Karlsruher Institut für Technologie

KASTEL

Mensch-ausführbare Kryptographie

Bachelor's Thesis von

Claude Dürr

an der Fakultät für Informatik
KASTEL – Institut für Informationssicherheit und Verlässlichkeit

Erstgutachter:	Prof. Dr. Jörn Müller-Quade
Zweitgutachter:	Prof. Dr. Thorsten Strufe
Betreuender Mitarbeiter:	M.Sc. Felix Dörre

01. November 2023 – 01. März 2024

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Änderungen entnommen wurde.

PLACE, DATE

. . . . . .

(Claude Dürr)

Zusammenfassung

Welche Möglichkeiten stehen einem Menschen zur Verfügung, ohne technische Hilfsmittel Nachrichten zu ver- und entschlüsseln oder sich zu identifizieren? Diese Frage nach mensch-ausführbarer Kryptographie wird anhand der Sicherheit und Nutzbarkeit verschiedener Protokolle untersucht und führt zu einem ernüchternden Ergebnis. Ohne technische Hilfsmittel kann sich ein Mensch bislang keinen signifikanten Vorteil gegenüber einem Computer erarbeiten. Die Art der Mensch-Ausführbarkeit wird in drei Kategorien eingeteilt, die unterschiedliche bzw. keine Hilfsmittel zulassen. Einführend wird auf die Möglichkeiten zur Generierung von Zufall und der Berechnung von Einwegfunktionen und Pseudozufallsgeneratoren kurz eingegangen. Ob Menschen ohne Hilfsmittel Zufall erzeugen können, bleibt eine offene Frage.

One-Time-Pads, Visuelle Kryptographie, Solitaire und LC4 werden als Verschlüsselungsverfahren beschrieben und verglichen. Die Aufwandsabschätzung erfolgt durch das Zählen elementarer Operationen. Die berücksichtigten Schutzziele sind Vertraulichkeit, Integrität und Authentifikation. Konkrete Abhängigkeiten in LC4 lassen IND-CPA-, Broadcast-Angriffe und einen verbesserten Known-Plaintext-Angriff zu. Eine Anpassung der LC4-Verschlüsselungsfunktion kann zumindest diese Abhängigkeiten vermeiden. Es ist bislang kein mensch-ausführbares Verschlüsselungsverfahren konstruiert worden, das alle drei Schutzziele mit einem Schlüssel kürzer als die verschlüsselten Nachrichten erfüllt.

Die beschriebenen Identifikationsprotokolle sind Hopper & Blum 1/2, Foxtail, Weinshall und CG, wobei jeweils bekannte Sicherheitsprobleme, wie z.B. Linearisierung und mögliche statistische Angriffe, genannt werden. Das iChip-Protokoll wird nur informell beschrieben. Ziel der Identifikationsprotokolle ist es, mindestens passive Sicherheit zu erreichen, wobei dies nur für Hopper & Blum 1/2 und gegebenenfalls iChip erreicht wird. Hopper & Blum 2 bietet als einziges Protokoll auch Schutz vor aktiven Angreifern. Für die Identifikationsprotokolle existiert ein erheblicher Trade-Off zwischen Sicherheit und Nutzbarkeit.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Mensch-ausführbare Kryptographie bezeichnet kryptographische Verfahren, die ohne (technische) Hilfsmittel durch einen Menschen ausgeführt werden können. Diese Verfahren können auch dann angewendet werden, wenn keine technischen Hilfsmittel zur Verfügung stehen oder diesen nicht vertraut wird. Vertrauen ist notwendig: Dass beispielsweise eine Messenger-App versendete Daten tatsächlich sicher verschlüsselt, ist für den Anwender alles andere als offensichtlich. Den alltäglichen Nachrichtenaustausch mag dieser Umstand vermutlich nicht stören, für Nachrichten von denen das Wohlbefinden Anderer oder der eigenen Person abhängt, ist es sicher naheliegend, das Vertrauen in technische Hilfsmittel zu hinterfragen. Versuche von Sicherheitsbehörden, Backdoors für den eigenen Zugriff auf verschlüsselte Daten einbauen zu lassen [79, 83], rechtfertigen Misstrauen über den Umstand hinaus, dass es keine Gewissheit darüber geben kann, dass sich Software so verhält, wie vom Nutzer erwartet - sei es durch Schwadsoftware oder Implementierungsfehler. Neben der Verschlüsselung von Daten benötigt auch die Identifikation Vertrauen in die Sicherheit der hinter der Login-Maske verborgenen Infrastruktur. Phishing-Angriffe, Spähsoftware auf dem verwendeten Gerät oder das Abhören eines unverschlüsselten Kanals können unmittelbar das gesamte Passwort offenlegen. Auch das analoge Ausspähen des Passworts in der Öffentlichkeit ist möglich. Mensch-ausführbare Identifikationsprotokolle können helfen, diese Probleme zu lösen.

Struktur der Arbeit Zunächst werden Definitionen für Mensch-Ausführbarkeit gegeben. Die Generierung von Zufall wird als wichtiges kryptographisches Primitiv betrachtet. Als Verschlüsselungsverfahren werden One-Time-Pads (OTPs), Visuelle Kryptographie, Solitaire und LC4 vorgestellt. Visuelle Kryptographie wird dabei nur kurz als Spezialfall von OTPs beschrieben. Die Mensch-Ausführbarkeit von OTPs, Solitaire und LC4 wird durch eine Aufwandsabschätzung und Beschreibung der Berechnung durch Menschen motiviert. Für LC4 wird gezeigt, dass es nicht IND-CPA-sicher und anfällig für Broadcast und Known-Plaintext-Angriffe ist. Unter den betrachteten Identifikationsprotokollen finden sich HB1, HB2, Foxtail, Weinshall und CG. Die Verfahren und Protokolle werden jeweils hinsichtlich ihres Aufwands, der benötigten Hilfsmittel und der erreichbaren Sicherheit verglichen. Der Aufwand von HB2 wird mit der Anzahl an elementaren Operationen, die ein Mensch ausführen muss, abgeschätzt. Die Berechnungen sowie Simulationen sind auf Github [1] verfügbar.

Verwandte Arbeiten Li et al. [58] beschreiben und untersuchen mehrere mensch-ausführbare Identifikationsprotokolle und kommen zu dem Ergebnis, dass die mutmaßlich frühesten mensch-ausführbaren Identifikationsprotokolle (Matsumoto-Imai [66], Matsumoto [64, 65]) keine ausreichende Sicherheit bieten. Yan et al. [96] identifizieren einige

[1] `https://github.com/CMRD24/HumanCrypto` und `https://github.com/CMRD24/LC4Statistics/`

Voraussetzungen, die Identifikationsprotokolle erfüllen müssen und stellt ein Framework zur Ermittlung des kognitiven Aufwands vor. Chakraborty et al. [25] beschreibt, wie opportunistisch-sichere mensch-ausführbare Identifikationsprotokolle praktisch umgesetzt werden können. Halunen et al. [37] erarbeiten eine Übersicht zur Frage, wie die menschlichen Sinne in der Kryptographie Verwendung finden können. Darüber hinaus existieren zahlreiche Vorschläge für mensch-ausführbare Identifikationsprotokolle und Verschlüsselungsverfahren. Für moderne mensch-ausführbare (»Pen-And-Paper«) Verschlüsselung, wie Solitaire, LC4 oder Pocket-RC4, fehlt es bislang an einer Übersicht der Verfahren.

2. Allgemeine Definitionen von Mensch-Ausführbarkeit

Dieser Abschnitt soll einige einleitende Definitionen enthalten, die über verschiedene Anwendungsbereiche hinweg Verwendung finden.

Definition 1 ($(\alpha, \beta, \gamma_1, \gamma_2)$-mensch-ausführbar) *Die Implementierung eines Verfahrens heißt* $(\alpha, \beta, \gamma_1, \gamma_2)$-mensch-ausführbar (human exectutable)*, wenn mindestens ein* $(1 - \alpha)$*-Anteil der Bevölkerung mit Erfolgswahrscheinlichkeit* $\geq (1 - \beta)$ *das Verfahren mit (normalerweise einmaliger) Vorbereitungszeit* $\leq \gamma_1$ *s und Bearbeitungszeit* $\leq \gamma_2$ *s ausführen kann [17, vgl. S.4]. Wird nur* γ_2 *betrachtet, wird das Verfahren als* $(\alpha, \beta, \gamma_2)$-mensch-ausführbar *bezeichnet, wobei keine Annahmen über* γ_1 *getroffen werden [57, S.6, Def.3] [41, 58].*

Ideal (und völlig unrealistisch) wäre $\alpha = \beta = \gamma_1 = \gamma_2 = 0$ [57, vgl.].

Definition 2 (Strenge Mensch-Ausführbarkeit) *Ein Verfahren heißt* streng mensch-ausführbar*, wenn es ohne jegliche Hilfsmittel ausgeführt werden kann [58, vgl. S.28].*

Definition 3 (Schwache Mensch-Ausführbarkeit) *Ein Verfahren heißt* schwach mensch-ausführbar*, wenn es einem Menschen nur Hilfsmittel ohne bedeutungsvollen internen Zustand erlaubt, welche zudem austauschbar sind. Die Vertrauenswürdigkeit der Hilfsmittel muss ohne nennenswerten Aufwand durch einen Menschen festgestellt werden können.*

Beispiel (Hilfsmittel): Papier, Stifte, Scheren, Würfel.

Damit schwach mensch-ausführbare Verfahren sicher verwendet werden können, müssen die verwendeten Hilfsmittel nach Berechnung entweder vernichtet (z.B. Papier für OTPs) oder ihr Zustand randomisiert werden (Kartendeck bei Solitaire, Spielsteine für LC4). Während der Verwendung dürfen die Hilfsmittel nicht beobachtbar sein. Die Austauschbarkeit der Hilfsmittel sollte einen gewissen Schutz vor Manipulationen ermöglichen.

Definition 4 (Unterstützte Mensch-Ausführbarkeit) *Ein Verfahren heißt* unterstützt mensch-ausführbar*, wenn es Hilfsmittel zulässt, die nicht austauschbar sind, da sie ein für das Verfahren benötigte Geheimnis kodieren. Die Vertrauenswürdigkeit der Hilfsmittel muss ohne nennenswerten Aufwand durch einen Menschen festgestellt werden können.*

Beispiel Visuelle Kryptographie und One-Time-Pads mit notiertem Schlüssel.

In den betrachteten Verfahren werden bestimmte Fähigkeiten erwartet, die Menschen besitzen sollten. Zu diesen Fähigkeiten gehören: Addition, Subtraktion, Multiplikation kleiner Zahlen, Zählen, Rotation/Verschiebung und das Finden gleicher Zeichen. Unter Zuhilfenahme zufallsgenerierender Hilfsmittel soll auch das Partitionieren einer Zeichenmenge (z.B. als Teil eines Kartenstapels) und das Generieren zufälliger Zeichen-/Zahlenfolgen möglich sein. Über kognitive Fähigkeiten hinaus werden für die Ausführung einfache motorische (z.B. Solitaire, LC4) oder visuelle (Visuelle Kryptographie) Fähigkeiten benötigt.

3. Kryptographische Primitive

Ein naheliegender Gedanke bei Kontruktion neuer kryptographischer Verfahren besteht in der Betrachtung oder Konstruktion kryptographischer Primitive. Dabei gibt es wenige Arbeiten, die sich mit mensch-ausführbaren kryptographischen Primitiven explizit beschäftigen. Dies mag daran liegen, dass die Komplexitätsanforderungen an derartige Primitive noch unter denen durch sie zusammengesetzten Verfahren liegen sollten, um die Mensch-Ausführbarkeit nicht zu beeinträchtigen.

3.1. Zufall erzeugen

Zur Zufallserzeugung mit Hilfsmitteln (schwach mensch-ausführbar) sind viele Möglichkeiten vorstellbar. Klassische Beispiele sind Münz- und Würfelwurf. Selbst mit gezinkten Münzen und Würfeln ist die Erzeugung einer gleichverteilten Bitfolge möglich: In einem von Von Neumann [74] beschriebenen Verfahren für gezinkte Münzen wird eine Münze (Kopf/Zahl) zwei mal geworfen. Fällt erst Kopf dann Zahl, ist die Ausgabe 'Kopf'. Analog wird 'Zahl' ausgegeben, wenn erst 'Zahl' dann 'Kopf' fällt. Andernfalls, wenn also zweimal die gleiche Seite fällt, wird der Vorgang ohne Ausgabe wiederholt. Juels et al. [49] stellen einen Algorithmus für gezinkte Würfel vor.

Das menschliche Gehirn ist ein schlechter Zufallsgenerator [31, 10, 85]. Unklar ist, ob Menschen dennoch *ohne* Hilfsmittel zuverlässig Zufall erzeugen können. Als Entropiequelle kämen zwar Funktionen des Körpers in Betracht, wie Puls, Blutdruck etc., jedoch müssen diese ohne Hilfsmittel gemessen werden (insbesondere ohne Uhr!). Vorstellbar wäre es, den Puls anhand einer bestimmten Tätigkeit zu messen: Ist die Anzahl Schläge pro definierter gelaufener Strecke gerade oder ungerade? Selbst wenn sich dieser Wert zur Zufallserzeugung nutzen lässt, ist die Bestimmung mit großem Aufwand verbunden, die der alltäglichen Nutzung entgegenspricht. Externe Entropiequellen[1] entfernen sich von der Definition der strengen Mensch-Ausführbarkeit und benötigen die Annahme, dass sie nicht ebenfalls von Angreifern beobachtet werden.

Zufallszahlengeneratoren die menschlich erzeugten Zufall als Entropiequelle verwenden, wurden u.a. in [4, 36, 38] vorgestellt, sind aber nicht mensch-ausführbar. Verfahren die die Entropie bewusst erzeugter Zufallszahlen nutzen, um mit diesen einen kryptographisch sicheren Zufallsstrom zu erzeugen, scheinen am zielführendsten. Derzeit ist kein streng mensch-ausführbares Verfahren bekannt. Zur Entwicklung besonders effizienter Verfahren, könnte die Bildung verallgemeinerter Annahmen über die Verteilung solcher bewusst erzeugten Zufallszahlen eine Herausforderung sein. Studien zeigen nicht nur

[1] Z.B. Nummernschilder auf vorbeifahrenden Fahrzeugen, Anzahl Personen mit roten Kleidungsstücken in Menge

einen Zusammenhang zwischen neurologischen Auffälligkeiten und Zufallsgeneration [80], sondern auch individuelle Unterschiede in den erzeugten Werten [86, 46].

In den meisten der betrachteten Verfahren (z.B. OTPs, Solitaire, LC4) wird davon ausgegangen, dass Geheimnisse zufällig erzeugt werden, um sich vor Wörterbuchangriffen zu schützen. Auch darüber hinaus benötigen manche (LC4, HB1, HB+, CG, iChip) der betrachteten Verfahren Zufall. In der Sicherheitsanalyse ist es naheliegend, für all diese Verfahren allenfalls schwache Mensch-Ausführbarkeit anzunehmen, sofern keine Möglichkeit bekannt werden sollte, auch ohne Hilfsmittel Zufall zu erzeugen.

3.2. Einwegfunktionen und Pseudozufallsgeneratoren

Blum et al. [17] beschreiben mensch-ausführbare Einwegfunktionen und Pseudozufallsgeneratoren [2], von denen konkrete Sicherheit angenommen wird. Eine mensch-ausführbare Einwegfunktion berechnet die Ausgabe mithilfe eines Schlüssels. Bleibt der Schlüssel geheim wird gefordert, dass sich nur eine »kleine«Menge an Ausgaben invertieren lässt, sofern der invertierende Algorithmus nicht mehr als 10^{24} Instruktionen ausführt [17, S.9]. Ein mensch-ausführbarer Pseudozufallsgenerator bildet anhand eines Schlüssels Eingaben der Länge n auf eine zufällig aussehende Ausgabe der Länge $\approx 2n$ ab. Zufällig aussehen heißt, dass die Ausgabe von einem Algorithmus mit höchstens $10^e (12 \leq e \leq 24)$ ausführenden Instruktionen und Fehlerwahrscheinlichkeit $< 1/4$ nicht von tatsächlich zufälligen Ausgaben unterschieden werden kann [17, S.10]. Die unter diesen konkreten Sicherheitsanforderungen vorgestellten Einwegfunktionen und Pseudozufallsgeneratoren sollen streng mensch-ausführbar sein, also ohne jegliche Hilfsmittel berechnet werden können [17, S.12]. Dies benötigt das Auswendiglernen einer geheimen (zufälligen) Zuordnung $M : A \to \mathbb{Z}_{10}$, von den Elementen des verwendeten Alphabets A auf Zahlen von 0 bis 9. Die vorgestellten Verfahren basieren auf dem Skip-To-My-Lou Schema (STML)[17, S.8]:

Protokoll 1 Skip-To-My-Lou (STML) [17, S.8]

Gegeben. geheime Abbildung $M : A \to \mathbb{Z}_{10}$, Challenge $C = (c_1, ..., c_n) \in A^n$

Ablauf:

1. $\Sigma \leftarrow M(c_n)$
2. Für $i \leftarrow 1, ..., n$
   - $\Sigma \leftarrow (\Sigma + M(c_i)) \bmod 10$
   - Falls $\Sigma < 5$, gebe Σ aus

Zunächst wird eine zufällige Challenge C fest gewählt, wobei diese lang genug sein muss, um alle Zeichen aus A zu enthalten ($n \approx |A| \log|A|$). Der Input $x \in \mathbb{Z}_{10}^{|A|}$ wird als Abbildung $M : A \to \mathbb{Z}_{10}$ interpretiert. Die vorgeschlagene Einwegfunktion besteht nun

[2] Im Kontext von Passwort-Schemata

darin, STML mit Challenge C und der aus x erhaltenen Abbildung M auszuführen. Die Ausgabe hat also erwartbar Länge $n/2$.

Für den Pseudozufallsgenerator wird eine abgewandte Form des STML-Schemas als Subroutine verwendet:

Protokoll 2 $STML_{PRG}(M, C, z)$ [17, S.10 f.]

Gegeben. geheime Abbildung $M : \mathbb{Z}_{10} \to \mathbb{Z}_{10}$, Challenge $C = (c_1, ..., c_n) \in \mathbb{Z}_{10}^n$, ein initialer Wert $z \in \mathbb{Z}_{10}$

Ablauf:

1. $\Sigma \leftarrow z$
2. Für $i \leftarrow 1, ..., n$
   - $\Sigma \leftarrow (M(\Sigma) + c_i) \bmod 10$
   - Falls $\Sigma < 5$, Gebe Σ aus

Protokoll 3 PRG1 [17, S.11 f.]

Gegeben. Zufälliger Seed $C = (c_1, ..., c_n) \in \mathbb{Z}_{10}^n$, Abbildung $M : \mathbb{Z}_{10} \to \mathbb{Z}_{10}$. $||$ steht für die Konkatenation.

Ablauf:

1. Bilde C_j die durch abwechselndes Überspringen und Einfügen von j Zeichen aus C für $(\lfloor n/2 \rfloor \geq j \geq 1)$ gebildete Folge. Z.B. $C_1 = (c_2, c_4, ...), C_2 = (c_3, c_4, c_7, c_8, ...)$.
2. Gebe $STML_{PRG}(M, C||C_1||...||C_{\lfloor n/2 \rfloor}, c_n)$ aus.

PRG1 nimmt also einen Seed aus Zahlen in $\mathbb{Z}_{10}$ [3] und bildet diesen auf Zahlen in $\mathbb{Z}_5$ ab. Der Erwartungswert der Ausgabelänge wird von Blum et al. mit $\frac{n(n+1)}{4}$ [17, S.11 f.] angegeben, müsste aber tatsächlich - dadurch dass zuerst je i Zeichen übersprungen und dann eingefügt werden - bei

$$\frac{1}{2}\left(n + \sum_{i=1}^{\lfloor n/2 \rfloor} i \cdot \left\lfloor \frac{n}{2i} \right\rfloor + (n \bmod i)\left(\left\lfloor \frac{n}{i} \right\rfloor \bmod 2\right)\right)$$

liegen. Die Ausgabelänge ist etwa um den Faktor zwei kleiner, als ursprünglich von Blum et al. geschätzt, was für kurze Seeds besonders bedeutsam ist[4]. Damit die Ausgabe im

[3] Kann auch auf $\mathbb{Z}_5$ eingeschränkt werden [17, S.11].

[4] Erwartete Ausgabelängen: Für $n = 10$: 22.5 vs. 13.5. Für $n = 20$: 95 vs. 49.5

Erwartungswert tatsächlich die Länge n des Seeds verdoppelt, muss $n \geq 14$ gelten. Der Zeichenstrom der bei Konkatenation von 10^7 Ausführungen von PRG1 mit fester Abbildung M und jeweils zufälligen Seeds C ($n = 10$) entsteht, wurde mithilfe der dieharder-Testsuite[5] [18] auf notwendige statistische Eigenschaften eines guten Zufallgenerators getestet: Insbesondere die Annahme einer Gleichverteilung der m-Tupel muss nach den sts-serial Tests für alle $1 \leq m \leq 16$ verworfen werden.

Dass manche m-Tupel häufiger auftreten als andere, lässt sich einfach erklären. PRG1 kann so verstanden werden, dass zunächst der Seed expandiert und dann auf $STML_{PRG}$ angewendet wird. Dabei entsteht das Problem, dass der expandierte Seed Muster aufweist, die sich mit nicht-vernachlässigbarer Wahrscheinlichkeit auch in der Ausgabe wiederfinden: Für bspw. $n = 20$ endet der expandierte Seed für einen Seed $C = (c_1, ..., c_{20})$ mit $C_9||C_{10} = c_{10}, c_{11}, ..., c_{18}, c_{11}, ..., c_{18}, c_{19}, c_{20}$. Entsprechend wiederholt sich ein großer Teil der $STML_{PRG}$-Eingabe am Ende. Bei zufälliger Abbildung M und Challenge C liegt die Wahrscheinlichkeit dafür, dass Σ zu Beginn der beiden Wiederholungen den gleichen Wert hat, bei $\frac{1}{10}$. Dies ist genau dann der Fall, wenn $M(...(M(M(\Sigma) + c_{11}) + c_{12})... + c_{18}) \bmod 10 = \Sigma$. Die Ausgabe von PRG1 hat also in 10% aller Fälle die Form $..., g_1, ..., g_\lambda, g_1, ..., g_\lambda, x_1, ..., x_\sigma$ mit $E[\lambda] = 4, E[\sigma] = 1$. Wobei maximal zwei anderen Zeichen am Ende der Ausgabe vorkommen (also $0 \leq \sigma \leq 2$). Bei echt zufälliger Ausgabe ist die Wahrscheinlichkeit, dass sich vier Zeichen an festem Index wiederholen $(\frac{1}{5})^4 = 0.16\%$.

Die Abbildung $M : \mathbb{Z}_{10} \rightarrow \mathbb{Z}_{10}$ kann durch eine Ziffernfolge $k = (k_1, ..., k_{10}) \in \mathbb{Z}_{10}^{10}$ mit $M(i) = k_i$ beschrieben werden. Um die C_j streng mensch-ausführbar berechnen zu können, schlagen Blum et al. [17] das Abzählen der Indizes an den Fingern vor. Darüber hinaus beschreiben sie den menschlichen Aufwand, sich Abbildungen wie M zu merken. Interessant dabei ist insbesondere die Modellierung des Menschen als Turingmaschine, in denen das Band durch zwei RAMs ersetzt wird: Je für das Kurzzeit- (klein und schnell) und das Langzeitgedächtnis (groß und langsam). Insbesondere wird der Unterschied zwischen dem Aufwand beim Lernen einer Liste (z.B. Alphabet) und dem Lernen für wahlfreien Zugriff (z.B. Abbildung auf Morsealphabet) betrachtet [17, S.4]. Während für das SMTL-Schema strenge Mensch-Ausführbarkeit argumentiert werden kann, sollte der zufällige Seed mit Hilfsmitteln generiert werden.

[5] https://linux.die.net/man/1/dieharder

4. Ver- und Entschlüsselung

4.1. Definitionen

Ein symmetrisches Verschlüsselungsverfahren besteht aus einem Paar von Funktionen $\langle Enc, Dec \rangle$. $Enc : K \times P \mapsto C$ verschlüsselt anhand eines Schlüssels $k \in K$ einen Klartext $p \in P$. $Dec : K \times C \mapsto P$ entschlüsselt ein Chiffrat $c \in C$ anhand eines Schlüssels $k \in K$. Eine notwendige Bedingung für die Korrektheit ist [52, S.7]:

$$\forall k \in K, m \in P : Dec(k, Enc(k, m)) = m$$

Einige der Begriffe, die in diesem Kapitel verwendet werden, werden im folgenden informell definiert.

4.1.1. Schutzziele

Vertraulichkeit (Confidentiality): Vertraulichkeit bezieht sich auf die Fähigkeit, Daten vor unberechtigtem lesenden Zugriff zu schützen [5, vgl. S.6]. Ohne Kenntnis des Schlüssels k dürfen aus einem Chiffrat c keine Informationen über den Klartext p gezogen werden können. Eine formalere Definition kann u.a. durch die Definiton von Ununterscheidbarkeit (Indistinguishability, kurz: IND) der Chiffrate erreicht werden [52]. In Unterabschnitt 4.1.2 wird für die etwas formalere Definition eines Angriffs auf die Ununterscheidbarkeit (IND-CPA) und damit Vertraulichkeit gefordert, dass der Angreifer polynomiell beschränkt ist. Wird dies nicht gefordert, ist der Angreifer also beliebig, entspricht die Ununterscheidbarkeit der Intuition von perfekter Sicherheit.

