Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Поворозник Віталій Степанович

«Модель криптоалгоритму підвищеної стійкості до часової атаки/ Model of cryptoalgorithm with increased resistance to timing analysis attack »

Студент групи КІм – 21
Поворозник Віталій Степанович
_____

Науковий керівник
к.т.н., доцент Дубчак Л.О.
_____

Тернопіль – 2020

РЕЗЮМЕ

Кваліфікаційна робота на тему “Модель криптоалгоритму підвищеної стійкості до часової атаки” зі спеціальності 123 «Комп’ютерна інженерія» освітнього ступеня «магістр» написана обсягом 78 сторінка і містить 6 ілюстрацій, 5 таблиць, 2 додатків та 57 джерел за переліком посилань.

Метою роботи є підвищення стійкості криптосистем систем до часового аналізу в реальному часі. Атаки в часі - це потужний метод зламу теоретично безпечних криптографічних примітивів.

Методи досліджень. Для розв’язання поставлених задач у кваліфікаційній роботі використано: методи теорії ймовірності і теорії інформації пов'язані зі стійкістю захисту інформації в різних аспектах, а також методи асиметричного захисту інформації та математичного аналізу.

Результати дослідження: удосконалено реалізацію криптоалгоритму RSA, що дозволило уникнути атаки в часі за рахунок використання удосконалених методів захисту.

Результати роботи можуть бути використані в сучасних комп'ютерних системах і мережах.

Можливими напрямками подальших досліджень є застосування запропонованих моделей в інших алгоритмах захисту інформації.

КЛЮЧОВІ СЛОВА: АЛГОРИТМ, АТАКА В ЧАСІ, СИНТЕЗ, АНАЛІЗ, КРИПТОАЛГОРИТМ, ПРОГРАМНА СИСТЕМА.

RESUME

The qualification work on the topic "Model of cryptoalgorithm of increased resistance to time attack" in the specialty 123 "Computer Engineering" of the educational degree "Master" is written in 78 pages and contains 6 illustrations, 5 tables, 2 appendices and 57 sources from the list of references.

The aim of the work is to increase the resilience of cryptosystems to real-time time analysis. Time attacks are a powerful method of breaking theoretically secure cryptographic primitives.

Research methods. To solve the tasks in the qualification work used: methods of probability theory and information theory related to the stability of information protection in various aspects, as well as methods of asymmetric information protection and mathematical analysis.

The results of the study: the implementation of the RSA cryptoalgorithm was improved, which allowed to avoid attacks in time due to the use of advanced protection methods.

The results of the work can be used in modern computer systems and networks.

Possible areas of further research are the application of the proposed models in other algorithms for information protection.

KEY WORDS: ALGORITHM, ATTACK IN TIME, SYNTHESIS, ANALYSIS, CRYPTOALGORITHM, SOFTWARE SYSTEM.

ЗМІСТ

ВСТУП

Актуальність теми. Зрозуміло, що в сучасному цивілізованому світі, в якому величезна кількість видів діяльності людей супроводжується комп'ютерною підтримкою, проблема безпеки комп'ютерних систем є надзвичайно актуальною. Врахування усіх недоліків захисних механізмів, передбачення можливих наслідків та загроз безпеки інформаційних ресурсів може убезпечити комп'ютерних користувачів від небажаних впливів різноманітних обставин і сторонніх людей на їхнє життя. Саме тому в наш час професіоналам потрібно володіти навиками використання апробованих методів і надійних засобів захисту комп'ютерних даних та розумітися у проблемі захисту інформаційних ресурсів в усій її багатогранності.

За всіх часів доступ до відомостей, що відносяться до певних сфер людської діяльності, дозволяв окремим особам чинити дії, що можуть нанести шкоду людям чи організаціям, у рамках яких протікає ця діяльність. Працюючи із сучасними інформаційними системами користувачі повинні бути досить обережними, адже паралельно з виконанням процедур опрацювання даних в комп'ютерах здійснюється реєстрація усіх слідів діяльності людини, а це іноді може спричинити непередбачувані наслідки.

Мета і завдання дослідження**.** Метою роботи є підвищення стійкості криптосистем систем до часового аналізу в реальному часі. Атаки бічних каналів - це потужний метод зламу теоретично безпечних криптографічних примітивів. З часу перших робіт Кохера, ці атаки широко використовувались, щоб порушити безпеку численних криптографічних реалізацій. На високому рівні можна розрізнити два типи атак бічних каналів, засновані на засобах, які використовує зловмисник: апаратні атаки, які контролюють витік через вимірювання (як правило, за допомогою спеціального лабораторного обладнання) фізичних явищ, таких як електромагнітні випромінювання, споживання енергії або акустичне випромінювання, а також програмні атаки, які не потребують додаткового

обладнання, а замість цього покладаються на програмне забезпечення зловмисника, що працює на цільовій машині або взаємодіє з нею. Приклади останніх включають синхронізовані атаки, які вимірюють варіації часу криптографічних операцій та атаки кеш-пам'яті, що спостерігають за шаблонами доступу до кешу.

Об’єкт дослідження – процеси моделювання методу захисту передачі інформації різного рівня конфіденційності в комп’ютерних системах з динамічно-розподіленим навантаженням.

Предмет дослідження – методи та засоби підвищення стійкості комп’ютерної системи до атаки в часі.

Методи дослідження – методи теорії ймовірності і теорії інформації пов'язані зі стійкістю захисту інформації в різних аспектах, а також методи асисиметричною захисту інформації та математичного аналізу.

Наукова новизна отриманих результатів. Удосконалено реалізацію криптоалгоритму RSA, що дозволило уникнути атаки в часі за рахунок використання удосконалених методів захисту.

Практичне значення отриманих результатів: Практична значимість виконаної роботи обумовлена тим, що в ній побудована вичерпна математична схема та створено програмний продукт, досліджено часові характеристики запропонованого підходу щодо захисту інформаційних потоків.

Публікації та апробація ВКР. Отримані результати апробовані в межах III Науково-практичної конференції молодих вчених і студентів «інтелектуальні комп’ютерні системи та мережі» [1,2]

1 СУЧАСНИЙ СТАН І ПЕРСПЕКТИВИ РОЗВИТКУ КОМП’ЮТЕРНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

1.1 Загальна постановка проблеми

Збільшення кількості мереж передачі даних різного призначення та масштабів з різними фізичними середовищами даних (дротовими та бездротовими), які базуються на різних протоколах та мають різну архітектуру, передбачає розробку надійних систем захисту. Системи безпеки реалізовані у відповідних протоколах та алгоритмах шифрування. Досить часто в мережах передачі даних, зокрема бездротових, використовуються потокові алгоритми шифрування, через низьку обчислювальну потужність, необхідну для роботи алгоритму шифрування та особливостей протоколів передачі даних. До найбільш відомих технологій належать: стандарт GSM (алгоритми A5 / 1, A5 / 2), UMTS (алгоритм UEA2 / UIA2 (інші назви A5 / 4, SNOW 3G)), стандарт CDMA2000 (алгоритм ORYX), стандарт IEEE 802.11 b / g (алгоритм RC4 (протокол WEP), IEEE 802.15.1-2002 (алгоритм E0), стандарт мобільного радіоінтерфейсу геостаціонарної орбіти Землі (GEO) (алгоритми GMR-1, GMR-2) та багато інших.

Криптографічна стабільність деяких із вищезазначених алгоритмів є незадовільною. Це пов'язано зі створенням нових методів та засобів криптоаналізу, тому необхідно перевірити існуючі та існуючі алгоритми шифрування.

Розглянемо методи криптоаналізу сучасних потокових шифрів (табл. 1.1). Слід зазначити, що більшість методів криптоаналізу дозволяють отримати результат лише з певною ймовірністю, тоді як повторення крипто аналітичного експерименту з іншими вхідними параметрами дозволяє збільшити ймовірність.

Таблиця 1.1 - Характеристики поширених потокових алгоритмів шифрування

Потоковий шифр	Дата створення	Розробник (патент/ стандарт)	Атака		Криптостійкий на даний час	Застосування
			Найбільш відома	Обчислювальна складність		
A5/1, A5/2	1987, 1989	Частина GSM-стандарту	На основі відомих відкритих текстів (райдужні таблиці, "time-memory tradeoff")	2^{39}	ні	шифрування голосу в GSM-мережах
GEA5/1, GEA5/2	1993	Частина GSM-стандарту	Атака "розділяй і володарюй" ("time-memory tradeoff")	2^{45}	ні	шифрування даних в GSM-мережах
Achterbahn -128/80	2006	Берндт Ґамел, Рейнер Ґотферт, Олівер Найфер	Підбір для довжини фреймуL ≤ 244. Кореляційна атака для L ≥ 248.	2^{80}	так	Проект eSTREAM (EU ECRYPT)
FISH	1993	Siemens	Текстова атака	2^{11}	ні	Телефонія
E0	1999	Частина стандарту IEEE 802.15.1	Статична атака	2^{64}	так	Bluetooth
RC4	1987	Рон Райвест	На основі відомих відкритих текстів	2^{13} (2^{33})	ні	Протоколи SSL, WEP
Salsa20	До - 2004	Деніел Бернстейн	Метод ймовірнісно-нейтральних бітів	2^{251} для 8 раундів	так	Проект eSTREAM (EU ECRYPT)
Scream	2002	Шай Холеві, Дон коперсміт та Чаранжит Жутала	-	-	так	Шифрування даних на жорстких магнітних дисках
SEAL	1997	U.S. Patent 5,454,039, U.S. Patent 5,675,652	Метод ймовірнісно-нейтральних бітів	-	ні	Шифрування даних на жорстких магнітних дисках
SNOW 3G 2.0	До-2003	ISO/IEC IS 18033-4	-	-	так	3rd Generation Partnership Project
SOSEMA NUK	До - 2004	Олівер Біллет, Ніколас Куртуа та ін. (без патентних обмежень)	-	-	так	3G-мережі
Turing	2000–2003	Ґреґорі Роуз і Філіп Хоукс (Qualcomm)	-	-	Так	CDMA-мережі

На основі теоретичних оцінок T і S, а також імовірнісних характеристик результату, які вказують на кількість необхідних експериментів, можна отримати аналітичні значення часу і пам'яті для пошуку бажаних результатів. Врахування специфічних особливостей апаратно-програмних комплексів дозволяє отримати числові характеристики часу та пам'яті, близькі до абсолютних.

Найчастіше в ході крипто аналітичних досліджень використовуються диференціальні та лінійні методи, а також ―mod n‖-атака та метод відповідних ключів. Серед відносно нових - метод алгебраїчного криптоаналізу, який вперше був запропонований Н. Куртуа. На думку авторів, цей метод є одним з найбільш перспективних, хоча його практичне використання вимагає подальших досліджень. Його ключові особливості в порівнянні з іншими включають: універсальність (підходить як для потокового, так і для блокових шифрів), можливість масштабування, можливість подальшого вдосконалення.

Алгебраїчні криптоаналітичні методи - це методи, що представляють представлення криптографічних перетворень ключа, вхідних та вихідних даних для шифрування у вигляді якогось рівняння. Тоді множини таких перетворень утворюють систему рівнянь. Сьогодні цей підхід застосовується для криптосистем, які мають практичне використання. Однак його застосування до широкого спектра шифрових систем у багатьох випадках має теоретичний характер або в даний час не вивчено повністю. Загальна схема алгебраїчного криптоаналізу наведена на рисунку 1.

Можливість використання даного методу пов'язана з необхідністю розв'язання цілої низки теоретичних та прикладних задач, зокрема:

1. Представлення алгоритму шифрування у вигляді системи рівнянь у аналітичній формі.

2. Перетворення аналітичної форми криптоалгоритму до кон'юнктивної нормальної форми.

3. Розв'язання системи рівнянь у кон'юнктивній нормальній формі.

Якщо п.1 і п.2 вимагають чіткої математичної формалізації криптоаналітичної системи, то п.3 пов'язаний з чисельним розв'язком відповідної системи рівнянь великої обчислювальної складності.

Рисунок 1.1 - Загальна схема алгебраїчного криптоаналізу

У було представлено AES-128 великою системою рівнянь: 8000 рівнянь з 1600 невідомими. У працях Н. Куртуа показано, що складність розв'язання системи квадратних рівнянь з багатьма невідомими може бути суттєво спрощена, якщо система є розрідженою і має регулярну структуру, на прикладі алгоритмів AES та Serpent, а також були показані практичні реалізації атак на потокові (LILI-128, Toyocrypt та інші) та блокові (спрощені варіанти AES, Camellia, KHAZAD, MISTY-1, KASUMI та інші) шифри.

З метою спрощення процедури верифікації досліджуваних алгоритмів використовуються апаратно-програмні засоби криптоаналізу. Серед програмного забезпечення для алгебраїчного криптоаналізу варто виділити наступне:

1. Система програм Ніколаса Куртуа, який є автором даного методу. Програмне забезпечення є закритим і орієнтованим на спрощені версії алгоритмів.

2. Система програм Мартіна Альбрехта є розширенням для відкритого математичного пакету sage.

3. Система програм Мета Суса містить у собі цілу низку утиліт для реалізації відповідного методу: cryptominisat2 (засіб для розв'язування рівнянь у кон'юнктивній нормальній формі), anf2cnf (конвертер з алгебраїчної нормальної в кон'юнктивну нормальну форму) та низка інших.

Дане програмне забезпечення варто розглядати, як прототипи, в яких реалізовано основні теоретичні положення автора методу, тому дані програмні продукти потребують доопрацювання та модифікації. Також варто зазначити, що програмне забезпечення п.2 і п.3 є відкритим і безкоштовним, що дає змогу скористатися ним при розробці власної криптоаналітичної системи.

Стосовно апаратних засобів, на думку авторів, доцільно використати для цього паралельну та розподілену комп'ютерну систему на базі GPU-кластера. Проте, авторами розглядаються можливості реалізації таких задач й у грід-середовищі.

Підвищити ефективність зазначених методів криптоаналізу та криптоаналітичних систем у цілому, можна, шляхом:

1) зменшення розмірності системи рівнянь, і, відповідно, зменшення трудомісткості алгоритму;

2) декомпозиції та наступного розпаралелення фрагментів обчислювальної задачі між обчислювальними засобами;

3) використання спеціалізованих апаратних та програмних технологій, які б дозволили оптимізувати час виконання програм.

Перші два шляхи передбачають модифікацію алгоритмічного та математичного, а третій програмного забезпечення.

Модифікація програмного забезпечення полягає у розпаралеленні та векторизації деяких блоків криптоаналітичних алгоритмів. Розпаралелення полягає у паралельному опрацюванні окремими підпроцесами на окремих обчислювальних пристроях фрагментів обчислювальних задач алгоритму. Оскільки, потокові шифри побудовані на використанні регістрів зсуву зі

зворотніми зв'язками, тобто йде опрацювання векторів даних, то доцільно здійснити векторизацію алгоритмів шляхом використання векторних інструкцій сучасних процесорів, а саме SSE, SSE2, AVX і т.п. Авторами здійснюється модифікація наведеного програмного забезпечення для алгебраїчного криптоаналізу потокових шифрів шляхом розпаралелення та векторизація фрагментів коду з метою реалізації апаратно-програмного комплексу для криптоаналізу. Декомпозицію обчислювальної задачі планується здійснювати на етапах опрацювання вхідних даних — використовуючи декомпозицію за даними та на етапі розв'язання системи рівнянь — шляхом використання функціональної декомпозиції. При створенні програмної системи, з урахуванням архітектурних особливостей обчислювального GPU-кластера [10], буде використано наступні види паралелелізму:

– дрібнозернистий та середньозернистий — на рівні GPU-пристроїв, використовуючи технологію OpenCL;

– крупнозернистий — на рівні кластера, або грід-системи, шляхом використання технології MPI.

Розглянуто теоретичні аспекти криптоаналізу потокових шифрів. Наведено та здійснено короткий аналіз відомих методів та засобів криптоаналізу потокових шифрів. Сформульовано рекомендації щодо подальшого вдосконалення відомих криптоаналітичних методів.

Подальші наукові дослідження передбачають:

– розробку систем рівнянь у нормальній алгебраїчній формі, які б описували спрощені аналоги вітчизняних алгоритмів шифрування, з метою їх криптоаналітичного дослідження методами алгебраїчного криптоаналізу;

– модифікацію існуючого програмного забезпечення для його виконання в паралельних та розподілених обчислювальних системах, зокрема на GPU-кластерах та в грід-системах;

– тестування, відлагодження й коригування існуючого програмного забезпечення з урахуванням недоліків його роботи;

– створення комплексної системи для криптоаналізу потокових шифрів з урахуванням вимог організацій стандартизації.

