Міністерство освіти і науки України

Університет митної справи та фінансів

Факультет інноваційних технологій

Кафедра інформаційних систем та технологій

# КУРСОВА РОБОТА

з дисципліни «Технології захисту інформації»

Студента \_\_3\_\_ курсу, групи К17-1,

напряму підготовки Комп’ютерні науки,

Горбуля І.В.

Керівник \_Яковенко В.О.\_

м. Дніпро

2019 рік

**ЗМІСТ**

[Введення..................................................................................................3](#_Toc388095334)

1. [Основні відомості..........................................................................](#_Toc388095335)3-12

[1.1 Постановка завдання](#_Toc388095336).....................................................................3

[1.2](#_Toc388095337) Опис алгориритму......................................................................3-6

1.3Обчислення ключа раунду.......................................................6-10

1.4S-блок........................................................................................8-10

1.5 Перетворення SHIFTROW.....................................................10-11

1.6 Перетворення MIXCOLUMN................................................11-12

1.7 Додавання з ключем раунду........................................................12

2 . [Розробка програми.........................................................................](#_Toc388095338)..13

2.1 Інтерфейс програми....................................................................13

2.2 Написання коду......................................................................14-15

3. Лістинг програми.........................................................................15-16

4. [Скріншоти](#_Toc388095341) програми........................................................................17

4.1 Шифратор.....................................................................................17

4.2 Дешифратор..................................................................................17

Примітки................................................................................................18

[Висновки](#_Toc388095342)................................................................................................18

Джерела....................…..........................................................................18

**Введення**

C# (вимовляється Сі-шарп) — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов С++, Delphi, Модула і Smalltalk — С#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

**1. Основні відомості**

**1.1 Постановка завдання**

Розробити програмне забезпечення, що реалізує симетричний блоковий алгоритм шифрування зі змінною довгою блоку і ключа Rijndael - покращений стандарт шифрування AES.

**1.2. Опис алгоритму**

**Advanced Encryption Standard** (AES), також відомий під назвою **Rijndael** — симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128/192/256 біт), фіналіст конкурсу AES і прийнятий як американський стандарт шифрування урядом США. Вибір припав на AES з розрахуванням на широке використання і активний аналіз алгоритму, як це було із його попередником, DES. Державний інститут стандартів і технологій (англ. National Institute of Standards and Technology, NIST) США опублікував попередню специфікацію AES 26 жовтня 2001 року, після п'ятилітньої підготовки. 26 травня 2002 року AES оголошено стандартом шифрування. Станом на 2009 рік AES є одним із найпоширеніших алгоритмів симетричного шифрування.

В принципі, алгоритм, запропонований Рейменом і Дейцменом, і AES не одне і те ж. Алгоритм Рейндол підтримує широкий діапазон розміру блоку та ключа. AES має фіксовану довжину у 128 біт, а розмір ключа може приймати значення 128, 192 або 256 біт. В той час як Рейндол підтримує розмірність блоку та ключа із кроком 32 біт у діапазоні від 128 до 256. Через фіксований розмір блоку AES оперує із масивом 4×4 байт, що називається станом (версії алгоритму із більшим розміром блоку мають додаткові колонки).

Кожен блок відкритого тексту зашифровується кілька разів в так званих раундах (round) за допомогою повторюваної послідовності різних функцій. Число раундів залежить від довжини блоку і ключа (див. Таблицю 1).

|  |  |  |  |
| --- | --- | --- | --- |
| Длина ключа в битах | Длина блока (в битах) | | |
| 128 | 192 | 256 |
| 128 | 10 | 12 | 14 |
| 192 | 12 | 12 | 14 |
| 256 | 14 | 14 | 14 |

Таблиця 1. Кількість раундів в алгоритмі Rijndael як функція від довжини блоку і ключа

Rijndael не відноситься до алгоритмів на мережах Фейстеля, які характеризуються тим, що блок тексту розбивається на ліву і праву половини, потім перетворення раунду застосовується до однієї половині, результат складається по модулю 2 з іншою половиною, після чого ці половини міняються місцями. Найвідомішим блоковим алгоритмом з цієї серії є DES. Rijndael, навпаки, складається з окремих рівнів, кожен з яких по-своєму впливає на блок в цілому. Для шифрування блоку послідовно виконуються наступні перетворення:

Перший раундовий ключ складається з блоком по модулю 2 (XOR).

Виконуються Lr - 1 звичайних раундів.

|  |  |
| --- | --- |
| Подстановка  (S-блок) |  |
| ShiftRow |
| MixColumn |
| Сложение с раундовым ключом |

Рис.1. Рівні перетворення всередині одного раунду алгоритму Rijndael

Виконується завершальний раунд, в якому, на відміну від звичайного, відсутня перетворення MixColumn.

Кожен звичайний раунд на етапі 2 складається з чотирьох окремих кроків.

**Підстановка**.

Кожен байт блоку замінюється значенням, яке визначається S-блоком.

**Перестановка**.

Байти в блоці переставляються за допомогою перетворення ShiftRow.

**Перемішування**.

Виконується перетворення MixColumn.

**Додавання з раундовим ключем.**

Поточний раундовий ключ складається з блоком по модулю 2.

Кожен рівень надає на кожний з блоків відкритого тексту певним чином впливати.

**1. Вплив ключа**

Додавання тексту з ключем до першого раунду і на останньому кроці всередині кожного раунду впливає на кожен біт результату раунду. У процесі шифрування результат кожного кроку в кожному бите залежить від ключа.

