Міністерство освіти і науки України

Департамент науки і освіти Харківської обласної державної адміністрації

Харківське територіальне відділення МАН України

|  |
| --- |
| Відділення: комп’ютерні науки  Секція:  безпека інформаційних та телекомунікаційних систем |

РОЗРОБКА СИСТЕМИ ШИФРУВАННЯ ПОВІДОМЛЕНЬ ВЕЛИКИХ РОЗМІРІВ

|  |
| --- |
| Роботу виконав:  Попович Ярослав Васильович,  учень 10 класу Харківського  Навчально-виховного комплексу №45 «Академічна гімназія» Харківської міської ради Харківської області  Науковий керівник:  Руккас Кирило Маркович,  професор кафедри теоретичної та прикладної інформатики механіко-математичного факультету Харківського національного університету  імені В.Н. Каразіна, доктор технічних наук, доцент |

Харків – 2020

РОЗРОБКА СИСТЕМИ ШИФРУВАННЯ ПОВІДОМЛЕНЬ ВЕЛИКИХ РОЗМІРІВ

Автор роботи: Попович Ярослав Васильович;

Харківське територіальне відділення МАН України;

Харківський навчально-виховний комплекс №45 «Академічна гімназія»

Харківської міської ради Харківської області; 10 клас; м. Харків;

Науковий керівник: Руккас Кирило Маркович,

професор кафедри теоретичної та прикладної інформатики механіко-математичного факультету Харківського національного університету

імені В. Н. Каразіна, доктор технічних наук, доцент

Вступ.

Нині активний розвиток і поява нових інформаційних технологій постійно вимагають підвищення ефективності та рівня захисту інформації. Забезпечити додатковий захист даних дозволяє шифрування - спотворення вихідної інформації за допомогою особливих алгоритмів. Метою даної роботи є вивчення існуючих алгоритмів шифрування даних, розробка власного алгоритму, аналіз ефективності роботи даних алгоритмів на прикладі розробленого чату, а також вибір оптимального рішення для захисту даних в повідомленнях великого розміру.

Результати.

В результаті проведеної роботи розроблено алгоритм шифрування даних, розглянуто алгоритмі Віженера, шифр Вєрнама, шифр DES. Також був створений чат для демонстрації роботи алгоритмів, що дозволяє експериментально оцінювати їх ефективність. На основі экспериментальных даних проведено порівняння роботи запропонованих алгоритмів за кількома критеріями: часом виконання їхньої роботи, пам’яттю, яку вони потребують та кількістю ключів, що знадобиться для їхньої роботи. Також показано, що на великих рядках краще всього використовувати розробленій алгоритм, а на малих рядках можна використовувати, як розробленій алгоритм, так і алгоритм шифрування DES.

**ЗМІСТ**

Вступ……………………………………………………………………………..….......4

Розділ 1. Характеристика алгоритмів шифрування на повідомленнях великого розміру............................................................................................................................. 6

1.1. Табличні шифри……….….……………………………………………………….7

1.2.Потоковий шифр ……….…………………………………………...……………11

1.3. Блочний шифр…………………………………………………………………….12

Розділ 2.Розробка алгоритму шифрування на повідомленнях великого розміру...17

Розділ 3.Порівняльній аналіз алгоритмів шифрування ….…………………….…..19

Висновки….……………………………………………………….…………...............25

Список використаних джерел …………………………………….…………...……. 26

**ВСТУП**

Активний розвиток і поява нових інформаційних технологій постійно вимагають підвищення ефективності та рівня захисту інформації. Забезпечити додатковий захист даних дозволяє шифрування - спотворення вихідної інформації за допомогою особливих алгоритмів. Вони використовуються в різних сферах життєдіяльності, як для захисту власної інформації, так і у медицині, освіті, воєнному ділі та інших.

Також є багато сфер, де обмін інформацією просто потрібний, але старе і слабке устаткування не дозволяє використати багато програмних продуктів. Наприклад, лікарні, де хворі можуть провести місяці, і спілкування в інтернеті практично єдиний спосіб зв'язку із зовнішнім світом. Саме для цих цілей був розроблений чат написаний за допомогою мови програмування python 3.7.4, та за допомогою бібліотеки . Що дозволяє запускати програму на будь-яких пристроях, та потребує малу потужність від клієнта. Програма – кросплатформена, тому, що код на мові програмування python можна запускати на будь-якій системі. Але з’являється проблема – зловмисники можуть потрапити до переписки, та перехоплювати повідомлення. Саме для захисту інформації і були розроблені алгоритми шифрування.



Основна ціль цієї роботи – на прикладі розробленої програми створити новий алгоритм та порівняти його роботу з існуючими алгоритми шифрування на прикладі розробленого чату. Знайти оптимальний спосіб шифрування даних.

Для реалізації цієї задачі порівняємо роботу табличних шифрів, блочних шифрів, та шифру, який не піддається дешифруванню.

Також створимо шифр, який в собі поєднує ідею блочних шифрів та реалізацію найвідомішого табличного шифру – шифру Віженера.

Для виконання цієї задачі:

1. Охарактеризуємо та розглянемо вже відомі алгоритми шифрування;
2. Розробимо новий алгоритм шифрування;
3. Створимо тести для деяких алгоритмів;
4. Порахуємо асимптотику роботи алгоритмів та протестуємо їх в реальних умовах (на прикладі розробленого чату);
5. Порівняємо роботу алгоритмів шифрування;
6. Знайдемо оптимальний варіант для шифрування даних;
7. Охарактеризуємо, в яких умовах краще використовувати які алгоритми.

Дане дослідження може бути застосоване при розробці програм, вхідні дані якої – рядок символів, та потрібно його зашифрувати чи дешифрувати.

