МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА
Факультет економіки і підприємництва
Кафедра інформаційних технологій

Допущено до захисту
"_____" _____ 2023 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ДЛЯ ШИФРУВАННЯ ПОВІДОМЛЕНЬ

Виконав – студент 4 курсу, групи 41-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки
Карпюк Андрій Віталійович

Науковий керівник – к.т.н., доцент
Ліщук Роман Ігорович

Умань – 2023

Зміст

ВСТУП 3

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ТА ПОСТАНОВКА ЗАДАЧІ 5

1.1. Огляд наявних рішень для шифрування повідомлень 5

1.2. Постановка задачі 11

РОЗДІЛ 2 КРИПТОГРАФІЧНІ МЕТОДИ ШИФРУВАННЯ ІНФОРМАЦІЇ 12

2.1. Симетричне шифрування 12

2.2. Асиметричне шифрування 21

2.3. Гібридне шифрування та аналіз методів шифрування 24

2.4. Опис програмного середовища 26

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ З ВИКОРИСТАННЯМ КРИПТОГРАФІЧНОГО МЕТОДУ ШИФРУВАННЯ ДАНИХ AES 28

3.1. Програмна реалізація 28

3.1.1. HTML 28

3.1.2. CSS 33

3.1.3. SVG 37

3.1.4. JavaScript 40

ВИСНОВКИ 45

СПИСОК ЛІТЕРАТУРИ 46

ВСТУП

В нинішній період ми переходимо від індустріального суспільства до інформаційного, де інформація стає найважливішим ресурсом, тому нині розроблено безліч способів захисту інформації. Тому в конкурентній боротьбі багато хто хоче отримати інформацію, самими різними способами, навіть такими, як шпигунство за допомогою технічних засобів розвідки. Використання автоматизованих систем обробки інформації та управління загострило захист інформації, від несанкціонованого доступу. Головними проблемами інформаційного захисту в комп'ютерних системах (КС) є проблема пов'язки інформації з носієм (у більшості випадків інформація ніяк не зв'язана з носієм).

Через це її можна скопіювати та передати по каналах зв'язку. Інформаційна система схильна як до зовнішніх, так і внутрішніх загроз із боку порушників.

Більшість витоків інформації пов'язаних з порушенням інформаційної безпеки викликана внутрішніми загрозами, джерелами яких є легальні користувачі системи. Треба зауважити, що загрозою номер один для будь якої КС є внутрішній витік інформації, тієї, що оброблюється та зберігається всередині цієї системи. Джерелами таких загроз, зазвичай, є недобросовісні співробітники, котрі з деяких причин прагнуть завдати організації матеріальний або фінансовий збиток.

Розв'язання проблем захисту електронної інформації базується в основному на використанні криптографічних методів. До того ж сучасні методи криптографічного перетворення зберігають вихідну продуктивність автоматизованої системи, що є важливим. Це є найбільш ефективним способом, що забезпечує конфіденційність даних, їх цілісність і автентичність відбитку.

Використання криптографічних методів в сукупності з технічними й організаційними заходами забезпечують надійний захист від широкого спектра загроз.

Сьогодні розвиток способів захисту комп’ютерних систем від несанкціонованого доступу обумовлений швидким зростанням ролі програмних і криптографічних механізмів в порівнянні з апаратними. Нові проблеми захисту інформації кожен раз вимагають застосування протоколів та механізмів з високою обчислювальною складністю. В результаті загальнодоступності інформації в глобальній мережі, виявляється слабкість традиційних механізмів і відставання вживання сучасних методів захисту.

Криптографія розширює резерв захисту інформації та забезпечує її безпеку в мережі. Стратегічно правильним розв'язання проблеми захисту інформації, є використання досягнень криптографії. Відсутність достатньої кількості засобів захисту інформації на внутрішньому ринку тривалий час не дозволяло втілювати в життя заходи з приводу захисту даних в необхідних масштабах. На даний час існує велика кількість різних видів шифрування тому, головним завданням моєї бакалаврської роботи є їх аналіз, та розробка додатку, який захищає данні за допомогою шифрування.

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Огляд наявних рішень для шифрування повідомлень

Нині в інформаційному просторі, дуже швидко впроваджуються комп'ютерні і телекомунікаційні технологій. Комп'ютерні системи впроваджуються в усі сфери нашого життя від фінансової до соціальної. Через це зростає інтерес користувачів до проблем захисту інформації. Згідно із Законом України «Про захист інформації в автоматизованих системах» захист інформації - це сукупність заходів і правових норм для попередження заподіяння збитку інтересам власника інформації.[1] Останнім часом, на фоні розвитку комерційної і підприємницької діяльності, збільшилося число спроб несанкціонованого доступу до конфіденційної інформації. Серед всіх методів захисту даних від несанкціонованого доступу важливе місце займають криптографічні методи. Посилаючись на веб-сайт Wikipedia, можна сказати, що криптографія – наука про математичні методи забезпечення конфіденційності і автентичності інформації [2]. У сучасній криптографії найчастіше використовуються відкриті алгоритми шифрування, що використовують обчислювальні засоби. Аналогічно згідно із Законом України «Про захист інформації в автоматизованих системах» криптографічний захист інформації – вид захисту інформації, що реалізується шляхом перетворення інформації з метою приховування змісту інформації. [1] Цінність інформації була усвідомлена людьми ще за до античних часів.

Звідти з’явилась потреба захисту інформації від сторонніх людей. Наші предки використовували багато різних методів, одним з яких був тайнопис. Тайнопис – це вміння складати повідомлення таким чином, щоб зрозуміти його міг тільки той, хто розумів таємницю цього методу захисту інформації. Є інформація, що це мистецтво з’явилося ще в до античні часи та існувало до недавнього часу. І лиш декілька десятків років тому інформація набула свою комерційну цінність.

Зараз з нею можна робити що заманеться: проводити, зберігати, транспортувати, продавати і купувати, а значить можна красти і підроблювати,

саме через це гостро потає питання її захисту. Нинішнє суспільство все більше і більше стає інформаційним, будь який вид діяльності тісно залежить від інформації.

Криптографічні методи є дуже важливою складовою захисту даних.

Сучасні методи шифрування гарантуюсь майже абсолютний захист, але у світі немає нічого ідеального, тому у цих методів є проблема реалізації, а саме її надійності. Через це оцінка їх ефективності стала актуальною як ніколи. Ця задача виявилась більш трудомісткою, ніж створення цих самих алгоритмів.

Аналіз потребує більш глибших знань, більш вищої кваліфікації у сфері криптографії. Саме цей факт обумовлює велику кількість засобів криптографічного захисту інформації про які ніхто не знає нічого. При цьому, деякі з них, розробники тримають в секреті.

Згідно з інформації наданої сайтом StudFiles шифрування — це спосіб зміни повідомлення або іншого документа, що забезпечує спотворення (захования) його вмісту [3]. Зашифровувати можна будь-який тип інформації від звичайного тексту до зображень. Ідея шифрування полягає у приховуванні змісту інформації для тих у кого немає засобів дешифровки, водночас залишаючи незмінною інформацію для тих, хто має ключі дешифровки. Шифрування з’явилось задовго до нашої ери. Першим шифрованим текстом вважається єгипетський текст що датується 1900 р до н.е., де замість звичайних єгипетських Для більш повного розуміння поставленої задачі розглянемо основи безпеки, які використовуються для захисту даних.

Витік інформації завжди призводить до небажаних наслідків, великих збитків. Отже безпеки будь-яких видів повинна вирішувати наслідки шкідливої людської діяльності. Усі види безпеки сильно пов’язані з інформаційною безпекою (ІБ), і сама безпека зазвичай забезпечується за допомогою ІБ.

Інформація - це деякі відомості про якийсь предмет незалежно від форми їх уявлення.

Захист інформації – це заходи, які проводяться з метою запобігання можливостей витікання, крадіжки, втрати, поширення, знищення, змінення, фальсифікації або блокування інформації.

Всі види шкідливих дій над інформацією можна поділити на декілька класів:

- Блокування інформації (власник не може отримати доступ до інформації).
- Порушення цілісності (втрата, вихід з ладу носія; змінення, порушення смислу інформації; втрата достовірності).
- Порушення конфіденційності (з інформацією ознайомлюються користувачі які не мають право на доступ до цієї інформації).
- Несанкціоноване тиражування (захист авторських прав і прав власності на інформацію).

Загроза - можлива подія, яка може нанести збитки інтересам деякої особи.

Втіленням загрози є шкідлива зміна процесу роботи системи або інформації.

Розглянемо види загроз.

- Природні загрози – загрози викликані не людською діяльністю, а дією природних явищ, фізичних процесів, такі як стихійні лиха, магнітні бурі, опади.
- Штучні загрози – загрози викликані саме людською, діяльністю.

У штучних загроз також є класифікація, вони поділяються на два типи:

- Ненавмисні – загрози, які з’являються без злого наміру, через випадкові дії людей, через незнання, халатність, цікавість.
- Навмисні – загрози які обумовленні умисною людською діяльністю, націлену на розлад роботи системи, виведення її з ладу, та подальшого злочинного проникнення в систему і несанкціонованого доступу до інформації. [4]

Нині, за допомогою сучасних пристроїв перехоплення, навіть на відстані в десятки метрів, є можливість реєструвати різні побічні сигнали, що виникають

під час роботи різноманітних технічних засобів. Саме по цих сигналах у подальшому можна відновлювати оброблювану, передану, прийняту, копійовану інформацію. Також інформацію можна отримувати не тільки за допомогою перехоплення побічних сигналів а й напряму з інформаційних каналів, за якими передається сама інформація. Такий засіб перехоплення реалізується набагато легше, тому і використовується частіше. З ростом кількості спроб перехоплення інформації саме таким чином, почав зростати попит на заходи захисту інформації від несанкціонованого доступу (НСД). Існує дуже багато видів захисту від таких атак. Одним із базових варіантів є фізичні заходи.

