia sme v odosielaných správach mali veľkosť 128B (čo je 1024 bitov). V súčastnosti je odporúčaná dĺžka pre modulus 2048 alebo 4096 bitov. Pri použití takejto veľkosti by sme vynaložili veľké množstvo energie na odosielanie správy obsahujúcej tento parameter.

5.3.1 Navrhované riešenia

Manuálne - Manuálne riešenie pozostáva z vygenerovania a nahratia uvedených parametrov do pamäte zariadenia, čím by sme pri prototypovom zefektívňovaní protokolu ušetrili 128B z celkovej veľkosti správy 638B.

Možné výhody/nevýhody riešenia:

- Potreba znovu-nahratia verejných parametrov servera na zariadenia, čo môže byť náročné pri veľkom počte IoT zariadení umiestnených na rôznych miestach.
- Výhodné v prípade, ak by sa parametre znovu vygenerovali a nahrali do zariadenia počas jeho nabíjania alebo výmeny batérie.
- Nutnosť generovania parametrov súčasne (v rovnakom čase) na všetky zariadenia komunikujúce s daným serverom - nevýhodné.

Vyplývajúce nutné predpoklady:

- používateľ/administrátor má jednoduchý prístup ku všetkých IoT zariadeniam (príklad: v rámci jednej budovy),
- nie je potrebná nepretržitá činnosť IoT zariadení (t.z. možnosť zastaviť IoT zariadenia na určitý čas).

Automatické - Pri tomto riešení nie je potrebné uvažovať zásah človeka. Riešenie spočíva v pregenerovaní daných parametrov po určitom čase.

Princíp fungovania:

1. Odoslanie a nahratie parametrov do pamäte zariadenia pri prvej komunikácii.
2. Používanie rovnakých parametrov verejného kľúča po určitú dobu zadefinovaním jeho platnosti uložením dátumu ich nahratia do pamäte. (Pozn.: Platnosť parametrov ostáva aj pri reštartovaní a znovu inicializovaní celého protokolu.)
3. Po vypršaní platnosti pregenerovanie, preposlanie a uloženie nových parametrov.

Podľa organizácie NIST sa neodporúča použitie 1024-bitového kľúča [10], preto navrhujeme pri manuálnom riešení použiť minimálne kľúč o veľkosti 2048b. V súčastnosti však nie je známe prelomenie 1024b. RSA kľúč o veľkosti 768b bol faktorizovaný v roku 2009, čo však trvalo viac ako 2 roky [37]. Preto určenie doby platnosti môže byť náročné. Za daných skutočností je potrebné zvážiť použitie automatického riešenia spolu so životnosťou použitej batérie pre dané IoT zariadenie.

Pri oboch navrhovaných riešeniach je potrebné predpokladať zabezpečenie neautorizovaného prístupu k IoT zariadeniu, keďže ukladáme verejný kľúč do pamäte zariadenia. Riešením by mohlo byť jeho uchovávanie v šifrovanej podobe čím by sme jednak zvýšili energetickú náročnosť, čo nie je našim cieľom. Druhou nevýhodou riešenia je uľahčenie útočníkovi poznať typ šifrovania, pretože pre návrh a implementáciu uvedených v tejto práci sme použili voľne dostupnú implementáciu protokolu (t.j. úprava existujúceho riešenia).

Každé navrhované riešenie je vhodné použiť pre iné prípady použitia.

Príklad aplikácie pre:

- manuálne riešenie - Monitorovanie stavu pacientov v nemocnici počas dňa,
- automatické - Monitorovanie naplnenosti kontajnerov v rámci jedného mesta.

5.4 Odstránenie parametra HIT

Teoretická časť bola vypracovaná na základe RFC dokumentov [51, 50, 49].

Parameter HIT (*z angl. Host Identity Tag*) reprezentuje skrátenú identitu zariadenia, ktorá je hašom verejného kľúča, pomocou ktorého dokážeme identifikovať konkrétne komunikujúce zariadenie. Jeho 128-bitová veľkosť umožňuje vytvorenie jedinečnej adresy pre zariadenie, rovnako ako adresovanie IPv6.

Návrhom bolo odstrániť tento parameter z hlavičky protokolu HIP v kontrolných paketoch a namiesto neho použiť iný identifikátor zariadenia z nižších vrstiev IP/OSI modelu, napríklad IPv6 adresu. Všetky štyri inicializačné správy protokolu HIP obsahujú dve polia s parametrom HIT (HIT odosielateľa, HIT príjemcu). Jeho odstránením by sme zmenšili prenášaný paket o 2x16B. Uvedenie pre jednotlivé typy správ pri prototypovom riešení:

1. I1: 32B z 86B (37%),
2. R1: 32B z 638B (5%),
3. I2: 32B z 702B (4.5%),
4. R2: 32B z 262B (12.2%).

Keďže daný parameter zabezpečuje rôzne bezpečnostné funkcionality (napr. vytvorenie-/overenie podpisu správy), jeho odstránenie v implementácii by bolo náročné. Podobné riešenie je však priamo podporované pre dátové pakety protokolu HIP, kde identifikácia paketu ku konkrétnej inštancii protokolu HIP bola nahradená parametrom SPI (*z angl. Security Parameter Index*).

5.5 Nahradenie parametru SPI

Samotné riešenie protokolu HIP pozostáva z niekoľkých iných bezpečnostných protokolov. Jedným z nich je aj protokol IPSec. Minimálnou požiadavkou pre prenos dát v implementácii protokolu je použitie ESP transportného režimu. Ten používame aj v prototype riešenia. Transportný režim ESP obsahuje SPI pre mapovanie inštancie protokolu a zodpovedajúcej SA (*z angl. Security Associations*) k danému prisluchajucému zariadeniu identifikovaného parametrom HIT [51, 50, 49].

Nami navrhované riešenie je odstrániť parametre ESP INFO v riadiacich paketoch protokolu HIP, čím by sme ušetrili spolu 24B v správach I2 a R2.

Nevýhody návrhu:

1. Parameter SPI je používaný na mapovanie inštancie protokolu a zariadenia, čo sme spomenuli v predchádzajúcej kapitole.
2. Podľa dokumentácie protokolu je zakázané mapovať inštanciu na IP adresu, čo by mohlo nastať v prípade, ak by sme implementovali súčasne tento aj predošlý návrh odstránenie parametru HIP.
3. Obmedzenie možného rozšírenia protokolu vytvorením viacerých SA identifikovaných rôznymi SPI k danému zariadeniu.

5.6 Optimalizácia výpočtových úloh protokolu

Jedným z možných návrhov optimalizácie protokolu, ktorému sa venujeme, je optimalizácia výpočtových úloh (napr.: výpočet Puzzle, výpočet SHA, ...).

Optimalizácia asemblerom - uvažovali sme nad implementáciou niektorých algoritmov v programovacom jazyku asembler, čím by sme mohli znížiť energetickú náročnosť znížením využitia procesora, či lepším využívaním jeho prostriedkov. Pri tomto návrhu je však potrebné zvážiť jeho implementáciu, keďže v súčasnosti existujú kvalitné kompilátory s optimalizáciou strojového kódu.

Optimalizácia pomocou hardvéru - v tomto riešení navrhujeme implementácie protokolu na zariadeniach, ktoré obsahujú hardvérovú podporu pre výpočet náročných kryptografických, hašovacích a iných úloh. Ich použitím by sme taktiež znížili čas využitia procesora, a tým aj jeho energetickú náročnosť.

V prípade tohto navrhovaného riešenia, odporúčame optimalizáciu pomocou hardvéru.

6. Implementácia

Jednou z častí tejto práce je realizácia navrhovaného riešenia opísaného v kapitole 5. V jednotlivých podkapitolách je opísaná realizácia prototypu, nasadenie na IoT zariadenia s použitím jednej z bezdrôtových technológie - Bluetooth a opis implementácie jednotlivých navrhnutých častí optimalizácie protokolu. Rozhodli sme sa pre implementáciu nášho riešenia do už existujúcej, voľne dostupnej (licencia *MIT*) implementácie protokolu HIP (názov: *openHIP*) [1, 55]. Nami navrhnuté riešenie a implementáciu sme nazvali *E-HIP*.

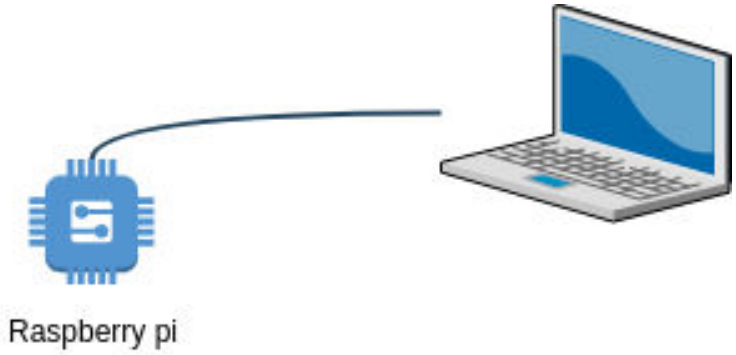
6.1 Prototypové riešenie E-HIP

Počas návrhu jednotlivých čiastočných optimalizácii bolo potrebné overiť ich správnosť a funkčnosť.

V prototype boli implementované nasledujúce časti:

- odstránenie správy CloseAck a priebežného ukončovacieho stavu,
- redukcia formátu parametrov protokolu HIP,
- odstránenie parametru HI-R (Manuálne riešenie).

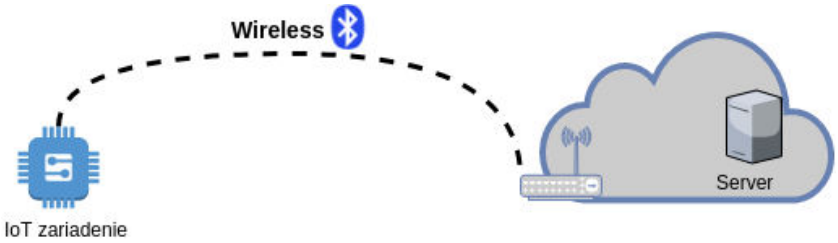
Toto riešenie prototypu bolo postavené na topológii zobrazenej na Obr. 6.1, ktorá pozostáva z dvoch uzlov siete, konkrétne malého počítača *Raspberry Pi* (operačný systém: Raspbian) a prenosného počítača (operačný systém: Ubuntu 18.04 LTS), prepojenými sieťovým ethernetovým káblom.



Obr. 6.1: Topológia zapojenia

6.2 Realizácia riešenia E-HIP

Pre otestovanie funkčnosti navrhnutého riešenia v časti 5 v prostredí Internetu vecí, sme sa rozhodli pre nasledujúcu architektúru zobrazenú na Obr. 6.2, kde cloud so serverom a Bluetooth bránou je repreyentovaný počítačom.

