



Sveučilište u Zagrebu  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Valter Vasić

**PROTOKOL ZA SIGURNO  
DOGOVARANJE KRIPTOGRAFSKI  
PRILAGODLJIVE KOMUNIKACIJE  
NEOVISAN O SLOJU**

DOKTORSKI RAD

Zagreb, 2016.



Sveučilište u Zagrebu  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Valter Vasić

**PROTOKOL ZA SIGURNO  
DOGOVARANJE KRIPTOGRAFSKI  
PRILAGODLJIVE KOMUNIKACIJE  
NEOVISAN O SLOJU**

DOKTORSKI RAD

Mentor: izv. prof. dr. sc. Miljenko Mikuc

Zagreb, 2016.



University of Zagreb  
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Valter Vasić

# **SECURE LAYER-AGNOSTIC AGREEMENT PROTOCOL FOR CRYPTOGRAPHICALLY AGILE COMMUNICATION**

DOCTORAL THESIS

Supervisor: Associate Professor Miljenko Mikuc, PhD

Zagreb, 2016

Doktorski rad izrađen je na Sveučilištu u Zagrebu, Fakultetu elektrotehnike i računarstva na Zavodu za telekomunikacije.

Mentor: izv. prof. dr. sc. Miljenko Mikuc

Doktorski rad ima: 89 stranica

Doktorski rad br.: \_\_\_\_\_

## O mentoru

Miljenko Mikuc doktorirao je 1997. godine na Sveučilištu u Zagrebu u području elektrotehnike. Trenutno je izvanredni profesor na Fakultetu elektrotehnike i računarstva, Zavodu za telekomunikacije na istom sveučilištu. Njegova područja istraživanja uključuju dizajn logičkih sklopova, mrežne protokole, simulacije računalnih mreža i računalnu sigurnost. Sudjelovao je u 7 znanstvenih projekata pod pokroviteljstvom Ministarstva znanosti, obrazovanja i sporta Republike Hrvatske. Bio je voditelj 2 projekta pod pokroviteljstvom istog ministarstva i voditelj zajedničkih istraživačkih projekata sa “The Boeing Company- IDS, LabNet Analysis, Modelling Simulation and Experimentation”, “International Computer Science Institute” i “The FreeBSD Foundation” iz SAD-a te Ericssonom Nikola Tesla d.d. iz Zagreba. Trenutno je voditelj istraživačkog projekta: “Prilagođen IMUNES za Ericsson (E-IMUNES)” u suradnji sa tvrtkom Ericsson Nikola Tesla d.d. Objavio je više od 40 radova u časopisima i na konferencijama u području komunikacijskih mreža, protokola, virtualizacije, formalnih metoda i sigurnosti.

Nositelj je dva predmeta na preddiplomskom studiju: “Digitalna logika” i “Mrežno programiranje”, dva predmeta na diplomskom studiju: “Sigurnost u Internetu” i “Upravljanje mrežom i uslugama”, i dva predmeta na postdiplomskom doktorskom studiju: “Odabrana poglavlja komunikacijskih protokola” i “Formalizmi u telekomunikacijama”.

Pod njegovim vodstvom diplomirala su 123 studenta po programu ETF-4 i FER-1, 28 studenata obranilo je završne radove na preddiplomskom studiju, a 25 studenata obranilo je diplomske radove na diplomskom studiju po programu FER-2. Na poslijediplomskom studiju, pod njegovim vodstvom, 15 studenata je obranilo magistarske radove. Jedna studentica obranila je svoju doktorsku disertaciju. Trenutno je mentor troje studenata na doktorskom studiju.

## About the Supervisor

Miljenko Mikuc received his PhD in Electrical Engineering from University of Zagreb, Croatia, in 1997. He is currently Associate Professor at the Faculty of Electrical Engineering and Computing, Department of Telecommunications within the same university. His research interests include digital logic design, network protocols, network simulation and security. He participated in 7 scientific projects financed by the Ministry of Science, Education and Sports of the Republic of Croatia. He was a project leader of 2 projects of Applications of Information Technology financed by the same Ministry and the project leader of cooperation research projects with “The Boeing Company – IDS, LabNet Analysis, Modelling Simulation and Experimentation”, “International Computer Science Institute” and “The FreeBSD Foundation” from USA and with Ericsson Nikola Tesla d.d. from Zagreb. Currently he is a project leader of the research project: “Ericsson Customized IMUNES (E-IMUNES)” in cooperation with Ericsson Nikola Tesla d.d. company. He published more than 40 papers in journals and conference proceedings in the area of communication networks, protocols, virtualization, formal methods and security.

He is a lecturer in charge on two undergraduate study courses: “Digital Logic” and “Network Programming”, two graduate study courses: “Internet Security” and “Network and Service Management”, and two postgraduate doctoral study courses: “Selected topics in communication protocols” and “Formalisms in telecommunications”.

Under his supervision 123 students graduated on graduate study program ETF-4 and FER-1, 28 students defended their bachelor thesis on undergraduate (BSc) study program FER-2, 25 students defended their master thesis. On postgraduate master study program, under his supervision 15 students defended their master thesis. One student defended her doctoral thesis. He is currently dissertation advisor for 3 PhD students.

## Zahvala

Prije svega htio bih zahvaliti svom mentoru Miljenku Mikucu na podršci tijekom cijelog doktorskog studija te motivaciji u istraživanju i praktičnom radu u području sigurnosti. Hvala mu što je uvijek bio spreman saslušati moje probleme te sa mnom dijeliti sreću nakon njihovog rješavanja.

Također bih htio puno zahvaliti kolegama sa Zavoda za telekomunikacije Marinu Vukoviću, Krešimiru Pripužiću i Aleksandru Antoniću na pomoći u pisanju znanstvenih članaka u kojima su primjenjena istraživanja iz mog doktorskog rada. Značajnu ulogu u završnoj fazi doktorata imao je Denis Salopek, koji je u svakodnevnom poslu bio pun podrške i razumijevanja. Hvala kolegi Marku Zecu na iskustvu i znanju koje je podijelio sa mnom, ali prije svega hvala na (najboljem) mrežnom emulatoru IMUNES. Snagu za dovršavanje doktorata dugujem i Marini Ivić, Martini Marjanović i Marku Paveliću, koji su se svakodnevno družili samnom.

Ovaj doktorat ne bi bio moguć bez moje najbolje prijateljice i ljubavi Sanje, koja je cijelim putem vjerovala u mene i davala mi snagu kad je najviše trebalo. Hvala mom najboljem prijatelju Borisu, koji mi je uvijek bio spreman dati realnu sliku mojih, često preoptimističnih, ideja. Hvala mojim roditeljima i sestri na svim motivacijskim razgovorima te na bezuvjetnoj ljubavi i podršci kroz cijelo moje školovanje.

Doktorat posvećujem svojoj noni Kati čijih bih se riječi sjetio svaki put kada bih zapeo na svome putu.

## Sažetak

Sigurna komunikacija ključan je dio suvremenih sustava za komunikaciju i može se postići korištenjem kriptografskih algoritama i protokola. Međutim, i u takvom sustavu sigurnost komunikacije može biti ugrožena ako se koriste kriptografski algoritmi kojima su otkrivene ranjivosti. Koncept kriptografski prilagodljive komunikacije omogućuje dinamičku promjenu kriptografskih algoritama i ključeva za vrijeme rada sustava bez promjene izvedbene logike tog sustava. U doktorskoj disertaciji dizajniran je protokol za sigurno dogovaranje kriptografski prilagodljive komunikacije koji, za razliku od postojećih rješenja, može neovisno o komunikacijskom sloju i korištenoj platformi dogovoriti preduvjete za daljnju sigurnu komunikaciju. Model protokola je formalno verificiran u alatu za automatsku verifikaciju sigurnih postavki protokola. Dogovor preduvjeta izведен je na učinkovit način koji ne postavlja visoke zahtjeve na računalnu moć i propusnost mrežne komunikacije. Izvedena je integracija protokola za dogovor u okolini Interneta stvari uzimajući u obzir mogućnosti različitih uređaja u okolini. Dodatno, izvedeno je programsko ostvarenje koje omogućava korištenje protokola u raznim mrežnim modelima i okruženjima na različitim mrežnim slojevima. U sklopu emulirane mrežne okoline pokazana je učinkovitost mrežne komunikacije na različitim komunikacijskim slojevima i korištenjem različitih vrsta kriptografije uz osvrt na ugrađene sigurnosne mehanizme.

U okviru doktorske disertacije ostvareni su sljedeći znanstveni doprinosi:

1. Protokol za sigurno dogovaranje kriptografskih algoritama neovisan o sloju protokolnog složaja, aplikaciji i operacijskom sustavu.
2. Metode za integraciju protokola u aplikacije koje koriste različite modele komunikacije, a posebice mreže ravnopravnih čvorova i model klijent-poslužitelj.
3. Ocjena učinkovitosti protokola u simuliranom i stvarnom mrežnom okružju.

**Ključne riječi:** kriptografska prilagodljivost, sigurna komunikacija, autentificirani dogovor ključeva, dogovor algoritama, neovisnost o sloju, formalno verificirani model

## Extended abstract

### Secure layer-agnostic agreement protocol for cryptographically agile communication

Securing data and communication channels implies the use of cryptography. To achieve primary security requirements (confidentiality, integrity and availability) different types of cryptographic algorithms are used. Cryptographic algorithms can be divided into three groups: symmetric encryption, asymmetric encryption and cryptographic hashes. These algorithms are often used in different combinations to achieve better performance and various security requirements. There is a vast amount of cryptographic algorithms currently available. They are all constantly being improved but at the same time new vulnerabilities are being discovered and exposed. An algorithm with vulnerabilities, from the cryptanalysis point of view, can still be safe for usage, but the identified vulnerability points to a weakness in its definition. Such a weakness can eventually be used for various attacks on systems that are secured by that algorithm, so vulnerable algorithms should be replaced as soon as possible.

However, replacing cryptographic algorithms that are tightly integrated into communication protocols would eventually require changing the protocols themselves. This usually means that the running software or firmware needs to be updated. Changing protocols may prove to be an issue when dealing with typically large and complex communication systems. This is the reason why cryptographic algorithms should never be tightly integrated into a communication protocol and the solution is to use the concept of cryptographic agility. Cryptographic agility is characterized by the ability to change cryptographic algorithms and keys while using the same protocols and deployed systems. On the other hand, using agility makes it possible to use the same cryptographic algorithms or protocols over a wide array of communication protocols thus enabling easier replacement of potentially insecure cryptographic algorithms.

Cryptographic agility is a known principle that is already integrated into most widely deployed secure channel protocols, such as IPsec, SSL, TLS and SSH. However, each of the implemented agility mechanisms is strictly tied to that solution and cannot be easily extracted as a standalone mechanism applicable for other purposes. The possible solution that would enable this functionality is an adaptable cryptographically agile protocol that is proposed in this thesis. This solution can be used on various network layers, architectures and devices. Furthermore, the aim is the agreement of cryptographic prerequisites (algorithms and keys) in order to enable various security requirements in an interconnected environment. Using the proposed cryptographically agile solution enables protection of protocols that don't have any security mechanisms deployed and enable development of new secure protocols on top of the agreed primitives.

---

The doctoral thesis describes a novel, lightweight and adaptable solution which is easily applicable in existing networked environments and currently emerging areas, such as Internet of Things (IoT). Security isn't always integrated in current IoT solutions and is typically planned at a later stage as an upgrade. The communication between IoT devices needs to be secure which requires an agreement protocol to enable communicating parties to agree on a cryptographic algorithm and keys used to protect the exchanged messages. The IoT ecosystem needs a flexible and adaptable agreement mechanisms because IoT devices have limited computing and bandwidth resources. Secure communication is possible only after communicating devices agree on a set of cryptographic algorithms and keys. Since IoT services are driven by the underlying data sources, it is critical to validate a sensor as a credible data source and to protect the exchanged data.

## **Chapter 2 - Cryptography**

The second chapter describes basic cryptographic primitives, cryptographic hashes, symmetric and asymmetric cryptosystems and focuses on their application in current communication systems. It specifically describes the primitives used for achieving an agile and adaptive agreement on cryptographic keys and algorithms. The needed primitives include a HMAC (Hashed Message Authentication Code) algorithm based on cryptographic hashes with a secret key, digital signatures for authentication that combine cryptographic hashes with asymmetric encryption algorithms and the Diffie-Hellman key exchange that uses asymmetric cryptography to generate secret data to conduct a secure exchange. These three mechanisms are needed to conduct a SIGMA (sign-and-mac) exchange between two communicating parties. The SIGMA exchange represents an authenticated key exchange algorithm that secures the exchange from man in the middle attacks by protecting the data used for digital signature verification with HMAC and vice versa. The resulting secret data needs to be further processed by using a key derivation function (KDF) based on hash functions (Hashed KDF). This is to ensure that keys generated from the same secret data are computationally independent. Computational independence represents that the discovery of one key won't reveal any data on other keys calculated from the same secret data.

## **Chapter 3 - Cryptographically agile communication**

The third chapter describes modern cryptographically agile protocols. Most of these protocols are secure channel protocols that have many components and thus tend to be fairly complex (e.g. SSL/TLS, IPsec, SSH). That complexity makes it hard, or even impossible in certain cases, to create a formal model that can be thoroughly verified to achieve provable security. The lack of security proofs is probably one of the causes of the recent increase of attacks on SSL/TLS.

---

Verification of certain solutions can also reveal previously unknown vulnerabilities. Protocol complexity also increases the possibility of implementation problems which can cause significant security problems even if the protocol design is provably secure. The chapter focuses on describing established cryptographically agile solutions and compares them to modern solutions like tcpcrypt, MinimaLT and QUIC that employ a less complex approach to secure channels. All these solutions are compared to ACAP, the lightweight layer-agnostic agreement protocol described in the thesis. The main advantages of ACAP are a layer-agnostic architecture that enables its application on all communication layers and a simple architecture that enables formal verification and reduces the possibility of implementation problems. ACAP represents an easy to integrate and adaptable building block that distributes secure communication prerequisites i.e. cryptographic keys and algorithms.

## **Chapter 4 - Secure cryptographic agreement protocol**

The ACAP protocol is described in the fourth chapter. It consists of four messages:  $\text{INIT}_I$ ,  $\text{INIT}_R$ ,  $\text{LIST}_I$  and  $\text{LIST}_R$ . After the message exchange, both communicating parties have access to the following data: secret key generated from the Diffie-Hellman shared secret, public keys of both sides for authentication and a list of cryptographic algorithms supported by both sides. The list of supported cryptographic protocols is divided into groups and agreed upon for each group. The first algorithm that is on the client list and is present on the server list is chosen as the agreed algorithm. ACAP employs the following security mechanisms: computational independence of secret keys by using HKDF, denial of service attack mitigation by separating the most expensive operations in an asynchronous background process, trust establishment that enables both manual and automatic trust verification through PKI. Additionally, the SIGMA design prevents the man in the middle and replay attack on the protocol. At the end of the chapter a comprehensive comparison of the SSH transport layer protocol negotiation and ACAP is given.

## **Chapter 5 - Formal model specification**

The formal security model and formal model verification is described in chapter five. The formal security model is described in the security protocol description language (SPDL) used by the Scyther security protocol verifier. Models in SPDL are defined by using protocols and roles. Each protocol can contain two or more roles that define multiple send and receive events. A comprehensive explanation of the model specification process is given on the Needham-Schroeder-Loewe protocol is give. After that the ACAP protocol model is described. The model is specified by using two protocols, a main protocol with the initiator and responder roles and a helper protocol that enabled the specification of the Diffie-Hellman key exchange. After

---

the message exchange specification the following security requirements (Scyther claims) were defined, immutability of the DH exponentials, secrecy of the shared secret used to derive session keys and the inability of the attacker to interfere with the message exchange. All three security requirements were successfully verified by using an unbounded amount of protocol runs.

## **Chapter 6 - Secure communication in the Internet of things**

Chapter six describes how the designed solution can be introduced in the Internet of Things environment. It focuses on the device capabilities and introduces three device tiers, sensors, middle and cloud tier. All secure communication scenarios are conducted between devices from all tiers. The main proposed secure communication scenarios include authenticated device management, reliable data collection and sensitive data protection that are built on a platform independent architecture. After scenario descriptions a secure communication model for the IoT environment is given. The model consists of devices, public keys, communication layers, hardware capabilities, cryptographic algorithms and a set of secure communication operations. Secure communication operations are performed to enable all previously described communication scenarios. Furthermore, this chapter describes an adaptive cryptographic algorithm agreement procedure that takes into account the required level of security, cryptographic algorithm complexity, device battery power, bandwidth and processing power to choose a cryptographic algorithm that is best suited in the current environment.

## **Chapter 7 - ACAP protocol implementation**

A prototype implementation in Python and measurement results are described in chapter seven. The proposed prototype architecture is composed from five key components: calculation and verification for cryptographic operations, message handling, layer independent communication, cryptographic algorithm agreement and communication with external applications. The prototype implementation supports communication on the following layers: Ethernet, IPv4, IPv6, TCP, UDP and SCTP with the same message formats that are shown in the appendix. It also supports standard and elliptic curve asymmetric algorithms for Diffie-Hellman exchanges and digital signatures. The prototype was tested in an emulated network environment in the IMUNES tool, which simplified network setup and enabled reliable measurements through environment automation. The measurements include the effect of public key size on protocol message size, agreement duration in regards to link delay and available bandwidth. They have shown that the solution is bandwidth efficient as the entire negotiation is performed in under two seconds on a 10 Kbps link. The use of elliptic curve cryptography is encouraged, especially for devices with constrained resources. Furthermore, this chapter provides an overview of cryptographic operation complexity for client and server which shows the deployed denial of

---

service mitigation mechanisms.

Finally, chapter eight concludes the thesis by giving a comprehensive overview of the designed layer-agnostic protocol that enables cryptographic agility for various devices and platforms.

The scope of this doctoral thesis covers the following scientific contributions:

1. Secure layer-agnostic agreement protocol for cryptographically agile communication that is application and operating system independent.
2. Methods for integrating the protocol with applications that use different communication models, especially P2P networks and client-server environments.
3. Evaluation of protocol communication and agreement efficiency in a simulated and real network environment.

**Keywords:** cryptographic agility, secure communication, layer-agnostic design, authenticated key exchange, algorithm agreement, formal model verification

# Sadržaj

<b>1. Uvod</b>	1
<b>2. Kriptografija</b>	4
2.1. Kriptografski algoritmi	4
2.2. Kriptografski sažetak	4
2.2.1. HMAC	5
2.3. Simetrični kriptosustav	6
2.3.1. Šifriranje tokova podataka	7
2.3.2. Šifriranje blokova podataka	7
2.3.3. Autentificirano šifriranje	9
2.4. Asimetrični kriptosustav	10
2.4.1. Digitalni potpis	10
2.4.2. Digitalna omotnica	11
2.4.3. Dogovaranje zajedničke tajne	11
2.4.4. Algoritmi temeljeni na eliptičnim krivuljama	12
2.5. Autentificirano dogovaranje zajedničke tajne	13
2.5.1. SIGMA	14
2.6. Pseudo nasumične funkcije	14
2.6.1. Funkcije izračunavanja ključeva	15
<b>3. Kriptografski prilagodljiva komunikacija</b>	16
3.1. Pregled kriptografski prilagodljivih protokola	17
3.1.1. Protokoli SSL/(D)TLS	18
3.1.2. Protokol IPsec	19
3.1.3. Protokol SSH	21
3.1.4. Protokoli nove generacije	22
3.2. Dogovor zajedničke tajne i kriptografskih algoritama	22
3.2.1. Novi algoritmi dogovora zajedničke tajne	23
3.3. Kriptografska prilagodljivost u okolini Interneta stvari	23

<b>4. Protokol za sigurno dogovaranje kriptografskih algoritama . . . . .</b>	<b>25</b>
4.1. Označavanje kriptografskih operacija . . . . .	26
4.2. Poruke i tok komunikacije . . . . .	27
4.2.1. Rukovanje iznimkama . . . . .	28
4.3. Kriptografski algoritmi i ključevi za dogovaranje . . . . .	28
4.4. Dogovor kriptografskih algoritama . . . . .	29
4.5. Analiza sigurnosti i mehanizama protokola ACAP . . . . .	31
4.5.1. Djelomični nedostatak dogovora algoritama u ACAP razmjeni . . . . .	31
4.5.2. Računalna neovisnost tajnih ključeva . . . . .	32
4.5.3. Smanjenje utjecaja DoS napada . . . . .	32
4.5.4. Uspostavljanje povjerenja između komunicirajućih strana . . . . .	33
4.5.5. Sprječavanje napada čovjek u sredini . . . . .	33
4.5.6. Sprječavanje napada ponavljanjem poruka . . . . .	34
4.6. Usporedba protokola ACAP s protokolom za upravljanje prijenosom podataka u protokolu SSH . . . . .	34
<b>5. Formalna verifikacija modela . . . . .</b>	<b>37</b>
5.1. Značajke jezika SPDL . . . . .	37
5.2. Primjer verifikacije jednostavnog protokola . . . . .	38
5.3. Model predloženog protokola ACAP . . . . .	41
5.3.1. Modeliranje Diffie-Hellman operacija . . . . .	42
5.4. Formalna verifikacija modela . . . . .	44
5.4.1. Definicija sigurnosnih zahtjeva . . . . .	45
5.4.2. Rezultati formalne verifikacije . . . . .	45
<b>6. Primjena protokola za sigurno dogovaranje u okolini Interneta stvari . . . . .</b>	<b>47</b>
6.1. Primjene sigurne komunikacije u okolini Interneta stvari . . . . .	47
6.1.1. Arhitektura neovisna o platformi . . . . .	48
6.1.2. Autentificirano upravljanje uređajima . . . . .	49
6.1.3. Pouzdano prikupljanje podataka . . . . .	49
6.1.4. Zaštita osjetljivih podataka . . . . .	49
6.2. Model sigurne komunikacije u okolini Interneta stvari . . . . .	50
6.3. Prilagodljivost modela sigurne komunikacije . . . . .	52
6.3.1. Prilagodbe odabira algoritma . . . . .	54
<b>7. Programsко ostvarenje protokola ACAP . . . . .</b>	<b>55</b>
7.1. Arhitektura programskog ostvarenja i načini integracije . . . . .	55
7.2. Mogućnosti programskog ostvarenja . . . . .	57

7.3.	Struktura poruka . . . . .	58
7.4.	Korištenje protokola na različitim komunikacijskim slojevima . . . . .	59
7.4.1.	Protokol Ethernet . . . . .	60
7.4.2.	Protokoli IP . . . . .	60
7.4.3.	Transportni protokoli . . . . .	61
7.5.	Mjerenje performansi protokola u emuliranoj mrežnoj okolini . . . . .	62
7.5.1.	Utjecaj veličine javnih ključeva na veličinu poruka . . . . .	63
7.5.2.	Trajanje dogovora ovisno o kašnjenju . . . . .	64
7.5.3.	Utjecaj propusnosti na trajanje dogovora . . . . .	65
7.6.	Složenost kriptografskih operacija u protokolu . . . . .	65
7.7.	Dohvaćanje podržanih algoritama . . . . .	67
7.8.	Ostvareni sigurnosni mehanizmi . . . . .	67
<b>8.</b>	<b>Zaključak</b> . . . . .	<b>69</b>
<b>Dodaci</b>	. . . . .	<b>70</b>
Dodatak A.	Potpuni model protokola ACAP . . . . .	70
Dodatak B.	Ispis izvođenja programskog ostvarenja . . . . .	72
Dodatak C.	Formati poruka . . . . .	76
<b>Literatura</b>	. . . . .	<b>78</b>
<b>Životopis</b>	. . . . .	<b>87</b>
<b>Biography</b>	. . . . .	<b>89</b>

# Poglavlje 1

## Uvod

Sigurna komunikacija je vrlo važan dio današnjeg računalnog svijeta i Interneta uopće. Ona se ostvaruje korištenjem kriptografije, odnosno kriptografskih algoritama i ključeva, za ostvarivanje osnovnih sigurnosnih zahtjeva tajnosti, cjelovitosti i dostupnosti. Za ostvarivanje sigurnosnih zahtjeva koriste se različite kombinacije kriptografskih algoritama. Primjerice, za računanje digitalnog potpisa potrebni su algoritam kriptografskog sažetka i algoritam asimetričnog kriptosustava uz potrebne ključeve.

Trenutno postoji velika količina različitih algoritama iste namjene koji ostvaruju iste sigurnosne zahtjeve. Neprestano se razvijaju novi algoritmi i otkrivaju ranjivosti prethodno dizajniranih algoritama. Uz to, zbog povećanja računalne moći zahtjevi na veličinu kriptografskih ključeva neprestano rastu. Stoga je potrebno osigurati siguran dogovor kriptografskih algoritama i ključeva koji će se moći dinamički prilagođavati trenutnim uvjetima i zahtjevima za sigurnost komunikacije. Dinamički dogovor kriptografskih algoritama i ključeva definiran je u okviru pojma kriptografski prilagodljive komunikacije.

Kriptografski prilagodljiva komunikacija predstavlja princip u kojem se inačice kriptografskih algoritama ne ugrađuju izravno u aplikacije i sustave, već se samo definira koja se vrsta algoritma želi koristiti. To omogućuje naknadnu promjenu inačice algoritma bez promjene aplikacije, odnosno sustava. Suvremeni protokoli za sigurnu komunikaciju, poput TLS-a, IPsec-a i SSH-a, koriste princip kriptografski prilagodljive komunikacije. Osnovni nedostatak tih protokola je njihova složenost koja značajno otežava jednostavno programsko ostvarenje i formalnu verifikaciju njihovog dizajna. S druge strane, ti protokoli se ne mogu koristiti na uređajima s ograničenim računalnim sposobnostima koji su sastavni dio novih okolina temeljenih na Internetu stvari. Dogovor algoritama i ključeva u tim protokolima nije moguće koristiti neovisno o protokolu i primijeniti na druge komunikacijske slojeve i primjene zbog velike razine integracije i povezanosti s komunikacijskim slojem i protokolom.

Stoga je u doktorskoj disertaciji dizajniran i izведен protokol za sigurno dogovaranje kriptografski prilagodljive komunikacije koji je neovisan o komunikacijskom sloju i može se koristiti

na uređajima ograničenih računalnih sposobnosti. Jednostavna arhitektura protokola omogućila je formalno modeliranje protokola i računalnu formalnu verifikaciju tog modela. Dodatno, jednostavnost arhitekture smanjuje vjerojatnost grešaka u programskom ostvarenju i omogućava lakše održavanje sustava temeljenog na tom protokolu. Opisana je izravna primjena modeliranog protokola u okruženju Interneta stvari koja se temelji na prilagodbi odabira algoritma ovisno o trenutnim mogućnostima uređaja i potrebnoj razini sigurnosti. Uz disertaciju je izvedeno i programsko ostvarenje koje je ispitano u različitim komunikacijskim uvjetima u emuliranoj mrežnoj okolini u alatu IMUNES.

U doktorskoj disertaciji ostvareni su sljedeći znanstveni doprinosi:

1. Protokol za sigurno dogovaranje kriptografskih algoritama neovisan o sloju protokolnog složaja, aplikaciji i operacijskom sustavu.
2. Metode za integraciju protokola u aplikacije koje koriste različite modele komunikacije, a posebice mreže ravnopravnih čvorova i model klijent-poslužitelj.
3. Ocjena učinkovitosti protokola u simuliranom i stvarnom mrežnom okružju.

Protokol je definiran u poglavlju 4 i formalno verificiran u poglavlju 5. Metode za integraciju protokola u okolinu Interneta stvari prikazane su u poglavlju 6, dok je arhitektura za integraciju protokola prikazana na početku poglavlja 7. U drugom dijelu poglavlja 7 prikazana je ocjena učinkovitosti protokola na temelju mjerena performansi protokola u različitim mrežnim uvjetima.

U poglavlju 2 dan je pregled vrsta kriptografskih algoritama. Kriptografski algoritmi su podijeljeni u tri osnovne skupine: algoritmi za stvaranje kriptografskih sažetaka, simetrični kriptografski algoritmi i asimetrični kriptografski algoritmi. Kriptografski sažetak služi za zaštitu cjelovitosti podataka u digitalnim potpisima ili zaštitnim sumama. Algoritmi simetričnog kriptosustava služe osiguravanje tajnosti podataka, a algoritmi asimetričnog kriptosustava se koriste za dogovaranje i prijenos ključeva te digitalno potpisivanje podataka. Na kraju poglavlja obuhvaćeni su asimetrični algoritmi za sigurno dogovaranje zajedničke tajne koji su temelj protokola opisanog u disertaciji.

Poglavlje 3 sadrži opis trenutno korištenih protokola za sigurnu komunikaciju koji koriste principe prilagodljive komunikacije, na kojima je temeljen predloženi protokol za sigurno dogovaranje kriptografski prilagodljive komunikacije (engl. *Agile Cryptographic Agreement Protocol* - ACAP). Opisani su dijelovi tih protokola i objašnjena je njihova interna arhitektura. Na kraju poglavlja opisani su novi protokoli za sigurnu komunikaciju koji se temelje na jednostavnoj arhitekturi za postizanje sigurne komunikacije u različitim okruženjima.

Poglavlje 4 sadrži opis protokola ACAP koji služi za siguran dogovor kriptografskih algoritama i ključeva za osiguravanje buduće komunikacije. Prikazane su sve poruke u protokolu i opisana je njihova razmjena uz pregled kriptografskih operacija potrebnih za siguran dogovor. Analiziran je algoritam dogovora kriptografskih algoritama uz pregled sigurnosnih mehanizama

ugrađenih u protokol.

U poglavlju 5 opisan je jezik SPDL (engl. *Security Protocol Definition Language*) koji je korišten za definiranje i verifikaciju formalnog modela protokola ACAP na jednostavnom primjeru. Potom je opisan model protokola i sigurnosnih zahtjeva koji se automatski verificiraju korištenjem alata Scyther. Na kraju poglavlja prikazani su rezultati verifikacije uz analizu zahtjevnosti postupka verifikacije.

Primjena protokola ACAP u okolini Interneta stvari prikazana je u poglavlju 6. Na početku poglavlja opisana je arhitektura okoline Interneta stvari i prepoznate su ključne primjene protokola na komunikaciju u toj okolini. Potom je dan formalan model sigurne komunikacije koji predstavlja minimalan skup objekata i operacija potrebnih za sigurnu komunikaciju, a temeljen na dogovoru algoritama i ključeva kroz protokol ACAP. Na kraju poglavlja prikazana je prilagodljiva procedura za odabir algoritama koja uzima u obzir trenutne mogućnosti uređaja i željenu razinu sigurnosti za usporedbu algoritama.

Poglavlje 7 opisuje mogućnosti programskog ostvarenja dizajniranog protokola ACAP na različitim komunikacijskim slojevima koji uključuju protokole TCP, UDP, IP i Ethernet. Potom je dan pregled mjerena učinkovitosti protokola u sklopu emulirane mrežne okoline alata IMUNES. Mjerenja demonstriraju nisku razinu zahtjevnosti protokola na mrežnu propusnost i analiziraju složenost kriptografskih operacija uključenih u dogovor. Na kraju je dan pregled ostvarenih sigurnosnih mehanizama koji su definirani u opisu protokola ACAP.

# Poglavlje 2

## Kriptografija

Kriptografija je znanost koja omogućava sigurnu komunikaciju. Način ostvarivanja sigurne komunikacije ovisi o sigurnosnim zahtjevima koje komunikacija mora zadovoljavati. Sigurnosni zahtjevi osiguravaju se korištenjem kriptografskih algoritama koji određene matematičke principe primjenjuju za ostvarivanje tih zahtjeva. Kriptografija omogućava ostvarivanje sljedećih sigurnosnih zahtjeva: tajnosti, cjelovitosti, autentifikacije i neporecivosti.

Tajnost predstavlja komunikaciju koja je poznata samo sudionicima komunikacije. Cjelovitost osigurava da su podaci nepromijenjeni prilikom prijenosa. Autentifikacija predstavlja mogućnost dokazivanja identiteta tijekom komunikacije. Neporecivost daje mogućnost dokaza da je jedan od sudionika obavio određenu radnju.

### 2.1 Kriptografski algoritmi

Kriptografske algoritme možemo podijeliti u tri osnovne skupine:

- algoritmi za stvaranje kriptografskih sažetaka (engl. *cryptographic hash*),
- algoritmi simetričnog kriptosustava (engl. *symmetric cryptosystem*) i
- algoritmi asimetričnog kriptosustava (engl. *asymmetric cryptosystem*).

U nastavku slijede kratki opisi navedenih vrsta kriptografskih algoritama uključujući i njihovu primjenu u osiguravanju sigurnosnih zahtjeva.

