МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп’ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Славський Андрій Михайлович

Програмний засіб шифрування даних на основі стійкого до сучасних атак криптоалгоритму / The software of data cyphering based on cryptoalgorithm resisted to modern attacks

напрям підготовки: 6.050102 - Комп’ютерна інженерія
фахове спрямування - Комп’ютерні системи та мережі
Бакалаврська робота

Виконав студент групи КСМ-42/1
А.М. Славський

Науковий керівник: к.т.н.,
Дубчак Л.О.

Тернопіль - 2018

РЕЗЮМЕ

Дипломний проект містить 63 сторінок пояснюючої записки, 13 рисунки, 7 таблиць, 2 додатки. Обсяг графічного матеріалу 2 аркуші формату А3.

Метою дипломного проекту є розроблення програмного засобу шифрування даних на основі стійкого до сучасних атак кристоалгоритму, який є правельзгідно його специфіки. Це було досегнено шляхом проектування алгоритму RSA. Отримано результати.

Досліджено сучасні методи шифрування даних. Крім того розгянуто області застосування алгоритму RSA. Проаналізувавши алгоритм роботи було вибрано середовище моделювання в якому буде розроблятися криптоалгоритм шифрування RSA. Опісля була здійснена верифікація і тестування розробленого алгоритму.

Розроблений програмний засіб шифрування даних має свої як позитивні так і негатисні сторони. До переваг можна віднести те що, алгоритм RSA є асиметричним, тобто він полягає в поширенні відкритих ключів у мережі. Це дозволяє кільком користувачам обмінюватися інформацією, по незахищеним каналів зв’язку. Користувач сам може змінювати як числа, і відкритий і закритий ключ на власний розсуд, потім він має поширити відкритий ключ у мережі. Це дає можливість збільшити криптостійкість.

Ключові слова: ШИФРУВАННЯ, RSA, КРИСТОАЛГОРИТМ.

RESUME

The diploma project contains 63 pages of explanatory note, 13 figures, 7 tables, 2 appendices. Volume of graphic material 2 sheets of A3 format.

The aim of the diploma project is to develop a software tool for data encryption on the basis of resistant to modern attacks crystal algorithm, which is the rule according to its specifics. This was achieved by designing the RSA algorithm. The results are obtained.

Modern methods of data encryption are studied. In addition, the scope of the RSA algorithm is expanded. After analyzing the algorithm, the simulation environment was chosen in which the cryptoalgorithm of RSA encryption will be developed. Subsequently, the developed algorithm was verified and tested.

The developed software means of data encryption has both positive and negative sides. The advantages include the fact that the RSA algorithm is asymmetric, ie it consists in the distribution of public keys in the network. This allows multiple users to share information over insecure channels. The user can change both the numbers, both public and private key at its discretion, then he must distribute the public key in the network. This makes it possible to increase cryptocurrency.

Keywords: ENCRYPTION, RSA, CRYSTAL ALGORITHM.

ЗМІСТ

ВСТУП

Бурхливий розвиток інформаційних технологій і використання їх у різних областях людської діяльності привело до того, що крім задач передачі, збереження й обробки інформації виникла не менш, а в ряді випадків і більш важлива задача захисту інформації. Постали такі проблеми, як незаконне використання алгоритмів, що є інтелектуальною власністю автора, несанкціоноване використання, модифікація, поширення і збут програмних продуктів. Це обумовило необхідність використання відповідних систем захисту.

На сьогоднішній день системи захисту програмного забезпечення широко поширені і знаходяться в постійному розвитку завдяки розширен- ню ринку програмних продуктів і телекомунікаційних технологій. У дипломній роботі представлено класифікацію систем захисту програмних розробок, їх види, основні алгоритми і методи, які використовуються при побудові систем захисту. Велика увага приділена аналізу сучасного стану програмних і апаратних засобів для захисту програм як вбудованих, так і навісних, а також засобам зламування існуючих захистів.

Сучасні компютерні системи для обміну інформації використовують різні канали передачі даних. Одним із широко розповсюджених каналів передачі даних є використання послідовних портів для встановлення доступу до корпоративної мережі, провайдера Інтернет та ін. Проте використання стандартних засобів передачі даних послідовними каналами не завжди забезпечують необхідний рівень захищеності. Тому розробка програмного засобу шифрування даних на основі стійкого до сучасних атак криптоалгоритму є важливою та актуальною задачею.

В якості основної мови програмування використовується мова C++.

C++ - мова програмування високого рівня з підтрискою декількох парадигм програмування: об’єктно – орієнтовної, узагальненої та процедурної. Мову використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнських програм.

В дипломній роботі розроблявся алгоритм RSA – це криптографічний алгоритм з відкритим ключем, що грунтується на обчислювальній складності задачі факторизації великих цілих чисел. RSA став першим алгоритмом такого типу, придатним і для шифрування і для цифрового підпису. Безпека алгоритму RSA заснована на труднощі вирішення задачі розкладання чисел на прості множники.

1 СУЧАСНІ АТАКИ ПОБІЧНИХ КАНАЛІВ ВИТОКУ ІНФОРМАЦІЇ

1.1 Поняття атаки на криптосистеми

Хакерська атака (кібератака) — спроба реалізаці небезпеки [1]. Тобто, це дії кібер-зловмисників (хакерів) або шкідливих програм, які спрямовані на захоплення інформаційних даних віддаленого компютера, отримання повного контролю над ресурсами компютера або на виведення системи з ладу.

Під атакою (англ. attack, англ. intrusion) на інформаційну систему розуміють дії (процеси) або послідовність зв'язаних між собою дій порушника, які приводять до реалізації загро інформаційним ресурсам, шляхом використання уразливостей цієї інформаційної системи

Типи атак на інформаційні системи:

- віддалене проникнення (remote penetration);
- локальне проникнення (local penetration);
- атака на відмову в обслуговуванні (denial of service);
- мережні сканери (network scanners);
- сканери уразливостей (vulnerability scanners);
- зламувачі паролів (password crackers);
- аналізатори протоколів (sniffers);
- спам e-mail (Mailbombing);
- перехоплення каналу зв'язку (Man-in-the-Middle).

DoS (від англ. Denial of Service - Відмова в обслуговуванні) — атака, що має своєю метою змусити сервер не відповідати на запити. Такий вид атаки не передбачає отримання деякої секретної інформації, але іноді буває підмогою в ініціалізації інших атак. Наприклад, деякі програми через помилки в своєму коді можуть викликати виняткові ситуації, і при відключенні сервісів здатні виконувати код, наданий зловмисником або атаки

лавинного типу, коли сервер не може обробити величезну кількість вхідних пакетів.

DDoS (від англ. Distributed Denial of Service - Розподілена DoS) — підтип небезпек, що проводяться не з одного компютера, а з декількох компютерів в мережі. У таких типах нападів використовується або виникнення помилок, що призводять до відмови сервісу, або спрацьовування захисту, що приводить до блокування роботи сервісу, а в результаті також до відмови в обслуговуванні. DDoS використовується там, де звичайний DoS неефективний. Для цього кілька компютерів об'єднуються, і кожен виробляє DoS-атаку на систему жертви. Разом це називається DDoS-атака.

Будь-яка атака являє собою не що інше, як спробу використовувати недосконаліть системи безпеки жертви або для отримання інформації, або для нанесення шкоди системі, тому причиною будь вдалої атаки є професіоналізм крекерів і цінність інфомації, а також недостатня компетенція адміністратора системи безпеки зокрема, недосконаліть програмного а недостатня увага до питань безпеки в компанії в цілому.

Також досить поширений вид атаки, заснований на роботі мережевої карти в режимі promiscuous mode, а також monitor mode для мереж Wi-Fi. В такому режимі всі пакети, отримані мережевою картою, пересилаються на обробку спеціальному додатку, званому сніффером. В результаті зловмисник може отримати велику кількість службової інформації: хто, звідки і куди передавав пакети, через які адреси ці пакети проходили. Найбільшою небезпекою такої атаки є отримання самої інформації, наприклад логінів і паролів співробітників, які можна використовувати для незаконного проникнення в систему під виглядом звичайного співробітника компанії. Вважається найстарішим методом атак, хоча суть його проста і примітивна: велика кількість поштових повідомлень роблять неможливими роботу з поштовими скриньками, а іноді і з цілими поштовими серверами.

Було розроблено безліч програм, і навіть користувач який не мав досвіду міг з провести атаку, вказавши e-mail жертви, текст повідомлення, і кількість необхідних повідомлень. Такі програми дозволяють

ховати реальний IP-адрес відправника, використовуючи для розсилки анонімний поштовий сервер. Цю атаку складно запобігти, так як навіть поштові фільтри провайдерів не можуть визначити реального відправника спаму.Провайдер може обмежити кількість листів від одного відправника, але адреса відправника і тема часто генеруються випадковим чином.

Вид атаки, коли зловмисник перехоплює канал зв'язку між двома системами, і отримує доступ до всієї інформації, що передається. При отриманні доступу на такому рівні зловмисник може модифікувати інформацію потрібним йому чином, щоб досягти своєї цілі. Мета такої атаки — незаконне отримання, крадіжка або фальсифікування переданої інформації, або ж отримання доступу до ресурсів мережі. Такі атаки вкрай складно відстежити, оскільки зазвичай зловмисник знаходиться всередині організації.

Сті́йкість (англ. Robustness) — це якість, що дозволяє системі витримувати зміни параметрів зовнішнього середовища, відмінні від розрахункових. Система організм, або проект може бути названо «стійким», якщо він в змозі впоратися з варіаціями (іноді непередбачуваними) в операційному середовищі з мінімальними: збитком, зміною або втратою функціональності.

Економічна стійкість у суспільних системах, на відміну від таких понять, як «стабільність», «стагнація», «спокій», має передбачати гнучке реагування на всі зовнішні й внутрішні впливи для того, щоб не запобігати новим обставинам, властивостям і відносинам, а уміло використовувати їх для постійного відновлення та самовдосконалення. «Економічна стійкість» суб'єктів господарювання є містким і багатогранним соціально-економічним понятям.

Варто розрізняти стійкість системи управління, системи показників і стійкість функціонування самого підприємства [2].

Напад може бути активним або пасивним [2].

"Активна атака" намагається змінити системні ресурси або впливати на їх роботу.

" Пасивна атака " намагається вивчити або використовувати інформацію з системи, але не впливає на системні ресурси (наприклад, прослуховування за промовчанням).

Атака може здійснюватися інсайдером або ззовні організації [2];

"Внутрішня атака" - це напад, ініційований суб'єктом всередині периметру безпеки ("інсайдер"), тобто суб'єкт, який має дозвіл на доступ до системних ресурсів, але використовує їх у спосіб, не схвалений тими, хто надав авторизацію.

"Зовнішня атака" ініціюється ззовні периметра, неавторизованим або незаконним користувачем системи ("сторонній"). В Інтернеті потенційні сторонні нападники варіюються від аматорських підручник до організованих злочинців, міжнародних терористів та ворожих урядів.

Поняття "атака" входить до декількох основних умов безпеки, як показано на наступній схемі [3].

Ресурс (як фізичний або логічний), називається активом, може мати одну або кілька вразливостей, які можуть бути експлуатованими на агента в дії загрози. У результаті конфіденційність, цілісність або доступність ресурсів може бути скомпрометована. Потенційно шкода може поширюватися на ресурси на додаток до того, що спочатку було визнано вразливим, включаючи додаткові ресурси організації та ресурси інших залучених сторін (клієнтів, постачальників).