**2. Нелінійний рівень**

Операція підстановки в S-блоці є нелінійної. Будова S-блоків забезпечує майже ідеальний захист від диференціального і лінійного криптоаналізу.

**3. Лінійний рівень**

Перетворення ShiftRow і MixColumn забезпечують максимальне перемішування бітів в блоці.

Далі в описі внутрішніх функцій алгоритму Rijndael, через Lb будемо позначати довжину блоку в чотирьохбайтового словах, через Lk - довжину ключа користувача в чотирьохбайтового словах (тобто Lb, Lk є {4, 6, 8}) і через Lr - число раундів (див . таблицю 1).

Відкритий і зашифрований тексти представлені у вигляді полів байтів і є відповідно входом і виходом алгоритму.

Блок відкритого тексту, що обробляється як поле m0, ..., m4Lb-1, представлений у вигляді двовимірної структури В (див. Таблицю 2). в якій байти відкритого тексту відсортовані в наступному порядку:



т.е. , где i = n mod 4 и; j = ⎣n/4⎦.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| b0,0 | b0,1 | b0,2 | b0,3 | b0,4 | … | b0,Lb-1 |
| b1,0 | b1,1 | b1,2 | b1,3 | b1,4 | … | b1,Lb-1 |
| b2,0 | b2,1 | b2,2 | b2,3 | b2,4 | … | b2,Lb-1 |
| b3,0 | b3,1 | b3,2 | b3,3 | b3,4 | … | b3,Lb-1 |

Таблиця 2.

Доступ до структури В у функціях алгоритму Rijndael здійснюється по-різному, в залежності від операції. S-блок оперує з битами, ShiftRow - з рядками (bi, 0, bi, 1, bi, 2, ..., bi, Lb-1) структури В, а функції AddRoundKey і MixColumn - з чотирьохбайтового словами, звертаючись до стовпців.

**ASCII** (Американський стандартний код для інформаційного обміну, англ. American Standard Code for Information Interchange) в обчислювальній техніці — система кодів, у якій числа від 0 до 127 включно поставлені у відповідність літерам, цифрам і символам пунктуації. Наприклад, 45 відповідає знаку переносу, а 65 — літері «А» великій. Перші 32 коди використовуються для керівних функцій, на зразок введення і стирання попереднього символу. Строго кажучи, ASCII — це семибітний код, а восьмий біт часто використовується для забезпечення відповідності чи додаткових символів. Такий 8-бітний варіант коду називають розширеним ASCII.

Система широко використовується для зберігання тексту і передачі інформації між комп'ютерами.

**Режим шифрування** - спосіб застосування блочного шифру (алгоритму), що дозволяє перетворити послідовність блоків відкритих даних в послідовність блоків зашифрованих даних. При цьому для шифрування одного блоку можуть вживатися дані другого блоку.

Зазвичай режими шифрування використовуються для зміни процесу шифрування так, щоб результат шифрування кожного блоку був унікальним незалежно від шифрованих даних і НЕ дозволяли зробити якісь висновки про їх структурі. Це обумовлено, перш за все, тим, що блокові шифри шифрують дані блоками фіксованого розміру, і тому існує потенційна можливість витоку інформації в повторюваних частинах даних, шіфруемих на одному і тому ж ключі.

**1.3 Обчислення ключа раунду**

І для шифрування, і для розшифрування потрібно згенерувати Lr раундовий ключів, сукупність яких називається розгорткою ключа (key schedule). Розгортка будується шляхом приєднання до секретного ключа користувача, рекурсивно одержуваних: чотирьохбайтового слів



Перші Lk слів  розгортки ключа - це сам секретний ключ користувача. Для Lk є {4, 6} чергове чотирьохбайтове слово ki визначається як сума по модулю 2 попереднього слова ki-1 зі словом ki-Lk. При i = 0 mod Lk перед операцією XOR потрібно застосувати функцію FLk (k, i), яка включає в себе циклічний зсув k байтів вліво (операція r (k)), підстановку S (r (k)) з використанням S-блоку алгоритму Rijndael (до цієї операції ми ще повернемося) і складання по модулю 2 з константою c (i / Lk). Підсумкове рівняння функції F таке:

FLk (k, i) = S (r (k)) ⊕ c (⎣i/Lk⎦).

Константи c (j) задаються рівністю c (j) = (rc (j), 0, 0, 0), де значення rc (j) визначаються рекурсивно як елементи поля F28:

rc (1) = 1, rc (j) = rc (j-1) х = хj-1.

Або у вигляді чисельних значень:

rc (1) = '01’, rc (j) = rc (j-1) •'02'.

Програмно значення rc (j) реалізується (j - 1) - кратним рекурсивним викликом функції xtime, з початковим значенням аргументу, що дорівнює 1 або швидше - з використанням таблиці передвичесленням (див. Таблицю 3).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| '01’ | '02' | '04' | '08' | '10' | '20' | '40' | '80' | '1B' | '36' |
| '6C | 'D8' | 'AB' | '4D' | '9A' | '2F' | '5E' | 'ВС | '63' | 'C6' |
| '97' | '35' | '6A' | 'D4' | 'B3' | 7O' | 'FA' | 'EF' | 'C5 | '91' |

Таблица 3. Константы rc (j) (в шістнадцатеричному вигляді)

Для ключів довжини 256 біт (тобто при Lk = 8) введена додаткова операція підстановки: при i = 4 mod Lk перед операцією XOR значення ki-1 замінюється на s (ki-1).

