НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут телекомунікаційних систем

Кафедра Телекомунікаційних систем

«На правах рукопису»
УДК 621.39

«До захисту допущено»
Завідувач кафедри
_____ Л.О. Уривський
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 172 Телекомунікації та радіотехніка
на тему: «Дослідження алгоритмів шифрування для забезпечення кібербезпеки телекомунікаційних систем та мереж»

Виконав:
студент ІІ курсу, групи ТС-81м
Пчелінцев Ілля Сергійович _____

Керівник:
доцент кафедри ТС, к.т.н., доцент
Григоренко Олена Григорівна _____

Рецензент:
професор кафедри ІТМ, д.т.н., с.н.с.
Скулиш Марія Анатолівна
_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем

Кафедра Телекомунікаційних систем

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (спеціалізація) – 172 «Телекомунікації та радіотехніка» (172.3620.1 «Телекомунікаційні системи та мережі»)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Л.О. Уривський

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Пчелінцеву Іллі Сергійовичу

1. Тема дисертації «Дослідження алгоритмів шифрування для забезпечення кібербезпеки телекомунікаційних систем і мереж», науковий керівник дисертації Григоренко Олена Григорівна, доцент, кандидат технічних наук, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації _____

3. Об’єкт дослідження криптографічні алгоритми шифрування на еліптичних кривих.

4. Предмет дослідження особливості генерації ключів шифрування при криптографії на еліптичних кривих.

5. Перелік завдань, які потрібно розробити

- дослідження основних відомостей про криптографію та шифрування для забезпечення кібербезпеки;
- аналіз особливостей асиметричних алгоритмів шифрування, що використовуються у сучасній криптографії;
- аналіз асиметричних алгоритмів криптографії, які використовують для параметри еліптичних кривих;
- розробка програмної реалізації криптографічних алгоритмів на еліптичних кривих.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу

Плакат № 1 «Тема, мета, актуальність, об’єкт, предмет, проблематика, завдання дослідження»

Плакат № 2 «Узагальнена схема криптосистеми»

Плакат № 3 «Класифікація алгоритмів шифрування»

Плакат № 4 «Еліптичні криві ($b=1, -3 \leq a \leq 2$)»

Плакат № 5 «Алгоритм генерації пари ключів та алгоритм генерації електронного підпису»

Плакат № 6 «Результат програмної реалізації алгоритму генерації пари ключів та алгоритму генерації електронного підпису»

Плакат № 7 «Висновки»

7. Перелік публікацій

Пчелінцев І.С., Григоренко О.Г. ЗАХОДИ ДЛЯ ЗАХИСТУ ВІД АТАК ДЛЯ ЗАБЕЗПЕЧЕННЯ КІБЕРБЕЗПЕКИ ОРГАНІЗАЦІЙ. ПРІТС 2019. - К.: КПІ ім. Ігоря Сікорського, 2019.

Пчелінцев І.С. ПОРІВНЯННЯ СУЧАСНИХ АЛГОРИТМІВ ШИФРУВАННЯ. ПРІТС 2019. - К.: КПІ ім. Ігоря Сікорського, 2019.

8. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Огляд літературних джерел по тематиці роботи	01.09.18 - 01.11.18	
2.	Формування мети та наукових завдань дослідження	01.11.18 - 01.12.18	
3.	Формування структури роботи та наповнення теоретичної частини	01.01.19 - 15.03.19	
4.	Підготовка матеріалів для доповіді на конференції	15.03.19 - 16.04.19	
5.	Формування вступної частини роботи	18.04.19 - 15.05.19	
6.	Проведення досліджень для вирішення наукових завдань	15.05.19 - 01.09.19	
7.	Формування останнього розділу роботи, висновків	01.09.19 - 01.11.19	
8.	Оформлення плакатів та пояснювальної записки дипломної роботи	01.11.19 - 01.12.19	
9.	Підготовка до захисту дипломної роботи	01.12.19 - 16.12.19	

Студент Пчелінцев І.С.

Науковий керівник дисертації Григоренко О.Г.

РЕФЕРАТ

Актуальність теми обумовлена постійним зростанням кількості абонентів у мережі, що потребують забезпечення конфіденційності даних, які вони передають через мережу.

Мета даної роботи полягає у дослідженні сучасних методів криптографії на еліптичних кривих та їх практичній реалізації.

Об’єктом дослідження є криптографічні алгоритми шифрування на еліптичних кривих.

Предмет дослідження особливості генерації ключів шифрування при криптографії на еліптичних кривих.

В даній роботі досліджуються загальні відомості про криптографію та шифрування, відбувається аналіз особливостей асиметричних алгоритмів шифрування, а особливо криптографічних алгоритмів шифрування на еліптичних кривих. Програмно реалізується один з алгоритмів для підтвердження роботоздатності.

ABSTRACT

The relevance of the topic is due to the constant increase in the number of subscribers in the network, who need to ensure the confidentiality of the data they transmit through the network.

The purpose of this work is to study modern methods of cryptography on elliptic curves and their practical implementation.

The object of the study is cryptographic algorithms for encryption on elliptic curves.

The subject of the study is the features of the generation of encryption keys in cryptography on elliptic curves.

This paper investigates the general information about cryptography and encryption, analyzes the features of asymmetric encryption algorithms, and in particular cryptographic encryption algorithms on elliptic curves. One of the algorithms for confirmation of working capacity is implemented in software.

.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ІКТ - інформаційно-комунікаційних технологіях.

КСЗІ - комплексну систему захисту інформації.

КС - криптографічні системи.

ГВП - генераторів випадкових послідовностей.

DES - Data Encryption Standart – стандарт шифрування даних.

IDEA - International Data Encryption Algoritm – міжнародний алгоритм шифрування даних.

ІВП - істинновипадкові послідовності.

ЕЦП - електронних цифрових підписів.

ЦСК - центром сертифікації ключів.

ІВК - інфраструктури відкритих ключів.

ПЗ – програмне забезпечення.

ECDLP - Elliptic Curve Discrete Logarithm Problem - задача дискретного логарифмування для еліптичних кривих.

ECDSA - Elliptic Curve Digital Signature Algorithm – алгоритм цифрових підписів на еліптичних кривих.

ЕК - еліптична крива.

SECG - Standards for Efficient Cryptography Group – група стандартів ефективної криптографії.

ВСТУП

Разом з розвитком людства стрімко розвиваються і телекомунікаційні технології, які вже сьогодні здатні об’єднати людей з різних куточків світу та забезпечити їх можливістю швидкого спілкування майже без обмежень. Однак розвиток телекомунікаційних систем і мереж на цьому не закінчується, а продовжує швидко розвиватися далі додаючи до мереж абонентів не лише людей, а й сервіси та різні апарати.

Все більше конфіденційних даних людей поширюється у мережах, що вимагає від мереж певного рівня захищеності для запобігання втручання в особисте життя користувачів. Цей рівень захищеності забезпечується спеціальними криптографічними алгоритмами, за допомогою яких дані перш ніж відправитися до отримувача певним чином шифруються, що запобігає тому, що третя сторона отримає доступ до цих даних.

В даній роботі буде зроблено акцент на асиметричних алгоритмах шифрування, які хоч і мають певні недоліки порівняно з симетричними, однак дозволяють значно легше організовувати зв'язок великих кількостей людей, що в умовах сучасності є дуже важливим.

Особливу увагу буде приділено криптографічним алгоритмам на еліптичних кривих, особливість яких полягає у великій варіативності можливих ключів шифрування навіть на одній єдиній кривій, що є значною перевагою у перспективі на майбутнє.

РОЗДІЛ 1 ОСНОВНІ ВІДОМОСТІ ПРО КРИПТОГРАФІЮ ТА ШИФРУВАННЯ

1.3 Криптографічні системи та їх стійкість

Шифрування - це спосіб заміни повідомлення чи якогось іншого документа, що забезпечує змінення (приховування) його змісту. Кодування - це перетворення абсолютно звичайного тексту в кодову послідовність. При цьому вважається, що існує взаємно однозначна відповідність між символами повідомлення (даних, чисел, слів) і символьного коду - в цьому принципова різниця між кодуванням та шифруванням. Часто кодування і шифрування вважають одним і тим самим, забуваючи, що для відновлення закодованого повідомлення, достатньо знати спосіб підстановки (заміни). Для відновлення ж зашифрованого повідомлення крім знання правил шифрування, потрібен і ключ до шифру. Ключ розуміється, як конкретний секретний стан параметрів алгоритмів шифрування і дешифрування. Знання ключа дає можливість прочитання засекреченого повідомлення. Втім, далеко не завжди незнання ключа гарантує те, що повідомлення не зможе прочитати стороння людина.

“Шифрувати можна не тільки текст, але і комп'ютерні файли - від файлів баз даних і текстових процесорів до файлів зображень. Шифрування використовується людьми з того самого моменту, як з'явилася перша секретна інформація.

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

Шифрування з'явилося приблизно чотири тисячі років тому. Першим відомим застосуванням шифру (коду) вважають єгипетський текст, датований

приблизно 1900 р. до н. е., автор якого використав замість звичайних (для єгиптян) ієрогліфів знаки ,що з ними не збігаються” [1].

Як вказано у [2] - “один з найвідоміших методів шифрування носить ім'я Цезаря, який якщо і не винайшов його, то активно його використовував. Не довіряючи своїм гінцям, він шифрував листи елементарної заміною літер по всьому латинському алфавіту. При такому кодуванні комбінація XYZ була б записана як ABC(прямий код N+3). Через 500 років шифрування стало всюди використовуватися при записі текстів релігійного змісту, молитов і важливих державних документів. Із середньовіччя і до наших днів необхідність шифрування військових, дипломатичних і державних документів підсилювала розвиток криптографії.

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

• паролі для доступу до мережевих ресурсів (домен, принтер, комп'ютери в мережі і т.п.);

• паролі для доступу в Інтернет за допомогою DialUp;

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

• сертифікати для доступу до мережевих ресурсів і зашифрованих даних на самому комп'ютері” [2].

Ці дані зберігаються або в pwl-файлі (в Windows 95), або в SAM-файлі (в Windows NT/2000/XP). Це файли Реєстру Windows, і тому операційна система нікому не дає до нього доступу навіть для читання. Зловмисник може скопіювати ці файли, тільки завантажившись в ОС або з дискети. Утиліт для їх злому багато, найсучасніші з них можуть підібрати ключ за кілька годин чи швидше.

“Криптографічна система - це сукупність засобів криптографічного захисту інформації, необхідної нормативної, експлуатаційної та іншої документації, які складають єдину систему з метою розв'язання конкретної задачі захисту інформації. Криптосистема складається з таких базових підсистем: шифрування, забезпечення цілісності (імітозахисту), ідентифікація, цифрового підпису тощо; кожна з них має свою підсистему ключів” [3].

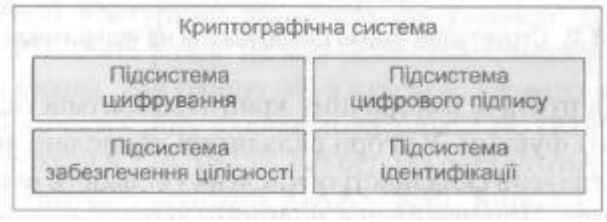

Рисунок 1.1 Структура криптографічної системи

“Криптографічна стійкість — здатність криптографічного алгоритму протистояти криптоаналізу. Стійким вважається алгоритм, який для успішної атаки вимагає від зловмисника недосяжних обчислювальних ресурсів, недосяжного обсягу перехоплених відкритих і зашифрованих повідомлень чи ж такого часу розкриття, що по його закінченню захищена інформація буде вже не актуальна, і т. д. У більшості випадків крипостійкість не можна математично довести, можна тільки довести уразливість криптографічного алгоритму”[3].

Типи криптостійких систем шифрування

Абсолютно стійкі системи

“Доказ існування абсолютно стійких алгоритмів шифрування виконав Клод Шенон і було опубліковано в його роботі «Теорія зв'язку в секретних системах». Там же визначено вимоги до такого роду систем:

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

- Ключ статистично надійний (тобто ймовірності появи кожного з можливих символів рівні, символи в ключовий послідовності незалежні і випадкові)
- Довжина ключа дорівнює або більше довжини повідомлення
- Вихідний (відкритий) текст має деяку надмірність (є критерієм оцінки правильності розшифрування)" [4]

Стійкість цих систем не залежить від того факту, якими обчислювальними можливостями володіє криптоаналітик. Практичне застосування систем, що відповідають вимогам абсолютної стійкості, обмежене міркуваннями вартості і зручності користування.

Деякі аналітики стверджують, що Шифр Вернама є одночасно абсолютно криптографічно стійким і єдиним шифром, який задовольняє цій умові.

Досить стійкі системи

"В основному застосовуються практично стійкі чи обчислювально стійкі системи. Стійкість цих систем залежить від того факту, якими обчислювальними можливостями володіє криптоаналітик. Практична стійкість таких систем спирається на теорію складності і оцінюється виключно на певний момент часу і послідовно с двох позицій:

- Обчислювальна складність повного перебору
- Відомі на даний момент уразливості та їх вплив на обчислювальну складність.

У кожному конкретному випадку можуть існувати додаткові критерії оцінки стійкості"[4].

Оцінка криптостійкості систем шифрування

Початкова оцінка

Оскільки атака методом «грубої сили» (повним перебором всіх ключів) можлива для майже всіх типів криптографічних алгоритмів, крім абсолютно стійких «по Шеннону», для новоствореного алгоритму вона може бути єдиною існуючою. Способи її оцінки ґрунтуються на обчислювальній складності, яка може бути виражена у часі, грошах, і необхідній продуктивності обчислювальних ресурсів. Ця оцінка поки є максимальною і мінімальною одночасно.

Поточна оцінка

“Подальше дослідження алгоритму з метою пошуку вразливостей (криптоаналіз) додає оцінки стійкості по відношенню до відомих криптографічних атак (Лінійний криптоаналіз, диференціальний криптоаналіз і більш специфічні) і можуть знизити відому стійкість.

Наприклад, для багатьох симетричних шифрів існують слабкі ключі, застосування яких знижує криптографічну стійкість. Також важливим способом перевірки стійкості є проведення атаки на реалізацію, що виконуються для окремого програмно-апаратно-людського комплексу.

Чим тривалішим і експертним є аналіз алгоритму і реалізацій, тим більше достовірним можна вважати його стійкість. У кількох випадках тривалий і уважний аналіз приводив до зменшення оцінки стійкості нижче необхідного рівня. Недостатня перевірка (на думку багатьох криптографів — штучне ослаблення) може призвести до успішної атаки”[4].