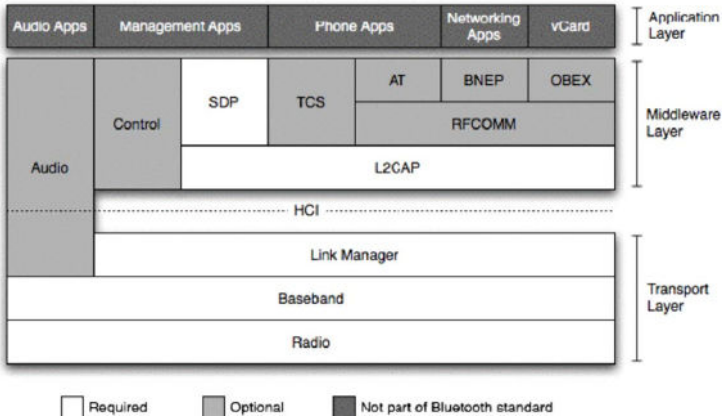


Obr. 6.2: Testovacia architektúra

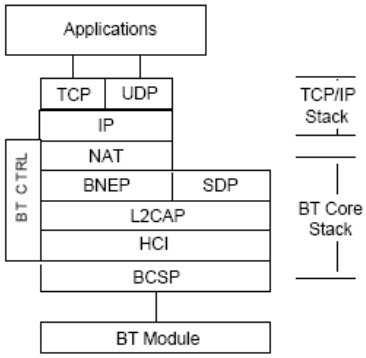
Vytvorené testovacie prostredie IoT pozostáva z nasledovných súčastí:

- IoT zariadenie s Unix OS,
- modul Bluetooth 4.1,
- server s Unix OS.

Protokol OpenHip funguje nad IP sieťou, a preto je potrebné zabezpečiť IP konektivitu medzi dvoma komunikujúcimi uzlami. Pre jej zabezpečenie sme sa rozhodli vytvoriť PAN sieť (*z angl. Personal Area Network*) s použitím protokolu BNEP (*z angl. Bluetooth Network Encapsulation Protocol*), ktorý dokáže enkapsulovať ethernetové rámce pre prenos cez Bluetooth sieť. Použitie protokolov na jednotlivých vrstvách zásobníka je zobrazené na Obr. 6.3b. Na Obr. 6.3a je zobrazený zásobník pre technológiu Bluetooth.



(a) Protokolový zásobník technológie Bluetooth[30]



(b) Časť zásobníka Bluetooth a TCP/IP[21]

Obr. 6.3: Bluetooth zásobník

V riešení používame priamo verziu technológie Bluetooth 4.1 z dôvodu priamej dostupnosti na zariadeniach Raspberry Pi. V budúcnosti by však bolo vhodnejšie použiť najnovšiu verziu Bluetooth 5.0 [4], ktorá je optimalizovaná a určená aj pre použitie v IoT sieťach. Pre jeho vhodnosť uvádzame aj porovnanie Bluetooth 5.0 a 4.1 v tabuľke 6.1[73].

Tabuľka 6.1: Porovnanie verzie Bluetooth 5.0 a 4.1

Vlastnosť	**Bluetooth 5.0**	**Bluetooth 4.1**
Rýchlosť	2 Mbps	1 Mbps
Dosah	40 m vo vnútri budov	10 m vo vnútri budov
Výkonové požiadavky	Nízke	Vysoké
Veľkosť správ	255 B	31 B
Odolnosť fungovať v husto pokrytom prostredí	Vyššia	Nižšia
Životnosť batérie	Dlhšia	Kratšia
Bezpečnosť	Väčšia	Menšia
Teoretická dátová priepustnosť	2 Mbps (reálne 1.6 Mbps s režijnými nákladmi)	1 Mbps
Spoľahlivosť	Vysoká	Nízka
Digitálna životnosť	Lepšia	Horšia oproti BT 5.0

6.3 Implementácia návrhu

V tejto časti kapitoly 6 uvádzame konkrétnu implementáciu navrhnutých optimalizácií z časti 5.

6.3.1 Odstránenie správy CloseAck a priebežného ukončovacieho stavu

Hlavnou náplňou tejto optimalizácie je odstrániť správu *CloseAck*, ktorá potvrdzuje ukončenie komunikácie medzi dvoma uzlami a stavu, kde zariadenie čaká na prijatie potvrdzujúcej správy. Celá implementácia spočíva v upravení funkcie *hip_handle_close(...)*, ktorá spracováva správu *Close* v súbore *src/protocol/hip_input.c*. Je potrebné:

1. odstrániť časť funkcionality, kde sa odosiela a kontroluje správa *CloseAck*,
2. odstrániť časť, kde sa nastavuje stav protokolu *Ukončný/Zatvorený (CLOSED)*,
3. presunúť volanie funkcie *delete_associations* za volanie nastavenia stavu *Neasociovaný (UNASSOCIATED)*.

6.3.2 Redukcia formátu parametrov protokolu HIP

Táto časť pozostáva z nasledovných úprav:

1. Úprava štruktúr použitých pre vyskladanie jednotlivých správ a odstránením nepotrebných polí. Tieto štruktúry sa nachádzajú v súbore *src/include/hip/hip_types.h* označené príponou k názvu _s. Príklad:

```
typedef struct _tlv_r1_counter_s
{
  __u32 reserved;
  __u64 r1_gen_counter;
} tlv_r1_counter_s;
```

2. Úprava vyskladania a odoslania samotnej správy. Odosielané správy sa nachádzajú v súbore *src/protocol/hip_output.c*. Konrétne sme upravili kontrolnú správu *I2* vo funkcii *int hip_send_I2(...)* kde bolo potrebné vykonať následné úpravy:
   - Nahradiť pôvodné štruktúry za upravené aj vo volaných funkciách.

- Odstrániť časti kódu, kde sa používajú odstránené súčasti pôvodných štruktúr.
- Upraviť pozície jednotlivých štruktúr vo výslednej vyskladanej správe.

3. Úprava prijatia správy *I2*, funkcie *int hip_parse_I2* v súbore *src/protocol/hip_input.c*. Predošlé spracovanie bolo implementované v cykle, kde sa postupne vyberali jednotlivé položky správy a následne na základe typu spracovali. Pre nami navrhované riešenie bolo potrebné implementovať pevné poradie spracovania jednotlivých parametrov správy a upraviť posúvanie na začiatok každého parametra.

6.3.3 Odstránenie parametru HI-R (Manuálne riešenie)

Implementácia odstránenia tohto parametru spočíva v odstránení volania funkcie *build_tlv_hostid*, ktorá ho prídáva do výsledného paketu. Následne je potrebné upraviť príjem tejto správy v súbore *src/protocol/hip_input.c*, konkrétne funkcii *int hip_parse_R1()*. Tu sme nahradili načítanie parametra *HostID* z prijatej spávy, načítaním zo súbora označeného ako *known_hi_filename*, ktorý obsahuje informácie (verejný kľúč pre RSA, ...) o serveri.

Pre jednoduchší prenos verejného kľúča zo servera na zariadenie sme upravili aj súbor *src/util/hitgen.c*, ktorý generuje verejný a súkromný kľúč pre zariadenie a príslušné súbory. Doplnili sme uloženie verejného kľúča (čísla N a E) do súboru pre následné zdieľanie.

7. Testovanie a overenie riešenia

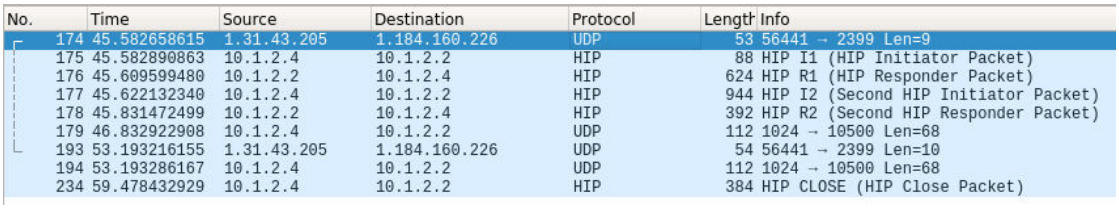
V tejto kapitole uvádzame výsledky testovania a overenia funkčnosti implementovaného riešenia optimalizácie bezpečnostného protokolu OpenHip:

1. Overenie funkčnosti
2. Overenie zefektívnenia vzhľadom na počet prenesených bajtov
3. Overenie zefektívnenia pri použití väčšieho kľúča
4. Overenie zefektívnenia vzhľadom na odstránený stav
5. Overenie z hľadiska spotreby energie
6. Overenie bezpečnostných vlastností
7. Porovnanie s existujúcimi riešeniami
8. Záver testovania

7.1 Overenie funkčnosti

V prvej fáze testovania sme overili celkovú funkčnosť nami optimalizovaného protokolu OpenHIP, a to nadviazaním spojenia, odoslaním niekoľkých dát a následným ukončením spojenia. Na Obr. 7.1 je zobrazená odchytená komunikácia pomocou nástroja *Wireshark*. Popis jednotlivých zobrazených paketov:

- 174 - odoslanie dát nad protokolom HIP,
- 175 - 178 - inicializácia a nadviazanie spojenia,
- 179 - odoslanie šifrovaných dát cez Bluetooth,



No.	Time	Source	Destination	Protocol	Length	Info
174	45.582658615	1.31.43.205	1.184.160.226	UDP	53	56441 → 2399 Len=9
175	45.582890863	10.1.2.4	10.1.2.2	HIP	88	HIP I1 (HIP Initiator Packet)
176	45.609599480	10.1.2.2	10.1.2.4	HIP	624	HIP R1 (HIP Responder Packet)
177	45.622132340	10.1.2.4	10.1.2.2	HIP	944	HIP I2 (Second HIP Initiator Packet)
178	45.831472499	10.1.2.2	10.1.2.4	HIP	392	HIP R2 (Second HIP Responder Packet)
179	46.832922908	10.1.2.4	10.1.2.2	UDP	112	1024 → 10500 Len=68
193	53.193216155	1.31.43.205	1.184.160.226	UDP	54	56441 → 2399 Len=10
194	53.193286167	10.1.2.4	10.1.2.2	UDP	112	1024 → 10500 Len=68
234	59.478432929	10.1.2.4	10.1.2.2	HIP	384	HIP CLOSE (HIP Close Packet)

Obr. 7.1: Odchytená komunikácia

- 193 - opakované odoslanie dát nad protokolom HIP,
- 194 - opakované odoslanie šifrovaných dát cez Bluetooth,
- 234 - ukončenie spojenia.

Z odchytenej komunikácie vyplýva, že optimalizovaný protokol funguje správne, t.j. dokáže nadviazať spojenie medzi dvoma komunikujúcimi uzlami, odoslať dátové správy a následne ukončiť spojenie.

7.2 Overenie zefektívnenia vzhľadom na počet prenesených bajtov

V tejto časti testovania sme sa zamerali na porovnanie pôvodného protokolu s upravenou verziou pri použití 1024-bitového kľúča. Zamerali sme sa najmä na počet odoslaných (TX) a prijatých (RX) správ/paketov a bajtov, podobne ako aj v iných článkoch [15, 56], na nasledovnom scenári:

1. nadviazanie spojenia, ktoré pozostáva zo 4 správ (I1, R1, I2, R2),
2. ukončenie spojenia, pozostávajúce zo správ Ukončenia / Podvrdenie Ukončenia.

Overenie sme realizovali pomocou nástroja *Wireshark*. Jednotlivé namerané výsledky sme zobrazili v nasledujúcich grafoch (Obr. 7.2 a 7.3) a tabuľkách (tabuľky 7.1 a 7.2). Pre daný scenár sme vykonali dve merania. Kvantitatívne merania by neukázali rôzne výsledky, pretože sa odosiela vždy rovnaký počet paketov a bajtov.