Таким чином, розгортка ключів складається з Lb (Lr + 1) чотирьохбайтового слів, включаючи і секретний ключ користувача. На кожному раунді i = 0, ..., Lr - 1 чергові Lb, чотирьохбайтового слова з kLbi по kLb (I + 1) вибираються з розгортки і використовуються в якості ключа раунду. Раундові ключі розглядаються, по аналогії з блоками відкритого тексту, як двовимірна структура (див. Таблицю 4).

Таблица 4. Представлення раундових ключів

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| k0,0 | k 0,1 | k 0,2 | k 0,3 | k 0,4 | … | k 0,Lb-1 |
| k 1,0 | k 1,1 | k 1,2 | k 1,3 | k 1,4 | … | k 1,Lb-1 |
| k 2,0 | k 2,1 | k 2,2 | k 2,3 | k 2,4 | … | k 2,Lb-1 |
| k 3,0 | k 3,1 | k 3,2 | k 3,3 | k 3,4 | … | k 3,Lb-1 |

Для ключів довжини 128 біт процес генерації ключа зображений на рис.2.

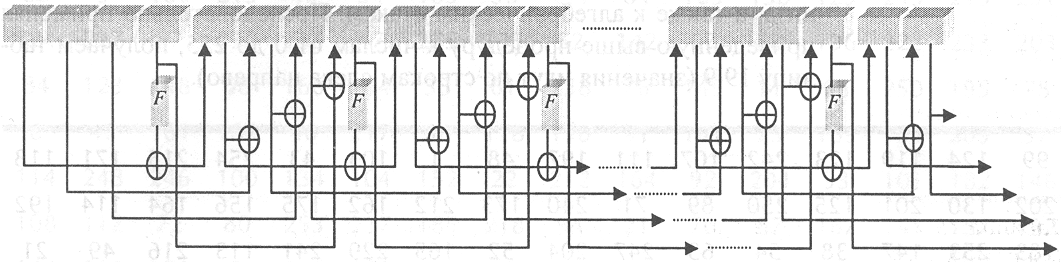


Рис.2. Діаграма раундовий ключів для Lk = 4

Поки не відомі слабкі ключі, використання яких несприятливо позначилося б на стійкості алгоритму Rijndael

**1.4 S-БЛОК**

Блок підстановки, або S-блок алгоритму Rijndael показує, яким значенням слід замінювати кожен байт блоку тексту на кожному раунді. S-блок являє собою список з 256 байтів. Спочатку кожен ненульовий байт розглядається як елемент поля F28 і замінюється мультиплікативно зворотним (нульові байти залишаються незмінними). Потім виконується наступне Афінний перетворення над полем F2 шляхом множення на матрицю і складання з вектором (11000110):



Тут через х0 і у0 позначені молодші, а через х7 і У7 - старші біти в байті; вектор (11000110) довжини 8 відповідає шістнадцятиричним числу '63'.

S-блок побудований так, щоб звести до мінімуму чутливість алгоритму до диференціального і лінійного методам криптоаналізу, а також до алгебраїчних атакам. Послідовно застосовуючи наведену вище процедуру до числам від 0 до 255, отримуємо таблицю 5 (значення йдуть по рядках зліва направо).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 99 | 124 | 119 | 123 | 242 | 107 | 111 | 197 | 48 | 1 | 103 | 43 | 254 | 215 | 171 | 118 |
| 202 | 130 | 201 | 125 | 250 | 89 | 71 | 240 | 173 | 212 | 162 | 175 | 156 | 164 | 114 | 192 |
| 183 | 253 | 147 | 38 | 54 | 63 | 247 | 204 | 52 | 165 | 229 | 241 | 113 | 216 | 49 | 21 |
| 4 | 199 | 35 | 195 | 24 | 150 | 5 | 154 | 7 | 18 | 128 | 226 | 235 | 39 | 178 | 117 |
| 9 | 131 | 44 | 26 | 27 | 110 | 90 | 160 | 82 | 59 | 214 | 179 | 41 | 227 | 47 | 132 |
| 83 | 209 | 0 | 237 | 32 | 252 | 177 | 91 | 106 | 203 | 190 | 57 | 74 | 76 | 88 | 207 |
| 208 | 239 | 170 | 251 | 67 | 77 | 51 | 133 | 69 | 249 | 2 | 127 | 80 | 60 | 159 | 168 |
| 81 | 163 | 64 | 143 | 146 | 157 | 56 | 245 | 188 | 182 | 218 | 33 | 16 | 255 | 243 | 210 |
| 205 | 12 | 19 | 236 | 95 | 151 | 68 | 23 | 196 | 167 | 126 | 61 | 100 | 93 | 25 | 115 |
| 96 | 129 | 79 | 220 | 34 | 42 | 144 | 136 | 70 | 238 | 184 | 20 | 222 | 94 | 11 | 219 |
| 224 | 50 | 58 | 10 | 73 | 6 | 36 | 92 | 194 | 211 | 172 | 98 | 145 | 149 | 228 | 121 |
| 231 | 200 | 55 | 109 | 141 | 213 | 78 | 169 | 108 | 86 | 244 | 234 | 101 | 122 | 174 | 8 |
| 186 | 120 | 37 | 46 | 28 | 166 | 180 | 198 | 232 | 221 | 116 | 31 | 75 | 189 | 139 | 138 |
| 112 | 62 | 181 | 102 | 72 | 3 | 246 | 14 | 97 | 53 | 87 | 185 | 134 | 193 | 29 | 158 |
| 225 | 248 | 152 | 17 | 105 | 217 | 142 | 148 | 155 | 30 | 135 | 233 | 206 | 85 | 40 | 223 |
| 140 | 161 | 137 | 13 | 191 | 230 | 66 | 104 | 65 | 153 | 45 | 15 | 176 | 84 | 187 | 22 |