### 2.2 Kriptografski sažetak

Funkcija za stvaranje kriptografskog sažetka jednosmjerna je funkcija koja sažima niz podataka proizvoljne veličine u niz podataka točno određene veličine. Kriptografski sažetak najčešće se koristi za provjeru cjelovitosti podataka. Da bi se funkcija  $h$  smatrala jednosmjernim kriptografskim sažetkom (engl. *one-way hash function*), treba zadovoljavati sljedeća svojstva [1, str. 544]:

1. Argument  $X$  može biti proizvoljne duljine, a rezultat  $h(X)$  je fiksne duljine od  $n$  okteta.

Izračunavanje sažetka se uobičajeno zapisuje:

$$h(X) = Y; \text{duljina}(Y) = n$$

2. Funkcija sažetka  $h$  mora biti jednosmjerna na način da je za ponuđeni  $Y$ , računalno neisplativo tražiti poruku  $X$  za koju vrijedi  $h(X) = Y$  (engl. *preimage resistant*).
3. Također, za danu poruku  $X$  i  $h(X)$  mora biti računalno neisplativo tražiti  $X' \neq X$  takav da je  $h(X') = h(X)$  (engl. *second preimage resistant*).
4. Poželjno svojstvo jednosmjernih funkcija sažetaka je otpornost na podudaranje (engl. *collision resistant*), odnosno da je računalno teško (i neisplativo) pronaći bilo koje dvije različite poruke koje daju isti sažetak.

Kriptografski sažetak se koristi za razne primjene: brzo uspoređivanje datoteka, adresiranje podataka u programiranju, spremanje podataka za autentifikaciju korisnika i provjeru cjelovitosti. Ako se provjera cjelovitosti koristi prilikom komunikacije, kriptografski sažetak se mora dodatno zaštititi kako ga napadač ne bi mogao promijeniti. Dodatna zaštita osigurava se korištenjem asimetrične kriptografije, a rezultat je digitalni potpis.

Sigurna provjera cjelovitosti može se obaviti i isključivo korištenjem kriptografskog sažetka ako komunicirajuće strane imaju na raspolaganju dijeljenu zajedničku tajnu. Zajednička tajna se tada uključuje u izračun sažetka kako napadač ne bi mogao promijeniti poruku i izračunati novi sažetak za provjeru cjelovitosti i autentičnosti poruke. Ova metoda izračunavanja naziva se izračun zaštitne sume korištenjem algoritama kriptografskog sažetka (engl. *Hashed Message Authentication Code - HMAC*) i pobliže je opisana u potpoglavlju 2.2.1.

Postoji velika količina algoritama za kriptografski sažetak, ali se za osiguravanje komunikacije najviše koriste MD5 (engl. *Message Digest*) i SHA (engl. *Secure Hash Algorithm*) algoritmi. Ako algoritam sažetka nije ranjiv na kriptografske napade, duljina sažetka dodatno povećava sigurnost algoritma jer povećava prostor mogućih vrijednosti sažetka i smanjuje mogućnost podudaranja sažetaka. Duljine navedenih sažetaka mogu biti sljedeće:

- MD5 - 128 bita,
- SHA1 - 160 bita,
- SHA2/SHA3 - 224, 256, 384 ili 512 bita.

## 2.2.1 HMAC

HMAC (*Hashed Message Authentication Code*) je poseban slučaj algoritma MAC (*Message Authentication Code*) koji služi za izračun podatka za autentifikaciju poruka u komunikaciji korištenjem kriptografskog sažetka. Algoritam MAC iz poruke  $X$  i ključa  $K$  računa autentika-

cijski kod  $A$ :

$$MAC_K(X) = A$$

Za generiranje takvog podatka mogu se koristiti različite metode, ali najbolji rezultati u praksi se postižu korištenjem kriptografskih funkcija [1, str. 746].

Postupak kreiranja rezultata HMAC s ključem  $K$  je za poruku  $X$  definiran je sljedećim izrazom [1, str. 559]:

$$HMAC_K(X) = h( (K \oplus opad) || h((K \oplus ipad) || X) )$$

Konstante  $opad$  i  $ipad$  su nizovi okteta duljine ključa  $K$ , znak  $||$  predstavlja operaciju konatenacije nizova, a  $\oplus$  operaciju isključivo ili (XOR).  $opad$  je niz okteta sadržaja 0x36, dok je  $ipad$  niz okteta sadržaja 0x5C. Algoritam HMAC može koristiti bilo koji algoritam za stvaranje kriptografskog sažetka prilikom izračuna rezultata.

S obzirom na to da HMAC koristi jedan ključ za sve poruke koje se razmjenjuju u komunikaciji, njegovim korištenjem ne osigurava se neporecivost poruke, ali HMAC predstavlja vrlo brz način izračunavanja podataka za provjeru cjelovitosti poruke i autentifikaciju pošiljatelja, zato što je stvaratelj poruke  $X$  i  $HMAC_K(X)$  morao biti upoznat s ključem  $K$ .

Kako bi se određeni ključ  $K$  mogao koristiti za HMAC operacije on mora biti jednak duljini rezultata algoritma za izračunavanje sažetka. Primjerice, ako se koristi algoritam MD5 tajni ključ mora biti duljine barem 128 bita, a ako se koristi algoritam SHA-256 tada bi ključ trebao biti duljine barem 256 bita.

## 2.3 Simetrični kriptosustav

Simetrični kriptosustav označava sustav u kojem mora postojati jedan ključ s kojim se odvijaju operacije u sustavu. Ključ koji se koristi za izvođenje operacija je zajednička tajna između komunicirajućih strana. Simetrični kriptosustav za razliku od kriptografskog sažetka mora imati ključ, dok se kriptografski sažetak može računati bez ključa ili s ključem u slučaju HMAC-a.

Cilj simetričnog kriptosustava je osiguravanje tajnosti komunikacije, stoga su definirane dvije osnovne operacije u sustavu: šifriranje podataka u šifrat i dešifriranje šifrata u izvorne podatke. Operacije šifriranja poruke  $X$  u šifrat  $Y$  i dešifriranja šifrata u poruku korištenjem ključa  $K$  definirane su sljedećim izrazima:

$$E_K(X) = Y; D_K(Y) = X$$

Simetrični kriptosustavi dijele se u dvije osnovne skupine s obzirom na način obrade podataka. Postoje algoritmi koji šifriraju tokove podataka (engl. *stream ciphers*) i algoritmi koji šifriraju blokove podataka (engl. *block ciphers*).

### 2.3.1 Šifriranje tokova podataka

Kod šifriranja tokova podataka niz znakova poruke  $m_0, m_1, m_2, \dots$  šifrira se u niz znakova šifrata  $c_0, c_1, c_2, \dots$  na sljedeći način. Pseudo-slučajan niz znakova  $s_0, s_1, s_2, \dots$  generira se automatom čiji je ulaz tajni ključ. Znak u nizu  $s_i$  ovisi o tajnom ključu i o prethodnom znaku  $s_{i-1}$  dok znak  $s_0$  ovisi o tajnom ključu i nekom javno definiranom parametru. Znak šifrata  $c_j$  izračunava se kombiniranjem znaka poruke  $m_j$  i znaka  $s_j$  iz pseudo-slučajnog niza.

Šifriranje tokova podataka najčešće se koristi u komunikaciji s puno smetnji kako bi se šifrat mogao prenositi u što manjim dijelovima tako da gubitak manjeg dijela podataka ne uzrokuje gubitak cijelog bloka podataka, kao što je to slučaj kod šifriranja blokova. Još jedna važna primjena je u sustavima koji zahtijevaju veliku brzinu prijenosa podataka jer se svaki znak šifrata može dešifrirati nezavisno čim dođe u sustav [1, str. 1265].

Algoritmi za šifriranje tokova podataka stvaraju se za specifične namjene, a najčešće korišteni algoritmi su RC4 za osiguravanje tokova pri prijenosu podataka u računalnim mrežama i A5/1 za osiguravanje podataka u mobilnim GSM mrežama.

### 2.3.2 Šifriranje blokova podataka

Algoritam za šifriranje blokova podataka za dani ključ  $K$  definira šifriranje kao izračunavanje  $n$  bita šifrata iz  $n$  bita poruke, te dešifriranje kao izračunavanje  $n$  bita poruke iz  $n$  bita šifrata. Algoritam za šifriranje blokova trebao bi zadovoljavati dva osnovna svojstva: zbrku (engl. *confusion*) i raspršenost (engl. *diffusion*). Zbrka kod šifriranja predstavlja minimalnu statističku vezu između poruke i šifrata dok raspršenost uvjetuje da bi svaki bit poruke trebao utjecati na što veći dio šifrata. [1, str. 152]

Šifriranje blokova podataka koristi se za osiguravanje tajnosti u komunikaciji, ali isto tako i za sigurnu pohranu podataka u sklopu računalnih sustava. Najzastupljeniji algoritmi za šifriranje blokova podataka su DES (engl. *Data Encryption Standard*) [1, str. 295] i AES (engl. *Advanced Encryption Standard*) [1, str. 1046] [2]. Algoritam DES predstavlja stari standard koji se koristi u sustavima koji su dizajnirani u prošlosti. Algoritam AES je brži i sigurniji nasljednik algoritma DES. Osnovne značajke tih algoritama su sljedeće:

- DES šifrira blokove veličine 64 bita s ključem duljine 56 bita.
- AES šifrira blokove veličine 128-bitna s ključevima duljine 128, 192 ili 256 bita.

Što je veličina ključa veća algoritam je u pravilu sigurniji jer je prostor u kojem se generiraju ključevi veći i teži za pretražiti od strane napadača. Veličina ključa izravno utječe i na brzinu šifriranja, odnosno dešifriranja. Što je ključ veći to je brzina manja.

Algoritmi za šifriranje blokova predstavljaju prilagodljivi primitiv za ostvarivanje raznovrsnih namjena, osim šifriranja podataka, poput autentificiranog šifriranja (engl. *authenticated encryption*), MAC algoritama i funkcija kriptografskog sažetka. Algoritmi šifriranja blokova

mogu se koristiti u različitim načinima rada kako bi mogli biti zadovoljeni različiti sigurnosni zahtjevi.

### Načini šifriranja blokova

Vrlo važan parametar kod šifriranja blokova je način primjene algoritma šifriranja blokova. Način rada algoritama za šifriranje blokova određuje na koji će se način šifrirani blokovi spajati u konačni šifrat. Odabir načina rada izravno utječe na sigurnost i brzinu šifriranja blokova [1, str. 790]. Najznačajniji i najčešće korišteni načini rada su CBC i CTR. Opis načina rada CBC i CTR prikazan je na primjeru algoritma AES koji je trenutni standard za simetrično šifriranje blokova podataka. Dok CBC način rada omogućuje samo šifriranje blokova podataka, CTR i drugi srodni načini rada mogu se koristiti za šifriranje tokova podataka zbog toga što iz tajnog ključa generiraju pseudo-slučajan niz znakova koji se mogu koristiti za šifriranje tokova podataka.

CBC (engl. *Cipher Block Chaining*) koristi ulančavanje blokova na način da se šifrat prošlog bloka bit po bit zbraja po modulu 2 s trenutnom porukom prije šifriranja te poruke. Početni dio poruke zbraja se s promjenjivom početnom vrijednosti koja se označava s IV (engl. *Initialization Vector*). Dodavanje operacije zbrajanja poruke u aritmetici modulo 2 omogućuje veću količinu raspršenosti podataka u odnosu na najjednostavniji ECB (engl. *Electronic CodeBook*) način rada. Blokovi postaju međusobno ovisni ali samo u okviru definiranog prozora. Prozor predstavlja niz određenog broja blokova koji izravno ili neizravno utječe jedni na druge. U CBC načinu rada odnos između poruke  $p_i$ , ključa  $K$  i šifrata  $c_i$  definiran je sljedećim izrazima:

$$c_i = E_K(p_i \oplus c_{i-1}); \quad p_i = D_K(c_i \oplus c_{i-1})$$

CTR (engl. *CounTeR mode*) način rada omogućuje potpunu nezavisnost između blokova, ali se za svaki blok koji se šifrira koristi dodatan podatak čija je svrha povećanje raspršenosti prilikom kodiranja. Kako su svi blokovi međusobno neovisni mogu se paralelno obrađivati što olakšava brzo izračunavanje šifrata iz poruke i obrnuto. Koristi se dodatni podatak stvoren iz jedinstvene jednokratne vrijednosti (engl. *nonce*), početne vrijednosti IV i brojača koji kreće od broja jedan. Za svaki blok vrijednost tog podatka uvećava se za jedan [3]. Izračun šifrata  $c_i$  iz poruke  $p_i$  uz ključ  $K$  i dodatnu vrijednost CTR bloka  $CTRBLK$  definiran je sa sljedećim izrazima:

$$CTRBLK = nonce || IV || i$$

$$c_i = p_i \oplus E_K(CTRBLK)$$

$$p_i = c_i \oplus E_K(CTRBLK)$$

gdje  $||$  označava operaciju konkatenacije nizova okteta.

CBC način rada se smatra standardom, ali CTR ostvaruje veću brzinu i sigurnost uz redovito

osvježavanje i izbjegavanje ponovnog korištenja istih dodatnih podataka.

### 2.3.3 Autentificirano šifriranje

Autentificirano šifriranje (engl. *Authenticated Encryption* - AE) omogućava da se tijekom izračuna šifrata izračuna i autentifikacijski podatak (MAC) koji će potvrditi da šifrat nije mijenjan prilikom prijenosa podataka. Kao osnova za AE koristi se algoritam za simetrično šifriranje blokova. Šifriranje se odradjuje s jednim od prethodno navedenih načina rada (CBC, CTR), ali postoje različiti pristupi izračuna autentifikacijskih podataka. Autentifikacijski podaci se, u pravilu, izračunavaju uz pomoć funkcije kriptografskog sažetka i danog ključa (MAC, odnosno HMAC). Za svaku operaciju potrebno je koristiti različite ključeve  $K_1$  za autentifikaciju i  $K_2$  za šifriranje. Postoje sljedeći načini izračunavanja AE podataka:

- Autentificiraj pa šifriraj (engl. *Mac-then-Encrypt* - MtE) - iz izvorne poruke  $M$  prvo se generiraju podaci za autentifikaciju pa se zatim šifrira poruku zajedno s autentifikacijskim podacima.

$$MtE = E_{K_2}(M, MAC_{K_1}(M))$$

- Šifriraj pa autentificiraj (engl. *Encrypt-then-Mac* - EtM) - prvo se izvornu poruku šifrira pa se podaci za autentifikaciju stvaraju iz šifrata i šalju zajedno.

$$C = E_{K_2}(M); EtM = C, MAC_{K_1}(C)$$

- Šifriraj i autentificiraj (engl. *Encrypt-and-Mac*) - odvojeno se šifrira poruka i izračunavaju autentifikacijski podaci iz poruke i šalju zajedno.

$$E\&M = E_{K_2}(M), MAC_{K_1}(M)$$

Trenutno se u praksi koriste AE programska ostvarenja čija je osnova algoritam šifriranja blokova AES, AES-CCM [4] i AES-GCM [5]. AES-CCM koristi način rada CBC i izračunavanje MtE, dok AES-GCM koristi način rada CTR i izračunavanje EtM. Oboje su se pokazali dovoljno sigurnim za primjenu, ali zbog pronađenih ranjivosti izračunavanja EtM savjetuje se koristiti MtE za sigurnu komunikaciju [6].

AE se smatra standardom za sigurno slanje podataka jer osigurava tri sigurnosna zahtjeva odjednom: tajnost, integritet i autentifikaciju, što predstavlja ključ sigurne komunikacije, jer nije dovoljno samo tajno razmjenjivati podatke, već i utvrditi da podaci nisu mijenjani tijekom prijenosa [1, str. 54].

## 2.4 Asimetrični kriptosustav

Asimetrični kriptosustav koristi različite ključeve za šifriranje i dešifriranje podataka. Ako se podatak šifrira s jednim ključem može se dešifrirati samo s odgovarajućim drugim ključem i obratno. U ovom sustavu moguće je jedan ključ učiniti javnim bez ugrožavanja tajnosti drugog ključa [1, str. 49].

Asimetrično šifriranje se općenito može definirati sljedećim izrazima, gdje  $E$  označava šifriranje,  $D$  dešifriranje,  $PK$  javni ključ,  $SK$  tajni ključ,  $M$  poruku, a  $C_1$  i  $C_2$  šifrate:

$$E_{PK}(M) = C_1; D_{SK}(C_1) = M$$

$$E_{SK}(M) = C_2; D_{PK}(C_2) = M$$

U asimetričnom kriptosustavu svaki tajni ključ ima odgovarajući javni ključ. Asimetrični kriptosustavi mogu se primjenjivati za razne namjene poput digitalnog potpisa, digitalne omotnice i dogovaranja zajedničkog tajnog ključa između komunicirajućih strana. U ovom sustavu također vrijedi da je sigurnost veća što je veća duljina ključa. Za postizanje iste razine sigurnosti, u asimetričnom kriptosustavu se koriste ključevi veće duljine u odnosu na ključeve korištene u simetričnom kriptosustavu.

### 2.4.1 Digitalni potpis

U slučaju da se podatak šifrira tajnim ključem, svi koji su u posjedu odgovarajućeg javnog ključa mogu doći do tih podataka. Navedeni postupak služi za osiguravanje neporecivosti, autentifikacije i integriteta poruke te predstavlja digitalni potpis. Neporecivost je moguće ostvariti jer svaka strana u komunikaciji posjeduje tajni ključ koji mora biti poznat samo toj strani i nikome drugome.

Kako se digitalnim potpisom osigurava integritet poruke nije potrebno šifrirati cijelokupni sadržaj poruke, već je dovoljno šifrirati samo njezin sažetak. Digitalni potpis stvara se korištenjem kriptografskog sažetka, čiji se rezultat šifrira tajnim ključem pošiljatelja. Stvaranje digitalnog potpisa poruke  $M$  korištenjem funkcije sažetka  $h$  korištenjem tajnog ključa  $SK$  opisano je sljedećim izrazom:

$$\text{digitalni\_potpis} = E_{SK}(h(M))$$

Digitalni potpis se uvijek šalje uz odgovarajuću poruku, kako bi se mogao provjeriti integritet i autentičnost poruke. Po primitku poruke primatelj uzima odgovarajući javni ključ  $PK$  kako bi dešifrirao primljeni izvorni sažetak poruke  $S_1$  i usporedio taj sažetak s izračunatim sažetkom  $S_2$ :

$$S_1 = E_{PK}(\text{digitalni\_potpis}); S_2 = h(M)$$

U slučaju da su sažeci  $S_1$  i  $S_2$  jednaki znamo da je poruku potpisao vlasnik tajnog ključa koji odgovara korištenom javnom ključu i da poruka nije mijenjana tijekom prijenosa.

Za digitalni potpis najzastupljeniji algoritmi su DSA(engl. *Digital Signature Algorithm*), RSA (Rivest Shamir Alderman) i ECDSA (engl. *Elliptic Curve DSA*). Algoritam DSA [7] koristi ElGamalov način stvaranja digitalnog potpisa [1, str. 395] koji se temelji na problemu diskretnog logaritma [1, str. 352]. U sklopu standarda definirane su tri veličine ključa, 1024, 2048 i 3072 bita [7].

Algoritam RSA temelji se na problemu faktorizacije velikih prostih brojeva i može koristiti ključeve različitih veličina. Algoritmi DSA i RSA pružaju usporedivu razinu sigurnosti za istu duljinu tajnog ključa. RSA koristi ključeve duljine 1024, 2048 i 4096 bitova. Razina sigurnosti raste s veličinom ključa, ali brzina šifriranja/dešifriranja pada.

## 2.4.2 Digitalna omotnica

Ako se sadržaj poruke šifrira javnim ključem do izvorne poruke može doći samo strana koja posjeduje odgovarajući tajni ključ. Taj postupak služi za ostvarivanje tajnosti u komunikaciji i predstavlja digitalnu omotnicu.

Kako je postupak asimetričnog šifriranja značajno sporiji od postupka simetričnog šifriranja ta se dva postupka zajednički primjenjuju u hibridnom načinu šifriranja. U hibridnom načinu šifriranja poruka se šifrira simetričnim ključem  $K$ , a taj simetrični ključ javnim ključem  $PK$  primatelja podataka kako bi samo on mogao doći do ključa, odnosno sadržaja šifrirane poruke  $M$ :

$$DO_1 = E_K(M); DO_2 = E_{PK}(K)$$

$$\text{digitalna\_omotnica} = DO_1, DO_2$$

Za otvaranje digitalne omotnice, odnosno dešifriranje izvorne poruke primatelju je dovoljan tajni ključ  $SK$  koji odgovara javnom ključu  $PK$  s kojim je stvorena omotnica:

$$K = D_{SK}(DO_2); M = D_K(DO_1)$$

Za stvaranje digitalne omotnice koristi se algoritam RSA koji je opisan u prethodnom poglavljju.

## 2.4.3 Dogovaranje zajedničke tajne

Ako se komunicirajuće strane prethodno ne poznaju te ne posjeduju tajne i javne ključeve za zaštitu komunikacije, one mogu dogovoriti zajedničku tajnu koja će se koristiti za zaštitu komunikacije. Protokoli za dogovaranje zajedničke tajne koriste tajnu vrijednost i odgovarajuću javnu vrijednost za dogovor, te predstavljaju poseban slučaj asimetričnog kriptosustava.

Većina algoritama za dogovaranje zajedničke tajne definirana je u protokolu Diffie-Hellman [8] (DH), a temelji se na problemu diskretnog logaritma. Postupak izračuna pobliže je opisan u nastavku.

Kako bi komunicirajuće strane mogle dogovoriti zajedničku tajnu moraju dijeliti iste početne vrijednosti za izračun. To su:

- prosti broj  $p$  i
- generator  $1 \leq g \leq p - 1$  takav da je  $g^{p-1} \equiv 1 \pmod{p}$ .

Neka su započinjatelj (I) i primatelj (R) dvije strane koje žele dogovoriti zajedničku tajnu. Tijek komunikacije je sljedeći:

1. I generira nasumičnu vrijednost  $1 \leq x \leq p - 2$  i izračunava  $z_x = g^x \pmod{p}$ . Rezultat operacije  $z_x$  šalje strani R.
2. R generira nasumičnu vrijednost  $1 \leq y \leq p - 2$  i izračunava  $z_y = g^y \pmod{p}$ . Rezultat operacije  $z_y$  šalje strani I.
3. Nakon primitka  $z_y$ , I izračunava  $K_I = z_y^x \pmod{p}$ .
4. Nakon primitka  $z_x$ , R izračunava  $K_R = z_x^y \pmod{p}$ .

Vrijednosti  $K_I$  i  $K_R$  su jednake te su time strane I i R uspješno dogovorile zajedničku tajnu. Vrijednosti  $x$  i  $y$  predstavljaju tajne ključeve u razmjeni dok su  $z_x$  i  $z_y$  javni ključevi. Napadač prisluškivanjem komunikacije ima pristup vrijednostima  $p$ ,  $g$ ,  $z_x$  i  $z_y$ . Pronalazak dogovorene tajne je ekvivalentan pronalaženju jednog tajnog ključa. U slučaju dobro odabranih parametara to je težak problem. Ako su ključevi dobro generirani i dovoljno veliki pasivni napadač nije u mogućnosti doći do zajedničke tajne. [1, str. 341]

#### 2.4.4 Algoritmi temeljeni na eliptičnim krivuljama

Nad skupom točaka eliptične krivulje mogu se definirati sve matematičke operacije potrebne za izvođenje kriptografskih operacija poput onih temeljenih na problemu diskretnog algoritma. Algoritmi koji koriste eliptične krivulje koriste manje kriptografske ključeve u odnosu na standardne algoritme. Posljedica toga je brži izračun kriptografskih operacija za jednaku razinu sigurnosti u odnosu na standardne algoritme. Algoritmi temeljeni na eliptičnim krivuljama mogu se koristiti za digitalno potpisivanje i dogovor zajedničke tajne. [1, str. 397]

Algoritam ECDSA [1, str. 407] je inačica algoritma DSA koja umjesto diskretnih algoritama za temelj koristi eliptične krivulje. Prednost eliptičnih krivulja je manja veličina tajnog ključa za postizanje jednakе razine sigurnosti. Algoritam RSA s ključem duljine 2048 daje podjednaku razinu sigurnosti kao algoritam ECDSA s veličinom ključa od 224 bita.

ECDH [1, str. 400] (engl. *Elliptic Curve DH*) je inačica protokola DH koja koristi eliptične krivulje za izračunavanje zajedničke tajne. Glavne prednosti su smanjenje veličine ključeva uz zadržavanje iste razine sigurnosti te jednostavnije i brže generiranje tajnih ključeva. Iako algoritmi temeljeni na eliptičnim krivuljama pružaju mnoge prednosti u vidu manje veličine

ključeva i bržeg izračuna, potrebno je odabrat pouzdanu eliptičnu krivulju za zadovoljavajuću razinu sigurnosti [9].

## 2.5 Autentificirano dogovaranje zajedničke tajne

Protokoli za dogovor zajedničke tajne ranjivi su na napad čovjek u sredini (engl. *Man-in-the-Middle attack* - MITM). Ako se napadač nađe između dvije komunicirajuće strane on može neovisno dogovoriti zajedničku tajnu sa svakom stranom i doći do svih podataka koje te dvije strane razmjenjuju. Tijek napada na protokol Diffie-Hellman (DH) je sljedeći:

1. Započinjatelj (I) generira vrijednost  $x$ , izračunava  $z_x = g^x \text{mod } p$  i šalje primatelju (R).
2. Napadač zaprima vrijednost  $z_x$ . Potom generira svoju vrijednost  $n$  i šalje započinjatelju vrijednost  $z_n$ . Započinjatelj I i napadač na kraju imaju dogovoren ključ  $K_I = z_x^n \text{mod } p$ .
3. S druge strane napadač započinje dogovor ključa sa stranom R s vrijednostima  $n$  i  $z_n$ . Primatelj R i napadač na kraju imaju dogovoren ključ  $K_R = z_y^n \text{mod } p$ .

Kao rezultat navedenog napada napadač ima dogovoren ključ  $K_I$  sa započinjateljem i ključ  $K_R$  s primateljem i može primati šifrirani promet s jedne strane, dešifrirati ga i ponovo šifrirati kako bi ga poslao drugoj strani.

Za zaštitu protokola DH od napada čovjek u sredini stvoreni su novi protokoli koji štite DH razmjenu kako bi onemogućili napadača da utječe na uspostavu zajedničke tajne. Protokol STS (engl. *Station-to-Station*) je prvi protokol koji sprječava napad čovjek u sredini korištenjem autentifikacije. Autentifikacija se postiže korištenjem asimetričnog algoritma za digitalno potpisivanje poruka u komunikaciji. Za digitalni potpis poruka potrebno je razmijeniti certifikate s kojima će se potpisivati. Certifikati se mogu razmijeniti u posebnoj inačici protokola STS koja se zove potpuni STS (engl. *Full STS* [1, str. 1256]).

Razmjena poruka u potpunoj inačici protokola STS definirana je na sljedeći način:

- Započinjatelj I generira tajnu vrijednost  $x$ , izračunava  $z_x = g^x \text{mod } p$  i šalje primatelju R.
- Primatelj R generira tajnu vrijednost  $y$  i izračunava  $z_y = g^y \text{mod } p$ . Uz to izračunava i ključ  $K = z_x^y \text{mod } p$  i s njim šifrira digitalno potpisane podatke  $z_y$  i  $z_x$ . Konačna poruka koja se šalje započinjatelju koja uključuje certifikat  $Cert_R$  je sljedeća:  $z_y, Cert_R, E_K(S_R(z_y, z_x))$
- Započinjatelj I prima poruku i izračunava ključ  $K$  s pomoću kojeg dešifrira  $S_R(z_y, z_x)$  i provjerava digitalni potpis korištenjem certifikata  $Cert_R$ . Ako je provjera uspješna započinjatelj odgovara sa sličnom porukom gdje je promijenjen redoslijed podataka digitalnom potpisu:  $Cert_I, E_K(S_I(z_x, z_y))$ .

Uvođenjem digitalnog potpisa, odnosno korištenjem javnih ključeva, osigurano je da napadač nije u mogućnosti izravno ugroziti dogovor i utjecati na dogovorenu zajedničku tajnu. Pritom postoje druga dva sigurnosna problema: mogućnost vezivanja krivog identiteta za dogovorenu zajedničku tajnu (engl. *identity misbinding attack*) i problem korištenja dogovorene

zajedničke tajne u sklopu dogovora što ne zadovoljava zahtjev korištenja odvojenih ključeva za različite kriptografske operacije (engl. *key separation*) [10].

### 2.5.1 SIGMA

Protokol SIGMA (engl. *SIGn and MAC*) je protokol za autentificirano dogovaranje zajedničke tajne koji je zasnovan na korištenju digitalnog potpisa i MAC algoritma za zaštitu dogovora. Protokol SIGMA je formalno verificiran i predstavlja rješenje koje omogućuje siguran dogovor zajedničke tajne [10].

Dogovaranje zajedničke tajne u protokolu SIGMA odvija se na sljedeći način:

1. Započinjatelj (I) generira nasumičnu vrijednost  $x$  i izračunava  $z_x = g^x \text{mod } p$  i šalje prema primatelju (R).
2. Primatelj (R) generira nasumičnu vrijednost  $y$  i izračunava  $z_y = g^y \text{mod } p$ . Potom izračunava tajni ključ  $K_R$  pomoću KDF (engl. *key derivation function*, pojam je objašnjen u sljedećem poglavlju) funkcije za izračunavanje ključeva iz zajedničke tajne  $SS_R = z_x^y$ ,  $K_R = KDF(SS_R)$ . Zatim šalje odgovor započinjatelju koji sadrži  $z_y$ , javni ključ  $PK_R$ , digitalni potpis  $S_R = E_{PK_R}(z_x, z_y)$ , i MAC od javnog ključa  $PK_R$  zaštićen s tajnim ključem  $K_R$ ,  $M_R = MAC_{K_R}(PK_R)$ .
3. Započinjatelj po primitku poruke izračunava ključ  $K_I$  na način analogan primatelju (iz vrijednosti  $z_y$  i  $x$ ). Zatim provjerava dobiveni digitalni potpis  $S_R$  i MAC  $M_R$ . Ako provjera uspije primatelju šalje poruku sa svojim javnim ključem  $PK_I$ , digitalnim potpisom  $S_I = E_{PK_I}(z_y, z_x)$  i MAC od javnog ključa  $PK_R$  zaštićen s tajnim ključem  $K_I$ ,  $M_I = MAC_{K_I}(PK_I)$ .
4. Primatelj po primitku provjerava valjanost digitalnog potpisa  $S_I$  i MAC-a  $M_I$ . Ako su vrijednosti valjane, uspješno je dogovorena zajednička tajna.

Ključna prednost protokola SIGMA je zaštita identiteta koji se razmjenjuje s pomoću MAC funkcije uz ključ dobiven iz Diffie-Hellman izračuna. Na taj se načini izravno vezuje identitet (javni ključ) dogovaratelja uz dogovorenou zajedničku tajnu. To izravno onemogućava napad čovjek u sredini jer treća strana nije u mogućnosti utjecati na dogovorenou zajedničku tajnu.

## 2.6 Pseudo nasumične funkcije

Pseudo nasumična funkcija (PRF, engl. *pseudo random function*) je deterministička funkcija koja za određeni ključ i ulaz daje rezultat koji se ne može razlikovati od nasumične funkcije ulaza. [1, str. 994]. Idealna PRF ima za cilj postizanje neprepoznatljivosti između rezultata funkcije i nasumičnih podataka. Primjena pseudo nasumičnih funkcija vrlo je široka u kriptografiji. Koriste se za dobivanje sigurnih ključeva iz zajedničkih tajni, kao zamjena za korištenje

generatora slučajnih brojeva, za dobivanje niza ključeva iz jednog izvornog ključa te za izračunavanje funkcija MAC za provjeru integriteta podataka. Pseudo nasumične funkcije se najčešće izvode korištenjem funkcija kriptografskog sažetka.

### 2.6.1 Funkcije izračunavanja ključeva

Nakon dogovora jedne zajedničke tajne, komunicirajuće strane žele koristiti tu tajnu za zaštitu komunikacije. Dogovorena tajna može imati različite duljine pa nastaje problem kako na siguran način dobiti ključ određene duljine za kriptografske algoritme koji će se koristiti. Uz to, da bi se ostvarili svi potrebni sigurnosni zahtjevi potrebno je više različitih vrsta kriptografskih algoritama. Za različite vrste algoritama nije preporučljivo koristiti iste tajne, već je potrebno korištenjem nekog algoritma izračunati različite ključeve za svaki kriptografski algoritam. Ta dva zahtjeva se ostvaruju korištenjem funkcije za izračun ključeva (engl. *Key Derivation Functions* - KDF). Te se funkcije koriste i u računalnim sustavim za pohranjivanje lozinki korisnika, kako bi se mogućim napadačima otežalo razotkrivanje lozinki.

KDF se izvodi korištenjem pseudo-nasumičnih funkcija koje se izvode određenim brojem ponavljanja funkcija kriptografskog sažetka nad tajnim podacima. Funkcija kriptografskog sažetka se u pravilu koristi u HMAC načinu rada za potrebe KDF-a. Broj ponavljanja određuje se ovisno o primjeni: veći je za pohranu lozinki u odnosu na izračunavanje tajnih ključeva potrebnih za komunikaciju.

