Analyse und Implementierung der DDR-Chiffriermaschinen T-310/50 und T-316

Michael Altenhuber

BACHELORARBEIT
Nr. 1510239014-A
ingereicht am
Fachhochschul-Bachelorstudiengang
Sichere Informationssysteme
in Hagenberg

im September 2018
(Um Vorwort ergänzte und korrigierte Fassung vom 24. Januar 2021)
Diese Arbeit entstand im Rahmen des Gegenstands

Kryptographische Grundlagen

im

Sommersemester 2018

Betreuung:

Prof. Bernhard Esslinger
FH-Prof. Dr. Jürgen Fuß
Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 3. September 2018

Michael Altenhuber
# Inhaltsverzeichnis

<table>
<thead>
<tr>
<th>Erklärung</th>
<th>iii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vorwort</td>
<td>vi</td>
</tr>
<tr>
<td>Preface</td>
<td>vi</td>
</tr>
<tr>
<td>Kurzfassung</td>
<td>vii</td>
</tr>
<tr>
<td>Abstract</td>
<td>viii</td>
</tr>
<tr>
<td>1 Einleitung</td>
<td></td>
</tr>
<tr>
<td>1.1 Historische Einordnung der Maschinen</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Über CrypTool 2</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Stand des Wissens</td>
<td>2</td>
</tr>
<tr>
<td>2 T-310/50</td>
<td></td>
</tr>
<tr>
<td>2.1 Allgemeiner Aufbau</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Chifferator</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Algorithmus der T-310</td>
<td>7</td>
</tr>
<tr>
<td>2.3.1 Schlüssel</td>
<td>7</td>
</tr>
<tr>
<td>2.3.2 Ver- und Entschlüsselung</td>
<td>9</td>
</tr>
<tr>
<td>2.3.3 Ableitung pseudozufälliger Bits aus dem Schlüssel</td>
<td>11</td>
</tr>
<tr>
<td>2.4 Analysen und Angriffe</td>
<td>16</td>
</tr>
<tr>
<td>2.5 Implementierung</td>
<td>17</td>
</tr>
<tr>
<td>3 T-316</td>
<td></td>
</tr>
<tr>
<td>3.1 Allgemeiner Aufbau</td>
<td>20</td>
</tr>
<tr>
<td>3.2 Software der T-316</td>
<td>21</td>
</tr>
<tr>
<td>3.3 Algorithmus LAMBDA1</td>
<td>23</td>
</tr>
<tr>
<td>3.3.1 Data Encryption Standard</td>
<td>23</td>
</tr>
<tr>
<td>3.3.2 Änderungen des LAMBDA1-Algorithmus</td>
<td>25</td>
</tr>
<tr>
<td>3.3.3 Sicherheit des Verfahrens</td>
<td>29</td>
</tr>
<tr>
<td>3.4 Implementierung</td>
<td>29</td>
</tr>
<tr>
<td>4 Fazit und Ausblick</td>
<td>32</td>
</tr>
<tr>
<td>A Begriffs- und Abkürzungsverzeichnis</td>
<td>33</td>
</tr>
</tbody>
</table>
Vorwort


Preface

After submission of this bachelor’s thesis, new insights on the topic were gained, which I would like to briefly summarize. Mr. Drobick found out that the T-316 does not use the specified version of LAMBDA1, but a slightly adapted program. Therefore, the plug-in for CrypTool2 has been renamed to LAMBDA1, as it implements the algorithm as described in the specification. I also released the same implementation of LAMBDA1 as stand-alone tool\[1\] with a command-line interface on GitHub under the GPL 3.0 license. This version includes a short summary of the algorithm as well as all specified tests from the documentation, which I digitalized. The implementation runs all of them correctly. Last, some slight spelling mistakes were fixed in this version of the thesis.

Michael Altenhuber, 24. Januar 2021

[1] https://github.com/tassadarius/LAMBDA1
Kurzfassung


Abstract

This thesis covers the analysis and implementation of two historic cipher machines developed in Eastern Germany (GDR) in the 70s and 80s, T-310/50 and T-316. The cipher machines, their historic context, and the used algorithms are described in more detail. The implementation in modern software is done for the free open-source software CrypTool 2.

The T-310/50 is a highly complex stream cipher, which derives the key stream through an algorithm resembling a block cipher. The entire process is done in hard-wired logic. This complicates the implementation in software. An already existing implementation of the T-310 in CrypTool 2 is taken and improved. In addition, some errors can be clarified.

The newer T-316 already had a microprocessor and used the algorithm LAMBDA1, which was a development based on the Data Encryption Standard (DES). LAMBDA1 tried to improve the security of DES. The algorithm is analysed and is described in more detail, including a cautious assessment on the security of the algorithm. The implementation for CrypTool 2 turns out to be more simple than the one of the T-310, due to the close relationship to DES and the properties of the T-316.

As a result, there are implementations of the cipher machines, which are not completely compatible with the original machines. The new implementations will be added to the CrypTool 2 repository. Full compatibility to the original machines is sought and could be achieved through future improvements.
Kapitel 1

Einleitung


Der Quellcode der Implementierungen befindet sich auf den beiliegenden CDs. Aufgrund vieler „altmodischer“ Begriffe in den Original-Unterlagen gibt es ein Begriffsverzeichnis in Anhang A.

Alle Abbildungen dieser Arbeit wurden selbst erstellt, sofern nicht anders durch eine Referenz angegeben. Die Abbildungen selbst zu erstellen, hatte vor allem den Grund, dass zahlreiche Abbildungen der Originaldokumente eine schlechte Qualität haben. Dabei wurden Beschriftungen aktualisiert und es wurde versucht, die Darstellung und Verständlichkeit zu verbessern.

1.1 Historische Einordnung der Maschinen

In der DDR wurde im Jahre 1951 eine eigene Abteilung namens „Zentrales Chiffriermaschinenorgan (ZCO) der DDR“ gegründet. Diese war als Abteilung XI Teil des Ministeriums für Staatssicherheit (MfS). Einige Quellen dieser Arbeit tragen deswegen die Zuordnung „MfS - Abt. XI“ als Dokumentenbezeichnung. Dies erfolgte durch die Behörde der BStU (Bundesbeauftragter für die Stasi-Unterlagen), die für die Verwaltung der Unterlagen der Staatssicherheit der DDR zuständig ist [28]. Die Aufgaben des ZCO waren vielfältig und umfassten unter anderem die Erforschung und Entwicklung von eigenen Verschlüs-
1. Einleitung

Selungsalgorithmen und dazugehörigen Maschinen, die Unterstützung und Überwachung des operativen Betriebes des vorhandenen Chiffriерwesens und die Kryptoanalyse von ausländischer beziehungsweise feindlicher Kommunikation [18].


1.2 Über CrypTool 2


Bisher war in CT2 eine Variante der T-310 als Komponente (Plug-in) implementiert. Durch diese Arbeit wird diese Komponente erweitert und korrigiert. Außerdem kommt die Komponente der T-316, deren Onlinehilfe und passende Templates hinzu. Templates sind Vorlagen in CT2, die die Komponenten in fertigen Szenarien aufrufbar machen.

1.3 Stand des Wissens


1 https://www.cryptool.org/de/cryptool2
1. Einleitung

Kapitel 2

T-310/50


Der verwendete Verschlüsselungsalgorithmus hat keinen eigenen Namen. Er wird lediglich einem älteren Algorithmus der DDR namens „Chiffrierklass Alpha“ zugeordnet, über den keine Aufzeichnungen mehr bekannt sind \[\text{[3]}\]. Um in dieser Arbeit die Maschine und den Algorithmus differenzieren zu können, wird der Name der taktischen Bezeichnung der T-310/50 übernommen und ARGON-Algorithmus genannt (siehe Abschnitt 2.3).

2.1 Allgemeiner Aufbau

Das Gerätesystem T-310 besteht aus dem Grundgerät (GG) und einem Stromversorgungsgerät (SV). Die Bedienung erfolgt über bis zu zwei separate Bediengeräte (BT und BTZ). Ein Foto der Maschine mit einem Bediengerät ist in Abbildung 2.1 zu sehen. Abbildung 2.2 zeigt eine grob vereinfachte Skizze des Aufbaus.

Abbildung 2.1: Das T-310/50-Gerätesystem – die Beschriftungen zeigen auf die jeweiligen Komponenten. Das ursprüngliche Foto stammt aus [3].

Es können bis zu zwei Bediengeräte angeschlossen werden, die bis zu 30 beziehungsweise bis zu 100 Meter entfernt sein können. Über diese werden die Betriebsmodi der T-310 gesteuert und der Status über LED angezeigt. Die Stromversorgung der Bediengeräte erfolgt nicht über das Grundgerät sondern über separate Anschlüsse.

