Київський національний торговельно-економічний університет

Кафедра комп’ютерних наук

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ
на тему:

«Розробка інформаційної системи захисту даних на підприємстві ТОВ«Domino’s Pizza»»»

Студента 4 курсу, 11 групи,
факультету обліку, аудиту та інформаційних систем, денної форми навчання спеціальності 122 «Комп’ютерні науки» _____ Гладуник Юлії Богданівни

Науковий керівник
кандидат фіз.-матем. н.
доцент _____ Самойленко Анна Тимофіївна

Гарант освітньої програми
кандидат техн. наук
доцент _____ Демідов Павло Георгійович

Київ 2019

ЗМІСТ

АННОТАЦІЯ

Гладуник Ю.Б. Розробка інформаційної системи захисту даних на підприємстві ТОВ «Domino’s Pizza».

Дослідження присвячене розробці програми шифрування текстових даних, для захисту від несанкціонованого доступу під час передачі інформації по інформаційно-телекомунікаційних каналах автоматизованої системи підприємства. В проекті досліджено сучасні алгоритми шифрування та розроблено програму криптографічного захисту інформаційних ресурсів згідно з потребами підприємства ТОВ «Domino’s Pizza».

ABSTRACT

Hladunyk Yu.B. Development of data protection information system at Domino's Pizza LLC.

The reseach is devoted to the development of a program for encrypting text data, to protect against unauthorized access during the transmission of information on the information and telecommunication channels of the automated enterprise system. The project explores modern encryption algorithms and developed a program for cryptographic protection of information resources in accordance with the needs of Domino's Pizza LLC.

ВСТУП

З розвитком інформаційних технологій, які використовуються в ERP-системі, зростає і складність забезпечення її інформаційної безпеки. Однією з найбільш поширених і різноманітних інформаційних загроз, яка може завдати суттєву шкоду інформаційній безпеці ERP-системи є несанкціонований доступ (НСД) до її інформаційних ресурсів.

Досвід експлуатації ERP-систем показує, що незважаючи на тенденцію до підвищення рівня її інформаційної захищеності та постійне розширення інформаційно-телекомунікаційних мереж вона є досить уразливою з точки зору НСД. В цих умовах криптографічний захист інформації в ERP-системі вважається найбільш надійним, а для інформації, яка передається по її інформаційно-телекомунікаційним каналам великої протяжності - єдиним засобом захисту інформації від НСД. Тому питання, що пов'язані з криптографічним захистом ERP-системи від НСД до її інформаційних ресурсів стають досить актуальними.

Метою кваліфікаційного проекту є розробка системи захисту даних на підприємстві ТОВ «Dominos Pizza».

Завданням дипломного проекту є розробка програми шифрування даних, яка забезпечить захист інформації, яка зберігається в інформаційній системі, передається електронною поштою, друкується в текстових редакторах і т.д., від НСД.

Об'єктом дослідження є система безпеки даних на підприємстві.

Предмет досліджень є технологія розробки інформаційної системи захисту даних на підприємстві за допомогою методів криптографічного захисту.

РОЗДІЛ 1 ERP СИСТЕМИ ТА ЗАХИСТ ІНФОРМАЦІЇ В ERP СИСТЕМАХ ПІДПРИЄМСТВА

1.1 Поняття ERP системи.

ERP система – це інформаційна система для ідентифікації і планування всіх ресурсів підприємства, які необхідні для здійснення продажу, виготовлення, закупок і обліку в процесі виконання замовлень клієнтів.

Проте термін ERP може означати не тільки інформаційну систему але й відповідну методологію управління, що реалізується і підтримується цією інформаційною системою.

Сучасна ERP-система має триланкову клієнт-серверну архітектуру, це: рівень бази даних (БД), рівень додатків і рівень представлення (призначений для користувача) (Рис.1.1).

Рис 1.1 Архітектура ERP-системи

Зберігання даних здійснюється в базі даних (рівень БД), їх обробка - на сервері додатків (рівень додатків) і безпосередня взаємодія з користувачем

відбувається через програму "Клієнт" з графічним інтерфейсом.[1,35]

1.2 Безпека ERP-систем

Захист інформації – це сукупність організаційних, технічних та правових заходів, спрямованих на запобігання нанесенню збитків інтересам власника інформації.

Всі заходи та засоби по захисту інформації можна поділити на декілька груп:

1. Юридичні заходи передбачають наявність законів, які визначають відповідальність осіб, що знищують, пошкоджують інформацію, використовують її без належного дозволу, або сприяють цьому.
2. Адміністративні (організаційні) - це заходи, що регламентують процес функціонування системи, використання її ресурсів, діяльність персоналу, тощо.
3. Фізичні заходи захисту включають охорону приміщень, техніки та персоналу, встановлення на дверях приміщень кодових замків, систем санкціонованого доступу тощо.
4. Технічні засоби передбачають використання пристроїв, які зменшують ймовірність руйнування та викрадання інформації.
5. Програмні засоби використовуються для:

- визначення та обмеження прав користувачів по доступу до системи
- шифрування та розшифрування інформації, що зберігається
- фіксування дій користувачів доступу до системи або інформації
- відновлення знищеної інформації на носіях, якщо знищення відбулось на логічному, а не фізичному рівні. [5,12]

Головні цілі і завдання інформаційної безпеки це:

- зменшення ризиків втрати / розкриття інформації;
- відповідність державним і внутрішньокорпоративним нормам захисту інформації;

- захист цілісності даних;
- гарантія конфіденційності внутрішньої інформації підприємства. [2,112]

Забезпечення в тій чи іншій мірі захищеності інформації можливо на кожному з рівнів ERP-систем. Загальним середовищем для компонентів, що знаходяться на різних архітектурних рівнях ERP, є мережева інфраструктура. У підсумку, розмірковуючи про інформаційну безпеку, умовно можна виділити наступні основні аспекти:

- мережева безпека;
- безпека БД;
- безпека на рівні сервера додатків;
- захист інформації на клієнтському комп'ютері.

Такі рівні в сукупності власне і складають ERP як систему. [4,65]

1.3 Прикладна безпека ERP-систем

Перш за все ERP-система повинна безперервно працювати і виконувати свої функції. Адже для будь-якого бізнесу такий збій загрожує великим втратам. Потрібно забезпечити високу ступінь надійності такої системи. Оскільки сьогодні жоден виробник ПО або апаратних засобів не може дати стовідсоткової гарантії надійності своїх рішень, слід заздалегідь передбачити процедури відновлення працездатності системи після збоїв.

Так як ERP являють собою складні системи з великою кількістю користувачів, то основою ефективного управління безпекою повинна служити можливість централізовано змінювати параметри політики безпеки. Такими змінами можуть бути:

- створення і видалення користувачів ERP;
- привласнення прав користувачам;
- оновлення ПЗ на клієнтських комп'ютерах і т.п. [4,113]

Часто ERP будується не як окрема монолітна система з одним сервером додатків, а як розподілений набір окремих систем.