Фізичні заходи захисту інформації базуються на застосуванні різного роду механічних, електронних пристроїв, призначених для створення фізичних перешкод на шляхах проникнення і доступу потенційних порушників до інформації. Розглянемо деякі з них: Ідентифікація – це процес привласнення об’єктам доступу власного ідентифікатора, або процес порівняння ідентифікаторів об’єкта з переліком дозволених ідентифікаторів. Якщо говорити про інформаційні технології, то це звичайне встановлення особистості користувача. Це необхідно для того щоб у подальшому система змогла дозволити доступ до деякої конфіденційної інформації ідентифікованому користувачу. Таким чином ідентифікація є базовим поняттям в ІБ.

Автентифікацією – називається процедура верифікації належності ідентифікатора суб'єкту. Тобто перевірка належності ідентифікатора суб’єкту,

який намагається отримати доступ до інформації. Зазвичай ця процедура відбувається на основі секретного елемента, сама ж система не має ніякого доступу до цього елементу. Зазвичай цей елемент зберігається у виді хеш-суми, що забезпечує безпеку для елементу, який зберігається. Прикладом автентифікації може бути початок роботи в будь якій системі, котра запитує ім’я та пароль користувача. Ідентифікацією буде введення та перевірка імені користувача, а автентифікацією – пароль.

Якщо об’єднувати усе вищесказане, то можна сказати, що для того щоб повністю захистити інформацію потрібно вирішити два типи задач: захист інформації від несанкціонованого доступу і захисту інформації від витоку технічними каналами. Під НСД мається на увазі доступ до інформації, несанкціонованими об’єктами за допомогою інформаційних каналів. Під технічними каналами розуміються канали стороннього електромагнітного випромінювання, акустичні канали, оптичні канали.

Для надійного захисту від НСД в інформаційних системах може здійснюватися за допомогою декількох складових:

- За допомогою системного та прикладного програмного забезпечення.
- За допомогою апаратна частини сервера та його робочих станцій.
- За допомогою комунікаційного устаткування та каналів зв'язку.
- За допомогою периметру інформаційної системи.

На рівні прикладного та системного програмного забезпечення захист інформації відбувається з використанням:

- системи які використовують розмежування доступу до інформації;
- системи які використовують ідентифікацію та автентифікацію;
- системи які використовують аудит й моніторинг інформації;
- системи які використовують антивірусний захист.

На рівні апаратного забезпечення захист інформації відбувається з використанням:

- апаратних ключів;
- систем сигналізації;
- засобів, що блокують пристроїв та інтерфейси вводу-виводу інформації.

Для захисту систем комунікації використовуються:

- Firewall (міжмережевий екран);
- системи які можуть ідентифікувати вторгнення

- додатки, функціонал котрих розрахований на створення віртуальної приватної мережі;
- засоби, які аналізують захищеність.

Захист периметру інформаційної системи здійснюється за допомогою:

- охоронних й пожежних сигналізацій;
- цифрового відеоспостереження;
- систем, які виконують контроль та керування доступом.

Під час створення засобів програмно-апаратного захисту від НСД завжди дотримуються таких принципів:

- принцип обґрунтованого доступу (виконавець повинен мати достатній доступ до інформації, щоб виконувати свої обов'язки)
- принцип глибини контролю доступу (системи захисту інформації повинні мати доступ до всіх видів програмних та інформаційних ресурсів)
- принцип розмежування інформаційних потоків (переписування закритої інформації відбувається тільки на закриті носії, під час переписування відбувається мічення та ідентифікація носія)
- принцип чистоти інформації ресурсів (видалення закритої інформації з ресурсів у випадку їх видаленні);
- принцип відповідальності виконавця (виконавець несе персональну відповідальність за його діяльність в системі)
- принцип цілісності засобів захисту (засоби інформаційного захисту повинні бути ізольовані від користувачів та досконало виконувати свій функціонал). [4]

Дуже поширеною помилкою є недооцінювання непрофесіональних комп'ютерних злочинців. Більшість фінансових злочинів відбулась за допомогою саме таких людей – нелояльних співробітників, котрі лише мали доступ до комп'ютера під'єднаного до системи. Процедури безпеки можуть забезпечувати абсолютну перевірку паролів і строгий контроль доступу до

цінних загальних даних, але зловмисника, обізнаного у внутрішньому устрої системи, практично неможливо зупинити.

Проаналізувавши усе вищесказане можна дійти до висновку, що фізичні методи захисту інформації є дуже ефективними, але мають свої недоліки. Для того щоб удосконалити захист переданої інформації, крім розглянутих методів, необхідно використовувати шифрування. В такому випадку, навидь якщо зловмисник зможе перехопити передану інформацію, то він отримає зашифровані дані, котрі він не зможе розшифрувати за відсутністю ключа дешифровки.

1.2. Постановка задачі

Метою цієї роботи є дослідження методів захисту передачі інформації за допомогою криптографічних методів та написання веб-додатку з шифруванням даних. Спираючись на мій опит програмування, для виконання роботи була обрана мова програмування JavaScript. Таким чином, можна сказати, що основними завданнями є:

1. Дослідження криптографічних методів захисту даних
2. Вибір методу шифрування
3. Розробка та реалізація веб-додатку з шифруванням даних

РОЗДІЛ 2 КРИПТОГРАФІЧНІ МЕТОДИ ШИФРУВАННЯ ІНФОРМАЦІЇ

2.1. Симетричне шифрування

Мало хто знає про реальну важливість SSl сертефікатів, хоча вони є основоположними для безпеки будь-якої інформації в Інтернеті. В свою чергу, всі ці сертифікати базуються на шифруванні за допомогою різних криптографічних методів. Ще раз, шифрування — це спосіб зміни повідомлення або іншого документа, що забезпечує спотворення (захования) його вмісту [3] В свою чергу шифрування базується на криптографічних ключах в суміші з математичними алгоритмами. Для більш глибокого розуміння різних методів шифрування я розгляну їх два основних види: симетричне, асиметричне.

Як можна здогадатись, метод симетричного шифрування використовує як для шифрування, так і для дешифрування один і той самий ключ. Саме це дуже сильно спрощує весь процес. Для повного розуміння цього методу, розглянемо простий приклад.

Існують два користувачі, котрі спілкуються на відстані. У випадку перехоплення повідомлень, для того щоб ніхто не зрозумів значення повідомлень, вони зашифровують їх. Для захисту, вони шифрують повідомлення таким чином, щоб кожна літера змінювалась на букву, яка стоїть в алфавіті на 4 позиції позаду.

Таким чином, замість слова «Єгор» користувачі отримує слово «Гякм». Для розшифрування потрібно зробити такі ж дії, але в зворотному порядку – на місця букв зашифрованого слова потрібно поставити літери, котрі стоять вже на 4 позиції попереду. Сам процес заміни літер і буде ключем до цього шифру. Для того щоб співрозмовники змогли спілкуватися, їм потрібно двом знати лише один ключ. Концепт шифру представленого у прикладі називається «шифром Цезаря». Ще за багато сторіч до нас, генерал Гай Юлій Цезар використовував його, для безпечної передачі інформації.

Розглянувши приклад, можна дійти до висновку, що симетричне шифрування є дуже простим, через те, що для шифрування і дешифрування використовується один ключ. Цей вид за часту використовують, коли працюють з великим об’ємом даних. Підсумовуючи можна сказати, що симетричне шифрування, набагато швидше ніж асиметричне(яке ми розглянемо у подальшому). Через свою простоту не потребує великої обчислювальної потужності та не знижує швидкість передачі даних.

Для ліпшого розуміння подальшого матеріалу відносно симетричних алгоритмів шифрування розглянемо деякі поняття. Симетричних шифри використовують комбінації з перестановок та підстановок. Кожна така комбінація називається прохід. Під час такого проходу шифри використовують прохідні ключі. Множиною таких ключів називають розклад ключів.

Типовим способом формування алгоритмів симетричного шифрування є мережа Фейстеля. За допомогою цієї мережі алгоритм будує схему шифрування на основі функції F(D,K), де D – дані в 2 рази менші за блок шифрування, K – ключ проходу для цього проходу. Повне співпадіння шифрування з дешифруванням – це одна з переваг мережі Фейстеля, що є великим плюсом для програмної реалізації.

Операція перестановки, яка усюди використовується в симетричному шифруванні, змішує біти інформації, по визначеному закону. Саме ці багаточисленні проходи і реалізують ефект сніжного кому, який з кожним виконаним проходом шифрує інформацію все більше і більше. Ці операції перестановки зазвичай лінійні, тому можна сказати що f(a) xor f(b) == f(a xor b).

В цей час операції підстановки виконуються як заміна значення частини повідомлення, на інтегроване в алгоритм, стандартне число. Саме таким чином ці операцій додають в алгоритм нелінійність.

Зазвичай, від вибору значень в таблицях підстановки (S- блоках) дуже сильно залежить стійкість алгоритму до криптоаналізу. Наявність нерухомих елементів S(x)=x як і відсутність впливу біту вхідного байту на біт результату є небажаним.