Der Unterschied zwischen den Varianten T-310/50 und T-310/51 liegt bei den Eingabedaten. Die Variante 50 kann lediglich 5-Bit-Fernschreibzeichen chiffrieren und übertragen. Die Maschine vom Typ 51 kann 8-Bit-Eingabewerte chiffrieren, indem die Daten auf 5-Bit-Blöcke aufgeteilt und danach wieder zusammengefügt werden [25]. Das ist bei der Implementierung relevant, weil es leicht zu Unklarheiten kommen kann (siehe Abschnitt 2.5).


2. Chifftrator


- Eingabeeinheit EE – Schnittstelle zur Schlüsseeingabe
- Komplizierungseinheit KE – leitet aus den Schlüsseln pseudozufällige Bits ab
- Verschlüsselungseinheit VE – übernimmt die zu verarbeitenden Zeichen von der Zentraleinheit und ver- oder entschlüsselt diese
- Synchronisationseinheit SE – erzeugt oder verarbeitet den Initialisierungsvektor
- Prüf- und Blockiereinheit PBE – prüft die Parität verschiedener Register und der Schlüssel


2.3 Algorithmus der T-310


2.3.1 Schlüssel

Der Algorithmus verwendet drei verschiedene Schlüssel. Diese heißen in der ursprünglichen Dokumentation Langzeitschlüssel LZS, Zeitschlüssel S0 und Spruchschlüssel F0.

Der Langzeitschlüssel LZS befindet sich auf drei fest installierten, aber austauschbaren Steckkarten [21]. Er gliedert sich in die drei Teile P, D und φ. Diese werden bei der Ableitung der pseudozufälligen Bits verwendet. Bei allen drei Teilen handelt es sich um einfache Argumente für Funktionen. Eine Beschreibung der Funktionsweise und nähere Details zum LZS sind in Abschnitt 2.3.3 zu finden.

Ein Tausch des Langzeitschlüssels war nur bei einer Kompromittierung vorgesehen, oder wenn sich ein Schlüssel im Nachhinein als unsicher herausstellen sollte. Deswegen wurde angenommen, „[...] daß die Gesamtheit der LZS, die für alle Geräte T-310/50 während ihrer gesamten Einsatzzeit benötigt werden, weniger oder nicht wesentlich mehr als

\footnote{Das entspricht einem One-Time-Pad mit pseudozufälligen Bits.}
Abbildung 2.3: Skizze des Chiffriermas im Original und vereinfacht; Abbildung (a) zeigt eine Skizze der Originaldokumentation [5] und (b) eine stark abstrahierte Variante mit den angepassten Beschriftungen zu dieser Arbeit.
10 Stück betragen wird.\footnote{Diese Aussage entspricht der Anzahl an Schlüsseln die laut \cite{21} auch real zum Einsatz kamen.}

Der \textbf{Zeitschlüssel $S_0$} besteht aus zwei 120 Bit langen Schlüsseln $S_1$ und $S_2$. Diese sind physisch auf jeweils fünf Register zu je 24 Bit verteilt. Die Parität beider Zeitschlüssel muss ungerade sein. Das wird vom Gerät bei der Eingabe der Schlüssel geprüft. Das dient nicht nur zur Sicherstellung der Integrität der Zeitschlüssel, sondern hat auch Einfluss auf die Qualität der abgeleiteten pseudozufälligen Bits \footnote{Der Fehler kommt unter anderem auch in \cite{10} und \cite{6} vor.}. Im Zusammenhang mit der Parität der Zeitschlüssel kommt in mehreren Dokumenten und Analysen\footnote{Das T-310/50 verfügt über einen Rauschgenerator. Die T-310/51 generierte den Zufall über das Signal beim Herausziehen einer Lochkarte.} der Fehler vor, dass 10 der 240 Bit für eine Paritätsprüfung reserviert sind. Dieser Irrtum stammt aus Entwicklungsdokumenten, in denen solche Paritätsbits geplant waren. Jedoch wurde diese Idee verworfen. Es stehen also alle 240 Bit für den Schlüsselraum zur Verfügung.

Die Eingabe des Zeitschlüssels erfolgt im Gegensatz zum LZS über leichter tauschbare Hollerith-Lochkarten. Ein Schlüsselwechsel musste laut \cite{12} Seite 8 zuerst täglich durchgeführt werden. In einer Kryptoanalyse während der Entwicklung \cite{8} Seite 25 wird davon ausgegangen, dass ein Zeitschlüssel, in Berücksichtigung der Anzahl an Geräten, mindestens eine Woche und wahrscheinlich länger sicher ist. Im Laufe des Betriebes wurde auf einen wöchentlichen Schlüsselwechselrhythmus umgestellt. Weil der Schlüsselraum mit $2^{240}$ Möglichkeiten minus aller geraden Schlüssel selbst für heutige Standards noch ausreichend groß ist, war dieser häufige Schlüsseltausch wohl reine Vorsichtsmaßnahme vor andersartigen Kompromittierungen.

Beim \textbf{Spruchschlüssel $F_0$} handelt es sich im modernen Sinn um eine Nonce als Initialisierungsvektor. Es ist eine zufällig erzeugte, 61 Bit lange Zeichenfolge, die bei der Ver- und Entschlüsselung in den Algorithmus einfließt. $F_0$ wird als erstes Element vor einer verschlüsselten Kommunikation im Klartext übertragen. Das soll verhindern, dass bei einem Klartext, der mehrmals mit demselben Schlüssel verarbeitet wird, das gleiche Chiffraut entsteht (Ciphertext Indistinguishability). In dieser Arbeit wird aus Gründen der besseren Verständlichkeit der Spruchschlüssel als Initialisierungsvektor $F_0$ bezeichnet. Als Anforderung an $F_0$ gilt, dass er nicht komplett aus Nullbits ($F_0 \neq 0, 0, 0, ..., 0$) oder Einsen ($F_0 \neq 1, 1, 1, ..., 1$) bestehen darf. Die Funktionsweise des Initialisierungsvektors ist in \textbf{Abschnitt 2.3.3} und der dazugehörigen \textbf{Abbildung 2.4} ersichtlich.

\textbf{2.3.2 Ver- und Entschlüsselung}

Dieser Abschnitt beschreibt die Verschlüsselungsweise des Verfahrens, also das Verknüpfen der pseudozufälligen Bits mit dem Klartext. Die Beschreibung beruht auf der technischen Dokumentation in \cite{6} Seite 7] und \cite{1}, sowie \cite{10}. Bei der Verknüpfung werden die 5-Bit-Zeichen des Klartexts zusätzlich zur üblichen XOR-Verknüpfung noch in einem linear rückgkekoppelten Schieberegister rotiert.

Der zugrundeliegende Vektor $V$ des Schieberegisters hat 31 Zustände\footnote{Der komplette Vektor $V$ ist $(1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1)$.}, alle natürlichen Zahlen von eins bis 31. Dieser Vektor lässt sich durch das Polynom $x^3 = x^2 + x + 1$
oder durch die Matrix $M$ definieren, denn eine Rotation des Schieberegisters kann auch als Matrixmultiplikation notiert werden. Dazu wird mit einem beliebigen Vektor bestehend aus einer Reihe und fünf Spalten aus $V$ begonnen, zum Beispiel $(1,1,1,1,1)$. Dieser wird mit jeder Spalte der Matrix multipliziert (Vektor-Matrix-Produkt) und jeweils Modulo 2 gerechnet. Das ergibt den nächsten Zustand $(1,1,1,0,0)$, der wiederum mit der Matrix multipliziert werden kann, um $(1,1,0,0,0)$ zu erhalten. Da es sich bei dem Polynom um eine periodische Funktion handelt, beginnt sich $V$ nach 32 Runden zu wiederholen.

Die Verknüpfung sieht folgendermaßen aus:

$$C_i = (K_i \oplus B_i) \cdot M^{r_i} \quad \text{mit } i = 1, 2, 3, \ldots, l$$

Die Anzahl der Matrixmultiplikationen (oder je nach Betrachtungsweise der Runden im Schieberegister) $r_i$ ist abhängig vom deterministisch generierten, pseudorandomen Wert $H_i$. Zur Bestimmung von $r_i$ gilt es, den niedrigsten Exponenten $e$ zu finden, sodass der Wert des Schieberegisters $(1,1,1,1,1)$ ist.

