Міністерство освіти та науки України

Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

Звіт

До індивідуального завдання

З дисципліни: «Безпека програм та даних»

Виконав:

ст. гр. ПЗПІ-19-3

Селевич О.В.

Перевірив:

асистент кафедри ПІ

Олійник О.О.

Харків 2022

Тема роботи: Виконання індивідуальне завдання.

Мета роботи: Розшифрування повідомлення зашифроване шифром Цезаря з вхідного файлу. Підписання файлу у форматі pdf за допомогою ЕЦП двома способами: так щоб ЕЦП дописувався у файл, та так щоб він додавався окремими файлом. Створення донат кнопки для будь-якої платіжної системи.

Хід роботи:

Увесь програмний код, створений для виконання індивідуального завдання написаний за допомогою мови програмування JavaScript, а користувацький інтерфейс був створений за допомогою JavaScript фреймворку Vue.js.

1) **Розшифрування повідомлення зашифрованного шифром Цезаря**

Першим завданням є розшифрування вихідного файлу на українській мові, зашифрованого за допомогою шифру Цезаря, з назвою 18.txt, котрий має наступний вміст → – *тсґойїц Ецовмтсоз, Укдрсткоов, Рсдсфконй, Дкоарз тсоз, ьзудсрз хг Пгойркдній; д Ткджзррспц Ґцшфанспц ргтувпнц - д угм.*

З метою виконання розшифрування файлу програмним чином була створена функція breakCaesarsCipher, котра виконує атаку на зашифроване повідомлення. Шифр Цезаря є простим шифром моноалфавітної заміни у якому ключ може мати n значень, де n → кількість літер у алфавіті мови на якій був написаний текст. Функція breakCaesarsCipher виконує повний перебір усіх можливих ключів, котрі могли бути використані для зашифрування файлу. Ключ у шифрі Цезаря є цілочисельним значенням яке може бути як позитивним так і негативним. Ключ у шифрі цезаря відповідає за зсув літер у алфавіті. У разі негативного значення зсув виконується вліво, а у разі позитивного вправо. Тобто якщо ключем є 2, а мова, якою був

написаний текст є українською то літера усі літери “а” вихідного тексту стануть літерами “в” у шифрі.

Для з’ясування того, який ключ є правильним було використано показник статистики х-квадрат (Chi-squared Statistic). Статистика х-квадрат є мірою того, наскільки схожими є два категоріальні розподіли ймовірностей. Якщо два розподіли ідентичні, статистика хі-квадрат дорівнює 0, якщо розподіли дуже відрізняються, то результатом буде деяке більше число. Формула для статистики х-квадрат можливо побачити на рисунку 1.

$$X^2 = \sum_{i=A}^{i=Z} \frac{(O_i - E_i)^2}{E_i}$$

Рисунок 1 – Формула статистики х-квадрат

У цій формулі Oi - спостережувана кількість літер у вашому тексті, а Ei - очікувана кількість цієї літери у довжині вашого тексту.

На кожній ітерації у функції breakCaesarsCipher виконується розрахунок статистики х-квадрат за допомогою функції countChiSquaredStatistic. Програмний код функції countChiSquaredStatistic:

```
1. const countChiSquaredStatistic = (
2.   observedFrequencyMap,
3.   expectedFrequencyMap,
4.   cipherTextLength
5. ) => {
6.   let sum = 0;
7.
8.   observedFrequencyMap.forEach((value, key) => {
9.     const expectedCount =
10.         (expectedFrequencyMap.get(key) / 100) * cipherTextLength;
11.      const tmp = Math.pow(value - expectedCount, 2) /
12.  expectedCount;
13.      sum += tmp;
14.    });
15.
16.    return sum;
17.  };
```

Ця функція приймає на вхід три параметри: таблицю з очікуваними кількостями літер у тексті, таблицю з спостереженими кількостями літер у тексті та довжину зашифрованого повідомлення. Далі у цій функції виконуються розрахунки за формулою, що зазначена вище на рисунку 1.

