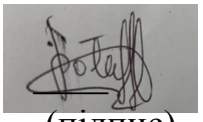
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

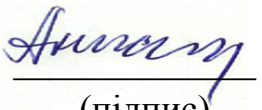
Кваліфікаційна робота

на здобуття ступеню бакалавру
за освітньо-професійною програмою "Інформатика"
спеціальності 122 Комп'ютерні науки
на тему:

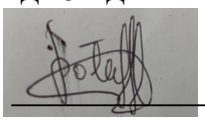
Криптопротоколи, що використовують квадратичні лишки

OBJ OBJ

Виконала студентка 4-го курсу
Осадча Поліна Ігорівна  (підпис)

Науковий керівник:
Член-кореспондент НАН України, професор
Анісімов Анатолій Васильович  (підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент  (підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри математичної інформатики

«____» _____ 2022 р.,
протокол № ___

Завідувач кафедри
професор, доктор фіз.-мат. наук

Терещенко В. М. _____
(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 52 сторінки, 4 таблиці, 21 джерел посилань.

КРИПТОГРАФІЧНІ АЛГОРИТМИ, АСИМЕТРИЧНЕ ШИФРУВАННЯ, СИМЕТРИЧНЕ ШИФРУВАННЯ, КРИПТОГРАФІЯ, АУТЕНТИФІКАЦІЙНІ ПРОТОКОЛИ, КВАДРАТНІ ЛИШКИ

Об’єктом дослідження і розробки є криптопротоколи, що використовують квадратні лишки, розробка програмного модуля одного з протоколів.

Метою даної роботи є проаналізувати існуючі криптографічні протоколи, які пропонують способи розв’язання задачі автентифікації та верифікації, розглянути і реалізувати відомий протокол Фейге-Фіата-Шаміра, дослідити його роботу і залежність від різних параметрів.

Методи розроблення: імплементація протоколу Фейге-Фіата-Шаміра, з використанням власноруч реалізованої багаторозрядної арифметики, швидкого піднесення до степеня за модулем, розширеного алгоритму Евкліда, тест Міллера-Рабіна для перевірки великого числа на простоту та принципів ООП. Інструменти розроблення: операційна система - Ubuntu 18.04.5 LTS, середовище програмування Clion. CMake для збірки вихідного коду, мова програмування C++, stl, та використання генератора випадкових чисел з використанням independent_bits_engine бібліотеки random.

Результати роботи: на базі алгоритмів для швидкої роботи з великими числами й основними кроками алгоритму Фейге-Фіата-Шаміра була успішно розроблена та реалізована швидка програмна реалізація протоколу. Даний алгоритм можна застосовувати для вирішення задачі швидкої автентифікації, на будь-яких обчислювальних потужностях. На сьогоднішній день будь-яка сфера діяльності не можлива без збереження і пересилання даних, і ефективність системи доступу залежить від того, наскільки надійно і швидко автентифікований користувач.

Результати проведених у дипломній роботі досліджень можуть бути використані у навчальних цілях при аналізі певних класів криптографічних протоколів.

Напрями подальших досліджень – більш поглиблений аналіз крипто протоколів, розширення реалізованої програмної системи новими рішеннями. Аналіз і програмна реалізація покращеного алгоритму.

ЗМІСТ

ВСТУП

У сучасному світі розвиток індустріально-інформаційного суспільства займає ключовий аспект нашого життя, і все більшого значення у цьому процесі займають автоматизовані системи зберігання, обробки та передачі інформації, основані на використанні комп'ютерних технологій. Такі системи вимагають постійного вдосконалення, адже разом з розвитком зусиль для забезпечення цілісності систем та безпечного збереження даних, зростають і спроби їх злому та розсекречування. Поширення таких систем в глобальних масштабах призводить до необхідності узагальнення принципів побудови відповідних технічних засобів, а також їх вживання.

Побудова автоматизованих систем, які вирішують задачу забезпечення цілісності даних здійснюється криптографією – наукою про захист інформації. Криптосистема складається з кількох учасників, що повністю довіряють один одному, і які потребують передавачі певної, не призначеної для інших осіб, інформації між собою. На цьому етапі й виникає завдання забезпечення конфіденційності, тобто захист секретної інформації від противника.

Запобігання загрози контролю за джерелами інформації потребує спеціальну систему контролю за доступом до ресурсів. Ця система повинна відповідати двом вимогам:

- кожен бажаючий повинен мати можливість звернутися до цієї системи анонімно;
- необхідно довести своє право на доступ до ресурсів.

Перелічені властивості в криптографії називається невідстежністю, забезпечення якої такой є одним із завдань криптографії.

При побудові криптографічних систем необхідно одночасно вирішувати багато питань для безпечного обміну інформацією, наприклад:

- Повідомлення прийшло від очікуваного конкретного відправника?
- Можливо воно було сфабриковано або змінене кимось стороннім?

- Повідомлення було створено даним відправником, чи він просто переслав чиєсь повідомлення?

Усі ці та подібні питання належать до галузі досліджень, що отримала назву протоколи забезпечення безпеки.

Варто наголосити, що захист інформації безпосередньо потрібен для забезпечення рентабельності автоматизованих систем. З цієї причини, одним з положень Закону України «Про електронні документи і електронний документообіг» є потреба впровадження засобів криптографічного захисту інформації (КЗІ). Зазначимо, що стандарти криптографічного захисту інформації, які діють в Україні на сьогоднішній день, не включають у себе всю різноманітність потрібних криптоалгоритмів та криптопротоколів (відсутні стандарти асиметричних криптосистем, систем узгодження симетричних ключів тощо).

В даний час створено безліч криптографічних протоколів, призначених для різних умов застосування та вирішення різноманітних прикладних задач. Проблема побудови криптографічних протоколів напряму пов'язана з дослідженнями рівня їх безпеки, який можна досягти при їх застосуванні. Багато протоколів, які спочатку вважались надійними, зазнали атаки і відшукання вразливих місць, що спростували статус надійності.

Криптографія є молодою наукою, досі після більш ніж двадцятирічних досліджень з'являються все нові і нові теоретичні методи аналізу протоколів, на основі яких розробляється автоматизовані засоби аналізу безпеки протоколів, з’являються удосконалення старих протоколів та розробка абсолютно нових. Наведені вище обставини визначають актуальність теми дипломної роботи. У цій дипломній роботі викладені основні поняття, пов'язані з криптографічними протоколами, розглянуті основні властивості, що характеризують їх безпеку, а також створено програмну реалізацію одного з відомих криптопротоколів, з детальним аналізом.

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ КРИПТОПРОТОКОЛІВ, ЩО ВИКОРИСТОВУЮТЬ КВАДРАТИЧНІ ЛИШКИ

Використання протоколів, зокрема криптологічних протоколів є одним з найважливіших та найрозповсюдженіших методів захисту інформації. Вони є основними механізмами при процесах автентифікації, управлінні та сертифікації ключів, та звісно, при безпосередньому захисті інформації та ресурсів. Метою цього розділу є ознайомлення з базою криптографічних протоколів, розгляд і аналіз декількох основних криптопротоколів які мають широке застосування у сучасному світі.

1.1 Криптографічний протокол

1.1.1 Визначення та основні поняття

Проблеми забезпечення безпеки обміну та збереження інформації є складними та обширними.

Убезпечення інформації має на меті не лише створення умов для збереження якоїсь таємниці(будь то службокої, або комерційної). Цей процес ставить на меті розробку ефективних моделей протидії різним атакам, проведення надійної аутентифікації офіційних суб'єктів і об'єктів обчислювальних мереж, безпечної передачі певної важливої інформації між ними.

Криптографічні протоколи свторення саме для вирішення таких завдань.

Існує декілька визначень криптографічних протоколів, але всі вони посилаються на визначення поняття “протокол”.

Протокол – деяка послідовність кроків, які виконує щонайменше дві сторони для спільного вирішення певної задачі. Кроки протоколу обов’язково відбуваються в порядку зазначеної наперед черговості дій, тобто жоден з них не може почати виконання раніше, ніж закінчиться попередній. Згідно з визначенням, протокол - це розподілений алгоритм для вирішення певного завдання деякою

сукупністю об'єктів і суб'єктів, кожен з яких досягає спільної мети з використанням приватних (розподілених) алгоритмів. Важливо, що всі об’єкти та суб’єкти виконують ідентичні дії та роботу з даними, кроки синхронізації і відновлення роботи в разі несправності системи(багів), при виконанні згаданих розподілених алгоритмів.

Найпростішими визначеннями криптопротоколу є наступний:

- *Криптографічний протокол* – це протокол, який використовує криптографію.

  Або ж: *криптографічним протоколом* називається протокол, в основі якого лежить криптографічний алгоритм.

Більш розгорнуте визначення криптографічного протоколу звучить наступним чином:

- Криптографічний протокол – протокол, призначений для виконання функцій криптографічного устрою, при виконанні якого всі учасники процесу застосовують криптографічні алгоритми.

Зазначимо, *криптографічний систем*а – це система, яка забезпечує захист інформації за допомогою криптографічних методів. Головними функціями цієї системи є забезпечення цілісності, конфіденційності, аутентфікації. А також, вона включає в себе такі підсистеми як: шифрування, підпис, ідентифікація, імітозахист та ін.

Є відмінності криптографічних протоколів від криптосистем, з яких ключовими виділяють наступні:

- не довіра учасників протоколу один одному;
- можливість участі більше двох учасників у протоколі;
- інтерактивна можливість протоколу, тобто є багатораундовий обмін повідомленнями між учасниками

Залежно від контексту і суті, сам термін «криптографічні протоколи» варто сприймати по різному.

