

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ЗАКЛАД  
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА”

Навчально-науковий інститут фізики, математики та інформаційних  
технологій

Кафедра інформаційних технологій та систем

Татарін Юрій Іванович

**Розробка бібліотеки для шифрування інформації  
з використанням методів AES і RSA.**

Бакалаврська робота  
за напрямом підготовки 123 Комп’ютерна інженерія

Особистий підпис – \_\_\_\_\_ Татарін Ю.І.

Науковий керівник – \_\_\_\_\_  
(підпис)

к.т.н., доц  
Г.А. Могильний

(посада, науковий ступінь,  
наукове звання, ініціали, прізвище)

Зав. кафедри – \_\_\_\_\_  
(підпис)

зав. кафедри ІТС, кандидат  
педагогічних наук, доцент,  
М.А. Семенов

(посада, науковий ступінь,  
наукове звання, ініціали, прізвище)

Полтава– 2023

Міністерство освіти і науки України	
Державний заклад «Луганський національний університет імені Тараса Шевченка»	
Факультет (інститут)	Навчально-науковий інститут фізики, математики та інформаційних технологій (повна назва)
Кафедра	Інформаційних технологій та систем (повна назва)
Освітньо-кваліфікаційний рівень	Бакалавр (код, назва)
Напрямок підготовки	123 - Комп'ютерна інженерія (код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТС  
М.А. Семенов

(підпис)

(ініціали, прізвище)

“ ” 2022 р.

## ЗАВДАННЯ НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ

(прізвище, ім'я, по батькові)	
1. Тема проекту (роботи)	Розробка бібліотеки для шифрування інформації з використанням методів AES і RSA
Керівник кваліфікаційної роботи	Могильний Г.А., к.т.н, доц (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджена наказом по університету	Від
2. Строк подання студентом проекту (роботи)	
3. Вихідні дані до роботи (проекту)	Розроблено бібліотеку для шифрування інформації з використанням методів AES і RSA
(визначаються кількісні або (та) якісні показники, яким повинен відповідати об'єкт розробки)	
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)	ТЕОРЕТИЧНІ ЗАСАДИ РОЗРОБКИ БІБЛІОТЕКИ ДЛЯ ШИФРУВАННЯ ІНФОРМАЦІЇ З ВИКОРИСТАННЯМ МЕТОДІВ AES і RSA, ПРОЕКТУВАННЯ БІБЛІОТЕКИ, РОЗРОБКА БІБЛІОТЕКИ.
(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)	
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)	
6. Консультанти розділів проекту/роботи	

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання „\_\_\_\_\_” \_\_\_\_\_ 2022р.

### КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи )	Примітка
1.	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	До 15 жовтня	
2.	Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень.	Другий тиждень листопада (10 листопада )	
3.	Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником.	До 15 грудня	
4.	Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання.	До 28 січня	
5.	Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи.	Перший тиждень березня	
6.	Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації.	До 31 березня	
7.	Попередній захист роботи на кафедрі	квітень	
8.	Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії	За 10 днів до державної атестації	
9.	Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом.	За 5 днів до державної атестації	

**Студент**

\_\_\_\_\_

підпис

**Ю.І. Татарін**

\_\_\_\_\_

(ініціали, прізвище)

**Керівник проекту (роботи)**

\_\_\_\_\_

підпис

**Г.А. Могильний**

\_\_\_\_\_

(ініціали, прізвище)

## **АНОТАЦІЯ**

**Ю.І. Татарін**

**Тема: Розробка бібліотеки для шифрування інформації з використанням методів AES і RSA.**

**Спеціальність: 123- Комп'ютерна інженерія**

**Установа: ЛНУ імені Тараса Шевченка, 2023 р.**

**Бакалаврська робота містить: 60 с., 12 рис., 21 джерело, 2 додатки.**

**Предмет дослідження – процес розробки бібліотеки для шифрування даних.**

**Мета роботи -** є розробка бібліотеки, яка представлятиме собою набір класів, що надає можливість швидкого та спрощеного використання методів шифрування AES і RSA у середі розробки Android Studio, використовуючи мову програмування Kotlin.

### **Результати роботи.**

В першу чергу у роботі розглянуто питання криптографії і сучасні методи шифрування інформації, файлову систему Android та загальне представлення про бібліотеки. Далі обрано середовище розробки Android Studio та мову програмування Kotlin. Представлено опис використаних бібліотек та основні принципи роботи з файлами. Наступним етапом було розглянуто принципи роботи алгоритмів шифрування AES і RSA. У етапі розробки спроектовано блок-схеми та структуру бібліотеки. У завершення продемонстровано інструкцію користувача.

**Висновки.** В результат роботи представлено бібліотеку для шифрування даних методами AES і RSA .

**Ключові слова.** бібліотека, шифрування, AES, RSA.

## ABSTRACT

**Tatarin U.I.**

**Topic:** Development of a library for encrypting information  
using AES and RSA methods.

**Specialty:** 123- Computer Engineering

**Institution:** *Taras Shevchenko Lviv National University, 2023*

**The bachelor's thesis contains:** 60 pages, 12 pictures, 21 sources, 2  
additional.

**The subject of research** is the process of developing a library for data encryption.

**The aim of the work** is to develop a library that will be a set of classes that allows you to quickly and easily use AES and RSA encryption methods in the Android Studio development environment using the Kotlin programming language.

### **Results of work.**

First of all, the paper considers the issues of cryptography and modern methods of encrypting information, the Android file system and a general idea of libraries. Next, the Android Studio development environment and Kotlin programming language were selected. The description of the used libraries and the basic principles of work with files is presented. The next step was to consider the principles of AES and RSA encryption algorithms. Block diagrams and structure of the library are designed at the development stage. Finally, the user manual is shown.

**Conclusions.** As a result, a library for data encryption using AES and RSA methods is presented.

**Keywords.** library, encryption AES,RSA.

**ІТС.4КІ.0723.01-ВП**  
**ВІДОМІСТЬ ПРОЕКТУ. РОЗРОБКА БІБЛІОТЕКИ ДЛЯ**  
**ШИФРОВАННЯ ІНФОРМАЦІЇ З ВИКОРИСТАННЯМ МЕТОДІВ AES і**  
**RSA.**

[illegible]

					ІТС.4КІ.0723.01-ВП							
Змн.	Арк.	№ докум.	Підпис	Дата								
Розроб.		Татарін Ю.І.			ВІДОМІСТЬ ПРОЕКТУ			Літ.	Арк.	Акрюшів		
Керівник		Могильний Г.А.								1	1	
Реценз.		Козуб Ю.Г.						ЛНУ Кафедра ІТС, Гр.4КІ				
Н. Контр.												
Зав. каф.		Семенов М.А..										

**Міністерство освіти і науки України**  
**Державний заклад «Луганський національний університет**  
**імені Тараса Шевченка»**

Факультет (інститут)

Навчально-науковий інститут фізики,  
математики та інформаційних технологій

(повна назва)

Кафедра

Інформаційних технологій та систем

(повна назва)

(код, назва)

**ТЕХНІЧНЕ ЗАВДАННЯ**

на виконання програмної розробки (ПР):

" Розробка бібліотеки для шифрування інформації  
з використанням методів AES і RSA "

**ІТС.4КІ.0723.02-ТЗ**

**ПОГОДЖЕНО**

Керівник кваліфікаційної роботи

Могильний Г.А

“ ” 2023р

**ВИКОНАВЕЦЬ**

Студент групи 4КІ

Татарін Ю.І.

“ ” 2023р

Полтава— 2023

## ЗМІСТ

<b>ВСТУП.....</b>	<b>9</b>
<b>1. ПРИЗНАЧЕННЯ ПРОДУКЦІЇ .....</b>	<b>Ошибка! Закладка не определена.</b>
<b>2. ТЕХНІЧНІ ХАРАКТЕРИСТИКИ.....</b>	<b>9</b>
<b>3. АЛГОРИТМ РОБОТИ ПРИСТРОЮ .</b>	<b>Ошибка! Закладка не определена.</b>
<b>4. ТЕХНІКО-ЕКОНОМІЧНІ ВИМОГИ ДО КІНЦЕВОГО ПРОДУКТУ</b>	<b>9</b>
<b>5. ВИМОГИ ДО МАТЕРІАЛІВ ТА КОМПЛЕКТУЮЧИХ.....</b>	<b>11</b>
<b>6. ЕТАПИ ВИКОНАННЯ ПР.....</b>	<b>11</b>
<b>7. ПРИЙМАННЯ .....</b>	<b>12</b>
<b>8. ПОРЯДОК ВНЕСЕННЯ ЗМІН ДО ТЕХНІЧНОГО ЗАВДАННЯ, ЩО</b>	
<b>ЗАТВЕРДЖЕНО.....</b>	<b>13</b>

					ІТС.4КІ.0723.02-ТЗ					
Змн.	Арк.	№ докум.	Підпис	Дата	ТЕХНІЧНЕ ЗАВДАННЯ			Літ.	Арк.	Акрушів
Розроб.		Татарін Ю.І.								
Керівник		Могильний Г.А.							2	7
Реценз.		Козуб Ю.Г.						ЛНУ Кафедра ІТС, Гр.4КІ		
Н. Контр.										
Зав. каф.		Семенов М.А.								



## ВСТУП

- 1.1 **Найменування:** Розробка бібліотеки для шифрування інформації з використанням методів AES і RSA.
- 1.2 **Шифр ПР:** ІТС.4КІ.0723
- 1.3 **Підстава до виконання ПР:** Підставою для виконання даної розробки є процес розробки бібліотеки для шифрування інформації методами AES і RSA.
- 1.4 **Терміни розробки:**
- 1.4.1 Початок 10 листопада 2022р.
- 1.4.2 Закінчення 5 червня 2023р.
- 1.5 **Фінансується** за рахунок коштів замовника. Умови фінансування – згідно договору №12/а та протоколу погодження ціни № 12/б.

## 1 ХАРАКТЕРИСТИКА ОБ'ЄКТА

- 1.1. Розроблювана бібліотека повинна виконувати шифрування інформації методами AES і RSA. До складу об'єкта, який створюється має входити:
- 1.1.1. Розроблювана бібліотека.
- 1.1.2. Додаток для демонстрації роботи бібліотеки.
- 1.2. До вхідної інформації відносяться текстові данні та секретний ключ.
- 1.3. До вихідної інформації належать зашифровані данні:

## 2. ПРИЗНАЧЕННЯ ТОВАРІВ

- 2.1. Призначення: Розробка та налаштування бібліотеки, яка виконує шифрування інформації методами AES і RSA.
- 2.2. Основні критерії ефективності.
- 2.2.1. Зручна структура бібліотеки.
- 2.2.1.1. Оператор повинен мати можливість обрати метод шифрування;
- 2.2.1.4. Оператор повинен мати можливість проводити зручний перегляд всієї інформації.
- 2.2.2. Програмний комплекс повинен:
- 2.2.2.1 Виконувати шифрування текстових даних методами AES і RSA.
- 2.2.2.2. Мати можливість дешифрування методами AES і RSA;

					ІТС.4КІ.0723.02-ТЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

2.2.2.3. Мати можливість генерації секретних ключів для представлених методів;

2.2.2.4. Надавати можливість збереження результату у файл

2.3. Основні функції оператора:

2.3.1. Запустити бібліотеку.

2.3.2. Вибрати метод шифрування.

2.3.3. Сгенерувати секретний ключ.

2.3.4. Провести шифрування даних.

2.4. Основною функцією бібліотеки є спрощення виконання шифрування даних.

### **3. ОСНОВНІ ВИМОГИ ДО ПРОГРАМНОГО КОМПЛЕКСУ**

3.1. Загальні вимоги

3.1.1. Бібліотека працює під операційною системою Android;

3.1.2. Вимоги до апаратного забезпечення пристрою - не передбачені і можуть встановлюватися розробником програмного комплексу;

3.1.3. До складу роботи повинен входити jar файл з розробкою;

3.2. Додаткові вимоги

3.2.1. Мова програмування Kotlin.

3.2.2. Вимоги до ліцензійного ПЗ не передбачаються і вирішуються замовником.

3.3. Вимоги до складу і архітектури

3.3.1. Розробник самостійно вибирає склад і виконує розробку архітектури ПР

3.3.2. Особливих умов до складу і архітектури ПР не передбачено.

3.4. Вимоги до якості і надійності

3.4.1. Бібліотека повинен надійно працювати.

3.4.2. Розробник вибирає технічні характеристики пристрою, налаштовує системне програмне забезпечення.

3.4.3. Розробник гарантує роботу бібліотеки без збоїв і перенастроювань.

3.5. Вимоги до експлуатації

					ІТС.4КІ.0723.02-ТЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

### 3.5.1. Розробник використовує пристрій, на якому бібліотека повинна надійно працювати.

#### 4 ТЕХНІКО-ЕКОНОМІЧНІ ВИМОГИ ДО КІНЦЕВОГО ПРОДУКТУ

Вартість робіт по розробці даної ПР визначається згідно договору на розробку. Вартість запропонованих аналогів повинна забезпечити економічну доцільність їх застосування.

## 5 ВИМОГИ ДО МАТЕРІАЛІВ ТА КОМПЛЕКТУЮЧИХ

В процесі розробки програмного комплексу можливе використання програмних засобів Java.