**РОЗДІЛ 1**

**ХАРАКТЕРИСТИКА АЛГОРИТМІВ ШИФРУВАННЯ НА ПОВІДОМЛЕННЯХ ВЕЛИКОГО РОЗМІРУ**

Шифрування буває двох типів: симетричні та асиметричні. Симетричні шифри – шифри в яких для шифрування та дешифрування даних використовується один й той самий секретний ключ. В асиметричних шифрах використовується два ключа – відкритий та закритий.

Відкритий ключ може побачити хто завгодно, а секретний доступний тільки сторонам передачі даних. Ідея криптографічних систем з відкритим ключем дуже схожа на ідею однобічних функцій, тобто таких функцій , що по відомому просто знайти , але при відомому значенні функції неможливо знайти значення . Сама однобічна функція безпечна у використовуванні, бо за допомогою неї можна зашифрувати повідомлення, але розшифрувати неможливо. Тому створюються функції, які неможна розшифрувати, але маючи секретне значення це зробити можливо [5].



Також шифри можна поділити на блочні та потокові. Потокові шифри – симетричні шифри, кожен символ яких перетворюється в символ шифротексту, керуючись не тільки ключем, але й положенням символу у відкритому тексті. Блочні шифри – різновид симетричних шифрів, які працюють із групами біт фіксованої довжини – блоками. Далі шифри працюють із кожним блоком окремо (іноді використовуючи результат обробки минулого блоку).

Для асиметричного шифрування даних необхідна генерація великих випадкових чисел (можливо простих), необхідний генератор випадкових чисел, що технічно сильно ускладнює створення програмного продукту, в нашому випадку чату. Тому не будемо зачіпати тему асиметричного шифрування в даному проекті.

* 1. **Табличні шифри**

Табличний шифр [2] – це метод симетричного шифрування, в якому елементи вихідного відкритого тексту міняють місцями згідно із правилом, описаним у таблиці. Елементами тексту можуть бути окремі символи (найпоширеніший випадок), пари букв, трійки букв, комбінування цих випадків і так далі.

Один із найвідоміших представників табличних та потокових шифрів – шифр Віженера. Саме на його прикладі буде розглянутий принцип роботи табличних шифрів.

Шифр Віженера - це метод шифрування, в якому використовуються різні "шифри Цезаря" на основі букв в ключовому слові. У шифрі Цезаря кожна літера зсувається на кілька позицій. Наприклад в шифрі Цезаря при зсуві на 2 символи праворуч A стане C, а X набуде значення Z. Шифр Віженера – поліалфавітний шифр, який складається з послідовності кількох шифрів Цезаря із різними зсувами. Для шифровки повідомлень може використовуватись таблиця алфавітів, чи як її ще називають – квадрат (таблиця) Віженера. Розмір (довжина сторони) таблиці Віженера дорівнює кількості символів в алфавіті. Таким чином застосовуючи шифр Віженера до латинського алфавіту вийде квадрат із стороною в 26 символів, тобто буде 26 різних шифрів Цезаря. На кожному етапі шифрування буде застосований окремий шифр Цезаря, засновуючись на літері ключового та секретного слів.

Принцип роботи шифру складається у наступному. Людина, яка надсилає повідомлення записує ключове слово циклічно, доки довжина його не буде дорівнювати повідомленню. Після цього i-й символ в шифротексті буде дорівнювати перетину i-го символу в повідомленні та ключовому слові у таблиці.

Щоб розшифрувати текст потрібно у стовбці із i-м символом ключа знайти i-й символ шифротексту, тож i-м символом повідомлення буде перший символ у рядку, який містить i-й символ шифротексту [1].

Перейдемо до криптоаналізу. На відміну від шифру Цезаря, шифр Віженера не можна дешифрувати тільки використовуючи частотний аналіз чи атаку по масці. Але його можна дешифрувати, дізнавшись про довжину його ключа. Індекс збігів (тест Касіскі) – один з методів криптоаналізу шифру Віженера. Метод заснований на обчисленні вірогідності того, що дві випадкових частини тексту співпадуть. Цю вірогідність будемо називати індексом збігів [6].

Припустимо рядок х є відкритим текстом або отриманий за допомогою звичайної перестановки. В цьому випадку індекс збігів зручно виразити через вірогідності з’явлення і-го символу. Позначимо їх як , тоді отримаємо наступну формулу:



Так як величини мають певні значення, то для відкритого тексту індекс збігів не залежить від його змісту, а залежить тільки від мови, на якій написаний текст. Більше того, значення вже давно були дослідженні та відомі, що дозволяє розрахувати значення індексу збігів відкритого тексту для різноманітних мов (табл. 1.1).



*Таблиця 1.1*

**Індекс збігів відкритого тексту для різноманітних мов**

|  |  |
| --- | --- |
| **Мова** | **Індекс збігів** |
| російська | 0.0553 |
| англійська | 0.0644 |
| італійська | 0.0738 |
| іспанська | 0.0775 |
| німецька | 0.0762 |
| французька | 0.0778 |

Якщо – випадковий рядок, тоді вірогідність з’явлення кожного символу дорівнює .



Використовуючи першу формулу, отримаємо: .



Цією формулою можна керуватися для оцінки індексу збігів поліалфавітного шифру. Для англійської мови індекс збігів поліалфавітного шифру дорівнює 0.03856, для української (без літери «Ґ») – 0.03125.

Значення індексу збігів для відкритого тексту і для поліалфавітного шифру суттєво відрізняються. Це дозволяє, маючи індекс збігів, визначити, чи отриманий текст з відкритого звичайною перестановкою, чи є поліалфавітним шифром.

Ще одним важливим поняттям є взаємний індекс збігів.

В загальному випадку розглянемо два рядки та із довжинами та відповідно. Алфавіт, як і раніше складається із символів. Взаємним індексом збігів цих рядків називають вірогідність того, що взявши по одному випадковому символу із кожного рядку, вони співпадуть. Нехай – кількість -х символів алфавіту в першому та другому рядках відповідно. Тоді взаємний індекс збігів буде дорівнювати: .