Так звана тріада ЦРУ є основою інформаційної безпеки .

Атака є активною тоді, коли змінюються системні ресурси або вплив на їх роботу: відповідно це загрожує цілісності або доступності.

" Пасивна атака " намагається вивчити або використовувати інформацію з системи, але не впливає на системні ресурси: таким чином компрометуючи конфіденційність.

Загроза - це потенціал для порушення безпеки, який існує в разі наявності обставин, можливостей, дій або подій, які можуть порушити безпеку та завдати шкоди. Одже можна сказати, що загроза несе за собою вразливість і є небезпекою. Небезпека буває "навмисною" (тобто інтелектуальною, наприклад, окремим зломщиком або злочинною організацією) або "випадковою" (наприклад, можливістю несправності компютера або можливістю "діяння Бога", наприклад землетрус, вогонь або торнадо) [4].

Існує комплекс політик, пов'язаних з управлінням інформаційною безпекою, системами управління інформаційною безпекою (ISMS), відповідно до принципів управління ризиками, керувати контрзаходами для досягнення стратегії безпеки, встановленої відповідно до правил та правил, що застосовуються в країна [7].

Атака повинна призвести до інциденту безпеки, тобто до події безпеки, що передбачає порушення безпеки. Іншими словами, подібна до безпеки система подія, в якій політика безпеки системи не підкоряється або інакше порушується.

Загальна картина являє собою фактори ризику сценарію ризику [8].

Організація повинна вжити заходів для виявлення, класифікації та управління випадками безпеки. Першим логічним кроком є створення плану реагування на аварію та, нарешті, команду з реагування на надзвичайні ситуації в компютері.

Для виявлення атак на організаційних, процедурних та технічних рівнях можна встановити ряд контрзаходів. Прикладом цього є команда з реагування на компютерні аварійні ситуації, аудит безпеки інформаційних технологій та система виявлення вторгнень [9].

Атака, завжди здійснюється особами, які мають озлоблені наміри: атаки чорного тіла потрапляють до цієї категорії, а інші проводять випробування на проникнення в інформаційну систему організації, щоб з'ясувати, чи передбачено всі передбачені елементи контролю.

Напади можна поділяти відповідно до їх походження: відповідно, якщо воно проводиться з використанням одного або декількох компютерів: в останньому випадку називається розподіленою атакою. Ботнети використовуються для розподілених атак.

Інші класифікації залежать від використовуваних процедур або типу вразливостей: атаки можуть бути зосереджені на мережевих механізмах або хостах.

Деякі фізичні небезпеки: тобто крадіжка або пошкодження компютерів та іншого обладнання. Інші - це намагання змусити зміни в логіці, що використовується компютерами чи мережевими протоколами, для того, щоб досягти непередбаченого результату (оригінальним дизайнером), але корисним для зловмисника. Програмне забезпечення, яке використовується для логічних атак на компютерах називаються шкідливими програмами .

Існують такі нижче вказані атаки:

– пасивний;
– мережа;
– дротизація;
– зчитування порту;
– сканування в режимі очікування;
– активний;
– атака на відмову в обслуговуванні;
– підмішування;
– мережа;
– посередині людина;
– АРП отруєння;

– затоплення пін;

– смертельний пінг;

– Smurf атака;

– хост;

– переповнення буфера;

– переповнення купи;

– переповнення стеку;

– формат ряту атаки.

Вся промисловість працює, намагаючись мінімізувати ймовірність та наслідки інформаційної атаки.

Вони пропонують різні товари та послуги, спрямовані на:

– вивчити всі можливі категорії;

– публікувати книги та статті про предмет;

– виявлення вразливостей;

– оцінка ризиків;

– усунення вразливостей;

– винаходити, розробити та застосувати контрзаходи;

– створити план дій на випадок надзвичайних ситуацій , щоб бути готовим відповісти.

Багато організацій намагаються класифікувати вразливість та їх наслідки: найвідоміша база вразливостей - це загальні вразливості та експозиції .

Компютерні аварійні команди створені урядом та великою організацією, що займається інформаційними інцидентами безпеки.

У криптографії атака побічного каналу - це будь-яка атака, здійснена на основі інформації, отриманої від фізичної реалізації у вигляді криптосистеми, а не грубої сили чи теоретичної слабкості алгоритмів. Наприклад, інформація про час, споживання електроенергії, електромагнітні витоки або навіть звук

можуть служити додатковим джерелом інформації, яка може бути використана для атаки на систему.

Деякі атаки побічних каналів вимагають технічних знань про внутрішню роботу системи, на якій здійснюється криптографія, хоча інші, такі як, наприклад, диференціальний аналіз потужності, є ефективними як атаки чорної коробки. Багато потужних атак побічного каналу базуються на статистичних методах, започаткованих Паулом Кочером [1].

Спроби зламати криптосистему шляхом обману або примушення людей з законним доступом, як правило, не називаються атаками побічних каналів. На даний час підвищилась можливість атак побічного каналу в Інтернеті, навіть якщо передача між веб-переглядачем та сервером зашифрована (наприклад, через шифрування через HTTPS або Wi-Fi) - вважають дослідники з Microsoft Research та IndianaUniversity [2]. Одже можна дійти висновку, що атаки відіграють надто небезпечну роль у компютерній системі, тому що йдеться про загрозу важливих інформаційних ресурсів.

1.2 Сучасні атаки витоку інформації

Загальні класи атак побічного каналів витоку інформації включають:

– атака кешу - атака, що базується на здатності атакуючого стежити за доступом до кешу, які робить користувач в загальній фізичній системі, у віртуалізованому середовищі або у вигляді хмарного сервісу;

– часова атака - атака, заснована на вимірюванні того, скільки часу вимагають виконання різних обчислень;

– атака аналізу енергоспоживання - атака, яка використовує аналіз енергоспоживання обладнання під час різних обчислень;

– електромагнітна атака - атака на основі аналізу витоку електромагнітного випромінювання, яке можна зчитати під час передачі інформації по кабелю. Такі вимірювання можуть бути використані для виведення криптографічних ключів за допомогою методів, еквівалентних аналізу енергоспоживання, або можуть використовуватися в некриптових атаках, наприклад, атаках TEMPEST;

– криптоаналіз акустичний - атака, яка аналізує звук, вироблений під час обчислення (швидше за все аналіз сили звуку);

– аналіз диференційних помилок – атака, в якій таємниці виявляються шляхом введення помилок в обчислення;

– зберігання інформації - в якому конфіденційні дані читаються після нібито видалення;

– напад на аварійне ініціювання програмного забезпечення - на даний момент рідкісний клас побічних каналів;

– оптичний аналіз – атака, в якій секрети і конфіденційні дані можна читати візуальним записом за допомогою камери високої роздільної здатності або інших пристроїв, що мають такі можливості.

У всіх випадках основний принцип полягає в тому, що фізичні ефекти, викликані функціонуванням криптосистеми, можуть надавати корисну додаткову інформацію про секрети в системі, наприклад, криптографічний ключ, часткова інформація про стан, повні або часткові відкриті тексти і т.д. Термін "криптофтора" (таємна деградація) іноді використовується для вираження деградації матеріалу таємного ключа в результаті витоку побічних каналів.

Керування атакою побічного каналу здійснюється шляхом моніторингу критично важливих операцій безпеки, таких як, наприклад, запис Т-таблиці AES або багаторазовий доступ до модульної витримки. Атака може відновити таємний ключ залежно від доступу, зробленого (або не зробленого) жертвою, виводячи ключ шифрування. Також, на відміну від

деяких інших нападів побічного каналу, цей метод не створює помилки в поточній криптографічній операції і невидимий для жертви.

Атака за часом спостерігає переміщення даних в і з центрального процесора або пам'яті апаратних засобів під керуванням криптосистеми або алгоритму. Просто, спостерігаючи за варіаціями того, скільки часу потрібно для виконання криптографічних операцій, можна визначити весь секретний ключ. Такі атаки включають статистичний аналіз часових вимірювань та демонструвалися в мережах [6].

Атака аналізу енергоспоживання може забезпечити ще більш детальну інформацію, спостерігаючи енергоспоживання апаратного пристрою. Ці атаки грубо класифіковані в простий аналіз (SPA) та диференціальний аналіз енергоспоживання (DPA).

Коливання струму також генерують радіохвилі, що дає змогу здійснювати напади, які аналізують вимірювання електромагнітних еманацій. Ці атаки зазвичай включають подібні статистичні методи, як напади силового аналізу.

Історичні аналоги сучасних атак побічних каналів давно відомі. Нещодавно розсекречений документ NSA показує, що ще в 1943 році інженер телефонії спостерігав розшифровані шипи на осцилограмі, пов'язані з розшифрованим виходом певного шифрувального телетайпа [3]. За даними колишнього офіцера Питера Райта, британська служба безпеки аналізувала викиди від французького шифрувального обладнання в 1960-х. У 1980-х роках радянські підслухувачі підозрювалися в виявленні помилок всередині друкарських машинок IBM Selectric для відстеження електричного шуму, який генерується головкою, що обертається та розташується, щоб пробити папір; характеристики цих сигналів могли б визначити, який ключ був натиснутий.

Споживана потужність пристроїв викликає нагрівання, що компенсується ефектами охолодження. Температурні зміни створюють

термічно індуковані механічні напруги. Цей стрес може спричинити викиди акустичних (тобто шумових) шумів від діючих центральних процесорів (у деяких випадках - близько 10 кГц). Недавні дослідження Шаміра та співавторів запропонував також отримати інформацію про роботу криптосистем та алгоритмів. Це акустична атака; якщо поверхню мікропроцесорного процесора, або в деяких випадках пакет CPU, можна спостерігати, інфрачервоні зображення також можуть надавати інформацію про код, що виконується на ЦП, відомий як атака термічного зображення.

Приклади оптичних прикладів побічного каналу включають:

– крадіжка даних з компютера;

– ексфільтрація таємного ключа за допомогою пікосекундного аналізу схем зображень.

Оскільки атаки побічних каналів залежать від взаємозв'язків між виділеною (витіковою) інформацією через побічний канал і таємними даними, контрзаходи потрапляють у дві основні категорії:

– усунення або зменшення витоку такої інформації;

– усунення зв'язку між витоком інформації та таємними дані, тобто

– зробити витік інформації незв'язаною або скоріше некорельованою з секретними

– даними, як правило, через деяку форму рандомізації шифрувального тексту, який перетворює дані таким чином, який можна відновити після криптографічної операції (наприклад, дешифрування).

За першою категорією, дисплеї зі спеціальним захистом для зменшення електромагнітних випромінювань, зменшуючи сприйнятливість до атак TEMPEST, тепер є комерційно доступними. Підготовка та фільтрація лінії електропередач можуть допомогти запобігти нападам на моніторинг енергоспоживання, хоча такі заходи повинні застосовуватися обережно, оскільки навіть дуже малі співвідношення можуть залишатися та створювати компромісну безпеку. Фізичні корпуси можуть зменшити ризик прихованого

встановлення мікрофонів (для протидії акустичним нападам) та інших пристроїв мікро-моніторингу (проти силових атак або термічних нападів центрального процесора).