Таблиця 5. Значення S-блоку

При розшифрування порядок дій змінюється на протилежний. Спочатку виконується зворотне Афінний перетворення, потім мультиплікативне звернення в поле F28. Зворотний S-блок наведено в таблиці 6.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 82 | 9 | 106 | 213 | 48 | 54 | 165 | 56 | 191 | 64 | 163 | 158 | 129 | 243 | 215 | 251 |
| 124 | 227 | 57 | 130 | 155 | 47 | 255 | 135 | 52 | 142 | 67 | 68 | 196 | 222 | 233 | 203 |
| 84 | 123 | 148 | 50 | 166 | 194 | 35 | 61 | 238 | 76 | 149 | 11 | 66 | 250 | 195 | 78 |
| 8 | 46 | 161 | 102 | 40 | 217 | 36 | 178 | 118 | 91 | 162 | 73 | 109 | 139 | 209 | 37 |
| 114 | 248 | 246 | 100 | 134 | 104 | 152 | 22 | 212 | 164 | 92 | 204 | 93 | 101 | 182 | 146 |
| 108 | 112 | 72 | 80 | 253 | 237 | 185 | 218 | 94 | 21 | 70 | 87 | 167 | 141 | 157 | 132 |
| 144 | 216 | 171 | 0 | 140 | 188 | 211 | 10 | 247 | 228 | 88 | 5 | 184 | 179 | 69 | 6 |
| 208 | 44 | 30 | 143 | 202 | 63 | 15 | 2 | 193 | 175 | 189 | 3 | 1 | 19 | 138 | 107 |
| 58 | 145 | 17 | 65 | 79 | 103 | 220 | 234 | 151 | 242 | 207 | 206 | 240 | 180 | 230 | 115 |
| 150 | 172 | 116 | 34 | 231 | 173 | 53 | 133 | 226 | 249 | 55 | 232 | 28 | 117 | 223 | 110 |
| 71 | 241 | 26 | 113 | 29 | 41 | 197 | 137 | 111 | 183 | 98 | 14 | 170 | 24 | 190 | 27 |
| 252 | 86 | 62 | 75 | 198 | 210 | 121 | 32 | 154 | 219 | 192 | 254 | 120 | 205 | 90 | 244 |
| 31 | 221 | 168 | 51 | 136 | 7 | 199 | 49 | 177 | 18 | 16 | 89 | 39 | 128 | 236 | 95 |
| 96 | 81 | 127 | 169 | 25 | 181 | 74 | 13 | 45 | 229 | 122 | 159 | 147 | 201 | 156 | 239 |
| 160 | 224 | 59 | 77 | 174 | 42 | 245 | 176 | 200 | 235 | 187 | 60 | 131 | 83 | 153 | 97 |
| 23 | 43 | 4 | 126 | 186 | 119 | 214 | 38 | 225 | 105 | 20 | 99 | 85 | 33 | 12 | 125 |

Таблиця 6. Значення зворотного S-блоку

**1.5 Перетворення SHIFTROW**

Наступний крок раунду - перестановка байтів в блоці. Порядок байтів змінюється в рядку (bi, 0, bi, 1, bi, 2, ..., bi, Lb-1) структури У відповідно до таблиць 7-9.

Операція ShiftRow для блоків довжини 128 біт (Lb = 4)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | До операции ShiftRow | | | | | 0 | 4 | 8 | 12 | | 1 | 5 | 9 | 13 | | 2 | 6 | 10 | 14 | | 3 | 7 | 11 | 15 | | |  |  |  |  | | --- | --- | --- | --- | | После операции ShiftRow | | | | | 0 | 4 | 8 | 12 | | 5 | 9 | 13 | 1 | | 10 | 14 | 2 | 6 | | 15 | 3 | 7 | 11 | |

Операція ShiftRow для блоків довжини 192 біт (Lb = 6)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | До операции ShiftRow | | | | | | | 0 | 4 | 8 | 12 | 16 | 20 | | 1 | 5 | 9 | 13 | 17 | 21 | | 2 | 6 | 10 | 14 | 18 | 22 | | 3 | 7 | 11 | 15 | 19 | 23 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | После операции ShiftRow | | | | | | | 0 | 4 | 8 | 12 | 16 | 20 | | 5 | 9 | 13 | 17 | 21 | 1 | | 10 | 14 | 18 | 22 | 2 | 6 | | 15 | 19 | 23 | 3 | 7 | 11 | |

Операція ShiftRow для блоків довжини 256 біт (Lb = 8)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | До операции ShiftRow | | | | | | | | | 0 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | | 1 | 5 | 9 | 13 | 17 | 21 | 25 | 29 | | 2 | 6 | 10 | 14 | 18 | 22 | 26 | 30 | | 3 | 7 | 11 | 15 | 19 | 23 | 27 | 31 | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | После операции ShiftRow | | | | | | | | | 0 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | | 5 | 9 | 13 | 17 | 21 | 25 | 29 | 1 | | 14 | 18 | 22 | 26 | 30 | 2 | 6 | 10 | | 19 | 23 | 27 | 31 | 3 | 7 | 11 | 15 | |