Безпека інформації в інформаційно-комунікаційних технологіях (ІКТ) забезпечується використанням організаційних та інженерно-технічних заходів, засобів і методів захисту інформації, що формують комплексну систему захисту інформації (КСЗІ). Найважливішими складовими частинами КСЗІ є

криптографічні системи (КС). Сучасні ІКТ потребують від криптосистем забезпечення стійкості та надійності захисту інформації при визначених умовах функціонування.

Криптостійкість майже всіх сучасних КС, повинна забезпечуватися лише параметрами (ключовими даними) криптоалгоритму, тому стійкість КС на пряму залежить від ефективності функціонування систем управління ключами (ключових систем).

“Генерація якісної випадкової послідовності – найскладніша частина більшості криптографічних операцій. Показано, що для забезпечення стійкості шифрування потрібно забезпечити надійний ключ, а це в свою чергу, досягається використанням криптографічно якісних генераторів випадкових послідовностей (ГВП).

Криптографічні методи захисту інформації - це методи захисту даних із використанням шифрування.

Головна мета шифрування інформації - її захист від несанкціонованого читання. Системи криптографічного захисту інформації (системи шифрування інформації) можна поділити за різними ознаками:

• за принципами використання криптографічного захисту (вбудований у систему або додатковий механізм);

• за способом реалізації (апаратний, програмний, програмно-апаратний);

• за криптографічними алгоритмами, які використовуються (загальні, спеціальні);

• за цілями захисту (забезпечення конфіденційності інформації користувача та захисту повідомлень і даних від зміни, регулювання доступу та привілеїв користувачів);

• за методом розподілу криптографічних ключів (базових або сеансових ключів, відкритих ключів) тощо”[5].

Додаткові механізми криптозахисту - це програмні або апаратні засоби, які не входять до складу системи, а додаються окремо. Така реалізація механізмів криптозахисту гнучка і має можливість швидкої заміни. Для більшої ефективності можна використовувати комбінацію додаткових і вбудованих механізмів криптографічного захисту.

“За способом реалізації захист інформації можна здійснювати різними способами: апаратним, програмним або програмно-апаратним. Апаратна реалізація криптографічного захисту – найнадійніший спосіб, але й найдорожчий. Інформація для апаратних засобів передається в електронній формі через обчислювальну машину всередину апаратури, де виконується шифрування інформації. Перехоплення та підробка інформації під час передачі в апаратуру може виконуватись за допомогою спеціальних програм типу "вірус".

Програмна реалізація криптографічного захисту набагато дешевша та гнучкіша в реалізації. Але виникають питання з приводу захисту криптографічних ключів від несанкціонованого доступу під час роботи програми та після її завершення. Тому, крім захисту від "вірусних" атак, потрібно вживати заходів для забезпечення повної очистки пам'яті від криптографічних ключів, що використовувались в процесі роботи програм.

Крім того, доцільно використовувати комбінацію апаратних і програмних методів криптографічного захисту. Найчастіше використовують програмну реалізацію криптоалгоритмів з апаратним зберіганням ключів. Такий спосіб криптозахисту досить надійний і не дуже дорогий. Але, обираючи апаратні засоби для зберігання криптографічних ключів, необхідно пам'ятати про забезпечення захисту від викрадення ключів під час їх зчитування з носія та використання в програмі.

В основу шифрування покладено два елементи: криптографічний алгоритм і ключ”[6].

Криптографічний алгоритм - це математична функція, яка комбінує відповідне повідомлення або іншу зрозумілу інформацію з ланцюжком чисел (ключем) для отримання незв’язаного (шифрованого) тексту.

“Усі криптографічні алгоритми можна поділити на дві групи: загальні і спеціальні.

Спеціальні криптоалгоритми мають неявний алгоритм шифрування, а загальні криптоалгоритми характерні наявністю повністю відкритого алгоритму, і їх криптостійкість визначається лише ключами шифрування. Спеціальні алгоритми найчастіше використовують саме в апаратних засобах криптозахисту.

Загальні криптографічні алгоритми дуже часто стають стандартами шифрування, якщо їхня наявна висока криптостійкість доведена. Ці алгоритми оприлюднюють усім для обговорення, при цьому навіть визначається премія за успішну спробу його зламу. Криптостійкість загальних алгоритмів визначається виключно ключем шифрування, який генерується методом випадкових чисел і ніяк не може бути повторений протягом певного часу. Криптостійкість таких алгоритмів буде зростати відповідно до збільшення довжини ключа”[5].

Є дві великі групи загальних криптоалгоритмів: симетричні і асиметричні. До симетричних криптографічних алгоритмів відносять такі алгоритми, для яких шифрування і розшифрування відбувається однаковим ключем, тобто і відправник, і отримувач повідомлень мають користуватися одним і тим самим ключем. Такі алгоритми мають відносно велику швидкість обробки як для апаратної, так і для програмної реалізації. Основним недоліком є важкість, пов'язана з дотриманням безпечного розподілу ключів між учасниками системи. Для асиметричних криптоалгоритмів шифрування і розшифрування відбувається за допомогою різних ключів, тобто, маючи лише один із ключів, не можна визначити парний для системи ключ. Такі алгоритми часто потребують набагато більше часу для обчислення, але не викликають

труднощів під час розподілу ключів, бо відкритий розподіл одного з ключів ніяк не зменшує криптостійкості алгоритму і не дає жодної можливості відновлення парного йому ключа.

Усі криптографічні алгоритми можна використовувати для різних цілей, зокрема:

• для шифрування інформації, тобто приховування змісту тексту чи даних;

• для забезпечення захисту даних і повідомлень від зміни.

“З найбільш поширених методів шифрування можна виділити американський алгоритм шифрування DES (Data Encryption Standart, розроблений фахівцями фірми IBM і затверджений урядом США 1977 року) із довжиною ключа, яку можна змінювати, та алгоритм ГОСТ 28147-89, який було розроблено та який набув широкого застосування в колишньому СРСР і має ключ фіксованої довжини. Ці алгоритми належать до симетричних алгоритмів шифрування.

Алгоритм DES3 був запропонований як альтернатива DES і призначений для потрійного шифрування даних трьома різними закритими ключами для часткового підвищення ступеня захисту.

RC2, RC4, RC5 - шифри зі змінюваною довжиною ключа для дуже швидкого шифрування дуже великих обсягів інформації. Вони здатні підвищувати ступінь захисту через вибір більшого ключа. IDEA (International Data Encryption Algoritm) призначений для досить швидкої роботи в програмній реалізації. Для приховування інформації також можна використовувати різні асиметричні алгоритми, наприклад, алгоритм RSA. Алгоритм також підтримує змінну довжину ключа та ще змінний розмір блоку тексту, що шифрується”[5].

Алгоритм RSA дозволяє виконувати шифрування в різних режимах:

• за допомогою таємного ключа від відправника. Тоді всі, хто має саме його відкритий ключ, можуть розшифрувати це повідомлення;

• за допомогою відкритого ключа від отримувача, тоді тільки власник таємного ключа, який є виключно парним до цього відкритого, може розшифрувати таке повідомлення;

• за допомогою таємного ключа від відправника і відкритого ключа від отримувача повідомлення. Тоді тільки цей отримувач може розшифрувати повідомлення від відправника.

Але не всі асиметричні алгоритми дозволяють хоч якось виконувати шифрування даних у таких режимах. Це визначається математичними функціями алгоритмів.

“Іншою метою для використання криптографічних методів є захист інформації від заміни, викривлення або підробки. Цього можна досягти без шифрування повідомлень, тобто повідомлення можна залишити відкритим, незашифрованим, але до нього додається інформація, перевірка якої за допомогою особливих алгоритмів може однозначно довести, що ці дані не були змінені. Для симетричних алгоритмів шифрування, наприклад, така додаткова інформація – це, так званий, код автентифікації, який формується за наявності лише ключа шифрування за допомогою криптографічних алгоритмів.

Для асиметричних криптографічних алгоритмів також формують додаткову інформацію, яка називається електронний цифровий підпис. Формуючи електронний цифровий підпис, виконують такі операції:

• за допомогою односторонньої хеш-функції визначають прообраз цифрового підпису, аналог контрольної суми повідомлення;

• отримане значення хеш-функції шифрується: а) таємним або відкритим; 6) таємним та відкритим ключами від відправника і від отримувача повідомлення - для алгоритму RSA

• використовуючи значення хеш-функції та також таємного ключа, за допомогою спеціального алгоритму визначають значення цифрового підпису.

Для того, щоб перевірити цифровий підпис, потрібно:

• виходячи із визначеного значення цифрового підпису та використовуючи лише відповідні ключі, обчислити значення хеш-функції;

• обчислити хеш-функцію для тексту повідомлення;

• порівняти ці значення. Якщо вони збігаються, то повідомлення не було змінено та відправлене саме цим відправником”[5].

Останнім часом використання електронного цифрового підпису дуже поширюється, у тому числі для обмеження доступу до конфіденційної банківської інформації та ресурсів системи. Ефективність захисту систем з використанням будь-яких криптографічних алгоритмів дуже залежить від безпечного розподілу ключів. Тому можна виділити такі основні методи розподілу ключів між учасниками системи.

1. Метод базових/сеансових ключів. Цей метод описаний у стандарті ISO 8532 і використовується для розподілу ключів серед симетричних алгоритмів шифрування. Для розподілу ключів вводиться така ієрархія ключів: головний ключ і ключ шифрування даних (тобто сеансовий ключ). Ієрархія також може бути і дворівневою: ключ для шифрування ключів/ключ для шифрування даних. Старший ключ у цій ієрархії необхідно розповсюджувати неелектронним способом, що виключає можливість його компрометації. Використання цієї схеми розподілу ключів потребує як значного часу так і значних затрат.

2. Метод відкритих ключів. Цей метод описаний у стандарті ISO 11166 і використовується для розподілу ключів для симетричного і асиметричного шифрування. З його допомогою можна забезпечити надійне функціонування всіх центрів сертифікації ключів для електронного цифрового підпису з використанням асиметричних алгоритмів та розподіл сертифікатів відкритих ключів для учасників інформаційних систем. Крім цього, використання методу відкритих ключів надає можливість кожне повідомлення шифрувати окремим

ключем від симетричного алгоритму та передавати цей ключ разом із самим повідомленням у зашифрованій асиметричним алгоритмом формі.

Вибір того чи іншого методу залежить як від структури системи так і від технології обробки даних. Жоден із цих методів не забезпечує повного захисту інформації, але гарантує, що вартість викриття (зламу) у кілька разів перевищує вартість зашифрованої інформації. Щоб користуватися системою криптографії з відкритим ключем, необхідно генерувати відкритий та особистий ключі. Після генерування ключової пари потрібно розповсюдити відкритий ключ кореспондентам. Найнадійніший спосіб для розповсюдження відкритих ключів - через сертифікаційні центри, які призначені для зберігання цифрових сертифікатів.

“Цифровий сертифікат - це електронний ідентифікатор, який підтверджує справжність особи користувача, він містить певну інформацію про нього, а також слугує електронним підтвердженням для відкритих ключів. Сертифікаційні центри несуть повну відповідальність за перевірку особистості користувача, надання цифрових сертифікатів та за перевірку їх справжності”[5].

Стійка криптографія

“Криптографія може бути як стійкою, так може бути і слабкою. Криптографічна стійкість вимірюється тим, як багато знадобиться часу і ресурсів, щоб із шифротексту відновити вихідний текст. Результатом стійкої криптографії є шифротекст, який дуже складно зламати без володіння визначеними інструментами для дешифрування. Використовуючи весь наявний обчислювальний потенціал сучасної цивілізації — навіть мільярди комп'ютерів, що виконують мільярди операцій за секунду — неможливо дешифрувати результат стійкої криптографії до кінця існування Всесвіту.

Хтось може вирішити, що стійка криптографія в змозі встояти навіть проти дуже серйозного криптоаналітика. Ніким ще не доведено, що доступне

сьогодні стійке шифрування, зможе встояти проти обчислювальних можливостей комп'ютерів, що будуть доступних завтра.

Стосовно абсолютно стійкої криптографії, то математичний доказ того, за яких умов шифр Вернама є абсолютно стійким шифром, тобто таким шифром, який неможливо зламати за будь-яких обставин, зробив американський дослідник К. Е. Шеннон у своїй класичній праці «Теорія зв'язку у секретних системах», що була розсекречена та вперше опублікована у США у відкритому друці у 1949 р”[7].

1.4 Вимоги до криптографічних систем та шифрів

“Криптографічна система захисту інформації — це така сукупність криптографічних алгоритмів, протоколів і таких процедур формування, розподілу, передачі та використання криптографічних ключів.

Нехай, відправник хоче послати повідомлення одержувачеві. Більш того, цей відправник дуже хоче послати своє повідомлення безпечно: він хоче бути впевнений в тому, що в разі перехоплення повідомлення зловмисник ніяк не зможе довідатися про зміст повідомлення. У цьому разі відправникові буде доцільніше використати криптографічні перетворення.

Саме повідомлення називається відкритим текстом. Зміна виду повідомлення для того щоб приховати його суть називається шифруванням. Шифроване повідомлення буде називатися шифротекстом. Процес перетворення шифротексту у відкритий текст називається розшифруванням. Узагальнену схему криптографічної системи, що забезпечує шифрування переданої інформації, показано на рисунку 1.2”[8].

