Fachhochschule
Nordwestschweiz

Endbericht

Perfekte Kryptographie

Programmierung 2 HS 23

Autor

Cédric Wespi

Rüeggisingerstrasse 18C

6020 Emmenbrücke

Inhalt

Einleitung

Das Projekt "Perfekte Kryptografie - Eine Anwendung von One-Time-Pads" verfolgt das Ziel, das historische und theoretisch sichere One-Time-Pad-Verschlüsselungsverfahren in einer benutzerfreundlichen Softwareanwendung umzusetzen. Dieses Verfahren, das auf der Arbeit des Kryptologen Frank Miller aus dem Jahr 1882 basiert, gilt auch heute noch als sicher, wenn es korrekt angewendet wird. Unsere Anwendung demonstriert die Umsetzung dieses Prinzips in einem Java-Code.

Die Kernfunktion der Anwendung besteht darin, eine vom Benutzer eingegebene Nachricht mit einem One-Time-Pad zu verschlüsseln bzw. zu entschlüsseln. Im Gegensatz zu modernen Anwendungen, die typischerweise Zufallszahlen als One-Time-Pads verwenden, greift unser Projekt auf eine ältere Idee zurück: Das Einlesen eines längeren Textes, der als Pad verwendet wird.

Die Umsetzung erfolgte in drei Phasen, beginnend mit der Planung und Entwicklung einer Benutzeroberfläche unter Verwendung von JavaFX, über die Implementierung der Verschlüsselungs- und Entschlüsselungsfunktionen, bis hin zur Integration der Möglichkeit, One-Time-Pads aus Dateien einzulesen. Besondere Aufmerksamkeit wurde auf die Benutzerfreundlichkeit und die Gestaltung der Anwendung gelegt

Insgesamt stellt das Projekt eine Demonstration eines klassischen Kryptografieverfahrens dar, und die Herausforderungen, die mit ihrer Implementierung verbunden sind.

Theoretische Grundlagen

Phase 1: Anforderungen und GUI Entwicklung

In der ersten Phase des Projekts stand die Planung der Funktionalität und das Design der Benutzeroberfläche im Vordergrund. Entscheidend war die Wahl von JavaFX als Framework für die GUI, da es eine hohe Flexibilität in der Gestaltung und eine einfache Integration von Event-Handling-Mechanismen bietet. JavaFX ermöglichte es uns, eine ansprechende und intuitive Benutzeroberfläche zu erstellen, die für den Benutzer leicht zu navigieren ist. Hierbei wurden klassische GUI-Elemente wie Buttons, Textfelder und Labels verwendet, die eine klare und benutzerfreundliche Interaktion mit der Anwendung ermöglichen.

Phase 2: Verschlüsselung und Entschlüsselung

Die zweite Phase konzentrierte sich auf die Implementierung der Verschlüsselungs- und Entschlüsselungslogik. Die Kernentscheidung hierbei war die Verwendung eines einfachen, aber effektiven Algorithmus, der auf ASCII-Zeichen beschränkt ist. Dieser Ansatz, der in der `Pad_Model`-Klasse implementiert wurde, ermöglichte eine unkomplizierte, aber sichere Verschlüsselung von Nachrichten. Der Algorithmus verwendet eine einfache Addition und Subtraktion von ASCII-Werten, um Zeichen zu verschlüsseln und zu entschlüsseln, und folgt damit dem Prinzip der Einfachheit, während er gleichzeitig eine hohe Sicherheit gewährleistet.

Die Entscheidung, ASCII-Zeichen zu verwenden, basierte auf ihrer universellen Anwendbarkeit und der leichten Handhabung in Java. Zudem ermöglichte diese Einschränkung eine vereinfachte Überprüfung der Eingabedaten, wie in der `isValidInput`-Methode implementiert, und stellte sicher, dass nur gültige Daten für die Verschlüsselungsoperationen verwendet werden.

Phase 3: One-Time-Pad von einer Datei einlesen

In der dritten Phase wurde die Funktionalität hinzugefügt, dass Benutzer eigene One-Time-Pads aus Dateien laden können. Diese Funktionalität erweiterte die Anwendung erheblich, da sie es Benutzern ermöglicht, ihre eigenen Schlüssel für die Verschlüsselung zu verwenden, was die Sicherheit und Vielseitigkeit der Anwendung erhöht. Die Herausforderung bestand darin, sicherzustellen, dass die geladenen Dateien korrekte ASCII-Zeichen enthalten und eine angemessene Länge für die Verschlüsselung bieten. Der FileChooser war ein Teil der Anforderung an das Projekt.

