Міністерство освіти і науки України
Державний вищий навчальний заклад
«Національний гірничий університет»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
дипломної роботи

магістра
(назва освітньо-кваліфікаційного рівня)

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

напрям підготовки *125 Кібербезпека*
(код і назва напряму підготовки)

спеціальність *Кібербезпека*
(код і назва спеціальності)

освітній рівень *магістр*
(назва освітнього рівня)

кваліфікація *професіонал із організації інформаційної безпеки*
(код і назва кваліфікації)

на тему: ***Розробка та дослідження програмної системи для аналізу криптографічних протоколів***

Виконавець: студент ***6*** **курсу, групи** ***125м-16-1***

Сізинцев Микита Андрійович
(підпис) (прізвище ім'я по-батькові)

Керівники	Прізвище, ініціали	Оцінка	Підпис
роботи	д.ф.-м.н., проф. Кагадій Т.С.		
розділів:			
спеціальний	ст. викл. Саксонов Г.М.		
економічний	к.е.н., доц. Волотковська Ю.О.		
Рецензент			
Нормоконтроль	к.ф.-м.н., доц. Гусєв О. Ю.		

Дніпропетровськ
2018

Міністерство освіти і науки України
Державний вищий навчальний заклад
«Національний гірничий університет»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ орнієнко В.І.

«_____»_____2018 року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *Кібербезпека*
(код і назва спеціальності)

студенту **125м-16-1** **Сізинцев Микита Андрійович**
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи ***Розробка та дослідження програмної системи для аналізу криптографічних протоколів***

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора Державного ВНЗ «НГУ» від _____№_____

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень ***Методи аналізу криптографічних протоколів***

Предмет досліджень ***Архітектура системи моделювання криптографічних протоколів***

Мета НДР ***Підвищення якості захисту криптографічних систем***

Вихідні дані для проведення роботи ***результати та матеріали з переддипломної практики***

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна ***Запропоновано новий уточнений вид опису протоколів, запропонована концептуальна модель системи моделювання, запропонована структура програмної реалізації системи моделювання протоколів.***

Практична цінність ***полягає в розробці архітектури та структури програмної системи моделювання криптографічних протоколів***

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє створити технічне завдання та програмний продукт для моделювання криптографічних протоколів

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Аналітичний огляд літератури. Постановка задачі.	01.10.17 - 25.10.17
Розробка структури системи моделювання.	26.10.17 - 14.11.17
Розробка пілотного проекту програми.	15.11.17 - 30.12.17
Економічний розділ. Оформлення роботи.	01.01.18 - 18.01.18

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект ***від розробки та реалізації запропонованої програмної системи – отримання прибутку від продажів системи. Користувачі системи можуть більш ефективно із меншими витратами проводити моделювання та верифікацію криптографічних протоколів.***

Соціальний ефект ***від результатів використання системи очікується позитивний. Програмна система має низький поріг входу, що дозволяє її використовувати студентам та людям, що не мають спеціальних знань. Вона може пришвидшити розуміння принципів побудови криптографічних протоколів.***

7 ДОДАТКОВІ ВИМОГИ

Відповідність оформлення «ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення» та «Методичні вказівки. Загальні вимоги до оформлення магістерських дипломних робіт і дипломних проектів спеціалістів для студентів галузей знань 1701 «Інформаційна безпека»

Завдання видав _____ Кагадій Т.С.
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ Сізинцев М.А.
(підпис) (прізвище, ініціали)

Дата видачі завдання: _____

Термін подання дипломної роботи до ДЕК _____

РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ додатків, ___ джерел.

Об’єкт дослідження: методи аналізу криптографічних протоколів.

Мета роботи (проекту): підвищення якості захисту криптографічних систем.

Методи дослідження: аналіз, системний підхід, прогнозування, синтез, класифікація, моделювання.

У спеціальній частині запропоновано вид уточненого опису протоколів, на його підставі розроблено концептуальну модель і структура програмної системи для моделювання протоколів.

У роботі проведено аналіз методів моделювання криптографічних протоколів, запропоновано структуру програмної системи моделювання цих протоколів і виконано пілотний проект системи.

В економічному розділі визначено економічну ефективність від розробки та реалізації кінцевому споживачу програмної системи.

Практичне значення роботи полягає у розробці архітектури та структури програмної системи моделювання криптографічних протоколів.

Результати здійснених у дипломній роботі (проекті) досліджень можуть бути використані у навчальних цілях та при аналізі певних класів криптографічних протоколів.

Наукова новизна дослідження полягає у розробці архітектури програмної системи моделювання.

Напрямки подальших досліджень - подальший аналіз криптографічних протоколів та доповнення програмної системи новими рішеннями.

Ключові слова: КРИПТОГРАФІЧНИЙ ПРОТОКОЛ, МОДЕЛЮВАННЯ, КЛАСИФІКАЦІЯ, УЧАСНИКИ ТА ЇХ РОЛІ, ВЛАСТИВОСТІ, АРХІТЕКТУРА СИСТЕМИ, МЕТОД КІНЦЕВИХ АВТОМАТІВ, МОДЕЛЬ ДОЛЕВА-ЯО.

РЕФЕРАТ

Пояснительная записка: ___ с., ___ рис., ___ табл., __ приложений, ___ источников.

Объект исследования: методы анализа криптографических протоколов.

Цель работы (проекта): повышение качества защиты криптографических систем.

Методы исследования: анализ, системный подход, прогнозирование, синтез, классификация, моделирование.

В специальной части предложено вид уточненного описания протоколов, на его основании разработана концептуальная модель и структура программной системы для моделирования протоколов.

В работе проведен анализ методов моделирования криптографических протоколов, предложена структура программной системы моделирования этих протоколов и выполнен пилотный проект системы.

В экономическом разделе определена экономическая эффективность от разработки и реализации конечному потребителю программной системы.

Практическое значение работы состоит в разработке архитектуры и структуры программной системы моделирования криптографических протоколов.

Результаты проведенных в дипломной работе (проекте) исследований могут быть использованы в учебных целях и при анализе определенных классов криптографических протоколов.

Научная новизна исследования заключается в разработке архитектуры программной системы моделирования.

Направления дальнейших исследований - дальнейший анализ криптографических протоколов и дополнение программной системы новыми решениями.

Ключевые слова: КРИПТОГРАФИЧЕСКИЙ ПРОТОКОЛ, МОДЕЛИРОВАНИЕ, КЛАССИФИКАЦИЯ, УЧАСТНИКИ И ИХ РОЛИ, СВОЙСТВА, АРХИТЕКТУРА СИСТЕМЫ, МЕТОД КОНЕЧНЫХ АВТОМАТОВ, МОДЕЛЬ ДЕЛЕВА-ЯО.

ABSTRACT

Explanatory note: ___ pages, ___ pictures, ___tables, __ applications, ___ sources.

Object of the study: methods of analyzing of cryptographic protocols.

The purpose of the work (project): improving the quality of protection of cryptographic systems.

Research methods: analysis, system approach, forecasting, synthesis, classification, modeling.

In the special part the form of the refined description of the protocols was proposed, on its basis the conceptual model and structure of the software system for protocol modeling was developed.

The work analyzes the methods for modeling cryptographic protocols, proposes the structure of the software system for modeling these protocols, and the pilot project of the system is performed.

In the economic section, the economic efficiency is determined from the development and implementation of the software system to the end user.

The practical significance of the work is to develop the architecture and structure of the software system for modeling cryptographic protocols.

The results of studies carried out in the thesis (project) can be used for educational purposes and for the analysis of certain classes of cryptographic protocols.

The scientific novelty of the study is to develop the architecture of a software modeling system.

The directions of further research are further analysis of cryptographic protocols and addition of the software system with new solutions.

Key words: CRYPTOGRAPHIC PROTOCOL, SIMULATION, CLASSIFICATION, PARTICIPANTS AND THEIR ROLES, PROPERTIES, SYSTEM ARCHITECTURE, METHOD OF FINITE AUTOMATES, DELOVA-YAO MODEL.

ЗМІСТ

с.

ВСТУП

Широкий розвиток технологій мережевого обміну інформацією в Інтернет, призвело до необхідності захисту інформації, що передається. Вирішення цієї задачі здійснюється криптографією – наукою про захист інформації. У криптосистемах є кілька учасників, що повністю довіряють один одному, яким необхідно передавати між собою інформацію, не призначену для інших осіб. Така інформація називається конфіденційної або секретної. Звідси виникає завдання забезпечення конфіденційності, тобто захист секретної інформації від противника.

При обміні інформацією між учасниками часто виникає ситуація, коли інформація не є конфіденційною, але важливий факт надходження повідомлень в неспотвореному вигляді, тобто наявність гарантії, що ніхто зможете не підробити повідомлення. Така гарантія називається забезпеченням цілісності інформації і також є одним із завдань криптографії.

Для запобігання загрози контролю за джерелами інформації необхідна система контролю за доступом до ресурсів, яка повинна відповідати двом вимогам. По-перше, кожен бажаючий повинен мати можливість звернутися до цієї системи анонімно, а по-друге, при цьому все-таки довести своє право на доступ до ресурсів. Ці властивості в криптографії називається невідстежністю. Забезпечення невідстежності теж завдання криптографії.

При побудові криптографічних систем необхідно вирішувати багато питань безпечного інформаційного обміну, наприклад:

- Чи дійсно повідомлення прийшло від конкретного відправника;
- чи не було воно підмінене або сфабриковано ким-небудь іншим;
- чи було повідомлення створено даними відправником, або він просто переслав чиєсь повідомлення;
- чи не є повідомлення (наприклад, про оплату будь-якого товару) повторно відправленим;

— і так далі.

Усі ці та подібні питання належать до галузі досліджень, що отримала назву протоколи забезпечення безпеки[1; 6].

Проблема побудови протоколів забезпечення безпеки при віддаленій мережевій взаємодії привела в свою чергу до створення нової дисципліни, що інтенсивно розвивається - криптографічні протоколи. Спочатку вони представляли собою надбудову над комунікаційними протоколами, що вводиться для забезпечення тих чи інших завдань безпеки. Але поступово вони інтегрувалися не тільки з комунікаційними протоколами, але і з іншими прикладними протоколами і стали їх безпосередньою частиною.

В даний час створено безліч криптографічних протоколів, призначених для вирішення самих різноманітних прикладних задач і для різних умов застосування.

Проблема побудови криптографічних протоколів нерозривно пов'язана з дослідженнями рівня їх безпеки, який можна забезпечити при їх використанні. Разом з пошуком слабких місць і доопрацюванням протоколів відбувалася формалізація самого поняття безпеки, що забезпечується даним протоколом. Для багатьох протоколів, що спочатку здавалися надійними, були знайдені вразливості та розроблені атаки, що їх використовують.

Навіть після більш ніж двадцятирічних досліджень з'являються теоретичні методи аналізу протоколів, на основі яких розробляється автоматизовані засоби аналізу безпеки протоколів. Дана обставина і визначає **актуальність теми дипломної роботи.**

У цій дипломній роботі викладені основні поняття, пов'язані з криптографічними протоколами, розглянуті основні властивості, що характеризують їх безпеку.

Слід зазначити що оскільки в цій дипломній роботі обговорюються саме криптографічні протоколи, передбачається, що криптографічні алгоритми та методи безпечні, і розглядаються тільки спроби злому цих протоколів.

У дипломній роботі виконано аналітичний огляд криптографічних протоколів, їх класифікацій, учасників, уявлення і властивостей безпеки. Описано атаки на криптопротоколи, методи їх аналізу і способи моделювання протоколів. Запропоновано вид уточненого опису протоколу, що відрізняється більш детальним описом шага і циклу протоколу. Запропоновано концептуальну модель і структуру програмної реалізації системи моделювання протоколів.

На прикладі відомого протоколу, що має вразливість виконаний прототип пілотної програми для реалізації цієї системи моделювання.

Метою роботи є підвищення якості захисту криптографічних систем.

Об'єктом дослідження є методи аналізу криптографічних протоколів.

Предметом дослідження є архітектура система моделювання криптографічних протоколів

Методи дослідження. Для вирішення поставленого завдання в даній роботі використовувалися методи криптографії, теорії кодування, теорії кінцевих автоматів, теорії компіляторів, поняття і методи об'єктно-орієнтованного програмування.

Достовірність і обґрунтованість результатів магістерської роботи забезпечується застосуванням коректних вихідних даних, апробованих методів, перевіркою несуперечності висновків і результатами аналізу пілотної програми.

Наукова новизна роботи визначається тим, що в ній запропонована:

1 Новий вид уточненого опису протоколу;
2 концептуальну модель системи моделювання;

3 структура програмної реалізації системи моделювання протоколів.

Практичне значення магістерської роботи полягає в тому, що розроблена структура і принцип програмної реалізації системи моделювання криптографічних протоколів, які дозволяють ефективно вирішувати верифікації цих протоколів. Розроблена система зможе застосовуватися фахівцями володіють специфічними знаннями і навичками в області інформаційної безпеки.

Особливий внесок здобувача. Автор самостійно сформулював мету, завдання дослідження, наукові положення і результати, виконав теоретичну і практичну частини роботи.

Структура і обсяг роботи. Робота складається з вступу, трьох розділів і висновків. Містить ___ сторінок друкованого тексту, в тому числі ___ сторінок тексту основної частини з ___ малюнками, списку використаних джерел з ___ найменуваннями на __ сторінках, ___ додатків на ___ сторінках з __ малюнками.

РОЗДІЛ 1
КРИПТОГРАФІЧНІ ПРОТОКОЛИ
1.1 Визначення протоколів

Існує кілька визначень криптографічних протоколів, але кожне з них посилається на визначення протоколу.

Протокол – це послідовність кроків, які роблять дві або більша кількість сторін для спільного вирішення деякої задачі. Всі кроки протоколу робляться в порядку суворої черговості, і жоден з них не може бути зроблений раніше, ніж закінчиться попередній. Згідно [1] протокол - це розподілений алгоритм вирішення деякої сукупності об'єктів і суб'єктів будь-якого завдання, кожен з яких досягає мети (вирішує завдання) з використанням приватних (розподілених) алгоритмів, причому при виконанні розподілених алгоритмів всі об'єкти і суб'єкти використовують однакову специфікацію даних і дій , процедури синхронізації і відновлення роботи після збоїв і ін.

Найпростіше визначення криптографічного протоколу дано в [13; 15] Криптографічний протокол – це протокол, який використовує криптографію. В [16], криптографічним протоколом називається протокол, в основі якого лежить криптографічний алгоритм.

Найбільш розгорнуте визначення криптографічного протоколу дано в [13]. Криптографічний протокол – протокол, призначений для виконання функцій криптографічного системи, в процесі виконання якого учасники використовують криптографічні алгоритми. Криптографічний система – це система, що забезпечує безпеку інформації криптографічними методами. Основними функціями криптографічної системи є забезпечення конфіденційності, цілісності, аутентифікації, неможливості відмови і включає підсистеми шифрування, ідентифікації, імітозахисту, цифрового підпису та ін., А також ключову систему.

Криптографічні протоколи повинні мати наступні властивості:

- Кожен учасник протоколу повинен знати протокол і всі необхідні кроки наперед;
- Кожен учасник повинен бути згоден йому слідувати;
- Протокол повинен бути чітко і однозначно визначено;
- Протокол повинен описувати реакцію учасників на будь-які ситуації, які можуть виникнути в ході його реалізації;
- Протокол повинен, бути повним, тобто приводити до вирішення поставленого завдання.

Типовий протокол передбачає від двох до чотирьох учасників. Один з них, наприклад, A, є ініціатором запуску протоколу. На підставі інформації, наявної у нього, він генерує інформацію для передачі, виконує над нею деяку послідовність дій, формує повідомлення і передає його учаснику B або C в залежності від даного протоколу. Користувач B (або C), отримавши повідомлення від A, виконує над ним відповідну послідовність дій, реєструє в своїй пам'яті виділену інформацію. На цьому завершується 1-ий цикл протоколу, а роль ініціатора передається учаснику B (або C).

Аналогічно, користувач B на підставі інформації, наявної тепер в його пам'яті, генерує інформацію для передачі, формує повідомлення і передає його A чи C в залежності від даного протоколу. Після прийому повідомлення, його обробки та реєстрації виділеної інформації в пам'яті одержувача завершується 2-ий цикл протоколу, і т.д.

Кожен учасник по протоколу має свою мету, що виражається в отриманні певної інформації.

Протокол завершується, коли кожен з учасників досягає своєї мети. В іншому випадку протокол обривається.

Крім учасників протоколу, які чесно його виконують, в протоколі можуть брати участь учасники-порушники (противники) . Вони можуть бути активними і пасивними.

Пасивний супротивник тільки перехоплює всі повідомлення в каналі зв'язку, намагаючись витягти з них максимум інформації, але не втручаючись в протокол. Такий противник є неявним учасником протоколу, стан якого також має враховуватися і аналізуватися з точки зору безпеки протоколу.

Якщо ж противник активний, то він також стає несанкціонованим учасником протоколу, прихованим для санкціонованих учасників. Такий противник не зобов'язаний дотримуватися протоколу. Він повинен тільки підтримувати видимість нормального виконання протоколу. Активний противник в протоколі може поперемінно грати роль чесних (зареєстрованих) учасників протоколу. Він може підставляти замість повідомлень, переданих санкціонованими учасниками, повідомлення, передані в попередніх запусках протоколу, в поточному запуску, або, він може ініціювати від імені чесних учасників протоколу новий запуск протоколу до закінчення поточного і скористатися повідомленнями цього паралельного протоколу.

Ще більш сильним противником є такий, який володіє ключем (ключами), чинним або виведеним з дії, причому учасники протоколу про це можуть не знати, принаймні, в протягом деякого часу.

Розрізняють примітивні і прикладні криптографічні протоколи.

Примітивний криптографічний протокол – це криптографічний протокол, який не має самостійного прикладного значення, але використовується як базовий компонент при побудові прикладних криптографічних протоколів. Як правило, він вирішує будь-яку одну абстрактну задачу. Приклади: протокол обміну секретами, протокол прив'язки до біту, протокол підкидання монети (по телефону).

Прикладний криптографічний протокол призначений для вирішення практичних завдань забезпечення функцій - сервісів безпеки за допомогою криптографічних систем. Прикладні протоколи, як правило, забезпечують не одну, а відразу кілька функцій безпеки. Більш того, такі протоколи насправді є великими сімействами різних протоколів, що включають багато різних варіантів для різних ситуацій і умов застосування. Прикладами прикладних протоколів є: система електронного обміну даними, протоколи електронного документообігу; система електронних платежів, протокол підписання контракту, протокол сертифікованої електронної пошти та багато інших.

Щоб опис протоколів було більш наочним, їх учасники носять імена, які однозначно визначають ролі, їм уготовані. Розрізняють:

— Протоколи з арбітражем (адвокатом);
— Протокол із суддівством;
— Самостверджувальні протоколи.

Самостверджувальні протоколи не вимагають присутності арбітра для завершення кожного кроку протоколу або наявності судді для вирішення конфліктних ситуацій. Самостверджуються протокол влаштований так, що якщо один з його учасників шахраювати, інші зможуть моментально розпізнати нечесність, виявлену цим учасником, і припинити виконання наступних кроків протоколу.

У протоколах з арбітром, він є незацікавленим учасником протоколу, якому інші учасники повністю довіряють, роблячи відповідні дії для завершення чергового кроку протоколу. Учасники протоколу також беруть на віру все, що скаже арбітр, і беззаперечно виконують всім його рекомендаціям.

В силу того, що всі учасники протоколу повинні користуватися послугами одного й того ж арбітра, дії зловмисника, який вирішить завдати їм шкоди, будуть спрямовані, в першу чергу, проти цього арбітра. Отже, арбітр є

слабкою ланкою в ланцюгу учасників будь-якого протоколу з арбітражем і тому протокол, в якому бере участь арбітр, часто ділиться на дві частини.