Всі нульові рядки залишаються без змін. У рядках i = 1,2,3 байти циклічно зсуваються вліво на cLb, i позицій: з позиції з номером j на позицію з номером j - cLb, i mod Lb, де значення cLb, i визначається за таблицею 10.

|  |  |  |  |
| --- | --- | --- | --- |
| Lb | cLb,1 | cLb,2 | cLb,3 |
| 4 | 1 | 2 | 3 |
| 6 | 1 | 2 | 3 |
| 8 | 1 | 3 | 4 |

Таблиця 10. Розмір зсуву рядків в операції ShiftRow

При зворотному перетворенні позиція з номером j в рядках i = 1,2,3 зсувається на позицію з номером j + cLb, i mod Lb.

**1.5 Перетворення MIXCOLUMN**

Після того як виконана остання порядкова перестановка, на наступному кроці кожен стовпець (bi, j) блоку тексту, де i = 0, ..., 3, j = 0, ..., Lb, представляється у вигляді полінома над полем F28 і множиться на фіксований поліном a (x) = а3x3 + а2x2 + а1x + a0 з коефіцієнтами a0 (x) = x, a1 (x) = 1, a2 (x) = 1, a3 (x) = х + 1. Потім обчислюється залишок від ділення отриманого добутку на модуль М (х) = х4 + 1. Таким чином, кожен байт стовпчика взаємодіє з усіма іншими байтами стовпчика. При построковом перетворенні ShiftRow на кожному раунді байти взаємодіють один з одним в інших комбінаціях. Тобто ці дві операції дають сильне перемішування.

Цей крок можна звести до множення на матрицю:



Множення на '02' (відповідно на х) ми вже реалізували у вигляді функції xtime (див. Таблицю 3). Множення на '03' (відповідно на х + 1) теж зроблено за аналогією.

Для звернення перетворення MixColumn множимо кожен стовпець (bi, j) блоку тексту на поліном г (х) = r3х3 + r2х2 + r1х + r0 з коефіцієнтами r0 (х) = х3 + х2 + х, r1 (х) = х3 + 1, r2 (х) = х3 + х2 + 1, r3 (х) = х3 + х + 1 і наводимо результат по модулю М (х) = х4 + 1. Відповідна матриця має вигляд:



**1.6 Додавання з ключем раунду**

На останньому кроці циклу раундовий ключ складається по модулю 2 з блоком тексту:



**2. Розробка програми**

**2.1. Інтерфейс програми**

Я використовував середу розробки Microsoft Visual Studio C# -> Windows Forms.

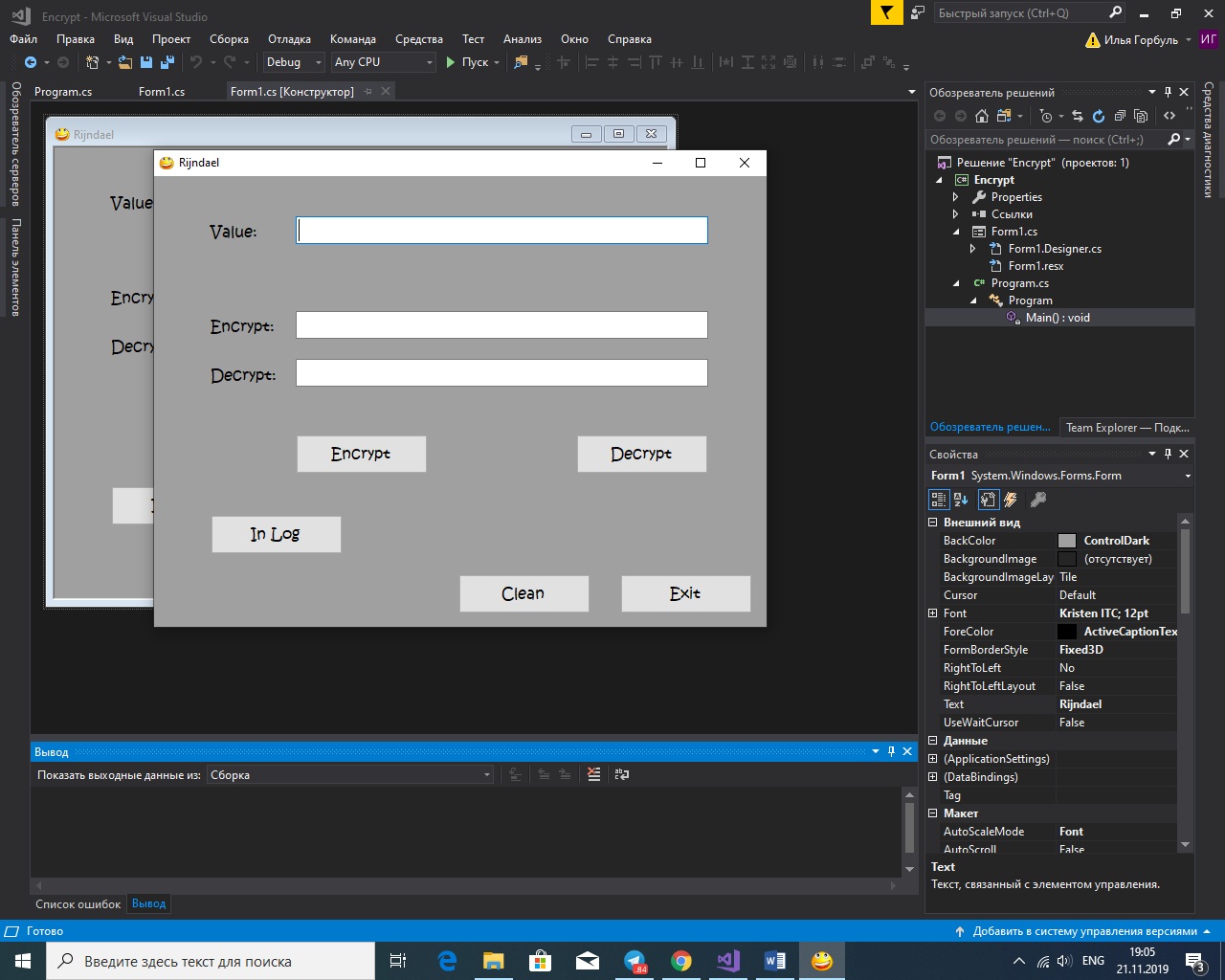
Спочатку побудував простий інтерфейс, котрий дозволяє вводити строку, а потім шифрувати та дешифрувати її. Форма складається з таких елементів:

* три Label;
* три TextBox;
* чотири Button.

У елементах Label містяться назви полей. У елементі txtValue ми можемо вводити строку, а потім, при натисканні на елементи btnEncrypt та btnDecrypt, шифрувати та дешифрувати її, транслюючи результати у txtEncrypt та txtDecrypt. Елементи Button відповідають за обробку процесів, таких як шифрування (btnEncrypt), дешифрування (btnDecrypt), видалення даних з TextBox-ів (btnClean) та вихід з програми (btnExit).

Також я зробив декілька косметичних перевтілень, таких як:

* змінив іконку програми;
* змінив вид Forms на 3D Forms:
* змінив шрифт в програмі на Kristen INC;



**2.2. Написання коду**

Використовую цільову платформу .NET Framework 3.5

Спочтаку обов’зяково треба підключити бібліотеку using System.Security.Cryptography; для подальної роботи з шифруванням.

Після цього ми оголошуємо декілька змінних для подальшого використання:

string cipherData

byte[] cipherbytes;

byte[] plainbytes;

byte[] plainbytes2;

byte[] plainKey;

Також називаємо симетричний алгоритм desObj:

SymmetricAlgorithm desObj;

Та вказуємо, що це алгоритм Rijndael:

desObj = Rijndael.Create();

Обробка натискання клавіші шифрування:

cipherData = txtValue.Text; // Передаємо тип String у txtValue

plainbytes = Encoding.ASCII.GetBytes(cipherData); // Вказуємо систему кодів ASCII

plainKey = Encoding.ASCII.GetBytes("0123456789abcdef"); // Вказуємо символи, якими будемо шифрувати. Повинно бути кратне 8

desObj.Key = plainKey; // Вказуємо, що ключ повинен зберігатися у desObj.Key

desObj.Mode = CipherMode.CBC; // Використовуємо режим зчеплення блоків шифротекста

desObj.Padding = PaddingMode.PKCS7; // Додаємо нічого не значущих символів для покращення криптостійкості

System.IO.MemoryStream ms = new System.IO.MemoryStream(); // створюємо поток, для тимчасового збереження інформації в масивах byte[] в оперативній пам’яті.

CryptoStream cs = new CryptoStream(ms, desObj.CreateEncryptor(), CryptoStreamMode.Write); // створюємо екземпляр потоку для створення шифратора.

cs.Write(plainbytes, 0, plainbytes.Length); // записуємо в цей екземпляр потоку усю інформацію

cs.Close(); // закриваємо екземпляр

cipherbytes = ms.ToArray(); // передаємо усю інформацію з потоку до масиву cipherbytes

ms.Close(); // та закриваємо поток

txtEncrypt.Text = Encoding.ASCII.GetString(chipherbytes); // отримаємо в цей TextBox інформацію з масиву chipherbytes

Обробка натискання клавіші дешифрування:

System.IO.MemoryStream ms1 = new System.IO.MemoryStream(chipherbytes);

CryptoStream cs1 = new CryptoStream(ms1, desObj.CreateDecryptor(), CryptoStreamMode.Read);

cs1.Read(chipherbytes, 0, chipherbytes.Length);

plainbytes2 = ms1.ToArray();

cs1.Close();

ms1.Close();

txtDecrypt.Text = Encoding.ASCII.GetString(plainbytes2);

Різниця полягає в тому, що в єкземплярі потоку ми повинні не створити, а зчитати шифр. А також ми записуємо інформацію в інші змінні.

