Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Підвищення швидкодії потокового шифрування аудіо інформації з використанням багатопотокової обробки в реальному часі»

Виконав: ст. 2-го курсу, групи УБ-20м
спеціальності 125– Кібербезпека
Освітня програма – Управління інформаційною безпекою_____
(шифр і назва напряму підготовки, спеціальності)
Медяна І.Л
(прізвище та ініціали)

Керівник: к.т.н., доц., зав. каф. МБІС
Карпінець В.В.
(прізвище та ініціали)
«____» _____ 2021 р.

Опонент: к.т.н., доц., доцент каф. ОТ
Войцеховська О.В.
(прізвище та ініціали)
«____» _____ 2021 р.

Допущено до захисту
Голова секції УБ кафедри МБІС
д.т.н., проф. Яремчук Ю.Є.
“_____”_____2021 р.

Вінниця ВНТУ - 2021 рік

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Управління інформаційною безпекою

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС

_____д.т.н., проф. Яремчук Ю.Є.
“_____”_____2021 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту

Медяній Ірині Леонідівні

1. Тема магістерської кваліфікаційної роботи «Підвищення швидкодії потокового шифрування аудіо інформації з використанням багатопотокової обробки в реальному часі»

Керівник магістерської кваліфікаційної роботи к.т.н., доц., зав. каф. МБІС Карпінець В.В. затверджені наказом вищого навчального закладу від “24 ” вересня 2021 року № 277

2. Строк подання студентом МКР 13 грудня 2021 року.

3. Вихідні дані до МКР нормативно-правова база, сучасні наукові розробки, які опубліковані в монографічній літературі, інтернет ресурси.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): В першому розділі проаналізувати методи та засоби захисту звукової інформації.

В другому розділі підвищита швидкодію алгоритму роботи програми шифрування з мікрофону

АНОТАЦІЯ

В роботі виконаний аналіз проблем, пов’язаних із захистом звукової інформації. Особлива увага була приділена методам захисту звукових файлів. Був розроблений і перевірений алгоритм для реалізації поставленої задачі. Згідно алгоритму було розроблено програму, що дозволяє шифрувати як аудіофайли, так і здійснювати безпосереднє потокове шифрування звуку з мікрофона, під‘єднаного до комп‘ютера. Додаткові можливості програми – це вибір кількості потоків та частоту дискретизації. Для написання програми використовувалася мова програмування Phyton.

Ключові слова: шифрування, підвищення швидкодії, розробка алгоритму.

ANNOTATION

The work analyzes the problems related to the protection of sound information. Special attention was paid to the methods of protecting audio files. The algorithm for realization of the set task was improved. According to the algorithm, the program was improved, which allows you to encrypt both audio files and perform direct streaming audio encryption from the microphone. Additional features of the program are the choice of the number of threads and the sampling rate. Phyton programming language is used to write programs.

Keywords: encryption, speed increase, algorithm development.

ЗМІСТ

ВСТУП

У наш час технологічного розвитку “безпека” здобуває розширений зміст і містить у собі такі складові, як фізична, юридична й інформаційна безпека.

Особливе місце займає інформаційна безпека у зв'язку зі зростаючою роллю інформації в житті суспільства та вимагає до себе все більшої уваги. Успіх виробничої й підприємницької діяльності в чималому ступені залежить від уміння розпоряджатися таким найціннішим товаром, як інформація. Зараз головним ресурсом замість капіталу стає інформація. Інформаційні ресурси є об'єктами власності громадян, організацій, громадських об'єднань, держави.

Інформація, як сукупність знань і фактичних даних і залежність між ними, стала стратегічним ресурсом: вона-основа для прийняття любого рішення. Тому захист інформації є складна, науковоємна проблема по своїй суті, в умовах запровадження сучасних інформаційних технологій, виникнення мережі Internet, інших мереж і набуває особливу важливість і необхідність.

Багатоваріантність побудови інформаційних систем дає багато рішень в сфері розробки систем захисту інформації. Широкомасштабне використання обчислювальної техніки і телекомунікаційних систем в рамках територіально розподіленої мережі, перехід на цій основі до безпаперової технології, збільшення об’ємів інформації, збільшення кількості користувачів приводить до того що необхідно збільшувати, покращувати рівень якості захисту інформації

У зв'язку з вищевикладеним, тематика даної роботи, присвяченої розв’язанню задачі захисту звукових файлів є актуальною.

Мета і задачі дослідження. Метою даної роботи є підвищення швидкодії та вдосконалення алгоритму для шифрування аудіофайлів і потокового шифрування звуку з мікрофона, під’єднаного до комп‘ютера.

Для досягнення поставленої мети у роботі необхідно розв’язати такі задачі:

– проаналізувати актуальність технічної проблеми захисту аудіо інформації;

– проаналізувати існуючі методи та засоби захисту звукової інформації;

– обгрунтувати вибір методу розв’язання задачі;

– вдосконалення алгоритму;

– обрати середовище програмування для підвищення швидкодії програми потокового аудіо;

– розробити програмні засоби для реалізації запропонованого алгоритму;

– тестування роботи програмної реалізації вдосконаленого методу.

Об’єкт дослідження – програмна реалізація шифрування звукової інформації.

Предмет дослідження – швидкість шифрування та ступінь захисту звукових файлів у WAV-форматі та аудіосигналів безпосередньо з мікрофону.

Практична цінність одержаних результатів полягає у наступному:

1. Вдосконалено алгоритм роботи програми потокового шифрування звукової інформації з мікрофону.

2. Розроблений програмний засіб шифрує звукову інформацію з мікрофону, що дозволяє забезпечити захист від несанкціонованого заволодіння секретною інформацією.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність технічної проблеми захисту звукової інформації

Вплив Інтернет та мобільного зв'язку на світ комп'ютерів і комунікацій не має історичних аналогів. Суть проблеми, що виникла на сучасному етапі розвитку науки, техніки і технологій в галузі комунікацій - це забезпечення безпеки передавання інформації [1].

Багатоваріантність побудови інформаційних систем дає багато рішень в сфері побудови систем захисту інформації. Широкомасштабне використання ОТ і телекомунікаційних систем в рамках територіально розподіленої мережі, перехід на цій основі до безпаперової технології, збільшення об'ємів інформації, збільшення кількості користувачів приводить до того що необхідно збільшувати, покращувати рівень якості захисту інформації [2].

І наскільки велика важливість будь-якої інформації, яка відноситься до бізнесу, це зрозуміло для багатьох. Користуючись зібраною і обробленою інформацією, можливо успішно конкурувати на своєму ринку та завойовувати нові. Інформація допомагає в пошуках партнерів і дозволяє виділяти чітку позицію по відношенню до них. Говорячи простіше: хто володіє достовірною і повною інформацією - той володіє ситуацією - той має можливості керувати нею в своїх інтересах, а хто має можливості керувати - той і перемагає. Це проста формула успіху любої діяльності [3].

При переході до ринкової економіки, інформація стає товаром і повинна підкорятися специфічним законам товарно-ринкових відносин. В цих умовах проблема захисту інформації дуже актуальна і для любої форми власності.

Питання безпеки – важлива частина концепції впровадження нових інформаційних технологій у всі сфери життя суспільства. До управління безпекою інформації можна віднести багато аспектів, в тому числі до яких входить забезпечення цілісності, конфіденційності і автентичності інформації [4].

До інформації, яка вимагає захисту, можна віднести конфіденційну, управлінську, науково-технічну, торгову і іншу, яка має цінність для підприємства в

використані переваг над конкурентами. Витік такої інформації може принести великі збитки її власникам.

Інформація, як сукупність знань і фактичних даних і залежність між ними, стала стратегічним ресурсом: вона-основа для прийняття любого рішення. Тому захист інформації є складна, науковоємна проблема по своїй суті, в умовах запровадження сучасних інформаційних технологій, виникнення мережі Internet, інших мереж і набуває особливу важливість і необхідність. Система захисту звукових файлів, що розробляється в даній роботі, призначена для шифрування інформації, яка передається через Інтернет у вигляді файлів або для захисту розмов по мобільному телефону.

1.2 Аналіз методів захисту звукової інформації

Інформація є інтелектуальною власністю, яку можна використовувати для виробництва товарів і послуг або, продавши, перетворити її в готівку. Крім цього, вона може бути легко втрачена, у тому числі й викрадена або скопійована конкурентами.

У більшості випадків підприємцям доводиться самим визначати, яку інформацію в їхньому бізнесі доцільно тримати в таємниці від конкурентів і захищати від зазіхань на неї.

Ринкові відносини неминуче приводять до посилення конкуренції між різними комерційними структурами. Підприємцям уже зараз наноситься збиток за рахунок несанкціонованого добування конкурентами й використання їхньої інформації. Цілі несанкціонованого збору інформації найпростіше пояснити в цей час одним поняттям: комерційний інтерес. Але інформація різнохарактерна й різна за ціною, ступінь її таємності (конфіденційності) залежить від імені або групи осіб, кому вона належить, а також сфери їхньої діяльності [5].

Бізнесменам необхідні дані про конкурентів: їх слабкі й сильні сторони, ринки збуту, умови фінансової діяльності, технологічні секрети. Незайвим буде й знання кроків, які конкуренти почнуть проти своїх колег по бізнесу.

Політик, адміністратор або просто відома у певних колах людина - вже інформант: цікавим є уклад його особистого життя, зв'язки, справжнє відношення до тих або інших суспільних явищ чи особам, джерела доходів і т.д. Конкуренти або недоброзичливці можуть використовувати яким-небудь чином отримані дані у своїх корисливих цілях. Особисте життя людини завжди було цікаве для будь-кого [6].

Масовим явищем стало запозичення інтелектуальної власності (інформації). Основна мета захисту будь-якої конфіденційної інформації полягає в тому, щоб запобігти незаконному оволодінню нею конкурентами або зловмисниками.

Найбільш ефективною мірою захисту інформації є використання криптографічних методів. У цей час використовуються два основних методи шифрування - аналоговий та цифровий [7].

Класифікація методів та пристроїв захисту звукової інформації представлена на рис.1.1.

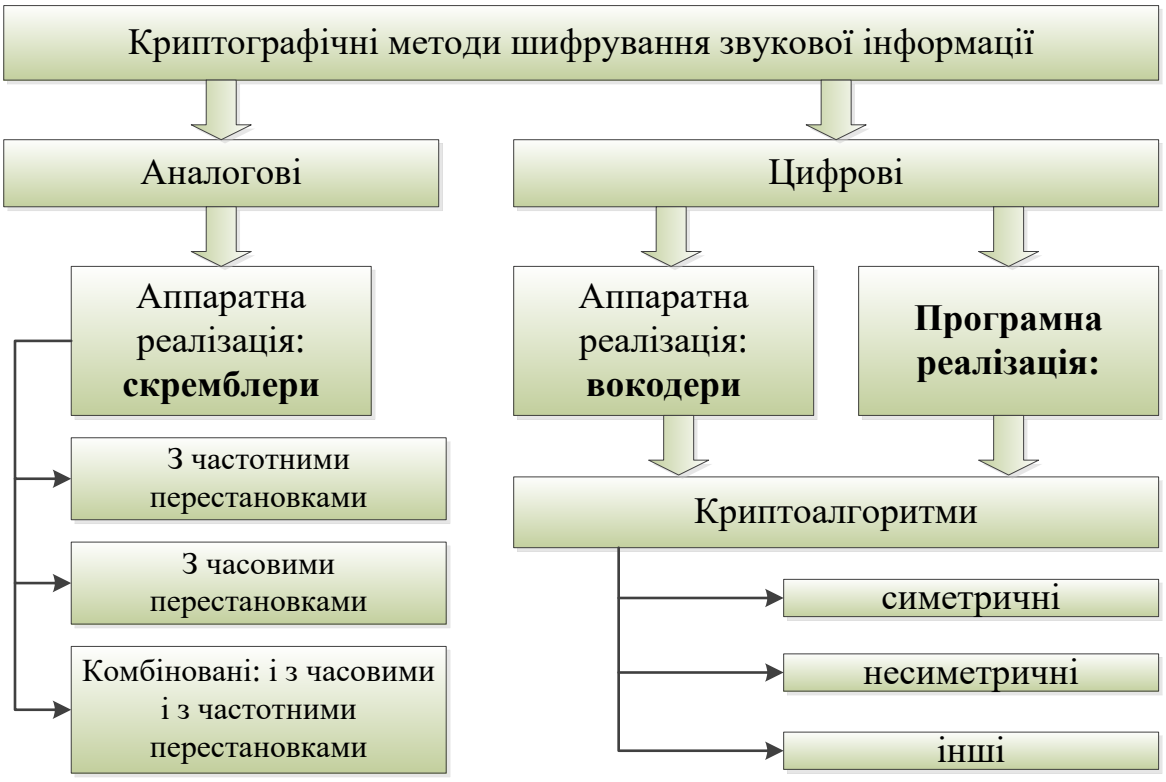

Рисунок 1.1 - Класифікація методів та пристроїв захисту звукової інформації

Крім апаратних реалізацій цих методів ще додано гілку програмних реалізацій, які можуть використовуватись не для звичайних аналогових телефонних ліній, а для цифрових ліній, для стільникового зв’язку, а також для захисту звукових файлів і

потокового звуку з мікрофона. Дана робота присвячена розробці саме програмної реалізації захисту звукової інформації.

Аналогові пристрої криптографічного захисту

Пристрої, представлені на ринку, які реалізують метод аналогового шифрування, одержали назви скремблерів. Основне застосування скремблери знаходять при захисті інформації, переданої по телефонних лініях (у тому числі стільникових). При аналоговому скремблюванні реалізуються, як правило, два основних способи шифрування: частотні та часові перестановки. При одному та другому способі характеристики переданого сигналу (мови) міняються таким чином, що сигнал, виділений за допомогою звичайного телефонного апарата, на передавальному кінці стає нерозбірливим, але займає ту ж частотну смугу, що дозволяє його передавати по лініях зв'язку у звичайному режимі. На прийомному кінці за допомогою відомого ключа проводиться зворотне перетворення. Найбільш високий рівень закриття виходить при використанні одночасно обох способів, тобто частотних і часових перестановок [8].

Стійкість аналогових скремблерів, представлених на ринку, до дешифрування становить у різних скремблерів від декількох годин до декількох тижнів залежно від складності перетворень, числа можливих ключових комбінацій і можливостей зловмисника.

У найпростіших скремблерах, що захищають лише від прямого прослуховування дилетантами, використовуються тільки частотні перестановки й інверсії (число каналів не перевищує 4, інтервали комутації постійні).

У скремблерах середнього класу, що забезпечують гарантовану стійкість на час у кілька годин, застосовуються частотно-часові перестановки із числом частотних каналів від 5 до 10.

Прикладом якісного скремблера, що реалізує високий ступінь захисту, можуть служити пристрої серії 5СК - М1.2., які в даний момент єдині мають сертифікати. Технічні характеристики вищевказаних скремблерів наведені в таблиці 1.1.

Перевагами скремблерів є [9]:

- Простота й менша вартість у порівнянні з подібними пристроями, що реалізують функцію захисту, а також малі габарити;
- Високий ступінь захисту інформації, переданої як на всьому шляху лінії зв'язку (абонент - абонент), так і на ділянці останньої милі (абонент - АТС);
- Захист як мовних, так і факсових повідомлень;
- Можливість роботи з будь-якими телефонними апаратами та різними АТС;
- Простота в керуванні при переході в закритий режим зв'язку;
- Захист від будь-яких сучасних пристроїв знімання інформації.

Недоліками скремблерів є [10]:

- Необхідність установки устаткування у всіх абонентів, що беруть участь у сеансах закритого зв'язку при організації зв'язку “абонент - абонент”, що можна виключити за умови організації закритого зв'язку “абонент - АТС”:
- Втрата невеликої кількості часу, необхідного для синхронізації скремблерів і обміну ключами на початку сеансу закритого зв'язку (від 8 сек. до 4 хв. залежно від якості лінії - каналу - зв'язку);
- Невеликі затримки мовних повідомлень при передачі повідомлень від одного абонента до іншого у зв'язку із шифруванням і дешифруванням інформації (до 1 сек.).

Таблиця 1.1 - Технічні характеристики скремблерів

№ п/п	Характеристика	SCR-M1.2	SCR-M1.2 mini	"ГРОТ"	"ГРОТ-М"	"З"
1.	Метод шифрування	мозаїчний: частотні й часові перестановки	мозаїчний: частотні й часові перестановки	мозаїчний: частотні й часові перестановки	мозаїчний: частотні й часові перестановки	мозаїчний: частотні й часові перестановки
2.	Формування ключових установок	метод відкритого розподілу ключів, можливе введення фіксованих даних	метод відкритого розподілу ключів, можливе уведення фіксованих даних	метод відкритого розподілу ключів, можливе уведення фіксованих даних	метод відкритого розподілу ключів, можливе уведення фіксованих даних	метод відкритого розподілу ключів, можливе уведення фіксованих даних
3.	Режим зв'язку	дуплекс	дуплекс	дуплекс	дуплекс	дуплекс
4.	Смуга частот у каналі зв'язку	0,3 - 3,4 кГц	0,3 - 3,4 кГц	0,3 - 3,4 кГц	0,3 - 3,4 кГц	0,3 - 3,4 кГц
5.	Оконечное встаткування	- телефонні апарати будь-якого типу з функцією тонального або-ра; - факсимільні апарати 2 і 3-й груп МККТТ; не вище 4800 біт/сек	- телефонні апарати будь-якого типу з функцією тонального набора; - факсимільні апарати 2 і 3-й груп МККТТ; не вище 4800 біт/сек	- телефонні апарати будь-якого типу з функцією тонального набора; - факсимільні апарати 2 і 3-й груп МККТТ;	- телефонні апарати будь-якого типу з функцією тонального набора; - факсимільні апарати 2 і 3-й груп МККТТ;	- телефонні апарати будь-якого типу з функцією тонального набора; - факсимільні апарати 2 і 3-й груп МККТТ;

Цифрові пристрої, криптографічного захисту:

Пристрої, представлені на ринку, які реалізують метод цифрового шифрування, одержали назву вокодери [11].

