Міністерство освіти і науки України
Дніпровський національний університет імені Олеся Гончара
Факультет прикладної математики і комп’ютерних технологій
Кафедра комп’ютерних технологій

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 4

з курсу «Методи кібербезпеки»

на тему «Алгоритм шифрування RSA»

Варіант №14

Виконав:
студент гр. ПА-22-2
Овдієнко Андрій

Дніпро
2025

Зміст

1. Постановка задачі

Програмно реалізувати алгоритми шифрування RSA.

2. Опис розв’язку

RSA - це асиметричний алгоритм шифрування, який названий на честь Рона Ріввеста, Аді Шаміра та Лена Адлемана, які його винайшли (Rivest, Shamir, Adleman). Перша публікація датується 1977 роком.

Обирається два випадкові, різні, прості числа: p, q.

Визначаємо $n = p \cdot q$.

Визначаємо функцію Ейлера $\phi = (p - 1) \cdot (q - 1)$.

Визначаємо e – взаємно просте число відносно ϕ, яке більше 1, але менше за ϕ.

Визначаємо d, за відношенням: $\mathrm{mod}(e \cdot d, \phi) = 1$, де mod – остаток від ділення першого числа на друге.

Маємо комбінації: (e,n) - шифрування; (d,n) - розшифрування.

Для шифрування/дешифрування перетворимо кожний символ на ASCII – код, але так, як будемо працювати виключно з латинецею – будемо перевіряти, щоб n було більшим за 1000, щоб остаток від ділення не спотворював результат. А після операції перетворимо з ASCII – назад в utf-8 символьний тип char.

Для того, щоб зашифрувати текст m, необхідно обчислити таку рівність: $c = \mathrm{mod}(m^e, n)$, де c - зашифрований текст, m - простий текст, mod - остаток від ділення першого числа на друге, e і n – відомі параметри для шифрування.

Для того, щоб розшифрувати текст m, необхідно обчислити таку рівність: $c = \mathrm{mod}(m^d, n)$, де c - простий текст, m - зашифрований текст, mod - остаток від ділення першого числа на друге, d і n – відомі параметри для дешифрування.