У фукції breakCaesarsCipher на кожній ітерації виконується зсув вихідного масиву з літерами алфавіту масиву згідно з поточним значенням ключа. Потім обчислюється частотний розподіл символів отриманного тексту, де усі букви зашифрованого тексту були замінені згідно з зсуненними літерами у масиві алфавіту. Далі обчислюється значення х-квадрат статистики. У обчисленні х-квадрат статистики визначається наскільки схожими є два категоріальні розподіли ймовірностей (у даному випадку розподіли ймовірностей нормальної появи літер у тексті написанному на певній мові та у отриманному тексті). Далі якщо значення отримане за результатом обчислення функції countChiSquaredStatistic є меншим за попередне найменше збережене значення то зберігається значення ключа та отримане повідомлення стає найбільш ймовірним на те, щоб бути вихідним текстом до виконання шифрування. Далі функція breakCaesarsCipher повертає назад вихідне повідомлення, що з найбільшою ймовірністю і буде вихідним текстом до виконання шифрування за допомогою алгоритму Цезаря. У функції breakCaesarsCipher для ініціалізацію частотної таблиці спостереженої появи символів у тексті використовується функція initializeFrequencyMap. Програмний код функції initializeFrequencyMap:

```
1. export const initializeFrequencyMap = (inputText, language) => {
2.   const frequencyMap = new Map();
3.   const languageArray = getLanguageArray(language);
4.
5.   for (let letter of inputText) {
6.     const lowerCaseLetter = letter.toLowerCase();
7.
8.     if (languageArray.indexOf(lowerCaseLetter) !== -1) {
9.       if (frequencyMap.has(lowerCaseLetter)) {
10.          const count = frequencyMap.get(lowerCaseLetter);
11.          frequencyMap.set(lowerCaseLetter, count + 1);
```

```
        } else {
          frequencyMap.set(lowerCaseLetter, 1);
        }
      }
    }

    languageArray.forEach((elem) => {
      if (!frequencyMap.has(elem)) frequencyMap.set(elem, 0);
    });

    return frequencyMap;
  };
```

Для виконання зсуву літер у масиві алфавіту згідно з поточним значенням ключа використовується функція shiftArray. Програмний код функції shiftArray:

```
export const shiftArray = (shift, encryptionArray) => {
  const n = encryptionArray.length;

  const shiftedArray = [...Array(n)];
  for (let i = 0; i < n; i++) {
    shiftedArray[i] = encryptionArray[(i + shift) % n];
  }
  encryptionArray.splice(0, n, ...shiftedArray);
};
```

Для виконання моноалфавітної заміни використовується функція mapText, що зіставляє літери зашифрованого повідомлення до літер у зсуненному масиві з літерами алфавіту. Програмний код функції mapText:

```
export const mapText = (
  text,
  textLanguageArray,
  textEncryptionArray,
  makeLowerCase
) => {
  let result = [];
  let textCharacters = [...text.split("")];

    textCharacters.forEach((elem) => {
      const lowerCaseElem = elem.toLowerCase();
      const index = textLanguageArray.indexOf(lowerCaseElem);

```

```
      if (index === -1) result.push(elem);
      else {
        let resultCharacter = "";
        if (lowerCaseElem === elem || (lowerCaseElem !== elem &&
  makeLowerCase)) {
          resultCharacter = textEncryptionArray[index];
        } else {
          resultCharacter =
  textEncryptionArray[index].toUpperCase();
        }
        result.push(resultCharacter);
      }
    });

    return result.join("");
  };
```

