**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**КАФЕДРА ЕЛЕКТРОННИХ ОБЧИСЛЮВАЛЬНИХ МАШИН**


# МЕТОДИЧНІ ВКАЗІВКИ

**до циклу лабораторних робіт**

**з курсу “Захист інформації в комп'ютерних системах” для студентів**

**базового напрямку 6.050102 “Комп’ютерна інженерія”**

Затверджено
на засіданні кафедри
електронних обчислювальних машин
Протокол №___ від “_____” _____ 2016 року

**Львів – 2016**

**Методичні вказівки до циклу лабораторних робіт з курсу " Захист інформації в комп'ютерних системах " для студентів базового напрямку 6.050102 "Комп'ютерна інженерія"** /Укл.: Ігнатович А.О. – Львів: Видавництво Національного університету "Львівська політехніка", 2016. – с.

Укладачі: Ігнатович А.О., асистент кафедри ЕОМ

Відповідальний за випуск:

Рецензенти:

# ЛАБОРАТОРНІ РОБОТИ №№ 6-7

# СУЧАСНІ КОМП'ЮТЕРИЗОВАНІ МЕТОДИ ШИФРУВАННЯ ТА ДЕШИФРУВАННЯ ТЕКСТОВИХ ПОВІДОМЛЕНЬ

## 1. МЕТА РОБОТИ

Мета роботи – дослідження статистичних властивостей відкритого тексту (надалі – ВТ) та шифрованого тексту (надалі – ШТ), вивчення простих методів шифрування та дешифрування інформації та їх властивостей для сучасних шифрів, які використовуються із застосуванням комп'ютерної техніки.

## 2. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

### 2.1. Криптографічні системи

Перетворення секретної інформації – це кодування даних, яке використовується для маскування інформації. Ці перетворення змінюють дані, представлені в явній формі, так, що вони стають нерозпізнаваними (з певною мірою надійності) неавторизованими людьми. Шифрування, тобто перетворення секретної інформації, є особливо ефективним засобом проти несанкціонованого доступу до повідомлень, які передаються по лініях зв'язку і неавторизованих доступів до даних, що зберігаються у віддалених файлах. Навіть просте перетворення секретної інформації є ефективним засобом, що дає можливість приховати її значення від більшості користувачів.

На практиці доцільно застосовувати шифр, суворо відповідний необхідним цілям. Якщо простий дешевий шифр буде в достатній мірі збільшувати "трудомісткість", розкриття зашифрованого повідомлення, то цього достатньо. У більшості випадків актуальність зашифрованого повідомлення з часом падає, особливо у військовій сфері. Тому часто достатньо щоб повідомлення не було дешифроване лише протягом певного часу. Але потрібно мати на увазі, що деякі схеми шифрування, які на перший погляд здаються стійкими, насправді легко розкриваються, особливо при

використанні ЕОМ. На практиці завжди потрібно прагнути до досягнення компромісу між вартістю шифрування і необхідним рівнем безпеки.

Щоб зрозуміти зміст зашифрованого повідомлення, неавторизовані особи повинні знати алгоритм шифрування (який їм іноді вдається дізнатися) і ключ (який не повинен бути їм доступний). Ключі не повинні зберігатися в системі довше, ніж це необхідно. У ідеальному випадку їм потрібно знаходитися там тільки на час дійсного використання.

Статистичні властивості відкритого тексту (ВТ) та шифрованого тексту (ШТ) досліджуються в криптології. Криптологія – наука, яка займається методами шифрування і дешифрування. Криптологія складається з двох частин – криптографії і криптоаналізу. Криптографія займається розробкою методів шифрування інформації, а криптоаналіз займається оцінкою сильних і слабких властивостей методів шифрування і розробкою методів, на основі яких будуються розкриття криптосистем.

Для кожної із мов (українська, польська, англійська, ...) є характерні статистичні характеристики ВТ і ШТ: йдеться про частоту використання усіх літер алфавіту у ВТ і ШТ. Окрім частотних характеристик використання окремих букв, криптоаналіз досліджує частоти використання буквосполучень у ВТ і ШТ.

У сучасній криптографії використовуються блокові шифри, які дозволяють одночасно зашифровувати декілька символів (2,3,4,5,6,7,...). В шифрі RSA рекомендується використовувати довжину блоку довжиною 1,5 символи, 2,5 символи, ... Такий шифрований блок має значно вищу стійкість ніж 2 символи, 3 символи, ... Сучасні комп'ютерні шифри майже непридатні для "ручного" використання за рахунок того, що вимагають виконувати багато обчислень, що суттєво знижує продуктивність праці криптографа.

До сучасних шифрів, які використовують апаратно-програмні комп'ютеризовані засоби відносяться шифр Хілла, шифр Фейстеля, шифр RSA, шифри на основі стандарту DES (стандарт USA) та інші. У всіх цих шифрах використовується блоковий метод шифрування і дешифрування інформації.

## 2.2. Шифр Хілла

Шифр Хілла – поліграмний шифр підстановки, заснований на лінійній алгебрі. Лестер С. Хілл винайшов цей шифр в 1929, і це був перший шифр, який дозволяв на практиці (хоча і з труднощами) оперувати більш ніж з трьома символами за раз. Подальше обговорення шифру припускає початкові знання матриць.

Шифрування інформації відбувається наступним чином. Кожній букві відкритого тексту присвоюється число. Для латинського алфавіту часто використовується найпростіша схема: A = 0, B = 1, ..., Z = 25, але це не є істотною властивістю шифру. Блок з n букв розглядається як n-мірний вектор і множиться на n*n матрицю по модулю 26. (Якщо в якості підстави модуля використовується число більше 26, то можна використовувати іншу числову схему – крім букв в алфавіт включають розділові знаки.) Ключем для шифру Хілла є матриця, яка представляється словом, чи довільним набором букв. Для шифрування використовується числова квадратна матриця (3x3, 4x4, 5x5,...). Матриця повинна мати обернену матрицю $\mathbb{Z}_{26}^{n}$ , щоб була можлива операція розшифрування.