Цифровий спосіб кодування інформації є істотно більш стійким до дешифрування. У даних пристроях сигнал попередньо перетвориться в цифровий вигляд. У канал зв'язку видається набір стандартних знаків (як правило, нулі й одиниці), як це відбувається при передачі даних [8]. Для кодування подібних сигналів застосовуються значно більш складні й витончені системи ключів. Сильне обмеження на використання вокодерів накладає їхня вартість (від 500 до 1500 доларів за один пристрій, а при організації закритого зв'язку “абонент - абонент” їх треба два) і ті обставини, що для передачі сигналу потрібна смуга частот більша, ніж може забезпечити стандартна двопровідна телефонна лінія. Головна проблема, яку доводиться вирішувати при роботі з вокодерами, складається в досягненні високої якості синтезованого мовного сигналу при реальних швидкостях його передачі по каналу зв'язку, що становлять 2400 - 9600 біт/с. За результатами дослідження з'ясувалося, що на міських лініях роботу на швидкості 2400 біт/с забезпечують 75% каналів зв'язку, на швидкості 4800 біт/с - 50% і на швидкості 9600 біт/с - 35%, тобто виникає необхідність подачі мовного сигналу на швидкостях починаючи з 2400 біт/с. Все це ускладнює просування вокодерів на ринок [12].

Перевагами вокодерів є:

- високий ступінь захисту.

Недоліками вокодерів є:

- висока вартість;
- необхідність установки встаткування у всіх абонентів, що беруть участь у сеансах закритого зв'язку при організації зв'язку “абонент - абонент”;
- проблеми при роботі на звичайних двопровідних телефонних лініях;
- затримки мовних сигналів при встановленні закритого зв'язку та після передачі його в процесі переговорів.

Для розв’язання поставленої задачі потрібно обрати між симетричним та асиметричним шифруванням.

Асиметричне шифрування – криптографічна система з відкритим ключем, система шифрування та електронного цифрового підпису, при якій відкритий ключ передається з відкритого каналу, і використовується для перевірки електронного цифрового підпису і для шифрування повідомлення [13]. Для генерації електронного цифрового підпису і для розшифрування повідомлення використовується секретний ключ. Криптографічні системи з відкритим ключем в даний час широко застосовуються в різних мережевих протоколах, зокрема, в протоколах TLS і його попереднику SSL.

Ідея криптографії з відкритим ключем дуже тісно пов'язана з ідеєю односторонніх функцій , тобто таких функцій f (x), що за відомим x досить просто знайти значення f (x), тоді як визначення x з f (x) складно в сенсі теорії.

Але сама одностороння функція марна в застосуванні: нею можна зашифрувати повідомлення, але розшифрувати не можна. Тому криптографія з відкритим ключем використовує односторонні функції з лазівкою. Лазівка – це якийсь секрет, який допомагає розшифрувати. Тобто існує такий y, що знаючи f (x), можна обчислити x.

Схема шифрування з відкритим ключем має наступний вигляд [14]:

Нехай K – простір ключів, а e і d – ключі шифрування і розшифрування відповідно. E_e – функція шифрування для довільного ключа e ∈ K, так що :

$$E_e(m) = c\ , \tag{1.1}$$

де c ∈ C , де C – простір шифротекстів, а m ∈ M, де M – простір повідомлень.

$$D_d(c) = m, \tag{1.2}$$

де D_d – функція розшифрування, за допомогою якої можна знайти вихідне повідомлення m, знаючи шифротекст c.

Нижче показана схема передачі інформації особою A особі B (рис.1.2)

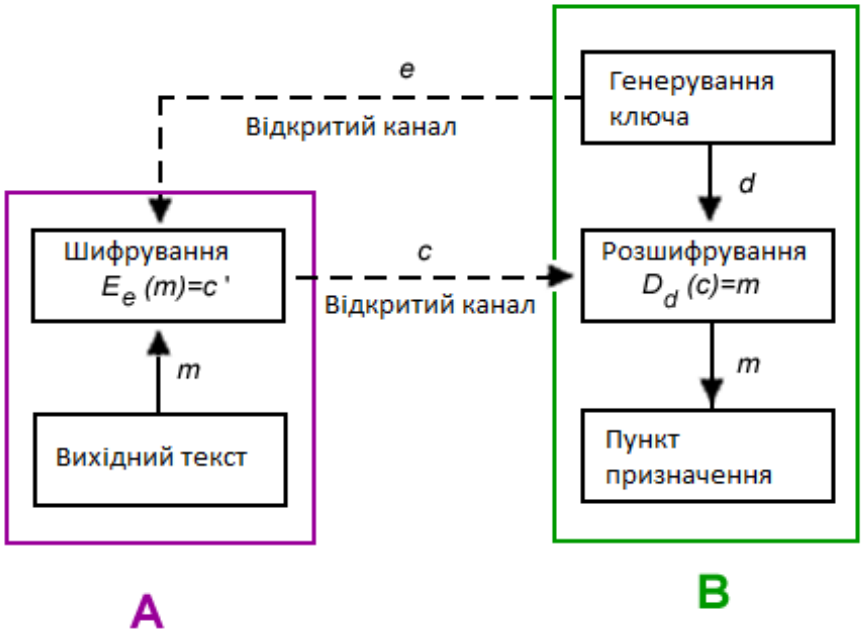

Рисунок 1.2 – Схема передачі інформації

Обгрунтування схеми передачі інформації:

1. Особа B обирає пару (e, d) і відсилає ключ шифрування e (відкритий ключ) особі A по відкритому каналу, а ключ розшифрування d (закритий ключ) захищений і секретний (не повинен передаватись по відкритому каналу).

2. Щоб відіслати повідомлення m особі B, особа A застосовує функцію шифрування, відповідну відкритим ключем e.

3. Особа B розшифровує шифротекст c, застосовуючи зворотнє перетворення D_d.

Алгоритми криптосистеми з відкритим ключем можна використовувати [15]:

– Як самостійні засоби для захисту переданої та збереженої інформації;

– Як засоби розподілу ключів. Звичайно за допомогою алгоритмів криптосистем з відкритим ключем розподіляють ключі, малі за об'ємом.. А саму передачу великих інформаційних потоків здійснюють за допомогою інших алгоритмів;

– Як засоби автентифікації користувачів.

Переваги асиметричних шифрів:

Перевага асиметричних шифрів перед симетричними шифрами полягає у відсутності необхідності попередньої передачі особистого ключа по надійному каналу.

У симетричної криптографії ключ тримається в секреті для обох сторін, а в асиметричної криптосистеми тільки один секретний [16].

При симетричному шифруванні необхідно оновлювати ключ після кожного факту передачі, тоді як в асиметричних криптосистемах пару (E, D) можна не змінювати значний час.

У великих мережах число ключів в асиметричної криптосистеми значно менше, ніж у симетричної.

Недоліки асиметричних шифрів:

Перевага алгоритму симетричного шифрування над несиметричним полягає в тому, що в перший відносно легко внести зміни [17].

Хоча повідомлення надійно шифруються, але «засвічуються» одержувач і відправник самим фактом пересилання шифрованого повідомлення.

Процес шифрування-розшифрування з використанням пари ключів проходить на два-три порядки повільніше, ніж шифрування-розшифрування того ж тексту симетричним алгоритмом.

У чистому вигляді асиметричні криптосистеми вимагають значно більших обчислювальних ресурсів [18].

1.3 Обгрунтування вибору методу розв’язання задачі

Після аналітичного огляду методів та засобів захисту звукової інформації для розв’язання поставленої задачі було обрано симетричне шифрування, а саме – потоковий шифр.

Симетричне шифрування – схема, в якій ключ дешифрування та ключ шифрування збігаються, або один легко обчислюється за допомогою іншого, на відміну від асиметричного, коли важко обчислити ключ дешифрування [19].

Симетричні алгоритми шифрування вимагають меньше обчислень, ніж асиметричні. Це означає, що якісні асиметричні алгоритми набагато разів повільніші за якісні симетричні [20].

Перевагами симетричного алгоритму є:

– швидкість;

– простота реалізації;

– вивченість;

– необхідна менша довжина ключа для порівнянної стійкості.

Недоліками є:

– складність обміну ключами

– складність управління ключами у великій мережі

Для компенсації недоліків симетричного шифрування широко застосовується гібридна криптографічна схема, де за допомогою асиметричного шифрування передається ключ, що використовується обома сторонами для обміну даними за допомогою симетричного шифрування.

Симетричні алгоритми шифрування розділяються на потокові та блочні.

Потокові шифри – група симетричних шифрів, які шифрують кожен символ відкритого тексту незалежно від інших символів.

У основі таких шифрів є шифр XOR який може бути записаний формулою 1.3:

$$C_i = P_i \text{ XOR } K_j, \quad (1.3)$$

З тією відмінністю, що гама для шифру XOR формується за певним алгоритмом, який є криптостійким генератором (з використанням ключа) псевдовипадкової послідовності символів [21].

Блочний шифр – різновид симетричного алгоритму, особливістю якого є обробка блоку декількох байт за одну ітерацію.

Робота блочного шифру — застосування функції, що шифрує, до блоку даних (проста заміна) викликає серйозну проблему: статистичні властивості відкритих даних частково зберігаються, тому що кожному однаковому блоку даних однозначно відповідає зашифрований блок даних. При великій кількості даних (відео, звук) це може дати деякі відомості для криптоаналізу про зміст даних [22].

Видалення статистичних залежностей у відкритому тексті можливо за допомогою попереднього архівування, але воно не вирішує завдання повністю, тому що у файлі залишається службова інформація архіватора, і не завжди технічно припустимо [23].

Через виникнення цієї проблеми для розробки програми краще використовувати потокове шифрування.

Для підвищення швидкодії потокового шифрування аудіо інформації використовуватиметься багатопотоковість.

Багатопотоковість — властивість операційної системи або застосунку, яка полягає в тому, що процес, породжений в операційній системі, може складатися з кількох потоків, що виконуються паралельно, або навіть одночасно на багатопроцесорних системах. При виконанні деяких завдань таке розділення може досягти ефективнішого використання ресурсів комп'ютера. Такі процеси виконання ще називають потоками [24].

Суттю багатопотоковості є квазі-багатозадачність на рівні одного виконуваного процесу, тобто всі потоки виконуються в адресному просторі процесу. Окрім цього, всі потоки процесу мають не тільки спільний адресний простір, але і спільні дескриптори файлів. Процес, що виконується, має як мінімум один (головний) потік.

Переваги багатопотоковості наступні:

– спрощення програми в деяких випадках, за рахунок використання загального адресного простору;

– менші відносно процесу часові витрати на створення нитки і взаємодію між нитками;

– підвищення продуктивності процесу за рахунок розпаралелювання процесорних обчислень і операцій вводу/виводу.

Типи реалізації потоків:

Потік в просторі користувача.

Кожен процес має таблицю потоків, аналогічну таблиці процесів ядра.

Переваги цього типу наступні:

- можливість реалізації на ядрі, що не підтримує багатопотоковість;
- швидше переключення, створення і завершення потоків;
- процес може мати власний алгоритм планування.

Недоліки:

- відсутність переривання по таймеру усередині одного процесу;
- при використанні блокуючого системного запиту решта всіх потоків блокується;
- відсутній виграш у швидкодії на багатопроцесорних системах;
- складність реалізації.

Потік в просторі ядра.

Разом з таблицею процесів в просторі ядра є таблиця потоків.

Змішана реалізація.

Нитки працюють в режимі користувача, але при системних викликах перемикаються в режим ядра. Перемикання в режим ядра і назад є ресурсоємною операцією і негативно позначається на продуктивності системи. Тому було введено поняття волокна — полегшеної нитки, що виконується виключно в режимі користувача. В кожної нитці може бути декілька волокон. Подібний тип багатонитковості реалізований в ОС Windows [25].

1.4 Висновки та постановка задачі

В даному розділі було визначено актуальність технічного захисту звукової інформації. Суть проблеми, що виникла на сучасному етапі розвитку науки, техніки і технологій в галузі комунікацій - це забезпечення безпеки передавання інформації. В наш час дуже важливо забезпечити безпеку передавання звукової інформації, адже потрапляння конфіденційної інформації до зловмисника може завдати чинемалих збитків.

Розглянуто декілька методів та засобів захисту звукової інформації. Обгрунтовано вибір методу розв’язання задачі. Для розв’язання поставленої задачі було обрано симетричне шифрування, а саме – потоковий шифр. Для підвищення швидкодії потокового шифрування аудіо інформації використовуватиметься багатопотоковість.

Для досягнення поставленої мети у роботі необхідно розв’язати такі задачі:

1. Розробити алгоритм роботи програми для підвищення швидкодії потокового шифрування аудіо інформації

2. Розробити програмні засоби для реалізації запропонованого алгоритму

2 ВДОСКОНАЛЕННЯ ТА РОЗРОБКА АЛГОРИТМУ ПІДВИЩЕННЯ ШВИДКОДІЇ ПОТОКОВОГО ШИФРУВАННЯ АУДІО ІНФОРМАЦІЇ

2.1 Розробка алгоритму підвищення швидкодії потокового шифрування аудіо інформації

Використання декількох потоків у застосуванні означає внесення в нього паралелізму. Паралелізм – це одночасне (з погляду прикладного програміста) виконання дій різними фрагментами коду застосування. Така одночасність може бути реалізована на одному процесорі шляхом перемикання задач, а може ґрунтуватися на паралельному виконанні коду на декількох процесорах. Потоки абстрагують цю відмінність, даючи можливість розробляти застосування, які в одно процесорних системах використовують псевдо паралелізм, а при додаванні процесорів – справжній паралелізм, такі застосування масштабуються зі збільшенням кількості процесорів.

Перш ніж розглянути основні підходи до реалізації моделі потоків, розглянемо означення важливих понять потоку користувача і потоку ядра.

Потік користувача – це послідовність виконання команд в адресному просторі процесу. Ядро ОС не має інформації про такі потоки, вся робота з ними виконується в режимі користувача. Засоби підтримки потоків користувача надають спеціальні системні бібліотеки; вони доступні для прикладних програмістів у вигляді бібліотечних функцій. Бібліотеки підтримки потоків у наш час звичайно реалізують набір функцій, визначений стандартом POSIX [26].

Потік ядра – це послідовність виконання команд в адресному просторі ядра. Потоками ядра управляє ОС, перемикання ними можливе тільки у привілейова-ному режимі. Є потоки ядра, які відповідають потокам користувача, і потоки, що не мають такої відповідності. Співвідношення між двома видами потоків визначає реалізацію моделі потоків.

Існують декілька підходів (моделей) у багатопотоковому програмуванні:

- синхронізація, блокування та ключове слово volatile;

− транзакційна пам'ять — прошарок між JVM і API програми, рекурсивний паралелізм;

− модель акторів — коли кожен об'єкт є потоком, який обмінюються повідомленнями з іншими потоками.

Способи організації багатопотоковості у програмах:

− потоки не взаємодіють один з одним, працюють самі собою;

− потоки взаємодіють один з одним;

− потоки працюють самі собою, а потім збирають дані в єдиний результат.

Для підвищення швидкодії потокового шифрування аудіо інформації в реальному часі необхідно створити декілька потоків що забезпечуватимуть паралельну роботу алгоритму шифрування [27].

Розробимо алгоритм, який матиме декілька етапів та кроків.

Етап 1. Запуск програми та шифрування.

На першому етапі користувач відкриває програму та знайомиться з інтерфейсом. Функціонал програми повинен бути доволі простим та зрозумілим. Обравши функцію шифрування користувач повинен мати вже записаний аудіо файл, який потрібно зашифрувати. Потрібно обрати шлях до записаного аудіо фійлу, шлях, куди файл шифруватиметься. Також програма повинна вивести вікно, в якому користувач введе пароль. Без введення цих даних програма не буде шифрувати файл, та запропонує їх вказати. Зробивши все правильно програма виведе вікно та розпочне шифрування. Після завершення користувач отримує зашифрований файл.

Етап 2. Шифрування з мікрофону.

На другому етапі користувач обирає функцію шифрування з мікрофону. Користувачу потрібно ввести пароль перед записом голосу також повинен вказати шлях, куди запишеться та зашифрується аудіо файл. Після введення даних, програма відкриє вікно для запису голосу з підключеного до комп’ютера мікрофону. Користувач натискає кнопку запуску запису та озвучує потрібну інформацію в мікрофон. Під час розмови програма шифрує дані та після закінчення користувач отримує зашифрований файл.

Етап 3. Дешифрування

На етапі дешифрування користувачу потрібно мати вже зашифрований файл в даній програмі. Обираючи функцію дешифрування файлу користувач повинен вказати пароль, який він ввів на етапі шифрування. Без паролю користувач не зможе дешифрувати файл. Ввівши пароль далі потрібно обрати шлях до зашифрованого файлу та шлях куди дешифрується аудіо файл. Після введення даних програма розпочне дешифрування аудіо файлу. Користувач може прослухати дешифрований файл після закінчення операції.

Розпишемо алгоритм шифрування по кроках та наведемо блок-схему алгоритму (рисунок 2.1):

Крок 1. Початок роботи з програмою, запуск програми.

Крок 2. Вибір функції шифрування.

Крок 3. Якщо ключ не введено перехід до кроку 4

Крок 4. Введення ключа користувачем

Крок 5. Ключ записується в змінну що бути доступним для всієї програми.