Усі функції зазначені вище знаходяться у файлі caesarsCipher.js. Програмний код функції breakCaesarsCipher:

```
export const breakCaesarCipher = (cipher, language,
isBestShiftReturned) => {
  const languageArray = getLanguageArray(language);
  let encryptionArray = getLanguageArray(language);
  const languageFrequencyMap = getTableFromJson(language);

  const maxShift = languageArray.length;

  let tmpDistributionValue = Number.MAX_VALUE;
    let bestShift = 0;

    for (let i = 0; i < maxShift; i++) {
      encryptionArray = getLanguageArray(language);
      shiftArray(i, encryptionArray);
      const shiftedText = mapText(cipher, languageArray,
  encryptionArray, true);
      const cipherFrequencyMap =
  initializeFrequencyMap(shiftedText, language);
      const squaredXDistribution = countChiSquaredStatistic(
        cipherFrequencyMap,
        languageFrequencyMap,
        cipher.length
      );
      if (squaredXDistribution < tmpDistributionValue) {
        bestShift = i;
        tmpDistributionValue = squaredXDistribution;
      }
```

```
28.     }
29.     if (isBestShiftReturned) return bestShift;
30.
31.     encryptionArray = getLanguageArray(language);
32.
33.     shiftArray(bestShift, encryptionArray);
34.     return mapText(cipher, languageArray, encryptionArray,
35.   false);
36.   };
```

Для виконання розшифрування даних з вихідного текстового файлу був створений користувацький інтерфейс, що можна побачити на рисунку 2.

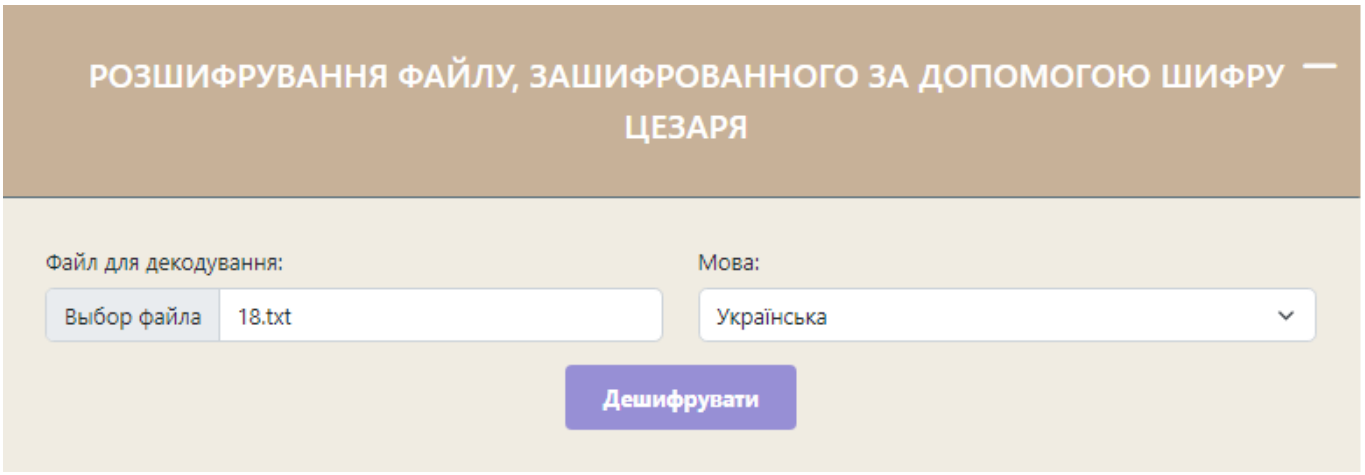

Рисунок 2 – Вигляд користувацького інтерфейсу для розшифрування даних

На рисунку два можна побачити форму, що містить два поля для вводу: для вихідного файлу з зашифрованим повідомленням та випадаючий список для вибору мови, на якій було написане зашифроване повідомлення. Після натискання на кнопку дешифрувати створений програмний застосунок поверне вам розшифрований файл та завантажить його на ваш пристрій.

У результаті роботи створений програмний застосунок повернув файл з розшифвррованим повідомленням, що має назву 18Output.txt та наступний вміст → *– поблизу Гуляйполе, Рівнопілля, Новосілки, Вільне поле, червоне та Малинівки; в Південному Бухському напрямку - в рай.*

2) Підписання файлу у форматі pdf за допомогою ЕЦП

З метою виконання другого завдання я створив pdf файл з назвою task2.pdf, котрий має наступний вміст → *Я Селевич Олексій Вадимович з групи ПЗПІ-19-3 бажаю виконати друге завдання для отримання 60 балів з дисципліни БПтД.* Далі цей файл був підписаний за допомогою ЕЦП двома способами: ЕЦП дописувався у файл (enveloped) та ЕЦП додавався окремим файлом (detached). Файл був підписаний за допомогою web застосунку Дія. Документ був підписаний за допомогою сервісу Дія.Підпис – Ua. Сторінку для вибору сервісу для підпису можна побачити на рисунку 3.

