

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ІНЖЕНЕРНО-ТЕХНІЧНИЙ ФАКУЛЬТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

МЕТОДИЧНІ ВКАЗІВКИ І ЗАВДАННЯ
ДО ЛАБОРАТОРНИХ РОБІТ З КУРСУ

ЗАХИСТ ІНФОРМАЦІЇ В КОМП'ЮТЕРНИХ СИСТЕМАХ

для студентів 3-4-го курсів інженерно-технічного факультету
спеціальності «комп'ютерна інженерія»

Методичні вказівки і завдання до лабораторних робіт
з курсу «Захист інформації в комп'ютерних системах»
для студентів 3-4-го курсу інженерно-технічного факультету
спеціальності «комп'ютерна інженерія»

Укладачі: Гапак О. М. – канд. пед. наук, доцент кафедри комп'ютерних систем та мереж.

Рецензент: Глебена М. І. – канд. фіз.-мат. наук, доцент кафедри системного аналізу та теорії оптимізації УжНУ.

Відповідальний за випуск – Турянця І.І., канд. фіз.-мат. наук, доцент, декан інженерно-технічного факультету.

Дані методичні вказівки розглянуто та схвалено на засіданні кафедри комп'ютерних систем та мереж, протокол № 8 від 26 лютого 2019 року

ВСТУП

Мета курсу – ознайомлення студентів із основами захисту інформації в комп'ютерних системах.

Завдання дисципліни – сформувати погляд на захист інформації і криптографію як на систематичну науково-практичну діяльність, що носить прикладний характер. Сформувати базисні теоретичні поняття та практичні навички щодо використання класичних шифрів, блочних шифрів, асиметричних криптосистем.

Для освоєння даної дисципліни необхідні знання із курсів: «Програмування», «Системне програмування», «Дискретна математика», «Лінійна алгебра та аналітична геометрія», «Теорія інформації і кодування», «Теорія ймовірності і математична статистика».

У результаті вивчення дисципліни студент повинен знати: особливості інформації як об'єкту захисту, перелік та особливості основних загроз інформації в комп'ютерних системах, підходи до розробки політики безпеки комп'ютерної системи, основні принципи захисту інформації, порядок формування комплексу засобів захисту інформації, принципи управління доступом до захищених ресурсів комп'ютерної системи, криптографічні методи і засоби захисту інформації, способи та засоби захисту інформації від витоку технічними каналами, порядок визначення вимог щодо захисту інформації в комп'ютерній системі, основні положення нормативної бази системи захисту інформації в комп'ютерних системах. Вміти: застосовувати математичні методи описання і дослідження криптосистем; оцінювати криптографічну стійкість шифрів.

Для засвоєння матеріалу курсу необхідна активна самостійна робота студентів.

ЛАБОРАТОРНА РОБОТА 1

Тема: Захист інформації за допомогою пароля

Мета: дослідження парольного захисту документів MS Word, Excel, Access та архівів даних, а також дослідження методів протидії атакам на пароль.

Теоретична частина

1. Атаки на пароль

На сьогоднішній день пароль є найбільш прийнятним і тому найбільше часто використовуваним засобом установадження дійсності, заснованим на знаннях суб'єктів доступу.

У будь-якій критичній системі помилки людини-оператора є чи ледве не самими дорогими і розповсюдженими. У випадку криптосистем, непрофесійні дії користувача зводять нанівець самий стійкий криптоалгоритм і саму коректну його реалізацію і застосування.

У першу чергу це зв'язано з вибором паролів. Очевидно, що короткі або осмислені паролі легко запам'ятовуються людиною, але вони набагато простіші для розкриття. Використання довгих і безглузких паролів безумовно краще з погляду криптостійкості, але людина звичайно не може них запам'ятати і записує на папірці, що потім або губиться, або попадає в руки зловмисників. Саме з того, що недосвідчені користувачі звичайно вибирають або короткі, або осмислені паролі, існують два методи їхнього розкриття: атака повним перебором і атака по словнику.

Захищеність пароля при його підборі залежить, у загальному випадку, від швидкості перевірки паролів і від розміру повної безлічі можливих паролів, що, у свою чергу, залежить від довжини пароля і розміру застосовуваного алфавіту символів. Крім того, на захищеність сильно впливає реалізація парольного захисту.

У зв'язку з різким ростом обчислювальних потужностей атаки повним перебором мають набагато більше шансів на успіх, ніж раніше. Крім того, активно використовуються розподілені обчислення, тобто рівномірний розподіл задачі на велику кількість машин, що працюють паралельно. Це дозволяє багаторазово скоротити час злому.

Чим більше довжина пароля, тим більшу безпеку буде забезпечувати система, тому що будуть потрібні великі зусилля для його відгадування. Це можна представити в термінах очікуваного часу розкриття пароля або очікуваного безпечного часу. Очікуваний безпечний час (T_6) — половина добутку числа можливих паролів і часу, необхідного для того, щоб спробувати кожен пароль з послідовності запитів. Представимо це у виді формули:

$$T_{\phi} = \frac{A^S \bullet t}{2},$$

де t — час, необхідний на спробу введення пароля, рівний E/R ;

E — число символів у переданому повідомленні при спробі одержати доступ (включаючи пароль і службові символи);

R — швидкість передачі (символи/хв) у лінії зв'язку;

S — довжина пароля;

A — число символів в алфавіті, з яких складається пароль.

Якщо після кожної невдалої спроби підбору автоматично передбачається десятисекундна затримка, то безпечний час різко збільшується.

Тому при використанні аутентифікації на основі паролів захищеною системою повинні дотримуватися наступні правила:

- не дозволяються паролі менше 6–8 символів;
- паролі повинні перевірятися відповідними контролерами;
- символи пароля при їхньому введенні не повинні з'являтися в явному виді;
- після введення правильного пароля видається інформація про останній вхід у систему;
- обмежується кількість спроб уведення пароля;
- уводиться затримка часу при неправильному паролі;
- при передачі по каналах зв'язку паролі повинні шифруватися;
- паролі повинні зберігатися в пам'яті тільки в зашифрованому вигляді у файлах, недоступних користувачам;
- користувач повинний мати можливість самому змінювати пароль;
- адміністратор не повинний знати паролі користувачів, хоча може їх змінювати;
- паролі повинні періодично мінятися;
- установлюються терміни дії паролів, після закінчення яких треба зв'язатися з адміністратором.

2. Проблема вибору пароля

Вибір довжини пароля в значній мірі визначається розвитком технічних засобів, їхньої елементної бази та її швидкодією. В даний час широко застосовуються багатосимвольні паролі, де $S > 10$. У зв'язку з цим виникають питання: як і де його зберігати і як зв'язати його з аутентифікацією особистості користувача? На ці питання відповідає комбінована система паролів, у якій код пароля складається з двох частин. Перша частина складається з 3–4-х десяткових знаків, якщо код цифровий, і більше 3–4-х, якщо код буквений, котрі легко запам'ятати людині. Друга частина містить кількість знаків, обумовлена вимогами до захисту і можливостями технічної реалізації системи, вона міститься на фізичному носії і визначає ключ-пароль, розрахунок довжини коду якого ведеться по зазначеній вище методиці. У цьому випадку частина пароля буде недоступна для порушника.

Однак при розрахунку довжини коду пароля не слід забувати про те, що при збільшенні довжини пароля не можна збільшувати періодичність його зміни. Коди паролів необхідно змінювати обов'язково, тому що за великий період часу збільшується імовірність їхнього перехоплення шляхом прямого розкрадання носія, зняття його копії, примуса людини. Вибір періодичності необхідно визначати з конкретних умов роботи системи, але не рідше одного разу в рік. Причому бажано, щоб дата заміни і періодичність повинні носити випадковий характер.

Для перевірки уразливості паролів використовуються спеціальні контролери паролів. Наприклад, відомий контролер Кляйна, здійснює спроби злому пароля шляхом перевірки використання як пароль вхідного імені користувача, його ініціалів і їхніх комбінацій, перевірки використання як пароль слів з різних словників, починаючи від найбільш уживаних як пароль, перевірки різних перестановок слів, а також перевірки слів мовою користувача-іноземця.

Приведений аналіз дозволяє сформулювати наступні правила зниження уразливості паролів і спрямовані на протидію відомим атакам на них:

- розширюйте застосовуваний у паролі алфавіт — використовуйте прописні і малі літери латинського і російського, українського алфавітів, цифри і знаки;

- не використовуйте в паролі осмислені слова;

- не використовуйте повторювані групи символів;

- не застосовуйте паролі довжиною менш 6–8 символів, тому що запам'ятати їх не представляє великої праці, а пароль саме потрібно запам'ятовувати, а не записувати. По тій же причині не має змісту вимагати довжину неосмисленого пароля більш 15 символів, тому що запам'ятати його нормальній людині практично неможливо;

- не використовуйте той самий пароль у різних системах, тому що при компрометації одного пароля постраждають усі системи;

- перевіряйте паролі перед їхнім використанням контролерами паролів.

Для складання пароля можна дати рекомендації, якими користуватися треба дуже обережно:

- виберіть кілька рядків з пісні або поеми (тільки не ті, котрі Ви повторюєте першому зустрічному) і використовуйте першу (або другу) букву кожного слова — при цьому пароль повинний мати велику довжину (більш 15 символів), інакше потрібно змінювати регістри букв, застосовувати латинські букви замість російських або навпаки, можна вставляти цифри і знаки;

- замініть в слові із семи-восьми букв одну приголосну й одну або дві голосні на знаки або цифри. Це дасть вам слово-абракадабру, що звичайно вимовне і тому легко запам'ятовується.

3. Захист документів MS OFFICE та архівів

Часто виникає необхідність захисту створених документів чи архівів з даними власним паролем. Для цього існує ряд як програмних засобів, так і засобів вбудованих безпосередньо у ті чи інші програмні продукти. Розглянемо вбудовані в MS OFFICE засоби аутентифікації.

3.1. Якщо виникає необхідність захистити свій файл від інших користувачів, MS Word дозволяє це зробити за допомогою спеціального пароля.

Для запобігання відкриттю документа використовується **Пароль** для відкриття файлу. Встановивши його, неможливо відкрити файл, не ввівши попередньо пароль. У іншому випадку можна дозволити відкрити файл, працювати з ним, але при цьому захистити його від змін. Тоді встановлюється **Пароль** дозволу запису.

3.2 Аналогічно до документів MS Word в табличному редакторі Excel є можливість встановити пароль на відкриття документа та на внесення у нього змін.

MS Excel надає можливість також захистити як цілу Книгу так і окремо створений Лист.

3.3. При захисті бази даних (БД) MS Access можливі два наступні способи: захист на рівні пароля; захист на рівні користувача.

3.4. Багато користувачів архівують свої дані за допомогою популярних архіваторів RAR, ARJ, ZIP. Потім цим архівам задається пароль, а їх вміст шифрується. Цей спосіб захисту кращий, ніж у MS Word. У нього є одна перевага – не потрібні додаткові криптографічні програми, оскільки шифрування здійснюється у архіваторі.

Хоча захист в архівах є досить надійний, проте далеко не досконалий. В алгоритмах шифрування, які застосовуються у ZIP та інших архіваторах, виявлені «діри», що дозволять зламати архів, не тільки підібравши пароль, але й іншим способом.

Архіватор WinRAR займає провідне місце серед утиліт свого класу за надійність шифрування вмісту файлів.

4. Злам паролів в документах MS Word і MS Excel

Існують десятки програм, за допомогою яких можна розкрити "запаролений" документ за лічені секунди. Прикладом такої програми є **Advanced Office XP Password Recovery**. Інтерфейс даної програми зображений на рис. 11 – 13.

Для завантаження файлу з невідомим паролем необхідно запустити програму, зайти в —»**file** —»**open file**. В діалоговому вікні вибрати файл текстового редактора **WORD**. Вибати в **minien** мінімальну довжину пароля, в **maxien** - максимальну. В розділі **rangeopts** вибрати шляхом встановлення галочок з яких символів може складатися пароль. Зайти в **recovery** і натиснути **start**. Далі ви побачите процес визначення паролю шляхом перебору всіх

можливих паролів. При успішному виконанні операції ви побачите діалогове вікно з визначеним паролем та деяку статистику пошуку паролю.

