Хмельницький національний університет
Факультет програмування
та комп'ютерних і телекомунікаційних систем
Кафедра кібербезпеки та комп'ютерних систем і мереж

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Криптографічна система з відкритим ключем для захисту інформації при передачі через мережу
Назва теми

КвРКБ.170155.17.01.16 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *125 «Кібербезпека»*
Шифр, назва

Освітня програма *«Кібербезпека»*
Назва

Виконав: студент IV курсу, група КБ–17–1 _____ А.А. Фаринюк
Підпис Ініціали, прізвище

Керівник _____ Ю.П. Кльоц
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ І.В. Муляр
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри кібербезпеки та комп'ютерних систем і мереж _____ Ю.П. Кльоц
Підпис Ініціали, прізвище

«18» червня 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ПРОГРАМУВАННЯ ТА КОМП'ЮТЕРНИХ І ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМ

Кафедра КІБЕРБЕЗПЕКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 125 КІБЕРБЕЗПЕКА

Освітня програма ОСВІТНЯ ПРОГРАМА «КІБЕРБЕЗПЕКА»

ЗАТВЕРДЖУЮ

Завідувач кафедри Ю.П.Кльоц

"5" 02 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Фаринюку Анатолію Анатолійовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Криптографічна система з відкритим ключем для захисту інформації при передачі через мережу

Керівник проекту (роботи) Кльоц Юрій Павлович

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

завідувач кафедри, к.т.н., доцент

Затверджена наказом ректора університету від 05.02.2021 № 11 додаток №9

2. Строк подання студентом проекту (роботи) на кафедру 28.05.2021

3. Вихідні дані до проекту (роботи) Дослідження даних щодо систем криптографії з відкритим ключем

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Визначити актуальність теми даної кваліфікаційної роботи, провести аналіз даних щодо криптографічних систем захисту інформації, проаналізувати існуючі рішення, практично впровадити і проаналізувати запропоновані рішення, виявити основні можливості , переваги і недоліки відомих алгоритмів на основі асиметричного шифрування, реалізація завдання.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Алгоритм роботи асиметричної криптосистеми
Блок-схема роботи алгоритму на основі еліптичних кривих
Спроектована структура програмного модуля
Схема загроза-контрзахід

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання
Нормоконтроль	Муляр І.В., доцент кафедри КбКСМ		
Антиплагіат	Муляр І.В., доцент кафедри КбКСМ		

7. Дата видачі завдання «5» 02 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Пр
1.	Вибір та затвердження теми кваліфікаційної роботи	Лютий - 2 декада	
2.	Отримання завдання на кваліфікаційну роботу	Лютий - 2 декада	
3.	Проектування та розробка загальної архітектури і структури системи захисту	Березень - 3 декада	
4.	Програмна реалізація запропонованого рішення та тестування системи	Квітень - 1 декада	
5.	Виконання розрахункової частини	Квітень - 3 декада	
6.	Формулювання висновків	Квітень - 3 декада	
7.	Погодження розділів з консультантом з нормоконтролю	Травень - 1 декада	
8.	Оформлення пояснювальної записки	Травень - 2 декада	
9.	Попередній захист кваліфікаційної роботи	Травень - 3 декада	
10.	Доопрацювання кваліфікаційної роботи	Червень - 1 декада	
11.	Подання роботи для перевірки на плагіат	Червень - 1 декада	
12.	Захист кваліфікаційної роботи	Червень - 1 декада	

Студент Підпис А.А. Фаринюк Ініціали, прізвище

Керівник проекту (роботи) Підпис Ю.П. Кльоц Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Криптографічна система з відкритим ключем для захисту інформації при передачі через мережу».

Автор роботи: Фаринюк Анатолій Анатолійович.

Керівник роботи: Кльоц Юрій Павлович.

Пояснювальна записка: 61 с., 13 рис., 2 дод., 15 джерел.

Графічна частина: 4 плакати.

Метою кваліфікаційної роботи: є розробка та аналіз алгоритму шифрування інформації з використанням відкритого ключа.

З самого початку роботи я перевірив актуальність питання криптографічного захисту в системах мережевої безпеки, далі я зробив аналіз предметної області перевіривши існуючі рішення питань криптозахисту у мережі, в другому розділі проаналізував проблеми які пов'язують з криптосистемами з відкритим ключем та загрози, у третьому розділі я переглянув наявні найвідоміші і найефективніші рішення, алгоритми які використовують, в останньому розділі я реалізував власний програмний модуль та запропонував апаратне рішення для попередження загроз для даного алгоритму. Я думаю, що поставлена задача була мною виконана в повному обсязі.

Підпис студента

18.06.2021

Дата

Зона	Позиц	Позначення	Найменування	Кільк.	Прим.
	1		Завдання на дипломний	1	
			проект		
	2		Анотація	1	
	3	КвРКБ.170155.17.01.16 ПЗ	Криптографічна система	1	
			з відкритим ключем для		
			захисту інформації при		
			передачі через мережу		
			Пояснювальна записка		
	4	КвРКБ.170155.17.01.16 Е8	Алгоритм роботи	1	
			асиметричної		
			криптосистеми		
			Алгоритм роботи		
	5	КвРКБ.170155.17.01.16 Е8	Блок-схема роботи	1	
			алгоритму на основі		
			еліптичних кривих		
			Алгоритм роботи		
	6	КвРКБ.170155.17.01.16 Е8	Структура програмного	1	
			модуля		
			Алгоритм роботи		

КвРКБ.170155.17.01.16 ВП

Арк.	№ Докум.	Підп.	Дата
робив	Фаринюк А.А.		
ев.	Кльоц Ю.П.		
онтр.	Муляр І.В.		
	Кльоц Ю.П.		

Криптографічна система з відкритим ключем для захисту інформації при передачі через мережу. Відомість проекту

Літера	Аркуш	Аркушів
н	1	2

ХНУ, КБ-17-1

Формат	Зона	Позиц	Позначення	Найменування	Кільк.	Прим.
А2		7	КвРКБ.170155.17.01.16 Е8	Схема загроза-	1	
				контрзахід		
				Схема структурна		

Зм.	Арк.	№ Докум.	Підп.	Дата	КвРКБ.170155.17.01.16 ВП	Арк. 2

ЗМІСТ

ВСТУП

В умовах швидкого зростання цифрового зв'язку та електронного обміну даними сьогодні багато хто з нас спілкується у кіберпросторі, не замислюючись про безпеку. Ми обмінюємося великою кількістю приватної інформації та своїми секретами у віртуальному просторі. Хочемо ми цього чи ні, але наш цифровий слід знаходиться у віртуальному частині. Якщо бути точним, де б ми не спілкувались, будь-яка частина цього простору є незахищеною та відкритою для кіберзлочинців.

На мою думку, існує нагальна потреба у сучасній криптографії, щоб знайти способи захисту приватної або конфіденційної інформації. Ефективне шифрування та дешифрування даних є ключем до безпеки в кіберпросторі. Таким чином, нам потрібно перетворити інформацію в нечитабельний формат, щоб її можна було захищати та отримувати доступ лише для тих людей, які мають на це дозвіл. Криптографія є важливим інструментом захисту інформації, яка передається за допомогою комп'ютерів. Криптографія - це художнє перетворення даних у нечитабельний формат, щоб лише призначений одержувач міг їх зрозуміти та використовувати. Криптографія - це мистецтво та наука приховування важливої таємної інформації від несанкціонованих порушень [2].

У нашому повсякденному житті використання криптографії є скрізь. Наприклад, ми використовуємо його для надійної розсилки паролів через великі мережі для онлайн-покупок. Банківські сервери та поштові клієнти також зберігають ваші паролі за допомогою криптографії. Криптографія використовується для захисту всієї переданої інформації у нашому світі, пов’язаному з IoT, для автентифікації людей та пристроїв та пристроїв на інших пристроях. Якби всі криптографічні двигуни / функції перестали

працювати на день, сучасне життя, таке яке ми знаємо, мабуть зупинилося б. Банківські операції не проходили, інтернет-трафік зупинявся, стільникові телефони більше не функціонували. У цей момент буде викрита вся наша важлива інформація, яка потім може бути використана, щоб завдати немислимої шкоди всім нам [3].

Отже, якщо підсумувати, криптографія полягає у захисті інформації від кіберзлочинців або когось іншого, крім передбачуваного одержувача. Криптографія дозволяє людям спілкуватися в Інтернеті, надійно передаючи важливу та конфіденційну інформацію. Таким чином, криптографія дозволяє користувачам використовувати державні чи приватні засоби масової інформації, такі як Інтернет, робити покупки в Інтернеті та уникати того, щоб бути жертвами злочинців. Це досягається шляхом використанням новітніх технологічних досягнень в галузі інформатики. Криптографія, також відома як криптологія, таким чином допомагає користувачам та установам шифрувати та розшифровувати приховані повідомлення в коди, шифри та номери, щоб інформація могла безпечно передаватися. Криптографія ініціюється ключами шифрування та дешифрування [7]. Процес кодування та перетворення простого тексту в нечитабельний формат називається шифруванням; тоді як процес декодування та перетворення нечитабельного тексту в читабельну інформацію за допомогою спеціального цифрового ключа називається дешифруванням. Як я згадував раніше, єдиною метою криптографії є захист інформації, електронної пошти, даних кредитної картки та інших персональних даних, що передаються через загальнодоступну мережу [1-4].

Проаналізувавши питання криптографічного захисту, я вирішив обрати тему кваліфікаційної роботи пов’язану з криптографічною системою з відкритим ключем для захисту інформації при передачі через мережу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1Криптографія в мережевій безпеці

У наш час мережі стали глобальнішими, а інформація набула цифрової форми бітів і байтів. Зараз критична інформація зберігається, обробляється та передається у цифровій формі на комп’ютерних системах та відкритих каналах зв'язку.

Оскільки інформація відіграє таку важливу роль, противники націлюються на комп’ютерні системи та відкривають канали зв'язку, щоб або викрасти конфіденційну інформацію, або порушити критично важливу інформаційну систему.

Сучасна криптографія надає надійний набір методів, що дозволяють перешкоджати зловмисним намірам супротивника, забезпечуючи законним користувачам доступ до інформації [9].

Після людських ресурсів інформація виступає найважливішим активом організації. Усі зусилля щодо захисту систем та мереж намагаються докладати задля трьох результатів: доступності даних, цілісності та конфіденційності. І, як ми також бачили, жоден контроль безпеки інфраструктури не є на 100% ефективним. У розшарованій моделі безпеки часто доводиться впроваджувати остаточний контроль запобігання, обгорнутий навколо конфіденційної інформації: шифрування.

Криптографія - це наука, яка застосовує складну математику та логіку для розробки сильних методів шифрування. Досягнення надійного шифрування, приховування значення даних, також вимагає інтуїтивних стрибків, які дозволяють творчо застосовувати відомі або нові методи. Тож криптографію можна назвати мистецтвом [3].

Шифрування не є панацеєю безпеки. Воно не вирішить усіх ваших проблем безпеки, орієнтованих на дані. Швидше, це просто один контроль серед багатьох.

Рушійною силою, яка приховувала значення інформації, була війна. Сунь Цзи писав: «З усіх тих, хто в армії наближений до командира, ніхто не є більш близьким, ніж таємний агент; з усіх нагород не більш ліберальних, ніж ті, що даються секретним агентам; з усіх питань ніхто не є конфіденційним, ніж ті, що стосуються таємних операцій ".

Секретні агенти, польові командири та інші людські елементи війни вимагали інформації. Зберігання інформації, якою вони ділилися від ворога, допомагало забезпечити переваги маневру, часу та сюрпризу. Єдиним надійним способом зберегти інформацію в таємниці було приховати її значення [12].

Ранні криптографи використовували три методи для шифрування інформації: заміщення, транспонування та коди.

Переваги використання криптографії:

Криптографія - важливий інструмент захисту інформації. Він надає чотири основні послуги з інформаційної безпеки -

- Конфіденційність - техніка шифрування може захищати інформацію та спілкування від несанкціонованого розголошення та доступу до інформації.
- Аутентифікація - криптографічні методи, такі як MAC та цифрові підписи, можуть захистити інформацію від підробки та підробки.
- Цілісність даних - криптографічні хеш-функції відіграють важливу роль у забезпеченні користувачів щодо цілісності даних.
- Неможливість відмови - Цифровий підпис забезпечує послугу неможливість відмови від суперечностей, яка може виникнути через відмову в передачі повідомлення відправником.

Всі ці основні послуги, пропоновані криптографією, дозволили вести бізнес через мережі з використанням комп'ютерних систем надзвичайно ефективно.

Також варто згадати недоліки криптографії:

Окрім чотирьох основних елементів інформаційної безпеки, є й інші проблеми, які впливають на ефективне використання інформації -

- До сильно зашифрованої, автентичної та цифро підписаної інформації може бути важко отримати доступ навіть законному користувачеві у відповідальний момент прийняття рішення. Зловмисник може атакувати мережу або комп’ютерну систему та зробити її непрацездатною.
- Висока доступність, один з фундаментальних аспектів інформаційної безпеки, не може бути забезпечена використанням криптографії. Інші методи необхідні для захисту від таких загроз, як відмова в обслуговуванні або повний збій інформаційної системи.
- Ще одна фундаментальна потреба інформаційної безпеки вибіркового контролю доступу також не може бути реалізована за допомогою криптографії. Адміністративний контроль та процедури повинні здійснюватися для цього самого.
- Криптографія не захищає від вразливостей та загроз, які виникають внаслідок поганого проектування систем, протоколів та процедур. Вони повинні бути виправлені за допомогою належного проектування та створення оборонної інфраструктури.

- Криптографія має високу собівартість. Вартість вказана в термінах і грошах -
  1. Додавання криптографічних методів при обробці інформації призводить до затримки.

2. Використання криптографії з відкритим ключем потребує створення та обслуговування інфраструктури відкритих ключів, що вимагає гарного фінансового бюджету.

- Безпека криптографічної техніки базується на обчислювальній складності математичних задач. Будь-який прорив у вирішенні таких математичних задач або збільшення обчислювальної потужності може зробити криптографічну техніку вразливою.

Раніше, як метод звичної класики криптографія використовувала "безпеку за допомогою неясності" як спосіб захистити передану інформацію. У цих випадках використовувана техніка трималася в таємниці від усіх, крім кількох, звідси термін "неясність". Це зробило спілкування безпечним, але здійснити його в широкому масштабі було не дуже просто. Класичні криптографічні методи безпечні лише тоді, коли дві сторони можуть спілкуватися в безпечній екосистемі.

На рисунку 1.1 я зобразив класичну криптографічну систему, що називають шифром Вернама. Відправник та одержувач спочатку домовляються про набір загальнодоступних ключів шифрування / дешифрування. Потім ці ключі використовуються послідовно для шифрування, а потім дешифрування кожного наступного повідомлення [10].

Одноразова панель - це техніка шифрування, яка вимагає використання попередньо спільного ключа, який можна використовувати лише один раз. Цей же ключ повинен використовуватися для шифрування та дешифрування. Термін «one-time pad» - пішов від того, що кожна клавіша знаходиться на сторінці планшета, яка була використана, була потім знищена. Після того, як попередньо спільні ключі закінчилися, відправник та одержувач повинні зустрітися в безпечному місці, щоб надійно обмінятися новим набором ключів, а потім зберегти їх у надійному місці на час наступного набору обміну повідомленнями [8].