У кожній подібній системі існує свій набір користувачів, які часто дублюються. Непогано було б в такому випадку
мати засіб централізованого управління користувачами і їх повноваженнями, і в деяких ERP такі засоби є.

У ERP-системі передбачений спеціальний механізм Central User Administration (CUA) для централізованого управління користувачами і їх повноваженнями. CUA дозволяє виконувати наступні функції:

- уніфікацію облікових записів;
- призначення прав користувачам;
- ведення локальних і глобальних властивостей в облікових записах.

В ERP-системах існують різні засоби контролю і аудиту дій користувачів. Для кожного конкретного проекту впровадження ERP в залежності від поточних внутрішніх і зовнішніх вимог, буде спроектована своя підсистема контролю.

Загальноприйняті етапи побудови такої системи:

- Визначення цілей контролю і стратегії відстеження ризиків.
- Аналіз ризиків.
- Визначення засобів контролю.
- Знаходження відповідних засобів контролю для кожного з ризиків.
- Моніторинг і аудит роботи системи контролю.

На першому етапі визначається стратегія системи контролю, заснована на внутрішніх і зовнішніх вимогах до інформаційної безпеки. На другому етапі складається набір ризиків, які будуть відслідковуватися. На третьому етапі визначаються всі доступні в даній ERP-системі механізми контролю.

Для прикладу в якості засобів контролю може використовуватися системний журнал, в який заносяться такі події:

- відкриття/закриття сесії користувачем;
- запит на доступ до захищеного ресурсу;
- створення і знищення об'єкта;

- дії зі зміни правил розмежування доступу. [3,45]

Існують також засоби вибіркового ознайомлення з цією реєстраційною інформацією.

На четвертому етапі для кожного з ідентифікованих раніше ризиків підбирається засіб його відстеження.

П'ятий етап - безпосередньо експлуатація розробленої системи контролю.

Висновки до розділу 1.

Для забезпечення надійного захисту ERP-системи на сьогодні і в подальшому, у системі інформаційної безпеки повинні бути реалізовані найпрогресивніші технології. В ERP-системах існують різні засоби контролю і аудиту дій користувачів, на основі яких формується індивідуальна підсистема контролю, що забезпечує високий ступінь надійності системи.

Для ефективного захисту інформації, постає необхідність реалізації основних методів та засобів захисту даних на всіх рівнях автоматизованої системи. Таких як, юридичні, адміністративні фізичні заходи захисту, технічні та програмні засоби захисту.

Система безпеки повинна надавати можливість зменшення ризиків втрати та розкриття інформації, захист цілісності даних, гарантувати конфіденційність внутрішньої інформації підприємства та повинна відповідати державним та внуьрішньокорпоративним нормам захисту інформації.

РОЗДІЛ 2 КРИПТОГРАФІЧНІ МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ

2.1 Основні положення та визначення.

У більшості операційних систем є механізми ідентифікації користувача, які забезпечують той чи інший рівень захисту інформації.

Основні методи захисту інформації в операційних системах наступні:

• захист інформації за допомогою матриці управління доступом та списків управління доступом;

• захист інформації за допомогою "паролів";

• захист інформації за допомогою шифрування-дешифрування. [8,57]

Недоліки двох перших методів полягають у тому, що "ключі" доступу зберігаються в самій системі. Це може призвести до того, що підготовлений недобросовісний користувач може їх розкрити і скористатись секретною інформацією. При шифруванні інформації ключ кодування не повинен зберігатись у системі. Користувач вводить його тільки тоді, коли зашифровує або розшифровує інформацію.

Проблемою захисту інформації шляхом її перетворення займається криптологія (kryptos - таємний, logos - повідомлення). Вона має два напрямки: криптографію і криптоаналіз. Цілі цих двох напрямків прямо протилежні.

Криптографія займається пошуком, дослідженням і розробкою математичних методів перетворення інформації, основою яких є шифрування, а криптоаналіз - дослідженням можливості розшифровки інформації. [6,34]

Основні напрямки використання криптографічних методів - це передача конфіденційної інформації через канали зв'язку (наприклад, електронна пошта), встановлення дійсності переданих повідомлень, збереження інформації

(документів, баз даних) на носіях у зашифрованому виді.

Сучасна криптографія вивчає і розвиває такі напрямки:

• симетричні криптосистеми (із секретним ключем);

• несиметричні криптосистеми (з відкритим ключем);

• системи електронного підпису;

• системи керування ключами. [10,102]

Криптографічний захист у більшості випадків є більш ефективним і дешевим. Конфіденційність інформації при цьому забезпечується шифруванням переданих документів або всього трафіка. Процес криптографічного захисту даних може здійснюватися як програмно, так і апаратно. Апаратна реалізація відрізняється істотно більшою вартістю, однак їй властиві і переваги це - висока продуктивність, простота, захищеність і т.д. Програмна реалізація більш практична, допускає значну гнучкість у використанні. [9.28]

Криптографічний алгоритм, названий алгоритмом шифрування, представлений деякою математичною функцією, яка використовується для шифрування і розшифрування. Точніше таких функцій дві: одна застосовується для шифрування, а інша – для розшифрування.

Розрізняється шифрування двох типів:

- симетричне (із секретним ключем);
- асиметричне (з відкритим ключем). [7,29]

При симетричному шифруванні (рис. 2.1) створюється ключ, файл разом з цим ключем пропускається через програму шифрування та отриманий результат пересилається адресатові, а сам ключ передається адресатові окремо, використовуючи інший (захищений або дуже надійний) канал зв'язку. Адресат, запустивши ту ж саму шифрувальну програму з отриманим ключем, зможе прочитати повідомлення. Симетричне шифрування не таке надійне, як несиметричне, оскільки ключ може бути перехоплений, але через високу швидкість обміну інформацією воно широко використовується. [13,46]

Рис. 2.1 Симетричне шифрування

Асиметричне шифрування складніше, але і надійніше. Для його реалізації (рис.2.2) потрібні два взаємозалежних ключі: відкритий і закритий. Одержувач повідомляє всім бажаючим свій відкритий ключ, що дозволяє шифрувати для нього повідомлення. Закритий ключ відомий тільки одержувачеві повідомлення. Коли комусь потрібно послати зашифроване повідомлення, він виконує шифрування, використовуючи відкритий ключ одержувача. Одержавши повідомлення, останній розшифровує його за допомогою свого закритого ключа. За підвищену надійність несиметричного шифрування приходиться платити: оскільки обчислення в цьому випадку складніше, то процедура розшифрування займає більше часу. [15,67]

Рис. 2.2 Асиметричне шифрування

2.2 Огляд та вибір алгоритмів шифрування даних.