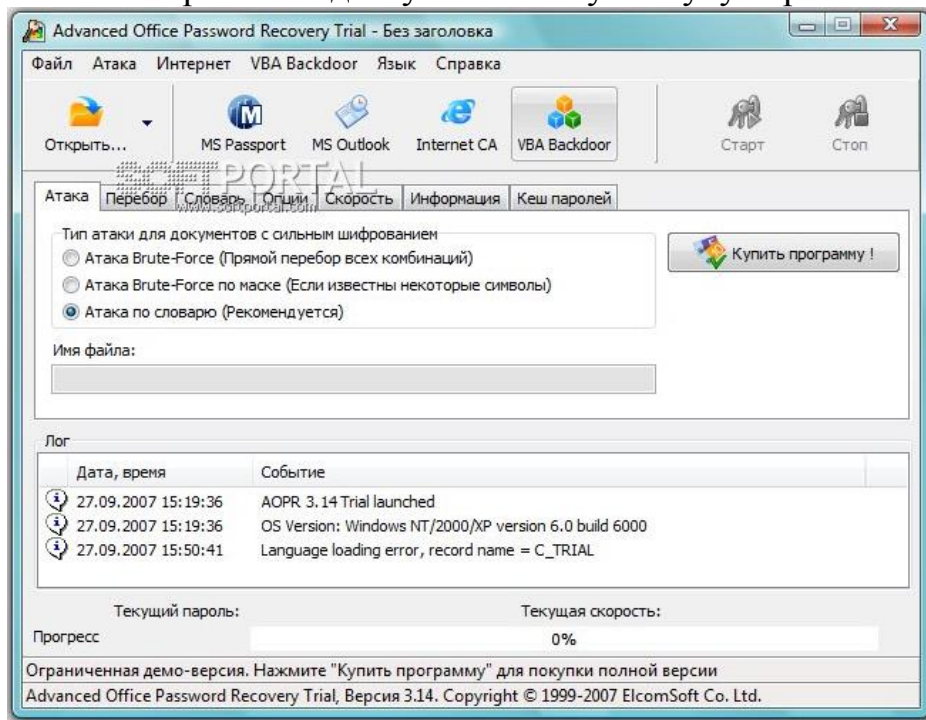


Рисунок 11 – Вибір атаки

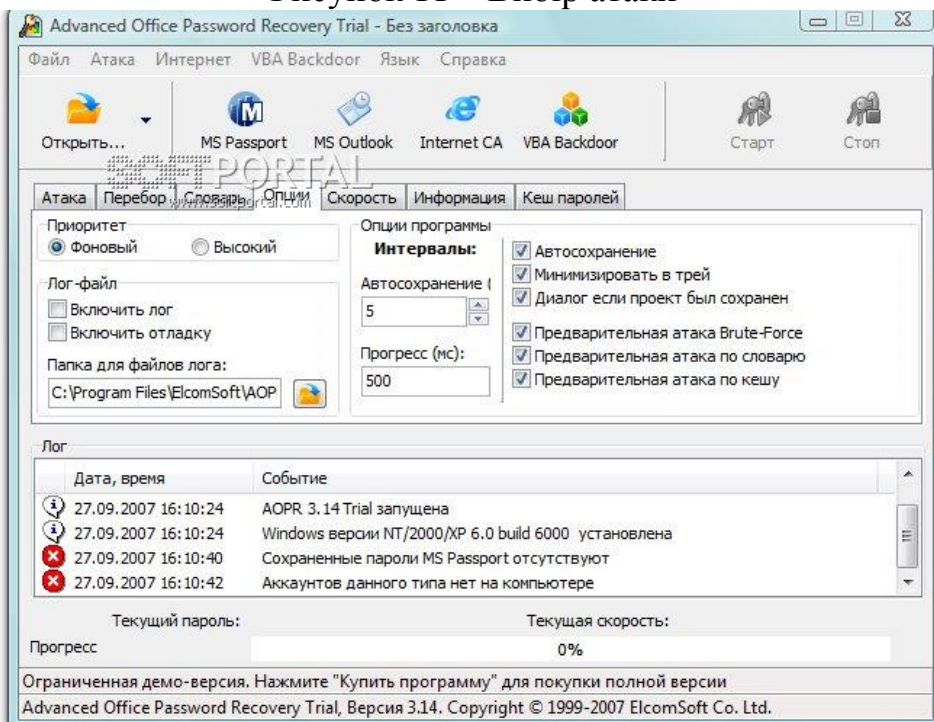


Рисунок 12 – Вибір оптимальних параметрів

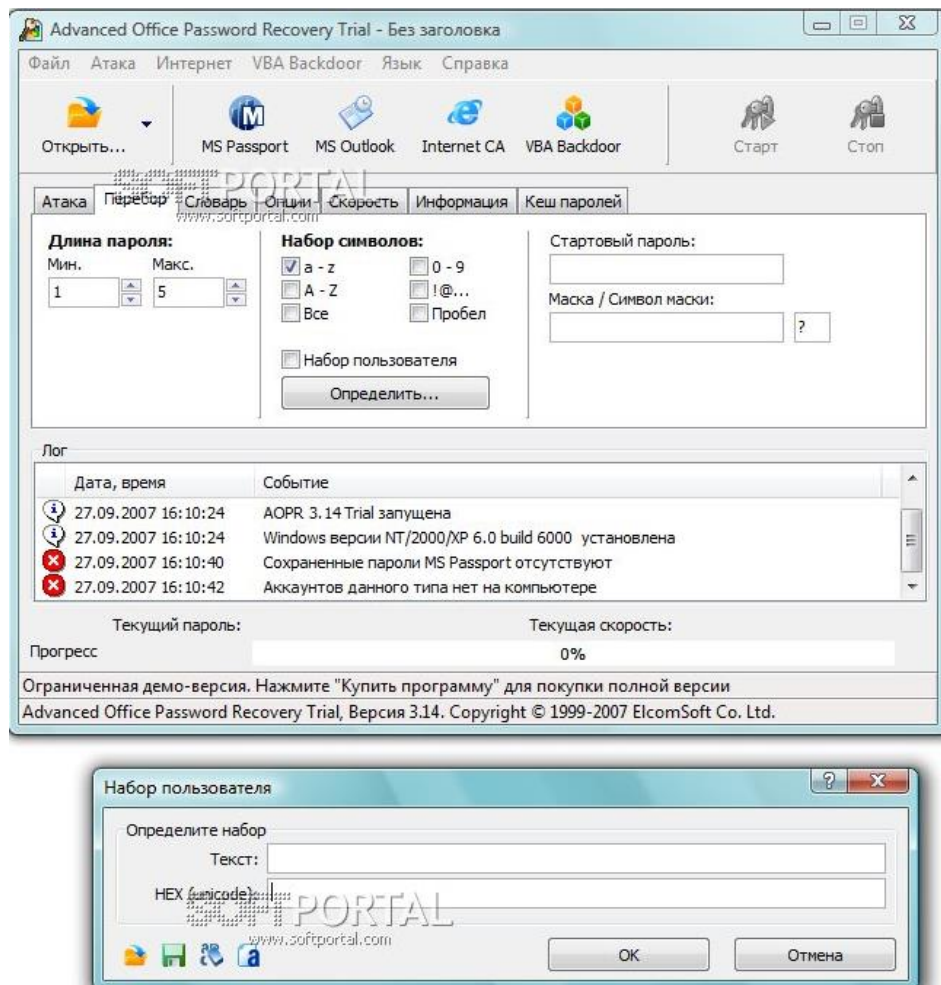


Рисунок 13 – Задання параметрів повного перебору паролів

5. Робота з програмами злому на прикладі AZPR

У даній лабораторній роботі використовується програмний продукт для розкриття закритих паролем архівів: Advanced ZIP Password Recovery .

Програма AZPR використовується для відновлення забутих паролів ZIP-архівів. Існує два способи розкриття паролів: перебір (brute force) і атака по словнику (dictionary-based attack).

Панель керування:

- кнопки Відкрити (Open) і Зберегти (Save) дозволяють працювати з проектом, у якому зазначений файл, що розкривається, набір символів, останній протестований пароль. Це дозволяє призупиняти і відновляти розкриття.
- кнопки Старт (Start) і Стоп (Stop) дозволяють відповідно починати і закінчувати підбір пароля.
- кнопка Набір (Range) дозволяє задати свою множину символів, якщо відомі символи, з яких складається пароль.
- кнопка Довідка (Help) виводить допомогу по програмі.
- кнопка О AZPR виводить інформацію про програму.
- кнопка Вихід (Quit) дозволяє вийти з програми

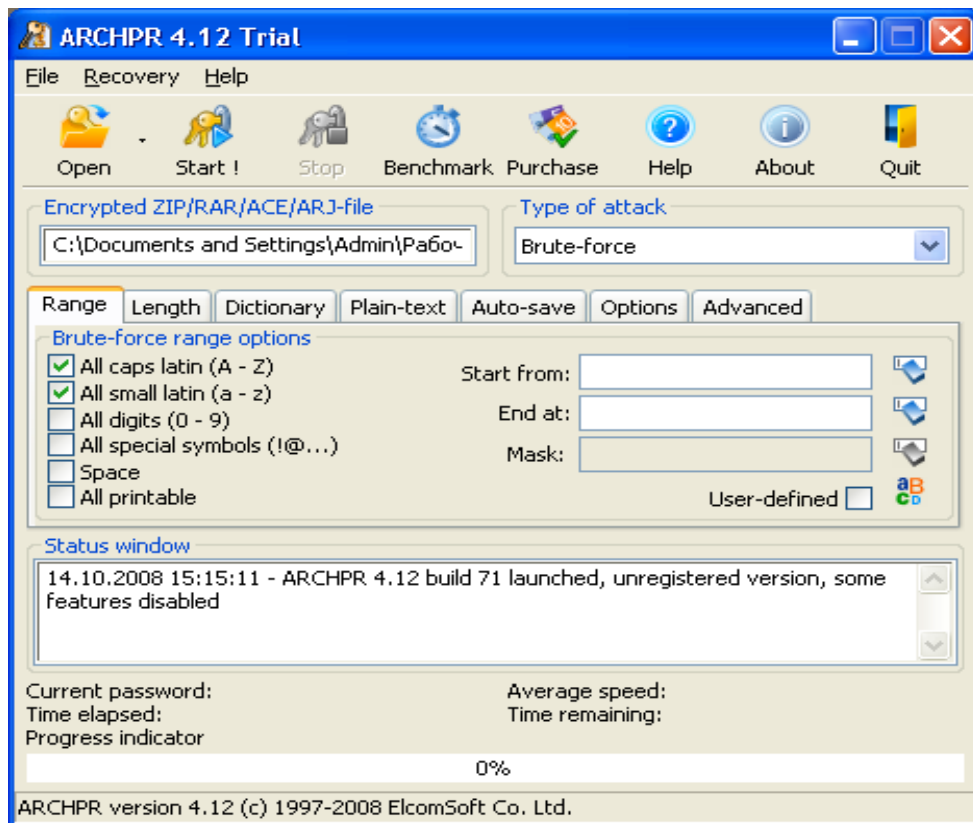


Рисунок 14 – Вікно програми

Розглянемо можливості програми:

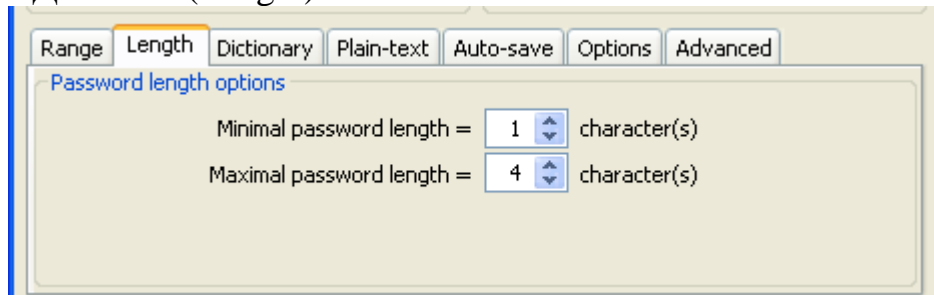
Вибирається архів для розкриття і тип атаки (рис. 14).

Вибираються параметри роботи:

- Закладка Набір (Range)

Програма дозволяє вибрати область перебору (набір символів). Це значно скорочує час перебору. Можна використовувати набір користувача, заданий за допомогою кнопки Набір. Можна обмежити кількість тестованих паролів, задавши початковий пароль. У випадку якщо відомо частину пароля, дуже ефективна атака по масці. Потрібно вибрати відповідний тип атаки, після цього стане доступним поле маски. У ньому потрібно увести відому частину пароля у виді P?s?W?r? , де на місці невідомих символів потрібно поставити знак питання. Можна використовувати будь-який інший символ, увівши його в поле символ маски.

Закладка Довжина (Length)



Дозволяє вибрати довжину пароля.

- Закладка Словник (Dictionary)

Дозволяє вибрати файл-словник. Вибирайте файл English.dic, він містить набір англійських слів і набори символів, що найбільше часто використовуються як паролі.

- **Закладка Автозберігання (Avto-save)**

Можна вибрати ім'я файлу для збереження результатів роботи й інтервал автозбереження.

- **Закладка Опції (Options)**

Вибирається пріоритет роботи (фоновий або високий), інтервал відновлення інформації про тестований у даний момент пароль. Збільшення інтервалу підвищує швидкодію, але знижує інформативність. Також можна установити режим ведення протоколу роботи і можливість мінімізації програми в tray (маленька іконка поруч з годинником).

Зміст завдання

1. Вивчити всі можливі варіанти захисту документів MS Word .Здійснити захист будь-якого документу MS Word різними способами.

2. Вивчити всі можливі варіанти захисту документів MS Excel. Здійснити захист будь-якого документу MS Excel різними способами.

3. Вивчити всі можливі варіанти захисту документів MS Access . Створити будь-яку базу даних та здійснити її захист.

4. За допомогою програми Advanced Office XP Password Recovery здійснити злом пароля *.doc документа.

5. Зархівувати попередньо створені незахищені файли за допомогою архіваторів RAR та ZIP. Задати їм парольний захист.

5.1. Використовуючи програму для розкриття паролів зробити атаку перебором (bruteforce attack) на зашифрований файл *.rar *.zip . Область перебору – усі символи, що друкуються, довжина пароля від 1 до 4 символів. Перевірити правильність визначеного пароля, розпакувавши файл і ознайомивши з його вмістом.

Скоротити область перебору до фактично використовуваного (наприклад якщо пароль 6D1A – то вибрати прописні англійські букви і цифри). Провести повторне розкриття. Порівняти витрачений час.

5.2. Проведення атаки по словнику (dictionary attack)

Стиснути який-небудь невеликий файл, вибравши як пароль англійське слово довжиною до 5 символів (наприклад love, god, table, admin і т.д.). Провести атаку по словнику. Для цього вибрати вид атаки й у закладці Словник вибрати файл English.dic. Він містить набір англійських слів і набори символів, що найбільше часто використовуються як паролі.

Спробувати визначити пароль методом прямого перебору. Порівняти витрачений час.

Зміст звіту

1. Опис результатів виконання.

2. Висновки, погоджені з метою роботи.

ЛАБОРАТОРНА РОБОТА № 2

Тема: Розмежування прав користувачів

Мета: Розробка програми розмежування повноважень користувачів на основі паролльної аутентифікації.

Зміст завдання

1. Програма повинна забезпечувати роботу в двох режимах: адміністратора (користувача з фіксованим ім'ям ADMIN) та звичайного користувача.

2. У режимі адміністратора (рис. 1) програма повинна підтримувати такі функції (при правильному введенні пароля):

- зміна пароля адміністратора (при правильному введенні старого пароля);



Рисунок 1 – Можливий вигляд головного вікна програми після входу адміністратора

- перегляд списку імен зареєстрованих користувачів і встановлених для них параметрів (блокування облікового запису, включення обмежень на паролі які вибираються) - всього списку повністю в одному вікні або по одному елементу списку з можливістю переміщення до його початку або кінця (рис. 2);

- додавання унікального імені нового користувача до списку з порожнім паролем (рядком нульової довжини) – рис. 3;

- блокування можливості роботи користувача з заданим ім'ям (рис. 2);

- включення або відключення обмежень на паролі які вибираються користувачем (у відповідності з індивідуальним завданням, визначеним номером варіанта) – рис. 2;

- завершення роботи з програмою.

3. У режимі звичайного користувача (рис. 4) програма повинна підтримувати тільки функції зміни пароля користувача (при правильному введенні старого пароля) та завершення роботи, а всі інші функції повинні бути заблоковані.

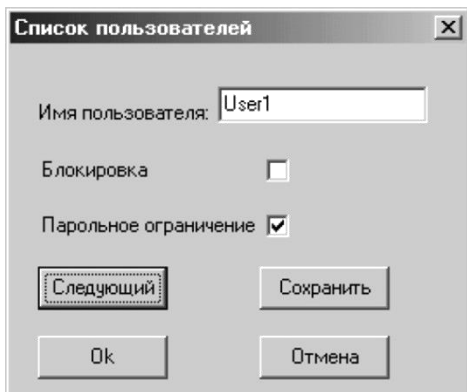


Рисунок 2 – Возможный вид виджетов (редагування) облікових записів

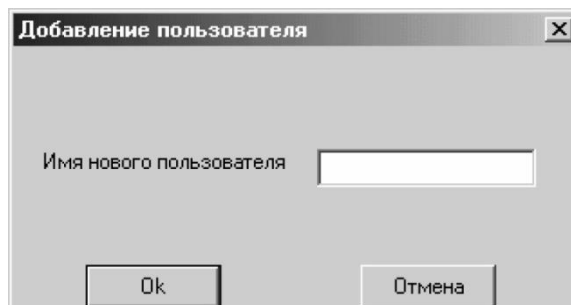


Рисунок 3 – Возможный вид виджетов додавання нового користувача



Рисунок 4 – Возможный вид главного виджетов програми після входу звичайного користувача

4. Після запуску (рис. 5) програма повинна запитувати у користувача в спеціальному вікні входу (рис. 6) введення його імені та пароля. При введенні пароля його символи завжди повинні на екрані замінюватися символом – '*'.
 5. При відсутності введення в вікні входу імені користувача в списку зареєстрованих адміністратором користувачів програма повинна видавати відповідне повідомлення і надавати користувачеві можливість повторного введення імені чи завершення роботи з програмою.

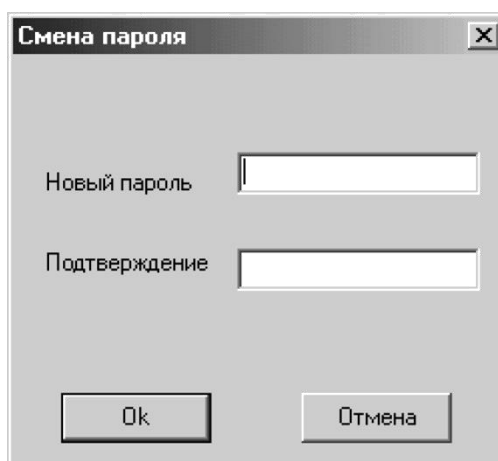


Рисунок 5 – Возможный вид главного виджетов програми після її запуску

5. При відсутності введення в вікні входу імені користувача в списку зареєстрованих адміністратором користувачів програма повинна видавати відповідне повідомлення і надавати користувачеві можливість повторного введення імені чи завершення роботи з програмою.

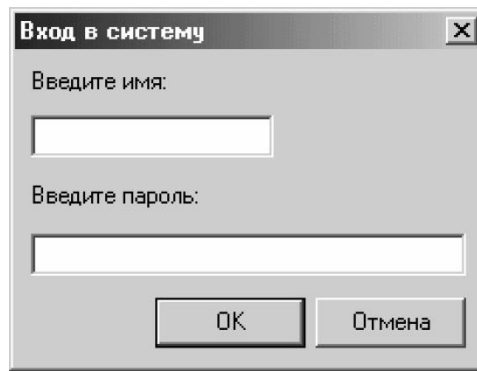


Рисунок 6 – Можливий вигляд вікна входу в програму

6. При неправильному введенні пароля програма повинна видавати відповідне повідомлення і надавати користувачеві можливість повторного введення. При триразовому введенні неправильного пароля робота програми повинна завершуватися.

7. При первинному введенні пароля (обов'язковому при першому вході адміністратора або користувача з зареєстрованим раніше ім'ям адміністратором) і при подальшій заміні пароля програма повинна просити користувача підтвердити введений пароль шляхом його повторного введення. Якщо вибраний користувачем пароль не відповідає необхідним обмеженням (при встановленні відповідного параметра облікового запису користувача), то програма повинна видавати відповідне повідомлення і надавати користувачеві можливість введення іншого пароля, завершення роботи з програмою (при першому вході даного користувача) або відмови від зміни паролю.

