УДК 004.056

© 2016 р. Г.Г. Киричек, А.О. Овчар

Запорізький національний технічний університет, Запоріжжя

СИСТЕМА ЗАХИСТУ ДАНИХ ПРИ ПЕРЕДАЧІ В МЕРЕЖІ

В роботі аналізуються сучасні методи захисту передачі даних, їх недоліки та переваги і пропонується рішення, у вигляді системи опосередкованої передачі, яке виключає безпосередню взаємодію між клієнтами, забезпечуючи підвищення показників надійності та крипостійкості, що є актуальним в умовах мережної передачі даних.

This paper examines the current methods of protecting data, their advantages and disadvantages and proposed solutions in the form of indirect transmission, which excludes direct interaction between customers, enabling them to improve the performances of the reliability and crypto stability that is actual when of data transfer on the network.

Вступ

Проведення фінансових операцій з використанням Інтернету, замовлення товарів і послуг, використання кредитних карток, доступ до закритих інформаційних ресурсів, передача телефонних розмов вимагають відповідного рівня безпеки [1]. Існує безліч сучасних криптографічних алгоритмів, які відповідають певним вимогам безпеки [2]. В основі шифрування лежать два основних поняття: алгоритм та ключ. Оскільки важливе місце в системах шифрування приділяється секретності ключа, то основною проблемою подібних систем є генерація і передача ключа. Стійкість таких алгоритмів ґрунтується на складності обчислення зворотної функції до функції шифрування [3]. Як видно з деяких успішних атак на алгоритми шифрування, виникає необхідність в розробці ефективних методів безпечного обміну при передачі даних в мережі.

Метою роботи є розробка системи опосередкованої передачі даних, в результаті використання якої виключається клієнтська взаємодія, що сприяє зменшенню ризиків пов’язаних з перехопленням або спотворенням інформації під час її передачі в мережах.

1. Постановка задачі

Основними завданням даної роботи є розробка методу опосередкованої передачі даних в мережі, розробка алгоритму випадкового визначення засобів передачі (серверів) та розробка гнучкої зміни шифрувального алгоритму в системі обміну, а також проведення дослідження ефективності розробленого методу, визначення ймовірності несанкціонованого доступу до інформації, яка передається, та тестування загальної крипостійкості системи.

Як і більшість технічних розробок, розробка алгоритму безпеки передачі даних стикається з проблемами надійності, швидкості роботи та ефективності використання. Перевага програмних засобів захисту інформації в універсальності, гнучкості, надійності, простоті встановлення та здатності до модифікації і розвитку. Недоліки - обмежена функціональність мережі, використання частини ресурсів файл-сервера і робочих станцій, висока чутливість до випадкових або навмисних змін і можлива залежність від апаратних засобів. Результати проведених досліджень показали, що спеціалізовані програмні засоби захисту інформації від несанкціонованого доступу мають кращі можливості і характеристики, ніж вбудовані засоби мережних ОС [4]. Крім програм шифрування і криптографічних систем є й інші доступні зовнішні засоби захисту інформації.

В даній роботі пропонується реалізація методу, який полягає в забезпеченні безперервної зміни засобів передачі та алгоритмів шифрування повідомлень за випадковим законом. Таким чином зловмисник ніколи не зможе отримати інформацію де на даний час знаходиться пакет та, відповідно, і шифрувальний алгоритм.

Швидкість роботи передачі даних визначають продуктивність системи або лінії зв'язку. З початку використання оптоволоконних серед передачі швидкість передачі даних в мережі зросла в рази. Останнім рекордом стало значення в 255Тбіт/с, тому наявність декількох до-

даткових вузлів в мережі не призведе до великих затримок при передачі але може позитивно вплинути на захист самих даних.

