HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTERARBEIT

Frau
Eva Pothmann

Benutzerspezifische Passwort-Rekonstruktion:

Ein Modell einer gezielten, mehrelementigen Offline-Attacke

2021

Fakultät **Angewandte Computer- und Biowissenschaften**

MASTERARBEIT

Benutzerspezifische Passwort-Rekonstruktion:

Ein Modell einer gezielten, mehrelementigen Offline-Attacke

Autorin:
Eva Pothmann

Studiengang:
Cybercrime/ Cybersecurity

Seminargruppe:
CY18wC-M

Erstprüfer:
Hr. Prof. Dr. rer. pol. Dirk Pawlaszczyk

Zweitprüfer:
Hr. Dipl. Inf. Axel Schulze

Stuttgart, 2021

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Vorwort

An dieser Stelle möchte ich mich beim Landeskriminalamt Baden-Württemberg, insbesondere beim Leiter der Inspektion 520 - Digitale Spuren, Hrn. Sandro Pittelkow, für die Unterstützung und die Gelegenheit zur Verfassung dieser Masterarbeit bedanken.

Des Weiteren gilt mein Dank dem gesamten Team der Inspektion 520 für die angenehme Zusammenarbeit und ihre Informationsbereitschaft.

Ein besonderer Dank gilt meinem Betreuer, Hrn. Dipl. Inf. Axel Schulze, der meine Masterarbeit betreut und begutachtet hat. Für die hilfreichen Anregungen und die konstruktive Kritik bei der Erstellung dieser Arbeit möchte ich mich herzlich bedanken.

1 Einleitung

„Sorry but your password must contain a capital letter, two numbers, a symbol, an inspiring message, a spell, a gang sign, a hieroglyph and the blood of a virgin.“ [32]

Dies ist die Aussage eines verbreiteten Internet-Meme, das die Vielzahl an Anforderungen, die ein Passwort zur dienstseitigen Akzeptanz erfüllen muss, symbolisiert. Instinktiv wählen Menschen oftmals besonders einfache Passwörter, um sie bis zum nächsten Anmeldevorgang nicht zu vergessen. Bei den meistgenutzten Passwörtern des Jahres 2020 handelt es sich nach dem Hasso-Plattner-Institut um einfache Ziffernfolgen oder typische Passphrasen wie „123456“, „123456789“ oder „passwort“ [20]. Zudem verwenden rund 60% aller Internetnutzer aufgrund des Besitzes einer Vielzahl an Online-Zugängen ein Passwort mehrfach für unterschiedliche Benutzerkonten [1].
Der Einsatz von dienstspezifischen Passwortrichtlinien soll die Internetnutzer zur Verwendung „sicherer“ Passwörter zwingen und die Angriffsfläche für Cyber-Kriminelle minimieren. Besonders kurze Passwörter oder Passwörter mit typischen Passphrasen sind über frei zugängliche Cracking-Anwendungen ohne großen Aufwand zu erraten. Zudem birgt die Wiederverwendung eines Passworts das Risiko, dass die personenbezogenen Zugangsdaten aufgrund der Kompromittierung eines der Benutzerkonten bereits im Internet kursieren. [21].
Zum Schutz der Benutzer vor Phänomenen der Internetkriminaliät, bedingt durch die Verwendung schwacher Passwörter, definiert das Bundesamt für Sicherheit in der Informatik einige Qualitätsanforderungen an ein Passwort. Demnach soll ein Passwort mindestens 8 Zeichen besitzen und die 4 Zeichentypen Großbuchstabe, Kleinbuchstabe, Ziffer und Sonderzeichen beinhalten. Zudem soll ein Passwort keine persönlichen Informationen enthalten und nicht kontenübergreifend verwendet werden. Zur Verbesserung des Erinnerungsvermögens sollen auf Hilfsstrategien zurückgegriffen werden. Eine Möglichkeit ist das Merken eines Satzes, bei dem die Anfangsbuchstaben der einzelnen Wörter das Passwort bilden. Darüber hinaus wird empfohlen, vor der Nutzung des Passworts sicherzustellen, dass es kein Bestandteil sogenannter „Black-Lists“ ist. Diese enthalten schwache und damit leicht angreifbare Passwörter bzw. Passwort-Elemente und sollten demnach kein Bestandteil eines Passworts sein [2]. Aus der Erfüllung der Qualitätsanforderungen folgen stärkere Passwörter, wodurch ihre Rekonstruktion erschwert wird.

Ziel dieser Arbeit ist die Entwicklung eines Modells, das über einen mehrstufigen Angriffsprozess das Passwort eines spezifischen Benutzers unabhängig von der Stärke des Passworts rekonstruiert. Der Fokus des Modells liegt auf dem benutzerspezifischen Angriff und dessen Präprozessor. Dieser soll unter Berücksichtigung der bisherigen Design- und Konstruktionsprinzipien des Benutzers sowie unter Einbeziehung seiner persönlichen Informationen die wahrscheinlichsten Passwort-Kandidaten generieren.

Zudem soll im Rahmen der Angriffsdurchführung ein optimaler Kompromiss zwischen dem notwendigen Ressourcenaufwand und der Effizienz geschaffen werden.

Die Arbeit gliedert sich in die folgenden Kapitel: Grundlagen (Kapitel 2), Modelle der Rekonstruktion (Kapitel 3), Analyse menschlicher Passwortgenerierung (Kapitel 4), benutzerspezifisches Modell (Kapitel 5), Experimente (Kapitel 6) und Evaluation (Kapitel 7).
In Kapitel 2 wird zunächst eine Einführung in das Gebiet der Passwort-Rekonstruktion gegeben. Neben der Erläuterung bereichsspezifischer Begriffe werden die Einflussfaktoren des Rekonstruktionsaufwands aufgezeigt sowie die Anwendung „Hashcat“ zur Angriffsdurchführung vorgestellt. Zudem werden 3 Messgrößen zur Abschätzung der Höhe der Korrelation zwischen 2 Passwörtern eines Benutzers vorgestellt.
Kapitel 3 gibt einen Überblick über den aktuellen Forschungsstand im Gebiet der Passwort-Rekonstruktion. Es werden fünf Modelle vorgestellt, die sich mit der Entwicklung von Präprozessoren zur Generierung der wahrscheinlichsten Passwort-Kandidaten beschäftigen.
Im vierten Kapitel wird mit dem Ziel der Identifizierung menschlicher Design- und Konstruktionsprinzipien bei Passwörtern eine Analyse eines Passwort-Leaks durchgeführt. Zunächst wird der Inhalt, die Struktur sowie die Länge der Passwörter bzw. der Passwort-Elemente auf benutzerübergreifende Muster hin untersucht. Im Anschluss wird der Fokus auf den Aspekt der Wiederverwendung von Passwörtern gerichtet. Neben der Untersuchung des Umfangs der exakten sowie partiellen Wiederverwendung werden beliebte Modifikationen zwischen Passwörtern herausgearbeitet. Zuletzt findet eine Untersuchung bezüglich des Auftretens von persönlicher Informationen innerhalb eines Passworts statt.
In Kapitel 5 wird das Modell zur benutzerspezifischen Passwort-Rekonstruktion vorgestellt. Nach der Erläuterung des Anwendungsszenarios sowie den Herausforderungen des Modells werden die einzelnen Modellphasen der Initialisierung, der Vorbereitung und des Angriffs durchlaufen und die jeweiligen Verarbeitungsprozesse aufgezeigt. Der Fokus des mehrstufigen Angriffsprozesses liegt auf dem benutzerspezifischen Angriff und dessen Präprozessor, der die wahrscheinlichsten Passwort-Kandidaten für einen spezifischen Benutzer generiert. Seine Strategie sowie seine Umsetzung werden im Detail erläutert und anhand der ersten Evaluations-Aufgabe aufgezeigt.
Im sechsten Kapitel werden 3 Experimente durch Anwendung des benutzerspezifischen Modells auf 3 fiktive Polizeifälle durchgeführt. Das Ziel ist jeweils die Rekonstruktion mehrerer Passwörter eines Benutzers unter minimalem Ressourcenaufwand. Im Rahmen der Angriffsdurchführung wird die Effizienz des benutzerspezifischen Modells in Form der Rekonstruktionszeit gemessen.
Im siebten und letzten Kapitel findet die Evaluation des benutzerspezifischen Modells statt. Zunächst wird die Effizienz des Präprozessors an der Erfolgsrate der Vorhersage von Passwortstrukturen gemessen. Im Anschluss findet ein Vergleich der Rekonstruktionszeiten mit der Bruteforce-Attacke statt. Zuletzt werden die in Kapitel 3 vorgestellten Präprozessoren auf das erste Experiment angewendet und die erzielten Ergebnisse den Ergebnissen des Modells dieser Arbeit gegenübergestellt.

2 Grundlagen

2.1 Passwort

Bei einem Passwort handelt es sich um eine Zeichenfolge, die der Authentifizierung der Identität sowie der Überprüfung der Zugangsberechtigung dient. Es wird eingesetzt, um den Zugang zu einem geschlossenen System zu ermöglichen sowie dieses vor Missbrauch durch Außenstehende zu schützen. Das Passwort stellt damit ein denkwürdiges Geheimnis dar, welches aus Zeichen eines definierten Zeichensatzes Z besteht [30, S. 7].

Definition 2.1 *Ein Zeichensatz Z ist eine endliche, nicht-leere Menge von Zeichen. Ein Passwort m über dem Zeichensatz Z ist eine endliche Folge*

$$m = m_0 m_1 ... m_{n-1} \text{ mit } m_i \in Z, n \in \mathbb{N} \tag{2.1}$$

wobei $|m| = n$ die Länge des Passworts m ist.

Ein gebräuchlicher Zeichensatz im Bereich der Informationstechnik ist der ASCII-Zeichensatz[1]. Er besteht aus insgesamt 95 druckbaren Zeichen und ist in Tabelle 2.1, nach den Zeichentypen gruppiert, abgebildet.

Tabelle 2.1: ASCII-Zeichensatz

Zeichentyp	Zeichen	Anzahl an Zeichen
Kleinbuchstabe	abcdefghijklmnopqrstuvwxyz	26
Großbuchstabe	ABCDEFGHIJKLMNOPQRSTUVWXYZ	26
Ziffer	0123456789	10
Sonderzeichen	<leerzeichen>!“#$%&’()*+,-./:;<=>?@[]_‘{|}~\	33

Quelle: eigene Darstellung.

Zur Verifizierung der Zugangsberechtigung eines Benutzers[2] speichert das geschützte, geschlossene System das entsprechende Passwort ab. Aus Sicherheitsgründen werden die Passwörter nicht in Klartext gespeichert, sondern in verschlüsselter Form als sogenannter „Hashwert“.

[1] American Standard Code for Information Interchange: 7-Bit Zeichenkodierung.

[2] Aus Vereinfachungsgründen wird im Rahmen dieser Arbeit die Bezeichnung „Benutzer“ geschlechtsübergreifend verwendet.

2.2 Hashfunktion

Ein Hashwert ist eine mathematische Prüfsumme, die durch Anwendung einer Hashfunktion auf einen Eingabe-Datensatz (das Passwort) erzeugt wird. Die Hashfunktion selbst ist ein Algorithmus, der einen Eingabe-Datensatz beliebiger Länge auf eine Zeichenkette fester Länge (den Hashwert) abbildet.
Im Vergleich zu einem Verschlüsselungsalgorithmus ist eine Hashfunktion nicht invertierbar. Der Hashwert lässt sich folglich nicht in das Passwort zurückrechnen. Für den Einsatz einer Hashfunktion in kryptografischen Verfahren erfüllt sie die folgenden drei Eigenschaften [3, S. 41 f.]:

Definition 2.2 *Eine Hashfunktion H ist ein kryptografischer Algorithmus, der eine binäre Zeichenkette $m \in \{0,1\}$ beliebiger Länge auf eine binäre Zeichenkette $h \in \{0,1\}^n$ fester Länge $n \in \mathbb{N}$ abbildet:*

$$H : m \to h \Leftrightarrow H : \{0,1\} \to \{0,1\}^n \tag{2.2}$$

1. Einweg-Funktion: Die Hashfunktion ist nicht invertierbar. Demzufolge ist es praktisch unmöglich, für einen gegebenen Ausgabewert h einen Eingabewert m zu finden, sodass $H(m) = h$.

2. Urbild-Resistenz: Zu einem gegebenen Passwort existiert kein zweites Passwort mit demselben Hashwert. Demzufolge ist es praktisch unmöglich, für einen gegebenen Eingabewert m eine Zeichenkette $m' \in \{0,1\}$ zu finden, sodass $H(m) = H(m')$ mit $m \neq m'$.

3. Starke Kollisionsresistenz: Es gibt keine zwei Passwörter mit demselben Hashwert. Demzufolge ist es praktisch unmöglich, zwei Zeichenketten $m, m' \in \{0,1\}$ zu finden, sodass $H(m) = H(m')$ mit $m \neq m'$.

Weiterhin ist eine Hashfunktion deterministisch. Sie erzeugt demnach bei mehrfacher Anwendung auf einen Eingabewert stets denselben Ausgabewert. Zudem ist sie effizient berechenbar und die Menge an Eingabewerten $|m|$ ist deutlich größer als die Menge an Ausgabewerten $|h|$ [3, S. 41 f.].
Ein Angreifer, der sich unautorisiert Zugang zu einem geschlossenen System verschafft, ist demzufolge zunächst gezwungen, das Passwort, zu dem im System gespeicherten Hashwert zu finden. Dieser Prozess wird als Passwort-Rekonstruktion bezeichnet. Er erfolgt über die Berechnung der Hashwerte möglicher Passwörter (im Folgenden „Passwort-Kandidaten“ genannt) und dessen Abgleich mit dem Ziel-Hashwert. Stimmen der berechnete Hashwert des Passwort-Kandidaten und der Ziel-Hashwert überein, ist das Passwort erfolgreich rekonstruiert und liegt in Klartext vor (siehe Abschnitt 2.5). Erst nach erfolgreicher Rekonstruktion kann der Angreifer auf die im System gespeicherten Daten zugreifen. Die Speicherung des Hashwerts anstelle des Klartext-Pass-worts erhöht somit die Datensicherheit durch Bewahrung der Geheimhaltung des Passworts.

2.3 Passwort-Leak

Immer wieder kommt es zur Veröffentlichung ganzer Listen mit Hashwerten von Passwörtern („Passwort-Leak“). Hiervon sind meist eine Vielzahl von Benutzern bei verschiedenen Diensten betroffen. Die Hashinformationen stammen entweder aus einem vorsätzlichen Datendiebstahl, bei dem ein Dienst Opfer eines Hackerangriffs ist („data breach“) oder aus einem unbeabsichtigten Datenabfluss aufgrund einer Sicherheitslücke im System eines Dienstes („data leakage“). Die Passwörter des Leaks werden zunächst innerhalb bestimmter Personenkreise rekonstruiert, um an die vertraulichen Zugangsdaten der Benutzer zu gelangen [7, S. D-9].
Da es sich bei den Passwörtern um menschlich generierte Passwörter aus der Realwelt handelt, geben sie Aufschluss über menschliche Design- und Konstruktionsprinzipien bei der Passwortgenerierung. Zudem geben sie Informationen über die Häufigkeit der Wiederverwendung von Passwörtern sowie über beliebte Modifikationen an Passwörtern preis. Die rekonstruierten Passwörter werden in sogenannten „Wörterbüchern“ zusammengefasst und bilden die Grundlage für den in Abschnitt 2.5.2 beschriebenen Wörterbuch-Angriff.

2.4 Passwort-Rekonstruktion

Der Prozess einer Passwort-Rekonstruktion ist stets mit einem Ressourceneinsatz verbunden. Bei den Ressourcen handelt es sich hierbei um Zeit, Hardware sowie Software. Der Umfang des notwendigen Ressourcenaufwands bestimmt sich zum einen durch die Höhe der Passwortsicherheit sowie zum anderen durch die Hashleistung. Im Folgenden werden beide Einflussfaktoren näher erläutert.

2.4.1 Passwortsicherheit

Die Sicherheit eines Passworts bestimmt sich maßgeblich anhand der Passwortstärke, welche sich aus der Länge sowie der Komplexität des Passworts zusammensetzt. Mit zunehmender Passwortlänge steigt der für die Rekonstruktion notwendige Rechenaufwand und desto schwieriger ist folglich die Rekonstruktion (siehe Abschnitt 2.5.1).
Die Komplexität eines Passworts ergibt sich aus der Größe des verwendeten Zeichensatzes sowie aus der Zusammensetzung der einzelnen Passwort-Elemente aus unterschiedlichen Zeichentypen. Mit zunehmender Größe des zugrundeliegenden Zeichensatzes sowie mit häufiger wechselnden Zeichentypen innerhalb eines Passwort steigt die Stärke eines Passworts sowie der Schwierigkeitsgrad der Rekonstruktion [41].

Bei Betrachtung der Passwörter „fortnite“ und „Fortnite69!“ steigt beispielsweise die Schwierigkeit der Rekonstruktion von 26^8 auf 95^{11} zu testende Passwort-Kandidaten. Während das Passwort „fortnite“ mit einer Länge von 8 Zeichen lediglich aus Kleinbuchstaben (26 Zeichen) besteht, wird bei dem Passwort „Fortnite69!“ mit einer Länge von 11 Zeichen bereits der vollständige ASCII-Zeichenraum von 95 Zeichen verwendet [5, S. 822].
Neben der Länge und der Komplexität stellt der Inhalt eines Passworts einen weiteren Einflussfaktor auf die Passwortstärke dar. Beinhaltet ein Passwort besonders beliebte Zeichenketten, wie beispielsweise „password“ oder „12345“ sinkt die Resistenz gegenüber einem Rekonstruktions-Angriff drastisch. Grund hierfür sind die in Abschnitt 2.5.2 näher erläuterten Wörterbuch-Angriffe [13, S.68].

Ein nicht zu vernachlässigender Aspekt liegt im Zielkonflikt zwischen der Passwortsicherheit und der Benutzerfreundlichkeit. Ein Zugewinn an Sicherheit geht stets mit einer sinkenden Anwendungsfreundlichkeit einher. Der Grund hierfür ist zum einen der zeitliche Mehraufwand, der bis zur erfolgreichen Generierung eines vom Dienst akzeptierten Passworts entsteht. Hinzu kommt der langwierige Anmeldeprozess aufgrund der Eingabe eines längeren und komplexeren Passworts [34, S. 64 f.].
Zum anderen erschweren komplexe Passwörter das menschliche Erinnerungsvermögen. Je komplexer und willkürlicher ein Passwort gewählt ist, desto schwieriger ist es für einen Benutzer, sich dieses zu merken. Diese Tatsache birgt neue Risiken, die zu einer sinkenden Passwortsicherheit führen können. Schreibt ein Benutzer beispielsweise ein Passwort nieder, um es bis zur nächsten Eingabe nicht zu vergessen, und legt es an einem unsicheren Ort ab, könnte sich ein Außenstehender Zugang zu seinem Konto verschaffen. Das Ziel des Einsatzes von Passwortrichtlinien, die Erhöhung der Passwortsicherheit, wäre somit verfehlt [13, S. 68].

Zur Abschätzung der Passwortstärke sowie des Rekonstruktionsaufwands eines Passworts definiert das „National Institute of Standards and Technology“ („NIST“) die „Entropie“ bzw. die „Min-Entropie“. Beide Messgrößen basieren auf der Theorie von Shannon [37] aus dem Gebiet der Informationstheorie, die den Informationsgehalt von Zeichenketten englischer Sprache misst. Aufgrund der getroffenen Annahme einer identischen Häufigkeitsverteilung von Zeichen in Zeichenketten und in benutzergenerierten Passwörtern ist ihre Qualifikation in der Kryptographie fraglich [4, S.101 ff.]. Eine weitere Betrachtung der Messgrößen findet daher an dieser Stelle nicht statt.

2.4.2 Hashleistung

Neben der Passwortsicherheit bestimmt die Hashleistung den Umfang des notwendigen Ressourcenaufwands einer Rekonstruktion. Die Hashleistung gibt die Berechnungsgeschwindigkeit[3] von Hashwerten an und bestimmt sich maßgeblich durch die Komplexität der zugrundeliegenden Hashfunktion.
Bei der NTLM-Hashfunktion handelt es sich beispielsweise um eine Hashfunktion geringer Komplexität, die eine durchschnittliche Hashrate von rund 10 GH/s erzielt. Im Vergleich dazu ist die Bitlocker-Verschlüsselung mit einer durchschnittlichen Hashrate von 1 kH/s eine komplexe und somit langsame Hashfunktion.
Zur Verdeutlichung des Einflusses der Hashgeschwindigkeit auf den zeitlichen Ressourcenaufwand ist in Tabelle 2.2 die zu erwartende Rechenzeit bei Durchführung von Bruteforce-Angriffen unterschiedlichen Umfangs (siehe Abschnitt 2.5.1) auf das Passwort eines NTLM- und eines Bitlocker-Hashes abgebildet.

Tabelle 2.2: Rechenzeit einer Bruteforce-Attacke i.A.v. der Hashfunktion[4]

Anzahl an Zeichen	NTLM	Bitlocker
4	<1 Sek.	20 Std.
6	1 Min.	21 J.
8	8 T.	192.000 J.
10	173 J.	1,7 Mrd. J.

Quelle: eigene Darstellung.

Zunächst ist deutlich zu erkennen, dass der zeitliche Rechenaufwand, unabhängig von der zugrundeliegenden Hashfunktion, mit zunehmender Passwortlänge steigt. Dieser Aspekt wird in Abschnitt 2.5.1 nochmals vertieft.
Des Weiteren lassen sich in Abhängigkeit von der Komplexität bzw. von der Geschwindigkeit der Hashfunktion deutliche Unterschiede in der zu erwartenden Rechenzeit eines Angriffs erkennen. Während die Rekonstruktion eines vierstelligen Passworts bei der Bitlocker-Verschlüsselung rund 20 Stunden benötigt, liegt die Berechnungsdauer bei einem NTLM-Hash bei weniger als 1 Sekunde.
Ein Angriff auf Passwörter der Länge 6 liegt bei der Bitlocker-Verschlüsselung mit einer Rechenzeit von rund 21 Jahren bereits außerhalb eines realistischen, zeitlichen Ressourcenaufwands. Unter zugrundeliegender NTLM-Hashfunktion hingegen ist ein Angriff bis zu 8 Zeichen mit einer Berechnungsdauer von rund 8 Tagen mit einem vertretbarem zeitlichen Aufwand durchführbar.
Weitere Einflussfaktoren auf die Hashgeschwindigkeit sind die verwendete Hardware sowie die eingesetzte Angriffsmethode, die im Folgenden im Rahmen der Anwendung „Hashcat“ näher erläutert werden.

[3] Die Hashgeschwindigkeit wird in der Einheit „Hash pro Sekunde“ (H/s) gemessen.
[4] Die Durchführung erfolgt über die in Abschnitt 2.5 vorgestellte Software „Hashcat“ auf einer „Geforce GTX 1070“-Grafikkarte. Es handelt sich hierbei um Rundungswerte.

2.5 Hashcat

Eine verbreitete Anwendung zur Durchführung einer Passwort-Rekonstruktion ist das Open-Source Tool „Hashcat“ [15]. Mit dem Ziel, zu einem vorliegenden Hashwert das zugehörige Passwort zu finden, berechnet Hashcat eine Vielzahl von Hashwerten möglicher Passwort-Kandidaten und vergleicht diese mit dem Ziel-Hashwert. Stimmt der Ziel-Hashwert mit dem berechneten Hashwert überein, handelt es sich bei dem verwendeten Passwort-Kandidaten um das gesuchte Klartext-Passwort und die Rekonstruktion ist erfolgreich.
Die Berechnung der Hashwerte erfolgt, wie in Abschnitt 2.2 beschrieben, über die entsprechende Hashfunktion des Ziel-Hashwerts. Hashcat unterstützt hierfür eine Vielzahl verschiedener Hashalgorithmen. Der Befehl zur Ausführung eines Rekonstruktions-Angriffs ist wie folgt definiert:

hashcat.exe -a <Angriffsmethode> -m <Hashmodus> <Ziel-Hashwert>
<Passwort-Kandidaten> (-r <Regelwerk>)

Der Parameter a gibt hierbei die Angriffsmethode an, siehe nachfolgende Abschnitte 2.5.1-2.5.4. Der Parameter m definiert die zugrundeliegende Hashfunktion, also den Hashtyp des Ziel-Hashwerts. Nachfolgend werden der Ziel-Hashwert sowie die Passwort-Kandidaten übergeben. In Abhängigkeit von der Angriffsmethode variiert die Über- bzw. Angabe des Kandidatenraums. Über den Parameter r kann zusätzlich ein Regelwerk übergeben werden, wodurch der zu testende Kandidatenraum zusätzlich erweitert wird (siehe Abschnitt 2.5.2).
Hashcat bietet eine Vielzahl verschiedener Angriffsmethoden an. Zu den häufigsten Angriffsarten zählt die Bruteforce-Attacke ($a = 3$), die Wörterbuch-Attacke ($a = 1$), die Kombinations-Attacke ($a = 0$) sowie die Hybrid-Attacke ($a = 6/7$), die im Folgenden näher erläutert werden.

2.5.1 Bruteforce-Angriff

Die Bruteforce-Attacke berechnet und vergleicht die Hashwerte aller Passwort-Kandidaten eines Passwortraums, der durch die Länge des gesuchten Passworts bestimmt wird. Hierbei wird der vollständige Zeichensatz aus Tabelle 2.1 berücksichtigt. Die Methode garantiert somit bei vollständiger Durchführung, also dem Testen aller Kandidaten eines Passwortraums, die Rekonstruktion des Passworts. Die Erfolgswahrscheinlichkeit der Rekonstruktion steigt mit zunehmender Anzahl an getesteten Kandidaten. Das „Bruteforcen“ des gesamten Passwortraums ist jedoch oftmals aufgrund von beschränkten Zeit- und Hardware-Ressourcen nicht realisierbar, da der Rechenaufwand des Angriffs exponentiell mit der Länge des gesuchten Passworts steigt.

Bei Betrachtung der beispielhaft herangezogenen Passwörter in Tabelle 2.3 wird die steigende Rechenzeit eines Bruteforce-Angriffs mit zunehmender Passwortlänge deutlich. Je länger ein gesuchtes Passwort ist, desto größer ist der zu testende Kandidatenraum mit der Folge eines zeitlich steigenden Ressourcenaufwands.

Tabelle 2.3: Rechenzeit einer Bruteforce-/Mask-Attacke i.A.v. der Passwortlänge[5]

Passwort	Länge	Bruteforce-Angriff		Zeichensatz	Mask-Angriff	
		Raum	Rechenzeit		Raum	Rechenzeit
$gatter$	5	95^5	0,8 Sek.	L	26^5	0,0012 Sek.
$?dani19$	8	95^8	7 T. 16 Std.	U,D,S	$33^1 \cdot 26^4 \cdot 10^2$	0,15 Sek.
$Want2Be:)$	9	95^9	2 J.	L,U,D,S	$26^4 \cdot 10^1 \cdot 26^2 \cdot 33^2$	6 Min.
$123friends$	10	95^{10}	262 T.	L,D	$10^3 \cdot 27^7$	17 Min.
$!Bowling0896$	12	95^{12}	1,7 Mio. J.	L,U,D,S	$33^2 \cdot 26^7 \cdot 10^4$	101 T.

Quelle: eigene Darstellung.

Liegen Informationen über das Design und den Aufbau des gesuchten Passworts vor, kann eine Bruteforce-Attacke mit verkleinertem Passwort-Kandidatenraum, eine sogenannte „Mask-Attacke“, durchgeführt werden. Über die Definition des zu berücksichtigenden Zeichensatzes pro Zeichen (L,U,D,S siehe Abschnitt 2.7) eines Passworts sinkt der zu testende Kandidatenraum des Angriffs. Dadurch wird eine Verbesserung der Berechnungsdauer erzielt [17].
Das Passwort „$!Bowling0896$“ ist beispielsweise über einen Mask-Angriff mit einer Berechnungsdauer von maximal 101 Tagen, deutlich realistischer zu rekonstruieren als über eine Bruteforce-Attacke mit einer Gesamtrechenzeit von rund $1,7$ Mio. Jahren. Dieses Beispiel verdeutlicht den begrenzt effizienten Einsatz der Bruteforce-Attacke. In Abhängigkeit von der Länge eines Passworts sowie unter Berücksichtigung der Komplexität der zugrundeliegenden Hashfunktion und der Leistungsfähigkeit der Hardware muss der Umfang einer Bruteforce-Attacke für einen effizienten Einsatz individuell festgelegt werden.

2.5.2 Wörterbuch-Angriff

Die Passwort-Kandidaten für einen Wörterbuch-Angriff bilden die Passwörter aus sogenannten „Wörterbüchern“. Hierbei handelt es sich um Passwortsammlungen aus der Realwelt, die aus Daten-Leaks stammen (siehe Abschnitt 2.3).
Im Vergleich zur Bruteforce-Attacke, die den gesamten Passwortraum berücksichtigt, begrenzt sich der Kandidatenraum bei dieser Methode auf die Anzahl an Einträgen des herangezogenen Wörterbuchs. Die Durchführung des Angriffs erfolgt über die Berechnung des Hashwerts jedes Elements der Wortliste und dessen Abgleich mit dem Ziel-Hashwert. Die Berechnungsdauer ist somit von der Größe des zugrundeliegenden Wörterbuchs abhängig.

[5] Die Durchführung erfolgt auf einer „Geforce GTX 1070“-Grafikkarte mit einer durchschnittlichen Hashrate von 10.000 MH/s. Es handelt sich hierbei um Rundungswerte.

Wie bereits erwähnt, kann über das Hinzufügen eines Regelwerks der zu testende Kandidatenraum über das Ursprungs-Wörterbuch hinaus erweitert werden. Bei einem Regelwerk handelt es sich um eine Datei, die Anweisungen zur Modifikation eines Passwort-Kandidaten enthält. Bei den Anweisungen handelt es sich um Funktionen, die eine eigene Syntax besitzen und auf jeden Eintrag des Wörterbuchs angewendet werden. In Abhängigkeit von der Anzahl an definierten Funktionen x werden aus einem Eintrag des Wörterbuchs x weitere zu testende Kandidaten generiert.
Hashcat besitzt hierbei eine Vielzahl verschiedener Modifikationsfunktionen, die auf Grundlage der Analyse menschlicher Passwort-Konstruktionsprinzipien definiert und weiterentwickelt werden. Auf diese Weise kann ein Kandidat beispielsweise in unterschiedlicher Groß- und Kleinschreibung erzeugt werden oder um festgelegte Fragmente an verschiedenen Positionen erweitert werden. Während Hashcat grundsätzlich über die CPU sowie über die GPU ausführbar ist, findet die Berechnung der Hashwerte der zusätzlichen Passwort-Kandidaten des Regelwerks ausschließlich über die GPU statt, wodurch die Performance verbessert wird [19].

2.5.3 Kombinations-Angriff

Der Kombinations-Angriff kombiniert zwei Wörterbücher miteinander, wodurch eine Vielzahl neuer Passwort-Kandidaten generiert werden. Jeder Eintrag des ersten Wörterbuchs wird mit jedem Kandidaten des zweiten Wörterbuchs konkateniert und stellt einen neuen zu testenden Kandidaten dar [14].

2.5.4 Hybrid-Angriff

Der Hybrid-Angriff ist eine Kombination aus der Bruteforce- und der Kombinations-Attacke. Anstelle von zwei Wörterbüchern wird ein Wörterbuch mit dem erzeugten Kandidatenraum einer Bruteforce-Attacke konkateniert. Das heißt, alle erzeugten Kandidaten der Bruteforce-Attacke werden jedem Eintrag des Wörterbuchs vorangestellt oder angehängt [16].

2.6 Messgrößen des Übereinstimmungsgrads

Neben der Abhängigkeit der Passwortstärke von der Länge und der Komplexität des Passworts wird sie von der Gegebenheit der Wiederverwendung des Passworts durch den Benutzer beeinflusst. Im Fall der Kompromittierung eines Passworts wäre nicht nur das zugehörige Benutzerkonto betroffen, sondern alle Konten, bei denen der Benutzer das kompromittierte Passwort verwendet. Die Stärke eines Passworts sinkt demnach mit gegebener Wiederverwendung. Die Folgeschäden wie der unauthorisierte Zugriff auf die jeweilig vom Dienst gespeicherten Daten durch den Angreifer steigen mit der

Anzahl der Nutzung wiederverwendeter Passwörter.
Zur Überprüfung, ob es sich bei einem von zwei vorliegenden Passwörtern um ein wiederverwendetes Passwort handelt, können verschiedene Messgrößen herangezogen werden, die jeweils den Grad der Übereinstimmung abschätzen. Diese untergliedern sich grundsätzlich in zeichenbasierte, tokenbasierte und sequenzbasierte Indikatoren. Im Folgenden wird aus jeder Kategorie ein Indikator zur Messung des Übereinstimmungsgrads zweier Zeichenketten vorgestellt.

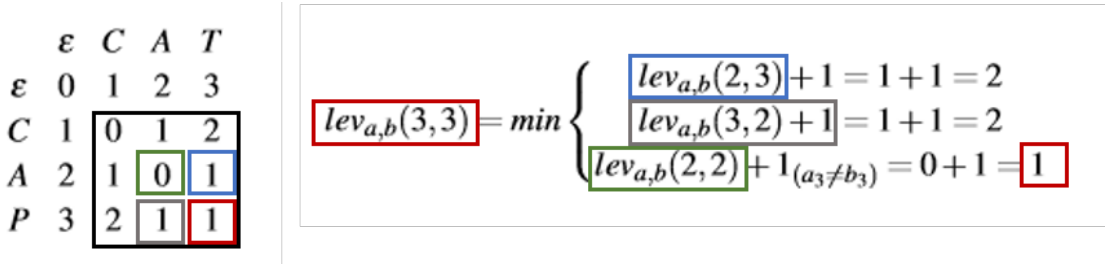
2.6.1 Zeichenbasierte Indikatoren

Zu den zeichenbasierten Indikatoren zählen die Editier-Indikatoren wie beispielsweise die Levenshtein-Distanz. Diese gibt die minimale Anzahl an Editier-Operationen zwischen zwei gegebenen Passwörtern an. Die Editier-Operationen beinhalten dabei das Einfügen, das Löschen sowie den Austausch eines Zeichens. Mit zunehmender Anzahl an Operationen, steigt der Wert der Levenshtein-Distanz und damit der Unterschied zwischen den gegebenen Passwörtern. Eine Levenshtein-Distanz von 0 hingegen bedeutet, dass es sich um zwei identische Passwörter handelt [26].
Zur Berechnung der Levenshtein-Distanz zwischen zwei Passwörtern a und b werden die einzelnen Zeichen der Passwörter in Abhängigkeit von ihrer Position zunächst indexiert. Danach wird zeichenweise unter Erstellung einer Matrix ihr Distanzwert mit folgendem Algorithmus berechnet [29]:

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) \text{ falls } min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} \end{cases} \tag{2.3}$$

wobei i und j mit $1 \leq i \leq |a|$ und $1 \leq j \leq |b|$ den jeweiligen Index der Passwörter a und b der Längen $|a|$ und $|b|$ angeben.
Zur Veranschaulichung der Funktionsweise des Algorithmus ist in Abbildung 2.1 die Berechnung der Levenshtein-Distanz zwischen den Passwörtern „CAT“ und „CAP“ dargestellt.

Abbildung 2.1: Berechnung der Levenshtein-Distanz von „CAT“ und „CAP“



Quelle: eigene Darstellung.

Die Abbildung zeigt den finalen Berechnungsschritt (rote Markierung) an der Matrixposition $(|a|, |b|) = (3, 3)$. Wie zu erkennen ist, besitzen die beiden Passwörter einen Distanzwert von 1, da lediglich eine Editier-Operation in Form des Austauschs des Buchstabens „P“ durch den Buchstaben „T“ für eine Überführung notwendig ist [29].

Der Levenshtein-Ratio berechnet sich über die Teilung der Anzahl an identischen Zeichen beider Passwörter durch die Anzahl an Zeichen des längeren Passworts.

Definition 2.3 *Sei $max(|a|, |b|)$ der maximale Wert der Längen $|a|$ und $|b|$ der Passwörter a und b. Weiterhin sei $lev_{a,b}(|a|, |b|)$ die Levenshtein-Distanz zwischen den Passwörtern a und b. Demzufolge ist der Grad der Übereinstimmung $lev - ratio_{a,b}$ gegeben durch:*

$$lev - ratio_{a,b} = \frac{max(|a|, |b|) - lev_{a,b}(|a|, |b|)}{max(|a|, |b|)} \tag{2.4}$$

Der Übereinstimmungs-Ratio des herangezogenen Beispiels beträgt somit $66,67\%$, da sich nur 1 von 3 Zeichen unterscheidet.

Ein entscheidender Nachteil des Levenshtein-Indikators ist seine starke Abhängigkeit von der Zeichenposition innerhalb eines Passworts [39, S. 458 f.]. Bei Betrachtung der Passwörter „1003bibi“ und „biby1003“ beispielsweise liegt die Levensthein-Distanz bei 7 Operationen. Der resultierende Übereinstimmungsgrad liegt damit bei lediglich 36%. Trotz der offensichtlich starken Ähnlichkeit der beiden Passwörter wird die Übereinstimmung nach Levenshtein niedrig bewertet.

2.6.2 Tokenbasierte Indikatoren

Die tokenbasierten Indikatoren bemessen die Ähnlichkeit zweier Passwörter auf Zeichenketten-Ebene. Hierfür werden die Passwörter zunächst in n-Gramme (auch „Token“ genannt) zerlegt, wobei n die Anzahl an Zeichen pro Token angibt. Besteht ein Token beispielsweise aus zwei Zeichen, handelt es sich um „2-Gramme“, auch „Bigramme“ genannt. Beide Passwörter bestehen somit jeweils aus einem Set an n-Grammen.
Zur Berechnung des Übereinstimmungsgrads, wird die Anzahl an identischen Token beider Sets gezählt. Je höher der Betrag an übereinstimmenden Token ist, desto größer ist die Ähnlichkeit zwischen den vorliegenden Passwörtern.
Ein beliebter tokenbasierter Indikator ist der Matching-Koeffizient auf Bigram-Basis. Die Berechnung erfolgt über die Teilung der Anzahl an übereinstimmenden Token durch die Anzahl an Bigrammen des längeren Passworts [8, S. 1550 f.].

Definition 2.4 *Sei $max(|a|,|b|)$ der maximale Wert der Anzahl an Bigrammen $|a|$ und $|b|$ der Bigram-Sets a und b. Demzufolge ist der Grad der Übereinstimmung gegeben durch:*

$$MC(a,b) = \frac{a \cap b}{max(|a|,|b|)} \tag{2.5}$$

Am Beispiel der Passwörter „1003bibi“ und „biby1003“ erzielt der Matching-Koeffizient mit 6 identischen Token ein Übereinstimmungs-Ratio von 86%. Im Vergleich zum Levenshtein-Ratio mit 36% misst er einen deutlich höheren Grad der Übereinstimmung der beiden Passwörter. Der Grund hierfür ist, dass der Tokenvergleich zwischen den beiden Passwörtern unabhängig von ihrer jeweiligen Position im Passwort stattfindet.

Ein Nachteil des Matching-Koeffizienten ist, dass ein Ratio von 100% nicht gewährleistet, dass es sich tatsächlich um identische Passwörter handelt. Die beiden Zeichenketten „Xanex“ und „Nexan“ beispielsweise besitzen aufgrund ihrer Zusammensetzung aus identischen Bigrammen einen Ähnlichkeits-Ratio von 100%. Aufgrund der unterschiedlichen Anordnung der Bigramme, sind sie jedoch nicht identisch. Dieser Aspekt muss bei der Interpretation der Daten berücksichtigt werden. [39, S. 458 f.].

2.6.3 Sequenzbasierte Indikatoren

Als Messgröße auf Sequenz-Ebene wird der Ratcliff-Obershelp-Indikator herangezogen. Der zugrundeliegende Algorithmus identifiziert in mehreren Schritten jeweils die längste, identische Zeichenkette zweier Passwörter und zählt die Anzahl an übereinstimmenden Zeichen.
Im ersten Schritt sucht er zunächst unabhängig von der Position in den Passwörtern nach der längsten, identischen Zeichenkette. Im Anschluss vergleicht er die resultierenden Subsequenzen linkerhand sowie rechterhand der bereits identifizierten Zeichenkette miteinander und sucht jeweils erneut nach der längsten, übereinstimmenden Zeichenkette. Dieser Prozess wird bis zum letzten Zeichen fortgeführt.
Der Übereinstimmungsgrad berechnet sich über die Teilung der doppelten Anzahl an übereinstimmenden Zeichen durch die Gesamtzahl an Zeichen der beiden Passwörter [23, S. 11 f.].

Definition 2.5 *Seien $|a|$ und $|b|$ die Längen der Passwörter a und b. Weiterhin sei K die Anzahl an übereinstimmenden Zeichen. Demzufolge ist der Grad der Übereinstimmung gegeben durch:*

$$RO(a,b) = \frac{2 \cdot K}{|a| + |b|} \tag{2.6}$$

Am herangezogenen Beispiel erzielt der Ratcliff-Obershelp-Indikator mit einer identischen Zeichenkette, bestehend aus 4 Zeichen, ein Übereinstimmungs-Ratio von 50%. Hierbei handelt es sich um die Subsequenz „1003“. Die Zeichenkette „bib“, die ebenfalls Bestandteil beider Passwörter ist, wird aufgrund ihres Auftretens auf unterschiedlichen Seiten, ausgehend von der bereits identifizierten Subsequenz „1003“, vom Algorithmus nicht erkannt. Dies zeigt analog zur Levenshtein-Distanz die Abhängigkeit von der Position der Passwort-Elemente innerhalb eines Passworts.

Für die Analyse von Realwelt-Passwörtern in Kapitel 4 werden im Folgenden einige Begriffe sowie Abkürzungen bezüglich des Aufbaus und der Struktur von Passwörtern definiert.

2.7 Passwort-Muster

Grundsätzlich kann ein Passwort aus den Zeichentypen „Großbuchstabe“, „Kleinbuchstabe’, „Ziffer“ und „Sonderzeichen“ bestehen (siehe Tabelle 2.1). Im weiteren Verlauf der Arbeit werden die Zeichentypen durch die folgenden Zeichen abgekürzt [5, S. 823 f.]:

- L für ein Kleinbuchstabe.
- U für ein Großbuchstabe.
- D für eine Ziffer.
- S für ein Sonderzeichen.

Die alphabetischen Elemente L und U können durch das Zeichen A zusammengefasst werden. Die Erstellung des Passwort-Musters zu einem Passwort erfolgt durch Substitution jedes einzelnen Zeichens durch das Kürzel des entsprechenden Zeichentyps (im Folgenden „Muster-Elemente“ genannt). Gleiche, aufeinanderfolgende Muster-Elemente werden zusammengefasst und mit ihrer Auftrittshäufigkeit gekennzeichnet. Die Muster-Erstellung erfolgt somit auf Zeichen-Ebene und berücksichtigt die Länge des Passworts [5, S. 823 f.].

Definition 2.6 *Ein Passwort m über dem Zeichensatz Z der Länge n, das ausschließlich aus Kleinbuchstaben besteht, besitzt das Passwort-Muster L_n. Die Definition lässt sich auf die Zeichentypen U, D und S übertragen.*

Zur Abbildung von Wörtern mit einem groß- oder kleingeschriebenen Anfangsbuchstaben auf Muster-Ebene, werden zwei weitere Muster-Elemente definiert:

- N_{n+1} ersetzt die Abfolge der Muster-Elemente $U_1 L_n$ mit $n \geq 2$ und stellt ein kleingeschriebenes Wort mit großem Anfangsbuchstaben dar.

- R_{n+1} ersetzt die Abfolge der Muster-Elemente $L_1 U_n$ mit $n \geq 2$ und stellt ein großgeschriebenes Wort mit kleinem Anfangsbuchstaben dar.

Definition 2.7 *Ein Passwort m über dem Zeichensatz Z der Länge n, das mit einem Großbuchstaben beginnt und dem mindestens zwei Kleinbuchstaben folgen, besitzt das Passwort-Muster N_n. Beginnt das Passwort mit einem Kleinbuchstaben, dem mindestens zwei Großbuchstaben folgen, besitzt es das Passwort-Muster R_n.*

Definition 2.8 *Ein Passwort-Muster ist die Repräsentation des Zeichentyps jedes Elements eines Passworts durch das entsprechende Kürzel L, U, D und S. Es besteht aus einer Abfolge von L_n, U_n, D_n, S_n, N_{n+1} und R_{n+1}, wobei n die Länge der Folge eines gleichen Kürzels angibt.*

Das Passwort „Fortnite69!“ beispielsweise besitzt gemäß der Definition 2.6 die Struktur $U_1 L_7 D_2 S_1$ und somit unter Anwendung der Definitionen 2.7-2.8 das Passwort-Muster $N_8 D_2 S_1$.

2.7.1 Muster-Klasse

Definition 2.9 *Ein Passwort m über dem Zeichensatz Z der Länge n, das ausschließlich aus Kleinbuchstaben besteht, entspringt der Muster-Klasse L^+. Die Definition lässt sich auf die Zeichentypen U, D und S übertragen.*
Beginnt das Passwort mit einem Großbuchstaben, dem mindestens zwei Kleinbuchstaben folgen, entspringt es der Muster-Klasse N^+. Beginnt das Passwort mit einem Kleinbuchstaben, dem mindestens zwei Großbuchstaben folgen, entspringt es der Muster-Klasse R^+.

Die Erstellung der Muster-Klassen erfolgt über die Gruppierung identischer Passwort-Muster unabhängig von der Länge der Muster-Elemente. Die Muster-Klassen beinhalten somit im Vergleich zum Passwort-Muster keine Längeninformationen [5, S. 823 f.].

Definition 2.10 *Eine Muster-Klasse β ist die Repräsentation einer Menge an Passwort-Mustern, die unabhängig von der Länge dieselbe Abfolge der Muster-Klassen-Elemente L^+, U^+, N^+, R^+, D^+ und S^+ aufweisen.*

Das Passwort „Fortnite69!“ mit dem Passwort-Muster $N_8D_2S_1$ gehört gemäß den Definitionen 2.9-2.10 der Muster-Klasse $N^+D^+S^+$ an.

2.7.2 Fragment

Definition 2.11 *Ein Password-Fragment α ist eine endliche, zusammenhängende Zeichenkette deren Zeichen entweder ausschließlich aus einem der Kürzel L,U,D und S oder aus einer der Abfolgen U_1L_n und L_1U_n bestehen.*

Ein Password-Fragment gibt damit die hinter einem Muster-Element stehende Zeichenkette wieder. Die Fragmente resultieren aus der Zerlegung eines Passworts (Fragmentierung), die auf Grundlage des Passwort-Musters erfolgt. Ein Passwort wird an den Positionen getrennt, an denen das Muster-Element wechselt. Die Fragmentierung findet somit auf Zeichenketten-Ebene statt und ist abhängig von der Bedeutung des Inhalts eines Passworts.

