МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія (кафедра) комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

МЕХАНІЗМ ЗАБЕЗПЕЧЕННЯ ЕФЕКТИВНОГО ЗАХИСТУ ДАНИХ З ВИКОРИСТАННЯМ КРИПТОПЕРЕТВОРЕНЬ

Виконав: студент групи 1П-21
Спеціальності
121 Інженерія програмного забезпечення
Катерина ДОБРОВОЛЬСЬКА
Керівник:
Майя ЛЮТА

Черкаси 2025

АНОТАЦІЯ

Кваліфікаційна робота на тему «Механізм забезпечення ефективного захисту даних з використанням криптоперетворень» складається з вступу, основної частини, що містить 3 розділи, висновку та списку використаних джерел. Загальний обсяг роботи – 74 сторінки. У роботі 41 рисунок та 5 таблиць. Перелік використаних ресурсів налічує 29 одиниць.

В даній роботі розглянуто та проаналізовано основні загрози даним, досліджено найбільш розповсюджені криптографічні алгоритми захисту даних. розроблено демонстраційний приклад роботи алгоритму за допомогою інструментів програми Visual Studio 2015, а також проведено тестування програмного продукту.

В першому розділі описані види даних, які підлягають захисту, основні типи загроз, а також засоби захисту даних.

В другому розділі описано класифікацію методів захисту даних та проаналізовано математичне обґрунтування трьох найбільш розповсюджених криптоалгоритмів.

В третьому розділі більш детально розглянуто рівень захисту даних в одному з алгоритмів, а також розроблено програмний продукт як демонстраційний приклад роботи даного алгоритму.

В четвертому розділі розглянуто та визначено способи тестування ПП, а також виконано тестування розроблених модулів програми за допомогою різних технік та методів.

Ключові слова: КРИПТОПЕРЕТВОРЕННЯ, КРИПТОСИСТЕМА, КРИПТОАНАЛІЗ, ЗАХИСТ ДАНИХ, ШИФРУВАННЯ, ДЕШИФРУВАННЯ, АЛГОРИТМ, AES, RSA, SHA.

ABSTRACT

The qualification work on the topic "Mechanism for ensuring effective data protection using cryptographic transformations" consists of an introduction, the main part, which contains 3 sections, a conclusion and a list of sources used. The total volume of the work is 74 pages. The work contains 41 figures and 5 tables. The list of resources used has 29 units.

This work considers and analyzes the main threats to data, investigates the most common cryptographic data protection algorithms. A demonstration example of the algorithm's operation using Visual Studio 2015 tools has been developed, and the software product has been tested.

The first section describes the types of data to be protected, the main types of threats, and means of data protection.

The second section describes the classification of data protection methods and analyzes the mathematical justification of the three most common cryptographic algorithms.

The third section considers the level of data protection in one of the algorithms in more detail, and a software product has been developed as a demonstration example of the operation of this algorithm.

The fourth section considers and defines the methods of testing PP, and also tests the developed program modules using various techniques and methods.

Keywords: CRYPTOTRANSFORMATION, CRYPTOSYSTEM, CRYPTANALYSIS, DATA PROTECTION, ENCRYPTION, DECRYPTION, ALGORITHM, AES, RSA, SHA.

ЗМІСТ

ВСТУП

Наприкінці 90-х років, коли активно почали розвиватися електронно-обчислювальні машини все більше і більше галузей поступово переходили до автоматизації роботи та впровадженню ЕОМ. Все більший обсяг даних почали зберігати на електронних носіях, дедалі більше інформації знаходилися в мережі і передавалися за допомогою її, а отже все більш актуальним ставало питання захисту даних, які зберігалися в електронному вигляді. Тоді почалася ера перших комп'ютерних криптографічних алгоритмів шифрування інформації.

Наразі все більше аспектів нашої життєдіяльності залежить від Інтернету, смартфонів та комп'ютерів, мережі стали глобальними, а інформація отримала цифрову форму бітів та байтів. Все більше автомобілів обладнано комп'ютерами, які повідомляють водіям про порушення в роботі системи, а також можуть повністю взяти на себе керування автомобільним транспортом. Комп'ютерні системи все більше проникають у наші будинки та застосовуються для керування домашньою технікою і координування її роботи. Люди отримують доступ до своїх банківських, а також медичних даних, що зберігаються, обробляється та передається в цифровому вигляді в комп'ютерних системах і відкритих каналах зв'язку, за допомогою мобільних девайсів, що спрощує віддалений доступ до них третіми особами. Кожна організація чи підприємство веде бухгалтерію і внутрішні бази даних в електронному вигляді. Це означає, що вся діяльність людини напряму залежить від комп'ютерної мережі, яка повинна також забезпечувати захист інформації якою керує. Зловмисники націлюються на комп'ютерні системи і відкривають канали зв'язку для того щоб вкрасти конфіденційну інформацію або порушити роботу критично важливої системи.

Актуальність. На сьогоднішній день чимало інформації зберігаються в мережі, це можуть бути дані особистого характеру, або комерційного, в будь-якому разі вони потребують надійного захисту. В сучасному світі існують багато криптографічних алгоритмів, які використовуються для захисту даних і

вважаються надійними, проте кожного дня з’являються більш потужні комп’ютери, а також все більше людей, які прагнуть зламати алгоритми шифрування та отримати захищені дані. Сучасна криптографія надає надійний набір методів, які гарантують, що зловмисні наміри порушника будуть зірвані, а законі користувачі отримають доступ до інформації.

Об’єкт дослідження. Сучасні ефективні механізми захисту даних. Основні криптоперетворення, які використовують як засоби захисту даних.

Предметом дослідження є сукупність засобів захисту даних, зокрема криптографічні алгоритми захисту даних.

Метою роботи є аналіз та визначення алгоритму шифрування, який має найбільший показник криптостійкості, а також розробити демонстраційний приклад роботи одного з алгоритмів.

Для досягнення цієї мети було вирішено такі **завдання**:

– класифікувати дані, які підлягають першочерговому захисту;

– розглянути та проаналізувати основні види загроз даних;

– розглянути методи захисту даних;

– проаналізувати сучасні алгоритми захисту даних;

– за допомогою аналізу математичних перетворень визначити найбільш надійні криптографічні алгоритми;

– розробити основні модулі демонстраційного прикладу.

Метод дослідження. Базується на математичному аналізі алгоритмів шифрування.

Практичне значення. Вдосконалення знань стосовно захисту інформації з використанням криптоперетворень, а також розкриття основних проблем пов’язаних з алгоритмами шифрування даних.

Постановка завдання. У процесі дослідження було вивчено причини, які можуть призвести до втрати даних, було вивчено загрози та проаналізовано способи їх запобіганню. Враховуючи, що вразливості удосконалюються час від часу і ті методи, які повинні їм запобігати з часом не діють, необхідно докладати більше зусиль у вивчення криптографії та алгоритмів шифрування задля забезпечення надійного обміну та збереження даних.

РОЗДІЛ 1
СУЧАСНІ ОСОБЛИВОСТІ ЗАХИСТУ ДАНИХ

1.1 Дані та їх види

Дані – це сукупність свідчень, зафіксованих на певному носії в форматі прийнятному для постійного зберігання, передачі та обробки. Перетворення та обробка даних дозволяє отримати інформацію. Компонентами даних являються цифри і символи мови або їх кодоване представлення у вигляді рядка двійкових бітів. Біт – просте двійкове число, яке приймає значення 1 або 0, які слугують для запису та зберігання даних в комп'ютері. Біт є найменшою двійковою одиницею вимірювання обсягу даних в комп'ютері. Інформація від латинського означає відомості, роз'яснення, викладення. Не дивлячись на широке розповсюдження значення цього слова, поняття інформації є одним з найбільш дискусійних в науці [1].

Щоб стати інформацією дані повинні правильно відображати об'єкти описання, в іншому випадку це буде дезінформацією.

Дані являють собою певним чином отриманні і зафіксовані спостереження відносно оточуючої дійсності. Виходячи з того для яких цілей зібрані дані, їх можна розділити на первинні та вторинні. Первинними є такі дані, котрі зібрані для рішення цілей даного дослідження. Відповідно, вторинними є дані, зібрані в рамках іншого дослідження.

За способом отримання, дані можна розділити на 4 типи:

1. спостереження – фіксація випадків або подій;
2. участь – дані отримуються завдяки досвіду;
3. вимірювання – фіксація величини або кількості будь-якого параметру;
4. інтерегація – дані отримуються за допомогою запитань до людей.

В науці про дані виділяють декілька основних категорій даних:

– Структуровані дані. Залежать від моделі даних і зберігаються в фіксованому полі всередині запису. Структуровані дані зручно зберігати в таблицях або базах даних. таке представлення зручне для людини та машини, проте частіше реальні дані зберігаються в неструктурованому вигляді.

– Неструктуровані дані. Їх важко підігнати під певну модель даних тому що частіше їх вміст залежить від контексту або має змінний характер. Одним з прикладів неструктурованих даних є повідомлення на електронній пошті. Таке повідомлення написане людиною також є прикладом даних написаних природньою мовою.

– Дані природньою мовою. Особливий різновид неструктурованих даних, обробка таких даних важка, через те що потребує знань лінгвістики та певних методів data science. Спільнота обробки даних природньою мовою домоглась успіху в області розпізнавання, узагальнення, завершення тексту і аналізу емоційного забарвлення. Проте навіть найсучасніші методи не зможуть розшифрувати сенс довільного фрагменту тексту, у людей також виникають проблеми із сприйняттям природньої мови. Вона неоднозначна за своєю природою.

– Машинні дані. Відноситься інформація автоматично генерована комп’ютером, процесом, додатком або пристроєм без втручання людини. Машинні дані стають одним з основних джерел інформації. Аналіз машинних даних через величезні об’єми і швидкості дуже сильно залежить від інструментів з високою масштабованістю.

– Графові дані. Будь-які дані можуть бути представлені у вигляді графа. В даному випадку мається на увазі поняття графа з математичної теорії графів – математична структура для моделювання попарних відношень між об’єктами. В графових або мережевих даних особливу увагу приділяють зв’язкам або суміжності об’єктів. Графові структури даних використовують вузли, ребра і властивості для представлення і зберігання графічних даних. графові дані підходять для представлення соціальних мереж, а їх структура дозволяє вираховувати найкоротший шлях між двома людьми.

– Аудіо, відео, графічні дані. Типи даних, які ставлять непрості задачі перед спеціалістом data science. Задачі звичайні для людини, виявляються важкими для комп'ютера. Ці дані використовуються для відслідковування камерами подій на футбольних матчах, на дорозі, для навчання штучного інтелекту за допомогою спостереження за людиною.

– Потокові дані. Можуть приймати майже будь-яку з представлених вище форм, однак в них є одна додаткова властивість. Дані поступають в систему при виникненні певних подій, а не завантажуються в сховище даних.

Спочатку з'явився засіб для опрацювання числових даних. В подальшому, особливо після широкого розповсюдження персональних комп'ютерів, комп'ютери стали використовувати для зберігання, обробки, передачі і пошуку текстових, числових, графічних, звукових та відеоданих [27].

З точки зору інформатики найбільш важливими є наступні властивості даних:

– об'єктивність;

– достовірність;

– повнота;

– точність;

– актуальність;

– корисність.

Дані як різновид інформації можуть бути загальнодоступними та конфіденційними. До загальнодоступних даних має доступ будь-яка людина, до конфіденційних – тільки окремі особи.

Головним завданням інформаційної безпеки є захист конфіденційної інформації, бо якщо доступ до неї отримає стороння особа це може призвести до негативних наслідків, наприклад крадіжці грошей, порушенню конституційних прав людини, втраті прибутку. Але це не означає, що загальнодоступну інформацію не треба захищати, вона повинна залишатися цілісною і доступною [2].

Основні види конфіденційних даних:

– Персональні дані. Інформація про конкретну людину: ПІБ, паспортні дані, номер телефону, сімейний стан, тощо. Той, хто працює з персональними даними зобов’язаний захищати їх і не передавати третім особам;

– Комерційна таємниця. Внутрішня інформація про роботу компанії: технології, методи керування, клієнтів. Якщо ця інформація стане відомою стороннім особам, компанія може втратити прибуток. Компанія сама вирішує, що вважати комерційною таємницею;

– Професійна таємниця. До цього відноситься лікарська, нотаріальна, адвокатська і інші види таємниць, які відносяться до професійної діяльності;

– Службова таємниця. Інформація, яка відома окремим службам, наприклад, податковій, або РАЦСу. Ці дані зазвичай зберігають державні органи і вони відповідають за їх захист і надають тільки за запитом;

– Державна таємниця. До цього відносяться військові таємниці, дані розвідки, інформація про стан науки та економіки, техніки та зовнішньої політики держави. Такі дані найбільш конфіденційні і до систем, що зберігають ці дані висуваються найбільш суворі вимоги.

1.2 Основні загрози даним

Розвиток інформаційного суспільства призвів до появи інформаційних технологій, які стали невід’ємною частиною нашого життя. Вони дають не тільки можливість для розвитку здібностей, покращення знань та розширення кола інтересів, але й містять у собі реальні загрози. Одна з найсерйозніших загроз – виникнення нового виду злочинності – комп’ютерна злочинність.

Інформаційна безпека – це сукупність заходів для захисту інформації від випадкового, або навмисного втручання третіх осіб. У доцифрову епоху люди складали важливі документи в сейфи, наймали охоронців та шифрували свої повідомлення на папері. Наразі частіше захищають цифрову інформацію аніж паперову однак способи захисту залишаються схожими: спеціалісти створюють

захищені простори, встановлюють захисне ПЗ, і використовують криптографічні алгоритми для шифрування. Однак необхідно захищати дані не лише віртуально, але й фізично, адже антивірус не допоможе, якщо зловмисник отримає доступ до серверу фізично, тому їх краще ставити в приміщення, які охороняються [17].

Інформаційна безпека має 4 основні принципи:

1. Цілісність. Здатність інформації зберігати початковий вигляд і структуру в процесі зберігання та передавання. Вносити зміни, видаляти чи доповнювати інформацію може тільки її власник, або людина, яка має легальний доступ до інформації;

2. Конфіденційність. Принцип який вказує на необхідність обмеження доступу до інформації для певного кола осіб. Доступ до інформації здійснює той, хто має на це право. В процесі дій та операцій інформація стає доступною тільки для тих, хто включений до інформаційної системи та успішно пройшов ідентифікацію.

3. Доступність. Означає, що інформація, яка знаходиться у вільному доступі, повинна надаватися повноправним користувачам ресурсів своєчасно та без перешкод.

4. Достовірність. Вказує на приналежність інформації довіреній особі, або власнику, котрий одночасно виступає в ролі джерела інформації [4].

Загрози безпеки – сукупність умов або факторів, які створюють небезпеку у відношенні даних, яка полягає в ознайомленні сторонніми особами з захищеними даними, несанкціонованій зміні, знищенні, розповсюдженні, а також інших неправомірних дій з даними.

Відповідно основні типи загроз ІБ є:

1. Загрози конфіденційності – несанкціонований доступ до інформації;

2. загрози цілісності – несанкціонована модифікація, доповнення, або знищення даних;

3. загрози доступності – обмеження або блокування доступу до даних [5].

Визначення каналів та причин, які приводять до неправомірних дій має першочергове значення при побудові моделі захисту даних. Загалом є 3 типи джерел загроз, кожне з яких має свої особливості:

1. Антропогенні пов’язані з людським фактором:

– Зовнішні – постачальники послуг, працівники контролюючих державних органів, хакери, представники конкуруючих організацій. Їх дії можуть бути навмисними, направленими на отримання інформації, або випадковими, наприклад, якщо витік даних відбувається в результаті технічного збою або не професійного створення проекту ІС.

– Внутрішні – працівники компанії, працівники програмного відділу, відділу кадрів, техперсонал або представники служби безпеки компанії. В процесі своєї діяльності вони можуть наражати на небезпеку СЗ через некомпетентність і помилкові дії, застосування не ліцензійного ПЗ, модифікацію чи знищення компонентів програм, надання доступу неуповноваженим особам або ігнорування правил зберігання інформації.

2. Стихійні. Фактори на які оператор ні як не може вплинути:

– повені;

– цунамі;

– пожежі;

– урагани;

– зсуви;

– радіаційні катастрофи;

– військові конфлікти.

3. Техногенні. Ці джерела обумовленні застосовуваним технічним обладнанням і бувають двох видів:

– Внутрішні – віруси, шкідливі програми, системи охорони та сигналізації, низькоякісне ПЗ і обладнання, яке задіяне в обробці інформації.

– Зовнішні – складники інфраструктурного призначення, наприклад, лінії телефонного та інтернет-зв’язку, системи опалення, каналізації, водопостачання, газопостачання [6].