Для адекватного розуміння проблем захисту даних необхідно ознайомитись з термінологією, властивою цій галузі інформатики. Давайте визначимо основні поняття в галузі захисту даних та наведемо загальну термінологію, яка стосується загроз безпеці (погроза безпеці) та деяких методів злому комп'ютерних систем.

інформаційна безпека означає захист інформаційних ресурсів та підтримку інфраструктури від випадкових та навмисних впливів природного або штучного характеру (несанкціонований доступ, знищення, модифікація, розкриття, затримка доступу тощо), що може призвести до шкоди власникам або користувачам інформаційних ресурсів та допоміжна інфраструктура. Під поняттям безпеки даних часто розуміють захист даних та програм від несанкціонованого доступу до них.

Існує три цілі захисту даних: цілісність даних, конфіденційність даних, доступність даних.

Цілісність даних - це гарантія того, що дані не були змінені, підроблені чи знищені внаслідок зловмисних дій чи інцидентів. Критичні дані означають дані, які потребують захисту через ймовірність заподіяння шкоди (ризику) у разі випадкового або навмисного розголошення, зміни чи знищення даних.

Конфіденційність даних - це статус даних, який визначає необхідний ступінь захисту. Конфіденційні дані - це дані, які потрібно захищати. Конфіденційні дані повинні бути відомі лише уповноваженим (і перевіреним) уповноваженим суб'єктам системи. До конфіденційних даних належать, наприклад, особисті дані користувачів, облікові записи (імена та паролі), дані про кредитні картки, дані про розробки та різні внутрішні документи, бухгалтерська інформація.

Витік даних - це результат дійсних повідомлень, внаслідок чого дані зберігаються відомими (доступними) суб'єктами, які не мають права на доступ до них.

1.2 Загальні вимоги до криптографічної стійкості

Тривалий час криптографія була скоріше мистецтвом, ніж наукою. Автори криптоалгоритмів діяли, як правило, "на удачу", оскільки не мали відповідної математичної теорії, за допомогою якої можна було б формалізувати криптографічні перетворення та перевести їх на мову науки.

Першою роботою, яка кардинально змінила цю ситуацію, вважалася робота американського інженера і математика Клода Шеннона "Теорія зв'язку в секретних системах", опублікована в 1949 р. У цій роботі Шеннон запропонував два загальних принципи створення криптографічних алгоритмів (шифри): розсіювання та змішування.

Шеннон назвав поширення впливу одного символу відкритого тексту на кілька символів розсіяння зашифрованого тексту. Узагальненням цього принципу є розширення ефекту одного ключового символу на декілька символів зашифрованого тексту, що перешкоджає відновленню ключа по частинах.

Перетасовка - це використання криптографічних перетворень, які ускладнюють узгодження символів відкритого тексту та зашифрованого тексту, а також ключа та зашифрованого тексту. Алгоритм криптовалюти повинен забезпечувати легкість шифрування та дешифрування (за умови, що відомий секретний ключ). Поширеним способом створення шифру є послідовне виконання простих зворотних перетворень, які часто використовують операції додавання, множення, заміщення та перестановки. Заміна визначає змішування, а перестановка - розсіювання. Перестановка змінює порядок символів відкритого тексту. Під час заміни набір символів відкритого тексту замінюється набором однакової кількості символів, а конкретний тип заміни визначається секретним ключем.

Використовуючи багаторазове чергування простих підстановок і перестановок, ви можете отримати стабільний алгоритм шифрування (шифрування) з хорошим загальним розсіюванням та змішуванням.

Відповідно до сучасних поглядів, розроблені на їх основі блокові симетричні криптовалюти та BSS є основним криптографічним механізмом забезпечення конфіденційності та цілісності, а також захисту інформації та інформаційних ресурсів від НРД. Ще однією важливою особливістю BSS є можливість забезпечити досить високу швидкість шифрування та розгортання ключів [9].

Для досягнення цієї мети ми спочатку аналізуємо вимоги до перспективних симетричних криптовалют, включаючи BSS, включаючи загальні вимоги до криптографічної стабільності та спеціальні вимоги, що визначають певні параметри криптографічного перетворення в загальновизнаних та закріплених у міжнародних стандартах режимах BSS.

У сучасних ІТС із криптографічним захистом інформації та інформаційних ресурсів довжина повідомлення, захищеного BSS, значно перевищує довжину ключа шифрування, тобто ентропія джерела повідомлення значно перевищує ентропію джерела ключа. У цьому випадку критерій безумовної стабільності не відповідає BSS [9], і в таких умовах доцільно ввести поліноміальний критерій, який передбачає обмеження обчислювальних ресурсів зловмисника та часу, протягом якого шифр залишається стабільним. Такий поліноміальний критерій призводить до практичного критерію стабільності - неможливості шифрування в сучасній комп'ютерній базі, зокрема, з огляду на постійне збільшення потужності комп'ютерних технологій та появу квантових комп'ютерів з часом.

Враховуючи результати дослідження, основні вимоги до проекту національної BSS [9, 10, 39], з точки зору його криптографічної стабільності, такі.

1. Криптографічна стабільність шифру залежить від складності атаки на BSS. Показниками складності криптоаналізу, як правило, є [9, 13, 15].

Час - це математичне сподівання часу (безпечного часу), необхідного для нападу на доступні / перспективні обчислювальні засоби.

Просторова складність - обсяг пам'яті, необхідний для виконання криптографічного аналізу.

Мінімальна кількість пар шифротексту / відкритого тексту або кількість пар відкритого тексту / шифротексту, необхідна для успішної атаки.

Попередній аналіз дає підстави зробити висновок, що якщо вам потрібен один із цих показників, реалізація атаки на практиці неможлива із значним запасом міцності, алгоритм шифрування можна вважати стабільним.

2. Як правило, первинна оцінка стійкості сприяє проведенню відносно силових атак: на BSS, атак на словник, створення сутичок тощо. Якщо у вас є необхідний рівень стійкості BSS до силових атак, ви можете перейти до оцінки стійкості BSSH до аналітичних атак.

3. Наведені результати аналізу: для сучасної BSS як критерію оцінки стійкості до аналітичних даних рекомендується використовувати [9, 15]:

- потужність набору зашифрованого / відкритого тексту, необхідного для здійснення криптоаналітичної атаки, необхідність перевищення потужності активного набору зашифрованого / відкритого тексту;

- складність будь-якої аналітичної атаки вимагає перевищення складності силової атаки або рівної їй;

- для здійснення аналітичної атаки необхідна кількість групових оперативних шифрувань не повинна бути меншою, ніж при повному пошуку ключів;

- обсяг пам'яті, необхідний для зберігання проміжних результатів у разі аналітичної атаки, не повинен змінюватися, ніж при виконанні атаки на словник із повним шифром;

- Слухаючи вдосконалення криптоаналітичних методів, потрібно використовувати критерій "запас стійкості" до аналітичних атак, згідно з яким складність атаки на весь алгоритм повинна значно перевищувати складність силових атак. Як правило, цей критерій розглядає версію алгоритму шифрування BSS для зменшеної кількості циклів, що незрозуміло для криптографічного аналізу (для цих випадків різниця в цьому циклі визначає рівень стійкості алгоритму до певної криптоаналітичної атаки, і особливо довга історія, тимчасово стабільний алгоритм);

- для оцінки криптографічної стійкості загальної структури шифру можна використовувати інший критерій, який розглядає можливість усунення будь-яких операцій або заміну їх менш складними операціями (наприклад, на деяких наборах вхідних даних операція додавання за модулем 232 є близький до або еквівалентний повний цикл спрощеного шифру повинен залишатися стійким до аналітичних атак.

4. Слід також мати на увазі, що більшість сучасних аналітичних атак, таких як диференціальний та лінійний криптоаналіз, є статистичними [9, 39]. Під час криптоаналізу виконується велика кількість шифрування для реагування на ключі та на основі шифротекстів, що утворюють варіанти з'єднання (ключі циклу). При обробці досить великої вибірки зашифрованого тексту, сформованого одним ключем, правильне значення бітових ключів виникає через інші параметри. Очевидно, що ймовірність знаходження початкової пари (що дає правильне значення ключа) залежить від статистичних потужностей шифру, і для збільшення складності криптоаналізу властивості криптограм повинні бути близькими до силових структур випадкового послідовність. Отже, необхідною (але недостатньою) умовою стійкості шифру до аналітичних атак є забезпечення відповідних статистичних потужностей вихідної послідовності (шифротекстів).

5. Було встановлено [39], що для захисту BSS від алгебраїчних атак необхідно, щоб не було практичного способу побудови систем шарів, що пов'язують відкритий текст, криптограму та ключ шифрування, або немає способів створити такі системи в поліноміальний час.

6. При розробці засобів блочного шифрування необхідно навчити організацію впроваджувати (змінна температура електронного пристрою, вхідна напруга, поява іонізуючого випромінювання, заміна споживчих конструкцій, час виконання). Такі атаки можуть бути ефективними проти всіх криптографічних алгоритмів, і захист від них при розробці криптографічних засобів захисту інформації вже вимагає інженерних рішень.

7. Загалом також можна сформулювати вимоги до стабільності сучасної ЗСУ:

- забезпечення опору силовим атакам, наприклад, для тимчасового або просторового артеріального зберігання проміжних результатів пам'яті;

- відсутність можливості будувати або підключати системи шарів, які вказують відкритий текст, криптограму та ключ шифрування;

- Неможливість здійснення відомих аналітичних атак на шифрування або їх складність повинна бути вищою, ніж складність реалізації силових атак (наприклад, один із таких критеріїв: потужність необхідного відкритого / зашифрованого повідомлення; необхідна кількість оперативних шифрування;));

- забезпечення "запасу міцності" шифру (додаткові цикли шифрування), що забезпечує безпечне використання алгоритму у разі поліпшення криптоаналітичних атак;

- стабільність спрощеної версії шифрування, якщо якась операція пов'язана або замінена на більш просту;

- забезпечення "необхідних" статистичних повноважень вихідної послідовності шифрування (криптограми або шкали шифрування), для яких криптограми та шкали шифрування практично не відображаються для повного використання у випадковій послідовності.

8. На додаток до необхідного рівня криптографічної стабільності, BSS повинен забезпечувати високий рівень продуктивності (складність шифрування, дешифрування та розгортання ключів).

З огляду на великий обсяг інформації, що обробляється в ІТС, ця вимога надзвичайно важлива і критична для ефективного функціонування всіх ІТС. Крім того, при розробці систем, що використовують BSS, крім вищих визначених вимог, потрібно вивчити рівень реалізації засобів шифрування та використання відповідних інструментів.

9. Попередній аналіз підтвердження, який визначає вимоги, насправді аргументи суперечливі: наприклад, більшість сучасних алгоритмів підвищення криптографічної стабільності вимагають додаткових циклів шифрування, що призводить до зниження продуктивності. Однак алгоритми-фіналісти міжнародних криптографічних конкурсів, таких як AES, NESSIE, CryptRec [30–

32] та інші, вказують на можливість досягнення раціональних показників, близьких до оптимальних.

1.3 Спеціальні вимоги до перспективного БСШ

Для побудови стабільного шифру, зручного для практичного використання, ми можемо запропонувати наступні підходи:

1. Блокада. Виберіть довжину ключа, меншу за довжину повідомлення, розділіть повідомлення на окремі блоки та зашифруйте кожен з них (крім, можливо, останнього), підсумувавши за допомогою ключа за модулем два.

2. Режими шифрування. Щоб підвищити стабільність шифрування блоків, можна використовувати результати шифрування кожного попереднього блоку при шифруванні кожного наступного блоку (наприклад, як новий ключ шифрування для блоку). Тоді зловмисник не зможе розшифрувати криптотекстовий блок, поки не розшифрує всі попередні блоки.

3. Багатокруглість. Ви можете ще більше підвищити стабільність шифрування блоків, повторно шифруючи кожен блок, за умови, що функція шифрування є нелінійним перетворенням.

У блокових алгоритмах вхідна послідовність ділиться на блоки - ділянки певної довжини (зазвичай 64 біти). Якщо довжина відкритого тексту кратна довжині блоку, використовується операція заповнення останнього блоку до необхідної довжини, яка полягає у додаванні необхідної кількості нулів або випадкового набору символів (рис. 1.2).

Рисунок 1.2 - Представлення даних в блоковому алгоритмі

При шифруванні блоків відкрите повідомлення C = {c1, c2,…, cn} для захисту від несанкціонованого доступу спочатку ділиться на блоки однакової довжини блоків символів C` = {c`1, c`2,…, c` k}, C`` = {c``1, c``2,…, c``k},…, C n = {cn 1, cn 2,…, cnk}. Потім, залежно від ключа, до кожного блоку застосовується функція шифрування, щоб перетворити їх у секретні блоки повідомлень однакової довжини. Результатом є блоки зашифрованого тексту S` = {s`1, s`2,…, s`k}, S`` = {s``1, s``2,…, s``k},…, S n = {sn 1, sn 2,…, snk}, які об'єднуються в зашифрований текст S = {s1, s2,…, sn}.

Слід зазначити, що довжина відкритого повідомлення не завжди кратна довжині блоку. Отже, існує проблема додавання, яка має кілька рішень. Наприклад, можна передавати в незашифрованому вигляді розмір корисної частини зашифрованих даних, а після розшифрування зайві символи просто відкидають. На практиці частіше використовується інший метод. Усі невикористані символи блоку, крім останнього, заповнюються будь-якими значеннями, і замість останнього символу пишуть число, рівне кількості невикористаних байт. Потім, отримавши та розшифрувавши всі блоки, необхідно відкинути з кінця стільки символів, скільки вказано в останньому символі останнього блоку. Але в цьому випадку виникають певні труднощі. Якщо довжина відкритого повідомлення кратна довжині блоку, потрібно додати 0 символів, а останній символ повинен містити кількість символів додавання. Для вирішення цієї проблеми з шифруванням додається новий блок, останній символ якого містить розмір блоку, який буде повністю відкинуто при дешифруванні.

Головною перевагою блокового шифрування є те, що у добре розробленій системі невеликі зміни у відкритому повідомленні або ключі викликають великі та непередбачені зміни в шифротексті. До недоліків блокового шифрування можна віднести: неправильне дешифрування всього блоку за наявності помилки лише в одному символі отриманого зашифрованого тексту, можливість криптоаналізу «зі словником».

Аналіз спеціальних вимог [9, 10, 39], що визначає певні параметри криптографічного перетворення, дозволяє сформувати такі обмеження та вимоги до перспективних BSS.

1. Захист алгоритму від криптоаналітичних атак. Основними методами криптографічного аналізу є: диференціальний криптоаналіз, розширення для диференціального криптоаналізу, пошук найкращих диференціальних характеристик, лінійний криптоаналіз; інтерполяційне вторгнення; вторгнення з приватним вгадуванням ключа; вторгнення з використанням пов'язаного ключа; вторгнення на основі плати за обробку; пошук лазівки та потенційних атак.

2. Статистична безпека криптографічного алгоритму з точки зору невідрізності масштабів шифрування та шифротекстів від справді випадкових [9].

3. Особливості конструкцій та відкритість конструкції. Криптоалгоритм потрібно розуміти, легко аналізувати структуру та основу для прикріплення до надійного математичного апарату.

4. Стійкість до модифікацій, всі кандидати перевіряються на стійкість до різних модифікацій: стійкість до криптоаналітичних атак при одночасному зменшенні циклів, зменшенні компонентів, що використовуються алгоритмом інших.

5. Обчислювальна складність (швидкість) для шифрування / дешифрування. Складність програмного, апаратного та програмно-апаратного забезпечення визначається підключенням до загальної пам'яті як для програмного, так і для апаратного забезпечення, зокрема для програмного забезпечення - кількість необхідної оперативної пам'яті, розмір вихідного коду, швидкі програми для програм на різних платформах для впровадження відомих мов програмування. Для апаратної оцінки декількох вентиляторів і швидкості в МБ / с.

6. Універсальність криптографічного алгоритму: можливість роботи з різною довжиною початкових ключів та інформаційних блоків; безпека

реалізації на різних платформах та додатках; ви можете використовувати криптографічний алгоритм у необхідних режимах обробки BSS.