Die Variablen sind folgend definiert:

- $C_i$ = 5-Bit-Zeichen des Chiffrats
- $K_i$ = 5-Bit-Zeichen des Klartexts
- $B_i$ = 5-Bit-Vektor aus $(a_i,7, \ldots, a_i,11)$
- $H_i$ = 5-Bit-Vektor aus $(a_i,1, \ldots, a_i,5)$
- $M = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$
- $e$ = kleinste natürliche Zahl, sodass $H_i \cdot M^e = (1,1,1,1,1)$
- $r_i = \begin{cases} 0, & \text{wenn } H_i \in \{(0,0,0,0,0),(1,1,1,1,1)\} \\ 31 - e, & \text{sonst} \end{cases}$
- $a_i$ = ein 13 Bit langer Vektor aus der T-310-Blockchiffre, siehe Abschnitt 2.3.3
- $i$ = Index des aktuell zu verschlüsselnden Zeichens
- $l$ = Anzahl an Klartextzeichen

Der Verschlüsselungsvorgang ist in Algorithmus 1 als Pseudocode aufgeführt. Zur technischen Realisierung werden zwei linear rückgekoppelte Schieberegister SRV2 und SRV3 mit einer Länge von 5 Bit verwendet. Zu Beginn wird SRV2 mit $H_i$ und SRV3 mit Einsen $(1,1,1,1,1)$ befüllt. Danach beginnt eine Schleife, in der parallel beide Register mit dem Polynom $x^5 = x^3 + x^1 + 1$ geschoben werden, bis das Register SRV2 den Zustand $(1,1,1,1,1)$ oder $(0,0,0,0,0)$ erreicht. Es kann vorkommen, dass $H_i$ und damit SRV2 bereits in diesem Zustand ist. In diesem Fall wird das Schieben der Register übersprungen. Das gilt für alle Schleifen des Ver- und Entschlüsselungsvorgangs.

Danach wird der Inhalt des Registers SRV3 nach SRV2 kopiert und damit der Zustand von SRV2 verworfen. SRV3 wird mit dem Klartext befüllt, der mit $B_i$ XOR-verknüpfen wird. Danach beginnt die zweite Schleife, bei der wieder SRV2 und SRV3
parallel geschoben werden. Erreicht SR V2 den Zustand \((1,1,1,1,1)\) oder \((0,0,0,0,0)\), ist das Verschlüsseln eines Zeichens abgeschlossen und der Inhalt von SRV3 wird zurückgegeben. Die Bits 6, 12 und 13 aus \(a\) bleiben ungenutzt und werden verworfen.

**Algorithmus 1:** Verschlüsseln eines Klartextzeichens

1. \(a_i \leftarrow 13\) Bit pseudorandom Folge aus der T-310-Blockchiffre
2. \(K_i \leftarrow 5\) Bit Klartextzeichen
3. \(C_i \leftarrow 5\) Bit chiffriertes Zeichen als Ergebnis des Algorithmus
4. \(SRV2 = H_i\)
5. \(SRV3 = 11111_b\)

6. while \(SRV2 \neq 11111_b\) and \(SRV2 \neq 00000_b\) do
7. \(SRV2 = (SRV2 \ll 1) \mid (((SRV2 \gg 4) \oplus (SRV2 \gg 2)) \& 1)\)
8. \(SRV3 = (SRV3 \ll 1) \mid (((SRV3 \gg 4) \oplus (SRV2 \gg 2)) \& 1)\)
9. end while

10. \(SRV2 = SRV3\)
11. \(SRV3 = B_i \oplus K\)

12. while \(SRV2 \neq 11111_b\) and \(SRV2 \neq 00000_b\) do
13. \(SRV2 = (SRV2 \ll 1) \mid (((SRV2 \gg 4) \oplus (SRV2 \gg 2)) \& 1)\)
14. \(SRV3 = (SRV3 \ll 1) \mid (((SRV3 \gg 4) \oplus (SRV2 \gg 2)) \& 1)\)
15. end while

16. \(C_i = SRV3\)

Der Entschlüsselungsvorgang ist in [Algorithmus 2](#) als Pseudocode zu finden. Dieser beginnt wieder mit der Befüllung der Register SRV2 und SRV3 mit den Werten \(H_i\) und \((1,1,1,1,1)\). Danach wird die Schleife berechnet. Das Ergebnis aus SRV3 wird anschließend mit \(B_i\) XOR-verknüpft und das ursprüngliche Zeichen des Klartextes ist wiederhergestellt. Die zweite Schleife entfällt, da durch die Periodizität des Polynoms \(x^5 = x^3 + x^1 + 1\) der ursprüngliche Zustand erreicht wird.

### 2.3.3 Ableitung pseudozufälliger Bits aus dem Schlüssel

Die T-310 verwendet zur Ableitung der pseudozufälligen Bits aus den Schlüsseln einen Algorithmus, der einer Blockchiffre ähnelt. Die logische Schaltung dafür befindet sich in der Komplizierungseinheit und wird in verschiedenen Dokumenten wahlweise als Wurmwreihe D-W oder als Funktion in den Variationen \(\phi\), \(\varphi\) und \(\Phi\) bezeichnet. Courtois nennt sie die „T-310-Blockchiffre“, weswegen der Begriff hier übernommen wird.

Als Eingabeparameter dienen die Zeitschlüssel \(S_1\) und \(S_2\) und der Initialisierungsvektor \(F_0\). Der LZS fließt über installierte Steckkarten ein. Als Ausgabe entsteht die Bitfolge \(a\) mit einer Länge von 13 Bit (siehe Abschnitt 2.3.2).

Kernelement der Logik ist das zentrale Register \(U\), das in Runden \(n\) verändert wird. In jeder 127. Runde wird daraus ein Bit als Ergebnis entnommen. Demnach benötigt die
Algorithmus 2: Entschlüsseln eines Geheimtextzeichens

1: $a_i \leftarrow$ 13 Bit pseudozufällige Folge aus der T-310-Blockchiffre
2: $C_i \leftarrow$ 5 Bit chiffriertes Zeichen
3: $K_i \leftarrow$ 5 Bit Klartextzeichen als Ergebnis des Algorithmus

4: $SRV2 = H_i$
5: $SRV3 = C_i$

6: while $SRV2 \neq 11111_b$ and $SRV2 \neq 00000_b$ do
7:   $SRV2 = (SRV2 \ll 1) | (((SRV2 \gg 4) \oplus (SRV2 \gg 2)) \& 1)$
8:   $SRV3 = (SRV3 \ll 1) | (((SRV3 \gg 4) \oplus (SRV2 \gg 2)) \& 1)$
9: end while

10: $C = SRV3 \oplus B_i$

Blockchiffre 1651 Runden, um einen 13-Bit-Rückgabewert zu liefern. Aus der Länge des Registers U von 36 Bit ergibt sich die Blocklänge. U wird zu Beginn mit der statischen Bitfolge $U_0$ initialisiert. Der Initialisierungsvektor $F_0$ wird nach Beginn in jeder Runde der Blockchiffre in einem Schieberegister F verändert. Um zwischen dem ursprünglichen Wert und dem jeweiligen Zustand der Runde unterscheiden zu können, wird die Variable $F_n$ eingeführt. Wichtige Bitfolgen sind:

- $U_0 = 0110 1001 1100 0111 1100 1000 1010 0011$ bzw. 0x06:9C:7C:85:A3
- $S_1$ – Zeitschlüssel als 120 Bit langer Vektor
- $S_2$ – Zeitschlüssel als 120 Bit langer Vektor
- $F_0$ – 61 Bit langer Initialisierungsvektor
- $F_n$ – Der Wert F der jeweiligen Runde, am Beginn ausgehend von $F_0$
- $P$ – Umfasst 27 Werte; Teil des Langzeitschlüssels
- $D$ – Umfasst 9 Werte; Teil des Langzeitschlüssels

Die Blockchiffre verwendet folgende Funktionen:

- P-Funktion – Bildet mithilfe von P auf dem Langzeitschlüssel 36 Bit des Registers U auf 27 Bit ab
- D-Funktion – Bildet mithilfe von D auf dem Langzeitschlüssel 36 Bit des Registers U auf 9 Bit ab
- Z-Funktion – Boolesche Funktion mit 6 Eingabewerten
- Rotationsfunktion von F

Anhand der Werte in den Vektoren P und D werden bestimmte Bits des Registers U durch die P- und D-Funktion ausgewählt. So entspricht zum Beispiel $P_1 = 24$ dem

\[ ^6 \text{Bei der Anzahl der Runden ergeben sich aus technischer Sicht Abweichungen, die in Abschnitt 2.5 erläutert werden.} \]

\[ ^7 \text{Die Z-Funktion ist vier Mal (Z}_1 - Z_4) \text{baugleich in der Komplizierungseinheit vorhanden, um alle Werte der P-Funktion gleichzeitig verarbeiten zu können.} \]

\[ P = (24, 3, 33, 30, 2, 8, 5, 12, 9, 1, 10, 6, 32, 22, 21, 18, 28, 25, 16, 20, 36, 13, 17, 29, 26, 4, 35) \]
\[ D = (0, 12, 24, 36, 28, 16, 32, 8, 4) \]
\[ \alpha = 3 \]

\( Z \) ist eine nichtlineare boolesche Funktion, die bereits beim Vorgängeralgorithmus der Chiffriermaschine SKS V/1 zum Einsatz kam. Sie nimmt sechs Bit als Eingabe entgegen und hat ein Bit als Ausgabewert:

\[
Z(e_1, e_2, ..., e_6) = 1 + e_1 + e_5 + e_6 + e_1e_4 + e_2e_3 + e_2e_5
\]
\[
\oplus e_4e_5 + e_5e_6 + e_1e_3e_4 + e_1e_2e_6 + e_1e_4e_5 + e_2e_3e_6 + e_3e_5e_6
\]
\[
\oplus e_1e_2e_3e_4 + e_1e_2e_3e_5 + e_1e_2e_5e_6
\]
\[
\oplus e_2e_3e_4e_6 + e_1e_2e_3e_4e_5 + e_1e_2e_3e_4e_6
\]


Die Rotationsfunktion von F ist in einem einfachen, linear rückgekoppelten Schieberegister realisiert. Nach jeder Runde wird \( F_n \) mit dem primitiven Polynom \( x^{61} = x^5 + x^2 + x^1 + 1 \) rückgekoppelt.