Джерелами загроз виступати можуть кіберзлочинні групи і державні спецслужби, які використовують зазвичай такі доступні засоби:

– небажаний контент;

– несанкціонований доступ;

– витік інформації;

– втрата даних;

– шахрайство;

– кібертероризм.

У табл. 1.1 наведено певні шляхи реалізації загроз, а також специфічні назви та терміни: «троянський кінь», «вірус», «черв’як». Хоча ці назви мають жаргонний відтінок, вони уже ввійшли в загальноприйнятий комп’ютерний лексикон.

Таблиця 1.1 – Шляхи реалізації загроз безпеці ІС

Об’єкти впливу	Порушення конфіденційності інформації	Порушення цілісності інформації	Порушення працездатності системи
Апаратні засоби	Несанкціонований доступ – підключення; використання ресурсів; викрадення носіїв	Несанкціонований доступ – підключення; використання ресурсів; модифікація, зміна режимів	Несанкціонований доступ – зміна режимів; виведення з ладу; пошкодження
Програмне забезпечення	Несанкціонований доступ – копіювання; викрадення; перехоплення	Несанкціонований доступ – впровадження «троянських коней», «вірусів», «черв’яків»	Несанкціонований доступ – спотворення; знищення; підміна
Дані	Несанкціонований доступ – копіювання; викрадення; перехоплення	Несанкціонований доступ – спотворення; модифікація	Несанкціонований доступ – спотворення; знищення; підміна
Персонал	Розголошення; передача відомостей про захист; халатність	«Маскарад»; вербування; підкуп персоналу	Покидання робочого місця; фізичне усунення

1.3 Засоби захисту даних

Засоби захисту даних – сукупність інженерно-технічних, електричних, електронних, оптичних і інших застосунків, приладів і технічних систем, які використовуються для рішення задач захисту даних в тому числі попередження витоку і забезпечення безпеки інформації, що захищається.

Забезпечення безпеки передбачає організацію протидії будь-якому несанкціонованому доступу до системи, її модифікації, викрадення, видалення, порушенню нормального функціонування, чи знищення її компонентів. Існує 2 типи підходів до забезпечення безпеки ІС:

1. Фрагментарний – направлений на протидію чітко визначеним загрозам. Перевагою такого підходу є висока вибірковість до конкретної загрози. Суттєвим недоліком такого підходу є відсутність єдиного захищеного середовища обробки інформації. Фрагментарні міри захисту інформації забезпечують захист конкретних об’єктів ІС тільки від конкретної загрози.

2. Комплексний – орієнтований на створення захищеного середовища обробки даних в ІС, яке об’єднує в єдиний комплекс різноманітні заходи протидії загрозам. Організація захищеного середовища обробки інформації дозволяє гарантувати певний рівень безпеки ІС, що є неодмінною перевагою комплексного підходу. Основними недоліками цього підходу є: обмеження на свободу дій користувачів ІС, велика чутливість до помилок встановлення та настроювання засобів захисту, складність управління.

В цілому засоби забезпечення захисту даних в залежності від способу реалізації можна розділити на групи:

1. Технічні засоби. Різні за типом пристрої, які апаратними засобами вирішують задачі захисту даних. Вони, або перешкоджають фізичному доступу до даних, або, якщо проникнення відбулося, маскують їх. Першу частину задачі вирішують замки, решітки на вікнах, сигналізація. Другу – генератори шуму, мережеві фільтри, скануючі радіоприймачі і інші пристрої перекриваючі потенційні канали витоку інформації не дозволяючи їх виявити. Переваги

технічних засобів в тому що вони надійні, незалежні від суб'єктивних факторів та стійкі до модифікацій. Слабкі сторони – недостатня гнучкість, відносно великі об'єм, маса та вартість.

2. Програмні засоби. Включають програми для ідентифікації користувачів, контролю доступу, шифрування даних, видалення залишкової робочої інформації, тестового контролю системи захисту. Перевагами програмних засобів є: універсальність, гнучкість, надійність, простота у використанні, здатність до модифікації та розвитку. Недоліки – обмежена функціональність мережі, використання частини ресурсів файл-сервера і робочих станцій, висока чутливість до випадкових або умисних змін, залежність від типів комп'ютерів або їх апаратних засобів.

3. Змішані апаратно-програмні засоби реалізують ті самі функції, що апаратні та програмні засоби [12]. Забезпечують підвищений рівень захисту, а також здатні захистити дані, які перебувають поза межами ОС, вони поділені на 5 основних груп:

– система ідентифікації (секретна інформація, якою володіє лише користувач: паролі, ключі, секретні коди, тощо) та автентифікації (біологічні параметри людини, тобто відбиток пальця, сітківка ока);

– шифрування дискових даних – основне завдання, яке виконує така система полягає в захисті від несанкціонованого доступу та розміщення чи модифікацію даних на магнітних носіях інформації. Шифрування відбувається за допомогою симетричних алгоритмів. Робота прикладних програм з накопичувальними дисками складається з двох етапів (логічного і фізичного), а за способом шифрування ці системі поділяються на: прозоре шифрування (криптографічні перетворення) відбуваються в режимі реального часу непомітно для користувача, а ось системи другого класу зазвичай представляють собою утиліти, які необхідно спеціально викликати для виконання шифрування. До них відносяться, наприклад, архіватори з вбудованими засобами парольного захисту.

– способи шифрування даних, що передаються по комп’ютерним мережам: канальне – у його випадку захищається уся інформація, що передається по каналу зв’язку, включаючи і службову; кінцеве (абонентське) шифрування дозволяє забезпечити конфіденційність даних, що передаються між двома прикладними об’єктами (абонентами). В цьому випадку захищається тільки зміст повідомлення, а вся службова інформація залишається відкритою.

– системи автентифікації електронних даних при обміні електронними даними по мережі виникає проблема автентифікації автора документу. Для автентифікації електронних даних застосовують електронний підпис. Електронний цифровий підпис (ЕЦП) представляє собою відносно невеликий об’єм додаткової автентифікуючої цифрової інформації, що передається разом із «підписаними» даними. Для реалізації ЕЦП використовуються принципи асиметричного шифрування [7].

– засоби управління ключовою інформацією. Під ключовою інформацією тут розуміється сукупність усіх використовуваних в комп’ютерній системі чи мережі криптографічних ключів. Способи генерації ключів розрізняються для симетричних і асиметричних криптосистем. Функція зберігання ключів передбачає організацію безпечного зберігання, обліку та знищення ключів. Для забезпечення безпечного зберігання та передачі ключів застосовують їх шифрування з використанням інших ключів. Розповсюдження ключів є найвідповідальнішим процесом в управлінні ключами. Цей процес повинен гарантувати секретність розповсюджуваних ключів, а також оперативність і точність їх розповсюдження.

4. організаційні засоби складаються з організаційно-технічних та організаційно-правових. Переваги організаційних засобів полягають у тому, що вони дозволяють вирішувати безліч різнорідних проблем, прості в реалізації, швидко реагують на небажані дії в мережі, мають необмежені можливості модифікації і розвитку. Недоліки – висока залежність від суб'єктивних чинників, в тому числі від загальної організації роботи в конкретному підрозділі.

Основні форми захисту інформації.

У відповідності з певною класифікацією даних, користувачів, апаратури, приміщень відповідальні за безпеку розробляють багаторівневу підсистему управління доступом, яка повинна виконувати наступні завдання:

1. ідентифікувати користувачів, персонал, ресурси комп'ютерної системи шляхом присвоювання кожному об'єкту персонального ідентифікатора (коду, імені, тощо);
2. автентифікувати (встановлювати справжність) об'єкти по представлених відомостях (паролях, ключах, кодах та інших ознаках);
3. проводити авторизацію (перевіряти повноваження) запитів суб'єкта у відповідності до встановленого регламенту роботи;
4. організовувати роботу у відповідності із загальним регламентом;
5. протоколювати звернення до захищених компонентів комп'ютерної системи;
6. реагувати при несанкціонованих діях (затримка чи відмова обслуговування, сигналізація) [3].

Висновки до розділу 1

Дані невід'ємна частина життя людини. Вони охоплюють і складають весь життєвий простір навколо людей. В першому розділі було розглянуто поняття даних та їх види, також було досліджено, які з видів даних підлягають першочерговому захисту. Досліджуючи питання інформаційної безпеки, було виділено її 4 основні принципи:

- цілісність;
- конфіденційність;
- доступність;
- достовірність.

Відповідно до цих принципів існують і типи загроз цілісності, конфіденційності та доступності даних. Проте існують і певні засоби захисту

інформації, які можна підібрати до загрози і забезпечити надійний захист даних, які поділяються на:

– програмні засоби;
– апаратні засоби;
– програмно-апаратні засоби.

Для забезпечення максимального рівня захисту рекомендується використовувати змішані засоби захисту, так як вони надають захист від більшого числа загроз.

З першого розділу стає зрозумілою необхідність захисту даних. Надійність захисту даних залежить від правильного розуміння загроз та вірного використання методів та засобів захисту даних.

РОЗДІЛ 2
МЕТОДИ ЗАХИСТУ ДАНИХ

2.1 Класифікація методів шифрування

Шифрування – це технологія кодування та декодування інформації. Зашифровані дані – це результат застосування алгоритму для кодування даних з цілю зробити їх недоступними для читання третіми особами. Дані можуть бути розкодовані в початкову форму тільки із застосуванням певного ключа. Шифрування є важливою частиною забезпечення безпеки інформації, оскільки захищає конфіденційну інформацію від загроз. Шифрування – це універсальне рішення: воно може застосовуватись до частини даних, наприклад, до паролю, до певної інформації в файлі або до всіх даних, які зберігаються на носії. [8]

Розглянемо приклад шифрування даних за допомогою найпростішого шифру ROT13. ROT13 – розповсюджений тип шифрування, де кожна буква алфавіту зсувається на 13 позицій як показано на рис. 2.1.

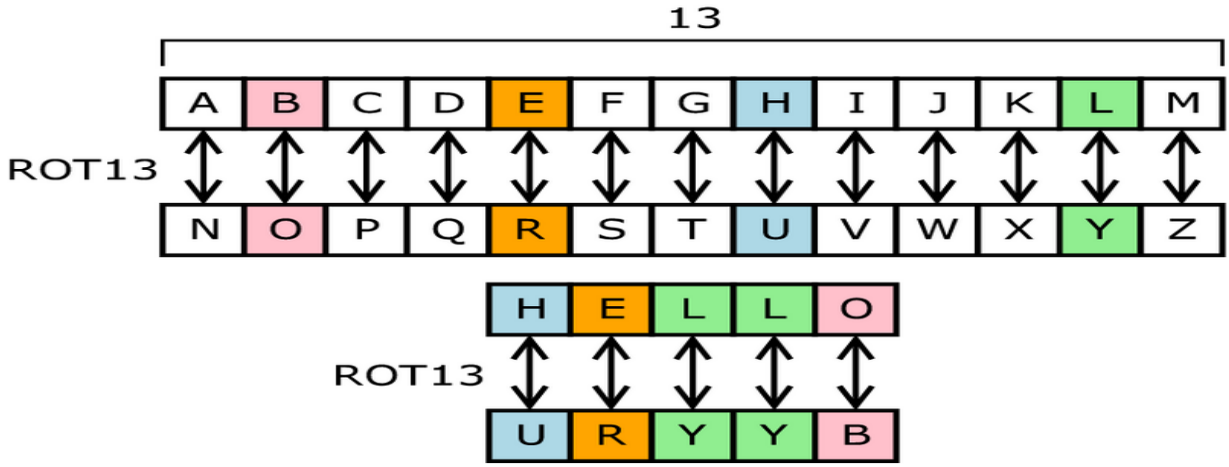

Рисунок 2.1 – Шифр ROT13

Як можна помітити, цей шифр не забезпечує по-справжньому надійний захист інформації, проте він є простим та зрозумілим прикладом шифрування даних.

Зазвичай сайти фінансових, державних, освітніх та торговельних організацій шифрують ваші дані задля того аби захистити їх від зловмисників та крадіїв. Сучасні методи шифрування побудовані на основі алгоритмів шифрування та ключах. Ключ – значення, яке не залежить від відкритого тексту [9].

Загалом існує декілька типів шифрування даних в залежності від ключів:

1. Симетричне шифрування.

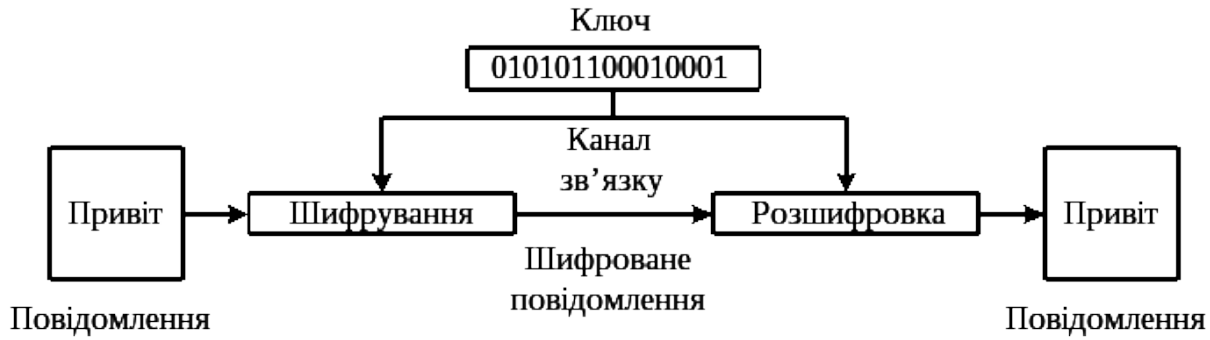

Рисунок 2.2 – Схема симетричного шифрування

Використовує один криптографічний ключ для шифрування та дешифрування даних (див. рис. 2.2). Ключ є найбільш важливою частиною симетричного шифрування. Він повинен бути прихований від сторонніх так як кожен хто має до нього доступ може дешифрувати приватні дані. В сучасних системах ключ зазвичай являє собою строку даних, яка отримана з надійного пароля або абсолютно випадкового джерела [10]. Прикладом симетричного алгоритму шифрування є алгоритм AES.

Переваги симетричного шифрування:

– значно швидший за аналоги;
– потребує менше обчислювальної потужності;
– не знижує швидкість Інтернет з'єднання.

Недоліки симетричного способу шифрування даних:

– важкість забезпечення безпеки ключа, а саме його зберігання та передача;

– неможливість ПЗ працювати з зашифрованими даними;

– компрометація програмного забезпечення за допомогою якого відбувалося шифрування інформації.

2. Асиметричне шифрування.

Рисунок 2.3 – Схема асиметричного шифрування

Метод асиметричного шифрування працює аналогічно симетричному, однак він використовує два ключі (див. рис. 2.3), один для шифрування інформації, він називається відкритий, другий для дешифрування інформації, він називається закритий. Зазвичай закритий та відкритий ключі пов’язані між собою математично. Оскільки секретний ключ потрібен тільки для розшифрування даних, його не потрібно щоразу передавати і, зазвичай, він відомий тільки одній людині – отримувачу вірогідність того, що зловмисник зможе його дешифрувати значно менша. Переваги асиметричних алгоритмів шифрування:

– гарантія, що дані захищені від атак «людина посередині» (MiTM);

– не перевантажуються вебсервери та сервери електронної пошти яким потрібно керувати лише одним ключем і захищати його;

– зашифроване з’єднання без зустрічі для попереднього обміну ключами;

– за допомогою закритого ключа гарантує, що ви розмовляєте з реальною людиною або організацією.

Недоліки асиметричного методу шифрування:

– атаки «людина посередині»;

– якщо є необхідність зашифрувати великий об’єм даних то на це буде витрачено більше часу, в порівнянні з симетричними методами шифрування. Прикладом асиметричного алгоритму шифрування є алгоритм RSA.

3. Хеш-функції.