8. Інформація про зареєстрованих користувачів, їх паролі, відсутність блокування їх роботи з програмою, а також включення або відключення обмежень на вибрані паролі повинна зберігатися в спеціальному файлі. При першому запуску програми цей файл повинен створюватися автоматично та міститиме інформацію тільки про адміністратора, що має порожній пароль.

9. Інтерфейс з програмою повинен бути організований на основі меню, обов'язковою частиною якого має бути підменю «Довідка» з командою «Про програму». При виборі цієї команди повинна видаватися інформація про автора програми і виданому індивідуальному завданні (рис. 7).

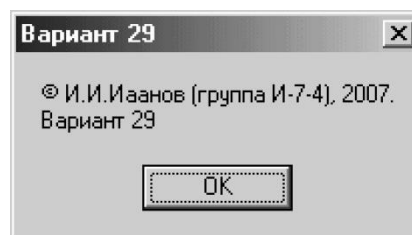


Рисунок 7 – Можливий вигляд вікна з інформацією про автора програми

10. Обмеження на паролі які вибираються користувачем програми залежать від вказаного викладачем номера індивідуального варіанта.

Варіанти індивідуальних завдань (обмеження на паролі)

1. Довжина не менше мінімальної довжини, установленної адміністратором, що зберігається в обліковому записі користувача.
2. Наявність малих і великих літер.
3. Наявність букв і цифр.
4. Наявність букв і розділових знаків.
5. Наявність цифр і розділових знаків.
6. Наявність букв і знаків арифметичних операцій.
7. Наявність цифр і знаків арифметичних операцій.
8. Наявність латинських букв і символів кирилиці.
9. Наявність букв, цифр і розділових знаків.
10. Наявність латинських букв, символів кирилиці та цифр.
11. Наявність латинських букв, символів кирилиці та розділових знаків.
12. Наявність малих і великих літер, а також цифр.
13. Наявність малих і великих літер, а також розділових знаків.
14. Наявність малих і великих літер, а також знаків арифметичних операцій.
15. Наявність латинських букв, символів кирилиці та знаків арифметичних операцій.
16. Наявність букв, цифр і знаків арифметичних операцій.
17. Наявність букв, розділових знаків та знаків арифметичних операцій.
18. Наявність цифр, розділових знаків і знаків арифметичних операцій.
19. Відсутність повторюваних символів.
20. Чергування літер, цифр і знову букв.
21. Чергування літер, розділових знаків і знову букв.
22. Чергування цифр, букв і знову цифр.
23. Відсутність підряд розташованих однакових символів.
24. Чергування цифр, розділових знаків і знову цифр.
25. Чергування цифр, знаків арифметичних операцій і знову цифр.
26. Розбіжність з ім'ям користувача.
27. Розбіжність з ім'ям користувача, записаним у зворотному порядку.
28. Наявність рядкових і прописних латинських літер, цифр і символів кирилиці.
29. Наявність малих і великих літер, цифр і знаків арифметичних операцій.
30. Наявність латинських букв, символів кирилиці, цифр і знаків арифметичних операцій.

Зміст звіту

1. Опис результатів виконання та лістинг програми.
2. Висновки, погоджені з метою роботи.

ЛАБОРАТОРНА РОБОТА №3

Тема: Класичні методи симетричного шифрування.

Мета: Ознайомлення з класичними методами симетричного шифрування, Отримати практичні навички створення програмного забезпечення за найпростішими криптографічними перетвореннями.

Теоретичні відомості

1. Шифри перестановок

Шифрування перестановкою полягає в тому, що символи тексту переставляються за визначеним правилом у межах деякого блоку цього тексту. При достатній довжині блоку, у межах якого здійснюється перестановка, і складному неповторюваному порядку перестановки можна досягти прийнятної для простих практичних додатків стійкості шифру. Шифри перестановки є найпростішими і, імовірно, самими давніми шифрами.

1.1. Шифр частоколу

Наприклад, зашифруємо слово «за шифрування» із висотою частоколу 2. Для цього запишемо так:

а и р в н я
з ш ф у а н

Тепер зчитуємо спочатку верхній рядок, а потім нижній, отже зашифроване слово: «аирвнязшфуан».

1.2. Табличні або матричні

З початку епохи Відродження (кінець XIV сторіччя) початку відроджуватися і криптографія. Поряд із традиційними застосуваннями криптографії в політику, дипломатії і військовій справі з'являються й інші задачі захист інтелектуальної власності від переслідувань інквізиції або запозичень зловмисників. У розроблених шифрах перестановки того часу застосовуються таблиці, що шифрують, що у сутності задають правила перестановки букв у повідомленні. Як ключ у табличних методах використовують: розмір таблиці; слово або фраза, що задають перестановку; особливості структури таблиці.

Проста перестановка

Одним із самих примітивних табличних шифрів перестановки є проста перестановка, для якої ключем служить розмір таблиці. Цей метод шифрування подібний із шифром скітала.

Наприклад, повідомлення «завтра пишемо контрольну роботу» записується в таблицю по черзі по стовпцях. Результат заповнення таблиці з 5 рядків і 7 стовпців показаний у табл.1.

Після заповнення таблиці текстом повідомлення по стовпцях для формування шифротексту зчитують вміст таблиці по рядках. Якщо шифротекст записувати групами по чотири букви, виходить таке шифроване повідомлення:

«зрпик рнба аеоо товп мнлр ттио тьоу»

Заповнення таблиці з 5 рядків і 7 стовпців

з	р	ш	к	р	н	б
а	а	е	о	о	у	о
в	п	м	н	л	р	т
т	и	о	т	ь	о	у

Природно, відправник і одержувач повідомлення повинні заздалегідь домовитися про загальний ключ у виді розміру таблиці. Варто помітити, що об'єднання букв шифротекста в 4-буквенні групи не входить у ключ шифру і здійснюється для зручності запису безмістового тексту. При розшифруванні дії виконують у зворотному порядку.

Перестановка за ключем

Дещо більшою стійкістю до розкриття володіє метод шифрування, називаний одиночною перестановкою за ключем. Цей метод відрізняється від попередніх тем, що стовпці таблиці переставляються по ключовому слову, фразі або наборові чисел довжиною в рядок таблиці. Цей шифр використовували під час Першої світової війни.

Застосуємо як ключ, наприклад, слово ПЕЛІКАН, а текст повідомлення візьмемо з попереднього прикладу. Дві таблиці, заповнені текстом повідомлення і ключовим словом, при цьому перша таблиця відповідає заповненню до перестановки, а друга таблиці – заповненню після перестановки.

Таблиця 2

Таблиця, заповнена ключовим словом и текстом повідомлення

п	е	л	і	к	а	н
7	2	5	3	4	1	6
з	р	ш	к	р	н	б
а	а	е	о	о	у	о
в	п	м	н	л	р	т
т	и	о	т	ь	о	у

Таблиця 3

Перестановка стовпців

1	2	3	4	5	6	7
н	р	к	р	ш	б	з
у	а	о	о	е	о	а
р	п	н	л	м	т	в
о	и	т	ь	о	у	т

У верхньому рядку лівої таблиці записаний ключ, а номери під буквами ключа визначені в залежності з нормальним порядком відповідних букв ключа в алфавіте. Якщо б в ключі зустрілись однакові букви, вони були б пронумеровані зліва на право. У правій таблиці стовпці переставлені в залежності з впорядкованими номерами букв ключа. При зчитуванні вмісту правої таблиці по рядках и запису шифротекста групами по п'ять букв отримаємо зашифроване повідомлення: «*нркр шбзу аооо оарп нлмт воит ьоут*»

Подвійна перестановка

Для забезпечення додаткової таємності можна повторно зашифрувати повідомлення, що вже пройшло шифрування. Такий метод шифрування називається подвійною перестановкою. У випадку подвійної перестановки стовпців і рядків таблиці перестановки визначаються окремо для стовпців і окремо для рядків. Спочатку в таблицю записується текст повідомлення, а потім по черзі переставляються стовпці, а потім рядки. При розшифруванні порядок перестановок повинний бути зворотним.

Приклад виконання шифрування методом подвійної перестановки показаний таблиці 4. Ключем до шифру подвійної перестановки служить послідовність номерів стовпців і номерів рядків вихідної таблиці (у нашому прикладі послідовності 4132 і 3142 відповідно). Повідомлення – «*прилітаю восьмого*».

Таблиця 4

Приклад виконання шифрування методом подвійної перестановки

	4	1	3	2			1	2	3	4			1	2	3	4
3	п	р	и	л		3	р	л	и	п		1	т	ю	а	і
1	і	т	а	ю		1	т	ю	а	і		2	о	о	г	м
4	в	о	с	ь		4	о	ь	с	в		3	р	л	и	п
2	м	о	г	о		2	о	о	г	м		4	о	ь	с	в

Якщо зчитувати шифртекст із правої таблиці по рядках блоками по чотири букви, то вийде наступне: „*тлюаі оогм рлип оьсв*”.

Число варіантів подвійної перестановки швидко зростає при збільшенні розміру таблиці: для таблиці 3x3 – 36 (3!x3!) варіантів; для таблиці 4x4 – 576 варіантів і так далі. Однак подвійна перестановка не відрізняється високою стійкістю і порівняно просто «зламується» при будь-якому розмірі таблиці шифрування.

Магічні квадрати

Магічними квадратами називають квадратні таблиці записаними в їхні клітки послідовними натуральними числами, починаючи від 1, що дають у сумі по кожному стовпці, кожному рядкові і кожній діагоналі те саме число.

Таблиця 5

Заповнення таблиці

16	3	2	13		о	и	р	м
5	10	11	8	→	і	о	с	ю
9	6	7	12		в	т	а	ь
4	15	14	1		л	г	о	п

«*прилітаю восьмого*» – «*оирм іосо втаь лгон*».

2. Шифри підстановок

Шифрування заміною (підстановкою) полягає в тому, що символи тексту замінюються символами того ж або іншого алфавіту відповідно до заздалегідь обумовленої схеми заміни. Вони поділяються на **моноалфавітні**, **багатоалфавітні (поліалфавітні)**, **гомофонічні**.

2.1. Моноалфавітні

Шифр Цезаря. При шифруванні вихідного тексту кожна буква замінюється на іншу букву того ж алфавіту шляхом зміщення на K букв. При досяганні кінця алфавіту здійснюється циклічний перехід до початку. Головне, щоб адресант повідомлення знав величину і напрямок зсуву. Існує лише $N!$ моноалфавітних підстановок.

Переваги: простота шифрування та дешифрування.

Недоліки: не маскуються частоти появи різних букв вихідного тексту; зберігається алфавітний порядок букв; кількість ключів невелика.

Криптоаналітична атака починається із підрахунку частот появи символів і порівняння з розподілом частот в деякому алфавіті.

Модифікація шифру Цезаря із ключовим словом

Ключове слово використовується для зміщення та зміни порядку символів (бажано, щоб букви були різними, якщо вони повторюються, то їх можна випускати). Ключове слово підписується під буквами алфавіту, починаючи з тієї букви числовий код якої співпадає із вибраним K , $0 \leq K \leq 25$ – для англійського алфавіту.

Наприклад ключеве слово – *дипломат* і $K=5$. Тоді зсув буде наступним:

a b c d e f g h i j k l m n o p q r s t u v w x y z
v w x y z d i p l o m a t b c e f g h i j k n q r s u

Тепер «*send more money*» – «*hzby tcgd tcbzs*».

Квадрат Полібія

Уперше даний квадрат був розроблений для грецького алфавіту. Для українського алфавіту квадрат може мати вигляд вказаний у таблиці 6.

Таблиця 6

Заповнення таблиці символами у довільному порядку

і	ж	ю	о	в	а
з	н	и	ш	ь	я
б	д	л	ф	є	ч
м	е	щ	й	п	г
к	у	т	х	ї	с
ц	р	.	,	-	;

Для зашифрування повідомлення квадрат використовується так: кожна літера відкритого тексту замінюється на літеру, що знаходиться під нею, якщо літера стоїть в останньому рядочку, то вона замінюється такою, що стоїть у першому рядочку саме над нею.

Отже «*зустріч перенесемо на завтра*» – «*бр;.жзгїужсуду;удидябья.жя*».

Модифікація даної таблиці – таблиця Трісімуса, де букви розташовуються в залежності від ключа. Ключ першим вписується в таблицю, а потім всі наступні букви, що не входять до ключа.

Шифр Плейфера (або play-fair)

Цей шифр теж використовувався в першій світовій війні. Застосовується на підстановці не окремих символів, а пари символів (2-грами). Алфавіт розташовується в таблиці в довільному порядку. Вихідний текст розбивається на 2-грами (кількість букв парна), біграми не повинні містити однакові букви.

Якщо дві букви не належать одному рядку чи стовпцю, то знаходяться букви в кутках прямокутника, що і визначає пару букв. Якщо дві букви належать одному рядку чи стовпцю, то пишуться букви, що знаходяться (**справа**) **або** (**під**) ними. Шифрування повідомлення із попереднього прикладу на основі таблиці 6 буде наступним:

«зу ст рі чп ер ен ес ем он аз ав тр а.» – «кн кх цж ге уж уд уг ще шж яі ія .у ;ю».

Афінна підстановка

Розглянемо на прикладі: $m=26$ (англійський алфавіт), $a=3$, $b=5$ – вибрані числа, так щоб виконувались умови: $0 \leq a, b \leq m$, $HCD(a, m) = 1$. Заміна здійснюється за формулою: $E_{a,b}(t) = at + b \pmod{m}$. Перевіримо $HCD(3, 26) = 1$. Тому зсув відбувається за формулою $3t + 5$, де $t \in (0, 25)$.

Таблиця 7

Заміна символів								
t	0	1	2	3	4	5	25
3t+5	5	8	11	14	17	20	2
Початкова	A	B	C	D	E	F	Z
Після заміни	F	I	L	C

Текст «*hore*» – «*avuk*».

Переваги: зручне керування ключами, ключі представляються у вигляді пари чисел a, b . Недоліки: легко знайти формулу.

2.2. Поліалфавітні

Шифри складної заміни називають багатолфавітними шифрами, так як для шифрування кожного символу вихідного повідомлення застосовується свій шифр простої заміни. Багато алфавітна підстановка послідовно і циклічно змінює використовувані алфавіти. Ефект багато алфавітної підстановки полягає в тому, щоб забезпечити маскування справжньої статистики мови, так як конкретний символ із вихідного алфавіту може бути перетворений в декілька різних символів інших алфавітів.

Багато алфавітні шифри замни запропонував і ввів в практику криптографії Леон Батис Альберті, який був відомим архітектором і теоретиком мистецтва криптології. Його книга «Трактат про шифр» написана в 1566 році, представила собою першу в Європі наукову працю по криптології. Крім шифру багатоалфавітної підстановки він описав пристрій із обертанням коліс для його реалізації. Криптологи всього світу вважають його основоположником криптології.

Система шифрування Віжінера

Система шифрування Віжінера вперше була опублікована в 1586 році. Свою назву вона отримала за іменем французького дипломата Блеза Віжінера, який розвивав і вдосконалював криптографію. У процесі шифрування і дешифрування використовувалася таблиця Віжінера (квадрат), що утворена наступним чином: у перший рядок виписується весь алфавіт, в наступних рядках вводиться зсув на 1 літеру вліво, так отримується квадратна таблиця;

щоб зашифрувати повідомлення, вибирають лозунг, який підписують під текстом; потім у стовпчику шукаємо літеру повідомлення, а у рядку – ключа, на їх перетині знаходиться наша підстановка. Таблиця Віжінера представлена у додатку А.

Приклад: Зашифруємо повідомлення *«переходьте до виконання плану номер два»* на ключі *резидент*.

п е р е х о д ь т е д о в и к о н а н н я п л а н у н о м е р д в а
р е з и д е н т р е з и д е н т р е з и д е н т р е з и д е н т р е
е л і ш л щ у к п з і й ч е л ь с г е x ц ф ю т г ш х ч р і г ц т е

Зашифрований текст: *«елішлщукпзійчельсгехцфютгшихчрігцте»*.