7. Загальні вимоги до BSS повинні бути визначені замовником у конкретних параметрах криптоалгоритму. Суть їх полягає в наступному.

8. Параметри криптоалгоритму: - криптоалгоритм повинен будуватися на базі BSS; - обидва використовувані розміри блоків даних - 128, 256 і 512 біт; - обидва використовувані розміри одноразового (сеансового) ключа - 128, 256, 512 біт.

9. Принципи побудови: - особливість протистояти відомим методам криптографічного аналізу та мати стійкість до стабільності, враховуючи тенденції розвитку електронних обчислень та криптографічної науки; - використовувані криптографічні перетворення повинні створюватися на надійній та прозорій математичній основі і не мати вбудованих лазівки; - швидкість криптоалгоритму повинна бути не менше швидкості чистого стандарту шифрування.

10. Впровадження криптоалгоритму: - криптоалгоритм повинен бути зосереджений на можливості реалізації на 32- або 64-розрядних процесорах; - операції, зазначені в криптоалгоритмі, повинні мати ефективну програмну та апаратну реалізацію; −потрібно працювати для досягнення пам’яті, слід прослухати можливість реалізації криптоалгоритму в мікроприладах; - дозволити паралельно виконувати кілька операцій (якщо це можливо).

11. Система ключів: - підтримка ключа сеансу; - криптоалгоритм може надати довгостроковий ключ; - довжина синхронізуючої посилки менше 64 біт.

12. Ефективність криптографічного захисту із використанням BSS залежить не тільки від потужності BSS, але і від методів його використання, завдяки властивостям криптографічного перетворення залежати від режиму застосування блочного симетричного шифрування. Криптоалгоритм повинен забезпечувати такі режими роботи [41]:

- Простий режим заміни, який є обов’язковим компонентом для всіх інших режимів застосування блочного симетричного шифрування. Це найпростіша з

точки зору реалізації електронна кодова книга, яка передбачає конфіденційність окремих блоків відкритого тексту з їх шифруванням введеними секретними ключами;

- Режим ігрової роботи, призначений для забезпечення конфіденційності за допомогою швидкого шифрування блоків відкритого тексту з можливістю використання паралельних обчислень. Він полягає в шифруванні набору вхідних блоків, які є вихідними даними лічильника, і в додаванні їх (гама-блоків) до блоків відкритого тексту;

- режим зворотного зв'язку, який призначений для забезпечення конфіденційності шляхом шифрування потоку даних із відтворенням помилок та запобігання маніпуляціям з окремими блоками відкритого тексту;

- режим блоку кодування, який призначений для забезпечення конфіденційності, де процес шифрування будується як комбінація блоків відкритого тексту з попередніми блоками зашифрованого тексту;

- режим виробництва імітовставки, заснований на використанні режиму зчеплення шифрових блоків з додатковим додаванням ключа до останнього шифрного блоку. Сформована таким чином імітаційна вставка призначена для забезпечення достовірності та цілісності інформації;

- Режим зворотного зв'язку з шифруванням, призначений для забезпечення конфіденційності. Цей режим заснований на шифруванні вектора ініціалізації (синхронізація) для формування послідовності вихідних блоків (шифрів), які додаються до простого тексту для формування зашифрованого тексту і навпаки, до зашифрованого тексту для його розшифровки;

− імітувати режим вставки з підробкою та без неї - режим Galois / Counter та код автентифікації повідомлень Galois (GCM та GMAC), який призначений для забезпечення конфіденційності та цілісності інформації;

- режим індексованої заміни (режим наскрізної кодової книги на основі XEX із цифровим викраденням тексту - XTS), який призначений для забезпечення конфіденційності даних, особливо при зберіганні на певних фізичних носіях (жорстких дисках, оптичних дисках тощо);

- Режим шифрування ключів (KW), який полягає у багаторазовому застосуванні функції шифрування до масиву ключових даних, який «обгорнутий» шифром і надійно захищений відповідно до певної схеми «обгортки». Правило "обтікання" визначається специфікацією режиму обтікання ключів.

13. Досвід та результати проектів AES, NESSIE, враховуючи також національні вимоги до BSS, дозволяють запропонувати класифікацію BSS за стабільністю у вигляді таких умов [9, 39]:

- надвисока стабільність, коли довжина інформаційного блоку і довжина вихідного ключа не менше 512 біт;

- висока стійкість, коли довжина інформаційного блоку і довжина ключа не менше 256 біт;

−нормальний рівень стійкості, коли довжина інформаційного блоку і довжина ключа не менше 128 біт;

- задовільний рівень стабільності, коли довжина інформаційного блоку не менше 64 біт, а довжина ключа не менше 128 біт.

Ці вимоги є необхідними, але недостатніми.

1.4 Висновки до розділу

Відповідно до сучасних поглядів, розроблені на їх основі блокові симетричні криптовалюти та BSS є основним криптографічним механізмом забезпечення конфіденційності та цілісності, а також захисту інформації та інформаційних ресурсів від НРД. Ще однією важливою особливістю BSS є можливість забезпечити досить високу швидкість шифрування та розгортання ключів [9].

Для досягнення цієї мети ми спочатку аналізуємо вимоги до перспективних симетричних криптовалют, зокрема BSS, зокрема загальні

вимоги до криптографічної стабільності та спеціальні вимоги, що визначають певні параметри криптографічного перетворення у визнаних на практиці та закріплених у міжнародних стандартах режимах BSS.

Відповідно до сформульованих вимог до блокових симетричних криптовалют, перспективна BSS повинна забезпечити стійкість до всіх відомих криптоаналітичних атак за допомогою ефективної програмної, апаратної та програмно-апаратної реалізації засобів захисту інформації. Ми проведемо порівняльний аналіз сучасної BSS за критерієм криптографічної стабільності та за показниками швидкості, ємності пам'яті.

2 ДОСЛІДЖЕННЯ ОСНОВНИХ ПАРАМЕТРІВ

2.1 Аналіз принципів синтезу сучасних блокових симетричних шифрів

Історія створення BSS поширюється із введенням в 1974 р. IBM алгоритму блочного симетричного шифрування DES (Data Encryption Standard) [11, 12]. Алгоритм DES був наданий IBM у відповідь на запитання про розробку Національного бюро стандартів США спеціалістам у галузі алгоритмів стандарту шифрування в державних та приватних установах. Однією з вимог було те, що алгоритм повинен публікуватися без ризику для його стабільності. Розділ IBM базувався на попередньому шифрі Люцифера із 128-бітовим ключем. В алгоритмі IBM запропонувала ключ довжиною 48 * 16 біт, відповідно, 48 біт на кожному з 16 циклів. У 1977 р. Алгоритм BSS DES був опублікований як федеральний стандарт США, який набув чинності в липні того ж року [11]. Цю дату можна вважати початком створення та застосування BSS DES у відкритому доступі. Можлива угода після прийняття стандарту DES є предметом суперечок.

Головним недоліком була мала довжина секретного ключа - 56 біт. Насправді, навіть у той час для потужності набору з 256 ключів (близько 1016,86) було очевидно можливим застосування силової атаки, що було поряд з іншими можливостями для її здійснення. Діффі та Геллман опублікували проект спеціалізованого багатопроцесорного комп'ютера вартістю близько 20 мільйонів доларів, який міг би виявити DES приблизно за 12 годин [14, 15]. Зазначимо, що DES на момент свого створення зумів реалізувати накопичені теоретичні та практичні здібності до побудови симетричних шифрів. Угода була перевірена після прийняття протягом наступних двадцяти років BSS DES, детально вивчена і досліджена, і багато методів криптоаналізу, пов'язаних з DES, щойно з'явилися. Розробка алгоритму шифрування була настільки далекою, що він продовжував розвиватися в інших відомих криптографічних алгоритмах - насамперед у ГОСТ 28147-89 [16], FEAL [17] та деяких інших відомих криптографічних алгоритмах, розроблених у 80-х роках XX століття. Крім того, для усунення такого недоліку,

як коротка довжина ключа, було запропоновано потроїти довжину ключа та використовувати потрійне шифрування, яке є відносно ефективним та застосовним.

Майже до початку 90-х років в іноземній відкритій компанії майже не було фундаментальних робіт з блокових симетричних шифрів. На початку 90-х років критика алгоритму DES почала посилюватися. У відкритій зарубіжній пресі з'явився ряд фундаментальних праць, в яких вирішується проблема криптоаналізу БСС. Це заздалегідь лінійний криптоаналіз Мацуї [18] та диференціальний криптоаналіз Біхама та Шаміра [19–21], а також розширення цих атак. У цих роботах також критикується схема формування циклічних підрозділів BSSH DES, скасування простої схеми розгортання ключів, описується існування 4 слабких і 12 напівслабких ключів, а також властивість інформування, що зменшує пряме перетворення всі ключі від 56 2 до 55 2. щодо DES та його модифікації DEA, слід зазначити публікацію в 1981 році федерального стандарту DES [12]. Він описує слабкі та напівслабкі клавіші, для яких було використано поняття подвійних клавіш. Слабкі та напівслабкі ключі у Мура, Саймонса [22] та Коперсміта [23] були вивчені більш детально.

Згодом ми використали роботу [24–28], яка була присвячена аналізу ключової схеми розгортання в BSS. Публікацію Біхама можна вважати фундаментальною працею [29]. У цій роботі вперше описується атака на схему розгортання ключа, пов'язану з ключем, яка може бути застосована до BSS DES та LOKI. Однак цей метод криптоаналізу з атакою на схему розгортання ключів мав теоретичне, а не практичне значення, а не сукупність порівняно з диференційованими та лінійними функціями криптоаналітичного циклу шифру реальної загрози для BSS.

Крім того, пошук нових рішень стосувався вдосконалення криптографічних перетворень функції циклу. З'явився ряд нових криптографічних алгоритмів. Найвидатнішим серед них є алгоритм PES, який закладає основу для розробки європейського стандарту IDEA, а також алгоритмів RC5 та SAFER. Особливістю цих шифрів є те, що ключові схеми

розгортання нового BSS формують циклічні зв’язки, подібно до BSS DES. По суті, це алгоритми вибору певних бітів із секретного ключа. Згодом імена шифрів були розроблені ефективні методи криптоаналізу за допомогою ключових схем розгортання.

Таким чином, аналіз схеми розгортання ключів алгоритму IDEA дозволив Daemen [30] виявити кілька класів слабких ключів. Найбільший із цих слабких класів - це набір з 251 ключа, і приналежність ключа, що використовується до цього класу, можна перевірити за допомогою двох шифрувань та невеликої кількості додаткових обчислень. Якщо ключ належить до цього класу, його можна відновити після 16 шифрування відкритого тексту із вибраною різницею, а також приблизно 216 групових операцій та 217 шифрувань перевірки ключів. У вищезазначеній роботі Дамен запропонував незначну модифікацію процедури розгортання ключів IDEA, що дозволило усунути знайдені ним слабкі ключі.

Схема розгортання ключів SASFER K-64 виявилася вразливою до атаки на пов'язані ключі, запропонованої Біхамом [19]. Це пов'язано з тим, що схема розгортання ключів має недостатній ефект формування лавини і генерує всі ключі циклу за одним і тим же алгоритмом. Ці дослідження описані в працях Л. Кнудсена [31]. Загалом, розробка запропонованої оригінальної схеми криптоалгоритму продовжується в криптоалгоритмах SAFER SK-128, SAFER + та SAFER ++ [32, 33].

У роботі Кнудсена Мейер [31] описав диференційовану атаку на RC5, а також визначив багато слабких ключів, для яких ця атака стає ще більш ефективною. Криптоалгоритм був розроблений в RC6 [34] із змінами в схемі розгортання ключів.

До кінця 90-х років з'явилися нові ефективні методи криптоаналізу схем розгортання ключових BSS, засновані на використанні як диференціальних, так і лінійних методів криптоаналізу. В принципі, необхідно розрізняти диференціальний криптоаналіз на пов'язаних клавішах [19], ковзну атаку та її розширення [35]. Особливістю таких атак, порівняно з атаками на циклічну функцію BSS (наприклад, диференціальний та лінійний криптоаналіз), є більша

ефективність, яка є слабкою залежно від кількості ітерацій, що використовуються в шифрі.

Загальна рекомендація полягає в тому, щоб уникати лінійних алгоритмів під час формування циклічних одиниць, і що процес створення одиниць важко змінити [9, 39], взаємозв'язок між циклічними одиницями повинен бути складним з точки зору складності аналізу.

Подальший аналіз показав, що до кінця 90-х років існувала ситуація, коли використання морально та технічно застарілих шифрів продовжувалось у несекретних державних та комерційних установах. У той же час бурхливий розвиток комп’ютерних технологій та безліч різних високоефективних методів криптоаналізу блокових симетричних шифрів стали для них потенційними загрозами. Крім того, розроблені криптографічні алгоритми не підтвердили необхідний рівень безпеки для заміни прийнятих стандартів шифрування даних (наприклад, Skipjack). Це призвело до однієї з найважливіших подій у галузі прикладної криптографії - оголошення міжнародного конкурсу AES (Advanced Encryption Standard) [36]. У січні 1997 року NIST USA оголосив про початок конкурсу на новий стандарт шифрування 21 століття, спрямований на вибір нового стандарту для симетричного блокування шифрування на конкурентній основі. В результаті цього проекту переможець був оголошений переможцем алгоритму Рейндаеля [37], пізніше на його основі у вузькій версії був прийнятий федеральний стандарт США FIPS-197, який зараз широко використовується в США та світі .

Вперше багато різних криптографічних алгоритмів з різною архітектурою було об’єднано в проект AES та цілеспрямовано детально проаналізовано. Насправді були визначені вимоги, яким повинен відповідати перспективний BSS. У процесі проведення змагань також були визначені нові вимоги до ключових схем розгортання. Ключові схеми розгортання, подані на конкурс BSSh, суттєво відрізняються від DES-подібних. Крім того, відсутність основних принципів для розробки ключових схем розгортання призвело до того, що розробники коду керуються суб'єктивним розумінням побудови схем

розгортання ключів. Як результат, більшість шифрів були вразливими до атак на ключові схеми розгортання. Так, BSS Deal, Rijndael, SAFER +, DFC, MAGENTA, CRYPTON, HPC, LOKI97, MARS, RC6, Serpent, Twofish, CAST, E2, FROG взяли участь у проекті AES. Але більшість цих BSS були вразливі до атак на ключові схеми розгортання, включаючи: SAFER +, CRYPTON, DFC, FROG, HPC, MAGENTA. Для інших атак BSS на ключові схеми розгортання були реалізовані на неповній кількості ітерацій, але виявилися найбільш ефективними порівняно з атаками на функцію циклу алгоритму. Наприклад, BSS Rijndael, для якого була знайдена дев'ятициклова атака на пов'язані ключі (версія алгоритму з довжиною ключа lk = 128 і кількістю ітерацій r = 12) та розширена атака Square на семицикловий варіант алгоритм.

За аналогією з проектом AES розгорнуті та реалізовані подібні проекти в Європі (NESSIE) та Японії (CRYPTREC). Проект NESSIE був розпочатий у 2000 році під егідою Європейської комісії. Основними цілями проекту NESSIE був вибір перших десяти криптографічних примітивів.

Щодо криптоалгоритмів нормального та високого рівня безпеки склалася така ситуація: Noekeon, Hierocrypt-L3 та SHACAL-1, вразливі до атак, підключених клавіш та проскакувань. Серед BSS, в яких були виявлені атаки, пов'язані зі схемою розгортання ключа, але не з повноцінною версією, ми відзначимо крипто-алгоритм Rijndael [37]. Camellia, Rijndael та SHACAL-256 були оголошені переможцями конкурсу серед криптографічних алгоритмів із нормальним та високим рівнем безпеки. Встановлено, що майже всі інші крипто-примітиви мають аналітичні атаки, які ефективніші за грубу силу або не забезпечують необхідної швидкості шифрування.

Зауважимо, що вперше вимоги були встановлені не лише щодо високої криптовалюти, а й щодо якісних характеристик (наприклад, швидкості). Результати дослідження також показали, що крипто алгоритми Grandcru і SC2000 мають низьку швидкість, в результаті чого вони не пройшли до другого туру. Найшвидшими серед алгоритмів є BSS RC6 та Rijndael, які були позитивно оцінені під час змагань з AES.