Криптографічна стійкість – здатність криптографічного алгоритму протистояти криптоаналізу. Стійким вважається алгоритм, який для успішної атаки вимагає недосяжних обчислювальних ресурсів, великого обсягу перехоплених відкритих і зашифрованих повідомлень чи такого часу розкриття, по закінченні якого захищена інформація не є актуальною. У більшості випадків крипостійкість не можна математично довести, а можна тільки довести уразливість криптографічного алгоритму [5]. В основному застосовуються практично стійкі або обчислювально стійкі системи стійкість яких залежить від можливостей криптоаналітика [6].

Якщо взяти за одиницю крипостійкість будь-якого алгоритму шифрування то можна зробити висновок що, при використанні розробленого в даній роботі методу передачі даних, значення збільшення криптостійкості всієї системи передачі залежить від кількості використаних проміжних вузлів (серверів) між клієнтами в процесі передачі. Кожен з серверів при цьому використовує свій алгоритм шифрування, який змінюється за випадковим законом.

2. Засоби розробки

Оскільки для розробки системи необхідні обчислювальні потужності, проведено аналіз та обрано серверну операційну систему (ОС), яка зручна в використанні і не займає багато апаратних ресурсів - це віртуальні образи серверів з ОС Ubuntu Server 16.04.1 на процесорі AMD 64 та MAAS (Metal-as-a-Service), який призначено для швидкого і зручного розгортання Ubuntu конфігурацій на безлічі серверів з використанням технік, що використовуються в хмарних платформах. Але на відміну від них, виділення ресурсів на такому кластері відбувається на рівні фізичних серверів, а не віртуальних оточень. Інструменти типу MAAS дозволяють розгорнути цілий кластер за кілька хвилин.

C# – об'єктно-орієнтована та є мовою програмування з C-подібним синтаксисом, який найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів, делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення та коментарі в форматі XML. C# на відміну від C++ не підтримує множинне успадкування класів. C# стандартизований в ECMA (ECMA-334) і ISO (ISO / IEC 23270) [7]. Відомо як мінімум про три незалежних реалізації C#, які базуються на цій специфікації і знаходяться в даний час на різних стадіях розробки: mono; dotGNU і Portable.NET. Mono – проект зі створення повноцінного втілення системи .NET Framework на базі вільного програмного забезпечення.

Забезпечення безпеки названо першою з п'яти головних завдань Інтернету в програмі дій міжнародної ініціативи побудови Інтернету майбутнього (Future Internet Design, FIND). В роботі під інформаційною безпекою розуміємо стан захищеності інформаційної системи, включаючи інформацію і інфраструктуру самої системи [8]. Організація служб безпеки мережі вимагає ретельного опрацювання політики інформаційної безпеки, яка включає кілька базових принципів.

Існує два класи криптосистем – симетричні і асиметричні. У симетричних схемах шифрування (класична криптографія) секретний ключ шифрування збігається з секретним ключем дешифрування. Популярні стандартні симетричні алгоритми шифрування даних є DES та AES. AES забезпечує кращий захист, використовує 128-бітові ключі (працює з 192- і 256-бітними ключами) і має високу швидкість роботи, кодуючи за один цикл 128-бітний блок на відміну від 64-бітного блоку DES. В даний час AES є найбільш поширеним симетричним алгоритмом шифрування [2].

Одним з найбільш популярних криптоалгоритмів з відкритим ключем є криптоалгоритм RSA (Rivest, Shamir, Adleman), який повністю відповідає принципам Діффі-Хеллмана. Саме з величезною обчислювальною складністю розкладання великого числа на прості множники пов'язана висока крипостійкість алгоритму RSA [8]. Як видно з вище приведеного опису алгоритмів шифрування кожен з них має свої недоліки тому в сучасних криптографічних системах використовують обидва способи шифрування (заміни і перестановки). Такий шифратор називають складовим (product cipher). Він більш стійкий, ніж шифратор, використовує тільки заміни або тільки перестановки.