Обробка натискання клавіші видалення даних:

txtValue.Clear();

txtEncrypt.Clear();

txtDecrypt.Clear();

Обробка натискання клавіші виходу:

Application.Exit();

Обробка натискання клавіші внесення у файл

private void btnLog\_Click(object sender, EventArgs e)

{

using (StreamWriter Writer = new StreamWriter(@"L:\Encrypt\Encrypt\Log.txt"))

{

Writer.WriteLine(txtEncrypt.Text);

Writer.WriteLine(txtDecrypt.Text);

}

}

**3. Лістинг програми**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Security.Cryptography;

namespace Encrypt

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

desObj = Rijndael.Create();

}

string chipherData;

byte[] chipherbytes;

byte[] plainbytes;

byte[] plainbytes2;

byte[] plainKey;

SymmetricAlgorithm desObj;

private void btnEncrypt\_Click(object sender, EventArgs e)

{

chipherData = txtValue.Text;

plainbytes = Encoding.ASCII.GetBytes(chipherData);

plainKey = Encoding.ASCII.GetBytes("0123456789abcdef");

desObj.Key = plainKey;

desObj.Mode = CipherMode.CBC;

desObj.Padding = PaddingMode.PKCS7;

System.IO.MemoryStream ms = new System.IO.MemoryStream();

CryptoStream cs = new CryptoStream(ms, desObj.CreateEncryptor(), CryptoStreamMode.Write);

cs.Write(plainbytes, 0, plainbytes.Length);

cs.Close();

chipherbytes = ms.ToArray();

ms.Close();

txtEncrypt.Text = Encoding.ASCII.GetString(chipherbytes);

}

private void btnDecrypt\_Click(object sender, EventArgs e)

{

System.IO.MemoryStream ms1 = new System.IO.MemoryStream(chipherbytes);

CryptoStream cs1 = new CryptoStream(ms1, desObj.CreateDecryptor(), CryptoStreamMode.Read);

cs1.Read(chipherbytes, 0, chipherbytes.Length);

plainbytes2 = ms1.ToArray();

cs1.Close();

ms1.Close();

txtDecrypt.Text = Encoding.ASCII.GetString(plainbytes2);

}

private void btnClean\_Click(object sender, EventArgs e)

{

txtValue.Clear();

txtEncrypt.Clear();

txtDecrypt.Clear();

}

private void btnExit\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void btnLog\_Click(object sender, EventArgs e)

{

using (StreamWriter Writer = new StreamWriter(@"L:\Encrypt\Encrypt\Log.txt"))

{

Writer.WriteLine(txtEncrypt.Text);

Writer.WriteLine(txtDecrypt.Text);