В сучасних криптоситемах, використовуються комбінації симетричних та асиметричних алгоритмів, для того, аби отримати переваги обох схем. Асиметричні алгоритми використовуються для розповсюдження ключів швидших симетричних алгоритмів. [11,47]

AES відноситься до SP-мереж і ґрунтується на новій архітектурі квадрата (square), для якої характерно:

- уявлення блоку, що шифрується, у вигляді двовимірного байтового масиву;
- шифрування за один раунд всього блоку даних;
- виконання криптографічних перетворень, як над окремими байтами масиву, так і над його рядками і стовпцями, що забезпечує дифузію даних одночасно в двох напрямках - по рядках і по стовпцях. [17,138]

Для шифрування і розшифрування використовуються 128-бітові блоки даних. Дозволяється використовувати три різних довжини ключа - 128,192 або 256 біт. Від розміру ключа залежить число раундів шифрування: довжина 128 біт-10 раундів; довжина 192 біта - 12 раундів; довжина 256 біт -14 раундів. Всі раунди, крім останнього, ідентичні.

При довжині ключа 128 біт вхідними даними для алгоритму є масив з 16 байт in0 , in1 , ..., in15 . Перед початком шифрування байти цього масиву розміщуються послідовно в стовпці в матриці InputBlock (зверху вниз). Усередині алгоритму операції виконуються над матрицею станів State. Кінцеве значення матриці стану OutputBlock є виходом алгоритму і перетворюється в послідовність байтів шифротексту out0 , out1 , ..., out15. [18,136] Основні матриці перетворень в AES-128 зображені на Рис.2.3.

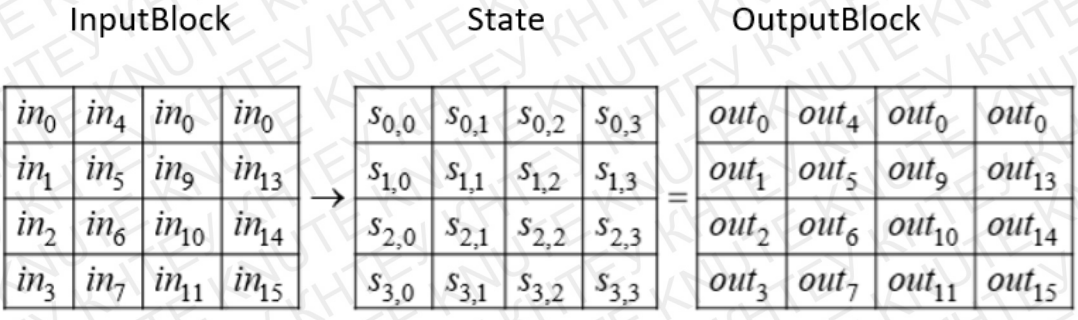

Рис.2.3 Основні матриці перетворень в AES-128

Аналогічно в стовпці матриці InputKey потрапляють і 16 байтів k0, k1, ..., k15 ключа шифру (Рис.2.4).

InputKey

k_0	k_4	k_0	k_0
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

Рис.2.4 Матриця ключа в AES-128

Кожен байт ключа InputKey утворює слово, тобто фактично ключ шифру - це чотири слова w0, w1, w2, w3, де w0 =k0 k1 k2 k3, w1 =k4 k5 k6 k7 і т.д. З цих слів за допомогою спеціального алгоритму (розширення ключа) утворюється послідовність з 44 слів: w0, w1, w2,…,w43 (кожне слово по 32 біта).На кожен раунд шифрування подаються по чотири слова цієї послідовності. Вони і будуть грати роль раундового ключа. Матриця, яка надходить на вхід кожного раунду, називається матрицею InputState, а на виході раунду утворюється матриця OutputState. Очевидно, на в ході першого раунду InputState = InputBlock, а на виході останнього раунду OutputState = OutputBlock. Схема перетворення даних показана на рисунку 2.5.

Перед першим раундом виконується операція AddRoundKey (підсумовування по модулю 2 з початковим ключем шифру). Перетворення, виконані в одному раунді, позначають Round (State, RoundKey), де змінна State - матриця, що описує дані на вході раунду і на його виході після шифрування; змінна RoundKey - матриця, що містить раундовий ключ. [19,142]

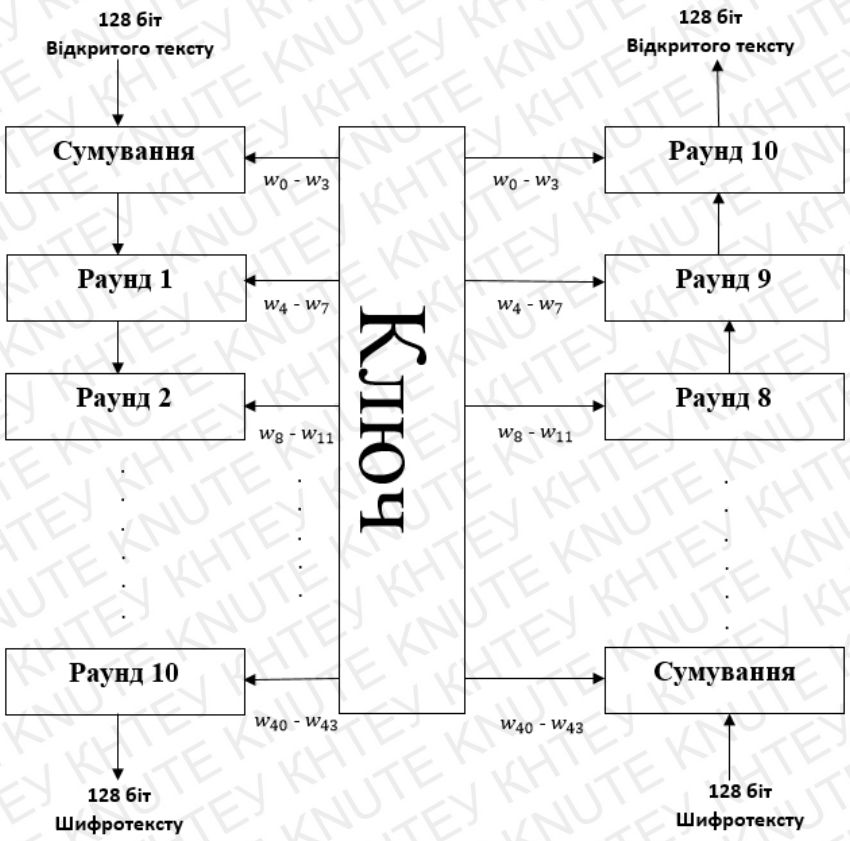

Рис.2.5 Схема перетворення даних в AES-128

Раунд шифрування складається з 4 різних перетворень (Рис.2.6):

1. SubBytes - побайтна підстановка а в S-боксі з фіксованою таблицею замін;
2. ShiftRows - побайтний зсув рядків матриці State на різну кількість байт;
3. MixColumns - перемішування байтів в стовпчиках;
4. AddRoundKey - складання з раундовим ключем (операції XOR).