Integrität (Integrity): Integrität bezieht sich auf die Fähigkeit, Daten vor unberechtigten schreibenden Zugriff zu schützen [5, vgl. S.6]. Ohne Kenntnis des Schlüssels k darf das Chiffrat c nicht unbemerkt verändert werden können.

Authentifikation (Authentification/Authenticity): Die Identität der Partei, die das Chiffrat c erstellt hat, muss sichergestellt werden können [5, vgl. S.8].

4.1.2. Angriffsmodelle

Der durch $\mathcal{A}$ bezeichnete Angreifer hat keine Kenntniss über den Schlüssel k. In allen beschriebenen Angriffen ist es offensichtlich hinreichend, den Schlüssel k zu bestimmen.

Ciphertext-Only-Angriff : $\mathcal{A}$ ist das Chiffrat c bekannt und versucht daraus den Klartext p zu bestimmen [43, S.16].

Known-Plaintext-Angriff : $\mathcal{A}$ sind Chiffrate c, c' die mit dem gleichen Schlüssel k erzeugt wurden und der Klartext p zum Chiffrat c bekannt. $\mathcal{A}$ versucht den Klartext p' zum Chiffrat c' zu bestimmen [43, S.16].

Broadcast-Angriff $\mathcal{A}$ sind unterschiedliche Chiffrate c bekannt, die den gleichen Klartext p verschlüsseln. Aus den unterschiedlichen Chiffraten wird dieser Klartext p bestimmt.

IND-CPA-Angriff (Chosen-Plaintext-Attack): Ein polynomiell beschränkter Angreifer $\mathcal{A}$ wählt zwei Klartexte und schickt diese an ein Verschlüsselungsorakel. Das Verschlüsselungsorakel verschlüsselt zufällig einen der beiden Klartexte mit dem Schlüssel k und schickt das Chiffrat an $\mathcal{A}$. Er hat weiterhin Zugriff auf das Orakel. $\mathcal{A}$ darf höchstens einen nicht-vernachlässigbaren Vorteil gegenüber Raten haben ($0.5 + negl(|k|)$) das Chiffrat dem Klartext richtig zuzuordnen, andernfalls ist der Angriff erfolgreich [52]. Vernachlässigbar ist eine Funktion $negl$, wenn für alle Konstanten a ein N existiert, sodass für alle $n > N$ gilt: $negl(n) < n^{-a}$ [52, S.48].

4.2. Verfahren

Zur Verschlüsselung von Daten existieren zahlreiche historische Verfahren [11], von der griechischen Antike bis heute, darunter historisch bedeutsame Verfahren wie z.B. ADFGVX [54], VIC [92] und der sog. Doppelwürfel (double-column transposition) [34]. Moderne mensch-ausführbare Chiffren sind als »Hand Ciphers«und »Pen-And-Paper-Ciphers«bekannt. Neben diesen Chiffren stehen einfachere Methoden wie One-Time-Pads (OTPs) und Visuelle Kryptographie.

Visuelle Kryptographie bietet dabei unterstützt mensch-ausführbare Entschlüsselung. Die betrachteten Chiffren sind alle schwach mensch-ausführbar. Solitaire [84] wird mithilfe eines Kartendecks berechnet. Pocket-RC4 und Card-Chameleon [67] sind ebenfalls zur Nutzung eines Kartendecks ausgelegt und orientieren sich an RC4, ein Verfahren auf das praktische Angriffe existieren [3]. Mirdek - auch unter Verwendung eines Kartendecks und von Solitaire inspiriert - soll einen in Solitaire vorhandenen Bias vermeiden, ist jedoch nicht IND-CPA sicher [29]. Statt eines Kartendecks verwenden [69, 91] als Hilfsmittel Zauberwürfel, wobei der Schlüssel eine Folge von Rotationen darstellt.

Solitaire und Pocket-RC4 sind Stromchiffren. Der erzeugte Schlüsselstrom wird als OTP mit dem Klartext kombiniert. Handycipher [50] ist eine sog. homophone monoalphabetische Ersetzungschiffre (homophonic monoalphabetic substitution cipher) [51, S.3]. Jedes Zeichen des Klartexts wird dabei durch Zeichenfolgen ersetzt. Chaocipher wurde bereits 1918 von John F. Byrne entwickelt [82] und ist wie Card-Chameleon, Mirdek und LC4 eine Chiffre, in der der erzeugte Zeichenstrom vom Klartext und Schlüssel abhängt. Auf Chaocipher ist sowohl ein Known-Plaintext- als auch Ciphertext-Only-Angriff bekannt [33].

Nur LC4 hat den Anspruch, alle drei in Unterabschnitt 4.1.1 angegebenen Schutzziele zu erfüllen. Prinzipiell lässt sich Integrität und Authentifikation wie in LC4 (Protokoll 4.2.4) auch für andere Chiffren umsetzen, deren Schlüssel sich abhängig von Klartext und Chiffrat verändert [51, vgl. S.2 f.].

4.2.1. OTPs

One-Time-Pads (OTPs) bieten informationstheoretisch perfekte Sicherheit (im Sinne der Vertraulichkeit) durch die bit- oder zeichenweise Kombination des Chiffrats mit einem Schlüssel gleicher Länge (dem sog. One-Time-Pad) [60]. Bits werden für gewöhnlich durch eine XOR-Operation kombiniert, wobei jedes Paar $\langle E, D\rangle$ von Funktionen $E : P \times K \mapsto C, D : C \times K \mapsto P$ mit

- $D(E(x,k),k) = x$
- $k = k' \Leftrightarrow E(x,k) = E(x,k')$
- $k = k' \Leftrightarrow D(x,k) = D(x,k')$

verwendet werden kann. Für die Verschlüsselung von Text bestehend aus 26 Zeichen $P = C = \{a, ..., z\}$ mit $M : P \mapsto \mathbb{Z}_{26}, M(a) = 0, M(b) = 1, ...M(z) = 25$ bietet sich beispielsweise $E(x,k) = M^{-1}(M(x) + M(k))$ und $D(x,k) = M^{-1}(M(x) - M(k))$ an [1]. Zwar bieten OTPs perfekte Sicherheit, sie stellen aber weder die Integrität der Daten noch die Authentizität der Gegenpartei sicher [60]. Eine weitere Schwäche in der Praxis verbirgt sich bereits im Namen: Ein One-Time-Pad darf nur ein einziges Mal verwendet werden, denn die Kombination unterschiedlicher Chiffrate entspricht der Kombination der Klartexte und verrät dadurch relevante Informationen [60, S.4]. OTPs lassen sich trotz dieser Schwächen sinnvoll nutzen. Wird eine Menge von Schlüsseln persönlich (oder über andere sichere Wege) ausgetauscht in der Erwartung dass sich diese Option bei zukünftigem Bedarf an geheimer Kommunikation nicht mehr bietet, könnten OTPs eingesetzt werden. Ein weiteres praktisches Problem stellt die Länge des Schlüssels und dessen Generierung dar: Die Schlüsselgenerierung durch einen Computer in einer Mensch-Maschine-Kommunikation kann durch kryptographisch sichere Zufallsgeneratoren erfolgen, erfolgt die Kommunikation jedoch zwischen zwei Menschen ohne vertrauenswürdige technische Hilfsmittel, gestaltet sich die Generierung eines sicheren Schlüssels schwierig. Selbst wenn die sichere Schlüsselgenerierung gelingt, ohne den Schlüssel zu notieren, ist die rein im Kopf ausgeführte Ver- und Entschlüsselung selbst kurzer Nachrichten sehr anspruchsvoll. Es ist deshalb vernünftig davon auszugehen, dass OTPs zumindest zwischen zwei Menschen nicht streng mensch-ausführbar sind. Dass sie unterstützt mensch-Ausführbar sind, steht jedoch schon aufgrund der historischen Verwendung außer Frage. Bereits 1882 wurde ein One-Time-Pad Verfahren beschrieben [12]. Auch für schwache Mensch-Ausführbarkeit kann argumentiert werden, solange sich der Schlüssel gemerkt werden kann.

Der menschliche Aufwand der Ver- und Entschlüsselung einer Textnachricht soll nun ein wenig genauer betrachtet werden. Der Klartext und das Chiffrat bestehen aus $|m|$ Zeichen, wobei jedes Zeichen $z \in Z$ aus einer Zeichenmenge der Größe $|Z|$ besteht. $\langle E, D\rangle$ sind analog zur obigen Definition (dort für a bis z) definiert. Die Schlüsselgenerierung verursacht $|m|$ Zufallsexperimente mit $|Z|$ Möglichkeiten. Der Aufwand dieser hängt vom verfügbaren Hilfsmittel ab: Mit einer Münze sind zur Generation einer Zufallszahl $X \in \mathbb{Z}_{|Z|}$ $\lceil \log_2(|Z|) \rceil$ Wiederholungen nötig, während mit einem Würfel nur $\lceil \log_6(|Z|) \rceil$ benötigt werden. Das

[1] Diese Konstruktion lässt sich natürlich für beliebige Zeichen bilden.

Ergebnis kann als Zahl zur entsprechenden Basis interpretiert werden [2]. Aus diesem Grund wird die Größe der zu generierenden Zufallszahl nicht weiter betrachtet und der Aufwand allgemein mit t_r angegeben. Die Addition lässt sich etwas genauer betrachten: Es wird angenommen, dass Menschen zur Basis 10 rechnen. $t(x, y)$ steht für den Additionsaufwand zweier Zahlen $x, y \in \mathbb{Z}_{|Z|}$ beliebiger Größe. Jede übertragsfreie Addition zweier Zahlen x und y ($x + y < 10$) stellt einen Aufwand von t_+ dar. Der Aufwand für den Übertrag einer Zahl oder das Übernehmen einer Stelle ohne Addition soll mit t_o bezeichnet werden. Es gilt z.B. $t(3, 5) = t_+$, $t(8, 7) = t_+ + t_o$, $t(32, 9) = 2t_+ + t_o$, $t(23, 37) = 3t_+ + t_o$, $t(3, 21) = t_+ + t_o$ und $t(22, 11) = 2t_+$. Für $x, y \in \mathbb{Z}_{26}$ lässt sich der erwartete Aufwand mit o.B.d.A. $x \geq y$ als

$$\begin{aligned} E[t(x,y)] =& t_+ \cdot Pr[x + y < 10] \\ &+ (t_+ + t_o)(Pr[\sigma \geq 10 \wedge \lambda = 0] + Pr[\sigma < 10 \wedge \lambda =]) \\ &+ (2t_+ + t_o) \cdot Pr[\sigma \geq 10 \wedge \lambda = 1] \\ &+ 2t_+ \cdot Pr[\sigma < 10 \wedge \lambda = 2] \\ &+ (3t_+ + t_o) \cdot Pr[\sigma \geq 10 \wedge \lambda = 2] \\ =& \frac{1128t_+ + 441t_o}{676} \approx 1.669t_+ + 0.652t_o \end{aligned}$$

unter Gleichverteilungsannahme berechnen, wobei

$$\begin{aligned} \lambda := 0 \Leftrightarrow & x < 10 \wedge y < 10 \\ \lambda := 1 \Leftrightarrow & x \geq 10 \wedge y < 10 \\ \lambda := 2 \Leftrightarrow & x \geq 10 \wedge y \geq 10 \end{aligned}$$

und

$$\sigma := (x \bmod 10) + (y \bmod 10)$$

Um den Aufwand der OTP-Berechnung angeben zu können, fehlt noch die Betrachtung der Modulo-Operation. Diese wird nur benötigt, wenn $x + y > |Z|$, was für $|Z| = 26$ unter Gleichverteilungsannahme mit Wahrscheinlichkeit 0.5 vorkommt. In diesen Fällen reicht eine einfache Subtraktion $x + y - |Z|$ um die gewünschte Zahl zu erhalten. Wenn für Subtraktionen der gleiche Aufwand wie für Additionen angenommen wird (und $|Z| < 50$), kann obige Formel wiederverwendet werden, um für $|Z| = 26$ $E[t(x, |Z| - 1)] = \frac{53t_+ + 16t_o}{26} \approx 2.15t_+ + 0.69t_o$ zu erhalten. Insgesamt ergibt sich somit der erwartete Aufwand für das Entschlüsseln eines Zeichens $z \in Z, |Z| = 26$:

$$t_{OTP} := \frac{1128t_+ + 441t_o}{676} + 0.5 \cdot \frac{53t_+ + 16t_o}{26} = \frac{1817t_+ + 649t_o}{676} \approx 2.688t_+ + 0.96t_o$$

Für $t_+ = t_o = 0.5s$ ergibt sich für eine Nachricht der Länge $|n| = 200$ ein Aufwand von $364s \approx 6\text{min}$.

[2] Falls die entstandene Zahl größer als $|Z|$ ist, muss das gesamte Experiment wiederholt werden. Die Wahrscheinlichkeit hierfür liegt für Basis B bei $1 - \frac{|Z|}{B^{\lceil \log_B(|Z|) \rceil}}$

4.2.2. Visuelle Kryptographie

Visuelle Kryptographie (VC) kann als eine besondere Art von OTPs verstanden werden, in denen ein Geheimnis in Form eines Bildes in $n \geq 2$ Transparente aufgeteilt wird, von denen k überlagert das geheime Bild erzeugen (sog. (k,n)-Schema) [13, 44, 27]. Das Verfahren wurde zuerst von Naor und Shamir beschrieben [73]. Wie andere OTPs bietet VC perfekte Sicherheit, kommt jedoch bei naiver Anwendung mit den gleichen Einschränkungen [27]. Der Vorteil gegenüber diesen liegt in der einfachen und schnellen Entschlüsselung, die keinerlei Berechnung und technische Hilfsmittel verlangt [27]. Die Verschlüsselung, also die Erstellung der Transparente, ist hingegen im Allgemeinen zu aufwändig und fehleranfällig für menschliche Erzeugung. Zur Entschlüsselung muss der Schlüssel in Form eines Transparents vorliegen, wodurch das Verfahren in den Bereich der unterstützten Mensch-Ausführbarkeit fällt.

Naor et al. [72] stellen mehrere Verfahren vor, die einmalig authentifizierte Verschlüsselung ermöglichen sollen und ein Verfahren, in dem der Schlüssel mehrfach verwendet werden darf. Um den Schlüssel mehrfach verwenden zu können, wird pro Verschlüsselung nur ein Teil des Transparents für die Nachricht verwendet und der Rest des Transparents so erstellt, dass Manipulationen in diesem Bereich sichtbar werden.[3] Die Abfolge der Bereiche ist geheim [72, S.331 f.].

Der Anwendungsbereich im Kontext mensch-ausführbarer Kryptographie beschränkt sich dadurch auf die Kommunikation zwischen Mensch und Maschine. Als Anwendungsbereiche nennen [44] u.A. digitale Wasserzeichen, Stegonographie und Authentifikationsverfahren und [78] elektronische Wahlen und Online Banking. Diese greifen jedoch teilweise auf für Mensch-Ausführbarkeit unzulässige technische Hilfsmittel zurück, wie beispielsweise mobile Endgeräte mit Kamera [28] oder VR-Brillen [32]. Zudem muss im von Chow et al. [28, S.248 f.] beschriebenen Verfahren die sich identifizierende Partei ein eigenes Transparent erstellen. Dazu nutzt sie ein sich nicht veränderndes Geheimnis und einen mit jedem Identifikationsversuch zufällig generierten Zufall. Alleine die Erstellung eines Transparents übersteigt dabei die Mensch-Ausfürbarkeit.

Praktische Erwägungen Das ursprünglich in [73] beschriebene (k,n)-Schema leidet an schlechtem Kontrast und verwendet Pixelexpansion [44]. Bei Pixelexpansion wird ein einzelnes Pixel im geheimen Bild durch mehrere Pixel in den Transparenten (üblicherweise 2x2) dargestellt [44]. In ihrer Übersicht betrachten Ibrahim et al. [44] unter anderem die notwendige Größe der Transparente, Verzerrungen, Kontrast bzw. Qualität der entschlüsselten Bilder. Ein allgemeines praktisches Problem von VC stellt die exakte Überlagerung der physischen Transparente dar. Dieses Problem könnte durch Vorrichtungen zur exakten Platzierung vermindert werden [28, S.250] [27]. Alternativ ist auch segmentbasierte VC (SBVC) einfacher für Menschen zu erkennen, kann aber dafür nur eine kleine Menge von Symbolen verschlüsseln [44, S.31933]. Größen-Invariante VC (SIVC) vermeidet Pixelexpansion und bietet damit den Vorteil kleinerer Transparente [44, S.31933] Neben Schwarz-Weiß-Bildern können auch Graustufen- oder Farbbilder in VC verwendet werden (CVC). Eine Möglichkeit dazu besteht im Halftoning: Halftoning beschreibt die Simulation

[3] Im Beipspiel von Naor et al. sollen alle 2x2 Blöcke außerhalb des verwendeten Bereichs genau 2 schwarze Pixel beinhalten

eines stetigen Farbspektrums durch Farbpunkte. Dies wird bei Farb- und Graustufenbildern verwendet, um sie in ein binäres Format zu bringen [44, S.31935].

Der Informationsgehalt, der über VC kommuniziert werden kann, ist in der Praxis durch die Größe der Transparente und Bildschirme beschränkt. Wird auf einem Full-HD Bildschirm (1920x360) ein Schwarz-Weiß VC mit 3x3 Pixel-Expansion zur einfacheren Überlagerung verwendet, könnten $640 \cdot 360 = 230400$ Bit Information dargestellt werden.

4.2.3. Solitaire

Solitaire [84] ist eine durch ein Kartendeck mit 54 Karten (2 Joker) erzeugte Stromchiffre. Jede Karte im Deck hat einen eindeutigen Wert $v \in \{1, ..., 52, A, B\} = V$. In der Praxis kann die Wertigkeit der Farbe für die Berechnung des Werts verwendet werden. Es bietet sich an, für A und B die beiden Joker zu verwenden. Diese müssen unterscheidbar sein. Mit $ix(v)$ mod 54 wird die Position der Karte mit Wert v im Stapel bezeichnet, wobei 1 die oberste Position ist. Der Schlüssel zur Generierung des Zeichenstroms ist der Zustand des Stapels, also die Permutation der 54 eindeutigen Zeichen aus V.

Cipher 1 Solitaire [84]

Generierung des Zeichenstroms:

1. Geg. $j = ix(A)$, $j{+}1 = ix(z)$: Finde Karte A und tausche sie mit der darunterliegenden Karte: $ix(A) = j + 1$, $ix(z) = j$.

2. Geg. $j = ix(B)$, $j + 1 = ix(z_1)$, $j + 2 = ix(z_2)$: Finde Karte B und verschiebe sie um zwei Karten nach unten: $ix(B) = j + 2$, $ix(z_2) = j + 1$, $ix(z_1) = j$.

3. Teile den Stapel anhand der Karten A und B in drei Blöcke $L, M, R \subset V$, $x_1, x_2 \in \{A, B\}$ $\langle L, x_1, M, x_2, R\rangle$ und vertausche L und R: $\langle R, x_1, M, x_2, L\rangle$ D.h. tausche die Karten über dem ersten »Joker«(A oder B) mit den Karten unter dem zweiten.

4. »Count cut«: Betrachte den Wert z der untersten Karte ($z \in Z$ mit $ix(z) = 54$). Wenn $z \in \{A, B\}$ setze $z = 53$ in diesem Schritt. Teile den Stapel in zwei Blöcke, (wobei die unterste Karte nicht betrachtet wird): $\langle L, R, z\rangle$ mit $|L| = z$ und vertausche diese: $\langle R, L, z\rangle$ [4]

5. Betrachte den Wert z der obersten Karte ($z \in Z$ mit $ix(z) = 0$). Wenn $z \in \{A, B\}$ setze $z = 53$. Sei $ix(z_r) = z$ (z_r ist die Karte an Position z). Wenn $z_r \in \{A, B\}$ gehe zu Schritt 1. Ansonsten: **Gebe** z_r **mod** 26 **aus** [5].

6. Gehe zu Schritt 1.

---

[4] Der Grund dafür, dass die letzte Karte an gleicher Stelle bleibt, ist die Reversibilität [84].

[5] Die Modulo-Operation kann eingespart werden, wenn die Ausgabe zwischen 1 und 52 liegen soll [88]. Für die Generierung von Buchstaben bieten sich Zahlen zwischen 1 und 26 (a bis z) allerdings besser an.

Aufwand In dieser Betrachtung wird vereinfachend für alle Wahrscheinlichkeiten eine Gleichverteilung angenommen. Der Aufwand im ersten Schritt besteht aus dem Finden der Karten A und B, sowie dem anschließenden Vertauschen nach hinten um ein bzw. zwei Karten. Da der zeitliche Unterschied des Verschiebens zwischen einer und zwei Karten vernachlässigbar ist, bezeichne t_s das Verschieben einer der beiden Karten und t_f das Finden beider Karten. Aus den ersten beiden Schritten sind die Positionen von A und B noch bekannt, es fällt im 3. also nur ein Vertauschen der Blöcke t_{sb} an. Anschließend muss im 4. die unterste Karte betrachtet werden (kein Aufwand) und z Karten mit Erwartungswert

$$E[z] = \frac{53 + \sum_{n=1}^{53} n}{54} \approx 27.48$$

mit jeweils Aufwand t_c abgezählt werden. Erneut werden zwei Blöcke vertauscht (t_{sb}). Für den 5. Schritt werden ebenfalls z Karten mit $E[z] = 27.48$ abgezählt und mit einer Wahrscheinlichkeit von $p = 26/27$ eine Zahl mithilfe einer Modulo-Operation t_{mod} ausgegeben. Wenn t_σ also den Aufwand eine Zahl im Strom zu erzeugen bezeichnet, gilt:

$$E[t_\sigma] \approx \frac{27}{26}(2t_s + t_f + 2t_{sb} + 54.96t_c) + t_{mod}$$

Mit $t_f = 5s, t_s = t_{sb} = 2s, t_c = 0.25s, t_{mod} = 2s$, braucht die Generierung von 200 Zeichen zur Ver- bzw. Entschlüsselung einer Nachricht gleicher Länge also $200 \cdot E[t_\sigma] \approx 5953.7s \approx 99$min. Die generierten Zeichen werden wie OTPs mit dem Klartext oder Chiffrat kombiniert.

Sicherheit Mit $54! \approx 2.31 \cdot 10^{71} \approx 2^{237}$ möglichen Anordnungen des Kartendecks bietet Solitaire ausreichende Sicherheit vor Brute-Force-Angriffen. Bislang wurde kein praktisch durchführbarer Angriff veröffentlicht, es sind aber sicherheitsbezogene Probleme bekannt [88, 30]. Die Ausgabewahrscheinlichkeit der einzelnen Zeichen ist zwar gleichverteilt, allerdings besteht ein Bias bei aufeinanderfolgenden Zeichen: Die Wahrscheinlichkeit dass auf ein Zeichen das gleiche Zeichen folgt, sollte in einer zufälligen Folge bei 1/26 liegen, liegt jedoch in Solitaire bei etwa 1/22.5 [88, 30]. Tatsächlich besteht ein solcher Bias auch zum n-ten folgenden Zeichen, fällt aber deutlich geringer aus [88, S.5, Abb.]. 3-Tupel und 4-Tupel sind hingegen seltener als in einer zufälligen Folge [88]. Aus dem Hauptbias folgt, dass nach dem ersten generierten Zeichen für jedes weitere Zeichen 0.0005 Bits an Information leakt [88, S.5 f.]. Shiu [88] gibt zwei beispielhafte Szenarien an, in denen der vorhandene Bias ausgenutzt werden kann. Beide Szenarien gehen von der Verschlüsselung langer Zeichenketten ($|m| = 10000$) (bzw. wiederholte Verschlüsselung $\sum |m| = 50000$) aus. Für kurze Nachrichten, wie in [84] vorgesehen, lässt sich dieser Bias in den Szenarien nicht ausnutzen.

Praktische Erwägungen Als praktischer Vorteil von Solitaire gegenüber OTPs muss die kürzere Länge des Schlüssels genannt werden. Zwar ließe sich dieser besser merken, darf jedoch offensichtlich nicht mehrfach verwendet werden: Ein gleicher Schlüssel führt zum gleichen Zeichenstrom, der als OTP nur ein einziges Mal verwendet werden darf. Gleichzeitig kommt Solitaire mit einem gravierenden Nachteil für menschliche Nutzung. Ein Fehler in der Verschlüsselung des i-ten Zeichen führt dazu, dass der entschlüsselte Klartext

ab dem i-ten Zeichen nicht mehr lesbar ist. Um solche Fehler in der Verschlüsselung zu erkennen, kommt in Betracht, das Verfahren mehrfach auszuführen und Ergebnisse abzugleichen.

4.2.4. LC4

LC4 [51] ist ein unter Verwendung von Spielsteinen schwach mensch-ausführbares Verschlüsselungsverfahren, das die Klartextzeichen anhand eines sich verändernden Zustands verschlüsselt. Der Schlüssel ist der initiale Zustand. Die Zustandsänderungen hängen je vom Klartext- und Chiffratzeichen ab, wodurch authentifizierte Verschlüsselung ermöglicht werden soll. $\div$ bezeichnet im Folgenden die restlose Ganzzahldivision.

Cipher 2 LC4 (unauthentifiziert) [51, S.4 ff.]