За результатами подальших досліджень існуючих та перспективних БСШ прийнято міжнародний стандарт відносно блокових симетричних шифрів, що увійшли в міжнародний стандарт ISO/IEC 18033 – 3 [38]. Перелік та деякі характеристики БСШ стандарту наведено у табл. 1.

Таблиця 2.1 Характеристики БСШ з ISO/IEC 18033

Довжина блока	Назва алгоритму	Довжина ключа
64 бітів	TDEA	128 або 192 бітів
	MISTY1	128 бітів
	CAST-128	
128 бітів	AES	128, 192 або 256 бітів
	Camellia	
	SEED (5.3)	128 бітів

Згідно з класифікацією, прийнятою в проекті NESSIE, шифри з довжиною блоку lb = 64 біта та довжиною ключа lk = 128 біт належать до BSS задовільного рівня стабільності. Основними BSS, які можна класифікувати як задовільний рівень стабільності, є BSS GOST 28147 - 89, TDEA, MISTY1 і CAST - 128. Також важливо визначити BSS DES і DEA, в рамках цієї класифікації вони класифікуються як незадовільний рівень BSS стабільності, оскільки вони вже не рекомендуються до використання.

BSS нормального рівня стабільності включає в себе шифри, в яких довжина блоку lb = 128 біт, а довжина ключа lk = 128 біт. Основними BSS, що належать до нормального рівня стабільності, є BSS FIPS 197 (AES), Camellia та S0EEDx.

Високостійкий BSS включає коди FIPS 197 (AES), камелії, Каліни, мухомора тощо, в яких довжина блоку дорівнює або не менше 128 біт і не більше 256 біт, а довжина ключа дорівнює або не менше 256 біт.

Важливою подією для України є підготовка та проведення національного конкурсу на кращий проект національного стандарту BSSH, який розпочався 15

жовтня 2006 р. І фактично закінчився у травні 2010 р. Конкурс ініціювали чотири кандидати в встановлена форма - шифри Kalina, Labyrinth, ADE та Amanita. По суті, два шифри, тобто Каліна та АДЕ, є певною мірою вдосконаленням шифру зі структурою SPN, лабіринтом - ланцюгом Фейстеля та мухомором - структурою IDEA. Специфікації та результати досліджень цих кодів також представлені в ряді публікацій. Результатом конкурсу є формування команд, здатних виконувати складні завдання з розробки та дослідження перспективних BSS, а також розробка та освоєння науково-методичного забезпечення в галузі синтезу та аналізу BSS. Цей досвід вже був використаний для гармонізації в Україні міжнародного стандарту ISO / IEC 18033 - 3 [38] та інших стандартів.

Вимоги до перспективної BSS в національному змаганні відрізняються від вимог Нессі тим, що в національному змаганні вводиться дуже високий рівень безпеки (гарантії), коли lb≥256 біт і довжина ключа lk≥512 біт.

До BSS надвисокого рівня стійкості ми включаємо коди BSSH SHACAL-2, "Калина", "Мухомор", Threefish, у яких довжина блоку дорівнює або не менше 256 і не більше 512 біт, а довжина ключа дорівнює або перевищує 512 біт.

Дослідження принципів синтезу симетричних криптовалют показали, що зараз необхідно виділити три методологічних підходи до побудови перспективних BSS [9]. Насправді вони вже висунуті кандидатами на стандарт BSS європейської програми NESSIE.

Перший передбачає використання структур SPN. Загальна структура - SPN, квадратного типу, байт - байт-орієнтований шифр. На основі таких структур були розроблені та визнані BSS Rijndael та його звужена версія AES (FIPS - 197) [9, 37], які базуються на попередній розробці авторів - кодексі Square. Ця область була досліджена в достатній мірі і за їх результатами запропоновано кандидата національного стандарту BSS "Калина" [39].

У процесі досліджень увагу було приділено БСГ, які мають структуру, подібну до IDEA. Відомо, що фактично європейський стандарт IDEA витримав випробування часом і досі забезпечує заявлений рівень стабільності. На початку XXI ст. Паскаль Юнод та Серж Водені запропонували проект вдосконаленої

BSS, який називається FOX [40]. Алгоритм IDEA NXT (раніше відомий як FOX) - це симетричний блок-шифр, розроблений Паскалем Юнодом та Сержем Водені з лабораторії EPFL. Задуманий між 2001 і 2003 роками, проект спочатку називався FOX і був опублікований у 2003 році. У травні 2005 року компанія MediaCrypt анонсувала його під назвою IDEA NXT. IDEA NXT є нащадком алгоритму IDEA і використовує розширену схему Лі-Массі, відому своєю стійкістю до криптоаналізу [9, 40]. Він належить швейцарській компанії MediaCrypt, яка володіє правами на розповсюдження IDEA і яка володіє патентами на IDEA NXT. Шифр IDEA NXT - це сімейство різних модифікацій шифру з різними розмірами блоків і розмірами ключів: стандартний NXT64 (64-бітний блок, 128-бітний ключ, 12 раундів) і стандартний NXT128 (128-бітний блок, біти, 256-бітний ключ) , 12 раундів). Також можуть бути побудовані стандартні версії (з розміром ключа від 0 до 256 біт, кількістю раундів від 2 до 255). А також можуть бути завантажені окремі таблиці (sbox, матриця перестановок - матриця перестановок), що замінюють стандартну таблицю.

Реалізація третього методологічного підходу базується на вже добре перевіреній схемі у формі обличчя [11, 12, 15–22]. Він реалізований у перевірених часом стандартах BSSH DES, DEA, TDEA, ГОСТ 28147-89, а також у MISTY1, Cammellia. Сьогодні стандарти ФСГ, які мають вигляд обличчя, все ще часто застосовуються на практиці і не втратили перспективи застосування, можливо, з деяким вдосконаленням.

Таким чином, аналіз принципів проектування сучасних шифрів показав, що однією з найпоширеніших та найпотужніших сучасних концепцій проектування блокових симетричних шифрів є стратегія широкого сліду, яка базується на множенні матриць у розширених полях. Він був використаний розробниками AES, що дозволило обґрунтувати значення окремих показників ефективності BSS, зокрема отримати просту специфікацію шифру, яку легко аналізувати за допомогою прозорого та надійного математичного апарату. Вважається, що одним із додаткових конструктивних рішень у перспективних розробках має бути використання циклічної функції з високими динамічними

характеристиками переходу до постійних (стаціонарних) значень максимумів повних диференціалів та лінійних корпусів.

В ході дослідження обґрунтовано загальну структуру, таблиці підстановок, блоки лінійного перетворення та схему створення підзв’язків перспективного алгоритму блочно-симетричного криптоперетворення «Калина». Як нелінійні елементи шифру використовувались випадково сформовані таблиці підстановки, відібрані відповідно до критеріїв стійкості до диференціалу, лінійного криптоаналізу та ступеня нелінійності булевих функцій. Це дозволяє досягти високих рівнів захисту від диференціального та лінійного криптоаналізу і одночасно захистити від потенціалу алгебраїчної атаки на такі шифри, як Рійндаель, Камелія та інші. В якості блоку лінійного перетворення пропонується використовувати добре перевірену MRL [24, 25, 38] (перетворення, засноване на використанні лінійних блокових кодів з максимально досяжною кодовою відстанню), що дозволяє досягти верхньої межі мінімальна "кількість гілок" для будь-якої (ненульової) різниця вхідних даних. Загальна форма обраної трансформації дозволяє досягти вищих значень "кількості гілок" порівняно з BSS Rijndael (9 проти 5). Завдяки використанню 64-розрядного коду MDV забезпечується повна залежність кожного біта від вводу на двох циклах шифрування, незалежно від розміру блоку. Характеристики кращі, ніж у Rijndael 256/256, де для розподілу різниці по всьому блоку потрібно більше циклів. У БСШ «Калина» ключова інформація вводиться згідно з декількома метриками гейміфікації, в результаті чого криптографічний алгоритм переходить до класу немарковських шифрів і визначення ймовірності диференціальних (лінійних) характеристик є експоненціальним щодо складності, що робить диференціальний (лінійний) криптоаналіз.

Таким чином, пропоновані вдосконалення шифру Рійндаеля, впроваджені в шифрі калини, дозволяють охопити виявлені потенційні вразливості шифру Рійндаеля та зробити кандидата на національний стандарт блочного симетричного шифрування стабільним проти всіх відомих криптоаналітичних

атак. Цей висновок підтверджується роботами, що виконуються в процесі експертизи та дослідження кандидатів, що подаються на національний конкурс.

2.2 Порівняльний аналіз БСШ за критерієм криптографічної стійкості

Відповідно до сформульованих вимог до блокових симетричних криптовалют, перспективна BSS повинна забезпечити стійкість до всіх відомих криптоаналітичних атак за допомогою ефективної програмної, апаратної та програмно-апаратної реалізації засобів захисту інформації. Ми проведемо порівняльний аналіз сучасної BSS за критерієм криптографічної стабільності та за показниками швидкості, ємності пам'яті.

Виконуючи порівняльний аналіз існуючих BSS, необхідно визначити набір умов та вихідні дані, що використовуються для реалізації атаки. Зрозуміло, що криптоаналітик буде використовувати найефективніший метод криптоаналізу, щоб мінімізувати матеріально-технічні ресурси, необхідні для успішної атаки. Визначається та застосовується найефективніший метод. Тобто, як правило, цей метод передбачає найменші фінансові та / або просторово-часові витрати порівняно з іншими методами аналізу.

Отже, ефективність методу криптоаналізу визначається можливістю застосувати цю атаку до BSS для отримання секретного ключа або публічного повідомлення зі складністю, меншою ніж складність силової атаки. Отже, оцінка ефективності методу симетричного шифрування шифру може бути виконана лише для певного алгоритму або класу шифрів, які використовують однакові принципи їх побудови. Але, незважаючи на вищесказане, основні принципи криптоаналізу можуть бути застосовані до алгоритмів, що використовують різні принципи побудови. Ступінь застосовності різних методів атаки на шифри, що використовують різні принципи побудови, визначає універсальність методу криптоаналізу.

Оцінюючи криптографічну стабільність, слід мати на увазі, що деякі модифікації або доповнення до алгоритму криптоаналізу можуть суттєво змінити складність атаки. Таким чином, використання 2R-атаки в диференціальному криптоаналізі DES дозволило зменшити складність і зробити цю атаку більш ефективною, ніж прямі натискання клавіш.

Тому при оцінці ефективності методу криптоаналізу необхідно враховувати наступні фактори: - рівень доступності відкритих повідомлень для криптоаналітика; −умови атаки - можливість вибору значень на входах кодера та / або залежностей між ними; - перелік алгоритмів, до яких може бути застосований метод криптоаналізу; −можливості для криптоаналітика вибрати значення відкритих повідомлень; - можливість і необхідність криптоаналізатора мати фізичний доступ до обладнання; - мінімальна складність виконання атаки на повний шифр; - Універсальність методу криптоаналізу, тобто можливість його застосування до загальних шифрів.

Слід також мати на увазі, що універсальність, мінімальні вимоги до умов нападу та низька складність підвищують ефективність методу криптоаналізу. У свою чергу, потреба у відкритих текстах, їх відбір або завдання залежності між секретними ключами знижує ефективність. Крім того, вимога фізичного доступу до обладнання значно звужує область застосування методу криптоаналізу і, відповідно, також знижує його ефективність.

Диференціальний криптоаналіз, лінійний криптоаналіз та алгебраїчні методи можна виділити як найбільш ефективні методи криптоаналізу для більшості сучасних симетричних блокових алгоритмів. Щодо алгоритму шифрування AES [37], можна додатково розрізнити інтегрований криптоаналіз та атаку за допомогою нездійсненних диференціалів.

Сьогодні BSS AES (Rijndael, FIPS 197) стала визнаною у світі BSS, вона міжнародно стандартизована та рекомендована міжнародним стандартом ISO / IEC 18033-3 як код, що забезпечує нормальний або високий рівень стабільності. Тому, використовуючи отримані результати досліджень, ми зробимо деякі

оцінки щодо AES BSS. Це необхідно для того, щоб показати, що потенційні вразливі місця AES вже охоплені BSH Kalyna.

Загалом щодо AES можна зробити такі висновки [9]:

1. Аналіз результатів дослідження стійкості алгоритму AES (FIPS 197), отриманих під час реалізації проектів AES та NESSIE, а також результати його тестування з часом показали, що він забезпечує необхідний рівень стабільності.

2. Для вирішення проблем криптоаналізу до алгоритму FIPS 197 можуть бути застосовані наступні атаки: атаки грубої сили, диференціальний та лінійний криптоаналіз, усічені та реалізовувані диференціальні атаки, атака бумеранга, атака інтерполяції та лінійна сума, вищі порядки диференціальної атаки та інтегровані криптоаналіз.

3. Теоретичні та експериментальні дослідження стабільності FIPS 197 показали, що висновки про атаку, зазначені в пункті 1, можуть бути реалізовані зі складністю, порівнянною зі складністю атаки "грубою силою", лише на зменшеній кількості циклів алгоритму . Таким чином, можна здійснити атаку «грубої сили» з меншою складністю, відповідно: диференціальний криптоаналіз - за три цикли, лінійний криптоаналіз - за три цикли, усічені диференціали - за три цикли, неможливі диференціали - за п’ять циклів, атака бумерангом - для чотирьох циклів, диференціали вищого порядку - для трьох циклів, атака інтерполяції - для чотирьох циклів, інтегрований криптоаналіз - для шести циклів.

4. Аналіз даних таблиці. 2 дає підстави зробити висновок про стабільність алгоритму FIPS - 197 проти розглянутих криптоаналітичних атак. і більше, ми можемо говорити про наявність певного запасу міцності для алгоритму із заданою кількістю циклів перетворення - 10, 12 та 14, залежно від довжини ключа - 128, 192 та 256 біт відповідно. Наявність цього запасу дає надію, що алгоритм FIPS 197 залишатиметься надійним протягом деякого часу в майбутньому. і навіть більше, BSS Rijndael має можливість збільшити довжину зашифрованих блоків до 192 та 256 біт, роблячи його більш стабільним, ніж FIPS

Таблиця 2.2 - Аналіз криптостійкості алгоритму FIPS 197

Види криптоатак	Мінімальна кількість циклів, за якої шифр стійкий			Показники відомих атак на AES (Rijndael-128)		
	Rijndael-128	Rijndael-192	Rijndael-256	Максимальна кількість циклів	Обчисл. ресурси	Пам’ять
Диференційна	4	7	8	3	254	мало
Лінійна	4	7	8			
Усіч. дифер	4	5	7	3	28	мало
Немож. дифер.	6	6	6	5	236	224
Бумеранг	5	6	7			
Диф. вищ. пор.	4	4	4			
Інтерполяційна	5	5	5			
Інтегральна	7	7	7	6	272	272

Одним з основних безумовних критеріїв оцінки BSS є критерій захисту шифру від усіх відомих і потенційно можливих криптоаналітичних атак. Це безумовно в тому сенсі, що якщо алгоритм шифрування не відповідає вимогам цього критерію, алгоритм шифрування відхиляється і не розглядається як учасник торгів. Алгоритм шифрування вважається надійним, якщо всі відомі криптоаналітичні атаки є більш складними, ніж атаки грубої сили.

Важливим умовним критерієм є реальний захист від відомих криптоаналітичних атак із зменшеною кількістю циклів. Справа в тому, що за безумовним критерієм шифри Рійндаель, Шакал-2 і Камелія на всю кількість циклів стійкі до всіх відомих атак. Тому одним із критеріїв порівняльної оцінки шифрів є порівняння мінімуму циклів, при якому жоден із методів криптоаналізу не має складності, меншої за складність атаки типу «груба сила». Тому важливо оцінити криптографічну стабільність симетричного алгоритму шифрування Rijndael (FIPS 197) щодо відомих криптоаналітичних атак, зокрема з оцінкою мінімальної кількості циклів, для яких забезпечується стабільність.

Аналіз робіт [11–40] показав, що вже розроблено понад 10 методів криптоаналізу блокових симетричних шифрів, які можна застосувати до алгоритму Рійндаеля.

Серед криптоаналітичних атак для обов'язкового розгляду обрані атаки грубої сили, диференціальний та лінійний криптоаналіз, атака зрізаних диференціалів, атака нереалізованих диференціалів, атака бумеранга, атака інтерполяції, атака лінійних сум, диференціали криптовалют вищого порядку та інтеграл.

