

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

**ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ**

**КАФЕДРА СИСТЕМОГО ПРОГРАМУВАННЯ І  
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ**

«На правах рукопису»  
УДК \_\_\_\_\_

«До захисту допущено»  
Завідувач кафедри СПСКС

\_\_\_\_\_ В.П.Тарасенко  
(підпис) (ініціали, прізвище)  
“ ” \_\_\_\_\_ 2018р.

**Магістерська дисертація**

**на здобуття ступеня магістра**

зі спеціальності 123 Комп'ютерна інженерія (Спеціалізовані комп'ютерні системи)

на тему СИСТЕМА ЗАХИСТУ РЕСУРСІВ ПРОГРАМНО-  
КОНФІГУРОВАНИХ МЕРЕЖ SDN

Виконав: студент II курсу, групи КВ-73мп  
(шифр групи)

Григор'єв Вадим Валерійович  
(прізвище, ім'я, по батькові)

\_\_\_\_\_ (підпис)

Науковий керівник к.т.н., доцент Орлова М.М.  
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

\_\_\_\_\_ (підпис)

Рецензент професор кафедри ОТ, д.т.н., проф. Кулаков Ю.О.  
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_ (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент \_\_\_\_\_  
(підпис)

Київ – 2018 рік

**Національний технічний університет України**  
**«Київський політехнічний інститут імені Ігоря Сікорського»**  
**Факультет прикладної математики**  
**Кафедра прикладної математики**

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність – 123 Комп'ютерна інженерія

Спеціалізовані комп'ютерні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_ В.П. Тарасенко

«\_\_» \_\_\_\_\_ 2018 р.

**ЗАВДАННЯ**  
**на магістерську дисертацію студенту**

Григор'єву Вадиму Валерійовичу

1. Тема дисертації «Система захисту ресурсів програмно-конфігурованих мереж SDN», науковий керівник дисертації Орлова Марія Миколаївна, к.т.н., доцент, затверджені наказом по університету від «\_\_» \_\_\_\_\_ 2018 р. № \_\_\_\_\_
2. Термін подання студентом дисертації «14» грудня 2018 р.
3. Об'єкт дослідження: захист ресурсів програмно-конфігурованих мереж SDN.
4. Предмет дослідження: модель системи захисту ресурсів програмно-конфігурованої мережі, що дозволяє захистити мережу від різних видів атак.
5. Перелік завдань, які потрібно розробити:
  - провести аналіз вразливостей програмно-конфігурованих мереж;
  - провести аналіз можливих загроз;
  - дослідити способи захисту програмно-конфігурованих мереж;
  - розробити та дослідити модель системи захисту ресурсів програмно-конфігурованої мережі;
  - обґрунтувати вибір загроз, від яких захищає розроблена система;
  - розробити програмне забезпечення для аналізу параметрів моделі системи.
6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
  - таблиці з результатами щодо обґрунтування критеріїв захисту моделі;
  - схема алгоритму автентифікації пристрою рівня передачі даних зі сторони контролера;
  - схема алгоритму автентифікації пристрою рівня передачі даних зі сторони пристрою;
  - моделювання коректної роботи системи;

- моделювання роботи системи при атаці підслуховування
- моделювання роботи системи при спробі автентифікації нелегітимного пристрою в мережі.

7. Орієнтовний перелік публікацій:

- Тези доповіді “ Алгоритм автентифікації пристрою в мережі SDN”
- Тези доповіді “ Система захисту ресурсів у мережах SDN”

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання «04» жовтня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Грунтовне ознайомлення з предметною галуззю	17.10.2017	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	04.12.2017	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	15.02.2018	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	05.04.2018	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	15.05.2018	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	15.06.2018	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка матеріалів доповіді на конференції ПМК-2018	05.11.2018	
8.	Оформлення текстової і графічної частини магістерської дисертації	04.12.2018	

Студент

В.В. Григор'єв

Науковий керівник дисертації

М.М. Орлова

## РЕФЕРАТ

**Актуальність теми.** Потребу захисту ресурсів необхідно вирішувати для будь-якого типу мереж. Зараз все більшу популярність набирають мережі SDN (software-defined networking, програмно-конфігуровні мережі), які мають переваги у гнучкості та можливостях масштабування, у порівнянні зі звичайними мережами. Проте ці мережі структурно досить сильно відрізняються від звичайних мереж, тому необхідно вирішувати потребу захисту ресурсів з огляду на специфіку цієї технології. В зв'язку з цим, актуальним є аналіз існуючих підходів до захисту ресурсів комп'ютерних мереж та експериментальне дослідження ефективності адаптації даних підходів відносно мереж SDN .

**Мета роботи:** Розробка системи захисту ресурсів мереж SDN за рахунок проведення аналізу, порівняння основних методів і алгоритмів, які використовуються в системах захисту ресурсів мереж SDN, проведення порівняння розповсюджених систем захисту ресурсів мереж SDN, підвищення функціональної ефективності та надання відповідних практичних рекомендацій. Для цього визначено наступні задачі, які вирішуються в даній роботі:

1. Провести аналіз технології SDN, можливих загроз та способів захисту.
2. Зробити порівняльну характеристику розповсюджених систем захисту ресурсів мережі SDN.
3. Розробка системи ресурсів мереж SDN.
4. Експериментально дослідити ефективність створеної системи захисту.

*Об'єкт дослідження* – захист ресурсів мереж SDN.

*Предмет дослідження* – система захисту ресурсів мереж SDN, оцінка функціональної ефективності та аналіз існуючих систем захисту комп'ютерних мереж, методи і алгоритми шифрування даних та

автентифікації пристроїв в мережах, критерії їх порівняння, узагальнена модель мережі SDN, обґрунтування вибору елементів системи захисту ресурсів.

**Методи досліджень** – для проведення аналізу загроз для програмно-конфігуровних мереж було використано метод систематизації, а для експериментального дослідження ефективності створеної системи захисту ресурсів мережі SDN було використано метод експериментального моделювання.

**Наукова новизна одержаних результатів** полягає в:

1. На основі проведеного аналізу технології програмно-конфігуровних мереж, способів атак та існуючих систем захисту було виявлено, що актуальним є питання забезпечення надійної та коректної роботи цих мереж за рахунок створення комплексної системи захисту.
2. Розроблено комплексний спосіб та систему захисту ресурсів комп'ютерної мережі, яка відрізняється від наявних тим, що носить більш комплексний характер та захищає одночасно від декількох типів атак, а саме – підслуховування та спроба авторизації нелегітимного пристрою в мережі.
3. Промодельована робота тестової мережі у випадку коректного функціонування, імітовано атаки типу підслуховування та спроби автентифікації нелегітимного пристрою в мережі, та показано як система реагуватиме на дані атаки.

**Практична цінність одержаних результатів** полягає в тому, що вони можуть бути використані для вибору способу та системи захисту ресурсів мережі SDN.

**Апробація роботи.** Основні положення і результати роботи представлені та обговорені на:

XI конференції молодих вчених «Прикладна математика та комп'ютинг – ПМК-2018-2» (Київ, 14 – 16 листопада 2018 року);

V міжнародній науково-технічній internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами» (Київ, 22 – 23 листопада 2018 року).

### **Публікації.**

За результатами магістерської дисертації було опубліковано 2 наукові праці, з них 2 тези доповідей.

**Структура та обсяг роботи.** Магістерська дисертація складається зі вступу, трьох розділів, висновків та 4-х додатків.

У *вступі* подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи.

У *першому розділі* наведено аналітичний огляд літературних джерел технології SDN, її загальні характеристики, особливості, структуру та архітектуру.

У *другому розділі* наведено та проаналізовано можливі загрози, а також способи захисту елементів мереж SDN.

У *третьому розділі* розроблено систему захисту ресурсів та проведено її тестування. Наведено та проаналізовано результати експериментального дослідження ефективності створеної системи.

У *висновках* представлені результати проведеної роботи.

Робота представлена на XX аркушах, містить посилання на список використаних літературних джерел.

Ключові слова: мережа SDN, програмно-конфігуровна мережа, ресурси мережі, система захисту, шифрування, автентифікація.

## ABSTRACT

**Actuality of theme.** Resource protection needs to be managed for any type of network. Nowadays, SDN networks (software-defined networking), which have the advantages of flexibility and scalability, in comparison with conventional networks, becomes more and more popular. But this type of networks is very structurally different from conventional networks, so it requires resource protection based on the specification of the technology. Therefore, it is actual question to analyze existing methods for protecting computer resources of the network and experimental research of effective adaptation of these methods for SDN.

**Purpose:** developing a system for protecting the resources of SDN by analyzing, comparing the basic methods and algorithms that are used in protection of resources systems for SDN, comparing existing protecting resources systems for SDN, improving the functional efficiency and providing relevant practical recommendations. To do this, the next tasks that are solved in this work are determined:

1. Conduct an analysis of SDN technology, possible threats and methods of protection.
2. To make a comparative description of the distributed systems of protection of SDN resources.
3. Development of the system of protection for SDN resources.
4. Experimentally investigate the effectiveness of the created protection system.

*The object of the research* – the protection of SDN network resources.

*The subject of research* – the system of protection for SDN resources, evaluation of functional efficiency and analysis of existing systems of protection for computer networks, the methods and algorithms of encryption and authentication of devices in networks, model of SDN, reasoning of choice of elements for resource protection systems.

**Methods of researches** – for realization of comparison of threats for software-defined networks, the method of systematization was used, and for experimental research of the efficiency of the created system of protection resources for SDN the experimental modeling method was used.

**The scientific novelty of the results is:**

1. Based on analysis of the technology of software-defined networks, methods of attack and resource protection systems was found out that it is relevant to ensure the reliable and correct work of these networks through the creation of a complex protection system.

2. Developed complex method and system of resource protection that differs from the existing ones in the way that it is complex and protecting simultaneously from several types of attacks, namely, eavesdropping and attempt to authorize illegitimate devices in the network.

3. The simulation of the test network in case of correct functioning, simulated attacks eavesdropping and trying of authentication of the illegitimate device in the network and it is shown how the system will respond to these attacks.

**The practical value of the results** is that they can be used to select the method and system of protection of the resources of the SDN network.

**Approbation.** The main provisions and results of work were presented and discussed at:

V International scientific and technical Internet-conference "Modern methods, information, software and technical support of control systems organizational and technical and technological complexes "NUFT (Kyiv 22 -23 November).

XI and scientific conferences undergraduates and graduate students "Applied mathematics and computing - AMC'2018" (Kyiv, 14 - 16 November 2018).



**Publications** According to the results of the master's dissertation, 2 scientific papers, 2 of them abstracts, are published.

**Structure and scope of work.** The master's dissertation consists of an introduction, three sections, conclusions and 4 annexes.

The *introduction* gives a general description of the work, assesses the current state of the problem, substantiates the relevance of the research direction, formulates the purpose and objectives of the research, shows the scientific novelty of the results obtained and the practical value of the work.

The *first section* provides an analytical review of the literature on SDN technology, its general characteristics, features, structure and architecture.

The *second section* presents and analyzes possible threats, as well as methods of protection of SDN elements.

In the *third section* the developed system of resource protection and its testing is described. The results of an experimental research of the effectiveness of the created system are presented and analyzed.

The *conclusions* contains the results of the work.

The work is presented on XX pages, contains a link to the list of used literary sources.

Keywords: SDN, software-defined networking, network resources, security system, encryption, authentication.

## Зміст

Реферат.....	4
Список умовних позначень, термінів та скорочень.....	12
Вступ.....	14
1.1. АНАЛІТИЧНИЙ ОГЛЯД ТЕХНОЛОГІЇ ПРОГРАМНО- КОНФІГУРОВНИХ МЕРЕЖ.....	16
1.1 Архітектура SDN.....	16
1.2 Протокол OpenFlow.....	19
1.3 Вразливості архітектури SDN.....	32
1.4 Вразливості протоколу OpenFlow.....	35
Висновки до розділу 1.....	38
2 ОГЛЯД ЗАГРОЗ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ ТА МОЖЛИВИХ ПРОТИДІЙ НИМ В SDN МЕРЕЖАХ.....	39
2.1 Аналіз загроз в SDN.....	39
2.1.1 DDoS/DoS-атаки.....	39
2.1.2 Компрометація контролера.....	41
2.1.3 Шкідливі додатки.....	42
2.1.4 Атаки на зв'язок рівнів управління та передачі даних.....	43
2.1.5 Атаки підслуховування.....	44
2.1.6 Порівняння атак.....	45
2.2 Способи захисту елементів мереж SDN.....	46
2.2.1 Пом'якшення наслідків DDoS/DoS-атаки.....	46
2.2.2 Захист від скомпрометованих контролерів.....	50
2.2.3 Захист від шкідливих додатків.....	55
2.2.4 Захист рівня управління та зв'язку з рівнем передачі даних.....	58
2.2.5 Захист від атак підслуховування.....	59

2.2.6 Порівняння методів захисту у мережах SDN.....	61
Висновки до розділу 2.....	63
3. РОЗРОБЛЕНА СИСТЕМА ЗАХИСТУ.....	64
3.1. Опис елементів системи захисту.....	64
3.2. Тестування розробленої системи.....	83
Висновки до розділу 3.....	94
Висновки до роботи.....	95
Список використаної літератури.....	96
Додатки	
Додаток 1. Лістинг програми	
Додаток 2. Копії тез доповідей	
Додаток 3. Презентація	
Додаток 4. Довідка про впровадження	

### Список термінів та скорочень

AES	- Advanced Encryption Standard;
API	- application programming interface;
ASIC	- application-specific integrated circuit;
BFD	- Bidirectional Forwarding Detection;
DDoS	- Distributed Denial of Service;
DoS	- Denial of Service;
EAP	- Extensible Authentication Protocol;
ECMP	- Equal-cost multi-path routing;
Ethernet	- найпопулярніший протокол кабельних комп'ютерних мереж, що працює на фізичному та каналному рівні мережевої моделі OSI;
FIB	– таблиця форвардингу;
Github	- один з найбільших веб-сервісів для спільної розробки програмного забезпечення.
HMAC	- hash-based message authentication code;
IDS	- Intrusion Detection System;
IS-IS	- протокол маршрутизації проміжних систем;
IT	- Information Technology;
JSON	- JavaScript Object Notation;
LACP	- link aggregation control protocol;
LSP	- label switch path;
MITM	- Man in the middle;
MPLS	- multiprotocol label switching;
NAT	- Network Address Translation;
NETCONF	– мережевий протокол конфігурації;
NOS	– Network Operation System;
ONF	- Open Networking Foundation;

OpenFlow	– протокол управління процесом обробки даних, що реалізує технологію SDN
OSPF	- Open Shortest Path First;
PBB	- Provider Backbone Bridging;
PDU	- Protocol Data Unit;
SDN	- software-defined networking;
STP	- Spanning Tree Protocol;
TCAM	- ternary content-addressable memory;
TCP	- Transmission Control Protocol;
TLS	- transport layer security;
TTP	- Table Type Patterns;
UDP	- User Datagram Protocol;
VLAN	- Virtual Local Area Network;
ПКМ	– Програмно-Конфігуровна Мережа.

## Вступ

Основними причинами розвитку сучасних мережевих технологій є підвищені вимоги до якості обслуговування і безпеки безперервно зростаючого трафіку. Традиційні мережі передачі даних мають складну структуру і важкі в управлінні, оскільки для впровадження глобальної мережевої політики доводиться окремо виконувати налаштування кожного мережевого пристрою, що призводить в тому числі до ризиків, пов'язаних з некоректною конфігурацією. Відзначимо, що традиційні функції автоматичної реконфігурації великої мережі в разі її зміни або несправності спираються на механізми, що знижують продуктивність мережевої інфраструктури, оскільки протоколи, які реалізують реконфігурацію, підвищують навантаження на мережеві пристрої, ускладнюючи і програмне, і апаратне забезпечення.

Термін Software Defined Networking (SDN) або програмно-конфігуровні мережі відносно не новий. Перші роботи і дослідження з даної тематики з'являються починаючи з 1995 року [1]. Технологія з'явилася у відповідь на потреби розподілених обчислень, використання мережевих технологій в хмарах (де в мережах кількість маршрутизаторів, якими треба керувати, може досягати декількох десятків тисяч), поява великих дата-центрів, початок віртуалізації інтернету [2]. По суті, це нова архітектура, яка змінює організацію традиційних мереж передачі даних. У мережах SDN основні функції комутаторів і маршрутизаторів перенесені на центральний мережевий контролер, що спрощує і застосування мережевих політик, і моніторинг стану мережі. При такому підході передавальні пристрої відповідають тільки за передачу даних, спираючись на таблицю потоків, яка будується централізованим мережним контролером, котрий взаємодіє з передавальним пристроєм [3].

Традиційно при впровадженні нових ІТ технологій з деяким запізненням виявляються і нові проблеми з інформаційною безпекою. У

цій роботі на основі аналізу наявних підходів до визначення і протидії загрозам для мереж SDN та протоколу OpenFlow ми визначаємо напрямки дій щодо захисту такого типу мереж.

## **РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ТЕХНОЛОГІЇ ПРОГРАМНО-КОНФІГУРОВНИХ МЕРЕЖ**

### **1.1 Архітектура SDN**

Архітектура програмно-конфігуровної мережі (ПКМ, SDN) задається чотирма основними принципами [4].

Функції управління та передачі даних розділені. Мережевий пристрій стає простим передавальним пристроєм без функцій управління.

Рішення про передачу даних приймаються на основі потоків, а не адреси призначення. Потік являє собою поєднання полів заголовка пакету, що діють як фільтр, котрий співставляє мережевий трафік з набором інструкцій по його обробці. У контексті SDN абстракція потоку використовується для забезпечення однакової обробки і передачі на мережевих пристроях для кожного пакета в послідовності від джерела до місця призначення.

Контролер SDN, або мережева операційна система, є зовнішнім об'єктом для передавальних пристроїв. Контролер SDN - це програмна служба, яка надає ресурси і інтерфейс для розробки програмних розширень, які використовують переваги централізованого управління мережею і обмежені тільки набором критеріїв і дій передавальних пристроїв.

Програмне забезпечення, запущене на мережевому контролері, взаємодіє з передавальними пристроями і відкриває можливості програмування функцій мережевої інфраструктури. Це фундаментальна властивість SDN, яка є основною перевагою даної архітектури.

На підставі визначення архітектури SDN мережева інфраструктура ділиться на три рівні. Рівень передачі даних і рівень управління є результатом спрощення передавальних пристроїв і перенесення функцій управління на централізовану програмну платформу. Третій рівень (рівень додатків) є абстракцією, тісно пов'язаною з можливістю програмування



мережевого контролера і його здатністю взаємодіяти з іншими додатками. В результаті архітектура SDN мають наступну структуру [5] .

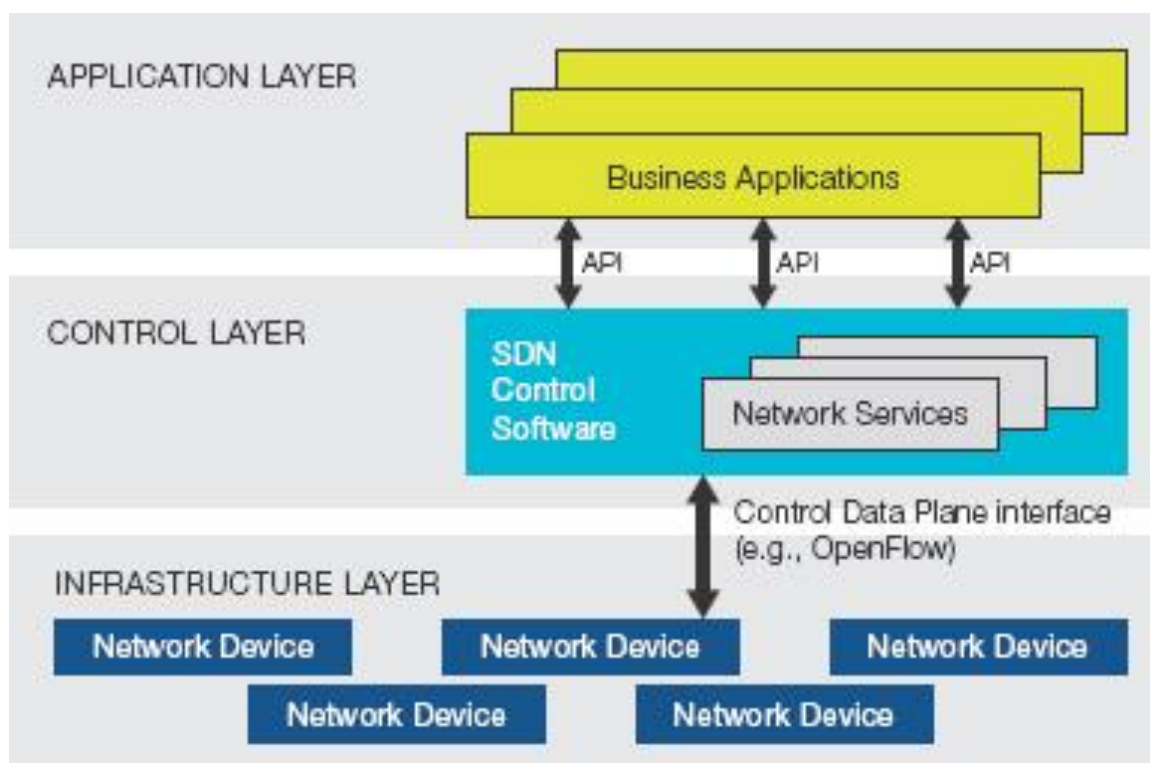


Рисунок 1 – Схематичне зображення структура SDN

Рівень передачі даних - передавальні пристрої, включаючи як фізичні, так і віртуальні комутатори, які доступні через уніфікований інтерфейс і виконують інструкції таблиць потоків для передачі мережевого трафіку.

Рівень управління складається з мережевого контролера або набору мережевих контролерів, що забезпечують функції управління мережевою інфраструктурою. Взаємодія з пристроями рівня передачі даних - через уніфікований програмний інтерфейс. На даному рівні є три інтерфейси для взаємодії з іншими елементами мережевої інфраструктури: південний інтерфейс (для взаємодії з рівнем передачі даних), північний інтерфейс (для взаємодії з рівнем додатків), східний та західний інтерфейс (для взаємодії між елементами рівня управління).

Рівень додатків складається з високорівневих додатків, які взаємодіють з мережевим контролером або набором контролерів для реалізації конкретних функцій, що відповідають вимогам мережевої інфраструктури.

Взаємодія між мережевим контролером і передавальними пристроями реалізується за допомогою програмного інтерфейсу, який використовується для прямого управління групами пристроїв. Найбільш розвиненим програмним інтерфейсом на даний момент є протокол OpenFlow [6]. Архітектура OpenFlow-комутатора базується на одній або декількох таблицях правил, що визначають механізм обробки потоків мережевого трафіку. Кожне правило є записом в таблиці OpenFlow-комутатора. Запис зіставляється з певним потоком трафіку. Залежно від результату зіставлення застосовується відповідна дія (блокування, передача, модифікація тощо) до пакетів з даного потоку.

Залежно від набору правил, встановлених мережевим контролером, комутатор OpenFlow може виконувати роль маршрутизатора, комутатора, брандмауера та інших елементів традиційної мережевої інфраструктури. Важливим наслідком використання OpenFlow є гнучкість, привнесена централізованим управлінням мережевої інфраструктури, яка веде до спрощення створення і модифікації конфігурації мережевої інфраструктури. Крім цього, ключовою перевагою архітектури SDN є можливість розширення функцій мережі за рахунок інтеграції різних програмних модулів (в тому числі призначених для користувача) в мережевий контролер.

Архітектура SDN відкриває ряд можливостей для інновацій в галузі управління процесом передачі даних і забезпечення інформаційної безпеки. Поєднання програмованості мережі і централізованої точки управління дозволяє вдосконалити сучасні засоби і системи захисту даних, розширивши їх функціональні можливості і підвищивши продуктивність

[7]. Механізми захоплення трафіку і збору статистики, активовані на мережевих пристроях, дозволяють формувати дані, які регулярно передаються на мережевий контролер. Додатки, що функціонують на контролері, можуть обробляти і аналізувати дані, одержувані з усієї мережі. На підставі результатів аналізу нові або модифіковані політики можуть бути застосовані на мережевих пристроях. Даний підхід може значно збільшити ефективність контролю і протидії загрозам безпеки мережі.