Система шифрування методом Вернама

Система Вернама є частинним випадком шифру Віжінера при $m=2$ (алфавіт 0 і 1). Конкретна версія цього шифру запропонована в 1926 році Гілбертом Вернамом, співробітником фірми AT&T США. Кожен символ вихідного відкритого тексту із англійського алфавіту $\{A,B,C,...,Z\}$, що розширений 5 допоміжними знаками (пробіл і т. д), спочатку кодувався в 5-бітовий блок телеграфного коду Бодо. Далі випадкова послідовність ключів $k_0, k_1, ...$ наперед записувалась на паперовій стрічці. \oplus – операція додавання за модулем 2. Схема передачі з використанням шифру Вернама показана на рис.1.

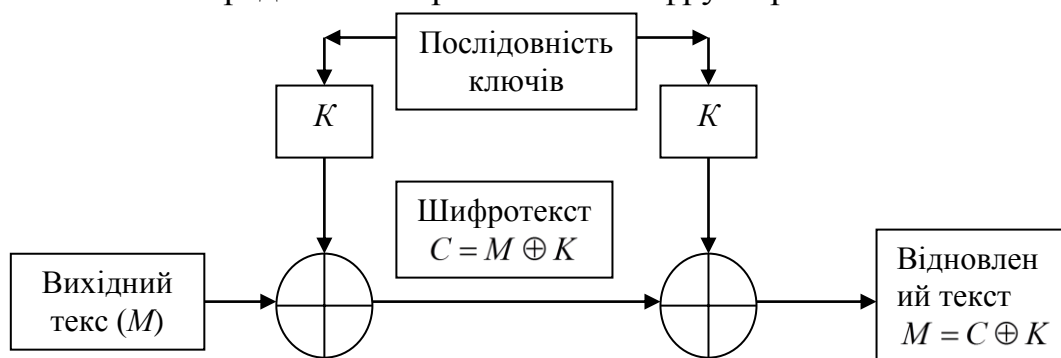


Рисунок 1 – Система шифрування за методом Вернама

Шифрування:

Текст	11001110101	– M
Ключ	10101010101	– K
Сума	21102120202	
Сума за модулем 2	01100100000	– C

Дешифрування:

Шифр	01100100000	– C
Ключ	10101010101	– K
Сума	11201110101	
Сума за модулем 2	11001110101	– M

Гомофонічні

Одному елементу ставиться у відповідність декілька символів шфротексту. Цей метод застосовується для викривлення статистичних властивостей тексту.

Заміна символів гомо фонічної підстановки

Алфавіт тексту	а	б	...	і	ж	з	...	м	н	...
Алфавіт шифру	17	23		97	47	76		32	55	
	31	44		51	67	19		28	84	
	48	63		15	33	59		61	34	

Приклад: «заміна» – «761732975531».

Таким чином, при гомофонічній підстановці кожна буква відкритого тексту замінюється по черзі цифрами відповідного стовпця.

3. Гамування

Шифрування гамування полягає в тому, що символи тексту складаються із символами деякої випадкової послідовності, іменованою гамою шифру. Стійкість шифрування визначається в основному довжиною (періодом) неповторюваної частини гами шифру. Оскільки за допомогою комп'ютера можна згенерувати практично нескінченну гаму шифру, те даний спосіб є одним з основних для шифрування інформації в автоматизованих системах. Перед шифруванням відкритий текст поділяється на блоки однакової довжини $T_0^{(i)}$, зазвичай по 64 біти. Гама шифру виробляється у вигляді послідовності блоків аналогічної довжини $\Gamma_u^{(i)}$.

$$T_u^{(i)} = \Gamma_u^{(i)} \oplus T_0^{(i)}, i=1, ..\kappa - \text{процес шифрування};$$

$$T_0^{(i)} = \Gamma_u^{(i)} \oplus T_u^{(i)}, i=1, ..\kappa - \text{процес дешифрування},$$

де $T_u^{(i)}$ – i -й блок шифру; $T_0^{(i)}$ – i -й блок відкритого тексту; $\Gamma_u^{(i)}$ – i -й блок гами; κ – кількість блоків.

Отриманий цим шифром текст досить складний до розкриття, оскільки ключ змінний. Гама шифру повинна змінюватись випадковим чином.

4. Аналітичні методи

Шифрування аналітичним перетворенням полягає в тому, що текст перетворюється за деяким аналітичним правилом. Наприклад, можна використовувати правило множення вектора на матрицю, причому множена матриця є ключем шифрування (тому її розмір і зміст повинні зберігатися в секреті), а символами множеного вектора послідовно служать символи зашифрованого тексту. Іншим прикладом може служити використання так званих односторонніх функцій для побудови криптосистем з відкритим ключем.

Будемо вважати, що функція $y = f(x)$ є односторонньою, якщо вона за порівняно невелику кількість операцій перетворює елемент відкритого тексту X в елемент шифротексту Y для всіх значень із області визначення, а обернена операція (обчислення $x = F^{-1}(y)$ при зданому шифротексті) є дуже трудомісткою або, навіть неможливою. В якості односторонньої функції можна навести множення матриць, факторизація великого числа, експоненціальні перетворення. Прикладів аналітичних перетворень є дуже багато, тому розглянемо лише декілька для розуміння матеріалу.

Зашифруємо текст «приказ» (16 17 09 11 01 08 – порядок букв в рос алфавіті) на

ключі $C: \begin{pmatrix} 1 & 3 & 2 \\ 2 & 1 & 5 \\ 3 & 2 & 1 \end{pmatrix}$.

Здійснюємо перетворення:

$$\begin{pmatrix} 1 & 3 & 2 \\ 2 & 1 & 5 \\ 3 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 16 \\ 17 \\ 09 \end{pmatrix} = (85 \ 94 \ 91) \text{ та } \begin{pmatrix} 1 & 3 & 2 \\ 2 & 1 & 5 \\ 3 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 11 \\ 01 \\ 08 \end{pmatrix} = (30 \ 63 \ 43).$$

Отже отримали шифр – «859491306343».

Шифр Хілла

Перетворення: $y_i = T \cdot x_i \pmod{m}$, де T – лінійне перетворення, що описується матрицею. T повинне мати обернене перетворення, а також необхідно щоб визначник T не ділився на будь-яке просте p , яке ділить m (розмір алфавіту).

Розглянемо цей метод на прикладі. Нехай $m=26=13*2$, $T = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix}$ – матриця перетворення, $\text{Det}(T)=9$, що не ділиться ні на 2, ні на 13, а також існує обернене перетворення (бо визначник не рівний нулю).

Спочатку текст «*frautoremoney*» розбивають на біграми (можна і на триграми і т.д.), потім кожну букву в біграмі замінюють її порядковим номером в алфавіті, починаючи з нуля.

р а у т о р е м о н е у
15 0 24 12 14 17 4 12 14 13 4 24

$$y_1 = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \cdot \begin{pmatrix} 15 \\ 0 \end{pmatrix} = \begin{pmatrix} 45 \\ 30 \end{pmatrix} \pmod{26} = \begin{pmatrix} 19 \\ 4 \end{pmatrix};$$

$$y_2 = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \cdot \begin{pmatrix} 24 \\ 12 \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix};$$

$$y_3 = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \cdot \begin{pmatrix} 14 \\ 17 \end{pmatrix} = \begin{pmatrix} 15 \\ 9 \end{pmatrix};$$

$$y_4 = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 12 \end{pmatrix} = \begin{pmatrix} 22 \\ 16 \end{pmatrix};$$

$$y_5 = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \cdot \begin{pmatrix} 14 \\ 13 \end{pmatrix} = \begin{pmatrix} 3 \\ 15 \end{pmatrix};$$

$$y_6 = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 24 \end{pmatrix} = \begin{pmatrix} 6 \\ 24 \end{pmatrix};$$

Отже, шифротекст . згідно тієї ж таблиці – «*teeeyiwqdygy*».

Зміст завдання

1. Ознайомитись з теоретичним матеріалом лекції «Класичні методи симетричного шифрування.».

2. Написати програму (на будь-якій мові програмування) для шифрування та дешифрування файлу одним із вказаних методів, які наведені в таблиці. Використати різні алфавіти, розміри таблиць, різні зсуви та інше.

Варіанти індивідуальних завдань

№ варіанту	Метод
1	Табличний (різний розмір таблиць)
2	Табличний за ключем
3	Табличний (подвійна перестановка)
4	«Магічні квадрати»
5	Шифр Цезаря із різним зсувом
6	Шифр Цезаря за ключовим словом
7	Шифр Полібія
8	Шифр Плейфера
9	Афінна підстановка
10	Шифр Віжінера
11	Шифр Вернама
12	Гомофонічні підстановки
13	Гамування
14	Аналітичний (шифр Хілла)
15	Аналітичний (експоненціальний)
16	Аналітичний (поліном)

Зміст звіту

1. Опис результатів виконання та лістинг програми.
2. Висновки, погоджені з метою роботи.

ЛАБОРАТОРНА РОБОТА № 4

Тема: Система блочного шифрування S-DES

Мета: створити просту криптографічну систему на основі спрощеного блочного алгоритму Simple DES (S-DES) та дослідити її роботу.

Теоретичні відомості

Спрощений DES – це алгоритм шифрування, який має, скоріше, навчальне, ніж практичне значення. За своїми властивостями він подібний до DES, але має значно менше параметрів.

Цей алгоритм приймає на вході 8-бітний блок відкритого тексту та 10-бітний ключ, а на виході генерує 8-бітний блок шифрованого тексту. При розшифруванні на вхід алгоритму подається 8-бітний блок шифротексту і 10-бітний ключ, а на виході генерує 8-бітний блок відкритого тексту. Алгоритм шифрування передбачає послідовне виконання п'яти операцій: початкової перестановки IP ; раундової функції, що складається з перестановок і підстановок; перестановки SW , коли дві половинки блока по 4 біти переставляються місцями; ще одного застосування раундової функції; і, нарешті, перестановки IP^{-1} оберненої до початкової. Послідовне використання кількох перестановок і підстановок значно ускладнюють криптоаналіз.

Раундова функція приймає на вході не лише блок тексту, а й 8-бітний цикловий підключ, який утворюється з 10-бітного ключа.

Блок-схему алгоритму подано на рис.1. З цього рисунка видно, що, оскільки це симетричний криптоалгоритм, він використовує для шифрування та розшифрування той самий ключ. Тому ключ має бути як на передавальній, так і на приймальній стороні. З цього ключа на певних етапах шифрування та розшифрування генеруються два 8-бітних раундових підключів.

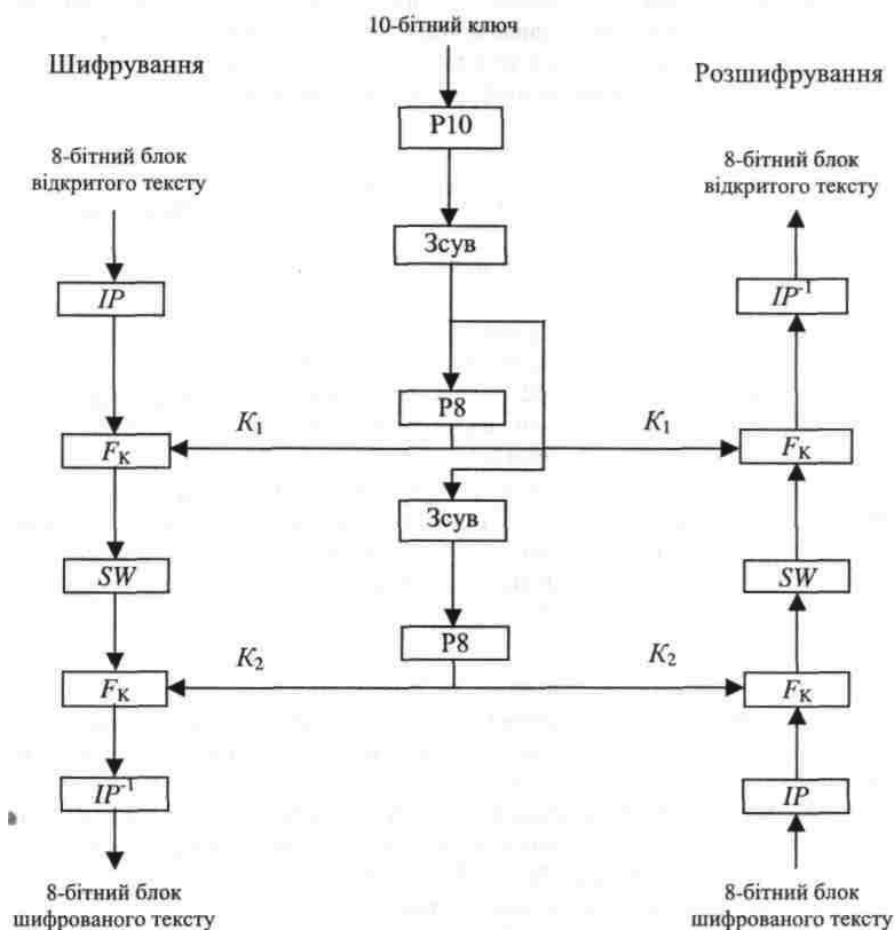


Рисунок 1 - Схема спрощеного алгоритму S-DES

Процедура генерування раундових підключів

1. Спочатку біти ключа переставляються так. Якщо 10-бітний ключ подати у вигляді k_1, k_2, \dots, k_{10} , то перестановка P10 задається у вигляді таблиці:

P10									
3	5	2	7	4	10	1	9	8	6

Ця таблиця символізує позицію біта вхідних даних у вихідній послідовності: першим стає 3-й біт, другим – 5-й, третім – 2-й і т.д. Наприклад, ключ (1010000010) відповідно до цієї перестановки перетворюється в послідовність (1000001100).

2. Ключ розділяється на дві 5-бітні половини. Окремо перша половина й окремо друга піддаються циклічному зсуву ліворуч на одну позицію. У нашому прикладі в результаті буде отримана послідовність (00001 11000).

3. Отримана послідовність піддається перестановці P8, у результаті якої з 10-бітного ключа обирається 8-бітна послідовність за таким правилом:

P8							
6	3	7	4	8	5	10	9

У результаті цієї операції ми отримаємо перший раундовий підключ (K_1). У нашому прикладі він буде мати вигляд (101001100).

4. Для генерування другого раундового підключу K_2 необхідно повернутися на крок назад, до двох 5-бітних рядків до застосування P8 та виконати для кожного з цих рядків циклічний зсув ліворуч на дві позиції. У нашому прикладі значення підключів (00001 11000) перетворюються у (00100 00011).

5. Нарешті, застосувавши до цієї послідовності перестановку P8, отримаємо другий раундовий підключ K_2 . Для нашого прикладу результатом буде (01000011).

Шифрування S-DES

1. Початкова й кінцева перестановки (IP та IP^{-1}). На вхід алгоритму подається 8-бітний блок відкритого тексту, до якого застосовується початкова перестановка IP . Можна пересвідчитися, що ці дві таблиці дійсно обернені одна до одної, тобто $IP^{-1}(IP(M))=M$.

IP							
2	6	3	1	4	8	5	7

IP^{-1}							
4	1	3	5	7	2	8	6

2. Раундова функція F . Розіб'ємо вхідний блок тексту після IP -перестановки на два 4-бітні півблоки. Лівий 4-бітний блок позначимо L , а правий – R . Тоді циклову функцію можна записати у вигляді формули

$$F_K(L,R) = (L \oplus F(R,K_i), R), \text{ тут } K_i \text{ означає цикловий підключ.}$$

Тепер опишемо саму циклову функцію. На вході вона отримує 4-бітне значення (n_1, n_2, n_3, n_4) , тобто праву половину вхідного блока. Перша операція – операція розширення та перестановки. Її можна також зобразити таблицею[^]

Розширення з перестановкою							
4	1	2	3	2	3	4	1

Зручніше цю операцію зобразити у вигляді матриці:

$$\begin{pmatrix} n_4 & n_1 & n_2 & n_3 \\ n_2 & n_3 & n_4 & n_1 \end{pmatrix}.$$

До цього значення додається 8-бітний підключ за допомогою операції XOR.

Це можна зобразити так:

$$\begin{pmatrix} n_4 + k_1 & n_1 + k_2 & n_2 + k_3 & n_3 + k_4 \\ n_2 + k_5 & n_3 + k_6 & n_4 + k_7 & n_1 + k_8 \end{pmatrix}.$$

Переименуємо отримані елементи:

$$\begin{pmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \end{pmatrix}.$$

Перші чотири біти (тобто перший рядок цієї матриці) далі подаються на вхід модуля заміни (S -матриці) S_0 , на виході якого отримується 2-бітна послідовність. Другий рядок матриці подається на вхід другого модуля заміни S_1 , на виході якого також отримується 2-бітна послідовність. Модулі S_0 і S_1 , задаються так:

$$S_0 = \begin{pmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 1 \end{pmatrix} \quad S_1 = \begin{pmatrix} 1 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{pmatrix}$$

Рядки та стовпчики нумеруються, починаючи з нуля.