Verschlüsselung: Input: Zustandsmatrix $S = (c_0|...|c_5) = (r_0|...|r_5)^T$ (c die Spalten, r die Zeilen), Indizes i, j, Klartext $m = (p_1, ..., p_{|m|})$ bestehend aus $|m|$ Zeichen
Für $k = 1...|m|$ (p_k das k-te Klartext-Zeichen)

1.
   - $r \leftarrow \arg\, j : p_k \in r_j$ (Zeile in der p_k steht)
   - $c \leftarrow \arg\, j : p_k \in c_j$ (Spalte in der p_k steht)
   - $x \leftarrow (r + (S[i][j] \div 6)) \bmod 6$ (Zeile in der das Chiffrat von p_k steht)
   - $y \leftarrow (c + S[i][j]) \bmod 6$ (Spalte in der das Chiffrat von p_k steht)
   - $C \leftarrow S[x][y]$
2. Gebe C aus.
3. Update der Zustandsmatrix und Indizes:
   - Rechtsrotation der Zeile r:
     - $(S[r][0], S[r][1], S[r][2], S[r][3], S[r][4], S[r][5]) \leftarrow$
       $(S[r][5], S[r][0], S[r][1], S[r][2], S[r][3], S[r][4])$
     - $c \leftarrow (c + 1) \bmod 6$
     - Falls $x = r$: $y \leftarrow (y + 1) \bmod 6$ (Wenn Klartext- und Chiffratspalte übereinstimmen)
     - Falls $i = r$: $j \leftarrow (j + 1) \bmod 6$
   - Verschiebung der Spalte y nach unten:
     - $(S[0][y], S[1][y], S[2][y], S[3][y], S[4][y], S[5][y]) \leftarrow$
       $(S[5][y], S[0][y], S[1][y], S[2][y], S[3][y], S[4][y])$
     - Falls $j = y$: $i \leftarrow (i + 1) \bmod 6$
   - $i \leftarrow (i + (C \div 6)) \bmod 6$
   - $j \leftarrow (j + (C \bmod 6)) \bmod 6$

Entschlüsselung: Input: Zustandsmatrix $S = (c_0|...|c_5) = (r_0|...|r_5)^T$ (c die Spalten, r die Zeilen), Indizes i, j, Chiffrat-Folge $C = (c_1, ..., c_{|m|})$
Für $k = 0...|m|$ (c_k das k-te Chiffrat-Zeichen)

1.
   - $x \leftarrow \arg\, j : c_k \in r_j$ (Zeile in der c_k steht)
   - $y \leftarrow \arg\, j : c_k \in c_j$ (Spalte in der c_k steht)
   - $r \leftarrow (x - (S[i][j] \div 6)) \bmod 6$ (Zeile in der das Chiffrat von p_k steht)
   - $c \leftarrow (y - (S[i][j] \bmod 6)) \bmod 6$ (Spalte in der das Chiffrat von p_k steht)
2. Gebe $P \leftarrow S[r][c]$ aus
3. Update der Zustandsmatrix und Indizes wie in der Verschlüsselung definiert.

Die Zustandsmatrix wird in der Entschlüsselung auf gleiche Weise wie in der Verschlüsselung verändert. Der einzige Unterschied liegt darin, dass die »Bewegung«(siehe nächster Absatz) in der Matrix rückwärts ausgeführt wird, um ein Zeichen zu entschlüsseln.

Aufwand und Berechnung durch Menschen Ein Mensch kann das einfache Verfahren unter Zuhilfenahme von 36 als Matrix S geordneten Spielsteinen ausführen, die neben dem Zeichen z auch die Werte $\text{move}_r := z \bmod 6$ und $\text{move}_t = z \div 6$ tragen. Zusätzlich ist eine Pinnadel oder andere Markierung für die Indizes i, j hilfreich [51, vgl. S.12 f.]. Soll nun ein Klartext-Zeichen verschlüsselt werden, muss zunächst der Spielstein mit diesem Zeichen gesucht werden (Aufwand: t_s). Anschließend werden ausgehend von diesem Spielstein move_r Schritte nach rechts und move_t nach unten gegangen (Aufwand t_m pro Schritt, erwarteter Aufwand: $5t_m$), wobei dafür move_r und move_t vom gepinnten Spielstein abgelesen werden. Das Zeichen, auf dem man landet, ist das Chiffrat des Klartext-Zeichens und kann als solches notiert werden. Nun wird die Zeile mit dem Klartext-Zeichen um eine Stelle nach rechts rotiert und die Spalte mit dem Chiffrat-Zeichen eine Stelle nach unten (Aufwand $2t_{\text{rot.}}$). Sollte sich der gepinnte Stein in dieser Zeile oder Spalte befinden, bewegt sich der Pin mit. Um den Pin zu verschieben, werden move_r und move_t des Steins mit dem Chiffrat-Zeichen betrachtet: Ausgehend vom gepinnten Stein werden move_r Schritte nach rechts und move_t nach unten gegangen (erwarteter Aufwand wie zuvor) und der Pin auf den Stein gesteckt, auf dem man landet. Für die Verschlüsselung weiterer Zeichen wiederholt sich der Vorgang. Der erwartete Aufwand für das Verschlüsseln einer Nachricht der Länge $|m|$ liegt also bei $|m|(10t_m + 2t_{\text{rot.}} + t_s)$. Mit $t_m = 0.2s, t_{\text{rot.}} = 2s, t_s = 4s$ und $|m| = 200$ gelingt die Verschlüsselung in $2000s \approx 33\text{min}$. Für die Entschlüsselung kann der gleiche Aufwand angenommen werden, da sie sich nur in einem Punkt unterscheidet: Statt ausgehend von einem Klartext-Zeichen Schritte nach rechts/unten zu gehen um beim Chiffrat zu landen, werden die gleichen Schritte nach links/oben gegangen, um beim Klartext-Zeichen herauszukommen. Durch das im Folgenden vorgestellte authentifizierte Verschlüsseln von Nachrichten kommt ein konstanter zusätzlicher Aufwand durch das Verschlüsseln von mindestens 16 weiteren Zeichen sowie dem Generieren einer Zufallszahl (Länge ≥ 6) hinzu. Mit $t_r = 5s$ erhöht sich der zuvor beschriebene Aufwand auf $\geq 2000s + 16 \cdot 10s + 6 \cdot 5s = 2190s = 36.5\text{min}$.

Das Update der Zustandsmatrix ist abhängig vom Klartext und Chiffrat, was authentifizierte Verschlüsselung ermöglicht [51, S.5]. Um eine authentifizierte Nachricht zu verschicken, schlägt [51] folgendes Protokoll vor: $Enc(S, m) = (S', c)$ sei dabei der oben definierte Verschlüsselungsalgorithmus, der neben dem Chiffrat c auch den entstandenen Zustands S' (bestehend aus Zustandsmatrix und Indizes i, j) ausgibt. Analog entspricht $Dec(S, c) = (S', m)$ der Entschlüsselung. $a||b$ soll für die Konkatenation zweier Folgen a und b stehen.

Protokoll 4 LC4 [51, S.6]

Geg: Gemeinsames Geheimnis als Zustandsmatrix S, geheimer MAC *mac* als eindeutig Zeichenfolge des Senders definierter Länge (≥ 10), (optional) Header-Daten h die sowohl beim Ver- als auch Entschlüsseln bekannt sein müssen.
Verschlüsselung einer Nachricht m:

1. Generiere zufällige Nonce r definierter Länge (≥ 6)
2. Berechne $Enc(S, r) = (S', c_r)$ (verwerfe c_r)
3. Falls h vorhanden, berechne $Enc(S', h) = (S'', c_h)$ (verwerfe c_h). Ansonsten $S'' = S'$
4. Berechne $Enc(S'', m||mac) = (S^*, c)$ (verwerfe S^*)
5. Gebe (c, r) aus.

Entschlüsselung eines Chiffrats (c, r)*:*

1. Berechne $Enc(S, r) = (S', c_r)$ (verwerfe c_r)
2. Falls h vorhanden, berechne $Enc(S', h) = (S'', c_h)$ (verwerfe c_h). Ansonsten $S'' = S'$
3. Berechne $Dec(S'', c) = (S^*, m||mac')$ mit $|mac'| = |mac|$ (verwerfe S^*)
4. Falls $mac' \neq mac$: Abbruch
5. Gebe m aus.

4.2.4.1. Sicherheit

Eine Permutation von 36 Zeichen führt zu $36! \approx 2^{138}$ möglichen Schlüsseln. Die Indizes bieten keine zusätzliche Sicherheit. Prinzipiell ist auch eine Erweiterung auf 49 zw. 64 Zeichen in einer 7x7 bzw. 8x8-Zustandsmatrix möglich, mit $\approx 2^{208,56}$ bzw. 2^{296} Schlüsseln.

Statistik Eine durch [51] durchgeführte statistische Analyse zur relativen Häufigkeit eines Zeichens an fester Stelle im Chiffrat bei zufälligem Schlüssel kommt zum Ergebniss, dass bereits eine Nonce der Länge 3 reicht, um an beliebiger Stelle im Chiffrat Gleichverteilung annehmen zu können.[6] [51, S.8]. Zusätzlich untersucht [51] empirisch die Wahrscheinlichkeit einer Falsch-Positiven Authentifizierung bei Veränderung eines einzelnen Chiffrat-Zeichens an beliebiger Stelle, wobei dabei alle möglichen Änderungen pro Chiffrat probiert wurden. In Abhängigkeit von der Länge des MAC $|mac|$, liefert dieses Experiment (10^6 Wiederholungen, Länge Nonce $|r| = 10$, Klartext-Länge $|m| = 100$) für $|mac| = 4$ $p = 1.1587 \cdot 10^{-6}$ und für $|mac| = 10$ $p = 5.665 \cdot 10^{-9}$. [51, S.9, §Tab.2]. Interessanterweise liegen diese Ergebnisse für $|mac| = 10$ deutlich über $(\frac{1}{36})^{10} \approx 2.735 \cdot 10^{-16}$: Das

[6] Der Klartext wurde aus der Zeichenverteilung (Bigramm-Häufigkeit) der englischen Sprache gezogen.

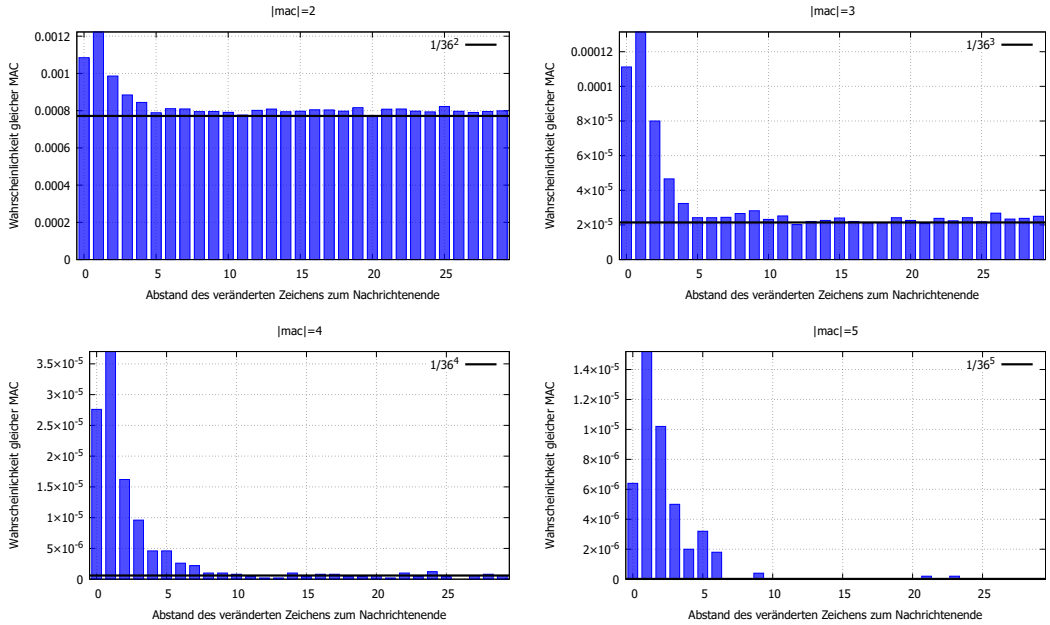

Abbildung 4.1.: Y-Achse: Empirische Wahrscheinlichkeit eines gleichen MAC für zufällige Schlüssel und zufällige Nachricht. Y-Achse: Abstand zwischen Position des veränderten Zeichens im Klartext und Beginn des MAC. (0: Das letzte Zeichen der Nachricht wurde verändert.) Zusätzlich jeweils die ideale Wahrscheinlichkeit $(\frac{1}{36})^{|mac|}$.

Ändern eines einzelnen Chiffrat-Zeichens verändert die letzten verschlüsselten Zeichen seltener, als es bei zufälliger Veränderung erwartbar wäre. Zwar wurde das Experiment bis zu MAC-Längen von $|mac| = 30$ durchgeführt, konnte jedoch ab $|mac| = 20$ keine derartigen MAC-Kollisionen mehr feststellen [51, S.9, §Tab.2].

Eine Erklärung für den Bias könnte sein, dass Folgezustände in dem Sinn ähnlich zum vorherigen Zustand sind, dass die relativen Positionen der meisten Zeichen zueinander im Zustand unverändert bleiben. Diese relativen Positionen zueinander spielen für die Verschlüsselung aber eine große Rolle. Für einen festen Zustand S existieren 36 mögliche Folgezustände (ein Folgezustand pro Klartextzeichen). Es gibt also 36^n mögliche Zustände für den n-ten Zustand nach S. Um also jeden möglichen Zustand erreichen zu können, müssen mindestens $\lceil log_{36}(36!) \rceil$ Zeichen verschlüsselt werden. Tatsächlich zeigt sich, dass ein großer Teil des Bias durch die Veränderung von Zeichen unmittelbar vor dem MAC erklärt werden kann (siehe Abbildung 4.1). Die letzten Zeichen der Nachricht m sind anfälliger für unbemerkte Manipulation. Als mögliche Gegenmaßnahme kann zwischen der Nachricht m und dem MAC ein zufälliges Padding eingefügt werden.

Konkrete Abhängigkeiten Mit $p = (p_0, ...)$ wird in diesem Abschnitt der Klartext bezeichnet und mit $c = (c_0, ...)$ das Chiffrat. Zunächst lässt sich beobachten, dass die Wahl der Indizes i, j keine zusätzliche Sicherheit bieten, da sich die Zustandsmatrix S immer so normalisieren lässt, so dass $i = j = 0$. Dafür wird die gesamte Matrix entsprechend verschoben (siehe Abbildung 4.2). Die Indizes dienen also ausschließlich der Mensch-Ausführbarkeit.

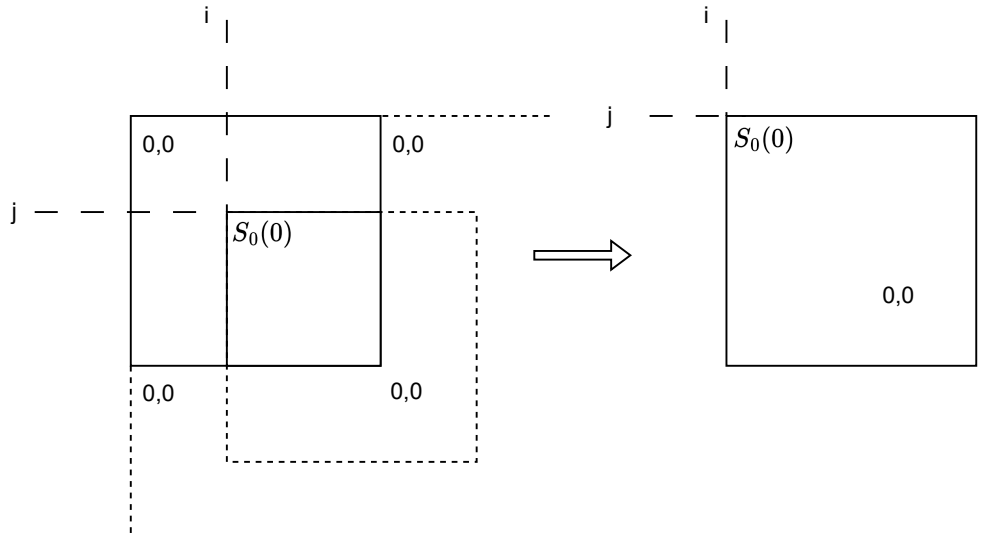

Abbildung 4.2.: Normalisierung des Zustands S, um die Indizes i, j zu entfernen.

$S_i(0) \in \{0, ..., 35\}$ soll nun den Wert des Zeichens links-oben in der normalisierten Zustandsmatrix im i-ten Verschlüsselungsschritt bezeichnen [7]. Dieser Wert gibt an, wie viele Schritte vom Klartextzeichen p_i zum Chiffratzeichen c_i gegangen wird ($S_i(0)$ mod 6 nach rechts und $S_i(0) \div 6$ nach unten). Offensichtlich gilt $S_i(0) = 0 \Leftrightarrow p_i = c_i$. (*Erinnerung:* Das Update von $S_i(0)$ erfolgt indem von $S_i(0)$ aus c_i mod 6 nach rechts und $c_i \div 6$ Schritte nach unten gegangen werden.)

Zwar stellte Kaminsky [51] einen Bias in der einfachen Verschlüsselung der ersten Zeichen fest, untersuchte jedoch nicht die Ursachen dafür. Um diesen initialen Bias zu vermeiden, wurde im Verschlüssungsprotokoll eine Nonce der Länge ≥ 6 vorgeschlagen. Dies reicht jedoch nicht, um Rückschlüsse auf den Klartext zu verhindern. Untersucht man die Verschlüsselung von zwei Zeichen $p_0, p_1 \in \{0, ..., 35\}$ zu Chiffratzeichen c_0, c_1 ($Enc(S, p_0|p_1) = (., c_0|c_1)$), mit festen p_0, p_1 und zufälligem Schlüssel S, kann die bedingte Verteilung $Pr[c_1 = X|c_0 = Y \wedge p_0 = Z_0 \wedge p_1 = Z_1]$ betrachtet werden. Daraus lassen sich zunächst folgende Beobachtungen ziehen:

1. $c_0 = p_0 = 0 \Rightarrow p_1 = c_1$
2. $c_0 = 0 \wedge p_0 \neq 0 \Rightarrow c_1 \neq p_1$
3. $c_0 = p_0 \neq 0 \Rightarrow c_1 \neq p_1$

Die Aussagen der Beobachtungen lassen sich beweisen:

1. Beobachtung: Aus $c_0 = p_0$ folgt $S_0(0) = 0$. Wegen $c_0 = 0$ ist $S_1(0) = S_0(0) = 0$, woraus direkt $p_1 = c_1$ folgt.

2. Beobachtung: Aus $c_0 = 0$ folgt $S_1(0) = S_0(0)$. Da $c_0 = 0 \neq p_0$ muss $S_0(0) \neq 0$ und somit auch $S_1(0) \neq 0$. Hieraus folgt $c_1 \neq p_1$.

[7] Entspricht dem Wert des »gepinnten Steins«

3. Beobachtung: Aus $c_0 = p_0$ folgt $S_0(0) = 0$. $S_1(0) \neq S_0(0)$ folgt aus $c_0 \neq 0$, weshalb $S_1(0) \neq 0$ und somit $c_1 \neq p_1$.

Diese Aussagen gelten nicht nur am Anfang des Chiffrats sondern über das gesamte Chiffrat hinweg. Es wurden keine Annahmen über den Anfangszustand getroffen. Die Argumentation bleibt also gültig, wenn statt c_0, c_1, p_0, p_1 Zeichen $c_i, c_{i+1}, p_i, p_{i+1}$ in einem Chiffrat und Klartext beliebiger Länge $> i$ betrachtet werden. Die Konstruktion des Protokolls bringt keine Abhilfe des Problems.

Um zu untersuchen, wie viel Information dadurch ein Chiffrat über den Klartext verrät, wird nun gegeben einem Chiffrat $(c_0, ..., c_n)$ unabhängig vom Schlüssel die Anzahl der gemäß den Beobachtungen unmöglichen Klartexte bestimmt. Dafür ist es notwendig, Nullen besonders zu berücksichtigen. Um die Untersuchung zu vereinfachen, wird von maximal einer Null im Klartext ausgegangen und nur vergleichsweise kurze Nachrichten (Länge ≤ 36) betrachtet. Zusätzlich wird in diesem Abschnitt vereinfachend von einer Gleichverteilung der Chiffrate ausgegangen. Die Anzahl der zum Chiffrat einfach[8] ausschließbaren Klartexte lässt sich über mehrere rekursive Folgen bestimmen, wobei Zeichen immer paarweise überlappend betrachtet werden. $M_c(k) \subset \{0, ..., 36\}^{n+1}$ soll die Menge der einfach ausschließbaren Klartexte bezeichnen, wenn dafür nur die ersten $k+2$ Zeichen $(c_0, ..., c_k, c_{k+1})$ berücksichtigt werden. Es gilt $M_c(0) \subset M_c(1) \subset ... \subset M_c(n-1)$

Beispiel Für ein einzelnes Chiffrat `ax0ca`: `axf8l` $\in M_c(0)$, `5x0db` $\in M_c(1)$. Dies folgt daraus, dass keine zwei Zeichen, von denen das erste keine Null ist, in Chiffrat und Klartext übereinstimmen können.

$I_c(k) := |M_c(k)| - |M_c(k-1)|$ bezeichnet die Anzahl der einfach ausschließbaren Klartexte zum Chiffrat, wenn nur die Zeichen c_k, c_{k+1} betrachtet werden und die in vorherigen Paaren berücksichtigten Eliminationen nicht erneut gezählt werden. Ist $c_0 \neq 0$ dann gilt $I_c(0) = 36^{n-1}$, denn alle Klartexte die mit c_0, c_1 anfangen, können nicht zum Chiffrat gehören. Beginnt das Chiffrat mit einer Null ($c_0 = 0$), dann gilt $I_c(0) = (35+35) \cdot 36^{n-1}$, da 70 der 36^2 möglichen zwei ersten Zeichen des Klartexts nicht zum Chiffrat gehören können. Für $k > 0$ gilt:

$$I_c(k) = \begin{cases} 36^{n-1} - \frac{I_c(k-1)}{36} - \sum_{i=0}^{k-2} \frac{I_c(i)}{36^2} & \text{für } c_k \neq 0 \wedge c_{k-1} \neq 0 \text{ (I)} \\ (35+35) \cdot 36^{n-1} - 35 \cdot \frac{I_c(k-1)}{36} - (35+35) \cdot \sum_{i=0}^{k-2} \frac{I_c(i)}{36^2} & \text{für } c_k = 0 \wedge c_{k-1} \neq 0 \text{ (II)} \\ 36^{n-1} - \star - \sum_{i=0}^{k-2} \frac{I_c(i)}{36^2} & \text{für } c_k \neq 0 \wedge c_{k-1} = 0 \text{ (III)} \end{cases}$$

$$\star := 35(36^{n-2} - \sum_{i=0}^{k-3} A(i)) \text{ mit } A(0) = 36^{n-4}, A(i) = 36^{n-4} - \frac{A(i-1)}{36} - \sum_{j=0}^{i-2} \frac{A(j)}{36^2}$$

(I) beschreibt die Situation, in der $c_k \neq 0$. Alle Klartexte mit $p_k = c_k, p_{k+1} = c_{k+1}$, also 36^{n-1} Möglichkeiten, fallen weg. Das Ergebnis muss berücksichtigen, dass manche der Möglichkeiten bereits zuvor betrachtet wurden. Da im vorherigen Schritt nur ein anderes Zeichen fixiert ist (nämlich c_{k-1}), wurden $\frac{I_c(k-1)}{36}$ der Eliminationen zuvor erfasst. In allen Schritten vor diesem Schritt waren zwei Zeichen fixiert, wodurch jeweils $\frac{I(k-1)}{36^2}$ der jetzt betrachteten Möglichkeiten bereits erfasst wurden.

[8] Gemäß den Beobachtungen

(II) beschreibt die Situation, in der $c_k = 0$. Alle Klartexte mit $p_k = c_k, p_{k+1} \neq c_{k+1}$ und $p_k \neq c_k, p_{k+1} = c_{k+1}$, also $(35 + 35)36^{n-1}$ Möglichkeiten, fallen weg. Auch hier muss analog zum ersten Fall berücksichtigt werden, dass manche der Möglichkeiten bereits zuvor betrachtet wurden. Im vorherigen Schritt fallen nur 35 statt 70 Möglichkeiten für p_k, p_{k+1} weg, da das Zeichen c_k in $I_c(k-1)$ fixiert ist.

(III) Nachdem eine Null betrachtet wurde, muss im nächsten Schritt die Anzahl der bereits betrachteten Klartext-Möglichkeiten anders berechnet werden (★). Ist $c_k \neq 0$, dann sind im vorherigen Schritt nur die Möglichkeiten zu betrachten, in denen $c_k = p_k$, womit nur $c_{k-1} \neq 0$ für die Eliminationen betrachtet wurden. Dies sind 35^{n-2} Möglichkeiten, die zwar in $I_c(k-1)$ berücksichtigt wurden, nicht aber im Schritt $I_c(k-2)$, da in diesem Schritt $c_{k-1} = 0$ fixiert war. In vorherigen Schritten werden jeweils die Möglichkeiten mit $c_{k-1} \neq 0$ berücksichtigt, wobei die Stellen k und $k+1$ nicht zu berücksichtigen sind, da diese in $I_c(k)$ fixiert sind. Es sind also $(n-1)-2-1$ Stellen mit 36 Zeichenmöglichkeiten und eine Stelle mit 35 Zeichenmöglichkeiten zu betrachten. Um auch hier keine Möglichkeiten doppelt zu zählen, wird die rekursive Folge $A(0) = 36^{n-4}, A(i) = 36^{n-4} - \frac{A(i-1)}{36} - \sum_{j=0}^{i-2} \frac{A(j)}{36^2}$ analog zum ersten Fall von $I_c(k)$ definiert und verwendet. Dadurch dass vereinfachend maximal eine Null im Chiffrat angenommen wird, muss die Null in $A(i)$ nicht berücksichtigt werden.

