

Reverse-Engineering der Transposition-Chiffre

Tamer, Yusuf
78159@studmail.htw-aalen.de

June 15, 2025

Abstract

Dieses Paper befasst sich mit der Transposition-Chiffre, bei der es um ein symmetrisches Kryptosystem handelt.

Einleitung

Moderne sowohl auch Chiffren aus der Antike basieren auf das Transposition-Verfahren. Dabei handelt es sich um eine Permutation mit der die einzelnen Zeichen vertauscht werden, sodass der eigentliche Klartext damit verborgen wird. Dieses Verfahren ist Zweidimensional, weshalb die einzelnen Zeichen auf einen 2D-Feld verteilt wird. Beispielsweise soll der folgende Klartext:

ASIMPLETRANSPPOSITION mit der Schlüssellänge 5 verschlüsselt werden.

A	S	I	M	P
L	E	T	R	A
N	S	P	O	S
I	T	I	O	N

« 4	« 3	« 2	« 1	
N	O	I	T	I
S	O	P	S	N
A	R	T	E	L
P	M	I	S	A

N	M	T	S	I
S	O	I	E	N
A	O	I	S	L
P	R	P	T	A

Programmierung der Transposition-Chiffre in C++

Formel

Mit folgenden C-Code lässt sich die Verschlüsselung sowohl auch die Entschlüsselung der Transposition-Chiffre implementieren:

```
for (int i = 0; i < plainTextSize; i++)
{
    this->cipherField[i % this->key]
    [i % (this->size/this->key)] = plainTextField[i];
}
```

Klasse

```
class transpositionCipher
{
public:
    int key;
    int size;
    char * plainField;
    char ** cipherField;
    char * decryptedField;
    transpositionCipher(int a, int b);
    void readPlainField();
    void setPlainField(char *buf);
    void encrypt();
    void decrypt();
    void readCipherField();
}
```

Das Hauptprogramm

Das Hauptprogramm lässt sich folgendermaßen implementieren:

```
int main()
{
    transpositionCipher tt(5, 20); // Übergabe vom Plaintext
    char buffer[] = { 'A', 'S', 'I', 'M', 'P', 'L', 'E', 'T', 'R', 'A', 'N', 'S', 'P', 'O', 'S', 'I', 'T', 'I', 'O', 'N' };
    tt.setPlainField(buffer);
    tt.encrypt(); // Verschlüsselungs- und Entschlüsselungsfunktion
    tt.decrypt();
    tt.printDecryptedField(); // Ausgabe vom Ciphertext
}
```

Assembler-Code vom Hauptprogramm

```

00401000: endbr64
00401001: push rbp
00401002: mov rbp, rsp
00401003: sub rsp, 0x40
00401004: mov rax, QWORD PTR fs:0x28
00401005: mov QWORD PTR [rbp-0x8], rax
00401006: xor eax, eax
00401007: lea rax, [rbp-0x40]
00401008: mov edx, 0x14
00401009: mov esi, 0x5
0040100a: mov rdi, rax
0040100b: call 00401010 <Zn21CC2E1>
0040100c: movabs rax, 0x54454c504d495341

```

Vorbereitung der Parameter für den Konstruktoraufbau

Aufruf des Konstruktors

Anhand des Konstruktors wird die Größe vom CipherField und ebenso auch die KeySize festgelegt. In diesem Fall enthält das CipherField 20 Zeichen. Die KeySize beträgt 5 und damit passen in die vertikale Zeile dementsprechend nur 5 Zeichen. Daher wird der Plaintext einer Funktion übergeben, um die Verschlüsselungs-, Entschlüsselungs- und auch die Ausgabefunktion nacheinander aufzurufen.

```

00401010: call 00401010 <Zn21CC2E1>
00401011: lea rax, [rbp-0x40]
00401012: mov rdi, rax
00401013: call 00401010 <Zn21CC2E1>
00401014: lea rax, [rbp-0x40]
00401015: mov rdi, rax
00401016: call 00401010 <Zn21CC2E1>
00401017: call 00401010 <Zn21CC2E1>
00401018: lea rax, [rbp-0x40]
00401019: mov rdi, rax
0040101a: call 00401010 <Zn21CC2E1>
0040101b: mov rax, 0x0
0040101c: mov rax, QWORD PTR [rbp-0x0]
0040101d: sub rax, QWORD PTR fs:0x28
0040101e: je 0040101f
0040101f: call 00401010 <Zn21CC2E1>

```

Eingabe vom Startwert

Aufruf der Verschlüsselungsfunktion

Ausgabe der Verschlüsselungsnachricht

Chronologischer Ablauf der einzelnen Funktionen, mit der die Transposition-Chiffre implementiert werden kann.

1. transpositionCipher(CipherField, KeySize)
2. setPlainField(buffer)
3. encrypt()
4. decrypt()
5. printEncryption()