Ці модулі заміни працюють так. Перший і четвертий біти вхідної послідовності вважаються двійковим представленням номера рядка, а другий і третій – номерами стовпця. Елемент, що знаходиться на перетині цих рядка і стовпця, задає двобітне вихідне значення. Наприклад, якщо $(p_{00}, p_{03}) = (00)$ та $(p_{01}, p_{02}) = (10)$, то вихідні два біти задаються значенням, яке знаходиться на

перетині 0-го рядка та 3-го стовпців, тобто це буде число 3, а у двійковому представленні – 11.

Аналогічну операцію виконують і з другим рядком.

Після застосування матриць заміни результат піддають перестановці P_4 за таким законом:

P_4			
2	4	3	1

Результат перестановки P_4 і буде результатом функції F_K . Отримана послідовність бітів додається за модулем 2 з лівою половиною L вхідного блока і буде новою лівою половиною. Права половина передається на вихід циклу без змін.

3. Перестановка підблоків. Як бачимо, за один раунд раундовою функцією обробляється лише ліва половина відкритого тексту, права половина залишається без змін. Для того, щоби зашифрувати й праву половину, використовується другий цикл, однак на його вхід треба подати переставлені підблоки: L і R поміняти місцями. Для цього й служить функція SW - перемикач блоків.

Після переставлення підблоків один раунд алгоритму закінчено.

До переставлених підблоків знову застосовується циклова функція, як це описано вище. При другому виклику раундової функції розширення з перестановкою модулів S_0 і S_1 , P_4 залишаються тими ж, тільки використовується підключ K_2 .

По закінченні другого раунду виконується IP^{-1} -перестановка, й роботу алгоритму закінчено, тобто на виході маємо зашифрований текст.

Розшифрування зашифрованого тексту

Як видно з рис. 1, розшифрування зашифрованого тексту виконується аналогічно шифруванню, за винятком того, що ключі подаються у зворотному порядку.

Зміст завдання

Завдання 1. Шифрування/розшифрування S-DES.

Створити просту криптографічну систему на основі спрощеного блочного алгоритму S-DES. Перевірити її роботу.

1. Використовуючи довільну мову програмування, створіть програму, яка б шифрувала та розшифровувала текстові повідомлення за допомогою криптосистеми S-DES.

2. Перевірте правильність функціонування криптосистеми, зашифрувавши та розшифрувавши текст. Порівняйте отриманий результат із відкритим текстом.

3. Здійсніть шифрування та розшифрування двох інших текстів на різних криптографічних ключах і зробіть висновок про правильність функціонування створеної програми.

4. Зробіть висновок із лабораторної роботи.

5. Підготуйте звіт із лабораторної роботи. Звіт повинен містити: а) протокол Ваших дій; б) код програми; в) відкриті та зашифровані тексти, а

також криптографічні ключі, на яких виконувалося шифрування; г) результати порівняння розшифрованих текстів із відкритими; д) висновки з лабораторної роботи; є) відповіді на контрольні запитання.

Завдання 2. Дослідження розсіювальних властивостей S-DES.

Дослідити властивості розсіювання та перемішування бітів у спрощеній криптосистемі S-DES. Дослідити залежність вихідних параметрів від структури блоків заміни алгоритму.

1. Використовуючи створену програму, вивчіть її властивості розсіювання в залежності від вхідних бітів ключа. Для цього оберіть ключ «0000000000» та зашифруйте на ньому вибраний текст. Після цього змініть молодший біт на 1 та виконайте шифрування того ж тексту. Скільки бітів зашифрованого тексту змінилося? Виконайте таку ж операцію з кожним бітом ключа. Поясніть отримані результати. Зміна якого біта ключа приводить до найбільших і найменших змін у зашифрованому тексті? Змініть ключ на «1111111111» та повторіть усі запропоновані операції. Оберіть випадковий ключ і виконайте всі дослідження для нього. Поясніть отримані результати.

2. Зашифруйте на випадковому ключі текст «00000000». Поясніть отримані результати. Змінюйте по одному біту вхідного тексту та послідовно зашифруйте його на тому ж ключі. Скільки бітів зашифрованого тексту змінилося? Зміна яких бітів відкритого тексту найбільше і найменше впливає на зашифрований. Ті ж операції виконайте для тексту «11111111» та довільного відкритого тексту. Поясніть отримані результати.

3. Виконайте шифрування довільного тексту на довільному ключі. Починаючи з першого рядка, змініть структуру матриць заміни, щоразу виконуючи шифрування. Як змінюється результат? Визначте, яка зміна структури матриць привела до найкращих і найгірших результатів.

Порівняйте розсіювальні властивості спрощеного алгоритму з такими ж властивостями реального DES. Дані про DES можна лекційному курсі та в Інтернет-джерелах.

4. Здійсніть атаку методом повного перебору ключів («грубою силою») визначте скільки часу знадобилося Вашому комп'ютеру для повного перебору ключів. Скільки всього ключів довелося перебрати? Скільки ключів довелося перебрати для досягнення успіху атаки?

.

Зміст звіту

1. Результати досліджень.
2. Протокол Ваших дій.
3. Висновки з досліджень впливу зміни ключа, вхідного тексту та матриць заміни на результати шифрування.

ЛАБОРАТОРНА РОБОТА №5

Тема: Генератори псевдовипадкових послідовностей

Мета: Вивчення методів генерування псевдовипадкових послідовностей

Теоретичні відомості

1. Основні способи генерування псевдовипадкових чисел

Псевдовипадкові числа, рівномірно розподілені в інтервалі $(0,1)$, можна отримати з допомогою трьох основних способів – апаратного, табличного та алгоритмічного.

Апаратний спосіб для генерування випадкових чисел використовує електронні пристрої – генератори випадкових чисел, які служать зовнішніми пристроями ЕОМ. В основу таких генераторів покладено використання фізичного ефекту шумів в електронних і напівпровідникових приладах (так звані “генератори білого шуму”).

Табличний спосіб реалізується шляхом формування відповідного файлу, в якому записані конкретні значення послідовності випадкових чисел в оперативній або зовнішній пам’яті ЕОМ.

Алгоритмічний спосіб ґрунтується на формуванні випадкових величин в ЕОМ з допомогою спеціальних програм.

Переваги та недоліки зазначених способів наведені в табл.1.

На практиці в основному віддають перевагу алгоритмічному способу генерування псевдовипадкових чисел.

Таблиця 1

Переваги та недоліки основних способів генерування псевдовипадкових чисел

Спосіб	Переваги	Недоліки
Апаратний	Кількість чисел, які генеруються, необмежена. Використовується невелика кількість операцій ЕОМ. Не потребує місця в пам’яті ЕОМ	Потрібна періодична перевірка якості послідовності випадкових чисел. Повторення ідентичної послідовності неможливе. Необхідно використовувати спеціальний пристрій
Табличний	Перевірка якості послідовності виконується один раз – у процесі формування файлу. Можливе повторення ідентичних послідовностей випадкових чисел	Кількість чисел необмежена. При розміщенні в оперативній пам’яті файл займає багато місця. При розміщенні в зовнішній пам’яті зростає час звертання до файлу
Алгоритмічний	Перевірка якості послідовності виконується один раз – при випробуванні програми. Можливе багаторазове повторення послідовності. Займає мало місця в пам’яті ЕОМ. Не використовуються зовнішні пристрої	Кількість чисел послідовності обмежена внаслідок періодичності датчика. Необхідні витрати машинного часу на отримання псевдовипадкових чисел

2. Методи отримання псевдовипадкових чисел

Метод серединних квадратів – це одна з перших процедур, яку запропонували в 1946р. Фон Нейман та Метрополіс. В наш час цей метод має лише історичний інтерес, тому що його роботу важко проаналізувати, працює він порівняно повільно й не дає статистично задовільних результатів. Наведемо основні кроки алгоритму, який реалізує даний метод:

Взяти довільне n – значне число.

Піднести це число в квадрат, і, якщо потрібно, доповнити його ліворуч нулями до $2n$ – значного числа.

Взяти n цифр з середини цього числа як наступне випадкове число.

Перейти до кроку 2.

Щоб отримати числа, рівномірно розподілені в інтервалі $(0,1)$, достатньо промасштабувати (тобто поділити на 10^n) результати, знайдені за допомогою описаного вище алгоритму. Обравши за початкове число 2152, в результаті роботи алгоритму отримаємо послідовність

$$\begin{array}{ll} x_0 = 2152 & x_0^2 = 04631104 \\ x_1 = 6311 & x_1^2 = 39828721 \\ x_2 = 8287 & x_2^2 = 68674369 \end{array}$$

Масштабуючи, дістанемо 0,2152; 0,6311; 0,8287 і т.д.

Часто послідовність виявляється надто короткою:

$$\begin{array}{ll} x_0 = 4500 & x_0^2 = 20250000 \\ x_1 = 2500 & x_1^2 = 06250000 \\ x_2 = 2500 & x_2^2 = 06250000 \end{array}$$

У цій послідовності випадковість взагалі відсутня, тому що починаючи з другого числа весь час буде генеруватись 2500.

Конгруентні методи будуються на основі кількох рекурентних формул з використанням поняття конгруентності – порівнянності чисел за модулем.

Лінійний конгруентний метод, який запропонував Томсон, використовує формулу

$$x_{i+1} = (a * x_i + c) \pmod{m}$$

З обчислювальної точки зору змішаний метод складніший за мультиплікативний на одну операцію додавання, але можливість вибору додаткового параметра c дозволяє зменшити можливу кореляцію між генерованими числами.

Адитивний конгруентний метод працює за формулою

$$x_{i+1} = (x_i + x_{i-1}) \pmod{m}$$

При $x_0=$ та $x_1=1$ цей алгоритм призводить до послідовності Фібоначчі. Використання адитивного генератора, який працює за формулою

$$x_{i+1} = (x_i - x_{i-k}) \pmod{m}$$

дає кращі результати, але потребує більшого обсягу пам'яті ЕОМ внаслідок необхідності збереження значень, проміжних між x_{i-k} та x_i . Якісно цей генератор працює при значенні $k = 16$.

Квадратний конгруентний генератор

$$x_{i+1} = (ax_i^2 + bx_i + c) \bmod m.$$

Кубічний конгруентний генератор:

$$X_n = (aX_{n-1}^3 + bX_{n-1}^2 + cX_{n-1} + d) \bmod m$$

Генератор Ейхенауера – Лена із зворотом

$$x_{i+1} = \begin{cases} (ax_i^{-1} + c) \bmod m, & \text{якщо } x_i \geq 1 \\ c, & \text{якщо } x_i = 0 \end{cases}, \text{ де } x_i * x_i^{-1} = (\bmod m).$$

Комбіновані методи генерують “ще більш випадкові” послідовності за рахунок зростання часу генерації. Так, наприклад, можна використати два генератори псевдовипадкових чисел, які генерують відповідно послідовності x_0, x_1, \dots, x_n та y_0, y_1, \dots, y_n псевдовипадкових чисел, що мають значення від нуля до $m-1$, незалежними способами і отримати вихідне псевдовипадкове число із співвідношення:

$$Z_n = (x_n + y_n) (\bmod m)$$

При цьому бажано, щоб довжини періодів $\{x_n\}$ $\{y_n\}$ були взаємно простими числами.

Генератор BBS

Програмний генератор двійкових послідовностей BBS вважають одним із найсильніших програмних генераторів псевдовипадкових послідовностей.

Нехай є два простих числа, p і q , причому $p \equiv q \equiv 3 \bmod 4$. Добуток цих чисел $n = pq$ називається цілим числом Блума. Оберемо ще одне випадкове число x , взаємно просте з n та обчислимо $x_0 \equiv x^2 \bmod n$. Це число вважається стартовим числом генератора. Далі можна обчислити наступні біти послідовності за формулами: $x_i \equiv x_{i-1}^2 \bmod n$ і $S_i \equiv x_i \bmod 2$. Останнє означає, що як вихід генератора обирається молодший біт числа x . Отже, можемо записати:

$$x_0 = x^2 \bmod n$$

$$\text{for } i = 1 \text{ to } \infty$$

$$x_i = x_{i-1}^2 \bmod n$$

$$S_i \equiv x_i \bmod 2.$$

Наприклад, $p = 19; q = 23, p = q \equiv 3 \bmod 4, n = 437, x_0 = 233$.

Таблиця 2

Генерація чисел генератором BBS

i	0	1	2	3	4	5	6
x_i	101	150	213	358	123	271	25
S_i	1	0	1	0	1	1	1

Генератор Блум – Мікалі

Безпека цього генератора базується на проблемах обчислення дискретного логарифма в кінцевому полі.

Для реалізації цього генератора генерують два простих числа: g та p . Зародок x_0 породжує такий процес генерації: $x_{i+1} \equiv g^{x_i} \bmod p$. Виходом генератора буде 1, якщо $x_i < (p-1)/2$, і 0 – якщо ця нерівність не виконується. Якщо модуль p достатньо великий для того, щоб обчислення дискретного логарифму було обчислювально складною задачею, цей алгоритм безпечний.

Генератор RSA

Генератор RSA використовує алгоритм шифрування RSA для утворення псевдовипадкових послідовностей. Застосовується публічний ключ (e, n) :

оберемо випадкове число $x_0 < n$ і обчислимо:

for $i = 1$ *to* ∞

$$x_i = (x_{i-1})^e \bmod n$$

$$B_i \equiv x_i \bmod 2.$$

Безпека цього генератора ґрунтується на складності розкладання великого числа на множники, тобто на тій самій задачі, яка лежить в основі криптостійкості самої системи RSA.

LFSR

Це пристрій, що складається з регістру зсуву, здатного запам'ятовувати двійкові послідовності кінцевої довжини та схеми, яка реалізує додавання за модулем 2 (\oplus) вибраних бітів з регістра. Початкова послідовність породжується поліномом з коефіцієнтом у двійковому представленні (0; 1): $P_n = A_n x^n + A_{n-1} x^{n-1} + \dots + A_1 x + A_0$, де $A_n = A_0 = 1$, інші рівні 0 або 1.

Максимальний період генератора – $2^k - 1$, k – кількість тригерів у регістрі (ступінь полінома). P_n не ділиться на ніякий інший поліном; P_n є дільником $x^{n+1} + 1$.

Припустимо, що ми хочемо побудувати LFSR із періодом 7. Для цього вибираємо поліном $x^3 + x + 1$, період якого становить $2^3 - 1 = 7$. Вхідний потік – 0001 (рис.1).

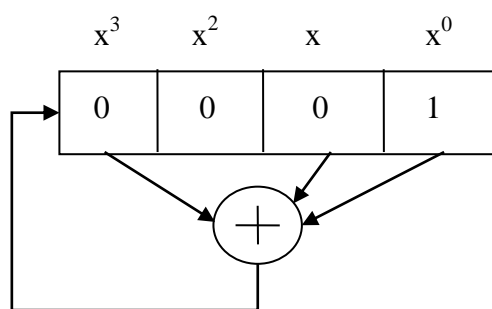


Рисунок 1 –Генератор LFSR-1

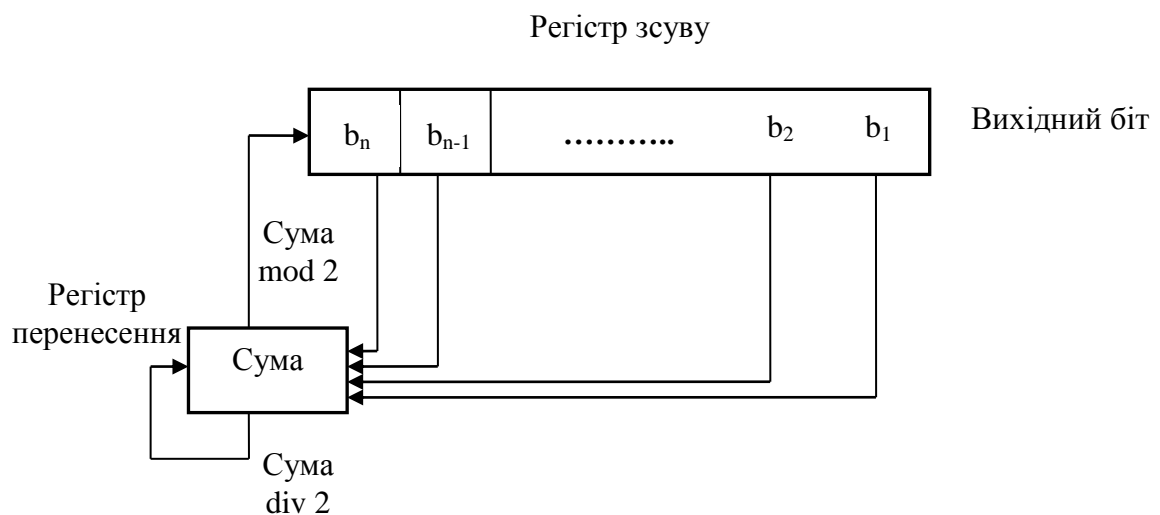
Приклад генерування послідовності 1

Вхідна послідовність				Вихідна послідовність
0	0	0	1	1
1	0	0	0	0
1	1	0	0	0
1	1	1	0	0
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
1	0	0	0	0
1	1	0	0	0
1	1	1	0	0
0	1	1	1	

Вихідна послідовність: 100011 100011 1...