Die Gesamtheit der einfach ausschließbaren Klartexte zu einem Chiffrat c soll als $\Omega(c) := M_c(n-1)$ bezeichnet werden. Für die Anzahl ω dieser Klartexte gilt:

$$\omega(c) := |\Omega(c)| = \sum_{k=0}^{|c|-2} I_c(k)$$

Für die Anzahl der unpassenden Klartexte ist nur die Position der Null und die Länge $(n+1)$ des Chiffrats relevant: $\omega(i, n+1) := \omega((c_0, ..., c_n))$ mit $c_i = 0$ und für $j \neq i$: $c_j \neq 0$. Falls an keiner Stelle eine Null vorkommt: $\omega_*(n+1) := \omega((c_0 \neq 0, ..., c_n \neq 0))$. Ist die Null die letzte Stelle im Chiffrat, liefert sie nicht mehr Informationen als ein beliebig anderes Zeichen. Andernfalls spielt die Position der Null nur eine geringe Rolle. (siehe Tabelle A.1) Unter der plausiblen Annahme, dass mehr Nullen zu mehr ausschließbaren Klartexten führen und der Annahme, dass Chiffrate gleichverteilt auftreten, gilt für ein solches zufälliges Chiffrat c der Länge $n+1$:

$$E[\omega(c)] > \left(1 - \left(\frac{35}{36}\right)^{n+1}\right)\left(\frac{1}{n+1}\sum_{i=0}^{n}\omega(i, n+1)\right) + \left(\frac{35}{36}\right)^{n+1}\omega_*(n+1)$$

Dies ist nur eine Abschätzung, da die Wahrscheinlichkeiten und Auswirkungen einer größeren Anzahl an Nullen im einzelnen Chiffrat nicht berücksichtigt wird. Konkrete Werte finden sich in Tabelle A.2 (Ergebnis aus lc4-leakage.py).

4.2.4.2. Konkrete Angriffe

IND-Angriffe Im Folgenden soll gezeigt werden, dass weder die einfache Verschlüsselung, noch die im Protokoll beschriebene Verschlüsselung IND-CPA-sicher ist. IND-CPA erlaubt dem Angreifer die Klartexte zu wählen. Tatsächlich ist auch die Unterscheidung von zufälligen Klartexten möglich. Für die Verteilung der Nachrichten und Chiffrate wird

angenommen, dass diese gleichverteilt sind, um die Berechnung der Erfolgswahrscheinlichkeiten zu vereinfachen. Um ein Chiffrat einem Klartext zuzuordnen, lässt sich wie folgt vorgehen:

Algorithmus 1 LC4-Zuordnung zufälliger Klartexte und Chiffrate

---

Geg. $X = (x_0, ..., x_n)$, Y die Chiffrate zu zwei Klartexten $A = (a_0, ..., a_n)$, B, wobei B und Y unbekannt seien und zufällig, welches Chiffrat zu welchem Klartext gehört.

1. Betrachte alle Vorkommen der Null im Chiffrat X ($i \in \{0, ..., n-1\}$ mit $x_i = 0$)
   - Falls $a_i = 0$ und $x_{i+1} \neq a_{i+1}$, gebe $\langle (X, B), (Y, A) \rangle$ aus.
   - Falls $a_i \neq 0$ und $x_{i+1} = a_{i+1}$, gebe $\langle (X, B), (Y, A) \rangle$ aus.
2. Betrachte alle übereinstimmenden Stellen $i \in \{0, ..., n-1\}$ mit $a_i = x_i \neq 0$. Falls $x_{i+1} = a_{i+1}$, gebe $\langle (X, B), (Y, A) \rangle$ aus.
3. Gebe $\langle (X, A), (Y, B) \rangle$ aus.

Der Aufwand ist offensichtlich linear in der Länge der Nachricht. Wird in den ersten beiden Schritten etwas ausgegeben, ist die Zuordnung zwingend richtig. Die Wahrscheinlichkeit hierfür ist abhängig von der Länge der Nachricht ($n + 1$). In einer Nachricht dieser Länge können $E[m_0] := \frac{n}{36}$ betrachtbare Vorkommen der Null und darüber hinaus $E[m_s] := \frac{(35/36)n}{36}$ Übereinstimmungen in Klartext und Chiffrat erwartet werden. Die Wahrscheinlichkeit, dass im 1. Schritt nichts ausgegeben wird, obwohl $\langle (X, B), (Y, A) \rangle$, dass also die Annahme (X, A) hält, liegt bei $((\frac{1}{36})^2 + (\frac{35}{36})^2)^{m_0}$. Die Wahrscheinlichkeit, dass auch im zweiten Schritt trotz $\langle (X, B), (Y, A) \rangle$ nichts ausgegeben wird, liegt analog bei $(\frac{35}{36})^{m_s}$. Insgesamt ist die Wahrscheinlichkeit einer sicheren Unterscheidung $p_s := 1 - (\frac{1+35^2}{36^2})^{m_0} (\frac{35}{36})^{m_s}$. Die Wahrscheinlichkeit der korrekten Zuordnung liegt also bei $p_s + 0.5(1 - p_s)$. Die erwartete Wahrscheinlichkeit einer richtigen Zuordnung bei Nachrichten mit $n = 2000$ betrachtbaren Zeichen liegt bei $\approx 99.5\%$ ($p_s \approx 0.99$). Mit $n = 200$ liegt die Wahrscheinlichkeit noch bei $\approx 68.46\%$ ($p_s \approx 0.3691$).

Sind auch B und Y bekannt, erhöht sich die Wahrscheinlichkeit einer erfolgreichen Unterscheidung. Dafür führe man den ersten Schritt des obigen Algortihmus für $\{X, A\}$, $\{X, B\}$, $\{Y, A\}$ und $\{Y, B\}$ aus. Die Wahrscheinlichkeiten in den einzelnen Ausführungen sind nicht unabhängig voneinander, weswegen die empirische Wahrscheinlichkeit in Abhängigkeit von der Nachrichtenlänge betrachtet wird (siehe Abbildung 4.3).

Können die Nachrichten wie im Fall eines IND-CPA Angriffs frei gewählt werden, lässt sich ein erfolgreicherer Algorithmus angeben. Dieser ergibt sich direkt aus der ersten Beobachtung. Wird eine Nachricht nur bestehend aus Nullen verschlüsselt und in einem Verschlüsselungsschritt eine Null verschlüsselt, sind alle folgenden Chiffratzeichen null.

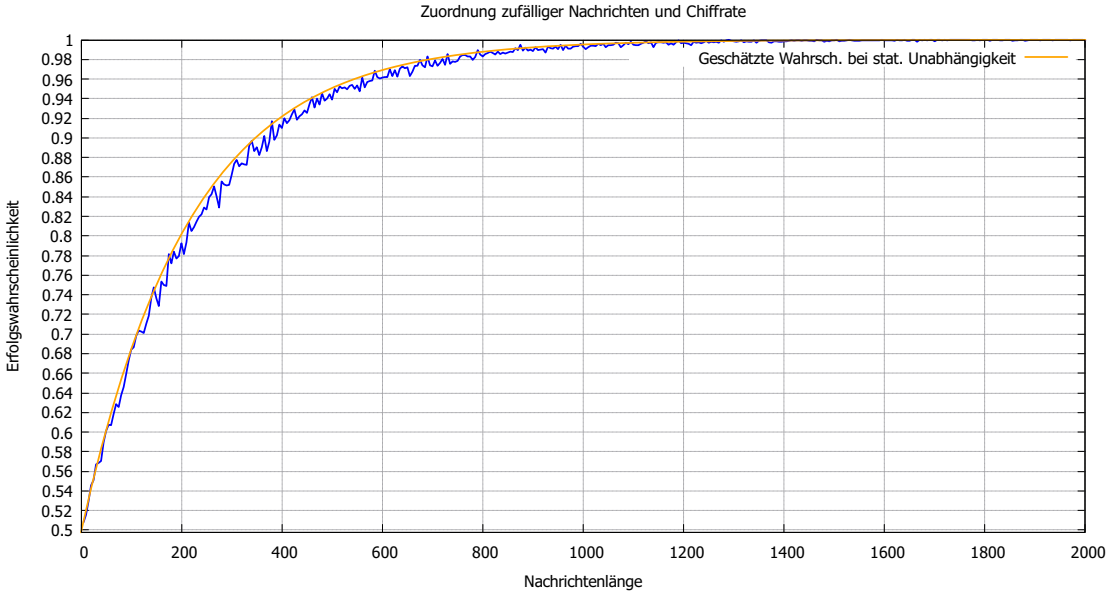

Abbildung 4.3.: Empirische Wahrscheinlichkeit einer korrekten Zuordnung zufälliger Nachrichten durch Algorithmus 1 für Nachrichtenlängen 5,10,...,2000. Je 1000 Versuche. In Orange ist die erwartete Wahrscheinlichkeit abgetragen, wenn statistische Unabhängigkeit und Gleichverteilung der Chiffrate angenommen wird.

Algorithmus 2 IND-CPA Angriff auf das LC4

Geg. Verschlüsselungsorakel $Enc(k, m)$

1. Wähle $P_0 = 0^n$ und $P_1 = 1^n$ und schicke diese an das Orakel.
2. Erhalte vom Orakel $c = (c_0, ..., c_{n-1}) = Enc(P_i)$, $i \in \{0, 1\}$
3. Falls ein j mit $c_j = 0$ und ein $k > j$ mit $c_k \neq 0$ existiert, gebe 1 aus.
4. Sonst gebe 0 aus.

Unter genannter Gleichverteilungsannahme für Chiffrate, ergibt sich eine theoretisch zu erwartende Erfolgswahrscheinlichkeit von $\frac{1}{2} + \frac{1}{2}\left(1 - \left(\frac{35}{36}\right)^{n-1}\right)$. Die empirisch gemessene Erfolgswahrscheinlichkeit für zufällige Nachrichten liegt nahe an diesem Wert: Für $n = 20$ liegt die gemessene Erfolgswahrscheinlichkeit bei 70.47%, für $n = 50$ bei 87.398% und $n = 200$ bei 99.812% (je 10^5 Wiederholungen des IND-CPA-Spiels mit zufälligen Schlüsseln) (LC4Statistics/IND/INDGames.cs).

Broadcast-Angriff Ein weiteres Problem, das sich aus den Beobachtungen ergibt: Wenn ein gleichbleibender Nachrichtenteil $p_j, ..., p_k$ mit unterschiedlichen Schlüsseln oder ansonsten anderer Nachricht mehrfach verschlüsselt wird, können Rückschlüsse auf diesen Nachrichtenteil gezogen werden. Beispielsweise lässt sich so der MAC in obigem Protokoll eingrenzen oder sogar entschlüsseln.

Aus der Menge aller bekannten Chiffrate C zur gleichen Nachricht werden für alle $k > i \geq j - 1$ Teilmengen $C_i = \{c = (c_0, ..., c_n) \in C : c_i = 0\}$ gebildet, deren Elemente an Stelle i eine Null stehen haben. Gilt für zwei unterschiedliche $c', c'' \in C_i$, dass $c'_{i+1} \neq c''_{i+1}$, dann kann gefolgert werden, dass $p_{i+1} \neq c'_{i+1}$ und $p_{i+1} \neq c''_{i+1}$. Gilt dagegen für festen Wert k dass für alle $c \in C_i$: $c_{i+1} = k$, wird gefolgert, dass $p_{i+1} = k$. Für diese Folgerung ist wichtig, dass $|C_i|$ ausreichend groß ist. [9] Dieses Vorgehen kann um die 3. Beobachtung erweitert werden. Diese besagt, dass keine zwei aufeinanderfolgenden Zeichen, deren erstes Zeichen *keine* Null ist, in Chiffrat und Klartext übereinstimmen können. Definiere dafür boolesche Variablen für alle $\sigma \in \{0, ..., 36\}$ $X_i^\sigma := (p_i = \sigma)$. Dann erzeugen $p_j, ..., p_k$ für alle Chiffrate $c \in C$ eine erfüllende Belegungen von

$$\bigwedge_{\substack{i=j,\\ c_i \neq 0}}^{k-1} \neg(X_i^{c_i} \wedge X_{i+1}^{c_{i+1}}) = \bigwedge_{\substack{i=j,\\ c_i \neq 0}}^{k-1} (\neg X_i^{c_i} \vee \neg X_{i+1}^{c_{i+1}})$$

Über alle Chiffrate bildet sich eine 2-KNF mit maximal $|C|(k - j)$ Termen und $36(k - j)$ Variablen.

Beispiel Für ein einzelnes Chiffrat `ax0ca` ist die KNF:

$$(\neg X_0^a \vee \neg X_1^x) \wedge (\neg X_1^x \vee \neg X_2^0) \wedge (\neg X_3^c \vee \neg X_4^a)$$

Z.B. passen weder Klartext `axf8l`, `5x0db` noch `_3gca` zum Chiffrat.

Die ersten beiden Beobachtungen lassen sich einfacher ausnutzen. Führen diese zu keinem eindeutigen Ergebnis, lässt sich die dritte Beobachtung als Erfüllbarkeitstest der 2-KNF einbeziehen. Für 2-KNF kann die Erfüllbarkeit in Polynomialzeit geprüft werden [9]. Sind viele Chiffrate bekannt, können bereits die ersten beiden Beobachtungen zu einer eindeutigen Lösung führen: In einem Experiment mit 10000 Chiffraten von Nachrichten deren erste 100 Zeichen zufällig und letzten 26 Zeichen aus dem Text »`diese_nachricht_ist_geheim`«bestand, wobei der Schlüssel pro Chiffrat zufällig gewählt wurde, konnte der Text mit den ersten beiden Beobachtungen, also ohne Erfüllbarkeitstest der 2-KNF, zuverlässig aus den Chiffraten extrahiert werden
(LC4Statistics/BroadcastAttackTest.cs): In 100 Wiederholungen des Experiments konnten im Durchschnitt 25.63 der 26 Zeichen eindeutig und richtig bestimmt werden. Für die restlichen 0.37 Zeichenstellen waren zwei Möglichkeiten geblieben. In 69 der 100 Versuche konnten alle 26 Zeichen eindeutig bestimmt werden.

Um bei unterschiedlichen Chiffraten des gleichen Klartextes auf den Klartext eindeutig zu schließen, sind durchschnittlich $log_{(1-p)}(\frac{1}{36^{n+1}})$ mit $p := E[\omega(c_0, ..., c_n)]/36^{n+1}$ nötig, unter der Annahme, dass diese Chiffrate gleichverteilt zufällig sind. Für eine Nachricht

[9] Dies folgt aus den ersten beiden Beobachtungen.

mit $n = 35$ (Länge 36) können demnach durchschnittlich $> 5.73\%$ der Klartexte ausgeschlossen werden. Für einen Broadcast-Angriff auf eine Nachricht dieser Länge sind also im Durchschnitt höchstens ≈ 2186 Chiffrate notwendig.

Known-Plaintext-Angriff Auf LC4 existiert ein Known-Plaintext-Angriff [51, S.10 f.]. Hierbei kennt ein Angreifer ein Klartext-Chiffrat-Paar. Initial sind alle Zeichen in der Zustandsmatrix S unbekannt. Nun wird das Zeichen an $S[i][j]$ (Spielstein mit Marker) geraten. Mit dem Chiffrat- und Klartext-Zeichen und geratenem $S[i][j]$ können mit jedem Schritt weitere Zeichen in der Matrix S bestimmt werden. Ist der Schlüssel eindeutig für ein Klartext-Chiffrat-Paar, so wird ein falsch geratenes $S[i][j]$ in einem folgenden Schritt zu einem Konflikt führen. Mit einer Nonce der Länge 6 wird der Median der Anzahl Konflikte, um den Schlüssel ohne Kenntniss einzelner Schlüsselstellen zu rekonstruieren, von Kaminsky [51] auf $\approx 2^{58.9}$ (Median) geschätzt. Waren 8 Schlüsselstellen bekannt, lag der Median bei $\approx 2^{31}$ Konflikten, bevor der Schlüssel rekonstruiert werden konnte [51, S.11, §Abb.1].

Mit einem generischen SAT-Solver[10] ließ sich in einer Formalisierung des Problems der Schlüssel nicht bestimmen (lc4-sat-adv.py). [11] Ein eigenes Backtracking (LC4Statistics/KnownPlaintextAttack/) nutzt $p_i = c_i \Rightarrow S_i(0) = 0$ und $p_i \neq c_i \Rightarrow S_i(0) \neq 0$ aus. Der Median der Anzahl geschätzter Konflikte kommt mit $\approx 2^{56.5}$ Konflikten (250 Wiederholungen) auf ein ähnliches Ergebnis wie Kaminsky, wobei zu beachten ist, dass die Klartexte rein zufällig und nicht anhand der Bigramm-Verteilung der englischen Sprache erstellt wurden. Statt eine Regression wie in [51] zu verwenden, wurden für je zufällige aber feste Klartext-Chiffratpaare die Anzahl der passenden A Folgezustände mit 5 bekannten Stellen berechnet. Von diesen Folgezuständen wurden 100 zufällig[12] gewählt, um die daraus möglichen Folgezustände mit 10 bekannten Stellen zu berechnen. Dabei wurde die durchschnittliche Anzahl α der auftretenden Konflikte notiert. Ausgehend von den Folgezustände mit 10 bekannten Stellen wurden erneut 100 zufällig gewählt, um die Anzahl der Folgezustände mit 15 bekannten Stellen zu berechnen. Die dabei durchschnittlich auftretenden Anzahl Konflikte wird mit β bezeichnet. Von den Zuständen mit 15 bekannten Stellen wurden 100 gewählt, um vollständige Folgezustände zu berechnen. Wobei durchschnittlich γ Konflikte auftraten. Die Anzahl der Konflikte für ein festes Klartext-Chiffrat-Paar wurde als $A \cdot \alpha \cdot \beta \cdot \gamma$ geschätzt.

Beobachtbar ist eine hohe Varianz der Konfliktanzahl: Die Standardabweichung liegt bei $\approx 2^{62.8}$ und die durchschnittliche Konfliktanzahl bei $2^{60.6}$. Bestimmte Klartext-Chiffratpaare führen zu messbar weniger Konflikten und bieten damit weniger Sicherheit vor Known-Plaintext-Angriffen. Ist ein beliebiger Folgezustand in einem Klartext-Chiffratpaar vollständig bekannt, können alle Zustände, insbesondere also der Schlüssel[13], zurückgerechnet werden. Die Varianz der geschätzten Konflikte kann also auch innerhalb eines Klartext-

[10] `https://github.com/Z3Prover`

[11] Programm nach 80 Stunden Laufzeit beendet.

[12] Zusätzlich wurde in jedem Schritt der tatsächliche Folgezustand mit 5,10,15 bekannten Stellen einbezogen, der auch immer Teil der Anzahl möglicher Folgezustände ist. Grund dafür ist zum einen das Sicherstellen, dass weitere Folgezustände gefunden werden, und zum anderen, dass die Konfliktanzahl möglicherweise von der der zufälligen Folgezustände signifikant abweicht.

[13] Für die Nonce: Rate im Folgezustand $pos(S_0(0)) \Rightarrow (S_0(0)$ und Chiffratzeichen $c) \Rightarrow pos(c) \Rightarrow$ rotiere rückwärts $\Rightarrow$ prüfe ob neue Positionen von c und Noncezeichen mit $S_0(0)$ übereinstimmt.

Chiffratpaars ausgenutzt werden, indem für das Backtracking ein vielversprechend erscheinender Index gewählt wird. Beispielsweise können Indizes i mit $p_i = c_i$ als Ausgangspunkt gewählt werden, um direkt den Wert von $S_i(0)$ zu kennen. Um vielversprechende Indizes systematischer zu finden, werden mehrere Abhängigkeiten betrachtet. Die folgenden Mengen sind eindeutig bestimmt, ist also ein Element unbekannt, kann es aus den anderen hergeleitet werden.

1. $\{pos(c_i)^i, pos(p_i)^i, S_0^i\}$
2. $\{pos(S_0^i)^i, pos(c_i)^i, pos(p_i)^i, p_i, pos(S_0^{i+1})^i\}$
3. $\{pos(c_i)^i, pos(c_{i+1})^i, pos(p_i)^i, p_i, pos(c_i)^{i+1}\}$
4. $\{pos(p_{i+1})^i, pos(c_i)^i, pos(p_i)^i, p_i, pos(p_{i+1})^{i+1}\}$

$pos(.)^i$ bezeichnet die Position eines Zeichens im i-ten Zustand (Zustände werden nicht normalisiert). S_0^i bezeichnet den Wert des »gepinnten Steins«[14]. Mit diesen Abhängigkeiten lassen sich Indizes finden, von denen aus möglichst wenig Werte und Positionen geraten werden müssen, sondern sich aus bereits geratenen Werten ableiten lassen. In 250 zufälligen Nachricht-Chiffratpaaren der Länge $n = 200$ wurde in der ersten Hälfte der Paare nach dem Index gesucht, für den die Anzahl der zu ratenden Werte innerhalb der nächsten 10 Zeichen des Klartexts und Chiffrats minimal ist. Die Verwendung dieser Indizes als Ausgangspunkt für das Backtracking führte im Median zu geschätzt $2^{51.7}$ Konflikten.

4.2.4.3. Mögliche Gegenmaßnahmen

Eine Protokollanpassung kann verwendet werden, um dem Bias in der Authentifikation entgegenzuwirken, verhindert aber keinen der beschriebenen Angriffe. Die Probleme liegen nicht im Protokoll, sondern in der Verschlüsselungsfunktion (4.2.4). Als mögliche Gegenmaßnahme kann ein zweiter Index (Pin) als Teil des Schlüssels eingeführt werden, der sich unabhängig vom ersten Index bewegt. Die Berechnung ausgehend vom Klartextzeichen würde über den Wert des ersten und zweiten gepinnten Steins erfolgen, wodurch verhindert wird, dass der Wert eines Steins eindeutig bestimmt werden kann. Um den Bias in der Authentifikation zu vermeiden, wird ein größerer Teil des Zustands für die Berechnung des Folgezustands verwendet, indem der zweite gepinnte Stein anhand des Klartextzeichens und des Werts des Steins über dem zweiten Pin berechnet wird.

Mit diesen Maßnahmen konnte für das Experiment, das die Wahrscheinlichkeit eines unveränderten MAC beim Verändern eines Zeichens in der Nachricht misst, kein Bias mehr festgestellt werden. Die beschriebenen Angriffe funktionieren ebenfalls nicht. Allerdings ist auch die angepasste Verschlüsselung (LC4Statistics/LC5.cs) nicht IND-CPA-sicher: Beim Verschlüsseln mit zufälligem Schlüssel einer nur aus Nullen bestehenden Nachricht der Länge 10^7, treten im Chiffrat durchschnittlich $\approx$ 0.296% mehr Nullen auf, als bei einer Gleichverteilung anzunehmen sind (100 Wiederholungen). [15] Das Chiffrat zu einer nur aus Nullen bestehenden Nachricht der Länge 10^{10} und zufälligem Schlüssel wurde zusätzlich mit der dieharder-Testsuite[16] [18] auf statistische Eigenschaften geprüft, die ein

[14] Wert von $S_i(0)$ im normalisierten i-ten Zustand

[15] Die Wahrscheinlichkeit, dass ein solcher Bias bei Gleichverteilung ($p = \frac{1}{36}$) entsteht, ist $< 10^{-50}$.

[16] https://linux.die.net/man/1/dieharder

Zufallsgenerator erfüllen muss. Als einziger Test schlägt 'dab_bytedistrib'[17] fehl, was auf das häufigere Vorkommen der Null als Ausgabe zurückzuführen ist.

4.2.5. Asymmetrische Kryptographie

Es existieren Versuche, für ressourcenbeschränkte Geräte asymmetrische Systeme zu entwickeln [71, 76]. Ein Ansatz der dabei teilweise in den Bereich der Mensch-Ausführbarkeit fällt, sind Systeme basierend auf dem Knapsack-Problem. Von Knapsack wird vermutet, dass es auch für Quantencomputer schwierig zu lösen ist [59, S.59388]. Das Merkle-Hellman Public-Key-Cryptosystem (MH PKC) [68] ist das erste solche System, wurde jedoch bereits 1982 durch Shamir [87] und Adleman [2] gebrochen [71, S.2]. Auch Vorschläge von Hwang et al. [42] und Kobayash et al. [53], die auf dem Knapsack-Problem basieren, haben sich als unsicher erwiesen [1, 39]. Ein weiteres Knapsack-Verfahren wurde von Murakami et al. [71] vorgeschlagen. Schlüsselgenerierung und Entschlüsselung sind hierbei für Menschen zwar mit keinem vertretbaren Aufwand ausführbar, die Verschlüsselung aber ist einfach. Ein Mensch erhält dazu den öffentlichen Schlüssel als Zahlenfolge $(a_1, ..., a_n) \in Z_N^n$ und berechnet zum Klartext $(m_1, ..., m_n) \in \{0, 1\}^n$ das Chiffrat $C = \sum_{i=1}^{n} a_i m_i$ [71, S.5]. Der Aufwand beschränkt sich also auf der Addition von Zahlen. Auch wenn viele Zahlen zu addieren sind, sollte dies mit Stift und Papier für die meisten Menschen keine Herausforderung darstellen. Der Klartext ist hier als Bitvektor dargestellt. Ist der Klartext eine Zahl, wird Schalkwijk's Algorithmus verwendet, um einen Bitvektor mit bestimmten Mindestgewicht (also $|\{m_i \neq 0 | i \in \{1, ..., n\}\}| > x$) zu erhalten und somit unsichere Nachrichten zu vermeiden [71, S.2 f.]. Schalkwijk's Algorithmus ist allerdings nicht mensch-ausführbar. Auf das System von Murakami et al. [71] existiert ein Broadcast-Angriff [77]: Wird eine Nachricht mit verschiedenen öffentlichen Schlüsseln verschlüsselt, kann ein Angreifer anhand dieser Chiffrate den Klartext ermitteln.