- а. Вимоги до екологічної безпеки під час експлуатації.

Не пред'являются.

- б. Спеціальні вимоги до кінцевого продукту.**

Не пред'являются.

- с. Вимоги до безпеки для населення під час експлуатації продукції.

Не пред'являются.

## 6 ЕТАПИ ВИКОНАННЯ ПР.

Етапи виконання ПР можуть уточнювати згідно календарного плану робіт по узгодженню між замовником та виконавцем

№	Етапи виконання роботи	Термін виконання та обсяг робіт	звітні матеріали
1	Аналіз розробки програмного комплексу та розробка першої версії. Аналіз вимог. Розробка структури. Попереднє тестування		Частковий програмний комплекс на ЕОМ замовника, що виконує всі основні функції та звітна документація п.8.2
2	Коректування структури. Розробка допоміжних функцій. Розробка остаточної версії програмного комплексу та його опрацювання. Тестування		Готовий програмний комплекс на ЕОМ замовника та звітна документація п.8.2

№	Етапи виконання роботи	Термін виконання та обсяг робіт	звітні матеріали
3	Доопрацювання окремих модулів та навчання користувачів. Розробка звітних матеріалів згідно п.8 цього ТЗ		звітні матеріали згідно пункту 8

## 7 ПРИЙМАННЯ

7.1. Необхідні вимоги для впровадження ПР та завершення робіт.

Оцінка результатів розробки і доцільність її продовження здійснюється замовником по представленню наступних матеріалів:

- встановлений програмний комплекс на ЕОМ замовника;
- перелік файлів на резервному носії;
- стислий опис роботи ПР та опис всіх файлів, які необхідні для роботи ПР.
- перелік документів
  - Технічне завдання
  - Пояснювальна записка

7.2. Перелік звітних документів, необхідних для прийняття етапів роботи:

- стислий опис результатів етапу у вигляді анотованого звіту(для 1 та 2 етапів);
- частковий програмний комплекс на ЕОМ замовника згідно календарного плану робіт;
- акт приймання продукції.

Звітні матеріали подаються у вигляді звітів на папері по ГОСТ 7.32-91

7.3. Загальний перелік до приймання звітних документів, макетів, експериментальних зразків.

До приймання пред'являються: акт здачі-приймання продукції, акт впровадження ПР.

7.4.Тестування ПР

					ІТС.4КІ.0723.02-ТЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

Тестування виконується до "Програми та методики тестування", яка розробляється виконавцем та затверджується замовником

## **8 ПОРЯДОК ВНЕСЕННЯ ЗМІН ДО ТЕХНІЧНОГО ЗАВДАННЯ, ЩО ЗАТВЕРДЖЕНО.**

Дане технічне завдання може уточнюватися в процесі розробки ПР при узгодженні сторін з оформленням доповнень до ТЗ.

					ІТС.4КІ.0723.02-ТЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**ДЗ «ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**  
**ІМЕНІ ТАРАСА ШЕВЧЕНКА»**

Навчально-науковий інститут фізики, математики та  
інформаційних технологій

---

(назва факультету, інституту)

Інформаційних технологій та систем

---

(назва кафедри)

**Пояснювальна записка**  
до дипломного проекту (роботи)  
**БАКАЛАВРА**  
(освітньо-кваліфікаційний рівень)

на тему:

Розробка бібліотеки для шифрування інформації  
з використанням методів AES і RSA.

Виконав: студент 4 курсу, групи \_\_\_\_\_  
напряму підготовки (спеціальності)  
123 - Комп'ютерна інженерія  
(шифр і назва напряму підготовки, спеціальності)

\_\_\_\_\_ Татарін Ю.І.  
(прізвище та ініціали)

Керівник \_\_\_\_\_ Могильний Г.А.  
(прізвище та ініціали)

Рецензент \_\_\_\_\_ Козуб Ю.Г.  
(прізвище та ініціали)

## ЗМІСТ

<b>Вступ .....</b>	<b>16</b>
<b>РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ РОЗРОБКИ БІБЛІОТЕКИ ДЛЯ ШИФРУВАННЯ ІНФОРМАЦІЇ З ВИКОРИСТАННЯМ МЕТОДІВ AES і RSA .....</b>	<b>18</b>
<b>1.1 Аналіз літератури.....</b>	<b>18</b>
<b>1.2 Шифрування інформації – криптографія .....</b>	<b>18</b>
<b>1.3 Методи шифрування .....</b>	<b>19</b>
1.3.1 Метод AES .....	21
1.3.2 Метод RSA .....	22
1.2.3 Метод MD5 .....	23
<b>1.4 Бібліотеки програмування .....</b>	<b>23</b>
<b>1.5. Файлова система Android.....</b>	<b>25</b>
1.5.1. Файлова система Ext4 .....	27
1.5.2 Файлова система Flash-Friendly File System.....	31
<b>1.6 Висновки до першого розділу.....</b>	<b>33</b>
<b>РОЗДІЛ 2. ПРОЕКТУВАННЯ БІБЛІОТЕКИ .....</b>	<b>34</b>
<b>2.1. Обґрунтування вибору мови програмування .....</b>	<b>34</b>
<b>2.2. Середовище розробника.....</b>	<b>34</b>
<b>2.3. Опис стандартних бібліотек.....</b>	<b>35</b>
2.3.1 Бібліотека Base64 .....	35
2.3.2 Бібліотека Java Cryptography Extension.....	36
2.3.3 Бібліотека Java.IO.....	37
<b>2.4 Робота з файлами .....</b>	<b>39</b>
<b>2.5 Алгоритм AES.....</b>	<b>40</b>
<b>2.6 Алгоритм RSA.....</b>	<b>42</b>
<b>2.8 Висновки до другого розділу .....</b>	<b>45</b>
<b>РОЗДІЛ 3 РОЗРОБКА БІБЛІОТЕКИ .....</b>	<b>46</b>
<b>3.1. Розробка блок схем. ....</b>	<b>46</b>
<b>3.2. Розробка бібліотеки .....</b>	<b>47</b>
3.3 Додаток для демонстрації роботи бібліотеки.....	50
<b>3.4 Інструкція користувача .....</b>	<b>51</b>
<b>3.5 Висновки до третього розділу.....</b>	<b>52</b>
<b>СПИСОК ВИКОРИСТОВАНИХ ДЖЕРЕЛ .....</b>	<b>54</b>
<b>ДОДАТОК А Код бібліотеки шифрування .....</b>	<b>56</b>
<b>ДОДАТОК Б Код програми для демонстрації .....</b>	<b>60</b>

					<i>ІТС.4КІ.0723.02-ПЗ</i>		
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>	<i>Зміст</i>		
<i>Розроб.</i>		<i>Татарін Ю.І.</i>					
<i>Керівник</i>		<i>Могильний Г.А.</i>					
<i>Реценз.</i>		<i>Козуб Ю.Г.</i>					
<i>Н. Контр.</i>							
<i>Зав. каф.</i>		<i>Семенов М.А.</i>			<i>ЛНУ</i> <i>Кафедра ІТС, Гр.4КІ</i>		
						<i>Лім.</i>	<i>Арк.</i>
							<i>Аркушів</i>

## Вступ

У сучасному світі питання безпеки інформації в інтернеті стає однією з найважливіших задач. Паролі, месенджери, пошта - все це знаходиться під ударом хакерів. Тому щоб захиститися використовується криптографія, основною метою якої є шифрування інформації.

Для реалізації шифрування існують різноманітні алгоритми, кожен з яких відповідає своєму запиту та має свій функціонал. В цієї роботі будуть використовуватися методи AES і RSA, як дуже швидкі і в той самий час достатньо прості в реалізації, що дуже актуально для мобільних систем.

Але сама по собі реалізація цих алгоритмів є дуже громізка і важка для сприйняття. Саме цьому розробляються бібліотеки, які скривають за собою реалізацію та максимально спрощують результат для подальшої роботи.

Тому **метою роботи** стала розробка бібліотеки, яка представлятиме собою набір класів, що надає можливість швидкого та спрощеного використання методів шифрування AES і RSA для мови Kotlin.

Бібліотека, що буде представлена у цьому проекті є об'єднанням реалізації методів шифрування AES і RSA для операційних систем Android, що буде виконувати функції шифрування та дешифрування текстових даних з можливістю роботи з файлами. Що робить застосування актуальним з практичною метою, для шифрування та передачі реальних даних. Також особливістю бібліотеки буде можливість генерації і ручного вводу секретних ключів, низькі вимоги до ресурсів смартфона та простота використання, завдяки чому кожен зможе з легкістю почати користуватися бібліотекою.

**Об'єктом дослідження** є методи шифрування даних.

**Предметом дослідження** є процес розробки бібліотеки для шифрування даних.

Так для реалізації проекту головними **завданнями роботи** бачимо:

- Дослідження літератури за темами: «шифрування даних», «алгоритми шифрування», «файли», «бібліотеки».

					ІТС.4КІ.0723.03-ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		



- Розгляд алгоритмів шифрування AES і RSA та особливості роботи з файлами .

- Розробка структури бібліотеки та її елементів.

- Розробка бібліотеки для шифрування даних.

В першому розділі роботи розглянуто питання криптографії та сучасні методи шифрування даних. Представлено опис використаних у операційній системі Andoid файлових систем та загальне положення про бібліотеки програмування.

Другий розділ присвячено розгляду роботи алгоритмів AES і RSA та роботи з файлами. Крім того у другому розділі обрано мову програмування, середовище розробки та є опис використаних бібліотек.

У третьому розділі описано розробку бібліотеки та представлено блок-схеми бібліотеки. Також у цьому розділі розроблено додаток для демонстрації роботи бібліотеки та представлена інструкція користування та демонстрація роботи бібліотеки.

					ІТС.4КІ.0723.03-ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

# РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ РОЗРОБКИ БІБЛІОТЕКИ ДЛЯ ШИФРОВАННЯ ІНФОРМАЦІЇ З ВИКОРИСТАННЯМ МЕТОДІВ AES і RSA

## 1.1 Аналіз літератури

Під час розробки бібліотеки було розглянуто основні поняття з даної теми, а саме: криптографія, сучасні методи шифрування та робота з файлами.

## 1.2 Шифрування інформації – криптографія

Криптографія (від грец. Κρυπτός «прихований» + γράφω «пишу») - наука про методи забезпечення конфіденційності (неможливості прочитання інформації стороннім), цілісності даних (неможливості непомітного зміни інформації), аутентифікації (перевірки справжності авторства чи інших властивостей об'єкта) , а також неможливості відмови від авторства. [1]

Основним видом криптографічного перетворення інформації у світі комп'ютерних систем є шифрування і дешифрування. Шифрування - це перетворення інформації з відкритої форми в закриту (зашифровану). Існує і зворотний процес - розшифрування.[2]

За багатовікову історію використання шифрування інформації людством винайдено безліч методів шифрування, або шифрів. Методом шифрування, або шифром, називається сукупність оборотних перетворень відкритої інформації в закриту відповідно до алгоритму шифрування. Поява ЕОМ і КС ініціювало процес розробки нових шифрів, які враховують можливості використання ЕОМ як для шифрування / розшифрування інформації, так і для атак на шифр. Атака на шифр (криптоаналіз) - це процес розшифрування закритої інформації без знання ключа і, можливо, за відсутності відомостей про алгоритм шифрування.

					ІТС.4КІ.04.19.03-ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Татарін Ю.І.			РОЗДІЛ І ТЕОРЕТИЧНІ ЗАСАДИ  РОЗРОБКИ БІБЛІОТЕКИ ДЛЯ ШИФРОВАННЯ ІНФОРМАЦІЇ З  ВИКОРИСТАННЯМ МЕТОДІВ AES і RSA			Літ.	Арк.	Акрушів	
Керівник		Могильний Г.А.								5	50
Реценз.		Козуб Ю.Г.						ЛНУ Кафедра ІТС, Гр.4КІ			
Н. Контр.											
Зав. каф.		Семенов М.А..									

Крипостійкість шифру є основним показником його ефективності. Вона вимірюється часом або вартістю засобів, необхідних криптоаналітику для отримання вихідної інформації по шифр тексту, за умови, що йому невідомий ключ.[3]

Зберегти в секреті широко використовуваний алгоритм шифрування практично неможливо. Тому алгоритм не повинен мати прихованих слабких місць, якими могли б скористатися криптоаналітики. Якщо ця умова виконується, то криптостойкість шифру визначається довжиною ключа, тому що єдиний шлях розкриття зашифрованої інформації - перебір комбінацій ключа і виконання алгоритму розшифрування. Таким чином, час і кошти, що витрачаються на криптоаналіз, залежать від довжини ключа і складності алгоритму шифрування.[2]

### 1.3 Методи шифрування

Методи шифрування можна поділити на три категорії: симетричні, асиметричні та метод хешированія.

Симетричне шифрування - метод, де для шифрування та дешифрування використовуються один й той самий криптографічний ключ, завдяки чому він також відомий як метод- секретний ключ.[4] Схема симетричного шифрування представлена на рисунку 1.1.