У вузькому розумінні - це криптографічний алгоритм, або ж просто послідовність деяких операцій з вихідними даними, які включають їх зберігання,

пересилання, шифрування, формування, генерацію і тд. Тут, у кінцевому висновку виконання певного алгоритму виконується, одна, або рідше дві функції.

Яскравим прикладом слугує алгоритм Діффі-Хеллмана, який використовують для формування єдиного симетричного ключа у незахищеному середовищі.

У широкому ж розумінні, поняття крипто-протоколу можна показати на прикладі Інтернет-стандартів. У даному випадку криптографічні протоколи є частиною взаємодії у зразку відкритої моделі OSI.

Таблиця 1.1

<table>
<tr><td colspan="2">Користувач А</td><td></td><td colspan="2">Користувач Б</td></tr>
<tr><td rowspan="3">Рівень додатків HTT, DNS, SMTP, FTP, SNMP, POP3, Telnet</td><td>7</td><td rowspan="3">S/MIME, PGP, SET, SSH,
⟷
Kerberos, PEM, S-HTTP</td><td>7</td><td rowspan="3">Рівень додатків HTT, DNS, SMTP, FTP, SNMP, POP3, Telnett</td></tr>
<tr><td>6</td><td>6</td></tr>
<tr><td>5</td><td>5</td></tr>
<tr><td>Транспортний рівень UDP/TCP</td><td>4</td><td>SSL, TLS
⟷</td><td>4</td><td>Транспортний рівень UDP/TCP</td></tr>
<tr><td>Рівень з’єднань IP</td><td>3</td><td>IPsec
⟷</td><td>3</td><td>Рівень з’єднань IP</td></tr>
<tr><td rowspan="2">Мережевий рівень Ethernet, FDDI, Token, Ring, PPP</td><td>2</td><td rowspan="2">ECP CHAP
⟷</td><td>2</td><td rowspan="2">Мережевий рівень Ethernet, FDDI, Token, Ring, PPP</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td colspan="5">Фізичне транспортне середовище</td></tr>
</table>

1.1.2. Завдання та властивості

Основними задами убезпечення інформації, які необхідно розв’язати з використанням криптографічних протоколів є наступні:

- обмін секретами(ключами), із забезпеченням захищеного обміну даними, без жодних припущень, про попереднє спілкування сторін обміну ключами;
- процес автентифікації сторін, між якими встановлюється зв’язок;
- авторизація, для встановлення доступу до телекомунікаційних інформаційних служб;
- моделювання різноманітних процесів аутентифікації;
- створення, розподіл та підтвердження криптографічних ключів;
- захист взаємодії між учасниками;
- створення розподілу відповідальності сторін.

Завдяки широкому застосуванню таких відкритих мереж передавання даних, як Internet і побудованих на їхній базі таких мереж як Intranet і Extranet, крипто-протоколи мають усе ширше застосування при розв’язуванні різного спектру задач та створення послуг, які отримують користувачі згаданих мереж.

Криптографічні протоколи повинні мати наступні властивості:

- Всі учасники протоколу повинні знати протокол і всі його кроки наперед;
- всі учасники мають дати згоду слідувати протоколу;
- протокол повинен бути чітко і однозначно визначено наперед;
- протокол повинен описувати реакцію учасників на будь-які ситуації, що теоретично можуть виникати під час реалізації;
- повнота алгоритма - тобто, він повинен вирішувати поставлене завдання.

1.1.3. Функції

Перелік функцій крипто-протоколів:

- аутентифікація сторін та джерела даних;
- заборона від відмови виконання;
- неможливість відмови з доказом джерела та отримання;
- конфіденційність;
- цілісність даних;
- забезпечення цілісності з'єднання без відновлення та з відновленням;
- розподілення доступу.

Зазвичай протокол розрахований на двох-чотирьох учасників. Один з учасників виступає ініціатором запуску протоколу. Він генерує інформацію для передачі, на підставі інформації, наявної у нього. Далі ініціатор виконує деяку послідовність дій, в ході якох формує повідомлення та зрештою передає його наступному учаснику, зазначеного у послідовності дій даного протоколу. Отримавши повідомлення від ініціатора, він також виконує над ним свою відповідну(зазначену наперед) послідовність дій, зберагіє в своїй пам'яті необхідну інформацію. Так завершується 1-ий цикл, після якого роль ініціатора отримує наступний учасник. Аналогічно, будуються наступні цикли виконання протоколу. Всі учасники протоколу мають власну мету - отримати певну інформацію.

При досягненні своєї мети кожним з учасників, протокол вважається завершеним. Якщо ж ні, то протокол переривається.

1.1.4. Вимоги до безпеки

Також, у протоколі можуть бути присутніми зловмисники, учасники-противники, це не основні учасники, які чесно його виконують. Таких ворогів розрізнають на активних та пасивних.

Пасивний ворог - той, хто перехоплює всі повідомлення в каналі зв'язку, намагається витягти з них максимум інформації, але безпосередньо не втручається у хід роботи протоколу. Він є неявним(скритим) учасником протоколу, присутність якого повина бути врахованою та проаналізованою з точки зору безпеки протоколу.

Активний противник в свою чергу, стає несанкціонованим учасником протоколу, прихованим для санкціонованих учасників. Очевидно противник не зобов'язаний дотримуватися протоколу, його мета лише підтримувати видимість нормального виконання протоколу.

Найнебезпечнішим противником є той, хто отримав доступ до ключів (навіть виведеними з дії), скоріш за все учасники протоколу про це не знають.

Тож для ускладнення і унеможливлення атаки з боку супротивників, криптопротоколи мають певні вимоги до безпеки:

1. Аутентифікація (нешироковомовний):
    1.1. аутентифікація суб'єкта;
    1.2. аутентифікація повідомлення;
    1.3. захист від повторень.
2. Аутентифікація під час розсики на багато адрес, або під час підключення до системи підписки:
    2.1. неявна (прихована) аутентифікація отримувача;
    2.2. аутентифікація самого джерела.
3. Авторизація (третя сторона, якій довіряють учасники);
4. Властивості генерації секретів(ключів):
    4.1. аутентифікація ключа;
    4.2. підтвердження правильності ключа;
    4.3. захищеність від читання тому;
    4.4. формування нових ключів;
    4.5. захищена можливість домовитися про параметри безпеки.
5. Конфіденційність;

6. Анонімність:
    6.1. захист ідентифікаторів від прослуховування (незв'язність);
    6.2. захист від інших учасників.
7. Обмежений захист від атак, таких як "відмова при обслуговуванні";
8. Заборона відмовитись від вже виконанних дій:
    8.1. підзвітність;
    8.2. доказ джерела;
    8.3. доказ одержувача.
9. Безпечна тимчасова властивість.

1.1.5. Приклади

У даному пункті наведено ознайомлення з деякими типами крипто протоколів та огляд математичних завдань, які виникають під час дослідження стійкості. Розглянемо декілька прикладів завдань:

1. Підписання контракту: Взаємодія двох абонентів, які хочуть підписати контракт, не довіряючи один одному. Задача зробити це, не допускаючи ситуації, коли один з учасників отримав підпис іншого, а сам підпис не поставив.

2. Ідентифікація абонента: Взаємодія двох абонентів A і B. Абоненту A необхідно довести абоненту B, що він являється саме “A”, тобто, що він не зловмисник. Наглядний приклад: A - власник дкякої карти(кредитної, пропускної, цифровий ключ, тощо), B - банк, адміністратор, пропускна система тощо.

3. Візантійська угода: Взаємодія декількох віддалених абонентів, які отримали накази від одного спільного центру, при цьому, деяка частина з усіх абонентів(включно з центром), можуть виявитись зловмисниками. Задача розробити унітарний план дій, який буде точно виграшним для абонентів. Цю задачу називають “задача про візантійських генералів”.

Опишемо приклад, якому це завдання зобов'язане своєю назвою. Уявимо ніч напередодні великої битви. Чисельність візантійсьї армії - n легіонів, на чолі кожного легіону стоїть свій генерал. Також у армії є головнокомандувач, якому

підпорядковуються усі генерали. Основна проблема на війні в тому, що у програшній ситуації імперії близько однієї третини генералів візантійської армії, включно з головнокомандуючим, теоретично можуть стати зрадниками. Протягом ночі кожен генерал отримує наказ про дії легіону на ранок битви від головнокомандувача. Тут можливі лише два варіанти наказу: “відступати” або ”атакувати”. У ситуації, що усі чесні генерали атакують - армія одержує перемогу. Якщо ж усі відступають, тоді вдається зберегти цілістність армії. Але у випадку, що частина атакує, а інша частина відступає - армію чекає поразка. У випадку, що зрадником став головнокомандувач, то разні генерали отримають різні накази, і тоді беззаперечне виконання його наказів не варто виконувати. При незалежній дії кожного з генералів від інших, результати битви, скоріш за все виявляться неуспішними. Отже, генералам необхідно обмінюватись інформації один з одним про отримані накази від головнокомандувача, з метою прийти до згоди.