## 1.2 Протокол OpenFlow

OpenFlow є основним використовуваним стандартом, який визначає процес взаємодії між рівнем передачі даних і рівнем управління архітектури SDN. OpenFlow це відкритий інтерфейс для управління передавальними пристроями мережевим контролером. Програмовані мережеві контролери дозволяють встановлювати правила обробки і передачі даних, спираючись на критерії, які є рідкісними для традиційних мереж, такі як дії користувачів, поведінку додатків або ресурсів інфраструктури [8].

OpenFlow спочатку був розроблений для мереж Ethernet, але може бути адаптований для розгортання в мережах інших типів. Мережеві пристрої можуть підтримувати як OpenFlow, так і традиційний метод передачі даних, що значно спрощує Інтеграцію протоколу OpenFlow в корпоративні системи [9]. OpenFlow широко впроваджується виробниками мережевого обладнання через просту (а отже, і більш дешеву в реалізації) структуру OpenFlow-комутатора, яка може бути реалізована за рахунок невеликих модифікацій програмного і апаратного забезпечення. В результаті перехід на протокол OpenFlow є відносно легким і може бути проведений покроково з впровадженням протоколу в ті мережеві сегменти, які вимагають функцій OpenFlow.

Логічна структура комутатора OpenFlow базується на двох основних елементах:

- таблиці (таблиці потоків і таблиці груп), які використовуються для зіставляючи-ня переданих пакетів з діями, які необхідно до нього застосувати;
- інтерфейс управління OpenFlow, який використовується для взаємодії з одним або декількома мережевими контролерами [6]. Контролер через інтерфейс управління OpenFlow створює і оновлює записи в таблицях потоків і відповідає на запити комутатора (рисунок 2).

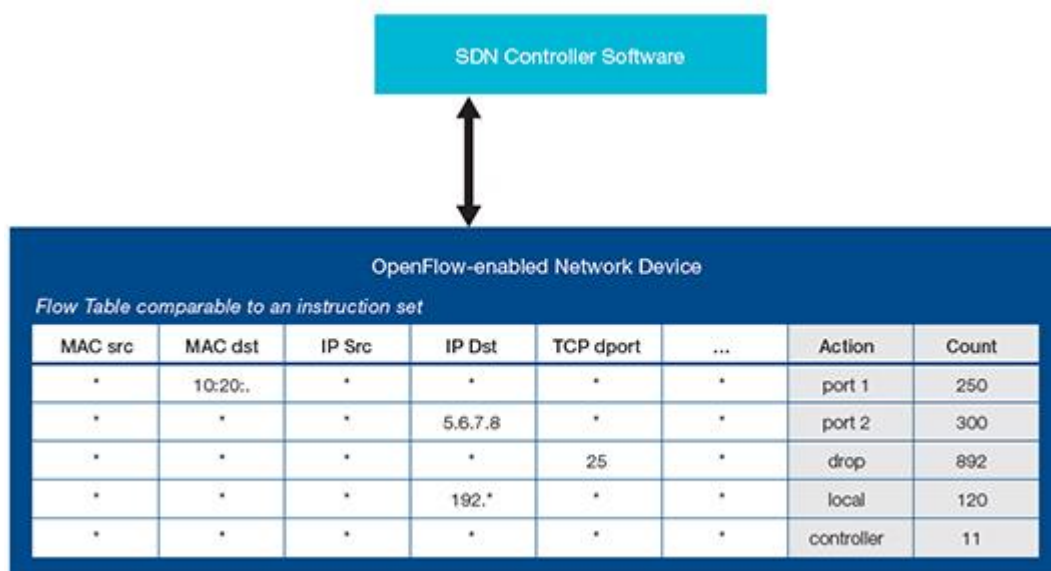


Рисунок 2 – Приклад набору команд OpenFlow в таблиці переходів

Розвиток OpenFlow пройшов через ряд версій - від версії 1.0 до 1.4. Більшість виробників мережевого устаткування спочатку реалізували підтримку версії 1.0, яка мала безліч обмежень і проблем з масштабованістю. Вона показала, що ONF запропонував ринку неповноцінний продукт. Версії 1.1 і 1.2 вважаються перехідними і практично ніхто з виробників не реалізує їх підтримку.

Найбільш перспективною вважається версія 1.3, в яку включена підтримка MPLS міток, per-flow лічильників, Provider Backbone Bridging

(PBB) і ще деяких корисних функцій. За результатами тестування великої кількості комутаторів різних виробників, можна сказати, що підтримка версії 1.3 на «залізних» комутаторах поки дуже обмежена.

OpenFlow не є повністю «відкритим» протоколом. Його розробка контролюється закритою групою, що складається приблизно з 150 компаній, які формують ONF. Робота ведеться в прихованому від широкої публіки режимі і її результат видно тільки після публікації нової версії в якості стандарту.

Практично кожний традиційний мережевий пристрій складається з трьох незалежних рівнів: рівня даних (data plane), рівня контролю (control plane) та рівня управління (management plane).

Data plane відповідає за пересилку Protocol Data Unit (PDU) відповідно до певних правил, що зберігаються в таблиці. Для рівня L2 це може бути MAC-таблиця, що складається з MAC-адрес і відповідних вихідних портів. На вхідний порт приходять PDU, обробляється і відправляється через вихідний порт. Data plane не відповідає за формування таблиці форвардинга (FIB); Control plane відповідає за надання необхідної інформації Data plane для забезпечення форвардинга і є «мозком», який приймає рішення про те, яким чином перенаправляти PDU. Control plane не обмежується лише протоколами маршрутизації. Будь-який протокол, **що** контролює взаємодію між суміжними вузлами, зазвичай є протоколом Control plane. Прикладом можуть служити BFD, STP і LACP. Ці протоколи не взаємодіють безпосередньо з FIB. Наприклад, протокол BFD виявляє фізичні проблеми на каналі зв'язку між пристроями, але самостійно нічого не видаляє з таблиці. Він інформує інші більш високорівневі протоколи про збій, надаючи їм самим прийняти рішення про зміну FIB. З іншого боку, протоколи маршрутизації OSPF або IS-IS безпосередньо впливають на таблицю маршрутизації і форвардинга;

Management plane надає інтерфейс управління і моніторингу, за допомогою якого проводиться конфігурація пристрою.

Ідея OpenFlow укладається в виносі Control plane на окремий пристрій - OpenFlow-контролер, залишаючи мережевих пристроїв лише функції Data Plane. Це сильно знижує вимоги до функціоналу обладнання, перетворюючи його в «простий пристрій по перенаправлення пакетів з порту в порт».

Контролер використовує OpenFlow, щоб підключатися до комутаторів і програмувати їх таблиці форвардинга, розраховуючи кращі шляхи для трафіку додатків. OpenFlow-агент приймає команди від контролера і формує таблицю flow, на основі якої комутатор приймає рішення про те, куди направити пакет, що прийшов. Тут під «пакетом» розуміється PDU (рисунок 3).

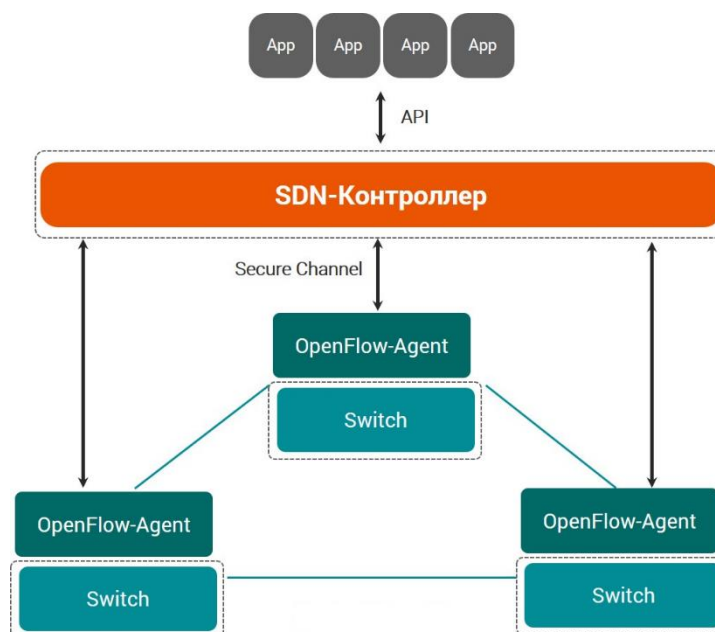


Рисунок 3 – Загальний вигляд SDN з OpenFlow

Отже, коли розглянуто основний принцип SDN - відділення data plane від control plane, необхідно пояснити основні терміни, якими оперує OpenFlow. Data Plane комутатора складається з наступних компонентів (рисунок 4):

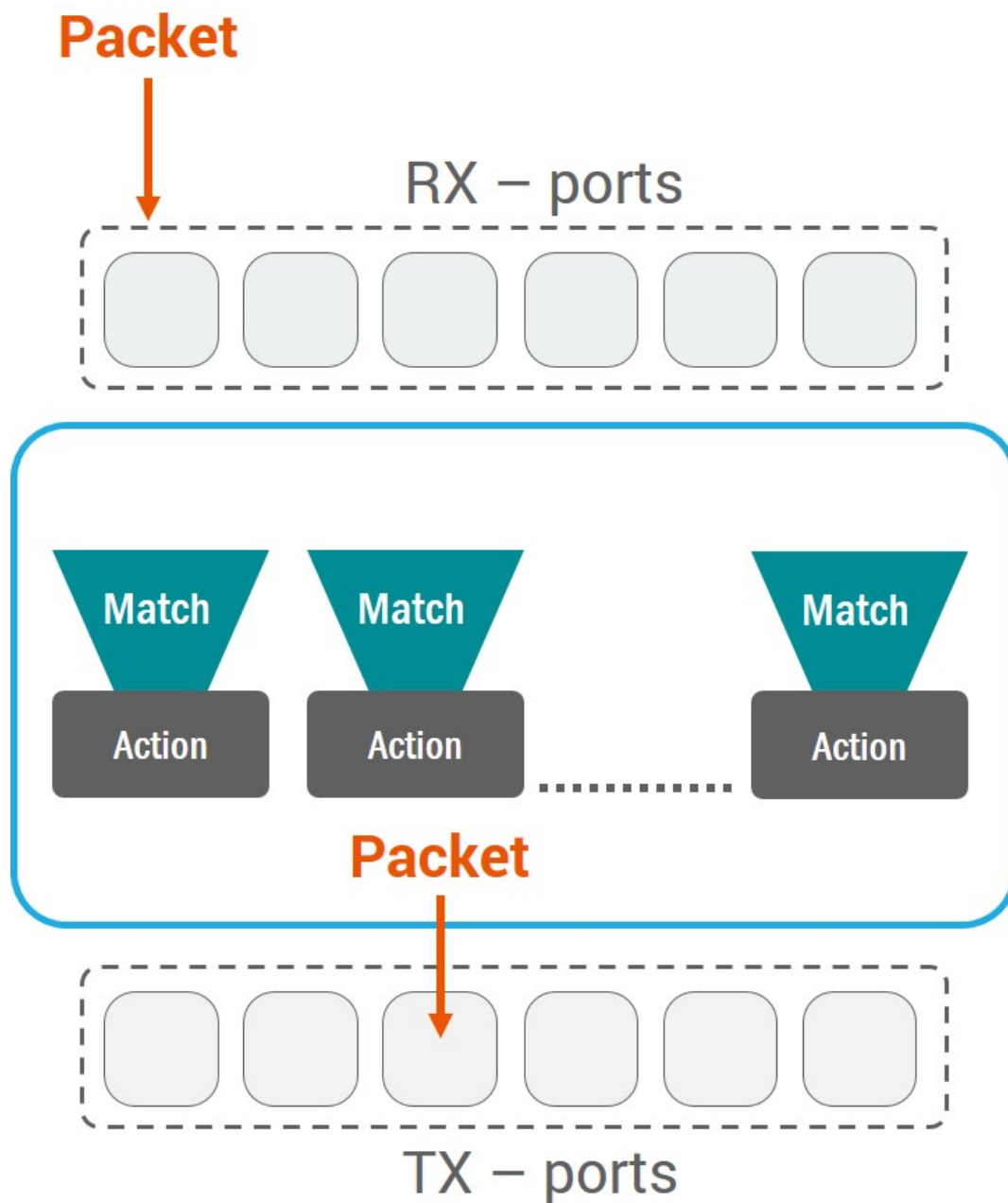


Рисунок 4 – Data Plane комутатора

Data Plane містить наступні елементи:

- OpenFlow-порти (Ports);
- потоки (Flow);
- таблиці потоків (Flow tables);

- класифікатори (Match);
- дії (Actions).

Пакети, що надходять на OpenFlow-порт, класифікуються за Flow в таблицях потоків за допомогою Match-класифікаторів. Кожен Flow складається з набору дій (Actions), які застосовуються до кожного пакету, який відповідає правилу.

OpenFlow-порти виконують ті ж функції, що і порти стандартного комутатора. З точки зору вхідних і вихідних потоків трафіку, немає ніякої різниці зі звичайним L2-комутатором. Вхідний порт одного потоку може бути вихідним для іншого. OpenFlow визначає набір стандартних портів - physical, logical і reserved. Physical - очевидно, фізичні порти комутатора. Логічні (logical) порти, як і в традиційних мережах, безпосередньо не прив'язані до портів обладнання, наприклад, MPLS LSP, Tunnel і Null інтерфейси. Зарезервовані порти використовуються для внутрішньої обробки трафіку і для гібридних типів впровадження (OpenFlow + традиційна мережа).

Зарезервовані порти можуть бути обов'язковими або опціональними. Обов'язкові порти включають порти типів ALL, CONTROLLER, TABLE, IN\_PORT і ANY. У той час як опціональні порти - LOCAL, NORMAL і FLOOD.

OpenFlow - звичайний протокол програмування TCAM. Якщо ви захочете створити тунель між двома кінцевими точками, ви не зможете це зробити за допомогою OpenFlow. Подібні завдання можна виконати за допомогою інших протоколів, наприклад, OF-CONFIG, який використовує NETCONF для цього. Порти можуть бути додані або видалені з конфігурації комутатора за допомогою OF-CONFIG, але не за допомогою OpenFlow.



Зміна стану порту не призводить до автоматичного перенаправлення Flow через альтернативний маршрут. Наприклад, якщо порт перейшов в стан Down, то запис про flow, який використовує цей інтерфейс для відправки, залишиться і всі наступні пакети будуть відкидатися. При зміні стану порту, комутатор спочатку повинен відправити повідомлення на контролер, щоб той вніс необхідні зміни у flow-таблицю комутатора.

Початкова модель форвардинга в OpenFlow була проста. Вона базувалася на тому, що OpenFlow використовує таблиці TCAM на недорогих комутаторах.

При отриманні пакету з нього витягуються метадані (наприклад, incoming interface) і інші поля пакета. Потім ці поля порівнюються з записами в таблиці Flow. Кожен запис в таблиці має поле "protocol", за яким йде порівняння. Результат з найвищим пріоритетом вважається переможцем. Кожен запис в таблиці Flow повинен мати свої пріоритети, а запис з найвищим пріоритетом визначає те, як далі буде оброблений пакет.

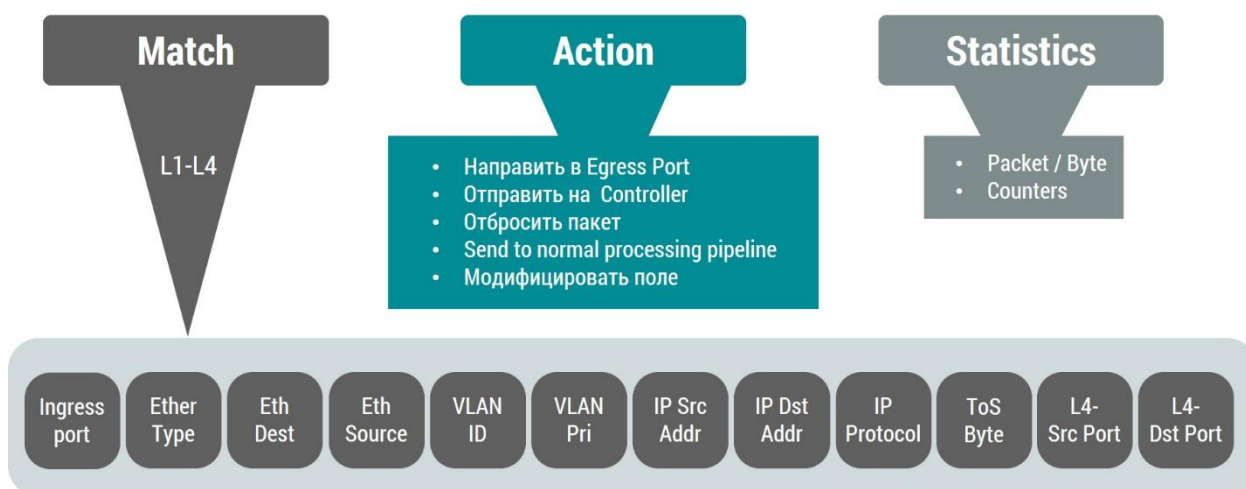


Рисунок 5 – Приклад таблиці Flow

Після того, як було вибрано Match-правило, до пакету застосовується певна дія (Action). Можливими діями можуть бути «відправити в порт X», «відкинути» або «відправити на контролер». За замовчуванням OpenFlow

1.0 відправляє на контролер все пакети, які не відповідають жодному правилу. У пізніших версіях протоколу цей механізм був змінений, так як він міг піддати контролер DoS-атаці з боку зловмисників.

Первісна специфікація OpenFlow 1.0 передбачала лише можливості «перенаправити на інший порт» або «відправити на контролер». Але пізніше прийшли до висновку, що в певних випадках потрібно модифікувати поля в пакеті, наприклад зменшити TTL або додати тег VLAN. Версія 1.0 була визнана неповною, тому що необхідна можливість виконання більш ніж однієї дії конкретним пакетом.

В якості класифікатора можна використовувати будь-яку комбінацію полів заголовка пакету. Наприклад, порівняння з MAC-адресами (OpenFlow 1.0), wildcard-маскам (OF 1.1), VLAN і MPLS-мітках (OF 1.1), PBB заголовкам (OF 1.3), IPv4 адресами з wildcard-масками, DSCP бітам, IPv6 адресами (OF 1.2), L4 протоколам, TCP і UDP портів.

Після того як порівняння виконано, можливе виконання ряду дій. Крім описаних вище, можна виконати, наприклад, header rewrite (зміна заголовка). Це означає, що OpenFlow може бути використаний для реалізації функцій NAT. Щоправда цей випадок теж має свої обмеження в зв'язку з обмеженою кількістю записів flow.

Також існує цікавий механізм OpenFlow, де пакет може бути оброблений з використанням, так званих, груп (GROUP). Починаючи з версій 1.1+ OpenFlow включає функціонал GROUPS, які вдосконалять механізми форвардинга і дозволяють реалізувати, наприклад, ECMP Load Balancing.

Група складається з набору підмножин, кожна підмножина - з набору Actions. В якості одного Action може бути «перенаправити на порт», встановити VLAN-тег або виконати операції push / pop для MPLS-міток. Група, наприклад, може складатися з двох підмножин:

- підмножина 1 (bucket 1) з дією «відправити на port1»;

- підмножина 2 (bucket 2) з діями «відправити на port2» і «додати тег».

Подібний підхід розширює можливості форвардинга. Наприклад, «відправка на всі підмножини з однієї групи» може бути використана при реалізації selective multicast. Або, «відправка на одну підмножину з групи» - для балансування навантаження в LAG або ECMP.

Один з економічних драйверів застосування концепції Software Defined Networking (SDN) пов'язаний з «можливістю використання недорогих bare-metal комутаторів, керованих централізовано за допомогою SDN-контролера». OpenFlow - найбільш популярний на даний момент протокол, який використовується SDN-контролерами для програмування мережевих пристроїв. Але він має кілька важливих технічних недоліків, які гальмують розвиток підходу - невеликі розміри TCAM в обладнанні, складність реалізації функціоналу OpenFlow 1.3 з використанням декількох таблиць в pipeline комутаторах тощо.

Розглянемо підхід OpenFlow Table Type Patterns (TTP), запропонований співтовариством ONF для вирішення проблем сумісності SDN-контролерів і обладнання, а саме - як укласти принципи роботи OpenFlow (версії > 1.0) в логіку, зашиту в ASIC комутаторах.

Основне завдання мережевого пристрою - прийняти пакет / фрейм на одному порту і перенаправити в інший, при цьому обробити його певним чином. За цією обробкою ховається безліч більш примітивних дій - призначення VLAN-міток, зменшення TTL тощо, які повинні бути визначені в комутуючій матриці (ASIC) для підтримки специфічного «Пайплайну» комутатора.

«Пайплайн» (від англ. «Pipeline») - термін, яким прийнято описувати певну послідовність дій, що здійснюються з пакетом всередині мережевого пристрою до моменту його відправки через вихідний порт.

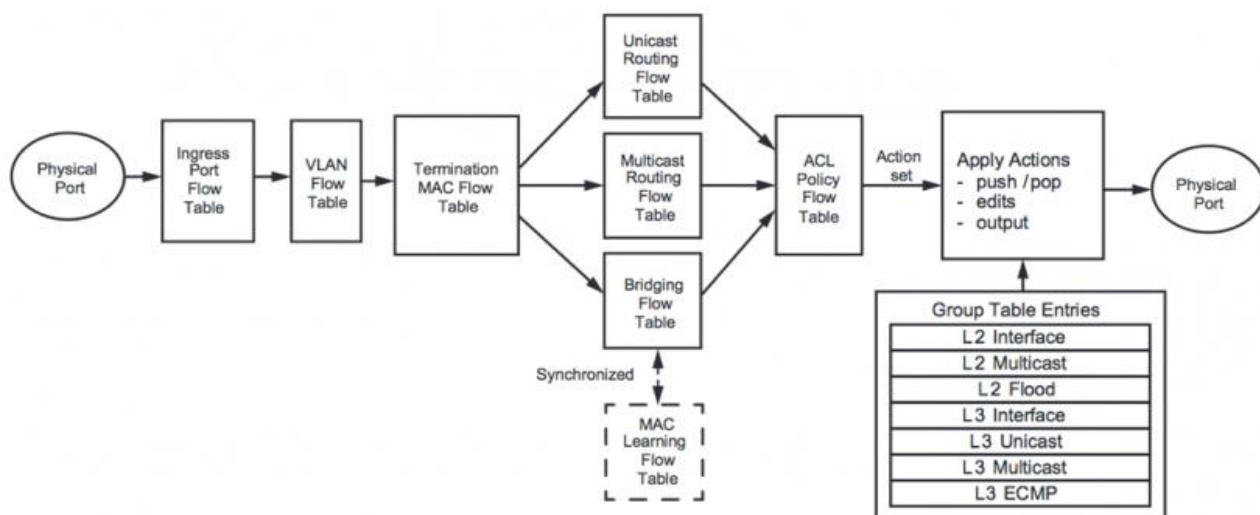


Рисунок 6 – Абстрактний пайплайн для функцій L2 Bridging і L3 Routing

Проектування комутуючих матриць завжди залежить від логіки роботи L2/L3 протоколів для реалізації відповідного Пайплайн. З цього випливає, що термін «Data Plane», який так часто використовується в SDN, завжди залежить від реалізації логіки в апаратному забезпеченні. Плюс до всього, в деяких чіпсетах може бути реалізована і пропрієтарна логіка виконання примітивних дій.

Спочатку, в OpenFlow спробували зробити ці примітиви вендор-незалежним шляхом використання поняття таблиць (TABLES). Теоретично, якщо комутатор підтримує OpenFlow, він повинен мати можливість конфігурувати площину передачі даних з використанням лише примітивів OpenFlow. Кожна дія може бути визначено як таблиця. Версія 1.0 протоколу підтримувала тільки одну таблицю, і логіку OF1.0 було досить легко вписати в пайплайн комутаторів.