Standardizirani i najčešće korišteni algoritmi za KDF su sljedeći:

- PBKDF i PBKDF2 (engl. *Password-Based KDF*) [11] koriste se za pohranu autentifikacijskih podataka iz korisničkih lozinki. Izvodi se veliki broj iteracija kako bi se napadaču otežalo pogadanje početne lozinke iz tih podataka. Preporučeni broj iteracija je barem 1000.
- HKDF (engl. *HMAC-based Extract-and-Expand KDF*) [12] namijenjen je za izračun tajnih ključeva iz dogovorene zajedničke tajne i ne koristi dodatne iteracije. Svaki oktet neovisno se generira kako bi se povećala sigurnost izračunatog ključa. HKDF omogućuje definiranje dodatnih varijabli kako bi se povećala kvaliteta ključeva. Veći broj iteracija u ovom slučaju nije poželjan jer se izračunavanje ključeva radi za vrijeme komunikacije. Preveliki broj iteracija bi uzrokovao usporavanje komunikacije [13].

## Poglavlje 3

# Kriptografski prilagodljiva komunikacija

Pojam kriptografski prilagodljive komunikacije značajan je za sve sustave koji aktivno koriste kriptografske protokole i temelj je svih današnjih protokola za sigurnu komunikaciju. Kriptografski prilagodljiva komunikacija predstavlja mogućnost promjene parametara s kojima se odvija komunikacija bez mijenjanja sustava koji koristi kriptografiju za ostvarivanje sigurnosnih zahtjeva.

Kriptografski prilagodljiva komunikacija mora podržavati mogućnost promjene sljedeća dva parametra:

- Kriptografskih algoritama s kojima se štiti komunikacija. Najčešće to uključuje više vrsta algoritama, npr. kriptografski sažetak i algoritam za digitalni potpis, ili skupove kriptografskih algoritama (engl. *cryptographic suite*) poput protokola SSL [14] / TLS [15].
- Ključeva koji će se koristiti s promijenjenim kriptografskim algoritmima. To se prije svega odnosi na tajne ključeve koji se koriste za simetrično šifriranje i izračunavanje MAC algoritama, dok se javni ključevi rjeđe mijenjaju. Različiti kriptografski algoritmi koriste ključeve različitih duljina i potrebno je za svaku novu komunikaciju promijeniti kriptografski ključ koji se koristi (engl. *key separation* [16]).

Glavne prednosti kriptografski prilagodljive komunikacije su sljedeće:

- Omogućuje korištenje različitih ostvarenja kriptografskih algoritama koje ne moraju biti sastavni dio postavljenog sustava.
- Mogućnost brze promjene kriptografskih algoritama koji imaju određene probleme u samom algoritmu ili korištenom programskom ostvarenju.
- Uvođenje novih i sigurnijih kriptografskih algoritama u sustav na jednostavan način koji ne utječe na trenutnu konfiguraciju sustava.
- Prilagođavanje sustava novim kriptografskim zahtjevima koje zahtjeva napredak metoda napada na kriptografske algoritme ili proces certifikacije sustava.
- Veća razina prilagodljivosti koja omogućuje komunikaciju s uređajima različitih računalnih sposobnosti koji podržavaju razne kriptografske algoritme.

U nastavku ovog poglavlja dan je pregled postojećih protokola koji koriste kriptografski prilagodljivu komunikaciju kako bi ostvarili potrebne sigurnosne zahtjeve. Uz to, istaknute su njihove glavne prednosti i nedostaci.

### 3.1 Pregled kriptografski prilagodljivih protokola

Kriptografija je najviše zastupljena u području zaštite mrežne komunikacije korištenjem protokola koji omogućavaju sigurne komunikacijske kanale (engl. *secure channel protocols*). Usporedba predloženog protokola ACAP (engl. *Agile Cryptographic Agreement Protocol*) i standar-diziranih protokola prikazana je u tablici 3.1. Početna ideja iza protokola ACAP na protokolu SEND [17] (engl. *Secure Neighbor Discovery*) prikazana je u [18]. Neovisnost o sloju i daljnja razrada principa dogovora opisana je u radu [19]. Posljednje proširenje protokola koje uključuje formate svih poruka i formalno verificirani model prezentirano je u radu [20].

Prvi stupac sadrži imena protokola, a drugi stupac sadrži podatak o parametrima koji se dogovaraju u protokolu. Dogovor kriptografskih algoritama i ključeva bi se u pravilu trebao ostvariti u sklopu istog dogovora jer se ključevi ne mogu koristiti bez dogovorenih algoritama. Samo protokol JFK [21] (engl. *Just Fast Keying*) služi za dogovor ključeva jer se dogovor algoritama odvija u sklopu protokola IPsec [22] koji ga koristi. U trećem stupcu nalaze se podaci o sloju na kojem se protokoli mogu koristiti. Protokol ACAP, koji je opisan u doktorskoj disertaciji, jedini je protokol koji se može koristiti neovisno o komunikacijskom sloju.

**Tablica 3.1:** Pregled kriptografski prilagodljivih protokola

Ime protokola	Dogovoreni parametri	Komunikacijski sloj	Namjena	Složenost	Formalna verifikacija
ACAP	ključ i algoritmi	svi slojevi	dogovor ključa i algoritama	niska	da
JFK <sup>[21]</sup>	ključ	vezan uz IPsec	dogovor ključeva	niska	da
IPsec <sup>[22]</sup>	ključ i algoritmi	sloj IP	zaštita prometa, VPN	visoka	djelomična
SSL <sup>[14]</sup> /TLS <sup>[15]</sup> DTLS <sup>[23]</sup>	ključ i algoritmi	transportni sloj (TCP, UDP)	sigurni komunikacijski kanali	visoka	djelomična
MinimaLT <sup>[24]</sup>	ključ i algoritmi	UDP	sigurni komunikacijski kanali	srednja	djelomična
SSH <sup>[25]</sup>	ključ i algoritmi	aplikacijski sloj	sigurni komunikacijski kanali i ljudska ( <i>shell</i> )	srednja	djelomična
Tcpcrypt <sup>[26]</sup>	ključ i algoritmi	TCP	sigurni komunikacijski kanali putem TCP-a	srednja	ne
QUIC <sup>[27]</sup>	ključ i algoritmi	UDP	siguran prijenos HTTP prometa	srednja	ne

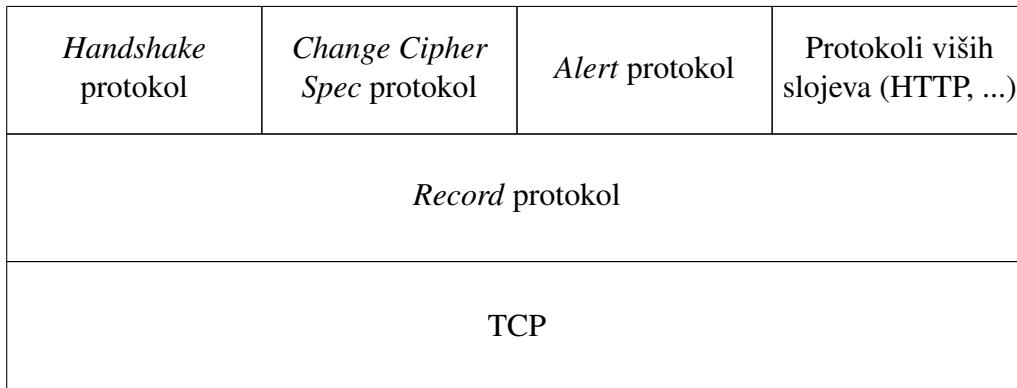
Četvrti stupac opisuje namjenu protokola, a peti stupac sadrži opis složenosti protokola. Složenost protokola izravno je ovisna o njegovoj namjeni. Namjena utječe na broj komponenti protokola, a broj komponenti određuje složenost protokola. Na primjer, SSL [14] / TLS [15] sastoje se od 4 komponente, a IPsec i SSH [25] od 3 komponente. S druge strane QUIC [27], Tcpcrypt [26] i MinimaLT [24] sastoje se od dvije komponente, jedna za razmjenu ključeva i druga za siguran prijenos podataka. Složenost protokola ključan je aspekt protokola jer otežava programsko ostvarenje protokola i izglede za uspješnim formalnim modeliranjem i verifikacijom. Dostupnost provedene formalne verifikacije navedena je u posljednjem stupcu.

### 3.1.1 Protokoli SSL/(D)TLS

Protokoli SSL [14] i TLS [15] omogućavaju sigurnu komunikaciju putem pouzdane veze koja koristi protokol TCP (engl. *Transmission Control Protocol*) za prijenos podataka. Osim protokola i komunikacijske logike u SSL/TLS rješenjima nalaze se i svi kriptografski algoritmi koji će se koristiti za zaštitu prenesenih podataka. To otežava promjenu programskog ostvarenja algoritma u sustavu. Oba protokola dizajnirana su na kraju prošlog stoljeća i njihovi su temelji postavljeni u vrijeme kada je znanje o dizajnu sigurnih protokola i kriptografskih algoritama bilo značajno manje od trenutka pisanja disertacije. SSL/TLS arhitektura je složena i sastoje se od više protokola:

- Protokol za dogovor početnih parametara komunikacije (engl. *Handshake protocol*),
- Protokol za promjenu parametara komunikacije (engl. *Change Cipher Spec protocol*),
- Protokol za upravljanje podacima (engl. *Record protocol*) i
- Protokol za dojavu grešaka u komunikaciji (engl. *Alert protocol*).

Cjelokupna arhitektura SSL/TLS protokola prikazana je na slici 3.1.



Slika 3.1: Arhitektura protokola SSL/TLS

Zbog složenosti arhitekture protokola i velikog broja komponenti potrebnih za ostvarivanje sigurne komunikacije počeli su se pronaći problemi u dizajnu protokola kao i u programskim ostvarenjima protokola SSL/TLS. U [28] prikazana je ranjivost CBC načina šifriranja dok [29]

opisuje problem koji se javlja zbog krivog načina korištenja kompresije prilikom šifriranja. Uz to otkrivene su ranjivosti u protokolu za upravljanje podacima [30] i korištenja šifre RC4 za generiranje nasumičnih vrijednosti [31]. Dodatno su se počele javljati i ranjivosti programskih ostvarenja koje su omogućile napad Heartbleed (OpenSSL) [32] i razne druge napade na sustav za upravljanje certifikatima (Apple) [33].

Iako protokol SSL/TLS predstavlja temelj moderne sigurne komunikacije putem Interneta jedan od osnovnih problema leži u kompleksnosti protokola i njegovog dizajna. Problem bi bio puno manje izražen kada bi kriptografski algoritmi bili odvojeni od sustava komunikacije i kada bi dizajn protokola bio pojednostavljen na način da podržava samo sigurne načine dogovora i zaštite komunikacije, koje ne bi uključivale trenutno nesigurne algoritme i načine dogovora (SSLv2, SSLv3).

Protokol DTLS [23] predstavlja prilagodbu protokola TLS za datagramsku komunikaciju putem protokola UDP i DCCP. On dodaje razne mehanizme za prilagodbu sigurne komunikacije datagramskom načinu komuniciranja koji uključuje retransmisiju poruka tijekom dogovaraњa parametara komunikacije i brine se za održavanje redoslijeda poruka tijekom komunikacije.

### 3.1.2 Protokol IPsec

IPsec [22] predstavlja skup kriptografskih mehanizama koji služi za uspostavljanje sigurne komunikacije kroz nesigurnu mrežu. Najčešće se koristi za povezivanje udaljenih računala na siguran način do odredišne mreže preko Interneta. Sastoji se od više različitih protokola i skupa kriptografskih algoritama.

Protokol IPsec se može podijeliti u tri osnovne kategorije:

- protokoli za zaštitu mrežnog prometa, AH (engl. *Authentication Header*) i ESP (engl. *Encapsulating security Payload*),
- sustav za upravljanje ključevima i parametrima za sigurnu komunikaciju što uključuje upravljanje sigurnosnim asocijacijama (engl. *Security Association*, SA) i
- sustav za određivanje koji promet treba biti zaštićen i s kojim parametrima što je izravno definirano s trenutno uspostavljenim SA u sustavu.

U standardu IPsec važan je pojam skupa kriptografskih algoritama (engl. *cryptographic suite*) koji predstavlja više različitih vrsta algoritama, poput algoritama za šifriranje, osiguravanje integriteta, Diffie-Hellman parametara i različitih funkcija za generiranje pseudo nasumičnih brojeva (npr. AES za šifriranje, SHA-256 za integritet, ...).

Protokol AH uključen je u IPsec standard iz povjesnih razloga, dok se protokol ESP koristi za šifriranje i autentifikaciju prometa koji se razmjenjuje. Zaštita se može provoditi na dva načina: transportni i tunelirani. Transportni način štiti samo sadržaj paketa bez promjene zaglavљa. Taj se način prijenosa koristi za zaštitu komunikacije između dvije krajnje točke u mreži. Tunelirani način rada štiti cijelo originalno zaglavje tako da cijeli paket enkapsulira

pod novim zaglavljem dok se na odredištu sadržaj paketa dekapsulira i šalje u istom početnom obliku. Ovaj se način prijenosa uspostavlja u dva scenarija:

- između dva čvora koji nisu krajnje točke u komunikaciji i omogućuje zaštićeno prosljedivanje svog prometa iz jednog mrežnog segmenta u drugi; VPN između dva mrežna segmenta i
- između jednog čvora koji je krajnja točka u komunikaciji i želi komunicirati s mrežnim segmentom koji se nalazi u zaštićenoj mreži; VPN veza udaljenog klijenta u zaštićeni mrežni segment.

Cjelokupna zaštita prometa provodi se na temelju dogovorenih sigurnosnih asocijacija, a proces zaštite prometa odvija se u jezgri operacijskog sustava računala koji provodi zaštitu.

Protokol IKE [34] (engl. *Internet Key Exchange*) služi za dogovor parametara za sigurnu komunikaciju i upravlja trenutno aktivnim sigurnosnim asocijacijama (SA). To je samostalan protokol koji omogućuje kriptografski prilagodljivu komunikaciju i dinamičko dogovaranje kriptografskih parametara za zaštitu komunikacije. Bez protokola IKE ne bi bio ostvariv automatski dogovor sigurnosnih asocijacija te bi se sve SA trebale ručno podešavati. Stoga protokol IKE izravno omogućuje široku primjenu i korištenje cjelokupnog standarda IPsec za zaštitu prometa koji se prenosi između dvije podmreže koje su povezane putem Interneta.

Posljednji dio IPsec-a nalazi se u jezgri operacijskog sustava koji se bavi odlučivanjem o tome koji se podaci štite, ovisno o adresama i protokolima koji se koriste za komunikaciju. U jezgri operacijskog sustava uz to se nalaze i kriptografski mehanizmi s pomoću kojih se štiti promet.

Cjelokupni standard IPsec omogućuje ostvarivanje sigurnosnih zahtjeva tajnosti i integriteta cjelokupnog prometa koji se prenosi s pomoću protokola IP. Samim time, to je složen sustav koji se sastoji od većeg broja komponenti koje se protežu od jezgre operacijskog sustava, kako bi se omogućilo rukovanje prometom, do korisničkih aplikacija koje omogućavaju automatizirano podešavanje sigurne komunikacije.

IPsec je složen sustav koji ima važne primjene, ali razina sigurnosti koju pruža nije uvijek potrebna i nije uvijek izvediva uz zadovoljavajuću razinu performansi. Uz to IPsec je čvrsto vezan uz operacijski sustav i promjene u slučaju sigurnosnih problema zahtijevaju potencijalno velike zahvate na cjelokupnom sustavu. Zaštita mrežnog prometa IPsec-om moguća je između računala i mobilnih uređaja nove generacije, ali je teško primjenjiva na uređajima u okolini Interneta stvari koji imaju značajno manje računalne mogućnosti.

### Protokol JFK

Protokol JFK (engl. *Just Fast Keying*) [21] bio je predložen kao alternativa protokolu IKE verzije 1 i postao je standardni dio trenutne verzije protokola IKE, IKE verzije 2. Razlog tomu su ranjivosti koje su otkrivene u protokolu IKE verzije 1 [35]. U protokolu JFK koristi se

SIGMA princip za zaštitu dogovora i razmjene ključeva te služi prije svega za dogovor oko sigurnosnih asocijacija. Uz to dodatno omogućuje zaštitu identiteta jedne od komunicirajućih strana. Zaštita identiteta je važan parametar u komunikaciji, ali se rijetko javlja potreba za tim zahtjevom.

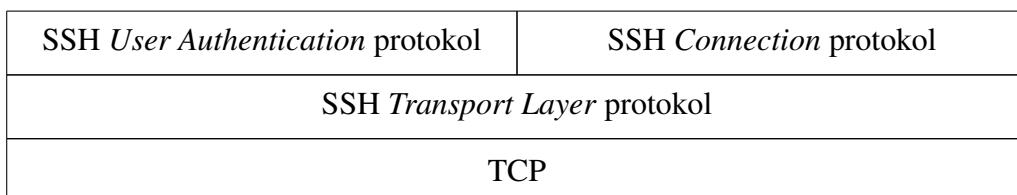
### 3.1.3 Protokol SSH

Protokol SSH [25] je standardni protokol za upravljanjem udaljenim računalima korištenjem komandno-linijskog sučelja za izvođenje naredbi, prenošenje datoteka i sigurno tuneliranje mrežnog prometa između dva računala.

Protokol SSH sastoji se od 3 protokola:

- Protokol za upravljanje prijenosom podataka (engl. *SSH Transport Layer protocol*) koji se brine za osiguravanje cijelovitosti i tajnosti podataka pri prijenosu kroz mrežu [36].
- Protokol za upravljanje vezama (engl. *SSH Connection protocol*) koji je zadužen za multiplexiranje više tokova podataka kroz istu SSH vezu (npr. istodobno udaljeno upravljanje uz prijenos podataka) [37].
- Protokol za autentifikaciju korisnika (engl. *SSH User Authentication protocol*) koji provjerava korisnikove podatke i određuje ima li korisnik pravo pristupa sustavu (računalu) [38].

Protokol SSH se izvodi kroz TCP vezu između dva računala u mreži, a međusobni odnosi sastavnih dijelova protokola SSH prikazani su na slici 3.2.



**Slika 3.2:** Arhitektura protokola SSH

Protokol SSH u sklopu upravljanja prijenosom podataka dogovara kriptografske ključeve i algoritme koji će se koristiti za zaštitu komunikacije. Odluka o kriptografskim algoritmima izvodi se na sljedeći način:

- Klijent i poslužitelj razmijene uređene liste podržanih algoritama.
- Odabire se prvi algoritam koji se nalazi na popisu klijenta, a ujedno je podržan od strane poslužitelja.
- U slučaju da se ne može pronaći zajednički algoritam veza se prekida.

Protokol SSH dogovara način razmjene ključeva, asimetrični algoritam šifriranja, simetrični algoritam šifriranja, algoritam za autentifikaciju poruka (engl. *message authentication algorithm*) i algoritam kriptografskog sažetka.

### 3.1.4 Protokoli nove generacije

Protokol tcpcrypt [26] predstavlja novi pristup zaštiti prometa koji se razmjenjuje putem protokola TCP. Nastao je kao moguće rješenje na probleme koji se pojavljuju prilikom korištenja protokola SSL/TLS. Postupak dogovaranja parametara za sigurnu komunikaciju integriran je u standardnu sinkronizaciju u tri koraka (engl. *three-way handshake*) s dodatkom jedne dodatne poruke. Uspostava sigurne veze može se uspostaviti u 2 vremena povrata (engl. *Round Trip Time* - RTT) dok je kod TLS-a potrebno barem 3 RTT-a. Opterećenje poslužitelja se isto značajno smanjuje jer se postupak asimetričnog šifriranja premješta na klijenta. Stoga, tcpcrypt je efikasnije rješenje, u odnosu na TLS, za zaštitu prometa koji se izmjenjuje protokolom TCP. Smanjena kompleksnost protokola omogućava lakše održavanje i razvijanje protokola te smanjuje mogućnosti grešaka u dizajnu i programskom ostvarenju.

Protokol QUIC [27] je novi pristup sigurnom prijenosu podataka putem protokola HTTP korištenjem protokola UDP kao transportnog protokola. UDP se koristi kako bi se smanjila kašnjenja i opterećenja na poslužiteljima. Trenutno je integriran u preglednik sjedišta weba Chromium kako bi se omogućio lakši razvoj i testiranje. QUIC uvodi nove mehanizme zaštite od lažiranja adresa IP i napada ponavljanjem prometa.

Protokol MinimaLT [24] je jedan od protokola nove generacije sličan QUICu, koji je dizajniran da pruža veću razinu sigurnosti tako da se Diffie-Hellman parametri i javni ključevi dijele s pomoću DNS upita i odgovora [39]. Komunicirajuće strane međusobno se autentificiraju korištenjem javnih ključeva. MinimaLT postiže veću razinu fleksibilnosti zbog prijenosa podataka protokolom UDP umjesto protokolom TCP, ali je i dalje primjenjiv isključivo na slojevima iznad transportnog.

## 3.2 Dogovor zajedničke tajne i kriptografskih algoritama

Dogovor zajedničke tajne u postojećim se protokolima izvodi korištenjem asimetričnih algoritama. Dogovor se može izvoditi korištenjem protokola Diffie-Hellman ili korištenjem asimetričnih algoritama (npr. RSA) za postizanje tajnosti korištenjem principa digitalne omotnice. U oba slučaju razmjenjuju se dijelovi s kojima se izračunava zajednička tajna. Kod korištenja Diffie-Hellman algoritma dijelovi koji se razmjenjuju su javni, a kod korištenja digitalne omotnice dijelovi su tajni.

Protokoli SSL/TLS podržavaju oba načina razmjene ključeva, dok SSH, IPsec, QUIC i MinimaLT koriste Diffie-Hellman za uspostavu početne tajne, a tcpcrypt koristi samo princip digitalne omotnice. IPsec razmjena koristi princip SIGMA u protokolu JFK koji je sličan principima korištenima u protokolu ACAP. Odluka o kriptografskim algoritmima se kod protokola SSL/TLS, SSH, tcpcrypt i QUIC odrađuje na način da komunicirajuće strane razmijene podržane algoritme koji su raspoređeni uzlazno po njihovim jačinama. U pravilu se uzima prvi

algoritam na popisu klijenta koji je podržan od strane poslužitelja.

### 3.2.1 Novi algoritmi dogovora zajedničke tajne

Uz princip SIGMA za autentificirano dogovaranje zajedničke tajne, postoje noviji algoritmi koji koriste implicitnu autentifikaciju bez potrebe izračuna i provjere digitalnih potpisa. Tim algoritmima potrebno je dvije poruke za dogovor zajedničke tajne umjesto tri poruke, ali je njihov dizajn puno osjetljiviji u odnosu na eksplisitno autentificirane protokole [40]. To su algoritmi zasnovani na principu MQV (Menezes–Qu–Vanstone)[41] i OAKE (engl. *Optimal (implicitly) Authenticated (Diffie-Hellman) Key-Exchange*) [40].

Početni MQV princip proširen je u vidu HMQV (engl. *Hashed MQV*) [42] zbog početnih napada na taj princip. Dodatnim istraživanjem pronađene su nove ranjivosti na napad čovjek u sredini koje su riješene uvođenjem principa FHMQV (engl. *Fully Hashed MQV*) [43]. FHMQV predstavlja formalno verificirani princip koji omogućuje siguran dogovor zajedničke tajne [44]. FHMQV se u vrijeme pisanja doktorske disertacije počeo aktivno koristiti u sklopu drugih rješenja i protokola [45] [46]. Dodatno se aktivno istražuju novi MQV principi koji se temelje na eliptičnim krivuljama (ECMQV) ili su posebno prilagođeni uređajima s ograničenim mogućnostima [47]. Kao alternativa MQV principu izведен je OAKE princip koji pruža određene prednosti u vidu smanjivanja zahtjevnosti izračuna zajedničke tajne i zaštite identiteta komunicirajućih strana [40]. Primjena OAKE principa je nedavno prepoznata u sklopu određenih protokola i primjena [48].

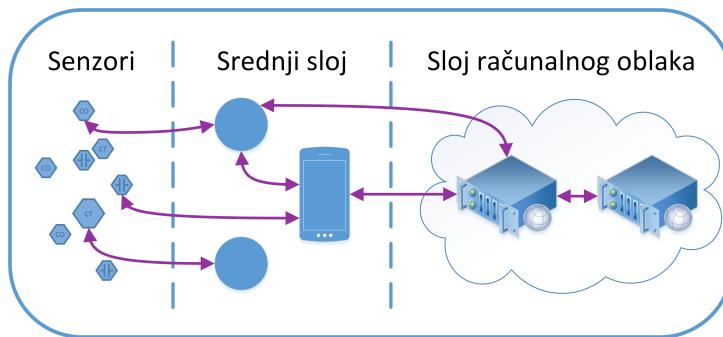
Osnovna prepreka u korištenju novih algoritama za autentificirani dogovor zajedničke tajne je nepostojanje standardnog programskog rješenja navedenih principa u sklopu kriptografskih biblioteka, što onemogućava praktično ostvarenje i mjerjenje utjecaja kriptografskih izračuna u odnosu na mrežnu komunikaciju. Drugi nedostatak povezan je s izostankom javnih ključeva (certifikata) u razmjeni što onemogućuje provjeru identiteta u sustavima koji se oslanjaju na infrastrukturu javnog ključa (engl. *Public Key Infrastructure - PKI*). Kroz buduće istraživanje napretka implicitno autentificiranih Diffie-Hellman protokola za dogovor zajedničke tajne (engl. *Implicitly Authenticated Diffie Hellman Key Exchange - IADHKE*) te pažljivi dizajn i verifikaciju novog rješenja mogao bi se unaprijediti mehanizam za dogovor zajedničke tajne u protokolu ACAP na način koji bi zahtijevao manje resursa i potencijalno manji broj poruka za dogovor svih parametara.

## 3.3 Kriptografska prilagodljivost u okolini Interneta stvari

Svi trenutno programski ostvareni protokoli koji su navedeni u tablici 3.1 imaju kompleksnu arhitekturu koja omogućuje sigurnu komunikaciju korištenjem komunikacijskih kanala. Uspostavljanje i održavanje ovih kanala zahtijeva veliki dio resursa, posebno u području memorije i procesiranja podataka. Osim toga, postoji veliki broj varijabli i konstanti koje moraju biti pravilno konfigurirane i podejmene za različite uslove rada. Ovo je jedan od glavnih razloga zašto su moderni protokoli tako složeni i zahtjevnici.

tava sigurnih komunikacijskih kanala i njihovo održavanje zahtijeva dodatne resurse, a često nisu potrebni u okolini Interneta stvari. Komunikacija u okolini Interneta stvari u pravilu je temeljena na porukama koje nose određenu informaciju (npr. izmjereni podatak iz okoliša, naredbu za upravljanje uređajima ili podatke za ažuriranje uređaja).

Arhitektura Interneta stvari je slojevita i sastoji se od mnoštva različitih uređaja koji se mogu rasporediti u različite skupine ovisno o njihovim trenutnim mogućnostima, poput računalnih sposobnosti, pristupa napajanju i propusnosti mrežne komunikacije. Na slici 3.3 prikazana je arhitektura Interneta stvari [49] i tri osnovna sloja arhitekture. Senzori predstavljaju najviše ograničenu skupinu uređaja, srednji sloj predstavlja djelomično ograničenu skupinu, dok sloj računalnog oblaka ima neograničen pristup resursima. Zbog velikog raspona različitih mogućnosti uređaja u okolini Interneta stvari nužno je ostvariti prilagodljivi algoritam koji će uzimati u obzir trenutne mogućnosti uređaja za odabir kriptografskih algoritama. Trenutni algoritmi dogovora algoritama uzimaju u obzir samo razinu sigurnosti što može negativno utjecati na brzinu komunikacije i potrošnje izvora napajanja.



Slika 3.3: Slojevita arhitektura Interneta stvari

U radu [50] prepoznati su sigurnosni zahtjevi za okolinu Interneta stvari. Prilagodljiva sigurna komunikacija omogućuje rješavanje problema zaštite podataka na različitim slojevima arhitekture Interneta stvari [51]. Uz to prepoznaće se i važnost manje zahtjevnih načina zaštite podataka za uređaje senzorskog sloja [52] kako bi se povećao broj sigurnih mjerjenja tih uređaja.

## Poglavlje 4

# Protokol za sigurno dogovaranje kriptografskih algoritama

U ovom poglavlju opisan je novi protokol za sigurno dogovaranje kriptografskih algoritama neovisan o komunikacijskom sloju, u dalnjem tekstu ACAP (engl. *Agile Cryptographic Agreement Protocol*). Osnovni zahtjevi na dizajn protokola ACAP su sljedeći:

- sigurna razmjena kriptografskih ključeva,
- prilagodljivi dogovor kriptografskih algoritama,
- jednostavna arhitektura koja omogućava formalnu verifikaciju,
- neovisnost o komunikacijskom sloju i uređaju i
- učinkovita komunikacija s malim brojem poruka.

Protokol služi za komunikaciju između dvije strane (klijenta i poslužitelja) i omogućava sigurno dogovaranje svih potrebnih preduvjeta kako bi se ostvarila buduća sigurna komunikacija između te dvije strane gdje je klijent strana koja započinje komunikaciju, a poslužitelj druga strana koja očekuje poruku za početak dogovora. Protokol ACAP omogućuje i dogovor u sklopu paradigmе ravnopravnih čvorova gdje su svi čvorovi istovremeno klijenti i poslužitelji.

Nakon uspješne razmjene poruka obje strane imaju pristup sljedećim podacima:

- Tajnom ključu koji se generira iz zajedničke tajne koja se uspostavlja korištenjem protokola Diffie-Hellman [8] i omogućava uspostavljanje principa PFS (engl. *Perfect Forward Secrecy*) [53]. Princip PFS omogućava tajnost razmjene podataka i u slučaju da se otkrije dugoročna zajednička tajna.
- Javnim ključevima koji se koriste za autentifikaciju komunicirajućih strana.
- Popisu kriptografskih algoritama koji su podržani na obje komunicirajuće strane te koji će se koristiti za ostvarivanje potrebnih sigurnosnih zahtjeva.

Na početku potpoglavlja 4.1 dan je pregled označavanja za sve kriptografske operacije koje se koriste za opis protokola. Potom slijedi opis poruka i razmjene poruka s kojima se ostvaruje komunikacija u protokolu ACAP. Na kraju poglavlja analizira se sigurnost predloženog

protokola i opisuju se sigurnosni mehanizmi koji su integrirani u protokol.

Protokol ACAP temeljen je na principu *sign-and-mac* (SIGMA) koji je opisan u poglavljiju 2.5.1. SIGMA kombinira razmjenu ključa korištenjem Diffie-Hellman algoritma, digitalni potpis i algoritam HMAC za zaštitu dogovora tako da onemogućuje napadača da uspostavi napad čovjek u sredini. Protokol JFK, koji je opisan u potpoglavlju 3.1.2, također koristi princip SIGMA, što objašnjava sličnost prve dvije poruke protokola ACAP s ovim protokolom. Dogovor algoritama koji će se koristiti za osiguravanje komunikacije sličan je onom koji je definiran u transportnom dijelu protokola SSH.

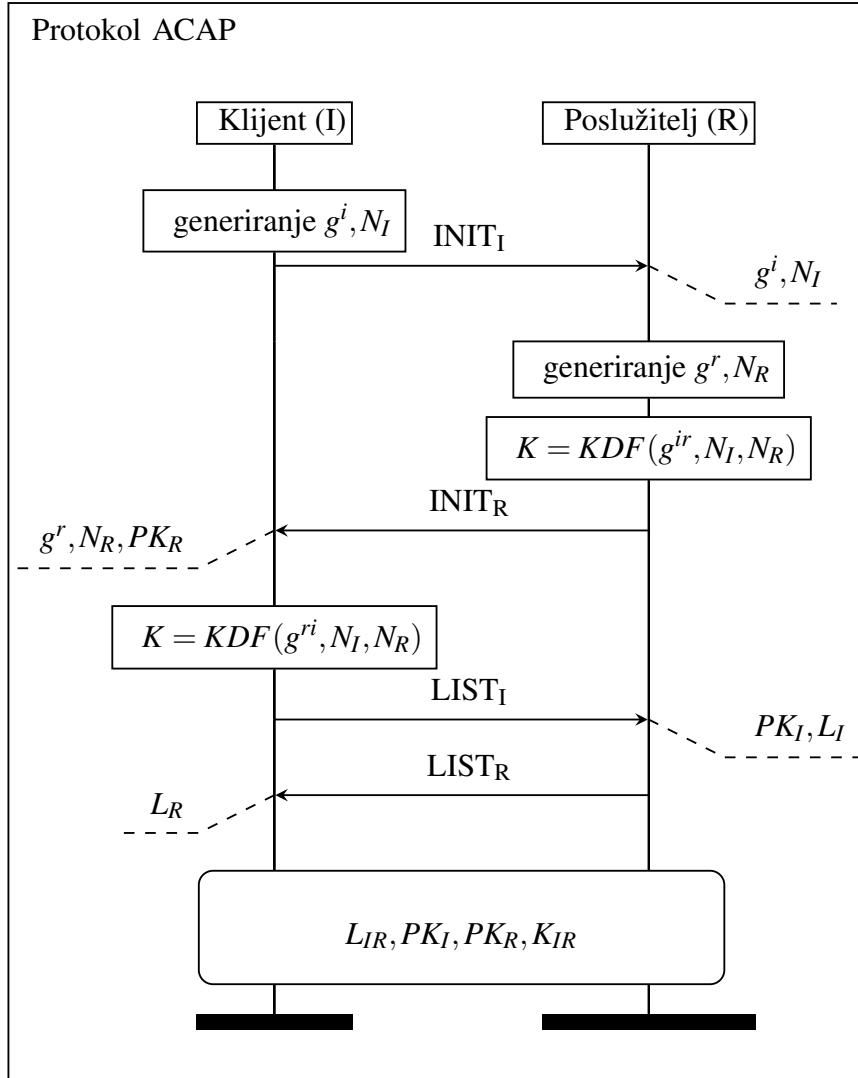
## 4.1 Označavanje kriptografskih operacija

Za opis protokola koristi se standardni način označavanja kriptografskih operacija. Oznaka  $X$  predstavlja jednu stranu u komunikaciji, a oznaka  $x$  predstavlja tajnu komponentu Diffie-Hellman razmjene na odgovarajućoj strani. Klijent se označava oznakom  $I$  (od engl. *initiator*, započinjatelj), a poslužitelj oznakom  $R$  (od engl. *responder*, odgovaratelj):

- $g^x$  javni dio Diffie-Hellman (DH) razmjene ključeva, uz specifikaciju DH grupe koja se želi koristiti za komunikaciju,
- $N_X$  jedinstvena privremena vrijednost (engl. *nonce*) koja se koristi za jedan dogovor,
- $K$  ključ sjednice, rezultat je KDF operacije nad dijeljenom tajnom (dijeljena tajna se računa iz  $(g^x, y)$  ili  $(g^y, x)$ , ovisno o komunikacijskoj strani, gdje su  $x$  i  $y$  tajni dijelovi DH razmjene) i *nonce* vrijednosti obje strane,  
$$K = KDF(g^{xy}, N_I, N_R),$$
- $PK_X$  javni ključ (ili digitalni certifikat),
- $S_X\{\}$  digitalni potpis s tajnim (privatnim) ključem koji se može provjeriti s odgovarajućim javnim ključem  $PK_X$ ,
- $H_K\{\}$  HMAC operacija uz korištenje ključa sjednice  $K$ ,
- $L_X$  popis podržanih kriptografskih algoritama, podijeljen u kategorije,
- $L_{IR}$  dogovoreni kriptografski algoritmi između klijenta ( $I$ ) i poslužitelja ( $R$ ) tijekom ACAP razmjene,
- $K_{IR}$  tajni ključ koji je dogovoren između klijenta ( $I$ ) i poslužitelja ( $R$ ),
- $E_X$  poruka koja sadrži podatke o grešci koja se dogodila tijekom komunikacije (koristi se samo u porukama ABORT).