4. Вгадування підкинутої монети по телефону: Взаємодія двох абонентів, які абсолюнто не довіряють один одному. Мета учасників підкинути жереб за використанням монети. Головна задачу у тому, щоб той, хто підкидає монету, не змінив результат підкидання після того, як другий учасник озвучить свою здогадку про результат підкидання. Наведемо описа схеми Блюма - Мікалі, що є одним з найпростіших протоколів підкидання монети по телефону. Для його реалізації у абоненти A і B повинні мати односторонню функція y=f(x), що задовольняє наступним умовам:

1) X – скінчена множина цілих чисел, що містить рівну кількість непарних і парних чисел;

2) будь-які числа x1, x2 із X, що мають один образ f(x1)=f(x2), являються однієї парності;

3) заданий образ f(x) є таким, що по ньому важко обчислити парність невідомого значення “x”.

Роль підкидання монети грає випадковий і рівноймовірний вибір елемента x із X. Орел і решка - це, відповідно, парність і непарність значення x. Нехай A -

учісник, який підкидає монету, а В - учасник, який намагається вгадати результат. Протокол описується з наступними кроками:

1) А обирає ***x*** (тобто підкидає монету), зашифровує **x**(обчислює ***y***=f (x)), і надсилає значення ***y*** абоненту В;

2) В отримує y. Далі він пробує вгадати парність x, насилає своє припущення А;

3) А отримує здогад від В і повідомляє В, вгадав він, посилаючи йому вибране число x;

4) В перевіряє, чи не обдурює його А. Для цього він обчислює значення f(x) та порівнює його з отриманим значенням y.

1.2. Класифікація криптографічних протоколів

За часи існування криптології було створено велике різноманіття прикладних криптографічних протоколів, які загалом можна поділити на класи за наступними ознаками:

1) За кількістю учасників: двосторонній, тристоронній і т. д.;
2) За кількістю переданих повідомлень:
    a) інтерактивний (коли взаємний обмін повідомленнями);
    b) не інтерактивний (тільки одноразова передача).
3) За цільовим призначенням протоколу:
    a) забезпечення цілісності повідомлень (з або без аутентифікації);
    b) протокол (схема) цифрового підпису (ЦП). Тут протокол індивідуального та групового цифрового підпису, який може бути з або без відновлення повідомлення, протоколи ЦП з доказовістю підробки, тощо;
    c) ідентифікація (аутентифікація) учасників (одностороння аутентифікація, двостороння (взаємна) аутентифікація);

d) розподіл ключів (схема попереднього розподілу ключів, передачі ключа (обміну ключами), спільної генерації ключа, схема генерації секрету та ін.);
e) протокол конфіденційної передачі;

4) За способом функціонування:
   a) інтерактивний / не інтерактивний;
   b) однопрохідний / дво- / три- і т.д. Прохідний;
   c) протокол з арбітром (протокол з посередником). Двосторонній / з довіреною третьою стороною (з центром довіри), і т.п.
5) За типом використовуваних криптографічних систем:
   a) в основі симетричні криптосистеми;
   b) в основі асиметричні криптосистем;
   c) змішані;
6) За складом і розподілу ролей;
   a) протоколи з арбітражем (адвокатом);
   b) протокол із суддівством;

У цьому розділі наведено більш детальне ознайомлення з деякими класами.

1.2.1. Протоколи шифрування/розшифрування (дешифрування)

В основі протоколів даного класу лежить певний симетричний чи асиметричний алгоритм шифрування або дешифрування. Перший виконується під час передачі відправником повідомлення, в результаті виконання якого повідомлення перетворюється з відкритої зрозумілої форми в зашифровану. Дешифрування виконується на прийомі одержувачем, де повідомлення перетворюється з зашифрованого форми у звичайну. Саме так відбувається забезпечення конфіденційності.

Щоб забезпечити цілістність переданих даних, симетричні алгоритми шифрування / розшифрування, зазвичай, поєднуються з алгоритмами обчислення

ІЗВ - імітозахисна вставка. Ключ шифрувіння використовується при передачі та перевірці ІЗВ на прийомі. При використання асиметричних алгоритмів шифрування та розшифрування для отримання властивості цілісності виконують окремий шлях обчислення електронного цифрового підпису (ЕЦП).

1.2.2. Протоколи електронного цифрового підпису (ЕЦП)

Базоб для протоколів даного класу лежить деякий алгоритм обчислення ЕЦП на передачі за використання секретного ключа, того, хто відправляє та перевірка ЕЦП на прийомі за використання відповідного відкритого ключа, який є захищеним від модифікацій та витягується з відкритого довідника. У випадку ствердного результату перевірки, протокол, зазвичай, завершується архівуванням прийнятого повідомлення, того самого відкритого ключа та ЕЦП. Проте, архівування не є обов’язковим, у випадку коли ЕЦП використовується лише задля забезпечення автентичності та цілісності отриманого повідомлення. В такому разі, ЕЦП може бути знищена або відразу після перевірки, або після обмеженого, зазначеного наперед, проміжку часу очікування.

ЕЦП є одним з важливих криптографічних перетворень, що застосовується для надання таких послуг як цілісність, справжність, автентичність, підтвердження авторства, неспростовність тощо, як при виконанні протоколів, так і при отриманні повідомлення.

1.2.3. Протоколи ідентифікації/аутентифікації

За основою протоколу ідентифікації стоїть певний алгоритм перевірки того, що ідентифікований об'єкт (це може бути користувач, якийсь процес, або пристрій), що має певне ім'я та знає секретну інформацію. Ця інформація є відомою лише зазначеному об'єкту. Метод перевірки є непрямим (без зазначення цієї секретної інформації).

Типовою практикою є пов’язування переліку прав і повноважень кожного

об’єкта з його ім'ям (ідентифікатором) у системі. Цей ідентифікатор записано у захищеній базі даних. Протокол ідентифікації можна розширений до протоколу аутентифікації, в якому ідентифікований об'єкт буде перевіряється на права доступу до послуги.

В разі використання ЕЦП у протоколі ідентифікації, секретний ключ ЕЦП відіграватиме роль секретної інформації. Перевірка ЕЦП відбуважться за використання відкритого ключа ЕЦП, знання якого лише дозволяє переконатися в тому, що цей ключ відомий автору, але неможливо визначити сам секретний ключ.

1.2.4. Протоколи автентифікованого розподілу ключів

Даному класу протоколів притамання поєднання аутентифікації користувачів з протоколом генерації і розподілу секретних ключів. Протокол, зазвичай, налічує два або три учасника; третій учасник - центр генерації та розподілу ключів (ЦГРК)(ще називають Тренет - Т).

Маємо тут три етапи протоколу:

1) генерація
2) реєстрація
3) комунікація.

Етап генерації: сервер Т генерує числові значення параметрів системи, включно з своїм закритим та відкритим ключем.

Етап реєстрації: сервер Т ідентифікує користувачів за допомогою пред’явлених документів. Далі, генерується ключова і(або) ідентифікаційна інформація, для кожного користувача, та формується деякий маркер безпеки. Цей маркер безпеки містить необхідні системні константи та відкритий ключ сервера Т.

Етап комунікації: реалізація самого протоколу автентифікованого ключового обміну. Завершення протоколу - формування спільного сеансового ключа.

Однією з основних властивостей протоколу встановлення ключів є його безпека. По суті, безпека встановлення ключів характеризує протистояння можливості обчислення ключа несанкціонованими об’єктами.

У найбільш узагальненому вигляді автентифікація забезпечує гарантії заявлених ідентифікаційних даних об’єкта. Автентифікація може розглядатися тільки в контексті взаємовідносин між комітетом і об’єктом, який перевіряє, тобто перевірником. Автентифікацію, залежно від певних особливостей її реалізації можна поділити на два варіант:

1) комітента представляє пред'явник, який має комунікаційні особливості взаємовідносин з тим, хто перевіряє(об’єктна автентифікація даних).
2) комітент є ресурсом елемента даних, що доступний перевірнику (оригінальна автентифікація даних).

Автентифікація об’єкта в рамках зазначених комінікаційних відносинах забезпечує підтвердження ідентифікаційних даних комітента. Автентифіковані дані комітента підтверджуються тільки тоді, коли надається такий сервіс.

Автентифікація походження даних забезпечує підтвердження ідентифікаційних даних комітента, що несе відповідальність за визначений елемент даних.

1.2.5. Аспекти деяких інших популярних протоколів

Одним з них, є клас протоколів, які забезпечують реалізацію функції *електронні гроші*. Такі протоколи мають забезпечувати дві основні властивості: неможливість підробки грошей та невідстежуваність операцій з ними.

Невідстежуваність досягається наступним чином:

При зверненні в банк, клієнт себе ідентифікує. Зі свого боку банк перевіряє, наявність рахунку у клієнта та можливість видання йому запитаної кількості грошей та валюти. Далі клієнт наредає банку банкноти й отримує так званий “сліпий” підпис. З цього підписаного матеріалу клієнт поступово дістає купюри, в необхідній і доступній для нього кількості, якими він зрештою може

розплачуватися. Варто наголосити, що банк, отримавши ці купюри, не знає, в якому саме з сеансів клієнт їх підписав. Протокол зняття з рахунку гарантує описане.

Простими словами: банк завжди впізнає купюру, яку він підписав, але він не може точно вказати, кому саме і під час якого сеансу він цю купюру видав.

Зазначимо тут, що всі розрахунки з використанням банкнот(купюр), підписаних даним банком-емітентом, можуть бути використаними лише між клієнтами цього банку одни з одним, та з самим банком.

Також цікавими є *протоколи голосування.* Такі протоколи використовуються при проведенні різних виборів. Також з їх допомогою можна гарантувати корректність підрахунку голосів.