На даний момент, можна сказати, що сучасні симетричні шифри поділяються на два види:

- Блокові
- Потокові

Блокові шифри

Блокові шифри оброблюють данні спеціальними блоками заданої довжини (64, 128 біт, у залежності від алгоритму), застосовуючи певний ключ, до певного блоку в встановленому порядку. Як правило, це відбувається декількома циклами перемішування. Ці цикли називають раундами. Результатом є ефект сніжного кому – чим більша кількість раундів, тим більше втрачається схожість відкритих бітів з зашифрованою інформацією. Розглянемо деяких представників блокового шифрування.

DES

DES (data encryption standard), був презентований у 80-тих роках в Америці. Цей шифр є найстаршим і найстарішим з симетричних методів шифрування. Розробили його для захисту конфіденційних даних влади США, та через рік після презентації став офіційним стандартом шифрування федеральних агентств країни.

Засилаючись на веб-ресурс instagalleryapp можна сказати, що алгоритм DES виконує шифрування 64-бітних блоків інформації за допомогою 54-бітного ключа та мережі Фейстеля. Спочатку блок інформації поділяється на 2 частини L та R, кожна з яких займає 32 біти. Далі відбувається шифрування протягом 16 циклів. Під час яких біти ключа здвигаються і з 56 бітів ключа обирається 48.

Права частина блоку даних збільшується до 48 бітів, за допомогою перестановки з розширенням та об’єднується з 48 бітами зміщеного ключа, проходить крізь 8 блоків утворюючи нових 32 біта і знову виконує перестановку. Усі ці операції виконуються у f функції, після чого, результат f функції лівої частини та результат f функції правої частини об’єднуються таким же чином як і біти у самій функції, за допомогою перестановки з розширенням.

Таким чином з’являється нова права частина, а стара стає новою лівою. Ці дії повторюються 16 разів утворюючи 16 етапів DES. Рисунок 1 більш наглядно зможе продемонструвати алгоритм роботи шифру[5]

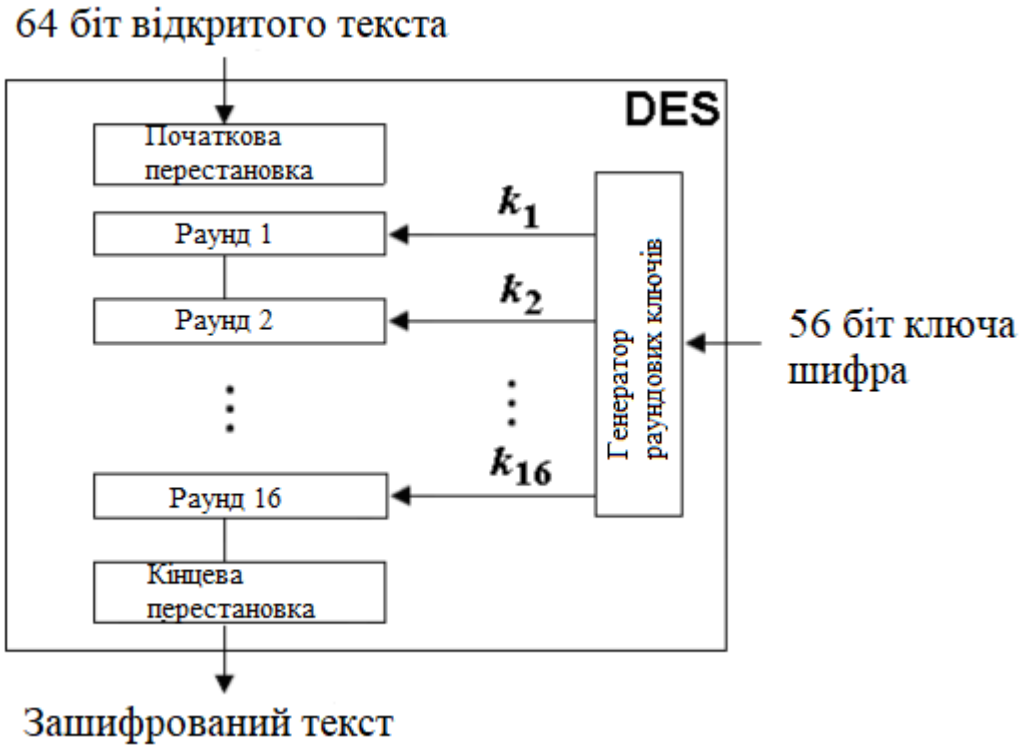


Рисунок 2.1-Алгоритм роботи шифру DES

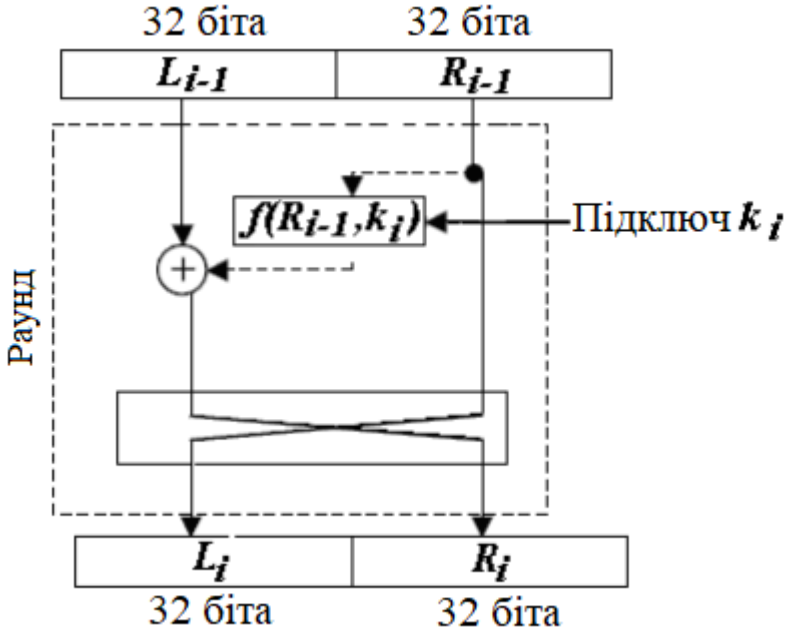


Рисунок 2.2 – Раунд шифрування DES

Перевагою алгоритму DES є єдиний ключ, як для шифрування, так і для дешифрування, що є дуже важливим аспектом при програмній реалізації. Цей метод шифрування також має свої недоліки такі як:

- Розмір ключа. Найбільшою проблемою для DES є маленький розмір ключа, що дорівнюється 56 біт. Цей аспект є недопустимим при сучасних обчислювальних можливостях. Оскільки даний ключ можна зламати за невеликий період часу звичайним силовим методом повного перебору (Brute-Force).
- Слабкість ключів шифрування. Раундові ключі створюють проблему тим, що під час ділення ключів на частини, та їх заміни можуть з'являтися однакові результати. Таке трапляється в тих випадках, коли ключі мають безперервні 1 або 0. Наслідком такого є використання одних і тих же ключів під час усіх раундів.
- Однакові результати шифрування. На виході зашифрована інформація може бути однакова для різних вхідних даних.

Через усі свої недоліки у свій час це й алгоритм був успішно замінений на іншого представника блочних шифрів а саме на удосконалену версію самого себе – 3DES. Суть цього алгоритму була проста - DES виконувався 3 рази, що значило, що і розмір ключа збільшується в 3 рази. Здавалося б, що недолік з довжиною ключа був вирішне, але навіть цього було недостатньо, тому на зміну цьому алгоритму прийшов новий AES.

AES

AES(advanced encryption system) ще відомий як Rijndael – це окремий представник симетричних блокових шифрів, який був розроблений урядом США для захисту секретної державної інформації. У 1997 році національний інститут стандартів та технологій оголосив шифр DES небезпечним, через виявлену вразливість до атаки грубою силою. В цей же рік почалась розробка шифру AES, як альтернативу вже застарілому DES.

Особливості AES такі:

- Симетричний ключ симетричний блок-шифр

- 128-бітні дані, 128/192/256-бітні ключі
- Сильніший і швидший, ніж Triple-DES
- Надайте повну специфікацію та деталі дизайну
- Програмне забезпечення, реалізоване на C та Java
- Функціонування AES

Спираючись на інформацію отриману веб-ресурсом tuturialspoint, AES – це представник ітераційних шифрів, він не відноситься до шифрів Фейстеля. Сам алгоритм представлений деяким рядом підстановочних або перестановочних операцій пов'язаних між собою. На відміну від інших алгоритмів, AES використовує байти а не біти, отже відкритий текст приймається не в розмірі 128 бітів а як 16 байт інформації. Для подальшої обробки, вхідна інформація завжди розташована в матриці. Потрібно зауважити, що алгоритм AES не має сталої кількості раундів, які він виконує при шифруванні. Їх кількість напряму залежить від довжини вхідного ключа. Для 128- розрядного ключа виконується 10 раундів, для 192-розрядного – 12 раундів, а для 256-розрядного – 14 раундів.[6] Процес шифрування в AES відбувається, як і усі блочні шифри, за допомогою раундів. Кожен раунд складається з чотирьох підпроцесів.(Рисунок 3)

