**AEBC**

**Alican Encryption Block Cipher**

**EINE SICHERE BLOCKVERSCHLÜSSELUNG**

Autor:

Alican Karatasli

Betreuer:

Robert Ringhofer

Maturaarbeit Kantonsschule Zofingen

20.10.2023

Inhalt

[1. Vorwort 3](#_Toc148953583)

[2. Zusammenfassung 3](#_Toc148953584)

[3. Einleitung 4](#_Toc148953585)

[3.1 Vorbereitung / Einteilung in Feldern 4](#_Toc148953586)

[3.2 Ablauf Funktionen 5](#_Toc148953587)

[4. Funktionen 6](#_Toc148953588)

[4.1 S-Box 6](#_Toc148953589)

[4.2 Displacement 7](#_Toc148953590)

[4.3 Redisplacement 7](#_Toc148953591)

[4.4 Transpositionsverschiebung 8](#_Toc148953592)

[4.5 Retranspositionsverschiebung 9](#_Toc148953593)

[4.6 Key Infusion 10](#_Toc148953594)

[5. Entwicklung 10](#_Toc148953595)

[6. Kryptanalysis 11](#_Toc148953596)

[6.1 Kryptanalysis von der Displacement und Transpositionsverschiebung Funktion 11](#_Toc148953597)

[7. Literaturverzeichnis 13](#_Toc148953598)

[7.1 Literaturverzeichnis 13](#_Toc148953599)

# Vorwort

Kryptografie spielt eine entscheidende Rolle in der heutigen vernetzten Welt. Informationen müssen sicher gespeichert und sicher weitergeleitet werden. Dies ermöglicht uns die Kryptografie. Schon in der Antike mussten wichtige Befehle der Generäle vor dem Feind geheim gehalten werden, und so wurden verschiedene Verschlüsselungstechniken entwickelt. Doch die moderne Kryptografie hat sich in den letzten Jahrzehnten enorm weiterentwickelt.

Ich persönlich bin auf das Thema aufmerksam geworden durch persönliches Interesse in Informatik. Ich wollte eine Arbeit schreiben, indem ich meine Kenntnisse in der Informatik verbessern könnte und trotzdem vieles über die Kryptografie lernen würde.

In einer Welt, in der all unsere persönlichen Daten digital zugänglich sind, ist es von entscheidender Bedeutung, diese durch Kryptografie zu schützen und sicherzustellen, dass sie nicht in die falschen Hände geraten. Ich persönlich wollte ein Programm für mich entwickeln welches ich täglich gebrauchen könnte, um gewünschte Dateien zu verschlüsseln und so speichern könnte.

# Zusammenfassung

In dieser Arbeit wurde eine Blockverschlüsselung entwickelt mit dem Ziel Schlüsselkonzepte der modernen Kryptografie zu untersuchen und eigene Verfahren zu entwickeln mit dem Fokus auf Positionsumstellungsfunktionen.

Mit der Leitfrage: «Welche Schlüsselkonzepte der modernen Kryptografie sind wichtig zu berücksichtigen, wenn man ein neues Verschlüsselungsverfahren entwickelt?», begann die Recherche, um die Grundlagen der Kryptografie zu untersuchen und eine solide Grundlage für die Entwicklung eines neuen Verschlüsselungsverfahrens zu schaffen.

Diese Blockverschlüsselung wurde danach in Python programmiert, um Benutzern zu ermöglichen ihre eigenen Eingaben automatisch mit AEBC zu verschlüsseln. Alle 4 selbstentwickelte Funktionen (Substitutionsbox, Displacement, Transpositionsverschiebung, Key-Infusion) werden in dieser Arbeit vorgeführt. Darüber hinaus wird erläutert, welche Überlegungen und Schritte bei der Entwicklung dieses Verschlüsselungsverfahrens angestellt wurden.

AEBC ist so konzipiert, dass es sich nahtlos in verschiedene alltägliche Situationen integrieren lässt, einschließlich Netzwerkkommunikation und den Austausch von Nachrichten. Sie ist ausserdem so angefertigt, dass man AEBC leicht weiterentwickeln kann und den persönlichen Anforderungen anpassen kann. Dies beinhaltet die Möglichkeit, die Sicherheit signifikant zu erhöhen, selbst wenn dies mit einer erheblichen Effizienzreduktion einhergeht, oder alternativ, die Effizienz zu steigern, gegebenenfalls auf Kosten der Sicherheit.