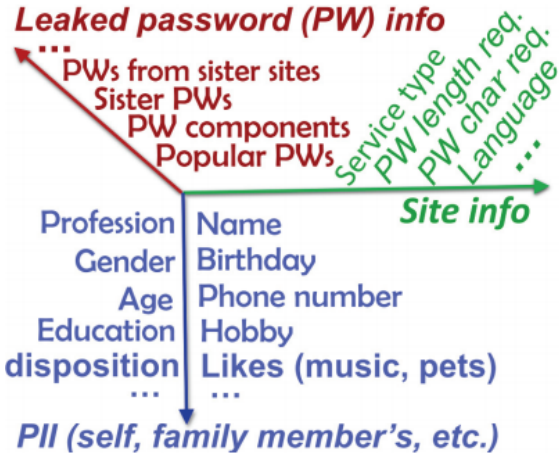
Definition 2.12 *Die Fragment-Klassen γ bestehen aus den Kategorien Alphabet, Numerik und Sonderzeichen. Die Fragmente der Muster-Elemente L_n, U_n, N_n und R_n gehören der alphabetischen Fragment-Klasse an. Die Fragmente der Muster-Elemente D_n und S_n gehören der numerischen Fragment-Klasse bzw. der Klasse der Sonderzeichen an.*

Das beispielhaft herangezogene Passwort „Fortnite69!“ besteht demnach aus dem Fragment „Fortnite“ aus der alphabetischen Fragment-Klasse, „69“ aus der numerischen Fragment-Klasse und dem Ausrufezeichen „!“ aus der Klasse der Sonderzeichen.

2.8 Arten von Informationen

Die in Abschnitt 2.3 vorgestellten Passwortlisten aus Daten-Leaks sind Bestandteil der benutzerspezifischen Informationen und stellen für die Rekonstruktion eines Passworts eines spezifischen Benutzers verwertbare Informationen dar. Sie liefern neben den allgemeinen Rückschlüssen auf menschliche Konstruktionsprinzipien gegebenenfalls veraltete oder bei anderen Diensten verwendete Passwörter eines Benutzers. Diese sogenannten „Vergleichspasswörter“ legen die Designprinzipien des betrachteten Benutzers offen (siehe roter Bereich in Abbildung 2.2).
Des Weiteren lassen sich die persönlichen Informationen eines Benutzers bzw. der ihm nahestehenden Personen den benutzerspezifischen Informationen zuordnen. Sie beinhalten zum einen die direkt-abbildbaren Daten wie beispielsweise den Namen, das Geburtsdatum oder die Hobbys des Benutzers. Zum anderen bestehen sie aus nicht direkt-abbildbaren Informationen wie dem Geschlecht oder der Bildung (blauer Bereich).

Abbildung 2.2: Benutzer- und dienstspezifische Informationen



Quelle: siehe Wang et. al. [38].

Neben den benutzerspezifischen Informationen beeinflussen die dienstspezifischen Informationen (grüner Bereich) die Passwort-Rekonstruktion. Ihr Hauptbestandteil sind die in Abschnitt 2.4.1 vorgestellten Passwortrichtlinien, die beispielsweise die Anforderungen an die Länge sowie die Zusammensetzung eines Passworts definieren. Wie bereits erläutert, erschweren sie aufgrund der Verwendung stärkerer Passwörter, die Rekonstruktion. Im Rahmen eines Rekonstruktions-Angriffs reduzieren sie andererseits den Kandidatenraum um die Passwörter, die die dienstspezifischen Anforderungen nicht erfüllen.
Ein weiterer Einflussfaktor ist die Art des Dienstes und die damit verbundene Sensibilität der vom Dienst gespeicherten Daten. So kann beispielsweise bei einem Zahlungsdienst von der Verwendung eines komplexeren Passworts ausgegangen werden als bei einem Dienst für Online-Spiele.
Zudem ist die Kenntnis der Sprache der Benutzer entscheidend für die Rekonstruktion. Hierfür sind die Länder, in denen der jeweilige Dienst seine Dienstleistung anbietet ein Indiz und zählen damit ebenso zu den dienstspezifischen Informationen [38].

3 Modelle der Rekonstruktion

Die Modelle im Bereich der Passwort-Rekonstruktion beschäftigen sich mit der Entwicklung von Präprozessoren, die mit dem Ziel der Rekonstruktion eines gesuchten Passworts die wahrscheinlichsten Passwort-Kandidaten generieren. Hierzu verfolgen sie unterschiedliche Strategien, die auf der gemeinsame Annahme basieren, dass die generierten Passwort-Kandidaten unterschiedliche Erfolgswahrscheinlichkeiten besitzen, das gesuchte Passwort zu sein.
Mit dem Ziel, die Anzahl der zu testenden Kandidaten sowie den Zeitaufwand bis zur erfolgreichen Rekonstruktion minimal zu halten, werden die Kandidaten in absteigender Reihenfolge ihrer Erfolgswahrscheinlichkeit generiert und getestet. Der Aufbau der Modelle unterteilt sich jeweils in eine Trainings- und Testphase. In der Trainingsphase lernt der Präprozessor anhand eines gegebenen Passwort-Datensatzes und generiert auf Basis des erlernten Wissens neue Passwort-Kandidaten. Der Lernerfolg wird im Anschluss anhand des Testdatensatzes gemessen.

Grundsätzlich lassen sich die Modelle in die Kategorien „benutzerübergreifend“ und „benutzerspezifisch“ untergliedern. Der Unterschied liegt im Benutzerbezug und damit in der Art und dem Umfang der Datengrundlage. Während das Ziel der benutzerübergreifenden Modelle die Maximierung der Anzahl an rekonstruierten Passwörtern verschiedener Benutzer ist, liegt der Fokus der spezifischen Modelle auf der Rekonstruktion des Passworts eines spezifischen Benutzers bei einem festgelegten Dienst.

3.1 Benutzerübergreifende Modelle

Zu den benutzerübergreifenden Modellen zählen das Modell der kontextfreien Grammatik von Weir et. al. sowie das Modell von Chou et. al.. Beide Modelle benötigen als Datengrundlage für die Trainingsphase eine hinreichend große Menge menschlich generierter Passwörter. Hierfür werden verschiedene Passwort-Leaks herangezogen. Das Ziel der Modelle ist es, möglichst viele Passwörter verschiedener Benutzer des Passwort-Leaks zu rekonstruieren.

3.1.1 Modell von Weir et. al.

Das von Weir et. al. [40] erstmalig etablierte Modell einer wahrscheinlichkeitsbasierten, kontextfreien Grammatik, beschreibt die Struktur von Passwörtern und bildet diese zur Generierung neuer Kandidaten nach. Die Beschreibung der Passwortstruktur erfolgt über die Erstellung der Passwort-Muster aus den Zeichentypen L, D und S. Das Passwort „$Password123“ beispielsweise besitzt die Basisstruktur $S_1L_8D_3$. Des Weiteren werden alle im Trainingsset auftretenden numerischen Fragmente sowie Sonderzei-

chen extrahiert. Die Basisstrukturen sowie die extrahierten Fragmente werden anhand ihrer Auftrittshäufigkeit gewichtet (Auftrittswahrscheinlichkeit).
In der anschließenden Testphase werden die identifizierten Basisstrukturen in absteigender Reihenfolge ihrer Auftrittswahrscheinlichkeit herangezogen und elementweise mit Fragmenten befüllt. Für die Muster-Elemente D und S wird hierfür auf die zuvor extrahierten Fragmente des Trainingsset zurückgegriffen. In absteigender Reihenfolge ihrer Auftrittswahrscheinlichkeit werden alle Fragmente gleicher Länge eingesetzt. Für die Substitution der alphabetischen Fragmente werden vordefinierte Wörterbücher herangezogen und alle Zeichenketten gleicher Länge verwendet. Den alphabetischen Fragmenten ist keine Erfolgswahrscheinlichkeit zugeordnet, sodass sie eine Auftrittswahrscheinlichkeit von 100% besitzen.
Die Erfolgswahrscheinlichkeit eines Kandidaten berechnet sich aus der Auftrittswahrscheinlichkeit der Basisstruktur sowie die der eingesetzten Fragmente.

Definition 3.1 *Sei $P(\beta_x)$ die Auftrittswahrscheinlichkeit des Musters β_x des Kandidaten x. Weiterhin sei $v_{l_i}(t_i)$ die Auftrittswahrscheinlichkeit des Fragments t_i an der Position l_i innerhalb der Basisstruktur der Länge l. Demzufolge ist die Erfolgswahrscheinlichkeit eines Kandidaten $P(x)$ gegeben durch:*

$$P(x) = P(\beta_x) \cdot \prod_{i=1}^{n} v_{l_i}(t_i) \tag{3.1}$$

Im Rahmen einer durchgeführten Studie konnte unter Anwendung des Präprozessors, eine Steigerung in der Anzahl der rekonstruierten Passwörtern um bis zu 129% im Vergleich zu der herkömmlichen Rekonstruktions-Anwendung „John the Ripper“[6] erzielt werden.

Ein Nachteil des Modells besteht in der Gleichbehandlung von Groß- und Kleinschreibung bei der Erstellung der Passwort-Muster. Daher entstehen für den Generierungsprozess sehr oberflächliche Strukturbeschreibungen. Das Passwort „$PasswordPassword123“ beispielsweise besitzt nach Weir et. al. die Basisstruktur $S_1L_{16}D_3$ und wird folglich mit alphabetischen Fragmenten der Länge 16 befüllt, anstatt mit zwei Wörtern der Länge 8. Der Präprozessor erkennt somit keine zwei aufeinanderfolgenden Fragmente derselben Fragment-Klasse, wodurch sie im Generierungsprozess nicht miteinander kombiniert werden.
Ein weiterer Nachteil besteht in der Begrenzung der zur Substitution herangezogenen Fragmente auf Elemente gleicher Länge. Demzufolge wird die Anzahl der generierten Passwort-Kandidaten sehr stark begrenzt.
Zuletzt ist die Erfolgswahrscheinlichkeit einer Rekonstruktion stark von den Passwörtern des zugrundeliegenden Trainingsdatensatzes abhängig. Der Grund hierfür ist zum einen, dass ausschließlich die darin auftretenden Strukturen nachgebildet werden. Zum

[6] John the Ripper ist neben Hashcat eine Anwendung zur Rekonstruktion von Passwörtern [31].

anderen sind die zur Befüllung herangezogenen D- und S-Fragmente auf die im Trainingssatz auftretenden numerischen Fragmente und Sonderzeichen begrenzt.

3.1.2 Modell von Chou et. al.

Bei dem Modell von Chou. et. al. [5] handelt es sich um eine Weiterentwicklung des Modells von Weir et. al., unter dessen Anwendung eine Steigerung der Rekonstruktionsleistung im Vergleich zu der herkömmlichen Rekonstruktions-Anwendung „John the Ripper“ um bis zu 273% erzielt werden konnte.

Die erste Erweiterung besteht in der Definition zusätzlicher Muster-Elemente zur detaillierten Beschreibung der Passwortstrukturen. Das Element U kennzeichnet demnach einen Großbuchstaben und das Muster-Element K repräsentiert ein Tastatur-Muster. Die zweite Erweiterungsmaßnahme besteht in der Vergrößerung des Wörterbuchs, welches zur Substitution der alphabetischen Muster-Elemente herangezogen wird. Dieses wird zum einen um die alphabetischen Fragmente des Trainingssets sowie um Zeichenketten von Tastatur-Mustern erweitert.
Die Generierung von Passwort-Kandidaten erfolgt analog zum Präprozessor von Weir et. al. über die Befüllung der einzelnen Muster-Elemente mit Fragmenten gleicher Länge der entsprechenden Fragment-Klasse.
Über die Definition eines Schwellenwerts, den die generierten Kandidaten zur Qualifikation als Passwort-Kandidaten übersteigen müssen, kann der zu testende Passwort-Kandidatenraum begrenzt werden. Die Berechnung der Erfolgswahrscheinlichkeit eines Kandidaten erfolgt analog zum Modell von Weir. et. al. nach Definition 3.1.

Wie bereits erwähnt, verarbeiten die Präprozessoren der beiden vorgestellten Modelle die Passwörter unabhängig von den zugehörigen Benutzern und zielen darauf ab möglichst viele Passwörter verschiedener Benutzer zu rekonstruieren. Aufgrund dieser Zielsetzung werden bei der Rekonstruktion keine benutzerspezifischen Faktoren, wie die individuelle Modifikation von Passwörtern durch den Benutzer oder die Integration von persönlichen Informationen berücksichtigt.

3.2 Benutzerspezifische Modelle

Der Fokus der benutzerspezifischen Modelle liegt nicht mehr auf der Anzahl an erfolgreich rekonstruierten Passwörtern, sondern auf der Rekonstruktion eines einzigen Passworts eines spezifischen Benutzers bei einem spezifischen Dienst. Die Datengrundlage besteht somit aus einem oder mehreren wenigen Passwörtern eines Benutzers sowie gegebenenfalls persönlicher Informationen über den Benutzer. Im Folgenden werden drei verschiedene, benutzerspezifische Präprozessoren vorgestellt, die auf Basis verschiedener Datengrundlagen mögliche Passwort-Kandidaten generieren.

3.2.1 Modell von Li et. al.

Li et. al. [27] untersucht zunächst im Rahmen einer Studie, auf welche Art und in welchem Umfang Benutzer persönliche Informationen in ihre Passwörter integrieren. Die Definition von persönlichen Informationen umfasst hierbei den Namen, das Geburtsdatum, den Benutzernamen, die Email-Adresse, die Telefonnummer sowie eine länderspezifische Identifikationsnummer[7]. Im Ergebnis konnte gezeigt werden, dass rund 60% der Passwörter des untersuchten Leaks mindestens eine der definierten persönlichen Informationen beinhalten.
Zur Abschätzung des Umfangs der Integration von persönlichen Informationen in ein Passwort definiert Li et. al. die „Coverage"-Messgröße („CVG"). Diese gibt den Grad der Korrelation zwischen einem Passwort und der detektierten persönlichen Information des zugehörigen Benutzers an. Zur Berechnung der Messgröße werden alle Zeichenketten eines Passworts, die eine der definierten persönlichen Informationen enthalten und eine Länge größer als 2 aufweisen, identifiziert (im Folgenden „Segmente" genannt). Im Anschluss wird der Coverage-Wert unter Anwendung der nachfolgenden Formel berechnet.

Definition 3.2 *Sei n die Anzahl an Segmenten und l_i die Länge des Segments i. Weiterhin sei L die Passwortlänge. Demzufolge ist ein Maß für den Grad der Korrelation zwischen einem Passwort und der persönlichen Information gegeben durch:*

$$CVG = \sum_{i=1}^{n} \frac{l_i^2}{L^2} \tag{3.2}$$

wobei $CVG \in [0, 1]$.

Ein steigender Coverage-Wert zeigt eine zunehmende Stärke der Korrelation zwischen dem Passwort und der persönlichen Information. Zu berücksichtigen gilt es, dass neben der Abhängigkeit des Coverage-Werts von der Anzahl an identifizierten Segmenten, eine Abhängigkeit von der Länge der Segmente besteht. Demnach ergibt beispielsweise ein Segment, das aus 6 Zeichen besteht, einen größeren Korrelationsgrad als zwei Segmente der Länge 3 unabhängig von der Gesamtlänge des Passworts.

In Bezug auf den Präprozessor des Modells stellen die definierten Arten von persönlichen Informationen zusätzliche Muster-Elemente dar, die wie folgt definiert sind:

- B für ein Geburtsdatum.
- N für ein Namen.
- E für eine Email-Adresse.
- A für einen Benutzernamen.

[7] Die Datengrundlage bildet ein Passwort-Leak chinesischer Benutzer, denen eine eindeutige Identifikationsnummer zugeordnet ist.

- C für eine Telefonnummer.
- I für eine Identifikationsnummer.

Auf diese Weise wird im Vergleich zu den benutzerübergreifenden Modellen die Struktur von Passwörtern detaillierter beschrieben. Die nachfolgende Generierung von Passwort-Kandidaten auf Basis der erstellten Passwort-Muster erfolgt analog zu dem Modell von Weir et. al..

Unter Berücksichtigung der persönlichen Informationen eines Benutzers bei der Generierung von Passwort-Kandidaten konnte eine deutliche Erfolgssteigerung in der Passwort-Rekonstruktion erzielt werden. Während das Modell von Weir et. al. bei einem generierten Kandidatenraum von 500.000 Passwörtern rund 25% der Passwörter eines Testdatensatzes rekonstruierte, konnte der Präprozessor von Li. et al. eine Erfolgsrate von rund 45% erzielen.

3.2.2 Modell von Zhang et. al.

Das Modell von Zang et. al. [42] beschäftigt sich mit der Vorhersage des Passworts eines Benutzers unter Kenntnis des derzeit bei dem betrachteten Dienst verwendeten Passworts. Das Modell basiert auf der Annahme, dass Benutzer neue Passwörter durch systematische Modifikation(en) ihrer aktuellen bzw. früheren Passwörter generieren.
Das Erstellen möglicher Passwort-Kandidaten für einen spezifischen Benutzer erfolgt über eine Vielzahl definierter Transformationsregeln auf das vorliegende Passwort. Die Transformationsregeln beinhalten beispielsweise die Veränderung der Groß- und Kleinschreibung, das Löschen oder Duplizieren von Zahlen und Sonderzeichen oder eine Veränderung der Passwortstruktur.

Im Rahmen einer Studie mit 10.374 Benutzern einer universitätsinternen Plattform, die aufgrund einer Sicherheitsrichtlinie ihr Passwort in regelmäßigen Zeitabständen neu wählen, konnten 41% der aktuellen Passwörter, auf Grundlage der Bekanntheit eines vorherigen Passworts, innerhalb von 3 Sekunden rekonstruiert werden.

3.2.3 Modell von Wang et. al.

Das Modell von Wang et. al. [38] berücksichtigt bei der Generierung von Passwort-Kandidaten zum einen direkt-abbildbare persönliche Informationen des Benutzers (Modell 1) und zum anderen vom Benutzer bekannte Passwörter bei anderen Diensten (Modell 2).

Im Rahmen des ersten Modells werden analog zum Modell von Li et. al. zusätzliche Muster-Elemente zur Repräsentation von persönlichen Informationen auf Muster-Ebene definiert. Für eine noch detailliertere Strukturbeschreibung der Passwörter, werden diese noch feiner untergliedert. Das Muster-Element N beispielsweise, welches einen Namen im Passwort repräsentiert, wird unter anderem in die Muster-Klassen $N3$ für einen Nachnamen und $N4$ für einen Vornamen unterteilt. Weiterhin wird beispielsweise das Muster-Element B für ein Geburtsdatum in Abhängigkeit der Anordnung von Tag, Monat und Jahr in 10 verschiedene Darstellungsvarianten untergliedert. Folglich geben die Muster-Klassen detaillierte Informationen über den Inhalt sowie die Schreibweise des zugehörigen Fragments wieder.
Die Generierung von Passwort-Kandidaten findet analog zum Modell von Weir et. al. mit dem Unterschied statt, dass die Befüllung der personenbezogenen Muster-Elemente nicht auf Fragmente gleicher Länge begrenzt sind.
Im Ergebnis konnte das Modell im Rahmen eines getesteten Kandidatenraums von $10 \sim 10^3$ Passwörtern, bis zu rund 73% mehr Passwörter als das Modell von Li et. al. rekonstruieren.

Das zweite Modell von Wang et. al. generiert mögliche Passwort-Kandidaten eines Benutzers durch Anwendung festgelegter Transformationsregeln auf das vorliegende Passwort auf Zeichen- sowie Fragment-Ebene. Die Transformationsregeln umfassen die folgenden Modifikationen:

- Einfügen und Entfernen von Fragmenten.
- Veränderung der Groß- und Kleinschreibung.
- Leet-speak[8] Substitutionen.
- Rotation von Fragmenten.
- Rückwärts schreiben eines Fragments.
- Einfügen und Entfernen einzelner Zeichen eines Fragments.

Die Berechnung der Erfolgswahrscheinlichkeit der generierten Kandidaten basiert auf den Ergebnissen einer zuvor durchgeführten Analyse eines Passwort-Leaks zur Bestimmung der Anwendungshäufigkeit der definierten Modifikationen. Die Häufigkeit der Anwendung durch die Benutzer des Leaks repräsentiert die Beliebtheit der jeweiligen Modifikation.
Die Datengrundlage der Untersuchung bilden jeweils zwei Passwörter eines Benutzers. Die definierten Transformationen werden nacheinander auf eines der Passwörter angewendet und in Folge jeweils überprüft, ob die vorgenommene Modifikation zu einer Annäherung der Passwörter führt. Als Messgröße zur Bestimmung einer Annäherung wird die Levenshtein-Distanz (siehe Abschnitt 2.6) herangezogen, die den Übereinstimmungsgrad zwischen den beiden Passwörtern misst. Ein gestiegener Ähnlichkeitsgrad

[8] Hierbei handelt es sich um eine Sprache im Internet-Jargon, die sich durch Substitution von Buchstaben durch ähnlich aussehende Ziffern und Sonderzeichen auszeichnet [25].

wird als „vom Benutzer angewendete Modifikation“ interpretiert und beibehalten. Andernfalls wird der generierte Passwort-Kandidat verworfen und die nächsten Transformation durchgeführt. In Abhängigkeit der gesamten Anwendungshäufigkeit über alle Benutzer des Leaks werden die Transformationen gewichtet und bestimmen auf diese Weise die Erfolgswahrscheinlichkeit der generierten Passwort-Kandidaten.

Im Ergebnis konnte das Modell im Rahmen eines getesteten Kandidatenraums von 100 Passwörtern durchschnittlich rund 111% mehr Passwörter als das vergleichbare Modell von Das et. al. [6] rekonstruieren.

4 Analyse menschlicher Passwortgenerierung

Wie bereits in den Grundlagen beschrieben, geben Leak-Listen sensible Passwortinformationen einer Vielzahl verschiedener Benutzern preis. Hierdurch lassen sich Rückschlüsse auf menschliche Konstruktionsprinzipien bei Passwörtern ziehen. Zudem können beliebte Modifikationen, die zur Wiederverwendung eines Passworts vorgenommen werden, offengelegt werden.
In Verfolgung dieses Ziels wird im Folgenden ein Passwort-Leak verschiedener Email-Dienste herangezogen und mit dem Fokus auf die Wiederverwendung von Passwörtern von den Benutzern sowie der Integration persönlicher Informationen in ein Passwort analysiert. Die gewonnenen Erkenntnisse finden bei der Generierung benutzerspezifischer Passwort-Kandidaten im Rahmen des in Kapitel 5 vorgestellten Präprozessors Anwendung.

4.1 Datengrundlage

Der Passwort-Leak beinhaltet im rekonstruierten Zustand Accountinformationen in Form der Email-Adresse eines Benutzers und dem dazugehörigen Passwort in Klartext. Informationen über die Entstehung des Leaks sowie über gegebenenfalls vorgenommene Einschränkungen im Rahmen der Rekonstruktion der Passwörter liegen nicht vor. Für die folgende Untersuchung wird die Passwortsammlung als gegeben und vollständig angenommen. Aus datenschutzrechtlichen Gründen sowie mit dem Ziel der Anonymisierung der Benutzer wird der domainspezifische Teil der Email-Adresse entfernt. Die Zuordnung der Passwörter zu einem Benutzer erfolgt über den Benutzernamen bei dem jeweiligen Dienst.
Bei den betroffenen Benutzern handelt es sich aufgrund der weltweit tätigen Dienstanbieter um Personen internationaler Herkunft. Da die Designprinzipien für Passwörter in Abhängigkeit von der verwendeten Sprache stark variieren, finden für eine konsistente Analyse ausschließlich die Passwörter von Benutzern aus dem deutschsprachigen Raum Berücksichtigung. Ein Indiz für die Herkunft der Benutzer ist das Länderkürzel der zugehörigen Email-Adresse. Demzufolge wird die Analyse auf die Passwörter von Email-Adressen aus Deutschland, Österreich und der Schweiz begrenzt.

Im Ergebnis liegen $51.023.099$ Datensätze, bestehend aus dem Benutzernamen und dem Klartext-Passwort, vor. Ein mehrfaches Auftreten eines Benutzernamens ist als Nutzung mehrerer, unterschiedlicher Email-Dienste zu interpretieren. Für eine konsistente und einheitliche Datengrundlage wird die Passwortliste zunächst um fehlerhafte Einträge bereinigt.

Der Bereinigungsprozess umschließt das Entfernen von Einträgen mit fehlender Passwortangabe (18.621 Datensätze) sowie mit leeren Passworteinträgen (69 Datensätze). Des Weiteren werden die Einträge, bei denen keine klare Trennung zwischen dem Benutzernamen und dem Passwort möglich ist ausgeschlossen (589 Datensätze). Zuletzt werden Benutzer mit identischen „Passwortsets“, also der Übereinstimmung aller Passwörter, eliminiert. Ihr Auftreten im Leak spricht für eine automatisierte Accounterstellung unter Verwendung eines Skripts mit voreingestellten Passwörtern. Diese Sets werden nur ein einziges Mal berücksichtigt. Demzufolge werden alle Benutzer, deren Passwörter sich nicht mindestens zu 80% von den identifizierten Passwortsets unterscheiden, eliminiert ($5.696.058$ Datensätze).

Die resultierende Datengrundlage bildet die Passwortsammlung 1 (siehe nachfolgende Auflistung). Im Verlauf der Analyse werden die folgenden unterschiedlichen Datengrundlagen herangezogen:

1. Passwortsammlung: Alle Datensätze inklusive Passwort-Duplikate je Benutzer. Insgesamt $45.307.762$ Passwörter von $19.637.876$ unterschiedlichen Benutzern.
2. Passwortsammlung: Alle Datensätze exklusive Passwort-Duplikate je Benutzer. Insgesamt $25.509.924$ Passwörter von $19.637.876$ unterschiedlichen Benutzern.
3. Passwortsammlung: Alle Datensätze von Benutzern, von denen mindestens zwei Passwörter gegeben sind inklusive Passwort-Duplikate. Insgesamt $33.238.240$ Passwörter von $7.568.355$ unterschiedlichen Benutzern.
4. Passwortsammlung: Alle Datensätze von Benutzern, von denen mindestens zwei Passwörter gegeben sind exklusive Passwort-Duplikate. Insgesamt $8.970.971$ Passwörter von $3.098.923$ unterschiedlichen Benutzern.

Bei keiner expliziten Angabe der zugrundeliegenden Datengrundlage handelt es sich um die 2. Passwortsammlung. Im Folgenden wird zunächst mit dem Ziel der Identifizierung beliebter Passwörter bzw. beliebter Passwort-Elemente eine inhaltliche Analyse des Passwort-Leaks durchgeführt.

4.2 Inhaltliche Analyse

Wie bereits in Abschnitt 2.4.1 erläutert, verwenden Benutzer aufgrund ihres eingeschränkten, menschlichen Erinnerungsvermögen gerne einfache, leicht zu merkende Passwörter. Die resultierende, niedrige Passwortsicherheit führt zu einer geringen Resistenz des Passworts gegenüber Rekonstruktions-Angriffen. Beinhaltet ein Passwort beispielsweise beliebte Zeichenketten (sogenannte „Standard-Passwörter“), sind diese unter geringem Aufwand über einen Wörterbuch-Angriff zu rekonstruieren.

4.2.1 Passwort-Ebene

Mit dem Ziel, die Anzahl an Benutzern zu messen, die Standard-Passwörter verwenden sowie die Anzahl von Passwörtern zu messen, die über einen gängigen Wörterbuch-Angriff rekonstruiert werden könnten, werden im Folgenden Wörterbuch-Angriffe unter Verwendung drei bekannter Wörterbücher unterschiedlicher Größe (siehe Tabelle 4.1) durchgeführt.

Zunächst wird ein Wörterbuch-Angriff zur Identifizierung der übereinstimmenden Passwörter zwischen dem Passwort-Leak und dem jeweiligen Wörterbuch durchgeführt. Im Anschluss wird unter Verwendung eines python-Skripts die Auftrittshäufigkeit der identifizierten Passwörter sowie die Anzahl an Benutzern, die eines der identifizierten Standard-Passwörter verwenden, bestimmt. Tabelle 4.1 zeigt die resultierenden Ergebnisse.[9]

Tabelle 4.1: Ergebnisse von Wörterbuch-Angriffen

	Top12k	Rockyou	Pkf_6-32char
Anzahl an Einträgen	12.644	14.344.391	3.032.835.983
Anzahl an übereinstimmenden Passwörtern	1.789	1.203.347	16.223.949
Anzahl an rekonstruierten Passwörtern	105.058	7.126.189	24.506.161
Anzahl an Benutzern	103.061	6.284.731	18.881.543
Rekonstruierte Passwörter (in %)	0,4118	27,935	96,0652
Benutzer (in %)	5,2481	32,0031	96,1486[10]

Quelle: eigene Darstellung.

Wie zu erkennen ist, sind unter Verwendung eines relativ kleinen Wörterbuchs wie dem „Top12k“ mit 12.644 Einträgen lediglich 105.058 Passwörter des Leaks rekonstruierbar. Dies entspricht rund $0,41\%$ der insgesamt $25.509.924$ Passwörter. Bei Bezug eines sehr umfangreichen Wörterbuchs wie dem „pkf_6-32char“ mit insgesamt $3.032.835.983$ Einträgen lassen sich hingegen bereits $24.506.161$ Passwörter rekonstruieren. Dies entspricht rund $96,07\%$ aller Passwörter des Passwort-Leaks. Bei Betrachtung auf Benutzer-Ebene handelt es sich hierbei um insgesamt $18.881.543$ Benutzer ($96,15\%$), die mindestens ein Standard-Passwort bei einem Dienst verwenden.

[9] Die Angriffsausführung erfolgt über Hashcat im Klartext-Modus $m = 99999$ auf einer “Geforce GTX 1070“-Grafikkarte.

[10] Hierbei handelt es um einen Schätzwert auf Basis von 500.000 Benutzern, von denen 480.743 mindestens ein Standard-Passwort verwenden.

Die 10 am häufigsten verwendeten Passwörter, die alle unter die Kategorie der Standard-Passwörter fallen, sind in Tabelle 4.2 aufgelistet. Bei über der Hälfte der Passwörter handelt es sich um einfache Ziffernfolgen, die aufgrund ihrer Zusammensetzung aus lediglich einem Muster-Klassen-Element[11] (D^+) eine besonders niedrige Komplexität aufweisen. Zudem stellen sie neben dem Passwort „qwerty“[12] aufgrund ihrer benachbarten Anordnung auf der Tastatur sogenannte „Tastatur-Muster“ dar. Der Grund für ihre häufige Verwendung ist in der Möglichkeit einer besonders schnellen Passworteingabe zu vermuten. Zu den beliebtesten alphabetischen Passwörtern zählen die Zeichenketten „passwort“ und „password“.

Tabelle 4.2: Passwörter höchster Auftrittshäufigkeit

Rang	Passwort	Häufigkeit	Anteil (in $\%$)
1	123456	78.400	$0,3073$
2	123456789	33.405	$0,1309$
3	12345	16.791	$0,0658$
4	qwerty	15.955	$0,0625$
5	12345678	14.089	$0,0523$
6	passwort	10.982	$0,0430$
7	1234567	9.229	$0,0362$
8	hallo123	8.082	$0,0317$
9	password	8.048	$0,0315$
10	1234	8.029	$0,0315$

Quelle: eigene Darstellung.

Zur Identifizierung weiterer, inhaltlicher Gemeinsamkeiten zwischen den Passwörtern der verschiedenen Benutzer werden die Passwörter im Folgenden auf Fragment-Ebene betrachtet.

[11] Siehe Definition 2.9.

[12] Bei dem US-amerikanischen Tastatur-Layout befinden sich die Buchstaben „Z“ und „Y“ an vertauschten Positionen.

4.2.2 Fragment-Ebene

Die Verteilung der Fragment auf die Fragment-Klassen in Tabelle 4.3 zeigt, dass es sich, wie zu erwarten, bei der Mehrheit der Fragmente mit $60,69\%$ um alphabetische Zeichenketten handelt. Die numerischen Fragmente folgen mit $35,87\%$ und die Sonderzeichen sind mit $3,44\%$ am seltensten ein Bestandteil eines Passworts.

Tabelle 4.3: Klassenverteilung der Fragmente

	Häufigkeit	Anteil (in %)	Anteil an Duplikaten (in %)
Alphabetische Fragmente	29.837.303	60,69	71,23
Numerische Fragmente	17.635.235	35,87	94,19
Sonderzeichen	1.691.804	3,44	99,81
$\sum$	49.164.342	100	

Quelle: eigene Darstellung.

Mit Blick auf den Anteil an Duplikaten pro Fragment-Klasse ist zu beobachten, dass die numerischen Fragmente und die Sonderzeichen mit $94,19\%$ bzw. $99,81\%$ weitaus mehr Duplikate als die alphabetischen Fragmente mit $71,23\%$ beinhalten. Dies ist auf den kleineren Zeichenraum der beiden Fragment-Klassen mit 10 bzw. 33 Zeichen im Vergleich zum Alphabet, das aus insgesamt 52 klein- und großgeschriebenen Buchstaben besteht, zurückzuführen.
Auffällig ist jedoch, dass der Anteil an Duplikaten bei den Sonderzeichen um mehr als 5 Prozentpunkte über dem Anteil der numerischen Fragmente liegt, obwohl diese aus einem größeren Zeichenraum bestehen. Dies lässt ähnliche Vorlieben der Benutzer bei der Verwendung von Sonderzeichen in einem Passwort vermuten. Aus diesem Grund werden im Folgenden die Passwortinhalte pro Fragment-Klasse betrachtet.

4.2.2.1 Alphabetische Fragmente

Die enorme Vielfalt der alphabetischen Fragment-Klasse wird bei Betrachtung der Tabelle 4.4 nochmals verdeutlicht. Sie zeigt die beliebtesten alphabetischen Passwort-Elemente mit entsprechender Auftrittshäufigkeit sowie dem prozentualem Anteil an der Gesamtheit der alphabetischen Fragment-Klasse.
Obwohl es sich hierbei um die am häufigsten verwendeten alphabetischen Fragmente handelt, besitzen sie jeweils nur eine prozentuale Auftrittshäufigkeit zwischen 0.04% und 0.14%. Dies unterstreicht nochmals den großen Raum von alphabetischen Zeichenketten, der grundsätzlich aus 52^x Möglichkeiten besteht, wobei x die Länge des Fragments angibt.

Tabelle 4.4: Alphabetische Fragmente höchster Auftrittshäufigkeit

Rang	Fragment	Häufigkeit	Anteil (in %)
1	qwerty	41.328	0.13854
2	hallo	39.358	0.1319
3	passwort	24.009	0,0805
4	ever	20.417	0,0684
5	love	18.962	0,0636
6	abc	18.959	0,0635
7	the	18.580	0,0623
8	alex	18.367	0,0616
9	daniel	17.595	0,05897
10	qwe	17.369	0,0582
11	ficken	16.279	0,0546
12	killer	15.872	0,0531
13	life	15.722	0,0527
14	lol	15.551	0,0521
15	master	15.433	0,0517
16	michael	14.941	0,0501
17	chris	14.559	0,0488
18	thomas	13.783	0,0462
19	schatz	13.239	0,0444
20	password	13.052	0,044

Quelle: eigene Darstellung.

Bei inhaltlicher Betrachtung handelt es sich bei der Mehrheit der Wörter um typische Passwort-Elemente. Wörter wie „hallo“, „love“ oder „life“ sind stets Bestandteil von Standard-Passwörtern. Auffällig ist jedoch, dass bereits in den Top-20 der alphabetischen Fragmenten 5 männliche Vornamen enthalten sind. Dieser Aspekt wird in Abschnitt 4.6.2 nochmals aufgegriffen und näher analysiert.

4.2.2.2 Numerische Fragmente

Bei der Wahl des numerischen Fragments für ein Passwort zeigen sich im Vergleich zu den alphabetischen Fragmenten benutzerübergreifende Vorlieben. Wie Tabelle 4.5 zu entnehmen ist, stellt die Ziffer „1“ mit einem prozentualen Anteil von $10,21\%$ das mit Abstand beliebteste numerische Fragment dar. Auffällig ist zudem, dass es sich fast ausschließlich um einstellige Zahlen handelt (eine Ausnahme bildet die Ziffernfolge „123“).

Tabelle 4.5: Numerische Fragmente höchster Auftrittshäufigkeit

Rang	Fragment	Häufigkeit	Anteil (in %)
1	1	1.801.060	10,2128
2	2	621.437	3,5238
3	3	559.677	3,1736
4	123	558.588	3,1675
5	4	525.697	2,9809
6	7	377.932	2,1431
7	5	373.261	2,1166
8	0	362.734	2,0569
9	6	361.668	2,0508
10	8	338.837	1,9214

Quelle: eigene Darstellung.

4.2.2.3 Sonderzeichen-Fragmente

Auch bei der Wahl der Sonderzeichen lassen sich identische Verhaltensweisen zwischen den Benutzern beobachten. Bei exklusiver Betrachtung der ersten 5 beliebtesten Sonderzeichen in Tabelle 4.6 sind bereits rund 85% aller in den Passwörtern verwendeten Sonderzeichen abgedeckt. Das mit Abstand am meisten verwendete Sonderzeichen ist das Fragment „_“ mit einer prozentualen Auftrittshäufigkeit von $46,02\%$. Die Fragmente „.“ und „!“ folgen mit $16,93\%$ bzw. $11,1\%$.

Tabelle 4.6: Sonderzeichen höchster Auftrittshäufigkeit

Rang	Fragment	Häufigkeit	Anteil (in %)
1	_	778.570	46,0201
2	.	286.474	16,9330
3	!	187.706	11,0950
4	-	149.664	8,8464
5	@	27.943	1,6517
6	$	26.938	1,5923
7	[	16.535	0,9774
8	*	14.094	0,8331
9	?	12.932	0,7644
10	]	12.848	0,7594

Quelle: eigene Darstellung.

Entgegen den Erwartungen befinden sich die Sonderzeichen „[“ und „] “ auf dem siebten und zehnten Rang und werden damit häufiger in einem Passwort verwendet als alltagsübliche Zeichen wie beispielsweise dem Pluszeichen „+“ (Rang 11).

Auffällig ist zudem, dass alle 10 beliebtesten Fragmente der Sonderzeichen-Klasse lediglich eine Länge von einem Zeichen besitzen. Dies führt zu der Annahme, dass ein Benutzer, wenn überhaupt, lediglich ein einziges Sonderzeichen in sein Passwort integriert.

4.3 Analyse der Längen

Zur Bestätigung dieser Vermutung sowie zur Identifizierung weiterer menschlicher Konstruktionsprinzipien werden im Folgenden die Längen der Passwörter sowie der einzelnen Fragmente analysiert. Zudem werden ihre Einflussfaktoren sowie ihre Einflussnahme auf die Passwortkomplexität untersucht. Zuletzt findet eine Analyse der Passwortlängen hinsichtlich des Wiederverwendungsaspekts statt.

4.3.1 Passwort-Ebene

Bei Betrachtung des gesamten Passworts liegt die minimal auftretende Länge bei einem Zeichen. Die maximale Passwortlänge beträgt 128 Zeichen. Durchschnittlich haben die Passwörter eine Länge von $8,766$, also rund 9 Zeichen. Die mediale Länge liegt bei 8 Zeichen und ist mit einem prozentualen Anteil von rund 33% an der Gesamtheit der Passwörter die am häufigsten auftretende Passwortlänge (siehe Tabelle 4.7).

Tabelle 4.7: Passwortlängen höchster Auftrittshäufigkeit

Rang	Anzahl an Zeichen	Häufigkeit	Anteil (in %)
1	8	8.415.228	32,9881
2	10	3.305.376	12,9572
3	9	3.207.692	12,5743
4	6	2.980.025	11,6818
5	7	2.707.502	10,6135
6	11	1.101.473	4,3178
7	12	823.138	3,2267
8	13	495.184	1,9411
9	5	493.885	1,9360
10	16	427.135	1,6744

Quelle: eigene Darstellung.

Weitere beliebte Gesamtlängen besitzen zwischen 7 und 10 Zeichen. Sie weisen jeweils eine prozentuale Auftrittshäufigkeit von über 10% auf.

Bei Eingrenzung der Passwortlängen auf 1 bis 6 Zeichen resultieren $3.952.977$ Passwörter, was einem prozentualen Anteil von $15,5\%$ entspricht. Diese Passwörter sind aufgrund ihrer Kürze oftmals ohne größeren Ressourcenaufwand über einen Bruteforce-Angriff rekonstruierbar.[13]

4.3.2 Fragment-Ebene

Im Folgenden werden die Passwörter erneut auf Fragment-Ebene betrachtet und bezüglich ihrer Länge analysiert.

4.3.2.1 Alphabetische Fragmente

In Tabelle 4.8 sind die 10 zahlreichsten Längen alphabetischer Fragmente mit entsprechender Auftrittshäufigkeit sowie ihrem prozentualen Anteil an der Gesamtheit der alphabetischen Fragment-Klasse abgebildet. Die minimale Länge beträgt 1 Zeichen und die maximale Fragmentlänge 91 Zeichen. Durchschnittlich haben die Fragmente eine Länge von $8,715$, also rund 9 Zeichen. Der Median liegt bei 6 Zeichen.

Tabelle 4.8: Alphabetische Fragmentlängen höchster Auftrittshäufigkeit

Rang	Anzahl an Zeichen	Häufigkeit	Anteil (in %)
1	8	5.560.509	18,6361
2	6	4.221.119	14,1471
3	1	3.820.282	12,8037
4	5	2.877.735	9,6448
5	7	2.745.196	9,2006
6	4	2.468.310	8,2726
7	3	2.135.395	7,1568
8	2	2.103.034	7,0483
9	9	1.419.051	4,756
10	10	909.292	3,0475

Quelle: eigene Darstellung.

Die Länge von 8 Zeichen ist analog zu den vollständigen Passwörtern (siehe Tabelle 4.7) die meist verwendete Länge bei alphabetischen Fragmenten. In absteigendem Rang sind alle Längen von 1 bis 10 Zeichen vertreten. Auffällig ist, dass die Länge von einem Zeichen mit $12,8\%$ die dritt-häufigste Länge ist. Dies führt zu der Annahme, dass Benutzer neben Wörtern gerne einzelne Buchstaben als ein Fragment in ihr Passwort integrieren.

[13] Der Ressourcenaufwand bestimmt sich in Abhängigkeit der Leistungsfähigkeit der Hardware sowie der Komplexität der Hashfunktion (siehe Abschnitt 2.5.1).

4.3.2.2 Numerische Fragmente

In Tabelle 4.9 sind die 10 am häufigsten verwendeten numerischen Fragmentlängen abgebildet. Die minimale Länge beträgt 1 Zeichen und die maximale Fragmentlänge 52 Zeichen. Durchschnittlich haben die Zahlen eine Länge von $2,903$, also ca. 3 Zeichen. Der Median liegt bei 2 Zeichen.

Tabelle 4.9: Numerische Fragmentlängen höchster Auftrittshäufigkeit

Rang	Anzahl an Zeichen	Häufigkeit	Anteil (in %)
1	1	5.634.441	31,9499
2	2	4.596.469	26,0641
3	4	2.378.865	13,4893
4	3	2.040.371	11,5699
5	6	1.095.129	6,2099
6	8	664.137	3,76597
7	5	524.086	3,2267
8	7	287.966	2,9718
9	9	164.618	0,9335
10	10	138.533	0,7855

Quelle: eigene Darstellung.

Mit einem prozentualen Anteil von rund $31,95\%$ verwenden die Benutzer am häufigsten Zahlen der Länge 1. Dies unterstreicht die in Abschnitt 4.2.2.2 getroffene Annahme, dass Benutzer bevorzugt einstellige Zahlen in ihre Passwörter integrieren. Bei der zweit beliebtesten Fragmentlänge handelt es sich mit einem Anteil von $26,06\%$ ebenfalls um ein kurzes Fragment, bestehend aus 2 Ziffern.
Interessant zu beobachten ist, dass die Zahlen mit dem Zeichenlängen 4, 6 und 8 häufiger verwendet werden als Zahlen ihrer kürzeren Vorgänger 3, 5 und 7. Ein möglicher Grund für die Beliebtheit dieser numerischen Zeichenlängen könnte die Abbildung von Geburtsdaten in einem Passwort sein. In Abhängigkeit von ihrer Darstellungsweise besitzen diese eine Länge von 4, 6 oder 8 Zeichen. Dieser Aspekt wird in Abschnitt 4.6.3 näher analysiert.
58% der numerischen Fragmente sind durch die Zahlen mit den Längen 1 und 2 abgedeckt. Bei Ausweitung auf 4 Zeichen, beträgt die Abdeckung 83%. Mit hoher Wahrscheinlichkeit verwenden also Benutzer ein- bis vierstellige Zahlen bei der Integration in das jeweilige Passwort.

4.3.2.3 Sonderzeichen-Fragmente

In Tabelle 4.10 sind die 5 häufigsten Längen der Fragmente mit Sonderzeichen abgebildet. Die minimale Länge, der Median sowie die durchschnittliche Länge liegen bei einem bzw. $1,077$ Zeichen.

Tabelle 4.10: Sonderzeichen-Fragmentlängen höchster Auftrittshäufigkeit

Rang	Anzahl an Zeichen	Häufigkeit	Anteil (in %)
1	1	1.592.650	94,1392
2	2	79.784	4,7159
3	3	14.863	0,8785
4	4	1.960	0,1159
5	6	830	0,0464

Quelle: eigene Darstellung.

Wie zu erkennen ist, besitzen $94,14\%$ der von den Benutzern verwendeten Sonderzeichen-Fragmenten eine Länge von 1. Mit hoher Wahrscheinlichkeit verwenden also Benutzer einstellige Sonderzeichen bei der Integration in das jeweilige Passwort, was die in Abschnitt 4.2.2.3 getroffene Vermutung bestätigt.
Bei exklusiver Betrachtung der Fragmentlängen 1 und 2 sind bereits rund 99% der Länge der verwendeten Sonderzeichen abgedeckt. Fragmente mit größerer Länge treten mit einer prozentualen Auftrittshäufigkeit von jeweils unter 1% nur in Ausnahmefällen auf.

4.3.3 Schlussfolgerungen

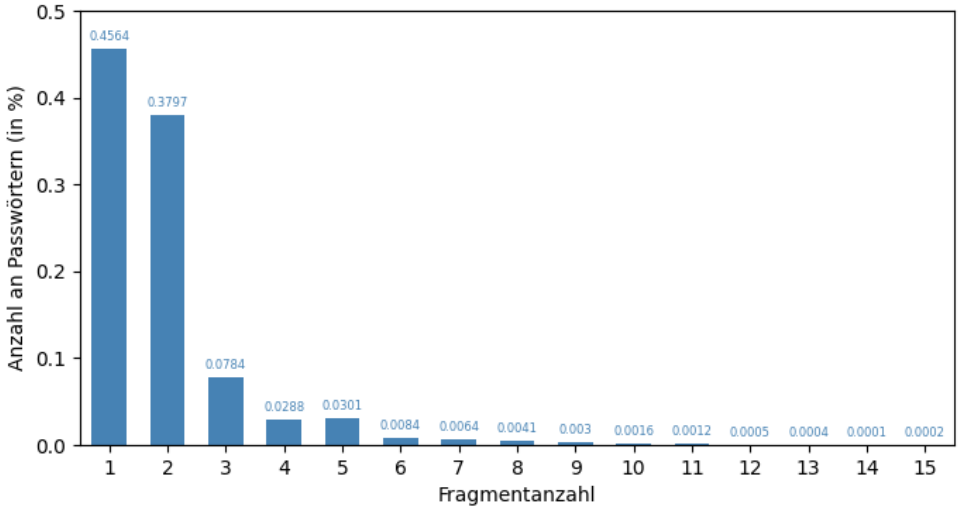
Auf Grundlage der soeben durchgeführten Analyse der Passwortlängen lassen sich Rückschlüsse auf die Komplexität der Passwörter des Leaks ziehen, die im Folgenden erläutert werden.