Останній раунд дещо відрізняється від попередніх тим, що НЕ активізує функцію MixColumns . При розшифруванні в кожному раунді виконуються зворотні операції: InvShiftRows, InvSubBytes, AddRoundKey і InvMixColumns.

Для трьох варіантів ключів AES повний перебір вимагає 2^{127}, 2^{191} або 2^{255}операцій відповідно. Навіть найменша з цих чисел свідчить, що атака з використанням перебору ключів не має практичного значення.

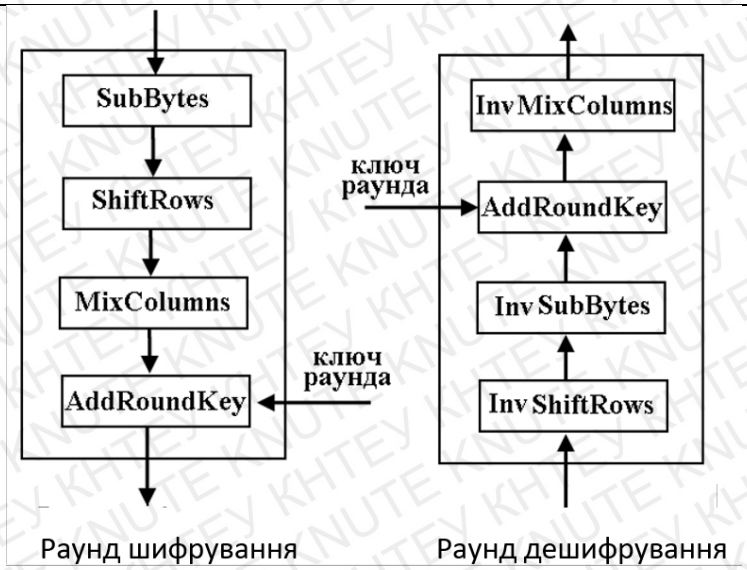

Рис.2.6 Операції в раундах шифрування/розшифрування AES-128

Відповідно до оцінок розробників шифр стійкий проти таких видів криптоаналітичних атак:

- диференціального криптоаналіз;
- лінійного криптоаналіз;
- криптоаналіз на основі пов'язаних ключів. [17,58]

RSA – це криптографічний алгоритм з відкритим ключем, що грунтується на обчислювальній складності задачі факторизації великих цілих чисел. Криптосистема RSA стала першою системою, придатною і для шифрування, і для цифрового підпису.[13,67]

Алгоритм створення відкритого і секретного ключів

- Вибираються два різних випадкових простих числа p і q
- Обчислюється їх добуток $n = pq$, який називається модулем.
- Обчислюється значення функції Ейлера від числа n за формулою (2.1) :

$$\varphi(n) = (p-1)(q-1) \quad (2.1)$$

- Вибирається ціле число e (1<e< φ(n)), взаємно просте з значенням функції $\varphi(n)$.

- Обчислюється число d, мультиплікативно зворотне до числа e по модулю $\varphi(n)$, Тобто число, що задовольняє умову:
  $de \equiv 1 \mod \varphi(n)$.

  Пара $\{e, n\}$ публікується в якості відкритого ключа RSA.

  Пара $\{d, n\}$ відіграє роль закритого ключа RSA і тримається в секреті. [20,53]

  Алгоритм шифрування:

- Взяти відкритий ключ (e, n) С
- Взяти відкритий текст m
- Зашифрувати повідомлення з використанням відкритого ключа С за формулою (2.2):

$$c = E(m) = m^e \, mod \, n \tag{2.2}$$

  Алгоритм дешифрування:

- Прийняти зашифроване повідомлення C
- Взяти закритий ключ (d, n)
- Застосувати закритий ключ для розшифрування повідомлення за формулою (2.3):

$$m = D(c) = c^d \, mod \, n \tag{2.3}$$

Головний плюс алгоритму RSA - маючи відкритий ключ і знаючи алгоритм шифрування неможливо повторити закодоване повідомлення. Хоча через низьку швидкості шифрування (близько 30 кбіт / с при 512 бітному ключі на процесорі 2 ГГц), повідомлення зазвичай шифрують за допомогою більш продуктивних симетричних алгоритмів, а за допомогою RSA шифрують лише цей ключ.

2.3 Вимоги до сучасних криптографічних систем захисту інформації.

Необхідність забезпечення високої пропускної спроможності системи зв'язку і вимога економного використання ресурсу пам'яті в обчислювальних системах висувають таку умову: шифрування не повинно істотно збільшувати довжину

довжину вихідного тексту, а додаткові біти, що вводяться в повідомлення в процесі шифрування, повинні бути повністю та надійно сховані в зашифрованому тексті.[14,23]

Основні вимоги до сучасних криптографічних систем захисту інформації:

- зашифроване повідомлення повинно піддаватися прочитанню тільки при наявності ключа;
- будь-який ключ повинен забезпечувати надійний захист інформації; незначна зміна ключа повинна приводити до істотної зміни виду зашифрованого повідомлення;
- структурні елементи алгоритму шифрування повинні бути незмінними.
- не повинно бути простих і легко встановлюваних залежностей між ключами, які використовуються в процесі експлуатації криптосистеми;
- число операцій, необхідних для визначення ключа з використанням шифрованого повідомлення і відповідного йому відкритого тексту, має бути не менше загального числа можливих ключів; [12,25]

Висновки до розділу 2.

На сьогодні існує багато методів забезпечення захисту інформації, основними з них є:

- Захист інформації за допомогою матриці управління доступом та списків управління доступом
- Захист за допомогою паролів
- Захист за допомогою шифрування даних(криптографія)

Два перші методи не є надійними, оскільки ключі доступу зберігаються в самій системі.

Криптографічний захист у більшості випадків є більш ефективним і дешевим.

Конфіденційність інформації при цьому забезпечується шифруванням переданих документів або всього трафіка.

Розрізняють шифрування двох типів: симетричне і асиметричне.

Для забезпечення високої пропускної спроможності системи зв'язку, надійності шифрування та економного використання ресурсу пам'яті було розглянуто та сформовано основні вимоги до сучасних криптографічних систем захисту інформації.

Для програмної реалізації було обрано наступні алгоритми шифрування:

-RSA

- AES

Основний недолік асиметричної криптографії полягає в низькій швидкості через складність обчислень, необхідних її алгоритмам, в той час як симетрична криптографія традиційно показує високу швидкість роботи.

Однак симетричні криптосистеми мають один істотний недолік – їх використання передбачає наявність захищеного каналу для передачі ключів. Для подолання цього недоліку вдаються до асиметричних криптосистем, які використовують пару ключів: відкритий і закритий. [11,35]

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМИ ДЛЯ ШИФРУВАННЯ ДАНИХ НА ПІДПРИЄМСТВІ

3.1 Вибір методів реалізації алгоритмів шифрування даних.