3. Код програми

```
import random
import threading
import time
import tkinter as tk
from tkinter import font, ttk, filedialog, messagebox, scrolledtext

text = None
filename = None
key1 = None
key2 = None
e = None
d = None
n = None
min_n = 10**3

def check_e():
    global key1, key2

    rsult = []

    for i in range(2,key2):
        for j in range(2, key2):
            if i * j == key2 and i != j:
                rsult.append([i,j])

    for value in rsult:

        p = value[0]
        q = value[1]

        n = p * q

        if n < min_n:
```

```
            continue

        if n != key2:
            continue

        phi = (p - 1) * (q - 1)

        e = [i for i in range(2,phi)]

        del_on_phi = []

        for i in range(2,phi+1):
            if phi % i == 0:
                del_on_phi.append(i)
        e = [i for i in e if i not in del_on_phi]

        if key1 in e:
            return True

    return False

def check_d():
    global key1, key2

    rsult = []

    for i in range(2,key2):
        for j in range(2, key2):
            if i * j == key2 and i != j:
                rsult.append([i,j])

    for value in rsult:

        p = value[0]
        q = value[1]
```

```
        n = p * q

        if n < min_n:
            continue

        if n != key2:
            continue

        phi = (p - 1) * (q - 1)

        e = [i for i in range(2,phi)]

        del_on_phi = []

        for i in range(2,phi+1):
            if phi % i == 0:
                del_on_phi.append(i)

        e = [i for i in e if i not in del_on_phi]

        for E in e:
            if (key1 * E) % phi == 1:
                return True


    return False


def main(root):

    def input_text():
        try:
            [widget.destroy() for widget in root.winfo_children()]
        except Exception:
```

```
            pass

        def go(temp_text):

            match(temp_text):
                case "Write":

                    def save(temp_text):
                        global text
                        text = temp_text
                        messagebox.showinfo("Greate", f"You saved a text.")
                        main(root)

                    try:
                        [widget.destroy() for widget in root.winfo_children()]
                    except Exception:
                        pass

                    back_button = tk.Button(root, text="Back", command=lambda: main(root))

                    label = tk.Label(root, text="Enter text")

                    frame = tk.Text(
                        root,
                        height=10,
                        width=50,
                        undo=True,
                        autoseparators=True,
                        maxundo=-1
                    )

                    save_button = tk.Button(root, text="Save", command=lambda:
save(frame.get("1.0", tk.END)))

                    root.update()
```

```
                back_button.place(x=0, y=0)
                label.place(x=0, y=0)
                frame.place(x=0, y=0)
                save_button.place(x=0, y=0)

                root.update()

                empty_height = (root.winfo_height() - label.winfo_height() - frame.winfo_height() - save_button.winfo_height())/4

                label.place(x=(root.winfo_width() - label.winfo_width())/2, y=empty_height)
                frame.place(x=(root.winfo_width() - frame.winfo_width())/2, y=2*empty_height + label.winfo_height())
                save_button.place(x=(root.winfo_width() - save_button.winfo_width())/2, y=3*empty_height + label.winfo_height() +frame.winfo_height())

                root.update()



            case "Read from file":
                file_path = filedialog.askopenfilename(
                    title="Give a txt file",
                    filetypes=[("File txt", "*.txt"), ("All files", "*.*")],
                )
                global text, filename
                if not file_path:
                    messagebox.showinfo("Cancel", "You didn't choice a file")
                    text = None
                    filename = None
                else:
                    filename = file_path
                    with open(file_path, 'r', encoding='utf-8') as file:
                        text = file.read()
```

```
                messagebox.showinfo("Greate", f"You read a file {file_path}.")
                main(root)
            return

        back_button = tk.Button(root, text="Back", command=lambda: main(root))

        hint = ttk.Label(root, text="Do you want write text or import from file?")

        choice = ttk.Combobox(
            root,
            values=["Write", "Read from file"],
            state="readonly"
        )
        choice.current(0)

        button = ttk.Button(root, text="Go", command=lambda: go(choice.get()))

        empty_height = (root.winfo_height() - hint.winfo_height() - choice.winfo_height() -
button.winfo_height())/4

        back_button.place(x=0, y=0)
        hint.place(x=0, y=0)
        choice.place(x=0, y=0)
        button.place(x=0, y=0)

        root.update()

        back_button.place(x=10, y=10)
        hint.place(x=(root.winfo_width() - hint.winfo_width()) / 2, y=empty_height)
        choice.place(x=(root.winfo_width() - choice.winfo_width()) / 2, y=2 * empty_height +
hint.winfo_height())
        button.place(x=(root.winfo_width() - button.winfo_width()) / 2,
                     y=3 * empty_height + hint.winfo_height() + choice.winfo_height())

        root.update()
```

```
        '''
        frame = tk.Text(
            root,
            height=10,
            width=50,
            undo=True,
            autoseparators=True,
            maxundo=-1
        )
        frame.place(x=0,y=0)
'''

    def int_to_str(x):
        return chr(int(x))

    def str_to_int(x):
        return int(ord(str(x)))

    def mod(a, b):
        return a % b

    def multiply(a, b):
        return a * b

    def eler(p, q):
        return (p - 1) * (q - 1)

    def rand_int_between(a, b):
        x = a
        while x == a or x == b:
            x = random.randint(a, b)
        return x

    def rand_simple_between(a, b):
```

```
    if a < 2:
        a = 2

    if round(b - a, 2) == 0 or a >= b or a < 2:
        return None

    array = [[i, 1] for i in range(a, b + 1)]

    for i, _ in enumerate(array):

        for j in range(0, i):
            if array[i][0] % array[j][0] == 0:
                array[i][1] = 0
                break

    array = [value[0] for value in array if value[1] == 1]

    if len(array) == 0:
        return None

    return random.choice(array)

def del_numbers(x):
    if x <= 1:
        return None

    return [value for value in [i for i in range(2, x + 1)] if x % value == 0]

def friend_simple(x):
    if x <= 1:
        return None

    result = [value for value in range(2, x + 1) if value not in del_numbers(x)]

    if len(result) == 0:
```

```
            return None

        return random.choice(result)

    def loading_window(root):
        loading = tk.Toplevel(root)
        loading.title("Завантаження...")
        loading.geometry("300x100")
        loading.transient(root)
        loading.grab_set()
        loading.attributes("-topmost", True)
        loading.protocol("WM_DELETE_WINDOW", lambda: None)
        for widget in loading.winfo_children():
            widget.destroy()
        label = tk.Label(loading, text="Завантаження...", bg="white", fg="black")
        label.place(x=0, y=0)
        root.update()
        label.place(x=(loading.winfo_width() - label.winfo_width()) / 2,
                    y=(loading.winfo_height() - label.winfo_height()) / 2)
        root.update()
        return loading

    def randomaze():
        global e, d, n
        while True:
            p = rand_simple_between(1, 100)
            if p is None:
                continue

            q = rand_simple_between(1, 100)
            if q is None:
                continue

            if p == q:
                continue
```

```
        n = multiply(p, q)

        if n < min_n:
            continue

        phi = eler(p, q)

        if phi is None:
            continue

        e = friend_simple(phi)

        if e is None:
            continue

        d = p
        start = time.time()
        while True:
            if time.time() - start > 0.1:
                d=None
                break
            d += 1
            if e == d:
                continue
            if mod(d * e, phi) == 1:
                break
        if d is None:
            continue

        break

def get_e_d_n():

    global e,d,n
```

```
            loading = loading_window(root)

            def on_complete():
                loading.destroy()
                show()

            def thread_wrapper():
                randomaze()
                root.after(0, on_complete)

            threading.Thread(target=thread_wrapper, daemon=True).start()

        def show():
            try:
                [widget.destroy() for widget in root.winfo_children()]
            except Exception:
                pass

            frame = tk.Text(
                root,
                height=10,
                width=50,
                undo=True,
                autoseparators=True,
                maxundo=-1,
                state='disabled'
            )
            frame.configure(state='normal')
            frame.delete("1.0", "end")
            frame.insert("1.0", f"I find this solution:\n\te = {e};\n\td = {d};\n\tn = {n}.")
            frame.configure(state='disabled')
            frame.place(x=(root.winfo_width() - frame.winfo_width())/2, y=(root.winfo_height() -
frame.winfo_height())/2)
```

```
    back_button = tk.Button(root, text="Back", command=lambda: main(root))
    back_button.place(x=10, y=10)

    root.update()

    back_button.place(x=10, y=10)
    frame.place(x=(root.winfo_width() - frame.winfo_width()) / 2,
                y=(root.winfo_height() - frame.winfo_height()) / 2)
    root.update()

def input_key():
    try:
        [widget.destroy() for widget in root.winfo_children()]
    except Exception:
        pass

    def go(temp_text):

        match (temp_text):
            case "Write":

                def save(temp_text):
                    global key1, key2
                    temp_text = str(temp_text).replace('  ',' ')
                    if temp_text[0] == ' ':
                        temp_text = temp_text[1:]
                    if temp_text[-1] == ' ':
                        temp_text = temp_text[:-1]

                    key1 = ""
                    key2 = ""

                    i = 0
                    while i < len(temp_text) and temp_text[i] != ' ':
```

```
            key1 += temp_text[i]
            i+=1
        i+=1
        while i < len(temp_text):
            key2 += temp_text[i]
            i+=1

        try:
            key1 = int(key1)
            key2 = int(key2)
        except Exception:
            key1 = None
            key2 = None
            messagebox.showerror("Error","You didn't enter two number separated by
space.")
            return

        if key1 <= 1 or key2 <= 1:
            key1 = None
            key2 = None
            messagebox.showerror("Error","The key is invalid.")
            return

        messagebox.showinfo("Greate", f"You saved a key.")
        main(root)

    try:
        [widget.destroy() for widget in root.winfo_children()]
    except Exception:
        pass

    back_button = tk.Button(root, text="Back", command=lambda: main(root))

    label = tk.Label(root, text="Enter text")
```

```
                    frame = tk.Text(
                        root,
                        height=10,
                        width=50,
                        undo=True,
                        autoseparators=True,
                        maxundo=-1
                    )

                    save_button = tk.Button(root, text="Save", command=lambda:
save(frame.get("1.0", tk.END)))

                    root.update()

                    back_button.place(x=0, y=0)
                    label.place(x=0, y=0)
                    frame.place(x=0, y=0)
                    save_button.place(x=0, y=0)

                    root.update()

                    empty_height = (
                                    root.winfo_height() - label.winfo_height() -
frame.winfo_height() - save_button.winfo_height()) / 4

                    label.place(x=(root.winfo_width() - label.winfo_width()) / 2, y=empty_height)
                    frame.place(x=(root.winfo_width() - frame.winfo_width()) / 2,
                            y=2 * empty_height + label.winfo_height())
                    save_button.place(x=(root.winfo_width() - save_button.winfo_width()) / 2,
                                y=3 * empty_height + label.winfo_height() + frame.winfo_height())

                    root.update()

                case "Read from file":
                    global key1, key2
```

```
file_path = filedialog.askopenfilename(
    title="Give a txt file",
    filetypes=[("File txt", "*.txt"), ("All files", "*.*")],
)

if not file_path:
    messagebox.showinfo("Cancel", "You didn't choice a file")
    key1 = None
    key2 = None
else:
    with open(file_path, 'r', encoding='utf-8') as file:
        temp_text = file.read()

    temp_text = str(temp_text).replace('  ', ' ')
    if temp_text[0] == ' ':
        temp_text = temp_text[1:]
    if temp_text[-1] == ' ':
        temp_text = temp_text[:-1]

    key1 = ""
    key2 = ""

    i = 0
    while i < len(temp_text) and temp_text[i] != ' ':
        key1 += temp_text[i]
        i += 1
    i += 1
    while i < len(temp_text):
        key2 += temp_text[i]
        i += 1

    try:
        key1 = int(key1)
        key2 = int(key2)
    except Exception:
```

```
                        key1 = None
                        key2 = None
                        messagebox.showerror("Error", "You didn't enter two number separated by space in the file.")
                        return

                    if key1 <= 1 or key2 <= 1:
                        key1 = None
                        key2 = None
                        messagebox.showerror("Error","The key is invalid.")
                        return

                    messagebox.showinfo("Greate", f"You read a file {file_path}.")
                    main(root)
                return

            back_button = tk.Button(root, text="Back", command=lambda: main(root))

            hint = ttk.Label(root, text="Do you want write key or import from file?")

            choice = ttk.Combobox(
                root,
                values=["Write", "Read from file"],
                state="readonly"
            )
            choice.current(0)

            button = ttk.Button(root, text="Go", command=lambda: go(choice.get()))

            empty_height = (
                            root.winfo_height() - hint.winfo_height() - choice.winfo_height() - button.winfo_height()) / 4

            back_button.place(x=0, y=0)
            hint.place(x=0, y=0)
```

```
            choice.place(x=0, y=0)
            button.place(x=0, y=0)

            root.update()

            back_button.place(x=10, y=10)
            hint.place(x=(root.winfo_width() - hint.winfo_width()) / 2, y=empty_height)
            choice.place(x=(root.winfo_width() - choice.winfo_width()) / 2, y=2 * empty_height +
hint.winfo_height())
            button.place(x=(root.winfo_width() - button.winfo_width()) / 2,
                         y=3 * empty_height + hint.winfo_height() + choice.winfo_height())

            root.update()

        def show_text_and_key():
            global text, key1,key2

            try:
               [widget.destroy() for widget in root.winfo_children()]
            except Exception:
               pass
            root.title("Show the text and the key")
            button_back = tk.Button(root, text='<-', font=main_font, command=lambda: main(root))
            button_back.place(x=10, y=10)


            label_hint = tk.Label(root, text="Your text is:", font=main_font)
            text_hint = scrolledtext.ScrolledText(root, wrap=tk.WORD, font=main_font,
                                                  width=50, height=10)
            text_hint.pack(fill=tk.BOTH, expand=True)
            text_hint.delete("1.0", "end")
            text_hint.insert("1.0", str(text if text is not None else ""))
            text_hint.config(state="disabled")
            label_key = tk.Label(root, text=f"Your key is ({key1 if key1 is not None else ""}; {key2
if key2 is not None else ""}).", font=main_font)
```

```
            label_hint.place(x=0, y=0)
            text_hint.place(x=0, y=0)
            label_key.place(x=0, y=0)

            root.update()


            empty_height = (root.winfo_height() - label_hint.winfo_height() -
text_hint.winfo_height() - label_key.winfo_height())/4

            label_hint.place(x=(root.winfo_width() - label_hint.winfo_width())/2, y=empty_height)
            text_hint.place(x=(root.winfo_width() - text_hint.winfo_width())/2, y=2*empty_height
+ label_hint.winfo_height())
            label_key.place(x=(root.winfo_width() - label_key.winfo_width())/2,
y=3*empty_height + label_hint.winfo_height() + text_hint.winfo_height())

            root.update()

        def write_to_file():
            global filename, text

            if not filename:
                filename = filedialog.askopenfilename(
                    title="Select text file",
                    filetypes=[("Text files", "*.txt"), ("All files", "*.*")]
                )
                if not filename:
                    filename = ""
                    messagebox.showerror("Error", "No file selected!\n")
                    main(root)
            else:
                filename = filename.replace('.txt', '_out.txt')

            try:
```

```
            with open(filename, 'w', encoding='utf-8') as file:
                file.write(text)
            messagebox.showinfo("Success", f"Text file: \"{filename}\" - saved successfully!")
            filename = ""
        except Exception:
            messagebox.showerror("Error", f"The file: \"{filename}\" doesn't have coding utf-8.")
            main(root)

    def rsa():
        global text, key1, key2

        if text is None or key1 is None or key2 is None:
            messagebox.showerror("Error", "You must fill all the fields before you proceed.")
            return

        try:
            [widget.destroy() for widget in root.winfo_children()]
        except Exception:
            pass

        def go(_):
            global text, key1, key2

            match(_):
                case "Encrypt":
                    if not check_e():
                        messagebox.showerror("Error", "The (e,n) is invalid.")
                        return
                case "Decrypt":
                    if not check_d():
                        messagebox.showerror("Error", "The (d,n) is invalid.")
                        return


            result = []
```

```
            for value in text:
                result.append(str_to_int(value))

            for i,value in enumerate(result):
                result[i] = mod(value ** key1,key2)

            text = ""

            for value in result:
                text += int_to_str(value)

            messagebox.showinfo("Success", f"The text is {_.lower()}.")
            main(root)

        back_button = tk.Button(root, text="Back", command=lambda: main(root))

        hint = ttk.Label(root, text="Do you want to Encrypt or Decrypt?")

        choice = ttk.Combobox(
            root,
            values=["Encrypt", "Decrypt"],
            state="readonly"
        )
        choice.current(0)

        button = ttk.Button(root, text="Go", command=lambda: go(choice.get()))

        empty_height = (root.winfo_height() - hint.winfo_height() - choice.winfo_height() - button.winfo_height()) / 4

        back_button.place(x=0, y=0)
        hint.place(x=0, y=0)
        choice.place(x=0, y=0)
        button.place(x=0, y=0)
```

```
            root.update()

            back_button.place(x=10, y=10)
            hint.place(x=(root.winfo_width() - hint.winfo_width()) / 2, y=empty_height)
            choice.place(x=(root.winfo_width() - choice.winfo_width()) / 2, y=2 * empty_height +
hint.winfo_height())
            button.place(x=(root.winfo_width() - button.winfo_width()) / 2,
                         y=3 * empty_height + hint.winfo_height() + choice.winfo_height())

            root.update()


        try:
            [widget.destroy() for widget in root.winfo_children()]
        except Exception:
            pass

        root.title("RSA (OVDIIENKO ANDRII)")
        root.geometry(f"900x800")
        root.resizable(False, False)

        main_font = tk.font.Font(family=font.families()[0] if "Times New Roman" not in
font.families() else "Times New Roman", size=24)
        root.option_add("*Font", main_font)


        hints = [
            ["Input the text", input_text],
            ["Input the key", input_key],
            ["Use the RSA", rsa],
            ["Create and show a random {e,d,n}", get_e_d_n],
            ["Show the text and the key", show_text_and_key],
            ["Write the text in the file", write_to_file]
        ]
```

```
        buttons = []

        for hint in hints:
            buttons.append(tk.Button(root, text=hint[0], font=main_font, command=hint[1]))
            buttons[-1].pack()

        root.update()

        empty_height = (root.winfo_height() - len(buttons) * max(button.winfo_height() for button
in buttons)) / (
                len(buttons) + 1)

        for i, button in enumerate(buttons):
            button.place(x=(root.winfo_width() - button.winfo_width()) / 2,
                         y=(i + 1) * empty_height + i * max(button.winfo_height() for button in buttons))
        root.update()

        root.mainloop()


    if __name__ == '__main__':
        window = tk.Tk()
        main(window)
```