4.3. Evaluation

In Tabelle 4.1 sind die Eigenschaften der betrachteten Verfahren kurz dargestellt. Bis auf die Verschlüsselung in VC benötigen alle Verfahren zusätzlich ein zufallsgenerierendes Hilfsmittel. Das LC4-Protokoll, das authentifizierte und integritätsprüfende Verschlüsselung bereitstellt, hat sich in mehrerlei Hinsicht als unsicher erwiesen. Zeichen am Ende der Nachricht sind schlechter vor unbemerkter Manipulation geschützt. IND-CPA- Broadcast- und Known-Plaintextangriffe zeigen, dass LC4 keine Vertraulichkeit der verschlüsselten Nachrichten herstellen kann.

VC kann über Verfahren wie in [72] beschrieben authentifiziert werden. Dieses Verfahren lässt sich auch auf andere OTPs anwenden, benötigt dafür aber sehr große Schlüssel und entsprechend großen Aufwand, da für jede Verschlüsselung nur ein Teil des Pads für die Nachricht verwendet werden darf. One-Time-Pads und Stromchiffren können zwar auch über One-Time-MACs [93] mensch-ausführbar authentifiziert und die Integrität

[17] 'dab_bytedistrib' extrahiert jeweils Bytes aus aufeinanderfolgenden 32bit-Wörtern. Die Häufigkeit der Byte-Werte aus verschiedenen Wortfolgen wird auf die Wahrscheinlichkeit einer zugrundeliegenden Gleichverteilung getestet. Die Konvertierung in 32Bit-Wörter erfolgte durch Rejection Sampling.

	Alphabetgröße $\|Z\|$	Schlüsselmenge $\|K\|$	k wiederverwendbar
OTP	-	$\|Z\|^{\|m\|}$	nein
VC (Entschl.)	-	$\|Z\|^{\|m\|}$	nein
Solitaire	26/52	54!	nein
LC4	36*	36!	ja

	Fehlerauswirkung	Art der Mensch-Ausführbarkeit	Hilfsmittel
OTP	gering	schwach	(Stift/Papier)
VC (Entschl.)	-	unterstützt	Transparente
Solitaire	hoch	schwach	Kartendeck
LC4	hoch	schwach	Spielsteine

	Aufwand (min)	sicher?	authentifiziert?
OTP ($\|Z\| = 26$)	≈ 6	ja	nein
VC (Entschlüsselung)	-	ja	nein
Solitaire	$\approx 99 + 6$	ja**	nein
LC4 (unauth.)	33.5	nein	nein
LC4 (auth.)	36.5	nein	ja

Tabelle 4.1.: Aufwand/Verwendung und Sicherheit der betrachteten Verfahren. Aufwand für $|m| = 200$ geschätzt mit $t_+ = t_o = 0.5s, t_r = 5s, t_m = 0.2s, t_{rot.} = 2s, t_s = 4s, t_f = 5s, t_{sb} = 2s, t_c = 0.25s, t_{mod} = 2s$. Der Aufwand zur Generierung eines Geheimnisses ist nicht einbezogen. * Auch 49,64,... möglich. ** Für kurze Nachrichten

sichergestellt werden, dies stellt aber einen erheblichen zusätzlichen Aufwand dar. Um die Integrität sicherzustellen, müssen alle Zeichen der Nachricht berücksichtigt werden, also in die Berechnung einfließen, da andernfalls die nicht-berücksichtigten Zeichen unbemerkt verändert werden können [93, S.270]. Zur Berechnung des MAC kann eine universelle Hashfunktionen (wie z.B. in [21]) mehrfach angewendet werden. Dafür wird die Nachricht in mehrere Blöcke geteilt, die als Eingabe der Hashfunktion dienen. Die Ergebnisse werden konkateniert und der Vorgang mit dem konkatenierten Ergebnis als Nachricht wiederholt [93, S.272]. Selbst mit einer einfach zu berechnenden Hashfunktion ist der menschliche Berechnungsaufwand für längere Nachrichten erheblich. Carter & Wegmann [93] raten zur Verwendung eines Computers. Die Authentifikation einer Nachricht über ein LC4-ähnliches Verschlüsselungsverfahren, das einen Zustand ausgehend von Klartext- und Chiffratzeichen verändert, ist weniger aufwändig. Entsprechend kann sich die Suche nach einem solchen Verfahren lohnen.

Die Fehlerwahrscheinlichkeit in der Berechnung durch Menschen wurde nicht genauer betrachtet. Werden Ermüdungs- und Routineeffekte ignoriert, steigt die Fehlerwahrscheinlichkeit linear mit der Länge des Klartexts. Fehler in der Anwendung von Solitaire und LC4 führen dazu, dass die entschlüsselte Nachricht ab dem Auftreten eines Fehlers unlesbar ist. Erfolgt die Entschlüsselung mit einem Computer könnten Annahmen über die Position und Art des Fehlers trotzdem verwendet werden, um den Klartext wiederherzustellen. Beispielsweise kann in einem Chiffrat für »`never_gonna_givdifhei#wofbw...`«vermutet werden, dass die Verschlüsselung des 16. Zeichens fehlerhaft ist. Wird davon ausgegangen, dass dem z.B. ein »Off-By-One«-Fehler in der Berechnung zugrunde liegt, können alle Schlüsselzustände berücksichtigt werden, die ausgehend vom 15. Schlüsselzustand bei einem Vorkommen dieses Fehlers das Zeichen 'd' ausgeben. Alternativ kann ein Mensch die Verschlüsselung mehrfach ausführen, um die Korrektheit des Chiffrats auf Kosten der Ausführungszeit sicherzustellen. One-Time-Pads bieten die Fehlertoleranz betreffend den Vorteil, dass pro Fehler höchstens ein Zeichen in Chiffrat und entschlüsseltem Klartext betroffen ist.

5. Identifikationsprotokolle

Während mensch-ausführbare Verschlüsselung eher wenig akademische Aufmerksamkeit geniest, gibt es schon lange den Versuch, Passwörter als Mittel zur Authentifikation durch sicherere Verfahren zu ersetzen [89, 66]. Das einmalige Abgreifen eines Passworts, sei es durch Spähsoftware, unverschlüsselte Kommunikation, Videoüberwachung oder analoge Beobachtung, reicht einem Angreifer, um sich erfolgreich zu authentifizieren. Der Mensch gibt mit jedem Authentifikationsversuch sein Geheimnis in Form des Passworts vollständig preis. Abhilfe können Challenge-Response-Verfahren (z.B. [58, 41, 96]) schaffen, in denen Nutzer, auf Grundlage ihres Geheimnisses, Antworten auf Fragen des Authentifikators berechnen. Zwar existieren einige solcher Verfahren, jedoch bieten sie in der Regel nur für eine begrenzte Anzahl Authentifikationen Schutz oder sind sehr aufwändig [15].

Im Folgenden wird mit $\mathcal{H}$ ein Mensch bezeichnet, der über ein System/Interface $\mathcal{I}$, mit einem System C kommuniziert, an welchem er sich identifizieren möchte. Zwischen C und $\mathcal{H}$ besteht ein Kanal $\mathcal{T}$. $\mathcal{A}$ bezeichnet einen Angreifer. Für die in diesem Abschnitt folgenden Protokolle, wird davon ausgegangen, dass $\mathcal{H}$ und C ein gemeinsames Geheimnis $k \in K$ besitzen. Alle folgenden Protokolle sind Challenge-Response-Verfahren. Authentifikation und Identifikation werden in diesem Kapitel synonym verwendet. Während Interface $\mathcal{I}$ und Kanal $\mathcal{T}$ im Folgenden als unsicher angenommen werden, wird die Sicherheit von C vorausgesetzt. Betrachtungen unter der Annahme, dass C korrumpierbar ist (z.B. Malicious-Administrator-Angriffe [58]) sind nicht Gegenstand dieses Kapitels.

5.1. Definitionen

Definition 5 (Identifikationsprotokoll) *Ein **Identifikationsprotokoll** ist ein Paar von Funktionen $\langle H, C\rangle$ für das gilt:*

- $\forall z \in K : Pr[\langle H(z), C(z)\rangle = akzeptiert] > p_a$
- $\forall x \neq y : Pr[\langle H(x), C(y)\rangle = akzeptiert] < p_w$

Mit Wahrscheinlichkeit $> p_a$ wird also bei gleicher Eingabe korrekterweise akzeptiert, der Nutzer also identifiziert. Mit Wahrscheinlichkeit $< p_w$ kommt es zu einer falschen Identifikation. Diese Definition findet sich auch bei Hopper & Blum [41, S.54, §Def.1] und Li et al. [57, vgl.Def.1&2]. Hopper & Blum [41] verwenden $p_a = 0.9$ und $p_w = 0.1$.

5.1.1. Angreifermodelle

Mit $T(H(z), C(z))$ wird im fogenden das Transkipt einer Identifikation, also die Gesamtheit aller zur Identifikation benötigten ausgetauschten Nachrichten bezeichnet. Mit

$T^s(H(z), C(z))$ werden die Transkripte aus s unabhängigen Identifikationsprozessen bezeichnet (vgl. [41, 58]).

Definition 6 (passiv sicher) *Ein Identifikationsprotokoll heißt (p, s)-sicher gegen **passive** Angreifer (Known-Response-Angriff), wenn für alle polynomiell beschränkten Angreifer $\mathcal{A}$ gilt [41, S.55, §Def.3][58]:*

$$Pr[\langle \mathcal{A}(T^s(H(z), C(z))), C(z)\rangle = akzeptiert] \leq p$$

Ein passiver Angreifer, der nur wenige Transkripte (z.B. $s \leq 10$) beobachten kann, wird **opportunistisch** genannt [25].

Definition 7 (aktiv sicher) *Ein Identifikationsprotokoll heißt (p, s)-sicher gegen **aktive** Angreifer (Chosen-Message-Angriff), wenn für alle polynomiell beschränkten Angreifer $\mathcal{A}$ gilt [41, S.55, §Def.4][58]:*

$$Pr[\langle \mathcal{A}(T^s(H(z), C(z))), C(z)\rangle = akzeptiert] \leq p$$

Wobei $\mathcal{A}$ beliebige Änderungen in der Kommunikation zwischen $\mathcal{H}$ und C vornehmen kann. $\mathcal{A}$ hat also vollen Zugriff auf den Kanal $\mathcal{T}$.

Definition 8 (sensitiv) *Ein Identifikationsprotokoll heißt (p, q, s)-**sensitiv** gegen aktive Angreifer, wenn für alle polynomiell beschränkten Angreifer $\mathcal{A}$ gilt [41, S.55, §Def.5][58]:*

- $Pr[\langle H(z), \mathcal{A}(T^s(H(z), C(z)))\rangle \neq \perp] < q$
- $Pr[\langle \mathcal{A}(T^s(H(z), C(z))), C(z)\rangle = akzeptiert] < p$

$\perp$ steht dafür, dass $\mathcal{H}$ ablehnt, sich bei C zu identifizieren.

5.1.2. Generische Angriffe

Bezeichne $K' \subset K$ die Menge der zu Transkripten $T^s(H(z), C(z))$ passende Passwortmenge, in dem Sinn dass: $\forall k' \in K' : T^s(H(k'), C(k')) = T^s(H(z), C(z))$. In jedem der betrachteten Verfahren sinkt $|K'|$ mit der Anzahl s der Transkripte. Dies wird als Informationsleck (information leak) bezeichnet [25, vgl. S.3].

Brute-Force-Angriff Suche über den gesamten Passwortraum K nach passenden Passwörtern. Ein derartiger Angriff ist im Allgemeinen nur möglich, wenn $\mathcal{A}$ $s \geq 1$ Transkripte beobachtet hat. In den vorgestellten Challenge-Response-Verfahren wird sich für kleines s die Passwortmenge K zwar einschränken lassen, aber zu keiner eindeutigen Lösung führen [58]. Ist der Passwortraum ausreichend groß, sind Brute-Force-Angriffe unpraktikabel.

Premature-Angriff Sind $\mathcal{A}$ mehrere Transkripte bekannt, die aufgrund eines Informationslecks eine Eingrenzung des Passwortraums auf mögliche Passwörter zulassen, kann $\mathcal{A}$ versuchen, sich mit einem dieser Passwörter zu authentifizieren, wobei die Erfolgswahrscheinlichkeit über der eines Rateangriffs liegt und mit weiteren Einschränkungen steigt [25].

Rateangriff (Guessing Attack) (auch: Blind Attack [22, S.22], Random Response Attack): Wahrscheinlichkeit von A durch C identifiziert zu werden bei zufälliger Antwort auf Challenges [58].

Wörterbuch-Angriff (Dictionary Attack): Dieser Angriff ist zielführend, wenn H das Geheimnis $k \in K$ frei wählen kann. Die Menge der Passwörter, die für einen Menschen eine Bedeutung haben oder einfach zu merken sind und entsprechend mit höherer Wahrscheinlichkeit gewählt werden, ist potentiell [1] viel kleiner, als die Menge aller möglichen Passwörter. Ein Angreifer nutzt dies durch eine Suche über all diese Passwörter [58].

Replay-Angriff (auch: Recording-Angriff [22, S.22]): $\mathcal{A}$ speichert ein Transkipt $T(H(z), C(z))$, um es zu einem späteren Zeitpunkt zur Identifikation bei C wieder einzuspielen. Klassische Identifikation mit Passwörtern ist grundsätzlich anfällig für Replay-Angriffe. In Challenge-Response Verfahren darf nicht mehrfach die gleiche Challenge geschickt werden und die Menge möglicher Challenges muss außreichend groß sein [58]. Wird in einem Identifikationsystem davon ausgegangen, dass sich H über 100 Jahre jeden Tag 100 mal identifiziert, werden $36.500.000 < 2^{26}$ verschiedene Challenges benötigt [58, S.16].

Denial-of-Logon (DoL) Angriff Anders als in den zuvor geschilderten Angriffen versucht der Angreifer nicht, sich zu identifizieren oder das Geheimnis k zu lernen, sondern behindert die Fähigkeit von $\mathcal{H}$, sich bei C zu identifizieren. Als Beispiel kann das wiederholte Eingeben einer bewusst falschen PIN bei EC-Karten dienen. Um DoL-Angriffe zu verhindern, muss auf Maßnahmen wie Zugangssperren nach einer bestimmten Anzahl fehlgeschlagener Identifikationen verzichtet werden [58, S.17].

5.2. Protokolle

Das mutmaßlich erste mensch-ausführbare Identifikationsprotokoll wurde 1991 von Matsumoto und Imai [66] vorgestellt. Wie auch nachfolgende Protokolle von Matsumoto [64, 65] erreicht es keine passive Sicherheit [58, S.19 f.].

[1] Ein möglicher Ansatz gegen Wörterbuch-Angriffe ist die Verwendung graphischer [58] oder zufälliger Passwörter

5.2.1. Hopper Blum 1

Protokoll 5 Hopper Blum 1 (HB1) [41, S.60 f.]

Gegeben. $k \in F^n$

Protokollablauf:

1. $\mathcal{H}$ schickt ID an C
2. C schickt Challenge-Matrix $A \in F^{m \times n}$ an C
3. $\mathcal{H}$ schickt $b = Ak + r$ an C, wobei $r \in F^m$ ein Zufallsvektor mit $Pr[r_i \neq 0] = \mu$ und $\mu < 1/2$ ist.
4. C akzeptiert, wenn für mindestens l Stellen $b_i = A_i k$ gilt, wobei A_i die i-te Zeile von A bezeichnet.

Hopper & Blum schlagen $F \in \{\mathbb{Z}_2, \mathbb{Z}_{10}\}$ vor. r enthält die Abweichungen vom korrekten Ergebnis, um durch zufällig falsche Antworten passive Angriffe zu erschweren. Die Zeilen der Matrix A entsprechen den einzelnen Challenges in m Runden, wie in [41] definiert. In den einzelnen Runden wird also jeweils das Skalarprodukt von k und einer Zeile in A berechnet. In $(1 - \mu)m$ Fällen wird das Skalarprodukt verwendet, in $\mu \cdot m$ Fällen ein zufällig falsches.

Aufwand Die von [41] vermutete (.9, .2, 300s)-Mensch-Ausführbarkeit (für $\mu = 1/7$, n=200, k=15) scheint je nach Formulierung des Problems und vorhandenen Hilfsmitteln zumindest in der ersten Komponente etwas zu pessimistisch. Wird die binäre Variante des Protokolls gewählt, ist das Skalarprodukt äquivalent mit dem Abzählen von Unterschieden zwischen zwei Vektoren und der Antwort auf die Frage, ob die Anzahl der Unterschiede gerade oder ungerade ist. Wird der Schlüssel beispielsweise als Hell/Dunkel-Muster auf einen Streifen Papier übertragen und dieser unter eine ebenfalls grafisch dargestellte Challenge gelegt, entfällt ebenso die Herausforderung, sich die einzelnen 1-Stellen des Schlüssels zu merken. Da Zählen und Vergleichen zwei für die meisten Menschen vergleichsweise einfache Tätigkeiten sind, lässt sich die Ausführungszeit einer solchen *unterstützt* mensch-ausführbaren Implementierung des Protokolls wahrscheinlich deutlich verbessern. Eine zeitliche Verbesserung kann allgemein auch dadurch erreicht werden, dass vor der Berechnung bereits ein Zufallsexperiment durchgeführt wird. So kann mit Wahrscheinlichkeit 2μ eine zufällige Zahl gewählt werden statt das Skalarprodukt zu berechnen. Wichtig hierbei wäre, dass die Antworten auf Challenges nicht einzeln sondern alle zusammen eingegeben werden, da sich andernfalls über die kürzere Antwortzeit ein möglicher Seitenkanalangriff eröffnet. Für $\mu = 1/7$, können 2/7 der Berechnungen eingespart werden.

Sicherheit Für Replay-Angriffe sind jeweils die Zeilen der Matrix A und b d.h. die einzelnen Challenge-Response-Paare relevant. Insgesamt sind $|F|^n$ Challenges möglich. Hopper

und Blum geben für $F = \mathbb{Z}_2$ die Erfolgswahrscheinlichkeit eines Rateangriffs mit

$$\left(\frac{1}{2}\right)^m \sum_{i=(1-\mu)m}^{m} \binom{m}{i}$$

an. Li et al. [58] weisen aber darauf hin, dass die Wahrscheinlichkeit, die Antwort auf eine einzelne Challenge korrekt zu erraten, nicht notwendigerweise $\frac{1}{2}$ ist, sondern vom Gewicht des Challengevektors und k abhängt [58, S.23 f.]. Mit einem Gewicht von $|\{i \in \{1, ..., n\} : k_i \neq 0\}| = 8$ ist selbst mit $n = 100000$ nur 49bit-Sicherheit zu erreichen [58, S.26]. Um das Protokoll sicher nutzen zu können, sollte also das Gewicht von k ausreichend groß sein. Allerdings sind k mit höherem Gewicht aufwändiger auswendig zu lernen.

HB1 basiert auf dem LPN (Learning parity with noise) Problem [48, 41, 61], einem NP-schweren Problem. Die Average-Case-Komplexität ist unbekannt, wobei sie als NP-schwer vermutet wird [41, S.56 f.] [61, S.4]. Die Kryptoanalyse von HB1 hat zu effizienten Algorithmen für LPN geführt (Levieil et al. [56], Carrijo et al. [20]). Die passiven Angriffe auf HB1 führen dazu, dass die Schlüssellänge n größer gewählt werden muss, als zunächst von [41] angenommen. Wenn die Anzahl der dazu benötigten Transkripte nicht betrachtet wird, muss, um 80bit-Sicherheit vor passiven Angriffen zu erreichen, mindestens ein 512bit-Schlüssel[2] gewählt werden [61, S.4]. Wird μ klein gewählt (≤ 0.10) sind probabilistische Angriffe mit einer Laufzeit von 2^{36} mit nur 2^{14} Challenge-Response-Paaren möglich [20, S.7 §Tab.1].

Offensichtlich schützt das Protokoll nicht vor aktiven Angriffen [48, 41]. Ein aktiver Angreifer kann eine Challenge v_i mehrfach senden und dadurch die richtige Antwort, also das Ergebnis der Gleichung $v_i \cdot k$ in Erfahrung bringen (in $(1 - \mu) > 0.5$ sendet H die richtige Antwort, bei mehrfacher Frage wird die richtige Antwort ersichtlich). Mit Gauß-Elimination lässt sich so das lineare Gleichungssystem $Ak = r$ nach k lösen. A ist die Matrix mit den Challenges v_i als Zeilen und ebenso wie k als Vektor der richtigen Ergebnisse dem Angreifer bekannt. Der Angreifer kann die Challenges v_i so wählen, dass sie linear unabhängig sind, wodurch sich das Gleichungssystem eindeutig bestimmen lässt [41].

Erweiterungen und Varianten

Einige Erweiterungen wurden für RFID (Radiofrequenzidentifikation) und Geräte mit geringer Rechenleistung konzipiert, nicht aber für die Ausführung durch Menschen. Ein solches Protokoll ist HB+. Juels et al. [48] machten sich hierbei den Umstand zu Nutze, dass Menschen ebenfalls als rechenschwache Geräte aufgefasst werden können, womit ein Protokoll wie HB1 das für Menschen ausgelegt ist, auch auf Geräten mit geringer Rechenleistung und Speicher ausgeführt werden kann. HB+ addressiert gleichzeitig die Schwäche von HB1 gegen aktive Angriffe durch Hinzufügen einer zufälligen Komponente (»Blinding Factor«) durch $\mathcal{H}$ [48, 61]:

[2] $n = 512$ für $F = \mathbb{Z}_2$

Protokoll 6 HB+

Gegeben. $k = (k_1, k_2) \in F^m \times F^m$

Protokollablauf:

1. $\mathcal{H}$ schickt ID und Blinding-Matrix $B \in F^{m \times n}$ an C
2. C schickt Challenge-Matrix $A \in F^{m \times n}$ an C
3. $\mathcal{H}$ schickt $b = A \cdot k_1 + B \cdot k_2 + r$ an C, wobei $r \in F^m$ ein Zufallsvektor mit $Pr[r_i \neq 0] = \mu$ und $\mu < 1/2$ ist.
4. C akzeptiert, wenn für l Stellen $b_i = A_i \cdot k_1 + B_i \cdot k_2$ gilt, wobei A_i, B_i jeweils die i-te Zeile der Matrix bezeichnet.

So benötigt HB+ zwei Vektor-Matrix-Multiplikationen und drei Vektor-Additionen. Ein Aufwand, der verglichen mit einer Matrix-Vektor Multiplpikation im HB1 Protokoll das übersteigt, was Menschen ohne technische Hilfsmittel vermutlich zu berechnen bereit sind. Ein weiteres Problem für Menschen besteht darin, dass sie für B eine zufällige $m \times n$-Matrix generieren müssten. Die Angriffe auf HB1 in [20] und [48] können auch auf HB+ erweitert werden.

Das durch HB+ inspirierte HB++ Protokoll fügt weitere Komplexität hinzu und kann als nicht mensch-ausführbar bezeichnet werden. Darüber hinaus wurden mit der Zeit weitere von HB inspirierte Protokolle vorgeschlagen, wie HB*, HB+DB, HB-MP [75], die aus den gleichen Erwägungen hier nicht besprochen werden.

5.2.2. Hopper Blum 2

Um im folgenden Protokoll aktive Sicherheit zu erreichen, wird der Challengevektor c auf eine Weise erstellt, die Manipulationsversuche sichtbar macht und bei kleinen Veränderungen der Challenge Fehlerkorrektur ermöglicht: c wird als eine Folge von 10x10 Matrizen S_i dargestellt. Für jede dieser Matrizen werden drei Zahlen $a, b, c \in \{0, ..., 9\}$ zufällig gezogen und die Matrixelemente an allen Stellen x, y berechnet: $S_i(x, y) = ax + by + c \bmod 10$. a, b und c lassen sich jeweils erhalten, indem $S_i(0, 0), S_i(0, 1)$ und $S_i(1, 0)$ betrachtet werden. c heißt (nahezu) linear, wenn für keine (nur wenige) Stellen $S_i(x, y) \neq ax + by + c$. In diesen Fällen ist eine Fehlerkorrektur möglich [41, S.62].

