Міністерство освіти і науки України

Всеукраїнський конкурс студентських наукових робіт
«Інформаційні технології»

Наукова робота

Технологія захисту інформаційної системи банку при встановленні з’єднань

Шифр Traffic

2019 рік

ЗМІСТ

ВСТУП

На сьогоднішній день саме в банківській сфері спостерігається як позитивний ефект (пов’язаний з впровадженням сучасних автоматизованих технологій обробки інформації та пов’язаного з цим розширення спектру послуг, що надаються, і прискоренням оборотності коштів), так і неминучі негативні вияви. Одне з вразливих місць – пересилання документів між клієнтом і банком.

В зв'язку з цим гостро постає проблема захисту інформації, інтерес до якої обумовлений ще й численними фактами промислового шпигунства та злочинів, здійснених за допомогою комп'ютерних технологій через корпоративні мережі або Інтернет.

Метою досліджень є прискорення шифрування інформації та збільшення криптостійкості блочних алгоритмів в системі «Клієнт-Банк» на основі введення динамічного ключа з використанням програмних та апаратних засобів.

Основні задачі, що визначаються поставленою метою:

1) Реалізувати математичну модель блоково-динамічно шифрування на основі алгоритму Blowfish в системі «клієнт-банк»;

2) Провести криптоаналіз алгоритму блоково-динамічного шифрування в системі «клієнт-банк»;

3) Провести дослідження моделі алгоритму блоково-динамічного шифрування в системі «клієнт-банк» на складність;

4) Розробити методику “відкриття” Р-блоку блоково-динамічного шифру за допомогою функцій випадкової величини в системі «клієнт-банк».

5) На основі запропонованої моделі алгоритму блоково-динамічного шифрування розробити програмне забезпечення у вигляді Марі-клієнта поштового серверу для системи «клієнт-банк».

Об'єктом досліджень є криптографічний захист інформації при встановленні з’єднання від несанкціонованого користування.

Предметом досліджень є методи та засоби блочного шифрування інформації на основі мережі Фейстеля в системі «клієнт-банк».

Методи досліджень базуються на теорії криптографії та криптології, теорії чисел, комбінаториці, теорії множин, криптографії та криптології.

1 Класифікація та аналіз систем захисту інформації при встановленні з’єднання, використовуваних в системах «клієнт-банк»

Методи захисту інформації в системі «клієнт-банк» можна умовно розділити на три великі групи (рис.1.1):

Рис. 1.1 – Класифікаційна схема методів захисту інформації

Організаційні методи – це захист інформації шляхом регулювання доступу до всіх ресурсів системи (технічними засобами, системами телекомунікацій та зв`язку, програмними елементами тощо).

Програмні засоби є програмним забезпеченням, що виконує функції захисту інформації, а апаратні засоби є пристрої, що виконують функції захисту інформації.

Фізичне обмеження доступу – створює фізичні перешкоди для зловмисника, які перегороджують йому шлях до інформації (сувора система пропуску на територію і в приміщення з апаратурою або з носіями інформації).

Адміністрування розділяє права доступу до об’єктів для користувач-чів системи (налаштування політики безпеки, аутентифікація користувач-ча).

Криптографія приховує інформацію шляхом її шифрування.

Антивіруси виявляють і знешкоджують комп'ютерні віруси.

Електронний сейф – програмно-апаратний засіб захисту інформації із фізичним обмеженням доступу через електронні пристрої.

Безпосереднє джерело живлення є апаратний засіб захисту електронних пристроїв від збоїв в постачанні електроенергії.

Своєчасний «апгрейд» дозволяє своєчасно оновлювати комп'ютерну техніку та програмні продукти, слідкувати за виходом нових систем з покращеною безпекою.

Екранування комп’ютерного обладнання і мережних кабелів запобігає перехопленню інформації, що полягає у віддаленому перехопленні побічного електромагнітного випромінювання і наводок від комп’ютера.

Генератор білого шуму – це апаратний пристрій, який призначений для захисту переговорів від прослуховування, що генерує так званий "білий шум" в акустичному діапазоні частот і робить неможливим прослуховування розмови після його передачі будь-яким типом передавальних систем.

Аналіз наукової літератури дозволив виділити наступні вимоги до сучасних криптографічних систем:

1. Вихідний текст із зашифрованого тексту повинен відтворюватись лише за допомогою ключа дешифрування.

2. Послідовне перебирання можливих ключів дешифрування з метою відтворення вихідного тексту потребує значного часу обчислень або великих витрат на реалізацію цих обчислень.

3. Інформація про алгоритм шифрування не повинна впливати на стійкість до зламування системи шифрування.

4. Незначна зміна ключа шифрування повинна призводити до суттєвих змін шифрограми одного і того ж тексту.

5. Незначна зміна вихідного тексту повинна призводити до суттєвих змін шифрограми у випадку використання одного і того ж ключа.

6. Структурні елементи алгоритму шифрування повинні бути незмінними.

7. Додаткові біти, які вводять у повідомлення у процесі шифрування, повинні бути надійно закриті в зашифрованому тексті.

8. Довжина зашифрованого повідомлення не повинна бути більшою, ніж саме

повідомлення.

9. Не повинно бути простих залежностей між ключами, які послідовно використовують під час шифрування.

10. Довільний ключ із множини використовуваних ключів повинен забезпечувати надійність системи шифрування.

11. Алгоритми шифрування й дешифрування повинні допускати як апаратну, так і програмну реалізацію, у цьому разі зміни довжин ключів не повинні приводити до якісного погіршення алгоритмів.

В основу криптосистеми покладено алгоритм шифрування та дешифрува-ння. Введемо наступні визначення.

Криптографічна система (криптосистема) – сімейство Т-перетворень відкритого тексту, члени якого індексуються чи позначаються символом k. Параметр k є ключем. Простір ключів К – це набір можливих значень ключа. Перетворення T_k визначається відповідним алгоритмом і значенням ключа.

Стеганографічна система чи стегосистема – сукупність засобів і методів, що використовуються для формування схованого каналу передачі інформації.

Поєднання криптографії і стеганографії повинно враховувати побудову стегосистеми. Для побудови стегосистеми необхідно враховувати наступні положення:

1. Зловмисник має повне уявлення про стеганографічну систему і деталі її реалізації. Єдиною інформацією, що залишається невідомою потенційному супротивнику, є ключ, за допомогою якого тільки його власник може встановити факт присутності і зміст прихованого повідомлення.

2. Якщо супротивник якимось чином довідається про факт існування прихованого повідомлення, це не повинно дозволити йому вилучити подібні повідомлення з інших даних доти, доки ключ зберігається в таємниці.

3. Потенційний супротивник повинен бути позбавлений будь-яких технічних і інших переваг у розпізнаванні чи розкритті змісту таємних повідомлень.

Криптографічні методи можна розділити на симетричні та асиметричні (рис. 1.2)

Рис. 1.2 – Класифікаційна схема криптографічних методів

Симетричні методи – це шифрування з єдиним (секретним) ключем і відкритим алгоритмом перетворень інформації та даних.

Водночас асиметричні методи представляють собою шифрування з відкритим ключем і відкритим алгоритмом.

Основні переваги шифрів з секретним ключем перед шифрами з відкритим ключем полягають в тому, що вони використовують при шифруванні більш короткий ключ та мають при апаратній реалізації набагато більшу швидкість шифрування. Крім цього, шифри з секретним ключем можуть застосовуватись як приклади побудови різних криптографічних механізмів, включаючи псевдовипадкові числові генератори, хеш-функції та інші. На відміну від шифрів з відкритим ключем шифри з секретним ключем мають більш тривалу історію свого розвитку.

Необхідно зазначити, що слабким місцем при практичній реалізації симетричних шифрів є проблема розподілу ключів. В цьому випадку методи шифрування з відкритим ключем мають явну перевагу, оскільки при їх застосуванні один ключ оголошується відкритим, а інший закритим, причому відкритий ключ публікується і доступний будь-кому, хто бажає послати повідомлення адресату.

Для усунення зазначених недоліків використовуються симетричні криптоалгоритми (рис. 1.3), які дають можливість значно підвищити ступінь секретності при передачі інформації, що використовуються в даній роботі.

Рис. 1.3 – Класифікаційна схема симетричних методів захисту інформації

Симетричні методи, в свою чергу, поділяються на потокові та блокові.

Потокові методи – це шифрування потоків даних з одноразовим або нескінченним ключем (гамування), із скінченним ключем (Вернама), на базі генератора псевдовипадкових чисел.

Блокові методи – це шифрування з переставленням символів у блоці, у алфавіті, системи із заміною символів (підстановками), моноалфавітні,

поліалфавітні, криптосистеми із множенням/діленням на поліном.

В даній роботі для захисту системи «клієнт-банк» використовуються блокові криптоалгоритми (рис. 1.4)

Рис. 1.4 – Класифікаційна схема симетричних блокових методів захисту інформації

У межах блокових криптоалгоритмів прийнято вирізняти методи шифрування з постійним та з динамічним ключами.

Метод шифрування з постійним ключем полягає в тому, що підключ шифрування утворюється за допомогою тільки одного ключа.

Метод шифрування з динамічним ключем полягає в тому, що підключ шифрування утворюється з основного ключа і ключа, який змінюється при кожній наступній операції генерування підключа.

В залежності від характеру дії, які виконують над даними, методи класифікують наступним чином (рис. 1.5):

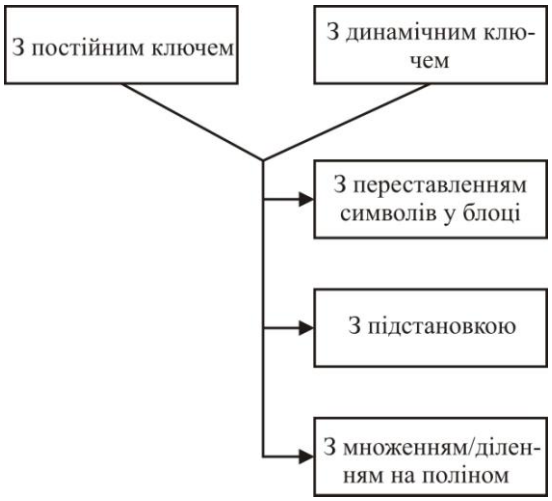

Рис. 1.5 – Класифікаційна схема симетричних методів за характером дії.

Шифрування з переставленням символів у блоці – це метод, при якому за деякими правилами біти інформації в блоці переставляються місцями.

Шифрування з підстановкою – це метод, при якому використо-вується за деякими правилами підстановка даних із таблиць, заповнених випадковими чи псевдовипадковими числами.

Шифрування з множенням/діленням – це метод, який полягає в тому, що кожний блок підлягає аналітичним перетворенням.

Використовуючи аналіз криптографічних та стеганографічних методів, визначимо узагальнюючу схему поетапного вибору запропонованого методу, яка дає їх оптимальне поєднання (рис. 1.6).

В даній системі захисту інформації використовується поєднання двох методів, а саме: криптографічного симетричного блочного методу і стеганографічного методу, заснованого на надлишковості аудіо інформації, що дає можливість підвищити надійність передачі за рахунок використання динамічно утвореного ключа.