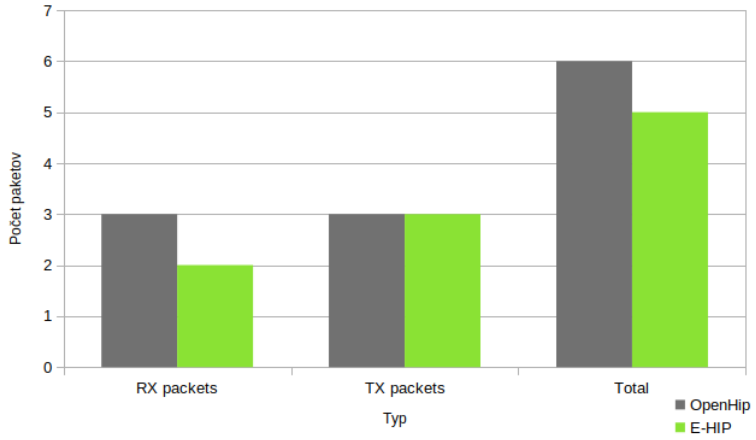
V závislosti od vykonaného testovania a dosiahnutých výsledkov môžme usúdiť, že celková energia potrebná na prenos určitého množstva bajtov (t.j. komunikačná náročnosť) je menšia pri porovnaní pôvodného a nami upraveného protokolu.

Tabuľka 7.1: Porovnanie počtu paketov

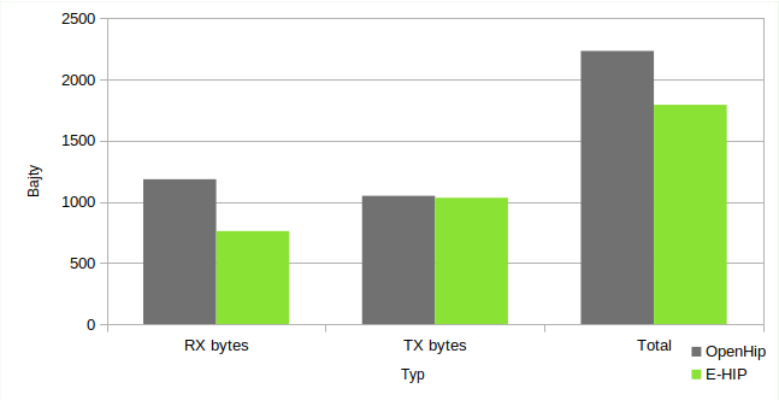
	RX pakety	**TX pakety**	**Celkovo**
OpenHip	3	3	6
E-Hip	2	3	5
Zefektívnenie (%)	33	0	16,6

Tabuľka 7.2: Porovnanie počtu bajtov

	RX bajty	**TX bajty**	**Celkovo**
OpenHip	1184	1048	2232
E-Hip	760	1032	1792
Zefektívnenie (%)	35,81	1,53	19,71



Obr. 7.2: Porovnanie počtu prenesených bajtov



Obr. 7.3: Porovnanie počtu paketov

7.3 Overenie zefektívnenia pri použití väčšieho kľúča

Na základe navrhnutého riešenia v časti 5.3.1, sme sa rozhodli otestovať jeho funkčnosť s použitím rôznych veľkostí RSA verejných kľúčov. Scenár testovania pozostával z nadviazania spojenia medzi zariadeniami. V tabuľke 7.3 uvádzame jednotlivé veľkosti správy R1 v bajtoch pre 1024 a 2048 bitový kľúč.

Tabuľka 7.3: Porovnanie veľkosti správy R1 pre 1024 a 2048-bit kľúč

	1024-bit	**2048-bit**
OpenHip (B)	664	920
E-Hip (B)	496	624
Rozdiel (B)	168	296
Zefektívnenie (%)	25,30	32,17

Predpokladané zefektívnenie pre 4096 bitový kľúč je 38%. V tabuľke 7.4 sme sa zamerali na celkovú komunikačnú náročnosť, vzhľadom na počet prenesených bajtov v jednotlivých správach protokolu OpenHip.

Tabuľka 7.4: Porovnanie počtu bajtov pre jednotlivé správy (1024 a 2048-bit kľúč)

	I1	**R1**	**I2**	**R2**	**Close**	**CloseAck**	**Celkovo**
OpenHip (1024-bit)	88	664	704	264	256	256	**2232**
E-Hip (2048-bit)	88	624	944	392	384	0	**2432**
OpenHip (2048-bit)	88	920	960	392	384	384	**3128**

Použitím 2048-bitového kľúča pre RSA, zvýšime celkovú bezpečnosť protokolu, čím sa samozrejme zvýši aj jeho energetická náročnosť. Ak sa však zameriame na komunikačnú náročnosť, tá sa pri použití 2048-bitového kľúča s pôvodným protokolom zvýši približne o 40%, zatiaľ čo pri jeho modifikácii o necelých 9%.

7.4 Overenie zefektívnenia vzhľadom na odstránený stav

V časti práce 5.1 uvádzame návrh optimalizácie odstránením ukončovacej správy a spolu s ňou aj odstránenie priebežného ukončovacieho stavu. V tomto stave protokol čaká na potvrdzovaciu správu *CloseAck* pre úplné ukončenie spojenia na oboch komunikujúcich zariadeniach. Pri nami navrhovanom riešení zariadenie ukončí spojenie hneď po odoslaní ukončovacej správy, zatiaľ čo pri pôvodnej implementácii protokolu zariadenie ukončí

spojenie až po prijatí potvrdzujúcej správy. Pri testovaní sme sa rozhodli odmerať čas medzi odoslanými správami, konkrétne čas medzi odoslaním ukončovacej správy a prijatím správy potvrdenia o ukončení.

Meranie sme opakovali 10-krát a priemerný čas od odoslania po prijatie danej správy je približne 84,35 milisekúnd. Toto teda predstavuje približne čas, ktorý je touto optimalizáciou ušetrený, a teda sa zariadenie môže o takýto čas skôr prepnúť do úsporného/spánkového režimu. Z nameraného výsledku si môžme myslieť, že takýto čas nemá veľký zmysel. Testovanie však prebiehalo na zariadeniach vzdialených približne 1 meter. Avšak pri umiestnení servera do cloudu môže tento čas výrazne narásť, vzhľadom na oneskorenia v IP sieťach alebo nastane situácia, kedy sa potvrdzovacia spáva stratí (nedoručí) a dané IoT zariadenie bude musieť čakať na vypršanie predvoleného časovača.

7.5 Overenie z hľadiska spotreby energie

Pre overenie energetickej náročnosti protokolu sme navrhli overenie z hľadiska spotreby energie. Jedným z možných spôsobov je pripojenie ampérmetra a voltmetra medzi zdroj napájania a zariadenie. Testovanie sme opakovali 20-krát na jednoduchom scenári nadviazania spojenia protokolu HIP a jeho ukončenie, kde sme pozorovali zmenu napätia a prúdu. Priemerné dosiahnuté výsledky uvádzame v nasledujúcej tabuľke 7.5.

Tabuľka 7.5: Energetická náročnosť

	Bez záťaže	**OpenHip - nadviazanie**	**E-Hip - nadviazanie**	**OpenHip - ukončenie**	**E-Hip - ukončenie**
Napätie (V)	5,20	5,20	5,20	5,20	5,20
Prúd (A)	0,31	0,335	0,33	0,32	0,31
Výkon (W)	1,612	1,742	1,716	1,664	1,612

Z priemerných nameraných hodnôt napätia a prúdu sme vypočítali výkon:

$$P(W) = U(V) * I(A)$$

kde:

P je výkon (Watt),

U je napätie (Volt),

I je prúd (Ampér).

Taktiež sme vypočítali výkon pri nulovej (statickej) záťaži zariadenia, čo predstavuje 1,612 W. Ten sme následne odčítali od výkonov nameraných so spusteným protokolom OpenHip a porovnali záťaž/energetickú náročnosť protokolu pri nadväzovaní spojenia (viď tabuľka 7.6). Priemerný potrebný výkon pre scenár ukončenia spojenia bol rovnaký ako výkon nameraný bez záťaže (spôsobené presnosťou multimetra).

Tabuľka 7.6: Vyhodnotenie záťaže pri odčítaní statickej záťaže zariadenia

	Nadviazanie spojenia (bez záťaže OS)
Výkon OpenHip (mW)	130
Výkon E-Hip (mW)	104
Zefektívnenie (mW)	26
Zefektívnenie (%)	20

7.6 Overenie bezpečnostných vlastností

Jednou zo špecifikácii zadania tejto práce je zachovanie bezpečnostných vlastností (integrita, dôvernosť, autentifikácia) protokolu HIP (tabuľka 2.2). Na ich zabezpečenie, protokol HIP Base Exchange (BEX) podporuje rôzne bezpečnostné mechanizmy ako výpočet puzzle (ochrana pred DoS), Diffie-Hellman výmenu kľúčov, digitálny podpis správ (integrita, autentifikácia) [63].

Overenie sme realizovali pre kontrolné správy počas nadviazania spojenia, a to porovnaním výpisov pre jednotlivé operácie modifikovaného protokolu HIP s jeho pôvodnou verziou.

Dôležité výpisy pri jednotlivých správach na zariadeniach:

- R1: porovnanie vypočítanej hodnoty SHA1,
- R1: *RSA HIP signature is good.*,
- I2: porovnanie výsledku puzzle - *solution: 0x...*,
- I2: *HMAC verified OK.*,

- I2: *RSA HIP signature is good.*,
- R2: *HMAC_2 verified OK.*,
- R2: *RSA HIP signature is good.*,
- *HIP exchange complete.*,
- porovnanie vytvorných SPI.

Kontrolou týchto výpisov počas testovania protokolu máme za to, že jeho optimalizáciou nedošlo k modifikácii bezpečnostných funkcionalít, t.j. bezpečnostné vlastnosti sú zachované.

7.7 Porovnanie s existujúcimi riešeniami

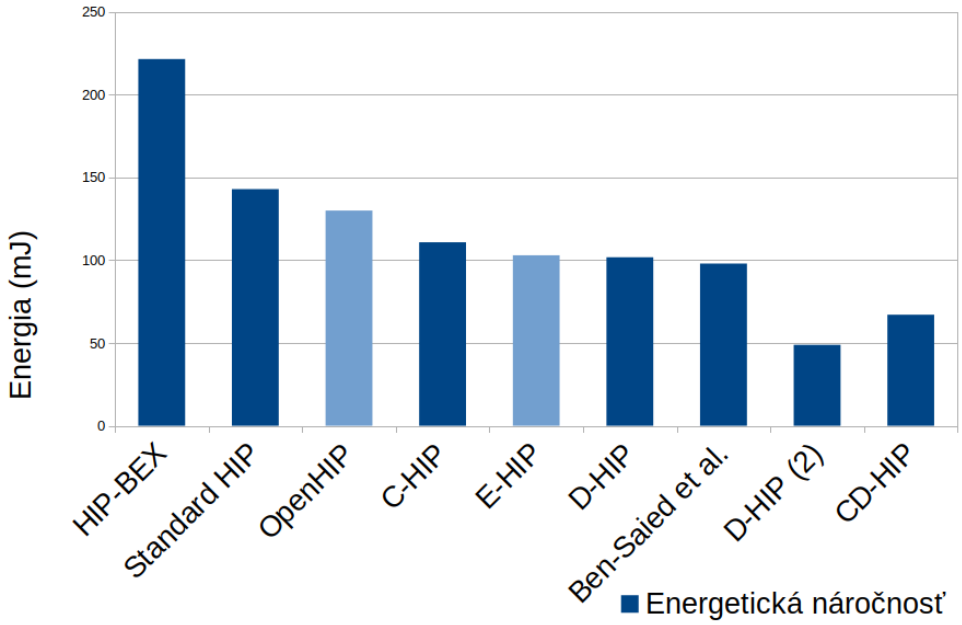
Existuje niekoľko optimalizácii protokolu HIP pre zefektívnenie komunikačnej alebo výpočtovej zložitosti, za účelom zníženia celkovej energetickej náročnosti. HIP-BEX, spolu so Standard HIP, ktorý je obohatený o autentifikáciu správ, predstavujú základnú implementáciu protokolu HIP. Protokoly s názvom Ben-Saied et al. a D-HIP sú reprezentantmi optimalizácie, kde riešenie spočíva v rozložení výpočtovej náročnosti. C-HIP je zameraný na kompresiu hlavičky protokolu HIP. CD-HIP kombinuje D-HIP a C-HIP riešenia.