Крок 6. Створення екземпляру кодувальника

Крок 7. Передача кількості потоків

Крок 8. Програма запускає процес шифрування

Крок 9. Вибір кількості потоків

Крок 10. Читання даних та створення потоку

Крок 11. Передача даних в поток для шифрування

Крок 12. Запуск потоку

Крок 13. Отримуємо шифровані дані

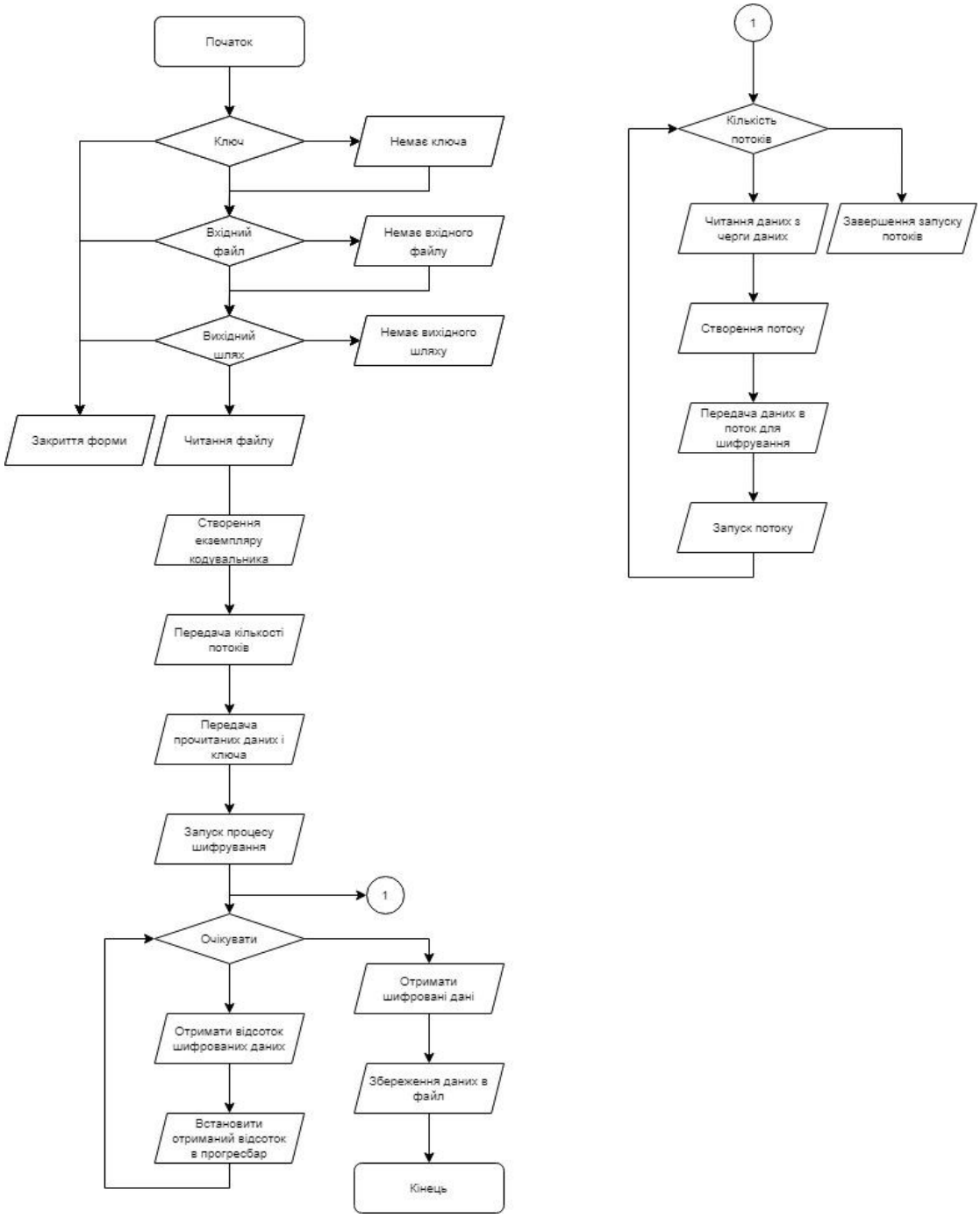

Рисунок 2.2 – блок-схема алгоритму шифрування

Алгоритм шифрування з мікрофону та блок-схема алгоритму (рисунок 2.3):

Крок 1. Початок роботи з програмою, запуск програми.

Крок 2. Вибір функції шифрування з мікрофону.

Крок 3. Якщо ключ не введено перехід до кроку 4

Крок 4. Введення ключа користувачем

Крок 5. Ключ записується в змінну щоб бути доступним для всієї програми.

Крок 6. Вибір вхідного і вихідного файлів

Крок 7. Створення нового потоку шифрування з мікрофону

Крок 8. Створення екземпляру бібліотеки PyAudio

Крок 9. Створення потоку аудіо з мікрофону

Крок 10. Створення файлу, шифрування даних і запис в файл

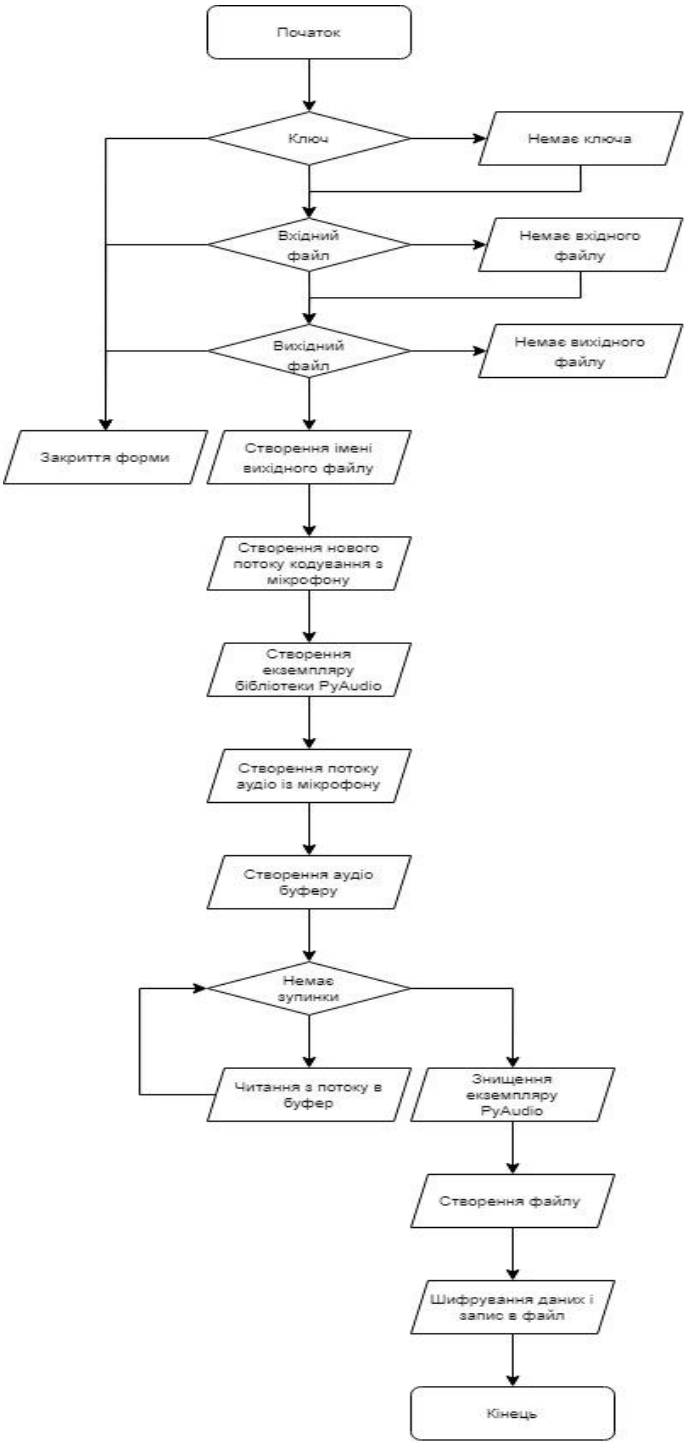

Рисунок 2.3 – блок-схема алгоритму шифрування з мікрофону

Алгоритм дешифрування та блок-схема алгоритму (рисунок 2.4):

Крок 1 Початок роботи з програмою, запуск програми.

Крок 2. Вибір функції дешифруванняя.

Крок 3. Якщо ключ не введено перехід до кроку 4

Крок 4. Введення ключа користувачем

Крок 5. Ключ записується в змінну що бути доступним для всієї програми.

Крок 6. Обираємо вхідний та вихідний файл

Крок 7. Читання файлу

Крок 8. Створення екземпляру кодувальника

Крок 9. Передача кількості потоків

Крок 10. Передача даних і ключа

Крок 11. Запуск процесу дешифрування

Крок 12. Отримуємо дешифровані дані

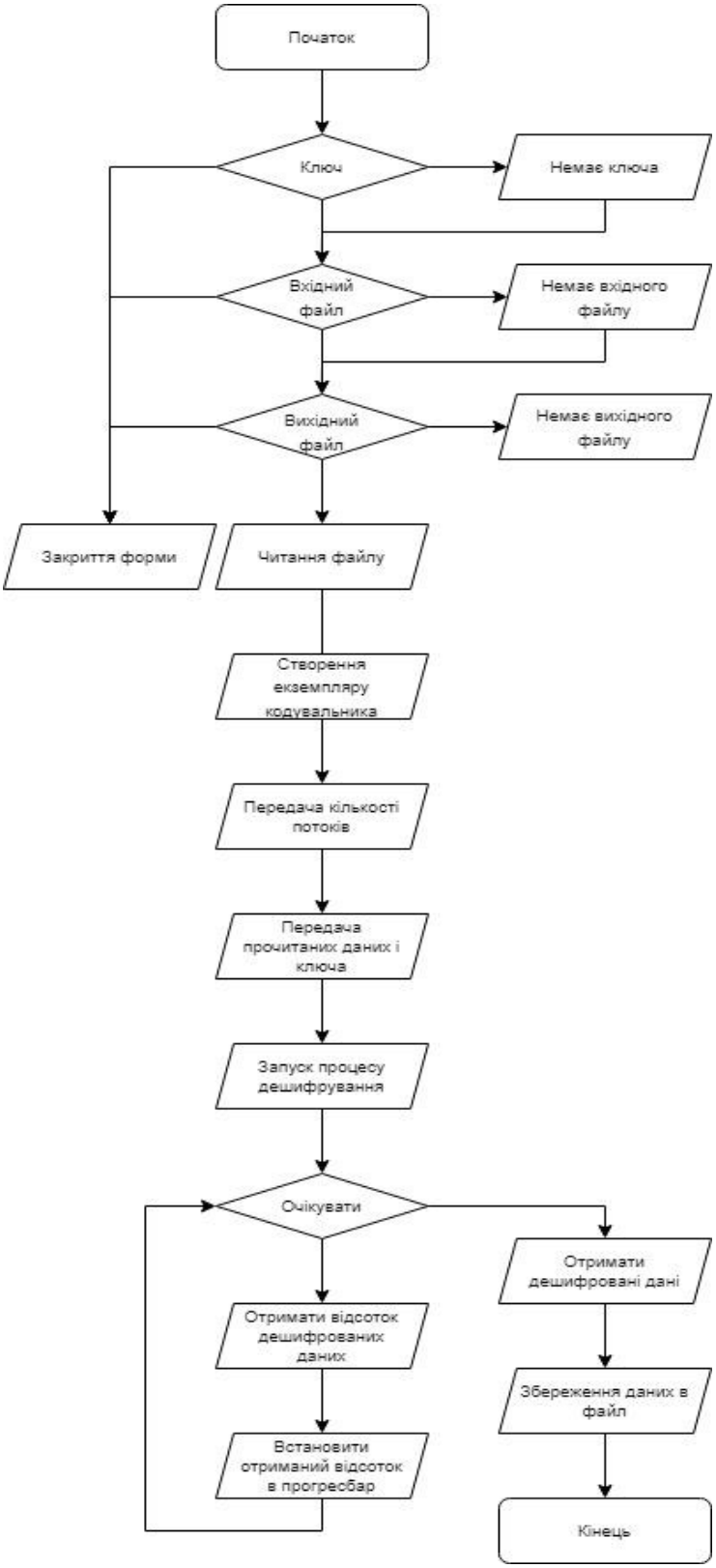

Рисунок 2.4 – блок-схема алгоритму дешифрування

Програма, написана за цим алгоритмом, дозволяє шифрувати як звукові файли, так і здійснювати безпосереднє потокове шифрування звуку з мікрофона,

під‘єднаного до комп‘ютера. Додаткові можливості програми – це вибір кількості потоків, частоти дискретизації та каналу.

2.2. Підвищення швидкодії шифрування потокового аудіо шляхом зміни середовища розробки

Програма розроблена для попередньої роботи виконувала шифрування методом XOR. Метод XOR є досить простим і потужним методом шифрування але його можна зламати якщо не використовувати довгі ключі. Для генерації використовувався ключ який генерував послідовність для генератора псевдовипадкових чисел. Для генерації псевдовипадкових чисел використовувався Алгоритм Блум - Блум – Шуба [28].

Цей генератор підходить для криптографії, але не для моделювання, тому що він недостатньо швидкий. Однак, він має високу стійкість, яка забезпечується якістю генератора виходячи з обчислювальної складності завдання факторизації чисел.

Функції шифрування і дешифрування в попередній програмі працювали швидко але недостатньо для того щоб працювати в реальному часі. Деякими чинниками що слугували малій швидкості роботи були: генератор псевдовипадкових чисел, реалізація обробки даних, і самим головним було це одно поточна обробка даних що забирала велику частину можливостей в програми.

Перед початком реалізації нової програми було проаналізовано вищенаведені складові роботи, що дало можливість зрозуміти потенційну область покращення і відкрило можливість для створення кращого алгоритму роботи програми. Для розробки цього рішення було використано мову Python.

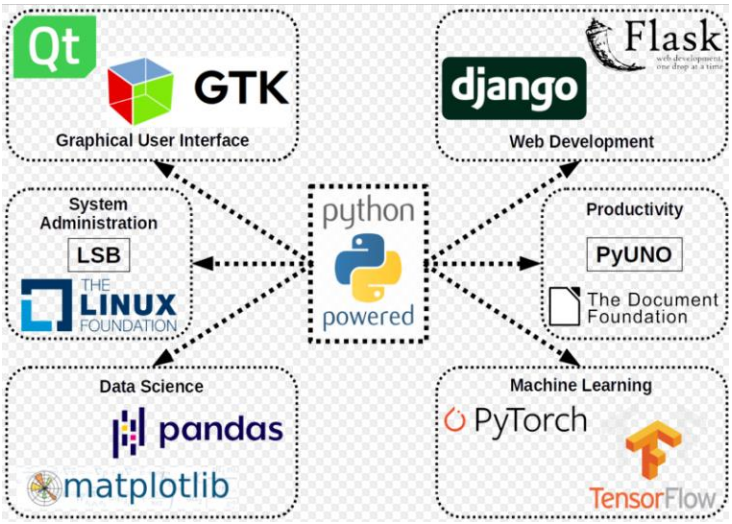

Рисунок 2.1 – Деякі можливості Python

Python широко використовується в різних середовищах. Будучи високо адаптованою мовою програмування. Python дозволяє легко розробляти та підтримувати проекти різного рівня складності. Найбільші переваги Python – це гнучкість, швидкий розвиток, масштабованість і відмінна продуктивність. NASA використовує Python в проектах, що стосується математичних розрахунків параметрів польоту [29].

Вибір саме Python покращує розробку програмного забезпечення тим, що дає вільні інструменти розробки, велику спільноту, та масу корисних бібліотек які досить просто інтегруються в проект що дозволяє швидко тестувати різні програмні реалізації алгоритмів і відповідно дає час на творчість в процесі озробки програмного забезпечення, а також час який можна використати на більш корисніші задачі а саме тестування. Перевагою Python ще є те, що він працює на динамічній компіляції, тобто його код по строково компілюється і виконується. Такий тип компіляції ще JIT.

Just-in-time compilation (JIT) (також відома як dynamic translation або run-time compilation) — компіляція «на льоту» — це технологія збільшення продуктивності програмних систем, що виконують програмний код, шляхом трансляції байт-коду в машинний код безпосередньо під час роботи програми. У такий спосіб досягається висока швидкість виконання за рахунок збільшення споживання пам'яті (для зберігання результатів компіляції) і витрат часу на компіляцію [30].

JIT компіляція є комбінацією двох основних методів трансляції в машинний код, інтерпретації та статичної компіляції, та наслідує якості обох підходів: переваги швидкості скомпільованого коду та гнучкості інтерпретатора поєднані з накладними витратами інтерпретації та компіляції коду. JIT-компіляція є підвидом динамічної компіляції що дозволяє використання технік адаптивної оптимізації, таких як динамічна рекомпіляція, використання інтерпретатором мікроархітектурних оптимізацій. JIT-компіляція підходить для динамічних мов програмування, оскільки системи компіляції реального часу можуть сконструювати пізньо-зв'язуванні типи даних та гарантувати безпеку.

2.3 Опис роботи проекту та його структура

Структуру проекту можна розділити на етапи: відтворення, запис, кодування, декодування, кодування з мікрофону, налаштування.

Розберемо кожен етап детальніше.

Відтворення.

Для створення функціоналу відтворення було використано бібліотеку PyGame яка дає потрібний функціонал запуску відтворення, паузи, зупинки відтворення і має можливість відкриття багатьох форматів. Для можливості роботи з файлами в візуальному інтерфейсі було створено файл form_record_play.py в якому реалізовано клас що наслідує клас візуального інтерфейсу і працює з його елементами в відповідній формі яка називається Ui_Dialog і знаходиться в файлі record_play.py який у свою чергу знаходиться в папці де лежать всі файли форм і візуального дизайну і має назву form_files.

Файл form_record_play.py який обробляє аудіо включає такі функції для відтворення аудіо як play_play, що відтворює аудіо якщо відтворення немає, тобто якщо не було запущено відтворення до цього або вже було запущено але зупинено. В інакшому випадку функція зупинить те аудіо що вже відтворюється і запустить його ще раз з початку. Якщо виникла якась помилка під час запуску або ж невірний формат програма викликає віджет що інформує користувача про помилку. Інакше

запускає відтворення. Серед можливих помилок можуть бути, помилка що вказує на невірний формат аудіо файлу який був переданий як шлях, або ж не вказаний шлях для запуску аудіо. Шлях потрібно вказувати в програмі в головному вікні. Також окрім прямого вказування в шлях відтворення буде поміщено шлях останнього записаного файлу, щоб мати можливість його запуску для відтворення без постійного перевибору в файлах системи [31].

Ще одна функція яка простіша але виконує свою функцію є за назвою pause_play, і виконує роль паузи коли відтворення іде і на неї натиснути або ж коли вже натиснули на паузу до то ще одне натискання виводить із паузи в програвання, тобто функція має обернену дію відносно натиску.

І найпростіша функція але не менш важлива має назву stop_play що виконує просто зупинку аудіо, ця зупинка не є паузою а є реальним зупиненням будь якого програвання аудіо. Щоб знову запустити потрібно натиснути на запуск аудіо а не на паузу.

Для того щоб можна було поставити на паузу в потрібний час або ж зупинити бібліотека PyGame реалізована з асинхронним запуском що дозволяє їй працювати паралельно до основної програми і відповідно мати можливість зупиняти відтворювати чи ставити на паузу програвання аудіо.

Ключовими перевагами над C++ є те, що запуск і керування потоком робиться буквально в пару строк що на порядок краще ніж в C++ і дає можливість абстрагуватись від реалізації і інших проблем з нею, та робити справжню програму використовуючи творчий підхід до її реалізації.

Для створення функціоналу запису використаємо бібліотеку PyAudio, з допомогою якої реалізуємо потрібний функціонал запису. А для збереження аудіо даних використаємо бібліотеку Wave. Функції які керують записом це play_encode_record для початку запису і функція stop_encode_record для зупинки запису і збереження його в потрібну директорію. Якщо шлях для збереження є то і запис можна буде почати але якщо шляху немає то і запис не можна почати. У випадку коли шлях для збереження не вказано і натиснуто на запис програма

повідомить про помилку і її причину, а саме відсутність можливості збереження через відсутність відповідного шляху для збереження.