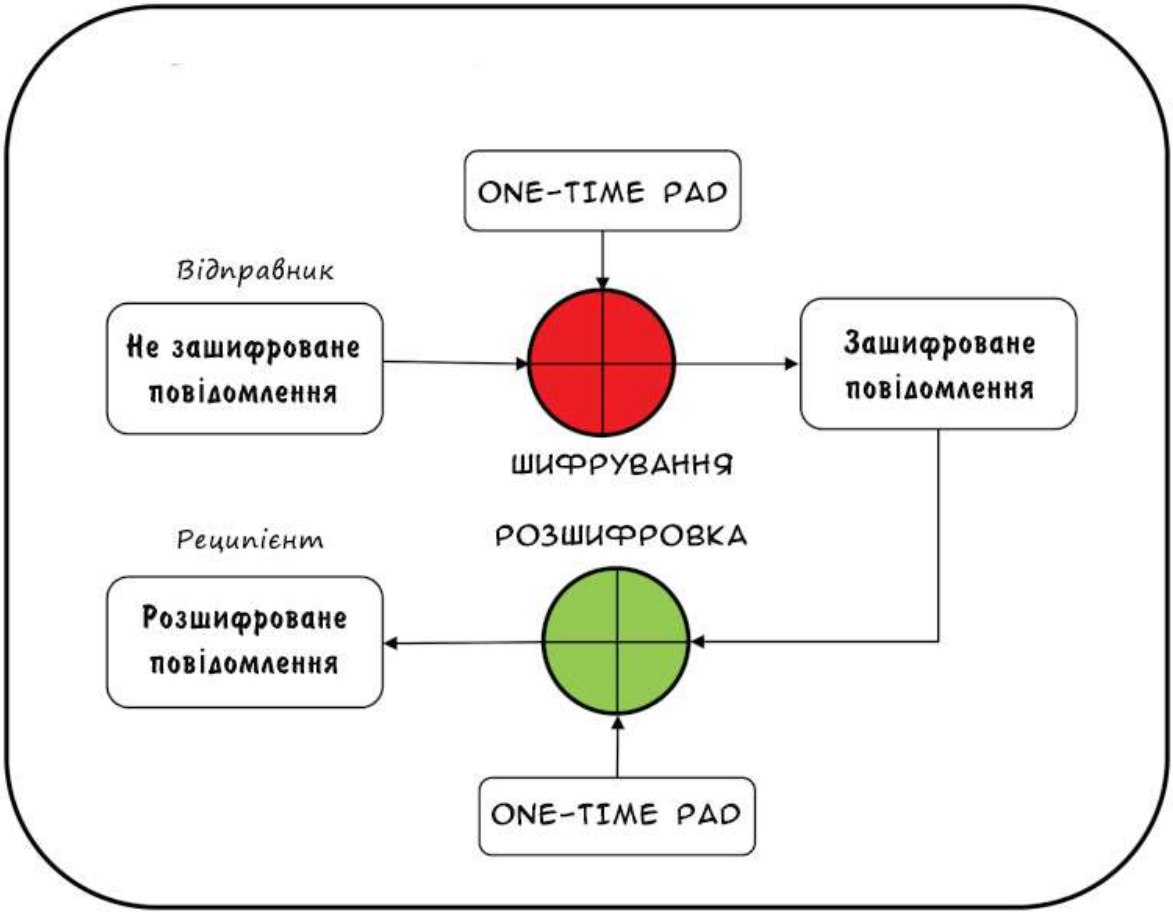


Рисунок 1.1 – Класична екосистема криптосистеми «Шифр Вернама»

Але очевидно, що застарілі класичні методи вже не життєздатні. Величезна система електронних комунікацій, комерції та інтелектуальних властивостей повинна бути захищена через океани та континенти, які інакше перехоплювали б люди з ворожими намірами.

У 1994 році Рада з архітектури Інтернету (IAB) опублікувала звіт "Безпека в архітектурі Інтернету" (RFC 1636). У звіті зазначено загальний консенсус щодо того, що Інтернет потребує більшої та кращої безпеки, а також визначено ключові сфери механізмів безпеки. Серед них були необхідність захистити мережеву інфраструктуру від несанкціонованого моніторингу та контролю мережевого трафіку та потреба захистити трафік від кінцевого користувача до кінцевого користувача за допомогою механізмів автентифікації та шифрування. Ці побоювання повністю

виправдані. Для їх підтвердження розглянемо тенденції, про які повідомляє Координаційний центр Комп’ютерної аварійної групи (CERT) (CERT / CC).. Сюди входять недоліки безпеки в операційних системах підключених комп’ютерів (наприклад, Windows, Linux), а також вразливості в Інтернет-маршрутизаторах та інших мережевих пристроях. Сюди входять атаки відмови в обслуговуванні; Підробка IP-адреси, при якій зловмисники створюють пакети з помилковими IP-адресами та експлуатують програми, які використовують аутентифікацію на основі IP; і різні форми прослуховування та нюху пакетів, коли зловмисники читають передану інформацію, включаючи інформацію про вхід та вміст бази даних.

Криптографія - це, мабуть, найважливіший аспект комунікаційної безпеки, і вона стає все більш важливою як основний будівельний матеріал для комп'ютерної безпеки.

Збільшення використання комп’ютерів та систем зв’язку в промислових системах збільшило ризик крадіжки власної інформації. Хоча ці загрози можуть вимагати різноманітних контрзаходів, шифрування є основним методом захисту цінної електронної інформації.

На сьогоднішній день найважливішим автоматизованим інструментом безпеки мережі та зв'язку є шифрування. Загальновживаними є дві форми шифрування: звичайна або симетрична або шифрування з відкритим ключем, або асиметричне шифрування.

Симетричне шифрування, яке також називають звичайним шифруванням або шифруванням за одним ключем, було єдиним типом шифрування, що використовувався до розробки шифрування з відкритим ключем у 1970-х роках. Це залишається далеко не найпоширеніший із двох типів шифрування. Оригінальне повідомлення відоме як відкритий текст, тоді як закодоване повідомлення називається зашифрованим текстом. Процес перетворення з відкритого тексту в зашифрований текст відомий

як шифрування або зашифровка; відновлення відкритого тексту із зашифрованого тексту - це дешифрування або розшифровка. Багато схем, що використовуються для шифрування, становлять область дослідження, відому як криптографія. Така схема відома як криптографічна система або шифр. Прийоми, що використовуються для розшифровки повідомлення без будь-якого знання деталей шифрування потрапляє в область криптоаналізу. Криптоаналіз - це те, що неспеціаліст називає "порушенням коду". Області криптографії та криптоаналізу разом називаються криптологією.

Майбутнє криптографії:

Криптографія з еліптичною кривою (ECC) вже винайдена, але її переваги та недоліки ще не повністю зрозумілі. ECC дозволяє виконувати шифрування та дешифрування за значно менший час, таким чином дозволяючи передавати більший обсяг даних з однаковою безпекою. Однак, як і інші методи шифрування, ECC також повинен бути перевірений та доведений у безпеці, перш ніж він буде прийнятий для державного, комерційного та приватного використання.

Квантові обчислення - це нове явище. У той час як сучасні комп’ютери зберігають дані у двійковому форматі, який називається «біт», в якому може зберігатися «1» або «0»; квантовий комп'ютер зберігає дані за допомогою квантової суперпозиції декількох станів. Ці багатозначні стани зберігаються у "квантових бітах" або "кубітах". Це дозволяє обчислювати числа на кілька порядків швидше, ніж традиційні транзисторні процесори.

Щоб зрозуміти потужність квантового комп’ютера, розглянемо RSA-640, число з 193 цифрами, яке можна врахувати за допомогою вісімдесяти комп’ютерів із частотою 2,2 ГГц протягом 5 місяців, і один квантовий комп’ютер буде мати коефіцієнт менш ніж за 17 секунд. Числа, для обчислення яких, як правило, знадобляться мільярди років, за допомогою

повністю розробленого квантового комп'ютера може знадобитися лише кілька годин або навіть хвилин.

З огляду на ці факти, сучасній криптографії доведеться шукати обчислювально-складніші рішення проблеми або розробляти абсолютно нові методи архівування цілей, які в даний час потребують сучасній криптографії.

1.2 Існуючі рішення для криптографії

Загалом в світі використовуються численні криптографічні алгоритми, але повністю їх можна розділити на три категорії: криптографія секретного ключа, криптографія відкритого ключа та хеш-функції. Кожен має свою роль у криптографічному пейзажі інформаційного світу.

Криптографія секретного ключа. Шифр Цезаря є чудовим прикладом криптографії секретного ключа. Шифри Цезаря використовують метод заміщення, де літери в алфавіті зміщуються на деяку фіксовану кількість пробілів, щоб отримати алфавіт кодування. Шифр Цезаря зі зміщенням11кодував би A як B, M як N і Z як A тощо. Метод названий на честь римського лідера Юлія Цезаря, який використовував його у своєму приватному листуванні. Шифр Цезаря дуже легко спроектувати, але також дуже легко декодувати.

Криптографія секретного ключа, яку іноді також називають симетричним ключем, широко використовується для збереження конфіденційності даних. Наприклад, це може бути дуже корисно для збереження приватного локального жорсткого диска; оскільки один і той самий користувач зазвичай шифрує та дешифрує захищені дані, спільний доступ до секретного ключа не є проблемою. Криптографія секретного ключа також може використовуватися для збереження конфіденційності

повідомлень, переданих через Інтернет; однак, щоб успішно це здійснити, вам потрібно розгорнути нашу наступну форму криптографії в тандемі з нею.

Криптографія з відкритим ключем. Можливо, Цезар міг особисто поспілкуватися зі своїми сотниками, але ви не хочете заходити до свого банку та розмовляти з касиром, щоб лише дізнатись, що таке приватний ключ для шифрування вашого електронного спілкування з банком - це перемогло б призначення Інтернет-банкінгу. Загалом, для того, щоб безпечно функціонувати, Інтернету потрібен спосіб для спілкування сторін для встановлення захищеного каналу зв'язку, лише спілкуючись між собою через невпевнену в собі мережу. Це працює за допомогою криптографії з відкритим ключем.

У криптографії відкритого ключа, яку іноді також називають асиметричним ключем , кожен учасник має два ключі. Один з них є загальнодоступним і надсилається кожному, з ким сторона бажає спілкуватися. Цей ключ використовується для шифрування повідомлень. Але інший ключ є приватним, з ним ні з ким не ділиться. Цим ключем необхідно розшифрувати ці повідомлення. Можна використати метафору: розгляньте відкритий ключ як дірку для листа на поштовій скриньці. Ви надаєте ці розміри кожному, хто, на вашу думку, може надіслати вам лист. Закритий ключ - це те, що ви використовуєте для відкриття поштової скриньки, щоб можна було дістати листи.

Математика того, як ви можете використовувати один ключ для шифрування повідомлення, а другий для його розшифровки, набагато менш інтуїтивна, ніж те, як працює ключ до шифру Цезаря. Основний принцип, який змушує процес працювати, полягає в тому, що два ключі насправді пов'язані між собою математично так, що легко отримати відкритий ключ із закритого ключа, але не навпаки. Наприклад, приватний

ключ може бути двома дуже великими простими числами, які ви множите разом, щоб отримати відкритий ключ.

Обчислення, необхідні для криптографії відкритих ключів, набагато складніші та ресурсоємніші, ніж ті, що стоять за інфраструктурою секретних ключів. На щастя, вам не потрібно використовувати його для захисту кожного повідомлення, яке ви надсилаєте в Інтернеті. Натомість зазвичай трапляється так, що одна сторона буде використовувати криптографію з відкритим ключем для шифрування повідомлення, що містить ще один криптографічний ключ. Цей ключ, безпечно переданий через незахищений Інтернет, стане приватним ключем, який кодує набагато довший сеанс зв'язку, зашифрований за допомогою шифрування секретного ключа.

Таким чином, криптографія з відкритим ключем допомагає забезпечити конфіденційність. Але ці відкриті ключі також є частиною більшого набору функцій, відомих як інфраструктура відкритих ключів , або PKI. PKI надає способи переконатися, що будь-який відкритий ключ пов'язаний з конкретною особою чи установою. Повідомлення, зашифроване відкритим ключем, таким чином підтверджує особу відправника, встановлюючи автентифікацію та попереджує відмову в обслуговуванні.

Хеш-функції. Криптографічні алгоритми відкритого та приватного ключів передбачають перетворення відкритого тексту в зашифрований текст, а потім назад у відкритий текст. Навпаки, хеш-функція є одностороннім алгоритмом шифрування: коли ви зашифрували свій відкритий текст, ви ніколи не зможете відновити його з отриманого зашифрованого тексту (іменованого як хеш).

Це може змушує думати, що створення хеш-функції це безглузде заняття. Цікавість такої функції полягає в тому, що для будь-якої заданої хеш-функції жодних два відкритих тексти не створюватимуть однаковий

хеш. (Математично це не зовсім правильно, але для будь-якої хеш-функції, яка фактично використовується, шанси на її виникнення, як правило, зникають і є зовсім незначними, тому їх можна сміливо ігнорувати.)

Це робить алгоритми хешування чудовим інструментом для забезпечення цілісності даних. Наприклад, повідомлення можна надіслати разом із власним хешем. Отримавши повідомлення, ви можете запустити той самий алгоритм хешування тексту; якщо хеш, який ви створюєте, відрізняється від того, що супроводжує повідомлення, ви знаєте, що повідомлення було змінено під час передачі.

Хешування також використовується для забезпечення конфіденційності паролів. Зберігання паролів як відкритого тексту - це велика небезпека, тому що це робить користувачів схильними до крадіжки облікових записів та особистих даних. Якщо замість цього ви збережете хешовану версію пароля користувача, хакери не зможуть його розшифрувати та використовувати в іншому місці, навіть якщо їм вдасться зламати ваш захист. Коли законний користувач входить зі своїм паролем, ви можете просто хешувати його та перевірити, чи є у вас хеш, який ви маєте у збереженому файлі.

1.3 Асиметричне шифрування в мережевій безпеці

З практичних міркувань бажано використовувати різне шифрування та ключі дешифрування в криптосистемі. Такі асиметричні системи дозволяють зробити ключ шифрування доступним кожному, зберігаючи впевненість, що розшифрувати інформацію можуть лише люди, які мають ключ дешифрування.

Після симетричного шифрування іншою основною формою шифрування є шифрування із відкритим ключем, яке революціонізувало для безпеки зв'язку. Пов’язана криптографічна область - це криптографічні

хеш-функції. Хеш-функції використовуються разом із симетричними шифрами для цифрових підписів. Крім того, хеш-функції використовуються для автентифікації повідомлень. Симетричні шифри також використовуються для управління ключами.

Розробка криптографії з відкритим ключем - це найбільша і, мабуть, єдина справжній революційний прогрес за всю історію криптографії. Від самого початку існування до сучасності практично всі криптографічні системи базувались на елементарних інструментах заміщення та перестановки. Після тисячоліть роботи з алгоритмами, які по суті можна було розрахувати вручну, з розвитком машини шифрування / дешифрування ротора відбувся значний прогрес у симетричній криптографії. Електромеханічний ротор дозволив розробити начисто складні шифрові системи. Завдяки наявності комп'ютерів були розроблені ще більш складні системи, найвизначнішою з яких була робота Люцифера в IBM, яка завершилася стандартом шифрування даних (DES). Але обидва ротора машини і DES, хоч і представляли значний прогрес, все ж покладались на простенькі інструменти заміщення та перестановки.

Криптографія з відкритим ключем забезпечує радикальний відхід від усього, що було раніше. З одного боку, алгоритми publickey засновані на математичних функціях, а не на заміщення та перестановку. Що ще важливіше, криптографія з відкритим ключем є асиметричною, включаючи використання двох окремих ключів, на відміну від симетричного шифрування, яке використовує лише один ключ. Як ми побачимо, використання двох ключів має глибокі наслідки у сферах конфіденційності, розподілу ключів та автентифікації.

На мою думку, слід згадати кілька типових помилок щодо систем шифрування з відкритим ключем. Одне з таких помилкових уявлень полягає в тому, що шифрування з відкритим ключем є більш захищеним від криптоаналізу, ніж симетричне шифрування. Насправді, безпека будь-

якої схеми шифрування залежить від довжини ключа і обчислювальної роботи, пов’язаної з розбиттям шифру. В принципі, ні симетричне шифрування, ні шифрування з відкритим ключем не робить одне вищим над іншим з точки зору опору криптоаналізу.

Друга помилкова думка полягає в тому, що шифрування з відкритим ключем - це метод загального призначення, який зробив симетричне шифрування застарілим. Навпаки, через обчислювальні накладні витрати на поточні схеми шифрування з відкритим ключем, здається, немає передбачуваної ймовірності відмови від симетричного шифрування. Як висловився один із винахідників шифрування з відкритим ключем, "обмеження криптографії відкритим ключем до програм управління ключами та підписів є майже загальновизнаним".

Нарешті, існує відчуття, що розподіл ключів є тривіальним при використанні шифрування з відкритим ключем, порівняно з досить громіздким рукостисканням, пов’язаним із центрами розподілу ключів для симетричного шифрування. Насправді потрібна якась форма протоколу, як правило, із залученням центрального агента, а залучені процедури не простіші та не ефективніші, ніж ті, що потрібні для симетричного шифрування.

1.4 Принципи криптосистем з відкритим ключем

Концепція криптографії з відкритим ключем виникла із спроби атакувати дві найскладніші проблеми, пов'язані з симетричним шифруванням. Перша проблема полягає в розподілі ключів. Як ми вже бачили, розподіл ключів при симетричному шифруванні вимагає або (1), що двох людей які вже мають цей ключ, або якимось чином їм був розданий; або (2) використання центру розподілу ключів. Уітфілд Діффі, один з першовідкривачів шифрування відкритих ключів (разом із

Мартіном Хеллманом, обидва тоді були в Стенфордському університеті), міркував, що ця друга вимога заперечує саму суть криптографії: здатність зберігати повну таємницю щодо власного спілкування. Як висловився Діффі, "яка б користь зрештою для розвитку непроникних криптосистем, якщо їх користувачі були змушені ділитися своїми ключами з KDC, що може бути скомпрометовано або крадіжкою, або повісткою в суд? ".