Formularbeginn

# Einleitung

## 3.1 Vorbereitung / Einteilung in Feldern

In iterativen Blockverschlüsselungen werden die Elemente der Eingabe meist in separate Abschnitte eingeteilt und in den respektiven Abschnitten operiert. In AEBC heissen diese Abschnitte Felder.

Der erste Schritt in der AEBC-Verschlüsselung beginnt mit der Umwandlung der Eingabe ins Binär. Jedes Zeichen ist 1 Byte und 1 Byte ist 8 Bits. Alle Bytes in der Eingabe werden der Reihe nach in diese Felder unterteilt um ein einfaches operieren für eine grosse Eingabe zu gewährleisten.

|  |  |
| --- | --- |
| 1 Subfeld | 9 Bytes |
| 1 Feld | 81 Bytes |
| 1 Upfeld | 729 Bytes |
| 1 Superfeld | 6'561 Bytes |
| 1 Departement | 59'049 Bytes |
| 1 Dimension | 531'441 Bytes |
| 1 Updimension | 4'782'969 Bytes |
| 1 Realm | 43'046'721 Bytes |
| 1 STRCH | 387’420'489 Bytes |
| 1 EXT | 3'486'784'401 Bytes |
| 1 EXP | 31’381’059’609 Bytes |
| 1 ADT | 282’429’536’4811 Bytes |

*Tabelle 1*

In der Abbildung 1 ist zu sehen, wie viele Bytes es in den einzelnen Feldern Platz hat.

Falls es weniger Bytes in der Eingabe vorhanden sind als die erwünschte Anzahl an Bytes um eines der folgenden Felder aufzufüllen, so wird der Rest mit zufälligen Zeichen ergänzt. Dies ist ein Code-Abschnitt zuständig um eine Eingabe kleiner als 81 Bytes aufzufüllen auf 1 Feld mit zufälligen Variablen.

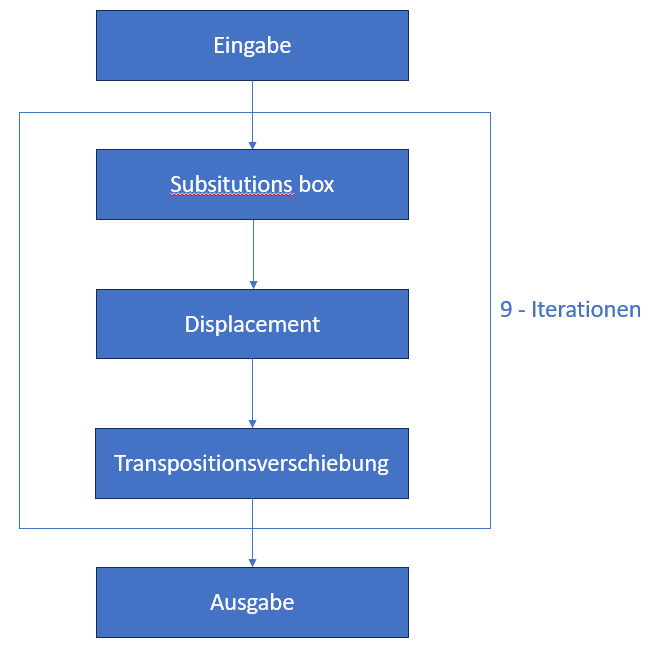
if len(e) != 81 and len(e)>0:  
 r = True  
 for i in range(81 - len(e)):  
 zufzhl = format(random.randint(129, 255),'08b')  
 e.append(zufzhl)  
 z = 81

Als Beispiel: Eine 24-stündige 720p HD-Videoaufnahme wäre 26’928’000'000 Bytes gross und eine durchschnittliche Textnachricht 1'280 Bytes. Dies bedeutet, dass

In der Praxis sind moderne Verschlüsselungsalgorithmen ihre maximale Eingabewerte wie bei AES viel kleiner. Das Ziel ist dabei nicht ein unnötig Grosses Algorithmus zu erstellen da es die Leistung und die Speicheranforderungen beeinflussen kann. Man sollte probieren ein perfektes Optimum zwischen Effizienz und Sicherheit zu finden. Diese Aussage gilt nicht nur für die maximale Eingabewert sondern auch für alle anderen Aspekte eines Verschlüsselungsalgorithmus.