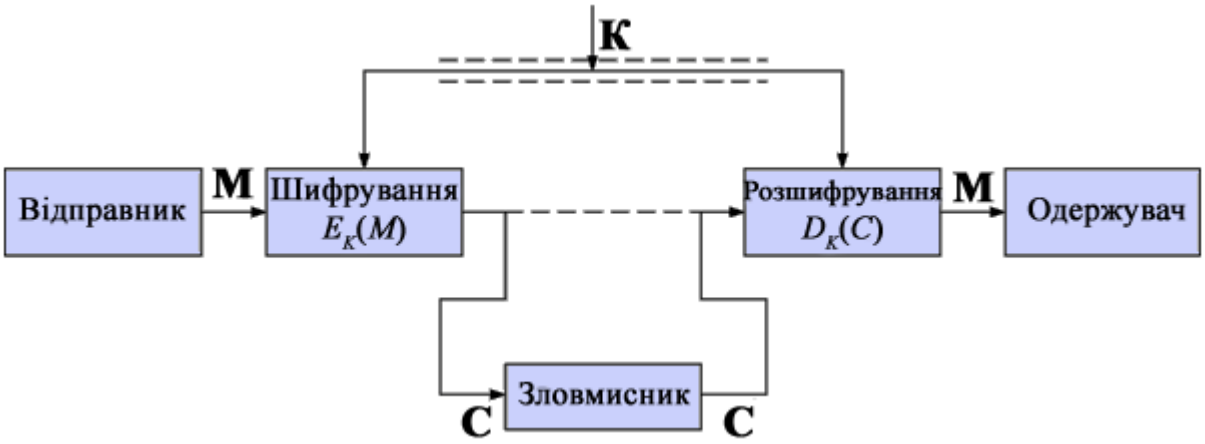

Рисунок 1.2 Узагальнена схема криптосистеми

“Криптоаналізом називається такий розділ прикладної математики, що вивчає різні моделі, методи, алгоритми, та програмні й апаратні засоби аналізу криптосистеми або її вхідних та вихідних сигналів з метою отримання секретних параметрів, також включаючи відкритий текст. Криптоаналіз займається завданнями, котрі в математичному змісті зворотні завданням криптографії. Система криптографії й криптоаналізу утворює криптологію.

Позначимо відкрите повідомлення як *M*. Це може бути потік бітів, чи текстовий файл, або бітове зображення, чи оцифрований звук, або цифрове відеозображення та ін. Далі будемо розглядати тільки двійкові дані й комп’ютерну криптографію”[8].

Якщо позначити шифротекст як *C*. Це також двійкові дані, інколи того ж розміру, що й *M*, інколи більше. Якщо процес шифрування супроводжується стисненням, *C* може бути менше *M*. Однак при цьому саме шифрування не забезпечує стиснення інформації. Функція шифрування *E* діє на відкритий текст, створюючи саме шифротекст $E(M) = C$.

“Процес відновлення відкритого тексту по шифротексту називається розшифруванням і виконується за допомогою такої функції розшифрування D:$D(C) = M$.

Оскільки змістом шифрування та розшифрування повідомлення є відновлення первинного відкритого тексту, то повинна виконуватись тотожність $D(E(M)) = M$.

Криптографічний алгоритм, який також називається шифром, являє собою математичну функцію, котра використовується для як для шифрування так и для розшифрування. Якщо безпека його алгоритму заснована на збереженні самого алгоритму в таємниці, то це буде обмежений алгоритм. Обмежені алгоритми являють лише історичний інтерес, але вони повністю не відповідають сьогоднішнім стандартам. Велика або змінна група користувачів не може використати такі алгоритми тому, що коли користувач залишає групу, її члени повинні переходити на інший алгоритм. Алгоритм також повинен бути замінений, якщо хтось ззовні випадково довідається секрет"[8].

Сучасна криптографія також розв'язує проблеми обмежених алгоритмів з використанням ключа K. "Ключ — це конкретний секретний стан у деяких певних параметрів алгоритму криптографічного перетворення даних, який забезпечує вибір лише одного варіанта перетворення з усіх можливих для даного алгоритму. Множину можливих ключів також називають простором ключів. Ключ, який використовується для ініціалізації системи, досить часто називають майстер-ключем системи.

З урахуванням використання ще й ключа, функції шифрування й розшифрування запишуться як $C = E_K(C)$ і $D_K(C) = M$. При цьому має виконуватися така тотожність $D_K(E_K(M)) = M$. Однак для деяких алгоритмів в процесі шифрування й розшифрування використовуються різні ключі. У цьому разі $C = E_{K1}(M)$, $D_{K2}(C) = M$, а $D_{K2}(E_{K1}(M)) \equiv M$. Якщо алгоритм перетворення даних залежить від ключа, інакше кажучи застосовуються управляючі операції, шифр називається шифром, що управляється"[8].

Криптографічні системи, загалом, класифікуються на основі таких трьох незалежних характеристик:

1) тип операцій для перетворення відкритого тексту в шифрований;

2) число ключів, які використовуються;

3) метод обробки відкритого тексту.

“Блочне шифрування також передбачає обробку відкритого тексту блоками, таким чином що в результаті обробки кожного блоку отримуємо блок шифрованого тексту. При використанні потокового шифрування, шифрування всіх елементів відкритого тексту буде здійснюється послідовно, одне за іншим, тому на кожному етапі отримують по одному елементу шифрованого тексту.

До шифрів, котрі використовуються для криптографічного захисту інформації, висувають ряд вимог:

- статистична безпека алгоритмів;
- надійність математичної бази алгоритмів;
- простота процедур шифрування й розшифрування;
- незначна надмірність інформації за рахунок шифрування;
- простота реалізації алгоритмів на різній апаратній базі.

Тією чи іншою мірою цим вимогам відповідають:

- шифри перестановок;
- шифри заміни;
- шифри гамування;
- шифри, засновані на аналітичних перетвореннях даних.

Основним питанням аналізу для будь-якої криптографічної системи захисту інформації є визначення її ступеня стійкості. Стійкість криптографічної системи захисту інформації це її здатність протистояти атакам зловмисника на інформацію, що захищається”[8].

Сучасні методи генерування ключових даних допускають використання різних генераторів випадкових послідовностей і засобів формування та тестування для ключів криптографічної системи.

Існуючі методи генерування криптографічно якісних ВП можна поділити на: *істинновипадкові*, *псевдовипадкові.*

Істинновипадкові послідовності (ІВП). Для отримання істинновипадкових послідовностей (ІВП) як джерела ентропії використовуються фізичні явища природи. У деяких криптографічних системах можуть використовуватися екзотичні (неспеціалізовані) методи отримання первинної ентропії [3], наприклад:

- тривалість натискання клавіш на клавіатурі («миші») та періоди між натисканням;

- параметри локальної (глобальної) комп'ютерної мережі;

- коливання часу доступу до жорсткого диску у зв'язку із турбулентністю повітря в його корпусі ;

- шуми звичайної цифрової камери, розміщеної у темряві тощо.

Але ці генератори мають дуже малу ентропію, низьку продуктивність та характеризуються досить високою кореляцією вихідної послідовності, тому вони не можуть бути використані для прямого застосування, а лише як криптографічно слабкі джерела для отримання початкових даних (ініціалізації) псевдовипадкових ГВП.

Спеціалізовані криптографічно сильні генератори ІВП як джерела ентропії використовують радіоактивний розпад, фізичні явища оптично-квантової механіки чи електричні шуми. Радіоактивний розпад є одним із «фундаментальних» фізичних процесів, який використовується для отримання криптографічно якісних випадкових послідовностей

Дійсно, припустити, що атомарне ядро розпадеться в дану секунду або в іншу – досить непередбачуване твердження. Для отримання первинної випадковості зазвичай використовують періоди між щигликами в лічильнику Гейгера. Довготривалість, стабільність, а головне - непередбачуваність такого фізичного процесу – головні переваги генераторів ІВП, які використовують

радіоактивний розпад. До недоліків таких генераторів зазвичай відносять: необхідність використання радіоактивних матеріалів, що призводить до загальної громіздкості та складності системи; технічна складність побудови чутливих датчиків Гейгера (недосконалість сучасних датчиків призводить до підвищення кореляції вихідної ВП); низька продуктивність; низька мобільність; висока собівартість. Часто, для отримання ВП, квантові генератори використовують «принцип Гейзенберга». Але є і інші підходи – наприклад, деякі генератори, використовують більш спрощені методи і дозволяють отримувати істинновипадкові послідовності біт із більшою швидкістю – до 100 кбіт на секунду. Висока статистична якість та мала кореляція вихідної ВП являються перевагами квантових ГВП. Недоліками є: відносно низька продуктивність; потреба у складних, високотехнологічних оптико-електронних приладах. В наслідок цього – низька загальна надійність і висока собівартість.

Найдоступніший та найбільш простий на сьогодні спосіб отримання первинної ентропії для генерації ВП є перетворення випадкових процесів електричних шумів. Природа виникнення електричних шумів може бути обумовлена самим механізмом протікання електричного струму. Та флуктуаціями інших неелектричних величин, які перетворюються у флуктуації токів та напруг. Причини виникнення ряду шумів до кінця нез'ясовані.

Фундаментальні аналогові електричні шуми, що використовуються для отримання ВП, можна поділити на тепловий шум (рівноважні флуктуації, шум Найквіста) та дробовий шум. Їх взаємодія породжує шуми підсилювачів, які теж можуть бути використані для генерації ІВП. Особливе місце займають «надлишкові шуми» (генераційно-рекомбінаційні шуми, шуми струморозподілення, вибухові та флікерні шуми). Їх існування взагалі не залежить від фундаментальних шумів і деякі з них можуть добре вплинути на отримання якісної статистики шумового процесу. Проте, наприклад, існування в системі флікерного шуму, який проявляє себе на інфранизьких частотах може

призвести до погіршення статистики ГВП та внести кореляцію у вихідну ВП. Описи ГВП (для криптографічних цілей):

- на основі термального шуму – «ZRANDOM» компанії Westphal Electronic

- дробового шуму – «HG400 Series Random Number Generators» компанії Random

- шумів підсилювача – «ComScire QNG» компанії The Quantum World Corporation

Перевагами ІВП, отриманих з використанням електричних шумів є: наявність глибоких теоретичних знань про механізми їх існування; простота реалізації (порівняно із іншими методами); висока продуктивність; висока мобільність та надійність; відносно низька собівартість. Головні недоліки: сильний вплив схемотехнічних, технологічних рішень та зовнішнього середовища на статистичну якість та кореляцію вихідної ВП. Окремо є можливість виділити метод, який, завдяки сучасним нанотехнологіям, використовує переваги ГВП, що побудовані на процесах електричного шуму та квантової механіки.

Запропонований компанією El-Mul Technologies Ltd. новітній та перспективний спосіб генерації ІВП отримує початкову ентропію з випадкового процесу дробового шуму польового транзистору, побудованого на карбонових нанотрубках. Його сполучення з наноскопічними лічильниками дозволяє збирати інформацію з точністю до окремих електронів та забезпечити продуктивність до 1 Гбіт на секунду. Окремий дискретний елемент з десятків тисяч таких ГВП може дати продуктивність до 10 терабіт на секунду. Таке рішення дозволяє зменшити вплив зовнішнього середовища на статистику та кореляцію ГВП. Недоліки – теоретично можливий вплив зовнішнього середовища на процес генерації ВП, складність та висока собівартість технологічного процесу виготовлення таких ГВП.

Генератори ІВП на сьогодні залишаються основним методом отримання первинної ентропії для потреб сучасних криптографічних систем. Їх перевага – можливість отримання універсальних та незміщених змінних , унеможливлення передбачення наступних випадкових елементів та повторення всієї ВП. Але такого результату можна досягти лише при спеціальних умовах використання ГВП із забезпеченням стабільних параметрів навколишнього середовища (температурні, тиску, електромагнітні, радіаційні тощо), спеціалізованих технічних, технологічних рішень та забезпеченню протидії атакам на ГВП.

Це, як правило, призводять до значного підвищення складності та собівартості генератора. Наявність будь-якого слабкого місця генератора ІВП, негайно призводить до втрати криптостійкості всієї КС – це головний недолік генераторів ІВП

1.3 Висновки до розділу 1

Криптографія – невідємна частина сучасного світу. Сукупністю криптографічних засобів захисту інформації є криптографічні системи, основним параметром яких є криптографічна стійкість. Для створення криптографічних систем використовується ряд різних алгоритмів та типів ключів для їх управління. Саме від вибору алгоритму і ключів буде залежати криптостійкість системи, що характеризує її захищеність та надійність.

При формуванні криптосисем визначну роль грають генератори випадкових чисел – саме вони формують шифри та їх алгоритми. Генератори поділяються на істинновипадкові та псевдовипадкові, кожний тип має свої переваги та недоліки.

РОЗДІЛ 2 АСИМЕТРИЧНІ АЛГОРИТМИ ШИФРУВАННЯ

2.1 Особливості асиметричних алгоритмів шифрування

В асиметричних криптоалгоритмах для шифрування повідомлення використовується один ключ, а для розшифровки інший. Ключ шифрування відомий усім, але перетворення є незворотним, тому зашифрований текст не може прочитати ніхто, крім отримувача – лише він один знає другий (закритий) ключ.

Кожен користувач асиметричної криптосистеми попередньо створює за певним алгоритмом пару ключів: закритий і відкритий – вони будуть у подальшому використовуватися для надсилання листів саме йому. Для відправлення листа іншому абоненту мережі необхідно буде скористатися саме його відкритим ключем.

“Симетричні криптосистеми, незважаючи на безліч переваг, володіють одним серйозним недоліком. Він пов’язаний із ситуацією, коли спілкуються не три-чотири людини, а сотні і тисячі. У цьому випадку для кожної пари, що листується, необхідно створювати свій секретний симетричний ключ. Це в підсумку призводить до існування в системі з N користувачів ключів. А це вже дуже “пристойне” число. Крім того, при порушенні конфіденційності будь-якої робочої станції, зловмисник отримує доступ до всіх ключів цього користувача і може відправляти, нібито від його імені, повідомлення всім абонентам, з якими “жертва” вела листування.

Своєрідним вирішенням цієї проблеми стала поява асиметричної криптографії. Ця область криптографії дуже молода в порівнянні з іншими представниками. Перша схема, що мала прикладну значимість, була запропонована всього близько 20 років тому. Але за цей час асиметрична криптографія перетворилася на один з основних напрямків криптології, і використовується в сучасному світі так само часто, як і симетричні схеми.

Асиметрична криптографія першопочатково була задумана як засіб передачі повідомлень від одного об’єкта до іншого (а не для конфіденційного збереження інформації, яку забезпечують тільки симетричні алгоритми). Тому подальше пояснення буде вестися в термінах “відправник” – особа, що шифрує, а потім відправляє інформацію по незахищеному каналу і “одержувач” – особа, яка приймає і відновлює інформацію в її початковому вигляді. Основна ідея асиметричних криптоалгоритмів полягає в тому, що для шифрування повідомлення використовується один ключ, а при дешифрування – інший (перша умова).

Крім того, процедура шифрування вибрана так, що вона незворотна навіть за відомим ключем шифрування – це друга необхідна умова асиметричної криптографії. Тобто, знаючи ключ шифрування і зашифрований текст, неможливо відновити вихідне повідомлення – прочитати його можна тільки за допомогою другого ключа – ключа дешифрування. А раз так, то ключ шифрування для відправки листів будь-якій особі можна взагалі не приховувати – знаючи його все одно неможливо прочитати зашифроване повідомлення. Тому, ключ шифрування називають в асиметричних системах “відкритим ключем”, а ось ключ дешифрування одержувачу повідомлень необхідно тримати в секреті – він називається “закритим ключем”. Третя необхідна умова асиметричної криптографії – алгоритми шифрування і дешифрування створюються так, щоб знаючи відкритий ключ, неможливо було вирахувати закритий ключ”[9].