Перша повністю збігається зі звичайним протоколом без арбітражу, а до другої вдаються лише в разі виникнення розбіжностей між учасниками. Для вирішення конфліктів між ними використовується особливий тип арбітра – суддя. Подібно арбітру, суддя є незацікавленим учасником протоколу, якому інші його учасники довіряють при прийнятті рішень. Однак на відміну від арбітра, суддя бере участь аж ніяк не в кожному кроці протоколу. Послугами судді користуються, тільки якщо потрібно дозволити сумніви щодо правильності дій учасників протоколу. Якщо таких сумнівів ні у кого не виникає, суддівство не знадобиться. У комп'ютерних протоколах із суддівством передбачається наявність даних, перевіривши які довірена третя особа може вирішити, чи не зшахраював хто-небудь з учасників цього протоколу. Хороший протокол із суддівством також дозволяє з'ясувати, хто саме веде себе нечесно. Це служить прекрасним превентивним засобом проти шахрайства з боку учасників такого протоколу.

1.2 Класифікація протоколів

Зусиллями криптологів в різний час було створено велику кількість прикладних криптографічних протоколів [1; 13], які умовно можуть бути класифіковані за такими ознаками:

1 Класифікація за кількістю учасників: двосторонній, тристоронній і т. п.;
2 Класифікація за кількістю переданих повідомлень:
   — Інтерактивний (є взаємний обмін повідомленнями);
   — Не інтерактивний (тільки одноразова передача). Не інтерактивні протоколи часто називають схемами;
3 Класифікація за цільовим призначенням протоколу:

— Протокол забезпечення цілісності повідомлень (з аутентифікацією джерела, без аутентифікації джерела);
— Протокол (схема) цифрового підпису (протокол індивідуальної / групової цифрового підпису, з відновленням / без відновлення повідомлення, протокол цифрового підпису наосліп, протокол конфіденційної цифрового підпису, протокол цифрового підпису з доказовою підробки);
— Протокол ідентифікації (аутентифікації) учасників (односторонньої аутентифікації, двосторонньої (взаємної) аутентифікації);
— Протокол конфіденційної передачі (звичайний обмін повідомленнями, широкомовна / циркулярна передача, чесний обмін секретами, забуває передача;
— Протокол розподілу ключів (схема попереднього розподілу ключів, передачі ключа (обміну ключами), спільного вироблення ключа (відкритого розподілу ключів), протокол парний / груповий, протокол (схема) поділу секрету, протокол (розподілу ключів для) телеконференції, і ін);

Класифікацію криптографічних протоколів можна проводити також і по іншим ознаками:

1 За типом використовуваних криптографічних систем:
— На основі симетричних криптосистем;
— На основі асиметричних криптосистем;
— Змішані.
2 За способом функціонування:
— Інтерактивний / не інтерактивний;
— Однопрохідний / дво- / три- і т.д. Прохідний;
— Протокол з арбітром (протокол з посередником),
— Двосторонній / с довіреною третьою стороною (з центром довіри), і т.п.

— За складом і розподілу ролей;

— Протоколи з арбітражем (адвокатом);

— Протокол із суддівством;

— Самоствердкуються протокол.

Будь-який криптографічний протокол після вдалого злому перестає бути надійним. Такі ситуації називають провалами протоколів. Існують приклади протоколів [2; 9], які можуть бути абсолютно ненадійними і без всякого злому. Наприклад, протоколи, що розсилають ключі не за призначенням, протоколи з цифровим підписом, що легко підробити і т.д. Тому протоколи класифікують ще й за надійністю:

— Імовірність злому протоколу не залежить від обсягу ресурсів і часу у зломщика;

— Для зламу протоколу зловмисникові необхідно вирішити деяку математичну задачу, теоретично вирішувану, але для якої всі відомі на сьогодні у день методи вирішення потребують нездійсненно великого обсягу обчислень. Більшість відомих криптографічний протоколів належать до цього класу.

Здійснення зламу протоколу рівносильно, за обсягом необхідних зусиль і витрат, вирішенню якогось математично складного завдання (наприклад, підрахунок факторіала великого цілого).

— Всі інші

1.3 Опис протоколів

У сучасній літературі з криптографії найчастіше протоколи описуються в вербальному вигляді, що супроводжується пояснювальними діаграмами і / або символьним описом дій, які виконуються на кожному кроці протоколу і уточнююче специфікацію протоколу.

Загальновизнаного виду інтерфейсу протоколу не існує. Нижче наведені найбільш часто використовувані опису криптографічних протоколів:

1 Вербальний опис. Наприклад:
    1 Аліса і Боб вибирають систему шифрування.
    2 Аліса і Боб вибирають ключ.
    3 Аліса шифрує відкритий текст свого повідомлення з використанням алгоритму шифрування і ключа, отримуючи зашифроване повідомлення.
    4 Аліса посилає зашифроване повідомлення Бобу.
    5 Боб дешифрує шифротекст повідомлення з використанням алгоритму дешифрування і ключа, отримуючи відкритий текст повідомлення.
2 Математичний опис виконуваних операцій з вербальним описом дій учасників. Наприклад:

    Обчислюється значення хеш-функції від повідомлення h (w).

    Далі учасник, що підписує, вибирає випадкове або псевдовипадкове значення k, 0 <k <q, обчислює k-1 (mod q), і генерує пару значень:

    r = gk (mod p) (mod q);

    s = к-1 (h (m) + xr) (mod q)

    Ця пара значень (r, s) і є електронним підписом під повідомленням M. Після створення цифрового підпису значення k знищується.
3 Описом по кроках протоколу. Приклад опису в таблиці 1.1.

Таблиця 1.1 - Протокол конфіденційного обміну

Цикл	Крок	Опис кроку
1	1	А формує текстову послідовність M1

Продовження таблиці 1.1

	2	A обчислює S1 = E_{kAB}(M1).
	3	A відправляє учаснику B повідомлення S1
	4	B отримує повідомлення S1 і із заголовку дізнається ідентифікатор відправника A
	5	B обчислює текст M1 = D_{kAB}(S1).
2	1	B формує текстову послідовність M2
	2	B обчислює S2 = E_{kBA}(M2).
	3	B відправляє учаснику A повідомлення S2
	4	A отримує повідомлення S2 S1 і із заголовку дізнається ідентифікатор відправника B.
	5	A обчислює текст M2 = D_{kBA}(S2).

4 Символічний опис. Приклад на рисунку 1.1.

$$(1)\ A \rightarrow B:\ E_{k_{AB}}(M_1),$$
$$(2)\ A \leftarrow B:\ E_{k_{BA}}(M_2).$$

Рисунок 1.1 – Приклад символічного опису протоколу

5 У вигляді відображення послідовності дій. Приклад на рисунку 1.2.

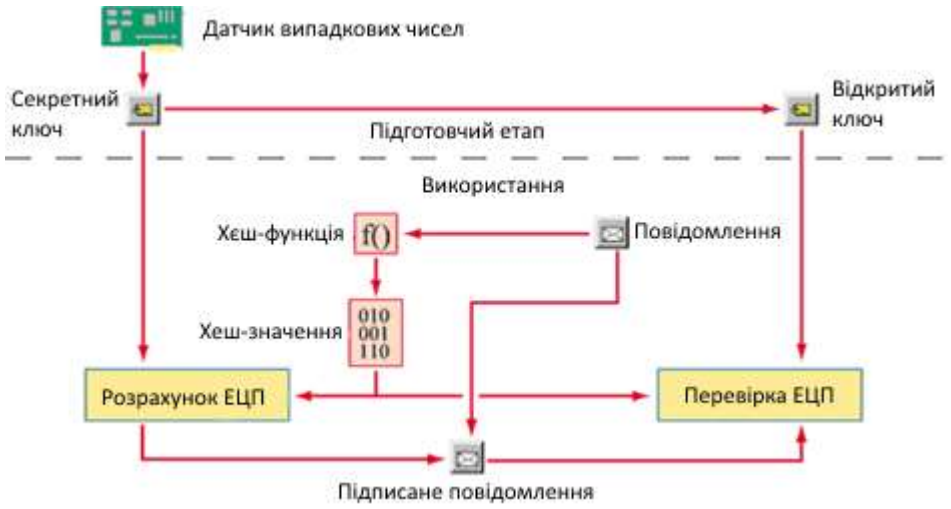

Рисунок 1.2 – Ілюстрація «Відображення послідовності дій»

1.4 Властивості, що визначають безпеку протоколів

Оскільки криптографічний система може забезпечувати різні функції безпеки, для реалізації яких застосовують різноманітні криптографічні протоколи, то і властивостей, які характеризують безпеку криптографічного протоколу, також досить багато. Зазвичай властивості протоколів, що характеризують їх стійкість до різних атак, формулюють як цілі (goals) або вимоги до протоколів. Трактування цих цілей з часом змінюється і уточнюється. Найбільш повне і сучасне тлумачення цих цілей дається в документах міжнародної організації Internet Engineering Task Force (IETF).

У документах IETF фігурують двадцять властивостей (цілей, вимог) безпеки, розподілених по десяти групам (див таб.1.2). Наведемо ці властивості.

Таблиця 1.2 – Групи властивостей безпеки криптографічних протоколів

Група безпеки	Властивість (ціль, вимоги)
Аутентифікація(не широкомовна)	G1 – аутентифікація суб'єкта
	G2 – аутентифікація повідомлення
	G3 – захист від повтору

Продовження таблиці 1.2.

Аутентифікація при розсилці за багатьма адресами або встановлення з'єднання зі службою підписки / повідомлення	G4 – неявна (прихована) аутентифікація одержувача
	G5 – аутентифікація джерела
Авторизація (довіреною третьою стороною)	G6 – авторизація (довіреною третьою стороною)
Властивості спільної генерації ключа	G7 – аутентифікація ключа
	G8 – підтвердження вірності ключа
	G9 – захищеність від читання назад
	G10 – формування нових ключів
	G11 – захищена можливість домовитися про параметри безпеки
Конфіденційність	G12 – конфіденційність
Анонімність	G13 – захист ідентифікаторів від прослуховування
	G14 – захист ідентифікаторів від інших учасників
Захищеність від атак типу «відмова в обслуговуванні»	G15 – обмежена захищеність від атак типу «відмова в обслуговуванні»
Інваріантність відправника	G16 – інваріантність відправника
Неможливість відмови від раніше вчинених дій	G17 – підзвітність
	G18 – доказ джерела
	G19 – доказ одержувача
Безпечна тимчасова властивість	G20 – безпечна тимчасова властивість

1 Аутентифікація (не широкомовна) – перевірка ідентичності, заявленої учасником або суб'єктом системи, в якості якого може виступати одна зі сторін комунікації або джерело деяких даних. Аутентифікація

зазвичай є односторонньою. Взаємна аутентифікація здійснюється в обох напрямках.

1 G1 – аутентифікація суб'єкта (аутентифікація сторін. Це перевірка з підтвердженням справжності однієї зі сторін або наявності повноважень (за допомогою наданих доказів і (або) документів) ідентичності другої сторони, що бере участь у виконанні протоколу, а також того, що вона дійсно бере участь у виконанні протоколу , а також того, що вона дійсно бере участь у виконанні поточного сеансу протоколу. Зазвичай така перевірка здійснюється за допомогою набору даних, який міг бути згенерований тільки втори м учасником (наприклад, відгук на запит). Таким чином, зазвичай аутентифікація суб'єкта має на увазі, що деякі дані можуть бути безпомилково повернуті деякого суб'єкту, що передбачає аутентифікацією джерела даних (англ. data origin authentication);

2 G2 – аутентифікація повідомлення. Полягає в забезпеченні аутентифікації джерела даних і цілісності переданого повідомлення. Аутентифікація джерела даних означає, що протокол повинен забезпечувати засоби гарантії того, що отримане повідомлення або частина даних були створені деякими учасником в певний момент часу, що передує одержанню повідомлення, і що ці дані не були спотворені або підроблені, але без надання гарантій однозначності і своєчасності;

3 G3 – захист від повтору. Це гарантування одним з учасників того, що аутентифіковане повідомлення не є старим (було згенеровано в даному сеансі протоколу, або протягом відомого проміжку часу, або повідомлення не було прийнято раніше.

2 Аутентифікація при розсилці за багатьма адресами або встановлення з'єднання зі службою підписки / повідомлення – це вимога

аутентифікації для груп учасників з одним джерелом і великим числом потенційних одержувачів (широкомовна передача) або джерело і служба, які відправляють інформацію підключеним і авторизованим користувачам.

1 G4 – неявна (прихована) аутентифікація одержувача. Протокол повинен надавати кошти гарантії того, що відправлене повідомлення буде прочитано тільки тими сторонами, яким воно було адресоване, тобто тільки законні авторизовані учасники отримають доступ до поточної інформації, широкомовним повідомленням або групової комунікації (передачі);
2 G5 – аутентифікація джерела. Законні групи учасників повинні бути здатні аутентифікувати джерело і зміст інформації або групової комунікації. Це стосується випадків, коли групи учасників не довіряють один одному.

3 Авторизація довіреною третьою стороною.

1 G6 – полягає в тому, що в деяких протоколах довірена третя сторона представляє одного суб'єкта В іншому суб'єкту А. В результаті цього суб'єкт А отримує гарантії того, що суб'єкт В посвідчений за допомогою довіреної третьої сторони і авторизований (наділений правами) в необхідному для протоколу сенсі . Коли протокол виконується трьома учасниками А, В, довірена третя сторона, то суб'єкт А, ймовірно, не може мати доступ до контрольного списку або іншим механізмам для авторизації В (бо ім'я В невідомо суб'єкту А або може бути псевдонім), але суб'єкт А отримує гарантії того, що суб'єкт В авторизований довіреною третьою стороною.

4 Властивості спільної генерації ключа.

1 G7 – аутентифікація ключа. Це властивість передбачає, що один з учасників отримує підтвердження того, що ніякий інший учасник крім заздалегідь визначеного другого учасника (і,

можливо, інших довірених учасників) не може отримати доступ до жодного секретного ключа;

2. G8 – підтвердження вірності ключа. Один з учасників отримує підтвердження того, що другий учасник (можливо, невизначений) дійсно володіє конкретним секретним ключем (або має доступ до всіх ключових матеріалів, необхідних для його обчислення);
3. G9 – захищеність від читання назад / досконала таємність в майбутньому. Протокол володіє цією властивістю, якщо компрометація довгострокових ключів не призводить до компрометації старих сеансових ключів;
4. G10 – формування нових ключів. Протокол використовує динамічний розподіл ключів з метою отримання нових ключів;
5. G11 – захищена можливість домовитися про параметри безпеки. Якщо протокол відкритого розподілу ключів дає можливість сторонам домовлятися про параметри безпеки (таких як ідентифікатори захищеної асоціації, довжина ключа і набори алгоритмів шифрування), то ця властивість важлива для підтвердження того, що заявлені властивості і параметри, про які домовилися учасники протоколів, не були підмінені зловмисником.

5 Конфіденційність.

1. G12 – конфіденційність полягає в тому, що специфічний набір даних (зазвичай посилається або отримується як частина "захищеного" повідомлення, а також сформований на основі даних, отриманих в результаті обміну) не стане доступним або розкритим для неавторизованих суб'єктів або процесів, а залишиться невідомим противнику. Секретність сеансового ключа, згенерованого в результаті процедури відкритого розподілу ключів, розглядається при описі властивості G7.

Секретність довготривалого ключа, використовуваного в протоколі, не розглядається як цільове властивість безпеки протоколу, а відноситься до вихідних припущеннями.

6 Анонімність. Багато протоколів не забезпечують властивості анонімності, оскільки, як правило, сторона хоче знати, з ким вона взаємодіє при формуванні ключа. Однак деякі протоколи дозволяють приховувати ідентифікатори.

    1 G13 – захист ідентифікаторів від прослуховування. Атакуючий, який здійснює перехоплення повідомлень; не повинен мати можливість зв'язати повідомлення одного з учасників з самим учасником;

    2 G14 – захист ідентифікаторів від інших учасників. В процесі взаємодії інші учасники не повинні мати можливості зв'язати повідомлення конкретного учасника з самим учасником, а тільки з непов'язаним з ним псевдонімом або приватним ідентифікатором.

7 Обмежена захищеність від атак типу "Відмова в обслуговуванні".

    1 G15 – обмежена захищеність від атак типу "відмова в обслуговуванні" полягає в тому, що важко забезпечити захищеність від DoS-атак. Протокол може бути об'єктом DoS-атак з різних причин; найпоширеніша полягає в тому, що протокол споживає занадто багато ресурсів (пам'яті, обчислювальної потужності), перед тим як сторони аутентифікують один одного.

8 Інваріантність відправника.

    1 G16 – інваріантність відправника полягає в тому, що учасник отримує гарантії того, що джерело повідомлень не змінювалось з початку виконання сеансу, хоча саме встановлення ідентичності цього джерела для одержувача не важливо.

9 Неможливість відмови від раніше вчинених дій передбачає запобігання того, що учасник відмовиться від реально вчиненого ним дії.

1 G17 – підзвітність. Ця властивість системи, складається у тому, що надаються гарантії того, що дії системних суб'єктів можуть бути однозначно простежено тими суб'єктами, хто відповідає за ці дії;

2 G18 – доказ джерела. Це безперечний доказ (очевидність) того, що повідомлення було відправлено;

3 G19 – доказ отримання. Це безперечний доказ (очевидність) того, що повідомлення було отримано.

10 Безпечна тимчасова властивість.

1 G20 – безпечне тимчасове властивість полягає в тому, що використовуючи два оператора часової логіки (лінійної часової логіки), а саме оператора "завжди" і "колись в минулому", можна формалізувати властивості виду: "Для будь-якого досяжного стану, що має властивість p , раніше був стан з властивістю q".

У деяких документах IETF обговорюються нові властивості безпеки.

Формування сеансу. Протокол забезпечує формування сеансу, якщо він пов'язує кожен конкретний сеанс протоколів з унікальним значенням, узагальненим ідентифікатором сеансу, який зазвичай є набором випадкових даних і ідентифікаторів. Цей узагальнений ідентифікатор сеансу дозволяє відрізнити кожен конкретний сеанс протоколу від інших сеансів цього протоколу. Після виконання початкової частини протоколу всі повідомлення містять узагальнений ідентифікатор сеансу (не обов'язково у відкритому вигляді).

Послідовне уявлення. Після виконання будь-якої частини протоколу всі учасники даного сеансу протоколу мають однакове уявлення про всіх учасників цього сеансу і виконуваних ними ролях, а також про стан виконання протоколу.

Іменування ключів. Для того щоб мати гарантії від неправильного використання ключових матеріалів у кожній захищеній асоціації протоколу обміну, криптографічний протокол повинен використати наявні імена ключів і відповідний контекст, що дозволяє інформувати перевіряючого про порядок використання цього ключового матеріалу. Якщо протокол забезпечує доказ володіння ключами, то протокол повинен застосовувати явні імена ключів, що використовуються протягом докази володіння, щоб запобігти ситуації, коли використовується більше одного набору матеріалів для виконання протоколу обміну.

Є і інші властивості, обговорювані в документах IETF, наприклад: криптографічний поділ ключів, можливість домовитися про вибір алгоритмів шифрування, стійкість до атаки зі словником, криптографічне зв'язування, підтримка швидкого відновлення з'єднання, підтвердження успішного завершення і повідомлень про помилку, незалежність сеансів, захищеність від атаки противник в середині і ін.

