МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Західноукраїнський національний університет

Кафедра комп'ютерної інженерії

**ГОЛОСАЙ Дмитро Юрійович**

**Алгоритми підвищення стійкості криптосистеми RSA/ Algorithms for increased resistance of RSA cryptosystem**

спеціальність: 123 - Комп’ютерна інженерія

освітньо-професійна програма - Комп’ютерна інженерія

Кваліфікаційна робота

Виконав студент

групи КІм-21

Д. Ю. Голосай

Науковий керівник:

к.т.н., доцент Л.О.Дубчак

Кваліфікаційну роботу допущено

до захисту:

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ р.

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Л. О. Дубчак

ТЕРНОПІЛЬ – 2022

ВСТУП

**Актуальність досліджень.** На сучасному етапі розвитку криптографії існує багато різноманітних методів та засобів захисту інформації. Поширеними та популярними стали методи захисту, які базуються на асиметричній криптографії, оскільки її застосування дозволило вирішити задачу розподілу ключів та електронного цифрового підпису [1-5].

Асиметричні криптосистеми на сьогоднішній день досліджені менше, ніж симетричні, оскільки концепція їх побудови була запропонована відносно недавно.

Вагомим недоліком алгоритмів шифрування є низька швидкодія виконання основних математичних процедур, які використовуються для шифрування та дешифрування. Цей факт особливо характерний для реалізацій алгоритмів на пристроях із невеликими обчислювальними можливостями.

Одним із шляхів вирішення цієї проблеми є зменшення розмірності параметрів системи, але це може привести до зниження її стійкості. Іншою можливістю є застосування алгоритмів, призначених для ефективного здійснення основних математичних обчислень, які використовуються в асиметричних криптосистемах. При цьому ефективність, залежно від ситуації, може трактуватися не тільки як висока швидкість роботи, але і як мінімальний об’єм використаної пам’яті, мінімальний програмний код та їх сукупність [4,8,9].

В умовах розвитку сучасних інформаційних технологій задача захисту інформаційних ресурсів володіє певними особливостями. Природно, що для її вирішення необхідно використовувати апаратну реалізацію відомих алгоритмів криптографічного захисту інформації. Проте вимоги, які ставляться до них на сучасному етапі, обумовили появу принципово нових видів криптоаналізу, які умовно можна назвати “Атаки спеціальних впливів”, або ж “Атаки на основі нестандартних (побічних) каналів витоку інформації” (англ. мовою side-channel attacks, covert-channel attacks) [4, 5, 7]. Тому розробка підходів, методів, алгоритмів та засобів проектування криптографічних засобів захисту інформації, що є стійкими до такого виду атак є важливою та актуальною задачею.

Як відомо, операції шифрування та дешифрування в асиметричних криптосистемах здійснюються за допомогою обчислення відповідної до алгоритму важкооборотної функції. Тому найбільшого часу вимагають операції множення за модулем та піднесення до степеня за модулем крім того, безпосереднє здійснення вказаних операцій вимагає великого об’єму пам’яті, що негативно впливає на загальний час обробки інформації. Сучасні алгоритми модулярного експоненціювання базовані лише на операціях модулярного додавання, віднімання, множення, порозрядного зсуву чи звичайного множення або додавання. Тому вони суттєво прискорюють процес шифрування та дешифрування і криптосистеми з їх використанням можна реалізовувати на пристроях із невеликими обчислювальними можливостями.

Час виконання криптографічних операцій залежить не лише від ефективності реалізації конкретного алгоритму, але й також (інколи суттєво) від вхідних даних. Особливо така залежність проявляється для алгоритмів модулярного експоненціювання асиметричних криптосистем та алгоритмів додавання (множення) точок на еліптичній кривій. Як правило, ці криптографічні операції є обчислювально складними і для підвищення продуктивності виконання процедури шифрування повідомлення, чи формування цифрового підпису використовують спеціальні алгоритми, які базуються на оцінці інформації про кількість бітів у ключі шифрування [4, 8, 9]. Така оцінка в алгоритмах дозволяє пришвидшити виконання криптографічних операцій за рахунок обходу виконання деяких операцій алгоритму при нульових бітах ключа, що дає змогу значно підвищити продуктивність засобів захисту інформації. Звідси очевидно, що завжди можна виявити певну кореляцію між кількістю одиничних бітів ключа та часом виконання такого алгоритму. Саме така інформація дозволяє зловмиснику висунути гіпотезу щодо кількості одиничних та нульових бітів у секретному ключі, кількісним еквівалентом якої може бути вага Хемінга, і на основі такої оцінки здійснити атаку повного перебору в певному під-діапазоні ключового простору, що потребує значно менших обчислювальних ресурсів [5, 7]. Таким чином оцінка залежності часу виконання криптографічних операцій від ваги Хемінга ключової інформації дозволяє зловмиснику зменшити обчислювальну складність атаки на систему захисту інформаційних ресурсів. Атаки такого виду називають “часовим аналізом”.

Ефективність та стійкість криптосистеми до атак є конфліктними цілями. З одного боку, постійно підвищуються вимоги до безпеки, оскільки алгоритм повинен мати запас стійкості не тільки до відомих криптоатак, але і до нових методів криптоаналізу. З іншого боку, підвищуються вимоги до продуктивності засобів шифрування. Повне вирішення протиріч між стійкістю і ефективністю отримати неможливо проте, якщо розглядати стійкість як складову ефективності алгоритму та узгодити різні показники, гостроту вирішення цього питання можна знизити.

**Метою даної роботи** є розробка алгоритмів підвищеної стійкості асиметричної криптосистеми RSA.

Для досягнення мети необхідно вирішити наступні **завдання**:

1) розробити класифікацію сучасних атак на реалізацію;

2) здійснити розробку методики побудови захищених засобів захисту інформації до атаки аналізу енергоспоживання;

3) реалізувати та здійснити експериментальне дослідження програмного засобу реалізації криптоалгоритму RSA підвищеної стійкості.

**Об’єкт дослідження** – засоби захисту інформації, стійкі до атак на реалізацію.

**Предмет дослідження** ‒ процес захисту інформації до атак на реалізацію.

**Методами дослідження** є теорія ймовірності, математичне моделювання.

**Наукова новизна** полягає у вдосконаленні алгоритму захисту таємної інформації RSA, що має підвищену стійкість до атаки аналізу енергоспоживання.

**Практичне значення** одержаних результатів полягає у розробленому методі, що може застосовуватися у будь-якій системі захисту інформації.

**Публікації та апробації результатів досліджень** здійснено на науково-практичній конференції молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі», м.Тернопіль, 2022 р. ( додаток А).

**Впровадження результатів досліджень** здійснено в системі захисту підприємства (додаток Б).

Випускна кваліфікаційна робота складається з трьох розділів.

В **першому розділі** здійснено аналіз сучасного стану систем захисту інформації. Крім того, досліджено сучасні атаки на реалізацію, зокрема атаку аналізу енергоспоживання, та методи їх протидії.

В **другому розділі** розроблено методи підвищення стійкості криптоалгоритму. Проведено їх аналіз та доведено актуальність впровадження з метою захисту інформації.

**В третьому розділі** розроблено та здійснено реалізацію на рівні програмних засобів для подальшого його застосування в системах захисту інформації.

1 АНАЛІЗ ЕНЕРГОСПОЖИВАННЯ ЯК СУЧАСНИЙ ТИП АТАКИ НА КРИПТОсистеми

1.1 Сучасні атаки побічних каналів на засоби захисту інформації

Задача захисту інформаційних ресурсів постає особливо гостро в умовах розвитку сучасних інформаційних технологій. Постійне зростання об’ємів інформаційних ресурсів обумовлює жорсткі вимоги до засобів шифрування/дешифрування стосовно швидкості опрацювання вхідних даних. Природно, що для вирішення цієї задачі необхідно використовувати апаратну реалізацію відомих алгоритмів криптографічного захисту інформації.

Сучасні системи обробки таємної та конфіденційної інформації являють собою складні програмно-апаратні комплекси, що володіють специфічними каналами витоку інформації, що супроводжують штатний процес обробки інформаційних ресурсів.

Останнім часом одним із самих актуальних напрямків криптоаналізу стало здійснення атак, що використовують особливості реалізації та робочого середовища. Атаки по сторонніх, або побічних, каналах – це вид криптографічних атак, що використовують інформацію, отриману по сторонніх каналах (рисунок 1.1).

Звичайно вважається, що криптографічні обчислення реалізуються у вигляді ідеальних “чорних ящиків” у тому розумінні, що поточний стан обчислювального процесу закритий від стороннього спостерігача. Єдиною інформацією, доступною криптоаналітику, є загальна структура алгоритму шифрування, шифртекст і, найчастіше відкритий текст, що відповідає йому. З урахуванням цих припущень, рівень безпеки визначається виходячи з математичних властивостей криптоалгоритму й обчислювальних можливостей криптоаналітика. Помітимо, що й у класичному криптоаналізі наявність якої-небудь додаткової інформації суттєво знижує стійкість системи.

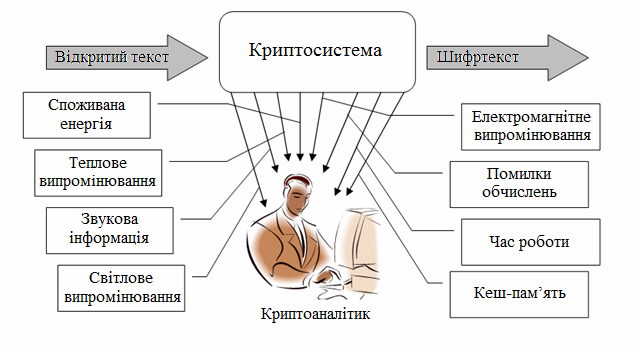


Рисунок 1.1 – Сторонні канали передачі інформації

У той же час криптографічний примітив можна розглядати як мінімум із двох точок зору: з одного боку, як абстрактний математичний об'єкт, а саме перетворення деяких вхідних даних у деякі вихідні, можливо, параметризоване ключем; з іншого боку, цей примітив повинен бути впроваджений у програму, яка буде запускатися на якомусь процесорі, у якомусь середовищі, і, отже, буде мати специфічні характеристики. У реальності криптоалгоритми завжди вбудовані в програмне забезпечення або апаратні засоби на фізичних пристроях, взаємодіючих з навколишнім середовищем і підданих її впливу.

Перша точка зору відповідає “класичному” криптоанналізу, друга – криптоаналізу по сторонніх каналах.

Під інформацією з побічних каналів розуміється інформація, яка може бути отримана із пристрою шифрування та не є при цьому ні відкритим текстом, ні шифртекстом. Майже всі здійснені на практиці вдалі атаки на криптосистеми використовують слабкості в реалізації й розміщенні механізмів криптоалгоритму. Такі атаки засновані на кореляції між значеннями фізичних параметрів, вимірюваних в різні моменти під час обчислень (споживання енергії, час обчислень, електромагнітне випромінювання й т.п.), і внутрішнім станом обчислювального пристрою, що мають відношення до секретного ключа.

На практиці атаки по сторонніх каналах на багато порядків більш ефективні, ніж традиційні атаки, основані тільки на математичному аналізі. При цьому атаки по побічних каналах використовують особливості реалізації (тому їх іноді називають також атаками на реалізацію - implementation attacks) для добування секретних параметрів, задіяних в обчисленнях. Такий підхід менш узагальнений, оскільки прив'язаний до конкретної реалізації, але найчастіше ефективніший, ніж класичний криптоаналіз.

В останні роки різко зросла кількість криптографічних атак, що використовують особливості реалізації й робочого середовища. Наприклад, зловмисник може відслідковувати енергію, споживану смарт-картою, коли вона виконує операції з закритим ключем, такі, як розшифрування, або генерація підпису. Зловмисник може також заміряти час, що затрачається на виконання криптографічної операції, або аналізувати поведінку криптографічного пристрою при виникненні певних помилок. Побічну інформацію на практиці зібрати нескладно, тому потрібно обов'язково враховувати таку загрозу при оцінці захищеності системи.

Атаки по сторонніх каналах класифікуються по наступних трьох типах:

• по контролю над обчислювальним процесом: пасивні й активні;

• по способу доступу до модуля: агресивні, напівагресивні і неагресивні;

• по методу, що застосовується в процесі аналізу: прості – simple side channel attack (SSCA) і диференційні – differential side channel attack (DSCA).

На сьогоднішній день виділено більше десяти сторонніх каналів. Атаки різняться по виду використовуваного стороннього каналу: атаки за часом виконання (Timing Attacks), атаки по енергоспоживанню (Power Analysis Attacks), атаки по помилках обчислень (Fault Attacks), атаки по електромагнітному випромінюванню (Electromagnetic Analysis), атаки по помилках у каналі зв'язку (Error Message Attacks). Існують і більш витончені види атак: атаки по кеш-пам'яті (Cache-based Attacks), акустичні атаки (Acoustic Attacks), атаки по світловому випромінюванню (Visible Light Attacks).

Атака за часом – спосіб одержання якої-небудь схованої інформації шляхом точного виміру часу, який потрібно затратити користувачеві для виконання криптографічних операцій. Це найперша з атак по сторонніх каналах, що з'явилася в цивільній криптографії. Найчастіше час обробки даних в криптосистемах небагато змінюється залежно від вхідних значень (наприклад, відкритого тексту або шифртексту). Це є наслідком оптимізації продуктивності й широкого кола інших причин. Атака за часом заснована на вимірі часу, необхідного модулю шифрування для виконання операції шифрування. Ця інформація може вести до розкриття інформації про секретний ключ. Наприклад, ретельно вимірюючи час, необхідний для виконання операцій із секретним ключем, зловмисник може знайти точне значення експоненти в алгоритмі Діффі-Хеллмана [19].

Атаки за часом наробили багато шуму в пресі в 1995 році: закриті ключі RSA можуть бути відновлені виміром відносних інтервалів часу, витрачених на добуток криптографічних операцій. Ці атаки були успішно застосовані до карток з мікропроцесорами й інших засобів надійної ідентифікації, а також до серверів електронної комерції в мережі.

Атака по аналізу енергоспоживання придатна, в основному, для апаратної реалізації криптографічних засобів і успішно застосовується при зломі смарт-карт та інших систем, у яких зберігається секретний ключ. Щоб виміряти споживану схемою енергію, необхідно послідовно з ланцюгом живлення або заземлення підключити резистор малого опору (наприклад, 50 Ом). Спадання напруги, ділене на опір, дасть силу струму. Сучасні лабораторії мають у своєму розпорядженні устаткування, здатне робити цифрові виміри напруги на винятково високих частотах (більш 1 Ггц) та з чудовою точністю (помилка менше 1%).

Атака по енергоспоживаннюможе бути розділена на просту (Simple Power Analysis, SPA) і диференційну (Differential Power Analysis, DPA). Метою SPA є одержання інформації про конкретні виконувані інструкції в системі й про конкретні оброблювані дані. У загальному випадку SPA може дати як відомості про роботу пристрою, так і інформацію про ключ. Для здійснення цієї атаки криптоаналітик повинен мати у своєму розпорядженні точні дані про реалізацію пристрою. Цей метод використовує безпосередні дані вимірів, зібрані під час виконання криптографічних операцій. Згідно з [20], проста атака по енергоспоживаннюдля смарт-карт звичайно займає кілька секунд, у той час як диференційна атака по енергоспоживаннюможе зайняти кілька годин.

На відміну від простих атак, різницеві атаки, засновані на аналізі споживаної енергії, включають в себе не лише візуальне представлення споживаної енергії, але також статистичний аналіз і статистичні методи виправлення помилок для одержання інформації про ключі. Більше того, DPA найчастіше не потребує даних про конкретну реалізацію й у якості альтернативи використовує статистичні методи аналізу. Диференційний аналіз енергії – одне із самих потужних засобів для проведення атак, що використовують сторонні канали, причому ця атака вимагає дуже маленьких витрат.