4.3.3.1 Passwortkomplexität

Wie in den Grundlagen in Abschnitt 2.4.1 erläutert, ergibt sich die Komplexität eines Passworts aus der Größe des verwendeten Zeichensatzes sowie aus der Zusammensetzung aus unterschiedlichen Zeichentypen.
Eine Aussage über die Größe des verwendeten Zeichensatzes lässt sich unter Betrachtung der Fragmentanzahl der Passwörter treffen. Grundsätzlich ist die Anzahl der Zeichentypen gleich hoch wie die Anzahl der Fragmente bei Berücksichtigung einer Obergrenze von 4 Zeichentypen. Die Ausnahme hiervon bildet eine veränderte Groß- und Kleinschreibung, wodurch die Anzahl der verwendeten Zeichentypen um 1 größer ist.

Insgesamt bestehen die $25.509.924$ Passwörter aus $49.164.342$ einzelnen Fragmenten. Die minimal auftretende Fragmentanzahl eines Passworts beträgt 1 Fragment und die maximale Fragmentanzahl liegt bei 63 Fragmenten. Die durchschnittliche Fragmentanzahl eines Passworts liegt bei $1,9273$, also ca. 2 Fragmenten, was dem Median entspricht. In Abbildung 4.1 ist die Häufigkeitsverteilung der Fragmentanzahl der Passwörter bis zu 15 Fragmenten abgebildet.

Abbildung 4.1: Häufigkeitsverteilung der Fragmentanzahl



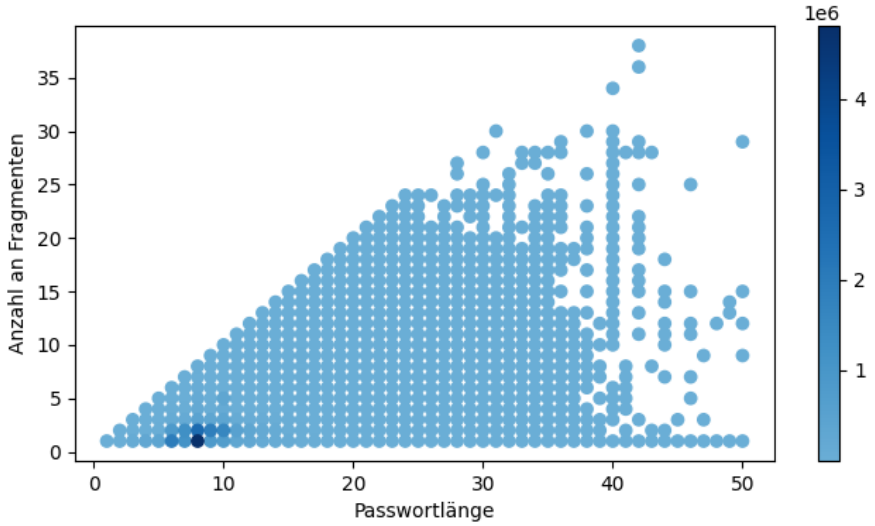
Quelle: eigene Darstellung.

Wie zu erkennen ist, bestehen rund 84% der Passwörter lediglich aus einem oder zwei Fragmenten. Hieraus lässt sich schlussfolgern, dass lediglich 2 bzw. maximal 3 verschiedene Zeichentypen enthalten sind. Die Mehrheit der Benutzer verwenden damit nicht den vollständigen ASCII-Zeichenraum aus Tabelle 2.1, wodurch ihre Passwörter nicht die maximale Komplexität besitzen.

Eine Aussage über die Zusammensetzung eines Passworts lässt sich unter Betrachtung des Zusammenhangs zwischen der Anzahl der Fragmente und der Gesamtlänge des Passworts treffen. Grundsätzlich besitzt ein Passwort unter der Annahme der Verwendung des vollständigen Zeichensatzes die maximale Komplexität, wenn es aus gleich vielen Fragmenten besteht, wie es Zeichen hat.
So hat das Passwort „h1a2l3l4o“ mit 9 Fragmenten im Vergleich zu dem Passwort „hallo1234“ mit 2 Fragmenten eine maximale Komplexität, obwohl beide Passwörter aus den gleichen Zeichen bestehen.

In Abbildung 4.2 ist der Zusammenhang zwischen der Anzahl an Fragmenten und der Passwortlänge bis zu 50 Zeichen dargestellt.

Abbildung 4.2: Zusammenhang zwischen der Passwortlänge und der Fragmentanzahl



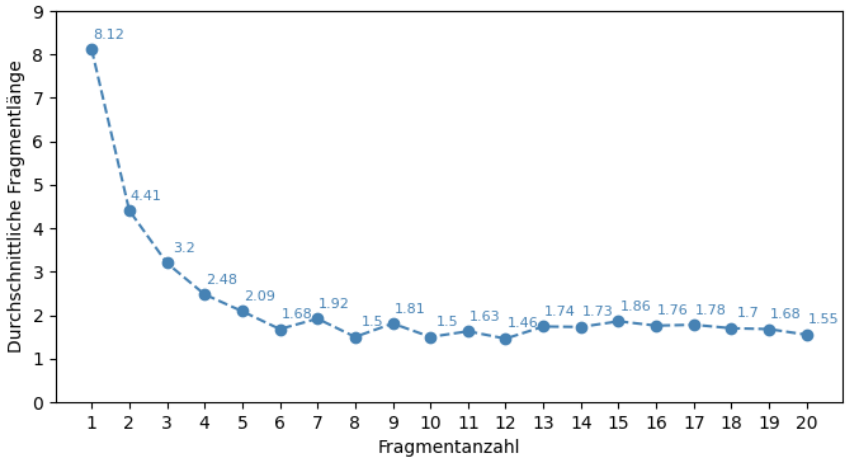
Quelle: eigene Darstellung.

Die Passwörter mit maximaler Komplexität bezüglich ihrer Zusammensetzung befinden sich auf der Hauptdiagonalen. Ihre Anzahl an Fragmenten entspricht ihrer Zeichenlänge. Wie farblich deutlich zu erkennen ist, liegt der Schwerpunkt der Passwörter weit unterhalb dieser Hauptdiagonalen, nämlich bei der Passwortlänge von 8 Zeichen mit 2 Fragmenten. Damit schöpft ein Großteil der Passwörter nur 20% der maximal möglichen Passwortkomplexität aus.

4.3.3.2 Fragmentlängen

Neben der soeben dargestellten Abhängigkeit zwischen der Anzahl an Fragmenten und der Passwortlänge beeinflusst die Fragmentanzahl weiterhin die Länge der einzelnen Fragmente (siehe Abbildung 4.3). Mit steigender Fragmentanzahl sinkt die durchschnittliche Anzahl an Zeichen pro Fragment. Je mehr Fragmente dahein ein Passwort besitzt, desto größer ist zwar die Gesamtlänge des Passworts. Dies geht jedoch einher mit kurzen Zeichenlängen der einzelnen Fragmente in dem Passwort. Auf Basis dieses Zusammenhangs findet die Verarbeitung der Passwort-Fragmente durch den Präprozessor (siehe Kapitel 5) pro Muster-Klasse statt.

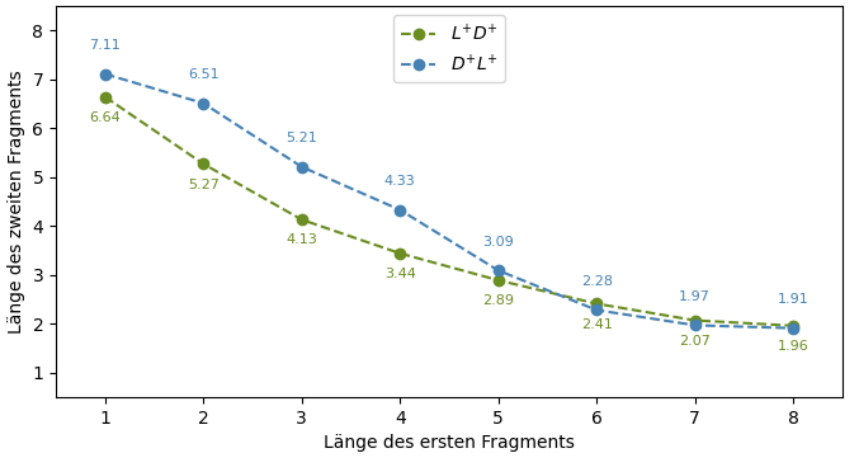
Abbildung 4.3: Zusammenhang zwischen der Fragmentanzahl und der Fragmentlänge



Quelle: eigene Darstellung.

Des Weiteren besteht eine Abhängigkeit der Längen zwischen den Fragmenten eines Passworts. Abbildung 4.4 zeigt am Beispiel der Muster-Klassen L^+D^+ und D^+L^+ den Verlauf der Fragmentlängen an erster sowie zweiter Position innerhalb des Passworts.

Abbildung 4.4: Zusammenhang zwischen den Fragmentlängen



Quelle: eigene Darstellung.

Wie zu beobachten ist, verkürzt sich das zweite Fragment mit zunehmender Länge des ersten Fragments, unabhängig davon, ob es sich um ein alphabetisches oder numerisches Fragment handelt. Da die Eingabe eines Passworts stets in der Reihenfolge vom ersten bis zum letzten Fragment erfolgt, lässt sich somit schlussfolgern, dass die Länge des zweiten Fragments von der Länge des ersten Fragments abhängt. Demnach wird die Länge eines Fragments zusätzlich von seiner Position innerhalb des Passworts beeinflusst.

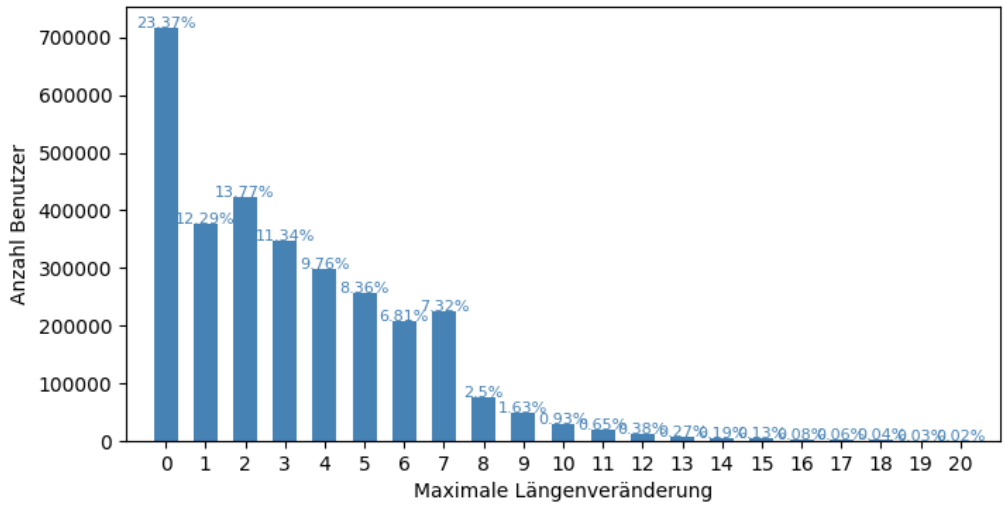
4.3.4 Wiederverwendung

Ziel dieses Abschnitts ist es, Rückschlüsse auf die Wiederverwendung von verwendeten Fragmentlängen pro Benutzer zu ziehen. Hierfür werden die Fragmente pro Fragment-Klasse für jeden Benutzer einzeln betrachtet und die maximale Längendifferenz der Fragmente pro Klasse gemessen.

4.3.4.1 Alphabetische Fragmente

Bei der alphabetischen Fragment-Klasse handelt es sich insgesamt um $3.068.742$ Benutzer, die mindestens zwei alphabetische Fragmente in ihren Passwörtern besitzen. Wie in Abbildung 4.5 zu sehen, variiert die maximale Differenz zwischen zwei Wörtern zweier unterschiedlicher Passwörter eines Benutzers relativ stark.

Abbildung 4.5: Maximale Längendifferenz alphabetischer Fragmente



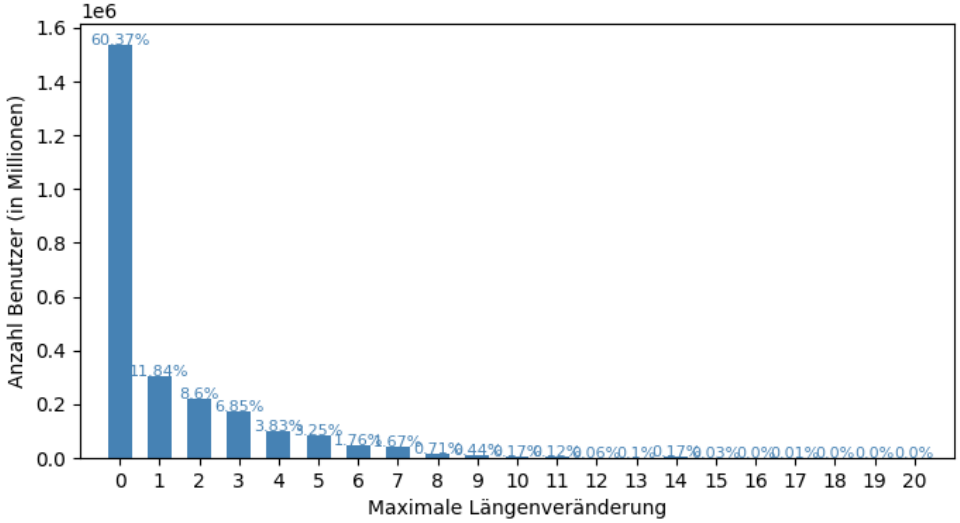
Quelle: eigene Darstellung.

Während $23,37\%$ der Benutzer keine Veränderung in der Länge des alphabetischen Fragments vornehmen, findet bei insgesamt ca. 76% der Benutzer eine Längenveränderung um 1 bis 10 Zeichen statt. Dabei beträgt der Durchschnittswert der maximalen Längenveränderung von alphabetischen Fragmenten über alle Benutzer $3,1596$ Zeichen.

4.3.4.2 Numerische Fragmente

Bei der numerischen Fragment-Klasse handelt es sich insgesamt um $2.548.951$ Benutzer, die mindestens zwei numerische Fragmente in ihren Passwörtern besitzen. Wie in Abbildung 4.6 zu erkennen, liegt bei der Mehrheit der Benutzer mit $60,37\%$ die maximale Längendifferenz bei 0 Zeichen. Sie verwenden demnach eine Zahl gleicher Länge für ein weiteres Passwort.

Abbildung 4.6: Maximale Längendifferenz numerischer Fragmente



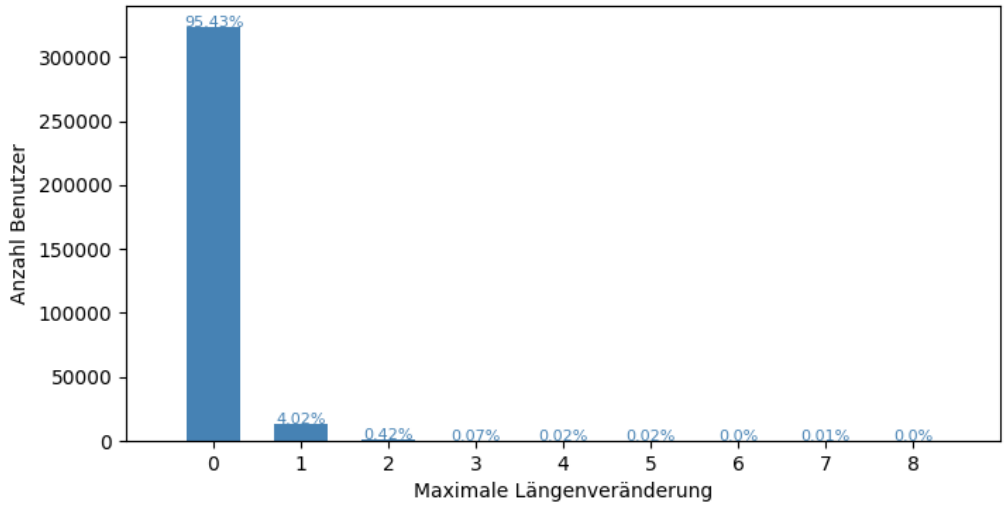
Quelle: eigene Darstellung.

Eine Längenveränderung um 1 bis 10 Zeichen findet nur bei rund 39% der Benutzer statt und liegt damit um rund 36 Prozentpunkte unterhalb der Längenveränderung bei den alphabetischen Zeichenketten. Der Durchschnittswert der maximalen Längenveränderung von numerischen Fragmenten liegt bei $1,214$ Zeichen.

4.3.4.3 Sonderzeichen-Fragmente

Bei der Fragment-Klasse der Sonderzeichen handelt es sich insgesamt um 339.534 Benutzer, die mindestens zwei Sonderzeichen in ihren Passwörtern besitzen. Wie in Abbildung 4.7 zu erkennen, liegt die maximale Längenveränderung bei $95,43\%$ der Benutzer bei 0 Zeichen.

Abbildung 4.7: Maximale Längendifferenz der Sonderzeichen-Fragmente



Quelle: eigene Darstellung.

Lediglich $4,02\%$ der Benutzer erweitern das Fragment um ein weiteres Sonderzeichen. Der Durchschnittswert der maximale Längenveränderung von Sonderzeichen liegt bei $0,0544$ Zeichen. Demnach bleibt die Länge der Sonderzeichen-Fragmente bei fast allen Benutzern über die Passwörter hinweg gleich.

Dies verdeutlicht, dass Benutzer bei der Generierung eines neuen Passworts Elemente gleicher bzw. nur geringfügig abweichender Länge pro Fragment-Klasse wählen. Bei den Wörtern tritt im Vergleich zu den Zahlen und Sonderzeichen die größte maximale Längenveränderung von durchschnittlich rund 3 Zeichen auf. Bei den Zahlen und Sonderzeichen liegt die maximale Abweichung im Durchschnitt bei rund 1 bzw. 0 Zeichen. Inhaltlich kann es sich bei einer Abweichung von 0 Zeichen um genau das gleiche Wort, die gleiche Zahl oder das gleiche Sonderzeichen handeln, was auf den Aspekt der Wiederverwendung (siehe Kapitel 4.5) zurückzuführen ist oder um ein noch unbekanntes Fragment.

4.4 Strukturelle Analyse

Mit dem Ziel detailliertere Informationen bezüglich der Struktur und des Aufbaus menschlich generierter Passwörtern zu erhalten, wird im Folgenden eine Analyse der Passwort-Muster durchgeführt. Entsprechend dem in Abschnitt 2.7 beschriebenen Vorgehen werden die Muster und Muster-Klassen der Passwörter des Leaks erstellt.

4.4.1 Passwort-Muster

Insgesamt bestehen die Passwörter des Leaks aus 241.709 verschiedenen Passwort-Mustern, sodass $99,05\%$ der Passwörter eine mehrfach auftretende Passwortstruktur besitzen. Die 10 meist verwendeten Muster sind in Tabelle 4.11 abgebildet.

Tabelle 4.11: Passwort-Muster höchster Auftrittshäufigkeit

Rang	Muster	Häufigkeit	Anteil (in %)
1	L_8	4.213.248	$16,9042$
2	L_6	1.226.046	$4,8062$
3	L_7	888.422	$3,4827$
4	L_6D_2	840.591	$3,2952$
5	L_9	673.564	$2,6404$
6	D_6	597.531	$2,3423$
7	L_{10}	589.634	$2,3114$
8	D_8	468.453	$1,8364$
9	L_7	439.818	$1,7241$
10	L_4D_4	412.168	$1,6157$

Quelle: eigene Darstellung.

Auffällig ist, dass alle Strukturen lediglich aus Kleinbuchstaben, Zahlen oder einer Kombination aus beidem bestehen. Die Passwort-Muster der ersten drei Ränge, die bereits rund 26% aller Passwörter besitzen, bestehen sogar nur aus Kleinbuchstaben mit einer Länge von 6 bis 8 Zeichen. Besonders interessant ist, dass weder Großbuchstaben noch Sonderzeichen Bestandteil der beliebtesten Muster sind.

In Abschnitt 4.3.3.1 konnte bereits festgestellt werden, dass die Mehrheit der Benutzer nicht den vollständigen Zeichenraum für eine maximale Passwortkomplexität ausschöpfen. Im Folgenden werden daher die in den Passwörtern verwendeten Zeichensätze auf enthaltene Muster-Elemente überprüft und kategorisiert. Die Ergebnisse der Kategorisierung sind in Tabelle 4.12 dargestellt.
Die zwei häufigsten prozentualen Auftrittshäufigkeiten sind mit $36,36\%$ und $41,96\%$ die Kategorien, bei denen ein Passwort lediglich aus Kleinbuchstaben ($\{L\}$) oder aus einer Kombination aus Kleinbuchstaben und Zahlen ($\{L,D\}$) bestehen. Des Weiteren haben die Kategorie der Zahlen ($\{D\}$) sowie die Kategorie der Großbuchstaben, Kleinbuchstaben und Zahlen ($\{L,U,D\}$) eine im Vergleich mit den restlichen Kategorien hohe Auftrittshäufigkeit mit $6,84\%$ bzw. $5,84\%$.

Tabelle 4.12: Verteilung der Muster-Elemente der Passwörter

Anzahl Muster-Elemente	Muster-Element	Häufigkeit	Anteil (in %)
1	$\{L\}$	9.274.400	$36,3560$
	$\{U\}$	89.100	$3,4928$
	$\{D\}$	1.744.483	$6,8384$
	$\{S\}$	2.535	$0,0099$
2	$\{L,U\}$	726.832	$2,8492$
	$\{L,D\}$	10.703.838	$41,9595$
	$\{L,S\}$	722.625	$2,8327$
	$\{U,D\}$	162.315	$0,6363$
	$\{U,S\}$	10.081	$0,0395$
	$\{D,S\}$	24.727	$0,0969$
3	$\{L,U,D\}$	1.489.069	$5,8372$
	$\{L,U,S\}$	35.048	$0,1374$
	$\{L,D,S\}$	422.675	$1,6569$
	$\{U,D,S\}$	9.143	$0,0358$
4	$\{L,U,D,S\}$	93.053	$0,3648$

Quelle: eigene Darstellung.

Auffällig ist, dass rund 95% der Passwörter keine Sonderzeichen und rund 87% keine Großbuchstaben enthalten. Außerdem verwenden die Benutzer nur in $0,36\%$ der Passwörter alle vier Zeichentypen und nutzen damit den vollständigen Zeichenraum für eine maximale Passwortkomplexität.

4.4.2 Muster-Klassen

Bei der Generierung der Muster-Klassen müssen die folgenden Möglichkeiten einer Fehlklassifizierung berücksichtigt werden:

- Auf die Muster-Klassen-Elemente L^+ und N^+ können beliebig viele L^+-Fragmente folgen ohne vom Algorithmus als eigenes Fragment erkannt zu werden. Das Passwort „Fortnitefornite“ beispielsweise wird demnach vom Algorithmus fälschlicherweise als Muster-Klasse N^+ anstatt als Muster-Klasse N^+L^+ detektiert. Analog verhält es sich bei den Muster-Klassen-Elementen U^+ und R^+ mit folgenden U^+-Fragmenten.

- Das Passwort-Muster U_5L_7 beispielsweise wird vom Algorithmus als Muster-Klasse U^+L^+ erkannt, obwohl es sich auch um die Muster-Klasse U^+N^+ handeln kann wie zum Beispiel bei dem Passwort „GAMEFortnite“. Analog verhält es sich mit den Muster-Klassen L^+U^+ und L^+R^+.

Zudem finden die Muster-Klassen-Elemente N^+ und R^+ erst ab einer Aneinanderreihung von drei Muster-Elementen Anwendung (siehe Abschnitt 2.7). Der Grund dieser

Annahme ist, dass bei alphabetischen Fragmenten, die aus weniger als 3 Zeichen bestehen, oftmals die inhaltliche Bedeutung fehlt und es sich um ein zufällig generiertes Fragment handelt. Das Ziel der Abbildung von Wörtern mit groß- bzw. kleingeschriebenem Anfangsbuchstaben auf Muster-Klassen-Ebene wäre damit verfehlt. Das alphabetische Fragment „Ja“ wird zum Beispiel von dem Algorithmus als Muster-Klasse U^+L^+ detektiert, während das Fragment „Jaa“ als Muster-Klasse N^+ klassifiziert wird.

Insgesamt bestehen die Passwörter aus 26.386 verschiedenen Muster-Klassen, sodass 99.89% der Passwörter einer mehrfach auftretenden Muster-Klasse angehören. In Tabelle 4.13 sind die 10 meist aufgetretenen Muster-Klassen abgebildet.

Tabelle 4.13: Muster-Klassen höchster Auftrittshäufigkeit

Rang	Muster-Klasse	Häufigkeit	Prozentualer Anteil
1	L^+	9.274.400	$36,3560$
2	L^+D^+	7.647.943	$29,9803$
3	D^+	1.744.483	$6,8396$
4	N^+D^+	801.464	$3,1418$
5	$L^+D^+L^+$	771.995	$3,0263$
6	D^+L^+	734.857	$2,8807$
7	N^+	528.485	$2,0717$
8	$L^+D^+L^+D^+L^+$	429.098	$1,6821$
9	$L^+S^+L^+$	370.914	$1,454$
10	$D^+L^+D^+$	245.147	$0,961$

Quelle: eigene Darstellung.

Die Top-3 Muster-Klassen entsprechen den meist verwendeten Passwort-Mustern aus Tabelle 4.11 und beinhalten ausschließlich Kleinbuchstaben und Zahlen. Bei exklusiver Betrachtung dieser Muster-Klassen sind bereits rund 73% der längenunabhängigen Strukturen aller Passwörter des Leaks abgedeckt.
Darüber hinaus fällt auf, dass 4 der Muster-Klassen mit einem kleingeschriebenen Wort bzw. einer Zahl beginnen und auf dieses auch wieder enden (Rang 5, 8, 9 und 10). Bei dem umschlossenen Fragment handelt es sich bevorzugt um eine Zahl oder ein Sonderzeichen.
Bezüglich der Schreibweise der alphabetischen Fragmente, ist neben der vollständigen Kleinschreibung, die Großschreibung des ersten Buchstabens (N^+) Bestandteil der beliebtesten Muster-Klassen (siehe Rang 4 und 7).

Die näherer Betrachtung der Verteilung der alphabetischen Muster-Klassen L^+, U^+, N^+ und R^+ bezüglich ihrer Auftrittshäufigkeit in Tabelle 4.14, verdeutlicht die Vorlieben der Benutzer bezüglich der Groß- und Kleinschreibung nochmals. Mit einem prozentualen Anteil von $88,61\%$ an der Gesamtheit der alphabetischen Fragmente sind kleingeschriebene Wörter bzw. Buchstaben mit großen Abstand die beliebteste Schreibweise. An zweiter Stelle stehen mit $6,47\%$ Wörter mit großem Anfangsbuchstaben, gefolgt

von vollständig großgeschriebenen Fragmenten. Die Großschreibung eines Worts mit kleinem Anfangsbuchstaben (R^+) wird mit einem Anteil $0,4\%$ nur sehr selten in einem Passwort verwendet.

Tabelle 4.14: Häufigkeitsverteilung der alphabetischen Muster-Klassen-Elemente

Muster-Klassen-Element	Häufigkeit	Anteil (in %)
L^+	26.559.421	$88,61$
U^+	1.353.610	$4,52$
N^+	1.939.325	$6,47$
R^+	120.669	$0,4$
Summe	29.973.025	100

Quelle: eigene Darstellung.

4.5 Analyse der Wiederverwendung

Das Ziel dieses Abschnitts ist es herauszufinden, wie häufig Benutzer Passwörter bzw. Passwort-Elemente wiederverwenden. Hierfür wird der Grad der Korrelation zwischen den einzelnen Passwörtern eines Benutzers abgeschätzt. Des Weiteren werden die vorgenommenen Modifikationen zwischen den Passwörtern eines Benutzers herausgearbeitet. Auf diese Weise sollen Muster und Vorgehensweisen bei der exakten sowie der partiellen Passwortwiederverwendung benutzerübergreifend identifiziert werden.
Die Datengrundlage bildet die Passwortsammlung 4. Hierdurch ist sichergestellt, dass von jedem Benutzer mindestens zwei Passwörter vorliegen. In der folgenden Analyse werden die Passwörter pro Benutzer betrachtet.

4.5.1 Passwortmenge

Insgesamt liegen $8.970.971$ Passwörter von $3.098.923$ Benutzern vor. Von $73,72\%$ der Benutzer sind, wie Tabelle 4.15 zu entnehmen, zwei Passwörter gegeben. Insgesamt liegen von $99,35\%$ der Benutzer eine Passwortmenge zwischen 1 und 6 Passwörtern vor. Lediglich von $0,65\%$ der Benutzer sind mehr als 6 Passwörter gegeben.

Tabelle 4.15: Verteilung der Anzahl an Passwörtern

Anzahl an Passwörtern	Anzahl an Benutzer	Anteil (in %)
2	2.284.654	73,7241
3	528.697	17,0606
4	170.006	5,4859
5	65.595	2,1167
6	29.952	0,9665
> 6	20.019	0,6524

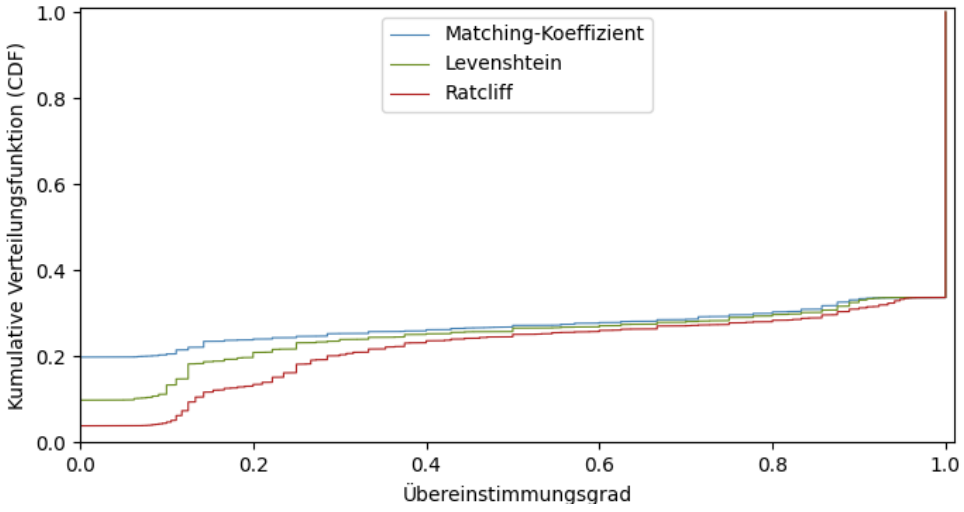
Quelle: eigene Darstellung.

Im Folgenden wird zunächst eine Annäherung zur Abschätzung der Ähnlichkeit der Passwörter eines Benutzers unternommen.

4.5.2 Korrelation der Passwörter

Hierfür werden die in Abschnitt 2.6 vorgestellten Indikatoren zur Messung des Übereinstimmungsgrads zweier Passwörter herangezogen und die Höhe der Korrelation zwischen den Passwörtern eines Benutzers gemessen.
In Abbildung 4.8 sind die kumulativen Verteilungsfunktionen des maximalen Übereinstimmungswerts zweier Passwörter pro Indikator inklusive Passwort-Duplikaten abgebildet.

Abbildung 4.8: Kumulative Verteilung der Übereinstimmungs-Ratio (inklusive Duplikate)[14]



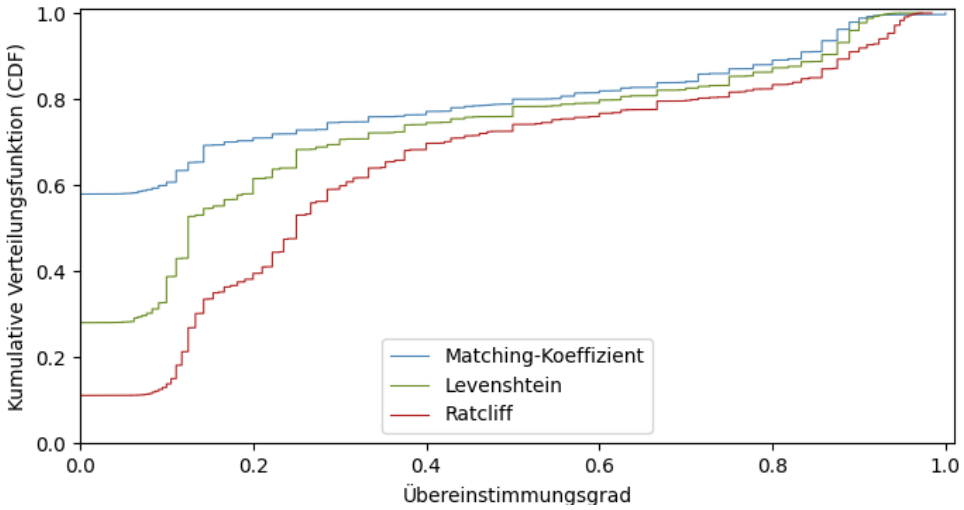
Quelle: eigene Darstellung.

[14] Hierbei handelt es sich um die 3. Passwortsammlung mit 7.568.355 Benutzern.

Aufgrund der vorliegenden, unterschiedlichen Passwortanzahl der Benutzer wird pro Benutzer jeweils der maximale Übereinstimmungsgrad zweier Passwörter herangezogen. Die Interpretation der Daten erfolgt daher auf Benutzer-Ebene. Ein hoher Übereinstimmungsgrad sagt demnach aus, dass mindestens zwei Passwörter eines Benutzers miteinander korrelieren, schließt jedoch nicht aus, dass der Benutzer zusätzlich vollkommen unterschiedliche Passwörter besitzt.
Wie zu erkennen ist, haben die Verteilungsfunktionen aller drei Indikatoren einen sehr ähnlichen Verlauf. Ein Übereinstimmungsgrad zweier Passwörter von $0,0$ besagt, dass der Benutzer ausschließlich vollständig verschiedene Passwörter besitzt. Der Ratcliff-Indikator bemisst diese Anzahl an Benutzern auf rund 4%, gefolgt von der Levenshtein-Distanz mit rund 10%. Den größten Benutzeranteil mit rund 20% bemisst der Matching-Koeffzient.
Auffällig ist der vertikale Verlauf aller drei Funktionen bei einem Übereinstimmungsgrad von $1,0$, welcher eine exakte Übereinstimmung zweier Passwörter besagt. Dies spiegelt eine exakte Wiederverwendung eines Passworts wieder und repräsentiert die im Datenset enthaltenen Duplikate. Demnach verwenden rund 66% der Benutzer eines ihrer Passwörter in identischer Form erneut. Der Bereich zwischen einem Übereinstimmungswert von $0,0$ und $1,0$ spricht für eine partielle Übereinstimmung zwischen mindestens 2 Passwörtern in unterschiedlicher Höhe. Die Funktionen nähern sich in diesem Bereich einander an, verlaufen jedoch sehr flach.

Für eine genauere Messung der Benutzeranzahl für die verschiedenen Stärken der Passwort-Korrelation bei der partiellen Wiederverwendung werden die Passwörter im Folgenden exklusive der Passwort-Duplikate betrachtet. Abbildung 4.9 zeigt die entsprechenden kumulativen Verteilungsfunktionen der drei Indikatoren.

Abbildung 4.9: Kumulative Verteilung der Übereinstimmungs-Ratio (exklusive Duplikate)[15]



Quelle: eigene Darstellung.

Alle drei Indikatoren bemessen die Anzahl an Benutzern, die einen maximalen Übereinstimmungsgrad von $0,0$ besitzen, als deutlich höher im Vergleich zu den Daten mit Duplikaten. Genauer ist die Anzahl an Benutzern hierbei um 7 bis 38 Prozentpunkte gestiegen. Hieraus lässt sich ableiten, dass 7% bis 38% der Benutzer, die auf vorheriger Datengrundlage eines ihrer Passwörter exakt wiederverwendet haben, ansonsten vollkommen voneinander unabhängige Passwörter besitzen.[16]
Im weiteren Verlauf steigen alle drei Funktionen bis zu einem Ähnlichkeitsgrad von $0,4$ relativ stark an. Im Anschluss verlaufen sie bis zu einem Korrelationswert von $0,8$ deutlich flacher und steigen in Folge zwischen $0,8$ und $1,0$ erneut stärker an.
Ab einem Übereinstimmungsgrads von $0,8$ kann von einer starken Korrelation zwischen zwei Passwörtern gesprochen werden. Nach dem Levenshtein-Ratio sind hierbei 80% der Zeichen identisch und an der gleichen Position innerhalb der Passwörter. Der Matching-Koeffizient besagt bei einem Wert von $0,8$, dass 80% der Bigramme der Passwörter identisch sind, unabhängig von ihrer Position im Passwort. Nach dem Ratcliff-Indikator machen die übereinstimmenden Zeichenketten 80% der Passwörter aus. Der Matching-Koeffizient bemisst die Anzahl an Benutzern, die einen maximalen Korrelationswert zwischen $0,8$ und $1,0$ aufweisen auf rund 12%, der Levenshtein-Ratio auf rund 13% und der Ratcliff-Indikator auf rund 17%.

Die Tatsachen, dass rund 66% der Benutzer mindestens eines ihrer Passwörter exakt wiederverwenden und rund 17% der Benutzer einen Übereinstimmungsgrad bei ihren nicht-identischen Passwörtern zwischen $0,8$ und $1,0$ aufweisen, deuten auf einen nichttrivialen Zusammenhang zwischen den einzelnen Passwörtern eines Benutzers hin.
Zur weiteren Analyse dieses Zusammenhangs wird im Folgenden der Inhalt sowie die Struktur der Passwörter pro Benutzer hinsichtlich des Wiederverwendungsaspekts analysiert. Hierfür werden die Passwörter auf Passwort-, Fragment- sowie Muster-Klassen-Ebene betrachtet.

4.5.3 Passwort-Ebene

Für die folgenden Analyse der exakten Wiederverwendung von Passwörtern wird als Einzelfall die Passwortsammlung 3 herangezogen, um die Passwort-Duplikate pro Benutzer identifizieren zu können.

4.5.3.1 Exakte Wiederverwendung

Um eine exakte Wiederverwendung handelt es sich, wenn im Passwortset eines Benutzers zwei exakt identische Passwörter enthalten sind. Da jedes Passwort einem Dienst zugehörig ist, verwendet der Benutzer somit zwei identische Passwörter bei zwei unterschiedlichen Diensten.

[15] Hierbei handelt es sich um die 4. Passwortsammlung mit $3.098.023$ Benutzern.
[16] Handelt sich um $5.025.726$ Benutzer (rund 66% aller Benutzer).

Von den insgesamt $7.568.355$ Benutzern verwenden $5.025.736$ Benutzer mindestens eines ihrer Passwörter exakt wieder. Dies entspricht einem prozentualen Anteil von $66,04\%$.
Bei exklusiver Betrachtung dieser Benutzer liegen insgesamt $26.595.373$ Passwörter vor, wovon es sich bei $19.797.837$ Passwörtern um Duplikate handelt. Somit sind $74,44\%$ der Passwörter exakt wiederverwendete Passwörter. Pro Benutzer beträgt die durchschnittliche exakte Wiederverwendungsrate 76%. Dies bedeutet, dass durchschnittlich 76% der vorliegenden Passwörter eines Benutzers Duplikate sind.

4.5.3.2 Partielle Wiederverwendung

Um eine partielle Wiederverwendung auf Passwort-Ebene handelt es sich, wenn zwei Passwörter ein sogenanntes „Subset“ bilden. Dies bedeutet, dass ein Passwort in vollständiger Form ein Teil eines zweiten Passworts ist. Die Modifikation zwischen zwei partiell identischen Passwörtern besteht folglich in der Erweiterung bzw. Verkürzung eines Passworts um ein oder mehrere Fragmente.
Von den insgesamt $3.098.923$ Benutzern haben 336.193 Benutzer mindestens ein Subset in ihren Passwörtern. Das bedeutet, dass $10,69\%$ der Benutzer bei der Generierung eines neuen Passworts auf ein bestehendes Passwort zurückgreifen und dieses zur erneuten Verwendung erweitern bzw. verkürzen.
Bei exklusiver Betrachtung dieser Benutzer liegen insgesamt $2.017.051$ Passwörter vor, wovon es bei 575.296 Passwörtern, dies entspricht $28,52\%$, um partiell wiederverwendete Passwörter handelt. Pro Benutzer beträgt die durchschnittliche partielle Wiederverwendungsrate $39,06\%$. Dies bedeutet, dass es sich bei $39,06\%$ der Passwörter eines Benutzers im Durchschnitt um ein erweitertes bzw. verkürztes bereits verwendetes Passwort handelt.

Bezüglich der Position der Erweiterung eines Passworts hängen 245.102 Benutzer, also $72,91\%$, ein oder mehrere Fragmente an ein Passwort an. 19.418 Benutzer, also $5,78\%$, stellen dem Passwort Fragmente vor und 66.793 Benutzer, also $19,87\%$, erweitern ihr Passwort beidseitig. Im Folgenden wird der Inhalt der Erweiterungsfragmente getrennt nach ihrer Position näher betrachtet.

Anhängen von Fragmenten

Insgesamt handelt es sich um 510.766 Erweiterungsfragmente, die aus 43.110 inhaltlich verschiedenen Fragmenten bestehen. Wie in Tabelle 4.16 ersichtlich, werden mit $61,23\%$ am häufigsten Zahlen an ein Passwort angehängt. Wörter bzw. Buchstaben folgen mit großem Abstand als zweit beliebtestes Anhang-Fragment.

Tabelle 4.16: Verteilung der Anhang-Fragmente

Fragment-Klasse	Häufigkeit	Anteil (in %)
Alphabetisch	100.966	19,7676
Numerisch	312.745	61,2306
Alphanumerisch	9.979	1,9537
Sonderzeichen	87.076	17,0481

Quelle: eigene Darstellung.

Interessant zu beobachten ist, dass die prozentuale Auftrittshäufigkeit von Sonderzeichen mit $17,05\%$ nur geringfügig kleiner ist wie die der alphabetischen Fragmente mit $19,77\%$. Ein Fragment bestehend aus Ziffern und Buchstaben wird hingegen mit einer prozentualen Häufigkeit von $1,95\%$ sehr selten verwendet.

In Tabelle 4.17 ist ein Ausschnitt der meist verwendeten Anhang-Fragmente gegeben. Die Ziffer 1 ist dabei mit Abstand das am häufigsten verwendete Fragment zur hinteren Erweiterung eines Passworts. Das Ausrufezeichen „!“ folgt trotz seiner Zugehörigkeit zur Kategorie der Sonderzeichen, die wie bereits festgestellt, eher selten integriert werden, auf dem zweiten Rang. Auffällig ist, dass die Mehrheit der Anhang-Fragmente aus einzelnen Buchstaben bestehen.

Tabelle 4.17: Anhang-Fragmente höchster Auftrittshäufigkeit

Rang	Fragment	Häufigkeit	Anteil (in %)
1	1	82.960	16,2423
2	!	43.104	8,4391
3	123	27.039	5,2938
4	a	16.509	3,2322
5	q	15.136	2,9634
6	w	12.542	2,4555
7	qwerty	9.155	1,7924
8	s	9.107	1,7830
9	d	9.082	1,7781
10	e	8.894	1,7413

Quelle: eigene Darstellung.

Voranstellen von Fragmenten

Insgesamt handelt es sich um 131.279 Voranstell-Fragmente, die aus 52.254 inhaltlich verschiedenen Fragmenten bestehen. Im Vergleich zu der Verteilung der Anhang-Fragmente sind die alphabetischen Fragmente mit einem prozentualen Anteil von $68,67\%$ weitaus beliebter bei den Benutzern. Die numerischen Fragmente hingegen sowie auch die Sonderzeichen werden mit einem Differenzwert von rund 35 bzw. 12 Prozentpunkten viel seltener einem Passwort vorangestellt als angehängt.

Tabelle 4.18: Verteilung der Voranstell-Fragmente

Fragment-Klasse	Häufigkeit	Anteil (in %)
Alphabetisch	90.150	68,6705
Numerisch	33.564	25,5669
Alphanumerisch	1.189	0,9057
Sonderzeichen	6.376	4,8568

Quelle: eigene Darstellung.

Bei Betrachtung der beliebtesten Voranstell-Fragmente in Tabelle 4.19 werden einige Übereinstimmungen mit den Anhang-Zeichenketten deutlich. So gehört die Ziffer „1“ sowie die Ziffernfolge „123“ an beiden Positionen zu den beliebtesten Erweiterungs-Fragmenten.

Tabelle 4.19: Voranstell-Fragmente höchster Auftrittshäufigkeit

Rang	Fragment	Häufigkeit	Anteil (in %)
1	1	4.164	3,1719
2	123	1.624	1,2371
3	.	1.476	1,1243
4	buzzmachines.	1.141	0,8691
5	qwerty	1.056	0,8044
6	-	863	0,6574
7	2	853	0,6498
8	12	737	0,5614
9	m	645	0,4913
10	4	608	0,4631

Quelle: eigene Darstellung.

Des Weiteren wird das Tastatur-Muster „qwerty“ in beiden Fällen häufig verwendet. Anstelle des Ausrufezeichens werden Passwörter zu Beginn bevorzugt um Sonderzeichen wie dem Bindestrich „-“ oder dem Punkt „.“ erweitert. Auffällig ist, dass die Zeichenkette „buzzmachines.“ als viert häufigstes Voranstell-Fragment verwendet wird.

4.5.4 Fragment-Ebene

Zur Identifizierung weiterer von den Benutzern vorgenommenen Modifikationen werden die Passwörter im Folgenden auf Fragment-Ebene betrachtet.

4.5.4.1 Exakte Wiederverwendung

Um eine exakte Wiederverwendung auf Fragment-Ebene handelt es sich, wenn ein Benutzer ein Fragment eines Passworts in identischer Form in einem zweiten Passwort erneut verwendet. Hierbei sind auch Fragmente miteingeschlossen, die nur mit einem Teil eines anderen Fragments übereinstimmen. Auf diese Weise wird beispielsweise das Passwort-Fragment „sabi“ als wiederverwendetes Fragment detektiert, wenn der Benutzer in einem weiteren Passwort den Namen „sabine“ verwendet.
Zudem wird hiermit die Auswirkung einer fehlerhaften Fragmentierung als Folge der bereits erläuterten Fehlklassifizierung (siehe Abschnitt 4.4.2) berücksichtigt. Sollten daher zwei Fragmente der gleichen Muster-Klasse aufeinander folgen, die vom Algorithmus fälschlicherweise als ein Fragment detektiert werden, wie beispielsweise bei den Passwörtern „love“ und „loveyou“, wird das Fragment „love“ trotzdem als ein exakt wiederverwendetes Fragment erkannt. Fragmente mit einer Länge von eins werden grundsätzlich nicht berücksichtigt, da in ihrem Fall nicht angenommen werden kann, dass es sich um eine bewusste Wiederverwendung durch den Benutzer handelt.