C# - це об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. синтаксис C# близький до C++ та легкий для вивчення. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Ваговим аспектом є підтримка великої кількості бібліотек і шаблонів. C# в значній мірі підтримується Microsoft. Нові функції і синтаксичні покращення виходять раніше, ніж в інших мовах програмування. [21,34]

Visual Studio – інтегроване середовище розробки програмного забезпечення від фірми Microsoft. Дане середовище дозволяє створювати різноманітні програмні продукти: консольні програми, програми з графічним інтерфейсом, наприклад віконні додатки Windows Forms, а також Web-додатки тощо.[27,13]

Visual Studio надає можливість вибору безлічі інструментів для шифрування даних, а також передбачає інтуїтивний стиль кодування. За замовчуванням Visual Studio форматує код під час введення, автоматично вставляє необхідні відступи, що прискорює процес написання програми.

3.2 Створення користувацького інтерфейсу для програми реалізації алгоритмів шифрування даних.

Користувацький інтерфейс об’єднує в собі всі елементи і компоненти програми, які здатні впливати на взаємодію користувача з програмним забезпеченням.

Для створення користувацького інтерфейсу було використано технології Windows Forms (Рис. 3.1).

Windows Forms - це набір різних керованих бібліотек, за допомогою яких можна виконати всі необхідні для віконної програми дії, починаючи від обміну повідомленнями з операційною системою для відстеження будь-яких подій клієнтського вікна, закінчуючи діалоговими системами, зв'язком з іншими комп'ютерами по мережі і багатьма іншими можливостями.[23,42]

Розроблена програма передбачає можливість шифрування даних двома найбільш поширеними алгоритмами. Вибір алгоритму реалізується за допомогою елемента управління «RadioButton». Також форма передбачає поля для вводу ключів шифрування представлених за допомогою елемента управління «TextBox». Після обрання методу шифрування та вводу ключового слова користувач має змогу зашифрувати або розшифрувати текст, відповідно до своїх потреб.

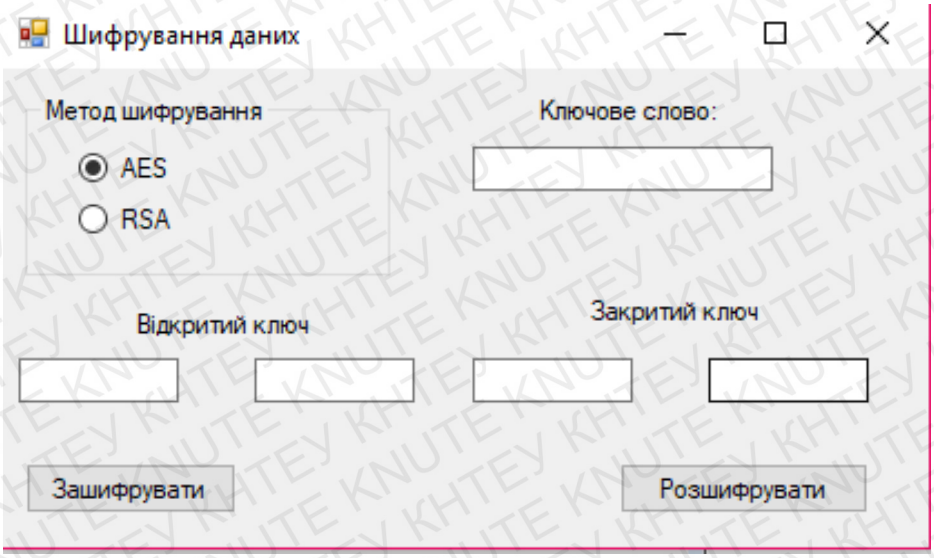

Рис. 3.1 Інтерфейс користувача

Необхідний текст вводиться до файлу «in.txt», який знаходиться в папці «Debug». Програма зчитує з нього всі данні, потім, відповідно до обраного типу шифрування реалізує алгоритм шифрування або розшифрування, виходячи з обраного користувачем режиму. Результат обробки записується в файл «out.txt», який теж знаходиться в папці «Debug». (Рис. 3.2, 3.3)

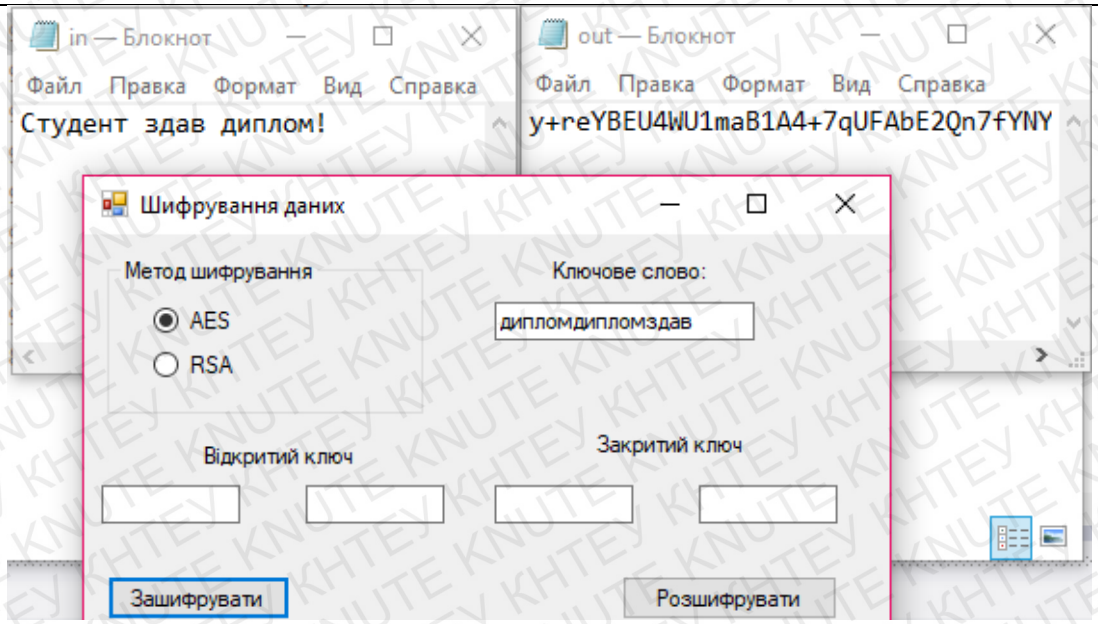

Рис. 3.2 Шифрування алгоритмом AES.