Щоб розшифрувати повідомлення, необхідно звернути шифротекст назад у вектор і потім просто помножити на обернену матрицю ключа. Необхідно обговорити деякі складнощі, пов'язані з вибором шифрувальної матриці. Не всі матриці мають обернену. Матриця буде мати обернену в тому і тільки в тому випадку, коли її детермінант не дорівнює нулю і не має спільних дільників з основою модуля. Таким чином, якщо ми працюємо з основою модуля 26 , то детермінант повинен бути ненульовим і не ділитися на 2 і 13. Якщо детермінант матриці дорівнює нулю або має спільні дільники з основою модуля, то така матриця не може використовуватися в шифрі Хілла, і повинна бути обрана інша матриця (в іншому випадку шифротекст буде неможливо розшифрувати). Тим не менш, матриці, які задовольняють вищенаведеним умовам, існують в достатку.

Стійкість шифру. На жаль, стандартний шифр Хілла вразливий до

атаки, тому що він повністю лінійний. Криптоаналітик, який перехопить $n^2$ пар символ повідомлення зможе скласти систему лінійних рівнянь, яку зазвичай не складно вирішити. Якщо виявиться, що система не має розв'язку, то необхідно всього лише додати ще кілька пар символів повідомлення. Такого роду розрахунки засобами звичайних алгоритмів лінійної алгебри вимагає зовсім небагато часу.

### 2.2. Шифр Хілла із використанням маскуючих символів

Перед процедурою шифрування перед (або після) перед визначеними символами ВТ вставляються додаткові маскуючі символи. Необхідно вставляти таку кількість маскуючих символів щоби в кожний блок шифрування потрапляв хоча би один маскуючий символ. Маскуючі символи вибираються керованим генератором випадкових чисел таким чином щоби статистичний аналіз ВТ до вставляння і після вставляння маскуючих символів змінювався в сторону рівномірної частоти вживання символів. Якщо при перемноженні у матрицю символів ВТ вставляється хоча би один маскуючий символ, то при перемноженні змінюються всі результуючі символи ШТ. Запропонований спосіб шифрування інформації має високі параметри по криптостійкості, нескладно реалізується апаратним, програмним або комбінованим способом.

При необхідності отримати високі параметри по криптостійкості необхідно вставляти достатньо маскуючих символів, кількість яких може перевищувати кількість символів відкритого тексту. Розглянемо алгоритм додавання маскуючих символів: вставляється один маскуючий символ перед кожним символом ВТ і один маскуючий символ після символу ВТ. В цьому випадку ВТ з маскуючими символами буде мати таку конфігурацію: в кожному блоці (якщо $\boldsymbol{\mu} = 3$) буде один маскуючий символ перед символом ВТ, символ ВТ і один маскуючий символ після символу ВТ. Блок має такий вигляд: $\{m_i; v_i; m_i;\}$, де $m_i$ – маскуючий символ, $v_i$ – символ ВТ. Якщо конфігурація ВТ з маскуючими символами буде така, яка розглядалася вище, а $\boldsymbol{\mu} = 4$, тоді блоки будуть мати такий вигляд: перший - $\{m_i; v_i; m_i; m_i;\}$,

другий - $\{v_i; m_i; m_i; v_i;\}$, третій - $\{m_i; m_i\, v_i; m_i;\}$, четвертий - $\{m_i; v_i; m_i; m_i\}$, п'ятий - $\{m_i; v_i; m_i; m_i;\}$, п'ятий такий як перший і весь цикл з періодом чотири буде повторятися. Якщо приймається алгоритм вставляння маскуючих символів: вставляється два маскуючі символи перед кожним символом ВТ і нуль маскуючих символів після символу ВТ при $\boldsymbol{\mu} = 3$. В цьому випадку блок має такий вигляд : $\{m_i; m_i; v_i;\}$, де $m_i$ – маскуючий символ, $v_i$ – символ ВТ. Запропоновано декілька методів встановлення маскуючих символів – вибирати їх з можливого набору, які визначаються нерівномірністю статистичної характеристики розподілу символів за принципом: кожний маскуючий символ, який вставляється повинен покращувати рівномірність статистичної характеристики розподілу символів ВТ до якого добавляються маскуючі символи. Розглянемо статичний метод встановлення маскуючих символів – маскуючі символи завжди вставляються у наперед визначені місця відносно символів відкритого тексту. Наведені три варіанти статичного методу встановлення маскуючих символів для формату $\boldsymbol{\mu} = 3$: 1 - $\{v_i; v_i; m_i;\}$, 2 - $\{v_i; m_i; v_i;\}$, 3 - $\{ m_i; v_i; v_i;\}$, де $m_i$ – маскуючий символ, $v_i$ – символ ВТ. Динамічний метод встановлення маскуючих символів - маскуючі символи вставляються по кількості і на позиції в залежності від номера символу відкритого тексту і їх кількість буде мінятися на кожному раунді процедури вставляння. Дослідження виконувались для формату $\boldsymbol{\mu} = 5$. Після кожного символу ВТ вставляється обчислена кількість маскуючих символів згідно з формулою $\{v_i + [\, n_j \mod 5 \,] * m_i;\}$, де $n_j$ - порядковий номер символу ВТ.

Динамічна функція вставляння маскуючих символів дає додатковий ефект. Якщо у звичайному шифрі Хілла повторення в тексті можуть появлятися на відстанях, які кратні довжині ключа (число $\boldsymbol{\mu}$ не може бути дуже велике), то у встановленні маскуючих символів після кожного символу відкритого тексту у кількості від 0 до 5 при $\boldsymbol{\mu} = 5$ період повторення буде 105 символів (взамін 5). Модель встановлення маскуючих символів може бути статична, чи динамічна. Можна використати поліном для розрахунку місця і кількості встановлення маскуючих символів $N = ( n_1 + n_2 * z + n_3 * z^2 + \ldots + n_k$