## 4.2 Poruke i tok komunikacije

Standardni tok komunikacije u protokolu ACAP prikazan je na slici 4.1. U njemu je prikazano računanje i distribucija svih parametara koji su potrebni za sigurnu komunikaciju.



**Slika 4.1:** Grafički prikaz razmjene poruka protokola ACAP

Protokol ACAP sastoji se od četiri osnovne poruke s pomoću kojih se izmjenjuju javni ključevi, dogovara zajednička tajna te kriptografski algoritmi koji će se koristiti za zaštitu daljnje komunikacije. Oznake u zagradama korištene u opisu poruka protokola označuju smjer poruke ( $I \rightarrow R$  označuje poruku koju klijent  $I$  šalje poslužitelju  $R$ ):

$$\text{INIT}_I(I \rightarrow R): \quad g^i, N_I$$

$$\text{INIT}_R(R \rightarrow I): \quad g^r, N_R, PK_R, S_R\{g^r\}, H_K\{PK_R, N_I, N_R\}$$

$$\text{LIST}_I(I \rightarrow R): \quad PK_I, L_I, S_I\{L_I, g^i, g^r\}, H_K\{PK_I, N_R, N_I\}$$

$$\text{LIST}_R(R \rightarrow I): \quad L_R, S_R\{L_R, g^i, g^r, N_I, N_R\}$$

Klijent pokreće komunikaciju slanjem poruke  $\text{INIT}_I$ . Ona sadrži javni dio Diffie-Hellman razmjene klijenta ( $g^i$ ) uz specifikaciju DH grupe i klijentov *nonce*  $N_I$ . Ako poslužitelj podržava predloženu grupu odgovara klijentu s porukom  $\text{INIT}_R$ .

Poruka  $\text{INIT}_R$  sadrži javni dio DH razmjene poslužitelja  $g^r$ , javni ključ  $PK_R$  poslužitelja i *nonce*  $N_R$ . Poruka je zaštićena s pomoću digitalnog potpisa javnog dijela DH razmjene  $S_R\{g^r\}$  i HMAC-a javnog ključa i obje privremene vrijednosti s ključem sjednice  $K$  koji je uspostavljen kroz DH,  $H_K\{PK_R, N_I, N_R\}$ .

$\text{LIST}_I$  sadrži klijentov javni ključ  $PK_I$  i popis podržanih algoritama  $L_I$ . Digitalni potpis u  $S_I\{L_I, g^i, g^r\}$  štiti popis podržanih algoritama dok HMAC, koristeći ključ sjednice  $K$ , štiti klijentov javni ključ  $H_K\{PK_I, N_R, N_I\}$ . Zadnja poruka u razmjeni je  $\text{LIST}_R$ . U toj poruci prenosi se lista podržanih protokola poslužitelja  $L_R$ , koja je zaštićena s digitalnim potpisom  $S_R\{g^i, g^r, N_I, N_R, L_R\}$ .

Nakon razmjene, klijent i poslužitelj imaju pristup popisu dogovorenih algoritama  $L_{IR}$ , međusobnim javnim ključevima i zajedničkoj tajni  $K_{IR}$  (poznatoj samo klijentu i poslužitelju).

#### 4.2.1 Rukovanje iznimkama

Tijekom komunikacije moguće su iznimke koje će prouzročiti prekid izmjena poruka. To je moguće u sljedećim slučajevima:

- različite Diffie-Hellman grupe u početnoj razmjeni,
- pogrešan zaštitni sažetak HMAC,
- pogrešan digitalni potpis ili
- nepostojanje zajedničkih algoritama za korištenje kod klijenta i poslužitelja.

U svrhu rukovanja iznimkama definiramo poruke ABORT koje su različite za klijenta i poslužitelja:

$$\text{ABORT}_R(R \rightarrow I): E_R, N_I, PK_R, S_R\{N_I, E_R\}$$

$$\text{ABORT}_I(I \rightarrow R): E_I, N_R, PK_I, S_I\{N_R, E_I\}$$

ABORT poruke sadrže kratak opis greške, odgovarajući *nonce*, javni ključ te digitalni potpis *noncea* i opisa greške.

### 4.3 Kriptografski algoritmi i ključevi za dogovaranje

Kako bi komunicirajuće strane mogle na siguran način dogovoriti kriptografske algoritme i ključeve potrebne za zaštitu komunikacije, obje strane u komunikaciji moraju podržavati nekoliko osnovnih vrsta kriptografskih algoritama za provedbu dogovora:

- postupak razmjene ključeva Diffie-Hellman za dobivanje zajedničke tajne  $g^{ir}$ ,

- operacija KDF za izračunavanje ključa sjednice  $K$  i tajne za zaštitu naknadne komunikacije  $K_{IR}$  iz zajedničke tajne dobivene DH postupkom i *nonceva*  $N_I$  i  $N_R$ ,
- funkcija kriptografskog sažetka za izračun HMAC operacija s ključem sjednice  $K$  i
- asimetrični algoritam za izračun i provjeru digitalnih potpisa tijekom dogovora.

Da bi se ostvario dogovor više različitih vrsta asimetričnih algoritama za zaštitu komunikacije nužno je razmijeniti javne ključeve za sve podržane asimetrične algoritme i na kraju odabratи ključ koji odgovara dogovorenom asimetričnom algoritmu.

Za zaštitu komunikacije u prototipu protokola koriste se sljedeći algoritmi: Diffie-Hellman 1024 MODP grupa [54], KDF operacija zasnovana na HMAC inaćici SHA-256 funkcije kriptografskog sažetka (HKDF [12]), SHA-256 kriptografski sažetak za HMAC operacije i RSA-1536 kao asimetrični algoritam za digitalne potpise. Uz te algoritme može se koristiti Diffie-Hellman algoritam zasnovan na eliptičnim krivuljama, kao i ECDSA algoritam za digitalni potpis (oba algoritma koriste ključeve veličine 256 bita).

## 4.4 Dogovor kriptografskih algoritama

Popisi podržanih kriptografskih algoritma podijeljeni su u kategorije. Te kategorije se mogu definirati prema trenutnim potrebama zaštite. Za zapis popisa koristi se format JSON [55]. Na slici 4.2 vidi se primjer s više različitih kategorija i skupova kriptografskih algoritama.

```
{  
    "hash": ["SHA-256", "RIPEMD", "SHA-1", "MD5"],  
    "secret_key": ["AES-CTR_256", "AES-CBC_128", "3DES_192"],  
    "public_key": ["ECDSA_192", "DSA_2048", "RSA_2048"],  
    "cryptographic_suite": [  
        "TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256",  
        "TLS_DHE_PSK_WITH_AES_128_CBC_SHA256",  
        "TLS_RSA_WITH_RC4_128_SHA" ]  
}
```

**Slika 4.2:** Popisi kriptografskih algoritama i skupova koji se dogovaraju

Za dogovor kriptografskih algoritama u programskom ostvarenju koriste se principi iz protokola SSH [36]. Ovisno o namjeni i okruženju u kojem se koristi protokol, moguće je definirati drukčije algoritme dogovora koji će uzimati dodatne parametre prilikom odluke. Prilagodba algoritma za dogovor za korištenje u okolini Interneta Stvari prikazana je u poglavljju 6.

Kriptografski algoritmi odabiru se zasebno za svaku vrstu algoritma: sažetak (hash), kriptografija tajnog ključa (*secret\_key*), kriptografija javnog ključa (*public\_key*) ili korištenjem skupa kriptografskih algoritama (*cryptographic suite*). Uz to, moguće je i dogovor drugih parametara sigurne komunikacije, kao što je način generiranja tajnih ključeva iz dobivene zajedničke tajne. U popisu algoritama za kriptografiju javnog i tajnog ključa zadnji dio u imenu

<pre>"hash": [     "SHA-256",     "RIPEMD",     "SHA-1"], "secret_key": [     "AES-CTR_256",     "AES-CBC_128",     "3DES_192"], "public_key": [     "RSA_1024",     "RSA_2048",     "ECDSA_192"]</pre>	<pre>"hash": [     "SHA3-512",     "SHA-512",     "SHA-256"], "secret_key": [     "AES-CTR_256",     "Salsa20_256",     "AES-CBC_128"], "public_key": [     "ECDSA_224",     "ECDSA_192",     "RSA_2048"]</pre>	<pre>"hash":     "SHA-256",  "secret_key":     "AES-CTR_256",  "public_key":     "RSA_2048"</pre>
---	---	---

(a) Popis algoritama klijenta    (b) Popis algoritama poslužitelja    (c) Dogovoreni popis algoritama

**Slika 4.3:** Primjer ponuđenih i dogovorenih algoritama

algoritma predstavlja duljinu ključa koji je podržan, npr. AES-CTR\_256 predstavlja algoritam AES koji koristi CTR način ulančavanja blokova s tajnim ključem duljine 256 bita.

Odabire se prvi algoritam s popisa klijenta koji se ujedno nalazi na popisu poslužitelja. Primjer dogovora algoritama između klijenta i poslužitelja prikazan je slikom 4.3. Preduvjet je da svi algoritmi budu poredani prema sigurnosti algoritma i duljini ključa. Algoritmi mogu biti poredani i prema zahtjevnosti izračunavanja ukoliko se radi o okruženju s ograničenim računalnim resursima. Prošireni model dogovora kriptografskih algoritama za okolinu Interneta stvari prikazan je u poglavljju 6.3.

Popisi preporuka za korištenje određenih algoritama redovito se izdaju od strane savjetodavnih agencija za pojedine države i zajednice. Te se preporuke mogu pronaći na web sjedištu tvrtke BlueKrypt \*.

Protokol ACAP ne podržava mogućnost ponovne razmjene popisa podržanih algoritama, već je potrebno ponoviti cijeli postupak dogovora kako bi se algoritmi i ključevi osvježili i prilagodili trenutnom stanju komunicirajućih strana. Nakon uspješnog dogovora ključeva i algoritama, aplikacija koja se koristi protokolom definira trajanje dogovorenih parametara. Preporučeni interval dogovora parametara je jedan sat, što je uobičajena vrijednost trajanja IPsec *Security Association* dogovora †. Taj interval može se prilagoditi ovisno o okruženju u kojem se koristi protokol ACAP.

\*BlueKrypt – Keylength – Cryptographic Key Length Recommendation – <http://www.keylength.com/en/>

†Trenutno strongSwan IPsec rješenje koristi preporučeni interval od jednog sata:

<https://wiki.strongswan.org/projects/strongswan/wiki/ExpiryRekey>, dok je to kod Cisco opreme interval od 24 sata: <http://www.cisco.com/c/en/us/td/docs/security/asa/asa95/configuration/vpn/asa-95-vpn-config.pdf>

## 4.5 Analiza sigurnosti i mehanizama protokola ACAP

Cilj protokola ACAP je omogućavanje jednostavne integracije i formalne verifikacije kako bi se mogla jamčiti sigurna komunikacija. Zbog toga je donekle ograničena količina funkcionalnosti kako bi se smanjila vjerojatnost pojavljivanja napada na protokol i njegovo programsko ostvarenje. Jedna od namjerno izostavljenih funkcionalnosti je podrška za ponovno dogovaranje koje se nastavlja na prethodni dogovor. Jedino što se pamti od početnog dogovora su javni ključevi obje komunicirajuće strane.

Poruke u protokolu sadrže samo podatke koji su nužni da bi se zaštitio dogovor algoritama i ključeva. Kako bi se spriječili napadi ponavljanjem poruka, u svaku poruku su uključene *nonce* vrijednosti i javni dijelovi DH razmjene. U porukama gdje nisu eksplicitno uključene one se koriste za izračun digitalnih potpisa i HMAC sažetaka.

Princip SIGMA [10] koristi se za sprječavanje napada čovjek u sredini jer implicitno povezuje ključ sjednice  $K$  (koji je dobiven iz zajedničke tajne uspostavljene kroz DH razmjenu korištenjem funkcije KDF) s javnim ključevima komunicirajućih strana  $PK_I$  i  $PK_R$ .

Pregled sigurnosnih mehanizama i razine na kojoj se ostvaruju prikazana je u tablici 4.1. Sigurnosni mehanizmi ostvareni na razini programskog ostvarenja dodatno su objašnjeni u poglavljiju 7.

**Tablica 4.1:** Pregled sigurnosnih mehanizama protokola ACAP

Sigurnosni mehanizam	Razina rješenja	Način ostvarivanja
Djelomični nedostatak dogovora algoritma u ACAP razmjeni	dizajn protokola	mali utjecaj i mogućnost dvostrukog dogovora
Računalna neovisnost tajnih ključeva	ostvarenje	ugrađen mehanizam za neovisno izračunavanje ključeva
Smanjenje utjecaja DoS napada	ostvarenje	najzahtjevniji izračuni odvojeni u pozadinski proces
Uspostavljanje povjerenja komunicirajućih strana	dizajn protokola	na razini postojećih rješenja
Sprječavanje napada čovjek u sredini	dizajn protokola	verificirano u sklopu formalne verifikacije
Sprječavanje napada ponavljanjem poruka	dizajn protokola i ostvarenje	verificirano u sklopu formalne verifikacije

### 4.5.1 Djelomični nedostatak dogovora algoritama u ACAP razmjeni

Cjelokupna razmjena sastoji se od samo četiri poruke te se uvijek radi sa svježim *nonce* vrijednostima. U slučaju da jedan od kriptografskih algoritama koji se koristi za razmjenu postane ranjiv, zbog malog broja poruka ne bi bilo moguće ozbiljno ugroziti razmjenu u sklopu

protokola ACAP. Zbog toga se na početku razmjene izmjenjuje isključivo informacija o Diffie-Hellman grupi kako bi se osiguralo dogovaranje s trenutno najboljim DH parametrima i dobila zajednička tajna dovoljne veličine. Trenutno programsko ostvarenje protokola ACAP koristi najnovije sigurne kriptografske algoritme, kao što su SHA-256 sažetak za HMAC izračune i kriptografija temeljena na eliptičnim krivuljama za algoritme asimetričnog kriptosustava (DH i digitalno potpisivanje). Važno je napomenuti da bi uvođenje jednostavnog načina dogovora algoritama za početni dogovor zahtijevalo još barem jednu poruku na početku komunikacije i omogućilo više točaka napada. U tom slučaju napadač bi mogao utjecati na razinu sigurnosti tako da se koristi najslabiji trenutno podržani kriptografski algoritam. Najbolji način izvedbe dogovora algoritma za provedbu protokola ACAP je dvostruko korištenje protokola: jednom za dogovor algoritama za razmjenu, a potom dogovor s novim algoritmima za konačni rezultat.

#### 4.5.2 Računalna neovisnost tajnih ključeva

Ključni dio principa SIGMA leži u računalnoj neovisnosti zajedničkih tajni koje se koriste za razmjene poruka. To se odnosi na ključ sjednice  $K$  i tajni ključ koji je rezultat cijelokupne razmjene. Oba ključa izračunavaju se iz iste zajedničke tajne, uz uvjet da jedan ključ ne otkriva nikakve podatke o drugom ključu, kako je opisano u dodatku članka [10]. Nakon što se završi razmjena protokola ACAP, komunicirajuće strane dobivaju tajni ključ odgovarajuće duljine za odabrani algoritam šifriranja tijekom dogovora koji je računalno neovisan od tajne koja je dogovorena putem protokola Diffie-Hellman. Kod programskog ostvarenja protokola važno je da je zajednička tajna dostupna samo u radnoj memoriji za vrijeme izvođenja dogovora i briše se odmah nakon razmjene kako bi se spriječilo otkrivanje podataka o tajni. U protokolu ACAP računalna neovisnost izvedena je pravilnim korištenjem funkcije KDF za dobivanjem računalno neovisnih ključeva.

#### 4.5.3 Smanjenje utjecaja DoS napada

Napadi uskraćivanja usluge (engl. *Denial of Service, DoS attacks*) imaju za cilj iscrpiti resurse koji su na raspolaganju određenoj usluzi kako bi se onemogućio pristup toj usluzi, što se najčešće postiže slanjem velikog broja poruka na određenu uslugu.

Da bi poslužitelj stvorio poruku  $\text{INIT}_R$ , kao odgovor na poruku  $\text{INIT}_I$ , mora odraditi vremenski zahtjevnu operaciju generiranja nasumičnog broja pogodnog za Diffie-Hellman razmjenu zajedno s izračunom ključa sjednice  $K$  te izračunom digitalnog potpisa i HMAC sažetka. To je računalno najzahtjevnija operacija cijelokupne razmjene i predstavlja moguću točku napada na protokol. Napadač bi mogao proizvesti ogromnu količinu lažnih  $\text{INIT}_I$  poruka koje bi uzrokovale nedostatak resursa na strani poslužitelja. Utjecaj takvog napada smanjuje se korištenjem Diffie-Hellman tajne tijekom kratkog vremenskog perioda i prethodnim izračunom ra-

čunalno zahtjevnijeg dijela  $\text{INIT}_R$  poruke  $(g^r, S_R\{g^r\})$ . Generiranje zahtjevnijeg dijela poruke u tom se slučaju obavlja kao pozadinska procedura u sklopu ACAP poslužitelja. Stoga su jedine operacije koje poslužitelj mora odraditi prilikom primitka poruke  $\text{INIT}_R$  izračun ključa sjednice  $K$  i izračun HMAC sažetka s tim ključem  $(H_K\{PK_R, N_I, N_R\})$ . Sličan mehanizam koristi se u protokolu JFK [21].

#### 4.5.4 Uspostavljanje povjerenja između komunicirajućih strana

Uspostavljanje povjerenja (engl. *trust establishment*) između dvije komunicirajuće strane predstavlja mogućnost da se na određeni način provjeri da svakoj od komunicirajućih strana pripada adresa s pomoću koje komunicira. To se u sklopu suvremenih računalnih sustava ostvaruje s pomoću infrastrukture javnog ključa (engl. *Public Key Infrastructure*, PKI) ili ranijom distribucijom identifikacijskih podataka. Na taj se način stvara početno povjerenje (engl. *root of trust*) na kojem se temelji sva buduća komunikacija.

Protokol ACAP ne postavlja ograničenja na razmjenu identifikacijskih podataka. Mogu se razmjenjivati samostalni javni ključevi kao i digitalno potpisani javni ključevi u obliku certifikata. To ne zahtijeva nikakvu promjenu u porukama  $\text{INIT}_R$  i  $\text{LIST}_I$ . U slučaju korištenja digitalno potpisanih certifikata aplikacija koja koristi ACAP mogla bi provjeravati valjanost certifikata kako bi se osiguralo povjerenje između komunicirajućih strana. To omogućava jednaku razinu sigurnosti kao u protokolima SSL/TLS i IPsec te zahtijeva istu infrastrukturu javnog ključa koja se trenutno koristi za osiguravanje povjerenja. Ako se ne koristi PKI, onda je potrebno odraditi prvu razmjenu protokola u kontroliranim uvjetima kako bi se zapamtili javni ključevi i ostvarilo početno povjerenje između komunicirajućih strana. Uz to, javne ključeve je moguće distribuirati drugim kanalom prije početne razmjene.

#### 4.5.5 Sprječavanje napada čovjek u sredini

U scenariju napada čovjek u sredini napadač se klijentu u komunikaciji predstavlja kao poslužitelj i obratno. Prepostavka protokola ACAP je da se dogovoreni tajni ključ i algoritmi vežu s javnim ključevima s kojima su ti parametri dogovoreni. Cilj je onemogućiti napadača da se predstavlja kao vlasnik klijentskog odnosno poslužiteljskog privatnog ključa, što izravno onemogućuje napadača da utječe na dogovorene kriptografske algoritme i zajedničku tajnu u slučaju prethodno uspostavljenog početnog povjerenja.

Prepostavimo da napadač pokuša promijeniti poruke u razmjeni kako bi došao do tajnog ključa koji će dvije strane koristiti za komuniciranje. Ako napadač promijeni poruku  $\text{INIT}_I$   $(g^i, N_I)$  tako da uvede drukčiji DH nasumični broj (npr.  $i'$  i izračunati  $g^{i'}$ ), mogao bi također izračunati povratnu  $\text{INIT}_R$  poruku korištenjem drugog para privatnog i javnog ključa. Uz to bi morao generirati još jedan DH nasumični broj (npr.  $r'$  i izračunati  $g^{r'}$ ) i tada bi poruka  $\text{INIT}_R$

bila oblika  $(g^{r'}, N_R, PK_{R'}, S_{R'}\{g^{r'}\}, H_{K'}\{PK_{R'}, N_I, N_R\})$ . To bi isto tako vodilo i do promjene ključa sjednice  $K$  u ključ  $K'$ . Sva naknadna komunikacija nakon te dvije početne poruke bila bi neuspješna zbog provjere HMAC sažetaka i komunicirajuće strane ne bi mogle provesti dogovor niti povezati javne ključeve s dogovorenim kriptografskim algoritmima i promijenjenom zajedničkom tajnom. Dodatno, otpornost na napade čovjek u sredini dokazana je postupkom formalne verifikacije koji je prikazan u poglavlju 5.4.

#### 4.5.6 Sprječavanje napada ponavljanjem poruka

U slučaju sprječavanja napada ponavljanjem poruka cilj napadača je u jednom trenutku spremiti cjelokupnu razmjenu poruka u dogovoru i kasnije odraditi identičnu razmjenu poruka kako bi komunicirajuće strane koristile stariji ključ i kriptografske algoritme. Ovakav napad bi omogućio da se svaki dogovor poništi tako da se ponovi stariji spremljeni dogovor.

U trenutku kad napadač šalje prethodno zabilježenu poruku, poslužitelj ima novu vrijednost *noncea*  $N_R$ . To znači da bi nova vrijednost trebala biti korištena za izračun HMAC sažetka  $(H_K\{PK_I, N_R, N_I\})$  koji je sadržan u poruci LIST<sub>I</sub>. To nije moguće odraditi u pasivnom scenariju slanja prethodno zabilježenih poruka. U slučaju aktivnog napada napadač bi trebao biti u mogućnosti promijeniti zabilježene poruke, što je onemogućeno promjenom Diffie-Hellman vrijednosti za kasnije razmjene. Shodno tome sve provjere HMAC sažetaka bile bi negativne.

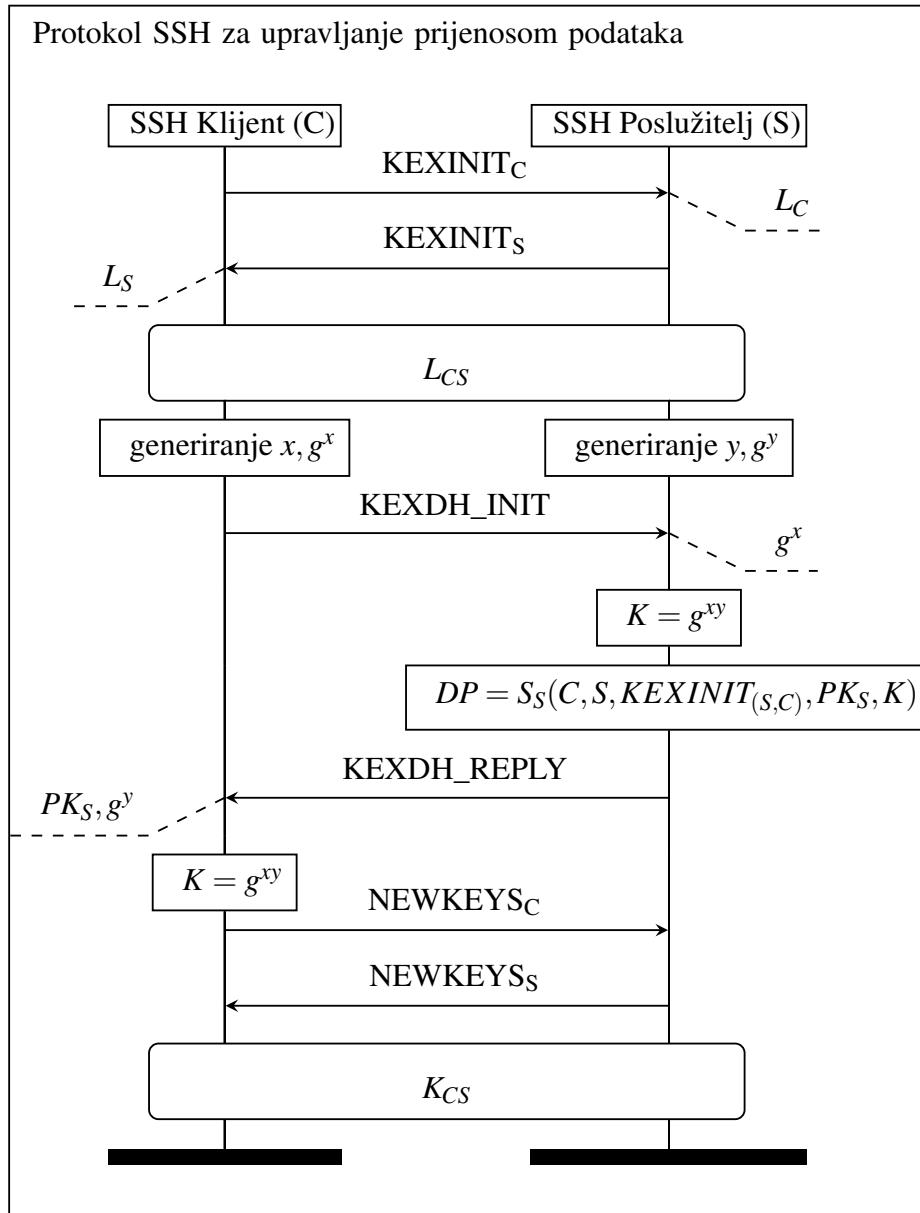
### 4.6 Usporedba protokola ACAP s protokolom za upravljanje prijenosom podataka u protokolu SSH

Za dogovor algoritama i ključeva u protokolu za upravljanjem prijenosom podataka protokola SSH potrebno je šest poruka. Prve dvije poruke (KEXINIT<sub>C</sub> i KEXINIT<sub>S</sub>) služe za razmjenu i dogovor algoritama koji će se koristiti za dogovor ključeva. Dogovaraju se algoritmi za Diffie-Hellman (DH) razmjenu, algoritmi simetričnog kriptosustava i algoritmi za stvaranje zaštitnog sažetka HMAC. Nakon dogovora algoritma razmjenjuju se poruke za dogovor DH zajedničke tajne (KEXDH\_INIT i KEXDH\_REPLY). Poruka KEXDH\_REPLY također sadrži javni ključ poslužitelja i digitalni potpis razmjene s kojim se provjerava vjerodostojnost dosadašnje razmjene. Na kraju se razmjenjuju poruke za dogovor novih ključeva (NEWKEYS<sub>C</sub> i NEWKEYS<sub>S</sub>) nakon kojih su uspostavljeni novi ključevi za zaštitu komunikacije. Cjelokupna razmjena poruka prikazana je na slici 4.4. Razmjenu je moguće skratiti na pet poruka tako da se poruka NEWKEYS<sub>S</sub> šalje zajedno s porukom KEXDH\_REPLY.

Protokol za upravljanje prijenosom podataka unutar SSH mogao bi se iskoristiti za uslugu koju pruža protokol ACAP (dogovor kriptografskih algoritama i ključeva), ali pritom je važno uzeti u obzir razlike između ta dva protokola. Razlike protokola za upravljanje prijenosom

podataka u odnosu na protokol ACAP su sljedeće:

- veći broj poruka u razmjeni,
- razmjena javnog ključa samo za poslužitelj,
- manji broj poruka s digitalnim potpisom,
- kasnije otkrivanje promjene poruka za dogovor algoritama i
- nemogućnost smanjivanja opterećenosti poslužitelja prilikom generiranja digitalnog potpisa razmjene.



**Slika 4.4:** Grafički prikaz razmjene poruka protokola za upravljanje prijenosom podataka protokola SSH

Kada bi se principi iz protokola za upravljanje prijenosom podataka primijenili za uslugu koju pruža protokol ACAP tada bi za potpunu razmjenu trebala barem jedna poruka više. Dodatni problem je što komunicirajuće strane nisu ravnopravne već se razmjenjuje samo javni ključ poslužitelja i ne postoji autentifikacija klijenta u sklopu protokola za upravljanje prijenosom

podataka. Autentifikacija klijenta se odrađuje u sklopu protokola za autentifikaciju korisnika protokola SSH. Kako se digitalni potpis koristi samo kod slanja digitalnog potpisa razmjene složenost kriptografskih operacija tijekom razmjene je niža nego kod protokola ACAP, ali isto tako nije moguće autentificirati klijenta.

U slučaju napada čovjek u sredini promjenu početnih poruka KEXINIT bit će moguće otkriti tek nakon primitka KEXDH\_REPLY poruke koja sadrži digitalni potpis razmjene. U sklopu protokola ACAP promjenu poruke u razmjeni moguće je prepoznati prije jer su sve poruke osim prve zaštićene zaštitnom sumom HMAC ili digitalnim potpisom. Konačno digitalni potpis razmjene sadrži podatke koje je poslao klijent, te ga je potrebno generirati u trenutku kad poslužitelj primi i poruku KEXINIT<sub>C</sub> i KEXDH\_INIT. Zbog toga je protokol SSH osjetljiviji na napad uskraćivanja usluge u odnosu na ACAP.

# Poglavlje 5

## Formalna verifikacija modela

Ispravnost protokola može se provjeriti formalnom verifikacijom modela protokola uz zadane preduvjete koje model mora zadovoljiti. Protokol ACAP je specificiran i formalno verificiran s pomoću alata Scyther [56]. Scyther omogućuje formalnu verifikaciju modela protokola i sigurnosnih zahtjeva uz korištenje sigurnosnih primitiva poput sažetaka, simetrične kriptografije i kriptografije javnog ključa.

U ovom poglavlju pojašnjena je sintaksa za specifikaciju modela protokola i specificiran je model protokola ACAP. U nastavku se prikazuje izvođenje definiranog modela i analiziraju se rezultati verifikacije.

### 5.1 Značajke jezika SPDL

Alat Scyther za opis protokola koristi jezik SPDL (*Security Protocol Definition Language*). U tom jeziku moguće je koristiti sljedeće pojmove:

- Kriptografske funkcije za standardne kriptografske operacije:
  - kriptografski sažetak (`hashfunction`),
  - simetrično šifriranje podataka (`{secret}key`)
  - asimetrično šifriranje podataka (`{secret}pk{I}` ili `{hash}sk{I}`).
- Korisnički definirane konstante za definiciju jednosmjernih funkcija (`Function`), jedinstvenih privremenih vrijednosti (`Nonce`) i univerzalnih varijabli (`Ticket`).
- Definiciju protokola (`protocol ime_protokola`) koja obuhvaća uloge (`role`) komunicirajućih strana. U svakoj ulozi definiraju se variable koje se koriste za komunikaciju zajedno s događajima slanja (`send`) i primanja (`recv`) poruka koji opisuju komunikaciju. Nakon opisa komunikacije slijede definicije sigurnosnih zahtjeva na protokol (`claim`) koje se provjeravaju prilikom verifikacije.

