

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. ПЗП-19-3

Селевич О.В.

Перевірив:

асистент кафедри ПІ

Олійник О.О.

Харків 2022

Тема роботи: Індивідуальне завдання.

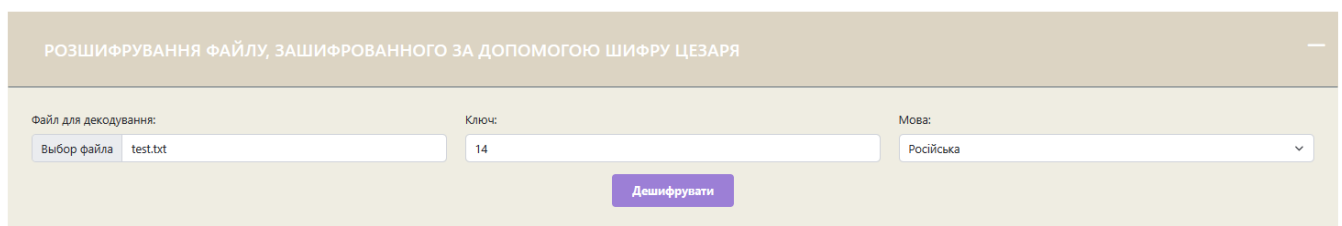
Мета роботи: Розшифрування повідомлення зашифроване шифром Цезаря з вхідного файлу. Підписання файлу у форматі pdf за допомогою ЕЦП двома способами: так щоб ЕЦП дописувався у файл, та так щоб він додавався окремим файлом. Створення донат кнопки для будь-якої платіжної системи.

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 зашифрованого шифром Цезаря.

З метою виконання декодування за допомогою шифру Цезаря була створена сторінка з формою, котру можна побачити на рисунку 1.



The image shows a web interface for decrypting a file using the Caesar cipher. The title bar at the top reads "РОЗШИФРУВАННЯ ФАЙЛУ, ЗАШИФРОВАННОГО ЗА ДОПОМОГОЮ ШИФРУ ЦЕЗАРЯ". Below the title bar, there are three input fields: "Файл для декодування:" with a file selection button labeled "Выбор файла" and the text "test.txt"; "Ключ:" with the value "14"; and "Мова:" with a dropdown menu showing "Російська". A purple button labeled "Дешифрувати" is positioned below the input fields.

Рисунок 1 – Створена сторінка з формою

Ця сторінка з формою містить три поля для вводу (inputs). Перший необхідний для обрання вихідного файлу з повідомленням, що було зашифроване за допомогою шифру Цезаря. У другому полі для вводу міститься значення ключа

(зсуву), котре є цілочисельним значенням, а у третьому полі для вводу обирається мова зашифрованого повідомлення. Функція, що відповідає за декодування шифру називається `decipherFile` та у результаті своєї роботи повертає користувачу назад розшифрований файл. Код функції `decipherFile`:

```
1.    decipherFile() {
2.
3.        const reader = new FileReader();
4.
5.        reader.readAsText(this.decryptionFile);
6.        reader.onload = () => {
7.            const inputFile = decrypt(this.shift,
8. reader.result, this.language);
9.            const link = document.createElement("a");
10.           link.download = "inputMessageFile.txt";
11.
12.           let blobData = new Blob([inputFile], {
13.               type: "text/plain"
14.           });
15.           link.href =
16. window.URL.createObjectURL(blobData);
17.           link.click();
18.       };
19.   }
```

Спочатку у цій функції виконується зчитування змісту файлу що був переданий у форму, а далі виконується виклик функції `decrypt` з файлу `caesarsCipher.js` для виконання декодування змісту файлу. Програмний код функції `decrypt`:

```
1. export const decrypt = (shift, cipher, language) => {
2.   const languageArray = getLanguageArray(language);
3.   let encryptionArray = getLanguageArray(language);
4.
5.   if (shift < 0) shift = languageArray.length + shift;
6.
7.   shiftArray(shift, encryptionArray);
8.
9.   return mapText(cipher, encryptionArray, languageArray, false);
```

```
10.  };
```