Запис.

Процес запису починається з запуску функції запису яка викликає play_record і запускає її, далі ця функція починає відпрацьовувати код закладений в неї а саме перевіряє чи вказано шлях для збереження вихідного файлу, і якщо він є формує назву файлу із мітки часу яку вона отримує за допомогою вищеописаної бібліотеки datetime беручи поточний час і додаючи до шляху та іншої частини назви вихідного файлу, та додає формат. Форматом для збереження запису є WAV. Після цього вона запускає в новому потоці функцію self_record. Ця функція запускає запис а так як вона викликається в новому потоці то починає працювати відразу і програма також продовжує завершення попередньої функції що розблоковує її для подальших дій керування програмою від користувача за допомогою кнопок та інших засобів [32].

Функція self_record використовує для запису вищеописану бібліотеку за назвою pyaudio, створюючи екземпляр класу, ініціалізується поток даних, після чого створюється буфер, і запускається цикл який працює наступним чином: спочатку перевіряється чи є команда на зупинку запису, далі якщо команди немає запускається читання даних з потоку і запис їх в буфер, далі повторно і поки не буде команди на зупинку йтиме запис.

Кодування.

Процес кодування починається з відповідної форми яка запускає кодування. Далі після запуску іде перевірка чи відповідні шляхи до файлів і змінні не пусті. Щоб почати кодування перевіряються такі змінні як: шлях до вхідного файлу, далі шлях до директорії збереження закодованого файлу, далі ключ. Якщо всі змінні є продовжуємо роботу, інакше програма виводить помилки. Для виведення помилок використовуємо відповідну функцію яка використовує бібліотеку PyQt5.

Якщо все вірно то продовжуємо роботу із читання файлу за допомогою бібліотеки numpy, в ній використовуємо функцію fromfile в якій першим параметром йде файл а другим параметер даних за назвою dtype,. Після цього ми

отримуємо кількість потоків за допомогою якого будемо обробляти дані. Далі створюємо екземпляр класу описаний в файлі cipher_xor_multithread.py, за назвою cipher_xor_multithread. Цей клас реалізує роботу з ключами, кодування і інші допоміжні функції. Також в файлі є функція xor_array, яка і реалізує кодування та декодування так як алгоритм кодування є оберненим.

Процес кодування починається з створення екземпляра класу та передачі йому кількості потоків яка буде обробляти дані, далі передача спеціальною функцією ключа, далі передаються дані, після йде запуск кодування, так як кодування багато поточне і асинхронне процес роботи програми продовжується і заходить в спеціальний цикл, в якому перевіряється стан роботи програми. Під час перевірки процесу йде отримання відсотку завершеної роботи, та передача в прогрес бар його значення для інформування користувача про стан роботи програми, і після пауза для того щоб не перенавантажувати перевірками процесор комп’ютера, пауза триває 0.2 секунди. Якщо перевірка повертає що процес кодування завершено то цикл завершується та відбувається отримання фінальних даних [33].

Після отримання даних йде отримання назви вхідного файлу та збереження даних за вказаною директорією і назвою вхідного файлу із зміненою назвою інформуючою про те що це закодований файл.

Далі збереження файлу за допомогою спеціальної функції із numpy за назвою tofile яка викликається в отриманих даних як метод класу типу цих даних. Після цього процес кодування файлу завершено.

Сам процес кодування і паралелізації виглядає наступним чином. Після створення екземпляру файлу і передачі кількості потоків, та після передачі необхідних даних а саме ключа і самих даних які потрібно кодувати відбувається запуск функції start_threads. Ця функція перевіряючи вже передані дані при успішності їх перевірки запускає функцію start_processing, а дані які вона перевіряє це кількість потоків, ключ, та перевірка на наявність запуску кодування. Функція start_processing в свою чергу після запуску отримує глобальні змінні, ці змінні є

членами вже ініціалізованого класу Queue бібліотеки multiprocessing, для передачі даних.

Декодування.

Так як ми використовуємо XOR, то кодування є оберненим до декодування, це означає що за різні по назві операції відповідає одна і та ж функціональність. Функціонал кодування обернений до декодування тому ми використовуємо однаковий алгоритм. Але так як ми заклали особливу роботу з ключами а саме його оброблення то можемо кодувати різною кількістю потоків, а декодувати також різною їх кількістю, що дозволяє виконувати роботу з програмою на різних пристроях з різними процесорами і кількістю потоків на них.

Кодування з мікрофону.

Функція кодування з мікрофону реалізована в класі форми запису і відтворення. Запуск кодування з мікрофону починається із функції play_encode_record яку запукаємо. Задача функції play_encode_record в тому щоб перевірити директорію для збереження файлу, перевірити ключ. Якщо одиз з параметрів відсутній а саме ключ чи шлях для збереження файлів, тоді відбувається повідомлення користувача про проблему, в іншому випадку функція продовжує роботу [34].

Далі ця функція використовуючи бібліотеку datetime виконує отримання часу для того щоб сформувати ім’я файлу. Далі функція формує шлях і назву файлу та з’єднує їх в одну строку передаючи її для новоствореного потоку, разом з функцією яка буде виконуватись а саме self_record_encode та ключом. Після цього сформований з цієї функції і параметрів потік запускається і функція завершується передаючи керування програмі. Ключовою особливістю кодування з мікрофону є об’єднання запису з мікрофону з кодуванням. Для цього було використано код запису, після запису файл який було збережено відкривається і виконується кодування, але ключовою особливістю в порівнянні з попередньою програмою він це робить в реальному часі.

Алгоритм оптимізований настільки що аудіо стандартної довжини обробляє миттєво в порівнянні з попередньою версією програми, при тому що написаний він

на мові python, яка себе ніколи не позиціонувала як швидка так як навіть js в аналогічних задачах швидший за неї. Але йому не вистачає таких можливостей які має python, а саме можливість багато поточності, такої кількості бібліотек, і повної свободи в їх використанні [35].

Налаштування.

Форма для налаштувань надає можливості налаштувати кількість потоків, частоту дискретизації, і кількість аудіо каналів. По замовчанню вони встановлені в стандартні значення які працюють. Якщо потрібно змінити кількість потоків то це потрібно робити перед запуском процесу кодування, щоб мати можливість передати кількість потоків перед початком кодування і клас кодування міг ініціалізуватись потрібним значенням.

Вищеописаний процес розробки дозволив скоротити час розробки програмного забезпечення на суттєву величину. Вибір мови програмування скоротив час розробки в більше ніж 2 рази. Процес тестування був скорочений також, тільки завдяки продуманим і швидким бібліотекам які як зазвичай пишуться на C++, та працюють значно швидше ніж зазвичай. Також варто зауважити що код може працювати без створення файлу .exe лише застосовуючи бібліотеки та інтерпретатор python. Але ми реалізували за допомогою бібліотеки pyinstaller файл .exe для запуску в операційній системі Windows.

2.4 Висновки

Таким чином, в даному розділі було вдосконалено алгоритм шифрування потокового аудіо шляхом зміни середовища розробки.

Також поетапно описано роботу алгориту підвищення швидкодії потокового шифрування аудіо інформації, структуру та роботу проекту.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ ЗАХИСТУ ЗВУКОВОЇ ІНФОРМАЦІЇ

3.1 Програмна реалізація вдосконаленого алгоритму

Почнемо тестування з головного вікна програми. Зображення головного вікна зображено на рисунку 3.1.

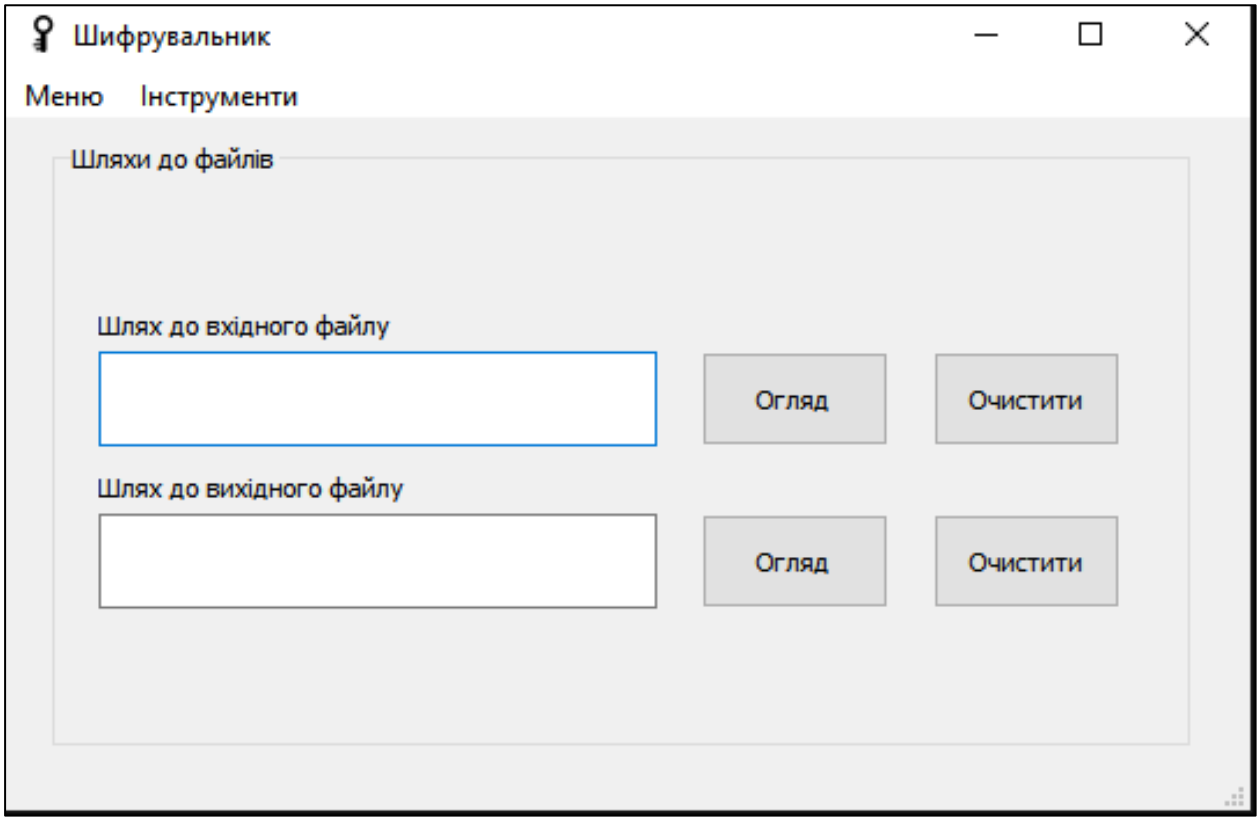

Рисунок 3.1 – Головне вікно програми

Для початку роботи потрібно вказати вхідний та/або вихідний шляхи. Якщо не вказати то операцію зв’язану з цим шляхом виконати буде неможливо. Головне вікно з обраними шляхом вхідного файлу, та шляхом вихідного файлу зображено на рисунку 3.2.

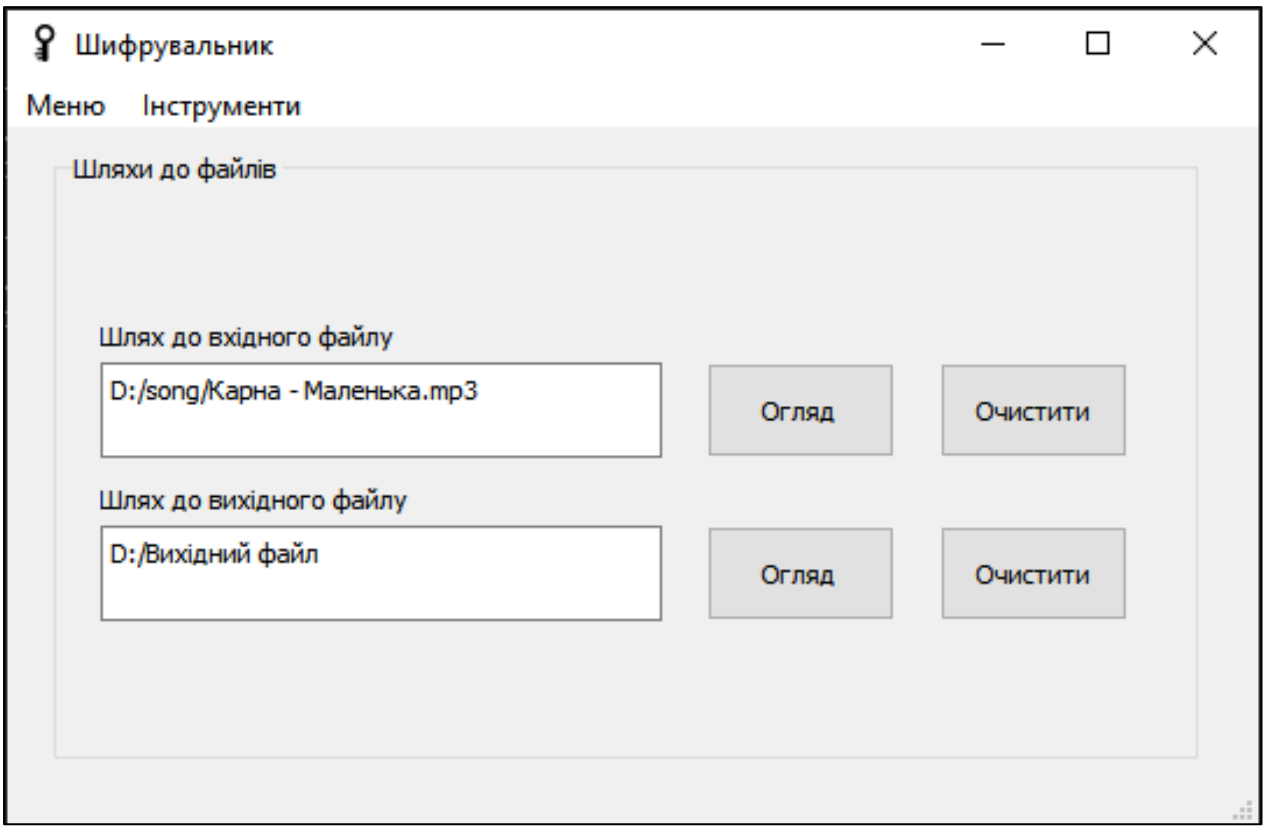

Рисунок 3.2 – Вибір шляху до файлу

Форма налаштувань дозволяє налаштувати канали, частоту дискретизації, та кількість потоків. Зображення стандартних налаштувань представлено на рисунку 3.3.

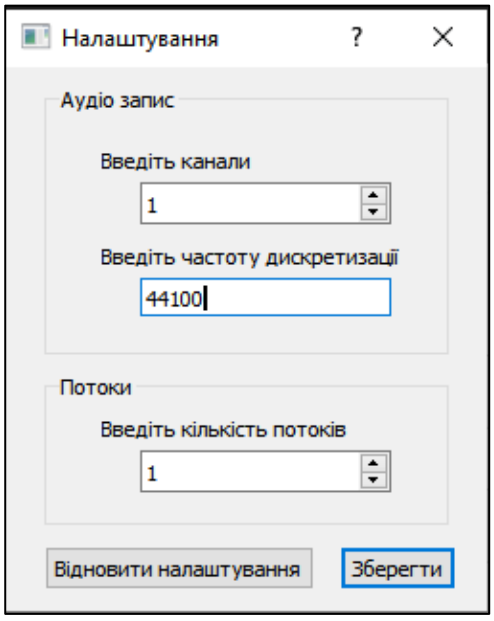

Рисунок 3.3 – Стандартні налаштування

Після зміни потрібно зберегти налаштування натиснувши на відповідну кнопку або ж відновити налаштування натиснувши на іншу відповідну кнопку. Налаштування будуть встановлені для всіх форм і параметрів, і будуть працювати до закриття програми чим ми відповідно і скористаємось.

Для того щоб вводити ключ створено відповідну форму введення ключа. Ця форма дозволяє ввести його в ручному режимі або вставити, а також біля поля введення є галочка яка дозволяє робити ключ видимим або зірочками. Для того щоб відкрити форму потрібно вибрати в меню відповідний пункт і натиснути. Відкриття форми представлено на рисунку 3.4.

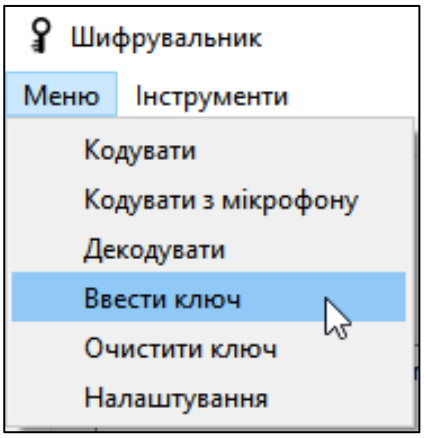

Рисунок 3.4 – Відкриття форми введення ключа

Форма введення ключа представлена на рисунку 3.5.

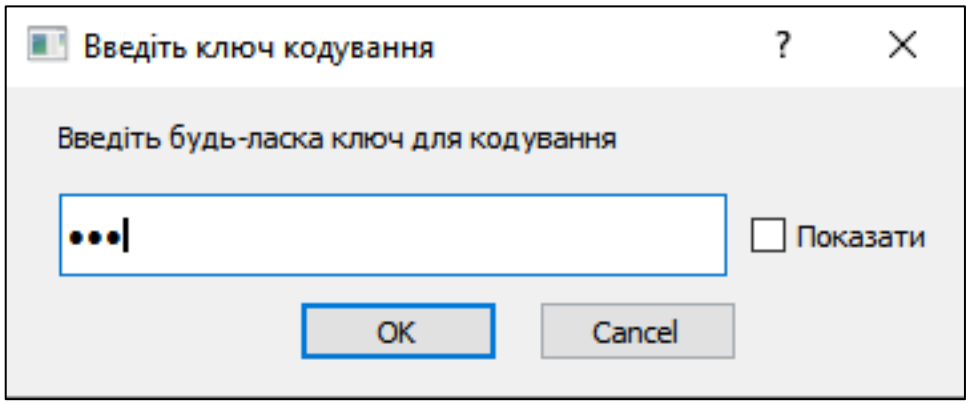

Рисунок 3.5 – Форма введення ключа

Після того як ми ввели ключ або змінили існуючий, потрібно натиснути на ОК, тоді введений ключ збережеться, в іншому випадку закрити вікно або натиснути кнопку Cancel.