}

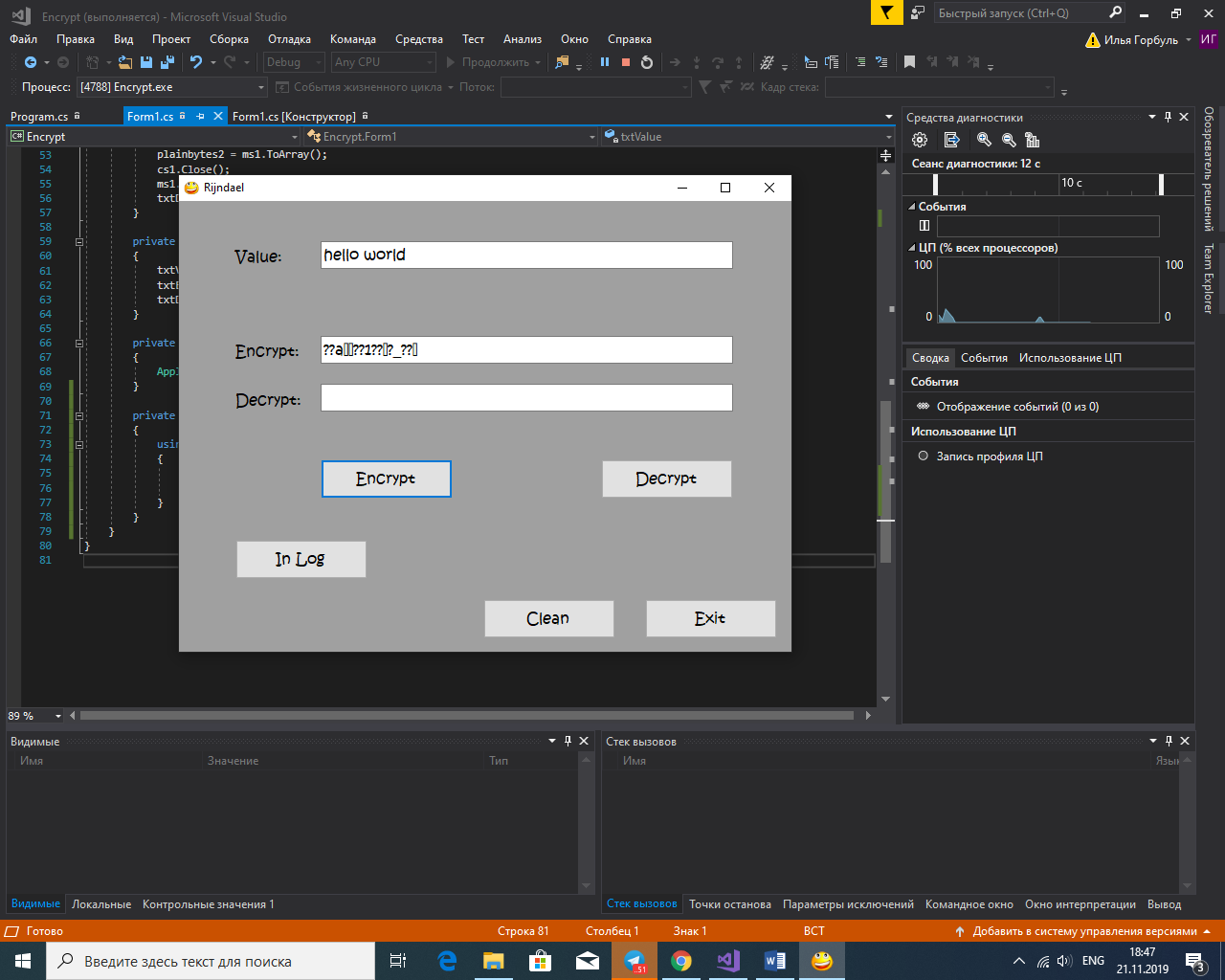
}

}

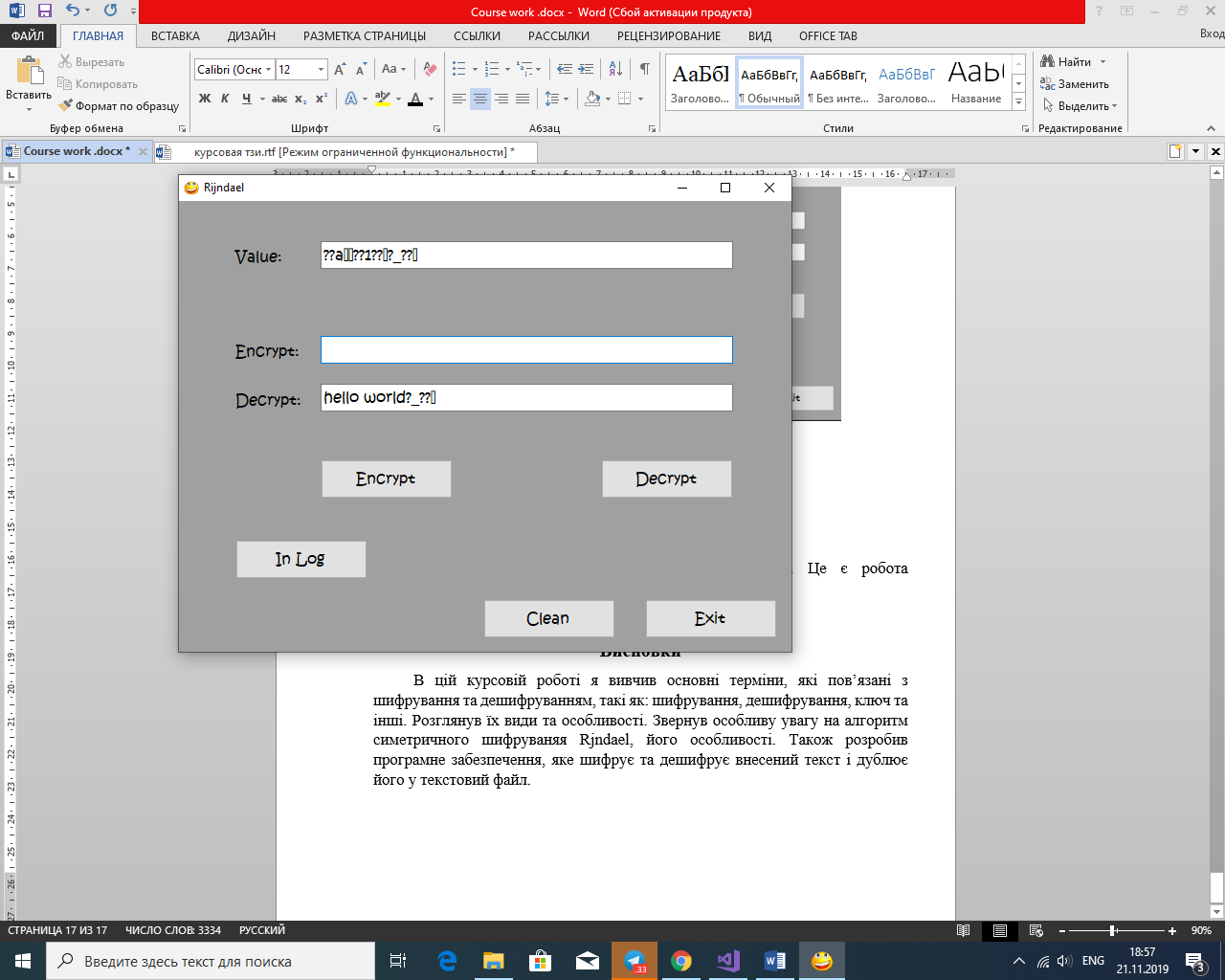
}

**4. Скріншоти програми**

**4.1. Шифратор**



**4.2.Дешифратор**



**5. Примітки**

При дешифровці тексту з’являються «зайві» символи. Це є робота «доповнення» - padding mode.

**Висновки**

В цій курсовій роботі я вивчив основні терміни, які пов’язані з шифрування та дешифруванням, такі як: шифрування, дешифрування, ключ та інші. Розглянув їх види та особливості. Звернув особливу увагу на алгоритм симетричного шифруваняя Rjndael, його особливості. Також розробив програмне забезпечення, яке шифрує та дешифрує внесений текст і дублює його у текстовий файл.

**Джерела**

<https://www.youtube.com/watch?v=QKGjWMlba9s>

<https://www.youtube.com/watch?v=cYLX94DR8_Q>

<https://www.youtube.com/watch?v=GraxONsZg1M&t=102s>

<https://uk.wikipedia.org/wiki/Advanced_Encryption_Standard>

<https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.rijndael?view=netframework-4.8>

<https://uk.wikipedia.org/wiki/ASCII>