1.5 Атаки на протоколи

Під атакою на протокол розуміється спроба проведення аналізу повідомлень протоколу і / або виконання не передбачених протоколом дій з метою порушення роботи протоколу і / або отримання інформації, що становить секрет його учасників. Атака вважається успішною, якщо порушено хоча б один з номінальних параметрів, що характеризують безпеку протоколу (див. табл. 1.2).

В основі атак можуть лежати різні методи аналізу протоколів.

Атаки на протоколи бувають спрямовані проти криптографічних алгоритмів, які в них задіяні, проти криптографічних методів, що застосовуються для їх реалізації, а також проти самих протоколів.

Ці атаки умовно можна розділити на наступні групи:

- Пасивні (підслуховування протоколу з метою отримання інформації);
- Активні (спроби змінити протокол – вставити, змінити повідомлення);
- Пасивні шахраї – одна зі сторін намагається отримати більше інформації, не перестаючи при цьому дотримуватися протоколу;
- Активні шахраї – одна зі сторін, намагається змінити протокол.

Особа, яка не є учасником протоколу, може спробувати підслухати інформацію, якою обмінюються його учасники. Це пасивна атака на протокол, яка названа так тому, що атакуючий може тільки накопичувати дані і спостерігати за ходом подій, але не в змозі впливати на нього.

Пасивна атака подібна крипто-аналітичній атаці зі знанням тільки шифротекста. Оскільки учасники протоколу не володіють надійними засобами, що дозволяють їм визначити, що вони стали об'єктом пасивної атаки, для захисту від неї використовуються протоколи, що дають можливість запобігати можливі несприятливі наслідки пасивної атаки, а не розпізнавати її.

Атакуючий може спробувати внести зміни до протоколу заради власної вигоди. Він може видати себе за учасника протоколу, внести зміни в повідомлення, якими обмінюються учасники протоколу, підмінити інформацію, яка зберігається в комп'ютері і використовується учасниками протоколу для прийняття рішень. Це активна атака на протокол, оскільки атакуючий може втручатися в процес виконання кроків протоколу його учасниками. Отже, пасивно атакуючий намагається зібрати максимум інформації про учасників протоколу і про їхні дії.

У активно атакуючого ж зовсім інші інтереси – погіршення продуктивності комп'ютерної мережі, отримання несанкціонованого доступу до її ресурсів, внесення спотворень в бази даних. При цьому всі атакуючі не

обов'язково є абсолютно сторонніми особами. Серед них можуть бути легальні користувачі, системні і мережеві адміністратори, розробники програмного забезпечення і навіть учасники протоколу, які поводяться непорядно або навіть зовсім не дотримуються цей протокол. В останньому випадку атакуючий учасник протоколу називається шахраєм.

Пасивний шахрай слідує усім правилам, які визначені протоколом, але при цьому ще й намагається дізнатися про інших учасників більше, ніж передбачено цим протоколом.

Активний шахрай вносить довільні зміни в протокол, щоб нечесним шляхом добитися для себе найбільшої вигоди.

Захист протоколу від дій декількох активних шахраїв є вельми нетривіальною проблемою. Проте за деяких умов цю проблему вдається вирішити, надавши учасникам протоколу можливість вчасно розпізнати ознаки активного шахрайства. А захист від пасивного шахрайства повинен надавати будь-який протокол незалежно від умов, в які поставлені його учасники.

Найбільш широко відомі атаки на криптографічні протоколи:

- Підміна – спроба підмінити одного користувача іншим. Порушник, виступаючи від імені однієї зі сторін і повністю імітуючи її дії, отримує у відповідь повідомлення певного формату, необхідні для підробки окремих кроків протоколу;
- Повторне нав'язування повідомлення – повторне використання раніше переданого в поточному або попередньому сеансі повідомлення або будь-якої його частини в поточному сеансі протоколу. Наприклад, повторна передача інформації раніше проведеного протоколу ідентифікації може привести до повторної успішної ідентифікації того ж самого або іншого користувача. У протоколах передачі ключів ця атака часто застосовується для

повторного нав'язування вже використаного раніше сеансового ключа - атака на основі новизни;

- Атака відображенням. Цей тип атак пов'язаний зі зворотним передачею адресату раніше переданих їм повідомлень. Часто атаки даного типу відносять до класу атак з повторним нав'язуванням повідомлення;
- Затримка передачі повідомлення – перехоплення противником повідомлення і нав'язування його в більш пізній момент часу. Це різновид атаки з повторним нав'язуванням повідомлення;
- Комбінована атака – підміна або інший метод обману, який використовує комбінацію даних з раніше виконаних протоколів, в тому числі протоколів, раніше нав'язаних супротивником;
- Атака з паралельними сеансами. Спеціальний окремий випадок попередньої атаки, в якому противник спеціально відкриває одночасно кілька паралельних сеансів з метою використання повідомлень з одного сеансу в іншому;
- Атака з використанням спеціально підібраних текстів - атака на протоколи типу «запит - відповідь» при якій противник за певним правилом вибирає запити з метою отримати інформацію про довготривале ключі доводить. Ця атака може включати спеціально підібрані відкриті тексти, якщо доводить повинен підписати або зашифрувати запит або шифровані тексти, якщо доводить повинен розшифрувати запит;
- Атака «противник в середині». Використовується противником своїх коштів в якості частини телекомунікаційної структури. Атака, при якій в протоколі ідентифікації між А і В противник входить в телекомунікаційний канал і стає його частиною при реалізації протоколу між А і В. При цьому противник може підмінити інформацію, передану між А і В. Ця атака особливо небезпечна в

разі формування учасниками A і B загального ключа по протоколу Діффі – Хеллмана;

— Атака з відомим сеансовим ключем. У багатьох випадках проведення атаки полегшує додаткова інформація. Атака полягає у спробі отримання інформації про довготривалі ключі або будь-якої іншої ключової інформації, що дозволяє відновлювати сеансові ключі для інших сеансів протоколу;

— Атака з невідомим спільним ключем – атака, при якій порушник відкриває два сеанси з двома зареєстрованими учасниками, виступаючи в першому випадку від імені одного з них, хоча останній може нічого не знати про це. При цьому в результаті буде сформований загальний ключ між чесними учасниками, причому один з них буде впевнений, що сформував загальний ключ іншим чесним учасником;

Існує велика кількість і інших типів атак, які залежать від конкретної реалізації протоколу.

1.6 Аналіз і моделювання протоколів

З точки зору додатків аналіз протоколів має дуже велике значення, тому що саме за допомогою протоколів принципи криптографії знаходять практичне застосування. Самі алгоритми шифрування безумовно також значимі, але є лише частиною протоколу. І надійність алгоритму аж ніяк не забезпечує безпеку протоколу, що його використовує. Криптографічний протокол забезпечує досягнення певних цілей безпеки. Однак, якщо протокол містить помилки, то він може не досягти або досягти не в повній мірі усіх поставлених перед ним цілей. Помилки в протоколі можуть бути неявними і важко виявляються. Існує безліч прикладів, коли помилки виявлялися в уже добре вивчених протоколах.

Багато системи "втрачають гарантію" безпеки, якщо використовуються неправильно. Так наприклад алгоритми шифрування необов'язково забезпечують цілісність даних, а протоколи обміну ключами необов'язково гарантують, що обидві сторони отримають той самий ключ. Ще одне можливе слабке місце - взаємодія між окремо безпечними протоколами шифрування [15].

Майже для кожного безпечного протоколу, як правило, можна знайти інший, не менш надійний, який зведе нанівець всі переваги першого, якщо вони обидва використовують однакові ключі на одному і тому ж пристрої.

Якщо різні стандарти захисту застосовуються в одному середовищі, недостатньо чітка взаємодія між ними часто може привести до небажаних наслідків. Багато цікавих способів подолання захисних рубежів пов'язані з моделями довірчих відносин всередині системи. Прості системи (засоби шифрування телефонних переговорів та інформації на жорстких дисках) використовують елементарні довірчі моделі. Комплексні системи (засоби електронної торгівлі або засоби захисту багатокористувацьких пакетів електронної пошти) побудовані на основі складних (і набагато більш надійних) моделей довірчих відносин, що описують взаємозв'язку безлічі елементів.

Аналіз протоколу зводиться до ретельної формальної перевірки всіх можливих ситуацій.

Нижче наведені три загальних правила, яким слідують при аналізі криптографічних протоколів [8; 11]:

1 Для всіх величин, які використовуються в протоколі, слід перерахувати всі їх властивості - як явно зазначені в специфікації протоколу, так і неявно передбачувані;
2 Слід вивчати поведінку протоколу при всіх можливих відхиленнях неявно заданих властивостей параметрів. Треба розглядати

поведінку протоколу при зміні не тільки окремих параметрів, але їх комбінацій;

3 Якщо з'ясується, що на хід протоколу можна вплинути зміною тих чи інших параметрів і тому слід з'ясувати наскільки серйозними будуть наслідки такої зміни.

Існує чотири основні підходи до аналізу криптографічних протоколів [3; 10; 13]:

1 Моделювання і перевірка роботи протоколу з використанням мов опису і засобів перевірки не розроблених спеціально для аналізу криптографічних протоколів;
2 Створення експертних систем, що дозволяють конструктору протоколу розробляти і досліджувати різні сценарії;
3 Вироблення вимог до сімейства протоколів, використовуючи якусь логіку для аналізу понять "знання" і "довіру";
4 Розробка формальних методів, заснованих на записи властивостей криптографічних систем в алгебраїчному вигляді.

Перший з підходів намагається довести правильність протоколу, розглядаючи його як звичайну комп'ютерну програму. Ряд дослідників представляють протокол як кінцевий автомат, інші використовують розширення методів обчислення предиката першого порядку, а треті для аналізу протоколів використовують мови опису. Однак, доказ правильності аж ніяк не є доказом безпеки, і цей підхід зазнав невдачі при аналізі багатьох "дірявих" протоколів. І хоча його використання спочатку широко вивчалося, з ростом популярності третього і четвертого підходів, роботи в цій галузі були переорієнтовані.

У другому підході для визначення того, чи може протокол перейти в небажаний стан (наприклад, втрата ключа), використовуються експертні системи. Хоча цей підхід дає кращі результати при пошуку "дірок", він не

гарантує безпеки і не надає методик розробки розтинів. Він хороший для перевірки того, чи містить протокол конкретну "діру", але навряд чи здатний виявити невідомі "дірки" в протоколі.

Третій підхід набагато популярніше. Він був вперше введений Майклом Берроуз, Мартіном Абеді і Роджером Неедхемом. Вони розробили формальну логічну модель для аналізу знання і довіри, названу БАН-логікою. БАН-логіка є найбільш широко розповсюдженою при аналізі протоколів перевірки автентичності. Вона розглядає справжність як функцію від цілісності і новизни, використовуючи логічні правила для відстеження стану цих атрибутів протягом усього протоколу. БАН-логіка не надає доказ безпеки, вона може тільки розмірковувати про перевірку достовірності. Вона є простою, прямолінійною логікою, легкою в застосуванні і корисною при пошуку "дірок" в протоколах.

Методи четвертого, формального підходу, з одного боку досить змістовні і дозволяють легко виконувати моделювання та аналіз протоколів; з іншого боку, вони досить складні і дозволяють виявляти складні для розуміння помилки, не виявлені при неформальному аналізі. Формальні методи протягом тривалого часу використовувалися для аналізу комунікаційних протоколів. Деякі роботи з аналізу криптографічних протоколів велися в кінці сімдесятих і на початку вісімдесятих років [7; 16]. Але в цілому, застосування формальних методів до криптографічних протоколів не було широко поширене до початку дев'яностих, коли при використанні методів формального аналізу були знайдені невиявлені раніше помилки в криптографічних протоколах.

Більшість формальних методів, засновані на теорії кінцевих автоматах і використовують модель зловмисника, запропоновану Долевим і Яо [5; 13]. У моделі Долева-Яо всі активні учасники протоколу поділяються на два види: чесні учасники і зловмисник. Чесні учасники виконують кроки протоколу без відхилень. Вони можуть одночасно виконувати кілька сеансів протоколу з різними учасниками. Модель містить повідомлення, якими обмінюються учасники протоколу, але не описує внутрішні стану учасників. У моделі Долева

і Яо робиться припущення, що середовище передачі контролюється зловмисником, який може читати весь трафік, змінювати і видаляти повідомлення, створювати нові повідомлення, і виконувати будь-які операції які можуть виконувати законні користувачі системи. Передбачається, що спочатку зловмисник не знає ніякої секретної інформації, наприклад секретних ключів належать легітимним користувачам системи. Оскільки зловмисник може видаляти повідомлення з каналу зв'язку і поміщати в канал зв'язку створені ним повідомлення, то можна розглянути будь-яке повідомлення, надіслане легітимним користувачем, як повідомлення, надіслане зловмисникові, і будь-яке повідомлення, отримане легітимним користувачем, як повідомлення, отримане від зловмисника. Таким чином, система стає автоматом, використовуваним зловмисником для генерації слів. Ці слова підкоряються певним правилам підстановки, наприклад таким, що шифрування і розшифрування на одному ключі. Таким чином, зловмисник управляє системою з підстановкою елементів. Якщо мета зловмисника полягає в тому, щоб дізнатися секретне слово, то проблема докази безпеки протоколу стає проблемою визначення слова в системі з підстановкою елементів. Долев і Яо, використовуючи останній висновок, створили кілька алгоритмів для аналізу обмеженої множини протоколів. Модель Долева і Яо занадто обмежена і не підходить для аналізу багатьох протоколів. Вона може використовуватися тільки для виявлення помилок, які можуть призвести до порушення конфіденційності. Більшість методів аналізу протоколів, що використовують модель зловмисника Долева і Яо, розширюють її, щоб описати поведінку учасників протоколу.

1.7 Висновки

У розділі проведено аналітичний огляд криптографічних протоколів.

Проаналізовані різноманітні визначення поняття «протокол».

Визначені властивості, які повинен мати криптографічний протокол.

Виконано огляд класифікації протоколів та властивостей безпеки.

Описані атаки на криптографічні протоколи, методи аналізу та способи моделювання.

РОЗДІЛ 2
ПРОГРАМНА СИСТЕМА МОДЕЛЮВАННЯ КРИПТОГРАФІЧНИХ ПРОТОКОЛІВ

В даному розділі дипломної роботи пропонується програмна система для моделювання криптографічних протоколів (СМКП).

Ця система призначена для імітаційного моделювання роботи криптосистем в умовах наявності зовнішнього порушника, який прагне її зламати. Аналіз вразливостей криптосистем процес складний і ітераційний [8; 9; 11; 13] і тому передбачається, що дана система буде інструментом, що допомагає в складанні і моделюванні роботи криптосистеми.

Слід зазначити, що дана система призначена саме для імітації роботи саме криптографічних протоколів і тому використовувані в протоколах криптографічні алгоритми та методи вважаються безпечними.

Фактично ця система є програмним стендом, на якому дослідник криптографічних протоколів може становити різні протоколи. Цей інструмент дозволить йому позбавиться від рутинної роботи і дати йому можливість зосередитися на творчих підходах до вирішення поставлених завдань.

Далі в дипломній роботі описується СМКП стосовно до протоколів, що мають двох зареєстрованих учасників і одного зовнішнього порушника протоколу, але основні принципи побудови систем можуть бути поширені і для більшої кількості учасників.

2.1 Представлення криптографічного протоколу в системі

Програмної системою для моделювання криптографічних протоколів називається комп'ютерна програма, призначена для моделювання роботи відомих криптографічних протоколів по т.зв. моделі протоколу. Моделлю

протоколу називається формальний опис системи інформаційного обміну конфіденційною інформацією, в якій виділені основні об'єкти, що становлять систему, і відносини між цими об'єктами [13].

СМКП засноване на теорії кінцевих автоматах і використовують модель зловмисника, запропоновану Долевим і Яо (див. роз. 1), тобто. представляються системою дискретно-подієвого типу в якій події відбуваються в дискретні моменти часу (виконання одного кроку протоколу) або під впливом певних подій (наприклад-кінець циклу протоколу).

Реакція системи на події полягає в зміні змінних стану цієї системи і визначається в зміні набору даних кожного з учасників протоколу.

Визначимо стан системи як упорядкований набор станів учасників протоколу. Стан учасника протоколу визначається вмістом даних в його пам'яті на даному етапі протоколу і може складатися / складається з інформації закладеної попередньо при завантаженні протоколу в систему або інформації, отриманої в процесі роботи протоколу. До отриманої інформації відноситься також інформація, що не адресована даному учаснику протоколу, але передана по відкритому каналу зв'язку.

Крок протоколу визначаємо в даній роботі як конкретну закінчену дію, яка виконується учасником протоколу під час одного циклу і приводить до зміни безлічі даних або їх значень для будь-якого з учасників протоколу. Наприклад, обчислення значення деякої функції, перевірка правильності сертифікату ключа, генерація випадкового числа, відправка повідомлення і т.п. При цьому цикл протоколу завжди завершується кроком розширенням безлічі даних у будь-якого учасника протоколу, тобто отриманням нових даних (обміном повідомлення).

Розглянемо криптопротокол обміну повідомленнями (див. роз. 1) у вигляді показаному в таблиці 2.1.

Таблиця 2.1 – Криптографічний протокол обміну повідомленнями

Цикл	Крок	Опис кроку	Зміна даних
0		Початкові дані	K_{AB}, ID_A, ID_B :A K_{BA}, ID_A, ID_B :B
1	1	А формує текстову послідовність M1	+ M1 :A
	2	А обчислює S1 = E_{kAB}(M1).	+S1 :A
	3	А обчислює S2 = S1+ ID_A	+S2 :A
	4	А відправляє учаснику В повідомлення S2	Кінець циклу
2	1	В отримує повідомлення S2	+S2 :B
	2	Із заголовка дізнається ідентифікатор відправника ID^1_A При $ID^1_A \neq ID_A$ - Кінець протоколу	+ ID^1_A :B
	3	В обчислює текст M1 = D_{kAB}(S1).	+M1 :B
	4	В формує текстову послідовність M2	+M2 :B
	5	В обчислює S3 = E_{kBA}(M2).	+S3 :B
	6	В обчислює S4 = S3+ ID_B	+S4 :B
	7	В відправляє учаснику А повідомлення S4	Кінець циклу
3	1	А отримує повідомлення S4	+S4 :A

Продовження таблиці 2.1

	2	Із заголовка дізнається ідентифікатор відправника $+ID^1_B$ При $ID^1_B \neq ID_B$ - Кінець протоколу	$+ID^1_B$:A
	3	A обчислює текст $M2 = D_{kBA}(S2)$. Кінець протоколу	+M2 :A

Запис виду + X: Y позначає розширення множини даних учасника протоколу з ім'ям Y елементом X.

Такий опис дозволяє формалізувати криптографічний протокол як кінцевий автомат і представити його вигляді дискретно-подієвої системи, в якій зміна даних стану відбувається тільки в явно певні моменти часу (виконання одного кроку протоколу) або під впливом певних подій. (Кінець кроку або циклу протоколу). Реакція системи на ці події моментальна і полягає у зміні безлічі або значень даних у учасників протоколу.

2.2 Модель СМКП

2.2.1 Концептуальна модель

Основні положення структурної моделі СМКП цієї моделі зводяться до наступного:

— Противник може використовувати будь-які доступні йому комбінації стандартних операцій для побудови нових повідомлень з тих, які йому відомі. Передбачається, що, спостерігаючи повідомлення в каналі зв'язку, противник завжди знає їх структуру.

