Protokoll Informatik 3 - Verteilte Systeme

Statisik zum Passwort knacken

Dominik Coordes, Matr.-Nr.: 1157745
Markus Mehrbrodt, Matr.-Nr.: 1310250
Mohamed Salheen, Matr.-Nr.: 1330355

Wintersemester 2023/2024

Inhaltsverzeichnis

1 Aufgabenstellung

Praktikum INF3 - Verteilte Systeme

Version 2.0 (24. Nov. 2020)

Projektziele

(A) Statistik zum Passwort knacken und zur Detektion eines korrupten Datentransfers

P3A00:
Erstellen Sie eine Statistik über die Anzahl der notwendigen Versuche zum erraten ein Passworts in Abhängigkeit des verwendeten Alphabets und der maximalen Passwortlänge.
Die Passwortabfrage erfolgt an einem Server. Der Server wird von Ihnen bzgl. gegebener Passwortlänge und Alphabet konfiguriert und gestartet.
Ein von ihnen implementierter Client sendet Passwörter an den Server und testet dessen Antwort, um festzustellen, ob das Passwort korrekt ist oder nicht.

Für die Erzeugung der Daten für ihre Statistik sollen Sie mehrere Server und entsprechend viele Client-Prozesse parallel laufen lassen. Die Client-Prozesse sollen mittels Threads-Programmierung Parallelisiert werden.

Nutzen, Sie für den Passworttest die Klasse `TASK1::BlackBoxSafe`, welche noch nach gegebener Spezifikation implementiert werden muss (siehe todo-Liste in der ursprünglichen Doxygen-Dokumentation).
Erledigen Sie ebenfalls die anderen zwei Punkte auf der todo-Liste in der ursprünglichen Doxygen-Dokumentation (`generateDataContent`(), `SimFileServer()`).

(B) Automatische Abarbeitung von Strategien für das Spiel „Einbahn Schiffe versenken“

P3A01:
Erstellen Sie ein Programm mit dem Sie eine Strategie für das Spiel „Einbahn Schiffe versenken“ bewerten können. Dafür sollen Sie ein Server implementiert auf dem eine Spielfläche erzeugt wird, der Server Koordinaten entgegen nimmt und entsprechend bzgl. Treffer und Nicht-Treffer zurück gibt.

Die Funktionalität des Spiel ist gegeben. Siehe dazu die entsprechenden

Implementierungen und das Demo in der Deklarationsdatei `TASK3.H`. Implementiert werden muss der Server.

Ihr Client soll ohne Benutzer-Interaktion die Anfragen an den Server senden, bis das Spiel beendet ist. D.h., Sie müssen eine Strategie vollautomatisch programmieren. Implementieren Sie verschiedene Strategien und vergleiche Sie diese z.B. bzgl. der mittleren Anzahl von benötigten Spielzügen.

(C) Server für Lernen und Abfrage Neuronale Netze

P3A02:
Erstellen Sie einen Server für die Arbeit mit vorwärts gerichteten Neuronalen Netzen. Der Server soll sowohl für das Lernen eines Netzes als auch für Abfrage eines Netzes verwendet werden können.

Der Server wird mit der Anzahl der Eingabe, der Hidden, der Ausgabe-Neuronen, einer gegebenen Transferfunktion und Lernrate konfiguriert und gestartet. Der Server kann verschiedene Anfragen (Lernen, Test) und die entsprechenden Wertemuster entgegennehmen und zurücksenden.

Auch sollen Kommandos zum Speichern und Einlesen von neuronalen Netzen an den Server gesendet werden können.

Projektdokumentation

(1) Sie arbeiten in einem Team von maximal drei Personen.

(2) Die Arbeit wird durch ein Protokoll dokumentiert, dass nach dem letzten Praktikumstermin und vor dem Ende des anschließenden Prüfungszeitraum abgegeben / eingereicht werden muss. Abgabe erfolgt dieses Semester per dem im Kursraum bereitgestellten ILIAS-Übungsobjekt.