$* z^{k-1}$ ) *mod* $\boldsymbol{\mu}$, де $N$ – кількість маскуючих символів, які вставляються після $z_i$ – го символу відкритого тексту, $n_i$ – коефіцієнти поліному, $\boldsymbol{\mu}$ – довжина блоку (формат шифрування). Сама процедура вставляння маскуючих символів і їх вилучення є процедурою, яка не зменшує продуктивність роботи криптографа. Якщо врахувати що маскуючі символи підбираються з допомогою генератора випадкових чисел з найменш вживаних символів у тексті. Такий алгоритм підбору маскуючих символів можна рахувати додатковим ключем для формування шифрованого тексту. Складність вилучення маскуючих символів не визначається їх номером чи назвою, так як вилучаються символи на відповідних позиціях шифрованого тексту. Якщо кількість маскуючих символів становить більше 50%, тоді частотний розподіл символів у шифрованому тексті наближається до рівноймовірного. Використання маскуючих символів має перспективу в напрямку створення шифрів підвищеної ефективності, які достатньо просто реалізовуються на сучасних комп'ютерних засобах. Графічна статична модель з форматом {$m_i$; $v_i$; $v_i$; $v_i$;}, де $m_i$ – маскуючий символ (чорна пунктирна вертикальна лінія), $v_i$ – символ ВТ (чорна вертикальна лінія), наведена на рис. 1. Формат шифрування – 4х4 ($\boldsymbol{\mu}$ = 4).

Рис. 1.

Графічна статична модель з форматом {$v_i$; $m_i$; $v_{i;}$; $v_{i;}$}, де $m_i$ – маскуючий символ (чорна пунктирна вертикальна лінія), $v_i$ – символ ВТ (чорна вертикальна лінія), наведена на рис. 2 . Формат шифрування - 4х4, ($\boldsymbol{\mu}$ = 4).

Рис.2.

Графічна статична модель з форматом{$v_i$; $v_i$; $m_i$ $v_i$;}, де $m_i$ – маскуючий символ (чорна пунктирна вертикальна лінія), $v_i$ – символ ВТ (чорна вертикальна лінія), наведена на рис. 3. Формат шифрування - 4х4, ($\boldsymbol{\mu} = 4$).

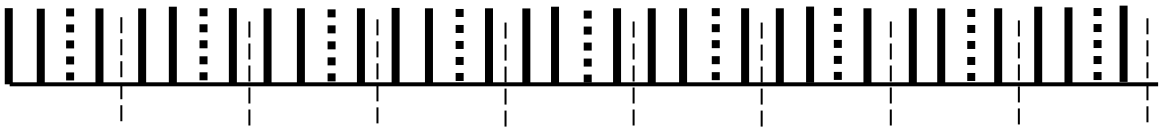


Рис. 3.

Графічна статична модель з форматом {$v_i$; $v_i$; $v_i$; $m_i$;}, де $m_i$ – маскуючий символ (чорна пунктирна вертикальна лінія), $v_i$ – символ ВТ (чорна вертикальна лінія), наведена на рис. 4. Формат шифрування - 4х4, ($\boldsymbol{\mu} = 4$).

Рис.4.

Динамічна графічна модель з форматом {$v_i$; $n_j$*$m_i$;}, де $m_i$ – маскуючий символ (чорна пунктирна вертикальна лінія), $v_i$ – символ ВТ (чорна вертикальна лінія), $n_j$ – коефіцієнт поступово змінюється від 0 до 5 (в залежності від порядкового номеру символу ВТ $v_i$). Формат шифрування – 4х4 ($\boldsymbol{\mu} = 4$).

Графічна динамічна модель встановлення маскуючих символів з описаними вище параметрами наведена на рис. 5.

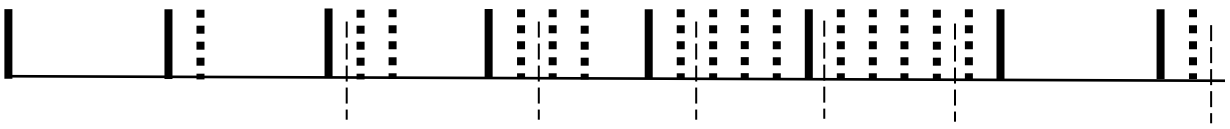


Рис. 5 (а).

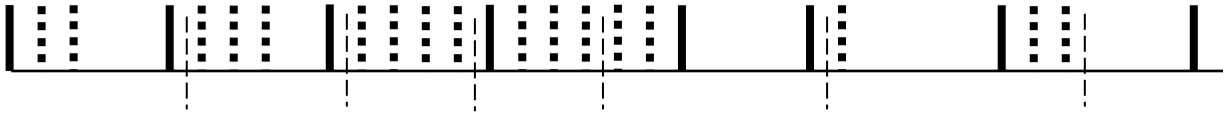


Рис. 5 (б).

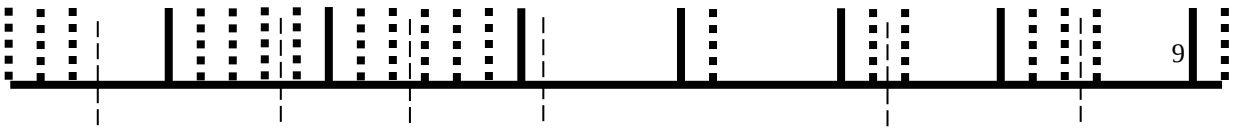


Рис. 5 (в).

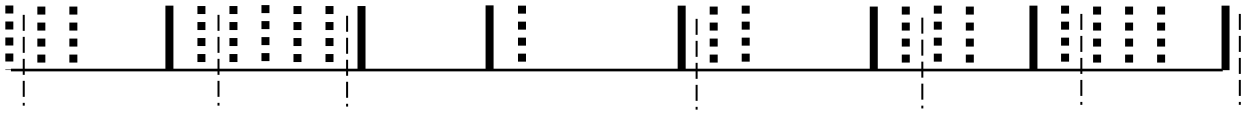


Рис. 5 (г).

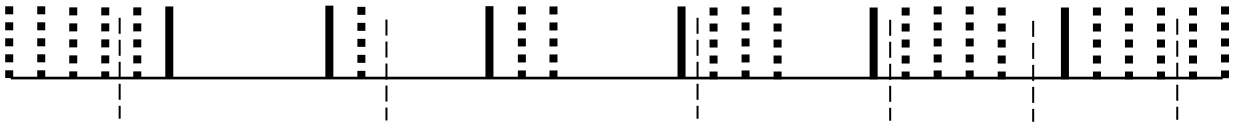


Рис. 5 (д).