Спираючись на проведений аналіз вже існуючих методів інформаційної безпеки та обрані засоби розробки розглянемо етапи реалізації методу передачі даних, який забезпечить можливість уникати безпосередньої передачі секре-

тних даних в мережі та захистити дані в процесі їх передачі від перехоплення та перенаправлення трафіку.

3. Реалізація системи

Виходячи з основної мети проведення досліджень та спираючись на загальні завдання розробки, перед початком реалізації системи наведемо модель її функціонування в мережному середовищі (рис.1).

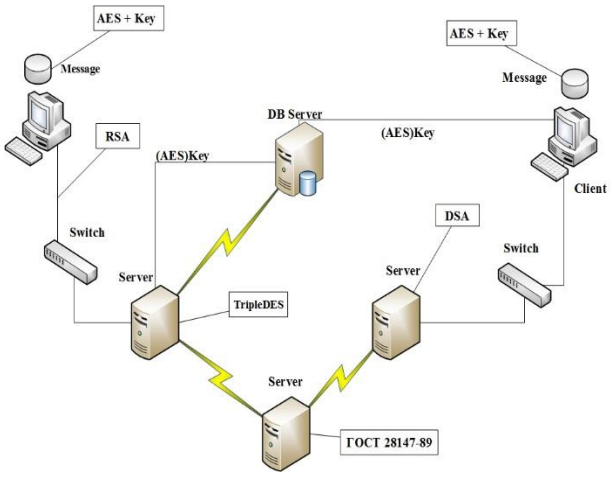

Рис.1. Модель запропонованої системи

В результаті проведеного аналізу визначено, що момент передачі даних в мережі є найбільш слабким місцем в електронному обміні інформацією. Згідно з наведеною моделлю системи робимо висновки, щодо обґрунтування доцільності використання подібного методу передачі даних в мережі.

Для забезпечення захищеної передачі пропонується установлення додаткових вузлів, через які передається повідомлення між кінцевими вузлами. При реалізації основних завдань дослідження, розроблено алгоритми та методи за допомого яких досягається непередбачуваність шляху передачі повідомлення, неповторність алгоритмів шифрування з випадковим вибором кількості перехідних вузлів та їх адрес.

Спираючись на загальну мету дослідження робимо висновок, що система повинна забезпечити шифрування повідомлення на стороні клієнта одним з крипостійких алгоритмів шифрування. Зашифроване повідомлення клієнта та ключ для його розшифрування передаються на попередньо випадково визначений сервер. Передача при цьому шифрується іншим алгоритмом, який визначено на сервері відповідно до методу призначення алгоритму шифрування.

Сервер розшифровує пакет, який містить зашифроване повідомлення та ключ, після чого відправляє ключ на сервер баз даних, про який знають тільки сервери в даній системі. Потім визначається наступний сервер з алгоритмом шифрування, що відповідає йому в даний момент часу та який повторно шифрує і пересилає повідомлення.

Клієнти, як і сервери, знають усі можливі алгоритми шифрування, які змінюються через визначений час. В процесі передачі число серверів в системі визначається попередньо, на відміну від числа проміжних вузлів, яке генерується при передачі за допомогою генератора псевдо-випадкових чисел [9].

Кожен з проміжних серверів має унікальний ідентифікатор, який дописується в кінець повідомлення, задля запобігання петель, при визначенні наступного серверу. Таким чином останній сервер знатиме, що його завдання в тому, щоб перед передачею повідомлення клієнтові, дістати ключ першого алгоритму з сервера баз даних (БД), додати його до повідомлення, зашифрувати своїм алгоритмом, а потім передати пакет клієнтові.

Клієнт розшифрує пакет відповідно до серверного алгоритму та дістанеться останнього (клієнтського) алгоритму з відповідним ключем. Таким чином основне завдання запропонованого методу передачі полягає у випадковості вибору серверів та неможливості визначити, яким з алгоритмів зашифроване повідомлення в даний проміжок часу.