U idućem poglavlju dan je primjer formalne verifikacije jednostavnog protokola za uzajamnu autentifikaciju.

## 5.2 Primjer verifikacije jednostavnog protokola

Za primjer modeliranja i verifikacije jednostavnog protokola odabran je protokol Needham-Schroeder-Lowe (NSL) [57]. Protokol NSL je popravljena inačica Needham-Schroeder protokola čija je osnovna namjena uzajamna autentifikacija dvije komunicirajuće strane. Razmjena poruka u protokolu NSL prikazana je izrazima na slici 5.1, gdje I predstavlja započinjatelja (engl. *initiator*), a R primatelja (engl. *responder*),  $N_X$  su *nonce* vrijednosti,  $PK_X$  javni ključevi, dok su uloge  $I$  i  $R$  identiteti komunicirajućih strana (certifikati ili javni ključevi).

$$\begin{aligned} I \rightarrow R: & \quad (I, N_I)_{PK_R} \\ R \rightarrow I: & \quad (N_I, N_R, R)_{PK_I} \\ I \rightarrow R: & \quad (N_R)_{PK_R} \end{aligned}$$

**Slika 5.1:** Opis protokola Needham-Schroeder-Lowe

U modelu protokola definiraju se dvije uloge: započinjatelj I i primatelj R. Definicija dvije osnovne uloge protokola NSL prikazana je na slici 5.2a. Uloga I predstavlja klijenta, odnosno započinjatelja komunikacije (engl. *initiator*), a uloga R predstavlja poslužitelja, odnosno primatelja (engl. *responder*).

```
protocol ns1(I, R) {
    # uloga započinjatelja
    role I {
        ...
        send_1(I, R, {I, ni}pk(R));
        recv_2(R, I, {ni, nr, R}pk(I));
        send_3(I, R, {nr}pk(R));
        ...
    }
    # uloga primatelja
    role R {
        ...
    }
}
```

(a) Definicija uloga

role I {  
    fresh ni: Nonce; var nr: Nonce;  
  
    send\_1(I, R, {I, ni}pk(R));  
    recv\_2(R, I, {ni, nr, R}pk(I));  
    send\_3(I, R, {nr}pk(R));  
    ...  
}

(b) Definicija varijabli i razmjene za ulogu I

**Slika 5.2:** Definicija protokola NSL

Na početku definicije svake uloge nalazi se definicija varijabli nužnih za pojedinu ulogu. Ulogama u primjeru potrebne su samo varijable za *nonce* vrijednosti. Kod definicije varijabli važno je napomenuti da li ta uloga stvara vrijednost ili zaprima vrijednost kroz razmjenu poruka. Kod uloge koja stvara vrijednost navodi se ključna riječ *fresh*, a kod uloge koja zaprima vrijednost navodi se ključna riječ *var*. Deklaracija varijabli za ulogu započinjatelja komunikacije prikazana je na slici 5.2b.

Uz deklaraciju varijabli na slici 5.2b prikazana je i razmjena poruka u protokolu NSL. Razmjena poruka u jeziku SPDL definira se događajima *send* i *recv*. U sklopu tih događaja specificira se pošiljatelj, primatelj i sadržaj poruke na sljedeći način:

- slanje poruke: *send\_n(pošiljatelj, primatelj, sadržaj)*,

- primanje poruke: `recv_n(pošiljatelj, primatelj, sadržaj)`, pri čemu  $n$  identificira te događaje i služi za njihovo grupiranje. Za svaki događaj slanja poruke u jednoj ulozi, u jednoj od preostalih uloga mora biti definiran odgovarajući događaj primanja poruke. U suprotnom slučaju formalna definicija protokola nije valjana.

Formalna definicija razmjene poruka za protokol NSL na slici 5.2b proizlazi iz opisa protokola na slici 5.1. Posljednji dio u definiciji uloge sadrži uvjete koje formalno modelirani protokol mora zadovoljavati. Protokol NSL prema definiciji mora zadovoljavati sljedeće zahtjeve:

- Tajnost parametara  $nonce$ ,  $N_I$  i  $N_R$ . U jeziku SPDL to se definira s ključnom riječi `Secret` koja kao argument prima parametar čija se tajnost ispituje.
- Da su svi događaji definirani unutar uloge uspješno završili. Definira se s ključnom riječi `Niagree` koja proizlazi iz izraza *Non-injective agreement* [58].
- Da se događaji slanja unutar jedne uloge moraju izvršiti prije odgovarajućeg događaja primanja unutar druge uloge. Time se označava da napadač nije u mogućnosti poslati poruku koja bi na neki način poremetila razmjenu. Definira se s ključnom riječi `Nisynch` koja označava pojam *Non-injective synchronization* [58].

Zadovoljavanje određenih zahtjeva definira se s ključnom riječi `claim`. Uz ključnu riječ navodi se uloga u kojoj zahtjev mora biti zadovoljen, definicija zahtjeva i parametar ako je potreban. Formalna definicija zahtjeva za ulogu započinjatelja I prikazana je na slici 5.3.

```
role I {
    ...
    claim_i1(I,Secret,ni);
    claim_i2(I,Secret,nr);
    claim_i3(I,Niagree);
    claim_i4(I,Nisynch);
}
```

**Slika 5.3:** Definicija zahtjeva za ulogu I protokola NSL

Cjelokupan model za protokol NSL prikazan je na slici 5.4a i u njemu vidljive su razlike između definicija varijabli te događaja primanja i slanja poruka između uloga započinjatelja I i primatelja R.

Potpuni formalni model protokola provjerava se pomoću alata Scyther i opcije `characterize`, koja dani model pokreće i provjerava jesu li definirane uloge uspješno završile definirane događaje slanja i primanja. Potom slijedi formalna verifikacija specificiranih uvjeta za modelirani protokol. Rezultat provjere i formalne verifikacije specificiranog modela prikazan je na slici 5.5. Dijagram toka za protokol NSL prikazan je na slici 5.4b.

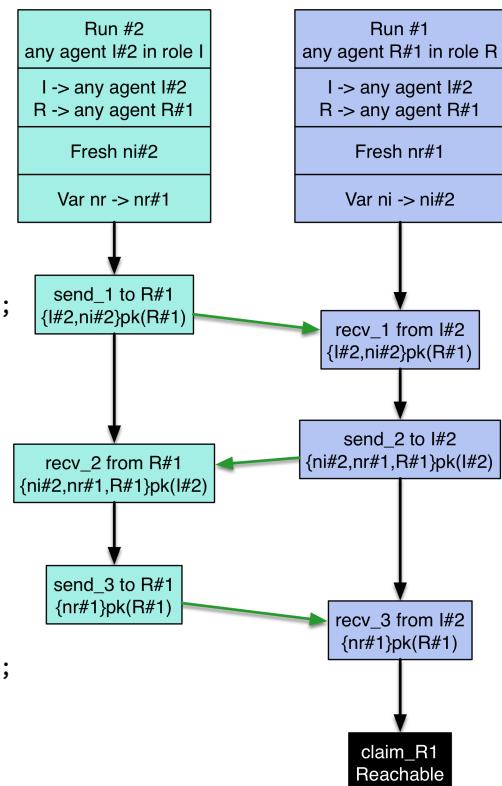
```
protocol ns1(I,R) {
    role I {
        fresh ni: Nonce; var nr: Nonce;

        send_1(I,R,{I,ni}pk(R));
        recv_2(R,I,{ni,nr,R}pk(I));
        send_3(I,R,{nr}pk(R));

        claim_i1(I,Secret,ni); claim_i2(I,Secret,nr);
        claim_i3(I,Niagree); claim_i4(I,Nisynch);
    }
    role R {
        var ni: Nonce; fresh nr: Nonce;

        recv_1(I,R,{I,ni}pk(R));
        send_2(R,I,{ni,nr,R}pk(I));
        recv_3(I,R,{nr}pk(R));

        claim_r1(R,Secret,ni); claim_r2(R,Secret,nr);
        claim_r3(R,Niagree); claim_r4(R,Nisynch);
    }
}
```



(a) SPDL model protokola NSL

(b) Dijagram toka za protokol NSL

Slika 5.4: SPDL model i dijagram toka za protokol Needham-Schroeder-Lowe

```
$ scyther-linux -c ns1.spdl
claim ns1,I Reachable_I1 - Ok [exactly 1 variant]
claim ns1,R Reachable_R1 - Ok [exactly 1 variant]
$ scyther-linux ns1.spdl
claim ns1,I Secret_i1 ni Ok [proof of correctness]
claim ns1,I Secret_i2 nr Ok [proof of correctness]
claim ns1,I Niagree_i3 - Ok [proof of correctness]
claim ns1,I Nisynch_i4 - Ok [proof of correctness]
claim ns1,R Secret_r1 ni Ok [proof of correctness]
claim ns1,R Secret_r2 nr Ok [proof of correctness]
claim ns1,R Niagree_r3 - Ok [proof of correctness]
claim ns1,R Nisynch_r4 - Ok [proof of correctness]
```

Slika 5.5: Rezultati provjere i verifikacije modela protokola NSL

### 5.3 Model predloženog protokola ACAP

Opis protokola ACAP korištenjem SPDL-a sastoji se od glavnog protokola (protocol acap{I, R}) i pomoćnog protokola (protocol @executability {DH, 0}), koji se koristi za operacije vezane uz Diffie-Hellman razmjenu ključeva.

Glavni protokol ACAP sastoji se od dvije glavne uloge I i R isto kao i protokol Needham-Schreoder-Lowe. U sklopu uloga definiraju se varijable koje su specifične za protokol ACAP.

Svakoj ulozi potreban je sljedeći niz varijabli:

- eksponent za Diffie-Hellman dogovor ključeva,
- nasumične vrijednosti za jednokratno korištenje (engl. *nonce*) za obje strane,
- univerzalna varijabla za zaprimanje javnog dijela Diffie-Hellman razmjene,
- dvije varijable koje sadrže popise algoritama.

Deklaracije varijabli za obje uloge prikazane su na slici 5.6.

```
protocol acap(I,R) {
    role I {
        # stvaranje novog DH eksponenta i noncea
        fresh i, Ni: Nonce;
        # varijabla za zaprimanje noncea
        var Nr: Nonce;
        # varijabla za zaprimanje javnog dijela DH razmjene
        var Gr: Ticket;
        # stvaranje novog popisa algoritama
        fresh Li: AlgList;
        # varijabla za zaprimanje popisa algoritama
        var Lr: AlgList;
        ...
    }
    role R {
        fresh r, Nr: Nonce; var Ni: Nonce;
        var Gi: Ticket;
        fresh Lr: AlgList; var Li: AlgList;
        ...
    }
}
```

**Slika 5.6:** Prikaz varijabli za uloge započinjatelja i primatelja

Nakon definicije varijabli i njihovog stvaranja slijedi tijek razmjene poruka između komunicirajućih strana. Da bi dvije uloge unutar protokola uspješno razmijenile poruke nužno je da svaki događaj slanja poruke ima odgovarajući događaj primanja poruke s druge strane. Na slici 5.7 prikazani su događaji slanja i primanja na strani započinjatelja dok je na slici 5.8 prikazano isto za ulogu primatelja.

Prvi događaj slanja (send\_1( I, R, g(i), Ni);) ide od uloge I prema ulozi R i sadrži parametre definirane za poruku INIT<sub>I</sub>. Odgovarajući događaj primanja (recv\_1( I, R, Gi,

```
role I {
    ...
    send_1( I, R, g(i), Ni);
    recv_!2( R, I, Gr, Nr, R, {h(Gr)}sk(R), MAC(h(Gr,i), R, Ni, Nr) );
    send_!3( I, R, I, Li, {h(Li, g(i), Gr)}sk(I), MAC(h(Gr,i), I) );
    recv_4( R, I, Lr, {h(g(i), Gr, Lr)}sk(R) );
    ...
}
```

**Slika 5.7:** Komunikacija sa strane započinjatelja

```
role R {
    ...
    recv_1( I, R, Gi, Ni);
    send_!2( R, I, g(r), Nr, R, {h(g(r))}sk(R), MAC(h(Gi,r), R, Ni, Nr) );
    recv_!3( I, R, I, Li, {h(Li, Gi, g(r))}sk(I), MAC(h(Gi,r), I) );
    send_4( R, I, Lr, {h(Gi, g(r), Lr)}sk(R) );
    ...
}
```

**Slika 5.8:** Komunikacija sa strane primatelja

$Ni$ );) definira da poruka dolazi od uloge  $I$ , a prima ju uloga  $R$ . Ovakav način pozivanja označava komunikaciju između uloga  $I$  i  $R$ .

U danom je primjeru vidljivo da događaj `send_1` odgovara događaju `recv_1` uz jednakost izraza  $g(i)$  i  $Gi$ . Isto vrijedi i za izraze `send_4` i `recv_4` ako se u obzir uzme i jednakost  $g(r)$  i  $Gr$ . Navedene jednakosti su potrebne kako bi komunicirajuće strane mogle izvesti potrebne operacije.

U modelu je potrebno specificirati i zasebnu ulogu koja će omogućiti razmjenu poruka definiranu događajima `send_!2` i `recv_!2` te `send_!3` i `recv_!3` koji izravno ne odgovaraju jedan drugome. Ti događaji uključuju poseban znak ! (uskličnik), koji specificira da se poruka šalje svim ulogama koje su dostupne u sklopu procesa verifikacije, a ne samo između definiranih primatelja i pošiljatelja. Uloge u procesu verifikacije obuhvaćaju sve definirane uloge, uključujući i ulogu napadača. U sljedećem potpoglavlju dan je opis dodatnog protokola za modeliranje Diffie-Hellman operacija.

### 5.3.1 Modeliranje Diffie-Hellman operacija

U alatu Scyther v1.1.3, koja je bila dostupna za vrijeme pisanja disertacije nije podržano izravno korištenje Diffie-Hellman operacija. Stoga je potrebno napraviti apstrakciju s pomoću integriranih operacija. Diffie-Hellman (DH) operacije izračunavanja zamijenjene su jednosmjernom funkcijom. DH operacija definirana izrazom  $A = g^a \text{mod } p$  zamijenjena je funkcijom  $g(\text{hashfunction } g)$  i izrazom  $g(a)$ , zbog pretpostavke da operacija nije reverzibilna, što vrijedi za tu vrstu potenciranja i računanja ostatka kao i za funkciju sažetka unutar alata Scyther. Zbog tog svojstva primatelj podataka poslanu informaciju  $g(i)$  prima pomoću univerzalne varijable

Ticket Gi.

Drugi dio DH izračuna odrađuje se s pomoću funkcije sažetka  $h$  (hashfunction  $h$ ) i rezultat te operacije je zajednička tajna u sklopu DH razmjene. Izraz  $S = (g^a \bmod p)^b$  istovjetan je izrazu  $h(g(a), b)$ . Potrebno je i definirati da je operacija komutativna kako bi eventualni napadač uz poznavanje jednog tajnog dijela DH razmjene ( $a$  ili  $b$ ) mogao doći do razmijenjene tajne. To je izvedeno s pomoću uloge DH u pomoćnom protokolu (@executability) kojim se postiže jednakost  $h(g(r), i) = h(g(i), r)$ , kako je prikazano na slici 5.9. U toj specifikaciji je korišten simbol uskličnika kod slanja i primanja kako bi ta jednakost vrijedila za sve definirane uloge i napadača.

```
protocol @executability (DH, 0) {
    role DH {
        # definicija DH eksponenata
        var i, r: Nonce;
        # primi od bilo kojeg pošiljatelja
        recv_!DH1( DH, DH, h(g(r),i) );
        # pošalji bilo kojem primatelju ekvivalentni izraz
        send_!DH2( DH, DH, h(g(i),r) );
    }
    role 0 { ... }
}
```

**Slika 5.9:** Komutativnost Diffie-Hellman operacije

Pomoćni protokol sadrži još jednu ulogu 0 koja se koristi za uspješnu razmjenu poruka  $\text{INIT}_R$  i  $\text{LIST}_I$ , koje koriste zajedničku tajnu uspostavljenu kroz DH za stvaranje zaštitnog HMAC sažetka. Uloga 0 prikazana je na slici 5.10. Razmjena poruka između uloga I i R odvija se kroz tu ulogu. Na slici su prikazani događaji send i recv koji su povezani tom ulogom.

```
role 0 {
    var i, r, Ni, Nr: Nonce;
    var I, R: Agent; var Li: AlgList;
    ### razmjena poruke INITR (R->I) #####
    #send događaj u ulozi R
    #send_!2( R, I, g(r), Nr, R, {h(g(r))}sk(R), MAC(h(Gi,r), R, Ni, Nr) );
    recv_!01( 0, 0, g(r), Nr, R, {h(g(r))}sk(R), MAC(h(g(i),r), R, Ni, Nr) );
    send_!02( 0, 0, g(r), Nr, R, {h(g(r))}sk(R), MAC(h(g(r),i), R, Ni, Nr) );
    #recv događaj u ulozi I
    #recv_!2( R, I, Gr, Nr, R, {h(Gr)}sk(R), MAC(h(Gr,i), R, Ni, Nr) );
    ### razmjena poruke LISTI (I->R) #####
    #send događaj u ulozi I
    #send_!3( I, R, I, Li, {h(Li, g(i), Gr)}sk(I), MAC(h(Gr,i), I) );
    recv_!03( 0, 0, I, Li, {h(Li, g(i), g(r))}sk(I), MAC(h(g(r),i), I) );
    send_!04( 0, 0, I, Li, {h(Li, g(i), g(r))}sk(I), MAC(h(g(i),r), I) );
    #recv događaj u ulozi I
    #recv_!3( I, R, I, Li, {h(Li, Gi, g(r))}sk(I), MAC(h(Gi,r), I) );
}
```

**Slika 5.10:** Razmjena poruka  $\text{INIT}_R$  i  $\text{LIST}_I$  definirana u ulozi 0

## 5.4 Formalna verifikacija modela

Prvi korak provjere ispravnosti modela je provjera uspješnog završetka komunikacije s obje strane. Alat Scyther nakon uspješne provjere generira dijagram toka za specificirani model. Dijagram toka za model protokola ACAP prikazan je na slici 5.11.



Slika 5.11: Dijagram toka za protokol ACAP

Na slici se vidi da su za jednu uspješnu razmjenu poruka potrebne tri uloge: uloga započinjatelja (Role I) i uloga primatelja (Role R) u sklopu glavnog protokola i dodatna uloga za upravljanje DH razmjenom (Role O) u sklopu pomoćnog protokola.

### 5.4.1 Definicija sigurnosnih zahtjeva

Nakon definiranja poruka i tijeka komunikacije, za svaku ulogu u modelu potrebno je odrediti sigurnosne zahtjeve koji trebaju biti zadovoljeni. Sigurnosni zahtjevi za modelirani protokol su sljedeći:

- Javni dijelovi DH razmjene moraju ostati nepromijenjeni za vrijeme razmjene kako bi obje strane imale istu zajedničku tajnu za vrijeme komunikacije. To se provjerava s pomoću sigurnosnog zahtjeva `claim` s ključnim riječima `Running` i `Commit`.
- Zajednička tajna ni u jednom trenutku tijekom razmjene ne smije doći do uloge napadača. Ta tvrdnja se provjerava koristeći ključnu riječ `SKR` (engl. *Session Key Reveal*) koja označava otkrivanje ključa trenutne sjednice.
- Napadač ne smije imati nikakav utjecaj na razmjenu poruka, što predstavlja otpornost protokola na napade ponavljanja i napade s napadačem u sredini (*man in the middle*). Sigurnosni zahtjev postavlja se s ključnom riječi `Nisynch` koja predstavlja pojam ne injektivne sinkronizacije (engl. *Non-injective synchronization*) [58].

Definirani sigurnosni zahtjevi prikazani su na slici 5.12.

```
role R {
    recv_1( I, R, Gi, Ni);
    # početak praćenja javnih dijelova DH razmjene
    claim_R1( R, Running, I, g(r), Gi);
    send_!2( ... ); recv_!3( ... ); send_4( ... );

    # provjera istovjetnosti početnih i krajnjih vrijednosti
    claim_R2( R, Commit, I, g(r), Gi);
    # zahtjev tajnosti nad zajedničkom tajnom
    claim_R3( R, SKR, KDF(h(Gi,r)) );
    # zahtjev da napadač nije u mogućnosti utjecati na komunikaciju
    claim_R4( R, Nisynch );
}
```

Slika 5.12: Definicija sigurnosnih zahtjeva za modelirani protokol

### 5.4.2 Rezultati formalne verifikacije

Verificirani model protokola ACAP nalazi se u dodatku A. Na adresi <http://public.tel.fer.hr/acap> nalazi se potpuni model i rezultati verifikacije uz programsko ostvarenje i primjer izvođenja. Prilikom pokretanja verifikacije moguće je odrediti broj paralelnih uloga koje će se izvoditi u pokušaju napada na protokol. Moguće je verificirati i uz neograničen broj paralelnih

uloga, što u potpunosti dokazuje istinitost definiranih sigurnosnih zahtjeva. Model protokola ACAP verificiran je uz neograničen broj paralelnih uloga kod testiranja i pokazano je da su svi definirani sigurnosni zahtjevi zadovoljeni. Rezultati verifikacije s alatom Scyther prikazani su na slici 5.13. Zahtjevi I2 i R2 pokazuju da ne dolazi do promjene parametara DH razmjene, zahtjevi I3 i R3 pokazuju da zajednička tajna nije otkrivena, a zahtjevi I4 i R4 da napadač ne može utjecati na razmjenu poruka.

```
$ scyther-linux acap.spdl
claim  acap,I Commit_I2      (R,g(i),Gr)      Ok  [no attack within bounds]
claim  acap,I SKR_I3 KDF(h(Gr,i))  Ok  [no attack within bounds]
claim  acap,I Nisynch_I4      -      Ok  [no attack within bounds]
claim  acap,R Commit_R2      (I,g(r),Gi)  Ok  [no attack within bounds]
claim  acap,R SKR_R3 KDF(h(Gi,r))  Ok  [no attack within bounds]
claim  acap,R Nisynch_R4      -      Ok  [no attack within bounds]
```

**Slika 5.13:** Rezultati verifikacije protokola s alatom Scyther

Proces formalne verifikacije posebno je zahtjevan zbog zasebnog modeliranja Diffie-Hellman operacija koje prouzrokuju veći broj stanja za provjeru nego što je potrebno. Zahtjevnost se očituje u količini vremena i resursa koji su potrebni za verifikaciju protokola. Što je veći broj paralelnih uloga u verifikaciji to verifikacija traje dulje i veći su zahtjevi na radnu memoriju (RAM) u sustavu. Formalna verifikacija izvodila se na 6 jezgara procesora Intel Xeon E5-2680 (frekvencije 2.70 Ghz) s 40 GB radne memorije. Trajanje verifikacije prikazano je u tablici 5.1. Verifikacija se izvodila paralelno na svih 6 jezgara kako bi se skratilo vrijeme izvođenja verifikacije, ali to je ujedno i povećalo zahtjeve za radnom memorijom.

**Tablica 5.1:** Trajanje formalne verifikacije ovisno o broju paralelnih uloga

Broj uloga	Trajanje verifikacije (min)
4	2.3
5	13.0
6	52.0
7	174.0
8	457.0
9	877.0
10	1255.0
neograničen	1532.0

Prema dobivenim vrijednostima trajanja verifikacije može se zaključiti kako se povećavanjem broja uloga značajno povećava broj stanja koje je potrebno provjeriti, što iziskuje veću količinu vremena. Može se primjetiti da su zadnje dvije verifikacije slične po trajanju, što upućuje da je verifikacija s 10 uloga došla blizu maksimalnog broja stanja kojeg je moguće postići sa specificiranim modelom protokola.

## Poglavlje 6

# Primjena protokola za sigurno dogovaranje u okolini Interneta stvari

Okolina Interneta stvari predstavlja mnoštvo uređaja različitih mogućnosti koji su povezani putem Interneta i služe za pružanje raznih usluga, a temelji se na vjerodostojnom prikupljanju podataka i pouzdanom upravljanju uređajima unutar okoline.

Uređaji koji komuniciraju u okolini Interneta stvari često raspolažu ograničenim mogućnostima. Temeljne mogućnosti uređaja Interneta stvari su:

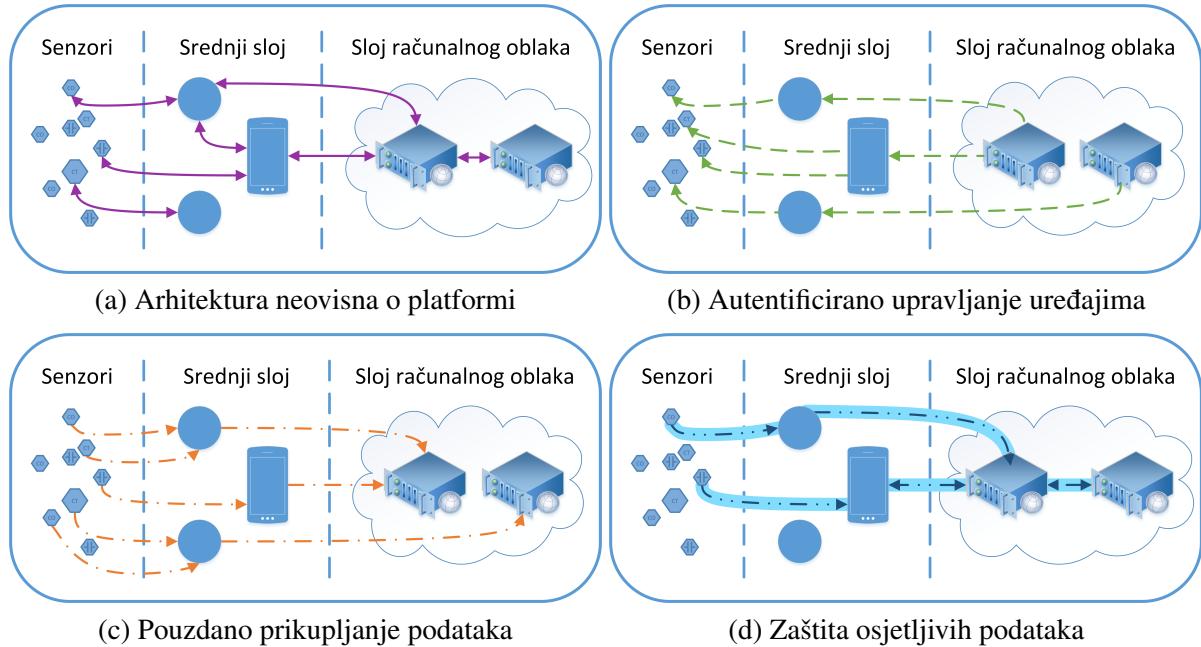
- vrsta i trenutna razina izvora napajanja,
- propusnost i kvaliteta mrežne povezanosti i
- računalna snaga na raspolaganju.

Uređaji u okolini Interneta stvari moraju biti u mogućnosti međusobno sigurno komunicirati i pružati određenu razinu usluge. To se postiže s pomoću sigurne komunikacije. Sigurna komunikacija zasniva se na prilagodljivom dogовору криптографских алгоритама и потребних ћљућева, који је изведен у склопу предложеног протокола ACAP. Уз протокол се користе додатне операције заштите које користе резултате протокола ACAP (договорене криптографске ћљућеве и алгоритме) као улаз за заштиту података оvisno o потребној разини заштите.

## 6.1 Primjene sigurne komunikacije u okolini Interneta stvari

Okolina Interneta stvari može se definirati kao slojevita arhitektura koja povezuje razne uređaje kroz različite mrežne okoline. Cjelokupna arhitektura prikazana je kroz par ključnih scenarija na slici 6.1, a temelji se na arhitekturi predstavljenoj u [49]. Cjelokupna arhitektura sastoji se od tri osnovna sloja:

- *Senzorski sloj* predstavlja izvor podataka cijele arhitekture, a čine ga uređaji ograničenih računalnih sposobnosti koji prikupljaju podatke iz okoline ili obavljaju jednostavne poslove.



**Slika 6.1:** Primjene sigurne komunikacije u okolini Interneta stvari

- *Srednji sloj* se koristi za prikupljanje podataka iz senzorskog sloja i njihovo prosljeđivanje u gornji sloj računalnog oblaka. Uređaji na ovom sloju raspolažu s više resursa i mogu obavljati filtriranje i grupiranje podataka srednjeg sloja prije prosljeđivanja.
- *Sloj računalnog oblaka* predstavlja sloj usluge na kojem se obrađuju i skladište podaci koji se prikupljaju u senzorskom sloju, a filtriraju u srednjem sloju. Podaci obrađeni na ovom sloju koriste se za pronalaženje novih informacija na temelju kojih se upravlja ostatkom arhitekture u cilju ostvarivanja kompleksnih usluga upravljanja arhitekturom.

Osnovne primjene sigurne komunikacije prikazane su na slici 6.1, a dijele se u četiri osnovne kategorije. Zaštita komunikacije ključna je za sigurnost cijele platforme kako bi se omogućila provjera autentičnosti izvora podataka i zaštitila komunikacija između uređaja koja predstavlja temelj cjelokupne okoline Interneta stvari.

### 6.1.1 Arhitektura neovisna o platformi

Neovisnost o platformi i uređajima u okolini Interneta stvari omogućena je kroz uporabu prilagodljivog protokola ACAP za dogovor ključeva i algoritama koji će se koristiti za zaštitu komunikacije. Protokol ACAP omogućuje dogovor u oportunističkom mrežnom okruženju koje je karakteristično za Internet stvari i podržava mogućnost dinamičke promjene kriptografskih algoritama i duljine ključeva kako bi se ostvarila sigurnija komunikacija u slučaju ranjivosti ili smanjilo opterećenje sustava u slučaju nedostatka resursa. Poruke u sustavu mogu se razmjenjivati između dva uređaja ili biti prosljeđene od senzorskog sloja sve do sloja računalnog oblaka, kako je prikazano na slici 6.1a.

### **6.1.2 Autentificirano upravljanje uređajima**

Svi uređaji, neovisno o sloju na kojem se nalaze, moraju imati sposobnost autentifikacije kako bi se omogućilo pravilno upravljanje cijelog sustava temeljenog na Internetu stvari. Upravljačke poruke obično dolaze iz sloja računalnog oblaka i služe za prilagodbu uređaja u nižim slojevima arhitekture. Tijek upravljačkih poruka prikazan je na slici 6.1b. Autentifikacija upravljačkih poruka omogućava se tako da pošiljatelj digitalno potpisuje poruke kako bi primatelj mogao provjeriti izvor poruke i njenu cjelovitost. Digitalni potpis je računalno zahtjevna operacija i koristi se samo za upravljačke poruke kako bi se uštedjelo na računalnim resursima. Upravljačke poruke se, između ostalog, mogu koristiti za mijenjanje frekvencije očitanja određenih senzora ili za slanje ažuriranja prema uređajima nižih slojeva.

### **6.1.3 Pouzdano prikupljanje podataka**

Uobičajeno je da se podaci prosljeđuju od izvora u nižim slojevima prema višim slojevima. Tijekom prijenosa tih podataka napadač ima mogućnost presretanja podataka i prosljeđivanja krivih mjerena prema uređajima viših slojeva. To se sprječava korištenjem HMAC zaštitne sume nad podacima koji se prosljeđuju. Time je osigurana cjelovitost razmijenjenih poruka i omogućena pouzdanost arhitekture. Pouzdani tok podataka za ovaj scenarij prikazan je na slici 6.1c. Korištenje HMAC zaštitne sume umjesto digitalnog potpisa omogućava značajno smanjenje složenosti operacija bez smanjivanja potrebne razine sigurnosti (oba postupka omogućuju ostvarivanje sigurnosnog zahtjeva cjelovitosti podataka). S obzirom na to da uređaji u sklopu senzorskog sloja imaju pristup ograničenom izvoru napajanja, cilj je uštedjeti na potrošnji energije prilikom zaštite podataka. HMAC zaštita koristi se za prikupljanje podataka koji nisu osjetljivi (npr. mjerena kvalitete zraka ili vode), ali bi trebali biti vjerodostojni kako bi se pružala kvalitetna usluga.

### **6.1.4 Zaštita osjetljivih podataka**

U određenim scenarijima prikupljeni podaci mogu biti osjetljive prirode i trebaju se zaštiti tako da sprječavaju curenje podataka kao što je prikazano na slici 6.1d. Potreba za zaštitom podataka može se pojaviti u komunikaciji između svih slojeva u sklopu arhitekture. Iako senzori najčešće prikupljaju neosjetljive podatke, određene primjene, poput sustava za praćenje tjelesnog zdravlja, ne bi smjele dozvoljavati uvid u izmjerene podatke. Isto tako, određene korporativne primjene zahtijevaju da svi podaci unutar arhitekture budu tajni kako se ne bi otkrivali osjetljivi podaci o korisnicima sustava. Zaštita osjetljivih podataka postiže se simetričnom kriptografijom, odnosno šifriranjem podataka na izvoru i dešifriranjem na odredištu korištenjem prethodno razmijenjenog tajnog ključa.