Після того як ми вибрали відповідні шляхи а саме шлях до вхідного файлу і шлях для збереження ми можемо натиснути на відповідну кнопку в меню. Форма кодування представлена на рисунку 3.6.

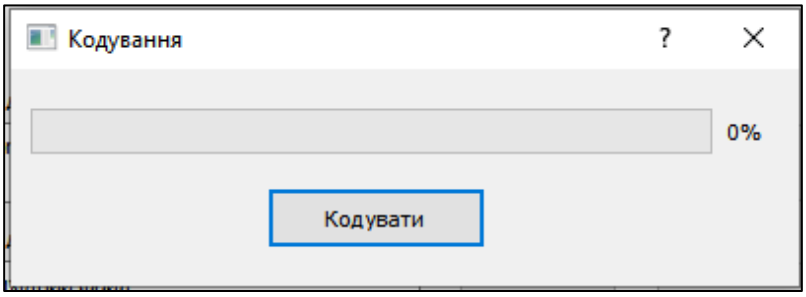

Рисунок 3.6 – Форма кодування

Якщо все вірно, і шляхи вказані правильно то запуститься кодування на тій кількості потоків які ми вказали.

В іншому випадку якщо ми не вказали всі необхідні параметри буде виведено помилку, яка вкаже на деяку змінну, яку ми не вказали, і форма буде закрита після підтвердження відповідною кнопкою. Запустити в такому разі кодування не вдасться. Нижче зображено повідомлення про те що ключ не вказано, і дійсно ми ще не вказували ключа, тому повідомлення нам про це і каже. В іншому випадку буде запуск або виведено помилку подібну до цієї. Форму помилки зображено на рисунку 3.7.

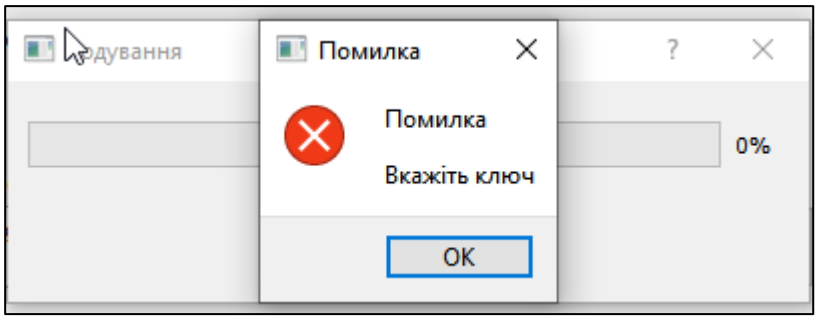

Рисунок 3.7 – Помилка кодування

Після того як ми вкажемо ключ відбудеться кодування, і якщо все успішно ми побачимо відповідне зображення, а саме прогрес бар буде заповнено повністю на 100% і ми матимемо можливість керувати програмою. Форма успішного кодування зображена на рисунку 3.8.

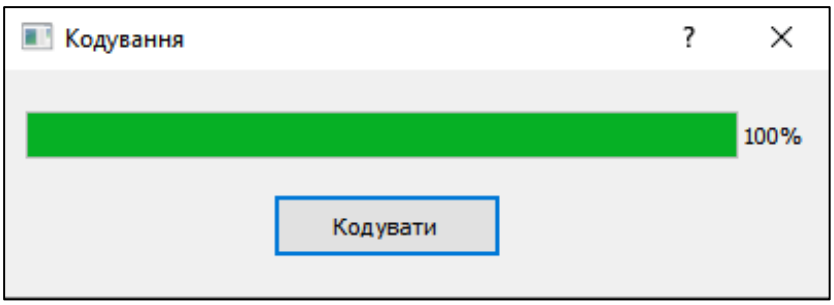

Рисунок 3.8 – Успішно завершене кодування

Після успішно завершеного кодування, в папці шлях якої ми раніше вказали буде лежати закодований файл.

Після кодування можемо передати файл в потрібне місце а далі виконати з ним наступну операцію, а саме декодування, щоб перетворити його в потрібний файл для прослуховування. Ключ не міняємо щоб знову не вводити.

Щоб почати декодувати виберемо потрібний файл так само як вибирали вхідний файл. Якщо ми вибираємо не закодований файл то операцією декодування ми закодуємо файл. Пояснення цьому таке. Згідно алгоритму кодування а в нас він XOR, знаємо що він обернений, тобто операція кодування і декодування це одна і та ж сама операція просто повторно проведена. І відносно цього ми маємо, що можна провести операцію кодування 2 рази і файл буде декодовано але в програмі відмінність лише в тому як буде називатись файл, щоб відрізнити його від інших файлів. Далі нам потрібно натиснути в меню на кнопку декодувати щоб відкрити необхідну форму.

Після відкриття ми потрапляємо на ту ж саму форму що і під час кодування але назви змінено, і різниця в наданні назв при збереженні файлів. Далі відбувається кодування після натиску кнопки декодувати. Форма декодування зображена на рисунку 3.9.

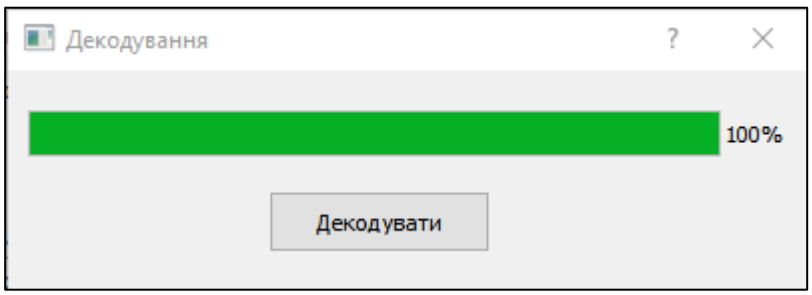

Рисунок 3.9 – Успішне декодування

Отже кодування, декодування виконані успішно, та швидко. Код працює відповідно правильно що доводять наші тести, а також ми можемо вказувати важливі додаткові параметри для того щоб змінювати хід роботи програми. Файли були правильно оброблені за допомогою одного потоку але це не означає що вони можуть бути оброблені лише таким чином, ми кодувати та декодувати різною кількістю потоків.

Кодування з мікрофону починається із меню і запуску відповідної форми через натиск на пункт меню. Форма запису з мікрофону представлена на рисунку 3.10.

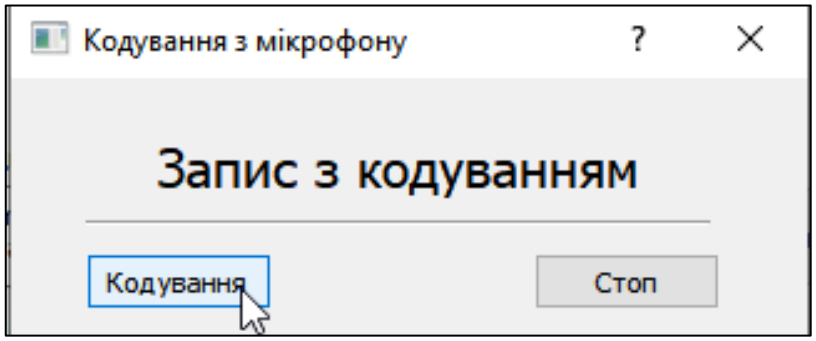

Рисунок 3.10 – Форма запису з мікрофону

Після кодування в директорії яку було вибрано на початку, з’являється файл, який має закодований запис в форматі WAV, що зображено нижче, і говорить про успішну операцію запису з кодуванням.

Згідно розробленого алгоритму у розділі 2 опишемо процес програмної реалізації додатку.

Важливі елементи, які використані в написанні коду:

QGroupBox – використовуються для надання ідентифікованого групування для інших елементів управління. Як правило, для поділу форми виконуваних функцій використовуються поля груп.

QPushButton – кнопка яку можна натискати наприклад щоб викликати функції.

QTextEdit – поле введення і виведення текстової інформації.

QLabel – виведення тексової інформації (мітка).

QMenuBar – призначений для додавання до програми головного меню, елемента, без якого не обходиться жодна з програм.

QProgressBar – показує прогрес виконання функцій.

QCheckBox – компонент вибору будь-якої опції шляхом установки або зняття галочки.

QLineEdit – поле введення і виведення текстової інформації в одну строку.

QSpinBox – поле для числового значення яке можна змінювати.

Line – лінія роздільник, візуальне розділення лінією [36].

Починаємо з етапу кодування вже записаного файлу. Для початку потрібно реалізувати введення паролю, відкривається вінко запису паролю та можливість самого вводу паролю відбувається за допомогою функцій enter_key, яка викликає показ форми введення ключа.

```
def enter_key(self):
    self.form_enter_key_app.show()
```

Після введення ключа його потрібно зберегти. Для цього є функція save_key [37].

```
def save_key(self):
    if (str(self.ui.lineEdit.text()) != ""):
        settings.key = self.ui.lineEdit.text()
        self.close()
```

Але якщо збереження не потрібне, то можна відмінити натиснувши наступну функцію. Яка закриє форму введення ключа.

```
def cancel(self):
    self.close()
```

Після введення ключа потрібно задати шлях до вхідного файлу, та шлях для збереження шифрованого файлу. Операції введення шляхів. Робляться на головному вікні.

```
def open_input_file(self):
    file_name_and_dir = QtWidgets.QFileDialog.getOpenFileName(self, "Виберіть вхідний файл", None, "")
    if (str(file_name_and_dir[0]) != ""):
        self.ui.textEdit.setPlainText(str(file_name_and_dir[0]))
        settings.input_file_name_and_dir = str(file_name_and_dir[0])

def open_output_dir_save(self):
```

```
dir_from_save_output_file = QtWidgets.QFileDialog.getExistingDirectory(
    self,
    "Виберіть директорію для збереження вихідного файлу",
    expanduser("~"),
    QtWidgets.QFileDialog.ShowDirsOnly
)
if (str(dir_from_save_output_file) != ""):
    self.ui.textEdit_2.setPlainText(str(dir_from_save_output_file))
    settings.output_dir_save = str(dir_from_save_output_file)
```

Для того щоб очистити шляхи, потрібно натиснути відповідні кнопки які очищують шляхи.

```
def clear_input_filename_path(self):

    self.ui.textEdit.setPlainText("")

    settings.input_file_name_and_dir = ""

def clear_output_filename_path(self):

    self.ui.textEdit_2.setPlainText("")

    settings.output_dir_save = ""
```

Після введення потрібних шляхів потрібно запустити кодування в меню, натиснувши відповідну кнопку, яка відкриє форму.

```
def encode(self):

    settings.title_form_encode_decode_progres_bar = "Кодування"

self.form_encode_decode_progres_bar_app.set_window_title(settings.title_form_encode_decode_progres_bar, "Кодувати")

    self.form_encode_decode_progres_bar_app.set_command(1)

    self.form_encode_decode_progres_bar_app.clearbar()

    self.form_encode_decode_progres_bar_app.show()
```

Після цього буде відкрито форму кодування з прогресс баром, який вказує на значення закодованого файлу. Щоб запустити потрібно натиснути відповідну кнопку.

```
def play_encode(self):
    if(settings.is_empty(settings.input_file_name_and_dir) or settings.is_empty(settings.output_dir_save) or settings.is_empty(settings.key)):
        exit_command = 0
        if (settings.is_empty(settings.input_file_name_and_dir)):
            settings.ErrorMessage("Вкажіть шлях до вхідного файлу")
            exit_command = 1
        if (settings.is_empty(settings.output_dir_save)):
            settings.ErrorMessage("Вкажіть директорію для збереження файлу")
            exit_command = 1
        if (settings.is_empty(settings.key)):
            settings.ErrorMessage("Вкажіть ключ")
            exit_command = 1
        if (exit_command == 1):
            self.close()
    else:
        self.key = settings.key
        try:
            with open(settings.input_file_name_and_dir, "rb") as f:
                numpy_data = np.array(np.fromfile(f, dtype=np.longlong))
```

```
except IOError:
    print('Error While Opening the file!')
# передаю дані
self.numers_of_threads = settings.maximum_number_threads
#len(os.sched_getaffinity(0))
start_time = time.time()
self.th = cipher_xor_multithread(self.numers_of_threads)
self.th.set_key(self.key)
self.th.set_all_data(numpy_data)
self.th.start_threads()
while not self.th.get_all_wait():
    self.ui.progressBar.setValue(self.th.get_percent_compete())
    time.sleep(0.2)
time.sleep(0.1)
self.ui.progressBar.setValue(self.th.get_percent_compete())
self_data = self.th.get_output_data()
base = os.path.basename(settings.input_file_name_and_dir)
self_data.tofile(settings.output_dir_save + "/" + "encode_" + base)
self.ui.progressBar.setValue(int(100))
print("--- %s seconds ---" % (time.time() - start_time))
```

Після запуску буде створено та кількість потоків яка вказана за замовчанням, якщо ми не міняли її кількість а саме один поток. Цей поток має функцію яка виконує кодування за допомогою функції представленої нижче.

```
def xor_array(index_data, data, mkey, size_min_complete_block, queue_complet_block_data, data_len_is_complet, process_complete):
    output_mas = np.empty(len(data), dtype=np.longlong)
    current_index_key = 0
    data_len_is_complet_local = 0
    for i in range(len(data)):
        if (data_len_is_complet_local >= size_min_complete_block):
            data_len_is_complet.value = data_len_is_complet.value + data_len_is_complet_local
            data_len_is_complet_local = 1
        else:
            data_len_is_complet_local += 1
        current_key = mkey[current_index_key]
        if (current_index_key == (len(mkey) - 1)):
            current_index_key = 0
        else:
            current_index_key += 1
        real_key = ord(current_key)
        output_mas[i] = data[i] ^ real_key
    queue_complet_block_data.put([index_data, output_mas.tolist()])
    process_complete.put([index_data, "complete"])
```

Для кодування з мікрофону, потрібно натиснути відповідну кнопку в головному меню. Вона викличе відповідну функцію представлену нижче.

```
def encode_microfone(self):
```

```
        settings.title_form_record_play = "Кодування з мікрофону"
        self.form_record_play_app.set_window_title(settings.title_form_record_play)
        self.form_record_play_app.ui.pushButton.setText("Кодування")
        self.form_record_play_app.set_status_form(3)
        self.form_record_play_app.show()
```

Після її запуску буде відкрито ту ж саму форму яка використовується для запису але змінена, для саме кодування з мікрофону.

```
    def play_encode_record(self):
        exit_command = 0
        if (settings.is_empty(settings.output_dir_save)):
            settings.ErrorMessage("Вкажіть директорію для збереження файлу")
            exit_command = 1
        elif (settings.is_empty(settings.key)):
            settings.ErrorMessage("Вкажіть ключ")
            exit_command = 1
        if (exit_command == 1):
            self.close()
        self.record_play_is_true = True
        self.record_stop_is_true = False
        now = datetime.datetime.now()
        filename_to_save = "my_record_encode_file_time_" + now.strftime("%Y_%m_%d_%H_%M_%S") + "_true.wav"
        save_var_path = settings.output_dir_save + "/" + filename_to_save
```

```
        my_thread = threading.Thread(target=self.self_record_encode, args=(save_var_path, settings.key,))
        my_thread.start()
    def self_record_encode(self, filename_and_path, mkey):
        chunk = 1024
        sample_format = pyaudio.paInt16
        channels = settings.audio_channels
        fs = settings.audio_framerate
        p = pyaudio.PyAudio()
        stream = p.open(format=sample_format,
                        channels=channels,
                        rate=fs,
                        frames_per_buffer=chunk,
                        input=True)
        frames = []
        while True:
            if (self.program_destoi):
                break
            if (self.record_stop_is_true):
                break
            data = stream.read(chunk)
            frames.append(data)
        stream.stop_stream()
```

```
stream.close()

p.terminate()

f_mf = open(filename_and_path, "w")

f_mf.close()

wf = wave.open(filename_and_path, 'wb')

wf.setnchannels(channels)

wf.setsampwidth(p.get_sample_size(sample_format))

wf.setframerate(fs)

wf.writeframes(b".join(frames))

wf.close()

self.key = mkey

try:

    with open(filename_and_path, "rb") as f:

        numpy_data = np.array(np.fromfile(f, dtype=np.longlong))

except IOError:

    print('Error While Opening the file!')

if os.path.exists(filename_and_path):

    os.remove(filename_and_path)

# передаю дані

self.numers_of_threads = settings.maximum_number_threads

self.th = cipher_xor_multithread(self.numers_of_threads)

self.th.set_key(self.key)
```

```
        self.th.set_all_data(numpy_data)
        self.th.start_threads()
        while not self.th.get_all_wait():
            time.sleep(0.2)
        self_data = self.th.get_output_data()
        base = os.path.basename(filename_and_path)
        self_data.tofile( settings.output_dir_save +"/"+ "encode_" +base)
        settings.input_file_name_and_dir = settings.output_dir_save +"/"+ "encode_" +base
```

Для відтворення використовується функція з цієї ж форми. Відтворення запускається із головного вікна відповідною кнопкою.

```
    def play(self):
        settings.title_form_record_play = "Відтворення"
     self.form_record_play_app.ui.pushButton.setText(settings.title_form_record_play)
        self.form_record_play_app.set_status_form(1)
        self.form_record_play_app.show()
    def pause_play(self):
        if (settings.is_empty(settings.input_file_name_and_dir)):
            settings.ErrorMessage("Немає файлу який потрібно програти, виберіть його будь-ласка")
        else:
            if (self.play_is_true):
                if (self.pause):
                    pg.mixer.music.unpause()
```

```
            self.pause = False
        else:
            pg.mixer.music.pause()
            self.pause = True
```

Для простого запису використовується наступна функція, яка поміщено в формі відтворення запису, і запускається з головного вікна.

```
    def record(self):
        settings.title_form_record_play = "Запис"
        self.form_record_play_app.ui.pushButton.setText(settings.title_form_record_play)
        self.form_record_play_app.set_status_form(2)
        self.form_record_play_app.show()
    def play_record(self):
        if (self.record_play_is_true == False):
            if (settings.is_empty(settings.output_dir_save)):
                settings.ErrorMessage("Вкажіть директорію для збереження записаного файлу")
            else:
                self.record_play_is_true = True
                self.record_stop_is_true = False
                now = datetime.datetime.now()
                filename_to_save = "my_record_file_time_" + now.strftime("%Y_%m_%d_%H_%M_%S") + "_true.wav"
                save_var_path = settings.output_dir_save + "/" + filename_to_save
                settings.input_file_name_and_dir = save_var_path
```

```
            my_thread = threading.Thread(target=self.self_record, args=(save_var_path,))
            my_thread.start()
    def self_record(self, filename_and_path):
        chunk = 1024
        sample_format = pyaudio.paInt16
        channels = settings.audio_channels
        fs = settings.audio_framerate
        p = pyaudio.PyAudio()
        stream = p.open(format=sample_format,
                        channels=channels,
                        rate=fs,
                        frames_per_buffer=chunk,
                        input=True)
        frames = []
        while True:
            if (self.program_destoi):
                break
            if (self.record_stop_is_true):
                break
            data = stream.read(chunk)
            frames.append(data)
        stream.stop_stream()
```

```
stream.close()

p.terminate()

f_mf = open(filename_and_path, "w")

f_mf.close()

wf = wave.open(filename_and_path, 'wb')

wf.setnchannels(channels)

wf.setsampwidth(p.get_sample_size(sample_format))

wf.setframerate(fs)

wf.writeframes(b".join(frames))

wf.close()
```

3.2 Тестування роботи програмної реалізації вдосконаленого алгоритму

Ключові методи дослідження роботи програмного продукту це вимірювання швидкості роботи програми в однопотоковому режимі, і також в багатопотоковому режимі. Однією з ключових характеристик вдосконаленого алгоритму є багатопотокова обробка даних. Дані в багатопотоковому режимі можуть, кодуватись так і декодуватись, що відповідно ставить завдання на вимірювання різниці, між однопотоковою версією та багатопотоковою [38].