FCSR

FCSR схожий на LFSR, в обох є регістр зсуву та функція зворотного зв'язку, різниця полягає у тому, що у FCSR є також регістр перенесення. Існують два варіанти реалізації FCSR: конфігурація Галуа та Фібоначчі. Ми розглядатимемо конфігурацію Фібоначчі. У порівнянні з LFSR, замість *xor* над усіма бітами відповідної послідовності, ці біти додаються один з одним і вмістом регістру перенесення. Результат *mod 2* стає новим бітом, результат *div 2* стає новим вмістом регістру перенесення (рис.2).



У таблиці 5 приведений приклад зміни вмісту регістрів та вихідна послідовність чисел.

Генерування послідовності

Регістр перенесення	Регістр зсуву	Вихідний біт
0	1010	
0	0101	0
1	0010	1
0	1001	0
0	1100	1
0	1110	0
0	1111	0
1	0111	1
1	1011	1
1	0101	1
1	1010	1
0	1101	0
1	0110	1
1	0011	0
1	0001	1
1	0000	1
0	1000	0
0	0100	0
0	1010	0
0	0101	0
1	0010	1

Як бачимо, період даної послідовності 010100111101011000, а довжина його становить $q-1=18$.

Основний підхід до проектування ГПВП на основі FCSR аналогічний як і для LFSR, а саме, використовуються декілька регістрів з різними довжинами, які об'єднані лінійним чи нелінійним способом. Розглянемо деякі види таких генераторів:

1. Генератор парності. Регістри FCSR або LFSR об'єднані функцією *xor*

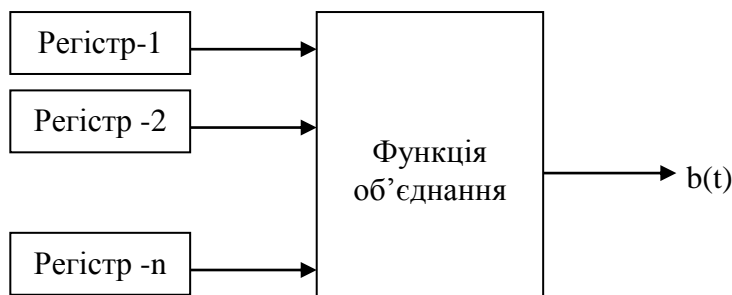


Рисунок 3 – Генератор парності

2. Генератор Геффа для регістрів FCSR або LFSR . Регістр-2,..., Регістр-n є входами мультимплексора, а Регістр-1 керує його виходом (рис.4).

Для трьох регістрів вихід генератора Геффа можна описати так:
 $b = a1 \wedge a2 \oplus \neg a1 \wedge a3$, де $a1, a2, a3$ – виходи Регістр-1, Регістр-2, Регістр-3.

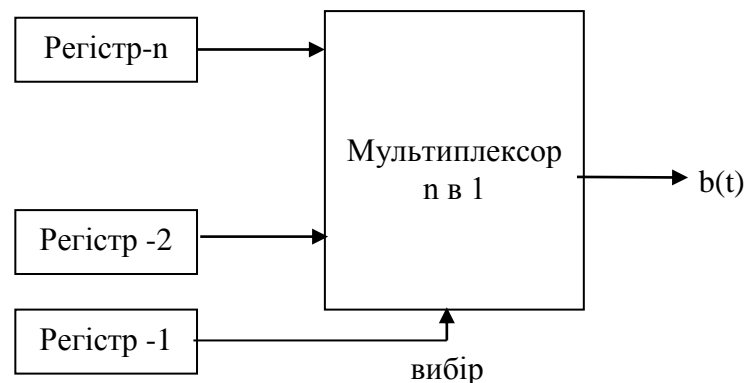


Рисунок 4 – Генератор Геффа

3. Пороговий (мажоритарний) генератор.

Для даного генератора необхідно використовувати непарну кількість регістрів. Якщо більше половини вихідних бітів регістра дорівнюють 1, то виходом генератора буде – 1, інакше – 0 (рис.5). Для трьох регістрів вихід генератора можна описати так: $b = a1 \wedge a2 \oplus a1 \wedge a3 \oplus a2 \wedge a3$.

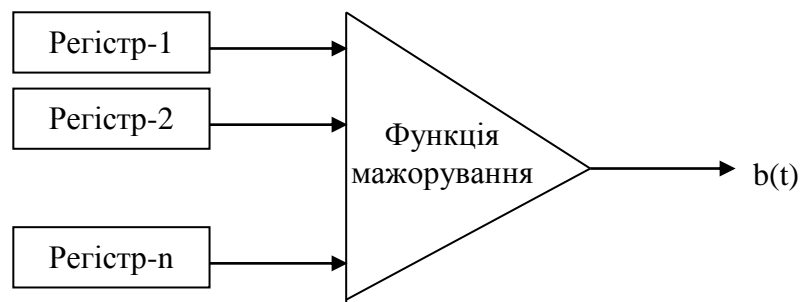


Рисунок 5 – Пороговий генератор

4. Генератор «stop-and-go». Використовуються три регістри. Регістр-2 змінює свій стан, якщо вихід Регістр-1 дорівнює 1, Регістр-3 змінює свій стан у протилежному випадку..

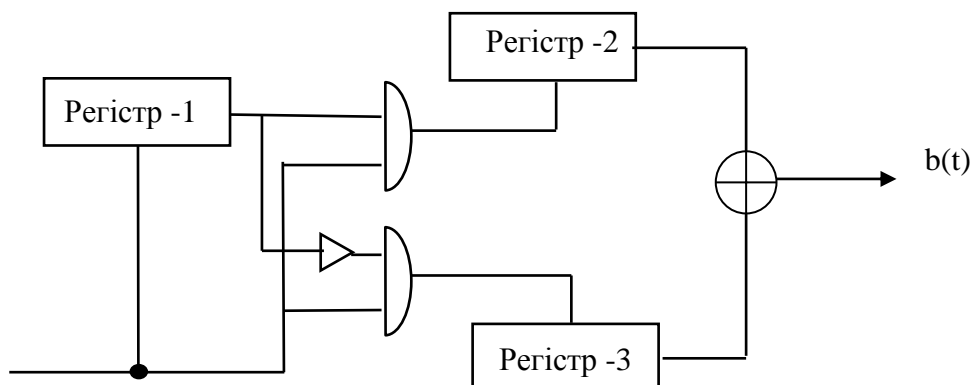


Рисунок 6 – Генератор «stop-and-go»

На рис.6 представлено структурну схему модифікованого генератора «stop-and-go», де Регістр-2 змінює свій стан, якщо вихід Регістр-1 дорівнює 1, Регістр-3 змінює свій стан, коли вихід Регістр-1 дорівнює 0. Вихід генератора є *xor* Регістр-2 та Регістр-3.

Розглянуті методи отримання псевдовипадкових чисел мають певні переваги та недоліки, що дозволяє в кожному конкретному випадку обрати найбільш доцільний метод. Крім того, генератори псевдовипадкових чисел чутливі до вибору початкових значень і констант. Найчастіше в моделюванні застосовується звичайний мультиплікативний генератор, який має високі статистичні характеристики та швидкодію.

Зміст завдання

1. Згідно із варіантом реалізувати програмний додаток, який генерує ПВПВ. Завдання варіантів 12-20 підвищеної складності.

2. Підібрати параметри даного генератора таким чином, щоб період послідовності мав максимальне значення.

3. Використовуючи тестування NIST STS перевірити статистичні властивості послідовності (для тестування необхідно мінімум 1000000 біт ПВП).

Варіанти індивідуальних завдань

№ варіанта	ГПВП	Додаткові відомості
1	2	3
1.	Лінійний конгруентний генератор	Значення початкових даних генераторів у табл.5
2.	Аддитивний генератор	Значення початкових даних генераторів у табл.5
3.	Квадратний конгруентний генератор	Значення початкових даних генераторів у табл.5
4.	Генератор на основі LFSR	Твірний поліном $g(x) = x^8 + x^4 + x^3 + x^2 + 1$
5.	Генератор на основі FCSR	Відвідна послідовність (6,4,2)
6.	Кубічний конгруентний генератор	Значення початкових даних генераторів у табл.5
7.	Генератор за методом квадратів	Початкові параметри обрати самостійно
8.	Генератор Ейхенауера – Лена із зворотом	Початкові параметри обрати самостійно
9.	Генератор BBS	Початкові параметри обрати самостійно
10.	Генератор Блум – Мікалі	Початкові параметри обрати самостійно
11.	Генератор RSA	Початкові параметри обрати самостійно
12.	Генератор Геффа на основі LFSR	Твірні поліноми $g1(x) = x^4 + x + 1; g2(x) = x^2 + x + 1;$ $g3(x) = x^7 + x^3 + 1$
13.	Генератор Геффа на основі FCSR	Відвідні послідовності (3,2); (4,2); (5,2,1)
14.	Генератор stop-and-go на основі LFSR	Твірні поліноми $g1(x) = x^4 + x + 1; g2(x) = x^2 + x + 1;$ $g3(x) = x^7 + x^3 + 1$

1	2	3
15.	Генератор stop-and-go на основі FCSR	Відвідні послідовності (3,2); (4,2); (5,2,1)
16.	Генератор парності на основі LFSR	Твірні поліноми $g1(x) = x^4 + x + 1$; $g2(x) = x^2 + x + 1$; $g3(x) = x^7 + x^3 + 1$
17.	Генератор парності на основі FCSR	Відвідні послідовності (3,2); (4,2); (5,2,1), N=500
18.	Пороговий генератор на основі LFSR	Твірні поліноми $g1(x) = x^4 + x + 1$; $g2(x) = x^2 + x + 1$; $g3(x) = x^7 + x^3 + 1$
19.	Пороговий генератор на основі FCSR	Відвідні послідовності (3,2); (4,2); (5,2,1)
20.	Комбінація будь-яких двох генераторів	Із варіантів 1-11 на вибір

Таблиця 5.

№	Значення початкових даних генераторів конгруентних генераторів				
	X_0	X_1	A	C	B
1	10253	275211	09865	10041	16
2	04527	21045	10057	05701	17
3	11429	22153	02359	14277	18
4	22111	24765	09867	00555	19
5	12121	25567	12353	21333	20
6	02263	00329	21351	31575	21
7	10133	21359	30899	01267	22
8	12999	01977	21455	08989	23
9	00579	15467	21579	30211	24
10	10101	15743	24211	03999	25

Зміст звіту

1. Вихідні дані варіанту індивідуального завдання.
2. Висновок за результатами проведеного аналізу.
3. Текст програми.

ЛАБОРАТОРНА РОБОТА № 6

Тема: Використання шифрувальної системи RSA для електронного цифрового підпису

Мета: створити просту криптографічну систему цифрового підпису на основі системи шифрування RSA та дослідити її роботу.

Теоретичні відомості

Криптографічна система RSA (Rivest, Shamir, Adleman) належить до криптографічних систем із відкритим ключем. Її стійкість зумовлена великими проблемами при знаходженні розкладання великих простих чисел на множники.

Для того, щоб організувати передачу шифрованих повідомлень за допомогою криптосистеми RSA, необхідно зробити таке:

- За допомогою спеціальних алгоритмів згенерувати два великих простих числа p і q , які необхідно тримати в таємниці.

- Повідомити відправнику повідомлень (або розмістити у відкритому каталозі) число $n = pq$, а також випадкове ціле число E , взаємно просте з добутком $(p-1)(q-1)$.

Для розшифровки повідомлень, зашифрованих на відкритому ключі n , E , отримувачу необхідно мати число D , яке є мультиплікативним оберненим числа E за модулем $(p-1)(q-1)$, тобто $DE \equiv 1 \pmod{(p-1)(q-1)}$. Знайти таке число дуже просто, оскільки найбільший спільний дільник E і $(p-1)(q-1)$ якраз і дорівнює одиниці за вибором E .

Отже, відправник знає свій відкритий ключ n , E , а отримувач, окрім того, знає ще й свій секретний ключ D .

Довільне відкрите повідомлення можна зобразити у вигляді послідовності цілих чисел із деякого інтервалу. Будемо вважати, що відправник передає секретне повідомлення у вигляді $X_1 \dots X_n$, $0 < X_i < n-1$, для всіх i від 1 до n .

Відправник для кожного блока X_i вираховує $C_i = X_i^E \pmod{n}$ і передає його відкритим каналом зв'язку.

Маючи n , D і C_i , отримувач може розшифрувати повідомлення, використовуючи співвідношення $X_i = C_i^D \pmod{n}$.

Розглянемо як приклад випадок $p = 3$, $q = 11$, $n = 3 \cdot 11 = 33$, $E = 7$, $D = 3$. Легко переконатися, що кожне з чисел E і D взаємно просте з $(p-1)(q-1) = 20$. Для передачі повідомлення $M = \langle 02 \rangle$ відправнику треба обчислити

$C = 2^7 \pmod{33} = 29$. Отримувач може розшифрувати повідомлення за допомогою такої операції: $X = 29^3 \pmod{33} = 2$.

Для шифрування зручно використовувати таблицю 1.

Таблиця 1 Таблиця заміни при шифруванні

А	Б	В	Г	Д	Е	Є	Ж	З	И	І
00	01	02	03	04	05	06	07	08	09	10
І	И	К	Л	М	Н	О	П	Р	С	Т
11	12	13	14	15	16	17	18	19	20	21
У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	
22	23	24	25	26	27	28	29	30	31	32

Система шифрування RSA може бути застосована для цифрового підпису. У випадку підпису повідомлення M відправник обчислює $S = M^D \bmod n$. Отримувач, який має M і S , перевіряє справедливість співвідношення $M = S^E \bmod n$ і впевнюється у справжності повідомлення M .

Приклад. Нехай $p = 3$, $q = 11$, $n = 3 \cdot 11 = 33$, $E = 7$, $D = 3$. Тоді відправник повідомлення $M = \langle 02 \rangle$ обчислює цифровий підпис $S = 2^3 \bmod 33 = 8$ і відправляє повідомлення $\langle 02, 08 \rangle$ отримувачу. Той, в свою чергу, перевіряв справжність повідомлення $\langle 02 \rangle$, обчисливши $M = 8^7 \bmod 33 = 2$.

Насправді підписують не саме повідомлення, а його хеш-образ.

Ми будемо користуватися найпростішою хеш-функцією, яка дуже недосконала й може викликати значні колізії. Однак вона дуже проста і не потребує витрат машинного часу, а також складного програмування. Ця функція просто підсумовує всі значення символів із табл. 1 за модулем 33:

$$H(M) = \sum m_i \bmod 33.$$

До отриманого в такий спосіб числа застосовують алгоритм прикладу отримуючи зашифрований цифровий підпис.

Отримувач, маючи повідомлення і цифровий підпис, розшифровує текст повідомлення, знаходить хеш-функцію від нього за вище вказаною формулою, розшифровує цифровий підпис і порівнює отримані значення. Якщо вони однакові повідомлення й цифровий підпис істинні.

Зміст завдання

Створити просту криптографічну систему цифрового підпису на основі шифру RSA. 2. Перевірити її роботу

1. Підгрупа розбивається на пари за бажанням. Один студент виконує цифровий підпис повідомлення, а другий цей підпис перевіряє.

2. Перший студент створює криптографічну систему на основі алгоритму RSA, викладеного в теоретичній частині.