V tabuľke 7.7, ktorú sme uviedli aj v analýze tejto práce (kapitola 3.4.2), uvádzame energetickú náročnosť jednotlivých modifikácii protokolu HIP z dostupnej uvedenej literatúry. Všetky hodnoty sú uvádzané v micro jouloch.

Tabuľka 7.7: Energetická náročnosť protokolu HIP (hodnoty uvedené v mJ) [62, 63]

Protokol	**Náročnosť na komunikáciu**	**Výpočtová náročnosť**	**Celková náročnosť**
Standard HIP	78,67	64,231	142,901
HIP-BEX	30,71	190,73	221,44
Ben-Saied et al.	94,55	3,43	97,98
C-HIP	47,186	63,622	110,808
D-HIP	86,59	15,17	101,76
D-HIP (2)	48,14	0,76	48,9
CD-HIP	53,16	14,93	67,09
OpenHIP	n/a	n/a	130
E-HIP	n/a	n/a	104

V tabuľke taktiež uvádzame nami nameranú hodnotu pre pôvodný protokol OpenHIP a nami optimalizovanú jeho verziu E-HIP. Pre lepšiu viditeľnosť a porovnanie sme sa rozhodli zobraziť výsledky na grafe (Obr. 7.4).



Obr. 7.4: Porovnanie energetickej náročnosti jednotlivých riešení

Z grafu je možné vidieť, že po implementovaní navrhnutých zmien do protokolu OpenHIP, sme dosiahli približne rovnakú energetickú náročnosť protokolu E-HIP ako iné optimalizované riešenia.

7.8 Záver testovania

Jednotlivými testovaniami v tejto kapitole sme ukázali funkčnosť nami navrhnutého protokolu E-HIP a splnenie jednotlivých požiadaviek uvedených v kapitole 4. Protokol funguje správne, bez chybových stavov a bol otestovaný s protokolom IPv4. Taktiež podporuje fungovanie s rôznymi veľkosťami RSA kľúčov. Pri použití 2048-bitového kľúča s E-HIP, sa počet prenesených bajtov (komunikačná náročnosť) zvýši o 9%, v porovnaní s použitím 1024-bitového kľúča v protokole OpenHIP. Jeho bezpečnosť sa však zvýši. Ak porovnáme zefektívnenie pôvodného a upraveného protokolu vyhodnotené na základe počtu prenesených bajtov (19,7%) a energetickej náročnosti odmeranej multimetrom (20%),

dosahujeme približne rovnaké percento zefektívnenia. Energetická náročnosť navrhnutého riešenia je porovnateľná s inými existujúcimi riešeniami, pričom boli použité unikátne optimalizačné riešenia. Tieto by sa ďalej dali skombinovať s niektorými existujúcimi (napr. C-HIP, D-HIP) a dosiahnuť ešte vyššiu energetickú efektívnosť.

8. Záver

S postupnou automatizáciou každodenných procesov (Priemyselná revolúcia 4.0) a pripájaním čoraz viac zariadení k Internetu vzniká tzv. Internet vecí, s čím súvisí aj nárast prenášaných dát v sieti. Tieto zariadenia sú častokrát napájané pomocou batérie, a preto je potrebné optimalizovať procesy, vzhľadom na ich energetickú náročnosť.
Jednou z oblastí sú aj bezpečnostné protokoly používané v doméne IoT. V tejto diplomovej práci sme analyzovali niekoľko existujúcich protokolov a ich optimalizácii (LKA, DTLS, a pod.). Na základe nej sme sa rozhodli pre zefektívnenie protokolu OpenHip, s cieľom znížiť jeho energetickú náročnosť pri zachovaní bezpečnostných vlastností alebo práve naopak, zvýšiť bezpečnosť pri rovnakej náročnosti.
V práci sme navrhli a opísali niekoľko čiastkových riešení, z ktorých sme tri (Odstránenie správy CloseAck a priebežného ukončovacieho stavu, Redukcia formátu parametrov protokolu OpenHIP, Odstránenie parametru HI-R - Manuálne riešenie) implementovali do existujúcej, voľne dostupnej knižnice protokolu OpenHip. Implementáciu celkového riešenia sme realizovali na zariadeniach podporujúcich operačný systém Unix, a to Raspberry Pi ako reprezentant IoT zariadení a prenosný počítač reprezentujúci server. Tie spolu komunikovali pomocou technológie Bluetooth. V takto vytvorenom testovacom prostredí sme navrhli niekoľko scenárov pre overenie zefektívnenia nami navrhnutého a implementovaného riešenia s dôrazom na splnenie špecifikácie požiadaviek.
Ukázali sme, že nami vytvorený protokol E-HIP dosahuje podobné výsledky ako existujúce iné riešenia optimalizácie tohto protokolu. Pri porovnaní pôvodného a optimalizovaného riešenia sme dosiahli 20 percentné zefektívnenie vzhľadom na počet prenesených bajtov v kontrolných správach protokolu, čo sa potvrdilo aj pri odmeraní energetickej náročnosti. Pri zvýšení bezpečnosti použitím väčšieho RSA kľúča sme ukázali, že náročnosť protokolu OpenHip by vzrástla o 40% narozdiel od 9% pri použití nami navrhovaného E-HIP. Taktiež sme otestovali splnenie bezpečnostných vlastností protokolu.
Výsledkom našej práce je modifikácia štandardného protokolu HIP pre potreby nízko-

energetických zariadení Internetu vecí. Zníženie energetickej náročnosti zvýši životnosť týchto zariadení napájaných pomocou batérií. Väčší verejný kľúč zvýši jeho bezpečnosť a zníženie počtu prenesených bajtov v správach zase uvoľní šírku pásma pre inú komunikáciu, čo minimalizuje rušenie, čakanie na prenos, prípadne výskyt kolízií. To všetko taktiež ovplyvňuje energetickú náročnosť. Nevýhodou navrhnutého riešenia je, že modifikáciou sme narušili kompatibilitu so štandardným protokolom, čo nie je veľký problém, pretože ako z analýzy vyplynulo, aplikačne špecifická komunikácia si vyžaduje aplikačne špecifické komunikačné protokoly pre zvýšenie efektívnosti.

V budúcnosti je možné rozšíriť túto prácu, a to prepojením s existujúcimi optimalizáciami protokolu HIP, implementáciou na IoT zariadenia bez potreby operačného systému a použitím komunikačnej technológie Bluetooth 5 alebo prichádzajúcou mobilnou sieťou 5G.

Literatúra

[1] Ver. openhip-0.9. URL: https://github.com/rektide/openhip.

[2] B. Aboba, D. Simon a P. Eronen. *Extensible Authentication Protocol (EAP) Key Management Framework*. RFC 5247. http://www.rfc-editor.org/rfc/rfc5247.txt. RFC Editor, 2008. URL: http://www.rfc-editor.org/rfc/rfc5247.txt.

[3] Shadi Al-Sarawi et al. "Internet of Things (IoT) communication protocols". In: *Information Technology (ICIT), 2017 8th International Conference on*. IEEE. 2017, s. 685–690.

[4] Laxmi Ashrit. *Bluetooth 5 Technology – Protocol Stack, Network Topology, Applications*. https://electricalfundablog.com/bluetooth-5-technology/. Accessed: 2019-03-03.

[5] Jay Young Atri Basu. *IKEv2 Packet Exchange and Protocol Level Debugging*. Accessed: 2018-03-13. 2013. URL: https://www.cisco.com/c/en/us/support/docs/security-vpn/ipsec-negotiation-ike-protocols/115936-understanding-ikev2-packet-exch-debug.html.

[6] Luigi Atzori, Antonio Iera a Giacomo Morabito. "The internet of things: A survey". In: *Computer networks* 54.15 (2010), s. 2787–2805.

[7] Tuomas Aura, Aarthi Nagarajan a Andrei Gurtov. "Analysis of the HIP base exchange protocol". In: *Australasian Conference on Information Security and Privacy*. Springer. 2005, s. 481–493.

[8] Sachin Babar et al. "Proposed security model and threat taxonomy for the Internet of Things (IoT)". In: *International Conference on Network Security and Applications*. Springer. 2010, s. 420–429.

[9] Utsav Banerjee et al. "eeDTLS: Energy-Efficient Datagram Transport Layer Security for the Internet of Things". In: *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE. 2017, s. 1–6.

[10] Elaine Barker. "NIST Special Publication 800-57 Part 1 Revision 4, Recommendation for Key Management Part 1: General". In: *NIST* (2016).

[11] Janina Bartje. "The top 10 IoT application areas-based on real IoT projects". In: *IoT analytics* (2016).

[12] Riccardo Bonetto et al. "Secure communication for smart IoT objects: Protocol stacks, use cases and practical examples". In: *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*. IEEE. 2012, s. 1–7.

[13] Angelo Capossele et al. "Security as a CoAP resource: an optimized DTLS implementation for the IoT". In: *Communications (ICC), 2015 IEEE International Conference on*. IEEE. 2015, s. 549–554.

[14] Simone Cirani et al. "Iot-oas: An oauth-based authorization service architecture for secure services in iot scenarios". In: *IEEE sensors journal* 15.2 (2015), s. 1224–1234.

[15] Alberto Compagno, Mauro Conti a Ralph Droms. "OnboardICNg: a secure protocol for on-boarding IoT devices in ICN". In: *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. ACM. 2016, s. 166–175.

[16] Cas Cremers. "Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2". In: *European Symposium on Research in Computer Security*. Springer. 2011, s. 315–334.

[17] T. Dierks a E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. `http://www.rfc-editor.org/rfc/rfc5246.txt`. RFC Editor, 2008. URL: `http://www.rfc-editor.org/rfc/rfc5246.txt`.

[18] Bruno Dorsemaine et al. "Internet of things: a definition & taxonomy". In: *Next Generation Mobile Applications, Services and Technologies, 2015 9th International Conference on*. IEEE. 2015, s. 72–77.

[19] Benedikt Driessen et al. "IPSecco: A lightweight and reconfigurable IPSec core". In: *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*. IEEE. 2012, s. 1–7.

[20] Otmane El Mouaatamid, Mohammed Lahmer a Mostafa Belkasmi. "Internet of Things Security: Layered classification of attacks and possible Countermeasures". In: *Electronic Journal of Information Technology* 9 (2016).

[21] Jens Eliasson, Chen Zhong a Jerker Delsing. "A heterogeneous sensor network architecture for highly mobile users". In: *2010 Sixth International conference on Wireless Communication and Sensor Networks*. IEEE. 2010, s. 1–6.