4. Опис інтерфейса

Після запуска програми можна побачити головне меню.

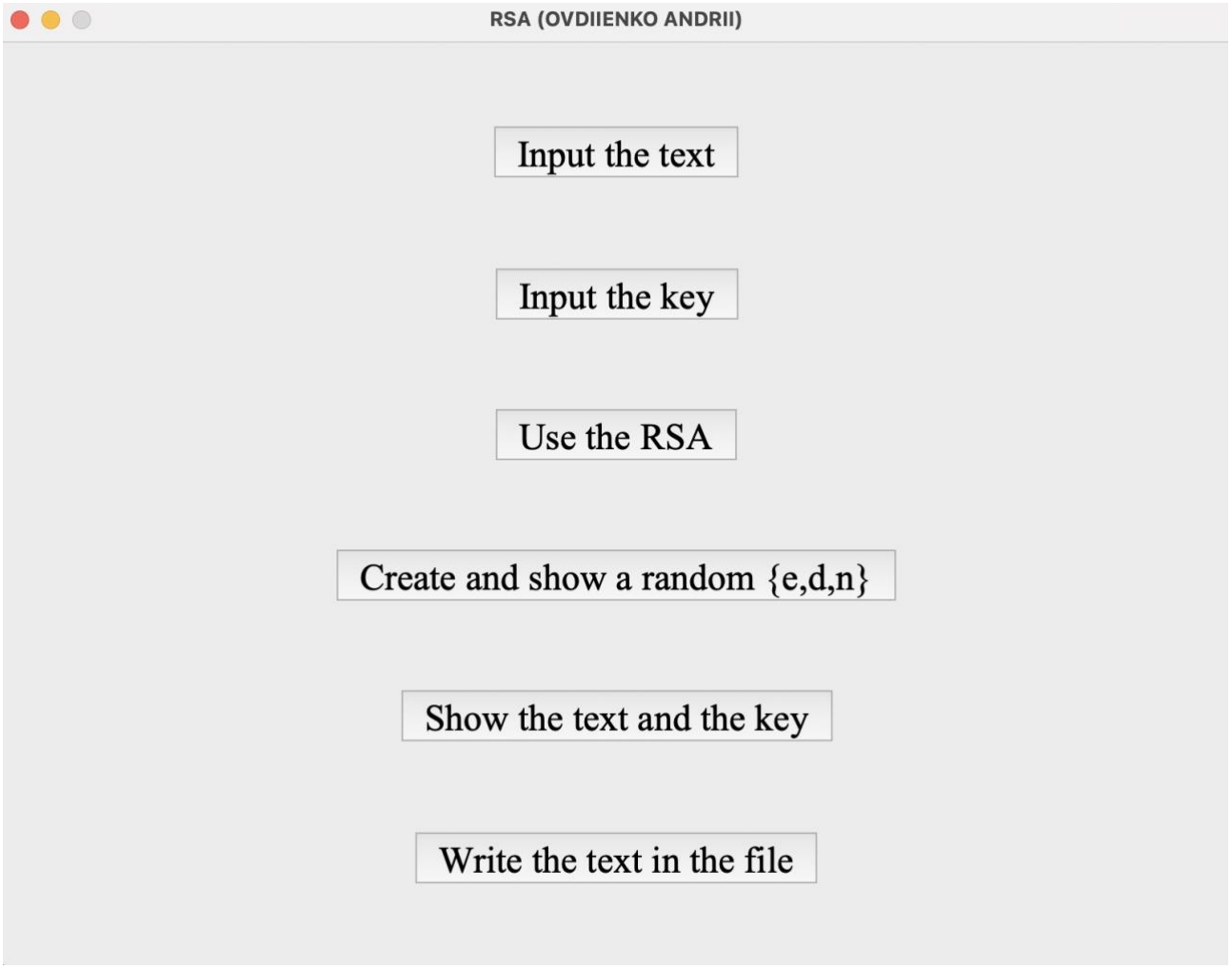

Рисунок 1 – Головне меню програми.

Кнопка “Input the text” дозволяє взяти текст із файла або написати самому.

Кнопка “Input the key” дозволяє взяти ключ із файла або написати самому. Тут комбінація мається на увазі, наприклад вам потрібна комбінація (e, n) – тому ви маєте написати в текстовому полі “123 456” – тоді присвояться значення (e, n) = (123; 456) – для (d, n) - аналогічно. Тобто в якосі ключа йде комбінація двох цілих чисел: (e, n) – для шифрування; (d, n) – для дешифрування.

Кнопка “Use the RSA” дозволяє зашифрувати або розшифрувати текст. Під час шифрування/розшифрування перевіряється достовірність параметрів (e, n) або (d, n) – чи можлива така комбінація взагалі.

Кнопка “Create and show a random {e, d, n}” – випадкова генерація можливого ключа, але не задання ключа безпосередньо в системі, рандомне обчислення і можливі ключі (можуть створюватися до 20 секунд, але у більшості випадків до 5 секунд).

Кнопка “Show the text and the key” – дозволяє подивитися на поточний текст та пару ключів.

Кнопка “Write the text in the file” – дозволяє записати текст у файл. Зауважимо, якщо текст був зчитаний з файла – то програма в кінці замість “.txt” підставить “_out,txt”, акщо текст був написаний власноруч – відкриється проводник та буде запропоновано вибрати файл для збереження.

Розмір вікна змінити не можна - це зроблено для того, щоб уникнути помилок відображення.

5. Приклад роботи програми

Відкриємо головне меню та натиснемо на кнопку для додавання текста.

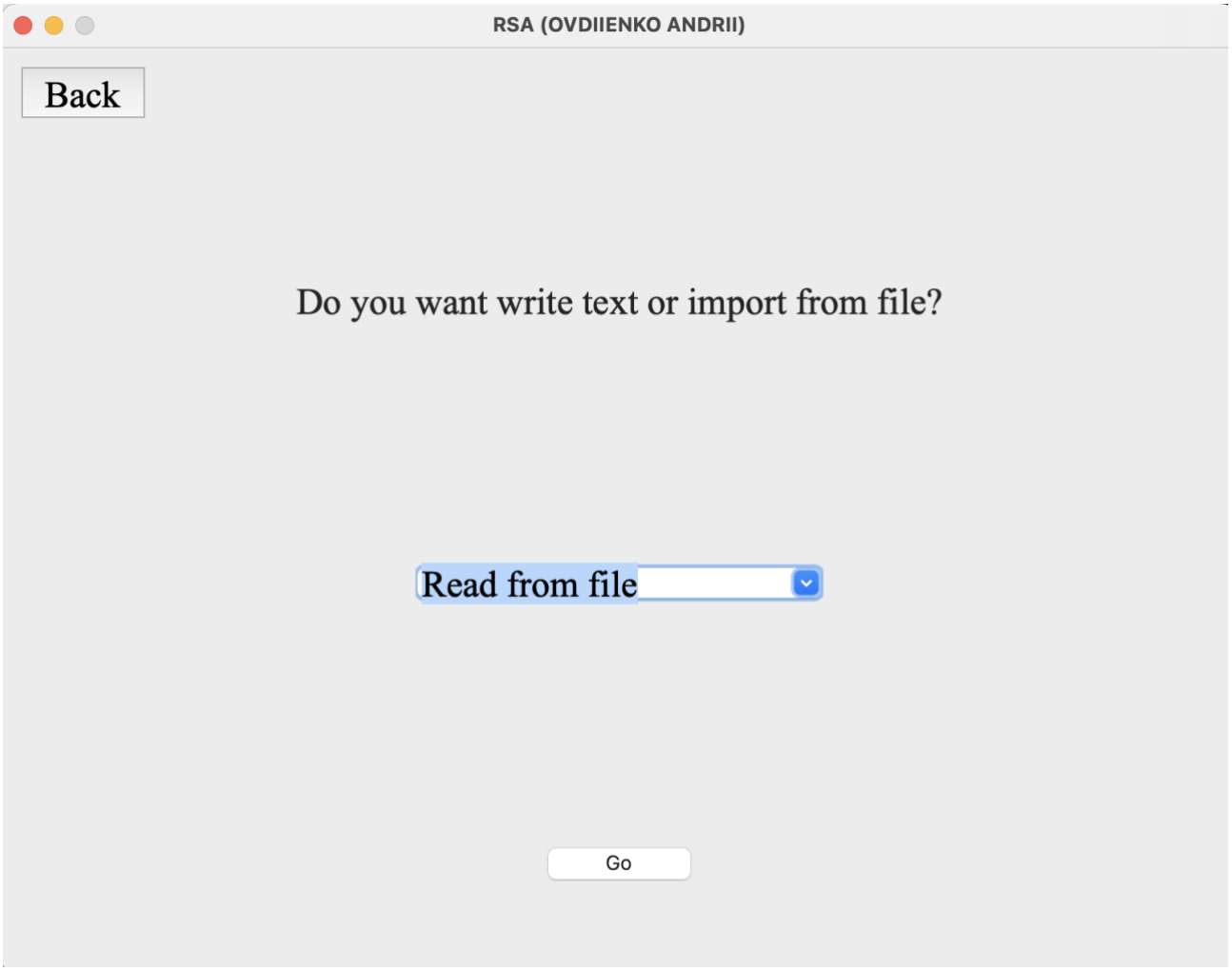

Рисунок 2 – Вибір завантаження текста з файла.

