

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ШИФРУВАННЯ
ДАНИХ ЗА УМОВ ОБМЕЖЕНОГО ПРОСТОРУ
SOFTWARE DEVELOPMENT FOR DATA ENCRYPTION IN
A LIMITED SPACE

Виконав: студент(ка) 4 курсу, групи КНТ-117
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Дроздов О.М.

(прізвище та ініціали)

Керівник Гладкова О.М.

(прізвище та ініціали)

Рецензент Морщавка С.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 20 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Дроздова Олександра Миколайовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмного забезпечення для шифрування даних за умов обмеженого простору. Software Development for Data Encryption in a Limited Space

керівник проєкту (роботи) Гладкова Ольга Миколаївна, к.т.н.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ ” 2021 року №

2. Строк подання студентом проєкту (роботи) травня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 розділ	Гладкова О.М., доцент		
Нормоконтроль	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “___” _____ 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент

_____ Дроздов О.М.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

_____ Гладкова О.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра:
117 с., 8 табл., 23 рис., 2 дод., 29 джерел.

ДВОВИМІРНИЙ ШТРИХ-КОД, ПРОГРЕСИВНИЙ
ВЕБЗАСТОСУНОК, СИНТАКСИЧНИЙ АНАЛІЗ, ШИФРУВАННЯ ДАНИХ,
WEBASSEMBLY, WEB WORKERS.

Об'єкт дослідження – шифрування даних та їх збереження у друкованому вигляді.

Предмет дослідження – комп'ютерні програми для шифрування даних та їх друку.

Мета роботи – створення програмного продукту для шифрування тексту, результатом якого є невеликі шифровані дані, які зручно використовувати для друку та введення назад для дешифрування.

Матеріали, методи та технічні засоби: структурне та імперативне програмування, синтаксичний аналіз, шифрування даних, мови програмування C та TypeScript, мова розмітки HTML, технологія WebAssembly, Web Workers, редактор VSCode, персональний комп'ютер з процесором Intel Core i5 під управлінням операційної системи Microsoft Windows 10.

Результати. Досліджено шифрування даних та способи їх збереження для друку. Розроблено прогресивний вебзастосунок, який шифрує та дешифрує дані, зберігає їх як текст або двовимірний штрих-код для друку.

Висновки. Створено вебзастосунок для шифрування тексту, результатом якого є невеликі шифровані дані, які зручно використовувати для друку, введення назад або сканування для подальшого дешифрування.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 117 pages, 23 figures, 8 tables, 2 appendixes, 29 sources.

DATA ENCRYPTION, PARSING, PROGRESSIVE WEB APPLICATION, TWO-DIMENSIONAL BARCODE, WEBASSEMBLY, WEB WORKERS.

Object of study is a data encryption and its storage for further printing.

Subject of study are computer programs for data encryption and its storage for further printing.

The purpose of the work is to create a software system for text encryption, which produces small encrypted data that is convenient to use for printing and input back for decryption.

Materials, methods, and tools: structured and imperative programming, parsing, data encryption, C and TypeScript programming languages, HTML markup language, WebAssembly, Web Workers, VSCode editor, IBM-compatible personal computer with an Intel Core i5 processor under the control of the Microsoft Windows 10 operating system.

Results. Investigated encryption of data and ways of its storage for printing. Developed a progressive web application that encrypts and decrypts data, saves it as text or a two-dimensional barcode for printing.

Conclusions. Created a web application for encrypting text, which produces small encrypted data that is convenient to use for printing, typing back or detecting 2D barcode for further decryption.

ЗМІСТ

Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Огляд ключових понять шифрування.....	11
1.2 Альтернативні способи збереження даних, двовимірні штрих-коди	13
1.3 Огляд існуючих рішень	14
1.4 Висновок за розділом 1.....	17
2 Розробка архітектури програми.....	18
2.1 Вибір мови програмування та інструментів для розробки бібліотеки ..	18
2.1.1 Сумісність мов програмування.....	18
2.1.2 Мова програмування C	19
2.1.3 Використання бібліотеки у браузері	20
2.2 Вибір мови програмування та фреймворку для вебзастосунку	22
2.2.1 Вибір мови програмування	22
2.2.2 Вибір фреймворку	23
2.3 Опис бібліотеки для роботи з файловим форматом, що містить шифровані дані	26
2.4 Опис архітектури вебзастосунку	30
2.4.1 Прогресивний вебзастосунок.....	31
2.4.2 Виконання важких обчислень у Web Workers	32
2.5 Вибір хостингу для застосунку.....	35
2.6 Висновок за розділом 2.....	36
3 Основні рішення щодо реалізації компонентів системи.....	37
3.1 Опис формату файлу, що містить шифровані дані.....	37
3.2 Опис реалізованої бібліотеки.....	41
3.3 Опис реалізованого вебзастосунку.....	48
3.4 Висновок за розділом 3.....	54
4 Експлуатація, тестування та експериментальне дослідження програми	55

4.1 Призначення програми	55
4.2 Робота користувача із застосунком.....	55
4.3 Експериментальне дослідження програми.....	59
Висновки	61
Перелік джерел посилання	62
Додаток А Текст програми.....	65
Додаток Б Слайди презентації	108

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

2D	– двовимірний;
ABI	– Application Binary Interface;
AES	– Advanced Encryption Standard;
API	– Application Programming Interface;
CSS	– Cascading Style Sheets;
DOM	– Document Object Model;
HTML	– HyperText Markup Language;
HTTPS	– Hypertext Transfer Protocol Secure;
RPC	– Remote Procedure Call;
URL	– Uniform Resource Locator.

ВСТУП

Основну частину свого часу в мережі люди проводять у соціальних мережах, форумах, онлайн іграх, користуючись хмарними сервісами збереження та обробки даних. Для підключення використовують Wi-Fi мережу, можливо навіть VPN до свого робочого місця. Усі перераховані ситуації вимагають автентифікації, яка має найпоширеніший спосіб у вигляді заповнення пари пошта-пароль.

Так як від встановленого паролю залежить надійність облікового запису, виникає питання, які характеристики він має мати. Національний інститут стандартів і технологій США рекомендує мінімальну довжину пароля у 8 символів, рекомендується створення паролю генератором випадкових чисел, не використовувати підказку до паролю, перевіряти пароль на наявність у словниках [1]. Бажано не використовувати пароль більше одного разу, хоча б для найбільш критичних сервісів [2].

Звісно, що люди не в змозі запам'ятати таку кількість паролів, тому наприклад браузері пропонують запам'ятати облікові дані, а їх останні версії також генерують унікальний пароль відповідно до наведених рекомендацій. І хоча такий спосіб збереження є зручним, більшість сучасних браузерів не дозволяють додати довільний запис до бази збережених паролів. Не варто забувати й про іншу інформацію, яку користувачі мають бажання зберегти. Зазвичай способом збереження цих даних стає звичайний папір, текстовий файл, або файл електронної таблиці, які за відсутності відповідного захисту є дуже вразливими.

Розповсюдження інформації про цифрову гігієну та правильне управління обліковими даними спонукає переглянути свій спосіб їх збереження та дізнатися, що наприклад розділи диску можна зашифрувати, таблицю можна захистити паролем або ж почати використовувати спеціалізоване програмне забезпечення – менеджери паролів. Захист в таких програмних рішеннях відбувається за допомогою криптографічних методів.

Однак, користувачі зберігають такі шифровані дані лише на цифрових носіях, втрачаючи переваги запису паролів на папері. Збереження даних на папері є не тільки додатковою копію даних, але й зручним способом їх організації. А для деяких людей є просто фактом упевненості володіння даними у фізичному вигляді.

Об'єкт дослідження – шифрування даних та їх збереження у друкованому вигляді.

Предмет дослідження – комп'ютерні програми для шифрування даних та їх друку.

Мета роботи – створення програмного продукту для шифрування тексту, результатом якого є невеликі шифровані дані, які зручно використовувати для друку та введення назад для дешифрування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд ключових понять шифрування

Криптографічні методи використовуються для перевірки автентичності даних, їх цілісності, усунення можливості відмовитися від їх авторства та насамперед для збереження конфіденційності даних [3].

Під шифруванням розуміють алгоритмічне (криптографічне) перетворення даних, яке виконується у посимвольній послідовності з метою одержання шифрованого тексту [4].

Повідомлення P (відкритий текст, plaintext), що може бути будь-якою бітовою послідовністю, будучи пропущеним через функцію шифрування E перетворюється у шифротекст C (ciphertext):

$$E(P) = C$$

Оскільки сенс шифрування, полягає в тому щоб відновити вихідний відкритий текст, повинна виконуватись наступна рівність із використанням функції дешифрування D :

$$D(E(P)) = P \quad (1.1)$$

Шифрування безпека якого залежить від зберігання криптографічного алгоритму у секреті є обмеженим, сучасна криптографія вирішує цю проблему за допомогою використання ключів [3], [5].

Виділяють два види алгоритмів:

- симетричні алгоритми;
- алгоритми з відкритим ключем.

Симетричні алгоритми використовують однаковий ключ для шифрування та дешифрування, що потребує відправнику та отримувачу мати ключ перед початком передачі повідомлень. Здебільшого застосовуються для

шифрування конфіденційних даних. За використання симетричного алгоритму із ключем K рівність 1.1 приймає наступний вигляд:

$$D_K(E_K(P)) = P$$

Симетричні алгоритми зазвичай поділяють на потокові та блочні. Потокові алгоритми оброблюють за раз один біт (або байт), блокові алгоритми працюють на блоку даних фіксованої довжини за раз. Блокові шифри можуть працювати в одному з режимів, серед яких найбільш важливі: ECB, CBC, CFB, OFB, CTR. Якщо повідомлення не можна розбити на ціле число блоків, то деякі з режимів вимагають доповнювати повідомлення, наприклад послідовністю із нульових байтів у кінці останнього блоку.

Найбільш часто згадуваними алгоритмами симетричного шифрування є AES, DES, Blowfish, Twofish та Serpent. Серед них алгоритм AES – переможець конкурсу задля заміни DES (який вже був не в змозі протистояти методу грубої сили), його встановлено як стандарт шифрування в США з 2001 року.

Алгоритми з відкритим ключем використовують один ключ для шифрування, він є відкритим, та інший для дешифрування – закритий. Здебільшого застосовуються для автентифікації та обміну ключами.

Так як алгоритм шифрування є відкритим, то безпека залежить від вибору ключа для шифрування. Використання текстового паролю як ключа без додаткової обробки відкриває небезпеку перед методом грубої сили, через те що алгоритми шифрування є швидкими у виконанні і здебільшого мають підтримку на апаратному рівні. Рішенням цієї проблеми є розтягування та зміцнення ключів шляхом використання функції формування ключа [1].

Функція формування ключа – криптографічна геш-функція, що формує один або кілька секретних ключів на основі секретного значення, такого як пароль або парольна фраза, за допомогою псевдовипадкової функції [6].

Параметрами функції формування ключа KDF є пароль P , так звана «сіль» S та коефіцієнт вартості обчислення I :

$$DK = KDF(P, S, I)$$

Параметр «сіль» є деякою випадковою байтовою послідовністю, яка не є секретом та зберігається поряд із шифрованими даними. Використовується для запобігання словникових атак, райдужних атак. Коефіцієнт вартості обчислення може наприклад визначати кількість ітерацій функції, яку основна функція формування ключа виконує, аби зробити процес достатньо повільним. Уповільнення захищає від випадків коли зловмисник таки вирішує скористуватися методом грубої сили.

1.2 Альтернативні способи збереження даних, двовимірні штрих-коди

Зберігання шифрованих даних тільки на накопичувачі може призвести до їх втрати. Збереження додаткових копій у хмарі або налаштування RAID1 дозволяє знизити ці ризики. До недоліків можна відзначити необхідність підтримання синхронізації з хмарою, необхідність підключення до мережі, для RAID1 проблемою може бути відсутність технічної можливості встановлення додаткових носіїв.

Як доповненням, але не заміною можна розглядати друк на папері. Для запису двійкового коду використовувати шістнадцяткову систему числення, що має абетку з друкованих символів. Зчитування шифротексту з паперу відбувається ручним способом, що дуже довго, або за допомогою оптичного розпізнання тексту, яке може допускати помилки, які важко відслідкувати і є додатковим посередником на шляху до програмного забезпечення для дешифрування. Двовимірні штрих-коди дозволяють зберігати велику кількість інформації та займають приблизно однаковий розмір, якщо були б записані у

шістнадцятковій абетці. Такі коди можна сканувати не лише за допомогою двовимірних сканерів, але й камерою звичайного смартфона. Найбільш поширеними у використанні є Data Matrix, який маючи малий розмір широко використовується у промисловості, QR-код – у комерційному сегменті, AztecCode – у квитках для авіа- та залізничних перевезень [7].

Двовимірні штрих-коди мають ліміт на максимально можливу довжину закодованого повідомлення, а практичні для використання розміри штрих-кодів накладають ще жорсткіші обмеження. Найбільш дієвим та швидким способом зменшення об'єму даних є використання алгоритмів стиснення даних.

1.3 Огляд існуючих рішень

Під час огляду існуючих програмних рішень було виявлено, що процес шифрування конфіденційних даних та їх збереження у альтернативні способи можливий лише при використанні різних спеціалізованих застосунків. Для першого етапу, зазвичай, використовуються менеджери паролів 1Password, LastPass, Dashlane. Дані продукти дозволяють зберігати записи різних типів: текст, посилання, контакт, дані банківської картки тощо. Користувач або взагалі не може експортувати дані на рівні одного запису, або експортований файл містить дуже багато метаданих. Другий етап – використати інший застосунок для створення двовимірного штрих-коду. До проблем можна віднести довгий пошук необхідного застосунку, адже більшість з них підтримують тільки введення тексту, а не вибір файлу із нашим експортованим записом.

Процес дешифрування виконується у зворотному порядку. Спочатку зберегти у файл те, що вдалося відсканувати камерою телефону. Потім імпортувати файл назад у менеджер паролів. Звісно, що такий ланцюг дій не дуже приємно виконувати.

У таблиці 1.1 наведено порівняння існуючих рішень для збереження конфіденційних даних. В якості експериментальних даних для вимірювання використано уявний обліковий запис «website.com; test@example.com – 11112222».

Таблиця 1.1 – Порівняння рішень для збереження конфіденційних даних

Назва додатку	Шифрування	Алгоритм стиснення	Платформи	Розмір шифрованого файлу, байт
Вихідний текстовий файл	—	—	—	39 (вихідний текст)
1Password, менеджер паролів	AES-GCM-256, PBKDF2 100000 ітерацій		Mac, iOS, Windows, Android, Web	1941
KeePass, менеджер паролів	AES-256, AES-KDF 60000 ітерацій	GZip	Windows, Linux	1550
7-zip, архівування з паролем	AES-256-CBC, SHA-256 500000 ітерацій	LZMA	Windows, Linux	218
Таблиця Excel захищена паролем	AES-256-CBC, SHA1 100000 ітерацій	Немає даних	Windows	14336
96Crypt, програма шифрування	AES-256-CTR	—	Windows	53

Усі розглянуті програми за замовчуванням використовують симетричний алгоритм шифрування AES із довжиною ключа 256 біт. Режим блокового шифрування та функція формування ключа обрані різні. Цікавим фактом є те, що 7-zip не використовує випадкову «сіль» у функції формування ключа [8], про що звичайний користувач, звісно, не має здогадки. На відміну від спеціалізованої програми шифрування 96Crypt та архіватора 7-zip, інші програми мають велику кількість метаданих доданих до файлу по відношенню до вихідного тексту, що робить використання шифрованого файлу не

практичним у двовимірних штрих-кодах. Усі програми підтримують Windows, деякі Linux, отримати доступ до своїх конфіденційних даних з мобільного пристрою зможуть тільки користувачі 1Password.

Таким чином можемо зробити висновок, що головними недоліками існуючих рішень є великий розмір файлу з шифрованими даними, нехтування рекомендацій щодо формування ключа для шифрування, відсутня можливість створення двовимірних штрих-кодів на рівні запису у менеджерах паролів, недостатня підтримка різних платформ. Жоден із знайдених застосунків не поєднує усі необхідні функції в собі.

Очікується, що застосунок буде використовуватись для зберігання невеликої кількості критично-важливих даних, для яких є бажання продублювати та зберігати їх на фізичному рівні, наприклад ламінований папір, лазерне гравірування, термодрук. 2D код може бути просканий камерою мобільного пристрою безпосередньо у застосунку. Шифровані дані можуть бути як кінцевою ціллю шифрування так і надавати доступ до інших конфіденційних даних (рис. 1.1).



Рисунок 1.1 – Очікуваний спосіб використання застосунку

1.4 Висновок за розділом 1

В даному розділі було розглянуто основні поняття шифрування, такі як симетричне шифрування та функція формування ключа, розглянуто альтернативні способи збереження даних на фізичних носіях, проведено аналіз програмних засобів, що виконують ці задачі.

Було визначено, що на об'єм необхідної пам'яті для збереження шифрованої бази даних впливає:

- режим блочного шифрування;
- алгоритм стиснення даних (або його відсутність) та обрані для нього параметри;
- довжина «солі» для функції формування ключа;
- внутрішнє улаштування формату файлу (кількість метаданих, спосіб кодування текстової інформації та внутрішніх структур у бінарний вид).

Шляхом створення багатоплатформеного додатку який використовує інші алгоритми та/або параметри можемо отримати файл, розмір якого буде менше існуючих рішень, а отже стане можливим використання двовимірних штрих-кодів малих розмірів.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

Провівши аналіз, було визначено, що для забезпечення сумісності із різними мовами програмування, швидкодії виконання алгоритмів та ефективного перевикористання коду є необхідність у виокремленні логіки обробки розроблюваного файлового формату, що містить шифровані дані в окрему бібліотеку. Розроблюваний вебзастосунок стає першим користувачем відповідної бібліотеки.

2.1 Вибір мови програмування та інструментів для розробки бібліотеки

Для розроблюваної бібліотеки було визначено, що вона повинна бути одночасно кроссплатформенною та сумісною з різними мовами програмування.

2.1.1 Сумісність мов програмування

Мовна сумісність – це здатність двох різних мов програмування взаємодіяти як частина однієї системи [9].

Необхідність мовної сумісності зумовлена тим що, різні мови програмування спеціалізуються на вирішенні різних предметних областей. Зазвичай під мовною сумісністю очікується можливість передачі даних та виклику функцій іншої мови програмування. Сумісність мов програмування може бути ускладнена:

- використанням збиральника мусора (необхідно враховувати щоб дані не були випадково зібрані збиральником мусора);
- не сумісними типами даних, складністю представлення структури в іншій мові програмування;
- використанням різних віртуальних машини;

- різницею між мовами програмування (наприклад неможливість гарантування інваріантів функційної мови при виклику процедурної мови).

Найбільш поширеними рішеннями для надання сумісності є:

- використання єдиної віртуальної машини для декількох мов програмування (наприклад мови C#, VB.NET, F# виконуються у Common Language Runtime, а Java, Scala, Kotlin у Java virtual machine);
- сумісність на рівні ABI.

Двійковий програмний інтерфейс – це інтерфейс низького рівня (машинний код) для взаємодії між двійковими програмними модулями на даній платформі. ABI може визначати тип, розмір та вирівнювання примітивних типів даних, спосіб виконання системних викликів, декорування імен функцій, підтримку винятків, спосіб погодження викликів тощо [10].

Де-факто стандартом для механізму виклику та надання інтерфейсу зовнішніх функцій є підтримка ABI сумісного з мовою C.

Майже кожна сучасна мова програмування надає сумісність для виконання викликів процедур бібліотеки, розроблених на C. У той же час лише невелика кількість мов програмування використовують однакову віртуальну машину. Таким чином було обрано мову програмування C для розробки бібліотеки.

2.1.2 Мова програмування C

Мова програмування C – процедурна, імперативна мова загального призначення.

Будучи структурованою мовою програмування, це допомагає міркувати про проблему з точки зору функціональних модулів та блоків. Колекція таких модулів створює повноцінну програму. Модульна структура полегшує налагодження, тестування та обслуговування.

Мова C легка для вивчення, стандарт ANSI C нараховує лише 32 ключових слова [11]. Мова C дуже близька до рівня асемблера, при

транспіляції можна легко відстежити перетворення рядків коду C в інструкції. Мова C є компілюванню та не містить бібліотеки часу виконання (runtime library), що дозволяє створювати програми дуже малого розміру та вкрай швидкими під час виконання. Мова C не має спеціальної структури для збереження рядків, рядок це просто масив байтів, що закінчується нульовим байтом. Створення компілятора для мови не є важким, що дозволяє знайти компілятор на будь-якій архітектурі та платформі. Стандартна бібліотека невелика, здебільшого містить алгоритми або надає абстракції над системними викликами. У порівнянні з іншими мовами не містить контейнерів через те що відсутня підтримка узагальненого програмування.

Мова C дуже мінімалістична та не має наступних можливостей, проте це не завжди є недоліком мови:

- відсутність вкладених функцій;
- відсутність підтримки області імен;
- відсутність конструкторів та деструкторів;
- відсутність перевантаження функцій;
- відсутність обробки винятків [11].

При розробці на мові C особливо важливо перевіряти коректність роботи із динамічним виділенням пам'яті, перевіряти межі масиву, а також не допускати невизначеної поведінки: ділення на нуль, арифметичні дії із переповненням над знаковими числами, спроба доступу до значення за нульовим вказівником тощо.

2.1.3 Використання бібліотеки у браузері

На відміну від мови C, що компілюється у машинний код та може бути виконана лише на даній архітектурі, вебсторінки використовують мову сценаріїв JavaScript. Однак факт того, що мова є інтерпретованою, має збірник мусору робить її не найкращим варіантом для виконання складних обчислень.

Використання технології WebAssembly надає компроміс між швидкістю виконання та підтримкою декількох архітектур у браузері.

WebAssembly (WASM) – це двійковий формат інструкцій для стекової віртуальної машини [12]. Двійкове представлення файлу дозволяє швидку передачу по мережі, може бути декодований, перевірений та скомпільований за один швидкий прохід, підтримує потокове завантаження, що дозволяє почати декодування не очікуючи завершення завантаження, підтримує паралельне декодування. Для досягнення швидкого виконання WebAssembly робить ряд припущень щодо середовища виконання та сучасних апаратних архітектур [13]: пам'ять адресується на рівні байту, підтримує невіривняний доступ до пам'яті, використовує доповняльний код для знакових чисел, порядок запису від молодшого до старшого, підтримує атомарні операції роботи із пам'яттю.

Для компіляції бібліотеки на мові C у WebAssembly використано набір інструментів Emscripten.

Emscripten дозволяє скомпілювати будь-який портативний код на мові C/C++ у WebAssembly [14]. Він використовує компілятор LLVM та надає стандартну бібліотеку сумісну із обраною мовою, яка замість системних викликів до API операційної системи виконує виклики JavaScript коду. Використання Emscripten дозволяє робити мінімум змін до вихідного коду, адже навіть роботу із файловою системою реалізовано емуляцією у пам'яті. Використання Emscripten не завершується процесом компіляції, він додатково створює необхідні JavaScript обгортки для завантаження двійкового WASM файлу на вебсторінку (рис. 2.1).

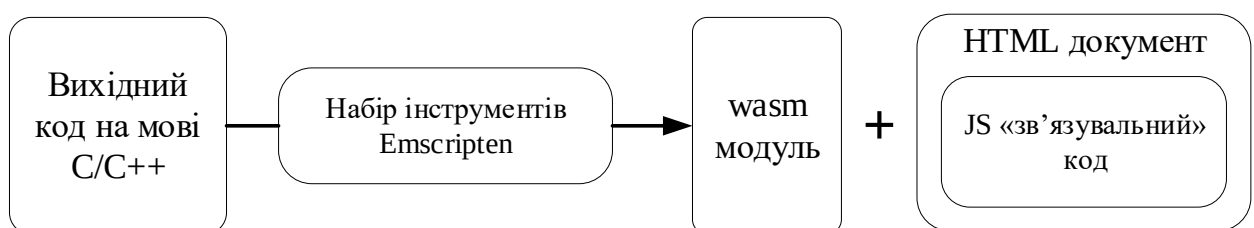


Рисунок 2.1 – Компіляція із використанням Emscripten [14]

2.2 Вибір мови програмування та фреймворку для вебзастосунку

Середовище браузеру надає стандартний набір засобів для розробки: мову гіпертекстової розмітки HTML, каскадні таблиці стилів CSS та мову створення сценаріїв JavaScript.

2.2.1 Вибір мови програмування

JavaScript – це високорівнева, мультипарадигмова мова програмування. Використання мови зосереджено навколо об'єктів та прототипів, прототипи надають механізм за яким об'єкти можуть успадковуватись аналогічно об'єктно-орієнтованим мовам. Функції є об'єктами першого класу, підтримується створення вкладених функцій, каррінг, анонімні функції, що робить мову придатною для функціонального програмування. Має динамічну типізацію, надає оператор eval для виконання сценаріїв у вигляді рядка під час виконання. JavaScript отримує оновлення у вигляді публікації стандарту ECMAScript. Остання стабільна версія була опублікована у 2020 році, має назву ECMAScript 2020 та включає такі нововведення:

- примітивний тип BigInt для створення цілих чисел будь-якого розміру;
- оператор null-об'єднання для розрізнення між хибними значеннями та null, undefined;
- необов'язкове ланцюгування, що дозволяє створювати коротші та простіші вирази для звернення через ланцюг властивостей, коли існує ймовірність зустріти відсутнє посилання;
- ключове слово globalThis для отримання глобального об'єкта незалежно від середовища виконання мови;
- нові методи Promise.allSettled, RegExp.matchAll [15].

Нововведення вже реалізовані у рушіях JavaScript: V8, SpiderMonkey, JavaScriptCore, – які використовуються в останніх версіях браузерів.

Мова JavaScript часто отримує критику, через те що дозволяє виконувати операції, які здавалося б не мають сенсу, замість створення помилок, є непослідовною та нелогічною у інших випадках, важка для підтримки великих баз коду. Це спричинило появу мов програмування із статичною типізацією, які транслюються у JavaScript та статичних аналізаторів.

TypeScript – це мова програмування розроблена і підтримувана компанією Microsoft. Мова використовує синтаксис, який є надмножиною синтаксису JavaScript і додає статичну типізацію до мови. TypeScript створювався для розробки застосунків з великою кодовою базою [16]. Усі існуючі сценарії на мові JavaScript є дійсними програмами TypeScript. Для таких JavaScript бібліотек, підтримуються файли визначень, які містять інформацію про типи, що є подібним до файлів заголовків C++. Репозиторій DefinitelyTyped містить високоякісні типи для великої кількості часто використовуваних бібліотек, що значно покращує якість коду при використанні існуючої екосистеми JavaScript бібліотек. Анотація типів змінних не є обов’язковою, окрім примітивних типів JavaScript, підтримуються перерахування, об’єднання типів, кортежі, інтерфейси. Інтерфейси особливо цікаві тим, що мова має структурну систему типів.

Для розробки вебзастосунку було обрано мову TypeScript, як таку що має систему типів, підтримку існуючих бібліотек, чудову підтримку редакторами коду.

2.2.2 Вибір фреймворку

Фреймворк – це заздалегідь створена структура програми на основі якої виконується власна розробка. Проще кажучи, це набір файлів і каталогів до яких додається власний код і файли [17]. Мета фреймворку допомогти швидше створювати застосунки, вирішуючи загальні проблеми розробки та надаючи:

- шаблонний код, який повторно використовується більшістю;

- структуру каталогів, яка дотримується деякої філософії;
- набір принципів дизайну, які зазвичай обов'язкові до виконання.