[22] Emarketer. *Internet of Things Is Changing How Media and Entertainment Companies Operate*. https://www.emarketer.com/Article/Internet-of-Things-Changing-How-Media-Entertainment-Companies-Operate/1013545. Accessed: 2018-02-20.

[23] Jaroslav Erdelyi. "Communication in the Internet of Thing Environment". In: *IIT.SRC* (2018).

[24] Pasi Eronen et al. "Internet key exchange protocol version 2 (IKEv2)". In: (2010).

[25] Aleksandr Fedorov et al. "Aspects of smart manufacturing via agent-based approach". In: *Procedia Engineering* 100 (2015), s. 1572–1581.

[26] Dan Forsberg et al. *Protocol for carrying authentication for network access (PANA)*. Tech. spr. 2008.

[27] S. Frankel a S. Krishnan. *IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap*. RFC 6071. http://www.rfc-editor.org/rfc/rfc6071.txt. RFC Editor, 2011. URL: http://www.rfc-editor.org/rfc/rfc6071.txt.

[28] Steven Friedl. "An illustrated guide to cryptographic hashes". In: *Www-dokumentti. Saatavissa: http://www. unixwiz. net/techtips/iguide-crypto-hashes. html. Luettu* 6 (2005), s. 2013.

[29] Hannes Gross et al. "Privacy-aware authentication in the internet of things". In: *International Conference on Cryptology and Network Security*. Springer. 2015, s. 32–39.

[30] Yang Hao a Robert Foster. "Wireless body sensor networks for health-monitoring applications". In: *Physiological measurement* 29.11 (2008), R27.

[31] Jose L Hernandez-Ramos et al. "Toward a lightweight authentication and authorization framework for smart objects". In: *IEEE Journal on Selected Areas in Communications* 33.4 (2015), s. 690–702.

[32] *Internet of Things (IoT)*. https://www.techopedia.com/definition/28247/internet-of-things-iot. Accessed: 2018-02-19.

[33] *IPsec tunnel and transport modes – Why doesn't transport mode work between routers?* https://diecarvi.wordpress.com/2013/07/04/ipsec-tunnel-and-transport-modes-why-doesnt-transport-mode-work-between-routers/. Accessed: 2018-03-12.

[34] Vilhelm Jutvik. *IPsec and IKEv2 for the Contiki Operating System*. 2014.

[35] Mitsuru Kanda et al. "PANA applicability in constrained environments". In: *Smart Object Security Wksp.* 2012.

[36] Sye Loong Keoh, Sandeep S Kumar a Hannes Tschofenig. "Securing the internet of things: A standardization perspective". In: *IEEE Internet of Things Journal* 1.3 (2014), s. 265–275.

[37] Thorsten Kleinjung et al. *Factorization of a 768-bit RSA modulus*. Cryptology ePrint Archive, Report 2010/006. `https://eprint.iacr.org/2010/006`. 2010.

[38] Thomas Kothmayr et al. "DTLS based security and two-way authentication for the Internet of Things". In: *Ad Hoc Networks* 11.8 (2013), s. 2710–2723.

[39] Thomas Kothmayr et al. "Poster: Securing the internet of things with DTLS". In: *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM. 2011, s. 345–346.

[40] KRISTOPHER SANDOVAL. *OAuth 2.0 – Why It's Vital to IoT Security*. `https://nordicapis.com/why-oauth-2-0-is-vital-to-iot-security/`. Accessed: 2018-02-19.

[41] M Lavanya a V Natarajan. "Lightweight key agreement protocol for IoT based on IKEv2". In: *Computers & Electrical Engineering* 64 (2017), s. 580–594.

[42] Gyu Myoung Lee et al. "The Internet of Things–Concept and Problem Statement. July 2011". In: *Internet Research Task Force* (2011).

[43] Dhiraj M Londe a Nitin Chopde. "A New Lightweight EAP-PK Authentication Method for IEEE 802.11 standard Wireless Network". In: *International Journal of Computer Science and Information Technolo*. Citeseer, 2014.

[44] Jing Lu a John Lockwood. "IPSec implementation on xilinx virtex-II Pro FPGA and its application". In: *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*. IEEE. 2005, 7–pp.

[45] James Macaulay, Lauren Buckalew a Gina Chung. "Internet of things in logistics". In: *A collaborative report by DHL and Cisco on implications and use cases for the logistics industry, Pub. DHL Customer Solutions & Innovation, Troisdorf* (2015).

[46] Yassine Maleh, Abdellah Ezzati a Mustapha Belaissaoui. "An enhanced DTLS protocol for Internet of Things applications". In: *Wireless Networks and Mobile Communications (WINCOM), 2016 International Conference on*. IEEE. 2016, s. 168–173.

[47] Diego M Mendez, Ioannis Papapanagiotou a Baijian Yang. "Internet of things: Survey on security and privacy". In: *arXiv preprint arXiv:1707.01879* (2017).

[48] Jacob Morgan. "A Simple Explanation Of'The Internet Of Things"'. In: *Retrieved November* 20 (2014), s. 2015.

[49] R. Moskowitz a P. Nikander. *Host Identity Protocol (HIP) Architecture*. RFC 4423. RFC Editor, 2006.

[50] R. Moskowitz et al. *Host Identity Protocol*. RFC 5201. RFC Editor, 2008.

[51] R. Moskowitz et al. *Host Identity Protocol Version 2 (HIPv2)*. RFC 7401. `http://www.rfc-editor.org/rfc/rfc7401.txt`. RFC Editor, 2015. URL: `http://www.rfc-editor.org/rfc/rfc7401.txt`.

[52] Kim Thuat Nguyen, Maryline Laurent a Nouha Oualha. "Survey on secure communication protocols for the Internet of Things". In: *Ad Hoc Networks* 32 (2015), s. 17–31.

[53] Pekka Nikander, Andrei Gurtov a Thomas R Henderson. "Host identity protocol (HIP): Connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks". In: *IEEE Communications Surveys & Tutorials* 12.2 (2010), s. 186–204.

[54] Yun Niu et al. "A configurable ipsec processor for high performance in-line security network processor". In: *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*. IEEE. 2011, s. 674–678.

[55] *OpenHip documentation*. `http://openhip.sourceforge.net/wiki/`. Accessed: 2018-12-05.

[56] Marcin Piotr Pawlowski, Antonio J Jara a Maciej J Ogorzalek. "EAP for IoT: More Efficient Transport of Authentication Data–TEPANOM Case Study". In: *Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on*. IEEE. 2015, s. 694–699.

[57] Nachiketh R Potlapally et al. "Analyzing the energy consumption of security protocols". In: *Proceedings of the 2003 international symposium on Low power electronics and design*. ACM. 2003, s. 30–35.

[58] Shahid Raza et al. "Lithe: Lightweight secure CoAP for the internet of things". In: *IEEE Sensors Journal* 13.10 (2013), s. 3711–3720.

[59] Shahid Raza et al. "Secure communication for the Internet of Things—a comparison of link-layer security and IPsec for 6LoWPAN". In: *Security and Communication Networks* 7.12 (2014), s. 2654–2668.

[60] Shahid Raza et al. "Securing communication in 6LoWPAN with compressed IPsec". In: *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*. IEEE. 2011, s. 1–8.

[61] E. Rescorla a N. Modadugu. *Datagram Transport Layer Security Version 1.2*. RFC 6347. `http://www.rfc-editor.org/rfc/rfc6347.txt`. RFC Editor, 2012. URL: `http://www.rfc-editor.org/rfc/rfc6347.txt`.

[62] Somia Sahraoui a Azeddine Bilami. "Efficient HIP-based Approach to Ensure Lightweight End-to-end Security in the Internet of Things". In: *Comput. Netw.* 91.C (nov. 2015), s. 26–45. ISSN: 1389-1286. DOI: `10.1016/j.comnet.2015.08.002`. URL: `http://dx.doi.org/10.1016/j.comnet.2015.08.002`.

[63] Yosra Ben Saied a Alexis Olivereau. "HIP Tiny Exchange (TEX): A distributed key exchange scheme for HIP-based Internet of Things". In: *Third International Conference on Communications and Networking*. IEEE. 2012, s. 1–8.

[64] Ahmad Salman, Marcin Rogawski a Jens-Peter Kaps. "Efficient hardware accelerator for IPSec based on partial reconfiguration on Xilinx FPGAs". In: *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*. IEEE. 2011, s. 242–248.

[65] Pedro Moreno Sanchez, Rafa Marin Lopez a Antonio F Gomez Skarmeta. "Panatiki: A network access control implementation based on PANA for IoT devices". In: *Sensors* 13.11 (2013), s. 14888–14917.

[66] Pallavi Sethi a Smruti R Sarangi. "Internet of things: architectures, protocols, and applications". In: *Journal of Electrical and Computer Engineering* 2017 (2017).

[67] Joshua R Stachel et al. "The impact of the internet of Things on implanted medical devices including pacemakers, and ICDs". In: *Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE International*. IEEE. 2013, s. 839–844.

[68] Harald Sundmaeker et al. "Vision and challenges for realising the Internet of Things". In: *Cluster of European Research Projects on the Internet of Things, European Commision* 3.3 (2010), s. 34–36.

[69] Klaus-Dieter Thoben, Stefan Wiesner a Thorsten Wuest. "Industrie 4.0" and smart manufa-cturing–a review of research issues and application examples". In: *Int. J. Autom. Technol* 11.1 (2017).

[70] Pascal Urien. "Innovative TLS/DTLS Security Modules for IoT Applications: Concepts and Experiments". In: *International Internet of Things Summit*. Springer. 2015, s. 3–15.

[71] Mališa Vučinić et al. "DTLS performance in duty-cycled networks". In: *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015 IEEE 26th Annual International Symposium on*. IEEE. 2015, s. 1333–1338.

[72] Haixin Wang, Guoqiang Bai a Hongyi Chen. "A gbps ipsec ssl security processor design and implementation in an fpga prototyping platform". In: *Journal of Signal Processing Systems* 58.3 (2010), s. 311–324.

[73] Muhyi Bin Yaakop et al. "Bluetooth 5.0 throughput comparison for internet of thing usability a survey". In: *2017 International Conference on Engineering Technology and Technopreneurship (ICE2T)*. IEEE. 2017, s. 1–6.

[74] Andrea Zanella et al. "Internet of things for smart cities". In: *IEEE Internet of Things journal* 1.1 (2014), s. 22–32.

A. Technická dokumentácia

A.1 Implementácia

Táto časť technickej dokumentácie obsahuje jednotlivé vykonané zmeny potrebné pre implementáciu navrhnutého riešenia.

A.1.1 Odstránenie CloseAck a priebežného ukončovacieho stavu

V nasledujúcej ukážke kódu je upravená funkcia pre spracovanie prijatej ukončovacej správy.