Обиремо файл.

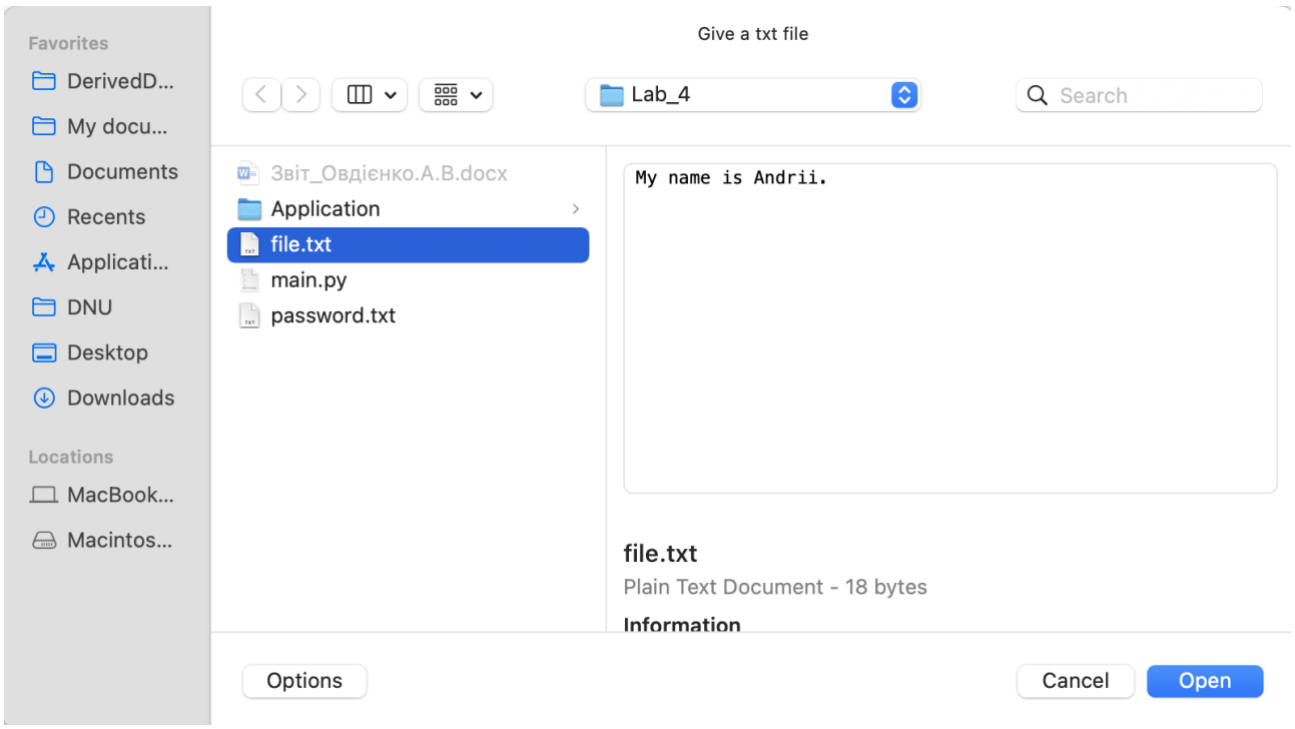

Рисунок 3 – Обираємо файл.

Згенеруємо випадкову пару ключів.

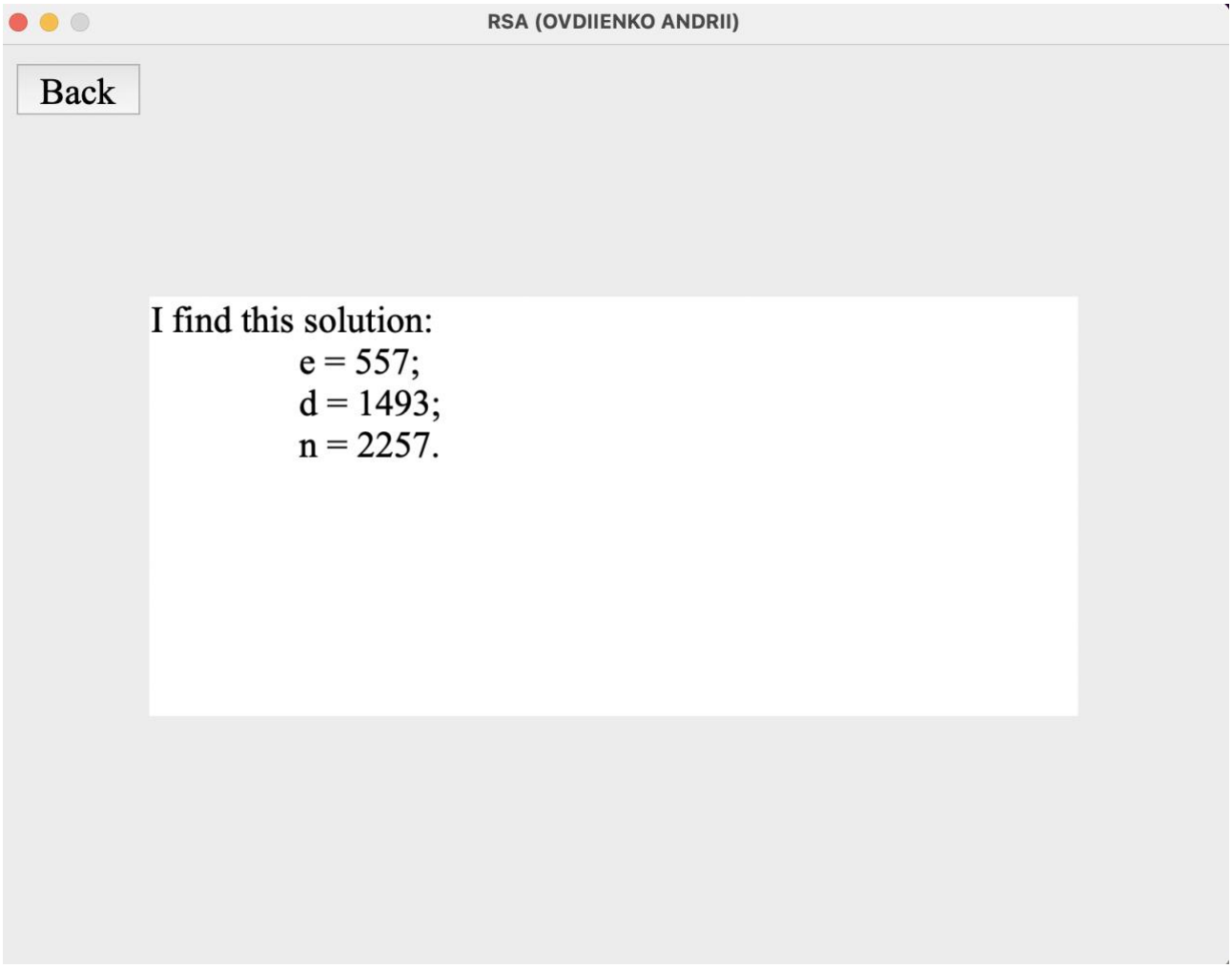

Рисунок 4 – Випадкова генерація валідних параметрів {e, d, n} = {557, 1493, 2257}.

Натиснемо на кнопку для додавання ключа.