Розглянемо вебфреймворки/бібліотеки Angular, React та Vue. Кожен з них реалізує компонентний підхід до розробки та має спеціальний синтаксис для шаблонів, який нагадує HTML.

Angular – це повноцінний вебфреймворк на мові TypeScript орієнтований на розробку великих застосунків. Використання мови TypeScript є обов'язковим, тому що фреймворк використовує директиви для додавання метаданих до класів, їх методів та властивостей. Якщо компонент потребує деякий сервіс або об'єкт – використовується шаблон проєктування «впровадження залежності» [18]. При композиції компонентів дані рухаються донизу, вкладений компонент створює події для їх оновлення нагору. Такий рух даних забезпечується бібліотекою RxJS, створеної для розробки асинхронних та подійно-орієнтованих програм, використовуючи спостережувані послідовності. Angular надає чітку архітектуру та стиль коду, має генератор типових файлів для компонентів, директив, модулів, сервісів та інших частин фреймворку. Angular надає власні пакети для роботи з формами, навігацією, мережевими запитами, локалізацією, анімаціями.

React та Vue це бібліотеки, які зосереджені лише на компонентах та способу їх подання. Більш того, React навіть не залежить від середовища виконання JavaScript (для вебсередовища використовується пакет react-dom) [19]. Обидві використовують концепт віртуального DOM для оптимізації викликів до DOM API. Мовою шаблонів у React є JSX, ідея якої не створювати штучні бар'єри між технологіями завдяки розширенню мови JavaScript замість рядків HTML [20]. Vue також має офіційний плагін підтримки JSX, але за замовчуванням використовує власні шаблони схожі на Angular. Бібліотека Vue може бути використана безпосередньо в HTML документі без попереднього кроку побудови, що робить її цікавим варіантом для поступової міграції вже існуючих кодових баз, наприклад з jQuery. Оновлення внутрішнього стану компоненту у React виконується через виклик відповідної функції, оновлення

стану відбувається асинхронно та потребує уваги для недопущення помилок. Vue асинхронно оновлює компоненти, проте не стан, використання внутрішнього стану нагадує використання звичайного об'єкту JavaScript, що робить його простим та інтуїтивно зрозумілим. Перетворення декількох синхронних оновлень стану в одне асинхронне оновлення компоненту відбувається завдяки спеціальній системі реактивності, яка відслідковує зміни значень властивостей об'єкту стану під час наступної ітерації циклу подій JavaScript [21]. Однак, така система відслідковування має свої технічні обмеження, потребує уваги для недопущення помилок аналогічно бібліотеці React.

React та Vue, будучи бібліотеками, не нав'язують власну ієрархію директорій, не мають додаткових вбудованих компонентів. Для швидкого створення проєкту розробники бібліотек розробили утиліти командного рядка `create-react-app` та `vue-cli` відповідно. Під час створення проєкту для вибору доступні бажані технології (опції залежать від утиліти): мова TypeScript, PWA, CSS препроцесор, Linter, форматування вихідного коду, бібліотеку та вид тестування, бібліотеку для навігації, управління спільним станом компонентів [22]. Варто зазначити, що React, інструменти розробника та `create-react-app` підтримуються інженерами Facebook, а усі інші бібліотеки спільнотою розробників. Тим часом бібліотека Vue та більшість компонентів для неї підтримуються основною командою Vue, що робить досвід використання наближеним до Angular – повноцінного фреймворку. Хоча документація й не знаходиться в одному місці, має єдиний стиль.

Вибір інструментів необхідно робити не лише за технічними характеристиками, а й виходячи з досвіду їх користувачів. Відсоток задоволеності та готовності до повторного використання у нових проєктах фреймворків, відповідно до опитування у 2020 році [23], наведено на рис. 2.2. Як можемо побачити задоволення бібліотеками React та Vue.js можна вважати однаковим, а фреймворк Angular втрачає позиції з кожним роком.

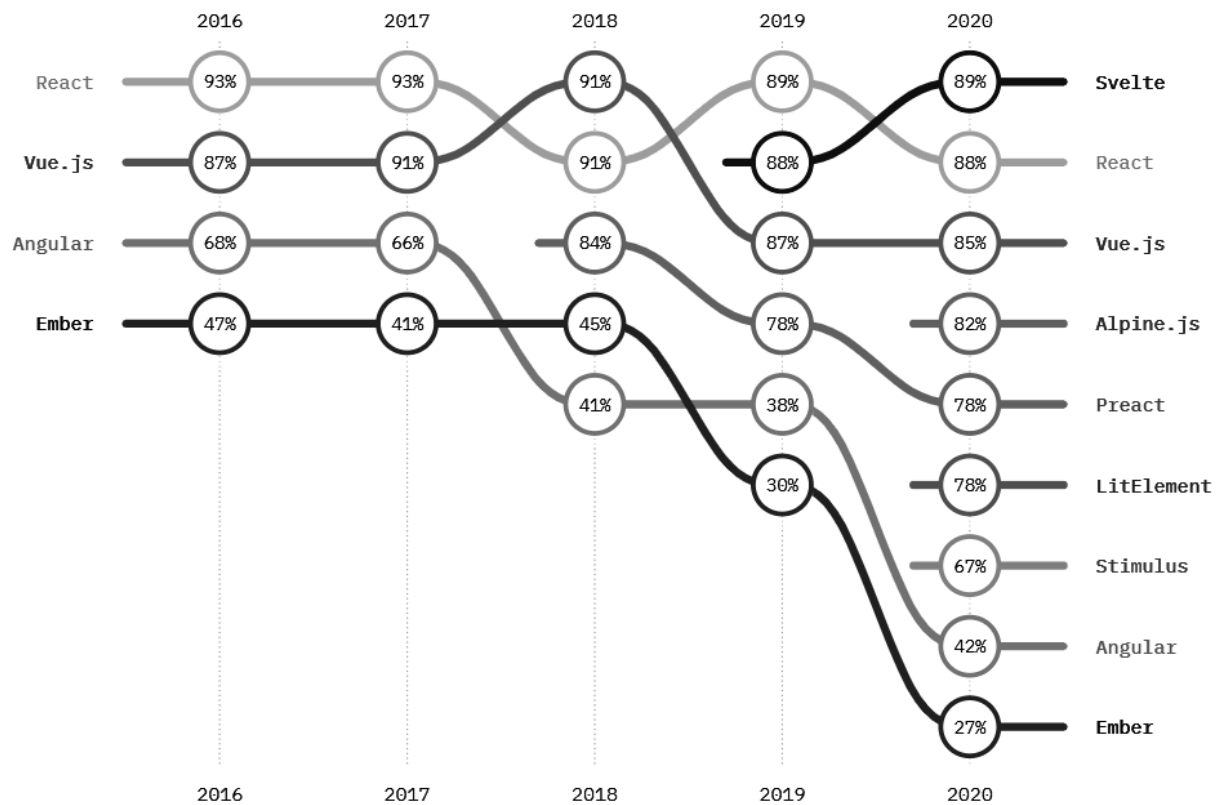


Рисунок 2.2 – Задоволеність вебфреймворками

Для розробки вебзастосунку було обрано бібліотеку Vue. До головних причин належать зручний механізм реактивності даних, продуманий синтаксис шаблонів, відсутність обов'язкової структури, стилю проекту та особистий досвід використання цієї бібліотеки.

2.3 Опис бібліотеки для роботи з файловим форматом, що містить шифровані дані

Бібліотека надає наступний програмний інтерфейс:

- зашифрувати текстовий документ;
- розшифрувати бінарні дані;
- перевірити цілісність шифрованих даних;
- отримати час шифрування із метаданих.

Бібліотека використовує проміжне представлення даних у пам'яті, що дозволяє їй трансформувати між текстовими даними та бінарними для

подальшого шифрування. Перетворення текстових даних у проміжне представлення відбувається завдяки підсистемі синтаксичного аналізу.

Типова система синтаксичного аналізу складається з:

- зчитувача, що зчитує побайтово потік даних та трансформує у символи відповідно до використаної таблиці кодування символів;
- токенизатора, який класифікує зчитані символи у можливі лексеми;
- парсеру, який аналізує лексеми відповідно до заданої граматики та формує деяку структуру даних, зазвичай – дерево [24].

Схему роботи підсистеми синтаксичного аналізу зображено на рис. 2.3.

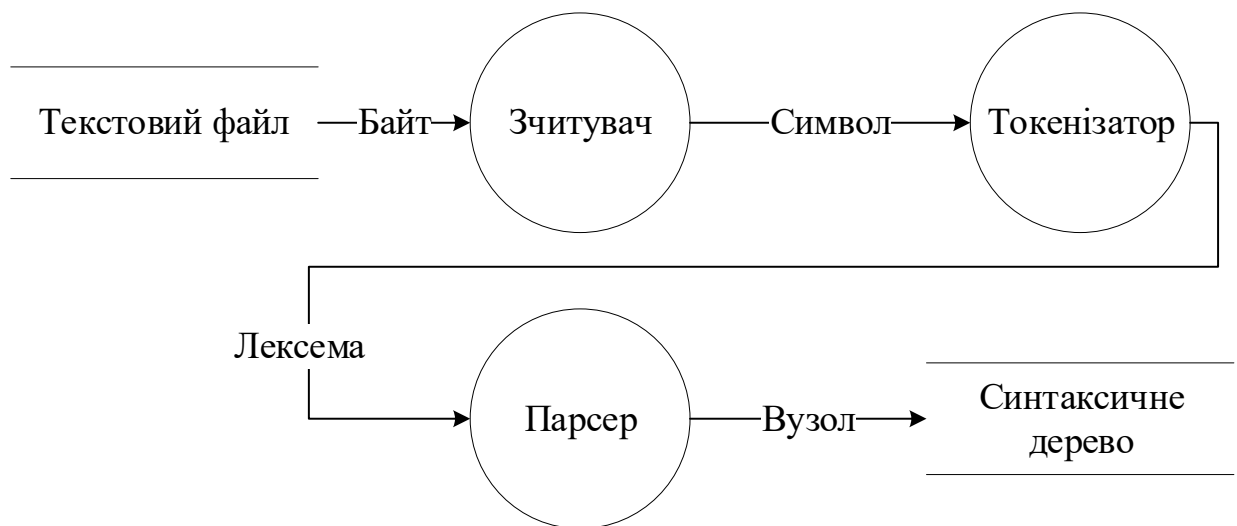


Рисунок 2.3 – Схеми роботи синтаксичного аналізу

Токенізатор бібліотеки повинен підтримувати наступні лексеми:

- ключ;
- значення;
- рядок багаторядкового значення;
- роздільник записів.

Не приймаючи до уваги обробку помилок, процес створення контейнеру із шифрованими даними можна представити як послідовність кроків:

- виконати синтаксичний аналіз вихідного документу;

- серіалізувати дерево проміжного представлення текстового документу у безперервну послідовність байтів у пам'яті;
- виконати стиснення, якщо послідовність піддається стисненню;
- сформувати сіль, ключ із парольної фрази;
- зашифрувати серіалізовану послідовність байтів ключем;
- обчислити контрольну суму зашифрованих даних;
- обгорнути шифровані дані разом із контрольною сумою та іншими метаданими відповідно до створеного формату.

Дешифрування текстового документу із створеного формату відбувається у зворотному порядку:

- перевірка формату даних, цілісності;
- зчитування збереженої солі, формування ключа;
- дешифрування;
- якщо дані були стиснуті, то відновити їх;
- десеріалізувати послідовність байтів у проміжне представлення документу;
- сформувати текстовий документ із проміжного представлення.

Кроки для шифрування та дешифрування формують послідовність виклику процедур як зображено на рис. 2.4.

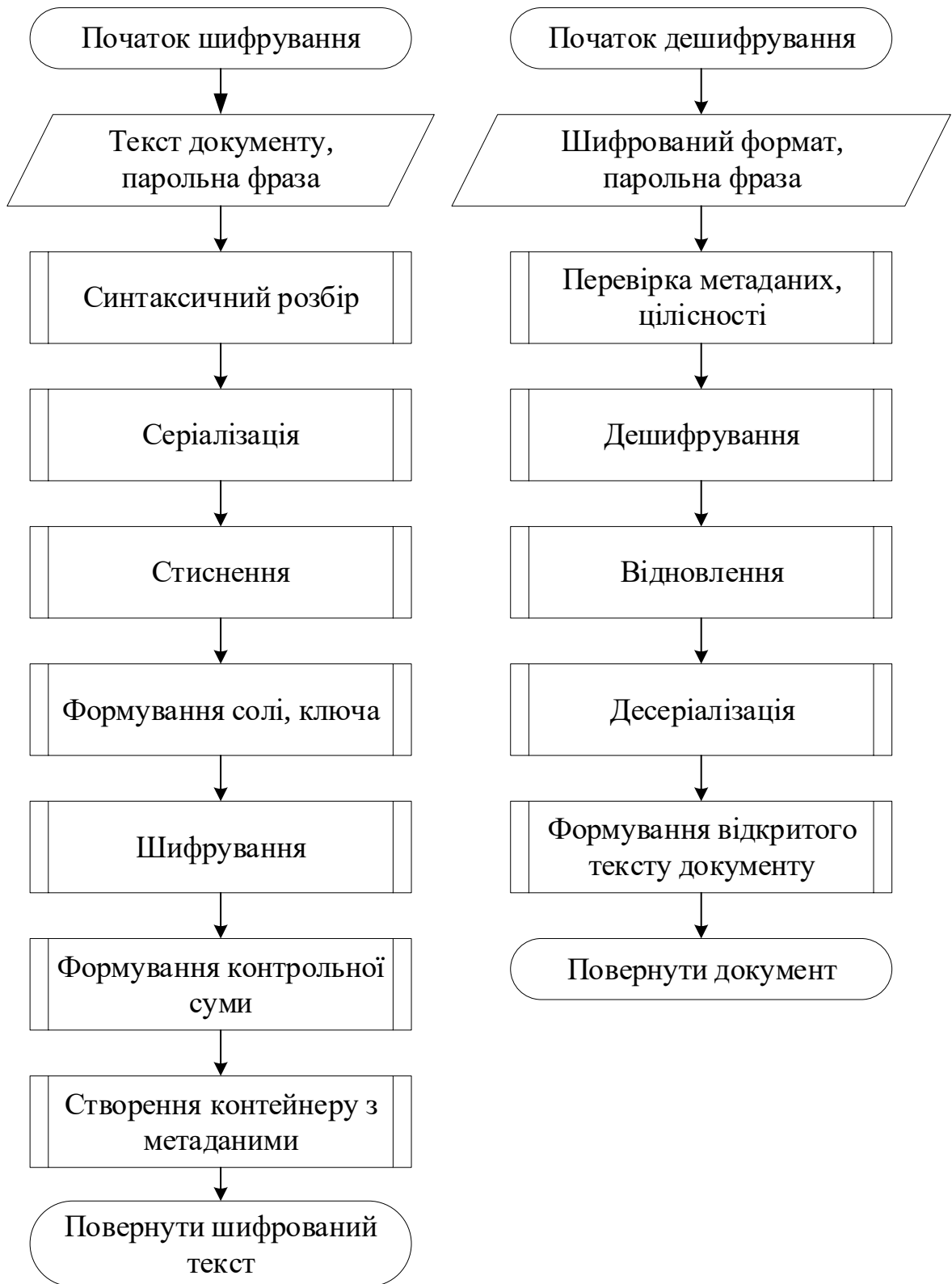


Рисунок 2.4 – Послідовність виклику процедур для шифрування та дешифрування

2.4 Опис архітектури вебзастосунку

Вебзастосунок є однією із можливих реалізацій надання зручного графічного інтерфейсу взаємодії із розробленим форматом файлу, що містить шифровані дані. Вебзастосунок додатково виконує функцію посередника, розпізнаючи та створюючи 2D-матричні коди, що містять зашифровані дані. На рис. 2.5 зображено функціональний розподіл між компонентами.

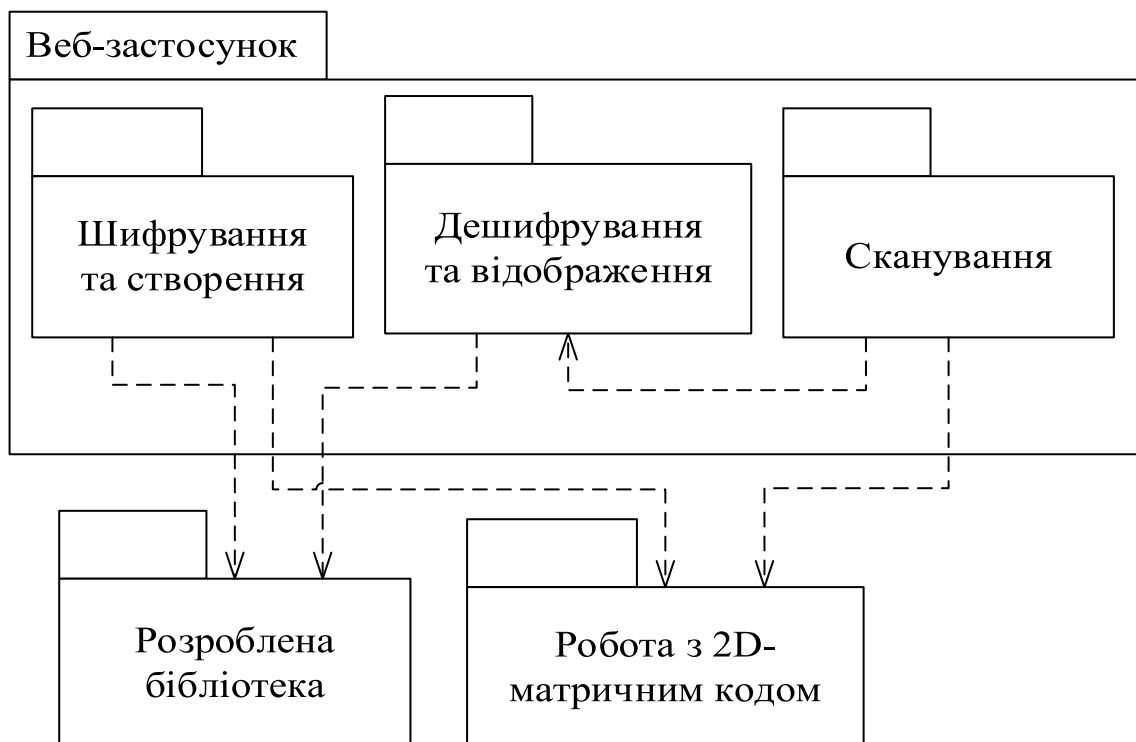


Рисунок 2.5 – Функціональний розподіл між компонентами

Користувач починає роботу з головного екрану, який має посилання для виконання:

- шифрування;
- дешифрування;
- сканування.

Компонент шифрування надає просту форму для введення текстових даних та паролльної фрази після чого шифровані дані можна зберегти в одному з виглядів:

- двійковий файл;
- транспортне текстове представлення (base16, base64);
- 2D-матричний код.

Для більш зручного введення текстового представлення при його створенні доступно розбиття суцільної послідовності на групи розділені пробілом, налаштування кількості текстових символів у групі, кількість груп на один рядок. Для 2D-матричного коду, будучи зображенням, доступне налаштування масштабування.

Компонент дешифрування надає просту форму для введення шифрованих даних у вигляді транспортного текстового представлення та паролльної фрази. У разі успішного дешифрування відображується кожен запис. Компонент дешифрування також перевіряє та відображає статус цілісності введених даних, дату їх створення. Відповідно до способів збереження шифрованих даних, компонент дешифрування підтримує імпорт двійкового файлу та має посилання на компонент для початку сканування 2D коду.

Компонент сканування 2D-матричного коду, у реальному часі, на пристрої користувача, аналізує кадри з камери та за найпершого збігу виконує перехід до компоненту дешифрування. Компонент сканування має налаштування максимально допустимого часу на розпізнання одного кадру, що може бути таким малим як частота запису камери або великим як ціла секунда для розпізнання за умов недостатньої освітленості.

Не дивлячись на використання вебтехнологій, застосунок не використовує клієнт-серверну архітектуру. Усі обчислення виконуються у браузері користувача, а отже є доцільним використання технології PWA.

2.4.1 Прогресивний вебзастосунок

Progressive Web Application (PWA) – це вебзастосунок, який використовує нові API-інтерфейси та можливості браузера разом із

традиційною стратегією прогресивного вдосконалення, щоб надати користувацький досвід близький до рідних застосунків для платформи користувача [25].

З технічної точки зору PWA-застосунк повинен:

- бути доступний лише через захищене HTTPS з'єднання, включаючи всі інші ресурси, що завантажуються;
- мати спеціальний файл маніфесту, який контролює як застосунок буде відображатися завдяки вказанню назви застосунку, початкової URL-адреси, іконок різних розмірів для різних платформ;
- мати Service Worker – сценарій на мові JavaScript, який має можливість перехоплювати та контролювати, як браузер оброблює мережеві запити та зберігання ресурсів до кешу.

Завдяки файлу маніфесту розроблений вебзастосунок може бути встановлений на домашній екран пристрою, а Service Worker дозволяє завантажити ресурси із кешу за відсутності доступу до мережі.

2.4.2 Виконання важких обчислень у Web Workers

Мова сценаріїв JavaScript, яка використовується для надання інтерактивності вебсторінкам, має асинхронну модель та так званий цикл подій. При взаємодії із вебсторінкою браузер створює відповідні події, які можуть бути оброблені у сценаріях користувача. Так як графічний інтерфейс та цикл обробки подій мають тісний взаємозв'язок та поділяють ресурси одного потоку системи, то сценарії користувача мають швидко виконуватися, щоб зберігати інтерактивний досвід користувача без затримок. У разі порушення цього правила браузер може зробити висновок, що довгий час виконання сценарію, можливо, спричинений помилкою розробника і запропонувати припинити його виконання (рис. 2.6).

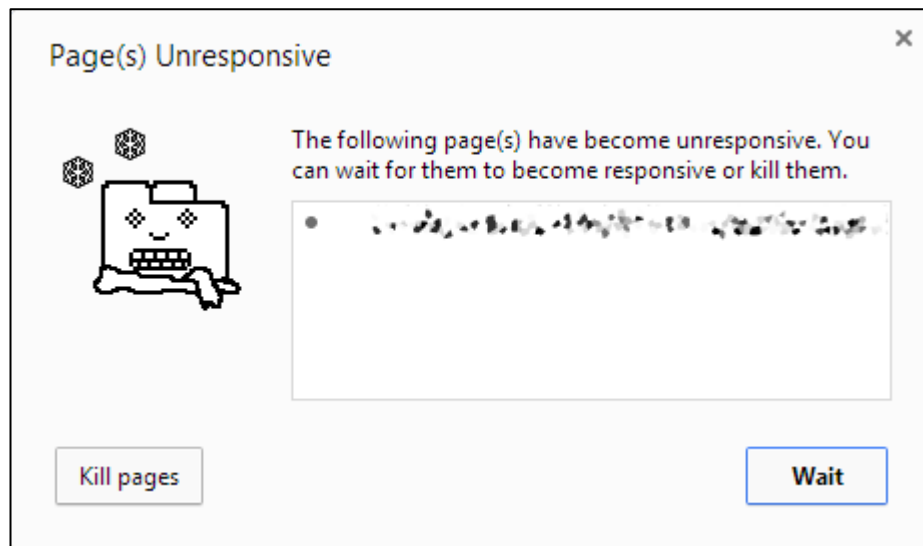


Рисунок 2.6 – Повідомлення браузеру про сторінку, що не реагує

Для виконання інтенсивних обчислень, що займають довгий час та можуть негативно впливати на інтерактивний досвід користувача використовуються Web Workers.

Web Workers API надає можливість запустити сценарій у фоновому потоці. Тобто сценарій виконується паралельно з основним потоком вебсторінки, що містить графічний інтерфейс та не блокує його важкими обчисленням [26]. Створений потік працює аналогічно основному – має цикл обробки подій. Основний потік вебсторінки та створений через Web Workers API спілкуються за допомогою подій.

Наступні операції є важкими для обчислення та повинні бути винесені у Web Worker:

- шифрування текстового документу;
- дешифрування даних;
- розпізнання 2D коду.

На рисунку 2.7 зображено обмін подіями між основним потоком та запущеним Web Worker при виконанні шифрування текстового документу. Текстовий документ відправляється у вигляді даних події у запущений потік. Бібліотека, що працює у потоці може викликати важку функцію для обчислення – функцію формування ключа. У цей же час основний потік

вебзастосунку може оновити стан кнопки на неактивний та відобразити індикатор обробки, підтримуючи інтерактивний досвід взаємодії. Через деякий час робочий потік повертає шифрований документ або помилку, що виникла.

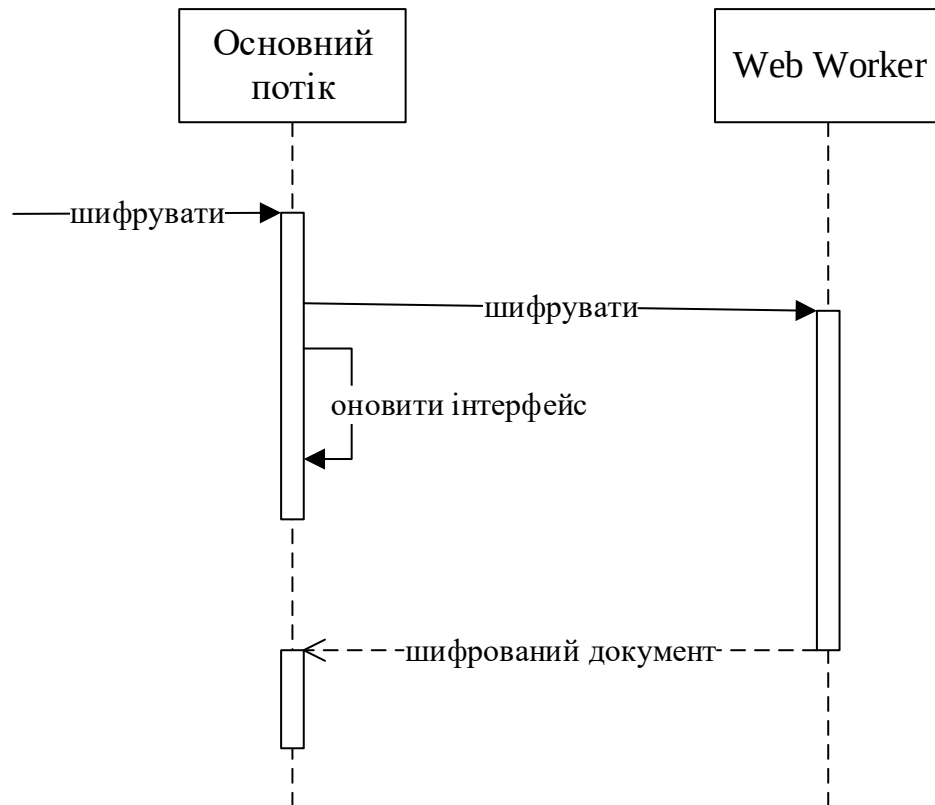


Рисунок 2.7 – Потік обміну подіями для виконання шифрування

На рисунку 2.8 зображено обмін подіями між основним потоком та запущеним Web Worker при виконанні розпізнання 2D-матричного коду. Основний потік підключається до камери телефона та відображує на сторінці відеопотік із частотою кадрів, що відповідає її технічним характеристикам. Основний потік періодично відправляє поточний кадр із відеопотоку до Web Worker де виконується розпізнання 2D коду. Очевидно, що розпізнання виконується повільніше ніж частота кадрів камери. Завдяки винесенню обчислень у додатковий потік, користувач бачить плавне відео, йому легше позиціонувати камеру на сканованому 2D коді.