3	4	5	6	7	8	9	0	1	2
5	6	7	8	9	0	1	2	3	4
7	8	9	0	1	2	3	4	5	6
9	0	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9	0
3	4	5	6	7	8	9	0	1	2
5	6	7	8	9	0	1	2	3	4
7	8	9	0	1	2	3	4	5	6
9	0	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9	0

Abbildung 5.1.: Beispiel für ein lineares S_i mit $a = 2, b = 1, c = 3$. Der Wert an Stelle $(3, 5)$ erfüllt die Gleichung $4 = (9 + 8 - 3) \bmod 10$

Protokoll 7 Hopper Blum 2 (HB2) [41, S.63] [58, S.27]

Gegeben. $k = (k_1, k_2, d) \in (\mathbb{Z}_n^m \times \mathbb{Z}_n^m, \mathbb{Z}_n^m \times \mathbb{Z}_n^m, \mathbb{Z}_{10})$

Protokollablauf:

1. $\mathcal{H}$ schickt ID an C
2. i=0; Für p Runden:
   a) • C erstellt Zufallsvektor $c \in \mathbb{Z}_{10}^n$ als $c = (S_0, ..., S_{n/100})$
      • Wiederhole solange bis $f(c, k_1) = d$. f ist definiert als $f(c, (a, b)) = \sum_{i=1}^{m} min(c[a_i], c[b_i]) \bmod 10$ ist die Summe der m kleineren Zahlen in c gegeben der in a und b gegebenen Indizes mod 10. (»Sum-of-m-mins«)
   b) C schickt c an $\mathcal{H}$.
   c) Wenn c nicht (nahezu) linear ist oder $f(c, k_1) \neq d$, meldet $\mathcal{H}$ eine Infiltration und bricht ab. Gegebenenfalls korrigiert $\mathcal{H}$ c.
   d) $\mathcal{H}$ schickt $r = f(c, k_2)$ an C
   e) C prüft ob $r = f(c, k_2)$, wenn ja i++
3. C akzeptiert genau dann wenn i=p

Bei linearem S_i gilt für alle Stellen x, y: $S_i(x, y) = S_i(x, 0) + S_i(0, y) - S_i(0, 0) \bmod 10$ [41, S.62]. Um zu entscheiden, ob c nahezu linear ist, werden zufällig gewählte Stellen in den S_i geprüft. Ist für eine dieser Stellen die Gleichung nicht erfüllt, werden alle Stellen überprüft. Weichen nur wenige ab, werden diese korrigiert.

Aufwand und Berechnung durch Menschen Anders als HB1 verlangt HB2 von $\mathcal{H}$ keine Erzeugung von Zufall. $\mathcal{H}$ muss sich zur Ausführung des Protokolls zwei Listen mit m

Elementen merken, die pro Listenelement zwei Werte aus $\mathbb{Z}_n$ beinhalten. Diese Werte werden als Indizes interpretiert. Zusätzlich beinhaltet das Geheimnis eine Zahl d aus $\mathbb{Z}_{10}$. Erhält $\mathcal{H}$ einen Challengevektor als Folge von h Matrizen S_i (vgl. Abbildung 5.1) und prüft pro Matrix durchschnittlich l Stellen auf Linearität, fallen je $h \cdot l$ Additionen, Subtraktionen und Modulo-Operationen an. Die Modulo-Operation kann in der Aufwandsbetrachtung vernachlässigt werden, da es für mod 10 reicht, die letzte Dezimalstelle zu verwenden. Pro Addition und Subtraktion wird ein Aufwand von t_+ angenommen. Für die Berechnung von $f(c, k_1)$ wird ein Akkumulator $s = 0$ verwendet. Für jedes Indexpaar in der ersten geheimen Liste werden die Werte im Challengevektor an diesen Indizes gesucht. Der kleinere der beiden Werte wird auf den Akkumulator s addiert (Aufwand t_+) und s mod 10 gerechnet (ohne Aufwand). Ist der addierte Wert oder s Null, entfällt der Aufwand[3]. t_s soll den Suchaufwand beschreiben, der zum Finden eines Werts an einem bestimmten Index im Challengevektor anfällt. Das Suchen der beiden Werte an den Indizes verursacht durchschnittlich einen Aufwand von $1.9t_s$, denn ist der erste gefundene Wert 0 muss der zweite nicht gesucht werden. Ist die Liste abgearbeitet, steht in s der Wert von $f(c, k_1)$. Ist $s \neq d$ bricht $\mathcal{H}$ ab. Zur Berechnung von $f(c, k_1)$ und $f(c, k_2)$ fällt je ein Aufwand von durchschnittlich $1.9m \cdot t_s + (m-1) \cdot 0.81t_+$ an. Der Aufwand einer Runde beträgt also $2h \cdot l \cdot t_s + 2(1.9m \cdot t_s + (m-1) \cdot 0.81t_+)$. Über p Runden ergibt sich ein Aufwand von:

$$p(2h \cdot l + 3.8m)t_s + 1.62p(m-1)t_+$$

Für $h = 9$ ($n = 900$), $p = 6, m = 12, l = 5$, einem angenommenen Suchaufwand $t_s = 3s$ und $t_+ = 0.5s$ ergibt sich ein Aufwand von $2494.26s \approx 41.5min$! Der Aufwand hängt stark von t_s und l ab. Mit $t_s = 2s$ liegt die Authentifikationszeit bei $1680.66 \approx 28min$. Wird zusätzlich nur eine Position pro S_i auf Linearität geprüft, ergibt sich ein Aufwand von $816.66s = 13.6min$.

Sicherheit Insgesamt sind durch die Redundanz zur Erreichung der Fehlererkennung $10^{(3n/100)-1}$ Challenges möglich [58, S.27]. Ein Rateangriff hat mit Wahrscheinlichkeit $\frac{1}{10^p}$ Erfolg. Um das Geheimnis k_2 zu berechnen, muss ein Angreifer das Sum-of-m-Mins-Problem lösen: Dieses ist wie folgt definiert. Gegeben t Paaren $(v_1, u_1), ..., (v_t, u_t)$ mit $v_i \in \{0, ..., 9\}^n, u_i \in \{0, ..., 9\}$ und $m \log_{10}(n) \leq t \leq \binom{n}{2}$, finde eine Menge z, sodass $u_i = f(v_i, z)$ für alle $i = 1, ..., m$. Ist $t \geq \binom{n}{2}$ wird erwartet, dass sich die Lösung eindeutig durch Gaussche Elimination finden lässt. [7, S.1654] [41, S.59] Andernfalls ist das Problem NP-vollständig, die Average-Case-Komplexität aber unbekannt [7]. Die Komplexität des Problems wird von Hopper & Blum für $n = 900, m = 12, p = 6$ mit $\approx 2^{89}$ angegeben [41, S.63]. Die verwendete Fehlerkorrektur verhindert, dass ein Angreifer unbemerkt lokale Änderungen in c vornehmen kann. Globale Änderungen führen mit Wahrscheinlichkeit $p \geq 0.9$ zu $f(c, k_1) \neq d$, weshalb das Protokoll von Hopper & Blum als $(0.1, 0.1, \binom{n}{2})$-sensitiv bezeichnet wird [41, S.63]. Li et al. [58] weißen darauf hin, dass auch in diesem Protokoll die Zahlen in den Challenges nicht gleichverteilt sind.

[3] Insbesondere für das erste Element der Liste, ansonsten in durchschnittlich $1 - 0.9^2 = 0.19$ der Fälle

5.2.3. Weinshall

Protokoll 8 Weinshall [94, S.2]

Gegeben. $k \subset B$, B ist eine (öffentlich einsehbare) Menge von Bildern

Protokollablauf:

1. $\mathcal{H}$ schickt ID an C
2. i=0; Für m Runden:
   a) C schickt zufällige Teilmenge $F \subset B$ und eine Multiple-Choice-Frage $Q_F(.)$ über F mit P Antwortmöglichkeiten an $\mathcal{H}$
   b) $\mathcal{H}$ schickt $r = Q_F(k)$ an C
   c) Wenn $r = Q_F(k)$, setzt C i++
3. C akzeptiert, wenn $m \geq i \geq t$ (Mindestens $t \leq m$ Fragen richtig beantwortet)

Weinshall unterscheidet zwei Arten von Multiple-Choice-Fragen ($Q_F(.)$):

1. »Komplexe«Fragen, in denen ein Nutzer einen Pfad berechnen muss. Dafür wird die Menge an Challenge-Bildern (F) als Matrix angeordnet. Der Pfad startet links-oben. Wenn das Bild an der aktuellen Position in der geheimen Bildmenge (k) liegt, gehe nach unten, andernfalls nach rechts. Wiederhole diesen Schritt so lange, bis man in der letzten (untersten) Zeile oder (rechtesten) Spalte der Matrix ankommt. Dieser Endposition ist eine nicht eindeutige Zahl $\in \{1, ..., P\}$ (Im Bsp. von [94] $P = 4$) zugeordnet, die der Antwort entspricht [94, S.3].
2. »Einfache«Fragen, in denen über eine kleinere Menge an Challenge-Bildern F eine Aussage getroffen werden muss. Jedem der Challenge-Bilder ist ein Bit zufällig zugeordnet, wobei genau die Hälfte der Bilder eine Eins, die andere Hälfte eine Null hält. Der Nutzer muss nun F in fester Reihenfolge nach Bildern durchsuchen, die auch in k vorkommen. Die Bits von drei Bildern werden betrachtet: Für die Bilder der ersten beiden und der letzten Übereinstimmung. Gefragt wird, ob die Bits mehrheitlich '1' oder '0' sind [94, S.3 f.].

Aufwand Weinshall untersucht die Nutzerfreundlichkeit des Protokolls mit »komplexen«Fragen durch eine nicht-repräsentative Studie mit 9 Studenten. Nach einer zweitägigen Vorbereitung [94, S.4], um sich das Geheimnis k zu merken, brauchte die Beantwortung einer Multiple-Choice-Frage $Q_F(.)$ etwa 15 bis 20 Sekunden [94, S.5]. Die Erfolgsrate lag bei $\approx$ 95%. [94, S.5 §Abb.2]. Die Vorbereitungszeit ist zwar lang, dafür ist das Protokoll streng mensch-ausführbar. Insbesondere wird kein Zufall verwendet. Die Herausforderung für Menschen besteht darin, sich eine Menge von Bildern zu merken und zuverlässig von anderen Bildern unterscheiden zu können. Wird mit $P = 4$ und $\frac{t}{m} \approx 95\%$ für Rateangriffe

eine Erfolgswahrscheinlichkeit von $p \leq 10^{-6}$ gefordert [94, vgl. S.6], werden $m \geq 13$ Fragen pro Authentifikationsvorgang benötigt, damit eine Frage falsch beantwortet werden darf:

$$\sum_{i=12}^{13} \binom{13}{i} \left(\frac{1}{4}\right)^{i} \left(\frac{3}{4}\right)^{13-i} \approx 5.96 \cdot 10^{-7}$$

Die Ergebnisse der Studie legen also eine Ausführungszeit von etwa $\approx 230s \approx 4$ Minuten nahe. Die Wahrscheinlichkeit einer erfolgreichen Authentifikation liegt bei $\sum_{i=t}^{m} \binom{m}{i} \cdot 0.95^{i} \cdot (0.05)^{m-i} \approx 86.45\%$ Werden keine Fehler zugelassen, reicht $m = 10$ und die Erfolgswahrscheinlichkeit beträgt $0.95^{10} \approx 59.87\%$.

Sicherheit Es sind $\binom{|B|}{|F|}$ Multiple-Choice-Fragen möglich. Selbst mit kleinen Werten $|B|, |F|$, können gleiche Fragen und einhergehende Replay-Angriffe vermieden werden ($\binom{50}{10} \approx 10^{10}$). Ein Rateangriff hat mit Wahrscheinlichkeit

$$\sum_{i=t}^{m} \binom{m}{i} \left(\frac{1}{P}\right)^{i} \left(1 - \frac{1}{P}\right)^{m-i}$$

Erfolg.[4] Um Wörterbuchangriffe zu vermeiden, wird die Menge k vom Computer generiert und dem Nutzer zugewiesen [94, S.2, S.4]. Ein Brute-Force-Angriff hat Komplexität $\binom{|B|}{|k|}$ [94, S.4]. Weinshall merkte für »komplexe«Fragen bereits an, dass die geheime Menge k viel effizienter berechnet werden kann, als mit Brute-Force [94, S.4 f., §Tab.1], gab aber weder für »komplexe«$Q_F(.)$ noch »einfache«$Q_F(.)$ einen Angriff an, außer dem Hinweis, dass das Protokoll für einfache Fragen anfällig für probabilistische Angriffe ist, da die Antwort auf die Frage von nur drei Bildern abhängt [94, S.4]. Bereits im Jahr der Veröffentlichung konnten Golle et al. [35] unter Verwendung eines SAT-Solvers das Verfahren brechen. Die Menge k lässt sich selbst mit großen Parameteren $|B|, |F|$ und $|k|$ innerhalb weniger Minuten, teilweise wenigen Sekunden rekonstruieren. Die Anzahl der dafür benötigten Challenge-Response-Paare entspricht den Transkripten weniger Authentifikationen: Mit $|B| = |F| = 80, |k| = 30, P = 4$, in denen die Bilder in einer 8×10-Matrix angeordnet waren (»komplexe«Fragen), wurden nur 60 Paare benötigt, um k zu berechnen. (Mit $|B| = 120, |F| = 80, |k| = 45, P = 2$ und gleicher Anordnung: 1000 Paare) [35, S.4, §Tab.1]. Wird für Rateangriffe eine Erfolgswahrscheinlichkeit von $p \leq 10^{-6}$ gefordert, werden pro Authentifikation 10 bzw. für $P = 2$ 20 Challenge-Response-Paare erzeugt. Es reicht einem Angreifer in diesem Fall, nur 6 bzw. 50 Authentifikationen zu beobachten. »Einfache«Fragen bieten keine bessere Sicherheit [35, S.6, §Tab.2]. Das Weinshall-Protokoll bietet also selbst mit großem Aufwand nur Sicherheit vor opportunistischen Angreifern.

5.2.4. Foxtail

Li et al. [57] stellen zwei generische Ansätze für mensch-ausführbare Authentifikationsprotokolle vor: Twins und Foxtail. In Twins schickt C zwei unabhängige Challenges, auf die $\mathcal{H}$ genau eine richtige Antwort schicken muss. Diese Vorgang wird eine definierte Anzahl Runden durchgeführt. Der Ansatz verspricht sich, Informationslecks durch

[4] Wahrscheinlichkeit, $i \geq t$ Fragen richtig zu beantworten.

Replay-Attacken zu verhindern, wofür die Wahrscheinlichkeit einer richtigen Antwort für beliebige Challenge genau 0.5 betragen muss. Twins hat eingeschränkte Nutzbarkeit, da sich der Aufwand verglichen mit dem zugrundeliegenden Protokoll verdoppelt [57, S.11 f.].

Für den zweiten generischen Ansatz Foxtail schlagen Li et al. ein unten besprochenes konkretes Protokoll vor. Ziel von Foxtail ist es, durch eine probabilistische sog. Foxtail Map F, die mehrere Eingaben auf eine Ausgabe mappt, die Antworten von $\mathcal{H}$ mit Unsicherheit zu versehen. Statt mit der »eigentlichen«(durch Challenge und Geheimnis bestimmten) Antwort r_h auf eine Challenge, sendet $\mathcal{H}$ $F(r_h)$ [57, S.12].

Protokoll 9 Foxtail [57, S.12 f.]

Gegeben. $k \subset B$ mit $|k| \geq 3$

Def. Für $B' \subset B$ ist $\mathrm{Sim}_k(B') = |(B' \cap k)|$ die Ähnlichkeit zwischen k und B' als Anzahl der geheimen Elemente in einer Menge B'.

Protokollablauf:

1. $\mathcal{H}$ schickt ID an C
2. i=0; Für m Runden:
   a) C schickt $\{C_1, C_2\} \in B \times B$ mit $|C_1| = |C_2| \geq 3$ an $\mathcal{H}$. C_1 wird dabei zufällig gewählt (Ran-Rule) und C_2 so, dass alle Antworten r_h mit gleicher Wahrscheinlichkeit auftreten (Uni-Rule) [57, S.13, S.17 f.].
   b) $\mathcal{H}$ schickt $r = F(r_h)$ mit $r_h = (\mathrm{Sim}_k(C_1) + \mathrm{Sim}_k(C_2)) \bmod 4$ und $F(r_h) = \lfloor r_h/2 \rfloor$ an C
   c) Wenn $r = \lfloor \frac{(\mathrm{Sim}_k(C_1)+\mathrm{Sim}_k(C_2)) \bmod 4}{2} \rfloor$, setzt C i++
3. C akzeptiert genau dann wenn i=m

Die Formulierung von k als Teilmenge einer beliebigen Menge B lässt verschiedenste Implementierungen zu. Neben dem Protokoll werden in [57] mehrere beispielhafte Implementierungen vorgeschlagen, darunter eine textbasierte, eine grafische und eine als Schachbrett angezeigte Oberfläche, in denen die Stellung dem Passwort entspricht. Eine Implementierung als Spiel kann die Nutzerfreundlichkeit erhöhen [57, S.14]. Wie für HB existiert für Foxtail eine Erweiterung für ressourcenbeschränkte Geräte [70].

Ausführbarkeit In der grafischen Implementierung besteht die Menge B (und daraus folgend k und die Challenges C_1, C_2) aus Icons bzw. Bildern [57, S.14]. Für jede Challenge zählt der Nutzer also die Anzahl der darin vorkommenden geheimen Bilder, rechnet die Anzahl Modulo 4, und antwortet 0, falls das Ergebnis 0 oder 1 ist, andernfalls 1. Für diese Implementierungen werden als Standard-Parameter $|B| = 140, |k| = 14, |C_1| + |C_2| = 30, m = 20$ genannt [57, S.14]. In einer Studie konnten Nutzer das Protokoll mit diesen

Parametern in 3 bis 4 Minuten ausführen [57, S.15]. Li et al. bezeichnen das Protokoll als (0.9,0.1,O(180))mensch-ausführbar [57, S.15].

Sicherheit Es sind $\binom{|B|}{|C_1|+|C_2|}$ Challenges möglich. Für die Standard-Parameter sind das $\binom{140}{30} \approx 3.2 \cdot 10^{30} \approx 2^{101}$ Ein Rateangriff hat mit Wahrscheinlichkeit 0.5^m Erfolg. Werden (wie in [57, S.14] vorgeschlagen) vom Nutzer gewählte Mengen k zugelassen, um die Ausführungszeit zu reduzieren, eröffnen sich Wörterbuchangriffe. Ein Brute-Force-Angriff hat Komplexität $\binom{|B|}{|k|}$ [57, S.17].

Die Generierung von $(\{C_1, C_2\})$ so dass für alle möglichen $r_h \in \{0, 1, 2, 3\}$ die Wahrscheinlichkeit $\frac{1}{4}$ beträgt, soll statistische Angriffe verhindern (Schritt 2.a im Protokoll). Tatsächlich korreliert das Erscheinen einzelne Elemente in Challenges nicht mit der Antwort r_h [96]. Wird aber das Auftreten von Element-Paaren betrachtet, lässt sich eine Korrelation für einen statistischen Angriff feststellen [96, S.8 f.]. Mit den von Li et al. vorgeschlagenen Standard-Parametern reichen durchschnittlich 711 Authentifikationen (14219.4 Runden), um die geheime Menge k exakt zu bestimmen [96, S.9]. Nach 540 Authentifikationen (10799.8 Runden) konnten Yan et al. bereits 90% der Elemente in k bestimmen [96, S.9]. Ein angepasstes Foxtail-Protokoll, das diesem statistischen Angriff standhalten sollte, hat sich als noch unsicherer erwiesen [8]. Auch wenn dieses Ergebniss weit unter der von Li et al. anvisierten Sicherheit liegt und das Foxtail-Protokoll auf Sicherheit vor opportunistischen Angreifern zurückfällt, ist es wesentlich sicherer als das Weinshall-Protokoll. Zudem benötigt das Protokoll keinen Zufall oder komplexe Berechnungen, ist also streng mensch-ausführbar. Wird das Geheimnis k nach einer besimmten Anzahl (z.B. 100) Authentifikationen erneuert, wäre die Nutzung des Protkolls möglicherweise gerechtfertigt. Zu beachten ist allerdings, dass die Sicherheit mit jeder erfolgreichen Authentifikation sinkt. Überlegungen zur aktiven Sicherheit erübrigen sich weitestgehend durch den passiven Angriff. Unkar ist, wie weit sich die Anzahl der benötigten Transkripte im statistischen Angriff von [96] durch aktive Interventionen verringern lässt.

5.2.5. Catuogno und Galdi (CG)

Protokoll 10 Catuogno und Galdi (CG) [22, S.21 f.]

Gegeben. $k = \{k_1, ..., k_m\} \subset B$, B ist eine Menge von Bildern, c Anzahl der korrekt zu beantwortenden Fragen ($1 \leq c \leq m$)
Protokollablauf:

1. $\mathcal{H}$ schickt ID an C
2. C schickt zufällige Partitionierung von B, also disjunkte Teilmengen $(F_1, ..., F_a) \subset B^a$ gleicher Größe ($|F_1| = ... = |F_a| = \frac{|B|}{a}$) an $\mathcal{H}$
3. $\mathcal{H}$ wählt zufällig c der m Elemente aus k, setzt $\rho_i = \arg\mathrm{t} : k_i \in F_t$ für diese und zufällig falsche Werte aus $\{1, ..., a\}$ für die restlichen ρ_i und schickt $(\rho_1, ..., \rho_m)$ an C
4. C akzeptiert genau dann wenn für genau c ρ_i: $k_i \in F_{\rho_i}$

$\mathcal{H}$ beantwortet also für eine bestimmte Azahl c aus m geheimen Elementen die i-te Frage: »In welcher Teilmenge befindet sich das i-te geheime Element?«. Das Protokoll oben beschreibt das Vorgehen, wenn für $m - c$ Fragen bewusst falsch statt zufällig geantwortet wird. Eine von Catuogno et al. beschriebene Variante antwortet auf $m - c$ Fragen rein zufällig. In diesem Fall akzeptiert C, wenn mindestens c der Fragen richtig beantwortet werden.

Ausführbarkeit Für die Protokollausführung wird Zufall benötigt. In einer weiteren Variante des Protokolls wird deterministisch in jeder Runde neu festgelegt, welche Fragen richtig, und welche falsch beantwortet werden müssen (folgend als CG2 bezeichnet). Hierbei wird davon ausgegangen, dass der Nutzer eine Smartcard besitzt, die die Auswahl pseudozufällig erzeugt. Smartcards gehen über die Definition der Mensch-Ausführbarkeit hinaus, möglicherweise lässt sich die Smartcard aber durch menschliche Berechnung ersetzen. Die strenge Mensch-Ausführbarkeit ist in jedem Fall nicht gegeben. Dadurch dass C ebenfalls die Auswahl kennt, steigt die Sicherheit vor Rateangriffen. Diese Variante ist aber nur sinnvoll, wenn davon ausgegangen wird, dass ein Angreifer die erzeugt Auswahl der Smartcard/PRG nicht beobachten kann [22, S.30]. Der Grund für diese Überlegung zeigt sich beim Betrachten von Rateangriff-Wahrscheinlichkeit und der Möglichkeit, Rückschlüsse über k zu ziehen. Wird c groß gewählt, erhält der Angreifer schnell Informationen über k. Wird dagegen $c \approx \frac{m}{a}$ gewählt, ist offensichtlich die Wahrscheinlichkeit hoch, dass ein Rateangriff erfolgreich ist [22].

Sicherheit Ein Rateangriff hat im beschriebenen Protokoll mit Wahrscheinlichkeit

$$\binom{m}{c}\left(\frac{1}{a}\right)^c\left(1-\frac{1}{a}\right)^{m-c}$$

Erfolg [22, vgl. S.31]. Die Brute-Force-Komplexität liegt bei $\binom{|B|}{m}$. Auch CG2 mit $c \approx \frac{m}{a}$ bietet keine hohe Sicherheit. Asghar et al. [8] beschreiben, wie sich CG2 linearisieren lässt. Sind 80 Challenge-Response-Paare bekannt, kann mit Gausscher Elimination die geheime Menge k in 80% der Fälle eindeutig bestimmt werden (Parameter: $|B| = 80, m = 15, c = 7, a = 2$) [8, S.1649]. Da CG/CG2 in jeder Authentifikation nur ein Challenge-Response-Paar erzeugt, entspricht dies 80 Authentifikationen. Auch CG erreicht also nur Sicherheit vor opportunistischen Angreifern. Wird $c = m$ ($a = 2$) gewählt, also keine Auswahl getroffen, kann das Geheimnis k nach durchschnittlich ≈ 10 beobachteten Authentifikationen berechnet werden. Für größeres a sinkt die Anzahl nötiger Transkripte weiter [22, S.23 f.].

5.2.6. iChip und weitere Protokolle

Seit CG wurden einige weitere Protokolle vorgestellt. Die oben vorgestellten Protokolle sind entweder nur sicher gegen opportunistische Angreifer oder unpraktikabel [6]. Als Antwort auf den Trade-Off zwischen Sicherheit und Nutzerfreundlichkeit [25, S.3] [96] richten sich tatsächlich die meisten Protokolle gegen opportunistische Angriffe [25, 97] und fokusieren sich auf die Nutzbarkeit, etwa als Ersatz zur klassischen PIN-Eingabe [81, 15, 14, 26]. Oder aber die kryptographische Analyse zeigt Schwächen, die wiederum die Sicherheit auf opportunistische Angriffe beschränken [24, 95].