## 6.2 Model sigurne komunikacije u okolini Interneta stvari

Model sigurne komunikacije u okolini Interneta stvari definiran je kao šestorka  $S_{IoT}$ :

$$S_{IoT} = \{D, K, L, C, A, O\}, \quad (6.1)$$

gdje je  $D$  skup svih uređaja (engl. *devices*),  $K$  skup svih parova ključeva (privatnog i odgovarajućeg javnog ključa) svih uređaja,  $L$  je skup svih komunikacijskih slojeva (engl. *communication layers*) na kojem uređaji mogu komunicirati,  $C$  skup svih mogućnosti uređaja (engl. *capabilities*),  $A$  je skup kriptografskih algoritama koje uređaji mogu koristiti i  $O$  je skup operacija s kojima se ostvaruje sigurna komunikacija između uređaja.

Za potrebe ovog modela razlikujemo tri različite vrste kriptografskih algoritama u sklopu skupa  $A = \{A^h, A^s, A^p\}$ : algoritme kriptografskog sažetka  $A^h$ , simetrične algoritme  $A^s$  i asimetrične algoritme  $A^p$ .

Uređaj  $D_i$  u modelu je označen sa svojim skupom podržanih komunikacijskih slojeva  $L_i$ , skupom hardverskih mogućnosti  $C_i$ , skupom podržanih kriptografskih algoritama  $A_i$  i skupom parova javnih i privatnih ključeva  $K_i$ :

$$D_i \in D : \{D_i = \{L_i, C_i, A_i, K_i\}, L_i \subseteq L, C_i \subset C, A_i \subseteq A, K_i \subset K\}, \quad (6.2)$$

gdje je  $A_i = \{A_i^h, A_i^s, A_i^p\}$  takav da vrijedi  $A_i^h \subseteq A^h$ ,  $A_i^s \subseteq A^s$  i  $A_i^p \subseteq A^p$ .

Svaki par ključeva  $k(a_i^p) \in K_i$  stvara se tako da odgovara asimetričnom algoritmu  $a_i^p \in A_i^p$ . Par ključeva  $k(a_i^p)$  sadrži privatni ključ  $pri(a_i^p)$  i njemu odgovarajući javni ključ  $pub(a_i^p)$ :

$$k(a_i^p) = \{pri(a_i^p), pub(a_i^p)\}. \quad (6.3)$$

Skup javnih ključeva za svaki uređaj mora se prebaciti na drugi uređaj prilikom uspostave sigurne komunikacije. Skup javnih ključeva  $PK_i$  za uređaj  $D_i$  definiran je kao:

$$PK_i = \{pub(a_i^p) \in k(a_i^p) : k(a_i^p) \in K_i\}. \quad (6.4)$$

U predloženom modelu, dva uređaja  $D_i$  i  $D_j$  mogu komunicirati samo ako imaju zajednički komunikacijski sloj na kojem mogu komunicirati, odnosno ako vrijedi sljedeće:  $L_i \cap L_j \neq \emptyset$ . Komunikacijski sloj definira vrstu sučelja i mrežne protokole koji se mogu koristiti za razmjenu podataka između uređaja (npr. WLAN, Ethernet, IP, TCP, UDP).

Za ostvarivanje sigurne komunikacije između uređaja u predloženom modelu definira se skup operacija  $O$  koji se sastoji od šest operacija namijenjenih sigurnoj komunikaciji koje se koriste za dogovor algoritama i ključeva, osiguravanje cjelovitosti podataka, autentifikaciju i tajnost podataka.

$$O = \{dogovor, potpis, provjera\_potpisa, hmac, sifriranje, desifriranje\} \quad (6.5)$$

Operacija dogovora kriptografskih algoritama i zajedničke tajne (*dogovor*) je preduvjet za daljnju sigurnu komunikaciju između klijenta (započinjatelja)  $D_i$  i poslužitelja (odgovaratelja)  $D_j$  i mora biti provedena prije svih drugih operacija. Ova operacija dogovora predstavlja protokol ACAP koji služi za dogovor oko preduvjeta sigurne komunikacije. Rezultat operacije je trojka kriptografskih algoritama ( $\tilde{A}_{ij}$ ) koja se može koristiti na oba uređaja, zajednička tajna ( $S_{ij}$ ) koju se može koristiti za stvaranje novih tajnih ključeva za zaštitu podataka i javni ključ svakog uređaja:

$$dogovor : (D_i, D_j) \longmapsto \{\tilde{A}_{ij}, S_{ij}, pub(a_i^p), pub(a_j^p)\}, \quad (6.6)$$

gdje vrijedi da je  $\tilde{A}_{ij} = \{\tilde{A}_{ij}^h, \tilde{A}_{ij}^s, \tilde{A}_{ij}^p\}$  takav da je  $\tilde{A}_{ij}^h \in A_i^h \cap A_j^h$ ,  $\tilde{A}_{ij}^s \in A_i^s \cap A_j^s$  i  $\tilde{A}_{ij}^p \in A_i^p \cap A_j^p$ .

Drugim riječima, dogovoreni algoritmi kriptografskog sažetka, simetrične i asimetrične kriptografije moraju biti podržani na oba uređaja. Javni ključevi  $pub(a_i^p)$  i  $pub(a_j^p)$  odgovaraju dogovorenom algoritmu asimetrične kriptografije  $\tilde{A}_{ij}^p$  tako da vrijedi  $a_i^p = a_j^p = \tilde{A}_{ij}^p$ . Do zajedničke tajne  $S_{ij}$  dolazi se s pomoću Diffie-Hellman izračuna opisanog u poglavlju 2.4.3.

U modelu definirane su sljedeće sigurne komunikacijske operacije koje se mogu izvoditi samo nakon uspješne operacije dogovora:

- Zaštita cjelovitosti podataka uz neporecivost (*potpis*) u obliku digitalnog potpisa za podatke koje se razmjenjuju između pošiljatelja  $D_i$  i primatelja  $D_j$ :

$$potpis : [podaci, \tilde{A}_{ij}^h, \tilde{A}_{ij}^p, pri(a_i^p)] \longmapsto \{digitalni\_potpis\}. \quad (6.7)$$

Digitalni potpis računa se iz podataka kroz izračun kriptografskog sažetka dogovorenim algoritmom  $\tilde{A}_{ij}^h$  te asimetričnom operacijom potpisivanja korištenjem algoritma  $a_i^p = \tilde{A}_{ij}^p$  i odgovarajućeg privatnog ključa  $pri(a_i^p)$ . Uređaj  $D_j$  po primitku podataka provjerava valjanost digitalnog potpisa korištenjem operacije *provjera\_potpisa*:

$$provjera\_potpisa : [\{podaci, potpis\}, \tilde{A}_{ij}^h, \tilde{A}_{ij}^p, pub(a_i^p)] \longmapsto [ispravan, neispravan] \quad (6.8)$$

Primatelj  $D_j$  provjerava dobiveni digitalni potpis tako da uspoređuje rezultat kriptografskog sažetka s algoritmom  $\tilde{A}_{ij}^h$  nad dobivenim podacima, s dešifriranim kriptografskim sažetkom iz digitalnog potpisa algoritmom  $a_j^p = \tilde{A}_{ij}^p$  i odgovarajućim javnim ključem  $pub(a_i^p)$ . Ako se sažeci podudaraju potpis je ispravan, u suprotnom potpis je neispravan.

- Autentifikacija podataka i zaštita cjelovitosti algoritmom HMAC (*hmac*) koristi se kao jednostavniji mehanizam zaštite koji zahtijeva manje resursa od digitalnog potpisa. Operacija *hmac* prilikom slanja podataka *data* između pošiljatelja  $D_i$  i primatelja  $D_j$  definirana je na sljedeći način:

$$hmac : (podaci, \tilde{A}_{ij}^h, S_{ij}) \longmapsto \{zaštitna\_suma\} \quad (6.9)$$

Zaštitna suma se računa korištenjem dogovorenog algoritma kriptografskog sažetka  $\tilde{A}_{ij}^h$  s dogovorenom zajedničkom tajnom  $S_{ij}$  nad podacima koji se šalju. Primatelj  $D_j$  uspoređuje dobivenu zaštitnu sumu sa zaštitnom sumom izračunatom nad dobivenim podacima s istim ključem. Provjera zaštitne sume je uspješna ako se primljena i izračunata zaštitna suma podudaraju.

- Tajnost podataka korištenjem operacija *sifriranje* i *desifriranje* za podatke razmijenjene između pošiljatelja  $D_i$  i primatelja  $D_j$ . Operacija šifriranja koristi prethodno dogovoren algoritam simetrične kriptografije  $\tilde{A}_{ij}^s$  i zajedničku tajnu  $S_{ij}$  nad podacima:

$$\text{sifriranje} : (\text{podaci}, \tilde{A}_{ij}^s, S_{ij}) \longmapsto \text{tajni\_podaci} \quad (6.10)$$

Izvorni podaci se izračunavaju s operacijom dešifriranja uz korištenje istog algoritma i zajedničke tajne:

$$\text{desifriranje} : (\text{tajni\_podaci}, \tilde{A}_{ij}^s, S_{ij}) \longmapsto \text{podaci} \quad (6.11)$$

Predstavljeni model sigurne komunikacije je minimalan skup objekata i operacija koje su potrebne za sigurnu komunikaciju između svih uređaja unutar okoline Interneta stvari neovisno o njihovim mogućnostima.

### 6.3 Prilagodljivost modela sigurne komunikacije

Predloženi model je prilagodljiv jer podržava različite načine dogovora korištenih algoritama ovisno o trenutnim mogućnostima uređaja  $D_i$  i  $D_j$ . Za odabir algoritama koristi se poopćenje procedure koja je definirana u poglavlju 4.4, a temeljeno je na protokolu SSH 3.1.3. Protokol SSH uzima u obzir samo redoslijed algoritama u popisu, odnosno prepostavlja da će algoritmi biti poredani po sigurnosti ili brzini izvođenja. Osnovni nedostatak tog pristupa je nemogućnost dinamičke promjene uvjeta odabira.

Za područje Interneta stvari prikidan je unaprijeđeni algoritam dogovora koji je prikazan u algoritmu 1, a omogućuje dinamičko odlučivanje oko algoritma koji će se koristiti ovisno o trenutnim sposobnostima komunicirajućih strana. Osnova za vrednovanje algoritama leži u odlučivanju koji je algoritam primjenjeni za upotrebu ovisno o uvjetima u kojima se odvija sigurna komunikacija. Vrijednost algoritma može biti povezana izravno s mjerljivim mogućnostima komunicirajućih uređaja, poput računalne moći, propusnosti komunikacije ili pristupa izvoru napajanja.

Algoritam uzima u obzir mogućnosti klijenta i poslužitelja  $C_i$  i  $C_j$  te potrebnu razinu sigurnosti  $rsig$  kako bi iz popisa podržanih algoritama  $A_i$  i  $A_j$  odabrao prikladne algoritme ovisno o trenutnim mogućnostima (linija 1).

Procedura dogovora prolazi kroz sve vrste algoritama koje su potrebne za osiguravanje dalj-

**Algoritam 1** Procedura za dogovor kriptografskih algoritama

---

```
1: procedure CRYPTO-AGREEMENT( $C_i, C_j, A_i, A_j, rsig$ )
2:   for svaka  $vrsta \in \{h, s, p\}$  do
3:     for svaki  $a^{vrsta} \in (A_i^{vrsta} \cap A_j^{vrsta})$  do
4:       if ( $\tilde{A}_{ij}^{vrsta} == \emptyset$ ) or ( $score(a^{vrsta}, C_i, C_j, rsig) > score(\tilde{A}_{ij}^{vrsta}, C_i, C_j, rsig)$ ) then
5:          $\tilde{A}_{ij}^{vrsta} = a^{vrsta}$ 
6:         break
7:       end if
8:     end for
9:     if  $\tilde{A}_{ij}^{vrsta} == \emptyset$  then
10:      return  $\emptyset$             $\triangleright$  Dogovor je neuspješan jer nema zajedničkih algoritama.
11:    end if
12:  end for
13:  return  $\tilde{A}_{ij} = \{\tilde{A}_{ij}^h, \tilde{A}_{ij}^s, \tilde{A}_{ij}^p\}$ 
14: end procedure
```

---

nje komunikacije (hash -  $h$ , simetrični -  $s$ , asimetrični -  $p$ ) (linija 2-12) i za svaku vrstu odabire algoritam  $a^{vrsta}$  iz presjeka algoritama obje strane ( $A_i^{vrsta} \cap A_j^{vrsta}$ ). Odabire se algoritam koji ima najveću vrijednost u odnosu na ostale algoritme (linije 3-8). Vrijednost pojedinog algoritma određuje se s pomoću funkcije za računanje vrijednosti  $score(a^{vrsta}, C_i, C_j, rsig)$  (linija 4). Konačno, procedura vraća n-torku odabranih algoritama (linija 13) ili prazan skup (linija 10) koji predstavlja neuspješan dogovor. Predložena procedura za dogovor je ostvarena uz pretpostavku da je razina sigurnosti najvažniji parametar. U tom slučaju se ova procedura može poistovjetiti s procedurom za odabir u protokolu SSH.

Prilikom postavljanja cjelokupne okoline za sigurnu komunikaciju potrebno je izmjeriti podatke o mogućnostima uređaja koji sudjeluju u komunikaciji i prema tome odrediti parametre za funkciju vrednovanja  $score$ . Funkcija  $score$  za vrednovanje algoritama omogućuje fleksibilno definiranje vrijednosti ovisno o pojedinoj primjeni. U okolini Interneta stvari važno je uzeti u obzir trenutne mogućnosti uređaja i prema tome odrediti koji je algoritam najbolji, a koji najgori u danom trenutku. Ukupna vrijednost algoritma  $a$ , koja je definirana izrazom  $score(a, C_i, C_j, rsig)$ , zapravo je kombinacija vrijednosti u više dimenzija. Osnovne dimenzije dolaze od trenutno zahtjevane razine sigurnosti  $rsig$ , računalne zahtjevnosti algoritma  $ac$  (engl. *algorithm complexity*) i trenutnih mogućnosti oba uređaja koje obuhvaćaju trenutnu razinu napajanja  $bp$  (engl. *battery power*), propusnosti komunikacijske veze  $bw$  (engl. *bandwidth*) i računalne snage uređaja  $pp$  (engl. *processing power*):

$$\begin{aligned} score(a, C_i, C_j, rsig) = \\ f[w_0 \cdot rsig, w_1 \cdot ac(a), w_2 \cdot min(bp_i, bp_j), w_3 \cdot min(bw_i, bw_j), w_4 \cdot min(pp_i, pp_j)], \quad (6.12) \end{aligned}$$

gdje su  $C_i$  i  $C_j$  redom definirani kao  $\{bp_i, bw_i, pp_i\}$  i  $\{bp_j, bw_j, pp_j\}$ . Različite težine  $w_x$  ukažuju na relativnu važnost i definiraju doprinos određene dimenzije u odnosu na cjelokupnu

vrijednost algoritma. Odnosi težina se određuju na temelju trenutne primjene i okoline u kojoj je sustav postavljen. Prioritet se daje uređaju koji trenutno ima manju razinu mogućnosti (funkcija min).

### 6.3.1 Prilagodbe odabira algoritma

Funkcija za vrednovanje *score* omogućuje prilagodljivost procedure za dogovor na razne uvjete komunikacije ovisno o namjeni i ograničenjima sustava. Primjerice:

- Ako se radi o sustavu koji prikuplja i obrađuje velike količine podataka, najveća težina će se postaviti na propusnost komunikacijske veze  $bw_x$  i na računalnu zahtjevnost algoritma  $ac_x$  kako se podaci mogli sigurno prikupljati zavodjavajućom brzinom.
- U slučaju postavljanja sustava za dugoročno prikupljanje, težina će biti postavljena na trenutnu razinu napajanja  $bw_x$ .
- Kod postavljanja sustava s visokim zahtjevima na sigurnost, težina će se postaviti na razinu sigurnosti komunikacije  $rsig$ , a drugi ključni parametar će biti računalna snaga uređaja  $pp_x$ .

Prilikom specifikacije funkcije vrednovanja važno je odrediti što je prioritet kod postavljanja sustava. U slučaju da imamo sustav za prikupljanje podataka o temperaturi na određenom području, važno je odrediti kroz koji će se period prikupljati očitanja i procijeniti trošak obrade pojedinog mjerenja koji uključuje mjerenje, zaštitu podataka i slanje podataka. Različiti algoritmi će imati različitu zahtjevnost, odnosno potrošnju električne energije. Stoga je važno pravilno izmjeriti težinu pojedinog algoritma za uređaj koji se postavlja i definirati računalne zahtjevnosti za svaki algoritam.

Ako se za vrijeme rada sustava promijene zahtjevi na sigurnost ili se postave drukčiji prioriteti na težinu određenih parametara, moguće je s viših slojeva ažurirati težine tih parametara i prilagoditi sustav trenutnim potrebama. Na taj način bi se mogla smanjiti učestalost očitanja kako bi se uštedjela energija potrebna za omogućavanje tajnosti očitanih podataka.

Prilikom komunikacije dva uređaja s različitim slojeva, prioritet kod izračunavanja funkcije vrednovanja daje se uređaju koji se nalazi na nižem sloju arhitekture. Na taj se način štede resursi u nižim slojevima arhitekture kako bi se prikupljanje podataka obavilo na što efikasniji način.

## Poglavlje 7

# Programsko ostvarenje protokola ACAP

Programsko ostvarenje ACAP izvedeno je u programskom jeziku Python 3 koji koristi dodatne biblioteke za kriptografiju (pycrypto\* i cryptography†) i biblioteku za prepoznavanje mrežnih sučelja za potrebe rada na sloju Ethernet (netifaces‡). Programsko ostvarenje podržava pokretanje aplikacije u klijentskom i poslužiteljskom načinu rada i kao rezultat vraća sljedeće podatke:

- popis dogovorenih algoritama razvrstanih u kategorije,
- tajni ključ koji se može koristiti za osiguravanje tajnosti korištenjem simetričnih algoritama šifriranja i dešifriranja ili osiguravanje cjelovitosti korištenjem algoritma HMAC i
- javni ključ druge strane koji se može koristiti za provjeru identiteta i osiguravanje cjelovitosti korištenjem digitalnog potpisa.

U ovom poglavlju objašnjen je način rada ostvarenja protokola ACAP kroz pokretanje na različitim mrežnim slojevima i prikaz strukture poruka. Nakon toga dana je analiza performansi uz mjerenja koja prikazuju složenost kriptografskih operacija. Na kraju poglavlja prikazani su ugrađeni sigurnosni mehanizmi. Aktualna verzija programskog ostvarenja uz upute za instalaciju može se dohvatiti sa sljedeće poveznice:

<http://public.tel.fer.hr/acap>

### 7.1 Arhitektura programskog ostvarenja i načini integracije

Protokol ACAP sastoji se od sljedećih sastavnih dijelova:

- izračun i provjera kriptografskih algoritama,
- obrada poruka,
- komunikacija neovisna o sloju,

---

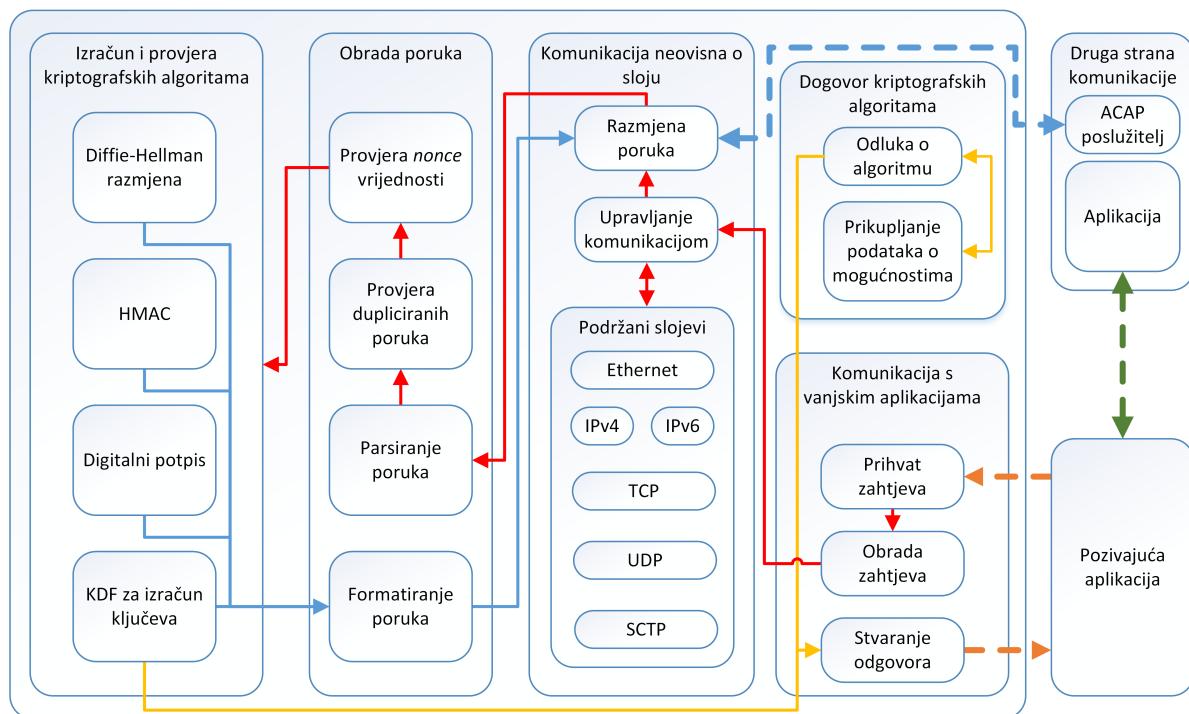
\*<https://www.dlitz.net/software/pycrypto/>

†<https://cryptography.io/en/latest/>

‡<https://pypi.python.org/pypi/netifaces>

- dogovor kriptografskih algoritama i
- komunikacija s vanjskim aplikacijama.

Cjelokupna arhitektura programskog ostvarenja prikazana je na slici 7.1. Nakon što protokol ACAP zaprimi zahtjev za dogовором kriptografskih algoritama kroz sustav za komunikaciju s vanjskim aplikacijama, pokreće se proces dogovora algoritma i ključeva prema ACAP poslužitelju na drugoj strani. U dogovoru se međusobno izmjenjuju sustavi za izračun i provjeru kriptografskih algoritama, obradu poruka te slanje i primanje paketa. Na kraju dogovora pokreće se sustav za dogovor algoritma koji daje popis dogovorenih algoritama, na temelju kojih se generira dogovorena duljina kriptografskog ključa uz pomoć funkcije za generiranje nasumičnih vrijednosti. Na kraju se popis dogovorenih algoritama zajedno s kriptografskim ključevima šalje aplikaciji koja je pokrenula dogovor. Nakon ACAP razmjene pozivajuća aplikacija može sigurno komunicirati korištenjem dogovorenih algoritama i ključeva.



Slika 7.1: Arhitektura programske realizacije protokola ACAP

Programsko ostvarenje protokola ACAP izravno podržava primjenu u sklopu modela klijent-poslužitelj. Primjena u mrežama ravnopravnih čvorova postiže se pokretanjem poslužitelja na svim čvorovima u mreži te se dogovor algoritama i ključeva odvija u parovima čvorova. Na taj način omogućeno je da komunikacija između svaka dva čvora u mreži dobije razinu zaštite komunikacije koja je primjerena mogućnostima tih čvorova. Grupni dogovor ključeva i algoritma između ravnopravnih čvorova nije omogućen jer bi to nužno smanjivalo razinu sigurnosti na mogućnosti najslabijeg čvora i omogućilo napadaču razne napade na sigurnost samog protokola.

## 7.2 Mogućnosti programskog ostvarenja

Prilikom pokretanja ostvarenja moguće je specificirati hoće li se pokrenuti poslužiteljska ili klijentska instanca protokola. U slučaju klijentske strane potrebno je definirati adresu poslužitelja. Nadalje, može se specificirati koji se protokol želi koristiti za prijenos poruka. Moguće je odabratizmeđu protokola TCP, SCTP, UDP, IP i Ethernet. Dodatno, za protokole TCP, SCTP, UDP i IP moguće je koristiti protokol IP verzije 4 i protokol IP verzije 6. Prepostavljene opcije prilikom pokretanja su korištenje protokola TCP s IPv4 adresama za transport, korištenje standardnog Diffie-Hellman postupka i algoritma RSA za digitalno potpisivanje.

Za osnovno pokretanje potrebno je definirati želi li se pokrenuti poslužitelj ili klijent. Za pokretanje poslužitelja koristi se opcija "-S", a za pokretanje klijenta opcija "-C". Potrebno je definirati popis podržanih algoritama s opcijom "-a", u suprotnom će se popis algoritama automatski dohvati iz trenutno dostupnih kriptografskih algoritama u sustavu. Nužno je još definirati privatni ključ s pomoću opcije "-k", koja uzima datoteku s RSA privatnim ključem veličine barem 768 okteta. Iz tog privatnog ključa dobiva se javni ključ koji služi za identifikaciju tijekom razmjene. Rezultati uspješne razmjene prikazani su na slikama 7.2 i 7.3.

Na slici 7.2 vidi se skraćeni ispis pokretanja klijenta kojemu je potrebno dodatno definirati adresu poslužitelja na koju će se spojiti (" -h 10.0.0.20"). Nakon razmjene poruka klijent dobiva javni ključ poslužitelja, popis dogovorenih algoritama i dijeljenu zajedničku tajnu. Na kraju ispisa prikazano je ukupno trajanje dogovora od početka razmjene. Potpuni ispis klijenta sa svim ključevima moguće je vidjeti u dodatku B.1.

```
$ ./acap.py -C -a client.json -k private_key_client -h 10.0.0.20
INFO - Server public key: -----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIB3DQEBAQUAA4GNADCBiQKBgQCTYB3xU17CS5AGxKbWraxuGOYb
7sff8AAjjiwXrdBr6jqJ5zSHtR/bKxsRNPMNvzihVI1Taa7/CQy3EIRKkLeG4I7M
USeStpBfvQ1pAK+lmyUE3haXVUNgPpALdXUpn+IaB+TDHuWl19z9dPDkSDLzQzga
pVwSgXpEZDUInwKK4QIDAQAB -----END PUBLIC KEY-----
INFO - Client negotiated:
{ "hash": {
    "algorithm": "SHA-256",
    "timestamp": "2015-11-22 14:59" },
  "public_key": {
    "algorithm": "RSA_2048",
    "timestamp": "2015-11-22 14:59" },
  "secret_key": {
    "algorithm": "AES-CTR_256",
    "timestamp": "2015-11-22 14:59" } }
INFO - Shared secret key (256 bit): bb:66:cc:98:10:fd:b7:a0:73:9b:9e:35:49:11:
43:7e:e5:28:3d:b3:69:5e:e2:69:4c:c9:a8:11:6d:d5:57:9a
INFO - TOTAL duration: 119820
```

**Slika 7.2:** Primjer pokretanja ACAP klijenta

Na slici 7.3 prikazan je skraćeni ispis sa strane poslužitelja. Jedina značajna razlika vidljiva je u prvoj liniji koja označava da su osvježene vrijednosti Diffie-Hellman eksponenta i

poslužiteljski *nonce* ("Refreshed DH and Nonce..."). Taj se proces kod poslužitelja odvija u pozadini i služi za smanjenje utjecaja DoS napada na protokol ACAP na način opisan u poglavlju 4.5. Potpuni prikaz poslužitelja sa svim ključevima prikazan je u dodatku B.2.

```
$ ./acap.py -S -a server.json -k private_key_server
Refreshed DH and Nonce...
INFO - Client public key: -----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC4TDSclGnCkXgP0C8mt889rBR1
iFoXBt01G4VF1XxhxursgU+Es2PCqRUYT28ZqdNrq2CAyuIT33G5CXCwg8x/CBh2
qmFJ43NpauGg13b2LFNSg3j8UxxwGSGIvKvx0maGnSJByepogXkWuY0bL5mR0n01
JPH/im05V2+Jox951QIDAQAB -----END PUBLIC KEY-----
INFO - Server negotiated:
{
    "hash": {
        "algorithm": "SHA-256",
        "timestamp": "2015-11-22 14:59" },
    "public_key": {
        "algorithm": "RSA_2048",
        "timestamp": "2015-11-22 14:59" },
    "secret_key": {
        "algorithm": "AES-CTR_256",
        "timestamp": "2015-11-22 14:59" } }
INFO - Shared secret key (256 bit): bb:66:cc:98:10:fd:b7:a0:73:9b:9e:35:49:11
:43:7e:e5:28:3d:b3:69:5e:e2:69:4c:c9:a8:11:6d:d5:57:9a
INFO - TOTAL duration: 17757
```

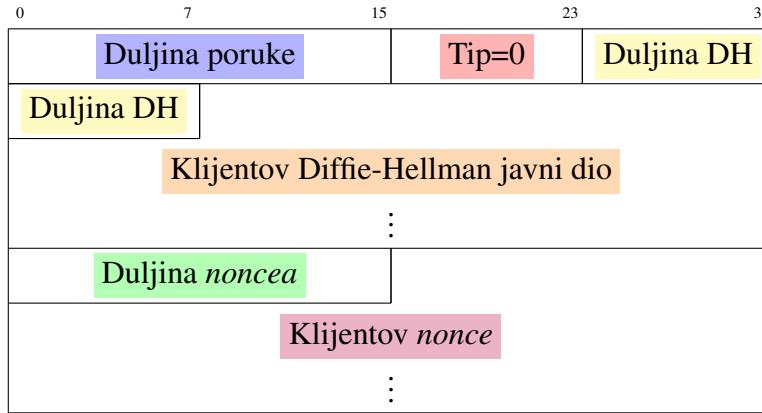
Slika 7.3: Primjer pokretanja ACAP poslužitelja

Prilikom pokretanja moguće je odabrati kriptografske algoritme koji se zasnivaju na korištenju eliptičkih krivulja. To se radi s pomoću opcije "-E" koja specificira da se umjesto standardnog Diffie-Hellman postupka koristi ECDH (engl. *Elliptic Curve Diffie-Hellman*) te da se umjesto algoritma RSA koristi algoritam ECDSA za digitalno potpisivanje. Tada se umjesto privatnog RSA ključa mora specificirati ECDSA privatni ključ.

### 7.3 Struktura poruka

Poruke koje se razmjenjuju u protokolu ACAP prikazane su u poglavlju 4.2. S obzirom na to da su osnovni dijelovi tih poruka varijabilne duljine, prije svakog od tih dijelova dodano je polje koje označava duljinu parametra. Bitovni zapis poruke INIT<sub>I</sub> prikazan je na slici 7.4. Za definiranje svih duljina koriste se dva okteta koja sadrže duljinu polja izraženu u broju oktetova. Formati ostalih poruka nalaze se u dodatku C.

Na slici 7.5 dan je bitovni prikaz jedne poruke INIT<sub>I</sub> koji je obojan u skladu s bitovnim zapisom iz slike 7.4 kako bi se lakše identificirala polja poruke. Na početku se nalazi duljina poruke u oktetima (0x0091=145), potom je naveden tip poruke (0x00), a zatim duljina DH javnog dijela klijenta (0x0080=128) i vrijednost DH javnog dijela. Na samom kraju poruke zapisana je duljina klijentovog *noncea* (0x000c=12) i vrijednost *noncea*. Cjelokupna duljina poruke jednaka je zbroju duljina svih navedenih dijelova ( $1 + 2 + 128 + 2 + 12 = 145$ ).

**Slika 7.4:** Bitovni zapis poruke INIT<sub>I</sub>

```

0x00: 00 91 00 00 80 27 2d d0 e9 6a f9 d7 37 2e a1 c8
0x10: 6c 23 0a c9 44 98 13 c6 ed 27 fc ac b8 3c 91 dc
0x20: 82 4a 1d f0 af 47 5a 87 63 42 24 0b e4 2b e1 0f
0x30: e0 1f fa 54 c6 14 5d 4f bf 13 c6 76 70 5f 65 06
0x40: 8d 74 c6 1f 9f 1f 00 dd 71 55 65 d3 09 44 da 59
0x50: cf 37 74 e1 f1 b3 38 55 38 51 20 b9 25 c8 68 b0
0x60: 03 cd e0 dd 97 34 ab 99 3e 75 36 c2 8f 78 fd b9
0x70: 9e a5 7e df bc ee ea 50 d0 f0 8e ce 4d 02 e8 75
0x80: b5 9b 5e 62 d7 00 0c c7 48 6b 13 6f d8 16 e0 00
0x90: 15 ce 6b

```

**Slika 7.5:** Bitovni prikaz primjera poruke INIT<sub>I</sub>

Formati poruka i njihov bitovni zapis u potpunosti je neovisan o transportnom protokolu. Transportni protokol određuje isključivo kako će se popunjavati zaglavlja i na koji će se način protokol ACAP razlikovati da bi se mogao obrađivati neovisno o drugom prometu.

## 7.4 Korištenje protokola na različitim komunikacijskim slojevima

Kako bi se mogli koristiti različiti komunikacijski protokoli za prijenos podataka uvedeni su posebni mehanizmi. Ti se mehanizmi dijele u dvije osnovne skupine: mehanizmi kod pouzdanog prijenosa podataka i mehanizmi kod nepouzdanog prijenosa podataka.

Pouzdani prijenos podataka obuhvaća prije svega protokol TCP [59]. S obzirom na to da protokol TCP za prijenos koristi tokove podataka, a protokol ACAP komunikaciju porukama, svakoj poruci potrebno je dodati duljinu poruke kako bi se mogla izdvajiti iz toka podataka.