Помилки апаратного забезпечення, що з'являються під час роботи відповідного криптографічного модуля, або помилкові вихідні дані можуть стати важливими сторонніми каналами й іноді суттєво збільшують уразливість шифру до криптоаналізу. Криптоаналіз на основі формування випадкових апаратних помилок — це вид атаки на шифри у випадку, коли передбачуваний порушник має можливість здійснити на шифратор зовнішній фізичний вплив і викликати одиночні помилки в процесі шифрування одного блоку даних. Атаки по помилках на криптографічні алгоритми вивчаються з 1996 року і відтоді майже всі криптографічні алгоритми були піддані атакам такого виду.

Можливість здійснення атаки по помилках (або, принаймні, її ефективність) залежить від можливостей зловмисника викликати помилки в системі спеціально або користуватися збоями природного походження. Помилки найчастіше відбуваються через стрибки напруги, збої годинника або через випромінювання різних типів.

Розгляд питання стійкості до цього методу особливо актуально для шифраторів, застосовуваних в інтелектуальних електронних картках. В основному помилки класифікуються по наступних аспектах:

• точність, якої порушник може досягти при виборі часу й місця, де з'являється помилка під час роботи криптографічного модуля;

• довжина даних, на які впливає помилка; наприклад, тільки один біт;

• сталість помилки - чи є помилка короткочасною, або постійною;

• тип помилки; такі як зміна одного біта; зміна одного біта, але тільки в одному напрямку (наприклад, з 1 на 0); зміна біта на випадкове значення та ін.

Загалом, успішна атака по помилках на криптографічні модулі або пристрої вимагає двох кроків: крок створення помилки й крок використання помилки. Помилки можуть бути викликані в смарт-картах шляхом зовнішнього впливу на неї й переміщення її в неправильні умови. Деякі з них – аномальне й раптове зниження або підвищення напруги, частоти, температури, випромінювання, висвітлення й ін.

Різницевий аналіз по помилках полягає у вивченні результату роботи алгоритму шифрування в нормальних і ненормальних умовах при тому самому вході (відкритому тексті). Ненормальні умови звичайно досягаються створенням помилки в процесі (короткочасна помилка) або перед процесом (постійна помилка) роботи.

Диференційний аналіз по помилках широко вивчений з теоретичної точки зору і є придатним майже до всіх симетричних криптосистем.

Виконання обчислювальних операцій на комп'ютері пов’язане з виділенням електромагнітного випромінювання. Вимірюючи й аналізуючи це випромінювання, зловмисник може одержати значну інформацію про обчислення, що виконуються, і використовувані дані.

Для електромагнітнихканалів характерним є побічне випромінювання таких типів :

1. електромагнітне (ЕМ) випромінювання елементів технічних засобів обробки інформації (ТЗОІ) (носієм інформації є електричний струм, сила струму, напруга, частота або фаза якого змінюються за законом інформаційного сигналу);
2. ЕМ випромінювання на частотах роботи високочастотних генераторів ТЗОІ (внаслідок зовнішніх впливів інформаційного сигналу на елементах генераторів наводяться електричні сигнали, що можуть викликати незловмисну модуляцію власних високочастотних коливань, та випромінювання їх в оточуюче середовище);
3. ЕМ випромінювання на частотах самозбудження підсилювачів низької частоти ТЗОІ (самозбудження виникає внаслідок виникнення випадкових перетворень від’ємних зворотних зв’язків в паразитні додатні, що призводить до переведення підсилювача з режиму підсилення у режим автогенерування сигналу модульованого інформаційним сигналом).

Можливими причинами виникнення електричнихканалів витоку є:

* 1. наводки ЕМ випромінювань елементів ТЗОІ (виникають при випромінюванні ТЗОІ інформаційних сигналів, а також при наявності гальванічного зв’язку з’єднувальних ліній ТЗОІ та побічних провідників та ліній);
  2. проникнення інформаційних сигналів в мережі електропостачання (можуть виникати при наявності магнітного зв’язку між вихідним трансформатором підсилювача та трансформатором електропостачання, а також за рахунок нерівномірного навантаження на випростувальний пристрій, що приводить до зміни споживаного струму за законом зміни інформаційного сигналу);
  3. проникнення інформаційних сигналів в ланки заземлення (можуть виникати при наявності гальванічного зв’язку із заземленням різноманітних провідників, що виходять за межі зони контролю, в тому числі нульового проводу мережі електропостачання, екранів, металевих труб систем опалення та водопостачання, металевої арматури і т.п.);
  4. перехват інформації з використанням закладних пристроїв (являють собою міні передавачі, що встановлюються в ТЗОІ, випромінювання яких модулюються інформаційним сигналом і приймаються за межами зони контролю).

Оскільки компоненти комп'ютера є електричними пристроями, то вони випромінюють електромагнітне випромінювання під час виконання операцій. Зловмисник, який може спостерігати ці випромінювання й зрозуміти їхній зв'язок з обчисленнями, що виконуються, й використовуваними даними, можливо, зможе одержати значну інформацію про ці обчислення й самі дані. Заходам боротьби з електромагнітним випромінюванням присвячено багато літератури й велика кількість нормативних документів. Як і атаки по аналізу енергоспоживання, атаки по електромагнітному аналізу можуть бути також розділені на дві більші категорії: прості (SEMA) і диференційні (DEMA).

Можливість використання електромагнітних випромінювань уже відома у військових колах протягом тривалого часу. Наприклад, недавно розсекречений документ TEMPEST, виданий Національним Агентством Безпеки (NSA), досліджує різні компрометуючі випромінювання, включаючи електромагнітне і поширення акустичного сигналу. Є багато несекретної літератури по засобах таких атак і методах боротьби з ними. Наприклад, Кун (Kuhn) [10] описує спосіб здійснення й запобігання атак, заснованих на обробці інформації з екрана монітора, отриманої з його електромагнітного випромінювання. Quisquater [9] і Gandolfi [8] уперше представили результати експериментів для атак по аналізу ЕМ випромінювання криптографічних пристроїв, таких як смарт-карти, і порівняли такі атаки з атаками по аналізу енергоспоживання.

Відомі також атаки по такому сторонньому каналу, як світловий. Кун (Kuhn) [11] продемонстрував (як за допомогою складного аналізу, так і експериментально), що середня яскравість світла, випромінюваного монітором і відбитого від стіни, може допомогти при відновленні сигналу, зображеного на ньому. Особливістю такої атаки є те, що вона не вимагає фізичного доступу до пристрою.

Хоча більшість досліджень атак через побічні канали зосереджені на електромагнітних випромінюваннях, споживанні потужності й з недавнього часу розсіюванню видимого світла з CRT дисплеїв, Шамір в [12] продемонстрував попередній доказ того, що існує зв'язок між звуком процесора й проведеними ним обчисленнями.

Якщо процесор здійснює доступ до даних, які не зберігалися в кеш, тобто якщо виявляється не достатньо кеш-пам'яті, то в обчисленнях відбудеться деяка затримка, поки необхідні дані не будуть завантажені з основної пам'яті в кеш. Вимір такої затримки може дати можливість зловмиснику визначити частоту переповнення кешу. Саме тут і виникає побічний канал витоку, близько пов'язаний з атаками за часом [13].

Однією із основних вимог комплексного захисту є системний підхід, тому при виявленні технічних каналів витоку інформації необхідно розглядати усю сукупність елементів захисту, включаючи основне обладнання технічних засобів обробки інформації (ТЗОІ), кінцеві пристрої, з’єднувальні лінії, розподілюючі та комутаційні пристрої, системи електропостачання, заземлення і т.п.

Поряд із основними технічними засобами, що безпосередньо залучені до обробки та передачі інформаційних ресурсів, необхідно враховувати також допоміжні технічні засоби та системи (ДТЗіС) такі, як технічні засоби відкритого телефонного, факсимільного зв’язку, системи охоронної та пожежної сигналізації, електрифікації, радіофікації, електропобутові пристрої та інші струмопровідні металоконструкції.

Два нові проекти, що мають відношення до досліджень атак по сторонніх каналах у Європі, привернули увагу криптографічної громадськості, особливо тих, хто цікавиться такими атаками. Це проекти SCARD (Side Channel Analysis Resistant Design Flow) [16] та ECRYPT (European Network of Excellence for Cryptology) [17]. Обидва ці проекти – міжнародні спільні проекти Європейських дослідників із криптографічних науково-дослідних інститутів і відповідних галузей промисловості.

У проекті SCARD пропонується поліпшити стандартний хід розробки мікрочіпа – від високорівневого опису системи й опису транспортного рівня до мережевого списку вентильного рівня, а також розміщення й напрямок мікрочіпа – для забезпечення можливості розробки ланцюгів і систем, стійких до аналізу побічних каналів. Більше того, планується вивчити цілком, крок за кроком, явище побічного аналізу й надати підходящі інструменти для аналізу, а також інструменти для розроблювачів систем безпеки. Ці доповнення до традиційного ходу розробки вважаються необхідними для забезпечення можливості розробки наступного покоління захищених пристроїв.

ECRYPT – створена 4 роки назад мережа, фінансована Information Societies Technology Programme of the European Commission. Вона прагне до глобальної надійності й структури безпеки, і її мета – підсилити співробітництво Європейських дослідників, що займаються інформаційною безпекою, зокрема, криптологією. Для досягнення цієї мети, 32 ведучих учасника об'єднують зусилля в п'яти віртуальних лабораторіях, що спеціалізуються в різних областях дослідження, одна з яких – безпечні й ефективні реалізації (VAMPIRE). Одна із чотирьох робочих груп VAMPIRE – група дослідників, що займаються аналізом атак по сторонніх каналах.

Уже із цих двох проектів можна зробити висновок, що в Європі ведуться інтенсивні спільні міжнародні дослідження такого типу атак.

1.2 Атаки енергоспоживання

Більшість сучасних криптографічних пристроїв виконані на напівпровідникових логічних елементах, побудованих із транзисторів (рисунок 1.2). Електрони, що протікають через кремнієву підкладку, коли до транзисторного зазору прикладений заряд, споживають напругу й випускають електромагнітне випромінювання.

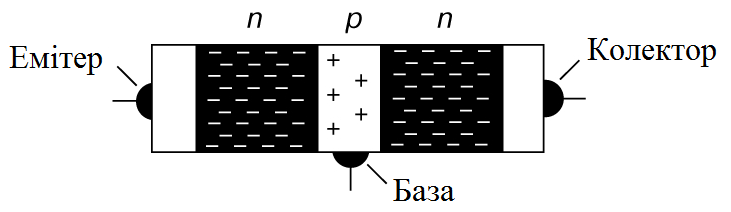


Рисунок 1.2 – Типовий транзистор

Електромагнітні випромінювання виникають як результат зміни потоків струму в шинах керування, вводу-виводу, обробки даних та інших частин криптопристрою. Ці зміни струму та ЕМ випромінювання можуть бути як очікуванні, так і неочікувані. Кожен компонент пристрою, через який протікає струм, випромінює ЕМ хвилі не лише свої власні (що базуються на його власних фізичних та електричних параметрах), але й результат ЕМ впливу від інших компонент внаслідок дії паразитних зв’язків та геометрії самої схеми. Зловмисник, як правило, зацікавлений у виявленні та накопиченні ЕМ випромінювань, спричинених операціями обробки даних. В пристроях CMOS (КМОН), в ідеалі, струм протікає лише у випадку, коли відбувається зміна логічного стану елементів пристрою. Ця зміна контролюється синхроімпульсом дельта-видної форми. Інколи це спричинює появу інших неочікуваних, “некорисних” ЕМ випромінювань. Такі ЕМ випромінювання несуть інформацію про потік струму, а відтак про події, що відбуваються протягом кожного синхроімпульсу. Оскільки кожен компонент криптопристрою виділяє різні типи ЕМ випромінювань, вони дозволяють отримати різні “описи” подій, що відбуваються протягом кожного синхроімпульсу.

Такі атаки, як диференційний аналіз, дозволяють проводити аналіз пристроїв з електромагнітним випромінюванням. Це не теоретичні атаки. Криптографічні дослідження успішно використовують ці атаки для аналізу великої кількості пристроїв, базованих на смарт-картах.

У той час, як деякі пристрої стійкі до простого аналізу енергоспоживання (Simple Power Analysis), дуже мало пристроїв протистоять диференційному аналізу напруг. Кількість часу, необхідного для проведення атаки, залежить від типу атаки (DPA, SPA) і від самого пристрою. SPA атаки зазвичай займають кілька секунд на картку, а DPA атаки можуть займати кілька годин.

Базові концепції нової методики розкриття були сформульовані у відомій роботі Пола Кочера (1995 р.) "Криптоаналіз на основі часової атаки" (Timing Attack Cryptoanalysis) [3], яка показала, що можна взламувати криптопристрої лише точно заміряючи інтервали часу, які їм потрібні на обробку даних. Криптографічні пристрої використовують секретний ключ для обробки вхідної інформації й/або для надання вихідної інформації. Розроблювачі протоколів припускають, що в атакуючих є вхідна й вихідна інформація, а інформація про ключ не відома. Атаки аналізом напруги працюють, тому що необхідна інформація часто доступна атакуючим.

Характеристики, необхідні для аналізу, можуть бути досить точно виміряні при виконанні пристроєм криптографічних операцій. Зокрема, простий амперметр, сконструйований з активного навантаження, може бути використаний для спостереження споживання енергії.

Інтегрований ланцюг побудований з окремих транзисторів, які діють як перемикачі, що контролюють живлення. Струм протікає через транзисторну підкладку, коли до зазору прикладений (або відведений) заряд. Цей струм потім доставляє заряд до зазорів інших транзисторів і інший ланцюг «завантажується». Рух електричного заряду витрачає енергію й робить електромагнітне випромінювання і ці два параметри ззовні можуть бути виміряні.

Тому, окремі транзистори представляють зовні спостережувану електричну поведінку. Так як мікропроцесорні логічні області виконують регулярне перемикання, це дає можливість для легкого визначення мікропроцесорних характеристик (таких як процесорна активність) простим спостереженням споживання енергії. Атаки типу диференційного аналізу живлення виконують більш складну інтерпретацію цих даних.

В атаках простим аналізом енергоспоживання, криптоаналітик безпосередньо досліджує споживання енергії системи. Кількість споживаної потужності змінюється в залежності від виконуваних мікропроцесором інструкцій. Більші обчислення, такі як DES раунди, RSA операції й.т.д. можуть бути ідентифіковані, оскільки операції, виконувані мікропроцесором, міняються дуже сильно в різних частинах цих операцій. При великому коефіцієнті підсилення окремі інструкції можуть бути помітні. Простий аналіз енергоспоживання може, наприклад, бути використаний для злому RSA реалізацій за допомогою виявлення відмінностей між операціями множення й добування кореня. Аналогічно, багато DES реалізацій мають видимі відмінності в перемішуваннях і зсувах та можуть у такий спосіб бути зламані, використовуючи SPA.

Рисунок 1.3 ілюструє SPA спостереження простої DES операції, виконаної звичайною смарт-картою. Верхня крива показує операцію шифрування, включаючи початкове перемішування 16 DES раундів, і кінцеве перемішування. Нижня крива детально показує 2-й і 3-й раунди.