Рис. 5 (а - д). Графічна динамічна модель з форматом $\{v_i; n_j*m_i;\}$, де $m_i$ – маскуючий символ (чорна пунктирна вертикальна лінія), $v_i$ – символ ВТ (чорна вертикальна лінія), $n_j$ – коефіцієнт поступово змінюється від 0 до 5 (в залежності від порядкового номеру символу ВТ $v_i$). Блоки розділені видовженими пунктирними лініями. Довжина блоку – 4 символи.

Динамічна функція вставляння маскуючих символів дає додатковий ефект. Якщо у звичайному блоковому шифрі повторення в тексті можуть виникати на відстанях, які кратні довжині ключа (число **μ** не може бути дуже велике), то у встановленні маскуючих символів після кожного символу відкритого тексту у кількості від 0 до 5 при **μ = 4** період повторення буде 84 символи, що виявлено при аналізі графічної моделі. Основні інструменти криптографа, який має задачу по статистичних характеристиках знайти ключ і використаний алгоритм шифрування, є повторення в тексті (в тому числі розривчасті) і частотна характеристика вживання символів у шифрованому тексті. Використання маскуючих символів нівелює частотну характеристику вживання символів у шифрованому тексті. А повторення з періодом 84 чи 168 мають дуже низьку імовірність формування. Враховуючи те, що маскуючі символи формуються генератором псевдовипадкових чисел, при зміні хоча би одного символу в блоці змінюється весь блок. Таким чином,

використання запропонованого способу шифрування інформації, створює передумови впровадження в практику високонадійних шифрів за рахунок використання різноманітних статичних і динамічних моделей встановлення маскуючих символів.

## 3. ЗАВДАННЯ

1. Ознайомитися із методами шифрування, описаними у коротких теоретичних відомостях.
2. Розробити програми для шифрування і дешифрування текстів шифром Хілла із використанням маскуючи символів із врахуванням, що:

– мова програмування – за вибором студента;

– відкритий текст (ВТ) для шифрування – довільний (за вибором студента);

– для шифрування ВТ з допомогою шифру Хілла використовувати ключ, який формується з латинських букв (ключ повинен відповідати прізвищу студента). Довжина блоку для шифрування 3 символи. Довжина ключа - 9 символів (матриця 3х3). Якщо прізвище складається з меншої кількості букв, то використати перші літери імені студента. Якщо матриця ключ не має оберненої, то необхідно змінити декілька букв в матриці-ключі;

– множення матриці ключа на матрицю ВТ виконувати у вигляді числових еквівалентних матриць. Обчислити обернену матрицю. Повне виведення оберненої матриці привести в звіті і виконати перевірку $A*A^{-1} = 1$, де 1 – це одинична матриця;

– варіант стосовно структури і методу використання маскувальних символів обирається відповідно до таблиці:

| **Порядковий номер у списку групи** | **Розмір блоку μ для шифрування** | **Спосіб додавання маскуючи символів** | **Структура додавання маскуючи символів** |
|---|---|---|---|
| 1. | 3 | статичний | $\{v_i; v_i; m_i\}$ |
| 2. | 3 | динамічний | $N = (2* z_i) \bmod \boldsymbol{\mu}$, **де** $N$ – кількість маскуючих символів, що вставляються після $z_i$ – го символу відкритого тексту |

| | | | |
|---|---|---|---|
| 3. | 3 | статичний | $\{v_i; m_i; v_i\}$ |
| 4. | 3 | динамічний | $N = (1+2* z_i) \bmod \mu$ |
| 5. | 3 | статичний | $\{m_i; v_i; v_i\}$ |
| 6. | 3 | динамічний | $N = (z^2+ z_i) \bmod \mu$ |
| 7. | 3 | статичний | $\{ m_i; m_i; v_i\}$ |
| 8. | 3 | динамічний | $N = (1+z^2) \bmod \mu$ |
| 9. | 3 | статичний | $\{v_i; m_i; m_i\}$ |
| 10. | 3 | динамічний | $N = (2+ z_i) \bmod \mu$ |

3. Для відкритого і шифрованого тексту виконати повний статистичний аналіз і привести графіки у звіті. На одному графіку побудувати характеристики з однаковим масштабом за спаданням частоти повторення кожного символу. Зробити висновки за результатами статистичного аналізу ШТ і ВТ.

4. Оформити та захистити звіт. У звіті обов’язково навести усі програми, які використані для шифрування та дешифрування, а також побудовані графіки. Зробити висновки за результатами статистичного аналізу ШТ і ВТ. Оцінити складність процедури шифрування і дешифрування.

## Лабораторна робота № 8

# ШИФРУВАННЯ ДАНИХ ЗА ДОПОМОГОЮ AES

## 1. МЕТА РОБОТИ

Мета роботи – ознайомитись з одним із сучасних симетричних алгоритмів блочного шифрування AES, навчитися його застосовувати для різних типів даних.

## 2. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Advanced Encryption Standard (AES) – симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128/192/256 біт), фіналіст конкурсу AES і прийнятий в якості американського стандарту шифрування урядом США. Станом на 2009 рік AES був одним із найпоширеніших алгоритмів симетричного шифрування.

**Опис алгоритму:**

AES являє собою симетричний алгоритм шифрування з ключем. Ключ може мати довжину 128, 192 і 256 біт. Алгоритми з ключем довжиною 128, 192 і 256 біт позначаються відповідно як AES-128, AES-192, AES-256.

AES-128, AES-192, AES-256 обробляють блоки даних за відповідно 10, 12 та 14 ітерацій. Кожна ітерація (раунд) являє собою визначену послідовність трансформацій. Всі ітерації однакові за виключенням останньої, з якої виключене одне з перетворень.

Кожен раунд працює з двома 128-бітними блоками: “поточний” та “ключ раунду”. Всі раунди використовують різні “ключі раунду”, які отримуються за допомогою алгоритму розширення ключа. Цей алгоритм не залежить від даних, які ми шифруємо, і може виконуватися незалежно від фази шифрування/дешифрування.