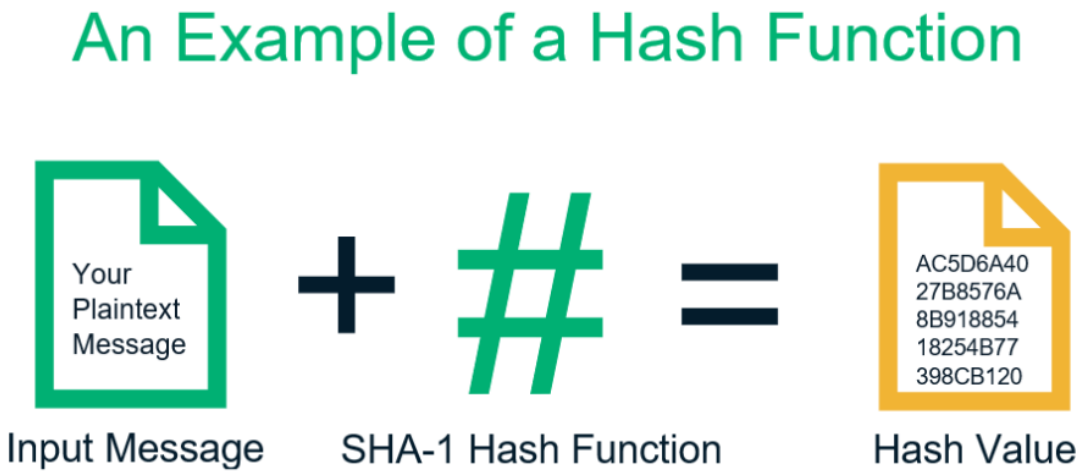

Рисунок 2.4 – Схема алгоритму хеш-функцій

Хеш-функції використовуються для генерації спеціального рядка з приведених даних, відомих як хеш (див. рис. 2.4). Цей хеш має декілька властивостей:

– одні й ті ж дані завжди виробляє той же самий хеш;

– неможливо генерувати вихідні дані з хешу поодинці;

– недоцільно пробувати різні комбінації вхідних даних, щоб згенерувати той же самий хеш [14].

Основна відмінність між хешуванням та двома іншими способами шифрування інформації полягає в тому, що як тільки дані зашифровані, вони не можуть бути отримані назад в початковому вигляді. Цей факт гарантує, що навіть якщо хакер отримає на руки хеш, це буде для нього марно так як він не зможе дешифрувати зміст повідомлення. Прикладом хеш-функції є алгоритм хешування SHA-3. Хеш-функції зберігаються в спеціальних таблицях, які

називають хеш-таблицями. Хеш-таблиця – спеціальна структура даних для зберігання пар ключів та значень в якій ключ представлений у вигляді хеш-функції [18].

Переваги хеш-таблиць:

– основною перевагою є швидкість, особливо ефективні при великій кількості записів;

– якщо набір пар ключ-значення фіксований і відомий завчасно, можливо зменшити середню вартість пошуку шляхом ретельного вибору хеш-функції, розміру таблиці сегментів і внутрішніх структур даних.

Недоліки:

– хеш-таблиці мало ефективні для малої кількості записів;

– записи, що зберігаються, можуть бути ефективно перераховані, але лише в деякому випадковому порядку;

– якщо ключі не збережені, може бути нелегко перерахувати ключі, які присутні в таблиці в будь-який даний момент;

– якщо в хеш-таблиці використовується динамічна міра розміру, операція вставки або видалення може іноді займати час, пропорційний кількості записів [11].

4. Гібридне шифрування. Полягає в комбінуванні всіх трьох методів шифрування описаних вище. Метод секретного ключа дає змогу швидко дешифрувати дані, а метод відкритого ключа пропонує більш безпечний і зручний спосіб передачі ключа. Ця комбінація методів відома як «цифровий конверт». Хешування застосовується як засіб перевірки надійності пароля. Якщо система зберігає хеш пароля замість самого пароля, він буде більш безпечним так як навіть якщо хакеру потрапить до рук цей хеш він нічого не зможе з нього зрозуміти. В ході перевірки система перевіряє хеш вхідного пароля і якщо він співпадає із хешем в таблиці пускає користувача в систему. Таким чином фактичний пароль можна побачити тільки в моменти, коли він має бути перевірений або змінений, що суттєво знижує вірогідність його потрапляння в чужі руки [13]. Оскільки метод гібридного шифрування

побудованих на симетричному, асиметричному методах та методі хеш-функцій, як таких переваг або недоліків цей спосіб шифрування не має. Гібридне шифрування використовується з урахуванням всіх переваг та недоліків трьох основних методів шифрування та заснований на комбінуванні їх найкращих сторін задля забезпечення ефективного та надійного захисту даних [16].

2.2 Симетричний алгоритм шифрування AES

Advanced Encryption Standart (AES) – симетричний алгоритм блочного шифрування, прийнятий урядом США в якості стандарту в результаті конкурсу, проведеного між технологічними інститутами. Він замінив застарілий алгоритм DES, який більше не відповідав вимогам мережевої безпеки. Суть AES полягає в тому, що будь-яка «лобова атака» на захищені дані дуже сильно затягується. Якщо зломщик має в своєму розпорядженні великі ресурси, тобто чимало суперкомп'ютерів, то при великих зусиллях він би міг отримати доступ до даних через десятки років. Якщо у зловмисника цього немає, то злом AES може зайняти більше часу, ніж ймовірний вік всесвіту. Алгоритм AES використовується агенцією національної безпеки США з 2003 року для захисту відомостей, які складають державну таємницю. Аж до рівня SECRET було дозволено використовувати ключі довжиною 128 біт, а для рівня TOP SECRET ключі довжиною 192 та 256 біт [23].

Загальні характеристики AES:

- AES зашифровує і розшифровує 128-бітові блоки даних.
- AES дозволяє використовувати три різних ключа довжиною 128, 192 або 256 біт (в залежності від довжини ключа версії шифру позначають AES-128, AES-192 або AES-256).
- 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

Основним елементом, яким оперує алгоритм AES, є байт – послідовність 8 біт, що обробляються як єдине ціле. Для формування байтів 128 бітів блоку відкритого тексту, вихідного блоку шифротекста і ключа шифру діляться на групи з 8-ми поруч стоячих біт так, щоб в цілому вийшов масив байт. Нижче на табл. 2.1 представлена прийнята нумерація біт в межах кожного байта:

Таблиця 2.1 – Розподіл бітів на вході

№ біта на вході	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
№ байта	0								1							
№ біта в байті	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Задавати значення байта зручно в шістнадцятковій системі обчислення. Для цього байт ділиться на дві групи з 4-х біт: група старших біт в байті представляється першим шістнадцятковим символом, а група молодших біт – другим. Наприклад, для байта 10101100 отримаємо:

10101100 = 1010 1100 = AC.

Необхідно зазначити, що:

in0, in1, ..., in15 – 16 байт блоку відкритого тексту;

k0, k1, ..., k15 – 16 байт ключа шифру;

out0, out1, …,out15 – 16 байт блоку шифротексту.

Масив in0, in1, ..., in15 – це вхідні дані для шифрування. Перед початком шифрування байти цього масиву розміщуються послідовно в стовбці матриці InputBlock. Всередині алгоритму операції виконуються над матрицею State, яка називається матрицею станів. Кінцеве значення матриці станів OutputBlock є виходом алгоритму та перетворюється на послідовність байтів шифротексту out0, out1, …,out15. Відповідно в стовпці матриці InputKey потрапляють байти k0, k1, ..., k15 ключа шифру. Схему цього процесу можна побачити у табл. 2.2

Таблиця 2.2 – Схема передачі даних всередині алгоритму

Вхідні байти InputBlock					Матриця станів State					Вихідні дані OutputBlock				Байти ключа InputKey			
in_0	in_4	in_8	in_{12}		$s_{0.0}$	$s_{0.1}$	$s_{0.2}$	$s_{0.3}$		out_0	out_4	out_8	out_{12}	k_0	k_4	k_8	k_{12}
in_1	in_5	in_9	in_{13}	→	$s_{1.0}$	$s_{1.1}$	$s_{1.2}$	$s_{1.3}$	=	out_1	out_5	out_9	out_{13}	k_1	k_5	k_9	k_{13}
in_2	in_6	in_{10}	in_{14}		$s_{2.0}$	$s_{2.1}$	$s_{2.2}$	$s_{2.3}$		out_2	out_6	out_{10}	out_{14}	k_2	k_6	k_{10}	k_{14}
in_3	in_7	in_{11}	in_{15}		$s_{3.0}$	$s_{3.1}$	$s_{3.2}$	$s_{3.3}$		out_3	out_7	out_{11}	out_{15}	k_3	k_7	k_{11}	k_{15}

Наприклад, представимо у вигляді матриці InputBlock текст:

БУДИНОКБІЛЯОЗЕРА = $(02\ 24\ 06\ 11\ 18\ 19\ 15\ 02\ 12\ 16\ 33\ 19\ 10\ 07\ 21\ 01)_{10}$=

= $(2\ 18\ 6\ B\ 12\ 13\ F\ 2\ C\ 10\ 21\ 13\ A\ 7\ 15\ 1)_{16}$

$$\text{InputBlock} = \begin{pmatrix} 2 & 12 & C & A \\ 18 & 13 & 1 & 7 \\ 6 & F & 21 & 15 \\ B & 2 & 13 & 1 \end{pmatrix}$$

Кожен блок даних в цьому алгоритмі шифрується в декілька етапів, які називаються раундами. Найменша кількість раундів 10. Перед початком шифрування виконується операція KeyExpansion, тобто операція розширення ключа. Як показано на рис. 2.5 розширення ключа використовується для додавання нової комбінації ключових слів на кожному етапі шифрування. Це, власне, ті ключі за допомогою яких відбувається і розшифрування інформації.

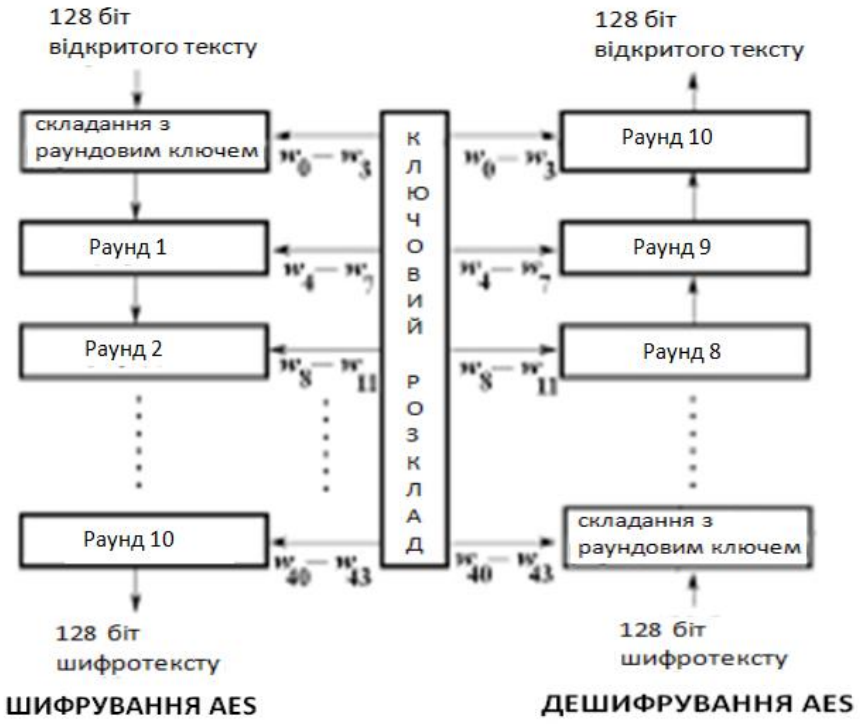

Рисунок 2.5 – Операція розширення ключа

Початковою операцією є AddRoundKey – складання з основним ключем по модулю 2.

Етапи шифрування (рис. 2.6):

1. SubBytes – побайтова підстановка в S-боксі з фіксованою таблицею замін.
2. ShiftRows – побайтовий зсув рядків матриці State на різну кількість байт.
3. MixColumns – перемішування байт в стовпцях.
4. AddRoundKey – складання з раундовим ключем.

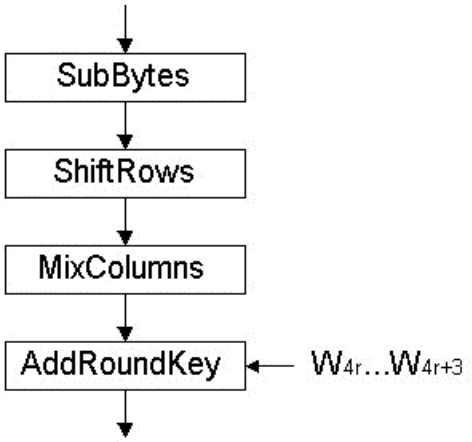

Рисунок 2.6 – Етапи шифрування

Розглянемо детальніше операції раундового перетворення шифрування.

1. SubBytes – операція виконує нелінійну заміну байтів, яка виконується незалежно з кожним байтом матриці State. На основі цієї операції створена спеціальна таблиця замін байтів в шістнадцятковій системі, яку називають S-боксом (рис. 2.7).

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

Рисунок 2.7 – S-бокс

За допомогою таблиці рис. 2.7 ми можемо змінити кожен вхідний байт. Наприклад, якщо $s_{1.1}$ = {6f}, то результатом заміни цього байта буде байт який знаходиться на перетині рядка з індексом 6 та стовпця з індексом f, тобто SubBytes (6f) = {a8}.

2. ShiftRows – операція застосовується до матриці State. Її перший рядок не рухається, а елементи нижніх трьох циклічно зсуваються вправо на 1, 2 та 3 байта, як показано на рис. 2.8. це перестановка елементів матриці в якій беруть участь лише рядки, тому перетворення оборотнє.

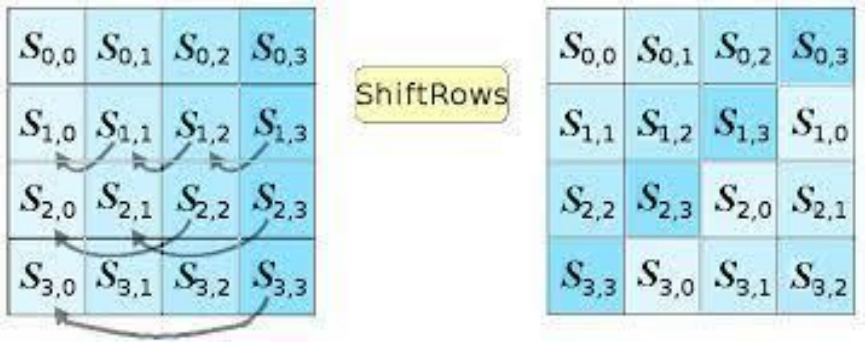

Рисунок 2.8 – Перестановка елементів матриці

3. MixColumns (рис. 2.9). За допомогою цієї операції виконується переміщування байтів в стовпцях матриці State. Кожен стовпець цієї матриці приймається за многочлен над полем GF(2^8) і помножується на фіксований многочлен:

$$c(x) = c_3x^3 + c_2x^2 + cx + c_0 = \{03\}x^3 + \{01\}x^2 + \quad (2.1)$$
$$+\{01\}x + \{02\}mod(x^4 + 1)$$

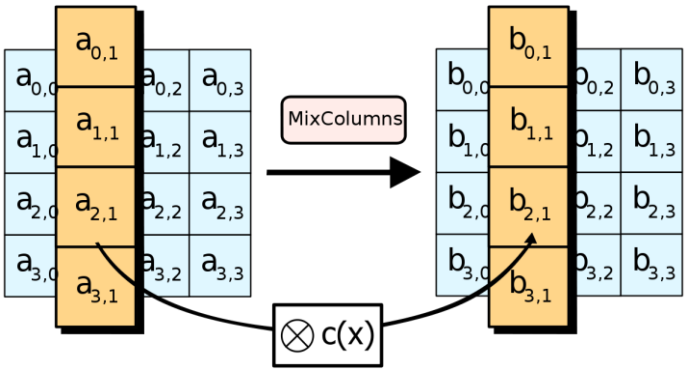

Рисунок 2.9 – Операція MixColumns

4. AddRoundKey (рис. 2.10). Функція побітово складає елементи змінної RoundKey і елементи змінної State за принципом: *i*-й стовбець даних складається з певним 4-байтовим фрагментом розширеного ключа:

$$W[4r+1], \quad (2.2)$$

де r – номер поточного раунду алгоритму.

При шифрування перше складання ключа раунду відбувається до першого виконання операції SubBytes.

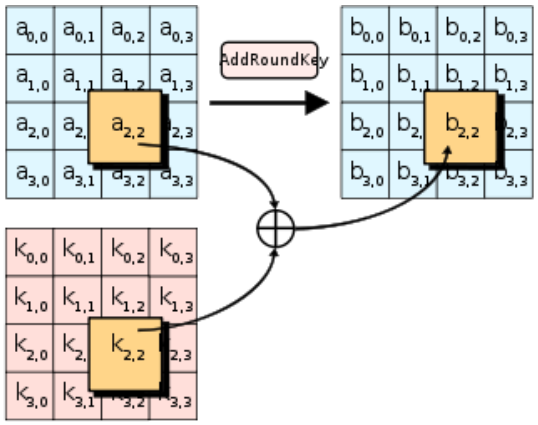

Рисунок 2.10 – Функція AddRoundKey

Раундові ключі створюються з основного ключа шифру K за допомогою процедури розширення ключа KeyExpansion. В результаті формується масив раундових ключів з якого потім безпосередньо вибирається необхідний раундовий ключ. Кожний раундовий ключ має довжину 128 біт (4 чотири-байтових слова) $w_i, w_{i+1}, w_{i+2}, w_{i+3}$, довжина в бітах всіх раундових ключів дорівнює 1408 біт (44 чотири-байтових слова $w_0, w_1, w_2, \dots, w_{42}, w_{43}$). Перші чотири слова w_0, w_1, w_2, w_3 в ключовому масиві заповнені ключем шифру, з інших 40 слів обираються по 4 слова для кожного з раундів. Нові слова $w_{i+4}, w_{i+5}, w_{i+6}, w_{i+7}$ наступного раундового ключа визначаються зі слів $w_i, w_{i+1}, w_{i+2}, w_{i+3}$ попереднього ключа як показано на малюнку рис. 2.11 [25].