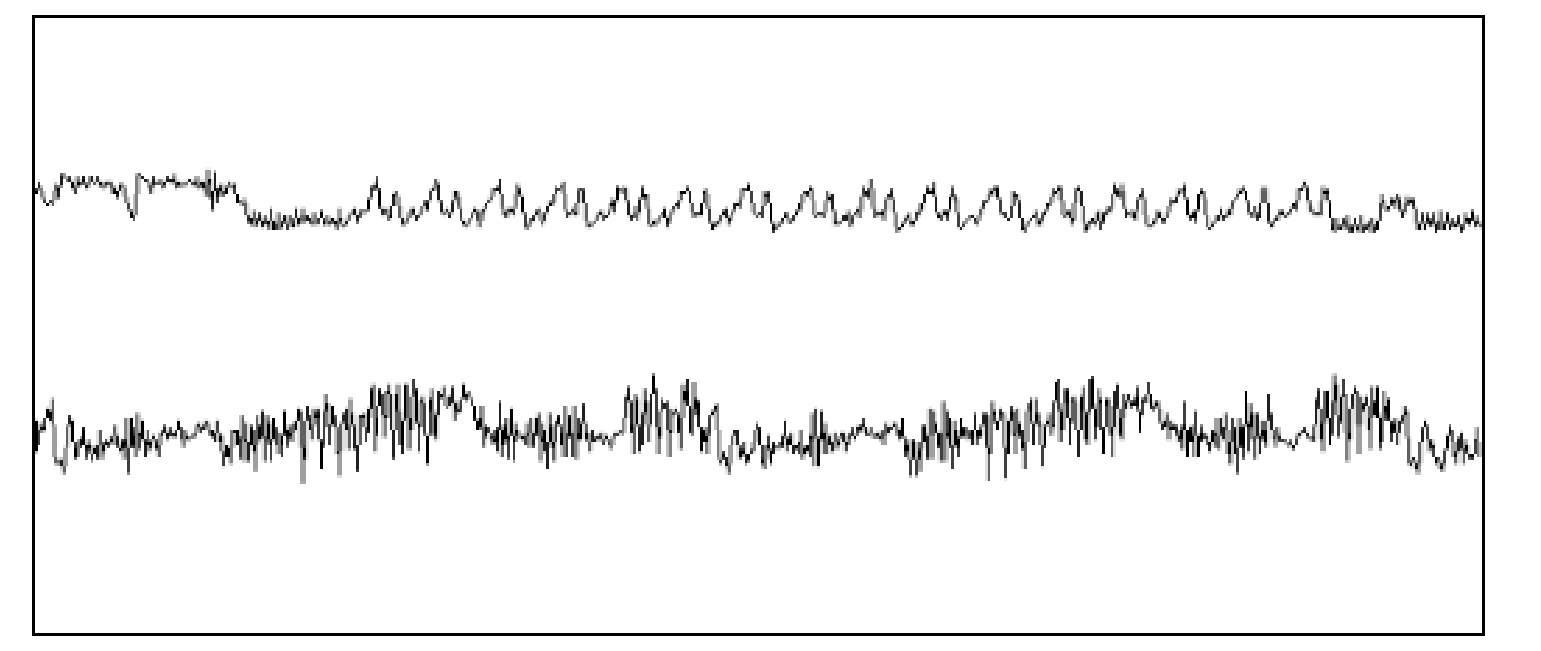


Рисунок 1.3 – SPA спостереження операції DES

Окремо ці криві зображені на рисунках 1.4 та 1.5.

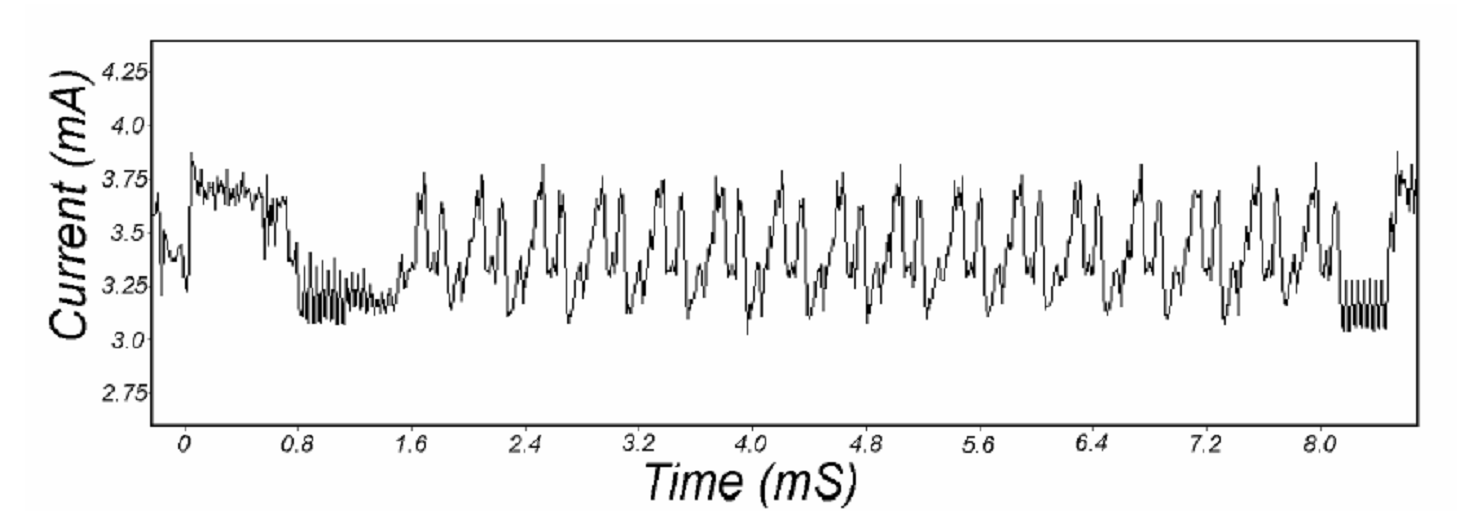


Рисунок 1.4 – Траса SPA усієї DES операції

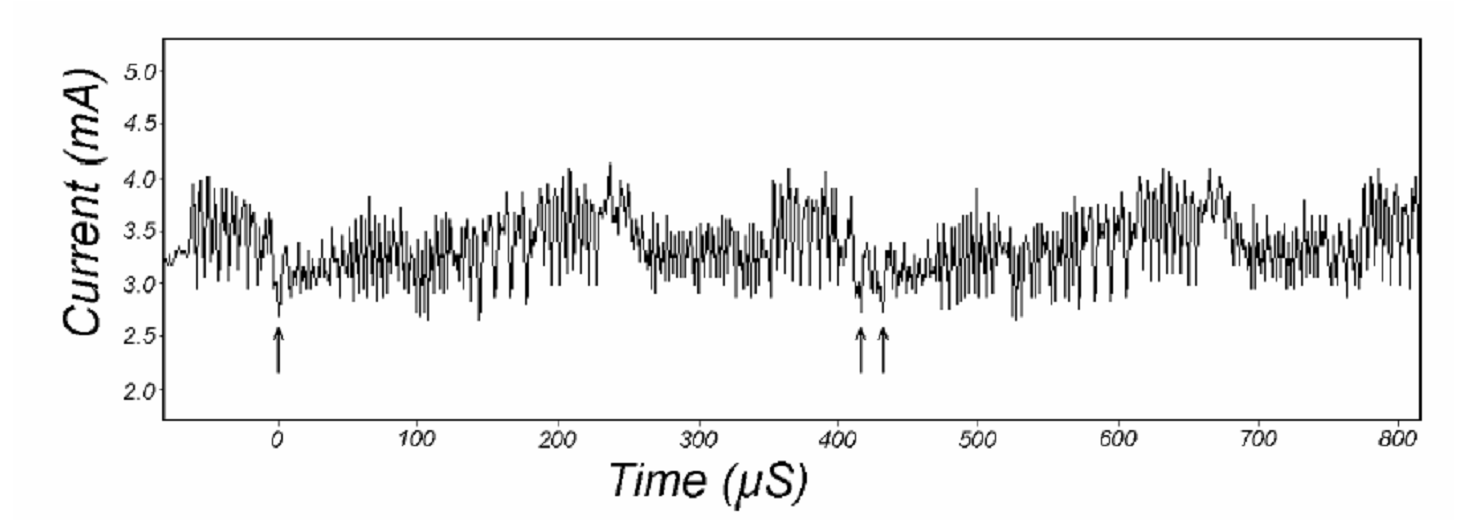


Рисунок 1.5 – Траса SPA 2го і 3го раундів DES операції

Диференційний аналіз енергоспоживання є могутнішою атакою, ніж SPA, і більш складний для запобігання. У той час, як SPA атаки головним чином будуються на візуальному аналізі з метою виділення значимих флуктуацій живлення, DPA атака використовує статистичний аналіз і техніку корекції помилок для виділення інформації, що має кореляції із секретними ключами. Для використання статичної техніки аналізуються результати 1000 трансакцій.

Реалізація DPA атак складається із двох етапів – нагромадження й аналізування даних. Нагромадження інформації для DPA може бути виконане, як описувалося раніше, відбором проб споживаної пристроєм напруги під час криптографічних операцій як функцією від часу. Для DPA безліч криптографічних операцій використовують шуканий ключ.

На рисунку 1.6 показано 4 криві, отримані при використанні введення відкритого тексту в DES функцію шифрування в смарт-карті. Угорі зображене середнє споживання енергії під час DES операцій. Унизу зображено три диференціальних кривих, де перша представляє використання правильного варіанту значення для ключа *Ks*. Нижні дві криві представляють використання некоректного значення для *Ks*. Ці залежності були отримані з використанням 1000 вимірів. Хоча сигнали чітко видно, у них присутня помірна кількість шуму.

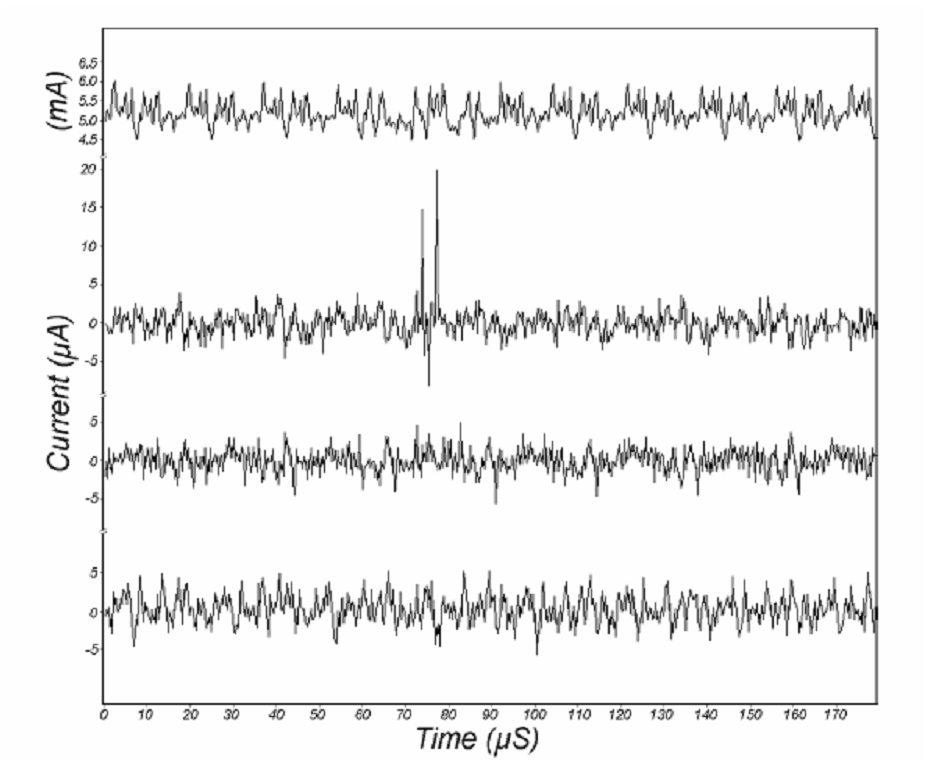


Рисунок 1.6 – Траси DPA

На рисунку 1.7 показаний середній результат одиничних біт при детальному вимірі споживання енергії. Угорі показана крива споживання енергії. У середині показане квадратичне відхилення у вимірах споживання енергії. Нарешті, остання крива являє собою диференціальну залежність, отриману з *m*=104. Існують області, які не корелюють із бітом ближче до нуля, ніж інші значення, показуючи тим самим, що невеликий шум або помилка зберігається.

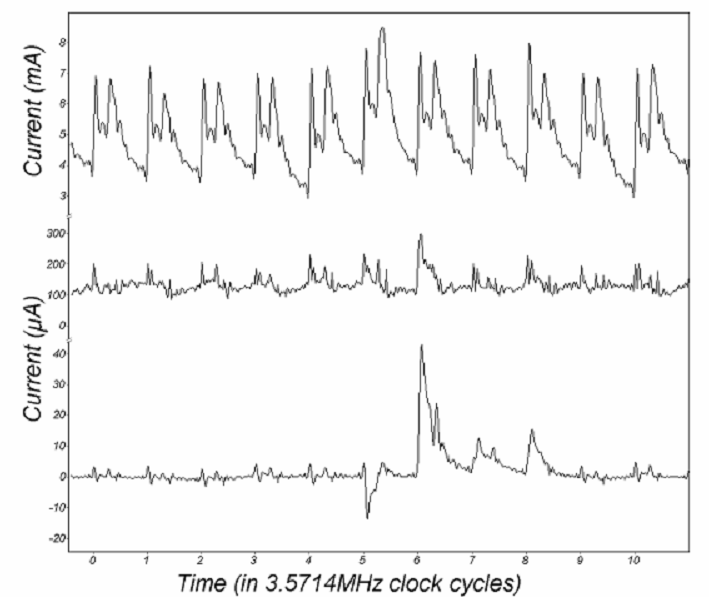


Рисунок 1.7 – Кількісні вимірювання DPA

У той час, як ефекти одного перемикаючого транзистора будуть нормальними, буде можливо ідентифікувати спостереження споживання енергії пристроєм. Статичні операції, що використовуються в DPA, здатні вірогідно визначити дуже малі відмінності між зміною живлення.

Рисунок 1.8 – це DPA крива звичайної смарт-карти, що показує зміну споживання енергії від обраного одного вхідного біта до DES функції шифрування, що використовуються як генератор випадкових чисел. Функція D була обрана рівною значенню 5-го біта нешифрованого тексту. Вхідне початкове перемішування поміщає цей біт як частину *R* регістру, що діє в першому раунді *F* функції обчислення й результатах. Раунд 2 діє (через використання режиму лічильника) ще сильніше. Вимір був представлений 1000 вимірами.

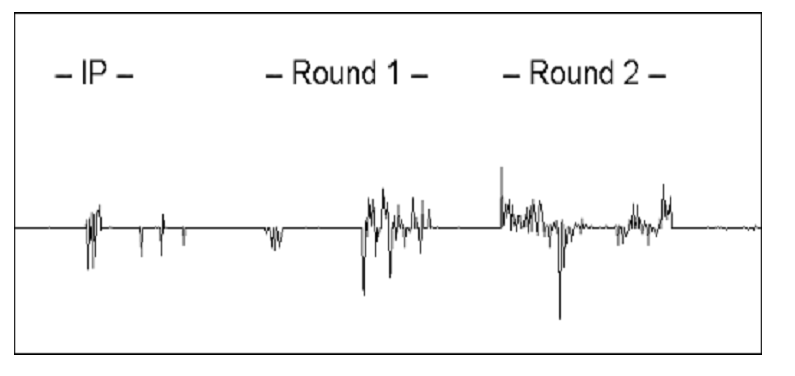


Рисунок 1.8 –DPA крива смарт-картки

DPA і схожі атаки охоплюють традиційні інженерні рівні абстракції. Поки більшість раніше відомих криптоаналітичних атак засновані на вивченні криптографічних алгоритмів, то DPA на уразливості транзисторів й режимів електричних ланцюгів, операцій мікропроцесора, і.т.д.

Ще більш складний метод аналізу – диференційний аналіз високого порядку (High-Order Differential Power Analysis – HO-DPA). У той час як DPA техніка аналізує інформацію між зразками даних протягом окремої події, метод аналізу диференціалів високого порядку можна використовувати для корелювання інформації між багатьма криптографічними субопераціями. В DPA атаках високого порядку сигнали збираються від багатьох джерел, для їхнього збору застосовуються різні техніки виміру, а сигнали з різними зсувами за часом комбінуються в процесі застосування методів DPA.

У високо-порядковій DPA атаці, сигнали збираються з багатьох джерел, використовуючи техніку різних вимірів, також збираються сигнали з різними часовими зсувами під час застосування техніки DPA. На додаток, можуть бути застосовані більшість диференційних функцій, а також багато сигнало-опрацьовуючих функцій. Основні HO-DPA обчислювальні функції є більш загальною формою стандартних DPA функцій, наприклад:

.

На сьогоднішній день HO-DPA становить великий інтерес для розроблювачів систем і дослідників, тому що не відомо реально існуючих систем, які вразливі до HO-DPA, але не вразливі до DPA. Однак HO-DPA уважається більш ефективною, ніж DPA.

1.3 Формування вимог та постановка задачі

Майже завжди існує небезпека зловмисного проникнення до даних, що опрацьовуються, зберігаються та передаються по каналах зв’язку. Це може привести до втрати чи спотворення важливої інформації, що може завдати значної шкоди. Тому, на сьогоднішній день тема криптографічного захисту даних є надзвичайно актуальною.