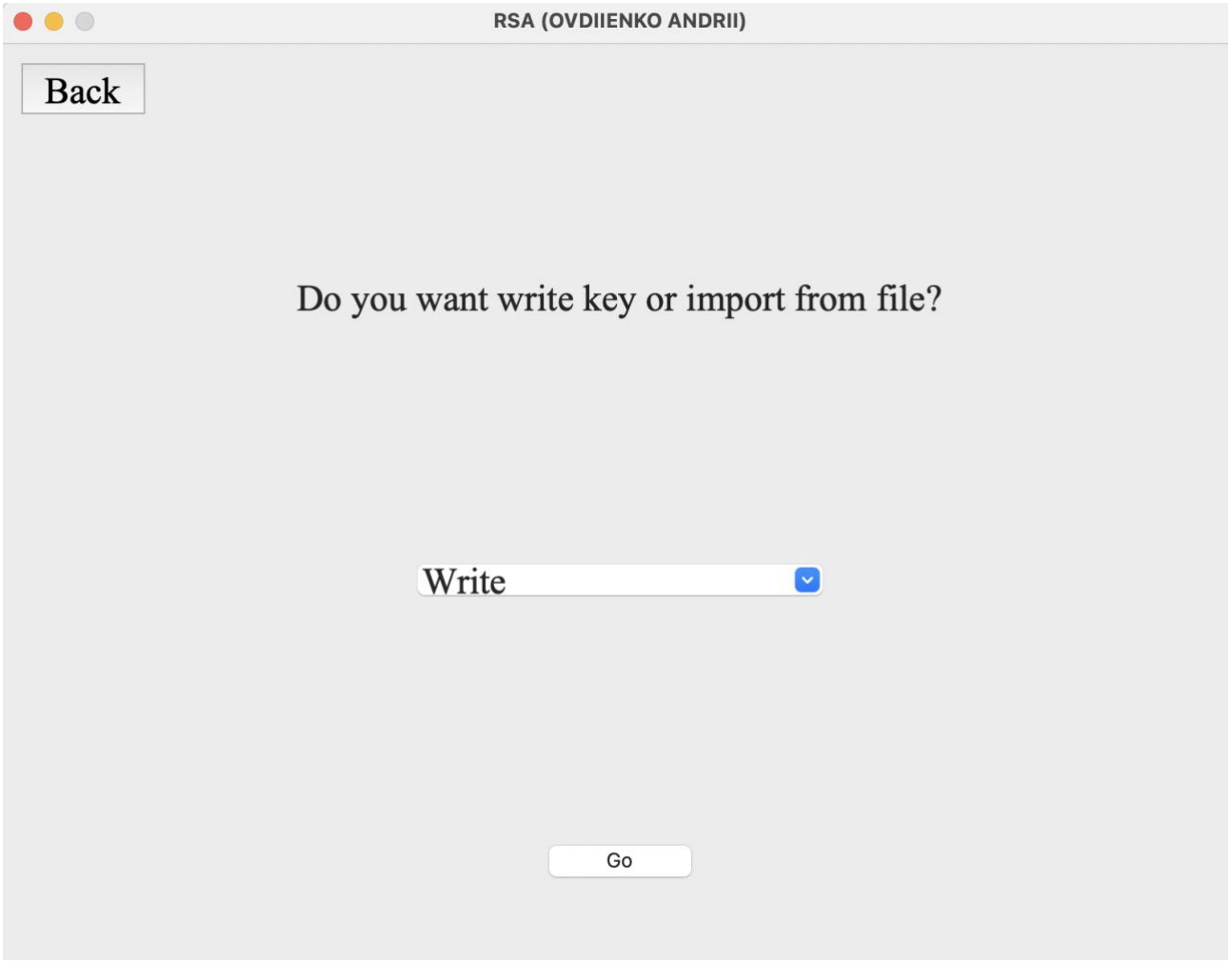

Рисунок 5 – Вибір прописання ключа.

Введемо пару ключів (e, n).

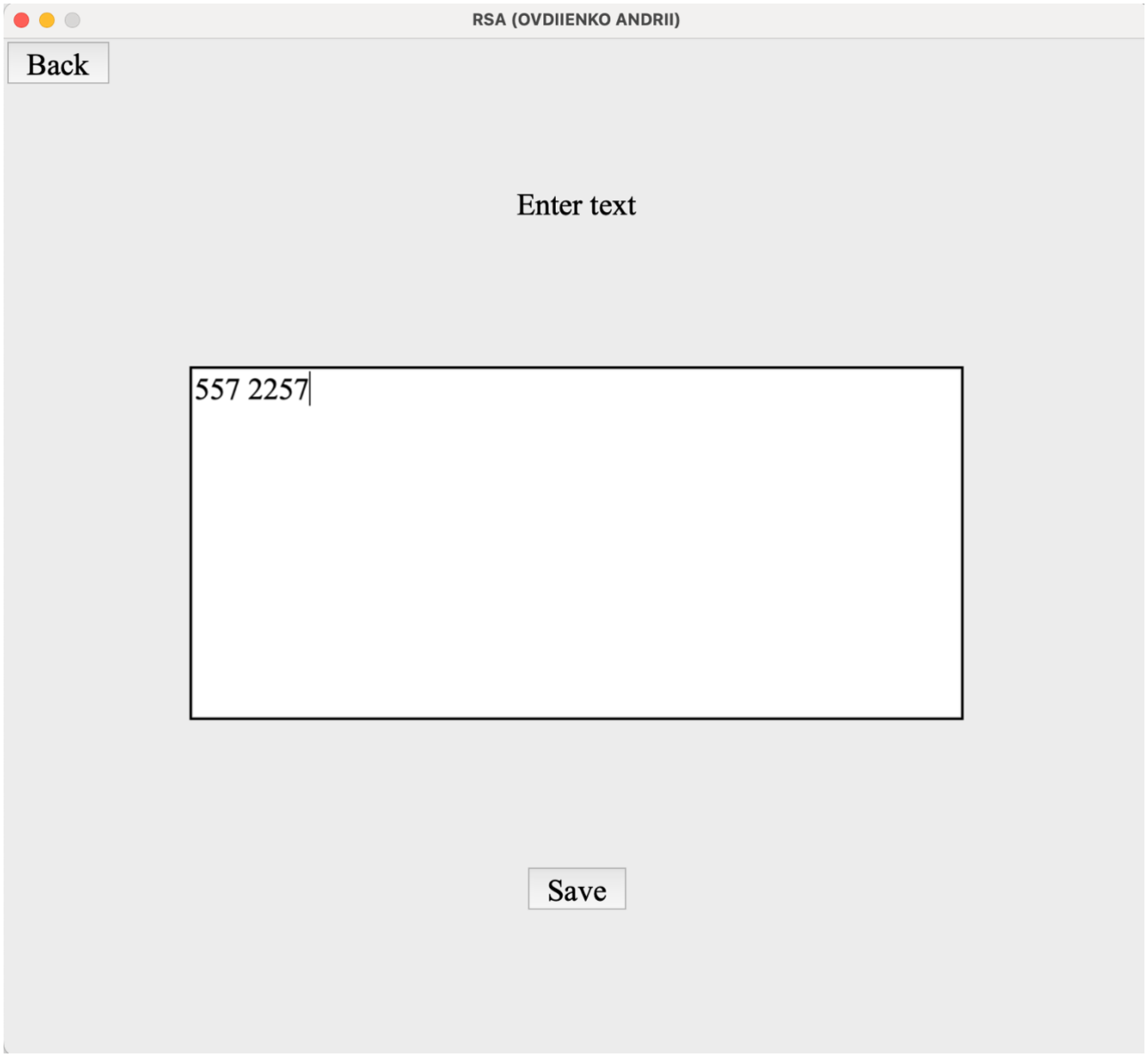

Рисунок 6 – введення параметрів (e , n), які запропонувала система.

Натиснемо на кнопку відображеня текста та пароля.

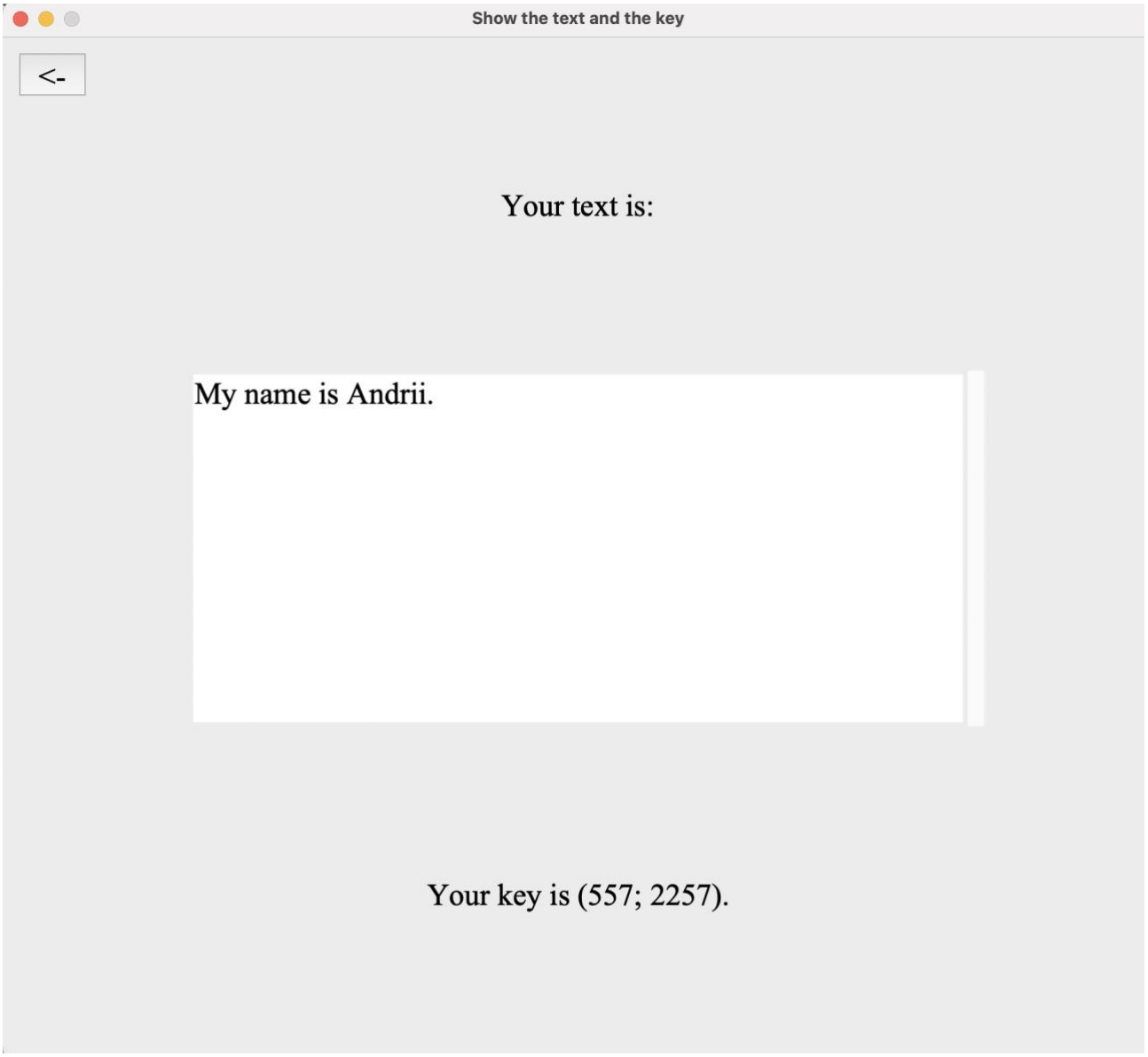

Рисунок 7 – Відображення текста та пароля.