У цілому система листування при використанні асиметричного шифрування виглядає наступним чином. “Для кожного з N абонентів, що ведуть листування, обрана своя пара ключів: “відкритий” і “закритий” , де j – номер абонента. Всі відкриті ключі відомі всім користувачам мережі, кожен закритий ключ, навпаки, зберігається тільки у того абонента, якому він належить. Якщо абонент, скажімо під номером 7, збирається передати

інформацію абоненту під номером 9, він шифрує дані ключем шифрування і відправляє її абоненту 9. Незважаючи на те, що всі користувачі мережі знають ключ і, можливо, мають доступ до каналу, по якому йде зашифроване послання, вони не можуть прочитати початковий текст, оскільки процедура шифрування незворотна по відкритому ключу. І тільки абонент №9, отримавши послання, здійснює над ним перетворення за допомогою відомого тільки йому ключа і відновлює текст послання. Якщо ж повідомлення потрібно відправити у протилежному напрямку (від абонента 9 до абонента 7), то потрібно буде використовувати вже іншу пару ключів (для шифрування ключ , а для дешифрування – ключ).

Отже, по-перше, в асиметричних системах кількість існуючих ключів пов’язана з кількістю абонентів лінійно (у системі з N користувачів використовуються ключів), а не квадратично, як у симетричних системах. По-друге, при порушенні конфіденційності k-ої робочої станції зловмисник дізнається лише ключ : це дозволяє йому читати всі листи, що надходять абоненту k, але не дозволяє видавати себе за нього при відправленні листів. Крім цього, асиметричні криптосистеми володіють ще декількома дуже цікавими можливостями, які буде розглянуто через кілька розділів”[9].

Асиметричні криптосистеми

Загальна схема асиметричної криптосистеми зображена на рисунку 2.1. За структурою вона практично ідентична симетричній криптосистемі з ключем сеансу.

Алгоритми з відкритим ключем розроблені таким чином, що ключ, використовуваний для шифрування, відрізняється від ключа розшифрування . Більш того, ключ розшифрування не може бути (принаймні протягом розумного інтервалу часу) розрахований по ключу шифрування.

Рисунок 2.1 – Загальна схема асиметричної криптосистеми

“Схематично шифрування та розшифрування при використанні асиметричних криптографічних систем зображені на рисунку 2.2.

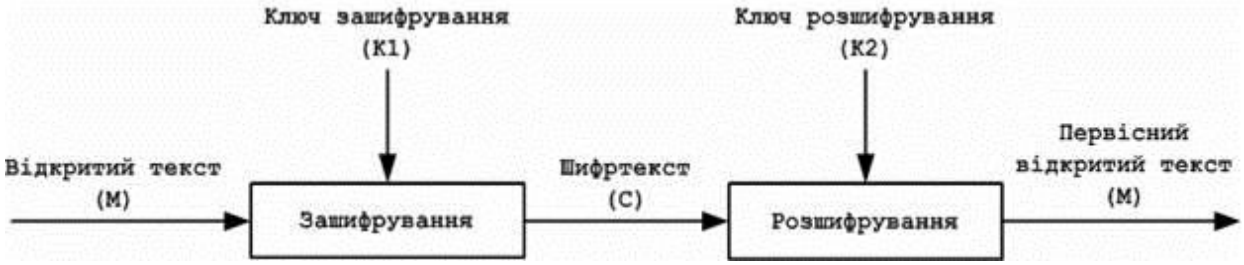

Рисунок 2.2 – Операції шифрування та розшифрування в асиметричних криптосистемах

Слід зауважити, що ключі К1 та К2 не обов’язково мають бути різними, але в тому випадку, коли вони співпадають, втрачається багато переваг алгоритму з відкритим ключем.

Від часу винайдення криптографії з відкритим ключем було запропоновано безліч криптографічних алгоритмів з відкритими ключами. Багато з них не є стійкими. А з тих, які є стійкими, багато непридатних для практичної реалізації . Або вони використовують дуже великий ключ, або розмір отриманого шифртекста набагато перевищує розмір відкритого тексту. Небагато алгоритмів є і безпечними, і практичними. Зазвичай ці алгоритми засновані на одній з важких проблем. Деякі з цих безпечних і практичних алгоритмів підходять тільки для розподілу ключів. Інші підходять для шифрування (і для розподілу ключів). Треті корисні тільки для цифрових підписів. Тільки три алгоритми добре працюють як при шифруванні, так і для цифрового підпису: RSA, EIGamal (Ель-Гамаля) та Rabin (Рабіна). Усі ці алгоритми повільні. Вони зашифровують і розшифровують дані набагато повільніше, ніж симетричні алгоритми. Зазвичай їх швидкість недостатня для шифрування великих обсягів даних”[9].

Гібридні криптосистеми дозволяють прискорити події: для шифрування повідомлення використовується симетричний алгоритм з випадковим ключем, а алгоритм з відкритим ключем застосовується для шифрування випадкового сеансового ключа.

Оскільки асиметричні криптоалгоритмами дуже повільні, в реальних системах використовуються швидкі надійні симетричні криптоалгоритмами по схемі з ключем сеансу. А от сам ключ сеансу кодується асиметричним криптоалгоритмами за допомогою відкритого ключа одержувача. Подібна система має всі властивості асиметричного криптоалгоритму і дуже високу швидкодію.

Порівняльна характеристика асиметричних та симетричних алгоритмів стосовно шифрування каналів зв’язку

Криптографія з відкритим ключем (асиметрична) та симетрична криптографія мають свої переваги та недоліки, але вони призначені для розв’язання проблем різних типів, і тому помилковими є спроби визначити який з цих двох видів криптографічних алгоритмів кращий або гірший.

Симетрична криптографія найкраще підходить для шифрування даних. Вона на декілька порядків швидша та стійкіша до розкриття з використанням підібраного шифртексту.

“Криптографія з відкритим ключем дозволяє робити речі, які недоступні симетричній криптографії; вона краще підходить для розподілення ключів та керування міріадами протоколів.

Розглядаючи обидва ці види криптографічних алгоритмів з позицій можливості застосування їх для апаратного шифрування каналів зв’язку, можна сказати, що симетричні криптографічні алгоритми підходять для цього найбільше, адже вони мають суттєві переваги:

1. позбавлені складних математичних обчислень – для шифрування використовуються прості логічні операції;
2. працюють на декілька порядків швидше;
3. більш стійкі до зламування.

Враховуючи те що каналами зв’язку пересилаються дуже великі об’єми даних (до того ж у більшості випадків послідовним кодом) на великих швидкостях (приблизно 100-1000 Мбіт/сек), а також те, що шифрування повинно відбуватися апаратно, криптографічний алгоритм повинен задовольняти наступним вимогам:

1. бути орієнтованим на апаратну реалізацію;
2. здійснювати шифрування даних на великих швидкостях без зайвих затримок;

3. бажано не використовувати буферизацію даних, через критичність затримок.

З огляду на вищевикладені вимоги, більш придатним для апаратного шифрування каналів зв’язку є потокові симетричні криптографічні алгоритми, які задовольняють усім вимогам щодо швидкості та стійкості, а також вільні від необхідності буферизації інформації, адже вони не є раундовимим”[9].

2.2 Алгоритм RSA. Цифрові підписи. Механізм розповсюдження відкритих ключів. Обмін по алгоритму Діффі-Хелмана

Алгоритм RSA

Алгоритми RSA є класикою асиметричної криптографії. У ньому в якості незворотного перетворення відправки використовується піднесення цілих чисел до великих ступенів за модулем.

Алгоритм RSA стоїть біля витоків асиметричної криптографії. Він був запропонований трьома дослідниками-математиками: Рональдом Рівестом (R. Rivest), Аді Шамір (A. Shamir) і Леонардом Адльманом (L. Adleman) в 1977-78 роках.

“Першим етапом будь-якого асиметричного алгоритму є створення пари ключів: відкритого та закритого та розповсюдження відкритого ключа “по всьому світу”. Для алгоритму RSA етап створення ключів складається з наступних операцій:

1. Вибираються два прості (!) числа p і q;
2. Обчислюється їх добуток n
3. Вибирається довільне число e , таке, що e повинно бути взаємно простим з числом n;

4. Методом Евкліда вирішується в цілих числах (!) рівняння. Тут невідомими є змінні d і y – метод Евкліда як раз і знаходить безліч пар (d, y), кожна з яких є рішенням рівняння в цілих числах.

5. Два числа (e, n) – публікуються як відкритий ключ.

6. Число d зберігається в найсуворішій таємниці – це і є закритий ключ, який дозволить читати всі послання, зашифровані за допомогою пари чисел (e, n)”[9].

Нижче наведено, як же здійснюється власне шифрування за допомогою цих чисел:

1. Відправник розбиває своє повідомлення на блоки, рівні біт, де квадратні дужки позначають взяття цілої частини від дробового числа.

2. Подібний блок може бути інтерпретований як число з діапазону . Для кожного такого числа обчислюється вираз. Блоки і є зашифрованим повідомленням. Їх можна спокійно передавати по відкритому каналу, оскільки операція піднесена до степеню по модулю простого числа, є незворотною математичною задачею. Обернена задача носить назву “логарифмування в кінцевому полі” і є на кілька порядків складнішим завданням. Тобто навіть якщо зловмисник знає числа e і n, то по прочитати вихідні повідомлення він не може ніяк, окрім як повним перебором .

“На приймальній стороні процес дешифрування все ж можливий, і допомогти у цьому може секретне число d. Досить давно була доведена теорема Ейлера, окремий випадок якої стверджує, що якщо число n, представлене у вигляді двох простих чисел p і q, то для будь-якого x має місце рівність . Для дешифрування RSA-повідомлень можна скористатись цією формулою. Звівши обидві її частини до степеню отримаємо: Тепер помноживши обидві її частини на x:

За допомогою алгоритму Евкліда підбирали таке d, що, тобто . А отже в останньому виразі попереднього абзацу можна замінити показник ступеня на

число. Отримуємо . Тобто, для того щоб прочитати повідомлення достатньо піднести його до степеню d по модулю m:

Операції піднесення до степеню великих чисел досить трудомісткі для сучасних процесорів, навіть якщо вони вираховуються за оптимізованими по часу алгоритмами. Тому звичайно весь текст повідомлення кодується звичайним блоковим шифром (набагато більш швидким), але з використанням ключа сеансу, а от сам ключ сеансу шифрується як раз асиметричним алгоритмом за допомогою відкритого ключа одержувача і поміщається в початок файлу”[9].

Технологія цифрових підписів

“Асиметрична криптографія, як виявилося, дозволяє дуже красиво вирішувати і завдання аутентифікації автора повідомлення – простим зміною порядку використання відкритого та закритого ключів.

Для вирішення цієї проблеми за допомогою симетричної криптографії була розроблена дуже трудомістка і складна схема. У той же час за допомогою, наприклад, того ж алгоритму RSA створити алгоритм перевірки справжності автора й незмінності повідомлення надзвичайно просто.

Припустимо, що потрібно передати будь-який текст, не обов’язково секретний, але важливо те, щоб у нього при передачі по незахищеному каналу не були внесені зміни. До таких текстів зазвичай відносяться різні розпорядження, довідки, і тому подібна документація, не представляє собою таємницю. Обчислимо від нашого тексту будь-яку хеш-функцію – це буде число, яке більш-менш унікально характеризує даний текст”[9].

Можна знайти інший текст, який дає те ж саме значення хеш-функції, але змінити в потрібному тексті десять-двадцять байт так, щоб текст залишився повністю осмисленим, та ще і змінився в вигідну користувачу сторону (наприклад, зменшив суму до оплати вдвічі) – надзвичайно складно. Саме для

усунення цієї можливості хеш-функції створюють такими ж складними як і криптоалгоритми – якщо текст з таким же значенням хеш-функції можна буде підібрати тільки методом повного перебору, а безліч значень буде становити як і для блокових шифрів 232-2128 можливих варіантів, то для пошуку подібного тексту зловмисникові “потрібні” ті ж самі мільйони років.

Якщо буде можливим передати одержувачеві захищеним від зміни методом хеш-суму від тексту, що пересилається, то у нього завжди буде можливість самостійно обчислити хеш-функцію від тексту вже на приймальній стороні і звірити її з присланою. Якщо хоча б один біт в обчисленій отримувачем самостійно контрольній сумі тексту не співпаде з відповідним бітом в отриманому хеш-значенні, значить, текст під час пересилки піддався несанкціонованій зміні.

“Представимо тепер готову до передачі хеш-суму у вигляді декількох k-бітних блоків, де k – це розмір повідомлень по алгоритму RSA в попередньому параграфі. Обчислимо для кожного блоку значення, де d – це той самий закритий ключ відправника. Тепер повідомлення, що складається з блоків можна “спокійно” передавати по мережі. Ніякої небезпеки знайти по відомих і секретний ключ немає – це настільки ж складне завдання, як і завдання “логарифмування в кінцевому полі”. А ось будь-який одержувач повідомлення може легко прочитати початкове значення , виконавши операцію Відкритий ключ відправника є у всіх, а те, що піднесення будь-якого числа до степеню по модулю n дає початкове число, було доведено в попередньому параграфі. При цьому ніхто інший, окрім користувача, не знаючи закритого ключа d не може, змінивши текст, а отже, і хеш-суму, обчислити такі , щоб при їх піднесенні до степеню e вийшла хеш-сума , що співпадала б з хеш-сумою фальсифікованого тексту.

Таким чином, маніпуляції з хеш-сумою тексту являють собою “асиметричне шифрування навпаки”: при відправці використовується закритий

ключ відправника, а для перевірки повідомлення – відкритий ключ відправника. Подібна технологія отримала назву “електронний підпис”. Інформацією, яка унікально ідентифікує відправника (його віртуальним підписом), є закритий ключ d. Жодна людина, що не володіє цією інформацією, не може створити таку пару (текст, підпис), щоб описаний вище алгоритм перевірки дав позитивний результат.

Подібний обмін місцями відкритого і закритого ключів для створення з процедури асиметричного шифрування алгоритму електронного підпису можливий тільки в тих системах, де виконується властивість комутативності ключів. Для інших асиметричних систем алгоритм електронного підпису або значно відрізняється від базового, або взагалі не реалізовується”[9].