Невідомими йому можуть бути тільки конкретні значення окремих фрагментів повідомлення;

— якщо в деякому повідомленні міститься зашифрована інформація, то противник може використовувати її тільки в даному, зашифрованому вигляді, або, щоб витягти її, противник повинен знати відповідний ключ розшифрування. Передбачається, що всі криптографічні алгоритми є ідеальними;

— передбачається, що противник має повний контроль над усіма каналами зв'язку. Це означає, що противник може прослуховувати всі повідомлення, видаляти повідомлення з каналів зв'язку і перенаправляти їх іншим адресатам, формувати і відправляти повідомлення будь-яким учасникам протоколу.

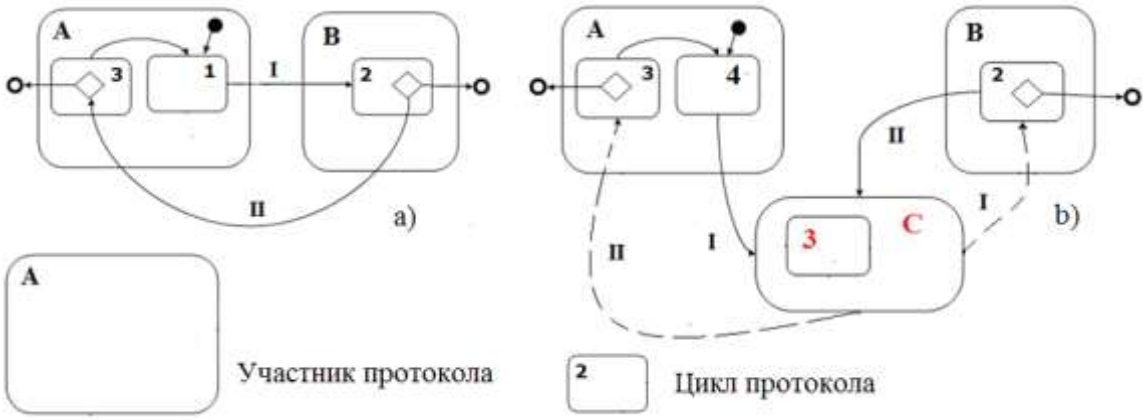

Рисунок 2.1 – Граф станів учасників протоколу у разі наявності зовнішнього противника

Щоб мати можливість аналізувати криптографічні протоколи з точки зору безпеки в умовах повної свободи дій противника, поняття стану учасника протоколу розширюється безліччю повідомлень, записаних в його пам'яті зловмисника.

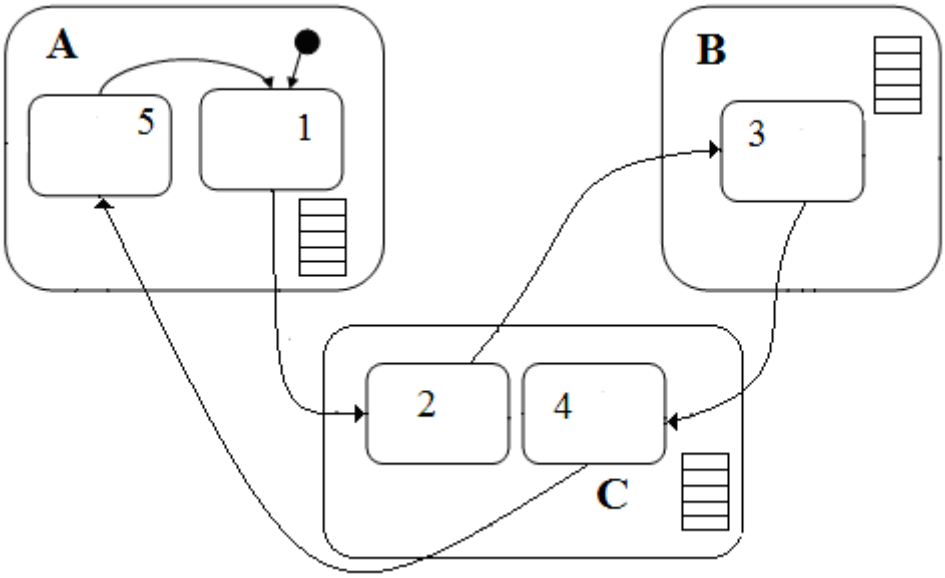

Рисунок 2.2 – Представлення протоколу з набором даних учасників

У системі, що реалізує криптографічні протоколи з зовнішнім ворогом, пропонується використовувати структуру портів прозорих для зареєстрованих учасників протоколу.

Порт – дві області пам'яті OUT і IN та керуючою функцією F. В пам'ять OUT завжди записуються повідомлення, що передаються від активного зареєстрованого користувача до пасивного зареєстрованого користувачу (див. рис. 2.3). Повідомлення зберігається в OUT до тих пір, доки керуюча функція F не видасть певну команду.

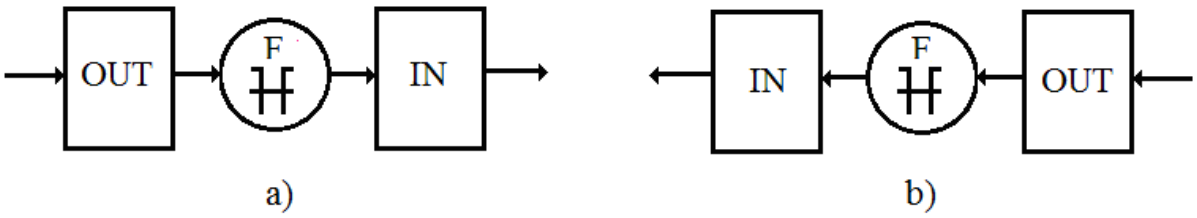

Рисунок 2.3 – Схема роботи портів

Порядок управління пам'яттю порту виглядає наступним чином:

1 Керуюча функція завжди зчитує дані з порту OUT в момент їх надходження, направляє їх в УФП і очищає порт OUT;
2 По закінченню окремого циклу протоколу (зміни активного учасника протоколу) пам'ять OUT працює як пам'ять IN і навпаки, IN працює як OUT (див. рис. 2.3).
3 Керуюча функція може виконувати одну з чотирьох команд, а саме:
   — Передати в порт IN отриману інформацію;
   — Нічого не передавати в порт IN;
   — Передати в порт IN заздалегідь сформований повідомлення;
   — Передати інформацію в пам'ять порушника.

Використання цих команд в певній послідовності дозволяє реалізувати будь-які можливі дії порушника: прослуховування, заміну повідомлення, переривання передачі або здійснювати передачу даних без втручань. Таким чином, така реалізація буферизованого введення-виведення дозволяє включити в криптографічний протокол порушника, у вигляді додаткових циклів протоколу, а завдання імітації дій порушника полягає у розробці таких кроків протоколу, які призводять до порушення роботи або злому протоколу. Подією, що активізує функцію F вважається наявність будь-якої інформації в порту IN.

Концепція роботи СМКП полягає в наступному (див. рис. 2.4):

1 Є якась підсистема дозволяє користувачеві системи формувати протокол криптосистеми (основний протокол);
2 Користувачем системи цей протокол доповнюється кроками порушника (доповнений протокол);
3 Система управління СМКП виконує доповнений протокол, здійснюючи кожен його крок. При цьому відбуваються всі описані в протоколі дії, як зареєстрованих учасників, так і дій порушника.

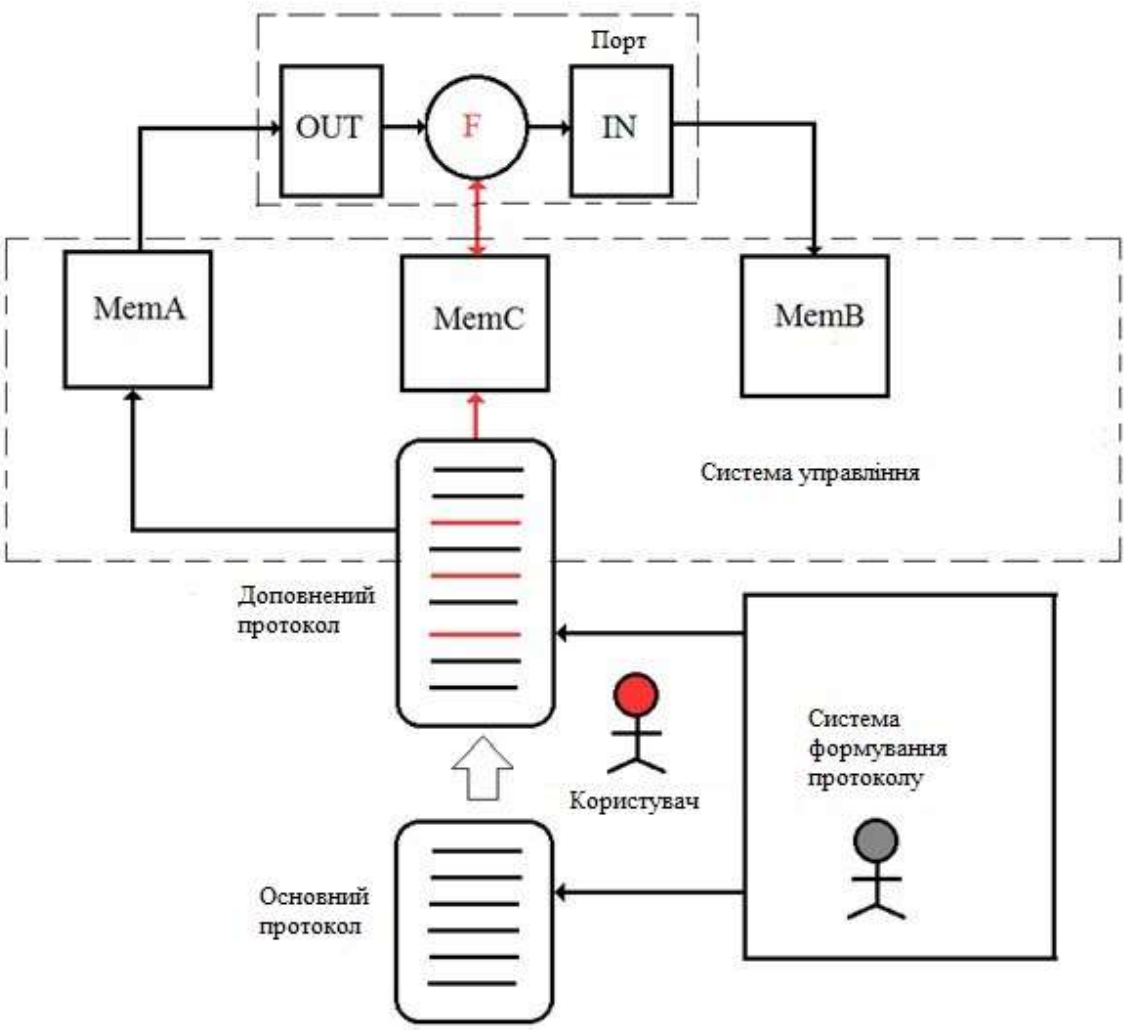

Рисунок 2.4 – Концептуальна модель системи для трьох циклу протоколу

Формально доповнений протокол описує дії двох зареєстрованих учасників протоколу та зовнішнього противника, у яких є своя множина даних і які виконують свої кроки по зміні і пересиланні цих даних.

Якщо всі учасники чесно виконують всі визначені протоколом дії, то сеанс повністю повторює опис протоколу, відрізняючись лише тим, що в ньому фігурують імена і ідентифікатори конкретних учасників.

Якщо в роботу протоколу втручається противник, то в результаті можуть виходити такі комбінації дій, при яких відбувається порушення одного або кількох властивостей протоколу, що призводить до реалізації сценарію атаки на протокол.

Передбачається, що дослідження протоколу може здійснюватися для безлічі сеансів роботи одного і того ж протоколу. Тому в пам'яті порушника зберігаються дані, отримані в ході моделювання попередніх сеансів. Приклад доповнення протоколу наведеного в табл. 2.2.

Таблиця 2.2 – Приклад доповненого протоколу

Цикл	Крок	Описание шага	Изменение данных
0		Початкові дані	K_{AB}, ID_A, ID_B :A K_{BA}, ID_A, ID_B :B
1	1	А формує текстову послідовність M1	+ M1 :A
	2	А обчислює S1 = E_{kAB}(M1).	+S1 :A
	3	А обчислює S2 = S1+ ID_A	+S2 :A
	4	А обчислює в OUT повідомлення S2	+S2 :OUT
2	1	С зчитує з OUT повідомлення S2	+S2 : C
	2	С обчислює ID_A= S2-S1	+ ID_A : C
	3	С відправляє S2 в IN	+S2 :IN
3	1	В зчитує з IN повідомлення S2	+S2 :B
	2	Із заголовка дізнається ідентифікатор відправника ID^1_A При $ID^1_A \neq ID_A$ - Кінець протоколу	+ ID^1_A :B
	3	В обчислює текст M1 = D_{kAB}(S1).	+M1 :B

2.2.2 Архітектура СМКП

Архітектура системи – це логічна побудова системи, що включає в себе перелік і формат команд, форми представлення даних, механізми введення / виведення, способи адресації пам'яті і т.д. Поняття структури системи включає в себе питання фізичного побудови: склад пристроїв, ємність пам'яті, наявність модулів обробки і підготовки даних і т. д.

У структурі СМКП зображеної на рис. 2.4, можна виділити наступні основні пристрої: систему підготовки протоколів, інтерпретатор команд, оперативну пам'ять учасників протоколу, порти обміну даними і обчислювальну систему.

Система підготовки протоколів забезпечує необхідний інтерфейс для:

- Створення нового основного протоколу або читання зі сховища протоколів раніше описаного протоколу;
- Створення додаткового протоколу або читання зі сховища протоколів раніше описаного додаткового протоколу;
- Створення коду (основного або розширеного) протоколу або читання зі сховища раніше створених протоколів.

Для роботи із кодом створеного протоколу необхідно розробити інтерпретатор. Інтерпретатор команд виконує порядковий аналіз і виконання коду протоколу по мірі надходження чергової команди. До складу інтерпретатора команд входить:

- Регістр лічильник команд – задає номер команди коду протоколу, яка буде виконуватися після виконання поточної команди. Якщо цей регістр обнуляється робота протоколу завершується. В початковий момент роботи йому присвоюється значення 1;
- Регістр імені учасника протоколу визначає ім'я учасника протоколу який виконує перетворення на даному етапі протоколу, а значить

визначає оперативну пам'ять в якій з якої будуть вилучатись дані і зберігатися результати виконуваних операцій;

— Парсер – програма для перетворення текстового представлення однієї команди коду протоколу в виклик певної функції математичного перетворення відповідних даних.

Синтаксичний аналіз (парсинг) в інформатиці – процес зіставлення лінійної послідовності слів (токенів) природної мови з його формальної граматикою [17]. Синтаксичний аналіз – це основа всіх компіляторів і інтерпретаторів мов високого рівня.

У ході розбору вихідний текст перетворюється в структуру даних, яка відображає синтаксичну структуру вхідної послідовності і добре підходить для подальшої обробки.

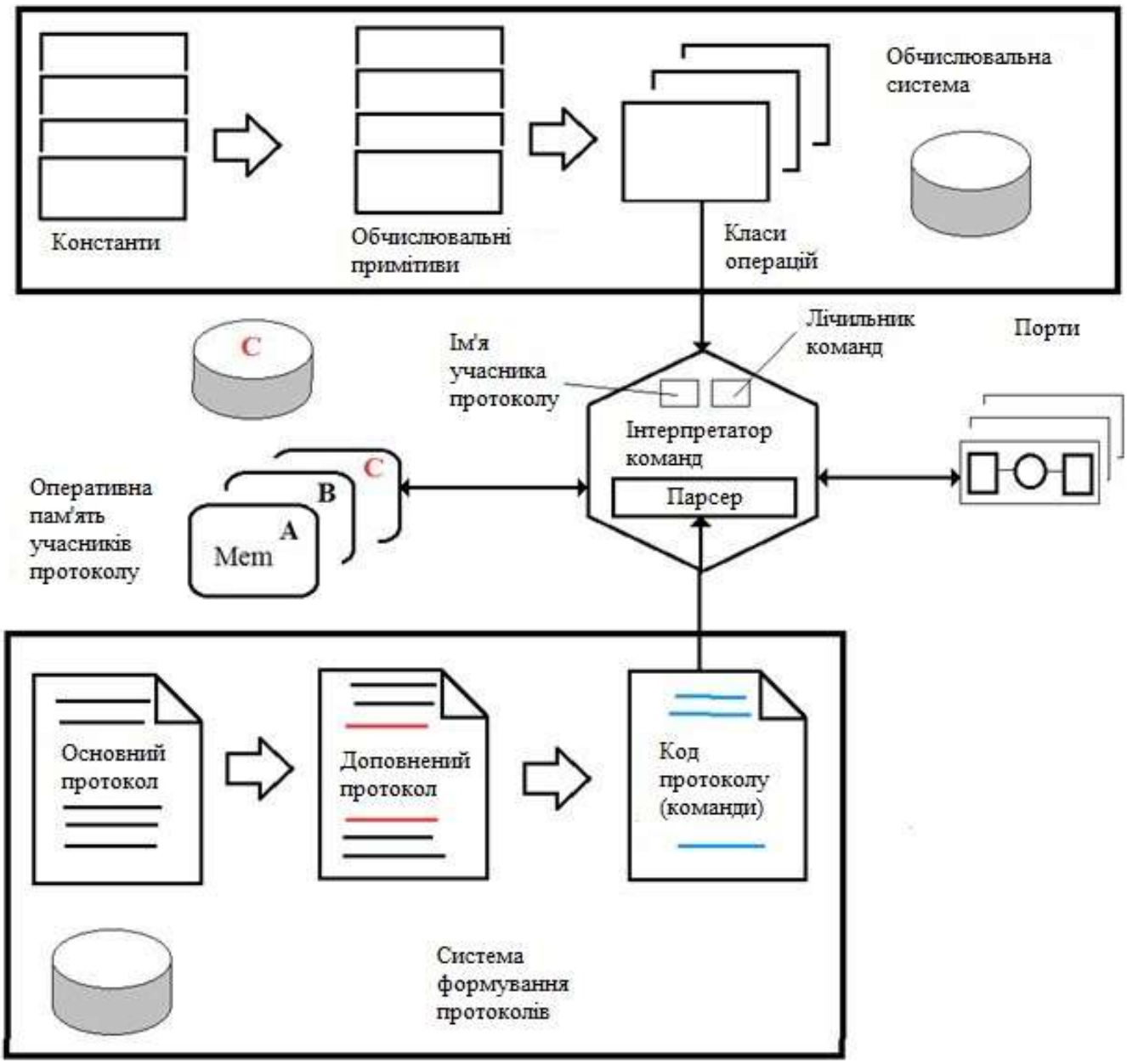

Рисунок 2.5 – Загальна структура системи.

Обчислювальна система являє собою програмний модуль, призначений для:

— Реалізації криптографічних операцій (шифрування, дешифрування, хешування та ін.), що викликаються інтерпретатором;

— Реалізації інших обчислювальних операцій, необхідних для реалізації криптографічних операцій або окремих команд протоколу (наприклад генерації випадкових чисел, перевірки чисел на простоту, операцій модульної арифметики і т.д.);

— Зберігання набору константних даних (наприклад простих чисел).

У цьому модулі знаходяться заздалегідь розроблені, налагоджені і перевірені функції і дані тих математичних перетворень, які можуть використовуватися в різних криптографічних протоколах.

Оперативна пам'ять учасників протоколу (Mem A, B, C) є динамічними масивами даних, що отримуються в ході виконання окремих кроків протоколу для кожного з його учасників в окремому його сеансі.

Порти обміну даними і порядок їх роботи описані раніше в п.2.2.1.

Команда, що обробляється інтерпретатором, визначає виконувану операцію і містить в явній або неявній формі інформацію про те, де буде поміщений результат операції, а також про адресу наступної команди. Код команди складається з декількох частин, які називаються полями. Довжина команди, кількість, розмір, положення, призначення і спосіб кодування її полів називається форматом команди.