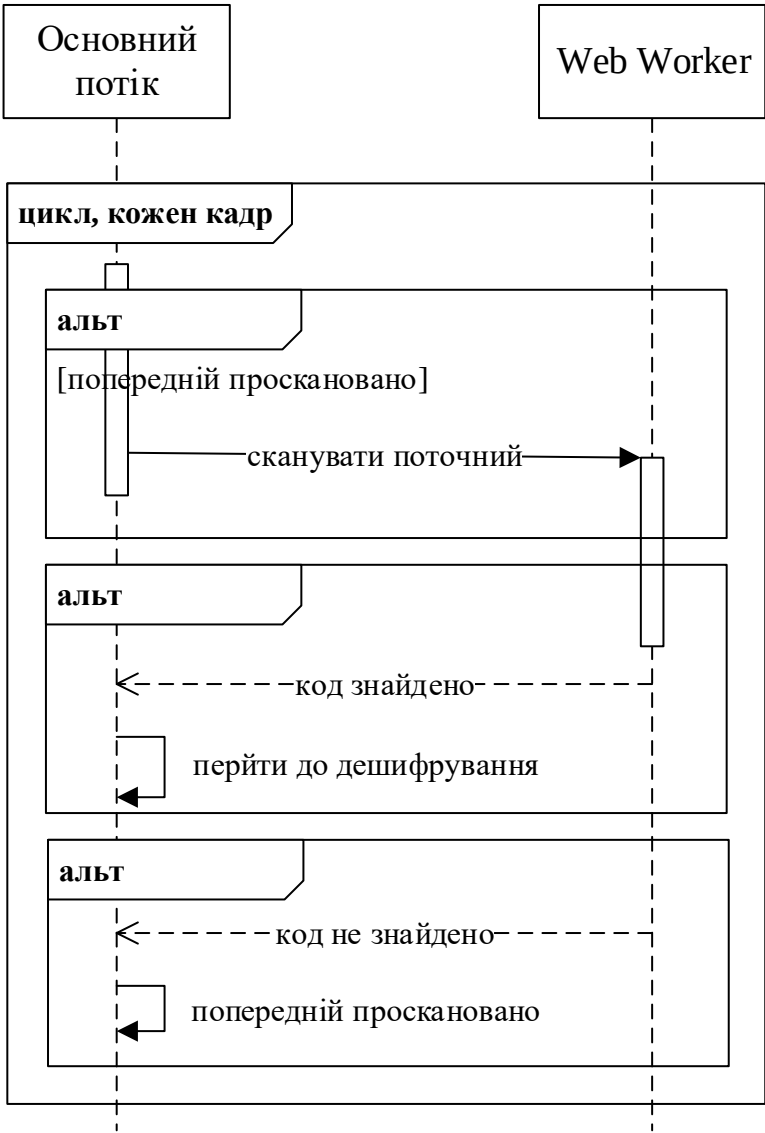


Рисунок 2.8 – Потік обміну подіями для сканування кадру

2.5 Вибір хостингу для застосунку

Не дивлячись на відсутність серверної частини та можливість роботи без інтернету, перше отримання файлів застосунку повинне виконуватися із серверу. Файли для розгортання (HTML, CSS, JS) повністю статичні і можуть зберігатись на будь-якому вебсервері. Популярними рішеннями є сервіси Netlify, Amazon S3, Google Cloud Storage, GitLab pages, GitHub pages.

Так як проєкт використовує систему керування версій Git та зберігається на сервісі GitHub, доцільно використовувати GitHub pages. GitHub

pages дозволяють в репозиторії створити гілку з ім'ям «gh-pages» та віддають будь-які файли в ній за адресою `USER_NAME.github.io/PROJECT_NAME`.

Для автоматизації процесу оновлення скомпільованих файлів до цієї гілки можна скористатися GitHub Actions. При кожній фіксації змін у репозиторії буде виконуватися YAML файл у директорії «.github/workflows», який визначає кроки для побудови проєкту та завантаження файлів до gh-pages [27].

Додатковою вимогою для віддачі сторінок застосунку є HTTPS через використання PWA. GitHub pages задовольняє цю вимогу, завдяки наданню сертифікату для всіх сайтів на домені *.github.io.

2.6 Висновок за розділом 2

В даному розділі було розглянуто шляхи підтримки мовної сумісності, розглянуто спосіб використання бібліотек написаних на мові C, C++ у браузері за допомогою технології WASM та інструменту Emscripten. Було визначено загальний алгоритм шифрування та дешифрування, визначено кроки типової системи синтаксичного розбору.

Для вебзастосунку виконано порівняння фреймворків, визначено переваги мови TypeScript по відношенню до JavaScript. Розглянуто технології PWA для постійної доступності застосунку та Web Workers для виконання важких обчислень, визначено операції які повинні в ньому виконуватись. Розглянуто та обрано GitHub pages для розгортання вебзастосунку.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Опис формату файлу, що містить шифровані дані

Збережений файл нагадує будь-які інші двійкові формати файлів, на початку зберігається невеликий заголовок, а після нього власне дані. Виходячи з проведеного аналізу, теоретично мінімальний розмір заголовку може складати 33 біт або 5 байт, з них 32 біт використовуються для збереження випадкових чисел (сіль), 1 біт – для позначення стану стиснення даних. Розроблений заголовок намагається підтримувати баланс між додатковою інформацією та її корисністю, а його загальний розмір складає 8 байт (табл. 3.1).

Таблиця 3.1 – Короткий опис властивостей у заголовку

№ з/п	Розмір, байт	Розмір, біт	Опис
1	2	16	Контрольна сума
2	3	1	Стан стиснення
3		23	Дата та час
4	3	24	Випадкові числа

Контрольна сума дозволяє переконатися у цілісності даних. В якості алгоритму обрано циклічний надлишковий код, який має ступінь многочлена 16 (CRC-16/CCITT-FALSE). Контрольна сума обчислюється останнім кроком під час створення файлу над усіма байтами окрім власне перших двох. Наявність контрольної суми дозволяє бути впевненим, що дешифрування не вдалося саме через пошкоджені дані, замість нервування та перебору усіх можливих парольних фраз користувачем.

Прапор стану стиснення займає один старший біт цілого числа, що зберігає дату та час. Такий спосіб збереження дозволяє не витрачати даремно цілий байт, але водночас не зменшує практичне максимальне значення дати та часу. Використання прапора стану стиснення є більш доцільним ніж завжди

вважати дані стиснутими, тому що у більшості випадків стиснення не можливе і результатом є дані більші за розміром, різниця розміру яких з вихідними є, звісно, більшою за 1 біт.

Дата та час займають 23 біти та зберігаються як кількість годин, які минули від початку епохи Unix. Ціле число годин не має проблеми аналогічної до проблеми 2038 року, адже обрана точність та розмір дозволяють створювати файли до 2926 року. Наявність дати та часу дозволяє:

- чітко визначити, яка версія шифрованих даних є останньою з поміж декількох конфліктних, які знайшов користувач;
- згадати, що для даної дати у минулому потрібно використовувати попередній майстер-пароль.

Випадкові числа займають 3 байти, які генеруються криптографічно стійким генератором псевдовипадкових чисел. Вони використовуються як частина солі для функції формування ключа. Другою частиною солі є байти, що використовується для збереження дати та часу. Так як години йдуть не швидко (лише 10 біт змінюються за рік), то фактично ентропія солі не складає повні 6 байтів.

Створений формат файлу не містить додаткової контрольної суми для перевірки дешифрованих даних, не містить розмір даних до стиснення. Якщо парольна фраза випадково підійшла та десеріалізація виконалась без помилок, то правильність отриманого текстового документу може бути оцінена лише користувачем. Параметри для алгоритмів шифрування та стиснення даних містяться у коді програми, якщо з часом виникне необхідність їх змінити, то застосунок не зможе дешифрувати дані створенні попередніми версіями.

Після заголовку зберігаються власне шифровані дані, які можливо додатково були стиснуті. Максимальний розмір даних до стиснення складає 64 кілобайт (практичним є використання не більше 100 байт). Дане обмеження дозволяє уникнути зір-бомб, файлу який викликає критичний стан системи, шляхом заповнення всього вільного місця тимчасової чи постійної пам'яті.

Для шифрування використовується симетричний алгоритм AES з довжиною ключа 256 біт та режимом блочного шифрування CTR. Дана довжина ключа є найбільш можливою та обрана з причин того що повідомлення малі за розміром, процес шифрування-дешифрування не виконується постійно і тому відсутня будь-яка доцільність більш малих ключів для швидкодії алгоритму. Обраний режим лічильника (CTR) має наступні цікаві для застосунку властивості:

- перетворює блочний шифр в потоковий шифр, що не вимагає доповнювати повідомлення, якщо його не можна розбити на ціле число блоків;
- використовує єдиний алгоритм як для шифрування, так і для дешифрування, що робить реалізацію меншу, ніж реалізації багатьох інших режимів.

Для формування ключа шифрування використовується функція формування ключа Argon2. Вона оптимізована для архітектури процесора x86 та навмисно використовує особливості організації кешу та пам'яті сучасних процесорів Intel та AMD, її реалізація на GPU, FPGA, ASIC не повинна бути швидшою чи дешевшою [28]. Argon2 також спроектована бути важкою для пам'яті, тобто потребує велику кількість пам'яті для виконання, а у випадку використання меншого об'єму пам'яті злоумисником відбувається зниження швидкодії алгоритму. Функція має декілька варіантів:

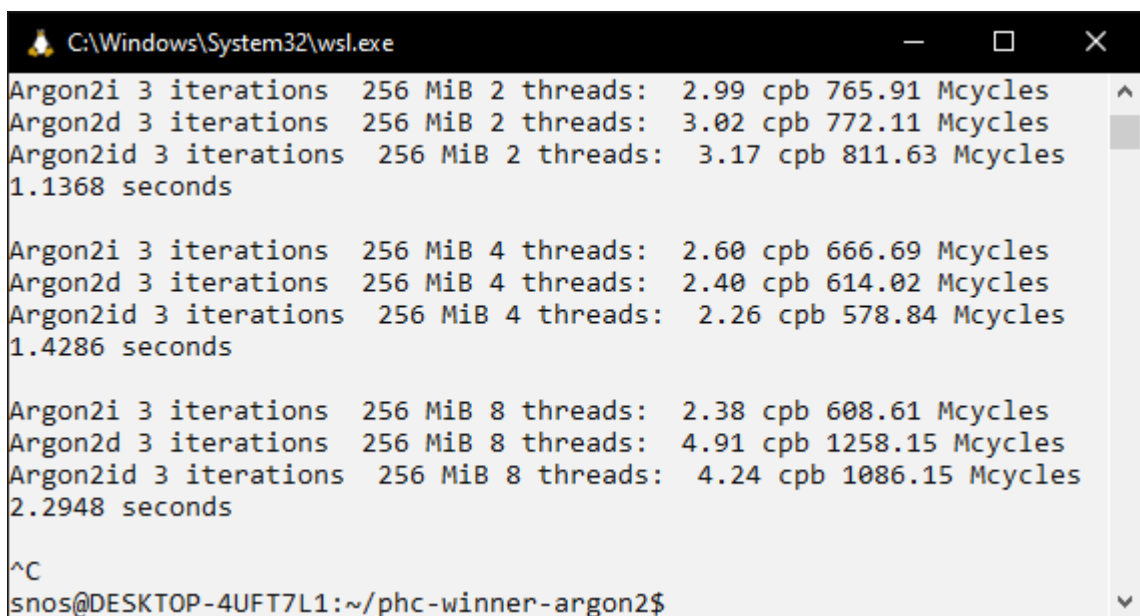
- Argon2d, яка є швидшою, підходить для захисту цифрової валюти та інформаційних систем, що не піддаються атакам по стороннім каналам;
- Argon2i, яку бажано використовувати для створення гешей паролів та формування ключа шифрування із пароллю, вона виконує більше ітерацій для захисту від атак компромісу час-пам'ять;
- Argon2id, гібридна версія, яка працює як Argon2i протягом першої половини першої ітерації над пам'яттю, а для решти як Argon2d.

Argon2 приймає параметри, які контролюють необхідний час та ресурси для виконання алгоритму:

- кількість пам'яті;

- кількість ітерацій над пам'яттю;
- ступінь паралелізму (кількість потоків обчислення).

Створений формат файлу використовує варіант Argon2id. Для визначення параметрів, за умови часу виконання до 2 секунд на найменш потужному пристрої (бюджетний смартфон, архітектура ARM, 2ГБ пам'яті), використано утиліту командного рядка із репозитарію зразкової реалізації Argon2 (рис. 3.1). Параметрами, що задовольняють стали: кількість ітерацій – 3, розмір пам'яті – 64 мегабайти, паралелізм – 2 потоки.



```

C:\Windows\System32\wsl.exe
Argon2i 3 iterations 256 MiB 2 threads: 2.99 cpb 765.91 Mcycles
Argon2d 3 iterations 256 MiB 2 threads: 3.02 cpb 772.11 Mcycles
Argon2id 3 iterations 256 MiB 2 threads: 3.17 cpb 811.63 Mcycles
1.1368 seconds

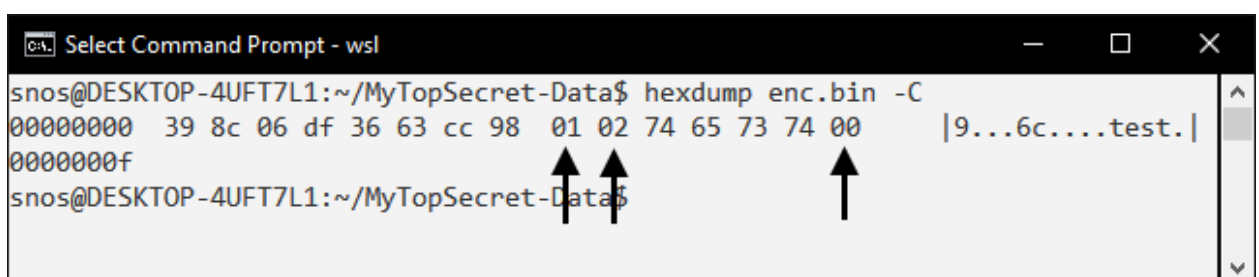
Argon2i 3 iterations 256 MiB 4 threads: 2.60 cpb 666.69 Mcycles
Argon2d 3 iterations 256 MiB 4 threads: 2.40 cpb 614.02 Mcycles
Argon2id 3 iterations 256 MiB 4 threads: 2.26 cpb 578.84 Mcycles
1.4286 seconds

Argon2i 3 iterations 256 MiB 8 threads: 2.38 cpb 608.61 Mcycles
Argon2d 3 iterations 256 MiB 8 threads: 4.91 cpb 1258.15 Mcycles
Argon2id 3 iterations 256 MiB 8 threads: 4.24 cpb 1086.15 Mcycles
2.2948 seconds

^C
snos@DESKTOP-4UFT7L1:~/phc-winner-argon2$
  
```

Рисунок 3.1 – Виконання утиліти на комп'ютері

Перед стисненням та власне шифруванням даних, спочатку вони повинні бути перетворені із проміжного ієрархічного представлення бібліотеки у безперервну послідовність байтів.



```

Select Command Prompt - wsl
snos@DESKTOP-4UFT7L1:~/MyTopSecret-Data$ hexdump enc.bin -C
00000000 39 8c 06 df 36 63 cc 98 01 02 74 65 73 74 00 |9...6c....test.|
0000000f
snos@DESKTOP-4UFT7L1:~/MyTopSecret-Data$
  
```

Рисунок 3.2 – Бінарне представлення файлу (не шифрованого)

Для кожного запису у документі виконується перетворення. Зберігається один байт, який визначає кількість пар ключ-значення в даному записі (перша стрілка на рис. 3.2), запис може мати максимум 255 пар ключ-значення.

У кожній парі ключ також зберігається в одному байті (друга стрілка на рис. 3.2). Бібліотека має вбудований словник найбільш часто використовуваних ключів, наприклад: `username`, `password`, `email`, `cvc`, `code`, `ip`, `secret`, `pin`. У випадку відсутності необхідного логічного імені ключа можна скористатися більш загальними: `key`, `value`, `name`, `text`.

У кожній парі значення зберігається після ключа як рядок із використанням кодування тексту UTF-8. Кінець рядка визначається нульовим байтом (третя стрілка на рис. 3.2).

3.2 Опис реалізованої бібліотеки

Бібліотека реалізована мовою програмування C у редакторі Visual Studio Code. Для зручної розробки на різних операційних системах, зручності компіляції, надання абстракції над різними компіляторами використано систему автоматизації збирання CMake. Завдяки її широкому поширенню, створений проєкт було легко інтегрувати з редактором коду та отримати підтримку мови не гірше ніж в інтегрованих середовищах розробки. CMake проєкт також значно полегшить використання інструментів Emscripten для використання бібліотеки у браузері, як було визначено раніше.

Файли бібліотеки згруповані та відображують основні кроки які вона виконує: розбір текстового документу, серіалізація, стиснення, шифрування, збереження у двійковий формат (рис. 3.3).

Мова C не має області імен, тому для запобігання конфліктів усі функції мають префікс – `mtsd` (скорочено від `My Top Secret Data`). За

можливості, функції позначаються із ключовим словом `static` для того щоб їх не було видно за межами файлу в якому вони оголошені.

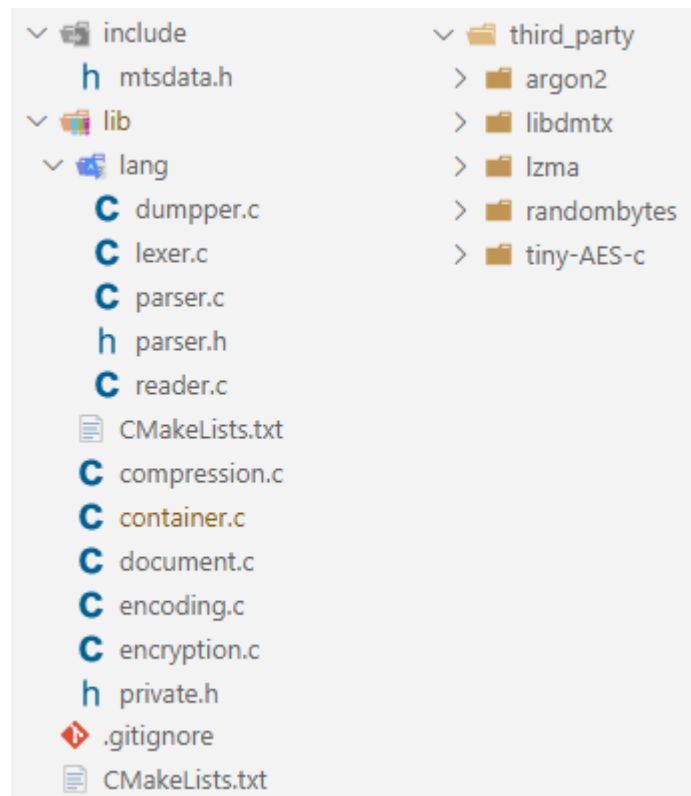


Рисунок 3.3 – Файлова структура бібліотеки в редакторі VS Code

Будучи мовою системного програмування, С не має виняткових ситуацій. Типові альтернативні рішення для обробки помилок:

- використання глобальної змінної для збереження коду помилки;
- повернення коду помилки функцією, збереження результату в один із аргументів при виклику;
- `goto` ланцюги обробки;
- функції `setjmp` та `longjmp`;
- у випадку розроблення застосунку використання макросу `assert` та аварійне завершення процесу (функція `abort`) [29].

Розроблена бібліотека використовує дещо інший підхід. Той хто використовує бібліотеку повинен реалізувати функцію (`mtsd_error`), яка буде викликатися бібліотекою у разі виникнення помилок, відповідно до її

оголошення у файлу заголовку (`include/mtsdata.h`). В якості аргументів функція приймає джерело виникнення помилки (джерелом є власне бібліотека чи одна із її залежностей), код помилки (унікальний в межах джерела), необов'язковий рядок з описом помилки.

Кожна функція, виклик якої може призвести до помилки, повертає статус виконання (перерахування `mtsd_res` із `MTSD_ERR` та `MTSD_OK`). Якщо функція, яка викликає її не може обробити помилку, то вона «заражається» і повинна також повернути відповідний статус. Для зменшення об'єму умовних виразів у вихідному коді, покращення його читабельності використовуються макроси `MTSD_CHECK`, `MTSD_CHECK_GOTO`.

Будучи мовою системного програмування, C не має автоматичного управління пам'яттю. Для виділення та звільнення динамічної пам'яті необхідно робити системні виклики або викликати обгортки навколо них із стандартної бібліотеки. Реалізована бібліотека делегує визначення цих функцій (`mtsd_malloc`, `mtsd_realloc`, `mtsd_free`) застосунку, що її використовує, відповідно до їх оголошення у файлу заголовку. Аналогічно до обробки помилок для зменшення об'єму умовних виразів у вихідному коді, покращення його читабельності використовуються макроси `MTSD_MALLOC`, `MTSD_REALLOC`, `MTSD_FREE`.

Відповідно до рисунку 2.4 першим етапом виконується синтаксичний розбір текстового документу. Система синтаксичного аналізу, відповідає типовій (рис. 2.3) та реалізована у файлах:

- `reader.c`, зчитувач;
- `lexer.c`, лексичний аналіз;
- `parser.c`, синтаксичний аналіз.

Зчитувач побайтово зчитує потік даних та трансформує у символи відповідно до використаної таблиці кодування символів. В розробленій бібліотеці підтримується кодування UTF-8, перевірка правильності зчитаних октетів, послідовностей октетів. Додатково зчитувач відслідковує номер поточного рядка та стовпця у тексті, кінець документу. Для отримання

наступного байту використовується функція зворотного виклику, яка визначається у кінцевому застосунку. У таблиці 3.2 наведено структуру даних для виконання синтаксичного аналізу.

Таблиця 3.2 – Опис структури парсеру (mtsd_parser)

Назва поля	Тип	Опис
line	size_t	Номер рядка
column	size_t	Номер стовпця у рядку
offset	size_t	Кількість прочитаних байтів
input.callback	mtsd_read_callback	Функція зворотного виклику отримання наступного байту
input.data	void*	Довільні дані, які будуть надані як аргумент до функції
reader.mb_char	uint8_t[4]	Октети для збереження UTF-8 символу
reader.mb_size	size_t	Кількість зайнятих октетів для символу
reader.eof	int	Флаг, що позначає кінець зчитаних даних
lexer.kind	mtsd_token_kind	Тип лексеми
lexer.start	size_t	Позиція початку лексеми у файлі
lexer.end	size_t	Позиція кінця лексеми
lexer.buffer	uint8_t*	Буфер для збереження тексту лексеми
lexer.buffer_size	size_t	Розмір буферу для тексту
lexer.consumed	int	Флаг, що визначає чи була лексема успішно використана

Підсистема лексичного аналізу зчитує символ за символом та формує лексеми. Розроблений синтаксис для документу не має спеціальних ключових символів, тому кожен документ завжди породжує деяку послідовність лексем. Правильність лексем буде перевірена на наступному кроці. Також виконується очищення тексту: прибираються зайві символи пробілу, стандартизуються символи закінчення рядків (CRLF, LF). Підтримувані лексеми знаходяться у перерахуванні mtsd_token_kind:

- MTSD_KEY_TOKEN, ключ;
- MTSD_VALUE_TOKEN, значення;

- MTSD_VALUE_MULTILINE_TOKEN, рядок багаторядкового значення;
- MTSD_RECORD_SEPARATOR_TOKEN, роздільник записів;
- MTSD_STREAM_START_TOKEN, технічна лексема початку документу;
- MTSD_STREAM_END_TOKEN, технічна лексема кінця документу.

На рисунку 3.4а зображено документ, що містить усі можливі лексеми.

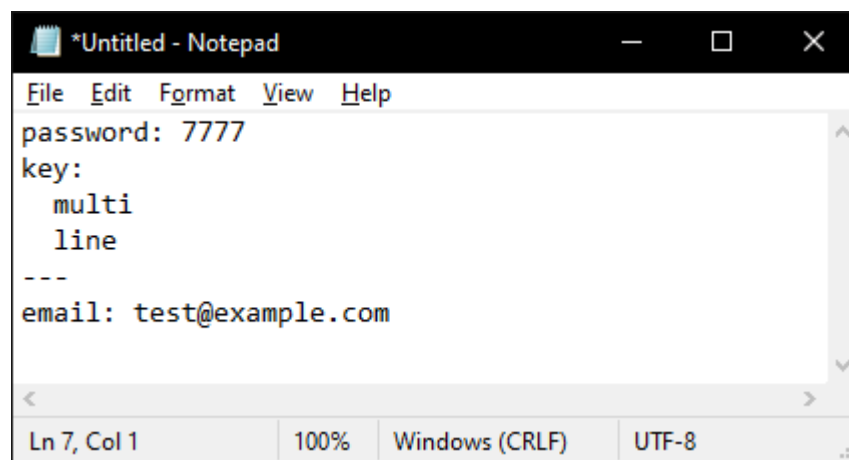


Рисунок 3.4а – Приклад документу з усіма можливими лексемами

Прибравши коментарі до коду, що відповідає за вивід лексем до командного рядка, можемо побачити, які лексеми бачить підсистема синтаксичного розбору (рис. 3.4б).

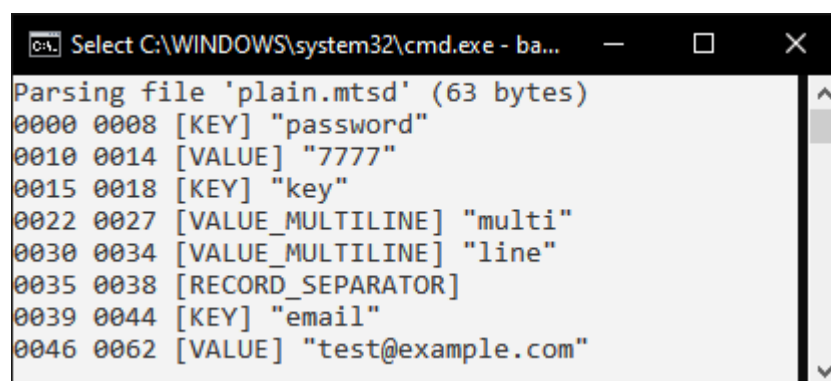


Рисунок 3.4б – Лексеми у документі на рисунку 3.4а

Синтаксичний розбір перевіряє правильність слідування лексем та одночасно формує дерево розбору, що представляє текстовий документ. У разі виявлення помилки появи неочікуваної лексеми чи її відсутності, можна легко отримати позиції рядка та колонки де вона відбулась. Розроблена бібліотека використовує спадний метод побудови дерева, тобто будується від кореня до кінцевих вузлів.

Корінь документа складається із записів, які відділяються один від одного лексемою «роздільник записів». Кожен запис містить набір пар ключ-значення, де значення може бути одно- чи багаторядковим. Структури, що представляють вузли дерева широко використовують однобічно зв'язний список, а корінь дерева це всього лише вказівник на перший запис у документі (рис. 3.5).