## 3.2 Ablauf Funktionen

Sobald die Eingabe ins Binär umgewandelt worden ist und in die Felder eingeteilt ist, finden die Iterationen statt. Es wurden 4 Funktionen entwickelt, welche zusammen je eine Iteration bilden. Es werden insgesamt 9 Iterationen durchgeführt.



*Abbildung 1*

Als erstes werden alle Elemente mittels einem Substitutionsbox ersetzt. Das bedeutet, dass jeder Eingabewert in einen anderen Wert umgewandelt wird, wobei es wichtig zu beachten ist, dass derselbe Eingabewert niemals zweimal denselben Ausgabewert erzeugt. Dies ist ein Verfahren welches in fast allen modernen Verschlüsselungen zu nutzen kommt.

Danach müssen die Positionen der Elemente in den Subfeldern mittels der Displacement Funktion neuumgeordnet werden. Danach werden die Elemente im ganzen Feld verstreut mittels der Transpositionsverschiebung Funktion. Schlussendlich werden alle Elemente mit den respektiven Subschlüsseln mittels dem XOR-Verfahren zusammengesetzt.

# Funktionen

**AEBC ist eine iterative Blockverschlüsselung. Eine iterative Blockverschlüsselung führt in jeder Iteration bestimmte Funktionen aus. Im Falle von AEBC sind diese: Substitution-Box, Displacement, Transpositionsverschiebung und Key Infusion.**

## 4.1 S-Box

Eine Substitution-Box, auch S-Box genannt, wird verwendet, um Daten in einem Verschlüsselungsalgorithmus zu ersetzen. Für jeden möglichen Eingabewert gibt die S-Box einen entsprechenden Ausgabewert zurück. Das Ziel einer S-Box in der Kryptografie ist es, die Ersetzung von Daten auf eine nicht-lineare Weise durchzuführen und der Verschlüsselung einen gewissen Grad an algebraischer Komplexität zu verleihen. Eine gut entworfene S-Box sollte widerstandsfähig gegen verschiedene Angriffstechniken wie differentielle Kryptanalyse oder lineare Kryptanalyse sein. Eine weitere Methode, um die Sicherheit zu erhöhen, besteht darin, den Eingabewert in mehrere Segmente aufzuteilen und diese dann jeweils mit verschiedenen S-Boxen zu substituieren.

Ein Beispiel für eine S-Box sieht wie folgt aus:

## 4.2 Displacement

Die Displacement Funktion vertauscht die Positionen der Elemente der einzelnen Sub-Felder abhängig vom Sub-Schlüssel. Die Reihenfolge der Elemente im Sub-Feld wird neu geordnet und ein neues Sub-Feld wird dadurch erstellt. Diese Anordnungen sind komplett zufällig erstellt worden und es bestehen insgesamt 256 verschiedene Displacement Anordnungen. Der Grund, warum es 256 Anordnungen gibt, liegt daran weil es 256 8 Bit grosse Zahlen gibt. Jede 8 Bit grosse Subschlüssel wurde einer Ordnungsreihenfolge festgelegt. Diese Funktion vertauscht die Position aller Elemente in allen Subfelder. Um die Sicherheit von dieser Funktion zu erhöhen könnte man in jeder Iteration nicht nur die Positionen der Elemente der Subfelder vertauschen, sondern auch alle Felder, Upfelder, etc. was jedoch die Effizienz stark reduzieren würde.

Ein Beispiel für so eine Displacement Funktion sieht wie folgt aus:

if key1 == "10100101":  
 new\_SUBF0 = [SUBF0[5], SUBF0[7], SUBF0[2], SUBF0[3], SUBF0[8], SUBF0[6], SUBF0[1], SUBF0[0], SUBF0[4]]

Ein normales Subfeld hätte die Form:

SUBFX = [SUBFX[0], SUBFX[1], SUBFX[2], SUBFX[3], SUBFX[4], SUBFX[5], SUBFX[6], SUBFX[7], SUBFX[8]]

Um die Positionierung der Elemente zu ändern, muss man den Index von den Elementen zu dem Index von der erwünschten Reihenfolge umschreiben.

## 4.3 Redisplacement

Die Redisplacement ist eine Funktion welches bei der Rückverschlüsselung gebraucht wird. Dabei wird der erwünschte Index des originalen Subfeldes umgeschrieben.