Далі наведено фрагмент коду сервера, який відправляє клієнтові свій відкритий ключ та отримує повідомлення зашифроване двома алгоритмами: алгоритмом клієнта та алгоритмом визначеним попередньо за домовленістю, для передачі (лістинг 1).

Лістинг 1. Фрагмент коду серверу

```
while (true)
{
    string data=null;
    Socket handler = serverSocket.Accept();
    StringBuilder stringBuilder = new String-
Builder();
    int bytes = 0;
    byte[] array = new byte[256];
```

```
        byte[] publicKey = new byte[256];
        if (!File.Exists("file.txt"))
    {
        Data d = rsa.CreateKeyPair();
        rsa.MakeKeyPairFile(d);
    }
        string[] keyPair = new string[2];
        keyPair = rsa.GetKeyPairFromFile();
        publicKey = Encoding.Unicode Get-
Bytes(keyPair[1]); serverSocket.Send(publicKey);
        do
    {
        bytes = handler.Receive(array);
        stringBuilder.Append(Encoding.Unicode.
GetString(array, 0, bytes));
    }
        while (handler.Available > 0);
        string aesMessage= rsa.Decrypt(keyPair[0],
array);
        byte[] aesByte = Encoding.Unicode. Get-
Bytes(aesMessage); byte[][] aesBytes = converter.
ConvertAesBytes(aesByte); string message =
"your message is delivered";
array = Encoding.Unicode.GetBytes(message);
data+=Encoding.Unicode.GetString(aesBytes[2],
0,bytes);
Console.WriteLine("Received text: {0} ", data);
        handler.Send(array);
        handler.Close(); …
    }
```

Кожен з проміжних серверів має унікальний ідентифікатор, який дописується в кінець повідомлення, задля запобігання петель, при визначенні наступного серверу.

Таким чином останній сервер знатиме, що перед передачею повідомлення клієнтові, треба дістати ключ першого алгоритму з сервера баз даних (БД), додати його до повідомлення, зашифрувати своїм алгоритмом і передати пакет клієнтові. Клієнт розшифрує пакет відповідно до серверного алгоритму та дістанеться останнього (клієнтського) алгоритму з відповідним ключем. Завдання запропонованого методу передачі полягає у випадковості вибору серверів та неможливості визначити, яким з алгоритмів зашифроване повідомлення в даний проміжок часу.

Задля запобігання можливості здійснення DDoS-атак, доступ до серверів в системі обмежуємо шляхом настроювання списків доступу так, щоб із зовні не виконувалась команда ping.

Далі наведемо етапи функціонування алгоритму реалізації процесу випадкового вибору сервера, спираючись на структуру заголовку TCP сегменту.

Заголовок сегмента складається з 32-розрядних слів і має змінну довжину, яка залежить від розміру поля Options і кратна 32 бітам [10]. За заголовком слідують дані – частина потоку даних користувача, що передається в даному сегменті. Оскільки потік TCP в загальному випадку може бути довшим, ніж число різних станів поля Sequence Number (порядковий номер), то всі операції з порядковим номером повинні виконуватися по модулю 2^{32}.

Це накладає практичне обмеження на використання TCP. Припустимо, що число проміжних вузлів в мережі дорівнює 15.

Комп’ютер клієнта посилає запит на всі сервери і перший алгоритм використовує 32-х бітний заголовок TCP пакету в якості даних для визначення номеру першого серверу. А саме: 32-х бітний заголовок пакету розбивається на вісім сегментів по чотири біта, відповідно максимальне значення, яке може бути закодоване 4 бітами дорівнює 15, після чого розпочинається процес почергового складання між собою восьми 4-х бітних сегментів за допомогою операції сума по модулю два, в результаті якого отримано число першого сервера з яким клієнт встановлює з'єднання [11].

Опис алгоритму у виді блок-схеми наведено на рисунку 2.

Вибір криптоалгоритмів шифрування даних здійснювався виходячи з можливостей бібліотек System.Security.Cryptography каркаса .NET Framework. Вони включені в аналіз реалізації симетричних алгоритмів шифрування Rijndael, DES, 3DES, RC2 і реалізацію несиметричного алгоритму шифрування RSA.