Рисунок 1.1 Схема використання методу симетричного шифрування

У сучасних системах, ключ зазвичай являє собою рядок даних, які отримані з надійного пароля, або з абсолютно випадкового джерела. Він

подається в симетричне шифрування програмного забезпечення, яке використовує його, щоб засекретити вхідні дані. Скремблирование даних досягається за допомогою симетричного алгоритму шифрування, такі як Стандарт шифрування даних (DES), розширений стандарт шифрування (AES), або міжнародний алгоритм шифрування даних (IDEA).[3]

Асиметричний метод шифрування працює аналогічно симетричному, в тому, що він використовує ключ для кодування переданих повідомлень. Однак, замість того, щоб використовувати той же ключ, для розшифровки цього повідомлення він використовує зовсім інший. Схема використання асиметричного шифрування - рисунок 1.2.

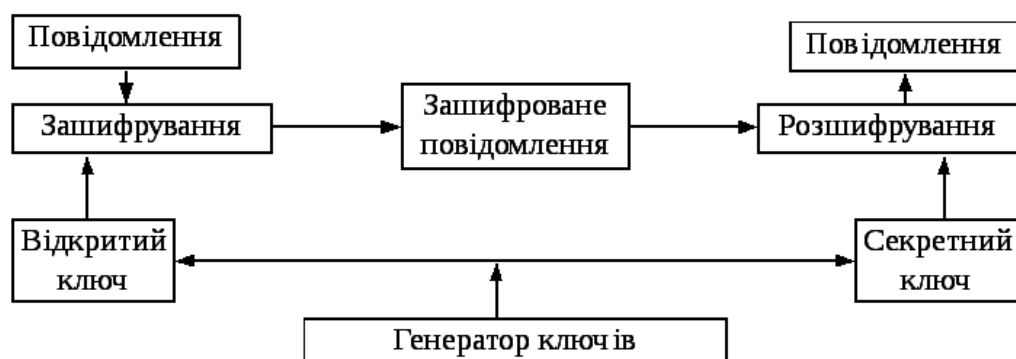


Рисунок 1.2 Схема використання методу асиметричного шифрування

Ключ, використовуваний для кодування доступний кожному і всім користувачам мережі. Як такий він відомий як «публічний» ключ. З іншого боку, ключ, який використовується для розшифровки, зберігається в таємниці, і призначений для використання в приватному порядку самим користувачем. Отже, він відомий як «приватний» ключ. Асиметричне шифрування також відомо, як шифрування з відкритим ключем.

Оскільки, при такому способі, секретний ключ, необхідний для розшифровки повідомлення не повинен передаватися кожен раз, і він звичайно відомий тільки користувачеві (приймача), ймовірність того, що хакер зможе розшифрувати повідомлення значно нижче.

У якості прикладу можливо визначити метод RSA.[4]

Методика хеширования використовує алгоритм, відомий як хеш-функція для генерації спеціального рядка з наведених даних, відомих як хеш. Цей хеш має такі властивості: одні й ті ж дані завжди справляє той же самий хеш, неможливо генерувати вихідні дані з хеша поодиночі.

Недоцільно пробувати різні комбінації вхідних даних, щоб спробувати генерувати той же самий хеш. Таким чином, основна відмінність між хешем і двома іншими формами шифрування даних полягає в тому, що, як тільки дані зашифровані (хешировані), вони не можуть бути отримані назад в первозданному вигляді (розшифровані). Цей факт гарантує, що навіть якщо хакер отримує на руки хеш, це буде марно для нього, так як він не зможе розшифрувати зміст повідомлення.[5]

### 1.3.1 Метод AES

AES - скорочення Advanced Encryption Standard (переклад з англійської мови. Advanced Encryption Standard). Після відкриття в 1997 році Національним інститутом стандартів і технологій США (NIST) програми розвитку AES відбулися 3 етапи міжнародного змагання. Новий стандарт повинен був мати:[6]

- стабільність не менше, ніж у 3DES;
- швидкість шифрування, що перевищує швидкість 3DES;
- прозора структура;
- ефективна реалізація на платформі Pentium Pro;
- ефективне впровадження обладнання.

Переможцями конкурсу стали бельгійці Іоан Дамен та Вінсент Рейман з алгоритмом RIJNDAEL (читайте Рейн-дано з перших листів авторів). Стандарт набув чинності в 2002 р. AES - симетричний ітеративний блок-алгоритм; AES - не зашифрований Feistel, заснований на принципах нової мережі перестановок. Він має нову архітектуру SQUARE (SQUARE), яка характеризується:

- представлення зашифрованого блоку як двовимірного байтового масиву;

					ІТС.4КІ.0723.03-ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

- шифрування за один раунд всього блоку даних (байтово-орієнтована структура);

- виконання криптографічних перетворень, як на окремих байтах масиву, так і на його рядках і стовпцях. Це забезпечує одночасне розповсюдження даних у двох напрямках - у рядках та стовпцях.

Архітектурі SQUARE притаманні, крім шифру AES (RIJNDAEL), шифрів SQUARE (його назва дала назву всій архітектурі), CRYPTON (один з кандидатів на AES). Друге місце у змаганні AES посів ще один SP-код, SERPENT. Мабуть, найближчим часом домінуватимуть SP-мережі та, зокрема, архітектура SQUARE.[5]

#### Загальна характеристика АЕС

- AES шифрує та розшифровує 128-бітні блоки
- дані.
- AES дозволяє використовувати три різні клавіші довжиною 128,
- 192 або 256 біт (залежно від довжини ключа версії шифру
- позначають AES-128, AES-192 або AES-256).
- Кількість раундів шифрування залежить від розміру ключа:
- довжина 128 біт - 10 патронів;
- довжина 192 біта - 12 раундів;
- довжина 256 біт - 14 патронів.
- Усі раунди, крім останнього, однакові. [6].

#### Переваги:

- висока швидкість
- низькі потреби в ресурсах
- простота реалізації [6]

### 1.3.2 Метод RSA

RSA - криптографічний метод шифрування із введенням відкритого ключа, заснований на обчислювальній складності проблеми факторизації

					ІТС.4КІ.0723.03-ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

величезних цілих чисел. Криптосистема RSA стала 1 системою, яка використовується як для шифрування, так і для цифрових підписів.

Метод RSA враховує, що відправлене кодоване повідомлення насправді має можливість читати одержувачем і тільки їм. Цей метод використовує 2 клавіші - щирі та секретні. Цей метод приємний навіть тоді, коли величезна кількість предметів (N) містить знати схему «все-з-усім». У випадку симетричної схеми шифрування будь-який із суб'єктів зобов'язаний доставити власні ключі іншим учасникам обміну, і загальна кількість використаних ключів буде досить великою з великим почуттям Н. Використання асиметричного метод вимагає лише загальна кількість клавіш точно однакова Н. [28]

Він виділяється величезною кількістю часу в обмін на завищену безпеку

### 1.2.3 Метод MD5

MD5 (Message Digest 5) - це 128-бітний метод хешування, розроблений доктором Рональдом Л. Рівестом з Массачусетського технологічного інституту (MIT) у 1991 році. [6]

Призначений для створення "відбитків" або "дайджестів" повідомлень довільної довжини.

Прийшов на заміну MD4, який був недосконалим. Працює на 32-бітних машинах.

Враховуючи MD5, неможливо відновити вхідне повідомлення, наприклад, оскільки 1 MD5 може відповідати різним повідомленням.

Використовується для перевірки достовірності опублікованих повідомлень шляхом зіставлення дайджесту повідомлення з опублікованими. Ця операція називається "хеш-перевірка". [8]

## 1.4 Бібліотеки програмування

Бібліотека - організована на машинному носії сукупність комп'ютерних програм, доступ до яких здійснюється за їхніми іменами (або індексами). Бібліотека. прискорює і полегшує процес створення користувальницької

					ІТС.4КІ.0723.03-ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

програми, дозволяючи застосовувати в якості складових частин вже готові функції (підпрограми), для чого достатньо вказати ім'я відповідної функції. Завдяки бібліотекам процес програмування стає більш технологічним, а сама призначена для користувача програма легше для сприйняття.

Бібліотеки. можуть складатися з об'єктних модулів (довічних файлів, отриманих в результаті трансляції), або текстів програм на вихідному мові програмування. Бібліотека об'єктних модулів - це, як правило, файл, що складається з заголовка і послідовно розташованих об'єктних модулів. У заголовку міститься список всіх модулів із зазначенням зміщення кожного з них від початку бібліотеки. Коли головна програма прямо або побічно викликає бібліотечну функцію, редактор зв'язків знаходить визначення цієї функції в заголовку бібліотеки. і додає відповідний модуль до головної програми, створюючи єдиний виконуваний файл.

У мультизадачній операційній системі така статична компоновка неефективна, оскільки призводить до неекономного використання оперативної пам'яті. Наприклад, якщо дві одночасно виконуються програми викликають одну функцію, то в пам'яті будуть перебувати дві копії цієї функції.

Динамічно зв'язані (компоновані) бібліотеки, на відміну від статичних, підключаються до основної програми безпосередньо під час звернення до бібліотечної функції. Під час компонування програми в файлі створюється посилання на бібліотеку., яке використовується для динамічного включення відповідної бібліотеки в адресний простір викликає процесу при його виконанні. Таким чином, з'являється можливість одночасного використання (поділу) динамічних бібліотек декількома процесами. Використання динамічних бібліотек скорочує обсяг програми, але дещо ускладнює установку додатка, оскільки воно потребує ще й у відповідних бібліотечних файлах.

Програмісти можуть створити свої власні бібліотеки. Однак для виконання часто повторюваних в різних програмах операцій (напр., Введення-

					ІТС.4КІ.0723.03-ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		



виведення, управління пам'яттю, обчислення елементарних математичних. Функцій і ін.), Як правило, застосовують функції зі стандартних бібліотек, які поставляються разом з транслятором мови або з операційною системою.[7]

### 1.5. Файлова система Android

Android OS - популярна і універсально використовувана операційна система для смартфонів. Хоча з боку користувача це може здатися простим і легким у використанні, файлові системи Android, як правило, досить складні, і багато користувачів чухають голову від задоволення. Давайте тепер докладно розглянемо файлові системи і те, що вони можуть запропонувати користувачам.

Ця інформативна стаття призначена для людей, які думають про розробку ROM, додатків і багатьох інших речей в операційній системі Android. Не гаючи більше хвилини, давайте почнемо з докладного вивчення файлової системи Android. Ми не будемо просто називати файлові системи, ми також дамо вам коротке пояснення конкретної файлової системи.

Флеш-пам'ять Android Файлова система

#### 1. exFAT

Спочатку створена Microsoft для флеш-пам'яті, файлова система exFAT не є частиною стандартного ядра Linux. Тим не менш, він все ще забезпечує підтримку пристроїв Android в деяких випадках. Він позначає розширену таблицю розміщення файлів.

#### 2. F2FS

Користувачі смартфонів Samsung обов'язково стикалися з файлової системою такого типу, якщо вони деякий час користувалися смартфоном. F2FS означає Flash-Friendly File System, яка є файлової системою Linux з відкритим вихідним кодом. Це було введено Samsung 4 роки тому, в 2012 році.

#### 3. JFFS2

Це означає, що файлова система Journal Flash версії 2. Це файлова система за замовчуванням для ядер Android Open Source Project. Ця версія

					ІТС.4КІ.0723.03-ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

файлової системи Android існує з моменту випуску ОС Android Ice Cream Sandwich. З тих пір JFFS2 замінив JFFS.

#### 4. YAFFS2

Це розшифровується як Another Another Flash File System version 2. Забавно, як може звучати назва, насправді це серйозний бізнес! Деякий час він не був частиною AOSP і рідко зустрічається в смартфонах Android. Проте, він, як правило, час від часу з'являється кілька разів.

#### Медіа файлова система Android

##### 1. EXT2 / EXT3 / EXT4

Ext, що позначає файлові системи EXTended, є стандартом для файлової системи Linux. Останнім з них є EXT4, який тепер замінює файлові системи YAFFS2 і JFFS2 на смартфонах Android.

##### 2. MSDOS

Microsoft Disk Operating System, як відомо, є одним з найстаріших імен в світі операційних систем, і вона допомагає запускати файлові системи FAT 12, FAT 16 і FAT 32.

##### 3. VFAT

Розширення для вищезазначених файлових систем FAT 12, FAT 16 і FAT 32, VFAT - це модуль ядра, який можна побачити поруч з модулем MSDOS. Зовнішні SD-карти, які допомагають розширити обсяг пам'яті, відформатовані з використанням VFAT.

#### Псевдо файлові системи

##### 1. CGroup

Cgroup означає Control Group. Це псевдофайлова система, яка забезпечує доступ до різних параметрів ядра і їх значення. Групи C дуже важливі для файлової системи Android, оскільки ОС Android використовує ці групи управління для обліку й керування процесором.

##### 2. Rootfs

Rootfs діє як точка монтування, і це мінімальна файлова система. Він розташований в точці монтування «/».

					ІТС.4КІ.0723.03-ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

### 3. Procsfs

Зазвичай зустрічається в каталозі / proc. У файлової системи procfs є файли, які демонструють діючі дані ядра. Іноді ця файлова система також відображає ряд структур даних ядра. Ці каталоги номерів відображають ідентифікатори процесів для всіх виконуються в даний момент задач.

### 4. Sysfs

Зазвичай монтується в каталозі / sys. Файлова система sysfs допомагає ядру ідентифікувати пристрої. Після ідентифікації нового пристрою ядро створює об'єкт в каталозі sys / module /. В папці / sys / зберігаються різні інші елементи, які допомагають ядру взаємодіяти з різними файловими системами Android.

