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

Michael Altenhuber


BACHELORARBEIT

Nr. 1510239014-A

eingereicht 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

# Vorwort

Da nach Abgabe dieser Abschlussarbeit noch weitere Erkenntnisse zum bearbeiteten Thema gewonnen wurden, möchte ich in diesem Vorwort ergänzend auf den aktuellen Kenntnisstand eingehen. Das betrifft vor allem die T-316, deren Quellcode detailliert von Herrn Drobick analysiert wurde. Dabei fand er heraus, dass die tatsächlich gebauten T-316 nicht den in den Dokumenten beschriebenen LAMBDA1-Algorithmus einsetzten, sondern eine leicht modifizierte Variante. Das Plug-in für CrypTool 2 wurde aufgrund dieser Erkenntnis auf LAMBDA1 umbenannt, da es den Algorithmus nach der gegebenen LAMBDA1-Spezifikation implementiert. Weiters wurden alle Tests der originalen Dokumente digitalisiert, zusammengefasst und in meinem *GitHub*-Repository[1] unter der Lizenz GPL 3.0 veröffentlicht. Dort befindet sich der LAMBDA1-Algorithmus in der identischen Ausführung aus CrypTool 2 mit den Tests, einem Kommandozeilen-Programm und einer ergänzenden Kurzbeschreibung. Die Implementierung erfüllt alle in den Dokumenten spezifizierten Tests. Zu guter Letzt wurden in dieser Fassung noch einige wenige Rechtschreibfehler ausgebessert.

# 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 CrypTool 2 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

In dieser Arbeit werden zwei historische Chiffriermaschinen aus der Deutschen Demokratischen Republik (DDR) analysiert und die Implementierung der Algorithmen in moderner Software beschrieben. Die beiden Maschinen T-310/50 und T-316 wurden in den 70er und 80er Jahren in der DDR entwickelt. Betrachtet werden der Aufbau der Chiffriermaschinen und ihr historischer Kontext, sowie die Funktionsweise der Algorithmen. Die Implementierung erfolgt für die freie Open-Source-Software CrypTool 2.

Bei der T-310/50 handelt es sich um eine komplexe Stromchiffre, die zur Ableitung der pseudozufälligen Bits aus dem Schlüssel einen Algorithmus ähnlich einer Blockchiffre verwendet. Dabei wurde der gesamte Prozess in logischen Schaltungen realisiert. Das erschwert die Implementierung in Software. Es gibt in CrypTool 2 schon eine eingeschränkte Version der T-310, deren Code verbessert wird. Zusätzlich werden einige Fehler aufgezeigt und verbessert.

Die neuere T-316 arbeitete bereits mit einem Mikroprozessor und verwendete als Algorithmus LAMBDA1. Dieser war eine Weiterentwicklung des Data Encryption Standard (DES). Dabei wurde versucht, die Sicherheit des DES zu erhöhen. Die dabei vorgenommenen Änderungen werden erläutert und eine vorsichtige Einschätzung der Sicherheit gegeben. Die Implementierung für CrypTool 2 gestaltet sich aufgrund der engen Verwandtschaft zu DES und den Eigenschaften der T-316 einfacher als die Implementierung der T-310.

Als Resultat liegen die Implementierungen der Chiffriermaschinen vor. Diese sind ähnlich zu den Originalmaschinen, erreichen aber keine vollständige Kompatibilität. Die neuen Implementierungen werden in das Repository von CrypTool 2 eingegliedert werden. Eine Kompatibilität zu den Originalmaschinen wird angestrebt und könnte durch zukünftige Verbesserungen erreicht werden.

# 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

In der Geschichte der Kryptographie wurden zahlreiche Verfahren und Algorithmen entwickelt und verwendet. Informationen zu Funktionsweise, mathematischen und technischen Details geraten dabei des Öfteren in Vergessenheit, oder sind nur mehr schwierig zu bekommen. Dazu zählen auch Algorithmen und dazugehörige Chiffriermaschinen, die in der Deutschen Demokratischen Republik (DDR) in den Jahren 1951 bis 1990 entwickelt wurden. Diese sind im westlichen Raum wenig erforscht. Das Ziel dieser Arbeit ist es, die zwei Maschinen T-310/50 und T-316 inklusive der Algorithmen hinsichtlich ihrer Funktionsweise zu analysieren und verständlich darzustellen. Im Zuge dieses Prozesses sollen die Algorithmen in die freie Software CrypTool 2 (CT2) eingebaut und dokumentiert werden. Die T-310/50 ist bereits in CT2 enthalten, wird aber verbessert. Eine Beschreibung der Funktionsweise der T-310/50 ist in Kapitel 2 enthalten; weitere Punkte zur Implementierung finden sich in Abschnitt 2.5. Die T-316 wird neu zu CT2 hinzugefügt. Hierzu gibt es eine Beschreibung der Maschine und des Algorithmus in Kapitel 3 und der Implementierung in Abschnitt 3.4.

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 Chiffrierorgan (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-

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

Die T-310 wurde ab 1973 vom ZCO entwickelt. In diesem Zeitraum entstanden auch andere bekannte Algorithmen, wie etwa der Data Encryption Standard der NSA. Die T-310 wurde ab 1982 verwendet, fand bei diversen Organen der DDR Verbreitung und wurde bis zur Auflösung im Jahre 1990 verwendet [15].

Die T-316 ist eine Entwicklung der späten 80er Jahre. Sie hat keinen technischen Bezug zur T-310 und wurde auf Grundlage des DES entworfen. Wegen des kurzen Zeitraums der Nutzung bis zur Auflösung der DDR erreichte sie keine große historische Relevanz [24].

## 1.2 Über CrypTool 2

CT2 ist eine weit verbreitete Kryptologie-Lernsoftware mit einer graphischen Benutzeroberfläche zur visuellen Programmierung. CT2 deckt dabei verschiedene moderne und historische Algorithmen ab – sowohl zur Kryptographie als auch zur Kryptoanalyse. CT2 ist freie Open-Source-Software und wird in C# auf Basis des Microsoft .NET Frameworks entwickelt. Das Projekt wird in einem SVN-Repository verwaltet, das über die offizielle Projektwebseite[1] erreichbar ist. Besonders ist, dass Benutzer verschiedene Algorithmen beliebig aneinanderketten können. So kann zum Beispiel das Ergebnis eines Verschlüsselungsalgorithmus automatisiert an ein Hashverfahren weitergeleitet werden.

Der Aufbau des Programms erfolgt modular über ein Plug-in-System und basiert auf der Windows Presentation Foundation. Zum Erstellen neuer Funktionen (Komponenten) kann eine beliebige C#-Entwicklungsumgebung verwendet werden. Zur Dokumentation des Codes bieten sich die in C# integrierten XML Documentation Comments an. Durch diese lässt sich Code strukturiert kommentieren und automatisch vom Compiler eine Codedokumentation generieren. Die Bedienungsanleitung für den Benutzer wird aus einer strukturierten XML-Datei generiert.

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

Es stehen archivierte Dokumente der Abteilung XI der BStU zur Verfügung, die auch von Herrn Jörg Drobick auf seiner Webseite aufbereitet worden sind [16]. Als wichtigste Quellen werden in dieser Arbeit für die T-310 die Originaldokumente aus dem NVA-Museum Harnekop, aufbereitete Beschreibungen in [10][3] und eine Kryptoanalyse [14] verwendet.

Für den Algorithmus LAMBDA1 der T-316 wurde ebenfalls auf archivierte Dokumente des NVA-Museums Harnekop zurückgegriffen. Eine Kryptoanalyse oder Untersuchung zur Sicherheit war lediglich als kurzer Abschnitt in [3] zu finden.

[1] https://www.cryptool.org/de/cryptool2

An dieser Stelle geht ein besonderer Dank an das NVA-Museum Harnekop für die Zusammenarbeit, insbesondere an Herrn Jörg Drobick, der mit großen Engagement als Berater für diese Arbeit zur Verfügung stand. Das Museum hat auch noch funktionierende Maschinen beider Typen ausgestellt.

# Kapitel 2

# T-310/50

Bei der T-310/50 handelt es sich um eine elektronische Chiffriermaschine, die sich aus mehreren Komponenten zusammensetzt (Gerätesystem). Sie ist ausgelegt, um Fernschreibzeichen zu ver- und entschlüsseln. Als Zeichenkodierung wird der Standard CCITT-2 verwendet [5], der auch als International Telegraph Alphabet No. 2 bekannt ist. In diesem sind 32 Zeichen in fünf Bit kodiert [27]. Die Übertragung kann über Funk und Richtfunk, oder mittels Fernmelde-Fernschreibnetzen im Wähl- oder Standleitungsbetrieb erfolgen [5]. Erste Dokumente zur Entwicklung gibt es seit Mitte der 70er Jahre. Die Zulassung über die Nutzung des Fernmeldenetzes der Deutschen Post der DDR ist vom 05.08.1982 [23]. Die Nutzung erfolgte bis zur Auflösung der DDR im Jahr 1990 im öffentlich zivilen und im militärischen Bereich [15].

Der verwendete Verschlüsselungsalgorithmus hat keinen eigenen Namen. Er wird lediglich einem älteren Algorithmus der DDR namens „Chiffrierklasse ALPHA“ zugeordnet, über den keine Aufzeichnungen mehr bekannt sind [4]. 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.

Die elektronische Komponente des Grundgeräts heißt Zentraleinheit und umfasst die Hauptfunktionen des Geräts. Es werden für verschiedene Komponenten mehrere Taktraten benötigt, die von der Zentralsteuerung erzeugt werden. Der Haupttakt für das Gerät ist 76,8 kHz. Des Weiteren steuert die Zentraleinheit die Betriebsmodi, Datenleitungen und die Schnittstelle zum Chiffrator. Dieser ver- und entschlüsselt Nachrichten und ist in Abschnitt 2.2 näher beschrieben. Zur technischen Realisierung sind hauptsächlich logische Schaltungen auf Basis der Transistor-Transistor-Logik (TTL) verbaut. Auch der komplette kryptographische Prozess ist verdrahtet. Nur der optionale Kodeumsetzer, der erst später in der Entwicklung geplant und gebaut wurde, verwendete

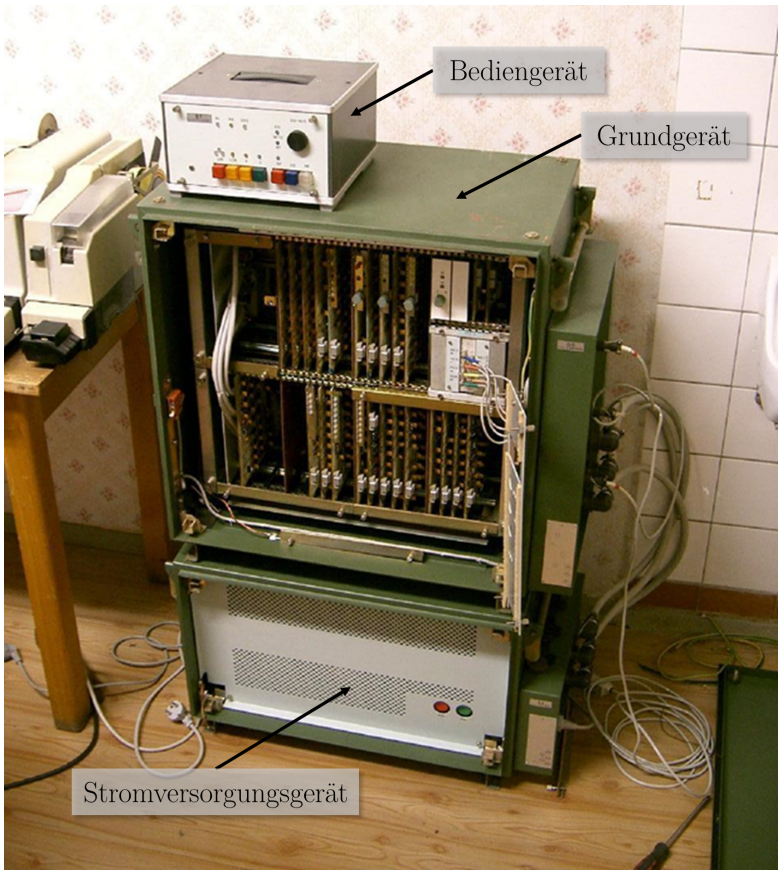



**Abbildung 2.1:** Das T-310/50-Gerätesystem – die Beschriftungen zeigen auf die jeweiligen Komponenten. Das ursprüngliche Foto stammt aus [3].

einen Mikroprozessor aus der Prozessorfamilie U880, der auf dem Zilog Z80 basiert. Der Kodeumsetzer hat die Aufgabe, beliebige Fernschreibzeichen in druckbare Zeichen zu verwandeln. So konnten Nachrichten beim Fehlen einer Fernschreibverbindung ausgedruckt werden. Es gibt mehrere Zwischenspeicher, wenn Daten im Gerät verschoben werden. Einen weiteren großen Anteil des Grundgeräts umfasst die Steuerung der Fernschreibperipherie zum Senden und Empfangen von Daten. Das Grundgerät wird über das Stromversorgungsgerät mit Energie gespeist.

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).

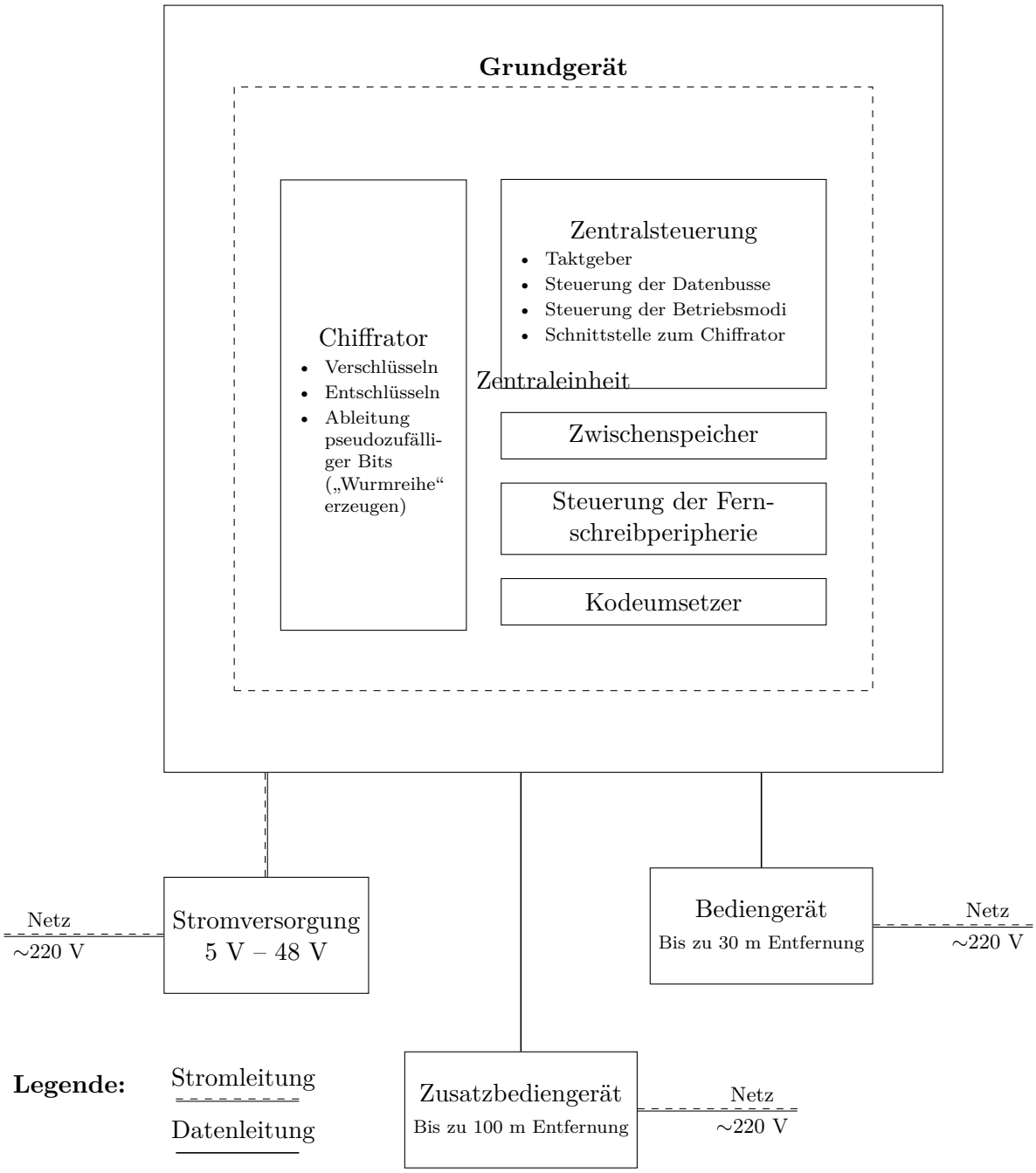



**Abbildung 2.2:** Stark vereinfachte Skizze der Gerätekomponenten der T-310; sie ist angelehnt an das Prinzipschaltbild der T-310 der Originaldokumentation in [5, Seite 234], wobei einige Komponenten aus Gründen der Übersichtlichkeit zusammengefasst oder weggelassen wurden. Zu sehen sind das Grundgerät und deren wichtigste Komponenten, die Stromversorgung und die externen Bediengeräte.

## 2.2 Chiffrator

Als Chiffrator wird jene Komponente der T-310 bezeichnet, welche die kryptographischen Operationen durchführt. In ihr läuft der ARGON-Algorithmus. Diese Komponente ist in Buch 2 der technischen Dokumentation beschrieben [6]. Der Chiffrator ist in folgende fünf Funktionseinheiten unterteilt.

- Eingabeeinheit EE – Schnittstelle zur Schlüsseleingabe
- 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

Die Komplizierungseinheit liegt in doppelter Ausführung vor (Prüfkomplizierungseinheit KEP), wobei die redundante Variante Teil der Prüf- und Blockiereinheit ist. Beide Komplizierungseinheiten führen parallel dieselben Berechnungen durch. Das Ergebnis wird verglichen und nur bei Übereinstimmung weiterverwendet. Warum diese besondere Vorsichtsmaßnahme verbaut wurde, ist nicht bekannt. Abbildung 2.3 zeigt eine Skizze des Chiffrators aus der Originaldokumentation, welche die Komplexität der Komponente zeigt. Zusätzlich wurde eine vereinfachte Variante mit den Bezeichnungen aus dieser Arbeit hinzugefügt.

## 2.3 Algorithmus der T-310

Der ARGON-Algorithmus wird in der T-310 verwendet. Dieser beschreibt den gesamten Prozess zum Ver- und Entschlüsseln von Daten. Es ist ein symmetrisches Verfahren, genauer gefasst eine Stromchiffre. Die Ableitung eines pseudozufälligen Bitstroms erfolgt über eine Art Blockchiffre (siehe Abschnitt 2.3.3). Die Verschlüsselung des Zeichenstroms erfolgt über eine Operation, die sich von der üblichen XOR-Verknüpfung[1] anderer Stromchiffren unterscheidet (siehe Abschnitt 2.3.2).

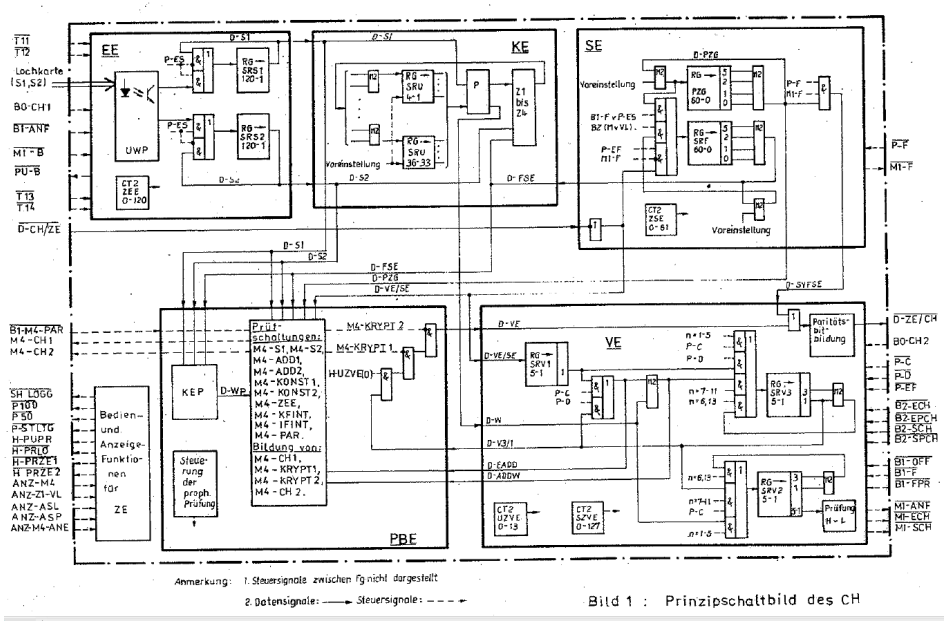
### 2.3.1 Schlüssel

Der Algorithmus verwendet drei verschiedene Schlüssel. Diese heißen in der ursprünglichen Dokumentation Langzeitschlüssel LZS, Zeitschlüssel $S_0$ und Spruchschlüssel $F_0$.

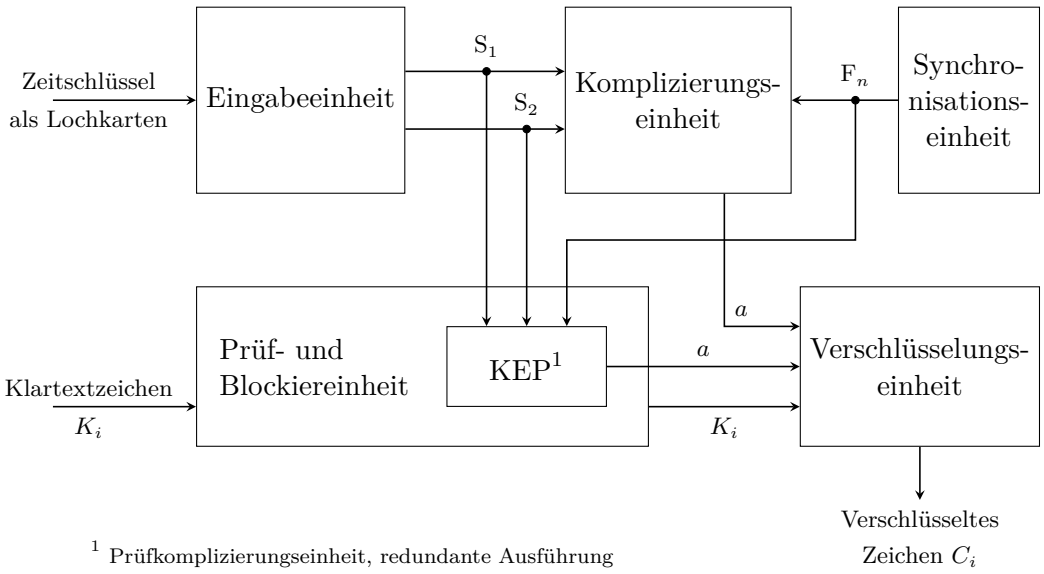
Der **Langzeitschlüssel LZS** befindet sich auf drei fest installierten, aber austauschbaren Steckkarten [21]. Er gliedert sich in die drei Teile P, D und $\alpha$. Diese werden bei der Ableitung der pseudozufälligen Bits verwendet. Bei allen drei Teilen handelt es sich um einfache Argumente für Funktionen. 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

[1] Das entspricht einem One-Time-Pad mit pseudozufälligen Bits.




**(a)** Original




**(b)** Vereinfacht und angepasst

**Abbildung 2.3:** Skizze des Chiffrators 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.“ [8, Seite 46]. Die Aussage entspricht der Anzahl an Schlüsseln die laut [21] auch real zum Einsatz kamen.

Der **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 [4]. Im Zusammenhang mit der Parität der Zeitschlüssel kommt in mehreren Dokumenten und Analysen[2] 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 [12, Seite 8] zuerst täglich durchgeführt werden. In einer Kryptoanalyse während der Entwicklung [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 **Spruchschlüssel $F_0$** handelt es sich im modernen Sinn um eine Nonce als Initialisierungsvektor. Es ist eine zufällig[3] 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 Chiffrat 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 Abschnitt 2.3.3 und der dazugehörigen Abbildung 2.4 ersichtlich.

### 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 [6, Seite 7] und [1], sowie [10]. Bei der Verknüpfung werden die 5-Bit-Zeichen des Klartexts zusätzlich zur üblichen XOR-Verknüpfung noch in einem linear rückgekoppelten Schieberegister rotiert.

Der zugrundeliegende Vektor V des Schieberegisters hat 31 Zustände[4] – alle natürlichen Zahlen von eins bis 31. Dieser Vektor lässt sich durch das Polynom $x^5 = x^3 + x^1 + 1$

[2] Dieser Fehler kommt unter anderem auch in [10] und [8] vor.

[3] Die T-310/50 verfügt über einen Rauschgenerator. Die T-310/51 generierte den Zufall über das Signal beim Herausziehen einer Lochkarte.

[4] Der komplette Vektor V ist (1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 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 (1,1,1,1,0), der wiederum mit der Matrix multipliziert werden kann, um (1,1,1,0,0) 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:

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

Die Anzahl der Matrixmultiplikationen (oder je nach Betrachtungsweise der Runden im Schieberegister) $r_i$ ist abhängig vom deterministisch generierten, pseudozufälligen 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 $(a_{i,7}, ..., a_{i,11})$
$H_i =$ 5-Bit-Vektor aus $(a_{i,1}, ..., a_{i,5})$

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$e =$ 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-verknüpft wird. Danach beginnt die zweite Schleife, bei der wieder SRV2 und SRV3

parallel geschoben werden. Erreicht SRV2 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 pseudozufällige 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 $\ll 1$) | (((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: SRV2 = SRV3
11: SRV3 = $B_i \oplus K$

12: **while** SRV2 $\neq 11111_b$ **and** SRV2 $\neq 00000_b$ **do**
13: SRV2 = (SRV2 $\ll 1$) | (((SRV2 $\gg 4$)$\oplus$(SRV2 $\gg 2$)) & 1)
14: SRV3 = (SRV3 $\ll 1$) | (((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 Wurmreihe D-W oder als Funktion in den Variationen $\phi$ , $\varphi$ und $\Phi$ bezeichnet. Courtois [14] 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[5]. 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 0101 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[6] – 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

[5] Bei der Anzahl der Runden ergeben sich aus technischer Sicht Abweichungen, die in Abschnitt 2.5 erläutert werden.

[6] Die Z-Funktion ist vier Mal ($Z_1$ – $Z_4$) baugleich in der Komplizierungseinheit vorhanden, um alle Werte der P-Funktion gleichzeitig verarbeiten zu können.

24. Bit aus U. Exemplarisch wird hier der LZS-33[7] aus dem Jahr 1990 aufgeführt.

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

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:

$$\begin{aligned}\mathrm{Z}(e_1, e_2, ..., e_6) = {} & 1 \oplus e_1 \oplus e_5 \oplus e_6 \oplus e_1e_4 \oplus e_2e_3 \oplus e_2e_5 \\ & \oplus e_4e_5 \oplus e_5e_6 \oplus e_1e_3e_4 \oplus e_1e_3e_6 \oplus e_1e_4e_5 \oplus e_2e_3e_6 \oplus e_3e_5e_6 \\ & \oplus e_1e_2e_3e_4 \oplus e_1e_2e_3e_5 \oplus e_1e_2e_5e_6 \\ & \oplus e_2e_3e_4e_6 \oplus e_1e_2e_3e_4e_5 \oplus e_1e_2e_3e_4e_5e_6\end{aligned}$$

Courtois geht in [3] näher auf die Anforderungen und Bedeutung der Z-Funktion ein. Er weist darauf hin, dass die Entwickler des Algorithmus zumindest ein grundsätzliches Verständnis für differenzielle Kryptoanalyse hatten und dies beim Entwurf der Z-Funktion berücksichtigten.

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

Eine Runde der Blockchiffre kann in zwei große Schritte unterteilt werden. Der erste Schritt ist in Abbildung 2.4 dargestellt und umfasst die Befüllung des logischen Registers T. Dieses ist nicht physisch vorhanden, wird aber zumeist verwendet, um die Funktionsweise einfacher darstellen zu können. Es hat eine Länge von 9 Bit und entspricht damit der Länge von D. Zu Beginn werden 27 Bit aus U durch die P-Funktion ausgewählt. Zum Großteil fließen diese in die Z-Funktionen ein. Ein Bit aus $\mathrm{S}_2$ ist ein Eingabewert für $\mathrm{Z}_4$. Die übrigen Bits aus der P-Funktion und die Ergebnisse der Z-Funktion bilden T. Zusätzlich fließt ein Bit von $\mathrm{F}_n$ und nochmals einmal das Bit aus $\mathrm{S}_2$ ein.

Im zweiten Schritt wird mithilfe des soeben gebildeten T und der D-Funktion der neue Wert für U gebildet. Das ist in Abbildung 2.5 dargestellt. Dazu wird U zunächst in neun logische Gruppen zu je vier Bit eingeteilt. Ein Bit der D-Funktion und ein Bit von T werden XOR-verknüpft und an die erste Stelle einer Gruppe gelegt. Die restlichen Bits werden jeweils an die nächste Stelle geschoben. Zusätzlich fließt noch ein Bit aus $\mathrm{S}_1$ ein. Damit ist eine Runde vollständig abgeschlossen. Jede 127. Runde wird ein Bit, bestimmt vom Wert $\alpha$ des Langzeitschlüssels, dem Ergebnisvektor $a$ hinzugefügt.

Auffällig bei P ist, dass an den Stellen $\mathrm{P}_3$, $\mathrm{P}_6$, $\mathrm{P}_8$, $\mathrm{P}_{14}$ und $\mathrm{P}_{23}$ in vielen Schlüsseln immer dieselben Werte stehen. Dies lässt sich vor allem bei Schlüsseln beobachten, die auch für den produktiven Einsatz genutzt wurden. Abbildung 2.4 verdeutlicht das durch die gestrichelten Linien der P-Funktion.

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.

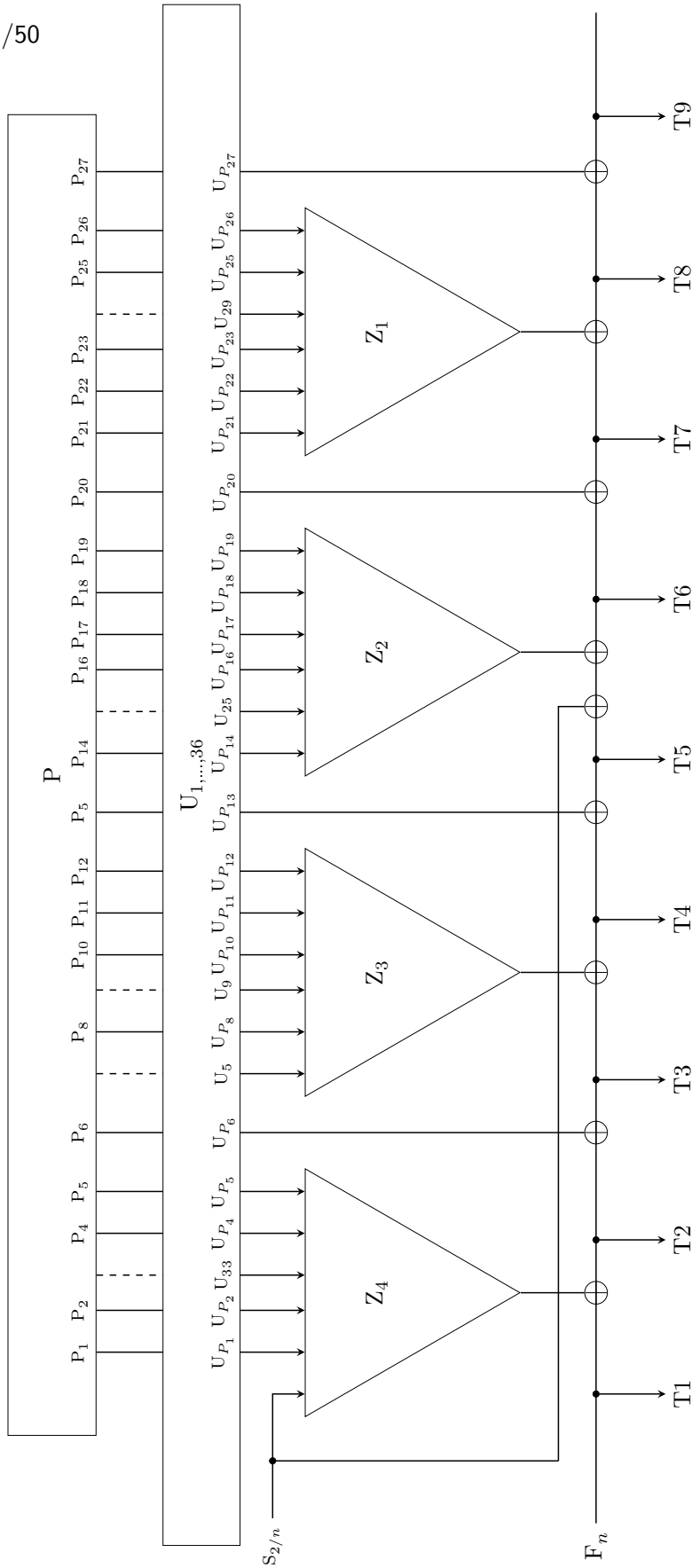



**Abbildung 2.4:** Graphische Darstellung der Abläufe in der T-310-Blockchiffre zur Bildung von T

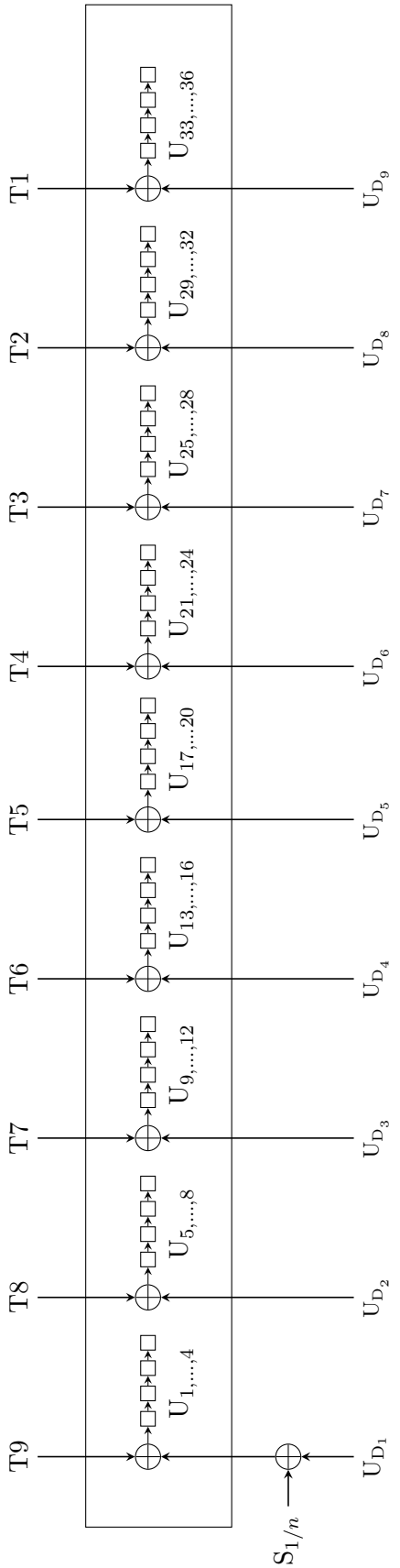



**Abbildung 2.5:** Graphische Darstellung einer neuen Runde in der T-310-Blockchiffre (Schiebevorgang)

**Tabelle 2.1:** Auflistung verschiedener Blockchiffren und deren Effizienz nach [3]

| Algorithmus | Jahr | Land | Blockgröße | Schlüsselgröße | Runden pro verschlüsseltem Bit | Genutzter Anteil des Schlüssels[1] |
|---|---|---|---|---|---|---|
| SKS V/1 | 1973 | DDR | 27 | 208 | 119 | 1 % |
| T-310[2] | 1976 | DDR | 36 | 240 | 330,2 | 0,8 % |
| DES | 1974 | USA | 64 | 56 | 0,25 | 75 % |
| GOST (MAGMA) | 1989 | Russland | 64 | 256 | 0,5 | 12,5 % |
| TEA | 1994 | Großbritannien | 64 | 128 | 1 | 50 % |
| AES | 1996 | Belgien | 128 | 128 | 0,08 | 100 % |
| PRESENT | 2007 | Deutschland, Frankreich | 64 | 128 | 2,1 | 100 % |
| Simon/ Speck | 2013 | USA | 64 | 128 | 0,42 | 100 % |

[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][3] 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

Qualität geprüft wurden. Einige Beispiele für im Nachhinein gefundene, unsichere Schlüssel mit verschiedenen schwachen Eigenschaften, sind in [14, Seite 63] zu finden. Als Beispiel sei hier der LZS-702 aufgeführt, der bei gewissen Eingabewerten zu linearen Wiederholungen nach jeweils acht Runden führt. Das bedeutet, dass bei Verwendung dieses Schlüssels die Eingabewerte 17 und 21 eine Wahrscheinlichkeit von 1.0 haben, nach acht Runden wiederum 17 und 21 zu ergeben:

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

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

Es gibt bereits mehrere Referenzimplementierungen: Zum einen die bereits vorhandene Version in CT2, die sich stark an die Simulationssoftware Drobicks [22] anlehnt. Zum anderen gibt es im Anhang I.4 von [14] eine Implementierung in C++, die nur die T-310-Blockchiffre (siehe Abschnitt 2.3.3) umfasst. In Anlehnung daran gibt es auch eine Python-Implementierung [13].

Im Zuge dieser Arbeit soll die Implementierung in CT2 verbessert werden, da sie laut Drobick [4] im Bezug auf seine Implementierung fehlerhaft ist. Außerdem entspricht der Code teilweise nicht gängigen C#-Coding-Standards, weil Funktionen, Variablen und Kommentare inkonsistent auf Deutsch oder Englisch benannt sind. Zudem soll die Dokumentation und Nachvollziehbarkeit verbessert werden.

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 Prüfkomplizeinheit werden nicht umgesetzt: Die Eingabe wird von CT2 selbst abgedeckt und die Prüfkomplizeinheit 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 wichtiger Aspekt der Implementierung der T-310 in einem Programm mit einem einzelnen Thread ist die schwierige Darstellbarkeit der Nebenläufigkeit der logischen Schaltungen. Vor allem in der T-310-Blockchiffre werden fast alle Berechnungen parallel abgewickelt. Das wirft zwei Fragen auf. Einerseits, welchen Zustand das System zu einem gewissen Zeitpunkt oder Takt hat, und andererseits, wie sich dieser Zustand sequentiell abbilden lässt. Eine Multithreaded-Variante würde aufgrund der hohen Anzahl an Komponenten sehr komplex ausfallen und wird deswegen nicht implementiert.

Zu betrachten ist insbesondere die zeitliche Abfolge beim Aufrufen der Z-Funktionen. Drobicks Version geht von T9 in Richtung T1 vor. Das würde einen Ablauf von rechts nach links in Abbildung 2.4 bedeuten. Die CT2-Implementierung und jene, die an Courtois angelehnt sind, berechnen von T1 aufwärts, also von links nach rechts.

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-signifcant) 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 51[8]. 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).

Zusätzlich dazu erlaubt die überarbeitete Komponente die Auswahl mehrerer verschiedener Langzeitschlüssel. Bei der alten Version stand nur der LZS-31 zur Verfügung. Zur besseren Verständlichkeit der Schlüsseleingabe werden die Zeitschlüssel $S_1$ und $S_2$ über jeweils eigene Schnittstellen angebunden. Dadurch kann dem Benutzer Feedback ausgegeben werden, falls ein Schlüssel nicht in Ordnung ist. Abbildung 2.6 zeigt das überarbeitete Template mit den neuen Optionen und einer zusätzlichen Beschreibung.

Im Laufe der Entwicklung wurden Details des Verfahrens immer wieder mit Drobick diskutiert. So konnte Drobick [4] bei der seiner Implementierung [22] noch zwei wichtige Fehler finden und korrigieren, die auch in die Implementierung dieser Arbeit aufgenommen werden. Zum einen durchläuft die Blockchiffre vor dem Verarbeiten des ersten Zeichens 133 Runden, deren Ergebnis nicht verwendet wird. Zum anderen werden zum Bilden des Vektors $a$ nach den ersten 13 Bit (1651 Runden) noch einmal zusätzliche 127 Runden durchlaufen. Das hat den Grund, dass in der Schaltung ein Ergebnisbit immer erst in den Runden des nächsten Bits gespeichert wird. Das Ergebnis der zusätzlichen Runden wird verworfen, der Zustand der Blockchiffre ändert sich aber dadurch.

Ein Testvektor für die T-310/50 befindet sich in Anhang B. Es ist weder mit Drobicks Implementierung, noch mit der alten CT2-Version und der aktuellen Implementierung dieser Arbeit möglich, die Daten korrekt zu entschlüsseln. Es bleibt also eine Annäherung an die ursprüngliche Maschine. Eine Möglichkeit zur originalgetreuen Umsetzung wird in Kapitel 4 erläutert. Die Implementierungen aus [14] und [13] enthalten nicht den kompletten Ver- und Entschlüsselungsalgorithmus. Deswegen wird nicht untersucht, ob diese Implementierungen die Originaldaten entschlüsseln können.

[8] Diese entspricht auch dem Programm, das Drobick unter [22] zur Verfügung stellt.

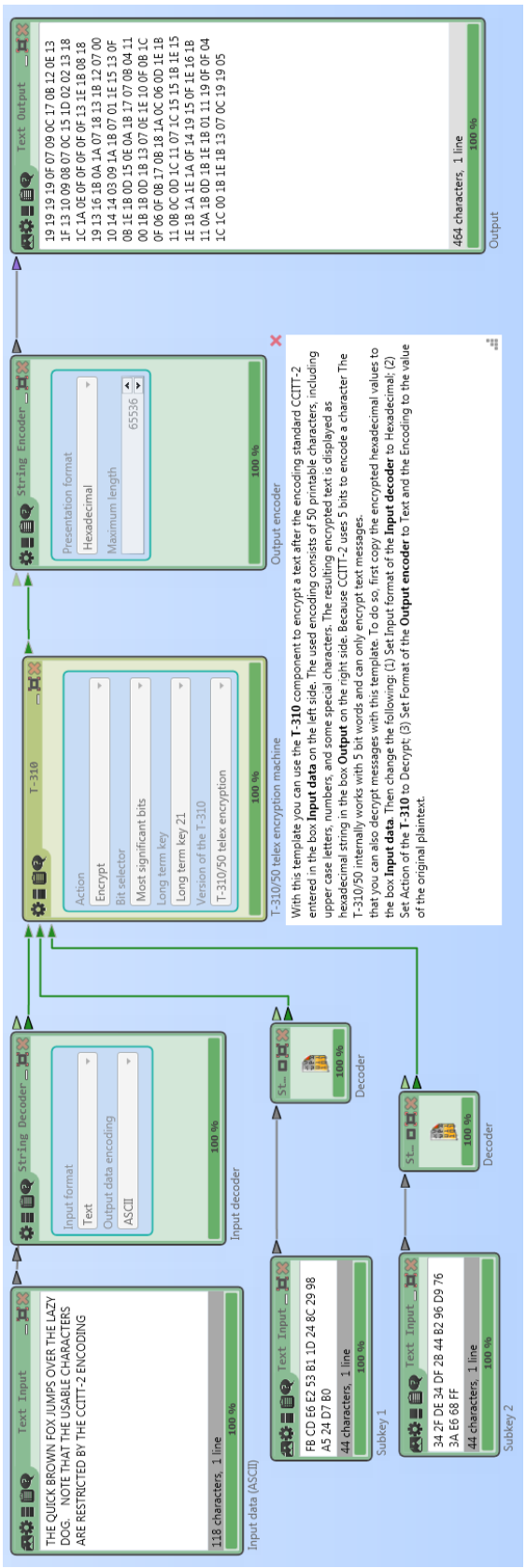



**Abbildung 2.6:** Template der überarbeiteten Komponente T-310 in CT2

# Kapitel 3

# T-316

Die T-316 GO ist ein Chiffriergerät basierend auf einem Mikroprozessor. Dieses setzt sich aus mehreren Komponenten zusammen und wird deswegen auch als Gerätesystem T-316 bezeichnet. GO war die taktische Bezeichnung der DDR für das Gerät, ähnlich wie ARGON für die T-310/50 (siehe Kapitel 2). Es handelt sich dabei um keine Abkürzung. Als Zeichenkodierung verwendet die T-316 den 7-Bit-ASCII-Zeichensatz und den Standard CCITT-2, der auch als International Telegraph Alphabet No. 2 bekannt ist und 5 Bit lang ist [27]. Die Daten werden vor dem Verschlüsseln von ASCII nach CCITT-2 umkodiert, beziehungsweise umgekehrt beim Entschlüsseln. Das System kann über verschiedene Übertragungskanäle Chiffrate austauschen. Dazu zählen Stand- und Wählverbindungen sowie Draht- oder Funkkanäle. Standardmäßig war eine Übertragung über das Telefonnetz der Deutschen Post der DDR vorgesehen. Deswegen war ein Akustikkoppler oder ein Modem zur Übertragung der binären Daten über das analoge Netz Bestandteil des Systems, die eine Durchsatzgeschwindigkeit von 200 Bit/s erreichten. Für Peripheriegeräte standen vier Anschlüsse zur Verfügung. Dazu zählen eine 5-polige Diodenbuchse (XB1) für den Akustikkoppler, eine weitere 5-polige Diodenbuchse (XB2) für den Anschluss eines Magnettonbandgeräts, eine 9-polige Cannonbuchse (XB3) für den Anschluss eines Modems nach CCITT V.21 oder der Funkperipherie und eine 25-polige Cannonbuchse (XB4) zum Anschließen eines Druckers. Die T-316 befindet sich für zivile Zwecke in einem Attachékoffer (Abbildung 3.1) oder in der militärischen Variante in einem Metallgehäuse. Die Stromzufuhr erfolgt entweder über das beigefügte Steckernetzteil oder über Batterien. Klartexte können über eine Tastatur eingegeben werden und dürfen maximal 1000 Zeichen lang sein. Als Ausgabe- und Informationsfeld dient ein kleines LCD-Display mit einer Auflösung von 120x10 [7]. Als Verschlüsselungsverfahren kommt der sogenannte LAMBDA1-Algorithmus zum Einsatz (siehe Abschnitt 3.3), eine Entwicklung auf Basis des Data Encryption Standard (DES) [2].

Zeitlich einzuordnen ist die Entwicklung gegen Ende der 80er Jahre. Laut Drobick [24] wurden bis 1989 6000 Platinen der T-316 hergestellt.

## 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-

**Abbildung 3.1:** Das Gerätesystem T-316 in der zivilen Variante[17]

gabe von Nachrichten und die Ver- und Entschlüsselungsroutinen. Diese sind umgesetzt auf einem Computer mit einem 8-Bit-CMOS-Mikroprozessor der Familie U880, der auf dem Zilog Z80 basiert. Die technische Dokumentation spezifiziert nicht, welcher Typ genau verbaut wurde.

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 Zeilennummer 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 Programmcode 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

**Tabelle 3.1:** SRAM-Speicherbereich der T-316

| | | | | |
|---|---|---|---|---|
| 0x8000 | - | 0x8FFF | 4096 Byte | |
| 0x9000 | - | 0x93FF | 1024 Byte | |
| 0xDC00 | - | 0xDFFF | 1024 Byte | |
| 0x9400 | - | 0x97FF | 1024 Byte | gestützter Speicherbereich |
| 0xA000 | - | 0xA3FF | 1024 Byte | Bildwiederholspeicher |
| 0x5C00 | - | 0x5FFF | 1024 Byte | |

**Tabelle 3.2:** EPROM-Speicherbereich der T-316

| | | | |
|---|---|---|---|
| 0x0000 | - | 0x1FFF | 8192 Byte |
| 0x2000 | - | 0x3FFF | 8192 Byte |

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.

Als Hauptprogramm dient das *Zentralprogramm ZP*, das mehrere betriebssystemähnliche Funktionen ausführt. Dazu zählen etwa die Datenbussteuerung, Initialisierung der CPU und I/O-Geräte nach dem Einschalten, Schlüssel- und Passwortkontrolle, Aufruf der Subroutinen und Erzeugung des Headers (Synchronfolge) für den ersten Verschlüsselungsvorgang. Der letzte Punkt unterscheidet sich grundlegend von den anderen Aufgaben des Programms. Es ist nicht ersichtlich, warum es sich im Hauptprogramm und nicht in einer Subroutine zur Verschlüsselung befindet. Es können zwei Schlüssel (Zeitschlüssel ZS-1 und ZS-2) gespeichert und durch ein Passwort gesichert werden.

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

**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 Verschlüsselungsroutine aus der Subroutine *Texteingabe* aufgerufen wird.

## 3.3 Algorithmus LAMBDA1

Der LAMBDA1-Algorithmus basiert auf dem Data Encryption Standard (DES). Bei DES handelt sich um eine Blockchiffre, die in den 1970er Jahren in den USA entwickelt und als Standard FIPS PUB 46 im Jahre 1977 veröffentlicht wurde. Dieser Standard war auch für eine zivile Verwendung vorgesehen und fand eine weite Verbreitung.

Vermutlich aus Gründen von Zeitdruck [20] 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.

Als Mode-of-Operation kann jede gültige Variante verwendet werden. Die technische Dokumentation [2] verweist auf den ISO-Standard „Betriebsarten für einen 64-bit-Block-Schlüsselalgorithmus“ (ISO 8372:1987). Bei der T-316 kam der Cipher Block Chaining Mode zum Einsatz. Als Padding für unvollständige Blöcke wurden Null-Bytes verwendet [4].

### 3.3.1 Data Encryption Standard

DES ist eine symmetrische Blockchiffre mit einer Blockgröße von 64 Bit und einer Schlüssellänge von 64 Bit. Jedoch werden nur 56 Bits des Schlüssels verwendet. Die restlichen acht Bit können als Paritätsbits für den Schlüssel verwendet werden. Grundlage des DES

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

Der erste Wert 58 bedeutet, dass das erste Bit des Ergebnisses das 58. des Eingangswertes ist. An die zweite Stelle gehört das 50. Bit. Alle Tabellen des DES verhalten sich nach diesem Schema. In der ersten Spalte der ersten Zeile ist das höchstwertige (most-significant) Bit und in der letzten Spalte in der letzten Zeile ist das niedrigstwertige (least-significant) Bit.

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:

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.

Die Funktion $f$ erweitert zu Beginn $R$ durch die Expansionsfunktion $E$ auf 48 Bit. Es werden dabei einfach Bits aus der ursprünglichen Menge mehrfach verwendet. Die Tabelle für $E$ und alle weiteren Permutationen von LAMBDA1 sind in [11] zu finden. Anschließend wird dieser Teil mit $K_n$ XOR-verknüpft. Die 48 Bit werden daraufhin in Blöcke zu je 6 Bit aufgeteilt. Das sind die Eingangswerte für die 8 verschiedenen S-Boxen. Diese reduzieren die 6 Bit auf 4 Bit. Jede der S-Boxen verwendet dafür eine eigene Funktion. Die daraus entstehenden Werte werden wieder zu einem 32-Bit-Block vereint, der als letzter Schritt einer Runde noch einmal durch die Permutation $P$ verändert wird.

Nachdem alle Runden abgeschlossen sind ($n = 16$), werden der L- und der R-Block wieder zusammengefügt. 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.

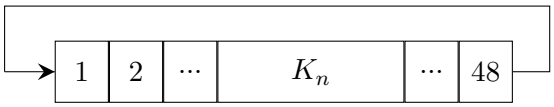



**Abbildung 3.2:** Schieberegister T

Direkt enthaltene Rundenschlüssel im Schlüssel von LAMBDA1:

$$
\begin{aligned}
K_1 &:= (S_1, S_2, ..., S_{48}) \\
K_2 &:= (S_{49}, S_{50}, ..., S_{96}) \\
K_3 &:= (S_{97}, S_{98}, ..., S_{144}) \\
K_4 &:= (S_{145}, S_{146}, ..., S_{192}) \\
K_{17} &:= (S_{193}, S_{194}, ..., S_{224}) \\
K_{18} &:= (S_{225}, S_{226}, ..., S_{256})
\end{aligned}
$$

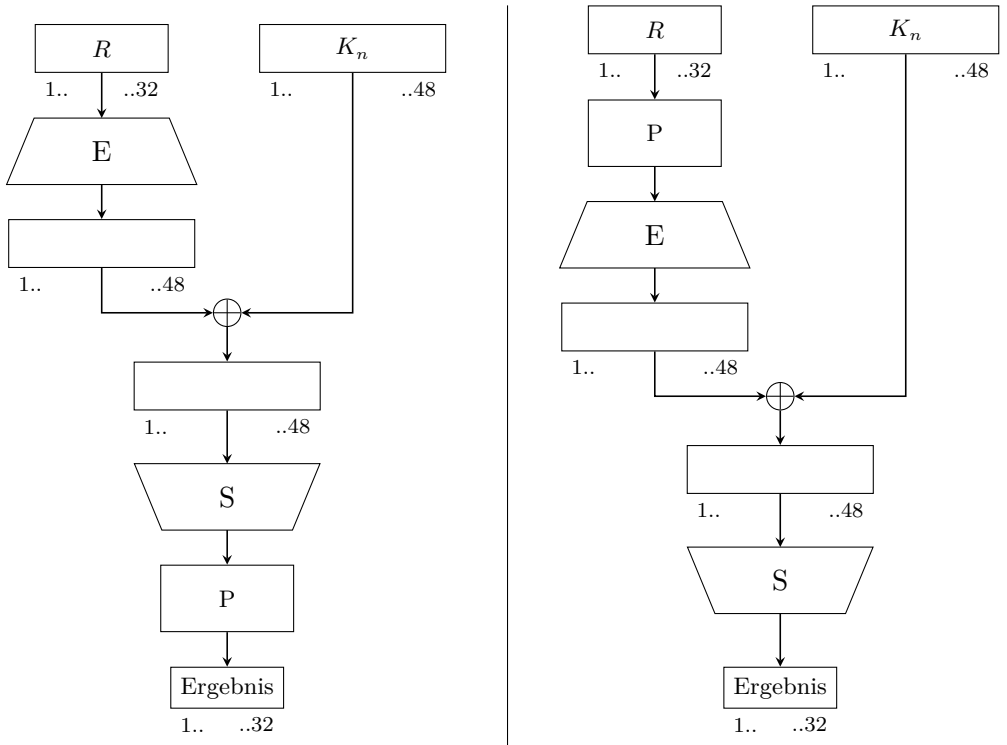
Die restlichen Schlüsselvektoren $K_5, ..., K_{12}$ und $K_{13}, ..., K_{16}$ werden über ein einfaches, linear rückgekoppeltes Schieberegister T erzeugt (siehe Abbildung 3.2). Dabei wird ein bereits bestehender 48-Bit-Rundeschlüssel $K_n$ um 11 Bit verschoben.

Die Notation $T^x(Y)$ steht für den Zustand des Schieberegisters T mit dem Startwert $Y$ nach $x$ Runden.
Die Notation $n \in \overline{x, y}$ steht für $\{n \in \mathbb{N} \,|\, x \leqq n \leqq y\}$.
Die Schlüsselableitung sieht dann wie folgt aus:

$$
\begin{aligned}
\forall n \in \overline{5, 12}: &\quad K_n := \mathrm{T}^{11}(K_{n-4}) \\
\forall n \in \overline{13, 16}: &\quad K_n := \mathrm{T}^{11}(K_{25-n})
\end{aligned}
$$

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.




**(a)** $f$ von DES **(b)** $f$ von LAMBDA1

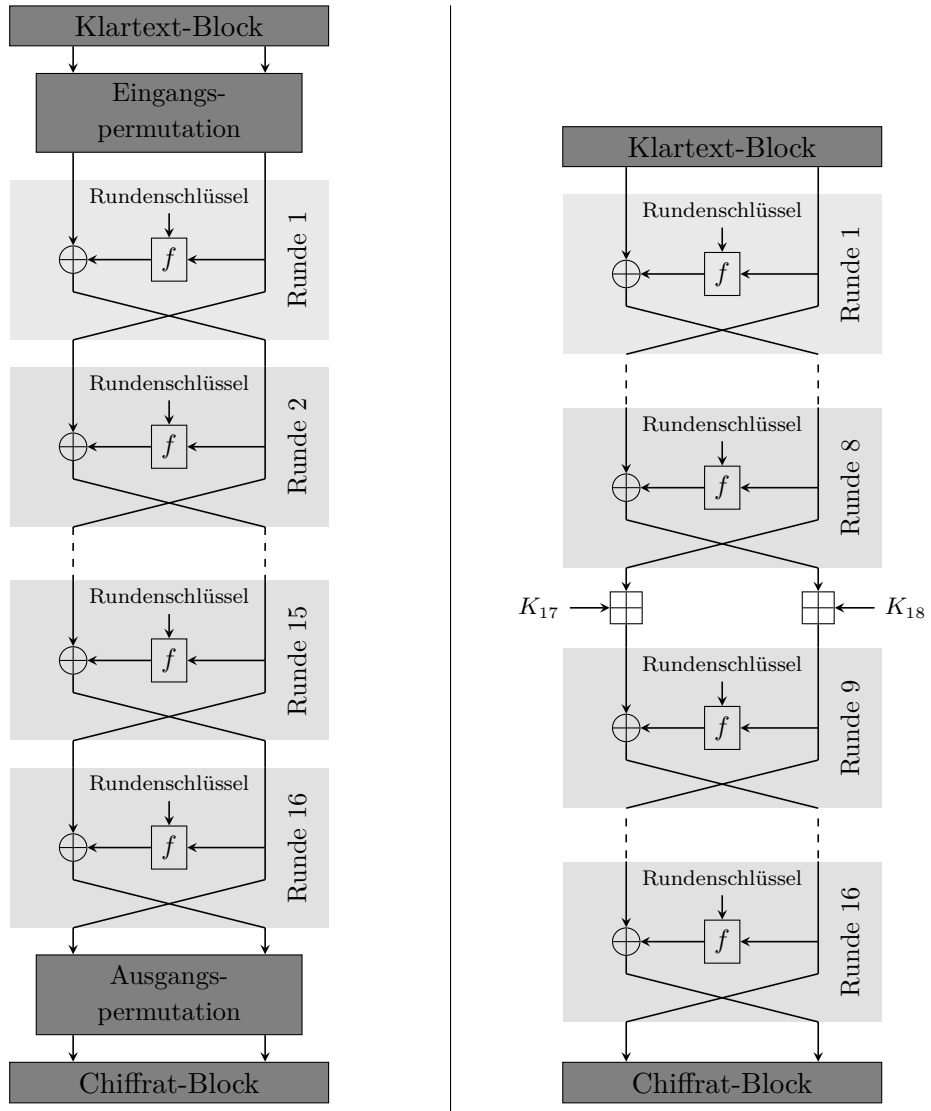
**Abbildung 3.3:** Vergleich zwischen der Funktion $f$ von LAMBDA1 und DES; der einzige Unterschied ist die Position der Permutation P.

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:

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

Die Ein- und Ausgangspermutationen $IP$ und $IP^{-1}$ wurden entfernt. Bei der Verknüpfung $\boxplus$ werden beide 32-Bit-Vektoren modular addiert.




**(a)** DES **(b)** LAMBDA1

**Abbildung 3.4:** Schematischer Vergleich zwischen den Verschlüsselungsalgorithmen von LAMBDA1 und DES nach [3]

Die Verknüpfung $\boxplus$ ist definiert als:

$$(X_1, ..., X_{32}), (Y_1, ..., Y_{32}) \mapsto X_1 * 2^{32-1} + X_2 * 2^{32-2} + ... + X_{32} * 2^0 + Y_1 * 2^{32-1} + Y_2 * 2^{32-2} + ... + Y_{32} * 2^0 \pmod{2^{32}}$$

Beim Entschlüsseln werden beim LAMBDA1-Algorithmus die Rundenschlüssel von hinten nach vorne angeordnet und die Vektoren $K'_{17}$ und $K'_{18}$ gebildet.

Neuanordnung der Schlüssel zum Entschlüsseln:

$$\begin{aligned}\forall n \in \overline{1,16}: \quad K'_n &:= K_{17-n} \\ K'_{17} &:= (0,0,...,0,1) \boxplus (K_{18} \oplus (1,1,...,1)) \\ K'_{18} &:= (0,0,...,0,1) \boxplus (K_{17} \oplus (1,1,...,1))\end{aligned}$$

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:

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

### 3.3.3 Sicherheit des Verfahrens

Die für die Sicherheit nicht relevanten Ein- und Ausgangspermutationen wurden entfernt. Durch die Vergrößerung des effektiven Schlüsselraums auf $2^{256}$ wurde ein großes Sicherheitsproblem des DES beseitigt. Ein Schlüssel lässt sich nicht mehr in realistischer Zeit mittels Durchsuchen des Schlüsselraums (brute-force) finden.

Courtois geht in [3] kurz darauf ein, dass zwar die Schlüssellänge ausreichend groß ist, aber durch die stark lineare Schlüsselauswahlfunktion Schwachstellen entstehen könnten. Einen weiteren sicherheitsmindernden Einfluss könnte die ineffiziente Nutzung des Schlüssels haben, weil große Teile davon nur in ein bis zwei Runden genutzt werden.

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 Chiffratblocks 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.

Während der Implementierung hat sich aber gezeigt, dass sich der bestehende Code wegen DES-spezifischer Bitstrukturen nicht so einfach auf die T-316 anwenden lässt. Es befinden sich noch Fehler in der Implementierung, die sie nicht kompatibel zu originalen Maschinen machen. Testvektor hierfür ist ein von Drobick [4] zur Verfügung gestelltes Paar aus einem Chiffrat samt Klartext, sowie dem dazugehörigen Schlüssel, das in Anhang B zu finden ist.

Die zugrundeliegende Struktur der Komponente ist fertig. Abbildung 3.5 zeigt das stark an die T-310 angelehnte Template. Für den Algorithmus ist nach Veröffentlichung dieser Arbeit noch eine Überarbeitung vorgesehen.

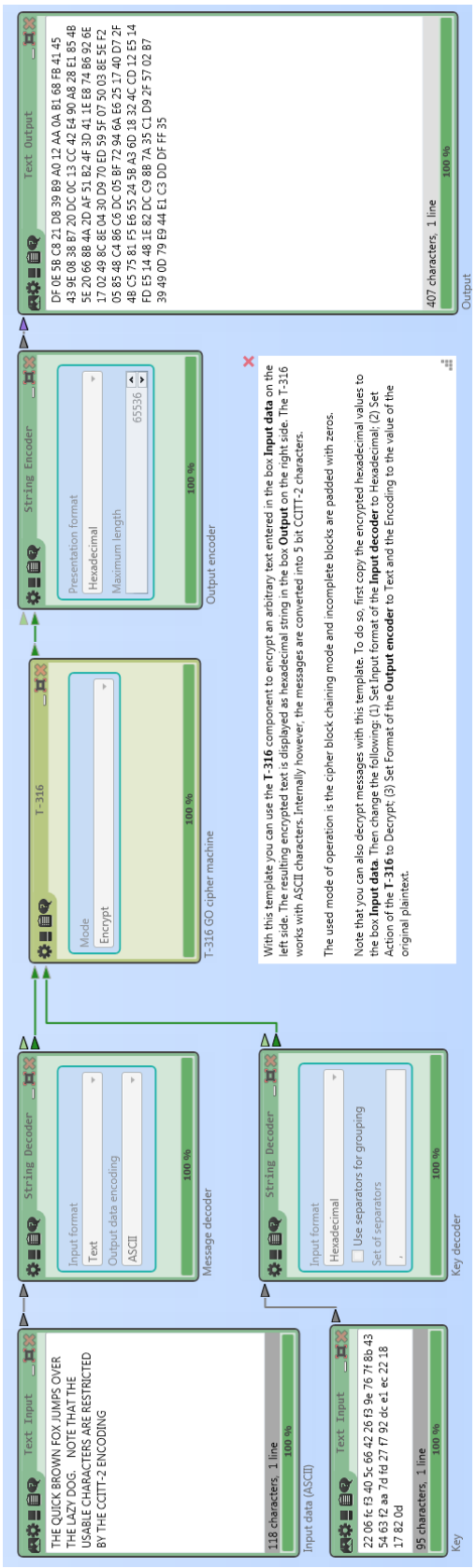



**Abbildung 3.5:** Template der neu hinzugefügten Komponente T-316 in CT2

# Kapitel 4

# Fazit und Ausblick

Diese Arbeit beleuchtete zum einen die Chiffriermaschine T-310/50, die bereits in [10] [14][3] untersucht wurde, genauer. Dazu stellte Herr Jörg Drobick vom NVA-Museum Harnekop rund 20 Original-Dokumente mit ca. 1250 Seiten zur Verfügung. Der Schwerpunkt lag dann auf der Implementierung in moderner Software (C#) und in der Integration in CrypTool 2. Wie sich dabei gezeigt hat, bestehen zwischen den derzeit vorhanden Implementierungen Unterschiede und verschiedene Ansätze. Mit der überarbeiteten Implementierung wurde versucht, diese Lücken zu schließen oder zu verkleinern, wobei dies durch zu große Unterschiede zwischen den Eigenschaften von logischen Schaltungen und moderner Software nicht immer gelang. Der Code der Komponente konnte jedoch wesentlich verbessert werden und es wurden einige neue Funktionen hinzugefügt. Auch Drobick versucht nach den Erkenntnissen aus der Kommunikation zu dieser Arbeit, seine Implementierung an die Originalmaschine anzupassen.

Eine weitere Möglichkeit, um die Funktionsweise der T-310 in Software nachzubilden, wäre die Verwendung von „Binary Decision Diagrams“. Die Schaltpläne könnten in ein solches Programm übertragen werden; zum Beispiel würde sich die Python-Bibliothek PyEda eignen[1]. Der Aufwand dafür wäre aber enorm, und es ist nicht sichergestellt, dass die dokumentierten Schaltpläne den tatsächlich verbauten Schaltungen entsprechen.

Zum zweiten wurde die Funktionsweise der T-316 untersucht und implementiert, wozu Herr Drobick sieben Dokumente mit ca. 500 Seiten zur Verfügung stellte. Die Implementierung aus dieser Arbeit weist noch Fehler auf, deren Behebung nach der Veröffentlichung geplant ist. Da es zur Sicherheit des LAMBDA1-Algorithmus nur eine kurze Einschätzung [3] gibt, wäre eine genauere Kryptoanalyse wünschenswert.

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.

Aus historischer Sicht gibt es einige interessante Punkte. Für die T-310 wurde mit russischen Mathematikern zusammengearbeitet, während die T-316 ausschließlich vom ZCO entwickelt wurde[4]. Dabei sind für den Vorgängeralgorithmus der T-310 nicht mehr alle Dokumente vorhanden. Auch einige Entwicklungsentscheidungen scheinen sich nicht komplett durch die Dokumente nachvollziehen zu lassen.

[1] Python EDA – Binary Decision Diagrams (https://pyeda.readthedocs.io/en/latest/bdd.html)

# 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

| | |
|---|---|
| T-310/50 | Chiffriermaschine zur 5-Bit-Fernschreibübertragung |
| T-310/51 | Chiffriermaschine zur 8-Bit-Datenchiffrierung |
| ARGON | Taktische Bezeichnung in der DDR für die T-310/50; wird in dieser Arbeit als Begriff für den Algorithmus verwendet |
| Grundgerät GG | Hauptkomponente der T-310 |
| Stromversorgungsgerät SV | Stromversorgungsgerät der T-310 |
| Bediengerät BT | Gerät zum Bedienen der T-310 |
| Zusatzbediengerät BTZ | Zusatzbediengerät, erfüllt gleiche Aufgaben wie das BT |
| | |
| Zentraleinheit | Logische Hauptkomponente des Grundgeräts |
| Chiffrator | Führt alle kryptographischen Operationen aus; Teil der Zentraleinheit |
| Zentralsteuerung | Steuerkomponente für z. B. Takt, Datenbusse und Betriebsmodi; Teil der Zentraleinheit |
| Kodeumsetzer | Kodewandler zwischen CCITT-2 und beliebigen Fernschreibzeichen |
| | |
| Funktionseinheiten FG | Teilkomponenten des Chiffrators |
| EE | Eingabeeinheit |
| KE | Komplizierungseinheit |
| VE | Verschlüsselungseinheit |
| SE | Synchronisationseinheit |
| PBE | Prüf- und Blockiereinheit |
| KEP | Prüfkomplizierungseinheit; Bestandteil der PBE |
| | |
| T-310-Blockchiffre | Begriff für den Algorithmus der Komplizierungseinheit |
| Wurmreihe D-W | Name der T-310-Blockchiffre in der Originaldokumentation |
| $\phi$ , $\varphi$ oder $\Phi$ | Weitere Bezeichnungen für die T-310-Blockchiffre |
| Z-Funktion | Boolesche Funktion der Blockchiffre |
| Rotationsfunktion von F | Funktion eines Schieberegisters für F; zu Beginn mit $F_0$ initialisiert |
| P-Funktion | Wählt mithilfe eines Wertes aus P des Langzeitschlüssels ein Bit aus U aus. |
| D-Funktion | Wählt mithilfe eines Wertes aus D des Langzeitschlüssels ein Bit aus U aus. |

Schlüssel:

| | |
|---|---|
| LZS | Langzeitschlüssel bestehend aus P, D und $\alpha$ |
| 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 |
| $\alpha$ | Teil des LZS |
| $S_0$ | Der 240 Bit lange Zeitschlüssel bestehend aus $S_1$ und $S_2$ |
| $S_1$ | Teil des Zeitschlüssels; 120 Bit lang |
| $S_2$ | Teil des Zeitschlüssels; 120 Bit lang |
| $F_0$ | 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 SRV2 und SRV3 |
| 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. |
| $U_0$ | 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 $a$ der Originaldokumentation |
| $F_n$ | Bitfolge der jeweiligen Runde, ursprünglich ausgehend vom Initialisierungsvektor $F_0$ |
| $K_i$ | Ein 5-Bit-Klartextzeichen |
| $B_i$ | Ein 5-Bit-Vektor aus $(a_7, ..., a_{11})$ |
| $H_i$ | Ein 5-Bit-Vektor aus $(a_1, ..., a_5)$ |
| $C_i$ | Ein 5-Bit-Zeichen des Chiffrats |
| $r_i$ | Anzahl der benötigten Runden der Register SRV2 und SRV3; ausgedrückt durch die Anzahl an Matrixmultiplikationen |
| $e$ | kleinste Exponent, sodass $H_i \cdot M^e = (1, 1, 1, 1, 1)$ |
| $i$ | 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 $F_n$ |
| SRU | 36-Bit Register U der KE |
| SRV1 | 5-Bit Empfangsschieberegister der VE (Zeichenempfang im Chiffrator) |
| 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

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

# Anhang B

# Testvektoren

## B.1 Testvektor für die T-310

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

Der Klartext und das Chiffrat lassen sich aus Abbildung B.1 ablesen. Der oberste Streifen enthält das Chiffrat. Gut erkennbar von Herrn Drobick eingezeichnet ist der Nachrichten-Header, der stets mit vier Zeichen B (0x19) beginnt. Darauf folgt der Initialisierungsvektor mit der Länge 25 und vier Zeichen K (0x0F). Der Header endet mit der „Maschinenbefehlsfolge (MBF)“, die zusätzlich von der T-310 eingefügt wird. Danach beginnen die verschlüsselten Zeichen des Klartexts.

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.

Der Zeitschlüssel lässt sich aus Abbildung B.2 ablesen. Eine Hilfe dazu ist die Tabelle, die Drobick unter [21] zur Verfügung stellt. Kurz zusammengefasst befinden sich die zehn Zeitschlüsselregister zu je 24 Bit in jeweils einer Spur (Reihe). Die oberste Vorzeichenspur, sowie 0 – 3 kodieren die Schlüsselregister von S1 (SRS1/1 – SRS1/5). Die Spuren 5 – 9 kodieren S2 (SRS2/1 – SRS2/5). Die Schlüsselbits befinden sich in jeder zweiten Spalte von 4 – 50. Es ist zu beachten, dass links das höchstwertige Bit und rechts das niedrigstwertige Bit ist. Beginnt man bei der Vorzeichenspur in Spalte 50 Richtung links zu lesen, erhält man die ersten Bits des Schlüssels S1. In Tabelle B.1 ist der Zeitschlüssel nochmals binär angeführt, um ihn leichter mit Abbildung B.2 vergleichen zu können.

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.

| Element | Wert |
|---|---|
| Langzeitschlüssel 21 | 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)<br>D: (0, 24, 36, 4, 16, 28, 12, 20, 32)<br>$\alpha$: 1 |
| Zeitschlüssel S1 | 17:77:d3:86:17:a3:18:3e:59:fc:23:88:1f:e1:0d2 |
| Zeitschlüssel S2 | 4c:4f:0d:19:eb:a3:ac:7a:7e:5d:01:fb:9e:1d:84 |
| Klartext (CCITT-2) | \LTRSABCDEFGHIJKLMNOPQRSTUVWXYZ\r\n\LTRS |
| Klartext | 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 |
| Nachrichten-Header | 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:08:0f:0f:0f:0f |
| Chiffrat | 0b:17:04:10:0b:11:14:01:1c:01:0e:1f:10:18:12:07:08:1e:1a:12:00:13:18:0b:17:19:09:16:07:0b:17:05 |
| Entschlüsselter Klartext | 19:19:19:19:0F:0F:0F:0F:1F:08:02:02:03:19:0E: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 |

Zeitschlüssel S1 (Binär)

| 1 | | | | | 24 |
|---|---|---|---|---|---|
| 0001 | 0111 | 0111 | 0111 | 1101 | 0011 |
| 25 | | | | | 48 |
| 1000 | 0110 | 0001 | 0111 | 1010 | 0011 |
| 49 | | | | | 72 |
| 0001 | 1000 | 0011 | 1110 | 0101 | 1001 |
| 73 | | | | | 96 |
| 1111 | 1100 | 0010 | 0011 | 1000 | 1000 |
| 97 | | | | | 120 |
| 0001 | 1111 | 1110 | 0001 | 1101 | 0010 |

Zeitschlüssel S2 (Binär)

| 1 | | | | | 24 |
|---|---|---|---|---|---|
| 0001 | 0111 | 0111 | 0111 | 1101 | 0011 |
| 25 | | | | | 48 |
| 1000 | 0110 | 0001 | 0111 | 1010 | 0011 |
| 49 | | | | | 72 |
| 0001 | 1000 | 0011 | 1110 | 0101 | 1001 |
| 73 | | | | | 96 |
| 1111 | 1100 | 0010 | 0011 | 1000 | 1000 |
| 97 | | | | | 120 |
| 0001 | 1111 | 1110 | 0001 | 1101 | 0010 |

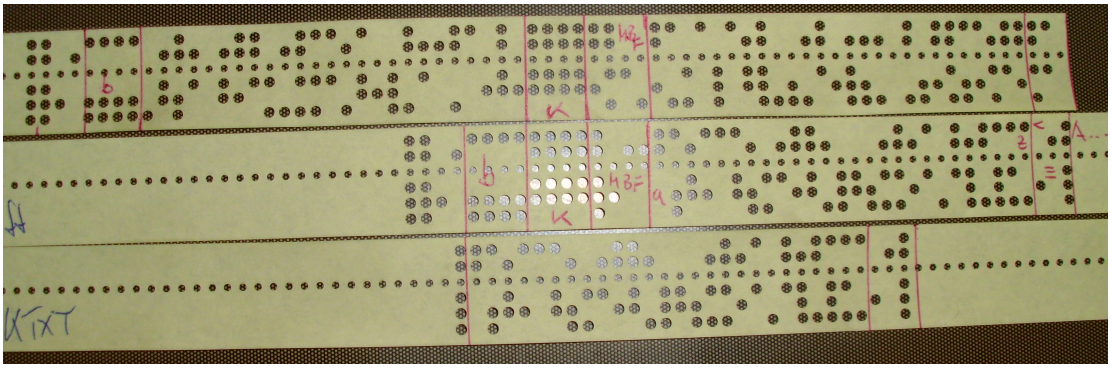


**Abbildung B.1:** Chiffrat (oben), Klartext (unten) und entschlüsselter Klartext (mittig) einer originalen T-310

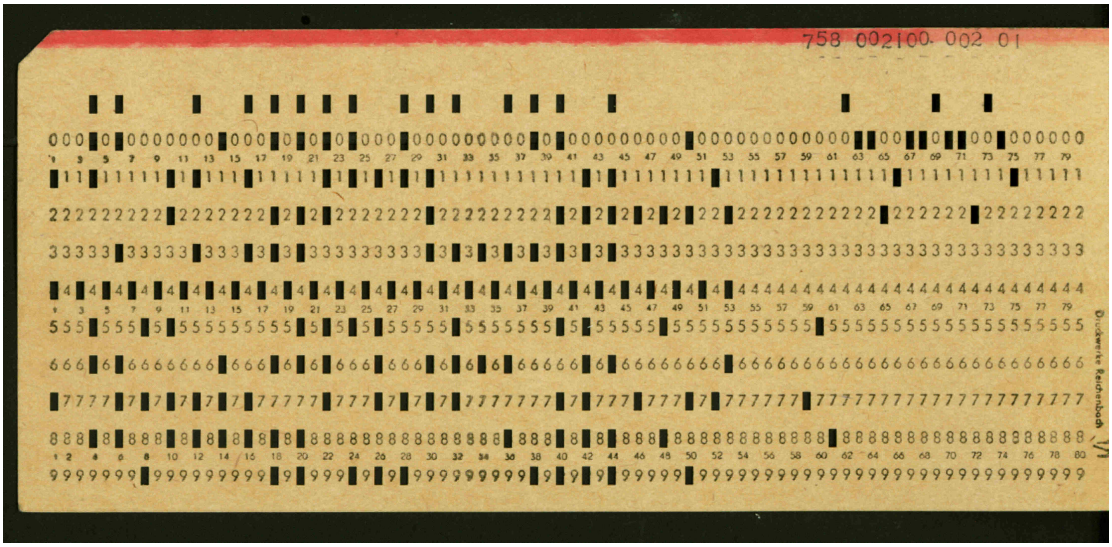



**Abbildung B.2:** Originaler Zeitschlüssel der T-310

## 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

| Element | Wert |
|---|---|
| Schlüssel (hexadezimal) | AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA |
| Nachricht (Text) | AAAAAAAA |
| Chiffrat (hexadezimal) | 2C 0A 7A 4A 1D 46 11 57 9A B5 0C 2F E0 97 07 2B C3 3F BB A4 |
| Nachricht (Text) | AAAAAAAA |
| Chiffrat (hexadezimal) | F9 D0 64 49 55 6C 66 B0 94 64 BE BC AF E7 AC DC 72 5B 4C 5E |

# Quellenverzeichnis

## Literatur


[1] „Beschreibung des Gerätesystems T310/50 - Schulungsmaterial (Folien)“ (1983). Zur Verfügung gestellt vom NVA-Museum Harnekop (siehe S. 9).

[2] „Beschreibung LAMBDA1. BStU MfS Abt. XI - 601“ (1990). Zur Verfügung gestellt vom NVA-Museum Harnekop (siehe S. 20, 23, 25).

[3] Nicolas Courtois, Jörg Drobick und Klaus Schmeh. „Feistel ciphers in East Germany in the communist era“. *Preprint zur Cryptologia* (2018), S. 1–18. eprint: https://doi.org/10.1080/01611194.2018.1428835. URL: https://doi.org/10.1080/01611194.2018.1428835 (siehe S. 2, 5, 13, 16, 28, 29, 32).

[4] Jörg Drobick. E-Mail Kommunikation. 2018 (siehe S. 4, 9, 17, 18, 21, 23, 30, 32, 37).

[5] „Gerätesystem T310/50 - Buch 1“ (1983). Zur Verfügung gestellt vom NVA-Museum Harnekop (siehe S. 4, 6, 8).

[6] „Gerätesystem T310/50 - Buch 2“ (1983). Zur Verfügung gestellt vom NVA-Museum Harnekop (siehe S. 7, 9).

[7] „Gerätesystem T316 - Technische Beschreibung. BStU MfS Abt. XI - 679“ (1990). Zur Verfügung gestellt vom NVA-Museum Harnekop (siehe S. 20, 22).

[8] „Kryptologische Analyse des Chiffriergerätes T310/50. BStU MfS Abt. XI - 594“ (1980). Zur Verfügung gestellt vom NVA-Museum Harnekop (siehe S. 9).

[9] J. H. Moore und G. J. Simmons. „Cycle Structure of the DES for Keys Having Palindromic (or Antipalindromic) Sequences of Round Keys“. *IEEE Transactions on Software Engineering* SE-13.2 (Feb. 1987), S. 262–273 (siehe S. 29).

[10] Klaus Schmeh. „The East German Encryption Machine T-310 and the Algorithm It Used“. *Cryptologia* 30.3 (2006), S. 251–257 (siehe S. 2, 9, 32).

[11] National Institute of Standards und Technology. „FIPS PUB 46-3 DATA ENCRYPTION STANDARD (DES)“. *PROCESSING STANDARDS PUBLICATION* (1999) (siehe S. 24, 25).

[12] „Verfahren ARGON (T310/50). BStU MfS Abt. XI - 504“ (1982). Zur Verfügung gestellt vom NVA-Museum Harnekop (siehe S. 9).

## Online-Quellen

[13] Om Bhallamudi. *Algebraic attack on the T-310*. URL: https://github.com/KillianDavitt/T-310-AlgebraicAttack (besucht am 18. 06. 2018) (siehe S. 16–18).

[14] Nicolas T Courtois u. a. *Cryptographic Security Analysis of T-310*. URL: https://eprint.iacr.org/2017/440.pdf (besucht am 29. 04. 2018) (siehe S. 2, 11, 16–18, 32).

[15] Jörg Drobick. *Chronologie der T-310*. URL: http://scz.bplaced.net/t310.html#chr (besucht am 30. 05. 2018) (siehe S. 2, 4).

[16] Jörg Drobick. *Der SAS- und Chiffrierdienst (SCD)*. URL: http://scz.bplaced.net (besucht am 13. 05. 2018) (siehe S. 2).

[17] Jörg Drobick. *Foto der T-316*. URL: http://scz.bplaced.net/t/t316koffer.jpg (besucht am 13. 05. 2018) (siehe S. 21).

[18] Jörg Drobick. *Historie des ZCO*. URL: http://scz.bplaced.net/zco.html#historie (besucht am 13. 05. 2018) (siehe S. 2).

[19] Jörg Drobick. *LAMBDA 64bit Blockchiffrierung*. URL: http://scz.bplaced.net/des.html#lambdaB (besucht am 05. 07. 2018) (siehe S. 29).

[20] Jörg Drobick. *Sachbestandsbericht LAMBDA1*. URL: http://scz.bplaced.net/des.html#lambda1 (besucht am 18. 05. 2018) (siehe S. 23).

[21] Jörg Drobick. *T-310 Schlüsselunterlagen*. URL: http://scz.bplaced.net/t310-schluessel.html (besucht am 29. 04. 2018) (siehe S. 7, 9, 37).

[22] Jörg Drobick. *T-310 Simulationssoftware*. URL: http://scz.bplaced.net/freeware.html (besucht am 29. 04. 2018) (siehe S. 17, 18).

[23] Jörg Drobick. *T-310/50 ARGON Postzulassung BStU*63*. URL: http://scz.bplaced.net/t310-post.html (besucht am 30. 05. 2018) (siehe S. 4).

[24] Jörg Drobick. *T-316 GO Kurzspruchgerät*. URL: http://scz.bplaced.net/t316.html (besucht am 13. 05. 2018) (siehe S. 2, 20).

[25] Jörg Drobick. *T310/51 SAGA*. URL: http://scz.bplaced.net/t310-51.html (besucht am 16. 06. 2018) (siehe S. 5).

[26] S. Przybylski u. a. *CrypTool 2 Plugin Developer Manual*. 2016. URL: https://www.cryptool.org/trac/CrypTool2/browser/trunk/Documentation/PluginHowTo/HowToDeveloper.pdf (besucht am 28. 06. 2018) (siehe S. 29).

[27] International Telecommunication Union. *CCITT Recommendation S.1(11/1988) - INTERNATIONAL TELEGRAPH ALPHABET NO. 2*. URL: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-S.1-198811-S!!PDF-E&type=items (besucht am 02. 07. 2018) (siehe S. 4, 20).

[28] Der Bundesbeauftragte für die Unterlagen des Staatssicherheitsdienstes der ehemaligen Deutschen Demokratischen Republik (BStU). *Aufgaben und Struktur*. URL: https://www.bstu.bund.de/DE/BundesbeauftragterUndBehoerde/AufgabenUndStruktur/_node.html (besucht am 30. 06. 2018) (siehe S. 1).

# Abbildungsverzeichnis

# Tabellenverzeichnis