Ukážka kódu A.1: src/protocol/hip_input.c

```
int hip_handle_close(__u8 *buff, hip_assoc *hip_a){

//Comment if to next else if; Comment !is_ack
if (is_ack && (hip_a->state != CLOSING) &&
    (hip_a->state != CLOSED))
...
//  else
if (/*!is_ack &&*/(hip_a->state != ESTABLISHED) &&
    (hip_a->state != CLOSING) && (hip_a->state != CLOSED) &&
    (hip_a->state != R2_SENT))

************************************************************
//Comment whole if statement
//if (!is_ack)          /* respond with CLOSE_ACK */
//...
```

```
//else
//...
//}
*************************************************************
//Change following lines to
log_hipa_fromto(QOUT, "Close completed",
        hip_a, TRUE, TRUE);
set_state(hip_a, UNASSOCIATED);
delete_associations(hip_a, 0, 0);
free_hip_assoc(hip_a);
return(0);
```

Ukážka kódu A.2: src/protocol/hip_main.c

```
void hip_handle_state_timeouts(struct timeval *time1){
...
set_state(hip_a, CLOSED);  //Change Closing to Closed
...
```

A.1.2 Redukcia parametrov

Je potrebné pridať nasledujúce upravené štruktúry.

Ukážka kódu A.3: include/hip_types.c

```
typedef struct _tlv_esp_info_s
{
  __u16 reserved;
  __u16 keymat_index;
  __u32 old_spi;
  __u32 new_spi;
} tlv_esp_info_s;

typedef struct _tlv_r1_counter_s
{
  __u32 reserved;
```

```
  __u64 r1_gen_counter;
} tlv_r1_counter_s;

typedef struct _tlv_solution_s
{
  hipcookie cookie;
  __u64 j;
} tlv_solution_s;

typedef struct _tlv_diffie_hellman_s
{
  __u16 length;
  __u8 group_id;
  __u16 pub_len;
  __u8 pub[1];        /* variable length */
} __attribute__ ((packed)) tlv_diffie_hellman_s;

typedef struct _tlv_hip_transform_s
{
  __u16 length;
  __u16 transform_id;
} tlv_hip_transform_s;

typedef struct _tlv_hmac_s
{
  __u8 hmac[20];
} tlv_hmac_s;

typedef struct _tlv_hip_sig_s
{
  __u8 algorithm;
  __u8 signature[0];        /* variable length */
} tlv_hip_sig_s;
```

Následne je potrebné upraviť funkciu hip_parse_I2, a to nahradiť pôvodné štruktúry za upravené a zadefinovať pevné poradie prijatia parametrov premiestnením z cyklu *while* pred jeho vykonanie. Za každým parametrom je potrebné pridať riadok s upravením lokácie (*location*). Nakoniec pridať upravenú funkciu *handle_dh_s*, kde je potrebné nahradiť pôvodné štruktúry za nové (viď nasledujúce ukážky).

Ukážka kódu A.4: src/protocol/hip_input.c

```
int hip_parse_I2(const __u8 *data, hip_assoc **hip_ar,
                 hi_node *my_host_id, struct sockaddr *src,
                 struct sockaddr *dst){

//1. parameter PARAM_ESP_INFO
tlv = (tlv_head*) &data[location];
esp_info = (tlv_esp_info_s*)tlv;
proposed_keymat_index = ntohs(esp_info->keymat_index);
proposed_spi_out = ntohl(esp_info->new_spi);
location += sizeof(tlv_esp_info_s)-4;

//2. parameter PARAM_R1_COUNTER
tlv = (tlv_head*) &data[location];
r1count = ntoh64(((tlv_r1_counter_s*)tlv)->r1_gen_counter);
...
location += sizeof(tlv_r1_counter_s);

//3. parameter PARAM_SOLUTION
tlv = (tlv_head*) &data[location];
memcpy(&cookie, &((tlv_solution_s*)tlv)->cookie,
        sizeof(hipcookie));
...
location += sizeof(tlv_solution_s)-4;

//4. parameter PARAM_DIFFIE_HELLMAN
length = ntohs(((tlv_diffie_hellman_s*)&data[location])->length);
    if (handle_dh_s(hip_a, &data[location], &g_id,
```

```
...
location = location + length + 2;

//5. parameter PARAM_HIP_TRANSFORM
length = ntohs(((tlv_hip_transform_s*)&data[location])->length);
p = &((tlv_hip_transform_s*)
    ((tlv_hip_transform_s*)&data[location]))->transform_id;
...
location = location + length + 5;
```

Ukážka kódu A.5: src/protocol/hip_input.c

```
int handle_dh_s(hip_assoc *hip_a,
    const __u8 *data, __u8 *g, DH *dh){
...
}
```

Vo funkcii *hip_send_I2* je potrebné taktiež zmeniť štruktúry a ich napĺňanie parametrov. Pridanie funkcií *build_tlv_dh_s build_tlv_transform_s* s upravenými štruktúrami.

Ukážka kódu A.6: src/protocol/hip_input.c

```
int hip_send_I2(hip_assoc *hip_a){
...
//esp_info->type = htons(PARAM_ESP_INFO);
//esp_info->length = htons(sizeof(tlv_esp_info) - 4);
...
location += sizeof(tlv_esp_info_s)-4;
//location = eight_byte_align(location);
...
r1cnt = (tlv_r1_counter_s*) &buff[location];
// r1cnt->type = htons(PARAM_R1_COUNTER);
// r1cnt->length = htons(sizeof(tlv_r1_counter) - 4);
...
location += sizeof(tlv_r1_counter_s);
// location = eight_byte_align(location);
```

```
...
sol = (tlv_solution_s*) &buff[location];
// sol->type = htons(PARAM_SOLUTION);
// sol->length = htons(sizeof(tlv_solution) - 4);
...
((tlv_solution_s*) &buff[location-4])->j = solution;
// sol->j = solution;
...
location += sizeof(tlv_solution_s)-4;
// location = eight_byte_align(location);
...
//need to replace spi here
esp_info->new_spi = htonl(hip_a->spi_in);
...
location += build_tlv_dh_s(&buff[location], hip_a->dh_group_id,
                           hip_a->dh, OPT.debug);
location += build_tlv_transform_s(&buff[location],
            PARAM_HIP_TRANSFORM, zero16, hip_a->hip_transform);
```

A.1.3 Odstránenie parametra HI-R

Odstránenie pridania parametra do správy.

Ukážka kódu A.7: src/protocol/hip_output.c

```
int hip_generate_R1(__u8 *data, hi_node *hi, hipcookie *cookie,
                    dh_cache_entry *dh_entry){
...
//location += build_tlv_hostid(&data[location], hi,
//                  HCNF.send_hi_name);
...
```

Je potrebné pridať načítavanie parametrov N, E, dĺžky kľúča a mena zo súboru a odstrániť časť načítavania z paketu (Na ukážke je zobrazená len časť načítavania zo súboru).

Ukážka kódu A.8: src/protocol/hip_input.c

```
int hip_parse_R1(const __u8 *data, hip_assoc *hip_a){
...
doc = xmlParseFile(HCNF.known_hi_filename);
fprintf(stderr, "Parsing xml file (%s)\n",
                        HCNF.known_hi_filename);
if (doc == NULL)
{
  fprintf(stderr, "Error parsing xml file (%s)\n",
            HCNF.known_hi_filename);
  return(-1);
}
node = xmlDocGetRootElement(doc);
for (node = node->children; node; node = node->next)
{
    if (strcmp((char *)node->name, "host_identity") == 0)
    {
        xmlAttrPtr attr = node->properties;
...
//else if (!sig_verified && (type == PARAM_HOST_ID)){
...
```

Pre jednoduchšie generovanie a prenos verejného kľúča na zariadenie je vhodné pridať zápis N a E do nasledujúceho výrazu.

Ukážka kódu A.9: src/util/hitgen.c

```
void publish_hits(char *out_filename){
...
if ((strcmp((char *)child->name, "name") != 0) &&
  (strcmp((char *)child->name, "HIT") != 0) &&
  (strcmp((char *)child->name, "LSI") != 0) &&
  (strcmp((char *)child->name, "addr") != 0) &&
  (strcmp((char *)child->name, "N") != 0) &&
  (strcmp((char *)child->name, "E") != 0))
...
```

A.2 Testovacie prostredie

Na otestovanie celého riešenia sme použili nasledovné hardvérové a softvérové prostriedky.

Hardvér:

- osobný prenosný počítač Lenovo M5400,
- Raspberry Pi 3B.

Softvér:

- OpenHip v0.9, do ktorej sme implementovali naše riešenie,
- OS Ubuntu 18.04,
- OS Raspbian,
- bt-pan pre vytvorenie Bluetooth siete (*z angl. PAN - Personal Area Network*) [1].

A.3 Používateľská príručka

V tejto kapitole opisujeme potrebné príkazy pre vytvorenie spojenia medzi dvoma komunikujúcimi uzlami, kompiláciu protokolu zo zdrojových súborov, nainštalovanie knižnice, či jej samotné použitie.

A.3.1 Vytvorenie Bluetooth siete

Spustenie Bluetooth a spárovanie zariadení:

Ukážka kódu A.10: Server side

```
% bluetoothctl
[bluetooth]# power on
[bluetooth]# discoverable on
[bluetooth]# agent on
```

[1] Source: `https://github.com/mk-fg/fgtk#bt-pan`

Ukážka kódu A.11: Client side

```
% bluetoothctl
[bluetooth]# power on
[bluetooth]# scan on
[bluetooth]# agent on
[bluetooth]# pair <Device HW Address>
[bluetooth]# trust <Device HW Address>
```

Vytvorenie rozhrania pomocou nasledujúceho skriptu na strane servera:

Ukážka kódu A.12: make_interface

```
#!/bin/bash

br=bnep

[[ -n "$(brctl show $br 2>&1 1>/dev/null)" ]] && {
  brctl addbr $br
  brctl setfd $br 0
  brctl stp $br off
  ip addr add 10.1.2.3/24 dev $br
  ip link set $br up
}
```

V prípade potreby doinštalovať *bridge-utils* príkazom #apt−get install bridge−utils

Stiahnúť a spustiť nasledujúci skript (`https://raw.githubusercontent.com/mk-fg/fgtk/master/bt-pan`).

Ukážka kódu A.13: Server side

```
% ./bt-pan --debug server bnep
```

Ukážka kódu A.14: Client side

```
% ./bt-pan client <Device HW Address>
```

Pridanie IP adresy na vytvorené Bluetooth rozhranie:

Ukážka kódu A.15: Client side

```
% ifconfig bnep0 <IPAddress> netmask <Mask>
```

Ukončenie spojenia

Ukážka kódu A.16: Client side

```
% ./bt-pan client -d <Device HW Address>
```

A.3.2 Inštalácia

Inštalácia potrebných knižníc:

Ukážka kódu A.17: Dependency

```
% sudo apt-get install autotools-dev
% sudo apt-get install automake
% sudo apt-get install libxml2-dev
% sudo apt-get install libssl-dev
% sudo apt-get install libssl1.0.0
```

Kompilácia a inštalácia protokolu z priečinka *src/* knižnice protokolu:

Ukážka kódu A.18: Installation

```
% sudo ./bootstrap.sh && sudo ./configure
% sudo make && sudo make install
```

A.3.3 Spustenie

Vygenerovanie RSA kľúčov, kde:

- type - označuje typ protokolu (v našom prípade RSA),
- bits - veľkosť vygenerováneho kľúča (1024/2048 bit).

Ukážka kódu A.19: Key generate

```
% sudo hitgen -type RSA -bits <size>
% sudo hitgen -publish
```

Do vygenerovaného súboru *host_identities.pub.xml* je potrebné pridať nasledujúci záznam (<addr>X.X.X.X</addr>) s informáciou o IP adrese servera na vytvorenom Bluetooth rozhraní a prekopírovať jeho obsah na zariadenie do súboru *known_host_identities.xml*.