Insgesamt verwenden 805.015 Benutzer mindestens ein Fragment aus einem ihrer Passwörter exakt wieder, was einem prozentualen Anteil von $25,97\%$ entspricht. Unter Ausschluss der bereits identifizierten Subsets aus Abschnitt 4.5.3.2 verbleiben 468.822 Benutzer $(15,13\%)$, die ausschließlich ein Fragment und nicht das gesamte Passwort wiederverwenden.
Bei exklusiver Betrachtung dieser Benutzer liegen insgesamt $3.596.391$ Passwörter vor, von denen $2.635.836$ Passwörtern ein mehrfach verwendetes Fragment enthalten. Dies entspricht einem prozentualen Anteil von $73,29\%$. Pro Benutzer beträgt die durchschnittliche exakte Wiederverwendungsrate $83,02\%$. Dies bedeutet, dass durchschnittlich $83,02\%$ der Passwörter eines Benutzers ein bereits verwendetes Fragment enthalten.

Bei inhaltlicher Betrachtung der wiederverwendeten Fragmente verwenden $74,16\%$ der 805.015 Benutzer ein alphabetisches und $45,13\%$ ein numerisches Fragment erneut (siehe Tabelle 4.20). Nur rund $0,2\%$ der Benutzer greifen auf ein bereits verwendetes Sonderzeichen zurück.[17]

Tabelle 4.20: Benutzerhäufigkeit exakter Fragment-Wiederverwerndung

Fragment-Klasse	Häufigkeit	Anteil (in %)
Alphabetisch	597.031	74,164
Numerisch	363.295	45,129
Sonderzeichen	1.425	0,1770

Quelle: eigene Darstellung.

[17] Die Interpretation der Daten erfolgt inklusive, das heißt ein Benutzer kann Bestandteil mehrerer Fragment-Klassen sein.

4.5.4.2 Partielle Wiederverwendung

Um eine partielle Wiederverwendung auf Fragment-Ebene handelt es sich, wenn ein Fragment in modifizierter Form Teil eines zweiten Passworts ist. Insgesamt verwenden 752.388 Benutzer ein Fragment eines Passworts partiell wieder. Dies entspricht einem prozentualen Anteil von $24,78\%$ der Benutzer.

Die identifizierten Modifikationen sind mit entsprechender Anwendungshäufigkeit in Tabelle 4.21 abgebildet. Die letzte Spalte gibt die prozentuale Anzahl von Benutzern der vorgenommenen Modifikation mit entsprechender Untergliederung der Modifikationsarten an.

Tabelle 4.21: Identifizierte Modifikationen der partiellen Wiederverwendung

Modifikation	Art	Häufigkeit	Anteil an Modifikation (in %)
1. Groß- und Kleinschreibung	Großschreibung (U^+)	68.915	26,5906
	Anfangsbuchstabe groß (N^+)	186.187	71,8394
	Anfangsbuchstabe klein (R^+)	4.069	1,5700
	$\sum$	259.171	100
	Anteil (in %)	8,3633	
2. Fragment-rotation	-	1.572	100
	Anteil (in %)	0,0507	
3. Zeichen-rotation	A^+-Rotation	525	0,4663
	D^+-Rotation	2.954	2,6237
	A^+Spiegelung	71.903	63,6325
	D^+Spiegelung	33.108	29,4061
	Beliebige A^+-Rotation	1.858	1,6502
	Beliebige D^+-Rotation	2.241	1,9904
	$\sum$	112.589	100
	Anteil (in %)	3,6332	
4. Duplizierung	Gesamtes Passwort	22.740	5,9991
	Alphabetisches Fragment	125.956	33,2289
	Erster Buchstabe	49.334	13,015
	Letzter Buchstabe	35.692	9,4160
	Numerisches Fragment	40.599	10,7106
	Erste Zahl	43.220	11,4020
	Letzte Zahl	60.448	15,947
	Sonderzeichen	1.067	0,2815
	$\sum$	379.056	100
	Anteil (in %)	12,2319	
$\sum$	-	752.388	-
	Anteil (in %)	24,78	

Quelle: eigene Darstellung.

Bei der ersten Modifikation handelt es sich um die Veränderung der Groß- und Kleinschreibung eines alphabetischen Fragments[18]. Zur Messung der Anwendungshäufigkeit werden alle kleingeschriebenen, alphabetischen Fragmente eines Benutzers extrahiert und ihre Existenz im Passwortset des Benutzers in den Schreibvarianten U^+, N^+ und R^+ überprüft.
Wie in Tabelle 4.21 zu erkennen ist, verwenden insgesamt $8,36\%$ aller Benutzer ein kleingeschriebenes Wort erneut in veränderter Schreibweise. Mit $71,84\%$ konvertieren die Mehrheit dieser Benutzergruppe den Anfangsbuchstaben zur Großschreibung N^+. Mit $26,59\%$ der Benutzer folgt die Veränderung zu vollständiger Großschreibung U^+. Lediglich $1,57\%$ der Benutzer verwenden die Schreibweise der Großschreibung mit kleinem Anfangsbuchstaben R^+.

Die zweite, identifizierte Operation besteht aus der Fragmentrotation. Hierbei beinhalten die Passwörter vor und nach der Modifikationsdurchführung dieselben Fragmente in gleicher Häufigkeit, jedoch in unterschiedlichen Reihenfolgen. Ein mehrfach auftretendes Rotationsmuster ist die Rotation um eine Fragmentposition nach links oder nach rechts. Eine Unterscheidung zwischen den Rotationsrichtungen ist aufgrund der fehlenden Information über den Zeitpunkt der jeweiligen Passwortgenerierung nicht möglich.[19] Im Ergebnis führen $0,05\%$ der Benutzer eine Rotationstransformation zwischen zwei Passwörtern durch.

Bei der dritten Modifikation handelt es sich um die Zeichenrotation innerhalb eines Passwort-Fragments. Hierbei beinhalten die Passwörter vor und nach der Durchführung dieselben Zeichen, jedoch in unterschiedlichen Reihenfolgen. Die folgenden Rotationsmuster werden benutzerübergreifend angewendet:[20]

- Rotation eines alphabetischen bzw. numerischen Fragments um ein Zeichen nach links oder rechts.
- Spiegelung eines alphabetischen bzw. numerischen Fragments.
- Beliebige Zeichenrotation innerhalb eines alphabetischen bzw. numerischen Fragments.

Wie in Tabelle 4.21 zu sehen ist, nehmen insgesamt 112.589 Benutzer $(3,63\%)$ eine der definierten Zeichenrotationen vor. Die Spiegelung eines Wortes bzw. einer Zahl sind dabei mit einer prozentualen Anwendungshäufigkeit von $63,63\%$ bzw. $29,41\%$ die mit Abstand meist verwendeten Rotationsmuster auf Fragment-Ebene.

[18] Hierbei werden ausschließlich Fragmente mit einer Länge größer als 1 berücksichtigt.
[19] Um festzustellen, ob eine Links- oder Rechtsrotation vorgenommen wurde, müsste die Information vorliegen, welches Passwort vom Benutzer zuerst erstellt wurde.
[20] Hierbei finden nur die Fragmente Berücksichtigung, die eine Länge größer als 1 aufweisen und aus mindestens zwei unterschiedlichen Zeichen bestehen.

Die vierte Modifikation ist das Prinzip der Duplizierung, welches aus den folgenden verschiedenen Duplizierungsarten besteht:

- Gesamtes Passwort
- Alphabetisches bzw. numerisches Fragment
- Erster bzw. letzter Buchstaben
- Erste bzw. letzte Zahl
- Sonderzeichen-Fragment

Hierbei gilt es zu beachten, dass zwischen zwei Passwörtern mehrere der definierten Duplikationsverfahren Anwendung finden können. Beispielsweise handelt es sich bei den Passwörtern „Fortnite“ und „FortniteFornite“ um eine Duplizierung des alphabetische Fragments sowie des gesamten Passworts. Die resultierenden Messwerte stellen daher Maximalwerte dar.
Wie in Tabelle 4.21 ersichtlich, nehmen insgesamt 379.056 Benutzer $(12,23\%)$ einer der definierten Duplizierungsmodifikationen vor. Mit einem Benutzeranteil von $33,23\%$ ist das Verdoppeln eines Wortes die beliebteste Modifikation. Weiterhin ist bei den alphabetischen Fragmenten das Duplizieren des ersten Buchstabens mit $13,02\%$ beliebter als das Duplizieren des letzten Buchstabens mit $9,42\%$. Gegenteilig verhält es sich bei den numerischen Fragmenten, bei denen mit einem Benutzeranteil von $15,95\%$ bevorzugt die letzte Zahl dupliziert wird.
Besonders interessant ist, dass alle Duplizierungsoperationen von einzelnen Passwort-Elementen (ausgenommen der Sonderzeichen) häufiger angewendet werden als das Duplizieren des gesamten Passworts mit einem Anteil von rund 6% der Benutzer.

4.5.5 Muster-Klassen-Ebene

Im bisherigen Verlauf der Analyse der Passwörter hinsichtlich des Aspekts der Wiederverwendung stand der Inhalt der Passwörter sowie dessen Modifikation im Vordergrund. Im Folgenden wird der Fokus auf die Konstruktion der Passwörter pro Benutzer gelegt. Das Ziel ist die Klärung, in welchem Umfang Benutzer an bereits verwendeten Passwortstrukturen festhalten und diese bei der Generierung eines neuen Passworts wiederverwenden.

4.5.5.1 Exakte Wiederverwendung

Um eine exakte Wiederverwendung auf Muster-Klassen-Ebene handelt es sich, wenn im Passwortset eines Benutzers zwei Passwörter derselben Muster-Klasse angehören und somit eine identische Struktur aufweisen.
Im Ergebnis verwenden $1.212.085$ Benutzer mindestens eine Muster-Klasse für mindestens ein weiteres Mal, ohne das es sich bei dem zugehörigen Passwort um ein Duplikat

handelt. Somit halten $39,11\%$ der Benutzer bei der Generierung eines neuen Passworts an bestehenden Passwortstrukturen fest.
Bei exklusiver Betrachtung dieser Benutzer liegen insgesamt $4.876.922$ Passwörter vor, wovon es sich bei $2.472.516$ Passwörter um Muster-Klassen-Duplikate handelt. $50,7\%$ der Passwörter gehören somit einer wiederverwendeten Muster-Klasse an. Pro Benutzer beträgt die durchschnittliche exakte Wiederverwendungsrate hinsichtlich der Muster-Klassen $45,47\%$. Dies bedeutet, dass durchschnittlich $45,47\%$ der Passwörter eines Benutzers eine bereits verwendete Passwortstruktur besitzen.

4.5.5.2 Partielle Wiederverwendung

Um eine partielle Wiederverwendung auf Muster-Klassen-Ebene handelt es sich, wenn die Muster-Klasse eines verwendeten Passworts in modifizierter Form der Struktur eines zweiten Passworts des Benutzers entspricht.
Hängt ein Benutzer beispielsweise ein „! “ an ein Passwort an, was einer partiellen Wiederverwendung auf Passwort-Ebene entspricht, besitzt die Muster-Klasse des neuen Passworts ein zusätzliches S^{+}-Muster-Klassen-Element am Ende. Dies entspricht einer partiellen Wiederverwendung einer Muster-Klasse.
Die Analyse der partiellen Wiederverwendung von Muster-Klassen findet in Kapitel 7 im Rahmen der Evaluierung des Muster-Generators des Präprozessors (siehe Kapitel 5) statt. Das Ziel des Muster-Generators ist die Erzeugung von Strukturen weiterer Passwörter der gleichen Benutzers auf Basis einer vom Benutzer vorliegenden Passwortstruktur. Zur Evaluierung der Effizienz des Generators werden erneut die Passwörter des Leaks herangezogen und auf einen Trainingsdatensatz jedes Benutzers angewendet. Auf dieser Basis wird gemessen, wie viele Muster-Klassen des Testdatensatzes der Generator pro Benutzer erzeugt (siehe Kapitel 7).

4.6 Persönliche Informationen

Das Ziel des folgenden Abschnitts ist es, die Häufigkeit des Auftretens von direktabbildbaren persönlichen Daten (siehe Abschnitt 2.8) in Passwörtern zu messen. Hierbei werden die folgenden 5 unterschiedlichen Arten von persönlichen Informationen berücksichtigt:

- Benutzername
- Name
- Geburtsdatum
- Email-Adresse
- Telefonnummer

Aufgrund der fehlenden persönlichen Information über die Benutzer des Passwort-Leaks ist es nicht möglich die Passwörter explizit auf benutzerspezifische Daten zu untersuchen. Die Analyse findet daher ohne Personenzuordnung statt. So kann es sich bei identifizierten, persönlichen Informationen um Daten des Benutzers selbst oder um Informationen von zweiten Personen handeln.

Im Ergebnis zeigt sich, dass $12.697.578$ der insgesamt $25.509.924$ Passwörter, also $49,78\%$, eine der definierten Arten von persönlichen Informationen beinhalten. Auf Benutzer-Ebene sind es $11.188.029$ der insgesamt $19.637.876$ Benutzer, also $56,97\%$, die persönliche Informationen von sich oder zweiten Personen in ihre Passwörter integrieren.[21]
Wie Tabelle 4.22 zu entnehmen ist, sind bevorzugt Vor-/ Spitznamen sowie Nachnamen Bestandteile eines Passworts. Am dritt häufigsten werden mit einem Benutzeranteil von $10,95\%$ Geburtsdaten in ein Passwort integriert, gefolgt von der Integration des Benutzernamens mit $8,17\%$ der Benutzer. Eine Email-Adresse sowie eine Telefonnummer finden sich nur in Ausnahmefällen in einem Passwort wieder.

Tabelle 4.22: Auftrittshäufigkeit persönlicher Informationen

Persönliche Information	Passwort-Häufigkeit	Anteil (in %)	Benutzer-Häufigkeit	Anteil (in %)
Benutzernamen	1.859.728	7,2902	1.603.590	8,1658
Vor-/ Spitznamen	4.665.237	18,2879	4.088.684	20,8204
Nachname	3.774.318	14,7955	3.327.284	16,9432
Geburtsdatum	2.378.654	9,3244	2.149.733	10,9469
Email-Adresse	11.679	0,0458	11.137	0,0567
Telefonnummer	7.962	0,0312	7.601	0,0387
$\sum$	12.697.578	49,7751	11.188.029	56.9717

Quelle: eigene Darstellung.

Im Folgenden werden die verschiedenen Arten von persönlichen Informationen näher beschrieben sowie das jeweilige Vorgehen bei der Messung erläutert.

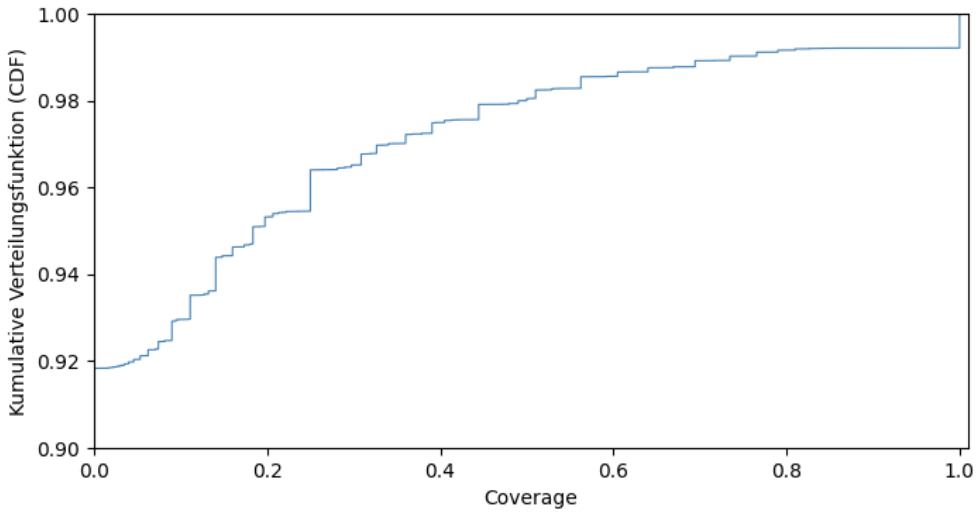
4.6.1 Benutzernamen

Der Benutzername als nicht-domainspezifischer Teil der Email-Adresse stellt die einzige, vorliegende persönliche Information der Benutzer dar. Die Passwörter werden daher pro Benutzer explizit auf die Existenz seines Benutzernamens hin untersucht.
Um die Anzahl an Benutzern zu messen sowie um die Höhe der Korrelation zwischen dem Passwort und dem Benutzernamen zu bestimmen, wird im Folgenden der in Kapitel 3.2.1 vorgestellte Coverage-Indikator herangezogen. Der Coverage-Wert wird für jedes Passwort eines Benutzers berechnet und jeweils der Maximalwert herangezogen.[22]

[21] Hierbei handelt es sich um Maximalwerte. Sollte ein Benutzer mehrere Arten persönlicher Information in seinen Passwörtern verarbeiten, wird es mehrfach berücksichtigt.
[22] Die Interpretation ist erneut nur auf Benutzer-Ebene möglich.

Die Abbildung 4.10 zeigt den Verlauf der kumulativen Verteilungsfunktion des maximalen Coverage-Werts der Benutzer. Wie zu erkennen ist, besitzen knapp 92% der Benutzer eine Coverage-Wert von 0 und integrieren damit keinen Benutzernamen in ihre Passwörter.
Im weiteren Verlauf steigt der CDF im Wertebereich von $0,1$ bis $0,3$ relativ steil an und nähert sich bis zu einem Coverage-Wert von $0,8$ einer Horizontalen an. Ein Korrelationswert in diesem Bereich bedeutet, dass das Passwort den Benutzernamen bzw. Elemente des Benutzernamens beinhaltet, jedoch zusätzlich noch weitere, vom Benutzernamen unabhängige Fragmente besitzt. In diesem Bereich muss die Möglichkeit einer Fehldetektion beachtet werden. Bei Betrachtung des Benutzernamens „-bernhard“ und des Passworts „berlin2“ beispielsweise ergibt sich aufgrund des übereinstimmenden Segments „ber“ ein Coverage von $0,18$, obwohl es sich in dem Passwort nicht um den Benutzernamen handelt.
Bei einem Coverage-Wert von $1,0$, was einer exakten Übereinstimmung von Passwort und Benutzernamen entspricht, steigt die Kurve vertikal um 1 Prozentpunkt an. Bei diesem Wert besteht das vollständige Passwort des Benutzers aus seinem Benutzernamen.

Abbildung 4.10: Kumulative Verteilung des maximalen Coverage-Indikators



Quelle: eigene Darstellung.

Auf Benutzer-Ebene besitzen somit $1.603.590$ Benutzer $(8,17\%)$ einen Coverage-Wert zwischen $0,0$ (exklusiv) und $1,0$ und integrieren damit ihren Benutzernamen bzw. einen Teil ihres Benutzernamens in mindestens eines ihrer Passwörter. Bei rund 196.379 Benutzern aus dieser Benutzergruppe (1% aller Benutzer) besteht das vollständige Passwort aus ihrem Benutzernamen.
Bei Betrachtung der Ergebnisse auf Passwort-Ebene lässt sich festhalten, dass $1.859.728$ Passwörter und damit $7,29\%$ aller Passwörter einen Benutzernamen bzw. einen Teil eines Benutzernamens enthalten.

4.6.2 Namen

Im Folgenden werden die Passwörter auf das Vorhandensein von Vornamen, Spitznamen und Nachnamen untersucht. Hierfür werden zwei Namenslisten mit den meist verbreiteten Vor- bzw. Spitznamen sowie Nachnamen herangezogen und dessen Auftrittshäufigkeit in den Passwörtern des Leaks gemessen.

Bei der durchgeführten Analyse findet keine Berücksichtigung der Groß- und Kleinschreibung statt. Die herangezogenen Namenslisten sowie die Passwörter werden daher vor der Verarbeitung in Kleinbuchstaben konvertiert. Zudem besteht die Möglichkeit einer Fehldetektion, da einige Namen eine zusätzlich namensunabhängige Bedeutung besitzen. Beispielsweise kann mit dem Passwort-Fragment „Juli“ neben dem Namen auch der Monat gemeint sein.
Die herangezogene Vornamen-Liste [12] enthält 132.455 Einträge weiblicher sowie männlicher Vornamen internationaler Herkunft. Im Ergebnis enthalten $4.665.237$ Passwörter, dies entspricht $18,29\%$, einen Vornamen. Auf Benutzer-Ebene integrieren $4.088.684$ Benutzer $(20,82\%)$ einen Vornamen in mindestens eines ihrer Passwörter.
Die herangezogene Nachnamen-Liste [12] enthält 97.002 Einträgen internationaler Familiennamen. Im Ergebnis enthalten $3.774.318$ Passwörter, dies entspricht $14,8\%$, einen Nachnachmen. Auf Benutzer-Ebene integrieren $3.327.284$ Benutzer $(16,94\%)$ einen Nachnamen in mindestens eines ihrer Passwörter.

4.6.3 Geburtsdatum

Eine weitere Art einer direkt-abbildbaren persönlichen Information ist das Geburtsdatum. Grundsätzlich gibt es eine Vielzahl an Möglichkeiten zur numerischen sowie alphabetischen Abbildung eines Geburtsdatums. Bei der numerischen Darstellung unterscheiden sich die verschiedenen Varianten in der Reihenfolge der Anordnung von Tag (im Folgenden „T“), Monat (im Folgenden „M“) und Jahr (im Folgenden „J“) sowie in ihrer Länge.

Zur Detektion eines numerischen Passwort-Fragments als Geburtsdatum muss es zum einen eine Gesamtlänge von 4, 6 oder 8 Zeichen aufweisen sowie eines der folgenden Formate besitzen:[23]

- Länge 4: $JJJJ$, $TTMM$, $MMTT$, $MMJJ$, $JJMM$.
- Länge 6: $TTMMJJ$, $MMTTJJ$, $JJMMTT$, $MMJJJJ$, $JJJJM$, $TTJJJJ$, $JJJJTT$.
- Länge 8: $TTMMJJJJ$, $MMTTJJJJ$, $JJJJMMTT$.

wobei $T \in [0,31]$, $M \in [0,12]$ und $J \in [1900,2099]$.

[23] Diese Annahme wurde mit dem Ziel der Minimierung der Falsch-Positiv-Rate getroffen.

Hierbei ist zu berücksichtigen, dass trotz der getroffenen Annahmen die Möglichkeit einer Fehldetektion besteht, wodurch Falsch-Positive im Ergebnis-Datensatz enthalten sein können. Bei Betrachtung des numerischen Fragments „123123“ beispielsweise, welches von Benutzern häufig als ein dupliziertes Tastatur-Muster verwendet wird, erkennt der Algorithmus fälschlicherweise das Datum des 31. Dezembers 1923 [28]. Zur Minimierung der Falsch-Positiv-Rate werden die 15 meist verwendeten numerischen Fragmente, die vom Algorithmus fehlerhaft detektiert werden würden, extrahiert und explizit ausgeschlossen. Neben der Ziffernfolge „123123“ handelt es sich dabei beispielsweise um die Fragmente „1234“, „0815“ und „1111“.

Im Ergebnis enthalten $2.378.654$ Passwörter $(9,32\%)$ ein Geburtsdatum. Es handelt sich hierbei um $2.149.733$ Benutzer bzw. $10,95\%$, die ein Geburtsdatum in mindestens eines ihrer Passwörter integrieren.
Von den numerischen Fragmenten, die ein Geburtsdatum abbilden, besitzen $64,58\%$ ein Format der Länge 4. Darüber hinaus $21,4\%$ ein Format der Länge 6 und $13,77\%$ ein Format der Länge 8. Die Darstellung eines Geburtsdatums in vierstelliger Variante ist folglich mit Abstand am beliebtesten bei den Benutzern. Dies unterstreicht nochmals die Ergebnisse der 10 meistgenutzten Längen bei numerischen Fragmenten in Abschnitt 4.3.2.2.

4.6.4 Email-Adresse

In Abschnitt 4.6.1 wurden die Passwörter bereits auf den Enthalt des Benutzernamens, dem nicht-domainspezifischen Teil der Email-Adresse untersucht. An dieser Stelle wird daher herausgearbeitet, wie viele Passwörter den domainspezifischen Teil einer Email-Adresse beinhalten. Zur Detektion eines Passworts als eine Email-Adresse muss es das folgende Format aufweisen:

@<Name des Email-Dienstes>.<Länderkürzel>

Im Ergebnis enthalten 11.679 Passwörter (0.046%) eine Email-Adresse. Es handelt sich um 11.137 Benutzer $(0,057\%)$, die den domainspezifschen Teil einer Email-Adresse in mindestens eines ihrer Passwörter integrieren.

4.6.5 Telefonnummer

Die Analyse der Passwörter hinsichtlich des Auftretens einer Telefonnummer ist aufgrund des vielfältigen Aufbaus von mobilen- sowie Festnetz-Rufnummern ohne Kenntnis der jeweiligen Rufnummer des Benutzers nur sehr begrenzt möglich.
Die Detektion eines Passwort-Fragments als Rufnummer erfolgt somit ausschließlich

bei deutschen Mobilfunknummern. Hierzu muss das Fragment aus Ziffern bestehen, eine Gesamtlänge von 11 oder 12 Zeichen aufweisen sowie mit einer der Ziffernfolgen „015“, „016“ oder „017“ beginnen.
Im Ergebnis enthalten 7.962 Passwörter $(0,03\%)$ eine deutsche Mobilfunknummer. Es handelt sich hierbei um 7.601 Benutzer, dies entspricht 0.03%, die eine Rufnummer in mindestens einem ihrer Passwörter integrieren.

5 Benutzerspezifisches Modell

Ziel des Modells ist die Rekonstruktion des Passworts eines spezifischen Benutzers über einen mehrstufigen Angriffsprozess. Der Fokus liegt hierbei auf dem benutzerspezifischen Angriff und dessen Präprozessor. Er generiert die wahrscheinlichsten Passwort-Kandidaten für einen spezifischen Benutzer und dient somit der Vorbereitung des Angriffs. Die Strategie des Präprozessors basiert auf den gewonnenen Erkenntnissen über menschliche Design- und Modifikationsprinzipien der in Kapitel 4 durchgeführten Untersuchung des Passwort-Leaks. Die Implementierung des Präprozessors erfolgt in Python und umfasst 3 Skripte. Nähere Informationen sind der README-Datei in Anhang E zu entnehmen.

5.1 Anwendungsszenario

Die Datengrundlage des Angriffszenarios besteht aus einem vorliegenden Verschlüsselungsgegenstand wie beispielsweise einem Passwort gesichertem Benutzerkonto oder einer verschlüsselten Datei. Zudem liegt mindestens ein Vergleichspasswort des zugehörigen Benutzers sowie seine persönlichen Informationen vor.
Die Durchführung des Angriffs findet im Rahmen einer Offline-Attacke statt. Die Rekonstruktion ist somit nicht zeitkritisch und es gibt keine Beschränkung in der Anzahl von zu testenden Passwort-Kandidaten. Um einen optimalen Kompromiss zwischen dem für die Rekonstruktion notwendigen Ressourcenaufwand in Bezug auf Zeit, Hard- und Software einerseits und einer effizienten Rekonstruktion andererseits zu schaffen, erfolgt der Angriff mit dem Ziel der Minimierung des zu testenden Passwort-Kandidatenraums. Die Aufgabe des Modells ist die Rekonstruktion eines spezifischen Passworts unter minimalem Ressourceneinsatz. Im Rahmen des benutzerspezifischen Angriffs erfolgt die Umsetzung über die Generierung der für den Benutzer wahrscheinlichsten Passwort-Kandidaten unter Einbezug der gegebenen Passwort- sowie Personeninformation.

5.2 Herausforderungen

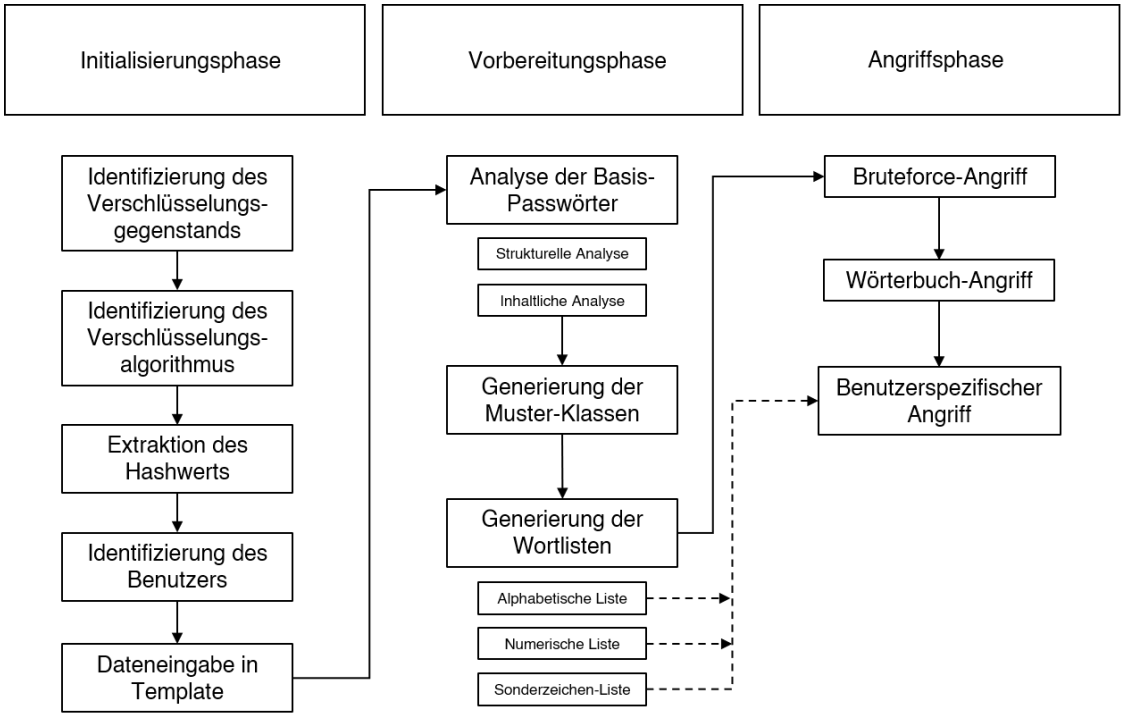
Eine der größten Herausforderungen des Modells resultiert aus der großen Bandbreite an Gestaltungsmöglichkeiten eines Passworts. Wie bereits in Abschnitt 2.4.1 erläutert, bestimmt sich der Schwierigkeitsgrad einer Rekonstruktion durch die Größe des verwendeten Zeichensatzes (inhaltliche Komponente) sowie durch die Zusammensetzung der einzelnen Passwort-Elemente aus unterschiedlichen Zeichentypen (strukturelle Komponente). Demnach muss für eine erfolgreiche Rekonstruktion zum einen die Struktur des gesuchten Passworts bekannt sein sowie die gesuchten Passwort-Elemente vorliegen.

Beide Gestaltungskomponenten sind stark von den individuellen Vorlieben des Benutzers abhängig. Während sich einige Benutzer Passwörter bestehend aus Buchstaben besser merken können, verwenden andere ausschließlich Zahlen in ihren Passwörtern. Einige Benutzer verwenden ausschließlich zufällig generierte Passwörter, während andere ihren bisherigen Passwort-Mustern folgen oder bevorzugt persönliche Informationen integrieren. Diese zwei Beispiele verdeutlichen die großen Unterschiede in den Vorlieben der Benutzer, wodurch sie schwer vorhersehbar sind.
In Bezug auf die inhaltliche Komponente stellt die Zuordnung einer Erfolgswahrscheinlichkeit zu alphabetischen Fragmenten eine weitere Schwierigkeit dar. Wie bereits aufgezeigt, besitzt ein Passwort-Element der Länge x allein 52^x mögliche alphabetische Zeichenketten. Eine Priorisierung dieser ist somit aufgrund der große Vielfalt der Fragment-Klasse eine Herausforderung.
Eine weitere Herausforderung besteht in der Abbildung von persönlichen Informationen in einem Passwort. Diese lassen sich aufgrund ihrer Diversität oft in einer Vielzahl unterschiedlicher Varianten darstellen. Ein Geburtsdatum beispielsweise kann in alphabetischen wie auch in numerischen Zeichen dargestellt werden, wobei eine numerische Abbildung wiederum unzählige Möglichkeiten eröffnet.
Zudem sind nicht alle persönlichen Informationen direkt in einem Passwort abbildbar. Daten wie beispielsweise das Geschlecht, die Bildung oder die Sprache eines Benutzers haben einen entscheidenden Einfluss auf das Design eines Passworts, sind jedoch nicht direkt-abbildbar [38]. Der Umfang der Einbeziehung von persönlichen Informationen sowie deren Abbildung ist somit nur begrenzt möglich.
Zur Erreichung des Ziels eines minimalen Ressourcenaufwands ist das Testen der generierten Passwort-Kandidaten in absteigender Reihenfolge ihrer Erfolgswahrscheinlichkeit maßgebend. Hierfür ist eine Sortierung der Passwort-Kandidaten notwendig, deren Umsetzung im Rahmen des Passwort-Angriffs eine weitere Herausforderung darstellt [6].

5.3 Aufbau des Modells

Der Aufbau des Modells ist in Abbildung 5.1 dargestellt und gliedert sich grundsätzlich in die drei Phasen: Initialisierung, Vorbereitung und Angriff. Bei jeder benutzerspezifischen Passwort-Rekonstruktion des beschriebenen Szenarios werden die einzelnen Phasen in der dargestellten Reihenfolge durchlaufen.

Abbildung 5.1: Aufbau des benutzerspezifischen Modells



Quelle: eigene Darstellung.

Im Folgenden werden die drei Phasen einzeln betrachtet und die jeweiligen Prozessschritte näher beschrieben. Zur Veranschaulichung der Strategie des Modells wird an einigen Stellen die erste Evaluations-Aufgabe aus Kapitel 7 (Experiment 1) herangezogen und die Vorgehensweise des Algorithmus beispielhaft aufgezeigt.

5.4 Initialisierungsphase

Das Ziel der Initialisierungsphase ist die Bereitstellung eines Überblicks über das vorliegende Verschlüsselungsszenario sowie die Aufbereitung der gegebenen Datengrundlage. Hierfür wird zunächst, wie in Abbildung 5.1 dargestellt, der Verschlüsselungsgegenstand sowie der zugrundeliegende Verschlüsselungsalgorithmus identifiziert. Zur Durchführung einer Offline-Attacke muss im Anschluss der vom Dienst bzw. von der Anwendung gespeicherte Hashwert des gesuchten Passworts extrahiert werden.

Wie anhand der Analyse von Realwelt-Passwörtern in Kapitel 4.6 gezeigt werden konnte, integrieren bis zu rund 57% der Benutzer persönliche Informationen in ihre Passwörter. Demzufolge werden die persönlichen Informationen des Benutzers bei der Generierung möglicher Passwort-Kandidaten durch den Präprozessor als Bestandteil der nachfolgenden Vorbereitungsphase miteinbezogen.
Die Eingabe der Daten für die spätere Verarbeitung durch den Präprozessor erfolgt über ein Template im Toml-Format[24]. Hierbei können die folgenden Arten von Informationen eingegeben werden:

- persönliche Informationen: Vorname, Spitzname, Nachname, Geburtsort, Adresse (Straße, Hausnummer, Wohnort, Postleitzahl).
- persönliche Informationen von zweiten Personen (Familienangehörige, Lebenspartner, Freunde).
- Email-Adresse und Telefonnummer.
- bekannte Benutzernamen bei dem gleichen oder weiteren Diensten.
- bekannte Passwörter bei dem gleichen oder weiteren Diensten.
- Interessen und Hobbys des Benutzers.
- URLs von Wikipedia-Seiten mit Bezug zu den Benutzerinteressen.

Die Eingabe der Vergleichspasswörter erfolgt zwecks Vermeidung der in Abschnitt 4.4.2 beschriebenen Möglichkeit einer Fehlfragmentierung auf Fragment-Basis. Demnach werden die Fragmente eines Passworts bereits getrennt voneinander eingegeben. Hierdurch wird sichergestellt, dass der Algorithmus die Passwort-Elemente korrekt verarbeitet.
Bei dem Passwort „gitgudkkthxbye“ beispielsweise werden die 4 einzelnen Fragmente „gitgud“, „kk“, „thx“ und „bye“ eingegeben, sodass der Algorithmus trotz vollständiger Kleinschreibung des Passworts 4 Fragmente erkennt und separat verarbeiten kann. In Anhang A ist das ausgefüllte Template am Beispiel der ersten Evaluations-Aufgabe abgebildet.

5.5 Vorbereitungsphase

Nach erfolgreicher Extraktion des Hashwerts des Verschlüsselungsgegenstands, der Identifizierung des Verschlüsselungsalgorithmus sowie der Aufnahme aller vorliegenden persönlichen Informationen des Benutzers beginnt die Vorbereitungsphase des Modells.

[24] Toml ist eine Konfigurationsdatei, welche eine einfach Weiterverarbeitung der enthaltenen Daten ermöglicht [33].

Die Vorbereitungsphase besteht aus dem Präprozessor, der, wie bereits erwähnt, der Vorbereitung des benutzerspezifischen Angriffs dient. Seine Aufgabe ist die Generierung der wahrscheinlichsten Passwort-Kandidaten zur Rekonstruktion des Passworts eines Benutzers. Wie in Abbildung 5.1 gezeigt, findet zunächst eine Analyse der vorliegenden Vergleichspasswörter statt. Im Anschluss werden neue Muster-Klassen generiert, die mit Fragmenten der im letzten Schritt generierten Wortlisten befüllt werden. Auf diese Weise entstehen für den Angriffsprozess zu testende Passwort-Kandidaten. Im Folgenden werden die drei Hauptprozesse des Präprozessors näher erläutert.

5.5.1 Analyse der Passwörter

Das Ziel der Passwortanalyse ist die Identifizierung der Muster-Klassen sowie der Passwort-Fragmente der vom Benutzer vorliegenden Passwörter (im Folgenden „Basis-Passwörter“ genannt).

Wie die Ergebnisse der Analyse des Passwort-Leaks hinsichtlich des Aspekts der Wiederverwendung (siehe Abschnitt 4.5) zeigen, verwenden $66,04\%$ der Benutzer mindestens eines ihrer Passwörter mehrfach. Dies impliziert eine exakte Wiederverwendung der Struktur sowie des Inhalts des Passworts.
Auf struktureller Ebene verwenden weiterhin $39,11\%$ der Benutzer die Muster-Klasse eines Passworts für mindestens ein weiteres Passwort. Hieraus wird deutlich, dass Benutzer bei der Generierung eines neuen Passworts an den bisher verwendeten Passwort-Mustern festhalten.
Auf inhaltlicher Ebene verwenden weiterhin $25,98\%$ der Benutzer ein Passwort-Fragment für mindestens ein weiteres Passwort exakt wieder.[25] Hieraus wird deutlich, dass Benutzer bei der Generierung eines neuen Passworts auf bereits verwendete Passwort-Elemente zurückgreifen.
Demzufolge bilden die Muster-Klassen der Basis-Passwörter (im Folgenden „Basis-Muster-Klasse“ genannt) die strukturelle Basis bei der Erstellung von Passwort-Kandidaten und die Fragmente der Basis-Passwörter (im Folgenden „Basis-Fragment“ genannt) die inhaltliche Basis.

5.5.1.1 Strukturelle Analyse

In der strukturellen Analyse werden die Basis-Muster-Klassen erstellt und ihre Erfolgswahrscheinlichkeit, die Struktur des gesuchten Passworts zu sein, berechnet.

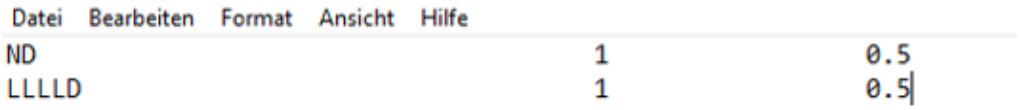
[25] Hierbei handel es sich um die Benutzer der partiellen Wiederverwendung auf Passwort-Ebene sowie der exakten Wiederverwendung auf Fragment-Ebene.

Definition 5.1 *Sei β_{anzahl} die Auftrittshäufigkeit der Muster-Klasse β_{Basis} und K die Gesamtzahl an Basis-Passwörtern. Demzufolge ist die Erfolgswahrscheinlichkeit einer Basis-Muster-Klasse $P(\beta_{Basis})$ gegeben durch:*

$$P(\beta_{Basis}) = \frac{\beta_{anzahl}}{K} \tag{5.1}$$

Am Beispiel der ersten Evaluations-Aufgabe besitzt das Passwort „Fortnite1337“ die Muster-Klasse N^+D^+ und das Passwort „gitgudkkthxbye69“ aufgrund der fragmentierten Eingabe in das Template die Muster-Klasse $L^+L^+L^+L^+D^+$. Beide Muster-Klassen besitzen unter Anwendung der Definition 5.1 eine Erfolgswahrscheinlichkeit in Höhe ihrer Auftrittshäufigkeit von 50%. Abbildung 5.2 zeigt die entsprechende Ausgabe des Präprozessors.

Abbildung 5.2: Ausgabe der Basis-Muster-Klassen



Quelle: python-Implementierung des Präprozessors.

5.5.1.2 Inhaltliche Analyse

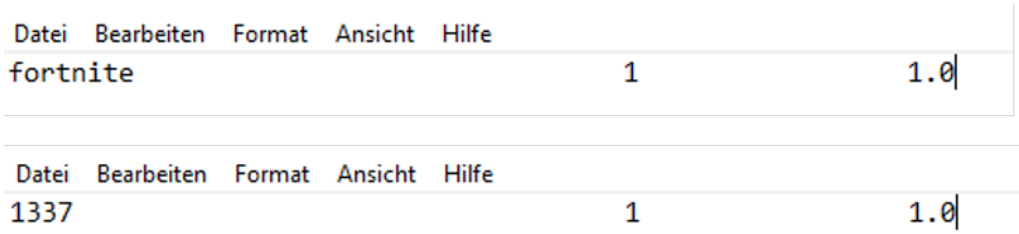
In der inhaltlichen Analyse werden die Fragmente der Basis-Passwörter in Abhängigkeit ihrer Zugehörigkeit zu einer Fragment-Klasse der alphabetischen Wortliste, der numerischen Wortliste oder der Wortliste der Sonderzeichen zugeordnet.
Der Zuordnungsprozess erfolgt pro Muster-Klasse, sodass im Ergebnis pro Muster-Klasse drei Wortlisten vorliegen. Analog zu den Basis-Muster-Klassen wird ihre Erfolgswahrscheinlichkeit, ein Element des gesuchten Passworts zu sein, berechnet.

Definition 5.2 *Sei $\alpha_{anzahl_{\beta,\gamma}}$ die Auftrittshäufigkeit des Fragments α innerhalb der Muster-Klasse β und innerhalb der Fragment-Klasse γ. Weiterhin sei F die Gesamtzahl an Fragmenten innerhalb der Muster-Klasse β sowie innerhalb der Fragment-Klasse γ. Demzufolge ist die Erfolgswahrscheinlichkeit eines Basis-Fragments $P(\alpha_{Basis})$ gegeben durch:*

$$P(\alpha_{Basis}) = \frac{\alpha_{anzahl_{\beta,\gamma}}}{F_{\beta,\gamma}} \tag{5.2}$$

Am herangezogenen Beispiel werden, wie in Abbildung 5.3 dargestellt, die Fragmente „fortnite“ und „1337“ der alphabetischen bzw. numerischen Wortliste der Basis-Muster-Klasse N^+D^+ zugeordnet. Da sie die einzigen Fragmente innerhalb der Fragment-Klasse sind, besitzen sie eine Erfolgswahrscheinlichkeit von 100%.

Abbildung 5.3: Ausgabe der Wortlisten der Muster-Klasse N^+D^+



Quelle: python-Implementierung des Präprozessors.

Wie in Abbildung 5.4 zu gezeigt, ist das Fragment „69“ Bestandteil der numerischen Wortliste der Basis-Muster-Klasse $L^+L^+L^+L^+D^+$ mit einer Erfolgswahrscheinlichkeit von 100%. Die Fragmente „gitgud“,„kk“,„thx“ und „bye“ werden der alphabetischen Wortliste zugeordnet und besitzen eine Erfolgswahrscheinlichkeit in Höhe von:

$$P(\alpha_{Basis}) = \frac{1}{4} \text{ mit } \alpha_{Basis} \in \{gitgud, kk, thx, bye\}$$

Abbildung 5.4: Ausgabe der Wortlisten der Muster-Klasse $L^+L^+L^+L^+D^+$

Quelle: python-Implementierung des Präprozessors.

Wie zudem zu erkennen ist, werden alle Fragmente in Kleinbuchstaben in die Wortlisten aufgenommen. Ein Veränderung der Groß- und Kleinschreibung findet zum Zeitpunkt der Kandidatengenerierung über die zusätzlich definierten Muster-Klassen-Elemente N^+ und R^+ statt.

5.5.2 Generierung neuer Muster-Klassen

Das Ziel des Muster-Generators ist die Erstellung der Muster-Klasse des gesuchten Passworts auf Grundlage der Basis-Muster-Klassen des Benutzers.
Hierfür werden die in Abschnitt 4.5 identifizierten Modifikationen zwischen den Passwörtern eines Benutzern herangezogen. Die Modifikationen werden im Folgenden über

die Definition von sogenannten „Modifikationsprinzipien“ von Fragment- bzw. Zeichen-Ebene auf Muster-Klasse-Ebene abstrahiert.
Über die Anwendung der Prinzipien auf eine Muster-Klasse werden neue Muster-Klassen generiert, die im Angriffsprozess zusätzlich befüllt werden und somit Passwort-Kandidaten mit neuen Strukturen generieren. Die Mustergenerierung führt demnach zu einer Erweiterung des zu testenden Passwort-Kandidatenraums mit dem Ziel, die Erfolgswahrscheinlichkeit der Rekonstruktion zu erhöhen.

5.5.2.1 Identifizierte Modifikationen

Grundsätzlich generiert jedes Modifikationsprinzip eine oder mehrere neue Muster-Klassen. Die Fragmentanzahl einer Muster-Klasse wird pro Anwendung eines Prinzips um maximal ein Fragment verändert.
Die Ausführung des Muster-Generators erfolgt rundenweise. In der ersten Runde bilden die Basis-Muster-Klassen des Benutzers die Generierungsgrundlage. In den folgenden Runden werden jeweils die aus der Vorrunde generierten Muster-Klassen als Grundlage herangezogen.
Da im Anwendungsfall keine Informationen über den Erstellungszeitpunkt der Basis-Passwörter sowie des gesuchten Passworts vorliegen, führen einige Prinzipien, zusätzlich zur zugrundeliegenden Modifikation, eine Rückwärtstransformation durch. Im Folgenden werden die einzelnen Modifikationsprinzipien des Muster-Generators vorgestellt und zur Veranschlichung auf eine der Basis-Muster-Klasse der ersten Evaluations-Aufgabe angewendet.