У загальному випадку формат команди містить операційну та адресну частини.

Операційна частина містить код операції (наприклад, модульне зведення числа в ступінь). Адресна частина складається з декількох полей і містить інформацію про адреси операндів, результату операції і наступної команди. Адресна частина, в свою чергу, може складатися з двох полів (типу адресації і адреси операнда).

2.3 Особливості програмної реалізації

2.3.1 Вимоги до програмної реалізації

Програмна реалізація системи повинна відповідати таким основним властивостям і виконувати наступні вимоги:

— Мати простий і зрозумілий візуальний інтерфейс протоколу;
— Бути відкритою і масштабованою;
— Взаємозамінність програмних модулів;
— Система повинна бути адаптивна, тобто повинна бути пристосована до змін зовнішніх умов, в тому числі організаційно-функціональної структури;
— Система повинна забезпечувати можливість розширення без зміни інших функціональних частин системи;
— Система повинна забезпечувати міграцію програмного забезпечення при модернізації або заміни апаратно-програмних платформ;

Комп'ютерна система для моделювання протоколів також повинна:

— Реалізовувати графічний інтерфейс роботи з криптографічними протоколами, що відображає всі етапи його роботи в часі;
— Дозволяти уповільнене і повторне відтворення роботи протоколу, в штатному режимі його роботи і при діях порушника;
— Зберігати і відображати будь-якої його фрагмент і мати довідкові дані.

Система моделювання криптографічних протоколів повинна реалізовувати в собі наступні обчислювальні примітиви:

— Шифрування;
— Хешування;
— Обчислювання цифрових підписів;
— Математичні операції;
— Додаткові функції , такі як генерація випадкових чисел.

Перелік доступних примітивів повинен бути легко змінюваним для розробника програмної системи, та регулярно доповнюватись новими алгоритмами, на заяву користувачів.

Програма, що реалізує систему моделювання криптографічних протоколів, повинна являти собою комплекс і бути сумісна з найбільш популярними ОС.

2.3.2 Реалізація прототипу пілотного проекту

Для створення програмної реалізації використовувалася мова програмування C # з підключенням .NET Framework 4.5.2. Типом проекту було обрано WPF (Windows Presentation Foundation).

C# – сучасна мова програмування надає широкий спектр можливостей для розробки програм для операційних систем сімейства Windows. Також фреймворк має потужну криптографічний бібліотеку, яка містить в собі всі необхідні для проекту алгоритми.

Через обмеження мови та фреймворку, програмна реалізація може працювати лише на ОС сімейства Windows, із встановленим .NetFramework 4.5.2.

Системні вимоги:

— ОС Windows XP, Vista, 7, 8, 8.1, 10;
— Встановлений .NetFramework 4.5.2;
— 512 МБ вільної оперативної пам'яті;
— Процесор із потужністю 2 GHz.

2.3.2.1 Опис інтерфейсу програми

Програма складається із 3 головних вікон:

— Стартове вікно;
— Вікно конструктора протоколів;

— Вікно виконувача протоколів.

Запуск програми здійснюється з ярлика. Відкривається стартове вікно (див. рис. 2.6), що містить 2 кнопки для навігації по програмі. З цього вікна користувач може перейти на конструювання протоколів або на їх виповнення відповідно.

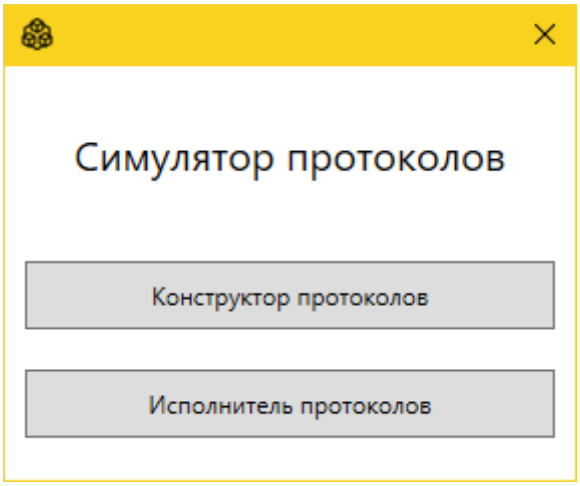

Рисунок 2.6 – Стартове вікно програми

При відкриті конструктора протоколів буде відкрито вікно, зображене на рисунку 2.7. Це вікно дозволяє користувачу сформувати протокол із обчислювальних примітивів.

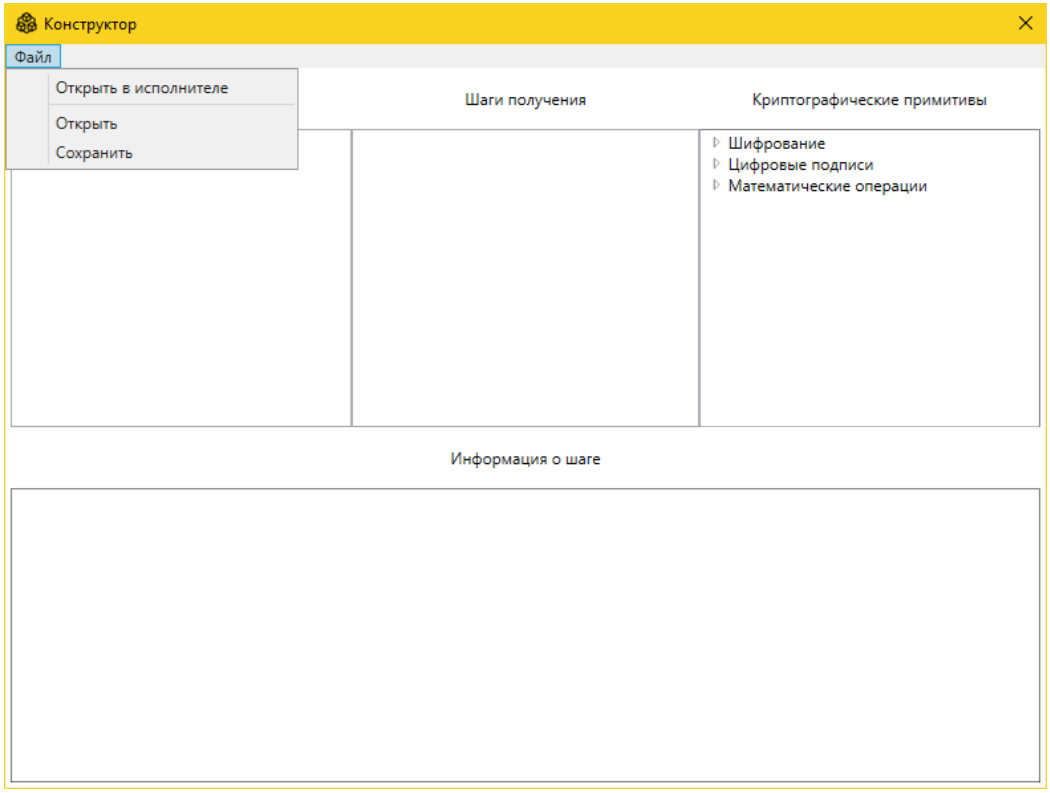

Рисунок 2.7 – Вікно програми «Конструктор протоколів»

Меню містить 3 пункти:

1 «Открыть в исполнителе» – при натисканні на цю кнопку програма зберігає поточний протокол в тимчасовому файлі і відкриває його в виконавці протоколів;
2 «Открыть» – відкриває вікно провідника для вибору файлу з кодом протоколу, після чого відбувається його завантаження для подальшого редагування;
3 «Сохранить» – відкриває вікно провідника для вибору файлу, куди буде збережений відкритий в конструкторі протокол.

«Шаги отправки» – кроки протоколу, що виконуються користувачем, який виступає відправником повідомлення.

Відповідно «Шаги получения» – кроки протоколу, що виконуються стороною, одержує повідомлення.

Поле «Информация о шаге» – містить інформацію про обраний крок протоколу в «Шаги отправки» або «Шаги получения».

Список «Криптографические примитивы» являє собою деревовидний список, де коренем виступає тип примітиву, а вузлом примітив. Наприклад, корінь «Шифрование», вузол «RSA».

При наведенні курсору миші, користувач отримує випадаючу підказку, яка містить коротку інформацію про примітив (рис. 2.8).

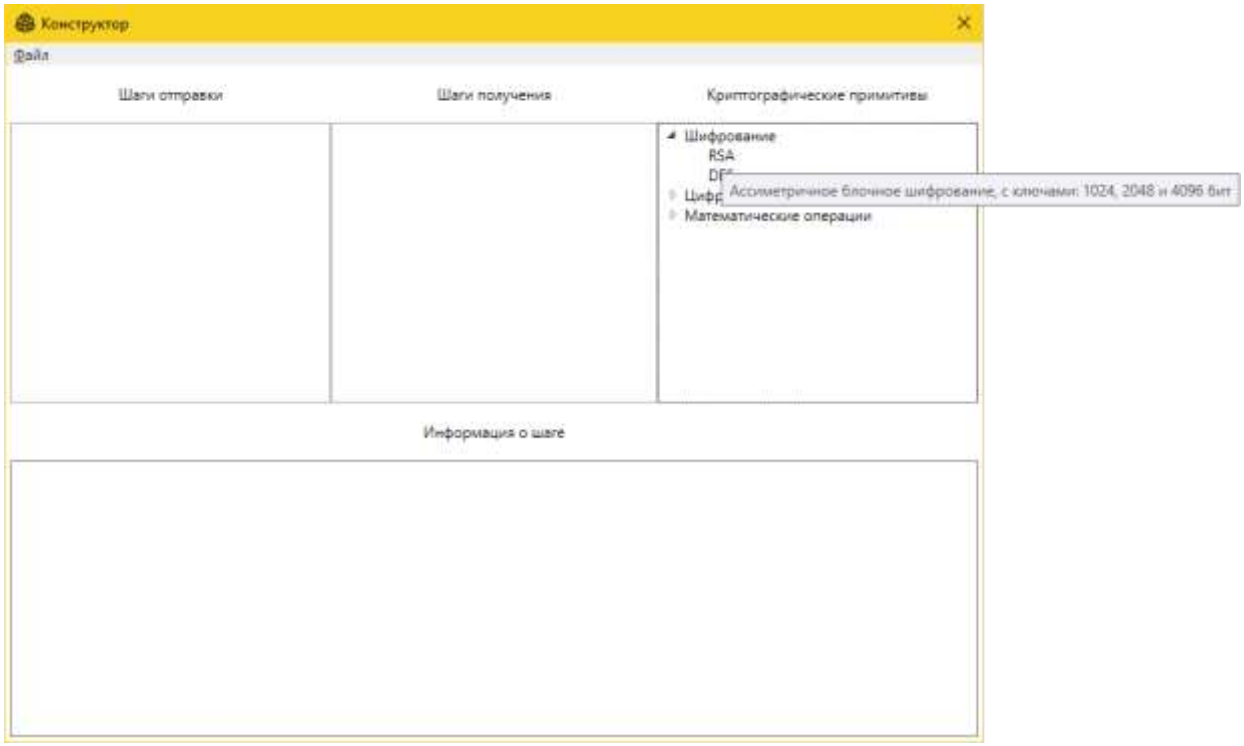

Рисунок 2.8 – Розгорнутий список примітивів з ви падаючою підказкою

Редагування протоколу відбувається шляхом перетягування, за допомогою миші, примітивів в списки «Кроки відправки» і «Кроки отримання». Примітиви деяких груп автоматично додаються в обидва списки в зворотному порядку (див. рис. 2.9)

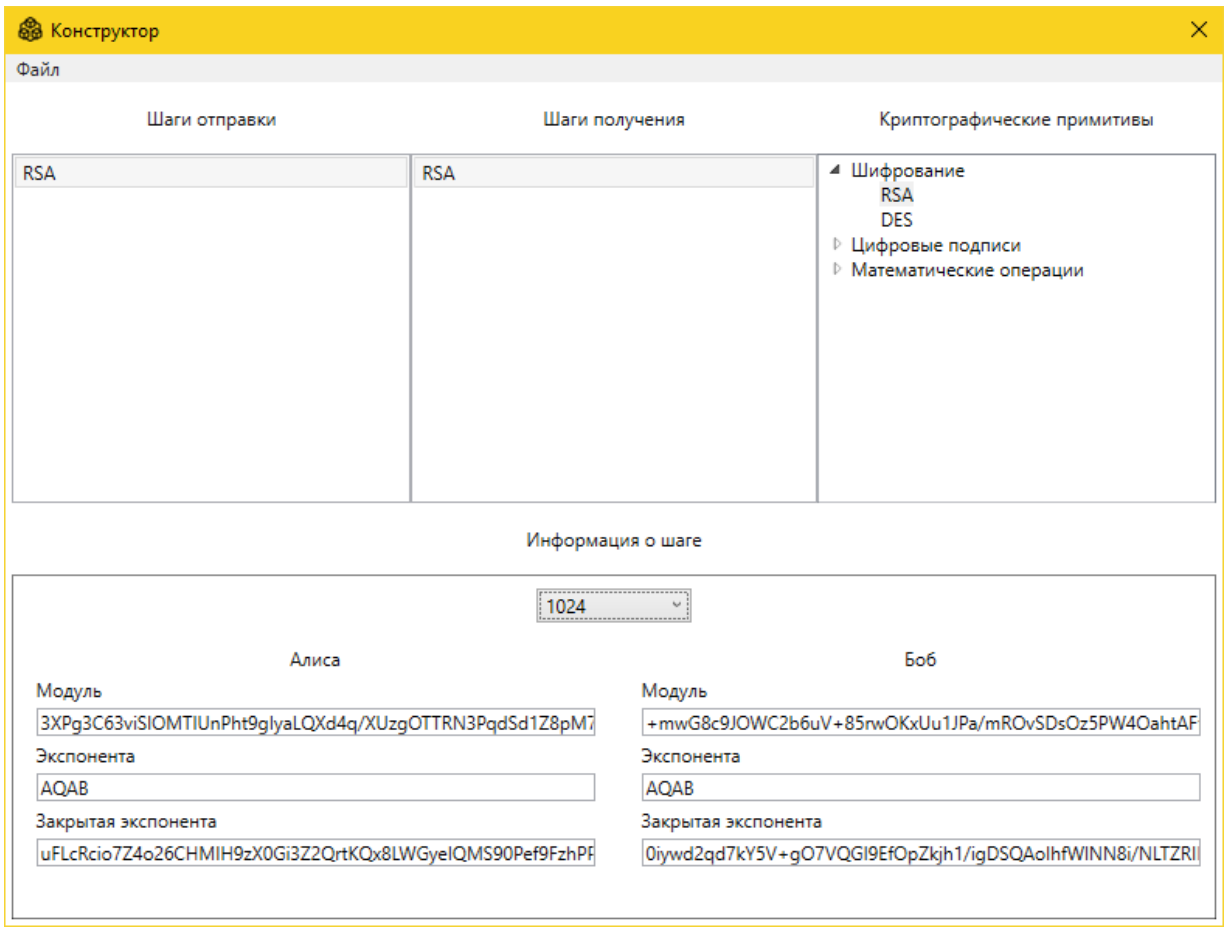

Рисунок 2.9 – Демонстрація редагування протоколу і відображення інформації про обраний крок протоколу.

Після перетягування, будь-якого примітиву до складу протоколу, він автоматично відзначається як обраний, і відкриваються його властивості. За замовчуванням поля автоматично заповнюються, проте користувач може їх змінювати, вписуючи свої дані, або використовуючи збережені константи. Значення цих полів будуть збережені в файл протоколу.

Якщо ж в стартовому вікні користувач вибирає пункт «Виконавець протоколів» буде відкрито провідник, для вибору файлу, в якому збережений протокол. Після цього буде відкрито вікно виконавця протоколів, який завантажить вибраний файл і надасть інтерфейс для роботи з обраним протоколом (див. рис. 2.10).

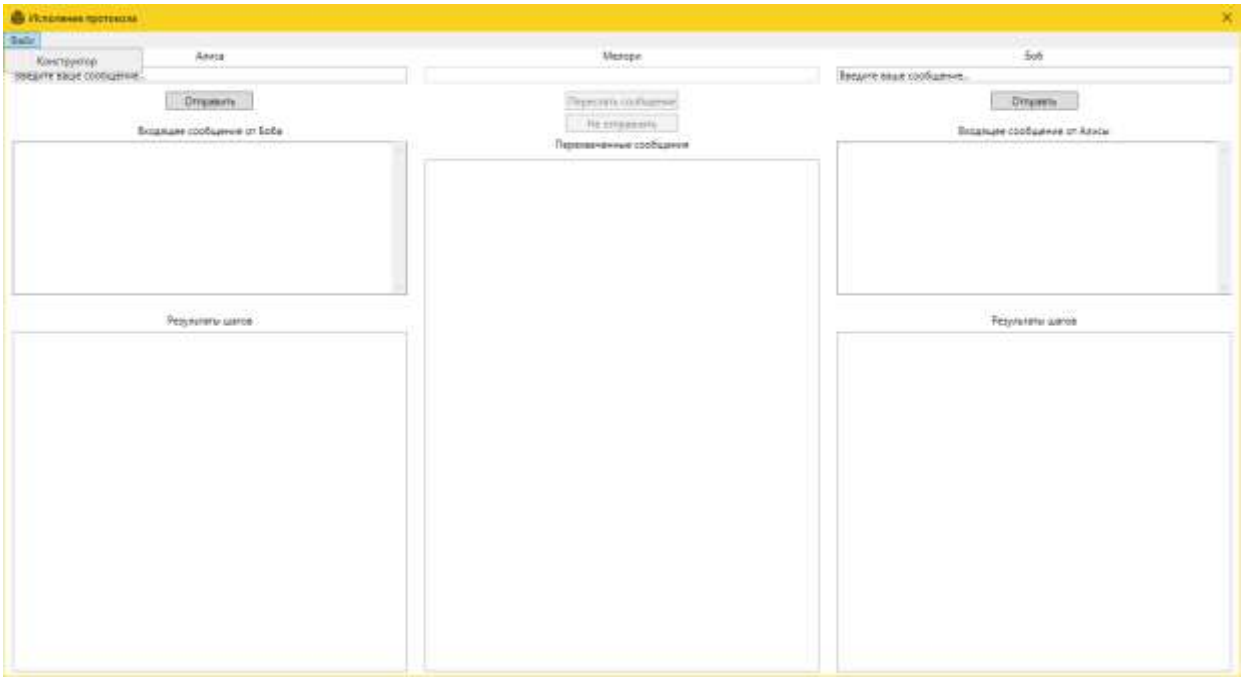

Рисунок 2.10 – Вікно виконавця протоколів, початковий стан

Пункт меню «Конструктор» відкриває конструктор для поточного протоколу.

Вікно має 3 поля введення - у Аліси, Мелорі і Боба. Вони призначені для створення та модифікування повідомлень.

Кнопки «Отправить» ініціюють сеанс протоколу. А «Переслать сообщение» і «Не отправлять» дозволяють противнику втручатися в роботу протоколу.

Списки «Результаты шагов» будуть відображати кроки та інформацію про них під час виконання протоколів.

Список «Перехваченные сообщения» – відображає усі перехоплені порушником повідомлення, та повідомлення, що були переслані далі.

Коли повідомлення введено, після натискання кнопки відправити, починається виконання кроків протоколу. Виконання тимчасово зупиняться, коли повідомлення досягає порушника. Після чого є 3 варіанти дій: переслати

повідомлення далі, змінити його в поле введення порушника, або перервати відправку повідомлення (див. рис. 2.11).