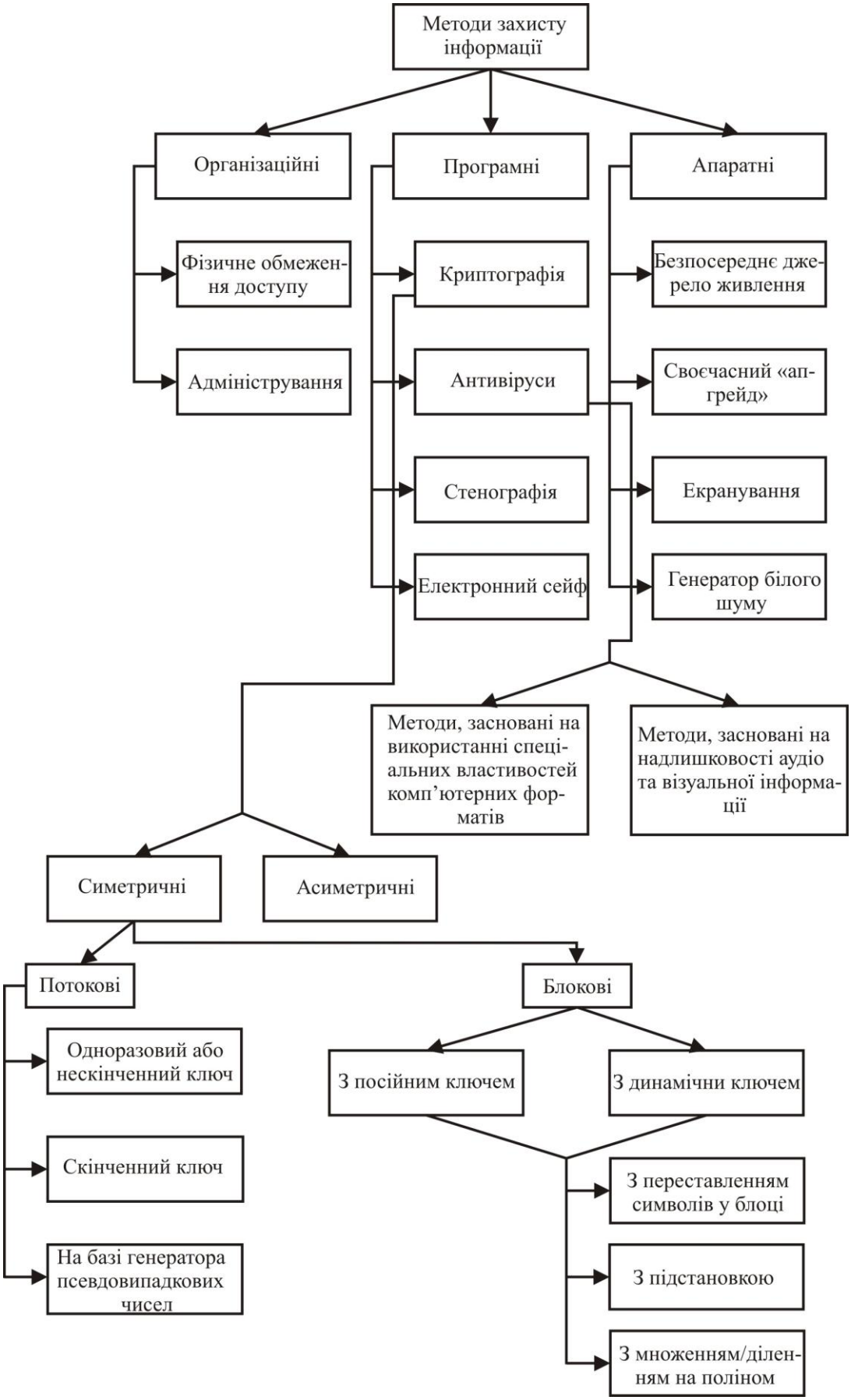

Рис. 1.6 – Загальна класифікаційна схема методів захисту інформації

2 Модифікована модель динамічної зміни підключів реалізації блочного шифрування в системах «клієнт-банк»

Для протидії комп'ютерним злочинам або зменшення збитку від них необхідно грамотно обирати заходи і засоби забезпечення захисту інформації від навмисного руйнування, крадіжки і несанкціонованого доступу. Найбільш ефективним блочним алгоритмом шифрування є Blowfish, який оптимізований для тих задач, у яких немає частої зміни ключів, таких, як поштовий клієнт. Blowfish, що реалізований як мережа Фейстеля (Feistel), має 16 ітерацій. Довжина блоку дорівнює 64 бітам, ключ може мати будь-яку довжину у межах 448 біт. Перед початком будь-якого шифрування виконується складна фаза, і алгоритм складається з двох частин: розширення ключа та шифрування даних. Розширення ключа перетворює ключ довжиною щонайменше 448 біт у кілька масивів підключів загальною довжиною 4168 байт.

Кожна ітерація складається з перестановки, що залежить від ключа, та підстановки, яка залежить від ключа та даних. Використовуються операції XOR та додавання 32-бітних слів. Blowfish використовує велику кількість підключів. Ці ключі повинні бути обчислені заздалегідь, до початку будь-якого шифрування або дешифрування даних. Саме шифрування відбувається достатньо швидко.

У даній роботі ставиться задача підвищення захисту, надійності та швидкодії при передачі інформації в системі «клієнт-банк» при встановленні з’єднання з використанням модифікованого криптографічного алгоритму шифрування Blowfish та програмна реалізація цього алгоритму у вигляді *Mapi*-клієнта поштового серверу (Додаток А).

Після проведення аналізу блочних алгоритмів, який здійснено в попередніх розділах, представимо запропонований нами блоково-динамічний алгоритм в загальному вигляді.

Метод шифрування включає перестановку, підстановку, додавання за модулем 2^{32} та побітний XOR.

Основна операція шифрування даним методом – це підстановка. Процедура підстановки має наступний вигляд:

$$F(x) = (S0_{(x \wedge FF)} + S1_{\overrightarrow{(x(8) \wedge FF)}}) \oplus S2_{\overrightarrow{(x(16) \wedge FF)}} + S3_{\overrightarrow{(x(24) \wedge FF)}}, \quad (2.1)$$

де x – вхідне число; $S0$-$S3$ – підстановочні блоки; $\overrightarrow{x(n)}$ – зсув числа x праворуч на n розрядів; операція додавання «+», виконується як додавання за модулем 2^{32}; $\oplus$ – операція додавання за модулем 2 (XOR); $\wedge$ – операція логічного множення (AND).

Процес шифрування можна описати рекурсивною функцією, розв'язком якої будуть числа, які отримуємо в результаті шифрування:

$$\begin{cases} L^{/} = L \oplus Pi \\ R^{/} = R \oplus F(L^{/}) \end{cases} \Rightarrow R_n, L_n \quad (2.2)$$

Розв'язання проходить на відрізку від 1 до n, де n – кількість ітерацій шифрування, яку можна збільшувати (зменшувати), посилюючи (послаблюючи) шифрування;

$R^{/}$ – права частина даних шифрування 32 біт;

$L^{/}$ – ліва частина даних шифрування 32 біт;

Pi – ключ шифрування для i – го циклу;

Початкові умови системи є вхідними даними шифрування:

L – ліва частина початкових даних шифрування;

R – права частина початкових даних шифрування.

Позначимо систему функцією E з параметрами *x1*, *x2*, тоді повне шифрування буде описуватись як:

$$\mathrm{E}(x1, x2), \Rightarrow \begin{vmatrix} Rr = R_n + P_{n+1} \\ Lr = L_n + P_{n+2} \end{vmatrix} \quad (2.3)$$

де R_n – результат обчислення n-ої ітерації правої частини; L_n – результат обчислення n-ої ітерації лівої частини; P_{n+1}, P_{n+2} – це (n + 1)-ий та (n + 2) ключі шифрування; Lr – ліва частина результату шифрування; Rr – права частина.

Для створення ключа використовується описаний метод шифрування, де в якості ключа виступає байтова послідовність у вигляді символів, а в якості підключа 64-бітне число. Шифруються дані самого ключа, а також дані для підстановок.

Процес створення ключа включає в себе використання функції шифрування.

Оскільки вхідний ключ може бути різної довжини, то перший етап створення ключа полягає у розгортанні вхідного ключа до 448 біт.

$$P_{i+1} = P_i \oplus date_k \quad (2.4)$$

де $i=0...17$

$$date_{k+1} = date_k \oplus 8 \vee key_j \quad (2.5)$$

де $k = 0...3,\ j = j+1,\ j = j \bmod keysize$

З (2.4), (2.5) отримуємо (2.6)

$$P = \left[P_i \oplus \sum_{n=0}^{3} \overleftarrow{key}\Big|_{\substack{j \leftarrow j+1 \\ j \leftarrow j \bmod keysize}} \right]_0^N, \quad (2.6)$$

де key – вхідний ключ шифрування; keysize – розмір ключа – кількість key-послідовностей; P – ключ для шифрування, який складається із $P_1, P_2, ...P_{18}$ частин; mod – залишок від цілочисельного ділення; N – довжина ключа шифрування в 32 бітних блоках; k –кількість ітерацій логічного додавання; j – поточний знак ключа key

$$N = n + 2.$$

Причому на початку кожної ітерації результат попередньої операції OR зсувається на 8 розрядів ліворуч.

Ця операція виконується для кожного P_i-го елемента ключа, де $i \in [0, N]$.

У нашому удосконаленому варіанті Blowfish додавання $\sum_{n=0}^{3} \overleftarrow{key}\Big|_{\substack{j \leftarrow j+1 \\ j \leftarrow j \bmod keysize}}$ починається із значення $key2_1$, де $key2_1$ – це перша частина 64-бітного числа, $key2_1$ – дозволяє змінити початкові ключі, що вносить додатковий захист до шифрування.

Для покращення надійності ключа для самого ключа та для S-блоків виконується операція шифрування, причому при шифруванні використовують-ся вже попередньо зашифровані S-блоки.

Шифрування ключа можна описати як рішення рекурсивної функції для кожної пари ключів P_i і P_{i+1}, де $i \in [0, N]$, причому початкові значення, які будуть циклічно

передаватись як початкові умови із рекурсивної функції в рекурсивну функцію у звичайному Blowfish приймають значення нулів, а в удосконаленому методі це допоміжний 64-бітний ключ, який розбивається на дві 32-бітні частини:

$$\mathrm{E}(key2_1, key2_2), \Rightarrow \begin{vmatrix} P_1 = P_n^1 \oplus P_{n+1} \\ P_2 = P_n^2 \oplus P_{n+2} \end{vmatrix}$$

$$\mathrm{E}(P_1, P_2) \Rightarrow \begin{vmatrix} P_3 = P_n^3 \oplus P_{n+1} \\ P_4 = P_n^4 \oplus P_{n+2} \end{vmatrix},$$

…

$$\mathrm{E}(P_{15}, P_{16}) \Rightarrow \begin{vmatrix} P_{N-1} = P_n^{N-1} \oplus P_{n+1} \\ P_N = P_n^N \oplus P_{n+2} \end{vmatrix},$$

де $\mathrm{key2}_1$ – перша частина 64-бітного другого ключа; $\mathrm{key2}_2$ – друга частина 64-бітного другого ключа; P_1 … P_N – нові ключі після шифрування; $P_n^1 - P_n^N$ – n-й розв'язок рекурсивної функцій E(*x1*,*x2*).

Після шифрування ми отримаємо нові ключі $P_1, P_2, \ldots, P_N$.

Вхідні значення для шифрування S-блоків: P_{17}, P_{18}. Шифруючи ці значення, ми отримуємо нові значення, які передаємо на наступний цикл шифрування. Так шифруються всі S-блоки.

Шифрування S-блоків проходить для кожної пари послідовно.

Для першого S-блоку *S*0:

$$\mathrm{E}(P_{17}, P_{18}) \Rightarrow \begin{vmatrix} S0_1 = S0_n^1 \oplus P_{n+1} \\ S0_2 = S0_n^2 \oplus P_{n+2} \end{vmatrix},$$

$$\mathrm{E}(S0_1, S0_2) \Rightarrow \begin{vmatrix} S0_3 = S0_n^3 \oplus P_{n+1} \\ S0_4 = S0_n^4 \oplus P_{n+2} \end{vmatrix},$$

…

$$\mathrm{E}(S0_{252}, S0_{253}) \Rightarrow \begin{vmatrix} S0_{254} = S0_n^{254} \oplus P_{n+1} \\ S0_{255} = S0_n^{255} \oplus P_{n+2} \end{vmatrix}.$$

При шифруванні у функції підстановки F, яка використовується у системі диференційних рівнянь E, використовуються елементи вже зашифрованих, а також незашифрованих S-блоків.

Таким чином шифруються наступні S-блоки.