Основні напрями використання криптографічних методів є передача конфіденційної інформації по каналах зв’язку (наприклад, електронна пошта), встановлення достовірності переданих повідомлень, зберігання інформації (документів, баз даних) на носіях в зашифрованому вигляді.

Які би не були складні і надійні криптографічні системи - їх слабке місце при практичній реалізації - проблема розподілу ключів. Для того, щоб був можливий обмін конфіденційною інформацією між двома суб’єктами інформаційної системи, ключ повинен генеруватися одним з них, а потім якимсь чином знову ж таки в конфіденційному порядку передаватися іншому. Тобто в загальному випадку для передачі ключа також потрібне використання якоїсь криптосистеми.

Для вирішення цієї проблеми на основі результатів, отриманих класичною і сучасною алгеброю, були запропоновані системи з відкритим ключем. Криптосистема RSA є одною із найуспішніших криптосистем такого виду[7].

На даний час криптосистема RSA є найбільш широко використовуваною асиметричною криптосистемою.

Криптосистема RSA використовується в різноманітних програмних продуктах, на різних платформах і в багатьох галузях. На даний час криптосистема RSA вбудовується в багато комерційних продуктів, число яких постійно збільшується. Також її використовують операційні системи Microsoft, Apple, Sun і Novell. Щодо апаратного використання алгоритм RSA застосовується в захищених телефонних лініях, у мережевих платах Ethernet, у смарт-картах, а також у криптографічному устаткуванні THALES (Racal), що використовується у платіжних системах Visa, Europay та mastercard. Крім того, алгоритм входить до складу всіх основних протоколів для захищених комунікацій Internet, зокрема S/MIME, SSL і S/WAN, а також використовується в багатьох установах, наприклад, в урядових службах, в більшості корпорацій, в державних лабораторіях і університетах. На осінь 2005 року технології із застосуванням алгоритму RSA ліцензіювали більш ніж 900 компаній [7].

Незважаючи на всесвітнє поширення RSA, практична реалізація криптосистеми зв’язана з певними проблемами. Перша і, найважливіша з них – обчислювальна складність алгоритму. В порівнянні з більшістю симетричних криптосистем RSA працює в декілька сот разів повільніше. До найбільш швидких з точки зору програмної реалізації належить bSafe 3.0, що здійснює шифрування із швидкістю 21.6 Kбит/сек для 512-бітового ключа. Це дуже швидко якщо взяти до уваги, що здійснюються операції над числами в 150 знаків, але практично це досить повільно. Тому звичай системи з відкритим ключем використовують у поєднанні з якою-небудь симетричною системою, наприклад, DES.

Ще одна складна проблема, пов’язана з реалізацією алгоритму, – це генерація простих чисел. Найпримітивніший варіант її вирішення – решето Ератосфена, але воно добре працює лише для чисел довжиною до 10 знаків. Для генерації придатного до використання ключа в 150 знаків цей метод не працює. Фірма RSA Data Security рекомендує використовувати імовірнісний підхід, тобто генерувати не насправді прості числа, а ті, для яких вірогідність того, що вони складені, менше наперед заданої малої величини.

Для захисту конфіденційної інформації, що передається по локальній чи глобальній мережі, необхідно застосовувати спеціальні програмні засоби.

На основі аналізу дослідження продуктивності сучасних методів модулярного експоненціювання можна зробити висновок, що найбільшу швидкодію та найбільший рівень продуктивності має бета-арний метод модулярного експоненціювання [1].

Також дослідивши залежність швидкодії кожного методу від ваги Хемінга; доведено, що при певних заданих умовах найвищу стійкість системи забезпечує алгоритм *β*-арного методу “зліва направо”, що дає змогу ґрунтовно обирати алгоритм та його параметри для реалізації засобів криптографічного захисту інформації, стійких до атак спеціальних впливів.

Отже, програмний продукт повинен являти собою сучасну систему захисту інформації, що виконує процедуру шифрування криптосистеми RSA на основі бета-арного методу модулярного експоненціювання “зліва направо”. Програма повинна мати високу крипостійкість. Криптостійкістю називається характеристика шифру, що визначає його стійкість до дешифрування без знання ключа (тобто криптоаналізу). Існує кілька показників криптостійкості, серед яких:

* + кількість усіх можливих ключів;
  + середній час, необхідний для криптоаналізу.

Реалізована програма повинна задовольняти всі загальноприйняті вимоги криптографічних систем захисту інформації:

* + зашифроване повідомлення повинне піддаватися читанню тільки при наявності ключа;
  + число операцій, необхідних для визначення використаного ключа шифрування по фрагменті зашифрованого повідомлення і відповідного йому відкритого тексту, повинно бути не менше загального числа можливих ключів;
  + число операцій, необхідних для розшифрування інформації шляхом перебору всіляких ключів повинне мати строгу нижню оцінку і виходити за межі можливостей сучасних комп’ютерів (з урахуванням можливості використання мережевих обчислень).

Програму доцільно писати під операційну систему Windows, оскільки ця система є більш незахищеною порівняно із операційними системами Unix і більшість корчувачів надають перевагу саме операційній їй через її зручний користувальний інтерфейс.

Інтерфейс програми повинен бути зручним як для професійних так і не для професійних користувачів.

2 АСИМЕТРИЧНИЙ АЛГОРИТМ ШИФРУВАННЯ RSA, СТІЙКИЙ ДО АТАКИ АНАЛІЗУ ЕНЕРГОСПОЖИВАННЯ

2.1 Асиметричні криптоалгоритми

Метою шифрування є перетворення вихідного тексту *Т* з використанням ключа *К* в зашифрований текст *t*. Симетричні криптосистеми для шифрування та дешифрування використовують той самий ключ *К*. Системи з відкритим ключем (асиметричні криптосистеми), здійснюють шифрування за допомогою загальнодоступного ключа, для дешифрування в цьому випадку необхідний секретний ключ, який породжується разом з відкритим. Як шифрування, так і дешифрування може реалізуватися програмно або апаратно. При цьому повинні виконуватися певні вимоги:

* знання використаного алгоритму не повинне знижувати надійність шифрування;
* довжина зашифрованого тексту повинна дорівнювати довжині вихідного відкритого тексту (ця вимога належить до бажаних і виконується не завжди);
* зашифрований текст не може бути прочитаний без знання ключа;
* кожний ключ із різноманіття ключів повинен забезпечувати достатню надійність;
* зміна довжини ключа не повинна приводити до зміни алгоритму шифрування;
* якщо відомий зашифрований і відкритий текст повідомлення, то число операцій, необхідних для визначення ключа, не повинне бути менше повного числа можливих ключів;
* дешифрування шляхом перебору всіх можливих ключів повинне виходити далеко за межі можливостей сучасних ЕОМ;
* якщо при шифруванні в текст уводяться додаткові біти, то алгоритм їх внесення повинен бути надійно схований;
* не повинно бути легко встановлюваної залежності між послідовно використовуваними ключами;
* алгоритм може бути реалізований апаратно.

У симетричних криптосистемах можуть використовуватися одно- або багатоалфавітні підстановки (наприклад, одно-алфавітна підстановка Цезаря), при цьому проводиться заміна символів вихідного тексту на інші з використанням досить складних алгоритмів. Багатоалфавітні підстановки є більш надійними. До простих методів шифрування відноситься спосіб перестановок символів вихідного тексту (цей метод ефективний тільки лише при досить великій довжині вихідного тексту). Безліч перестановок символів для тексту з *N* символів рівно *N*!, що до якогось ступеня гарантує надійність процедури. Трохи більшу надійність пропонує метод гамування, коли на вихідний текст накладається псевдовипадкова послідовність біт, генерована на основі ключа шифрування, наприклад, з використанням операції “виключаюче АБО”. Зворотне перетворення (дешифрування) виконується генерацією точно такої ж псевдовипадкової послідовності й накладенням її на зашифрований текст. Гамування вразливе для випадку, коли зловмисникові стає відомий фрагмент вихідного тексту. У цих обставинах він досить просто відновить фрагмент псевдовипадкової послідовності, а по ньому й усю послідовність. Так якщо досить велика кількість повідомлень починається зі слів "Секретно", а наприкінці ставиться дата повідомлення, розшифрування стає питанням часу й терпіння.

Ключ може бути одноразового й багаторазового використання. Одноразовий ключ досить великої довжини (або нескінченний) може забезпечити дуже високу надійність, але його використання створює незручності, пов'язані з його транспортуванням (ключ повинен бути якось доставлений одержувачеві зашифрованого повідомлення).

Найбільше поширення останнім часом одержало блочне шифрування, де послідовність процедур впливає на блок вхідного тексту. Одним з найбільш відомих таких методів став DES (Data Encryption Standard), який працює із блоками даних по 64 байта (1998 рік). На рисунку 2.1 зображено загальну схему алгоритму шифрування DES.

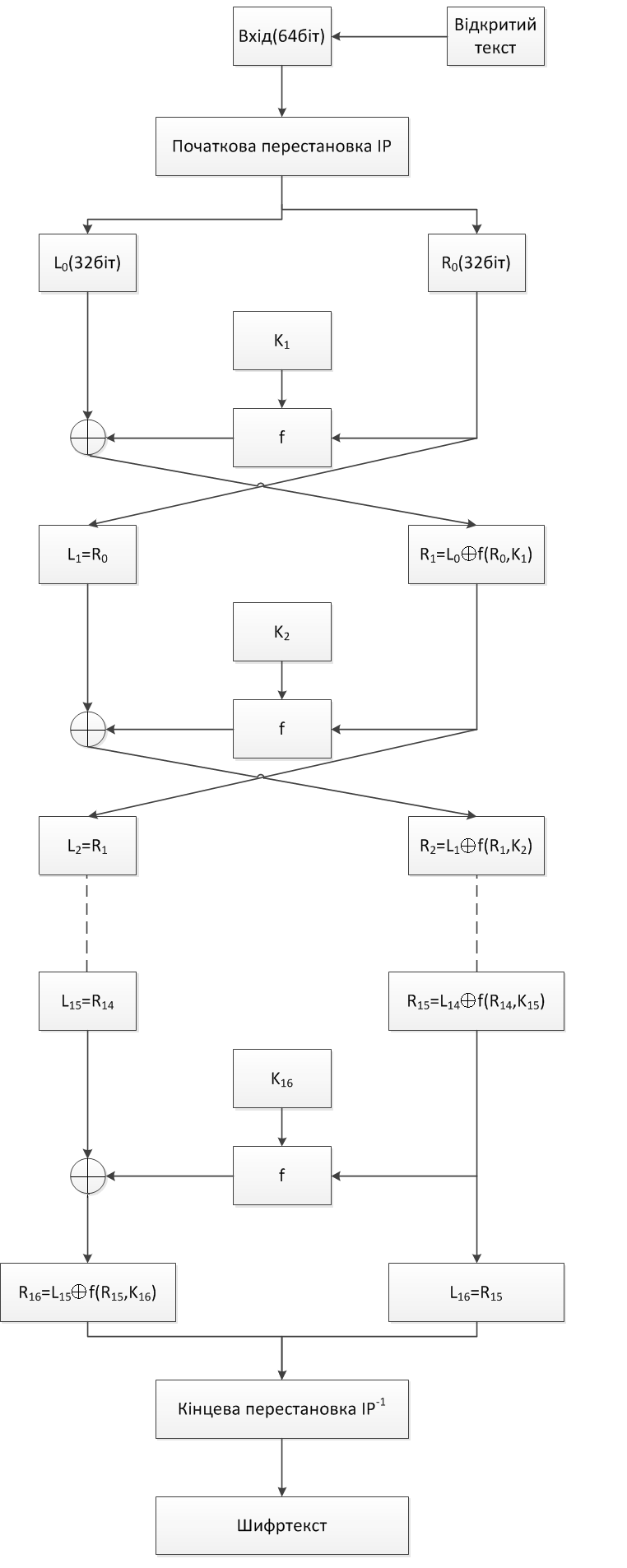


Рисунок 2.1 – Схема алгоритму DES

Існує чотири режими роботи:

* ECB electronic code book.
* CBC cipher block chaining
* CFB cipher feedback
* OFB output feedback.

Через те, що алгоритм DES у цей час представляється застарілим і не забезпечує достатньої надійності, досить часто вихідний текст послідовно шифрується тричі за допомогою різних ключів.

Ефективність системи шифрування визначається числом кодових комбінацій або ключів, яке необхідно перебрати, щоб прочитати зашифрований текст. Існує дві системи ключів шифрування/дешифрування. Для симетричних алгоритмів передбачається, що ключ дешифрування може бути обчислений по відомому ключу шифрування. Обидва ключа при цьому повинні бути секретними (наприклад, система DES). Відправник і одержувач повинні знати ключі до початку обміну (ці ключі можуть і збігатися). Набір таких ключів може бути досить великим і в процесі ініціалізації здійснюється вибір пари ключів, які будуть використані в даній сесії. У загальному випадку можуть використовуватися досить великі кодові таблиці, але така схема незручна для багатоточкового обміну.

Шифри бувають потоковими та блоковими. У першому випадку обробка вихідного тексту проводиться по-бітно або по-байтно. У другому – текст перед обробкою розбивається на блоки.

Асиметричні схеми шифрування/дешифрування припускають існування незалежних ключів для шифрування й дешифрування. Причому один не може бути отриманий з іншого й навпаки. В ідеалі ключ дешифрування не може бути отриманий із ключа, що шифрує, за будь-який розумний час. У цьому випадку ключ шифрування може бути зроблений загальнодоступним (наприклад, алгоритм RSA). Партнери до початку комунікацій повинні послати один одному ключі шифрування *КШВ* й *КШО* (ключі шифрування відправника й одержувача). Можливість перехоплення таких ключів небезпеки не представляє. Дешифрування виконується за допомогою ключів *КДВ* й *КДО*,які утворюють пари із *КШВ* й *КШО* , відповідно, й формуються спільно. Ключі *КШВ* й *КШО*звичайнопересилаються на фазі ініціалізації сесії інформаційного обміну.

Процес передачі повідомлення, з використанням асиметричних криптосистем зображено на рисунку 2.2.

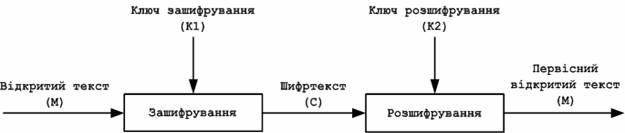


Рисунок 2.2 – Схема передачі повідомлення з використанням асиметричних криптосистем

Поняття криптосистеми з відкритим ключемвключає такі об’єкти:

* алфавіт , у якому записуються повідомлення (відкриті тексти), і алфавіт , у якому записуються криптотексти;
* простір ключів K (множина слів у деякому алфавіті);
* алгоритм генерування ключів. Це поліноміальний імовірнісний алгоритм, який на вході записаний в унарній системі як параметр надійності , де , видає випадкову пару значень .  називається відкритим ключем і використовується для шифрування, а  — таємним ключем і використовується для дешифрування;
* поліноміальний детермінований алгоритм шифрування , який отримує на вхід повідомлення  і відкритий ключ , а видає криптотекст , що записується як ;
* поліноміальний детермінований алгоритм дешифрування , який отримує на вхід криптотекст  і таємний ключ , а видає відкритий текст , що записується як .

Для з’ясування переваги асиметричної криптографічної схеми над симетричною криптосистемою (для шифрування і дешифрування повідомлення використовується один і той самий таємний ключ ) розглядається комунікаційна мережа, якою користуються  абонентів. Нехай кожна пара абонентів хоче мати можливість спілкуватися таємно від решти. Оскільки всіх пар є , то саме така кількість ключів необхідна при використанні симетричної схеми.

Асиметрична схема дозволяє влаштувати довірливе спілкування у мережі оптимальніше. Кожен абонент, власноручно або через менеджера мережі, генерує власну пару ключів . Перший абонент стає власником пари ключів , другий —  і т.д. Кожен абонент свій приватний ключ зберігає в таємниці, у той час як список всіх відкритих ключів  є загально доступним. Коли один абонент хоче відіслати іншому повідомлення , то він надсилає йому криптотекст . Оскільки тільки перший абонент знає таємний ключ , то лише він здатний дешифрувати . Таким чином, таємницю листування збережено з використанням лише *n* пар відкритого та таємного ключів, на противагу до необхідних раніше  ключів.