Ця функція приймає три параметри: шифр, мова тексту та ключ. Функція Спочатку виконується виклик функції `getLanguageArray`, що повертає масив з алфавітом. Далі виконується перевірка на наявність ключа з негативним зсувом. Функція `shiftArray` відповідає за виконання зсуву масиву для кодування/декодування. Код функції `shiftArray`:

```
1. export const shiftArray = (shift, encryptionArray) => {
2.   const n = encryptionArray.length;
3.
4.   const shiftedArray = [...Array(n)];
5.   for (let i = 0; i < n; i++) {
6.     shiftedArray[i] = encryptionArray[(i + shift) % n];
7.   }
8.   encryptionArray.splice(0, n, ...shiftedArray);
9. };
```

Далі виконується виклик функції `mapText` що повертає результуюче вихідне повідомлення, котре було зашифровано у вихідному файлі, та зіставляє символи з двох масивів (масиву алфавіту та масиву декодування) для виконання моноалфавітної заміни. Код функції `mapText`:

```
1. export const mapText = (
2.   text,
3.   textLanguageArray,
4.   textEncryptionArray,
5.   makeLowerCase
6. ) => {
7.   let result = [];
8.   let textCharacters = [...text.split("")];
9.
10.   textCharacters.forEach((elem) => {
11.     const lowerCaseElem = elem.toLowerCase();
12.     const index = textLanguageArray.indexOf(lowerCaseElem);
13.
14.     if (index === -1) result.push(elem);
15.     else {
16.       let resultCharacter = "";
17.       if (lowerCaseElem === elem || (lowerCaseElem !== elem &&
```

```

18. makeLowerCase)) {
19.     resultCharacter = textEncryptionArray[index];
20. } else {
21.                                     resultCharacter =
    textEncryptionArray[index].toUpperCase();
22. }
23.     result.push(resultCharacter);
24. }
25. });
26.
27. return result.join("");
28. };

```

Для тестування роботи було створено файл test.txt що має наступний зміст → юяэцрвёоъэ аъэрэ "щоябо", чц ёусэ н цощъмёчъо, ёбэ юояуьк бу твяощ юзяоцръуёкан. Этчь яоц эь вюэбъяупчь. Ключ 15 і обрана російська мова.

У результаті створений застосунок повернув файл з наступним розшифрованим повідомленням → прозвучало слово "карта", из чего я заключила, что парень не дурак поразвлечься. Один раз он употребил.

2) Підписання файлу у форматі pdf за допомогою ЕЦП

З метою виконання другого завдання я створив pdf файл, що має назву task2.pdf. У результаті виконання другого завдання документ був підписаний двома способами: ЕЦП дописувався у файл (enveloped) і додавався окремими файлом (detached). Документ був підписаний за допомогою ЕЦП у web застосунку “Дія” для підписання документів (інтерфейс застосунку можна побачити на рисунку 2).