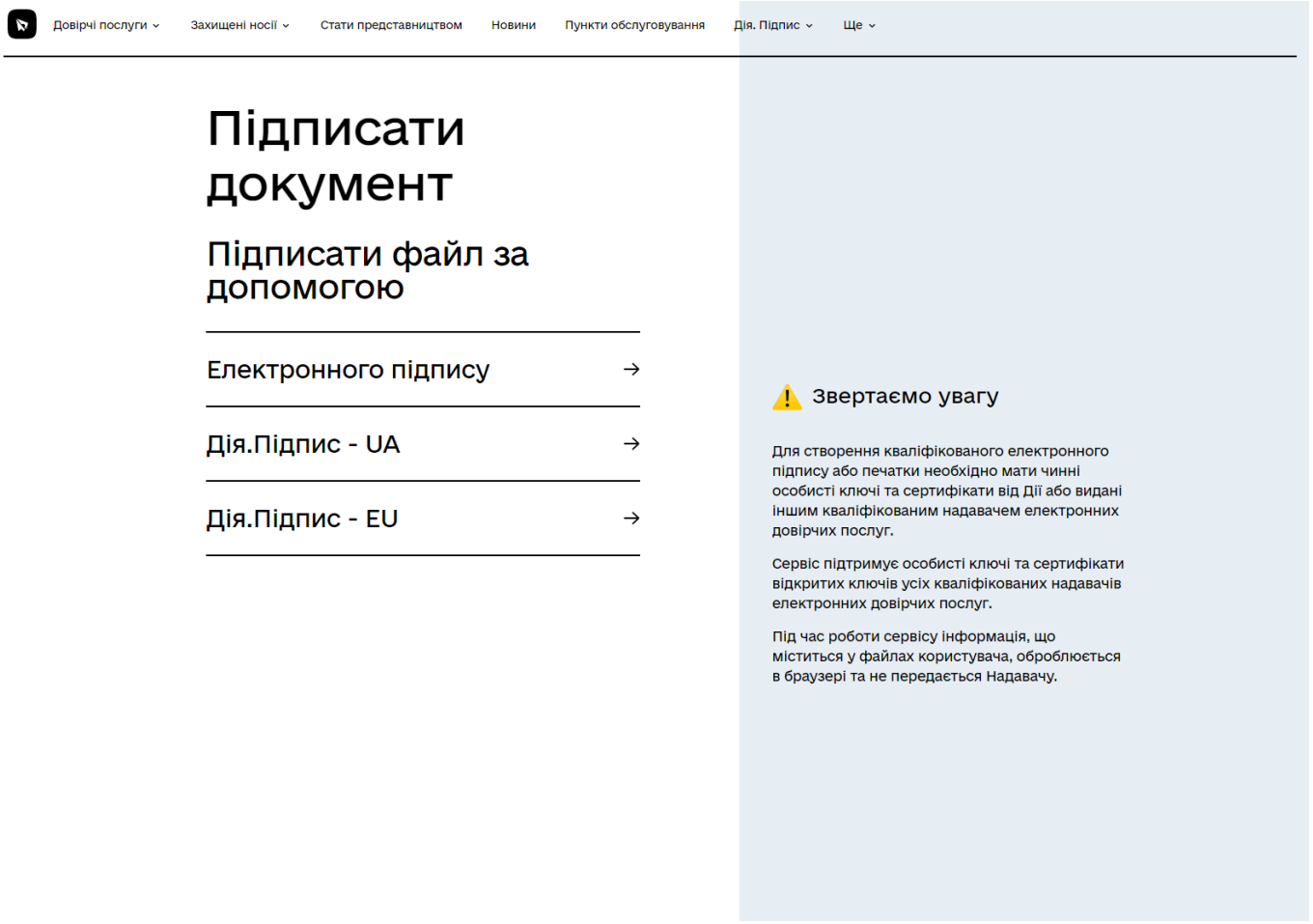

Рисунок 3 – Сторінка web застосунку Дія для вибру сервісу для підпису

Далі після вибору сервісу для підпису необхідно відсканувати QR-код за допомогою мобільного застосунку Дія з метою підтвердження особи, що виконуватиме підпис. Сторінку для сканування QR-коду можна побачити на рисунку 4.