Другою проблемою, про яку міркував Діффі, була проблема "цифрових підписів". Якщо використання криптографії мало набути широкого поширення не лише у військових, але для комерційних та приватних цілей, тоді електронні повідомлення та документи потребували б еквівалента підписів, що використовуються у паперових документах. Тобто, чи можна розробити метод, який би передбачав, на задоволення всіх сторін, що цифрове повідомлення було надіслане конкретною особою?

1.5 Висновки

Після аналізу вище перерахованого можна зробити висновок, що системи з відкритим ключем набули досить велику популярність сьогодні. Вони є досить потужним і надійним через системи з використанням закритого ключа, що майже унеможливлює злам повідомлення зловмисником. Тому я вважаю, що потрібно детальніше розглянути криптографічні системи з відкритим ключем і побачити які проблеми їх переслідують сьогодні.

2 АНАЛІЗ КРИПТОСИСТЕМ ЗАХИСТУ ПОВІДОМЛЕНЬ З ВИКОРИСТАННЯМ ВІДКРИТОГО КЛЮЧА

2.1 Аналіз проблем криптосистем з відкритим ключем

Системи шифрування, що використовуються бізнесом, поділяються на дві широкі категорії. Системи із закритим ключем або секретним ключем використовують той самий ключ для шифрування та дешифрування даних, тому вам потрібно тримати свій ключ прихованим, щоб ніхто інший не мав до нього доступу. У системі відкритих ключів ви використовуєте два ключі. За даними Cloudflare , ваш приватний ключ, який ви зберігаєте прихованим, розшифровує дані, але відкритий ключ використовується для шифрування даних.

Оскільки відкритий ключ, по суті, не використовується для кодування інформації, ви можете безпечно поділитися нею з будь-ким. Хоча це добре працює у ситуаціях, коли ви не можете надійно надати спільний доступ до ключа, наприклад через Інтернет, є деякі недоліки шифрування, яке використовує відкритий ключ [13].

1. Можливі недоліки продуктивності шифрування

Шифрування відкритим ключем працює дуже добре і надзвичайно безпечно, але воно базується на складній математиці. Через це комп’ютерам у минулому доводилося дуже багато працювати, щоб як шифрувати, так і дешифрувати дані за допомогою системи. У програмах, де вам потрібно було регулярно працювати з великою кількістю зашифрованих даних, обчислювальні накладні витрати означали, що системи відкритих ключів можуть бути дуже повільними.

На щастя, це не така велика проблема сьогодні, як системи працюють набагато швидше. Однак TechBeacon попереджає, що слід використовувати належні методи шифрування, щоб ви не відчували

багатьох недоліків швидкості шифрування за допомогою відкритого ключа. Сюди входить використання якомога менше мережевих з'єднань і прив'язка до близьких серверів, коли це можливо.

2. Потенційні проблеми сертифікації

У багатьох системах відкритих ключів використовується третя сторона для підтвердження надійності відкритих ключів. Наприклад, якщо ви мали б зашифрувати конфіденційні корпоративні дані для надсилання на комп'ютер свого адвоката, ви хотіли б бути впевнені, що комп'ютер, на який ви його надсилали, справді пов'язаний з його юридичною фірмою. Третя сторона, яку називають сертифікаційним органом, цифрово підписує свій відкритий ключ, перетворюючи його на цифровий сертифікат, щоб ви могли бути впевнені, що він безпечний у використанні.

Однак, якщо орган сертифікації зазнає компрометації, злочинець, який вчинив це, може видавати фальшиві сертифікати і обдурити людей надсиланням даних не в те місце. Це вже сталося.

3. Потенціал для прямого компромісу

Існує два способи зламати дані, зашифровані за допомогою системи відкритих ключів. Перший - знайти діру в основній математиці, яку можна використати для розбиття шифру. На дату публікації про таку діру не відомо загальновідомо.

Інший спосіб зламати шифрування - вгадати правильний ключ. Академія Хан пояснює, що шифрування відкритим ключем працює на основі наявності надзвичайно великої кількості операцій, яка походить від множення великої кількості прихованої у відкритому ключі, на велику кількість, прихованої у приватному ключі. Отже, якщо ви зможете врахувати це надзвичайно велике число, ви можете порушити шифрування.

По мірі того, як комп'ютери стають потужнішими, а квантові обчислення, які використовують світло для створення ще більшої швидкості, ніж

традиційні суперкомп'ютери, стають реальністю, атаки грубої сили на зашифровані дані із відкритим ключем стають практичними.

4. Помилкове відчуття безпеки

Якою б безпечною не була ваша криптографічна система з відкритим ключем, вона захищає лише те, що призначена для захисту. Наприклад, коли ваші клієнти надсилають вам дані своїх кредитних карток через Інтернет, цей переказ захищений сумішшю шифрування відкритого та приватного ключів і надзвичайно безпечний. Однак, як тільки ви отримаєте дані цієї кредитної картки, якщо залишите комп'ютер з доступом до свого сервера відкритим, хтось може сісти за клавіатуру, завантажити всі надійно передані дані та викрасти їх. Шифрування відкритим ключем не захистить захисту від такого, і як таке, це лише частина загальної системи безпеки.

5. Інші слабкості

Ключів у криптографії з відкритим ключем, завдяки їх унікальній природі, набагато більше ніж у їх аналогів в криптографії із секретними ключами. Асиметричні ключі також повинні бути у багато разів довші, щоб мати еквівалентну їм безпеку [5]. Ключі в асиметричній криптографії також вразливіші до атак грубої сили, ніж у криптографії із секретним ключем. Існують алгоритми криптографії з відкритим ключем, які дозволяють зловмисникам зламувати приватні ключі швидше, ніж потрібно методом грубої сили. Широко використовуваний та новаторський алгоритм RSA має такий алгоритм, який робить його сприйнятливим до атак менш ніж за час грубої сили [3]. Хоча генерація довших ключів в інших алгоритмах, як правило, перешкоджає успіху атаки грубої сили за будь-який значущий проміжок часу, ці обчислення стають більш вираховуваними. Ефективність цих довших клавіш все одно може відрізнятися залежно від обчисленої потужності, доступної зловмиснику. Криптографія з відкритим ключем також має вразливості до

атак, таких як людина в середині атаки (Man-in-the-Middle)[3]. У цій ситуації зловмисна третя сторона перехоплює відкритий ключ на шляху до однієї із залучених сторін. Потім третя сторона може передати власний відкритий ключ із повідомленням, яке стверджує, що воно було від оригінального відправника. Зловмисник може використовувати цей процес на кожному кроці обміну, щоб успішно видавати себе за кожного учасника розмови, не маючи жодної сторони, яка знає про цей обман. [3]

2.2 Аналіз загроз для криптографічного алгоритму з відкритим ключем

Криптографія лежить в основі сучасного бізнесу - захист електронних комунікацій та фінансових транзакцій, підтримка конфіденційності конфіденційних даних та забезпечення безпечної автентифікації та авторизації. Нові правила , як GDPR і PSD2 , комерційне тиск для цифрового перетворення, прийняття хмарних технологій і останні тенденції в IoT і blockchain / DLT все допомагає керувати потрібно вставляти криптографія практично в будь-який області застосування - від тостерів до основних банківських систем!

Хороша новина полягає в тому, що сучасні криптографічні алгоритми при правильній реалізації дуже стійкі до атак - їх єдиним слабким місцем є ключі. Однак, якщо ключ скомпрометований, значить, гра закінчена! Це робить такі криптографічні ключі одним із найцінніших активів вашої компанії, і їх слід розглядати як такі. Вартість будь-якого ключа еквівалентна вартості всіх даних та / або активів, які він використовує для захисту.

Існує три основних типи ключів, які потрібно захищати:

- Симетричні ключі - зазвичай використовуються для шифрування масових даних за допомогою симетричних алгоритмів, таких як

3DES або AES; кожен, хто має секретний ключ, може розшифрувати дані

- Приватні ключі - таємна половина пар відкритих / приватних ключів, що використовуються в криптографії з відкритими ключами з асиметричними алгоритмами, такими як RSA або ECDSA; кожен, хто має приватний ключ, може видавати себе за власника приватного ключа, щоб розшифрувати приватні дані, отримати несанкціонований доступ до систем або створити шахрайський цифровий підпис, який видається справжнім
- Хеш-ключі - використовуються для захисту цілісності та достовірності даних та транзакцій за допомогою таких алгоритмів, як HMAC-SHA256; кожен, хто має секретний ключ, може видавати себе за ініціатора даних / транзакцій і, таким чином, змінювати вихідні дані / транзакції або створювати повністю неправдиві дані / транзакції, які будь-який одержувач вважатиме автентичними

Завдяки дедалі більшій кількості ключів для захисту та постійному зростанню цінності даних, що захищаються цими ключами, не кажучи вже про вимоги PCI-DSS або GDPR, це виклик, з яким стикається та вирішує майже кожен бізнес в терміновому порядку.

Є багато загроз, які можуть призвести до компрометації ключа - як правило, ви навіть не будете знати, що ключ був скомпрометований, поки його не використає зловмисник, що робить загрози ще більш небезпечними.

Виділимо деякі основні загрози для ключів алгоритму (рис 2.1):

1. Слабкі ключі

Ключ - це, по суті, просто випадкове число - чим довше і випадковіше воно, тим складніше його зламати. Міцність ключа повинна відповідати значенню даних, які він захищає, та періоду часу, протягом якого його потрібно захищати. Ключ повинен бути достатньо довгим за прямим

призначенням і генеруватися за допомогою високоякісного (ідеально сертифікованого) генератора випадкових чисел (RNG), що ідеально збирає ентропію з відповідного апаратного джерела шуму .

Є багато випадків, коли погане впровадження RNG призводило до ключових вразливостей.

2. Неправильне використання ключа

Кожен ключ повинен генеруватися для однієї конкретної мети (тобто передбачуваної програми та алгоритму) - якщо він використовується для чогось іншого, він може не забезпечити очікуваний або необхідний рівень захисту.

3. Повторне використання ключа

Неправильне повторне використання ключів за певних обставин може полегшити зловмисникові злом ключа.

4. Не обертання ключів

Якщо ключ використовується надмірно (наприклад, використовується для шифрування забагато даних), це робить ключ більш уразливим до злому , особливо при використанні старих симетричних алгоритмів; це також означає, що у випадку компрометації ключів може бути відкритий великий обсяг даних. Щоб цього уникнути, клавіші слід обертати (тобто оновлювати) через відповідні проміжки часу.

5. Невідповідне зберігання ключів

Ключі ніколи не слід зберігати поруч із даними, які вони захищають (наприклад, на сервері, в базі даних тощо), оскільки будь-яке розповсюдження захищених даних також може скомпрометувати ключ.

6. Неадекватний захист ключів

Навіть ключі, що зберігаються лише в пам'яті сервера, можуть бути вразливими до компромісів. Там, де це вимагає цінність даних, ключі слід шифрувати щоразу, коли вони зберігаються, і надавати їх у

незашифрованому вигляді лише в захищеному середовищі та навіть (у крайньому випадку) зберігати в автономному режимі.

Існує низка вразливих місць, які можуть відкрити криптографічні ключі в пам'яті сервера, включаючи Heartbleed , Flip Feng Shui та Meltdown / Spectre .

7. Невпевнена передача ключів

Часто доводиться пересувати ключ між системами. Це повинно бути здійснено шляхом шифрування ("обгортання") ключа під загальнодоступним транспортним ключем (ключ шифрування ключа або KEK), який може бути як симетричним, так і асиметричним. Якщо це неможливо (наприклад, при спільному використанні симетричних транспортних ключів для завантаження системи), ключ слід розділити на декілька компонентів, які потім слід тримати окремо до повторного введення в цільову систему (а потім компоненти знищуються).

8. Не знищення ключів

Ключі слід знищувати (тобто надійно видаляти, не залишаючи слідів) після закінчення терміну їх дії, якщо це явно не потрібно для подальшого використання (наприклад, для дешифрування даних). Це усуває ризик випадкових компромісів у майбутньому.

9. Інсайдерські загрози (автентифікація користувача, подвійне управління, розподіл ролей)

Одним з найбільших класів загроз, з якими стикається ключова особа, є інсайдерські загрози. Якщо неправомірний працівник має вільний доступ до ключа, він може використовувати його зі зловмисною метою або передати комусь іншому з тим самим кінцем.

10. Відсутність стійкості

Необхідно захищати не лише конфіденційність та цілісність ключів, але й їх наявність. Якщо ключ недоступний при необхідності або, що ще гірше, все-таки втрачений через якусь несправність, аварію чи катастрофу без

резервної копії, то захищені дані також можуть бути недоступними / втраченими.

11.Відсутність реєстрації аудиту

Якщо життєвий цикл ключа не буде повністю записаний або зареєстрований, буде важче визначити, коли стався компроміс і будь-яке подальше судово-медичне розслідування буде заважати.

12.Процеси управління ключами вручну

Використання процесів керування ключами вручну, використання паперу або невідповідних інструментів, таких як електронні таблиці, а також супровід ручних церемоній введення ключів може легко призвести до людських помилок, які часто залишаються непоміченими та можуть залишати ключі вразливими.

Рисунок 2.1 – Загрози для ключів алгоритму

Якщо зробити висновок щодо проблем і загроз криптосистем захисту з відкритим ключем, то стає очевидним що сам алгоритм не є таким простим для зламу і не має багато слабких місць. В основному, якщо говорити про загрози то вони стосуються більше людського фактору, менеджементу і тд. Тому, я вважаю, що такі системи є досить надійними і тому потрібно розглянути вже існуючі асиметричні алгоритми і побачити їх відмінності і переваги.

Щоб поглибитися ще сильніше в алгоритми шифрування з відкритим ключем згадаємо найвідоміші з них які досить часто використовуються для шифрування повідомлень в мережах.

2.3 Алгоритм RSA

У 1978 році Рон Рівест, Аді Шамір та Леонард Адльман запровадили криптографічний алгоритм, який, по суті, повинен був замінити менш безпечний алгоритм Національного бюро стандартів (НБС). Більшість що важливо, RSA реалізує криптосистему з відкритим ключем, а також цифрові підписи. RSA мотивовано опублікованими роботами Діффі та Хеллмана за кілька років до цього, які описали ідею такого алгоритму, але ніколи по-справжньому її не реалізовували. Введена в той час, коли здавалося, що незабаром настане ера електронної пошти, RSA реалізував дві важливі ідеї:

1. Шифрування відкритим ключем (рисунок 2.2). Ця ідея опускає необхідність “кур’єра” доставляти ключі одержувачам за іншим захищеним каналом перед передачею спочатку призначеного повідомлення. У RSA ключі шифрування є загальнодоступними, тоді як ключі дешифрування - ні, тому лише особа з правильним ключем дешифрування може розшифрувати зашифроване повідомлення. У кожного свої ключі шифрування та розшифрування. Ключі повинні бути

зроблені таким чином, щоб ключ розшифрування не міг бути легко виведений із відкритого ключа шифрування.

2. Цифрові підписи. Одержувачу може знадобитися підтвердження, що передане повідомлення насправді походить від відправника (підпис), а не просто надходить незрозуміло звідти і від кого (автентифікація). Це робиться за допомогою ключа розшифровки відправника, а підпис згодом може перевірити будь-хто, використовуючи відповідний відкритий ключ шифрування. Тому підписи не можна підробляти. Крім того, жоден відправник такого ЦП пізніше не може заперечити, що підписав повідомлення.

Рисунок 2.2 – Алгоритм роботи RSA

2.4 Цифровий підпис

Digital signature algorithm або Електронний цифровий підпис був введений в 1991 році Національним інститутом стандартів і технологій (NIST), алгоритм цифрового підпису є стандартом для цифрових підписів. Більшість типів цифрового підпису використовуються для підписання повідомлень із використанням приватного ключа ініціатора повідомлення.

Оскільки підписується лише дайджест повідомлення, підпис, як правило, набагато менший, ніж дані, що підписуються [5].

З іншого боку, DSA не використовує приватний ключ для шифрування повідомлень, а також не використовує відкритий ключ для дешифрування повідомлень. Натомість DSA використовує унікальні математичні функції, які створюють цифровий підпис з двома 160-бітними номерами. Ці номери походять із дайджестів повідомлень та приватного ключа. Хоча DSA і використовує відкритий ключ для автентифікації підпису, процес перевірки та автентифікації часто є більш тривалим і складним у порівнянні з іншими типами цифрових підписів.