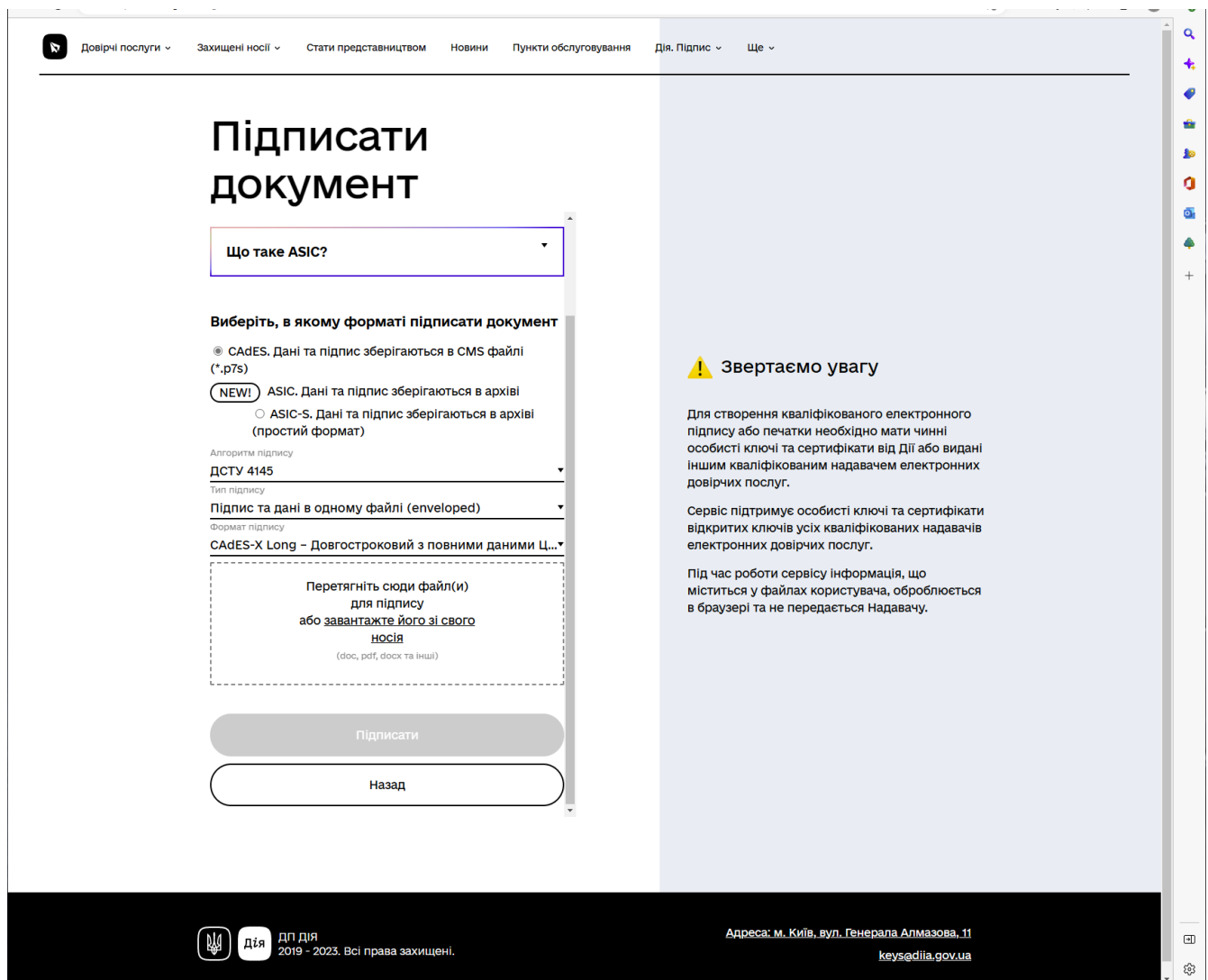


Рисунок 2 – Інтерфейс веб застосунку “Дія”

У результаті виконання підписання файлу двома способами були створені файли task2Detached.p7s (додавання окремим файлом) та task2Enveloped.p7s (дописання 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 веб застосунку у наступній формі, що зображена на рисунку 3.

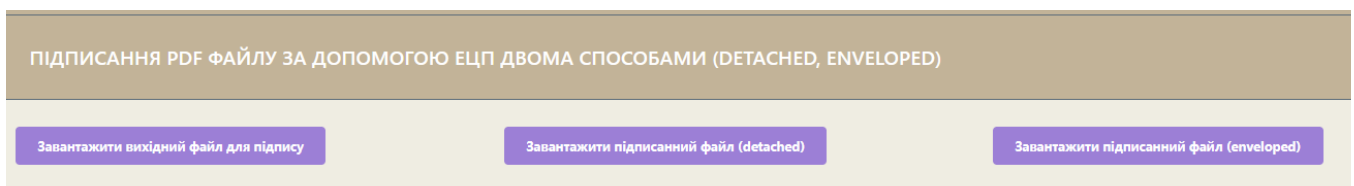


Рисунок 3 – Форма для завантаження файлів, створених у результаті виконання другого завдання

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. Для створення кнопки донату було використано PayPal's Donate SDK. 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',
20.        hosted_button_id: 'JJ6D8VDUQSCLC',
21.        image: {
22.            src:
23.      'https://www.paypalobjects.com/en_US/i/btn/btn_donate_LG.gif',
24.            alt: 'Donate with PayPal button',
25.            title: 'PayPal - The safer, easier way to pay
26.  online!',
27.        },
28.        onComplete: (params) => {
29.            showDialog(params.amt, params.st, params.cc);
30.        },
31.    }).render('#donate-button');
32. },
33.   methods: {
34.       handleTransaction() {
35.           this.donationMade = false;
36.       }

```


37. }

Вигляд двох створених кнопок можливо побачити на рисунку 4.



Рисунок 4 – 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.