### 5. Tmpfs

Тимчасова файлова система, tmpfs зазвичай монтується в каталог / dev. Дані про це губляться при перезавантаженні пристрою.

#### 1.5.1. Файлова система Ext4

Файлова система EXT4 в першу чергу підвищує продуктивність, надійність і ємність. Для підвищення надійності були додані метадані та контрольні суми журналу. Для задоволення різних критично важливих вимог тимчасові мітки файлової системи були поліпшені з додаванням інтервалів аж до наносекунд. Додавання двох старших бітів в поле мітки часу відкладає проблему 2038 року до 2446 року - принаймні для файлових систем EXT4.[12]

У EXT4 розподіл даних було змінено з фіксованих блоків на екстенти. Ступінь описується його початковим і кінцевим місцем на жорсткому диску. Це дозволяє описувати дуже довгі, фізично суміжні файли в одному записі покажчика inode, що може значно скоротити створено додаткове ознакування міста, необхідних для опису розташування всіх даних у великих файлах. Інші стратегії розподілу були реалізовані в EXT4 для подальшого зменшення фрагментації.

EXT4 зменшує фрагментацію, розподіляючи новостворені файли по диску, щоб вони не групувалися в одному місці на початку диска, як це робили

					ІТС.4КІ.0723.03-ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

багато ранніх файлові системи ПК. Алгоритми розміщення файлів намагаються розподілити файли настільки рівномірно, наскільки це можливо, між групами циліндрів і, коли необхідна фрагментація, зберігати екстенти переривчастих файлів якомога ближче до інших в тому ж файлі, щоб звести до мінімуму пошук головки і затримку обертання. наскільки це можливо. Додаткові стратегії використовуються для попереднього виділення додаткового дискового простору при створенні нового файлу або при розширенні існуючого файлу. Це допомагає гарантувати, що розширення файлу не призведе до його фрагментації автоматично. Нові файли ніколи не розміщуються відразу після існуючих файлів, що також запобігає фрагментації існуючих файлів.

Крім фактичного розташування даних на диску, EXT4 використовує функціональні стратегії, такі як відкладене виділення, щоб дозволити файлової системи збирати всі дані, що записуються на диск, перш ніж виділяти для нього місце. Це може підвищити ймовірність того, що простір даних буде суміжним.

Старі файлові системи EXT, такі як EXT2 і EXT3, можуть бути змонтовані як EXT4, щоб домогтися невеликого приросту продуктивності. На жаль, для цього необхідно відключити деякі важливі нові функції EXT4, тому я рекомендую проти цього.

EXT4 є файловою системою за замовчуванням для Fedora починаючи з Fedora 14. Файлова система EXT3 може бути оновлена до EXT4 з використанням процедури, описаної в документації Fedora, проте її продуктивність, однак постраждає через залишкових структур метаданих EXT3. Кращий спосіб поновлення до EXT4 з EXT3 - створити резервну копію всіх даних в цільовому розділі файлової системи, використовувати команду `mkfs`, щоб записати порожню файлову систему EXT4 в розділ, а потім відновити всі дані з резервної копії.[13]

- Особливості ext4:

					ІТС.4КІ.0723.03-ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

- Використання розширень. У FS ext3 адреса даних проводилася традиційним способом - в блоках. Цей спосіб адресації став менш ефективним із збільшенням розміру файлу. Розширення дозволяють адресувати велику кількість (до 128 Мб) послідовних блоків однією ручкою. До чотирьох покажчиків на розширення можна розмістити безпосередньо в inode, цього достатньо для малих та середніх файлів.

- 48-бітні номери блоків. При розмірі блоку в 4 Кб це дозволяє адресувати до одного екбабайта ( $248 \cdot (4 \text{ КБ}) = 248 \cdot (22) \cdot (210) \text{ В} = 260 \text{ В} = 1 \text{ ЕБ}$ ).

- Розподіл мультиблоків. FS зберігає інформацію не тільки про розташування вільних блоків, але і про кількість вільних блоків, розташованих один за одним. Вибираючи місце, ФС знаходить фрагмент, в який можна записати дані без фрагментації. Використання цієї методики дозволяє знизити рівень фрагментації ФС.

- Затримка розподілу. Блоки виділяються для зберігання вмісту файлу безпосередньо перед записом на диск (наприклад, при виклику синхронізації), а не під час виклику запису. Через це блоки можна вибирати не один за одним, а групами, що в свою чергу мінімізує фрагментацію та прискорює процес вибору блоків. З іншого боку, ризик втрати даних збільшується у разі раптового відключення електроенергії.

- Піднято обмеження на кількість підкаталогів. ext3 (без використання патчів) дозволяють розміщувати в одній директорії не більше 32 000 підкаталогів (до 65,535, якщо ви змінюєте константи ядра).

- Забезпечення резервування входів під час створення каталогу (резервування inodes у англомовних каталогах) Під час створення каталогу зарезервується кілька входів. Згодом при створенні файлів у цьому каталозі спочатку використовуються зарезервовані вставки, і якщо їх не залишилося, виконується звичайна процедура вибору інода.

					ІТС.4КІ.0723.03-ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

- Розмір вводу. Розмір inode (за замовчуванням) збільшено з 128 (ext3) до 256 байт. Це дозволило реалізувати наступне.

- Тимчасові місця з наносекундною точністю (англійські наносекундні часові позначки). Точність часових позначок, що зберігаються в inode, збільшується до наносекунд. Діапазон значень також розширюється: у ext3 верхня межа збереженого часу становила 18 січня 2038 року, а у ext4 - 25 квітня 2514 року.

- версія Inode Поле, яке зберігає номер версії, з'являється в структурі inode. Число можна збільшувати щоразу, коли зміна inode змінюється, якщо FS встановлюється за допомогою параметра `iversion`. Це використовується мережевою файловою системою NFS версії 4 (NFSv4) для відстеження змін файлів.

- Зберігання розширених атрибутів у структурі інода (Extended attributes (EA) inode). Продуктивність файлової системи підвищується за рахунок виключення операції пошуку атрибутів деінде на диску.

- Додаткові атрибути включають списки контролю доступу ACL, атрибути SELinux тощо. Атрибути, для яких недостатньо місця в структурі inode, зберігаються в окремому блоці 4 Кб. Планується зняти це обмеження в майбутньому.

- Розрахунок контрольних сум на записи журналів. Використання контрольних сум для транзакцій журналу дозволяє швидко знаходити та (іноді) виправляти системні помилки після відмови (при перевірці цілісності).

- Стійке попереднє розміщення. Для виділення місця на ext2 та ext3 з додатком довелося записати у файл нульові байти. У ext4 є можливість бронювання блоків. Тепер вам не потрібно витрачати час на написання нульових байтів, просто використовуйте системний виклик `Fallocate`. `fallocate` вибирає блоки для файлу та встановлює для них прапор "заповненим нулем байтів". Під час читання з файлу програма отримуватиме нульові байти (як і під час читання розрідженого файлу). Під час запису у файл прапор

					ІТС.4КІ.0723.03-ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

"заповнений нульовими байтами" буде видалений. На відміну від розріджених файлів, запис у такий файл ніколи не буде перервано через брак вільного місця.

- Дефрагментація без відключення (англ. Онлайн дефрагментація). Дефрагментація виконується утилітою e4defrag, яка включена в пакет e2fsprogs з 2011 року.

- неініціалізовані групи. Ця функція ще не реалізована і розроблена для прискорення перевірки цілісності fsck утиліти FS. Блоки, позначені як невикористані, перевірятимуться групами, а детальна перевірка проводитиметься лише у тому випадку, якщо групова перевірка виявила пошкодження. Передбачається, що час перевірки складе від 1/2 до 1/10 поточного в залежності від способу розміщення даних.[14]

### 1.5.2 Файлова система Flash-Friendly File System

F2FS (англ. Flash-Friendly File System) - файлова система, спрямована на реалізацію на флеш-пам'яті, обсяг якої оптимізований для використання з SSD, картами пам'яті (eMMC / SD) та інтегрований у всі види споживчих пристроїв із флеш-чіпом. Творцем вважається Кім Яггек (Hangul :) від Samsung. Стартовий код F2FS не було закрито компанією в жовтні 2012 року, що згодом було переглянуто інженерами Samsung з урахуванням коментарів компанії. Паралельно розробляється пакет f2fs-tools, який містить набір утиліт для служби розділів F2FS (mkfs.f2fs, fsck.f2fs). F2FS розроблений спеціально для специфіки флеш-пам'яті і забезпечує такі функції як постійний час доступу та вузьке число ресурсів циклів перезапису даних.

F2FS заснований на дизайні файлової системи, структурованої журналом (LFS). Основні компоненти зрубленого дизайну:

- запит під час копіювання, наприклад, про те, що дані завжди записуються у раніше не використовуваному місці.

- Цей вільний простір відрізняється великими "регіонами", дешевими для нескінченного запису. Коли кількість цих "регіонів" мініатюризується,

					ІТС.4КІ.0723.03-ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

раніше збережені дані зберігаються від декількох "діапазонів" до 1 вільних, що формується більш вільними "регіонами". Цей процес називається очищенням і вважається одним із найбільш трудомістких у структурі журналів.

#### Особливості файлової системи F2FS:

- Збереження структур даних співпрацює у формі журналу, а при оновленні інформації використовується пристрій копіювання (CoW), у якому дані не перезаписуються при зміні, а зберігаються у абсолютно новому просторі . Спосіб роботи F2FS істотно продовжить термін служби флеш-накопичувачів, оскільки файлова система забезпечує внутрішню геометрію розташування мікросхем у носіях та роботу контролера.

- Для зменшення зносу флешки дані про ємність розподіляються рівномірно, зводячи до мінімуму перезапис у ті самі блоки. Для цього використовується метод послідовного заповнення накопичувача, в якому свіжі дані завжди записуються лише в області, ці згодом раніше записані дані, незалежно від можливої фрагментації. Згодом заслуги кінця лідерів рекордів приходять з самого початку, займаючи, відповідно до здібностей, звільнені блоки. Щоб не вирішувати інциденти з логікою контролера приводу, F2FS забезпечує специфіку шару FTL (Flash Translate Layer), що робить подібне завдання на багатьох накопичувачах для рівномірного заповнення.

- Для забезпечення єдності використовується модель з фіксацією точки та ймовірністю відкоту в разі проблем.

- Для привчання F2FS до різних типів флеш-накопичувачів, які відрізняються власними особливостями залежно від внутрішньої геометрії та схеми управління, для контролю структури розподілу даних у розділі враховується широкий спектр характеристик.

- Розділ F2FS складається з 2 Мб частин, розділи згруповані в секції, які в свою чергу з'єднані в зони.

- Процес розробки F2FS передбачає труднощі попередніх спеціалізованих файлових систем на основі журналів та докладає всіх зусиль

					ІТС.4КІ.0723.03-ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		



для усунення популярних дефектів, таких як високе використання пам'яті та найвища вартість очисних операцій.

- Файлова система F2FS захищена від "ефекту снігової кулі", який виникає при використанні мандрівних дерев: в історії, коли свіжі компоненти утворюються замість перезапису (зміни кількості блоків), для дерев, у яких батьківський вузол посилається на дочірні вузли, змінюється вузол призводить до перестановки всіх вищих вузлів.

- Індекси з інформацією про розподіл даних зберігаються в оперативній пам'яті для прискорення операцій під час роботи.

- Для виконання операцій на складах сміття продається спеціальний колектор, який фактично виробляється у фоновому режимі в умовах простою системи.

- F2FS підтримує класичну схему обмеження доступу UNIX, таку як такі вдосконалені механізми, як xattr та POSIX ACL. [16]

## 1.6 Висновки до першого розділу

У першому розділі розглянуто основні сучасні методи шифрування та обрано для використання у роботі AES і RSA. Тому, що ці методи є одними за найрозповсюдженими при використанні у мобільних системах.

Також розглянуто файлову систему Android, де використовуються ідентичні Linux системи.

					ІТС.4КІ.0723.03-ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

## РОЗДІЛ 2. ПРОЕКТУВАННЯ БІБЛІОТЕКИ

### 2.1. Обґрунтування вибору мови програмування

Для розробки програмного продукту була обрана мова програмування Kotlin, а саме розробка мобільного додатку у середовищі Android Studio. Основними причинами чого є наступне.

Мова програмування Kotlin в водночас підтримує як об'єктно-орієнтоване, так і процедурне програмування та володіє відкритим вихідним кодом.[8]

Відрізняється гнучкістю і простотою синтаксису. Прості функції і структури можна оголосити одним рядком. Геттери і сеттери задаються за лаштунками для інтероперабельності з Java-кодом. Додавання data-анотації до класу активує автоматичну генерацію різних шаблонів.

Також Kotlin не допускає виникнення NullPointerException, видаючи помилку компіляції.

Крім цього не мало важним є 100% сумісність з мовою Java.[10]

### 2.2. Середовище розробника

У якості середовища розробки було обрано Android Studio.

Android Studio - це інтегроване середовище розробки (IDE) від Google, яка надає розробникам інструменти, необхідні для створення додатків для платформи Android.

Основа Android Studio на програмному забезпеченні IntelliJ IDEA використовуючи мову програмування Java під девізом «ця середовище розробки створюється спеціально для Android».