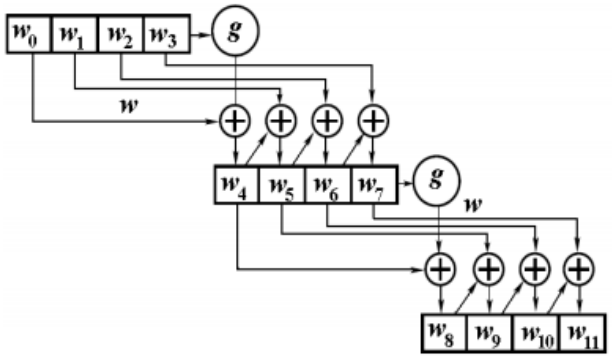

Рисунок 2.11 – Процедура розширення ключа

Для розшифрування шифротексту всі використані перетворення можуть бути інвертовані та застосовуватися в зворотному порядку. Перед першим раундом дешифрування виконується операція AddRoundKey, яка накладає на шифротекст чотири останніх слова розширеного ключа. Далі виконуються 10 раундів дешифрування, кожен з яких складається з таких операцій:

1. Операція InvShiftRows, зворотна до операції ShiftRows. Байти в останніх трьох рядках матриці State циклічно зсуваються вліво на різну кількість байтів. Перший рядок не зсувається, а інші три зсуваються вліво на 1, 2 та 3 байта відповідно.

2. Операція InvSubBytes, зворотна до операції SubBytes. Байти матриці State замінюються на нові значення за приведеною на рис. 2.12 таблиці зворотної заміни, яка є інверсним S-боксом.

		y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Рисунок 2.12 – Таблиця з інверсним S-боксом.

3. Операція InvMixColumns – процедура, зворотна до процедури MixColumns. Кожен стовпець матриці State розглядається як чотиричленний многочлен над полем GF(2^8) і помножується на фіксований многочлен:

$$c^{-1}(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\} mod(x^4 + 1) \qquad (2.3)$$

4. Операція AddRoundKey є зворотною сама до себе , так як полягає тільки в складанні по mod2 [31].

Останній раунд дешифрування не містить операцію InvMixColumns.

В алгоритмі розшифровки послідовність перетворень відрізняється від порядку операцій шифрування, а алгоритм розширення ключа залишається незмінним.

2.3 Асиметричний алгоритм шифрування RSA

Асиметричні алгоритми шифрування. Асиметричні алгоритми для

шифрування інформації використовують ключову пару <КВ, КС>, де КВ – відкритий (public) ключ, КС – секретний (private) ключ. Між відкритим і секретним ключем існує взаємно однозначна відповідність. Тобто, якщо змінити відкритий ключ, то зміниться секретний, і навпаки, якщо змінити секретний ключ, то зміниться відкритий. Якщо інформацію шифрувати відкритим ключем, то розшифровувати треба секретним, і навпаки, якщо шифрувати секретним, то розшифровувати треба відкритим. Відкритий ключ може бути опублікований для використання всіма користувачами системи обміну інформацією. Часто асиметричні алгоритми шифрування називають шифрами з відкритим ключем.

Стійкість асиметричних алгоритмів полягає в складності вирахування великих простих чисел. Найбільш відомі криптосистеми з відкритим ключем:

– рюкзачна криптосистема (Knapsack Cryptosystem);
– криптосистема RSA;
– криптосистема Ель-Гамаля (El Gamal Cryptosystem);
– криптосистема заснована на властивостях еліптичних кривих.

Застосування алгоритмів шифрування з відкритим ключем дозволяє позбутися секретних каналів зв’язку для попереднього обміну ключами, вирішує задачу забезпечення юридичної значимості електронних документів.

Прикладом асиметричного алгоритму шифрування є алгоритм RSA (рис. 2.13). Його запропонували в 1978 р. три автори: Р. Райвест (Rivest), А. Шамір (Shamir) і А. Адлеман (Adleman). Алгоритм одержав свою назву за першими буквами прізвищ його авторів.

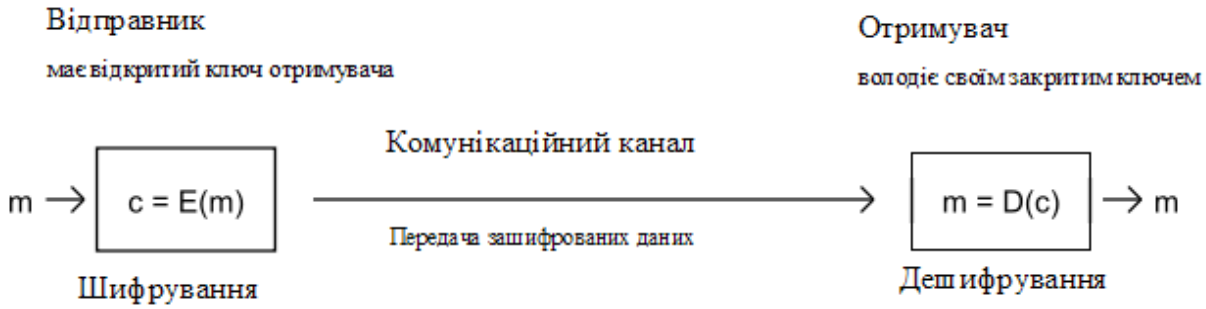

Рисунок 2.13 – Схема алгоритму RSA

Криптосистема RSA побудована на складності вирахування великих простих чисел і позначає, так звані, напівпрості числа, які отримано в результаті перемноження двох простих чисел. Взлом цього алгоритму полягає в тому щоб знайти початкові прості числа.

Першим етапом будь-якого асиметричного алгоритму є створення пари ключів – відкритого і закритого і розповсюдження відкритого ключа.

Для алгоритму RSA етап створення ключів складається з наступних операцій:

1. Обираються два великих простих числа p та q.
2. Знаходимо їх добуток

$$n = p * q \tag{2.4}$$

який називається модулем.

3. Далі знаходимо значення функції Ейлера від числа n:

$$\varphi(n) = (p-1)(q-1). \tag{2.5}$$

4. Обираємо довільне число e, яке повинно задовольняти наступним критеріям:

– Вони повинно бути просте;

– Бути меншим за $\varphi(n)$ (формула 2.5);

– Повинно бути взаємно простим з $\varphi(n)$ (формула 2.5).

Тобто

$$e\big(1 < e < \varphi(n)\big) \tag{2.6}$$

Число e називається відкритою експонентою.

В результаті цих дій ми маємо пару чисел $\{e, n\}$ – це відкритий ключ, який можна відправляти по мережі для того аби ваш співрозмовник зашифрував своє повідомлення. Проте ще необхідно отримати секретний ключ.

5. Необхідно обчислити число d зворотне до числа e по модулю $\varphi(n)$.

$$d = e^{-1} mod\ \varphi(n) \tag{2.7}$$

Таким чином отримуємо число d і в нас формується пара чисел $\{d, n\}$ – це секретний ключ, який не передається по мережі, його не можна нікому повідомляти. Тільки володар секретного ключа може дешифрувати повідомлення, яке було зашифровано відкритим ключем [32].

Після того як визначені відкритий та секретний ключі можна розпочинати шифрування повідомлення. Повідомленнями є цілі числа в інтервалі від 0 до $n - 1$.

Шифрування повідомлення:

1. Береться відкритий ключ $\{e, n\}$.
2. Відкритий текст m.
3. За допомогою відкритого ключа шифрується повідомлення m за формулою

$$\mathrm{c} = E(m) = m^e \, mod \, n \tag{2.8}$$

В результаті вищеописаних дій отримується зашифроване повідомлення, яке називають шифротекстом. Отримані дані відправляються по мережі отримувачу не боячись, що вони можуть бути розшифровані, адже закритий ключ за допомогою якого може відбутись дешифрування відомий тільки відправнику.

Алгоритм дешифрування повідомлення за допомогою секретного ключа:

1. Отримане зашифроване повідомлення с.
2. Береться свій закритий ключ $\{d, n\}$.
3. Застосовується закритий ключ до шифротексту за формулою:

$$m = D(c) = c^d \, mod \, n \tag{2.9}$$

Таким чином отримується відправлене повідомлення у початковому значенні. Через те, що закритий ключ зберігається тільки в отримувача в секреті ніхто окрім нього не може розшифрувати це повідомлення [15].

А тепер на табл. 2.3 ми можемо розглянути виконання алгоритму RSA на практиці.

Таблиця 2.3 – Поетапне виконання алгоритму RSA

Етап	Опис операції	Результат операції
Генерація ключів	Обрати два простих числа	$p = 31$ $q = 17$
	Обчислити модуль	$n = p * q = 31 * 17 = 527$ (формула 2.4)
	Обчислити функцію Ейлера	$\varphi(n) = (p-1)(q-1) = 480$ (формула 2.5)
	Обрати відкриту експоненту	$e = 7$
	Обчислити закриту експоненту	$d = e^{-1} mod\ \varphi(n)$ (формула 2.7) $d = 343$
	Опублікувати «відкритий ключ»	$\{e, n\}$ = {7, 527}
	Зберегти «закритий ключ»	$\{d, n\} = \{343, 527\}$
Шифрування	Обрати текст для шифрування	$m = 356$
	Визначити шифротекст	$c = E(m) = m^e\ mod\ n =$ (формула 2.8) $= 356^7\ mod\ 527 = 271$
Розшифрування	Визначити вихідне повідомлення	$m = D(c) = c^d\ mod\ n =$ (формула 2.9) $= 271^{343}\ mod\ 527 = 356$

Гарантія надійності цього алгоритму полягає в тому, що третій особі, яка намагається зламати шифр, дуже важко вирахувати закритий ключ по відкритому. Обидва ключі обчислюються з однієї пари простих чисел, проте важко встановити цей зв'язок. Основною проблемою стає декомпозиція числа n на множники.

Проте коли ми шифруємо повідомлення маємо на увазі не числа, а букви, слова у цифровому представлені. Одна й та сама буква буде шифруватися одним і тим самим числом таким чином, якщо зловмисник перехопить досить велике повідомлення по однаковим кодам можна здогадатися про пробіли та поділити шифротекст на слова, згодом визначити букви, які там зашифровані. Таким чином йому зовсім не потрібно буде визначати секретні ключі. Для уникнення таких ситуацій використовуються спеціальні додаткові алгоритми суть яких в тому, що кожна попередня частина повідомлення впливає на

наступну. Тобто числове представлення кожної попередньої букви додається до наступної, ми застосовує таке правило:

$$b := (b + a)\ mod\ n \qquad (2.10)$$

Де a – попередня частина повідомлення, b – наступна. Тоді послідовність чисел є більш заплутаною. Також в вихідне повідомлення додають випадкові букви на початок, щоб навіть по першому коду неможливо було нічого зрозуміти.

Шифрування інформації відбувається блоками певної довжини. Блоки на які розбивається повідомлення зазвичай більше однієї букви, тому спершу застосовуються алгоритми вирівнювання, потім алгоритми розбиття на блоки з заплутуванням і тільки після цього застосовують сам алгоритм RSA.

2.4 Алгоритм хеш-функцій SHA-3

Криптографічна хеш-функція – це математичний алгоритм перетворюючий будь-які вхідні дані в серію вихідних символів фіксованої або змінної довжини (рис. 2.14). В алгоритмах хешування з фіксованою вихідною довжиною ця довжина буде однаковою незалежно від розміру вхідних даних.

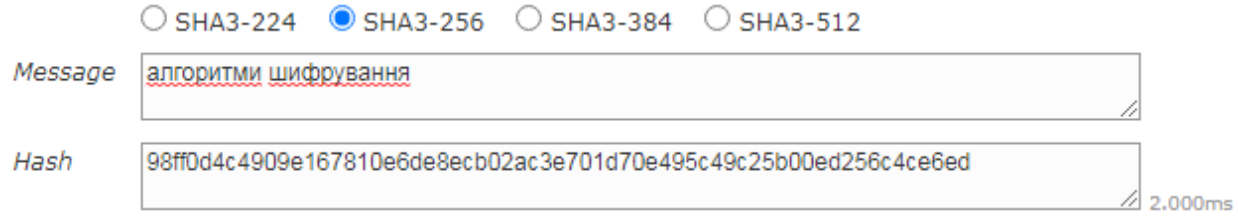

Рисунок 2.14 – Приклад роботи алгоритму

В хеш-функціях існує 2 поняття даних. Результат дії хеш-функції називається хеш-образом, а масив початкових даних називають прообразом [24].

Будь-яка криптографічна хеш-функція $h(x)$ повинна задовольняти наступним вимогам:

– Хеш-образ повинен залежати від усіх біт прообразу і від їх взаємного розташування;

– При будь-якій зміні вхідної інформації, хеш-образ повинен змінюватись непередбачувано, інакше кажучи, в середньому повинна змінюватися половина біт хеш-образу;

– При заданому значенню прообразу задача знаходження хеш-образа повинна бути обчислювально вирішувана;

– При заданому значенню хеш-образу задача знаходження прообразу повинна бути обчислювально невирішувана;

– Задача знаходження колізій хеш-функції повинна бути обчислювально невирішуваною.

SHA – означає алгоритм безпечного хешування і використовується для криптографічної безпеки. Найважливіша якість цього алгоритму – незворотність та унікальність хешу. Незворотність – вхідні дані залишаться в безпеці і невідомими. Унікальність – два різних фрагмента даних не можуть генерувати один і той самий ключ. SHA-3 удосконалена версія алгоритмів SHA-1 та SHA-2. [19] SHA-3 також іменований Keccak являє собою односпрямовану функцію для створення цифрових даних фіксованої довжини з вхідних даних будь-якого розміру. Функція Keccak побудована с використанням конструкції з назвою Sponge (губка), яка показана на рис. 2.15.

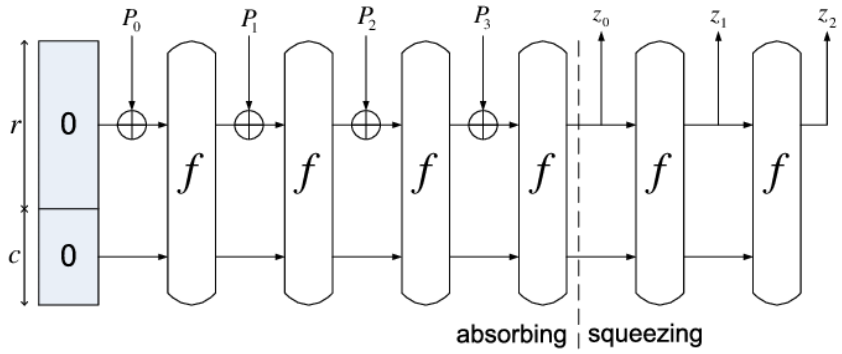

Рисунок 2.15 – Конструкція Sponge

Губчата функція Keccak використовує одну з семи можливих перестановок Keccak – f, які позначають Keccak

f[b] (2.11)

де bÎ{25, 50,100,200,400,800,1600}

Keccak – f перестановки являють собою ітераційні конструкції які складаються з послідовності майже однакових раундів. Кількість раундів залежить від ширини перестановки

$$n_r = 12 + 2^l \quad (2.12)$$

де $2^l = b/25$

В якості стандарту SHA-3 була вибрана перестановка Keccak – f[1600] для неї кількість раундів $n_r = 24$.

Схема Sponge має внутрішній стан S розрядністю b, яка поділяється на r та c частини , які називаються швидкістю та ємністю стану відповідно

$$b = r + c \quad (2.13)$$

Для SHA3-256 $r = 1088, c = 512$.

В SHA-3внутрішній стан S представлений у вигляді масиву $5 * 5$ слів довжини $w = 64$ біт [21].

Процес хешування складається з двох багатораундових етапів: ввід (Absorbing, поглинання) и вивід (Squeezing, витискання).