Інший контрзахід (як і раніше в першій категорії) полягає в тому, щоб забити канал шумом. Наприклад, випадкові затримки можуть бути додані для запобігання тимчасових атак, хоча супротивники можуть компенсувати ці затримки шляхом усереднення кількох вимірювань (або, більш загальним чином, використовуючи більше вимірювань в аналізі). Оскільки кількість шумів у побічному каналі збільшується, противник повинен збирати більше вимірювань.

У випадку часової атаки, обчислення яких квантуються в дискретний цикл циклів, ефективним контрзаходом проти полягає в тому, щоб спроектувати програмне забезпечення бути ізохронним, тобто працювати в рівно постійній кількості часу, незалежно від секретних значень. Це робить атаки часу

неможливими. Такі контрзаходи можуть бути важко реалізовувані на практиці, оскільки навіть окремі вказівки можуть мати змінний час для деяких центральних процесорів.

Однією частковою протидією проти простих атак аналізу помилок, але не диференційних атак на аналіз силової спроможності, є розробка програмного забезпечення таким чином, щоб воно було "безпечним для ПК" у "моделі безпеки контенту програми". У захищеній компютером програмі шлях виконання не залежить від секретних значень. Іншими словами, всі умовні гілки залежать тільки від публічної інформації. Це більш обмежувальна умова, ніж ізохронний код, але менш обмежувальний стан, ніж розгалужений код. Хоча операції множення призводять до більшої потужності, ніж NOR, практично у всіх процесорах, використання постійного шляху виконання запобігає подібним різницям потужності від витоку будь-якої секретної інформації. У архітектурах, де час виконання інструкцій не

залежить від даних, програма, захищена від ПК, також захищена від часових атак [4, 5].

Інший спосіб, у який код може бути неізохронним, полягає в тому, що сучасні процесори мають кеш пам'ять: доступ до нечасто використовуваної інформації призводить до великої затримки, показуючи деяку інформацію про частоту використання блоків пам'яті. Криптографічний код, призначений для протидії атаці кешу, намагається використовувати пам'ять лише передбачуваним способом (наприклад, доступ до тільки вхідних, вихідних та програмних даних, і це робиться відповідно до фіксованого шаблону). Наприклад, слід уникати таблиць пошуку, залежних від даних, тому що кеш може виявити, яка частина пошукової таблиці була доступна.

Інші часткові контрзаходи намагаються зменшити обсяг інформації, що вийшов із кешу залежно від даних відмінності потужності. Деякі операції використовують інформацію, корелюючу з кількістю 1 біт у секретному значенні. Використання коду константної ваги (наприклад, використання воріт Фредкіна або кодування з подвійною рельєфною лінією) може зменшити виток інформації про вагу Хеммінга (кількість 1 в бітовій послідовності) таємного

значення, хоча експлуатаційні кореляції, ймовірно, залишиться без змін. Цей "збалансований дизайн" можна апроксимувати в програмному забезпеченні, маніпулюючи разом даними та їхніми доповненнями [6].

Кілька "захищених процесорів" були побудовані як асинхронні процесори вони не мають глобальної довідки про час. Хоча ці процесори призначалися для того, щоб зробити часові та силові атаки, подальші дослідження виявили, що часові варіації в асинхронних схемах важче видалити.

Типовим прикладом другої категорії є техніка, відома як сліпота. У випадку розшифровки RSA з секретним показником і відповідним показником шифрування і модулем, техніка застосовується наступним чином

(для простоти, модульне скорочення на m опущено в формулах): перед розшифровкою, тобто перед обчисленням результату для даного шифрувального тексту, система вибирає випадкове число і шифрує його за допомогою публічного показника. Потім виконується дешифрування оскільки система дешифрування може обчислити його зворотний модуль, скасувати фактор виводу результату і отримати фактичний результат розшифровки. Для атак, які вимагають збору інформації побічного каналу від операцій з даними, контрольованими зловмисником, такий підхід є ефективним контрзаходом, оскільки фактична операція виконується за рандомізованою версією даних, над якою атакуючий не має контролю або навіть знань.

Інша більш загальний контрзахід - це маскування. Принцип маскування полягає у здатності уникати маніпулювання будь-яким чутливим значеннямбезпосередньо, а скоріше маніпулювати поділом на нього. Зловмисник повинен відновити всі значення акцій, щоб отримати будь-яку важливу інформацію.

Сюди додати ще про внесення додаткових операцій в код чи алгоритм як метод захисту від атак побічних каналів витоку інформації

1.3 Постановка задачі та аналіз дерева рішень

В даному дипломному проекті поставлена задача розробити програмний засіб шифрування даних на основі стійкого до сучасних атак криптоалгоритму.

Процес проектування складається з етапів, зображених на рисунку 1.1.

Сучасні компютерні системи для обміну інформації використовують різні канали передачі даних. Одним із широко розповсюджених каналів

передачі даних є використання послідовних портів для встановлення доступу до корпоративної мережі, провайдера Інтернет та ін.

Рисунок 1.1 – Процес дипломного проектування

Проте використання стандартних засобів передачі даних послідовними каналами не завжди забезпечують необхідний рівень захищеності. Тому розробка програмного засобу шифрування даних на основі стійкого до сучасних атак криптоалгоритму є важливою та актуальною задачею.

2 СУЧАСНІ СИМЕТРИЧНІ І АСИМЕТРИЧНІ КРИПТОСИСТЕМИ

2.1 Сучасний стан криптографії

Шифрування - це спосіб зміни повідомлення або іншого документа, що забезпечує спотворення (приховування) його вмісту. (Кодування - це перетворення звичайного, зрозумілого, тексту в код. При цьому мається на увазі, що існує взаємно однозначна відповідність між символами тексту (даних, чисел, слів) і символьного коду - в цьому принципова відмінність кодування від шифрування. Часто кодування і шифрування вважають одним і тим же, забуваючи про те, що для відновлення закодованого повідомлення, достатньо знати правило підстановки (заміни). Для відновлення ж зашифрованого повідомлення крім знання правил шифрування, потрібно і ключ до шифру. Ключ розуміється нами як конкретне секретне стан параметрів алгоритмів шифрування і дешифрування. Знання ключа дає можливість прочитання секретного повідомлення. Втім, як видно нижче, далеко не завжди незнання ключа гарантує те, що повідомлення не зможе прочитати стороння людина.). Шифрувати можна не тільки текст, але і різні компютерні файли - від файлів баз даних і текстових процесорів до файлів зображень.

Шифрування використовується людством з того самого моменту, як з'явилася перша секретна інформація, тобто така, доступ до якої повинен бути обмежений.

Ідея шифрування полягає в запобіганні перегляду істинного змісту повідомлення (тексту, файлу і т.п.) тими, у кого немає коштів його дешифрування. А прочитати файл зможе лише той, хто зможе його дешифрувати.

Шифрування з'явилося приблизно чотири тисячі років тому. Першим

відомим застосуванням шифру (коду) вважається єгипетський текст, датований приблизно 1900 р. до н. е.., автор якого використав замість звичайних (для єгиптян) ієрогліфів не збігаються з ними знаки.

Один з найбільш відомих методів шифрування носить ім'я Цезаря, який якщо і не сам його винайшов, то активно ним користувався. Не довіряючи своїм посильним, він шифрував листи елементарної заміною A на D, B на E і так далі по всьому латинським алфавітом. При такому кодуванні комбінація XYZ була б записана як ABC, а слово «ключ» перетворилося б на неоковирне «ноб'» (прямий код N +3).

Через 500 років шифрування стало повсюдно використовуватися при залишенні текстів релігійного змісту, молитов і важливих державних документів.

Із середніх віків і до наших днів необхідність шифрування військових, дипломатичних і державних документів стимулювало розвиток криптографії. Сьогодні потреба в коштах, які забезпечують безпеку обміну інформацією, багаторазовозросла.

Більшість з нас постійно використовують шифрування, хоча і не завжди знають про це. Якщо у вас встановлена операційна система Microsoft, то знайте, що Windows зберігає про вас (як мінімум) таку секретну інформацію:

– паролі для доступу до мережевих ресурсів (домен, принтер, компютери в мережі і т.п.);

– паролі для доступу в Інтернет за допомогою DialUp;

– кеш паролів (в браузері є така функція - кешувати паролі, і Windows зберігає всі коли-небудь вводяться вами в Інтернеті паролі);

– сертифікати для доступу до мережевих ресурсів і зашифрованих даних на самому компютері.

Ці дані зберігаються або в pwl-файлі (в Windows 95), або в SAM-файлі (в Windows NT/2000/XP). Це файл Реєстру Windows, і тому операційна

система нікому не дасть до нього доступу навіть на читання. Зловмисник може скопіювати такі файли, тільки завантажившись в іншу ОС або з дискети. Утиліт для їх злому досить багато, найсучасніші з них здатні підібрати ключ за кілька годин.

2.2 Основні поняття та визначення криптографії

Отже, криптографія дає можливість перетворити інформацію таким чином, що її прочитання (відновлення) можливе тільки при знанні ключа.

Перерахую спочатку деякі основні понятя і визначення.

Алфавіт - кінцеве безліч використовуваних для кодування інформації знаків.

Текст - упорядкований набір з елементів алфавіту.

В якості прикладів алфавітів, які в сучасних ІС можна навести такі:

– алфавіт Z 33 - 32 літери російського алфавіту і пробіл;

– алфавіт Z 256 - символи, що входять стандартні коди ASCII і КОІ-8;

– бінарний алфавіт - $Z\,2 = \{0,1\}$;

– восьмеричний алфавіт або шістнадцятковий алфавіт;

Шифрування - перетворювальний процес: вихідний текст, що має також назву відкритого тексту, замінюється шифрованим текстом.

Дешифрування - зворотний шифруванню процес. На основі ключа шифрований текст перетвориться у вихідний.

Ключ - інформація, необхідна для безперешкодного шифрування й дешифрування текстів.

Криптографічна система являє собою сімейство T перетворень відкритого тексту. члени цього сімейства індексуються, чи позначаються

символом k; параметр k є ключем. Простір ключів K - це набір можливих значень ключа. Зазвичай ключ являє собою послідовний ряд букв алфавіту.

Криптосистеми поділяються на симетричні й з відкритим ключем

У симетричних криптосистемах і для шифрування, і для дешифрування використовується один і той самий ключ.

У системах з відкритим ключем використовуються два ключі - відкритий і закритий, які математично пов'язані один з одним. Інформація шифрується за допомогою відкритого ключа, що доступний усім бажаючим, а розшифровується за допомогою закритого ключа, відомого тільки одержувачу повідомлення. Терміни розподіл ключів і керування ключами відносяться до процесів системи обробки інформації, змістом яких є складання і розподіл ключів між користувачами.

Електронній (цифровій) підписом називається приєднане до тексту його криптографічне перетворення, яке дозволяє при отриманні тексту іншим користувачем перевірити авторство і достовірність повідомлення.

Криптостійкість називається характеристика шифру, що визначає його стійкість до дешифрування без знання ключа (тобто криптоаналізу). Є декілька показників криптостійкості, серед яких:

– кількість всіх можливих ключів;

– середній час, необхідне для криптоаналізу.

Перетворення $T(k)$ визначається відповідним алгоритмом і значенням параметра k. Ефективність шифрування з метою захисту інформації залежить від збереження таємниці ключа і криптостійкості шифру.