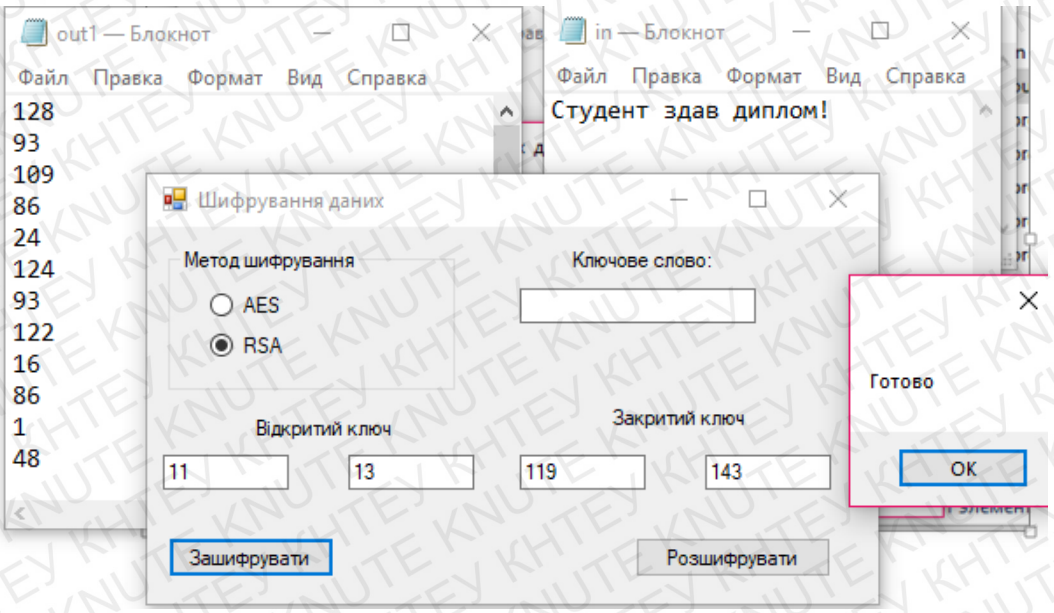

Рис. 3.3 Шифрування алгоритмом RSA.

Оскільки при шифруванні алгоритмом AES ключ повинен складатися з 16 символів, а в алгоритмі RSA відкритий ключ повинен складатися з простих чисел, при некоректному введенні ключа система попереджую користувача про помилку. (Рис. 3.4, 3.5)

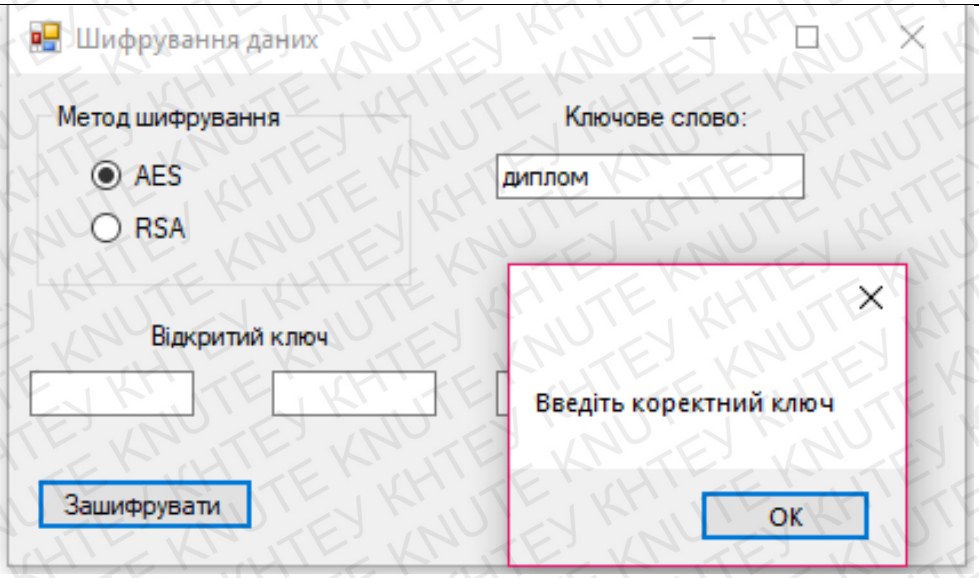

Рис.3.4 Попередження програми при некоректному введенні ключа AES.

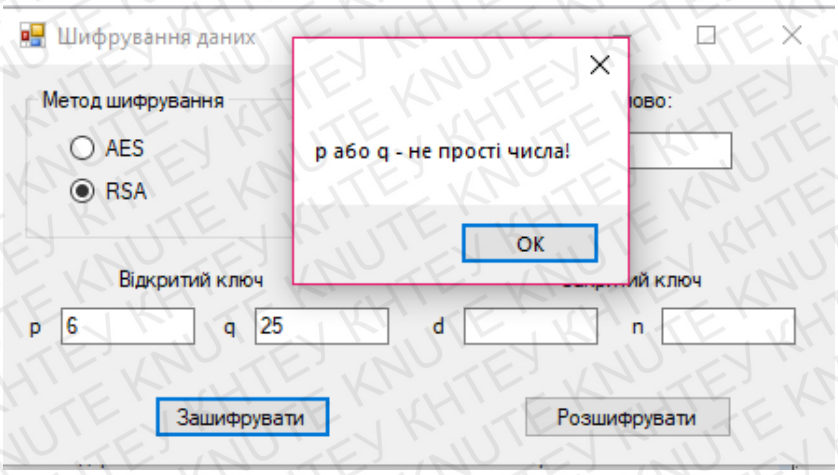

Рис. 3.5 Попередження програми при некоректному введенні ключа RSA.

3.3 Програмна реалізація алгоритмів шифрування даних.

Для реалізації кожного виду шифрування та розшифрування було розроблено методи, які враховують усі алгоритмічні аспекти для кожного з них. В програмі міститься 9 методів, кожен з який має унікальну структуру.

Метод реалізації алгоритму AES (Рис.3.6) отримує як вхідні параметри текст для шифрування та ключ для реалізації шифрування. Для роботи з алгоритмом шифрування AES в Microsoft Visual Studio передбачено клас “Aes”, для того, щоб мати змогу взаємодіяти з цим класом необхідно підключити простір імен «System.Security.Cryptography».[27,68]

Після отримання даних метод перетворює ключ у масив байтів та генерує вектор ініціалізації, який буде необхідний для подальшого шифрування. Після цього створюються потоки для шифрування та обробки даних, які і забезпечують реалізацію алгоритму. Після шифрування до отриманих даних додається значення вектору ініціалізації, що є необхідною умовою для розшифрування даних.(Рис. 3.6)

```
public static byte[] ToAes256(string plainText, string key)
{
    byte[] Key = System.Text.UTF8Encoding.UTF8.GetBytes(key);
    if (plainText == null || plainText.Length <= 0)
        throw new ArgumentNullException("plainText");
    if (Key == null || Key.Length <= 0)
        throw new ArgumentNullException("Key");
    byte[] encrypted;
    using (Aes aesAlg = Aes.Create())
    {
        aesAlg.GenerateIV();
        aesAlg.Key = Key;
        ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);
        using (MemoryStream msEncrypt = new MemoryStream())
        {
            using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))
            {
                using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
                {
                    swEncrypt.Write(plainText);
                }
                encrypted = msEncrypt.ToArray();
                encrypted = encrypted.Concat(aesAlg.IV).ToArray();

            }
        }

    }
    return encrypted;
```

Рис. 3.6 Метод реалізації алгоритму AES.