Spustenie protokolu, kde:

- -v pre debug výstup,
- -a pre povolenie pripojenia sa ľubovoľného zariadenia bez potreby poznania jeho identity na serveri.

Ukážka kódu A.20: Server side

```
% sudo hip -v -a
```

Ukážka kódu A.21: Client side

```
% sudo hip -v
```

B. Plán práce

Tabuľka B.1: Plán práce na DP I.

Týždeň	**Obsah**
1	Definícia IoT, využitie, oblasti pôsobnosti
2	Taxonómia/Rozdelenie bezpečnostných protokolov používaných v IoT
3 - 4	Analýza prvého vybratého protokolu / riešenia (komunikačného a šifrovacieho)*
5 - 6	Analýza druhého vybratého protokolu / riešenia *
7 - 8	Analýza tretieho vybratého protokolu / riešenia *
9	Zhodnotenie analyzovaných protokolov a špecifikácia zadania
9 - 12	Hrubý návrh riešenia a čiastočná implementácia

* Pri rýchlejšom splnení plánu v daných týždňoch, analýza ďalších riešení

V tabuľke B.1 uvádzame plán práce stanovený na tento semester, ktorý sa nám čiastočne podarilo plniť. V písaní sme začínali až v druhom týždni, kde sme dobehli aj zameškaný prvý bod. Stanovený plán sa nám podarilo plniť na 100% počas druhého až ôsmeho týždňa. Naplánované body v deviatom až dvanástom týždni sa dokončili koncom jedenásteho týždňa, kde sme analyzovali štvrtý protokol používaný v Internete vecí a navrhli jeho modifikáciu.

Tabuľka B.2: Plán práce DP II.

Týždeň	**Obsah**
1 - 2	Implementácia protokolu
3 - 4	Návrh a implementácia optimalizácie I.
5	Testovanie a vyhodnotenie návrhu I.
6 - 7	Návrh a implementácia optimalizácie II.
8	Testovanie a vyhodnotenie návrhu II.
9 - 10	Návrh a implementácia optimalizácie III.
11	Testovanie a vyhodnotenie návrhu III.
12	Celkové zhodnotenie návrhu a implementácie

Navrhnutý plán práce pre DP II. (viď tabuľka B.2) sa nám nepodarilo naplniť podľa stanovených týždňov. Počas semestra sme sa zaoberali siedmymi návrhmi, ktoré sme opísali v práci. V prototypovom riešení návrhu sme však implementovali štyri z nich. Vyhodnotenie návrhu z hľadiska jeho účinnosti a existujúcich riešení plánujeme v ďalšom semestri.

Tabuľka B.3: Plán práce DP III.

Týždeň	**Obsah**
1 - 6	Implementácia na IoT zariadeniach
7 - 8	Testovanie, vyhodnotenie a porovnanie s existujúcimi riešeniami
9 - 10	Príprava článku
11 - 12	Finalizácia dokumentu

Tabuľka B.3 zobrazuje plán práce pre posledný semester. Finálne dokončenie implementácie prebiehalo približne prvé 4 týždne. V týždňoch 5 - 8 sme sa venovali testovaniu a oprave drobných chýb. Posledné týždne sme venovali príprave článku do časopisu.

C. Obsah digitálnej časti práce

Evidenčné číslo práce v informačnom systéme: FIIT-136227-73957

Elektronické médium, priložené k diplomovej práci, obsahuje nasledovné súbory:

- `/openhip` — zdrojové súbory E-HIP,
- `/docs/DiplomovaPraca.pdf` — pdf dokument diplomovej práce,
- `/docs/TSP.pdf` — podaný článok,
- `/docs/latex` — zdrojové súbory dokumentu (LaTeX),
- `/docs/anotácie/Anotacia.pdf` — Slovenská anotácia,
- `/docs/anotácie/Annotation.pdf` — Anglická anotácia.

Názov odovzdaného archívu: DP_prilohy_digital_kanuch.zip

D. Článok

Článok podaný na konferenciu - 2019 42nd International Conference on Telecommunications and Signal Processing.

Optimizing Energy Efficiency of Secured IoT Communication by OpenHip

Peter Kaňuch, Dominik Macko
Faculty of Informatics and Information Technologies
Slovak University of Technology
Bratislava, Slovakia
Email: xkanuch@stuba.sk, dominik.macko@stuba.sk

***Abstract*—The research in the area of Internet-of-Things (IoT) security is still underway due to the growing IoT networks. The main goal of the existing works, optimizing security protocols, is to make them more efficient in order to reduce their energy requirements. The result is the extended lifetime of the device powered by a battery, while preserving all the security features of the protocol such as confidentiality, integrity, authenticity, etc. Based on the analysis, we have decided to focus on the protocol HIP (Host Identity Protocol), identified several optimization possibilities for efficient use in the IoT area, and proposed its modification. The analytical and experimental evaluation shows that the proposed modifications are beneficial regarding the energy efficiency of the HIP protocol.**

***Keywords*—energy efficiency; Internet of Things; low-power communication; security; wireless sensor networks**

I. Introduction

More and more interconnected devices are now being connected to the Internet, referred to as the Internet of Things (IoT) [1]. Since the number of interconnected IoT devices in the world grows rapidly (used in industry, smart cities, agriculture, etc. [2]), they gained the attention of network attackers. Therefore, IoT security is the most crucial and we must think about security features, such as authenticity, integrity, confidentiality, policy [3]–[5]. However, it is not easy for IoT devices to offer strong security features since many IoT devices are constrained on power and resources side. Most of the IoT sensor nodes are powered by batteries or harvesting energy from the environment. Therefore, optimization of traditionally used security protocols for the usage in IoT is a must. There already exist multiple research works in this area (e.g. [6]–[8]); however, there is still a space for optimization of communication energy efficiency.

In this paper, we focus on the HIP protocol [9], which was developed for a key-exchange procedure. We have identified several possible optimizations of its energy efficiency and optimized this protocol to efficiently secure the IoT communication while keeping the security level offered by the original protocol.

This work was partially supported by the Slovak Research and Development Agency(APVV-15-0789), the Slovak Cultural and Educational Grant Agency (KEGA 011STU-4/2017), the Slovak Scientific Grant Agency (VEGA 1/0836/16), the project "University Science Park of STU Bratislava" (ITMS 26240220084), co-funded by the European Regional Development Fund, and the project no. 2018/14427:1-26C0.

The organization of the paper follows this structure: In the next section, the works related to energy-efficient IoT security protocols are summarized and analyzed. In Section III, the proposed HIP modifications, optimizing its energy efficiency, are described in more detail. Section IV outlines validation of the selected modifications by their implementation into the OpenHIP library [10]. And finally, the conclusions of this work are provided in Section V.

II. Related Works

There are several existing solutions of energy-efficient communication security [11], optimized for usage in the IoT area.

Protocol DTLS (Datagram Transport Layer Security) [12] supports UDP communication, which is more energy efficient than TCP due to the transport-protocol header size. There was also proposed its modification eeDTLS (Energy-Efficient DTLS) [6], which reduced protocol headers and optimized the handshake process. Lithe (Lightweight Secure CoAP - Constrained Application Protocol - for the IoT) [7] is also one of the existing DTLS optimizations, which uses a combination of DTLS and CoAP to provide security. The proposed optimization consists especially of headers compression.

The authentication and data-encryption security features are not supported directly in the CoAP protocol, which is one of the most widely used IoT application protocols. Therefore, IPSec (Internet Protocol Security) is also used in the area of IoT [13], or its energy-efficient modification LKA (Lightweight Key Agreement) [8]. It is a minimized configuration of the IKEv2 (Internet Key Exchange) protocol that offers basic options only, such as the usage of a single cryptographic algorithm.

Protocol HIP [9] is designed for key exchange, as an alternative to IKE of the IPSec, which separates the identification (i.e. cryptographic identifiers) of devices from their locations (i.e. IP addresses). Such a security feature, enabling anonymous locations and supporting mobility, is very useful for many IoT applications. There are multiple works targeting the HIP protocol and making it more energy efficient. For example, the HIP-TEX modification [14] integrated a distribution mechanism into the HIP to cope with limited resources of IoT devices. Another modification, HIP-DEX [15], used the Elliptic-curve cryptography to lighten the computational requirements of the

protocol. This solution was further optimized by the Slimfit modification [16], which introduced a compression into the HIP header reducing the fragmentation rate. A combination of compression and distribution mechanisms was proposed in CD-HIP [17] to optimize the HIP for usage in constrained IoT devices using 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) communication.

III. The Proposed HIP Optimizations

The analyzed existing related works revealed that the optimization of the standard protocols for usage in constrained IoT devices and networks is quite common. We have selected the HIP protocol for our interests due to its unique benefits of hiding locations and mobility support, which are often required in healthcare or military industry. In our work, we propose six small modifications of the HIP protocol, which are further described in the following subsections.

A. Removal of the CloseAck Message and the Temporary Closing State

There are two HIP messages used for termination of a connection between the devices (see Fig. 1):

- Close - signals the end of the connection,
- CloseAck - confirms the end of the connection between the communicating devices.

We propose to remove the CloseAck message. Such a modification enables us to also remove the state (in the protocol state machine) between the messages when the device is waiting for the confirmation message, labeled as a temporary closing state.

This proposal saves the processor time for execution of instructions intended for message processing and the time in which the device is only waiting for the confirmation message.

Such a solution is intended for the IoT area, where the server is powered directly from a grid (i.e. an "unlimited" energy source) and the IoT device is powered by a battery (i.e. a limited energy source). The following situations can occur:

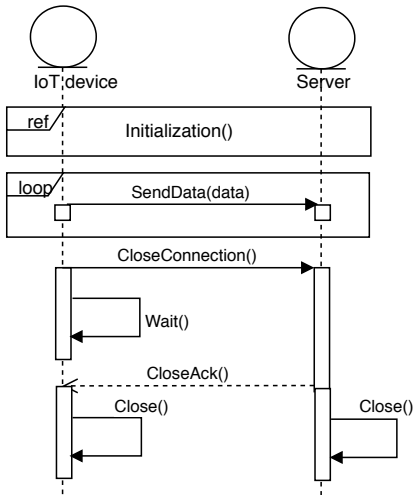


Fig. 1. The communication process of the HIP protocol.

1) The IoT device terminates the connection by sending the Close message, afterward, the server ends the connection by receiving this message. Also, the reverse situation can occur.
2) The IoT device terminates the connection, but the server does not receive the message due to a message loss. In such a case, the server waits for a time-out timer and ends the connection upon its expiration.

Due to the second situation, we have to state a precondition: only an IoT device can terminate the connection, because waiting for a timer expiration on the IoT device is not energy efficient.

B. Reduction of the Parameter Format

During the association of communicating devices, there are four types of messages exchanged: I1, R1, I2, and R2.

The R1 and I2 messages carry multiple parameters (e.g. the R1 contains: Puzzle, DH-R, HI-R, HIP, Transform, ESP Transforms, Echo Request, SIG). All the messages include information about a type and a length. In our work, we propose to remove the type and length fields of parameters. For this purpose, it is necessary to define a fixed order of the parameters. A difference between the packets before and after the proposed modification is shown below.

```
HIP Parameters Before:
ESP_INFO (type=65. length=12)
Reserved: 0x0000
...
R1_COUNTER (type=321. length=20)
Reserved: 0x00000000
...
**********************************
HIP Parameters After:
ESP_INFO
Reserved: 0x0000
...
R1_COUNTER
Reserved: 0x00000000
...
```

Using this modification, it is possible to reduce the sending message size by 32B. In the prototype solution, we have modified the I2 message as a proof-of-concept and saved 16B.