1. Modifikationsprinzip

$0,73\%$ der Benutzer duplizieren ein bereits verwendetes Passwort. Demzufolge besteht das erste Modifikationsprinzip aus der Duplizierung der gesamten Muster-Klasse. Liegt bereits eine duplizierte Muster-Klasse, in Form einer zweifachen Aneinanderreihung vor, wird sie halbiert. Dieses Prinzip findet ausschließlich in der ersten Generierungsrunde Anwendung und führt als einziges Prinzip zu einer Verdopplung bzw. Halbierung der Gesamtfragmentanzahl.
Aus der Muster-Klasse N^+D^+ wird unter Anwendung des Prinzips die Muster-Klasse $N^+D^+N^+D^+$.

2. Modifikationsprinzip

$4,06\%$ der Benutzer duplizieren ein alphabetisches Fragment, $1,31\%$ ein numerisches Fragment und $0,03\%$ ein Sonderzeichen. Demzufolge besteht die zweite Modifikation in der Duplizierung jedes Muster-Klassen-Elements. Die Anzahl an neu generierten Muster-Klassen entspricht folglich der Anzahl an Elementen der zugrundeliegenden Muster-Klasse.

Die Muster-Klasse N^+D^+ generiert somit die zwei neuen Strukturen $N^+N^+D^+$ und $N^+D^+D^+$.

3. Modifikationsprinzip

Dieses Prinzip behandelt einen Sonderfall des zweiten Modifikationsprinzips. Liegen die Muster-Klassen-Elemente N^+ oder R^+ vor, werden diese zusätzlich um die Elemente L^+ bzw. U^+ erweitert, um keine Veränderung in der Groß- und Kleinschreibung zu erzeugen.
Aus der Muster-Klasse N^+D^+ resultiert somit die Klasse $N^+L^+D^+$.

4. Modifikationsprinzip

Das vierte Prinzip behandelt die Rückwärtstransformation einer vorliegenden Duplikation. Im Fall von zwei oder mehreren, gleichen, aufeinanderfolgenden Muster-Klassen-Elementen wird rundenweise eins gelöscht. Sollte die Elemente L^+ bzw. U^+ auf die Muster-Klassen-Elementen N^+ und R^+ folgen, werden diese ebenfalls gelöscht.
Die Muster-Klasse $L^+L^+L^+L^+D^+$ erzeugt in der ersten Runde somit die Klasse $L^+L^+L^+D^+$.

5. Modifikationsprinzip

$0,63\%$ der Benutzer stellen einem Passwort ein Fragment voran. $7,9\%$ der Benutzer hängen einem Passwort ein Fragment an und $2,16\%$ der Benutzer nehmen beide Operationen gleichzeitig vor. Beliebte Erweiterungsfragmente sind hierbei Zahlen oder Sonderzeichen.[26]
Demzufolge erweitert das fünfte Prinzip eine vorliegende Muster-Klasse um ein D^+- und ein S^+-Element zu Beginn sowie am Ende eines Passworts. Eine beidseitige Erweiterung erfolgt erst in der zweiten Generierungsrunde. Das Prinzip findet nur Anwendung, falls die Muster-Klasse nicht bereits mit einer Zahl oder einem Sonderzeichen beginnt bzw. auf diesem endet.
Die Muster-Klasse N^+D^+ erzeugt somit in der ersten Generierungsrunde die Muster-Klassen $D^+N^+D^+$, $S^+N^+D^+$ und $N^+D^+S^+$.

6. Modifikationsprinzip

Das sechste Prinzip behandelt die Rückwärtstransformation einer Passworterweiterung (5. Prinzip). Um eine durch den Benutzer vorgenommmene Erweiterungen rückgängig zu machen, werden vorangestellte oder angehängte D^+- oder S^+-Elemente gelöscht.
Die Muster-Klasse N^+D^+ generiert daher unter Anwendung dieses Prinzips die Muster-Klasse N^+.

[26] Weitere Erweiterungsfragmente stellen einzelne Buchstaben dar, welche im Angriffsprozess über den Einsatz eines Regelwerks berücksichtigt werden können (siehe Abschnitt 5.6.3.4).

7. Modifikationsprinzip

$8,36\%$ der Benutzer übernehmen ein Wort eines bereits verwendeten Passworts in veränderter Groß- und Kleinschreibung. $6,01\%$ der Benutzern verändern hierbei den Anfangsbuchstaben zur Großschreibung (Muster-Klasse L^+ wird zur Muster-Klasse N^+). $2,22\%$ der Benutzer verändern die Schreibweise zu einer vollständigen Großschreibung (Muster-Klasse U^+) und $0,13\%$ konvertieren alle Buchstaben bis auf den ersten zu Großbuchstaben (Muster-Klasse R^+).
Im Rahmen der Umsetzung werden zunächst alle verwendeten Schreibweisen des Benutzers identifiziert und in Folge jedes alphabetische Muster-Klassen-Element durch ein identifiziertes Elemente anderer Schreibweise ersetzt. Die Reihenfolge des Ersetzens erfolgt in absteigender Auftrittshäufigkeit der Elemente. Im Falle einer vorliegenden Muster-Klasse mit mehr als drei Fragmenten wird aus Performancegründen pro Generierungsrunde lediglich ein alphabetisches Fragment ersetzt.
Für den Fall, dass der Benutzer keine unterschiedlichen Schreibvarianten verwendet, wird die zu Beginn erwähnte allgemeine Anwendungshäufigkeit zur Priorisierung der Schreibweise herangezogen. Der Ersetzungsprozess erfolgt somit in der Reihenfolge $[L^+, N^+, U^+, R^+]$. Das jeweilig betrachtete Muster-Klassen-Element wird hierbei ausschließlich durch die wahrscheinlicheren Elemente ersetzt.
Im betrachteten Beispiel verwendet der Benutzer die Schreibweisen der Muster-Klassen-Elemente L^+ und N^+. Die Muster-Klasse $L^+L^+L^+L^+D^+$ generiert somit in der ersten Runde die Muster-Klasse $N^+L^+L^+L^+D^+$.

8. Modifikationsprinzip

0.051% der Benutzer rotieren ein Passwort um eine Fragmentposition. Demzufolge rotiert das achte Prinzip die Fragmente einer Muster-Klasse um eine Stelle nach rechts sowie um eine Stelle nach links.
Aus der Muster-Klasse $L^+L^+L^+L^+D^+$ werden somit die zwei neuen Muster-Klassen $L^+L^+L^+D^+L^+$ und $D^+L^+L^+L^+L^+$.

Über die Angabe der Rundenanzahl kann die Anzahl neu generierter Muster-Klassen reguliert werden. Grundsätzlich entfernen sich die neuen Muster-Klassen mit zunehmender Rundenanzahl von den Basis-Muster-Klassen des Benutzers. Bei Anwendung mehrerer Prinzipien auf eine Basis-Muster-Klasse können Duplikate, also bereits generierte Muster-Klassen, entstehen. Da ausgeschlossen ist, dass diese eine höhere Erfolgsswahrscheinlichkeit besitzen als zu einem früheren Generierungszeitpunkt, werden sie verworfen (siehe Abschnitt 5.5.2.2). Aus Performancegründen wird eine Fragmentobergrenze von 5 Fragmenten festgelegt. Neu generierte Muster-Klassen mit einer höheren Fragmentanzahl finden keine Berücksichtigung im Angriffsprozess.

5.5.2.2 Erfolgswahrscheinlichkeit der Muster-Klassen

Wie bereits erläutert, besitzen die Basis-Muster-Klassen eine Erfolgswahrscheinlichkeit in Höhe ihrer Auftrittswahrscheinlichkeit (siehe Definition 5.1). Gemeinsam mit der Erfolgswahrscheinlichkeit der Modifikationsprinzipien bestimmen sie nach Definition 5.3 die Erfolgswahrscheinlichkeit einer neu generierten Muster-Klasse.

Definition 5.3 *Sei $P(\beta_{Basis})$ die Erfolgswahrscheinlichkeit der Basis-Muster-Klasse β_{Basis}. Weiterhin sei $P(mp_i)$ die Erfolgswahrscheinlichkeit des i-ten Modifikationsprinzips $mp_i \in MP$, wobei MP die Menge aller angewendeten Modifikationsprinzipien ist. Demzufolge ist die Erfolgswahrscheinlichkeit einer neuen Muster-Klasse $P(\beta)$ gegeben durch:*

$$P(\beta) = P(\beta_{Basis}) \cdot \prod_{i \in MP} P(mp_i) \tag{5.3}$$

Die Bestimmung der Erfolgswahrscheinlichkeit eines Modifikationsprinzipis $P(mp_i)$ erfolgt über die Gewichtung der Prinzipien anhand des bisherigen Anwendungsverhalten des Benutzers. Unterscheiden sich zwei Basis-Passwörter eines Benutzers beispielsweise durch ein angehängtes Ausrufezeichen, ist die Vermutung naheliegend, dass er weitere seiner Passwörter um ein Ausrufezeichen erweitert. Demzufolge wird das fünfte Modifikationsprinzip stärker gewichtet als beispielsweise eine Fragmentrotation im Rahmen des achten Prinzips.

Findet keines der Prinzipien durch den Benutzer Anwendung, bestimmt sich die Gewichtung anhand der allgemeinen Anwendungshäufigkeit der Benutzer des Passwort-Leaks. Die Anzahl an Benutzern, die die jeweilige Modifikation bei ihrer Passwortgenerierung angewendet haben, repräsentiert die Beliebtheit der Modifikation und ermöglicht auf diese Weise die Zuordnung einer Anwendungswahrscheinlichkeit. Insgesamt wenden 759.678 Benutzer $(24,51\%)$ eines der Modifikationsprinzipien an, woraus sich die folgende Gewichtung ergibt:

Tabelle 5.1: Gewichtung der Modifikationsprinzipien

Modifikationsprinzip	Häufigkeit	Erfolgswahrscheinlichkeit (in %)
1.	22.740	2,99
2./ 3./ 4.	167.622	22,06
5./ 6.	331.313	43,61
7.	259.171	34,12
8.	1.572	0,21
Σ	759.678	100

Quelle: eigene Darstellung.

Finden bereits, wie im zuvor beschriebenen Beispiel, Prinzipien zwischen den Basis-Muster-Klassen des Benutzers Anwendung, erhalten diese die stärkste Gewichtung. Die nicht angewendeten Prinzipien erhalten in analoger Reihenfolge zu der Gewichtung in Tabelle 5.1 sinkende Erfolgswahrscheinlichkeiten. In Summe ergeben die Gewichtungswerte der Prinzipien immer 1. Sollen alle Prinzipien von einem Benutzer bereits angewendet worden sein, liegt eine Gleichverteilung mit einer jeweiligen Erfolgswahrscheinlichkeit von 20% vor.

In Abbildung 5.5 ist die Ausgabe des Präprozessors bei Durchführung einer Generierungsrunde auf Grundlage der Basis-Muster-Klassen N^+D^+ und $L^+L^+L^+L^+D^+$ dargestellt. Die ersten beiden Spalten zeigen die neuen Muster-Klassen sowie die jeweilige Basis-Muster-Klasse. Im Anschluss ist die Anzahl an Muster-Klassen-Elementen sowie die jeweilige Erfolgswahrscheinlichkeit abgebildet.

Abbildung 5.5: Ausgabe der Muster-Klassen einer Generierungsrunde

Datei Bearbeiten Format Ansicht Hilfe

ND	ND	2	0.5
LLLLD	LLLLD	5	0.5
NDS	ND	3	0.21805
DND	ND	3	0.21805
SND	ND	3	0.21805
N	ND	1	0.21805
LLLL	LLLLD	4	0.21805
LD	ND	2	0.1706
NLLLD	LLLLD	5	0.1706
NND	ND	3	0.1103
NDD	ND	3	0.1103
NLD	ND	3	0.1103
LLLD	LLLLD	4	0.1103
NDND	ND	4	0.01495
DN	ND	2	0.00105
LLLDL	LLLLD	5	0.00105
DLLLL	LLLLD	5	0.00105

Quelle: python-Implementierung des Präprozessors.

Bei den vorliegenden Basis-Muster-Klassen findet keines der Modifikationsprinzipien Anwendung, sodass die Gewichtungswerte aus Abbildung 5.1 herangezogen werden. Die Muster-Klasse $N^+D^+S^+$ ist beispielsweise das Ergebnis der Anwendung des fünften Modifikationsprinzip, welches ein Passwort unter anderem um ein Sonderzeichen am Ende erweitert. Die Erfolgswahrscheinlichkeit ist unter Anwendung der Definition 5.3 gegeben durch:

$$\begin{aligned} P(N^+D^+S^+) &= P(N^+D^+) \cdot \prod_{i \in MP} P(mp_i) \text{ mit } MP \in \{5\} \\ &= 0,5 \cdot 0,4361 = 0,21805 = 21,805\% \end{aligned}$$

5.5.2.3 Muster-Klassen-Raum

Neben den Basis-Muster-Klassen werden im Angriffsprozess alle neu generierten Muster-Klassen nachgebildet, wodurch sie einen direkten Einfluss auf den Ressourcenaufwand des Angriffs nehmen. Mit zunehmender Anzahl an Muster-Klassen steigt der Zeitaufwand des benutzerspezifischen Angriffs.

Die Anzahl an generierten Muster-Klassen ist von mehreren Faktoren abhängig. Zum einen bestimmt sie sich durch die Anzahl an Basis-Muster-Klassen, da der Generierungsalgorithmus auf jede Muster-Klasse einzeln angewendet wird. Des Weiteren ist sie von der Anzahl an Muster-Klassen-Elementen abhängig, da ausschließlich Muster-Klassen bestehend aus maximal 5 Fragmenten berücksichtigt werden. Zudem besteht eine Abhängigkeit von dem jeweiligen Muster-Klassen-Element. So findet beispielsweise das sechste Prinzip nur Anwendung, falls der Muster-Klasse ein D^+- oder S^+-Element vorangestellt bzw. angehängt ist. Zuletzt spielt auch die Reihenfolge der Anordnung der einzelnen Muster-Klassen-Elemente eine Rolle. So findet beispielsweise das vierte Prinzip nur Anwendung, wenn zwei gleiche Elemente aufeinanderfolgen.

Für eine Abschätzung der Größe des Muster-Klassen-Raums ist in Tabelle 5.2 eine Übersicht über die minimale sowie die maximale Anzahl an generierten Muster-Klassen auf Basis einer Muster-Klasse gegeben. Die Abbildung zeigt die jeweilige Anzahl an Muster-Klassen für 1 bis 5 Fragmente bei Durchführung von 1 bis 3 Runden.

Tabelle 5.2: Muster-Klassen-Raum

		Minimale Anzahl	Maximale Anzahl
1 Fragment	1 Runde	3	9
	2 Runden	11	56
	3 Runden	27	256
2 Fragmente	1 Runde	4	12
	2 Runden	15	111
	3 Runden	41	599
3 Fragmente	1 Runde	4	18
	2 Runden	17	176
	3 Runden	30	986
4 Fragmente	1 Runde	6	24
	2 Runden	12	217
	3 Runden	24	992
5 Fragmente	1 Runde	2	17
	2 Runden	4	123
	3 Runden	9	513

Quelle: eigene Darstellung.

Wie zu erkennen ist, kann der Raum an Muster-Klassen bei mehr als einer Durchführungsrunde sehr groß werden. So werden beispielsweise auf Basis einer Muster-Klasse, bestehend aus 3 Fragmenten, bis zu 986 neue Muster-Klassen bei 3 Generierungsrunden erstellt.

5.5.3 Generierung der Wortlisten

Nach erfolgter Generierung der nachzubildenden Muster-Klassen, die die wahrscheinlichkeitsbasierte Struktur der Passwort-Kandidaten angibt, ist der letzte Schritt der Vorbereitungsphase die Erweiterung der Wortlisten pro Fragment-Klasse.
Die Einträge der Wortlisten bilden den Inhalt der Passwort-Kandidaten und werden im Angriffsprozess zur Befüllung der Muster-Klassen herangezogen. Bislang bestehen die Wortlisten ausschließlich aus den Basis-Fragmenten der Basis-Passwörter (siehe Abschnitt 5.5.1). Zur Steigerung der Erfolgswahrscheinlichkeit der Rekonstruktion werden sie im Folgenden um benutzerspezifische Fragmente erweitert.

5.5.3.1 Alphabetische Wortliste

Die alphabetische Wortliste wird zunächst um die persönlichen Informationen des Benutzers erweitert. Die Daten werden aus dem Template bezogen, welches im Rahmen der Initialisierungsphase ausgefüllt wird. Hierbei werden ausschließlich die Daten alphabetischer Art berücksichtigt. Hierzu zählt der Vorname, der Nachname, die Straße, der Wohnort sowie der Geburtsort.
Des Weiteren werden alle Begriffe der Wortarten Nomen, Verben und Adjektive aus den für die Benutzerinteressen relevanten Wikipedia-Seiten aufgenommen.
Eine weitere Möglichkeit der Erweiterung der alphabetischen Wortliste, besteht in der Übergabe einer vordefinierte Liste in Form einer Text-Datei. Hierbei kann es sich um ein gängiges Wörterbuch sowie um eine individuell angefertigte Wortliste handeln. Die Aufnahme aller Fragmente erfolgt in Kleinbuchstaben.

5.5.3.2 Numerische Wortliste

Die numerische Wortliste wird zunächst um die einzelnen Bestandteile der numerischen Basis-Fragmente erweitert. Demzufolge werden die Zahlen an jeder Position getrennt und stellen im Ergebnis jeweils zwei neue Zahlen dar. Jede Zahl generiert somit neue Zahlen der Länge 1 bis $n-1$, wobei n der Länge der Zahl entspricht.
Des Weiteren werden die Angaben numerischer Art aus dem Template miteinbezogen. Hierbei handelt es sich zum einen um das Geburtsdatum. Jede Möglichkeit der Aneinanderreihung von Tag, Monat und Jahr zur Darstellung eines Datums mit den Längen von 2, 4, 6 und 8 Zeichen wird berücksichtigt. Ein Geburtsdatum generiert somit 22 neue Einträge für die Wortliste. Zum anderen handelt es sich um die Telefonnummer,

die Postleitzahl, die Hausnummer sowie gegebenenfalls um die Zahlen im Benutzernamen. Neben der Aufnahme in vollständiger Form werden sie ebenfalls in ihre Bestandteile verschiedener Länge zerlegt und aufgenommen.
Zudem besteht analog zur alphabetischen Wortliste, die Möglichkeit der Übergabe einer vordefinierten Liste.
Die letzte Möglichkeit der Fragmenterweiterung besteht im Einsatz des „Maskprocessors“ [18]. Hierbei handelt es sich um eine von Hashcat bereitgestellte Anwendung zur Generierung von Fragmenten. Er findet ausschließlich im Rahmen der Angriffsausweitung Anwendung (siehe Abschnitt 5.6.3.4).

Liegen keine numerischen Fragmente in den Basis-Passwörtern des Benutzers vor, wird die numerische Wortliste zusätzlich um die 20 meistverwendeten Ziffernfolgen zur vorderen oder hinteren Erweiterung eines Passworts ergänzt.[27]

5.5.3.3 Wortliste der Sonderzeichen

Die Wortliste der Sonderzeichen wird analog zum Trennungsvorgang der numerischen Fragmente um die einzelnen Bestandteile der Basis-Fragmente erweitert. Zudem werden die Sonderzeichen als Teil eines Benutzernamens aufgenommen. Des Weiteren besteht ebenfalls die Möglichkeit der Übergabe einer vordefinierten Liste.
Die letzte Möglichkeit besteht erneut im Einsatz des „Maskprocessors“, der analog zur numerischen Wortliste ausschließlich im Rahmen der Angriffsausweitung Anwendung findet (siehe Abschnitt 5.6.3.4).

Liegen keine Sonderzeichen in den Basis-Passwörtern des Benutzers vor, wird die Wortliste der Sonderzeichen zusätzlich um die 20 meistverwendeten Fragmenten mit Sonderzeichen zur vorderen oder hinteren Erweiterung eines Passworts ergänzt.[27]

5.5.3.4 Erfolgswahrscheinlichkeit der Fragmente

Zur Festlegung der Reihenfolge, in der die neuen Fragmente zur Befüllung der Muster-Klassen herangezogen werden, muss ihre Erfolgswahrscheinlichkeit, ein Fragment des gesuchten Passworts zu sein, berechnet werden. Die Basis-Fragmente besitzen nach Definition 5.2 eine Erfolgswahrscheinlichkeit in Höhe ihrer Auftrittswahrscheinlichkeit innerhalb der Muster-Klasse. Eine analoge Zuordnung bei den neuen Fragmenten ist ausgeschlossen, da sie kein Bestandteil der Basis-Passwörter sind.

Die Berechnung der Erfolgswahrscheinlichkeit der neuen Fragmente erfolgt hingegen in Abhängigkeit von der Längendifferenz des Fragments zu den Basis-Fragmenten der entsprechenden Fragment-Klasse. Diese Strategie basiert auf dem in Abschnitt 4.5 aufgezeigten Zusammenhang der Fragmentlängen zwischen den Passwörtern eines Be-

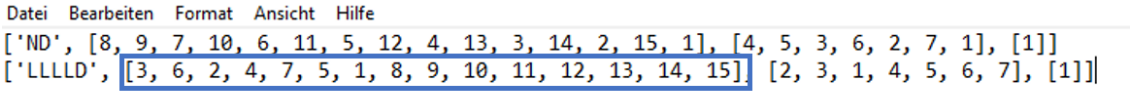
[27] Die Daten stammen aus der Untersuchung des Passwort-Leaks (siehe Abschnitt 4.5.3.2).

nutzers. Hierbei weisen alphabetische Fragmente durchschnittlich eine maximale Längenveränderung von 3 Zeichen auf und die Zahlen sowie die Sonderzeichen besitzen eine maximale Veränderung von einem bzw. 0 Zeichen. Die Benutzer weichen folglich bei der Generierung eines neuen Passworts nur geringfügig von den bisher verwendeten Längen der jeweiligen Fragment-Klasse ab.
Die Wahrscheinlichkeitszuordnung findet grundsätzlich pro Muster-Klasse statt. Grund hierfür ist der in Abschnitt 4.3 aufgezeigte Zusammenhang zwischen der Fragmentlänge und der Gesamtlänge eines Passworts. Demzufolge handelt es sich bei zum Beispiel einer Muster-Klasse mit 5 Elementen durchschnittlich um kürzere Fragmente wie bei einer Muster-Klasse bestehend aus 3 Fragmenten. Die drei Wortlisten pro Muster-Klasse besitzen somit dieselben Einträge, jedoch mit unterschiedlichen Erfolgswahrscheinlichkeiten.

Bei Betrachtung einer Muster-Klasse erhalten somit im Ergebnis die Fragmente, die die gleiche Länge wie eines der Basis-Fragmente (im Folgenden „Basis-Länge“ genannt) aufweisen, die höchste Erfolgswahrscheinlichkeit hinsichtlich ihres Auftretens im gesuchten Passwort. Im Anschluss folgen die Fragmente, die eine Längendifferenz von einem Zeichen zu den Basis-Fragmenten haben. Hierbei wird die Erhöhung einer Fragmentlänge stärker gewichtet als die Verkleinerung einer Länge. Daraufhin folgen die Fragmente mit einem Differenzwert von zwei Zeichen. Dieser Prozess wird fortgeführt, bis eine Fragmentlänge von null bzw. die doppelte Fragmentlänge des längsten Basis-Fragments erreicht ist. Diese Grenzfragmente erhalten eine Erfolgswahrscheinlichkeit von 0% und finden folglich keine Berücksichtigung in der jeweiligen Wortliste. Dieser Strategie folgend wird pro Muster-Klasse sowie pro Fragment-Klasse eine Priorisierungsliste PL der Längen erstellt.

Abbildung 5.6 zeigt am Beispiel der ersten Evaluations-Aufgabe, die Ausgabe des Präprozessors mit den Priorisierungslisten der Muster-Klassen N^+D^+ und $L^+L^+L^+L^+D^+$. Pro Muster-Klasse sind drei Listen gegeben, die jeweils die Priorisierung der Längen der alphabetischen und numerischen Wortliste sowie der Wortliste mit den Sonderzeichen angeben.

Abbildung 5.6: Ausgabe der Priorisierungslisten pro Basis-Muster-Klasse



Quelle: python-Implementierung des Präprozessors.

Grundsätzlich gilt: Je weiter hinten eine Länge in der Liste steht, desto größer ist die Längenabweichung von den Basis-Längen der Fragment-Klasse und desto geringer ist daher die Erfolgswahrscheinlichkeit des zugehörigen Fragments

Zur Veranschaulichung wird der Erstellungsprozess der alphabetischen Priorisierungsliste der Muster-Klasse $L^+L^+L^+L^+D^+$ aufgezeigt (siehe blaue Markierung in Abbildung 5.6). Die alphabetischen Basis-Fragmente „gitgud“„kk“„thx“ und „bye“ besitzen die Längen 6, 3 und 2. Sie stellen die Basis-Längen zur Erstellung der Priorisierungsliste dar. Die Anordnung der Basis-Längen erfolgt in absteigender Reihenfolge ihrer Auftrittswahrscheinlichkeit. Die Fragmentlänge 3 tritt im Gegensatz zu den anderen beiden Längen zweimal auf, wodurch sie am höchsten priorisiert wird. Die vorläufige Priorisierungsliste ist somit gegeben durch $[3,6,2]$.
Anschließend werden die Basis-Längen in vorliegender Reihenfolge bis zur Erreichung der Maximallänge von 12 Zeichen (doppelte Anzahl des längsten Basis-Fragments) durchlaufen und jeweils um den Wert von eins erhöht bzw. verringert. Sollte die jeweils berechnete Länge noch kein Bestandteil der Liste sein, wird sie angehängt. Im Ergebnis resultiert die Priorisierungsliste $[3,6,2,4,7,5,1,8,9,10,11,12]$
Damit die Wortlisten zwischen den verschiedenen Basis-Muster-Klassen dieselben Fragmente berücksichtigen, werden die bislang noch unberücksichtigten Längen anderer Muster-Klassen als die unwahrscheinlichsten Längen an die Liste angehängt.
Im betrachteten Beispiel, berücksichtigt die Muster-Klasse N^+D^+ aufgrund des Basis-Fragments „Fortnite“ (8 Zeichen) alle Längen bis einschließlich 15 Zeichen. Folglich werden die Längen 13, 14 und 15 an die alphabetische Priorisierungsliste der Muster-Klasse $L^+L^+L^+L^+D^+$ angehängt. Demzufolge lautet die finale Priorisierungsliste der Basis-Muster-Klasse $L^+L^+L^+L^+D^+$ für die Längen der alphabetischen Fragmente $[3,6,2,4,7,5,1,8,9,10,11,12,13,14,15]$ (siehe Abbildung 5.6).

Die Wahrscheinlichkeit einer Länge aus der Priorisierungsliste berechnet sich nach Definition 5.4.

Definition 5.4 *Sei PL_{anzahl} die Anzahl an Elementen der Priorisierungsliste PL. Weiterhin sei D der Differenzwert, der dem Stellenwert der Länge L innerhalb der Priorisierungsliste PL entspricht. Demzufolge ist die Wahrscheinlichkeit einer Länge $P(L_\alpha)$ eines Fragments α gegeben durch:*

$$P(L_\alpha) = \frac{(PL_{anzahl}+1)-D}{(PL_{anzahl}+1)} \tag{5.4}$$

Die Berechnung der Erfolgswahrscheinlichkeit eines Fragments erfolgt über die minimale Erfolgswahrscheinlichkeit der Basis-Fragmente sowie über die Erfolgswahrscheinlichkeit der Länge des Fragments.

Definition 5.5 *Sei $\min(P(\alpha_{Basis}))$ die minimale Erfolgswahrscheinlicheit der Basis-Fragmente α_{Basis}. Weiterhin sei L_α die Länge L des alphabetischen Fragments α. Demzufolge ist die Erfolgswahrscheinlichkeit eines alphabetischen Fragments $P(\alpha)$ gegeben durch:*

$$P(\alpha) = \min(P(\alpha_{Basis})) \cdot P(L_\alpha) \tag{5.5}$$

Die Basis-Wahrscheinlichkeit als Berechnungsgrundlage stellt sicher, dass ein Basis-Fragment des Benutzers stets eine höhere Erfolgswahrscheinlichkeit bezüglich seines Auftretens im zu rekonstruierenden Passwort besitzt als bislang nicht verwendete Fragmente.

Liegt kein alphabetisches, numerisches oder Sonderzeichen-Fragment in den Basis-Passwörtern vor, wird jeweils eine Basis-Länge zur Erstellung der Priorisierungsliste von 6, 3 bzw. 1 Zeichen angenommen. Hierbei handelt es sich um die durchschnittliche bzw. mediale Fragmentlänge der jeweiligen Fragment-Klassen, gemessen an den Passwörtern des Leaks. Als Basis-Wahrscheinlichkeit wird der prozentuale Anteil der Fragmente der jeweiligen Klasse an der Gesamtfragmentanzahl der Klasse in Höhe von $60,69\%$, $35,87\%$ bzw. $3,44\%$ herangezogen.

Abbildung 5.7 zeigt am herangezogenen Beispiel einen Ausschnitt der Ausgabe des Präprozessors von den resultierenden alphabetischen Wortlisten.

Abbildung 5.7: Ausgabe der alphabetischen Wortlisten der Muster-Klassen $L^+L^+L^+L^+D^+$ (links) und N^+D^+ (rechts)

Datei Bearbeiten Format Ansicht Hilfe

gitgud	1	0.25
kk	1	0.25
thx	1	0.25
bye	1	0.25
emp	3	0.234375
pay	3	0.234375
rod	3	0.234375
glider	6	0.21875
fandom	6	0.21875
prison	6	0.21875
us	2	0.203125
in	2	0.203125
go	2	0.203125
krug	4	0.1875
card	4	0.1875
ammo	4	0.1875
nerf	4	0.1875
clan	4	0.1875
epic	4	0.1875
limited	7	0.171875
destroy	7	0.171875
anarchy	7	0.171875
cheaten	7	0.171875
harpoon	7	0.171875
klein	5	0.15625
melee	5	0.15625
order	5	0.15625
polar	5	0.15625
woods	5	0.15625
fortnite	8	0.125
fortress	8	0.125
crossbow	8	0.125
charlotta	9	0.109375
structures	10	0.09375
eliminated	10	0.09375

Datei Bearbeiten Format Ansicht Hilfe

fortnite	1	1.0
augsburg	8	0.9375
hexakill	8	0.9375
valorant	8	0.9375
starcraft	9	0.875
resources	9	0.875
publisher	9	0.875
killing	7	0.8125
kinetic	7	0.8125
eliminated	10	0.75
foundation	10	0.75
samuel	6	0.6875
online	6	0.6875
gaming	6	0.6875
gitgud	6	0.6875
goldlimited	11	0.625
datasegment	11	0.625
store	5	0.5625
first	5	0.5625
melee	5	0.5625
woods	5	0.5625
farmaccounts	12	0.5
krug	4	0.4375
card	4	0.4375
guns	4	0.4375
trap	4	0.4375
shotgunsniper	13	0.375
stormtroopers	13	0.375
thx	3	0.3125
bye	3	0.3125
pistolshotguns	14	0.25
kk	2	0.1875
roundsresources	15	0.125
lightsaberparty	15	0.125
s	1	0.0625

Quelle: python-Implementierung des Präprozessors.

Bei Betrachtung des Fragments „fortnite“ (blau markiert) wird zum Beispiel die Zuordnung der Erfolgswahrscheinlichkeit in Abhängigkeit der Muster-Klasse deutlich.

Während es als Bestandteil der Muster-Klasse N^+D^+, in der es ein Basis-Fragment darstellt, eine Erfolgswahrscheinlichkeit von 100% besitzt (siehe Abschnitt 5.5.1.2), ist die Wahrscheinlichkeit als Teil der Muster-Klasse $L^+L^+L^+L^+D^+$ unter Anwendung der Definition 5.5 gegeben durch:

$$\begin{aligned} P(\alpha_{fortnite}) &= \min(P(\alpha_{gitgud}), P(\alpha_{kk}), P(\alpha_{thx}), P(\alpha_{bye})) \cdot P(L_{fortnite}) \\ &= 0,25 \cdot 0,5 = 0,125 = 12,5\% \end{aligned}$$

Die Wahrscheinlichkeit der Länge des „fortnite“-Fragments in Höhe von 8 Zeichen ist entsprechend der Priorisierungsliste aus Abbildung 5.1 unter Anwendung der Definition 5.4 gegeben durch:

$$P(L_{fortnite} = L_8) = \frac{16-8}{16} = 0,5 = 50\%$$

wobei der Differenzwert $D = 8$ und die Anzahl an Elementen der Priorisierungsliste $PL_{anzahl} = 15$ entspricht.

In Abbildung 5.8 ist ein Ausschnitt der Ausgabe der resultierenden numerischen Wortlisten der zwei Muster-Klassen gegeben.

Abbildung 5.8: Ausgabe der numerischen Wortlisten der Muster-Klassen $L^+L^+L^+L^+D^+$ (links) und N^+D^+ (rechts)

Datei Bearbeiten Format Ansicht Hilfe

69	1	1.0
13	2	0.875
33	2	0.875
37	2	0.875
25	2	0.875
08	2	0.875
83	2	0.875
63	2	0.875
36	2	0.875
68	2	0.875
92	2	0.875
133	3	0.75
337	3	0.75
863	3	0.75
636	3	0.75
368	3	0.75
1	1	0.625
3	1	0.625
7	1	0.625
6	1	0.625
9	1	0.625
8	1	0.625
2	1	0.625
1111	4	0.5
1983	4	0.5
2508	4	0.5
0825	4	0.5
0883	4	0.5
8308	4	0.5
6368	4	0.5
1337	4	0.5
83425	5	0.375
86368	5	0.375
081983	6	0.25
198308	6	0.25
251983	6	0.25

Datei Bearbeiten Format Ansicht Hilfe

1337	1	1.0
1111	4	0.875
1983	4	0.875
2508	4	0.875
0825	4	0.875
0883	4	0.875
8308	4	0.875
2583	4	0.875
83425	5	0.75
86368	5	0.75
133	3	0.625
337	3	0.625
863	3	0.625
636	3	0.625
368	3	0.625
081983	6	0.5
198308	6	0.5
251983	6	0.5
198325	6	0.5
250883	6	0.5
13	2	0.375
33	2	0.375
37	2	0.375
25	2	0.375
08	2	0.375
83	2	0.375
18	2	0.375
04	2	0.375
92	2	0.375
69	2	0.375
1	1	0.125
3	1	0.125
7	1	0.125
6	1	0.125
9	1	0.125
8	1	0.125

Quelle: python-Implementierung des Präprozessors.

Das Fragment „69“ (grün markiert) besitzt beispielsweise als Basis-Fragment der Muster-Klasse $L^+L^+L^+L^+D^+$ eine Erfolgswahrscheinlichkeit von 100%. Als Element der Muster-Klasse N^+D^+ hingegen besitzt es unter Anwendung der Definition 5.5 eine Erfolgswahrscheinlichkeit in Höhe von:

$$P(\alpha_{69}) = min(P(\alpha_{1337})) \cdot P(L_{69}) = 1,0 \cdot 0,375 = 37,5\%$$

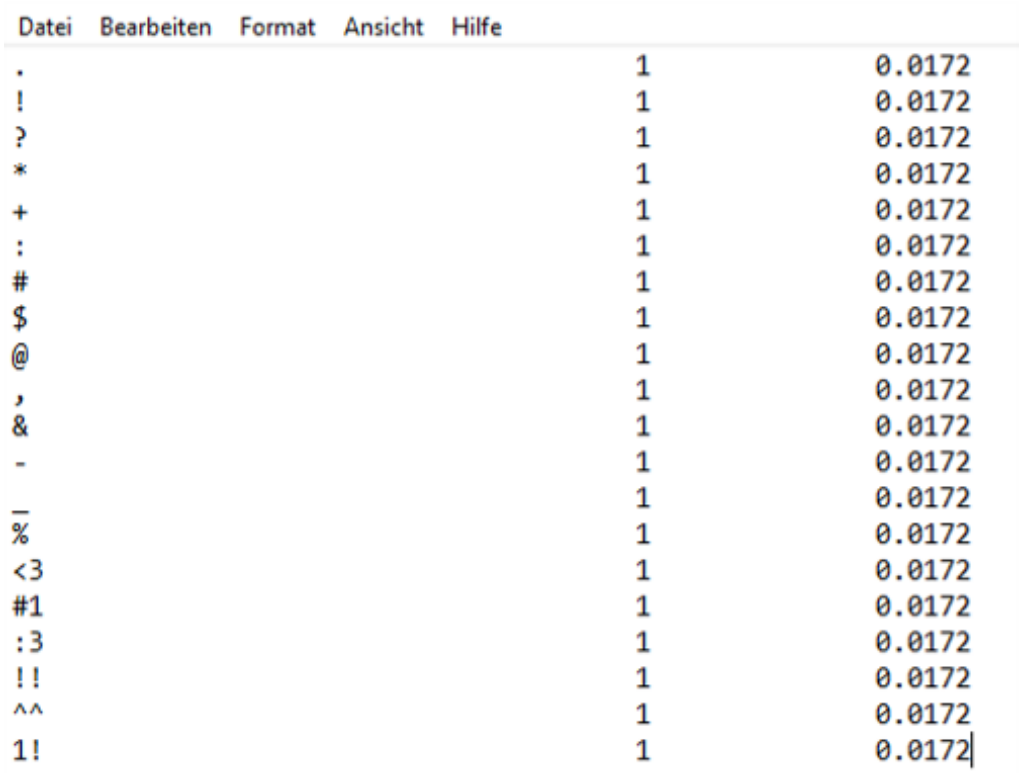
mit

$$P(L_{69} = L_2) = \frac{8-5}{8} = \frac{3}{8} = 37,5\%$$

wobei der Differenzwert $D = 5$ und die Anzahl an Elementen der Priorisierungsliste $PL_{anzahl} = 7$ entspricht.

In Abbildung 5.9 ist die Ausgabe der Sonderzeichen-Wortliste der zwei Muster-Klassen gegeben. Da keines der Basis-Passwörter ein Sonderzeichen enthält, besteht die Wortliste der Sonderzeichen aus den 20 beliebtesten Sonderzeichen-Fragmente zur Erweiterung eines Passworts. Zudem wird als Basis-Länge 1 Zeichen angenommen, sodass die Priorisierungsliste für die Längen der Sonderzeichen in Abbildung 5.1 ausschließlich die Länge 1 enthält.

Abbildung 5.9: Ausgabe der Wortliste der Sonderzeichen der Muster-Klassen $L^+L^+L^+L^+D^+$ und N^+D^+



Quelle: python-Implementierung des Präprozessors.

Die Erfolgswahrscheinlichkeit des Sonderzeichens „.“ beispielsweise ist unter Anwendung der Definition 5.5 gegeben durch:

$$P(\alpha_{.}) = min(P(\alpha_{Basis})) \cdot P(L_{.}) = 0,0344 \cdot 0,5 = 0,0172 = 1,72\%$$

mit

$$P(L_{.} = L_1) = \frac{2-1}{2} = \frac{1}{2} = 50\%$$

Am Ende der Vorbereitungsphase liegen zum einen alle nachzubildenden Muster-Klassen vor, die nach ihrer Erfolgswahrscheinlichkeit absteigend sortiert sind.
Zum anderen sind pro Basis-Muster-Klasse drei Wortlisten gegeben, die jeweils die Fragmente einer Fragment-Klasse enthalten. Inhaltlich sind die drei Wortlisten zwischen den Muster-Klassen identisch. Die Einträge unterscheiden sich ausschließlich in ihrer Erfolgswahrscheinlichkeit und damit in ihrer Sortierung innerhalb der Wortliste. Analog zu den Muster-Klassen sind auch die Fragmente innerhalb einer Wortliste in absteigender Reihenfolge ihrer Erfolgswahrscheinlichkeit sortiert. Hiermit ist die Grundlage des benutzerspezifischen Passwort-Angriffs geschaffen.

5.6 Angriffsphase

Die Angriffsphase besteht grundsätzlich aus drei voneinander unabhängigen Angriffsprozessen, die auf unterschiedlichen Annahmen basieren und verschiedene Ziele verfolgen. Der erste Angriff ist eine Bruteforce-Attacke, der kurze Passwörter rekonstruiert. Bei dem zweiten Angriff handelt es sich um einen Wörterbuch-Angriff, der Standard-Passwörter bzw. Passwörter, die in Wörterbüchern enthalten sind, rekonstruiert. Beide Angriffsprozesse werden manuell durchgeführt.
Der dritte Angriff ist die benutzerspezifische Attacke, die unter Einbezug der persönlichen Informationen des Benutzers sowie unter Berücksichtigung seiner bisherigen Passwortgestaltung das Passwort rekonstruiert. Die jeweiligen Passwort-Angriffe werden über die Anwendung Hashcat (siehe Abschnitt 2.5) ausgeführt.

Über diesen Angriffsaufbau soll die große Bandbreite an Designmöglichkeiten eines Passworts abgedeckt werden. Die Notwendigkeit der ersten beiden Angriffsprozesse ist dadurch begründet, dass die vorliegenden Vergleichspasswörter des Benutzers zwar Hinweise über die Gestaltung seiner Passwörter liefern, jedoch nicht gewährleisten, dass er diesem Konstruktionsschema auch bei dem gesuchten Passwort gefolgt ist.
Zudem soll das Vorgehen einen optimalen Kompromiss zwischen dem notwendigen Ressourcenaufwand und der Effizienz der Passwort-Rekonstruktion schaffen. Im Folgenden werden die drei Angriffsphasen näher erläutert.

5.6.1 Bruteforce-Angriff

Im ersten Angriffsprozess wird ein Bruteforce-Angriff durchgeführt. Wie in Abschnitt 2.5.1 aufgezeigt, ist diese Angriffsart eine effektive Methode zur Rekonstruktion von Passwörtern einer geringen Gesamtlänge. In Abhängigkeit des vorliegenden Verschlüsselungsalgorithmus sowie der Leistungsfähigkeit der zugrundeliegenden Hardware, wird unter Verwendung des vollständigen ASCII-Zeichensatzes, der zu testende Passwortraum individuell festgelegt.
Trotz des vermehrten Einsatzes von Passwortrichtlinien sowie einem steigendem Bewusstsein der Benutzer bezüglich der Passwortsicherheit, bestehen rund $15,5\%$ aller Passwörter des Leaks aus maximal 6 Zeichen (siehe Abschnitt 4.3.1) und sind folglich mit geringem Zeitaufwand über eine Bruteforce-Attacke rekonstruierbar.

5.6.2 Wörterbuch-Angriff

Der zweite Angriffsprozess ist der Wörterbuch-Angriff. Wie in Abschnitt 2.5.2 beschrieben, ist der Rechenaufwand von der Größe des herangezogenen Wörterbuchs sowie des Regelwerks abhängig. Die Auswahl des Wörterbuchs sowie des Regelwerks findet daher in Abhängigkeit von der Komplexität der zugrundliegenden Hashfunktion des Ziel-Hashwerts statt.
Wie anhand der Passwörter des Leaks verdeutlicht wird, stellt dieser Angriff eine effektive Methode zur Rekonstruktion von Standard-Passwörtern dar. Unter Verwendung des Wörterbuchs „pkf_6-32char“ wären rund 96% aller Passwörter des Leaks rekonstruiert worden (siehe Abschnitt 4.2.1).

5.6.3 Benutzerspezifischer Angriff

Der Hauptangriffsprozess des Modells ist der benutzerspezifische Angriff. Hierfür werden die Ergebnisse aus der Vorbereitungsphase herangezogen. Wie bereits erwähnt, sind die Muster-Klassen sowie die Fragmente in den Wortlisten bereits vorsortiert und liegen in absteigender Erfolgswahrscheinlichkeit vor. Je weiter hinten ein Element in der Liste enthalten ist, desto unwahrscheinlicher ist sein Auftreten im gesuchten Passwort.

Im Rahmen der Kandidatengenerierung werden die Muster-Klassen in vorliegender Reihenfolge einzeln herangezogen und in Abhängigkeit von ihrer Zusammensetzung aus den Muster-Klassen-Elementen L^+, U^+, N^+, R^+, D^+ und S^+ an den einzelnen Positionen mit Fragmenten befüllt. Die Fragmente stammen hierbei aus der zum Muster-Klassen-Element zugehörigen Wortliste der Basis-Muster-Klasse.
Handelt es sich um eines der Muster-Klassen-Elemente L^+, U^+, N^+ oder R^+, wird ein Fragment aus der alphabetischen Wortliste in entsprechender Groß- und Kleinschreibung eingesetzt. Liegt ein D^+-Element vor, wird ein Fragment der numerischen

Wortliste verwendet und bei dem Muster-Klassen-Element S^+ kommt ein Fragment aus der Wortliste der Sonderzeichen zum Einsatz. Nach der Befüllung aller Elemente einer Muster-Klasse liegt ein zu testender Passwort-Kandidaten vor.

5.6.3.1 Erfolgswahrscheinlichkeit der Passwort-Kandidaten

Die Erfolgswahrscheinlichkeit eines Passwort-Kandidaten berechnet sich aus der Erfolgswahrscheinlichkeit der Muster-Klasse nach Definition 5.3 sowie aus der Erfolgswahrscheinlichkeit der einzelnen Fragmente des Kandidaten nach Definition 5.5.

Definition 5.6 *Sei $P(\beta_x)$ die Erfolgswahrscheinlichkeit der Muster-Klasse β_x des zugehörigen Passwort-Kandidaten x. Weiterhin sei $P(\alpha_{l_i})$ die Erfolgswahrscheinlichkeit des Fragments α_{l_i} an Position l_i innerhalb der Muster-Klasse der Länge l. Demzufolge ist ein Maß für die Relevanz des Passwort-Kandidaten x, das gesuchte Passwort eines spezifischen Benutzers zu sein, gegeben durch:*

$$P(x) = P(\beta_x) \cdot \prod_{i=1}^{n} P(\alpha_{l_i}) \tag{5.6}$$

wobei n der Anzahl an Fragmenten eines Kandidaten entspricht.