Задача забезпечити конфіденційність обчислень можна описати наступним чином:

Маємо N процесорів та стільки ж аргументів. Кожен процесор має свій аргумент і знає його. Задача: процесори повинні спільно обчислити задану функцію від аргументів, таким чином, що жоден процесор в процесі обчислення та кінцевому результаті нічого не знає про чужі аргументи. Крім того, деякі учасники описаного обчислення можуть виявитись супротивниками, і вони намагатимуться перешкоджати правильному обчисленню результату функції.

Відомим стає той факт, що при умові, що частина супротивників не перевищує деякого порогу, то будь-яку функцію можна обчислити конфіденційно.

1.3. Відомі протоколи автентифікації

Для початку розглянемо найпростіший алгоритм автентифікації з публічним ключем.

1. Хост надсилає Алісі випадковий рядок.

2. Аліса зашифровує рядок своїм захищеним ключем та разом зі своїм іменем надсилає хосту.
3. Хост дістає з бази даних публічний ключ Аліси і за його допомогою розшифровує повідомлення. Якщо зашифрований рядок ріврим до того, який хост надіслав Алісі на першому кроці, то Аліса отримує доступ до входу у систему.

Лише сама Аліса має доступ до її приватного ключа, а отже ніхто не може видати себе за Алісу. Зловмисник Єва, яка підслуховує значення, не може визначити секретний ключ Аліси.

Проте, не дуже розумно зашифровувати довільний рядок. Для доведення надійності протоколу було запропоновано наступну модифікацію:

1. Маючи деяке випадкове числа і використовуючи свій приватний ключ, Аліса робить необхідні обчислення та відсилає хосту.
2. Хост обирає(генерує) інше випадкове число та надсилає його Алісі.
3. Базуючись на зазначених вище випадкових числах, Аліса робить деякі обчислення з використанням свого приватного ключа. Далі вона відсилає результат хосту.
4. На отриманих на попередньому кроці числах хост виконує обчислення за допомогою публічного ключа Аліси. Результатом обчислення є перевірка того, що це справді Аліса, і вона знає приватний ключ.

Якщо ж Аліса сумнівається у достовірності хоста, вона може попросити йогов той самий шлях довести свою ідентичність.

Тепер розглянемо алгоритми які дозволяють провести автентифікацію і обмінятися ключами.

1.3.1. Алгоритм Рабіна

Система Рабіна - це асиметрична криптосистема, яка як і будь-яка інша, використовує відкритий + закритий ключі. Відкритий(публічний) ключ

використовується при шифруванні повідомлень, він може бути відомий публічно(зберігатись у відкритій базі), закритий ключ відомий тільки отримувачу.

Алгоритм Рабіна грунтується на складності знаходження дискретного квадратного кореня від числа по складеному модулю. Складність обчислень еквівалентна алгоритму факторизації великих чисел.

Генерація ключів:

1. Генерується два великих простих числа p *та* q . Кожне з чисел є конгруентним 3 по модулю 4.
2. Обчислюється добуток $n = p*q$, де n – буде публічним ключем.

Процес генерації ключів на цьому є завершеним. Для передачі повідомлення M необхідно за допомогою оборотної функції отримати число **m**, яке ми співставляємо повідомленню M. Далі потрібно передати його в зашифрованому вигляді наступним чином: $s = m * m \pmod n$.

Для розшифрування необхідно по черзі обчислити квадратні корені **s** по модулю **p** і **q**. Це можна зробити наступним чином:

$$m_p = s^{(p+1)/4} \bmod p$$

$$m_q = s^{(q+1)/4} \bmod q$$

Наступним кроком, за допомогою розширеного алгоритму Евкліда знаходимо такі p і q, що виконується рівність: $y_p * p + y_q * q = 1$.

Застосувавши Китайську теорему про залишки отримуємо наступні формули:

$$f1 = (y_p * p * m_q + y_q * q * m_p) \bmod n$$

$$f2 = n - f_1$$

$$f3 = (y_p * p * m_q - y_q * q * m_p) \bmod n$$

$$f4 = n - f_3$$

Одне з цих чисел і буде коренем рівняння.

1.3.2. Алгоритм Вільямса

Алгоритмі Рабіна має неоднозначності з коренем, тому Вільямс запропоновав модифікацію алгоритму Рабіна.

Генерація ключів:

1. Генеруємо два великих простих числа ***p*** і ***q*** які задовольняються умовам: p = 3 (mod 8), q = 7 (mod 8).
2. Аналогічно ***n = p * q***. Також необхідно згенерувати мале ***s***, таке що J(t, n) = -1, де J(x, y) - символ Якобі. ***n*** і ***t*** є публічними.

   Закритий ключ ***k*** генерується так:

$$k = ((p - 1) * (q - 1) / 4 + 1) / 2$$

Для зашифрування повідомлення ***m*** потрібно обчислити таке s_1, що $sign(s_1) = J(m, n)$. Далі обчислити таке $m_1 = (t^{s_1} * m) \bmod n$.

Як і в схемі Рабіна

$$s = m_1 * m_1 (mod\ n).$$

$$s_2 = m_1 \bmod 2$$

Трійка ***(s, s1, s2) -*** це фінальний шифротекст.

Щоб розшифрувати ***S*** отримувач обчислює m_2використовуючи:

$$sk = sign(s_2) * m_2 \bmod n$$

Врешті решт m обчислюється за наступною формулою:

$$m = (t^{s_1} * sign(s_1) * m_2) \bmod n$$

За надійністю алгоритми Рабіна і Вільямса, обидва, еквівалентні факторизації числа. Недоліком даних алгоритмів є те, що вони не є стійким до атаки зловмисника, коли останній може сам обрати шифротекст. З цього слідує, що для використання ЦП у алгоритмі, алгоритм потребує модифікацій. Наприклад, можна застосувати односторонню геш-функцію перед підписанням. Або можна дописувати випадковий рядок до повідомлення перед його хешування і підписанням. Але цей підхід руйнує можливість доведення того, що надійність системи еквівалентна факторизація числа.

1.3.3. Протокол Гіллоу - Кіскатра

Протокол Гіллоу - Кіскатра це розширення більш раннього протоколу Фіата - Шаміра. Протокол дозволяє одному учаснику (***A*** - prover) довести іншому учаснику (***B*** - verifier), що він володіє секретною інформацією, не розкриваючи жодного біта цієї інформації.

Безпека протоколу заснована на складності добування квадратного кореня по модулю досить великого складеного числа по заданому модулю n.

Сторона ***A*** відправляє стороні ***B*** свої атрибути ***J***. Стороні ***A*** необхідно переконати сторону ***B***, що це саме її атрибути. Для цього сторона ***A*** доводить своє знання секрету ***x*** стороні ***B***, не розкриваючи при цьому жодного біта самого секрету ***x***. Для цього сторонам потрібно всього 1 раунд.

Алгоритм створення відкритого і закритого ключів:

1. Центр довіри ***T*** вибирає два різних випадкових простих числа ***p*** і ***q,*** після чого обчислюємо модуль ***n = p*q.***
2. ***T*** обирає ціле число e($1 < e < \varphi(n)$) взаємно просте із значенням функції $\varphi(n)$. Де $\varphi(n)$ - функція Ейлера.
3. ***T*** обчислює $s = e^{-1}(mod\ n)$ і секрет $x = J^{-s}(mod\ n)$.
4. ***T*** обчислює $y = x^e \bmod n$.

5. Трійка {n,e,y} публікується в якості відкритого ключа.
6. ***x*** грає роль закритого ключа, і передається стороні ***A***.

Обмін повідомленнями:

1. ***A*** вибирає випадкове ціле ***r***, що знаходиться в діапазоні від 1 до n-1. ***A*** обчислює $a = r^e \bmod n$ і відправляє його ***B***.
2. ***B*** вибирає випадкове ціле ***c***, що знаходиться в діапазоні від 0 до e-1. ***B*** надсилає ***c*** стороні ***A***.
3. ***A*** обчислює $z = rx^c \bmod n$ і надсилає його ***B***.
4. ***B*** перевіряє: якщо $z^e = ay^c \bmod n$, то ***A*** довела знання секрету ***x***.

У порівнянні з протоколом Фіата-Шаміра протокол Гіллоу - Кіскатра має менше число повідомлень, якими необхідно обмінятися сторонам для ідентифікації. Протокол вимагає тільки один раунд обміну повідомленнями, має більш низькі вимоги до пам'яті, використовуваної для зберігання секретів користувачів, проте вимагає більшого обсягу обчислень.

Крім того, схему ідентифікації Гіллоу - Кіскатра можна легко перетворити в схему підпису.

РОЗДІЛ 2 ДЕТАЛЬНИЙ ОГЛЯД ПРОТОКОЛУ ФЕЙГЕ-ФІАТА-ШАМІРА

Чому ми все ще передаємо наші паролі через мережу? Чому ми досі хешуємо паролі? Чому ми не можемо придумати власне випадкове значення, а потім довести всім, що ми знаємо секрет?

Що ж, метод Файге-Фіат-Шаміра забезпечує доказ без розголошення — створений Уріелем Файгі, Амосом Фіатом і Аді Шаміром у 1988 році, де Боб може щось довести Алісі, не розкриваючи наявної у нього інформації. Це один з найефективніших алогритмів, побудованих з використанням підходу ZKP. У цьому розділі, ми познайомимось з методом ZKP, детальніше розглянемо принцип роботи протоколу Файге-Фіата-Шаміра, тонкоші його реалізації, та переваги перед іншими алгоритмами.