DSA - це пара великих чисел. Номери генеруються в межах визначеного алгоритму, який забезпечує автентифікацію підписанта. Підписи генеруються із використанням приватного ключа, а перевірка використовує відкритий ключ. Кожен підписант матиме власний відкритий та приватний ключ. Однак підпис може бути сформована лише уповноваженою особою за допомогою її приватного ключа, а відповідний відкритий ключ може бути використаний будь-ким для перевірки підпису.

Дайджест повідомлення (також відомий як зведення інформації) створюється за допомогою хеш-функції (стандарт Secure Hash). Цей результат спільно з алгоритмом DSA використовується для створення цифрового підпису, який надсилається разом із повідомленнями. Процес перевірки також використовує ту саму хеш-функцію.

Алгоритм цифрового підпису (DSA) є одним із Федеральних стандартів обробки інформації для створення цифрових підписів, що залежить від математичної концепції, або ми можемо сказати формули модульної експоненції та задачі дискретного логарифму для криптографії цифрового підпису в цьому алгоритмі.

Саме Цифрові підписи - це примітиви відкритого ключа для автентифікації повідомлень у криптографії. Насправді, у фізичному світі

на сьогодні прийнято використовувати рукописні підписи на рукописних або набраних повідомленнях. В основному вони використовуються для прив'язки підписанта до повідомлення для захисту повідомлення.

Можна сказати, що цифровий підпис - це техніка, яка пов’язує особу чи організацію з цифровими даними підпису. Тепер це прив'язування може бути незалежно перевірено одержувачем, а також будь-якою третьою стороною для доступу до цих даних.

Цифровий підпис - це криптографічне значення, яке обчислюється за даними та секретним ключем, відомим лише відправнику або особі, чий підпис це [15].

Насправді в реальному світі одержувач повідомлення потребує впевненості у тому, що повідомлення належить відправнику, і він не повинен мати можливість зламати походження цього повідомлення для неправильного використання чи чогось іншого. Їх вимога є дуже важливою в бізнес-додатках або будь-яких інших речах, оскільки ймовірність суперечки щодо обмінюваних даних дуже велика для забезпечення цих даних.

На рисунку 2.3 я зобразив блок-схему цифрового підпису.

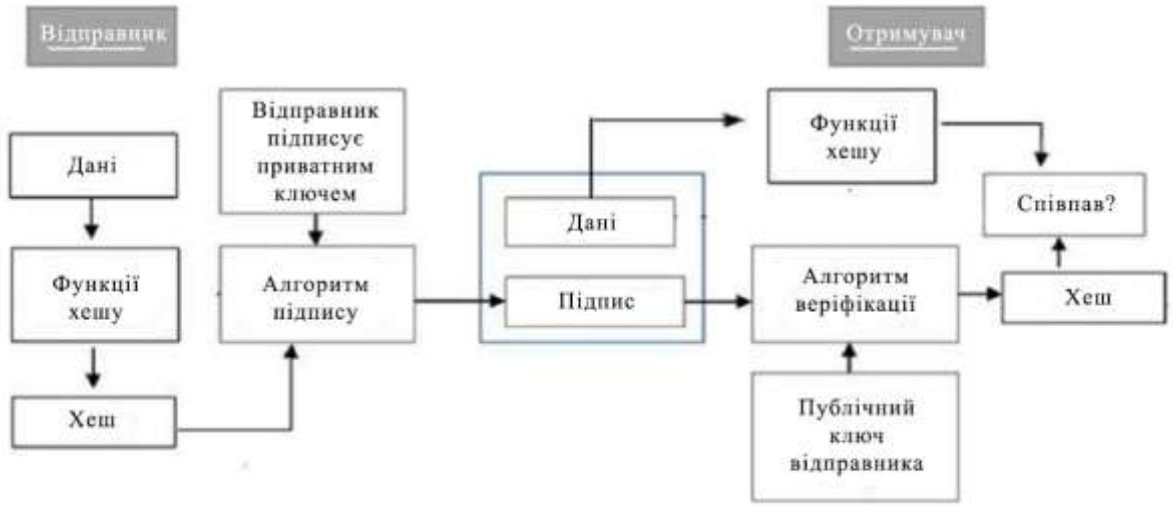


Рисунок 2.3 – Блок-схема цифрового підпису

- По-перше, кожна особа, яка приймає цю схему, має пару криптографічного відкрито-приватного ключа.
- Пари ключів, що використовуються для шифрування або дешифрування та підписання або перевірки, різні для кожного підпису. Тут приватний ключ, що використовується для підписання, позначається як ключ підпису, а відкритий ключ як ключ перевірки в цьому алгоритмі.
- Потім відправник підписує дані до хеш-функції та генерує хеш даних цього повідомлення [5].
- Тепер значення хешу та ключ підпису подаються до алгоритму підпису, який виробляє цифровий підпис на заданому хеші цього повідомлення. Цей підпис додається до даних, а потім обидва надсилаються отримувачу для захисту цього повідомлення. Прикладом такого буде попередній алгоритм РСА (рисунок 2.4)

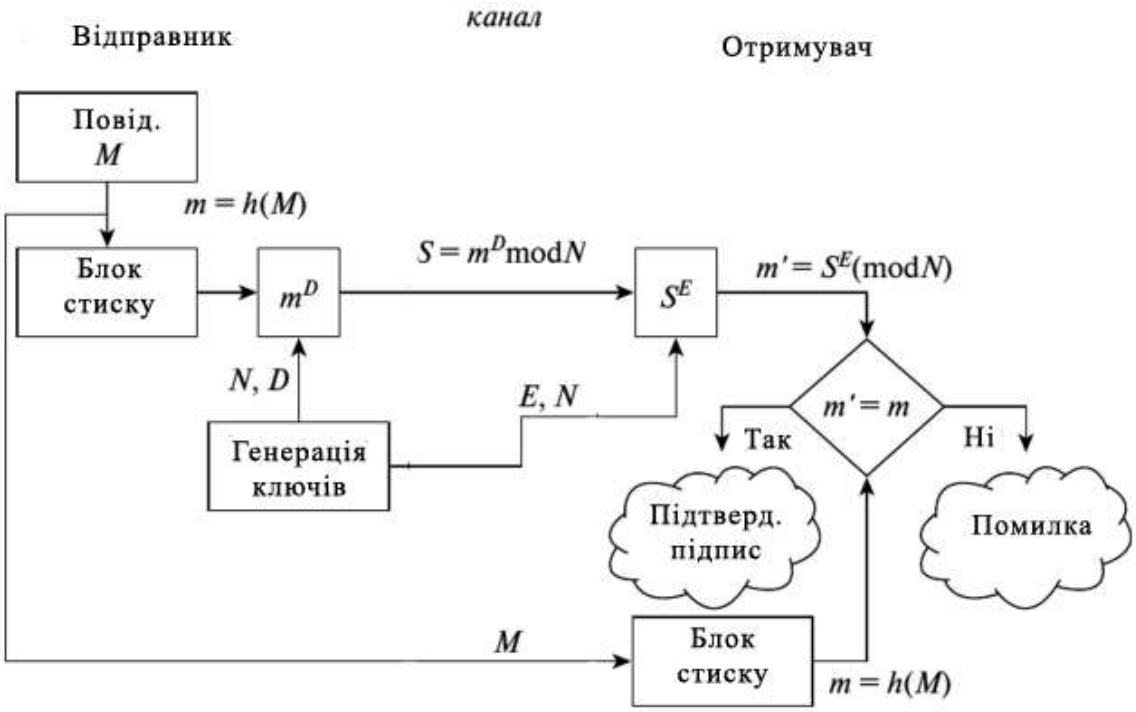


Рисунок 2.4 – Алгоритм роботи електронного підпису RSA

Потім отримувач подає цифровий підпис і ключ перевірки в алгоритм перевірки DSA . Таким чином, алгоритм перевірки надає деяке значення як вихідний файл у вигляді зашифрованого тексту.

- Таким чином, отримувач також запускає ту ж хеш-функцію для отриманих даних, щоб генерувати хеш-значення в цьому алгоритмі.
- Тепер щодо перевірки підпису, використаємо хеш-значення та вихід алгоритму перевірки, що порівнюються з кожною змінною. На основі результату порівняння отримувач вирішує, чи є цифровий підпис дійсним для цього чи недійсним.

Можна зробити висновок, що цифровий підпис генерується за допомогою `` приватного " ключа відправника, і ніхто інший не може мати цей ключ для захисту даних; відправник не може відмовитись від підписаних даних у майбутньому, для подальшого захисту цих даних за допомогою криптографії [11].

Це корисно не лише для електронної пошти, але й для інших електронних транзакцій та передач, таких як переказ коштів. Дотепер безпека алгоритму RSA перевірена, оскільки жодні відомі спроби його зламати досі не були успішними, здебільшого через труднощі факторизації великих чисел n = pq, де p і q - великі прості числа.

Стандарт цифрового підпису (DSS) - це тип алгоритму цифрового підпису, який був розроблений Агентством національної безпеки США (NSA) для створення цифрових підписів для автентифікації електронних систем. Цей алгоритм цифрового підпису, вироблений Національним інститутом стандартів і технологій (NIST) в 1994 році, став стандартним алгоритмом автентифікації електронних документів. Це зазначено у Федеральному стандарті обробки інформації (FIPS) [6].

Існує три алгоритми, які підходять для генерації цифрових підписів за стандартом DSS. Це алгоритм цифрового підпису (DSA, про який я

детальніше розповім пізніше), алгоритм RSA та алгоритм цифрового підпису Еліптичної кривої (ECDSA). Також у цьому стандарті є хеш-функція, яка буде використана в процесі генерації підписів. Він використовується для отримання скороченої версії даних, яка називається дайджестом повідомлення. Далі дайджест повідомлення вноситься в алгоритм цифрового підпису для формування повідомлення з цифровим підписом. Ця ж хеш-функція використовується і в процесі перевірки. Хеш-функція, яка використовується в стандарті DSS, зазначена в стандарті Secure Hash (SHS), який є специфікацією алгоритму Secure Hash (SHA). SHA базується на принципах, подібних до принципів, які використовував професор Рональд Л.Рівест з Массачусетського технологічного інституту при розробці алгоритму дайджесту повідомлень MD4, і тісно змодельований за цим алгоритмом. Коли вводиться повідомлення будь-якої довжини <264 біта, SHA видає 160-бітний вихід (дайджест повідомлення). Підписання дайджесту повідомлення, а не повідомлення, часто покращує ефективність процесу, оскільки дайджест повідомлення зазвичай має набагато менший розмір, ніж повідомлення [11].

Алгоритм DSS може бути реалізований в програмному забезпеченні програмного забезпечення, апаратному забезпеченні або будь-якій їх комбінації. NIST розробив програму перевірки для перевірки реалізацій на відповідність DSA. В даний час тести відповідності для ANSI X9.31 та ANSI X9.62 не розроблені. Ці тести будуть розроблені та надані в майбутньому. Інформацію про заплановану програму перевірки можна отримати в Національному інституті стандартів і технологій, Лабораторія інформаційних технологій, Attn: DSS Validation, 100 Bureau Drive Stop 8930, Gaithersburg, MD 20899-8930 [6].

2.5 Алгоритм Діффі-Хелмана

Алгоритм Діффі-Хеллмана використовує математику, тобто модульну арифметику і дискретний логарифм для створення загального ключа як для відправника, так і для одержувача за допомогою каналу зв'язку, де відправник та одержувач вибирають спільне просте число p і q як його примітивний корінь, де q <p. [4] Існує багато атак на алгоритм Діффі-Хеллмана, таких як відома атака відкритого тексту, атака "людина посередині", атака "Інсайдер" тощо. Відома атака відкритого тексту використовує звичайний текст і текст шифру із загальнодоступного каналу, щоб отримати доступ до секретний ключ. Під час атаки "Людина посередині" зловмисник зберігає відкриті ключі відправника та одержувача і надсилає їм підроблені відкриті ключі. Рисунок 2.5 відображає чітку картину алгоритму Діффі-Хеллмана. Алгоритм Діффі-Хеллмана дає спільно-секретний ключ, беручи примітивний корінь простого числа, знаходячи відкриті ключі та обмінюючись ними по загальнодоступному каналу використовуючи модульну арифметику.

1. Відправник та одержувач узгоджують просте число p, q, оскільки це примітивний корінь.

2. Відправник та одержувач вибирають свої так звані приватні ключі „k_a” та „k_b”, які відомі лише їм самим.

3. Відкритий ключ відправника

4. Відкритий ключ отримувача B = qb mod p.

5. Відправник та отримувач обмінюються своїм відкритим ключем. Тепер відправник має B, а одержувач A.

6. Відправник обчислює $y_A = g^{Ka} (\text{mod } N)$

7. Отримувач обчислює $Y_A = g^{Kb} (\text{mod } N)$

8. Отже, відправник і отримувач отримують 'K' як їх спільний секретний ключ. Ось як відбувається обмін ключами в алгоритмі Діффі-

Хеллмана. Параметром, який відомий зловмиснику, є p, q, A, B, оскільки ці параметри обмінюються на загальнодоступному каналі. Отже, щоб знати загально секретний ключ відправника та одержувача, зловмиснику доведеться обчислити значення a та b, яке відомо лише відправнику та одержувачу. Зловмиснику важко отримати секретний ключ, але це НЕ неможливо. [5]

Обмін ключами Діффі-Хелмана демонструє приклад того, як користувачі можуть безпечно обмінюватися криптографічними ключами через загальнодоступний канал.

Раніше безпечний зашифрований зв’язок вимагав, щоб особи спочатку обмінювалися ключами безпечними засобами, такими як паперові списки ключів, які перевозив надійний кур’єр. Метод обміну ключами Діффі – Хеллмана дозволяє двом сторонам, які не мають попередніх знань одна про одну, спільно встановити спільний секретний ключ по незахищеному каналу [14].

PreVeil використовує обмін ключами Діффі-Хелмана, щоб увімкнути Web PreVeil. Web PreVeil - це наскрізна зашифрована поштова служба електронної пошти, яка дозволяє користувачам легко отримати доступ до свого захищеного облікового запису електронної пошти в Інтернеті без будь-якого завантаження програмного забезпечення або будь-яких паролів, які потрібно запам'ятати.

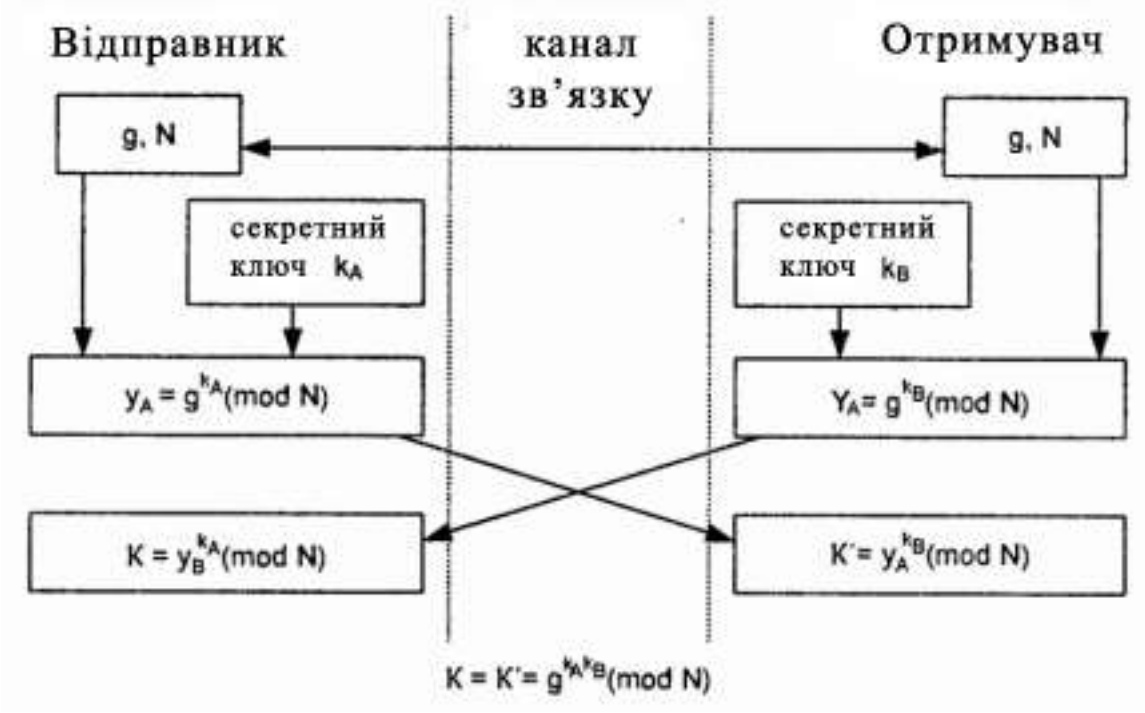


Рисунок 2.5 – Алгоритм шифрування Діффі-Хелмана

2.6 Еліптичні криві

Криптографічна система на основі еліптичних кривих працює, концентруючись на певних парах відкритих та приватних ключів для шифрування та дешифрування веб-трафіку [1].