Досить важливим для методу індекс збігів є його окремий випадок, коли обидві рядки отримані зсувом алфавіту відкритого тексту. Позначимо – вірогідності з’явлення -го символу в рядку , – зсув алфавіту рядка відносно алфавіту рядка (ліворуч). Тоді вірогідності появи -го символу алфавіту в рядку дорівнюють (якщо використовувати нумерацію алфавіту рядка ). Для взаємного індексу збігів отримаємо наступну формулу:



Помітимо, що так як зсув циклічний, то і взаємний індекс збігів для зсувів і приймає одне й те саме значення. Нижче приведені значення взаємного індексу збігів в залежності від зсуву для англійської мови (табл. 1.2). Значення наведені для зсувів від 0 до . Як згадувалось вище, на основі цих значень взаємний індекс збігів може бути вирахуваний для будь-якого зсуву.



*Таблиця 1.2*

**Взаємний індекс збігів в залежності від зсуву для англійської мови**



Помітимо, що при нульовому зсуві взаємний індекс збігів набагато більший, ніж при ненульових зсувах, тож по відомому значенню взаємного індексу збігів можна зробити висновок, чи є зсув алфавітів рядків нульовим, або ні.

Розіб’ємо текст на стовбці розміру .



… … … …



Якщо довжині ключа, то кожні два елементи тексту, які відрізняються друг від друга на позицій, , зашифровані одним і тим самим алфавітом. А це означає, що кожний рядок в виписаній вище таблиці отримана з відкритого тексту перестановки. Якщо ж не ділиться на довжину ключа, то рядки є поліалфавітним шифром.



Оскільки індекс збігів для перестановки відкритого тексту та для поліалфавітного шифру суттєво відрізняються, то перебравши різні значення та вираховуючи для кожного з них індекс збігів, можна виділити ті , які ділиться на довжину ключа. Визначити довжину ключа по цим даним буде не важко.



Допустимо, ми визначили довжину ключа . Тепер знайдемо сам ключ. Знову випишемо текст в стовбці розміру .



… … … …



Розглянемо два рядки цієї таблиці. Зсунемо алфавіт одного з рядків на символів та вирахуємо взаємний індекс збігів отриманих рядків. Так як кожний з цих двох рядків отриманий зсувом алфавіту відкритого тексту, то максимум взаємного індексу збігів буде спостерігатися при нульовому кінцевому відносному зсуві.



Тому використовується наступний алгоритм: вираховується взаємний індекс збігів для різних , шукається значення , при якому взаємний індекс збігів буде максимальним. Тоді початковий відносний зсув буде дорівнювати ( – розмір алфавіту). Рахуються відносні зсуви між кожною парою рядків. Так як зсуви рядків таблиці відповідають зсувам літер ключа, то залишається перебрати можливих ключів, і вибрати серед них найбільш вдалий.



* 1. **Потоковий шифр**

Потоковий шифр – група симетричних шифрів, які шифрують кожен символ відкритого тексту незалежно від інших символів.

Із потокових шифрів найбільш криптостійким шифром є шифр Вєрнама. На прикладі цього шифру буде розглянутий принцип роботи потокових шифрів.

Для початку роботи шифру Вернама потрібно виписати весь алфавіт, та проставити біля кожного символу унікальний двійковий код (однакової довжини). Після чого, замінити в тексті символи на двійкові коди. Далі потрібно створити ключ. Ключем буде послідовність випадкових двійкових чисел. Їх кількість потрібна дорівнювати кількості символів в тексті.

Вийшло два рядки. В першому написані двійкові числа, якими замінили символи в тексті, а в другому рядку написано стільки ж випадкових двійкових чисел (довжина кожного числа в першому та другому рядках – однакова). Після цього потрібно застосувати операцію до кожної пари -их елементу в першому та другому рядах. Отримані числа й будуть шифротекстом.



У цього шифру є багато недоліків. По перше, для його роботи потрібен ключ, довжина якого дорівнює довжині тексту. По друге, кожен елемент в ключі потрібен бути випадковим (не псевдовипадковим). А, як відомо, будь який алгоритм, що генерує послідовність є псевдовипадковим, тому потрібно щось, дійсно, випадкове. Наприклад, радіоактивний розпад, чи створений електрогенератором білий шум. Також не можна використовувати одні й ті самі випадкові числа кілька разів поспіль. Ще, при великих розмірах тексту потрібно мати велику кількість випадкових чисел. Наприклад для тексту із ста слів може потребуватись кількасот випадкових чисел.

Для шифру Вернама доведена його абсолютна крипостійкість. Іншими словами, якщо хтось перехопить повідомлення, то єдиний спосіб дізнатися текст – повний перебір ключа. Біля того, цей шифр – єдиний шифр для якого була доведена абсолютна крипостійкість. Але з цього випливає інша проблема. Якщо канал зв’язку прослуховується, то для кожного наступного повідомлення потрібно мати нові випадкові числа обом сторонам, але тоді вони не зможуть їх передати. Натомість, якщо канал зв’язку не прослуховується, то й в шифруванні повідомлень нема ніякого сенсу. Саме тому цей алгоритм припинили використовувати.

* 1. **Блочний шифр**

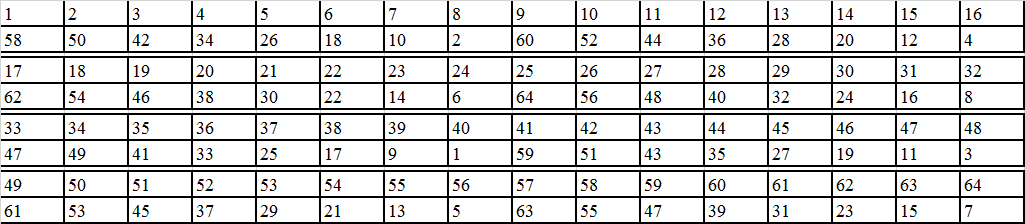
Особливістю блочного шифру є обробка блоку декількох байт за одну ітерацію. Найвідомішими блочними шифрами є AES і DES, вони досить схожі за принципом роботи та мають спільні переваги та недоліки. AES – новий шифр із достатньою довжиною ключа, щоб неможна було підібрати його за невеликий час, але DES використовували, як стандарт, протягом багатьох десятиліть. А також шифр AES можна представити, як просту алгебраїчну структуру. Жоден блочний шифр не має такого простого алгебраїчного представлення, тому невідомо чи призведе це до взлому шифру або ні. Для розробки цього проекту розглянемо саме шифр DES.