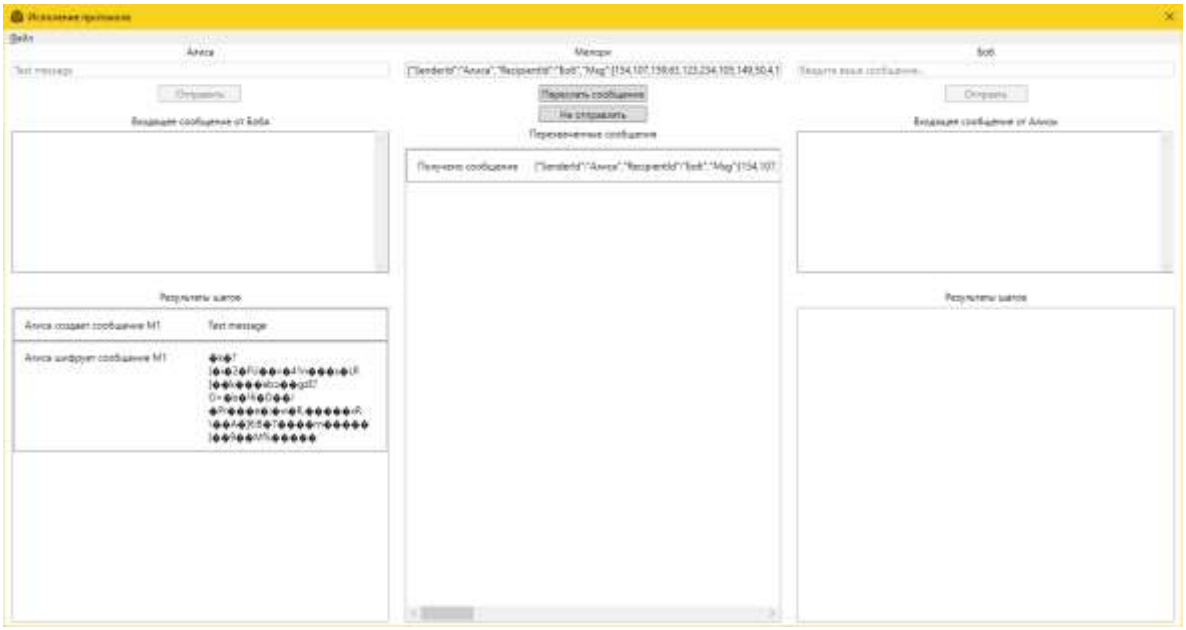

Рисунок 2.11 – Виконання протоколу

При пересиланні порушником повідомлення одержувачу, одержувач виконує визначені кроки протоколу, записуючи результати кожного кроку в «Результати кроків».

Після виконання циклу протоколу, в вікні остаються результати виконання кроків кожним учасником. На рисунку 2.13 зображено вікно програми, після завершення усіх кроків протоколу.

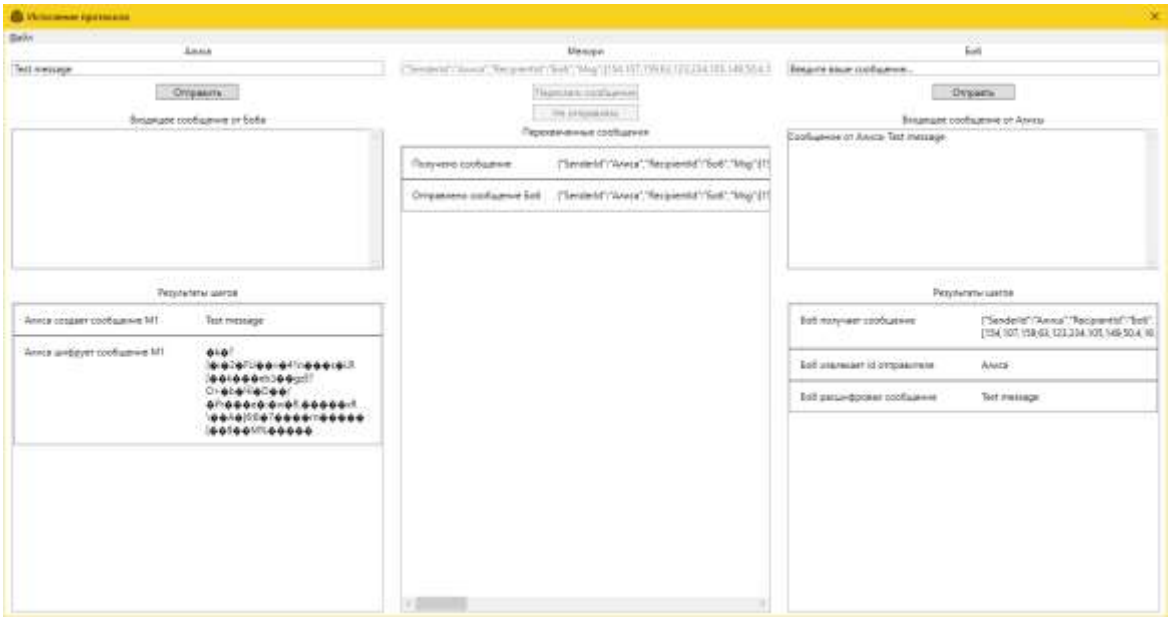

Рисунок 2.12 – Вікно програми після завершення кроків протоколу

2.3.2.2 Опис формату даних програми

Для збереження даних програми використовується формат JSON «ECMA-404 The JSON Data Interchange Standard».

JSON - простий текстовий формат даних, вперше описаний в 1999 році в «Standard ECMA-262 3rd Edition - December 1999». Він дозволяє описати структуру даних у вигляді «Ключ-Значення», де «Ключ» - строка, а значення може бути об'єктом, примітивним типом даних або масивом об'єктів або примітивних типів. До підтримуваних примітивних типів даних відносяться: строки, символи, цілі числа, числа з плаваючою комою, логічний тип даних (true / false), null.

Даний формат має низку переваг:

— Інтуїтивно зрозумілий, може бути прочитаний і змінений людиною;

— Може бути сформований і прочитаний програмою;

— У всіх сучасних мовах програмування присутні вбудовані або сторонні інструменти для роботи з JSON, що дозволяє з легкістю передавати дані між програмами, написаними на різних мовах.

Інтуїтивно зрозумілий формат даних дозволить експертному користувачу вносити зміни в протокол одразу в файлі, а не в графічному інтерфейсі, що прискорює формування та зміну протоколів.

Константи зберігаються у файлах <код_примітиву>.cpc (Crypto Primitives Constants). Файл містить у собі JSON масив об'єктів або примітивів, в залежності від примітиву. Наприклад для DES ключі можна зберігати як масив строк, але для RSA, це буде масив об'єктів, які містять необхідні дані для генерації ключів, та самі ключі.

Змодельовані протоколи зберігаються у файлах <назва_протоколу>. cpsc (Crypto Protocol Source Code). Файл містить у собі JSON об'єкт з двома полями, «sender» і «receiver». Значеннями кожного поля є масиви, кожен з яких містить об'єкти, які описують крок протоколу при відправці і отриманні повідомлення відповідно, елементи в масивах розташовані в порядку їх виконання.

Для того, що б описати крок протоколу досить 3 полів:

— «type» – описує тип кроку, наприклад «encryption» або «digital_signature»;
— «name» – назва кроку, наприклад» DES» або «RSA»;
— «payload» – об'єкт, який містить в собі дані необхідні для виконання кроку, наприклад ключі шифрування учасників протоколу.

Наприклад, для протоколу, який перед відправкою шифрує повідомлення за допомогою DES, а при отриманні розшифровує, код протоколу буде виглядати наступним чином:

Рисунок 2.13 – Приклад коду протоколу з використанням DES

Коли парсер зчитує файл протоколу, порядок виконання команд регламентується порядком об'єкта в масиві. На основі «type» та «name» вибирається відповідна частина програми, що представляє відповідну обробку даних та вміє обробляти відповідний «payload».

2.3.3 Розробка фінальної реалізації ПЗ

Розробка реальної версії такого проекту, важкий трудомісткий процес, що триває декілька місяців.

На основі дипломного проекту потрібно розробити деталізоване технічне завдання, в основу якого буде включена архітектура розробленої системи.

Фінальну версію цієї системи має сенс розробляти у вигляді веб-додатку. Такий тип реалізації має ряд переваг, над розробкою десктопної версії програми.

По-перше, веб-дадоток буде багатоплатформовий, його реалізація не буде прив'язана до певної операційної системи. Що дозволить його використовувати більшій кількості людей, а розробнику зекономити на розробці.

Також усе більше набирають популярність хром буки та док станції, що дозволяють використовувати андроїд пристрої як комп’ютери. Обраний мною формат не тільки вирішує проблему з різноманітною кількістю систем операційних систем, а також дозволяє використовувати хмарні обчислення, що дозволяє запускати програму майже без прив’язки до апаратного забезпечення користувача.

По-друге, дозволяє розробити гнучкий красивий зрозумілий інтерфейс, що буде відображатися на усіх типах ОС та екранах коректно.

По-третє, веб спрощує монетизацію додатку, та робить піратство майже не можливим.

2.4 Висновки

У розділі запропоновано уточнений спосіб описання криптографічних протоколів, с більш детальним описом їх шагів та циклів.

Для цього було введене нове визначення поняття «крок протоколу», яке дає можливість представити протокол у вигляді, що дозволяє змоделювати його в системі дискретно-подієвого типу.

Обґрунтована практична цінність розробки програмної системи для моделювання криптографічних протоколів.

Розроблена концепція та архітектура програмної системи на основі моделі Долева-Яо.

Сформовані вимоги до системи моделювання протоколів.

Розроблено прототип пілотної версії програми на основі запропонованої архітектури. Описані деталі реалізації прототипу, та представлені рекомендації стосовно розробки повноцінної фінальної версії системи.

РОЗДІЛ 3
РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА РЕАЛІЗАЦІЇ ЙОГО КІНЦЕВОМУ КОРИСТУВАЧУ

При розробці програмного забезпечення важливими етапами є визначення трудомісткості розробки ПЗ, розрахунок початкових та експлуатаційних витрат.

Проект, для якого виконується розрахунок, буде розроблено як веб-додаток, що буде розповсюджуватись по моделі SaaS (Програмне забезпечення як послуга).

Так як проект є складним, але не потребує постійної підтримки, а тільки періодичні доповнення та виправлення, тому немає необхідності в постійному штаті співробітників. Розробка програмного забезпечення буде виконуватись фрілансерами або компанією - підрядником. Такий хід, при розробці ПЗ, дозволить уникнути наступних витрат:

1. На закупівлю апаратного забезпечення;
2. На закупівлю ліцензійного програмного забезпечення;
3. На оренду та утримання офісного приміщення;
4. На комунікації (електроспоживання, інтернет, тощо);
5. На оплата заробітної плати постійним співробітникам.

Так як, ПЗ являє собою веб-додаток, то до розрахунку витрат необхідно включити хостинг, що буде займатися обслуговуванням.

3.1 Розрахунок трудомісткості розробки програмного забезпечення

По-перше необхідно сформувати технічне завдання, що буде передане для виконання дизайнеру та програмісту. Складанням технічного завдання повинен займатися замовник, або довірена особа. Так як дипломний проект містить абстрактний опис архітектури ПЗ та вимог до нього (див. роз. 2), на створення завдання закладемо:

$$t_{тз} = 56 \text{ годин}$$

Трудомісткість на розробку програмної частини ПЗ можна розрахувати за формулою:

$$t_{пз} = t_{тз} + t_{дос} + t_а + t_п + t_н + t_{док} \text{ годин}, \quad (3.1)$$

де: $t_{дос}$ – витрати праці на дослідження алгоритму розв'язання задачі;

$t_а$ – витрати праці на розробку блок-схеми алгоритму;

$t_п$ – витрати праці на програмування по готовій блок-схемі;

$t_н$ – витрати праці на налагодження програми на ЕОМ;

$t_{док}$ − витрати праці на підготовку документації.

Складові трудомісткості визначаються на підставі умовної кількості операторів у програмному продукті Q (з урахуванням можливих уточнень у процесі роботи над алгоритмом і програмою). Умовна кількість операндів у програмі:

$$Q = q * C * (1 + p), \quad (3.2)$$

де: q – передбачувана кількість операторів;

C – коефіцієнт складності програми. Коефіцієнт складності завдання З характеризує відносну складність програми по відношенню до так званої типової задачі, що реалізує стандартні методи рішення, складність якої прийнята рівною одиниці (величина С лежить в межах від 1,25 до 2);

p – коефіцієнт корекції програми, збільшення обсягу робіт за рахунок внесення змін до алгоритму або програму за результатами уточнення постановок. Його величина знаходиться в межах від 0,05 до 0,1.

Для цього проекту q = 1640.

Так як ПЗ є складним, містить елементи складного динамічного графічного інтерфейсу, та логіку що програмується користувачем. Коефіцієнт С = 1,7.

Так я ТЗ буде детально опрацьоване, то коефіцієнт p = 0,06.

Виходячи з обраних коефіцієнтів, умовна кількість операндів у програмі:

Q = 1640 * 1,7 * 1,06 = 2955 операндів

Витрати праці на вивчення опису завдання визначається з урахуванням уточнення опису і кваліфікації програміста.

$$t_{дос} = Q * B / [(75…85) * K], \quad (3.3)$$

де: B − коефіцієнт збільшення витрат праці внаслідок недостатнього опису завдання. Коефіцієнт приймається від 1,2 до 1,5;

K – коефіцієнт кваліфікації програміста, який визначається від стажу роботи за даною спеціальністю. Коефіцієнт становить: для працюючих до двох років - 0,8; від двох до трьох років - 1,0; від трьох до п'яти років - 1,1 - 1,2; від п'яти до семи - 1,3 - 1,4; понад сім років - 1,5 - 1,6.

Для даного проекту B = 1,2.

Так як програма складна, будемо розраховувати на роботу із програмістом зі стажем від трьох до п'яти років, тому K = 1,2.

Враховуючи коефіцієнти та попередні розрахунки:

$$t_{дос} = (2955 * 1,3) / (80 * 1,2) = 40 \text{ годин},$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = Q / [(20…25) * K], \quad (3.4)$$

$$t_a = 2955 / (25 * 1,2) = 98,5 \text{ годин},$$

Витрати праці на складання програми по готовій блок-схемі:

$$t_п = Q / [(20…25) * K], \quad (3.5)$$

$$t_п = 2955 / (20 * 1,2) = 123 \text{ годин},$$

Витрати праці на налагодження програми за умови автономного налагодження одного завдання:

$$t_н = Q / [(4…5) * K], \quad (3.6)$$

$$t_н = 2955 / (5 * 1,2) = 492.5 \text{ годин},$$

При комплексному налагодженні програми витрати на налагодження зростають в півтора рази. Таким чином, витрати праці на налагодження програми за умови комплексного налагодження завдання:

$$t^K_н = 1,5 * t_н, \quad (3.7)$$

$$t^K_н = 1,5 * 492,5 = 738,75 \text{ годин},$$

Витрати на підготовку документації:

$$t_{док} = t_{др} + t_{до}, \quad (3.8)$$

де: $t_{др}$ − трудомісткість підготовки матеріалів і рукописи;

$t_{до}$ − трудомісткість редагування, друку і оформлення документації.

Трудомісткість підготовки матеріалів і рукописи визначається за формулою:

$$t_{др} = Q / [(15…20) * K], \quad (3.9)$$

$$t_{др} = 2955 / (20 * 1,2) = 123,13 \text{ годин},$$

Трудомісткість редагування, друку і оформлення документації:

$$t_{до} = 0,75 * t_{др}, \quad (3.10)$$

$$t_{до} = 0,75 * 111,7 = 92,34 \text{ годин}.$$

Общие затраты на подготовку документации составят:

$$t_{док} = 123,125 + 92,34 = 215,47 \text{ годин}.$$

Розраховуємо трудомісткість розробки ПЗ:

$$t_{пз} = 56 + 40 + 98,5 + 123 + 492,5 + 215,47 = 1025,47 \text{ годин}.$$

3.2 Розрахунок витрат на створення ПЗ

Витрати на створення ПЗ ($К_{пз}$) включають витрати на заробітну плату розробників програми ($З_{зп}$), яка визначається множенням сумарної трудомісткості розробки ПЗ (t) на середню заробітну плату програміста з нарахуваннями та вартості машинного часу на налагодження.

$$К_{пз} = З_{зп} + З_{мв} \quad (3.11)$$

Заработная плата разработчиков определяется по формуле:

$$З_{зп} = t * С_{пр}, \quad (3.12)$$

Так як ми обчислювали час на розробку програмного забезпечення с залученням програміста зі стажем від трьох до п'яти років, то середня заробітна плата в Україні для розробників такого класу складає 280 гривень на годину.

$$З_{зп} = 1025,47 * 280 = 287131,6 \text{ грн}.$$

Так як ми використовуємо труд фрілансерів або компанії- підрядника, витратами на машинний час для налагодження ПЗ можна знехтувати, так як вони включені в заробітну платню розробника.

Витрати на створення програмного забезпечення складуть:

$$К_{пз} = 287131,6 + 0 = 287131,6.$$

Певні таким чином витрати на створення програмного забезпечення є одноразовими капітальними витратами.

Очікуваний період створення ПЗ:

$$T = t / (B_k * F_p) \text{ мес.}, \quad (3.13)$$

де: B_k – кількість залучених розробників;

F_p − місячний фонд робочого часу (при 40 годинному робочому тижні $F_p = 176$ годин).

$$T = 1025{,}47 / (1 * 176) = 5{,}8 \text{ місяців.}$$

Таким чином, очікувана тривалість розробки складе близько 5,8 місяців, а витрати на створення програмного забезпечення - 287131,6 грн.

3.3 Розрахунок річних експлуатаційних витрат

Витрати на річну експлуатацію становлять витрати на оплату хостингу для веб-додатку.

Проаналізувавши список компаній, що надають послуги хостингу, в середньому на рік витрати будуть складати 3600 грн на рік.

3.4 Маркетинговий аналіз

Проаналізувавши ринок, програм для моделювання систем, можна зробити висновок, що запропоноване програмне забезпечення не має аналогів на ринку. Існують програми для моделювання фізичних систем, бізнес процесів, стратегій, графічного дизайну програм і так далі. Але системи для моделювання криптографічних протоколів, з низьким порогом входження не

має. Тому був зроблений список ПЗ, для моделювання фізичних систем та явищ.

Таблиця 3.1 – програмне забезпечення для моделювання явищ та систем.

Назва	Ціна
Stella Online	285 грн на місяць
Lucid Chart	253 грн на місяць
Maple	236 грн на місяць
Matlab	15675 грн

Приведені вище системи мають широкий спектр можливостей для моделювання систем, але не підтримують моделювання криптографічних протоколів.

Можна зробити висновок, що найпопулярнішою моделлю розповсюдження є модель із підпискою. Для запропонованого ПЗ також будемо використовувати подібну систему розповсюдження, бо вона є найоптимальнішою для користувачів та власника програмного забезпечення, бо не потребує разової покупки ліцензії за велику суму, та надає стабільний дохід для власника ПЗ.

Ціною за місяць підписки, опираючись на вартість подібних систем, з урахуванням вузькопрофільності ПЗ, в подальших обчисленнях буде обрано 140 грн.

3.5 Визначення економічного ефекту

Будемо вважати, що кожен місяць аудиторія користувачів збільшується на 40, з урахуванням, що проектом перестають користуватися деякі старі

користувачі. Виходячи з цього, прибуток від ПЗ, можливо розрахувати по формулі:

$$П_n = (\sum_{m=0}^{n} 40 * m) * 140 - 287131{,}6 - \left(\frac{3600}{12}\right) * n \qquad (3.14)$$

Де:

— n – кількість місяців від старту продажів ПЗ. 40

Керуючись формулою 2.14, вирішимо нерівність:

$$(\sum_{m=0}^{n} 40 * m) * 140 - 287131{,}6 - \left(\frac{3600}{12}\right) * n > 0$$

$n > 9$ місяців.