Для четвертого S-блоку *S*3:

$$\mathrm{E}(S2_{254}, S2_{255}) \Rightarrow \begin{vmatrix} S3_1 = S3_n^1 \oplus P_{n+1} \\ S3_2 = S3_n^2 \oplus P_{n+2} \end{vmatrix},$$

$$\mathrm{E}(S3_1, S3_2) \Rightarrow \begin{vmatrix} S3_3 = S3_n^3 \oplus P_{n+1} \\ S3_4 = S3_n^4 \oplus P_{n+2} \end{vmatrix},$$

…

$$\mathrm{E}(S3_{252}, S3_{253}) \Rightarrow \begin{vmatrix} S3_{254} = S3_n^{254} \oplus P_{n+1} \\ S3_{255} = S3_n^{255} \oplus P_{n+2} \end{vmatrix},$$

де $S0_1 \ldots S0_{255}$ – значення першого S-блоку; $S1_1 \ \ldots \ S1_{255}$ – значення другого S-блоку; $\ldots S3_1 \ldots S3_{255}$ – значення четвертого S-блоку; $S0_n^1 \ldots S0_n^{255} \ldots S3_n^1 \ldots S3_n^{255}$ – n-й розв'язок рекурсивної функцій E(*x1,x2*), для елементів 1…255 блоків S0-S3.

Отже, повна операція створення ключа буде мати наступний вигляд:

$$P = \left[P_i \oplus \sum_{n=0}^{3} \overleftarrow{key}\Big|_{\substack{j \leftarrow j+1 \\ j \leftarrow j \bmod keysize}} \right]_0^N,$$

$$\mathrm{E}(key2_1, key2_2) \Rightarrow \begin{vmatrix} P_1 = P_n^1 \oplus P_{n+1} \\ P_2 = P_n^2 \oplus P_{n+2} \end{vmatrix} \ldots \mathrm{E}(\mathrm{P}^{\mathrm{N-3}}\mathrm{P}^{\mathrm{N-2}}) \Rightarrow \begin{vmatrix} P_{N-1} = P_n^{N-1} \oplus P_{n+1} \\ P_N = P_n^N \oplus P_{n+2} \end{vmatrix}$$

$$\mathrm{E}(P_{N-1}, P_N) \Rightarrow \begin{vmatrix} S0_1 = S0_n^1 \oplus P_{n+1} \\ S0_2 = S0_n^2 \oplus P_{n+2} \end{vmatrix} \ldots \mathrm{E}(\mathrm{S3}^{252}, \mathrm{S3}^{253})$$

$$\Rightarrow \begin{vmatrix} S3_{254} = S3_n^{254} \oplus P_{n+1} \\ S3_{255} = S3_n^{255} \oplus P_{n+2} \end{vmatrix},$$

а операція шифрування:

$$\begin{cases} L^{/} = L \oplus Pi \\ R^{/} = R \oplus F(L^{/}) \end{cases} \Rightarrow R_n, L_n \Rightarrow \begin{vmatrix} Rr = R_n \oplus P_{n+1} \\ Lr = L_n \oplus P_{n+2} \end{vmatrix}.$$

Операція створення ключа починається формуванням ключа, який має розмір (keysize) у 448 біт. Для цього виконують наступні дії.

Початкове значення в кожній *i*-тій операції, яке дорівнює першій частині другого ключа $key2_1$ переміщують ліворуч на 8 розрядів і операцією OR об'єднують із *j*-ю частиною ключа key, при цьому збільшуючи *j* на 1, якщо воно менше keysize, і збиваючи в 0, якщо більше або рівне. Ці операції здійснюються чотири рази, після чого отримане значення додається за модулем 2 до *i*-ї частини ключа *P*, в результаті отримується новий ключ P_i. Описані вище операції здійснюються для кожних P_0-P_N.

Наступна операція, операція шифрування ключа *P* починається із шифрування допоміжних ключів $key2_1$, $key2_2$:

$$E(key2_1, key2_2) \Rightarrow \begin{vmatrix} P_1 = P_n^1 \oplus P_{n+1} \\ P_2 = P_n^2 \oplus P_{n+2} \end{vmatrix}$$, в результаті чого отримуються ключі P_1 і P_2.

В свою чергу ці ключі передаються як початкові дані і використовуються як початкові умови в наступній операції розв'язку рекурсивної функції. Таким чином формуються всі 18 *P*-ключів.

Останні результати P_{17} і P_{18} використовуються як початкові умови для шифрування S-блоків. Для кожної наступної операції шифрування використовуються дані попередньої ітерації. Таким чином формуються всі S-блоки, які потім будуть використовуватись для шифрування даних.

Дешифрування виконується аналогічно шифруванню, але P_1, P_2, . . ., P_{18} використовуються у зворотному порядку.

Графік залежності вхідної ($Data_i$) та вихідної ($Data1_i$) інформації при застосуванні запропонованого алгоритму наведено на рис. 2.1.

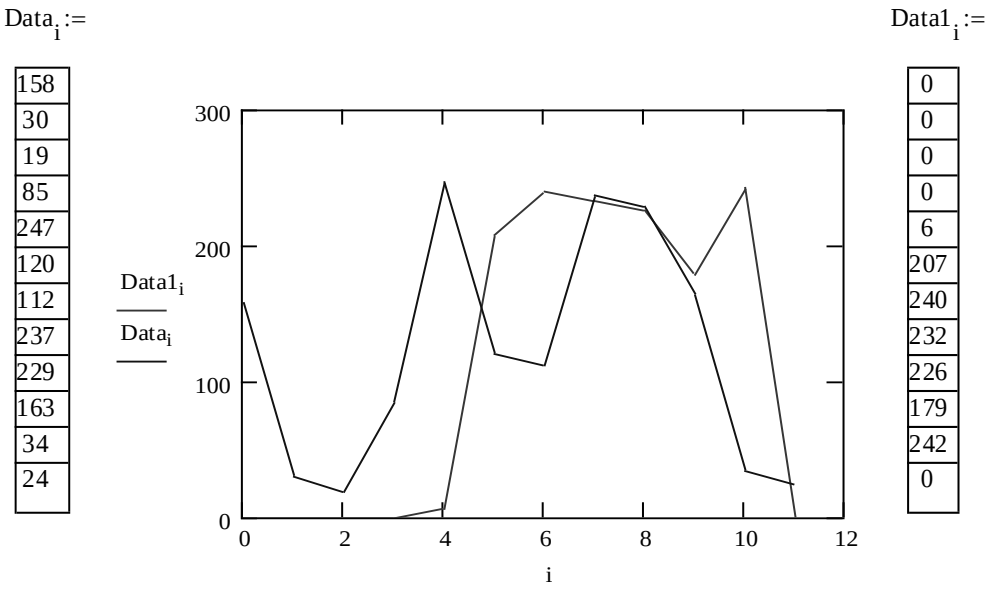

Рис. 2.1 – Графік залежності вихідної від вхідної інформації

Як бачимо з графіка, зашифровані дані не мають лінійних властивостей, що свідчить, що в даному криптографічному методі не має чітко виражених логічних дій, а це в свою чергу, свідчить про криптографічну стійкість даного методу щодо лінійного криптоаналізу.

Для слабких ключів, які генерують погані S-блоки, додавання двох додаткових 32-бітних ключів підвищує захищеність ключа, що підвищує якість шифрування. При невідомих S-блоках ми можемо виявити використання слабкого ключа, але не можемо визначити сам ключ (S-блоки та P-масив). Тому цей метод ефективний тільки проти варіантів із зменшеною кількістю етапів та абсолютно недієвий – проти 16-етапного.

3 Методика оцінки стійкості алгоритму блоково-динамічного шифрування в системах «клієнт-банк» при встановленні з'єднання

3.1 Імовірнісний метод "відкриття" Р-блоку блоково-динамічного шифру за допомогою функцій випадкової величини в системі «клієнт-банк»

Припустимо, що існує відкрите повідомлення та шифрований текст, що йому відповідає, а також частина алгоритму блоково-динамічного шифрування за винятком функції розгортання ключа в Р-блок. Імовірнісний метод "відкриття" Р-блоку блоково-динамічного шифру полягає в циклічній перевірці ключа Р-блоку, отриманого на основі генератора випадкових чисел до тих пір, поки він не задовольнить рівність T(x, P) = y, де x – відкрите повідомлення; y – шифрований текст; T – шифр.

Для блоково-динамічного шифру Р-блок являє собою масив 32-бітних чисел.

Розглянемо такі функції випадкової величини, як стандартна функція Random та ітеративна функція виду $X_{i+1}=\{11X_i+\pi\}$.

Випадковий Р-блок генерується за співвідношеннями для функції Random

$$P_i = \text{Random}(2^{4\times 8})_i, \tag{3.1}$$

для функції $X_{i+1}=\{11X_i+\pi\}$

$$P_i = \text{Trunc}(\text{d}_\text{j} \cdot 2^{4\times 8})_i; \tag{3.2}$$

$$d_{j+1} = Frac(11d_j + \pi), \tag{3.3}$$

де i – поточний індекс елементу Р-блоку (i=0…n-1); n – кількість елементів Р-блоку (для блоково-динамічного шифру n=18); Random(a) – функція випадкової величини 0≤Random<a; d – випадкова величина 0≤d<1; Trunc – функція цілої частини числа; Frac – функція дробової частини числа; $2^{4\times 8}$ – кількість можливих варіантів 32-бітного числа.

На рис. 3.1 показана схема "відкриття" Р-блоку за допомогою імовірнісного методу.

Оскільки основну частину часу в блоково-динамічному алгоритмі займає генерування S-блоків, то відсутність необхідності розгортання ключа до P-блоку несуттєво збільшить швидкодію вказаного алгоритму. Тому приймаємо швидкодію “відкриття” P-блоку рівною швидкодії “відкриття” ключа, яка складає близько 550 варіантів за секунду.

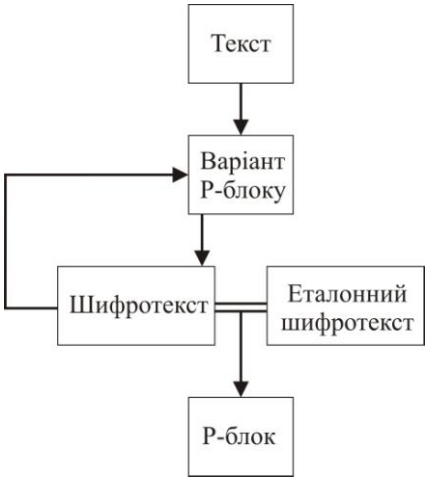

Рис. 3.1 – Схема “відкриття” P-блоку за допомогою імовірнісного методу

Отже, для оцінки трудомісткості “відкриття” P-блоку блоково-динамічного шифру можна скористатися формулами $E = 0{,}9426\mathrm{e}^{4{,}7174m}$ та $E = 0{,}0506\mathrm{e}^{5{,}3214m}$ для функцій Random та $X_{i+1}=\{11X_i+\pi\}$ відповідно, підставивши замість m кількість байт у P-блоці:

$$m = 4n = 4 \cdot 18 = 72, \tag{3.4}$$

де n – кількість елементів P-блоку (для блоково-динамічного шифру n=18); 4 – кількість байт у 32-бітному числі.

Таким чином, трудомісткість “відкриття” P-блоку для функції Random становитиме

$$E = 0{,}9426\mathrm{e}^{4{,}7174 \cdot 72} = 3{,}0445 \cdot 10^{147}.$$

Для функції $X_{i+1}=\{11X_i+\pi\}$

$$E = 0{,}0506\mathrm{e}^{5{,}3214 \cdot 72} = 1{,}2592 \cdot 10^{165}.$$

Для оцінки часу “відкриття” Р-блоку блоково-динамічного шифру можна скористатися формулами $t = 0{,}0009\text{x}^{13{,}997}$ та $t = 0{,}0007\text{x}^{13{,}534}$ для функцій Random та $X_{i+1}=\{11X_i+\pi\}$ відповідно, підставивши замість x кількість байт у Р-блоці

Для функції Random

$$t = 0{,}0009 \cdot 72^{13{,}997} = 8{,}9398 \cdot 10^{22} \text{ с.}$$

Для функції $X_{i+1}=\{11X_i+\pi\}$

$$t = 0{,}0007 \cdot 72^{13{,}534} = 9{,}5992 \cdot 10^{21} \text{ с.}$$

В результаті порівняння часу “відкриття” Р-блоку імовірнісним методом приходимо до висновку, що функція $X_{i+1}=\{11X_i+\pi\}$ швидкодійніша за функцію Random приблизно в 9,313 рази.