Сертифікація електронних підписів

“Метою застосування систем цифрового підпису є автентифікація інформації – захист учасників інформаційного обміну від нав’язування хибної інформації, встановлення факту модифікації інформації, яка передається або зберігається, й отримання гарантії її справжності, а також вирішення питання про авторство повідомлень. Система цифрового підпису припускає, що кожен користувач мережі має свій таємний ключ, який використовується для формування підпису, а також відповідний цьому таємному ключу відкритий ключ, відомий решті користувачів мережі й призначений для перевірки підпису. Цифровий підпис обчислюється на основі таємного ключа відправника інформації й власне інформаційних бітів документу (файлу). Один з користувачів може бути обраним в якості “нотаріуса” й завіряти за допомогою свого таємного ключа будь-які документи. Решта користувачів можуть провести верифікацію його підпису, тобто пересвідчитися у справжності отриманого документу. Спосіб обчислення цифрового підпису такий, що знання відкритого ключа не може призвести до підробки підпису. Перевірити

підпис може будь-який користувач, що має відкритий ключ, в тому числі незалежний арбітр, якого уповноважено вирішувати можливі суперечки про авторство повідомлення (документу).

Простий аналіз інформаційних ризиків, які виникають при використанні несертифікованих засобів криптографічного захисту інформації (до яких, безумовно, відносяться засоби електронного цифрового підпису) без будь-яких обумовлень, вже дає достатньо підстав зробити вибір на користь застосування сертифікованих Департаментом спеціальних телекомунікаційних систем та захисту інформації Служби безпеки України засобів для здійснення електронних цифрових підписів(ЕЦП)" [9].

Власне ризик фальсифікації електронного підпису навряд чи можна вважати значним. По-перше, більшість несертифікованих засобів ЕЦП використовуються вже досить давно й на практиці довели свою надійність; по-друге, навіть для того, щоб фальсифікувати підпис, який накладається за допомогою будь-якого недосконалого алгоритму, можуть знадобитися досить значні витрати. Між тим є менш затратні способи компрометації системи, ніж спроби прямого злому.

"Перший ризик полягає в тому, що учасник системи документообігу може, посилаючись на те, що засоби ЕЦП не сертифіковані (а значить, не мають гарантій криптографічної стійкості), заявити, що його підпис було підроблено, й відкликати таким чином електронний документ із своїм підписом. Відзначимо, що для цього зовсім не потрібно, щоб підробка дійсно мала місце! Віртуальний, за великим рахунком, ризик фальсифікації ЕЦП породжує абсолютно реальний ризик відмови від ЕЦП.

З цією проблемою в системах документообігу, які використовують несертифіковані засоби, справляються єдиним чином: укладаючи додаткові угоди між учасниками документообігу, в яких сторони визнають даний засіб ЕЦП як достатній для забезпечення юридичної сили підписаних ЕЦП

документів. Підписуючи таку угоду, учасники документообігу фактично визнають, що даний засіб ЕЦП забезпечує високий рівень криптостійкості, підпис не може бути підробленим, а тому вони добровільно відмовляються від можливості надати рекламацію у зв’язку з фальсифікацією підпису. Проблема в тому, що підписання такої угоди вимагає достатньої сміливості - адже рядовий учасник документообігу навряд чи здатен самостійно провести експертизу й пересвідчитися в істинності того твердження, під яким він підписується”[9].

Інший ризик, що пов’язаний з можливістю відмовитися від авторства підписаного документу, ще більш серйозний. “Річ у тім, що несертифікований засіб ЕЦП ніхто не перевіряв, по-перше, з точки зору якості виконання основної функції (й на цьому засновано описаний вище ризик), а по-друге, з точки зору відсутності бічної дії. Автор завіреного ЕЦП електронного документу може в принципі спробувати відмовитися від змісту цього документу, стверджуючи, що засіб ЕЦП неадекватно перетворив запропонований йому файл: не тільки поставив ЕЦП, але й "випадково" щось ще змінив у файлі в силу помилки у програмі. Скоріше за все це твердження хибне. Досить малоймовірно, щоб випробуваний на практиці засіб ЕЦП дійсно дав такий збій. Ризик неадекватного перетворення вхідного файлу - чисто віртуальний. В той же час, заснований на ньому ризик відмови від змісту файлу цілком реальний. Автор стверджує, що програма дала збій й на виході отримано правильно завірений ЕЦП файл, відмінний від того, який було передано програмі на вхід.

За використання сертифікованих засобів криптографічного захисту інформації гарантом якості виконання основної функції й відсутності бічної дії виступає Департамент спеціальних телекомунікаційних систем захисту інформації Служби безпеки України. А при використанні несертифікованих засобів криптографічного захисту інформації таких гарантій не може дати ніхто”[9].

Згідно Закону України “Про електронний цифровий підпис”, послуги ЕЦП, в тому числі й видачу сертифікатів відкритих ключів, здійснює центр сертифікації ключів.

Центром сертифікації ключів (далі – ЦСК) може бути юридична чи фізична особа – суб’єкт підприємницької діяльності, який надає послуги в сфері ЕЦП й засвідчив свій відкритий ключ в органі технічного управління сфери ЕЦП – центральному засвідчувальному органі. В загальному вигляді визначається два типи надавачів послуг в сфері ЕЦП – центр сертифікації ключів й акредитований центр сертифікації ключів. Акредитованим ЦСК є центр сертифікації, який пройшов добровільну процедуру акредитації, яка підтверджує його здатність виконувати зобов’язання щодо обслуговування посилених сертифікатів. Додатково до обов’язків й вимог, встановленим Законом для ЦСК, акредитований ЦСК повинен використовувати для надання послуг в сфері цифрового підпису лише надійні засоби ЕЦП. Надійним вважається засіб ЕЦП, який має сертифікат відповідності або позитивний експертний висновок за результатами державної експертизи в сфері криптографічного захисту інформації.

“В загальному вигляді система ЕЦП функціонує наступним чином. Фізична або юридична особа, яка бажає стати учасником системи, в термінах Закону - підписувачем, звертається безпосередньо в ЦСК (акредитований ЦСК) або до його повноважного представника, який в ході процедури реєстрації з певним ступенем достовірності здійснює ідентифікацію заявника й отримуваних від нього даних, необхідних для формування сертифікату (посиленого сертифікату). ЦСК формує сертифікат відкритого ключа користувача й завіряє його своїм підписом. Новий сертифікат переміщується до бази даних дійсних сертифікатів ЦСК й стає доступним для всіх користувачів по загальнодоступним телекомунікаційним каналам.

Тепер разом з підписаним документом отримувачу може направлятися сертифікат відправника та/або, отримавши повідомлення, отримувач звертається до бази даних сертифікатів, по ідентифікаційним даним відправника отримує його сертифікат й перевіряє статус цього сертифікату (чинний, заблокований, скасований). Якщо сертифікат дійсний на момент перевірки ЕЦП з отриманого сертифікату вилучається відкритий ключ відправника й виконується перевірка його підпису. Слід відмітити, що клієнтське програмне забезпечення, яке забезпечує формування й перевірку ЕЦП, проводить ці операції автоматизовано й абсолютно прозоро для користувачів”[9].

Для вирішення проблем, пов’язаних з управлінням ключами в системах електронного документообігу з необмеженою кількістю учасників інформаційного обміну, необхідно створити спеціальну інфраструктуру підтримки управління ключами, так звану інфраструктуру відкритих ключів.

“В основі інфраструктури відкритих ключів (ІВК) лежить спеціальний суб’єкт – центр сертифікації ключів (ЦСК), основною метою якого є забезпечення безпечного обміну відкритими ключами між учасниками електронної взаємодії.

У відповідності до Закону України “Про електронний цифровий підпис”, ЦСК має право:

1. надавати послуги ЕЦП й обслуговувати посилені сертифікати ключів;

2. отримувати й перевіряти інформацію, необхідну для реєстрації користувача й формування сертифіката ключа безпосередньо в юридичної або фізичної особи або в їх представника.

Відповідно до Закону, ЦСК повинен:

1. вживати заходи для забезпечення безпеки інформації під час сертифікації ключів й зберігання сертифікатів ключів;

2. встановлювати під час формування сертифікату ключа, відповідності відкритого ключа й особистого ключа користувача;

3. забезпечувати захист персональних даних, отриманих від користувача, у відповідності до законодавства;

4. своєчасно скасовувати, блокувати й поновлювати сертифікати ключів у випадках, передбачених Законом;

5. перевіряти законність звернень про скасування й блокування сертифікатів ключів й зберігати документи, на основі яких були скасовані або блоковані сертифікати ключів;

6. цілодобово приймати заявки про скасування, блокування й поновлення сертифікатів ключів;

7. вести електронні реєстри чинних, скасованих й блокованих сертифікатів ключів й документацію, перелік якої визначається контролюючим органом;

8. забезпечення цілодобового доступу користувачів до сертифікатів ключів й відповідних електронних реєстрів через загальнодоступні телекомунікаційні канали;

9. забезпечувати зберігання сформованих сертифікатів ключів протягом термінів, передбачених законодавством для зберігання відповідних документів на паперовій основі;

10. надавати консультації з питань, пов’язаних з цифровим підписом”[9].

Центри сертифікації ключів (ЦСК) є єдиними суб’єктами системи ЕЦП, які надають послуги сертифікації відкритих ключів безпосередньо кінцевим користувачам.

“З технічної точки зору, функції, що їх виконує ЦСК, умовно можливо розділити на основні (функції управління сертифікатами) та додаткові.

До основних функцій відносяться:

1. генерація власної ключової пари;
2. реєстрація (ідентифікація) кінцевого користувача;
3. сертифікація відкритих ключів користувачів (процес формування сертифікатів відкритих ключів для кінцевих користувачів);
4. публікація (розповсюдження) сертифікатів у відкритому каталозі для забезпечення доступу до них кінцевих користувачів;
5. забезпечення відкликання сертифікатів (блокування або скасування дії сертифікату за умови виникнення певних обставин).
6. забезпечення перевірки легітимності сертифікату (розповсюдження списків відкликаних сертифікатів);
7. архівацію сертифікатів;
8. До додаткових функцій можливо віднести:
9. -генерація ключів користувачам;
10. забезпечення підтримки неможливості відмови від ЕЦП;
11. управління “історією” сертифікату;
12. часові штампи;
13. нотаріальне засвідчення;
14. розбір конфліктних ситуацій”[9].

Відповідальність за зберігання особистого ключа у таємниці покладається на його власника. Передача ключа іншим особам є фактом компрометації ключа – в такому випадку його власник не може контролювати особистий ключ, хоча несе відповідальність за його застосування. Для випадків, коли власник ключа відсутній з поважної причини, слід використовувати ЕЦП іншої особи, яка має свій власний ключ.

“Слід розуміти, що визначення Законом України “Про електронний цифровий підпис” терміну “підписувач” як особи, яка на законних підставах володіє особистим ключем та від свого імені або за дорученням особи, яку вона

представляє, накладає електронний цифровий підпис під час створення електронного документа, не припускає передачі особистого ключа іншим особам. Поняття “за дорученням” слід розуміти як передачу повноважень, а не передачу власного підпису. Якщо провести аналогію до паперового документообігу, неможна доручити будь-кому накладати власноручний підпис іншої особи. Наприклад, коли посадова особа перебуває у відпустці, вона доручає іншій особі накладати підпис на документах, але накладати підпис саме особи, яка заміщує.

Таким чином, для передачі повноважень накладання електронного цифрового підпису до іншої особи, вона повинна мати свій особистий ключ електронного цифрового підпису, засвідчений встановленим порядком”[9].

Механізм розповсюдження відкритих ключів

Асиметрична криптографія зробила ще і достатньо могутній прорив в технології первинного розповсюдження ключів. Якщо для симетричних криптосистем обов’язковим був попередній обмін по закритому каналу (зазвичай особисто з рук в руки), то тепер з’явилися абсолютно нові способи для цього.

Асиметричні криптосистеми позбавлені одного з найголовніших недоліків симетричних алгоритмів – необхідності попереднього обміну сторонами секретним ключем по захищеній схемі. Начебто досить “розтрубити” по всьому світу про свій відкритий ключ, і ось готова надійна лінія передачі повідомлень.

“Для того, щоб відправити зашифроване повідомлення, відправник повинен дізнатися відкритий ключ отримувача. Якщо ключ не приносили особисто на дискеті, то це означає, що відправник його просто узяв з інформаційної мережі. Зловмисник може згенерувати довільну пару (закритий ключ, відкритий ключ), потім активно поширювати або пасивно підміняти при

запиті бажаний відкритий ключ створеним ним. В цьому випадку при відправці повідомлення:

1. Відправник зашифрує його тим ключем, що отримав від зловмисника;

2. Зловмисник, перехопивши повідомлення дешифрує його парним закритим ключем, прочитає;

3. Може переслати далі, зашифрувавши вже відкритим ключем відправника. Так само, але по інверсній схемі, він може підмінити і електронний підпис вже відправника під його ж листом”[9].

Якщо між відправником і одержувачем немає конфіденційної схеми передачі асиметричних ключів, то виникає серйозна небезпека появи зловмисника-посередника. Але асиметрична криптографія знайшла витончений спосіб значно знизити ризик подібної атаки. Якщо задуматися, то неправильно говорити, що між відправником та отримувачем немає гарантованої лінії зв’язку. Поза сумнівом у всіх знайдеться троє-четверо надійних знайомих в столиці або за кордоном, у них у свою чергу також знайдеться безліч знайомих в багатьох точках країни і світу. Врешті-решт, люди користуються програмним забезпеченням фірм, якщо не центри, то хоч би філії яких знаходяться в тій країні або в тому місті, куди відправник хоче відправити лист. Проблема тільки в тому, що починаючи, з другої від відправника ланки ні він не знає людину, ні вона його, і вірогідність того, що ця людина, або більш того, крупна компанія, що-небудь робитимуть для відправника, дуже мала.