Криптографія ECC - це метод, заснований на ключах, який використовує техніку шифрування відкритого ключа для шифрування даних на основі теорії еліптичної кривої. За допомогою теорії еліптичних кривих створюються швидші, менші та ефективніші криптографічні ключі.

За допомогою шифрування еліптичної кривої, складні та математично надійні ключі генеруються за допомогою особливостей рівняння еліптичної кривої на зміну традиційній техніці як добутку величезних простих чисел. Криптосистемна технологія еліптичної кривої може працювати одночасно з багатьма методами шифрування з відкритим ключем, включаючи RSA та Діффі-Хеллмана. Різні дослідження показують, що системи ECC можуть досягти подібного рівня безпеки за

допомогою 164-бітового ключа, коли інші методи хочуть отримати 1024-бітний ключ. Це головним чином тому, що метод еліптичної кривої підтримує створення рівноцінної безпеки з меншою обчислювальною потужністю та з зменшеним використанням заряду акумулятора, завдяки чому він широко використовується для різних мобільних додатків.

Криптосистема еліптичної кривої стала ключовою точкою в галузі досліджень безпеки з високоміцним захистом, найшвидшою швидкістю обробки та найнижчою вартістю [4]. На даний момент довжина ключа алгоритму шифрування еліптичної кривої коротша, як правило, приблизно від 30 до 60 , який визначається швидкістю виконання програмної реалізації. І довжина ключа є відносно великою в апаратній реалізації еліптичної кривої. Для алгоритму шифрування знадобиться багато апаратних ресурсів.

Еліптична крива - це функція алгоритму для сучасного використання ECC, яка є плоскою та асиметричною кривою (рис 2.6), яка перетинає кінцеве поле, що містить точки, що підтримують таке рівняння еліптичної кривої:

$y^2=x^3$ ax b

Криптографічне шифрування еліптичної кривої є одним із найбільш загальноприйнятих методів застосування цифрових підписів у різних криптовалютах. Популярні криптовалюти, такі як біткойн та ефір, використовують алгоритм цифрового підпису Еліптичної кривої (ключ ECDSA), особливо для підписання транзакцій через рівні безпеки, запропоновані ECC.

Для цифрових підписів ECC застосовується через еліптичну криву DSA (ключ ECDSA) та при обміні ключами через еліптичну криву Діффі-Хелмана (ECDH). Ці алгоритми використовуються в різних частинах стандарту SSL, також займаються підписання сертифікатів SSL за допомогою ECDSA замість RSA.

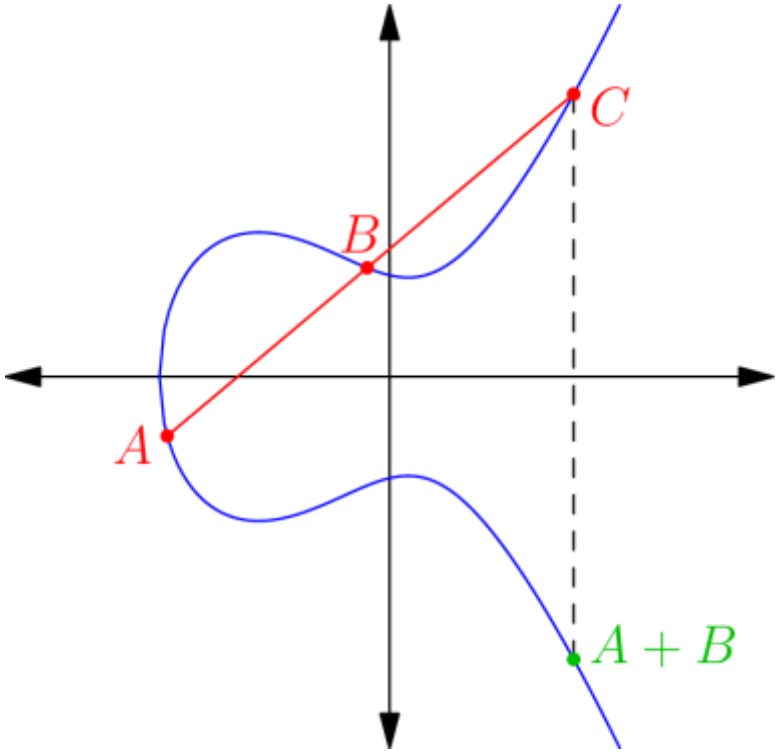


Рисунок 2.7 – Геометрична візуалізація додавання на еліптичній кривій

Окрім криптовалют, ECC також виконує функції стандартного режиму шифрування, який широко використовується різними веб-додатками і, як очікується, зростатиме в майбутньому завдяки меншій довжині ключа, безпеці та ефективності.

Криптографія еліптичної кривої також використовується в ряді функцій, таких як:

- Захист конфіденційних даних та внутрішньої взаємодії деяких урядових баз,
- Підтримка та гарантування анонімності в проекті TOR,
- Метод або основа, на яких підтверджується право власності на біткойнах,
- Надання підписів у службі iMessage від Apple,

- Кодування даних DNS за допомогою DNSCurve.
- Криптографія кривої еліптичної кривої також є найбільш улюбленим процесом автентифікації через SSL / TLS для безпечного та безпечного перегляду веб-сторінок.

Еліптичне шифрування за допомогою криптографії з відкритим ключем на основі алгоритмів порівняно легко обробити в одному напрямку і складно працювати в зворотному напрямку. Для кращого розуміння ключі ECC ефективніші, ніж RSA, оскільки RSA залежить від теорії, згідно з якою множення простих чисел, щоб отримати більше число, є простим, а множення великих чисел для повернення до вихідних простих чисел досить важке [1].

Звичайний розмір ключа ECC у 256 біт дорівнює 3072 бітовому ключу RSA, що в 10 000 разів ефективніше, ніж у 2048 бітовому ключі RSA. Отже, щоб залишатися в безпеці та випереджати обчислювальну потужність хакера, ключі RSA повинні бути довгими та вимагати ключів довжиною 2048 біт або довше, що робить процес повільнішим.

Оскільки ECC використовує простіші та менші ключі, розмір є однією з головних переваг криптографії еліптичної кривої. Завдяки можливості споживати менше енергії для перетворення та перетворення більшої потужності на невеликі мобільні пристрої, це робить шифрування факторингу RSA слабшим.

На відміну від інших методів шифрування, за допомогою ECC подібний рівень безпеки та високий рівень безпеки можна досягти за допомогою менших і швидших ключів з меншою обчислювальною потужністю.

Еліптична крива також забезпечує переваги безпеки та виступає ідеальною альтернативою RSA та DSA у ситуаціях, коли в RSA виявляються будь-які загрози або слабкі місця, особливо у випадках, коли засоби захисту від погроз потребують значного збільшення розміру ключа.

Використання менших ключів у ECC робить це швидшим, оскільки менше даних передається від сервера до клієнта під час процесу координації SSL. Крім того, ECC спричиняє меншу обчислювальну потужність та пам'ять, як наслідок чого покращується та пришвидшується час відгуку на веб-серверах під час використання.

Ідеальна пряма секретність (PFS) також є важливою перевагою ECC, особливо для веб-серверів, які бажають ефемерного ECDH (ECDHE) із використанням шифрувальних наборів, оскільки вони користуються перевагами як ECC, так і PFS.

На рисунку 2.8 зображена блок-схема роботи алгоритму на основі еліптичних кривих.

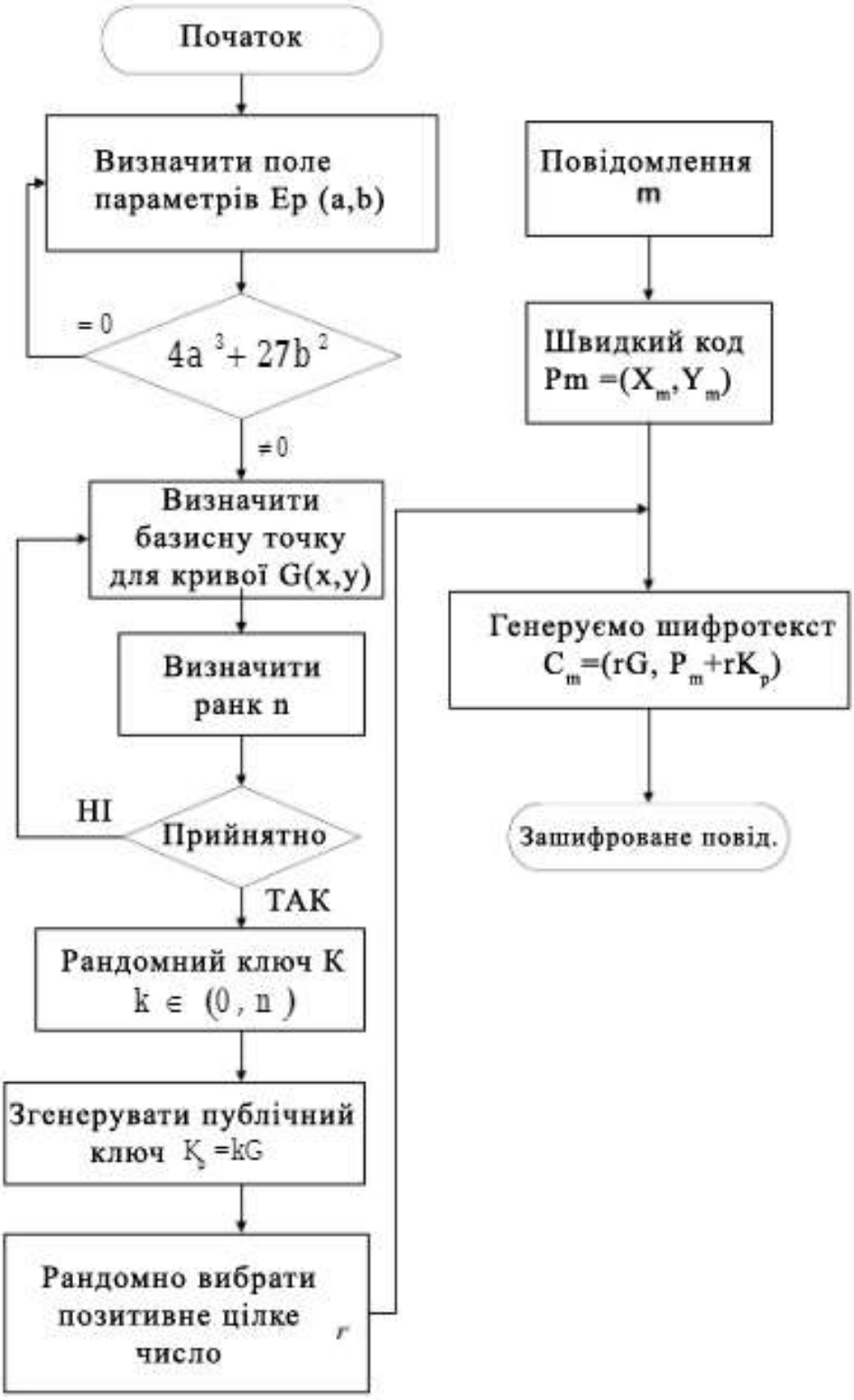


Рисунок 2.8 – Блок-схема роботи алгоритму на основі еліптичних кривих

2.7 Використання даних криптосистем з відкритим ключем

У криптографії з відкритим ключем кожен користувач повинен створити пару ключів, серед яких один зберігається в секреті (приватний ключ), а інший роблять відкритим, і тому цей алгоритм так і назвали. Тепер рішення щодо використання приватного ключа відправника або публічного ключа одержувача для шифрування оригінального повідомлення повністю залежить від конкретного додатка і завдання.

Можна класифікувати конкретні завдання для криптосистеми з відкритим ключем:

a. Для шифрування / дешифрування

Якщо метою програми є шифрування та розшифровка повідомлення, тоді відправник повинен зашифрувати повідомлення, використовуючи загальнодоступні одержувачі, і одержувач може розшифрувати повідомлення за допомогою власного приватного ключа.

b. Для цифрового підпису

Якщо метою програми є аутентифікація користувача, тоді повідомлення підписується або зашифровується за допомогою приватного ключа відправників. Оскільки лише відправник може мати свій приватний ключ, він гарантує всім сторонам, що повідомлення надіслано конкретною особою.

c. Для обміну ключами

Дві сторони, що спілкуються, обмінюються секретним ключем (можливо, приватним ключем) для симетричного шифрування для забезпечення конкретної транзакції. Цей секретний ключ дійсний протягом короткого періоду [4].

Деякі алгоритми реалізують усі три програми, а деякі реалізують один або два серед них. Нижче наведено рисунок 2.9, що демонструє більш детальніше який алгоритм для якого завдання найбільше підходить.

Алгоритм	Шифрування/ розшифровка	Цифровий підпис	Обмін ключами
RSA	так	так	так
Еліптичні криві	так	так	так
Діффі-Хелман	ні	ні	так
DSS	ні	так	ні

Рисунок 2.9 – Класифікація завдань для алгоритмів з відкритим ключем

2.8 Висновки

Отже, асиметричних алгоритмів існує не так багато, але в кожного з відомих є свої переваги і недоліки. Кожен з них можна розподілити на сфери використання, тобто для яких потреб найкраще взяти той чи інший алгоритм. DSS – для цифрового підпису, Діффі-Хелман найкраще використовувати для обміну секретними і публічними ключами. РСА та Еліптичні криві є більш універсальніший. Тому, проаналізувавши всі складові системи я хочу перейти до рішення власного завдання.

3 ПРОЕКТУВАННЯ КРИПТОГРАФІЧНОЇ СИСТЕМИ

3.1 Проектування алгоритму власного модуля

Для того, щоб реалізувати власний програмний модуль спроектуємо алгоритм її роботи.

Цю реалізацію я зобразив на рисунку 3.1:

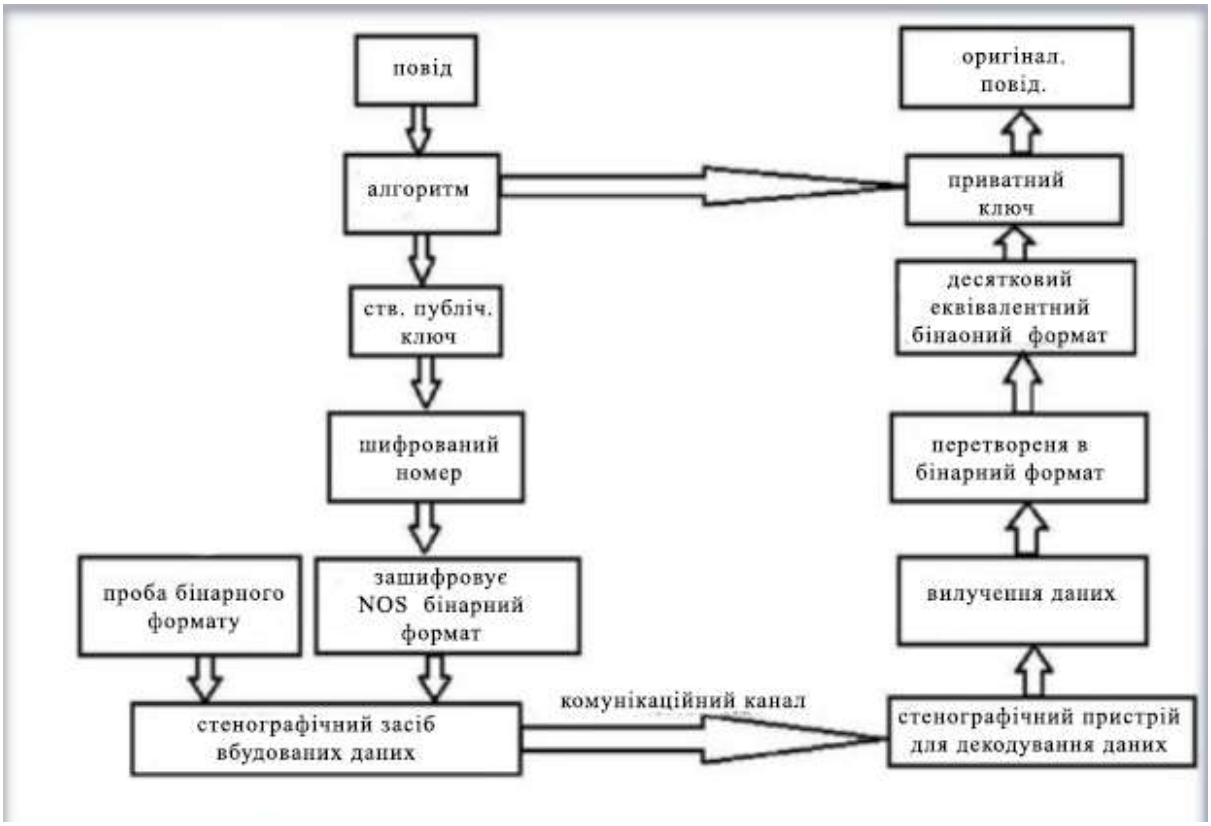


Рисунок 3.1 – Спроектований алгоритм роботи власного модуля

1. По-перше ми генеруємо два простих числа p і q з допомогою функції генерації простих чисел.