Процес криптографічного закриття даних може здійснюватися як програмно, так і апаратно. Апаратна реалізація відрізняється істотно більшою вартістю, проте їй притаманні і переваги: висока продуктивність, простота, захищеність і т.д. Програмна реалізація більш практична, допускає відому гнучкість у використанні.

Для сучасних криптографічних систем захисту інформації

сформульовані наступні загальноприйняті вимоги:

– зашифроване повідомлення повинно піддаватися читанню тільки при наявності ключа;

– число операцій, необхідних для визначення використаного ключа шифрування за фрагментом шифрованого повідомлення і відповідного йому відкритого тексту,число операцій, необхідних для розшифровки інформації шляхом

– перебору різноманітних ключів повинно мати строгу нижню оцінку і виходити за межі можливостей сучасних компютерів (з урахуванням можливості використання мережевих обчислень);

– знання алгоритму шифрування не повинно впливати на надійність захисту;

– незначна зміна ключа повинна приводити до істотної зміни виду зашифрованого повідомлення навіть при використанні одного і того ж ключа;

– структурні елементи алгоритму шифрування повинні бути незмінними;

– додаткові біти, що вводяться в повідомлення в процесі шифрування, повинен бути повністю та надійно сховані в зашифрованому тексті;

– довжина шифрованого тексту повинна бути рівною довжині вихідного тексту;

– не повинно бути простих і легко встановлюваних залежністю між ключами, послідовно використовуються в процесі шифрування;

– будь-який ключ із безлічі можливих повинен забезпечувати надійний захист інформації;

– алгоритм повинен допускати як програмну, так і апаратну реалізацію, при цьому зміна довжини ключа не повинна призводити до якісного погіршення алгоритму шифрування.

2.3 Сучасні симетричні і асиметричні криптосистеми

Перш ніж перейти до окремих алгоритмам, розглянемо коротко концепцію симетричних і асиметричних криптосистем. Згенерувати секретний ключ і зашифрувати їм повідомлення - це ще півсправи. А ось як переслати такий ключ того, хто повинен з його допомогою розшифрувати вихідне повідомлення? Передача шифрувального ключа вважається однією з основних проблем криптографії.

Залишаючись у рамках симетричної системи, необхідно мати надійний канал зв'язку для передачі особистого ключа. Але такий канал не завжди буває доступний, і тому американські математики Діффі, Хеллман і Меркле розробили в 1976 р. концепцію відкритого ключа та асиметричного шифрування.

У таких криптосистемах загальнодоступним є лише ключ для процесу шифрування, а процедура дешифрування відома лише володарю секретного ключа. Наприклад, коли я хочу, щоб мені вислали повідомлення, то генеруючи відкритий і секретний ключі. Відкритий посилаю вам, ви шифруєте їм повідомлення і відправляєте мені. Дешифрувати повідомлення можу тільки я, так як секретний ключ я нікому не передавав. Звичайно, обидва ключі пов'язані особливим чином (у кожній криптосистеме по-різному), і поширення відкритого ключа не руйнує крипостійкість системи.

В асиметричних системах має задовольнятися наступну вимогу: немає такого алгоритму (або він поки невідомий), який би з криптотекст і відкритого ключа виводив вихідний текст.

Серед найрізноманітніших способів шифрування можна виділити наступні основні методи.

Алгоритми заміни або підстановки - символи вихідного тексту замінюються на символи іншого (або того ж) алфавіту відповідно до

заздалегідь визначеною схемою, яка і буде ключем даного шифру. Окремо цей метод в сучасних криптосистемах практично не використовується через надзвичайно низьку криптостійкості.

Алгоритми перестановки - символи оригінального тексту міняються місцями за певним принципом, що є секретним ключем. Алгоритм перестановки сам по собі має низьку криптостійкість, але входить у ролі елемента у дуже багато сучасних криптосистеми.

Алгоритми гамування - символи вихідного тексту складаються з символами якоїсь випадкової послідовності. Найпоширенішим прикладом вважається шифрування файлів «ім'я пользователя.pwl», в яких операційна система Microsoft Windows 95 зберігає паролі до мережевих ресурсів даного користувача (паролі на вхід в NT-сервери, паролі для DialUp-доступу в Інтернет і т.д.). Коли користувач вводить свій пароль при вході в Windows 95, з нього за алгоритмом шифрування RC4 генерується гамма (завжди одна й та сама), застосовувана для шифрування мережевих паролів. Простота підбору пароля обумовлюється в даному випадку тим, що Windows завжди віддає перевагу одну і ту ж гаму.

Алгоритми, засновані на складних математичних перетвореннях вихідного тексту за деякою формулою. Багато з них використовують невирішені математичні завдання. Наприклад, широко використовуваний в Інтернеті алгоритм шифрування RSA заснований на властивостях простих чисел.

Комбіновані методи. Послідовне шифрування вихідного тексту за допомогою двох і більше методів.

Порівняння з асиметричними криптосистемами.

В основному, симетричні алгоритми шифрування вимагають менше обчислень, ніж асиметричні. На практиці, це означає, що якісні асиметричні алгоритми в сотні або в тисячі разів повільніші за якісні симетричні алгоритми. Недоліком симетричних алгоритмів є необхідність мати

секретний ключ з обох боків передачі інформації. Так як ключі є предметом можливого перехоплення, їх необхідно часто змінювати та передавати по безпечних каналах передачі інформації під час розповсюдження.

Переваги:

– швидкість (за даними AppliedCryptography - на 3 порядки вище);
– простота реалізації (за рахунок більш простих операцій);
– менша необхідна довжина ключа для порівнянної стійкості;
– вивченість (за рахунок більшого віку).

Недоліки:

– складність управління ключами у великій мережі. Це означає квадратичне зростання числа пар ключів, які треба генерувати, передавати, зберігати і знищувати в мережі. Для мережі в 10 абонентів потрібно 45 ключів, для 100 вже 4950, для 1000 - 499500 і т. д.

– складність обміну ключами. Для застосування необхідно вирішити проблему надійної передачі ключів кожному абоненту, тому що потрібен секретний канал для передачі кожного ключа обом сторонам.

Для компенсації недоліків симетричного шифрування в даний час широко застосовується комбінована (гібридна) криптографічний схема, де за допомогою асиметричного шифрування передається сеансовий ключ, що використовується сторонами для обміну даними за допомогою симетричного шифрування.

Важливою властивістю симетричних шифрів є неможливість їх використання для підтвердження авторства, так як ключ відомий кожній стороні.

RSA— криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел.

RSA став першим алгоритмом такого типу, придатним і для шифрування, і для цифрового підпису. Алгоритм застосовується до великої кількості криптографічних застосунків.

Опис RSA було опубліковано у 1977 році Рональдом Райвестом, Аді Шаміром і Леонардом Адлеманом з Массачусетського технологічного інституту (MIT).

Британський математик Кліфорд Кокс (англ. Clifford Cocks), що працював в центрі урядового зв'язку (GCHQ) Великої Британії, описав аналогічну систему в 1973 році у внутрішніх документах центру, але ця робота не була розкрита до 1997 року, тож Райвест, Шамір і Адлеман розробили RSA незалежно від роботи Кокса.

В 1983 році був виданий патент 4405829 США, термін дії якого минув 21 вересня 2000 року.

Алгоритм RSA складається з 4 етапів: генерації кодів, шифрування, розшифрування та розповсюдження ключів.

Безпека алгоритму RSA побудована на принципі складності факторизації цілих чисел. Алгоритм використовує два ключі — відкритий (public) і секретний (private), разом відкритий і відповідний йому секретний ключі утворюють пари
ключів (keypair). Відкритий код не потрібно зберігати в таємниці, він використовується для шифрування даних. Якщо повідомлення було зашифровано відкритим ключем, то розшифрувати його можна тільки відповідним секретним ключем.

Найбільш ефективна атака: знайти секретний (private) матеріал, відповідний необхідному відкритому (public) ключу. Це дозволить нападнику читати всі повідомлення, зашифровані відкритим (public) матеріалом і підробляти підписи. Таку атаку можна провести, знайшовши головні співмножники (фактори) загального модуля $n - p$ і q. На підставі p, q і e (загальний показник), нападник може легко обчислити приватний показник d. Основна складність — пошук головних співмножників (факторинг) n; безпека RSA залежить від розкладання на множники (факторингу), що є складнорозв'язним завданням, тобто не має ефективних способів вирішення.

Фактично, завдання відновлення секретного (private) ключа еквівалентне задачі розкладання на множники (факторингу) модуля: можна використовувати *d* для пошуку співмножників *n*, і навпаки, можна використовувати *n* для пошуку *d*. Треба зазначити, що удосконалення обчислювального обладнання само по собі не зменшить стійкість криптосистеми RSA, якщо ключі будуть мати достатню довжину. Фактично ж вдосконалення устаткування збільшує стійкість криптосистеми.

Інший спосіб зламати RSA полягає в тому, щоб знайти метод обчислення кореня ступеня з *e* (mod *n*). Оскільки *3 = M^e**(mod *n*), то коренем ступеня *e* (mod *n*) є повідомлення *M*. Обчисливши корінь, можна розкрити зашифровані повідомлення і підробляти підписи, навіть не знаючи приватний (private) ключ. Така атака не еквівалентна факторингу, але в даний час є невідомі методи, які дозволяють зламати RSA таким чином. Проте, в особливих випадках, коли на основі одного і того ж показника відносно невеликої величини шифрується досить багато пов'язаних повідомлень, є можливість розкрити повідомлення. Згадані атаки — єдині способи розшифрувати всі повідомлення, зашифровані ключем RSA.

Існують і інші типи атак, що дозволяють, однак, розкрити тільки одне повідомлення і не дозволяють нападнику розкрити інші повідомлення, зашифровані тим же ключем.

Найпростіший напад на єдине повідомлення — атака по передбачуваному відкритому тексту. Нападник, маючи зашифрований текст, передбачає, що повідомлення містить якийсь певний текст, наприклад, «Напад на світанку», потім шифрує передбачуваний текст відкритими (public) ключем одержувача і порівнює отриманий текст з наявним зашифрованим текстом. Таку атаку можна запобігти, додавши в кінці повідомлення кілька випадкових бітів.

Інша атака єдиного повідомлення застосовується в тому разі, якщо хтось посилає одне і те ж повідомлення *M* трьом кореспондентам, кожен з

яких використовує загальний показник $e = 3$. Знаючи це, форвард може перехопити ці повідомлення і розшифрувати повідомлення M. Таку атаку можна запобігти вводячи в повідомлення перед кожним шифруванням кілька випадкових біт.

Також існують декілька атак за зашифрованим текстом (або атаки окремих повідомлень з метою підробки підпису), при яких нападник створює деякий зашифрований текст і отримує відповідний відкритий текст, наприклад, змушуючи обманним шляхом зареєстрованого користувача розшифрувати підроблене повідомлення.

Зрозуміло, існують і атаки націлені не на криптосистему безпосередньо, а на вразливі місця всієї системи комунікацій в цілому; такі атаки не можуть розглядатися як злом RSA, тому що говорять не про слабкість алгоритму RSA, а швидше про уразливості його конкретної реалізації. Наприклад, нападник може заволодіти секретним (private) ключем, якщо той зберігається без належних обережностей. Необхідно підкреслити, що для повного захисту недостатньо захистити виконання алгоритму RSA і вжити заходів обчислювальної безпеки, тобто використовувати ключ достатньої довжини. На практиці ж найбільший успіх мають атаки на незахищені етапи управління ключами системи RSA.