					ІТС.4КІ.0723.03-ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата	РОЗДІЛ І ПРОЕКТУВАННЯ БІБЛІОТЕКИ		
Розроб.		Татарін Ю.І.					
Керівник		Могильний Г.А.					
Реценз.		Козуб Ю.Г.					
Н. Контр.							
Зав. каф.		Семенов М.А..					
					Літ.	Арк.	Акрушів
						24	50
					ЛНУ Кафедра ІТС, Гр.4КІ		

Android Studio підтримує Google App Engine для швидкої інтеграції хмарних додатків з новими API і функціями. Також підтримує API-інтерфейсів Google Play та інших сервісів Google. Має підтримку Android починаючи з версії 1.6.

Серед переваг даною середо розробки можна виділити наступні:

- Наявність безлічі помічників і шаблонів для загальних елементів програмування для Android
- Можливість перегляду на різних версія Android та з різними налаштуваннями.
- Велика гнучкість процесу розробки завдяки системи автоматичної зборки Gradle.[11]

### 2.3. Опис стандартних бібліотек

При розробці додатку були використані наступні бібліотеки: Base64, Java Cryptography Extension, Java.IO

#### 2.3.1 Бібліотека Base64

Кодування Base64 історично з'явилися у відповідь на обмеження широко використовуваного стандарту передачі електронної пошти, який був спрямований на передачу лише текстових даних. Після цього було відмічено необхідність надсилання нетекстової інформації електронною поштою: зображення, аудіо, відео тощо. Однак здійснити перехід до свіжого поштового стереотипу було б непросто, оскільки існувало велика кількість більшості різних незалежних реалізацій текстового стандарту. У зв'язку з даними було запропоновано кодувати довільні дані у формі слова.

Оскільки безперервна кількість нормальних набраних символів (26 малих регістрів + 26 малих регістрів + 10 цифр + невелика кількість пунктуацій та знавців. Символів) набагато менша, ніж діапазон значень випадкових байтів даних (0 ~ 255), Метод Base64 заснований на перетворенні 8-бітної послідовності в послідовність компонентів найменшої ємності. З практичної

					ІТС.4КІ.0723.03-ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

точки зору зручніше вибирати бітову глибину компонентів початкового порядку, щоб було використано саму найбільш ймовірну кількість дозволених друкованих символів. Доповнюючи велику кількість букв і цифр (всього 62 елементи) двома випадковими знаками пунктуації (зазвичай '+' і '/'), ми отримуємо сприятливий код, будь-який символ якого, безумовно, можна поставити у співвідношенні 6 -бітове двійкове число ( $2^6 = 64$ ).

В результаті ми отримуємо ефективний метод кодування: послідовність введення 8-бітових даних поділяється на блоки по 3 байти (24 біта), будь-який блок ділиться на 4 6-бітні речовини ( $24/6 = 4$ ). Далі будь-яку 6-бітну речовину ставлять у співвідношенні разів друкованих символів коду. В результаті виходить послідовність друкованих символів, дозволена для передачі по текстовому каналу. Якщо вихідні дані є несправними для формування 3-байтового блоку, вони надходять таким чином: якщо 1-й байт несправний, вихідний блок доповнюється спеціальною емблемою, що не входить до провідного алфавіту коду (зазвичай '='), з 2 байтами несправними - 2-2 з цими ознаками. [2]

При розшифровці виконується зворотна процедура - послідовність коду "розрізається" на блоки по 4 символи, потім будь-який з них ставиться у співвідношенні 6-бітного значення, а набутий 24-бітний рядок "розрізається" на блоки 8 біт.

### 2.3.2 Бібліотека Java Cryptography Extension

Java Cryptography Extension (JCE) - офіційно випущено звичайне розширення для платформи Java та частку Java Cryptography Architecture (JCA). Він надає набір пакетів, які гарантують рамки та реалізацію цих криптографічних завдань, таких як шифрування та дешифрування даних, генерування та тестування справжності керуючих ключів, а також реалізація алгоритмів коду автентифікації повідомлення (MAC) [1].

Криптографічне розширення Java ґрунтується на тому, що, власне, є криптографічною архітектурою Java (JCA) і розглядається як частка JCA. Справа в тому, що закони Південної Америки забороняють вивозити певні

					ІТС.4КІ.0723.03-ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

типи криптографічного програмного забезпечення (зокрема, симетричне шифрування та розробку основного спільного матеріалу) за межі США та Канади або дозволяють експортувати усіченими ключами. Звичайні класи JCA містять лише хеш-функції, генератори ключів та інші функції, які не підпадають під це обмеження і мають усі шанси безпечно експортуватися як частина платформи Java 2. Однак потрібні сильні методи шифрування, які підпадають під обмеження експорту криптографії в США, з інших джерел, в результаті чого я доставляю їх у вигляді окремого продукту - JCE.

Криптографічне розширення Java розроблено таким чином, що інші криптографічні бібліотеки можуть бути підключені для забезпечення нових алгоритмів без проблем [3]

Він використовується для впровадження алгоритмів шифрування AES та RSA в наданій роботі.

### 2.3.3 Бібліотека Java.ІО.

Більша частина програм обмінюється даними із зовнішнім світом. Усі види мережевих додатків роблять це абсолютно - вони передають та отримують інформацію з інших комп'ютерів та спеціальних пристроїв, підключених до мережі. Як виявилось, можна буквально однаково уявити підстановки даними між пристроями всередині однієї машини. Так, наприклад, програма має можливість зчитувати дані з клавіатури і записувати їх у файл, або, як варіант, читати дані з файлу і відображати їх на екрані. Таким чином, пристрій, з якого він може читати інформацію, має всі шанси бути самим різним - файл, клавіатура, об'єднання мережі тощо. Це ж стосується пристроїв виводу - він має можливість бути файлом, екраном монітора, принтер, злиття мережі тощо. Зрештою, всі дані в комп'ютерній системі під час обробки передаються з пристроїв введення на пристрої виводу.[21]

Як правило, частка обчислювальної платформи, яка відповідає за заміну даних, наприклад, називається системою вводу-виводу. На Java він представлений пакетом java.io (вхід / вихід). Реалізація системи введення /

					ІТС.4КІ.0723.03-ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

виведення ускладнюється не тільки широким колом джерел та одержувачів даних, але й різними форматами передачі інформації. Можна поширити його у бінарному поданні, символічному чи текстовому, використовуючи деяке кодування (лише для російської мови їх понад 4 штуки) або трансляцію кількості у всіх видах подань. Доступ до інформації може знадобитися як послідовно (наприклад, читання HTML-сторінки), наприклад, випадковий (складна робота з кількома частинами 1-го файлу). Найчастіше буферизацію використовують для підвищення продуктивності.

У Java для опису роботи введення / виводу використовується спеціальне поняття потоку даних. Потік даних пов'язаний з деяким джерелом або приймачем даних, здатних приймати або надавати інформацію. Відповідно до цього, струмені поділяються на вхідні дані - зчитування та вихідні - передають (записують) дані. Впровадження концепції потоку дозволяє відокремити провідну логіку програми, обмінюється інформацією хоча б з деякими пристроями аналогічним чином, від операцій низького рівня з цими пристроями введення / виводу. . [17]

На Яві струмені природно представлені об'єктами. Вони окреслюють свої класи як один і складають провідну частку пакету java.io. Вони дуже різноманітні і відповідають за різні показники. Усі класи розділені на 2 частини - одні будуть реалізовувати введення даних, інші - виводити дані. (Рисунок 1.1)

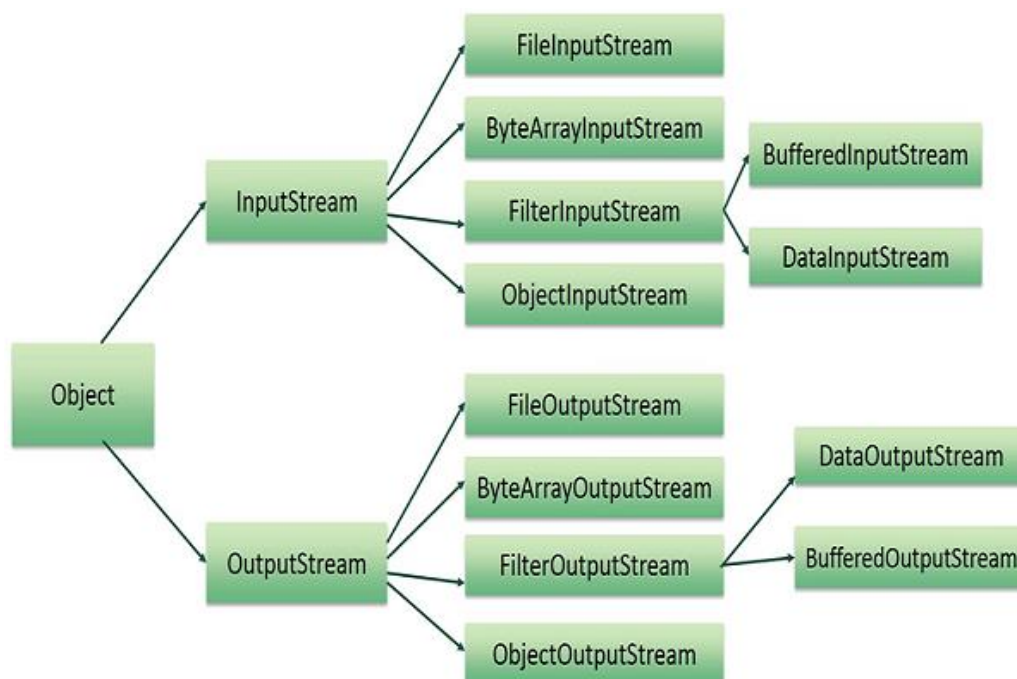


Рисунок 1.1 – Ієрархія класів пакету Java.IO

## 2.4 Робота з файлами

Робота з файлами буде виконуватися за допомогою класу File бібліотеки java.io (розділ 2.3.3).

Відповідно до назви, він призначений для різних операцій з файлами; об'єкт цього типу відповідає якому-небудь реальному файлу, найчастіше знаходиться на жорсткому диску. Для створення файлу використовується спеціальна функція-конструктор: File (inputName) або File (outputName) в прикладі. Якщо в аргументі конструктора вказано тільки ім'я файлу - пошук файлу відбувається в поточній директорії, а якщо аргумент утримує також шлях до файлу - то в директорії, зазначеній цим шляхом. Специфіка конструктора полягає в тому, що його ім'я збігається з типом об'єкта, яку він створює, і він має результат відповідного типу. Більш детально ми поговоримо про конструкторах в наступному уроці.

Обмін даними з файлом може відбуватися в режимі читання або в режимі запису. У режимі читання інформації, задане ім'я має відповідати вже існуючого файлу. Один із способів отримання інформації з файлу - виклик

функції `file.readLine ()`. Результат виклику - список рядків, з яких складається файл. Кожен `String` в цьому списку відповідає одному рядку файлу, рядка файлу поділяються символом «повернення каретки» і / або «новий рядок».

У режимі запису інформації, задане ім'я може не відповідати існуючому файлу - в цьому випадку він буде створений. Для запису інформації, необхідно створити один з об'єктів, що забезпечують таку можливість. У прикладі, таким об'єктом є `val writer = File (outputName).bufferedWriter ()` - тобто необхідно викликати функцію `bufferedWriter ()` на одержувача, відповідному вихідного файлу. Як видно з тексту прикладу, `writer` (письменник) має функції `writer.newLine ()` (додавання в файл нового рядка), `writer.write (string)` (додавання в файл заданого рядка) і `writer.close ()` (закриття письменника, виконується суворо ПІСЛЯ виконання всіх інших дій і фіксує підсумкове стан файлу).[16]

## 2.5 Алгоритм AES.

В якості першого алгоритму у роботі представлено AES (розділ 1.2.1), принцип роботи якого описано далі.

Для шифрування потрібне джерело. Розмір ключа у методі AES 128 біти, як правило, джерело приймається як матриця 4x4 байти.

На початку процесу шифрування вхідні дані діляться на блоки по 16 байт або 128 біт. Якщо абсолютне значення даних не кратне 16 байтам, дані доповнюються об'ємом, кратним 16 байт. Блок даних у методі AES називається станом і зазвичай розглядається у вигляді матриці 4x4 b (див. Рис. 2.1). Операція шифрування кожного блоку даних виконується незалежно від вмісту інших блоків. Коли шифрування блоку завершено, матриця заповнюється відповідною частиною даних і процес повторюється. Завдяки незалежності шифрування 1-го блоку від іншого, процес шифрування цілком піддається паралелізації. [19]

					ІТС.4КІ.0723.03-ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		



*State array*

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Рисунок 2.1 – Масив «state».

Будь-який блок шифрується в ряд меж - раундів. Схема перетворення криптовалюти має можливість записуватися так.

1. Розширення ключа KeyExpansion.

2. Оригінальна операція - AddRoundKey - підсумовування за допомогою головного ключа.

3. 9 раундів по 4 кроки.