Блочні шифри влаштовані досить просто. Для початку потрібно встановити вектор ініціалізації та ключ. За допомогою ключа та вектору ініціалізації формується вектор ініціалізації для наступного блоку, тобто виходить лавинний ефект. І при зміненні будь якого символу зміниться й усе, що йде після нього в кінцевому шифротексті [4]. Дані шифруються блоками по 64 біти. Алгоритм працює із кожним 64-бітним блоком явного тексту, а в результаті повертає 64-бітний блок шифрограми. Для шифрування і дешифрування кожен раз використовується один й той самий алгоритм (за винятком дещо іншої генерації робочих ключів). Довжина ключа кожен раз однакова – 56 біт (в джерельному варіанті використовується 64-бітний ключ, тому що кожний восьмий біт – біт паритету, вони можуть бути винесені в останній байт ключа). Ключем може бути довільна 64-бітна комбінація, яка може бути змінена в будь-який момент часу. Деякі з цих комбінацій вважаються слабкими ключами, оскільки можуть бути легко визначені. Безпечність алгоритму на пряму залежить від безпечності ключа.

У алгоритму на вході подаються 64-бітний блок даних, які переставляються згідно з таблицею (табл. 1.3) [3].

*Таблиця 1.3*

**Перестановка 64-бітного блоку даних**



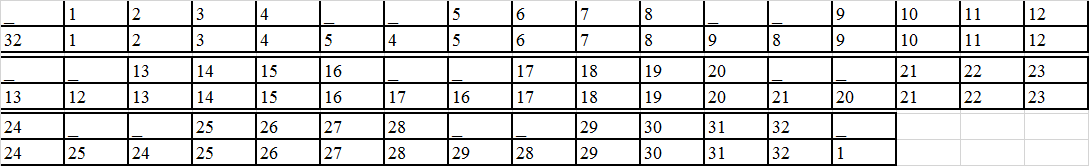
Далі йде повторюється наступні операції шістнадцять разів.

Функція розширює 32-бітовий вектор до 48-бітового вектору шляхом повторення деяких біт з згідно з таблицею (табл. 1.4):



*Таблиця 1.4*

**Розширення 32-бітового**



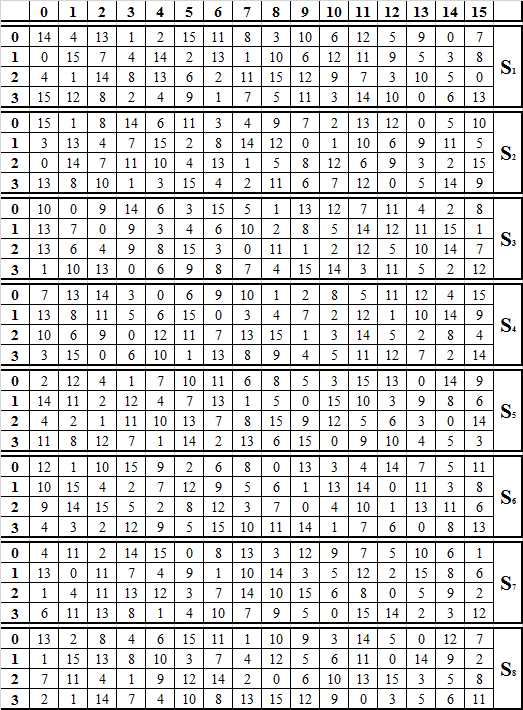
Перший рядок – номера вхідних біт, другий рядок – вхідні біти. Повторення номерів означає повторення біт.

Блок 48 біт -иться з раундовим ключем .



*Таблиця 1.5*

**S-блоки**



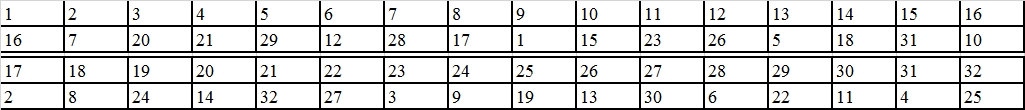
«Розширені» біти використовуються для визначення номера 0-1-2-3 таблиці (ліва колонка).

Далі виконується перестановка (табл. 1.6).

*Таблиця 1.6*

**Перестановка**





За формулою отримуємо значення .

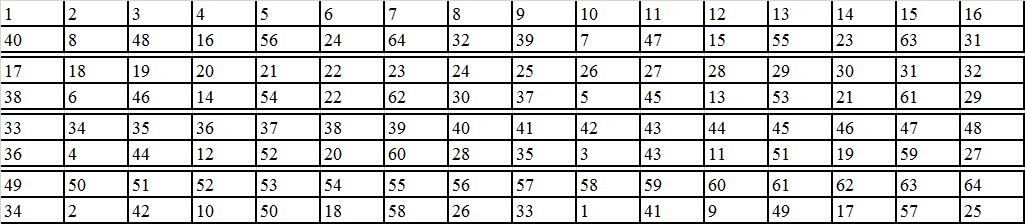


Після 16 раундів перестановка біт виглядає ось так (табл. 1.7).

Вхідний біт ставить на місце номер 40, другий номер 8 і т.п.

*Таблиця 1.7*

**Перестановка біт після 16 раундів**



В DES використовується 16 циклів вихідними даними для кожного з них є біти відкритого тексту(на вході першого раунду), чи 64 біти результату роботи попереднього раунду(у всіх наступних).