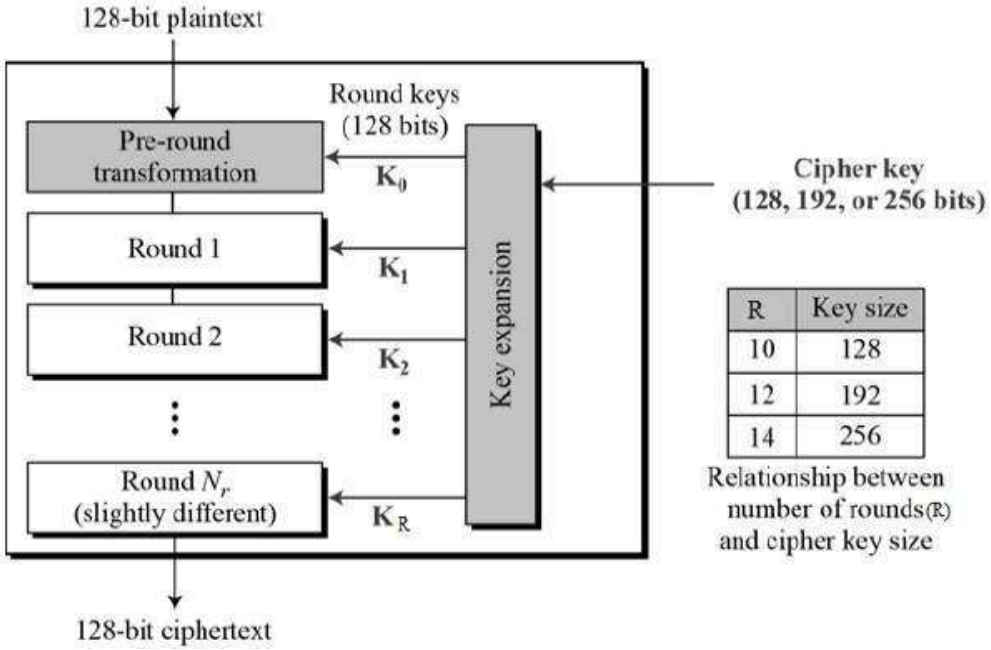


Рисунок 2.3- Схема роботи алгоритму AES

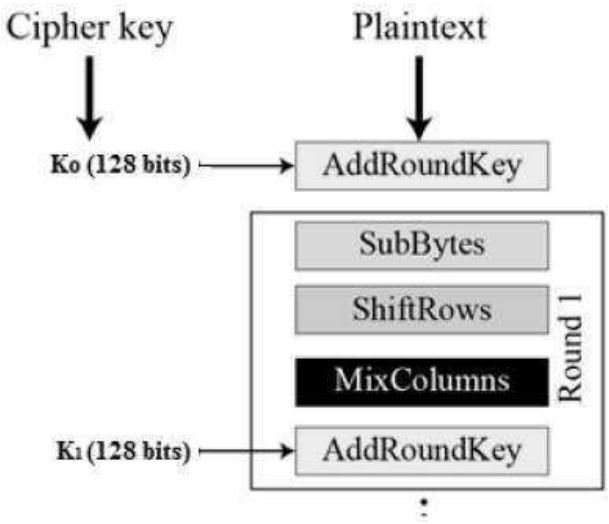


Рисунок 2.4- Схема раунду алгоритму AES

Першим підпроцесом є заміна байтів (SubBytes) під час якої, 16 вхідних байтів замінюються пошуком у фіксованій таблиці (S-box), заданої спочатку. Результатом є матриця з чотирьох рядків і чотирьох стовпців. Наступним етапом є зміщення (ShiftRows) кожного з чотирьох рядків матриці вліво. Будь-які записи, які «відпадають», виходять за рамки масиву, повторно вставляються в праву частину рядка. Зміна чотирьох рядків здійснюється наступним чином:

- Перший ряд не зміщується.
- Другий рядок зміщується на одне (байтове) положення вліво.
- Третій ряд зміщається на дві позиції вліво.
- Четвертий ряд зміщається на три позиції вліво.

Результатом є нова матриця, що складається з тих самих 16 байт, але зміщених відносно один одного. Після цього відбувається переміщування стовбців (MixColumns) Кожен стовпець із чотирьох байтів тепер трансформується за допомогою спеціальної математичної функції. Ця функція бере вхідні дані чотири байти одного стовпця і виводить чотири абсолютно нові байти, які замінюють вихідний стовпець. Результатом є ще одна нова матриця, що складається з 16 нових байт. Слід зазначити, що цей етап не виконується в останньому раунді. І останнім процесом є додавання ключа (AddRoundKey). 16 байт матриці тепер розглядаються як 128 біт і XOR до 128 бітів круглого ключа.

Якщо це останній раунд, то на виході буде зашифрований текст. В іншому випадку отримані 128 біт інтерпретуються як 16 байт, і ми починаємо ще один подібний раунд.

На відміну від усіх шифрів, які базуються на шифрі Фейстеля, дешифрування AES відбувається за допомогою окремого виконання усіх алгоритмів. Але постає питання, чим відрізняється шифрування від дешифрування? При шифруванні усі раунди, усі 4 процеси цих раундів виконуються у звичайному порядку, а при дешифруванні в зворотному.

Потокові шифри

Потоковий шифр - це вид симетричних шифрів, де відкриті дані поєднуються з псевдовипадковим потоком ключів. У потоковому шифрі кожна відкрита цифра зашифровується з відповідною цифрою потоку ключів, щоб отримати цифру потоку шифротексту. На практиці цифра, як правило, є бітом, а операція комбінування є ексклюзивним. Потокові шифри призначені для наближення до ідеалізованого шифру, відомого як One-Time Pad (схема одноразових блокнотів). Ця схема щоразу повинна використовувати випадковий ключ, який потенційно може досягти "ідеальної секретності". Тобто він повинен бути повністю захищений від атаки перебором. Проблема схема одноразових блокнотів полягає в тому, що при створенні такого шифру його ключ повинен бути таким же або навіть довшим, аніж відкритий текст. Іншими словами, якщо є 100-мегабайтний відео файл, котрий потрібно зашифрувати, то ключ буде не менше 800 мегабайт.

Потік псевдовипадкових ключів, як правило, генерується послідовно із випадкового значення за допомогою цифрових регістрів зсуву. Початкове

значення служить криптографічним ключем для дешифрування потоку зашифрованого тексту. Потокові шифри, як правило, виконують завдання з більшою швидкістю, ніж блок-шифри, і мають меншу апаратну складність.

Однак потокові шифри можуть спричинити серйозні проблеми з безпекою, якщо їх використовувати неправильно, зокрема, той самий вихідний стан

ніколи не можна використовувати двічі. Для ліпшого розуміння схеми роботи потокових алгоритмів, розглянемо один з найвідоміших шифрів – RC4.

RC4

Алгоритм RC4 базується на параметризованому генераторі псевдовипадкових бітів з рівномірним розподілом. Ключ може бути в довжину від 40 до 256 біт. Увесь алгоритм можна поділити на декілька головних частин. Перша це генератор ключового потоку.

Ядром цього генератора є звичайна функція, яка генерує послідовність бітів, котра у подальшому об’єднується з інформацією для шифрування. Це об’єднання відбувається за допомогою додавання по модулю два. Якщо розглянути генератор більш детально, то можна сказати, що сам генератор виробляє значення, які зберігаються в S-блоках, та кожен раз обирає деяке неоднакове значення S, як результат. В цей час, в кожному циклі цього RC4 визначається одне n-бітне число з ключового потоку, з котрим у подальшому буде відбуватися об’єднання з вхідною інформацією. Саме цю частину називають генератором псевдовипадкової послідовності. [7] Розшифрування відбувається схожим чином, але тепер додавання по модулю два застосовується до зашифрованого тексту та ключового потоку. Через властивості додавання по модулю на виході ми отримаємо вже розшифрований текст

Другою, важливою частиною RC4 є функція ініціалізації. Ця функція потрібна для утворення початкового стану ключового потоку, за допомогою деякого ключа, деякої довжини. Алгоритм ініціалізації, використовує деякий ключ, збережений в KEY та маючий довжину L. Сама ініціалізація починається с заповнення масиву S, після чого, цей масив змінюють, за допомогою перестановок, котрі визначаються ключем. Через те, що над S виконується лише одна дія, то кожен раз відбувається процес підтвердження щодо того, що S завжди містить кодове слово.

На відміну від інших шифрів, RC4 не використовує окрему подію як ключ.

Це значить, що при довгорічному використанні однакового ключа для декількох потоків, сама система RC4 повинна комбінувати подію та цей ключ

для генерації потокового ключа. Це є проблемою, але є варіанти вирішення. Одним з варіантів є генерація нового ключа за допомогою хеш функції, але багато додатків, які використовують RC4 просто конкатенують подію та ключ. Саме через це у додатку з'являються вразливі місця.

2.2. Асиметричне шифрування

Другою, дуже великою групою криптографічних алгоритмів є асиметричне шифрування. У цьому виді шифрів використовується декілька ключів, перший для шифрування, другий для дешифровки. Перший з них відомий як відкритий, через те що доступний для всіх, а другий – закритий, через те що до нього має доступ лише одна людина. Асиметричне шифрування також відоме як шифрування за допомогою відкритого ключа.

Якщо повертатись до нашого простого прикладу з двома співрозмовниками, то можна сказати, що метод симетричного шифрування, під час листування на відстані, працює гарно. Але що якщо наші співрозмовники бажають спілкуватися захищено і з іншими людьми, їх багато. Постає питання, щодо зберігання великої кількості ключів. Використання та зберігання різних ключів для кожного співрозмовника є дуже непрактичним і некомфортним. Для вирішення цієї проблеми співрозмовники починають використовувати шифрування з відкритим ключем. Співрозмовник віддає відкритий ключ кожному, хто надсилає йому інформацію, а секретний ключ зберігає у себе. За допомогою відкритого ключа інформація шифрується, а співрозмовник маючи свій закритий ключ, розшифровує отриману інформацію.