В якості вхідних параметрів метод розшифрування даних приймає байтовий масив, який містить зашифрований текст та ключ, для його розшифрування (Рис. 3.7). Оскільки значення вектору ініціалізації міститься в отриманому байтовому масиві, постає необхідність його відділення, яка реалізується за допомогою циклу та передання його в окремий масив.

На основі отриманих значень ключа та вектору ініціалізації створюється екземпляр об’єкту для розшифрування даних, який передається в один з потоків, які реалізують алгоритм розшифрування та представляють результат у вигляді текстової змінної.

```
public static string FromAes256(byte[] cipherText, string key)
{
    byte[] bytesIv = new byte[16];
    byte[] Key = System.Text.UTF8Encoding.UTF8.GetBytes(key);
    if (cipherText == null || cipherText.Length <= 0)
        throw new ArgumentNullException("cipherText");
    if (Key == null || Key.Length <= 0)
        throw new ArgumentNullException("Key");
    for (int i = cipherText.Length - 16, j = 0; i < cipherText.Length; i++, j++)
        bytesIv[j] = cipherText[i];
    byte[] mess = new byte[cipherText.Length - 16];
    for (int i = 0; i < cipherText.Length - 16; i++)
        mess[i] = cipherText[i];
    string plaintext = null;
    using (Aes aesAlg = Aes.Create())
    {
        aesAlg.Key = Key;
        aesAlg.IV = bytesIv;
        ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);
        using (MemoryStream msDecrypt = new MemoryStream(mess, 0, mess.Length))
        {
            using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
            {  using (StreamReader srDecrypt = new StreamReader(csDecrypt))
                {
                    plaintext = srDecrypt.ReadToEnd();
                }
            }
        }
    }
    return plaintext;
```

Рис. 3.7 Метод розшифрування алгоритму AES

Для шифрування за алгоритмом RSA створенно ряд методів, які забезбечують реалізацію усіх етапів алгоритму шифрування даних.

Метод, який міститься в обробнику події натиску кнопки, зчитує дані для шифрування з текстового файлу та прості числа, які будуть використані для створення відкритого числа, при введені чисел, які не є простими, що перевіряється за допомогою метода «IsTheNumberSimple», користувач отримує повідомлення про помилку. Метод вираховує добуток отриманих простих чисел та обчислює функцію Ейлера (m).

За допомогою метода «Calculate_d» визначається число d, яке є взаємно простим з числом m, за допомогою метода «Calculate_e» розраховується число e, яке відповідає умові: (e · d) mod (m) = 1. Отримані дані передаються в метод «RSA_Endoce», що повертає зашифрований текст, який записується до текстового файлу. Значення чисел закритого ключа відображаються на формі. Реалізація даного функціоналу зображена на Рис. 3.8.

```
if (RSArbutton.Checked)
{
    if ((textBox_p.Text.Length > 0) && (textBox_q.Text.Length > 0))
    {
        long p = Convert.ToInt64(textBox_p.Text);
        long q = Convert.ToInt64(textBox_q.Text);

        if (IsTheNumberSimple(p) && IsTheNumberSimple(q))
        {
            string s = "";

            StreamReader sr = new StreamReader("in.txt");

            while (!sr.EndOfStream)
            {
                s += sr.ReadLine();
            }

            sr.Close();

            s = s.ToUpper();

            long n = p * q;
            long m = (p - 1) * (q - 1);
            long d = Calculate_d(m);
            long e_ = Calculate_e(d, m);

            List<string> result = RSA_Endoce(s, e_, n);

            StreamWriter sw = new StreamWriter("out1.txt");
            foreach (string item in result)
                sw.WriteLine(item);
            sw.Close();

            textBox_d.Text = d.ToString();
            textBox_n.Text = n.ToString();

            Process.Start("out1.txt");
        }
        else
            MessageBox.Show("p або q - не прості числа!");
    }
    else
        MessageBox.Show("Введіть p і q!");
}
```

Рис. 3.8 Обробник події натиску кнопки шифрування за алгоритмом RSA.

Метод «RSA_Endoce» отримує як вхідні параметри строку, яку необхідно зашифрувати, число e, розраховане за допомогою методу «Calculate_e» і число n – добуток зчитаних простих чисел.

Метод реалізує шифрування відповідно до формули C(i) = (M(i)e) mod n та повертає масив отриманих значень. При приведенні числа до степеня в даному випадку виходять дуже великі числа, які не вміщуються ні в один зі стандартних типів. Тому для їх зберігання використовується екземпляр класу BigInteger.(Рис. 3.9)

```
ссылка: 1
private List<string> RSA_Endoce(string s, long e, long n)
{
    List<string> result = new List<string>();

    BigInteger bi;

    for (int i = 0; i < s.Length; i++)
    {
        int index = Array.IndexOf(characters, s[i]);

        bi = new BigInteger(index);
        bi = BigInteger.Pow(bi, (int)e);

        BigInteger n_ = new BigInteger((int)n);

        bi = bi % n_;

        result.Add(bi.ToString());
    }

    return result;
```

Рис.3.9 Метод реалізації шифрування за алгоритмом RSA.

Для розшифрування використовується метод, який зчитує зашифровані дані та числа, які є закритим ключем. Отримані дані передаються в метод «RSA_Dedoce», який реалізує алгоритм розшифрування, відповідно до формули M(i) = (C(i)d) mod n та повертає масив, в якому містять розшифровані дані. Отримані дані записуються в текстовий файл, який виводиться на екран.(Рис. 3.10)

```
ссылка: 1
private string RSA_Dedoce(List<string> input, long d, long n)
{
    string result = "";

    BigInteger bi;

    foreach (string item in input)
    {
        bi = new BigInteger(Convert.ToDouble(item));
        bi = BigInteger.Pow(bi, (int)d);

        BigInteger n_ = new BigInteger((int)n);

        bi = bi % n_;

        int index = Convert.ToInt32(bi.ToString());

        result += characters[index].ToString();
    }

    return result;
}
ссылка: 1
```

Рис. 3.10 Метод реалізації розшифрування за алгоритмом RSA.

Висновки до розділу 3.

Для програмної реалізації було обрано мову програмування C#, оскільки в ній на достатньому рівні організована підтримка усіх відомих алгоритмів шифрування даних. Ваговим аспектом є підтримка великої кількості бібліотек і шаблонів. C# в значній мірі підтримується Microsoft. Нові функції і синтаксичні покращення виходять раніше, ніж в інших мовах програмування.

Для створення користувацького інтерфейсу, були використані технології Windows Forms. Функціонал інтерфейсу користувача надає можливість вибору одного з двох алгоритмів шифрування. Після обробки даних програма видає сповіщення про завершення роботи. Також врахована можливість некоректного введення ключа. Розроблений інтерфейс, зрозумілий і простий у використанні.