Для вирішення цієї задачі ми будемо використовувати бібліотеку time, яка дозволяє засікати час. В ній будемо використовувати функцію time(), за допомогою якої можна отримати час відрахований в секундах від деякої події.

Для того щоб відрахувати поставимо засікання в точках початку і закінчення роботи алгоритму, та отримаємо різницю. Різниця буде значенням в секундах часу виконання. Цю різницю ми запишемо як ключову і проведемо відповідно декілька тестів [39].

Проведемо тест, який покаже швидкість виконання кодування, на мелодії розміром 91.9 мегабайт.

Ключ має довжину 3 символи. Час виконання був засічений за допомогою бібліотеки time, та склав 9.02 секунд. Кодування проводилося на 1 потокові. Час витрачений на кодування зображено на рисунку 3.11.

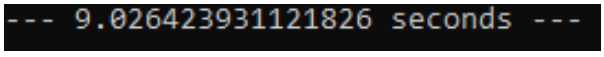

Рисунок 3.11 – Час витрачений на кодування

Далі проведемо тестування декодування, файл який будемо декодувати це цей файл який ми тільки що кодували. Час витрачений на декодування зображено на рисунку 3.12.

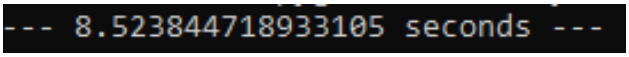

Рисунок 3.12 – Час витрачений на декодування

Наступні тести проведемо на 4 потоках, щоб мати можливість порівняти швидкість виконання. Файли для тесту будуть такі самі. Час кодування та декодування зображено на рисунках 3.13 та 3.14 відповідно.

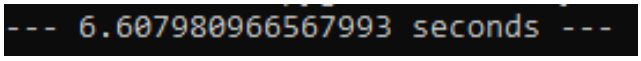

Рисунок 3.13 – Час витрачений на кодування

Рисунок 3.14 – Час витрачений на декодування

Як бачимо тести показали що прискорення роботи кодування відбувається, при збільшенні кількості потоків які, обробляють дані. Але час дуже мало відрізняється а саме близько двох секунд. Така відмінність полягає лише в тому що файл не є великим, і однопоточна версія також досить швидко працює. Але основний час обробки іде на розбиття на блоки, та збирання вихідного файлу.

Далі проведемо тестування програми без розпаралелювання.

Для тесту беремо той самий файл розміром 91.9 мегабайт. В програмі без розпаралелювання неможливо було обрати потоки для кодування та декодування, тому витрачений час набагато більший за вдосконалениу версію алгоритму. Час кодування та декодування файлу зображено на рисунках 3.15 та 3.16 відповідно.

Рисунок 3.15 – Час кодування файлу

Рисунок 3.16 – Час декодування файлу

Як бачимо швидкість кодування двох програм значно відрізняється. Різниця часу витраченого на кодування двох програм складає 25 секунд. А різниця часу декодування складає 17 секунд.

Кодування на багатьох потоках прискорює обробку але час витрачається не тільки на обробку а й на перетворення даних в потрібну форму, а саме розбиття на

блоки та перетворення ключів. Але також має свій вклад створення потоків на яке також використовується час. І в кінці закодовані блоки потрібно об’єднати на що також іде час. В результат ми використовуємо час на всі операції з даними що зрівнює деякі результати створюючи фінальний час обробки даних

3.3 Експерементальні дослідження швидкодії вдосконаленого алгоритму шифрування

Створена програма досягнула свого результату. Результат роботи програми це закодований файл який можна вільно передавати не боячись зламу. Тестування показало що основна із ключових особливостей проявила себе досить гарно. А саме багатопотокове кодування дає прискорення обробки даних. Згідно з останніх тестів вийшло що відбувається прискорення на 30% відносно однопотокової версії. Ключова особливість це розбиття на блоки що дозволяє створювати потоки із виділеним блоком, та не плутати блоки. Кожен блок даних помічається своїм індексом, і після обробки закодований поміщається в свою позицію відносно індексу, який йому було присвоєно при створенні. Загальний графік прискорення відносно багатопоточності виглядає так що підкоряється закону Амдала [40].

Закон амдала інтерпритується наступним чином. Оптимальним прискоренням від розпаралелювання могло б бути лінійне — збільшення кількості процесорів вдвічі має вдвічі скорочувати час виконання. Наступне збільшення кількості процесорів вдвічі мало б знову прискорювати програму. Тим не менш, лиш кілька паралельних алгоритмів досягають такого прискорення. Більшість з них мають майже лінійне прискорення при невеликій кількості процесорів, яке сповільнюється до константи при великій кількості обчислювальних елементів.

Потенційне прискорення алгоритму при збільшенні числа процесорів задається законом Амдала, що вперше був сформульований Джином Амдалем у 1960-тих. Він стверджує, що невелика частина програми що не піддається паралелізації обмежить загальне прискорення від розпаралелювання [41]. Будь-яка велика математична чи інженерна задача зазвичай буде складатись з кількох частин що можуть

виконуватись паралельно, та кількох частин що виконуються тільки послідовно. Такий зв’язок можна задати формулою 3.1 яка пояснює наглядніше цей закон.

$$S = \frac{1}{1-P} \tag{3.1}$$

де S — прискорення програми (як відношення до її початкового часу роботи)

P — частка яку можна виконувати паралельно.

На рисунку 3.17 зображено графік залежності прискорення від кількості потоків.

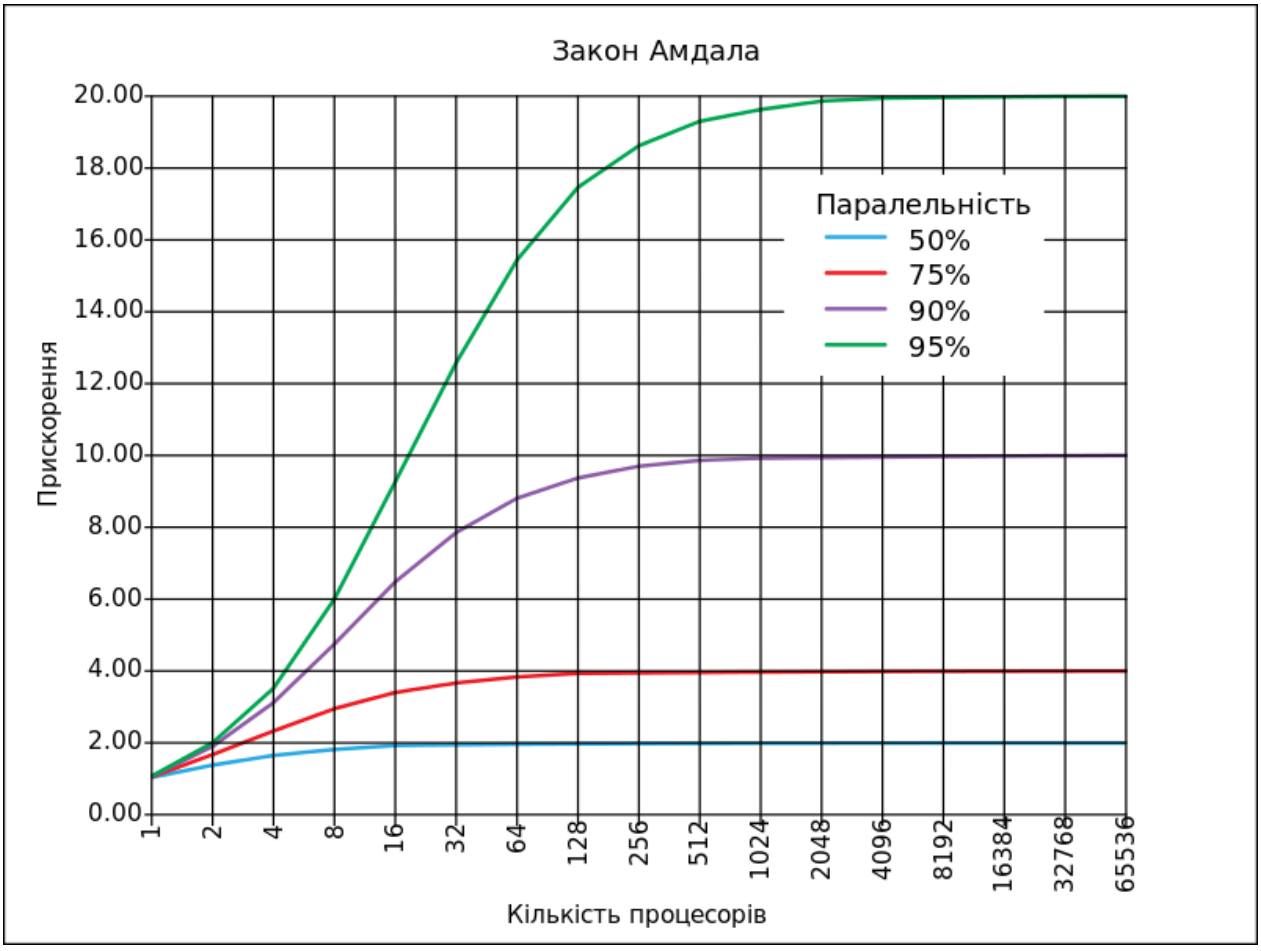

Рисунок 3.17 - Залежність прискорення від кількості потоків

Якщо послідовна частина програми виконується 10 % всього часу роботи, ми не можемо прискоритись більш ніж в 10 разів, незалежно від того скільки процесорів застосуємо. Це ставить верхню межу корисності збільшення кількості обчислювальних елементів.

Закон Амдала базується на припущенні того, що задача має фіксований розмір, і що розмір послідовної частини незалежний від кількості процесорів, при тому що не

всі задачі можуть так описуватись, але конкретно наша задача саме так і описується, так як частина яка може бути розпаралелена має фіксований час кодування файлу який передано їй для кодування [42].

Через те що наша програма має декілька складових ми її оцінюємо конкретно відносно тільки процесу кодування, та декодування. Якщо б ми її оцінювали на різних пристроях то мали б різні результати. Це зумовлене планувальниками задач в системі де тестується програма, завантаженістю процесора, кількості ядер, та потоків, і іншим параметрам які залежать як від програмної реалізації так і від апаратної частини.

Ключовими елементами які на перший погляд можуть не датись до уваги є створення екземплярів класу. Так як загально екземпляри класу створюються в кожній формі то мають свій глобальний характер, але деякі створюються в самих функціях при використанні. Наприклад клас кодування і декодування створюється там де його будуть використовувати. Інші класи і форми створюються на початку або в інших компонентах. Кожна з форм створюється на початку, наприклад форма запису і відтворення та запису з кодуванням, форма кодування і декодування. Форма введення ключа створюється також на початку але виконує лише одну функцію. Також щоб досягати асинхронності при керуванні програмою, деякі процеси запускаються в окремих потоках. Ці окремі потоки не є повноцінними потоками так як вони підкоряються GIL [43].

Основна задача кодування і декодування була досягнута шляхом створення відповідних функцій, та створенням необхідної обв’язки. Шляхом дослідження необхідних компонентів ми визначили що необхідна саме така реалізація. Кожен з компонентів виконує свою задачу, а саме головний скрипт запускає задачу. Після створюється головне вікно яке в свою чергу будує функції його компонентів, щоб викликати необхідні алгоритми виконання через виклик потрібних компонентів. Інші форми створюються відразу як екземпляри інших класів, які в свою чергу наслідують візуальний інтерфейс необхідних компонентів, а саме кожен свою форму.

Підвищення швидкості потокового шифрування з використанням багатопотокової обробки в реальному часі виконано завдяки використанню мультипроцесорності, яка є більш загальним аналогом багатопотоковості, але має такий самий ключовий інтерфейс керування, тільки відмінність в пулі задач які можна задавати за допомогою спеціального інтерфейсу. В нашій задачі він не використовується за непотрібністю. Ми використовуємо стандартний цикл створення потоків які насправді називаються процесами але з деяким спрощенням виконують реальну адекванту функцію потоків. Подібна багатопотоковість в C++ реалізується дуже просто і потоки мають доступ до тієї самої області памяті з якої вийшли. В Python така сама робота неможлива через порушення принципу порядку виконання коду, і за допомогою подібного можна порушити правильність виконання коду [44]. Ця проблема виникає через те що доступ до змінних не обмежений і таким чином при реалізації реальних потоків, кожен зміг би міняти змінні, і в той самий час інший поток міг би змінити змінну, а інший читав би її, і це призвело б до помилки. Щоб не допустити таких помилок, головний інтерпретатор міг би блокувати змінні, але так як Python, започатковано ще тоді коли про багатопотоковість не говорили, то і відповідно було закладено однопоточність, як головний варіант обробки даних.

В програмі використовується як мультипроцесорність так і багатопоточність (яка так називається в Python). Перший варіант використовується для кодування і декодування, а наступний для відтворення і запису. При запуску відтворення або запису відбувається створення екземпляру класу потоку, який відповідно створює поток, та йому передаємо параметри в яких вказуємо потрібні деталі, а також функцію яка буде виконуватись в потокові умовному [45]. Далі іде запуск, а програма продовжує працювати. Всі ці операції, дозволяють створити запущену потокову функцію окремо від основного ходу програми. Так як ми для того щоб запустити, натискаємо кнопку то кнопка не блокується, що дозволяє працювати з програмою в подальшому.

Мультипроцесорність дає можливість працювати функціям і програмі окремо від основної програми, що дозволяє запускати їх з керуванням. Керуємо ми за

допомогою кнопок які мають бути завжди доступні, щоб ми могли змінити стан програми в будь, який момент, але для кодування таке не варто робити тому що процес може просто почати висіти це не правильно.

Завдяки такій реалізації програми ми досягаємо потокового шифрування в реальному часі. Під час запису ми отримуємо дані а далі ми їх шифруємо і зберігаємо в файл. Все відбувається в реальному часі, нам після того не потрібно шукати збережений записаний файл, вводити його як вхідний, вказувати папку збереження, і шифрувати, так як всі ці операції вже давно відбулись під час запису. Після запису ми маємо готовий зашифрований файл, який на потрібно просто розшифрувати з метою отримання даних із нього для себе щоб отримати інформацію яка може бути корисна нам, але ми не можемо отримати її коли дані є зашифрованими, тому ми робимо саме так [46]. Ключ має збігатись із ними щоб правильно розшифрувати повідомлення. Без розшифрування файл неможливо буде відкрити програмами які на цьому спеціалізуються.

3.4 Висновки

В даному розділі було вдосконалено метод потокового шифрування аудіо інформації з використанням багатопотокової обробки в реальному часі. Створено інструкцію користувача роботи з програмою, де поетапно пояснений весь функціонал розробленої програми. Наведено опис розробки. Порівняно швидкість кодування та декодування програм. Представлено та узагальнено результати роботи.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Визначення комерційного потенціалу розробки

Метою проведення комерційного і технологічного аудиту є підвищення швидкодії потокового шифрування аудіо інформації з використанням багато потокової обробки в реальному часі, науковий керівник – Карпінець В.В. В результаті оцінювання можна буде зробити висновок щодо напрямів організації подальшого впровадження програми з врахуванням встановленого рейтингу.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів: Поплавський А.В., Азарова А.О., Катаєв В.С. Здійснюємо оцінювання комерційного потенціалу розробки за 12-ю критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорів-нює цінам аналогів	Ціна продукту дещо нижча за ціни аналогів	Ціна продукту значно нижча за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та спо-живчі властивості продукту на рівні аналогів	Технічні та споживчі влас-тивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 4.1

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною ди-намікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конку-ренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні дося-жні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 4.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більший 5-ти років	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій менший 3-х років
12	Необхідна розробка регламе-нтних документів та отриман-ня великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що потребує значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту потребує незначних коштів та часу	Необхідно тільки повідомлен-ня відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки наведено в таблиці 4.2.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки

Критерії	Прозвище, ініціали, посада експерта		
	Поплавський А.В	Азарова А.О	Катаєв В.С
	Бали		
1. Технічна здійсненність концепції	3	3	3
2. Ринкові переваги (наявність аналогів)	1	2	1
3. Ринкові переваги (ціна продукту)	3	3	4
4. Ринкові переваги (технічні властивості)	4	4	4
5. Ринкові переваги (експлуатаційні витрати)	4	4	3

Продовження таблиці 4.2

6. Ринкові перспективи (розмір ринку)	4	4	3
7. Ринкові перспективи (конкуренція)	2	3	2
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	4	4	4
10. Практична здійсненність (необхідність нових матеріа-лів)	4	4	4
11. Практична здійсненність (термін реалізації)	3	3	4
12. Практична здійсненність (розробка документів)	3	2	2
Сума балів	$СБ_1$=38	$СБ_2$=39	$СБ_3$=37
Середньоарифметична сума балів СБс	38		

За даними таблиці 4.2 робимо висновок щодо рівня комерційного потенціалу розробки. При цьому користуємося рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вищий середнього
21…30	Середній
11…20	Нижчий середнього
0…10	Низький

Отже. Виходячи з даних таблиці 4.1 та враховуючи дані таблиці 4.2 можна зробити висновок, що нова розробка має рівень комерційного потенціалу вище середнього.