Als Beispiel betrachten wir das vorherige Beispiel aus 4.2 Displacement:

if key1 == "10100101":  
 new\_SUBF0 = [SUBF0[5], SUBF0[7], SUBF0[2], SUBF0[3], SUBF0[8], SUBF0[6], SUBF0[1], SUBF0[0], SUBF0[4]]

Die Liste wird völlig durcheinandergebracht. SUBF0[0] wird hier neu definiert als new\_SUBF0[7], SUBF0[1] wird neu definiert als new\_SUBF0[6], Um dies

if key1 == "10100101":  
 decoded\_SUBF0 = [new\_SUBF0[7], new\_SUBF0[6], new\_SUBF0[2], new\_SUBF0[3], new\_SUBF0[8], new\_SUBF0[0], new\_SUBF0[5], new\_SUBF0[1], new\_SUBF0[4]]

## 4.4 Transpositionsverschiebung

Die Transpositionsverschiebung wurde speziell entworfen, um in Kombination mit der Displacement-Funktion eine effektive und starke Verschlüsselung zu ermöglichen. Da die Displacement-Funktion nur in den respektiven Feldern operiert, könnte es für einen Angreifer einfach sein, diese mittels kryptanalytischer Verfahren in Segmente zu unterteilen und zu entschlüsseln. Durch die Transpositionsverschiebung wird die Verschlüsselung komplexer und der Avalanche Effekt wird verstärkt.

Bei der Transpositionsverschiebung handelt es sich um, wie bei der Displacement Funktion, um eine Positionsumstellungsfunktion. Fast alle Elemente in einem Feld werden neuumgeordnet. Es gibt insgesamt wie bei der Displacement Funktion insgesamt 256 verschiedene Anordnunsmöglichkeiten. Alle diese 256 Anordnungsmöglichkeiten wurden zufällig generiert.

Ein Beispiel sieht wie folgt aus:

if key1 == "00000000": new\_FX = [SUBF1[5], SUBF8[1], SUBF5[5], SUBF7[4], SUBF6[6], SUBF8[7], SUBF6[5], SUBF2[1],  
 SUBF5[3], SUBF4[2], SUBF4[7], SUBF8[4], SUBF7[1], SUBF5[4], SUBF6[3], SUBF0[0],  
 SUBF3[5], SUBF3[3], SUBF2[3], SUBF5[0], SUBF1[0], SUBF7[5], SUBF4[8], SUBF0[8],  
 SUBF0[6], SUBF0[1], SUBF3[2], SUBF8[2], SUBF1[7], SUBF3[4], SUBF4[6], SUBF7[2],  
 SUBF2[0], SUBF3[7], SUBF3[1], SUBF8[5], SUBF6[8], SUBF2[8], SUBF1[4], SUBF0[7],  
 SUBF0[4], SUBF6[0], SUBF7[8], SUBF2[2], SUBF1[3], SUBF5[6], SUBF5[8], SUBF1[2],  
 SUBF8[3], SUBF4[0], SUBF2[7], SUBF8[0], SUBF0[5], SUBF1[6], SUBF7[3], SUBF7[7],  
 SUBF5[7], SUBF5[1], SUBF6[2], SUBF6[4], SUBF1[1], SUBF6[7], SUBF3[8], SUBF1[8],  
 SUBF3[6], SUBF6[1], SUBF4[1], SUBF0[3], SUBF4[5], SUBF2[4], SUBF5[2], SUBF2[5],  
 SUBF7[6], SUBF0[2], SUBF4[4], SUBF2[6], SUBF7[0], SUBF8[6], SUBF8[8], SUBF3[0],  
 SUBF4[3]]

Es ist zu sehen, wie die Transpositionsverschiebung-Funktion ähnlich strukturiert ist wie die Displacement-Funktion. Jedoch sind in diesem Fall 81 Elemente in der neu definierten Liste (new\_F0). Alle Werte werden wie bei der Displacement Funktion mit dem Index von den vorherigen Subfeldern neuangeordnet.

## 4.5 Retranspositionsverschiebung

Das gleiche Prinzip wie bei der Redisplacement-Funktion gilt auch für die Retranspositionsverschiebungs-Funktion. Dabei wird der erwünschte Index des originalen Feldes umgeschrieben.