Die Möglichkeit, Dateien als One-Time-Pads zu verwenden erweiterte die Applikation, da Benutzer nun nicht mehr auf vordefinierte Pads beschränkt sind.

Gesamtbetrachtung

Jede Phase des Projekts brachte wichtige Entscheidungen mit sich, die auf dem Ziel basierten, eine sichere, benutzerfreundliche und flexible Anwendung zu schaffen. Die Entscheidungen hinsichtlich der GUI-Gestaltung, des Verschlüsselungsalgorithmus und der Integration von benutzerdefinierten One-Time-Pads wurden getroffen, um die Kernanforderungen des Projekts zu erfüllen: Sicherheit, Benutzerfreundlichkeit und Anpassungsfähigkeit. Diese Entscheidungen formten eine Anwendung, die sowohl als Bildungswerkzeug als auch als praktisches Tool für sichere Kommunikation dienen kann.

Praktisches Implementationskonzept

Überblick

Die Anwendung basiert auf einer MVC (Model-View-Controller) Architektur, die eine klare Trennung ermöglicht.

Simples UML – Diagramm

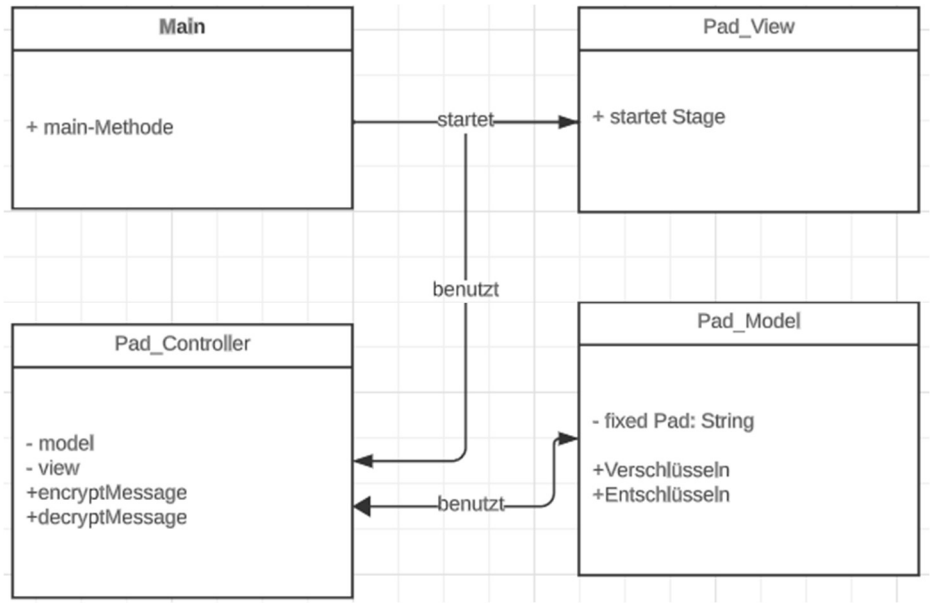

Abbildung 1 – UML – Diagramm der Applikation

Model: Pad_Model

Das Model ist das Herzstück der Anwendung. Es enthält die Geschäftslogik für die Verschlüsselung und Entschlüsselung von Nachrichten. Der Kern des Models ist durch zwei Hauptmethoden dargestellt: `encrypt` und `decrypt`. Diese Methoden implementieren die Logik des One-Time-Pad-Verschlüsselungsverfahrens, indem sie ASCII-Werte der Zeichen in den Nachrichten und Pads manipulieren.

```
1 usage   Cédric Wespi
public String encrypt(String message, String pad) {
    if (!isValidInput(message) || !isValidInput(pad)) {
        throw new IllegalArgumentException("Text und Pad dürfen nur ASCII-Zeichen zwischen 0x20 und 0x7E enthalten.");
    }

    char[] messageChars = message.toCharArray();
    char[] padChars = pad.toCharArray();

    if (padChars.length < messageChars.length) {
        throw new IllegalArgumentException("Das Pad muss mindestens so lang sein wie die Nachricht.");
    }

    char[] encryptedChars = new char[messageChars.length];
    for (int i = 0; i < messageChars.length; i++) {
        int messageVal = messageChars[i] - 0x20;
        int padVal = padChars[i] - 0x20;
        int encryptedVal = (messageVal + padVal) % 95; // Falls Summe > 94, subtrahieren wir 95
        encryptedChars[i] = (char) (encryptedVal + 0x20);
    }
```