Для роботи алгоритму потрібно згенерувати раундові ключі. Ключі отримують з початкового ключа (один символ ASII 8 байтів 64 біт) наступним чином. Вісім бітів, що знаходяться в позиціях 8, 16, 24, 32, 40, 48, 56, 64 додаються в ключ таким чином, щоб кожен байт містив непарне число одиниць [7].



*Таблиця 1.8*

**Початкова перестановка біт ключа**



Ця перестановка визначається двома блоками і по 28 біт кожен. Перші 3 біта є біти 57, 49, 41 розширеного ключа (табл.1.8). А перші 3 біта є біти 63, 55, 47 розширеного ключа. Для раундів , отримують з одним або двома лівими циклічними зрушеннями (табл. 1.9).



*Таблиця 1.9*

**Циклічний зсув**



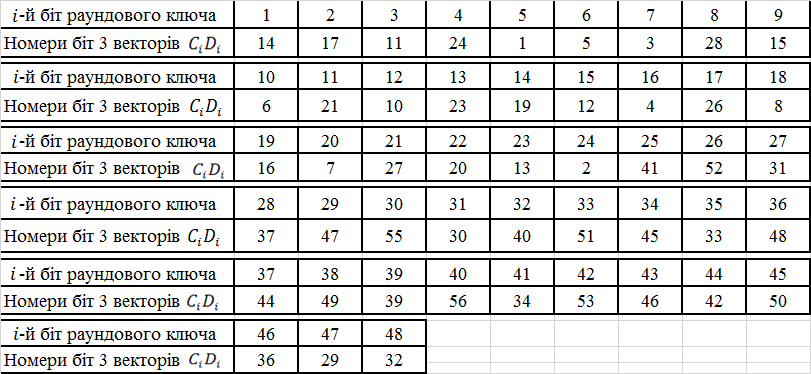
Ключ складається з 48 біт, вибраних з бітів вектору (56 біт) (табл. 1.10). Перший і другий біти є біти 14, 17 вектору .



*Таблиця 1.10*

**Вибір бітів з вектору**





Біти 1-48 використовують для раундового -у.



В цього алгоритму є недолік. Він нестійкий до «лобової» атаки, тобто ключ можна просто підібрати, бо його розмір лише бітів.

**РОЗДІЛ 2**

**РОЗРОБКА АЛГОРИТМУ ШИФРУВАННЯ НА ПОВІДОМЛЕННЯХ ВЕЛИКОГО РОЗМІРУ**

Розроблений шифр поєднує в собі ідею блочних шифрів та табличних. Через це розроблений алгоритм буде стійкий до атак методом Касіскі [2], та до всіх атак, до яких стійкий шифр Віженера. Перед початком роботи алгоритму потрібно встановити алфавіт з яким шифр буде працювати. Для прикладу було взято маленькі латинські літери з по . Після чого алгоритм формує псевдовипадкову послідовність з 26 чисел від 1 до 26 (кожне число зустрічається один раз). Після чого згідно із набором псевдо випадкових чисел формується алфавіт.

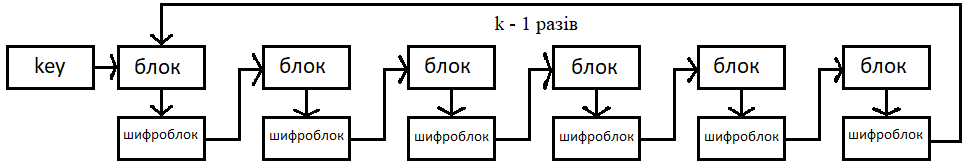


Ця послідовність літер й буде першим рядком в квадраті Віженера. Далі алгоритм приймає відкритий текст, ключ, кількість проходів, і можна встановити довжину кінцевого шифротексту (не менше за довжину відкритого тексту), та потрібно встановити довжину ключа (алгоритму буде простіше працювати якщо обрана довжина ключа буде націло ділитись на обрану довжину шифротексту).

Після встановлення довжини шифротексту та ключа, алгоритм циклічно записує відкритий текст доти його довжина не буде дорівнювати довжині шифротексту, те саме треба зробити і з ключем. Після цього текст розбивається на блоки довжини ключа, та кожен блок шифрується використовуючи шифр Віженера та вектор ініціалізації, як ключ до нього.

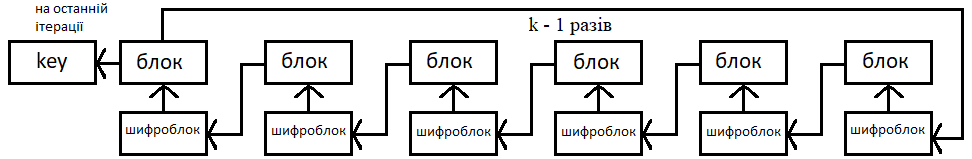
Вектором ініціалізації для першого блоку буде ключ (довжини блока), а для всіх інших блоків вектором ініціалізації буде шифротекст, який вийшов із минулого блоку. Це зроблено для того, щоб домогтись лавинного ефекту, та при зміні будь-якого символу весь шифротекст змінювався. Коли алгоритм дійде до останнього блоку, він присвоїть вектору ініціалізації шифротекст, отриманий з останнього блоку, та буде використовувати його у наступному проході для першого блоку. Так буде повторюватись разів (рис. 2.1). Для дешифрування тексту потрібно пройти за схожим алгоритмом. Весь шифротекст розбивається на блоки довжини ключа, та починаючи з останнього блоку іде у початок.





**Рис 2.1 блок-схема роботи алгоритму**

Кожен блок дешифрується, використовуючи минулий шифроблок, як ключ. Коли алгоритм дійде до першого блоку, він використає останній блок, як вектор ініціалізації. Проходів знов таки буде *k.* На останній ітерації алгоритму для першого блоку використовується ключ, як вектор ініціалізації (рис. 2.2).