(3) Dokumentation erfolgt im Sourcecode per Doxygen.

(4) Der Sourcecode muss vollständig und lauffähig in Ihrem Projekt-Repository hinterlegt sein.

Projektumsetzung – Praktikumstermin P1

Erstes Ziel ist es, sich mit der Entwicklungsumgebung Eclipse unter Linux vertraut zu machen. Weiterhin sollen Sie Ihr SW-Repository zur Archivierung und Dokumentation Ihrer Programmierarbeit eingerichtet haben. Sie sollten außerdem als Beispiel-Anwendung den Standard-Image-Server in Ihrer Umgebung ausführen können.

P1A00: Erstellen eines git-Repositories mit eigenem Benutzerkonto

http://www.github.com

P1A01: Erstellung einer eigenen Entwicklungsumgebung unter Linux

P1A02: Integration der gegebenen Software in ihre Entwicklungsumgebung

http://www.github.com/amlmsh/INF3_Prak

P1A03: Ausführen des Beispiels zum Standard-Image-Server

http://www.github.com/amlmsh/StdImgSrv

Projektumsetzung – Praktikumstermin P2

Sie sollten sich zum P2 entschieden haben, welches Projektaufgabe Sie umsetzen wollen und schon Konzepte für dessen Umsetzung erarbeitet haben und erklären können.

Zu diesem Praktikum sollen Sie eine erste Version Ihres Protokolls präsentieren können. Sie sollen in Ihrem Repository erste Programmiertätigkeiten nachweisen können. Sonst soll das Praktikum P2 nutzen um konkrete Fragen zur SW-technischen Umsetzung klären zu können.

Im Protokoll sollen die Ergebnisse der Aufgabenstellung **P1A00** bis **P1A03** dokumentiert sein.

P2A00: In der ursprünglichen Doxygen-Dokumentation ist unter der todo-List die Implementierung der Klasse `BlackBoxSafe` gelistet. Erledigen Sie diesen Punkt und schreiben Sie einen Unit-Test dafür. Als Unitest-Umgebung nutzen Sie bitte nur die SW aus TASK6.

Erstellen Sie das vollständige UML-Klassendiagramm inkl. aller Elternklassen in elektronischer Form für die Klassen der Unit-Test Umgebung und für die Klassen der Black-Box-Klassen.

P2A01: Variieren Sie die Demo `TASK5::demoOB00` um die Wirkungsweise des Design-Muster *Observer* (*Publish-Subscribe*) zu erläutern zu können. Erstellen Sie das vollständigen UML-Klassendiagramm in elektronischer Form für die Demo inkl. aller Elternklassen.

Projektumsetzung – Praktikumstermin P3

Im Protokoll sollen nun auch die Ergebnisse der Aufgabenstellung **P2A00** bis **P2A01** dokumentiert sein.

Sie sollten einen Prototypen demonstrieren und erklären können. Sie sollten dokumentieren und lauffähigen Sourcecode in Ihrem Repository und in Ihrer Entwicklungsumgebung nachweisen können. Das Protokoll sollte in finaler Version vorliegen.