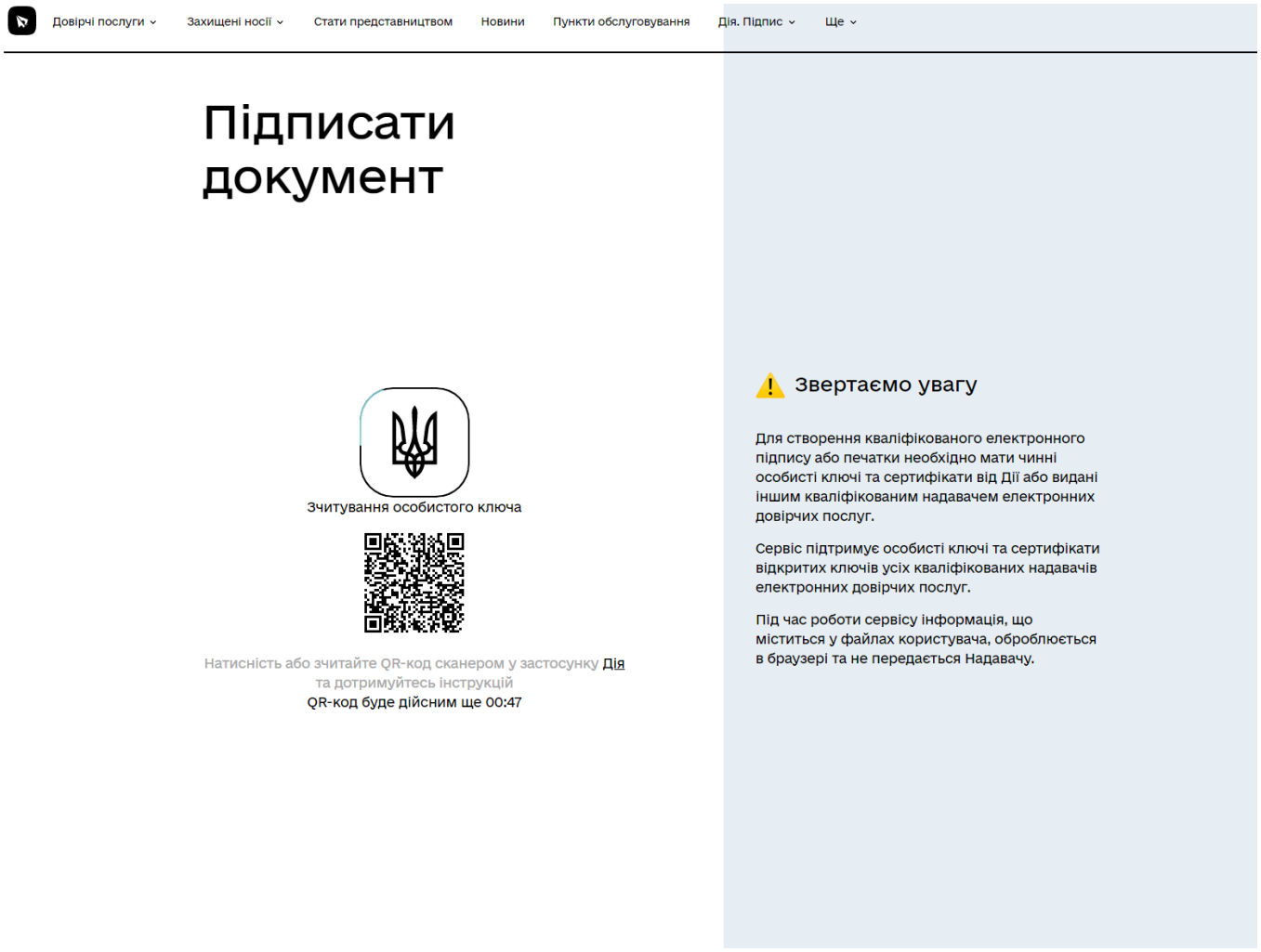

Рисунок 4 – Сторінка для сканування QR-коду

Далі після підтвердження особи ви потрапляєте на сторінку де бачите персональні дані та можете завантажити ЕЦП. На рисунку 5 можливо побачити елемент для завантаження ЕЦП.

Рисунок 5 – Елемент для завантаження ЕЦП

Потім на наступному кроці виконання підпису потрібно обрати алгоритм підпису, тип підпису, формат підпису та сам файл, котрий необхідно підписати. Тип підпису саме і відповідає за те, чи буде ЕЦП дописуватись у файл (enveloped) або буде доданий окремим файлом (detached). Сторінку для вибору налаштувань для виконання підпису можна побачити на рисунку 6.