Однак, в OpenFlow 1.3 з'явилася підтримка декількох таблиць, наприклад, одна - для призначення VLAN-міток, інша - для ECMP і так далі. У OF1.3 також специфіковане, яким чином ці таблиці можуть взаємодіяти один з одним. В результаті, з'явилося безліч Пайплайн

OpenFlow, які дуже складно зіставити з Пайплайн, закладеним в ASIC комутатора. З програмними комутаторами все набагато простіше, тому що їх логіку можна адаптувати яким завгодно чином.

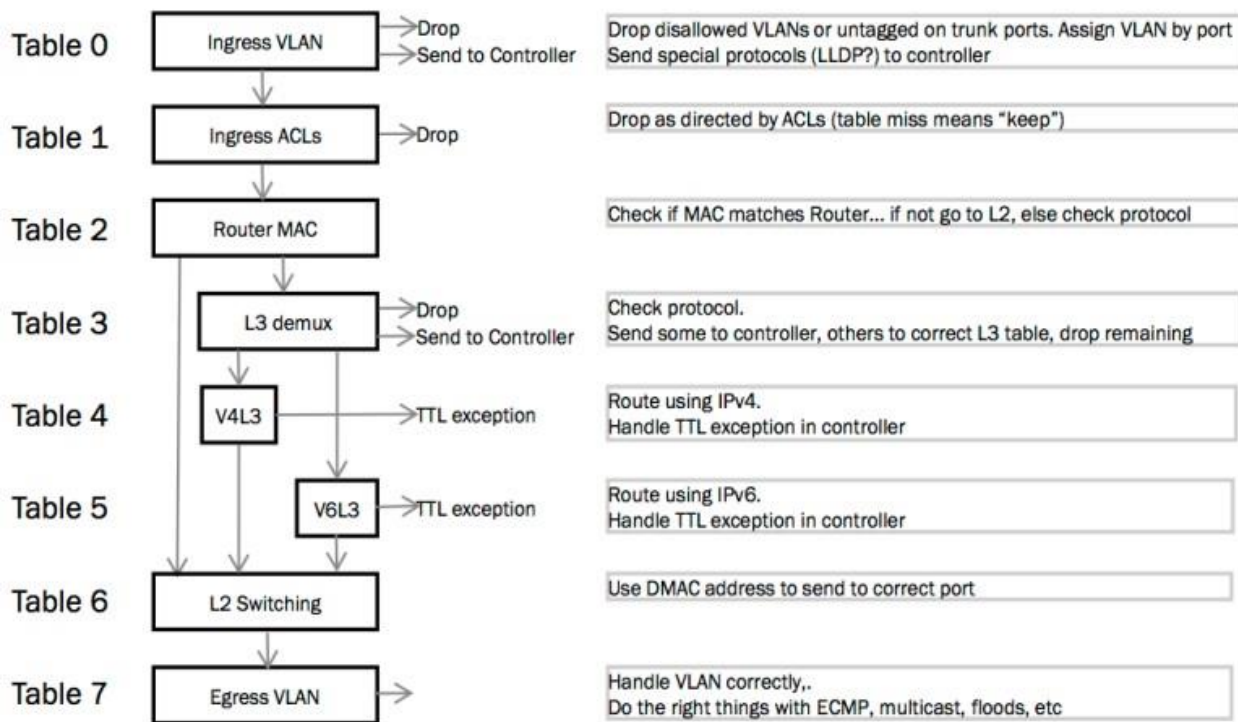


Рисунок 7 – Приклад таблиць в OpenFlow 1.3 (і старше)

Для розуміння Table Type Patterns, спочатку потрібно позначити два важливих моменти.

По-перше, ми знаємо, що OpenFlow-агент на комутаторі конфігурує Data Plane виходячи з інструкцій, отриманих від контролера. Якись інструкції агент не зможе виконати, наприклад, якщо ASIC не підтримує необхідну кількість таблиць або не має потрібної кількості ресурсів. Ці проблеми можуть бути виявлені лише на етапі ініціалізації - додаванні комутатора під управління контролера. Кожен пайплайн OpenFlow повинен бути зіставлений з ASIC з урахуванням підтримуваного

функціоналу чіпсета. На даний момент, у контролерів немає механізму визначення того, які пайплайн підтримуються в ASIC комутатора.

Щоб вирішити це завдання, необхідний процес узгодження між мережним пристроєм і контролером, причому до того моменту, як вони почнуть працювати разом.

По-друге, на етапі ініціалізації необхідно чітко визначення функціоналу OpenFlow, який підтримується контроллером і OF-комутатором - типи повідомлень (вони називаються FlowMod повідомлення), кількість таблиць тощо. Потім, якщо контролер і світч змогли «домовитися», вони починають використовувати лише обмежений функціонал OpenFlow для реалізації підтримуваного пайплайну. Подібний підхід виключає несподівану поведінку мережевого пристрою у відповідь на отримання непідтримуваного типу повідомлення.

У багатьох випадках немає необхідності підтримувати всі можливі пайплайни OpenFlow. Можна визначити якийсь обмежений, зі зрозумілим описом OF-повідомлень та таблиць. Якщо контролер і мережевий пристрій обидва його підтримують, можна сказати, що вони сумісні. Цей «обмежений pipeline» і отримав назву Table Type Patterns (TTP).

TTP націлені на вирішення двох вищеописаних завдань. Фактично, TTP є описом в форматі JSON, що містить інформацію про кількість таблиць, типи повідомлень і про те, які дії робляться з пакетом при переході з однієї таблиці в іншу. Кожен випадок застосування OpenFlow 1.3 може бути описаний в рамках окремого TTP.

Для того, щоб уникнути отримання комутатором незрозумілих йому директив від контролера, спочатку повинен бути визначений підтримуваний TTP з обох сторін, після чого відбувається переведення

комутатора в робочий режим і оновлення flow-записів. Процес узгодження відбувається за допомогою протоколу OF-Config. Але звідки SDN-розробники і вендори комутаторів візьмуть ці TTP? Спільнота ONF передбачає, що процес створення TTP відбуватиметься приблизно таким чином:

Спочатку ініціатор процесу (постачальник обладнання, співтовариство ONF або розробник SDN-додатку), бажаючи отримати підтримку логіки роботи свого продукту від інших учасників ринку, створює новий опис TTP.

Після цього він ділиться цим TTP або тільки з партнерами, або з усіма, викладаючи його на Github.

Далі, постачальники обладнання і SDN-розробники, додають підтримку нового TTP в свій продукт. Сама реалізація залишається на вибір кожної зі сторін.

При додаванні комутатора під управління контролера, обидва пристрої визначають підтримувані TTP і починають працювати тільки в рамках цих TTP.

Такий підхід приносить кілька переваг. По-перше, це підвищення сумісності SDN-контролерів і різного комутаційного обладнання. По-друге, - незалежність розробки SDN-додатків і логіки Data Plane, при цьому TTP використовується як референсний документ. По-третє, спрощення тестування OpenFlow пристроїв.

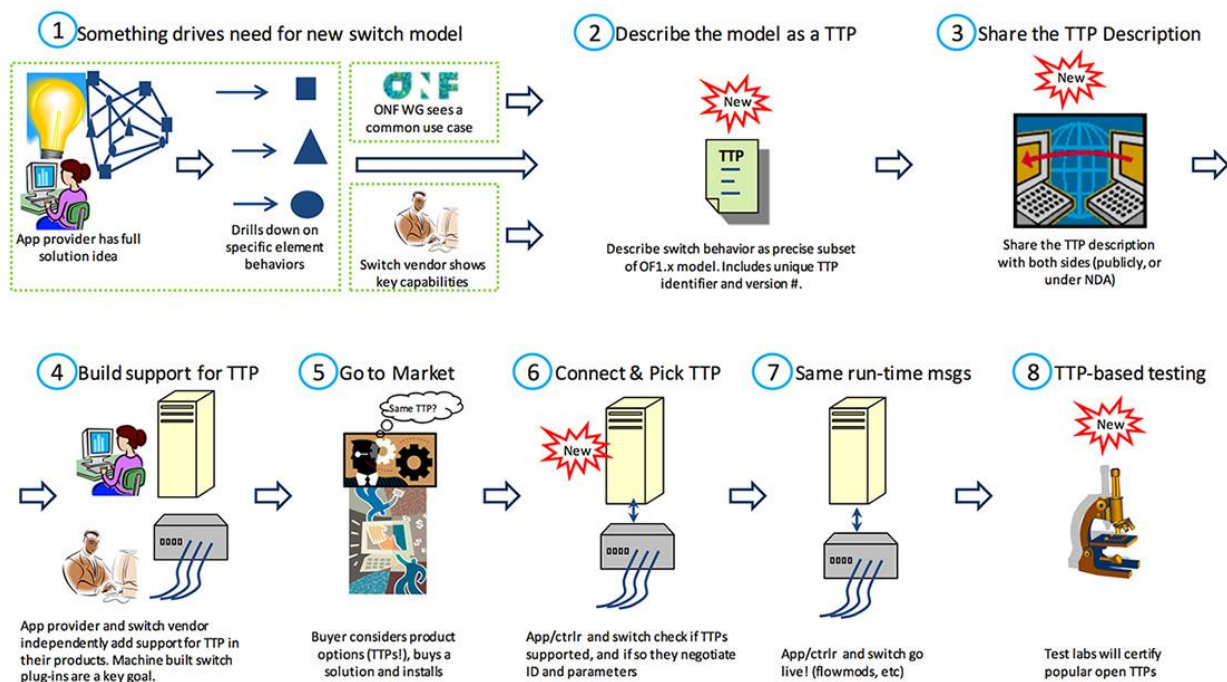


Рисунок 8 – Процес створення ТТР

### 1.3 Вразливості архітектури SDN

Архітектура SDN, припускаючи істотно інший підхід до реалізації мережевої інфраструктури, не позбавлена потенційних вразливостей з точки зору інформаційної безпеки. Необхідність поділу доступу мережевих додатків при роботі з контролером, питання автентифікації і авторизації при роботі додатків з контролером - це лише деякі аспекти безпеки, які доводиться приймати до уваги при проектуванні SDN-мереж.

Контролер як ключовий компонент в управлінні всією інфраструктурою SDN є найбільш вразливим елементом, атака на який може спричинити критичні для всієї інфраструктури наслідки [10]. Поділ доступу мережевих додатків при їх роботі з контролером SDN - актуальна проблема розмежування зон відповідальності мережевих додатків. Ситуація, коли будь-який мережевий додаток здатний змінювати flow-таблиці будь-якого керованого даним контролером комутатора, не відповідає сучасним вимогам інформаційної безпеки. Різні види додатків вимагають різного рівня доступу, і чим більш детально описані обмеження



кожної програми (відповідно до характеру виконуваного завдання), тим більш надійною буде мережа. Різні моделі розподілу доступу можуть застосовуватися для вирішення цього завдання, наприклад, рольові, мандатні і дискреційні, а також комбінації цих моделей з урахуванням специфіки інфраструктури, що захищається.

Основними загрозами, що виникають з боку мережевих пристроїв, що працюють за принципом програмно-конфігурованої мережі, залишаються варіації таких атак, як «відмова в обслуговуванні», підміна контролера тощо. Перенесення «аналітичної» компоненти мережі на контролер природним чином переносить акцент багатьох атак з мережевого обладнання на ПЗ, що забезпечує функціонування мережі: контролер мережі і мережеві додатки, котрі звертаються до контролера [11].

Найбільш простим і одночасно ефективним способом порушення цілісності роботи мережі SDN є атаки типу «відмова в обслуговуванні». Небезпека атаки впливає з самого алгоритму роботи SDN-комутатора при отриманні невідомого (тобто такого, що не підходить під наявні в flow-таблиці правила) пакета. У такій ситуації можливі два варіанта:

- пакет цілком відправляється на контролер для аналізу;
- пакет залишається в пам'яті комутатора, на контролер відправляються виключно заголовки пакету.

Обидва способи залишають для атакуючого широке поле для ефективної реалізації відмови в обслуговуванні шляхом формування потоку різних пакетів в SDN-мережі.

Роздивимося реакцію мережі в обох вищенаведених випадках.

1. Комутатор починає формування великої кількості повідомлень для передачі невідомих пакетів на контролер. Витрачаються процесорні ресурси комутатора, збільшується витрата пам'яті. Особливо сильно

витрачається пам'ять в тому випадку, якщо комутатор буферизує самі пакети і пересилає контролеру тільки їх заголовки.

2. Потік пакетів від комутатора на контролер навантажує канал зв'язку між контролером і комутатором. Якщо середовище зв'язку розподілюване, то зниження оперативності доставки повідомлень можуть відчутти на собі всі комутатори. Підвищений вплив на канал зв'язку буде надано в ситуації, коли комутатор пересилає пакети для аналізу цілком.

3. Контролер приймає і обробляє потік повідомлень, витрачаючи процесорний час і пам'ять свого середовища виконання. Формування черг повідомлень змусить легітимні повідомлення очікувати своєї черги і знизить оперативність прийняття рішень в мережі.

4. Контролер генерує потік різних повідомлень у відповідь на запити атакованого комутатора. Витрачаються ресурси каналу зв'язку між комутатором і контролерами.

5. Комутатор приймає команди від контролера і виконує їх, витрачаючи ресурси процесора і пам'ять. Якщо команди містять в собі створення нових правил таблиць потоків, то відбувається їх лавиноподібне збільшення, час перевірки кожного нового пакета по таблиці збільшується, зростають витрати на обслуговування такої таблиці, а також можливе переповнення таблиць потоків.

В результаті реалізація атаки може призвести до таких наслідків:

- вичерпання ресурсів комутатора. Легітимні пакети або взагалі не будуть оброблені даним мережевим вузлом, або їх обробка буде супроводжуватися затримками;
- канал зв'язку між контролером і комутатором не забезпечить доставки керуючих повідомлень, будучи завантаженим потоками даних;
- контролер буде перевантажений запитами і не зможе обробляти керуючі повідомлення, викликані легітимним трафіком.

## 1.4 Вразливості протоколу OpenFlow

Переваги і недоліки архітектури SDN для безпеки мережевої інфраструктури широко вивчені. Але оцінка вразливостей архітектури повинна ґрунтуватися не тільки на міркуваннях про теоретичну архітектуру, але і на експериментах і результатах впровадження протоколу OpenFlow в промисловій мережі. В даний час визначені наступні типи загроз для OpenFlow-мереж.

Режими роботи контролера (реактивний або проактивний) можуть бути легко ідентифіковані зловмисником без застосування специфічних підходів і програмного забезпечення. Ідентифікація заснована на затримці першого пакету для нового потоку трафіка і доступна для кожного користувача, що підключений до мережі або використовує сервіси даної інфраструктури з зовнішніх мереж. В результаті деякі атаки можуть використовувати конкретний режим роботи контролера [12].

Наприклад, несанкціонована установка правил обробки на комутаторах, яка призводить до зниження ефективності або порушення роботи мережі, простіше реалізується в реактивному режимі в зв'язку з особливістю підходу контролера до управління таблицями потоків в цьому режимі. У той же час виконання цієї атаки на реактивний контролер можливе, але є більш важкою, так як потрібно комплексна атака і ймовірність швидкого виявлення факту атаки значно вище.

Загрози безпеці, актуальні для більшості інформаційних систем, такі як сканування портів і визначення мережевих служб, є критичними для архітектури SDN через вразливості каналу OpenFlow і наявності великої кількості трафіку управління, що передається між комутаторами і мережевими контролерами. Відзначимо вразливість архітектури SDN до DoS-атак - одним з найнебезпечніших типів атак для архітектури з централізованою точкою управління. Так як передавальні пристрої не мають функцій управління, вони повинні мати стабільне підключення до

мережевого контролера, щоб забезпечувати передачу даних. У реактивному режимі контролера навіть короткий період недоступності OpenFlow-контролера може викликати багато проблем для мережевої інфраструктури, а довгострокове блокування мережевого контролера може бути використано, щоб повністю зупинити обробку мережевого трафіку або зробити більш складну атаку. Наприклад, помилковий мережевий контролер може зайняти місце заблокованого, що ставить під загрозу всю мережеву інфраструктуру.

Іншою потенційною проблемою є вразливість програмної інфраструктури OpenFlow-мережі. Можливість програмування мережі і наявність відкритих програмних інтерфейсів, котрі використовуються для інтеграції з мережевим контроллером, відкривають ще одні двері для появи вразливостей програмного забезпечення. У той час як контролери OpenFlow, розроблені професійними командами і підтримувані великими товариствами, теоретично можуть бути надійно захищеними, програмні модулі для контролера, які розробляються мережевими інженерами для потреб конкретної мережевої інфраструктури, можуть створити непередбачувані вразливості, які вплинуть на безпеку всієї мережі. Ця проблема безпеки є результатом низького рівня стан-стандартизації рівнів управління і додатки в архітектурі SDN [13].

Ще одна загроза, характерна для традиційної мережевої інфраструктури, - атаки з підміною - має більш високий потенціал для мереж SDN, ніж в традиційних мережах. У той час як вся мережа керується централізованим контролером, ризик підміни менеджмент-трафіку в мережі OpenFlow досить високий. Підміна може спричинити несанкціонований доступ до мережевих пристроїв, отримання контролером некоректної статистики або даних про стан мережі, що може позначитися на роботі всієї мережі.

Однією з причин уразливості OpenFlow-мереж до атак підміни є надмірна гнучкість стандарту OpenFlow. Стандарт дозволяє реалізувати взаємодію між мережевим контролером і комутаторами на базі протоколу TCP без шифрування, а підтримка протоколу TLS є необов'язковою для реалізації.

Цікаво, що всі спостережувані загрози безпеки пов'язані не зі специфічними версіями протоколу OpenFlow. Ці загрози актуальні для всіх випущених версій протоколу і, можливо, не можуть бути усунені через фундаментальні особливості архітектури SDN.

## **Висновки до розділу 1**

Розглянуто технологію програмно-конфігуровних мереж, її архітектуру, переваги та недоліки її застосування. На основі протоколу OpenFlow була також розглянута структура протоколів взаємодії між рівнем передачі даних та рівнем управління архітектури SDN. Також були проаналізовані вразливості даного протоколу. Проаналізовані та продемонстровані поведінка мережі під час порушення цілісності її захисту та наслідки до яких це може призвести.

На основі проведеного аналізу продемонстровано, що на сьогоднішній день залишаються невирішеними задачі захисту ресурсів програмно-конфігуровних мереж.

## **2. АНАЛІЗ ЗАГРОЗ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ ТА МОЖЛИВИХ ПРОТИДІЙ НИМ В SDN МЕРЕЖАХ**

### **2.1 Аналіз загроз в SDN**

#### **2.1.1 DDoS/DoS-атаки**

Мережі, що працюють за парадигмою SDN, все ще мають такі ж вимоги до безпеки, що й традиційні мережі, оскільки вони, також можуть використовуватися для пересилання приватної та конфіденційної інформації [13]. SDN повністю змінює архітектуру та внутрішньо-комунікативні аспекти компонентів у мережі - звідси виникає зовсім нова платформа для зловмисників, які виконують атаки, порушуючи безпеку даних мереж. Це призводить до необхідності створення аналогічного рівня безпеки як і у традиційних мереж, але для захисту від загроз іншого характеру [14]. Цей розділ роботи розглядає деякі з цих основних загроз та демонструє важливість захисту від них у мережі SDN.

Численні типи звичайних атак DDoS можуть бути виконані в середовищі SDN, але тут найбільш розповсюдженою є варіація атаки з використанням підроблених записів потоку, які можуть бути використані нападником для атаки контролера та компрометації його доступності. Заваливши контролер запитами на рішення про вибір потоку, атакувальник призводить до того, що обчислювальні ресурси контролеру можуть перевантажуватись, внаслідок чого контролер буде не в змозі впоратися з будь-якими легальними запитами, які він отримує. Націлюючи атаку на централізовану точку контролю (тобто контролер), зловмисник робить всю мережу переважно непридатною для використання. Незважаючи на те, що потоки даних в мережі можуть ще деякий час бути нормально функціонуючими, навіть при «зваленому» контролері, в решті решт пристрої мережі після закінчення термінів дії правил у їхніх таблицях опитають контролер щодо нових правил у мережі, а той, в свою чергу, буде не в змозі розглядати запити. Якщо ж мережеві пристрої не отримають

відповіді на свої запити, вони обнуляють свої таблиці маршрутизації та комутації або ж використовують застарілі версії даних таблиць. В результаті таких дій мережа може «впасти».

На рівні передачі даних можливе використання помилково створених записів, котрі споживають весь простір у таблицях потоку, для флудингу інших пристроїв. Це робить пристрої пересилання нездатними додавати будь-які легітимні записи потоку до своїх таблиць. Це призводить до того, що пристрої не можуть включати нові записи потоків до своїх таблиць, залишаючи мережу в розрізненому стані. Однією з ключових проблем, пов'язаних з пристроями на рівні передачі даних в межах архітектури, визначеної програмним забезпеченням, є те, що пристрої не вміють розрізняти легітимні запити потоку та скомпрометовані. Цей недолік дозволяє зловмисникам здійснювати успішні атаки DoS на рівні передачі даних, заповнюючи поточний буфер комутаторів помилковими запитами .

Незважаючи на те, що супротивник може націлюватися на індивідуальний шлях до даних та намагатись знизити його доступність, набагато більш імовірно, що атака буде проводитись саме на контролер, оскільки так набагато легше створити та поширити доступ до системи в цілому. Перспектива цього може бути потенційно руйнівною, особливо в умовах виробництва, де послуги, якими користуються найчастіше, будуть недоступними для клієнтів та працівників. Крім того, зловмисник може планувати та здійснювати подальші атаки, які можуть бути спрямовані на руйнування цілісності та конфіденційності даних у мережі. З цих причин згадані вище атаки DDoS/DoS-атаки розглядаються в числі найважливіших типів атак.



### 2.1.2 Компрометація контролера

Контролер можна розглядати як централізований "мозок" SDN. Він контролює всю мережу з однієї точки, що робить його, мабуть, найважливішим компонентом архітектури SDN. Зловмисник, якому вдається скомпрометувати контролер, по суті має контроль над всією мережею [15]. Можливість керувати діями контролера дозволить зловмисникові маніпулювати поточними записами будь-яким способом, який він обере, наприклад, зупиняючи певні типи пакетів до того як вони досягнуть кінцевих вузлів, перенаправляючи пакети на свої власні, «захоплені» вузли в інфраструктурі. У зв'язку з цим зловмисник може поставити під загрозу певній пристрій у мережі та забезпечити його функціонування як вузол типу «man-in-the-middle» або «blackhole/greyhole». Це дозволить зловмиснику потенційно знищити, змінити або прочитати вміст будь-якого отриманого ним пакета [16]. Інша можливість полягає в тому, що зловмисник успішно реєструє нелегітимний контролер на рівні управління мережі SDN. За допомогою цього контролера-зловмисника атакуючий може впливати на доступність інших контролерів, змінювати правила, і ефективно зупиняти/маніпулювати роботою додатків на рівні додатків.

Будь-яка атака, яка націлена на контролер в архітектурі SDN, може мати потенційно руйнівні ефекти. Незважаючи на те, що централізований характер і здатність координувати інформацію в один момент може бути дуже зручною для мережевих адміністраторів та програмістів, в невірних руках він може бути використаний для атак на перевірку цілісності управління повідомлень або отримання чутливої інформації про додаток, наявності важливих послуг для користувачів системи та конфіденційність інформації користувача, що використовується програмами на рівні додатків. Для того аби успішно призупинити роботу скомпрометованого контролеру, система захисту має зосереджуватися на забезпеченні

автентичності контролера, перш ніж дозволити йому вносити будь-які зміни в мережу.