Abbildung 2: Verschlüsselungslogik

View: Pad_View

Die View-Komponente, implementiert durch die Klasse `Pad_View`, ist für die Darstellung der Benutzeroberfläche zuständig. Sie enthält alle JavaFX-Elemente wie Textfelder, Buttons und Labels. Die View ist so gestaltet, dass sie auf Benutzeraktionen reagiert und entsprechende Anfragen an den Controller weiterleitet.

Ein wichtiger Aspekt der View ist die Möglichkeit für den Benutzer, Dateien über den `FileChooser` zu laden. Dies ermöglicht die Verwendung benutzerdefinierter Pads für die Verschlüsselung und Entschlüsselung.

```
1 usage  Cédric Wespi
private void loadPadFile(Stage stage) {
    FileChooser fileChooser = new FileChooser();
    File file = fileChooser.showOpenDialog(stage);
    if (file != null) {
        try {
            pad = new String(Files.readAllBytes(Paths.get(file.toURI())))
                    .replaceAll( regex: "[^\\x20-\\x7E]", replacement: "");
            isPadLoadedFromFile = true;
            loadPadButton.setStyle("-fx-text-fill: green;");
        } catch (Exception e) {
            showError( errorMessage: "Fehler beim Lesen der Pad-Datei: " + e.getMessage());
        }
    }
}
```

Abbildung 3: Logik zum Laden des Pads aus einer Datei.

Controller: Pad_Controller

Der Controller agiert als Mittler zwischen View und Model. Er empfängt Eingaben von der View, wie die Benutzernachricht und das One-Time-Pad, und ruft die entsprechenden Methoden des Models auf. Der Controller handhabt auch Fehler und aktualisiert die View mit Ergebnissen oder Fehlermeldungen.

```
1 usage  Cédric Wespi
public void encryptMessage() {
    String message = view.getMessageInput();
    String pad = view.isPadLoadedFromFile() ? view.getPad() : model.getFixedPad();

    try {
        String encryptedMessage = model.encrypt(message, pad);
        view.setOutput(encryptedMessage);
    } catch (Exception e) {
        view.showError(e.getMessage());
    }
}
```

Abbildung 4: Anfrage an das Model, um eine Nachricht zu verschlüsseln.

Zusammenspiel der Komponenten

- Die Main-Klasse initiiert die Anwendung und startet die Pad_View.
- Die Pad_View präsentiert die Benutzeroberfläche und leitet Benutzeraktionen an den Pad_Controller weiter.
- Der Pad_Controller verarbeitet diese Aktionen, indem er mit dem Pad_Model interagiert.
- Das Pad_Model führt die eigentliche Logik für Verschlüsselung und Entschlüsselung aus und gibt die Ergebnisse an den Controller zurück.
- Der Controller aktualisiert anschließend die View mit den Ergebnissen oder Fehlermeldungen.

Diese Struktur ermöglicht eine klare Trennung von Anliegen und erleichtert die Wartung und Erweiterung der Anwendung. Das MVC-Design sorgt dafür, dass die Anwendung gut organisiert bleibt, was die Lesbarkeit und Wartbarkeit des Codes verbessert.

Fazit

Das praktische Implementationskonzept basiert auf der MVC-Architektur und nutzt die Stärken von JavaFX für die Gestaltung einer benutzerfreundlichen Oberfläche. Die klare Trennung zwischen Model, View und Controller ermöglicht eine flexible Entwicklung und Wartung der Anwendung und stellt sicher, dass die einzelnen Komponenten spezifische, wohldefinierte Rollen haben.

Ergebnisse

Funktionalität und Leistung

Das Hauptziel des Projekts war die Entwicklung einer Anwendung, die das One-Time-Pad-Verfahren zur Verschlüsselung und Entschlüsselung von Nachrichten anwendet. Dieses Ziel wurde erfolgreich erreicht. Die Anwendung ermöglicht es den Benutzern, Textnachrichten einzugeben und diese mit einem vorgegebenen oder einem aus einer Datei geladenen Pad zu verschlüsseln bzw. zu entschlüsseln.