2.1 Zero-Knowledge Proofs

Протокол з нульовим розголошенням (ZKP) - це спосіб автентифікації, без обміну жодними паролями, що означає, що їх не можна вкрасти. Це дозволяє робити обмін даними добре захищеним, оскільки не існує якогось певного ключа, який хтось може розсекретити і отримати доступ до всієї інформації, якою діляться між собою два учасника. ZKP дозволяє вам довести, що ви знаєте якусь таємницю (або багато таємниць) комусь із іншого «кінця» спілкування, не розкриваючи її.

Сам термін “нульове розголошення” походить від того, що ніякої ("нульової") інформації про таємницю не розголошується, але друга сторона (“Verifier” -8 верифікатор) переконана, що перша сторона (“Prover” - випробувач) знає, про яку таємницю йдеться. Часто використовується простий приклад, щоб проілюструвати логіку даної криптографічної технології: «Печера Алі Баби».

Давайте уявимо, що два персонажі, Аліса та Боб, опиняються біля печери, яка має два окремі входи до двох окремих доріжок (A та B). Усередині печери є двері, що з'єднують обидві стежки, але їх можна відкрити лише за допомогою секретного коду. Боб ("prover") володіє цим кодом, і Аліса ("verifier") хоче його купити, але спочатку їй необхідно впевнитися, що Боб не бреше. Як Боб може показати Алісі, що він має код, не розкриваючи його зміст? Щоб досягти цього, вони роблять наступне: Аліса чекає біля печери, а Боб навмання входить через одну з дверей (A або B). Опинившись всередині, Аліса підходить до входу, кличе Боба і просить його вийти одним із двох шляхів. Оскільки Боб має секретний код, він завжди зможе повернутися тим шляхом, який обрала Аліса, хоча це може не збігатися з тим шляхом, який він обрав спочатку, оскільки в цьому випадку він може відкрити двері і вийти через інший бік.

Аліса все ж хоче впевнитись, що це не просто удача, отже з ймовірністю 50% обидва обрали один і той же шлях. Тож, якщо цю вправу виконувати кілька разів, ймовірність того, що Боб виходить через той самий шлях, вибраний Алісою, не маючи коду, поступово зменшується, поки “просто удача” стає практично неможливою.

Висновок - якщо Боб виходить достатню кількість разів цим шляхом, він однозначно продемонстрував Алісі, що у нього є секретний код. І для цього не було необхідності його розкривати.

2.2. Протокол Фейге-Фіата-Шаміра

2.2.1. Протокол Фіата-Шаміра

Одним з найбільш відомих протоколів ідентифікації особистості за допомогою ZKP є протокол, запропонований Амосом Фіатом і Аді Шаміром, стійкість якого грунтується на складності обчислення квадратного кореня по модулю досить великого складеного числа n, факторизація якого невідома.

Спочатку третя особа(Т), якій довіряють учасники протоколу(Аліса та Боб), обирає два великі прості числа ***p*** і ***q***, щоб обчислити значення ***n = p*q*** - публічний модуль. Значення ***p*** і ***q*** зберігаються прихованими. Аліса обирає секретне число ***s*** в межах від 1 і n-1. Вона обчислює $v = s^2 mod\, n.$ Аліса зберігає ***s*** як свій секретний ключ і реєструє ***v*** як свій публічний ключ разом з третьою особою. Саме знання цього ключа ***s*** необхідно довести А стороні В без його розголошення за t раундів. Кожна акредитація складається з наступних етапів:

1. Аліса обирає випадкове ***r*** з інтервалу (1, n-1) і надсилає $x = r^2 (mod\, n)$ Бобу.
2. Боб випадково обирає біт ***e*** (0 або 1) і надсилає їх Алісі.
3. Аліса обчислює $y = r * v^e (mod\, n)$ і відправляє його назад до Боба.
4. Боб перевіряє рівність $y^2 = x * v^e (mod\, n)$. Якщо воно вірне, то відбувається перехід до наступного раунду протоколу, інакше цей доказ не приймається.

Вибір e з множини передбачає, що якщо Аліса дійсно знає секрет, то вона завжди зможе правильно відповісти, незалежно від обраного ***e***.

Припустимо, що Аліса хоче обдурити Боба, вибирає випадкове ***r*** і відсилає $x = r^2/v$, тоді якщо e = 0, то Аліса вдало повертає Бобу y = r, але якщо ж e = 1, Аліса не зможе правильно відповісти, тому що не знає s, а витягти квадратний корінь з v по модулю n досить складно.

Ймовірність того, що Аліса не знає секрету ***s***, але переконує в зворотному перевіряючого Боба буде оцінюватися імовірністю рівною $p = 2^{-t}$, де t - число акредитацій. Для досягнення високої точності його вибирають досить великим (t = 20 - 40). Таким чином, Боб засвідчується в правдивості знання секретного ключа s Алісою тоді і тільки тоді, коли всі t раундів пройшли успішно.

Для того щоб цей протокол коректно виконувався, Аліса ніколи не повинна повторно використовувати значення x. Адже, в іншому випадку Боб під час іншого циклу відправив би Алісі на 2-му кроці інший випадковий біт r, то Боб б мав обидві відповіді Аліси. Після цього Боб може обчислити значення s, і йому буде відомий секретний ключ Аліси, що вже руйнує надійність.

2.2.2 Схема ідентифікації Фейге-Фіата-Шаміра

У своїх роботах Фейге, Фіат та Шамір показали, як паралельна схема може підвищити кількість акредитацій на раунд та зменшити обсяг взаємодії між Алісою та Бобом.

Спочатку, як і в попередньому прикладі, генерується добуток двох простих числа $\boldsymbol{p}$ і $\boldsymbol{q}$, щоб обчислити значення $\boldsymbol{n = p*q}$. Для генерації відкритого та закритого ключів Аліси, спочатку обирається $\boldsymbol{k}$ різних чисел: $v_1, v_2, \dots, v_k$. Тут кожне v_i є таким, що рівняння $x^2 = v_i (mod\, n)$ має рішення, та існує $v^{-1} mod\, n$. Рядок $v_1, v_2, \dots, v_k$ - є *відкритим ключем.* Далі обчислюються такі найменші значення $s_i = sqrt(v^{-1}) mod\, n$. Рядок $s_1, s_2, \dots, s_k$ є *закритим ключем.*

Основний протокол має наступний вигляд:

1. Аліса обирає випадкове $\boldsymbol{r}$ з інтервалу (1, n-1) і надсилає $x = r^2 (mod\, n)$ Бобу.
2. Боб надсилає Алісі рядок з k випадкових бітів: $b_1, b_2, \dots, b_k$
3. Аліса обчислює число $y = r(s_1^{b_1} s_2^{b_2} \dots s_k^{b_k}) mod\, n$ і надсилає результат Бобу. (Вона перемножує значення s_i, які відповідають $b_i = 1$).
4. Боб перевіряє умову $x = y^2 (v_1^{b_1} v_2^{b_2} \dots v_k^{b_k}) \, mod\, n$

Протокол повторюється t разів, доки Боб не переконається, що Аліса знає числа$s_1, s_2, \dots, s_k$.

Імовірність, що Алісі вдасться обдурити Боба t разів, дорівнює $\frac{1}{2^{kt}}$. Автори протоколу рекомендують допускати ймовірність шахрайства $\frac{1}{2^{20}}$ і рекомендують для цього значення t = 4, k = 5. В даній роботі, ми відслідкуємо точність роботи протоколу і оцінимо час його виконання в залежності від цих параметрів.

Існують деякі поліпшення основної версії протоколу, що дозволяють зменшити складність взаємодій між учасниками або вбудувати в схему ідентифікаційні дані.

Схему ідентифікації Фейге-Фіата-Шаміра можна легко перетворити в схему підпису.

2.2.3 Приклад роботи протоколу

Розглянемо роботу цього протоколу на невеликих числах.

Якщо n = 35 (тут обрані прості множники - 5 і 7), то квадратичними залишками можуть бути наступні числа:

1. $x^2 \equiv 1 \pmod{35}$. Маємо такі розв'язки x: 1; 6; 29; 34.
2. $x^2 \equiv 4 \pmod{35}$. Маємо такі розв'язки x: 2; 12; 23; 33.
3. $x^2 \equiv 9 \pmod{35}$. Маємо такі розв'язки x: 3; 17; 18; 32.
4. $x^2 \equiv 11 \pmod{35}$. Маємо такі розв'язки x: 9; 16; 19; 26.
5. $x^2 \equiv 14 \pmod{35}$. Маємо такі розв'язки x: 7; 28.
6. $x^2 \equiv 15 \pmod{35}$. Маємо такі розв'язки x: 15; 20.
7. $x^2 \equiv 16 \pmod{35}$. Маємо такі розв'язки x: 4; 11; 24; 31.
8. $x^2 \equiv 21 \pmod{35}$. Маємо такі розв'язки x: 14; 21.
9. $x^2 \equiv 25 \pmod{35}$. Маємо такі розв'язки x: 5; 30.

10. $x^2 \equiv 29 \pmod{35}$. Маємо такі розв'язки x: 8; 13; 22; 27.

11. $x^2 \equiv 30 \pmod{35}$. Маємо такі розв'язки x: 10; 25.