Як при шифруванні і розшифруванні, так і при створенні і перевірці підпису, алгоритм RSA у своїй сутності складається з піднесення в степінь, яке виконується як ряд множень.

У практичних додатках для відкритого ключа зазвичай обирається відносно невеликий показник, а часто групи користувачів використовують один і той же відкритий показник, але кожен з різним модулем. Якщо відкритий показник незмінний, вводяться деякі обмеження на головні співмножники (фактори) модуля. При цьому шифрування даних йде швидше, ніж розшифровка, а перевірка підпису — швидше, ніж підписання.

Якщо k — кількість бітів у модулі, то зазвичай в алгоритмах, що

використовуються для RSA, кількість кроків, необхідна для виконання операції з відкритим ключем, пропорційна другій степені *k*, кількість кроків для операцій секретного ключа — третій степені *k*, кількість кроків для операції створення ключів — четвертій степені *k*.

Методи «швидкого множення» — наприклад, методи основані на швидкому перетворенні Фур'є — виконуються меншою кількістю кроків. Проте вони не набули широкого поширення через складність програмного забезпечення, а також тому, що з типовими розмірами ключів вони фактично працюють повільніше. Однак продуктивність та ефективність програм і обладнання, які реалізовують алгоритм RSA, швидко збільшується.

Алгоритм RSA набагато повільніший, ніж DES та інші алгоритми блокового шифрування. Програмна реалізація DES працює швидше принаймні в 100 разів і від 1,000 до 10,000 — в апаратній реалізації (в залежності від конкретного пристрою). Завдяки проведенню розробок, швидкість алгоритму RSA, ймовірно, прискориться, але одночасно прискориться і робота алгоритмів блокового шифрування.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ЗАХИСТУ ІНФОРМАЦІЇ НА ОСНОВІ АЛГОРИТМУ RSA, СТІЙКОГО ДО АТАКИ АНАЛІЗУ ЕНЕРГОСПОЖИВАННЯ

3.1 Функціональна структура програмного засобу

Асиметричні алгоритми, вирішивши завдання передачі ключів, все частіше починають використовуватися у засобах шифрування даних. Але разом із своїми перевагами вони мають ряд недоліків, на розв'язання яких спрямовані наукові дослідження провідних компаній сфери інформаційних технологій. Навіть якщо проблеми розв'язати повністю не вдасться, то і мінімальне покращення використання асиметричних алгоритмів може надати їм значних переваг.

Сучасні атаки на системи захисту інформації є дуже простими у виконанні та вимагають серйозних засобів захисту від них. Крім того, дуже важливо не забувати про продуктивність системи. Тому дуже важливо, щоб розроблювана програма захисту інформації на основі алгоритму RSA була стійкою до атаки аналізу енергоспоживання та з достатнім рівнем швидкодії.

Отже, реалізовано програму, яка виконує процедуру шифрування даних, використовуючи асиметричний алгоритм шифрування RSA з використанням додаткових операцій. Основне завдання реалізованої програми – це використовуючи запропонований метод перетворення інформації зробити її зміст недоступним для читання і тим самим забезпечити конфіденційність інформації та її цілісність.

Запропонована система захисту інформації призначена для використання в сфері документообігу, обміну інформації, та в навчальних цілях з метою забезпечення конфіденційності та повного розмежування доступу до інформації, яка може становити комерційну таємницю.

Функціонально система захисту інформації забезпечує захист інформації за допомогою алгоритмів криптування та розкриптовування, гарантує її цілісність у системі обміну інформацією.

Мета розробки полягає у тому, щоб за допомогою програмного засобу була можливість досліджувати систему та впровадити її у використання.

Програма генерує пару ключів – відкритий і закритий, шифрує вибране користувачем повідомлення.

Розробляти програму захисту інформації на основі алгоритму RSA, стійкому до аналізу енергоспоживання, доцільно розробити на мові програмування С++.

Мова С++ виникла на початку 1980-х років. Ранні версії мови, відомі під іменем «С із класами», почали з'являтися з 1980 року. Ідея створення нової мови бере початок від досвіду програмування Страуструпа. Він виявив, що мова моделювання Сімула (Simula) має такі можливості, які були б дуже корисні для розробки великого програмного забезпечення, але працює занадто повільно. У той же час мова BCPL досить швидка, але занадто близька до мов низького рівня й не підходить для розробки великого програмного забезпечення. Страуструп почав працювати в Bell Labs над завданнями теорії черг (у додатку до моделювання телефонних викликів). Спроби застосування існуючих у той час мов моделювання виявилися неефективними. Згадуючи досвід своєї дисертації, Страуструп вирішив доповнити мову С (спадкоємець BCPL) можливостями, наявними в мові Смула. Мова С, будучи базовою мовою системи UNIX, на якій працювали компютери Bell, є швидкою, багатофункціональною і переносимою. Страуструп додав до неї можливість роботи із класами й об'єктами. У результаті, практичні завдання моделювання виявилися доступними для розв'язку як з боку часу розробки (завдяки використанню Смула-подібних класів) так і з боку часу обчислень (завдяки швидкодії С). На початку в С були додані класи (з інкапсуляцією), похідні класи, жорстка перевірка типів,

inline-функції й аргументи за замовчуванням.

Розробляючи С із класами (пізніше С++), Страуструп також написав програму cfront – транслятор, що переробляв вихідний код С із класами у вихідний код простого С. Нова мова, зненацька для автора, набула високої популярності серед колег і незабаром Страуструп уже не міг особисто підтримувати її, відповідаючи на тисячі питань.

В 1983 році відбулося перейменування мови із С «із класами» в С++. Крім того, у неї були додані нові можливості, такі як віртуальні функції, перевантаження функцій і операторів, посилання, константи, користувацький контроль над керуванням вільною пам'яттю, покращена перевірка типів і новий стиль коментарів (//). Її перший комерційний випуск відбувся в жовтні 1985 року. В 1985 році вийшло також перше видання « Мови програмування С++» перший опис цієї мови, що було надзвичайно важливо через відсутність офіційного стандарту. В 1989 році відбувся вихід С++ версії 2.0. Її нові можливості включали множинне спадкування, абстрактні класи, статичні функції-члени, функції-константи й захищені члени.

В 1990 році вийшло «Коментований довідковий посібник з С++», покладений згодом в основу стандарту. Останні оновлення включали шаблони, виключення, простори імен, нові способи приведення типів і булевий тип.

Стандартна бібліотека С++ також розвивалася разом з мовою. Першим додатком до стандартної бібліотеки С++ стали потоки введення/виводу, що забезпечують засоби для заміни традиційних функцій С printf і scanf. Пізніше самим значним розвитком стандартної бібліотеки стало включення в неї стандартної бібліотеки шаблонів.

Після багатьох років роботи спільний комітет ANSI-ISO стандартизував С++ в 1998 році (ISO/IEC 19982:1998 – Мова програмування С++). Протягом декількох років після офіційного виходу стандарту комітет обробляв повідомлення про помилки й у підсумку випустив виправлену

версію стандарту C++ в 2003 році. У цей час робоча група ISO працює над новою версією стандарту під кодовою назвою C++0x (раніше відомою як C++09).

Ніхто не має права на мову C++, вона є вільною. Однак сам документ стандарту мови (за винятком чернеток) не доступний безкоштовно.

Назва «C++» була придумана Ріком Маскітті (Rick Mascitti) і вперше було використана в грудні 1983 року. Ім'я, що вийшло в підсумку, походить від оператора C «++» (збільшення значення змінної на одиницю). Ім'я «C+» не було використано тому, що є синтаксичною помилкою в C й, крім того, це ім'я було зайнято іншою мовою. Мова також не названа «D», оскільки є розширенням C й не намагається усувати проблеми шляхом видалення елементів C.

Стандарт C++ складається із двох основних частин: ядра мови й стандартної бібліотеки.

Стандартна бібліотека C++ увібрала в себе бібліотеку шаблонів STL, що розроблялася одночасно зі стандартом. Зараз назва STL офіційно не вживається, однак у колах програмістів на C++ ця назва використовується для позначення частини стандартної бібліотеки, що містить визначення шаблонів контейнерів, ітераторів, алгоритмів і функторів.

Стандарт C++ містить нормативне посилання на стандарт C і не визначає самостійно ті функції стандартної бібліотеки, які запозичаться зі стандартної бібліотеки C.

Крім того, існує величезна кількість бібліотек C++, що не входять у стандарт. У програмах на C++ можна використовувати багато бібліотек C.

Стандартизація визначила мову програмування C++, однак за цією назвою можуть ховатися також неповні, обмежені, достандартні варіанти мови. На початку мова розвивалася поза формальними рамками, спонтанно, у міру завдань, що ставилися перед нею. Розвитку мови сприяв розвиток кросс-компілятора cfront. Нововведення в мові відбивалися в зміні номера версії

кросс-компілятора.

Мова C++ багато в чому є надмножиною C. Нові можливості C++ включають оголошення у вигляді виразів, перетворення типів у вигляді функцій, оператори new і delete, тип bool, посилання, розширене понятя константності функції, що підставляються, аргументи за замовчуванням, перевизначення, простори імен, класи (включаючи й усі, пов'язані із класами, можливості, такі як спадкування, функції-члени, віртуальні функції, абстрактні конструктори), перевизначення операторів, шаблони, оператор ::, обробку виключень, динамічну ідентифікацію й багато чого іншого. Мова C++ також у багатьох випадках суворіше ставиться до перевірки типів, ніж C.

В C++ з'явилися коментарі у вигляді подвійної косої риски (//), які були в попереднику C – мові BCPL.

Деякі особливості C++ пізніше були перенесені в C, наприклад, ключові слова const і inline, оголошення в циклах for і коментарі в стилі C++. У більш пізніх реалізаціях C також були представлені можливості, яких немає в C++, наприклад, макроси vararg і поліпшена робота з масивами-параметрами.

Стандартна бібліотека C++ включає стандартну бібліотеку C з невеликими змінами, які роблять її більш підходящою для мови C++. Інша більша частина бібліотеки C++ заснована на Стандартній Бібліотеці Шаблонів (STL). Вона надає такі важливі інструменти, як контейнери (наприклад, вектори й списки) та ітератори (узагальнені покажчики), що надають доступ до цих контейнерів як до масивів. Крім того, STL дозволяє подібним чином працювати й з іншими типами контейнерів, наприклад, асоціативними списками, стеками, чергами.

Використовуючи шаблони, можна писати узагальнені алгоритми, здатні працювати з будь-якими контейнерами або послідовностями, обумовленими ітераторами.

Так само, як і в C, можливості бібліотек активізуються використанням

директиви #include для включення стандартних файлів. Усього в стандарті C++ визначено 50 таких файлів.

STL до включення в стандарт C++ була сторонньою розробкою, на початку – фірми HP, а потім SGI. Стандарт мови не називає її «STL», тому що ця бібліотека стала невід'ємною частиною мови, однак багато з людей дотепер використовують цю назву, щоб відрізняти її від іншої частини стандартної бібліотеки (потоки введення/виводу (iostream), підрозділ C й ін.).

C++ додає до C об'єктно-орієнтовані можливості. Вона уводить класи, які забезпечують три найважливіші властивості об’єктно-орієнтованого програмування (ООП): інкапсуляцію, наслідуванння і поліморфізм.