****

**Рис 2.2 блок-схема дешифрування алгоритму**

На відміну від шифру Віженера, розроблений шифр не піддається дешифруванню за допомогою метода Касіскі. Це обумовлено тим, що здобувається лавинний ефект під час проходу по блоках.

Для дешифрування повідомлення зловмиснику потрібно знати ключ та кількість проходів по блокам. Якщо зловмисник буде знати ключ, то йому знадобиться підбирати кількість проходів, що при великих значення кількості проходів використає дуже багато часу. Якщо зловмисник знає кількість проходів, то він все одно не знає довжину ключа, тому її потрібно буде підбирати, щоб звести завдання до дешифрування шифру Віженера.

Від довжини ключа залежить кількість блоків, але на час та асимптотику це ніяк не впливає, бо програма проходить по кожному символу в усіх блоках *k* разів. Можна сказати, що чим складніший ключ, тим важче дешифрувати повідомлення зловмиснику.

Треба зауважити, що дізнатися довжину ключа методом Касіскі неможливо (якщо було зроблено більше 1 проходу), тому що для цього йому потрібно буде знати кількість проходів по масиву, та саму довжину ключа.

Алгоритм був реалізований мовою програмування python 3.7.4. Використовування цієї мови програмування обумовлено тим, щоби показати ці алгоритми на прикладі простого консольного чату. В даній мові програмування досить просто працювати із мережею. Одна із можливостей – бібліотека socket. Для реалізації чату був використаний протокол UDP для зв’язку клієнта із сервером.

**РОЗДІЛ 3**

**ПОРІВНЯЛЬНІЙ АНАЛІЗ АЛГОРИТМІВ ШІФРУВАННЯ**

Порівняємо алгоритми шифрування за кількома критеріями: часом виконання їхньої роботи, пам’яттю, яку вони потребують та кількістю ключів, що знадобиться для їхньої роботи.

Почнемо з шифру Віженера. На початку своєї роботи він повинен побудувати квадратну таблицю із стороною, яка дорівнює кількості символів в алфавіті. Якщо використовувати тільки латинські символи, то потрібно створити квадрат, із 676 символів, потім потрібно обробити кожен символ вхідного тексту, що робиться за ітерацій. Тобто цей алгоритм працює досить швидко. Натомість в нього є великий недолік – його можна досить легко взломати, тому він програє в цьому питанні всім іншим алгоритмам в даному списку.

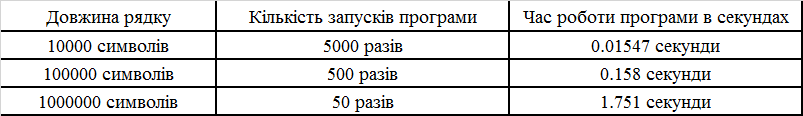


Для аналізу запустимо шифр Віженера для рядку довжиною символів разів. Середній показник часу роботи програми – секунди. При довжині рядку в символів, запустивши програму разів середній час роботи програми дорівнює приблизно секунди. А при довжині рядка в символів, при запуску програми разів, середній час роботи програми дорівнював секунди (табл. 3.1). При збільшенні довжини рядка(більше ніж на символів) час роботи програми буде не лінійно збільшуватися, тому що в мові програмування python 3 чим більше число, тим довше буде проходити операції із ним, тому робити тести на більших рядках немає сенсу.



*Таблиця 3.1*

**Час роботи алгоритму Віженера**

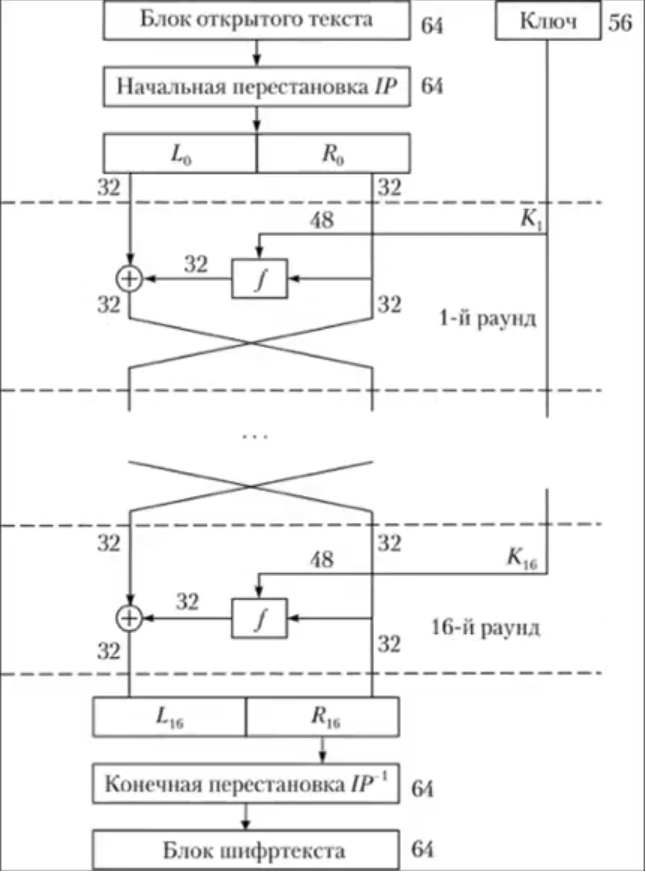


Перейдемо до наступного алгоритму. Для алгоритму Вєрнама, як вже було сказано, доведена абсолютна крипостійкість. Але в нього є великий недолік, а саме те, що для кожного нового повідомлення потрібно обом сторонам мати однакові випадкові числа, а так як потрібно для кожного нового повідомлення мати величезну кількість випадкових чисел, а також обмінятися ними через захищений канал зв’язку, то використовувати цей алгоритм в реальному житті майже неможливо. Тепер кілька слів про його асимптотику. Нехай – час, за який отримується випадкове число. Тоді, якщо довжина повідомлення – символів, то час витрачений на створення випадкових чисел дорівнює , для того, що зробити додавання по модулю 2, потрібно ще ітерацій, тому асимптотика роботи цього алгоритму дорівнює . Також цей алгоритм може займати багато пам’яті, тому що при великому розмірі алфавіту (хоча б символів) потрібно для кожного символу мати 2 числа довжини , де – найближча ступінь двійки більша за кількість символів в алфавіті. І для того, що зберегти шифротекст такого повідомлення знадобиться байтів (де – кількість символів в алфавіті), тому що ще потрібно зберігати чому дорівнює кожен символ алфавіту.