3. Система шифрування повинна задовольняти такі вимоги: 1) читати з текстового файлу відкрите повідомлення; 2) шифрувати повідомлення за допомогою публічного ключа; 3) обчислювати просту хеш-функцію повідомлення у вигляді (3); 4) обчислювати цифровий підпис знайденої хеш-функції і записувати його значення у файл (той самий, в якому міститься повідомлення, або інший).

4. Другий студент створює систему перевірки електронного підпису. Система повинна задовольняти такі вимоги: 1) читати з текстового файла

зашифроване повідомлення та цифровий підпис; 2) розшифровувати повідомлення за допомогою приватного ключа D і знаходити хеш-функцію (3); 3) розшифровувати цифровий ключ і порівнювати отримане значення хеш-функції з обчисленим у п.2); 4) робити висновок про істинність отриманого повідомлення й цифрового підпису.

5. Замініть цифровий підпис довільним числом із діапазону 0-32 і знову перевірте, чи «помітить» програма розшифровки заміну.

Зміст звіту

1. Вихідні дані та розрахунки завдання.
2. Тестування програми для шифрування/дешифрування (копії екрану).
3. Тестування програми для створення ЦЕП та його перевірки (копії екрану).
4. Висновок про якість роботи системи.
5. Текст програми

ЛАБОРАТОРНА РОБОТА № 7

Тема: Відкритий розподіл криптографічних ключів

Мета: ознайомитися з відкритим розподілом криптографічних ключів її допомогою алгоритму Діффі-Хеллмана.

Теоретичні відомості

Проблема адміністрування криптографічними ключами вважається основним недоліком симетричних криптоалгоритмів. Цю проблему можна розв'язати за допомогою асиметричної криптографії, тобто взагалі не використовувати симетричні криптоалгоритми. Однак такий підхід вважають нерациональним, оскільки асиметричні алгоритми працюють значно повільніше за симетричні й не можуть використовуватися в ряді важливих криптографічних операцій.

Одним із популярних алгоритмів узгодження ключів є алгоритм Діффі-Хеллмана.

Нехай учасники інформаційного обміну, сторони А і В, домовилися використати цей алгоритм для обміну ключами. Для цього необхідно виконати такі обчислення.

Спочатку А і В обирають велике просте число p , модуль системи. Для цього для числа p обирають первісний корінь a . Числа p і a відкрито передають каналам зв'язку, так щоб їх мали обидві сторони.

Далі виконується такий протокол:

- 1) Користувачі домовляються про використання відкритих даних a, p .
- 2) Незалежно один від одного обирають навчання секретні числа X_A і X_B ($1 \leq X_{A,B} \leq p-1$)
- 3) А і В незалежно обчислюють $y_A = a^{X_A} \bmod p$ та $y_B = a^{X_B} \bmod p$
- 4) А і В обмінюються y_A та y_B .
- 5) Користувач А, отримавши y_B обчислює: $y_B^{X_A} \bmod p = (a^{X_B})^{X_A} \bmod p$.
- 6) Користувач В, отримавши y_A обчислює: $y_A^{X_B} \bmod p = (a^{X_A})^{X_B} \bmod p$
- 7) $k = a^{X_A X_B} \bmod p$ – спільний ресурс.

Отже, сторони А і В отримали однаковий криптографічний ключ, не пересилаючи його каналами зв'язку. Ніхто з осіб, що прослуховують цей канал, не зможе обчислити значення ключа. Адже їм відомі тільки a, p, y_A, y_B , а для знаходження ключа необхідно розв'язати задачу дискретного логарифмування. Тому А і В мають цілком таємний ключ, який більше ніхто не знає.

Вибір a і p може помітно впливати на безпеку системи. Найголовніше, це те, що повинно бути великим, таким, щоби задача дискретного логарифмування у скінченному полі була складною обчислювальною проблемою. Можна обирати довільне a , яке є первісним коренем за модулем p ; немає причин, за якими не можна було б обрати a найменшим із можливих, навіть однорозрядним. Навіть необов'язково, щоб a було первісним коренем, воно

повинно лише утворювати досить велику підгрупу мультиплікативної групи за модулем p .

Приклад 1.

Згенерувати спільний ключ для двох користувачів. Нехай $a=7$, $p=13$.

Секретний ключ А – $X_A=5$; В – $X_B=11$.

$$Y_A = 7^5 \bmod 13 = 11$$

$$Y_B = 7^{11} \bmod 13 = 2$$

$$2^5 \bmod 13 = 32 \bmod 13 = 6$$

$$11^{11} \bmod 13 = 6$$

Це і є секретний ключ – 6.

Зміст завдання

Створити просту систему відкритого розподілу криптографічних ключів за алгоритмом Діффі-Хеллмана. Перевірити її роботу.

Для цього:

1. Оберіть просте число p і його первісний корінь a . Число p повинно бути як мінімум 4-значним. Для пошуку простих чисел можна скористатися кодом, наведеним у додатку.
2. Для обраного p знайдіть первісний корінь a . Для цього скористайтесь малою теоремою Ферма.
3. Для обраних p і a виконайте обчислення k і k' та порівняйте їх.
4. Зробіть висновок про якість роботи Вашої системи обміну ключами.

Зміст звіту

1. Результати досліджень.
2. Протокол дій.
3. Код програми.
4. Висновок з лабораторної роботи.

ЛАБОРАТОРНА РОБОТА № 8

Тема: Криптологічний пакет CrypTool

Мета: Ознайомитись з основними можливостями пакету CrypTool. Навчитися проводити шифрування і дешифрування тексту класичними методами, симетричними та асиметричними алгоритмами, в тому числі, застосовуючи демонстраційний модуль пакету CrypTool.

Теоретичні відомості

CrypTool (e-learning software for cryptography and cryptanalysis) — найпоширеніший в галузі криптології безкоштовний програмний засіб з відкритим вихідним кодом. CrypTool було презентовано як безкоштовне програмне забезпечення з відкритим кодом для залучення до проекту інших додаткових розробників. Нині багато людей працюють по всьому світу над удосконаленням і технічним обслуговуванням цього програмного продукту. CrypTool складається з п'яти незалежних складових (CT1, CT2, JCT, CTO, MTC3), що пов'язані між собою, але мають відмінні принципи роботи й використовуються в різних аспектах.

CrypTool 1 (CT1) є першим найповнішим і досконалим варіантом реалізації майже усіх наявних криптографічних алгоритмів та функцій. CT1 написаний на мові програмування C++ під операційну систему Windows. Програмне забезпечення доступне англійською, німецькою, польською, іспанською та сербською мовами. Нещодавно розпочався процес локалізації CT1 грецькою та російською мовами. Певним недоліком є відсутність українського інтерфейсу, проте в майбутньому є ймовірність появи версії і цією мовою. CT1 має простий графічний інтерфейс та зручне меню, за допомогою якого користувач може створювати нові або відкривати вже існуючі текстові повідомлення, здійснювати їх редагування, зберігання, видалення тощо. Шифрування повідомлень відбувається у робочій області програми з використанням симетричних класичних алгоритмів (шифри Цезаря Віженера, Хілла, Атбаш, Плейфера, ADFGVX, Вернама, гомофонної заміни, маршрутною перестановки [1], Солітер, Сцитала) і сучасних алгоритмів (IDEA, RC2, RC4, RC6, DES, AES, MARS, Serpent, Twofish). Асиметричні алгоритми представлені шифром RSA, а змішані (гібридні) — шифрами RSA-AES та ECC –AES.

Для прикладу, під час вивчення теми «Класичні шифри» дисципліни «Криптологія» можна запропонувати розглянути перетворення повідомлення в середовищі CT1. Створимо повідомлення з текстом «WELL BEGUN IS HALF DONE» і зашифруємо його, використовуючи алгоритм Плейфера. Для цього оберемо у меню команду Encrypt/Decrypt. У діалоговому вікні, що відкриється, потрібно вказати символи, які будуть розділяти біграми (пари символів) з однаковими літерами й увести ключ, у даному випадку ми вибрали «PLAYFAIREXAMPLE» (див. рис. 1). Програма автоматично створить ключову матрицю, за якою відбуватиметься перетворення, після чого натискаємо кнопку

Encrypt. У головному вікні програми з'явиться два вікна, одне з яких міститиме розбите на біграми початкове повідомлення, а друге — шифротекст (див. рис. 2). Даний приклад забезпечує детальне наочне представлення процесу шифрування.

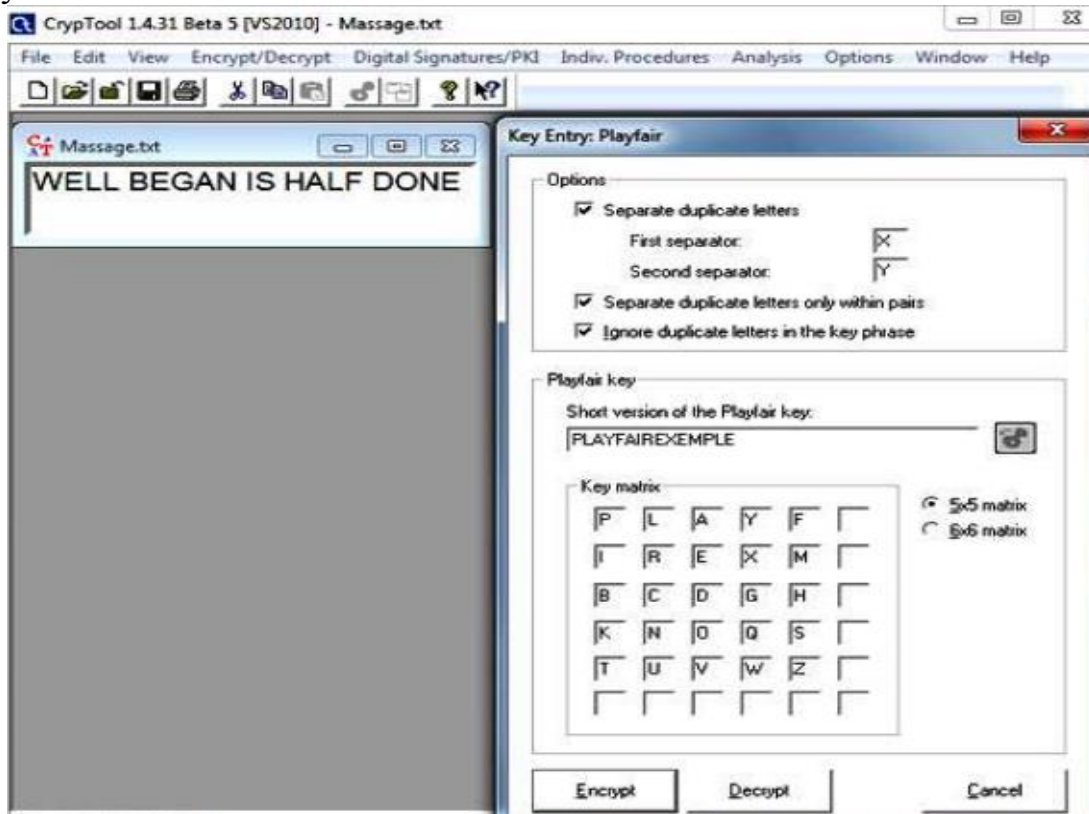


Рисунок 1 - Введення повідомлення та визначення параметрів перетворення у шифрі Плейфера

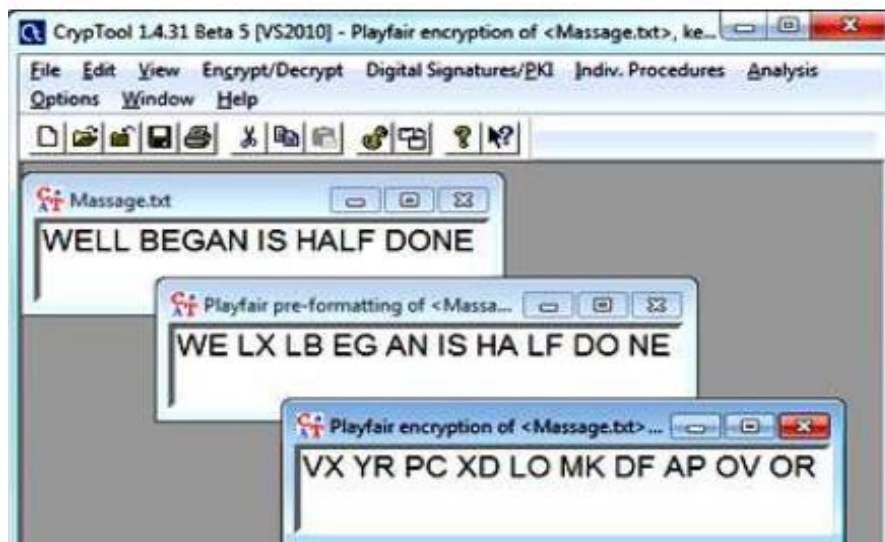


Рисунок 2 - Розбите на біграми початкове повідомлення та зашифроване повідомлення

Пакет СТ1 забезпечує процедури створення і перевірки електронного цифрового підпису на основі декількох алгоритмів (RSA , DSA та ін.), генерування, експортування та імпортування ключів. Обчислення хеш-значення документа у програмі відбувається з використанням хеш-функцій MD2, MD4,

MD5, SHA, SHA-1, SHA-256, SHA-512, RIPEMD-160. Потужний математичний апарат програми сприяє поглибленому вивченню асиметричних алгоритмів, зокрема RSA, та дозволяє визначити, чи є число простим, здійснювати генерацію простих чисел у заданому діапазоні, факторизацію числа (розклад числа на прості множники). СТ1 передбачає вивчення криптографічних протоколів, таких як протоколи обміну ключами Діффі-Хелмана, мережевої автентифікації тощо.

Значною перевагою першої версії CrypTool є широкий діапазон анімації ідемонстрації шифрів Цезаря, Віженера, Енігми, DES, AES, RSA, цифрового підпису, алгоритму шифрування на основі еліптичних кривих тощо.

Під час вивчення теми «Симетричні шифри» пропонується розглянути один із найскладніших для вивчення симетричний блоковий алгоритм — DES (державний стандарт шифрування США), який доцільно продемонструвати покроково, пояснивши кожен крок детально (див. рис. 3).

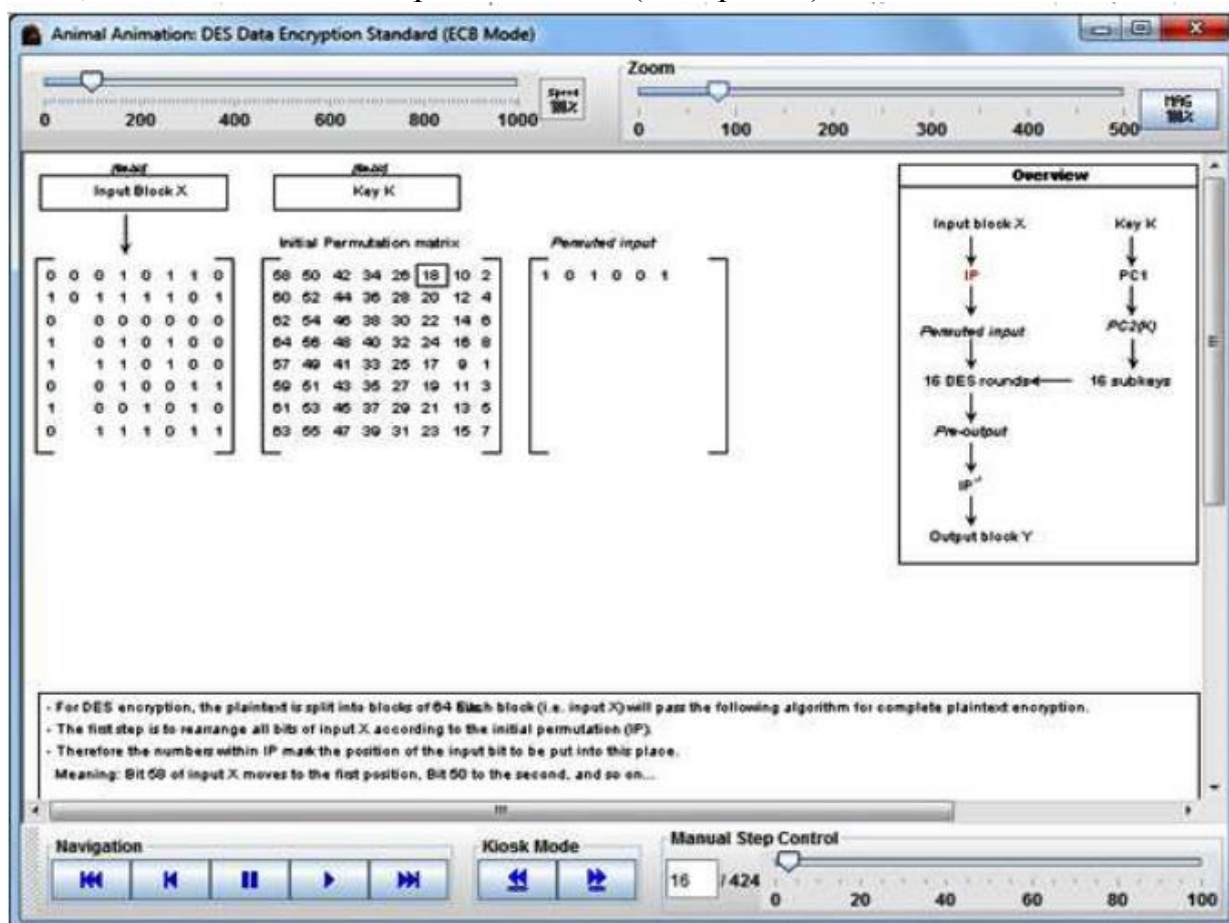


Рисунок 3 - Демонстрація початкової перестановки бітів
вхідного 64-бітового блоку у шифрі DES

Щодо **CrypTool 2 (СТ2)**, то це сучасний наступник СТ1, який реалізує концепцію візуального програмування та виконання каскадів криптографічних процедур. Отже, криптографічний алгоритм з усіма складовими (наприклад, відкритий текст, ключ, шифротекст тощо) є єдиним проектом. Програма має зручний сучасний графічний інтерфейс, завдяки чому володіє яскравою наочністю й отримала широке розповсюдження (див. рис. 4). СТ2 в даний час доступна німецькою й англійською мовами.