C++ – надзвичайно потужна мова, що містить засоби створення ефективних програм практично будь-якого призначення, від низькорівневих утиліт і драйверів до складних програмних комплексів будь-якого призначення. Зокрема можна виділити такі переваги мови:

– підтримуються різні стилі й технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування, метапрограмуванння (шаблони, макроси);

– передбачуване виконання програм є важливою перевагою для побудови систем реального часу. Увесь код, неявно генерований компілятором для реалізації мовних можливостей (наприклад, при перетворенні змінної до іншого типу), визначений у стандарті. Також строго визначені місця програми, у яких цей код виконується. Це дає можливість виміряти або розраховувати час реакції програми на зовнішню подію;

– автоматичний виклик деструкторів об'єктів при їхньому знищенні, причому в порядку, зворотному виклику конструкторів. Це спрощує (досить оголосити змінну) і робить більш надійним звільнення ресурсів (пам'ять, файли, семафори й т.п.), а також дозволяє гарантовано виконувати переходи станів програми, не обов'язково пов'язані зі звільненням ресурсів (наприклад, запис у журнал);

– користувацькі функції-оператори дозволяють коротко і ємко записувати вирази над користувацькими типами в природній алгебраїчній формі;

– мова підтримує понятя фізичної (const) і логічної (mutable) константності. Це робить програму надійнішою, тому що дозволяє компілятору, наприклад, діагностувати помилкові спроби зміни значення змінної. Оголошення константності дає програмістові, що читає текст програми додаткове уявлення про правильне використання класів і функцій, а також може бути підказкою для оптимізації. Перевантаження функцій-членів за ознакою константності дозволяє визначати зсередини об'єкта мету виклику методу (константний для читання, неконстантний для зміни). Оголошення mutable дозволяє зберігати логічну константність при використанні кешів і ледачих обчислень;

– використовуючи шаблони, можливо створювати узагальнені контейнери і алгоритми для різних типів даних, а також спеціалізувати й обчислювати на етапі компіляції;

– можливість імітації розширення мови для підтримки парадигм, які не підтримуються компіляторами прямо. Наприклад, бібліотека Boost.Bind дозволяє зв'язувати аргументи функцій;

– можливість створення вбудованих об’єктно-орієнтованих мов програмування. Такий підхід використовує, наприклад, бібліотека Boost.Spirit, що дозволяє задавати Ebnf-Граматику парсерів безпосередньо в коді C++;

– використовуючи шаблони й множинне спадкування можна імітувати комбінаторну параметризацію бібліотек. Такий підхід застосований у бібліотеці Loki, клас Smartprt якої дозволяє, управляючи всього декількома параметрами часу компіляції, згенерувати близько 300 видів “розумних покажчиків” для керування ресурсами;

– кросплатформенність: стандарт мови накладає мінімальні вимоги

на ЕОМ для запуску скомпільованих програм. Для визначення реальних властивостей системи виконання в стандартній бібліотеці присутні відповідні можливості (наприклад, std::numeric_limits <T>). Доступні компілятори для великої кількості платформ, мовою C++ розробляють програми для різних платформ і систем;

– ефективність. Мова спроектована так, щоб дати програмістові максимальний контроль над усіма аспектами структури й порядку виконання програми. Жодна з мовних можливостей, що приводить до додаткових накладних витрат, не є обов'язковою для використання – при необхідності мова дозволяє забезпечити максимальну ефективність програми;

– є можливість роботи на низькому рівні з пам'яттю, адресами;

– висока сумісність із мовою C, що дозволяє використовувати весь існуючий C-код (код на C може бути з мінімальними переробками скомпільований компілятором C++; бібліотеки, написані на C, звичайно можуть бути викликані з C++ безпосередньо без яких-небудь додаткових витрат, у тому числі й на рівні функцій зворотного виклику, дозволяючи бібліотекам, написаним на C, викликати код, написаний на C++).

Для того, щоб програмний засіб захисту інформації на основі алгоритму RSA, стійкого до атаки енергоспоживання, був правильно побудований та описаний, його необхідно розвити на окремі модулі. Кожен модуль реалізується на мові C++ за допомогою відповідних функцій.

Розроблюваний програмний засіб складається з таких основних модулів:

– isPrime – визначення простих чисел;

– KeyGen – генерування ключів;

– isCoprime – перевірка взаємної простоти чисел;

– BRIP – шифрування даних;

– A1modB – обчислення оберненого числа за модулем.

3.2 Реалізація програмного засобу на основі криптоалгоритму RSA, стійкого до атаки аналізу енергоспоживання

Для вирішення завдання, поставленого в першому розділі, було спроектовано і реалізовано алгоритм захисту інформації RSA, стійкий до аналізу енергоспоживання з використанням мови C++.

Для шифрування з допомогою RSA користувач повинен ввести прості числа *p* і *q*. Ввід значень зміних *p* і *q* відбувається у відповідні поля вікна програми. На рисунку 3.1 показані дані поля.

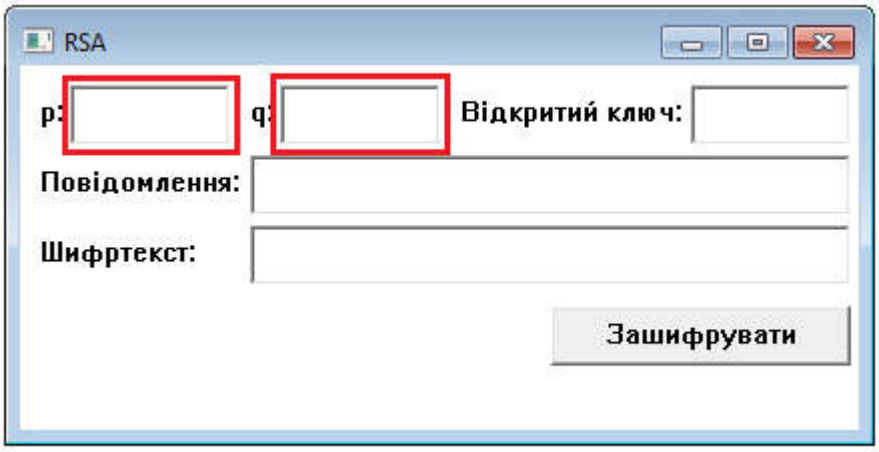

Рисунок 3.1 – Поля для вводу простих чисел.

Зчитування введених даних і запис їх у змінні відбувається за допомогою наступного коду:

```
GetWindowText(hp, t, 10);
int p = atoi(t);
GetWindowText(hq, t, 10);
int q = atoi(t);
```

Після зчитування проводиться перевірка введених чисел на простоту:

```
if(!isPrime(p))
{
	MessageBox(hwnd, "P не просте", "", 0);
	SetWindowText(hp, NULL);
break;
}
if(!isPrime(q))
{
	MessageBox(hwnd, "Q не просте", "", 0);
	SetWindowText(hq, NULL);
	break;
}
```

Дана перевірка відбувається за допомогою функції isPrime:

```
bool isPrime(int z)
{
	for(int i = 2; i < z; i++)
	{
		if(z%i)
		{
			continue;
		}
		else
		{
			return false;
```

```
            }
        }
        return true;
    }
```

Блок схема алгоритму роботи ДП. КСМ. 07143/14.00.00.000А2

Наступним кроком в алгоритмі шифрування є вибір відкритого ключа. У програмі вибір реалізовано за допомогою функції-генератора псевдовипадкових чисел. Обране число має задовольняти умовам: $1 < e < \varphi(n)$ та e – взаємно-просте з $\varphi(n)$. Генерацію ключа реалізовано за допомогою функції KeyGen:

```
int KeyGen(int F_n)
{
    srand(time(NULL));
    int rnd = rand() % F_n + 1;
    while(!isCoprime(rnd, F_n))
    {
        rnd = rand() % F_n + 1;
    }
return rnd;
}
```

Функція приймає значення функції Ейлера для чисел *p* і *q*, обчисленої за допомогою виразу $Fn = (p-1)(q-1)$. Для перевірки на взаємнопростість використовується функція isCoprime:

```
bool isCoprime(int a, int b)
{
    int c;
```

```
        while (b)
        {
                c = a % b;
                a = b;
                b = c;
        }
        if(a == 1)
                return true;
        else
                return false;
}
```

Після генерації відкритого ключа можна виконувати процедуру шифрування. Користувач вводить повідомлення для шифрування у відповідне поле програми (рисунок 3.3).

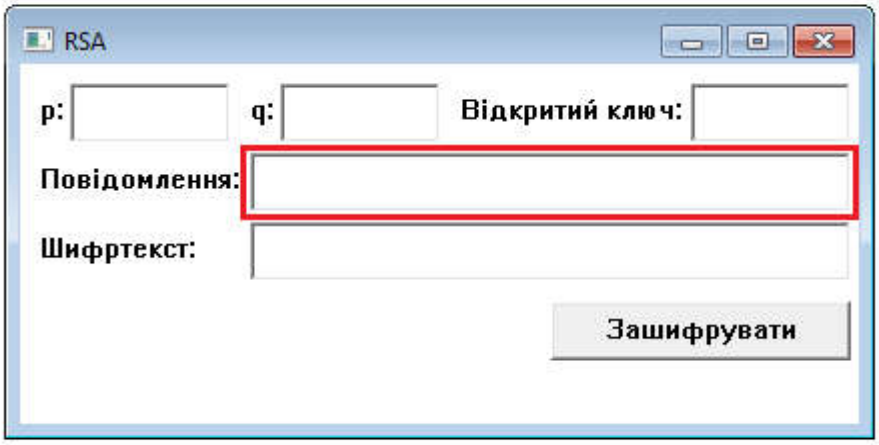

Рисунок 3.3 – Поле для вводу повідомлення

Повідомлення розбивається на блоки, які поступають на вхід функції шифрування. Сам процес шифрування реалізовано у функції BRIP, яка приймає на вхід три змінних: відкритий ключ, блок для шифрування і число

$N = p * q$.

```
int BRIP(int d, int m, int modulus)
{
    int r = 0;
    srand(time(NULL));
    r = rand() % (modulus - 1) + 1;
    while(!isCoprime(r, modulus))
    {
        r = rand() % (modulus - 1) + 1;
    }
    int T[3];
    T[0] = r;
    T[1] = A1modB(r, modulus);
    T[2] = T[1] * m;
    int en = log(d)/log(2) + 1;
    char *cd = new char[en];
    int td = 0;
    for(int i = 0; i < en; i++)
    {
        td = d / pow(2, i);
        cd[en - (i + 1)] = td % 2;
    }
    for(i = 0; i < en; i++)
    {
        T[0] = ost((T[0] * T[0]), modulus);
        T[0] = ost((T[0] * T[cd[i] + 1]), modulus);
    }
    int C = ost(T[0] * T[1], modulus);
    return C;
```

```
}
```

Функція повертає число C, яке є результатом операції шифрування. Функція генерує випадкове число *r*, за тим самим алгоритмом, що і відкритий ключ. Також тут використовується функція для знаходження оберненого числа за певним модулем - A1modB. На вхід подається число, для якого шукають обернене і модуль.

```
int A1modB(int a, int b)
{
	int tb = b;
	int x = 0;
	tb++;
	for(;;)
	{
		if(tb%a)
		{
			tb += b;
		}
		else
		{
			x = tb / a;
			return x;
		}
	}
}
```

Схема структурна КС, що використовує RSA для шифоування подана на ДП. КСМ. 07143/14.00.00.000.С1.