- Далі з допомогою наслідків функцій Ейлера і Кармайкла ми вираховуємо нашу лямбду. Функція Ейлера, це функція, яка дорівнює кількості натуральних чисел, менших m і взаємно простих з m. Передбачається, що число 1 взаємно просто з усіма

натуральними числами (і з одиниці). Позначається функція Ейлера грецької буквою φ. В математиці гіпотеза тотент-функції Кармайкла стосується кратності значень сумарної функції Ейлера φ (n), який підраховує кількість цілих чисел менше і взаємно простих з n. Він стверджує, що для кожного n існує хоча б одне ціле число m ≠ n таке, що φ (m) = φ (n). Роберт Кармайкл вперше висловив цю гіпотезу в 1907 році, але як теорему, а не як гіпотезу. Однак його доказ було помилковим, і в 1922 році він відмовився від свого твердження і сформулював гіпотезу як відкриту проблему .

- φ(pq):=(p − 1) * (q − 1) - функція Ейлера (Euler totient function)
- λ(pq):=lcm(p − 1, q − 1) - функція Кармайкла (Carmichael function)
- φ(N) = λ(N) * gcd(p − 1, q − 1)
- найменше спільне кратне (least common multiple)
- lcm(a, b) = |a * b| // gcd(a, b)
- попросту λ меньше, но з тими ж властивостями

2. Після цього створюємо публічний ключ
3. Далі вираховуємо експоненту з допомогою числа Ферма, що має в нас стале значення F(4) = 2 ** 16, тобто наша експонента 65537.
4. Далі з допомогою нашої функції mod_mul_inverse, лямбди та експоненти ми визначаємо наш приватний ключ. Який буде знаходитись в пункті призначення нашого повідомлення
5. Робимо перебір дільників з 3 до квадратного кореню
6. Визначаєм зашифрований номер нашого ключа
7. Зашифровуємо наше повідомлення в NOS бінарний формат
8. Створюємо пробу нашого бінарного формату повідомлення
9. Далі додаємо наше повідомлення перетворене в бінарний формат і наш алгоритм в наш стенографічний засіб вбудованих даних

10. Далі передаємо нашу інформацію з нашого стенографічного засобі обробки даних в стенографічний пристрій декодування даних пристрою отримувача нашого повідомлення.
11. Після цього наш алгоритм вилучає зайві дані інформації
12. Вся ця інформація повинна бути в бітовому форматі, а якщо ні то перетворюємо їх
13. Далі для декодування ми перетворюємо наші дані в десятковий еквівалентний бінарний формат
14. В завершенні з допомогою алгоритму, функції та приватного ключа отримувач повідомлення розшифровує дані в оригінальне повідомлення.

В завершенні хотілося би згадати наші використовувані функції які будуть потрібні для реалізації.

1. Бінарний пошук для знаходження квадратного кореня з цілого додатного числа
2. Модульне піднесення до степеню шляхом повторного піднесення в квадрат для швидкого вираховування великих додатніх ступенів числа
3. Алгоритм Евкліда для обчислення найбільшого спільного дільника (НСД) двох чисел
4. Розширений алгоритм Евкліда, який обчислює коефіцієнти ідентичності Безу для пошуку експоненти приватного ключа (модульна мультиплікативна обернена експонента відкритого ключа). Рівнянням Безу називається представлення найбільшого спільного дільника d двох натуральних чисел A і B у вигляді лінійної комбінації Ax + By = d, де x, y - цілі числа, називані коефіцієнтами Безу. Зазвичай коефіцієнти Безу обчислюються з використанням розширеної версії класичного алгоритму Евкліда, що я і використав

5. Перебір дільників (найпростіший для розуміння цілочисельних алгоритмів розкладання на множники), щоб визначити, чи є число простим

6. Тест на «фермальність» для перевірки його експоненціально швидшим способом. Вважається ніби швидшим ніж перебір дільників

Я не заглиблююся в теорію (поки не ясно, скільки тренінгів слід проводити читачам), але я вважаю, що читання цієї короткої ілюстрації полегшить кожному зрозуміти формулу.

3.2 Проведення розрахунків

І так. Припустимо, я хочу отримати від вас деякі дані. Ми не хочемо, щоб хтось крім нас знав цю інформацію. І ми не впевнені в надійності каналу даних.

Перший крок. Підготовка ключів

Мені потрібно зробити попередні кроки: створити відкриті та закриті ключі. Я вибираю два простих числа. Нехай p = 3 і q = 7. Ми обчислюємо модуль-добуток наших pі та q: n = p × q = 3 × 7 = 21.

Обчисліть функцію Ейлера: φ = (p-1) × (q-1) = 2 × 6 = 12.

Виберемо число e, яке відповідає наступним умовам: (i) має бути простим, Варіанти 5, 7, 11 зарезервовані. Виберіть e = 5. Це так званий відкритий індекс.

Зараз ця пара чисел {e, n} - мій відкритий ключ. Я надішлю його вам для шифрування вашого повідомлення. Але це не все я. Я повинен отримати приватний ключ.

Мені потрібно обчислити число d, яке є зворотним значенням епо модуля φ. Іншими словами, залишок модульного поділу φ добутку d × e

повинен дорівнювати 1. Ми використовуємо символи, що використовуються в багатьох мовах програмування, щоб записати:

(d × e)% φ = 1. Або (d × 5)% 12 = 1.

D може дорівнювати 5 ((5 × 5)% 12 = 25% 12 = 1), але, щоб не плутати з e у наступній історії, ми вважаємо, що воно дорівнює 17. Ви можете самі перевірити (17 × 5)% 12 насправді дорівнює 1 (17 × 5-12 × 7 = 1). Отже d = 17. Ця пара {d, n} - це секретний ключ, який я зберігаю який я не можу нікому повідомити. Лише власник приватного ключа може розшифрувати вміст, зашифрований відкритим ключем.

Другий крок. шифрування

Тепер ваша черга зашифрувати своє повідомлення. Припустимо, ваше повідомлення - це число 19. Позначимо його P = 19. Крім того, у вас вже є мій відкритий ключ: {e, n} = {5, 21}. Зашифровано згідно з таким алгоритмом:

Зменшіть свою інформацію до рівня модуля n. Тобто, обчислити 19 до 5-го ступеня (2476099) і розділити залишок від ділення на 21. Результат 10 - це наші закодовані дані.

Третій крок розшифровка

Я отримав ваші дані (E = 10), і у мене є приватний ключ {d, n} = {17, 21}. Я обчислюю залишок від ділення на 21 і отримую 19 - ваше повідомлення.

Ніхто, крім мене (навіть ви!), Не може розшифрувати повідомлення (E = 10), оскільки ніхто не має приватного ключа.

Що є запорукою надійності шифрування

Той факт, що третій стороні (намагаючись зламати пароль) важко вивести приватний ключ із відкритого ключа, забезпечує надійність шифрування. Обидві клавіші обчислюються з пари простих чисел (pі та q). Іншими словами, клавіші взаємопов’язані. Але встановити такий зв’язок дуже важко. Основна проблема полягає в розкладанні модуля n на прості

множники pi та q. Якщо число є добутком двох дуже великих простих чисел, то розкладання дуже важке.

Я спробую це продемонструвати на прикладі. Давайте розкладемо число 360 на множники:

Насправді, викладений спосіб шифрування дуже слабкий і ніколи не використовується. Причина проста - шифрування по буквах. Одна і та ж буква шифруватиметься одним і тим же числом. Якщо зловмисник перехопить досить велике повідомлення, він зможе здогадатися про його вмісті. Спершу він зверне увагу на часті коди прогалин і розділить шифровку на слова. Потім він помітить однобуквені слова і здогадається, як кодуються букви «а», «і», «о», «в», «до» ... Шляхом недовгого перебору, він визначить додаткові літери по коротким словами, типу «але», «Не »,« по ». І по більш довгих слів без праці відновить всі залишилися літери.

Таким чином, зловмиснику не доведеться відгадувати ваші секретні ключі. Він зламає ваше повідомлення, не знаючи їх.

Щоб цього не відбувалося, використовуються спеціальні додаткові алгоритми, суть яких в тому, що кожна попередня частина повідомлення починає впливати на наступну.

Спрощено, це виглядає так. Перед шифруванням, ми застосовуємо до повідомлення правило: b := (b + a) % n. Де a- попередня частина повідомлення, а b- наступна. Тобто наше повідомлення (11, 17, 15, 19) змінюється. 11 залишається без змін. 17 перетворюється в (11 + 17) % 323 = 28. 15 стає (15 + 28) % 323 = 43. А 19 перетворюється в 62.

Послідовність (11, 28, 43, 62) виходить «заплутаною». Всі букви в ній як би перемішані, в тому сенсі, що на кожен код впливає не одна буква, а всі попередні.

Той, хто отримає ваше повідомлення, повинен буде виконати зворотну операцію, зі знаком «мінус»: b := (b - a) % n. І тільки тоді він отримає коди букв.

На практиці, в вихідне повідомлення спеціально додаються випадкові і безглузді літери в початок. Щоб навіть по першому коду було неможливо нічого зрозуміти. Одержувач просто відкидає початок повідомлення.

Тобто ми можемо додати випадкове число в початок і отримати (299, 11, 17, 15, 19). Після перемішування вийде: 299, 310, 4, 19, 38. Після шифрування вже неможливо буде здогадатися де була якась буква.

3.3 Висновки

У реальному житті все працює трохи складніше. Блоки, на які б'ється повідомлення довше однієї літери. Тому, спершу застосовуються алгоритми вирівнювання, потім алгоритми розбиття на блоки з змішування, і тільки потім застосовується саме шифрування.

Одержувач робить все в зворотному порядку: розшифровує, «розплутує» блоки і відкидає непотрібну інформацію, додану просто для вирівнювання (щоб повідомлення можна було розбити на ціле число блоків).

Далі перейдемо до програмної реалізації.

4 РЕАЛІЗАЦІЯ ЗАВДАННЯ І ЗАПРОПОНОВАНЕ РІШЕННЯ

4.1 Реалізація програмного модуля

Для рішення задачі передачі повідомлення через мережу використовуючи алгоритм криптозахисту з відкритим ключем я проаналізував наявні найпопулярніші і найефективніші алгоритми та побудував програмний модуль, який реалізовує асиметричне шифрування з допомогою відкритого ключа в Python. Цей модуль працює за принципом RSA але є простішим і зрозумілішим.

Вбудовані математичні функції використовуватись не будуть окрім основних операторів:

- Floor Division
- Modulo Division
- Exponent
- Bitwise Operators

В нашому модулі ми реалізуємо такі функції:

1. Бінарний пошук для знаходження квадратного кореня з цілого додатного числа:

```
def bin_sqrt(x):
    start = 1
    end = x
    while start <= end:
        mid = (start + end) // 2
        if mid * mid == x:
            return mid
        elif mid * mid < x:
            start = mid + 1
            r = mid
```

```
            else:
                end = mid - 1
        return r
```

2. Модульне піднесення до степеню шляхом повторного піднесення в квадрат для швидкого вираховування великих додатніх ступенів числа:

```
"""
Молуляр мультиплікативно-зворотній
"""
    def mul_inverse(a, b):
        g, x, _ = extended_gcd(a, b)
        return x % b
    #перебів дільників
    def trial_division(x):
        l = range(3, bin_sqrt(x), 2)
        i = 1
        if not x & 1: #x - парне (x % 2)
            return False
        for y in l: # x з кроком 2
            show_stat(f'CHECK {x}', i, 10000, len(l))
            if not x % y:
                return False
            i += 1
        return True
```

3. Алгоритм Евкліда для обчислення найбільшого спільного дільника (НСД) двох чисел. Виглядає він так:

```
        """
        Найбільший спільний дільник для алгоритму Евкліда (x > y)
        ((12 * math.log(2)) / math.pow(math.pi, 2)) * math.log(x)
```

```
    """
    def gcd(x, y):
        while(y):
            x, y = y, x % y
        return x
```

4. Розширений алгоритм Евкліда, який обчислює коефіцієнти ідентичності Безу для пошуку експоненти приватного ключа (модульна мультиплікативна обернена експонента відкритого ключа). Рівнянням Безу називається представлення найбільшого спільного дільника d двох натуральних чисел A і B у вигляді лінійної комбінації Ax + By = d, де x, y - цілі числа, називані коефіцієнтами Безу. Зазвичай коефіцієнти Безу обчислюються з використанням розширеної версії класичного алгоритму Евкліда, що я і використав:

```
    """
    Розширений алгоритм Евкліда
    return (g, x, y) such that a*x + b*y = g = gcd(a, b)
    """
    def extended_gcd(a, b):
        x0, x1, y0, y1 = 0, 1, 1, 0
        while a:
            q, b, a = b // a, a, b % a
            y0, y1 = y1, y0 - q * y1
            x0, x1 = x1, x0 - q * x1
        return b, x0, y0


    """
    modular multiplicative inverse
    модульна мультиплікативна інверсія
    return x such that (x * a) % b == 1
```

```
    """
    def mod_mul_inverse(a, b):
        g, x, _ = extended_gcd(a, b)
        return x % b
```

5. Перебір дільників (найпростіший для розуміння цілочисельних алгоритмів розкладання на множники), щоб визначити, чи є число простим:

```
def trial_division(x):
        l = range(3, bin_sqrt(x), 2)
        i = 1
        if not x & 1: #x - чётное (x % 2)
            return False
        for y in l: #починаючи з 3 до квадратного кореня з x крок 2
            show_stat(f'CHECK {x}', i, 100000, len(l))
            if not x % y:
                return False
            i += 1
        return True
```

6. Тест на фермальність для перевірки його експоненціально швидшим способом:

```
"""
тест Ферма
експоненціально швидший, ніж перебір дільників
"""
def fermat_test(n, k):
    for _ in range(k):
        a = random.randint(2, n - 1)
        if mod_exp(a, n - 1, n) != 1: #піднесення в степінь по модулю
```

return False

return True

Великі повідомлення розбиваються на блоки.

Розібравши на окремі функції програму тепер можна її знову зібрати.

Результат представлений на рисунку 4.1 і 4.2

```
Генерація простих чисел
Перебір всіх дільників пройдено 48111515740751: 98%
Підтверджено за 2.1728415489196777 секунд
Перебір всіх дільників пройдено 225497189721833: 99%
Підтверджено за 6.0918474197387695 секунд
Публічний: 10849011592797082990632516583
Лямбда: 774929399485486384423361000
Приватний: 32681193329538059690064047З
--------------
ПОВІДОМ
--------------
16676813828883275323789044687178653489834020301134368460762837357713877932807473933462291382871335558686
--------------
КІЛЬКІСТЬ ЧАСТИН: 153717356518960758995255616660428006689021580317286449679186113840320385926558256978822
ЗАЛИШОК: 89165616397303608403242243l9
ЧАСТИНА: 10849011592797082990632516582
Закодований шлях: 10849011592797082990632516582
Закодований залишок: 655404584645563344163375875З
Розшифрований шлях: 10849011592797082990632516582
Розшифрований залишок: 89165616397303608403242243l9
--------------
РОЗШИФРОВКА
--------------
16676813828883275323789044687178653489834020301134368460762837357713877932807473933462291382871335558686
--------------
```

Рисунок 4.1– Результат виконання модулю для простого повідомлення

Крім того, для додаткової перевірки ми взяли повідомлення більшого розміру для перевірки розбиття.

```
main()

Генерація простих чисел
CHECK 243791939704921: 99%
TRUE in 4.5113205909729 seconds
CHECK 11161595065193: 95%
TRUE in 0.9338772296905518 seconds
Публічний: 272110691114427563417791 4753
Лямбда: 340138363893002585080393080
Приватний: 272662909279991208169179713
--------------
Сьогодні криптографія відіграє вирішальну роль в мережевій безпеці
--------------
29801463751092859740882285986146669300790016047942579676463999669425354783212181362165814392501770302033603
--------------
КІЛЬКІСТЬ ЧАСТИН: 109519635663858553376216645557225888210575792259942357540502500377110656118418400787171906
ЗАЛИШОК: 26833528295210075903579233 88
ЧАСТИНА: 272110691114427563417791 4752
Закодований шлях: 272110691114427563417791 4752
Закодований залишок: 62982207329281647187485 9208
Розшифрований шлях: 272110691114427563417791 4752
Розшифрований залишок: 26833528295210075903579233 88
--------------
РОЗШИФРОВКА
--------------
29801463751092859740882285986146669300790016047942579676463999669425354783212181362165814392501770302033603
--------------
```

Рисунок 4.2 – Результат виконання програмного модулю для шифрування і розшифровки повідомлення «Сьогодні криптографія відіграє вирішальну роль в мережевій безпеці»

Лістинг програми представлений в додатку А

4.2 Додаткові рішення

Для кожної загрози існує своя протидія або контрзахід. Єдиним ефективним способом попередження попередньо визначених нами загроз є використання спеціальної електронної системи управління ключами, в ідеалі зрілого, перевіреного рішення від надійного постачальника з хорошими рекомендаціями клієнтів. Будь-яка така система управління ключами повинна використовувати апаратний модуль захисту (HSM) для генерації та захисту ключів та для забезпечення безпеки всієї системи.