Якщо безліч однодумців об’єднаються з метою створити надійну мережу розповсюдження ключів, то це буде їм цілком під силу. А сама асиметрична криптографія допоможе їм в цьому таким чином: насправді нікуди ходити з дискетою, отримавши прохання від свого знайомого передати відкритий ключ містера V.M.B. містерові R.H.J., не потрібно. Адже співбесідник знає відкритий ключ іншого, отриманий яким-небудь надійним способом. А отже, він може

інший співбесідник може прислати цей відкритий ключ містера V.M.B., підписавши повідомлення своїм електронним підписом. Наступному у свою чергу потрібно всього лише відправити цей ключ далі по ланцюжку у напрямі містера R.H.J., підписавши вже своїм електронним підписом. Таким чином, минувши декілька перепідписів, відкритий ключ дійде від місця відправлення до місця вимоги по надійному шляху. В принципі від користувача може навіть не вимагатися ніяких дій – просто потрібно поставити на ПК спеціальний сервер розповсюдження ключів, і він всі тільки що описані дії виконуватиме автоматично.

“На сьогоднішній день не існує єдиної мережі розповсюдження відкритих ключів, і справа, як це часто буває, полягає у війні стандартів. Розвиваються декілька незалежних систем, але жодна з них не отримала достатньої переваги над іншими, щоб назватися “світовим стандартом”.

Необхідно відзначити, що ланцюжок розповсюдження ключів в реальних випадках не дуже великий. Зазвичай він складається з двох-чотирьох ланок. Із залученням до процесу розповсюдження ключів крупних фірм-виробників програмних продуктів він стає ще коротшим. Дійсно, якщо на компакт-диску (не піратському!) з купленим програмним забезпеченням вже знаходиться відкритий ключ цієї фірми, а сама вона має крупний ринок збуту, то ланцюжок складатиметься або з однієї ланки (якщо ПЗ ця ж фірма стоїть і у потенційного отримувача), або з двох (другим стане який-небудь інший гігантський концерн, чиє ПЗ встановлено у співбесідника –між собою всі крупні компанії обмінялися ключами електронних підписів досить давно). Відкритий ключ, підписаний якою-небудь третьою стороною, називається завіреним за допомогою сертифікату. Сертифікатом називається інформаційний пакет, що містить який-небудь об’єкт (зазвичай ключ) і електронний підпис, підтверджуючий цей об’єкт від імені чиєї-небудь особи”[9].

Обмін по алгоритму Діффі-Хелмана

“Метод Діффі-Хелмана використовує алгоритм, подібний до алгоритму RSA, для первинного обміну ключами в симетричних криптосистемах по відкритому каналу, але тільки такому, в якому неможлива фальсифікація повідомлень.

Це цікавий алгоритм, який достатньо важко класифікувати. Він допомагає обмінюватися секретним ключем для симетричних криптосистем, але використовує метод, дуже схожий на асиметричний алгоритм RSA. Алгоритм названий по прізвищах його творців Діффі (Diffie) і Хелмана (Hellman)”[9].

Визначимо круг його можливостей. Нехай, двом абонентам необхідно провести конфіденційне листування, а в їх розпорядженні немає початково обумовленого секретного ключа. Проте, між ними існує канал, захищений від модифікації, тобто дані, що передаються по ньому, можуть прослуховувати, але не змінюватись (такі умови бувають досить часто). В цьому випадку дві сторони можуть створити однаковий секретний ключ, жодного разу не передавши його по мережі, по наступному алгоритму.

“Нехай, обом абонентам відомо деякі два числа v і n. Вони, втім, відомі і всім іншим зацікавленим особам. Наприклад, вони можуть бути просто фіксовано “зашиті” в програмне забезпечення. Для того, щоб створити нікому більше невідомий секретний ключ, обидва абонента генерують випадкові або псевдовипадкові прості числа: перший абонент – число x, другий абонент – число y. Потім перший абонент обчислює значення і пересилає його другому, а другий обчислює і передає першому. Зловмисник отримує обидва цих значення, але модифікувати їх (втрутитися в процес передачі) не може. На другому етапі перший абонент на основі того, що є у нього x і отримане по мережі значення обчислює значення , а другий абонент на основі того, що є у нього y і отриманого по мережі обчислює значення. Насправді операція піднесення до степеня переноситься через операцію взяття модуля по простому

числу (тобто комутативна в кінцевому полі), тобто у обох абонентів вийшло одне і те ж число. Його вони і можуть використовувати як секретний ключ, оскільки тут зловмисник знову зустрінеться з проблемою RSA при спробі з'ясувати по перехоплених і самі числа x і y – ця операція потребує дуже багато ресурсів, якщо числа v, n, x, y вибрані достатньо великими.

Необхідно відзначити, що алгоритм Діффі-Хелмана працює тільки на лініях зв'язку, надійно захищених від модифікації. Якби він був застосовний на будь-яких відкритих каналах, то давно зняв би проблему розповсюдження ключів і, можливо, замінив собою всю асиметричну криптографію. Проте, в тих випадках, коли в каналі можлива модифікація даних, з'являється очевидна можливість уклинення в процес генерації ключів "зловмисника-посередника" по тій же самій схемі, що і для асиметричної криптографії"[9].

2.3 Висновки до розділу 2

Асиметричні криптоалгоритми для шифрування працюють на основі пари з двох різних ключів – один для шифрування, а інший для розшифрування. Ключ шифрування відомий усім (відкритий), але перетворення є незворотним, тому зашифрований текст не може прочитати ніхто, крім отримувача – лише він один знає інший (закритий) ключ. Значною перевагою асиметричних алгоритмів шифрування є те, що їх використання дозволяє забезпечити безпечний спосіб зв'язку для великих(сотні і тисячі) груп людей.

Найвідомішим, класичним, алгоритмом асиметричного шифрування є алгоритм RSA, сутність якого полягає у шифруванні окремих блоків тексту закритим ключем і подальшому розшифруванні на стороні приймача за допомогою закритого ключа.

Крім того, асиметричні алгоритми шифрування активно використовуються в цифрових підписах, як показник того, що передані данні не були змінені жодним чином.

РОЗДІЛ 3 КРИПТОГРАФІЯ НА ЕЛІПТИЧНИХ КРИВИХ

3.1 Принципи еліптичної криптографії

Еліптична криптографія – це такий розділ криптографії, що використовує еліптичні криві з різними параметрами, визначеними над скінченними полями для реалізації схем шифрування. Головний напрямок застосування еліптичних кривих в криптографічних схемах це системи з відкритим ключем. Ключовим математичним об'єктом еліптичної криптографії є еліптична крива.

Еліптичною кривою E [10], визначеною над скінченним полем K, називається крива, описана рівнянням Вейєрштраса (1):

$$y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6 \qquad (1)$$

де $\{a_1, a_2, a_3, a_4, a_5, a_6\} \in K,\ \Delta \neq 0$.

Δ називається дискримінантом E і визначається як:

$$\Delta = - d_2^2 d_8 - 8 d_4^3 - 27 d_6^2 + 9 d_2 d_4 d_6$$

$$d_2 = a_1^2 + 4a_2$$

$$d_4 = 2a_4 + a_1 a_3$$

$$d_6 = a_3^2 + 4a_6$$

$$d_8 = a_1^2 a_6 + 4a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2$$

Якщо скінченне поле K є простим – $GF(p)$, E трансформується у криву, що описується таким рівнянням:

$$y^2 = x^3 + ax^2 + b$$

У випадку $K = GF(2^m)$ E трансформується у криву, описану рівнянням:

$$2 + xy = x^3 + ax^2 + b$$

де $a, b \in K$

Розглянемо арифметику над точками еліптичної кривої.

Нехай E – еліптична крива, що визначена над полем K та $P, Q \in E$ - точки еліптичної кривої. Сума точок P та Q графічно визначається наступним способом.

1. Провести пряму через точки P та Q.
2. Відобразити перетин даної прямої з еліптичною кривою відносно осі OY.

Подвоєння точки P графічно визначається наступним чином.

1. Провести дотичну до еліптичної кривої в точці P.
2. Перетин даної дотичної, відображений симетрично осі OY, називається подвоєною точкою P.

Вимоги до еліптичної кривої

Для того, щоб уникнути відомих атак, що засновані на проблемі дискретного логарифма у групі точок ЕК, необхідно, щоб кількість точок ЕК ділилась на достатньо велике просте число n. Стандарт ANSI X9.62 вимагає $n > 2^{160}$. Рівняння ЕК будується специфічним способом, використовуючи випадкові/псевдовипадкові коефіцієнти.

Основними параметрами при побудові ЕК над полем $GF(p)$ є:

1. Розмірність поля p, де p є простим числом.
2. Два елементи скінченного поля a та b, визначені рівнянням еліптичної кривої E, що має наступний вигляд:

$$y^2 = x^3 + ax + b$$

де $a,b \in GF(p)$ та $4a^3 + 27b^2 \neq 0 (\mathrm{mod}\ p)$.

3. Два елементи поля $GF(p) - x_G$ та y_G, які визначають кінцеву точку

$G=(x_G, y_G)$ – генератор групи.

4. Порядок точки G, де $q > 2^{160}$ та $q > 4\sqrt{p}$.

5. Співмножник $h = \#E / q$, де $\#E$ означає порядок групи точок ЕК.

Рисунок 3.1 Еліптичні криві (b=1. a змінюється в межах [-3;2])

Генерація основних параметрів

Один з основних способів генерації криптографічно надійних параметрів полягає в наступному:

1. Обираємо коефіцієнти a та b специфічним чином, використовуючи в обчисленнях випадкові/псевдовипадкові числа. Нехай E – еліптична крива $y^2 = x^3 + ax + b$[12].

2. Обчислюємо $N = \#E$

3. Перевіряємо, що N має дільник, який є великим простим числом $q (q > 2^{160}, q > 4\sqrt{p})$. Якщо не виконується, то необхідно повернутись на крок 1.

4. Перевіряємо, що q не ділить $p^k - 1$ для кожного k, $1 \le k \le 100$. Якщо не виконується, то необхідно повернутись на крок 1

5. Перевірити, що $q \neq p$. Якщо не виконується, то необхідно повернутись на крок 1

6. Обрати випадкову точку $G' \in E$ і покласти $G = (N / q)G'$. Повторювати, доки $G \neq O$.

3.2 Методи шифрування в еліптичній криптографії

Найпопулярнішими напрямками еліптичної криптографії, тобто сферами, в яких криптостійкість базується на задачі дискретного логарифмування для еліптичних кривих або ECDLP, є шифрування з відкритим ключем та алгоритм електронно-цифрового підпису.

В цьому підрозділі буде розглянуто різноманітні схеми обміну ключами. Існують два види використання шифрування з ключами – симетричний (існує єдиний секретний ключ, що має бути переданий між відправником та отримувачем захищеним каналом) та асиметричний (існує пара ключів – приватний та публічний, що може бути переданий незахищеним каналом). В межах даної роботи детально розглядається другий варіант, а саме асиметричне шифрування.

Шифрування з відкритим ключем

Еліптичний варіант схеми обміну ключами Діффі-Хелмана

“Протокол Діффі-Хелмана – це метод обміну криптографічними ключами. Даний метод дозволяє двом учасникам, що не мають жодних

попередніх даних один про одного, отримати спільний секретний ключ, що використовуватиметься для шифрування даних, якими обмінюються сторони, за допомогою незахищеного каналу зв'язку. Цей ключ можна використати для шифрування наступних сеансів зв'язку, що використовують шифр з симетричним ключем”[20].

Нехай існує еліптична крива, що забезпечує достатню криптостійкість (несуперсингулярну, в якій генеруюча точка $G = (x; y)$ має великий порядок, тобто число n, при якому $nG = O$ є дуже великим простим числом), визначена параметрами a та b:

$$y^2 = x^3 + ax + b$$

Позначимо сторону відправника як A, сторону отримувача як B, тоді обмін ключами між сторонами A та B проводиться таким чином:

1. Сторона A обирає ціле число $Priv_A < n$. Дане число називається приватним ключем учасника, а точка еліптичної кривої $Pub_A = Priv_A \times G$ називається публічним ключем.

2. Сторона B обирає аналогічно секретний ключ $Priv_B$ та обчислює відкритий ключ $Pub_B = Priv_B \times G$.

3. Учасник A генерує секретний ключ $K = k_A \times P_B$, а учасник B генерує секретний ключ $K = k_B \times P_A$.

Дві формули, отримані в третьому пункті дають один й той самий результат, оскільки:

$$k_A \times P_B = k_A \times (k_B \times G) = k_B \times (k_A \times G) = k_B \times P_A.$$

Проблема, яка стоїть перед сторонніми спостерігачами, які хочуть дізнатись секретний ключ, полягає в обчисленні $k_A \times k_B \times G$ за відомими G, k_A

$\times G$, $k_B \times G$, але не знаючи k_A та k_B. Це і є проблемою Діффі-Хеллмана для еліптичних кривих.

Протокол Мессі-Омура(Massey-Omura)

“Криптосистема Мессі-Омура [13] була запропонована в 1978 році Джеймсом Мессі і Джимом К. Омура в якості поліпшення протоколу Шаміра. Є два варіанти реалізації даного протоколу: класичний і еліптичний. Перший побудований на складності завдання дискретного логарифмування, другий на властивостях еліптичної кривої”[21].

“Еліптичний варіант даного протоколу надає можливість передавати повідомлення від відправника A до отримувача B по відкритому каналу. Нехай порядок еліптичної кривої дорівнює N, e – ціле число, взаємно просте з N. За алгоритмом Евкліда можна знайти:

$$d \equiv e^{-1} \bmod N$$

За визначенням,

$$e \times d = j \times N + 1$$

Значить для будь-якої точки P еліптичної кривої порядку N виконується:

$$(e \times d) \times Q = (j \times N + 1)P = (j \times N) \times P + P = P + O = P$$

Тепер, використовуючи e і d і будь-яку точку P еліптичної кривої, можна обчислити:

$$Q = e \times P$$

$$P = d \times Q$$

Де $P = R$.

Обчислення точки P по $e \times P$ еквівалентно вирішенню завдання дискретного логарифма для еліптичної кривої”[21].

Розглянемо загальну схему роботи систем шифрування з відкритим ключем. Для ілюстрації таких систем розглянемо наступну ситуацію:

Необхідно забезпечити конфіденційну та захищену передачу деяких зашифрованих даних між відправником та отримувачем використовуючи незахищений канал для зв'язку, тобто такий, в якому, з ненульовою ймовірністю, існує сторона, яка здатна перехоплювати будь-які повідомлення, що передаються даною мережею. Позначимо сторону відправника як A, сторону отримувача як B, повідомлення, що передається як m. Для організації безпечного обміну повідомленнями між A та B використовується наступна методика:

1. Сторони A та B генерують пари ключів $(Pub_A, Priv_A)$ для сторони A та $(Pub_B, Priv_B)$ для сторони B, де $(Pub, Priv)$ – публічний та приватний ключі

2. Сторона B надсилає незахищеним каналом стороні A свій публічний ключ - Pub_B.