Після завершення шифрування усіх блоків вони збираються в один масив і виводяться у полі програми (рисунок 3.4).

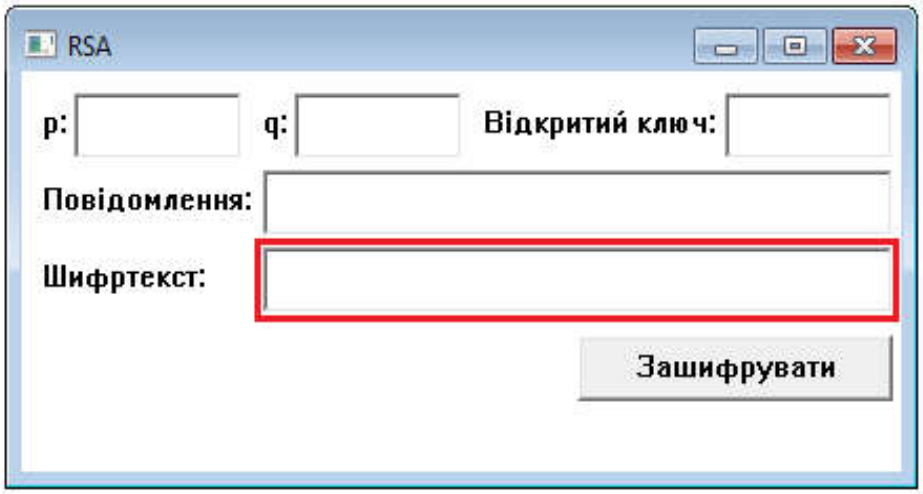

Рисунок 3.4 – Поле для виведення зашифрованого повідомлення

Для початку шифрування необхідно натисну кнопку “Зашифрувати”. По закінченні процедури у відповідні поля будуть виведені відкритий ключ і шифротекст.

3.3 Тестування та верифікація розробленого програмного засобу

Оскільки програма призначена для забезпечення конфіденційності даних та повного розмежування доступу до інформації, яка може становити комерційну таємницю, зашифрований текст абсолютно має бути нечитабельним і не повинен містити ніякої інформації. Інформація може бути одержана лише тоді, коли відбудеться процес дешифрування цього файлу.

В процесі тестування було введено: p=29 і q=19. Для прикладу було взято повідомлення “diplom”. Результати тестування програми приведено на рисунку 3.5.

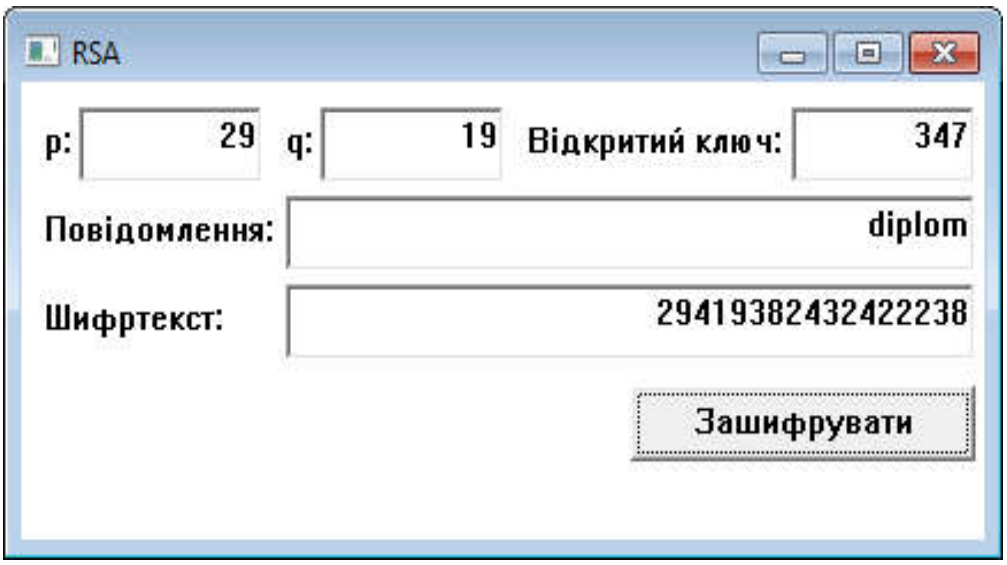

Рисунок 3.5 – Результати тестування

Перевірити результати можна наступними обрахунками:

$N = p * q = 29*19 = 551$

$\varphi(N) = 28*18 = 504$

$e = 347$

$1 < 347 < 504$

504=347*1-157

347=157*2-33

157=33*4-25

33=25*1-8

25=8*3-1

e взаємнопросте з $\varphi(N)$

ASCII коди повідомлення для шифрування:

d=100

i=105

p=112

l=108

o=111

m=109

$100^{347} \bmod 551 = 294$

$105^{347} \bmod 551 = 193$

$112^{347} \bmod 551 = 82$

$108^{347} \bmod 551 = 432$

$111^{347} \bmod 551 = 422$

$109^{347} \bmod 551 = 238$

Варто зазначити, що швидкість генерування ключів шифрування залежить від частоти процесора компютера. В таблиці 3.1 подано часові характеристики генерування ключів різної довжини на компютері AMD Duron 700 в процесі тестування. Звичайно, на практиці використовуються ключі значно більшої розмірності (1024 біт, 2048 біт і т.д.).

Таблиця 3.1 – Швидкість генерування ключів шифрування

Довжина ключа	Час генерування			
32 біт	0,059454 сек.	0,054945 сек.	0,052944 сек.	0,354947 сек.
64 біт	0,109893 сек.	0,989323 сек.	0,108893 сек.	0,119891 сек.
128 біт	0,209830 сек.	0,229536 сек.	0,219835 сек.	0,214831 сек.
256 біт	0,398830 сек.	0,418834 сек.	0,399731 сек.	0,378832 сек.

Отже, розроблений програмний засіб для реалізації процесу шифрування криптосистеми RSA, стійкої до атаки аналізу енергоспоживання, має доступний інтерфейс, є стійким і надійним, а тому може успішно використовуватись для захисту конфіденційної інформації.

4 ТЕХНІКО-ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Розрахунок витрат на розробку програмного забезпечення

У розробці проектного рішення задіяні наступні спеціалісти - розробники, а саме: керівник проекту (К); студент-дипломник (С); консультант техніко-економічного розділу (КТЕО) [17].

Форму поділу робіт по всіх основних етапах і видах робіт, які повинні бути виконані показано в таблиці 4.1.

Таблиця 4.1 - Середній час виконання проекту та стадії технологічного процесу

№ п/п	Назва операції (стадії)	Виконавець, посада	Середній час виконання операції, год.
1.	Підготовка	Студент	7
2	Розробка проекту системи	Керівник ДП	16
		Консультант ТЕО, доцент	2
		Студент	216
3	Проектування технічної частини системи	Студент	16
4	Розробка програмного продукту системи	Студент	2
5	Встановлення та налаштування прогр. зас.	Студент	4
6	Тестування системи	Студент	2
Разом			265

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування.

До складу елементу витрат "Витрати на оплату праці" включається заробітна плата за окладами і тарифами, премії та заохочення, компенсаційні

виплати, оплата відпусток та іншого невідпрацьованого часу, інші витрати на оплату праці.

Витрати на оплату праці розробників проекту визначаються за формулою:

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij} \ , \qquad (4.1)$$

де n_{ij} – чисельність розробників i-ої спеціальності j-го тарифного розряду, осіб;

t_{ij} – затрачений час на розробку проекту співробітником i-ої спеціальності j-го тарифного розряду, год;

C_{ij} – годинна ставка працівника i-ої спеціальності j-го тарифного розряду, грн.

Середньо годинна ставка працівника може бути розрахована за формулою:

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{РЧ_{i}}, \qquad (4.2)$$

де C_{ij} – основна місячна заробітна плата розробника i-ої спеціальності j-го тарифного розряду, грн.;

h – коефіцієнт, що визначає розмір додаткової заробітної плати (при умові наявності доплат);

$РЧ_i$ - місячний фонд робочого часу працівника i-ої спеціальності j-го тарифного розряду, год. (приймаємо 168 год.)

Таблиця 4.2 - Вихідні дані для розрахунку витрат на оплату праці

№ п/п	Посада виконавців	Місячний оклад (стипендія), грн.	Коефіцієнт Додаткової з/п	Сума
1	Керівник ДП, доцент, к.т.н.	5286	0,94	10254,84
2	Консультант техніко-економічного розділу, доцент	6026	1,47	14884,22
3	Студент	1287	0	1287

Звідси, загальні витрати на оплату праці ($B_{ОП}$) дорівнюють:

$$B_{ОП} = 16 \cdot \frac{10254,84}{168} + 2 \cdot \frac{14884\ ,22}{168} + 247\ \cdot \frac{1287}{168} = 3046\ \text{(грн.)}$$

Крім того, слід визначити відрахування на соціальні заходи. Величну відрахувань у спеціальні державні фонди визначають у відсотковому співвідношенні від суми основної та додаткової заробітних плат. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 20,5% від суми заробітної плати:

$$B_{Ф} = \frac{20,5}{100} \cdot 3046 = 624\ \text{(грн.)}$$

Матеріальні витрати — це вартість витрачених матеріалів, малоцінних та швидкозношуваних предметів на виробництво продукції, робіт або послуг, а також матеріалів і МШП, витрачених на адміністративні, збутові та інші потреби підприємства [17].

Загальна сума витрат на матеріальні ресурси (B_M) визначається за формулою:

$$B_M = \sum_{i=1}^{n} K_i \cdot Ц_i, \qquad (4.3)$$

де K_i - витрата i-го типу матеріалу, натуральні одиниці вимірювання;

$Ц_i$ - ціна за одиницю i-го типу матеріалу, грн.;

i- тип матеріального ресурсу;

n - кількість типів матеріальних ресурсів.

Звідси, витрати на матеріальні ресурси дорівнюватимуть:

$$B_M = 263{,}0 \text{ грн.}$$

Проведені розрахунки занесемо у таблицю 4.3.

Таблиця 4.3- Розрахунок витрат на матеріали та комплектуючі

№ п/п	Найменування купованих виробів	Одиниця виміру	Ціна, грн	Кількість купованих виробів	Сума, грн	Транспортні витрати (10% від суми)	Загальна сума, грн
1	Папір (формат А4)	уп	80,0	2	160,00	16,0	176,0
2	Ручка кулькова	шт	5,0	2	10,00	1,0	11,0
3	Олівець простий	шт	2	2	4,00	0,4	4,40
4	Диски CD-R	шт	5,0	2	10,00	1,0	11,10
5	Зошит, 96 арк	шт	15,0	1	15,0	1,5	16,5
6	Тонер для принтера	уп	40,0	1	40,0	4,0	44,0
Разом							263,0

Якщо для розробки КС використовується електрообладнання, то необхідно розрахувати витрати на електроенергію.

Загальна сума витрат на електроенергію розраховується за формулою:

$$B_E = \sum_{i=1}^{n} P_i \cdot k_i \cdot T_i \cdot Ц, \qquad (4.4)$$

де P_i - паспортна потужність *i*-го електрообладнання, кВт;

k_i - коефіцієнт використання потужності *i*-го електрообладнання (приймається 0,7 , 0,9);

T_i - час роботи *i*-го устаткування за весь період розробки, год;

Ц - ціна електроенергії, грн / кВт· год;

i - тип електрообладнання;

n - кількість електрообладнання.