### 2.1.3 Шкідливі додатки

Внаслідок використання технології SDN для інтеграції сторонніх додатків виникає проблема шкідливих додатків. Додатки, що демонструють зловмисну поведінку в середовищі SDN, можуть призводити до катастрофічних наслідків, які подібні до тих, що виникають у ситуації, коли піддається компрометуванню контролер. Автентифікацію та авторизацію додатків для роботи в середовищі SDN важко виконати.

Додатки, що займаються глибокою перевіркою пакетів, можуть призвести до потенційних ризиків для мережі, оскільки вони можуть мати можливість опосередковано контролювати всю мережу через інформацію, яку вони зібрали під час перевірки пакетів [17].

Збільшена кількість даних, а також спосіб, який використовується для її централізації, це те, що дає шкідливим додаткам можливість загрожувати цілісності та конфіденційності інформації про користувача та мережу, до якої вони мають доступ.

Захист північного інтерфейсу є складним завданням, оскільки для кожного додатку, який використовує його, може знадобитися доступ до унікального піднабору інформації від контролера. Для того, щоб успішно стежити за цим, потрібно запровадити певну сувору політику щодо доступу до інформації. Це гарантує, що додаток оголошує, яку інформацію він потребує, і зможе отримати доступ лише до цієї інформації. Це може гарантувати, що додатки не крадуть таємну інформацію та не використовують інформацію з інших програм. Слід також забезпечити автентичність, перш ніж додаток зможе зв'язатися з контролером.

### **2.1.4 Атаки на зв'язок рівнів управління та передачі даних**

Інша ключова область SDN, що представляє можливості для нападу, - це зв'язок між рівнем управління та рівнем передачі даних. Специфікація OpenFlow визначає використання TLS (Security Layer Transport) необов'язковим [18], що робить його слабким місцем і сприйнятливим до різних атак, таких як man-in-the-middle та blackhole.

Атака типу man-in-the-middle відбувається, коли зловмисний вузол встановлюється між контролером і пристроями, що знаходяться на рівні передачі даних. Замість прямого пересилання повідомлень безпосередньо до контролера (або навпаки), вузол man-in-the-middle здатний маніпулювати та перевіряти вміст пакетів [19].

Також може бути виконана атака типу blackhole, в якій вузол встановлюється між цільовим пристроєм і контролером і просто скидає будь-які отримані ним пакети, не пересилаючи їх контролеру. Це призводить до розбиття мережевих комунікацій і недоступності надання послуг легітимним користувачам.

Якщо зловмиснику вдається встановити себе як посередника між рівнем управління та рівнем передачі даних, це може потенційно спустошувати всю мережу. Атака типу man-in-the-middle - це атака спрямована на цілісність керуючих повідомлень між мережевими пристроями на рівні передачі даних та контролером. Зловмисник може змінювати керуючі повідомлення та створювати спосіб формування мережі таким чином, щоб вони давали йому певні переваги. З іншого боку, атака типу blackhole – це атака на доступність послуг мережі. Якщо всі повідомлення між мережевими пристроями та контролером не пересилаються через зловмисний вузол, це неминуче призведе до поломки зв'язку, при цьому пристрої на рівні передачі даних не зможуть отримати доступ до контролеру у разі необхідності. Цей зв'язок між рівнем управління та рівнем передачі даних, безумовно, є слабкою точкою, і

виступає в якості цілі для атаки зловмисників. Тому надзвичайно важливо, щоб він був добре захищеним, перш ніж мережа SDN почне використовуватися в налаштуваннях виробництва.

### **2.1.5 Атаки підслуховування**

Нападники, які намагаються отримати незаконний доступ до мережі SDN або зупинити доступ до послуг які вона надає, можуть використовувати атаки підслуховування (акт незаконного захоплення та перевірки пакетів, що передаються через з'єднання) на певні зв'язки в мережі [20]. Це може дозволити їм зібрати важливу інформацію, яка потім може бути використана для здійснення більш небезпечних атак. Атаки, що використовують підслуховуючий спосіб, тривалий час виконувалися в традиційних мережевих налаштуваннях - бездротові архітектури особливо слабкі до таких атак через їх передачу через повітря. Проте, в контексті SDN, підслуховування можна здійснювати для перевірки пакетів, що використовуються при зв'язку між рівнем управління та рівнем передачі даних, або ж виключно на рівні передачі даних. На рівні передачі даних, пристрій, що використовується у режимі прослуховування, інтегрований у комутаторі OpenFlow, може бути використаний зловмисником (який міг скомпрометувати комутатор), щоб перевірити пакети, передані навколишніми комутаторами, що дозволяє зловмисникам вивчати важливу інформацію. У певному сенсі прослуховування, яке здійснюється на рівні управління та рівні передачі даних, є більшою мірою пасивною атакою та не впливає безпосередньо на доступність, конфіденційність чи цілісність даних. Проте, це дає можливість для подальших атак, які погіршують ці аспекти безпеки.

В контексті прослуховування, яке здійснюється у зв'язку між рівнями додатків та управління, конфіденційність інформації може бути скомпрометована. Зловмисник може вивчати інформацію, яка стосується

певного користувача, якщо їм вдалося прослухати підключення, передаючи важливі дані додатків. Це робить підслуховування особливо небезпечним типом атак - необхідно забезпечити конфіденційність, перш ніж критичні додатки, які містять конфіденційну інформацію, можуть бути розгорнуті в середовищі SDN.

### 2.1.6 Порівняння атак

У наведеній нижче таблиці наведено підсумок вищезазначеного дослідження атак на мережу SDN. Також у ній ми можемо побачити рівні мережі SDN, на які впливають ті чи інші типи атак, та конкретні аспекти безпеки, які вони потенційно можуть скомпрометувати (табл. 1).

Таблиця 1 - Порівняння атак у мережах SDN

Тип атаки	Рівень мережі SDN, на який відбувається атака	Аспект безпеки, що вражений атакою		
		Доступність	Конфіденційність	Цілісність
<b>DDoS</b>	Передачі даних та управління	X		
<b>DoS</b>	Передачі даних та управління	X		
<b>Скомпрометований контролер</b>	Додатків, передачі даних та управління	X	X	X
<b>Шкідливі додатки</b>	Передачі даних та управління		X	X
<b>Man-in-the-middle</b>	Передачі даних, управління, зв'язок даних-управління		X	X
<b>Blackhole</b>	Передачі даних, управління, зв'язок даних-управління	X	X	
<b>Підслуховування</b>	Додатків, передачі даних та управління		X	

Кожна атака, досліджена в цьому розділі, була або залишається досить поширеною для традиційних мереж. Найцікавіше, що з появою нових платформ для атак, притаманних архітектурним змінам SDN, виникають варіації нападу, які є ексклюзивними для SDN. Незважаючи на те, що традиційні варіанти кожного з цих видів атак можна більш ефективно вирішити завдяки централізованому характеру SDN, саме модифіковані варіанти звичайних атак, що пристосовані до SDN, становлять найбільшу загрозу, і потребують найбільшої уваги при роботі з безпекою SDN. Наступний розділ має на меті ознайомитись із запропонованими рішеннями щодо вищезгаданих нападів та оцінити їх ефективність.

## **2.2 Способи захисту елементів мереж SDN**

У попередньому розділі визначено деякі основні загрози як для рівня управління, так і для рівня передачі даних у контексті мереж SDN. Відокремлення цих рівнів призводить до того, що мережі стає досить легко конфігурувати; проте це також створює можливість появи ряду загроз безпеки. У цьому розділі розглядаються деякі способи захисту, запропоновані для цих окремих загроз, та пропонуються деякі загальні рішення, які спрямовані на захист як рівня управління, так і рівня передачі даних.

### **2.2.1 Пом'якшення наслідків DDoS/DoS-атаки**

DoS-атаки на мережі SDN можуть виконуватися як на рівні управління, так і на рівні передачі даних. Нижче наведено ряд рішень; деякі спеціально націлені на захист рівня управління, деякі на рівень передачі даних, інші забезпечують захист обох цих рівнів.

Деякі автори [22] вводять рішення для DoS-атак на рівні управління на основі TCP - AVANT-GUARD. Це рішення складається з двох компонентів; механізм перенесення з'єднання, який використовується для відділення вдалих TCP-сеансів від невдалих і тригерів активації, які

дозволяють пристроям рівня передачі даних активувати правила потоку в певних, заздалегідь визначених, умовах. Передача з'єднання уповноважує «рукостискання» TCP, яке відбувається, коли вузли ініціюють TCP-з'єднання, і забезпечує успішне завершення сеансу, перш ніж дозволити будь-які записи потоку, що відносяться до цього сеансу, для пересилання контролеру. Це зменшує можливість з використанням пакетів TCP-SYN, оскільки «рукостискання» не буде завершено для цих сеансів. Механізм активації тригерів, представлений в [22], також зменшує обчислювальні навантаження, понесені контролером, дозволяючи пристроям активувати певні правила потоку у своїх таблицях за попередньо визначених умов. Це зменшує кількість переданих потоків запитів. Оціночні випробування доводять, що за наявності DDoS-атаки на основі TCP-SYN, час відгуку контролера на легітимні запити на потоки збільшується на незначну кількість мілісекунд разом із відсотком накладних витрат [22]. Хоча продуктивність AVANT-GUARD є досить бажаною, цей підхід, як правило, обмежений внаслідок того, що він спрямований на одну конкретну варіацію атаки DDoS (атаки на основі TCP-SYN). Можливо було б розгорнути це рішення разом з іншими механізмами запобігання DDoS, націлених на інші варіанти нападів, однак ця конфігурація може стати громіздкою та ресурсоємною. Замість цього рішення було б бажано захистити рівень управління від широкого кола типів атак DDoS.

Також було представлено [23] новий механізм запобігання атакам DoS на рівень управління - CPRecovery. Цей елемент захисту, що базується на реплікації, дозволяє передавати управління від одного контролера, якщо на нього в даний момент відбувається атака насичення, до вторинного контролера. Для цього комутаторі перевіряють на наявність належним чином працюючого контролера, відправивши зонд бездіяльності. Якщо цей зонд визначає неможливість роботи контролера, його «бездіяльність», зв'язок здійснюється з вторинним контролером, який бере на себе роль

невдалого первинного. Щоб забезпечити безперервність цього процесу, початковий первинний контролер надсилає повідомлення про стан оновлень до вторинних контролерів у мережі [23]. Цей багатошаровий підхід забезпечує стійкість, і, на відміну від AVANT-GUARD [16], він забезпечує рівень управління захистом від різноманітних атак. Однак у [23] зазначено, що після того, як вторинний контролер бере на себе первинну роль, він стає вразливим для атак DoS. Ще одним недоліком CPRecovery є його продуктивність. Через накладні витрати, пов'язані з створенням і підключенням контролерів відновлення, CPRecovery може бути досить повільним [23], час відгуку набагато вище, ніж у AVANT-GUARD.

Представлений в [24] спосіб захисту - це FlowRanger, пропозиція, що описується як алгоритм визначення пріоритету запиту для атак DoS на рівні управління. На фундаментальному рівні FlowRanger реалізує систему планування на основі пріоритетів, основною метрикою якої є цінність довіри, що зберігається кожним вузлом у мережі. Контролери, що реалізують FlowRanger, можуть оцінювати значення довіри; від кожного вузла вони отримують запити і буферизують їх у окремих чергах [24]. Інші пропозиції ([22], [23]) впроваджують обмежувач для кількості запитів, які можуть бути відправлені контролеру, однак це призводить до виключення деяких законних запитів. Це явно є проблемою; у виробничих середовищах неприпустимо, щоб законні поточні запити були скинуті. FlowRanger натомість використовує пріоритетний механізм чергування. Запити, що підозрюються в атаці, як і раніше, будуть обслуговуватися, хоча і мають менший пріоритет, ніж інші. Це добре працює, наприклад, якщо легітимний комутатор має проблеми і повинен повторно передати поточні запити контролеру. Вони будуть буферизовані в останню чергу, але в кінці кінців запит буде поданий. Це означає, що після того, як комутатор зарекомендував себе і поповнив свій кеш потоку коректними правилами,



значення довіри до нього збільшуватиметься, і його запити повертаються до більш високих пріоритетних черг. FlowRanger відрізняється від попередньої реалізації тим, що його пріоритетні черги реалізуються на контролері, а не розподілено. Це забезпечує додатковий рівень захисту (до тих пір, поки контролер не буде скомпрометований). Результати моделювання також показують, що централізований характер цього механізму робить його кращим, ніж раніше запропоновані рішення [24].

Був запропонований метод [25], що використовує аналіз поведінки користувачів. Незважаючи на те, що це звучить ефективно, прийняття сильних припущень стосовно кількості запитів на потоки, створених користувачами, залишає бажати кращого. Існуючі IP-адреси запитів відслідковуються контролером, і якщо профіль пакетів, отриманих з цього IP, потрапляє в певну категорію, IP позначається як шкідливий. Контролер згодом скидає запити з цього IP. Цей метод страждає від проблем аналогічних до AVANT-GUARD [22], в якому справжні запити від легітимних користувачів можуть бути проігноровані.

Методи [22], [23], [24] та [25] зосереджуються на DoS/DDoS-атаках на рівень управління. Хоча, напевно, більш важливим є те, що захищений рівень управління, атаки на рівень передачі даних існують і також вимагають достатнього захисту.

У [26] пропонується ліквідація наслідків DoS-атак, орієнтована на захист інформації рівня передачі даних, відома як віртуальне джерело адресної межі механізму перевірки достовірності (VAVE). VAVE намагається виявити наявність вузла, який використовує методи IP-спуфінгу для маскуванню своєї реальної ідентичності - такий тип поведінки часто призводить до атак DoS. Порівнюючи вхідні пакети із записами у таблиці потоків, інтерфейс VAVE в мережі визначає, чи пакет визначеного типу, якщо ні, перевіряється важливість пакету за списком заздалегідь визначених правил [26]. Це може призвести до виникнення проблем, якщо

попередньо встановлений список правил не є ретельним, деякі легітимні пакети можуть бути виключені, що спричиняє проблеми з доступністю для справжніх клієнтів. Оскільки це пом'якшення захищає лише від атак на рівні передачі даних, це призведе до того, що контролер стає сприятливим до атак DoS. Щоб забезпечити ретельний захист від усіх типів DoS, паралельне впровадження VAVE разом з AVANT-GUARD або CPRecovery було б більш доречним. Проте ця паралельна реалізація призводить до недоліків високої обчислювальної вартості та можливості конфліктуючих конфігурацій.

Однією з найважливіших проблем, пов'язаних із запропонованими рішеннями для атак DoS/DDoS, є те, що вони чекають від зловмисника атаки, а потім намагаються впоратися з наслідками. У випадку [22], [23] та [27] мережа може бути тимчасово недоступною до того, як буде зроблена спроба пом'якшення. FlowRanger [24] справляється з цим до певної міри, і в цілому, як видається, є найсильнішим рішенням з вищезазначеного вибору. Одне рішення, яке робить акцент на ранньому виявленні DDoS, представлено в [28]. У схемі роботи цього рішення використовується міра зміни ентропії в полі призначення IP-адреси атаки. Метод стверджує, що зупиняє атаку протягом перших 500 переданих пакетів атаки. Хоча це рано, цілком імовірно, що протягом цього періоду часу буде здійснено достатню шкоду. Здається очевидним, що захист від нападів на рівні управління та рівні передачі даних є двома цілком відокремленими речами. Для мережі будуть дуже корисними рішення, які одночасно дозволять захистити обидва ці рівні, доки досягається достатній рівень продуктивності.

### **2.2.2 Захист від скомпрометованих контролерів**

Неавторизований доступ до контролера в середовищі SDN був визначений як одна з найбільш сильних загроз. Запропоновано різні підходи щодо захисту SDN від несанкціонованого доступу на рівні

управління. Деякі автори [15] пропонують архітектуру для декількох контролерів у поєднанні з механізмом «протоколу візантійської угоди», в якому ряд контролерів диктує роботу комутаторів середовища. Кожен комутатор в мережі підключений до набору контролерів, і коли одному з контролерів не вдається, через успішно проведену на нього атаку, інший контролер бере керування на себе і з'єднання з першим, несправним, контролером буде перервано. Ця конкретна пропозиція потребує високого рівня використання ресурсів; дана розробка вводиться для зменшення споживання ресурсів шляхом визначення оптимальної кількості контролерів, підключених до кожного комутатора, на основі вимог латентності та толерантності до комутатора. Результати моделювання, представлені в [15], вказують на те, що метод досягає розумних рівнів продуктивності, коли розмір рівня даних невеликий; однак, оскільки кількість комутаторів збільшується, кількість необхідних контролерів також збільшується, що призводить до зниження продуктивності у середовищах більшого масштабу. Можна сказати, що рішення, наведене в [15] страждає від тих самих недоліків, що і CRRestore, в тому, що хоча заміна контролеру надає контроль над мережею, новий контролер піддається тим же атакам, що і попередній контролер. Здається, що найбільш ефективні рішення забезпечують захист від нападу, а не лише забезпечують відмовостійкість та стійкість при нападі.

На відміну від вищезгаданої пропозиції, автори [29] запроваджують механізм, спрямований на забезпечення захисту мереж SDN, змушуючи їх працювати розподіленим способом. Більш рання робота [30], представлена тими самими авторами, визначає роботу цієї гібридної розподіленої системи SDN, в якій контролер продовжує централізувати правила потоку, але встановлює алгоритми, щоб комутатори могли поширювати потоки на інші пристрої в мережі. У цій роботі запропоновано ряд механізмів, які працюють спільно для забезпечення безпеки в цьому розподіленому

середовищі. Впроваджено використання системи менеджера довіри. Це активно підтримуваний список, що містить ідентифікатори для всіх пристроїв, які в даний час працюють у мережі. Потім він поєднується з зашифрованими передачами всіх записів потоку, що відбуваються між пристроєм рівня передачі даних і контролером та між декількох пристроїв рівня передачі даних. Встановлений механізм автентифікації, який дозволяє кожному пристрою в мережі даних перевіряти автентичність та легітимність вузла, від якого потік походить. Цей підхід суттєво контрастує з методом наведеним у роботі [15], який спирається просто на механізм відмовостійкості. Як вже було попередньо зазначено, [29] забезпечує використання TLS у всіх його передавальних пристроях, що забезпечують ці зв'язки. Однією з проблем, пов'язаних із розподіленим характером [29], буде складний процес конфігурування та відлагодження проблем з ним. Оскільки загальне рішення складається з безлічі пов'язаних механізмів, помилка в одному місці знизить захист всієї системи або принаймні залишить її частково діючою та відкритою для атак. Процес точного визначення точки відмови може стати складним процесом. Оскільки сутність SDN призводить до того, що мережі діють за централізованим функціонуванням, здавалося б логічним розвивати централізовані механізми безпеки для її доповнення.

Жоден результат моделювання не представлений в [29], що призвело до припущення, що спосіб може не підлягати адекватному тестуванню на сьогоднішній день - це зробить [15] більш надійним, однак нижчі обчислювальні витрати, що потрапляють у [29], зробили б цей метод більш бажаним для використання у налаштуваннях мережі. Представлена розподілена архітектура також виглядає більш ефективною і її загальний характер дозволяє використовувати будь-який метод шифрування як частину його реалізації, що робить його більш гнучким варіантом. Одна з подібностей між цими двома рішеннями полягає в тому, що вони обидва

забезпечують захист лише для зв'язків, що існують на рівні управління, і при зв'язку рівня управління з рівнем передачі даних. Метод захисту, що представлений в [31] - це метод, який забезпечує захист як для рівня управління, так і для рівня передачі даних. AuthFlow забороняє доступ до неавторизованих вузлів у мережі та забезпечує належну автентифікацію за допомогою сервера RADIUS. Протокол розширеної перевірки автентичності (EAP) використовується для інкапсуляції будь-яких повідомлень, які хостів надсилає на сервер RADIUS, вимагаючи автентифікації. Перехоплюючий автентифікатор передає ці повідомлення контролеру, який потім додає хост, що відправляє ці пакети, до свого списку авторизованих хостів. Особливість цієї пропозиції полягає в тому, що пакети, що містять оновлення вхідного потоку для інших пристроїв, не передаються, доки обидва пристрої не будуть автентифіковані. Незважаючи на те, що цей підхід забезпечує надійний захист та авторизацію для легітимних хостів на рівні передачі даних, його слабкість полягає в його власних основних технологіях. EAP, як правило, вважається слабким у сфері автентифікації [32], і, хоча це дозволяє тим, хто налаштовує, вибирати власні методи автентифікації та шифрування, будь-який хакер, який володіє знаннями в традиційних атаках, має можливість примусового входження в мережу через несанкціонований шлях до даних.

Порівняно з механізмом «протоколу візантійської угоди» толерантності до помилок, описаним у [15] та розподіленою схемою безпеки в [29], AuthFlow, як видається, пропонує більш ретельне рішення, що забезпечує автентифікацію та авторизацію як на рівні управління, так і на рівні передачі даних мережі SDN. Результати тестування, представлені в документі AuthFlow, свідчать про те, що метод успішно запобігає несанкціонованому доступу мережевих хостів на рівні передачі даних і контролерів, і робить це ефективно - при низьких обчислювальних та комунікаційних витратах через низьку кількість контролерів [31]. Крім

того, механізм AuthFlow більш масштабований, ніж інші дві пропозиції, оскільки він не вимагає додавання більшої кількості контролерів, при збільшенні розміру рівня передачі даних; нові пристрої можуть просто автентифікуватись з сервером RADIUS і бути доданими до списку надійних пристроїв, що зберігаються на контролері. Однією з подальших пропозицій, які, як видається, пропонують надійну автентичність, обґрунтованість та цілісність з точки зору правил потоку, встановлених на комутаторах, є PERM-GUARD [33]. Забезпечуючи захист на всіх шарах моделі SDN (Control, Data, Application та зв'язок між ними), PERM-GUARD використовує схему, яка керує дозволами на виробництво правил потоку контролерів та додатків у інфраструктурі SDN. Якщо контролер або додаток хоче вивести протоколи потоку за маршрути даних в мережі, вони повинні автентифікувати себе централізованим органом влади за допомогою ідентифікуючого підпису. Кожен легітимний контролер або додаток буде мати один з цих підписів, а відповідні дозволи на виробництво потоку будуть встановлені для автентифікації [33]. Якщо один з цих підписів не може бути представлений, тоді відповідний контролер або додаток буде вважатись нелегітимним. Хоча це може не зупинити зловмисників від намагання взломати контролер чи від створення власних контролерів-зловмисників і спроб підключати їх до мережі, він попереджує їх здатність робити шкідливі зміни до структури мережі.

Здається, що PERM-GUARD є найкращим з усіх рішень, наведених у вищезазначеному розділі, - він забезпечує захист не лише контролерів, а й додатків. Це величезна перевага для будь-якого рішення, коли система здатна забезпечити перехресний захист шарів від загальної атаки. Загалом здається, що завдання зупинки вже скомпрометованого контролера не є важким – найскладнішою частиною є саме їх виявлення. Необхідно розробити попереджувальні механізми, які запобігають можливості

зловмисників та нелегітимних контролерів постійно отримувати доступ до мережі, щоб забезпечити відсутність шансів на скомпрометування.

### **2.2.3 Захист від шкідливих додатків**

Шкідливі додатки - це ще одна небезпечна загроза в контексті SDN. Скомпрометований додаток або додаток, що був запрограмований зі зловмисним наміром, може дозволити зловмисникові отримати контроль над всією мережею [34]. Подібним чином, додаток зі зловмисним кодом може ввести вразливості, які зловмисники можуть використати для отримання несанкціонованого доступу до мережі.

Автори в [27] пропонують "FortNOX" - механізм, який контролює вставку потоків-правил з додатків безпеки до пристроїв у мережі. Нові правила потоку спочатку перевіряються на всі інші існуючі правила потоку на приймаючому пристрої, і при наявності конфлікту новий потік відкидається і не додається до локального кеша. Цей механізм дозволяє адміністраторам виконувати набір жорстко кодованих правил, які перевищують будь-які динамічно створені правила. Це гарантує, що помилкові потоки, додані в мережу шкідливими додатками, які спрямовуватимуть трафік на зловмисні вузли, не будуть дозволені [27]. Однією з особливостей цього підходу є той факт, що законні додатки безпеки, які покладаються на динамічну модифікацію потоків всередині мережі, не працюватимуть належним чином за наявності адміністративно створених жорстко закодованих правил. Автори статті [27] не вивчають цей сценарій, тому майбутня робота може бути пов'язана з цією областю та запровадження механізму дозволів, який дозволить певним динамічним потокам мати перевагу.