Auffällig bei der Blockchiffre ist die hohe Anzahl an Runden bei vergleichsweise geringem Einfluss der Schlüssel. Mit einer Folge \( a \) kann ein 5-Bit-Klartextzeichen verschlüsselt werden (siehe [Abschnitt 2.3.2]). Das bedeutet, dass pro verschlüsseltem Bit

\(^7\)Die Langzeitschlüssel waren nicht chronologisch nummeriert.
Abbildung 2.4: Graphische Darstellung der Abläufe in der T-310-Blockchiffre zur Bildung von T
Abbildung 2.5: Graphische Darstellung einer neuen Runde in der T-310-Blockchiffre (Schiebevorgang)
2. T-310/50

<table>
<thead>
<tr>
<th>Algorithmus</th>
<th>Jahr</th>
<th>Land</th>
<th>Blockgröße</th>
<th>Schlüsselgröße</th>
<th>Runden pro verschlüsseltem Bit</th>
<th>Genutzter Anteil des Schlüssels¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKS V/1</td>
<td>1973</td>
<td>DDR</td>
<td>27</td>
<td>208</td>
<td>119</td>
<td>1 %</td>
</tr>
<tr>
<td>T-310²</td>
<td>1976</td>
<td>DDR</td>
<td>36</td>
<td>240</td>
<td>330,2</td>
<td>0,8 %</td>
</tr>
<tr>
<td>DES</td>
<td>1974</td>
<td>USA</td>
<td>64</td>
<td>56</td>
<td>0,25</td>
<td>75 %</td>
</tr>
<tr>
<td>GOST (MAGMA)</td>
<td>1989</td>
<td>Russland</td>
<td>64</td>
<td>256</td>
<td>0,5</td>
<td>12,5 %</td>
</tr>
<tr>
<td>TEA</td>
<td>1994</td>
<td>Großbritannien</td>
<td>64</td>
<td>128</td>
<td>1</td>
<td>50 %</td>
</tr>
<tr>
<td>AES</td>
<td>1996</td>
<td>Belgien</td>
<td>128</td>
<td>128</td>
<td>0,08</td>
<td>100 %</td>
</tr>
<tr>
<td>PRESENT</td>
<td>2007</td>
<td>Deutschland, Frankreich</td>
<td>64</td>
<td>128</td>
<td>2,1</td>
<td>100 %</td>
</tr>
<tr>
<td>Simon/ Speck</td>
<td>2013</td>
<td>USA</td>
<td>64</td>
<td>128</td>
<td>0,42</td>
<td>100 %</td>
</tr>
</tbody>
</table>

¹ pro Runde ² Die T-316 wird in [3] behandelt, deren Effizienz wurde aber nicht verglichen.

330,2 Runden benötigt werden. Im Vergleich dazu zeigt Tabelle 2.1, wie andere Algorithmen wesentlich effizienter und schneller sind. Durch den geringen Einfluss der Schlüssel pro Runde steigt jedoch auch die Sicherheit der Blockchiffre. Es müsste für eine Kryptoanalyse eine große Menge an Chiffraten vorliegen, um daraus Rückschlüsse auf den Zustand der Blockchiffre ziehen zu können [3].

2.4 Analysen und Angriffe

Die bisher durchgeführten Kryptoanalysen zur T-310 [14] zeigen einige Schwachstellen oder mögliche Sicherheitsrisiken auf:

- Es gibt eine mögliche Schwachstelle in der Blockchiffre durch die geteilte Nutzung des Schlüssels. \( S_1 \) und \( S_2 \) beeinflussen \( U \) und \( T \) jeweils unabhängig voneinander. Dadurch könnte der effektive Schlüsselraum verkleinert werden.
- Durch die lineare Reduzierung von 36 auf 27 Bit zur Befüllung der Z-Funktionen und des logischen Registers \( T \), neigt die Blockchiffre dazu, Bits systematisch nicht zu verwenden. Unter akademischen Umständen lässt sich mithilfe dieser Eigenschaft unter Einsatz differentieller Kryptoanalyse ein Angriff konzipieren [13], der aber in der realen T-310 abgefangen werden würde [14, Seite 37 f.].
- Bei der Verschlüsselung eines Zeichens kann, durch die Verwendung der linear rückgekoppelten Schieberegister \( SRV2 \) und \( SRV3 \) mit einer variablen Anzahl an Runden (siehe Abschnitt 2.3.2), Information über einen Seitenkanal abfließen. Aufgrund der damaligen Fernschreibtechnik bleibt das jedoch ein theoretischer Angriff, weil die Zeit nicht gemessen werden konnte.
- Es lassen sich Langzeitschlüssel finden, die zu Schwächen in der Blockchiffre führen. Real verwendete Schlüssel wiesen solche Fehler aber nicht auf, weil sie auf

\[
P = (22, 24, 33, 32, 14, 4, 5, 28, 9, 11, 27, 18, 36, 16, 21, 15, 20, 25, 35, 8, 1, 6, 23, 29, 19, 12, 13)
\]
\[
D = (12, 16, 0, 36, 28, 32, 24, 4, 20)
\]
\[
a = \text{n. a.}
\]

Trotz der oben aufgeführten Schwächen, ist ein effizienter und verlässlicher Angriff auf die T-310, der über ein theoretisches Stadium hinausgeht, nicht bekannt.

2.5 Implementierung


Um das Zusammenspiel der Komponenten besser zu verdeutlichen, wird der Algorithmus auf vier Klassen aufgeteilt, die den Funktionseinheiten entsprechen (siehe [Abschnitt 2.2]) und deren jeweilige Aufgabe erfüllen. Die Eingabe- und die Prüfkompilziereinheit werden nicht umgesetzt: Die Eingabe wird von CT2 selbst abgedeckt und die Prüfkompilziereinheit ist überflüssig, weil ein ohnehin schon unwahrscheinlicher Berechnungsfehler kaum Auswirkungen hätte, die Performance aber um fast das doppelte reduziert werden würde.

- ComplexUnit.cs (Komplizierungseinheit)
- SynchronizationUnit.cs (Synchronisationseinheit)
- ControlUnit.cs (Prüf- und Blockiereinheit)
- EncryptionUnit.cs (Verschlüsselungseinheit)


Ein weiterer Punkt sind die verschiedenen Schlüssel. In der T-310-Blockchiffre wird pro Runde jeweils ein Bit der Schlüssel $S_1$, $S_2$ und $F_n$ verwendet. Ob dies das höchstwertige (most-significant) Bit oder das niedrigstwertige (least-significant) Bit ist, unterscheidet sich bei den Implementierungen. So verwendet Drobicks Implementierung das höchstwertige Bit von $F_n$ und CT2 das niedrigstwertige. Für die neue Version wird eine Option zur Verfügung gestellt, bei der man auswählen kann, welche Bit-Ordnung verwendet wird.

Die Komponente der CT2-Variante heißt T-310/50, implementiert aber eigentlich die Variante 518. Das lässt sich daran erkennen, dass Daten Byte-weise auf 5-Bit-Blöcke reduziert und dann verschlüsselt werden. Das wird auch in der Dokumentation zur Komponente erläutert. Um diesen Sachverhalt klarer darzustellen, und um den alten Code nicht verwerfen zu müssen, wird eine Option zum Wählen des Modus eingebaut. So kann zwischen der T-310/50 und der T-310/51 unterschieden werden (siehe dazu auch [Abschnitt 2.1]).