З метою аналізу можливості “відкриття” Р-блоку імовірнісним методом за допомогою функцій Random та $X_{i+1}=\{11X_i+\pi\}$ перерахуємо значення трудомісткості та часу “відкриття”, отриманих за допомогою програмної реалізації на персональному комп’ютері з процесором K6-2-500 в значення трудомісткості та часу “відкриття”, які можна отримати за допомогою програмної реалізації на більш сучасних персональних комп’ютерах (Duron-1800, Celeron-2200, Sempron-2600, Pentium-IV-3000), а також за допомогою суперкомп’ютера, оснащеного паралельними процесорами Pentium-IV-3000 в кількості 8192 штук (табл. 2.1). З метою спрощення аналізу будемо вважати, що час “відкриття” ключа обернено пропорційний робочій частоті процесора, без врахування архітектурних розширень типу MMX, 3DNow!, SSE, Extended 3DNow!, SSE2 та ін. і для перерахунку часу “відкриття” Р-блоку доцільно скористатися формулою $t_i = t_0 * \frac{f_0}{f_i}$.

Для зручності час переведемо із секунд в роки за формулою $t_{років} = \frac{t_c}{365{,}25 \cdot 24 \cdot 60 \cdot 60}$.

Таблиця 2.1 – Теоретичні значення трудомісткості та часу “відкриття” P-блоку

Процесор	Трудомісткість E		час “відкриття” t, років	
	Random	$X_{i+1}=\{11X_i+\pi\}$	Random	$X_{i+1}=\{11X_i+\pi\}$
K6-2-500	$3{,}0445\cdot10^{147}$	$1{,}2592\cdot10^{165}$	$2{,}8309\cdot10^{15}$	$3{,}0397\cdot10^{14}$
Duron-1800			$7{,}8664\cdot10^{14}$	$8{,}4436\cdot10^{13}$
Celeron-2200			$6{,}4339\cdot10^{14}$	$6{,}9084\cdot10^{13}$
Sempron-2600			$5{,}4440\cdot10^{14}$	$5{,}8456\cdot10^{13}$
Pentium-IV-3000			$4{,}7182\cdot10^{14}$	$5{,}0662\cdot10^{13}$
8192xPentium-IV-3000			$5{,}7595\cdot10^{10}$	$6{,}1843\cdot10^{9}$

На основі даних, представлених в табл. 3.1, побудуємо графіки залежностей часу “відкриття” P-блоку блоково-динамічного шифру від частоти процесора (для зручності відображення у логарифмічній системі координат) для функцій Random (рис. 3.2) та $X_{i+1}=\{11X_i+\pi\}$ (рис. 2.3) відповідно.

За допомогою табл. 3.1 та рис. 3.2, 3.3 можна визначити, за який час можна “відкрити” P-блок алгоритму блоково-динамічного шифрування імовірнісним методом криптоаналізу за допомогою функцій Random та $X_{i+1}=\{11X_i+\pi\}$ відповідно. Із порівняння результатів таблиці можна зробити висновок, що функція $X_{i+1}=\{11X_i+\pi\}$ швидкодійніша за функцію Random приблизно в 9,313 рази. Але використовуючи навіть потужний суперкомп’ютер 8192xPentium-IV-3000 P-блок блоково-динамічного шифру можна “відкрити” лише за $6{,}1843\cdot10^{9}$ років, що дає підстави вважати P-блок блоково-динамічного шифру стійким до криптоаналізу імовірнісним методом.

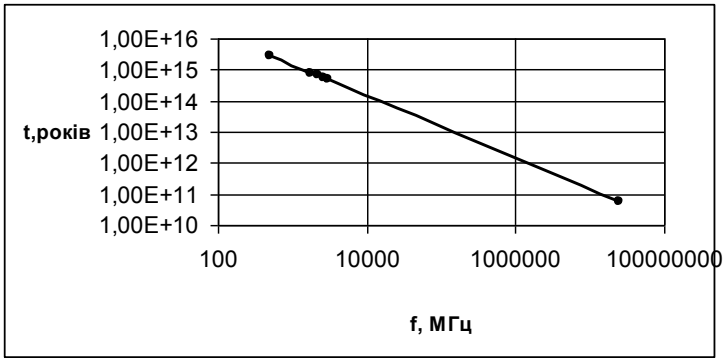

Рис. 3.2 – Графік залежності часу “відкриття” P-блоку блоково-динамічного шифру від частоти процесора для функції Random

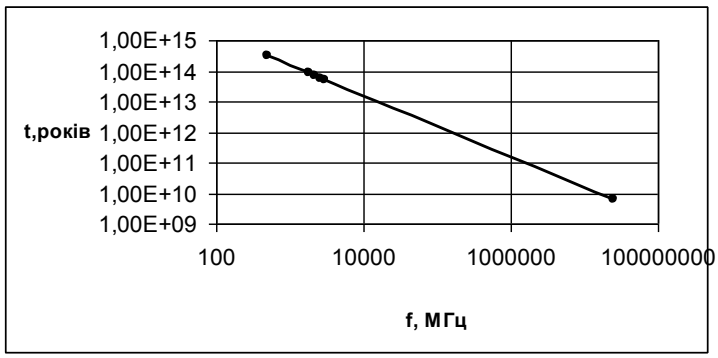

Рис. 3.3 – Графік залежності часу “відкриття” P-блоку блоково-динамічного шифру від частоти процесора для функції $X_{i+1}=\{11X_i+\pi\}$

3.2 Визначення стійкості ключа із обмеженого алфавіту символів блоково-динамічного шифру імовірнісним методом та методом повного перебору в системах «клієнт-банк» при встановленні з’єднання

Порівняльний аналіз можливо провести коли є відкрите повідомлення та шифрований текст, що йому відповідає, а також алгоритм блоково-динамічного шифрування. Відома також множина символів ключа (літери – великі та малі, цифри, спеціальні символи тощо). Необхідно знайти ключ, за допомогою якого відкрите повідомлення із застосуванням алгоритму блоково-динамічного шифрування BlowFish перетворюється в шифрований текст.

Ця задача є частинним випадком задачі “відкриття” ключа, множина символів якого складає всю таблицю ASCII. Очевидно, що розв’язання такого частинного випадку задачі є набагато швидшим ніж повного.

Розділимо умовно алфавіт ASCII на такі множини:

A – малі латинські літери: a-z;

B – великі латинські літери: A-Z;

C – малі літери кирилиці: а-я;

D – великі літери кирилиці: А-Я;

E – цифри: 0-9;

F – спеціальні символи: ~!@#$%^&*()_+|`-=\{}[]:;" <,>.?/

G – всі символи.

Зведемо до табл. 3.2 дані про вказані множини символів ключа.

Таблиця 3.2 – Дані про множини символів ключа

Група символів	Множина	Кількість символів у групі
малі латинські літери: a-z	A	26
великі латинські літери: A-Z	B	26
малі літери кирилиці: а-я	C	32
великі літери кирилиці: А-Я	D	32
цифри: 0-9	E	10
спеціальні символи: ~!@#$%^&*()_+|`-=\{}[]:;" <,>.?/	F	32
всі символи	G	256

На основі вказаних (базових) множин символів можна створити нові множини, які можуть реально утворитись під час набору користувачем ключа:

– малі та великі латинські літери a-z, A-Z (утворюється при наборі ключа при англійській розкладці клавіатури за умови відсутності зміни розкладки клавіатури під час набору ключа і без використання цифрової клавіатури та спецсимволів);

– малі та великі літери кирилиці а-я, А-Я (утворюється при наборі ключа при українській розкладці клавіатури за умови відсутності зміни розкладки клавіатури під час набору ключа і без використання цифрової клавіатури та спецсимволів);

– малі та великі латинські літери a-z, A-Z, цифри 0-9 (утворюється при наборі ключа при англійській розкладці клавіатури за умови відсутності зміни розкладки клавіатури під час набору ключа та без використання спецсимволів);

– малі та великі літери кирилиці а-я, А-Я, цифри 0-9 (утворюється при наборі ключа при українській розкладці клавіатури за умови відсутності зміни розкладки клавіатури під час набору ключа та без використання спецсимволів);

– малі та великі латинські літери a-z, A-Z, спецсимволи (утворюється при наборі ключа при англійській розкладці клавіатури за умови відсутності зміни розкладки клавіатури під час набору ключа та без використання цифрової клавіатури);

– малі та великі літери кирилиці а-я, А-Я, спецсимволи (утворюється при наборі ключа при українській розкладці клавіатури за умови відсутності зміни розкладки клавіатури під час набору ключа та без використання цифрової клавіатури);

– малі та великі латинські літери a-z, A-Z, цифри 0-9, спецсимволи (утворюється при наборі ключа при англійській розкладці клавіатури за умови відсутності зміни

розкладки клавіатури під час набору ключа);

– малі та великі літери кирилиці а-я, А-Я, цифри 0-9, спецсимволи (утворюється при наборі ключа при українській розкладці клавіатури за умови відсутності зміни розкладки клавіатури під час набору ключа);

– малі та великі латинські літери a-z, A-Z, малі та великі літери кирилиці а-я, А-Я (утворюється при наборі ключа при зміні розкладки клавіатури під час набору ключа та без використання цифрової клавіатури і спецсимволів);

– малі та великі латинські літери a-z, A-Z, малі та великі літери кирилиці а-я, А-Я, цифри 0-9 (утворюється при наборі ключа при зміні розкладки клавіатури під час набору ключа та без використання спецсимволів);

– малі та великі латинські літери a-z, A-Z, малі та великі літери кирилиці а-я, А-Я, спецсимволи (утворюється при наборі ключа при зміні розкладки клавіатури під час набору ключа та без використання цифрової клавіатури);

– малі та великі латинські літери a-z, A-Z, малі та великі літери кирилиці а-я, А-Я, цифри 0-9, спецсимволи (утворюється при наборі ключа при зміні розкладки клавіатури під час набору ключа).

Зведемо в табл. 3.3 дані про утворені реальні множини символів ключа.

Таблиця 3.3 – Дані про утворені реальні множини символів ключа

Група символів	Множина	Кількість символів у групі
малі та великі латинські літери a-z, A-Z	A+B	52
малі та великі літери кирилиці а-я, А-Я	C+D	64
малі та великі латинські літери a-z, A-Z, цифри 0-9	A+B+E	62
малі та великі літери кирилиці а-я, А-Я, цифри 0-9	C+D+E	74
малі та великі латинські літери a-z, A-Z, спецсимволи	A+B+F	84
малі та великі літери кирилиці а-я, А-Я, спецсимволи	C+D+F	96
малі та великі латинські літери a-z, A-Z, цифри 0-9, спецсимволи	A+B+E+F	94
малі та великі літери кирилиці а-я, А-Я, цифри 0-9, спецсимволи	C+D+E+F	106
малі та великі латинські літери a-z, A-Z, малі та великі літери кирилиці а-я, А-Я	A+B+C+D	116
- малі та великі латинські літери a-z, A-Z, малі та великі літери кирилиці а-я, А-Я, цифри 0-9	A+B+C+D+E	126
- малі та великі латинські літери a-z, A-Z, малі та великі літери кирилиці а-я, А-Я, спецсимволи	A+B+C+D+F	148
малі та великі латинські літери a-z, A-Z, малі та великі літери кирилиці а-я, А-Я, цифри 0-9, спецсимволи	A+B+C+D+E+F	158

Кількість можливих варіантів ключів (трудомісткість E) для кожної множини символів можна знайти за формулою:

$$E = n^m, \tag{3.5}$$

де n – кількість символів у групі (множині символів); m – довжина ключа в символах.