Зашифруємо текст.

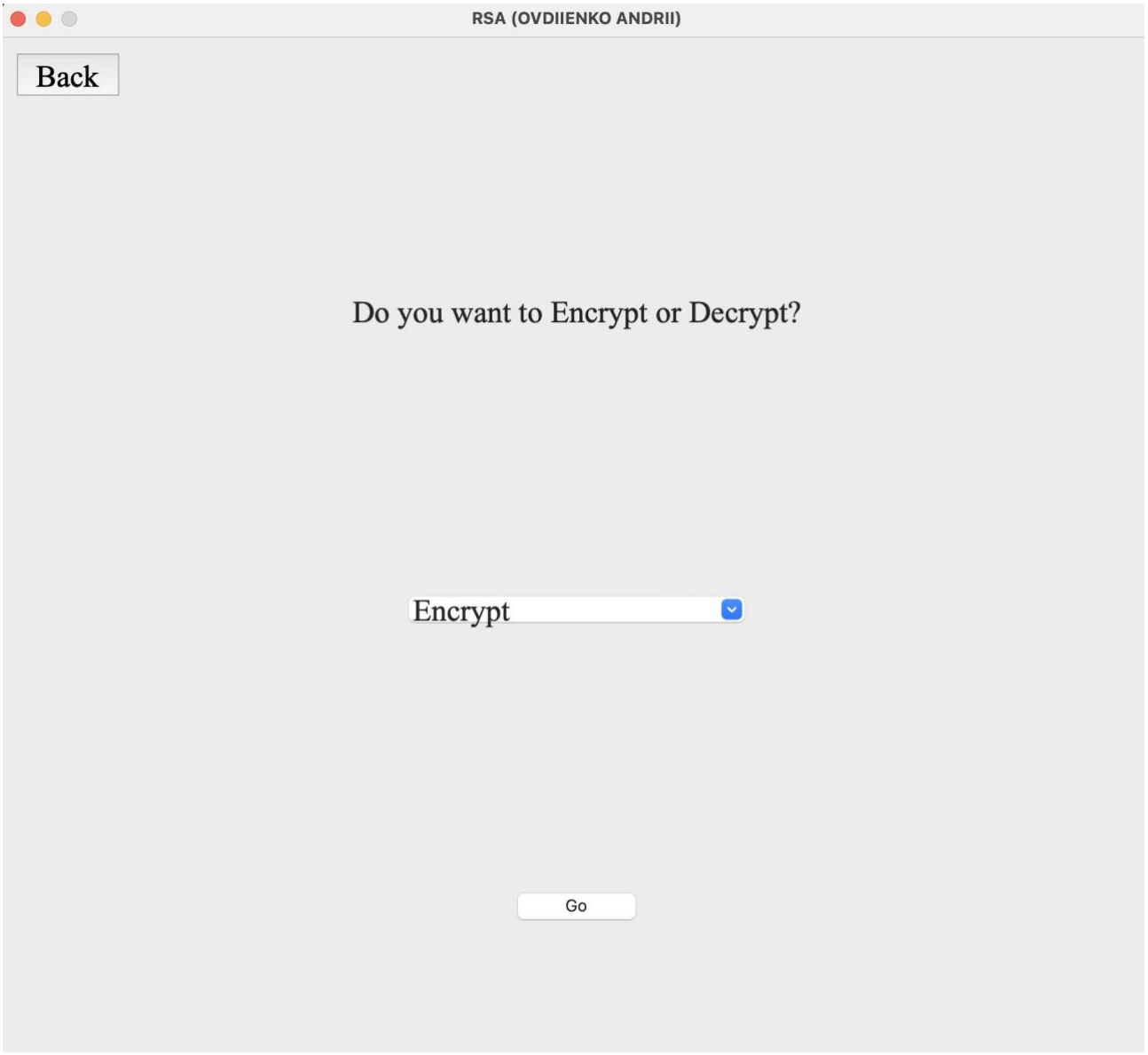

Рисунок 8 – Шифруємо текст.

Натиснемо на кнопку відображеня текста та пароля.

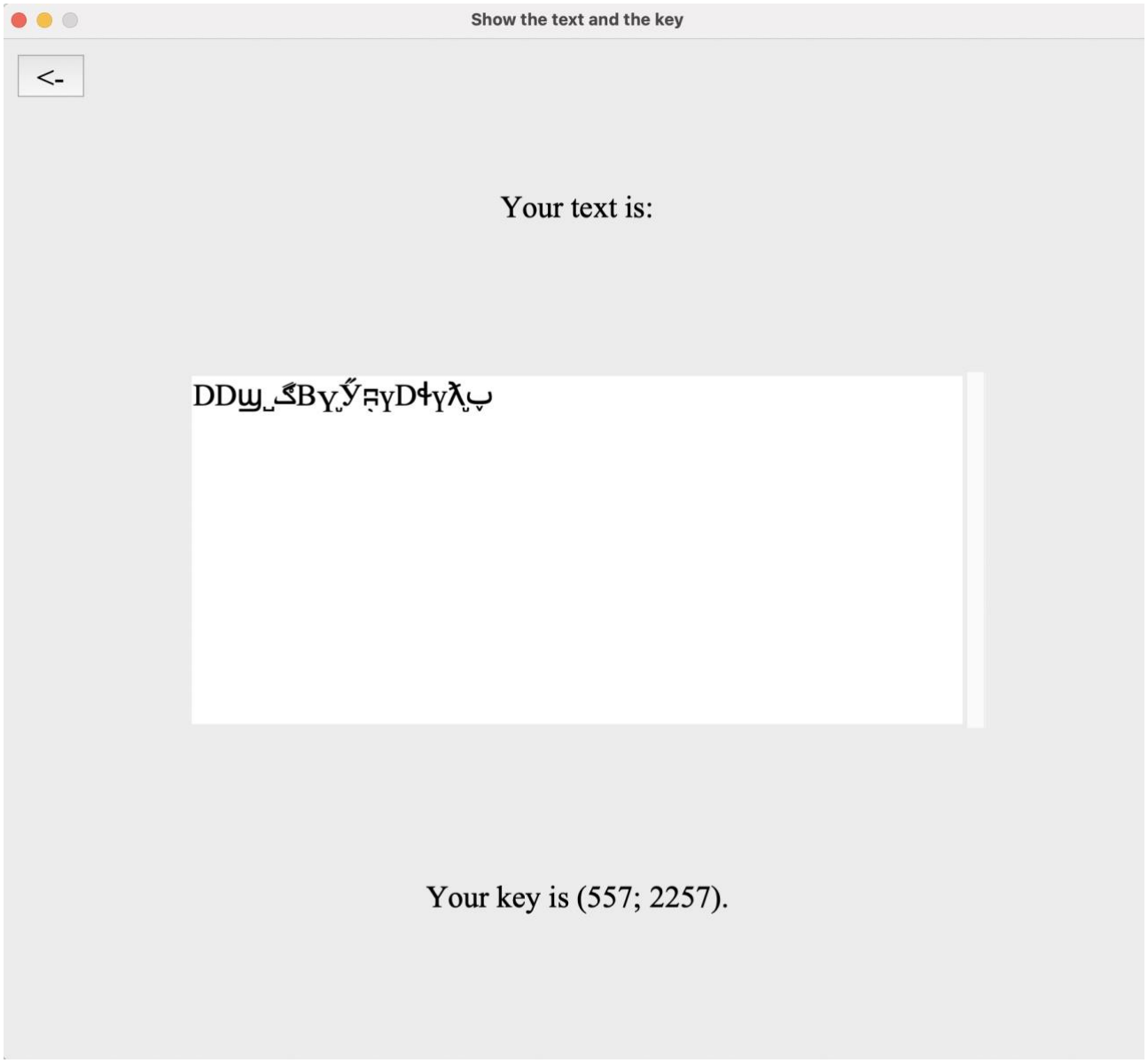

Рисунок 9 – Відображення текста та пароля.

Збережемо текст у файл. Так як брали дані з файла – то отримаємо той самий файл, але з розширенням “_out.txt”.

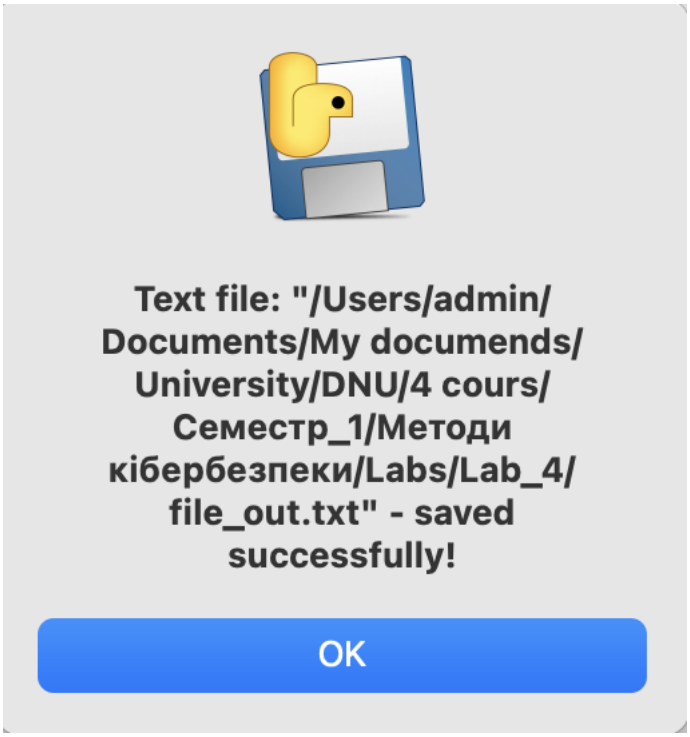

Рисунок 10 – Успішне збереження текста до файла.