2 Das Projekt

2.1 git-Repository

https://github.com/Demerdragon/INF3_A1_Prak

2.2 Aufruf und Verwendung

1. die gewünschte Passwortlänge in der Datei „client.C“ in der Variabel „int len“ speichern.
2. die gewünschte Alphabetlänge in der Datei „client.C“ in der Variabel „int Slen“ speichern.
3. Konsole öffnen, in den Ordner navigieren und 2x „xterm&“ eingeben
4. in einem der neu geöffneten Fenster „./server“ aufrufen (wichtig: Server zuerst starten.)
5. im anderen der neu geöffneten Fenster „./client“ aufrufen
6. optional: Einen Unterordner „versuche“ erstellen und dann „./client ->versuche/dateiname.csv“ aufrufen, um die Ergebnisse in eine csv-Datei zu schreiben. Diese csv-Datei befindet sich in diesem Unterordner „versuche“, da beim kompilieren die sonst die alte gelöscht werden würde. Die Ergebnisse sind entsprechend für einen Excel-Import formatiert. Neben der Anzahl pro Passwort werden ebenfalls der Mittelwert und die Standardabweichung angegeben.
7. um eine andere Konfiguration an Passwort- und Alphabetlänge zu testen, muss nach dem speichern neu kompiliert werden.
8. die Anzahl der richtig erratenen Passwörter ist in der Variabel „hacked“ in der Datei „client.C“ gespeichert und sollte bei größerer Alphabet- oder Passwortlänge entsprechend angepasst werden.