Als Beispiel betrachten wir die Retranspositionsverschiebung vom Beispiel aus 4.4 Transpositionsverschiebung:

if key1 == "00000000": decoded\_F0 = [new\_F0[15], new\_F0[25], new\_F0[73], new\_F0[67], new\_F0[40], new\_F0[52],  
 new\_F0[24], new\_F0[39], new\_F0[23], new\_F0[20], new\_F0[60], new\_F0[47],  
 new\_F0[44], new\_F0[38], new\_F0[0], new\_F0[53], new\_F0[28], new\_F0[63],  
 new\_F0[32], new\_F0[7], new\_F0[43], new\_F0[18], new\_F0[69], new\_F0[71],  
 new\_F0[75], new\_F0[50], new\_F0[37], new\_F0[79], new\_F0[34], new\_F0[26],  
 new\_F0[17], new\_F0[29], new\_F0[16], new\_F0[64], new\_F0[33], new\_F0[62],  
 new\_F0[49], new\_F0[66], new\_F0[9], new\_F0[80], new\_F0[74], new\_F0[68],  
 new\_F0[30], new\_F0[10], new\_F0[22], new\_F0[19], new\_F0[57], new\_F0[70],  
 new\_F0[8], new\_F0[13], new\_F0[2], new\_F0[45], new\_F0[56], new\_F0[46],  
 new\_F0[41], new\_F0[65], new\_F0[58], new\_F0[14], new\_F0[59], new\_F0[6],  
 new\_F0[4], new\_F0[61], new\_F0[36], new\_F0[76], new\_F0[12], new\_F0[31],  
 new\_F0[54], new\_F0[3], new\_F0[21], new\_F0[72], new\_F0[55], new\_F0[42],  
 new\_F0[51], new\_F0[1], new\_F0[27], new\_F0[48], new\_F0[11], new\_F0[35],  
 new\_F0[77], new\_F0[5], new\_F0[78]]

Um einen Effizienteren Programm zu schreiben, wird darauf verzichtet die erzeugten Listen nochmals in die Felder zu unterteilen und darum sind hier keine Subfelder ersichtlich sondern nur die Indexen der Elemente von der new\_FX welches von der Transpositionsverschiebung-Funktion erzeugt wurde.

## 4.6 Key Infusion

Bei der Key Infusion wird jedes Subfeld mit einem der Subschlüssel mittels der XOR-Funktion gegenübergestellt. Der Gedanke dabei ist die Eingabe möglichst viel zu verändern.

… es kommt eine XOR Tabelle und paar Sätze…

# Entwicklung

Als erstes wurden zahlreiche Blockverschlüsselungsverfahren sowie die wesentlichen Kriterien, die bei der Entwicklung eines solchen Verfahrens zu berücksichtigen sind, sorgfältig analysiert. Durch diese umfassende Recherche konnten bereits einzelne Schlüsselkonzepte erkannt werden. Einige dieser Schlüsselkonzepte sind: Alle Blockverschlüsselungsverfahren benutzen S-Boxe, alle haben eine oder mehrere Funktionen, in welchen die Positionen der einzelnen Elemente möglichst komplex vertauscht werden und alle spalten den Hauptschlüssel in kleinere Subschlüssel und setzen die einzelnen Elemente mit den Subschlüsseln zusammen mittels der XOR-Funktion.

Die Idee der Felder Generierung war aufgekommen als viele der Funktionsideen auf Excel farbig aneinander geordnet waren. Mit der Weiterentwicklung dieser Idee entstanden rasch Konzepte für Subfelder, Upfelder und vieles mehr, welche in der Abbildung 1 ersichtlich sind.

Das Motiv bei der Entwicklung der Displacement Funktion war es, mit einer Funktion aufzukommen welches in den einzelnen Subfelder die Positionen vertauschen würde. Jedoch wurde nach kurzer Zeit erkennbar, dass dies nicht ausreichen würde. Darum musste eine neue Funktion erarbeitet werden welches unter den Subfeldern Elemente tauschen würde, um eine weitere Komplexion in der Verschlüsselung einzubauen.

Zuerst gab es jedoch nur 2 Szenarien für die Displacement Funktion und 2 Szenarien für die Transpositionsverschiebung Funktion, was nicht genug sicher gewesen wäre.   
Die Idee bestand darin, den Subschlüssel für das jeweilige Feld darauf zu prüfen, ob es sich um eine durch 2 teilbare oder nicht teilbare Zahl handelte. Dies war der Fall für beide Funktionen.