Етапи вводу інформації:

– Стан S ініціалізується нулями. За допомогою функції доповнення вхідне повідомлення M доповнюється до рядка довжини кратної r;

– Рядок P ділиться на n блоків довжини r;

– Перший блок повідомлення M складається по модулю 2 з рядком стану S, а саме з r частиною;

– Результат додавання та чистина c рядка стану S подається в функцію перестановки f і виходить новий рядок стану S;

– Другий блок повідомлення M знову складається по модулю 2 з першими r бітами виходу функції F;

– Результат операції XOR і останні c біт результату дії функції f знов подаються у функцію перестановки другого раунду. [20]

Цей етап продовжується доки не будуть оброблені всі блоки повідомлення M.

Особливістю цього етапу є те, що у кожному раунді блок повідомлення складається за модулем 2 тільки з частиною стану, а функція f виконує перетворення всього стану і робить його залежним від усього повідомлення M.

Етапи виводу інформації:

Поки довжина результату z менша за d, де d – кількість біт в вихідному масиві хеш-функції, r перших біт рядка стану S додається до результату z. Ця операція повторюється кінцеве число разів, при цьому над r бітами, зчитаними на кожному етапі, виконується операція конкатенація. Повторення відбувається доти, поки не буде отримано результат, тобто хеш-образ, потрібної довжини.

Функція доповнення.

Перш ніж хешувати повідомлення необхідно, щоб воно мало певну довжину і якщо її недостатньо то виконується операція доповнення. Спершу необхідно вирахувати якому розміру повинен відповідати наш хеш. Вирахувати розмір стану можна за формулою:

$$b = 25 * 2^l \tag{2.14}$$

де b-розмір стану

значення $l = \{0,1,2,3,4,5,6\}$

Для SHA-3 значення l = 6. Тепер знаючи значення l можна визначити скільки раундів необхідно виконати для кожної частини доповненого повідомлення [22].

Кількість раундів = $12 + 2 * l = 12 + 12 = 24$

Доповнення має бути зроблено таким чином щоб довжина доповненого повідомлення була кратна r для відповідної хеш-функції. На рис. 2.16 позначено відповідність між кількістю біт в хеш-образах алгоритмів та значенням r та c.

Type	Output length	Rate, r	Capacity, c
SHA3-224	224	1152	448
SHA3-256	256	1088	512
SHA3-384	384	832	768
SHA3-512	512	576	1024

Рисунок 2.16 – Відповідність біт в хеш-образах

Перший і останній біт будуть заповнені «1», всі інші між ними будуть «0». Після заповнення вони поділяються на n частин, де

$$n * r \tag{2.15}$$

еквівалентно довжині доповненого повідомлення. Математичного виглядає так:

$$p = n * r \tag{2.16}$$

де p-довжина повідомлення після доповнення;

n-кількість частин на які ми поділяємо p;

r-довжина швидкості.

Функція перестановок.

Кожна функція перестановки в алгоритмі SHA3-256 виконує 24 раунди, в кожному раунді виконується по 5 кроків перестановки бітів. Так як стан S має форму тривимірного масиву 5*5*64, позначимо $a\,[x][y][z]$. В його складі виділяють стовпці, рядки, лінії та зрізи як показано на рис. 2.17.

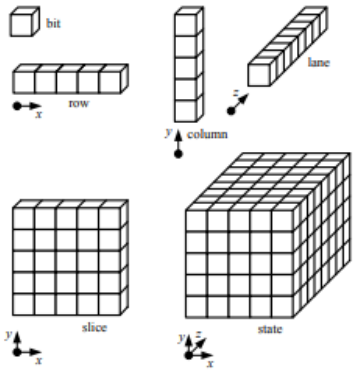

Рисунок 2.17 – Тривимірний масив S

1. Крок θ

Додає до кожного біта $a\,[x][y][z]$ побітову суму двох стовпців

$$a\,[x-1][\cdot][z] \tag{2.17}$$

та

$$a\,[x-1][\cdot][z-1] \tag{2.18}$$

Тобто додає по модулю 2 кожний біт внутрішнього стану с кожним бітом двох суміжних стовпців. На рис. 2.18 схематично зображено виконання цієї операції.

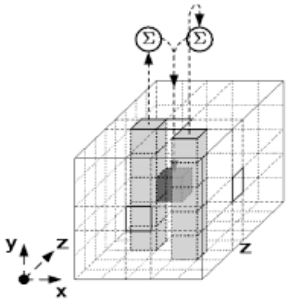

Рисунок 2.18 – Схематичне зображення кроку θ

2. Крок ρ

Циклічно зсуває рядки внутрішнього стану на різну кількість бітів в залежності від номера рядка. (рис. 2.19)

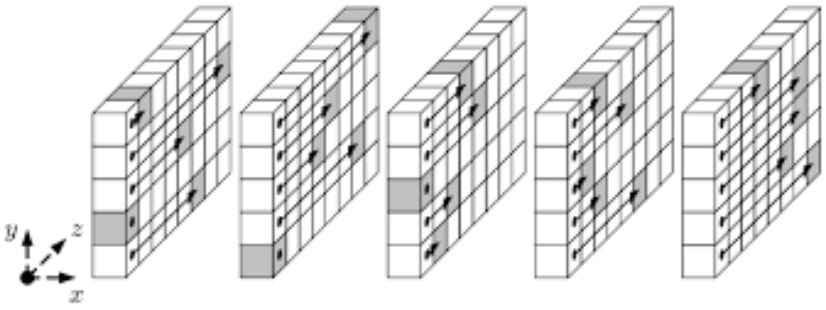

Рисунок 2.19 – Схематичне зображеня кроку ρ

На таблиці 2.4 продемонстровано величини зсуву ліній для всіх можливих значень $\binom{x}{y}$.

Таблиця 2.4 – Величини зсуву ліній

-	x=3	x=4	x=0	x=1	x=2
y=2	25	39	3	10	43
y=1	55	20	36	44	6
y=0	28	27	0	1	62
y=4	56	14	18	2	61
y=3	21	8	41	45	15

3. Крок π

Сутність перетворення – перестановка ліній у площині (x, y):

$$\begin{pmatrix} x \\ y \end{pmatrix} \leftarrow \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} y \\ 2x + 3y \end{pmatrix} \quad (2.19)$$

На рис. 2.20 схематично зображено перетворення π

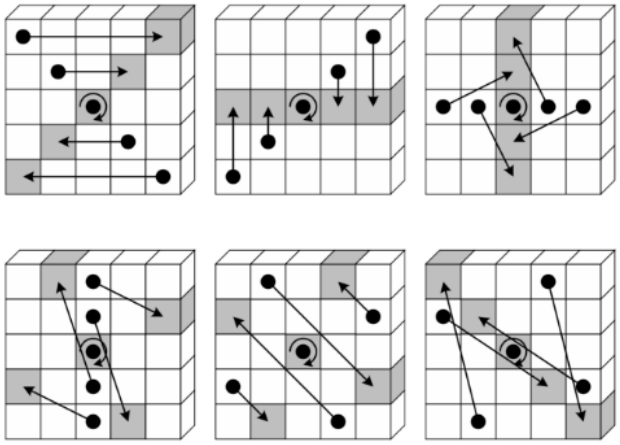

Рисунок 2.20 – Схематичне зображення кроку π

4. Крок χ

Ця функція складає по модулю 2 кожний біт з нелінійною функцією двох інших біт того ж рядка:

$$\chi(a) = a(x, y, z)\text{XOR}\left((\text{a}[(\text{x} + 1)\text{mod } 5, \text{y}, \text{z}]\text{XOR } 1) * a\begin{bmatrix}(x+2) \\ mod\ 5, y, z\end{bmatrix}\right) \quad (2.20)$$

Схематично дію функції показано на рис. 2.21

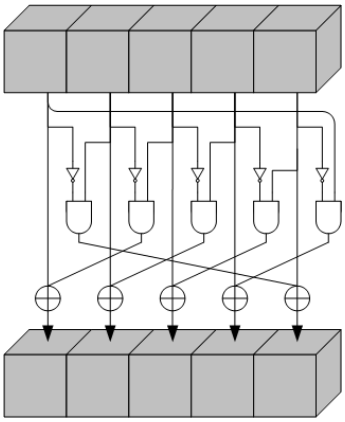

Рисунок 2.21 – Схематичне зображення кроку χ

5. Крок ι

Складається з складання з раундовими константами і направлено на порушення симетрії. Нижче наведено рис. 2.22 раундових констант $RC[i]$ для $w = 64$ біт.

Table 1: The round constants RC[i]

RC[0]	0x0000000000000001	RC[12]	0x000000008000808B
RC[1]	0x0000000000008082	RC[13]	0x800000000000008B
RC[2]	0x800000000000808A	RC[14]	0x8000000000008089
RC[3]	0x8000000080008000	RC[15]	0x8000000000008003
RC[4]	0x000000000000808B	RC[16]	0x8000000000008002
RC[5]	0x0000000080000001	RC[17]	0x8000000000000080
RC[6]	0x8000000080008081	RC[18]	0x000000000000800A
RC[7]	0x8000000000008009	RC[19]	0x800000008000000A
RC[8]	0x000000000000008A	RC[20]	0x8000000080008081
RC[9]	0x0000000000000088	RC[21]	0x8000000000008080
RC[10]	0x0000000080008009	RC[22]	0x0000000080000001
RC[11]	0x000000008000000A	RC[23]	0x8000000080008008

Рисунок 2.22 – Раундові константи

Безпека алгоритму хеш-функції SHA-3 полягає в тому, що за своєю структурою та принципом роботи він відповідає всім вимогам поставленим до засобів захисту даних та хеш-функцій [26].

2.5 Обґрунтування вибору алгоритмів

1. Advanced Encryption Standard (AES) – симетричний алгоритм блочного шифрування, прийнятий в якості стандарту шифрування правлінням США по результатах конкурсу. Цей алгоритм добре проаналізований і зараз широко використовується. За станом на 2017 рік AES є одним з найрозповсюдженіших алгоритмів симетричного шифрування. Дизайн AES заснований на принципі substitution-permutation network (мережі замін-перестановок). Він швидкий як для програмного так і для апаратного забезпечення. Ще в червні 2003 року Агенція національної безпеки США постановила факт, що шифр AES є досить надійним щоб використовувати його для захисту інформації, яка складає державну таємницю. Аж до рівня SECRET було дозволено використовувати ключі довжиною 128 біт. У порівнянні з DES, AES має більш криптостійкий ключ (128-256 біт проти 56 у DES).

Безпека алгоритму полягає в тому, що для 128-бітного ключа повний перебір вимагає 2^{127}операцій. Це свідчить, що атака з використанням перебору ключів на сьогодні не має практичного значення. А також при шифруванні 128-розрядним ключем алгоритм виконує 10 раундів. Криптографи проаналізували і виявили, що починаючи вже з 8-го раунду алгоритм стає досить стійким до атак (рис. 2.23) і ще після цього виконує ще 2 раунди перестановок.

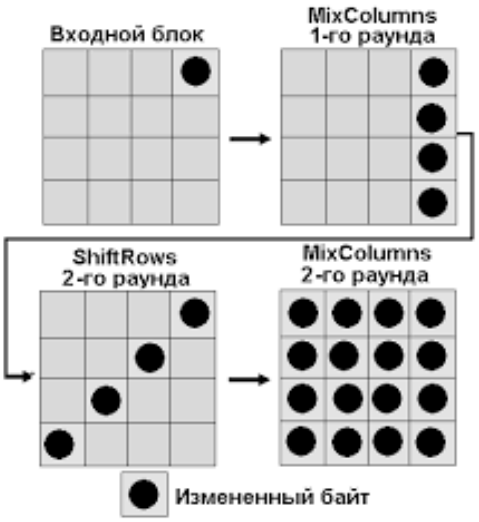

Рисунок 2.23 – Кількість змінених байт

Шифр стійкий проти таких видів криптоаналітичних атак:

– Диференційного криптоаналізу;

– Лінійного криптоаналізу;

– Криптоаналізу на основі зв’язаних ключів.

AES при всій своїй простоті, дуже потужний алгоритм шифрування, який поки що не вдавалося зламати.

2. Назва RSA складається з перших букв прізвищ Рівест, Шамір і Адлеман – вчених, які вперше його опублікували. Принцип функціонування даної криптосистеми в тому що застосовуються 2 різні ключі – відкритий і закритий. Перший використовують для шифрування інформації, другий для розшифрування.

Криптостійкість алгоритму RSA заснована на важкості розкладання на множники великих чисел, а саме – на виключній важкості задачі визначити секретний ключ на основі відкритого, так як для цього необхідно вирішити задачу про існування дільників цілого числа. RSA є безпечним алгоритмом, якщо прості числа, які використовуються для генерації ключів, досить великі. В цілях безпеки цілі числа повинні бути вибрані випадковим чином і бути однаковими за величиною, але при цьому відрізнятись за довжиною на декілька цифр, щоб зробити факторинг складніше. Якщо основне число 300 біт або коротше, шифротекст може бути розкладений протягом декількох годин. Ключі довжиною 512 біт можливо розкласти протягом декількох тижнів з використанням загальнодоступного апаратного забезпечення. В даний час рекомендується мати довжину ключа не менше 2048 біт.

3. В даний момент алгоритм SHA-3 не використовується для шифрування інформації під час обміну повідомленнями. Проте цей алгоритм використовують в області криптовалюти. Хеш-функція SHA-3 використовується на рівні блоку, щоб генерувати задачу доказів роботи, виконання якої стає ціллю для майнерів, яким необхідно створити наступний блок в блокчейні. Криптовалюту в якості винагороди отримує майнер який успішно обраховує хеш SHA-3, який відповідає вимогам. На відміну від хеш-функцій Біткоїна і інших блокчейнів, SHA-3 був розроблений в рамках спільної

роботи NIST, хеш-функція проходила суспільний контроль та вичерпне тестування, щоб бути стандартом тестування [29]. SHA-3 маючи конструкцію криптографічної губки, дозволяє вводити та виводити будь-які об’єми даних, розширюючи вихідну функцію і роблячи її більш гнучкою в користування. SHA-3 легко перевершує SHA-1 та SHA-2 за швидкість хешування на апаратних компонентах. SHA-3 помітно відрізняється від першого та другого стандарту, які мають однакову математичну та криптографічну структуру [28].

Висновки до розділу 2

У другому розділі було розглянуто та проаналізовано методи шифрування як спосіб захисту даних та їх класифікацію. В ході написання розділу було визначено та обрано для розгляду 3 основні найпоширеніші на даний час алгоритми шифрування інформації:

- симетричний алгоритм AES;
- асиметричний алгоритм RSA;
- алгоритм хеш-функції SHA-3.

Було проаналізовано та описано математичне обґрунтування цих алгоритмів.

Симетричний алгоритм AES-128 складається з 10 раундів в кожному з яких відбувається 4 типи перестановок вхідних байтів даних. Стійкість криптоалгоритму полягає в тому що необхідно перебрати 2^{128}. Проте недоліком є використання однакового ключа як для шифрування так і для дешифрування, якщо ключ опиниться у злочинців дані будуть втрачені.

Асиметричний алгоритм RSA заснований на проблемі розкладання великих простих чисел на множники. Має декілька етапів перший з яких це вирахування з двох великих простих чисел відкриту та закриту експоненти, які є частинами відкритого та закритого ключів відповідно. Потім вже йде етап шифрування інформації за допомогою публічного ключа. Потім отримувач розшифровує повідомлення за допомогою закритого ключа. Стійкість

криптоалгоритму полягає у його концепції двох ключів та використанні великих простих чисел. Недоліками є повільне виконання та потреба у використанні все більшої довжини ключа у зв’язку з ростом потужності апаратного забезпечення, яке вдало та досить швидко вирішує проблему розкладу великих простих чисел.

Алгоритм хеш-функції SHA-3 має конструкцію губки, яка спочатку вбирає в себе вхідні дані, потім за допомогою чисельних перестановок міняє місцями вхідні байти, а потім наче губка витискає зашифрований рядок певної довжини, який називається хешем. з якого неможливо отримати назад вхідні дані. Алгоритм виконує 24 раунди, в кожному раунді виконується по 5 кроків перестановки бітів. Стійкість криптоалгоритму в тому, що навіть якщо зловмисник отримає зашифрований рядок даних він не зможе з нього дізнатись ніяких корисних даних, відсутність колізій. Також неможливість розшифрування даних за допомогою алгоритму є його і недоліком, що дозволяє користуватись цим алгоритмом тільки для певних цілей, наприклад зберігання паролів, інформації по запиту.

РОЗДІЛ 3
ЗАСТОСУВАННЯ АЛГОРИТМІВ ШИФРУВАННЯ

3.1 Основні механізми захисту даних

Механізм захисту даних – сукупність засобів і прийомів, направлених на гарантування інформаційної безпеки або інформаційної надійності.