Для наочності ця залежність показана в геометричній інтерпретації на рис. 3.4.

Середню трудомісткість “відкриття” ключа методом повного перебору можна визначити як половину повної:

$$E_{ПП} = \frac{n^m}{2}. \tag{3.6}$$

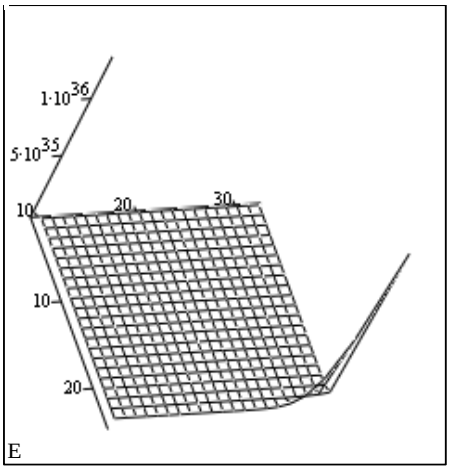

Рис. 3.4 Геометрична інтерпретація залежності (3.5)

На основі регресійних залежностей $E = 0{,}9426\mathrm{e}^{4{,}7174\mathrm{m}}$, $E = 0{,}0506\mathrm{e}^{5{,}3214\mathrm{m}}$, отриманих для імовірнісного методу для ключів зі всієї множини ASCII можна перейти до залежностей, за якими можна визначити трудомісткість для множин–груп символів.

Для функції Random

$$E = 0{,}9426\mathrm{e}^{4{,}7174\mathrm{mn}/36} = 0{,}9426\mathrm{e}^{0{,}0184\mathrm{mn}}. \tag{3.7}$$

Для функції $X_{i+1}=\{11X_i+\pi\}$

$$E = 0{,}0506e^{5{,}3214mn/36} = 0{,}0506e^{0{,}0208mn}. \quad (3.8)$$

Використовуючи формули (3.6-3.8), розраховуємо для множин з кількістю символів n і ключів з довжиною $m \in \{14,28,56\}$ символів значення трудомісткос-ті "відкриття" ключів для методу повного перебору, імовірнісного методу з використанням функцій Random та $X_{i+1}=\{11X_i+\pi\}$ відповідно.

На основі даних таблиці 3.4 побудуємо графіки залежностей 3.5-3.7 трудомісткості "відкриття" ключів від кількості символів у множині n для різних значень довжини ключа m. Для зручності шкалу по вісі ординат вибрано логарифмічною.

Таблиця 3.4 – Значення трудомісткості "відкриття" ключів для множин з кількістю символів n і ключів з довжиною $m \in \{14,28,56\}$ символів для методу повного перебору, імовірнісного методу з використанням функцій Random та $X_{i+1}=\{11X_i+\pi\}$ відповідно

n\m	Повний перебір			Random			$X_{i+1}=\{11X_i+\pi\}$		
	14	28	56	14	28	56	14	28	56
10	5E+13	5E+27	5E+55	12,38996	162,8593	28138,29	0,611522	7,390507	1079,439
26	3,23E+19	2,08E+39	8,66E+78	763,9381	619140	4,07E+11	32,96335	21473,96	9,11E+09
32	5,9E+20	6,97E+41	9,71E+83	3583,469	13623227	1,97E+14	147,0241	427195,2	3,61E+12
52	5,28E+23	5,59E+47	6,24E+95	619140	4,07E+11	1,75E+23	21473,96	9,11E+09	1,64E+21
62	6,2E+24	7, +49	1,2E+100	8138258	7,03E+13	5,24E+27	259521,9	1,33E+12	3,5E+25
64	9,67E+24	1,87E+50	7E+100	13623227	1,97E+14	4,11E+28	427195,2	3,61E+12	2,57E+26
74	7,38E+25	1,09E+52	2,4E+104	1,79E+08	3,4E+16	1,23E+33	5162834	5,27E+14	5,48E+30
84	4,35E+26	3,79E+53	2,9E+107	2,35E+09	5,88E+18	3,67E+37	62395037	7,69E+16	1,17E+35
94	2,1E+27	8,84E+54	1,6E+110	3,09E+10	1,02E+21	1,09E+42	7,54E+08	1,12E+19	2,5E+39
96	2,82E+27	1,59E+55	5,1E+110	5,18E+10	2,85E+21	8,59E+42	1,24E+09	3,04E+19	1,83E+40
106	1,13E+28	2,56E+56	1,3E+113	6,81E+11	4,92E+23	2,56E+47	1,5E+10	4,45E+21	3,91E+44
116	3,99E+28	3,19E+57	2E+115	8,95E+12	8,49E+25	7,66E+51	1,81E+11	6,5E+23	8,34E+48
126	1,27E+29	3,23E+58	2,1E+117	1,18E+14	1,47E+28	2,29E+56	2,19E+12	9,49E+25	1,78E+53
148	1,21E+30	2,93E+60	1,7E+121	3,4E+16	1,23E+33	1,6E+66	5,27E+14	5,48E+30	5,94E+62
158	3,02E+30	1,83E+61	6,7E+122	4,47E+17	2,12E+35	4,77E+70	6,37E+15	8,01E+32	1,27E+67
256	2,6E+33	1,35E+67	3,6E+134	4,11E+28	1,79E+57	3,4E+114	2,57E+26	1,31E+54	3,4E+109

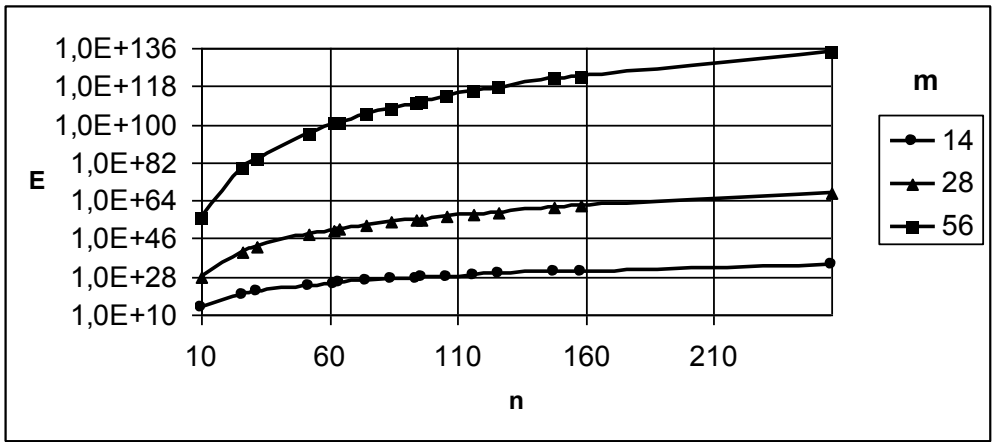

Рис. 3.5 – Графік залежності трудомісткості від кількості символів у множині n для різних значень довжини ключа m для методу повного перебору

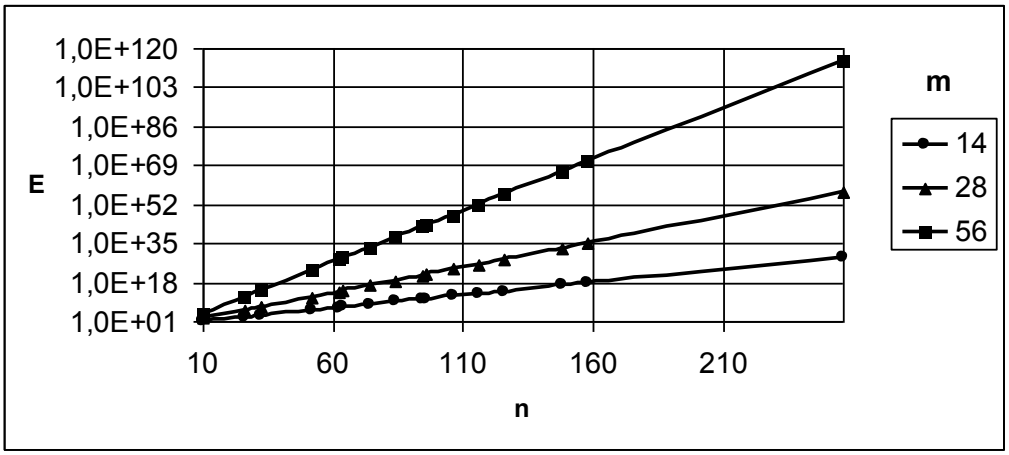

Рис. 3.6 – Графік залежності трудомісткості від кількості символів у множині n для різних значень довжини ключа m для імовірнісного методу (Random)

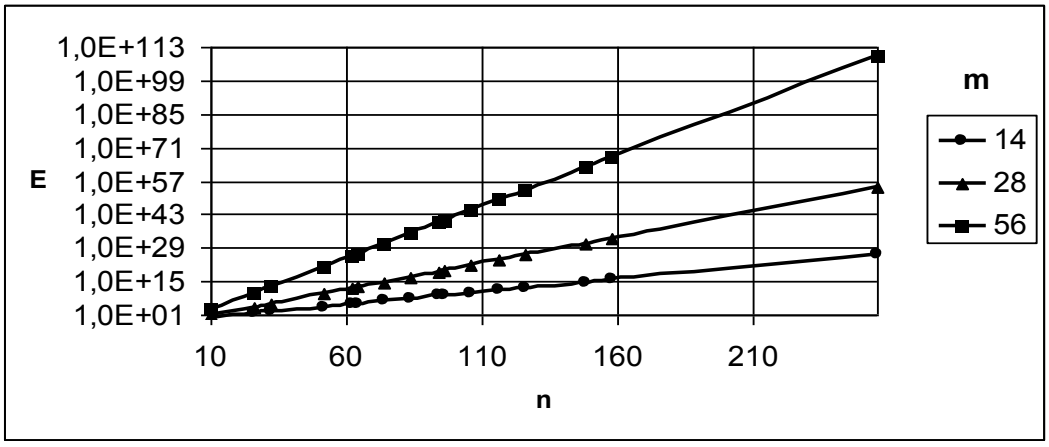

Рис. 3.7 – Графік залежності трудомісткості від кількості символів у множині n для різних значень довжини ключа m для імовірнісного методу ($X_{i+1}=\{11X_i+\pi\}$)

Таким чином, для визначення трудомісткості “відкриття” ключів для множин з кількістю символів n і ключів з довжиною m∈{14,28,56} символів для методу повного перебору, імовірнісного методу з використанням функцій Random та $X_{i+1}=\{11X_i+\pi\}$ відповідно можна скористатись табл. 3.4 та рис. 3.5-3.7. Для визначення часу “відкриття” можна скористатися формулою

$$t = \frac{Ef_0}{vf}, \tag{3.9}$$

де f_0 – тактова частота базового процесора (К6-2-500); f – тактова частота даного процесора; v – швидкодія перебору ключів (для блоково-динамічного шифру і процесора К6-2-500, v=550 вар./с).

Отже, на основі проведеного аналізу можна зробити висновок, що найбільш швидкодійнішим методом криптоаналізу блоково-динамічного шифру є імовірнісний метод з ітеративною функцією випадкової величини $X_{i+1}=\{11X_i+\pi\}$. Встановлено також, що при введені ключа із вказаних в табл. 3.2, 3.3 множин символів різко знижується криптостійкість блоково-динамічного шифру при будь-якій довжині ключа, то при введені ключа потрібно намагатись, щоб його символи входили до якомога більшої кількості множин табл. 3.3, а також специфічні символи псевдографіки та різного роду керуючі символи. Довжину ключа слід вибирати якомога більшою (краще 56 символів – максимум для блоково-динамічного шифру). Виявлено також, що модифікування шифру за допомогою початкового присвоєння змінним datar і datal відповідно правої та лівої частини дати створення файлу, що підлягає шифруванню суттєво збільшує стійкість шифру, оскільки це модифікування зловмиснику невідомо. Навіть, якщо зловмисник знає, що шифр модифіковано датою створення файлу без уточнення порядку присвоєння лівої та правої її частини, то навіть це збільшує стійкість шифру в 2 рази.