На наш погляд, для того, щоб оцінити стійкість кожної з цих атак, спершу необхідно переглянути відомі методи, що дозволяють оцінити стійкість шифру AES до цієї атаки. Тоді, беручи до уваги доцільність та точність отриманої оцінки, необхідно вибрати найбільш ефективний метод проведення оцінки стійкості та застосувати його до аналізованого шифру. Крім того, для вивчення найбільш ефективних методів криптоаналізу необхідно впровадити їх для шифру AES із зменшеною кількістю циклів.

Більшість атак грубої сили можуть бути застосовані до будь-якого блочного шифру, і складність цих атак залежить лише від довжини блоку n або довжини ключа k і не залежить від структури алгоритму.

Повна атака пошуку ключів - це найпростіший спосіб знайти ключ шифрування. Складність такої атаки залежить від довжини ключа і, як відомо, не менше 2k - 1 шифрування за допомогою вивченого шифру. Для захисту від цієї атаки в шифрах використовуються великі ключі. Оскільки мінімальна довжина ключа шифру AES становить 128 біт, комплексний пошук ключа вимагає 2128 шифрувань, і на практиці це неможливо.

Шифри з недостатньою довжиною блоку вразливі до атаки за словником. Для атаки потрібен блок 2n, а для побудови такої таблиці - шифрування 2n. Мінімальна довжина блоку розглянутого шифру становить 128 біт, отже, атака зі словником 2128 також неможлива на практиці.

Далі ми припустимо, що алгоритм захищений від якоїсь криптоаналітичної атаки, якщо його реалізація перевищує або дорівнює складності атаки грубої сили. Найвідомішими та найпотужнішими методами здійснення атак на БСС є диференціальний криптоаналіз та лінійний криптоаналіз, запропоновані на початку 90-х [18 - 21].

Існує чотири критерії стійкості n-бітового шифру до цих криптовалют: - точний критерій - максимальне значення ймовірностей диференціалів та закономірностей лінійного наближення нижче 2 - n; - теоретичний критерій - верхні межі значень ймовірностей диференціалів і закономірностей лінійного наближення нижче 2 - n; - евристичний критерій - максимальні ймовірності диференціальних та лінійних характеристик нижче 2 - n; практичний критерій - верхні межі ймовірності диференціальних та лінійних характеристик нижче 2- n.

Ідеальним варіантом, з точки зору перевірки можливості проведення диференціального або лінійного криптоаналізу, є перевірка точного критерію. Створення теоретично стабільного шифрувальника SPN на основі відомих теорем поєднується з досить суворими вимогами до перетворення, що використовується в кожному циклі, що призводить до уповільнення процедури шифрування.

Критерій перевірки евристики застосовується до шифрів з невеликим розміром блоку (не більше 64 біт), таких як DES, FEAL, і вимагає значних обчислювальних ресурсів. Для шифру Рійндаеля, розмір блоку якого не менше 128 біт, розглянутий алгоритм вимагає занадто великих обчислювальних ресурсів, що практично неможливо реалізувати на практиці.

Результатом оцінки стійкості шифру Рійндаеля до атак постійного струму та РХ є висновок про те, що шифр відповідає практичному критерію стійкості до лінійного та диференціального криптоаналізу.

Однак попередній аналіз показує, що Рійндаель (FIPS 197) має потенційні уразливості, зокрема: - виявив кілька нових властивостей компонентів та цілого алгоритму, таких як лінійні втрати заміщення (S-блок); - можливість алгебраїчного подання циклічної функції з усіма операціями в одному полі; -

можливість побудови системи рівнянь, що описують весь шифр за допомогою ланцюгових дробів; - можливість побудови заздалегідь визначеної системи рівнянь для всього шифру.

Однак, незважаючи на ці потенційні уразливості, використання кожної з них для аналізу всього алгоритму невідоме. Практичне використання цих функцій для всього алгоритму невідоме. і навіть більше, немає математичного апарату для розв’язування спеціальних систем рівнянь у вигляді ланцюгових дробів.

Загалом це вказує на те, що стабільний шифр може бути побудований на основі SPN.

2.3 Порівняльний аналіз сучасних БСШ щодо швидкодії та обсягу пам’яті

Однією з основних характеристик криптографічного примітиву є також його продуктивність із програмною реалізацією для універсальної платформи. Більш висока продуктивність на аналогічному або вищому рівні криптографічної стабільності є важливою перевагою алгоритму.

Ми проведемо порівняльний аналіз швидкості та обсягу пам'яті, необхідної для реалізації сучасної BSS, та обґрунтуємо вибір найбільш вдалого рішення.

При тестуванні продуктивності блочного шифру враховуються такі фактори: −змінність часу виконання шифрування залежно від потенційної активності інших процесів / потоків в операційній системі загального призначення; −залежність швидкості перетворення від наявності даних у кеші процесора; - можливість значно уповільнити процес шифрування, якщо вам потрібно використовувати файл віртуальної пам'яті операційної системи.

Варіативність часу виконання була зменшена завдяки багаторазовому (не менше 100 тисяч разів) вимірюванню часу шифрування та подальшому усередненню результатів.

Забезпечення продуктивності алгоритму шифрування, наближеного до реальних умов (обробка мережевого трафіку, введення-виведення диска тощо), здійснювалось шляхом обробки блоку оперативної пам'яті в кілька десятків мегабайт. Це значення гарантує відсутність даних у кеш-пам'яті та необхідність їх завантаження з оперативної пам'яті, що дозволяє отримати швидкісні характеристики, відповідні застосуванню шифру.

У той же час розмір оброблюваного блоку становить кілька десятків мегабайт, не вимагає використання файлу віртуальної пам'яті операційної системи (за відсутності інших додатків, які активно використовують оперативну пам'ять), що забезпечує тестування продуктивності лише в межах процесор та оперативна пам’ять, без набагато повільніших дискових підсистем.

Порівняння показників виконано для алгоритмів «Калина», ГОСТ 28147 89 (чинний стандарт, рекомендований до використання в Україні та застосовується в Росії), СТБ 34.101.31-2011 (Бел-Т, стандарт Республіки Білорусь) та AES (національний стандарт США та найпоширеніший у світі алгоритм). Теоретичний розрахунок кількості необхідних інструкцій процесора для обробки одного байта даних наведено в табл. 2.3.

У той же час сучасні процесори дозволяють апаратно оптимізувати продуктивність, перекодуючи та переставляючи інструкції, використовуючи їх паралельне виконання в межах одного потоку тощо.

Крім того, велике значення має блок оптимізації компілятора, який дозволяє значно збільшити продуктивність за рахунок розгортання циклів (відсутність скидання конвеєра під час умовного переходу) тощо. Тому, не докладно знаючи внутрішньої архітектури процесора, доступної лише розробникам (Intel, AMD та ін.), Та особливостей оптимізаторів компіляторів, ви можете оцінити фактичну продуктивність лише за допомогою експериментальних вимірювань.

Таблиця 2.3 - Теоретичний розрахунок кількості необхідних процесорних інструкцій для обробки одного байта різними шифрами

	Симетричний блоковий шифр			
	Калина (128/128)	ГОСТ 28147-89	СТБ 34.101.31-2011	AES
Кількість операцій на 1 байт	40,375	72	40,5	45,375

Вимірювання продуктивності блочного шифру проводили на таких програмних та апаратних платформах: - Intel Core i5 / Windows Server 2008 R2 x64 з компілятором Visual C ++ 2008; - Intel Core 2 Duo E8500 / Linux (64 біт) з останньою версією компілятора GCC.

Для вимірювання продуктивності були взяті найшвидші реалізації блокових шифрів на C / C ++: AES - з криптографічної бібліотеки OpenSSL; ГОСТ 28147-89 - з бібліотеки ВАТ «ІІТ»; "Калина" - авторська реалізація. Дані про виміряні показники наведені в табл. 4.

Таким чином, продуктивність перспективного блок-шифру "Калина" на 64-розрядної платформі набагато перевищує діючий стандарт в Україні та Росії ГОСТ 28147-89, а також стандарт Республіки Білорусь СТБ 34.101.31-2011 . На цій платформі Kalyna також має перевагу у продуктивності порівняно з алгоритмом AES зі схожою довжиною ключа (при цьому забезпечуючи набагато більший запас криптографічної стабільності).

Таблиця 2.4 - Продуктивність блокових шифрів, Мбіт/с

Таблиця Платформа	Симетричний блоковий шифр								
	K128/128	K128/256	K-256/256	K-256/512	K-512/512	AES-128	AES-256	ГОСТ	СТБ
Win 2008Srv	1538,31	1098,17	1256,23	977,61	995,72	1483,26	1095,19	376,3	609,18
Linux	1828,57	1291,72	1219,05	948,15	948,15	1667,95	1254,01	492,31	753,5

Ми проведемо порівняльні дослідження обсягу пам’яті, який повинен бути виділений в комп’ютерній системі для реалізації відповідних алгоритмів блочного симетричного шифрування.

Попередній аналіз показує, що для підвищення продуктивності комп'ютерів, тимчасового зберігання вмісту оперативної пам'яті, прискорення обміну між процесором і лише читання пам'яті в сучасних високошвидкісних комп'ютерних системах використовується принцип ієрархії пам'яті, тобто постійно збільшувати обсяг пам'яті. блокувальні пристрої (з певним зменшенням швидкості) під час "віддалення" від центрального процесора. Найшвидшим і, відповідно, найдорожчим і найменшим обсягом пам'яті є внутрішня пам'ять процесора (регістри, організовані у файлі реєстру, і кеш-пам'ять процесора). Щоб максимізувати швидкість реалізації криптографічних алгоритмів, бажано максимізувати можливості цієї пам'яті, оскільки в цьому випадку швидкість обчислення буде оптимізована відповідно до певної конфігурації системи.

Дослідження показали, що більшість провідних світових виробників сучасних процесорів випускають вироби різного призначення та технічних характеристик, які відрізняються обсягом кеш-пам'яті другого та третього рівнів, а також тактовою частотою та відповідною тепловою віддачею. Однак, що стосується рівня кеш-пам’яті першого рівня, майже всі сучасні процесори мають однакові обмеження на кількість кешу першого рівня, яка становить 64 кБ.

Тому загальні вимоги до сучасних криптографічних алгоритмів для пам'яті включають обмеження в 64 кБ, тобто розмір таблиць попереднього обчислення для швидкої реалізації шифрів не повинен перевищувати обсяг пам'яті кешу першого рівня. Це основна вимога пам’яті для перспективного симетричного блочного шифру, оскільки при виконанні встановлених вимог усі таблиці попередньої обробки можуть бути розміщені в кеші першого рівня, і продуктивність реалізації буде максимальною.

Обсяг таблиць передобчислень визначається такими рівняннями: мінімальний обсяг пам’яті:

$$V_{min} = k \cdot 2^t \cdot s \ ;$$

обсяг пам’яті, необхідний для досягнення максимальної швидкодії:

$$V_{max} = 2^t \cdot s^2,$$

де k – кількість нелінійних вузлів замість (S-блоків), які необхідно зберігати в пам’яті;

t – розрядність входу S-блока, бітів;

s – розрядність МДР-матриці, байтів.

Із застосуванням наведених формул підрахуємо обсяг таблиць передобчислень та порівняємо його із обсягом кеш-пам’яті першого рівня більшості сучасних процесорів (обмеження в 64 кБайт).

Маємо такі показники:

для блокового симетричного шифру AES

$$V_{min} = k \cdot 2^t \cdot s = 1024 \ ; \ V_{max} = 2^t \cdot s^2 = 4096.$$

Для блокового симетричного шифру “Калина-2”

$$V_{min} = k \cdot 2^t \cdot s = 8192 \ ; \ V_{max} = 2^t \cdot s^2 = 16384 \ .$$

для блокового симетричного шифру “Кузнечик”

$$V_{min} = k \cdot 2^t \cdot s = 4096 \ , \ V_{max} = 2^t \cdot s^2 = 65536 \ .$$

Отже, як показують проведені дослідження блокові симетричні шифри AES та “Калина” відповідають обмеженням на обсяг пам’яті кешу першого рівня, тобто їх практична реалізація дозволяє максимально застосовувати можливості цієї найшвидшої пам'яті й швидкість обчислень буде оптимізовано відповідно до певної конфігурації системи. Особливості побудови перспективного блокового шифру РФ “Кузнечик” такі, що обсяг пам’яті, необхідний для досягнення максимальної швидкодії, лежить на верхній межі обсягу пам’яті кешу першого рівня. Практично це означає, що під час обробки (шифрування) дані витіснятимуть таблиці передобчислень з необхідністю їх повторного завантаження, і це призводитиме до зниження продуктивності.

2.4 Висновки до розділу

Отримані результати дають підстави зробити висновок [39], що продуктивність перспективного блочного шифру "Калина" на 64-бітній платформі набагато перевищує діючий стандарт в Україні та Росії ГОСТ 28147-89, а також стандарт стандарту Республіка Білорусь СТБ 34.101.312011. На цій платформі Kalyna також має перевагу у продуктивності порівняно з алгоритмом AES зі схожою довжиною ключа (при цьому забезпечуючи набагато більший запас криптографічної стабільності).

Також порівняльні дослідження показали, що симетричні блокові шифри AES та "Kalyna" відповідають обмеженням обсягу кеш-пам'яті першого рівня сучасних процесорів. Практична реалізація цих шифрів дозволить вам максимізувати можливості цього найшвидшого кешу, а застосовані обчислення будуть оптимізовані відповідно до певної конфігурації системи.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ЗАХИСТУ ІНФОРМАЦІЇ СТІЙКОГО ДО АТАКИ В ЧАСІ

3.1 Функціональна структура програмного засобу

Найпоширеніший тип шифрування - це метод повного пошуку (або метод грубої сили).

Повний перебір (англ. brute force) - загальний метод вирішення проблем шляхом пошуку всіх можливих потенційних рішень.

У криптографії оцінка криптографічної стабільності шифрів базується на обчислювальній складності повного пошуку. Зокрема, шифр вважається криптостійким, якщо не існує методу "злому" значно швидше, ніж повний пошук усіх ключів. Криптографічні атаки, засновані на методі повного пошуку, є найбільш універсальними, але і найтривалішими.

Стійкість до атаки грубої сили визначає ключ шифрування, що використовується в криптосистемі. Таким чином, зі збільшенням довжини ключа складність злому за допомогою цього методу зростає в геометричній прогресії. У найпростішому випадку шифр довжиною n біт буде розбитий, у гіршому випадку - за час O (2n).

Взагалі, будь-яку проблему з класу NP можна вирішити повним пошуком, але це вимагатиме експоненціального часу.

Є способи підвищення стійкості шифру до "грубої сили", наприклад, плутанина (затуманення) зашифрованих даних, що робить нетривіальну різницю між зашифрованими даними та незашифрованими.

Виходячи з фізичних міркувань, можна показати, що існує можливість вибрати таку довжину ключа, що така атака методом "грубої сили" в принципі буде неможливою, незалежно від потужності існуючої комп'ютерної технології.

Абсолютний криптостійкість означає, що: результатом дешифрування може бути будь-який текст, немає можливості перевірити, що дешифрування виконано правильно.

Доказ існування абсолютно стабільних алгоритмів шифрування був виконаний Клодом Шенноном і опублікований у "Теорії зв'язку в секретних системах" (1946). існують також визначені вимоги до таких систем:

1. Ключ генерується для кожного повідомлення (кожен ключ використовується один раз).

2. Ключ статистично надійний (тобто ймовірності кожного з можливих символів однакові, символи в послідовності ключів незалежні та випадкові).

3. Довжина ключа дорівнює або перевищує довжину повідомлення.

Стабільність цих систем не залежить від того, якими обчислювальними можливостями володіє криптоаналітик.

Майже всі шифри, що використовуються на практиці, характеризуються як відносно надійні, оскільки вони можуть розкриватися в принципі за наявності необмежених обчислювальних можливостей. Абсолютно надійні шифри неможливо знищити навіть за наявності необмежених обчислювальних можливостей.

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

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

Отже, реалізовано програму, яка виконує процедуру шифрування даних, використовуючи алгоритм шифрування з використанням додаткових операцій. Основне завдання реалізованої програми – це використовуючи запропонований

метод перетворення інформації зробити її зміст недоступним для читання і тим самим забезпечити конфіденційність інформації та її цілісність.