3. Сторона A виконує шифрування повідомлення m за допомогою деякої функції Enc, що приймає аргументи Pub та m. Результатом виконання даної функції є деяке значення $e = Enc(Pub_B, m)$.

4. Сторона A надсилає стороні B незахищеним каналом зашифроване повідомлення e.

5. Сторона B отримує зашифроване повідомлення e, та за допомогою деякої функції Dec, що приймає аргументи $Priv$ та e, отримує вихідне повідомлення $m = Dec(Priv_B, e)$.

Для зворотного зв'язку виконуються аналогічні дії, де відправником є сторона B, отримувачем – сторона A, і замість $(Pub_B, Priv_B)$ виконуються маніпуляції з парою ключів $(Pub_A, Priv_A)$.

Опис алгоритму електронно-цифрового підпису

Ще однією сферою, де активно використовується еліптична криптографія, є алгоритм електронно-цифрового підпису[14]. Розглянемо детальніше цей алгоритм, запропонувавши наступний приклад. Нехай існує деякий відправник повідомлення, оригінальність та незмінність якого після передачі незахищеним каналом повинна бути гарантованою, та отримувач, який повинен мати можливість переконатись у оригінальності та незмінності отриманого повідомлення. Позначимо відправника як A, отримувача як B, повідомлення як m. Для виконання алгоритму електронно-цифрового підпису необхідно виконати наступні дії:

1. Сторона A обирає таке значення $Priv_A$, що є числом великої розрядності. Обчислюється точка еліптичної кривої $Pub_A = Priv_A \times G$, де G – генеруюча точка еліптичної кривої, тобто точка з великим порядком. Точка Pub_A називається електронно-цифровим підписом відправника A.

2. Обчислюється хеш-сума повідомлення використовуючи деяку функцію $Hash$, $h = Hash(m)$.

3. Використовуючи електронно-цифровий підпис Pub_A, виконується підпис хеш-суми повідомлення, використовуючи деяку функцію $Sign$, що приймає в якості параметрів хеш та електронно-цифровий підпис $signature = Sign(h, Pub_A)$.

4. Відправнику B надсилається $(m, signatre, Pub_A)$.

Для перевірки електронно-цифрового підпису на стороні отримувача виконуються наступне:

1. Обчислюється хеш повідомлення $h = Hash(m)$.

2. Використовується деяка функція $Verify$, що приймає параметрами хеш-суму h, підпис повідомлення $signature$, електронно-цифровий підпис Pub_A,

та повертає булеве значення $verified = Verify(h, signature, Pub_A)$.

3. Якщо значення $verified = true$, то повідомлення є оригінальним та не було змінене в процесі передачі незахищеним каналом.

3.3 Алгоритми генерування та перевірки цифрового підпису

Еліптична криптографія також знайшла своє застосування і в алгоритмах електронно-цифрового підпису, а саме в алгоритмі ECDSA (elliptic curve digital signature algorithm). Розглянемо схеми генерування та перевірки електронно-цифрового підпису.

3.3.1 Схема цифрового підпису Єль-Гамаля

“Схема Ель-Гамаля(El-Gamal)[15] – схема шифрування з відкритим ключем, заснована на складності обчислення дискретних логарифмів. Дана схема може використовуватись як для шифрування так і як алгоритм електронно-цифрового підпису. Схема була запропонована Тахером Ель-Гамалем в 1985 році. Він удосконалив систему Діффі-Хеллмана і отримав два алгоритми, які призначені для шифрування і для автентифікації.

При використанні даної схеми в режимі електронно-цифрового підпису, одержувач підписаного повідомлення може використовувати цифровий підпис для перевірки незмінності та оригінальності повідомлення підписаного відправником. Для перевірки електронно-цифрового підпису необхідно використовувати деяку надійну хеш-функцію. Розглянемо узагальнену схему електронного підпису Ель-Гамаля”[22].

Генерація ключів:

1. Генерується випадкове просте число довжиною p бітів.

2. Вибирається випадковий примітивний елемент $g \in Z_p$.

3. Вибирається випадкове ціле x таке що $1 < x < p-1$.

4. Обчислюється $y = g^x \bmod p$.

5. Відкритим ключем є трійка (p, g, y), а закритим число x.

Алгоритм підпису повідомлення M:

1. Обчислити хеш повідомлення: $m = h(M)$.

2. Вибрати випадкове число $1 < k < p-1$ взаємно просте з $p-1$ і обчислюється $r = g^k \bmod p$.

3. Обчислити число $s \equiv (m - xr)k^{-1} \bmod p-1$.

4. Підписом повідомлення M є пара (r, s).

Перевірка підпису:

1. Перевіряється справедливість умов: $0 < r < p$ і $0 < s < p-1$. Якщо хоча б одна з них не виконується, то підпис вважається невірним.

2. Обчислюється хеш повідомлення $m = h(M)$.

3. Підпис вважається вірним, якщо виконується порівняння:

$$y^r r^s \equiv g^m \bmod p$$

3.3.2 Алгоритм ECDSA

“ECDSA (Elliptic Curve Digital Signature Algorithm) – алгоритм з відкритим ключем, що використовується для створення цифрового підпису, аналогічний, за будовою до DSA, але визначений, на відміну від нього, не над полем цілих чисел, а у групі точок ЕК. Розглянемо використання алгоритму ECDSA з використанням еліптичної кривої з параметрами,

визначеними над полем $GF(p)$.

Генерація ключів ECDSA

Нехай E – еліптична крива (ЕК), визначена над полем $GF(p)$ і P – точка високого порядку q кривої E. Відправник A конструює свій ключ, виконуючи наступні кроки:

1. Обирає випадкове ціле число x з інтервалу $[1, q-1]$.
2. Обчислює добуток $Q = x \cdot P$.

Відкритим ключем відправника A є точка Q, а закритим – x'' [23].

Обчислення ключів ECDSA

Для того, щоб підписати деяке повідомлення m відправник A повинен зробити наступне[16]:

1. Обрати випадкове ціле число $k \in [1, q-1]$.
2. Обчислити $k \cdot P = (x_1, y_1)$ та $r = x_1 \bmod q$. Якщо $r \equiv 0 (\bmod q)$, то обирають нове випадкове число $k \in [1, q-1]$.
3. Обчислити $k^{-1} (\bmod q)$ та $s = k^{-1}(h + x \cdot r) \bmod q$, де h – значення хеш-функції повідомлення, що підписується. Якщо $s = 0$, то значення $s^{-1} \bmod q$ не існує, тому необхідно повернутись до п.1.

Підписом для повідомлення є пара цілих чисел (r, s).

Перевірка ключів ECDSA

Для того, щоб перевірити підпис відправника A на повідомлення, отримувач B повинен зробити наступне:

1. Отримати копію відкритого ключа Q відправника A.
2. Перевірити, що числа r та s є цілими числами з інтервалу $[1, q-1]$ та обчислити значення хеш-функції h від повідомлення.
3. Обчислити $u_1 = s^{-1} h \bmod q$ та $u_2 = s^{-1} r \bmod q$.

4. Обчислити $u_1 \cdot P + u_2 \cdot Q = (x_0, y_0)$ та $v = x_0 \bmod q$.

5. Прийняти підпис, якщо $v = r$.

Можна легко показати, що даний алгоритм дійсно успішно засвідчує цифровий підпис. Якщо підпис (r, s) повідомлення m було дійсно згенеровано з використанням секретного ключа, то $s \equiv k^{-1}(h + xr)(\bmod q)$.

Розкриваючи дужки отримаємо:

$k \equiv s^{-1}(h + xr) \equiv s^{-1}h + s^{-1}rx = u_1 + u_2 x (\bmod q)$.

Тоді $u_1 \cdot P + u_2 \cdot Q = (u_1 + u_2 x) \cdot P = kP$ і значить $v = r$, що і вимагається для підтвердження підпису.

3.4 Висновки до розділу 3

Еліптична криптографія – це такий розділ криптографії, що використовує еліптичні криві з різними параметрами, визначеними над скінченними полями для реалізації схем шифрування. Ключовим математичним об'єктом еліптичної криптографії є еліптична крива.

Загальний вигляд еліптичних кривих, що використовуються при шифруванні: $y^2 = x^3 + ax + b$. На основі цих кривих розраховуються додаткові параметри шифрування, а саме: розмірність поля P, елементі поля x_G та y_G які формують точку G на кривій, порядок точки G на кривій n та порядок групи точок кривої h.

В цьому розділі було розглянуто такі методи асиметричного шифрування на основі еліптичних кривих як метод Діффі-Хілмана, метод Мессі-Омура. Крім того, було описано алгоритм генерації та перевірки електронно-цифрових підписів за допомогою схем Ель-Гамаля та ECDSE.

РОЗДІЛ 4 РЕАЛІЗАЦІЯ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ НА ЕЛІПТИЧНИХ КРИВИХ

4.1 Вибір мови програмування

“Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня з строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. В мові програмування Python підтримується такі парадигми програмування, як: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Основними її перевагами є:

1. Чистий та інтуїтивно зрозумілий синтаксис.

2. Портабельність програм пов’язана з інтерпретативнісю.

3. Наявність вбудованих модулів, шо надають можливість використовувати колекції, особливі структури даних, математичний пакет тощо.

4. Можливість програмування на Python в скрипковому режимі (з консолі).

5. Зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини).

6. Політика відкритого коду (open source).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ. Інтерпретатор мови Python і Стандартна бібліотека можуть вільно розповсюджуватись та використовуватись.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C або на іншій мові, яку можна викликати із C. Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження” [25].

Крім того, Python було обрано як популярну мову з великою кількістю джерел до яких можна звернутися в разі необхідності та вже наявних знань у цій мові програмування.

4.2 Генерація пари відкритих і закритих ключів

Реалізація алгоритму генерації відкритих і закритих ключів на певній еліптичній кривій потребує знати ряд параметрів еліптичної кривої, її площини, групи та підгрупи точок на ній.

Для полегшення даної задачі було взято параметри стандартизованої еліптичної кривої secp256k1, запропонованої групою SECG (Standards for Efficient Cryptography Group), а саме:

p = 0xffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffffe fffffc2f

a = 0

b = 7

x_G = 0x79be667e f9dcbbac 55a06295 ce870b07 029bfcdb 2dce28d9 59f2815b 16f81798

y_G = 0x483ada77 26a3c465 5da4fbfc 0e1108a8 fd17b448 a6855419 9c47d08f fb10d4b8

n = 0xffffffff ffffffff ffffffff fffffffe baaedce6 af48a03b bfd25e8c d0364141

h = 1 [24, ст. 9]

Написаний скрипт дозволяє вільно змінити вхідні дані на параметри інших кривих, однак рекомендується використовувати відомі параметри

стандартизованих кривих.

Скрипт було написано за допомогою мови програмування Python на основі алгоритму шифрування ECDH (різновид алгоритму Діффі-Хілмана для еліптичних кривих), описаного в попередніх пунктах.

Даний скрипт повторює всю послідовність дій і розрахунків за допомогою відповідних команд, та на виході генерує пари відкритого і закритого ключів для двох користувачів.

```
Curve: secp256k1
Alice's private key:
0xe32868331fa8ef0138de0de85478346aec5e3912b6029ae71691c384237a3eeb
Alice's public key:
(0x86b1aa5120f079594348c67647679e7ac4c365b2c01330db782b0ba611c1d677,
0x5f4376a23eed633657a90f385ba21068ed7e29859a7fab09e953cc5b3e89beba)
Bob's private key:
0xcef147652aa90162e1fff9cf07f2605ea05529ca215a04350a98ecc24aa34342
Bob's public key:
(0x4034127647bb7fdab7f1526c7d10be8b28174e2bba35b06ffd8a26fc2c20134a,
0x9e773199edc1ea792b150270ea3317689286c9fe239dd5b9c5cfd9e81b4b632)
```

Повний текст скрипту надано у Додатку А.

4.3 Генерація та перевірка електронного підпису

Для написання цього скрипту також було обрано мову Python та ту ж криву, однак як і раніше він дозволяє ввести інші вхідні параметри.

Скрипт було реалізовано на основі алгоритму ECDSA повністю повторюючи всі необхідні кроки за допомогою програмних інструментів, якими може оперувати дана мова програмування.

На виході скрипт надає деяке повідомлення, підписане згенерованим ключем та виконує перевірку спочатку самого повідомлення, а потім підпису

для його ідентифікації. Результат:

```
Curve: secp256k1
Private key:
0x9f4c9eb899bd86e0e83ecca659602a15b2edb648e2ae4ee4a256b17bb29a1a1e
Public key:
(0xabd9791437093d377ca25ea974ddc099eafa3d97c7250d2ea32af6a1556f92a,
0x3fe60f6150b6d87ae8d64b78199b13f26977407c801f233288c97ddc4acca326)
Message: b'Hello!'
Signature:
(0xddcb8b5abfe46902f2ac54ab9cd5cf205e359c03fdf66ead1130826f79d45478,
0x551a5b2cd8465db43254df998ba577cb28e1ee73c5530430395e4fba96610151)
Verification: signature matches
Message: b'Hi there!'
Verification: invalid signature
Message: b'Hello!'
Public key:
(0xc40572bb38dec72b82b3efb1efc8552588b8774149a32e546fb703021cf3b78a,
0x8c6e5c5a9c1ea4cad778072fe955ed1c6a2a92f516f02cab57e0ba7d0765f8bb)
Verification: invalid signature
```

Повний текст скрипту надано у Додатку Б.

4.4 Висновки до розділу 4

У цьому розділі було програмно реалізовано алгоритм генерації відкритих і закритих ключів та алгоритм генерації і перевірки електронно-цифрових підписів за допомогою мови програмування Python.

Створені скрипти повністю відтворюють алгоритми описані вище за допомогою програмного коду та можливостей обраної мови програмування. В якості вхідних параметрів було обрано стандартизовану еліптичну криву secp256k1, а розмірність параметрів вказана у шістнадцятковій системі числення.

ВИСНОВКИ

В рамках даної роботи було проведено дослідження методів шифрування та дешифрування з використанням еліптичних кривих.

Було проведено загальний аналіз понять «криптографія» та «шифрування», на основі чого зроблено висновок стосовно важливості криптографії у сучасному світі та необхідності вдосконалення різних методів шифрування для забезпечення належного рівня захисту персональних даних користувачів в умовах сучасних тенденцій до швидкого зросту телекомунікаційних мереж і різних способів отримати несанкціонований доступ до чужих даних.