А тепер зчитаємо текст з цього файла.

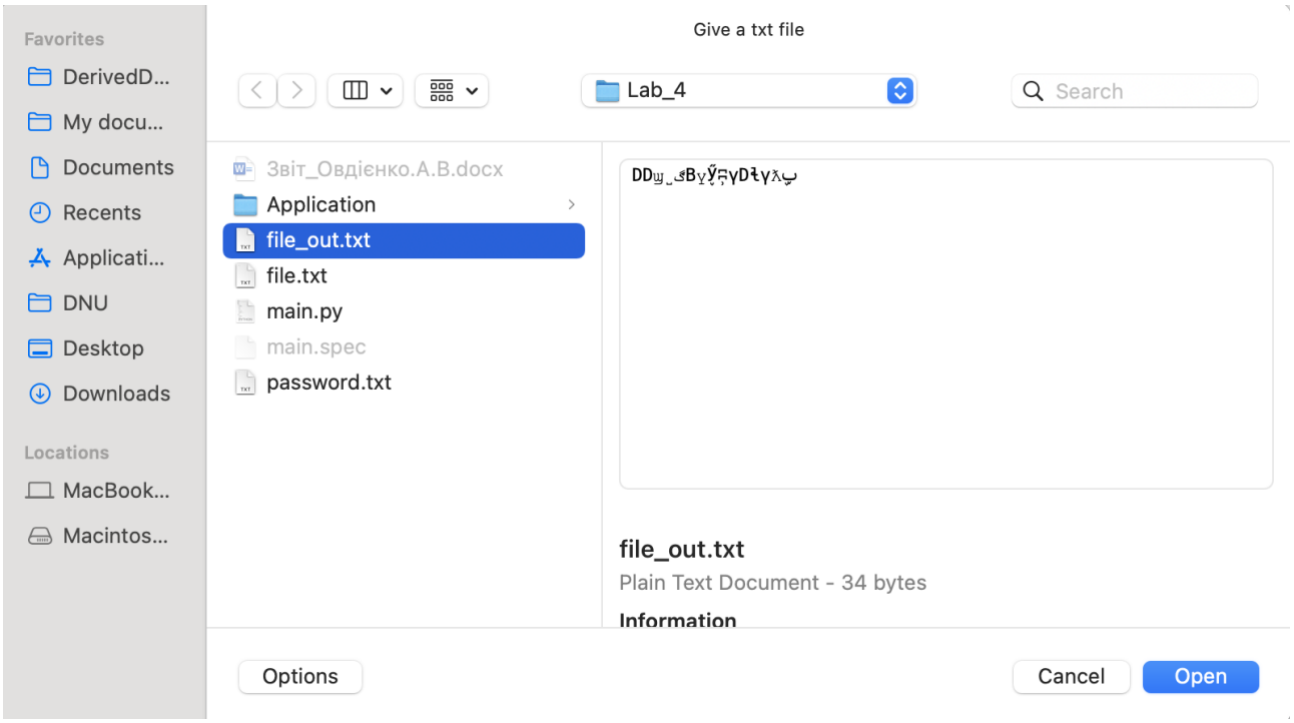

Рисунок 11 – Обираємо файл для зчитування текста.

Введемо ключ відповідно до параметрів, які надала система.

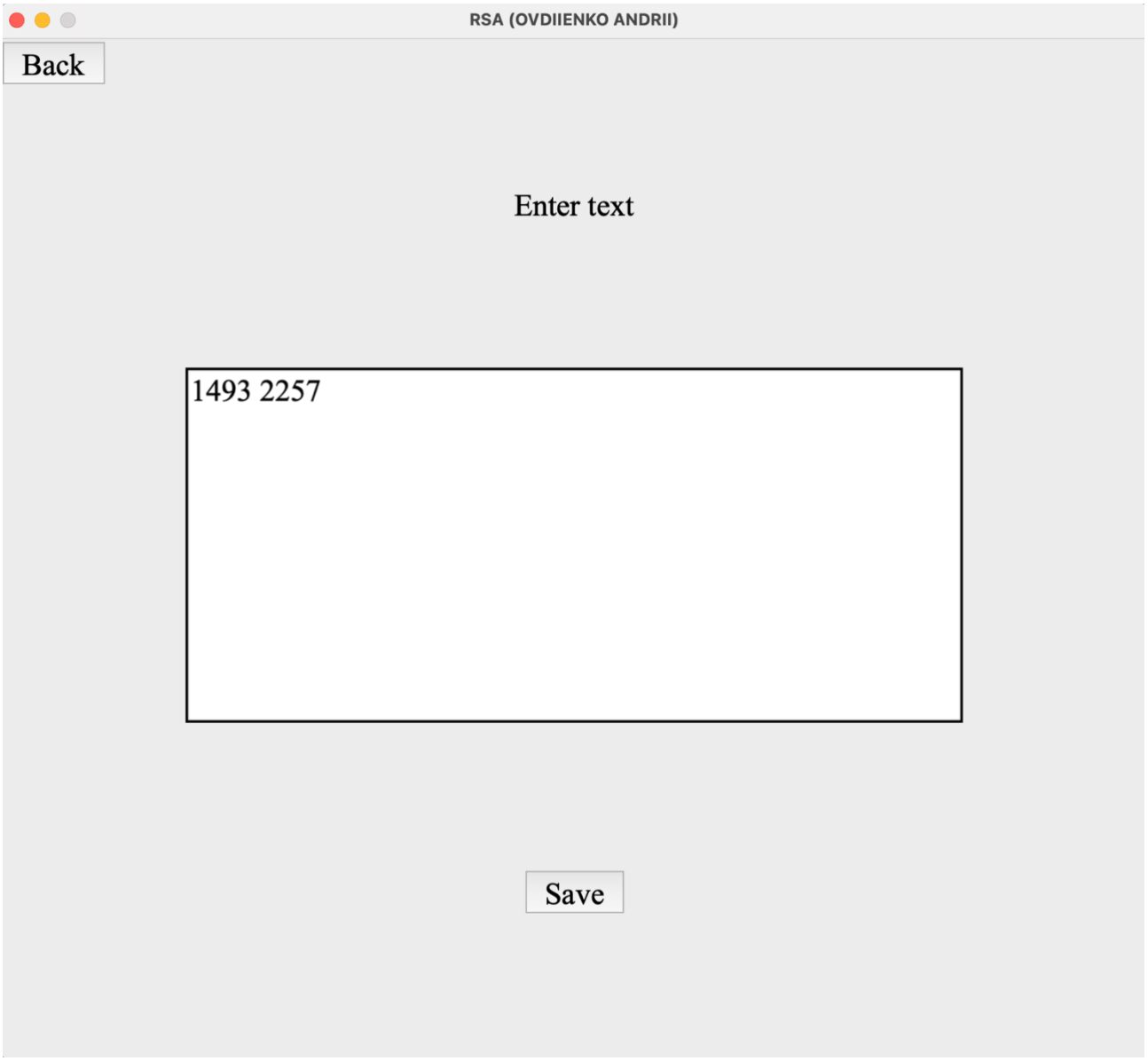

Рисунок 12 – Введення параметрів (d, n) для розшифрування.

Натиснемо на кнопку відображеня текста та пароля.

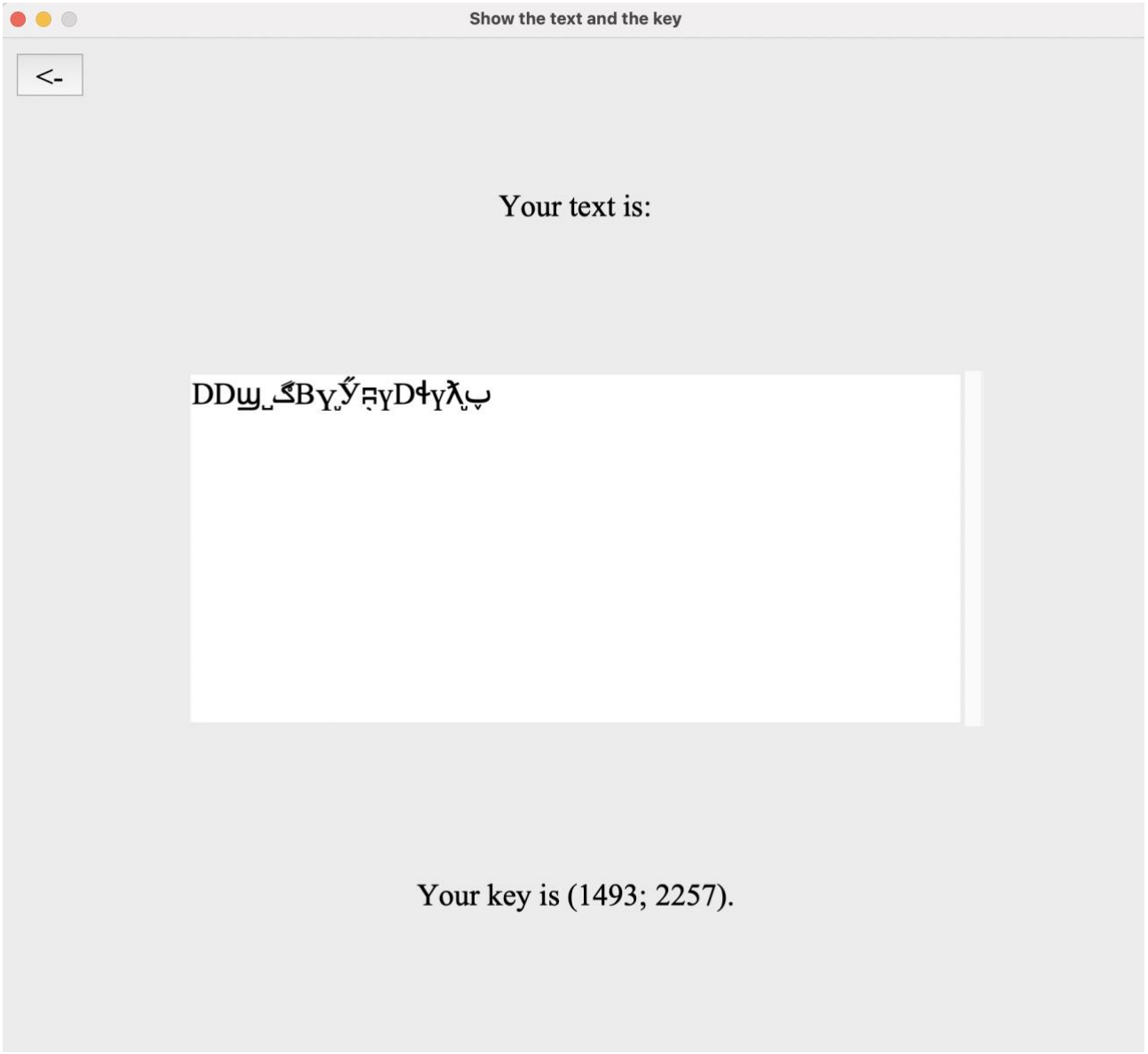

Рисунок 13 – Відображення текста та пароля.