2.3 UML-Klassendiragram

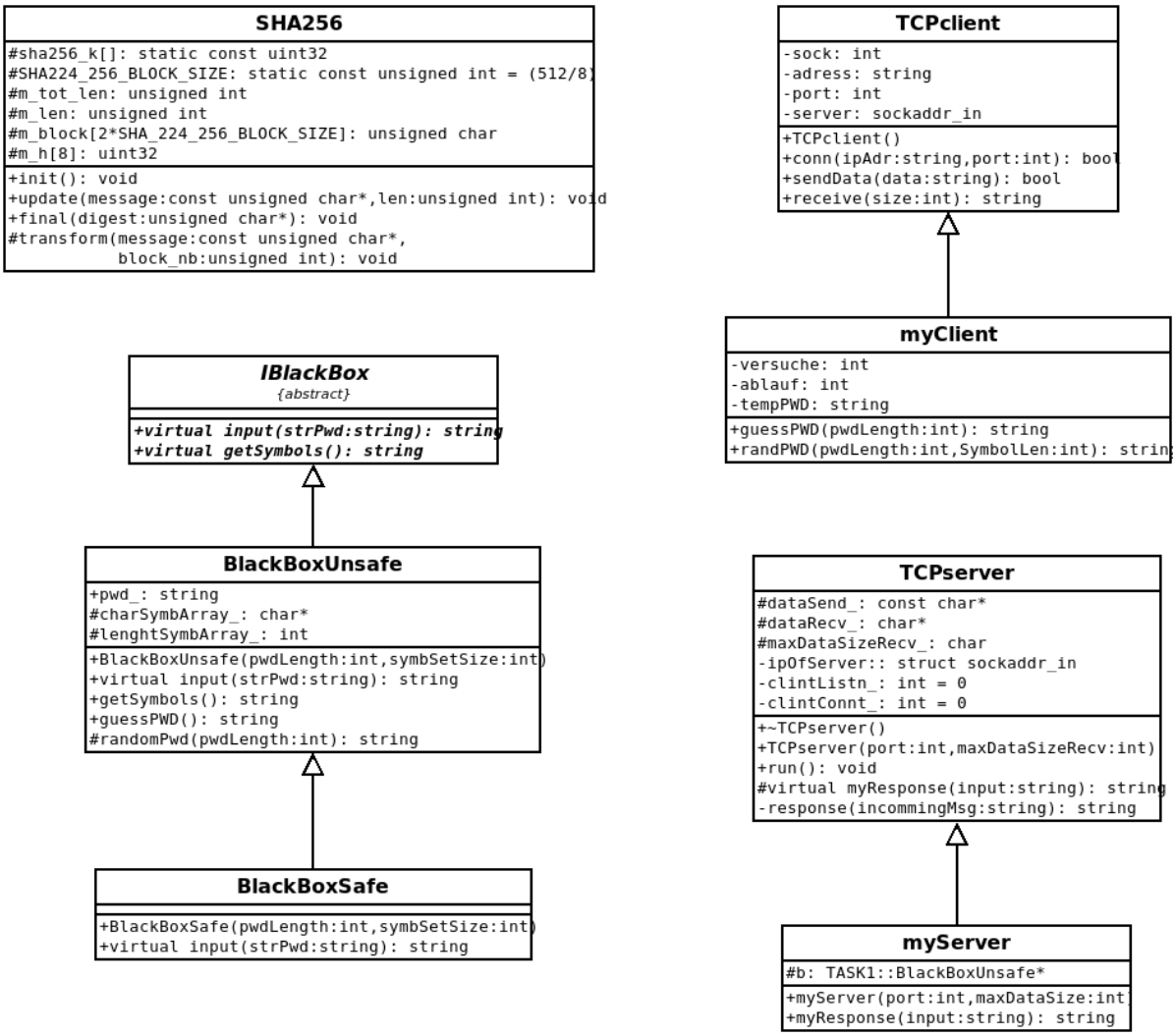

Abbildung 1: UML Diagramm

2.4 Kommunikationsprotokoll

Client	Server
baut Verbindung zu festgelegter Adresse inkl. Port auf	
	bestätigt die Verbindung
fordert die Erstellung eines Passwortes mit vorgegebener Buchstaben- und Alphabetlänge	
	erstellt Passwort entsprechen der Vorgaben und sendet das Passwort zurück an den Client
empfängt und sendet ein zufällig generiertes Passwort mit den angegebenen Vorgaben	
	empfängt das zufällige Passwort vom Client und überprüft es mit dem abgespeicherten. Sind die Passwörter identisch, wird „ACCESS ACCEPTED“ an den Client zurückgegeben, ansonsten „ACCESS DENIED“.
empfängt der Client „ACCESS DENIED“ wird ein neues zufälliges Passwort generiert, bei „ACCESS ACCEPTED“ wird die Variabel „hacked“ um 1 hochgezählt und es wird der Server angewiesen, ein neues Passwort zu erstellen, s.o.; ist die Variabel „hacked“ bei ihrem eingestellten Limit angekommen, geht der Client in Stand-By	

3 Brute-Force Statistik

Die Daten wurden sortiert um einmal die Bedeutung der Passwortlänge und der Alphabetgröße separat aufzuzeigen, indem die jeweils andere Größe konstant gehalten wurde. Zur Veranschaulichung wurden Diagramme erzeugt. Die folgenden Diagramme sind die aussagekräftigsten Beispiele unserer Messungen, bei denen jeweils nur 1 Variabel verändert wurde:

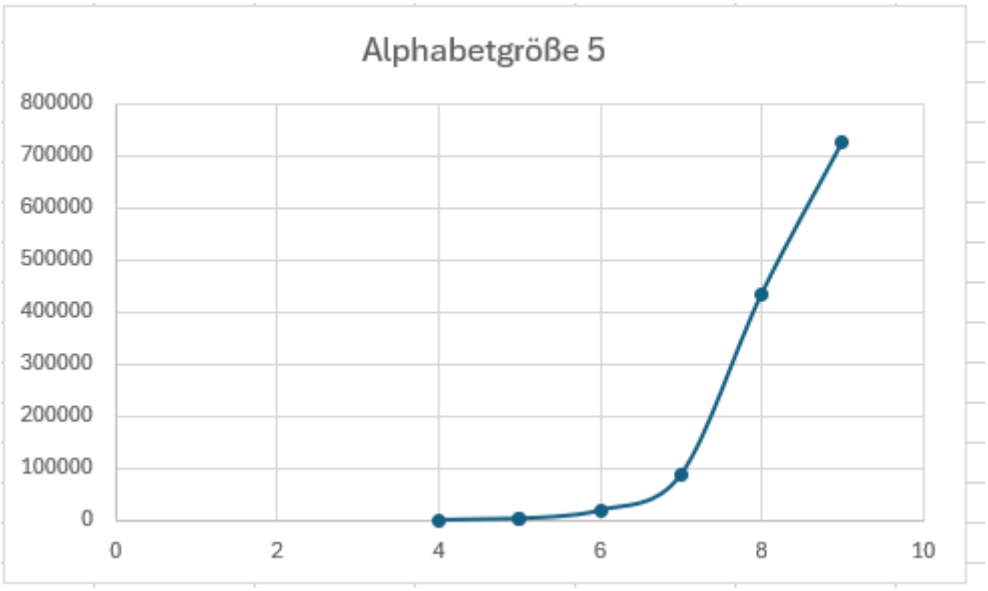

Abbildung 2: feste Alphabetgröße von 5, Passwortlänge variiert von 4-9

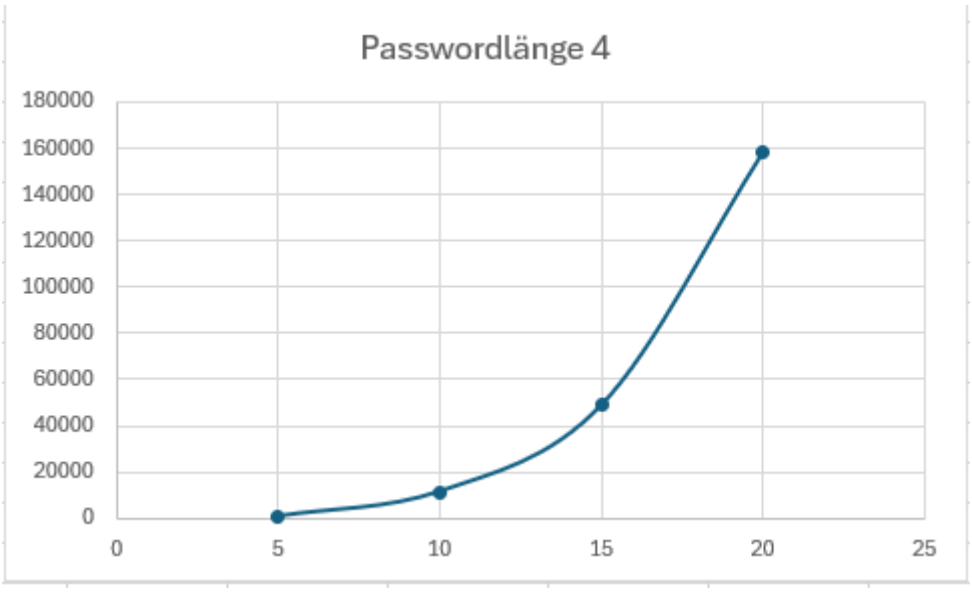

Abbildung 3: feste Passwortlänge von 4, Alphabetgröße variiert von 5-15

Ebenfalls sehr bemerkenswert ist die Standardabweichung. Diese befindet sich in Wertebereich der jeweiligen Mittelwerte.

3.1 Beurteilung bezüglich Länge und Alphabetgröße der Passwörter

Der Einfluss der Passwortlänge ist deutlich größer als der Einfluss der Alphabetlänge. Während sich die Anzahl der benötigten Versuche bei der festen Passwortlänge bei einem variablen Alphabet von 10 auf 15 ungefähr um den faktor 5 wächst, ist die Steigerung der Passwortlänge von 7 auf 8 bei einer festen Alphabetgröße bereits bei Faktor 4,5. Dies ist in sofern zu erwarten, da die Anzahl der möglichen Passwörter durch

$$n = Alphabetgrösse^{Passwortlänge}$$

beschrieben wird. Dies ist vergleichbar mit der Effizienz von Algorithmen, bei denen 2^n der die ineffizienteste Klasse beschreibt.
Die Varianz der benötigten Versuche ist extrem hoch, so ist zum Beispiel die Standardabweichung bei 100 Versuchen (Alphabetgröße 10, Passwortlänge 4) bei 10459 bei einem Mittelwert von 11383. Dadurch lässt sich die Dauer bis das Passwort geknackt ist nur sehr schlecht vorhersagen.