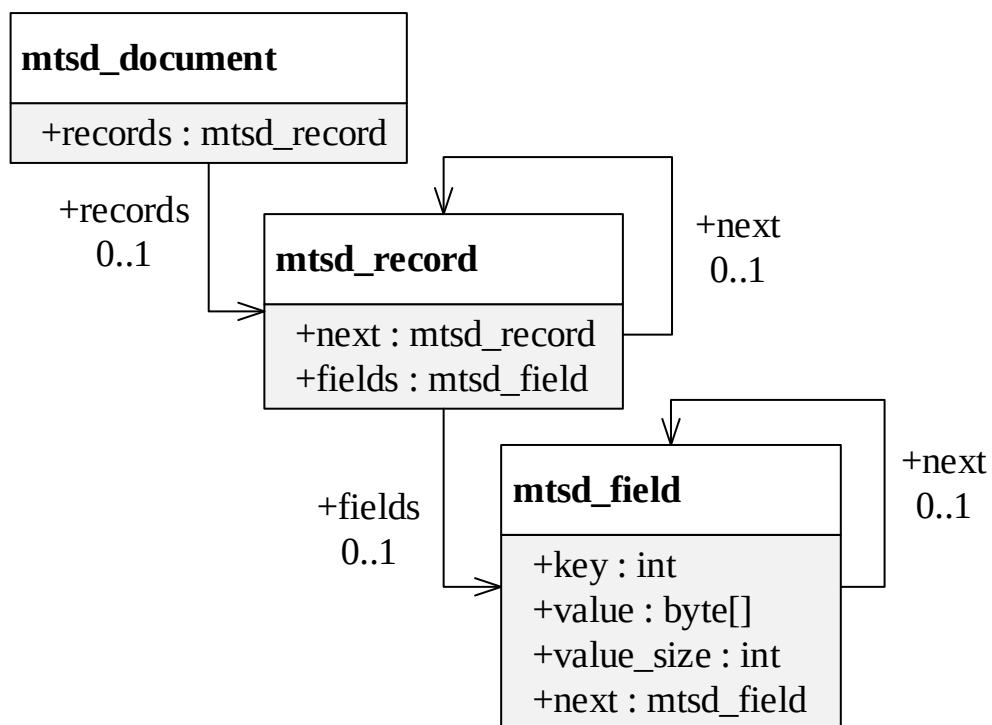


Рисунок 3.5 – Діаграма структур, що утворюють дерево

Відповідно до рисунку 2.4 наступним кроком є серіалізація, тобто перетворення дерева у безперервну послідовність байтів. Вона визначена у файлі `encoding.c` та виконується за правилами описаними в розділі 3.1.

У файлі `compression.c` містяться функції для стиснення та відновлення даних. Вони є невеликими обгортками навколо бібліотеки, що реалізує LZMA. На сайті 7-zip можна отримати LZMA SDK, який надає документацію, приклади коду, файли заголовків, бібліотеки та інструменти, необхідні для розробки програм, що використовують стиснення LZMA. Рівень стиснення обраний найбільший – 9, розмір словника обмежений тільки розміром даних, додаткові параметри для збереження контексту експериментально підібрані для найменшого розміру метаданих алгоритму, вимкнена позначка кінця стиснених даних через відсутність необхідності в ній.

У файлі `encryption.c` містяться функції для шифрування та дешифрування послідовності байтів. Вони є невеликими обгортками навколо бібліотеки `tiny-AES-c`, яка власне реалізує алгоритм AES із режимами ECB, CTR, CBC. Дана бібліотека має тести перевірки шифрування на основі опублікованих зразкових векторів режимів роботи AES. Для формування ключа шифрування використовується функція формування ключа Argon2 із однойменної бібліотеки її зразкової реалізації. Параметри для неї та спосіб їх знаходження описані в розділі 3.1.

Завершаючий етап створення визначений у файлі `container.c`, який має функцію обчислення циклічного надлишкового коду, отримання поточного часу системи та упакування їх разом із шифрованими даними відповідно до розробленого заголовку файлу (таблиця 3.1).

Розроблена бібліотека є кросплатформеною та викликає необхідні для неї системні функції у таблиці 3.3.

Таблиця 3.3 – Використані API на різних платформах

Платформа/Ціль виклику функції	Отримання поточного часу створення файлу	Отримати криптографічно стійкі випадкові значення для формування солі
Windows	<code>GetSystemTimeAsFileTime</code>	<code>CryptGenRandom</code>
POSIX	<code>gettimeofday</code>	<code>getrandom</code>
WASM	<code>Date.now</code>	<code>crypto.getRandomValues</code>

3.3 Опис реалізованого вебзастосунку

Вебзастосунок створений із використанням бібліотеки Vue.js версії 3 у редакторі Visual Studio Code. Для початку розробки встановлено середовище виконання JavaScript на сервері Node.js та менеджер пакетів Yarn. Після чого завантажено утиліту vue-cli:

```
yarn global add @vue/cli  
vue create
```

Дана утиліта створює проєкт, додає необхідні команди для запуску у режимі розробки, побудови файлів для розгортання, надає вибір додаткових компонентів, створює мінімальну структуру файлів. При створенні проєкту було обрано: створення на мові TypeScript, підтримка PWA, Linter, бібліотеку навігації vue-router. Кінцева структура проєкту зображена на рис. 3.6.

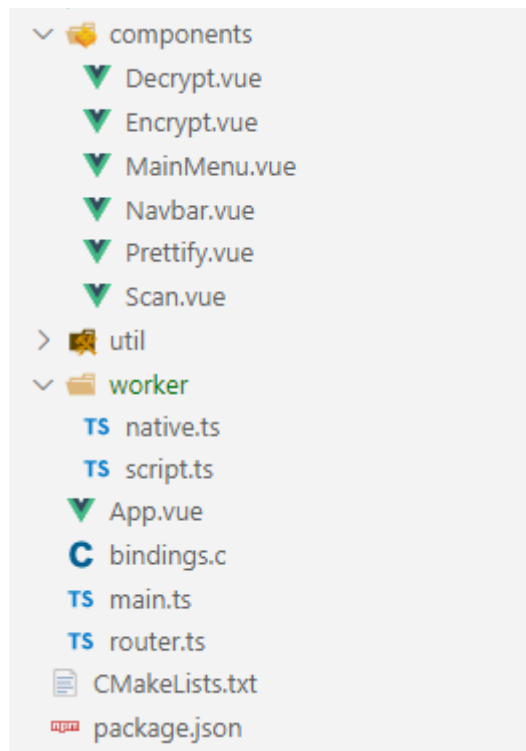


Рисунок 3.6 – Файлова структура вебзастосунку в редакторі VS Code

Бібліотека навігації vue-router тісно інтегрується з Vue.js та перетворює проєкт у односторінковий застосунок (single-page application). На відміну від традиційних сторінок, де кожний перехід за посиланням викликає повне завантаження сторінки із HTML коду сформованим сервером, односторінковий застосунок завантажує статичні HTML, JavaScript, CSS файли на початку роботи, перехід за посиланням оброблюється у JavaScript та змінює сторінку повністю у браузері користувача, необхідний динамічний контент завантажується асинхронно із серверу у деякому форматі обміну даними, зазвичай JSON. Для виконання навігації обрано режим хеш, тобто логічний шлях додається після символу «#», стандартна функціональність якого є перехід до фрагменту сторінки (визначеного атрибутом id) при її завантаженні. Даний режим не потребує додаткового налаштування з боку сервера, але не є дружнім до пошукових системи. Так як розроблений застосунок не має сенсу індексувати, то обраний режим є обґрунтованим. Доданий бібліотекою тег «router-view» відображає компонент, що відповідає поточному шляху, тег «router-link» є невеликою обгорткою навколо тегу «a» та відображає посилання. У файлі router.ts створюється об'єкт навігації та зв'язуються шляхи із компонентами як наведено у таблиці 3.4.

Таблиця 3.4 – Шлях та компонент, що за нього відповідає

Шлях	Компонент	Опис
#/	MainMenu.vue	Головний екран навігації
#/decrypt	Decrypt.vue	Містить форму дешифрування файлу
#/decrypt/scan	Scan.vue	Відображає відеопотік з камери для розпізнання 2D коду
#/encrypt	Encrypt.vue	Містить форму для шифрування текстового документу
#/prettify	Prettify.vue	Відображає результат шифрування із можливістю збереження у різні формати

Структурною одиницею розробки є компонент та їх композиція. Файл компоненту поділений на розділи: шаблон, сценарій, стилі. На кроці побудови

проєкту для розгортання розділи компонентів об'єднуються в один файл стилів, сценаріїв. У розділі опису сценарію визначається функція `setup`, яка повертає об'єкт із полями, які будуть доступними із шаблону. Значеннями полів є реактивні значення, функції, обчислювані значення.

Користувач починає роботу з головного меню. Одне з посилань веде до компоненту шифрування. Компонент шифрування містить довідник формату документа, просту форму із двох полів: текст документа, парольна фраза. Шифрування виконується розробленою бібліотекою у потоці `Web Worker`. На час шифрування кнопка шифрування стає неактивною. У таблиці 3.5 наведено перелік даних компоненту.

Таблиця 3.5 – Внутрішні дані компоненту шифрування

Назва	Тип	Опис
Реактивні дані		
<code>rawText</code>	<code>string</code>	Поле форми, вихідний текст для шифрування
<code>secret</code>	<code>string</code>	Поле форми, парольна фраза
<code>isEncrypting</code>	<code>boolean</code>	Флаг, чи виконується шифрування зараз
<code>encryptError</code>	<code>string null</code>	Рядок, що містить текст помилки під час шифрування
<code>showHelp</code>	<code>boolean</code>	Флаг, який визначає чи відображається підказка до текстового формату документа
Функції		
<code>encrypt</code>	<code>void</code>	Почати шифрування
<code>toggleHelp</code>	<code>void</code>	Змінити стан відображення довідки
Відслідковування даних		
<code>rawText</code>		У разі редагування тексту попередня помилка очищується

У разі успішного шифрування користувач перенаправляється на екран збереження, шифровані дані поміщуються у спільний стан компонентів:

```
export const globalState = shallowReactive({
  encrypted: null as Uint8Array | null
})
```

В якості спільного стану використано простий реактивний об'єкт, у більш складних застосунках варто використовувати бібліотеку централізованого сховища Vuex. Користувач може зберегти файл як: звичайний двійковий файл, у транспортному кодуванні (base16, base64), як зображення 2D коду.

Компоненти шифрування та дешифрування використовують розроблену бібліотеку, для її компіляції у WASM використано набір інструментів Emscripten. Зазвичай, функції із таких бібліотек не викликаються у чистому вигляді, а створюється невелика обгортка навколо викликів, наприклад обгортка на мові програмування C# буде інкапсулювати виклики та надавати об'єктно орієнтований інтерфейс, що є звичним для цієї мови програмування. Файл «bindings.c» є невеликим мостом між бібліотекою та її використанням у застосунку. В ньому визначено функції, які потребує бібліотека: виділення пам'яті та обробки помилок. Файл «native.ts» є аналогічним мостом, але з боку JavaScript. Він відповідає за завантаження WASM модуля, містить деякі константи, які співпадають із тими, що визначені у бібліотеці, надає типізовані функції обгортки, які у свою чергу викликають функції бібліотеки.

Розроблений вебзастосунок не завантажує файл «native.ts» із WASM модулем бібліотеки в основному потоці виконання. Натомість, для цього використовується Web Workers API. Сценарій, що виконується у додатковому потоці визначено у файлі «worker/script.ts». Застосунок використовує бібліотеку Comlink, яка завдяки сучасним можливостям стандарту ECMAScript (Проху об'єкти) перетворює подійно-орієнтовну взаємодію із Web Worker у стиль RPC, повертає Promise об'єкти. Файл «worker/interface.ts» запускає сценарій Web Worker із основного потоку та взаємодіє з ним через бібліотеку Comlink. Умовно структуру файлів (директорії «worker») разом із потоками їх виконання зображено на рисунку 3.7.

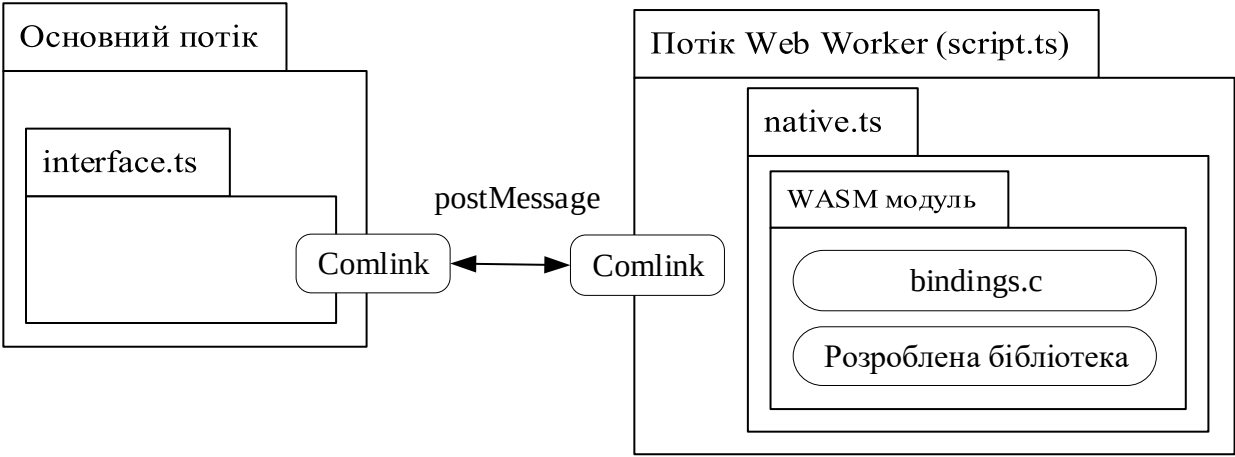


Рисунок 3.7 – Потоки виконання та файли, що в них завантажені

Компонент збереження шифрованих даних та компонент сканування залежать від створення та розпізнання 2D коду відповідно. Під час розробки було розглянуто шляхи задоволення цієї потреби (табл. 3.6).

Таблиця 3.6 – API та бібліотеки для роботи з 2D кодом

Назва	Розпізнання	Створення	Нотатки
Barcode Detection API	Aztec, Data Matrix, QR	Не передбачено	Стандарт у процесі розробки, все ще низька підтримка
ZXing	5 форматів	5 форматів	API для створення не має просунутих налаштувань
bwip-js	Відсутнє	Усі існуючі	–
libdmtx	Data Matrix	Data Matrix	–

Таким чином було обрано бібліотеку libdmtx, не зважаючи на те що вона підтримує тільки один вид 2D коду, підтримується як розпізнання так і створення. API для створення має безліч параметрів налаштування, а для розпізнання коду є цікавий параметр ліміту витраченого часу. Дана бібліотека написана мовою C, тому її було об’єднано в один WASM модуль із власною бібліотекою.

Компонент сканування 2D коду при відкритті використовує MediaDevices API, запитує доступ до тилової камери пристрою. У разі отримання доступу, при кожному перемальовуванні сторінки (функція requestAnimationFrame), кадр з відеопотоку камери відображається на

елементі canvas. Додатково, якщо потік Web Worker зараз не розпізнає попередній кадр, то поточний копіюється та відправляється на обробку до бібліотеки libdmtx. На рисунку 3.8 зображено результат профілювання, ліміт часу розпізнання встановлено на 41 мілісекунду, як можна побачити потік із розпізнанням завжди зайнятий, в той же час виконання копії поточного кадру в основному потоці займає 16 мілісекунд. Після знаходження 2D коду шифровані дані зберігаються до спільного стану компонентів та виконується перехід до сторінки дешифрування.

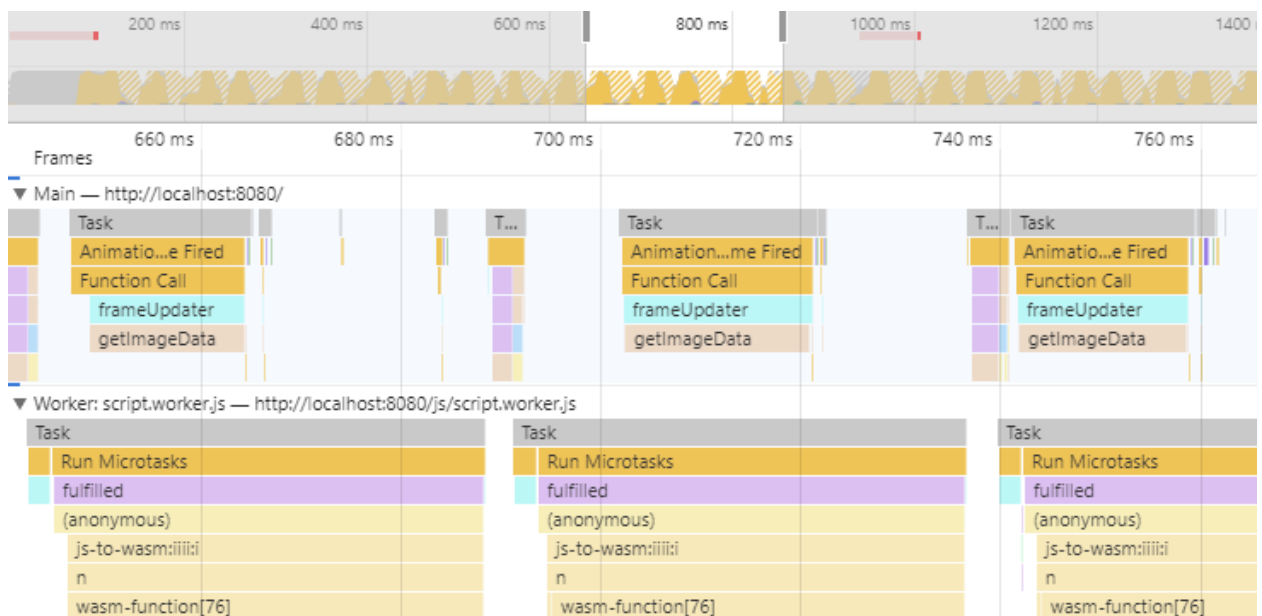


Рисунок 3.8 – Результат профілювання сторінки сканування коду

Для надання єдиного стилю інтерфейсу застосунку, єдиного ритму відступів та адаптивності використано CSS фреймворк tailwindcss. Цей фреймворк побудований на принципі мінімалізму, один клас визначає лише одну-дві CSS властивості. Декілька таких класів на одному елементі дозволяють отримати бажаний результат без написання власних стилів, швидше експериментувати, не витратити час на вигадкування унікальних назв класів.

Для перетворення застосунку у прогресивний, достатньо лише вказати підтримку PWA при створенні проєкту через vue-cli. Файл, який буде

відповідати за перехоплення мережових запитів, автоматично генерується плагіном для системи побудови проєкту Webpack.

Сучасний стан розвитку вебтехнологій дуже швидкий, за різних причин поточний стандарт технологій може не підтримуватися у браузері користувача. Дана проблема вирішується завдяки транспіляції та поліфілам, які виконує компілятор Babel для JavaScript сценаріїв. У свою чергу для CSS використовується інструмент трансформації PostCSS із плагіном autoprefixer. Дані інструменти також були автоматично встановлені при створенні проєкту.

3.4 Висновок за розділом 3

В даному розділі було описано та обґрунтовано улаштування розробленого файлу із шифрованими даними. Описано використання рішення для типових проблем, які виникають при реалізації бібліотеки на мові C. Наведено опис структур для виконання синтаксичного аналізу та збереження побудованого дерева. Наведено деталі реалізації для досягнення багатоплатформеності.

Для вебзастосунку надано структуру шляхів та компонентів, описано використання глобального стану між компонентами. Описано спосіб завантаження WASM модуля у додатковий потік та спосіб взаємодії із ним. Розглянуто реалізацію сканування, створення двовимірного штрих-коду та бібліотеку обрану для цього. Описано реалізацію графічного інтерфейсу.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення програми

Програмний продукт призначений для шифрування текстового документу підтримуваного формату та збереження його у текстовому вигляді, бінарному чи двовимірному штрих-кодi. Програмний продукт має можливість розпізнання створених штрих-кодів камерою та інших способів збереження, виконує дешифрування вихідного документу із них.

Даний продукт автоматично одноразово завантажується при відкритті вебсторінки застосунку та може працювати без Інтернет з'єднання.

4.2 Робота користувача із застосунком

Для початку роботи користувач запускає його через іконку на домашньому екрані пристрою або відкриває адресу сайту. Якщо сайт відкрито у браузері, то йому пропонується додати іконку на домашній екран (рис. 4.1).

У головному меню, що відкрилося, містяться посилання для початку сканування 2D коду, дешифрування, шифрування.

Якщо користувач перейшов до шифрування відкривається екран із формою, яка має поля: текстовий документ, парольна фраза (рис. 4.2). Обидва поля є не обов'язковими до заповнення. Після чого натискається кнопка Encrypt та очікується завершення операції. У разі виникнення будь-яких помилок буде зображено елемент червоного кольору із текстом помилки. За необхідності можна звернутися до підказки щодо формату очікуваного тексту (кнопка «Document format reference»).

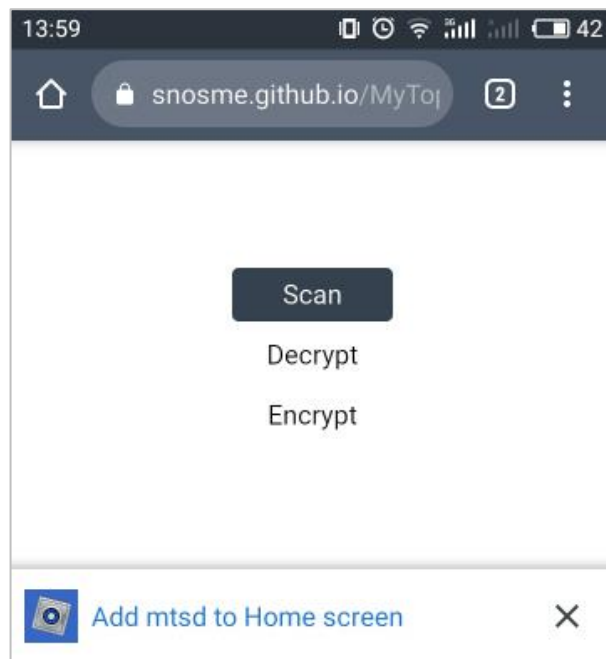


Рисунок 4.1 – Головне меню застосунку відкритого у браузері

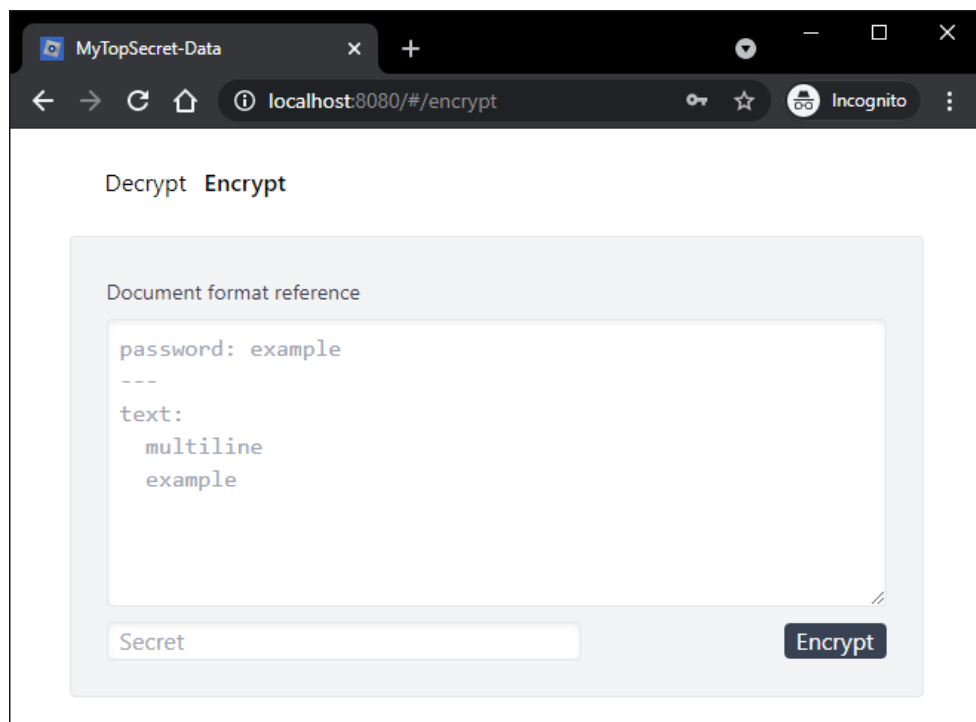


Рисунок 4.2 – Екран шифрування

У разі успіху відбувається перехід до екрану збереження шифрованих даних (рис. 4.3). Перший варіант збереження – завантажити двійковий файл кнопкою «Download `_mtsd.bin`». Другий варіант збереження – скопіювати текст. Для цього тексту можна налаштувати: кількість символів в одній групі

символів, кількість груп на одному рядку, абетку для шифрування (hex або base64). Також можна завантажити шрифт для друку, який використовує застосунок. Третій варіант збереження – зображення із 2D-матричним кодом. Для зображення можна налаштувати розмір модуля у пікселях та ширину відступів.

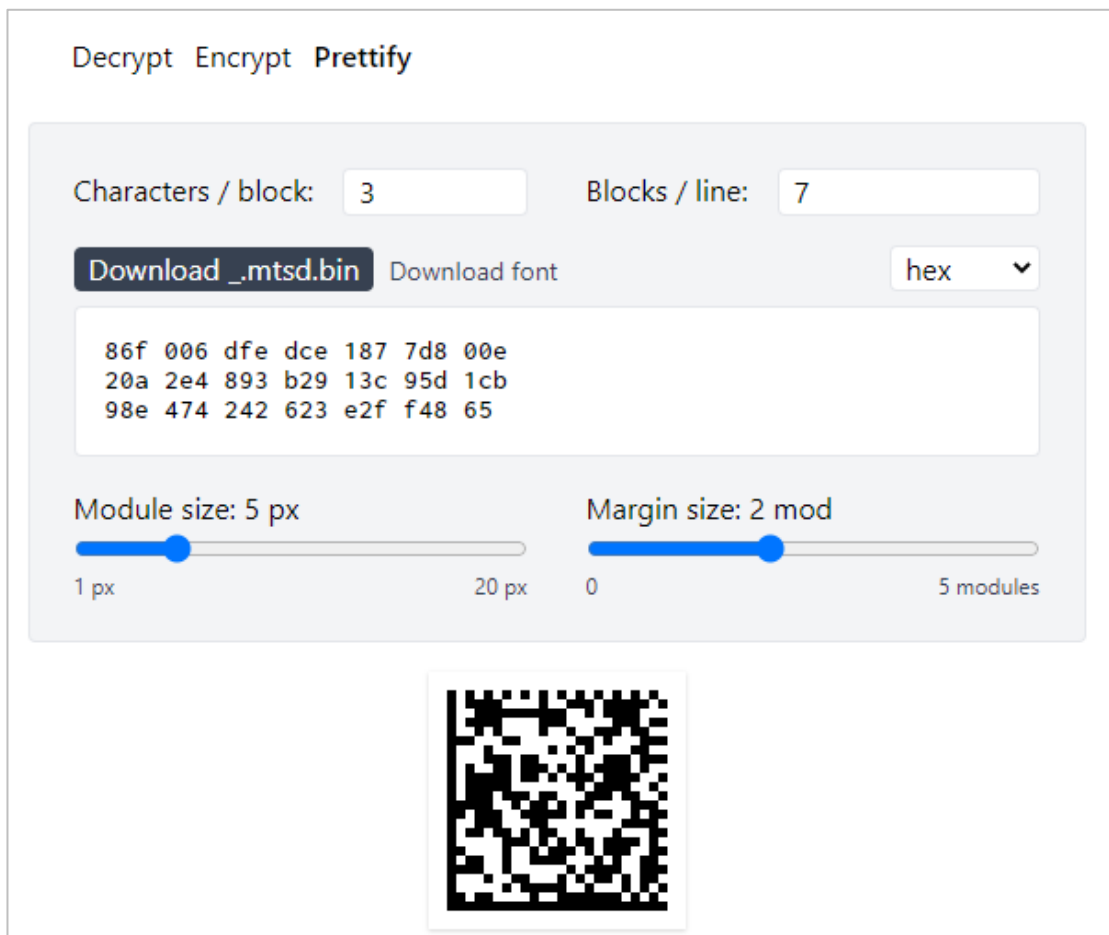


Рисунок 4.3 – Екран збереження шифрованих даних

Якщо користувач із головного меню перейшов до сканування, то спочатку повинен буде погодитися із наданням прав до камери пристрою. Після чого навести камеру на 2D код. Якщо код не вдається розпізнати, наприклад через погане освітлення, можна відредагувати ліміт часу для розпізнання (рис. 4.4). Після успішного розпізнання користувач перенаправляється на екран дешифрування.