Це робить неможливим компрометацію ключа, через те що інформація може бути розшифрована закритим ключем співрозмовника.

Найбільш очевидна особливість та перевага цього методу - це безпека, яку забезпечує асиметричне шифрування. Безпека даних обумовлена наявністю двох ключів, котрі використовується для шифрування та дешифрування інформації.

Відкритий ключ дозволяє усім шифрувати інформацію, але розшифрувати її, може лиш той, хто володіє закритим ключем. Таким чином асиметричне шифрування гарантує захист даних від атак «людини посередині». Ця особливість є важливою для роботи веб-серверів та серверів пошти, тому що криптографія з відкритим ключем дозволяє створювати шифроване з'єднання в автономному режимі без обов'язкового обміну ключами. Другою важливою особливістю є автентифікація. Те що інформація може бути розшифрована тільки закритим ключем гарантує захищену автентифікацію (те що розшифрувати і прочитати інформацію може тільки той об'єкт, котрий повинен отримати зашифровану інформацію). Це дає впевненість у тому, що спілкування відбувається саме з потрібною особою.

Для більш глибокого розуміння теми асиметричних шифрів, буде розглянутий один з найвідоміших шифрів цього типу – RSA.

RSA

RSA був винайдений 1977 році трьома вченими Роном Рівсетом, Аді Шамиром та Леонардом Адельманом. Алгоритм має своє ім'я завдяки своїм творцям (R-Rivest, S-Shamir, A-Adleman). Алгоритм RSA є найпопулярнішим з усіх асиметричних криптографічних шифрів. Його ефективність базується на дуже надійному методі «первинної факторизації». Суть цього методу полягає у тому, що спочатку обирається два випадкових простих числа і перемножуються між собою для отримання величезного вхідного числа. Це робиться через те, що на даний момент задача визначення простих чисел по результату їх множення є майже нереальною, як для комп'ютерного розуму, так і для людського. У 2010 році проводились досліди, щодо визначення часу вирішення цієї задачі. Було виявлено, що для того щоб зламати 768-рійний ключ, котрий базується на алгоритмі перемноження двох простих чисел, потрібно витратити більше ніж 1500 років обчислювального часу.

Розглянемо основні два аспекти криптосистеми RSA,:

- генерацію пари ключів
- алгоритми шифрування-розшифрування.

Розглянемо процес генерації пари ключів. Кожна особа або сторона, яка бере участь у спілкуванні за допомогою шифрування, створює пару ключів, а саме відкритий ключ і приватний ключ. Алгоритм генерування ключів складається з декількох етапів. Перше це формування модулю RSA (n). Спочатку обирається два великих простих числа, p і q, далі обчислюється n = p * q. Для сильного шифрування потрібно щоб n було великим числом, як правило, мінімум 512 біт. Другим етапом буде знаходження похідного числа (e). Число e має бути більше 1 і менше (p - 1) (q - 1), для e та (p - 1) (q - 1) не повинно бути спільного множника, крім 1. Іншими словами, два числа e та (p - 1) (q - 1) є спільними.

Третім етапом буде формування відкритого ключа. Пара чисел (n, e) утворює відкритий ключ RSA і робиться загальнодоступною. Хоча n є частиною відкритого ключа, труднощі з факторизацією великого простого числа гарантують, що зловмисник не може знайти два простих числа (p & q), які використовувались для отримання n. Це і є силою RSA. Четвертим етапом буде створення закритого ключа. Приватний ключ d обчислюється з p, q та e. Для даних n та e повинно існувати унікальне число d. Число d - обернене за модулем e (p - 1) (q - 1). Це означає, що d - число, менше (p - 1) (q - 1), таке, що, якщо його помножити на e, воно буде дорівнювати 1 за модулем (p - 1) (q - 1). Після створення пари ключів процес шифрування та дешифрування є відносно простим. RSA безпосередньо не працює на рядках бітів, як у випадку симетричного шифрування ключа. Він оперує числами за модулем n. Отже, необхідно представляти відкритий текст як ряд чисел, менших за n.

Розглянемо більш детально алгоритм шифрування і дешифрування. Припустимо, відправник хоче надіслати якесь текстове повідомлення комусь, чий відкритий ключ (n, e). Після відправник представляє відкритий текст у вигляді серії чисел, менших за n. Для шифрування першого відкритого тексту P, що є числом за модулем n. Процес шифрування - це простий математичний крок, оскільки $C = P^e \bmod n$. Іншими словами, зашифрований текст C дорівнює

відкритому тексту P, помноженому на себе e разів, а потім зменшеному за модулем n. Це означає, що C - це також число менше n.

Процес розшифрування RSA також дуже простий. Припустимо, що приймач пари відкритих ключів (n, e) отримав зашифрований текст C. Приймач піднімає C до рівня свого приватного ключа d. Результатом за модулем n буде відкритий текст P. Отже, відкритий текст = C^d mod n Безпека RSA залежить від сильних сторін двох окремих функцій.

Криптосистема RSA є найпопулярнішою криптосистемою з відкритим ключем, сила якої базується на практичній складності факторингу дуже великих чисел.

- Функція шифрування - вона розглядається як одностороння функція перетворення відкритого тексту в зашифрований текст, і вона може працювати в іншу сторону (у сторону дешифровки) лише за допомогою приватного ключа d
- Генерація ключа - складність визначення приватного ключа за відкритим ключем RSA еквівалентна факторингу модуля n. Таким чином, зловмисник не може використовувати знання відкритого ключа для визначення приватного ключа, якщо він не може врахувати n. Це також одностороння функція, перехід від значень p & q до модуля n є простим, але незворотним.

Якщо будь-яка з цих двох функцій буде неодносторонньою, RSA буде порушено. Насправді, якщо у подальшому буде розроблена ефективна методика факторингу, тоді RSA перестане бути найбезпечнішим. Також потрібно зауважити, що сила шифрування RSA різко падає, якщо число p і q не є великими простими числами та/або вибраний відкритий ключ e не є великим числом. [8]

2.3. Гібридне шифрування та аналіз методів шифрування

Важливо зазначити ще такий «метод шифрування» як гібридне. Цей вид не є окремим методом, через те, що такий тип шифрування просто поєднує в собі

два методи: симетричне та асиметричне. Таким чином гібридне шифрування поєднує в собі переваги двох методів і створює надійну синергію двох систем криптографії.

Кожен з розглянутих видів шифрування, як асиметричне, так і асиметричне має свої недоліки. Наприклад симетричне шифрування має велику швидкість шифрування, але не має перевірки особистості, що є дуже критичним для використання цього алгоритму в Інтернеті. Асиметричне шифрування у свою чергу має цю перевірку, але швидкість зашифровки та перевірки особистості залишає бажати кращого. Отже, для того щоб нівелювати недоліки кожного алгоритму було і створено гібридне шифрування. Цей метод дуже активно використовується в SSL/TLS сертифікатах. Алгоритм його роботи полягає у декількох пунктах. Спочатку, за допомогою приватного ключа, відбувається перевірка особистості з обох сторін. Після цієї перевірки відбувається шифрування даних за допомогою ефемерного, сеансового (симетричного) ключа, що забезпечує дуже швидку передачу великої кількості даних.

Симетричне шифрування	Асиметричне шифрування
Використання одного ключу для шифрування та дешифрування	Використання двох різних ключів для шифрування та дешифрування
Велика швидкість шифрування, через використання єдиного ключу	Невелика швидкість шифрування, через використання різних ключів
Переважно використовується для шифрування великої кількості інформації	Переважно використовується у випадках , коли потрібна автентифікація особи
Забезпечує велику продуктивність, потребує невелику обчислювальну потужність через простий алгоритм шифрування.	Має меншу продуктивність та потребує більше потужності обчислювання через більш складні алгоритми шифрування
Для шифрування використовує ключи невеликої довжини (128-256)	Для шифрування використовує ключи великої довжини (1024-4096)

Після розгляду різних методів шифрування, таких як асиметричне та симетричне, та більш детального аналізу, можна зробити висновок, що найбільш оптимальний метод шифрування, який буде використовуватися для написання програми – це методом є асиметричний блочний шифр AES. Порівняльна таблиця методів шифрування показує у яких аспектах, один або інший метод краще або гірше.

2.4. Опис програмного середовища

JavaScript - це крос-платформна, об’єктно-орієнтована мова сценаріїв, яка використовується для створення інтерактивних веб-сторінок. Існують також більш вдосконалені версії JavaScript на стороні сервера, такі як Node.js. Вони дозволяють додавати веб-сайту більше функціональних можливостей, більше ніж звичайне завантаження файлів (наприклад, співпраця в режимі реального часу між кількома комп'ютерами). Всередині хост-середовища (наприклад, веб-браузера) JavaScript можна підключитись до об’єктів свого середовища, щоб забезпечити програмний контроль над ними. JavaScript містить стандартну бібліотеку об’єктів, таких як Array, Date та Math, і основний набір мовних елементів, таких як структури управління та оператори [9]. Саме за допомогою цієї мови я зміг розробити backend програми. Для шифрування я використовував бібліотеку CryptoJS. CryptoJS - це колекція стандартних та безпечних криптографічних алгоритмів, реалізованих у JavaScript з використанням найкращих практик та шаблонів.[10]

Для frontend використовується HTML5. HTML5 - це мова розмітки, яка використовується для структурування та представлення вмісту веб-сторінок. Поточна специфікація відома як HTML Living Standard. Вона підтримується основними браузерами від Apple, Google, Mozilla та Microsoft, робочою групою технологій веб-гіпертехнологічних додатків (WHATWG). Також, для візуальної презентації веб-сторінок програми, використовується CSS. А всі ці можливості об'єднані в одному середовище програмування IntelliJ Idea.