Ein von Asghar et al. [6] vorgestelltes Protokoll, das passive Sicherheit erreichen soll, wird als (.,.,213)-mensch-ausführbar bezeichnet. Das Passwort besteht dabei aus einem Pfad aus einer vertikalen und mehreren horizontalen Bewegungen in einer beliebigen Challengematrix [6, S.353 §Protocol 1]. Asghar et al. gehen in ihrer Sicherheitsanalyse davon aus, dass sich dieses Problem nicht leicht linearisieren lässt [6, S.356 f.]. Blocki et al. [16] stellen Protokolle zur Generierung von Passwörtern vor, die ebenfalls als Identifikationsprotokolle verwendet werden könnten. Dabei wird dem Menschen $\mathcal{H}$ die Verwendung eines semi-vertrauenswürdigen Computers zugestanden. Ein semi-vertauenswürdiger Computer ist ein Computer, der Berechnungen korrekt ausführt aber keine Vertraulichkeit gewährleistet.

Ein kürzlich vorgestelltes Protokoll von Matelski [62] möchte mit iChip den Trade-Off zwischen Sicherheit und Nutzerfreundlichkeit behoben haben. Die formale Beschreibung von iChip ist deutlich komplexer als die der bisher beschriebenen Protokolle. Um dennoch Nutzerfreundlichkeit zu gewährleisten, ist eine informelle Beschreibung notwendig. Die folgende informelle Beschreibung von iChip basiert auf der formalen Beschreibung in [62, S.66 f.].

Auf einem Feld bestehend aus $c \times r$ Blöcken definiert ein Nutzer $\mathcal{H}$ sein Geheimnis: Das Geheimnis besteht aus m Initialblöcken, m Pfadabschnitten (Menge von Blöcken, müssen nicht zusammenhängend sein) und m Wurmloch-Blockmengen. Nicht alle Blöcke müssen in eine dieser Mengen fallen. C sendet nun eine Challenge, bestehend aus einer Zahlenbelegung (0 bis 9) aller Blöcke. $\mathcal{H}$ liest den Wert ψ des ersten Initialblocks, merkt sich diesen und fängt an, sich auf den Weg zu machen. $\mathcal{H}$ beginnt die Pfadabschnitte abzulaufen auf der Suche nach diesem Wert. Beginnend bei dem ersten Pfadabschnitt so lange, bis ein Block auf einem der Abschnitte dem Wert des Initialblocks entspricht. Auf dem ϕ-ten Pfadabschnitt wird er fündig. Doch plötzlich öffnet sich das Wurmloch. $\mathcal{H}$ spürt,

wie sich über ihm der Himmel öffnet. Panisch schaut er auf den Weg, der noch vor ihm liegt, bis der nächste Pfadabschnitt beginnt. In seiner Panik und um sich abzulenken, summiert er die Werte der noch vor ihm liegenden Blöcke bis zum Ende des Pfadabschnitts (allerdings maximal 2) auf. [5] Das Wumloch schluckt ihn und spuckt ihn auf einem beliebigen[6] Block der ϕ-ten Wurmloch-Blockmenge aus. Als $\mathcal{H}$ wieder zu sich kommt, kann er sich nur noch an zwei Dinge erinnern: Die aufsummierte Zahl und der Wert ψ, nachdem er gesucht hat. Verwirrt stellt er fest, dass er sich an den Block auf dem er sich befindet, überhaupt nicht erinnern kann. $\mathcal{H}$ liest den Wert des Blocks ab, auf dem er steht. Da es nichts Besseres zu tun gibt, summiert er alle Informationen, die er nun kennt, auf: Der Wert des Initialblocks, die Summe des verbliebenen Pfadabschnitts [7] und den Wert des Wurmloch-Blocks, auf dem er sich befindet. Da das in dem Moment viel zu viele Informationen sind, rechnet $\mathcal{H}$ die Summe mod 10. Dieser Wert ist die erste Zahl der Antwort an C. Eine Wurmloch-Erfahrung kommt selten allein. Deshalb wiederholt sich für $\mathcal{H}$ die Erfahrung m mal. In der j-ten Runde ($j = 1, ..., m$) wird der j-te Initialblock verwendet. Ist in der Suche auf den Pfadabschnitten der Wert nicht vorhanden, dann wird in dieser Runde 2ψ mod 10 ausgegeben. Um die mögliche Schlüsselmenge weiter zu vergrößern, kann eine Wurmloch-Blockmenge auch leer sein. Findet $\mathcal{H}$ den Wert des Initialblocks auf einem Pfadabschnitt, dessen Wurmloch-Blockmenge leer ist, geht er auf dem Pfadabschnitt einen Schritt weiter (wenn am Ende des Pfadabschnitts, kommt er wieder am Anfang des gleichen Abschitts raus), verwendet den dortigen Wert, addiert dazu den Wert des Initialblocks Modulo 10 und gibt diesen aus. [8]

In einer Variante des Verfahrens (»Turbochip«) die schnellere Authentifizierung erreichen soll, berechnet $\mathcal{H}$ zunächst die erste Zahl v der Antwort wie oben beschrieben und merkt sich dabei ϕ (also auf dem wievielten Pfadabschnitt der gesuchte Wert ψ gefunden wurde). Die Antwortwerte berechnen sich aus der Addition von v mit dem Wert des $(\psi + 2i)$-ten Blocks im i-ten Pfadabschnitt Modulo 10. [63, S.460] Am Anfang ist $i = 1$ und wird in jedem Schritt um eins hochgezählt. Wenn ein Pfadabschnitt weniger als die erforderlichen $(\psi + 2i) + 1$ Blöcke hat, wird $\psi \leftarrow 1, i \leftarrow 0$ gesetzt.

5.3. Evaluation

Ein Framework für die Aufwandsabschätzung verschiedener Identifikationsprotokolle wird von Yan et al. [96] vorgestellt und berücksichtigt die Zeit für Wiedererkennung, Erinnerung, visuelle Suche und einfache Arithmetik als atomare kognitive Operationen der jeweiligen Protokolle. Das Framework wurde u.a. auf HB1, Foxtail und Weinshall angewendet und führt zu den in Tabelle 5.1 unter 'analy.Aufwand' abgetragenen Ergebnissen. Yan et al. [96] weisen darauf hin, dass die Aufwände für Erinnerung und Berechnung möglicherweise nicht unabhängig voneinander sind, und infolge dessen der Gesamtaufwand überschätzt wird. Zusätzlich ist davon auszugehen, dass sich die Authentifikationszeit bei routinierter Anwendung verringert.

[5] vgl. Ausnahme *I in [62, S.66 f.]
[6] Dies entspricht der 'Ausnahme' *O in [62, S.66 f.]
[7] 0 falls auf letztem Block des Abschnitts.
[8] Dies entspricht der Ausnahme Θ* in [63, S.458]

Protokoll	Angreifermodell	Art der Mensch-Ausführbarkeit
HB1	passiv	schwach
HB2	aktiv	streng
Weinshall 1 (komplex)	opportunistisch	streng
Weinshall 2 (einfach)	opportunistisch	streng
Foxtail	opportunistisch	streng
CG	opportunistisch	schwach
(iChip)	passiv	schwach

Protokoll	Parameter	analy.Aufwand	exp.Aufwand	gesch.Aufw.
HB1				≈300s [41]
	$n = 200, \|k\| = 15, m = 20$	668.45s [96]		
	$n = 200, \|k\| = 15, m = 7$		166s [41]	
HB2	siehe 5.2.2	≈15-45min		
Weinsh.1	$\|F\| = 240, \|k\| = 60, m = 10$	121.46s [96]		
	$\|F\| = 80, \|k\| = 30, m = 1$		≈15-20s [94]	
Weinsh.2	$\|F\| = 60, \|k\| = 30, m = 10$	220.99s [96]		
Foxtail	$\|B\| = 140, \|k\| = 14, m = 20$	212.76 [96]		
	$\|B\| = 140, \|k\| = 14, m = 20$		180-240s [58]	≈180s [58]
CG (iChip)	$c \cdot r = 256, n = 30, m = 6$		36s [62]	

Tabelle 5.1.: Erreichbare Sicherheit und Aufwand der vorgestellten Protokolle. Aufgeschlüsselt nach analysiertem/experimentellem/geschätztem Aufwand.

Für HB2 und CG sind keine experimentellen Ergebnisse für die Identifikationszeit durch Menschen bekannt. HB2 wird selbst von Hopper & Blum als zu aufwändig bezeichnet, um praktische Relevanz anzunehmen [41, S.63]. Dies deckt dich mit den Ergebnissen der Aufwandsabschätzung in Unterabschnitt 5.2.2. CG wurde für die Nutzung mit Smart-Cards (nicht mensch-ausführbar) konzipiert. Bis auf eine zweitägige Vorbereitung für Weinshall sind keine weiteren Vorbereitungszeiten der experimentellen Ergebnisse bekannt. Außerdem fehlt es an Studien oder Berechnungsgrundlagen, um jeweils α (Anteil Bevölkerung) und β (Erfolgswahrscheinlichkeit) in der $(\alpha, \beta, \gamma_1, \gamma_2)$-Mensch-Ausführbarkeit anzugeben. Hopper & Blum [41] sowie Li et al. [57] gehen mit $\alpha = 0.9$ davon aus, das lediglich 10% der Bevölkerung mit einer Erfolgswahrscheinlichkeit von 80% (90%) in 300s (180s) ausführen können.

Es existiert ein Trade-Off zwischen erreichbarer Sicherheit und Nutzbarkeit der mensch-ausführbaren Identifikationsprotokolle: Yan et al. [96] stellen fest, dass für das Reduzieren des Informationslecks mehrere Prinzipien beachtet werden müssen, die die Nutzbarkeit negativ beeinflussen. Zu diesen Prinzipien gehört u.a. dass Elemente des Challengevektors so gleichverteilt wie möglich sein sollten und der Passwortraum K möglichst groß. Wie in Tabelle 5.1 zu sehen, ist unter den beschriebenen Verfahren nur HB1 passiv sicher und benötigt mit Ausnahme von HB2 den höchsten Aufwand. Ob iChip diesen Trade-Off tatsächlich behebt, müssen Sicherheitsanalysen des Protokolls zeigen. Während HB1 Informationslecks durch zufällig falsche Antwortwerte einschränkt (LPN) [96], versucht iChip Informationslecks durch Learning with Options (LWO) [62] klein zu halten. Beide Ansätze benötigen Zufall. Keines der streng mensch-ausführbaren Protokolle erreicht Sicherheit vor passiven Angreifern. Unter den opportunistisch-sicheren Protokollen erreicht Foxtail die höchste Sicherheit. Wird das Geheimnis regelmäßig erneuert (z.B. alle 100 Authentifikationen), kann passive Sicherheit erreicht werden.

Čagalj et al. [19] beschreiben einen Seitenkanalangriff auf HB1 basierend auf der benötigten Berechnungszeit für einzelne Challenges. Die Addition kleinerer Zahlen führt zu schnelleren Antworten [19, S.586 f.]. Derartige Seitenkanalangriffe können auf alle Protokolle angewendet werden, die die Addition von Zahlen (HB1, HB2, iChip) oder andere Operationen beinhalten, deren Aufwand von den Operanden abhängt. Z.B. könnte in den Weinshall- und Foxtail-Protokollen ausgenutzt werden, dass Bilder die in einer Challenge an füherer Stelle vorkommen schneller gefunden werden. Als mögliche Gegenmaßnahme kann (wiederum auf Kosten der Ausführungszeit) eine minimale Wartezeit für die Eingabe von Challenge-Antworten eingeführt werden [19, S.593]. Auch Seitenkanalangriffe, die Handbewegungen auf grafisch dargestellten Challenges ausnutzen, sind möglich [23].

Die langen Identifikationszeiten sprechen gegen einen weitverbreiteten bzw. regelmäßigen Einsatz der vorgestellten Protokolle. Vorstellbar ist die Verwendung der Protokolle parallel zur klassischen Identifikation via Passwort, etwa wenn einem Eingabegerät nicht vertraut wird[9] oder ein Nutzer in gewissen Zeitintervallen seine Identität beweisen soll [10]. Für die praktische Implementierung derartiger Protokolle werden von Chakraborty et al. [25] 'Honeywords' vorgeschlagen, um Angreifer-Aktivitäten aus einem Verlust der

[9] z.B. in Internet-Cafes
[10] z.B. im Online-Banking

Passwörter auf Serverseite erkennen zu können. Ein 'Honeyword' ist dabei ein falsches Passwort, das bei Verwendung C signalisiert, dass Passwörter geleakt wurden [47].

6. Fazit

Zufall erzeugen zu können, ist ein wichtiges kryptographisches Primitiv, etwa bei der Generierung von Schlüsseln, Nonces (LC4) oder der Auswahl der Antworten in Identifikationsverfahren (HB1, CG, iChip). Ob Menschen ohne Hilfsmittel sicher Zufall generieren können, ist eine offene Frage, zu der es bislang an Forschung fehlt.

Bis auf ein Modell von Blum et al. [17], das für die Primitve in Abschnitt 3.2 strenge Mensch-Ausführbarkeit zeigen soll, wird für die Mensch-Ausfühbarkeit anhand einfacher Beschreibungen der Verfahren argumentiert. Die dabei erwarteten Fähigkeiten umfassen Zählen (STML, Solitaire, LC4, IChip), Addition kleiner Zahlen (OTP, STML, HB1, HB2, iChip), Entscheidung ob eine Zahl größer oder kleiner einer anderen Zahl ist (STML, Solitaire), Modulo-Rechnen kleiner Zahlen (STML, OTP, Solitaire, HB1, HB2, Foxtail, iChip), feststellen ob eine Zahl gerade oder ungerade ist (Weinshall), das Suchen und Finden von Zeichen/Zahlen in Kartendecks/Spielsteinen/Vektoren/Mengen (Solitaire, Weinshall, Foxtail, CG, iChip), visuelle Fähigkeiten (VC), motorische Fähigkeiten (VC, Solitaire, LC4), Auswendiglernen von Listen und Abbildungen (STML, iChip) oder anderen Geheimnissen und generell Zufallsgenerierung. Yan et al. [96] stellen ein Framework für die Aufwandsabschätzung einiger atomarer Operationen vor. Da ein Modell für die Berechnung von α (Anteil Bevölkerung) und β (Erfolgswahrscheinlichkeit) in der $(\alpha, \beta, \gamma_1, \gamma_2)$-Mensch-Ausführbarkeit fehlt [55, vgl.], besteht nur die Möglichkeit diese Werte empirisch zu bestimmen.

Die vorgestellten Verschlüsselungsverfahren sind höchstens schwach mensch-ausfühbar. Dass keine streng mensch-ausführbaren Verfahren bekannt geworden sind, liegt mutmaßlich daran, dass dazu für die Verschlüsselung sowohl der Schlüssel als auch der Klartext im Gedächtnis bleiben müssen. Visuelle Kryptographie (VC) ist das einzige Verfahren, das auch ohne nennenswerten Aufwand ausgeführt werden kann. VC ist nur unterstützt mensch-ausführbar. Den geringsten Rechenaufwand und die höchste Fehlertoleranz haben One-Time-Pads. Solitaire kann für kurze Nachrichten verwendet werden, und benötigt dafür das Merken einer Kartendeck-Permutation. LC4 ist das einzige Verfahren, das explizit Integrität und Authentifikation sicherstellen möchte und dessen Schlüssel wiederverwendbar sein sollen. LC4 hat sich als unsicher erwiesen. Für 200-Zeichen-lange Nachrichten sind IND-CPA Angriffe mit Erfolgswahrscheinlichkeit $\approx$ 99.8% und Known-Plaintext-Angriffe mit etwa 52Bit-Komplexität möglich. Schwächen in der Authentifikation und die Möglichkeit zu Broadcast-Angriffen sind weitere Probleme von LC4. Ob diese Probleme durch kleine Anpassungen des Verschlüsselungsalgorithmus behoben werden können ist fraglich.

In [58] argumentieren Li et al. für die Wichtigkeit strenger Mensch-Ausführbarkeit der Identifikationsprotokolle. Dies hat jedoch auch zur Folge, dass nicht auf Zufall zur Begrenzung des Informationslecks pro Identifikation vertraut werden darf. Das betrifft sowohl das 2001 veröffentlichte HB1 Protokoll als auch das von Matelski 2022 vorgestellte

iChip-Protokoll, die beide passive Sicherheit erreichen möchten. HB2 erreicht als einziges Protokoll Sicherheit vor aktiven Angreifern. Von den opportunistisch-sicheren Protokollen erreicht Foxtail die höchste Sicherheit.

Die langen Identifikationszeiten sprechen gegen den alltäglichen Einsatz der vorgestellten Identifikationsprotokolle. Ein eigenes Interesse der protokollausführenden Personen an der sicheren Ausführung ist unabdingbar. Nicht nur die Wahl schwacher Passwörter kann die Sicherheit kompromittieren [90, 45], auch der bewusste Verzicht auf zufällige Antworten in Challenge-Response-Verfahren schränken die erreichbare Sicherheit stark ein. Ein sinnvoller Einsatz der Identifikationsprotokolle ist zudem nur in Bereichen möglich, in denen eine Identifikation regelmäßig vorkommt. Andernfalls kann auf Maßnahmen zur Reduzierung des Informationslecks verzichtet und ein Einmalpasswort verwendet werden. Durch die langen Identifikationszeiten muss vom Gebrauch in Situationen, in denen die Identifikation schnell erfolgen muss, abgeraten werden. Als Anwendungsbereich ist der Einsatz zur Zugangskontrolle zu besonders sensiblen Informationen vorstellbar. Eine weitere Möglichkeit ist der Einsatz der Protokolle als freiwillige Anmeldeoption in Situationen, in denen eine Anmeldung mit Passwort vom Nutzer oder System als unsicher wahrgenommen wird.

Informationen auch ohne Computer verschlüsseln zu können, kann in Situationen hilfreich sein, in denen keine vertrauenswürdigen Hilfsmittel vorhanden sind. Auch wenn Verschlüsselung in Zeiten des Internets alltäglich zur Anwendung kommt, [40] ist eine weitverbreitete Verwendung mensch-ausführbarer Verschlüsselungsverfahren nicht absehbar. Ein Einsatz in nachrichtendienstlichen oder diplomatischen Angelegenheiten wäre als Fallback zur computergestützten Verschlüsselung denkbar. Etwa, wenn während längerer Stromausfälle Informationen sicher ausgetauscht werden sollen, oder der Besitz technischer Geräte untersagt oder Geräte entzogen werden.

Die in dieser Arbeit betrachtete mensch-ausführbare Kryptographie ist alltagsuntauglich. Trotzdem sind mit ihr einige interessante Fragen verbunden. Ob Menschen ohne Hilfsmittel Zufall erzeugen können, ist eine solche Frage, die es noch zu beantworten gilt. Ebenso wie die Existenz eines für Menschen schnell ausführbaren authentifizierten Verschlüsselungsverfahrens, dessen Schlüssel mehrfach verwendet werden darf. Oder die Frage, ob ein schnelles Identifikationsverfahren existiert, dass auch Schutz vor aktiven Angreifern bietet. Die gewonnenen Erkentnisse ließen sich mit hoher Wahrscheinlichkeit auch auf andere Forschungsfelder übertragen. Erweiterungen von HB1 und Foxtail wurden beispielsweise für ressourcenbeschränkte Geräte entwickelt [48, 61, 70]. Die Eigenheiten menschlicher Zufallsgeneration wurden für biometrische Identifikation genutzt [86, 46].

Wie oben skizziert, bedeutet der Mangel an alltäglichen Anwendungsfällen nicht, dass keine sinnvolle Nutzung der vorgestellten Protokolle möglich ist. Wichtig ist in allen Fällen, dass sich der Mensch darüber im Klaren ist, dass die Sicherheit vom eigenen Verhalten abhängt. Dazu gehört die Erkentniss, dass Passwörter zufällig erstellt werden müssen, dass Zufall sicher generiert werden muss, dass Hilfsmittel vernichtet oder randomisiert werden müssen und dass unterschiedliche Berechnungszeiten und Verhaltensmuster für Seitenkanalangriffe genutzt werden können.

Literatur

[1] Sattar Aboud. „An Improved Knapsack Public Key Cryptography System“. In: *International Journal of Internet Technology and Secured Transactions* Vol. 3 (Jan. 2011), S. 310–319. DOI: 10.1504/IJITST.2011.041298.

[2] Leonard M. Adleman. „On Breaking the Iterated Merkle-Hellman Public-Key Cryptosystem“. In: *Advances in Cryptology*. Hrsg. von David Chaum, Ronald L. Rivest und Alan T. Sherman. Boston, MA: Springer US, 1983, S. 303–308. ISBN: 978-1-4757-0602-4. DOI: 10.1007/978-1-4757-0602-4_29.

[3] Nadhem AlFardan u. a. „On the Security of RC4 in TLS“. In: *22nd USENIX Security Symposium (USENIX Security 13)*. Washington, D.C.: USENIX Association, Aug. 2013, S. 305–320. ISBN: 978-1-931971-03-4. URL: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/alFardan.

[4] Mohsen Alimomeni, Reihaneh Safavi-Naini und Setareh Sharifian. „A True Random Generator Using Human Gameplay“. In: *Decision and Game Theory for Security*. Hrsg. von Sajal K. Das, Cristina Nita-Rotaru und Murat Kantarcioglu. Cham: Springer International Publishing, 2013, S. 10–28. ISBN: 978-3-319-02786-9.

[5] Jason Andress. „Chapter 1 - What is Information Security?“ In: *The Basics of Information Security (Second Edition)*. Hrsg. von Jason Andress. Second Edition. Boston: Syngress, 2014, S. 1–22. ISBN: 978-0-12-800744-0. DOI: 10.1016/B978-0-12-800744-0.00001-4.

[6] Hassan Jameel Asghar, Josef Pieprzyk und Huaxiong Wang. „A New Human Identification Protocol and Coppersmith's Baby-Step Giant-Step Algorithm“. In: *Applied Cryptography and Network Security*. Hrsg. von Jianying Zhou und Moti Yung. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 349–366. ISBN: 978-3-642-13708-2. DOI: 10.1007/978-3-642-13708-2_21.

[7] Hassan Jameel Asghar, Josef Pieprzyk und Huaxiong Wang. „On the Hardness of the Sum of k Mins Problem“. In: *Comput. J.* 54.10 (Okt. 2011), S. 1652–1660. ISSN: 0010-4620. DOI: 10.1093/comjnl/bxr070.

[8] Hassan Jameel Asghar u. a. „On the Linearization of Human Identification Protocols: Attacks Based on Linear Algebra, Coding Theory, and Lattices“. In: *IEEE Transactions on Information Forensics and Security* 10.8 (2015), S. 1643–1655. DOI: 10.1109/TIFS.2015.2421875.

[9] Bengt Aspvall, Michael F. Plass und Robert Endre Tarjan. „A linear-time algorithm for testing the truth of certain quantified boolean formulas“. In: *Information Processing Letters* 8.3 (1979), S. 121–123. ISSN: 0020-0190. DOI: 10.1016/0020-0190(79)90002-4.

[10] William Bains. „Random number generation and creativity“. In: *Medical Hypotheses* 70.1 (2008), S. 186–190. ISSN: 0306-9877. DOI: 10.1016/j.mehy.2007.08.004.

[11] Craig P. Bauer. *Secret History: The Story of Cryptology*. Chapman & Hall/CRC, 2013. ISBN: 9781466561861.

[12] Steven Bellovin. „Frank Miller: Inventor of the One-Time Pad“. In: *Cryptologia* 35 (Juli 2011), S. 203–222. DOI: 10.1080/01611194.2011.583711.

[13] Rajat Bhatnagar und Manoj Kumar. „Visual Cryptography: A Literature Survey“. In: *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. 2018, S. 78–83. DOI: 10.1109/ICECA.2018.8474649.

[14] Farid Binbeshr u. a. „A systematic review of PIN-entry methods resistant to shoulder-surfing attacks“. In: *Computers & Security* 101 (2021), S. 102116. ISSN: 0167-4048. DOI: 10.1016/j.cose.2020.102116.

[15] Farid Binbeshr u. a. „Secure PIN-Entry Method Using One-Time PIN (OTP)“. In: *IEEE Access* PP (Jan. 2023), S. 1–1. DOI: 10.1109/ACCESS.2023.3243114.

[16] Jeremiah Blocki u. a. *Towards Human Computable Passwords*. 2016. arXiv: 1404.0024 [cs.CR].

[17] Manuel Blum und Santosh Vempala. *The Complexity of Human Computation: A Concrete Model with an Application to Passwords*. 2017. arXiv: 1707.01204 [cs.HC].

[18] Robert G Brown, Dirk Eddelbuettel und David Bauer. „Dieharder“. In: *Duke University Physics Department Durham, NC* (2018), S. 27708–0305.

[19] Mario Čagalj, Toni Perković und Marin Bugarić. „Timing Attacks on Cognitive Authentication Schemes“. In: *IEEE Transactions on Information Forensics and Security* 10.3 (2015), S. 584–596. DOI: 10.1109/TIFS.2014.2376177.

[20] Jose Carrijo u. a. *A Novel Probabilistic Passive Attack on the Protocols HB and HB+*. Cryptology ePrint Archive, Paper 2008/231. 2008. URL: https://eprint.iacr.org/2008/231.

[21] J.Lawrence Carter und Mark N. Wegman. „Universal classes of hash functions“. In: *Journal of Computer and System Sciences* 18.2 (1979), S. 143–154. ISSN: 0022-0000. DOI: 10.1016/0022-0000(79)90044-8.

[22] Luigi Catuogno und Clemente Galdi. „A Graphical PIN Authentication Mechanism with Applications to Smart Cards and Low-Cost Devices“. In: *Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks*. Hrsg. von Jose A. Onieva u. a. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 16–35. ISBN: 978-3-540-79966-5. DOI: 10.1007/978-3-540-79966-5_2.