*Diese entspricht auch dem Programm, das Drobick unter [22] zur Verfügung stellt.*
Abbildung 2.6: Template der überarbeiteten Komponente T-310 in CT2
Kapitel 3

T-316


3.1 Allgemeiner Aufbau

Kernelement des T-316-Gerätesystems ist das Grundgerät, das die meisten Funktionen der Chiffriermaschine beinhaltet. Dazu zählen die Schlüsselverwaltung, Ein- und Aus-

Der Speicher besteht aus zwei EPROM-Modulen nichtflüchtigen Speichers und aus zwei SRAM-Modulen flüchtigen Speichers. Hervorzuheben ist, dass der zweite RAM mit der Bezeichnung D10 auch bei ausgeschaltetem Gerät mit Strom versorgt wird, um den Zustand zu stützen. Die adressierbaren Speicherbereiche des SRAMs werden in Tabelle 3.1 dargestellt. Über den Bildwiederholspeicher lässt sich das LCD-Display ansteuern. Jeweils eine Zeile kann über die 7-Byte Adressen 0xA0N0 -- 0xA0NE angesteuert werden, wobei N der Zeilenummer 0–9 entspricht. Ein Zeichen ist 5x7 Pixel groß. Tabelle 3.2 zeigt den adressierbaren Speicherbereich des EPROMs. Die Tastatur verfügt über 37 Tasten für das lateinische Alphabet mit 26 Zeichen, Ziffern und einige Sonderzeichen. Auffällig ist das englische Tastaturlayout QWERTY, obwohl andere Computertastaturen in der DDR durchaus QWERTZ verwendeten. Die Tastatur musste gewisse Anforderungen an die Größe und Qualität erfüllen und das gewählte Modell war nur in englischem Layout verfügbar [4].

3.2 Software der T-316

Der Programmc ode der T-316 ist komplett in U880 Assembly geschrieben, der außer geringen Abweichungen zu Z80 Assembly kompatibel ist. Welche Mnemonic-Notation und welcher Assembler verwendet wurden, ist nicht mehr bekannt [4]. Der Quellcode
Tabellen 3.1: SRAM-Speicherbereich der T-316

<table>
<thead>
<tr>
<th>Adresse</th>
<th>Größe</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x8000 - 0x8FFF</td>
<td>4096 Byte</td>
</tr>
<tr>
<td>0x9000 - 0x93FF</td>
<td>1024 Byte</td>
</tr>
<tr>
<td>0xDC00 - 0xDFFF</td>
<td>1024 Byte</td>
</tr>
<tr>
<td>0x9400 - 0x97FF</td>
<td>1024 Byte</td>
</tr>
<tr>
<td>0xA000 - 0xA3FF</td>
<td>1024 Byte</td>
</tr>
<tr>
<td>0x5C00 - 0x5FFF</td>
<td>1024 Byte</td>
</tr>
</tbody>
</table>

Tabellen 3.2: EPROM-Speicherbereich der T-316

<table>
<thead>
<tr>
<th>Adresse</th>
<th>Größe</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000 - 0x1FFF</td>
<td>8192 Byte</td>
</tr>
<tr>
<td>0x2000 - 0x3FFF</td>
<td>8192 Byte</td>
</tr>
</tbody>
</table>

befindet sich in [7, Seite 35–136 (PDF-Zählung)] und ist als Datei auf der beigelegten CD enthalten. Die Verschlüsselungsroutine selbst befindet sich an der Adresse 0x1b69 und lässt sich leicht aus Drobicks Kommentaren über die Jump-Table in Zeile 413 finden.


Die aufrufbaren Programme werden als Betriebsartensteuerprogramme bezeichnet und umfassen folgende Subroutinen [7, Seite 21]:

Schlüsseleingabe SC
- Prüfung des Geräts mithilfe des Programms Prophylaktische Prüfung
- Eingabe neuer Schlüssel mit dem Programm Textprozessor
- Kontrolle der eingegebenen Schlüssel
- Abspeichern der Schlüssel

Schlüsselwechsel (Keine Abkürzung)
- Prüfung von ZS-2
- Überschreiben von ZS-1 mit ZS-2
- Löschen von ZS-2

Klartexterstellung KT
- Eingabe einer Klartextnachricht mit dem Programm Textprozessor
- Umkodieren des ASCII-Codes auf CCITT-2 ("Zusammenziehen") und Anbringen des Paddings
- Prüfung des Geräts und der vorhandenen Schlüssel
- Chiffrierung des eingegebenen Klartexts
3. T-316

Chiffrierung CK
- Prüfung der vorhandenen Schlüssel
- Erneutes Chiffrieren des Klartexts
- Vergleich des Vorgangs mit dem Ergebnis aus dem Programm Texterstellung
- Löschen des Klartexts im Speicher
- Generierung des Initialisierungsvektors für den nächsten Chiffriervorgang

Senden SE
- Prüfen des Vorhandenseins eines Chiﬀrats
- Je nach ausgewähltem Modus kann das Programm Senden das Chiﬀrat über das LCD-Display ausgeben oder an eine äußere Schnittstelle weitergeben. Falls nötig kann das Programm das Chiﬀrat in ein frequenzmoduliertes Signal wandeln.

Empfang EM
- Chiffrat über Schnittstelle empfangen und in den Speicher ablegen
- Bei Bedarf wird das Signal einer externen Schnittstelle demoduliert und in den Speicher abgelegt.

Ähnlich zur T-310 (siehe Abschnitt 2.2) wird der Klartext zweimal verschlüsselt, und das Ergebnis der Verschlüsselung geprüft. Nur wenn beide Ergebnisse ident sind, ist der Vorgang gültig. Aus moderner Sicht ungewöhnlich ist, dass die Verschlüsselungsroute aus der Subroutine Texteingabe aufgerufen wird.

3.3 Algorithmus LAMBDA1


Vermutlich aus Gründen von Zeitdruck wurde DES als Grundlage eines neuen Algorithmus für die T-316 und T-325 gewählt. DES wurde analysiert und darauf aufbauend wurden Änderungen vorgenommen, um die Sicherheit zu erhöhen. Die folgenden Abschnitte beschreiben den Aufbau des nativen DES in Abschnitt 3.3.1 und die vorgenommenen Modifikationen in Abschnitt 3.3.2. Die Bezeichnung LAMBDA1 in Großbuchstaben folgt der Schreibweise der Originaldokumentation. Dabei handelt es sich wahrscheinlich um keine Abkürzung.


3.3.1 Data Encryption Standard

sind verschiedene Funktionen und Permutationen, die auf den Block und die Schlüssel angewandt werden [11]. Kurz zusammengefasst sieht der Prozess folgendermaßen aus:

1. Der Block durchläuft die Eingangspermutation.
2. Der Block wird in einen linken und rechten Teilblock zu je 32 Bit geteilt.
3. Der rechte Teilblock wird mithilfe einer Expansionsfunktion auf 48 Bit erweitert.
5. Beide 48-Bit-Teile werden XOR-verknüpft und zu jeweils 6 Bit auf 8 verschiedene S-Boxen aufgeteilt.
6. Die S-Boxen reduzieren die 48 Bit wieder auf 32 Bit.
7. Diese durchlaufen eine weitere Permutation.
8. Der linke und der rechte Teilblock werden XOR-verknüpft.
9. Dieser neu entstandene Teil wird zum nächsten rechten Teilblock und der alte rechte Block zum linken.
10. Der Prozess ab Schritt 3 wird 16 Mal wiederholt.
11. Danach werden beide Blöcke vereinigt und durchlaufen eine finale Ausgangspermutation.

Der Vorgang ist in Abbildung 3.4 graphisch dargestellt.

Es folgt eine detaillierte Beschreibung der Abläufe. Zu Beginn durchläuft der Block die Eingangspermutation IP (auch initiale Permutation genannt). Diese kann als einfache Tabelle angegeben werden und beschreibt, welches Bit an welche Stelle platziert wird.