Bei Anwendung auf das in Kapitel 5.5 herangezogene Beispiel der ersten Evaluations-Aufgabe ist die Erfolgswahrscheinlichkeit des Kandidaten „Fortnite69.“ nach der Definition 5.6 gegeben durch:

$$\begin{aligned} P(Fortnite69.) &= P(\beta_{Fortnite69.} = N^+D^+S^+) \cdot \prod_{i=1}^{3} P(\alpha_{l_i}) \\ &= P(N^+D^+) \cdot P(mp_5) \cdot P(\alpha_{Fortnite}) \cdot P(\alpha_{69}) \cdot P(\alpha_.) \\ &= 0.5 \cdot 0.4361 \cdot 1.0 \cdot 0.375 \cdot 1.0 \\ &= 0.0818 \\ &= 8.18\% \end{aligned}$$

5.6.3.2 Passwort-Kandidatenraum

Der gesamte Passwort-Kandidatenraum des benutzerspezifischen Angriffs ergibt sich aus der Gesamtzahl an Kandidaten pro nachzubildender Muster-Klasse (Muster-Klassen-Raum). Die Anzahl an generierten Kandidaten pro Muster-Klasse ist wiederum abhängig von der Anzahl an Muster-Klassen-Elementen sowie der Größe der jeweils zur Befüllung herangezogenen Wortliste.

Definition 5.7 *Sei S die Strategie des benutzerspezifischen Angriffs mit allen nachzubildenden Muster-Klassen β. Der Passwort-Kandidatenraum $Kandidatenraum(S)$ ist gegeben durch die Gesamtheit der einzelnen Kandidatenräume der Muster-Klassen [22]:*

$$Kandidatenraum(S) = \sum_{\beta \in S} Kandidatenraum(\beta) \tag{5.7}$$

Definition 5.8 *Sei $W(\beta_{l_i})$ die Anzahl an Einträgen der Wortliste des Muster-Klassen-Elements β_{l_i} an Position l_i innerhalb der Muster-Klasse der Länge l. Der Passwort-Kandidatenraum $Kandidatenraum(\beta)$ einer Muster-Klasse β ist gegeben durch [22]:*

$$Kandidatenraum(\beta) = \prod_{i=1}^{l} W(\beta_{l_i}) \tag{5.8}$$

wobei $\beta_{l_i} \in \{L^+, U^+, N^+, R^+, D^+, S^+\}$.

Bei Anwendung auf die erste Evaluations-Aufgabe besteht der gesamte Passwort-Kandidatenraum des Angriffs aus $253.067.105.648.000$ Passwörtern und ist nach Definition 5.7 gegeben durch:

$$\begin{aligned} Kandidatenraum(S) &= \sum_{\beta \in S} Kandidatenraum(\beta) \\ &= 253.067.105.648.000 \end{aligned}$$

wobei ein einzelner Kandidatenraum einer Muster-Klasse am Beispiel der Basis-Muster-Klasse N^+D^+ nach Definition 5.8 gegeben ist durch:

$$\begin{aligned} Kandidatenraum(\beta_{N^+D^+}) &= \prod_{i=1} W(\beta_{l_i}) \\ &= W(\beta_{N^+}) \cdot W(\beta_{D^+}) \\ &= 1000 \cdot 63 = 63.000 \end{aligned}$$

Zur Annäherung an das Ziel der Minimierung des Zeitaufwands der Rekonstruktion müssen die Kandidaten in absteigender Reihenfolge ihrer Erfolgswahrscheinlichkeit generiert und getestet werden. Hierfür wird auf den von Weir et. al. [40] entwickelten Algorithmus zur Generierung von Passwort-Kandidaten in absteigender Erfolgswahrscheinlichkeit zurückgegriffen, dessen Strategie im Folgenden erläutert wird.

5.6.3.3 Algorithmus der Kandidatengenerierung

Die Gewährleistung der Kandidatengenerierung in absteigender Erfolgswahrscheinlichkeit ermöglicht eine direkte Weiterverarbeitung der Passwort-Kandidaten. Der Algorithmus generiert automatisch den nächstwahrscheinlichen Kandidaten und übergibt diesen „on-the-fly“ als zu testenden Kandidaten an Hashcat („Online-Modus“). Auf diese Weise kann auf eine Zwischenspeicherung aller Passwort-Kandidaten einer Muster-Klasse und eine anschließende Sortierung dieser Kandidaten bezüglich ihrer Erfolgswahrscheinlichkeit verzichtet werden.

Die Umsetzung erfolgt über eine Warteschlange, die als Zwischenspeicher für die nächstwahrscheinlichen Passwort-Kandidaten fungiert. Die Kandidaten innerhalb der Warteschlange sind in absteigender Reihenfolge ihrer Erfolgswahrscheinlichkeit sortiert, sodass sich an erster Stelle stets der nächste zu testende Passwort-Kandidat befindet. Die Warteschlange enthält somit stets nur einen Bruchteil aller Kandidaten. Die Kandidaten in der Warteschlange werden im Folgenden als „Kind-Kandidaten“ bezeichnet und der Kandidat, auf dessen Basis sie in die Warteschlange aufgenommen werden, als „Eltern-Kandidat“. Der Eltern-Kandidat entspricht dem zum Testen an Hashcat übergebenen Passwort-Kandidat.

Der Startkandidat des Algorithmus und damit der erste Eltern-Kandidat wird aus dem ersten Eintrag der jeweiligen Wortliste der Muster-Klassen-Elemente generiert. Jeder Eintrag der Wortlisten erhält einen Indexwert, der seine Position innerhalb der Wortliste angibt (im Folgenden „Positionsindex“genannt). Die Fragmente des Startkandidaten besitzen somit jeweils den Positionsindex 0.
Ein zweiter Indexwert, der pro Passwort-Kandidat gespeichert wird, ist der „Iterationsindex“. Dieser gibt die Position an, an der die letzte Fragmentiteration innerhalb des Kandidaten stattgefunden hat. Der Startkandidat besitzt einen Iterationsindex von 0.
Jeder getestete Eltern-Kandidat generiert eine Reihe von Kind-Kandidaten, die jeweils einen möglichen nächstwahrscheinlichen Kandidaten darstellen und in die Warteschlange aufgenommen werden. Die Generierung der neuen Kind-Kandidaten erfolgt hierbei ab der Fragmentposition des Iterationsindex des Eltern-Kandidaten. Jedes Fragment (ab der Position des Iterationindex) wird durch das nächste Fragment in der dazugehörigen Wortliste ersetzt. Es findet somit eine Iteration über den Positionsindex in der jeweiligen Wortliste um einen Eintrag statt.
Die erzeugten Kind-Kandidaten enthalten jeweils nur ein vom Eltern-Kandidat unterschiedliches Fragment, sodass in Abhängigkeit von der Höhe des Iterationsindex sowie von der Gesamtlänge der Muster-Klasse <Gesamtlänge-Iterationsindex> neue Kind-Kandidaten generiert werden. Jeder Kind-Kandidat erhält den Iterationsindex an dessen Stelle die Fragmentsubstitution stattgefunden hat.

Nach jeder Generierung von neuen Kind-Kandidaten durch Ausgabe eines Eltern- Kandidaten (entspricht einer Ausführungsrunde des Algorithmus) werden die Kandidaten

in der Warteschlange nach ihrer Erfolgswahrscheinlichkeit absteigend sortiert. Der erste Eintrag der Warteschlange stellt somit den nächstwahrscheinlichen Kandidaten dar und ist damit der neue Eltern-Kandidat. Dieser wird der Warteschlange entnommen, als Passwort-Kandidat von Hashcat getestet und generiert im Anschluss, sollte es sich nicht um den gesuchten Ziel-Hashwert handeln, neue Kind-Kandidaten.

Der Algorithmus endet, sobald die Warteschlange keine weiteren Kind-Kandidaten enthält oder es sich bei einem getesteten Eltern-Kandidaten um das gesuchte Passwort handelt. Aus dem ersten Szenario folgt, dass es sich bei keinem der generierten Passwort-Kandidaten der zugrundeliegenden Muster-Klasse um das gesuchte Passwort handelt. Demzufolge wird die nächste Muster-Klasse herangezogen und der Generierungsprozess beginnt von neuem.

Der Vorteil des Algorithmus im Vergleich zum „Offline-Modus“ liegt in der Zwischenspeicherung einer Teilmenge von Passwort-Kandidaten (Elementen der Warteschlange) anstelle des gesamten Kandidatenraums. Pro Passwort-Kandidat müssen zusätzlich die folgenden Werte gespeichert werden: die Erfolgswahrscheinlichkeit, der Positionsindex pro Wortliste sowie der Iterationsindex.

5.6.3.4 Erweiterung des Kandidatenraums

Zur Ausweitung des benutzerspezifischen Angriffs nach einer erfolglosen Rekonstruktionsattacke muss der Raum an zu testenden Passwort-Kandidaten erweitert werden. Dies ist über verschiedene Maßnahmen realisierbar. Die erste Erweiterungsmaßnahme besteht in der Vergrößerung der numerischen Wortliste. Hierfür wird der bereits erwähnte „Maskprocessor“ zur Fragmentgenerierung verwendet. Der Umfang der Generierung zusätzlicher Fragmente bestimmt sich an den Längen der numerischen Basis-Fragmenten. Demnach wird der vollständige Zahlenraum der Längen der bereits verwendeten Ziffernfolgen generiert. Liegen keine numerischen Fragmente in den Basis-Passwörtern des Benutzers vor, werden vor dem Hintergrund, dass über 58% der Benutzer des Passwort-Leaks ein- oder zweistellige Zahlen verwenden, alle Ziffernfolgen der Längen von 1 und 2 generiert (siehe Tabelle 4.9).
Die zweite Erweiterungsmaßnahme besteht in der Vergrößerung der Wortliste der Sonderzeichen. Analog zur Erweiterung der Zahlen werden alle Sonderzeichen-Fragmente der Längen der Basis-Fragmente über den „Maskprocessor “ generiert. Liegen keine Sonderzeichen in den Basis-Passwörtern des Benutzers vor, wird vor dem Hintergrund, dass 96% der Benutzer des Leaks einzelne Sonderzeichen in ihrem Passwort verwenden, alle Zeichen der Länge von 1 generiert (siehe Tabelle 4.10).
Die dritte Maßnahme besteht in der Erweiterung der alphabetischen Wortliste. Hierfür können beispielsweise weitere Wörterbücher herangezogen werden oder manuell erstellte Wortlisten miteinbezogen werden.

Die vierte Erweiterung besteht in der Generierung weiterer Muster-Klassen, die im Angriffsprozess nachgebildet werden. Hierfür wird der Muster-Generator des Präprozessors für eine weitere Runde ausgeführt.
Zuletzt kann die Angriffsausführung über Hashcat um ein Regelwerk erweitert werden (siehe Abschnitt 2.5.2). Eine mögliche Funktion des Regelwerks besteht im Anhängen bzw. Voranstellen eines alphabetischen Fragments an ein Passwort. Wie die Ergebnisse der Analyse des Passwort-Leaks in Abschnitt 4.5.3.2 zeigen, ist die Verwendung von Tastatur-Mustern sowie von einzelnen Buchstaben sehr beliebt zur vorderen oder hinteren Erweiterung eines Passworts. Weitere mögliche Funktionen bilden Modifikationen auf Zeichenebene der Passwörter. Demnach kann beispielsweise die in Abschnitt 4.5.4.2 identifizierte Modifikation des Duplizierens des ersten oder letzten Buchstabens bzw. der ersten oder letzte Zahl eines Passworts umgesetzt werden. Außerdem können sogenannte Leet-speak Transformationen definiert werden. Hierbei handelt es sich um die Substitution von Buchstaben durch ähnlich aussehende Ziffern wie beispielsweise „a“ durch „@“ oder „i“ durch „!“ [42].

6 Experimente

Zur Anwendung des Modells der benutzerspezifischen Passwort-Rekonstruktion wird der Ringversuch von Europol herangezogen. Er beinhaltet fiktive, an die Realität angelehnte Polizeifälle aus dem Bereich der Passwort-Rekonstruktion. Die Fälle des Versuchs wurden von der „Zentralen Stelle für Informationstechnik im Sicherheitsbereich“ („ZITiS“[28]) entwickelt und zu Übungszwecken polizeiintern bereitgestellt.
Das Ziel des Ringversuchs ist es, mögliche Vorgehensweisen einer Rekonstruktion aufzuzeigen sowie die Effizienz der zur Verfügung stehen Angriffsmethoden zu beurteilen. Zudem sollen Einblicke in neue Phänomene im Bereich der Passwort-Rekonstruktion gewährt werden.

Jeder Fall umfasst mehrere, mit unterschiedlichen Passwörtern verschlüsselte Dateien bzw. Benutzerkonten eines Benutzers. Das Ziel ist die Rekonstruktion der jeweiligen Passwörter bei minimalem Ressourceneinsatz. Hierfür ist zum einen ein umfangreiches Profil mit persönlichen Informationen zum Benutzer gegeben. Zum anderen liegen in Form von Benutzernamen und Passwort mehrere Zugangsdaten des Benutzers für verschiedene Dienste vor. Diese Information ist optimal in den Rekonstruktionsprozess einzubinden, um die Erfolgswahrscheinlichkeit des Passwortangriffs zu erhöhen. Aufgrund der vorliegende Datengrundlage eignet sich der Ringversuch optimal zum Testen sowie zum Evaluieren des benutzerspezifischen Modells.

Im Folgenden wird das in Kapitel 5 vorgestellte Modell auf drei Fälle des Ringversuchs „Experiment 1-3“ angewendet und die Effizienz der Rekonstruktion gemessen. Die Messgrößen zur Bewertung der Effizienz sind zum einen der Stellenwert des Kandidaten im gesamten Kandidatenraum. Dies entspricht der Anzahl an Kandidaten, die bis zu einer erfolgreichen Rekonstruktion getestet werden müssen. Zum anderen wird die notwendige Rechenzeit bis zur erfolgreiche Rekonstruktion betrachtet. Die jeweiligen Passwort-Angriffe werden über die Anwendung Hashcat auf einer „Geforce GTX 1070“-Grafikkarte ausgeführt.

6.1 Experiment 1

Die Datengrundlage des ersten Experiments umfasst ein Image eines Bitlocker verschlüsselten USB-Sticks, das Logfile eines verschlüsselten Windows-Rechners sowie einen verschlüsselten Zip-Container. Des Weiteren liegen die in Tabelle 6.1 abgebildeten persönlichen Informationen über den Benutzer vor. Das vollständige Profil des Benutzers befindet sich in Anhang B.

[28] Die ZITiS ist eine Forschungs- und Entwicklungseinrichtung für technisches Know-How und agiert als technischer Dienstleister für deutsche Sicherheitsbehörden [43].

Tabelle 6.1: Experiment 1 - Persönliche Informationen des Benutzers

Benutzer	Samuel Krug		
Persönliche Informationen	Geburtstsdatum:	25.08.1983	
	Geburtsort:	Augsburg	
	Adresse:	Flurstr. 92, 86368 Gersthofen	
	Nationalität:	deutsch	
	Beruf:	Pharmazeut	
	Beziehungsstatus:	verheiratet	
Interessen	Online-Spiele (Fortnite, Valorant und Kartenspiele), E-Sport-Wetten, Teilnahmen an E-Sport Events, Mitglied in Fortnite Spielgruppen		
Zugangsdaten	Steam-Profil	Benutzername:	hexakill1111
		Passwort:	Fornite1337
	Google-Profil	Benutzername:	Krug.S.83425
		Passwort:	gitgudkkthxbye69
Zusatzinformation	Persönliche Informationen über die Ehefrau des Benutzers		

Quelle: Ringversuch der ZITiS.

6.1.1 Initialisierungsphase

Entsprechend dem Modellaufbau in Abbildung 5.1 sind die Verschlüsselungsgegenstände sowie der Benutzer bereits identifiziert. Im nächsten Schritt sind die Hashwerte der gesuchten Passwörter zwecks Durchführung einer Offline-Attacke zu extrahieren.
Bei dem Windows-Rechner handelt es sich um einen NTLM[29]-Hash, der dem Logfile entnommen wird. Für die Extraktion des Hashwerts des Zip-Containers sowie des verschlüsselten Image werden die von der Anwendung „John the Ripper“ bereitgestellten Extraktionsalgorithmen „zip2john.c“ und „bitlocker2john.c“ verwendet [9].[30]
Zum Schluss der Initialisierungsphase werden alle vorliegenden Informationen über den Benutzer, wie in Abschnitt 5.4 aufgelistet, in das Template eingetragen. Das ausgefüllte Template befindet sich zu Anhang A.

6.1.2 Vorbereitungsphase

Die Analyse der Basis-Passwörter sowie die Generierung zusätzlicher Muster-Klassen wurden bereits in Kapitel 5 aufgezeigt. Im Ergebnis liegen 18 Muster-Klassen vor, bestehend aus den 2 Basis-Muster-Klassen sowie 16 neu generierten Klassen. Sie stammen aus der Durchführung einer Generierungsrunde (siehe Abbildung 5.5).

[29] NTLM steht für „NT LAN Manager“ und ist ein Authentifzierungsprotokoll, welches bei der Benutzeranmeldung an einem Windows Rechner verwendet wird [24].
[30] Der NTLM-Hash besitzt den Hashcat-internen Hashmodus 1000, der Zip-Hash den Hashmodus 17200 und der Bitlocker-Hash den Hashmodus 22100.

Der dritte Prozess der Vorbereitungsphase besteht aus der Erstellung der Wortlisten pro Fragment-Klasse. Die alphabetische Wortliste wird, wie in Abschnitt 5.5.3.1 beschrieben, neben den persönlichen Informationen des Benutzers sowie seiner Ehefrau (bezogen aus dem Template) um Begriffe aus Wikipedia-Seiten mit Bezug zum Benutzerinteresse sowie um eine vordefinierte Wortliste erweitert.
Bei Betrachtung der Basis-Passwörter des Benutzers wird seine Affinität zu dem Spiel „Fortnite“ deutlich. Demzufolge werden zum einen Begriffe von den „Fortnite“-Wikipedia-Seiten[31] bezogen und zum anderen eine Wortliste mit spielspezifischen Begriffen von „Fortnite“-Fanseiten hinzugefügt. Im Ergebnis besteht die alphabetische Wortliste aus 1.000 Einträgen.
Die numerische Wortliste wird, entsprechend dem in Abschnitt 5.5.3.2 aufgezeigten Vorgehen, aus den numerischen Basis-Fragmenten sowie den gegebenen persönlichen Informationen numerischer Art generiert. Im Ergebnis enthält die Wortliste 63 Einträge.
Die Sonderzeichen-Wortliste besteht, wie in Abschnitt 5.5.3.3 beschrieben, aufgrund der fehlenden Sonderzeichen in den Basis-Passwörter des Benutzers, aus den 20 beliebtesten Sonderzeichen-Fragmenten zur Erweiterung eines Passworts.

6.1.3 Angriffsphase

6.1.3.1 Bruteforce-Angriff

Mit dem Ziel der Rekonstruktion von kurzen Passwörtern, besteht der erste Angriffsprozess aus der Bruteforce-Attacke. Wie in Abschnitt 5.6.1 des Modells beschrieben, wird der Umfang des Angriffs in Abhängigkeit der Geschwindigkeit der zugrundeliegenden Hashfunktion festgelegt. Abbildung 6.2 gibt pro Hashtyp eine Übersicht über die zu erwartende Rechenzeit für Passworträume verschiedener Größen.

Tabelle 6.2: Experiment 1 - Rechenzeit der Bruteforce-Angriffe[32]

Passwortlänge	Raum	Rechenzeit		
		NTLM	Pkzip	Bitlocker
$1-4$ Zeichen	$\sum_{i=1}^{4} 95^i$	< 1 Sek.	< 1 Sek.	20 Std.
$1-5$ Zeichen	$\sum_{i=1}^{5} 95^i$	1 Sek.	3 Sek.	82 T.
$1-6$ Zeichen	$\sum_{i=1}^{6} 95^i$	1 Min.	4 Min.	21 J.
$1-7$ Zeichen	$\sum_{i=1}^{7} 95^i$	2 Std.	5 Std.	2.000 J.
$1-8$ Zeichen	$\sum_{i=1}^{8} 95^i$	7 T.	21 T.	193.000 J.

Quelle: eigene Darstellung.

[31] Dies erfolgt über die Angabe der entsprechenden URL im Template (siehe Anhang A).
[32] Bei der zu erwartenden Rechenzeit handelt es sich um Rundungswerte.

Unter Berücksichtigung des Ziels eines minimalen Ressourcenaufwands wird das Passworts des NTLM-Hash sowie des Zip-Hash mit einer Rechenzeit von rund 2 und 5 Stunden, bis zu einer Passwortlänge von 7 Zeichen über die Bruteforce-Methode angegriffen. Das Passwort des Bitlocker-Hash wird hingegen mit einer zu erwartenden Rechenzeit von rund 20 Stunden lediglich bis zu einer Passwortlänge von 4 Zeichen angegriffen.
Der Grund für den höheren Zeitaufwand liegt in der hohen Komplexität der Hashfunktion der Bitlocker-Verschlüsselung. Im Vergleich zu der Hashrate der NTLM- und der Zip-Hashfunktion, die bei rund 10.000 MH/s bzw. 3.700 MH/s liegt, erzielt der Angriff des Passworts des Bitlocker-Hash lediglich eine Hashgeschwindigkeit von rund 1.100 H/s. Die jeweilige Berechnungsdauer über die festgelegten Passwortlängen hinaus liegt bei allen drei Hashtypen bereits im Tage-Bereich und somit außerhalb einer effizienten Angriffsdurchführung.

Abbildung 6.1 zeigt die Angrifftsausführung über Hashcat auf das Passwort des NTLM-Hash. Wie unter dem Spiegelpunkt „Status: Exhausted“ zu erkennen, enthält der getestete Kandidatenraum nicht das gesuchte Passwort.

Abbildung 6.1: Experiment 1 - Ausgabe nach Bruteforce-Angriff

```
Session..........: hashcat
Status...........: Exhausted
Hash.Name........: NTLM
Hash.Target......: 78ee80ea9aeb294f72a0f77c8bfc5f9f
Time.Started.....: Mon Jan 11 17:21:01 2021 (1 hour, 52 mins)
Time.Estimated...: Mon Jan 11 19:13:21 2021 (0 secs)
Guess.Mask.......: ?a?a?a?a?a?a?a [7]
Guess.Queue......: 7/7 (100.00%)
Speed.#1.........: 10080.8 MH/s (8.32ms) @ Accel:32 Loops:256 Thr:1024 Vec:1
Recovered........: 0/1 (0.00%) Digests
Progress.........: 69833729609375/69833729609375 (100.00%)
Rejected.........: 0/69833729609375 (0.00%)
Restore.Point....: 81450625/81450625 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:857344-857375 Iteration:0-256
Candidates.#1....: (}~FlM~ ->  ~? ~}?
Hardware.Mon.#1..: Temp: 72c Fan: 73% Util: 95% Core:1860MHz Mem:3802MHz Bus:8
```

Quelle: Hashcat [15].

Analog verlaufen auch die Bruteforce-Angriffe auf die zwei weiteren Passwörter ohne Erfolg. Hieraus lässt sich schlussfolgern, dass die Passwörter aus mindestens 8 bzw. 5 Zeichen bestehen.

6.1.3.2 Wörterbuch-Angriff

Wie in Abschnitt 5.6.2 beschrieben, bestimmt sich der Rechenaufwand eines Wörterbuch-Angriffs maßgeblich durch die Größe des herangezogenen Wörterbuchs.
In Folge werden, wie Tabelle 6.3 zu entnehmen, in Abhängigkeit der jeweiligen Hashgeschwindigkeit, Wörterbücher unterschiedlicher Größe herangezogen. Bei dem Bitlocker-

Hash liegt die zu erwartenden Rechenzeit unter Verwendung des „pkf_6-32char“- Wörterbuchs bei rund 32 Tage, weshalb auf das kleinere Wörterbuch „rockyou-top50k“ zurückgegriffen wird.
Vor der Angriffsdurchführung kann der Kandidatenraum aufgrund der Kenntnis über die Mindestlänge der Passwörter von 8 bzw. 5 Zeichen reduziert werden. Alle Einträge des Wörterbuchs, die eine geringere Länge aufweisen, werden ausgeschlossen. Zur Steigerung der Erfolgswahrscheinlichkeit des Angriffs wird zusätzlich ein Regelwerk mit den 64 beliebtesten Transformationen hinzugefügt.

Tabelle 6.3: Experiment 1 - Rechenzeit der Wörterbuch-Angriffe

Hashtyp	Wörterbuch	Raum	Hashrate	Rechenzeit
NTLM	pkf_6-32char	2.775.947.428	249,4 MH/s	13 Min. 16 Sek.
Zip	pkf_6-32char	2.775.947.428	223,8 MH/s	14 Min. 9 Sek.
Bitlocker	rockyou-top50k	49.776	893 H/s	1 Std. 11 Min.

Quelle: eigene Darstellung.

Im Ergebnis verlaufen alle drei Angriffe ohne Erfolg. Wie in Abbildung 6.2 ersichtlich, enthält der getestete Kandidatenraum des „pkf_6-32char“-Wörterbuchs nicht das gesuchte Passwort des NTLM-Hashwerts. Hieraus lässt sich schlussfolgern, dass der Benutzer kein Standard-Passwort verwendet.

Abbildung 6.2: Experiment 1 - Ausgabe nach Wörterbuch-Angriff

```
Session..........: hashcat
Status...........: Exhausted
Hash.Name........: NTLM
Hash.Target......: 78ee80ea9aeb294f72a0f77c8bfc5f9f
Time.Started.....: Mon Jan 11 10:04:52 2021 (13 mins, 16 secs)
Time.Estimated...: Mon Jan 11 10:18:08 2021 (0 secs)
Guess.Base.......: File (C:\Users\Eva\Documents\Leak\Analysis\Dictionary\pkf_6-32_min8.txt)
Guess.Mod........: Rules (C:\Users\Eva\Documents\Leak\Analysis\Dictionary\best64.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   249.2 MH/s (12.16ms) @ Accel:128 Loops:77 Thr:64 Vec:1
Recovered........: 0/1 (0.00%) Digests
Progress.........: 213747951956/213747951956 (100.00%)
Rejected.........: 0/213747951956 (0.00%)
Restore.Point....: 2775947428/2775947428 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-77 Iteration:0-77
Candidates.#1....: ðá┬áðáÔÇªðá┬áðÆ┬▒ðáðÄð▓ðéÐÕðá┬áðÆ┬▒ðáðÄð▓ÔÇÜ┬%ðá┬áðíÔÇØðá┬áðÆ┬▒05 -> ðòð┐Ðù
Hardware.Mon.#1..: Temp: 53c Fan: 50% Util: 38% Core:1885MHz Mem:3802MHz Bus:8
```

Quelle: Hashcat [15].

6.1.3.3 Benutzerspezifischer Angriff

Im dritten, benutzerspezifischen Angriffsprozess werden die Ergebnisse der Vorbereitungsphase herangezogen. Hierbei handelt es sich um 18 Muster-Klassen, die alphabetische und numerische Wortliste mit 1.000 bzw. 63 Einträgen sowie um die Liste der Sonderzeichen mit 33 Einträgen.

Wie in Abschnitt 5.6.3 beschrieben, werden die Muster-Klassen in absteigender Reihenfolge ihrer Erfolgswahrscheinlichkeit herangezogen und in Abhängigkeit ihrer Zusammensetzung mit Einträgen der entsprechenden Wortliste befüllt. Die Kandidaten werden im Online-Modus in absteigender Reihenfolge ihrer Erfolgswahrscheinlichkeit generiert und direkt ohne Zwischenspeicherung an Hashcat zum Testen übergeben. Erst nach dem Testen des vollständigen Kandidatenraums einer Muster-Klasse wird die nächstwahrscheinliche Muster-Klasse herangezogen und der Generierungsprozess von neuem gestartet.

Der gesamte, zu testende Kandidatenraum des Angriffs besteht, wie in Abschnitt 5.6.3.2 bereits aufgezeigt, aus $253.067.105.648.000$ Passwörtern und ist nach Definition 5.7 gegeben durch:

$$\begin{aligned} Kandidatenraum(S) &= \sum_{\beta \in S} Kandidatenraum(\beta) \\ &= 253.067.105.648.000 \end{aligned}$$

wobei $Kandidatenraum(\beta)$ die Anzahl an generierten Passwort-Kandidaten einer Muster-Klasse β aus der Gesamtheit aller Muster-Klassen der Strategie S des Modells ist.

Grundsätzlich basiert die Strategie S des Präprozessors auf der Annahme, dass Benutzer bei der Generierung eines neuen Passworts an ihren bisherigen Passwortstrukturen sowie an den bisherigen Passwort-Elementen festhalten. Folglich besitzen die Basis-Muster-Klassen sowie die Basis-Fragmente die höchste Erfolgswahrscheinlichkeit. Die Basis-Passwörter müssten nach dieser Strategie die ersten zwei Passwort-Kandidaten im Angriffsprozess sein. Wie in Abbildung 6.3, die einen Ausschnitt der Top-3 Muster-Klassen mit den jeweils 10 Passwort-Kandidaten höchster Erfolgswahrscheinlichkeit zeigt, zu beobachten, trifft dies jedoch nicht zu.

Abbildung 6.3: Experiment 1 - Ausgabe der Kandidaten höchster Erfolgswahrscheinlichkeit

N^+D^+

Fortnite1337	0.5
Augsburg1337	0.46875
Hexakill1337	0.46875
Valorant1337	0.46875
Warcraft1337	0.46875
Passages1337	0.46875
Barcraft1337	0.46875
Destroys1337	0.46875
Gamesspy1337	0.46875
Creative1337	0.46875

$L^+L^+L^+L^+D^+$

gitgudgitgudgitgudgitgud69	0.001953
kkgitgudgitgudgitgud69	0.001953
gitgudkkgitgudgitgud69	0.001953
gitgudgitgudkkgitgud69	0.001953
gitgudgitgudgitgudkk69	0.001953
thxgitgudgitgudgitgud69	0.001953
kkkkgitgudgitgud69	0.001953
kkgitgudkkgitgud69	0.001953
kkgitgudgitgudkk69	0.001953
gitgudthxgitgudgitgud69	0.001953

$N^+D^+S^+$

Fortnite1337.	0.00375
Fortnite1337!	0.00375
Fortnite1337?	0.00375
Fortnite1337*	0.00375
Fortnite1337+	0.00375
Fortnite1337:	0.00375
Fortnite1337#	0.00375
Fortnite1337$	0.00375
Fortnite1337@	0.00375
Fortnite1337,	0.00375

Quelle: python-Implementierung des Präprozessors.

Das Basis-Passwort „Fornite1337“ folgt der Strategie und bildet den ersten Kandidaten der ersten, nachzubildenden Muster-Klasse N^+D^+. Das zweite Vergleichspassworts „gitgudkkthybye69“ hingegen gehört der Muster-Klasse $L^+L^+L^+L^+D^+$ an, die

erst als zweite Muster-Klasse im Angriffsprozess nachgebildet wird. Folglich wird zunächst der vollständige Kandidatenraum der ersten Muster-Klasse N^+D^+ generiert und getestet. Verstärkt wird die Abweichung von der Strategie, dass das Passwort innerhalb der Muster-Klasse erst als $149.$ Kandidat getestet wird. Ursächlich hierfür ist die Verarbeitung der Basis-Passwörter auf Fragment-Basis. Die 4 alphabetischen Basis-Fragmente „gitgud“, „kk“, „thx“ und „bye“ besitzen dieselbe Auftrittswahrscheinlichkeit, wodurch insgesamt $4^4 = 256$ Kandidaten mit derselben Erfolgswahrscheinlichkeit von rund $0,2\%$ generiert werden (siehe 2. Spalte).
Es lässt sich somit feststellen, dass die Basis-Passwörter trotz der höchsten Erfolgswahrscheinlichkeit innerhalb der Muster-Klasse nicht die ersten zwei Passwort-Kandidaten im Angriffsprozess bilden.

Im Ergebnis der benutzerspezifischen Attacke kann zunächst keines der Passwörter rekonstruiert werden. Der Grund hierfür liegt in der Komplexität der Muster-Klassen. Analog zur Basis-Muster-Klasse $L^+L^+L^+L^+D^+$ in Abbildung 6.3 besteht die Mehrheit der 18 nachzubildenden Muster-Klassen aus 4 oder 5 Elementen, wodurch sehr lange Passwörter generiert werden. Folglich ist die Datenmenge, die der Präprozessor zur Identifizierung des nächstwahrscheinlichen Kandidaten zwischenspeichert, sehr groß. Die Kandidatengenerierung ist somit sehr zeitaufwändig, wodurch die Weiterleitung der Kandidaten an Hashcat in sehr langsamer Geschwindigkeit erfolgt.

Die Problematik spiegelt sich in der geringen Hashrate beim Angriff auf das Passwort des NTLM-Hashwerts, in Abbildung 6.4, wieder.

Abbildung 6.4: Experiment 1 - Ausgabe nach benutzerspezifischem Angriff

```
78ee80ea9aeb294f72a0f77c8bfc5f9f:TiltedTowers1337
Session..........: hashcat
Status...........: Cracked
Hash.Name........: NTLM
Hash.Target......: 78ee80ea9aeb294f72a0f77c8bfc5f9f
Time.Started.....: Fri Dec 11 11:35:35 2020 (1 day, 6 hours)
Time.Estimated...: Sat Dec 12 17:59:48 2020 (0 secs)
Guess.Base.......: Pipe
Speed.#1.........:       56 H/s (5.53ms) @ Accel:1024 Loops:1 Thr:64 Vec:1
Recovered........: 1/1 (100.00%) Digests
Progress.........: 5898240
Rejected.........: 0
Restore.Point....: 0
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: ArcaneLohnen1337 -> AbandonedThermal133
Hardware.Mon.#1..: Temp: 54c Fan:  0% Util:  0% Core:1556MHz Mem:3802MHz Bus:8
```

Quelle: Hashcat [15].

Der Angriffsumfang ist hierbei auf die gesuchte Muster-Klasse $N^+N^+D^+$ begrenzt, die unter Verfolgung der Strategie des Präprozessors erst als 10. Muster-Klasse herangezogen und nachgebildet werden würde (10. Rang in Abbildung 5.5). Durch die Einschränkung auf die gesuchte Muster-Klasse kann das Passwort erfolgreich rekonstruiert

werden und lautet im Klartext „TiltedTowers1337“. Wie unter Spiegelpunkt „Speed.#1“ erkennbar, erreicht der Angriff lediglich eine Hashgeschwindigkeit von 56 H/s, wodurch der zeitliche Rechenaufwand für $5.898.240$ Kandidaten bereits bei 1 Tag und 6 Stunden liegt.
Zum Vergleich: Die erzielte Hashrate im Rahmen des zuvor durchgeführten Wörterbuch-Angriffs liegt bei rund 250 MH/s (siehe Tabelle 6.3). Unter Annahme dieser Hashgeschwindigkeit liegt die zu erwartenden Rekonstruktionsdauer des Passworts bei unter einer Sekunde.

Eine Annäherung an diese Hashrate kann über den Wechsel vom Online-Modus in den Offline-Modus erreicht werden.[33] Anstelle der direkten Weiterleitung der Kandidaten an Hashcat werden alle generierten Kandidaten einer Muster-Klasse zwischengespeichert und nach ihrer Erfolgswahrscheinlichkeit absteigend sortiert. Die resultierende Liste repräsentiert ein Wörterbuch und kann im Rahmen eines Wörterbuch-Angriffs an Hashcat übergeben werden.
Im Folgenden findet daher eine Analyse der Rekonstruktion der Passwörter im Offline-Modus statt.[34] Als Hashgeschwindigkeit wird jeweils die Hashrate des im vorherigen Abschnitt durchgeführten Wörterbuch-Angriffs herangezogen. Die Tabelle 6.4 zeigt pro Klartext-Passwort den zu testenden Kandidatenraum sowie die bis zur erfolgreichen Rekonstruktion zu erwartende Rechenzeit.

Tabelle 6.4: Experiment 1 - Rekonstruktions-Ergebnisse im Offline-Modus[35]

Hashtyp	Passwort	Anzahl zu testender Kandidaten	Hashrate	Rechenzeit
NTLM	TiltedTowers1337	127.000.012.514.240	250 MH/s	5 T. 21 Std.
Zip	BushCamper1337	127.000.028.125.460	220 MH/s	6 T. 16 Std.
Bitlocker	stormflip1337	64.000.006.553.115	900 H/s	2.255 T.

Quelle: eigene Darstellung.

1. Passwort: NTLM-Hash

Die Rekonstruktion des Passworts „TiltedTowers1337“ des NTLM-Hashwerts benötigt bei einer Hashrate von 250 MH/s rund 5 Tage und 21 Stunden. Das Passwort entspringt der Muster-Klasse $N^+N^+D^+$, die vom Muster-Generator durch Verdopplung des alphabetischen Fragments (2. Modifikationsprinzip) generiert wird. Innerhalb der Gesamtheit an nachzubildenden Muster-Klassen befindet sie sich auf dem $10.$ Rang. Demnach werden zunächst alle Muster-Klassen der Ränge 1 bis 9 nachgebildet, bis die Struktur des gesuchten Passworts herangezogen wird. Demzufolge ist die Anzahl an zu testenden Kandidaten mit rund 127 Bio. Passwort-Kandidaten sehr groß.

[33] Zudem kann eine Annäherung über eine optimierte Implementierung des Algorithmus erzielt werden.
[34] Der Zeitaufwand sowie der notwendige Speicherplatz für die Generierung der Passwort-Listen und dessen Sortierung wird an dieser Stelle nicht berücksichtigt (siehe Kapitel 7).
[35] Bei der angenommmenen Hashrate sowie der Rechenzeit handelt es sich um Rundungswerte.

Die Erfolgswahrscheinlichkeit des Passwort-Kandidaten „TiltedTowers1337“ ist nach Definition 5.6 gegeben durch:

$$\begin{aligned} P(TiltedTowers1337) &= P(\beta_{TiltedTowers1337}) \cdot \prod_{i=1}^{3} P(\alpha_{l_i}) \\ &= P(N^+N^+D^+) \cdot P(\alpha_{Tilted}) \cdot P(\alpha_{Towers}) \cdot P(\alpha_{1337}) \\ &= 0,1103 \cdot 0,6875 \cdot 0,6875 \cdot 1,0 \\ &= 0,052134 \\ &= 5,2134\% \end{aligned}$$

Zum Zwecke der Nachvollziehbarkeit werden im Folgenden die Berechnungsschritte der Einzelwahrscheinlichkeiten des Passwort-Kandidaten aufgezeigt.
Die Erfolgswahrscheinlichkeit der Muster-Klasse $P(N^+N^+D^+)$ ist nach Definition 5.3 gegeben durch:

$$\begin{aligned} P(N^+N^+D^+) &= P(\beta_{N^+D^+}) \cdot \prod_{i \in MP} P(mp_i) \text{ mit } MP \in \{2\} \\ &= 0,5 \cdot 0,2206 = 0,1103 = 11,03\% \end{aligned}$$

Bei den alphabetischen Fragmenten „Tilted“ und „Towers“ handelt es sich um Bezeichnungen aus der „Fortnite“-Spielwelt. Die Erfolgswahrscheinlichkeit der Fragmente ist jeweils nach Definition 5.5 gegeben durch:

$$\begin{aligned} P(\alpha_{tilted/towers}) &= \min(P(\alpha_{fortnite})) \cdot P(L_{tilted/towers}) \\ &= 1,0 \cdot 0,6875 = 0,6875 = 68,75\% \end{aligned}$$

mit

$$P(L_{Tilted/Towers} = L_6) = \frac{(15+1)-5}{15+1} = 0,6875$$

wobei nach Definition 5.4 der Differenzwert $D = 5$ dem Stellenwert der Länge 6 innerhalb der Priosierungsliste entspricht. Weiterhin ist die Anzahl an Elementen der Priorisierungsliste $PL_{anzahl} = 15$.

Die Ziffernfolge „1337“ stammt aus der Leet-speak Sprache und ist ein Bestandteil des Basis-Passworts „Fortnite1337“ des Benutzers. Es handelt sich somit um ein wiederverwendetes numerisches Fragment mit folgender Erfolgswahrscheinlichkeit nach Definition 5.2:

$$P(\alpha_{1337}) = 1 = 100\%$$

2. Passwort: Zip-Hash

Die Rekonstruktion des Passworts „BushCamper1337“ des Zip-Hashwerts benötigt bei einer Hashrate von 220 MH/s rund 6 Tage und 16 Stunden. Das Passwort gehört ebenfalls der Muster-Klasse $N^+N^+D^+$ an, sodass die Anzahl an zu testenden Kandidaten mit rund 127 Bio. Passwort-Kandidaten gleichermaßen hoch ist.
Die alphabetischen Fragmente „Bush“ und „Camper“, stammen ebenfalls aus der „Fortnite“-Spielwelt und beschreiben einen als Strauch getarnten Spieler. Zudem wird erneut die Ziffernfolge „1337“ vom Benutzer verwendet.

Die Erfolgswahrscheinlichkeit des Passworts ist nach Definition 5.6 gegeben durch:

$$\begin{aligned} P(BushCamper1337) &= P(\beta_{BushCamper1337}) \cdot \prod_{i=1}^{3} P(\alpha_{l_i}) \\ &= P(N^+N^+D^+) \cdot P(\alpha_{Bush}) \cdot P(\alpha_{Camper}) \cdot P(\alpha_{1337}) \\ &= 0,1103 \cdot 0,4375 \cdot 0,6875 \cdot 1,0 \\ &= 0,033176 \\ &= 3,3176\% \end{aligned}$$

Obwohl das Passwort der gleichen Muster-Klasse wie das Passwort „TiltedTowers1337“ angehört, besitzt es eine geringere Erfolgswahrscheinlichkeit von $3,32\%$. Der Grund hierfür liegt im Längenunterschied zwischen den ersten Fragmenten der beiden Passwörter. Während das Fragment „Bush“ aus 4 Zeichen besteht und damit eine Längendifferenz von 4 Zeichen vom Basis-Fragment „Fortnite“ aufweist (Differenzwert $D = 9$), weicht das Fragment „Tilted“ lediglich um 2 Zeichen ($D = 5$) vom Basis-Fragment ab. Demzufolge besitzt es eine niedrigere Erfolgswahrscheinlichkeit (siehe Abschnitt 5.5.3.1).

3. Passwort: Bitlocker-Hash

Die Rekonstruktion des Passworts „stormflip1337“ des Bitlocker-Hashwerts benötigt bei einer Hashrate von ca. 900 H/s rund 2.255 Jahre. Das Passwort entspringt der Muster-Klasse L^+D^+, die sich auf dem 8. Rang befindet. Folglich ergibt sich ein zu testender Passwort-Kandidatenraum von rund 64 Bio. Kandidaten.
Bei dem alphabetischen Fragment „stormflip“ handelt es sich um einen Gebrauchsgegenstand aus der „Fortnite“-Spielwelt. Die Ziffernfolge „1337“ wird erneut als wiederverwendetes Fragment in das Passwort integriert.

Die Erfolgswahrscheinlichkeit des Passworts ist nach Definition 5.6 gegeben durch:

$$\begin{aligned} P(stormflip1337) &= P(\beta_{stormflip1337}) \cdot \prod_{i=1}^{2} P(\alpha_{l_i}) \\ &= P(L^+D^+) \cdot P(\alpha_{stormflip}) \cdot P(\alpha_{1337}) \\ &= 0,1706 \cdot 0,875 \cdot 1,0 \\ &= 0,149275 \\ &= 14,9275\% \end{aligned}$$

Die höhere Erfolgswahrscheinlichkeit von $14,93\%$ im Vergleich zu den anderen beiden Passwörtern mit $5,21\%$ bzw. $3,32\%$ stammt zum einen aus der höheren Erfolgswahrscheinlichkeit der zugehörigen Muster-Klasse (Rang 8 statt Rang 10).
Zum anderen unterscheidet sich die Länge des Fragments „stormflip“ lediglich um 1 Zeichen vom Basis-Fragment „Fortnite“ ($D = 2$), wodurch es folglich eine höhere Erfolgswahrscheinlichkeit als beispielsweise die Fragmente „Bush“ oder „Camper“ mit einer Längendifferenz von 4 bzw. 2 Zeichen besitzt.
Zusätzlich besteht das Passwort aus 2 anstatt aus 3 Fragmenten, wodurch ein Fragment weniger identifiziert werden muss. Folglich erhöht sich die Erfolgswahrscheinlichkeit des Passworts durch die geringere Anzahl der Fragment-Einzelwahrscheinlichkeiten.

Der Ablauf sowie die Strategie des Modells findet in den nachfolgenden Experimenten 2 und 3 analog zum Experiment 1 statt, sodass im Folgenden ausschließlich abweichende Aspekte näher erläutert werden.

6.2 Experiment 2

Die Datengrundlage des zweiten Experiments umfasst eine verschlüsselte PDF-Datei, eine verschlüsselte Office-Datei, ein verschlüsseltes Krypto-Wallet sowie eine Login-Datei der Anwendung „Putty“[36]. Zudem liegen die in Tabelle 6.5 abgebildeten, persönlichen Informationen über den Benutzer vor. Das vollständige Profil des Benutzers befindet sich in Anhang C.

6.2.1 Initialisierungsphase

Der Hashwert der Putty-Zugangsdaten kann der Login-Datei entnommen werden. Zur Extraktion der Hashwerte der drei weiteren Verschlüsselungsgegenstände kommen die Extraktionsalgorithmen „office2john.py“, „pdf2john.pl“ und „electrum2john.py“ zum Einsatz [9].[37]

[36] Hierbei handelt es sich um eine Client-Anwendung zum Verbindungsaufbau zu entfernten Servern [35].
[37] Der PDF-Hash besitzt den Hashcat-internen Hashmodus 10500, der Office-Hash den Hashmodus 9600 und der Electrum-Hash den Hashmodus 21700.

Tabelle 6.5: Experiment 2 - Persönliche Informationen des Benutzers

Benutzer	Erwin August Bachmeier		
Persönliche Informationen	Geburtstsdatum:	11.12.1975	
	Geburtsort:	Bonn	
	Adresse:	Schweinfurter Weg 99, 60599 Frankfurt a.M.	
	Nationalität:	deutsch	
	Beruf:	Steuerberater	
	Beziehungsstatus:	verheiratet	
Interessen	Schatzsuche mit Metaldetektoren, Lesen über die römische Geschichte und Mythologien, Holzbearbeitung von Möbeln, Pflegen eines Blogs über die lateinische Sprache und Dichter		
Zugangsdaten	Grapper-Profil	Benutzernamen:	Erwin bachmeier
		Passwort:	AVGVSTV.LXXV
	Twitter-Profil	Benutzername:	Erwin Bachmeier
		Passwort:	MCCXXXIV
	Ebay-Profil	Benutzername:	AugustusXIXII
		Passwort:	PANTOPOLIVM.XI
Zusatzinformation	-		

Quelle: Ringversuch der ZITiS.

6.2.2 Vorbereitungsphase

Nach der Analyse der Basis-Passwörter sowie der Anwendung des Muster-Generators für eine Runde liegen 16 Muster-Klassen vor, bestehend aus 2 Basis-Muster-Klassen und 14 zusätzlich generierten Klassen.
Bei Betrachtung der Basis-Passwörter, siehe Tabelle 6.5, wird deutlich, dass der Benutzer bevorzugt großgeschriebene, lateinische Begriffe in seinen Passwörtern verwendet. Zusätzlich modifiziert er diese durch Substitution der Buchstaben „U/W“ und „J“ durch die Buchstaben „V“ bzw. „I“. Bei der Generierung der alphabetischen Wortliste wird demnach als vordefinierte Liste eine Sammlung [11] häufig verwendeter lateinischer Begriffe herangezogen. Die Listeneinträge werden über die Kommandozeilen-Anwendung „sed“ entsprechend den identifizierten Modifikationen vorverarbeitet. Im Ergebnis besteht die alphabetische Wortliste aus 35.191 Einträgen.