3.1. SubBytes - підміна державних байтів на обмінній таблиці.

3.2. ShiftRows - повторне зміщення ліній стану.

3.3. MixColumns - перестановка стовпців стану.

3.4. AddRoundKey - підсумовування круглої клавіші.

4. Заключний 10 тур

4.1. SubBytes - підміна державних байтів на обмінній таблиці.

4.2. ShiftRows - повторне зміщення ліній стану.

4.3. AddRoundKey - підсумовування кругло

(Див. Рисунок 2.2)

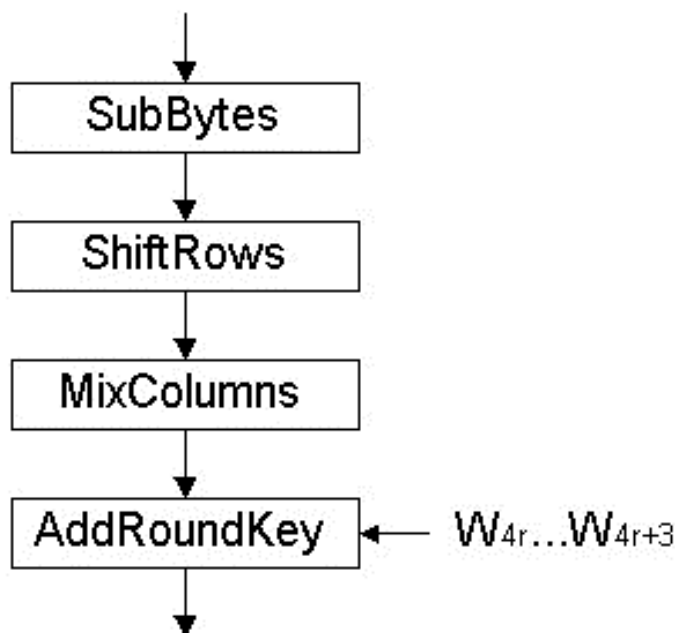


Рисунок 2.2 Раунди шифрування

## 2.6 Алгоритм RSA

Другим методом став RSA (розділ 1.2.2), як більш безпечний метод, який працює в майбутньому.

Повідомлення подається у формі числа  $M$ . Шифрування виконується за підтримки публічної функції  $f(M)$ , і лише отримувач знає, як виконати операцію  $f^{-1}$ . Одержувач вибирає 2 великі прості (прості) числа  $p$  і  $q$ , які він готує приховано. Він констатує  $n = pq$  і число  $d$ ,  $c(d, p-1) = (d, q-1) = 1$  (одним із ймовірних методів виконання цієї умови є вибір  $d$  більше  $p/2$  і  $q/2$ ). Шифрування здійснюється за формулою:

$$f(M) \equiv M^d \pmod{n},$$

де  $M$  і  $f(M)$  обидва  $< n$ . Як було показано, його можна обчислити за розумний час, у тому числі якщо  $M$ ,  $d$  і  $n$  мають досить велику кількість символів. Одержувач обчислює  $M$  на основі  $M^d$ , застосовуючи свої знання про  $p$  і  $q$ . Відповідно до наслідку 6, якщо

$$dc \equiv 1 \pmod{(p-1)}, \text{ протягом цього часу } (M^d)^c \equiv M \pmod{p}.$$

Справжнє слово  $M$  отримується одержувачем із зашифрованого  $F(M)$  методом перестановки:  $M = (F(M))^e \pmod{pq}$ . Тут як вихідне слово, наприклад, і зашифроване розглядаються як подовжені двійкові числа.

Подібно до  $(Md)^e \equiv M \pmod{pq}$ , якщо  $d \equiv (q-1)^{-1} \pmod{e}$  задовольняє цим 2 умовам, якщо  $c \equiv (p-1)^{-1} \pmod{e}$ . Аксиома 1 говорить, адже ми можемо вирішити  $e = x$ , коли  $x$  вважається висновком рівняння  $dx \pmod{(p-1)(q-1)} = 1$ .

Наприклад, коли ми ділимо  $(Md)^e - M$  на  $p$  і  $q$ , ми також ділимо його на  $pq$ , тому ми можемо кваліфікувати  $M$ , враховуючи  $Md$ , обчислюючи його значення в потужності  $e$  і визначаючи залишок ділення на  $pq$ . Для збереження секретності, в принципі, так, щоб з урахуванням  $n$  було неможливо визначити  $p$  і  $q$ . Якщо  $n$  має 100 цифр, вибір шифру пов'язаний з пошуком  $\sim 10^{50}$  пісень. Надана невідповідність досліджується вже понад 100 років. Алгоритм RSA запатентований (20 вересня 1983 р., Діє до 2000 р.).

На теоретичному рівні можна фактично уявити, що цілком ймовірно, що операція  $f^{-1}$  виконується без обчислення  $p$  і  $q$ . Але в будь-якому випадку ця проблема не є звичайною, і творці вважають її важкою факторизувати.

Припустимо, насправді, що у нас є зашифроване слово  $f(M)$  та вихідне слово  $M$ , і ми хочемо знайти значення  $p$  і  $q$ . Неважко продемонструвати, що насправді цих початкових даних для вирішення проблеми недостатньо - вам потрібна аристократія, все можливе значення  $M_i$ .

Як приклад застосування методу RSA, розглянемо відповідну історію. Вибираємо 2 прості величини  $p = 7$ ;  $q = 17$  (на практиці ці великі великі кількості на одну довшу). У цьому випадку  $n = p * q$  точно такий же, як 119. Тепер нам потрібно вибрати  $e$ , вибираємо  $e = 5$ . Наступний крок пов'язаний з утворенням числа  $d$  так, що  $d * e \equiv 1 \pmod{(p-1)(q-1)}$ .  $d = 77$  (використовується розширений евклідовий метод).  $d$  - класифіковане джерело, а  $e$  і  $n$  характеризують щире джерело. Нехай слово, яке нам потрібно зашифрувати, побачимо  $M = 19$ .  $C = M \pmod{n}$ . Ми отримуємо зашифроване слово  $C = 66$ . Це "слово" має можливість надсилатись відповідному одержувачу. Одержувач

					ІТС.4КІ.0723.03-ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

розшифровує набуте повідомлення, використовуючи  $M = C d \bmod n$  та  $C = 66$ .  
Результат  $M = 19$  [15]

На практиці відкриті ключі мають усі шанси помістити їх у спеціальну базу даних. Якщо необхідно надіслати зашифроване повідомлення партнеру, можна оформити запит на його відкритий ключ на початку. Отримавши його, можна запустити програму шифрування та надіслати результат її роботи одержувачу. Використання відкритих ключів також ґрунтується, наприклад, на так званому електричному підписі, який дозволяє однозначно ідентифікувати відправника. Такі засоби можна використовувати для запобігання будь-яких коригувань у повідомленні на шляху від відправника до одержувача. Високошвидкісні апаратні 512-бітні модулі можуть забезпечити швидкість шифрування 64 кбіт в секунду. Готуються ІМС, здатні виконувати такі операції зі швидкістю 1 Мб / с. Розумний вибір параметра  $e$  дозволяє значно прискорити реалізацію алгоритму. [18] Схема шифрування представлена на рисунку 2.2.

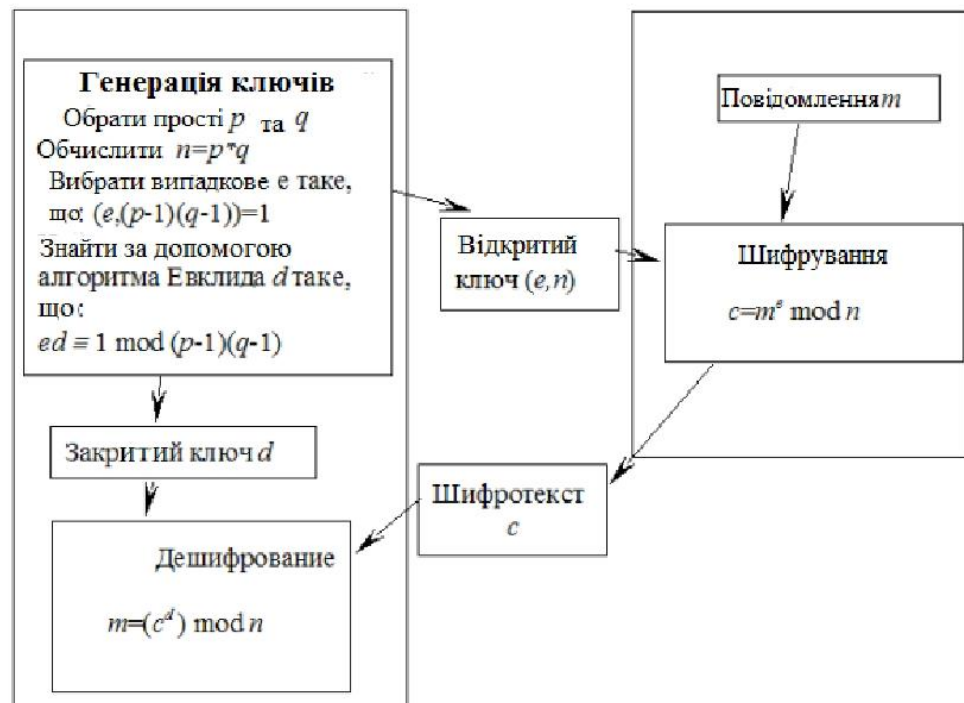


Рисунок 2.2 Схема шифрування алгоритму RSA

## 2.8 Висновки до другого розділу

У другому розділі обрано мову програмування Kotlin, як сучасну та просту у роботі з Android додатками. Крім того представлено опис середовища розробки Android Studio та загальні характеристики стандартних бібліотек: Base64, Java Cryptography Extension, Java.IO.

Також другий розділ роботи описує алгоритми методів шифрування AES і RSA.

					ІТС.4КІ.0723.03-ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

## РОЗДІЛ 3 РОЗРОБКА БІБЛІОТЕКИ

### 3.1. Розробка блок схем.

Для розробки блок-схеми роботи була використана програма Microsoft Visio 2013.

Microsoft Visio надає можливість швидко створювати бізнес-графіку різного ступеня складності: схеми бізнес-процесів, технічні, інженерні креслення, демонстрації, всілякі організаційні, рекламні та технічні схеми електричних та електронних схем, транспортних комунікаційних систем тощо. Основне ідея, вбудована в Microsoft Visio - дозволяє ефективно застосовувати готові професійні практики в особистих планах, представлених під виглядом багатой інтегрованої колекції бібліотек Visio, в якій цілий арсенал компонентів розділений на категорії, спрямовані на певну тему і розміщені в трафарети. Таким чином, завдання створення важливої графіки поєднується, щоб вибрати важливий трафарет і перетягнути потрібну фігуру на будь-яку сторінку. Після цього до розроблених тем просто додаються кольорові теми, поле та заголовки, виходять майстерно розроблені папери. Більш привабливий Visio готує можливість розвитку власних персональних бібліотек (Stencils) з графічними фігурами (Master).

Будучи улюбленим у галузі створення всіх видів ділових документів, Visio має можливість використовуватись як системи CAD, методи проектування або як звичайний векторний редактор Visio - прекрасний спосіб організувати найбільш ефективну візуалізацію експериментальних та наукових даних та статистична інформація.[20]

					ІТС.4КІ.0723.03-ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.	Татарін Ю.І.				РОЗДІЛ III РОЗРОБКА БІБЛІОТЕКИ	Літ.		Арк.	Акрушів
Керівник	Могильний Г.А.							34	50
Реценз.	Козуб Ю.Г.					ЛНУ Кафедра ІТС, Гр.4КІ			
Н. Контр.									
Зав. каф.	Семенов М.А..								

### 3.2. Розробка бібліотеки

Представлена в цієї роботи бібліотека складається з трьох основних класів: клас реалізація методу AES (схема класу див. рисунок 3.1), клас методу RSA та генератор ключів.

Перший клас, який присвячено методу AES містить в собі чотири основні функції: encryptAES, decryptAES, encryptAESToFile.

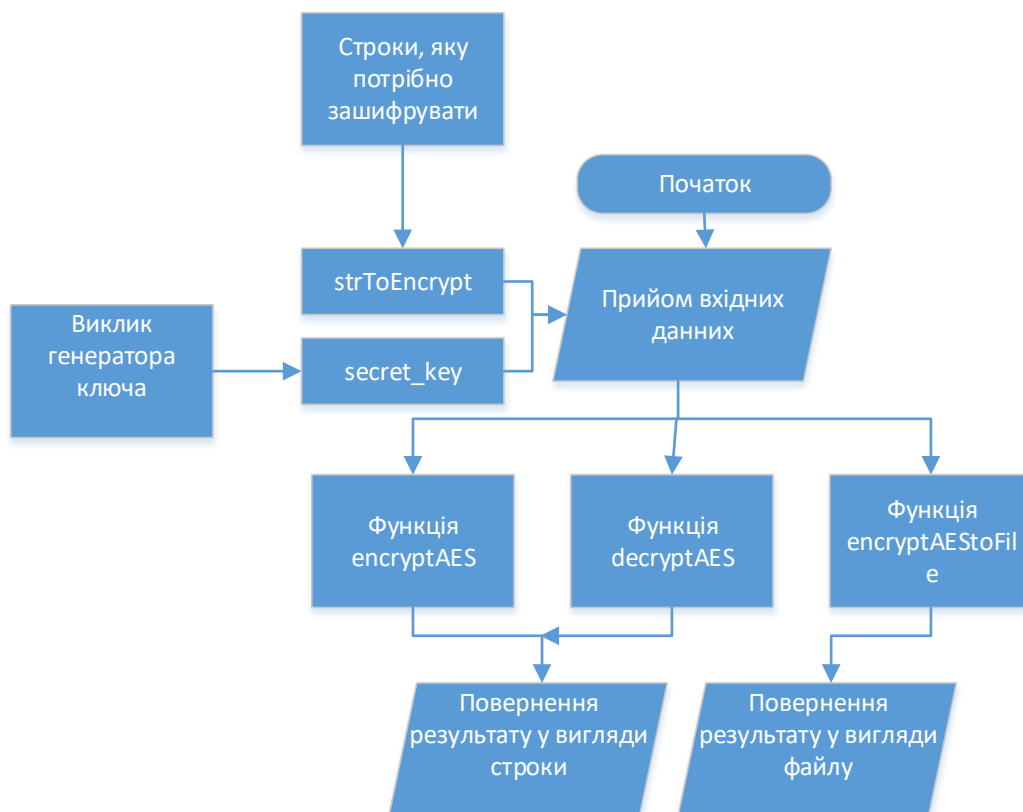


Рисунок 3.1 Блок схема класу AES

Функція encryptAES (рис.3.1) виконує шифрування методом AES використовуючи бібліотеку Java Cryptography Extension (розділ 2.3.2).