**Tabelle 3.3: Initiale Permutation IP**

| 58 &middot; 50 &middot; 42 &middot; 34 &middot; 26 &middot; 18 &middot; 10 &middot; 2 |
| 60 &middot; 52 &middot; 44 &middot; 36 &middot; 28 &middot; 20 &middot; 12 &middot; 4 |
| 62 &middot; 54 &middot; 46 &middot; 38 &middot; 30 &middot; 22 &middot; 14 &middot; 6 |
| 64 &middot; 56 &middot; 48 &middot; 40 &middot; 32 &middot; 24 &middot; 16 &middot; 8 |
| 57 &middot; 49 &middot; 41 &middot; 33 &middot; 25 &middot; 17 &middot; 9 &middot; 1 |
| 59 &middot; 51 &middot; 43 &middot; 35 &middot; 27 &middot; 19 &middot; 11 &middot; 3 |
| 61 &middot; 53 &middot; 45 &middot; 37 &middot; 29 &middot; 21 &middot; 13 &middot; 5 |
| 63 &middot; 55 &middot; 47 &middot; 39 &middot; 31 &middot; 23 &middot; 15 &middot; 7 |


Der 64-Bit-Block wird in zwei Teilblöcke geteilt, die linker und rechter Block (L und R) genannt werden. Danach beginnen die 16 Runden, in denen jeweils $L_n \rightarrow L_{n+1}$ und $R_n \rightarrow R_{n+1}$ verändert werden. Dies geschieht nach folgendem Schema:
3. T-316

DES – eine Runde der Verschlüsselung:

\[ L_n = R_{n-1} \]
\[ R_n = L_{n-1} \oplus f(R_{n-1}, K_n) \]

Die Rundenfunktion \( f \) nimmt als Eingangswerte einen 32-Bit-Teilblock und den Rundenschlüssel \( K_n \) entgegen und erzeugt daraus einen 32 Bit langen Ausgangswert. \( K_n \) ist ein 48 Bit langer Vektor, der pro Runde \( n \) aus dem Hauptschlüssel mithilfe der Schlüsselauswahlfunktion \( KS \) abgeleitet wird. Da \( KS \) im LAMBDA1-Algorithmus ausgetauscht wurde, ist diese hier nicht näher beschrieben. Die Schlüsselauswahlfunktion von LAMBDA1 wird in Abschnitt 3.3.2 erläutert.


Nachdem alle Runden abgeschlossen sind \( (n = 16) \), werden der L- und der R-Block wieder zusammengeführt. Dieser Block durchläuft eine letzte Permutation. Das ist die Inverse der ursprünglichen Eingangspermutation \( IP^{-1} \) und wird auch als Ausgangspermutation bezeichnet. Das Ergebnis daraus ist der verschlüsselte Block.

Das Entschlüsseln entspricht dem Verschlüsselungsvorgang in umgekehrter Reihenfolge.

3.3.2 Änderungen des LAMBDA1-Algorithmus

Mit dem Ziel, DES sicherer zu machen, wurden folgende Änderungen am Algorithmus vorgenommen [2]:

- Vergrößerung der effektiven Schlüssellänge von 56 auf 256 Bits
- Anpassung der Schlüsselauswahlfunktion \( KS \)
- Spezielle Verknüpfung zweier zusätzlicher Schlüsselvektoren in der achten Runde.
- Weglassen der Ein- und Ausgangspermutationen \( IP \) und \( IP^{-1} \)
- Änderung der Funktion \( f \)

Die Länge des Schlüssels \( S \) wurde auf 256 Bit \( (S_1, ..., S_{256}) \) angehoben. Deswegen musste die Schlüsselauswahlfunktion \( KS \) geändert werden. Der LAMBDA1-Algorithmus benötigt 18 statt 16 Rundenschlüssel, weil in der achten Runde eine zusätzliche Operation durchgeführt wird. In \( S \) sind bereits sechs Schlüsselvektoren enthalten, die unverändert übernommen werden. Bei \( K_1 \) bis \( K_4 \) handelt es sich um normale Schlüssel für die jeweilige Runde mit einer Länge von 48 Bit, die für \( f \) benötigt werden. \( K_{17} \) und \( K_{18} \) sind neu hinzugefügte Schlüssel mit einer Länge von 32 Bit.
Die restlichen Schlüsselvektoren $K_5, ..., K_{12}$ und $K_{13}, ..., K_{16}$ werden über ein einfaches, linear rückgekoppeltes Schieberegister $T$ erzeugt (siehe Abbildung 3.2). Dabei wird ein bereits bestehender 48-Bit-Rundenschlüssel $K_n$ um 11 Bit verschoben.

Die Schlüsselableitung sieht dann wie folgt aus:

\[
\forall n \in \{5, 12\} : \quad K_n := T^{11}(K_{n-4})
\]

\[
\forall n \in \{13, 16\} : \quad K_n := T^{11}(K_{25-n})
\]

Damit sind alle 18 benötigten Rundenschlüssel definiert. Durch die erweiterte Länge von $S$ ist die Schlüsselauswahlfunktion des LAMBDA1-Algorithmus sogar einfacher, als die des unveränderten DES.

In der Funktion $f$ wurde zwar keine Funktion oder Permutation geändert, jedoch wurde die Permutation $P$ verschoben und befindet sich nun direkt nach dem Eingangswert noch vor der Expansionsfunktion. Abbildung 3.3 stellt beide Varianten schematisch gegenüber.
Abbildung 3.3: Vergleich zwischen der Funktion $f$ von LAMBDA1 und DES; der einzige Unterschied ist die Position der Permutation P.

Der gesamte Verschlüsselungsvorgang des LAMBDA1-Algorithmus ist im nachfolgenden Kästchen zusammengefasst. Dort wird auch ersichtlich, wie in der achten Runde die beiden zusätzlichen 32-Bit-Schlüssel einfließen. Abbildung 3.4 zeigt den Vorgang schematisch im direkten Vergleich zu DES.

LAMBDA1-Verschlüsselung:

$$\forall n \in \{1,16\} : \begin{cases} (L_n, R_n) := (R_{n-1}, L_{n-1} \oplus f(R_{n-1}, K_n)) , \text{ falls } n \notin \{8,16\} \\ (L_8, R_8) := (R_7 \oplus K_{17}, (L_7 \oplus f(R_7, K_8)) \oplus K_{18}) \\ (L_{16}, R_{16}) := (L_{15} \oplus f(R_{15}, K_{16}), R_{15}) \end{cases}$$

Die Ein- und Ausgangspermutationen $IP$ und $IP^{-1}$ wurden entfernt. Bei der Verknüpfung $\oplus$ werden beide 32-Bit-Vektoren modular addiert.
Abbildung 3.4: Schematischer Vergleich zwischen den Verschlüsselungsalgorithmen von LAMBDA1 und DES nach [3]

Die Verknüpfung ⊕ ist definiert als:

\[(X_1, \ldots, X_{32}), (Y_1, \ldots, Y_{32}) \mapsto X_1 \cdot 2^{32-1} + X_2 \cdot 2^{32-2} + \ldots + X_{32} \cdot 2^0 + Y_1 \cdot 2^{32-1} + Y_2 \cdot 2^{32-2} + \ldots + Y_{32} \cdot 2^0 \mod 2^{32})\]

Beim Entschlüsseln werden beim LAMBDA1-Algorithmus die Rundenschlüssel von hinten nach vorne angeordnet und die Vektoren \(K'_17\) und \(K'_{18}\) gebildet.
3. Neuanordnung der Schlüssel zum Entschlüsseln:

\[
\forall n \in \mathbb{1}, \mathbb{16} : \quad K'_n := K_{17-n} \\
K'_{17} := (0, 0, ..., 0, 1) \oplus (K_8 \oplus (1, 1, ..., 1)) \\
K'_{18} := (0, 0, ..., 0, 1) \oplus (K_{17} \oplus (1, 1, ..., 1))
\]

Der Vorgang zum Bilden von \(K'_{17}\) und \(K'_{18}\) entspricht einer Subtraktion via Einerkomplement. Durch die Neuanordnung der Schlüssel muss der Entschlüsselungsalgorithmus nicht verändert werden.

LAMBDA1-Entschlüsselung:

\[
\forall n \in \mathbb{1}, \mathbb{16} : \quad \begin{cases}
(L'_n, R'_n) := (R'_{n-1}, L'_{n-1} \oplus f(R'_{n-1}, K'_n)) & , \text{falls } n \notin \{8, 16\} \\
(L'_8, R'_8) := (R'_7 \oplus K'_{17}, (L'_7 \oplus f(R'_7, K'_8)) \oplus K'_{18}) \\
(L'_{16}, R'_{16}) := (L'_{15} \oplus f(R'_{15}, K'_{16}), R'_{15})
\end{cases}
\]

3.3.3 Sicherheit des Verfahrens


Dabei ist zu beachten, dass es für LAMBDA1 genauso wie für DES schwache und halbschwache Schlüssel gibt. Schwach heißt, dass der Schlüssel eine Palindromstruktur aufweist. Das bedeutet, dass der Teilschlüssel der Runde \(n\) gleich dem Teilschlüssel der Runde \(17-n\) ist. Daraus folgt, dass das Verschlüsseln eines bereits verschlüsselten Chiffertextblocks mit einem schwachen Schlüssel wieder zum Klartext führt. Halbschwache Schlüssel bilden ein Paar \(K_1\) und \(K_2\), bei denen diese Eigenschaft auftritt, wenn zuerst mit dem Schlüssel \(K_1\), und dann das Ergebnis mit \(K_2\) verschlüsselt wird [9]. Für LAMBDA1 gibt es \(2^{34}\) mit schwacher und \(2^{68}\) Schlüssel mit halbschwacher Palindromstruktur [19], also deutlich mehr als die zwei schwachen und zwölf halbschwachen Schlüssel des DES.