[23] Nilesh Chakraborty, Vijay S. Anand und Samrat Mondal. „Towards identifying and preventing behavioral side channel attack on recording attack resilient unaided authentication services“. In: *Computers & Security* 84 (2019), S. 193–205. ISSN: 0167-4048. DOI: 10.1016/j.cose.2019.03.019.

[24] Nilesh Chakraborty und Samrat Mondal. *Lighting Two Candles With One Flame: An Unaided Human Identification Protocol With Security Beyond Conventional Limit*. 2017. arXiv: 1705.10747 [cs.CR].

[25] Nilesh Chakraborty und Samrat Mondal. „On designing an unaided authentication service with threat detection and leakage control for defeating opportunistic adversaries“. In: *Frontiers of Computer Science* 15 (Apr. 2021). DOI: 10.1007/s11704-019-9134-9.

[26] Nilesh Chakraborty u. a. „On Overcoming the Identified Limitations of a Usable PIN Entry Method“. In: *IEEE Access* PP (Aug. 2019), S. 1–1. DOI: 10.1109/ACCESS.2019.2937948.

[27] Arup Kumar Chattonadhvav u. a. „Visual Cryptography: Review and Analysis of Existing Methods“. In: *2018 Global Wireless Summit (GWS)*. 2018, S. 236–241. DOI: 10.1109/GWS.2018.8686653.

[28] Yang-Wai Chow u. a. „A Visual One-Time Password Authentication Scheme Using Mobile Devices“. In: *Information and Communications Security*. Hrsg. von Lucas C. K. Hui u. a. Cham: Springer International Publishing, 2015, S. 243–257. ISBN: 978-3-319-21966-0. DOI: 10.1007/978-3-319-21966-0_18.

[29] Paul Crowley. *Mirdek: a card cipher inspired by SSolitaire"*. 2000. URL: http://www.ciphergoth.org/crypto/mirdek/ (besucht am 03. 12. 2023).

[30] Paul Crowley. *Problems with Bruce Schneier's SSolitaire"*. 1999. URL: http://www.ciphergoth.org/crypto/solitaire/ (besucht am 17. 11. 2023).

[31] Małgorzata Figurska, Maciej Stańczyk und Kamil Kulesza. „Humans cannot consciously generate random numbers sequences: Polemic study“. In: *Medical Hypotheses* 70.1 (2008), S. 182–185. ISSN: 0306-9877. DOI: 10.1016/j.mehy.2007.06.038.

[32] Andrea G. Forte u. a. „EyeDecrypt — Private Interactions in Plain Sight“. In: *Security and Cryptography for Networks*. Hrsg. von Michel Abdalla und Roberto De Prisco. Cham: Springer International Publishing, 2014, S. 255–276. ISBN: 978-3-319-10879-7. DOI: 10.1007/978-3-319-10879-7_15.

[33] Nils Kopal George Lasry Moshe Rubin und Arno Wacker. „Cryptanalysis of Chaocipher and solution of Exhibit 6“. In: *Cryptologia* 40.6 (2016), S. 487–514. DOI: 10.1080/01611194.2015.1091797.

[34] Nils Kopal George Lasry und Arno Wacker. „Solving the Double Transposition Challenge with a Divide-and-Conquer Approach“. In: *Cryptologia* 38.3 (2014), S. 197–214. DOI: 10.1080/01611194.2014.915269.

[35] Philippe Golle und David Wagner. *Cryptanalysis of a Cognitive Authentication Scheme*. Cryptology ePrint Archive, Paper 2006/258. 2006. URL: https://eprint.iacr.org/2006/258.

[36] Ran Halprin und Moni Naor. „Games for extracting randomness“. In: *Proceedings of the 5th Symposium on Usable Privacy and Security*. SOUPS '09. Mountain View, California, USA: Association for Computing Machinery, 2009. ISBN: 9781605587363. DOI: 10.1145/1572532.1572548.

[37] Kimmo Halunen und Outi-Marja Latvala. „Review of the use of human senses and capabilities in cryptography“. In: *Computer Science Review* 39 (2021), S. 100340. ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2020.100340.

[38] Jaak Henno, Hannu Jaakkola und Jukka Mäkelä. „Creating Randomness with Games“. In: *2019 IEEE 23rd International Conference on Intelligent Engineering Systems (INES)*. 2019, S. 000331–000338. DOI: 10.1109/INES46365.2019.9109491.

[39] Gottfried Herold und Alexander Meurer. „New Attacks for Knapsack Based Cryptosystems“. In: *Security and Cryptography for Networks*. Hrsg. von Ivan Visconti und Roberto De Prisco. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 326–342. ISBN: 978-3-642-32928-9. DOI: 10.1007/978-3-642-32928-9_18.

[40] Ralph Holz u. a. „TLS in the Wild: An Internet-wide Analysis of TLS-based Protocols for Electronic Communication“. In: *Proceedings 2016 Network and Distributed System Security Symposium*. NDSS 2016. Internet Society, 2016. DOI: 10.14722/ndss.2016.23055.

[41] Nicholas J. Hopper und Manuel Blum. „Secure Human Identification Protocols“. In: *Advances in Cryptology — ASIACRYPT 2001*. Hrsg. von Colin Boyd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, S. 52–66. ISBN: 978-3-540-45682-7. DOI: 10.1007/3-540-45682-1_4.

[42] Min-Shiang Hwang, Cheng-Chi Lee und S.-F Tzeng. „A new knapsack public-key cryptosystem based on Permutation combination algorithm“. In: *World Academy of Science, Engineering and Technology* 33 (Jan. 2009), S. 888–893.

[43] Seong Hwang, Intae Kim und Wai Kong Lee. *Modern Cryptography with Proof Techniques and Applications*. Feb. 2021. ISBN: 9781003152569. DOI: 10.1201/9781003152569.

[44] Dyala Ibrahim, Je Sen Teh und Rosni Abdullah. „An overview of visual cryptography techniques“. In: *Multimedia Tools and Applications* 80 (Sep. 2021). DOI: 10.1007/s11042-021-11229-9.

[45] Paul Safonov James E. Weber Dennis Guster und Mark B. Schmidt. „Weak Password Security: An Empirical Study“. In: *Information Security Journal: A Global Perspective* 17.1 (2008), S. 45–54. DOI: 10.1080/10658980701824432.

[46] Elham Jokar und Mohammad Mikaili. „Assessment of human random number generation for biometric verification“. In: *J. Med. Signals Sens.* 2.2 (Apr. 2012), S. 82–87. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3632045/.

[47] Ari Juels und Ronald L. Rivest. „Honeywords: making password-cracking detectable“. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS '13. Berlin, Germany: Association for Computing Machinery, 2013, S. 145–160. ISBN: 9781450324779. DOI: 10.1145/2508859.2516671.

[48] Ari Juels und Stephen A. Weis. „Authenticating Pervasive Devices with Human Protocols“. In: *Advances in Cryptology – CRYPTO 2005*. Hrsg. von Victor Shoup. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, S. 293–308. ISBN: 978-3-540-31870-5.

[49] Ari Juels u. a. „How to Turn Loaded Dice into Fair Coins“. In: *IEEE Transactions on Information Theory* 46 (März 2000). DOI: 10.1109/18.841170.

[50] Bruce Kallick. *Handycipher: a Low-tech, Randomized, Symmetric-key Cryptosystem*. Cryptology ePrint Archive, Paper 2014/257. 2014. URL: https://eprint.iacr.org/2014/257.

[51] Alan Kaminsky. *ElsieFour: A Low-Tech Authenticated Encryption Algorithm For Human-to-Human Communication*. Cryptology ePrint Archive, Paper 2017/339. 2017. URL: https://eprint.iacr.org/2017/339.

[52] Jonathan Katz und Yehuda Lindell. *Introduction to Modern Cryptography*. Hrsg. von Douglas R. Stinson. Chapman und Hall/CRC Press, 2015. ISBN: 978-1-4665-7027-6. URL: https://eclass.uniwa.gr/modules/document/file.php/CSCYB105/Reading%20Material/%5BJonathan_Katz%2C_Yehuda_Lindell%5D_Introduction_to_Mo%282nd%29.pdf.

[53] Kunikatsu Kobayashi u. a. „A knapsack cryptosystem based on multiple knapsacks“. In: *2010 International Symposium On Information Theory & Its Applications*. 2010, S. 428–432. DOI: 10.1109/ISITA.2010.5649307.

[54] Alan G. Konheim. „Cryptanalysis of Adfgvx Encipherment Systems“. In: *Advances in Cryptology*. Hrsg. von George Robert Blakley und David Chaum. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, S. 339–341. ISBN: 978-3-540-39568-3. DOI: 10.1007/3-540-39568-7_26.

[55] Vijay Kothari u. a. „Human-Computability Boundaries“. In: *Security Protocols XXVII*. Hrsg. von Jonathan Anderson u. a. Cham: Springer International Publishing, 2020, S. 157–166. ISBN: 978-3-030-57043-9.

[56] Éric Levieil und Pierre-Alain Fouque. „An Improved LPN Algorithm“. In: *Security and Cryptography for Networks*. Hrsg. von Roberto De Prisco und Moti Yung. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 348–359. ISBN: 978-3-540-38081-8. DOI: 10.1007/11832072_24.

[57] Shujun Li und Heung-Yeung Shum. *Secure Human-Computer Identification (Interface) Systems against Peeping Attacks: SecHCI*. Cryptology ePrint Archive, Paper 2005/268. 2005. URL: https://eprint.iacr.org/2005/268.

[58] Shujun Li und Heung-yeung Shum. *Secure Human-Computer Identification against Peeping Attacks (SecHCI): A Survey*. 2003. URL: https://www.hooklee.com/papers/sechci-survey.pdf.

[59] Jiayang Liu, Jingguo Bi und Songyan Xu. „An Improved Attack on the Basic Merkle–Hellman Knapsack Cryptosystem“. In: *IEEE Access* 7 (2019), S. 59388–59393. DOI: 10.1109/ACCESS.2019.2913678.

[60] Thomas Lugrin. „One-Time Pad“. In: *Trends in Data Protection and Encryption Technologies*. Hrsg. von Valentin Mulder u. a. Cham: Springer Nature Switzerland, 2023, S. 3–6. ISBN: 978-3-031-33386-6. DOI: 10.1007/978-3-031-33386-6_1.

[61] Mukundan Madhavan u. a. *NLHB : A Non-Linear Hopper Blum Protocol*. 2010. arXiv: 1001.2140 [cs.CR].

[62] Sławomir Matelski. „Human-Computable OTP Generator as an Alternative of the Two-Factor Authentication“. In: *Proceedings of the 2022 European Interdisciplinary Cybersecurity Conference*. EICC '22. Barcelona, Spain: Association for Computing Machinery, 2022, S. 64–71. ISBN: 9781450396035. DOI: 10.1145/3528580.3532842.

[63] Sławomir Matelski. „Secure Human Identification Protocol with Human-Computable Passwords“. In: *Information Security Practice and Experience*. Hrsg. von Chunhua Su, Dimitris Gritzalis und Vincenzo Piuri. Cham: Springer International Publishing, 2022, S. 452–467. ISBN: 978-3-031-21280-2.

[64] Tsutomu Matsumoto. „Cryptographic Human Identification — Extended Abstract —“. In: *Symbiosis of Human and Artifact*. Hrsg. von Yuichiro Anzai, Katsuhiko Ogawa und Hirohiko Mori. Bd. 20. Advances in Human Factors/Ergonomics. Elsevier, 1995, S. 147–152. DOI: 10.1016/S0921-2647(06)80209-1.

[65] Tsutomu Matsumoto. „Human-computer cryptography: an attempt“. In: *Proceedings of the 3rd ACM Conference on Computer and Communications Security*. CCS '96. New Delhi, India: Association for Computing Machinery, 1996, S. 68–75. ISBN: 0897918290. DOI: 10.1145/238168.238190.

[66] Tsutomu Matsumoto und Hideki Imai. „Human Identification Through Insecure Channel“. In: *Advances in Cryptology — EUROCRYPT '91*. Hrsg. von Donald W. Davies. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, S. 409–421. ISBN: 978-3-540-46416-7. DOI: 10.1007/3-540-46416-6_35.

[67] McKague, Matthew. „Design and Analysis of RC4-like Stream Ciphers“. Magisterarb. 2005. URL: http://hdl.handle.net/10012/1141.

[68] R. Merkle und M. Hellman. „Hiding information and signatures in trapdoor knapsacks“. In: *IEEE Transactions on Information Theory* 24.5 (1978), S. 525–530. DOI: 10.1109/TIT.1978.1055927.

[69] Douglas W. Mitchell. „“RUBIK'S CUBE” AS A TRANSPOSITION DEVICE“. In: *Cryptologia* 16.3 (1992), S. 250–256. DOI: 10.1080/0161-119291866928.

[70] Matthieu Monteiro u. a. *Foxtail+: A Learning with Errors-based Authentication Protocol for Resource-Constrained Devices*. Cryptology ePrint Archive, Paper 2020/261. 2020. URL: https://eprint.iacr.org/2020/261.

[71] Yasuyuki MURAKAMI und Takeshi NASAKO. *Knapsack Public-Key Cryptosystem Using Chinese Remainder Theorem*. Cryptology ePrint Archive, Paper 2007/107. 2007. URL: https://eprint.iacr.org/2007/107.

[72] Moni Naor und Benny Pinkas. „Visual authentication and identification“. In: *Advances in Cryptology — CRYPTO '97*. Hrsg. von Burton S. Kaliski. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, S. 322–336. ISBN: 978-3-540-69528-8. DOI: 10.1007/BFb0052245.

[73] Moni Naor und Adi Shamir. „Visual cryptography“. In: *Advances in Cryptology — EUROCRYPT'94*. Hrsg. von Alfredo De Santis. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, S. 1–12. ISBN: 978-3-540-44717-7. DOI: 10.1007/BFb0053419.

[74] John von Neumann. „John von Neumann Collected Works“. In: Hrsg. von A. H. Taub. Bd. 5: Design of Computers, Theory of Automata and Numerical Analysis. Notes by George E. Forsythe and reproduced from J. Res. Nat. Bus. Stand. Appl. Math. Series, vol. 3, pp. 36–38 (1955). Oxford, England: Pergamon Press, 1961. Kap. Various Techniques Used in Connection with Random Digits, S. 768–770. URL: https://mcnp.lanl.gov/pdf_files/InBook_Computing_1961_Neumann_JohnVonNeumannCollectedWorks_VariousTechniquesUsedinConnectionwithRandomDigits.pdf.

[75] Elena Pagnin u. a. „HB+DB, mitigating man-in-the-middle attacks against HB+ with distance bounding“. In: *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. WiSec '15. New York, New York: Association for Computing Machinery, 2015. ISBN: 9781450336239. DOI: 10.1145/2766498.2766516.

[76] Jacques Patarin und Louis Goubin. „Asymmetric cryptography with S-Boxes Is it easier than expected to design efficient asymmetric cryptosystems?“ In: *Information and Communications Security*. Hrsg. von Yongfei Han, Tatsuaki Okamoto und Sihan Qing. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, S. 369–380. ISBN: 978-3-540-69628-5. DOI: 10.1007/BFb0028492.

[77] Thomas Plantard und Willy Susilo. „Broadcast Attacks against Lattice-Based Cryptosystems“. In: *Applied Cryptography and Network Security*. Hrsg. von Michel Abdalla u. a. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 456–472. ISBN: 978-3-642-01957-9. DOI: 10.1007/978-3-642-01957-9_28.

[78] P. Punithavathi und S. Geetha. „Visual cryptography: A brief survey“. In: *Information Security Journal: A Global Perspective* 26.6 (2017), S. 305–317. DOI: 10.1080/19393555.2017.1386249.

[79] Siladitya Ray. *United States, Six Other Nations Ask Tech Companies To Build Backdoors To Encrypted Communications*. 2020. URL: https://www.forbes.com/sites/siladityaray/2020/10/12/united-states-six-other-nations-ask-tech-companies-to-build-backdoors-to-encrypted-communications/ (besucht am 30. 01. 2024).

[80] Stewart Rosenberg u. a. „Random number generation by normal, alcoholic and schizophrenic subjects“. In: *Psychological Medicine* 20.4 (1990), S. 953–960. DOI: 10.1017/S0033291700036643.

[81] Volker Roth, Kai Richter und Rene Freidinger. „A PIN-Entry Method Resilient against Shoulder Surfing“. In: *Proceedings of the 11th ACM Conference on Computer and Communications Security*. CCS '04. Washington DC, USA: Association for Computing Machinery, 2004, S. 236–245. ISBN: 1581139616. DOI: 10.1145/1030083.1030116.

[82] Moshe Rubin. „John F. Byrne's Chaocipher Revealed: An Historical and Technical Appraisal“. In: *Cryptologia* 35.4 (2011), S. 328–379. DOI: 10.1080/01611194.2011.606751.

[83] Bruce Schneier. *Did NSA Put a Secret Backdoor in New Encryption Standard?* 2007. URL: https://www.schneier.com/essays/archives/2007/11/did_nsa_put_a_secret.html (besucht am 30. 01. 2024).

[84] Bruce Schneier. *The Solitaire Encryption Algorithm*. 1999. URL: https://www.schneier.com/academic/solitaire/ (besucht am 16. 11. 2023).

[85] Marc-Andre Schulz u. a. *A Cognitive Fingerprint in Human Random Number Generation*. Jan. 2020. DOI: 10.31234/osf.io/gub8e. URL: osf.io/preprints/psyarxiv/gub8e.

[86] Marc-André Schulz u. a. „A cognitive fingerprint in human random number generation". In: *Scientific Reports* 11.1 (Okt. 2021), S. 20217. ISSN: 2045-2322. DOI: 10.1038/s41598-021-98315-y.

[87] A. Shamir. „A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem". In: *23rd Annual Symposium on Foundations of Computer Science*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 1982. DOI: 10.1109/SFCS.1982.5.

[88] Daniel Shiu. *Analysis of Solitaire*. 2019. arXiv: 1909.06300 [cs.CR].

[89] Sidney L. Smith. „Authenticating users by word association". In: *Computers & Security* 6.6 (1987), S. 464–470. ISSN: 0167-4048. DOI: 10.1016/0167-4048(87)90027-7.

[90] Blase Ur u. a. „"I Added '!' at the End to Make It Secure": Observing Password Creation in the Lab". In: *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*. Ottawa: USENIX Association, Juli 2015, S. 123–140. ISBN: 978-1-931971-249. URL: https://www.usenix.org/conference/soups2015/proceedings/presentation/ur.

[91] Daniel R. Van der Vieren. *The Rubik's Crypto-Cube: a Trans-Composite Cipher*. 2010. URL: https://epublications.regis.edu/cgi/viewcontent.cgi?article=1511&context=theses.

[92] Donald S. Vogel. „INSIDE A KGB CIPHER". In: *Cryptologia* 14.1 (1990), S. 37–52. DOI: 10.1080/0161-119091864760.

[93] Mark N. Wegman und J.Lawrence Carter. „New hash functions and their use in authentication and set equality". In: *Journal of Computer and System Sciences* 22.3 (1981), S. 265–279. ISSN: 0022-0000. DOI: 10.1016/0022-0000(81)90033-7.

[94] D. Weinshall. „Cognitive authentication schemes safe against spyware". In: *2006 IEEE Symposium on Security and Privacy (S&P'06)*. 2006. DOI: 10.1109/SP.2006.10.

[95] Susan Wiedenbeck u. a. „Design and Evaluation of a Shoulder-Surfing Resistant Graphical Password Scheme". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '06. Venezia, Italy: Association for Computing Machinery, 2006, S. 177–184. ISBN: 1595933530. DOI: 10.1145/1133265.1133303.

[96] Qiang Yan u. a. „On limitations of designing usable leakage-resilient password systems: Attacks, principles and usability". In: *19th Network and Distributed System Security Symposium (NDSS)*. 2012. URL: https://ink.library.smu.edu.sg/sis_research/1435.

[97] Huanyu Zhao und Xiaolin Li. „S3PAS: A Scalable Shoulder-Surfing Resistant Textual-Graphical Password Authentication Scheme". In: *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*. Bd. 2. 2007, S. 467–472. DOI: 10.1109/AINAW.2007.317.

A. Anhang

A.1. LC4

i	$\omega(i, 36)$	$\omega(i, 36)/36^{36}$
0	821576171707129901034591729549220332419839494437260492 8	0.07722498048816391
1	814411575887637324904749646922685606989720020113476812 8	0.0765515362094475
2	814605355264155244637105395320579481410777541151345868 8	0.07656975072087803
3	814600114152484784447149285009642532272026955963537817 6	0.07656925807666508
4	814600255907773895748055985884268767569326354906847641 6	0.07656927140111364
5	814600252073747213652577324825947294466705716205636812 8	0.07656927104072997
6	814600252177445358227682896032182327760080703767917363 2	0.07656927105047719
7	814600252174640656389778002950327792650215388392875622 4	0.07656927105021355
8	814600252174716514917008471917000640929920917516897484 8	0.07656927105022068
9	814600252174714462064906920791498035027750795269583667 2	0.07656927105022049
10	814600252174714518388874481428086154608645059404365824 0	0.0765692710502205
11	814600252174714517199675605214934267303779221911057203 2	0.0765692710502205
12	814600252174714517441371348736747439192012276549392793 6	0.0765692710502205
13	814600252174714515825999400978351818405482880144139878 4	0.0765692710502205
14	814600252174714516384606812128007163790105576181491302 4	0.0765692710502205
15	814600252174714516455259632737830553646584851768908185 6	0.0765692710502205
16	814600252174714516348105964812513940646757169644332646 4	0.0765692710502205
17	814600252174714516208060360769852922279016786235595161 6	0.0765692710502205
18	814600252174714516261619119028821175743590762312787558 4	0.0765692710502205
19	814600252174714516688969851320762404298822425471300403 2	0.0765692710502205
20	814600252174714516960220868586896744682951115342977433 6	0.0765692710502205
21	814600252174714516400093697810798714988029355770249216 0	0.0765692710502205
22	814600252174714516496461032685106778194014517801779200 0	0.0765692710502205
23	814600252174714516153468624027826962980736740662601318 4	0.0765692710502205
24	814600252174714518109106494167257921651067731024071884 8	0.0765692710502205
25	814600252174714461909430057474625766786170636680036352 0	0.07656927105022049
26	814600252174716514308634952053891726614132715630991769 6	0.07656927105022068
27	814600252174640656120068047572668225696134391485222092 8	0.07656927105021355
28	814600252177445358063849712456220882502697723334806732 8	0.07656927105047719
29	814600252073747212786556895325506613246139625601027276 8	0.07656927104072997
30	814600255907773895978698450716246082270432687280671948 8	0.07656927140111364
31	814600114152484783562989684440432922209199720639758336 0	0.07656925807666508
32	814605355264155244259840442969912753694481377641732505 6	0.07656975072087803
33	814411575887637323758934196706016520326461275664023552 0	0.0765515362094475
34	821576171707129899990241125849360097086020486484407091 2	0.07722498048816391
35	276412229933305661234483437402242810340362266195735347 2	0.025981679847086245

Tabelle A.1.: Anzahl und Verhältnis der einfach ausschließbaren Klartexte eines 36-Zeichen langen LC4-Chiffrats in Abhängigkeit der Position einer Null.

$n+1$	$\approx E(\omega((c_0, ..., c_n)))$	$\approx E(\omega((c_0, ..., c_n)))/36^{n+1}$
2	2.890046296296299	0.002229973994055786
3	203.36525491540937	0.004358823193488712
4	10830.880969072694	0.006448426883926263
5	513862.7941336328	0.008498351113416413
6	22877436.989354443	0.010509749464153334
7	978274851.2523918	0.012483701734557602
8	40683953992.18559	0.01442125806040406
9	1657807723765.3118	0.016323438667642434
10	66510036475851.75	0.018191234752590792
11	2635804817592690.0	0.020025609307955675
12	1.0342700883517094e+17	0.02182749792607571
13	4.025355139283268e+18	0.023597809579262885
14	1.555956773909688e+20	0.02533742737791403
15	5.979432579011218e+21	0.02704720930702546
16	2.2863632835748584e+23	0.02872798894172611
17	8.704393554329964e+24	0.030380576142426976
18	3.3012098122652165e+26	0.03200575773016767
19	1.2477924219570695e+28	0.0336042981427244
20	4.7022755430605754e+29	0.03517694007202725
21	1.7672878238216342e+31	0.03672440508341973
22	6.626082959885035e+32	0.03824739421727766
23	2.478890693444971e+34	0.039746588573489915
24	9.255415584698612e+35	0.041222649879289724
25	3.449439046031584e+37	0.042676221040910574
26	1.2834580076226257e+39	0.0441079266795276
27	4.768197736773907e+40	0.04551837365193239
28	1.7689613378361795e+42	0.046908151556376
29	6.554209096098164e+43	0.048277833224002906
30	2.425501678576921e+45	0.04962797519628611
31	8.966014317152513e+46	0.050959118188862784
32	3.310910007306287e+48	0.05227178754215743
33	1.2214501430047617e+50	0.05356649365916918
34	4.5020677830483136e+51	0.05484373243078893
35	1.657987460636496e+53	0.0561039856490017
36	6.10107262092416e+54	0.05734772140831924

Tabelle A.2.: Untere Schranke für die erwartbare Anzahl und Verhältnis der einfach ausschließbaren Klartexte eines LC4-Chiffrats der Länge $n+1$.