Eine weitere Besonderheit in den Basis-Passwörtern stellt die Verwendung von römischen Zahlen dar. Zur Gewährleistung ihrer korrekten Verarbeitung durch den Präprozessor wird der Algorithmus um eine zusätzliche Funktion ergänzt. Vor der Verarbeitung der numerischen Fragmente zwecks Generierung der Wortliste werden die römischen Zahlen in das Dezimalsystem konvertiert.
Hierbei wird erkennbar, dass der Benutzer sein Geburtsjahr „LVVX“ (entspricht „75“) sowie sein Geburtstag „XI“ (entspricht „11“) in seine Passwörter integriert. Nach der Verarbeitung der Zahlen (siehe Abschnitt 5.5.3.2), jedoch vor der Aufnahme in die numerische Wortliste, werden sie zurück ins römische Zahlensystem konvertiert. Die Zu-

ordnung der Erfolgswahrscheinlichkeit erfolgt weiterhin auf Grundlage der Längen der römischen Schreibweise, da die Abhängigkeit zwischen den Längen der Fragmente und der Fragmentanzahl ausschließlich im finalen Passwort-Kandidaten vorliegt (siehe Abschnitt 4.3). Im Ergebnis besteht die numerische Wortliste aus 29 Einträgen.
Die Wortliste der Sonderzeichen besitzt lediglich einen Eintrag in Form des Punkts „.“ als einziges, verwendetes Sonderzeichen des Benutzers.

6.2.3 Angriffsphase

Die ersten zwei Angriffsprozesse bilden erneut die Bruteforce- und die Wörterbuch-Attacke. Analog zum Ergebnis des ersten Experiments verwendet der Benutzer weder ein besonders kurzes Passwort noch ein Standard-Passwort. Demzufolge wird an dieser Stelle auf eine Darstellung dieser Angriffsdurchführungen verzichtet und die Beschreibung mit der benutzerspezifischen Attacke fortgesetzt.

Der zu testende Kandidatenraum des benutzerspezifischen Angriffs besteht aus insgesamt $35.981.179.642$ Passwörtern und ist nach Definition 5.7 gegeben durch:

$$\begin{aligned} Kandidatenraum(S) &= \sum_{\beta \in S} Kandidatenraum(\beta) \\ &= 35.981.179.642 \end{aligned}$$

Die Abbildung 6.5 zeigt einen Ausschnitt der Top-3 Muster-Klassen mit den jeweils 10 Passwort-Kandidaten höchster Erfolgswahrscheinlichkeit.

Abbildung 6.5: Experiment 2 - Ausgabe der Kandidaten höchster Erfolgswahrscheinlichkeit

N^+D^+

```
AVGVSTV.LXXV    0.166667
PANTOPOLIVM.LXXV    0.166667
AVGVSTV.XI    0.166667
PANTOPOLIVM.XI    0.166667
HUNTING.LXXV    0.159091
HISTORY.LXXV    0.159091
HUNTING.XI    0.159091
ITALIEN.LXXV    0.159091
HISTORY.XI    0.159091
ABACTOR.LXXV    0.159091
```

D^+

```
MCCXXXIV 0.333333
MMMMMMLIX 0.291667
CCXXXIV 0.270833
MCMLXXV 0.270833
MCCLXXV 0.270833
MMMMMMMDXI 0.25
CXXIII 0.229167
MCLXXV 0.229167
MMMMMMMDXII 0.208333
XXIII 0.1875
```

$U^+S^+D^+S^+$

```
AVGVSTV.LXXV.    0.072683
PANTOPOLIVM.LXXV.    0.072683
AVGVSTV.XI.    0.072683
PANTOPOLIVM.XI.    0.072683
HUNTING.LXXV.    0.06938
HISTORY.LXXV.    0.06938
HUNTING.XI.    0.06938
ITALIEN.LXXV.    0.06938
HISTORY.XI.    0.06938
ABACTOR.LXXV.    0.06938
```

Quelle: python-Implementierung des Präprozessors.

Im Ergebnis können 3 der insgesamt 4 gesuchten Passwörter rekonstruiert werden. Für den Putty-Hashwert bietet Hashcat keine Unterstützung an, weshalb die Rekonstruktion ausschließlich theoretisch betrachtet wird.

Die jeweiligen Ergebnisse des Angriffs mit dem getesteten Kandidatenraum, der erzielten Hashrate sowie der Rechenzeit pro Klartext-Passwort sind in Tabelle 6.8 abgebildet.

Tabelle 6.6: Experiment 2 - Rekonstruktions-Ergebnisse im Online-Modus[38]

Hashtyp	Gesuchtes Passwort	Anzahl getester Kandidaten	Hashrate	Rechenzeit
PDF	TESTIMONIVM.LXXV	122.880	8218,5 kH/s	1 Min. 16 Sek.
Office	OCCLTVS.CXXIII	614.400	726 H/s	14 Min. 36 Sek.
Electrum	ITALIA.MMXX	3.932.160	1.575 H/s	43 Min. 2 Sek.
Putty	AVGVSTVS.MCMLXXV	-	-	-

Quelle: eigene Darstellung.

Alle vier Passwörter entspringen der Basis-Muster-Klasse $U^+S^+D^+$. Der Benutzer hält demnach an seinen bisherigen Konstruktionsprinzipien fest und verwendet die bereits genutzte Passwortstruktur erneut. Die Muster-Klasse besitzt nach Definition 5.1 eine Erfolgswahrscheinlichkeit in Höhe ihrer prozentualen Auftrittshäufigkeit von $66,67\%$ und ist damit die Muster-Klasse mit der höchsten Erfolgswahrscheinlichkeit. Sie befindet sich auf dem 1. Rang der nachzubildenden Muster-Klassen und wird im Generierungsprozess als erstes herangezogen.
Bei den alphabetischen Fragmenten handelt es sich bei allen Passwörtern analog zu den Basis-Passwörtern um lateinische Begriffe, die Teil der alphabetischen Wortliste des Präprozessors sind. Besonders interessant an den rekonstruierten Passwörtern sind die numerischen Fragmente, die im Folgenden näher betrachtet werden.

1. Passwort: PDF-Hash

Die Rekonstruktion des Passworts „TESTIMONIVM.LXXV“ des PDF-Hashwerts benötigt bei der erzielten Hashrate von rund 8.200 kH/s 1 Minute und 16 Sekunden. Bei dem Passwort handelt es sich um den $122.880.$ getesteten Passwort-Kandidaten.
Das numerische Fragment „LXXV“ entspricht dem Geburtsjahr „75“ des Benutzers und ist Bestandteil eines der Basis-Passwörter. Es handelt sich somit um ein wiederverwendetes Fragment und besitzt mit 50% die höchste Erfolgswahrscheinlichkeit innerhalb der numerischen Wortliste der Basis-Muster-Klasse $U^+S^+D^+$ (siehe Definition 5.2).

[38] Hierbei handelt es sich um die Hashrate zum Zeitpunkt der erfolgreichen Rekonstruktion. Die Hashrate kann während des Angriffs stark variieren.

Demzufolge ergibt sich nach Definition 5.6 eine im Vergleich mit den anderen Passwörtern höhere Erfolgswahrscheinlichkeit von:

$$\begin{aligned} P(TESTIMONIVM.LXXV) &= P(\beta_{TESTIMONIVM.LXXV}) \cdot \prod_{i=1}^{3} P(\alpha_{l_i}) \\ &= P(U^+S^+D^+) \cdot P(\alpha_{TESTIMONIVM}) \cdot P(\alpha_.) \cdot P(\alpha_{LXXV}) \\ &= 0,6667 \cdot 0.4545 \cdot 1,0 \cdot 0,5 \\ &= 0,1515 \\ &= 15,15\% \end{aligned}$$

Das alphabetische Fragment „TESTIMONIVM“ besitzt dieselbe Länge wie das Basis-Fragment „PANTOPOLIVM“, sodass es von den neuen, alphabetischen Fragmenten am höchsten priorisiert ist.

2. Passwort: Office-Hash

Die Rekonstruktion des Passworts „OCCLTVS.CXXIII“ des Office-Hashwerts benötigt bei der erzielten Hashrate von 726 H/s 14 Min. und 36 Sek.. Bei dem Passwort handelt es sich um den $614.400.$ getesteten Passwort-Kandidaten.
Das numerische Fragment „CXXIII“ entspricht der Ziffernfolge „123“ und stammt vom Basis-Fragment „1234“ ab, welches der Benutzer in römischer Darstellung „MCCXXIV“ bereits in seinen Basis-Passwörtern verwendet. Im Rahmen des Generierungsprozesses der numerischen Wortliste wird das Fragment „1234“ in seine einzelnen Bestandteile zerlegt, wodurch das gesuchte Fragment „123“ generiert und in die Wortliste aufgenommen wird.

Die Erfolgswahrscheinlichkeit des Passwort ist nach Definition 5.6 gegeben durch:

$$\begin{aligned} P(OCCLTVS.CXXIII &= P(\beta_{OCCLTVS.CXXII}) \cdot \prod_{i=1}^{3} P(\alpha_{l_i}) \\ &= P(U^+S^+D^+) \cdot P(\alpha_{OCCLTVS}) \cdot P(\alpha_.) \cdot P(\alpha_{CXXIII}) \\ &= 0,6667 \cdot 0,4773 \cdot 1,0 \cdot 0,125 \\ &= 0,0644 \\ &= 6,44\% \end{aligned}$$

3. Passwort: Electrum-Hash

Auf Basis der in der Vorbereitungsphase generierten Datengrundlage des Präprozessors kann das gesuchte Passwort „ITALIA.MMXX“ zunächst nicht rekonstruiert werden. Die Ursache liegt im numerischen Fragment „MMXX“, das dem Jahr „2020“ entspricht. Hierbei handelt es sich weder um ein wiederverwendetes Fragment noch um ein Element seiner persönlichen Informationen, wodurch es in Folge zunächst kein Bestandteil

der numerischen Wortliste ist.
Wie in Abschnitt 5.6.3.4 beschrieben, werden nach einem erfolglosen Passwortangriff Ausweitungsmaßnahmen zur Erweiterung des Kandidatenraums durchgeführt. Eine der Maßnahmen ist die Erweiterung der numerischen Wortliste durch den Einsatz des „Maskprocessors“. Er generiert alle Zahlen, die dieselbe Länge wie die numerischen Basis-Fragmente des Benutzers besitzen. Im vorliegenden Szenario entspricht dies den Längen 2 und 4, wodurch unter anderem die Zahl „2020“ generiert wird. Nach Umwandlung der neu generierten Fragmente in die römische Schreibweise entstehen 239 neue Fragmente, die in die numerischen Wortliste aufgenommen werden.[39]
Die erweiterte, numerische Wortliste besteht im Ergebnis aus 268 Einträgen, wodurch der Kandidatenraum auf $337.023.610.999$ Passwörter steigt.

Theoretisch betrachtet müsste für die erfolgreiche Rekonstruktion des Passworts zunächst der vollständige Kandidatenraum des ersten Angriffs getestet werden. Bei einem Raum von $35.981.179.642$ Kandidaten und einer erzielten Hashrate von 1.575 H/s, beträgt die zu erwartende Rechenzeit im Online-Modus rund 264 Tage. Erst nach erfolglosem Testen aller Kandidaten würde der erweiterte Angriff durchgeführt werden.

Bei der praktischen Durchführung wurde der erste Angriff nach einigen Tagen abgebrochen und die Datengrundlage des Angriffs frühzeitig erweitert. Auf Basis der neuen Datengrundlage konnte das Passwort innerhalb von 43 Minuten und 2 Sekunden rekonstruiert werden (siehe Tabelle 6.8). Im erweiterten Kandidatenraum handelt es sich bei dem Passwort „ITALIA.MMXX“ um den $3.293.160.$ getesteten Kandidaten.

Die Erfolgswahrscheinlichkeit ist nach Definition 5.6 gegeben durch:

$$\begin{aligned} P(ITALIA.MMXX) &= P(\beta_{ITALIA.MMXX}) \cdot \prod_{i=1}^{3} P(\alpha_{l_i}) \\ &= P(U^+S^+D^+) \cdot P(\alpha_{ITALIA}) \cdot P(\alpha_{.}) \cdot P(\alpha_{MMXX}) \\ &= 0,6667 \cdot 0,4091 \cdot 1,0 \cdot 0,4688 \\ &= 0,1278 \\ &= 12,78\% \end{aligned}$$

4. Passwort: Putty-Hash

Wie bereits erwähnt, bietet Hashcat keine Unterstützung für den Hashwert des Putty-Dienstes an, weshalb eine theoretische Betrachtung der Rekonstruktion folgt.
Bei dem alphabetischen Fragment „AVGVSTVS“ handelt es sich um ein wiederverwendetes Fragment des Benutzers, wodurch es nach Definition 5.2 mit 50% die höchste Erfolgswahrscheinlichkeit innerhalb der alphabetischen Wortliste besitzt.

[39] Römische Zahlen sind ohne die Verwendung von mathematischen Zeichen bis einschließlich 9999 darstellbar.

Das numerische Fragment „MCMLXXV“ entspricht dem vierstelligen Geburtsjahr „1975“ des Benutzers. Durch Aufnahme des Geburtsdatums in allen möglichen Darstellungsvarianten in die numerische Wortliste (siehe Abschnitt 5.5.3.2) ist das gesuchte Fragment mit einer Erfolgswahrscheinlichkeit von $28,125\%$ ein Eintrag der Liste.
Demzufolge handelt es sich bei dem gesuchten Passwort um den 24. Passwort-Kandidaten.

Die Erfolgswahrscheinlichkeit des Passworts beträgt nach Definition 5.6:

$$\begin{aligned} P(AVGVSTVS.MCMLXXV) &= P(\beta_{AVGVSTVS.MCMLXXV}) \cdot \prod_{i=1}^{3} P(\alpha_{l_i}) \\ &= P(U^+S^+D^+) \cdot P(\alpha_{AVGVSTVS}) \cdot P(\alpha_{.}) \cdot P(\alpha_{MCMLXXV}) \\ &= 0,6667 \cdot 0,5 \cdot 1,0 \cdot 0,28125 \\ &= 0,0937 \\ &= 9,37\% \end{aligned}$$

6.3 Experiment 3

Die Datengrundlage des dritten Experiments umfasst eine verschlüsselte Sqlite-Datenbank des Apple-Notizblocks, eine verschlüsselte Yaml-Datei sowie einen verschlüsselten Rar-Container. Zudem liegen die in Abbildung 6.7 abgebildeten, persönlichen Informationen über den Benutzer vor. Das vollständige Profil befindet sich in Anhang D.

Tabelle 6.7: Experiment 3 - Persönliche Informationen des Benutzers

Benutzer	Natasha Rybak (ehemalige Wosz)		
Persönliche Informationen	Geburtstsdatum:	04.01.1983	
	Geburtsort:	Kiev, Ukraine	
	Adresse:	Smolenki reki naberezhnaya 27	
	Nationalität:	russisch	
	Beruf:	Designerin	
	Beziehungsstatus:	verheiratet	
Interessen	Professionelle Schachspielerin, Ballet, Kickboxing		
Zugangsdaten	Twitter-Profil	Benutzernamen:	Nat.Ryb
		Passwort:	Octopuse6
	Facebook-Profil	Benutzername:	Natasha R.
		Passwort:	GreekGiftBxh7+
	Google-Profil	Benutzername:	NatashaRybak
		Passwort:	Gambitexf4
Zusatzinformation	Persönliche Informationen zu Ihrem Ehemann		

Quelle: eigene Darstellung.

6.3.1 Initialisierungsphase

Die Yaml-Datei ist über die Ansible-Anwendung[40] verschlüsselt und kann Hashcat im Angriff im Datei-Format übergeben werden. Folglich ist die Extraktion des Hashwerts an dieser Stelle entbehrlich. Bei den zwei weiteren Verschlüsselungsgegenständen werden die Extraktionsalgorithmen „applenotes2john.py“ und „rar2john.py“eingesetzt [9].[41]
Bei Betrachtung der Basis-Passwörter des Benutzers (siehe Tabelle 6.7), wird deutlich, dass der Benutzer Schachzüge in algebraischer Schreibweise („e6“, „BXh7+“ und „exf4“) an seine Passwörter anhängt. Die Fragmentierung der Passwörter bei der Eingabe in das Template findet unter Berücksichtigung des identifizierten Konstruktionsprinzips statt. Demnach wird das Basis-Passwort „Gambitexf4“ beispielsweise in die zwei Fragmente „Gambit“ und „exf4“ zerlegt und in das Template eingetragen.
Bei einer automatisierten Fragmentierung würde das Passwort in die Fragmente „G“, „ambitexf“ und „4“ zerlegt werden. Hieran wird der in Abschnitt 5.4 beschriebene Vorteil gegenüber einer automatisierten Fragmentierung deutlich.

6.3.2 Vorbereitungsphase

Nach der Analyse der Basis-Passwörter sowie der Anwendung des Muster-Generators für eine Runde liegen 22 Muster-Klassen vor, bestehend aus 2 Basis-Muster-Klassen und 20 zusätzlich generierten Klassen.
Bei der Generierung der alphabetischen Wortliste wird als vordefinierte Liste eine Sammlung [10] bekannter Schachbegriffe herangezogen. Im Ergebnis besteht die Wortliste aus 1.018 Einträgen.
Die alphanumerischen Fragmente der Basis-Passwörter wie beispielsweise das Fragment „exf4“ werden im Rahmen des Präprozessors der Wortliste der Sonderzeichen zugeordnet. Demnach verwendet der Benutzer keine numerischen Fragmente in den Basis-Passwörter, sodass die numerische Wortliste neben den persönlichen Informationen des Benutzers um die 20 beliebtesten Erweiterungsfragmente ergänzt wird (siehe Abschnitt 5.5.3.2). Im Ergebnis besitzt sie 39 Einträge.
Die Wortliste der Sonderzeichen wird um eine selbst generierte Liste möglicher Schachzüge erweitert. Grundsätzlich besitzen Schachzüge die folgende algebraische Notation:

<K,Q,R,B,N>(<x>)<Zielfeld>(<+,#,++>)

wobei das erste Element die verwendete Schachfigur des Spielzugs angibt und das zweite Element „x“, ob es sich um ein Schlagzug handelt. Das Zielfeld entspricht der

[40] Hierbei handelt sich um eine Automatisierungs-Anwendung zur System-Administration und -Konfiguration [36].
[41] Der Ansible-Hash besitzt den Hashcat-internen Hashmodus 16900, der Securenotes-Hash den Hashmodus 16200 und der Rar-Hash den Hashmodus 13000.

Position der Schachfigur nach dem Spielzug und die Elemente „+,#,++“ stehen für besondere Schachstellungen wie beispielsweise das Matt.
Im Rahmen des Angriffsprozesses findet das optionale Anfügen eines der Elemente „+“, „#“ und „++“ über ein Regelwerk statt, um die Wortliste der Sonderzeichen nicht um ein Vierfaches zu vergrößern. Im Ergebnis besteht die Liste aus 1.238 Fragmenten.

6.3.3 Angriffsphase

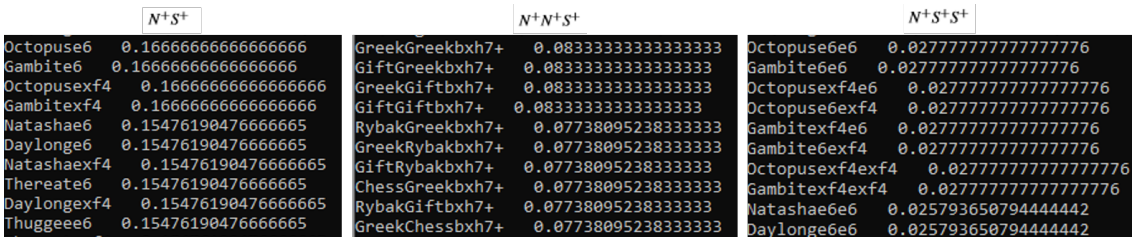
Die ersten beiden Angriffsprozesse bilden erneut die Bruteforce- und die Wörterbuch-Attacke. Analog zu den ersten beiden Experimenten verwendet der Benutzer weder ein besonders kurzes Passwort noch ein Standard-Passwort. Demzufolge wird an dieser Stelle auf eine Darstellung der Angriffsdurchführung verzichtet und die Beschreibung mit der benutzerspezifischen Attacke fortgesetzt.

Der zu testende Kandidatenraum des benutzerspezifischen Angriffs besteht aus $8.792.844.969.842$ Passwörtern und ist nach Definitionen 5.7 gegeben durch:[42]

$$\begin{aligned} Kandidatenraum(S) &= \sum_{\beta \in S} Kandidatenraum(\beta) \\ &= 8.792.844.969.842 \end{aligned}$$

Die Abbildung 6.6 zeigt einen Ausschnitt der Top-3 Muster-Klassen mit den jeweils 10 Passwort-Kandidaten höchster Erfolgswahrscheinlichkeit.

Abbildung 6.6: Experiment 3 - Ausgabe der Kandidaten höchster Erfolgswahrscheinlichkeit



Quelle: python-Implementierung des Präprozessors.

[42] Mit Einbezug des Regelwerks besteht der Kandidatenraum aus $4 \cdot 8.792.844.969.842 = 35.171.379.879.368$ Passwörtern.

Im Ergebnis können alle drei Passwörter rekonstruiert werden. Die jeweiligen Ergebnisse des Angriffs mit dem zu testenden Kandidatenraum, der erzielten Hashrate sowie der Rechenzeit sind in Tabelle 6.8 abgebildet.

Tabelle 6.8: Experiment 3 - Rekonstruktions-Ergebnisse im Online-Modus[43]

Hashtyp	Gesuchtes Passwort	Anzahl getesteter Kandidaten	Hashrate	Rechenzeit
Ansible	QueeningQd4+	2.826.240	49.361 H/s	1 Min.
SecureNotes	IntermezzoRxa7+	8.785.920	37.951 H/s	3 Min. 50 Sek.
Rar	AlekhinesGunQc1	184.712.096	600 H/s	3 T. 14 Std.

Quelle: eigene Darstellung.

Alle drei Passwörter entspringen einer der Basis-Muster-Klasse N^+S^+ oder $N^+N^+S^+$. Der Benutzer hält demnach an seinen bisherigen Konstruktinsprinzipien fest und verwendet eine bereits genutzte Passwortstruktur erneut. Die Muster-Klassen besitzen nach Definition 5.1 eine Erfolgswahrscheinlichkeit in Höhe ihrer prozentualen Auftrittshäufigkeit von $66,67\%$ bzw. $33,33\%$. Sie befinden sich auf dem $1.$ und $2.$ Rang der nachzubildenden Muster-Klassen und werden im Generierungsprozess als erstes bzw. als zweites herangezogen.

Bei der Wahl der Fragmente folgt der Benutzer ebenfalls seinem bisherigen Muster. Bei den alphabetischen Fragmenten handelt es sich bei allen Passwörtern um Begriffe aus der Schachszene. Zudem stellen die numerischen Fragmente einen Schachzug der zuvor beschriebenen Notation dar. Alle Passwort-Elemente sind Bestandteil der Wortliste der entsprechenden Fragment-Klasse.

1. Passwort: Ansible-Hash

Die Rekonstruktion des Passworts „QueeningQd4+“ des Ansible-Hashwerts benötigt bei der erzielten Hashrate von 49.361 H/s 1 Minute. Bei dem Passwort handelt es sich um den $2.826.240.$ getesteten Kandidaten.
Die Erfolgswahrscheinlichkeit des Passworts ist nach Definition 5.6 gegeben durch:

$$\begin{aligned} P(QueeningQd4+) &= P(\beta_{QueeningQd4+}) \cdot \prod_{i=1}^{2} P(\alpha_{l_i}) \\ &= P(N^+S^+) \cdot P(\alpha_{Queening}) \cdot P(\alpha_{Qd4}) \\ &= 0,6667 \cdot 0,3929 \cdot 0,35 \\ &= 0,0917 \\ &= 9,17\% \end{aligned}$$

[43] Hierbei handelt sich um die Hashrate zum Zeitpunkt der erfolgreichen Rekonstruktion. Die Hashrate kann während des Angriffs stark variieren.

2. Passwort: Securenote-Hash

Die Rekonstruktion des Passworts „IntermezzoRxa7+“ des Securenote-Hashwerts benötigt bei der erzielten Hashrate von 37.951 H/s 3 Minuten und 50 Sekunden. Bei dem Passwort handelt es sich um den $8.785.920.$ getesteten Kandidaten.
Die Erfolgswahrscheinlichkeit des Passworts ist nach Definition 5.6 gegeben durch:

$$\begin{aligned} P(IntermezzoRxa7+) &= P(\beta_{IntermezzoRxa7+}) \cdot \prod_{i=1}^{2} P(\alpha_{l_i}) \\ &= P(N^+S^+) \cdot P(\alpha_{Intermezzo}) \cdot P(\alpha_{Rxa7}) \\ &= 0,6667 \cdot 0,25 \cdot 0,4 \\ &= 0,0667 \\ &= 6,67\% \end{aligned}$$

3. Passwort: Rar-Hash

Die Rekonstruktion des Passworts „AlekhinesGunQc1“ des Rar-Hashwerts benötigt bei der erzielten Hashrate von 600 H/s 3 Tage und 14 Stunden. Bei dem Passwort handelt es sich um den $184.712.096.$ getesteten Kandidaten.
Die Erfolgswahrscheinlichkeit des Passworts ist nach Definition 5.6 gegeben durch:

$$\begin{aligned} P(AlekhinesGunQc1) &= P(\beta_{AlekhinesGunQc1}) \cdot \prod_{i=1}^{3} P(\alpha_{l_i}) \\ &= P(N^+N^+S^+) \cdot P(\alpha_{Alekhines}) \cdot P(\alpha_{Gun}) \cdot P(\alpha_{Qc1}) \\ &= 0,3333 \cdot 0,1786 \cdot 0,3571 \cdot 0,5 \\ &= 0,0106 \\ &= 1,06\% \end{aligned}$$

7 Evaluation

Die Schwierigkeit der benutzerspezifischen Passwort-Rekonstruktion liegt zum einen in der Generierung der Muster-Klasse des gesuchten Passworts und zum anderen in der Berücksichtigung der vom Benutzer verwendeten Fragmente in der entsprechenden Wortliste.
Für die Generierung der wahrscheinlichsten Passwortstrukturen ist der Muster-Generator des Präprozessors zuständig. Grundsätzlich ist sein Ziel, auf Basis der Struktur der bekannten Vergleichspasswörter eines Benutzers die Struktur des gesuchten Passworts zu generieren. Zur Bewertung der Rekonstruktionsleistung des Generators werden erneut die Passwörter des Leaks herangezogen.

7.1 Effizienz des Muster-Generators

Ziel dieses Abschnitts ist die Ermittlung der Anzahl von Passwortstrukturen eines Benutzers, die der Generator auf Basis eines bzw. von zwei gegebenen Passwörtern.
Hierfür wird jedes Passwortset der Benutzer in einen Trainings- und Testdatensatz geteilt. Der Trainingsdatensatz besteht aus einem bzw. zwei zufällig ausgewählten Muster-Klassen und bildet die Grundlage der Generierung neuer Muster-Klassen. Die Auswahlhöhe an Muster-Klassen des Trainingssets ist hierbei von der Gesamtzahl an vorliegenden Passwörtern abhängig. Liegen 2 bzw. 3 Passwörter eines Benutzers vor besteht der Trainingsdatensatz aus einer Muster-Klasse. Andernfalls werden 2 Muster-Klassen herangezogen.
Im Anschluss werden über die Anwendung der in Abschnitt 5.5.2.1 definierten Modifikationsprinzipien neue Muster-Klassen auf Basis des Trainingssets generiert. Die resultierenden Muster-Klassen werden in Folge mit den Muster-Klassen des Testsets des Benutzers verglichen und die Anzahl an Übereinstimmungen ermittelt.

Die Datengrundlage bildet die Passwortsammlung 4. Zusätzlich werden alle Muster-Klassen-Duplikate im Passwortset eines Benutzers entfernt, damit ausschließlich modifizierte Muster-Klassen als eine Übereinstimmung zählen. Zudem müssen pro Benutzer mindestens 2 unterschiedliche Muster-Klassen vorliegen.
Im Ergebnis besteht der Datensatz aus $5.875.568$ Passwörtern von $2.476.036$ verschiedenen Benutzern. Nach erfolgter Trennung beinhaltet der Trainingsdatensatz insgesamt $2.644.886$ Muster-Klassen und der Testdatensatz $3.230.682$ Muster-Klassen.

Definition 7.1 *Sei T_{B_i} die Anzahl an Übereinstimmungen zwischen den generierten Muster-Klassen $\beta \in S$ der Strategie S und den Muster-Klassen des Testdatensatzes des Benutzers B_i. Weiterhin sei β_{Test} die Gesamtzahl an Muster-Klassen des Testsets. Demzufolge ist ein Maß für die Effizienz des Muster-Generators gegeben durch:*

$$P(T) = \frac{\sum_{i=1}^{n} T_{B_i}}{\beta_{Test}} \tag{7.1}$$

wobei $\beta_{Test} = 3.230.682$ und $n = 2.476.036$ der Anzahl an Benutzern entspricht.

In Tabelle 7.1 sind die Ergebnisse der Messung der Effizienz des Muster-Generators mit der Anzahl an generierten Muster-Klassen ($|\beta|$) und der Anzahl an rekonstruierten Muster-Klassen ($\sum_{i=1}^{n} T_{B_i}$) abgebildet. Die Mustergenerierung erfolgt für eine sowie für zwei Runden.

Tabelle 7.1: Ergebnisse der Effizienzmessung des Muster-Generators

Prinzip	1 Runde			2 Runde		
	$\lvert\beta\rvert$	$\sum_{i=1}^{n} T_{B_i}$	$P(T)$ in %	$\lvert\beta\rvert$	$\sum_{i=1}^{n} T_{B_i}$	$P(T)$ in %
1.	2.126.932	34.043	1,05	2.127.375	33.853	1,05
2./ 3./ 4.	4.559.871	2.313	0,07	11.177.331	2.380	0,07
5./ 6.	9.377.866	1.133.307	35,08	27.229.430	1.281.074	39,65
7.	597.218	141.202	4,37	799.837	144.891	4,48
8.	1.874.099	65.189	2,02	2.147.099	65.673	2,03
Σ	18.535.986	1.376.054	**42,59**	40.744.700	1.527.871	**47,29**

Quelle: eigene Darstellung.

Von den insgesamt $3.230.682$ gesuchten Muster-Klassen (Testset) werden bei Durchführung einer Generierungsrunde $1.376.054$ Muster-Klassen rekonstruiert. Dies entspricht einer Erfolgsrate $P(T)$ von $42,59\%$. Hierfür werden insgesamt $18.535.986$ Muster-Klassen auf Basis des Trainingsset generiert. Demzufolge entsprechen $7,42\%$ der generierten Muster-Klassen einer von den Benutzern verwendeten Passwortstruktur. Pro Benutzer entspricht dies rund 7 nachzubildenden Muster-Klassen.
Bei Durchführung einer weiteren Generierungsrunde ist der Generator in der Lage $1.527.871$ Muster-Klassen zu rekonstruieren. Dadurch steigt die Erfolgsrate $P(T)$ um rund 5 Prozentpunkte auf $47,29\%$. Die Gesamtzahl an generierten Muster-Klassen hat sich mit $40.774.700$ Klassen mehr als verdoppelt. Demzufolge entsprechen $3,75\%$ der generierten Muster-Klassen einer von den Benutzern verwendeten Passwortstruktur. Pro Benutzer entspricht dies rund 16 nachzubildenden Muster-Klassen.
Bei Durchführung einer dritten Runde erzielt der Generator keine weitere, nennenswerte Verbesserung.

Im Ergebnis ist der Generator somit in der Lage ca. 47% der Muster-Klassen der Benutzer zu rekonstruieren. Zur Erzielung dieser Erfolgsrate, werden im Angriffsprozess jedoch $96,25\%$ der generierten Muster-Klassen ohne Erfolg nachgebildet, da lediglich $3,75\%$ aller generierten Muster-Klassen eine tatsächlich verwendete Passwortstruktur sind.
Zur Minimierung der Anzahl an nachzubildenden Muster-Klassen kann der Generierungsumfang begrenzt werden. Bei der Wahl der anzuwendenden Modikfikationsprinzipien könnte der Fokus noch stärker auf die beliebtesten Passwort-Modifikationen von Benutzern gelegt werden. Demnach könnte beispielsweise auf die Modifikation des Duplizierens einzelner Muster-Klassen-Elemente ($2./3./4.$ Prinzip in Tabelle 7.1) verzichtet werden, da es mit rund $4,5$ Mio. bzw. 11 Mio. Muster-Klassen eine hohe Anzahl zusätzlicher Klassen mit einer vergleichbar geringen Erfolgsrate von $0,07\%$ generiert.

Generiert der Muster-Generator im Anwendungsfall die gesuchte Muster-Klasse und beinhalten die Wortlisten die gesuchten Passwort-Fragmente, sind die Voraussetzungen zur Erzeugung des gesuchten Passwort-Kandidaten erfüllt. Der tatsächliche Rekonstruktionserfolg im nachfolgenden Angriff bestimmt sich durch den Stellenwert des gesuchten Passworts im Kandidatenraum und damit durch die Rechenzeit bis zum Testen des gesuchten Kandidaten.

Im Folgenden werden die Ergebnisse der benutzerspezifischen Attacke nochmals zusammengefasst und mit den zu erwartenden Rechenzeiten über einen Bruteforce-Angriff als alternative Angriffsmethode verglichen. Des Weiteren werden die in Kapitel 3 vorgestellten Präprozessoren herangezogen und auf das $1.$ Experiment angewendet. Die jeweils erzielten Ergebnisse werden anschließend den Ergebnissen des Präprozessors dieser Arbeit gegenübergestellt.

7.2 Ergebnisse des benutzerspezifischen Angriffs

Wie bereits im Rahmen des ersten Experiments erläutert, kann die Hashgeschwindigkeit im Angriff über einen Wechsel vom Online- in den Offline-Modus erhöht werden. Demzufolge werden im Folgenden analog zum ersten Experiment die Rekonstruktions-Ergebnisse auf theoretischer Ebene bei Durchführung des Angriffs im Offline-Modus betrachtet.

Als Hashrate wird jeweils die im Rahmen des Wörterbuch-Angriffs erzielte Geschwindigkeit herangezogen. Tabelle 7.2 zeigt die Ergebnisse der Rekonstruktion der Passwörter im Offline-Modus für die Experimente 1 bis 3.

Tabelle 7.2: Übersicht der Rekonstruktions-Ergebnisse im Offline-Modus[44]

Kandidatenraum	Passwort	Getestete Kandidaten	Hashrate	Rechenzeit	Erfolgswkt. (in %)
253.067.105.648.000	TiltedTowers1337	127.000.012.514.240	250 MH/s	5 T. 21 Std.	5,21
	BushCamper1337	127.000.028.125.460	220 MH/s	6 T. 16 Std.	3,32
	stormflip1337	64.000.006.553.115	900 H/s	2.255 T.	14,93
35.981.179.642	TESTIMONIVM.LXXV	112.880	10 MH/s	< 1 Sek.	15,15
	OCCLTVS.CXXIII	614.400	7.3 kH/s	1 Min.	6,44
	ITALIA.MMXX	3.932.160	115 kH/s	35 Sek.	12,78
	AVGVSTVS.MCMLXXV	-	-	-	9,37
8.792.844.969.842	QueeningQd4+	2.826.240	95 kH/s	29 Sek.	9,17
	IntermezzoRxa7+	8.785.920	47 kH/s	3 Min.	6,67
	AlekhinesGunQc1	184.712.096	29 kH/s	1 Std. 46 Min.	1,06

Quelle: eigene Darstellung.

7.2.1 Experiment 1

Bei dem ersten Experiment sind die zu erwartenden Rekonstruktionszeiten der ersten beiden Passwörter trotz vergleichbar hoher Hashraten von 250 MH/s bzw. 220 MH/s mit rund 6 und 7 Tagen deutlich höher als alle anderen Rekonstruktionszeiten. Ursächlich hierfür ist der große Kandidatenraum mit rund 253 Bio. Kandidaten, der sich aus der hohen Anzahl an Elementen pro nachzubildender Muster-Klasse ergibt.
Bis zu einer erfolgreichen Rekonstruktion der gesuchten Passwörter müssen rund 127 Bio. Hashwerte berechnet und erfolglos mit dem Ziel-Hashwert verglichen werden. Wie bereits in Abschnitt 6.1.3.3 erwähnt, liegt der Grund für die hohe Anzahl an zu testenden Kandidaten in der abweichenden Struktur der gesuchten Passwörter.
Der Zusammenhang zwischen der Erfolgswahrscheinlichkeit eines Kandidaten und der Rekonstruktionszeit lässt sich hieran verdeutlichen. Je höher die Erfolgswahrscheinlichkeit eines Passwort-Kandidaten ist, desto höher ist sein Stellenwert im Kandidatenraum und desto früher findet er folglich im Angriffsprozess Berücksichtigung. Dies impliziert eine geringere Anzahl von bereits getesteten Kandidaten, wodurch die Rechenzeit der Rekonstruktion sinkt.
Bei Betrachtung des Passworts „stormflip1337“ ist die hohe Berechnungsdauer mit 2.255 Tagen auffällig. Die Ursache liegt in der langsamen Hashfunktion der Bitlocker-Verschlüsselung mit einer Hashrate von lediglich rund 900 H/s. Die im Vergleich zu den anderen zwei Passwörtern hohe Erfolgswahrscheinlichkeit ist auf wahrscheinlichere Muster-Klasse, die geringe Längendifferenz des alphabetischen Fragments sowie die geringere Anzahl an Fragmenten zurückzuführen (siehe Abschnitt 6.1.3.3).

[44] Bei der angenommmenen Hashrate sowie der Rechenzeit handelt es sich um Rundungswerte.

7.2.2 Experiment 2

Im zweiten Experiment sind die zu erwartenden Rechenzeiten der Rekonstruktion von maximal 1 Minute sehr viel kürzer als im Rahmen des ersten Experiments. Grund hierfür ist zum einen der kleinere Kandidatenraum mit rund 36 Mrd. Passwörtern. Dieser resultiert aus den kürzeren, nachzubildenden Muster-Klassen. Während im ersten Experiment bereits eines der Basis-Passwörter aus 5 Fragmenten besteht, wodurch folglich auch die neuen Muster-Klassen eine größere Länge aufweisen, bestehen die Basis-Passwörter in diesem Experiment aus jeweils 3 Fragmenten.
Zum anderen ist die jeweilige Anzahl an zu testenden Kandidaten deutlich geringer, da der Benutzer, wie bereits in Abschnitt 6.2.3 erläutert, an seinen bisherigen Konstruktionsprinzipien festhält. Die gesuchte Muster-Klasse $U^+S^+D^+$ besitzt folglich die höchste Erfolgswahrscheinlichkeit und wird im Angriffsprozess als erste Muster-Klasse zur Nachbildung herangezogen.
Die Differenz in den Erfolgswahrscheinlichkeiten der Passwörter resultiert zum einen aus der Intergration von wiederverwendeten Passwort-Elementen wie der Fragmente „LXXV“ und „AVGVSTV“, welche die höchste Erfolgswahrscheinlichkeit innerhalb ihrer Fragment-Klasse aufweisen. Zum anderen ergibt sich der Unterschied aus den Längen der verwendeten alphabetischen und numerischen Fragmente. Folglich besitzt, wie in Abschnitt 6.2.3 aufgezeigt, das Passwort „TESTIMONIVM.LXXV“ mit $15,15\%$ die höchste Erfolgswahrscheinlichkeit.

7.2.3 Experiment 3

Analog zum zweiten Experiment werden im dritten Experiment sehr effiziente, zu erwartende Rekonstruktionszeiten von maximal rund einer Stunde und 46 Minuten erzielt. Grund hierfür ist erneut die Tatsache, dass der Benutzer seine bisherigen Passwortstrukturen wiederverwendet und die gesuchte Muster-Klasse folglich im Angriffsprozess als erstes nachgebildet wird.
Der Zusammenhang zwischen der Erfolgswahrscheinlichkeit eines Kandidaten sowie der zu erwartenden Rekonstruktionszeit ist zudem erneut deutlich zu beobachten. Das Passwort „QueeningQd4+“ beispielsweise wird mit einer Wahrscheinlichkeit von $9,17\%$ innerhalb von 29 Sekunden rekonstruiert, während das Passwort „IntermezzoRxa7+“ derselben Muster-Klasse mit einer Erfolgswarhschienlichkeit von $6,67\%$ eine Rekonstruktionszeit von 3 Minuten aufweist.
Das Passwort „AlekhinesGunQc1“ hat aufgrund seiner Zusammensetzung aus 3 Fragmenten sowie aufgrund der großen Längendifferenz des alphabetischen Fragments „Alekhines“ zum Basis-Fragment „Greek“ in Höhe von 4 Zeichen, die geringste Erfolgswahrscheinlichkeit.

7.3 Alternative Angriffsmethode

Eine gängige Angriffsmethode zur Rekonstruktion von Passwörtern ist der in Abschnitt 2.5.1 vorgestellte Bruteforce-Angriff bzw. Mask-Angriff bei der zusätzlichen Annahme der Kenntnis des zugrundeliegenden Zeichensatzes des gesuchten Passworts.
Die Tabelle 7.3 zeigt die zu erwartenden, maximalen Berechnungszeiten für die Rekonstruktion der gesuchten Passwörter der Experimente 1 bis 3 über die Bruteforce- sowie über die Mask-Methode. Die hierbei getroffene Annahme ist die Kenntnis der Länge des gesuchten Passworts. Der Kandidatenraum wird somit jeweils auf Passwörter der angenommene Länge begrenzt.

Tabelle 7.3: Zu erwartende Rekonstruktionszeiten über Bruteforce-/ Mask-Angriff[45]

		Bruteforce-Angriff		Mask-Angriff	
Passwort	Hashrate	Raum	Rechenzeit	Raum	Rechenzeit
TiltedTowers1337	10.000 MH/s	95^{16}	$1.4 \cdot 10^{15}$ J.	$26^{12} \cdot 10^4$	$3 \cdot 10^3$ J.
BushCamper1337	3.700 MH/s	95^{14}	$42 \cdot 10^9$ J.	$26^{10} \cdot 10^4$	12 J.
stormflip1337	1.100 H/s	95^{13}	$1.5 \cdot 10^{15}$ J.	$26^9 \cdot 10^4$	$1.6 \cdot 10^6$ J.
TESTIMONIVM.LXXV	11.800 kH/s	95^{16}	$1.2 \cdot 10^{18}$ J.	$26^{11} \cdot 33^1 \cdot 26^4$	$1.5 \cdot 10^9$ J.
OCCLTVS.CXXIII	7.300 H/s	95^{15}	$2 \cdot 10^{18}$ J.	$26^8 \cdot 33^1 \cdot 26^6$	$9 \cdot 10^9$ J.
ITALIA.MMXX	113 kH/s	95^{11}	$1.6 \cdot 10^9$ J.	$26^6 \cdot 33^1 \cdot 26^4$	$1.3 \cdot 10^3$ J.
AVGVSTVS.MCMLXXV	-	95^{16}	-	$26^8 \cdot 33^1 \cdot 26^7$	-
QueeningQd4+	96 kH/s	95^{12}	$1.8 \cdot 10^{12}$ J.	$26^{10} \cdot 10^1 \cdot 33^1$	$15 \cdot 10^3$ J.
IntermezzoRxa7+	48 kH/s	95^{15}	$3.1 \cdot 10^{18}$ J.	$26^{13} \cdot 10^1 \cdot 33^1$	$5.4 \cdot 10^9$ J.
AlekhinesGunQc1	29 kH/s	95^{15}	$5.1 \cdot 10^{18}$ J.	$26^{14} \cdot 10^1$	$7.1 \cdot 10^9$ J.

Quelle: eigene Darstellung.

Wie zu erkennen ist, liegt die Rekonstruktionszeit bei der zugrundeliegenden Rechenleistung bei der Bruteforce-Attacke mindestens im Bereich von Milliarden von Jahren. Über einen Mask-Angriff kann der Kandidatenraum zwar deutlich verkleinert werden. Jedoch liegen die maximalen Rechenzeiten (mit einer Ausnahme) noch immer mindestens im Bereich von Jahrtausenden. Eine erfolgreiche Rekonstruktion unter Berücksichtigung des Ziels eines effizienten Ressourceneinsatzes ist somit vollkommen ausgeschlossen.
Die einzige Ausnahme bildet das Passwort „BushCamper1337“ des 1. Experiments. Bei einer Hashrate von rund 3.700 MH/s ergibt sich eine maximale Berechnungsdauer von rund 12 Jahren. Im direkten Vergleich zu den Rechenzeiten der anderen Passwörter liegt es im Bereich einer möglichen Rekonstruktion. Aufgrund des großen zeitlichen Aufwands ist die Durchführung jedoch sehr ineffizient.

In beiden Angriffsszenarien sind die hohen Rechenzeiten den Längen der gesuchten Passwörter geschuldet, die jeweils zwischen 11 und 16 Zeichen besitzen. Wie bereits in Abschnitt 2.5.1 aufgezeigt, lässt sich die Bruteforce-Methode ausschließlich zur Rekonstruktion kurzer Passwörter effizient einsetzen. Hinzukommt die begrenzte Rechen-

[45] Die jeweilige Hashrate entspricht der erzielten Hashgeschwindigkeit bei Durchführung der Bruteforce-Attacke im Rahmen der Experimente $1-3$.

leistung der zugrundliegenden Hardware, bei der es sich lediglich um eine Grafikkarte handelt.

Der Vorteil der Bruteforce-Methode liegt bei vollständiger Durchführung in der Gewährleistung einer erfolgreichen Passwort-Rekonstruktion. Jede alternative Angriffsmethode begrenzt mit dem Ziel der Optimierung der Rechenzeit den Passwort-Kandidatenraum, wodurch der Rekonstruktionserfolg nicht sichergestellt ist. Hieran wird der Zielkonflikt zwischen der begrenzten Ressourcenverfügbarkeit und dem Rekonstruktionserfolg deutlich.

7.4 Alternative Präprozessoren

Während die Bruteforce-Methode alle Passwort-Kandidaten des gesamten Kandidantenraums generiert, zielen die Präprozessoren der in Kapitel 3 vorgestellten Modelle auf eine exklusive Generierung der Kandidaten höchster Erfolgswahrscheinlichkeit ab. Im Folgenden werden die einzelnen Präprozessoren herangezogen und ihre Anwendbarkeit auf das betrachtete Szenario dieser Arbeit bei Verwendung des erste Experiments untersucht. Die erzielten Ergebnisse werden in Folge mit den Ergebnissen der benutzerspezifischen Attacke verglichen.