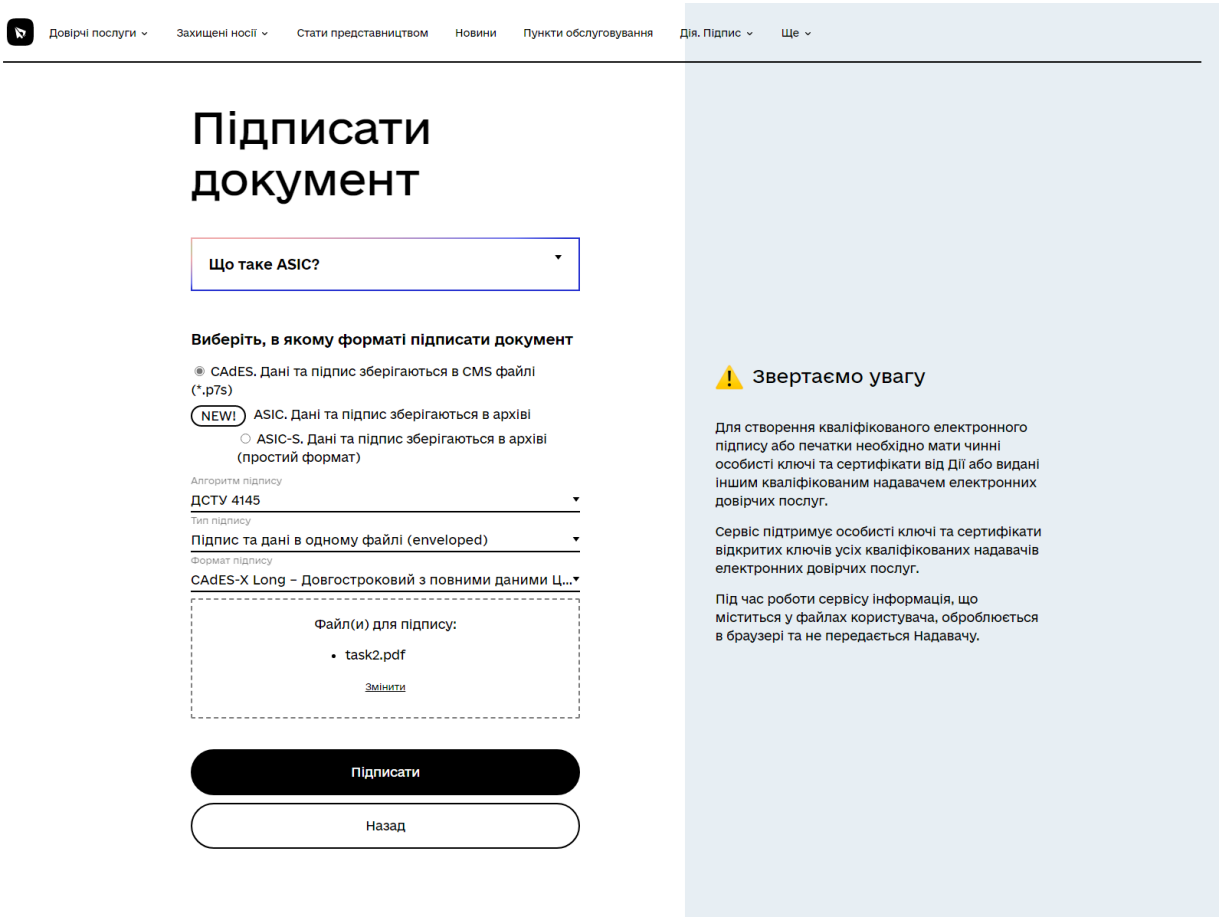

Рисунок 6 – Сторінка для вибору налаштувань для виконання підпису за допомогою ЕЦП

Після цього документ буде підписаний за допомогою ЕЦП. У результаті виконання підписання файлу двома способами були створені файли task2Detached.p7s (додавання окремим файлом) та task2Enveloped.p7s (дописування у файл). Підписані файли та вихідний pdf файл можливо знайти за наступними посиланнями у Google Drive:

https://drive.google.com/file/d/1zsYXAnds6pAeki9CLRTuXW4TsTMTWq9q/view?usp=sharing

https://drive.google.com/file/d/1dNWdbOJBAzEOmsZMvEVsri5Mrdk02ct3/view?usp=sharing

https://drive.google.com/file/d/13VoHRGoEd37H2uOo9ic8WQKHG49CBS3X/view?usp=sharing

Також ці файли можливо завантажити у створеному веб застосунку у наступній формі, що зображена на рисунку 7.

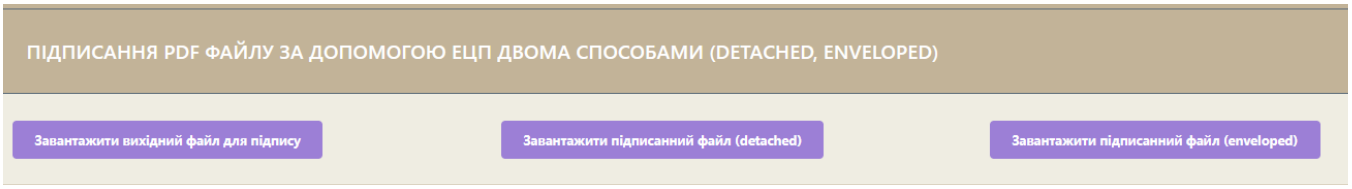

Рисунок 7 – Форма для завантаження файлів, створених у результаті виконання другого завдання

3) Створення донат кнопки для будь-якої платіжної системи

Була створена донат кнопка для платіжної системи PayPal. Ця кнопка була створена двома способами: через форму та програмним чином з коду. Для створення кнопки донату було використано PayPal's Donate SDK. Для використання PayPal’s Donate SDK потрібно зареєструватись у платіжній системі PayPal. Кнопка донату, котра створена за допомогою форми є реальною, тобто вона прив’язана до мого справжнього PayPal бізнес рахунку та за допомогою неї можливо виконувати реальні донати. Кнопка донату, що була створена програмним чином використовує PayPal sandbox. PayPal sandbox надає можливість імітувати реальні транзакції та це є дуже гарною можливістю для розробників. Коли існує зареєстрований аккаунт у платіжній системі PayPal то людині автоматично надається можливість використання усього списку функціоналу для розробників, а також два аккаунти Sandbox. Один аккаунт типу бізнес (у даному випадку саме він і використовується для створення тестової кнопки програмний чином), а інший персональний, з якого будуть виконуватись донати. На рисунку 8 можливо побачити Sandbox PayPal акаунти, що використовувались для тестування кнопки донату.