Розроблений додаток може обширно використовуватись для шифрованої передачі даних. Гарним прикладом використання може бути офіс, де є потреба

шифрування даних під час передачі інформації. При такому типі обміну інформації зловмисники, котрі можуть підключитися до каналу передачі даних цього офісу, зможуть перехопити лише зашифровані дані.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ З ВИКОРИСТАННЯМ КРИПТОГРАФІЧНОГО МЕТОДУ ШИФРУВАННЯ ДАНИХ AES

3.1. Програмна реалізація

При написанні програми виникла проблема з некоректністю роботи атрибуту download в HTML5. При завантаженні великих за об’ємом файлів поточна вкладка в браузері Google Chrome зависала, при використанні Firefox припинявся весь процес браузера.

Для вирішення цієї проблеми можна було б використати File System API с записом двійкових даних у нього, але на даний час цей додаток підтримується тільки в Google Chrome. Таке рішення проблеми не було вирішене, вибір був зроблений в бік крос-платформеності, тому ліміт об’єму файлів дорівнює 1 Мб.

3.1.1. HTML

Першим етапом написання всього веб додатку була розмітка всіх сторінок програми в файлі index.html

Для початку, у <head> сторінки, в блоці <title> задаємо назву вкладки JavaCrypt, назва під якою буде відкриватися сторінка нашого додатку. Також у <head> сторінки підключаємо усі бібліотеки, які будемо використовувати в подальшому. Код всіх цих кроків зображений на рисунку 3.1

```
<head>
    <meta charset="utf-8"/>
    <title>JavaCrypt</title>

    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link href="http://fonts.googleapis.com/css?family=Raleway:400,700" rel="stylesheet" />
    <link href="assets/css/style.css" rel="stylesheet" />

</head>
```

Рисунок 3.1 - <head> додатку JavaCrypt

Наступним етапом розмітки додатку буде написання його основної частини <body>. Ця частина буде містити в собі декілька різних елементів.

Перший елемент – це клас для кнопки повернення назад, котра буде з’являтися на сторінці після того, як користувач перейшов на нову сторінку, яка не є першою. Цей етап відображений на рисунку 3.2

```
<a class="back"></a>
```

Рисунок 3.2 – Створення класу для кнопки повернення назад

Другим елементом <body> буде під назвою stage , котрий містить у собі ще декілька , які відповідають за розмітку сторінок додатку. Деякі з сторінок будуть містити в собі конструкцію if. Ця конструкція допомагає у досяганні варіативності під час переходу на нову сторінку(користувач переходить на нову сторінку, в залежності . Цей етап відображений на рисунку 3.3.

```
<body>

    <a class="back"></a>

    <div id="stage">

        <div id="Win1">
            <div class="content">
                <h1>Choose what you want to do with your file</h1>
                <a class="b encrypt">Encrypt a file</a>
                <a class="b decrypt">Decrypt a file</a>
            </div>
        </div>

        <div id="Win2">

            <div class="content if-encrypt">
                <h1>Choose your file</h1>
                <h2>An encrypted copy of the file will be generated. File size must be less than 1 mb</h2>
                <a class="b browse">Browse</a>

                <input type="file" id="encrypt-input" />
            </div>

            <div class="content if-decrypt">
                <h1>Choose your file for decryption</h1>
                <h2>Only files encrypted by this tool are possible.</h2>
                <a class="b browse">Browse</a>

                <input type="file" id="decrypt-input" />
            </div>
```

```
<div id="Win3">

    <div class="content if-encrypt">
        <h1>Create a password for your encryption</h1>
        <h2>Remember your password, you won't be able to restore the file without it. </h2>

        <input type="password" />
        <a class="b process">Encrypt</a>
    </div>

    <div class="content if-decrypt">
        <h1>Enter the pass phrase</h1>
        <h2>Enter the pass phrase that was used to encrypt this file.</h2>

        <input type="password" />
        <a class="b process">Decrypt</a>
    </div>

</div>

<div id="Win4">

    <div class="content">
        <h1>Your file is ready</h1>
        <a class="b download">Download</a>
    </div>

</div>
</div>
```

Рисунок 3.3 – Створення під назвою stage для подальшої розмітки сторінок

Розглянемо більш детально кожен з сторінок.

Для першої сторінки буде мати такий код, як зображено на рисунку 3.4. У своєму лістингу він матиме виведення на екран запитання, щодо подальших дій користувача, та дві кнопки з обиранням подальших дій.

```
<div id="Win1">
    <div class="content">
        <h1>Choose what you want to do with your file</h1>
        <a class="b encrypt">Encrypt a file</a>
        <a class="b decrypt">Decrypt a file</a>
    </div>
</div>
```

Рисунок 3.4 – Створення першої сторінки

Для другої сторінки буде мати такий код як зображено на рисунку 3.5. у своєму лістингу він матиме два які будуть виконуватись після вибору користувача на першій сторінці.

Перший , який відкривається після натискання першої кнопки (encrypt), має такий лістинг: виведення на екран інструкції, щодо подальших дій користувача та виведення попередження, щодо розміру файлу, та саму кнопку, після натискання якої з'явиться контекстне меню з вибором файлу для шифрування.

Другий , який відкривається після натискання другої кнопки (decrypt), має такий лістинг: виведення на екран інструкції, щодо подальших дій користувача та виведення попередження, що дешифруванню підлягають тільки ті файли, які були зашифровані цією програмою, та саму кнопку, після натискання якої з'явиться контекстне меню з вибором файлу для дешифрування.

```
<div id="Win2">

    <div class="content if-encrypt">
        <h1>Choose your file</h1>
        <h2>An encrypted copy of the file will be generated. File size must be less than 1 mb</h2>
        <a class="b browse">Browse</a>

        <input type="file" id="encrypt-input" />
    </div>

    <div class="content if-decrypt">
        <h1>Choose your file for decryption</h1>
        <h2>Only files encrypted by this tool are possible.</h2>
        <a class="b browse">Browse</a>

        <input type="file" id="decrypt-input" />
    </div>

</div>
```

Рисунок 3.5 – Створення другої сторінки

Для третьої сторінки буде мати такий код як зображено на рисунку 3.6. у своєму лістингу він матиме два які будуть виконуватись в залежності від попередньої сторінки.

Перший , буде виконуватись у разі, якщо на попередній сторінці виконувався перший (шифрування). Він матиме такий лістинг: виведення

на екран інструкції, щодо подальших дій користувача та створення паролю для шифрування файлу, обраного на попередній сторінці.

Другий , буде виконуватись у разі, якщо на попередній сторінці виконувався другий (дешифрування). Він матиме такий лістинг: виведення на екран інструкції, щодо подальших дій користувача та введення паролю для дешифрування файлу, обраного на попередній сторінці.

```
<div id="Win3">

    <div class="content if-encrypt">
        <h1>Create a password for your encryption</h1>
        <h2>Remember your password, you won't be able to restore the file without it. </h2>

        <input type="password" />
        <a class="b process">Encrypt</a>
    </div>

    <div class="content if-decrypt">
        <h1>Enter the pass phrase</h1>
        <h2>Enter the pass phrase that was used to encrypt this file.</h2>

        <input type="password" />
        <a class="b process">Decrypt</a>
    </div>

</div>
```

Рисунок 3.6 – Створення третьої сторінки

Для четвертої сторінки буде мати такий код як зображено на рисунку 3.7. Він буде відкриватися тільки в тому випадку, коли над файлом здійснюється шифрування. У своєму лістингу він матиме. інструкції, щодо подальших дій користувача та кнопку загрузки зашифрованого файлу.

```
<div id="Win4">

    <div class="content">
        <h1>Your file is ready</h1>
        <a class="b download">Download</a>
    </div>

</div>
```

Рисунок 3.7 – Створення четвертої сторінки

Останнім елементом коду в файлі index.html буде підключення скриптів, від яких залежить функціонал всього додатку. Лістинг цього блоку буде мати такий вид, як зображено на рисунку 3.8.

```
<!-- Include the AES algorithm of the crypto library -->
<script src="assets/js/aes.js"></script>
<script src="http://cdnjs.cloudflare.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
<script src="assets/js/script.js"></script>
```

Рисунок 3.8 – Підключення скриптів

3.1.2. CSS

Більш детально розглянемо файл CSS, який відповідає за візуальне оформлення усіх елементів мого веб-додатку.