Забезпечення інформаційної безпеки досягається шляхом наступних дій, які направлені на:

– Виявлення загроз. Виражається в аналізі і контролі допустимих появ потенційних або реальних загроз, а також у своєчасних заходах з їх попередження.

– Попередження загроз. Досягається шляхом забезпечення інформаційної безпеки на користь попередження їх виникнення.

– Включення заходів зі знищення загроз або злочинних дій та локалізацію цих дій.

– Виявлення загроз досягається шляхом визначення конкретних злочинних дій і реальних загроз.

– Ліквідацію наслідків відносно загроз і злочинних конкретних дій.

Основними механізмами забезпечення захисту даних є:

– Перешкоджання – метод фізичного перешкоджання шляху зловмисника до інформації, що захищається.

– Управління доступом – механізм захисту даних заснований на регулюванні використання всіх ресурсів ІС. Ці методи мають протистояти всім можливим шляхам несанкціонованого доступу до даних. цей механізм включає в себе деякі функції захисту: ідентифікацію користувачів, впізнання, перевірку повноважень, дозвіл і створення умов роботи в межах встановленого регламенту, реєстрацію звернень до ресурсів, що захищаються, реагування при спробах несанкціонованих дій.

– Механізми шифрування – криптографічне закриття інформації. Ці методи захисту все більше застосовуються як при розробці, так і при збереженні інформації на носіях. При передачі даних по каналах зв'язку цей метод є єдино надійним.

– Протидія атакам шкідливим програм передбачає комплекс різноманітних заходів організаційного характеру і використання антивірусних програм. Цілі вжитих заходів – це зменшення вірогідності зараження системи, виявлення факторів зараження системи, зменшення наслідків інформаційних інфекцій, локалізація та знищення вірусів, відновлення даних в системі.

– Регламентація – створення таких умов автоматизованої обробки, збереження та передачі даних, що захищаються, при яких норми та стандарти захисту виконуються найбільшою мірою.

– Примушення – механізм захисту при якому користувачі та персонал вимушені дотримуватися правила обробки, передачі і використання інформації, що захищається під загрозою матеріальної, адміністративної або кримінальної відповідальності.

– Спонукання – механізм захисту, спонукаючий користувачів та персонал не порушувати встановленні порядки за рахунок дотримання сформованих моральних та етичних норм.

3.2 Визначення рівня захисту даних в алгоритмі

Чимало алгоритмів шифрування з відкритим ключем спираються на дуже великі числа, які в свою чергу отримуються шляхом перемноження двох або більше простих чисел. Прості числа в даному випадку слугують секретними даними, які в свою чергу допомагають отримати потрібний ключ і розшифрувати повідомлення. Розкладання досить великих чисел на прості множники і вирахування дискретного логарифму – це задачі, які не підлягають взлому, а тільки вирахуванню. У 2019 році новий рекорд поставлений спеціалістами, дозволив розкласти на прості числа ключ RSA-240 довжиною

240 десяткових розрядів або 795 біт. Попередній рекорд був поставлений в 2010 році тоді вдалося зламати ключ RSA розрядністю 768 біт. Ключ RSA 1024 біт в небезпеці і необхідно переходити на 2048-розрядні ключі. За словами дослідників, після їх роботи в якості надійної системи шифрування можна розглядати тільки RSA-ключі довжиною 2048 біта і більше. Причому від шифрування ключем довжиною в 2048 біт варто відмовитися в найближчі три-чотири роки. На перший крок (вибір пари поліномів степенів 6 і 1) було витрачено близько півроку обчислень на 80 процесорах, що склало близько 3% часу, витраченого на головний етап алгоритму (просіювання), який виконувався на сотнях комп'ютерів протягом майже двох років. Якщо інтерполювати цей час на роботу одного процесора AMD Opteron 2.2ГГц з 2Гб пам'яті, то вийшло б близько 1500 років. Обробка даних після просіювання для наступного ресурсоємного кроку (лінійної алгебри) було потрібно декілька тижнів на малій кількості процесорів. Заключний крок після знаходження нетривіальних рішень ослу зайняв не більше 12 годин. Рішення проводилося методом Відемана на декількох роздільних кластерах і тривало трохи менше 4 місяців. Розмірність матриці при цьому виявилася 192 796 550x192 795 550 при наявності 27 795 115 920 ненульових елементів (тобто в середньому 144 ненульових елементів на рядок). Для зберігання матриці на жорсткому диску знадобилося близько 105 гігабайт. У той же час знадобилося близько 5 терабайт стислих даних для побудови даної матриці. Групі вчених вдалося обчислити 232-цифровий ключ, що відкриває доступ до зашифрованих даних. Дослідники впевнені, що використовуючи їх метод факторизації, зламати 1024-бітний RSA-ключ буде можливо. Знаючи розклад на добуток двох простих чисел, зловмисник зможе визначити секретну експоненту і зламати RSA. Для того щоб зрозуміти чому вкрай не рекомендується використовувати RSA в його найбільш простій формі спершу зазначимо які вимоги висувають до асиметричних криптосистем.

1. Зашифроване повідомлення повинно піддаватися дешифруванню тільки за наявності ключа;

2. Число операцій, необхідних для визначення використаного ключа шифрування за фрагментом шифрованого повідомлення і відповідного йому відкритого тексту, повинно бути менше загального числа можливих ключів;

3. Знання алгоритму шифрування не повинно впливати на надійність захисту;

4. Число операцій, необхідних для розшифрування інформації шляхом перебору можливих ключів повинно мати сувору нижню оцінку і виходити за межі можливостей сучасних комп'ютерів;

5. Незначна зміна ключа повинна призводити до суттєвої зміни вигляди зашифрованого повідомлення навіть при використанні одного й того самого ключа;

6. Структурні елементи алгоритму шифрування повинні бути незмінними;

7. Додаткові біти, які вводяться в повідомлення в процесі шифрування, повинні бути повністю і надійно приховані в криптотексті;

8. Не повинно бути простих і легко встановлюваних залежностей між ключами, послідовно використовуваних в процесі шифрування;

9. Будь-який ключ з множини можливих повинен забезпечувати надійний захист інформації;

10. Алгоритм повинен допускати як програмну так і апаратну реалізацію, при цьому зміна довжини ключа не повинно вести до якісному погіршенню алгоритма шифрування;

11. Розшифрування повинно відновлювати відкритий текст;

12. Функції знаходження параметрів відкритого ключа e та закритого ключа d повинні бути простими в реалізації;

13. При розкритті перетворення, яке виконується за допомогою функції параметру e, не повинно розкриватися перетворення за допомогою функції параметру d, тобто із відкритого ключа не можна отримати закритий;

14. Можливість використання закритого ключа для шифрування, а відкритого для дешифрування;

Зараз можна сказати, що RSA у всіх своїх проявах будь то PGP або SSL не шифрує тільки відправлені на вхід шифрувальної функції дані. Алгоритм спершу додає до цих даних блоки, які містять випадковий набір біт. І тільки після цього отриманий результат шифрується. Тобто замість звичної всім

$$c = E(m) = m^e \ mod \ n \quad (3.1)$$

отримуємо ближчу до дійсності

$$c = E(m) = (m||rand)^e \ mod \ n \quad (3.2)$$

де Rand випадкове число.

Таку методику називають схемами доповнення. В даний час використання RSA без схем доповнення є не стільки поганим тоном, скільки безпосередньо порушенням стандартів.

3.3 Розробка інтерфейсу користувача

Для реалізації кваліфікаційної роботи обрано платформу .NET, отже розробка всіх модулів системи буде виконуватись за допомогою мови програмування C#.

Мова програмування C# – це мова програмування, що поєднує об'єктно-орієнтовані і контекстно-орієнтовані концепції. Для того щоб було зручно користуватись алгоритмом необхідно створити користувацький інтерфейс.

Деякі складові ІК описані на рис. 3.1.

Рисунок 3.1 – Складові інтерфейсу користувача

Для розробки інтерфейсу користувача необхідно у середовищі Visual Studio створити проект «додаток для Windows Form».

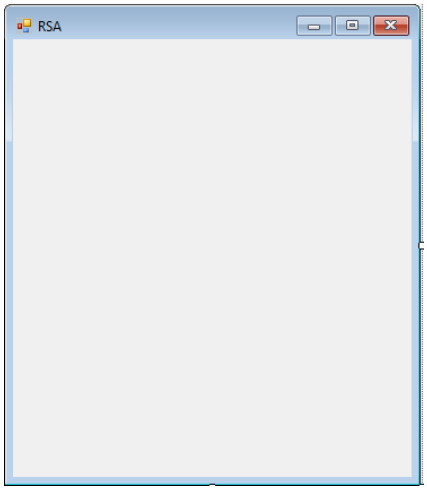

Рисунок 3.2 – Пуста форма інтерфейсу

На рис. 3.2 показана пуста форма для заповнення і створення інтерфейсу користувача.

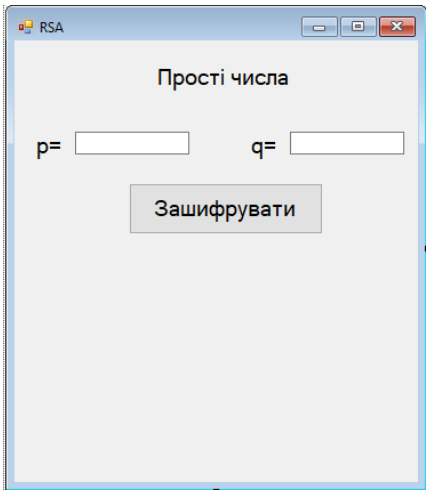

Рисунок 3.3 – Створення полів вводу простих чисел

На рис. 3.3 зображено створення полів для введення простих чисел та кнопки за допомогою якої відбуватиметься шифрування даних. Для зручності користувачів над полями введення зроблено надпис, що саме прості числа необхідно вводити у форму. Потім до кнопки «Зашифрувати» будуть додані функції, які і будуть виконувати шифрування даних.

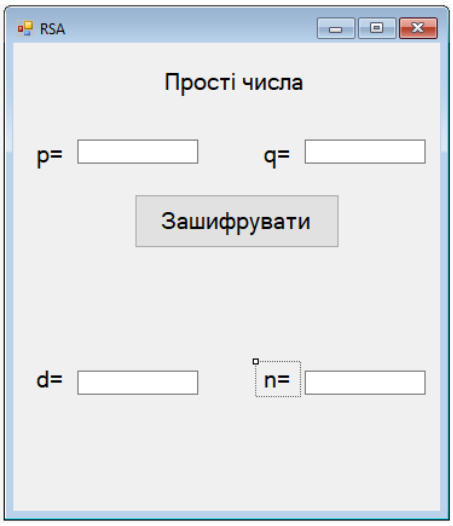

Рисунок 3.4 – Створення полів вводу секретного ключа

На рис. 3.4 показано, що далі було створено ще одні поля для вводу секретного ключа із підписами параметрів, які користувачу потрібно туди вписати.

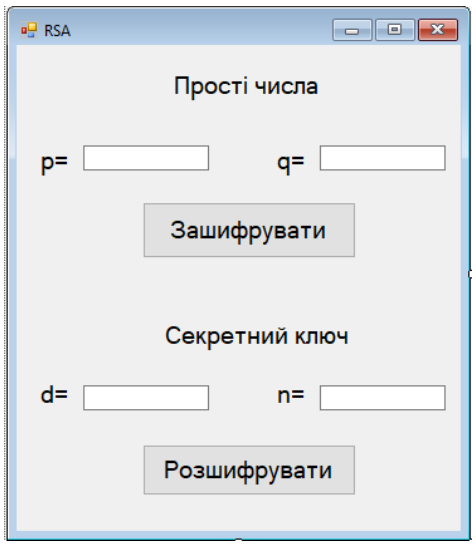

Рисунок 3.5 – Готова форма ІК

З рис. 3.5 зрозуміло, що для полегшення і зрозумілості користування цією формою було додано, ще підпис, які саме дані необхідно вводити в поля, тому що для користувача, який не розуміється на математичних формулах, може бути не зрозумілим, що саме означають параметри d та n в алгоритмі. Також створено кнопку «Розшифрувати» за допомогою якої відбуватимуться дешифрувальні дії згідно з алгоритмом.

3.4 Розробка основних модулів системи

Мова програмування C# – це мова програмування, що поєднує об'єктно-орієнтовані і контекстно-орієнтовані концепції. Розроблено в 1998-2001 роках групою інженерів під керівництвом Андерса-Хейлсберга в компанії Microsoft як основна мова розробки додатків для платформи Microsoft .NET. C # відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML.

Перейнявши багато від своїх попередників – мов C++, Delphi, Modula і Smalltalk – C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем: так, C# не підтримує множинне успадкування класів (на відміну від C++), або виведення типів (на відміну від Haskell).

Ще деякі переваги C#:

– Лямбда-функції, які набагато краще, ніж внутрішні анонімні класи.

– Делегати. У C# є делегати, які по суті є методами, які можуть бути викликані без повідомлення цільового об'єкта.

– Перевантаження операторів. Це штука може перетворити все в пекло, але вона до цих пір зручна, особливо в бібліотеках.

– Властивості. Немає потреби писати getters і setters. Все виглядає, як пряме звернення, навіть якщо це не так, foo.x + = 1 більш читабельно, ніж foo.setX (foo.getX () + 1)

– Підтримка yield в ітераціях.

– Методи розширення. Це дозволяє вам розширити існуючі класи, навіть якщо вони готові, без реального розширення.

– Підтримка оператора, який пропонує простий синтаксис отримання значення посилального типу, якщо воно не дорівнює null або вказане значення за замовчуванням.

Для початку необхідно створити масив символів (Лістинг 3.1), за допомогою якого відбуватиметься шифрування.

Лістинг 3.1 – Масив символів

```
public partial class Form1 : Form
   {
      char[] characters = new char[] {' ', 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Є', 'Ж', 'З', 'И', 'І','Ї',
                                      'Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф',
                                      'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ь', 'Ю', 'Я',
                                      '1', '2', '3', '4', '5', '6', '7','8', '9', '0' };
```

Потім за допомогою циклів if і while створюємо блок із кодом програми в якому описані дії, які відбуватимуться при нажаті на кнопку «зашифрувати» (Лістинг3.2) та при яких умовах алгоритм буде працювати.

Лістинг 3.2 – Шифрування даних

```
public Form1()
      {
         InitializeComponent();
      }
      private void buttonEncrypt_Click(object sender, EventArgs e)
      {
         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!");
   }
```

Далі створено схожий блок, але вже для перевірки даних при розшифруванні прив’язаний до кнопки «Розшифрувати». (Лістинг3.3).

Лістинг 3.3 – Блок розшифрування

```
private void buttonDecipher_Click(object sender, EventArgs e)
   {
      if ((textBox_d.Text.Length > 0) && (textBox_n.Text.Length > 0))
      {
         long d = Convert.ToInt64(textBox_d.Text);
         long n = Convert.ToInt64(textBox_n.Text);

         List<string> input = new List<string>();

         StreamReader sr = new StreamReader("out1.txt");
         while (!sr.EndOfStream)
         {
            input.Add(sr.ReadLine());
         }
         sr.Close();

         string result = RSA_Dedoce(input, d, n);
         StreamWriter sw = new StreamWriter("out2.txt");
         sw.WriteLine(result);
         sw.Close();
         Process.Start("out2.txt");
      }
      else
         MessageBox.Show("Введіть секретний ключ!");
   }
```

Далі на лістингу 3.4 продемонстровано код перевірки чисел, які ввів користувач, вони повинні бути простими, використовуючи умовний оператор if.

Лістинг 3.4 – Перевірка простих чисел.

```
private bool IsTheNumberSimple(long n)
      {
         if (n < 2)
            return false;
         if (n == 2)
            return true;

         for (long i = 2; i < n; i++)
            if (n % i == 0)
               return false;
         return true;
      }
```

В основі алгоритму RSA лежить принцип використання відкритого і закритого ключа для забезпечення більш надійного захисту інформації, яку шифрують користувачі. Отже у лістингу 3.5 описано розрахунок параметру *e*, який є елементом відкритого ключа, та який необхідний для обрахування закритого ключа.

Лістинг 3.5 – Обрахування параметру *e*

```
private long Calculate_e(long d, long m)
{
   long e = 10;
   while (true)
   {
      if ((e * d) % m == 1)
         break;
      else
         e++;
   }
   return e;
```

Як сказано вище за формулою алгоритму він має два елементи: закритий і відкритий ключ. Зараз на лістингу 3.6 продемонстровано обрахунок параметру *d*, який є елементом закритого ключа.