Висновки

В результаті порівняння часу “відкриття” Р-блоку імовірнісним методом приходимо до висновку, що функція Xi+1={11Xi+π} швидкодійніша за функцію Random приблизно в 9,313 рази. Але використовуючи навіть потужний суперкомп’ютер 8192xPentium-IV-3000 Р-блок блоково-динамічного шифру в системі «клієнт-банк» можна “відкрити” лише за $6{,}1843*10^9$ років, що дає підстави вважати Р-блок блоково-динамічного шифру стійким до криптоаналізу імовірнісним методом.

Отже, на основі проведеного аналізу визначення стійкості ключа із обмеженого алфавіту символів блоково-динамічного шифру імовірнісним методом та методом повного перебору в системах «клієнт-банк» при встановленні з’єднання, можна зробити висновок, що найбільш швидкодійнішим методом криптоаналізу блоково-динамічного шифру є імовірнісний метод з ітеративною функцією випадкової величини $X_{i+1}=\{11X_i+\pi\}$. Довжину ключа слід вибирати якомога більшою (краще 56 символів – максимум для блоково-динамічного шифру). Виявлено також, що модифікування шифру за допомогою початкового присвоєння змінним datar і datal відповідно правої та лівої частини дати створення файлу, що підлягає шифруванню суттєво збільшує стійкість шифру, оскільки це модифікування зловмиснику невідомо. Навіть, якщо зловмисник знає, що шифр модифіковано датою створення файлу без уточнення порядку присвоєння лівої та правої її частини, то навіть це збільшує стійкість шифру в 2 рази.

Перелік посилань

1. Организация и современные методы защиты информации: Учебное пособие. – М.: Концерн «Банковский Деловой Центр», 1998. – 465 с.

2. Корченко А. Г. Несанкционированный доступ к компьютерным системам и методы защиты: Учебное пособие / Корченко А. Г. – К.: КМУГА, 1998. – 115 с.

3. Мельников В. В. Защита информации в компьютерных системах. – М.: Финансы и статистика, Электроинформ, 1997. – 364 с.

4. Большаков А. А. Основы обеспечения безопасности данных в компьютерных системах и сетях. Методы средства и механизмы защиты данных. / Большаков А. А., Петряев А. Б. – М.: ВИККА им. Можайского СПБ, 1995. – С. 28 – 42.

5. Задірака В. К. Методи захисту банківської інформації / Задірака В. К., Олесюк О. С., Недашковський Н. О. – К.: Вища школа, 1999. – 227 с.

6. Спесивцев А.В. Защита информации в персональных ЭВМ. / Спесивцев А. В., Вегнер В. А., Крутяков А. Ю. – М.: Радио и связь, 1992. – 192 с.

9. 10. Білечук П. Д. Комп’ютерна злочиність: Навчальний посібник. / Білечук П. Д., Романюк Б. В., Цимбалюк В. С. – К.: Атіка, 2002. – 240 с.

11. Кудін А. М. Інформаційна безпека „від А до Я” 3000 термінів та понять / Кудін А. М. – К.: Техніка, 1999. – 264 с.

12. Жельников В. Криптография от папируса до комп’ютера / Жельников В. – М.: ABF, 1997. – 335с.

13. Брикелл Э. Ф. Криптоанализ: Обзор новейших результатов / Брикелл Э. Ф., Одлижко Э. М. – 1988. – №5. – с. 75-94.

14. Я. М. Ніколайчук. Захист даних в функціонально орієнтованих мережах зв'язку. Вісник Хмельницького національного університету / Я. М. Ніколайчук, І. М. Лях, Я. Г. Притуляк – Хмельницький. – 2005. – №4. – 41. – Т.1. – с. 259-263.

16. Дориченко С. А. 25 этюдов о шифрах / Дориченко С. А., Ященко В. В. – М.: ТЭИС, 1994. – 118с.

18. Бабичев С. Г. Основы современной криптографии / Бабичев С. Г., Гончаров В. В., Серов Р. Е. – М.: ЮНИТИ, 2001. – 296 с.

19. В. Ємець, А. Мельник., Р. Попович Сучасна криптографія. Основні поняття / В. Ємець, А. Мельник., Р. Попович – Львів: БаК, 2003. – 144 с

20. Захист інформації в комп'ютерних системах від несанкціонованого доступу:

Навч. Посібник / М. С. Вертузаєв, О. М. Юрченко; За ред. Лаптєва С.Г. – К.: Вид-во Європ. ун-ту, 2001. – 321 с.

21. Молдовян А. А. Криптография / Молдовян А. А., Молдовян Н. А., Советов Б. Я. – М.: Радио и связь, 2000. –224 с.