Блок даних послідовно проходить через наступні стадії:

– Над блоком виконується операція XOR з першими 128 бітами ключа, на виході отримуємо “поточний” блок (ця стадія також називається нульовим раундом, з використанням нульового ключа раунду).

– Тоді поточний блок проходить через 10/12/14 раундів шифрування, після яких він перетворюється в шифрований (або дешифрований) блок.

В кожному раунді алгоритму виконуються перетворення, зображені на рисунку 6:

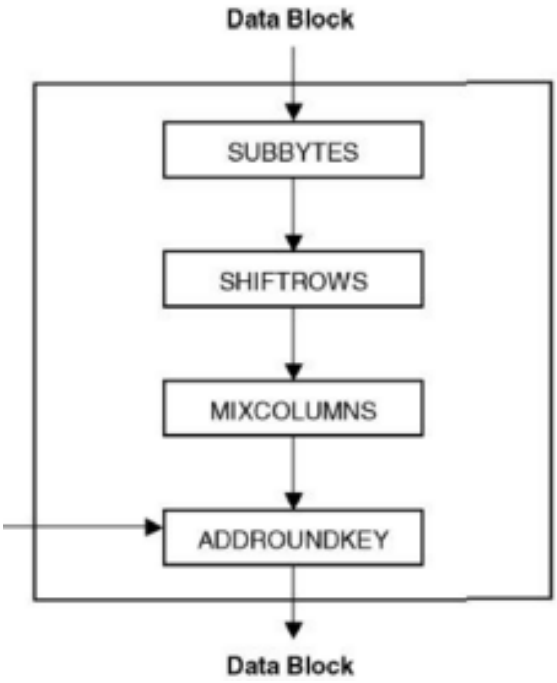



Рис.6. Раунд алгоритму

Операція SubBytes – являє собою табличну заміну кожного байта блоку відповідно до табл.1 (рис. 7). Вхідне значення 0 замінюється на 63 (шістнадцяткова система), 1 – на 7C, 2 – на 77 і т.д.

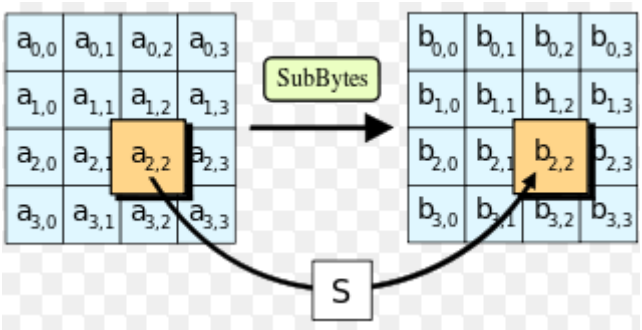



Рис.7 Операція SubBytes

Таблиця 1.

| | | y | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| x | 0 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| | 1 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| | 2 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| | 3 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| | 4 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| | 5 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| | 6 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| | 7 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| | 8 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| | 9 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| | a | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| | b | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| | c | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| | d | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| | e | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| | f | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

Операція ShiftRows, яка виконує циклічний зсув вліво усіх рядків блоку даних, за винятком нульового (рис. 8). Зсув N-го рядка масиву (для i = 1,2,3) виконується на N байтів.

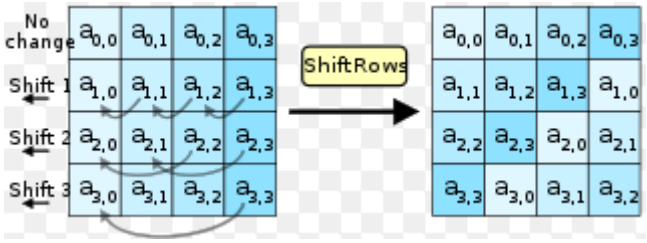



Рис.8. Операція ShiftRows

Операція MixColumns виконує множення кожного стовпця блоку даних (рис.9), який розглядається, як поліном в кінцевому полі GF($2^8$), на фіксований поліном $c(x) = 3x^3 + x^2 + x + 2$. Множення виконується за модулем $x^4 + 1$.

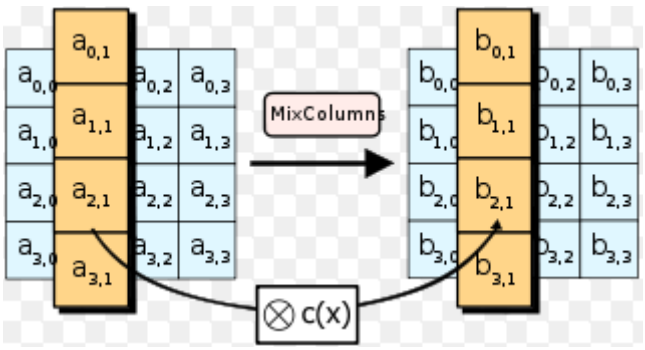



Рис.9. Операція MixColumns

Операція AddRoundKey виконує накладання на блок даних матеріалу ключа. А саме, на N-й стовбець блоку даних ( N = 0…3) за допомогою XOR накладається певне слово розширеного ключа W4r + i, де r – номер поточного раунду алгоритму, починаючи з 1 (процедура розширення ключа буде описана далі). Операція AddRoundKey представлена на рис.10.

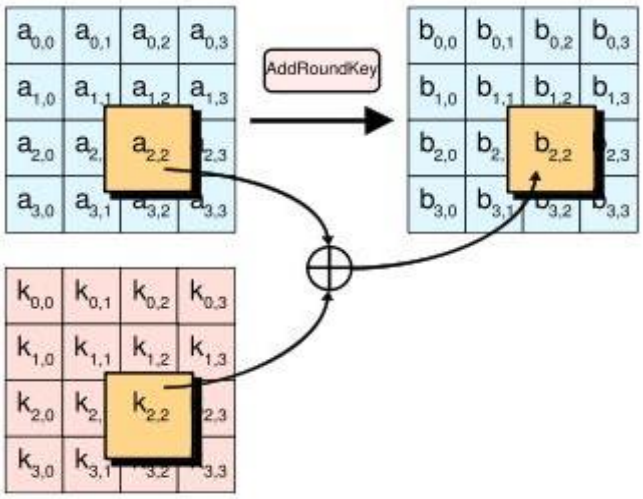



Рис.10. Операція AddRoundKey

Перед першим раундом алгоритму виконується попереднє накладання матеріалу ключа за допомогою операції AddRoundKey, яка накладає перші чотири слова розширеного ключа $1У_0…1У_3$ на відкритий текст. Останній раунд відрізняється від попередніх тим, що в ньому не виконується операція MixColumns.