Альтернативою попередньому рішення, автори [34], представляють Rosemary - невелику мережеву операційну систему (NOS), яка запускається в декількох випадках на вершині рівня управління SDN. Кожна програма, що працює в середовищі, виконується в окремому

екземплярі Rosemary, ефективно ізолюючи кожен додаток. Це дозволяє точно контролювати кожне застосування, що використовується в мережі, з точки зору ресурсів, які він використовує, та пакетів, які він передає або отримує. Окрім забезпечення платформи для моніторингу шкідливих дій окремих мережевих додатків, Rosemary забезпечує стійкість у тому випадку, коли один додаток не працює або працює з перебоями, інші продовжуватимуть працювати в їх ізольованому екземплярі NOS [34]. Опис Rosemary як надійного та високопродуктивного ноу-хау узгоджується з результатами оціночного тесту. Випробування показують, що успішні показники атаки дуже низькі для Rosemary, високий рівень продуктивності досягнуто з точки зору пропускну здатності та затримки роботи. Це, однак, призводить до недоліків високих обчислювальних накладних витрат, особливо у порівнянні з FortNOX, який має відносно низький рівень накладних витрат [27].

Інша пропозиція - LegoSDN [35] концептуально схожа на Rosemary, оскільки вона забезпечує шар ізоляції між контролером і рівнем додатків середовища SDN. Основна відмінність полягає в тому, що всі додатки згруповані разом на одному рівні, тоді як Rosemary реалізує окремий контейнер для кожної окремої програми. Хоча LegoSDN не розроблений спеціально для боротьби з атаками, він виділяє додатки, які визначаються як непридатні - непридатність додатку є однією з ключових ознак початку атаки на рівні додатків [35]. В цьому сенсі, LegoSDN забезпечує попереджувальний механізм - ізолює потенційно шкідливі додатки від мережі, перш ніж вони зможуть завдати шкоди. У порівнянні з FortNOX і Rosemary, LegoSDN не забезпечує достатньо ретельного пом'якшення наслідків атак. Паралельне впровадження LegoSDN та FortNOX може бути більш доцільним, забезпечуючи, відповідно, захист першої та другої ліній відповідно.



OperationCheckpoint вводиться в [16] як система управління додатками SDN, беручи до уваги інформацію, яку читає додаток SDN з основної мережі, а також правила та політики, які він створює для шляхів передачі даних в мережі. OperationCheckpoint використовує ретельний набір дозволених користувачем дозволів, що охоплюють всі завдання, пов'язані з OpenFlow. Кожен додаток, який потім повинен бути використаний, буде співставлений з певною підмножиною цих дозволів. Додатки не зможуть виконати будь-які дії, які виходять за межі встановлених дозволів (до тих пір, поки зміни не дозволено адміністраторами чи операторами) [16]. Це фактично припиняє роботу додатків, створених з метою нанесення шкоди зловмисниками, по захопленню конфіденційних даних або виконанню шкідливих команд на рівні передачі даних, оскільки ці дії спочатку повинні бути оголошені. Застосування цієї декларації необхідних функцій дозволяє операторам мережі створювати спеціальні набори дозволів і гарантувати, що вони точно знають, як працює кожен додаток.

OperationCheckpoint, здається, є найбільш ретельним методом захисту інфраструктури SDN від шкідливих додатків. У порівнянні з іншими рішеннями, він, здається, працює ефективніше за рахунок спрощеної природи своєї системи дозволів. Це особливо помітно у порівнянні з рішеннями, такими як Rosemary, яка, з точки зору продуктивності, має досить низькі показники, через те, що кожен додаток працює в своєму власному середовищі. Це призводить до високих рівнів накладних витрат. Однією з особливостей SDN є той факт, що існує безліч загальних інтерфейсів API, які можуть використовуватися на північному інтерфейсі між додатками та контролером. Це вигідно в тому сенсі, що він усуває можливість поширення вразливостей у крос-платформі, що виникає через погану конфігурацію та кодування проміжного програмного забезпечення.

## 2.2.4 Захист рівня управління та зв'язку з рівнем передачі даних

Зв'язок між рівнем управління та рівнем передачі даних переміщує передачі поточних записів вниз до рівня передачі даних для пристроїв, щоб додати їх до локального кеша [34]. Специфікація для OpenFlow, найбільш широко використовуваного та підтримуваного протоколу для операцій SDN, вказує на необов'язкову реалізацію транспортного рівня безпеки (TLS) для цього зв'язку, однак, оскільки це необов'язково, багато з доступних API-контролерів не застосовують або не підтримують цю опцію. Для того, щоб це з'єднання було повністю захищено, специфікації контролерів повинні забезпечувати використання TLS - забезпечуючи захищену зашифровану передачу та автентифікацію об'єктів на кожному кінці зв'язку.

На додаток до цього, системи стилів IDS можуть бути реалізовані, щоб перешкоджати роботі атак типу man-in-the-middle і blackhole, що відбуваються на цьому конкретному зв'язку. Автори [36] подають методологію, в якій система Bro IDS інтегрована з контролером Python на основі Ryu. При отриманні пакета IDS використовує глибокі методи перевірки пакетів, щоб перевірити пакет та визначити його джерело, призначення, навантаження та інші речі. Діючи як IDS на основі підпису, пошкодження трафіку та шкідливі дії попередньо налаштовуються на контролері, а потім невірні пакети перевіряються на ці підписи, щоб виявити наявність шкідливих пакетів. Оціночні дослідження, проведені в [36], вказують на те, що час відгуку цього механізму є досить повільним. Це пов'язано з додатковим навантаженням контролера від інтеграції системи IDS.

Через те, що більшість атак, що відбуваються в рамках даного зв'язку, є варіаціями на традиційні атаки man-in-the-middle і blackhole, можливо, можна буде захистити зв'язок між рівнями управління та

передачі даних за допомогою традиційних засобів. Це не означає, що це буде ретельним рішенням. Як зазначено вище, використання системи IDS для ідентифікації потенційно шкідливих вузлів, що пересилають повідомлення за цим зв'язком, гарантує, що фальсифіковані повідомлення не будуть пересилатися між шляхами даних та контролерами, конфіденційні дані не витягуються, а доступність не буде порушена.

### **2.2.5 Захист від атак підслуховування**

Захист від підслуховувань в будь-якому середовищі може бути особливо важким завданням. Це пов'язано з пасивним характером атаки; хакери можуть просто встановити себе як вузол у мережі та активувати режим прослуховування. Це дає їм можливість переглядати потік пакетів, що їх пересуває. Оскільки підслуховуючі вузли, як видається, мають схожі поведінкові характеристики для законних клієнтів, їх важко виявити. У середовищі SDN це ускладнюється з введенням нових платформ атак, що виникають з того, що дані мережі мають централізований характер. Прикладом цього може служити зв'язок рівня управління та рівня передачі даних або зв'язок рівня управління з рівнем додатків.

Пасивний підхід до захисту від підслуховування на рівні передачі даних та на зв'язку між рівнем управління та рівнем передачі даних, представлені в [32] під назвою Random Route Mutation. Її методика захисту полягає у випадковому зміні курсу потоку пакетів, таким чином, що IP-адреса призначення залишається незмінною. Зміна цього потоку спрямована на приховування слідів пакетів, зібраних клієнтами. Такий спосіб був реалізований у традиційних мережевих налаштуваннях із помірним успіхом. Автори [32] стверджують, що результати моделювання показують, що механізм RRM є продуктивним та ефективним у зменшенні кількості успішно підслуханих пакетів. Цей підхід також був застосований в контролері NOX, що передбачає, що він також може застосовуватися до середовищ SDN. Хоча ця технологія може показати значне зменшення

кількості підслуханих пакетів, зловмисники, дізнавшись про алгоритми, які використовуються для рандомізації потоку пакетів, вони можуть змінити вплив, застосовуючи зворотний алгоритм для захоплення пакетів. Незважаючи на те, що цей пасивний підхід забезпечує деяке пом'якшення, це очевидно, що його ефект обмежений, і було б необхідним більш глибоке рішення, якщо SDN планується до використання у виробничих установках.

У [21] вводиться Combat-Sniff – система боротьби зі сніффінгом. Combat-Sniff реалізує як механізм активного виявлення, який активно сканує підслуховуючі вузли, так і активний метод захисту, який покликаний запобігти тому, щоб зловмисники не мали можливості аналізувати трафік. Зловмисник може примусити комутатор зберігати нелегітимний запис потоку в кеші, який пересилає всі пакети собі. Combat-Sniff бореться з цим, приймаючи випадкові зразки записів потоку, встановлених у кожному з таблиць комутаторів, і перевіряючи їх цілісність [21]. Якщо потік слід визначити як незаконний, відповідний порт на комутаторі вимикається, і через нього більше не передаються пакети. Combat-Sniff також має на меті зберігати конфіденційність інформації про пакети, що проходять через комутатор, забезпечуючи це тим, що комутатор є частково сліпим. Це означає, що комутатор досить «розумний», щоб знати, як передати пакет, але не знає вміст пакетів. Хоча здається, що це рішення, є технічно доцільним, і оціночні випробування доводять, що воно має досить прийнятний рівень ефективності, також було б важливо забезпечити, щоб вага випадкового відбору проб на вході була достатня для забезпечення ідентифікації незаконних потоків.

Здається, що підслуховування, як правило, є дуже складною атакою в плані захисту від неї, з нечисленними рішеннями, існуючими виключно для захисту від нього. Системи виявлення вторгнень та системи захисту від вторгнень потенційно можуть бути налаштовані на виявлення підслуховуючих вузлів, однак пасивний характер атаки ускладнює її

виявлення. Здається, що два існуючих рішення ([32], [21]) більше спрямовані на зменшення шкоди будь-яким атакам підслуховування. Більш ефективне рішення дозволить запобігти будь-якому прослуховуванню. Це може бути досягнуто шляхом використання деякого сильного методу шифрування, а надання лише перевіреним шляхам даних та контролерам правильного ключа для незашифрованого пакету.

### **2.2.6 Порівняння методів захисту у мережах SDN**

Кожен із розглянутих вище методів захисту від атак оцінювався з точки зору його сильних і слабких сторін. Рішення були проаналізовані, щоб визначити прогалини, які ще не заповнені з точки зору безпеки SDN. Для того, щоб підсумувати проаналізовані вище методи, була складена таблиця, яка показує кожне з рішень, тип нападу, від якого вони намагаються захистити, логічний рівень SDN, на якому вони працюють, та аспекти безпеки, які вони прагнуть підтримувати (табл. 2).

Таблиця 2 - Порівняння методів захисту у мережах SDN

Тип атаки	Аспекти безпеки, що захищається	Запропоноване рішення	Рівень SDN				
			Передача даних	Зв'язування-пер.даних	Управління	Зв'язування-додатків	Додатків
1	2	3	4	5	6	7	8
DoS/DDoS	Доступність	AVANT-GUARD			x		
		CPRecovery			x		
		FlowRanger			x		
		VAVE	x				
		Entropy-based detection	x		x		
Скомпрометований контролер	Доступність Конфіденційність Цілісність	Byzantine Fault Tolerance			x		
		Trust Management System	x		x		
		AuthFlow		x	x	x	x
		PERM-GUARD	x		x		x
Шкідливі додатки	Конфіденційність Цілісність	FortNOX		x	x	x	x
		Rosemary			x		x
		LegoSDN			x	x	x
		OperationCheckpoint			x	x	x
Атака рівня управління та зв'язку з рівнем передачі даних (MITM, Blackhole)	Доступність Конфіденційність Цілісність	BroIDS	x	x	x	x	x
Підслухування	Конфіденційність	Random Route Mutation	x	x			
		Combat-Sniff	x	x			

## Висновки до розділу 2

Розглянуто можливі атаки на всі рівні програмно-конфігуровної мережі та проаналізовані системи захисту. Проаналізована структура даних систем, їх переваги та недоліки, проведена порівняльна характеристика способів захисту.

Показано, що загальною проблемою запропонованих рішень захисту є те, що вони спрямовані на захист лише від одного конкретного типу атаки і не забезпечують загальний захист у налаштуваннях мережі. Щоб забезпечити захищену програмно-конфігуровну мережу, варто було б розглянути паралельну реалізацію комбінацій запропонованих схем. Це може призвести до більшої кількості обчислювальних витрат і створення конфліктуючих конфігурацій. Ці проблеми призводять до можливої появи додаткових вразливостей та можливості отримання помилкових спрацьовувань при скануванні мережі для атак.

Однак, зрозуміло, що з додаванням до контролера додаткових механізмів захисту, продуктивність мережі з точки зору її пропускної спроможності може бути негативно змінена. Щоб підвищити безпеку в мережах SDN, було б важливо знайти баланс між безпекою та продуктивністю, зробивши ретельний, загально мережний режим безпеки.

Тому необхідно розробити комплексну систему, яка б одночасно захищала від декількох типів атак.

### **3. РОЗРОБЛЕНА СИСТЕМА ЗАХИСТУ**

#### **3.1 Опис елементів системи захисту**

На основі проведеного дослідження існуючих загроз та можливих методів захисту можемо виділити основні загрози та основні рівні програмно-конфігуровної мережі на які відбуваються ці атаки. Так, загрози, від яких буде захищати розроблена система захисту це:

- підслуховування (як атака, яку досить важко виявити);
- спроба підключення нелегітимного пристрою;
- Man-in-the-Middle.

Рівні програмно-конфігуровної мережі, котрі будуть захищені це, відповідно, ті рівні на які і відбуваються дані атаки, а саме:

- рівень управління;
- рівень передачі даних;
- зв'язок між рівнем управління та передачі даних.

Для захисту наведених рівнів, розроблена система використовує наступні модулі:

- автентифікація нового пристрою рівня передачі даних;
- шифрування трафіку між пристроями та контролером;
- генерування секретного ключа для автентифікації.

Від спроб приєднання о мережі нелегітимного пристрою, система захищена генеруванням секретного ключа на основі протоколу Діффі-Хеллмана. Цей протокол робить практично неможливим для зловмисника виявлення ключа.

Для захисту від прослуховування використовується шифрування трафіку між пристроями та контролером, на базі AES (Advanced Encryption Standard), також відомого як Rijndael – симетричного алгоритму блочного шифрування, що прийнято як державний стандарт шифрування у Сполучених Штатах Америки.



Для того, щоб автентифікувати пристрій в мережі, необхідно пройти автентифікацію, розроблену на базі протоколу з використанням загального секретного ключа та HMAC ((hash-based message authentication code, хеш-код ідентифікації повідомлень). Аби мати можливість автентифікувати пристрій, обидві сторони (контролер і пристрій-клієнт) повинні мати у себе якийсь спільний секретний ключ. В розробленій системі даний ключ генерується завдяки використанню протоколу Діффі-Хеллмана.

Також, реалізований спосіб автентифікації дозволяє створити захист і від атак типу Man-in-the-Middle, оскільки пристрій зломисника буде виявлено ще на етапі спроби автентифікації.

Протокол Діффі-Хеллмана — це метод обміну криптографічними ключами. Один з перших практичних прикладів обміну ключами, що дозволяє двом учасникам, що не мають жодних попередніх даних один про одного, отримати спільний секретний ключ із використанням незахищеного каналу зв'язку. Цей ключ можна використати для шифрування наступних сеансів зв'язку, що використовують шифр з симетричним ключем.

Хоча протокол Діффі-Хеллмана є анонімним (без автентифікації) протоколом встановлення ключа, він забезпечує базу для різноманітних протоколів з автентифікацією, і використовується для забезпечення цілковитої прямої секретності в недовговічних режимах Transport Layer Security (відомих як EDH або DHE залежно від комплектації шифру).

Припустимо, що існують два користувача: Аліса та Боб. Обом користувачам відомі деякі два числа  $g$  і  $p$ , які не є секретними та можуть бути відомі також іншим зацікавленим персонам. Для створення секретного ключа, який буде не відомий більше нікому, обидва користувачі генерують великі випадкові числа: Аліса генерує число  $a$ , а Боб генерує число  $b$ . Після цього, Аліса вираховує залишок від ділення  $A$ :

$$A = g^{a*} \text{mod}(p)$$

Після чого, вона пересилає його Бобу, який в свою чергу вираховує залишок від ділення В:

$$B = g^{b*} \text{mod}(p)$$

Цей залишок він пересилає Алісі. Зловмисник може отримати обидва ці значення, проте він не має можливості втрутитися в процес передачі, тобто в нього не має можливості якось змінити дані значення.

Далі, Аліса, виходячи з ключа а, який вона має та отриманого по мережі ключа В вираховує наступне значення:

$$B^{a*} \text{mod}(p) = g^{ab*} \text{mod}(p)$$

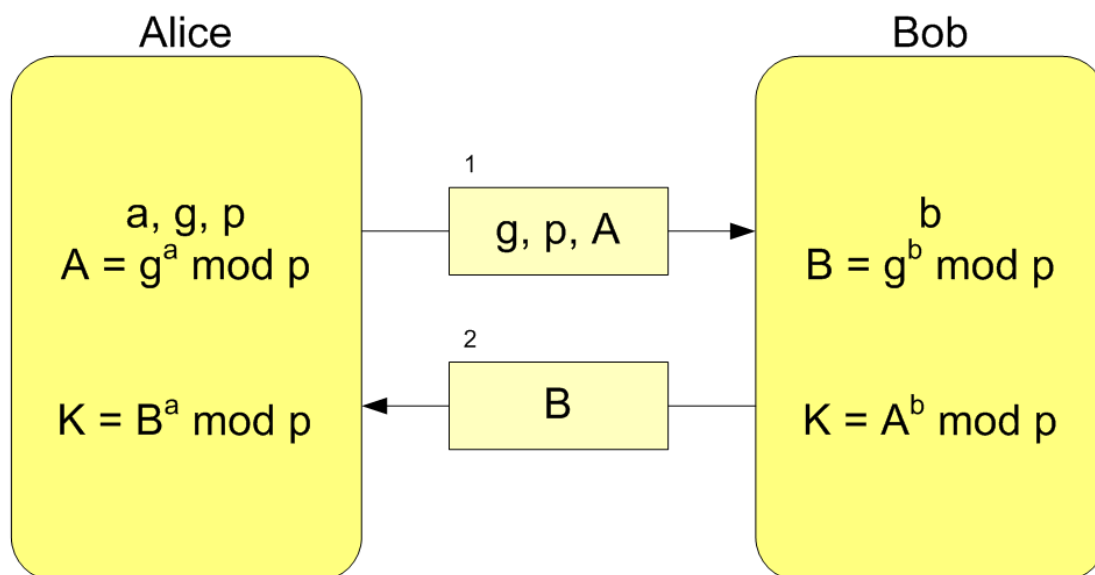
Боб, в свою чергу, на основі b, який він має та отриманого по мережі А вираховує значення:

$$A^{b*} \text{mod}(p) = g^{ab*} \text{mod}(p)$$

Отже, у Аліси та Боба було вираховано одне і те саме значення, яке вони і можуть використати в якості секретного ключа:

$$K = g^{ab*} \text{mod}(p)$$

Використання такого секретного ключа призводить до того, що зловмисник стикається з задачею, яку практично неможливо вирішити за розумний час, за рахунок вирахування  $g^a* \text{mod}(p)$  та  $g^b* \text{mod}(p)$ , якщо числа р, а та b обрані достатньо великими. Наглядна схема роботи алгоритму зображена на рис. 9.



$$K = A^b \text{ mod } p = (g^a \text{ mod } p)^b \text{ mod } p = g^{ab} \text{ mod } p = (g^b \text{ mod } p)^a \text{ mod } p = B^a \text{ mod } p$$

Рисунок 9 – схематичне зображення роботи протоколу Діффі-Хеллмана

НМАС це один з механізмів у криптографії, який використовують для перевірки цілісності інформації. Даний механізм дозволяє гарантувати те, що дані, які передаються в мережі, не були видозмінені нелегітимними персонами. В залежності від того, яка саме хеш-функція використовується розрізняють: НМАС-MD5, НМАС-SHA1, НМАС-RIPMD128, НМАС-RIPMD160 и т. п.

Переваги використання НМАС:

- можливість використання хеш-функцій, які вже присутні в програмному продукті;
- відсутність необхідності внесення змін до реалізації вже існуючих хеш-функцій (внесення змін може призвести до погіршення крипостійкості та продуктивності);
- можливість заміни хеш-функції у випадку появи більш безпечної або швидшої хеш-функції.

Механізм HMAC був описаний в стандартах організацій ANSI, IETF, ISO і NIST.

Алгоритм HMAC можна описати у вигляді формули:

$HMAC_K(text) = H((K \text{ xor } \text{opad}) \parallel (H((K \text{ xor } \text{ipad}) \parallel \text{text})))$ , де:

- xor – операція побітового виключного АБО;
- $\parallel$  – операція склеювання строк (послідовностей байт);
- H – хеш-функція;
- K – секретний ключ;
- opad – блок вигляду (0x5c 0x5c 0x5c ... 0x5c), де байт 0x5c повторюється кількість разів, рівну розміру блоку в байтах;
- ipad – блок вигляду (0x36 0x36 0x36 ... 0x36), де байт 0x36 повторюється кількість разів, рівну розміру блоку в байтах;
- text – повідомлення (дані), які будуть передаватися відправником і справжність якого буде перевірятися отримувачем.

Схема роботи алгоритму HMAC приведена на рис. 10 та 11.