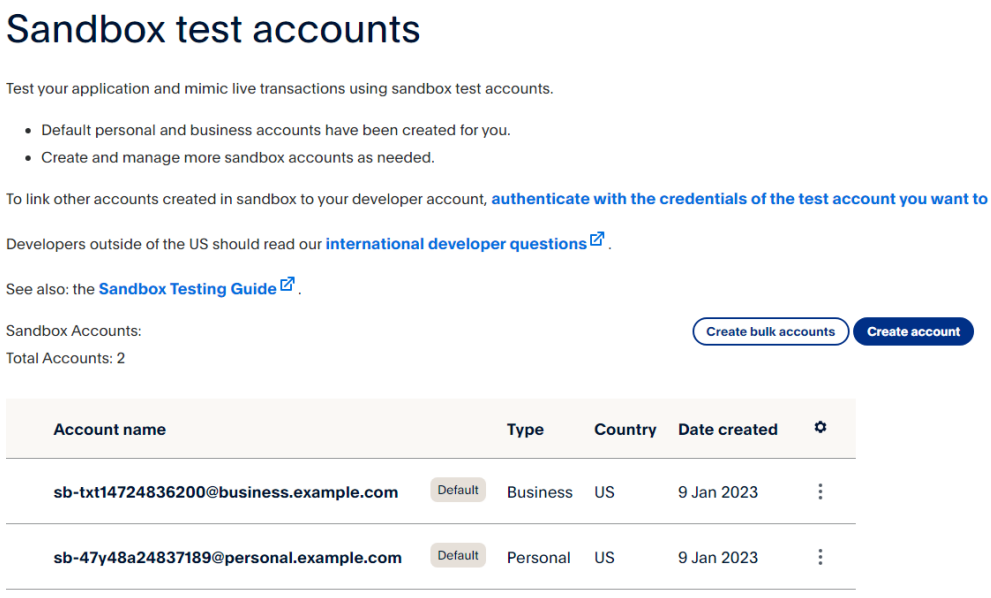

Рисунок 8 – PayPal Sandbox акаунти для тестування

Скрипт для створення донат кнопки за допомогою форми:

```
1.  <div>
2.             <form class="centered"
3. action="https://www.paypal.com/donate" method="post"
4. target="_top">
5.                 <input type="hidden" name="hosted_button_id"
6. value="7T7GLSPZSEBTG" />
7.                 <input type="image"
8. src="https://www.paypalobjects.com/en_US/i/btn/btn_donateCC_LG.g
9. f" border="0"
10.                        name="submit" title="PayPal - The safer,
11.  easier way to pay online!"
12.                        alt="Donate with PayPal button" />
13.             </form>
14.          </div>
```

Скрипт для створення донат кнопки програмним чином:

```
1.  data() {
2.         return {
3.             donationMade: false,
4.             donationStatus: '',
5.             donatedAmount: 0,
6.             currency: ''
7.         }
8.     },
9.     mounted() {
10.           const showDialog = (donatedAmount, donationStatus,
11.  currency) => {
12.              this.donationMade = true;
13.              this.donatedAmount = donatedAmount;
14.              this.currency = currency ;
15.              this.donationStatus = donationStatus;
16.              console.log(this);
17.          };
18.          PayPal.Donation.Button({
19.              env: 'sandbox',
```

```
            hosted_button_id: 'JJ6D8VDUQSCLC',
            image: {
                src:
'https://www.paypalobjects.com/en_US/i/btn/btn_donate_LG.gif',
                alt: 'Donate with PayPal button',
                title: 'PayPal - The safer, easier way to pay
online!',
            },
            onComplete: (params) => {
               showDialog(params.amt, params.st, params.cc);
            },
        }).render('#donate-button');
    },
    methods: {
        handleTransaction() {
            this.donationMade = false;
        }
    }
```

Вигляд двух створених кнопок можливо побачити на рисунку 9.

Рисунок 9 – Створені донат кнопки

Висновки: Створив програмну реалізацію алгоритму для розшифрування повідомлення зашифрованого шифром Цезаря з вхідного файлу. Підписав файл у форматі pdf за допомогою ЕЦП двома способами: так щоб ЕЦП дописувався у файл, та так щоб він додавався окремими файлом. Створив донат кнопку для платіжної системи PayPal.