Унікальною особливістю СТ2 є модульна конструкція, яка пропонує набір інструментів, котрі можуть бути об'єднані в нові проекти для реалізації криптографічних алгоритмів. Викладач має можливість використовувати це для підготовки практичних завдань для студентів. Також засобами СТ2 можна забезпечити творчу роботу, пізнавальну активність студентів, яка полягає у власному створенні криптографічних алгоритмів. Для зручності користувачів, яким важко працювати з проектами, що є набором модулів, розроблено компонент Wizard для візуалізації та кращого розуміння більшості криптографічних алгоритмів та процедур. За винятком вищезгаданих шифрів, представлених у першій версії пакета CrypTool, у СТ2 додатково реалізовані такі симетричні класичні алгоритми

Зазначимо, що у СТ2, на відміну від усіх інших версій, є можливість задавати алфавіт повідомлення, яке підлягатиме перетворенню. Це означає, що відкритий текст може бути **українською або російською мовами**, на відміну від СТ1.

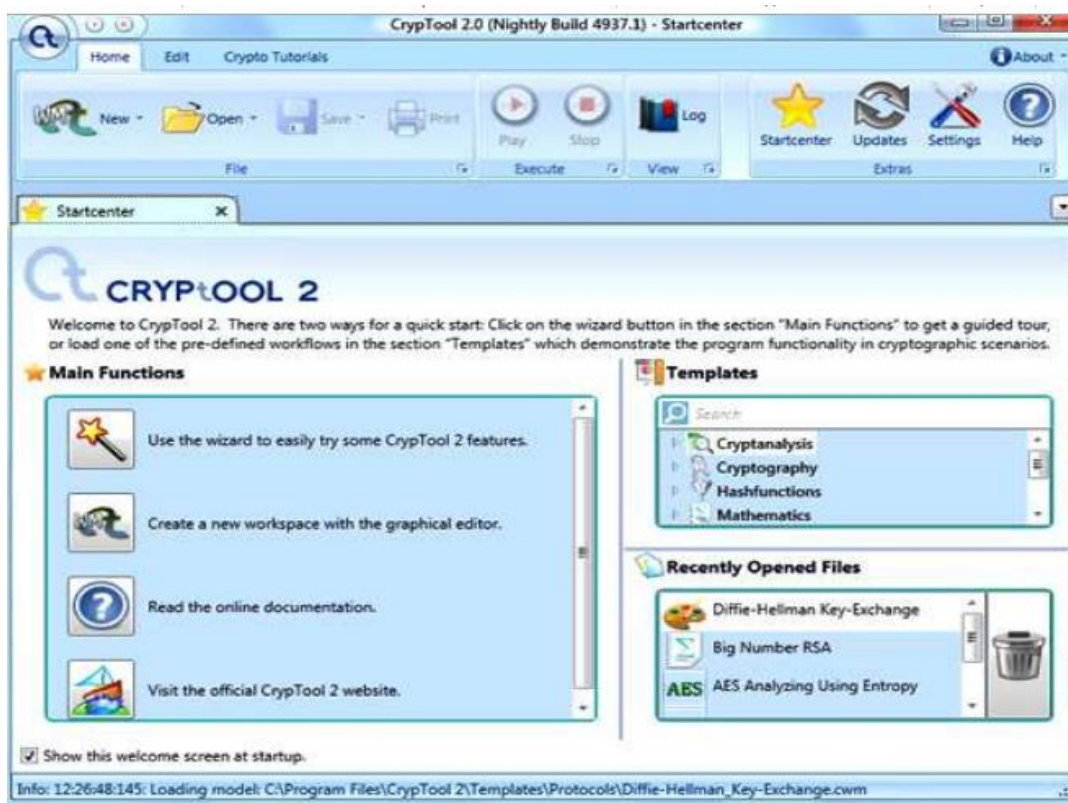


Рисунок 4 - Вікно завантаження *CrypTool 2*

Повідомлення «З ВИДИМОГО ПІЗНАВАЙ НЕВИДИМЕ» та ключ «ВІЖЕНЕР» записуємо у відповідні текстові поля (Message, Key). Далі визначаємо параметри перетворень, використовуючи наступні дві комірки (Vigenere), що реалізують алгоритми шифрування та дешифрування. На панелі інструментів натискаємо кнопку Play. У текстових полях (Encrypted, Result) одночасно з'являться зашифроване повідомлення та результат його дешифрування (див. рис. 5). Цей приклад забезпечує наочність представлення процесу шифрування та дешифрування одночасно.

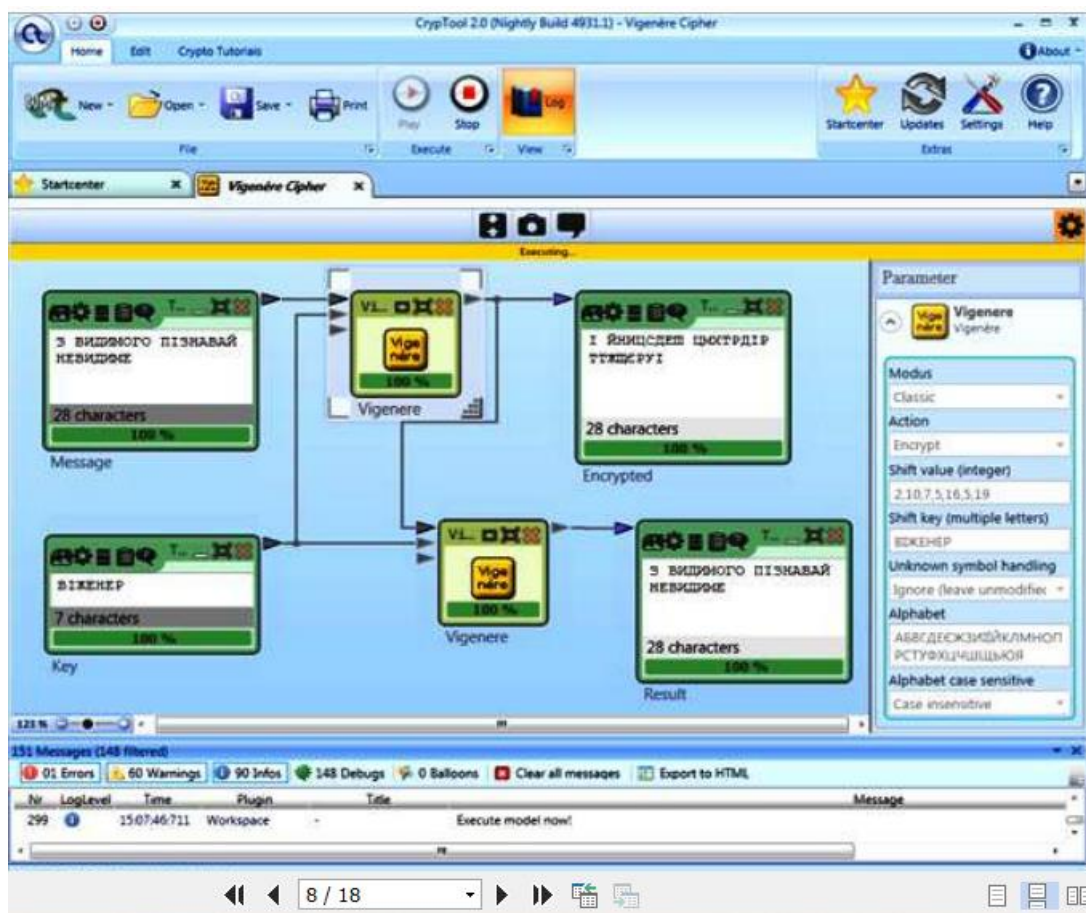


Рисунок 5 - Перетворення повідомлення
за допомогою шифру Віженера у СТ2

У програмному забезпеченні СТ2 відсутня схема створення та перевірки електронного цифрового підпису, проте автоматизовано більшість відомих криптографічних хеш-функції (починаючи з MD5 і закінчуючи Whirlpool).

Новизною у СТ2 є використання **стеганографії**, яку використовують спільно з методами криптографії. Під стеганографією найчастіше розуміють приховування інформації в текстових, графічних або аудіофайлах шляхом використання спеціального програмного забезпечення.

JCrypTool (JCT) — незалежний від платформи програмний засіб, сумісний з операційними системами MAC, Windows і Linux, а також обов'язково для роботи потребує встановлення Java 1.6 або новіше. JCrypTool має сучасний користувацький інтерфейс та підтримує англійську, німецьку або іспанську мови. JCrypTool має два режими роботи:

основний (за замовчуванням) — документо-орієнтований, забезпечує доступ до криптографічних функцій через головне меню та CryptoExplorer (провідник у правій частині вікна).

функціональний — функціонально-орієнтований, забезпечує доступ до сховища ключів, роботу з бібліотекою FlexiProvider, містить розширений перелік алгоритмів і дозволяє вибір всіх можливих параметрів перетворення.

JCT містить багато функцій для обчислення хешу документа та створення імітовставки (CBC MACs, CMAC, Two-Track-MAC). Особливістю JCT є використання процедур шифрування й електронного підпису вмісту XML-

документів. Візуалізація алгоритмів містить значний перелік криптографічних алгоритмів і процедур.

CrypTool-Online — це он-лайн-версія програми електронного навчання CrypTool, яка, у першу чергу, призначена для вивчення класичних шифрів. Криптографічні перетворення відбуваються безпосередньо у браузері комп'ютера або смартфона. Крім великого різноманіття класичних шифрів, CrypTool-Online пропонує такі методи кодування: ASCII, код Бекона, генератор штрих-кодів, код Хаффмана, код Морзе. У CrypTool-Online представлено шифр AES, генератор паролів, також є можливість вивчення теорії чисел на основі навчальної гри Тахман (збирач податків). Варто зазначити, що практично всі наведені на сайті методи мають детальне теоретичне пояснення, підкріплені прикладами та ілюстраціями.

Зміст завдання

1. Вивчення особливості побудови симетричних шифрів. Використовуючи програмну установку CrypTool необхідно здійснити порівняльний аналіз для 2-х шифрів: обрати 1 класичний та 1 сучасний шифр самостійно. Порівняльний аналіз слід оформити у вигляді таблиці. Параметр за яким порівнюються класичні шифри та сучасні шифри: тип симетричного шифру; основна конструкція шифру; основні математичні перетворення; режими роботи шифру; розмір загальносистемних параметрів; розмір ключа; кількість та розмір циклових ключів; число раундів шифрування/дешифрування; довжина блоку повідомлення, що обробляється; складність атаки груба сила; статистична безпечність криптограми. Зробити стислі висновки, щодо аналізу.

2. Ознайомитись з алгоритмами і методами симетричного блокового шифрування. Отримати навички роботи з шифрами DES-EBC, DES-CBC. Ознайомитись з алгоритмом AES. Навчитися проводити шифрування і дешифрування тексту, застосовуючи демонстраційний модуль пакету CrypTool.

3. Проаналізувати роботу асиметричних шифрів RSA та створення ЦЕП з використанням RSA та DSA. Створити текстовий файл «Ваше прізвище.txt». Згенерувати нову пару ключів. Експортувати пару ключів у файл «keys_ Ваше прізвище.p12». Імпортувати пару ключів вашого сусіда. Отримати від нього підписаний файл. Провести верифікацію ЦЕП.

4. Додатково: вивчення ще одного, будь-якого, криптографічного чи стеганографічного алгоритму, запропонованого у пакеті CrypTool.

Зміст звіту

1. Результати досліджень в електронному варіанті.
2. Висновки з лабораторної роботи.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Вербицький О. В. Вступ до криптології / О. В. Вербицький. – Львів, 1998. – 247 с.
2. Гапак О.М. Захист інформації в комп'ютерних системах. Навчально-методичний посібник для студентів напряму підготовки «комп'ютерна інженерія». – Ужгород: видавництво ПП «АУТДОР-ШАРК», 2015. – 172 с.
3. Гатчин Ю. А. Основы криптографических алгоритмов. Учебное пособие. / Ю. А. Гатчин, А. Г. Коробейников. – ГИТМО(ТУ), 2002. – 29 с.
4. Гундарь К. Ю. Защита информации в компьютерных системах / К. Ю. Гундарь, А. Ю. Гундарь, Д. А. Янишевський. – К.: «Корнейчук», С. 200.– 152.
5. Иванов М. А. Теория, применение и оценка качества генераторов псевдослучайных последовательностей / М. А. Иванов, И. В. Чугунков. – М.: Изд-во «КУДИЦ-ОБРАЗ», 2003. – 240 с.
6. Остапов С. Е. Основы криптографії: навчальний посібник / С. Е. Остапов, Л. О. Валь. – Чернівці: Книги–ХХІ, 2008. – 188 с.
7. Харин Ю. С. Математические и компьютерные основы криптологии: учеб. пособие / Ю. С. Харин, В. И. Берник, Г. В. Матвеев. – Минск: Новое знание, 1999. – 319 с.
8. Шнайер Б. Прикладная криптография: Протоколы, алгоритмы и исходные тексты на языке С / Б. Шнайер.– М. : Триумф, 2002. – 816 с.
9. Sibylle Hick. Reducing the complexity of understanding cryptology using CrypTool [Електронний ресурс] / Sibylle Hick, Bernhard Esslinger, Arno Wacker. — Режим доступу :
http://www.iis.org/CDs2012/CD2012SCI/EISTA_2012/PapersPdf/EA678TR.pdf.
10. The CrypTool Portal [Електронний ресурс]. — Режим доступу :
<http://www.cryptool.org/en>.
11. CrypTool 1 [Електронний ресурс]. — Режим доступу :
<http://www.cryptool.org/en/download-ct1-en/215-ct1-downloads-eng>.
12. JCrypTool — The cryptography e-learning platform [Електронний ресурс]. — Режим доступу : <http://www.cryptool.org/en/jct-documentation-en/jct-resources-en>.
13. CrypTool-Online [Електронний ресурс]. — Режим доступу :
<http://www.cryptool-online.org>
14. MysteryTwister C3 [Електронний ресурс]. — Режим доступу :
<http://www.mysterytwisterc3.org>.
15. Н.О.Загацька. Огляд різних версій пакета **Cryptool** як засобу захисту інформаційних ресурсів / Інформаційні технології і засоби навчання. 2012. №5 (31). Режим доступу до журналу: <http://www.journal.iitta.gov.ua>.

ЗМІСТ

Вступ.....	3
Лабораторна робота №1.....	4
Лабораторна робота №2.....	12
Лабораторна робота №3.....	16
Лабораторна робота №4.....	25
Лабораторна робота №5.....	30
Лабораторна робота №6.....	39
Лабораторна робота №7.....	42
Лабораторна робота №8.....	44
Рекомендована література.....	50

ДОДАТКИ

Додаток А

Таблиця Віжінера

[illegible]