Отримуємо, що для отримання прибутку, за обраною формулою з обраними змінними проект повинен працювати 10 місяців або 0,83 роки.

3.6 Висновки

Порівняльний аналіз відомих програм для моделювання показує, що усі вони є платними, та більшість із них, розповсюджуються по моделі із підпискою.

Також під час пошуку програм моделювання не було знайдено жодної для моделювання криптографічних протоколів. Що робить ПЗ розглянуте у дипломній роботі унікальним. Для тестування протоколу без розглянутої системи, потрібно або виконувати усі розрахунки вручну, або одразу програмувати криптографічний протокол.

Вартість розробленого програмного забезпечення – приблизно 287131,6 грн. Що складає не велику суму для программного продукту, проте запропонує своїм користувачам широкий спектр прикладних можливостей.

ВИСНОВКИ

У дипломній роботі розв’язано актуальне наукове завдання щодо розробки програмної системи для аналізу криптографічних протоколів. В ході розв’язання поставлених задач були отримані наступні наукові та практичні результати:

1. Проведено аналітичний огляд криптографічних протоколів, їх властивостей безпеки, класифікації та учасників;
2. Описані атаки на криптографічні протоколи, методи аналізу та способи моделювання;
3. Запропоновано уточнений спосіб описання криптографічних протоколів, с більш детальним описом їх шагів та циклів;
4. Введене нове визначення поняття «крок протоколу», яке дає можливість представити протокол у вигляді, що дозволяє змоделювати його в системі дискретно-подієвого типу;
5. Розроблена концепція та архітектура програмної системи на основі моделі Долева-Яо;
6. Сформовані вимоги до системи моделювання протоколів;
7. Розроблено прототип пілотної версії програми на основі запропонованої архітектури. Представлені рекомендації стосовно розробки повноцінної фінальної версії системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1 Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке / Шнайер Б. – Москва : ТРИУМФ, 2002.

2 Шнайер Б. Cryptographic Design Vulnerabilities [Електронний ресурс] / Шнайер Б. – Режим доступу: https://www.schneier.com/academic/paperfiles/paper-design-vulnerabilities.pdf.

3 Варновский Н.П. "Протоколы = распределенные алгоритмы + соглашения о форматах" [Електронний ресурс] / Варновский Н.П. – Режим доступу: http://old.computerra.ru/1998/260/194242/.

4 Гатчин Ю.А. Основы криптографических алгоритмов : навчальний посібник / Гатчин Ю.А., Коробейников А. Г. – СПб : ГИТМО, 2002.

5 Давидов А.Н. Формальный анализ криптографических протоколов: методы, основанные на моделях конечных автоматов / Давидов А.Н. / Безопасность информационных технологий : научно-техническая конференция (2005 г., Пенза). – Пенза, 2005.

6 Деднев М.А. Защита информации в банковском деле и электронном бизнесе. / Деднев М.А., Дильнов Д.В., Иванов М.А. – Москва : КУДИЦ-ОБРАЗ, 2004.

7 Запечников С. В. Криптографические протоколы и их применение в финансовой и коммерческой деятельности / Запечников С. В. – Москва : 2007.

8 Сапегин Л.Н. Анализ криптографических протоколов. Специальная техника средств связи. / Сапегин Л.Н. – Пенза : ПНИЭИ, 1996.

9 Сапегин Л.Н. Типичные дефекты в криптографических протоколах. Специальная техника средств связи / Сапегин Л.Н. – Пенза : ПНИЭИ, 1996.

10 Смарт Н. Криптография / Смарт Н. – Москва : Техносфера, 2005.

11 Могилевская Н.С. Формализация и анализ протоколов аутентификации / Могилевская Н.С., Новиков А.М. // Информационное противодействие угрозам терроризма. – 2009. – №12. – С. 99-104

12 Михайлов А.С. Архитектура сетевого программного комплекса для анализа криптографических протоколов / Михайлов А.С., Першиков А.Г., Тарасов И.Л. // Научная сессия МИФИ-2001 – Москва : МИФИ, 2001.

13 Косачев Л.С. Анализ подходов к верификации функций безопасности и мобильности [Електронний ресурс] / Косачев Л.С., Пономоренко В. П. – Режим доступу: http://www.ispras.ru/preprints/docs/prep_15_2006.pdf

14 Панасенко С.В. Протоколы аутентификации. Безопасность [Електронний ресурс] / Панасенко С.В. – Режим доступу: https://www.bytemag.ru/articles/detail.php?ID=9059

15 Черемушкин А. В. Криптографические протоколы. Основные свойства и уязвимости : учебное пособие / Черемушкин А. В. – Москва : Изд. центр «Академия», 2009.

16 Черемушкин А. В. Криптографические протоколы: основные свойства и уязвимости. Прикладная дискретная математика : учебное пособие / Черемушкин А. В. – Москва : Изд. центр «Академия» , 2009.

17 Чернышев М.Е. Визуальное моделирование протоколов информационного обмена / Чернышев М.Е. // Научная сессия МИФИ-2006 – Москва : МИФИ, 2006.

18 Фергюсон Н. Практическая криптография / Фергюсон Н., Шнайер Б. – Москва : Издательский дом "Вильяме", 2005.

19 Методичні рекомендації до підготовки та захисту дипломної роботи (проекту) для студентів галузі знань 1701 «Інформаційна безпека» та спеціальності 125 «Кібербезпека» / Т.В. Бабенко, М.В. Корнєєв, О.В. Кручінін, Д.С. Тимофєєв ; Нац. гірн. ун-т. – Д. : НГУ, 2016. – 44 с.

20 Методичні вказівки до виконання економічної частини дипломного проекту (для студентів напряму підготовки 1701 Інформаційна безпека)/ Упорядн.: О.Г. Вагонова, Ю.О. Волотковська, Н.М. Романюк. – Дніпропетровськ: ДВНЗ "Національний гірничий університет", 2013. – 17 с.

ДОДАТОК А

Перелік матеріалів на оптичному носії

1 Титульний_лист.docx;

2 Завдання.docx;

3 Реферат.docx;

4 Зміст.docx;

5 Вступ.docx;

6 Розділ_1.docx;

7 Розділ_2.docx;

8 Розділ_3.docx;

9 Висновки.docx;

10 Література.docx;

11 Додаток_А.docx;

12 Додаток_Б.docx;

13 Додаток_В.docx;

14 Додаток_Д.docx;

15 Додаток_Е.docx;

16 Додаток_Ж.docx;

17 Додаток_З.docx;

18 Презентація.pptx;

19 Директорія CrypticStuff, що містить вихідний код пілотного проекту.

ДОДАТОК Б

Методи формальної верифікації [13]

1. Верифікація на основі кінцевих автоматів

Для проведення аналізу криптографічних протоколів з використанням кінцевих автоматів використовується методика, відома під назвою методика аналізу досяжності.

Ця методика передбачає опис системи в наступному вигляді. Для кожного переходу будується глобальний стан системи, яке виражається через стану сутностей системи і стану комунікаційних каналів між ними. Кожний глобальний стан потім аналізується, і визначаються його властивості, такі як глухий кут і коректність. Якщо сутність не здатна отримати повідомлення, а передбачалося, що воно повинно було бути отримано в цьому стані, тоді існує проблема в протоколі. Методика аналізу досяжності ефективна для визначення коректності протоколу по відношенню до його специфікаціям, проте вона не гарантує безпеки від активного зловмисника.

В інших роботах абстракція кінцевих автоматів використовується для верифікації властивості «не впливу». Автори розробили сукупність умов, що розкручуються, якої є достатньо для встановлення властивості «не впливу» N1в кінцевих автоматах. Незважаючи на те, що ці умови відносно прості, їх застосування залежить від моделі кінцевого автомата для даної системи. Ця верифікаційна методика допомагає зробити властивість N1 такою ж корисною на практиці, як і BLP. Хоча верифікація N1, взагалі кажучи, може бути важче, ніж верифікація BLP, в першій відсутній аналіз прихованих каналів, який необхідний після верифікації BLP.

Також описується автоматичний завершений метод для верифікації властивостей конфіденційності криптографічних протоколів. Метод заснований на надійній абстрактній інтерпретації криптографічних протоколів, в якому використовується розширення деревовидних автоматів, а саме V-параметрезованих деревовидних автоматів, які змішують автоматотеоретичні методики з особливостями дедуктивних методик. Всупереч більшості підходів,

заснованих на перевірці моделі, цей метод пропонує фактичні гарантії захисту. Описується можливість аналізу протоколів в присутності паралельних многосеансних принципалів.

2. Перевірка моделі

Метод перевірки моделі вперше був запропонований на початку 80-их. У останній час він стає індустріальної практикою і широко використовується в практичних додатках, особливо для перевірки апаратури і комунікаційних протоколів, для аналізу гібридних систем.

Метод полягає в специфікації властивостей системи на мові алгебри процесів або у вигляді формул темпоральної логіки, побудові моделі у вигляді кінцевого автомата, автоматичної перевірки того, що ця модель задовольняє специфікації, причому при негативному висновку виробляється контрприклад. Метод перевірки моделі полягає в переборі всіх можливих переходів автомата з одного стану в інший з якогось початкового стану системи. Всі можливі траси з початкового набору станів перебираються, щоб переконатися в тому, що всі вони є безпечними, або довести, що живучість системи не може бути порушена. Метод перевірки моделі страждає недоліком, широко відомим під назвою "вибух кількості станів". При моделюванні часу як безперервної суті навіть найпростіша модель має нескінченне число станів. Найбільш відомим підходом до вирішення цієї проблеми є метод символічної перевірки моделі, в якому проблема спрощується за допомогою формування класів еквівалентності. При цьому використовуються компактні булевські форми для подання наборів станів і переходів, як, наприклад, BDDs (Binary Decision Diagrams), і накладаються деякі обмеження на структуру простору станів. В області комунікаційних протоколів проводилися численні дослідження для аналізу умов, при яких кінцеве число станів при перевірці моделі стає достатнім. По всій видимості, найкращих результатів досягли дослідження, в яких наводиться набір умов, при яких перевірка невеликої кількості сесій стає достатнім для доказу секретності ключа. Успішне застосування методики перевірки моделі до

аналізу протоколів захисту показали роботи, в яких використовувалися аналізатори моделей універсального застосування. Широкі можливості методики перевірки моделі привели до створення деяких спеціалізованих для протоколів інструментів, заснованих на цьому підході, і використання спеціалізованих інструментів, що спочатку призначалися для інших додатків.

Багато дослідників в цій області зосередили свою увагу на розробці методу перевірки моделі, в якому передбачається обмежена кількість сесій, але при цьому присутній доказ повноти, причому не накладаються обмеження на складність повідомлення, наприклад, на глибину шифрування. У роботах показано, що властивість BNDC вирішується для процесів з кінцевим числом станів, і пропонується метод верифікації, заснований на часткової перевірки моделі. Однак проблема ефективного способу верифікації BNDC поки ще залишається відкритою, також як і проблема можливості розв'язання BNDC. Вирішення цих проблем можна знайти на шляху адаптації достатніх умов для BNDC.

Також досліджуються властивості безпеки інформаційного потоку на основі бісімуляції. Ці властивості є постійними в тому що специфікації протоколів, зазвичай мають нескінченну кількість станів через можливість зловмисника генерувати нескінченну кількість повідомлень, є моделями, на яких була визначена здійсненність логічних формул. Як логіки використовується логіка, певна для аналізатора моделей, яка дозволяє висловити широкий клас властивостей безпеки, таких як конфіденційність, цілісність, справжність, деяка слабка форма анонімності і загальних властивостей надійності. Забезпечення процедури здійсненності на моделях, отриманим з SPID-специфікацій протоколів, є основним досягненням цієї роботи. Планується розробка інструменту для автоматичної перевірки моделі на основі цієї концепції. Для цього необхідний перехід до кінцевих моделям. В якості першого рішення на цьому шляху дається обмеження на довжину повідомлень, які зловмисник може генерувати.

У деяких роботах також досліджується автоматична перевірка моделі, заснована на зведенні проблем ненадійності протоколів до проблем здійсненності в пропозиціональній логіці (SAT), яку можна використовувати для виявлення атак на протоколи захисту. Підхід виник з комбінації відомості проблем ненадійності протоколів до проблем планування і відомих методик SAT-редукції, званих лінійним шифруванням, розроблених для планування. Експериментальні результати підтвердили ефективність підходу, але також показали, що час, витрачений на генерацію SAT-формули, значно перевищує час, який необхідно SAT-вирішального пристрою для перевірки її здійсненності. Крім того, згенеровані SAT-шаблони мають некерований розмір на найскладніших протоколах. Досліджується додаток методики шифрування, заснованої на Ciraphplan, до аналізу протоколів захисту і представляються експериментальні дані, що показують, що шифрування, засноване на Oraphplan, значно простіше лінійного кодування. Ці результати підтверджують ефективність SAT-підходу до аналізу протоколів захисту і прокладають шлях до його додатків для аналізу великих протоколів, які існують у практичних додатках.

3. Доведення теорем

Статичний аналіз програм можна використовувати для управління інформаційним потоком з великою точністю і малими накладними витратами. Статичне визначення характеристик інформаційного потоку було реалізовано за допомогою програм для доведення теорем. Успішне застосування методики доведення теорем до аналізу протоколів захисту показали дослідження, про які повідомляється в роботах Паульсона із застосуванням інструменту Isabelle (на основі методу доведення теорем) для аналізу досить складних протоколів.

Хоча за своєю природою застосування методу доведення теорем вимагає більшої інтерактивності, ніж метод перевірки моделі, Паульсон виробив сукупність теорем і методик, яка може перевикористати в інших аналізах за допомогою Isabelle.

В одній роботі вводиться визначення біосимуляції для криптографічних протоколів. Це визначення включає просту і точну модель знань про середовище, в якій діє протокол. Біосимуляція є основою ефективної методики доказів, яка здійснює доказ класичних властивостей безпеки, а також встановлює деяку оптимізацію протоколу. Доводиться семантична несуперечливість біосимуляційної методики доказів всередині spi-обчислення. Розроблено також спеціалізовані алгоритми і інструменти, засновані на методиці доведення теорем, які налаштовані на аналіз криптографічних протоколів - це система TAPS і система Хуіми.

Спеціалізовані системи вимагають значно меншого взаємодії з користувачем, ніж універсальні, а так забезпечують більше покриття, ніж аналізатори моделей. Однак, на відміну від аналізаторів моделей, вони не виробляють контрприкладів, якщо доказ властивостей безпеки призводить до помилкових висновків. Використовуються розвиваються умови для визначення системи доказів для процесів, що мають властивість постійної BNDC. Ця система доказів забезпечує потужну методику для верифікації та розробки таких процесів. Справді, ця система доказів дозволяє верифікувати, чи є процес безпечним, простим інспекцією його синтаксису, що дає можливість уникнути проблеми вибуху станів. Зокрема, вона допускає рекурсивні процеси, які можуть виконувати необмежені послідовності дій, при цьому, можливо, досягаючи нескінченного числа станів.

4. Метод перевірки типу

Найновішим підходом до формального аналізу протоколів є використання методу перевірки типу (type checking), що ввівм Абаді. Абаді ввів тип untrusted (Un) для відкритих повідомлень, які виходять від опонента (в якості опонентів виступають всі, крім засвідчують принципалів).

У методі перевірки типу повідомленнями і каналам присвоюються типи (наприклад, одиниця даних приватного типу з'являється на відкритому каналі). Метод перевірки типу має істотну перевагу, що складається в тому, що він, як

метод перевірки моделі, є повністю автоматичним, але на відміну від останнього здатний оперувати з кількома класами нескінченних систем. Однак він має потенційний недолік, який полягає в тому, що, так як порушення безпеки визначені в термінах неузгодженості типу, вимоги безпеки, які повинні бути доведені, повинні бути сформульовані в специфікації в процесі її написання. Це відрізняє метод перевірки типів від методу перевірки моделі, для будь-якої властивісті безпеки, яке може бути виражено в термінах темпоральної логіки, може специфіковані незалежно, вже після того, як сам протокол специфікований.

В іншій роботі пропонується метод для перевірки властивостей аутентифікації криптографічних протоколів, який заснований на двох ідеях - By і Лема, про твердження відповідності для специфікації властивостей аутентифікації і ідеї Абаді про верифікації властивостей протоколу за допомогою перевірки типів. Описується формальна семантика тверджень відповідності, але не пропонує методу верифікації протокола. Протокол специфікується за допомогою типизованого розширення spi. Властивості аутентифікації виражаються у вигляді тверджень в стилі By і Лема і записуються як коментарі. Обчислюються типи для ключів, унікальні ідентифікатори з'єднання) і повідомлень протоколу, і потім проводиться перевірка типів за методом Абаді. Формальна операційна семантика для spi-обчислення описується за допомогою семантики трас, заснованої на Chemical Abstract Machine.

Метод перевірки типу для статичного управління інформаційним потоком був реалізований в компіляторі Jaff. Кожен вираз в програмі має тип безпеки, який складається з двох частин-звичайний типу (наприклад, int) і мітки, яка описує, як можна використовувати значення. Безпека гарантується перевіркою типів-компілятор читає програму, яка містить розмічені типи, і перевірка типів гарантує, що програма не міститиме невідповідні інформаційні потоки під час виконання. Система типів в такій мові є системою типів безпеки, яка здійснює політику, засновану на інформаційних потоках.

В іншій роботі розглядається система типів, в якій всі змінні програми класифікуються як L (відкритий) або I (приватний). Така типізація перешкоджає витоку інформації про змінних H в L змінні. У багатопотоковій імперативній мовою з імовірнісним плануванням це формалізує властивість імовірнісного «невпливу». Стверджується, що з такою типізацією можна запобігти витоку інформації, пов'язані з часом, якщо зажадати, щоб ніяке присвоювання змінної не могло слідувати за командою, тривалість якої залежить від n-змінних. У такій системі з'являється можливість використовувати n-змінні більш гнучко; наприклад, потік, в якому обробляються тільки n-змінні, завжди добре.

5. Інші підходи

Опублікована методика автоматичної верифікації криптографічних протоколів, заснована на проміжному поданні протоколу за допомогою набору фраз Хорна (логічна програма). Ця методика дає можливість верифікувати властивості безпеки протоколів, таких як конфіденційність і аутентифікація, повністю автоматичним способом. Крім того, одержувані з її допомогою докази є правильними для необмеженого числа сеансів протоколу.

Опублікований метод обґрунтування на основі слабшої передумови. Ця методика розглядає три компоненти: стан до виконання інструкції програми, інструкція програми безпосередньо, і мета, яка повинна бути істинною після того, як інструкція виконується. Недолік цієї методики полягає в труднощі докази для складних предикатів. Для довгих програм, з великою кількістю цілей, докази можуть бути неможливі.

В роботі, присвяченій інтегрованому середовищу CPAL-ES, для верифікації застосовується метод «слабшого передумови». Оскільки криптографічні протоколи мають тенденцію бути короткими, цей метод успішно застосовується для цих протоколів.

ДОДАТОК В

Огляд методів формальної специфікації криптографічних протоколів

1 Специфікаційні мови

1.1. Універсальні специфікаційні мови

Мова формальної специфікації InaJo. Заснована на розширенні обчислення предикатів першого порядку. Ця мова тверджень був розроблений як універсальний засіб для підтримки розробки ПЗ і докази його коректності. За допомогою критеріїв InaJo описуються критичні вимоги, яким система повинна задовольняти у всіх станах (наприклад, що ніякої ключ не повинен бути доступний зловмисникові, за допомогою якого він зміг би здійснити дешифрування).