Отже асиметрична криптосистема визначається трьома алгоритмами: генерації ключів, шифрування й розшифрування. Алгоритм генерації ключів відкритий, усякий може подати йому на вхід випадковий рядок *r* належної довжини й одержати пару ключів (*k*1, *k*2). Один із ключів (наприклад, *k*1) публікується, він називається відкритим, а другий, секретний, зберігається в таємниці. Алгоритми шифрування  й розшифрування такі, що для будь-якого відкритого тексту *m* .

Варто розглянути тепер гіпотетичну атаку зловмисника на цю систему. Зловмисникові відомий відкритий ключ *k*1, але невідомий відповідний секретний ключ *k*2. Зловмисник перехопив криптограму *d* і намагається знайти повідомлення *m*, де *.* Оскільки алгоритм шифрування відкритий, супротивник може просто послідовно перебрати всі можливі повідомлення довжини *n*, обчислити для кожного такого повідомлення *mi* криптограму  й зрівняти *di* з *d.* Те повідомлення, для якого *di = d* і буде шуканим відкритим текстом. Якщо пощастить, то відкритий текст буде знайдений досить швидко. У гіршому ж випадку перебір буде виконаний за час порядку 2*nT*(*n*), де *T*(*n*) – час, необхідний для шифрування повідомлення довжини *п.* Якщо повідомлення мають довжину порядку 1000 біт, то такий перебір нездійсненний на практиці ні на яких самих потужних комп'ютерах.

Розглянутий лише один з можливих способів атаки на криптосистему й найпростіший алгоритм пошуку відкритого тексту, який називають алгоритмом повного перебору. Використовується також і інша назва: «метод грубої сили». Інший найпростіший алгоритм пошуку відкритого тексту – угадування. Цей очевидний алгоритм вимагає невеликих обчислень, але спрацьовує із дуже малою ймовірністю (при більших довжинах текстів). Насправді зловмисник може намагатися атакувати криптосистему різними способами й використовувати різні, більш витончені алгоритми пошуку відкритого тексту.

Крім того, зловмисник може спробувати відновити секретний ключ, використовуючи знання (у загальному випадку несекретні) про математичну залежність між відкритим і секретним ключами. Природно вважати криптосистему стійкою, якщо будь-який такий алгоритм вимагає практично нездійсненного обсягу обчислень або спрацьовує із дуже малою ймовірністю. При цьому зловмисник може використовувати не тільки детерміновані, але й імовірнісні алгоритми. Це і є один з підходів до визначення стійкості. Для його реалізації у відношенні того або іншого типу криптографічних систем необхідно виконати наступне:

1) дати формальне визначення системи даного типу;

2) дати формальне визначення стійкості системи;

3) довести стійкість конкретної конструкції системи даного типу.

Тут відразу ж виникає ряд проблем. По-перше, для застосування даного підходу необхідно побудувати математичну модель криптографічної системи, що залежить від деякого параметра, що називається параметром безпеки, який може приймати як завгодно великі значення (звичайно для простоти передбачається, що параметр безпеки може пробігати весь натуральний ряд).

По-друге, визначення стійкості криптографічної системи залежить від того завдання, яке стоїть перед зловмисником, і від того, яка інформація про схему йому доступна. Тому стійкість систем доводиться визначати й досліджувати окремо для кожного припущення про зловмисника.

По-третє, необхідно уточнити, який обсяг обчислень можна вважати «практично нездійсненним». Зі сказаного вище випливає, що ця величина не може бути просто константою, вона повинна бути представлена функцією від зростаючого параметра безпеки. Відповідно до тези Едмондса алгоритм уважається ефективним, якщо час його виконання обмежений деяким поліномом від довжини вхідного слова (у нашому випадку – від параметра безпеки). А якщо ні, то говорять, що обчислення по даному алгоритму практично нездійсненні. Відзначимо також, що самі криптографічні системи повинні бути ефективними, тобто всі обчислення, запропоновані тією, або іншою схемою, повинні виконуватися за поліноміальний час.

По-четверте, необхідно визначити, яку ймовірність можна вважати такою, якою можна знехтувати. У криптографії прийнято вважати такою будь-яку ймовірність, яка для будь-якого полінома *р* і для всіх досить великих *n* не перевершує 1/*р*(*n*), де *п* – параметр безпеки.

Отже, при наявності всіх зазначених вище визначень, проблема обґрунтування стійкості криптографічної системи звелася до доказу відсутності поліноміального алгоритму, який вирішує завдання, що стоїть перед зловмисником. Але тут виникає ще одна й досить серйозна перешкода: сучасний стан теорії складності обчислень не дозволяє доводити поліноміальні нижні оцінки складності для конкретних завдань розглянутого класу. Із цього випливає, що на даний момент стійкість криптографічних систем може бути встановлена лише із залученням яких-небудь недоведених припущень. Тому основний напрямок досліджень полягає в пошуку найбільш слабких достатніх умов (в ідеалі – необхідних і достатніх) для існування стійких систем кожного з типів.

В основному, розглядаються припущення двох типів – загальні і теоретико-числові, тобто припущення про складність конкретних теоретико-числових завдань. Усі ці припущення в літературі звичайно називаються криптографічними.

Центральним поняттям у теорії асиметричних криптографічних систем є поняття однобічної функції.

Неформально під однобічною функцією розуміється ефективно обчислювана функція, для обернення якої (тобто для пошуку хоча б одного значення аргументу за заданим значенням функції) не існує ефективних алгоритмів. Зворотна функція може й не існувати.

Під функцією буде розумітися сімейство відображень {*fn*}, де *fn*: S*n* ® S*m*, *m* = *m*(*n*). Для простоти припустимо, що *n* пробігає натуральний ряд, а відображення *fn* визначені всюди. Функція *f* називається чесною, якщо $ поліном *q*(*x*), такий що " *n* *q*(*m*(*n*)) і *n*.

Формальне поняття однобічної функції описується в такий спосіб: парна функція *f* називається однобічної, якщо

1. існує поліноміальний алгоритм, який для будь-якого *x* обчислює *f*(*x*);
2. для будь-якої поліноміальної імовірнісної машини Тьюрінга *A* виконане наступне. Нехай стрічка *x* обрана випадковим чином з множини S*n*. Тоді для будь-якого полінома *p* і всіх досить більших *n*

*P*{*f*(*A*(*f*(*x*))) = *f*(*x*)} < 1/*p*(*n*).

Друга умова якісна і означає, що будь-яка поліноміальна імовірнісна машина Тьюрінга *A* може по даному *y* знайти *x* з рівняння *f*(*x*) = *y* лише із дуже малою ймовірністю.

Вимогу чесності вилучити не можна. Оскільки довжина вхідного слова *f*(*x*) машини *A* рівна *m*, їй може не вистачити поліноміального від *m* часу просто на виписування стрічки *x*.

Існування однобічних функцій є необхідною умовою стійкості багатьох криптосистем.. Розглянемо функцію *f*, таку, що *f*(*r*) = *k*1. Вона може бути обчислена за допомогою алгоритму *G* за поліноміальний час. Покажемо, що якщо *f* – не однобічна функція, то криптосистема нестійка. Припустимо, що існує поліноміальний імовірнісний алгоритм *A*, що обертає *f* з імовірністю принаймні 1/*p*(*n*) для деякого полінома *p*. Зловмисник може подати на вхід алгоритму значення ключа *k*1 і одержати із зазначеною ймовірністю деяке значення *r'* із прообразу. Далі зловмисник подає *r'* на вхід алгоритму *G* і одержує пару ключів (*k*1, *k*2'). Хоча *k*2' не обов'язково збігається з *k*2, по визначенню криптосистеми  для будь-якого відкритого тексту *m*. Оскільки *k*2' знайдений з імовірністю принаймні 1/*p*(*n*), схема нестійка.

Функцією-пасткою називається однобічна функція, для якої зворотну функцію обчислити просто, якщо є деяка додаткова інформація, і складно, якщо така інформація відсутня.

У якості завдань, що приводять до однобічних функцій, можна привести наступні.

1. Розкладання числа на прості співмножники.

Обчислити добуток двох простих чисел дуже просто. Однак, для розв'язку зворотного завдання – розкладання заданого числа на прості співмножники, ефективного алгоритму в даний час не існує.

2. Дискретне логарифмування.

Припустимо, задане велике просте число *p* і нехай *g* – примітивний елемент поля *GF*(*p*). Тоді для будь-якого *a* обчислити *ga*(*mod* *p*) просто, а обчислити *a* по заданому *k* = *ga*(*mod* *p*) і *p* виявляється складно.

Криптосистеми з відкритим ключем ґрунтуються на однобічних функціях-пастках. При цьому відкритий ключ визначає конкретну реалізацію функції, а секретний ключ подає інформацію про пастку. Будь-хто, знаючий пастку, може легко обчислювати функцію в обох напрямках, але той, у кого така інформація відсутня, може робити обчислення тільки в одному напрямку. Прямий напрямок використовується для шифрування й для верифікації цифрових підписів, а зворотній – для розшифрування й вироблення цифрового підпису.

У всіх криптосистемах з відкритим ключем чим більша довжина ключа, тем вища відмінність між зусиллями, необхідними для обчислення функції в прямому й зворотному напрямках.

Усі практичні криптосистеми з відкритим ключем ґрунтуються на функціях, що вважаються однобічними, але ця властивість не була доведена у відношенні ні однієї з них. Це означає, що теоретично можливе створення алгоритму, що дозволяє легко обчислювати зворотну функцію без знання інформації про пастку. У цьому випадку, криптосистема, заснована на цій функції, стане неефективною. З іншого боку, теоретичні дослідження можуть привести до доведення існування конкретної нижньої границі складності обернення деякої функції, і це доведення буде істотною подією, яка виявить істотний позитивний вплив на розвиток криптографії.

Одним із відомих представників класу асиметричних алгоритмів можна назвати схему шифрування Полліга-Хелмана. Алгоритм, що тут використовується, схожий до RSA. Проте цю систему не можна віднести до криптосистем з відкритим ключем, оскільки обидва ключі (шифрування/розшифрування) легко можуть бути визначеними один з одного. Тому обидва ключі в цій системі є секретними.

Аналогічно до алгоритму RSA, криптограма *C* та відкритий текст *M* визначаються із співвідношень , .

В цій схемі модуль дискретного експоненціювання *N* не визначається на основі двох простих великих чисел. Тому він також повинен зберігатися в секреті.

Схема шифрування Ель-Гамаля була запропонована в 1985 році і може використовуватися як в режимі шифрування даних, так і в режимі електронного цифрового підпису. Надійність алгоритму базується на складності обчислення дискретних логарифмів.

Даний алгоритм відноситься до класу імовірнісних алгоритмів криптозахисту інформації. Особливість алгоритмів даного класу полягає в тому, що повідомленню *M* відповідає не один криптотекст *C*, а деяка множина криптотекстів, причому кожен член цієї множини вибирається з певною ймовірністю. Іншими словами, для кожного повідомлення *M* результат роботи алгоритму є випадковою величиною, розподіленою на множині .

2.2 Алгоритм шифрування даних RSA

Найпоширенішим на сьогодні асиметричним алгоритмом шифрування є RSA. Опис RSA було опубліковано у 1977 році Рональдом Райвестом (Ronald Linn Rivest), Аді Шаміром (Adi Shamir) і Леонардом Адлеманом (Leonard Adleman) з Масачусетського Технологічного Інституту (MIT). На рисунку 2.3 зображено схему алгоритму RSA.

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



Рисунок 2.3 – Схема шифрування і розшифрування RSA

Для того, щоб згенерувати пари ключів виконуються наступні дії:

1. вибираються два великих простих числа *p* і *q*;
2. обчислюється їх добуток ;
3. обчислюється функція Ейлера ;
4. вибирається ціле *e* таке, що  та *e* взаємно просте з ;
5. за допомогою розширеного алгоритма Евкліда знаходиться число *d* таке, що .

Число *n* називається модулем, а числа *e* і *d* — відкритою й секретною експонентами, відповідно. Пари чисел (*n, e*) є відкритою частиною ключа, а (*n, d*) — секретною. Числа *p* і *q* після генерації пари ключів можуть бути знищені, але в жодному разі не повинні бути розкриті.

Для того, щоб зашифрувати повідомлення *m<n* обчислюється 

Число *c* використовується в якості шифртексту. Для розшифрування потрібно обчислити 

Неважко переконатися, що при розшифруванні відновлюється вихідне повідомлення 

З умови  виходить, що  для деякого цілого *k*, отже 

Згідно теореми Ейлера  тому  

Криптосистема RSA використовується в різноманітних програмних продуктах, на різних платформах і в багатьох галузях. Також її використовують операційні системи Microsoft, Apple, Sun і Novell. Щодо апаратного використання, алгоритм RSA застосовується в захищених телефонних лініях, у мережевих платах Ethernet, у смарт-картах, а також у криптографічному устаткуванні THALES (Racal), що використовується у платіжних системах Visa, Europay та mastercard. Крім того, алгоритм входить до складу всіх основних протоколів для захищених комунікацій Internet, зокрема S/MIME, SSL і S/WAN, а також використовується в багатьох установах, наприклад, в урядових службах, в більшості корпорацій, в державних лабораторіях і університетах.

Незважаючи на всесвітнє поширення RSA, практична реалізація криптосистеми зв’язана з певними проблемами. Перша і, найважливіша з них – обчислювальна складність алгоритму. В порівнянні з більшістю симетричних криптосистем, RSA працює в декілька сот разів повільніше. До найбільш швидких з точки зору програмної реалізації належить bSafe 3.0, що здійснює шифрування із швидкістю 21.6 Kбит/сек для 512-бітового ключа. Це дуже швидко, якщо взяти до уваги, що здійснюються операції над числами в 150 знаків, але практично це досить повільно. Тому зазвичай системи з відкритим ключем використовують у поєднанні з якою-небудь симетричною системою, наприклад, DES.

Ще одна складна проблема, пов’язана з реалізацією алгоритму, – це генерація простих чисел. Найпримітивніший варіант її вирішення – решето Ератосфена, але воно добре працює лише для чисел довжиною до 10 знаків. Для генерації придатного до використання ключа в 150 знаків цей метод не працює. Фірма RSA Data Security рекомендує використовувати імовірнісний підхід, тобто генерувати не насправді прості числа, а ті, для яких вірогідність того, що вони складені, менше наперед заданої малої величини.

2.3 Забезпечення стійкості алгоритму RSA до атаки енергоспоживання

Атаки аналізу енергоспоживання зазвичай діляться на два типи:

* прямий аналіз енергоспоживання (SPA), що базується на прямій інтерпретації сигналів енергоспоживання;
* диференційний аналіз енергоспоживання (DPA) комбінує напад SPA з помилково-коректуючою технікою, використовуючи статистичний аналіз.

Прямий аналіз енергоспоживання (SPA)складається зі спостереження енергоспоживання протягом єдиного виконання криптографічного алгоритму.

Класичний диференційний аналіз енергоспоживання (DPA) заснований на ідеї атаки SPA, але використовує методи виправлення помилок і статистичний аналіз для виявлення невеликих різниць сигналів енергоспоживання. Щоб бути стійким до нападу DPA, деякі системні параметри або процедури обчислення повинні бути змішані.

Існує також подвоююча атака (DA). Вона працює на основі секретного скалярного додавання та двійкового еліптичного скалярного множення. Вона працює на використанні двійкового методу. Головна ідея цього нападу базована на факті, що навіть, якщо зловмисник не може побачити обчислення, він може визначити коли та ж дія робиться двічі. Точніше, якщо смарт-карта рахує 2*A* і 2*B* в будь-яких діях, зловмисник не може вгадати значення *А* або *B*, але він може перевірити *А*=*B* чи *А*≠*B*. Це припущення зазвичай забирає дуже багато часу, бо в ньому багато циклів і вони залежать від значення змінних. Якщо шум незначний, то порівняння двох енергетичних трас протягом подвоєння буде ефективне для визначення цієї рівності.