На черзі алгоритм шифрування DES. Якщо говорити про недоліки цього алгоритму, то найважливішим буде довжина його ключа, бо вона дорівнює бітам, тобто можна провести «лобову» атаку на цей алгоритм. Тепер поговоримо про час його роботи. По перше, в самому початку його роботи алгоритм робить перестановку бітів в кожному блоці, тобто робить мінімум ітерацій (на початку роботи із блоком, та у кінці). Після цього алгоритм 16 разів повторює наступну операцію: ділить блок на 2 рівні частини по біти, бере праву частину, та пропускає її через функцію подаючи на вхід ключ розміром 48 біт, після чого лівий блок ксориться із тим, що вийшло з функції, після чого правий блок ставиться без змін на позицію лівого, а лівий (проксорений) опиняється на позиції правого блоку. Але на останній з ітерацій ліва та праві частини не міняються містами. Після чого йде перестановка, яка схожа на ту, що була на початку роботи алгоритму. З цього можна вирахувати асимптотику, яка дорівнює , але ще потрібно сказати за який час працює функція (рис. 3.1).





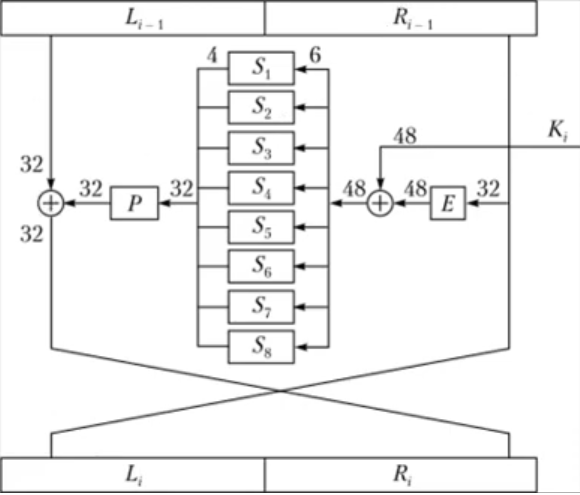
**Рис. 3.1 схема принципу роботи DES**

Функція отримує на вхід бітовий блок. Після чого він проходить через функцію розширення, яка повертає 48 бітовий блок. бітовий блок додається до бітового ключа по модулю 2. Після чого отримується новий блок із біт, який ділиться на підблоків по бітів у кожному. Кожен блок проходить через бокс підстановки, який приймає -ти бітовий блок, а повертає -х бітовий блок. Далі кожен -х бітовий блок додаємо, і отримуємо новий бітовий блок, який проходить через перестановку (рис. 3.2).



Можна сказати, що ця функція працює за лінійний час, бо із кожною частиною блока проходять маніпуляції, та в усьому блоку символи змінюють свої позиції. Тому асимптотика роботи функції буде , де – довжина блоку. Тому асимптотика роботи всієї програми становить (коли рахується асимптотика – не враховуються константи, але, звісно, алгоритм буде працювати трохи довше). Ще тут не врахований час побудови ключа, але його і не потрібно враховувати, бо працює ця побудова для кожного раунду за лінійний час.





**Рис. 3.2 принцип роботи функції**

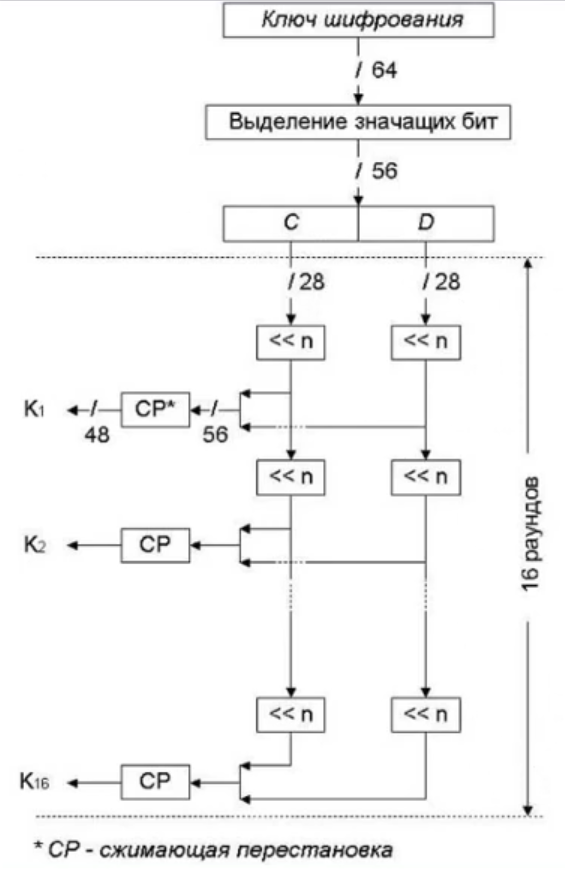


Ключ для кожного раунду формується наступним чином. Спочатку виділяються значущі біти, тому кожний восьмий біт не враховується, та довжина ключа стає рівною не бітам, як на вході, а бітам. Далі ключ ділиться на дві рівні частини по біт в кожній. Треба зазначити, що виділення бітів – не видалення кожного восьмого, а ще й перестановка. В кожному раунді кожний з бітів у блоках зсуваються на біт ліворуч(при яких значеннях бітів на скільки потрібно зсунути елемент описано в розділі про алгоритм DES, і також там приведена таблиця). Після цього ці блоки склеюються, і входять до стискаючої функції, яка перетворює бітний блок у бітний. І так відбувається раундів, для кожного з яких генерується свій бітний ключ (рис. 3.3).