Nepouzdan prijenos podataka odnosi se na prijenos s pomoću protokola UDP [60], IPv4 [61], IPv6 [62] Ethernet (IEEE 802.3) i sličnim. Prijenos s pomoću tih protokola odvija se porukama. Kako se radi o nepouzdanom prijenosu, potrebno je detektirati gubitak poruka i obaviti ponovno slanje poruke u slučaju gubitka. Taj problem se rješava na sličan način kao kod pro-

tokola DTLS [23], korištenjem brojila (*expire timer*). Brojilo počinje odbrojavati nakon što se pošalje poruka. Ako brojilo odbroji do kraja i ne primi povratnu poruku, ponovno šalje istu poruku. Ako se poruka izgubi tri puta uzastopno, dogovor se prekida. Uz to, potrebno je riješiti problem duplicitiranja datagrama na putu do odredišta. To se rješava prepoznavanjem kopija poruka i njihovim odbacivanjem u okviru određenog vremenskog prozora. Za svaku dolaznu poruku računa se kriptografski sažetak i uspoređuje se sa sažecima prijašnjih poruka. Ako sažetak postoji u privremenoj bazi sažetaka poruka se odbacuje, a u suprotnom se sažetak poruke spremi u privremenu bazu sažetaka.

Kod korištenja datagramskog prijenosa podataka potrebno je obratiti pozornost na to da poruka svojom veličinom ne prijeđe veličinu određenu najmanjom vrijednošću MTU-a (*Maximum Transmission Unit*) na putu do odredišta [63]. Kod korištenja protokola UDP i IP to je nužno da bi se spriječila fragmentacija datagrama u više paketa što bi povećalo mogućnost gubitka paketa. Protokolom ACAP moguće je poslati pakete proizvoljne veličine, ali u slučaju korištenja protokola Ethernet kao transportnog protokola, najveća veličina okvira može biti 1500 okteta, kako bi se izbjegla nekompatibilnost s postojećim mrežama.

#### 7.4.1 Protokol Ethernet

Komunikacija na sloju podatkovne poveznice protokolom Ethernet specificira se korištenjem opcije "`-e`" uz specifikaciju mrežnog sučelja preko kojeg se želi komunicirati. Primjeri poziva za klijenta i poslužitelja:

```
# ./acap.py -C -a client.json -k private_key_client -e eth0 -h 00:01:29:f0:83:a4
# ./acap.py -S -a server.json -k private_key_server -e eth0
```

Ethernet zaglavljje za slanje poruke INIT<sub>I</sub> bit će definirano na sljedeći način:

0x00: 00 01 29 f0 83 a4 74 d0 2b 93 24 f8 08 13

Prve dvije vrijednosti označavaju redom MAC adresu poslužitelja i klijenta, a zadnja vrijednost specificira identifikator protokola ACAP (0x0813) na Ethernet sloju. Podaci koje prenosi Protokol Ethernet za poruku INIT<sub>I</sub> formatirani su kao na slici 7.4.

#### 7.4.2 Protokoli IP

Ako se podaci u sklopu razmjene poruka prenose protokolom IP ili višim, moguće je odabrati želi li se koristiti protokol IPv4 ili IPv6. Pretpostavljena opcija je korištenje protokola IPv4 ("`-4`"), a korištenje protokola IPv6 definira se korištenjem opcije "`-6`". Primjeri poziva za klijenta i za poslužitelja za protokol IPv4:

```
# ./acap.py -C -a client.json -k private_key_client -i 10.0.0.21
# ./acap.py -S -a server.json -k private_key_server -i
```

IPv4 zaglavljje za slanje početne poruke definirano je na sljedeći način:

0x00: 45 00 00 a7 d6 8d 40 00 40 b7 4e ea 0a 00 00 14  
0x10: 0a 00 00 15

Prva naznačena vrijednost predstavlja verziju protokola IP (4), a druga vrijednost je identifikator protokola ACAP kojeg prenosi protokol IPv4 (0xb7=183). Posljednje dvije vrijednosti su IP adrese klijenta (0x0a 0x00 0x00 0x14=10.0.0.20) i poslužitelja (0x0a 0x00 0x00 0x15=10.0.0.21).

Primjeri poziva za klijenta i poslužitelja za protokol IPv6:

```
# ./acap.py -C -a client.json -k private_key_client -6 -i -h fc00::21  
# ./acap.py -S -a server.json -k private_key_server -6 -i
```

Za dani primjer poziva IPv6 zaglavlje će biti sljedeće:

0x00: 60 00 00 00 00 93 b7 40 fc 00 00 00 00 00 00 00  
0x10: 00 00 00 00 00 00 00 20 fc 00 00 00 00 00 00 00  
0x20: 00 00 00 00 00 00 00 21

Prva obojana vrijednost je verzija protokola IP (6), a druga vrijednost je identifikator protokola ACAP (0xb7=183). Posljednje dvije vrijednosti su IP adrese klijenta (fc00::20) i poslužitelja (fc00::21).

### 7.4.3 Transportni protokoli

U programskom ostvarenju trenutno su podržana tri transportna protokola: TCP, UDP i SCTP. Kod sva tri protokola poslužitelji slušaju na vratima 13000 za dogovor putem protokola ACAP. Pretpostavljena vrijednost je korištenje protokola TCP ("‐t"). Korištenje protokola UDP definira se opcijom "‐u", a korištenje protokola SCTP opcijom "‐s". Definiranje adrese klijenta ovisi o tome koristi li se protokol IPv4 ili IPv6.

Za cjelokupnu razmjenu protokolu TCP potrebno je 12 IP paketa, protokolu UDP 4 paketa, a protokolu SCTP 13 paketa. Zbog početnih kontrolnih poruka koje su nužne za uspostavu veze kod protokola TCP i SCTP te veće složenosti zaglavlja prikazano je samo zaglavlje prilikom korištenja protokola UDP. Zaglavlje kod korištenja protokola UDP za prvu poruku je sljedeće:

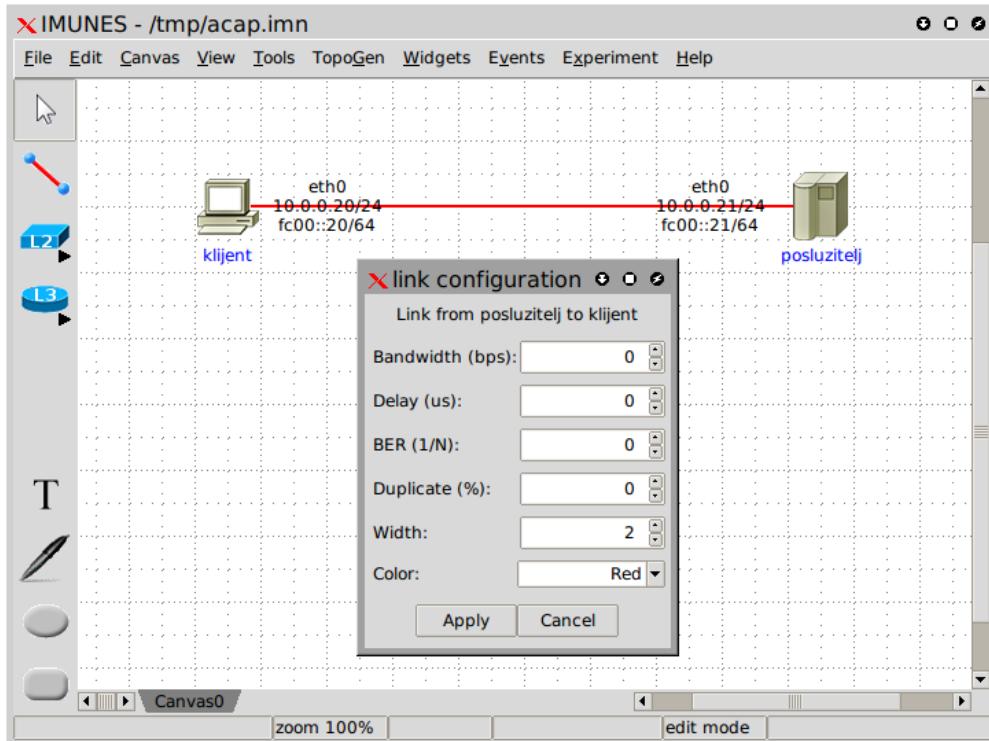
0x00: b8 d2 32 c8 00 9b fe ae

Prva obojana vrijednost označava odlazna vrata koja se automatski dodjeljuju od strane operacijskog sustava (0xb8d2=47314), a druga vrijednost označava vrata na kojima sluša ACAP poslužitelj (0x32c8=13000).

## 7.5 Mjerenje performansi protokola u emuliranoj mrežnoj okolini

Mjerenje performansi protokola izvedeno je u emuliranoj mrežnoj okolini IMUNES. Emulirana mrežna okolina predstavlja mrežnu okolinu koja se izvodi na jednom ili više računala u stvarnom vremenu i uz standardan način obrade paketa. IMUNES<sup>§</sup> je mrežni emulator koji može na vrlo efikasan način izvršavati emulirane mrežne okoline od 100 i više mrežnih uređaja na jednom računalu [64]. Paketi se u IMUNES emuliranoj mrežnoj okolini obrađuju na način na koji je to definirano u sklopu FreeBSD ili Linux jezgre operacijskog sustava. U toj emuliranoj mrežnoj okolini moguće je definirati razne smetnje na poveznicama između čvorova kako bi se lakše ispitala robusnost programskog ostvarenja.

Za potrebe testiranja i mjerenja protokola ACAP koristi se jednostavna emulirana mrežna topologija koja je prikazana na slici 7.6. Topologija se sastoji od dva čvora, jedan za klijenta, a drugi za poslužitelja. Na poveznici između ta dva čvora moguće je mijenjati propusnost poveznice (engl. *bandwidth*), kašnjenje na poveznici (engl. *delay*), greške na poveznici (BER, engl. *bit-error rate*) i unositi smetnje u obliku udvostručavanja paketa (engl. *duplicate*). Prozor za promjenu parametara prikazan je na slici 7.6, a pokreće se desnim klikom na link i odabirom opcije *Configure*.



Slika 7.6: Emulirana mrežna topologija za mjerenja u alatu IMUNES

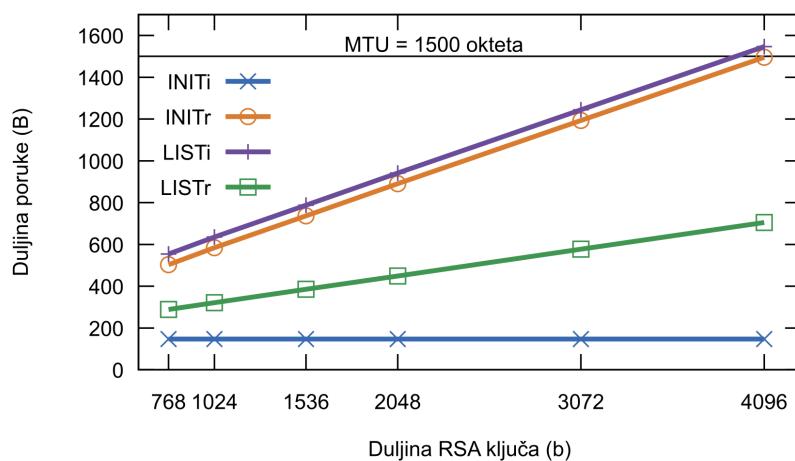
Sva opisana mjerenja izvedena su na priloženoj topologiji uz promjenu parametara na po-

<sup>§</sup><http://www.imunes.net>

veznici. Mjerenje kašnjenja izvedeno je u 50 iteracija te je konačni prosjek prikazan pomoću grafikona i tablica.

### 7.5.1 Utjecaj veličine javnih ključeva na veličinu poruka

Duljina javnog ključa izravno utječe na veličinu digitalnog potpisa koji se koristi za zaštitu poruka. S druge strane, duljina ključa povećava razinu sigurnosti razmjene, ali ujedno i usporava računanje i verifikaciju digitalnih potpisa. Kako se razvija i napreduje računalna moć tako se javlja potreba za sve većom razinom sigurnosti. Na slici 7.7 prikazane su veličine poruka za različite veličine RSA ključeva. Duljine poruka izražene su u oktetima, dok su duljine ključeva prikazane na standardan način, u bitovima. Može se zamijetiti kako poruka  $\text{INIT}_I$  uopće ne ovisi o veličini javnog ključa jer ne sadrži niti javni ključ niti digitalni potpis. Poruka  $\text{LIST}_R$  sadrži samo digitalni potpis, a poruke  $\text{INIT}_R$  i  $\text{LIST}_I$  sadrže i digitalni potpis i javni ključ što je vidljivo iz slike jer crte koje ih označuju najbrže rastu.

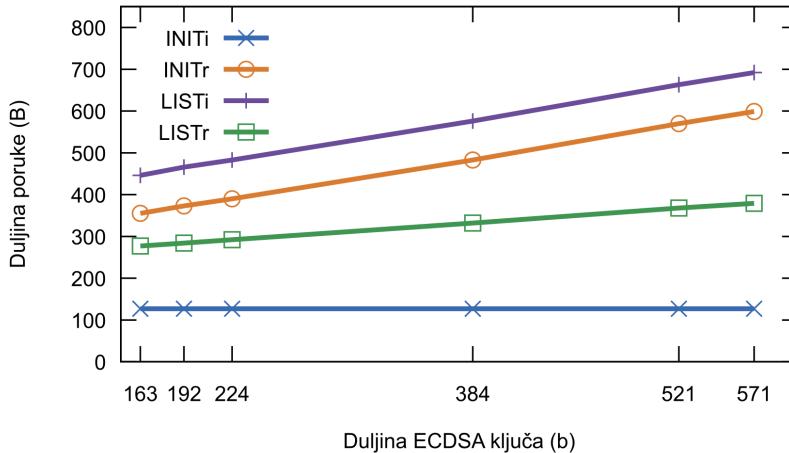


Slika 7.7: Veličina poruka ovisno o veličini RSA javnih ključeva

Protokol Ethernet standardno podržava prijenos od najviše 1500 okteta podataka, što je definirano najvećom jedinicom prijenosa (engl. *Maximum Transmission Unit* - MTU). Povećanje veličine RSA ključeva ili veliki popis podržanih algoritama mogli bi onemogućiti korištenje protokola na sloju Ethernet, odnosno zahtijevalo bi se korištenje više poruka za razmjenu cijelog popisa, što nije predviđeno specifikacijom i ostvarenjem protokola. Uz to, cilj je smanjiti veličinu poruka kako bi se povećala efikasnost i smanjila propusnost potrebna za razmjenu poruka i ubrzao sam dogovor.

Smanjenje veličine poruka moguće je uvođenjem kriptografije koja se zasniva na eliptičnim krivuljama i koristi manju veličinu javnih ključeva i digitalnih potpisa. Mjerenja veličine poruka za različite veličine ECDSA ključeva prikazana su na slici 7.8.

Korištenjem ECDSA ključeva značajno se smanjuje veličina poruka protokola ACAP i smanjuje potrebna propusnost za protokol. Važno je napomenuti da je sigurnost koju pruža 224 bitni

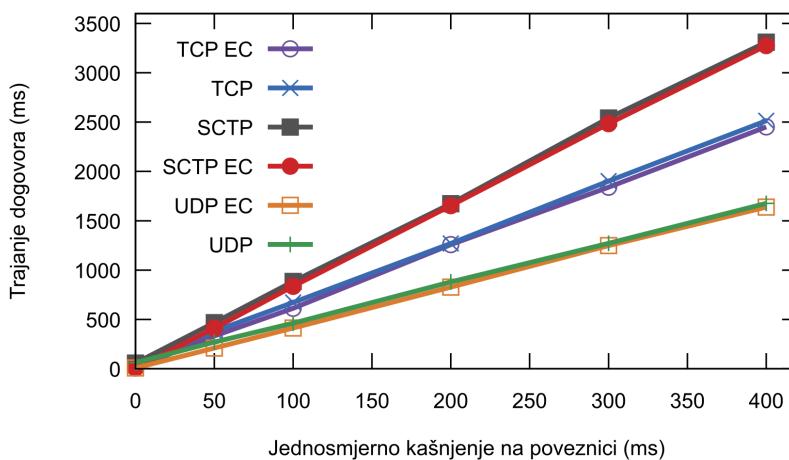


Slika 7.8: Veličina poruka ovisno o veličini ECDSA javnih ključeva

ECDSA ključ usporediva sa sigurnošću koju pruža 2048 bitni RSA ključ. Stoga, uvođenje kriptografije zasnovane na eliptičnim krivuljama smanjuje potrebnu propusnost i ujedno povećava razinu sigurnosti.

### 7.5.2 Trajanje dogovora ovisno o kašnjenju

Trajanje dogovora mjereno je na strani klijenta tako da se jednosmjerno kašnjenje mijenjalo od 0 ms sve do 400 ms, što predstavlja maksimalno kašnjenje u oba smjera od 800 ms. Mjerenja su izvršena za sve transportne protokole, ali razlike između protokola UDP, IPv4, IPv6 i Ethernet su premale i ne daju nikakav dodatan uvid u mjerjenja. Na slici 7.9 prikazana su mjerena kašnjenja za protokole UDP, SCTP i TCP uz korištenje standardne kriptografije i uz korištenje kriptografije zasnovane na eliptičkim krivuljama.



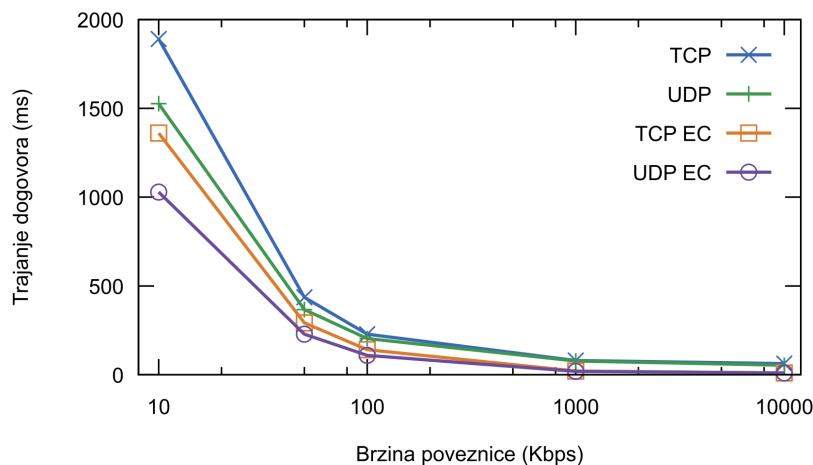
Slika 7.9: Trajanje dogovora ovisno o kašnjenju, protokolu i korištenim algoritmima

Prije svega, mogu se zamijetiti razlike između protokola UDP, SCTP i TCP koje su ovisne o uspostavi veze kod tih protokola. UDP nema nikakve uspostave, protokolu SCTP su potrebne

4 poruke, a protokolu TCP 3 poruke za uspostavu veze. Slika 7.9 ujedno pokazuje da je korištenje eliptičnih krivulja manje zahtjevno, ali da i mala vrijednost kašnjenja značajnije utječe na trajanje dogovora u odnosu na promjenu kriptografskih operacija.

### 7.5.3 Utjecaj propusnosti na trajanje dogovora

Na slici 7.10 prikazano je trajanje dogovora u odnosu na brzinu u kilobitima u sekundi (Kbps). U mjerena nisu uključeni rezultati za više od 10 Mbps jer iznad te brzine nema razlike u trajanju dogovora. Protokol ACAP ima relativno male zahtjeve na propusnost jer se dogovor može dovršiti i na poveznici brzine 10 Kbps ispod 2 sekunde.



Slika 7.10: Trajanje dogovora ovisno o brzini poveznice, protokolu i korištenim algoritmima

Uz korištenje kriptografije zasnovane na eliptičnim krivuljama dodatno se smanjuju zahtjevi na propusnost poveznice, što ih čini osobito pogodnim za dogovor na uređajima s ograničenim sposobnostima koji su prisutni u okolini Interneta stvari.

Slika također pokazuje da je dodatan promet koji koriste standardni kriptografski algoritmi u odnosu na EC algoritme veći od dodatnog prometa koji generira signalizacija protokola TCP. U slučaju korištenja EC algoritama dovoljno se ubrzava dogovor korištenjem protokola TCP, koji u tom slučaju postaje valjana alternativa protokolu UDP i u uvjetima niske propusnosti.

## 7.6 Složenost kriptografskih operacija u protokolu

U protokolu ACAP izvode se sljedeće kriptografske operacije:

- generiranje tajnog i javnog dijela za Diffie-Hellman razmjenu,
- računanje DH zajedničke tajne,
- računanje i provjera digitalnog potpisa i
- računanje i provjera HMAC zaštitne sume.

Trajanje stvaranja i obrade svake poruke u protokolu ACAP prikazano je u tablici 7.1. U tablici su vremena za stvaranje poruka označena žutom bojom, dok su vremena obrade neoznačena. Mjerenja su rađena korištenjem standardnih Diffie-Hellman parametara veličine 1536 bita te za ECDH korištenjem 163 bitne eliptične krivulje koji pružaju usporedive razine sigurnosti. Za digitalni potpis korišten je RSA-1536 kod standardnih algoritama, a ECDSA-sect163r2 kod EC algoritama.

**Tablica 7.1:** Trajanje stvaranja i obrade poruka

	Klijent	Poslužitelj	Klijent EC	Poslužitelj EC
INIT <sub>I</sub>	47.207 ms	4.157 ms	1.483 ms	0.676 ms
INIT <sub>R</sub>	4.284 ms	0.172 ms	1.125 ms	0.162 ms
LIST <sub>I</sub>	0.365 ms	0.282 ms	0.432 ms	0.636 ms
LIST <sub>R</sub>	0.171 ms	0.257 ms	0.599 ms	0.342 ms
Ukupno	52.009 ms	4.869 ms	3.460 ms	1.817 ms

Za stvaranje poruke INIT<sub>I</sub> potrebno je generirati tajni i javni dio za Diffie-Hellman razmjenu na klijentskoj strani. To je računalno najzahtjevniji dio cijelog dogovora i za standardne i za EC algoritme, što se odražava u vremenima u tablici. Obrada poruke INIT<sub>I</sub> na poslužiteljskoj strani sastoji se od računanja Diffie-Hellman zajedničke tajne, što predstavlja drugu najzahtjevniju operaciju u sklopu razmjene poruka.

Stvaranje poruke INIT<sub>R</sub> uključuje izračun digitalnog potpisa i HMAC sažetka. Izračun digitalnog potpisa kod poslužitelja odrađuje se u sklopu pozadinskog procesa u svrhu smanjivanja utjecaj DoS napada na protokol ACAP, a izračun HMAC-a može se odraditi tek po primitku poruke INIT<sub>I</sub>. Obrada poruke INIT<sub>R</sub> sastoji se od provjere digitalnog potpisa i HMAC-a te računanja DH zajedničke tajne. Dok su za standardne algoritme stvaranje i obrada INIT<sub>R</sub> slične zahtjevnosti, kod ECDSA algoritama složenost provjere digitalnog potpisa veća je od složenosti računanja (za algoritam RSA vrijedi suprotno).

Stvaranje poruke LIST<sub>I</sub> obuhvaća računanje digitalnog potpisa i HMAC zaštitne sume koji se po primitku provjeravaju na strani poslužitelja. Posljednja poruka LIST<sub>R</sub> sadrži samo digitalni potpis koji se provjerava na strani klijenta nakon primitka poruke. Kod ove poruke uočava se složenost stvaranja i provjeravanja digitalnog potpisa kod standardnih i EC algoritama.

Ukupni rezultati mjerjenja pokazuju kako je opterećenje veće na strani klijenta u odnosu na poslužitelja, kod standardnih algoritama taj je odnos veći nego kod EC algoritama zbog računalno zahtjevnog generiranja dobrih Diffie-Hellman vrijednosti za zaštitu komunikacije. Razlika između klijenta i poslužitelja postoji jer poslužitelj u pozadini generira DH vrijednosti. Po potrebi bi se to moglo raditi i na klijentskoj strani kako bi se smanjilo opterećenje kod početka dogovora.

## 7.7 Dohvaćanje podržanih algoritama

Dohvaćanje trenutno podržanih algoritama u sustavu izvedeno je pomoću vanjskog alata koji generira listu u JSON formatu koja je kompatibilna s aktualnim programskim ostvarenjem protokola ACAP. Rezultat izvođenja prikazan je na slici 7.11. U danom primjeru algoritmi su raspoređeni po abecedi, a konačan raspored uvjetovan je željenom razinom sigurnosti i ostalim uvjetima okoline u kojoj se protokol izvodi. Primjerice, u uređaju se mogu ugraditi prioriteti ili se oni mogu dohvaćati s nekog vanjskog servisa koji će redovito rangirati kriptografske algoritme ovisno o duljini ključa ili izlaza i postojećim napadima na te algoritme. Na raspoređivanje algoritama mogu utjecati i različiti dokumenti i specifikacije vezani uz određenu ustanovu, propisane standarde<sup>¶</sup> ili postupak certifikacije.

```
{  
    "algorithm_types": {  
        "hash": [  
            "MD5", "RIPEMD160", "SHA1", "SHA224", "SHA256", "SHA384", "SHA512", "Whirlpool" ],  
        "public_key": [  
            "DSA_1024", "DSA_2048", "DSA_3072", "RSA_1024", "RSA_1536", "RSA_2048",  
            "RSA_3072", "RSA_4096", "ECDSA_prime256v1_256", "ECDSA_sect163k1_163",  
            "ECDSA_secp384r1_384", ... ],  
        "secret_key": [  
            "AES_ECB_128", "AES_CBC_128", "AES_CTR_128", "AES_ECB_192", "AES_CBC_192",  
            "AES_CTR_192", "AES_ECB_256", "AES_CBC_256", "AES_CTR_256", "Blowfish_ECB_128",  
            "Blowfish_CBC_128", ... "Blowfish_CBC_152", "Blowfish_CTR_152", "CAST5_ECB_64",  
            "CAST5_CBC_80", ... "CAST5_ECB_112", "CAST5_CBC_112", "CAST5_CTR_112",  
            "Camellia_ECB_128", "Camellia_CBC_128", ... "Camellia_ECB_256", "Camellia_CBC_256",  
            "Camellia_CTR_256", "IDEA_ECB_128", "IDEA_CBC_128", "IDEA_CTR_128", "SEED_ECB_128",  
            "SEED_CBC_128", "SEED_CTR_128", "TripleDES_ECB_64", "TripleDES_CBC_64",  
            "TripleDES_CTR_64", "TripleDES_ECB_128", "TripleDES_CBC_128", "TripleDES_CTR_128",  
            "TripleDES_ECB_192", "TripleDES_CBC_192", "TripleDES_CTR_192" ]  
    }  
}
```

Slika 7.11: Skraćeni ispis alata za dohvaćanje podržanih kriptografskih algoritama

Trenutno podržani algoritmi ovise isključivo o biblioteci koja će se koristiti za zaštitu komunikacije nakon dogovora algoritma, odnosno o vanjskoj aplikaciji koja koristi protokol ACAP za dogovor svih preduvjeta potrebnih za zaštitu komunikacije. U danom primjeru dohvaćaju se podržani algoritmi koje koristi biblioteka *cryptography* za Python 3 koja se radi brzine izvođenja oslanja na biblioteku OpenSSL<sup>||</sup>.

## 7.8 Ostvareni sigurnosni mehanizmi

Sigurnosni mehanizmi u protokolu ACAP uvedeni su na više različitih razina. Osnovni mehanizmi izravno su vezani uz komunikacijski model te su opisani i formalno verificirani kroz

<sup>¶</sup>Popis aktualnih standarda i preporuka može se naći na: <http://www.keylength.com/>.

<sup>||</sup><https://www.openssl.org/>

alat Scyther u poglavlju 5. Ostatak sigurnosnih mehanizama izravno je vezan uz ostvarenje prototipa. Ostvareni su sljedeći sigurnosni mehanizmi:

- smanjenje utjecaja DoS napada iscrpljivanjem resursa na strani poslužitelja,
- sprječavanje napada ponavljanjem poruka,
- neovisnost kriptografskih ključeva za dogovor i ključeva za zaštitu komunikacije nakon dogovora.

Iz poglavlja 7.6 vidljivo je da su najzahtjevnije operacije vezane uz generiranje Diffie-Hellman vrijednosti i računanje DH tajnog ključa. Na poslužiteljskoj strani bi se obje te operacije trebale odraditi po primitku poruke  $\text{INIT}_I$ . Na taj način bi programsko ostvarenje bilo izravno izloženo napadu gdje se generira veliki broj  $\text{INIT}_I$  poruka. Kako bi se spriječio taj napad, generiranje DH vrijednosti odvojeno je u zasebnu proceduru koja periodički u pozadini generira DH vrijednosti. Dodatno se uz generiranje vrijednosti generira i digitalni potpis koji je sastavni dio poruke  $\text{INIT}_R (S_R(g^r))$  kako bi se smanjilo opterećenje stvaranja poruke  $\text{INIT}_R$ . U zadnjem redu tablice 7.1 prikazane su ukupne vrijednosti obrade za klijenta i poslužitelja. Te vrijednosti pokazuju da je trošak na strani mogućeg napadača, odnosno klijenta, veći od troška na strani poslužitelja.

Napadač koji se nalazi na putu između poslužitelja i klijenta mogao bi spremiti sve pakete uključene tijekom jedne razmjene i kasnije ponoviti cjelokupnu razmjenu tako da se opet koriste prije dogovoren ključevi i algoritmi. Prvi dio obrane od takvog napada leži u osvježavanju DH vrijednosti na strani poslužitelja, koja se generira svakih 30 sekundi. Novi javni dio DH razmjene izravno će utjecati na ostatak razmjene te će provjere HMAC sažetka biti neuspješne. Drugi dio obrane nalazi se u pamćenju prijašnjih *nonce* vrijednosti i odbijanju paketa s pretходно korištenim *nonce* vrijednostima.

Neovisnost kriptografskih ključeva postiže se uvođenjem pseudo nasumičnih funkcija (engl. *pseudo random function*, PRF) koje će iterativnim korištenjem algoritma HMAC jamčiti računalnu neovisnost ključeva. Koncepti za ostvarenje PRF-a preuzeti su iz protokola IKEv2 [34] i jamče da ključ koji se koristi za dogovor u protokolu ACAP neće otkriti nikakve podatke o dijeljenom tajnom ključu koji je rezultat protokola ACAP.

# Poglavlje 8

## Zaključak

U doktorskoj disertaciji dizajniran je i programski ostvaren protokol za sigurno dogovaranje kriptografskih algoritama i ključeva, koji omogućuje kriptografski prilagodljivu komunikaciju između dva komunicirajuća uređaja. Kriptografski prilagodljiva komunikacija ključna je u ostvarivanju dugoročno sigurne komunikacije, što je prikazano u pregledu područja istraživanja. Model protokola formalno je specificiran i verificiran u alatu za automatsku provjeru sigurnosti u odnosu na prethodno postavljene uvjete sigurne komunikacije. Primjena formalne verifikacije omogućila je dizajn protokola koji zadovoljava sve željene uvjete sigurne komunikacije.

Dogovor algoritama i ključeva u protokolu ACAP može se izvoditi na način koji je neovisan o sloju, aplikaciji i trenutnom operacijskom sustavu. Analizirana je integracija protokola u sustav za sigurnu komunikaciju u okolini Interneta stvari s osvrtom na prilagodljivost odabira algoritma koji uzima u obzir trenutne mogućnosti uređaja. Arhitektura protokola omogućuje učinkovit dogovor preduvjeta sigurne komunikacije u modelu klijent poslužitelj i mrežama ravнопravnih čvorova s niskim zahtjevima na mrežnu propusnost i računalnu snagu komunicirajućih strana. Programsko ostvarenje protokola omogućuje korištenje protokola na svim razinama protokolnog složaja TCP/IP te pokazuje robusnost rješenja i ostvarene sigurnosne mehanizme.