7.4.1 Modell von Weir et. al.

Nach dem Modell von Weir et. al. besitzen die Basis-Passwörter des Benutzers („Fortnite1337“ und „gitgudkkthxbye69“) die Basisstrukturen L_8D_4 und $L_{14}D_2$ mit einer Auftrittswahrscheinlichkeit von jeweils 50%. Die numerischen Fragmente, „1337“ und „69“, die aus den Basis-Passwörtern extrahiert werden, besitzen innerhalb ihrer Längen-Kategorie jeweils eine Auftrittswahrscheinlichkeit von 100%. Für den Generierungsprozess ergeben sich hieraus die zwei vorläufigen Passwort-Kandidaten $L_8 1337$ und $L_{14} 69$ mit einer Erfolgswahrscheinlichkeit von jeweils 50%.

Im Rahmen der Kandidatengenerierung werden die vorläufigen Kandidaten mit alphabetischen Fragmenten gleicher Länge aus einem herangezogenen Wörterbücher befüllt. Als Wörterbuch wird die alphabetische Wortliste aus Experiment 1 mit 1.018 Einträgen zugrundegelegt. Sie beinhaltet 93 Fragmente der Länge 8 sowie 11 Einträge der Länge 14. Nach der Substitution der L-Elemente mit den entsprechenden Fragmenten ergeben sich 104 zu testende Passwort-Kandidaten. Die Erfolgswahrscheinlichkeit der Kandidaten entspricht nach Definition 3.1 der Wahrscheinlichkeit der vorläufigen Passwort-Kandidaten in Höhe von 50%. Den alphabetischen Fragmenten selbst ist nämlich keine Erfolgswahrscheinlichkeit zugewiesen.

Obwohl die gesuchten Passwort-Elemente Bestandteil des herangezogenen Wörterbuchs sind, wird keines der gesuchten Passwörter vom Präprozessor generiert. Grund hierfür ist zum einen die Begrenzung der Substitution auf Fragmente gleicher Länge. Daher wird das gesuchte Fragment „stormflip“, welches eine Länge von 9 Zeichen aufweist, nicht zur Befüllung des vorläufigen Kandidaten $L_8 1337$ herangezogen.
Zum anderen liegt der Grund in der exklusiven Nachbildung der im Trainingsset auftretenden Muster-Klassen. Keines der Basis-Passwörter beinhaltet eine Aneinanderreihung von zwei alphabetischen Fragmenten, sodass die Passwörter „TiltedTowers1337“ und „BushCamper1337“ nicht generiert werden.

Die soeben geschilderten Begrenzungen liegen im Modell dieser Arbeit nicht vor. Zum einen werden zur Befüllung der Muster-Klassen auch Fragmente mit einer unterschiedlichen Länge als die Basis-Fragmente herangezogen. Je weiter sich die Länge eines Fragments von der Basis-Länge entfernt, desto niedriger wird ihre Erfolgswahrscheinlichkeit bemessen und umso geringer ist daher die Erfolgswahrscheinlichkeit des zugehörigen Fragments (siehe Abschnitt 5.5.3.1). Diese Strategie wird unabhängig von der Fragment-Klasse verfolgt und findet somit ebenfalls bei den numerischen Fragmenten sowie bei den Sonderzeichen Anwendung.
Das Fragment „stormflip“ mit einer Längendifferenz von einem Zeichen (zu dem Basis-Fragment „Fortnite“) besitzt somit, nach den Fragmenten gleicher Länge, die größtmögliche Erfolgswahrscheinlichkeit innerhalb der alphabetischen Fragment-Klasse und findet somit frühzeitig Berücksichtigung bei Befüllung der Muster-Klasse.
Der zweiten Problematik, der Begrenzung der nachzubildenden Muster-Klassen, wird über die Definition der Modifikationsprinzipien begegnet. Die Modifikationsprinzipien werden auf die Basis-Strukturen eines Benutzers angewendet und generieren weitere, im Angriffsprozess zu befüllende Muster-Klassen. Folglich werden Kandidaten mit einer neuen Passwortstruktur generiert. Die Modifikationsprinzipien repräsentieren dabei die beliebtesten, von Benutzern vorgenommenen Modifikationen an Passwörtern.

Eine weitere in Abschnitt 3.1.1 beschriebene Begrenzung des Modells besteht in der Gleichbehhandlung der Groß- und Kleinschreibung bei der Erstellung der Passwort-Muster. Das Passwort „TiltedTowers1337“ besitzt demnach die Basisstruktur $L_{12}D_4$, wodurch im Generierungsprozess anstelle von zwei Wörtern der Länge 6 lediglich ein alphabetisches Fragment der Länge 12 eingesetzt wird. Die Aneinanderreihung von zwei Fragmenten der gleichen Fragment-Klasse ist demnach ausgeschlossen.
Zur detaillierten Sturkturbeschreibung der Basis-Passwörter werden im Rahmen dieser Arbeit, neben der Unterscheidung zwischen vollständiger Gr0ß- und Kleinschreibung von Fragmenten, zwei weitere Muster-Klassen-Elemente definiert. Demzufolge repräsentieren die Elemente N^+ und R^+ Wörter mit groß- bzw. kleingeschriebenem Anfangsbuchstaben. Bei der Befüllung der Muster-Klassen werden die Fragmente in die Schreibweise des entsprechenden Muster-Klassen-Elements konvertiert.

Zuletzt werden im Modell von Weir et. al. die zur Befüllung herangezogenen numerischen Fragmente und Sonderzeichen auf die im Trainingsset enthaltenen D- und S-Fragmente begrenzt.
Im Rahmen dieser Arbeit findet in beiden Fragment-Klassen eine Erweiterung um zusätzliche Fragmente statt. Wie in den Abschnitten 5.5.3.2-5.5.3.3 beschrieben, werden auf Grundlage der persönlichen Informationen eines Benutzers weitere Ziffernfolgen generiert. Zudem finden die 20 beliebtesten Ziffernfolgen sowie Sonderzeichen Berücksichtigung in den jeweiligen Wortlisten. Des Weiteren können über die beschriebenen Ausweitungsmaßnahmen (siehe Abschnitt 5.6.3.4) unter Verwendung des „Maskprocessors“ weitere Fragmente gewünschter Länge generiert werden. Die Berechnung der Erfolgswahrscheinlichkeit findet, wie bereits erwähnt, über die jeweilige Längendifferenz zum Basis-Fragment statt.

7.4.2 Modell von Chou et. al.

Nach dem Modell von Chou et. al. besitzen die Basis-Passwörter aufgrund der Unterscheidung zwischen Groß- und Kleinschreibung die Basisstrukturen $U_1L_7D_4$ und $L_{14}D_2$. Der Generierungsprozess von Kandidaten erfolgt analog zum Modell von Weir et. al. und führt somit aufgrund der Längen- sowie Muster-Klassen-Begrenzung zu demselben Ergebnis. Folglich wird keines der gesuchten Passwörter vom Präprozessor generiert.

7.4.3 Modell von Li et. al

Wie bereits in Abschnitt 3.2.1 dargestellt, folgt der Präprozessor von Li et. al. dem Modell von Weir et. al. Der einzige Unterschied liegt in der detaillierteren Beschreibung der Passwortstrukturen durch Abbildung von persönlichen Informationen in den nachzubildenden Muster-Klassen. Im ersten Experiment liegen keine persönlichen Informationen in den Vergleichspasswörtern vor, wodurch die Basisstrukturen analog zum Modell von Weir et. al. L_8D_4 und $L_{14}D_2$ lauten. Im Ergebnis wird somit keines der gesuchten Passwörter vom Präprozessor generiert.

Als Folge der Abbildung von persönlichen Informationen auf Muster-Ebene, werden diese ausschließlich beim Vorliegen eines der entsprechenden Muster-Elemente B, N, E, A, C, I in die Kandidatengenerierung miteinbezogen. Demnach ist die Voraussetzung für die Berücksichtigung der persönlichen Informationen in den Passwort-Kandidaten, dass der Benutzer bereits mindestens eine persönliche Information in die Vergleichspasswörter integriert hat.
Im Rahmen dieser Arbeit werden die persönlichen Informationen unabhängig von den bisherigen Designprinzipien eines Benutzers berücksichtigt. Demnach werden die persönlichen Daten eines Benutzers auch dann, wenn keine persönlichen Informationen in den Basis-Passwörtern vorliegen, in den Generierungsprozess von Kandidaten miteinbezogen. Aus diesem Grund werden persönliche Informationen bei der Erstellung der

Muster-Klasse nicht explizit detektiert.
Die Umsetzung erfolgt über die Aufnahme der persönlichen Informationen in die Wortliste der zugehörigen Fragment-Klasse. Der Name beispielsweise findet somit immer Berücksichtigung in der alphabetischen Wortliste und das Geburtsdatum ist immer Teil der numerischen Liste (siehe Abschnitt 5.5.3). Die Erfolgswahrscheinlichkeit der Fragmente bestimmt sich analog zu den anderen Fragmenten der Liste über die Längendifferenz zum Basis-Fragment der befüllenden Muster-Klasse.

7.4.4 Modell von Zhang et. al

Das Modell von Zhang et. al. konzentriert sich auf den Aspekt der exakten sowie partiellen Wiederverwendung eines Passworts von einem Benutzer. Der Präprozessor generiert neue Kandidaten durch Anwendung definierter Transformationsregeln auf die vom Benutzer vorliegenden Passwörter (siehe Abschnitt 3.2.2). Die Substitution eines alphabetischen Fragments wird hierbei nicht berücksichtigt, sodass der Präprozessor im Ergebnis keines der gesuchten Passwörter aus Experiment 1 generiert.

Im Rahmen des benutzerspezifischen Modells wird die Möglichkeit, dass ein Benutzer eines der Basis-Passwörter erneut verwendet über die Nachbildung der Basis-Muster-Klassen sowie über die Aufnahme der Basis-Fragmente in die jeweilige Wortliste berücksichtigt. Es ist somit sichergestellt, dass der Präprozessor die vorliegenden Vergleichspasswörter generiert.
Die Möglichkeit, dass ein Benutzer eines der Basis-Passwörter in modifizierter Form wiederverwendet, wird über die Generierung zusätzlicher Muster-Klassen durch Anwendung der definierten Modifikationsprinzipien berücksichtigt. Neben einer strukturellen Modifikation der Passwörter wird auch eine inhaltliche Veränderung berücksichtigt. Hierfür werden die zur Befüllung der Muster-Klassen herangezogenen Wortlisten, wie in den Abschnitten 5.5.3.1-5.5.3.3 beschrieben, um eine Vielzahl von Fragmenten unterschiedlicher Quellen erweitert.

7.4.5 Modell von Wang et. al.

Die Strategie des ersten Modells von Wang et. al. folgt, wie in Abschnitt 3.2.3 beschrieben, dem Modell von Weir et. al. mit dem Fokus der Abbildung persönlicher Informationen auf Muster-Klassen-Ebene. Folglich generiert auch dieser Präprozessor keines der gesuchten Passwörter des ersten Experiments.

Das zweite Modell von Wang et. al. beschäftigt sich, wie das Modell von Zhang et. al., mit der partiellen Wiederverwendung von Passwörtern von einem Benutzer. Wie in Abschnitt 3.2.3 aufgelistet, umfassen die Transformationsregeln das Einfügen sowie das Löschen von Fragmenten eines Passworts. Über die Kombination beider Transformationen kann die Substitution eines Passwort-Elements erzielt werden. Zudem beinhalten

die Transformationen die Veränderung der Groß- und Kleinschreibung. Demzufolge generiert der Präprozessor unter Verwendung der alphabetischen Wortliste des 1. Experiments alle 3 gesuchten Passwörter.
Für die Überführung des Basis-Passworts „Fortnite1337“ in das gesuchte Passwort „stormflip1337“ ist das Fragment „Fortnite“ über eine Löschoperation zu entnehmen und das Fragment „stormflip“ über eine Einfügeoperation zu ergänzen. Für die Überführung in die Passwörter „TiltedTowers1337“ und „BushCamper1337“ sind eine Löschoperation, zwei Einfügeoperationen sowie zwei Operationen zur Konvertierung des jeweils ersten Buchstaben zu einem Großbuchstaben notwendig.

Die Gemeinsamkeit mit dem Modell dieser Arbeit besteht in der Abhängigkeit der Erfolgswahrscheinlichkeit eines Passwort-Kandidaten von der Art der vorgenommenen Modifikation(en). Diese sind anhand ihrer Anwendungshäufigkeit durch eine repräsentative Gruppe von Benutzern eines Passwort-Leaks gewichtet.
Dieser Strategie folgend, bestimmt sich die Erfolgswahrscheinlichkeit eines Modifikationsprinzips anhand des bisherigen Modifikationsverhalten des betrachteten Benutzers (siehe Abschnitt 5.5.2.2). Wendet der Benutzer bereits eines der definierten Modifikationsprinzipien zwischen seinen Vergleichspasswörtern an, werden diese Modifikationen am stärksten gewichtet. Findet keines der Prinzipien bislang durch den Benutzer Anwendung, bestimmt sich die Gewichtung, wie im Modell von Wang et. al., anhand der Anwendungshäufigkeit durch die Benutzer eines Passwort-Leaks. Nach Definition 5.3 bestimmt die Erfolgswahrscheinlichkeit einer Modifikation die Erfolgswahrscheinlichkeit einer neu generierten Muster-Klasse und diese beeinflusst wiederum nach Definition 5.6 die Erfolgswahrscheinlichkeit eines Kandidaten. Demzufolge liegt ein direkter Zusammenhang zwischen der Modifikationsart und der Erfolgswahrscheinlichkeit eines Kandidaten vor.

7.5 Schlussfolgerungen

Die Effizienz der benutzerspezifischen Attacke wird grundsätzlich an der Rekonstruktionszeit, d.h. an der Rechenzeit, die eine erfolgreiche Rekonstruktion in Anspruch nimmt, gemessen. Sie bestimmt sich durch die Effizienz des Präprozessors, die Komplexität der zugrundliegenden Hashfunktion, der verwendeten Hardware sowie durch den Angriffsmodus.

1. Effizienz des Präprozessors

Der maßgebliche Einflussfaktor ist die Effizienz des Präprozessors. Diese bemisst sich anhand der Erfolgswahrscheinlichkeit des gesuchten Passwort-Kandidaten, die den Stellenwert des Passworts im gesamten Kandidatenraum festlegt. Die durchgeführten Experimente zeigen, dass je enger der Benutzer seinen bisherigen Design- und Konstruktionsprinzipien folgt, desto größer ist die Effizienz des Präprozessors.

Auf struktureller Ebene folgt daraus, dass je ähnlicher sich die Strukturen der Vergleichspasswörter und die des gesuchten Passworts sind, desto geringer ist die Rekonstruktionszeit. Grund hierfür ist, dass die Basis-Muster-Klassen stets die höchste Erfolgswahrscheinlichkeit und somit den höchsten Stellenwert innerhalb der Gesamtheit an Muster-Klassen besitzen. Folglich werden sie im Erzeugungsprozess neuer Passwort-Kandidaten als erstes herangezogen und nachgebildet. In absteigender Reihenfolge der Erfolgswahrscheinlichkeit folgen die zusätzlich generierten Muster-Klassen[46]. Ihre Erfolgswahrscheinlichkeit bestimmt sich zum einen durch die Art der Modifikation sowie durch das bisherige Modifikationsverhalten des Benutzers.
Es ist zu beachten, dass die nachzubildenden Muster-Klassen auf die Basis-Muster-Klassen und die beliebtesten Modifikationen (Modifikationsprinzipien 1 bis 8) begrenzt sind. Es besteht somit die Möglichkeit, dass eine gesuchte Passwortstruktur kein Bestandteil der nachzubildenden Muster-Klassen ist.

Auf inhaltlicher Ebene bedeutet das Festhalten an den Konstruktions- und Designprinzipien, dass je geringer der Längenunterschied zwischen dem gesuchten Fragment und den Basis-Fragmenten der Muster-Klasse ist, desto geringer ist die Rekonstruktionszeit. Grund hierfür ist, dass die Basis-Fragmente stets die höchste Erfolgswahrscheinlichkeit und somit die höchste Stelle innerhalb der jeweiligen Wortliste besitzen. Folglich werden sie als die ersten Fragmente zur Befüllung einer Muster-Klasse verwendet. In absteigender Reihenfolge der Erfolgswahrscheinlichkeit folgen die neuen, zusätzlichen Fragmente, deren Erfolgswahrscheinlichkeit mit zunehmender Entfernung von den Längen der Basis-Fragmente sinkt.
Es ist zu beachten, dass die zur Befüllung einer Muster-Klasse herangezogenen Fragmente auf die Einträge in den entsprechenden Wortlisten begrenzt sind. Diese werden ausschließlich unter Berücksichtigung, der in Abschnitt 5.5.3 aufgelisteten Quellen erstellt. Es besteht somit die Möglichkeit, dass ein gesuchtes Fragment kein Bestandteil der Wortlisten ist. Besonders in der alphabetischen Fragment-Klasse, die im Vergleich zu den Zahlen und Sonderzeichen, einen sehr großen Korpus aufweist, wird eine starke Begrenzung in der Anzahl an berücksichtigten Fragmenten vollzogen.

Es lässt sich somit schlussfolgern, dass je größer die strukturelle und inhaltliche Übereinstimmung zwischen dem gesuchten Passwort-Kandidaten und einem der Vergleichspasswörter ist, desto höher wird die Erfolgswahrscheinlichkeit des Kandidaten vom Präprozessor bemessen. Der entsprechende Kandidat findet daher frühzeitig Berücksichtigung im Angriffsprozess, wodurch die Anzahl an bereits getesteten Kandidaten minimiert wird und effizientere Rekonstruktionszeiten erzielt werden.

[46] Die Generierung erfolgt über den Muster-Generator durch Anwendung der Modifikaktionsprinzipien auf die Basis-Muster-Klassen.

2. Komplexität der Hashfunktion

Eine weitere Einflussgröße der Rekonstruktionszeit stellt die Komplexität der zugrundeliegenden Hashfunktion des Ziel-Hashwerts dar. Sie bestimmt die Geschwindigkeit der Hashwertberechnung der einzelnen Passwort-Kandidaten sowie dessen Abgleich mit dem Ziel-Hashwert.

3. Hardware

Darüber hinaus spielt die zugrundeliegende Hardware bei der Angriffsdurchführung eine entscheidende Rolle. Die Verwendung einer GPU statt einer CPU beispielsweise ermöglicht durch den Einsatz einer Vielzahl von Threads eine parallele Hashwertberechnung, wodurch die Hashgeschwindigkeit gesteigert und die Rechenzeit verkürzt werden kann.

4. Angriffsmodus

Zuletzt hängt die Rekonstruktionszeit von dem Angriffsmodus ab. Im Online-Modus wird aufgrund des Verzichts auf eine Zwischenspeicherung sowie auf eine Sortierung der Gesamtheit an Kandidaten eine Zeit- sowie Speicherplatzersparnis erzielt. Der Zeitersparnis steht jedoch eine geringere Hashrate als im Offline-Modus entgegen. Der Grund hierfür ist die zeitaufwändige Kandidatenerstellung, die durch die Bedingung der Ausgabe der Kandidaten in absteigender Reihenfolge ihrer Erfolgswahrscheinlichkeit verursacht wird. Die Weiterleitung der Kandidaten an Hashcat findet daher mit sehr langsamer Geschwindigkeit statt.
Im Offline-Modus hingegen liegen die zu testenden Kandidaten bereits vorsortiert in einem Wörterbuch vor, wodurch die erzielte Hashrate im Rahmen eines Wörterbuchangriffs deutlich höher ist. Hierbei muss jedoch der zeitliche Aufwand zur Generierung und Sortierung des Wörterbuchs sowie der notwendige Speicherplatz berücksichtigt werden.
Am Beispiel des zweiten Experiments mit einem Kandidatenraum von rund 36 Mrd. Einträgen wären rund 560 GB Speicherplatz notwendig. Dieser Umfang stößt bereits an die Kapazitätsgrenzen eines handelsüblichen Rechners. Im Rahmen des ersten Experiments mit rund 253 Bio. Einträgen liegt der notwendige Speicherplatz bei schätzungsweise 4 PB. Diese Dimension liegt weit außerhalb des Umfangs einer realistischen Speicherung.

8 Schluss

Die in der Einleitung getroffene Vermutung, dass Benutzer instinktiv einfache, leicht zu merkende Passwörter wählen, konnte durch die Analyse eines Passwort-Leaks bestätigt werden. Rund $15,5\%$ der Passwörter besitzen lediglich eine Länge zwischen einem und sechs Zeichen und sind somit unter geringem Ressourcenaufwand über eine Bruteforce-Attacke zu rekonstruieren. Die Rekonstruktionsrate eines Wörterbuch-Angriffs liegt bei rund 96%, was die häufige Verwendung von Standard-Passwörtern verdeutlicht. Bei den drei meistgenutzten Passwörtern handelt es sich um die Ziffernfolgen „123456“, „123456789“ und „12345“. Die geringe Passwortstärke wird durch die Tatsache verdeutlicht, dass die Mehrheit der Benutzer nur einen Teil der verfügbaren Zeichentypen nutzen, wodurch ihre Passwörter lediglich 20% der maximalen Komplexität besitzen. Der vermehrte Einsatz von Passwortrichtlinien von den Online-Diensten führt jedoch zu einem steigenden Bewusstsein der Benutzer bezüglich der Passwortsicherheit und in Folge zur Verwendung stärkerer Passwörter. Dies führt zu einem steigenden Schwierigkeitsgrad bei der Rekonstruktion von Passwörtern.

Für die Umsetzung des Ziels der Rekonstruktion des Passworts eines spezifischen Benutzers unabhängig von dessen Stärke, gliedert sich das Modell in die Bruteforce-Attacke, den Wörterbuch-Angriff und den benutzerspezifischen Angriff. Die ersten zwei Angriffsprozesse rekonstruieren das gesuchte Passwort für den Fall, dass es sich um ein kurzes bzw. ein Standard-Passwort handelt. Bei erfolglosem Angriff oder ineffizienten Rekonstruktionszeiten kommt der benutzerspezifische Angriff zum Einsatz. Mit dem Ziel der Optimierung der Rekonstruktionszeit werden ausschließlich die wahrscheinlichsten Passwort-Kandidaten für den spezifischen Benutzer getestet.

Die Generierung der wahrscheinlichsten Kandidaten erfolgt über den Präprozessor. Er berücksichtigt eine große Bandbreite an Gestaltungsmöglichkeiten eines Passworts auf Basis der vorliegenden Vergleichspasswörter sowie der persönlichen Information des Benutzers. Die Bandbreite beinhaltet unter anderem die exakte Wiederverwendung von Passwörtern sowie die partielle Wiederverwendung durch erneute Benutzung der Muster-Klasse oder der Passwort-Fragmente.
Wie die Untersuchung des Leaks gezeigt hat, verwenden rund 66% der Benutzer ein Passwort mehrfach für verschiedene Konten. Eine erneute Verwendung von einzelnen Passwort-Elementen nehmen rund 52% der Benutzer vor und rund 39% halten an ihren bisherigen Passwortstrukturen fest. Demnach bilden die vom Benutzer verwendeten Passwortstrukturen und Passwort-Fragmente („Basis-Muster-Klasse“ und „Basis-Fragmente“) die Grundlage der Kandidatengenerierung.
Grundsätzlich findet der Generierungsprozess über die Nachbildung von Muster-Klassen statt. Jedes Muster-Klassen-Element wird mit Fragmenten der Wortlisten der zugehörigen Fragment-Klasse befüllt und erzeugt einen Passwort-Kandidaten. Zur Berück-

sichtigung einer strukturellen Modifikation eines Passworts durch den Benutzer werden zusätzliche Muster-Klassen generiert, die im Angriff ebenfalls nachgebildet werden. Die Generierung erfolgt über die Anwendung definierter Modifikationsprinzipien auf die Basis-Muster-Klassen, welche die beliebtesten, von Benutzern vorgenommenen Modifikationen zwischen Passwörtern repräsentieren.
Zur Berücksichtigung einer inhaltlichen Modifikation werden die Wortlisten, die zur Befüllung der Muster-Klassen herangezogen werden, um benutzerspezifische Einträge ergänzt. Eine Quelle sind die persönlichen Daten des Benutzers. Wie gezeigt werden konnte, integrieren rund 57% der Benutzer persönliche Informationen in Form des Vor- oder Nachnamens, dem Geburtsdatum oder dem Benutzernamen in ihre Passwörter. Weitere Quellen sind unter anderem Internetseiten mit Bezug zu den Benutzerinteressen, vordefinierte Wörterbücher oder einzelne Bestandteile der Basis-Fragmente sowie Zahlen- und Sonderzeichenräume definierter Länge.
Der Stellenwert eines Passwort-Kandidaten im gesamten Kandidatenraum bestimmt sich durch seine Erfolgswahrscheinlichkeit, das gesuchte Passwort zu sein. Diese berechnet sich aus der Erfolgswahrscheinlichkeit der Muster-Klasse des Kandidaten sowie aus den Einzelwahrscheinlichkeiten der verwendeten Fragmente. Bei beiden Komponenten besitzen die Basis-Elemente des Benutzers die höchste Erfolgswahrscheinlichkeit. Den neuen Muster-Klassen wird in Abhängigkeit des angewendeten Modifikationsprinzips sowie des bisherigen Modifikationsverhalten des Benutzers eine Erfolgswahrscheinlichkeit zugeordnet. Die neuen Fragmente hingegen, werden in Abhängigkeit ihrer Längendifferenz zu den Basis-Fragmenten der Muster-Klasse gewichtet. Wie gezeigt werden konnte, besitzen die alphabetischen Fragmente zwischen den Passwörtern eines Benutzers durchschnittlich maximal einen Längenunterschied von 3 Zeichen. Bei den Zahlen und Sonderzeichen weichen die Längen durchschnittlich nur um 1 bzw. um 0 Zeichen ab. Demzufolge sinkt die Erfolgswahrscheinlichkeit eines Fragments mit zunehmender Längenabweichung.
Im Angriffsprozess werden die Kandidaten in absteigender Reihenfolge ihrer Erfolgswahrscheinlichkeit getestet. Die Effizienz des Präprozessors, den den Stellenwert eines Kandidaten im Kandidatenraum bestimmt, ist somit die maßgebliche Einflussgröße der Rekonstruktionszeit. Zudem hat die Komplexität der Hashfunktion, die die maximale Hashgeschwindigkeit bestimmt, sowie die zugrundeliegende Hardware einen entscheidenden Einfluss auf die Effizienz des benutzerspezifischen Angriffs. Zuletzt spielt der Modus der Angriffsdurchführung eine Rolle. Während im Online-Modus eine Zeitersparnis durch den Verzicht auf eine Zwischenspeicherung und Sortierung der Kandidaten erzielt wird, werden im Offline-Modus höhere Hashgeschwindigkeiten erreicht.

Wie im Rahmen der Experimente gezeigt wurde, ist das Modell in der Lage benutzerspezifische Passwörter höherer Stärke zu rekonstruieren. Je enger der Benutzer seinen bisherigen Design-und Konstruktionsprinzipien folgt, desto effizienter ist der benutzerspezifische Angriff. Dies bedeutet: Je geringer die strukturelle Abweichung sowie die Längenabweichung der Fragmente von den Basis-Passwörtern ist, desto größer ist die Erfolgswahrscheinlichkeit des Kandidaten. In Folge dessen besitzt er einen umso hö-

heren Stellenwert im Kandidatenraum, wodurch die Rekonstruktionszeit aufgrund einer geringeren Anzahl von zuvor getesteten Kandidaten sinkt. Des Weiteren wurde verdeutlicht, dass alternative Angriffsmethoden wie die Bruteforce-Attacke zur Rekonstruktion stärkerer Passwörter nicht effizient einsetzbar sind. Eine erfolgreiche Rekonstruktion ist aufgrund der höheren Gesamtlänge der Passwörter und der daraus resultierenden sehr hohen Rekonstruktionszeiten vollkommen ausgeschlossen. Zudem konnte bewiesen werden, dass alternative Präprozessoren durch eine zu starke Begrenzung des Kandidatenraums bei größerer Abweichung des Benutzers von seiner bisherigen Passwortgestaltung nicht in der Lage sind, das gesuchte Passwort zu rekonstruieren. So werden unter anderem keine neuen Passwortstrukturen sowie keine neuen Passwort-Elemente berücksichtigt. Zudem finden die persönlichen Informationen des Benutzers ausschließlich Berücksichtigung bei der Kandidatengenerierung, wenn der Benutzer bereits mindestens einmal eine persönliche Information in ein Passwort integriert hat.

Ein möglicher Optimierungsansatz zur Verbesserung der Rekonstruktionszeit sowie zur Behebung des Zielkonflikts zwischen Rechenzeit und Speicherkapazität liegt in der Definition eines Schwellenwerts. Die Erfolgswahrscheinlichkeit des generierten Kandidaten muss zur Qualifikation als Passwort-Kandidat einen definierten Schwellenwert übersteigen. Auf diese Weise kann der Kandidatenraum pro Muster-Klasse auf die wahrscheinlichsten Kandidaten begrenzt werden. Die Höhe des Schwellenwerts könnte zudem in Abhängigkeit von der Komplexität der Hashfunktion sowie der Größe der Wortlisten automatisiert festgelegt werden.
Ein weiterer Optimierungsansatz liegt in der Bestimmung der Erfolgswahrscheinlichkeit eines Fragments in Abhängigkeit von den anderen Fragmenten im Passwort-Kandidaten. Auf diese Weise können Kandidaten, die an unterschiedlichen Fragmentpositionen dasselbe Fragment besitzen, schwächer gewichtet werden und später im Angriffsprozess als unwahrscheinlichere Kandidaten berücksichtigt werden.
Im Falle eines erfolglosen Angriffsprozesses kann als vierter Angriffsprozess ein standardisierter Wörterbuch-Angriff unter Verwendung eines benutzerspezifischen Regelwerks durchgeführt werden. Die Ergebnisse des Präprozessors in Form von den identifizierten Passwortstrukturen sowie den identifizierten Passwort-Fragmente können in einem Regelwerk definiert werden und auf ein gängiges Wörterbuch angewendet werden. Auf diese Weise können die benutzerspezifischen Designprinzipien mit den meistgenutzten Passwörter bzw. Passwort-Elementen kombiniert werden.
Des Weiteren können die gegebenen Vergleichspasswörter unabhängig von der Kandidatengenerierung durch den Präprozessor als die ersten Kandidaten im Angriffsprozess getestet werden.
Trotz automatisierter Kandidatengenerierung durch den Präprozessor sowie der Vielzahl an Möglichkeiten weiterer Optimierungen hat sich im Rahmen dieser Forschungsarbeit gezeigt, dass die Rekonstruktion eines benutzerspezifischen Passworts stets manuelle Arbeit und menschliches Wissen bzw. Kombinationsfähigkeit erfordert. Sobald die individuellen Vorlieben und Gewohnheiten von Benutzern Berücksichtigung finden sollen, sind weiterhin Kombinationsfähigkeiten und Humankreativität erforderlich.

Anhang A: Template des 1. Experiments

```
#Data recording
title = "Case_number_1.0"

#Personal information of suspected person and related person
[[information]]

prename = "Samuel"
second_prename = ""
lastname = "Krug"
street = "Flurstraße"
streetnumber = 92
postalcode = 86368
placeofresidence = "Gersthofen"
dateofbirth = 1983-08-25
placeofbirth = "Augsburg"
nationality = "deutsch"
mobilenumber = ""

[[information]]

prename = "Charlotta"
second_prename = ""
lastname = "Krug"
street = "Flurstraße"
streetnumber = 92
postalcode = 86368
placeofresidence = "Gersthofen"
dateofbirth = 1988-04-18
placeofbirth = "Augsburg"
nationality = "deutsch"
mobilenumber = ""

#Account names
[[accounts]]

Fragment1 = "hexakill"
Fragment2 = "1111"
Fragment3 = ""
Fragment4 = ""
Fragment5 = ""
```

```
[[accounts]]

Fragment1 = "Krug"
Fragment2 = "."
Fragment3 = "S"
Fragment4 = "."
Fragment5 = "83425"

#Passwords
[[passwords]]

Fragment1 = "Fortnite"
Fragment2 = "1337"
Fragment3 = ""
Fragment4 = ""
Fragment5 = ""

[[passwords]]

Fragment1 = "gitgud"
Fragment2 = "kk"
Fragment3 = "thx"
Fragment4 = "bye"
Fragment5 = "69"

#URLs
[urls]

Url1 = "https://de.wikipedia.org/wiki/Fornite"
Url2 = "https://en.wikipedia.org/wiki/Fornite"

#Occupation
[occupation]

job = ["Pharmacist"]

#Leisure activities
[activities]

hobbies = ["online","gaming","esport","betting","league",
                "events","fornite","valorant","card","games"]
```

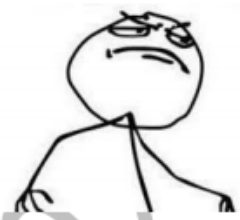
Anhang B: Benutzer-Profil des 1. Experiments

Central Office
for Information Technology
in the Security Sector

ZITiS

Case #: 2020-01-F085-2

Suspect Profile

Name	Samuel Krug	**Photo:** 
Date of birth	25.08.1983	
Place of birth	Augsburg, Germany	
Nationality	German	
Place of residence	Flurstr. 92, 86368 Gersthofen	
Marital status	Married, two adopted kids	

Related Persons

Person 1:

Name	Charlotta Krug
Relation to suspect	Wife
Date of birth	18.04.1988
Place of birth	Augsburg, Germany
Nationality	German
Place of residence	Flurstr. 92, 86368 Gersthofen

Person 2:

Name	---
Relation to suspect	
Date of birth	
Place of birth	
Nationality	
Place of residence	

Person 3:

Name	---
Relation to suspect	
Date of birth	
Place of birth	
Nationality	
Place of residence	

Further Information

Occupation	Employee at a pharmacy
Education/training	Pharmacist (study at Ludwig Maximilian University, Munich)
Social environment	Several online gaming "clans"; parents-in-law live next door; other than that only social contacts via his wife
Leisure activities	Online gaming (competitive Fortnite, Valorant and several card games) Esport betting, attending major esport league events
Social networks	Steam Profile: • Several reviews of games (all rather angry) • Owns 287 games on this platform, mostly shooter games Instagram: • Mostly pictures of his daughters • Custom skin designs for online games • Game Conventions
Characteristics	Plays in a semi-professional Fortnite clan, very toxic behaviour in online games His room in the family house is very messy, but not dirty (family has a cleaner coming once a week) Expensive and luxurious PC setup (watercooled PC, expensive case, curved monitor, etc.)

Known Account Data

Platform	Account name	Password
Steampowered.com	hexakill1111	Fortnite1337
Google	Krug.S.83425	gitgudkkthxbye69

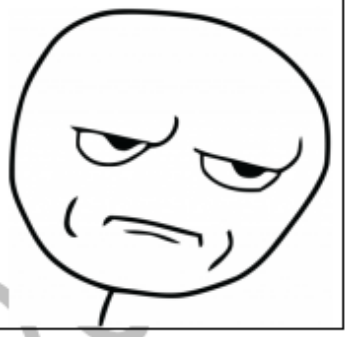
Anhang C: Benutzer-Profil des 2. Experiments

Central Office
for Information Technology
in the Security Sector

ZITiS

Case #: 2020-04-B101-1

Suspect Profile

Name	Erwin August Bachmeier	**Photo:** 
Date of birth	11.12.1975	
Place of birth	Bonn, Germany	
Nationality	German	
Place of residence	Schweinfurter Weg 99 60599 Frankfurt am Main	
Marital status	Divorced, no kids	

Related Persons

Person 1:

Name	
Relation to suspect	
Date of birth	
Place of birth	
Nationality	
Place of residence	

Person 2:

Name	
Relation to suspect	
Date of birth	
Place of birth	
Nationality	
Place of residence	

Person 3:

Name	
Relation to suspect	
Date of birth	
Place of birth	
Nationality	
Place of residence	

Further Information

Occupation	Tax consultant, works from home
Education/training	German business administration degree Certified Tax Consultant Certified Specialist in international tax law
Social environment	Lives alone in his own hose Besides business contacts there are no social contacts No living relatives Active correspondence to various experts in roman history
Leisure activities	Woodworking furniture in his garage Treasure hunting with metal detectors Reading books related to roman history and mythology
Social networks	Facebook profile sporadically updated with new photos
Characteristics	House decorated with various roman sculptures and artifacts Owns a summer estate in Italy Online-Blog about latin language and poems

Known Account Data

Platform	**Account name**	**Password**
www.grapper.de	Erwin Bachmeier	AVGVSTV.LXXV
Facebook	Erwin Bachmeier	MCCXXXIV
Ebay	AugustusXIXII	PANTOPOLIVM.XI

Anhang D: Benutzer-Profil des 3. Experiments

Central Office
for Information Technology
in the Security Sector

ZITiS

Case #: 2020-06-K357-1

Suspect Profile

Name	Natasha Rybak, former Wosz	**Photo:** 
Date of birth	04.01.1983	
Place of birth	Kiev, Ukraine	
Nationality	Russian	
Place of residence	Smolenki reki naberezhnaya 27, Saint Petersburg, Russia	
Marital status	Married, no kids	

Related Persons

Person 1:

Name	Lucasz Rybak
Relation to suspect	Husband
Date of birth	10.09.1979
Place of birth	Saint Petersburg
Nationality	Russian
Place of residence	Smolenki reki naberezhnaya 27, Saint Petersburg, Russia

Person 2:

Name	---
Relation to suspect	
Date of birth	
Place of birth	
Nationality	
Place of residence	

Person 3:

Name	---
Relation to suspect	
Date of birth	
Place of birth	
Nationality	
Place of residence	

Further Information

Occupation	Designer
Education/training	Art School Kiev
Social environment	Lives with husband in the same apartment
Leisure activities	Professional Chess player (Grandmaster level) Ballet Kickboxing
Social networks	Twitter, mostly for connecting within the chess community
Characteristics	Very ambitious, competitive person while being ordinary, unflashy in the public

Known Account Data

Platform	**Account name**	**Password**
twitter	Nat.Ryb	Octopuse6
facebook	Natasha R.	GreekGiftBxh7+
google	NatashaRybak	Gambitexf4

Anhang E: Struktur des Modell-Ordners

Modell/

analysis.py
generate_dictionaries.py
generate_candidates.py
README.txt

Modell/Formular/

Kandidatenraum Passwort Getestete Kandidaten Hashrate Rechenzeit Erfolgswkt. (in %)

Literaturverzeichnis

[1] ANDERSON B.: *In the Age of the Data Breach, Why Do We Still Use Bad Passwords?.* https://securityboulevard.com/2018/08/in-the-age-of-the-data-breach-why-do-we-still-use-bad-passwords/. Zuletzt zugegriffen am 31.01.2021.

[2] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Sichere Passwörter erstellen*. https://www.bsi-fuer-buerger.de/BSIFB/DE/Empfehlungen/Passwoerter/passwoerter_node.html. Zuletzt zugegriffen am 31.01.2021.

[3] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Kryptographische Verfahren: Empfehlungen und Schlüssellängen*. In: BSI – Technische Richtlinie, 2020.

[4] BURR W. E., D. F. DODSON, E. M. NEWTON, R. A. PERLNER, W. T. POLK, S. GUPTA UND E. A. NABBUS: *Electronic Authentication Guideline: Information Security*. In: NIST Special Publication 800-63-1, Gaithersburg, MD, USA, 2011.

[5] CHOU H.-C., H.-C. LEE, H.-J. YU, F.-P. LAI, K.-H. HUANG UND C.-W. HSUEH: *Password cracking based on learned patterns from disclosed passwords*. In: International Journal of Innovative Computing, Information and Control, S. 821–839, 2013.

[6] DAS A., J. BONNEAU, M. CAESAR, N. BORISOV UND X. WANG: *The Tangled Web of Password Reuse*. *In: NDSS*, 2014.

[7] DEMPSEY K., N.S. CHAWLA, A. JOHNSON, R. JOHNSTON, A.C. JONES, A. OREBAUGH, M. SCHOLL UND K. STINE: *Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations: Information Security*. In: NIST Special Publication 800-137, 2011.

[8] GALI N., R. MARIESCU-ISTODOR UND P. FRÄNTI: *Similarity measures for title matching*. *In: 2016 23rd International Conference on Pattern Recognition (ICPR)*, S. 1548–1553, 2016.

[9] GITHUB: *John The Ripper*. https://github.com/openwall/john. Zuletzt zugegriffen am 31.01.2021.

[10] GITHUB: *knight-moves*. https://github.com/opalkale/knight-moves/blob/master/wordlist.txt. Zuletzt zugegriffen am 31.01.2021.

[11] GITHUB: *latin_proper_names_cltk*. https://github.com/cltk/latin_proper_names_cltk/blob/master/proper_names.txt. Zuletzt zugegriffen am 31.01.2021.

[12] GITHUB: *name-dataset*. https://github.com/philipperemy/name-dataset/tree/master/names_dataset. Zuletzt zugegriffen am 31.01.2021.

[13] GRASSI P., E. NEWTON, R. PERLNER, A. REGENSCHEID, J. FENTON, W. BURR, J. RICHER, N. LEFKOVITZ, J. DANKER, Y. CHOONG, K. GREENE UND M. THEOFANOS: *Digital Identity Guidelines: Authentication and Lifecycle Management*. In: NIST Special Publication 800-63B, 2017.

[14] HASHCAT: *Combinator Attack*. https://hashcat.net/wiki/doku.php?id=combinator_attack. Zuletzt zugegriffen am 31.01.2021.

[15] HASHCAT: *Hashcat - advanced password recovery*. https://hashcat.net/hashcat/. Zuletzt zugegriffen am 31.01.2021.

[16] HASHCAT: *Hybrid Attack*. https://hashcat.net/wiki/doku.php?id=hybrid_attack. Zuletzt zugegriffen am 31.01.2021.

[17] HASHCAT: *Mask Attack*. https://hashcat.net/wiki/doku.php?id=mask_attack. Zuletzt zugegriffen am 31.01.2021.

[18] HASHCAT: *Maskprocessor*. https://hashcat.net/wiki/doku.php?id=maskprocessor. Zuletzt zugegriffen am 31.01.2021.

[19] HASHCAT: *Rule-based Attack*. https://hashcat.net/wiki/doku.php?id=rule_based_attack. Zuletzt zugegriffen am 31.01.2021.

[20] HASSO-PLATTNER-INSTITUT: *Die beliebtesten deutschen Passwörter 2020*. https://hpi.de/news/jahrgaenge/2020/die-beliebtesten-deutschen-passwoerter-2020-platz-6-diesmal-ichliebedich.html. Zuletzt zugegriffen am 31.01.2021.

[21] HONAN M.: *Kill the Password: A String of Characters Won't Protect You*. https://www.wired.com/2012/11/ff-mat-honan-password-hacker/. Zuletzt zugegriffen am 31.01.2021.

[22] HRANICKÝ R., F. LIŠTIAK, D. MIKUŠ UND O. RYŠAVÝ: *On Practical Aspects of PCFG Password Cracking*. *In: 33th IFIP Annual Conference on Data and Applications Security and Privacy*, Data and Applications Security and Privacy XXXIII, S.43–60. Springer International Publishing, 2019.

[23] ILYANKOU I.: *Comparion of Jaro-Winkler and Ratcliff/Obershelp algorithms in spell check*. *In: Open Journal of Applied Sciences*, Vol.9 N.12, 2014.

[24] IONOS SE: *NTLM: WIe funktioniert das Protokoll?* https://www.ionos.de/digitalguide/server/knowhow/ntlm-nt-lan-manager/. Zuletzt zugegriffen am 31.01.2021.

[25] KUHLEMANN O.: *Leetspeak Code*. http://kryptografie.de/kryptografie/chiffre/leetspeak.htm, 2021. Zuletzt zugegriffen am 31.01.2021.

[26] LEVENSHTEIN V.I.: *Binary Codes Capable of Correcting Deletions, Insertions and Reversals*. In: Soviet Physics Doklady, Vol.10:S. 707–710, 1966.

[27] LI Y., H. WANG UND K. SUN: *A Study of Personal information in Human-chosen Passwords and its Security Implications*. *In: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016.

[28] LI Z., W. HAN UND W. XU: *A Large-Scale Empirical Analysis of Chinese Web Passwords*. *In: USENIX Security Symposium*, 2014.

[29] NAM E.: *Understanding the Levenshtein Distance Equation for Beginners*. https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0. Zuletzt zugegriffen am 31.01.2021.

[30] NATIONAL INSTITUTE OF STANDARDS TECHNOLOGY: *Security Requirements for Cryptographic Modules*. In: Federal Information Processsing Standards Publication FIPS PUB 140-2, National Institute of Standards Technology, Gaithersburg, MD, USA, 2001.

[31] OPENWALL: *John the Ripper password cracker*. https://www.openwall.com/john/. Zuletzt zugegriffen am 31.01.2021.

[32] PINTEREST: *Pinterest*. https://www.pinterest.at/pin/725149977471420214/. Zuletzt zugegriffen am 31.01.2021.

[33] PRESTON-WERNER T.: *Toml: Toms obvious minimal language*. https://toml.io/en/. Zuletzt zugegriffen am 31.01.2021.

[34] PROCTOR R., M.C. LIEN, K.P. VU, E. SCHULTZ UND G. SALVENDY: *Improving computer security for authentication of users: Influence of proactive password restrictions*. In: Behavior research methods, Instruments, computers, Vol.34:S. 163–169, 2002.

[35] PUTTY: *Download: PuTTY*. https://www.putty.org/. Zuletzt zugegriffen am 31.01.2021.

[36] RED HAT: *Ansible Documentation*. https://docs.ansible.com/ansible/latest/index.html. Zuletzt zugegriffen am 31.01.2021.

[37] SHANNON C.E.: *Prediction and entropy of printed English*. In: The Bell System Technical Journal, Vol.30:S. 50–64, 1951.

[38] WANG D., Z. ZHANG, W. ZIJIAN, P. WANG UND Y. JEFF: *Targeted Online Password Guessing: An Underestimated Threat*. *In: CCS*, 2016.

[39] WANG J., L. GUOLIANG UND F. JIANHUA: *Fast-Join: An Efficient Method for Fuzzy Token Matching based String Similarity Join*. *In: 2011 IEEE 27th International Conference on Data Engineering*, S. 458–469, 2011.

[40] WEIR M., S. AGGARWAL, B.D. MEDEIROS UND B. GLODEK: *Password Cracking Using Probabilistic Context-Free Grammars*. *In: 30th IEEE Symposium on Security and Privacy*, S.391–405, 2009.

[41] ZHANG-KENNEDY L., S. CHIASSON UND P. OORSCHOT: *Revisiting Password Rules: Facilitating Human Management of Passwords*. *In: 2016 APWG Symposium on Electronic Crime Research*, S. 1–10, 2016.

[42] ZHANG Y., F. MONROSE UND M.REITER: *The security of modern password expiration: An algorithmic framework and empirical analysis*. *In: Proceedings of the ACM Conference on Computer and Communications Security*, S. 176–186, 2010.

[43] ZITIS: *ZITiS*. https://www.zitis.bund.de/DE/ZITiS/Ueber_Uns/ueber_uns_node.html. Zuletzt zugegriffen am 31.01.2021.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Stuttgart, 25. Februar 2021

Unterschrift
Eva Pothmann