З метою мінімізації витрат на реалізацію програмного забезпечення обираємо прямий канал збуту “виробник-споживач”, що забезпечить економічну ефективність операцій продажу інновації за рахунок уникнення залучення торгівельних посередників. На початковому етапі збуту розповсюдження розробки можна здійснити безпосередньо завдяки розміщенню рекламних оголошень в інтернеті.

4.2 Розрахунок витрат на здійснення науково-дослідної роботи

Кожен розробник програми автоматично стає її виробником, що на свій ризик, вкладаючи власні кошти у виробництво здійснює виготовлення нового технічного рішення для реалізації його споживачам.

Тому буде складено кошторис витрат потрібних для розробки та виготовлення програмного засобу підвищення швидкодії потокового шифрування аудіо інформації з використанням багатопотокової обробки в реальному часі.

Витрати на основну заробітну плату визначаються за формулою 4.1:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.1)$$

де k – кількість посад дослідників, залучених до процесу досліджень;

$M_{пі}$ – місячний посадовий оклад конкретного дослідника, грн;

t_i – кількість днів роботи конкретного дослідника, дн.;

$T_р$ – середня кількість робочих днів в місяці, $T_р$=22 дні.

Розрахунки заробітних плат для керівника та інженера наведені в таблиці 4.4.

Таблиця 4.4 – Розрахунок основної заробітної плати

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
Керівник проекту(інженер-програміст)	7000	318	120	38181
Всього				38181

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою 4.2:

$$З_р = \sum_{i=1}^{n} C_1 \cdot t_i, \quad (4.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за вико-нану відповідну роботу, грн/год;

t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою 4.3:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (4.3)$$

де $М_М$ – розмір мінімальної місячної заробітної плати грн, у грудні 2021 року становила 6500 грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мініма-льної заробітної плати.

$Т_р$ – середня кількість робочих днів в місяці, $Т_р$ = 22 дні;

$t_{зм}$ – тривалість зміни, год.

Погодинна ставка робітника C_i=81 грн/год.

Витрати на основну заробітну плату робітників $З_р$ =486

Таблиця 4.5 – Витрати на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Тестування	6	1	1.0	96	486
Всього					486

Розраховуємо додаткову заробітну плату розробників та робітників.

Додаткова заробітна плата розраховується як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати.

Додаткова заробітна плата $З_{дод}$=3866 грн.

Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$Н_{зп} = (З_о + З_р) \cdot \frac{\beta}{100}, \quad (4.5)$$

$$Н_{зп}=(38181+486) \cdot 0.22=8506 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для основних фондів, що використовуються в процесі розробки програмного забезпечення виконується за такою формулою:

$$А_{обл} = \frac{Ц}{Т_{кор}} \cdot \frac{Т_{фак}}{12}, \quad (4.6)$$

де Ц – балансова вартість обладнання, приміщень, тощо, які використовувались для розробки нового технічного рішення без врахування ПДВ, 69т.69.

$Т_{фак}$ – термін фактичного використання обладнання, приміщень під час розробки, місяців

$Т_{кор}$ – термін використання обладнання, приміщень згідно діючого ПКУ, років.

Таблиця 4.6 – Розрахунок амортизаційних відрахувань матеріальних і нематеріальних ресурсів

Найменування програмного забезпечення	Балансова вартість, 70т.70.	Термін корисного використання, роки	Термін фактичного використання, міс.	Величина амортизаційних відрахувань, 70т.70.
Системний блок	10800	2	4	1782
Монітор	5000	2	4	825
Роутер	500	2	4	82,5
Приміщення	200000	20	4	3300
УПС	3000	2	4	495
Принтер	2000	2	4	33
Всього				6517,5

Таблиця 4.7 – Витрати на комплектуючі, що були використані для розробки

Найменування матеріалу	Одиниці виміру	Ціна, 70т.70.	Витрачено, 70т..	Вартість, 70т.70.
Провайдер	70т..	250	5	1250
Папір	уп..	150	1	150
Фарба	70т..	75	5	375
Катриджі	70т..	400	1	400
Всього				2175

Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (4.7)$$

де В – вартість 1кВТ-години електроенергії(В=1,68 грн/кВт)

П – установлена потужність обладнання (П=0.9кВт)

Ф – фактична кількість годин роботи обладнання (Ф=192 год)

$К_п$ – коефіцієнт використання потужності ($К_п$<1, $К_п$=0.7)

$$В_е=1,68×0.9×192×0.7=203,21(грн.)$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників, які виконували дану роботу, тобто:

$$В_{ін} = (1 \dots 3) \cdot (З_о + З_р), \quad (4.8)$$

$$В_{ін}=1×38667=38667(грн).$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_р + Н_{зп} + А + ДМ + В_е + І_в$$

$$В=38181+486+8506+3866+6517,5+2175+203,21+38667=97601,7(грн.)$$

Розрахуємо загальну вартість наукової роботи $В_{заг}$ за формулою:

$$В_{заг} = \frac{в}{\alpha}\ , \quad (4.9)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях=0.8.

$$В_{заг}= 123252 грн.$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконанох наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}, \quad (4.10)$$

де β – коефіцієнт, який характеризує етап виконання даної роботи. Оскільки розробка знаходиться на стадії впровадження, то β=0.9.

ЗВ=136946 грн.

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації розробленого програмного забезпечення.Зростання чистого прибутку можна оцінити у теперішній віртості грошей.

Оцінка чистого прибутку розробників від реалізації інноваційного програмного забезпечення $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів, розраховується за формулою:

$$\Delta П_i = Ц_р \cdot N \cdot \lambda \cdot p \cdot \left(\frac{1-v}{100}\right), \quad (4.11)$$

де $Ц_р$ – ціна реалізації нової розробки

N – кількість реалізованих екземплярів програми

λ – коефіцієнт, який враховує сплату податку на додану вартість. У 2021 році податку на додану вартість 20%, а коефіцієнт λ=0.8333.

р – коефіцієнт, який враховує рентабельність продукту, рекомендується приймати 0.15…0.3

v – ставка податку на прибуток, 18%.

Припустимо, що при прогнозованій ціні 500 грн. за одиницю, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти ціну на 300 грн. Кількість одиниць реалізованох продукції також збільшиться: протягом першого року – 1500 шт., протягому другого року – 2500 шт., протягом третього року – 3000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

протягом першого року:

$\Delta П_1$ =(0×300+(500+300)×1500)×0.8333×0.3×0.82=245990 грн.

протягом другого року:

$\Delta П_2$ =(0×300+(1000+300)×(1500+2500))×0.8333×0.3×0.82=1065957 грн.

протягом третього року:

$\Delta П_3$ =(0×300+(1000+300)×(1500+2500+3000))×0.8333×0.3×0.82=1865425 грн.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 3177372 грн.

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Визначимо відносну і абсолютну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$ вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \quad (4.12)$$

де ПП – приведена вартість усіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової роботи, грн..

PV – теперішня вартість інвестицій, PV=ЗВ=136946 грн.

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.13)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн.

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні. Для України цей показник 0.1

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$П = \frac{245990}{(1+0.1)^1} + \frac{1065957}{(1+0.1)^2} + \frac{186542}{(1+0.1)^3} = 223627+880956+140151=1244734(грн.)$$

Оскільки період часу розробки інноваційного продукту є відносно незначним, то для спрощення розрахунків ним можна знехтувати і тому життєвий цикл наукової розробки буде дорівнювати періоду отримання прибутків.

Збільшення прибутку ми отримаємо починаючи з першого року.

Весь час, що характеризує рух платежів буде мати вигляд, представлений на рисунку 4.1.

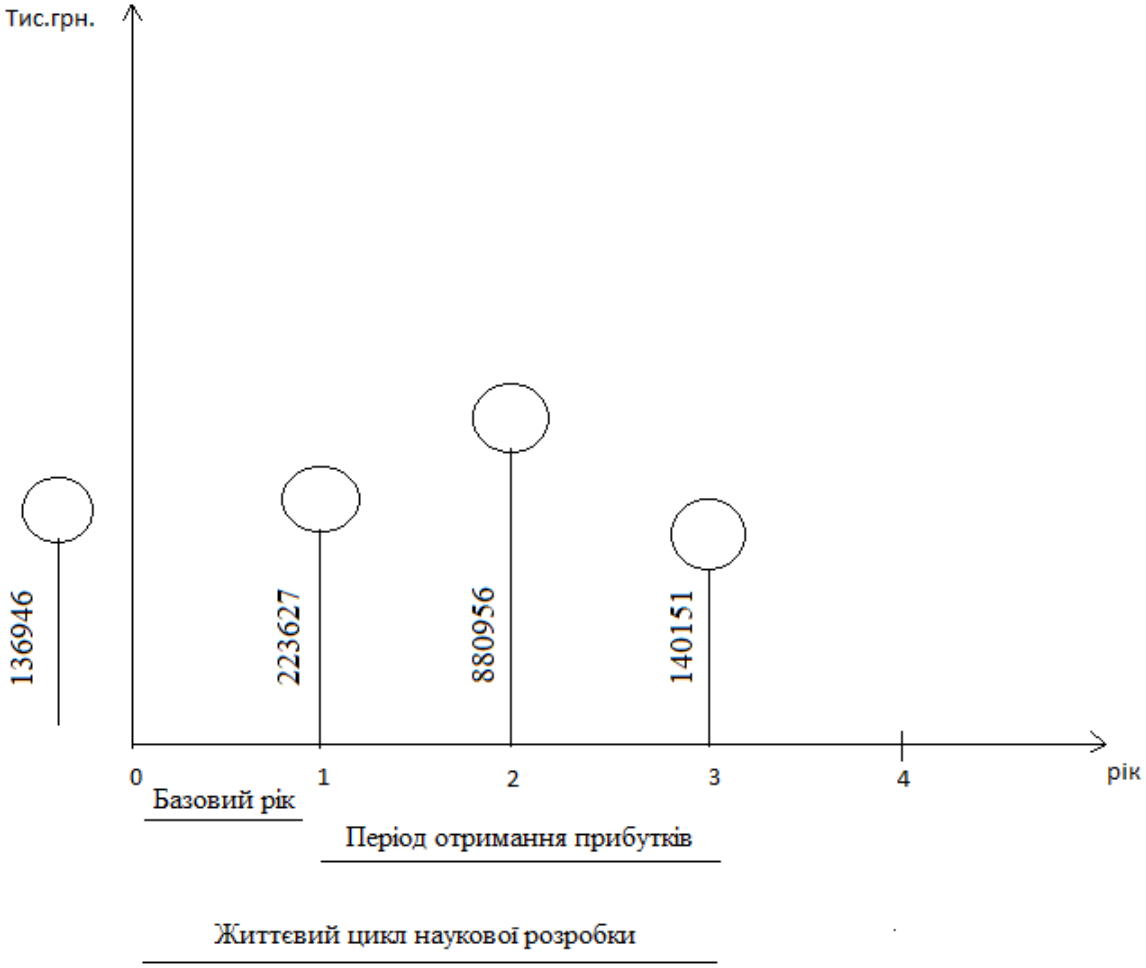

Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо $E_{абс}$:

$$E_{абс} = 1244734-136946=1107788 \text{ грн.}$$

Оскільки $E_{абс}>0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відновну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[T]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.14)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн.

PV – теперішня вартість інвестицій, PV=ЗВ=136946 грн.

Т – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + \frac{1107788}{136946}} - 1 = 1.086 \text{ або } 108\%$$

Далі, розраховану величину $E_в$ порівнюємо з мінімальною ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна ставка дисконтування визначається за формулою:

$$\tau = d + f, \quad (4.15)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках, d=0.2

f – показник, що характеризує ризикованість вкладень, величина f=0.1.

$$\tau = 0.2 + 0.1 = 0.3$$

Оскільки $E_в$ більше $\tau_{мін}$, то інвестор буде зацікавлений вкладати гроші в дану розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}, \quad (4.16)$$

$T_{ок}$=0.9 року

Отже, термін окупності даної наукової розробки менше року, тому можна зробити висновок, що її фінансування буде доцільним.

4.5 Висновки

В даному розділі було проведене економічне обґрунтування доцільності розробки програмного засобу підвищення швидкодії потокового шифрування аудіо інформації з використанням багато потокової обробки в реальному часі.

Незалежними експертами було здійснено оцінювання комерційного потенціалу розробки, за результатами якого було визначено, що нова розробка має рівень комерційного потенціалу вище середнього.

Також було виконано прогнозування витрат на виконання розробки, де розраховано основну заробітну плату кожного із робітників, додаткову заробітну плату всіх робітників, нарахування на заробітну плату, амортизацію обладнання, витрати на допоміжні матеріали,витрати на основну електроенергію.

Визначено, що абсолютна ефективність вкладених інвестицій свідчить про те, що вкладання коштів на виконання та впровадження результатів НДДКР є доцільним.

Розраховано відносну ефективність вкладених в наукову розробку інвестицій, її величина більша за бар’єрну ставку дисконтування, отже інвестор буде зацікавлений у фінансуванні даної розробки.

Проведено розрахунок терміну окупності що складає 0.9 року, це означає, що фінансування розробки програмного засобу підвищення швидкодії потокового шифрування аудіо інформації з використанням багато потокової обробки в реальному часі є доцільним.

ВИСНОВКИ

В магістерській дипломній роботі було проаналізовано можливі методи та засоби захисту звукової інформації. Було виявлено, що найбільш ефективним при захисті звукової інформації, що передається по цифрових лініях зв'язку, є використання криптографічних методів шифрування, причому найбільші переваги мають цифрові методи. Було проаналізовано відомі алгоритми і схеми криптографічного захисту інформації і обрано для реалізації метод однократного гамування, який був дещо модифікований з метою підвищення безпеки використання.

За цим модифікованим методом було вдосконалено алгоритм роботи і програму для реалізації захисту звукової інформації. Для вдосконалення використовувалось середовище програмування Phyton. Розроблена програма дозволяє шифрувати звук з мікрофону , так і з звукових файлів WAV-формату. Тестування програми показало, що швидкодія процесу шифрування набагато відрізняється від результатів програми без розпаралелювання. Різниця часу витраченого на кодування двох програм складає 25 секунд. А різниця часу декодування складає 17 секунд.

Внаслідок проведення даного вдосконалення, та впровадження його до нового програмного продукту користувач отримує:

– збільшення швидкості шифрування. Оскільки в програмі використовується потокове шифрування, чим більше обрано потоків тим швидше шифрування

– шифрування безпосередньо з мікрофону під час запису файлу

– простоту використання програми. Завдяки наявності зручного, практично інтуїтивного, графічного інтерфейсу, при якому користувач може вводити власні початкові дані або вибирати їх з бази даних, підключеної до системи, йому необхідно лише слідувати простим рекомендаціям, що пропонуються. При цьому також зменшується кількість помилок оператора-користувача, що можуть бути здійснені на початкових етапах роботи з системою.

Виконаний проект показує що, результат був досягнутий, а швидкість роботи в парі з оптимізацією досягнула бажаного результату. Використання мови прогрмування Python, принесло свою користь яка показується в швидкості розробки

програмного забезпечення та високому рівню абстракції при використанні. Багатопотокова обробка даних дає свою користь в тому що дані обробляються паралельно, і кожен поток відповідає за свій блок даних та має власний ключ на цей блок.

Ключ який генерується динамічно для кожного блоку даних, дозволяє працювати з одинми і тими даними на різній кількості потоків, маючи можливість кодувати і декодувати дані з використання різної кількості потоків, наприклад кодуючи 1 потоком або більше і відповідно декодуючи, багатьма потоками або 1 відповідно.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Malkin T. Applied Cryptography and Network Security / T. Malkin, V. Kolesnikov. – New York, 2015.

2. Бобала Ю. Я. Інформаційна безпека / Ю. Я. Бобала, І. В. Горбатий., 2019. – 580 с.

3. Рябко Б. Я. Основы современной криптографии и стеганографии / Б. Я. Рябко, А. Н. Фионов., 2016. – 232 с.

4. Ворона В. А. Способы и средства защиты информации от утечки по техническим каналам [Електронний ресурс] / В. А. Ворона, В. О. Костенко – Режим доступу до ресурсу:https://cyberleninka.ru/article/n/sposoby-i-sredstva-zaschity-informatsii-ot-utechki-po-tehnicheskim-kanalam.

5. Гребенников В. Прослушка. Перехват информации / В. Гребенников., 2018.

6. Криптографическая защита телефонных сообщений [Електронний ресурс] – Режим доступу до ресурсу: http://www.bnti.ru/showart.asp?aid=15&lvl=04.03.07.01.

7. Тарнавський Ю. А. Технології захисту інформації / Ю. А. Тарнавський., 2018. – 162 с.

8. Глинчук Л. Я. Криптологія / Л. Я. Глинчук. – Луцьк, 2014. – 164 с.

9. Animabilis шифрование файлов [Електронний ресурс] / Режим доступу: http://www.rsfileencryption.com/rus/file_encryption.htm

10. Шифр XOR [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Шифр_XOR.

11. Антоненко О. В. Криптографічно методи перетворення інформації / О. В. Антоненко., 2015.

12. Корченко О. Г. Прикладна криптологія : системи шифрування / О. Г. Корченко, В. П. Сіденко., 2014. – 448 с.

13. Пекарський Б. Г. Основи програмування / Б. Г. Пекарський., 2018. – 364 с.

14. Шилдт Г. Java. Полное руководство,10-е издание / Г. Шилдт., 2019. – 730 с.

15. Яковенко А. В. Основи програмування. Python. / А. В. Яковенко., 2018. – 195 с.

16. Учебник. Начало работы с Visual Basic в Visual Studio [Електронний ресурс] // 2019 – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-basic/tutorial-console?view=vs-2019.

17. Трофименко О. Г. С++.Алгоритмізація та програмування / О. Г. Трофименко, О. В. Задерейко., 2019. – 447 с

18. Аутентификация. Теория и практика обеспечения безопасного доступа к информационным ресурсам. - Москва: Наука, 2015. - 552 с.

19. Бабаш, А. В. Информационная безопасность (+ CD-ROM) / А.В. Бабаш, Е.К. Баранова, Ю.Н. Мельников. - М.: КноРус, 2013. - 136 с.

20. Васильков, А. В. Безопасность и управление доступом в информационных системах / А.В. Васильков, И.А. Васильков. - М.: Форум, 2015. - 368 с.

21. Гафнер, В. В. Информационная безопасность / В.В. Гафнер. - М.: Феникс, 2014. - 336 с.