Запропонована система захисту інформації призначена для використання в сфері документообігу, обміну інформації, та в навчальних цілях з метою забезпечення конфіденційності та повного розмежування доступу до інформації, яка може становити комерційну таємницю. Функціонально система захисту інформації забезпечує захист інформації за допомогою алгоритмів криптування та розкриптовування, гарантує її цілісність у системі обміну інформацією.

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

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

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

На сьогоднішній момент мова програмування C # один з найпотужніших, що швидко розвиваються і затребуваних мов в iT-галузі. На даний момент на ньому пишуться найрізноманітніші програми: від невеликих десктопних програмок до великих веб-порталів і веб-сервісів, які обслуговують щодня мільйони користувачів.

У порівнянні з іншими мовами C # досить молодий, але в той же час він вже пройшов великий шлях. Перша версія мови вийшла разом з релізом Microsoft Visual Studio .NET в лютому 2002 року. Поточною версією мови є версія C # 8.0, яка вийшла у вересні 2019 року разом з релізом .NET Core 3.

C # є мовою з Сі-подібним синтаксисом і близький в цьому відношенні до C ++ і Java. Тому, якщо ви знайомі з одним з цих мов, то опанувати C # буде легше.

C # є об'єктно-орієнтованим і в цьому плані багато перейняв у Java і C ++. Наприклад, C # підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє

вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків. і C # продовжує активно розвиватися, і з кожною новою версією з'являється все більше цікавих функціональностей, як, наприклад, лямбда, динамічне зв'язування, асинхронні методи і т.д.

Роль платформи .NET. Коли говорять C #, нерідко мають на увазі технології платформи .NET (Windows Forms, WPF, ASP.NET, Xamarin). і, навпаки, коли говорять .NET, нерідко мають на увазі C #. Однак, хоча ці поняття пов'язані, ототожнювати їх невірно. Мова C # був створений спеціально для роботи з фреймворком .NET, проте саме поняття .NET дещо ширше.

Якось Білл Гейтс сказав, що платформа .NET - це найкраще, що створила компанія Microsoft. Можливо, він мав рацію. Фреймворк .NET представляє потужну платформу для створення додатків. Можна виділити наступні її основні риси:

Підтримка декількох мов. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов: поряд з C # це також VB.NET, C ++, F #, а також різні діалекти інших мов, прив'язані до .NET, наприклад, Delphi. NET. При компіляції код на будь-якому з цих мов компілюється в збірку спільною мовою CIL (Common intermediate Language) - свого роду асемблер платформи .NET. Тому ми можемо зробити окремі модулі однієї програми на окремих мовах.

Кросплатформеність. .NET є яку переносять платформою (з деякими обмеженнями). Наприклад, остання версія платформи на даний момент .NET Core підтримується на більшості сучасних ОС Windows, MacOS, Linux. Використовуючи різні технології на платформі .NET, можна розробляти програми на мові C # для самих різних платформ - Windows, MacOS, Linux, Android, iOS, Tizen.

Потужна бібліотека класів. .NET представляє єдину для всіх підтримуваних мов бібліотеку класів. і яке б додаток ми не збиралися писати на C # - текстовий редактор, чат або складний веб-сайт - так чи інакше ми задіємо бібліотеку класів .NET.

Різноманітність технологій. Загальномовне середовище виконання CLR і базова бібліотека класів є основою для цілого стека технологій, які розробники можуть задіяти при побудові тих чи інших додатків. Наприклад, для роботи з базами даних в цьому стеку технологій призначена технологія ADO.NET і Entity Framework Core. Для побудови графічних додатків з багатим насиченим інтерфейсом - технологія WPF і UWP, для створення більш простих графічних додатків - Windows Forms. Для розробки мобільних додатків - Xamarin. Для створення веб-сайтів - ASP.NET і т.д.

Також ще слід відзначити таку особливість мови C # і фреймворка .NET, як автоматичне прибирання сміття. А це означає, що нам в більшості випадків не доведеться, на відміну від C ++, піклуватися про звільнення пам'яті. Вищезазначена загальномовного середовища CLR сама викличе збирач сміття і очистить пам'ять.

.NET Framework і .NET Core. Варто відзначити, що .NET довгий час розвивався головним чином як платформа для Windows під назвою .NET Framework. В 2019 вищла остання версія цієї платформи - .NET Framework 4.8. Вона більше не розвивається

З 2014 Microsoft став розвивати альтернативну платформу - .NET Core, яка вже призначалася для різних платформ і повинна була увібрати в себе всі можливості застарілого .NET Framework і додати нову функціональність. Тому слід розрізняти .NET Framework, який призначений переважно для Windows, і кросплатформенних .NET Core. У цьому посібнику йтиметься про C # в зв'язці з .NET Core, оскільки це актуальна платформа.

Також варто згадати про платформу Mono, яка була створена ще в 2004 році і представляла опенсорс-версію платформи .NET Framework для Linux і MacOS. Використовуючи Mono, можна було створювати кросплатформенних додатки на C #.

Керований і некерований код. Нерідко додаток, створене на C #, називають керованим кодом (managed code). Що це означає? А це означає, що для цієї програми створено на основі платформи .NET і тому керується загальномовна

середовищем CLR, яка завантажує додаток і при необхідності очищає пам'ять. Але є також додатки, наприклад, створені на мові C ++, які компілюються не в спільну мову CIL, як C # або VB.NET, а в звичайний машинний код. В цьому випадку .NET не керує додатком.

У той же час платформа .NET надає можливості для взаємодії з некерованим кодом ..

JIT-компіляція. Як вище писалося, код на C # компілюється в додатку або складання з розширеннями exe або dll на мові CIL. Далі при запуску на виконання подібного програми відбувається JIT-компіляція (Just-in-Time) в машинний код, який потім виконується. При цьому, оскільки наш додаток може бути великим і містити купу інструкцій, в поточний момент часу компілюватиметься лише та частина програми, до якої безпосередньо йде звернення. Якщо ми звернемося до іншої частини коду, то вона буде скомпільована з CIL в машинний код. При тому вже скомпільована частина програми зберігається до завершення роботи програми. У підсумку це підвищує продуктивність.

Хоча визначення мови C# і CLI стандартизовані ISO та Ecma, що забезпечує розумний і недискримінаційний ліцензійний захист (RAND) від патентних позовів, Microsoft використовує C# і CLI у своїй бібліотеці Base Class Library (BCL), яка є фундаментом їхньої власницької платформи .NET framework, і яка забезпечує безліч нестандартизованих класів (розширений i/O, GUi Windows Forms, веб-служби тощо). У деяких випадках, де патенти Microsoft відносяться до стандартів, використаних у .NET framework, документовані Microsoft, і застосовані патенти доступні через інші RAND умови або через Обітницю Відкритої Специфікації Microsoft (Microsoft's Open Specification Promise, OSP), які випускають патентні права публічно[10]. Але є деякі застереження і обговорення про те, що існують додаткові аспекти, патентовані Microsoft, що не покриті, які можуть утримувати незалежних реалізаторів повного фреймворку. Microsoft також погодився не позиватися проти розробників відкритого програмного забезпечення щодо порушення прав у неприбуткових проектах для частини свого фреймворку, покритого OSP.

Microsoft погодився не порушувати патентних вимог щодо продуктів Novell проти платних клієнтів Novell за винятком переліку продуктів, що явно не згадують C#, .NET чи реалізацію .NET від Novell (проект Mono). Проте Novell дотримується точки зору, що Mono не порушує жодного патенту Microsoft.. Microsoft також уклав спеціальну угоду не позиватися проти браузерного плагіну Moonlight, який спирається на Mono, отриманного від Novell.

У зауваженні, опублікованому на сайті новин Free Software Foundation у червні 2009 Річард Столлман попереджає, що він вважає, що «Microsoft можливо планує одного дня оголосити всі вільні реалізації C# такими, що використовують програмні патенти» і рекомендував розробникам уникати того, що він називає «безвідплатним ризиком», пов'язаним із «залежністю вільних реалізацій C#». Free Software Foundation пізніше повторила свої попередження, стверджуючи, що розширення Microsoft Community Promise на специфікації ECMA C# і CLI можуть не вберігти від шкідництва Microsoft відкритим реалізаціям C#, оскільки багато специфічних для Windows бібліотек, включених у .NET та Mono, не покриті цими обіцянками. Тому більшість провідних дистрибутивів Лінукс, за винятком Novell SUSE Linux, не включають Mono в установку за умовчанням (хоча його і можна завантажити з репозиторіїв).

3.2 Реалізація програмного засобу на основі криптоалгоритму стійкого до атаки в часі

Якщо Аліса хоче захистити свій дім, вона може придбати якісні замки і встановити кілька із них на свої двері. Однак розумний грабіжник може просто відкрутити петлі, зняти двері і піти з усіма цінностями Аліси з мінімальними зусиллями. Цей приклад непрямого нападу на безпеку домогосподарств є дещо штучним, але існує паралель у світі шифрування, яка є цілком реальною. Це

називається синхронізацією, і воно було використано для перемоги над деякими з найпопулярніших методів шифрування.

Чи вважаєте ви, що ваша комп'ютерна система захищена, оскільки ви використовуєте потужну криптографію? Чи знаєте ви, що зловмисники можуть атакувати вашу криптографію в абсолютно несподіваному напрямку, не порушуючи безпосередньо криптографічний алгоритм?

RSA [27] - це криптографічний алгоритм із відкритим ключем, який сьогодні широко використовується для забезпечення електронної передачі даних. Він включений як частина веб-браузерів від Microsoft та Netscape і використовується SSL (рівень захищених сокетів), що забезпечує безпеку та конфіденційність через інтернет. Алгоритм RSA був винайдений командою Рівеста, Шаміра та Адлемана на MiT у 1978 р. Незалежно від того, Кліфф Кокс відкрив ту саму ідею на початку 1970-х [25]. Криптосистема з відкритим ключем використовує односторонню функцію, яку легко обчислити в одному напрямку і важко обчислити в зворотному напрямку. Наприклад, порівняно легко сформувати два простих числа p, q і обчислити їх добуток N = p * q. Але з огляду на N важко знайти його фактори p і q. Шифрування використовує загальнодоступну цінність або ключ, який розповсюджується та відомий кожному, хто хоче надіслати повідомлення. Розшифровка передбачає пов’язаний приватний ключ, який передбачуваний одержувач тримає в таємниці, і його неможливо вивести з відкритого ключа. Криптографія з відкритим ключем працює, не вимагаючи від обох залучених сторін зберігати узгоджену таємницю; закритий ключ ніколи не потрібно надсилати відправнику.

Відкритий ключ RSA включає число N, яке є добутком двох великих простих чисел p, q. Сила RSA випливає з того, що факторинг великих чисел важкий. Найвідоміші методи факторингу все ще дуже повільні. Наприклад, у нещодавньому виклику RSA (серпень 1999 р.) 512-розрядний номер виклику RSA було враховано за допомогою 292 робочих станцій та високошвидкісних комп'ютерів. На факторинг знадобилося 35,7 CPU-років, що еквівалентно приблизно 80 000 MiPS років. На подвиг потрібно 3,7 місяця календарного часу

[8]. Оскільки дуже багато людей намагаються знайти ефективні способи врахування великих чисел, дотепер, не маючи великого успіху, ми можемо припустити, що RSA захищений від факторингової атаки для типового ключа N довжиною 1024 біта. RSA можна зробити більш захищеним від факторингу, збільшивши довжину ключа до 2048 біт і більше.

Незважаючи на цю грізну математичну силу, дослідження показали, що можливо відновити приватні ключі RSA, не порушуючи безпосередньо RSA. Цей тип атаки відомий як синхронізована атака, коли зловмисник спостерігає за часом роботи криптографічного алгоритму і тим самим визначає секретний параметр, що бере участь в операціях. Хоча загальновизнано, що RSA захищений від прямої атаки, вразливість RSA до атак часу не настільки відома і часто не помічається.

За останні кілька років у літературі почали з'являтися нові види криптоаналітичних атак: атаки, які націлені на конкретні деталі реалізації. „Атака в часі" викликала великий сплеск: приватні ключі RSA можна було відновити, вимірявши відносний час криптографічних операцій. Ця атака була успішно реалізована на смарт-картки та інші маркери безпеки, а також на сервери електронної комерції в інтернеті.

Дослідники узагальнили ці методи, включивши атаки на систему, вимірюючи споживання енергії, випромінювання випромінювання та інші «побічні канали», і застосували їх проти різноманітних відкритих ключів та симетричних алгоритмів у «захищених» токенах. Супутні дослідження розглядали аналіз несправностей: навмисне введення несправностей у криптографічні процесори з метою визначення секретних ключів. Наслідки цієї атаки можуть бути руйнівними.

У чому головна ідея? Є два способи розглянути криптографічний примітив: блоковий шифр, функція цифрового підпису тощо. Перший - це шматок математики. Другий - це фізична (або програмна) реалізація цієї математики.

Традиційно криптоаналіз був спрямований виключно проти математики. Хорошими прикладами цього є диференціальний та лінійний криптоаналіз: потужні математичні інструменти, які можна використовувати для розбиття різних блокових шифрів.

З іншого боку, синхронізація атак, аналіз потужності та аналіз несправностей - все це робить припущення щодо реалізації та використовує додаткову інформацію, отриману від нападу на ці реалізації. Аналіз помилок передбачає однобітовий зворотний зв'язок від реалізації - чи було повідомлення успішно розшифровано - для того, щоб зламати базовий криптографічний примітив. Атаки часу передбачають, що зловмисник знає, скільки часу триває певна операція шифрування.

Я люблю вважати ці напади біологічними. Є деякі речі, про які ви просто не можете дізнатися про організм, розібравши його на частини. іноді доводиться дивитись на вхідні та вихідні дані. Як воно рухається? Що воно їсть? Якщо ви пробиваєте його певним чином, як він реагує? Якщо ви його зламаєте, що станеться?

Називайте їх бічними атаками. Звичайні атаки переглядають відкритий текст і зашифрований текст і намагаються відновити ключ. Атаки бічних каналів також розглядають деяку іншу інформацію - як змінюється споживання енергії під час виконання шифру, як виглядає висновок, коли ви обрізаєте кілька проводів - у спробі відновити ключ.

Ці атаки не обов’язково узагальнюють. Атака аналізу несправностей просто неможлива щодо реалізації, яка не дозволяє зловмисникові створювати та використовувати необхідні несправності. Але ці атаки можуть бути набагато потужнішими. Наприклад, для диференціального аналізу помилок DES для відновлення ключа потрібно від 50 до 200 блоків зашифрованого тексту (без відкритого тексту). Порівняйте це з найкращою атакою сторонніх каналів на сторонній канал, для якої потрібно трохи менше 64 терабайт відкритого тексту та зашифрованого тексту під одним ключем.

Деякі дослідники стверджували, що це обман. Правда, але в реальних системах зловмисники обманюють. Їхня робота полягає у тому, щоб відновити ключ, а не дотримуватися деяких правил поведінки. Розсудливі інженери захищених систем це передбачають і пристосовуються до цього. Ми вважаємо, що більшість оперативних криптоаналізів використовують побічну інформацію. Звук як бічний канал - прослуховування обертання електромеханічних роторних машин - був згаданий у The Codebreakers. TEMPEST - це ще один бічний канал, який може бути дуже ефективним. А у своїй книзі «Виловлювач шпигунів» Пітер Райт обговорював витікання даних на лінію електропередачі як бічний канал, який використовується для розриву французького криптографічного пристрою.

Захист важкий. Ви можете або зменшити кількість витоків інформації про бічні канали, або зробити витік неактуальним. У обох є проблеми, хоча над ними працюють дослідники. Це потужна атака, і через деякий час з’явиться хороша теорія оборони. Тим часом будь-яка система, де пристрій зберігається однією особою, а секрети в пристрої зберігається іншою людиною, знаходиться під загрозою.

Порівнюючи два MAC або хеші (також хеші паролів) для рівності, спочатку ви думаєте, що просте порівняння буде нормальним? Подумайте ще раз, оскільки вони сприйнятливі до атак часу.

У криптографії синхронізація - це атака бічного каналу, при якій зловмисник намагається скомпрометувати криптосистему, аналізуючи час, необхідний для виконання криптографічних алгоритмів.

```
if ($hash1 == $hash2)
{
     //mac verification is Okay
     return "hashs are equal"
     } else {
     //something happened
     return "hashs verification failed!";
}
```

Обидва аргументи мають бути однакової довжини для успішного порівняння. Коли подаються аргументи різної довжини, FALSE негайно

повертається, і довжина відомого рядка може просочитися у випадку атаки синхронізації. Це робиться шляхом хронометражу часу, який потрібно для порівняння відомої довжини рядка з HASH, і оскільки ви потім знайдете довжину рядка, виходячи з часу, який потрібно під час порівняння.