Die Anzahl an Szenarien musste erhöht werden, um eine höhere Sicherheit zu gewährleisten. Dafür wurde das Kriterium von durch zwei Teilbar oder nicht durch zwei Teilbar umgeändert zu welche 8 Bit Zahl der Subschlüssel ist. Für jede mögliche 8 Bit grosse Subschlüssel wurde eine Index Ordnung angeordnet. Jedoch würde das bedeuten, dass es für einen Feld, nach der Iteration, nur 4 verschiedene Positionierungsmöglichkeiten geben würde.

Um dies zu verbessern, wurde für jede 8 Bit grosse Zahl eine Anordnungsreihenfolge festgelegt. Da es 256 8 Bit grosse Zahlen hat, mussten 256 verschieden Anordnungsreihenfolgen erstellt werden.

# Kryptanalysis

Bei der Kryptanalysis handelt es sich um die Analyse über die Sicherheit des Verschlüsselungsverfahrens.

## 6.1 Kryptanalysis von der Displacement und Transpositionsverschiebung Funktion

Es gibt insgesamt wie im Kapitel **4.2 Displacement** und **4.3 Transpositionsverschiebung** aufgezeigt 256 verschiedene Anordnungsmöglichkeiten für die beiden Funktionen. Dies bedeutet, dass es in einem Feld (9 Subfelder) insgesamt 256^9 Anordnungsmöglichkeiten entstehen wegen der Displacement Funktion. Hinzu kommt noch 256 Anordnungsmöglichkeiten von der Transpositionsverschiebung Funktion. Insgesamt sind das (256^9) \* 256 Verschiedene Anordnungsmöglichkeiten für je ein Feld pro Iteration.

Formel für die Bestimmung an Anzahl Möglichkeiten pro Iteration ist wie folgt:

*Formel 1*

|  |  |
| --- | --- |
| Anzahl Iterationen | Anzahl Möglichkeiten |
| 1 | (256^9) \*256 |
| 2 | ((256^9) \*256) \* ((256^9) \*256) |
| 3 | ((256^9) \*256) \* ((256^9) \*256) \* ((256^9) \*256) |
| 4 | ((256^9) \*256) \* ((256^9) \*256) \* ((256^9) \*256) \* ((256^9) \*256) |
| 5 | ((256^9) \*256) \* ((256^9) \*256) \* ((256^9) \*256) \* ((256^9) \*256) \* ((256^9) \*256) |
| 6 | 3.1217 \* 10^144 |
| 7 | 3.774 \* 10^168 |
| 8 | 4.5624 \* 10^192 |
| 9 | 5.5157 \* 10^216 |
| 10 | 6.668 \* 10^240 |

*Tabelle 2*

In der Tabelle 2 ist zu sehen, wie viele Anordnungsmöglichkeiten pro steigende Iterationen zu entstehen sind.

Pro steigende Iteration gewinnt die Verschlüsselung an Sicherheit, verliert jedoch viel an Effizienz.

Ein Bild, das Text, Reihe, Diagramm, Screenshot enthält.

Automatisch generierte Beschreibung *Graph 1*

Die Formel 1 ist eine exponentielle Funktion welches wie Im Graph 1 zu sehen ist, dass die Anzahl an Möglichkeiten Exponentiell wächst. Es wurde eine Log-Log-Skala erstellt da ansonsten der Wachstum nicht klar ersichtlich sein würde, weil der Wert exponentiell so schnell wächst, dass die anderen Werte vor dem letzten Wert nicht ersichtlich sind.

# Literaturverzeichnis

## 7.1 Literaturverzeichnis

The Design of Rijndael, Joan Daemen und Vincent Rijmen

AES Proposal: Rijndael, Joan Daemen und Vincent Rijmen

Applied Cryptanalysis, Mark Stamp und Richard M. Low

Algorithmic Cryptanalysis, Antoine Joux

Advanced Linear Cryptanalysis of Block and Stream Ciphers, Pascal Junod und Anne Canteaut

Cryptography, Simon Rubinstein-Salzedo

Lai-Massey Cipher Designs, Jorge Nakahara Jr.

Modern Cryptanalysis, Christopher Swenson

Cryptanalysis of Block Ciphers, Jiqiang Lu

Techniques for Block Cipher Cryptanalysis, Yunwen Liu