Було проаналізовано особливості сучасних асиметричних алгоритмів шифрування та проведено порівняння з основними особливостями симетричних алгоритмів шифрування. Провівши аналіз різних асиметричних алгоритмів шифрування було встановлено, що ці алгоритми можна використовувати для вирішення різних задач, пов’язаних з забезпеченням кібербезпеки у телекомунікаційних системах, наприклад генерація ключів для шифрування чи генерація електронно-цифрових підписів для можливості підтвердження істинності переданих даних.

Серед асиметричних алгоритмів шифрування було обрано алгоритми з використанням еліптичних кривих та проведено аналіз їх особливостей. Було встановлено ефективність даного варіанту шифрування з точки зору варіативності можливих згенерованих електронних підписів та ключів як для шифрування, так і для розшифрування.

Було програмно реалізовано алгоритм генерації відкритих і закритих ключів за допомогою мови програмування Python на основі параметрів стандартизованої кривої, яку було визначено групою SECG. Після цього було реалізовано алгоритм генерації та перевірки електронно-цифрових підписів на

основі тієї ж стандартизованої прямої. Таким чином було підтверджено практичну цінність даних алгоритмів, бо вони надійні з точки зору захисту інформації та досить легкі в реалізації, що робить їх надзвичайно ефективними у сучасному світі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. http://l.lekciya.com.ua/doc/21101/index.html
2. http://jak.bono.odessa.ua/articles/kriptografija-i-shifruvannja-shho-take.php
3. https://wiki.tntu.edu.ua/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%87%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0
4. https://uk.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%87%D0%BD%D0%B0_%D1%81%D1%82%D1%96%D0%B9%D0%BA%D1%96%D1%81%D1%82%D1%8C
5. https://pidruchniki.com/15070412/bankivska_sprava/kriptografichniy_zahist_informatsiyi_sistemi_rozpodilu_klyuchiv
6. http://refpin.ru/ref_otratyyfsaty.html
7. https://sites.google.com/site/kriptografia17/stijka-kriptografia
8. http://www.znanius.com/3851.html
9. https://works.doklad.ru/view/Jl0lQEgYEMs.html
10. Darrel Hankerson, Alfred Menezes, Scott Vanstone Guide to Elliptic Curve Cryptography / Darrel Hankerson // 2004,Springer – Verlag New York Inc. – 205 p
11. Мао, Венбао. Современная криптография: теория и практика. : Пер. с англ. — М. : Издательский дом «Вильямс», 2005. — 768 с.
12. N.Koblitz, Elliptic Curve Cryptosystems [Text]/ Neal Koblitz// Mathematics of Computation – Volume 48 – Number 177 – January 1987 – pp. 203-209
13. Cetin Kaya Coc, Cryptographic Engineering/ Cetin Kaya Coc// Springer Science+Business Media, LLC 2009 – 171 p

14. Василенко О. Н., Теоретико-числовые алгоритмы в криптографии/ Василенко О. Н.// М.: МЦНМО, 2003. – 178 с.

15. J. Solinas. Low-weight binary representations for pairs of integers/ J. Solinas// National Security Agency, USA 2001.

16. V. S. Dimitrov, L. Imbert, and P. Mishra. Fast elliptic curve point multiplication using double-base chains. 2005.

17. Шеннон К. Работы по теории информации и кибернетике / К. Шеннон ; ed. byР. Л. Добрушина,О. Лупанова. – Москва: Иностранная Литература,1963. –829pp.

18. СингхС.Книга шифров.Тайная история шифров и их расшифровки/С. Сингх.–Москва:АСТ:Астрель,2007.–446pp.

19. TuringA.Programmers' Handbook for the Manchester Electronic Computer MarkI/A.Turing. –Oxford: UniversityPress,1952. –110pp.

20. https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%94%D1%96%D1%84%D1%84%D1%96-%D0%93%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0

21. https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%9C%D1%8D%D1%81%D1%81%D0%B8_%E2%80%94_%D0%9E%D0%BC%D1%83%D1%80%D1%8B

22. https://uk.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%95%D0%BB%D1%8C-%D0%93%D0%B0%D0%BC%D0%B0%D0%BB%D1%8F

23. https://uk.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm

24. Standards for Efficient Cryptography SEC 2: Recommended Elliptic Curve Domain Parameters, 2010

25. https://uk.wikipedia.org/wiki/Python

ДОДАТОК А
ЛІСТИНГ КОДУ ГЕНЕРАЦІЇ ВІДКРИТИХ І ЗАКРИТИХ КЛЮЧІВ

```
   #!/usr/bin
/env python3


                    import collections
                    import random


                    EllipticCurve = collections.namedtuple('EllipticCurve', 'name p a b g n h')


                    curve = EllipticCurve(
                        'secp256k1',
                        # Field characteristic.
                        p=0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f,
                        # Curve coefficients.
                        a=0,
                        b=7,
                        # Base point.
                        g=(0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798,
                           0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8),
                        # Subgroup order.
                        n=0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141,
                        # Subgroup cofactor.
                        h=1,
                    )




                    #                                Modular                                arithmetic
                ############################################################


                    def inverse_mod(k, p):
                        """Returns the inverse of k modulo p.
                        This function returns the only integer x such that (x * k) % p == 1.
                        k must be non-zero and p must be a prime.
                        """
```

```
        if k == 0:
            raise ZeroDivisionError('division by zero')


        if k < 0:
            # k ** -1 = p - (-k) ** -1  (mod p)
            return p - inverse_mod(-k, p)


        # Extended Euclidean algorithm.
        s, old_s = 0, 1
        t, old_t = 1, 0
        r, old_r = p, k


        while r != 0:
            quotient = old_r // r
            old_r, r = r, old_r - quotient * r
            old_s, s = s, old_s - quotient * s
            old_t, t = t, old_t - quotient * t


        gcd, x, y = old_r, old_s, old_t


        assert gcd == 1
        assert (k * x) % p == 1


        return x % p




    #       Functions       that       work       on       curve       points
##########################################


    def is_on_curve(point):
        """Returns True if the given point lies on the elliptic curve."""
        if point is None:
            # None represents the point at infinity.
            return True
```

```
    x, y = point

    return (y * y - x * x * x - curve.a * x - curve.b) % curve.p == 0



def point_neg(point):
    """Returns -point."""
    assert is_on_curve(point)

    if point is None:
        # -0 = 0
        return None

    x, y = point
    result = (x, -y % curve.p)

    assert is_on_curve(result)

    return result



def point_add(point1, point2):
    """Returns the result of point1 + point2 according to the group law."""
    assert is_on_curve(point1)
    assert is_on_curve(point2)

    if point1 is None:
        # 0 + point2 = point2
        return point2
    if point2 is None:
        # point1 + 0 = point1
        return point1
```

```
        x1, y1 = point1
        x2, y2 = point2


        if x1 == x2 and y1 != y2:
            # point1 + (-point1) = 0
            return None


        if x1 == x2:
            # This is the case point1 == point2.
            m = (3 * x1 * x1 + curve.a) * inverse_mod(2 * y1, curve.p)
        else:
            # This is the case point1 != point2.
            m = (y1 - y2) * inverse_mod(x1 - x2, curve.p)


        x3 = m * m - x1 - x2
        y3 = y1 + m * (x3 - x1)
        result = (x3 % curve.p,
                  -y3 % curve.p)


        assert is_on_curve(result)


        return result




    def scalar_mult(k, point):
        """Returns  k  *  point  computed  using  the  double  and  point_add
algorithm."""
        assert is_on_curve(point)


        if k % curve.n == 0 or point is None:
            return None


        if k < 0:
            # k * point = -k * (-point)
```

```
            return scalar_mult(-k, point_neg(point))

        result = None
        addend = point

        while k:
            if k & 1:
                # Add.
                result = point_add(result, addend)

            # Double.
            addend = point_add(addend, addend)

            k >>= 1

        assert is_on_curve(result)

        return result



    #             Keypair             generation             and             ECDHE
################################################

    def make_keypair():
        """Generates a random private-public key pair."""
        private_key = random.randrange(1, curve.n)
        public_key = scalar_mult(private_key, curve.g)

        return private_key, public_key



    print('Curve:', curve.name)
```

```
# Alice generates her own keypair.
alice_private_key, alice_public_key = make_keypair()
print("Alice's private key:", hex(alice_private_key))
print("Alice's public key: (0x{:x}, 0x{:x})".format(*alice_public_key))


# Bob generates his own key pair.
bob_private_key, bob_public_key = make_keypair()
print("Bob's private key:", hex(bob_private_key))
print("Bob's public key: (0x{:x}, 0x{:x})".format(*bob_public_key))


# Alice and Bob exchange their public keys and calculate the shared secret.
s1 = scalar_mult(alice_private_key, bob_public_key)
s2 = scalar_mult(bob_private_key, alice_public_key)
assert s1 == s2


print('Shared secret: (0x{:x}, 0x{:x})'.format(*s1))
```

ДОДАТОК Б

ЛІСТИНГ КОДУ ГЕНЕРАЦІЇ ТА ПЕРЕВІРКИ ЕЛЕКТРОННО-ЦИФРОВИХ ПІДПИСІВ

```
#!/usr/bin/env python3


import collections
import hashlib
import random


EllipticCurve = collections.namedtuple('EllipticCurve', 'name p a b g n h')


curve = EllipticCurve(
    'secp256k1',
    # Field characteristic.
    p=0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f,
    # Curve coefficients.
    a=0,
    b=7,
    # Base point.
    g=(0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798,
       0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8),
    # Subgroup order.
    n=0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141,
    # Subgroup cofactor.
    h=1,
)




# Modular arithmetic ##########################################################


def inverse_mod(k, p):
    """Returns the inverse of k modulo p.
    This function returns the only integer x such that (x * k) % p == 1.
    k must be non-zero and p must be a prime.
    """
    if k == 0:
```

```
        raise ZeroDivisionError('division by zero')


    if k < 0:
        # k ** -1 = p - (-k) ** -1  (mod p)
        return p - inverse_mod(-k, p)


    # Extended Euclidean algorithm.
    s, old_s = 0, 1
    t, old_t = 1, 0
    r, old_r = p, k


    while r != 0:
        quotient = old_r // r
        old_r, r = r, old_r - quotient * r
        old_s, s = s, old_s - quotient * s
        old_t, t = t, old_t - quotient * t


    gcd, x, y = old_r, old_s, old_t


    assert gcd == 1
    assert (k * x) % p == 1


    return x % p




# Functions that work on curve points #########################################


def is_on_curve(point):
    """Returns True if the given point lies on the elliptic curve."""
    if point is None:
        # None represents the point at infinity.
        return True


    x, y = point
```

```
    return (y * y - x * x * x - curve.a * x - curve.b) % curve.p == 0



def point_neg(point):
    """Returns -point."""
    assert is_on_curve(point)

    if point is None:
        # -0 = 0
        return None

    x, y = point
    result = (x, -y % curve.p)

    assert is_on_curve(result)

    return result



def point_add(point1, point2):
    """Returns the result of point1 + point2 according to the group law."""
    assert is_on_curve(point1)
    assert is_on_curve(point2)

    if point1 is None:
        # 0 + point2 = point2
        return point2
    if point2 is None:
        # point1 + 0 = point1
        return point1

    x1, y1 = point1
```

```
    x2, y2 = point2


    if x1 == x2 and y1 != y2:
        # point1 + (-point1) = 0
        return None


    if x1 == x2:
        # This is the case point1 == point2.
        m = (3 * x1 * x1 + curve.a) * inverse_mod(2 * y1, curve.p)
    else:
        # This is the case point1 != point2.
        m = (y1 - y2) * inverse_mod(x1 - x2, curve.p)


    x3 = m * m - x1 - x2
    y3 = y1 + m * (x3 - x1)
    result = (x3 % curve.p,
              -y3 % curve.p)


    assert is_on_curve(result)


    return result




def scalar_mult(k, point):
    """Returns k * point computed using the double and point_add algorithm."""
    assert is_on_curve(point)


    if k % curve.n == 0 or point is None:
        return None


    if k < 0:
        # k * point = -k * (-point)
        return scalar_mult(-k, point_neg(point))
```

```
    result = None
    addend = point

    while k:
        if k & 1:
            # Add.
            result = point_add(result, addend)

        # Double.
        addend = point_add(addend, addend)

        k >>= 1

    assert is_on_curve(result)

    return result


# Keypair generation and ECDSA ########################################################

def make_keypair():
    """Generates a random private-public key pair."""
    private_key = random.randrange(1, curve.n)
    public_key = scalar_mult(private_key, curve.g)

    return private_key, public_key


def hash_message(message):
    """Returns the truncated SHA521 hash of the message."""
    message_hash = hashlib.sha512(message).digest()
    e = int.from_bytes(message_hash, 'big')
```

```
    # FIPS 180 says that when a hash needs to be truncated, the rightmost bits
    # should be discarded.
    z = e >> (e.bit_length() - curve.n.bit_length())

    assert z.bit_length() <= curve.n.bit_length()

    return z


def sign_message(private_key, message):
    z = hash_message(message)

    r = 0
    s = 0

    while not r or not s:
        k = random.randrange(1, curve.n)
        x, y = scalar_mult(k, curve.g)

        r = x % curve.n
        s = ((z + r * private_key) * inverse_mod(k, curve.n)) % curve.n

    return (r, s)


def verify_signature(public_key, message, signature):
    z = hash_message(message)

    r, s = signature

    w = inverse_mod(s, curve.n)
```

```
    u1 = (z * w) % curve.n
    u2 = (r * w) % curve.n


    x, y = point_add(scalar_mult(u1, curve.g),
                     scalar_mult(u2, public_key))


    if (r % curve.n) == (x % curve.n):
        return 'signature matches'
    else:
        return 'invalid signature'




print('Curve:', curve.name)


private, public = make_keypair()
print("Private key:", hex(private))
print("Public key: (0x{:x}, 0x{:x})".format(*public))


msg = b'Hello!'
signature = sign_message(private, msg)


print()
print('Message:', msg)
print('Signature: (0x{:x}, 0x{:x})'.format(*signature))
print('Verification:', verify_signature(public, msg, signature))


msg = b'Hi there!'
print()
print('Message:', msg)
print('Verification:', verify_signature(public, msg, signature))


private, public = make_keypair()


msg = b'Hello!'
```

```
print()
print('Message:', msg)
print("Public key: (0x{:x}, 0x{:x})".format(*public))
print('Verification:', verify_signature(public, msg, signature))
```