22. Гришина, Н. В. Информационная безопасность предприятия. Учебное пособие / Н.В. Гришина. - М.: Форум, 2015. - 240 с.

23. Девянин, П.Н. Анализ безопасности управления доступом и информационными потоками в компьютерных системах / П.Н. Девянин. - М.: Радио и связь, 2013. - 176 с.

24. Мельников, В. П. Информационная безопасность / В.П. Мельников, С.А. Клейменов, А.М. Петраков. - М.: Academia, 2017. - 336 с.

25. Мельников, В. П. Информационная безопасность и защита информации / В.П. Мельников, С.А. Клейменов, А.М. Петраков. - Москва: Мир, 2013. - 336 с.

26. Партыка, Т. Л. Информационная безопасность / Т.Л. Партыка, И.И. Попов. - М.: Форум, Инфра-М, 2016. - 368 с.

27. Рассел, Джесси Honeypot (информационная безопасность) / Джесси Рассел. - М.: VSD, 2013. - 686 с.

28. Степанов, Е.А. Информационная безопасность и защита информации. Учебное пособие / Е.А. Степанов, И.К. Корнеев. - М.: ИНФРА-М, 2017. - 304 с.

29. Чипига, А. Ф. Информационная безопасность автоматизированных систем / А.Ф. Чипига. - М.: Гелиос АРВ, 2013. - 336 с.

30. Шаньгин, В. Ф. Информационная безопасность компьютерных систем и сетей / В.Ф. Шаньгин. - М.: Форум, Инфра-М, 2017. - 416 с.

31. Шаньгин, Владимир Федорович Информационная безопасность и защита информации / Шаньгин Владимир Федорович. - М.: ДМК Пресс, 2017. - 249 с.

32. Ярочкин, В. Безопасность информационных систем / В. Ярочкин. - М.: Ось-89, 2016. - 320 с.

33. Ярочкин, В.И. Информационная безопасность / В.И. Ярочкин. - М.: Академический проект, 2014. - 544 с.

34. Джозеф Албахари, Бен Албахари C# 5.0. Справочник. Полное описание языка C# 5.0 in a Nutshell: The Definitive Reference. – М.: «Вильямс», 2012. – 1008 с.

35. Плахотникова М. А. Информационные технологии в менеджменте: учебник и практикум для СПО / М. А. Плахотникова, Ю. В. Вертакова. — Москва: Издательство Юрайт, 2019. — 462 с.

36. Информационные технологии обеспечения конфиденциальности и сохранности данных: учеб. пособие/ Н.В.Титовская, С.Н. Титовский; Краснояр. гос. аграр. ун-т. – Красноярск, 2018. – 188 с.

37. Власова Л. А. Защита информации/ Л. А. Власова. – Хабаровск: РИЦХГАЭП, 2007. – 84 с.

38. ANDREW TROELSEN - C# 6.0 and the .NET 4.6 Framework 7th ed. 2015 Edition - ISBN-13: 978-1484213339.

39. Божко Ю. А., Новикова Е. П. Кибертерроризм - угроза безопасности современного общества // Научно-исследовательская деятельность как фактор личностного и профессионального развития студентов. – 2018. – С. 234-238.

40. Булай Ю. Г., Булай Р. И. Профилактика и противодействие киберпреступности, а также международным киберугрозам //Академическая мысль. – 2017. – №. 1. – С. 31-35. https://cyberleninka.ru/article/n/profilaktika-i-protivodeystvie-kiberprestupnosti-a-takzhe-mezhdunarodnym-kiberugrozam

41. Карпова Д. Н. Киберпреступность: глобальная проблема и ее решение // Власть. – 2014. – №. 8. – С. 46-50. https://cyberleninka.ru/article/n/kiberprestupnost-globalnaya-problema-i-ee-reshenie

42. Филимонов С. А. Некоторые особенности борьбы с транснациональным компьютерным мошенничеством // Вопросы управления. – 2014. – №. 5 (11). – С. 236-243.https://cyberleninka.ru/article/n/nekotorye-osobennosti-borby-stransnatsionalnym-kompyuternym-moshennichestvom

43. Kennedy E., Millard C. Data security and multi-factor authentication: Analysis of requirements under EU law and in selected EU Member States // Computer Law & Security Review. – 2016. – Vol. 32. – No. 1. – P. 91-110.

44. Weber R. H., Studer E. Cybersecurity in the Internet of Things: Legal aspects // Computer Law & Security Review. – 2016. – Vol. 32. – No. 5. – P. 715-728.

45. Smith M. E. Implementing the Global Strategy where it matters most: the EU's credibility deficit and the European neighbourhood // Contemporary Security Policy. – 2016. – Vol. 37. – No 3. – P. 446-460.

46. Newmeyer K. P. Elements of national cybersecurity strategy for developing nations // National Cybersecurity Institute Journal. – 2015. – Vol. 1. – No. 3. – P. 9-19.

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції “Управління інформаційною
безпекою” кафедри МБІС
д.т.н., професор
_____Ю.Є.Яремчук
“____” __ _____ _____2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:
«Підвищення швидкодії потокового шифрування аудіо інформації з використанням багатопотокової обробки в реальному часі»
08-42.МКР.006.00.101.ТЗ

Керівник магістерської кваліфікаційної роботи

к.т.н., доц. зав. каф. МБІС Карпінець В. В.

Вінниця – 2021 р.

1. Найменування та область застосування

Програмний засіб для захисту аудіо інформації з мікрофона.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ № від 2021 р.

3. Мета та призначення розробки

3.1 Мета розробки: Підвищення швидкодії потокового шифрування аудіо інформації з мікрофона

3.2 Призначення: розроблений програмний засіб виконує захист від несанкціонованого заволодіння аудіо інформацією.

4. Джерела розробки

4.1 Бобала Ю. Я. Інформаційна безпека / Ю. Я. Бобала, І. В. Горбатий., 2019. – 580 с.

4.2. Рябко Б. Я. Основы современной криптографии и стеганографии / Б. Я. Рябко, А. Н. Фионов., 2016. – 232 с.

4.3. Царьов Р.Ю. Біометричні технології: навч. посіб. [для вищих навчальних закладів] / Р.Ю. Царьов, Т. М. Лемеха. – Одеса: ОНАЗ ім. О.С. Попова, 2016. – 140 с.: іл.

4.4. Тарнавський Ю. А. Технології захисту інформації / Ю. А. Тарнавський., 2018. – 162 с.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

- програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

- програмний засіб повинен шифрувати створену аудіо інформацію користувача або з мікрофону.

5.2 Вимоги до надійності:

- програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

- програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

процесор Intel Pentium з частотою 1 ГГц і вище;

– об’єм оперативної пам’яті – від 128 Мбайт;

– об’єм жорсткого диску – від 2 Гбайт;

– монітор з роздільною здатністю не нижче 1024x768;

– операційна система Windows XP SP2, Windows Vista, Windows Server 2003/2008.

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.1

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу до аудіо інформації записаної з мікрофона.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1	Визначення напрямку магістерської роботи, формулювання теми		
2	Аналіз предметної області обраної теми		
3	Апробація отриманих результатів		
4	Розробка алгоритму роботи		
5	Написання магістерської роботи на основі розробленої теми		
6	Розробка економічної частини		
7	Передзахист магістерської кваліфікаційної роботи		
8	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи		
9	Захист магістерської кваліфікаційної роботи		

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відзив керівника роботи;

– відзив рецензента

Технічне завдання до виконання прийняв _____Медяна І.Л.

Додаток Б. Лістинг програми

```
void __fastcall TForm2::Button1Click(TObject *Sender)
{
Edit1->Clear();
CheckBox1->Checked = false;
Form2->Close();
        }
void __fastcall TForm2::Button2Click(TObject *Sender)
{
Form1->Key = Edit1->Text;
if(Form1->kod){Form1->kod = 1;Form1->dec = 0;Form5->Show();}
if(Form1->dec){Form1->dec = 1;Form1->kod = 0;Form5->Show();}
if(Form1->kod_m)
{
Form4->Show();
}

Form2->Close();
        }
void __fastcall TForm5::Button2Click(TObject *Sender)
{

   if((Form1->InputName == "")||(Form1->OutputName == ""))
            {
            Form5->Close();
void __fastcall TForm4::Button1Click(TObject *Sender)
{
String dir_k;

t_directori = Form1->OutputName;
cap_dir = Form1->OutputName;

if(t_directori == "")
{
            if(SelectDirectory(",иберґть папку","",t_directori))
            {
                        cap_dir = t_directori;

                        string strk (t_directori.begin(), t_directori.end());

                        strk = strk + "/record_sound.wav";

      std::string st = N_Name(strk);
                        strk.clear();
                        strk = st;

                        WAV_XOR file(1);

                        file.CreateWavFile(strk);

                        t_directori = strk.c_str();
strk.c_str()
                        dir_k = strk.c_str();

                        Form1->InputName = dir_k;
Form1->OutputName = cap_dir;
```

```
            }
void __fastcall TForm2::CheckBox1Click(TObject *Sender)
{
if(CheckBox1->Checked)
  {Edit1->PasswordChar = 0;}
else
        {Edit1->PasswordChar = '*';}
void __fastcall TForm4::Button1Click(TObject *Sender)
{
String dir_k;

t_directori = Form1->OutputName;
cap_dir = Form1->OutputName;
MediaPlayer1->FileName = dir_k;
MediaPlayer1->Open();
 MediaPlayer1->StartRecording();
 Timer1->Enabled = true;
       }
void __fastcall TForm4::Button2Click(TObject *Sender)
{
   MediaPlayer1->Stop();
   MediaPlayer1->Save();
   Label1->Caption = "";
   time_s = 0;
   time_m = 0;
   Timer1->Enabled = false;
       }
void __fastcall TForm4::Button3Click(TObject *Sender)
{
MediaPlayer1->Play();
       }
void __fastcall TForm4::FormClose(TObject *Sender, TCloseAction &Action)
{
 t_directori = "";

 MediaPlayer1->Close();

 if(Form1->kod_m == 1){Form1->kod =1;Form5->Show();}
       }
if(Form1->kod == 1)
            {out_n = out_n + "/Sound_Coding.wav";}
            else
            {out_n = out_n + "/Sound_Decoding.wav";}
std::string st = N_Name(out_n);
            out_n.clear();
            out_n = st;

            WAV_XOR file(inp_n, out_n);
            std::string key(Form1->Key.begin(),Form1->Key.end());
            file.key_gen(key);

            ProgressBar1->Max = 99;
            ProgressBar1->Min = 0;
            Timer1->Interval = 10;
```

```
            file.heder_copi();

            Timer1->Enabled = true;

            file.data_xor_copi();

            Timer1->Enabled = false;
            a = 0;
            pos =0;
            d_s = 0;
   ProgressBar1->Position = a;
            Form5->Close();
            }
public:

    WAV_XOR()
   {
            }

            WAV_XOR(int s)
            {
     if(s){}
   }

   WAV_XOR(const std::string _in, const std::string _ou)
   {
                        if((filename_in != "1")&&(filename_ou !="1"))                        {
                        this->filename_in = _in;
                        this->filename_ou = _ou;
      this_file_size = file_size(filename_in);
                        this_heder_size = find_word(filename_in, "data");
                        this_heder_size = (this_heder_size + (4 - 1) + 4) + 1;
                        file_size_div_heder = this_file_size - this_heder_size;
this_file_wav = 1;

      int heder_size = find_word(filename_in, "data");
                        heder_size = (heder_size + (4 - 1) + 4) + 1;
      int size = file_size(filename_in);
                        d_s = size - heder_size;
      Application->ProcessMessages();
                        }
                        else if((filename_in ==" ")&&(filename_ou ==" "))
            {

                        }
                        else
                        {
int BBS(int x_,int sdvig,int shag)
   {
      int nmr = 0;
      if (x_ == 0)
      {
         x_ = x0;
                                    if (sdvig == 0) { sdvig = 1; }
         for (size_t i = 0; i < sdvig; i++)
         {
            x_ = x_ * x_ % M_;
            nmr = x_;
```

```
        }
      }
      else
                    {   if(shag == 0){shag = 1;}

        for (size_t i = 0; i < shag; i++)
        {
          x_ = x_ * x_ % M_;
          nmr = x_;
        }

      }

      return nmr;
    }

    char xor_f(char a)
    {
      char b = '0';
      int number = 0;


                    if (this_x == 0)
                    {
                          number = BBS(0,sdvig,shag);
                          this_x = number;
                    }
                    else
                    {
                    number = BBS(this_x, sdvig, shag);
                    this_x = number;
                    }


      b = number;
                    b = a ^ b;
      return b;
    }
    void key_gen(std::string &key)
    {
      KEY = key;
                    int size_key = key.size();
                    int sdv = 0;
                    for (size_t i = 0; i < size_key; i++)
                    {
      sdv = sdv + key[i];
                    }
                    sdv = sdv / size_key;

      sdvig = sdv;
      shag = size_key;
         }
void __fastcall TForm3::FormActivate(TObject *Sender)
{
 t_directori = Form1->InputName;
 Label1->Caption = t_directori;
 if(t_directori != NULL)
 {
```

```
 MediaPlayer1->FileName = t_directori;
MediaPlayer1->Open();
 }
 else
 {
  Form3->Close();
}
        }


void __fastcall TForm3::Button1Click(TObject *Sender)
{
MediaPlayer1->Play();
}

void __fastcall TForm3::Button2Click(TObject *Sender)
{
MediaPlayer1->Pause();
}

void __fastcall TForm3::Button3Click(TObject *Sender)
{
MediaPlayer1->Stop();
        }
```

Додаток В. Ілюстративний матеріал

Магістерська дипломна робота

Підвищення швидкодії потокового шифрування аудіо інформації з використанням багатопотокової обробки в реальному часі

Студент гр. УБ-20м
Медяна І.Л.

Керівник доцент
Карпінець В.В.

Актуальність

- В житті сучасного суспільства високу позицію займає інформація. Вона є базою для прийняття будь-яких рішень. Захист інформації – актуальна тема протягом довгого часу.
- Збільшення об’ємів інформації, збільшення кількості користувачів приводить до того що необхідно збільшувати, покращувати рівень якості захисту інформації

Мета і задачі дослідження

- Метою даної роботи є підвищення швидкодії та вдосконалення алгоритму для шифрування аудіофайлів і потокового шифрування звуку з мікрофона, під’єднаного до комп‘ютера.

Об'єкт та предмет дослідження

- **Об’єкт дослідження** – програмна реалізація шифрування звукової інформації.
- **Предмет дослідження** – швидкість шифрування та ступінь захисту звукових файлів у WAV-форматі та аудіосигналів безпосередньо з мікрофону.

Підвищення швидкодії потокового шифрування

- Для підвищення швидкодії потокового шифрування аудіо інформації використовуватиметься багатопотоковість.

- Багатопотоковість — властивість операційної системи або застосунку, яка полягає в тому, що процес, може складатися з кількох потоків, що виконуються паралельно, або навіть одночасно на багатопроцесорних системах.

- При виконанні деяких завдань таке розділення може досягти ефективнішого використання ресурсів комп'ютера.
- Використання декількох потоків у застосуванні означає внесення в нього паралелізму. Паралелізм – це одночасне виконання дій різними фрагментами коду застосування.
- Така одночасність може бути реалізована на одному процесорі шляхом перемикання задач, а може ґрунтуватися на паралельному виконанні коду на декількох процесорах.

Алгоритм кодування з мікрофону

- **Крок 1.** Початок роботи з програмою, запуск програми.
- **Крок 2.** Вибір функції шифрування з мікрофону.
- **Крок 3.** Якщо ключ не введено перехід до кроку 4
- **Крок 4.** Введення ключа користувачем
- **Крок 5.** Ключ записується в змінну щоб бути доступним для всієї програми.
- **Крок 6.** Вибір вхідного і вихідного файлів
- **Крок 7.** Створення нового потоку шифрування з мікрофону
- **Крок 8.** Створення екземпляру бібліотеки PyAudio
- **Крок 9.** Створення потоку аудіо з мікрофону
- **Крок 10.** Створення файлу, шифрування даних і запис в файл

Тестування роботи програмної реалізації вдосконаленого алгоритму

- Ключові методи дослідження роботи програмного продукту це вимірювання швидкості роботи програми в однопотоковому режимі, і також в багатопотоковому режимі.
- Проведено тест, який показав швидкість виконання кодування, на мелодії розміром 91.9 мегабайт.

- Ключ має довжину 3 символи. Час виконання був засічений за допомогою бібліотеки time, та склав 9.02 секунд.

```
--- 9.026423931121826 seconds ---
```

- Далі проведено тестування декодування, файл який було декодовано той самий. Час витрачений на декодування 8.5 секунд.

```
--- 8.523844718933105 seconds ---
```

- Наступні тести проведені на 4 потоках, щоб мати можливість порівняти швидкість виконання. Файли для тесту такі самі.
- Швидкість кодування становить 6.6 секунд.

```
--- 6.607980966567993 seconds ---
```

- Швидкість декодування 6.4 секунди.

```
--- 6.49871039390564 seconds ---
```

- Далі проведено тестування програми без розпаралелювання.
- В програмі без розпаралелювання неможливо було обрати потоки для кодування та декодування, тому витрачений час набагато більший за вдосконалениу версію алгоритму.

- Час кодування становить 31.9 секунд.

```
--- 31.91883373260498 seconds ---
```

- Час декодування 23.2 секунди.

```
--- 23.23665976524353 seconds ---
```

- Як бачимо швидкість кодування двох програм значно відрізняється. Різниця часу витраченого на кодування двох програм складає 25 секунд. А різниця часу декодування складає 17 секунд.

Дякую за увагу!

Додаток Г. Протокол перевірки МКР на антиплагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Підвищення швидкодії потокового шифрування аудіо інформації з використанням багатопотокової обробки в реальному часі

Тип роботи: Магістерська кваліфікаційна робота

Підрозділ: Факультет МІБ, кафедра менеджменту та безпеки інформаційних систем, гр. УБ-20м

Науковий керівник Карпінець В.В., зав.каф. МБІС, доцент, к.т.н.

Показники звіту подібності

Plagiat.pl (StrikePlagiarism)		Unicheck	
КП1			
КП2		Оригінальність	86 %
Тривога/Білі знаки	/	Схожість	14 %

Аналіз звіту подібності (відмінити подібне)

- ☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений(-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Медяна І.Л.

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

Ступінь оригінальності роботи відповідає вимогам, що висуваються до МКР

Особа, відповідальна за перевірку_____ Коваль Н.П.

(підпис) (прізвище, ініціали)

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали)