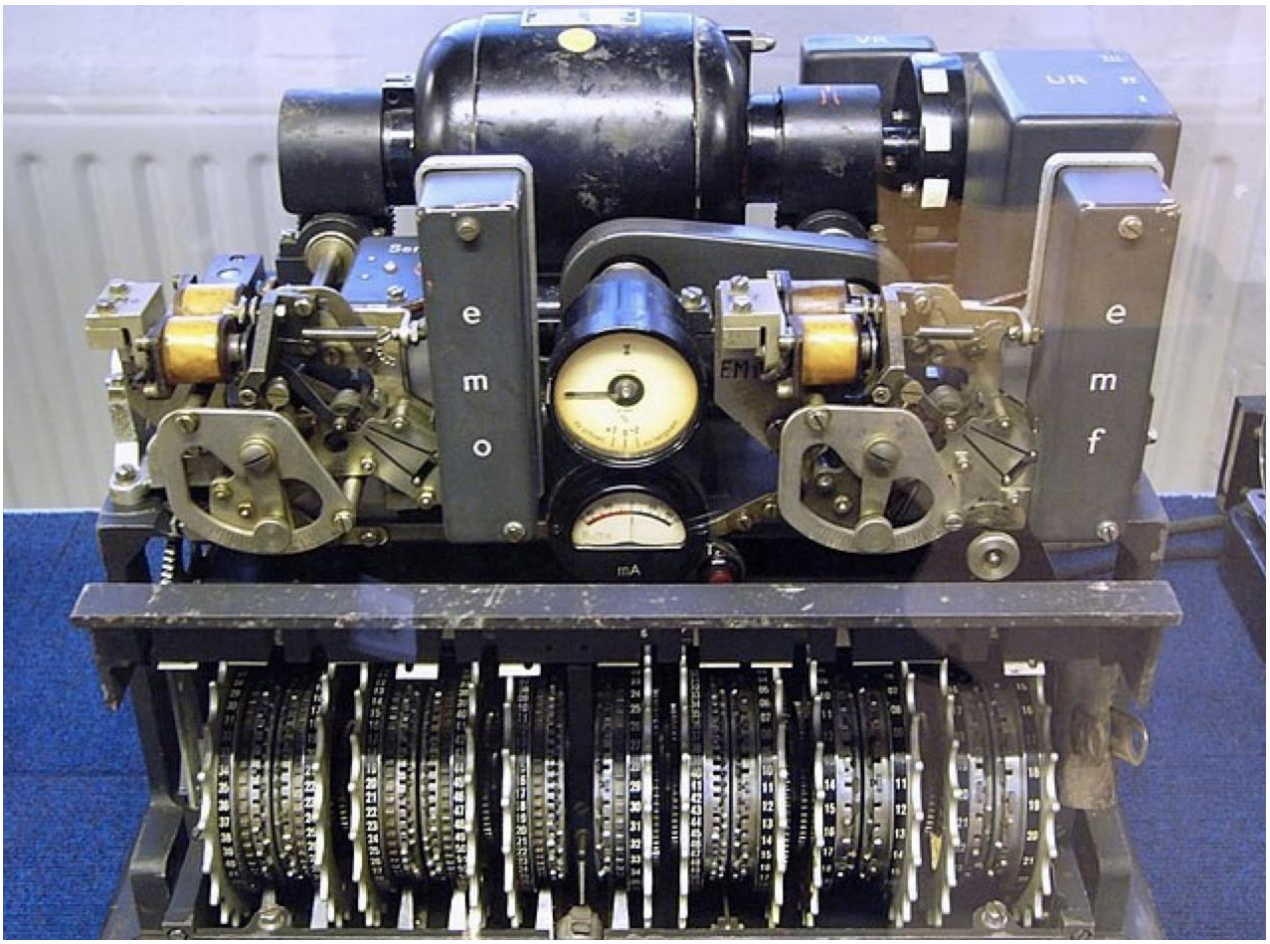
Fachhochschule Nordwestschweiz

One-Time-Pad Dokumentation

Programmierung 2



Autoren

Jazib Chaudhary
WIVZ 1.51
jazib.chaudhary@students.fhnw.ch

Nik von Arx

WIVZ 1.51

nik.vonarx@students.fhnw.ch

Fachhochschule
Nordwestschweiz

Inhalt

Einleitung

Die Sicherheit der Kommunikation ist seit Jahrhunderten ein bedeutendes Thema, das durch die Entwicklung verschiedener Verschlüsselungsmethoden geprägt wurde. Eine dieser Methoden ist das One-Time-Pad-Verfahren, das 1882 von Frank Miller entwickelt wurde. Das Besondere an diesem Verfahren ist seine theoretisch unknackbare Sicherheit, wenn es korrekt angewendet wird.

In unserem Projekt implementiert unsere Gruppe ein System, das dieses Verfahren verwendet, um Nachrichten zu verschlüsseln und zu entschlüsseln. Das System liest einen längeren Text als One-Time-Pad ein und ermöglicht es dem Benutzer, Textnachrichten entweder zu verschlüsseln oder zu entschlüsseln. Der Fokus liegt dabei auf einer einfachen, aber effektiven Anwendung, die die Prinzipien des One-Time-Pad-Verfahrens veranschaulicht und nutzbar macht. Am Anfang wird auf die Anforderungen eingegangen, danach geht es weiter mit dem Praktischem Implementationskonzept. Nach dem Implementationskonzept behandelt dieses Dokument unsere Ergebnisse und zu guter Letzt eine Zusammenfassung.

Anforderung

Zuerst haben wir uns über die funktionalen Anforderungen und über eine grafische Benutzeroberfläche Gedanken gemacht. Nun hatte die erste Phase des Projektes bereits begonnen. Die erste Phase des Projekts bestand darin, die funktionalen Anforderungen zu definieren und eine grafische Benutzeroberfläche (GUI) zu entwickeln, die den Nutzer durch die Anwendung führt. Hier sind die wesentlichen Anforderungen:

1. Benutzeroberfläche (GUI)

- Ein Textfeld zur Eingabe der zu verschlüsselnden oder zu entschlüsselnden Nachricht
- Ein Textfeld zur Anzeige der verschlüsselten oder entschlüsselten Nachricht
- Eine Schaltfläche zur Auswahl der Datei, die als One-Time-Pad dient.
- Schaltflächen für die Verschlüsselung und Entschlüsselung der Nachricht
- Evtl. eine Schaltfläche für einen Reset Button.

2. Verschlüsselungs- und Entschlüsselungslogik

- Implementierung eines Algorithmus, der die Zeichen der Nachricht und des Pads kombiniert, um die Nachricht zu verschlüsseln oder zu entschlüsseln.
- Überprüfung der Zeichen, um sicherzustellen, dass nur erlaubte ASCII-Zeichen verwendet werden.

3. Dateieingabe

- Verwendung des JavaFX FileChooser, um eine Datei auszuwählen, die als One-Time-Pad dient
- Validierung der Datei, um sicherzustellen, dass sie lesbar ist und nur gültige Zeichen enthält

4. Fehlermeldungen

- Anzeige von Fehlermeldungen bei ungültigen Eingaben oder Dateifehler

Praktisches Implementationskonzept

Das One-Time-Pad-Verschlüsselungssystem wurde in Java entwickelt und verwendet JavaFX für die grafische Benutzeroberfläche (GUI). Das System besteht aus mehreren Klassen, die zusammenarbeiten, um die Ver- und Entschlüsselung von Nachrichten mit einem One-Time-Pad zu ermöglichen. Das UML-Diagramm und die folgende Beschreibung geben einen Überblick über die Struktur und das Zusammenspiel der einzelnen Komponenten des Systems.

Komponentenbeschreibung

1. OTPMain

Die OTPMain-Klasse ist der Einstiegspunkt der Anwendung. Sie startet die JavaFX-Anwendung, indem sie die OTPView-Klasse aufruft und die GUI initialisiert.

2. OTPView

Die OTPView-Klasse erweitert Application und ist für die Initialisierung der Stage und der GUI verantwortlich. Sie erstellt die GUI-Komponenten, wie Textfelder und Buttons, und fügt diese in die Benutzeroberfläche ein.

3. OTPController

Der OTPController verwaltet die Benutzerinteraktionen und kommuniziert mit dem Modell, um die Verschlüsselungs- und Entschlüsselungslogik auszuführen. Die Hauptmethoden sind:

encrypt(ActionEvent e): Ruft die Verschlüsselungslogik auf und zeigt das Ergebnis an.

decrypt(ActionEvent e): Ruft die Entschlüsselungslogik auf und zeigt das Ergebnis an.

loadPad(ActionEvent e): Ermöglicht dem Benutzer, ein One-Time-Pad aus einer Datei zu laden.

reset(): Setzt die Eingabefelder zurück.

4. OTPModel

Das OTPModel enthält die Logik für die Verschlüsselung und Entschlüsselung von Nachrichten. Es stellt folgende Methoden zur Verfügung:

encryptMessage(String message): Verschlüsselt die Nachricht mithilfe des One-Time-Pads.

decryptMessage(String message): Entschlüsselt die Nachricht mithilfe des One-Time-Pads.

setPad(String newPad): Setzt das One-Time-Pad.

validateMessage(String message): Überprüft die Eingabe auf gültige Zeichen und ausreichende Länge des Pads.

normalizeInput(String input): Normalisiert die Eingabe, um Sonderzeichen wie "ä", "ö" und "ü" zu entfernen.

processMessage(String message, boolean encrypt): Führt die eigentliche Verschlüsselung oder Entschlüsselung durch.

Code-Beispiel: Verschlüsselung

Hier ein Ausschnitt aus der OTPModel-Klasse, der die Verschlüsselungslogik zeigt:

```
public class OTPModel {
    3 usages
    private String pad = ""; // Initialisieren mit leerem String
    4 usages
    private int padPosition = 0; // Position im Pad tracken

    1 usage  NIIKKcoding
    public void setPad(String newPad) {
        pad = newPad;
        padPosition = 0; // Position zurücksetzen, wenn ein neues Pad geladen wird
    }

    1 usage  NIIKKcoding
    public String encryptMessage(String message) {
        message = normalizeInput(message);
        validateMessage(message);
        return processMessage(message,  encrypt: true);
    }

    1 usage  NIIKKcoding
    public String decryptMessage(String message) {
        message = normalizeInput(message);
        validateMessage(message);
        return processMessage(message,  encrypt: false);
    }

    2 usages  NIIKKcoding
    private String normalizeInput(String input) {

        return input.replaceAll( regex: "ä",  replacement: "ae")
                .replaceAll( regex: "ö",  replacement: "oe")
                .replaceAll( regex: "ü",  replacement: "ue");
    }
    private void validateMessage(String message) {

        if (!message.chars().allMatch(c -> c >= 0x20 && c <= 0x7E) || padPosition + message.length() > pad.length()) {
            throw new IllegalArgumentException("Invalid input or insufficient pad length.");
        }
    }

    2 usages  NIIKKcoding
    private String processMessage(String message, boolean encrypt) {

        StringBuilder processed = new StringBuilder();
        for (int i = 0; i < message.length(); i++) {
            int messageVal = message.charAt(i) - 0x20;
            int padVal = pad.charAt(padPosition + i) - 0x20;
            int resultVal = encrypt ? (messageVal + padVal) % 95 : (messageVal - padVal + 95) % 95;
            processed.append((char) (resultVal + 0x20));
        }
        padPosition += message.length();
        return processed.toString();
    }
}
```

Abbildung 1 Klasse OTPModel

Interaktion der Komponenten

Benutzerinteraktion: Der Benutzer gibt eine Nachricht ein und wählt ein One-Time-Pad aus.

Eingabevalidierung: Der OTPController ruft validateMessage() im OTPModel auf, um sicherzustellen, dass die Eingabe gültig ist.

Verschlüsselung/Entschlüsselung: Der OTPController ruft encryptMessage() oder decryptMessage() im OTPModel auf, um die Nachricht zu verarbeiten.

Ergebnisanzeige: Der OTPController zeigt das verschlüsselte oder entschlüsselte Ergebnis in der GUI an.

UML-Diagramm:

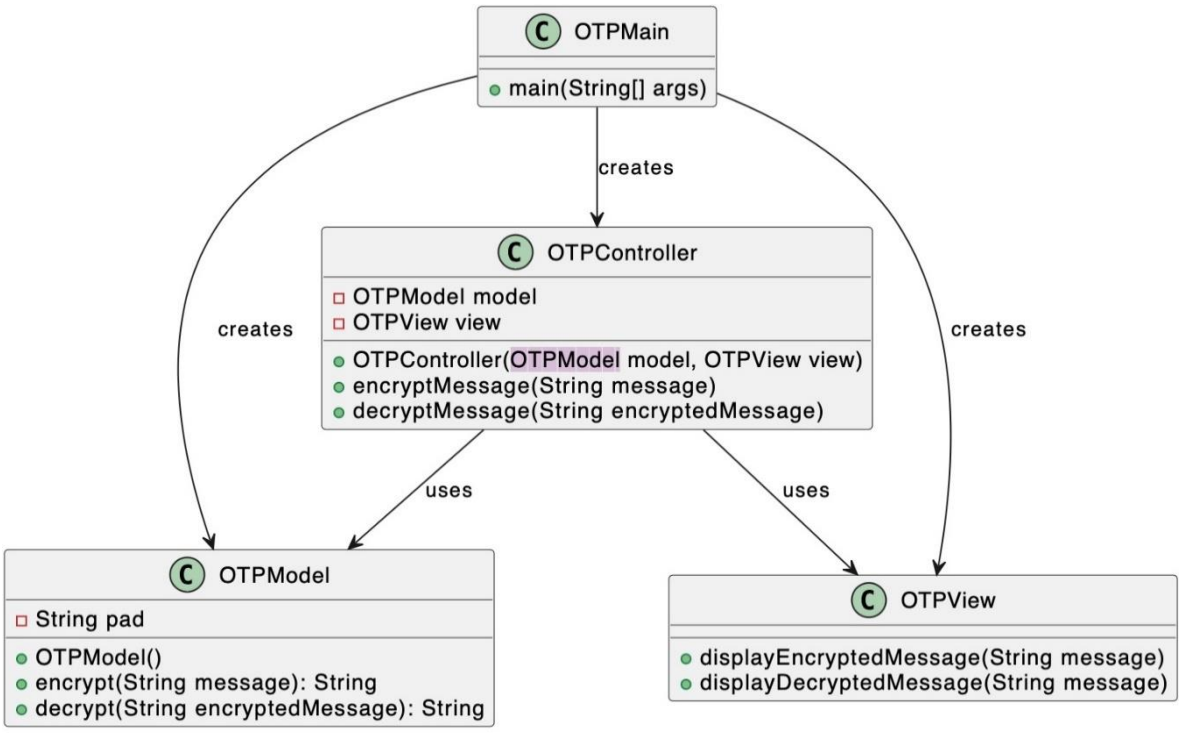


Abbildung 2 UML-Diagramm der Anwendung

Ergebnisse

Das Hauptziel des Projekts, die Entwicklung eines funktionsfähigen One-Time-Pad-Verschlüsselungssystems, wurde erreicht. Die Anwendung ermöglicht die Eingabe und Verarbeitung von Nachrichten mit einem eingelesenen One-Time-Pad. Die Ergebnisse im Einzelnen

1. Funktionalität:

- **Verschlüsselung und Entschlüsselung:** Die Implementierung der Ver- und Entschlüsselungsmechanismen funktioniert wie vorgesehen. Nachrichten werden entsprechend den Projektanforderungen korrekt ver- und entschlüsselt. Dies konnte

durch mehrere Tests verifiziert werden, bei denen verschlüsselte Nachrichten erfolgreich in den ursprünglichen Text zurückverwandelt wurden.