Перших два блоку коду відповідають за скидання усіх попередніх налаштувань, та задання нових для усього документу. Лістинг коду цих блоків зображено на рисунку 3.9.

```
*{
    margin:0;
    padding:0;
}


/*--------------------------
    Общие настройк стилей
---------------------------*/


html{
    overflow:hidden;
}

a, a:visited {
    outline:none;
    color:#FFE4E1;
}

a:hover{
    text-decoration:none;
}

section, header, aside{
    display: block;
}
```

Рисунок 3.9 – Блоки коду, що відповідають за загальні налаштування для усіх сторінок

Наступний блок коду відповідає за вид елементів головної сторінки. Лістинг цього блоку зображений на рисунку 3.10.

```
body{
    font:15px/1.3 'Raleway', sans-serif;
    color: #A0522D;
    width:100%;
    height:100%;
    position:absolute;
    overflow:hidden;
}

#stage{
    width:100%;
    height:100%;
    position:absolute;
    top:0;
    left:0;

    transition:top 0.4s;
}

#stage > div{
    height:100%;
    position:relative;
}


#stage h1{
    font-weight:normal;
    font-size:48px;
    text-align:center;
    color:#A0522D;
    margin-bottom:60px;
}
```

```
#stage h2{
    font-weight: normal;
    font-size: 14px;
    font-family: Arial, Helvetica, sans-serif;
    margin: -40px 0 45px;
    font-style: bold;
}
.content{
    position:absolute;
    text-align:center;
    left:0;
    top:50%;
    width:100%;
}

.content input[type=file]{
    display:none;
}

a.back{
    width: 32px;
    height: 32px;
    background: url('../img/icons.svg') no-repeat 94% -3px;
    position: absolute;
    cursor: pointer;
    top: 50px;
    left: 50%;
    z-index: 10;
    opacity: 0.8;
    margin-left: -16px;
    display:none;
}
a.back:hover{
    opacity:1;
}
```

Рисунок 3.10 – Блок коду, що відповідає за вид елементів головної сторінки

Наступний блок коду взаємодіє з умовними класами та розкриває функцію if, яка присутня у більшості блоків коду сторінок у html файлі. Лістинг цього блоку зображений на рисунку 3.11

```
[class*="if-"]{
    display:none;
}

body.encrypt .if-encrypt{
    display:block;
}

body.decrypt .if-decrypt{
    display:block;
}
```

Рисунок 3.11 – Блок коду, що відповідає за умовні класи

Наступний блок коду відповідає за оформлення звичайних кнопок, котрі використовуються у всьому файлі. Лістинг цього блоку зображений на рисунку 3.12

```
.b{
    width:240px;
    height:70px;
    text-align:center;
    text-decoration: none !important;
    color:#FFEBCD !important;
    text-transform: uppercase;
    font-weight: bold;
    border-radius:1px;
    display:block;
    line-height:70px;
    box-shadow:3px 3px 0 rgba(0,0,0,0.08);
    cursor: pointer;
    font-size:18px;
    margin:10px auto;
    opacity:0.9;
    background-color:#5B5C61;
}
 .b:hover{
    opacity:3;
 }
.b::before{
    content:'';
    background: url(../img/icons.svg) no-repeat;
    display: inline-block;
    width: 32px;
    height: 32px;
    vertical-align: middle;
    padding-right: 13px;
}
 .b{
    background-color:#A0522D;
 }
```

Рисунок 3.12 – Блок коду, що відповідає за вид кнопок

Цей блок відповідає за зовнішній вигляд спеціальних кнопок, які відповідають за функції, які виконуються за допомогою скриптів. Лістинг цього блоку зображений на рисунку 3.13

```
.b.browse{
    width:180px;
}

.b.process{
    width:190px;
}

.b.download{
    width:216px;
}


.b.decrypt::before{
    background-position: 18% -5px;
}

.b.browse::before{
    background-position: 40% -4px;
}

.b.process::before{
    background-position: 60% -3px;
}

.b.download:before{
    background-position: 80% -3px;
}
```

Рисунок 3.13 – Блок коду, що відповідає за вид кнопок, що використовують скрипти

Останнім блоком коду є код, який відповідає за загальний зовнішній вигляд усіх веб-сторінок, котрі використовуються у додатку. Лістинг цього блоку зображений на рисунку 3.14

```
#Win1{
    background-color:#FFEBCD;
}
body.encrypt #Win2{
    background-color: #FFEBCD;
}
body.decrypt #Win2{
    background-color: #FFEBCD;
}
#Win3{
    background-color:#FFEBCD;
}
#Win3 input[type=password]{
    background-color: #fff;
    border: none;
    padding: 8px 18px;
    line-height: 1;
    font: inherit;
    display: inline-block;
    outline: none;
    width: 400px;
    margin-bottom: 18px;
    border-radius: 2px;
    box-shadow: 3px 3px 0 rgba(0,0,0,0.05);
    font-size: 36px;
    color: #555;
}
#Win4{
    background-color:#FFEBCD;
}
#Win1 .content{ margin-top: -140px;}
#Win2 .content{ margin-top:-110px;}
#Win3 .content{ margin-top: -157px;}
#Win4 .content{ margin-top: -100px;}
```

Рисунок 3.14 – Блок коду, що відповідає за загальний вид усіх сторінок

3.1.3. SVG

Наступним елементом у моєму проекті є файл з розширенням SVG. Цей файл відповідає за іконки, які використовувалися для оформлення кнопок. Сам файл являє собою код, який складається з декількох частин.

Перша частина файлу, це код, який підключає цей файл до усього проекту, та задає дяку площину на якій будуть знаходитись самі іконки. Лістинг коду зображений на рисунку 3.15

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="272" height="35" viewBox="0 0 308 30">
```

Рисунок 3.15 – Блок коду, з підключенням файлу

Друга ж частина складається з декількох блоків, які відповідають за вигляд іконок, а в свою чергу іконки складаються з двох рядків. Перший рядок відповідає за положення елементу на площині, яку ми задавали у першому блоці.

Другий рядок – це код, який конструює іконку.

Лістинг коду першої іконки <g transform="translate(0 -4)"> <path style="" fill="#FFEBCD" stroke="none" d="M23.5 14h-0.5v-6c0- 3.308-2.692-6-6-6h-4c-3.308 0-6 2.692-6 6v6h-0.5c-0.825 0-1.5 0.675-1.5 1.5v15c0 0.825 0.675 1.5 1.5 1.5h17c0.825 0 1.5-0.675 1.5-1.5v-15c0-0.825-0.675-1.5-1.5- 1.5zM11 8c0-1.103 0.897-2 2-2h4c1.103 0 2 0.897 2 2v6h-8v-6z"></path>

</g>

ЇЇ вигляд в проекті зображений на рисунку 3.16

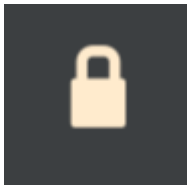


Рисунок 3.16 – Зовнішній вигляд першої іконки

Лістинг коду другої іконки <g transform="translate(50 0)">

<path style="" fill="#FFEBCD" stroke="none" d="M24 2c3.308 0 6 2.692 6 6v6h-4v-6c0-1.103-0.897-2-2-2h-4c-1.103 0-2 0.897-2 2v6h0.5c0.825 0 1.5 0.675 1.5 1.5v15c0 0.825-0.675 1.5-1.5 1.5h-17c-0.825 0-1.5-0.675-1.5-1.5v-15c0- 0.825 0.675-1.5 1.5-1.5h12.5v-6c0-3.308 2.692-6 6-6h4z"></path>

</g>

ЇЇ вигляд в проекті зображений на рисунку 3.17



Рисунок 3.17 – Зовнішній вигляд другої іконки

Лістинг коду третьої іконки <g transform="translate(100 0)">

<path style="" fill="#FFEBCD" stroke="none" d="M27 0h-24c- 1.65 0-3 1.35-3 3v26c0 1.65 1.35 3 3 3h24c1.65 0 3-1.35 3-3v-26c0-1.65-1.35-3-3- 3zM26 28h-

22v-24h22v24zM8 14h14v2h-14zM8 18h14v2h-14zM8 22h14v2h- 14zM8 10h14v2h-14z"></path>

</g>

ЇЇ вигляд в проекті зображений на рисунку 3.18



Рисунок 3.18 – Зовнішній вигляд третьої іконки

Лістинг коду четвертої іконки <g transform="translate(150 0)"> <path style="" fill="#FFEBCD" stroke="none" d="M30 0l-14 4-14-4c0 0-0.141 1.616 0 4l14 4.378 14-4.378c0.141-2.384 0-4 0-4zM2.256 6.097c0.75 7.834 3.547 21.007 13.744 25.903 10.197-4.896 12.995-18.069 13.744-25.903l-13.744 5.167-13.744-5.167z"></path>

</g>

ЇЇ вигляд в проекті зображений на рисунку 3.19



Рисунок 3.19 – Зовнішній вигляд четвертої іконки

Лістинг коду п’ятої іконки <g transform="translate(200 0)">

<path style="" fill="#FFEBCD" stroke="none" d="M16 31l15- 15h-9v-16h-12v16h-9z"></path>

</g>

ЇЇ вигляд в проекті зображений на рисунку 3.20



Рисунок 3.20 – Зовнішній вигляд п’ятої іконки

Лістинг коду шостої іконки. Ця іконка складається з двох елементів, тому і блоків з її описом два.

/g>

<g transform="translate(200 0)"> <path style="" fill="#FFEBCD" stroke="none" d="M16 31l15- 15h-9v-16h-12v16h-9z"></path>

</g>

<g transform="translate(250 0)"> <path style="" fill="#A0522D" stroke="none" d="M0 16c0 8.837 7.163 16 16 16s16-7.163 16-16-7.163-16-16-16-16 7.163-16 16zM29 16c0 7.18-5.82 13-13 13s-13-5.82-13-13 5.82-13 13-13 13 5.82 13 13z"></path> <path style="" fill="#A0522D" stroke="none" d="M22.086 20.914l2.829-2.829-8.914-8.914-8.914 8.914 2.828 2.828 6.086-6.086z"></path>

</g>

ЇЇ вигляд в проекті зображений на рисунку 3.21



Рисунок 3.21 – Зовнішній вигляд шостої іконки

3.1.4. JavaScript

Останніми файлами у проекті є два JavaScript файли. Перший файл, під назвою aes.js, є стандартною бібліотекою шифрування в JavaScript. Дана бібліотека використовується у другому файлі для шифрування та дешифрування файлів.

Другий файл, під назвою script.js, являє собою файл, який відповідає за всю програмну частину мого додатка. Розглянемо його більш детально. Перший блок файлу script.js відповідає за ініціалізацію змінних, які у подальшому будуть використовуватись для написання скриптів. Лістинг цього блоку виглядає так як зображено на рисунку 3.21