Для розробки проекту даної системи використовується один ноутбук потужністю P = 0,5 кВт, який за весь період розробки працює 200 годин, та друкуючий пристрій потужністю P = 0,37 кВт, який працює 2 години.

Проміжні розрахунки на витрату електроенергії подані в таблиці 4.4.

Таблиця 4.4 - Витрати на електроенергію

Найменування устаткування	Паспортна потужність, кВт	Коефіцієнт використання потужності	Час роботи обладнання для розробки, год	Ціна електроенергії, $\frac{\text{грн}}{\text{кВт} \cdot \text{год}}$	Сума, грн.
Ноутбук	0,5	0,9	200	0,98	88,2
Принтер	0,37	0,9	2	0,98	0,653
Разом					88,85

Амортизація – це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення. Амортизаційні відрахування використовуються для повної реновації, а також для їх часткового відновлення, тобто на модернізацію або капітальний ремонт.

Для визначення амортизаційних відрахувань застосуємо метод прямолінійного списання. Загальна сума амортизаційних відрахувань ($В_{АМ}$) визначається за формулою:

$$B_{AM} = \sum_{i=1}^{n} \frac{B_i \cdot H_i}{100}, \qquad (4.5)$$

де B_i - вартість i-го устаткування на початок звітного періоду, грн.;

H_i - річна норма амортизації i-го устаткування, %;

i - тип обладнання;

n - кількість устаткування.

Для проектування даної системи використовувався один ноутбук 6700 грн., та принтер вартістю 3000 грн.

Тоді:

$$B_{AM} = \frac{6700{,}0 \cdot 10}{100} + \frac{3000{,}0 \cdot 20}{100} = 1270{,}0\,\text{грн.}$$

Таблиця 4.5 - Амортизація основних фондів

Найменування устаткування	Вартість устаткування, грн.	Річна норма амортизації, %	Сума, грн.
Ноутбук	6700,0	10	670,0
Принтер	3000,00	20	600,0
Разом			1270,0

Транспортні витрати слід прогнозувати у розмірі 8–12 % від загальної суми матеріальних витрат.

$$B_T = 0{,}12 \cdot B_M, \qquad (4.6)$$

де B_T – транспортні витрати.

$$B_T = 0{,}12 \cdot 263{,}0 = 31{,}56 \text{ грн.}$$

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці можуть становити 60–100 % від суми основної та додаткової заробітної плати працівників. Накладні витрати для даного проекту подані далі [17].

$$H_B = 0{,}7 \cdot B_{ОП}\,, \tag{4.7}$$

де H_B – накладні витрати.

$$H_B = 0{,}7 \cdot 3046 = 2132{,}2 \text{ грн.}$$

4.2 Складання кошторису витрат та розрахунок ціни проекту

Загальні витрати ($B_{КС}$) розрахуємо за формулою:

$$B_{КС} = B_{ОП} + B_Ф + B_М + B_Е + B_{АМ} + B_Т + H_B \tag{4.8}$$

Тобто:

$$B_{КС} = 7455{,}61 \text{грн.}$$

Результати проведених розрахунків зведемо у таблицю 4.6.

Таблиця 4.6 - Кошторис витрат

Зміст витрат	Сума, грн.
Витрати на оплату праці (осн. і дод. ЗП)	3046
Відрахування на соціальні заходи	624
Матеріальні витрати	263,0
Витрати на електроенергію	88,85
Амортизаційні відрахування	1270
Транспортні витрати	31,56
Накладні витрати	2132,2
Разом	7455,61

Договірна ціна ($Ц_Д$) для проектних рішень розраховується за формулою:

$$Ц_Д = B_{КС} \cdot \left(1 + \frac{p}{100}\right), \quad (4.9)$$

де $B_{КС}$ – кошторисна вартість, грн.;

p - середній рівень рентабельності, % (приймаємо 30% за погодженням з керівником).

$$Ц_Д = 7455,61 \cdot (1+0,3) = 9692,3 \text{ грн.}$$

4.3 Визначення економічної ефективності і терміну окупності капітальних вкладень

Економічна ефективність — досягнення найбільших результатів за найменших затрат живої та уречевленої праці. Економічна ефективність є конкретною формою дії закону економії часу.

За капіталістичного способу виробництва узагальнюючий показник

економічної ефективності — норма прибутку.

Економічна ефективність (E_P) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_P = \frac{П}{B_{KC}} , \qquad (4.10)$$

де $П$ – прибуток, грн.;

B_{KC} – кошторисна вартість, грн..

$$E_P = 2131{,}2 \text{ грн.} / 7455{,}61 \text{ грн.} = 0{,}3.$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень (T_p):

$$T_P = \frac{1}{E_P} . \qquad (4.11)$$

Тобто:

$$T_P = 1/0{,}3 = 3{,}3 \text{ р.}$$

Прийнятним вважається термін окупності близький до 7 років.

Розраховані економічні показники проекту занесемо до таблиці 4.7.

Таблиця 4.7 - Економічні показники розробки

№ п/п	Показник	Значення
1.	Собівартість, грн.	7455,61
2.	Плановий прибу’ток, грн.	2367,99
3.	Ціна, грн.	9692,3
4.	Економічна ефективність	0, 3
5.	Термін окупності, рік	3,3

Враховуючи основі економічні показники з таблиці 4.7, можна зробити висновок, що при економічній ефективності 0,3 та терміні окупності – 3,3 роки проводити роботи по впровадженню даного програмного засобу є доцільним та економічно вигідним.

ВИСНОВКИ

Основною метою дипломної роботи – було потрібно розробити програмний засіб шифрування даних на основі стійкого до сучасних атак кристоалгоритму, який є правельзгідно його специфіки. Це було досегнено шляхом проектування алгоритму RSA. Правильність і функціональність роботи алгоритму підтверджено тестуванням і аналізом реальних результатів роботи симуляї.

В результаті проведеного проектування:

1) проаналізовано сучасні системи шифрування даних;

2) досліджено сучасні методи шифрування даних. Крім того розгянуто області застосування алгоритму RSA;

3) обрано мову проектування C++. Після чого був створений алгоритм шифрування RSA;

4) проаналізувавши алгоритм роботи було вибрано середовище моделювання в якому буде розроблятися криптоалгоритм шифрування RSA. Опісля була здійснена верифікація і тестування розробленого алгоритму.

Розроблений програмний засіб шифрування даних має свої як позитивні так і негатисні сторони. До переваг можна віднести те що, алгоритм RSA є асиметричним, тобто він полягає в поширенні відкритих ключів у мережі. Це дозволяє кільком користувачам обмінюватися інформацією, по незахищеним каналів зв’язку. Користувач сам може змінювати як числа, і відкритий і закритий ключ на власний розсуд, потім він має поширити відкритий ключ у мережі. Це дає можливість збільшити криптостійкість.

До недоліків можна віднести невисоку швидкість роботи та виявлення граматичний помилок під час роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. PHP: Мова програмування [Електронний ресурс] Режим доступу: http://www.php.net

2. Лавріщева К.М. Програмна інженерія / К. М. Лавріщева // - К.: Видавництво «Академперіодика», 2008.-319 с.

3. Ларман К. Применение UML и шаблонов проектирования. / К. Лармак // - М.: Издат. дом «Вильямс», 2002. - 617 с.

4. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. / А.М. Вендров // - М.: Финансы и статистика, 1998.

5. Липаева В. В. 'Программная инженерия. Методологические основы. Инженерия программного обеспечения. / В.В.Липаева // - М. : Издательский дом «Вильямс», 2002. - 624 с.

6. Петюшкин А. В. HTML в дизайне. / А.В. Петюшкин // - СПб .: БХВ-Петербург, 2004. — 400с.: ил. ISBN 5-94157-513-0.

7. Лекае В. А. Некоторые аспекты разработки, создания и эксплуатации web-сайтов и порталов / В.А. Лекае, М. А. Григорьева // Межотраслевая информационная служба. – 2007. - №2. – С.44-47

8. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. / A.J. Menezes, P.C. van Oorschot, S.A. Vanstone // - CRC Press, 1996. — 794 p.

9. Брассар Ж. Современная криптология / Ж. Брассар // - М. : Полимед, 1999. — 176 с.

10. Земор Ж. Курс криптографии / Ж. Земор // - Ижевск : РХД, 2006. — 256 с.

11. Коутинхо С. Введение в теорию чисел. Алгоритм RSA / С. Коутинхо // - М. : Постмаркет, 2001. — 328 с.

12. Тилборг И.Основы криптологии / И. Тилборг // - М. : Мир, 2006. — 472 с.

13. Ян С. Криптоанализ RSA / С. Ян // - Ижевск : РХД, 2011. — 312 с.

14. Ященко В. Введение в криптографию / В. Ященко // - М. : МЦНМО, 2012. — 348 с.

15. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер // - М.: Триумф, 2002. — 816 с. — 3000 экз. — ISBN 5-89392-055-4.

16. Фергюсон Н. Практическая криптография. / Н. Фергюсон, Б. Шнайер // - М. : Диалектика, 2004. — 432 с. — 3000 экз. — ISBN 5-8459-0733-0, ISBN 0-4712-2357-3.

17. Bakhtiari M., Maarof M. Serious Security Weakness in RSA Cryptosystem / M. Bakhtiari, M. Maarof //- IJCSI — 2012. — Vol. 9, Iss. 1, No 3. — P. 175–178. — ISSN 1694-0814; 1694-0784

18. Boneh D. Twenty Years of attacks on the RSA Cryptosystem / D. Boneh // - AMS, 1999. — Vol. 46, Iss. 2. — P. 203–213. — ISSN 0002-9920

19. Rivest R. Method for obtaining digital signatures and public-key cryptosystems / R. Rivest, A. Shamir, L. Adleman // - New York City: ACM, 2003. - 356 p.

20. Чмора А.Л. Силовая атака на основе распределенных вычислений // Современная прикладная криптография. / А.Л. Чмора //- 2002. — 2000 экз. — ISBN 5-85438-046-3.

21. Багрова І.В. Організація виробництва / І.В Багрова // - Київ, ЦНЛ, 2005. – 248 с.

22. Скляров Д. Искусство защиты и взлома информации. / Д. Скляров // – СПб.: БХВ- Петербург, 2004. – 288 с.

23. Столлингс В. Операционные системы. / В. Столлингс // – Москва-Санкт-Петер- бург-Киев: Вильямс, 2002. – 845 с

24. Методичні рекомендації до виконання дипломного проекту з освітньо-кваліфікаційного рівня «Бакалавр» напряму підготовки 6.050102 «Комп'ютерна інженерія» фахового спрямування «Комп'ютерні системи та мережі» / О.М. Березький, Л.О. Дубчак, Р.Б. Трембач, Г.М. Мельник, Ю.М. Батько, С.В. Івасьєв / Під ред. О.М. Березького. – Тернопіль: ТНЕУ, 2016. – 60 с.

25. Методичні вказівки до написання техніко-економічного розділу для дипломних проектів на здобуття освітньо-кваліфікаційного рівня «Бакалавр» напряму підготовки 6.050102 «Комп'ютерна інженерія» / І.Р. Паздрій. – Тернопіль: ТНЕУ, 2015.- 36 с.