HSM або апаратний модуль безпеки - це захищений криптографічний процесор, який зосереджений на наданні ключів шифрування, а також використовує ці ключі для забезпечення прискорених операцій шифрування. Як якір довіри, цей модуль забезпечує захист ідентифікаційних даних, програм та транзакцій та забезпечує чітке

шифрування, дешифрування та автентифікацію для різних програм. Модуль апаратного захисту включає такі функції безпеки, як фізичний опір несанкціонованому захисту та надійну автентифікацію. Хоча модуль фізично ізольований, як і смарт-карти та зворотні стрічки, він забезпечує більш високий рівень безпеки, оскільки не має операційної системи і, отже, майже не захищений від мережевих атак.

Модульні апаратні системи безпеки мають різні функції та фактори формування, і їх нелегко пошкодити та вийти з ладу. Це пов’язано з тим, що вони не мають операційної системи і зовні під’єднані до пристроїв, які вони обслуговують. Приклади модульних апаратних систем безпеки включають фізично захищені пристрої локальної мережі, смарт-карти та плагіни PCI. Апаратний модуль безпеки використовує двофакторну автентифікацію для забезпечення захисту від внутрішніх і зовнішніх зловмисників.

Якщо система добре спроектована, вона матиме такі переваги (рис 4.3):

- Повне управління життєвим циклом ключів
- Генерація надійних ключів із використанням сертифікованого FIPS стандарту RNG та апаратної ентропії
- Захист ключів за допомогою захисту від втручання HSM
- Строгий контроль на основі політики, щоб запобігти неправильному / повторному використанню ключів
- Автоматичне обертання(апдейту) ключів
- Автоматичний безпечний розподіл ключів

Рисунок 4.3 – Контрміри для захисту ключів алгоритму апаратного модулю HSM

- Можливість безпечного імпорту / експорту ключів у компонентах або під транспортним ключем
- Можливість надійно знищити ключі в кінці їх життєвого циклу
- Надійна автентифікація користувача, розподіл обов'язків та подвійний контроль над критичними операціями
- Інтуїтивно зрозумілий користувальницький інтерфейс та безпечне управління робочим процесом для мінімізації ризику людських помилок
- Підтримка високої доступності та безперервності бізнесу
- Журнал аудиту, журнал використання та історія ключових явищ для демонстрації відповідності
- Можливість швидко реагувати на виявлений компроміс

4.3 Висновки

Наша система захисту працює за принципами шифрування з відкритим ключем, тому така система є досить ефективною і може допомогти зашифрувати повідомлення і безпечно розшифрувати його отримувачу.

Така система не тільки допоможе захистити ваші ключі, але також підвищить ефективність, зменшить залежність від висококваліфікованого персоналу та спростить досягнення, підтримку та демонстрацію відповідності безлічі стандартів та норм, таких як GDPR, PCI-DSS, HIPAA, SOX та ISO 27001

ВИСНОВКИ

В ході розробки даної криптосистеми захисту з відкритим ключем, було досліджено область різноманітних криптосистем, проаналізовано теоретичну інформацію пов’язану з тематикою роботи. Перед початком розробки даної кваліфікаційної роботи я проаналізував його актуальність і далі продовжив над ним роботу. Дана криптографічна система, на відміну від системи з симетричним ключем є набагато надійнішою. Існує досить багато переваг, які допомагають нівелювати проблеми які стосуються передачі повідомлень через мережу.

На мою думку, існує нагальна потреба в сучасній криптографії для забезпечення захисту та цифрових ключів для забезпечення того, щоб інформація, яка передається у віртуальному просторі, залишалася недоторканою та захищеною. Сьогодні, в епоху комп’ютеризації, ми стикаємося із зростаючим ризиком компрометації нашої інтелектуальної власності та жертвами шахрайства. В представлені вже відомі рішення криптосистем з відкритим ключем і крім того була розроблений спрощений програмний модуль який виконує функції подібних систем. Крім того було додане апаратне рішення, що допоможе зменшити загрози для даного алгоритму криптозахисту.

На мою думку, поставлена задача була мною виконана у повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Application of Elliptic Curve Cryptography in ZigBee Wireless Sensor Network [Електронний ресурс] : researchgate. – Режим доступу : https://www.researchgate.net/publication/290575270_Application_of_Elliptic_Curve_Cryptography_in_ZigBee_Wireless_Sensor_Network/ – Назва з екрана.(електронне джерело)(дата звернення 11.04.2021)

2. Chapter 7: The role of cryptography [Електронний ресурс] : infosec. – Режим доступу : https://resources.infosecinstitute.com/topic/role-of-cryptography/ – Назва з екрана.(електронне джерело)(дата звернення 02.04.2021)

3. Cryptography and Network Security [Електронний ресурс] : ecpi.university. – Режим доступу : https://www.ecpi.edu/blog/crypotgraphy-and-network-security – Назва з екрана.(електронне джерело)(дата звернення 12.04.2021)

4. Cryptographic Key Management [Електронний ресурс] : cryptomathic. – Режим доступу : https://www.cryptomathic.com/news-events/blog/cryptographic-key-management-the-risks-and-mitigations – Назва з екрана.(електронне джерело)(дата звернення 02.04.2021)

5. Digital Signature Algorithm [Електронний ресурс] : includehelp.com. – Режим доступу : https://www.includehelp.com/cryptography/digital-signature-algorithm-dsa.aspx/ – Назва з екрана.(електронне джерело)(дата звернення 08.04.2021)

6. Digital Signature Standard (DSS) [Електронний ресурс] : wikisource.org. – Режим доступу : https://en.wikisource.org/wiki/Digital_Signature_Standard_(DSS) – Назва з екрана.(електронне джерело)(дата звернення 11.04.2021)

7. Disadvantages of SSL [Електронний ресурс] : chron. – Режим доступу : https://smallbusiness.chron.com/disadvantages-ssl-66719.html – Назва з екрана.(електронне джерело)(дата звернення 05.04.2021)
8. One Time Pad Cipher [Електронний ресурс] : tutorialspoint. – Режим доступу : https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_one_time_pad_cipher.htm/ – Назва з екрана.(електронне джерело)(дата звернення 13.04.2021)
9. Public Key Cryptography [Електронний ресурс] : binaryterms. – Режим доступу : https://binaryterms.com/public-key-cryptography.html – Назва з екрана.(електронне джерело)(дата звернення 02.04.2021)
10. Stallings Cryptography and Network Security [Електронний ресурс] : inf.ufsc.br. – Режим доступу : http://www.inf.ufsc.br/~bosco.sobral/ensino/ine5680/material-cripto-seg/2014-1/Stallings/Stallings_Cryptography_and_Network_Security – Назва з екрана.(електронне джерело)(дата звернення 02.04.2021)
11. What is Digital Signature Algorithm and Digital Signature Standard [Електронний ресурс] : wondershare. – Режим доступу : https://signx.wondershare.com/knowledge/digital-signature-algorithm.html/ – Назва з екрана.(електронне джерело)(дата звернення 09.04.2021)
12. Криптографія: навіщо вона нам потрібна? [Електронний ресурс] : ElectronicDesign. – Режим доступу : https://www.electronicdesign.com/technologies/embedded-revolution/article/21127827/maxim-integrated-cryptography-why-do-we-need-it – Назва з екрана.(електронне джерело)(дата звернення 12.04.2021)
13.Криптография: история шифровального дела [Електронний ресурс] : rostec. – Режим доступу : https://rostec.ru/news/kriptografiya-istoriya-

shifrovalnogo-dela/ – Назва з екрана.(електронне джерело)(дата звернення 14.04.2021)

14.Покращений алгоритм Diffie-Hellman для надійного обміну ключами [Електронний ресурс] : researchgate. – Режим доступу : https://www.researchgate.net/publication/321482944_Enhanced_diffie-hellman_algorithm_for_reliable_key_exchange/ – Назва з екрана.(електронне джерело)(дата звернення 12.04.2021)

15. Стандарт цифрового підпису (DSS) [Електронний ресурс] : ecpi.university. – Режим доступу : https://searchsecurity.techtarget.com/definition/Digital-Signature-Standard – Назва з екрана.(електронне джерело)(дата звернення 11.04.2021)

ДОДАТОК А

(Обов’язковий)

Програмна реалізація

```
def show_stat(m, i, s = 1, h = 1):
    if i % s == 0:
        sys.stdout.write('\r')
        sys.stdout.write(f'{m}: {int(i / h * 100)}%')
def get_rand(b):
    return int.from_bytes(os.urandom(b), byteorder='little')

def mod_exp(x, y, z):
    r = 1
    while y:
        if y % 2: #если чётно (y & 1)
            r = r * x % z
        y = y // 2
        x = x ** 2 % z
    return r

def bin_sqrt(x):
    start = 1
    end = x
    while start <= end:
        mid = (start + end) // 2
        if mid * mid == x:
            return mid
        elif mid * mid < x:
```

```
                start = mid + 1
                r = mid
            else:
                end = mid - 1
        return r

def gcd(x, y):
        while(y):
            x, y = y, x % y
        return x

def extended_gcd(a, b):
        x0, x1, y0, y1 = 0, 1, 1, 0
        while a:
            q, b, a = b // a, a, b % a
            y0, y1 = y1, y0 - q * y1
            x0, x1 = x1, x0 - q * x1
        return b, x0, y0
def mod_mul_inverse(a, b):
        g, x, _ = extended_gcd(a, b)
        return x % b
def trial_division(x):
        l = range(3, bin_sqrt(x), 2)
        i = 1
        if not x & 1: #x - чётное (x % 2)
            return False
        for y in l:
            show_stat(f'CHECK {x}', i, 100000, len(l))
```

```
            if not x % y:
                return False
            i += 1
        return True
def fermat (n, k):
        for _ in range(k):
            a = random.randint(2, n - 1)
            if mod_exp(a, n - 1, n) != 1:
                return False
        return True
def gen_prime(b = 128, v = 'd', k = 10):
        x = get_rand(b)
        t = time.time()
        if v == 'd':
            check = lambda: trial_division(x)
        else:
            check = lambda: fermat_test(x, k)
        while not check():
            x = get_rand(b)
        print(f'\nTRUE in {time.time() - t} seconds')
        return x
def main():
        print('Генерація простих чисел')
        p = gen_prime(6)
        q = gen_prime(6)
        #p = 3
        #q = 5
        N = p * q #модуль
```

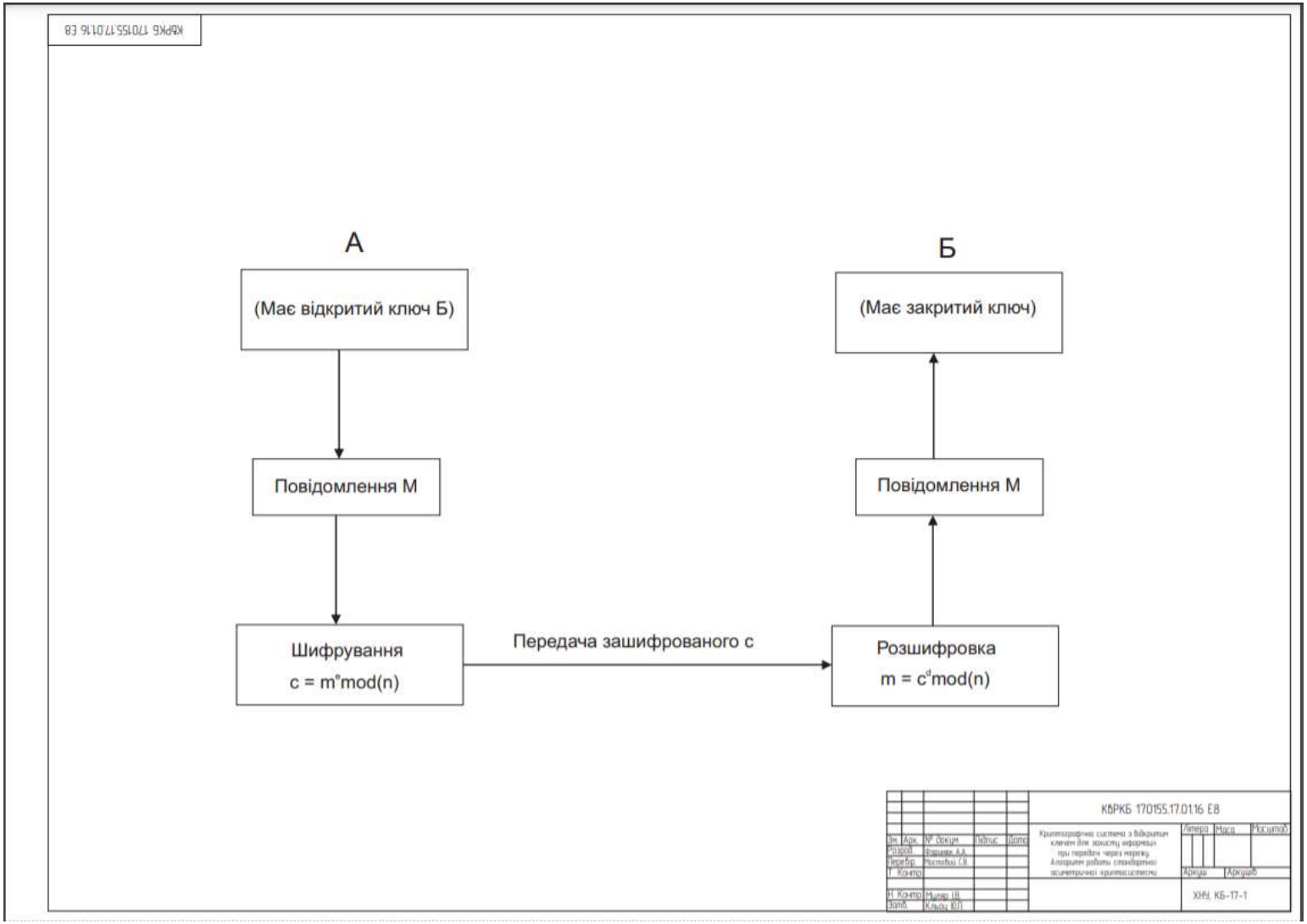
```
        print('Публічний:', N)
        M = ((p - 1) * (q - 1)) // gcd(p - 1, q - 1)
        print(Лямбда:', M)
        E = 65537
        D = mod_mul_inverse(E, M)
        print('Приватний:', D)
        m = get_rand(128)
        print(
f"""--------------
ПОВІДОМлення
--------------
{m}
--------------""")

        n = m // N
        r = m % N
        print(
f"""КІЛЬКІСТЬ ЧАСТИН: {n}
ЗАЛИШОК: {r}
ЧАСТИНА: {p}""")
        p_enc = mod_exp(p, E, N)
        r_enc = mod_exp(r, E, N)
        p_dec = mod_exp(p_enc, D, N)
        r_dec = mod_exp(r_enc, D, N)
        print(
f"""Закодований шлях: {p_enc}
Закодований залишок: {r_enc}
Розшифрований шлях: {p_dec}
```