Для симетричних алгоритмів проведено аналіз їх швидкодії з точки зору шифрування документів різного об'єму, а також зроблена порівняльна характеристика різних симетричних алгоритмів [12].

Для несиметричних алгоритмів шифрування проведено аналіз швидкодії алгоритму RSA при обробці документів різного об'єму при фіксованій довжині модуля і з точки зору обробки документів фіксованого обсягу при різних значеннях модуля.

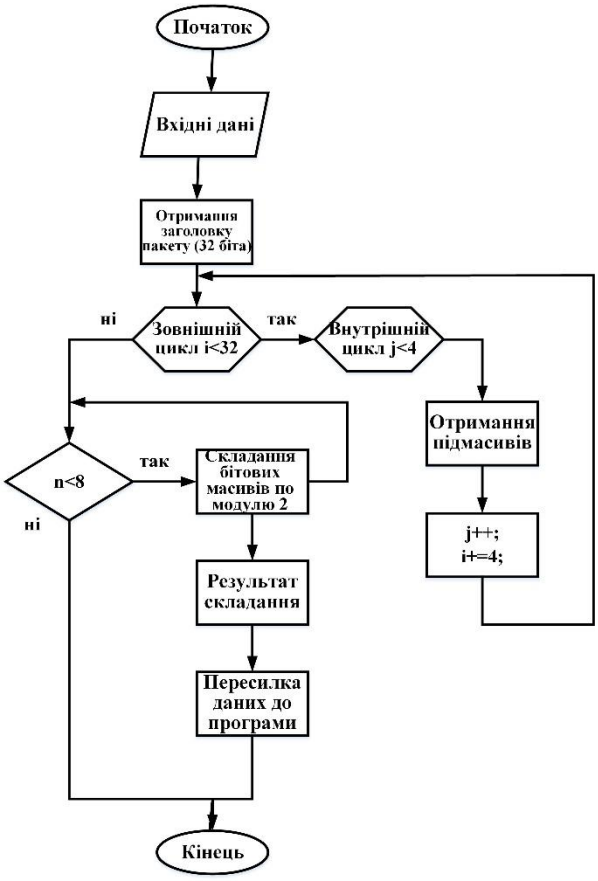

Рис.2. Блок-схема алгоритму

Як характеристика часу, який витрачається на процес шифрування, тут і далі використовувалася тимчасова одиниця тик (tick). Microsoft визначає тик, як найменшу одиницю виміру часу: 1 тик прирівнюється до 100 нс.

При виконанні одноразового аналізу, на графіках є різкі піки, обумовлені в основному розподілом процесорного часу на системні потреби (ресурси ОС і фонових процесів).

Так як криптоперетворення – ресурсномістке завдання, доступність ресурсів системи, таких, як процесорний час і оперативна пам'ять, впливає на результат аналізу. Для згладжування результуючих даних проводилося усереднення їх значень за результатами десяти вимірів [12].

У каркасі .NET Framework симетричний криптоалгоритм Rijndael представлений managed-реалізацією RigndaelManaged. З назви типу, згідно з класифікацією Microsoft маємо, що реалізація даного криптоалгоритму виконана повністю засобами .NET Framework. В проведеному досліджені розглядалися реалізації криптоалгоритмів з розмірами ключів 128, 192, 256 бітів [12].

Симетричний алгоритм RC2 представлений типом RC2CryptoService Provider. Даний тип є оболонкою для реалізації алгоритму з бібліотеки CryptoAPI. У каркасі .NET Framework для алгоритму RC2 допустимі довжини ключа в діапазоні від 40 до 128 бітів з кроком 8 біт. При проведенні аналізу використовувались екземпляри класу провайдера з ключами 40, 64, 96 і 128 бітів. Симетричні алгоритми DES і 3DES представлені типами DESCryptoServiceProvider і TripleDESCryptoServiceProvider відповідно. Дані типи також є оболонками і делегують відповідність реалізацій алгоритмів бібліотеки CryptoAPI.