Лістинг 3.6 – Обрахунок параметру *d*

```
private long Calculate_d(long m)
    {
       long d = m - 1;
       for (long i = 2; i <= m; i++)
          if ((m % i == 0) && (d % i == 0)) //якщо мають спільні дільники
          {
             d--;
             i = 1;
          }
       return d;
    }
```

Далі (Лістинг 3.7) представлений блок коду, який безпосередньо виконує алгебраїчні обчислення за формулою алгоритму та виконує функції шифрування даних.

Лістинг 3.7 – Блок шифрування

```
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;
    }
```

Для повноцінної роботи алгоритму шифрування даних необхідно створити також блок дешифрації даних, який буде виконувати функції розшифрування введених користувачем даних,і який буде реалізовано аналогічно до блоку шифрування використовуючи певні необхідні математичні формули відповідно до умов алгоритму. Створення блок розшифрування представлено на лістингу 3.86.

Лістинг 3.8 – Блок дефишрування

```
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;
      }
```

Висновки до розділу 3

У третьому розділі були розглянуті та проаналізовані основні механізми захисту даних. Також було детальніше розглянуто рівень захисту даних в алгоритмі RSA та розроблено демонстраційний приклад для поліпшення уяви принципу його роботи. Розробка відбувалася у середовищі Visual Studio за допомогою мови програмування C#. Алгоритм RSA один з найяскравіших та найзрозуміліших прикладів роботи асиметричних криптосистем тому вирішено було реалізувати саме цей алгоритм шифрування даних. У процесі розробки були складнощі з описанням на мові програмування математичних операцій.

РОЗДІЛ 4
ОЦІНКА РІВНЯ ЗАХИСТУ ДАНИХ

4.1 Методи тестування системи

Всі види тестування програмного забезпечення, залежно від переслідуваних цілей, можна умовно розділити на наступні групи:

- Функціональні (Functional testing).
- Нефункціональні (Non-functional testing).
- Пов’язані зі змінами (Regression testing).

Функціональні види тестування

Функціональні тести базуються на функціях і особливостях, а також взаємодії з іншими системами, і можуть бути представлені на всіх рівнях тестування: компонентному або модульному (Component / Unit testing), інтеграційному (Integration testing), системному (System testing) і приймальному (Acceptance testing). Функціональні види тестування розглядають зовнішню поведінку системи. Далі перераховані одні з найпоширеніших видів функціональних тестів:

- Функціональне тестування (Functional testing).
- Тестування безпеки (Security and Access Control Testing).
- Тестування взаємодії (Interoperability Testing).

Нефункціональні види тестування

Нефункціональне тестування описує тести, необхідні для визначення характеристик програмного забезпечення, які можуть бути виміряні різними величинами. В цілому, це тестування того, «як» система працює. Далі перераховані основні види не функціональних тестів:

Всі види тестування продуктивності:

- тестування навантаження (Performance and Load Testing)
- стресове тестування (Stress Testing);

– тестування стабільності або надійності (Stability / Reliability Testing);

– об’ємне тестування (Volume Testing);

– Тестування установки (Installation testing);

– Тестування зручності користування (Usability Testing);

– Тестування на відмову і відновлення (Failover and Recovery Testing);

– Конфігураційне тестування (Configuration Testing);

Тестування, пов’язане зі змінами

Після проведення необхідних змін, таких як виправлення бага/дефекту, програмне забезпечення повинне бути перетестоване для підтвердження того факту, що проблема була дійсно вирішена. Нижче перераховані види тестування, які необхідно проводити після установки програмного забезпечення, для підтвердження працездатності програми або правильності здійсненого виправлення дефекту:

– Димове тестування (Smoke Testing).

– Регресійне тестування (Regression Testing).

– Тестування збірки (Build Verification Test).

– Санітарне тестування або перевірка узгодженості/справності (Sanity Testing).

– Функціональне тестування (Functional testing).

Тестування функціональності може проводиться у двох аспектах:

- вимоги;
- бізнес-процеси.

Тестування в аспекті «вимоги» використовує специфікацію функціональних вимог до системи як основу для дизайну тестових випадків (Test Cases). У цьому випадку необхідно зробити список того, що буде тестуватися, а що ні, приорітезувати вимоги на основі ризиків (якщо це не зроблено в документі з вимогами), а на основі цього приорітезувати тестові сценарії (test cases). Це дозволить сфокусуватися і не упустити при тестуванні

найбільш важливий функціонал. Тестування в сенсі «бізнес-процеси» використовує знання цих самих бізнес-процесів, які описують сценарії щоденного використання системи. У цьому випадку тестові сценарії (test scripts), як правило, ґрунтуються на випадках використання системи (use cases).

Перевагою функціонального тестування є те, що воно імітує фактичне використання системи, а недоліком функціонального тестування є: можливість упущення логічних помилок у програмному забезпеченні та ймовірність надмірного тестування. Досить поширеною є автоматизація функціонального тестування.

Тестування безпеки (Security and Access Control Testing) – це стратегія тестування, що використовується для перевірки безпеки системи, а також для аналізу ризиків, пов’язаних із забезпеченням цілісного підходу до захисту додатків, атак хакерів, вірусів, несанкціонованого доступу до конфіденційних даних. Загальна стратегія безпеки ґрунтується на трьох основних принципах: конфіденційність, цілісність, доступність.

Конфіденційність – це приховування певних ресурсів або інформації. Під конфіденційністю можна розуміти обмеження доступу до ресурсу деякої категорії користувачів, або іншими словами, за яких умов авторизований користувач може отримати доступ до даного ресурсу.

Цілісність. Існує два основних критерії при визначенні поняття цілісності:

Довіра – очікується, що ресурс буде змінений тільки відповідним способом певною групою користувачів.

Пошкодження і відновлення – у разі коли дані пошкоджуються або неправильно змінюються авторизованим користувачем, потрібно визначити наскільки важливою є процедура відновлення даних.

Доступність являє собою вимоги про те, що ресурси повинні бути доступні авторизованому користувачеві, внутрішньому об’єкту, або пристрою. Як правило, чим більш критичний ресурс тим вище рівень доступності.

Тестування взаємодії (Interoperability Testing) – це функціональне тестування, що перевіряє здатність програми взаємодіяти з одним і більше компонентами або системами і включає в себе тестування сумісності (compatibility testing) і інтеграційне тестування (integration testing).

4.2 Розробка тестових даних для проекту

Тестування додатку для шифрування даних буде відбуватись за допомогою мануального методу тестування чорної скриньки.

При створені програмного продукту створюється перелік вимог яким буде відповідати продукт перевірка на відповідність програми до вимог також є тестуванням, отже перелік вимог, яким повинна відповідати дана програма:

– шифрування та дешифрування даних, які приймаються з документу;

– перевірка чисел, які вводить користувач прості, чи ні;

– правильність проведення обрахування за алгоритмом.

Тестування окремих елементів програми:

1. Тестування інтерфейсу користувача (рис. 4.1)

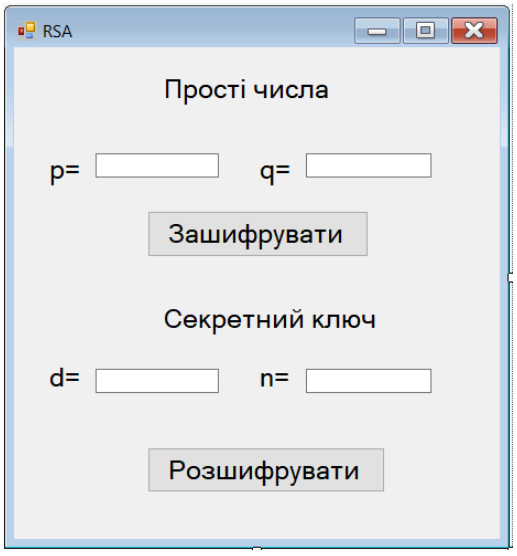

Рисунок 4.1 – Інтерфейс користувача

Для полегшення взаємодії користувача із програмою було створено користувацький інтерфейс. Із рис. 4.1 зрозуміло, що програма має 4 поля для вводу даних, 2 кнопки «Зашифрувати» та «Розшифрувати». Користувач у поля простих чисел повинен ввести будь-які 2 простих числа, після цього натиснути кнопку «Зашифрувати» і програма видасть результат шифровки текстових даних. Потім необхідно ввести правильний секретний ключ і натиснути кнопку «розшифрувати» тоді програма видасть результат дешифрації. Загалом інтерфейс користувача зрозумілий, але можуть виникнути проблеми в звичайних користувачів, які не розуміють, що це за параметри: p, q, d, та n.

2. Тестування дій, які відбуваються якщо користувач натиснув кнопку «Зашифрувати», але при цьому не було введено простих чисел. На рис. 4.2 зображено, що система попереджає користувача про невірність, або відсутність необхідних для шифрування даних.

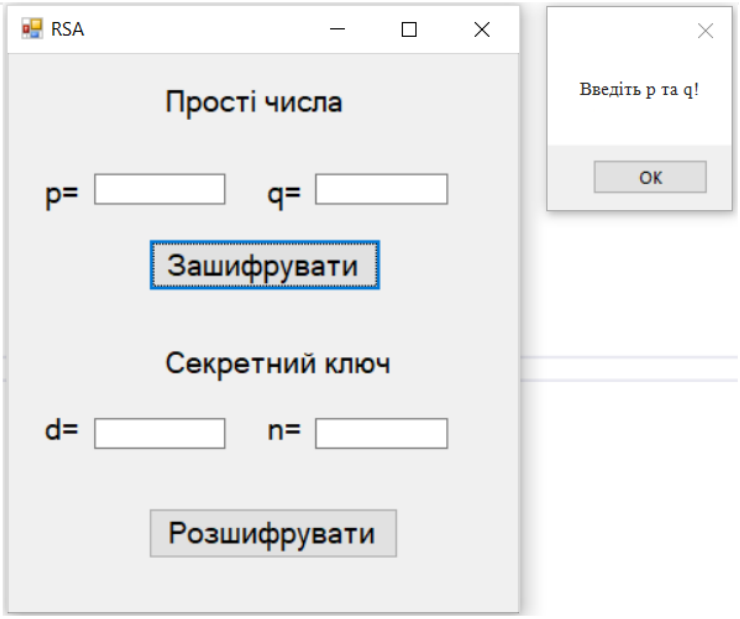

Рисунок 4.2 – Повідомлення про відсутність необхідних даних

3. Тестування полів вводу простих чисел. Користувач повинен ввести в ці поля 2 простих числа, якщо користувач вводить не прості числа, обрахування за алгоритмом не повинні проводитися далі.

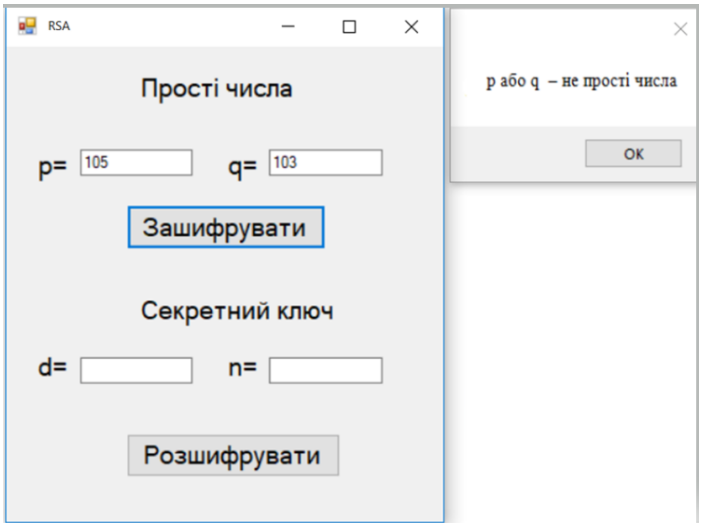

Рисунок 4.3 – Повідомлення системи про невірність необхідних чисел

З цього зображення (див. рис. 4.3) слідує, що при введені користувачем чисел, які не відповідають вимогам виконання алгоритму виводиться модальне діалогове вікно, яке повідомляє користувачу, що введені числа не є простими, а отже не відповідають умовам виконання алгоритму.

4. Перевірка, чи приймає взагалі програма прості числа змальована на зображені рис. 4.4. Також при правильному введені простих чисел користувачем автоматично повинні обчислюватись та записуватись параметри секретного ключа.

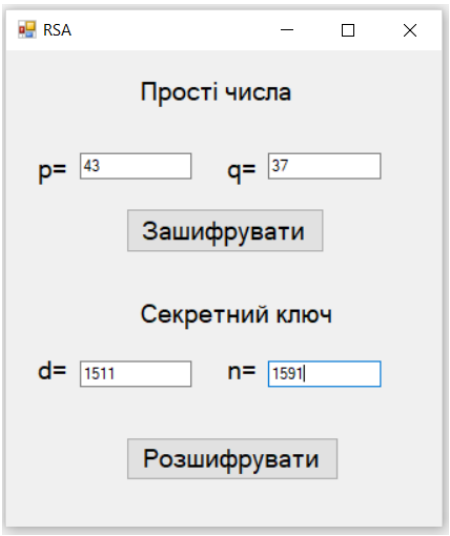

Рисунок 4.4 – Робота програми при правильних діях користувача

Із рис. 4.4 зрозуміло, що програма працює правильно, після того як користувач вводить правильно прості числа в поля секретного ключа автоматично вводяться необхідні параметри і виводиться текстовий файл (рис. 4.5) в якому відображено числа, які утворюються в результаті шифрування повідомлення.

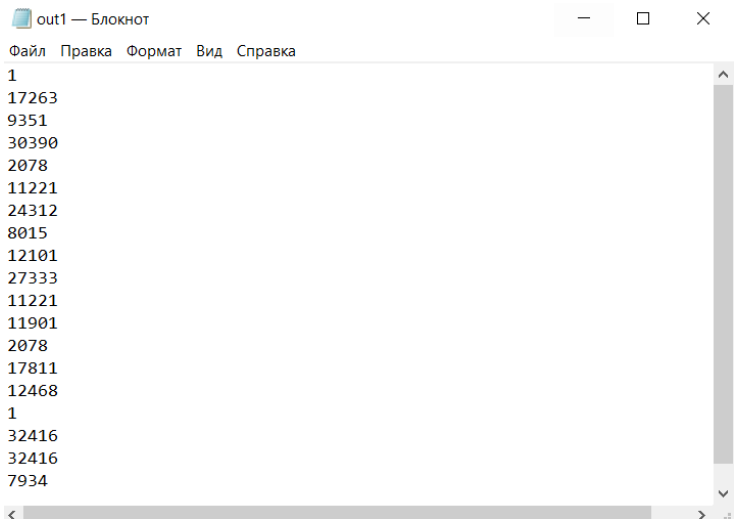

Рисунок 4.5 – Файл із результатами шифрування

5. Тепер необхідно протестувати поля вводу секретного ключа. Чи відбудеться дешифрування даних при відсутньому введеному секретному ключі.

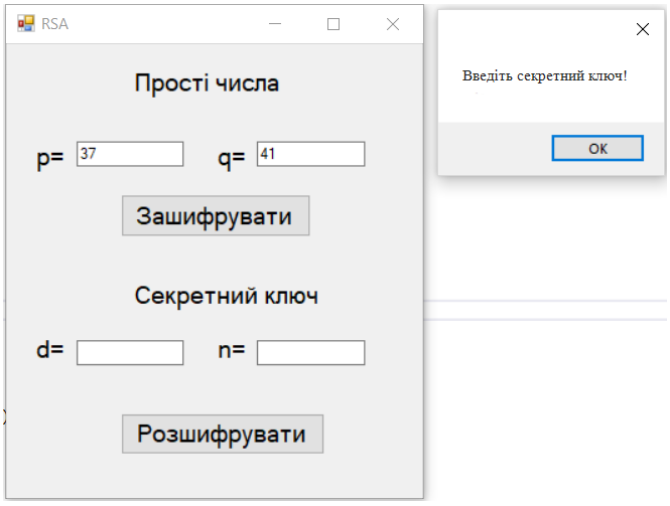

Рисунок 4.6 – Тестування полів секретного ключа

Отже з рис. 4.6 стає зрозуміло, що при відсутності параметрів секретного ключа, не відбувається обрахування за алгоритмом, потім користувачу виводиться модальне вікно з повідомленням про те, що необхідно заповнити поля секретного ключа.

6. Тестування поведінки при невірному заповненні параметрів секретного ключа змальовано на рис. 4.7.