Для реалізації алгоритмів шифрування і розшифрування створені методи, які враховують алгоритмічні особливості кожного способу шифрування. В якості вхідних параметрів приймаються дані для шифрування чи розшифрування і ключ, які задає користувач. Результат виводиться в окремий файл. Виходячи з розробленої програми, були сформовані вимоги, необхідні для її коректного функціювання.

ВИСНОВКИ

З розвитком інформаційних технологій, які використовуються в ERP-системі, передбачається використання найсучасніших засобів, на основі яких відбувається контроль і аудит дій користувачів, що і забезпечує високий рівень надійності системи.

Ключове завдання системи захисту полягає у забезпеченні можливості зменшення ризиків втрати або розкриття конфіденційної інформації та захисту цілісності даних.

В роботі було розглянуто та проаналізовано сучасні методи забезпечення захисту інформації, яких на даному етапі розвитку існує досить велика кількість. З яких в свою чергу виділяють кілька основних. Після розгляду і дослідження кожного з них було визначено, що найбільш ефективним, надійним і дешевим є криптографічний захист (захист за допомогою шифрування даних). Було визначено головні недоліки і переваги найвідоміших алгоритмів шифрування. Найважливішими критеріями при виборі оптимального алгоритму були поставлені швидкість роботи, криптостійкість та тонкощі реалізації. Завдяки даному аналізу було обрано симетричний алгоритм AES, також відомий як Rijndael, та асиметричний – RSA.

В ході роботи було розроблено та реалізовано програмне забезпечення для шифрування будь-яких файлів по користувацькому паролю. Також створено зручний користувацький інтерфейс. Програмний продукт використовує алгоритми шифрування AES та RSA. Розроблене програмне забезпечення призначене для шифрування інформації, яка передається в інформаційній системі чи зберігається в базах даних підприємства. Для шифрування текстових даних доцільним буде використання симетричного алгоритму AES, оскільки він забезпечує високу швидкість обробки даних.

Проте ключ даного алгоритму не варто передавати по незахищених каналах зв’язку, тому для шифрування ключа можна використати асиметричний алгоритм RSA, який забезпечить надійність даної системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шаповал В. Л. Фактори успіху впровадження ERP-систем / В. Л. Шаповал. // Сучасний захист інформації. – 2014. – №3. – С. 120.
2. Інформаційні системи і технології на підприємствах: підручник / В.Л. Плескач, Т.Г. Затонацька. - К.: Знання, 2011. - 718 с.
3. Управління проектами, навчальний посібник. / Н. О. Петренко, Л. О. Кустріч, М. О. Гоменюк. - К.: "Центр учбової літератури", 2015. - 244 с.
4. Жданов О.Н. Методы и средства криптографической защиты: учеб. пособие / О.Н Жданов, В.В. Золотарев; СибГАУ. – Красноярск, 2017. – 217 с. 6. Яковлев А.В.
5. Криптографическая защита информации: учеб. пособие / А.В. Яковлев, А.А. Безбогов, В.В. Родин, В.Н. Шамкин. – Тамбов: Изд-во Тамб. гос. техн. ун-та, 2013. – 140 с..
6. Варлатая. С.К. Программно-аппаратная защита информации: учеб. пособие /С.К. Варлатая, М.В. Шаханова. - Владивосток: Изд-во ДВГТУ, 2010. – 318 с.
7. Венбо Мао. Современная криптография. Теория и практика. М: Вильямс, 2015. – 768 с.
8. Бернет С., Пэйн С., Криптография. Официальное руководство RSA Security. Изд. 2-е, стереотипное. – М.: ООО «Бином-Пресс», 2017. – 384 с.: ил.
9. Аграновский А.В., Хади Р.А. Практическая криптография: алгоритмы и их программирование. – М.: СОЛОН-Пресс, 2012. – 256 с.
10. Адаменко М.В. Основы классической криптологии: секреты шифров и кодов. – М.: ДМК Пресс, 2012. – 256 с.
11. Бабаш А. В. Криптография. – М.: СОЛОН-Пресс, 2010. – 511 с.
12. Баричев С. Г., Гончаров В. В., Серов Р. Е. Основы современной криптографии: Учебное пособие. М.: Горячая Линия - Телеком, 2012. – 175 с.
13. Ростовцев А.Г., Маховенко Е.Б. Теоретическая криптография. М.: Издательство: АНО НПО "Профессионал", 2015. – 480 с.

14. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. – М.: Гелиос АРВ, 2011. – 480 с.
15. Маховенко Е. Б. Теоретико-числовые методы в криптографии. М.: Гелиос АРВ,2016.–320 с.
16. Мухачев В.А., Хорошко В.А. Методы практической криптографии. К.: ООО Полиграф-Консалдинг, 2009. – 209 с.
17. Рябко Б.Я., Фионов А.Н. Криптографические методы защиты информации. М.: Горячая линия – Телеком, 2009. – 229 с.
18. Ростовцев А.Г. Алгебраические основы криптографии. М.: НПО «Мир и семья», ООО «Интерлайн», 2010. – 256 с.
19. Лотка, Рокфорд C# и CSLA .NET Framework. Разработка бизнес-объектов / Рокфорд Лотка. - М.: Вильямс, 2010. - 816 с.
20. Мак-Дональд, Мэтью Silverlight 5 с примерами на C# для профессионалов / Мэтью Мак-Дональд. - М.: Вильямс, 2013. - 848 с.
21. Марченко, А. Л. Основы программирования на C# 2.0 / А.Л. Марченко. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2011. - 552 с.
22. Подбельский, В. В. Язык C#. Базовый курс / В.В. Подбельский. - М.: Финансы и статистика, Инфра-М, 2011. - 384 с.
23. Прайс, Джейсон Visual C# 2.0. Полное руководство / Джейсон Прайс, Майк Гандэрлой. - М.: Век +, Корона-Век, Энтроп, 2010. - 736 с.
24. Рихтер, Джеффри CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C# / Джеффри Рихтер. - М.: Питер, 2013. - 928 с.
25. Смоленцев, Н. К. MATLAB. Программирование на Visual C#, Borland JBuilder, VBA (+ CD-ROM) / Н.К. Смоленцев. - М.: ДМК Пресс, 2011. - 456 с.

26. Сердюк В.А. Уязвимость и информационная безопасность ERP-систем. – Режим доступа: http://www.connect.ru/article.asp?id=3167.
27. Модель криптографии .NET Framework [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/dotnet/standard/security/cryptography-model.
28. Шифрование данных [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/dotnet/standard/security/encrypting-data.
29. Пошаговое руководство. Создание криптографического приложения [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/dotnet/standard/security/walkthrough-creating-a-cryptographic-application.
30. Знакомство с возможностями развертывания в Visual Studio [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/visualstudio/deployment/deploying-applications-services-and-components?view=vs-2019
31. Чуйко Ф. Зачем необходимо защищать ERPсистему и как это сделать. – Режим доступа: http://ko.com.ua/zachem_neobhodimo_zashhishhat_erp-sistemu_i_kak_jeto_sdelat_88529.