Чиста атака аналізу енергоспоживання (RPA) припускає, що нападаючий може ввести адаптивні повідомлення. Аналіз RPA атак понизив його ефективність у великому ряді порядку чисел. Проте, напад RPA все ще становить загрозу еліптичним криптосистемам.

Нульове значення атаки (ZPA) − це продовження атаки RPA.

Розглянемо можливі заходи підвищення стійкості до атак побічних каналів:

1) маскування (Blinding). Методи, що використовуються для підпису «наосліп», можуть бути пристосовані для захисту оброблюваних даних. Це повинно захистити практично від усіх видів атак по зовнішньому каналу;

2) обчислення, що не залежать від даних. В ідеалі час виконання операцій, виконуваних модулем, не повинен залежати від оброблюваних даних. Інакше кажучи, час, який потрібен на виконання операції, повинен бути повністю незалежним від вхідних даних і значення ключа. Якщо залежно від значення бітів входу або ключа виконуються різні підоперації, то ці підоперації повинні виконуватися за однакову кількість тактів. Можливість зробити час виконання операції постійним для будь-яких даних запобігає всім часовим атакам;

3) умовні переходи. Відмова від процедур, які використовують для умовних переходів секретні проміжні значення або ключі, замаскує багато характеристик атак по зовнішньому каналу. В ідеалі програмна реалізація коду не повинна містити операторів розгалуження. Аналогічно, вони не повинні містити оператори умовного виконання, такі як IF. Обчислення повинні виконуватися через функції, що використовують елементарні операції (такі як AND, OR і XOR), і що не використовують розгалуження й умовне виконання ділянок коду;

4) додавання затримок. Найбільш очевидний шлях запобігти часовим атакам – зробити так, щоб усі операції забирали однаковий час. На жаль, це не завжди можливо. Крім того реалізації з фіксованим часом, імовірно, будуть повільними; багато чого з оптимізації не може застосовуватися, тому що всі операції повинні займати стільки часу, скільки потрібно для самої повільної з них;

5) вирівнювання часу множення й зведення у квадрат. Час, що затрачається пристроєм на множення й на піднесення до степеня повинен бути однаковим. Завдяки цій властивості зловмисник не зможе довідатися коли й скільки було виконано множень і піднесень до степеня. Вирівнювання часу може бути досягнуто шляхом виконання обох операцій незалежно від того, яка з них необхідна в даний момент. На будь-якому етапі, якщо потрібна була одна операція, то повинні бути виконано обидві, а результат однієї з них зігнорований;

6) балансування споживаної потужності. По можливості повинні застосовуватися методи балансування споживаної потужності. Слід додати не використовувані (з точки зору алгоритму) регістри й вентилі, на яких виконуються пошуки операції для того, щоб зробити рівень споживаної енергії постійним значенням. Такі методи, за допомогою яких енергоспоживання (якщо дивитися ззовні) залишається постійним і не залежить від біт входу й ключа, запобігають усім видам атак по каналу енергоспоживання;

7) додавання шуму. Засоби захисту від ЕМ атак по специфічності реалізації розділяються на дві широкі категорії: зменшення потужності сигналу й зменшення інформативності сигналу. Другий підхід проти DPA включає внесення шумів у виміри споживаної потужності. Одне із запропонованих рішень в [15] проти DPA з використанням шуму полягає в додаванні випадкових обчислень, які збільшують рівень шуму настільки, що стає неможливим визначити зсуви сплесків DPA;

8) екранування. На практиці сильне фізичне екранування, як зазначено в [4], може зробити атаки нездійсненними, але при цьому суттєво впливає на розміри й вартість пристрою;

9) виконання шифрування двічі. Можливе рішення проти атак по помилках обчислень (представлене в [7]) полягає в прогоні шифрування двічі й видачі результату, тільки якщо вони збігаються.

Розглянемо алгоритм RSA, стійкий до DPA.

На вхід подаються значення змінних *d*, *m* i *N*:

* *d* - відкритий ключ;
* *m* - повідомлення для шифрування;
* *N* – добуток *p* i *q.*

Кроки алгоритму:

1. Обирається довільне *r* взаємнопросте з *N*.
2. , , .
3. Для *і* від  до  виконати:

3.1 

3.2 

4. .

Для отримання  відкритий ключ *d* переводиться в двійкову форму, а *n* – кількість двійкових розрядів *d.*

Таким чином, внесення випадкової величини в процес шифрування дозволяє зробити алгоритм невразливим до атаки енергоспоживання (рисунок 2.4).



Рисунок 2.4 ‒ Алгоритм процесу шифрування криптоалгоритму RSA? Стійкого до атаки аналізу енергоспоживання

Запропонований підхід в асиметричних криптоалгоритмах дозволяє пришвидшити виконання криптографічних операцій за рахунок обходу виконання деяких операцій алгоритму при нульових бітах ключа, що дає змогу значно підвищити продуктивність засобів захисту інформації.

3 ТЕХНІЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСОБУ ПРОЦЕДУРИ ШИФРУВАННЯ КРИПТОСИСТЕМИ RSA НА ОСНОВІ БЕТА-АРНОГО МЕТОДУ МОДУЛЯРНОГО ЕКСПОНЕНЦІЮВАННЯ

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.

Одною зі найголовніших проблем асиметричних алгоритмів є низька швидкодія виконання основних математичних процедур, а саме операції множення за модулем та піднесення до степеня за модулем.

Виконання цих операцій вимагає дуже багато часу та великого об’єму пам’яті. Сучасні алгоритми модулярного експоненціювання суттєво прискорюють процес шифрування та дешифрування. Тому проаналізувавши дослідження зроблені у попередніх розділах, доцільно спроектувати програмний засіб, що використовує один із методів модулярного експоненціювання.

Отже, реалізовано програму яка виконує процедуру шифрування даних, використовуючи бета-арний метод модулярного експоненціювання на основі асиметричного алгоритму шифрування RSA. 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, відкритий ключ переводить в бета-арну систему, після чого шифрує вибраний користувачем файл. Зашифрований файл можна розшифрувати, вказавши місце знаходження файлу. Текстовий файл, що шифрувався і файл, що був розшифрований можна порівняти на схожість, тобто перевірити правильність роботи методу шифрування.

Програма написана для використання в операційних системах типу Windows NT. Цю систему відрізняє швидкість, надійність і безпека. Швидкість NT обумовлена двома причинами: NT є повністю 32-розрядною операційною системою і підтримує симетричну обробку. Симетрична обробка означає, що NT може працювати на комп'ютерах з декількома мікропроцесорами. Існує декілька версій Windows NT. NT Workstation може працювати на комп'ютерах з двома мікропроцесорами. NT Server використовує до 32 мікропроцесорів.

Другою відмінною рисою NT є вбудована система безпеки. На відміну від інших операційних систем, NT контролює допуск до комп'ютера. По-перше, NT не дасть людині скористатися комп'ютером, якщо він не введе правильне ім'я і пароль. По-друге, NT не дозволить користувачеві (людині, яка ввела ім'я і пароль) працювати з файлом, якщо він не входить в список тих, кому це дозволено.

На комп'ютерах з MS-DOS і старими версіями Windows у разі ненормальної роботи однієї програми решта програм, що виконувалися в цей час, теж постраждає. На відміну від цього, якщо Windows NT помічає, що одна програма дає збій, то вона не дасть завдати збитку іншим програмам або даним. Висока надійність роботи досягається за рахунок всеосяжного контролю, що приводить до уповільнення роботи комп'ютера і вимагає швидкодіючого процесора і великого об'єму пам'яті.

Для роботи Windows NT необхідно не меншого 32 Мбайт ОЗУ. NT може працювати з обмеженим набором периферійних пристроїв. У NT є п'ять засобів забезпечення безпеки: облікові записи, права дозволу, групи і журнали аудиту. Облікові записи реєструють користувача. Права рівнозначні особливим привілеям. Дозволи використовуються для управління доступом до різних об'єктів. Групи спрощують роботу адміністратора. Журнали аудиту відстежують певні дії і події. Всіма цими засобами управляє "Диспетчер користувачів".

Крім того, необхідний програмний засіб електронного захисту інформації, розроблений на базі Microsoft Visual C/C++. Обґрунтованість того, що програма повинна бути написана на мові С++ пов’язана з тим, що, перш за все, С++ − це мова програмування загального призначення. Його первісна сфера застосування − системні програми. В даний час С++ успішно використовується в багатьох прикладних областях, що не попадають у категорію системних. Реалізації С++ існують як на мікрокомп'ютерах, так і наймогутніших суперкомп'ютерах. С++ використовується сотнями тисяч програмістів практично у всіх прикладних областях.

Visual C++ − це компілятор C++, але це також і середовище, компоненти якого, взаємодіючи один з одним, спрощують процес розробки програм. Середовище Visual C++ пропонує великі можливості для програмування Windows-програм. Найхарактернішою його компонентою є бібліотека основних класів Microsoft (Microsoft Foundation Classes – MFS). Великий набір класів С++ інкапсулює основну частину API (Application Standart Interface) Win32 і пропонує могутню основу для написання типових програм. До складу Visual C++ входить Microsoft Developer Studio Integrated Development Environment – інтегроване середовище для розробки програм (IDE). Visual Studio 97 – остання версія Microsoft Developer Studio – є ядром системи розробки Visual C++. Вона пропонує багато різних можливостей, надає доступ до багатьох компонент системи розроблювача Visual C++, а також взаємодіє з такими засобами розробки Micrisoft, як Visual J++ або Microsoft Network Library. Visual Studio підтримує інтерфейс для компонентів проектів, початкових файлів і файлів ресурсів, а також ряд інструментів для побудови, запуску і тестування програм, хоча компілятор C/C++ та інші компоненти середовища розробника можна використовувати і з командного рядка. Visual Studio пропонує інтегрований інтерфейс для таких основних інструментів Visual C++ як майстри. Майстри (Wizards), такі як AppWizard, − це інструменти генерації структур програм. За допомогою таких майстрів можна створювати структури програм серверів або контейнерів компонентів OLE, програм ODBC та DAO, серверів автоматизації, елементів управління ActiveX.

Програмний продукт оформлено у вигляді виконавчих модулів та бібліотек підпрограм, які можуть використовуватися при розробці перспективних проектів. У комплект системи захисту інформації входять такі засоби:

– динамічні DLL бібліотеки функцій системи для розробки проектів на платформі WINDOWS;

– статичні LIB бібліотеки підпрограм системи, звертання до яких допускається із DOS аплікацій;

– PLB бібліотеки для виконання функцій системи в середовищі MS FoxPro 2.6 for DOS;

– ERR файли зі списками помилок та попереджень, які можуть виникнути в процесі функціонування системи.

Система надає доступ до засобів генерації ключів з метою дослідження та подальшого вдосконалення системи.

Програма реалізовувалася з використанням об’єктно-орієнтованого підходу.

Коли розмір програми великий, список команд стає дуже громіздким. Дуже багато програмістів здатні утримувати в голові більше 500 рядків програмного коду, якщо цей код не розділений на дрібніші логічні частини. Функція є засобом, що полегшує сприйняття при читанні тексту програми (термін функція уживається в мовах C і C++; у інших мовах програмування це ж поняття називають підпрограмою або процедурою). Програма, побудована на основі процедурного методу, розділена на функції, кожна з яких в ідеальному випадку виконує деяку закінчену послідовність дій і має явно виражені зв'язки з іншими функціями програми.

Можна розвинути ідею розбиття програми на функції, об'єднавши декілька функцій в модуль (часто модуль є окремим файлом). При цьому зберігається процедурний принцип: програма ділиться на декілька компонентів, кожний з яких є набором інструкцій.

Ділення програми на функції і модулі є основою структурного програмування. Структурне програмування є не цілком визначеним, проте протягом декількох десятків років, поки не була розроблена концепція об'єктно-орієнтованого програмування, воно залишалося важливим способом організації програм [20].

Основоположною ідеєю об'єктно-орієнтованого підходу є об'єднання даних і дій, виконуваних над цими даними, в єдине ціле, яке називається об'єктом. Функції об'єкту, звані в C++ методами або функціями-членами, зазвичай призначені для доступу до даних об'єкту. Якщо необхідно рахувати які-небудь дані об'єкту, то потрібно викликати відповідний метод, який виконає читання і поверне необхідне значення. Прямий доступ до даних неможливий. Дані приховані від зовнішньої дії, що захищає їх від випадкової зміни. Говорять, що дані і методи інкапсульовані. Терміни заховання і інкапсуляція даних є ключовими в описі об'єктно-орієнтованих мов.

Якщо необхідно змінити дані об'єкту, то, очевидно, ця дія також буде покладена на методи об'єкту. Ніякі інші функції не можуть змінювати дані класу. Такий підхід полегшує написання, відлагодження і використання програми.

Ядром програми можна назвати клас Cl\_Dlg, оскільки він виконує найголовніші операції, пов’язані із генеруванням криптографічних ключів. Клас містить також методи класу, що забезпечують діалог користувача, шифрування та дешифрування даних, а також генерацію простих чисел.

В програмі використовуються Microsoft Foundation Classes (MFC) –бібліотека базових класів Microsoft. Це бібліотека класів, що підтримує розробку програм для Microsoft Windows. Вона поставляється з Visual C++ і іншими компіляторами.

Код програми розбитий на декілька модулів, що значно полегшує розуміння коду програми а також дозволяє зручно тестувати програму та вдосконалювати шляхом додавання нових модулів. Серед найголовніших модулів програми можна виділити наступні:

* Файл rsa.cpp є одним із найголовніших модулів. Він відповідає за процедуру генерації відкритого та таємного ключів, а також переведення відкритого ключа криптосистеми RSA в бета-арну систему числення, що дозволяє використовувати бета-арний метод модулярного експоненціювання.
* GenPrimeNo.cpp. У файлі реалізовано процедуру генерування простих чисел що використовуються для генерування ключів та обчислення функції Ейлера.
* Dlg.h – хеадер-файл в якому реалізовується об’явлення класу Cl\_Dlg.
* Dlg.cpp – опис структури даних, які використовуються для представлення об'єкту і код методів доступу до них. Реалізація класу Cl\_Dlg, що забезпечує правильне функціонування системи та забезпечує діалог користувача.
* Encrypt.cpp – модуль реалізує процедуру шифрування використовуючи бета-арний метод модулярного експоненціювання.
* Decrypt.cpp – Файл реалізації дешифрування.
* rsa.cpp – файл виконує найголовніші операції пов’язані із генеруванням криптографічних ключів.
* Stdafx.h – хедер-файл, що забезпечує правильну роботу та правильну компіляцію програми в середовищі Visual C++. Підключає MFC бібліотеки.

Функціональна структура програми у вигляді схеми взаємозв’язку програмних модулів та файлів зображена в додатку Б.

Саме такий поділ на модулі доцільно може бути використаний для надання функціям більшої ясності, можливості повторного використання, що, таким чином, допомагає скоротити витрати, пов'язані з їх реалізацією та супроводом. Для прикладу розглянемо файл GenPrimeNo.cpp, у якому реалізовано процедуру генерування простих чисел (додаток В), використовуючи імовірнісний тест Соловея-Штрасена. Оскільки існує ще безліч підходів для генерування простих чисел, щоб дослідити ефективність інших генераторів, можна замінити цей файл файлом з реалізацією іншого алгоритму (наприклад, алгоритмом Міллера-Рабіна) не чіпаючи інших файлів. Таким чином, можна робити дослідження над програмою і покращувати її, не змінюючи основного коду.

3.2 Реалізація програмного забезпечення

Основою реалізації програмного забезпечення є реалізація класу Cl\_Dlg, який містить методи, що виконують основні операції шифрування даних. Методи класу – це функції які призначенні для доступу до даних об’єкту. Якщо потрібно буде зчитати які-небудь дані з об’єкту, то необхідно викликати відповідний метод, який виконає зчитування і поверне потрібне значення.