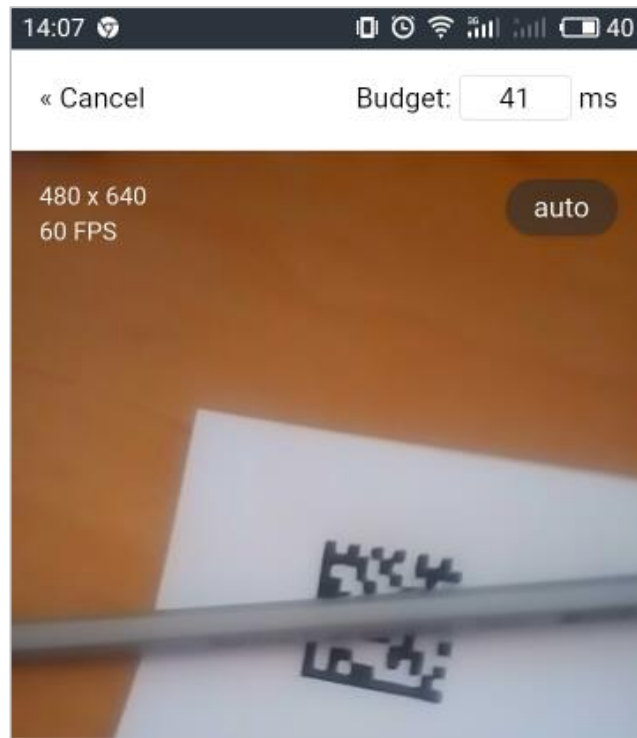


Рисунок 4.4 – Екран сканування 2D коду

Якщо користувач із головного меню перейшов до дешифрування відображається форма із полями: шифровані дані у вигляді тексту, парольна фраза (рис. 4.5).

Заповнити текстове поле можна відповідно до можливих способів збереження:

- обрати файл на пристрої кнопкою File;
- ввести самостійно та вказати використану абетку у випадяючому списку (hex або base64);
- сканувати 2D код.

Якщо введений текст дійсний, то буде зображена відповідна позначка та виведено час шифрування у місцевому часі (з точністю до годин). Після натиснення кнопки Open із правильною парольною фразою відображаються розшифровані записи.

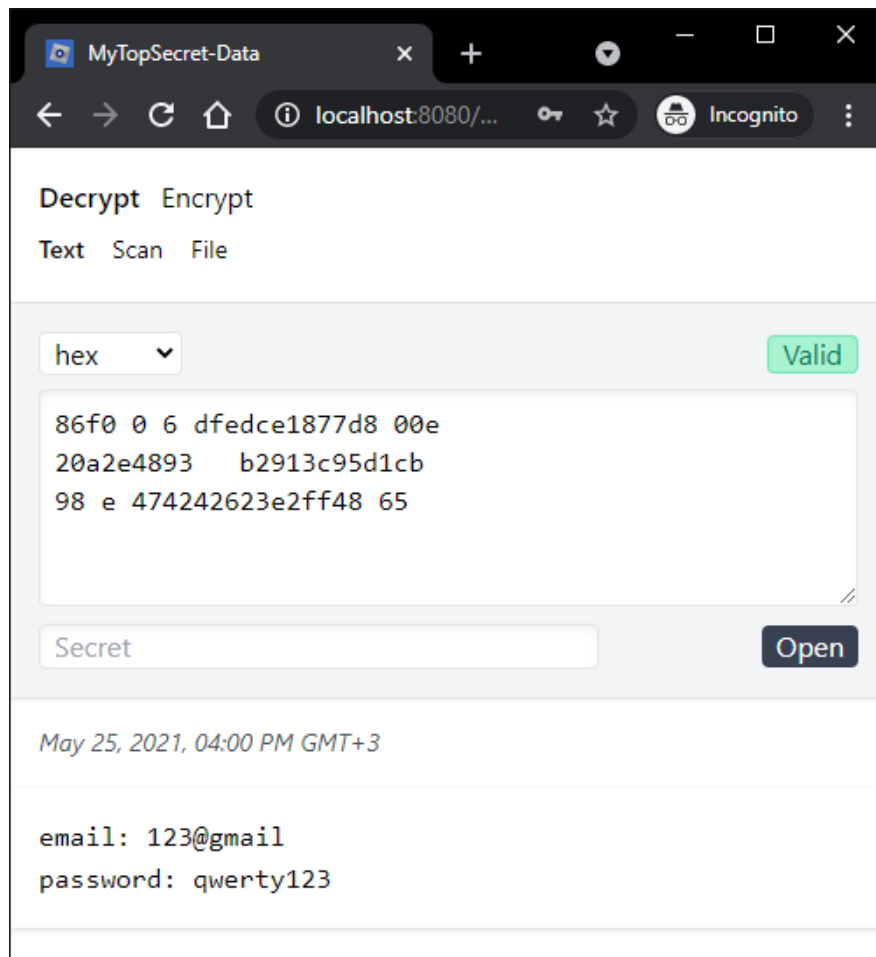


Рисунок 4.5 – Екран дешифрування

4.3 Експериментальне дослідження програми

В ході проведення експерименту було зашифровано документ у вебзастосунку. В якості вихідних даних обрано запис, який був використан для порівняння існуючих рішень у розділі 1.3: «website.com; test@example.com – 11112222». У розробленому текстовому форматі документа він виглядає як:

```
url: website.com
email: test@example.com
password: 11112222
```

Зашифровані дані було завантажено на сторінці як двійковий файл. Розмір файлу склав 47 байт. У таблиці 4.1 наведено відносне порівняння розмірів із таблиці 1.1 до розміру файлу створеного у застосунку.

Таблиця 4.1 – Порівняння розмірів шифрованих даних

Назва додатку	Розмір шифрованого файлу, байт	У порівнянні із розробленим застосунком
Вихідний текстовий файл	39 (вихідний текст)	0,83
Створений застосунок	47	1
1Password, менеджер паролів	1941	41
KeePass, менеджер паролів	1550	33
7-zip, архівування з паролем	218	4,6
Таблиця Excel захищена паролем	14336	305
96Crypt, програма шифрування	53	1,12

У порівнянні з вихідними даними файл є більшим лише на 8 байтів, що збіглося із розміром заголовку файлу. Також файл виявився меншим за розміром ніж той, що був створений програмою шифрування 96Crypt.

Таким чином, файл є меншим за розміром ніж усі проаналізовані аналоги, може бути використан для створення 2D коду та друку у вигляді невеликої кількості текстових символів.

ВИСНОВКИ

У роботі було досліджено основні поняття шифрування та способ збереження двійкових даних у сумісному вигляді для друкування. В ході аналізу було визначено, що таким є двовимірні штрих-коди, які бажано використовувати для кодування невеликих повідомлень. Було визначено ключові параметри, які впливають на розмір шифрованого файлу.

В результаті роботи створено вебзастосунок для шифрування тексту, який створює шифровані дані відносно однакового розміру до вихідних даних, такі невеликі шифровані дані можна використовувати для друку (текст та двовимірний штрих-код), їх введення назад у застосунок та сканування для подальшого дешифрування.

На мові програмування С розроблено бібліотеку для шифрування, яка має підсистему синтаксичного аналізу для розбору текстового документу. Розроблено вебзастосунок, який надає інтерфейс до бібліотеки та вирішує питання інтенсивних обчислень у браузері за допомогою технології Web Worker та WebAssembly. Застосунок – адаптивний, підтримується використання на мобільних пристроях та комп'ютерах. Останнім кроком було розміщення застосунку у публічному доступі на сторінках GitHub Pages.

У роботі експериментально доведено, що створений застосунком файл є меншим за розміром ніж ті, що були створені існуючими аналогами. Всі завдання, поставленні у роботі бакалавра, були повністю виконані.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Digital Identity Guidelines: Authentication and Lifecycle Management [Electronic resource] / P. Grassi, E. Newton, R. Perlner, A. Regenscheid, W. Burr, J. Richer, N. Lefkovitz, J. Danker, Y. Choong, K. Greene, M. Theofanos. – Access mode: <https://doi.org/10.6028/NIST.SP.800-63b>.
2. Ives, B. F1. The Domino Effect of Password Reuse [Text] / B. Ives, K. R. Walsh, H. Schneider // Commun. ACM. – 2004. – Vol. 47, №4. – P. 75–78.
3. Schneier B. Foundations / B. Schneier // Applied Cryptography, Second Edition. – John Wiley & Sons, Ltd, 2015. – Section. 1. – 1-18 p.
4. Шифрування // Великий тлумачний словник сучасної української мови / уклад. та голов. ред. В. Т. Бусел. – Київ; Ірпінь: Перун, 2005. – VIII, 1728 с.
5. Kessler, G. C. An Overview of Cryptography [Electronic resource] / G. C. Kessler. – Access mode: <https://www.garykessler.net/library/crypto.html>.
6. Moriarty, K. PKCS #5: Password-Based Cryptography Specification Version 2.1 [Electronic resource] / K. Moriarty, B. Kaliski, A. Rusch. – Access mode: <https://www.rfc-editor.org/rfc/rfc8018.txt>.
7. Жураковський Б. Ю. Сфери застосування двовимірних штрихових кодів / Б. Ю. Жураковський, Н. М. Довженко // Системи управління, навігації та зв'язку. – 2016. – Вип. 2. – С. 83-87. – Режим доступу: http://nbuv.gov.ua/UJRN/suntz_2016_2_21.
8. Hušek J. The use of cryptography in 7-zip [Electronic resource] / J. Hušek. – Access mode: <https://dspace.cvut.cz/handle/10467/83024>.
9. Malone, T. Interoperability in Programming Languages [Text] / T. Malone // Scholarly Horizons: University of Minnesota, Morris Undergraduate Journal. – 2014. – Vol. 1, № 2. – P. 1-6.
10. Application binary interface [Electronic resource] // Wikipedia, The Free Encyclopedia. – Access mode: https://en.wikipedia.org/wiki/Application_binary_interface.

11. Advantages and Disadvantages of C Programming – Discover the Secrets of C [Electronic resource]. – Access mode: <https://data-flair.training/blogs/advantages-and-disadvantages-of-c>.
12. WebAssembly Introduction [Electronic resource]. – Access mode: <https://webassembly.github.io/spec/core/intro/introduction.html>.
13. WebAssembly Portability [Electronic resource]. – Access mode: <https://webassembly.org/docs/portability>.
14. WebAssembly Concepts [Electronic resource]. – Access mode: <https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts>.
15. ECMAScript® 2020 language specification : ECMA-262 Edition 11. – [Effective from 2020-06-01]. – Geneva : Ecma International, 2020. – 860 p.
16. TypeScript [Electronic resource] // Wikipedia, The Free Encyclopedia. – Access mode: <https://en.wikipedia.org/wiki/TypeScript>.
17. Pekarsky, M. Does your web app need a front-end framework? [Electronic resource] / M. Pekarsky. – Access mode: <https://stackoverflow.blog/2020/02/03/is-it-time-for-a-front-end-framework>.
18. Introduction to Angular concepts [Electronic resource] // Angular. – Access mode: <https://angular.io/guide/architecture>.
19. Evans, J. React vs ReactDOM [Electronic resource] / J. Evans – Access mode: <https://coursework.vschool.io/react-vs-reactdom>.
20. Introducing JSX [Electronic resource] // React. – Access mode: <https://reactjs.org/docs/introducing-jsx.html>.
21. Change Detection Caveats in Vue 2 [Electronic resource] // Vue.js. – Access mode: <https://v3.vuejs.org/guide/change-detection.html>.
22. Bouchefra, A. A Beginner's Guide to Vue CLI [Electronic resource] / A. Bouchefra. – Access mode: <https://www.sitepoint.com/vue-cli-intro>.
23. Technologies – Front-end Frameworks [Electronic resource]. – Access mode: <https://2020.stateofjs.com/en-US/technologies/front-end-frameworks>.
24. Parsing [Electronic resource] // Wikipedia, The Free Encyclopedia. – Access mode: <https://en.wikipedia.org/wiki/Parsing>.

25. Progressive web apps (PWAs) [Electronic resource]. – Access mode: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps.
26. Web Workers API [Electronic resource]. – Access mode: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API.
27. Tutorial: How to set up and automatically deploy your website to GitHub Pages [Electronic resource]. – Access mode: <https://roelofjanelsinga.com/articles/how-to-set-up-automatically-deploy-website-github-pages>.
28. Biryukov, A. Argon2: the memory-hard function for password hashing and other applications [Electronic resource] / A. Biryukov, D. Dinu, D. Khovratovich. – Access mode: <https://github.com/P-H-C/phc-winner-argon2/blob/master/argon2-specs.pdf>.
29. Svoboda, D. Beyond errno Error Handling in C [Electronic resource] / D. Svoboda. – Access mode: <https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=484203>.

ДОДАТОК А
Текст програми

A.1. Файл package.json

```
{
  "name": "web-app",
  "version": "1.0.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
  },
  "dependencies": {
    "comlink": "^4.3.1",
    "register-service-worker": "^1.7.2",
    "tailwindcss": "^2.1.2",
    "vue": "^3.0.4",
    "vue-router": "^4.0.3"
  },
  "devDependencies": {
    "@types/emscripten": "^1.39.4",
    "@typescript-eslint/eslint-plugin": "^4.15.1",
    "@typescript-eslint/parser": "^4.15.1",
    "@vue/cli-plugin-eslint": "5.0.0-beta.1",
    "@vue/cli-plugin-pwa": "5.0.0-beta.1",
    "@vue/cli-plugin-router": "5.0.0-beta.1",
    "@vue/cli-plugin-typescript": "5.0.0-beta.1",
    "@vue/cli-service": "5.0.0-beta.1",
    "@vue/compiler-sfc": "^3.0.4",
    "@vue/eslint-config-standard": "^6.0.0",
    "@vue/eslint-config-typescript": "^7.0.0",
    "eslint": "^7.20.0",
    "eslint-plugin-import": "^2.20.2",
    "eslint-plugin-node": "^11.1.0",
    "eslint-plugin-promise": "^4.3.1",
    "eslint-plugin-vue": "^7.2.0",
    "typescript": "~4.1.5",
    "worker-loader": "^3.0.8"
  },
  "eslintConfig": {
    "root": true,
    "env": {
      "node": true
    },
    "extends": [
      "plugin:vue/vue3-essential",
      "@vue/standard",
      "@vue/typescript/recommended"
    ],
    "parserOptions": {
      "ecmaVersion": 2020
    },
    "rules": {},
    "ignorePatterns": [
      "src/worker/mtsd-native.js"
    ]
  },
  "postcss": {
    "plugins": {
      "tailwindcss": {},
      "autoprefixer": {}
    }
  },
}
```

```

    "browserslist": [
        "> 1%",
        "last 2 versions",
        "not dead",
        "not ie 11"
    ]
}

```

A.2. Файл CMakeLists.txt

```

add_subdirectory("../third_party/libdmtx"
"${PROJECT_BINARY_DIR}/third_party/libdmtx")

add_executable(mtsd-native src/bindings.c)

target_link_libraries(mtsd-native PUBLIC
    mtsd
    dmtx)

target_include_directories(mtsd-native PUBLIC "${PROJECT_BINARY_DIR}")

target_link_options(mtsd-native PRIVATE
    "SHELL:-s WASM=1"
    "SHELL:-s FILESYSTEM=0"
    "SHELL:-s MODULARIZE=1"
    "SHELL:-s STRICT=1"
    "SHELL:-s SINGLE_FILE=1"
    "SHELL:-s ALLOW_MEMORY_GROWTH=1"
    "SHELL:-s MEMORY_GROWTH_GEOMETRIC_STEP=0"
    "SHELL:-s
EXPORTED_FUNCTIONS='[_malloc,_free,_b_mtsd_decrypt,_mtsd_container_is_valid,_
mtsd_container_get_date,_b_dmtx_find_regions,_b_dmtx_create,_b_last_error,_b_
mtsd_encrypt]'"
    "SHELL:-s
EXTRA_EXPORTED_RUNTIME_METHODS='[stackSave,stackAlloc,stackRestore]'"
    "-O3"
    "-fno-rtti"
    "-fno-exceptions"
)

```

A.3. Файл native.ts

```

/// <reference types="emscripten" />

interface MtsdNative extends EmscriptenModule { /* eslint-disable camelcase */
    _b_mtsd_decrypt(encryptedDataPtr: number, encryptedSize: number,
        passwordPtr: number, passLen: number): number

    _mtsd_container_is_valid(encryptedDataPtr: number, encryptedSize: number):
        number

    _mtsd_container_get_date(encryptedDataPtr: number): number

    _b_dmtx_find_regions(imgDataPtr: number, imgW: number, imgH: number, budget:
        number): number

    _b_dmtx_create(dataPtr: number, dataSize: number,

```

```

        modulePx: number, marginPx: number,
        widthPtr: number, heightPtr: number): number

    _b_last_error(srcPtr: number, codePtr: number, msgPtr: number, msgLenPtr:
number): void

    _b_mtsd_encrypt(textPtr: number, textSize: number,
        passwordPtr: number, passLen: number): number

    stackSave(): number
    stackRestore(ptr: number): void
    stackAlloc(size: number): number
}

// eslint-disable-next-line @typescript-eslint/no-var-requires
const Module = require('./mtsd-native.js') as
EmscriptenModuleFactory<MtsdNative>

let module_: ReturnType<typeof Module>

function load () {
    if (!module_) {
        module_ = Module()
    }
    return module_
}

const TEXT_DECODER = new TextDecoder('utf-8')
const TEXT_ENCODER = new TextEncoder()

export const KEYID = [
    '__NULL',
    'email',
    'password',
    'login'
]

interface MtsdField {
    key: string
    value: string
}
type MtsdRecord = MtsdField[]
export type MtsdDocument = MtsdRecord[]

function unpackMtsdDocument (packed: Uint8Array): MtsdDocument {
    let offset = 0
    const reader = new DataView(packed.buffer, packed.byteOffset,
packed.byteLength)

    const recordsCount = reader.getInt32(offset, true)
    offset += 4
    return new Array(recordsCount).fill(undefined).map<MtsdRecord>(() => {
        const fieldsCount = reader.getInt32(offset, true)
        offset += 4
        return new Array(fieldsCount).fill(undefined).map<MtsdField>(() => {
            const keyid = reader.getUint8(offset)
            offset += 1
            const valueLength = reader.getInt32(offset, true)
            offset += 4
            const value = TEXT_DECODER.decode(packed.subarray(offset, offset +
valueLength))
            offset += valueLength
            return {
                key: KEYID[keyid],

```

```

        value: value
    }
    })
  })
}

export interface DmtxRegion {
  posMatrix: number[]
  data: Uint8Array | null
}

function unpackDmtxRegions (packed: Uint8Array): DmtxRegion[] {
  let offset = 0
  const reader = new DataView(packed.buffer, packed.byteOffset,
    packed.byteLength)

  const regionsCount = reader.getInt32(offset, true)
  offset += 4
  return new Array(regionsCount).fill(undefined).map<DmtxRegion>(() => {
    const posMatrix = new Array(9).fill(undefined).map<number>((_, idx) =>
      reader.getFloat64(offset + idx * 8, true)
    )
    offset += posMatrix.length * 8

    const dataLength = reader.getInt32(offset, true)
    offset += 4
    let data: DmtxRegion['data'] = null
    if (dataLength) {
      data = new Uint8Array(dataLength)
      data.set(packed.subarray(offset, offset + dataLength))
      offset += dataLength
    }

    return {
      posMatrix,
      data
    }
  })
}

export async function mtsdDecrypt (data: Uint8Array, password: string):
Promise<MtsdDocument> {
  const module = await load()

  const encryptedPtr = module._malloc(data.byteLength)
  module.HEAPU8.set(data, encryptedPtr)

  const passUtf8 = TEXT_ENCODER.encode(password)
  const passPtr = module._malloc(passUtf8.byteLength)
  module.HEAPU8.set(passUtf8, passPtr)

  try {
    const binPtr = module._b_mtsd_decrypt(encryptedPtr, data.byteLength,
      passPtr, passUtf8.byteLength)
    if (binPtr) {
      const doc = unpackMtsdDocument(module.HEAPU8.subarray(binPtr))
      module._free(binPtr)
      return doc
    } else {
      throw new Error('Invalid password') // TODO: handle lowmem error
    }
  } finally {
    module._free(encryptedPtr)
    module._free(passPtr)
  }
}

```

```

    }
}

export async function mtsdIsValid (data: Uint8Array): Promise<boolean> {
    const module = await load()

    const encryptedPtr = module._malloc(data.byteLength)
    module.HEAPU8.set(data, encryptedPtr)

    const      isValid      =      module._mtsd_container_is_valid(encryptedPtr,
data.byteLength)
    module._free(encryptedPtr)

    return Boolean(isValid)
}

export function mtsdCreationDate (data: Uint8Array): Date {
    const OFFSET = 2
    const hours = (
        ((data[OFFSET + 0] & 0x7F) << (8 * 2)) |
        ((data[OFFSET + 1]) << (8 * 1)) |
        ((data[OFFSET + 2]) << (8 * 0))
    )
    return new Date(hours * 3600 * 1000)
}

export async function dmtxFindRegions (image: ImageData, budgetMs: number):
Promise<DmtxRegion[]> {
    const module = await load()

    const imgDataPtr = module._malloc(image.data.byteLength)
    module.HEAPU8.set(image.data, imgDataPtr)

    const  resPtr  =  module._b_dmtx_find_regions(imgDataPtr,  image.width,
image.height, budgetMs)
    module._free(imgDataPtr)

    if (resPtr) {
        const res = unpackDmtxRegions(module.HEAPU8.subarray(resPtr))
        module._free(resPtr)
        return res
    } else {
        return []
    }
}

export async function dmtxCreate (data: Uint8Array, modulePx: number, marginPx:
number): Promise<ImageData> {
    const module = await load()

    const _stack = module.stackSave()
    const widthPtr = module.stackAlloc(4)
    const heightPtr = module.stackAlloc(4)

    const dataPtr = module._malloc(data.byteLength)
    module.HEAPU8.set(data, dataPtr)

    const imgPtr = module._b_dmtx_create(dataPtr,  data.byteLength,  modulePx,
marginPx, widthPtr, heightPtr)
    module._free(dataPtr)

    if (!imgPtr) {
        module.stackRestore(_stack)
        throw new Error('Cannot create Data Matrix')
    }
}

```

```

    }

    const width = module.HEAP32[widthPtr / 4]
    const height = module.HEAP32[heightPtr / 4]
    module.stackRestore(_stack)

    const imgArray = new Uint8ClampedArray(
        module.HEAPU8.subarray(imgPtr, imgPtr + (4 * width * height))
    )
    module._free(imgPtr)

    return new ImageData(imgArray, width, height)
}

export async function mtsdEncrypt (text: string, password: string):
Promise<Uint8Array> {
    const module = await load()

    const textUtf8 = TEXT_ENCODER.encode(text)
    const textPtr = module._malloc(textUtf8.byteLength)
    module.HEAPU8.set(textUtf8, textPtr)

    const passUtf8 = TEXT_ENCODER.encode(password)
    const passPtr = module._malloc(passUtf8.byteLength)
    module.HEAPU8.set(passUtf8, passPtr)

    try {
        const binPtr = module._b_mtsd_encrypt(textPtr, textUtf8.byteLength,
passPtr, passUtf8.byteLength)
        if (binPtr) {
            const encryptedSize = module.HEAP32[binPtr / 4]
            const encrypted = new Uint8Array(module.HEAPU8.subarray(binPtr + 4,
(binPtr + 4) + encryptedSize))
            module._free(binPtr)
            return encrypted
        } else {
            throw new Error('Cannot encrypt')
        }
    } finally {
        module._free(textPtr)
        module._free(passPtr)
    }
}

const ERROR_SOURCE = [
    'MTSD_ESELF',
    'MTSD_EARGON2',
    'MTSD_ELZMA',
    'MTSD_ERANDOMBYTES'
]

const MTSD_ERROR = [
    'MTSD_EMEMORY',
    'MTSD_EREADER',
    'MTSD_EENCODE_RECORD_SIZE',
    'MTSD_EENCODE_PAYLOAD_SIZE',
    'MTSD_EDECODE_CORRUPTED_PAYLOAD',
    'MTSD_EPARSE_UNKNOWN_KEY',
    'MTSD_ETIME'
]

interface MtsdError {
    source: string
    code: number | string
}

```

```

    message: string | null
  }

export async function getLastError (): Promise<MtsdError> {
  const module = await load()

  let error: MtsdError

  const _stack = module.stackSave()
  {
    const srcPtr = module.stackAlloc(4)
    const codePtr = module.stackAlloc(4)
    const msgPtr = module.stackAlloc(4)
    const msgLenPtr = module.stackAlloc(4)

    module._b_last_error(srcPtr, codePtr, msgPtr, msgLenPtr)

    const src = module.HEAP32[srcPtr / 4]
    const code = module.HEAP32[codePtr / 4]
    const msgOffset = module.HEAP32[msgPtr / 4]
    const msgLen = module.HEAP32[msgLenPtr / 4]

    error = {
      source: ERROR_SOURCE[src],
      code: src === 0
        ? MTSD_ERROR[code]
        : code,
      message: msgLen
        ? TEXT_DECODER.decode(module.HEAPU8.subarray(msgOffset, msgOffset +
msgLen))
        : null
    }
  }
  module.stackRestore(_stack)

  return error
}

```

A.4. Файл script.ts

```

import * as Comlink from 'comlink'
import {
  mtsdDecrypt,
  mtsdEncrypt,
  mtsdIsValid,
  dmtxCreate,
  dmtxFindRegions,
  DmtxRegion,
  getLastError
} from './native'

const WorkerBody = {
  mtsdDecrypt (data: Uint8Array, password: string) {
    return mtsdDecrypt(data, password)
  },
  mtsdIsValid (data: Uint8Array) {
    return mtsdIsValid(data)
  },
  async dmtxFindRegions (image: ImageData, budgetMs: number):
Promise<DmtxRegion[]> {

```

```

    const res = await dmtxFindRegions(image, budgetMs)
    return Comlink.transfer(res, res.filter(region => region.data).map(region
=> region.data!.buffer))
  },
  async dmtxCreate (data: Uint8Array, modulePx: number, marginPx: number):
Promise<ImageData> {
    const res = await dmtxCreate(data, modulePx, marginPx)
    return Comlink.transfer(res, [res.data.buffer])
  },
  async mtsdEncrypt (text: string, password: string): Promise<Uint8Array> {
    const res = await mtsdEncrypt(text, password)
    return Comlink.transfer(res, [res.buffer])
  },
  getLastError () {
    return getLastError()
  }
}

Comlink.expose(WorkerBody)

export type WorkerRPC = Comlink.Remote<typeof WorkerBody>