Реалізація алгоритму DES підтримує довжину ключа 64 біта, тоді як реалізація 3DES дозволяє задавати ключі довжиною 128 біт (2 різних ключа) і 192 біта (3 різних ключа).

У таблиці 1 наведені результати порівняльного аналізу симетричних алгоритмів і несиметричного алгоритму RSA. В таблиці наведено середній час в секундах, який потрібен вказаному криптоалгоритму, щоб перетворити документ певного розміру, який вказано у відповідному стовбці таблиці.

Як видно з таблиці, алгоритм RSA з довжиною модуля в 512 біт працює в середньому на два порядки довше, ніж в середньому реалізації симетричних алгоритмів. При цьому збільшення довжини модуля в 2 і 4 рази (1024 і 2048 біт) призводить до збільшення часу криптоперетворень приблизно в 2 і в 4 рази відповідно [3].

Таблиця 1. Результати аналізу

Крипто-алгоритм	Об'єм документа, Кбайт		
	20000	50000	100000
DES 64 bit	$1{,}10\cdot10^{-3}$	$2{,}03\cdot10^{-3}$	$3{,}80\cdot10^{-3}$
3DES 192 bit	$1{,}85\cdot10^{-3}$	$4{,}24\cdot10^{-3}$	$8{,}50\cdot10^{-3}$
RC2 128 bit	$1{,}35\cdot10^{-3}$	$3{,}60\cdot10^{-3}$	$4{,}72\cdot10^{-3}$
Rijndael 128 bit	$2{,}12\cdot10^{-3}$	$4{,}46\cdot10^{-3}$	$8{,}54\cdot10^{-3}$
RSA 512 bit	$1{,}25\cdot10^{-1}$	$3{,}08\cdot10^{-1}$	$6{,}14\cdot10^{-1}$
RSA 1024 bit	$2{,}51\cdot10^{-1}$	$6{,}31\cdot10^{-1}$	1,26
RSA 2048 bit	$6{,}51\cdot10^{-1}$	1,80	3,61

Фактично продуктивність мережі залежить від наступних параметрів:

– швидкість передачі даних в мережі. Даний фактор визначає, який обсяг даних можна передати за одиницю часу та визначається, в основному, характеристиками каналу зв'язку;

– час відгуку або час реакції так само впливає на продуктивність мережі. Він визначає інтервал часу від посилки запиту до отримання відповіді на нього;

– з серверами пов'язана значна частина мережевого трафіку, тому вони повинні мати плати мережевих адаптерів з високою продуктивністю;

– кількість проміжних вузлів.

4. Швидкодія системи

Процес шифрування повідомлення на кожному з проміжних серверів безумовно впливатиме на швидкість передачі даних. Згідно таблиці 1 найбільш витратним, з точки зору часу, є використання асиметричного алгоритму RSA з ключем 2048 bit.

Візьмемо це значення за максимально можливе значення часу в системі, яке сповільнюватиме загальну швидкість передачі даних.

Тоді швидкість визначаємо наступним способом. Візьмемо за основу, що час передачі першого пакету від вузла N_1 до S_1 можна представити у вигляді суми декількох складових:

$$T_{N1-S1} = t_1 + t_4 + t_5 + t_6 + t_7 + t_8 + t_9, \quad (1)$$

де t_1 - інтервал між пакетами, що дорівнює часу формування пакета – час пакетизації, дорівнює 1 мс; t_4 - час поширення сигналу, який дорівнює передачі одного біту інформації, від вузла N_1 до серверу S_1.

Цей час дорівнює частці від ділення відстані L_1 між джерелом і сервером S_1 на швидкість S поширення сигналу. Відобразимо цей факт в позначенні:

$$t_4 = L_1 / S \quad (2)$$

t_5 – час прийому пакета з його заголовком з каналу під вхідний буфер S_1; це час, який дорівнює (t_2 + t_3) сумі часів буферизації пакета і заголовка.