Die Anwendung erfüllt ihre Kernfunktionen zuverlässig. Die Tests zeigten, dass die Verschlüsselungs- und Entschlüsselungsfunktionen korrekt arbeiten, solange die Eingaben den Anforderungen entsprechen. Der Algorithmus, wie im PDF beschrieben, behandelt ASCII-Zeichen zwischen 0x20 und 0x7E.

Benutzererfahrung und GUI

Die Benutzeroberfläche, entwickelt mit JavaFX, bietet eine angenehme Benutzererfahrung. Die Einbindung von Elementen wie dem FileChooser für das Laden von One-Time-Pads oder Nachrichten erhöht die Benutzerfreundlichkeit der Anwendung. Die Gestaltung der GUI mit Buttons, Textfeldern und Labels macht die Navigation und Bedienung der Anwendung einfach und unkompliziert.

Einschränkungen

Eine wesentliche Einschränkung der Anwendung ist die Beschränkung auf ASCII-Zeichen. Dies schränkt die Vielseitigkeit der Anwendung ein, insbesondere in Bezug auf internationale Sprachen, die Zeichen außerhalb des ASCII-Bereichs verwenden. Darüber hinaus basiert die Sicherheit der Anwendung auf der Einmaligkeit und Geheimhaltung des One-Time-Pads, was in der Praxis eine Herausforderung darstellen kann, insbesondere wenn Pads zwischen den Kommunikationspartnern sicher ausgetauscht werden müssen.

Lernerfahrungen und Erkenntnisse

Aus dem Projekt lassen sich mehrere wichtige Erkenntnisse ableiten. Erstens zeigt es die praktische Anwendbarkeit des One-Time-Pad-Verfahrens, eines der sichersten Verschlüsselungsverfahren, das jedoch in der Praxis aufgrund seiner Einschränkungen selten verwendet wird. Zweitens verdeutlicht das Projekt die Bedeutung von Benutzerfreundlichkeit bei der Entwicklung von Softwareanwendungen, insbesondere wenn diese komplexe Konzepte wie Verschlüsselung inklusive Uploads beinhalten.

Weiterführende Ideen

Für die Weiterführung der Applikation wäre es interessant, diese um die Möglichkeit zu erweitern, internationale Zeichen zu unterstützen. Dies würde die Anwendung vielseitiger und weltweit anwendbarer machen. Eine weitere mögliche Erweiterung könnte in der automatisierten Generierung sicherer One-Time-Pads liegen, um die Anwendung noch benutzerfreundlicher zu gestalten.

Fazit

Insgesamt hat das Projekt sein Ziel erreicht, indem es eine funktionsfähige und benutzerfreundliche Anwendung zur Demonstration des One-Time-Pad-Verfahrens entwickelt hat. Die Einschränkungen und Lernerfahrungen bieten wertvolle Einblicke für zukünftige Projekte und Verbesserungen.

Zusammenfassung

Das Projekt zielte darauf ab, das theoretisch sichere One-Time-Pad-Verschlüsselungsverfahren in einer benutzerfreundlichen Anwendung umzusetzen. Es wurde eine Anwendung entwickelt, die eine JavaFX-Oberfläche nutzt, um Nachrichten zu verschlüsseln und zu entschlüsseln.

Die Anwendung demonstriert das One-Time-Pad-Verfahren, wobei die Implementierung auf ASCII-Zeichen begrenzt ist. Die MVC-Architektur sorgt für eine effiziente Strukturierung des Codes, was die Wartung und zukünftige Erweiterungen erleichtert. Die Benutzeroberfläche ist benutzerfreundlich gestaltet und ermöglicht es den Nutzern, eigene Pads aus Dateien zu laden, was die Anwendung flexibel macht.

Aus dem Projekt ergeben sich mehrere Schlussfolgerungen und Erkenntnisse. Es bestätigt es die Machbarkeit der Implementierung Kryptografieverfahren in Anwendungen.

Für zukünftige Projekte wäre es interessant, die Anwendung auf die Unterstützung internationaler Zeichensätze zu erweitern und automatisierte Verfahren zur sicheren Pad-Generierung zu integrieren. Dies würde die Vielseitigkeit und Sicherheit der Anwendung erhöhen.

Quellen und Literatur

https://www.w3schools.com/java/default.asp

https://openai.com/chatgpt

https://stackoverflow.com/

https://moodle.fhnw.ch/course/view.php?id=55540

Abbildungsverzeichnis