Ця функція приймає дві текстові строки, перша строка є текстом, який потрібно зашифрувати. Друга же – секретний ключ.

Секретний ключ цієї функції може бути звичайною текстовою строкою будь якого формату, з яким може працювати метод AES.

Тому під час роботи в першу чергу функція перетворює вхідний ключ до байтового масиву формату метода AES.

```
keyBytes = secret_key.toByteArray(charset("UTF8"))
val skey = SecretKeySpec(keyBytes, "AES")
```

Далі вхідний текст також перетворюється до байтового масиву.

```
val input = strToEncrypt.toByteArray(charset("UTF8"))
```

Потім виконується шифрування методами стандартної бібліотеки та в якості результату повертається строка з зашифрованою інформацією. Повна блок схема функції зображена на риснку 3.2.

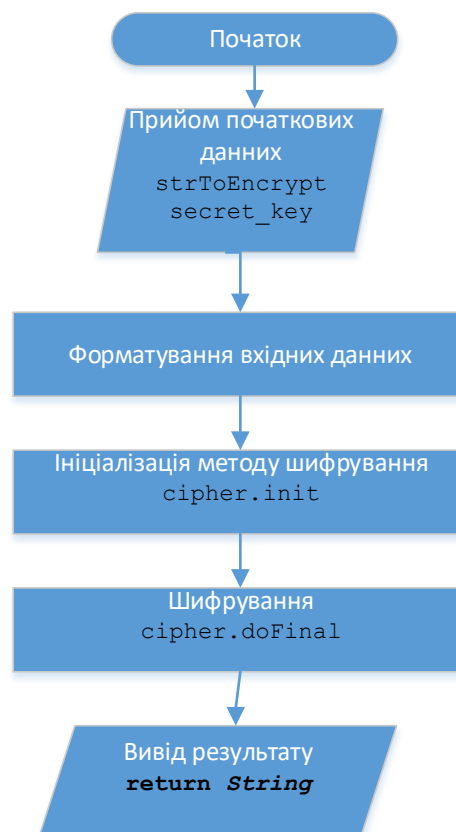


Рисунок 3.2 Блок схема методу encryptAES.

Друга функція decryptAES виконую операцію зворотну першій, окрім того, що вхідний шифр спочатку перетворюється до необхідного формату, а саме видаляються усі пробели.

```
val input = android.util.Base64.decode(strToDecrypt?.trim { it <= ' ' }?.toByteArray(charset("UTF8")),1)
```

Потім так саме повертається розшифрований результат.



Остання функція encryptAESToFile виконує запис зашифрованої строки до файлу.

Другий клас присв'ячено методу RSA. Цей клас виконує шифрування так саме, як і перший за винятком того, що у етапі ініціалізації вибирається метод RSA. Схема класу зображена на рис 3.3.

```
val cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding")
```

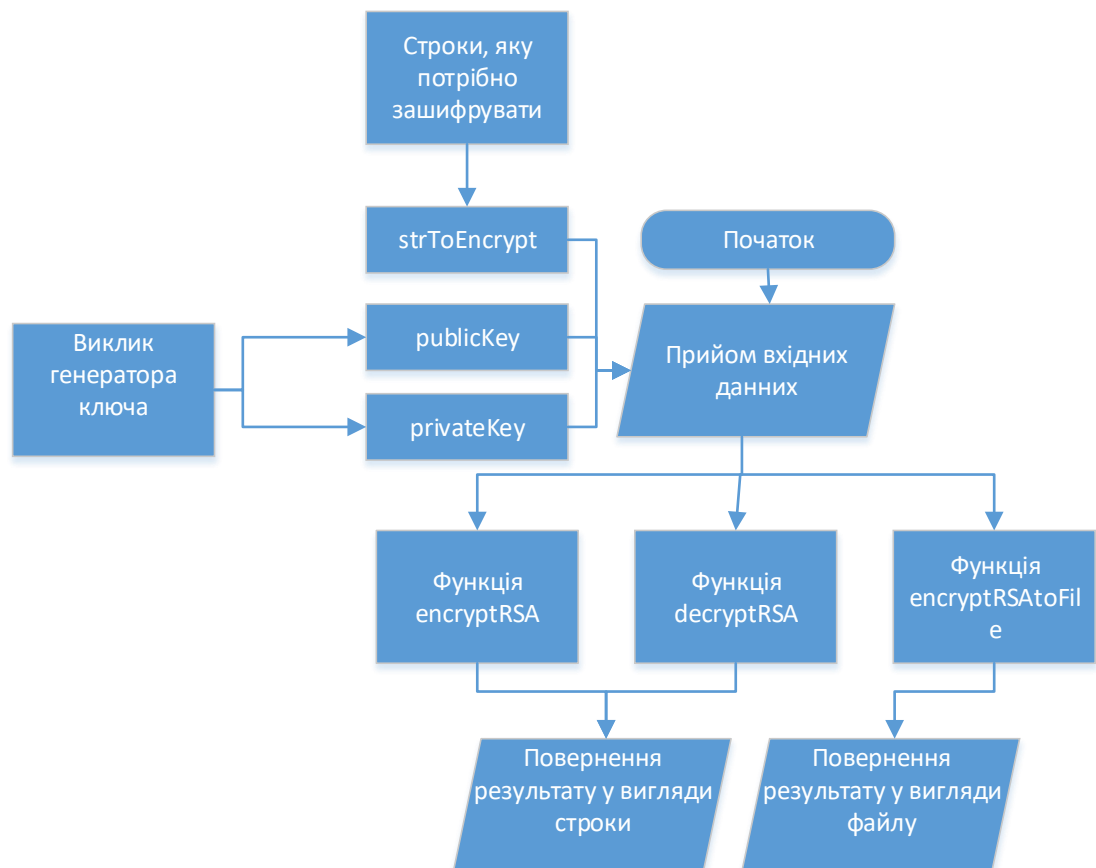


Рисунок 3.3 Блок схема класу RSA

Клас генератора ключів складається з двох частин: перша частина виконує генерацію одного ключа з можливістю обрати його довжину для метода AES, друга частина генерує парний ключ методу RSA.

Для реалізації генератора ключа AES використовується наступна функція:

```
fun AESkeygen(view: View) {
fun getRandomString(length: Int) : String {
    val allowedChars =
    "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
```

yz"

```
return (1..length)
  .map { allowedChars.random() }
  .joinToString("")
```

Яка генерує випадковий ключ з набору символів `allowedChars`, який може бути змінено користувачем за бажанням.

Парний ключ генеруються засобами стандартної бібліотеки генератора ключів.

### 3.3 Додаток для демонстрації роботи бібліотеки

Для того, щоб продемонструвати роботу бібліотеки було розроблено додаток, що виконує шифрування вхідного тексту. Блок схему його представлена на рисунку 3.4

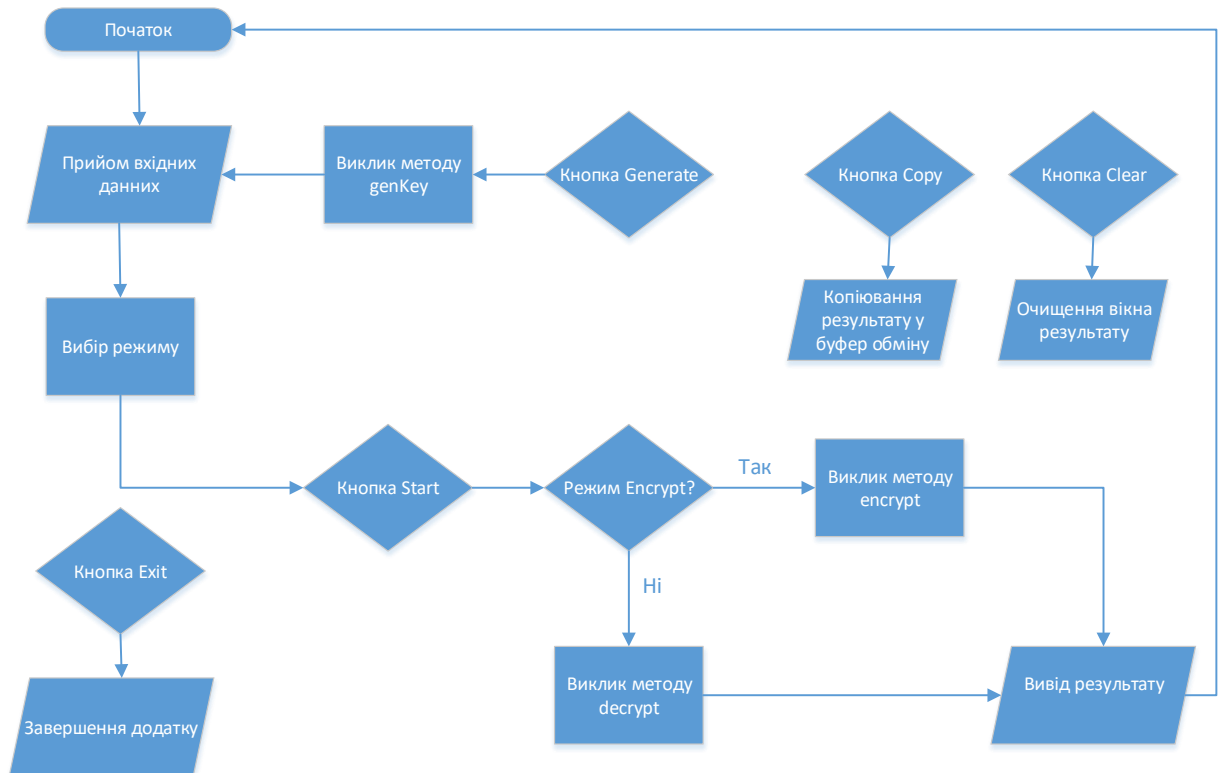


Рисунок 3.4 Блок схема додатку для демонстрації.

Цей додаток використовує представлені у роботі класи для проведення шифрування введеної у спеціальне поле. (Рисунок 3.5)

Рисунок 3.5 Строка для ввода текста

Для роботи з секретним ключем є спеціальне поле з можливістю ручного вводу та генерації.(Рисунок 3.6) Але це поле і ключ підтримується лише методом AES. Для реалізації методу RSA потрібно викликати інший генератор, який представлено у бібліотеці. Який в свій час генерую два зв’язаних між собою ключа для шифрування та дешифрування.