Візьмемо розмір пакета в бітах: 4Кбайта (дані) + 40байт (заголовок) = (4х1024 +40) х8 біт = 33088 біт. Тоді час буферизації пакета: t_5 = (33088 біт) / (2Мб / с) = 0,0165с = 16,5 мс.

t_6 – час очікування пакета в черзі. Він коливається в широких межах і заздалегідь невідомий, так як залежить від поточної завантаження мережі.

В даному випадку, якщо припустити, що мережа працює в недозавантаженому режимі, даною величиною можна знехтувати, тоді t_6 = 0.

t_7 – час комутації пакета при його передачі в вихідний порт.

Має фіксоване значення для конкретної моделі проміжного обладнання і зазвичай невеликий (від мікросекунд до декількох мілісекунд). Візьмемо його за рівним 2 мс.

t_8 – час на роботу шифрувального алгоритму. Будемо вважати, що клієнт зашифрував повідомлення за допомогою AES зі 128 bit ключем та для передачі на S_1 за допомогою RSA 1024 bit. З урахуванням даних в таблиці 1 маємо:

$$t_8 = (2{,}12_{\text{мс}} + 251_{\text{мс}}) = 253{,}12_{\text{мс}} \quad (3)$$

t_9 – час роботи алгоритму вибору першого серверу, який в свою чергу дорівнює сумі часу на отримання пакету від першого серверу і часу на виконання операцій складання по модулю 2. Її візьмемо рівною 2мс.

$$t_9 = (16{,}5_{\text{мс}} + 2_{\text{мс}}) \quad (4)$$

З урахуванням вище зазначеного отримали:

$$T_{N1-S1} = 1_{\text{мс}} + t_4^{(1)} + 16{,}15_{\text{мс}} + 2_{\text{мс}} + 253{,}12_{\text{мс}} + 18{,}5_{\text{мс}} = 291{,}12_{\text{мс}} + t_4^{(1)} \quad (5)$$

Час T_{S1-S2} передачі першого пакету від серверу S_1 до серверу S_2 відрізняється відсутністю складової $t_1 = 1_{мс}$ (пакет вже сформований) та складовою t_9(необхідно тільки для визначення першого сервера), зміненим значенням складової t_8, яка залежить від вибору алгоритму шифрування і значенням складової t_4, яка залежить від відстані L_2 між серверами S_1 і S_2.

Відобразимо цей факт в позначенні $t_4^{(2)}$. З урахуванням сказаного вище, маємо:

$$T_{S1-S2} = 18{,}5_{мс} + t_4^{(2)} + t_8 \quad (6)$$

Аналогічним чином визначаємо час для кожного з проміжних вузлів.

Кінцеве значення швидкодії запропонованої системи T_{SS} визначимо так:

$$T_{SS} = T_{N1-S1} + T_{S1-S2} + \cdots + T_{Sn-Sn+1}, \quad (7)$$

де n – кількість проміжних вузлів між клієнтами, а m – номер сегмента мережі, для якого визначається довжина відстані між серверами.

Визначимо, що кількість проміжних вузлів $n = 3$. На сегменті $S_1 - S_2$ зашифруємо дані алгоритмом 3DES, на наступному $S_2 - S_3$ – RC2, та на останньому $S_3 - S_4$ – RSA 512 bit.

Тоді отримаємо наступне:

$$T_{SS} = 291{,}12_{мс} + t_4^{(1)} + T_{S1-S2} + t_4^{(2)} + T_{S2-S3} + t_4^{(3)} + T_{S3-S4(client)} + t_4^{(4)} = t_4^{(1)} + t_4^{(2)} + t_4^{(3)} + t_4^{(4)}(291{,}12 + 20{,}35 + 19{,}85 + 143{,}5) = t_4^{(1+2+3+4)}(474{,}85) \approx 0{,}4c \quad (8)$$

У той час як передача даних традиційним методом займає приблизно 0,02с робимо висновок, що подібна система програє в швидкості трансляції даних. Але зауважимо, що приведені вище дані є теоретичними розрахунками, і не можуть використовуватися як фактичні дані а також, що кінцевий показник швидкості прямо пропорційний до швидкості інтернет з'єднання.