Залишилося проаналізувати роботу розробленого алгоритму. То, на скільки важко буде розшифрувати цей алгоритм залежить від кількох факторів. По перше, від кількості проходів по блокам, по друге, від складності й довжини ключа. Час роботи алгоритму залежить тільки від кількості проходів по блоках та довжині відкритого тексту. Якщо довжина тексту дорівнює символів, а кількість проходів – , то асимптотика роботи алгоритму дорівнює , бо в сумі для кожного блоку буде застосований алгоритм Віженера разів, а як було сказано раніше, асимптотика шифру Віженера – лінійна, тож алгоритм буде працювати ітерацій, де – довжина кожного блоку, а – їх кількість. Так як то й асимптотика дорівнює .





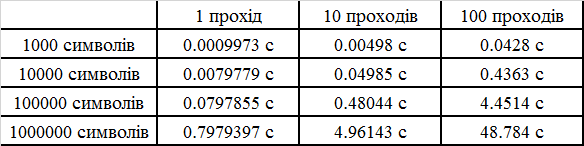
**Рис. 3.3 принцип побудови ключа**

Подивимось на результат роботи алгоритму в різних умовах (табл. 3.2). Всі результати досить схожі друг на друга. Також можна зауважити, що результат при довжині рядка в 1000 символів при 1 проході не схожий на інші, це можна пояснити, як не точність функції в мові програмування python 3. А на великих тестах результати можуть бути непропорційно більшими, по тим самим причинам чому і в шифрі Віженера.



*Таблиця 3.2*

**Результат роботи алгоритму в різних умовах**



Так як мета роботи обрати найкращій шифр для чату, то шифр Вєрнама не підходить, бо постійно потрібно буде генерувати велику кількість випадкових чисел, та обмінюватися німі, що зробити не вийде, якщо канал зв’язку прослуховується. Шифр Віженера також не підходить, тому що вже є способи по дешифруванню повідомлень, які зашифровані цим шифром. В шифрі DES недолік в тому, що ключ можна просто підібрати, якщо є багато комп’ютерів із великою потужністю. А в розробленому алгоритмі є інший недолік. Як показує таблиця, коли довжина рядку досить велика, зробити багато проходів по блокам не вийде, тому що програма буде досить довго працювати, натомість із рядками порядку символів та менше, алгоритм показує себе добре. Хоча дивлячись на асимптотику шифру DES, можна побачити, що він на великих рядках буде також досить довго працювати. Але перевага мого алгоритму в тому, що можна змінювати швидкість роботи програми, зменшуючи або збільшуючи кількість проходів по блокам, тобто чим довше буде працювати програма, тим надійніше буде зашифровано повідомлення.



**ВИСНОВКИ**

Були розглянуті різні алгоритми шифрування на прикладі створеного чату. В кожного алгоритму є свої переваги та недоліки.

1. Шифр Віженера працює швидше за всіх інших конкурентів в даному списку, але він піддається до дешифрування.
2. Алгоритм Вєрнама – єдиний алгоритм для якого була доведена абсолютна криптостійкість. Але для його роботи потрібно багато випадкових чисел, тому його використовувати в написанні чату неможливо.
3. Алгоритм шифрування DES був стандартом протягом багатьох десятків років, та його криптостійкість залежить від важкості його ключа, натомість його ключ можна підібрати, якщо маєш багато потужності та часу.
4. Розробленій алгоритм може добре зашифрувати повідомлення, але витратити на це багато часу, або може зробити це гірше, але швидше.
5. На рядках порядку символів краще використовувати розроблений алгоритм, тому що алгоритм DES буде дуже довго працювати, а шифр Віженера зашифрує повідомлення не надійно.



1. На рядках більшої довжини якщо обирати між шифром Віженера та розробленим алгоритмом, краще обрати розроблений алгоритм, тому що, навіть, зробивши проходи по блокам він зашифрує повідомлення надійніше, ніж шифр Віженера, а працювати, в цьому випадку, вони будуть приблизно один й той самий час.



1. На рядках меншої довжини не слід використовувати шифр Віженера, бо при таких маленьких рядках шифр DES буде працювати приблизно той самий час, що і шифр Віженера (у секундах). На малих рядках краще використовувати розроблений алгоритм чи шифр DES, бо вони обидва швидко закінчать роботу.
2. На великих рядках краще всього використовувати саме розроблений алгоритм, а на малих рядках можна використовувати, як розроблений алгоритм, так і алгоритм шифрування DES.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. М.: Гелиос АРВ, 2001. 479 с.
2. Баричев С.Г., В. В. Гончаров В.В., Р. Е. Серов Р.Е. Основы современной криптографии. М.: ДИАЛОГ-МИФИ, 2011. 175 с.
3. Бауэр Ф. Расшифрованные секреты. Методы и принципы криптологии. М.: Мир, 2007. 550 c.
4. Зубов А.Ю. Криптографические методы защиты информации. Совершенные шифры: Учебное пособие. М.: Гелиос АРВ, 2005. – 192 с.
5. Мао B. Современная криптография. Теория и практика. М.: Вильямс, 2005. 763 с.
6. Сингх С. Книга шифров. Тайная история шифров и их расшифровки. М.: Аст, Астрель, 2006. 447 с.
7. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: Триумф, 2003. 806 с.

**ІНТЕРНЕТ-РЕСУРСИ**

<http://bit.nmu.org.ua/ua/student/metod/cryptology/лекция%206.pdf>

https://uk.wikipedia.org/wiki/Шифр\_Віженера