```
var body = $('body'),
    stage = $('#stage'),
    back = $('a.back');
```

Рисунок 3.21 – Ініціалізація змінних

Наступний блок відповідає за скрипти, які реалізуються на першій сторінці додатку. Перший скрипт - це скрипт, який виконує шифрування після натискання кнопки відповідальної за шифрування. Другий - це скрипт, який виконує дешифрування після натискання відповідної кнопки на інтерфейсі першої сторінки. Після виконання одного з скриптів, код преходить до наступного блоку (скрипта наступної сторінки). Лістинг блоку першої сторінки виглядає так як зображено на рисунку 3.22

```
$('#Win1 .encrypt').click(function(){
    body.attr('class', 'encrypt');


    Win(2);
});

$('#Win1 .decrypt').click(function(){
    body.attr('class', 'decrypt');
    Win(2);
});
```

Рисунок 3.22 – Блок коду для першої сторінки

Наступний блок коду відповідає за скрипти другої сторінки. В собі він містить два блоки які виконуються в залежності від вибору користувача в попередньому блоці скриптів. Якщо у попередньому блоці він обрав скрипт з шифруванням, то виконується перший блок цього блоку скриптів. Він в собі містить зчитування файлу для шифрування та перевірку його розміру. У протилежному випадку, якщо у попередньому блоці користувач обрав скрипт з дешифруванням, виконується другий блок цього блоку скриптів, який зчитує зашифрований файл. Лістинг блоку другої сторінки виглядає так як зображено на рисунку 3.23

```
$('#Win2 .b').click(function(){
    // нажатие кнопки для выбора файла
    $(this).parent().find('input').click();
});
// Действия для выбранного файла
var file = null;
$('#Win2').on('change', '#encrypt-input', function(e){

    // проверка на наличие файла

    if(e.target.files.length!=1){
        alert('Please select a file to encrypt!');
        return false;
    }

    file = e.target.files[0];

    if(file.size > 1024*1024){
        alert('Please choose files smaller than 1mb, otherwise you may crash your browser. \nThis is a known issue. See the tutorial.');
        return;
    }
    Win(3);
});
$('#Win2').on('change', '#decrypt-input', function(e){

    if(e.target.files.length!=1){
        alert('Please select a file to decrypt!');
        return false;
    }
    file = e.target.files[0];
    Win(3);
});
```

Рисунок 3.23 – Блок коду для другої сторінки

Третім блоком коду є код, який відповідає за шифрування/дешифрування.

З самого початку відбувається запит ключа для шифрування файлу та перевірка його на кількість символів (кількість повинна бути більше 5 символів). Далі відкривається функція if(else), яка визначає що потрібно робити з зчитаним файлом. Якщо файл має позначку ‘encrypt’, то виконується перша частина функції, яка зашифровує файл звертаючись до бібліотеки шифрів AES. Після виконання цього процесу код переходить на четверту сторінку на якій можна завантажити зашифрований файл. У протилежному випадку, якщо файл не має позначку ‘encrypt’, виконується друга частина функції і файл розшифровується, знову ж таки звертаючись до бібліотеки шифрів AES. Як і в випадку з шифруванням, після виконання частини else, код переходить на четверту сторінку, де можна завантажити розшифрований файл. Лістинг блоку третьої сторінки (частина if) виглядає так як зображено на рисунку 3.24. Лістинг блоку третьої сторінки (частина else) виглядає так як зображено на рисунку 3.25

```
$('a.b.process').click(function(){
    var input = $(this).parent().find('input[type=password]'),
        a = $('#Win4 a.download'),
        password = input.val();

    input.val('');
    if(password.length<5){
        alert('Please choose a longer password!');
        return;
    }
    var reader = new FileReader();

    if(body.hasClass('encrypt')){

        // Шифрование файла

        reader.onload = function(e){

            // Использование CryptoJS library и AES шифр для шифрования файла содержащего e.target.result, с паролем

            var encrypted = CryptoJS.AES.encrypt(e.target.result, password);

            // Атрибут загрузки приведет к загрузке содержимого атрибута href при нажатии.
            // Атрибут загрузки также содержит имя файла, который предлагается для загрузки.

            a.attr('href', 'data:application/octet-stream,' + encrypted);
            a.attr('download', file.name + '.encrypted');
            Win(4);
        };
        reader.readAsDataURL(file);
    }
```

Рисунок 3.24 – Блок коду для третьої сторінки, частина if

```
    else {

        //Расшифровка

        reader.onload = function(e){

            var decrypted = CryptoJS.AES.decrypt(e.target.result, password)
                                .toString(CryptoJS.enc.Latin1);

            if(!/^data:/.test(decrypted)){
                alert("Invalid pass phrase or file! Please try again.");
                return false;
            }

            a.attr('href', decrypted);
            a.attr('download', file.name.replace('.encrypted',''));
            Win(4);
        };
        reader.readAsText(file);
    }
});
/* Кнопка назад*/
back.click(function(){

    // Повторно инициализировать скрытые входные файлы,
    // чтобы они не удерживали выбор с прошлого раза

    $('#Win2 input[type=file]').replaceWith(function(){
        return $(this).clone();
    });
    Win(1);
});
```

Рисунок 3.25 – Блок коду для третьої сторінки, частина else

Передостанній блок коду описує скрипт для функціонування кнопки «назад». Лістинг блоку скрипта для кнопки «назад» має вигляд (рис. 3.26)

```
back.click(function(){



    $('#Win2 input[type=file]').replaceWith(function(){
        return $(this).clone();
    });
    Win(1);
});
```

Рисунок 3.26 – Блок коду для кнопки «назад»

Останній блок коду призначений для допоміжної функції, що допомагає програмі коректно переходити поміж різних сторінок і виконувати потрібні скріпти. Лістинг блоку допоміжного скрипта виглядає так як зображено на рисунку 3.27

```
function Win(i){

    if(i == 1){
        back.fadeOut();
    }
    else{
        back.fadeIn();
    }



    stage.css('top',(-(i-1)*100)+'%');
}
```

Рисунок 3.27 – Блок коду допоміжного скрипта

ВИСНОВКИ

У результаті роботи був досліджений сучасний стан проблеми захисту інформації. Були досконально досліджені сучасні методи захисту інформації, проаналізовані їх переваги та недоліки. При дослідженні основну увагу приділено криптографічним методам. Детально були розглянуті два методи шифрування інформації: симетричний та асиметричний. Також детально було досліджено два види симетричних шифрів: блоковий та потоковий. Для кожного виду був розглянутий типовий представник свого методу шифрування: для симетричного, блокового – DES, AES; для симетричного, потокового - RC4; для асиметричного – RSA. Також був розглянутий підвид – гібридне шифрування. Для симетричного й асиметричного шифрування була складена порівняльна таблиця.

Для програмної реалізації був визначений алгоритм шифрування, у вигляді шифру AES, та реалізований за допомогою програмного середовища JavaScript.

На основі обраного шифру був створений веб-додаток, що може шифрування та дешифрування дані. Програма виконує свій функціонал справно, має закінчений дизайн та є крос-платформною, ефективна для використання в офісному середовищу для шифрованої передачі даних.

Підсумовуючи програмну реалізацію, можна сказати, що додаток працює добре, але може бути допрацьованим у сторону розширення методів шифрування даних. Гарним оновленням було б введення вибору методу шифрування, замість єдиного криптографічного шифру AES.

СПИСОК ЛІТЕРАТУРИ

1. Закон України "Про захист інформації в інформаційно-телекомунікаційних системах" від 31.05.2005 року № 2594-IV // Відомості Верховної Ради України. – 2005. – № 26.

2. Криптографічні системи та протоколи : навч. посіб. [для студ. баз. напряму 6.170101 "Безпека інформ. і комунікац. систем" усіх форм навчання] / А. Е. Лагун ; М-во освіти і науки України, Нац. ун-т "Львів. політехніка". – Л. : Вид-во Львів. політехніки, 2013.

3. О.В.Вербіцький. Вступ до криптології. Видавництво НТЛ., Львів, 2008, с.248.

4. Вакалюк Т. А. Основні поняття захисту інформаційних ресурсів у комп'ютерних системах / Т. А. Вакалюк // Науковий пошук молодих дослідників: збірник наукових праць / за ред. канд. пед. наук Королюк О.М. – Випуск 6. – Житомир: Вид-во ЖДУ ім. І.Франка, 2013.

5. Що таке шифрування 3DES і як працює DES? – [Електронний ресурс] – https://instagalleryapp.com/informacijna-bezpeka/shho-take-shifruvannja- 3des-i-jak-pracjue-des/

6. Advanced encrypted standard – [Електронний ресурс] – https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm

7. Інформаційна безпека комп'ютерних систем і мереж: Методичні вказівки // Укл. А.Ф.Карачка, М.П.Карпінський, А.В.Кулик, Т.В.Лендюк. – Тернопіль: ТАНГ, 2007. – 68 с

8. Understanding RSA Algorithm – [Електронний ресурс] –

https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_understanding_rsa_algorithm.htm

9. What is JavaScript? – [Електронний ресурс] – https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

10. CryptoJS – [Електронний ресурс] – https://code.google.com/archive/p/crypto- js/