C. Removal of the HI-R Parameter

The next partial proposal of the modification is to remove the HI-R (Host Identity - Responder) parameter, which is used as a public key in the cryptographic algorithms, such as RSA, DSA, etc. By its removing, it is also possible to reduce the message size. This parameter consists of three parts:

1) information about the size of the exponent,
2) the exponent, and
3) the modulus.

The RSA security is based on the size of the modulus. In the prototype solution, we have used the size of 128B (i.e. 1024b). Nowadays, such a size is not recommended by the NIST organization [18]. The minimal recommended size of the key is 2048b.

We have proposed two solutions, manual and automatic, each suitable for different kind of applications.

1) Manual solution: The manual solution is based on manual uploading of the mentioned HI-R information on the device. We can reduce the packet size by 128B from the whole size of the message (638B).

Advantages and disadvantages of the proposed solution:

- Re-generating and re-uploading of public parameters from the server to the device can be impossible. It depends on the quantity and distance of devices.
- Re-generating and re-uploading of public parameters is possible during the charging of some specific devices.
- It is necessary to regenerate keys on all devices at the same time.

The preconditions of the proposed solution:

- A simple access to all devices by the administrator (e.g. within a single building).
- It must be possible to power off the device for some time.

2) Automatic solution: For the automatic solution, an action of the administrator is not needed. The key is re-generated automatically in specified periods.

The basic working principle:

- Sending and uploading the parameters into a persistent memory of the device during the initial communication.
- Using the same parameters during the whole defined period. (Note: Even after restarting and re-initialization of the protocol).
- Re-generating, re-sending, and uploading of parameters after the specified period expire.

For both solutions, there are some possible applications. For example, the manual solution can be used within devices monitoring the patients' health during the clinical examinations, and the automatic solution can be used within devices monitoring the fullness of the container distributed around the city.

D. Removal of the HIT Parameter

The Host Identity Tag (HIT) is a 128-bit hash of the public key. Based on this hash value, the device can be identified using the same address-space size as in the IPv6 protocol. All the initialization messages of the HIP protocol contain two HIT fields, one for a responder and one for a sender. We propose to remove these fields and thus shorten the packet length. The following size of the messages would be reduced:

1) I1: 32B from 86B (i.e. 37%),
2) R1: 32B from 638B (i.e. 5%),
3) I2: 32B from 702B (i.e. 4.5%),
4) R2: 32B from 262B (i.e. 12.2%).

The HIT parameter is used in a variety of security functions, e.g. verifying the signature or puzzle mechanism, where it is used for validating the solution. The HIP puzzle mechanism protects the server from denial-of-service attacks [19]. To not decrease the security level of the protocol, we decided to not implement this solution in our prototype at this time. A similar solution is already implemented for data packets, where the HIT parameter is replaced by the SPI (Security Parameter Index) parameter.

E. Replacement of the SPI Parameter

The HIP protocol is used in combination with other protocols, such as IPSec.

The minimal requirement for data transmission in the protocol implementation is the use of the ESP transport mode. We also used this mode in the prototype. The ESP transport mode includes the SPI parameter for mapping the instance of the protocol and the corresponding SA (Security Association) to the relevant device, identified by the HIT parameter.

The proposed solution is to remove the ESP INFO parameters in the control messages and thus reduce the size of packets (i.e. 24B reduced in the I2 and R2 messages).

The identified disadvantages:

1) According to the HIP documentation, it is forbidden to map the instance to an IP address, which could happen if we implemented this proposal along with the previous one (i.e. the anonymous-location feature would be lost).
2) Limited protocol extensions enabling mapping of multiple SAs (identified by different SPI values) to the same device.

F. Optimization of Computing Tasks

The last proposed solution is an optimization of computing tasks, such as computing of Puzzle, hash, etc. We have identified two possibilities:

1) Optimization by Assembler: For better use of the processor and its resources, some parts of code (communication library, protocol stack) can be implemented in the assembler language. It is necessary a good knowledge of the IoT system and the assembler programming language. On the other side, nowadays, there are good compilers that make assembler-based optimization less effective.

2) Optimization by Hardware: This optimization means an optimization of computing tasks by dedicated hardware parts, supporting cryptographic, hash, and similar functionality. By using specialized hardware, the resource-intensive tasks can be processed more efficiently and the processor time can be spared. However, the energy efficiency of the hardware part must be carefully considered.

IV. Experimental Results

To verify the functionality of the proposed solution, we have implemented a prototype so far, as a proof-of-concept. The proposed modifications are implemented into an open-source implementation of the HIP protocol, called OpenHIP[1] [10]. Three of the six proposals were realized, namely Removal of the CloseAck message and the temporary closing state, Reduction of the parameter format and Removal of the HI-R parameter (the manual solution).

The prototype was tested on the topology illustrated in Fig. 2, which consists of two RPi 3 (Raspberry Pi) microcomputers (Raspbian operating system), interconnected by the

[1]Source code: https://github.com/rektide/openhip, version: openhip-0.9

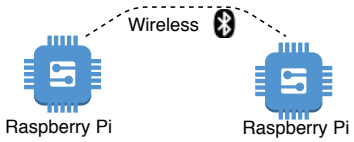


Fig. 2. The prototype architecture.

integrated Bluetooth 4.1 modules. The Bluetooth was used as part of the RPi and for the purpose of testing functionality.

After modifying the protocol by the above-mentioned optimizations (called E-HIP), the connection between two devices was successfully established and the communication worked correctly. Thus, the experiment has proved that the implemented optimizations are realizable and the protocol function is not corrupted. During the testing, we have measured the power consumption of the HIP protocol before (OpenHIP) and after (E-HIP) the modification. It represents an improvement of about 20%. The achieved results were compared to the existing works [17] in Fig. 3. Based on the comparison, we can tell that the achieved efficiency is comparable to other improvements. However, the proposed modifications are unique, and thus they can be further combined with other proposals to maximize the HIP energy efficiency.

V. Conclusions

There are some optimized security protocols used in IoT area, however, the battery-powered device lifetime is still limited. This was the target of our work. Based on the analysis, we decided to use the OpenHIP protocol and proposed its modification for efficient use in this area. The energy intensity of the proposed solution is comparable to other existing works. We can expect some contributions as reduction of network load, reduction of processor-time usage and reduction of energy required for communication. The modified protocol E-HIP is not compatible with standard one, but the application-specific communication requires the application-specific protocols. Also, there is no impact on the data plane, because the modification is focused only on the control plane.

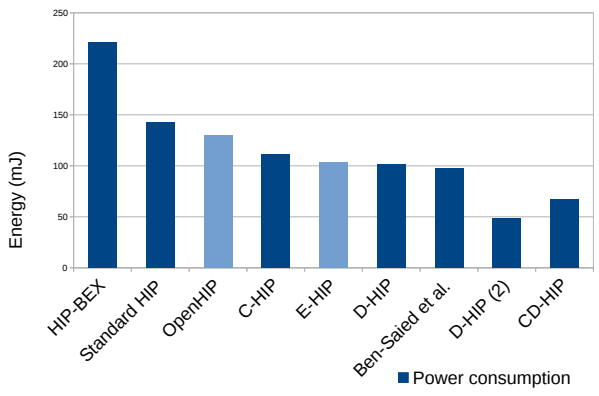


Fig. 3. The comparison of the power consumption.

In our further work, we are going to test it in a real IoT network using a low-power communication technology for IoT/mMTC (e.g. Bluetooth Low Energy, NB-IoT, LTE Cat-M1, etc.). The future work can also bring some possibilities for improvement since there are still unimplemented parts of the proposed solution that could be incorporated or it can be combined with some existing optimizations (e.g. C-HIP, D-HIP) and achieve even higher energy efficiency.

References

[1] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, "Vision and challenges for realising the Internet of Things," *Cluster of European Research Projects on the Internet of Things, European Commision*, vol. 3, no. 3, pp. 34–36, 2010.

[2] J. Bartje, "The top 10 IoT application areas – based on real IoT projects," *IoT analytics*, 2016.

[3] B. Dorsemaine, J.-P. Gaulier, J.-P. Wary, N. Kheir, and P. Urien, "Internet of Things: A definition & taxonomy," in *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*. IEEE, 2015, pp. 72–77.

[4] O. El Mouaatamid, M. Lahmer, and M. Belkasmi, "Internet of Things security: Layered classification of attacks and possible countermeasures," *Electronic Journal of Information Technology*, no. 9, 2016.

[5] J. Morgan, "A simple explanation of 'The Internet of Things'," *Forbes*, May 2014.

[6] U. Banerjee, C. Juvekar, S. H. Fuller, and A. P. Chandrakasan, "eeDTLS: Energy-efficient datagram transport layer security for the Internet of Things," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 2014–2019.

[7] S. Raza, H. Shafagh, K. Hewage, R. Hummen, T. Voigt *et al.*, "Lithe: Lightweight secure CoAP for the Internet of Things," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3711–3720, 2013.

[8] M. Lavanya and V. Natarajan, "Lightweight key agreement protocol for IoT based on IKEv2," *Computers & Electrical Engineering*, vol. 64, pp. 580–594, 2017.

[9] R. Moskowitz, T. Heer, P. Jokela, and T. Henderson, "Host identity protocol version 2 (HIPv2)," Tech. Rep., 2015, rFC 7401. [Online]. Available: https://www.rfc-editor.org/info/rfc7401

[10] (2018) OpenHIP: Host identity protocol implementation. [Online]. Available: https://bitbucket.org/openhip/openhip

[11] H. Hellaoui, M. Koudil, and A. Bouabdallah, "Energy-efficient mechanisms in security of the internet of things: A survey," *Computer Networks*, vol. 127, pp. 173–189, 2017.

[12] P. Urien, "Innovative TLS/DTLS security modules for IoT applications: Concepts and experiments," in *International Internet of Things Summit*. Springer, 2015, pp. 3–15.

[13] R. Bonetto, N. Bui, V. Lakkundi, A. Olivereau, A. Serbanati, and M. Rossi, "Secure communication for smart IoT objects: Protocol stacks, use cases and practical examples," in *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2012, pp. 686–692.

[14] Y. B. Saied and A. Olivereau, "HIP Tiny Exchange (TEX): A distributed key exchange scheme for HIP-based Internet of Things," in *Third International Conference on Communications and Networking*. IEEE, 2012, pp. 1–8.

[15] R. Moskowitz and R. Hummen, "HIP Diet EXchange (DEX)," Tech. Rep., 2017, draft-ietf-hip-dex-06. [Online]. Available: https://tools.ietf.org/html/draft-ietf-hip-dex-06

[16] R. Hummen, J. Hiller, M. Henze, and K. Wehrle, "Slimfit-A HIP DEX compression layer for the IP-based Internet of Things," in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2013, pp. 259–266.

[17] S. Sahraoui and A. Bilami, "Efficient HIP-based approach to ensure lightweight end-to-end security in the internet of things," *Computer Networks*, vol. 91, pp. 26–45, 2015.

[18] E. Barker, "Recommendation for key management part 1: General (revision 4)," *NIST Special Publication*, vol. 800, no. 57, pp. 1–160, 2016.

[19] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host identity protocol," Tech. Rep., 2008, rfc 5201. [Online]. Available: https://www.rfc-editor.org/info/rfc5201