Порівнює двобайтові масиви за постійним часом. Цей метод порівняння використовується для того, щоб хеші паролів не могли бути вилучені із систем за допомогою синхронізації, а потім атаковані.

```
private static bool SlowEquals(byte[] a, byte[] b)
 {
 uint diff = (uint)a.Length ^ (uint)b.Length;
 for (int i = 0; i < a.Length && i < b.Length; i++)
 diff |= (uint)(a[i] ^ b[i]);
 return diff == 0;
 }
```

Що означає рядок diff | = (uint) (a [i] ^ b [i]); Цей набір різниться залежно від того, чи є різниця між a і b.

Він дозволяє уникнути синхронізації, завжди проходячи повний коротший з двох a і b, незалежно від того, чи є невідповідність раніше цього чи ні.

Різниця | = (uint) (a [i] ^ (uint) b [i]) приймає ексклюзив-або байта a з відповідним байтом b. Це буде 0, якщо два байти однакові, або ненульові, якщо вони різні. Тоді це або з різницею.

Отже, для ітерації diff буде встановлено ненульове значення, якщо між входами в цій ітерації було знайдено різницю. Як тільки diff отримає ненульове значення на будь-якій ітерації циклу, він збереже ненульове значення через подальші ітерації.

Отже, кінцевий результат у diff буде ненульовим, якщо буде знайдено різницю між відповідними байтами a і b, і 0, лише якщо всі байти (і довжини) a і b рівні.

Однак, на відміну від типового порівняння, це завжди буде виконувати цикл, поки всі байти на коротшому з двох входів не будуть порівняні з байтами

на іншому. Типове порівняння могло б мати місце на початку, коли цикл буде розірвано, як тільки буде виявлено невідповідність:

Корпорація Майкрософт надає SecureString для зберігання конфіденційних даних, і для цього використовується загальний метод Equals(Object), який визначає, чи дорівнює вказаний об'єкт поточному об'єкту. Немає інформації про те, що метод Equals захищає від атак часу. Тому ми можемо лише зробити висновок, що це не безпечно для цієї операції.

Загалом, будь-який канал, який може передавати інформацію із захищеної зони назовні, слід вивчати як потенційний ризик. Характеристики синхронізації, специфічні для реалізації, забезпечують один такий канал і іноді можуть використовуватися для компрометації секретних ключів. Вразливі алгоритми, протоколи та системи повинні бути переглянуті, щоб включити заходи щодо протидії терміновому криптоаналізу та пов'язаним з ним атакам.

3.3 Тестування та верифікація розробленого програмного засобу

Суть атаки полягає в тому що зловмисник знає, чи хоча б здогадується, про алгоритмі перевірки автентичності і пробує підібрати ключ поступово. Підставляючи різні значення і заміряючи час перевірки, можна помітити, що для деяких варіантів ключів час виконання довше (або швидше) ніж для інших.

На початку хотів зробити клієнт і сервер, але вирішив обійтися без складнощів, адже хотілося перевірити саму ідею, на скільки вона працює хоча б в лабораторних умовах. Для проведення експерименту в якості піддослідного буде використовуватися функція перевірки автентичності, в якій будемо просто порівнювати два ключа поелементно.

```
static bool check_secret_key(int[] key)
        {
```

```
            for (int i = 0; i < _secret_key.Length && i <
key.Length; i++)
                if (key[i] != _secret_key[i])
                return false;
            return _secret_key.Length == key.Length;
        }
```

В цьому випадку, при збігу перших елементів ключа операція виконується довше, так як переходить до перевірки таких елементів. Алгоритм підбору досить простий: перебираємо перше число, варіант який виконується найдовше - залишаємо, потім друге і т.д.(додаток А)

Розрахунок часу на скільки метод ефективніше повного перебору в даному конкретному випадку.

Ключ - масив чисел з n = 10 елементів зі значеннями від 0 до 99(m = 100) включно.

Тоді для повного перебору кількість перевірок одно m ^ n = 100^10 =10^20;

Для реалізованої атаки часу n * a * m * b, де a і b це емпіричні константи і рівні 1500, отримуємо 10 * 1500 * 100 * 1500 = 2,25 * 10 ^ 9

Як можна бачити, в даному конкретному випадку, результат відрізняється більш ніж на 10 порядків, що говорить про його ефективність у порівнянні з повним перебором.

При перших запусках отримував практично випадковий результат. Почав поступово збільшувати кількість тестів і вже на третьому запуску отримав правильні послідовності. і як наслідок, в лабораторних умовах спосіб працює і працює досить добре.

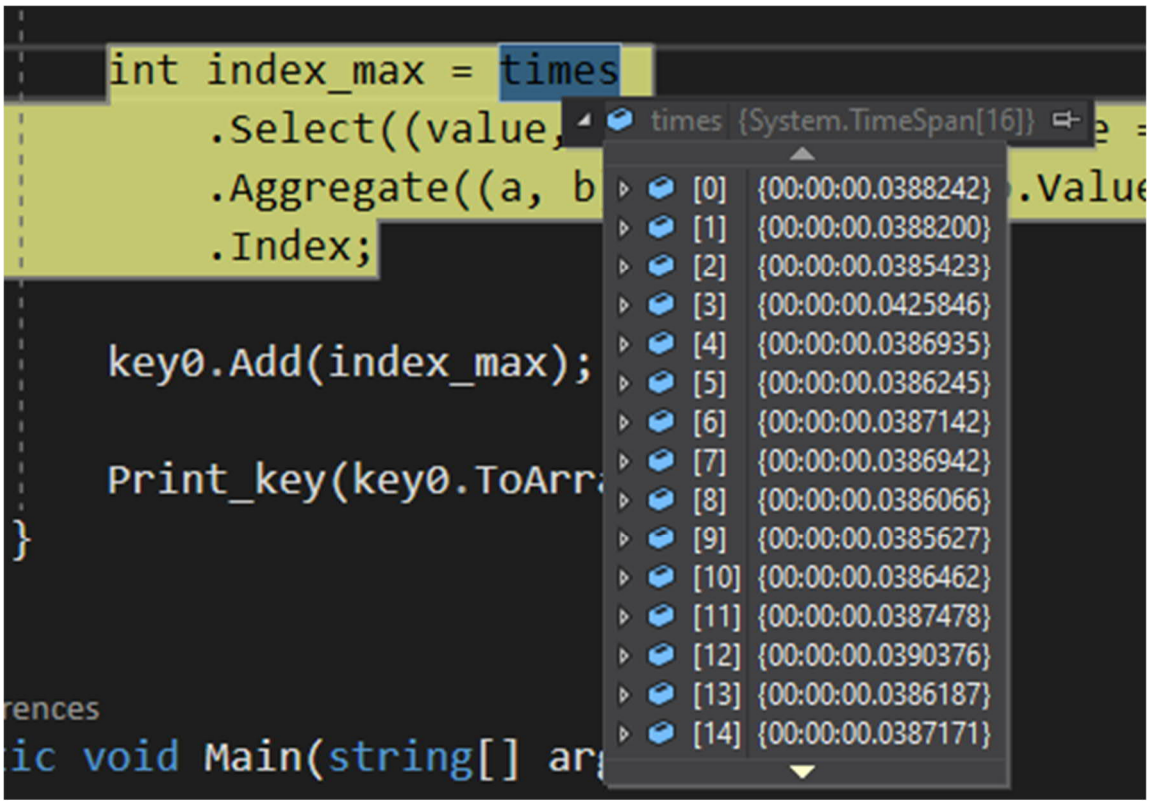

Рисунок 3.1 Приклад роботи програми часової атаки

Ось що отримуємо в консолі (в першому рядку виводиться секретний ключ):

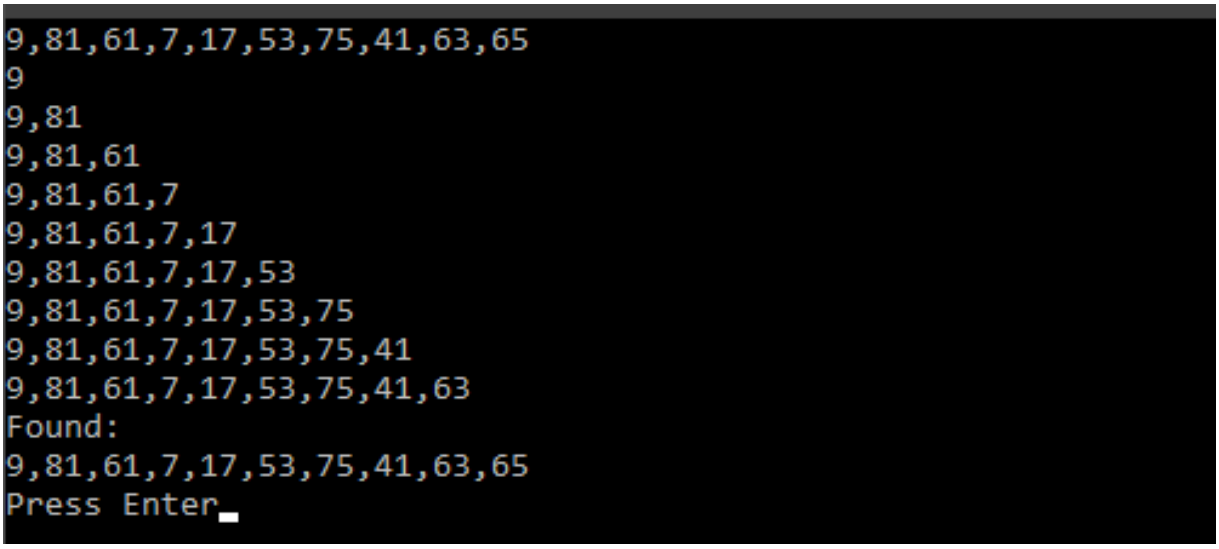

Рисунок 3.2 Результат роботи програми часової атаки

Тепе запустимо модифіковану програму захищену від атаки в часі.

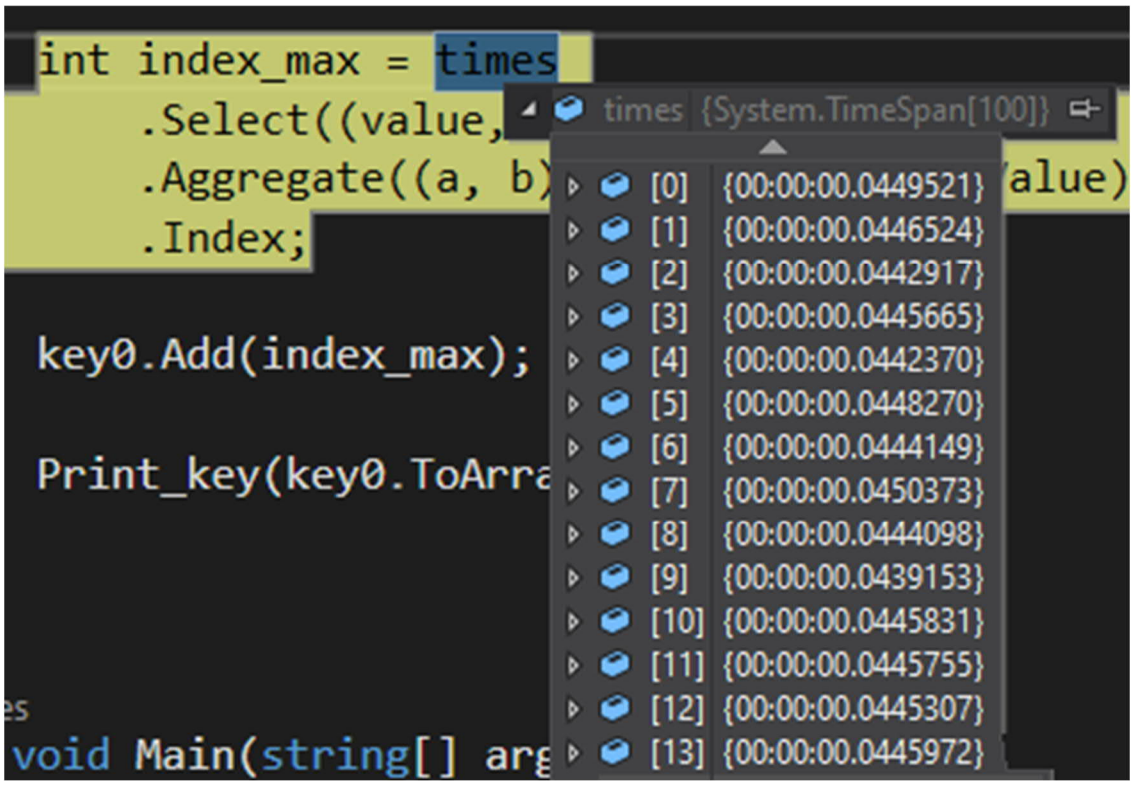

Рисунок 3.3 Приклад роботи програми захищеної від атаки в часі

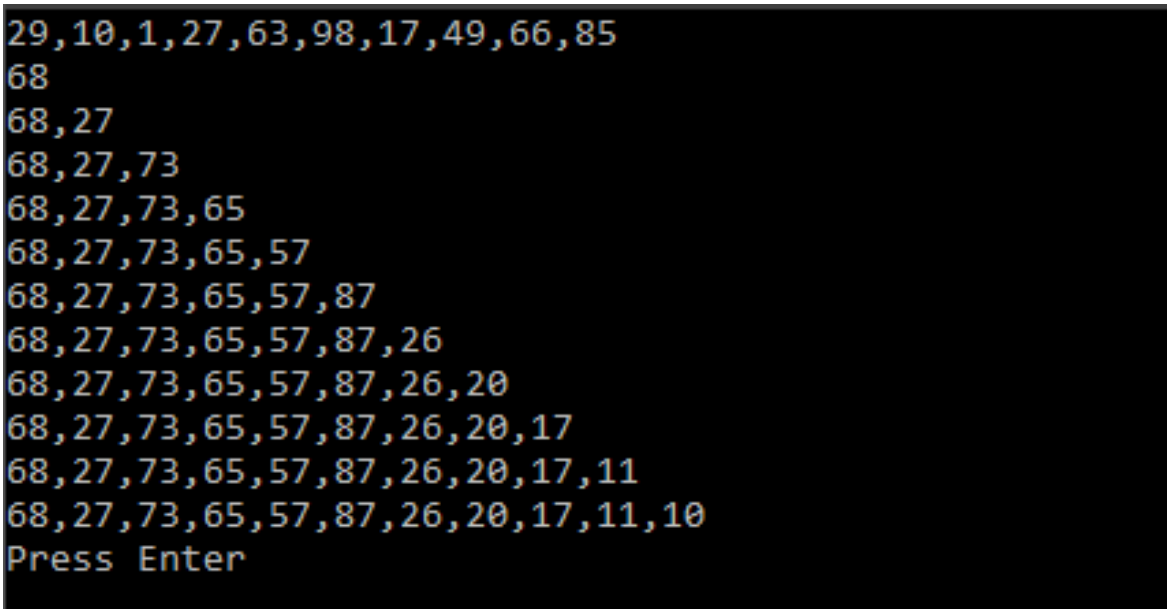

Рисунок 3.4 Результат роботи програми захищеної від атаки в часі

Як бачимо модифіковвана програма не дає можливосіті здійснити атаку в часі, завдяки тому що час обробки кожного елемента одинаковий.

Зважаючи на всі різні типи атак та вразливостей, написання абсолютно безпечного коду майже неможливе (саме тому ми віримо в Управління захистом додатків), але впроваджуючи найкращі практики, такі як перевірка безпечного коду та роблячи розумні рішення, ви можете зробити позитивна різниця у вашій безпеці. Розуміння того, як працюють різні атаки, є важливим елементом цього. Атаки синхронізації - прекрасний приклад вразливості, яку ви можете пом'якшити в коді. Звичайно, як і в будь-якому іншому, пов’язаному з безпекою, диявол полягає в деталях.