На основі створених специфікацій InaJo виробляє теореми, які потім використовуються для верифікації критичних вимог. Переваги опису системи в формальної нотації і подальшого доведення властивостей щодо цієї специфікації складаються в тому, що якщо згенеровані теореми не можуть бути доведені, це часто вказує на слабкі місця системи або на неповноту специфікацій.

Однак, значення цього методу обмежується тією обставиною, що доказ критеріїв про специфікації Ina Jo не може гарантувати безпеку протоколу. До того ж, щоб специфікувати вимоги до безпеки системи, необхідно вміти описувати потенційні атаки.

Чен і Глікор стверджують, що багаторівнева мережева безпека може бути досягнута і формально верифікована незалежно від специфічних протоколів транспортного рівня, навіть в присутності зловмисника в мережі. Це досягається за допомогою застосування багаторівневого протоколу безпечної сесії і протоколу розподілу ключів, які допомагають підтримувати конфіденційність і цілісність повідомлень в незахищеної мережі.

LOTOS. LOTOS був розроблений для систем, що відносяться до Open Systems Interconnection (OSI) і в його основі лежать алгебри процесів. Система в LOTOS моделюється як набір процесів, в якому вказується порядок подій. LOTOS може бути використаний для моделювання процесу обміну повідомленнями в криптографічних протоколах.

Запропоновано використовувати LOTOS для аналізу криптографічних протоколів. Він наводить приклади специфікації двох протоколів, прийнятих в якості стандартів ISO / DP 9798 і CCLTT X.509. Він застосований в методі, заснованим на моделі, в якому протокол специфікується за допомогою алгебри процесів, заснованої на CSP.CCS. Криптографічні операції моделюються на абстрактному рівні як продукт поведінки і правил та розуміються як властивості безпеки, і їх можна легко перевести в специфікацію. Кожен принципал специфікується як сутність, яка має поведінка і абстрактний тип даних для відображення дій, повідомлень і знань. Процес взаємодії здійснюється через синхронізуючі шлюзи між принципалами, і зловмисник явно збожеволіє між двома принципалами. Зловмисник моделюється під час кожної синхронізації разом зі своїми характер пішаками (поведінку, знання, старі простежені повідомлення). Всі специфікації пишуться на LOTOS, потім вони транслюються в систему розмічених переходів для верифікації.

Система заснована на моделі верифікація є досить калиткою для знаходження дефектів в протоколі, але з її допомогою неможливо довести повну коректність через проблеми вибуху станів.

Розглядається застосування LOTOS для специфікації протоколів захисту і криптографічних операцій. Описується, як властивості безпеки можуть бути модельовані через властивості надійності і як метол верифікації на основі моделі використовується для верифікації стійкості протоколу щодо атак зловмисника.

ASTRAL - специфікаційна мова універсального призначення, яка використовується для специфікації систем реального часу. Створений на його основі аналізатор моделей ASTRAL, був застосований до аналізу криптографічних протоколів. Про застосування інших специфікаційних мов для аналізу криптографічних протоколів повідомляється в декількох роботах.

1.2. Мови специфікації спеціального призначення

CAPSL. Мова CAPSL був розроблений Мілном і позиціонується як специфікаційні мову для криптографічних протоколів. Ця мова має близьку подібність зі стандартною нотацією протоколу, тому він легко читається, і на ньому легко писати. CAPSL має проміжну форму, яка заснована на логіці переписування і забезпечує формальну семантику CAPSL. CAPSL позиціонується як єдина мова специфікації протоколів, який може використовуватися як вхідний формат для будь-якої методики формального аналізу. CAPSL використовується досить широко. Його формальна семантика визначається засобами перед- і постумовою.

NPATRL. Сіверсон і Мідоуз розробили мову вимог для аналізатора NRL Необхідно було розробити простий темпоральний мову, який може застосовуватися для специфікації вимог, які зазвичай використовуються в протоколах аутентифікації і протоколах розподілу ключів. Атомарні компоненти мови відповідають подіям в протоколі (наприклад, посилка і отримання повідомлень, або вивчення зловмисником якогось терма). Крім звичайних логічних зв'язок, мова містить тільки один темпоральний оператор, "сталося раніше". Використання цього єдиного логічного оператора відображає той факт, що більшість вимог пересилання повідомлень може бути виражено в термінах подій, які повинні або не повинні відбутися перед деякими іншими подіями.

ISL. Мова Interface Specification Language (ISL), яка служить інтерфейсною мовою для інструменту Automatic Authentication Protocol Analyzer.

BMSL. Мова специфікації поведінкового моніторингу дає можливість створювати короткі специфікації властивостей безпеки, засновані на подіях. Ці властивості можуть відрізняти будь-яка нормальна поведінка програм і систем від неправильної поведінки, пов'язаного з можливою атакою.

Заснований на XML мова специфікації безпеки. Вона призначена для документів Веб-сервісів. Підхід заснований на застосуванні рольової моделі контролю доступу (RBAC). Модель RBAC розширюється можливістю специфікації політики контролю доступу, а саме додаються поняття ієрархії ролей і поділу обмежень доступу. Передбачається робота тільки з XML-документами. Дається можливість специфікувати доступ до контенту на концептуальному рівні, а також на рівнях схеми, примірника і елемента документа. Крім того, запропоноване розширення моделі забезпечує можливість аналізу контексту, в якому робиться спроба доступу. Заснована на XML мова специфікації контролю доступу описується в термінах мандатів користувачів, ролей і повноважень.

2. Алгебри процесів

Алгебри процесів (PA - Process algebras) - добре відома рамкова концепція, яка представляє сімейство обчислень для опису розподілених і паралельних систем. Для опису властивостей безпеки запропоновано велику кількість розширень відомих алгебр процесів CCS (Calculus of Communicating Systems) і CSP (Communicating sequential processes). У більшості випадків застосування таких обчислень до протоколів захисту, спирається на безлічі примітивів, які виражаються в термінах мови алгебри процесів.

Протоколи захисту можуть бути описані за допомогою деякого підмножини мови PA, де кожне взаємодія відбувається через мережу ($P_{ПС}$, - процес, керуючий мережею як відкритим каналом, де кожен P_p посилає і приймає повідомлення).

В системі присутній зловмисник з деякими початковими знаннями, здатний перехоплювати і фальсифікувати повідомлення, що передаються по мережі. Кожен принципал стартує протокол, граючи певну роль p.

Протокол захисту, що включає набір ролей p, виражається в підмножині мови PA як процес Q_n що складається з декількох компонент опису поведінку мережі. При цьому копіюються повідомлення з каналу N, (вхід мережі) в N_0 (вихід з мережі), реалізуючи асинхронну форму передачі повідомлень-процесів представляють послідовність дій, які визначають роль. Це специфікація моделі зловмисника в стилі Долева-Яо. Кожен P описує одну здатність зловмисника. Спеціалізований канал зберігає інформацію, якою оперує зловмисник (вона може бути початковою, перехопленої, або фальсифікованої).

Першою публікацією про застосування алгебри взаємодіючих процесів CSP для аналізу протоколів була робота Роско і стала більш розвинутою в інших роботах.

3. Мультимножина підстановка

Мультімножественная підстановка (MSR - Multiset Rewriting) сягає своїм корінням в теорію паралельних процесів і логіку підстановок. Цей формальних підхід широко застосовується для вивчення фундаментальних питань протоколів захисту. Він також грає практичну роль в якості мови проміжного рівня CIL в системі для аналізу протоколів CASPL.

Мова мультімножественной підстановки першого порядку визначається спеціальною граматикою і спирається на наступні предикатні символи:

— Мережеві повідомлення - це предикати для моделювання мережі;
— Стану ролей - це предикати для моделювання ролей;
— Зловмисник - це предикати для моделювання зловмисника;
— Постійні предикати - це базові предикати для зберігання даних, які не змінюються під час розгортання протоколу. Правила використовують ці предикати для доступу до значень постійних даних.

Протокол захисту виражається як безліч правил підстановок спеціального формату, званого теорією протоколів захисту.

Будується дерево, в якому кожен вузол являє елемент даних, а діти вузла представляють ті елементи даних, які потрібні для знання про дані, представлених в батьківському вузлі. Таким чином, можна побудувати дерево, в якому кореневий вузол являє дані, необхідні зловмисникові для атаки (наприклад, криптографічний ключ), а листя представляють елементи даних, які потрібні для кореневого елемента. Відповідний інструмент дозволяє користувачеві взаємодіяти з системою. Користувач може визначити, може чи не може елемент даних бути знайдений зловмисником. Якщо робиться висновок, що такий елемент даних може бути доступний, інформація вставляється в систему, і процес триває. За допомогою такого підходу були знайдені тонкі і раніше невідомі дефекти в ієрархічній схемі управління ключами.

4. Кінцеві автомати

Кінцеві автомати (або системи розмічених переходів) можуть бути визначені різними способами. . Звичайний кінцевий автомат описується:

— Безліччю станів;
— Початковим станом;
— Кінцевим безліччю впливів;

— Ставленням переходів.

Специфікація протоколів за допомогою теорії кінцевих автоматів, використовує діаграми станів. Для уявлення кожного принципала використовується орієнтований граф. Специфікує початковий стан, потім проводиться дуга до іншого стану для кожного повідомлення, яке може бути послано або отримано з попереднього стану. Таким чином, робиться спроба описати поведінку принципалів в протоколі.

В системі Interrogator, запропонованої Мілленом і ін., Учасники протоколу моделюються як взаємодіючі кінцеві автомати, чиї повідомлення перехоплюються зловмисником, який здатний читати, знищувати, або модифікувати повідомлення. Стан, в якому зловмисник знає якесь слово, яке повинно бути секретним, приймається як кінцевий стан, і з нього Interrogator намагається відтворити маршрут в просторі станів, за яким можна було прийти в це кінцевий стан. Якщо такий шлях знаходиться, це означає, що існує дефект в безпеки протоколу.

Система NRL Protocol Analyzer, запропонована Мідоуз, схожа на Interrogator. Специфікується небезпечний стан і Analyzer намагається відтворити шлях до цього стану з початкового стану. На відміну від Interrogator'а дозволяється з'єднувати необмежену кількість виконань протоколу в одному маршруті. Це дозволяє виявляти атаки, які засновані на здатності зловмисника з'єднувати кілька різних прогонів протоколу разом. Крім того. Analyzer не тільки знаходить шляхи до небезпечних станів, але з його допомогою можна доводити, що ці стани недосяжні. Стає можливим доводити, що деякі шляхи, що ведуть назад з небезпечних станів, призводять до нескінченних петель, з яких неможливо потрапити в початковий стан. При видаленні цих шляхів результуючий простір стає досить малим для проведення ефективного пошуку. Однак доказ того, що шлях веде до нескінченних петель, в основному лягає на користувача, і таким чином, пошук в Analyzer'е менш автоматизовано, ніж в Interrogator.

Інтерактивні кінцеві автомати ISMs. ISMs є автоматами, в яких переходи між станами можуть допускати багаторазові вхідні і вихідні впливу одночасно на будь-якій кількості портів. Ключовими концепціями ISMs є стану (і особливо переходи між ними) і взаємодія (interaction). Під взаємодією мається на увазі явна буферизована комунікація через іменовані порти (які також називають підключеннями), де на кожному порту один одержувач слухає, можливо, багатьох відправників. Кожен ISM має єдине локальне початковий стан.

Кожен ISM оголошує два набору імен портів, для введення і для виведення. Вхідні буфера - сімейство черг (необмежених) повідомлень типу FIFO, яке індексується іменами портів. Обмін повідомленнями викликається будь-яким ISM, що посилає повідомлення в межах системи або в навколишнє середовище. Вхідні впливу не можуть блокуватися, т. Е. Вони можуть відбутися в будь-який час, при цьому отримане значення додається до відповідної FIFO. Значення, збережені у вхідних буферах ISM, можуть бути оброблені ISM, коли ISM буде готовий зробити це. Це робиться за допомогою переходів, визначеними користувачем, які можуть бути не детермінованими і можуть бути специфіковані в будь-якому реляційному стилі. Правила переходу визначають, що (відповідно до деякого передумовою, до якої можуть входити порівняння на відповідність повідомлень у вхідних буферах) ISM переробляє всі вхідні впливу зі своїх буферів, робить локальний перехід і виробляє певний висновок. Висновок відправляється у вхідні буфера всіх ISMs, які слухають на відповідних портах, які можуть привести до прямої або непрямої зворотного зв'язку. Виконання ISM або системи ISM - будь-яка кінцева послідовність конфігурацій, досяжних з початкової конфігурації. Переходи різних ISMs, які виконуються паралельно, пов'язані тільки причинного зв'язком через обмін повідомленнями. Виконання застряє, коли немає ніякого компонента, який може виконати який-небудь крок.

Також пропонуються до використання та інші види кінцевих автоматів:

— Алгебраїчні кінцеві автомати;

— Деревовидні автомати;

— Групові автомати.

5. Модальні логіки

Модальні логіки набули широкого поширення в області специфікування криптографічних протоколів. Застосовуються епістемістичні, темпоральні і дохастичні логіки. Дохастична логіка заснована на довірі, і є системою висновків, яка застосовує правила про те, як вивести нове довіру на підставі наявного довіри. Епістемістична логіка нагадує дохастичну, з тією відмінністю, що вона заснована на знанні.

Такі логіки складаються з мови, який описує різні твердження про довіру і знанні принципалів щодо повідомлень, і деяких правил виведення, які використовуються для виведення нових тверджень з попередніх. Метою аналізу є вивести твердження, яке буде представляти коректне умова протоколу. Неможливість виведення такого твердження означає, що протокол, можливо, не є коректним.Особливо широкого поширення набули логіки довіри після появи логіки BAN.

Логіка BAN, для аналізу протоколів, є логікою довіри. Логіка BAN дозволяє описувати криптографічні протоколи за допомогою формальних термінів і міркувати про стан довіри між принципалами в системі. Ця логіка складається з набору модальних операторів, що описують взаємозв'язку між принципалами і даними, набору можливих взаємних довір принципалів (наприклад, переконання, що повідомлення було послано якимось іншим принципалом) і деякого набору правил виведення для отримання нових довір зі старих. Ось неформальний приклад логіки BAN: "Якщо A переконаний, що A отримав повідомлення, зашифроване за допомогою ключа K, і A переконаний,

що тільки B і A знають K, тоді A переконаний, що повідомлення було створено або B, або A".

Логіка BAN складається з простого інтуїтивного набору правил, що сприяє її широкому розповсюдженню, і призвело до сукупності логік, чи будуть розширювати логіку BAN, або застосовують ту ж саму концепцію до різних типів проблем в криптографічних протоколах.

6. Мережі Петрі

Них і Таварес запропонували використовувати мережі Петрі для формального моделювання та аналізу криптографічних протоколів. Зокрема, вони застосували розфарбовані мережі Петрі для моделювання протоколів. Їх модель включає модель зловмисника, яка може бути використана для опису атак зловмисника і для генерації тестових варіантів. Аналіз властивостей безпеки криптографічних протоколів заснований на вичерпному тестуванні на проникнення, при якому шукаються сценарії, які порушують деякі зазначені критерії. Такі критерії визначаються в термінах станів мереж Петрі для даного протоколу. Хоча розфарбовані мережі Петрі дозволяють створювати компактні і здійсненні опису протоколів, інструменти для підтримки ефективного виконання вичерпного пошуку все ще відсутні. Можна перетворити розфарбовані мережі Петрі в звичайні, і потім застосувати інструменти, що існують для звичайних мереж Петрі.

7. Графічне моделювання

Моделювання різних аспектів безпеки систем (включаючи багаторівневу безпеку, безпеку інформаційних потоків) і протоколів безпеки розглядається за допомогою мови UML. Ядро UML використовується для специфікації

контролю доступу в розподілених Java-системах. Показано, як визначити вимоги безпеки і довести, що моделюються механізми контролю доступу типу задовольняють вимогам безпеки, і що ці механізми є сумісними з повною функціональністю, яка вимагається від системи. Автори цього підходу вважають, що механізми контролю доступу, які надає JDK. 1.2, застосовувати практично досить складно (особливо, коли допускаються непрямі повноваження на доступ), тому дуже корисним здається використання UML. який забезпечує коректну специфікацію цих механізмів. Також пропонується розширення UML, так зване UMLsec, для опису властивостей безпеки.

ДОДАТОК В

ВІДГУК

на дипломну роботу (проект)

студента ***Сізинцев Микита Андрійович*** гр. 125М-16-1

(прізвище, ім’я)

на тему: ***Розробка та дослідженняз програмної системи для аналізу криптографічних протоколів***

Навіть після більш ніж двадцятирічних досліджень постійно з'являються теоретичні методи аналізу протоколів, на основі яких розробляється автоматизовані засоби аналізу безпеки протоколів. Дана обставина і визначає актуальність теми дипломної роботи.

Повнота розкриття теми достатня. Теоретичний рівень високий.

Практична значущість магістерської роботи полягає в тому, що розроблена структура і принцип програмної реалізації системи моделювання криптографічних протоколів, які дозволяють ефективно вирішувати верифікації цих протоколів.

Автор самостійно виконав роботу, сформулював мету, завдання дослідження, наукові положення і результати, виконав теоретичну і практичну частини роботи.

Оформлення виконано згідно вимог. Загальна та спеціальна грамотність висока.

Перевагами роботи є те, що було розроблено новий вид опису протоколу, концептуальна модель системи моделювання та програмна реалізація системи моделювання протоколів.

Потрібно відмітити окремі недоліки в оформленні роботи та нечітко сформульований формат команд для інтерпретатора.

Дипломна робота заслуговує оцінки "відмінно", а студент Сізинцев Микита Андрійович присвоєнню йому кваліфікації "професіонал з організації інформаційної безпеки".

Науковий керівник

_____	_____	_____
(посада)	(підпис)	(ініціали, прізвище)

“____”_____200__р.

ДОДАТОК Е

Відгук керівника економічного розділу

Керівник розділу доц. Волотковська Ю.О.

(підпис) (ініціали, прізвище)

РЕЦЕНЗІЯ

на дипломну роботу (проект)

студента ***Сізинцев Микита Андрійович*** гр. 125М-16-1
(прізвище, ім’я)

на тему: ***Розробка та дослідженняз програмної системи для аналізу криптографічних протоколів***

Навіть після більш ніж двадцятирічних досліджень постійно з'являються теоретичні методи аналізу протоколів, на основі яких розробляється автоматизовані засоби аналізу безпеки протоколів. Дана обставина і визначає актуальність теми дипломної роботи.

У дипломній роботі розв’язано актуальне наукове завдання щодо розробки програмної системи для аналізу криптографічних протоколів.

Наукова новизна полягає у тому, що було розроблено новий вид опису протоколу, концептуальна модель системи моделювання та програмна реалізація системи моделювання протоколів.

Проведено аналітичний огляд криптографічних протоколів, їх властивостей безпеки, класифікації та учасників. Описані атаки на криптографічні протоколи, методи аналізу та способи моделювання.

Практична цінність полягає у тому, що розроблена структура і принцип програмної реалізації системи моделювання криптографічних протоколів, може ефективно вирішувати верифікації цих протоколів.

Перевагами роботи є те, що було розроблено новий вид опису протоколу, концептуальна модель системи моделювання та програмна реалізація системи моделювання протоколів.

Потрібно відмітити окремі недоліки в оформленні роботи.

Дипломна робота магістра заслуговує оцінки "відмінно".

Рецензент

_____	_____	_____
(посада)	(підпис)	(ініціали, прізвище)

“____”_____200__р.