**Процедура розширення ключа**

Завдання процедури розширення ключа полягає в формуванні потрібної кількості слів ключа для їх використання в операції AddRoundKey. Під «словом» розуміється 4-байтний фрагмент ключа, один з яких використовується в первинному накладанні матеріалу ключа і по одному в кожному раунді алгоритму. Таким чином в процесі розширення ключа формується 4*(N + 1) слів.

## 3. ЗАВДАННЯ

1. Розробити графічний інтерфейс на мові C# для коду алгоритму реалізації шифрування файлів за допомогою AES (наведений у додатку).

2. Налаштувати алгоритм згідно свого варіанту відповідно до таблиці варіантів.

| **Варіант** | **Вектор ініціалізації** | **Режим** | **Довжина ключа** | **Ключ*** |
|---|---|---|---|---|
| 1 | Ім’я та Прізвище у форматі: [NameSurname]<br><br>Довжина вектора не менше і не більше 16 символів, при недостачі продублювати літери. | CBC | 128 | Номер телефону + номер студентського |
| 2 | | CFB | 192 | Номер студентського + номер телефону |
| 3 | | CTS | 256 | Ввести з клавіатури |
| 4 | | ECB | 128 | Дата народження + прізвище + ім’я |
| 5 | | OFB | 192 | Прізвище + ім’я + по батькові |
| 6 | | OFB | 256 | Номер телефону + номер студентського |
| 7 | | CTS | 128 | Номер студентського + номер телефону |
| 8 | | ECB | 192 | Ввести з клавіатури |
| 9 | | CBC | 256 | Дата народження + прізвище + ім’я |
| 10 | | CFB | 128 | Прізвище + ім’я + по батькові |

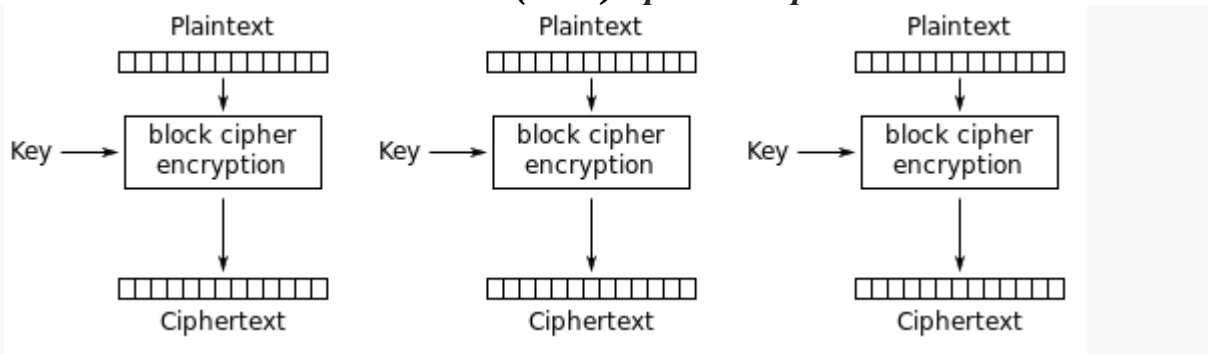
*** - отримане значення потрібно продублювати до розміру свого ключа**

**Вектор ініціалізації (IV)** – являє собою деяке число, як правило воно має бути випадковим, або псевдовипадковим. Випадковість має велике значення для досягнення семантичної безпеки, яка при повторному

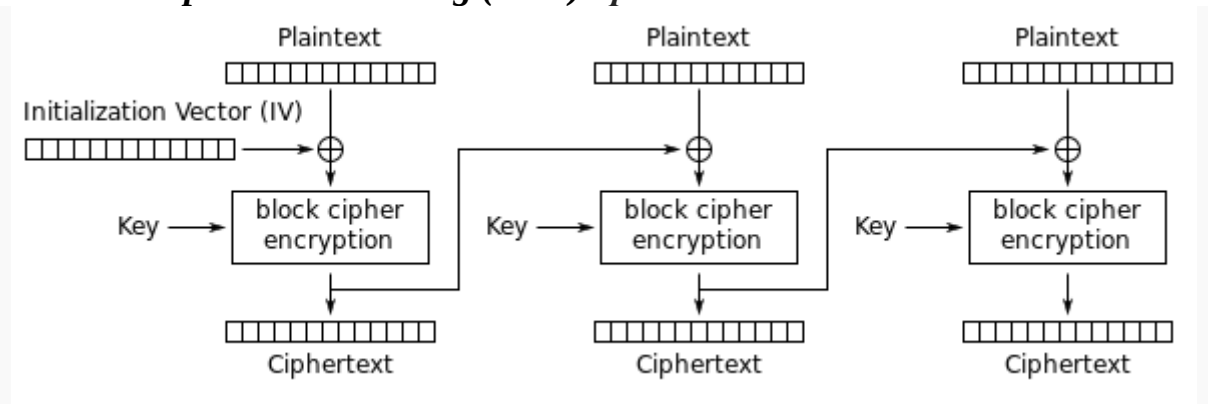
використанні схеми з тим же ключем не дасть зловмиснику вивести співвідношення між сегментами зашифрованого повідомлення.