```

A.5. Файл Decrypt.vue

```

<template>
  <navbar>
    <template #secondary>
      <div class="flex -mx-1 mt-1 text-sm">
        <span class="p-1 font-semibold">Text</span>
        <router-link to="/decrypt/scan" class="p-1 ml-2">Scan</router-link>
        <label type="button" class="p-1 ml-2 cursor-pointer"
for="decrypt_file">File</label>
        <input @input="fromFile" type="file" id="decrypt_file" class="hidden">
      </div>
    </template>
  </navbar>
  <page-content class="relative">
    <form>
      <div class="flex mb-2 items-center">
        <select class="border rounded mr-2 px-1">
          <option>hex</option>
          <option>base64</option>
          <option>bocrl6</option>
        </select>
        <div v-if="encoded.isValid !== undefined"
class="border rounded px-2 leading-tight ml-auto"
: class="encoded.isValid ? 'bg-green-200 border-green-300 text-green-
700' : 'bg-red-200 border-red-300 text-red-700'"
>{{ encoded.isValid ? 'Valid' : 'Invalid' }}</div>
      </div>
      <textarea class="border w-full font-mono leading-snug p-2 rounded shadow-
inner mb-1"
spellcheck="false" rows="8"
v-model="encoded.text"></textarea>
      <div class="flex items-center justify-between flex-wrap -m-1">
        <input type="password"
placeholder="Secret"
:disabled="decryption.isRunning"
class="border shadow-inner rounded px-2 min-w-0 m-1 flex-grow max-w-
xs"

```

```

        v-model="encoded.password" />
<button class="rounded px-2 text-white m-1"
        :disabled="!decryptBtn"
        :class="decryptBtn ? 'bg-gray-700' : 'bg-gray-400'"
        @click.prevent="decrypt"
        >Open</button>
</div>
<div v-if="decryption.error"
        class="bg-red-200 rounded text-red-700 px-2 mt-2 inline-block">{{
decryption.error }}</div>
</form>
</page-content>
<div class="max-w-xl w-full">
    <div v-if="encodedAt"
        class="mb-4 text-sm italic text-gray-600 px-4 sm:px-6">{{ encodedAt
    }}</div>
    <pre v-if="doc"
        class="shadow p-4 sm:rounded mb-4"
        v-html="doc" />
    </div>
</template>

<script lang="ts">
import { computed, defineComponent, reactive, watch, ref } from 'vue'
import Navbar from './Navbar.vue'
import PageContent from './PageContent.vue'
import { thread } from '@worker/interface'
import { mtsdCreationDate } from '@worker/native'
import { toBocrl6, fromBocrl6 } from '@util/bocrl6'
import { globalState } from '@util/global'

export default defineComponent({
  components: { Navbar, PageContent },
  setup () {
    const encoded = reactive({
      text: '',
      password: '',
      isValid: undefined as boolean | undefined
    })

    if (globalState.encrypted) {
      encoded.text = toBocrl6(globalState.encrypted)
    }

    const textBin = computed(() => {
      const text = encoded.text.replaceAll(/\s+/g, '')
      if (!text.length) return null

      try {
        return fromBocrl6(text)
      } catch (e) {
        return null
      }
    })

    watch(textBin, async () => {
      if (!textBin.value) {
        encoded.isValid = !encoded.text.trim().length ? undefined : false
        return
      }
      if (encoded.isValid) {
        encoded.isValid = undefined
      }
      encoded.isValid = await thread.mtsdIsValid(textBin.value)
    })
  }
})

```

```

    }, { immediate: true })

    const encodedAt = computed(() => {
      if (!encoded.isValid || !textBin.value) {
        return undefined
      }
      return mtsdCreationDate(textBin.value).toLocaleString(undefined, { year:
'numeric', month: 'short', day: '2-digit', hour: '2-digit', minute: '2-digit',
timeZoneName: 'short' })
    })

    const decryption = reactive({
      isRunning: false,
      error: ''
    })
    const doc = ref(null as null | string)
    const decryptBtn = computed(() => encoded.isValid && !decryption.isRunning)
    async function decrypt () {
      try {
        decryption.isRunning = true
        decryption.error = ''
        const pass = encoded.password
        encoded.password = ''
        const doc_ = await thread.mtsdDecrypt(textBin.value!, pass)
        doc.value = doc_.map(record =>
          record.map(field => {
            const isMultiline = field.value.includes('\n')
            return isMultiline
              ? `${field.key}:\n${field.value.split('\n').map(ln =>
`${ln}`)}.join('\n')}`
              : `${field.key}: ${field.value}`
          }).join('\n')
        ).join('\n---\n')
      } catch (e) {
        decryption.error = (e as Error).message
      } finally {
        decryption.isRunning = false
      }
    }

    async function fromFile (e: Event) {
      const file = (e.target as HTMLInputElement)!.files![0]
      const content = new Uint8Array(await file.arrayBuffer())
      encoded.text = toBocrl6(content)
      globalState.encrypted = content
    }

    return {
      encoded,
      encodedAt,
      decryptBtn,
      decrypt,
      decryption,
      doc,
      fromFile
    }
  }
})
</script>

<style lang="postcss">
</style>

```

A.6. Файл Encrypt.vue

```

<template>
  <navbar />
  <page-content>
    <button @click="toggleHelp" class="text-sm text-gray-700 mb-2">Document
    format reference</button>
    <div v-if="showHelp" class="mb-2 border border-gray-200 rounded p-2">
      <p>Document consists of records separated by three dashes.</p>
      <p>Each record has multiple key-value pairs. Key and value are separated
      by a colon. Keys can be repeated.
        The value is a text, either on same line or starts on next line indented
        by 2 spaces.</p>
      <p>Supported keys: <i>{{ supportedKeys }}</i>.</p>
    </div>
    <form>
      <textarea placeholder="password:      example&#10;---&#10;text:&#10;
      multiline&#10; example"
        class="border w-full font-mono leading-snug p-2 rounded shadow-inner
        mb-1"
        spellcheck="false" rows="8"
        v-model="rawText" />
      <div class="flex items-center justify-between flex-wrap -m-1">
        <input type="password"
          placeholder="Secret"
          :disabled="isEncrypting"
          class="border shadow-inner rounded px-2 min-w-0 m-1 flex-grow max-w-
          xs"
          v-model="secret" />
        <button class="rounded px-2 text-white m-1"
          :disabled="isEncrypting"
          :class="!isEncrypting ? 'bg-gray-700' : 'bg-gray-400'"
          @click.prevent="encrypt"
          >Encrypt</button>
      </div>
      <div v-if="encryptError"
        class="bg-red-200 rounded text-red-700 px-2 mt-2 inline-block">{{
        encryptError }}</div>
    </form>
  </page-content>
</template>

<script lang="ts">
import { defineComponent, shallowRef, watch } from 'vue'
import { thread } from '@worker/interface'
import Navbar from './Navbar.vue'
import PageContent from './PageContent.vue'
import { globalState } from '@util/global'
import { KEYID } from '../worker/native'
import router from '@router'

export default defineComponent({
  components: { Navbar, PageContent },
  setup () {
    const rawText = shallowRef('')
    const secret = shallowRef('')
    const isEncrypting = shallowRef(false)
    const encryptError = shallowRef<string | null>(null)
    const showHelp = shallowRef(false)

    async function encrypt () {
      isEncrypting.value = true

```

```

    try {
      const secretText = secret.value
      secret.value = ''
      globalState.encrypted = await thread.mtsdEncrypt(rawText.value,
secretText)
      await router.push('/prettify')
    } catch (e) {
      const err = await thread.getLastError()
      encryptError.value = err.message || e.message
    } finally {
      isEncrypting.value = false
    }
  }

  function toggleHelp () {
    showHelp.value = !showHelp.value
  }

  watch(rawText, () => {
    encryptError.value = null
  })

  return {
    rawText,
    isEncrypting,
    secret,
    encrypt,
    encryptError,
    showHelp,
    toggleHelp,
    supportedKeys: KEYID.slice(1).sort().join(', ')
  }
}
})
</script>

```

A.7. Файл MainMenu.vue

```

<template>
  <div class="flex-1 flex items-center">
    <div class="flex flex-col text-center">
      <router-link to="/decrypt/scan" class="py-1 px-2 rounded text-white bg-
gray-700 w-24">Scan</router-link>
      <router-link to="/decrypt" class="py-1 px-2 mt-1 rounded hover:bg-gray-
200">Decrypt</router-link>
      <router-link to="/encrypt" class="py-1 px-2 mt-1 rounded hover:bg-gray-
200">Encrypt</router-link>
    </div>
  </div>
</template>

```

A.8. Файл Navbar.vue

```

<template>
  <div class="p-4 sm:p-6 max-w-xl border-b sm:border-none w-full">
    <slot name="primary">

```

```

    <div class="flex -m-1" id="nav">
      <router-link to="/decrypt" class="p-1">Decrypt</router-link>
      <router-link to="/encrypt" class="p-1 ml-1">Encrypt</router-link>
      <router-link to="/prettify" class="p-1 ml-1" v-
if="showPrettify">Prettify</router-link>
    </div>
  </slot>
  <slot name="secondary" />
</div>
</template>

<script lang="ts">
import { defineComponent, computed } from 'vue'
import { globalState } from '@/util/global'

export default defineComponent({
  setup () {
    return {
      showPrettify: computed(() => Boolean(globalState.encrypted))
    }
  }
})
</script>

<style lang="postcss">
#nav > .router-link-active {
  @apply font-semibold;
}
</style>

```

A.9. Файл Prettify.vue

```

<template>
  <navbar />
  <page-content>
    <div class="flex-wrap sm:flex-nowrap flex items-baseline">
      <div class="w-full sm:w-1/2 flex">
        <label for="bocr16_chars" class="whitespace-nowrap mr-4">Characters /
block:</label>
        <input v-model.number="charsPerBlock" type="number" id="bocr16_chars"
min="1" :class="$style.numberInput">
      </div>
      <div class="w-full sm:w-1/2 flex sm:ml-8 mb-4 mt-1 sm:mt-0">
        <label for="bocr16_blocks" class="whitespace-nowrap mr-4">Blocks /
line:</label>
        <input v-model.number="blocksPerLine" type="number" id="bocr16_blocks"
min="1" :class="$style.numberInput">
      </div>
    </div>
    <div class="flex mb-2 items-baseline">
      <button class="rounded px-2 text-white bg-gray-700"
@click="saveMtsd">Download _mtsd.bin</button>
      <a href="`${publicPath}fonts/InconsolataSemiExpanded-SemiBold.ttf`"
download
class="text-sm text-gray-700 ml-2 hidden sm:block">Download font</a>
      <select class="ml-auto border rounded ml-2 px-1">
        <option>hex</option>
        <option>base64</option>
        <option>bocr16</option>
      </select>

```

```

</div>
<pre v-if="encryptedText"
  :class="$style.encryptedText" class="font-ocr"
  v-html="encryptedText" />
<div class="mt-4 sm:flex">
  <div class="flex-1 mb-4 sm:mb-0">
    <label for="dmtx_module_size">Module size: {{ moduleSize }} px</label>
    <div>
      <input
        v-model.number="moduleSize"
        type="range"
        id="dmtx_module_size" min="1" max="20" class="w-full">
      <div :class="$style.rangeInputBounds">
        <span>1 px</span><span>20 px</span>
      </div>
    </div>
  </div>
  <div class="flex-1 sm:ml-8">
    <label for="dmtx_margin">Margin size: {{ marginSize }} mod</label>
    <div>
      <input v-model.number="marginSize" type="range" id="dmtx_margin"
        min="0" max="5" class="w-full">
      <div :class="$style.rangeInputBounds">
        <span>0</span><span>5 modules</span>
      </div>
    </div>
  </div>
</div>
</page-content>

</template>

<script lang="ts">
import { defineComponent, shallowRef, watch, computed } from 'vue'
import Navbar from './Navbar.vue'
import PageContent from './PageContent.vue'
import { thread } from '@worker/interface'
import { toBocrl6 } from '@util/bocrl6'
import { saveFileAs } from '@util/saveFileAs'
import { globalState } from '@util/global'

export default defineComponent({
  components: { Navbar, PageContent },
  setup () {
    const charsPerBlock = shallowRef(5)
    const blocksPerLine = shallowRef(4)

    const encryptedText = computed(() => {
      // eslint-disable-next-line @typescript-eslint/no-non-null-assertion
      return toBocrl6(globalState.encrypted!)
        .match(new RegExp(`.{1,${charsPerBlock.value}}`, 'g'))!.join(' ')
        .match(new RegExp(`.{1,${(charsPerBlock.value + 1) *
          blocksPerLine.value}}`, 'g'))!
        .map(_ => _.trim())
        .join('\n')
    })

    function saveMtsd () {
      saveFileAs(globalState.encrypted!, '_mtsd.bin', 'application/octet-
        stream')
    }

    // --- Data Matrix ---

```

```

const dmtxImg = shallowRef<string | null>(null)
const moduleSize = shallowRef(5)
const marginSize = shallowRef(2)

const canvas = document.createElement('canvas')
const canvasCtx = canvas.getContext('2d', { alpha: false })
if (!canvasCtx) throw new Error('Browser is not supported (canvas 2d)')

watch([moduleSize, marginSize], async () => {
  const img = await thread.dmtxCreate(globalState.encrypted!,
moduleSize.value, moduleSize.value * marginSize.value)
  canvas.width = img.width
  canvas.height = img.height
  canvasCtx.putImageData(img, 0, 0)
  dmtxImg.value = canvas.toDataURL('image/png')
}, { immediate: true })

return {
  dmtxImg,
  moduleSize,
  marginSize,
  encryptedText,
  charsPerBlock,
  blocksPerLine,
  saveMtsd,
  publicPath: process.env.BASE_URL
}
})
</script>

<style lang="postcss" module>
.dmtxImg {
  @apply mx-auto mb-4 shadow;
  image-rendering: pixelated;
}

.rangeInputBounds {
  @apply flex justify-between text-xs text-gray-700 leading-none;
}

.encryptedText {
  @apply p-4 bg-white rounded border overflow-auto text-sm;
  line-height: 1.15;
  letter-spacing: 0.03125em;
}

.numberInput {
  @apply min-w-0 w-full border rounded px-2;
}
</style>

```

A.10. Файл Scan.vue

```

<template>
  <div class="p-4 sm:p-6 border-b sm:border-none w-full max-w-xl flex justify-
between items-baseline">
    <router-link to="/decrypt">&laquo; Cancel</router-link>
    <div>Budget: <input v-model.number="finderBudget" type="number" class="w-
16 text-center border rounded" min="4"> ms</div>

```

```

</div>
<div v-if="!streamInfo.error"
  class="relative flex-1 overflow-hidden mb-4"
  style="max-height: 100vh;"
>
  <video ref="video" />
  <div class="absolute origin-top-left"
    :style="{
      width: streamInfo.width + 'px',
      height: streamInfo.height + 'px',
      transform: `scale(${streamInfo.scale}) scaleY(-1)`
    }">
    <!-- eslint-disable-next-line vue/require-v-for-key -->
    <div v-for="region in dtmxRegions"
      :class="[$style.region, region.isValid ? 'bg-green-500' : 'bg-red-
500']"
      :style="`transform: matrix3d(${region.matrix})`"
    />
  </div>
  <div class="absolute right-0 top-0 p-4">
    <button class="px-4 py-1 rounded-full bg-gray-800 shadow text-white bg-
opacity-50"
      @click="autoSelect = !autoSelect"
      :class="{ 'line-through': !autoSelect }">auto</button>
  </div>
  <div class="absolute left-0 top-0 p-4 text-sm text-white">
    <div v-if="streamInfo.width">{{ streamInfo.width }} x {{
streamInfo.height }}</div>
    <div v-if="streamInfo.fps">{{ streamInfo.fps }} FPS</div>
  </div>
  <div v-if="streamInfo.error"
    class="bg-red-200 rounded text-red-700 px-2 m-4"
    >{{ streamInfo.error }}</div>
</template>

<script lang="ts">
import { defineComponent, reactive, onMounted, onBeforeUnmount, shallowRef }
from 'vue'
import { thread } from '@worker/interface'
import * as Comlink from 'comlink'
import { globalState } from '@util/global'
import router from '@router'

export default defineComponent({
  setup () {
    const video = shallowRef<HTMLVideoElement | null>(null)
    const streamInfo = reactive({
      width: 0,
      height: 0,
      fps: 0,
      scale: 1,
      error: ''
    })
    const dtmxRegions = shallowRef([] as Array<{
      matrix: string
      isValid: boolean
    }>)
    const finderBudget = shallowRef(41 /* 24 fps */)
    const autoSelect = shallowRef(true)

    let stream = null as MediaStream | null

    const canvas = document.createElement('canvas')

```

```

const canvasCtx = canvas.getContext('2d', { alpha: false })!

let rafId = 0
let nextFrame = null as ImageData | null
let isProcessingFrame = false

function frameUpdater () {
  if (!nextFrame) {
    canvasCtx.drawImage(video.value!, 0, 0)
    nextFrame = canvasCtx.getImageData(0, 0, canvas.width, canvas.height)
  }

  rafId = requestAnimationFrame(frameUpdater)
  if (!isProcessingFrame) {
    processFrame()
  }
}

onBeforeUnmount(() => {
  cancelAnimationFrame(rafId)
  nextFrame = null

  if (stream) {
    stream.getTracks().forEach(track => {
      track.stop()
    })
  }
})

async function processFrame () {
  isProcessingFrame = true
  const image = nextFrame!
  nextFrame = null

  const found = await thread.dmtxFindRegions(
    Comlink.transfer(image, [image.data.buffer]),
    Number(finderBudget.value) || 4
  )

  if (autoSelect.value) {
    const firstValid = found.find(region => Boolean(region.data))
    if (firstValid) {
      globalState.encrypted = firstValid.data!
      await router.push('/decrypt')
      return
    }
  }

  dtmxRegions.value = found.map(region => {
    const mx = region.posMatrix
    return {
      matrix: (/* eslint-disable */
        `${mx[0]}, ${mx[1]},    0, ${mx[2]},` +
        `${mx[3]}, ${mx[4]},    0, ${mx[5]},` +
        `    0,    0,    1,    0,` +
        `${mx[6]}, ${mx[7]},    0, ${mx[8]}`
      ),
      isValid: Boolean(region.data)
    }
  })
})

isProcessingFrame = false
if (nextFrame) {
  processFrame()
}

```

```

    }
  }

  onMounted(async () => {
    const el = video.value!

    try {
      stream = await navigator.mediaDevices.getUserMedia({
        video: {
          facingMode: { ideal: 'environment' }
        }
      })
      const track = stream.getVideoTracks()[0]
      const cap = track.getCapabilities()

      // await track.applyConstraints({
      //   width: cap.width?.max,
      //   height: cap.height?.max
      // })

      el.srcObject = stream
      await el.play()

      const settings = track.getSettings()
      streamInfo.width = el.videoWidth
      streamInfo.height = el.videoHeight
      streamInfo.fps = Math.round(settings.frameRate ?? 0)
      streamInfo.scale = el.clientWidth / streamInfo.width

      canvas.width = streamInfo.width
      canvas.height = streamInfo.height

      rafId = requestAnimationFrame(frameUpdater)
    } catch (e) {
      if ((e as DOMException).name === 'NotFoundError') {
        streamInfo.error = 'No video devices found on your system'
      } else {
        streamInfo.error = `Unhandled error occurred (${String(e)} || 'n/a'
      })`
    }
  })

  return {
    video,
    streamInfo,
    dtmxRegions,
    finderBudget,
    autoSelect
  }
}
})
</script>

<style lang="postcss" module>
.region {
  @apply absolute origin-top-left bg-opacity-50;
  width: 1px;
  height: 1px;
}
</style>

```

A.11. Файл mtsdata.h

```

#ifndef _LIB_MTSD_H_
#define _LIB_MTSD_H_

#include <stdint.h>
#include <stddef.h>

typedef enum {
    MTSD_ERR = 0, // if (!ok) { handle... }
    MTSD_OK = 1, // if (ok) { continue... }
} mtsd_res;

typedef enum {
    MTSD_ESELF,
    MTSD_EARGON2,
    MTSD_ELZMA,
    MTSD_ERANDOMBYTES,
} mtsd_error_source;

typedef enum {
    MTSD_EMEMORY,
    MTSD_EREADER,
    MTSD_EENCODE_RECORD_SIZE,
    MTSD_EENCODE_PAYLOAD_SIZE,
    MTSD_EDECODE_CORRUPTED_PAYLOAD,
    MTSD_EPARSE_UNKNOWN_KEY,
    MTSD_ETIME,
} mtsd_err;

typedef struct mtsd_field {
    uint8_t key;
    uint8_t *value;
    size_t value_size;
    struct mtsd_field *next;
} mtsd_field;

typedef struct mtsd_record {
    mtsd_field *fields;
    struct mtsd_record *next;
} mtsd_record;

typedef struct {
    mtsd_record* records;
} mtsd_document;

typedef int (*mtsd_read_callback)(void *data,
                                   uint8_t *buffer,
                                   size_t size,
                                   size_t *size_read);

extern void mtsd_error(mtsd_error_source src, int error, char *msg);
extern void* mtsd_malloc(size_t size);
extern void* mtsd_realloc(void* ptr, size_t size);
extern void mtsd_free(void* ptr);

mtsd_res mtsd_parse(/* In */ mtsd_read_callback read_callback,
                   /* In */ void *callback_data,
                   /* Out */ mtsd_document *doc);

mtsd_res mtsd_encrypt(/* In */ mtsd_document* doc,
                     /* In */ uint8_t* password,

```

```

        /* In */ size_t password_len,
        /* Out */ uint8_t** encrypted,
        /* Out */ size_t* size);

mtsd_res mtsd_decrypt(/* In */ uint8_t* encrypted,
        /* In */ size_t size,
        /* In */ uint8_t* password,
        /* In */ size_t password_len,
        /* Out */ mtsd_document* doc);

mtsd_res mtsd_to_text(/* In */ mtsd_document* doc,
        /* Out */ uint8_t** out,
        /* Out */ size_t* size);

size_t mtsd_doc_records_count(const mtsd_document* doc);
size_t mtsd_doc_record_fields_count(const mtsd_record* record);
int mtsd_doc_field_is_multiline_value(const mtsd_field* field);
void mtsd_doc_init(mtsd_document* doc);
void mtsd_doc_record_init(mtsd_record* record);
void mtsd_doc_field_init(mtsd_field* field);
void mtsd_doc_free(mtsd_document* doc);
uint8_t mtsd_doc_get_key_id(const uint8_t* str, size_t str_size);
char* mtsd_doc_keyid_to_string(uint8_t keyid);
int mtsd_doc_is_valid_keyid(uint8_t keyid);

int mtsd_container_is_valid(uint8_t* data, size_t size);
uint64_t mtsd_container_get_date(uint8_t* data);

#endif

```

A.12. Файл private.h

```

#include "mtsdata.h"

#define MTSD_RANDOM_BYTES      3
#define MTSD_SALT_SIZE         16
#define MTSD_PAYLOAD_MAX_SIZE  0xFFFF
#define MTSD_LZMA_IC           0
#define MTSD_LZMA_LP           0
#define MTSD_LZMA_PB           0
#define MTSD_ARGON2_ITER       3
#define MTSD_ARGON2_MEM        (1 << 16)
#define MTSD_ARGON2_THREADS    2

#define MTSD_CHECK(result)      \
    if ((result) != MTSD_OK) {  \
        return MTSD_ERR;       \
    }

#define MTSD_CHECK_GOTO(label, result) \
    if ((result) != MTSD_OK) {         \
        goto label;                   \
    }

#define MTSD_MALLOC(var, size) \
    (var) = mtsd_malloc((size)); \
    if (!(var)) {                 \
        mtsd_error(MTSD_ESELF, MTSD_EMEMORY, NULL); \
        return MTSD_ERR;         \
    }

```

```

#define MTSD_REALLOC(var, size) \
    (var) = mtsd_realloc((var), (size)); \
    if (!(var)) { \
        mtsd_error(MTSD_ESELF, MTSD_EMEMORY, NULL); \
        return MTSD_ERR; \
    }

#define MTSD_FREE(var) \
    mtsd_free((var)); \
    (var) = NULL;

mtsd_res mtsd_encode(/* In */ mtsd_document* doc,
                    /* Out */ uint8_t** out,
                    /* Out */ size_t* size);

mtsd_res mtsd_decode(/* In */ uint8_t* data,
                    /* In */ size_t size,
                    /* In Out */ mtsd_document* doc);

mtsd_res mtsd_compress_payload(/* In Out */ uint8_t* data,
                               /* In Out */ size_t* size,
                               /* Out */ uint8_t* is_compressed);

mtsd_res mtsd_decompress_payload(/* In */ uint8_t* compressed,
                                 /* In */ size_t compressed_size,
                                 /* Out */ uint8_t* data,
                                 /* In Out */ size_t* size);

mtsd_res mtsd_encrypt_payload(/* In Out */ uint8_t* data,
                              /* In */ size_t data_size,
                              /* In */ uint8_t* pwd,
                              /* In */ size_t pwd_size,
                              /* In */ uint8_t* salt);

mtsd_res mtsd_decrypt_payload(/* In Out */ uint8_t* data,
                              /* In */ size_t data_size,
                              /* In */ uint8_t* pwd,
                              /* In */ size_t pwd_size,
                              /* In */ uint8_t* salt);

mtsd_res mtsd_random_bytes(uint8_t* out, size_t size);

```

A.13. Файл parser.c

```

#include "parser.h"

#include "../private.h"

#include <string.h>

#define TOKEN(state) ((state)->lexer)

static mtsd_res parser_unexpected();
static mtsd_res parse_field(mtsd_parser* state, mtsd_field* field);
static mtsd_res parse_record(mtsd_parser* state, mtsd_record* record);
static mtsd_res parse_doc(mtsd_parser* state, mtsd_document* doc);

```

```

static mtsd_res parser_unexpected()
{
    // TODO err_msg
    return MTSD_ERR;
}

static mtsd_res parse_field(mtsd_parser* state, mtsd_field* field)
{
    if (TOKEN(state).kind != MTSD_KEY_TOKEN) {
        return parser_unexpected();
    } else {
        state->lexer.consumed = 1;
    }

    field->key = mtsd_doc_get_key_id(state->lexer.buffer, state->lexer.buffer_size);
    if (!field->key) {
        mtsd_error(MTSD_ESELF, MTSD_EPARSE_UNKNOWN_KEY, "unknown key");
        return MTSD_ERR;
    }

    int is_multiline = 0;
    for (;;) {
        MTSD_CHECK(mtsd_parser_lexer_next(state));

        if (TOKEN(state).kind == MTSD_VALUE_TOKEN && !is_multiline) {
            field->value = state->lexer.buffer;
            field->value_size = state->lexer.buffer_size;
            state->lexer.buffer = NULL;
            state->lexer.consumed = 1;
            return MTSD_OK;
        } else if (TOKEN(state).kind == MTSD_VALUE_MULTILINE_TOKEN) {
            field->value_size += (state->lexer.buffer_size + (is_multiline == 0 ? 0
: 1));
            MTSD_REALLOC(field->value, field->value_size);
            if (is_multiline) {
                field->value[field->value_size - state->lexer.buffer_size - 1] = '\n';
            }
            memcpy(field->value + (field->value_size - state->lexer.buffer_size),
state->lexer.buffer, state->lexer.buffer_size);
            state->lexer.consumed = 1;
            is_multiline = 1;
        } else {
            return MTSD_OK;
        }
    }
}

static mtsd_res parse_record(mtsd_parser* state, mtsd_record* record)
{
    mtsd_field* current = record->fields;
    if (!current) {
        MTSD_MALLOC(current, sizeof(mtsd_field));
        record->fields = current;
    } else {
        while (current->next) {
            current = current->next;
        }
        MTSD_MALLOC(current->next, sizeof(mtsd_field));
        current = current->next;
    }
    mtsd_doc_field_init(current);

    MTSD_CHECK(parse_field(state, current));
}