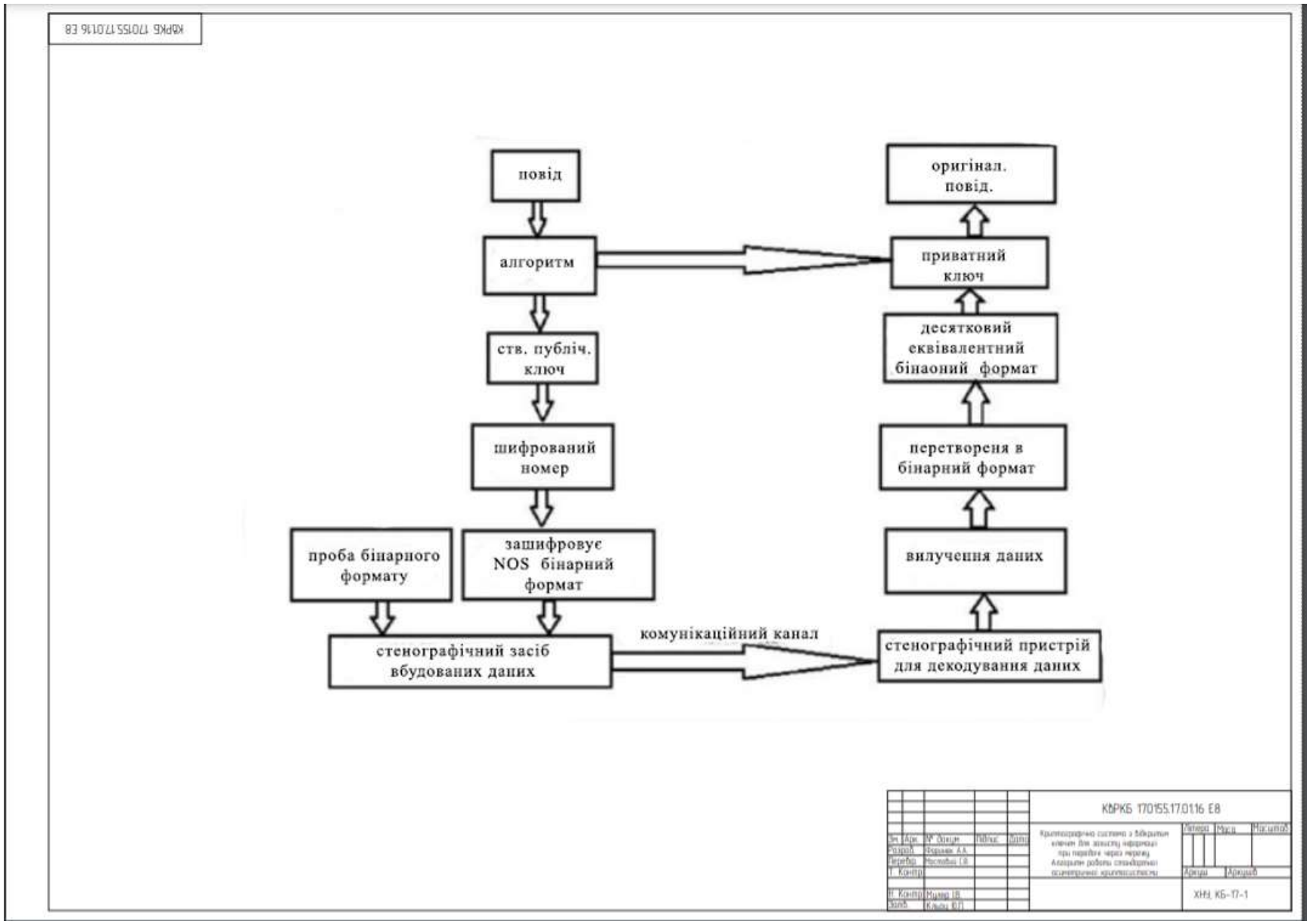
```
Розшифрований залишок: {r_dec}""")
        d = (p_dec + 1) * n + r_dec
        print(
f"""--------------
РОЗШИФРОВКА
--------------
{d}
--------------""")
main()
```

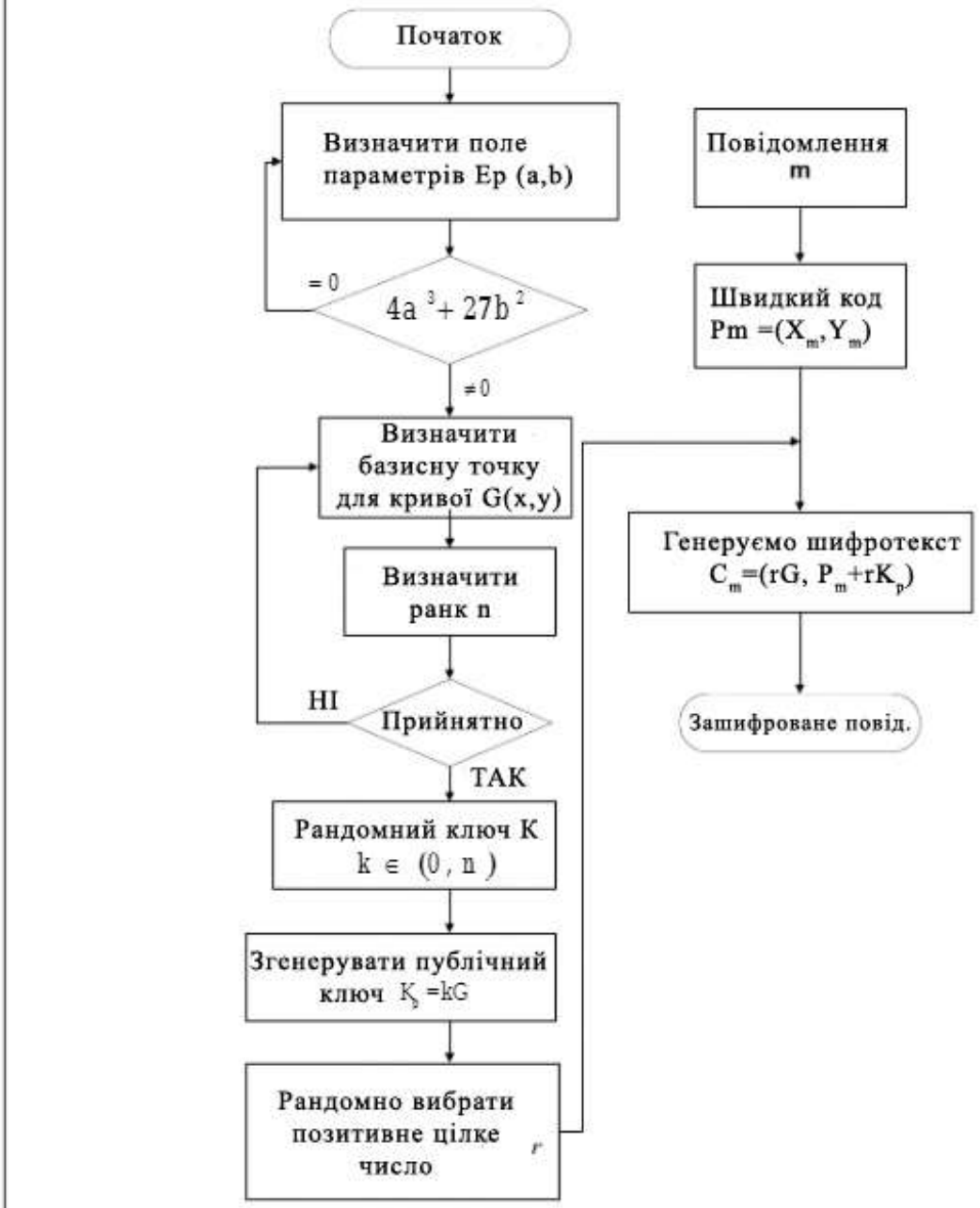
ДОДАТОК Б

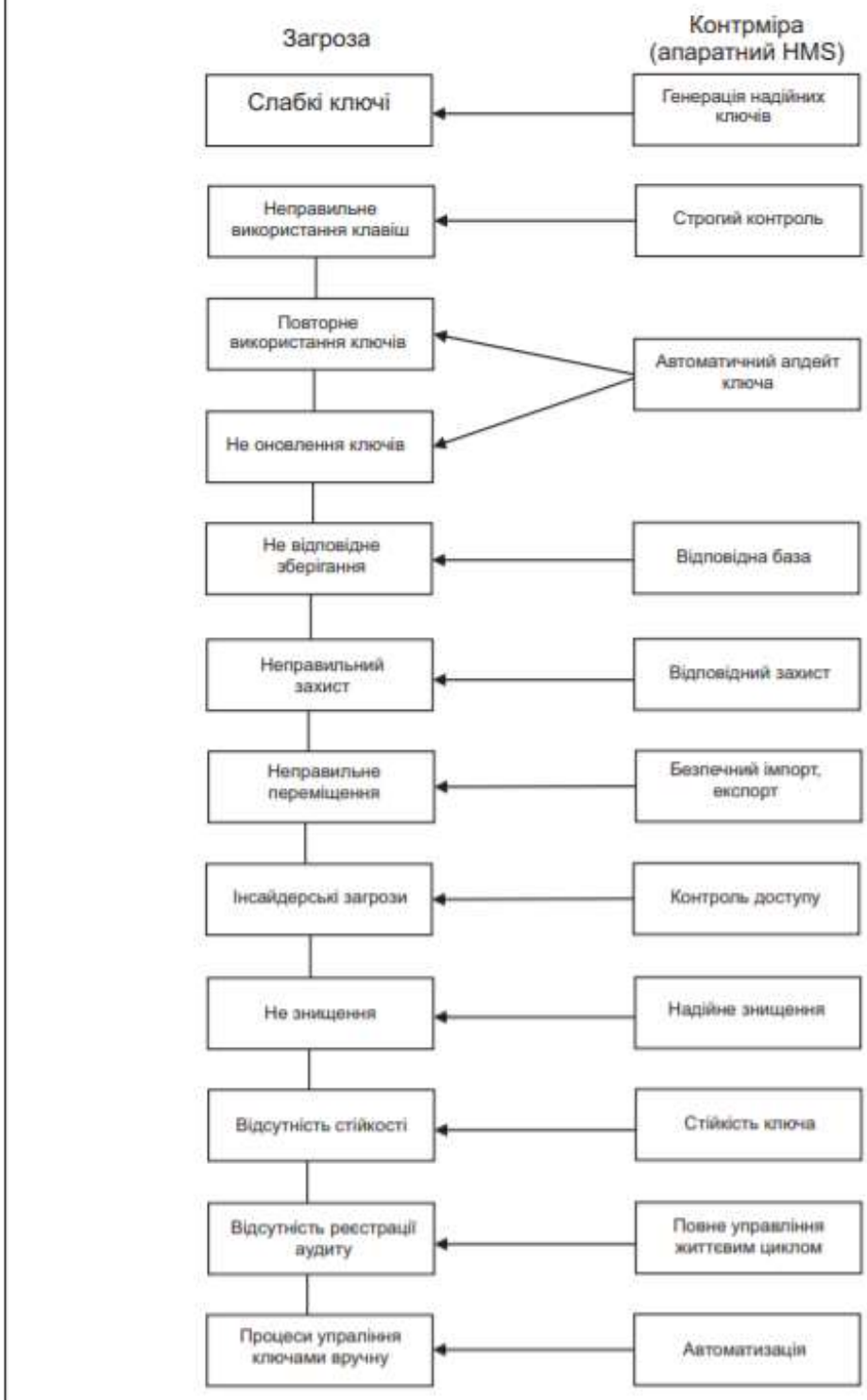
(Обов’язковий)

Копія графічної частини




повід
алгоритм
ств. публіч. ключ
шифрований номер
проба бінарного формату
зашифровує NOS бінарний формат
стеганографічний засіб вбудованих даних
комунікаційний канал
стеганографічний пристрій для декодування даних
вилучення даних
перетворення в бінарний формат
десятковий еквівалентний бінаоний формат
приватний ключ
оригінал. повід.


Початок
Визначити поле параметрів Ep (a,b)
= 0
$4a^3 + 27b^2$
≠ 0
Визначити базисну точку для кривої G(x,y)
Визначити ранк n
НІ
Прийнятно
ТАК
Рандомний ключ K
k ∈ (0, n)
Згенерувати публічний ключ $K_p = kG$
Рандомно вибрати позитивне цілке число r
Повідомлення m
Швидкий код $Pm = (X_m, Y_m)$
Генеруємо шифротекст $C_m = (rG, P_m + rK_p)$
Зашифроване повід.


Загроза
Контрміра (апаратний HMS)
Слабкі ключі
Генерація надійних ключів
Неправильне використання клавіш
Строгий контроль
Повторне використання ключів
Автоматичний апдейт ключа
Не оновлення ключів
Не відповідне зберігання
Відповідна база
Неправильний захист
Відповідний захист
Неправильне переміщення
Безпечний імпорт, експорт
Інсайдерські загрози
Контроль доступу
Не знищення
Надійне знищення
Відсутність стійкості
Стійкість ключа
Відсутність реєстрації аудиту
Повне управління життєвим циклом
Процеси управління ключами вручну
Автоматизація
КВРКБ 170155.17.0116 ЕВ

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Криптографічна система з відкритим ключем для захисту інформації при передачі через мережу

Автор: Фаринюк Анатолій Анатолійович

Спеціальність: 125 – Кібербезпека

Освітня програма: Кібербезпека

Керівник: Кльоц Юрій Павлович, завідувач кафедри, к.т.н., доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (дали – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (дали – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латиньких символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ідентичності/схожості, складає 0.53% що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи — Ю.П. Кльоц

Доцент кафедри КбКСМ, Гарант ОП — В.М. Чешун

Дата: 18.06.2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «бакалавр»

Студент Фаринюк Анатолій Анатолійович

Тема Криптографічна система з відкритим ключем для захисту інформації при передачі через мережу

Спеціальність 125 – Кібербезпека

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «бакалавр»:

кількість листів креслень 4 ; кількість сторінок записки 61

1. Короткий зміст роботи та прийнятих рішень У кваліфікаційній роботі розроблено розглянуто та розроблено криптографічну систему на основі відкритого ключа.

2. Висновок про відповідність кваліфікаційної роботи завданню Кваліфікаційна робота у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині роботи

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: Студент у вступі визначив основні питання актуальності систем на основі відкритого ключа. Були визначені задачі, необхідні для досягнення цілі кваліфікаційної роботи. У першому розділі проведено огляд існуючих криптосистем захисту, проаналізовано як системи впливають на сучасний кіберпростір, були розглянуті основні принципи та основи криптосистем на основі асиметричного шифрування, проаналізовано проблеми та загрози для такою системи криптозахисту. Другий розділ роботи студента включає детальний огляд найвідоміших алгоритмів з відкритим ключем. В третьому розділі був спроектований алгоритм програмного модуля та були проведені розрахунки. В четвертому розділі розроблена власна система криптозахисту на основі проаналізованих даних та власних знань студента.

4. Позитивні сторони роботи Кваліфікаційна робота має високу теоретичну цінність, оскільки системи з асиметричним шифруванням на сьогоднішній день є досить популярними і надійними. І окрім вже відомих алгоритмів студент проявив ерудованість у цих питаннях і сам реалізував одну з подібних проаналізованих систем.

5. Негативні сторони роботи У цій кваліфікаційній роботі студент присутня досить мала кількість практичного, наочного застосування алгоритму на основі відкритого ключа.

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення виконане відповідно до теми кваліфікаційної роботи з дотриманням стандартів. В загальному графічне оформлення виконане якісно, пояснювальна записка відповідає нормам щодо її оформлення.

7. Відгук про роботу в цілому В загальному кваліфікаційна робота заслуговує позитивної оцінки. Весь матеріал кваліфікаційної роботи структурований, чіткий та послідовний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики кваліфікаційної роботи.

8. Інші зауваження
Враховані лише найвідоміші алгоритми криптографії з відкритим ключем.

9. Оцінка кваліфікаційної роботи Враховуючи всі позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінку «добре».

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи)
Бездашний І.А. Зав. каф. ІПЗ

«13» 06 2021.

(підпис)

Thu Jun 17 15:15:07 EEST 2021, Музиря І.В, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 0.0%

Словари проверок: en_US, ru_RU, ua_UA. **Ошибок в документах: 6%**

ID: 94578 Название: Криптографічна система з відкритим ключем для захисту інформації при передачі через мережу Добавлено в БД: 2021-06-17 Авторы: Фаринюк Анатолій Руководители: Кльоц Ю.П. Консультанты: Оппоненты:	Документ		Суммарное совпадение по Базе Данных	
	Символы	Лексемы	Символы	Лексемы
	71244	575	90 (0%)	2 (0%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

UNICHECK

User name:
Кафедра кибербезпеки

Check ID:
1008321920

Check date:
17.06.2021 16:09:05 EEST

Check type:
Doc vs Internet

Report date:
17.06.2021 16:11:12 EEST

User ID:
100005590

File name: **Кваліфікаційна робота Фаринюк_пл**

Page count: **60** Word count: **11114** Character count: **83692** File size: **1.18 MB** File ID: **1008393796**

0.53% Matches

Highest match: **0.11%** with Internet source (**https://www.yaneuch.ru/cat_28/rozrobka-uml-dagrami-precendentv-ta/398161.2700...**

0.53% Internet sources **43** ..... Page 62

No Library search was conducted

0% Quotes

Exclusion of quotes is off

Exclusion of references is off

0% Exclusions

No exclusions

Modifind

Text modifications detected. Find more details in the online report.