**Режими шифрування**

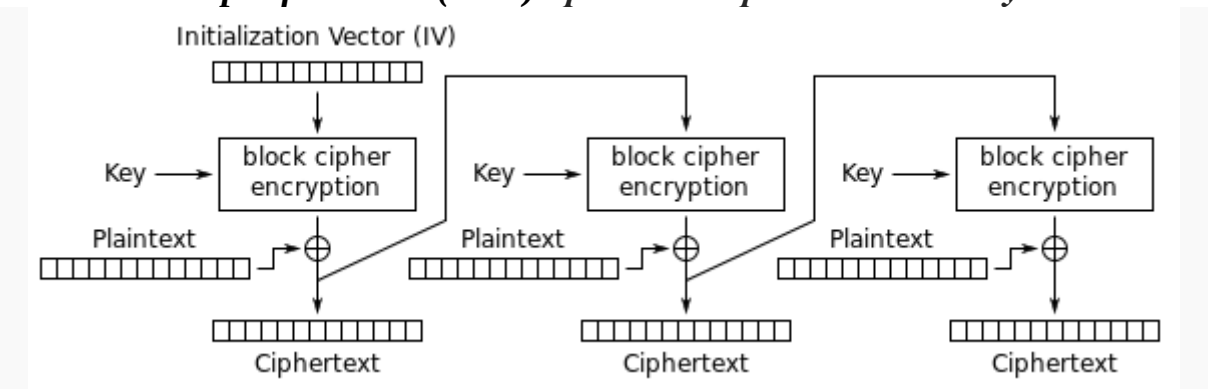
***Electronic code book (ECB) - режим простої заміни***



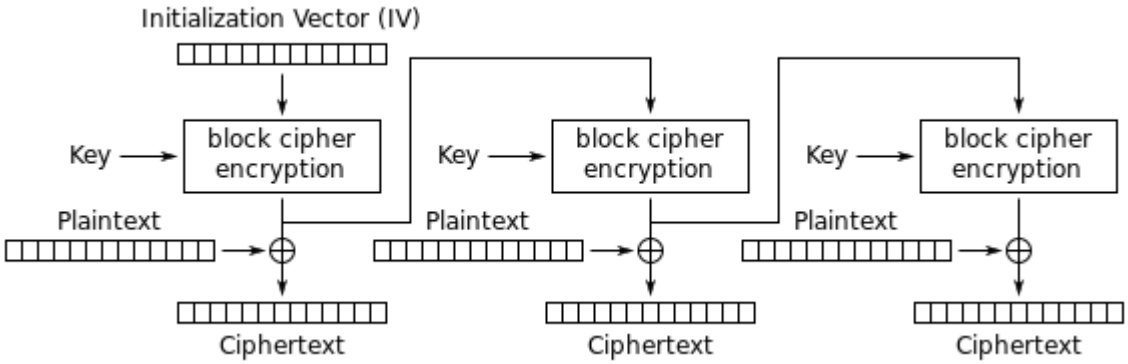

***Ciper block chaining (CBC) - режим зчіплювання блоків***



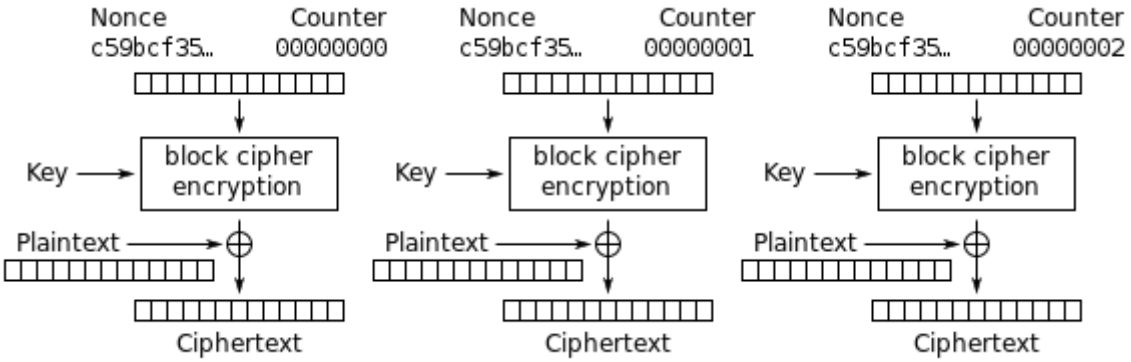

***Ciper feed back (CFB) - режим зворотнього зв'язку***




***Output feed back (OFB) - режим зворотнього зв'язку по виходу***



***Counter mode (CTR) – режим лічильника***




3. Зашифрувати файл розміром не менше 10 Мб.
4. Дешифрувати зашифрований файл.
5. Оформити та захистити звіт.