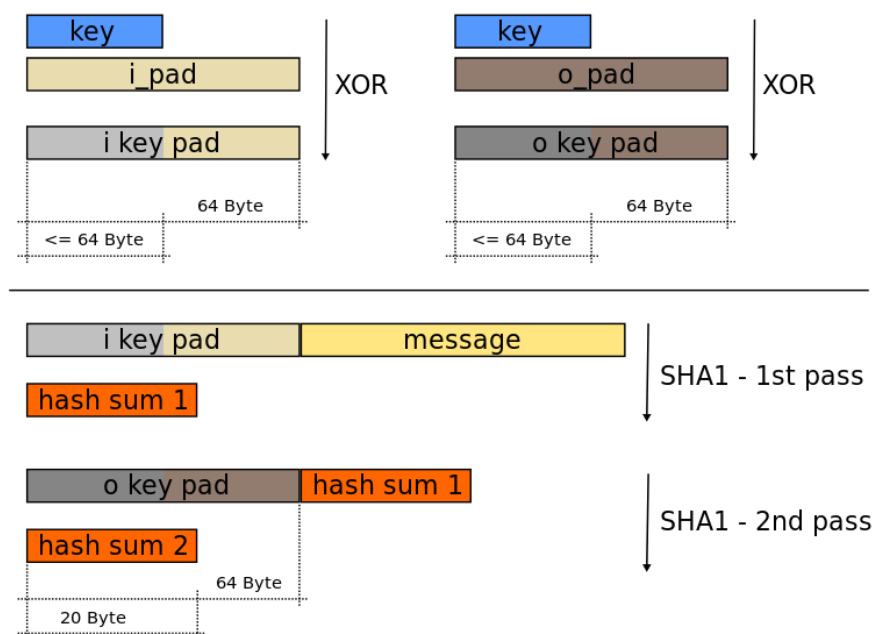


Рисунок 10 – Схематичне зображення реалізації HMAC-SHA1

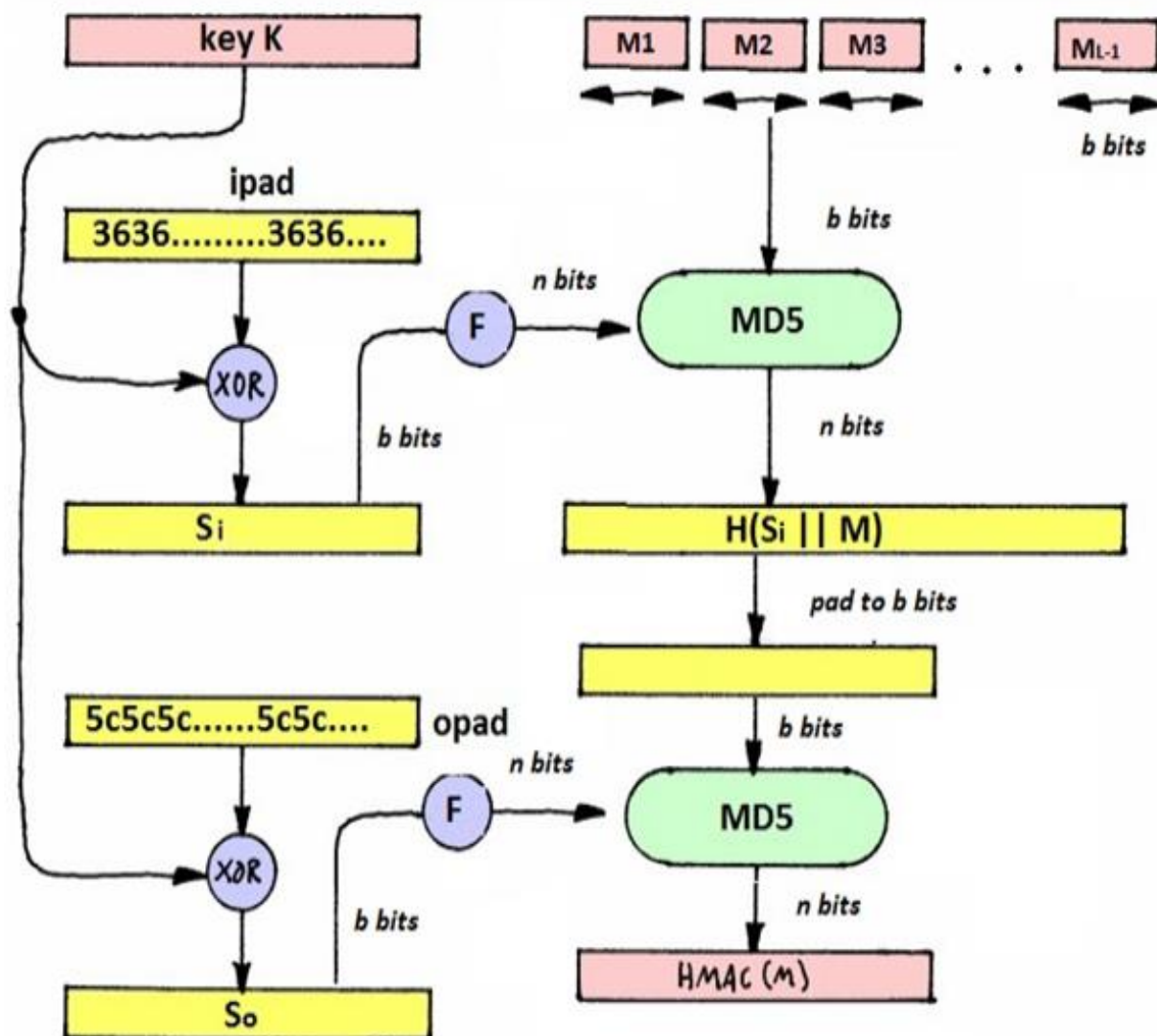


Рисунок 11 – Схематичне зображення реалізації HMAC-MD5

Етапи виконання алгоритму HMAC:

1. Отримати  $K_0$  шляхом зменшення або збільшення ключа  $K$  до розміру блока (до  $b$  байт).
  - 1.1 Якщо довжина ключа  $K$  дорівнює розміру блока, то копіюємо  $K$  в  $K_0$  без змін і переходимо до кроку 2.

1.2 Якщо довжина ключа  $K$  більше розміру блоку, то до ключа  $K$  застосовуємо хеш-функцію  $H$ , отримуємо рядок розміром в  $L$  байт, додаємо нулі до правої частини цього рядка для створення рядка розміром в  $b$  байт, копіюємо результат в  $K_0$  і переходимо до кроку 2.

1.3 Якщо довжина ключа  $K$  менше розміру блоку, то додаємо нулі до правої частини  $K$  для створення рядка розміром в  $b$  байт, копіюємо результат в  $K_0$  (наприклад, якщо  $\text{length}(K) = 20$  (в байтах) і  $b = 64$  (в байтах), то до правої частини  $K$  буде додано  $64 - 20 = 44$  нульових байта ( $0x00$ )) і переходимо до кроку 2.

2. Отримати блок  $S_i$  розміром в  $b$  байт за допомогою операції «побітове виключне АБО» (xor):

$$S_i = \text{xor}(K_0, \text{ipad}) = K_0 \text{ xor } \text{ipad}.$$

3. Отримати блок  $S_o$  розміром в  $b$  байт за допомогою операції «побітове виключне АБО»:

$$S_o = \text{xor}(K_0, \text{opad}) = K_0 \text{ xor } \text{opad}.$$

4. Розбити повідомлення (дані, набір байт)  $\text{text}$  на блоки розміром  $b$  байт.
5. Склеїти рядок  $S_i$  з кожним блоком повідомлення  $M$ .
6. До рядку, отриманого на попередньому кроці, застосувати хеш-функцію  $H$ .
7. Склеїти рядок  $S_o$  з рядком, отриманим від хеш-функції  $H$  на попередньому кроці.
8. До рядку, отриманому на попередньому кроці, застосувати хеш-функцію  $H$ .

Ключі розміром, меншим  $L$  байт, вважаються небезпечними. Рекомендується вибирати ключі випадковим чином і регулярно їх міняти. Ключі з розміром, більшим за  $L$  байт, істотно не збільшують стійкість

функції, але можуть використовуватися, якщо є сумніви у випадковості даних, що використовуються для створення ключа і одержуваних від генератора випадкових чисел.

Розмір ключа  $K$  повинен бути більше або дорівнювати  $L/2$  байт.

Більш ефективна реалізація алгоритму HMAC-MD5 представлена на рисунку 10. Реалізація відрізняється використанням функції  $F$ . Використання цієї реалізації доцільно, якщо більшість повідомлень, для яких обчислюється MAC, короткі. Функція  $F$  - функція стиснення для хеш-функції  $H$ . В якості аргументів  $F$  приймає змінну  $n$  і блок довжиною в  $b$  байт.  $F$  виробляє розбиття блоку на ланцюжок ланок з довжиною кожної ланки в  $n$  байт. Функція  $F$  викликається для кожного нового ключа один раз.

Отриманий код автентичності дозволяє переконатися в тому, що дані не змінювалися яким би то не було способом з тих пір, як вони були створені, передані або збережені довіреним джерелом. Для такого роду перевірки необхідно, щоб, наприклад, два пристрої, що довіряють один одному заздалегідь домовилися про використання секретного ключа, який відомий тільки їм. Тим самим гарантується автентичність джерела та шляхів сполучення. Недолік такого підходу очевидний - необхідна наявність двох пристроїв, що довіряють один одному.

Безпека будь-якої функції MAC на основі вбудованих хеш-функцій залежить від криптостійкості базової хеш-функції. Привабливість HMAC - в тому, що його творці змогли довести точне співвідношення між стійкістю вбудованих хеш-функцій і стійкістю HMAC.

Основою на даних протоколах та алгоритмах розроблено алгоритм автентифікації. Вона розпочинається після створення секретних ключів. Для пристрою рівня передачі даних даний алгоритм виглядає наступним чином:

- відбувається перевірка існування загального для пристрою та контролера секретного ключа;
- якщо ключа не існує, він встановлюється;
- у випадку, коли ключ вже існує, відбувається операція автентифікації пристрою з контролером;
- якщо операція автентифікації не пройшла успішно, повторюється попередній крок після деякого очікування;
- якщо операція автентифікації пройшла вдало, відбувається перевірка чи не застарів секретний ключ;
- у випадку, якщо ключ не застарів, процес завершується;
- у випадку, якщо ключ застарів, встановлюється новий ключ та відбувається розрив з'єднання, після якого пристрій знов розпочинає процес автентифікації з контролером.

Зі сторони контролера алгоритм автентифікації буде схожим, проте він буде містити також ініціацію генерації секретних ключів (рис.13):

- перш за все, перевіряється чи не існує правил, що забороняють автентифікувати пристрій;
- далі йде процес генерування секретного ключа, якщо його ще не було встановлено раніше;
- після цього можна переходити до автентифікування пристрою;
- у випадку, коли автентифікація пройшла вдало, починається передача даних, аж доки секретний ключ, що було встановлено, не застаріває;
- після завершення часу використання даного секретного ключа, ініціюється його зміна.



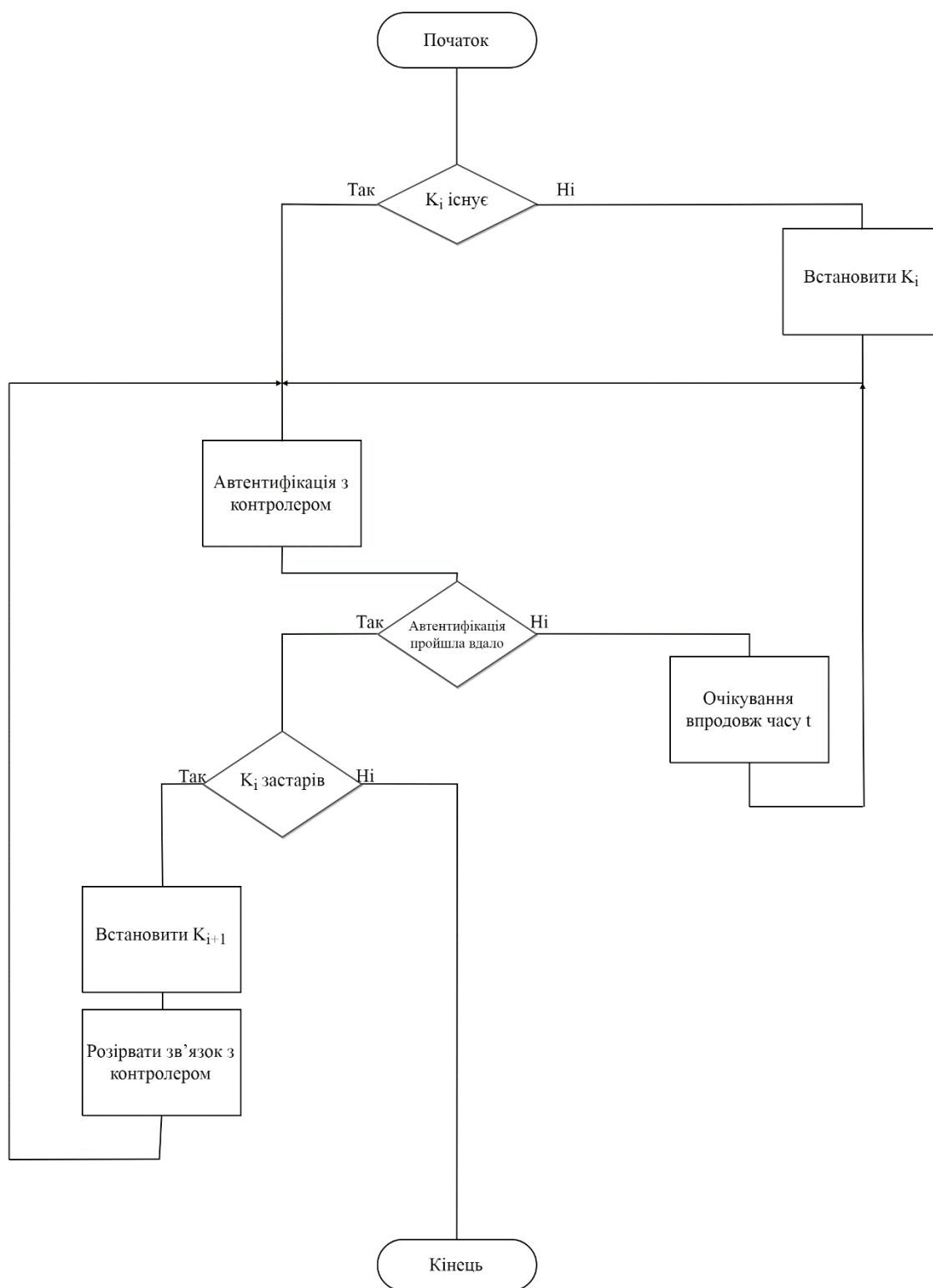


Рис.12 – схематичне зображення алгоритму автентифікації пристрою

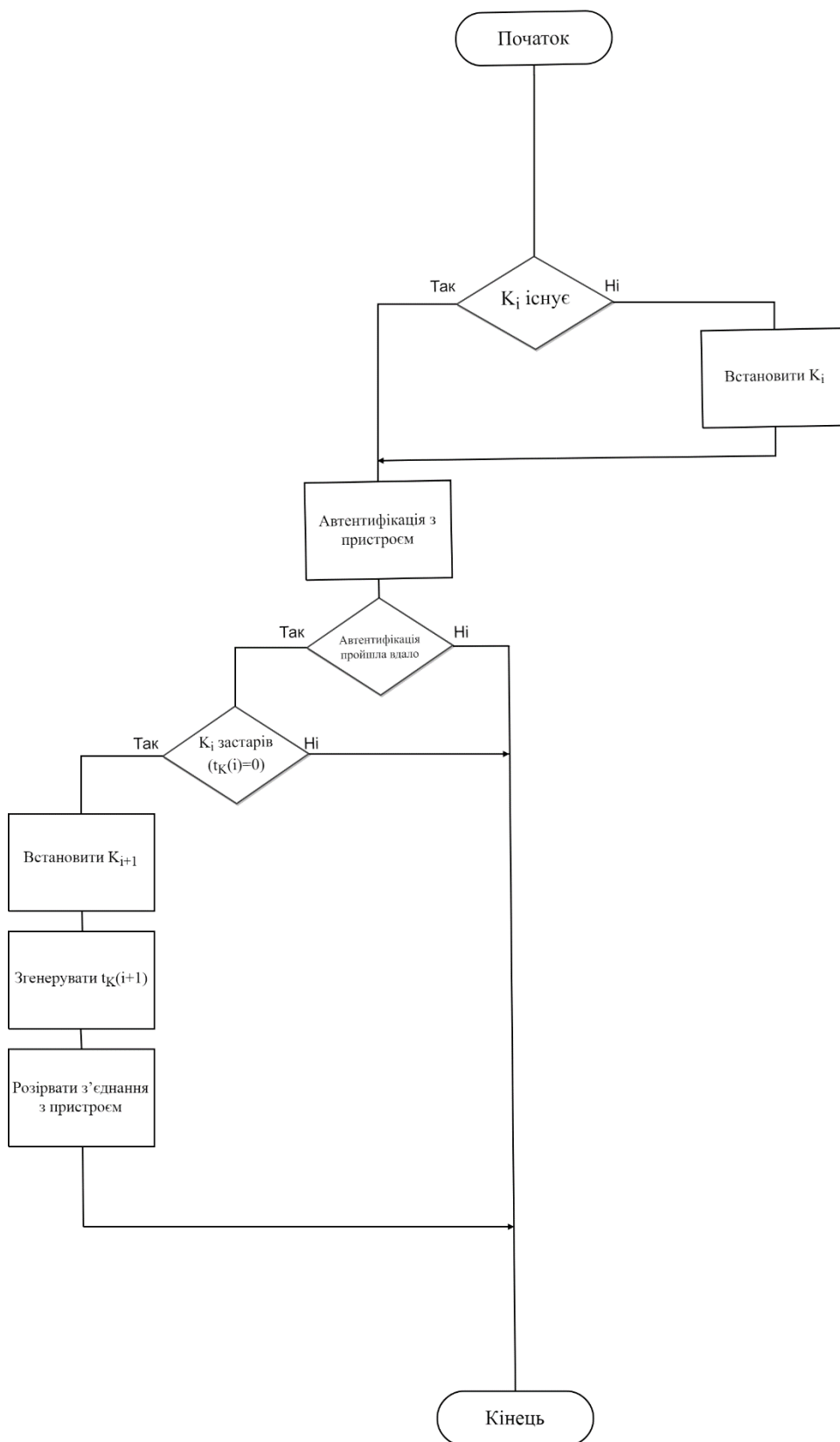


Рис.13 – схематичне зображення алгоритму автентифікації зі сторони контролера

Власне, алгоритм автентифікації буде виглядати наступним чином:

1. Спочатку, з пристрою, який повинен бути автентифікований, посилається випадково обране число на контролер, щоб той міг зашифрувати його загальним для них ключем. Це робиться для того, щоб контролер міг підтвердити, що він насправді є контролером цієї мережі, а не зловмисником, який ним прикидається.
2. Потім з контролера відправляється на пристрій випадкове число  $N_c$ , а також хеш HMAC, утворений шляхом хешування значень  $N_k$ ,  $N_c$ ,  $K$ ,  $C$  і  $K_s$  за допомогою HMAC-функції. Оскільки пристрій, що автентифікується, знає  $N_k$ ,  $N_c$ ,  $K$ ,  $C$  і  $K_s$ , то на ньому може бути сформований свій хеш HMAC, котрий порівнюється з тим, що прийшов від контролера. Якщо хеші будуть збігатися, то контролер є тим, за кого себе видає. У той же час зловмисник не міг би сформувати хеш HMAC ( $N_k$ ,  $N_c$ ,  $K$ ,  $C$ ,  $K_s$ ), оскільки він не знає секретного ключа  $K_s$ .
3. На третьому кроці прийшла черга мережевому пристрою підтвердити свою легітимність. Процес здійснюється за аналогією з другим кроком, тобто пристрій-клієнт так само шифрує випадкове число за допомогою HMAC та відправляє його на контролер, після чого той має у себе його розшифрувати та підтвердити, що отримав те саме число, що «загадав» пристрій.

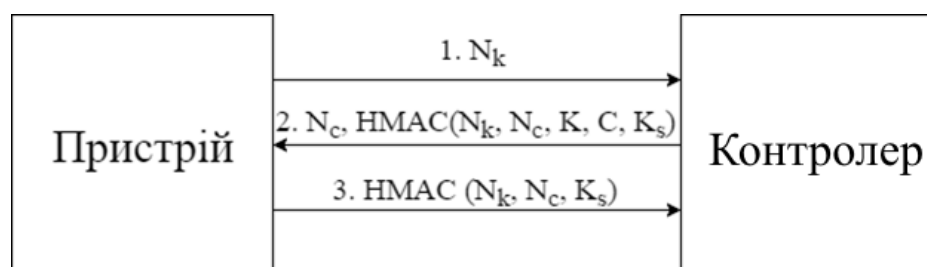


Рисунок 13 - Схематичне зображення взаємодії модулів при автентифікації

$N_k$ ,  $N_c$  – 128-бітні числа, обрані випадковим чином пристроєм і контролером окремо;

- $K$ ,  $C$  - ідентифікатори пристрою і контролера;
- HMAC – хеш-код ідентифікації повідомлення;
- $K_s$  - загальний секретний ключ.

Після того, як автентифікація була завершена, з'являється можливість передачі даних між пристроєм та контролером. Проте, у розробленій системі, дані також підлягають попередньому шифруванню. Як вже було зазначено, використовуваний алгоритм шифрування це американський стандарт під назвою Advanced Encryption Standard або Rijndael з використанням створеного секретного ключа  $K_s$ .

AES є симетричним алгоритмом блочного шифрування з розміром блоку у 128 біт та можливістю встановити ключ у 128/192/256 біт.

Розроблена система використовує ключ розміром 128 біт, оскільки він одночасно задовольняє вимозі неможливості підбору зловмисником паролю за розумний час, а з іншого боку не так сильно навантажить нашу систему, оскільки вона і так потребує від контролера та пристроїв додаткових обчислювальних потужностей. Зазвичай, ключ представляють як матрицю 4x4 байти.

На початку процесу шифрування, вхідні дані розбиваються на блоки розміром 16 байт або 128 біт. Якщо повний розмір даних не кратний 16 байтам

- дані доповнюються до розміру кратного 16 байтам. Блок даних в алгоритмі AES називається state і зазвичай представляється у вигляді матриці 4x4 байти (рис.14). Операція шифрування кожного блоку даних проводиться незалежно від вмісту інших блоків. По закінченню шифрування блоку - матриця заповнюється наступною порцією даних і процес повторюється. В силу незалежності шифрування одного блоку від іншого процес шифрування добре піддається розпаралелюванню.

$S_{00}$	$S_{01}$	$S_{02}$	$S_{03}$
$S_{10}$	$S_{11}$	$S_{12}$	$S_{13}$
$S_{20}$	$S_{21}$	$S_{22}$	$S_{23}$
$S_{30}$	$S_{31}$	$S_{32}$	$S_{33}$

Рисунок 14 – Представлення блоку state

Кожен блок шифрується в кілька етапів - раундів. Схема криптоперетворень може бути записана так:

1. розширення ключа KeyExpansion;
  2. початкова операція - AddRoundKey - підсумовування з основним ключем;
  3. 9 раундів з чотирьох кроків кожен:
    - 3.1. SubBytes - заміна байтів state по таблиці замін;
    - 3.2. ShiftRows - циклічний зсув рядків state;
    - 3.3. MixColumns - перестановка стовпців state;
    - 3.4. AddRoundKey - підсумовування з раундовим ключем;
  4. заключний 10-й раунд:
    - 4.1. SubBytes - заміна байтів state по таблиці замін;
    - 4.2. ShiftRows - циклічний зсув рядків state;
    - 4.3. AddRoundKey - підсумовування з раундовим ключем.
- Далі докладніше описано кожне з перетворень.

Перетворення SubBytes - це нелінійна заміна байт, що проводиться над кожним байтом state, використовуючи таблицю заміни S-box (рис. 15). Мета застосування таблиці замін - затруднити лінійний і диференціальний криптоаналіз. Таблиця замін в алгоритмі AES фіксована. У таблиці числа представлені в шістнадцятковій системі числення. В цій системі числення будь-яке значення байта можна представити не більше ніж двома шістнадцятковими розрядами.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Рисунок 15 – Таблиця заміни байта S-Box

Заміна байта по таблиці S-box проводиться так:

1. Байт  $Z$  переводиться в шістнадцяткову систему числення, наприклад  $XUh$ ,  $X$  - старший розряд,  $U$  - молодший розряд. Якщо старшого розряду немає - він замінюється нулем.
2. У S-box обирається рядок  $X$  і стовпець  $U$ .
3. Значення  $Z'$  на перетині рядка  $X$  і стовпці  $U$  таблиці S-box використовується як заміна  $Z$ .

Наприклад, для значення байта  $Z = 9Ah$ , заміною згідно таблиці S-box буде  $Z' = B8h$ . Повний процес SubBytes полягає в заміні всіх 16 байт матриці state.

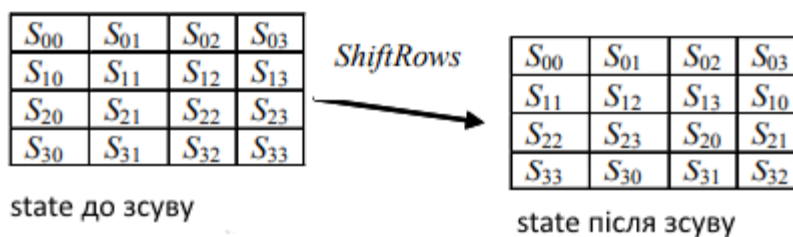
$S_{00}$	$S_{01}$	$S_{02}$	$S_{03}$		0	1	2	3	4	5	6	7	8	9	a	b	c	d	
$S_{10}$	$S_{11} = 9Ah$	$S_{12}$	$S_{13}$		00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7
$S_{20}$	$S_{21}$	$S_{22}$	$S_{23}$		10	ca	82	e9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4
$S_{30}$	$S_{31}$	$S_{32}$	$S_{33}$		20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8
					30	04	e7	23	c3	18	96	05	9a	07	12	80	e2	eb	27
					40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3
					50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	fc
					60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c
					70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff
					80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d
					90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e
					a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95
					b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a
					c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd
					d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1
					e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55
					f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54

state до заміни

state після заміни

Рисунок 16 – заміна одного байту за допомогою S-Box

У перетворенні ShiftRows байти в останніх трьох рядках state циклічно зміщуються вліво на різне число байт. Рядок 1 (нумерація рядків з нуля) Зміщується на один байт, рядок 2 - на два байти, рядок 3 - на три байти. На рис. 17 проілюстровано застосування перетворення ShiftRows до state.



state до зсуву

state після зсуву

Рисунок 17 – Процедура ShiftRows

У перетворенні MixColumns - перемішування стовпців - стовпці стану (state) розглядаються як поліноми над полем  $F(2^8)$  і множаться по модулю  $x^4 + 1$  на постійний поліном:

$$a(x) = 3x^3 + 1x^2 + 1x + 2$$

Процес множення поліномів еквівалентний матричному множенню

$$\begin{bmatrix} S'_{0c} \\ S'_{1c} \\ S'_{2c} \\ S'_{3c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \bullet \begin{bmatrix} S_{0c} \\ S_{1c} \\ S_{2c} \\ S_{3c} \end{bmatrix},$$

де  $c$  - номер стовпця масиву state і  $0 \leq c \leq 3$ .