3.4 Implementierung

Um CT2 eine neue Komponente hinzuzufügen, gibt es eine Anleitung [26]. Dabei kann auf ein einfaches Skelett für Visual Studio zurückgegriffen werden, das die wichtigsten Elemente bereits enthält. Zusätzlich dazu müssen noch zwei Dateien namens Resources.resx und Resources.de.resx erstellt werden, die die mehrsprachigen Zeichenketten für Deutsch und Englisch enthalten. Die Schlüssel- und Klartextdaten gelangen über eine
Schnittstelle aus Auto-Implemented-Properties an das Objekt der Komponente. Diese werden an die Implementierung des Algorithmus weitergegeben.

Für die Implementierung des LAMBDA1-Algorithmus wird auf eine bereits bestehende DES-Implementierung in CT2 zurückgegriffen, die anschließend angepasst wird. Vor allem die Tabellen für die Funktionen und Permutationen lassen sich einfach wiederverwenden. Der Hauptaufwand liegt bei der Erzeugung der Rundenschlüssel, weil diese sich grundlegend von DES unterscheiden.


Abbildung 3.5: Template der neu hinzugefügten Komponente T-316 in CT2
Kapitel 4

Fazit und Ausblick


Es ist von mir geplant, die beiden Komponenten T-310/50 und T-316 noch einmal auf die bekannten Fehler zu untersuchen und einem weiteren Qualitäts-Check zu unterziehen, und dann im Herbst 2018 beide Komponenten in die normale, ausgelieferte Produktiv-Version von CT2 zu integrieren.


Anhang A

Begriffs- und Abkürzungsverzeichnis

A.1 Allgemeine Begriffe und Abkürzungen

CT2 CrypTool 2 – freie Kryptographie-Lernsoftware
ZCO Zentrales Chiffrierorgan der DDR
MfS Ministerium für Staatssicherheit
BStU Der Bundesbeauftragte für die Unterlagen des Staatssicherheitsdienstes der ehemaligen DDR
CCITT-2 Standard für eine 5-Bit-Zeichenkodierung
CCITT V.21 Standard für analoge Wählleitungsmodems
ASCII Standard für eine 7-Bit-Zeichenkodierung
Chiffrierklasse ALPHA Symmetrische Chiffre der DDR; keine Aufzeichnungen bekannt
SKS V/1 Symmetrische Chiffriermaschine/Chiffre der DDR; wird ebenso wie die T-310 der Klasse ALPHA zugeordnet

Verschiedene alte Begriffe:
Spruch Klartext
Spruchschlüssel Initialisierungsvektor
Synchronfolge Nachrichten-Header; Beginn einer Nachricht mit Synchronisationszeichen und dem Spruchschlüssel
A.2 Begriffe und Abkürzungen zur T-310/50

<table>
<thead>
<tr>
<th>Bezeichnung</th>
<th>Deutscher Begriff</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-310/50</td>
<td>Chiffriermaschine zur 5-Bit-Fernschreibübertragung</td>
</tr>
<tr>
<td>T-310/51</td>
<td>Chiffriermaschine zur 8-Bit-Datenchiffrierung</td>
</tr>
<tr>
<td>ARGON</td>
<td>Taktische Bezeichnung in der DDR für die T-310/50; wird in dieser Arbeit als Begriff für den Algorithmus verwendet</td>
</tr>
<tr>
<td>Grundgerät GG</td>
<td>Hauptkomponente der T-310</td>
</tr>
<tr>
<td>Stromversorgungsgerät SV</td>
<td>Stromversorgungsgerät der T-310</td>
</tr>
<tr>
<td>Bediengerät BT</td>
<td>Gerät zum Bedienen der T-310</td>
</tr>
<tr>
<td>Zusatzbediengerät BTZ</td>
<td>Zusatzbediengerät, erfüllt gleiche Aufgaben wie das BT</td>
</tr>
<tr>
<td>Zentraleinheit</td>
<td>Logische Hauptkomponente des Grundgeräts</td>
</tr>
<tr>
<td>Chiffrator</td>
<td>Führt alle kryptographischen Operationen aus; Teil der Zentraleinheit</td>
</tr>
<tr>
<td>Zentralsteuerung</td>
<td>Steuerkomponente für z. B. Takt, Datenbusse und Betriebsmodi; Teil der Zentraleinheit</td>
</tr>
<tr>
<td>Kodeumsetzer</td>
<td>Kodewandler zwischen CCITT-2 und beliebigen Fernschreibzeichen</td>
</tr>
<tr>
<td>FG</td>
<td>Teilkomponenten des Chiffrators</td>
</tr>
<tr>
<td>EE</td>
<td>Eingabeeinheit</td>
</tr>
<tr>
<td>KE</td>
<td>Komplizierungseinheit</td>
</tr>
<tr>
<td>VE</td>
<td>Verschlüsselungseinheit</td>
</tr>
<tr>
<td>SE</td>
<td>Synchronisationseinheit</td>
</tr>
<tr>
<td>PBE</td>
<td>Prüf- und Blockiereinheit</td>
</tr>
<tr>
<td>KEP</td>
<td>Prüfkomplizierungseinheit; Bestandteil der PBE</td>
</tr>
<tr>
<td>T-310-Blockchiffre</td>
<td>Begriff für den Algorithmus der Komplizierungseinheit</td>
</tr>
<tr>
<td>Wurmreihe D-W</td>
<td>Name der T-310-Blockchiffre in der Originaldokumentation</td>
</tr>
<tr>
<td>φ, ϕ oder Φ</td>
<td>Weitere Bezeichnungen für die T-310-Blockchiffre</td>
</tr>
<tr>
<td>Z-Funktion</td>
<td>Boolesche Funktion der Blockchiffre</td>
</tr>
<tr>
<td>Rotationsfunktion von F</td>
<td>Funktion eines Schieberegisters für F; zu Beginn mit F₀ initialisiert</td>
</tr>
<tr>
<td>P-Funktion</td>
<td>Wählt mithilfe eines Wertes aus P des Langzeitschlüssels ein Bit aus U aus.</td>
</tr>
<tr>
<td>D-Funktion</td>
<td>Wählt mithilfe eines Wertes aus D des Langzeitschlüssels ein Bit aus U aus.</td>
</tr>
</tbody>
</table>
Schlüssel:
LZS  Langzeitschlüssel bestehend aus P, D und α
P      Vektor bestehend aus 27 Argumenten für die P-Funktion als Teil des LZS
D      Vektor bestehend aus neun Argumenten für die D-Funktion als Teil des LZS
α     Teil des LZS
S₀     Der 240 Bit lange Zeitschlüssel bestehend aus S₁ und S₂
S₁     Teil des Zeitschlüssels; 120 Bit lang
S₂     Teil des Zeitschlüssels; 120 Bit lang
F₀     Initialisierungsvektor (auch genannt Spruchschlüssel)

Variablen und Bitfolgen:
V     vorgegebener Vektor mit 31 sich nicht wiederholenden Zuständen (natürliche Zahlen 1 – 31); bildet die Grundlage der Schieberegister SR V₂ und SR V₃
M     Über diese Matrix kann V beschrieben werden, indem das Vektor-Matrix-Produkt mit einem 5-Bit-Vektor gebildet wird; V kann auch über das Polynom \( x^5 = x^3 + x^1 + 1 \) beschrieben werden.
U₀    Anfangswert für das Register U; 0x06:9C:7C:85:A3
a     13 Bit langer Vektor, abgeleitet aus der T-310-Blockchiffre
D-W   Name für α der Originaldokumentation
Fₙ    Bitfolge der jeweiligen Runde, ursprünglich ausgehend vom Initialisierungsvektor F₀
Kᵢ    Ein 5-Bit-Klartextzeichen
Bᵢ    Ein 5-Bit-Vektor aus \( (a_7, \ldots, a_{11}) \)
Hᵢ    Ein 5-Bit-Vektor aus \( (a_1, \ldots, a_5) \)
Cᵢ    Ein 5-Bit-Zeichen des Chiffrats
rᵢ    Anzahl der benötigten Runden der Register SR V₂ und SR V₃; ausgedrückt durch die Anzahl an Matrixmultiplikationen
e     kleinste Exponent, sodass \( Hᵢ \cdot M^e = (1, 1, 1, 1, 1) \)
i     Index des Zeichens in einem Klar- oder Geheimtext, bestehend aus mehreren Zeichen
n     Rundenindex der T-310-Blockchiffre

Register und Speicher:
U     36 Bit langes Hauptregister der T-310-Blockchiffre
SRS1  Schlüsselregister der EE bestehend aus 5 einzelnen 24-Bit-Registern SRS1/1 bis SRS1/5
SRS2  Schlüsselregister der EE bestehend aus 5 einzelnen 24-Bit-Registern SRS2/1 bis SRS2/5
F     Schieberegister des Vektors Fₙ
SRU   36-Bit Register U der KE
SRV1  5-Bit Empfangsschieberegister der VE (Zeichenempfang im Chifftrator)
SRV2  5-Bit Schleifenregister der VE (Berechnung des Chiffrats/Klartexts)
SRV3  5-Bit Schleifenregister der VE (Berechnung des Chiffrats/Klartexts)
### A.3 Begriffe und Abkürzungen zur T-316

<table>
<thead>
<tr>
<th>Abkürzung</th>
<th>Begriff/Bezeichnung</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO</td>
<td>Taktische Bezeichnung der T-316</td>
</tr>
<tr>
<td>DES</td>
<td>Data Encryption Standard</td>
</tr>
<tr>
<td>ZS-1</td>
<td>Zeitschlüssel 1; 32-Byte-Schlüssel der T-316</td>
</tr>
<tr>
<td>ZS-2</td>
<td>Zeitschlüssel 2; zusätzlicher 32-Byte-Schlüssel der T-316</td>
</tr>
<tr>
<td>IP</td>
<td>Eingangspermutation; initiale Permutation</td>
</tr>
<tr>
<td>$IP^{-1}$</td>
<td>Ausgangspermutation; Inverse von IP</td>
</tr>
<tr>
<td>KS</td>
<td>Schlüsselableitungsfunktion (nur für DES)</td>
</tr>
<tr>
<td>$n$</td>
<td>Runde von LAMBDA1</td>
</tr>
<tr>
<td>$K_n$</td>
<td>Rundenschlüssel</td>
</tr>
<tr>
<td>$f$</td>
<td>Rundenfunktion</td>
</tr>
<tr>
<td>S-Box</td>
<td>Funktion, die 6 auf 4 Bit reduziert</td>
</tr>
<tr>
<td>$P$</td>
<td>Rundenpermutation</td>
</tr>
<tr>
<td>$E$</td>
<td>Expansionsfunktion</td>
</tr>
<tr>
<td>Register T</td>
<td>Schieberegister zur Ableitung der Schlüssel</td>
</tr>
<tr>
<td>R</td>
<td>Rechter Teilblock</td>
</tr>
<tr>
<td>L</td>
<td>Linker Teilblock</td>
</tr>
</tbody>
</table>
Anhang B

Testvektoren

B.1 Testvektor für die T-310

Es gibt ein von Drobick zur Verfügung gestelltes Originalchiffрат, mit dazugehörigen Schlüsseln und dem Klartext. Die nötigen Daten befinden sich zusammengefasst in Tabelle B.1 und wurden folgendermaßen abgelesen:


Der mittlere Streifen ist die entschlüsselte Nachricht. Gut erkennbar sind die verbleibenden Kontrollzeichen B und K, sowie die MBF. Auf diese folgt der entschlüsselte Klartext.

Der unterste Streifen enthält die originale Nachricht. Beginnend mit dem CCITT-2-Kontrollzeichen LTRS (0x1F) ist das lateinische Alphabet von A bis Z kodiert. Die Nachricht endet mit einem Wagenrücklauf (0x08), Zeilenvorschub (0x02) und einem weiteren LTRS.


Als Langzeitschlüssel wurde der LZS-21 verwendet [4].
### Tabelle B.1: Testvektor – Der Langzeitschlüssel-21 ist dezimal angeführt. Die restlichen Daten sind, sofern nicht anders angegeben, in hexadezimaler Form.

<table>
<thead>
<tr>
<th>Element</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Langzeitschlüssel 21</td>
<td>P:  (36, 4, 33, 11, 1, 20, 5, 26, 9, 24, 32, 7, 12, 2, 21, 3, 28, 25, 34, 8, 31, 13, 18, 29, 16, 19, 6)</td>
</tr>
<tr>
<td></td>
<td>D:  (0, 24, 36, 4, 16, 28, 12, 20, 32)</td>
</tr>
<tr>
<td></td>
<td>α:  1</td>
</tr>
<tr>
<td>Klartext (CCITT-2)</td>
<td>\LTRSABCDEF\HJKLMN\OPQRSTUVWXYZ\r\n\LTRS</td>
</tr>
<tr>
<td></td>
<td>1F:03:19:0E:09:01:0D:1A:14:06:0B:0F:12:1C:0C:18:16:17:0A:05:10:07:1E:13:1D:15:11:08:02:1F</td>
</tr>
<tr>
<td>Nachrichten-Header</td>
<td>19:19:19:19:10:0e:1b:06:00:0e:0b:07:14:12:17:15:05:12:0d:08:1c:13:06:02:13:01:08:0f:0f:0f:0f:0f</td>
</tr>
<tr>
<td>Zeitschlüssel S1 (Binär)</td>
<td>1 24</td>
</tr>
<tr>
<td></td>
<td>0001 0111 0111 0111 1101 0011 25 48</td>
</tr>
<tr>
<td></td>
<td>1000 0110 0001 0111 1010 0011 49 72</td>
</tr>
<tr>
<td></td>
<td>0001 1000 0011 1110 0101 1001 73 96</td>
</tr>
<tr>
<td></td>
<td>1111 1100 0010 0011 1000 1000 97 120</td>
</tr>
<tr>
<td></td>
<td>0001 1111 1110 0001 1101 0010</td>
</tr>
<tr>
<td>Zeitschlüssel S2 (Binär)</td>
<td>1 24</td>
</tr>
<tr>
<td></td>
<td>0001 0111 0111 0111 1101 0011 25 48</td>
</tr>
<tr>
<td></td>
<td>1000 0110 0001 0111 1010 0011 49 72</td>
</tr>
<tr>
<td></td>
<td>0001 1000 0011 1110 0101 1001 73 96</td>
</tr>
<tr>
<td></td>
<td>1111 1100 0010 0011 1000 1000 97 120</td>
</tr>
<tr>
<td></td>
<td>0001 1111 1110 0001 1101 0010</td>
</tr>
</tbody>
</table>
B. Testvektoren

Abbildung B.1: Chiffat (oben), Klartext (unten) und entschlüsselter Klartext (mittig) einer originalen T-310

Abbildung B.2: Originaler Zeitschlüssel der T-310
B.2 Testvektor für die T-316

Der Testvektor in Tabelle B.2 wurde von Drobick zur Verfügung gestellt. Das Erstellen weiterer Chiffrate sollte mit der noch funktionierenden Maschine des NVA-Museums Harnekop möglich sein.

Tabelle B.2: Testvektor für die T-316

<table>
<thead>
<tr>
<th>Element</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nachricht (Text)</td>
<td>AAAAAAAA</td>
</tr>
<tr>
<td>Chiffrat (hexadezimal)</td>
<td>2C 0A 7A 4A 1D 46 11 57 9A B5 0C 2F E0 97 07 2B C3 3F BB A4</td>
</tr>
<tr>
<td>Nachricht (Text)</td>
<td>AAAAAAAA</td>
</tr>
<tr>
<td>Chiffrat (hexadezimal)</td>
<td>F9 D0 64 49 55 6C 66 B0 94 64 BE BC AF E7 AC DC 72 5B 4C 5E</td>
</tr>
</tbody>
</table>
Quellenverzeichnis

Literatur


Online-Quellen


Abbildungsverzeichnis

2.1 Foto des T-310/50-Gerätesystems samt Beschriftungen ............ 5
2.2 Vereinfachte Skizze der T-310-Gerätekomponenten ................. 6
2.3 Skizze des Chiffriers im Original und vereinfacht .................. 8
2.4 Graphische Darstellung der Abläufe in der T-310-Blockchiffre ......... 14
2.5 Graphische Darstellung einer neuen Runde in der T-310-Blockchiffre .. 15
2.6 Template der überarbeiteten Komponente T-310 in CT2 ............... 19

3.1 Das Gerätesystem T-316 in der zivilen Variante ...................... 21
3.2 Schieberegister T ................................................. 26
3.3 Vergleich zwischen der Funktion f von LAMBDA1 und DES .......... 27
3.4 Schematischer Vergleich zwischen den Verschlüsselungsalgorithmen von
   LAMBDA1 und DES .............................................. 28
3.5 Template der neu hinzugefügten Komponente T-316 in CT2 ............. 31

B.1 Chiffrat, Klartext und entschlüsselter Klartext einer originalen T-310 .. 39
B.2 Originaler Zeitschlüssel der T-310 ................................. 39
Tabellenverzeichnis

2.1 Auflistung verschiedener Blockchiffren und deren Effizienz ........................... 16
3.1 SRAM-Speicherbereich der T-316 ................................................................. 22
3.2 EPROM-Speicherbereich der T-316 ................................................................. 22
3.3 Initiale Permutation IP ................................................................................. 24
B.1 Testvektor für die T-310 .............................................................................. 38
B.2 Testvektor für die T-316 .............................................................................. 40