Розшифруємо текст.

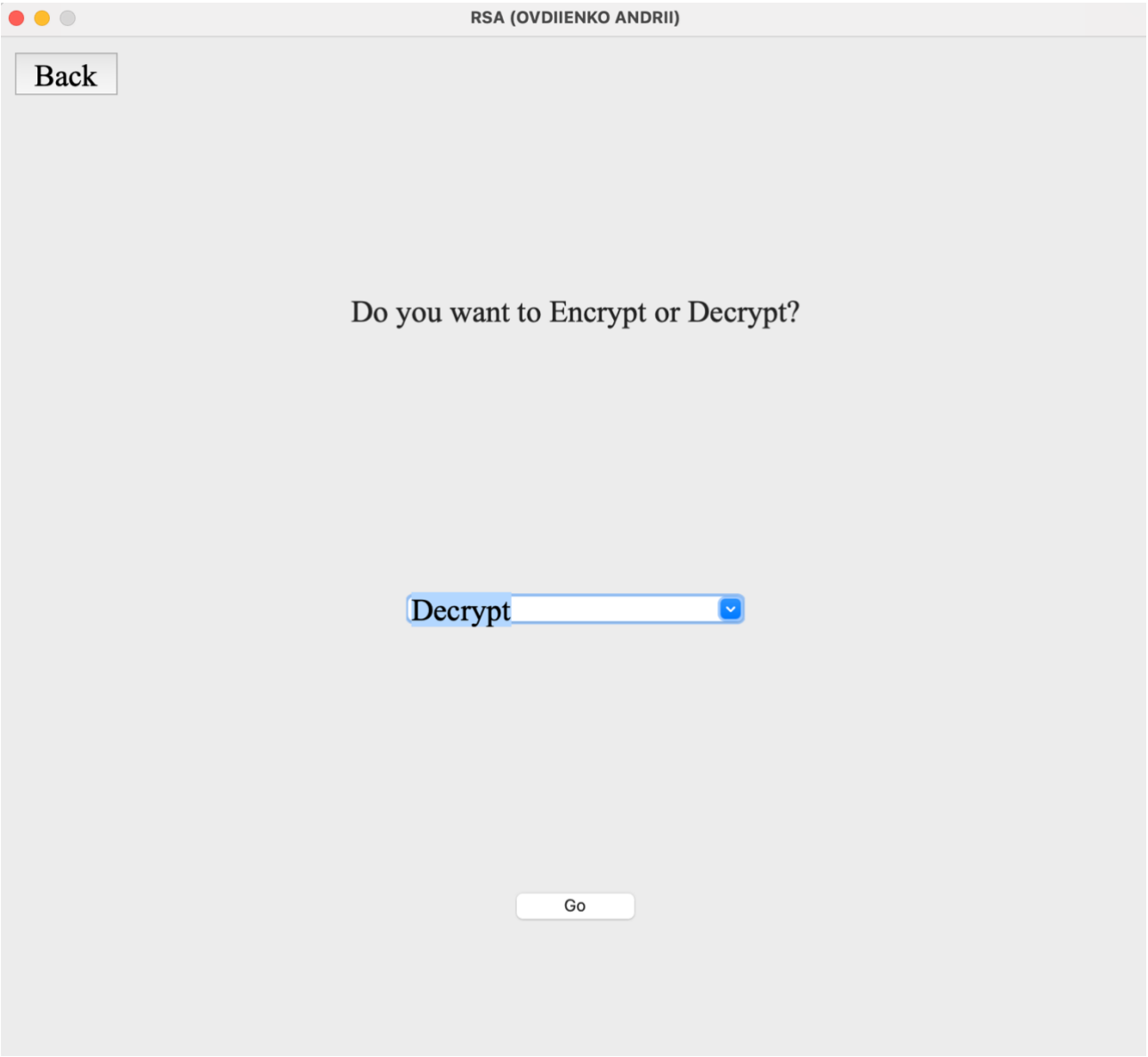

Рисунок 14 – Розшифровуємо текст.

Натиснемо на кнопку відображеня текста та пароля.

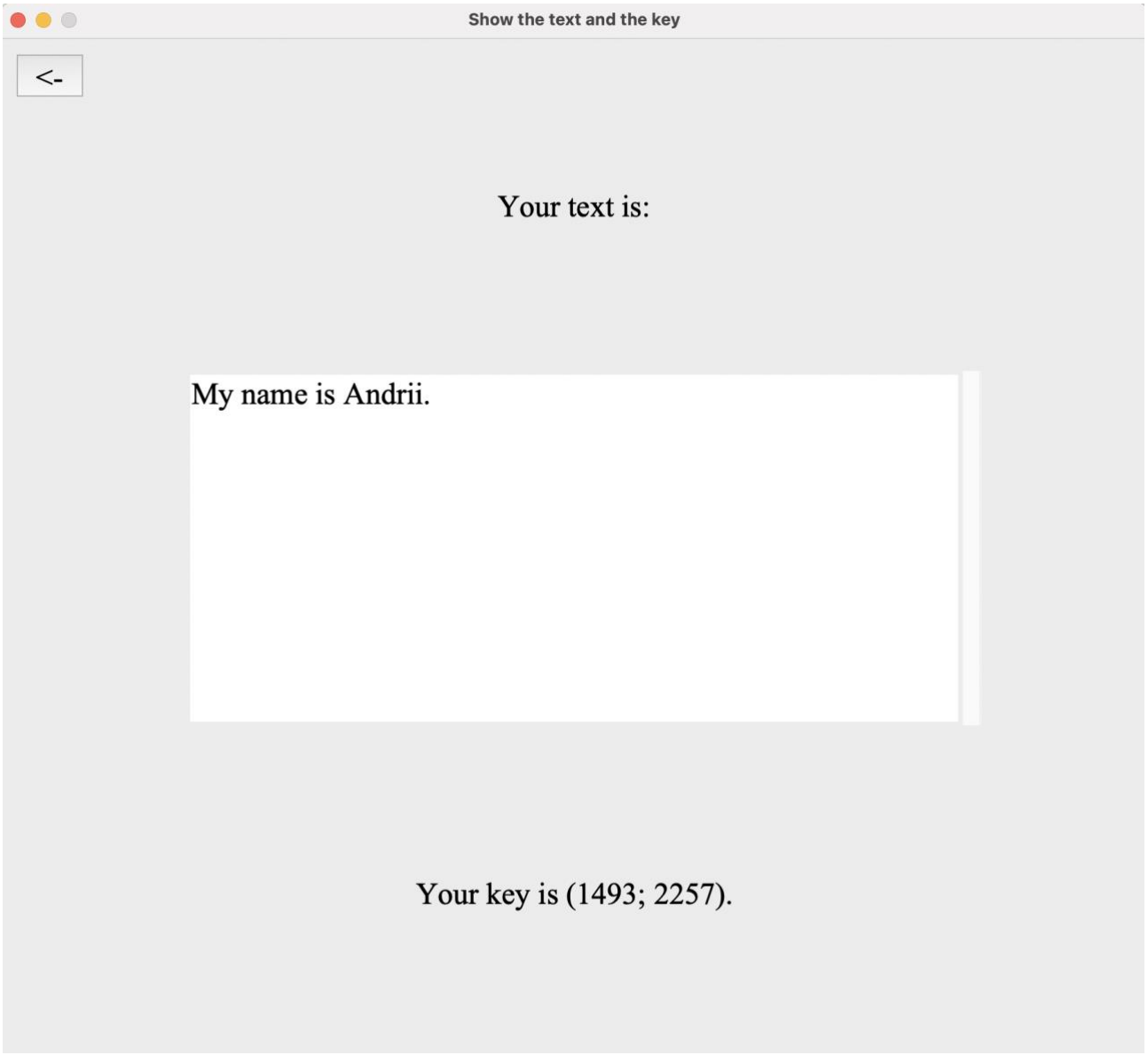

Рисунок 15 – Відображення текста та пароля.

6. Висновки

Вивчено суть асиметричного алгоритма шифрування. Виявлено надзвичайну стійкість, бо ключ для шифрування один, а дешифрування інший. Зрозуміло, що ключ має бути з двох цілочисельних елементів. Реалізовано метод шифрування RSA - а також дешифрування у графічному режимі та можливістю роботи з файлами. Додаток написан та скомпільован як для Mac OS так і Windows. Завантажити додатки та код можна із сайта: https://github.com/OvdiienkoAndrew/RSA.