Зворотні значення по модулю 35 та їх квадратне корені наведені в наступній таблиці:

Таблиця 2.2.3

v	v^{-1}	$x = sqrt(v^{-1})$
1	1	1
4	9	3
9	4	2
11	16	4
16	11	9
29	29	8

Зверніть увагу, що у чисел 14, 15, 21, 25 та 30 немає зворотних значень за модулем 35, оскільки вони є взаємно простими з числом 35.

Це має сенс, оскільки таких чисел загалом має бути $\frac{(5-1)*(7-1)}{4} = 6$

Отже, Аліса отримує відкритий ключ, що складається з k = 4 значень: {4, 11, 16, 29}. Відповідним закритим ключем є {3, 4, 9, 8}. Розглянемо один раунд протоколу.

1. Аліса обирає випадкове число r = 16, обчислює $16^2 \bmod 35 = 11$ і посилає це значення Бобу.
2. Боб надсилає Алісі рядок випадкових бітів: {1, 1, 0, 1}.
3. Аліса обчислює число $y = 16(3^1 * 4^1 * 19^0 * 8^1) mod\ 35 = 31$ і надсилає його Бобу.

4. Боб перевіряє умову $31^2(4^1 * 11^1 * 16^0 * 29^1) \, mod \, 35 = 11.$

Аліса та Боб повторюють цей протокол t разів, щоразу з новим випадковим числом r, доки Боб не буде задоволений, щоб довіряти Алісі.

Невеликі числа, подібні до використаних у прикладі, не забезпечують реальної безпеки. Але якщо довжина числа n дорівнює 512 і більше біт, то Боб нічого не зможе дізнатися про закритий ключ Аліси, крім того, що Аліса справді його знає.

2.2.4 Покращення роботи протоколу

У протокол можна вбудувати ідентифікаційні дані. Нехай I - це двійковий рядок, що представляє ідентифікатор Аліси: ім'я, адреса, номер соціального страхування, розмір взуття, улюблений фрукт та іншу особисту інформацію.

Використовуємо односторонню геш-функцію H(x) для обчислення H(I, j), де j - невелике число, додане до рядка I. Знайдемо набір чисел j, для яких H(I, j) являє собою квадратичні лишки за модулем n. Ці значення H(I, j) стають рядком $v_1, v_2, \ldots, v_k$ (числа j не повинні бути квадратичними лишками). Тепер відкритим ключем Аліси є рядок I і список чисел j. Аліса посилає рядок I та список чисел j Бобу перед першим кроком протоколу (в якості альтернативи Боб може завантажити ці значення з якоїсь відкритої дошки оголошень). Віктор генерує рядок $v_1, v_2, \ldots, v_k$ за значеннями геш-функції H(I, j).

Тепер, після успішного завершення протоколу, Боб буде переконаний, що Трент, якому відоме розкладання модуля на множники, сертифікував зв'язок між рядком I та Алісою, надавши їй квадратні корені v_i, отримані з рядка I.

Фейге, Фіат та Шамір додали наступні зауваження:

- Для неідеальних геш-функцій можна порадити рандомізувати рядок I, додаючи до нього довгий випадковий рядок R. Цей рядок обирається арбітром і відкривається Бобу разом із рядком I.

- У типових реалізаціях число k має бути від 1 до 18. Великі значення k можуть скоротити час і знизити складності зв'язку, зменшуючи кількість раундів.
- Довжина числа n має бути не менше 512 бітів. (Звичайно, з часу розробки протоколу в області факторизації досягнуто значних успіхів.)
- Якщо кожен користувач вибере власне число n і опублікує його у файлі відкритих ключів, можна обійтися і без арбітра. Однак такий RSA-подібний варіант робить схему набагато менш зручною.

2.3. Використані алгоритми

В описаному протоколі використанні наступні алгоритми:

1. Тест Міллера-Рабіна перевірки числа на простоту.
2. Символ Лежандра.
3. Алгоритм Чіполли.
4. Розширений алгоритм Евкліда.

2.3.1. Тест Міллера-Рабіна на простоту

Тест Міллера-Рабіна на простоту грунтується на тому, що для простого числа не існує тривіальних дискретних квадратних коренів з одиниці.

З рівності:

$x^2 - 1 = 0 \pmod p$

Отримаємо еквівалентну до неї:

$(x - 1) * (x + 1) = 0 \pmod p$

Оскільки $\boldsymbol{p}$ просте, то один з множників ділиться на $\boldsymbol{p}$. Таким чином довели наведене вище твердження.

На цій лемі базується алгоритм Міллера-Рабіна.

Нехай маємо просте ***n***. Тоді при n > 2, ***n–1*** - непарне число, запишемо його у вигляді n – 1 = 2^t * q, де ***q*** деяке непарне число.

Тоді маємо, що для, будь-якого a, виконується, одна з умов.

1) a^q = 1 (mod n)

2) Для r >= 0 і r < t: a^r = -1 (mod n)

Це є наслідком малої теореми Ферма і твердження вище. Справді, маємо, якщо ***n*** – просте то, для довільного ***a*** виконується: a^{n-1} = 1 (mod n).

Послідовно розкладаючи на різницю квадратів отримаємо необхідне твердження.

Нехай маємо деяке ***n*** і деяке число ***a*** для якого одна з умов 1) або 2) не виконується. Тоді це означає, що ***n*** є складеним, а число ***a*** для нього буде називатися свідком його складеності.

На жаль, наразі не існує коректного і швидкого способу генерувати свідків складеності числа. Теорема Рабіна зазначає, що для будь-якого складеного непарного числа існує не менше ніж ¾ свідків складеності. З цього ми й отримуємо тест на простоту. Послідовність дій наступна: генеруємо випадкове число, яке є меншим за дане і перевіряємо для нього умови 1) і 2). Після k кроків перевірки умов, ймовірність того, що ми цього довели складеність числа рівна $1 - (1/4)^k$.

Таким чином отримується ефективний ймовірнісний алгоритм перевірки числа на простоту.

2.3.2. Символ Лежандра

Символом Лежандра називається мультиплікативна функція, яка широко використовується в теорії чисел. Названа на честь французького математика Адрієна-Марі Лежандра.

Визначення: Нехай a деяке ціле число і p просте число. Символ Лежандра визначається таким чином:

- $\frac{a}{p} = 0$, якщо a ділиться на p.
- $\frac{a}{p} = 1$, якщо a є квадратним лишком за модулем p.
- $\frac{a}{p} =$ -1, якщо a є квадратним нелишком за модулем p.

Символ Лежандра використовуємо для генерації відкритого ключа v.

2.3.3. Алгоритм Чіполли

Алгоритм Чіполли - це техніка вирішення конгруентного рівняння виду:

$$x^2 \equiv n \ (mod\ p)$$

Де x,$n \in F_p$, так що n буде квадратом числа x, і де p є непарним простим числом. Тут F_p означає кінцеве поле з p елементами $\{0, 1, \dots, p-1\}$. Алгоритм названий на честь італійського математика Мікеле Чіполли(Cipolla), який відкрив метод у 1907 році.

Алгоритм:

Вхід:

- p - просте число
- $n \in F_p$ - квадрат числа.

Вихід:

Необхідно знайти число $x \in F_p$, таке що $x^2 \equiv n \ (mod\ p)$

1. Знаходимо число $a \in F_p$, таке що $a^2 - n$ не є квадратом. Конкретного алгоритму для пошуку таких чисел не існує, окрім методу спроб і помилок. Просто вибираємо якесь число a і обчислюємо символ Лежандр, щоб переконатися, що a задовольняє умову. Шанс, що випадкове число a підходить, дорівнює $(p-1)/2p$. Для достатньо великиз p ця

ймовірність прямує до ½, таким чином, очікуване число спроб отримання відповідного a дорівнює 2.

2. Обчислюємо $x = (a + \sqrt{a^2 - n})^{(p+1)/2}$ $F_{p^2} = F_p(\sqrt{a^2 - n})$.

Отримане число x і буде одним з рішень рівняння $x^2 = n$.

Для знайденого рішення x завжди існує друге рішення, що дорівнює -x.

Даний алгоритм застосовуємо для швидкої генерації закритого ключа s.

2.3.4. Розширений алгоритм Евкліда

Розширений алгоритм Евкліда необхідний для того, щоб крім знаходження найбільшого дільника, ще знайти розв'язки рівняння:

$$ax + by = \gcd(a, b)$$

Його використання у протоколі необхідно для знаходження оберненого елемента в кільці по модулю.

Алгоритм.

Якщо b = 0, то НСД дорівнює ***a***(база) , і коефіцієнт ***x*** біля ***a*** - одиниця, а коефіцієнт ***y*** біля b — нуль.

Інакше алгоритм рекурсивно викликає себе з параметрами ***(a mod b, b)***

В результаті отримаємо:

$$d = bx_1 + (a \bmod b)\, y_1.$$

$$d = bx_1 + (a - (a \operatorname{div} b) * b)y_1$$

$$d = ay_1 + b(x_1 - (a \operatorname{div} b)y_1)$$

Звідси, коефіцієнт біля a та b:

$$x = y_1$$

$$y = x_1 - (a \operatorname{div} b) * y_1.$$

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА

3.1. Обрані інструменти реалізації.

Реалізації алгоритму була виконана за допомогою мови програмування C++. Її було обрано з метою продемонструвати високу швидкодію, що є головною умовою вирішення даної задачі.

C++ - мова програмування високого рівня, яка підтримує основні парадигми програмування: процедурну, об'єктно-орієнтовану та узагальнену. Зокрема для роботи було важливим підтримка інкапсуляції, наслідування, програмування за допомогою шаблонів, прозоре перевизначення операторів для власних типів даних, простий механізм виявлення помилок та статичний і динамічний поліморфізм.

Для збірки проекту було використано CMake – кросплатформена система автоматизації збірки програмного забезпечення з вихідного коду.

Генерації великих випадкових чисел була написана за допомогою використання бібліотеки random та типу independent_bits_engine.

Принципи ООП та CMake було використано з метою подальшого використання програми й простого її інтегрування в інший проект, та забезпечення кросплатформеності.

Алгоритм було імплементовано, скомпільовано та протестовано на комп’ютері з операційною системою Ubuntu18.04.5 LTS з параметрами комп’ютера Intel(R) Core(TM) i7 16 Gb RAM.

3.2. Опис програмної реалізації

Проект складається з 4 основних класів:

1. ***BigInt*** – клас для ефективної роботи з багаторозрядними числами. Розмір числа обмежується лише розміром оперативної пам’яті.

Реалізовано 5 основних найпростіших операцій: додавання, віднімання,

множення, цілочисельне ділення та ділення за модулем великих чисел.

Для піднесення в степінь було застосовано алгоритм дискретного перетворення Фур'є, оскільки він є ефективним з точки зору швидкодії множення великих чисел. Інтерфейс класу наведений в додатках.

2. Клас ***Utilities*** – містить необхідні утиліти такі, як тест Міллера-Рабіна на простоту, розширений алгоритм Евкліда, обчислення НСД, пошук оберненого елемента по модулю та обчислення символів Лежандра. Обернений елемент по модулю шукаємо за допомогою розширеного алгоритму Евкліда.

Інтерфейс класу:

```
class Utilities{
static const BigInt gcdEx(BigInt a, BigInt b, BigInt& x, BigInt& y);
static const BigInt findInverse(const BigInt& y, const BigInt& modulo);
static bool isPrime(const BigInt& n);
static const int legendre_symbol(const BigInt& a, const BigInt& p)
};
```

3. ***RNG*** - це клас який генерує випадкові прості числа заданої довжини у бітах, або у межах між двома значеннями. Для генерування випадкових чисел було використано бібліотеку range. Перевірки числа на простоту відбувається за алгоритмом Міллера-Рабіна, з початковою перевіркою на маленькі прості дільники.

Інтерфейс класу:

```
typedef std::independent_bits_engine<std::mt19937, 1, bool> BitGenerator;
typedef std::independent_bits_engine<std::mt19937, 32, uint> IntGenerator;
class RNG{
private:
  BitGenerator bit_generator;
  IntGenerator digit_generator;
public:
  RNG();
```

```
 const BigInt next(int length_in_bits);
 const BigInt next(const BigInt& from, const BigInt& to);
};
```

4. ***FFSProtocol*** – безпосередня реалізація протоколу Фейге-Фіата-Шаміра. Модуль n і число k - публічні поле класу. Основні публічно доступні методи - ***setK***, ***init*** та головний метод ***sendAndVerify***. Також дві структури A та B, що містять основні методи і поля для реалізації протоколу, що відповідають діям Аліси і Боба відповідно.

Інтерфейс класу:

```
class FFSProtocol {
   BigInt n;
   int k;
   A userA;
   B userB;
public:
   void init(BigInt n);
   void setK(int k){
        this->k = k;
   }
   bool sendAndVerify();
};
```

3.3. Запуск роботи програми

```
    std::cout << "p = " << p.toStr() << "\nq = " << q.toStr() << "\n";

    auto start : time_point<...> =  std::chrono::steady_clock::now();
    bool resultAuth = ffsProtocol.sendAndVerify();
    auto end : time_point<...> =  std::chrono::steady_clock::now();
    auto elapsed : duration<...> =  std::chrono::duration_cast< std::chrono::milliseconds>( d: end - start);
    cout << "Time for authentication: " << elapsed.count() << "\n";
    cout << (resultAuth ? "Success authentication\n" : "Authentication failed\n");
    return 0;
}

main
```

```
ffs
"/home/polli/University/ Diploma/cryptography-master/cmake-build-debug/ffs"
p = 904531807
q = 678685477
Time for authentication: 272
Success authentication

Process finished with exit code 0
```

РОЗДІЛ 4. АНАЛІЗ РЕЗУЛЬТАТІВ

Результати роботи ffs- протоколу в залежності від значення k наведено у Таблиці 1, що нижче. Програмна реалізація була протестована на комп'ютері з параметрами Ubuntu 18.04.5 LTS Intel(R) Core(TM) i7 16 Gb RAM.

Таблиця 4.1

k	Час верифікації у млс
1	502
2	669
3	752
4	876
5	928

Таблиця 4.2, показує кількість успішних верифікацій для різних параметрів t при фіксованому значенні рекомендованого автомари параметру k = 5.

Таблиця 4.2

t	Відсоток успішності верифікацій
1	99.16%
2	99.48%
3	99.92%
4	100%

З Таблиці 4.1 видно, що швидкість роботи алгоритму прямо пропорційно залежить від довжини відкритого ключа, що очевидно, з огляду на реалізацію, адже навіть попри швидкий алгоритм реалізації операції множення для великих чисел, алгоритм все одно вимагає відчутних втрат по часу. Звісно, швидкість роботи алгоритму у фізичному плані часу, а саме мілісекунд, напряму залежить від програмної реалізації всіх операцій на кожному кроці алгоритму. Оскільки в даній роботі використовувались виключно власноруч написані бібліотеки, то швидкість роботи протоколу не є найкращою з можливих, адже також варто зважати на пряму залежність від середовища програмування, мови, процесора та інших факторів.

З Таблиці 4.2 можемо підтвердити теоретичну оцінку похибки про ймовірність того що Алісі вдасться обдурити Боба t разів, дорівнює $\frac{1}{2^{kt}}$.

ВИСНОВКИ

Сучасні технології, що виконують критично важливі завдання, відзначаються великою різноманітністю типів взаємодії та експоненціально зростаючою складністю. Прикладами таких систем є різні мережі P2P(Peer-to-peer), "людина-машина"(man-machine) та "пристрій до пристрою"(device-to-device) комплекси, військові та корпоративні коаліційні підрозділи, фінансові системи, операції з криптовалютою, банківські картки, інтернет речей (IoT) та багато інших. Разом з розвитком технологій й поширенням важливості їх використання, спостерігається посилення активності технологічно розроблених кібератак з метою зловмисного вторгнення в роботу таких систем. Саме тому, швидкість та точність процедур верифікації мають особливе значення.
У багатьох випадках встановлення обгрунтованої ідентифікації клієнта вимагає суттєвих зусиль щодо обчислень. Прикладом є відома система автентифікації Kerberos. У критичних випадках встановлення процедури автентифікації може суперечити необхідності негайної відповіді на запит автентифікації. Особливо ця проблема стає актуальною завдяки розповсюдженій появі IoT із сотнями і тисячами взаємодіючих пристроїв.

Таким чином, виникає досить складна проблема. З одного боку, необхідно отримати досить складну, безпечну і, за можливості, математично обґрунтовану ідентифікацію суб'єкта. З іншого боку, це слід робити за мінімальний час.

Відомі загальні інтерактивні та не інтерактивні протоколи ZKP в основному концентруються на доказах загальних NP-relations(задачі, що можна розв'язати недетермінованими алгоритмами за поліноміальний час). Незважаючи на багато вражаючих досягнень у цьому напрямку, вони все ще досить складні у реалізаціях і не дуже придатні для негайної відповіді автентифікації.

У даній роботі було детально розглянуто і реалізовано швидкий протокол автентифікації - схема Фейге-Фіата-Шаміра. Надана обгрунтовані факти

коректності цього алгоритму, покроковий опис реалізації та програмний модуль, який демонструє роботу протоколу, підтверджує його коректність. На практиці було доведено оцінку можливості “обдурити” верифікацію, а також показано пряму залежність ефективності роботи алгоритму від таких параметрів як довжину ключів (публичного та приватного), та кількості ітерацій протоколу. Надано таблицю аналізу швидкості роботи протоколу від описаних вище параметрів.

Висновком даної роботи слугує підтвердження того факту, що застосування протоколу Фейге-Фіата-Шаміра є ефективним при розв’язанні задачі швидкої автентифікації, що будується на основі ключів з відкритим(публічним) доступом. А також наведено детальний аналіз залежності швидкості протоколу від різних факторів та розглянуто важливість ролі криптопротоколів у сучасному світі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ю.І. Горбенко, І.Д. Горбенко. «Інфраструктури відкритих ключів. Електронний цифровий підпис. Теорія та практика». - 2010.- 608 с.
2. Алгоритмы: построение и анализ, 2-е издание / [Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.]. – М. : Вильямс, 2005. – 1296 с.
3. Гулак Г.М., Мухачев В.А., Хорошко В.О., Яремчук Ю.Є. «Основи криптографічного захисту інформації» підручник, Вінниця, ВНТУ, 2012.
4. Лужецький В. А., Кожухівський А. Д., Войтович О. П. «Основи інформаційної безпеки» Міністерство освіти і науки України Вінницький національний технічний університет. Затверджено Вченою радою Вінницького національного технічного університету як навчальний посібник для студентів напрямку підготовки 6.170101 «Інформаційна безпека». Протокол № 3від 29 жовтня 2008 р. Вінниця ВНТУ 2009.
5. Горбенко Ю. І. Інфраструктури відкритих ключів. Електронний цифровий підпис. Теорія та практика : моногр. / Ю. І. Горбенко, І. Д. Горбенко ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки, ЗАТ "Ін-т інформ. технологій". – Харків : Форт, 2010. – 608 с.
6. Посібник з курсу «Компютерна криптографія» призначено для студентів інженерно-технічного факультету ДВНЗ «УжНУ» спеціальності 123- «комп’ютерна інженерія». Укладачі: Гапак О. М. – канд. пед. наук, доцент кафедри комп’ютерних систем та мереж ДВНЗ «УжНУ».
7. Кваліфікаційна наукова праця на правах рукопису Циганкова Оксана Валентинівна УДК 004.9 ДИСЕРТАЦІЯ Методи підвищення швидкодії асиметричних криптосистем з використанням еліптичних кривих у формі Едвардса 05.13.21 системи захисту інформації Інформаційна безпека
8. Прикладная криптография: протоколы, алгоритмы и исходный код на С, 2-е юбилейное издание.
9. УДК 621.391.7 : 336.71 (075.8) «Коалиционные схемы с ключами общего доступа», Анісімов А. В., 26 с.