Визначення класу проводиться у файлі Dlg.h. Після об’явлення класу проводиться ініціалізація його змінних. Якщо необхідно захистити які-небудь дані, то їх поміщають всередину класу з ключовим словом private. Мається на увазі, що дані поміщені усередині класу і захищені від несанкціонованого доступу функцій, розташованих поза класом. До даних класу типу private відносяться змінні : f\_Eiler – значення функції Ейлера; m\_Prime1, m\_Prime2 –прості числа вибрані випадковим чином; m\_n – значення модуля; m\_e, m\_d – значення відкритого ї закритого ключів відповідно; m\_key\_gen\_done\_flg – значення булевого типу, яке показує чи відбулася генерація ключів, чи ні; m\_inFPName\_to\_encrypt, m\_inFPName\_to\_decrypt – імена файлів для шифрування і розшифрування. Після ініціалізації змінних слідує ініціалізація методів, тобто функцій класу.

Слід зауважити, що функції лише ініціалізуються в межах об’явлення класу, тобто усередині визначення класу міститься лише прототип функцій. Така форма означає, що функція є методом класу, проте її визначення слід шукати не усередині визначення класу, а десь у іншому місці або навіть у іншому файлі.

До захищених функцій класу Cl\_Dlg відносяться :

* CheckKeyGenStatus() – булеава функція. яка перевіряє чи відбулося генерування ключів;
* GeneratePrimeNumbers() – функція, яка здійснює генерацію простих чисел;
* DumpNotes() – функція, яка виводить процес шифрування та значення змінних у вікно програми в розділі „Процес шифрування”;
* WriteValues() – функція, яка виводить значення змінних у вікно програми в розділі „Деталі генерування ключів”;
* SelectE() – функція, яка генерує значення відкритого ключа;
* IsRelativePrime() – функція, що визначає чи функція Ейлера і відкритий ключ взаємно прості;
* Execute() – функція, що керує процесом генерування ключів;
* CalculateKsekr () – функція, що обчислює значення закритого ключа;
* Encrypt(), Decrypt() – функції для шифрування і розшифрування файлів;
* Convert\_to\_Sys() – функція переводить значення ключа у відповідну систему числення;
* ChangeFileExtension() – функція змінює розширення файлу.

В класі Cl\_dlg реалізовано лише одну функцію, яка може реалізовуватись за межами класу – це функція-конструктор. Конструктор – це метод класу, що виконується автоматично у момент створення об'єкту [20].

До захищених методів класу Cl\_Dlg відносяться функції, які в основному обробляють події від користувача, тобто реагують на дії, що виконує користувач, і передають управління на виконання певної функції типу private:

* OnGenKey() – передає управління функції Execute() коли активована кнопка генерації ключів „Згенерувати ключі”;
* OnEmptyNotes() – перевіряє чи введено значення в полях;
* OnEncrypt() – передає управління функції Encrypt() коли активована кнопка генерації ключів „Шифрувати”;
* OnDecrypt() – передає управління функції Decryptt() коли активована кнопка генерації ключів „Дешифрувати”;
* OnCompare() – виконує порівняння текстового файлу та файлу що був одержаний після дешифрування;
* OnAbstract() – виводить в окремому вікні опис програми.

Коли користувач запускає програму щоб приступити до процесу шифрування, йому перш за все потрібно виконати генерацію ключів, натиснувши кнопку „Згенерувати ключі”. Метод класу Cl\_Dlg OnGenKey(), проаналізувавши подію, передає виконання функції Execute().

Незважаючи на те що об’явлення класу, до якого належить дана функція знаходиться у файлі Dlg.h, реалізація функції знаходиться у файлі rsa.cpp. Засоби програмування C++ дозволяють виконувати таку процедуру побудови коду програми, що дає можливість зручно розбивати програму на модулі та окремі файли, цим самим полегшуючи процедуру написання та редагування файлів. Функція Execute() звертається до функції GeneratePrimeNumbers(), реалізація якої знаходиться у файлі genPrimeNo.cpp. Дана функція реалізує генерацію двох великих простих чисел розміром від 32 до 256 біт, викликаючи при цьому функцію SolShtr(), і присвоює їх значенням m\_Prime1 і m\_Prime2. Виконання програми повертається до функції Execute(), яка обчислює значення модуля m\_n та значення функції Ейлера – змінна f\_Eiler.

SelectKvidkr() вибирає відкритий ключ відповідно до процедури вибору ключа в криптосистемі RSA переводить цей ключ у відповідну систему числення згідно таблиці 1.2 При виборі ключа також функцією IsRelativePrime() проходить перевірка на те чи відкритий ключ та функція Ейлера взаємно прості.

Після того, як відкритий ключ вибрано, обчислюється значення закритого ключа, який обчислюється за допомогою функції CalculateKsekr(). Значення відкритого ключа зберігається у змінній m\_d. У процесі виконання обчислення ключів походить вивід результатів у спеціальні вікна, перше з яких показує процес виконання обчислень, а друге показує інформацію про згенеровані ключі, а саме : розмір (в бітах) або значення першого і другого простого числа, значення або розмір значення модуля, значення або розмір відкритого та закритого ключів. Якщо користувач сам вибрав значення простих чисел то у інформаційних вікнах буде показано значення цих ключів, якщо ж прості числа згенерував комп’ютер то буде показано розмір числа в бітах. Цю процедуру зроблено тому. що недоцільно показувати прості числа дуже великих розмірів.

Коли повністю пройшла процедура генерації відкритого і секретного ключів, користувач може вільно вибирати текстовий файл для того, щоб його зашифрувати. Після активації кнопки „Шифрувати” появляється діалогове вікно вибору файлу, де потрібно вибрати відповідний текстовий файл.

Натискання кнопки приводить у дію функцію OnEncrypt(), яка перевіряє чи згенеровано ключі і якщо так, то виконується функція Encrypt(), якій передаються як параметри шляху файлу, що потрібно зашифрувати, і шлях, куди зберегти зашифрований файл. Функція знаходиться в окремому файлі Encrypt.cpp.

Функція Encrypt() здійснює шифрування даних, що знаходяться у текстовому файлі. Зашифрований файл зберігається з іншим розширенням у тій же дерикторії, в якій знаходився вхідний файл. Зміну розширення файлу з текстового на файл з розширенням enc виконує функція ChangeFileExtension().

Процедура дешифрування відбувається натисненням кнопки „Дешифрувати” і виконанням функції OnDecrypt(), яка також викликає функцію CheckKeyGenStatus(), що перевіряє чи згенеровано ключі. Якщо згенеровано, то з’являється вікно для вибору зашифрованого файлу з розширенням .enc, якщо ключі не згенеровано, то виводиться повідомлення з пропозицією згенерувати ключі. Коли вибрано файл, проводиться процедура розшифрування цього файлу.

Повний лістинг розробленого програмного засобу поданий в додатку А.

3.3 Сценарій діалогу користувача

Будь-яку прикладну програму розробляють для широкого кола корис­тувачів і вона повинна мати прості та зручні засоби взаємозв'язку з нею. Ці засоби називають інтерфейсом або діалогом користувача.

Інтерфейс користувача являє собою сукупність засобів для обробки та відображення інформації, максимально пристосованих для зручності користувача. У графічних системах інтерфейс користувача реалізовується багатовіконним режимом, змінами кольору, розміру, видимості (прозорість, напівпрозорість, невидимість) вікон, їх розташуванням, сортуванням елементів вікон, гнучкими налаштовуваннями як самих вікон, так і окремих їх елементів (файли, папки, ярлики, шрифти тощо), доступністю багатокористувацьких налаштувань.

Через інтерфейс користувач керує роботою програми, отримує від неї повідомлення, відповідає на запити програми та ін.

Сценарій діалогу користувача розробленої програми реалізований у віконному режимі. Вигляд вікна зображений на рисунку 3.1

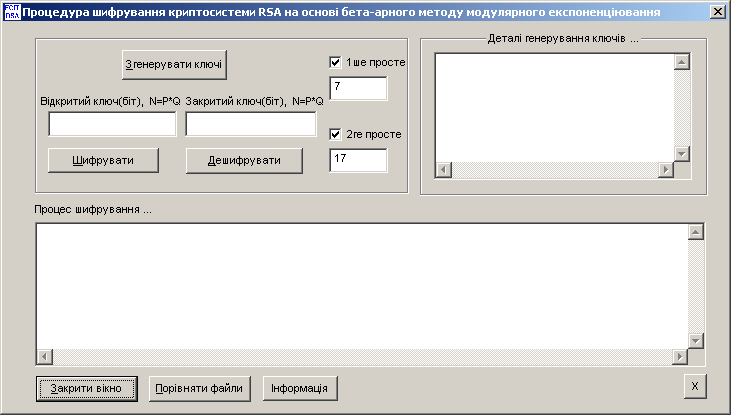


Рисунок 3.1 – Загальний вигляд вікна реалізованої програми

Програма задовольняє такі загальноприйняті правила роботи з вікнами діалогу:

1.) якщо варіант заповнення вікна діалогу повністю задовольняє користува­ча, він повинен натиснути на клавішу Еnter;

2.) якщо повідомлення у вікні діалогу необхідно відредагувати, то потрібно спочатку натиснути на будь-яку клавішу зі стрілкою переміщення курсора, а потім здійснити редагування;

3.) якщо вікно діалогу потрібно заповнити новою інформацією, то необхідно набрати її. При натискуванні на клавішу з будь-яким символом запропоноване системою значення стирається;

4.) для відміни роботи з вікном діалогу натискується клавіша Еsc. Можливості вікон діалогу графічних систем набагато ширші текстових. Вони включають додаткову кількість елементів, таких, наприклад, як командні кнопки, прапорці, поля вибору та ін.

Вікно програми складається з наступних частин це :

* заголовок програми;
* частина, в якій проводиться управління процесом шифрування;
* частина, в якій зображується процес роботи програми;
* частина, що відображає деталі генерування ключів.

Для того, щоб розпочати шифрування даних користувачу перш за все, треба згенерувати ключі. Щоб виконати генерацію ключів користувачу потрібно натиснути кнопку „Згенерувати ключі”. При цьому користувач може самостійно вказати прості числа для генерації ключів, вводячи їх у полях „1ше Просте” та „2ге Просте”. Якщо ж користувач хоче, щоб комп’ютер згенерував дуже великі прості числа, то йому потрібно забрати галочки над полями вводу. В процесі генерування в правій верхній частині буде проводитися відображення деталей генерації ключів (див. рисунок 3.2).

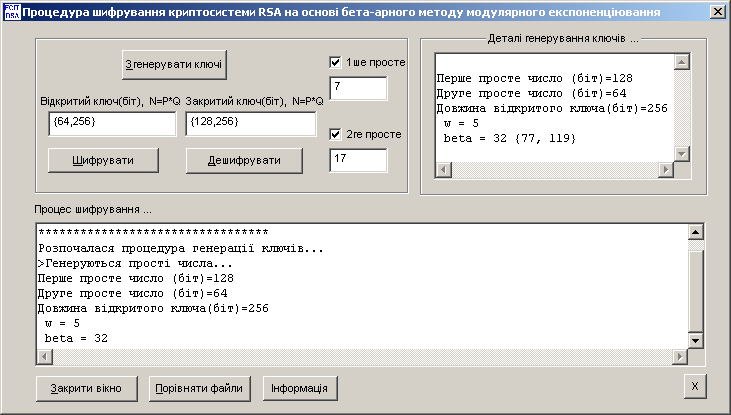


Рисунок 3.2 – Відображення процесу та деталей генерування ключів

При натисненні кнопки “Шифрувати” появляється інше діалогове вікно для того, щоб користувач міг вибрати файл для шифрування (див. рисунок 3.3).

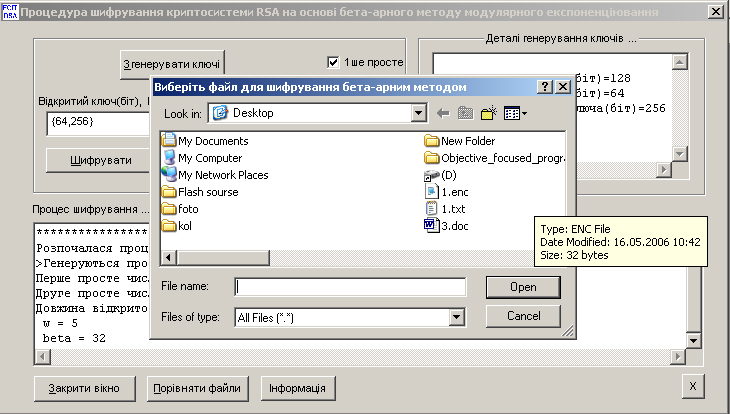


Рисунок 3.3 – Вікно вибору файлу для шифрування

Після вибору файлу проходить процедура шифрування. У частині “Процес шифрування” відображається процес шифрування, а зокрема відображається шлях до файлу, який шифрується і назва та шлях до зашифрованого файлу (див. рисунок 3.4).

Повідомлення, що відображаються у частині „Процес шифрування” можна стерти за допомогою кнопки „Х”, що знаходиться у нижній правій частині вікна.

У програмі діалог користувача також реалізований з використанням гарячих клавіш. Підкреслена буква у слові, що розміщене на кнопці, яка виконує якусь дію, вказує на гарячу клавішу, яку можна натиснути на клавіатурі комп’ютера і відбудеться виконання саме тої дії, яку передбачає кнопка.

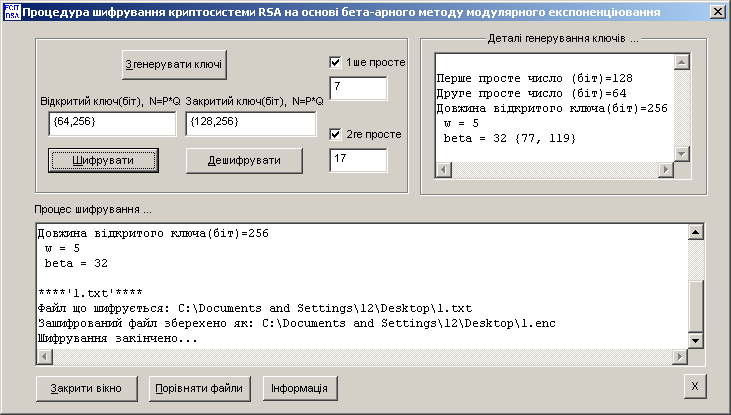


Рисунок 3.4 – Відображення процесу шифрування

В програмі також реалізовано вікно, яке дає загальну інформацію про правила користування програмою, викликати це вікно можна натиснувши клавішу „Інформація” (див. рисунок 3.5), а також вікно, яке дає загальну інформацію про програму та розробника.

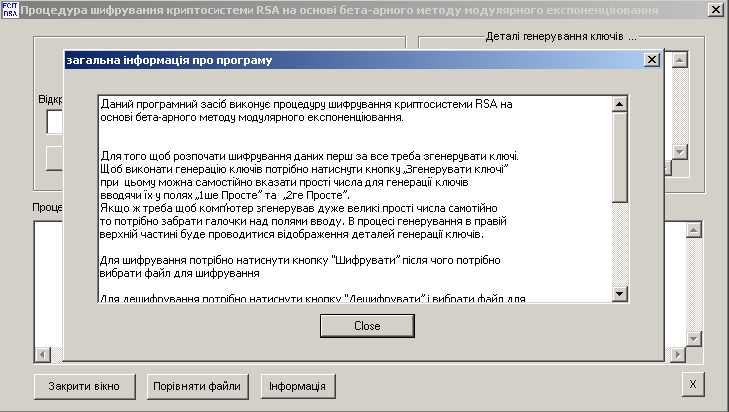


Рисунок 3.5 – Відображення загальної інформації про правила користування програмою

3.4 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 файл абсолютно має бути нечитабельним і не повинен містити ніякої інформації. Інформація може бути одержана лише тоді, коли відбудеться процес дешифрування цього файлу.

Для прикладу зашифруємо файл, який містить слова „реалізація криптосистеми RSA”. На рисунку 3.6 зображено вміст файлу, що шифрується.

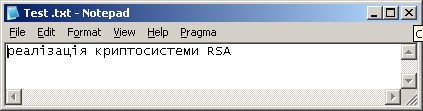


Рисунок 3.6 –Вміст файлу, що шифрується

Після проведення процесу шифрування програма створює файл, що містить криптотекст, який без дешифрування абсолютно не несе ніякої інформації, а отже не може бути прочитаний. На рисунку 3.7 зображений зашифрований файл, відкритий у текстовому редакторі.