```

```

    return MTSD_OK;
}

static mtsd_res parse_doc(mtsd_parser* state, mtsd_document* doc)
{
    mtsd_record* current = NULL;
    MTSD_MALLOC(current, sizeof(mtsd_field));
    doc->records = current;
    mtsd_doc_record_init(current);

    for (;;) {
        MTSD_CHECK(mtsd_parser_lexer_next(state));
        if (TOKEN(state).kind == MTSD_STREAM_END_TOKEN)
            return MTSD_OK;

        if (TOKEN(state).kind == MTSD_RECORD_SEPARATOR_TOKEN) {
            MTSD_MALLOC(current->next, sizeof(mtsd_field));
            current = current->next;
            mtsd_doc_record_init(current);
            state->lexer.consumed = 1;
        } else {
            MTSD_CHECK(parse_record(state, current));
        }
    }
}

mtsd_res mtsd_parse(mtsd_read_callback read_callback, void* callback_data,
mtsd_document* doc)
{
    mtsd_parser state = {
        .line = 1,
        .column = 0,
        .offset = -1,

        .input = {
            .callback = read_callback,
            .data = callback_data
        },

        .reader = {
            .mb_char = {0, 0, 0, 0},
            .mb_size = 0,
            .eof = 0,
        },

        .lexer = {
            .kind = MTSD_STREAM_START_TOKEN,
            .start = 0,
            .end = 0,
            .buffer = NULL,
            .buffer_size = 0,
            .consumed = 1
        }
    };

    mtsd_doc_init(doc);
    if (!parse_doc(&state, doc)) {
        mtsd_doc_free(doc);
        MTSD_FREE(state.lexer.buffer);
        return MTSD_ERR;
    } else {
        MTSD_FREE(state.lexer.buffer);
        return MTSD_OK;
    }
}

```

```

// for (;;) {
//     MTSD_CHECK(mtsd_parser_lexer_next(&state));
//     if (TOKEN(&state).kind == MTSD_STREAM_END_TOKEN) {
//         break;
//     } else {
//         mtsd_parser_print_token(&state);
//         state.lexer.consumed = 1;
//     }
// }
}

```

A.14. Файл parser.h

```

#ifndef MTSD_LANG_PARSER_H
#define MTSD_LANG_PARSER_H

#include "mtsdata.h"

#include <stddef.h>
#include <stdint.h>

typedef enum {
    MTSD_STREAM_START_TOKEN,
    MTSD_STREAM_END_TOKEN,
    MTSD_RECORD_SEPARATOR_TOKEN,
    MTSD_KEY_TOKEN,
    MTSD_VALUE_TOKEN,
    MTSD_VALUE_MULTILINE_TOKEN,
} mtsd_token_kind;

typedef struct {
    size_t line;
    size_t column;
    size_t offset;

    struct {
        mtsd_read_callback callback;
        void* data;
    } input;

    struct {
        uint8_t mb_char[4];
        size_t mb_size;
        int eof;
    } reader;

    struct {
        mtsd_token_kind kind;
        size_t start;
        size_t end;
        uint8_t* buffer;
        size_t buffer_size;
        int consumed;
    } lexer;
} mtsd_parser;

mtsd_res mtsd_parser_input_next(mtsd_parser* state);
mtsd_res mtsd_parser_lexer_next(mtsd_parser* state);
void mtsd_parser_print_token(mtsd_parser* state);

```

```
#endif
```

A.15. Файл reader.c

```
#include "../private.h"
#include "parser.h"

#define UTF8_LEN(bytes) (((bytes)[0] & 0x80) == 0x00 ? 1 : \
\
((bytes)[0] & 0xE0) == 0xC0 ? 2 : \
\
((bytes)[0] & 0xF0) == 0xE0 ? 3 : \
\
((bytes)[0] & 0xF8) == 0xF0 ? 4 : 0)

static mtspd_res input_read_utf8(mtsd_parser* state)
{
    size_t read;
    uint8_t* buff = state->reader.mb_char;
    if (!state->input.callback(state->input.data, buff, 1, &read)) {
        mtsd_error(MTSD_ESELF, MTSD_EREADER, NULL);
        return MTSD_ERR;
    }

    if (read == 0) {
        state->reader.eof = 1;
    } else {
        // Decode a UTF-8 character

        size_t width = UTF8_LEN(buff);
        state->reader.mb_size = width;

        if (!width) {
            mtsd_error(MTSD_ESELF, MTSD_EREADER, "invalid leading UTF-8 octet");
            return MTSD_ERR;
        } else if (width > 1) {
            if (!state->input.callback(state->input.data, buff + 1, width - 1,
&read)) {
                mtsd_error(MTSD_ESELF, MTSD_EREADER, NULL);
                return MTSD_ERR;
            }
            if (read != (width - 1)) {
                mtsd_error(MTSD_ESELF, MTSD_EREADER, "incomplete UTF-8 octet
sequence");
                return MTSD_ERR;
            }
        }

        uint8_t octet = buff[0];
        uint32_t value = (octet & 0x80) == 0x00 ? octet & 0x7F :
                        (octet & 0xE0) == 0xC0 ? octet & 0x1F :
                        (octet & 0xF0) == 0xE0 ? octet & 0x0F :
                        (octet & 0xF8) == 0xF0 ? octet & 0x07 : 0;

        for (size_t k = 1; k < width; k++) {
            octet = buff[k];
            if ((octet & 0xC0) != 0x80) {
                mtsd_error(MTSD_ESELF, MTSD_EREADER, "invalid trailing UTF-8 octet");
                return MTSD_ERR;
            }
        }
    }
}
```

```

    } else {
        value = (value << 6) + (octet & 0x3F);
    }
}

if (!(
    (width == 1) ||
    (width == 2 && value >= 0x80) ||
    (width == 3 && value >= 0x800) ||
    (width == 4 && value >= 0x10000)
)) {
    mtsd_error(MTSD_ESELF, MTSD_EREADER, "invalid length of a UTF-8
sequence");
    return MTSD_ERR;
}

if ((value >= 0xD800 && value <= 0xDFFF) || value > 0x10FFFF) {
    mtsd_error(MTSD_ESELF, MTSD_EREADER, "invalid Unicode character");
    return MTSD_ERR;
}
}

return MTSD_OK;
}

mtsd_res mtsd_parser_input_next(mtsd_parser* state)
{
    MTSD_CHECK(input_read_utf8(state));
    state->offset += 1;
    if (state->reader.mb_char[0] == '\n') {
        state->line += 1;
        state->column = 0;
    } else if (!state->reader.eof) {
        state->column += 1;
    }
    return MTSD_OK;
}

```

A.16. Файл `lexer.c`

```

#include "../private.h"
#include "parser.h"

#include <string.h>

#define CURRENT(state) ((state)->reader.mb_char[0])

static mtsd_res lexer_add(mtsd_parser* state);
static void lexer_clear(mtsd_parser* state);
static mtsd_res lexer_read_key(mtsd_parser* state);
static mtsd_res lexer_read_value(mtsd_parser* state);
static void lexer_trim(mtsd_parser* state, size_t max_start, size_t max_end);

static mtsd_res lexer_add(mtsd_parser* state)
{
    state->lexer.end += 1;
    size_t len = state->reader.mb_size;
    MTSD_REALLOC(state->lexer.buffer, state->lexer.buffer_size + len);
    memcpy(state->lexer.buffer + state->lexer.buffer_size, state-
>reader.mb_char, len);
}

```

```

    state->lexer.buffer_size += len;
    return MTSD_OK;
}

static void lexer_clear(mtsd_parser* state)
{
    state->lexer.start = state->offset;
    state->lexer.end = state->offset;
    state->lexer.buffer_size = 0;
    MTSD_FREE(state->lexer.buffer);
}

static mtsd_res lexer_read_key(mtsd_parser* state)
{
    for (;;) {
        if (CURRENT(state) == ':' || CURRENT(state) == '\\n') {
            return MTSD_OK;
        } else {
            MTSD_CHECK(lexer_add(state));
        }

        MTSD_CHECK(mtsd_parser_input_next(state));
        if (state->reader.eof)
            return MTSD_OK;
    }
}

static mtsd_res lexer_read_value(mtsd_parser* state)
{
    for (;;) {
        if (CURRENT(state) == '\\n') {
            return MTSD_OK;
        } else {
            MTSD_CHECK(lexer_add(state));
        }

        MTSD_CHECK(mtsd_parser_input_next(state));
        if (state->reader.eof)
            return MTSD_OK;
    }
}

static void lexer_trim(mtsd_parser* state, size_t max_start, size_t max_end)
{
    if (max_start > 0) {
        size_t dt = 0;
        for (size_t i = 0; i < state->lexer.buffer_size && i < max_start; i += 1)
        {
            if (state->lexer.buffer[i] == ' ') {
                dt += 1;
            } else {
                break;
            }
        }
        memmove(state->lexer.buffer, state->lexer.buffer + dt, state->lexer.buffer_size - dt);
        state->lexer.start += dt;
        state->lexer.buffer_size -= dt;
    }
    if (max_end > 0) {
        size_t dt = 0;
        for (size_t i = 0; i < state->lexer.buffer_size && i < max_end; i += 1) {
            if (
                state->lexer.buffer[state->lexer.buffer_size - i - 1] == ' ' ||

```

```

        state->lexer.buffer[state->lexer.buffer_size - i - 1] == '\r'
    ) {
        dt += 1;
    } else {
        break;
    }
}
state->lexer.end -= dt;
state->lexer.buffer_size -= dt;
}
}

mtsd_res mtsd_parser_lexer_next(mtsd_parser* state)
{
    if (!state->lexer.consumed) {
        return MTSD_OK;
    } else {
        state->lexer.consumed = 0;
    }

    for (;;) {
        MTSD_CHECK(mtsd_parser_input_next(state));
        if (state->reader.eof)
            break;

        lexer_clear(state);
        if (state->column == 1) {
            MTSD_CHECK(lexer_read_key(state));

            if (
                state->lexer.buffer_size >= 2 &&
                state->lexer.buffer[0] == ' ' &&
                state->lexer.buffer[1] == ' '
            ) {
                state->lexer.kind = MTSD_VALUE_MULTILINE_TOKEN;
                if (CURRENT(state) == ':') {
                    MTSD_CHECK(lexer_read_value(state));
                }
                lexer_trim(state, 2, -1);
                return MTSD_OK;
            }

            lexer_trim(state, -1, -1);

            if (
                state->lexer.buffer_size == 3 &&
                state->lexer.buffer[0] == '-' &&
                state->lexer.buffer[1] == '-' &&
                state->lexer.buffer[2] == '-'
            ) {
                state->lexer.kind = MTSD_RECORD_SEPARATOR_TOKEN;
                return MTSD_OK;
            }

            if (state->lexer.buffer_size > 0) {
                state->lexer.kind = MTSD_KEY_TOKEN;
                return MTSD_OK;
            }
        } else {
            MTSD_CHECK(lexer_read_value(state));

            lexer_trim(state, -1, -1);

            if (state->lexer.buffer_size > 0) {

```

```

        state->lexer.kind = MTSD_VALUE_TOKEN;
        return MTSD_OK;
    }
}

state->lexer.kind = MTSD_STREAM_END_TOKEN;
state->lexer.start = state->offset;
state->lexer.end = state->offset;
return MTSD_OK;
}

```

A.17. Файл compression.c

```

#include "private.h"

#include <LzmaDec.h>
#include <LzmaEnc.h>

static void* SzAlloc(ISzAllocPtr p, size_t size)
{
    return mttd_malloc(size);
}

static void SzFree(ISzAllocPtr p, void* address)
{
    mttd_free(address);
}

static ISzAlloc g_Alloc = { SzAlloc, SzFree };

mttd_res mttd_compress_payload(uint8_t* data, size_t* size, uint8_t*
is_compressed)
{
    size_t data_size = *size;
    CLzmaEncProps props;
    LzmaEncProps_Init(&props);
    props.lc = MTSD_LZMA_LC;
    props.lp = MTSD_LZMA_LP;
    props.pb = MTSD_LZMA_PB;

    props.level = 9;
    props.dictSize = data_size;

    uint8_t props_encoded[LZMA_PROPS_SIZE];
    size_t props_encoded_size = LZMA_PROPS_SIZE;

    SRes res = LzmaEncode(
        data, size,
        data, data_size,
        &props, props_encoded, &props_encoded_size, 0, NULL, &g_Alloc, &g_Alloc);

    if (res != SZ_OK) {
        if (res == SZ_ERROR_OUTPUT_EOF) {
            *is_compressed = 0;
            return MTSD_OK;
        } else {
            mttd_error(MTSD_ELZMA, res, NULL);
            return MTSD_ERR;
        }
    }

    *is_compressed = 1;
}

```

```

    return MTSD_OK;
}

mtsd_res mtsd_decompress_payload(uint8_t* compressed, size_t compressed_size,
uint8_t* data, size_t* size)
{
    CLzmaEncProps props;
    props.lc = MTSD_LZMA_LC;
    props.lp = MTSD_LZMA_LP;
    props.pb = MTSD_LZMA_PB;
    props.dictSize = *size;

    uint8_t props_encoded[LZMA_PROPS_SIZE];
    props_encoded[0] = (uint8_t)((props.pb * 5 + props.lp) * 9 + props.lc);
    *(uint32_t*)(props_encoded + 1) = props.dictSize;

    ELzmaStatus status;
    SRes res = LzmaDecode(data, size, compressed, &compressed_size,
                          props_encoded, LZMA_PROPS_SIZE, LZMA_FINISH_ANY,
&status, &g_Alloc);

    if (res != SZ_OK && status != LZMA_STATUS_NEEDS_MORE_INPUT) {
        mtsd_error(MTSD_ELZMA, res, NULL);
        return MTSD_ERR;
    }
    return MTSD_OK;
}

```

A.18. Файл encoding.c

```

#include "lang/parser.h"
#include "mtsdata.h"
#include "private.h"

#include <string.h>

#define RECORD_SIZE      1
#define KEY_SIZE          1
#define STR_TERMINATOR_SIZE 1
#define STR_TERMINATOR  '\0'

mtsd_res mtsd_encode__(mtsd_document* doc, uint8_t** out, size_t* size)
{
#define buf (*out)
    size_t written = 0;
    buf = NULL;

    mtsd_record* record = doc->records;
    while (record) {
        size_t fields_num = mtsd_doc_record_fields_count(record);
        if (fields_num > 0xFF) {
            mtsd_error(MTSD_ESELF, MTSD_EENCODE_RECORD_SIZE, "maximum 255 fields per
record");
            return MTSD_ERR;
        }

        if (fields_num) {
            MTSD_REALLOC(buf, written + RECORD_SIZE);
            buf[written] = fields_num;
            written += RECORD_SIZE;

```

```

    }
    mtsd_field* field = record->fields;
    while (field) {
        MTSD_REALLOC(buf, written + KEY_SIZE + field->value_size +
STR_TERMINATOR_SIZE);
        buf[written] = field->key;
        written += KEY_SIZE;
        memcpy(buf + written, field->value, field->value_size);
        written += field->value_size;
        buf[written] = STR_TERMINATOR;
        written += STR_TERMINATOR_SIZE;

        field = field->next;
    }

    record = record->next;
}
#undef buf

*size = written;
return MTSD_OK;
}

mtsd_res mtsd_encode(mtsd_document* doc, uint8_t** out, size_t* size)
{
    *out = NULL;
    if (!mtsd_encode__(doc, out, size)) {
        MTSD_FREE(*out);
        return MTSD_ERR;
    }
    return MTSD_OK;
}

static inline mtsd_res mtsd_decode__(uint8_t* data, size_t size, mtsd_document*
doc)
{
    for (size_t i = 0; i < size; i++) {
        mtsd_record* record = doc->records;
        if (!record) {
            MTSD_MALLOC(record, sizeof(mtsd_record));
            doc->records = record;
        } else {
            while (record->next) {
                record = record->next;
            }
            MTSD_MALLOC(record->next, sizeof(mtsd_record));
            record = record->next;
        }
        mtsd_doc_record_init(record);

        uint8_t fields_num = data[i];
        i += RECORD_SIZE;

        for (uint8_t field_i = 0; field_i < fields_num; field_i += 1, i += 1) {
            if (i >= size) {
                mtsd_error(MTSD_ESELF, MTSD_EDECODE_CORRUPTED_PAYLOAD, NULL);
                return MTSD_ERR;
            }

            mtsd_field* field = record->fields;
            if (!field) {
                MTSD_MALLOC(field, sizeof(mtsd_field));
                record->fields = field;
            } else {

```

```

        while (field->next) {
            field = field->next;
        }
        MTSD_MALLOC(field->next, sizeof(mtsd_field));
        field = field->next;
    }
    mtsd_doc_field_init(field);

    field->key = data[i];
    i += KEY_SIZE;
    if (!mtsd_doc_is_valid_keyid(field->key)) {
        mtsd_error(MTSD_ESELF, MTSD_EDECODE_CORRUPTED_PAYLOAD, "invalid
keyid");
        return MTSD_ERR;
    }

    for (; i < size; i += 1) {
        if (data[i] == STR_TERMINATOR) {
            if (field->value_size) {
                MTSD_MALLOC(field->value, field->value_size);
                memcpy(field->value, data + i - field->value_size, field-
>value_size);
            }
            break;
        } else {
            field->value_size += 1;
        }
    }
    if (i >= size) {
        mtsd_error(MTSD_ESELF, MTSD_EDECODE_CORRUPTED_PAYLOAD, NULL);
        return MTSD_ERR;
    }
}

return MTSD_OK;
}

mtsd_res mtsd_decode(uint8_t* data, size_t size, mtsd_document* doc)
{
    mtsd_doc_init(doc);
    if (!mtsd_decode__(data, size, doc)) {
        mtsd_doc_free(doc);
        return MTSD_ERR;
    }
    return MTSD_OK;
}

static inline mtsd_res mtsd_to_text__(mtsd_document* doc, uint8_t** out,
size_t* size)
{
#define buf (*out)
    size_t written = 0;
    buf = NULL;

    mtsd_record* record = doc->records;
    while (record) {
        mtsd_field* field = record->fields;
        while (field) {
            char* key = mtsd_doc_keyid_to_string(field->key);
            size_t key_len = strlen(key);
            MTSD_REALLOC(buf, written + key_len + 2);
            memcpy(buf + written, key, key_len);
            written += key_len;
        }
        record = record->next;
    }
}

```

```

int is_multiline = mtzd_doc_field_is_multiline_value(field);

buf[written] = ':';
written += 1;
buf[written] = (is_multiline) ? '\n' : ' ';
written += 1;

if (is_multiline) {
    size_t value_offset = 0;
    size_t value_length = 0;
    for (;;) {
        while (value_offset < field->value_size) {
            value_offset += 1;
            value_length += 1;
            if (field->value[value_offset] == '\n') {
                value_offset += 1;
                value_length += 1;
                break;
            }
        }

        MTSD_REALLOC(buf, written + value_length + 2);
        buf[written] = ' ', written += 1;
        buf[written] = ' ', written += 1;
        memcpy(buf + written, field->value + value_offset - value_length,
value_length);
        written += value_length;

        if (value_offset >= field->value_size) {
            break;
        } else {
            value_length = 0;
        }
    }
} else {
    MTSD_REALLOC(buf, written + field->value_size);
    memcpy(buf + written, field->value, field->value_size);
    written += field->value_size;
}

MTSD_REALLOC(buf, written + 1);
buf[written] = '\n';
written += 1;

field = field->next;
}

if (record->next) {
    MTSD_REALLOC(buf, written + 4);
    buf[written + 0] = '-';
    buf[written + 1] = '-';
    buf[written + 2] = '-';
    buf[written + 3] = '\n';
    written += 4;
}

record = record->next;
}
#undef buf

*size = written;
return MTSD_OK;
}

```

```

mtsd_res mtsd_to_text(mtsd_document* doc, uint8_t** out, size_t* size)
{
    *out = NULL;
    if (!mtsd_to_text__(doc, out, size)) {
        MTSD_FREE(*out);
        return MTSD_ERR;
    }
    return MTSD_OK;
}

```

A.19. Файл encryption.c

```

#include "private.h"

#include <aes.h>
#include <argon2.h>
#include <randombytes.h>

static mtsd_res derive_bytes(uint8_t* salt, uint8_t* pwd, size_t pwd_size,
uint8_t* out);

mtsd_res mtsd_encrypt_payload(uint8_t* data, size_t data_size,
uint8_t* pwd, size_t pwd_size, uint8_t* salt)
{
    uint8_t derived_bytes[AES_KEYLEN + AES_BLOCKLEN];
    MTSD_CHECK(derive_bytes(salt, pwd, pwd_size, derived_bytes));
    uint8_t* key = derived_bytes;
    uint8_t* iv = derived_bytes + AES_KEYLEN;

    struct AES_ctx ctx;
    AES_init_ctx_iv(&ctx, key, iv);
    AES_CTR_xcrypt_buffer(&ctx, data, data_size);
    return MTSD_OK;
}

mtsd_res mtsd_decrypt_payload(uint8_t* data, size_t data_size,
uint8_t* pwd, size_t pwd_size, uint8_t* salt)
{
    uint8_t derived_bytes[AES_KEYLEN + AES_BLOCKLEN];
    MTSD_CHECK(derive_bytes(salt, pwd, pwd_size, derived_bytes));
    uint8_t* key = derived_bytes;
    uint8_t* iv = derived_bytes + AES_KEYLEN;

    struct AES_ctx ctx;
    AES_init_ctx_iv(&ctx, key, iv);
    AES_CTR_xcrypt_buffer(&ctx, data, data_size);
    return MTSD_OK;
}

static int argon2_alloc(uint8_t** memory, size_t bytes_to_allocate)
{
    *memory = mtsd_malloc(bytes_to_allocate);
    return 0;
}

static void argon2_free(uint8_t* memory, size_t bytes_to_allocate)
{
    mtsd_free(memory);
}

```

```

static mtsd_res derive_bytes(uint8_t* salt, uint8_t* pwd, size_t pwd_size,
uint8_t* out)
{
    struct Argon2_Context ctx = {
        .out = out, .outlen = (AES_KEYLEN + AES_BLOCKLEN),
        .pwd = pwd, .pwdlen = pwd_size,
        .salt = salt, .saltlen = MTSD_SALT_SIZE,

        .t_cost = MTSD_ARGON2_ITER,
        .m_cost = MTSD_ARGON2_MEM,
        .lanes = MTSD_ARGON2_THREADS,
        .threads = MTSD_ARGON2_THREADS,

        .secret = NULL, .secretlen = 0,
        .ad = NULL, .adlen = 0,
        .version = ARGON2_VERSION_13,
        .allocate_cbk = argon2_alloc, .free_cbk = argon2_free,
        .flags = ARGON2_DEFAULT_FLAGS
    };

    int err = argon2id_ctx(&ctx);
    if (err != ARGON2_OK) {
        mtsd_error(MTSD_EARGON2, err, NULL);
        return MTSD_ERR;
    }
    return MTSD_OK;
}

mtsd_res mtsd_random_bytes(uint8_t* out, size_t size)
{
    int err = randombytes(out, size);
    if (err != 0) {
        mtsd_error(MTSD_ERANDOMBYTES, err, NULL);
        return MTSD_ERR;
    }
    return MTSD_OK;
}

```

A.20. Файл document.c

```

#include "private.h"

static char* MTSD_KEYS[] = {
    /* 000 */ "__NULL",
    /* 001 */ "email",
    /* 002 */ "password",
    /* 003 */ "login",
    /* ... */ "",
    /* 254 */ "",
    /* 255 */ "__RESERVED"
};

size_t mtsd_doc_records_count(const mtsd_document* doc)
{
    size_t records_num = 0;
    mtsd_record* record = doc->records;
    while (record) {
        records_num += 1;
        record = record->next;
    }
}

```

```

    return records_num;
}

size_t mtsd_doc_record_fields_count(const mtsd_record* record)
{
    size_t fields_num = 0;
    mtsd_field* field = record->fields;
    while (field) {
        fields_num += 1;
        field = field->next;
    }
    return fields_num;
}

int mtsd_doc_field_is_multiline_value(const mtsd_field* field)
{
    for (size_t i = 0; i < field->value_size; i += 1) {
        if (field->value[i] == '\n') {
            return 1;
        }
    }
    return 0;
}

void mtsd_doc_init(mtsd_document* doc)
{
    doc->records = NULL;
}

void mtsd_doc_record_init(mtsd_record* record)
{
    record->fields = NULL;
    record->next = NULL;
}

void mtsd_doc_field_init(mtsd_field* field)
{
    field->key = 0;
    field->value_size = 0;
    field->value = NULL;
    field->next = NULL;
}

void mtsd_doc_free(mtsd_document* doc)
{
    mtsd_record* record = doc->records;
    while (record) {
        mtsd_field* field = record->fields;
        while (field) {
            if (field->value) {
                MTSD_FREE(field->value);
            }

            mtsd_field* next = field->next;
            MTSD_FREE(field);
            field = next;
        }

        mtsd_record* next = record->next;
        MTSD_FREE(record);
        record = next;
    }

    mtsd_doc_init(doc);
}

```

```

}

uint8_t mtsd_doc_get_key_id(const uint8_t* str, size_t str_size)
{
    if (!str)
        return 0;

    for (uint8_t keyid = 1; keyid < 0xFF; keyid += 1) {
        char* key = MTSD_KEYS[keyid];
        for (size_t i = 0; /* */; i += 1) {
            if (key[i] == '\\0') {
                if (str_size == i) {
                    return keyid;
                } else {
                    break;
                }
            }

            if (i >= str_size || key[i] != str[i])
                break;
        }
    }

    return 0;
}

int mtsd_doc_is_valid_keyid(uint8_t keyid)
{
    if (keyid == 0 || keyid == 0xFF || MTSD_KEYS[keyid][0] == '\\0') {
        return 0;
    }
    return 1;
}

char* mtsd_doc_keyid_to_string(uint8_t keyid)
{
    if (!mtsd_doc_is_valid_keyid(keyid)) {
        return "__NULL__";
    }
    return MTSD_KEYS[keyid];
}

```

A.21. Файл container.c

```

#include "mtsdata.h"
#include "private.h"

#include <stddef.h>
#include <string.h>
#include <time.h>

#define MTSD_CRC_START      2
#define MTSD_DATE_SIZE      3
#define MTSD_DATE_RESOLUTION 3600

#pragma pack(push, 1)
typedef struct {
    uint16_t crc16;
    union {
        uint8_t is_compressed;
    }
}