10. Oded Goldreich and Yair Oren Department of Computer Science: Definitions and Properties of Zero-Knowledge Proof Systems, 22.09.1992.
11. Fiat A., Shamir A. How to Prove Yourself: Practical Solutions to Identification and Signature Problems // Advances in Cryptology — CRYPTO ’86: 6th Annual International Cryptology Conference, Santa Barbara, California, USA, 1986, Proceedings / A. M. Odlyzko — Springer Berlin Heidelberg, 1987. — P. 186–194. — 490 p. — (Lecture Notes in Computer Science; Vol. 263).
12. Черемушкин А.В. Криптографические протоколы. Основные свойства и уязвимости. - М.: Академия, 2009. - 269 с.
13. Feige U., Fiat A., Shamir A. Zero-Knowledge Proofs of Identity // Journal of Cryptology / I. Damgård — Springer-Verlag, 1988. — Vol. 1, Iss. 2. — P. 77–94.
14. Мао В. Современная криптография: Теория и практика — М.: Вильямс, 2005. — 768 с. — ISBN 5-8459-0847-7.
15. Rabin M. O. Probabilistic algorithm for testing primality // J. Number Theor. / D. Goss — Elsevier, 1980. — Vol. 12, Iss. 1. — P. 128–138. — ISSN 0022-314X; 1096-1658.
16. Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ. — 3. — Москва: Вильямс, 2013. — С. 1012-1015. — 1328 с.
17. А. В. Анісімов Алгоритмічна теорія великих чисел. Модулярна арифметика великих чисел, 2001. С. 140-144.
18. International Journal of Open Problems in Computer Science and Mathematics , June 2018 - с. 3- 6.
19. Feige-Fiat-Shamir ZKP Scheme Revisited, July 2010 - с 10-14.
20. WADE TRAPPE, LAWRENCE C. WASHINGTON, Introduction to Cryptography with Coding Theory (Prentice-Hall, Inc., 2003), pp. 231–233.
21. Ryan Henry : Efficient Zero-Knowledge Proofs and Applications (Waterloo, Ontario, Canada, 2014) - pp. 88-118

ДОДАТКИ

Інтерфейс багаторозрядної арифметики:

```
class BigInt {
private:
  BigInt(std::vector<int> bits, bool sign=true);

public:
  static const int base = 10000;
  bool sign = true; // true for >= 0
  std::vector<int> bits;
  BigInt();
  BigInt(std::string s);
  BigInt(int n);

  bool operator > (BigInt const &other) const;
  bool operator >= (BigInt const &other) const;
  bool operator < (BigInt const &other) const;
  bool operator <= (BigInt const &other) const;
  bool operator == (BigInt const &other) const;
  bool operator != (BigInt const &other) const;

  std::string toStr() const;
  std::string toStrBase2(int n) const;
  int firstDigit() const;

  friend const BigInt abs(const BigInt& value);
  friend const BigInt sqrt(const BigInt& a);
  friend const BigInt mul(BigInt a, BigInt b, BigInt m);
  friend const BigInt binPow(BigInt num, BigInt st, BigInt mod);
  friend const BigInt operator - (const BigInt& value);
  friend const BigInt operator + (const BigInt& a, const BigInt& b);
  friend const BigInt operator - (const BigInt& a, const BigInt& b);
  friend const BigInt operator * (const BigInt& a, const BigInt& b);
  friend const BigInt operator / (const BigInt& a, const BigInt& b);
  friend const BigInt operator % (const BigInt& a, const BigInt& b);
```

```
  friend const BigInt operator * (const BigInt& a, int b);
   friend const BigInt pow(const BigInt& a, const BigInt& n, const BigInt&
modulo);

};
```

Безпосередня реалізація протоколу:

Файл ffs.h

```
struct A {
   BigInt r;
   std::vector<BigInt> s;
   std::vector<bool> b;

   void initS(std::vector<BigInt> v, BigInt n) {
       for (auto vi : v) {
           BigInt inv = findInverse(vi, n);
           s.push_back(Crypto::discrete_sqrt(inv, n)) ;
       }
   }

   void receiveB(std::vector<bool> b){
       this->b = b;
   }

   BigInt sendX(BigInt n) {
       this->r = Crypto::generateRNG(8) % (n-1) + 1;
       return pow(r, BigInt(2), n);
   }

   BigInt sendY(int k, BigInt n){
       BigInt res = this->r;
       for(int i = 0; i < k; i++){
           if(b[i]){
               res = res * s[i];
```

```
            }
        }
        return res % n;
    }
};

struct B {
    std::vector<bool> b;
    BigInt x;

    std::vector<BigInt> v;
    void initV(BigInt n, int k) {
        int t = 8;
        for (int i = 0; i < k; i++) {
            BigInt a = Crypto::generateRNG(t);
            while (Crypto::legendre_symbol(BigInt(a), BigInt(n)) == -1) {
                a = Crypto::generateRNG(t);
            }
            v.push_back(a);
        }
    }

    std::vector<bool> sendB(int k){
        for(int i = 0; i < k; i++){
            b.push_back(rand()%2 == 0);
        }
        return b;
    }

    void receiveX(BigInt x){
        this->x = x;
    }

    bool verify(BigInt y, BigInt n){
        BigInt calcX = pow(y, BigInt(2), n);
        for(int i = 0; i < b.size(); i++){
            if(b[i]){
                calcX = (calcX * v[i]) % n;
```

```
            }
        }
        return x == (calcX % n);
    }

};

class FFSProtocol {
    BigInt n;
    int k;
    A userA;
    B userB;

public:
    void init(BigInt n);
    void setK(int k){
        this->k = k;
    }
    bool sendAndVerify();
};
```

Файл ffs.cpp

```
void FFSProtocol::init(BigInt n) {
    this->n = n;
    this->setK(4);
    this->userB.initV(this->n, this->k);
    this->userA.initS(this->userB.v, this->n);
}

bool FFSProtocol::sendAndVerify() {
    this->userB.receiveX(this->userA.sendX(this->n));
    this->userA.receiveB(this->userB.sendB(this->k));
    return this->userB.verify(this->userA.sendY(this->k, this->n), this->n);
}
```

Файл ffsMain.cpp

```
int main(){
   FFSProtocol ffsProtocol;
   int t = 16;
   BigInt p = Crypto::generatePrime(t);
   BigInt q = Crypto::generatePrime(t);
   while(gcd(p,q) != 1){
       q = Crypto::generatePrime(t);
   }
   ffsProtocol.init(p * q);

   std::cout << "p = " << p.toStr() << "\nq = " << q.toStr() << "\n";
   auto start =  std::chrono::steady_clock::now();
   bool resultAuth = ffsProtocol.sendAndVerify();
   auto end =  std::chrono::steady_clock::now();
    auto elapsed =  std::chrono::duration_cast< std::chrono::milliseconds>(end
- start);
   cout << "Time for authentication: " << elapsed.count() << "\n";
      cout  <<  (resultAuth  ?  "Success  authentication\n"  :  "Authentication
failed\n");
   return 0;
}
```

Реалізація розширеного алгоритму Евкліда:

```
const BigInt gcdEx(BigInt a, BigInt b, BigInt& x, BigInt& y){
  if (a == 0){
     x = BigInt(0);
     y = BigInt(1);
     return b;
  }
  BigInt x1, y1, res;
  res = gcdEx(b%a, a, x1, y1);
  y = x1;
  x = y1 - (b / a) * x1;
  return res;
}
```

Символ Лежандра:

```
const int legendre_symbol(const BigInt& a, const BigInt& p){
   return pow(a, (p - 1)/2, p) == BigInt(1) ? 1 : -1;
}
```

Тест Міллера-Рабіна на простоту:

```
bool isPrime(const BigInt &n) {
   if(n == 2) return true;
   if(n%2 == 0) return false;
   BigInt a;
   for(int i = 0; i < S; i++){
       a = rng.next(n-1) + 1;
       if(::witness(a,n)) return false;
   }
   return true;
}

bool witness(const BigInt& a, const BigInt& n){
   BigInt t = 0, u = n-1;
   while(u%2 == 0){
       t = t + 1;
       u = u/2;
   }

   BigInt xPrev = pow(a,u,n), x;

   for(BigInt i = 0; i < t; i = i + 1){
       x = xPrev * xPrev % n;
       if(x == 1 && xPrev != 1 && xPrev != n-1){
           return true;
       }
       xPrev = x;
   }
   return x != 1;
}
```