3.4 Висновки до розділу

Використання оптимізованих функцій порівняння рядків та підтримка атомного порівняння 64-розрядних слів робить використання побічних каналів синхронізації на основі порівняння дуже малоймовірним для програм, що використовують ці функції на сучасному обладнанні. Дуже привілейована мережева позиція або атаки, здійснені на одному і тому ж хості, значно збільшують роздільну здатність атак і можуть надати зловмисникові додаткову перевагу. Однак, на основі аналізу, ця перевага виявляється недостатньою для практичних віддалених атак. Оскільки регулярне порівняння рядків конфіденційних даних все ще витікає з деякої інформації (навіть якщо її не можна використовувати віддалено) як частину найкращого підходу, слід застосовувати захист, як обговорювалося. Зокрема, вбудовані системи залишаються під більшим ризиком через їх більш низьку швидкість обробки, і Arduino та інші вбудовані системи, як видається, є реальними цілями.

Вплив побічних каналів, що базуються на розгалуженні, залежить від програми, але загальні складні операції, як видається, призводять до затримок, які є досить значними для розрізнення віддаленим зловмисником.

Наприклад, будь-яка форма вводу-виводу, запити до бази даних, навіть обчислення хеш-функцій додає достатньо часу для обробки віддаленим зловмисником для виявлення та потенційного використання.

ВИСНОВКИ

У роботі вирішено важливу задачу – підвищення стійкості до часового атаки комп’ютерних систем в реальному часі.

В результаті досліджень були вирішені наступні завдання:

1. Проведено аналіз асиметричних систем захисту інформаційних потоків, в результаті якого були встановлені основні трудомісткі операції.

2. Проведено аналіз сучасних атак на криптопристрої показав, що найпоширенішими є атаки на реалізацію, а особливо небезпечною є пасивна часова атака, яку важко виявити в процесі роботи підсистеми захисту інформації комп’ютерної системи.

3. Запропоновані нові теоретичні положення опрацювання інформаційних потоків, які дозволяють зменшити ризики при генеруванні ключів, шифруванні/дешифруванні асиметричних алгоритмів з відкритими ключами RSA тощо.

4. Проведено дослідження часової складності запропонованого підходу щодо захисту в часі реалізації криптосистеми RSA на основі використання нових методів захисту.

5. Проведено програмну імплементацію удосконалення реалізації криптосистеми RSA на основі використання нових методів захисту та досліджено часові характеристики вхідних параметрів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Момотюк О.В., Поворозник В.С. Застосування нечіткої логіки в системах захисту інформації. III Науково-практична конференція молодих вчених і студентів «інтелектуальні комп'ютерні системи та мережі» – Тернопіль, 2020. – С.15

2. Поворозник В.С. Сучасні методи підвищення стійкості криптоалгоритмів. III Науково-практична конференція молодих вчених і студентів «інтелектуальні комп'ютерні системи та мережі» – Тернопіль, 2020. – С.55

3. ДСТУ 8302:2015. інформація та документація. Бібліографічне посилання. Загальні положення та правила складання / Нац. стандарт України. – Вид. офіц. – [Уведено вперше ; чинний від 2016-07-01]. – Київ : ДП «УкрНДНЦ», 2016. – 17 с.

4. Методичні вказівки до написання техніко-економічного розділу дипломних проектів освітньо-кваліфікаційного рівня «бакалавр» напряму підготовки 6.050102 «Комп'ютерна інженерія» / І.Р. Паздрій – Тернопіль: ТАНГ, 2014. – 37 с.

5. Методичні рекомендації до виконання дипломного проекту з освітньо-кваліфікаційного рівня "Спеціаліст". Спеціальність «Комп'ютерні системи та мережі» / О.М. Березький, Р.Б. Трембач, Л.О.Дубчак, Г.М. Мельник / Під ред. О.М. Березького. - Тернопіль: ТНЕУ, 2012.–63 с.

6. Методичні рекомендації до виконання дипломної роботи з освітньокваліфікаційного рівня "Магістр". Спеціальність „Комп'ютерні системи та мережі" / О.М. Березький, Л.О. Дубчак /Під ред. О.М. Березького – Тернопіль: ТНЕУ, 2012.– 47 с.

7. Дипломне проектування за напрямами підготовки "Прикладна математика", "Комп'ютерна інженерія", "Програмна інженерія" [Текст]: навч.-метод. посіб. / Є.С. Сулема; за заг. ред. І.А. Дички. – К.:НТУУ"КПі", 2011. – 224с.

8. Загальні рекомендації з підготовки, оформлення, захисту й оцінювання випускних кваліфікаційних робіт здобувачів вищої освіти першого бакалаврського і другого магістерського рівнів / за ред. доц. М.І. Шинкарика. – Тернопіль:ТНЕУ, 2018. – 60с.

9. Широчин В.П. Вопросы проектирования механизмов защиты информации в компьютерных системах и сетях / В.П.Широчин, В.Е.Мухин, А.В.Кулик. - К.: “ВЕК+”, 2000. – 112 с.

10. Дудикевич В.Б. Розробка клієнт-орієнтованих засобів шифрування абонентських даних в мобільному зв’язку / В.Б.Дудикевич, Ю.Л.Пархуць // інформаційна безпека. – 2011. - №1(5). – С.83-87.

11. Василенко В. Методики визначення вихідних даних для оцінки залишкових ризиків у ЛОМ / В.Василенко, М.Будько. // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. – 2004. – Випуск 9. – С.110-120.

12. Тичковський Р.О. Математичне та програмне забезпечення оптимального розподілу ресурсів серед вузлів комп’ютерних мереж: автореф. дис. на здобуття наук. ступеня канд. техн. наук: спец. 01.05.03 «Математичне та програмне забезпечення обчислювальних машин і систем» / Р.О.Тичковський. – Львів, 2010. – 20 с.

13. Столлингс В. Криптография и защита сетей: принципы и практика, 2-е изд.: Пер. с англ. / В.Столлингс. – М.: Изд. Дом «Вильямс», 2001. – 672 с.

14. Безмалый Н.В. Как ломаются пароли / Н.В.Безмалый // Журнал информационных технологий CHIP. – 2008. - №7. – С.124-126.

15. Україна значно піднялася в рейтингу країн з найбільшою кількістю кібер-загроз [Електронний ресурс] - Режим доступу: http://www.rbc.ua/ukr/top/show/

16. Зайчук А.В. Основные пути утечки информации и несанкионированного доступа в корпоративных сетях / А.В.Зайчук // Захист інформації. – 2003. – № 4. – С. 19-24.

17. Чеховский С.А. Побочные излучения и защита информации в локальных сетях. / С.А.Чеховский, Ю.М.Рудаков // Захист інформації. – 2003. – № 4. – С. 30-38.

18. Koeune F. A Tutorial on Physical Security and Side-Channel Attacks/ F.Koeune, F.-X. Standaert : Foundations of Security Analysis and Design iii (FOSAD 2004/2005), November 2006. – 2006. - LNCS 3655. – P. 78-108.

19. Заболотний В.І. Класифікація технічних каналів витоку інформації / В.І.Заболотний // Радіотехніка. Тематичний випуск “інформаційна безпека”. - 2003. - № 134. - С.210-218.

20. Журавель Т.Н. Некоторые особенности защиты информации с ограниченным доступом от утечки по виброакустическому каналу / Т.Н.Журавель // Защита информации: Сборник научных трудов. Выпуск 10. – Киев: НАУ, 2003. - С.91-95.

21. Васильченко И.И. Магнитоэлектрические виброизлучатели с пониженным уровнем акустического шума для систем технической защиты информации / И.И.Васильченко, И.А.Кравченко / Защита информации: Сборник научных трудов. Выпуск 10. – Киев: НАУ, 2003. - С.96-105.

22. Безруков К. Н. Классификация компьютерных вирусов MS DOS и методы защиты от них / К.Н.Безруков.— М.; СПб "iCE", 1990. – 48 с.

23. Brier E. Chemical Combinatorial Attacks on Keyboards / E.Brier, D.Naccache, P.Paillier [Електронний ресурс] - Режим доступу: http://eprint.iacr.org/2003/217.pdf

24. Skorobogatov S. Optical Fault induction Attacks / S.Skorobogatov, R.Anderson // Cryptographic Hardware and Embedded Systems – CHES 2002): 4th international Workshop, August 13-15, 2002: Proceedings. − San Francisco (USA), 2002.− P.2-12.

25. Горбачёв В. Модель угроз, реализуемых аппаратными ресурсами компьютерных систем / В.Горбачёв, В.Степаненко, Т.Гриценко. // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. – 2004. – Випуск 9. – С.75-78.

26. Agrawal D. The EM Side-Channel(s) / D.Agrawal, B.Archambeault, J.R.Rao, P.Rohatgi // Cryptographic Hardware and Embedded Systems – CHES 2002): 4th international Workshop, August 13-15, 2002: Proceedings. - San Francisco 2002. - LNCS 2523. - P. 29 - 45.

27. Kocher P. Differential Power Analysis / P.Kocher, J.Jaffe, B.Jun // in Advances in Cryptology (CRYPTO'99): 19th Annual international Cryptology Conf., August 1999: Proceedings. - Santa Barbara, California, USA, 1999. - LNCS 1666. - P.388-397.

28. Гопиенко А.В. Формирование потайных каналов передачи информации в компьютеризированных измерительных системах / А.В.Гопиенко, Ю.В.Куц, Е.В.Монченко // Системи обробки інформації. – 2012. – Вип. 3(101). – Т.1. – С.123-126.

29. Васильцов І.В. Методи захисту проти атак спеціального виду / І.В.Васильцов, Л.О.Дубчак // Вісник Хмельницького національного університету. Технічні науки. – 2007. - №5. – С.174-182.

30. Васильцов І.В. Класифікація сучасних атак спеціального виду на реалізацію / І.В.Васильцов, Л.О.Дубчак // Захист інформації. – 2007. - №4. – С.10-21.

31. December 19, 1996. – 24 p. [Електронний ресурс] - Режим доступу: http://people.cs.umass.edu/~kevinfu/papers/rc5-dfa-paper.pdf

32. E.Biham, A.Shamir // Advances in Cryptology (CRYPTO ’97): 17th Annual international Cryptology Conf., 1997: Proceedings. -Santa Barbara (USA). - LNCS 1294 . Springer-Verlag, 1997.− P.513-525.

33. Messerges T.S. Power Analysis Attacks of Modular Exponentiation in Smartcards / T.S.Messerges, E.A.Dabbish, R.H.Sloan // Cryptographic Hardware and Embedded Systems (CHES’99): First international Workshop, August 1999. - Worcester, MA, USA. - LNCS 1717. – Springer-Verlag Berlin Heidelberg 1999. – P.144-157.

34. Biham E. Diferential fault analysis of secret key cryptosystems / E.Biham, A.Shamir // Advances in Cryptology {Crypto '97): LNCS, Vol.1294. - Berlin, Springer-Verlag, 1997. - P. 513-525.

35. Muir J.A. Techniques of Side Channel Cryptanalysis: thesis requirement for the degree of Master of Mathematics in Combinatorics and Optimization. / J.A.Muir – Waterloo, Ontario, Canada, 2001. – 92 p.

36. University of Waterloo. Dept. of Combinatorics and Optimization.− Waterloo (CA), 2001. – 153 p.

37. Kelsey J. Side Channel Cryptanalysis of Product Ciphers / J.Kelsey, B.Schneier, D.Wagner, C.Hall // Journal of Computer Security. – 2000. - v.8, №2-3. - P. 141-158.

38. Tiri K. Side-Channel Attack Pitfalls / K.Tiri // Design Automation Conference (DAC 2007), June 4-8, 2007: Proceedings. - San Diego, California, USA, 2007. - P.15-20.

39. Коркішко Л. Статистична модель суматора за модулем 2N для проведення інженерно-криптографічних атак за побічними каналами витоку інформації / Л.Коркішко, І.Васильцов. // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. – 2004. – Випуск 8. – С.115-121.

40. Карпінський Б.З. Пристрої потокового шифрування підвищеної стійкості до спеціальних впливів: автореф. дис. на здобуття наук. ступеня канд. техн. наук: спец. 05.13.05 «Елементи та пристрої обчислювальної техніки та систем керування» / Б.З.Карпінський. - Тернопіль, 2007. – 21 с.

41. Чмора А.Л. Современная прикладная криптография.−2-е изд./ А.Л.Чмора. − М.: Гелиос АРВ, 2002. – 256 с.

42. Якименко І.З. Методи та алгоритми опрацювання інформаційних потоків в комп’ютерних мережах за умови застосування еліптичних кривих: автореф. дис. на здобуття наук. ступеня канд. техн. наук: спец. 05.13.05 «Комп’ютерні системи та компоненти» / І.З.Якименко. – Тернопіль, 2012. – 20 с.

43. Wollinger T. How Secure Are FPGAs in Cryptographic Applications? / T.Wollinger, C. Paar // Field Programmable Logic and Applications (FPL 2003): 13th international Conf., September 1-3, 2003: Proceedings. - Lisbon, Portugal, 2003. – P.91-100.

44. Messerges T.S. Power Analysis Attacks of Modular Exponentiation in Smartcards / T.S.Messerges, E.A.Dabbish, R.H.Sloan // Cryptographic Hardware and Embedded Systems (CHES'99): First international Workshop, August 1999. - Worcester, MA, USA. - LNCS 1717. – Springer-Verlag Berlin Heidelberg 1999. – P.144-157.

45. I. Power and Fault Analysis in ECC. Problems and Solutions / I.Vasyltsov, H.-K.Son, E.Baek // e-Smart 2005 Conference, September, 2005. - Sophia-Anthipolis, France, 2005.

46. Ding C. The Differential Cryptanalysis and Design of Natural Stream Ciphers / C.Ding //in Fast Software Encryption: Cambridge Security Workshop, December 1993. - Springer-Verlag, Berlin, 1994. - P. 101-115.

47. Hanley N. Correlation Power Analysis of Large Word Sizes / N.Hanley, R.McEvoy, M.Tunstall, C.Whelan, C.Murphy, W.P.Marnane // irish Signals and Systems Conference (iSSC 2007), September 13-14, 2007: Proceeding. - Derry, 2007 – 6 p.

48. Groza B. Cryptanalysis of an Autentication Protocol / B.Groza, D.Petrica // Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05): 7-th Symposium, 2006: Proceedings. – 2005. – P.147-157.

49. Xiao L. An improved Power Analysis Attack Against Camellia's Key Schedule / L.Xiao, H.M.Heys // September 22, 2005. – 16 p. [Електронний ресурс] - Режим доступу: http://eprint.iacr.org/2005/338.pdf

50. Fouque P. Power Attack on Small RSA Public Exponent / P.Fouque, S.Kunz-Jacques, G.Martinet, F.Muller, F.Valette / 15 p. [Електронний ресурс] - Режим доступу: http://www.iacr.org/archive/ches2006/27/27.pdf

51. Капустян М.В. Оценка эффективности функционирования сложных систем / М.В.Капустян, В.А.Хорошко // інформаційна безпека. – 2011. - №1(5). – С.5-8.

52. Brumley D. Remote Timing Attacks are Practical / D.Brumley, D. Boneh [Електронний ресурс] - Режим доступу: http://crypto.stanford.edu/~dabo/pubs/

53. papers/ssl-timing.pdf

54. Quisquater J.-J. Side Channel Attacks / J.-J.Quisquater, F.Koeune.// State-of-the-art regarding side channel attacks: report, October, 2010. – 2010. – 47 p. [Електронний ресурс] - Режим доступу: http://www.ipa.go.jp/security/enc/ CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf

55. Вельшенбах М. Криптография на Си и С++ в действии: Учебное пособие. / М.Вельшенбах. – М.: Издательство Триумф, 2004. – 464 с.

56. Молдовян А.А. Криптография. / А.А.Молдовян, В.А.Молдовян, Б.Я.Советов –СПб.: Издательство “Лань”, 2000. – 224 с.

57. Задірака В.К. Методи захисту фінансової інформації: Навч. посібник / В.К.Задірака, О.С.Олексюк. – К.: Вища школа, 2000. – 460 с.

58. Ємець В. Сучасна криптографія. Основні поняття / В.Ємець, А.Мельник, Р.Попович. - Львів: БаК, 2003. – 144 с.

59. Березький О.М., Дубчак Л.О., Мельник Г.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня “Магістр”. Спеціальність: 123 - Комп’ютерна інженерія. Магістерська програма - Комп’ютерна інженерія"/ Під ред. О.М. Березького. Тернопіль:ЗУНУ,2020.32 с.

60. Гураль І.В., Дубчак Л.О. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп’ютерна інженерія»/Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.