ДОДАТОК А

Головний модуль реалізації блоково-динамічного шифрування в системі «клієнт-банк»

```
#include "stdafx.h"
#include "emailmanager.h"
#include "CryptEncrypt.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

#define N            16
#define KEYBYTES        8

const DWORD ks0[] =
                {
        0xd1310ba6, 0x98dfb5ac, 0x2ffd72db,     0xd01adfb7,     0xb8e1afed,     0x6a267e96,
        0xba7c9045, 0xf12c7f99, 0x24a19947,     0xb3916cf7,     0x0801f2e2,     0x858efc16,
        0x636920d8, 0x71574e69,         0xa458fea3,     0xf4933d7e,     0x0d95748f,     0x728eb658,
        0x718bcd58,     0x82154aee,     0x7b54a41d,     0xc25a59b5,     0x9c30d539,     0x2af26013,
        0xc5d1b023,     0x286085f0,     0xca417918,     0xb8db38ef,     0x8e79dcb0,     0x603a180e,
        0x6c9e0e8b,     0xb01e8a3e,     0xd71577c1,     0xbd314b27,     0x78af2fda,     0x55605c60,
        0xe65525f3,     0xaa55ab94,     0x57489862,     0x63e81440,     0x55ca396a,     0x2aab10b6,
        0xb4cc5c34,     0x1141e8ce,     0xa15486af,     0x7c72e993,     0xb3ee1411,x636fbc2a,
        0x2ba9c55d,     0x741831f6,     0xce5c3e16,     0x9b87931e,     0xafd6ba33,     0x6c24cf5c,
        0x7a325381,     0x28958677,     0x3b8f4898,     0x6b4bb9af,     0xc4bfe81b,     0x66282193,
        0x61d809cc,     0xfb21a991,     0x487cac60,     0x5dec8032,     0xef845d5d,     0xe98575b1,
        0xdc262302,     0xeb651b88,     0x23893e81,     0xd396acc5,     0x0f6d6ff3,     0x83f44239,
        0x2e0b4482,     0xa4842004,     0x69c8f04a,     0x9e1f9b5e,     0x21c66842,     0xf6e96c9a,
        0x670c9c61,     0xabd388f0,     0x6a51a0d2,     0xd8542f68,     0x960fa728,     0xab5133a3,
        0x6eef0b6c,     0x137a3be4,     0xba3bf050,     0x7efb2a98,     0xa1f1651d,     0x39af0176,
        0x66ca593e,     0x82430e88,     0x8cee8619,     0x456f9fb4,     0x7d84a5c3,     0x3b8b5ebe,
        0xe06f75d8,     0x85c12073,     0x401a449f,     0x56c16aa6,     0x4ed3aa62,     0x363f7706,
        0x1bfedf72,     0x429b023d,     0x37d0d724,     0xd00a1248,     0xdb0fead3,     0x49f1c09b,
        0x075372c9,     0x80991b7b,     0x25d479d8,     0xf6e8def7,     0xe3fe501a,     0xb6794c3b,
        0x976ce0bd,     0x04c006ba,     0xc1a94fb6,     0x409f60c4,     0x5e5c9ec2,     0x196a2463,
        0x68fb6faf,     0x3e6c53b5,     0x1339b2eb,     0x3b52ec6f,     0x6dfc511f,     0x9b30952c,
        0xcc814544,     0xaf5ebd09,     0xbee3d004,     0xde334afd,     0x660f2807,     0x192e4bb3,
        0xc0cba857,     0x45c8740f,     0xd20b5f39,     0xb9d3fbdb,     0x5579c0bd,     0x1a60320a,
        0xd6a100c6,     0x402c7279,     0x679f25fe,     0xfb1fa3cc,     0x8ea5e9f8,     0xdb3222f8,
        0x3c7516df,     0xfd616b15,     0x2f501ec8,     0xad0552ab,     0x323db5fa,     0xfd238760,
        0x53317b48,     0x3e00df82,     0x9e5c57bb,     0xca6f8ca0,     0x1a87562e,     0xdf1769db,
        0xd542a8f6,     0x287effc3,     0xac6732c6,     0x8c4f5573,     0x695b27b0,     0xbbca58c8,
        0xe1ffa35d,     0xb8f011a0,     0x10fa3d98,     0xfd2183b8,     0x4afcb56c,     0x2dd1d35b,
        0x9a53e479,     0xb6f84565,     0xd28e49bc,     0x4bfb9790,     0xe1ddf2da,     0xa4cb7e33,
        0x62fb1341,     0xcee4c6e8,     0xef20cada,     0x36774c01,     0xd07e9efe,     0x2bf11fb4,
        0x95dbda4d,     0xae909198,     0xeaad8e71,     0x6b93d5a0,     0xd08ed1d0,     0xafc725e0,
        0x8e3c5b2f,     0x8e7594b7,     0x8ff6e2fb,     0xf2122b64,     0x8888b812,     0x900df01c,
        0x4fad5ea0,     0x688fc31c,     0xd1cff191,     0xb3a8c1ad,     0x2f2f2218,     0xbe0e1777,
        0xea752dfe,     0x8b021fa1,     0xe5a0cc0f,     0xb56f74e8,     0x18acf3d6,     0xce89e299,
        0xb4a84fe0,     0xfd13e0b7,     0x7cc43b81,     0xd2ada8d9,     0x165fa266,     0x80957705,
        0x93cc7314,     0x211a1477,     0xe6ad2065,     0x77b5fa86,     0xc75442f5,     0xfb9d35cf,
        0xebcdaf0c,     0x7b3e89a0,     0xd6411bd3,     0xae1e7e49,     0x00250e2d,     0x2071b35e,
        0x226800bb,     0x57b8e0af,     0x2464369b,     0xf009b91e,     0x5563911d,     0x59dfa6aa,
        0x78c14389,     0xd95a537f,     0x207d5ba2,     0x02e5b9c5,     0x83260376,     0x6295cfa9,
        0x11c81968,     0x4e734a41,     0xb3472dca,     0x7b14a94a,     0x1b510052,     0x9a532915,
        0xd60f573f,     0xbc9bc6e4,     0x2b60a476,     0x81e67400,     0x08ba6fb5,     0x571be91f,
```

```
	0xf296ec6b,	0x2a0dd915,	0xb6636521,	0xe7b9f9b6,	0xff34052e,	0xc5855664,
	0x53b02d5d,	0xa99f8fa1,	0x08ba4799,	0x6e85076a
};
const DWORD ks1[] =
{
	0x4b7a70e9,	0xb5b32944,	0xdb75092e,	0xc4192623,	0xad6ea6b0,	0x49a7df7d,
	0x9cee60b8,	0x8fedb266,	0xecaa8c71,	0x699a17ff,	0x5664526c,	0xc2b19ee1,
	0x193602a5,	0x75094c29,	0xa0591340,	0xe4183a3e,	0x3f54989a,	0x5b429d65,
	0x6b8fe4d6,	0x99f73fd6,	0xa1d29c07,	0xefe830f5,	0x4d2d38e6,	0xf0255dc1,
	0x4cdd2086,	0x8470eb26,	0x6382e9c6,	0x021ecc5e,	0x09686b3f,	0x3ebaefc9,
	0x3c971814,	0x6b6a70a1,	0x687f3584,	0x52a0e286,	0xb79c5305,	0xaa500737,
	0x3e07841c,	0x7fdeae5c,	0x8e7d44ec,	0x5716f2b8,	0xb03ada37,	0xf0500c0d,
	0xf01c1f04,	0x0200b3ff,	0xae0cf51a,	0x3cb574b2,	0x25837a58,	0xdc0921bd,
	0xd19113f9,	0x7ca92ff6,	0x94324773,	0x22f54701,	0x3ae5e581,	0x37c2dadc,
	0xc8b57634,	0x9af3dda7,	0xa9446146,	0x0fd0030e,	0xecc8c73e,	0xa4751e41,
	0xe238cd99,	0x3bea0e2f,	0x3280bba1,	0x183eb331,	0x4e548b38,	0x4f6db908,
	0x6f420d03,	0xf60a04bf,	0x2cb81290,	0x24977c79,	0x5679b072,	0xbcaf89af,
	0xde9a771f,	0xd9930810,	0xb38bae12,	0xdccf3f2e,	0x5512721f,	0x2e6b7124,
	0x501adde6,	0x9f84cd87,	0x7a584718,	0x7408da17,	0xbc9f9abc,	0xe94b7d8c,
	0xec7aec3a,	0xdb851dfa,	0x63094366,	0xc464c3d2,	0xef1c1847,	0x3215d908,
	0xdd433b37,	0x24c2ba16,	0x12a14d43,	0x2a65c451,	0x50940002,	0x133ae4dd,
	0x71dff89e,	0x10314e55,	0x81ac77d6,	0x5f11199b,	0x043556f1,	0xd7a3c76b,
	0x3c11183b,	0x5924a509,	0xf28fe6ed,	0x97f1fbfa,	0x9ebabf2c,	0x1e153c6e,
	0x86e34570,	0xeae96fb1,	0x860e5e0a,	0x5a3e2ab3,	0x771fe71c,	0x4e3d06fa,
	0x2965dcb9,	0x99e71d0f,	0x803e89d6,	0x5266c825,	0x2e4cc978,	0x9c10b36a,
	0xc6150eba,	0x94e2ea78,	0xa5fc3c53,	0x1e0a2df4,	0xf2f74ea7,	0x361d2b3d,
	0x1939260f,	0x19c27960,	0x5223a708,	0xf71312b6,	0xebadfe6e,	0xeac31f66,
	0xe3bc4595,	0xa67bc883,	0xb17f37d1,	0x018cff28,	0xc332ddef,	0xbe6c5aa5,
	0x65582185,	0x68ab9802,	0xeecea50f,	0xdb2f953b,	0x2aef7dad,	0x5b6e2f84,
	0x1521b628,	0x29076170,	0xecdd4775,	0x619f1510,	0x13cca830,	0xeb61bd96,
	0x0334fe1e,	0xaa0363cf,	0xb5735c90,	0x4c70a239,	0xd59e9e0b,	0xcbaade14,
	0xeecc86bc,	0x60622ca7,	0x9cab5cab,	0xb2f3846e,	0x648b1eaf,	0x19bdf0ca,
	0xa02369b9,	0x655abb50,	0x40685a32,	0x3c2ab4b3,	0x319ee9d5,	0xc021b8f7,
	0x9b540b19,	0x875fa099,	0x95f7997e,	0x623d7da8,	0xf837889a,	0x97e32d77,
	0x11ed935f,	0x16681281,	0x0e358829,	0xc7e61fd6,	0x96dedfa1,	0x7858ba99,
	0x57f584a5,	0x1b227263,	0x9b83c3ff,	0x1ac24696,	0xcdb30aeb,	0x532e3054,
	0x8fd948e4,	0x6dbc3128,	0x58ebf2ef,	0x34c6ffea,	0xfe28ed61,	0xee7c3c73,
	0x5d4a14d9,	0xe864b7e3,	0x42105d14,	0x203e13e0,	0x45eee2b6,	0xa3aaabea,
	0xdb6c4f15,	0xfacb4fd0,	0xc742f442,	0xef6abbb5,	0x654f3b1d,	0x41cd2105,
	0xd81e799e,	0x86854dc7,	0xe44b476a,	0x3d816250,	0xcf62a1f2,	0x5b8d2646,
	0xfc8883a0,	0xc1c7b6a3,	0x7f1524c3,	0x69cb7492,	0x47848a0b,	0x5692b285,
	0x095bbf00,	0xad19489d,	0x1462b174,	0x23820e00,	0x58428d2a,	0x0c55f5ea,
	0x1dadf43e,	0x233f7061,	0x3372f092,	0x8d937e41,	0xd65fecf1,	0x6c223bdb,
	0x7cde3759,	0xcbee7460,	0x4085f2a7,	0xce77326e,	0xa6078084,	0x19f8509e,
	0xe8efd855,	0x61d99735,	0xa969a7aa,	0xc50c06c2,	0x5a04abfc,	0x800bcadc,
	0x9e447a2e,	0xc3453484,	0xfdd56705,	0x0e1e9ec9,	0xdb73dbd3,	0x105588cd,
	0x675fda79,	0xe3674340,	0xc5c43465,	0x713e38d8,	0x3d28f89e,	0xf16dff20,
	0x153e21e7,	0x8fb03d4a,	0xe6e39f2b,	0xdb83adf7
};
const DWORD ks2[] =
{
	0xe93d5a68,	0x948140f7,	0xf64c261c,	0x94692934,	0x411520f7,	0x7602d4f7,
	0xbcf46b2e,	0xd4a20068,	0xd4082471,	0x3320f46a,	0x43b7d4b7,	0x500061af,
	0x1e39f62e,	0x97244546,	0x14214f74,	0xbf8b8840,	0x4d95fc1d,	0x96b591af,
	0x70f4ddd3,	0x66a02f45,	0xbfbc09ec,	0x03bd9785,	0x7fac6dd0,	0x31cb8504,
	0x96eb27b3,	0x55fd3941,	0xda2547e6,	0xabca0a9a,	0x28507825,	0x530429f4,
	0x0a2c86da,	0xe9b66dfb,	0x68dc1462,	0xd7486900,	0x680ec0a4,	0x27a18dee,
	0x4f3ffea2,	0xe887ad8c,	0xb58ce006,	0x7af4d6b6,	0xaace1e7c,	0xd3375fec,
	0xce78a399,	0x406b2a42,	0x20fe9e35,	0xd9f385b9,	0xee39d7ab,	0x3b124e8b,
	0x1dc9faf7,	0x4b6d1856,	0x26a36631,	0xeae397b2,	0x3a6efa74,	0xdd5b4332,
	0x6841e7f7,	0xca7820fb,	0xfb0af54e,	0xd8feb397,	0x454056ac,	0xba489527,
	0x55533a3a,	0x20838d87,	0xfe6ba9b7,	0xd096954b,	0x55a867bc,	0xa1159a58,
	0xcca92963,	0x99e1db33,	0xa62a4a56,	0x3f3125f9,	0x5ef47e1c,	0x9029317c,
	0xfdf8e802,	0x04272f70,	0x80bb155c,	0x05282ce3,	0x95c11548,	0xe4c66d22,
```

```
        0x48c1133f,     0xc70f86dc,     0x07f9c9ee,     0x41041f0f,     0x404779a4,     0x5d886e17,
        0x325f51eb,     0xd59bc0d1,     0xf2bcc18f,     0x41113564,     0x257b7834,     0x602a9c60,
        0xdff8e8a3,     0x1f636c1b,     0x0e12b4c2,     0x02e1329e,     0xaf664fd1,     0xcad18115,
        0x6b2395e0,     0x333e92e1,     0x3b240b62,     0xeebeb922,     0x85b2a20e,     0xe6ba0d99,
        0xde720c8c,     0x2da2f728,     0xd0127845,     0x95b794fd,     0x647d0862,     0xe7ccf5f0,
        0x5449a36f,     0x877d48fa,     0xc39dfd27,     0xf33e8d1e,     0x0a476341,     0x992eff74,
        0x3a6f6eab,     0xf4f8fd37,     0xa812dc60,     0xa1ebddf8,     0x991be14c,     0xdb6e6b0d,
        0xc67b5510,     0x6d672c37,     0x2765d43b,     0xdcd0e804,     0xf1290dc7,     0xcc00ffa3,
        0xb5390f92,     0x690fed0b,     0x667b9ffb,     0xcedb7d9c,     0xa091cf0b,     0xd9155ea3,
        0xbb132f88,     0x515bad24,     0x7b9479bf,     0x763bd6eb,     0x37392eb3,     0xcc115979,
        0x8026e297,     0xf42e312d,     0x6842ada7,     0xc66a2b3b,     0x12754ccc,     0x782ef11c,
        0x6a124237,     0xb79251e7,     0x06a1bbe6,     0x4bfb6350,     0x1a6b1018,     0x11caedfa,
        0x3d25bdd8,     0xe2e1c3c9,     0x44421659,     0x0a121386,     0xd90cec6e,     0xd5abea2a,
        0x64af674e,     0xda86a85f,     0xbebfe988,     0x64e4c3fe,     0x9dbc8057,     0xf0f7c086,
        0x60787bf8,     0x6003604d,     0xd1fd8346,     0xf6381fb0,     0x7745ae04,     0xd736fccc,
        0x83426b33,     0xf01eab71,     0xb0804187,     0x3c005e5f,     0x77a057be,     0xbde8ae24,
        0x55464299,     0xbf582e61,     0x4e58f48f,     0xf2ddfda2,     0xf474ef38,     0x8789bdc2,
        0x5366f9c3,     0xc8b38e74,     0xb475f255,     0x46fcd9b9,     0x7aeb2661,     0x8b1ddf84,
        0x846a0e79,     0x915f95e2,     0x466e598e,     0x20b45770,     0x8cd55591,     0xc902de4c,
        0xb90bace1,     0xbb8205d0,     0x11a86248,     0x7574a99e,     0xb77f19b6,     0xe0a9dc09,
        0x662d09a1,     0xc4324633,     0xe85a1f02,     0x09f0be8c,     0x4a99a025,     0x1d6efe10,
        0x1ab93d1d,     0x0ba5a4df,     0xa186f20f,     0x2868f169,     0xdcb7da83,     0x573906fe,
        0xa1e2ce9b,     0x4fcd7f52,     0x50115e01,     0xa70683fa,     0xa002b5c4,     0x0de6d027,
        0x9af88c27,     0x773f8641,     0xc3604c06,     0x61a806b5,     0xf0177a28,     0xc0f586e0,
        0x006058aa,     0x30dc7d62,     0x11e69ed7,     0x2338ea63,     0x53c2dd94,     0xc2c21634,
        0xbbcbee56,     0x90bcb6de,     0xebfc7da1,     0xce591d76,     0x6f05e409,     0x4b7c0188,
        0x39720a3d, 0x7c927c24,0x86e3725f,      0x724d9db9,     0x1ac15bb4,     0xd39eb8fc,
        0xed545578, 0x08fca5b5, 0xd83d7cd3,     0x4dad0fc4,     0x1e50ef5e,     0xb161e6f8,
        0xa28514d9, 0x6c51133c,0x6fd5c7e7,      0x56e14ec4,     0x362abfce,     0xddc6c837,
        0xd79a3234, 0x92638212,         0x670efa8e,     0x406000e0
};
const DWORD ks3[] =
                {
        0x3a39ce37, 0xd3faf5cf,  0xabc27737,     0x5ac52d1b,     0x5cb0679e,     0x4fa33742,
        0xd3822740, 0x99bc9bbe,         0xd5118e9d,     0xbf0f7315,     0xd62d1c7e,     0xc700c47b,
        0xb78c1b6b, 0x21a19045,         0xb26eb1be,     0x6a366eb4,     0x5748ab2f,     0xbc946e79,
        0xc6a376d2, 0x6549c2c8,0x530ff8ee,      0x468dde7d,     0xd5730a1d,     0x4cd04dc6,
        0x2939bbdb, 0xa9ba4650,         0xac9526e8,     0xbe5ee304,     0xa1fad5f0,     0x6a2d519a,
        0x63ef8ce2, 0x9a86ee22, 0xc089c2b8,     0x43242ef6,     0xa51e03aa,     0x9cf2d0a4,
        0x83c061ba, 0x9be96a4d,0x8fe51550,      0xba645bd6,     0x2826a2f9,     0xa73a3ae1,
        0x4ba99586, 0xef5562e9, 0xc72fefd3,     0xf752f7da,     0x3f046f69,     0x77fa0a59,
        0x80e4a915, 0x87b08601,         0x9b09e6ad,     0x3b3ee593,     0xe990fd5a,     0x9e34d797,
        0x2cf0b7d9, 0x022b8b51,0x96d5ac3a,      0x017da67d,     0xd1cf3ed6,     0x7c7d2d28,
        0x1f9f25cf, 0xadf2b89b,  0x5ad6b472,     0x5a88f54c,     0xe029ac71,     0xe019a5e6,
                        0x47b0acfd, 0xed93fa9b,  0xe8d3c48d,     0x283b57cc,     0xf8d56629,     0x79132e28,
                        0x785f0191, 0xed756055,0xf7960e44,      0xe3d35e8c,     0x15056dd4,     0x88f46dba,
                        0x03a16125, 0x0564f0bd,0xc3eb9e15,      0x3c9057a2,     0x97271aec,     0xa93a072a,
                        0x1b3f6d9b, 0x1e6321f5, 0xf59c66fb,     0x26dcf319,     0x7533d928,     0xb155fdf5,
                        0x03563482, 0x8aba3cbb,0x28517711,      0xc20ad9f8,     0xabcc5167,     0xccad925f,
                        0x4de81751, 0x3830dc8e,0x379d5862,      0x9320f991,     0xea7a90c2,     0xfb3e7bce,
                        0x5121ce64, 0x774fbe32, 0xa8b6e37e,     0xc3293d46,     0x48de5369,     0x6413e680,
                        0xa2ae0810, 0xdd6db224,0x69852dfd,      0x09072166,     0xb39a460a,     0x6445c0dd,
                        0x586cdecf, 0x1c20c8ae,  0x5bbef7dd,     0x1b588d40,     0xccd2017f,     0x6bb4e3bb,
                        0xdda26a7e, 0x3a59ff45,  0x3e350a44,     0xbcb4cdd5,     0x72eacea8,     0xfa6484bb,
                        0x8d6612ae, 0xbf3c6f47,  0xd29be463,     0x542f5d9e,     0xaec2771b,     0xf64e6370,
        0x740e0d8d, 0xe75b1357,         0xf8721671,     0xaf537d5d,     0x4040cb08,     0x4eb4e2cc,
                        0x34d2466a, 0x0115af84,0xe1b00428,      0x95983a1d,     0x06b89fb4,     0xce6ea048,
                        0x6f3f3b82, 0x3520ab82, 0x011a1d4b,     0x277227f8,     0x611560b1,     0xe7933fdc,
        0xbb3a792b, 0x344525bd,         0xa08839e1,     0x51ce794b,     0x2f32c9b7,     0xa01fbac9,
                        0xe01cc87e, 0xbcc7d1f6,  0xcf0111c3,     0xa1e8aac7,     0x1a908749,     0xd44fbd9a,
                        0xd0dadecb, 0xd50ada38,0x0339c32a,      0xc6913667,     0x8df9317c,     0xe0b12b4f,
                        0xf79e59b7, 0x43f5bb3a,  0xf2d519ff,     0x27d9459c,     0xbf97222c,     0x15e6fc2a,
                        0x0f91fc71, 0x9b941525,  0xfae59361,     0xceb69ceb,     0xc2a86459,     0x12baa8d1,
                        0xb6c1075e, 0xe3056a0c,0x10d25065,      0xcb03a442,     0xe0ec6e0e,     0x1698db3b,
```

```
                0x4c98a0be, 0x3278e964, 0x9f1f9532,  0xe0d392df,  0xd3a0342b,  0x8971f21e,
                0x1b0a7441, 0x4ba3348c, 0xc5be7120,  0xc37632d8,  0xdf359f8d,  0x9b992f2e,
                0xe60b6f47, 0x0fe3f11d, 0xe54cda54,  0x1edad891,  0xce6279cf,  0xcd3e7e6f,
                0x1618b166, 0xfd2c1d05, 0x848fd2c5,  0xf6fb2299,  0xf523f357,  0xa6327623,
                0x93a83531, 0x56cccd02, 0xacf08162,  0x5a75ebb5,  0x6e163697,  0x88d273cc,
        0xde966292, 0x81b949d0,  0x4c50901b,  0x71c65614,  0xe6c6c7bd,  0x327a140a,
                0x45e1d006, 0xc3f27b9a, 0xc9aa53fd,  0x62a80f00,  0xbb25bfe2,  0x35bdd2f6,
        0x71126905, 0xb2040222,  0xb6cbcf7c,  0xcd769c2b,  0x53113ec0,  0x1640e3d3,
                0x38abbd60, 0x2547adf0, 0xba38209c,  0xf746ce76,  0x77afa1c5,  0x20756060,
                0x85cbfe4e, 0x8ae88dd8, 0x7aaaf9b0,  0x4cf9aa7e,  0x1948c25c,  0x02fb8a8c,
                0x01c36ae4, 0xd6ebe1f9, 0x90d4f869,  0xa65cdea0,  0x3f09252d,  0xc208e69f,
                0xb74e6132, 0xce77e25b, 0x578fdfe3,  0x3ac372e6
        };

////////////////////////////////////////////////////////////////////
// Construction/Destruction
////////////////////////////////////////////////////////////////////

CCryptEncrypt::CCryptEncrypt(blf_ctx *c,const char key[], WORD keybytes, CTime time)
{
        short   i;
        short   j;
        short   k;
        DWORD           data;
        DWORD           data1;
        DWORD           datar;

        SYSTEMTIME sTime;
        time.GetAsSystemTime(sTime);
        FILETIME fileTime;
        SystemTimeToFileTime(&sTime, &fileTime);

        /* Initialize s-boxes without file read. */
        for(i=0; i<256; i++)
        {
                c->S[0][i] = ks0[i];
                c->S[1][i] = ks1[i];
                c->S[2][i] = ks2[i];
                c->S[3][i] = ks3[i];
        }
        j = 0;
        for (i = 0; i < N + 2; ++i)
        {
                data =  fileTime.dwHighDateTime;
                for (k  = 0; k < 4; ++k)
                {

                        data = (data ^ 8) | key[j];
                        j ++;

                        if (j >= keybytes)
                                j = 0;
                }

                c->P[i] = c->P[i] * data;
        }

        data1 = fileTime.dwHighDateTime;
        datar = fileTime.dwLowDateTime;

        for (i = 0; i < N + 2; i += 2)
        {
                Encipher(c, &data1, &datar);
                c->P[i] = data1;
```

```
                c->P[i + 1] = datar;
        }

        for (i = 0; i < 4; ++i)
        {
                for (j = 0; j < 256; j += 2)
                {
                        Encipher(c, &data1, &datar);
                        c->S[i][j] = data1;
                        c->S[i][j + 1] = datar;
                }
        }
}

CCryptEncrypt::~CCryptEncrypt()
{

}

DWORD CCryptEncrypt::Func(blf_ctx *bc, DWORD x)
{
        WORD a;
        WORD b;
        WORD c;
        WORD d;
        d = (WORD)(x & 0x00FF);
        x >>= 8;
        c = (WORD)(x & 0x00FF);
        x >>= 8;
        b = (WORD)(x & 0x00FF);
        x >>= 8;
        a = (WORD)(x & 0x00FF);
        return ((bc->S[0][a] + bc->S[1][b]) ^ bc->S[2][c]) + bc->S[3][d];
}

void CCryptEncrypt::Encipher(blf_ctx *c, DWORD *x1, DWORD *xr)
{
        DWORD X1;
        DWORD Xr;
        DWORD  temp;
        short        i;
        X1 = *x1;
        Xr = *xr;
        for (i = 0; i < N; ++i)
        {
                X1 = X1 ^ c->P[i];
                Xr = Func(c, X1) ^ Xr;
                temp = X1;

                X1 = Xr;
                Xr = temp;
        }

        temp = X1;
        X1 = Xr;
        Xr = temp;
        Xr = Xr ^ c->P[N];
        X1 = X1 ^ c->P[N + 1];
        *x1 = X1;
        *xr = Xr;
}

void CCryptEncrypt::Decipher(blf_ctx *c, DWORD *x1, DWORD *xr)
{
```

```
	DWORD X1;
	DWORD Xr;
	DWORD  temp;
	short	i;
	X1 = *x1;
	Xr = *xr;
	for (i = N + 1; i > 1; --i)
	{
		X1 = X1 ^ c->P[i];
		Xr = Func(c, X1) ^ Xr;
		/* Exchange X1 and Xr */
		temp = X1;
		X1 = Xr;
		Xr = temp;
	}
	/* Exchange X1 and Xr */
	temp = X1;
	X1 = Xr;
	Xr = temp;
	Xr = Xr ^ c->P[1];
	X1 = X1 ^ c->P[0] ;
	*x1 = X1;
	*xr = Xr;
}

void CCryptEncrypt::EncodeData(blf_ctx *c, DWORD *data, int iSize)
{
	if (iSize == 1) return;

	int iCount = iSize/2;
	for(int i = 0; i < iCount; i++)
		Encipher(c, &data[i], &data[iSize - (i + 1)]);

	if (iCount*2 != iSize)
	{
		Encipher(c, &data[iCount + 1], &data[iCount]);
	}
}

void CCryptEncrypt::DecodeData(blf_ctx *c, DWORD *data, int iSize)
{
	if (iSize == 1) return;

	int iCount = iSize/2;
	if (iCount*2 != iSize)
	{
		Decipher(c, &data[iCount + 1], &data[iCount]);
	}

	for(int i = 0; i < iCount; i++)
		Decipher(c, &data[i], &data[iSize - (i + 1)]);
	}
```