Рисунок 3.6 Поле ввода секретного ключа

Результат функцій виводиться у спеціальний блок, з можливістю швидкого копіювання.(Рисунок 3.7)

Рисунок 3.7 Блок вывода результата

### 3.4 Інструкція користувача

Для того, щоб почати використовувати представлену у роботі бібліотеки в першу чергу потрібно додати її до свого проекту методом копіювання до папки. (Рисунок 3.8)

					ІТС.4КІ.0723.03-ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

name	Date modified	type	Size
.gradle	03/12/2019 3:57 PM	File folder	
.idea	15/06/2020 2:54 PM	File folder	
app	15/06/2020 2:54 PM	File folder	
build	03/06/2020 1:57 A...	File folder	
encrypt	15/06/2020 2:47 PM	File folder	
gradle	03/12/2019 3:57 PM	File folder	

Рисунок 3.8 Бібліотека у папці проекту

Далі потрібно додати зв'язок у файлі gradle. (Рисунок 3.9)

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.core:core-ktx:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
    implementation 'com.google.android.material:material:1.0.0'
    implementation 'androidx.gridlayout:gridlayout:1.0.0'
    implementation project(path: ':encrypt')
```

Рисунок 3.9 Зв'язок у файлі Gradle.

Після цього бібліотека готова к роботі. Залишається лише викликати одну з необхідних функцій, кожна з яких повертає результат у вигляді строки.(Рисунок 3.10)

```
fun onClick(view: View) {

    val key = autoCompleteTextView2.text.toString()

    if (switch1.isChecked)
        textView.text = decryptRSA( publ, autoCompleteTextView.text.toString())
    else
        textView.text = encryptRSA(autoCompleteTextView.text.toString(), priv)
```

Рисуно 3.10 Виклик функції шифрування.

### 3.5 Висновки до третього розділу

У третьому розділі в першу чергу розроблено структуру основних класів бібліотеки та їх блок схеми. Також у цьому розділі представлено додаток, який демонструє роботу бібліотеки та інструкція користувача.

## ВИСНОВКИ

Таким чином у роботи було розроблено бібліотеку для шифрування даних методами AES і RSA на мові програмування Kotlin. Бібліотека відрізняється можливістю спрощеного використання представлених у роботі методів. Має свій генератор секретних ключів та можливість збереження результатів у файл.

У ході роботи було розглянуто сучасні методи шифрування та обрано методи AES і RSA, як одні з найрозповсюджених варіантів. Також розглянуто принципи роботи файлових систем Android.

Далі було обрано мову програмування Kotlin, яка є дуже простою у роботі та при цьому повністю сумісна з будь якими рішеннями на мові Java. Крім того важливим є і швидкий набір популярності цієї мови програмування.

Також під час проектування бібліотеки детально розглянуто роботу використаних алгоритмів та описано стандартні бібліотеки. Серед яких основними можна виділити Java Cryptography Extension, яка використовується для реалізацію шифрування та дешифрування. І Java.IO, один з модулів якої необхіден для реалізації запису у файл.

Останнім кроком розробки бібліотеки було створення трьох основних класів. Серед яких є самі методи AES і RSA з додатковою функцією зипусу результату у файл та генератора секретних ключів.

Крім цього робота містить інструкцію користувача до представленої бібліотеки.

Що до вдосконалення бібліотеки можливо створити вибір формату секретних ключів методу RSA.

					ІТС.4КІ.0723.03-ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Татарін Ю.І.			ВИСНОВКИ	Літ.	Арк.
Керівник		Могильний Г.А.					Акрушів
Реценз.		Козуб Ю.Г.					43
Н. Контр.						ЛНУ	
Зав. каф.		Семенов М.А.				Кафедра ІТС, Гр.4КІ	
						50	

## СПИСОК ВИКОРИСТОВАНИХ ДЖЕРЕЛ

1. Криптографія [Електронний ресурс] – Режим доступу до ресурсу:  
<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%8F>.
2. Бауэр Ф. Розшифровані секрети. Методи і принципи криптології. / Ф. 2. Бауэр., 2007. – 550 с.
3. Мао В. Сучасна криптографія. Теорія та практика. М.: Вільямс, 2005. 763 с.
4. Алфьоров А.П., Зубов А.Ю., Кузьмін А.С., Черьомушкін А. В. Основи криптографії. М.: Геліос АРВ, 2001. 479 с.
5. Практична криптографія [Електронний ресурс] – Режим доступу до ресурсу:  
<http://bit.nmu.org.ua/ua/student/metod/cryptology/%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F%209.pdf>.
6. Панасенко С. П. Алгоритмы шифрования. Специальный справочник / Сергей Петрович Панасенко..
7. Бібліотека програм [Електронний ресурс] – Режим доступу до ресурсу: <https://www.info.jinr.ru/programs/publ/enc.htm>.
8. Котлін [Електронний ресурс] – Режим доступу до ресурсу:  
<https://ru.wikipedia.org/wiki/Kotlin>.
9. Антонио Л. Kotlin for Android Developers: Learn Kotlin the easy way while developing an Android App / Лейва Антонио..
10. Филлипс Б. Android. Программирование для профессионалов / Б. Филлипс, К. Стюарт..
11. Як працює андроїд [Електронний ресурс] – Режим доступу до ресурсу:  
<https://habr.com/ru/company/solarsecurity/blog/427431/>.
12. Особливості Ext4 [Електронний ресурс] – Режим доступу до ресурсу:  
<https://habr.com/ru/post/58183/>.

- 13.Ext4 [Электронный ресурс] – Режим доступа до ресурсу:  
<https://ru.wikipedia.org/wiki/Ext4>.
- 14.F2Fs [Электронный ресурс] – Режим доступа до ресурсу:  
[https://ru.bmstu.wiki/F2FS \(Flash-Friendly File System\)](https://ru.bmstu.wiki/F2FS_(Flash-Friendly_File_System)).
- 15.Richard A. M. RSA and Public-Key Cryptography / A. Mollin Richard..
- 16.Основи файлових операцій [Электронный ресурс] – Режим доступа до ресурсу: <https://www.fandroid.info/7-osnovy-kotlin-fajlovy-e-operatsii/>.
- 17.Пакет Java.Ио [Электронный ресурс] – Режим доступа до ресурсу:  
<https://wm-help.net/lib/b/book/3683783285/38>
- 18.Алгоритм RSA [Электронный ресурс] – Режим доступа до ресурсу:  
[https://www.opennet.ru/docs/RUS/inet\\_book/6/rsa\\_642.html](https://www.opennet.ru/docs/RUS/inet_book/6/rsa_642.html).
- 19.Алгоритм AES [Электронный ресурс] – Режим доступа до ресурсу:  
<http://sgpek.ru/files/electronbook/BFIS/lab4.pdf>.
- 20.Скотт А. Microsoft Visio 2013.Шаг за шагом / Александр Скотт..
- 21.Джошуа Б. Эффективная Java / Блох Джошуа..

## ДОДАТОК А Код бібліотеки шифрування

```
package com.example.encrypt

import android.util.Base64
import java.io.File
import java.io.UnsupportedEncodingException
import java.security.InvalidKeyException
import java.security.NoSuchAlgorithmException
import javax.crypto.*
import javax.crypto.spec.SecretKeySpec

fun encryptAES(strToEncrypt: String, secret_key: String): String? {
    var keyBytes: ByteArray

    try {
        keyBytes = secret_key.toByteArray(charset("UTF8"))
        val skey = SecretKeySpec(keyBytes, "AES")
        val input = strToEncrypt.toByteArray(charset("UTF8"))

        synchronized(Cipher::class.java) {
            val cipher = Cipher.getInstance("AES/ECB/PKCS7Padding")
            cipher.init(Cipher.ENCRYPT_MODE, skey)

            val cipherText = ByteArray(cipher.getOutputSize(input.size))
            var ctLength = cipher.update(
                input, 0, input.size,
                cipherText, 0
            )
            ctLength += cipher.doFinal(cipherText, ctLength)
            return String(
                Base64.encode(cipherText, 1)
            )
        }
    } catch (uee: UnsupportedEncodingException) {
        uee.printStackTrace()
    } catch (ibse: IllegalBlockSizeException) {
        ibse.printStackTrace()
    } catch (bpe: BadPaddingException) {
        bpe.printStackTrace()
    } catch (ike: InvalidKeyException) {
        ike.printStackTrace()
    } catch (nspe: NoSuchPaddingException) {
        nspe.printStackTrace()
    } catch (nsae: NoSuchAlgorithmException) {
        nsae.printStackTrace()
    } catch (e: ShortBufferException) {
        e.printStackTrace()
    }

    return null
}

fun encryptAESToFile( secret_key: String, strToEncrypt: String): String? {

    var res = encryptAES(strToEncrypt,secret_key)

    return null
}

fun decryptAES(key: String, strToDecrypt: String?): String? {
```



```

var keyBytes: ByteArray

try {
    keyBytes = key.toByteArray(charset("UTF8"))
    val skey = SecretKeySpec(keyBytes, "AES")
    val input = android.util.Base64.decode(strToDecrypt?.trim { it <= ' ' }
)?.toByteArray(charset("UTF8")),1)

    synchronized(Cipher::class.java) {
        val cipher = Cipher.getInstance("AES/ECB/PKCS7Padding")
        cipher.init(Cipher.DECRYPT_MODE, skey)

        val plainText = ByteArray(cipher.getOutputSize(input.size))
        var ptLength = cipher.update(input, 0, input.size, plainText, 0)
        ptLength += cipher.doFinal(plainText, ptLength)
        val decryptedString = String(plainText)
        return decryptedString.trim { it <= ' ' }
    }
} catch (uee: UnsupportedOperationException) {
    uee.printStackTrace()
} catch (ibse: IllegalBlockSizeException) {
    ibse.printStackTrace()
} catch (bpe: BadPaddingException) {
    bpe.printStackTrace()
} catch (ike: InvalidKeyException) {
    ike.printStackTrace()
} catch (nspe: NoSuchPaddingException) {
    nspe.printStackTrace()
} catch (nsae: NoSuchAlgorithmException) {
    nsae.printStackTrace()
} catch (e: ShortBufferException) {
    e.printStackTrace()
}

return null
}

```

```
package com.example.encrypt
```

```
import android.util.Base64
```

```

import java.io.UnsupportedEncodingException
import java.security.InvalidKeyException
import java.security.Key
import java.security.NoSuchAlgorithmException
import javax.crypto.*
import javax.crypto.spec.SecretKeySpec

```

```
fun encryptRSA(strToEncrypt: String, privateKey: Key) :String? {
```

```

    try {
        val skey = privateKey
        val input = strToEncrypt.toByteArray(charset("UTF8"))

        val cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding")
        cipher.init(Cipher.ENCRYPT_MODE, skey)
        val cipherText = ByteArray(cipher.getOutputSize(input.size))
        var ctLength = cipher.update(
            input, 0, input.size,
            cipherText, 0
        )
        ctLength += cipher.doFinal(cipherText, ctLength)
        return String(

```

```

        Base64.encode(cipherText, 1)
    )

    } catch (uee: UnsupportedOperationException) {
        uee.printStackTrace()
    } catch (ibse: IllegalBlockSizeException) {
        ibse.printStackTrace()
    } catch (bpe: BadPaddingException) {
        bpe.printStackTrace()
    } catch (ike: InvalidKeyException) {
        ike.printStackTrace()
    } catch (nspe: NoSuchPaddingException) {
        nspe.printStackTrace()
    } catch (nsae: NoSuchAlgorithmException) {
        nsae.printStackTrace()
    } catch (e: ShortBufferException) {
        e.printStackTrace()
    }

    return null
}

fun decryptRSA(publicKey: Key, strToDecrypt: String?): String? {

    try {
        val skey = publicKey
        val input = android.util.Base64.decode(strToDecrypt?.trim { it <= ' ' }
        }?.toByteArray(charset("UTF8")),1)

        synchronized(Cipher::class.java) {
            val cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding")
            cipher.init(Cipher.DECRYPT_MODE, skey)

            val plainText = ByteArray(cipher.getOutputSize(input.size))
            var ptLength = cipher.update(input, 0,
                input.size,
                plainText, 0)
            ptLength += cipher.doFinal(plainText, ptLength)

            val decryptedString = String(plainText)

            return decryptedString.trim { it <= ' ' }
        }
    } catch (uee: UnsupportedOperationException) {
        uee.printStackTrace()
    } catch (ibse: IllegalBlockSizeException) {
        ibse.printStackTrace()
    } catch (bpe: BadPaddingException) {
        bpe.printStackTrace()
    } catch (ike: InvalidKeyException) {
        ike.printStackTrace()
    } catch (nspe: NoSuchPaddingException) {
        nspe.printStackTrace()
    } catch (nsae: NoSuchAlgorithmException) {
        nsae.printStackTrace()
    } catch (e: ShortBufferException) {
        e.printStackTrace()
    }
}

var err = "error"
return err
}

package com.example.encrypt
import java.security.Key

```

```
import java.security.KeyPair
import java.security.KeyPairGenerator
import java.util.*

fun AESkeygen(length: Int) : String {
    val allowedChars =
        "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
    return (1..length)
        .map { allowedChars.random() }
        .joinToString("")
}

fun RSAPair() : KeyPair {

    val pair = KeyPairGenerator.getInstance("RSA")
    pair.initialize(1024)
    val kp = pair.genKeyPair()

    return kp
}

fun getPrivate (kp: KeyPair) : Key {
    val privatek = kp.getPrivate()
    return privatek
}

fun getPublic (kp: KeyPair) : Key {
    val publick = kp.getPublic()
    return publick
}
```

## ДОДАТОК Б Код програми для демонстрації

```
package com.example.kurs_turtsevich

import android.content.ClipData
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.content.ClipboardManager
import android.content.Context
import com.example.encrypt.*

import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

    var pair = RSApair()
    var priv = getPrivate(pair)
    var publ = getPublic(pair)

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        autoCompleteTextView2.setText("662ede816988e58Nb6d057d9d85605e0")

        switch1.setOnCheckedChangeListener { _, isChecked ->
            if (isChecked) {
                autoCompleteTextView.setText("Enter value to decrypt")
            } else {
                autoCompleteTextView.setText("Enter value to encrypt")
            }
        }
    }

    fun onClick(view: View) {

        val key = autoCompleteTextView2.text.toString()

        if (switch1.isChecked)
            textView.text = decryptRSA( publ,
autoCompleteTextView.text.toString())
        else
            textView.text = encryptRSA(autoCompleteTextView.text.toString(),
priv)

    }

    fun onClick1(view: View) {
        finishAffinity()
    }

    fun copyResult(view: View) {
        val clipboard = getSystemService(Context.CLIPBOARD_SERVICE) as
ClipboardManager
        val clip = ClipData.newPlainText(textView.text, textView.text)
        clipboard.setPrimaryClip(clip)
    }

    fun clearRes(view: View) {
```

```
        autoCompleteTextView.text.clear()
    }
    fun genKey(view: View) {
        autoCompleteTextView2.setText(AESkeygen(32))
    }
}
```