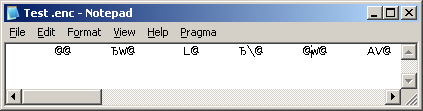


Рисунок 3.7 – Вміст зашифрованого файлу

Криптографічний файл з розширенням .enc може бути використаний при передачі файлу по каналу зв’язку або просто для надійного зберігання, таким чином забезпечується криптографічний захист даних файлу.

Особливістю реалізації даного програмного засобу є те, що в ньому є можливість протестувати виконані процедури шифрування та дешифрування. Таким чином користувач може бути впевнений, що файл, який він зашифрує, буде успішно розшифрованим.

Коли процедура шифрування та дешифрування відбулася, користувач має можливість перевірити правильність методу шифрування, порівнявши два файли – текстовий файл, який був потім зашифрований, та вихідний розшифрований файл. Порівняння здійснюється автоматично натиском кнопки „Порівняти файли”. У частині „Процес шифрування” відображається інформація про схожість файлів (див. рисунок 3.8).

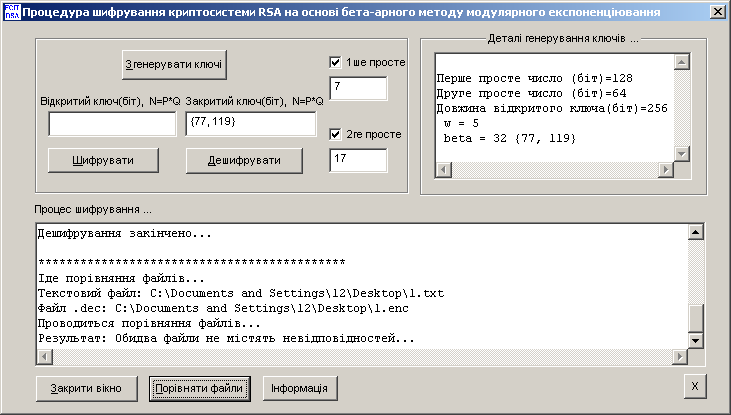


Рисунок 3.8 – Відображення процесу порівняння файлів

Може виникнути ситуація коли, файли містять різну інформацію. Це може статися у тому випадку, коли файл, що шифрувався був не текстовий або пошкоджений. Для прикладу зашифруємо графічний файл з розширенням .jpg (див. рисунок 3.9).

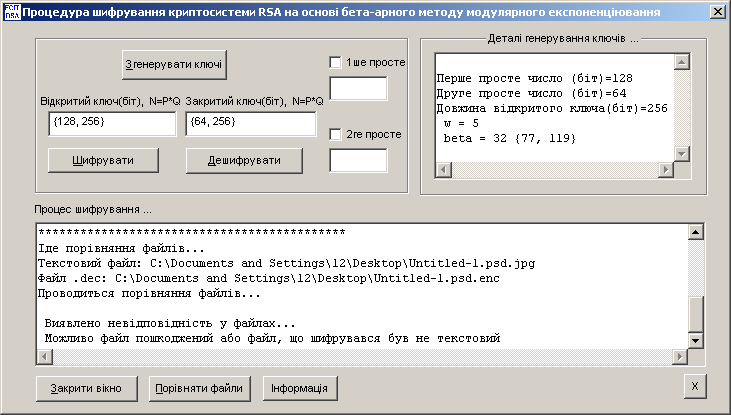


Рисунок 3.9 –Результат порівняння файлу

У такому випадку шифрування відбулося невірно і тому у частині „Процес шифрування” з’являється повідомлення про те, що виявлено невідповідність у файлах.

Тестування показало, що одним із недоліків програми є те, що зашифрований файл, що має розширення enc займає набагато більше місця на диску, ніж файл, який шифрувався. Крім того можна прослідити залежність розміру зашифрованого файлу від розміру файлу, що шифрувався а також довжиною ключа, що використовувався при шифруванні (див. таблицю 3.1 ).

Слід зазначити, що в міру того, як збільшується розмір зашифрованого файлу, збільшується крипостійкість системи, а тому, якщо інформація є дуже важливою, слід використовувати великі ключі для шифрування.

Таблиця 3.1 – Залежність розміру зашифрованого файлу від файлу що шифрувався та довжини ключа

|  |  |  |
| --- | --- | --- |
| Розмір файлу що шифрується | Довжина ключа (біт) | Розмір зашифрованого файлу |
| 1 КБ | 32 | 8 КБ |
| 64 | 16.1 КБ |
| 128 | 32.5 КБ |
| 256 | 65 КБ |
| 1 МБ | 32 | 8,06 МБ |
| 64 | 17.0 МБ |
| 128 | 35.5 МБ |
| 256 | 67.03 МБ |
| 5 МБ | 32 | 42,06 МБ |
| 64 | 85.0 МБ |
| 128 | 170.5 МБ |
| 256 | 345.07 МБ |

Швидкість генерування ключів шифрування залежить від частоти процесора комп’ютера. В таблиці 3.2 подано часові характеристики генерування ключів різної довжини на комп’ютері AMD Duron 700.

Таблиця 3.2 – Швидкість генерування ключів шифрування

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Довжина ключа | Час генерування | | | |
|  | 1 | 2 | 3 | 4 |
| 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 на основі бета-арного методу модулярного експоненціювання має доступний інтерфейс, є стійким і надійним, а тому може успішно використовуватись як на підприємстві, так і приватними користувачами для захисту конфіденційної інформації.

ВИСНОВКИ

В результаті розробки дипломного проекту:

1. побудовано математичні моделі виконання кожного з розглянутих алгоритмів, що дало змогу провести детальний аналіз їх продуктивності та стійкості до атак спеціального виду, а зокрема до атаки аналізу енергоспоживання;
2. проведено аналіз залежності часу виконання алгоритму *β*-арного методу від значення степеня основи, який показав, що для кожного значення довжини  можна знайти таке оптимальне , при якому відбувається найменша затримка, що дає змогу проектувати високопродуктивні засоби, які реалізують операції сучасних асиметричних криптоалгоритмів;
3. досліджено залежність енергоспоживання від кількості операцій алгоритму захисту інформації, що дало змогу ґрунтовно обирати алгоритм та його параметри для реалізації засобів криптографічного захисту інформації, стійких до атак спеціальних впливів;
4. наведено результати досліджень значення нормованої стійкості та швидкодії кожного з досліджуваних методів для заданих параметрів, що можуть бути використанні при побудові сучасних високопродуктивних засобів захисту інформації, стійких до атак спеціального виду, що базуються на асиметричних криптографічних алгоритмах;
5. розроблено, протестовано та впроваджено програмний засіб для шифрування даних на основі криптосистеми RSA. Програмний засіб може використовуватися як практично, так і в навчальних цілях. Він здатен забезпечувати найвищу швидкодію та стійкість криптосистеми не тільки до часового аналізу, а й до інших атак спеціального виду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Морзе Н.В. Навчально-наукові електронні ресурси сучасного університету в умовах становлення суспільства знань. – Інформаційні технології і засоби навчання. 2011. No3 (23).
2. Аллатова И.В. Новые информационные технологии в обучении. – М.: Изд. МГПУ, 2006. – 318 с.
3. Сисоєва С. О. Організація неперервної професійної освіти на базі Інтернет-технологій / C. О. Сисоєва / Педагогічний процес: теорія і практика: зб. наук. праць - К.: Екмо, 2006. - Вип. 3. – С. 286-294.
4. Биков В. Ю. Дистанційний навчальний процес: навч. посібник / В. Ю. Биков, В. М. Кухаренко. - К.: Міленіум, 2005. - 292 с.
5. A. Shamir, E.Tramer. Acoustic cryptanalysis: on nosy people and noisy machines. Eurocrypt 2004 rump session, 2004.
6. Y. Tsunoo, T. Saito, T. Suzaki, M. Shigeri, H. Miyauchi.Cryptanalysis of DES Implemented on Computers with Cache. CHES 2003, LNCS 2779, pp.62–76, 2003.
7. Добровольский Г. А. Основы работы в Интернете: учеб. пособие к факультативному курсу / Г. А. Добровольский. - Запорожье: ЗГУ, 2001. - 36 с.
8. Кухаренко В. М. Дистанційне навчання: умови застосування. Дистанційний курс: навч. посібник. - 2-е вид., доп. / В. М. Кухаренко. - Х. : НТУ «ХПІ», «Торсінг», 2001. - 320 с.
9. Матушанський Г. У. Открытое и дистанционное образование / Г. У. Матушанський, А. В. Юсупова // Специалист. - 2001. - No 11. - С. 23-24.
10. Ничкало Н. Г. Проблема формування сучасного виробничого персоналу в Україні: стратегія і перспективи наукових пошуків / Н. Г. Ничкало // Педагогіка і психологія професійної освіти: результати досліджень і перспективи: зб. наук. праць за ред. І. А. Зязюна та Н. Г. Ничкало. - К., 2003. – С. 139-152.
11. Позднеев Б. М. Разработка национальных и международных стандартов в области электронного обучения / Б. М. Позднеев // Информатизация образования и науки. - 2009. - No 2. - С. 4-12.
12. Полат Е. С. Теория и практика дистанционного обучения: учеб. пособие / Е. С. Полат. - М.: Академия, 2004. - 416 с.
13. Протасова Н. Г. Післядипломна педагогічна освіта: сутність і сучасне розуміння / Н. Г. Протасова, B. І Пуцов // Неперервна професійна освіта: теорія і практика: наук.-метод. Журнал. - 2002. - Вип. 2. - С. 160-165.
14. Федорчук І.І. Нові інформаційні технології навчання, дистанційна освіта: реалії сьогодення і перспективи розвитку /І.І.Федорчук, І.П.Федорчук// Сучасні інформаційні технології та інноваційні методики навчання у підготовці фахівців: методологія, теорія, досвід, проблеми: збірник наукових праць: у 2-х ч. / [редкол. : І.А.Зязюн (голова) та ін.]. – Київ; Вінниця: ДОВ «Вінниця», 2002. – Ч.1. – Вінниця. – 2002. – С. 515–520.
15. Захарова И.Г. Информационные технологии в образовании: учеб. пособие для студ. высш. пед. учеб. заведений / И.Г. Захарова. – М.: Издательский центр «Академия», 2003. – 192 с.
16. Бабич М.П., Жуков І.А. Комп’ютерна схемотехніка: Навчальний посібник.- К.:МК-Прес, 2004.-412с
17. Бойко В.І. Схемотехніка електронних систем. Цифрова схемотехніка. Підручник / В.І. Бойко, А.М. Гуржій, В.Я Жуйков - К.:Вища школа, 2004.-423с.
18. Прянишников В.А. Электроника: Полный курс лекцій. – СПб.Корона принт; М.: Бином – Пресс, 2006.- 416с.
19. Терехин В.Б., Соловьев Ю.А. Моделирование электронных схем в программе Electronics Workbench. Ч. 1. Создание схем. Ч.2. Элементная база: лабораторный практикум. – Северск: СТИ ТПУ, 2000. – 244 с.
20. Шило В.Л. Популярные цифровые микросхемы: Справочник. 2 - е изд., испр. – Челябинск: Металлургия, Челябинское отд., 1989.- 352 с.
21. Лещенко М.Є. Основи мікроелектроніки / М.Є. Лещенко, В.Є. Овчаренко. – Х. : Нац. аерокосм. ун-т „Харк. авіац. ін-т”, 2005. – 312с.
22. Комп`ютерна електроніка: Навч. посібник. Частина І/ІІ А.П.Оксанич, С.Е.Притчин, О.В.Вашерук.- Харків: "Компанія СМІТ", 2006.- С200 - 256.
23. Рябенький В.М., Жуйков В.Я., Гулий В.Д. Цифрова схемотехніка: Навч. Посібник. - Львів: Видавництво «Новий світ 2000», 2009.-736с.
24. Резисторы, конденсаторы, трансформаторы, дроссели, комутационные устройства РЭА: Справочник./ Н.Н. Акимов, Е.П. Ващуков, В.А. Прохоренко, Ю.П. Ходоренок. – Мн.:Беларусь, 1994.- 591с.
25. Токхейм Р. Основы цифровой электроники.- М.:Мир, 1989.
26. Хернитер М. Е. MULTISIM. Современная система компьютерного моделирования и анализа схем электронных устройств. – М.: Издательство: ДМК-Пресс. – 2009. – 409с.
27. Марченко А. Лабораторный практикум по электротехнике и электронике в среде Multisim. Учебное пособие для вузов. / А. Марченко, С. Освальд. – М.: НИЯУ МИФИ, 2009. – 364 с.
28. Алексенко А. Г. Основы микросхемотехники — М.: Юнимедиастайл, 2009. — 448 с.
29. Никитин В.А. Схемотехника интегральных схем ТТЛ, ТТЛШ и КМОП: Учебное пособие. М.: НИЯУ МИФИ, 2010. – 64 с.
30. Васильцов І. В., Васильків Л. О.Стійкість сучасних алгоритмів модульярного експоненціювання до часового аналізу // Науково технічний журнал „Захист інформації”, №1, 2005, С. 54-69
31. Vasyltsov I., Vasylkiv N., Chyrka M., Vasylkiv L. Investigation of Modern Exponentiation Algorithms // Матеріали міжнародної науково-технічної конференції „Сучасні проблеми радіотехніки, телекомунікації, комп’ютерної інженерії” TCSET ‘2004, 24-28 лютого 2004, Львів-Славсько, Україна, С. 291-293
32. Вербіцький О.В. Вступ до криптографії. – Львів: Видавницво науково-технічної літератури, 1998. – 247 с.
33. Ємець В., Мельник А., Попович Р. Сучасна криптографія. Основні поняття. – Львів: БаК, 2003. –144 с.
34. Молдовян А.А., Молдовян В.А., и др. Криптография. – Серия “Учебники для вузов. Специальная литература”. – Спб.: Издательство “Лань”, 2000. – 224 с.
35. Столлингс В. Криптография и защита сетей: принципы и практика.: Пер. с англ. – М.: Изд. Дом «Вильямс», 2001 – 672 с.
36. Чмора А.Л. Современная прикладная криптографія. – М.: Гелиос АРВ, 2002. – 256 с.і
37. Горпенюк А.Я., Дудикевич В.Б., Ломницький І.Б. Підвищення швидкодії при обчисленні важкооборотних функцій в асиметричних алгоритмах шифрування. // Захист інформації. – 2003. – №1(18) – С.36-43.
38. Seong-Min Hong, Sang-Yeop Oh, Hyunsoo Yoon. New Modular Multiplication Algorithms for Fast Modular Exponentiation. – Spring-Verlag, 2018.
39. Ященко В. В. Основные понятия криптографии // Математическое просвещение. Сер. 3. 2016.- №2.- С. 53-70.
40. Виноградов И. М. Основы теории чисел.- М.: Наука, 2012.
41. Карацуба А. А. Основы аналитической теории чисел.-М.: Наука, 2013.
42. Василенко О. Н. Современные способы проверки простоты чисел // Кибернетический сборник, - 2008. Вып. 25. - С. 162-188.
43. Боревич З.И. Шафаревич И.Р. Теория чисел. - М.: Наука, 2014.
44. Кострикин А.И. Введение в алгебру. - М.: Наука, 2006.
45. Прахар К. Распределение простых чисел. - М.: Мир, 2007.
46. Ахо А., Хопкрофт Дж.., Ульман Дж. Построение и анализ вычисли­тельных алгоритмов. - М.: Мир, 2009.
47. T.S. Messerges, E.A. Dabbish, R.H. Sloan, Examining smart-card security under the threat of power analysis attacks. IEEE Trans. Computers, 51(5), pp.541-552, 2020.
48. Варновский Н. П. Криптография и теория сложности // Математи­ческое просвещение. Сер. 3. №2. 2018. С. 71-86.
49. Брассар Дж. Современная криптология. Мир ПК.- 2017.- №3.
50. Кнут Д. Искусство программирования на ЭВМ. Т.2: Получисленные алгоритмы. - М.: Мир, 2017.