```

```

    uint8_t date[MTSD_DATE_SIZE];
};
uint8_t random_bytes[MTSD_RANDOM_BYTES];
} mtzd_header;
#pragma pack(pop)

static uint16_t crc16(uint8_t* data, size_t size);
static mtzd_res store_date_now(mtzd_header* header);
static int is_compressed_payload(mtzd_header* header);
static void set_is_compressed_payload(mtzd_header* header, int is_compressed);
static void derive_salt(const uint8_t* data, size_t data_size,
                       const uint8_t* pass, size_t pass_len, uint8_t* salt);

mtzd_res mtzd_encrypt(mtzd_document* doc, uint8_t* password, size_t
password_len,
                    uint8_t** encrypted, size_t* size)
{
    uint8_t* out = NULL;
    uint8_t* encoded = NULL;

    MTSD_MALLOC(out, sizeof(mtzd_header) + MTSD_PAYLOAD_MAX_SIZE);

    size_t encoded_size = 0;
    MTSD_CHECK_GOTO(error,
        mtzd_encode(doc, &encoded, &encoded_size));
    if (encoded_size > MTSD_PAYLOAD_MAX_SIZE) {
        mtzd_error(MTSD_ESELF, MTSD_EENCODE_PAYLOAD_SIZE, NULL);
        goto error;
    }

    uint8_t is_compressed;
    memcpy(out + sizeof(mtzd_header), encoded, encoded_size);
    *size = encoded_size;
    MTSD_CHECK_GOTO(error,
        mtzd_compress_payload(out + sizeof(mtzd_header), size, &is_compressed));
    set_is_compressed_payload((mtzd_header*)out, is_compressed);
    if (!is_compressed) {
        memcpy(out + sizeof(mtzd_header), encoded, encoded_size);
        *size = encoded_size;
    }
    MTSD_FREE(encoded);

    MTSD_CHECK_GOTO(error,
        store_date_now((mtzd_header*)out));
    MTSD_CHECK_GOTO(error,
        mtzd_random_bytes(((mtzd_header*)out)->random_bytes, MTSD_RANDOM_BYTES));

    uint8_t salt[MTSD_SALT_SIZE];
    derive_salt(out, (sizeof(mtzd_header) + *size), password, password_len,
salt);
    // MTSD_CHECK_GOTO(error,
    //     mtzd_encrypt_payload(out + sizeof(mtzd_header), *size, password,
password_len, salt));

    ((mtzd_header*)out)->crc16 = crc16(out + MTSD_CRC_START, sizeof(mtzd_header)
+ *size - MTSD_CRC_START);

    *encrypted = (uint8_t*)out;
    *size = (sizeof(mtzd_header) + *size);
    return MTSD_OK;

error:
    MTSD_FREE(out);
    MTSD_FREE(encoded);
}

```

```

    return MTSD_ERR;
}

mtsd_res mtsd_decrypt(uint8_t* encrypted, size_t size,
                     uint8_t* password, size_t password_len, mtsd_document*
doc)
{
    uint8_t* cloned = NULL;
    uint8_t* encoded = NULL;

    MTSD_MALLOC(cloned, size - sizeof(mtsd_header));
    memcpy(cloned, encrypted + sizeof(mtsd_header), size - sizeof(mtsd_header));

    uint8_t salt[MTSD_SALT_SIZE];
    derive_salt(encrypted, size, password, password_len, salt);
    MTSD_CHECK_GOTO(error,
        mtsd_decrypt_payload(cloned, size - sizeof(mtsd_header), password,
password_len, salt));

    if (is_compressed_payload((mtsd_header*)encrypted)) {
        encoded = mtsd_malloc(MTSD_PAYLOAD_MAX_SIZE);
        if (!encoded) {
            mtsd_error(MTSD_ESELF, MTSD_EMEMORY, NULL);
            goto error;
        }
        size_t encoded_size = MTSD_PAYLOAD_MAX_SIZE;
        MTSD_CHECK_GOTO(error,
            mtsd_decompress_payload(cloned, size - sizeof(mtsd_header), encoded,
&encoded_size));
        MTSD_FREE(cloned);

        MTSD_CHECK_GOTO(error,
            mtsd_decode(encoded, encoded_size, doc));
        MTSD_FREE(encoded);
    } else {
        MTSD_CHECK_GOTO(error,
            mtsd_decode(cloned, size - sizeof(mtsd_header), doc));
    }

    return MTSD_OK;

error:
    MTSD_FREE(cloned);
    MTSD_FREE(encoded);
    return MTSD_ERR;
}

/*
Name   : CRC-16/CCITT-FALSE
Poly   : 0x1021
Init   : 0xFFFF
RefIn  : false
RefOut : false
XorOut : 0x0000
Check  : 0x29B1
*/
static uint16_t crc16(uint8_t* data, size_t size)
{
    uint16_t crc = 0xFFFF;

    while (size--) {
        crc ^= *(data++) << 8;

        for (unsigned i = 0; i < 8; i += 1)

```

```

        crc = (crc & 0x8000) ? (crc << 1) ^ 0x1021 : crc << 1;
    }

    return crc;
}

static mtsd_res store_date_now(mtsd_header* header)
{
    time_t t = time(NULL);
    if (t == (time_t)-1) {
        mtsd_error(MTSD_ESELF, MTSD_ETIME, "cannot get system time");
        return MTSD_ERR;
    }
    uint64_t scaled = ((uint64_t)t / MTSD_DATE_RESOLUTION);

    int is_compressed = is_compressed_payload(header);
    header->date[2] = (scaled >> (8 * 0)) & 0xFF;
    header->date[1] = (scaled >> (8 * 1)) & 0xFF;
    header->date[0] = (scaled >> (8 * 2)) & 0xFF;
    set_is_compressed_payload(header, is_compressed);

    return MTSD_OK;
}

static void derive_salt(const uint8_t* data, size_t data_size,
                       const uint8_t* pass, size_t pass_len, uint8_t* salt)
{
    #ifndef DEBUG
        mtsd_header* header = (mtsd_header*)data;
        salt[0] = (data_size >> (8 * 0)) & 0xFF;
        salt[1] = (data_size >> (8 * 1)) & 0xFF;
        salt[2] = header->date[0];
        salt[3] = header->date[1];
        salt[4] = header->date[2];
        salt[5] = header->random_bytes[0];
        salt[6] = header->random_bytes[1];
        salt[7] = header->random_bytes[2];
        salt[8] = (pass_len >> (8 * 0)) & 0xFF;
        salt[9] = 0xc1;
        salt[10] = 0x12;
        salt[11] = 0x42;
        salt[12] = 0xf0;
        salt[13] = 0xe1;
        salt[14] = 0xeb;
        salt[15] = 0xa9;

        for (size_t pi = 0, si = 9; pi < pass_len && si < MTSD_SALT_SIZE;
             pi += 1, si += 1) {
            salt[si] ^= pass[pi];
        }
    #else
        salt[0] = 0x0; salt[1] = 0x1; salt[2] = 0x2; salt[3] = 0x3;
        salt[4] = 0x4; salt[5] = 0x5; salt[6] = 0x6; salt[7] = 0x7;
        salt[8] = 0xF; salt[9] = 0xE; salt[10] = 0xD; salt[11] = 0xC;
        salt[12] = 0xB; salt[13] = 0xA; salt[14] = 0x9; salt[15] = 0x8;
    #endif
}

int mtsd_container_is_valid(uint8_t* data, size_t size)
{
    if (size < sizeof(mtsd_header)) {
        return 0;
    }
}

```

```

uint16_t crc = ((mtsd_header*)data)->crc16;
uint16_t actual = crc16(data + MTSD_CRC_START, size - MTSD_CRC_START);

return (crc == actual);
}

uint64_t mtsd_container_get_date(uint8_t* data)
{
    uint64_t result = 0;
    mtsd_header* header = (mtsd_header*)data;
    result |= (header->date[0] & 0x7F) << (8 * 2);
    result |= (header->date[1]) << (8 * 1);
    result |= (header->date[2]) << (8 * 0);
    return result * MTSD_DATE_RESOLUTION;
}

static int is_compressed_payload(mtsd_header* header)
{
    return header->is_compressed & 0x80;
}

static void set_is_compressed_payload(mtsd_header* header, int is_compressed)
{
    header->is_compressed = (is_compressed)
        ? header->is_compressed | 0x80
        : header->is_compressed & 0x7F;
}

```

A.22. Файл CMakeLists.txt

```

add_subdirectory("../third_party/lzma"
"${PROJECT_BINARY_DIR}/third_party/lzma")
add_subdirectory("../third_party/randombytes"
"${PROJECT_BINARY_DIR}/third_party/randombytes")
add_subdirectory("../third_party/tiny-AES-c"
"${PROJECT_BINARY_DIR}/third_party/tiny-AES-c")
add_subdirectory("../third_party/argon2"
"${PROJECT_BINARY_DIR}/third_party/argon2")

add_library(mtsd STATIC)

target_sources(mtsd PRIVATE
    "encoding.c"
    "compression.c"
    "encryption.c"
    "container.c"
    "document.c"
    "lang/reader.c"
    "lang/lexer.c"
    "lang/parser.c"
    "lang/dumper.c"
)

target_link_libraries(mtsd PRIVATE
    lzma
    randombytes
    tiny-AES-c
    argon2
)

```

```
target_include_directories(mtsd PRIVATE "${PROJECT_BINARY_DIR}")  
target_include_directories(mtsd PUBLIC "../include")
```

ДОДАТОК Б
Слайди презентації



Рисунок Б.1 – Слайд 1

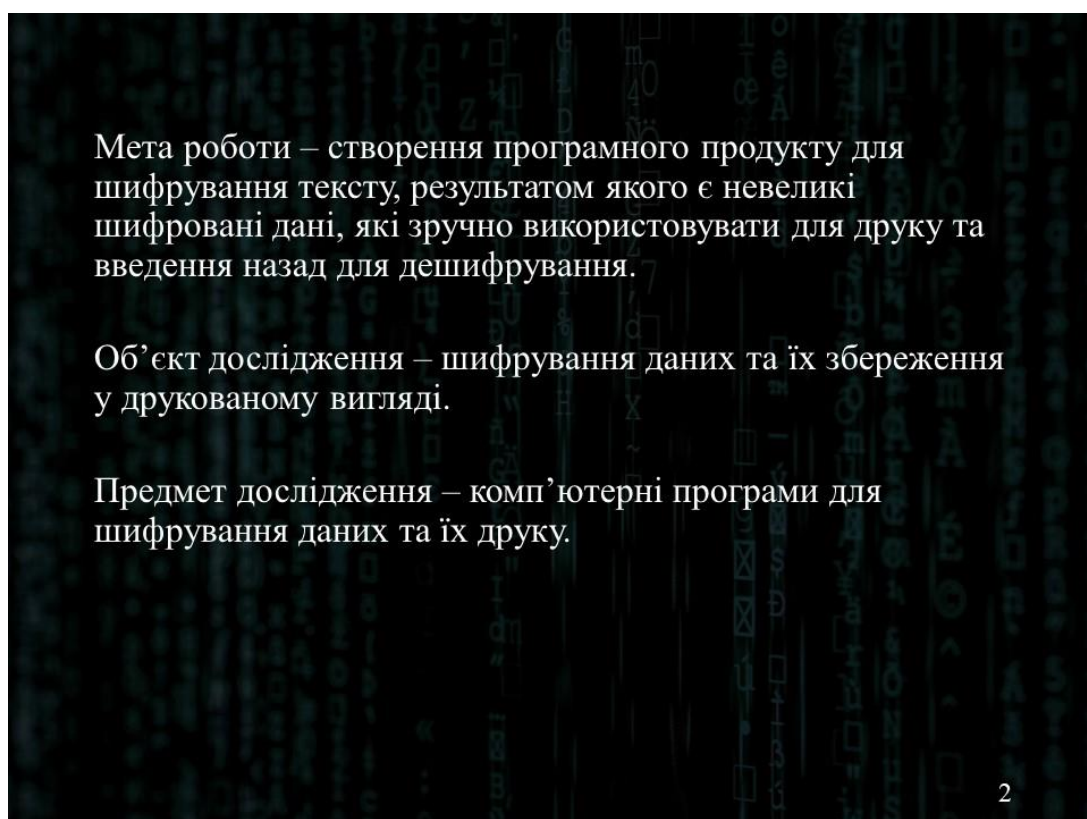


Рисунок Б.2 – Слайд 2

Завдання

- аналіз вимог до програмного забезпечення шифрування даних;
- аналіз способів збереження даних у друкованому вигляді;
- вибір засобів розробки;
- проектування програмного забезпечення;
- розроблення програмного забезпечення;
- тестування програмного забезпечення.

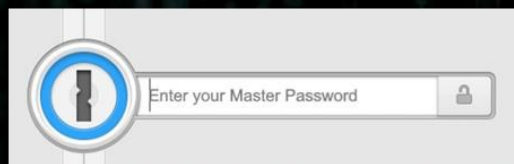
3

Рисунок Б.3 – Слайд 3

Аналіз предметної області, шифрування даних паролем

Впливають на розмір шифротексту:

- обраний режим блочного шифрування;
- обраний розмір сілі.



4

Рисунок Б.4 – Слайд 4

Аналіз предметної області, друк шифротексту

Способи представлення двійкових даних:

- двовимірний штрих-код;
- транспортне кодування.

Практичними є невеликі повідомлення.

Шляхи досягнення:

- попереднє стиснення;
- мінімізація метаданих.



Data Matrix Code



QR Code



Aztec Code

```
06 e0 0d 83 45 22
ba 2f d2 5f a8 77
6e 78 a1 2a ff 5a
```

5

Рисунок Б.5 – Слайд 5

Опис формату файлу

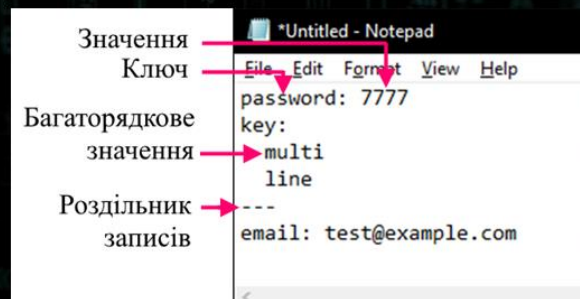
Розмір, біт	Опис
16	Контрольна сума
1	Стан стиснення
23	Дата та час
24	Випадкові числа

Заголовок: 8 байт.

- кількість пар: 1 байт;
- назва ключа: 1 байт;
- кінець рядку значення: 1 нульовий байт.

Мінімальна кількість метаданих у заголовку.

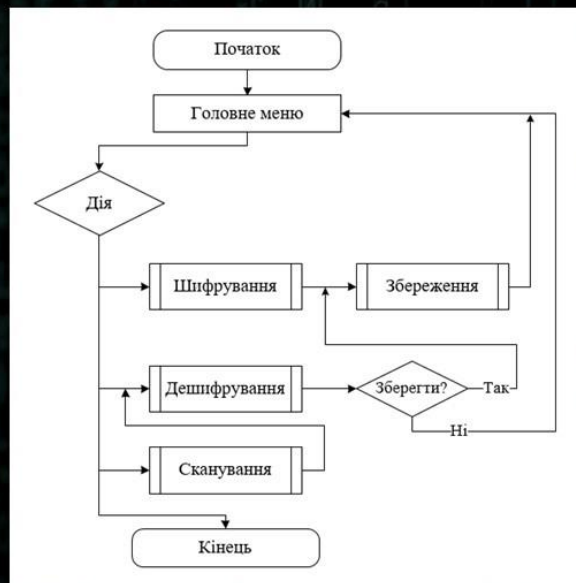
Ефективний спосіб кодування документу.



6

Рисунок Б.6 – Слайд 6

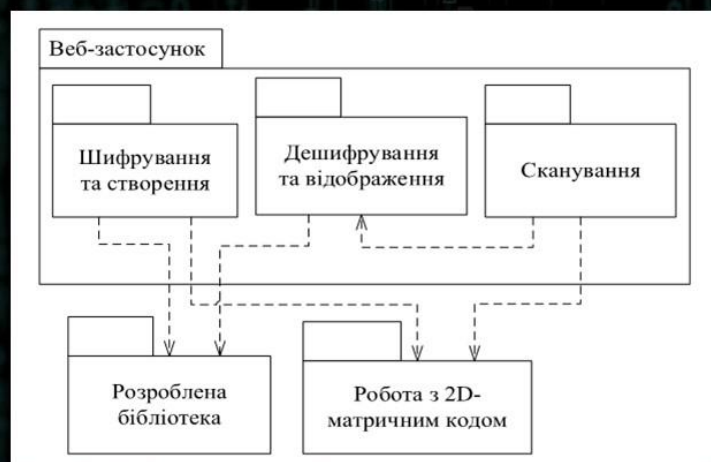
Схема функціонування застосунку



7

Рисунок Б.7 – Слайд 7

Діаграма залежностей застосунку



8

Рисунок Б.8 – Слайд 8

Вибір середовища розробки

Характеристика	Редактор VS Code	Середовища від JetBrains
Розповсюдження	Безкоштовно	Комерційний продукт
Підтримка мов для веб-розробки та мови C	Плагіни, LSP	Окремі середовища: WebStorm, CLion
Інструменти рефакторингу, авто-доповнення	Залежить від можливостей плагіну та підтримуваної їм конфігурації проекту	Гарантовано працює
Інтерфейс	Простий та інтуїтивний	Функціональний, але перевантажений
Швидкодія	Швидке відкриття проекту та навігації серед файлів	Повільне завантаження, важкі операції індексування у фоні
Віддалена розробка	Плагіни для SSH, WSL, Docker	Потребує ручного розгортання серверної частини Projector

9

Рисунок Б.9 – Слайд 9

Реалізація вебзастосунку

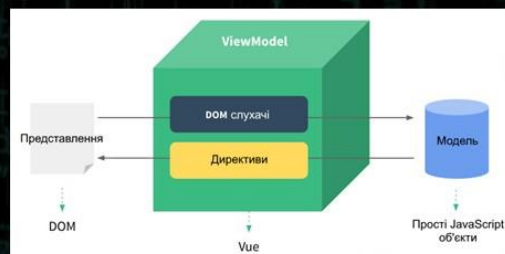
Обрано MVVM бібліотеку Vue.

Переваги:

- інтуїтивний механізм реактивності даних;
- детальна документація;
- однофайлові компоненти;
- підтримка TypeScript.

Недоліки:

- брак бібліотек;
- надмірна гнучкість.



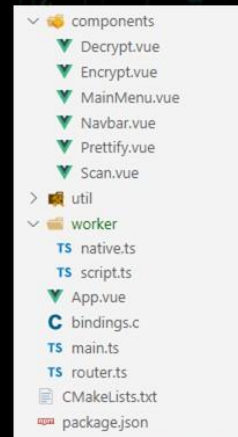
10

Рисунок Б.10 – Слайд 10

Структура програмної реалізації



Бібліотека

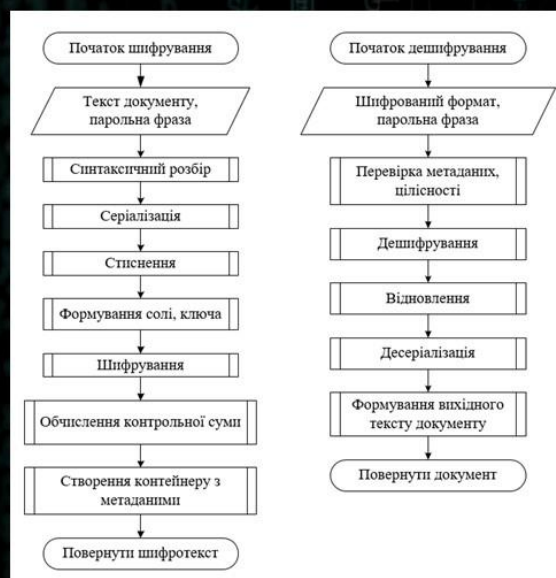


Вебзастосунок

11

Рисунок Б.11 – Слайд 11

Алгоритм для створення файлу із шифротекстом



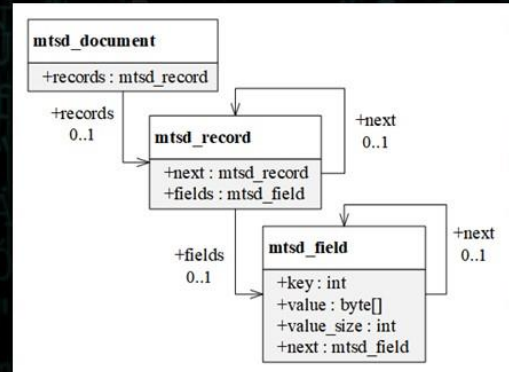
12

Рисунок Б.12 – Слайд 12

Синтаксичний розбір текстового документу для шифрування

Програмно реалізовано:

- 4 лексеми;
- підтримка LF, CRLF;
- кодування UTF-8;
- спадний метод побудови дерева;
- 3 вузла дерева.

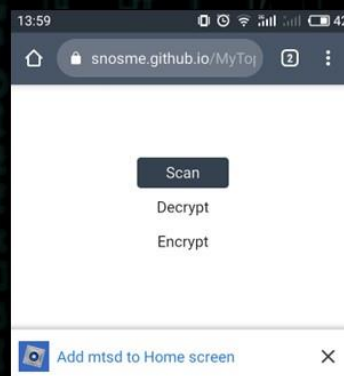


UML-діаграма структур, які формують синтаксичне дерево

13

Рисунок Б.13 – Слайд 13

Інтерфейс головного меню



14

Рисунок Б.14 – Слайд 14

Інтерфейс форми для шифрування

Decrypt Encrypt

Document format reference

```
password: example
---
text:
  multiline
  example
```

Secret

15

Рисунок Б.15 – Слайд 15

Інтерфейс збереження даних

Decrypt Encrypt Prettify

Characters / block: Blocks / line:

[Download _mtsd.bin](#) [Download font](#) hex

```
86f 006 dfe dce 187 7d8 00e
20a 2e4 893 b29 13c 95d 1cb
98e 474 242 623 e2f f48 65
```

Module size: 5 px 1 px 20 px

Margin size: 2 mod 0 5 modules

16

Рисунок Б.16 – Слайд 16

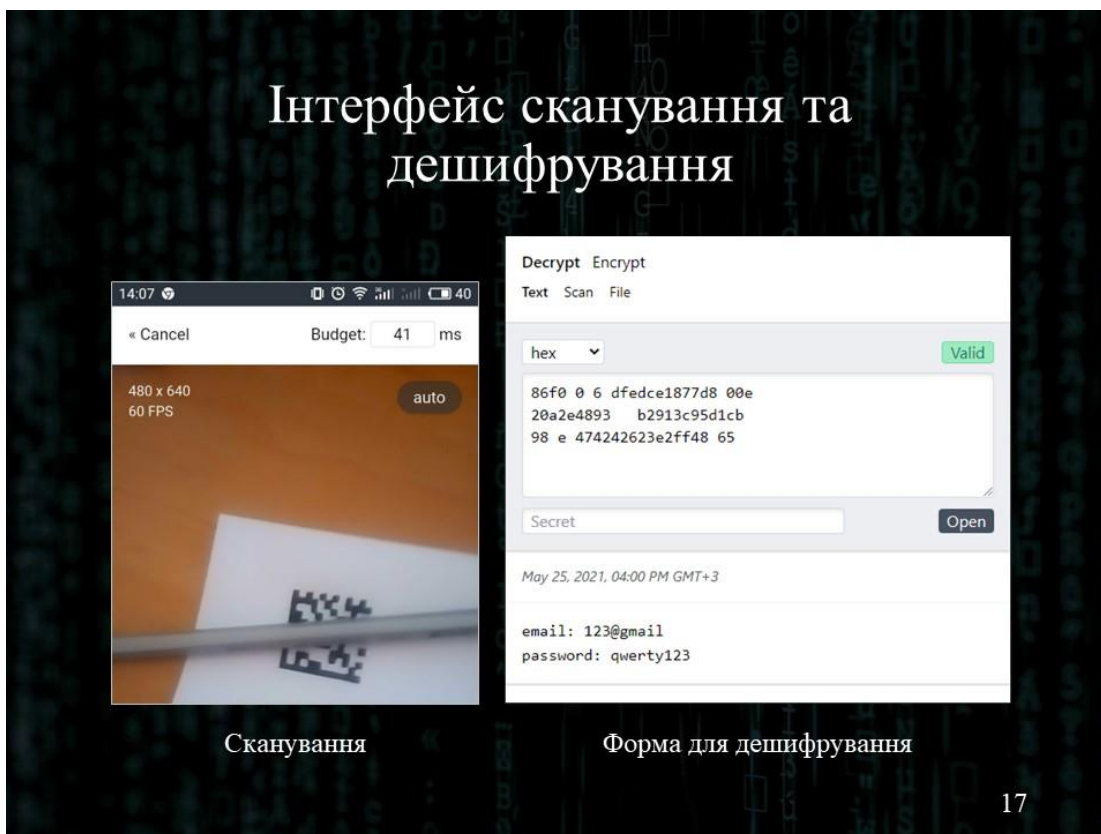


Рисунок Б.17 – Слайд 17

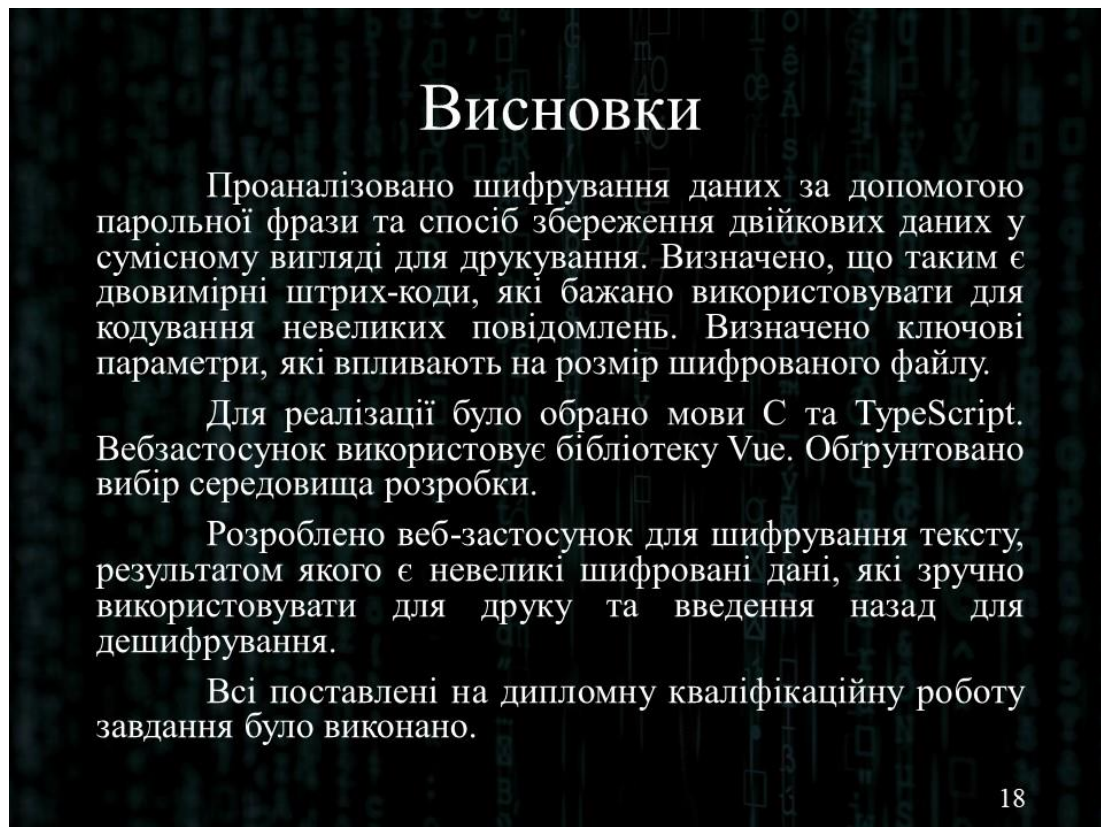


Рисунок Б.18 – Слайд 18