В результаті такого множення, байти стовпчика  $c$   $\{S_{0c}, S_{1c}, S_{2c}, S_{3c}\}$  замінюються, відповідно, на байти:

$$\begin{aligned} S'_{0c} &= (2 \bullet S_{0c}) \oplus (3 \bullet S_{1c}) \oplus S_{2c} \oplus S_{3c}; \\ S'_{1c} &= S_{0c} \oplus (2 \bullet S_{1c}) \oplus (3 \bullet S_{2c}) \oplus S_{3c}; \\ S'_{2c} &= S_{0c} \oplus S_{1c} \oplus (2 \bullet S_{2c}) \oplus (3 \bullet S_{3c}); \\ S'_{3c} &= (3 \bullet S_{0c}) \oplus S_{1c} \oplus S_{2c} \oplus (2 \bullet S_{3c}). \end{aligned}$$

Перетворення застосовується до кожного з чотирьох стовпців state.

У процедурі AddRoundKey, раундовий ключ RK додається до state за допомогою порозрядного XOR. Кожен раундовий ключ складається з 16 байт розширеного ключа. Байти раундового ключа записуються в матрицю 4x4, подібну state. Кожен байт раундового ключа підсумовується з відповідним байтом з state, як показано на рис.18.



Рисунок 18 – Процедура AddRoundKey

Алгоритм AES бере ключ шифрування  $K$  і виконує операцію розширення ключа, щоб створити набір даних для раундових ключів. Розширений ключ  $W$  містить  $4 \cdot (10 + 1)$  слів - початковий ключ в 4 слова і по 4 слова розширеного ключа на кожен з 10 раундів. Розширений ключ  $W$  складається з слів (чотири байти на слово), що позначаються як  $w_i$ , де  $i$



знаходиться в діапазоні  $[0..44]$ . Повна довжина розширеного ключа складає 1408 біт, по 128 біт на кожен раунд.

У процесі розширення ключа використовується масив констант  $Rcon$ . Елементи масиву  $Rcon$  пронумеровані від 1 до  $256 + 3$ . Значення елементів масиву визначені наступним чином:

$$Rcon_1 = 1;$$

$$Rcon_k = 2 \cdot Rcon_{k-1} = 2^{k-1}, \text{ для } k = 2, 3, \dots, 255;$$

$$Rcon_k = 0, \text{ для } k = 256, 257, 258;$$

Розширення ключа можна описати такою послідовністю операцій:

1) чотири слова ключа шифрування  $K$  копіюються в перші чотири слова розширеного ключа  $W$ :  $w_i = k_i$  для  $i = 0, 1, 2, 3$ .

2) Решта чотири слова розширеного ключа  $W$  для  $i = 4, 5, \dots, 44$  генеруються так:

- якщо  $i$  кратно 4, то  $w_i = \text{SubBytes}(\text{RotByte}(w_{i-1})) \text{ xor } Rcon(i/4)$ ;
- якщо  $i$  не кратно 4, то  $w_i = w_{i-4} \text{ xor } w_{i-1}$ .

Функція  $\text{RotByte}$  переставляє чотири байти вихідного слова  $\{a_0, a_1, a_2, a_3\}$  за допомогою циклічної перестановки, перетворюючи в слово  $\{a_3, a_1, a_2, a_0\}$ . Функція  $\text{SubBytes}$  застосовує до кожного з чотирьох байтів слова заміну по таблиці  $S\text{-box}$ .

Раундовий ключ  $RK$  для раунду  $k$  вибирається з розширеного ключа  $W$  як слова с  $w_{4k}$  по  $w_{4(k+1)}$ .

Оскільки, всі перетворення шифрування однозначні і, то це означає, що вони мають зворотне перетворення, тобто можуть бути інвертовані і виконані в зворотному порядку, щоб виконати дешифрування для алгоритму AES.

Схема криптоперетворень може бути записана таким чином:

1. розширення ключа  $\text{KeyExpansion}$ ;
3. 9 раундів з чотирьох кроків кожен;
- 3.1.  $\text{AddRoundKey}$  - підсумовування з раундовим ключем;

- 3.2. InvMixColumns - зворотна перестановка стовпців state;
- 3.3. InvShiftRows - зворотний циклічний зсув рядків state;
- 3.4. InvSubBytes - зворотна заміна байтів state по таблиці замін;
- 4. Заключний 10-й раунд:
  - 4.1. AddRoundKey - підсумовування з раундовим ключем;
  - 4.2. InvShiftRows - зворотний циклічний зсув рядків state;
  - 4.3. InvSubBytes - зворотна заміна байтів state по таблиці замін.

Далі докладніше описано кожне з перетворень.

Перетворення InvMixColumns є зворотним для перетворення MixColumns. У перетворенні InvMixColumns, стовпці стану (state) розглядаються як поліноми над полем  $F(2^8)$  і множаться по модулю  $x^4 + 1$  з постійним поліномом  $d(x) = a^{-1}(x)$ , в полі  $F(2^8)$ :

$$d(x) = 0Bhx^3 + 0Dhx^2 + 09hx + 0Eh.$$

Перетворення InvShiftRows зворотне перетворенню ShiftRows. Байти останніх трьох рядів масиву state циклічно зсуваються вправо. Рядок 1 (нумерація з нуля) зміщується на 1 байт, рядок 2 - на 2 байти, рядок 3 - на 3 байти.

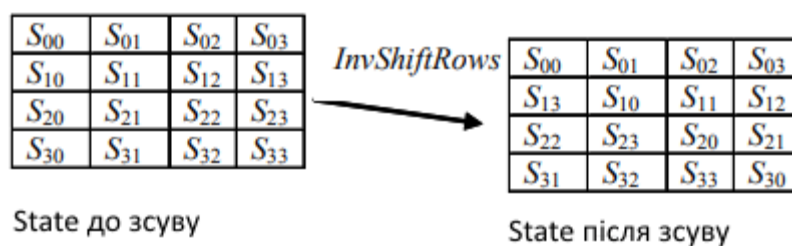


Рисунок 19 – Процедура InvShiftRows

Перетворення InvSubBytes виконує зворотну заміну байт за допомогою зворотної таблиці замін, назвемо її InvS-box. Зворотня таблиця замін приведена на рис. 20.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Рисунок 20 – Таблица InvS-box

Перетворення AddRoundKey є зворотнім для самого себе, так як використовує операцію XOR.

### 3.2 Тестування розробленої системи

Для тестування розробленої системи захисту ресурсів мережі SDN Була промодельована така мережа, яка включає в себе контролер та пристрої. Також для імітації атак були створені моделі зловмисників: Eva та Chack. Eva імітує атаку підслуховування трафіку, а Chack спробує провести авторизацію нелегітимного пристрою на рівні передачі даних.

Спочатку промодельюємо коректну авторизацію легітимного пристрою (рис. 21-26).

```

[INFO] [1542994201.126331]: I heard 1
[INFO] [1542994201.126839]: send ACK
[INFO] [1542994201.127562]: send: header = 0
[INFO] [1542994201.222351]: send: header = 2
[INFO] [1542994201.225755]: I heard 0
[INFO] [1542994201.319567]: send: header = 3
[INFO] [1542994201.322832]: I heard 0
[INFO] [1542994201.326797]: I heard 4
[INFO] [1542994201.327932]: send ACK
[INFO] [1542994201.329150]: send: header = 0
[INFO] [1542994201.418861]: client is authorized
[INFO] [1542994201.420459]: send: header = 6
[INFO] [1542994201.427081]: I heard 0
[INFO] [1542994201.518264]: send: header = 9
[INFO] [1542994201.520183]: I heard 0
[INFO] [1542994201.618932]: send: header = 10
[INFO] [1542994201.623148]: I heard 0
[INFO] [1542994201.719524]: send: header = 7
[INFO] [1542994202.250695]: I heard 8
[INFO] [1542994202.251164]: send ACK
[INFO] [1542994202.251584]: send: header = 0
[INFO] [1542994202.252105]: I heard 0
[INFO] [1542994202.347081]: cipher_key = 433142128654657676640136543834801
[INFO] [1542994204.423906]: I heard 5
[INFO] [1542994204.424351]: Receive data: Hello, world!
[INFO] [1542994204.424656]: send ACK
[INFO] [1542994204.424992]: send: header = 0
[INFO] [1542994207.525252]: I heard 5
[INFO] [1542994207.525666]: Receive data: Hello, world!
[INFO] [1542994207.525927]: send ACK
[INFO] [1542994207.526249]: send: header = 0

```

Рисунок 21 – Коректне функціонування контролеру

```

[INFO] [1542994207.526249]: send: header = 0
[INFO] [1542994210.626520]: I heard 5
[INFO] [1542994210.626935]: Receive data: Hello, world!
[INFO] [1542994210.627183]: send ACK
[INFO] [1542994210.627581]: send: header = 0
[INFO] [1542994212.418190]: send: header = 6
[INFO] [1542994212.795415]: I heard 0
[INFO] [1542994212.818143]: send: header = 9
[INFO] [1542994212.867858]: I heard 0
[INFO] [1542994212.919001]: send: header = 10
[INFO] [1542994212.921458]: I heard 0
[INFO] [1542994213.022786]: send: header = 7
[INFO] [1542994213.025826]: I heard 0
[INFO] [1542994213.373634]: I heard 8
[INFO] [1542994213.374225]: send ACK
[INFO] [1542994213.374824]: send: header = 0
[INFO] [1542994214.189511]: cipher_key = 325239150536601575437785117467521
[INFO] [1542994214.189755]: I heard 5
[INFO] [1542994214.190953]: Receive data: Hello, world!
[INFO] [1542994214.191223]: send ACK
[INFO] [1542994214.191714]: send: header = 0
[INFO] [1542994217.275206]: I heard 5
[INFO] [1542994217.275581]: Receive data: Hello, world!
[INFO] [1542994217.275835]: send ACK
[INFO] [1542994217.276253]: send: header = 0
[INFO] [1542994220.376493]: I heard 5
[INFO] [1542994220.376947]: Receive data: Hello, world!
[INFO] [1542994220.377234]: send ACK
[INFO] [1542994220.377742]: send: header = 0
[INFO] [1542994223.478276]: I heard 5
[INFO] [1542994223.478641]: Receive data: Hello, world!
[INFO] [1542994223.478929]: send ACK
[INFO] [1542994223.479305]: send: header = 0
[INFO] [1542994224.191036]: send: header = 6
[INFO] [1542994224.630834]: I heard 0
[INFO] [1542994224.691475]: send: header = 9

```

Рисунок 22 – Коректне функціонування контролеру (продовження)



```

[INFO] [1542994224.630834]: I heard 0
[INFO] [1542994224.691475]: send: header = 9
[INFO] [1542994224.694224]: I heard 0
[INFO] [1542994224.791801]: send: header = 10
[INFO] [1542994224.796511]: I heard 0
[INFO] [1542994224.895609]: send: header = 7
[INFO] [1542994224.899662]: I heard 0
[INFO] [1542994225.443905]: I heard 8
[INFO] [1542994225.444395]: send ACK
[INFO] [1542994225.444740]: send: header = 0
[INFO] [1542994226.405150]: cipher_key = 138918936553935804955893739401449
[INFO] [1542994226.579608]: I heard 5
[INFO] [1542994226.579986]: Receive data: Hello, world!
[INFO] [1542994226.580240]: send ACK
[INFO] [1542994226.580554]: send: header = 0
[INFO] [1542994229.681074]: I heard 5
[INFO] [1542994229.681451]: Receive data: Hello, world!
[INFO] [1542994229.681885]: send ACK
[INFO] [1542994229.682408]: send: header = 0
[INFO] [1542994232.781981]: I heard 5
[INFO] [1542994232.782378]: Receive data: Hello, world!
[INFO] [1542994232.782667]: send ACK
[INFO] [1542994232.782967]: send: header = 0
[INFO] [1542994235.883351]: I heard 5
[INFO] [1542994235.883762]: Receive data: Hello, world!
[INFO] [1542994235.884059]: send ACK
[INFO] [1542994235.884371]: send: header = 0
[INFO] [1542994236.406061]: send: header = 6
[INFO] [1542994236.744254]: I heard 0
[INFO] [1542994236.806653]: send: header = 9
[INFO] [1542994236.810339]: I heard 0
[INFO] [1542994236.906871]: send: header = 10
[INFO] [1542994236.911148]: I heard 0
[INFO] [1542994237.010219]: send: header = 7
[INFO] [1542994237.014036]: I heard 0
[INFO] [1542994237.233036]: I heard 8

```

Рисунок 23 – Коректне функціонування контролера (продовження)

```

[INFO] [1542994201.126227]: send: header = 1
[INFO] [1542994201.223424]: I heard 2
[INFO] [1542994201.224222]: send ACK
[INFO] [1542994201.225086]: send: header = 0
[INFO] [1542994201.320593]: I heard 3
[INFO] [1542994201.321578]: send ACK
[INFO] [1542994201.322573]: send: header = 0
[INFO] [1542994201.324701]: controller is authorized
[INFO] [1542994201.326060]: send: header = 4
[INFO] [1542994201.329229]: I heard 0
[INFO] [1542994201.420654]: I heard 6
[INFO] [1542994201.422213]: send ACK
[INFO] [1542994201.424184]: send: header = 0
[INFO] [1542994201.518939]: I heard 9
[INFO] [1542994201.519495]: send ACK
[INFO] [1542994201.520064]: send: header = 0
[INFO] [1542994201.619990]: I heard 10
[INFO] [1542994201.621290]: send ACK
[INFO] [1542994201.623015]: send: header = 0
[INFO] [1542994201.720307]: I heard 7
[INFO] [1542994201.721639]: send ACK
[INFO] [1542994202.250775]: send: header = 0
[INFO] [1542994202.251240]: send: header = 0
[INFO] [1542994202.251771]: cipher_key = 433142128654657676640136543834801
[INFO] [1542994202.252280]: I heard 0
[INFO] [1542994204.423368]: Send data: Hello, world!
[INFO] [1542994204.423820]: send: header = 5
[INFO] [1542994204.425386]: I heard 0
[INFO] [1542994207.524597]: Send data: Hello, world!
[INFO] [1542994207.525064]: send: header = 5
[INFO] [1542994207.526363]: I heard 0
[INFO] [1542994210.625977]: Send data: Hello, world!
[INFO] [1542994210.626394]: send: header = 5
[INFO] [1542994210.628028]: I heard 0

```

Рисунок 24 – Коректне функціонування пристрою рівня передачі даних

```

[INFO] [1542994212.595221]: I heard 6
[INFO] [1542994212.764574]: send ACK
[INFO] [1542994212.795365]: send: header = 0
[INFO] [1542994212.866869]: I heard 9
[INFO] [1542994212.867180]: send ACK
[INFO] [1542994212.867568]: send: header = 0
[INFO] [1542994212.919001]: I heard 10
[INFO] [1542994212.920062]: send ACK
[INFO] [1542994212.921061]: send: header = 0
[INFO] [1542994213.023073]: I heard 7
[INFO] [1542994213.024385]: send ACK
[INFO] [1542994213.025728]: send: header = 0
[INFO] [1542994213.372974]: send: header = 8
[INFO] [1542994213.375065]: I heard 0
[INFO] [1542994213.474193]: cipher_key = 325239150536601575437785117467521
[INFO] [1542994213.727452]: Send data: Hello, world!
[INFO] [1542994213.727896]: send: header = 5
[INFO] [1542994214.191840]: I heard 0
[INFO] [1542994217.274580]: Send data: Hello, world!
[INFO] [1542994217.274985]: send: header = 5
[INFO] [1542994217.276311]: I heard 0
[INFO] [1542994220.375898]: Send data: Hello, world!
[INFO] [1542994220.376377]: send: header = 5
[INFO] [1542994220.377800]: I heard 0
[INFO] [1542994223.477405]: Send data: Hello, world!
[INFO] [1542994223.477843]: send: header = 5
[INFO] [1542994223.479586]: I heard 0
[INFO] [1542994224.246663]: I heard 6
[INFO] [1542994224.414632]: send ACK
[INFO] [1542994224.630753]: send: header = 0
[INFO] [1542994224.691588]: I heard 9
[INFO] [1542994224.692504]: send ACK
[INFO] [1542994224.693526]: send: header = 0
[INFO] [1542994224.793129]: I heard 10
[INFO] [1542994224.794185]: send ACK
[INFO] [1542994224.795766]: send: header = 0

```

Рисунок 25 – Коректне функціонування пристрою рівня передачі даних (продовження)

```

[INFO] [1542994224.897008]: I heard 7
[INFO] [1542994224.898192]: send ACK
[INFO] [1542994224.899203]: send: header = 0
[INFO] [1542994225.443874]: send: header = 8
[INFO] [1542994225.444916]: I heard 0
[INFO] [1542994225.544689]: cipher_key = 138918936553935804955893739401449
[INFO] [1542994226.578833]: Send data: Hello, world!
[INFO] [1542994226.579225]: send: header = 5
[INFO] [1542994226.580754]: I heard 0
[INFO] [1542994229.680203]: Send data: Hello, world!
[INFO] [1542994229.680593]: send: header = 5
[INFO] [1542994229.682652]: I heard 0
[INFO] [1542994232.781359]: Send data: Hello, world!
[INFO] [1542994232.781770]: send: header = 5
[INFO] [1542994232.783160]: I heard 0
[INFO] [1542994235.882681]: Send data: Hello, world!
[INFO] [1542994235.883136]: send: header = 5
[INFO] [1542994235.884661]: I heard 0
[INFO] [1542994236.498664]: I heard 6
[INFO] [1542994236.558670]: send ACK
[INFO] [1542994236.691088]: send: header = 0
[INFO] [1542994236.807897]: I heard 9
[INFO] [1542994236.809009]: send ACK
[INFO] [1542994236.810224]: send: header = 0
[INFO] [1542994236.907670]: I heard 10
[INFO] [1542994236.909090]: send ACK
[INFO] [1542994236.910425]: send: header = 0
[INFO] [1542994237.011328]: I heard 7
[INFO] [1542994237.012497]: send ACK
[INFO] [1542994237.013856]: send: header = 0
[INFO] [1542994237.233139]: send: header = 8
[INFO] [1542994237.234044]: I heard 0
[INFO] [1542994237.244687]: cipher_key = 446150953453282219748716239463041
[INFO] [1542994238.984554]: Send data: Hello, world!
[INFO] [1542994238.984970]: send: header = 5
[INFO] [1542994238.986802]: I heard 0

```

Рисунок 26 – Коректне функціонування пристрою рівня передачі даних (продовження)

Підтвердження правильності роботи моделей можна побачити з того, що їх виконання відбувається згідно з обраними алгоритмами.

Значення отриманих запитів:

- 0 – підтвердження;
- 1 – запит пристрою на автентифікацію;
- 2 – відповідна реакція контролера на запит автентифікації пристрою;
- 3 – відповідь контролера з передачею НМАС;
- 4 – відповідна реакція пристрою на автентифікацію;
- 5 – безпосередньо передача трафіку;
- 6 – генерація ключа;
- 7 – число контролера для генерації ключа;
- 8 – число пристрою для генерації ключа;
- 9 – число  $g$ , що приймає участь в генерації ключа;
- 10 – число  $p$ , що приймає участь в генерації ключа.

Згенерований граф передачі даних представлено на рис.27.

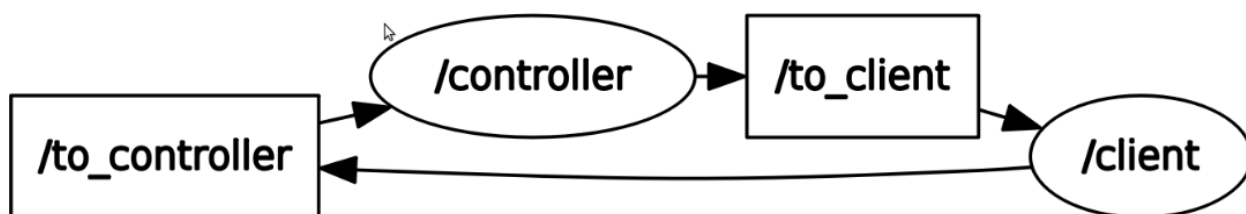


Рисунок 27 – Граф передачі даних

Далі продемонструємо які зміни відбуватимуться в даній моделі при наявності пристрою, налаштованого на прослуховування даних, що йде від контролера до легітимного пристрою та зворотному напрямку. Контролер (рис. 28-30), а також легітимний пристрій (рис. 31-33) мають санкціонований доступ до мережі, а ще один пристрій, під кодовим іменем Єва (рис. 34) намагається прослухати дані між пристроєм та контролером.



```

[INFO] [1542994904.144087]: I heard 1
[INFO] [1542994904.144745]: send ACK
[INFO] [1542994904.145367]: send: header = 0
[INFO] [1542994904.158873]: send: header = 2
[INFO] [1542994904.160604]: I heard 0
[INFO] [1542994904.257156]: send: header = 3
[INFO] [1542994904.260624]: I heard 0
[INFO] [1542994904.345639]: I heard 4
[INFO] [1542994904.346820]: send ACK
[INFO] [1542994904.348287]: send: header = 0
[INFO] [1542994904.356250]: client is authorized
[INFO] [1542994904.358795]: send: header = 6
[INFO] [1542994904.359450]: I heard 0
[INFO] [1542994904.456476]: send: header = 9
[INFO] [1542994904.461410]: I heard 0
[INFO] [1542994904.556823]: send: header = 10
[INFO] [1542994904.560089]: I heard 0
[INFO] [1542994904.660144]: send: header = 7
[INFO] [1542994904.663543]: I heard 0
[INFO] [1542994905.034718]: I heard 8
[INFO] [1542994905.035213]: send ACK
[INFO] [1542994905.035741]: send: header = 0
[INFO] [1542994905.996390]: cipher_key = 415520726437375556741987950492969
[INFO] [1542994907.443701]: I heard 5
[INFO] [1542994907.444125]: Receive data: Hello, world!
[INFO] [1542994907.444375]: send ACK
[INFO] [1542994907.444672]: send: header = 0
[INFO] [1542994910.545060]: I heard 5
[INFO] [1542994910.545471]: Receive data: Hello, world!
[INFO] [1542994910.545754]: send ACK
[INFO] [1542994910.546209]: send: header = 0
[INFO] [1542994913.646308]: I heard 5
[INFO] [1542994913.646795]: Receive data: Hello, world!
[INFO] [1542994913.647112]: send ACK
[INFO] [1542994913.647513]: send: header = 0

```

Рисунок 28 – Функціонування контролера в мережі з підслуховувачем

```

[INFO] [1542994915.997356]: send: header = 6
[INFO] [1542994916.422861]: I heard 0
[INFO] [1542994916.497420]: send: header = 9
[INFO] [1542994916.510119]: I heard 0
[INFO] [1542994916.597932]: send: header = 10
[INFO] [1542994916.600928]: I heard 0
[INFO] [1542994916.703722]: send: header = 7
[INFO] [1542994916.709042]: I heard 0
[INFO] [1542994916.857944]: I heard 8
[INFO] [1542994916.858475]: send ACK
[INFO] [1542994916.858953]: send: header = 0
[INFO] [1542994918.013747]: cipher_key = 201149080432077387120557693789203
[INFO] [1542994918.014036]: I heard 5
[INFO] [1542994918.015071]: Receive data: Hello, world!
[INFO] [1542994918.015311]: send ACK
[INFO] [1542994918.015604]: send: header = 0
[INFO] [1542994921.110112]: I heard 5
[INFO] [1542994921.110577]: Receive data: Hello, world!
[INFO] [1542994921.110904]: send ACK
[INFO] [1542994921.111334]: send: header = 0
[INFO] [1542994924.211652]: I heard 5
[INFO] [1542994924.212119]: Receive data: Hello, world!
[INFO] [1542994924.212443]: send ACK
[INFO] [1542994924.212788]: send: header = 0
[INFO] [1542994927.313218]: I heard 5
[INFO] [1542994927.313666]: Receive data: Hello, world!
[INFO] [1542994927.313923]: send ACK
[INFO] [1542994927.314335]: send: header = 0
[INFO] [1542994928.015340]: send: header = 6
[INFO] [1542994928.313317]: I heard 0
[INFO] [1542994928.315356]: send: header = 9
[INFO] [1542994928.559672]: I heard 0

```

Рисунок 29 – Функціонування контролера в мережі з підслуховувачем (продовження)



```
[INFO] [1542994927.314335]: send: header = 0
[INFO] [1542994928.015340]: send: header = 6
[INFO] [1542994928.313317]: I heard 0
[INFO] [1542994928.315356]: send: header = 9
[INFO] [1542994928.559672]: I heard 0
[INFO] [1542994928.615580]: send: header = 10
[INFO] [1542994928.618591]: I heard 0
[INFO] [1542994928.718073]: send: header = 7
[INFO] [1542994928.723982]: I heard 0
[INFO] [1542994929.104961]: I heard 8
[INFO] [1542994929.105311]: send ACK
[INFO] [1542994929.105700]: send: header = 0
[INFO] [1542994929.601599]: cipher_key = 853980255956904454592642353109361
[INFO] [1542994930.414828]: I heard 5
[INFO] [1542994930.415227]: Receive data: Hello, world!
[INFO] [1542994930.415516]: send ACK
[INFO] [1542994930.415916]: send: header = 0
[INFO] [1542994933.516414]: I heard 5
[INFO] [1542994933.516836]: Receive data: Hello, world!
[INFO] [1542994933.517094]: send ACK
[INFO] [1542994933.517421]: send: header = 0
[INFO] [1542994936.618116]: I heard 5
[INFO] [1542994936.618584]: Receive data: Hello, world!
[INFO] [1542994936.618848]: send ACK
[INFO] [1542994936.619262]: send: header = 0
[INFO] [1542994939.602481]: send: header = 6
[INFO] [1542994939.719572]: I heard 0
[INFO] [1542994939.720769]: I heard 5
[INFO] [1542994939.721145]: Receive data: Hello, world!
[INFO] [1542994939.721390]: send ACK
[INFO] [1542994939.721703]: send: header = 0
[INFO] [1542994939.803362]: send: header = 9
^C[INFO] [1542994962.644219]: send: header = 10
```

Рисунок 30 – Функціонування контролера в мережі з підслуховувачем  
(продовження)

```

[INFO] [1542994904.143948]: send: header = 1
[INFO] [1542994904.159563]: I heard 2
[INFO] [1542994904.160007]: send ACK
[INFO] [1542994904.160446]: send: header = 0
[INFO] [1542994904.257918]: I heard 3
[INFO] [1542994904.258876]: send ACK
[INFO] [1542994904.259772]: send: header = 0
[INFO] [1542994904.343535]: controller is authorized
[INFO] [1542994904.344576]: send: header = 4
[INFO] [1542994904.348503]: I heard 0
[INFO] [1542994904.357559]: I heard 6
[INFO] [1542994904.358351]: send ACK
[INFO] [1542994904.358937]: send: header = 0
[INFO] [1542994904.458235]: I heard 9
[INFO] [1542994904.459445]: send ACK
[INFO] [1542994904.460749]: send: header = 0
[INFO] [1542994904.556962]: I heard 10
[INFO] [1542994904.558459]: send ACK
[INFO] [1542994904.559867]: send: header = 0
[INFO] [1542994904.660864]: I heard 7
[INFO] [1542994904.661901]: send ACK
[INFO] [1542994904.662895]: send: header = 0
[INFO] [1542994905.034493]: send: header = 8
[INFO] [1542994905.035780]: I heard 0
[INFO] [1542994905.042445]: cipher_key = 415520726437375556741987950492969
[INFO] [1542994907.443082]: Send data: Hello, world!
[INFO] [1542994907.443465]: send: header = 5
[INFO] [1542994907.444755]: I heard 0
[INFO] [1542994910.544344]: Send data: Hello, world!
[INFO] [1542994910.544786]: send: header = 5
[INFO] [1542994910.546736]: I heard 0
[INFO] [1542994913.645764]: Send data: Hello, world!
[INFO] [1542994913.646195]: send: header = 5
[INFO] [1542994913.647607]: I heard 0
[INFO] [1542994915.997621]: I heard 6

```

Рисунок 31 – Функціонування легітимного пристрою в мережі з підслуховувачем

```

[INFO] [1542994916.226643]: send ACK
[INFO] [1542994916.430666]: send: header = 0
[INFO] [1542994916.509102]: I heard 9
[INFO] [1542994916.509556]: send ACK
[INFO] [1542994916.510185]: send: header = 0
[INFO] [1542994916.598832]: I heard 10
[INFO] [1542994916.599566]: send ACK
[INFO] [1542994916.600539]: send: header = 0
[INFO] [1542994916.704583]: I heard 7
[INFO] [1542994916.706080]: send ACK
[INFO] [1542994916.707423]: send: header = 0
[INFO] [1542994916.857902]: send: header = 8
[INFO] [1542994916.858969]: I heard 0
[INFO] [1542994916.909117]: cipher_key = 201149080432077387120557693789203
[INFO] [1542994916.909638]: Send data: Hello, world!
[INFO] [1542994916.910025]: send: header = 5
[INFO] [1542994918.015870]: I heard 0
[INFO] [1542994921.109590]: Send data: Hello, world!
[INFO] [1542994921.110061]: send: header = 5
[INFO] [1542994921.111755]: I heard 0
[INFO] [1542994924.211104]: Send data: Hello, world!
[INFO] [1542994924.211581]: send: header = 5
[INFO] [1542994924.213168]: I heard 0
[INFO] [1542994927.312652]: Send data: Hello, world!
[INFO] [1542994927.313151]: send: header = 5
[INFO] [1542994927.314352]: I heard 0
[INFO] [1542994928.015772]: I heard 6
[INFO] [1542994928.050663]: send ACK
[INFO] [1542994928.314124]: send: header = 0
[INFO] [1542994928.558703]: I heard 9
[INFO] [1542994928.559128]: send ACK
[INFO] [1542994928.559594]: send: header = 0
[INFO] [1542994928.616924]: I heard 10
[INFO] [1542994928.617746]: send ACK
[INFO] [1542994928.618581]: send: header = 0

```

Рисунок 32– Функціонування легітимного пристрою в мережі з підслуховувачем (продовження)



```

[INFO] [1542994927.314352]: I heard 0
[INFO] [1542994928.015772]: I heard 6
[INFO] [1542994928.050663]: send ACK
[INFO] [1542994928.314124]: send: header = 0
[INFO] [1542994928.558703]: I heard 9
[INFO] [1542994928.559128]: send ACK
[INFO] [1542994928.559594]: send: header = 0
[INFO] [1542994928.616924]: I heard 10
[INFO] [1542994928.617746]: send ACK
[INFO] [1542994928.618581]: send: header = 0
[INFO] [1542994928.718851]: I heard 7
[INFO] [1542994928.720638]: send ACK
[INFO] [1542994928.722221]: send: header = 0
[INFO] [1542994929.104120]: send: header = 8
[INFO] [1542994929.105713]: I heard 0
[INFO] [1542994929.204994]: cipher_key = 853980255956904454592642353109361
[INFO] [1542994930.414279]: Send data: Hello, world!
[INFO] [1542994930.414787]: send: header = 5
[INFO] [1542994930.415887]: I heard 0

```

Рисунок 33– Функціонування легітимного пристрою в мережі з підслуховувачем (продовження)

```

[INFO] [1542994916.601025]: I intercepted the message from client with header = 0 and data:
[INFO] [1542994916.704489]: I intercepted the message from controller with header = 7 and data: 601080015116217976750887436584987
[INFO] [1542994916.708308]: I intercepted the message from client with header = 0 and data:
[INFO] [1542994916.858245]: I intercepted the message from client with header = 8 and data: 37216094714607752962296201158827
[INFO] [1542994916.859002]: I intercepted the message from controller with header = 0 and data:
[INFO] [1542994916.910290]: I intercepted the message from client with header = 5 and data: }t++8++Q5+*++D+4++3[]cZ
[INFO] [1542994918.015906]: I intercepted the message from controller with header = 0 and data:
[INFO] [1542994921.110709]: I intercepted the message from client with header = 5 and data: ++kQb[++S++[]n+++Y++ ++n++4
[INFO] [1542994921.111373]: I intercepted the message from controller with header = 0 and data:
[INFO] [1542994924.212147]: I intercepted the message from client with header = 5 and data: ++{++<+p++<}+++C+i++k0N ++h++
[INFO] [1542994924.212826]: I intercepted the message from controller with header = 0 and data:
[INFO] [1542994927.313588]: I intercepted the message from client with header = 5 and data: US+++ ++>+x++N++[]U++L0+S5B
[INFO] [1542994927.314526]: I intercepted the message from controller with header = 0 and data:
[INFO] [1542994928.015695]: I intercepted the message from controller with header = 6 and data:
[INFO] [1542994928.313707]: I intercepted the message from client with header = 0 and data:
[INFO] [1542994928.315356]: I intercepted the message from controller with header = 9 and data: 3
[INFO] [1542994928.559675]: I intercepted the message from client with header = 0 and data:
[INFO] [1542994928.616925]: I intercepted the message from controller with header = 10 and data:
[INFO] [1542994928.618634]: I intercepted the message from client with header = 0 and data:
[INFO] [1542994928.718986]: I intercepted the message from controller with header = 7 and data: 321211138785491103710039177581769
[INFO] [1542994928.722348]: I intercepted the message from client with header = 0 and data:
[INFO] [1542994929.104600]: I intercepted the message from client with header = 8 and data: 582215481283767392427949307890009
[INFO] [1542994929.105888]: I intercepted the message from controller with header = 0 and data:
[INFO] [1542994930.414860]: I intercepted the message from client with header = 5 and data: ,++Qg+'L++8
"++S+++n++J+++Pi

```

Рисунок 34– Функціонування підслуховувача

Як видно з наведених зображень роботи моделі, коли в мережі з'являється пристрій, який здатний читати трафік, мережеві пристрої та контролер продовжують функціонувати в нормальному режимі. Підслуховувач навіть отримуючи дані все одно нічого корисного з них отримати не здатен, оскільки вони повністю зашифровані. Навіть у випадку, коли підслуховувач якимось чином отримає ключ, розшифрувати він зможе лише частину повідомлень. Він ніяк не зможе розшифрувати повідомлення, що були до встановлення цього ключа і ті, що будуть після встановлення нового ключа. Можна стверджувати, що завдяки цьому в мережі забезпечена секретність майбутніх та минулих повідомлень.

На рис. 33 представлено згенерований граф мережі з підслухувачем.

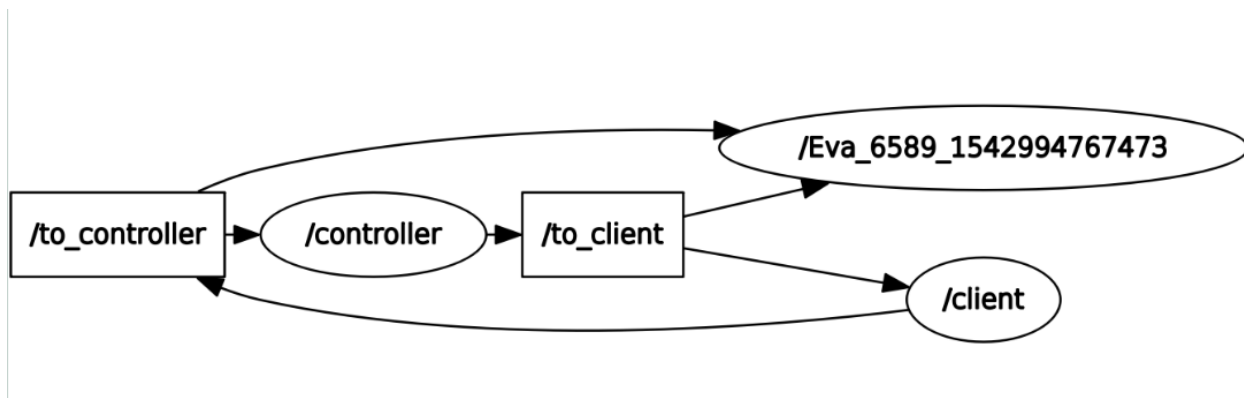


Рисунок 35 - Згенерований граф передачі даних у мережі з підслухувачем (Eva)

Розглянемо моделювання роботи контролера у випадку спроби приєднання до нього нелегітимного пристрою.

```

[INFO] [1542995495.000681]: I heard 1
[INFO] [1542995495.002166]: send ACK
[INFO] [1542995495.003797]: send: header = 0
[INFO] [1542995495.032058]: send: header = 2
[INFO] [1542995495.036292]: I heard 0
[INFO] [1542995495.129458]: send: header = 3
[INFO] [1542995495.134004]: I heard 0
[INFO] [1542995495.201040]: I heard 4
[INFO] [1542995495.203176]: send ACK
[INFO] [1542995495.204654]: send: header = 0
[INFO] [1542995495.228782]: client isn't authorized
  
```

Рисунок 36 – Результат роботи контролера при спробі приєднання нелегітимного пристрою

```

[INFO] [1542995492.901287]: send: header = 1
[INFO] [1542995495.004054]: I heard 0
[INFO] [1542995495.033168]: I heard 2
[INFO] [1542995495.034485]: send ACK
[INFO] [1542995495.035974]: send: header = 0
[INFO] [1542995495.130394]: I heard 3
[INFO] [1542995495.131810]: send ACK
[INFO] [1542995495.133051]: send: header = 0
[INFO] [1542995495.201035]: send: header = 4
[INFO] [1542995495.204833]: I heard 0
  
```

Рисунок 36 – Результат спроби приєднання нелегітимного пристрою

Оскільки, зловмиснику бракує згенерованого нами ключа шифрування він ніяк не зможе автентифікувати нелегітимний пристрій в даній мережі.

### Висновки до розділу 3

Розроблено спосіб та систему захисту ресурсів програмно-конфігуровної мережі, наведені та детально проаналізовані елементи, які використовуються у розробленій системі захисту ресурсів програмно-конфігуровної мережі. До таких елементів увійшли: генерування секретних ключів за допомогою протоколу Діффі-Хеллмана, автентифікація пристрою в мережі та шифрування даних, що передаються південним шляхом, тобто між контролером та пристроями рівня передачі даних.

Слід зазначити, що використання цієї системи потребує більшої потужності від елементів мережі, ніж інші, менш комплексні, системи захисту. Проте, вона захищає мережу від одних з найрозповсюдженіших типів атак, а саме підслуховування та спроба автентифікації пристрою, який не є легітимним для даної мережі. Тому використання розробленої системи захисту ресурсів доцільно лише в мережах з підвищеними вимогами до безпеки.

Промодельовано роботу даної системи захисту і наведені результати її відпрацювання у випадках коректної роботи та при різних видах атаки.

## Висновки до роботи

Проведена робота розглядає такий аспект досить нових на наш час програмно-конфігурованих мереж SDN, як забезпечення захисту їх інформаційних ресурсів. Було розглянуто та проаналізовано структуру мережі SDN, а також протоколів, які вона найчастіше використовує. Проаналізовані характеристики та вразливості як мережі SDN, так і розглянутих протоколів.

Проаналізовані основні напрямки атак на мережі SDN та наведені вразливості кожного архітектурного рівня. Проведено огляд та порівняльний аналіз існуючих систем захисту.

Також запропонована власна система захисту, наведені та детально проаналізовані елементи, котрі використовуються у ній. До таких елементів увійшли: генерування секретних ключів за допомогою протоколу Діффі-Хеллмана, автентифікація пристрою в мережі та шифрування даних, що передаються між контролером та пристроями рівня передачі даних.

З'ясовано, що використання запропонованої системи потребує більшої потужності від елементів мережі, ніж інші, менш комплексні, системи захисту. Проте, вона захищає мережу від одних з найрозповсюдженіших типів атак. Тому використання розробленої системи захисту ресурсів доцільно лише в мережах з підвищеними вимогами до безпеки.

Також була промодельована робота даної системи захисту і наведені результати її відпрацювання у випадках роботи в нормальному середовищі та при різних видах атаки.

## Список використаної літератури

1. *K. Calvert, S. Bhattacharjee, E. Zegura, and J. Sterbenz*, Directions in Active Networks IEEE Communications magazine, p. 72–78, October 1998.
2. *Diego Kreutz, Fernando MV Ramos, P Esteves Verissimo, Christian Esteve Rothenberg, Sia-mak Azodolmolky, and Steve Uhlig*. Software-defined networking: A comprehensive survey. proceedings of the IEEE, 103[1]:14–76, 2015.
3. *Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner*. Openflow: enabling innovation in campus net-works. ACM SIGCOMM Computer Communication Review, 38[2]:69–74, 2008.
4. *Diego Kreutz, Fernando Ramos, and Paulo Verissimo*. Towards secure and dependable soft-ware-defined networks. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, p. 55–60. 2013.
5. *Смелянский Р. Л.* Программно-конфигурируемые сети. Открытые системы. СУБД 9. 2012. с. 23–26.
6. OpenFlow Switch Specification Ver 1.5.1, 2016 [accessed January 11, 2016]. <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf>.
7. *Никульчев Е. В., Паяин С. В., Плужник Е. В.* Динамическое управление трафиком программно-конфигурируемых сетей в облачной инфраструктуре. Вестник РГРТУ. № 3. 2013. с. 45.
8. *Sandra Scott-Hayward, Gemma O’Callaghan, and Sakir Sezer*. Sdn security: A survey. In Future Networks and Services (SDN4FNS), 2013 IEEE SDN For, p. 1–7. IEEE, 2013.



9. Open Networking Foundation. Software-defined networking: The new norm for networks. ONF White Paper, 2012.
10. *Kevin Benton, L Jean Camp, and Chris Small*. Openflow vulnerability assessment. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined net-working, p. 151–152. 2013.
11. *Seungwon Shin and Guofei Gu*. Attacking software-defined networks: A first feasibility study. In Proceedings of the second ACM SIGCOMM work- shop on Hot topics in software defined networking, p. 165–166. 2013.
12. *Diego Kreutz, Fernando Ramos, and Paulo Verissimo*. Towards secure and dependable soft-ware-defined networks. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, p. 55–60. 2013.
13. L. Schehlmann, S. Abt and H. Baier, 'Blessing or curse? Revisiting security aspects of Software-Defined Networking', International Conference on Network and Service Management (CNSM), pp. 382- 387, 2014.
14. D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards Secure and Dependable Software-Defined Networks," in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN '13. ACM, August 2013, pp. 55–60.
15. H. Li, P. Li, S. Guo, and S. Yu, "Byzantine-resilient secure software- defined networks with multiple controllers," in Communications (ICC), 2014 IEEE International Conference on. IEEE, 2014, pp. 695–700.
16. S. Scott-Hayward, C. Kane, and S. Sezer, "OperationCheckpoint: SDN Application Control," in 22nd IEEE International Conference on Network Protocols (ICNP). IEEE, 2014, pp. 618–623.
17. J. Hizver, 'Taxonomic Modelling of Security Threats in Software Defined Networking', BlackHat Conference, pp. 1-16, 2015.

18. S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *Communications Magazine, IEEE*, vol. 51, no. 7, 2013
19. E. de la Hoz, R. Paez-Reyes, G. Cochrane, I. Marsa-Maestre, J. Moreira Lemus and B. Alarcos, "Detecting and Defeating Advanced Man-In-The-Middle Attacks against TLS", *International Conference on Cyber Conflict*, pp. 209-221, 2014.
20. H. Dai, Q. Wang, D. Li and R. Chi-Wing Wong, "On Eavesdropping Attacks in Wireless Sensor Networks with Directional Antennas", *International Journal of Distributed Sensor Networks*, pp. 1-13
21. F. Jiang, C. Song, H. Xun and Z. Xu, "Combat-Sniff: A Comprehensive Countermeasure to Resist Data Plane Eavesdropping in Software-Defined Networks", *American Journal of Networks and Communications*, vol. 5, no. 2, pp. 27-34, 2016.
22. S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & Communications Security*. ACM, 2013, pp. 413–424.
23. P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient OpenFlow-based networking," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 933–939.
24. L. Wei and C. Fung, "FlowRanger: A Request Prioritizing Algorithm for Controller DoS Attacks in Software Defined Networks", *Next Generation Networking Symposium*, pp. 5254-5259, 2015.
25. N. Dao, J. Park, M. Park and S. Cho, "A feasible method to combat against DDoS attack in SDN Network", *International Conference on Information Networking*, pp. 309-311, 2015.

26. G. Yao, J. Bi, and P. Xiao, "Source address validation solution with OpenFlow/NOX architecture," in 19th IEEE International Conference on Network Protocols (ICNP). IEEE, 2011, pp. 7–12.
27. P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," in Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 121–126.
28. S. Mousavi and M. St-Hilaire, "Early Detection of DDoS Attacks against SDN Controllers", International Conference on Computing, Networking and Communications, pp. 77-81, 2015.
29. MM and K. Okamura, "Securing Distributed Control of Software Defined Networks," in International Journal of Computer Science & Network Security, vol. 13, no. 9, 2013.
30. M. Othman and K. Okamura, "Hybrid Control Model for Flow-Based Networks," in the international conference COMPSAC 2013 - The First IEEE International Workshop on Future Internet Technologies, Kyoto, Japan, 2013.
31. D. M. F. Mattos, L. H. G. Ferraz, and O. C. M. B. Duarte, "AuthFlow: Authentication and Access Control Mechanism for Software Defined Networking.", pp. 1-7.
32. Q. Duan, E. Al-Shaer and H. Jafarian, "Efficient Random Route Mutation considering flow and network constraints", Conference on Communications and Network Security (CNS), pp. 260-268, 2013.
33. M. Wang, J. Liu, J. Chen, X. Liu and J. Mao, "PERM-GUARD: Authenticating the Validity of Flow Rules in Software Defined Networking", International Conference on Cyber Security and Cloud Computing, pp. 127-133, 2015.
34. S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. B. Kang, "Rosemary: A Robust, Secure, and

HighPerformance Network Operating System,” in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014, pp. 78–89.

35. B. Chandrasekaran and T. Benson, “Tolerating SDN application failures with LegoSDN,” in Proceedings of the 13th ACM Workshop on Hot Topics in Networks. ACM, 2014, p. 22.

36. P. Zanna, B. O'Neill, P. Radcliffe, S. Hosseini and S. Ul Hoque, 'Adaptive threat management through the integration of IDS into Software Defined Networks', 2014 International Conference and Workshop on the Network of the Future, pp. 1-5, 2014.

37. Система захисту ресурсів у мережах SDN: матеріали 11-ї міжнар. наук.-техн. конф. ПМК'2018, Київ, 14 – 16 листопада 2018 р.: Збірник тез доповідей. – К.: Просвіта, 2018.

38. Алгоритм автентифікації пристрою в мережі SDN: матеріали п'ятої міжнародної науково-технічної internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами», Київ, 22 – 23 листопада 2018 р.: Збірник тез доповідей.