- **Benutzeroberfläche:** Die Benutzeroberfläche (GUI) ist intuitiv und benutzerfreundlich gestaltet. Benutzer können leicht zwischen den Optionen zum Verschlüsseln und Entschlüsseln wechseln und sowohl die Nachricht als auch das One-Time-Pad über einfache Eingabefelder eingeben.

- **Fehlermeldungen:** Bei ungültigen Eingaben oder Dateifehlern zeigt das System aussagekräftige und verständliche Fehlermeldungen an. Diese helfen den Benutzern, Fehler schnell zu erkennen und zu korrigieren.

2. Einschränkungen:

- **ASCII-Zeichen:** Der implementierte Algorithmus unterstützt nur ASCII-Zeichen. Dies bedeutet, dass internationale Zeichen oder Sonderzeichen ausserhalb des ASCII-Bereichs nicht verschlüsselt werden können.

- **Pad-Verwaltung:** Die Verwaltung des One-Time-Pads muss manuell erfolgen. Die Benutzer müssen das Pad selbst auswählen und sicherstellen, dass es gross genug ist, um die gesamte Nachricht zu verschlüsseln. Dies kann bei längeren Nachrichten mühsam sein. Eine mögliche Verbesserung wäre die Implementierung einer automatischen Pad-Verwaltung oder Generierung von Pads.

3. Zusatzfunktionalitäten:

- **Dateieingabe:** Eine wichtige Funktion, die erfolgreich implementiert wurde, ist das Einlesen des One-Time-Pads aus einer Datei. Die Benutzer können eine Textdatei als Pad auswählen, was die Flexibilität und Benutzerfreundlichkeit der Anwendung erhöht.

- **Fehlerbehandlung:** Verbesserte Fehlermeldungen und Eingabevalidierung sorgen für eine robustere Anwendung.

4. Screenshots:

Hauptbildschirm:

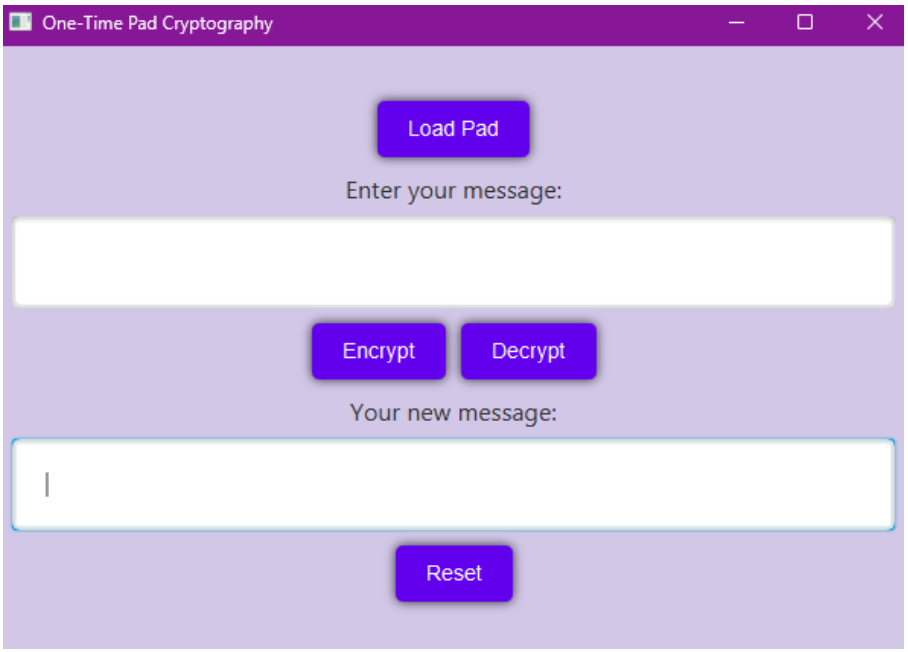


Abbildung 3 Screenshot Hauptbildschirm

Dateiauswahl:

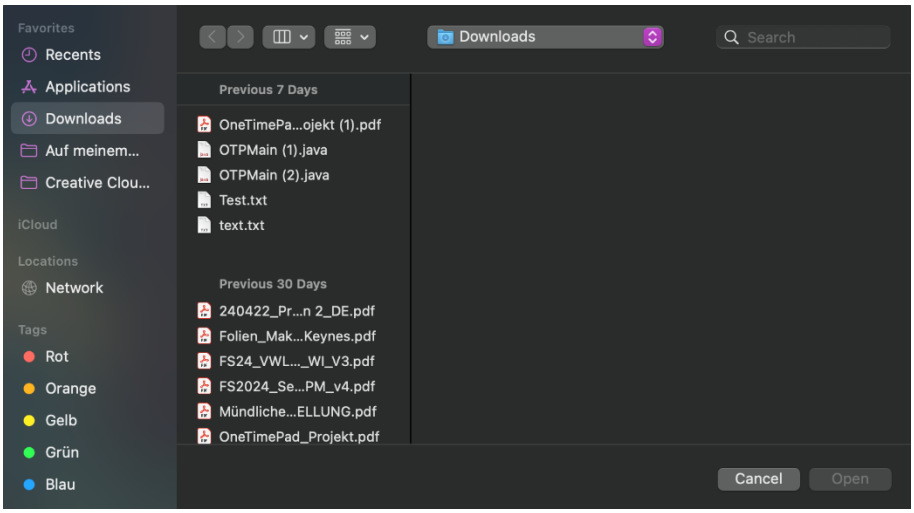


Abbildung 4 Screenshot Dateiauswahl

Verschlüsselung einer Nachricht:

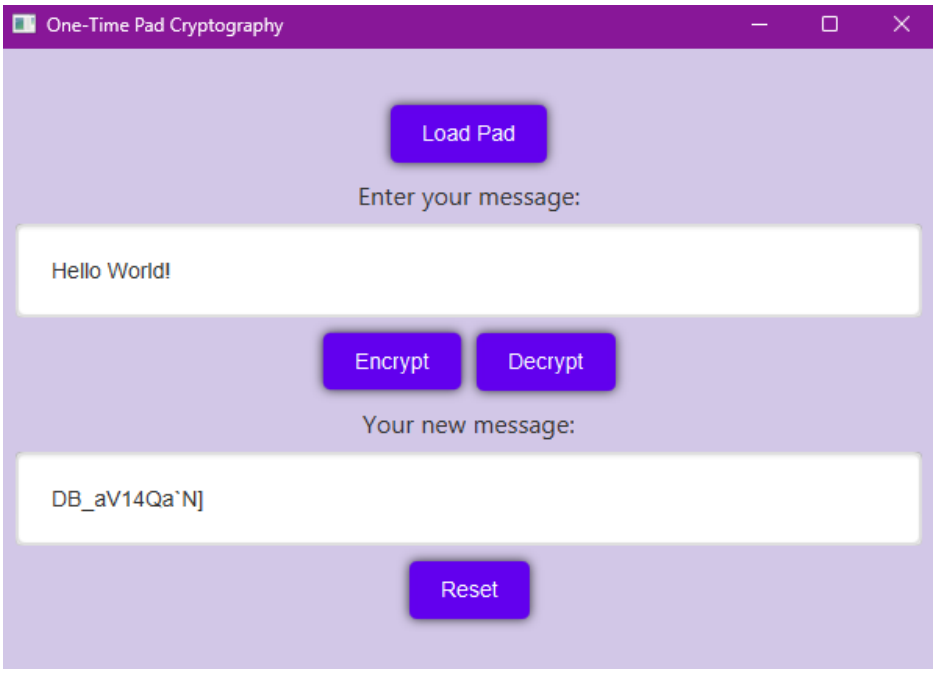


Abbildung 5 Screenshot Verschlüsselung einer Nachricht

Entschlüsselung einer Nachricht:

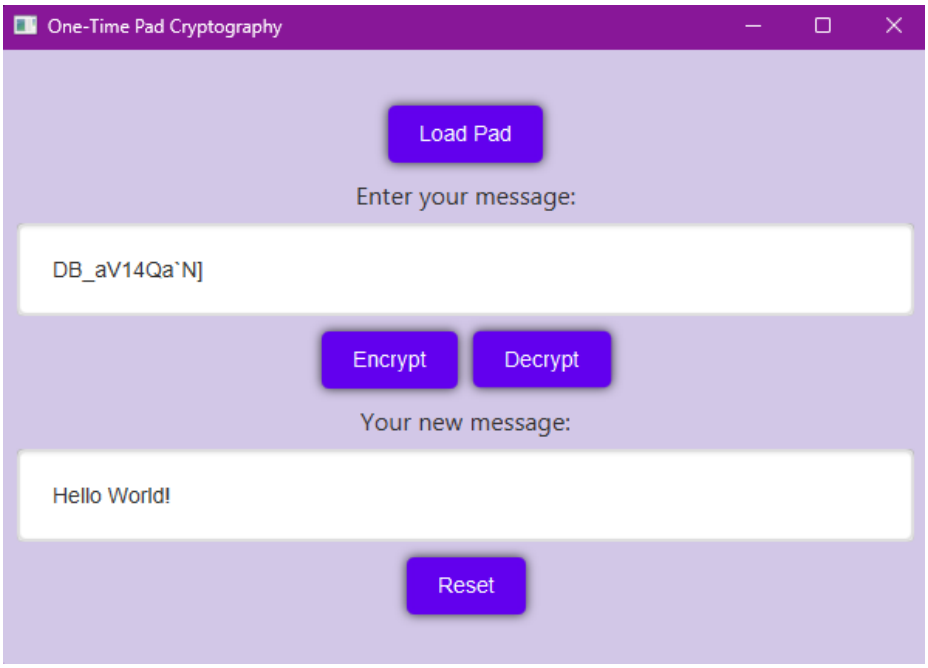


Abbildung 6 Screenshot Entschlüsselung einer Nachricht

Zusammenfassung

Das One-Time-Pad-Verfahren bietet eine theoretisch unknackbare Verschlüsselungsmethode, die durch unsere Implementierung effektiv demonstriert wird. Die wichtigsten Erkenntnisse aus diesem Projekt sind

- **Sicherheit durch Einfachheit:** Die Stärke des One-Time-Pad Verfahrens liegt in seiner Einfachheit und der Verwendung eines ausreichend grossen und zufällig generierten Pads.

- **Benutzerfreundlichkeit:** Eine intuitive Benutzeroberfläche ist entscheidend für die Benutzerfreundlichkeit der Anwendung.

- **Fehlerbehandlung:** Robuste Fehlerbehandlung und Eingabevalidierung sind für die Zuverlässigkeit der Anwendung unerlässlich.

Weiterführende Untersuchungen:

- **Unterstützung für internationale Zeichen:** Erweiterung des Algorithmus, um auch internationale Zeichen zu verschlüsseln.

- **Langfristige Pad-Verwaltung:** Implementierung einer Funktion zur Speicherung des Pads und der aktuellen Position, um eine nahtlose Fortsetzung der Verschlüsselung zu ermöglichen.

Quellen und Literatur

Abbildungen:

Literatur:

Barry B. (2014), *Java Programmieren lernen für Dummies*. (1. Aufl.). Wiles-VCH GmbH & Co. KGaA

Riesen K. (2020). *Java in 14 Wochen: Ein Lehrbuch für Studierende der Wirtschaftsinformatik*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-30313-6

https://moodle.fhnw.ch/course/view.php?id=57896