# Dodaci

## A Potpuni model protokola ACAP

```
hashfunction KDF, MAC, g, h;
const incr: Function;
usertype AlgList;

protocol @executability (DH, O) {
    role DH {
        var i, r: Nonce;
        recv_!DH1( DH, DH, h(g(r),i) );
        send_!DH2( DH, DH, h(g(i),r) );
    }
    role O {
        var i, r, Ni, Nr: Nonce;
        var I, R: Agent;
        var Li: AlgList;

        #send_!2( R, I, g(r), Nr, R, {h(g(r))}sk(R), MAC(h( Gi,r), R, Ni, Nr) );
        recv_!01( O, O, g(r), Nr, R, {h(g(r))}sk(R), MAC(h(g(i),r), R, Ni, Nr) );
        send_!02( O, O, g(r), Nr, R, {h(g(r))}sk(R), MAC(h(g(r),i), R, Ni, Nr) );
        #recv_!2( R, I, Gr, Nr, R, {h( Gr)}sk(R), MAC(h( Gr,i), R, Ni, Nr) );

        #send_!3( I, R, I, Li, {h(Li, g(i), Gr)}sk(I), MAC(h( Gr,i), I) );
        recv_!03( O, O, I, Li, {h(Li, g(i), g(r))}sk(I), MAC(h(g(r),i), I) );
        send_!04( O, O, I, Li, {h(Li, g(i), g(r))}sk(I), MAC(h(g(i),r), I) );
        #recv_!3( I, R, I, Li, {h(Li, Gi, g(r))}sk(I), MAC(h( Gi,r), I) );
    }
}
```

```

protocol acap(I, R) {
    role I {
        fresh i, Ni: Nonce;
        var Nr: Nonce;
        var Gr: Ticket;
        fresh Li: AlgList;
        var Lr: AlgList;

        send_1( I, R, g(i), Ni);
        recv_!2( R, I, Gr, Nr, R, {h(Gr)}sk(R), MAC(h( Gr,i), R, Ni, Nr) );
        claim_I1( I, Running, R, g(i), Gr);
        send_!3( I, R, I, Li, {h(Li, g(i), Gr)}sk(I), MAC(h(Gr,i), I) );
        recv_4( R, I, Lr, {h(g(i), Gr, Lr)}sk(R) );

        claim_I2( I, SKR, KDF(h(Gr,i)) );
        claim_I5( I, Commit, R, g(i), Gr);
        claim_I8( I, Nisynch );
    }

    role R {
        fresh r, Nr: Nonce;
        var Ni: Nonce;
        var Gi: Ticket;

        fresh Lr: AlgList;
        var Li: AlgList;
        recv_1( I, R, Gi, Ni);
        claim_R1( R, Running, I, g(r), Gi);
        send_!2( R, I, g(r), Nr, R, {h(g(r))}sk(R), MAC(h(Gi,r), R, Ni, Nr) );
        recv_!3( I, R, I, Li, {h(Li, Gi, g(r))}sk(I), MAC(h( Gi,r), I) );
        send_4( R, I, Lr, {h(Gi, g(r), Lr)}sk(R) );

        claim_R2( R, SKR, KDF(h(Gi,r)) );
        claim_R5( R, Commit, I, g(r), Gi);
        claim_R8( R, Nisynch );
    }
}

```

## B Ispis izvođenja programskog ostvarenja

### B.1 Klijentska strana

```
$ ./acap.py -C -a client.json -k private_key_client -l 10 -h 127.0.0.1
DEBUG - CREATE duration: 98489
DEBUG - Client created
DEBUG - host: ('127.0.0.1', 13000)
DEBUG - Client setup
DEBUG - Client connect
DEBUG - Client dh: 68:1c:97:72:1f:ca:8a:7c:5b:15:c3:a6:6a:88:67:37:99:b5:2c:67
:67:23:ae:e0:52:94:fd:a0:99:f7:d7:85:2d:2a:52:94:b4:f4:fc:2e:c8:d2:20:ec:9b:0a
:56:da:8a:7e:36:87:ce:5e:6e:ac:42:50:ee:ad:73:cc:6a:d5:8c:fb:b5:3d:53:bb:32:21
:3b:57:80:df:34:69:6d:67:62:42:93:7f:09:fb:b7:3c:48:4a:8e:d6:ae:bf:78:b6:ca:2b
:d5:7c:4c:ab:ee:64:e0:99:25:bf:17:10:0b:61:ec:cb:b8:5b:a7:d7:02:b0:ea:1e:03:8a
:cd:a4:4a:cf
DEBUG - Client Nonce: 23:16:66:f8:a4:85:18:16:64:0c:b6:06
INFO - Client public key:
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIB3DQEBAQUAA4GNADCBiQKBgQC4TDSclGnCkXgP0C8mt889rBR1
iFoXBt01G4VF1XhxursgU+Es2PCqRUYT28ZqdNrq2CAyuIT33G5CXCwg8x/CBh2
qmFJ43NpauGg13b2LFNSg3j8UxxwGSGIvKvx0maGnSJByepogXkWuY0bL5mR0n01
JPH/im05V2+Jox951QIDAQAB
-----END PUBLIC KEY-----
DEBUG - PREP_INIT_I duration: 79
DEBUG - PROC_INIT_R duration: 6586
DEBUG - Server dh: 94:44:cc:29:f1:dc:14:66:d8:ce:07:90:ca:5d:77:64:0b:7c:21:16
:ef:42:31:a4:bf:50:cd:94:8b:71:1b:29:30:1d:20:f0:0e:6e:d6:56:8e:b4:9c:cd:58:5f
:54:12:a1:04:eb:32:43:6d:e7:5a:d0:6d:23:0b:6b:2f:7e:5a:0d:40:fb:4c:c7:bd:ee:b7
:0a:e3:ba:df:06:11:49:78:77:6b:fc:37:f5:f2:c3:84:18:44:e3:9a:28:42:0d:76:ea:59
:27:71:41:24:07:07:3d:88:40:eb:c9:67:25:bc:ea:4c:72:c7:98:9d:7d:dd:11:25:50:b5
:2d:b3:84:79
DEBUG - DH key: f7:d8:2a:83:a2:58:55:e4:af:02:aa:da:d2:f2:c7:f2:75:62:a7:95:cb
:e7:b7:50:ee:f7:6d:cd:4b:35:98:ee:38:d4:53:bc:b8:91:ba:a4:82:88:cc:18:2d:55:fd
:79:ab:3d:48:16:17:84:78:bb:45:cb:60:a9:05:ee:78:78:e4:bc:7c:75:cc:b0:c0:a2:7e
:69:53:fa:e4:1f:d2:d0:26:75:61:97:94:33:7c:6e:3e:88:44:61:92:10:ee:92:bc:71:ab
:e4:84:ae:68:0b:fc:aa:db:3b:73:9d:e8:94:86:0e:ef:fe:10:33:12:0e:6d:a5:59:ac:78
:a6:bd:a9
DEBUG - Server Nonce: df:69:de:1a:d8:13:c7:b0:74:0a:71:7a
INFO - Server public key:
```

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCTYB3xU17CS5AGxKbWraxuGOYb
7dff8AAjjiwXrdBr6jqJ5zSHtR/bKxsRNPMNvzihVI1Taa7/CQy3EIRKkLeG4I7M
USeStpBfvQlpAK+lmyUE3haXVUNgPpALdXUpn+IaB+TDHuWl19z9dPDkSDLzQzga
pVwSgXpEZDUInwKK4QIDAQAB
-----END PUBLIC KEY-----
DEBUG - Client LIST: {"algorithm types": {"hash": ["SHA-256", "RIPEMD",
"SHA-1"], "public_key": ["RSA_1024", "RSA_2048", "ECDSA_192"], "secret_key":
["AES-CTR_256", "3DES_192", "AES-CBC_128"]}}
DEBUG - PREP_LIST_I duration: 621
DEBUG - PROC_LIST_R duration: 259
DEBUG - Server LIST: {"algorithm types": {"hash": ["SHA3-512", "SHA-512",
"SHA-256"], "public_key": ["ECDSA_192", "ECDSA_224", "RSA_2048"], "secret_key":
["AES-CTR_256", "Salsa20_256", "AES-CBC_128"]}}
DEBUG - NEGOTIATE duration: 104
INFO - Client negotiated:
{
  "hash": {
    "algorithm": "SHA-256",
    "timestamp": "2015-11-22 14:59"
  },
  "public_key": {
    "algorithm": "RSA_2048",
    "timestamp": "2015-11-22 14:59"
  },
  "secret_key": {
    "algorithm": "AES-CTR_256",
    "timestamp": "2015-11-22 14:59"
  }
}
INFO - Shared secret key (256 bit): bb:66:cc:98:10:fd:b7:a0:73:9b:9e:35:49:11:
43:7e:e5:28:3d:b3:69:5e:e2:69:4c:c9:a8:11:6d:d5:57:9a
INFO - TOTAL duration: 119820
```

## B.2 Poslužiteljska strana

```
$ ./acap.py -S -a server.json -k private_key_server -l 10
DEBUG - CREATE duration: 1897
Refreshed DH and Nonce...
DEBUG - Server created
```

```
DEBUG - Server listening
DEBUG - Server accepted
DEBUG - Server dh: 94:44:cc:29:f1:dc:14:66:d8:ce:07:90:ca:5d:77:64:0b:7c:21:16
:ef:42:31:a4:bf:50:cd:94:8b:71:1b:29:30:1d:20:f0:0e:6e:d6:56:8e:b4:9c:cd:58:5f
:54:12:a1:04:eb:32:43:6d:e7:5a:d0:6d:23:0b:6b:2f:7e:5a:0d:40:fb:4c:c7:bd:ee:b7
:0a:e3:ba:df:06:11:49:78:77:6b:fc:37:f5:f2:c3:84:18:44:e3:9a:28:42:0d:76:ea:59
:27:71:41:24:07:07:3d:88:40:eb:c9:67:25:bc:ea:4c:72:c7:98:9d:7d:dd:11:25:50:b5
:2d:b3:84:79
DEBUG - Server Nonce: df:69:de:1a:d8:13:c7:b0:74:0a:71:7a
INFO - Server public key:
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIB3DQEBAQUAA4GNADCBiQKBgQCTYB3xU17CS5AGxKbWraxuGOYb
7ssf8AAjjiwXrdBr6jqJ5zShtR/bKxsRNPMNvzihVI1Taa7/CQy3EIRKkLeG4I7M
USeStpBfvQlpAK+lmyUE3haXVUNgPpALdXUpn+IaB+TDHuWl19z9dPDkSDLzQzga
pVwSgXpEZDUIInwKK4QIDAQAB
-----END PUBLIC KEY-----
DEBUG - RECV duration: 1863387
DEBUG - PROC_INIT_I duration: 5955
DEBUG - Client dh: 68:1c:97:72:1f:ca:8a:7c:5b:15:c3:a6:6a:88:67:37:99:b5:2c:67
:67:23:ae:e0:52:94:fd:a0:99:f7:d7:85:2d:2a:52:94:b4:f4:fc:2e:c8:d2:20:ec:9b:0a
:56:da:8a:7e:36:87:ce:5e:6e:ac:42:50:ee:ad:73:cc:6a:d5:8c:fb:b5:3d:53:bb:32:21
:3b:57:80:df:34:69:6d:67:62:42:93:7f:09:fb:b7:3c:48:4a:8e:d6:ae:bf:78:b6:ca:2b
:d5:7c:4c:ab:ee:64:e0:99:25:bf:17:10:0b:61:ec:cb:b8:5b:a7:d7:02:b0:ea:1e:03:8a
:cd:a4:4a:cf
DEBUG - Client Nonce: 23:16:66:f8:a4:85:18:16:64:0c:b6:06
DEBUG - DH key: f7:d8:2a:83:a2:58:55:e4:af:02:aa:da:d2:f2:c7:f2:75:62:a7:95:cb
:e7:b7:50:ee:f7:6d:cd:4b:35:98:ee:38:d4:53:bc:b8:91:ba:a4:82:88:cc:18:2d:55:fd
:79:ab:3d:48:16:17:84:78:bb:45:cb:60:a9:05:ee:78:78:e4:bc:7c:75:cc:b0:c0:a2:7e
:69:53:fa:e4:1f:d2:d0:26:75:61:97:94:33:7c:6e:3e:88:44:61:92:10:ee:92:bc:71:ab
:e4:84:ae:68:0b:fc:aa:db:3b:73:9d:e8:94:86:0e:ef:fe:10:33:12:0e:6d:a5:59:ac:78
:a6:bd:a9
DEBUG - PREP_INIT_R duration: 295
DEBUG - SEND duration: 31
DEBUG - RECV duration: 8441
DEBUG - PROC_LIST_I duration: 492
INFO - Client public key:
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIB3DQEBAQUAA4GNADCBiQKBgQC4TDSclGnCkXgP0C8mt889rBR1
iFoXBt01G4VF1XhxursgU+Es2PCqRUYT28ZqdNrq2CAyuIT33G5CXCwg8x/CBh2
qmFJ43NpauGg13b2LFNSg3j8UxxwGSGIvKvx0maGnSJByepogXkWuY0bL5mR0n01
-----END PUBLIC KEY-----
```

```
JPH/im05V2+Jox951QIDAQAB
-----END PUBLIC KEY-----
DEBUG - Client LIST: {"algorithm types": {"hash": ["SHA-256", "RIPEMD",
"SHA-1"], "public_key": ["RSA_1024", "RSA_2048", "ECDSA_192"], "secret_key":
["AES-CTR_256", "3DES_192", "AES-CBC_128"]}}
DEBUG - Server LIST: {"algorithm types": {"hash": ["SHA3-512", "SHA-512",
"SHA-256"], "public_key": ["ECDSA_192", "ECDSA_224", "RSA_2048"], "secret_key":
["AES-CTR_256", "Salsa20_256", "AES-CBC_128"]}}
DEBUG - PREP_LIST_R duration: 357
DEBUG - SEND duration: 28
DEBUG - NEGOTIATE duration: 75
INFO - Server negotiated:
{
  "hash": {
    "algorithm": "SHA-256",
    "timestamp": "2015-11-22 14:59"
  },
  "public_key": {
    "algorithm": "RSA_2048",
    "timestamp": "2015-11-22 14:59"
  },
  "secret_key": {
    "algorithm": "AES-CTR_256",
    "timestamp": "2015-11-22 14:59"
  }
}
INFO - Shared secret key (256 bit): bb:66:cc:98:10:fd:b7:a0:73:9b:9e:35:49:11
:43:7e:e5:28:3d:b3:69:5e:e2:69:4c:c9:a8:11:6d:d5:57:9a
INFO - TOTAL duration: 17757
```

## C Formati poruka

### C.1 Format poruke INIT<sub>I</sub>

0	7	15	23	31
Duljina poruke		Tip=0		Duljina DH
Duljina DH				
Klijentov Diffie-Hellman javni dio				
		⋮		
Duljina <i>noncea</i>				
Klijentov <i>nonce</i>				
		⋮		

### C.2 Format poruke INIT<sub>R</sub>

0	7	15	23	31
Duljina poruke		Tip=1		Duljina DH
Duljina DH				
Poslužiteljev Diffie-Hellman javni dio				
		⋮		
Duljina <i>noncea</i>				
Poslužiteljev <i>nonce</i>				
		⋮		
Duljina ključa				
Javni ključ poslužitelja				
		⋮		
Duljina potpisa				
Digitalni potpis				
		⋮		
Duljina HMAC-a				
HMAC				
		⋮		

### C.3 Format poruke LIST<sub>I</sub>

0	7	15	23	31
Duljina poruke		Tip=2		Duljina ključa
Duljina ključa				
	Javni ključ poslužitelja			
	⋮			
Duljina popisa				
	Popis algoritama klijenta			
	⋮			
Duljina potpisa				
	Digitalni potpis			
	⋮			
Duljina HMAC-a				
	HMAC			
	⋮			

### C.4 Format poruke LIST<sub>R</sub>

0	7	15	23	31
Duljina poruke		Tip=3		Duljina popisa
Duljina popisa				
	Popis algoritama poslužitelja			
	⋮			
Duljina potpisa				
	Digitalni potpis			
	⋮			

# Literatura

- [1] Van Tilborg, H. C., Jajodia, S., Encyclopedia of cryptography and security. Springer Science & Business Media, 2011.
- [2] Daemen, J., Rijmen, V., The design of Rijndael: AES-the advanced encryption standard. Springer Science & Business Media, 2013.
- [3] Housley, R., “Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)”, RFC 3686 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc3686.txt> Jan. 2004.
- [4] McGrew, D., Bailey, D., “AES-CCM Cipher Suites for Transport Layer Security (TLS)”, RFC 6655 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc6655.txt> Jul. 2012.
- [5] Salowey, J., Choudhury, A., McGrew, D., “AES Galois Counter Mode (GCM) Cipher Suites for TLS”, RFC 5288 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc5288.txt> Aug. 2008.
- [6] Gutmann, P., “Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)”, RFC 7366 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc7366.txt> Sep. 2014.
- [7] Gallagher, P., Kerry, C., “Fips pub 186-4: Digital signature standard, dss”, 2013.
- [8] Diffie, W., Hellman, M., “New directions in cryptography”, Information Theory, IEEE Transactions on, Vol. 22, No. 6, Nov 1976, str. 644-654.
- [9] Bernstein, D. J., Lange, T., “Safecurves: choosing safe curves for elliptic-curve cryptography”, <http://safecurves.cr.yp.to>, datum pristupa: 23. travnja 2015.
- [10] Krawczyk, H., “Sigma: The ‘sign-and-mac’ approach to authenticated diffie-hellman and its use in the ike protocols”, in Advances in Cryptology-CRYPTO 2003. Springer, 2003, str. 400–425.

- [11] Chen, L., “Recommendation for key derivation using pseudorandom functions”, NIST Special Publication, Vol. 800, 2008, str. 108.
- [12] Krawczyk, H., Eronen, P., “HMAC-based Extract-and-Expand Key Derivation Function (HKDF)”, RFC 5869 (Informational), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc5869.txt> May 2010.
- [13] Krawczyk, H., “Cryptographic extraction and key derivation: The hkdf scheme”, in Advances in Cryptology–CRYPTO 2010. Springer, 2010, str. 631–648.
- [14] Freier, A., Karlton, P., Kocher, P., “The Secure Sockets Layer (SSL) Protocol Version 3.0”, RFC 6101 (Historic), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc6101.txt> Aug. 2011.
- [15] Dierks, T., Rescorla, E., “The Transport Layer Security (TLS) Protocol Version 1.2”, RFC 5246 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc5246.txt> Updated by RFCs 5746, 5878, 6176. Aug. 2008.
- [16] Krawczyk, H., “Skeme: A versatile secure key exchange mechanism for internet”, in Network and Distributed System Security, 1996., Proceedings of the Symposium on. IEEE, 1996, str. 114–127.
- [17] Arkko, J., Kempf, J., Zill, B., Nikander, P., “SEcure Neighbor Discovery (SEND)”, RFC 3971 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc3971.txt> Updated by RFCs 6494, 6495, 6980. Mar. 2005.
- [18] Vasić, V., Kukec, A., Mikuc, M., “Deploying new hash algorithms in secure neighbor discovery”, in Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on. IEEE, 2011, str. 1–5.
- [19] Vasic, V., Mikuc, M., “Security agility solution independent of the underlying protocol architecture”, in Proceedings of the First International Conference on Agreement Technologies, Oct. 2012.
- [20] Vasić, V., Mikuc, M., Vuković, M., “Lightweight and adaptable solution for security agility”, KSII Transactions on Internet & Information Systems, Vol. 10, 2016, str. 1212–1228, dostupno na: <http://dx.doi.org/10.3837/tiis.2016.03.015>
- [21] Aiello, W., Bellovin, S., Blaze, M., Canetti, R., Ioannidis, J., Keromytis, A., Reingold, O., “Just fast keying: Key agreement in a hostile internet”, ACM Transactions on Information and System Security (TISSEC), Vol. 7, No. 2, 2004, str. 242–273.

- [22] Kent, S., Seo, K., "Security Architecture for the Internet Protocol", RFC 4301 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc4301.txt> Updated by RFC 6040. Dec. 2005.
- [23] Rescorla, E., Modadugu, N., "Datagram Transport Layer Security Version 1.2", RFC 6347 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc6347.txt> Jan. 2012.
- [24] Petullo, W. M., Zhang, X., Solworth, J. A., Bernstein, D. J., Lange, T., "Minimalt: Minimal-latency networking through better security", in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013, str. 425–438.
- [25] Ylonen, T., Lonvick, C., "The Secure Shell (SSH) Protocol Architecture", RFC 4251 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc4251.txt> Jan. 2006.
- [26] Bittau, A., Hamburg, M., Handley, M., Mazieres, D., Boneh, D., "The case for ubiquitous transport-level encryption.", in USENIX Security Symposium, 2010, str. 403–418.
- [27] Roskind, J., "Quick udp internet connections", 2013.
- [28] Duong, T., Rizzo, J., "Here come the xor ninjas", Unpublished manuscript, 2011.
- [29] Be'ery, T., Shulman, A., "A perfect crime? only time will tell", Black Hat Europe, Vol. 2013, 2013.
- [30] Fardan, N. J. A., Paterson, K., "Lucky thirteen: Breaking the tls and dtls record protocols", in Security and Privacy (SP), 2013 IEEE Symposium on. IEEE, 2013, str. 526–540.
- [31] AlFardan, N. J., Bernstein, D. J., Paterson, K. G., Poettering, B., Schuldt, J., "On the security of rc4 in tls.", in USENIX Security, 2013, str. 305–320.
- [32] Durumeric, Z., Kasten, J., Adrian, D., Halderman, J. A., Bailey, M., Li, F., Weaver, N., Amann, J., Beekman, J., Payer, M. *et al.*, "The matter of heartbleed", in Proceedings of the 2014 Conference on Internet Measurement Conference. ACM, 2014, str. 475–488.
- [33] Bland, M., "Finding more than one worm in the apple", Queue, Vol. 12, No. 5, May 2014, str. 10:10–10:21, dostupno na: <http://doi.acm.org/10.1145/2620660.2620662>
- [34] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc5996.txt> Obsoleted by RFC 7296, updated by RFCs 5998, 6989. Sep. 2010.

- [35] Canetti, R., Krawczyk, H., “Security analysis of ike’s signature-based key-exchange protocol”, in Advances in Cryptology—CRYPTO 2002. Springer, 2002, str. 143–161.
- [36] Ylonen, T., Lonvick, C., “The Secure Shell (SSH) Transport Layer Protocol”, RFC 4253 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc4253.txt> Updated by RFC 6668. Jan. 2006.
- [37] Ylonen, T., Lonvick, C., “The Secure Shell (SSH) Connection Protocol”, RFC 4254 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc4254.txt> Jan. 2006.
- [38] Ylonen, T., Lonvick, C., “The Secure Shell (SSH) Authentication Protocol”, RFC 4252 (Proposed Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc4252.txt> Jan. 2006.
- [39] Freire, E. S., Hofheinz, D., Kiltz, E., Paterson, K. G., “Non-interactive key exchange”, in Public-Key Cryptography—PKC 2013. Springer, 2013, str. 254–271.
- [40] Yao, A. C.-C., Zhao, Y., “Oake: A new family of implicitly authenticated diffie-hellman protocols”, in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013, str. 1113–1128.
- [41] Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S., “An efficient protocol for authenticated key agreement”, Designs, Codes and Cryptography, Vol. 28, No. 2, 2003, str. 119–134.
- [42] Krawczyk, H., “Hmqv: A high-performance secure diffie-hellman protocol”, in Advances in cryptology—CRYPTO 2005. Springer Berlin Heidelberg, 2005, str. 546–566.
- [43] Sarr, A. P., Elbaz-Vincent, P., Bajard, J.-C., “A secure and efficient authenticated diffie–hellman protocol”, in Public Key Infrastructures, Services and Applications. Springer, 2010, str. 83–98.
- [44] Liu, S., Sakurai, K., Weng, J., Zhang, F., Zhao, Y., “Security model and analysis of fhmqv, revisited”, in Information Security and Cryptology. Springer, 2014, str. 255–269.
- [45] Gruber, M., Maier, M. K., Schafferer, M., Schanes, C., Grechenig, T., “Concept and design of a transparent security layer to enable anonymous voip calls”, Distributed Systems and Applications, 2014, str. 58.
- [46] Barenghi, A., Beretta, M., Di Federico, A., Pelosi, G., “Snake: An end-to-end encrypted online social network”, in High Performance Computing and Communications, 2014

- IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICCESS), 2014 IEEE Intl Conf on. IEEE, 2014, str. 763–770.
- [47] Zhao, S., Zhang, Q., “shmrv: An efficient key exchange protocol for power-limited devices”, Information Security Practice and Experience, 2015, str. 154–167.
- [48] Dagdelen, Ö., Fischlin, M., Gagliardoni, T., Marson, G. A., Mittelbach, A., Onete, C., “A cryptographic analysis of opacity”, in Computer Security—ESORICS 2013. Springer, 2013, str. 345–362.
- [49] Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D., “Context aware computing for the internet of things: A survey”, Communications Surveys & Tutorials, IEEE, Vol. 16, No. 1, 2014, str. 414–454, dostupno na: <http://arxiv.org/pdf/1305.0982.pdf>
- [50] Kapadia, A., Kotz, D., Triandopoulos, N., “Opportunistic sensing: Security challenges for the new paradigm”, in Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International, Jan 2009, str. 1-10.
- [51] Zhang, B., Ma, X.-X., Qin, Z.-G., “Security architecture on the trusting internet of things”, Journal of Electronic Science and Technology, Vol. 9, No. 4, 2011, str. 364–367.
- [52] Suo, H., Wan, J., Zou, C., Liu, J., “Security in the internet of things: A review”, in Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on, Vol. 3, March 2012, str. 648-651.
- [53] Krawczyk, H., “Perfect forward secrecy”, in Encyclopedia of Cryptography and Security. Springer, 2011, str. 921–922.
- [54] Lepinski, M., Kent, S., “Additional Diffie-Hellman Groups for Use with IETF Standards”, RFC 5114 (Informational), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc5114.txt> Jan. 2008.
- [55] Crockford, D., “The application/json Media Type for JavaScript Object Notation (JSON)”, RFC 4627 (Informational), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc4627.txt> Obsoleted by RFC 7159. Jul. 2006.
- [56] Cremers, C. J. F., Scyther: Unbounded Verification of Security Protocols. ETH, Department of Computer Science, 2007.
- [57] Lowe, G., “Breaking and fixing the needham-schroeder public-key protocol using fdr”, in Tools and Algorithms for the Construction and Analysis of Systems. Springer, 1996, str. 147–166.

- [58] Cremers, C. J. F., “Scyther: Semantics and verification of security protocols”, 2006.
- [59] Postel, J., “Transmission Control Protocol”, RFC 793 (INTERNET STANDARD), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc793.txt> Updated by RFCs 1122, 3168, 6093, 6528. Sep. 1981.
- [60] Postel, J., “User Datagram Protocol”, RFC 768 (INTERNET STANDARD), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc768.txt> Aug. 1980.
- [61] Postel, J., “Internet Protocol”, RFC 791 (INTERNET STANDARD), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc791.txt> Updated by RFCs 1349, 2474, 6864. Sep. 1981.
- [62] Deering, S., Hinden, R., “Internet Protocol, Version 6 (IPv6) Specification”, RFC 2460 (Draft Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc2460.txt> Updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112. Dec. 1998.
- [63] Mogul, J., Deering, S., “Path MTU discovery”, RFC 1191 (Draft Standard), Internet Engineering Task Force, dostupno na: <http://www.ietf.org/rfc/rfc1191.txt> Nov. 1990.
- [64] Salopek, D., Vasic, V., Zec, M., Mikuc, M., Vasarevic, M., Koncar, V., “A network testbed for commercial telecommunications product testing”, in Software, Telecommunications and Computer Networks (SoftCOM), 2014 22nd International Conference on. IEEE, 2014, str. 372–377.

# Popis oznaka

ACAP *Agile Cryptographic Agreement Protocol*

DH protokol Diffie-Hellman

EC kriptografski algoritam temeljen na eliptičnim krivuljama

ECDSA *Elliptic Curve Digital Signature Algorithm*

HMAC *Hashed Message Authentication Code*

KDF *Key Derivation Function* - funkcija za izračunavanje ključeva

MAC *Message Authentication Code*

MITM Napad čovjek u sredini

PKI *Public Key Infrastructure* - infrastruktura javnog ključa

PRF *Pseudo Random Function* - pseudo nasumična funkcija

RSA Rivest Shamir Alderman asimetrični algoritam

SIGMA protokol *SIGN and MAC*

# Popis slika

3.1.	Arhitektura protokola SSL/TLS . . . . .	18
3.2.	Arhitektura protokola SSH . . . . .	21
3.3.	Slojevita arhitektura Interneta stvari . . . . .	24
4.1.	Grafički prikaz razmjene poruka protokola ACAP . . . . .	27
4.2.	Popisi kriptografskih algoritama i skupova koji se dogovaraju . . . . .	29
4.3.	Primjer ponuđenih i dogovorenih algoritama . . . . .	30
4.4.	Grafički prikaz razmjene poruka protokola za upravljanje prijenosom podataka protokola SSH . . . . .	35
5.1.	Opis protokola Needham-Schroeder-Lowe . . . . .	38
5.2.	Definicija protokola NSL . . . . .	38
5.3.	Definicija zahtjeva za ulogu I protokola NSL . . . . .	39
5.4.	SPDL model i dijagram toka za protokol Needham-Schroeder-Lowe . . . . .	40
5.5.	Rezultati provjere i verifikacije modela protokola NSL . . . . .	40
5.6.	Prikaz varijabli za uloge započinjatelja i primatelja . . . . .	41
5.7.	Komunikacija sa strane započinjatelja . . . . .	42
5.8.	Komunikacija sa strane primatelja . . . . .	42
5.9.	Komutativnost Diffie-Hellman operacije . . . . .	43
5.10.	Razmjena poruka INIT <sub>R</sub> i LIST <sub>I</sub> definirana u ulozi 0 . . . . .	43
5.11.	Dijagram toka za protokol ACAP . . . . .	44
5.12.	Definicija sigurnosnih zahtjeva za modelirani protokol . . . . .	45
5.13.	Rezultati verifikacije protokola s alatom Scyther . . . . .	46
6.1.	Primjene sigurne komunikacije u okolini Interneta stvari . . . . .	48
7.1.	Arhitektura programskog ostvarenja protokola ACAP . . . . .	56
7.2.	Primjer pokretanja ACAP klijenta . . . . .	57
7.3.	Primjer pokretanja ACAP poslužitelja . . . . .	58
7.4.	Bitovni zapis poruke INIT <sub>I</sub> . . . . .	59
7.5.	Bitovni prikaz primjera poruke INIT <sub>I</sub> . . . . .	59

## Popis slika

---

7.6. Emulirana mrežna topologija za mjerena u alatu IMUNES . . . . .	62
7.7. Veličina poruka ovisno o veličini RSA javnih ključeva . . . . .	63
7.8. Veličina poruka ovisno o veličini ECDSA javnih ključeva . . . . .	64
7.9. Trajanje dogovora ovisno o kašnjenju, protokolu i korištenim algoritmima . . .	64
7.10. Trajanje dogovora ovisno o brzini poveznice, protokolu i korištenim algoritmima	65
7.11. Skraćeni ispis alata za dohvaćanje podržanih kriptografskih algoritama . . . . .	67

# **Životopis**

Valter Vasić rođen je u Puli 1987. godine. Završio je opću gimnaziju u Srednjoj školi Mate Balote u Poreču. Potom je magistrirao u području informacijske i komunikacijske tehnologije na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu 2010. godine. Nakon tog zaposlio se na istom fakultetu kao znanstveni novak gdje je postao razvijatelj mrežnog emulatora IMUNES i upisao Doktorski studij računarstva pod mentorstvom izv. prof. dr. sc. Miljenka Mikuća. Radio je na projektu E-IMUNES pod pokroviteljstvom Ericsona Nikole Tesle d.d te aktivno sudjelovao u raznim projektima u suradnji s industrijom. Objavio je više od 10 radova u časopisima i na konferencijama. Od 2015. godine član je udruge HoneyNet i član upravnog odbora međunarodnog istraživačkog projekta COST IC1306 “Cryptography for Secure Digital Interaction”. Njegova područja istraživanja uključuju mrežnu komunikaciju, računalnu i komunikacijsku sigurnost te virtualizaciju.

## **Popis objavljenih djela**

### **Radovi u časopisima**

1. Vasić, V., Mikuc M., Vuković M., “Lightweight and adaptable solution for security agility”, KSII Transactions on Internet & Information Systems (1976-7277), Vol. 10, No. 3, Ožujak 2016., str. 1212-1228.
2. Vasić V., Sužnjević M., Mikuc M., Matijašević M., “Scalable software architecture for distributed MMORPG traffic generation based on integration of UrBBaN-Gen and IMUNES”, Journal of Communications Software and Systems (1845-6421), Vol. 8, No. 4, Prosinac 2012., str. 93-101.
3. Bujas, G., Vuković M., Vasić V., Mikuc M., “Smart Detection and Classification of Application-Layer Intrusions in Web Directories”, Smart Computing Review (2234-4624), Vol. 5, No 6, Prosinac 2015., str. 510-519.

## **Radovi na konferencijama**

1. Salopek, D., Vasić, V., Zec, M., Mikuc, M., Vašarević, M., Končar, V., “A network testbed for commercial telecommunications product testing”, Proceedings of 22nd International Conference on Software, Telecommunications and Computer Networks - SoftCOM 2014, Sveučilište u Splitu, Rujan 2014.
2. Vasić, V., Mikuc, M., “Security Agility Solution Independent of the Underlaying Protocol Architecture”, Proceedings of the First International Conference on Agreement Technologies - AT 2012, Dubrovnik, Rujan 2012.
3. Vasić, V., Kukec, A., Mikuc, M., “Deploying New Hash Algorithms in Secure Neighbor Discovery”, Proceedings of the 19th International Conference on Software, Telecommunications and Computer Networks - SoftCOM 2012, Sveučilište u Splitu, Rujan 2012.

# Biography

Valter Vasić was born in Pula in 1987. He received his M.Sc. degree in information and communication technology from University of Zagreb in 2010. After that he started working as a research associate at the University of Zagreb, Faculty of Electrical Engineering and Computing as a developer for the IMUNES network emulator. In 2010 he started his Ph.D. in Computer Science under the supervision of associate professor Miljenko Mikuc, Ph.D. He was researcher on the E-IMUNES project funded by Ericsson Nikola Tesla, Zagreb and actively contributed to various project with the industry. He published more than 10 papers in journals and conference proceedings. Since 2015 he is a member of the Honeynet project and a management committee member for the COST Action IC1306 “Cryptography for Secure Digital Interaction”. The focus of his research is secure network communication. His research interests are in the area of network communication, computer security and virtualization.