Анотація

Наукова робота

Шифр Traffic

Об’єм роботи складає 30 сторінок. Робота містить 15 рисунків та 3 таблиці. 21 бібліографічне джерело.

На сьогоднішній день саме в банківській сфері спостерігається як позитивний ефект (пов’язаний з впровадженням сучасних автоматизованих технологій обробки інформації та пов’язаного з цим розширення спектру послуг, що надаються, і прискоренням оборотності коштів), так і неминучі негативні вияви. Одне з вразливих місць – пересилання документів між клієнтом і банком.

В зв'язку з цим гостро постає проблема захисту інформації, інтерес до якої обумовлений ще й численними фактами промислового шпигунства та злочинів, здійснених за допомогою комп'ютерних технологій через корпоративні мережі або Інтернет.

Метою досліджень є прискорення шифрування інформації та збільшення криптостійкості блочних алгоритмів в системі «Клієнт-Банк» на основі введення динамічного ключа з використанням програмних та апаратних засобів.

Основні задачі, що визначаються поставленою метою:

1) Реалізувати математичну модель блоково-динамічно шифрування на основі алгоритму Blowfish в системі «клієнт-банк»;

2) Провести криптоаналіз алгоритму блоково-динамічного шифрування в системі «клієнт-банк»;

3) Провести дослідження моделі алгоритму блоково-динамічного шифрування в системі «клієнт-банк» на складність;

4) Розробити методику “відкриття” Р-блоку блоково-динамічного шифру за допомогою функцій випадкової величини в системі «клієнт-банк».

5) На основі запропонованої моделі алгоритму блоково-динамічного шифрування розробити програмне забезпечення у вигляді Марі-клієнта поштового серверу для системи «клієнт-банк».

Об'єктом досліджень є криптографічний захист інформації при встановленні з’єднання від несанкціонованого користування.

Предметом досліджень є методи та засоби блочного шифрування інформації на основі мережі Фейстеля в системі «клієнт-банк».

Методи досліджень базуються на теорії криптографії та криптології, теорії чисел, комбінаториці, теорії множин, криптографії та криптології.