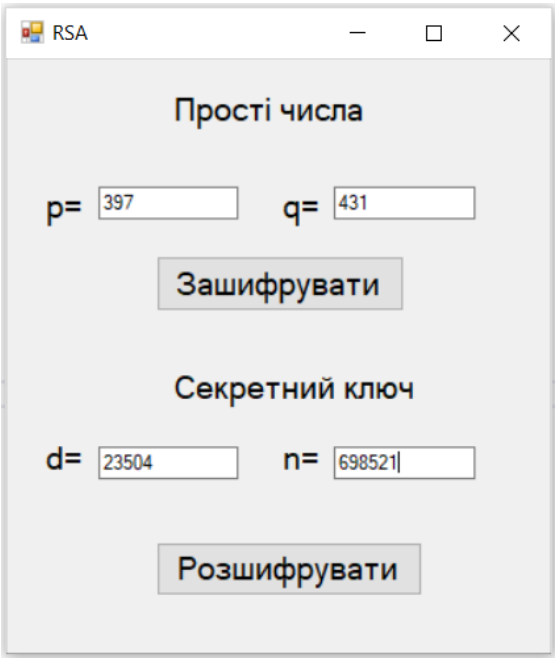

Рисунок 4.7 – Невірне заповнення параметрів секретного ключа

Це введений неправильний секретний ключ і система виводить нам повідомлення про те що ключ є невірним (рис. 4.8).

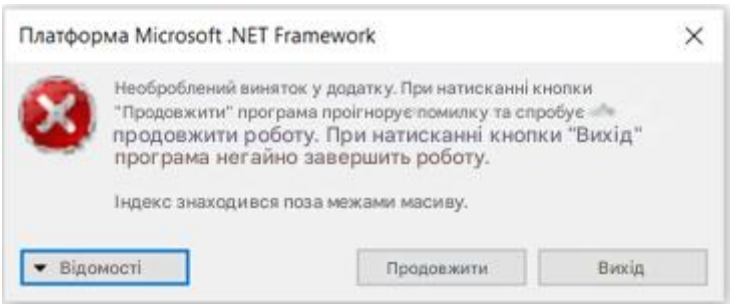

Рисунок 4.8 – Повідомлення щодо неправильних даних

7. При вірному заповненні параметрів секретного ключа, зображено на рис. 4.9, програмою виводиться текстовий документ (рис. 4.10) у якому відображено текст, який було зашифровано.

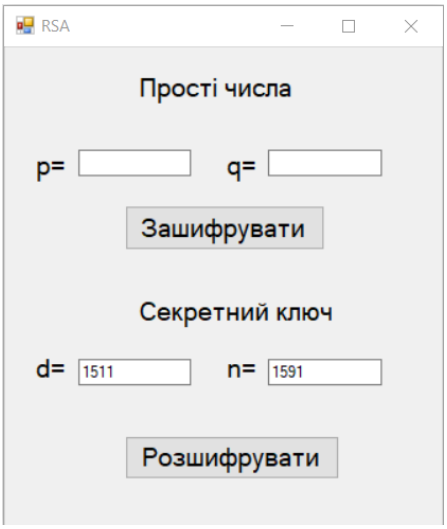

Рисунок 4.9 – Правильне введення секретного ключа

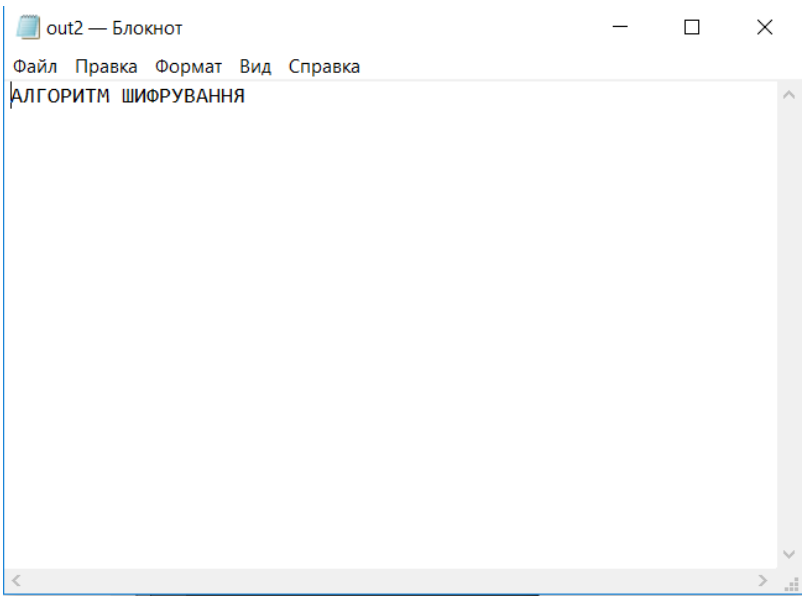

Рисунок 4.10 – Результат дешифрації даних

8. Необхідно протестувати, чи може ця програма шифрувати текст на інших мовах, наприклад англійською.

8.1. В текстовий документ користувач вводить дані на англійській мові, зображено на рис. 4.11.

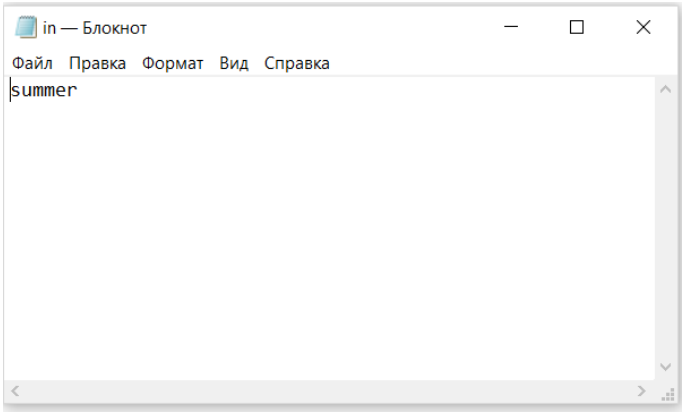

Рисунок 4.11 – Файл із вхідними даними

8.2. Далі це слово шифрується за допомогою алгоритму і результат виводиться в текстовий файл (рис 4.12).

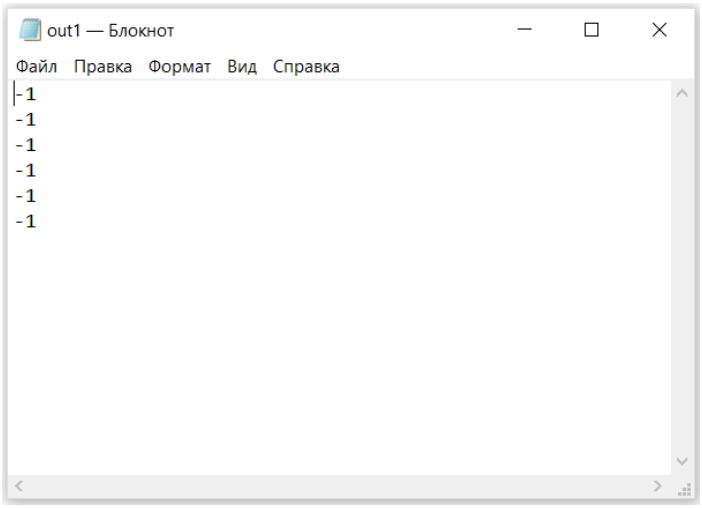

Рисунок 4.12 – Результати шифрування на англійській мові

Зрозуміло, що є проблеми в обрахуванні шифрування, якщо інформація подана саме на англійській мові.

В результаті система виводить повідомлення, що зробити дешифрування цієї інформації неможливо через не сприймання алгоритмом англійської мови.

9. Тестування безпеки

Чи безпечно використовувати даний програмний продукт для шифрування інформації? Наприклад, секретний ключ, який відображається користувачу, коли він шифрує не захищений надійно від зловмисника він може елементарно піддивитись, або завдяки цій програмі, якщо зловмисник знає

прості числа за допомогою яких певний користувач зашифрував інформацію, може вирахувати секретний ключ. Для того щоб надійно захистити секретний ключ від зловмисників краще використовувати хеш-функції.

4.3 Оцінка результатів тестування

Тестування невід'ємний етап розробки програмного продукту, без нього взагалі не може існувати якісний програмний продукт, саме на етапі тестування стає зрозумілим чи придатний цей продукт до використання. Тестування виконує такі завдання:

– Тестування дозволяє перевірити, чи правильно реалізовані усі вимоги до ПП, що розроблялось.

– Тестування також демонструє, що створене ПП працює відповідно до специфікацій і відповідає визначеним вимогам до продуктивності.

– Тестування гарантує зацікавленим сторонам, що продукт працюватиме так як передбачено і як того бажається.

– Тестування допомагає у виявленні дефектів, помилок. Та забезпечує розпізнавання критичних помилок, їх вирішення – ще до етапу розгортання програмного забезпечення.

– Тестування допомагає перевірити належну інтеграцію та взаємодію кожного компонента в системі.

– Фактично, тестування економить час розробки ПП, виявляючи проблеми та вузькі місця ще на ранніх етапах розробки.

– Як правило: дефекти, виявлені на початкових фазах, призводять до меншої вартості та використання ресурсів для корекції.

– Тестування запобігає потраплянню неякісного програмного продукту до кінцевого користувач, уникнення ризику отримати погану репутацію чи відгук компанії розробнику.

– Команда із тестування, на відміну від команди розробників дивиться під іншим кутом зору на процес розробки ПП, тим самим виступає у ролі неупередженого арбітра.

Представлений програмний продукт має достатню кількість недоліків, а саме:

– не відбувається шифрування на інших мовах окрім української;

– ймовірність для зловмисника дізнатися секретний ключ.

Проте програма наочно демонструє роботу алгоритму, за допомогою цієї програми можна роботи елементарне шифрування даних,але вона не придатна до великих обсягів інформації, та не достатньо надійно захищає інформацію. Для підвищення надійності захисту даних краще використовувати не тільки просте шифрування даних, але і хеш-функції та функції доповнення за допомогою яких шифрування стає більш надійним.

Висновки до розділу 4

Тестування заключний етап створення програмного продукту, саме на етапі тестування можуть виявитися недосконалості системи. У розділі 4 із загальним поняттям тестування, його видами та методами тестування системи.

У ході роботи над цим розділом було проведено та розроблено тестові дані стосовно реалізованого алгоритму. Було виявлено, що в цілому програмний продукт працює правильно та шифрування відбувається, проте в деяких окремих випадках було виявлено деякі порушення та недосконалості в роботі програми.

ВИСНОВКИ

У ході написання кваліфікаційної роботи було проаналізовано види даних, визначено основні проблеми та загрози захисту даних, а також визначено засоби за допомогою яких можливо здійснювати ефективний захист даних. Було обрано та розглянуто 3 різні алгоритми захисту даних та їх математичну основу, а також рівень їх криптостійкості. В процесі написання роботи було програмно реалізовано демонстраційний приклад роботи одного з розглянутих алгоритмів та проаналізовано переваги та недоліки його використання, в даній реалізації.

Отже захист даних невід'ємна частина нашого життя, це необхідно для забезпечення секретності певних даних та запобіганню витоку інформації за межі кола довірених осіб. Для захисту інформації необхідно вираховувати види загроз та ризиків, яким повинні запобігати засоби захисту інформації. Для ефективного захисту даних необхідно розуміти, вміти визначати та вираховувати, які саме загрози можуть спричинити втрату даних. Захист даних полягає не тільки в розробці алгоритму шифрування даних, а у комплексі дій направлених на захист інформації таких як: аналізування можливих загроз, дії по запобіганню цих загроз, або дії скеровані на протидію та боротьбу із загрозами.

Проаналізувавши представлені механізми захисту необхідно відзначити, що для певних цілей необхідно використовувати певні алгоритми. Симетричні алгоритми шифрування більш швидкі та мають більш стійкі ключі, проте вони не захищені від перехвату ключа та витоку даних. Асиметричні алгоритм повільніші, проте їх конструкція заснована на різних ключах сприяє їх розповсюдженню, вони підходять для передачі повідомлень у мережі. Хеш-функції завдяки своїм особливостям не можуть дешифрувати хеш у відкритий текст, в результаті можна зберігати за допомогою таких алгоритмів паролі, або важливу інформацію.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Теоретичні основи інформатики. URL: https://www.lessons-tva.info/edu/e-inf1/inf1-1-2.html. (дата звернення: 25.03.2025)

2. Критерії захисту даних. URL: http://rati.domen.uz.ua/index.php?id=temp-6 (дата звернення: 03.04.2025)

3. Загрози безпеки персональних даних. URL: https://data-sec.ru/personal-data/threats/ (дата звернення: 30.04.2025)

4. IT – законодавство. URL: http://aphd.ua/publication-180/. (дата звернення: 02.04.2025)

5. Криптографія і головні способи шифрування інформації. URL: https://proglib.io/p/methods-of-encryption. (дата звернення: 06.04.2025)

6. Переваги та недоліки хеш-таблиць. URL: https://alextoolsblog.blogspot.com/2019/12/hash-table-advantages-drawbacks.html. (дата звернення: 09.04.2025)

7. Програмний та апаратний захист інформації. URL: http://detektor.ua/prod/common/protect/. (дата звернення: 01.04.2025)

8. Що таке хешування: функції та застосування. URL: https://otziv-broker.com/obuchenie/chto-takoe-hjeshirovanie-funkcii-i-preimushhestva. (дата звернення: 08.04.2025)

9. Ілюстрація роботи RSA на прикладі. URL: http://www.michurin.net/computer-science/rsa.html. (дата звернення: 15.04.2025)

10. Класифікація криптографічних систем. URL: https://studizba.com/lectures/10-informatika-i-programmirovanie/316-lekcii-po-bezopasnosti-informacii/4240-3-klassifikaciya-kriptograficheskih-sistem.html. (дата звернення: 12.04.2025)

11. Поняття і види інформаційних загроз. Система забезпечення інформаційної безпеки. URL: https://drakkar11.com/osnovnye-vidy-ugroz-informatsionnoy-bezopasnosti/. (дата звернення: 28.03.2025)

12. Криптографічна хеш-функція. URL: https://gadgetshelp.com/internet/kriptograficheskaia-khesh-funktsiia/. (дата звернення: 19.04.2025)

13. Latest news about keccak. URL: https://keccak.team/index.html. (дата звернення: 20.04.2025)

14. Алгоритми / Хеш-функція SHA-3. URL: https://medium.com/dtechlog/%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B-sha-3-b2151ca08007. (дата звернення: 20.04.2025)

15. Загальний опис криптоалгоритма AES. URL: https://bit.nmu.org.ua/ua/student/metod/cryptology/%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F%209.pdf. (дата звернення: 12.04.2025)

16. Криптографічні хеш-функції. URL: https://bit.nmu.org.ua/ua/student/metod/cryptology/%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F17.pdf. (дата звернення: 16.04.2025)

17. Види даних в соціальних науках. URL: http://soc-research.info/principles/3.html. (дата звернення: 25.03.2025)

18. Хеш-функція SHA-3 - майбутнє блокчейна? URL: https://bits.media/khesh-funktsiya-sha-3-maybutne-blokcheyna-dlya-rynkov-kapitala/. (дата звернення: 21.04.2025)

19. SHA-3 Cryptographic Hash Algorithms. URL: https://www.movable-type.co.uk/scripts/sha3.html. (дата звернення: 20.04.2025)

20. Алгоритм шифрування AES. URL: https://www.opengsm.com/blog/algoritm-shifrovaniya-aes/. (дата звернення: 14.04.2025)

21. Введення в криптографію. URL: https://2hourscrypto.info/. (дата звернення: 15.04.2025)

22. Механізм забезпечення інформаційної безпеки. URL: http://infoprotect.net/note/mehanizmyi_informacionnoy_bezopasnosti. (дата звернення: 23.04.2025)

23. Основні відомості про C#. URL: http://progopedia.ua/language/csharp/. (дата звернення: 26.04.2025)

24. Короткий огляд C#. URL: https://docs.microsoft.com/dotnet/csharp/tour-of-csharp/. (дата звернення: 28.04.2025)

25. Чому краще обрати C#.URL: https://habr.com/post/66170/. (дата звернення: 29.04.2025)

26. Інтерфейс користувача.URL: http://elar.khnu.km.ua/jspui/bitstream/123456789/1415/2/Rozdil1.pdf. (дата звернення: 27.04.2025)

27. Керівництво для створення криптографічного додатку. URL:https://docs.microsoft.com/dotnet/standard/security/walkthrough-creating-a-cryptographic-application. (дата звернення: 29.04.2025)

28. Види тестування.URL: https://www.quality-assurance-group.com/vydy-testuvannya-ta-vidminnosti-mizh-nymy-shpargalka-z-testuvannya-chastyna-4/. (дата звернення: 12.05.2025)

29. Роль тестування в розробці ПЗ.URL: https://www.quality-assurance-group.com/shpargalka-dlya-qa-z-testuvannya-100-najposhyrenishyh-zapytan-na-intervyu/. (дата звернення: 14.05.2025)