## Додаток. Код алгоритму на C# реалізації шифрування файлів за допомогою AES

```
private static SymmetricAlgorithm _alg;
private static readonly string _hash = "SHA512";
private static readonly int _keylen = [YourKeyLen];
private static readonly CipherMode _mode = CipherMode.CBC;
private static readonly string _iv = "[YourInitializationVector]";

private static void Encrypt(string infile, string key)
{
   string outfile = string.Empty;

   if (infile.Contains('\\'))
   {
      if (infile.Substring(infile.LastIndexOf('\\')).Contains('.'))
      {
         outfile = infile.Substring(0, infile.LastIndexOf('.')) + "_enc" +
                infile.Substring(infile.LastIndexOf('.'));
      }
      else
      {
         outfile = infile + "_enc";
      }
   }
   else
   {
      if (infile.Contains('.'))
      {
         outfile = infile.Substring(0, infile.LastIndexOf('.')) + "_enc" +
                infile.Substring(infile.LastIndexOf('.'));
      }
      else
      {
         outfile = infile + "_enc";
      }
   }
   _alg = (SymmetricAlgorithm)RijndaelManaged.Create();

   PasswordDeriveBytes pdb = new PasswordDeriveBytes(key, null) {HashName = _hash};
   _alg.KeySize = _keylen;
   _alg.Key = pdb.GetBytes(_keylen >> 3);
   _alg.Mode = _mode;
   _alg.IV = GetBytes(_iv);

   Console.WriteLine("Hash:              {0}", pdb.HashName);
   Console.WriteLine("Key lenght:        {0}", _alg.KeySize);
   Console.WriteLine("Encrypt key:       {0}", GetString(pdb.GetBytes(_keylen >> 3)));
   Console.WriteLine("Encrypt mode:      {0}", _alg.Mode);
   Console.WriteLine("IV:                {0}", GetString(_alg.IV));

   ICryptoTransform tr = _alg.CreateEncryptor();

   FileStream instream = new FileStream(infile, FileMode.Open, FileAccess.Read, FileShare.Read);
   FileStream outstream = new FileStream(outfile, FileMode.Create, FileAccess.Write, FileShare.None);

   int buflen = ((2 << 16) / _alg.BlockSize) * _alg.BlockSize;
   byte[] inbuf = new byte[buflen];
   byte[] outbuf = new byte[buflen];
```

```
    int len;
    while ((len = instream.Read(inbuf, 0, buflen)) == buflen)
    {
        int enclen = tr.TransformBlock(inbuf, 0, buflen, outbuf, 0);
        outstream.Write(outbuf, 0, enclen);
    }
    instream.Close();
    outbuf = tr.TransformFinalBlock(inbuf, 0, len);
    outstream.Write(outbuf, 0, outbuf.Length);
    outstream.Close();
    _alg.Clear();
}
private static void Decrypt(string infile, string key)
{
    string outfile = string.Empty;

    if (infile.Contains('\\'))
    {
        if (infile.Substring(infile.LastIndexOf('\\')).Contains('.'))
        {
            if (infile.Contains("_enc"))
                outfile = infile.Substring(0, infile.LastIndexOf('_')) + "_dec" +
                        infile.Substring(infile.LastIndexOf('.'));
            else
                outfile = infile.Substring(0, infile.LastIndexOf('.')) + "_dec" +
                        infile.Substring(infile.LastIndexOf('.'));
        }
        else
        {
            if (infile.Contains("_enc"))
                outfile = infile.Substring(0, infile.LastIndexOf('_')) + "_dec";
            else
                outfile = infile + "_dec";
        }
    }
    else
    {
        if (infile.Contains('.'))
        {
            if (infile.Contains("_enc"))
                outfile = infile.Substring(0, infile.LastIndexOf('_')) + "_dec" +
                        infile.Substring(infile.LastIndexOf('.'));
            else
                outfile = infile.Substring(0, infile.LastIndexOf('.')) + "_dec" +
                        infile.Substring(infile.LastIndexOf('.'));
        }
        else
        {
            if (infile.Contains("_enc"))
                outfile = infile.Substring(0, infile.LastIndexOf('_')) + "_dec";
            else
                outfile = infile + "_dec";
        }
    }

    _alg = (SymmetricAlgorithm)RijndaelManaged.Create();

    PasswordDeriveBytes pdb = new PasswordDeriveBytes(key, null) {HashName = _hash};
    _alg.KeySize = _keylen;
```

```
    _alg.Key = pdb.GetBytes(_keylen >> 3);
    _alg.Mode = _mode;
    _alg.IV = GetBytes(_iv);

    Console.WriteLine("Hash:               {0}", pdb.HashName);
    Console.WriteLine("Key lenght:         {0}", _alg.KeySize);
    Console.WriteLine("Encrypt key:        {0}", GetString(pdb.GetBytes(_keylen >> 3)));
    Console.WriteLine("Encrypt mode:       {0}", _alg.Mode);
    Console.WriteLine("IV:                 {0}", GetString(_alg.IV));

    ICryptoTransform tr = _alg.CreateDecryptor();

    FileStream instream = new FileStream(infile, FileMode.Open, FileAccess.Read, FileShare.Read);
    FileStream outstream = new FileStream(outfile, FileMode.Create, FileAccess.Write, FileShare.None);

    int buflen = ((2 << 16) / _alg.BlockSize) * _alg.BlockSize;
    byte[] inbuf = new byte[buflen];
    byte[] outbuf = new byte[buflen];
    int len;
    while ((len = instream.Read(inbuf, 0, buflen)) == buflen)
    {
       int enclen = tr.TransformBlock(inbuf, 0, buflen, outbuf, 0);
       outstream.Write(outbuf, 0, enclen);
    }
    instream.Close();
    outbuf = tr.TransformFinalBlock(inbuf, 0, len);
    outstream.Write(outbuf, 0, outbuf.Length);
    outstream.Close();
    _alg.Clear();
}

private static string GetString(byte[] bytes)
{
    char[] chars = System.Text.Encoding.UTF8.GetChars(bytes);
    return new string(chars);
}
private static byte[] GetBytes(string str)
{
    byte[] bytes = System.Text.Encoding.UTF8.GetBytes(str);
    return bytes;
}
```

# ЛІТЕРАТУРА


1. Menezes A., van Oorshot P., Vanstone S. Handbook of applied cryptography. CRC Press, 1997.
2. W. Stallings. Computer security: principles and practice / William Stallings, Lawrie Brown.—2nd ed. – Pearson. – 2012. – 817 p.
3. W. Stallings. Cryptography and network security. Principles and practice / William Stallings. – Pearson. – 2011. – 744 p.
4. Ignatowych A. Effectiveness evaluation of modified block ciphers using standardized NIST statistical tests / Ignatowych A. // 5th Interntional Youth Science Forum "LITERIS ET ARTIBUS" – Lviv Polytechnic Publishing House. – Lviv, Ukraine. - 26 – 28.11.2015. – P. 76 – 79.
5. Ігнатович А.О. Методи шифрування інформації із використанням маскуючи символів / Ігнатович А.О., Парамуд Я.С. // Вісник Національного університету "Львівська політехніка". Збірник наукових праць. Серія "Комп'ютерні науки та інформаційні технології". – 2015, № 826. - С. 21 – 27.
6. Мельник А.О., Ємець В.Ф., Попович Р. Сучасна криптографія. Основні поняття. Львів, БаК, 2003. – 144 с.
7. Спосіб шифрування інформації. Патент України на корисну модель №99073. Бюл. № 9 від 12.05.2015. Ігнатович А.О., Іванців В. Р., Іванців Р-А. Д., Павич Н. Я.
8. Т. Коркішко, А. Мельник, В. Мельник. Алгоритми та процесори симетричного блокового шифрування – Львів, БаК, 2003.
9. Яковина В.С., Федасюк Д.В. Основи безпеки комп'ютерних мереж. Львів, Українські технології, 2008.