Але коли мова йде про забезпечення надійної передачі даних у сферах потребуючих найвищого рівня захисту питання швидкодії не є настільки важливим.

Висновки

Результатом проведених досліджень та розроблених алгоритмів є підвищення показника надійності при обміні даними за рахунок збільшення проміжних вузлів, випадкового вибору наступного вузла, шифрування даних різними алгоритмами під час передачі, неповторність крипто-алгоритмів та їх зміна на вузлах відповідно до певного показника часу. В ході подальших досліджень планується реалізація алгоритму зі збільшення швидкодії системи при передачі даних в мережі.

СПИСОК ЛІТЕРАТУРИ

1. *Бережем свои деньги от интернет-мошенников: что нового в безопасности онлайн-платежей.* – [Електронний ресурс]. – Режим доступу: http://www.prostobank.ua/internet_banking/.
2. *Баричев С.Г.* Основи сучасної криптографії / С.Г. Баричев, Р.Е. Серов. // Учбовий посібник. – М.: Горячая линия, Телеком, 2002. – 152с.:іл.
3. *Шнайер Б.* Прикладна криптографія. Протоколи, алгоритми та ісходні тексти на мові С / Б. Шнайер. – 2-е вид. – М., 2012. – 610с.
4. *Особливості захисту інформації в комп'ютерних мережах.* – [Електронний ресурс]. – Режим доступу: http://um.co.ua/3/3-2/3-26063.html
5. *Шеннон К.* Работы по теории информации и кибернетике / К. Шеннон. – М.: ИЛ, 1963. – С. 333-369.
6. *Мао В.* Сучасна криптографія. Теорія і практика / В. Мао. – М.: Вильямс, 2005. – 768 с.
7. Статистичні дані Canonical по кількості завантажень ОС Ubuntu. – [Електронний ресурс]. – Режим доступу: http://www.computerworlduk.com/it-vendors/canonical-launches-ubuntu-1110-update-3310952/.
8. *Олифер В. Г.* Комп'ютерні мережі. Принципи, технології, протоколи / В. Г. Олифер, Н. А.Олифер. // Підручник для вузів. – 5-е вид. – СПб.: Питер, 2016. – 922с.: іл.
9. *Gutmann P.* Software Generation of Random Numbers for Cryptographic Purposes / P. Gutmann. – Proc. 1998 Usenic Security Symp., Usenix Assoc., Berkeley, Calif., 1998, pp. 243-257.
10. *Internet protocol Darpa internet program Protocol specification.* – [Електронний ресурс]. – Режим

доступу: https://tools.ietf.org/html/rfc79.
11. *Сергеев Н.П.* Основы вычислительной техники / Н.П. Сергеев, Н.П.Вашкевич. – М.: Высш.шк., 1988. – С. 272.
12. *Проценко А.Г.* Исследование быстродействия алгоритмов шифрования данных на базе технологии .Net Framework / А.Г. Проценко, И.В. Лысенко. – Харків: ХУПС. – 2011. – Вип. 4(94). – С. 179–184.

G.G. Kirichek, A.O. Ovchar. **The system of data protection at transmission in the network.**

Г.Г. Киричек, А.А. Овчар. **Система защиты данных при передаче в сети.**

В работе анализируются современные методы защиты передачи данных, их недостатки и преимущества и предлагается решение в виде системы опосредованной передачи, которое исключает непосредственное взаимодействие между клиентами, обеспечивая повышение показателей надежности и криптостойкости, что является актуальным в условиях передачи данных в сети.