**Hochschule RheinMain**
Fachbereich Design Informatik Medien
Studiengang Angewandte Informatik / Informatik Technische Systeme
Prof. Dr. Bernhard Geib

# Security

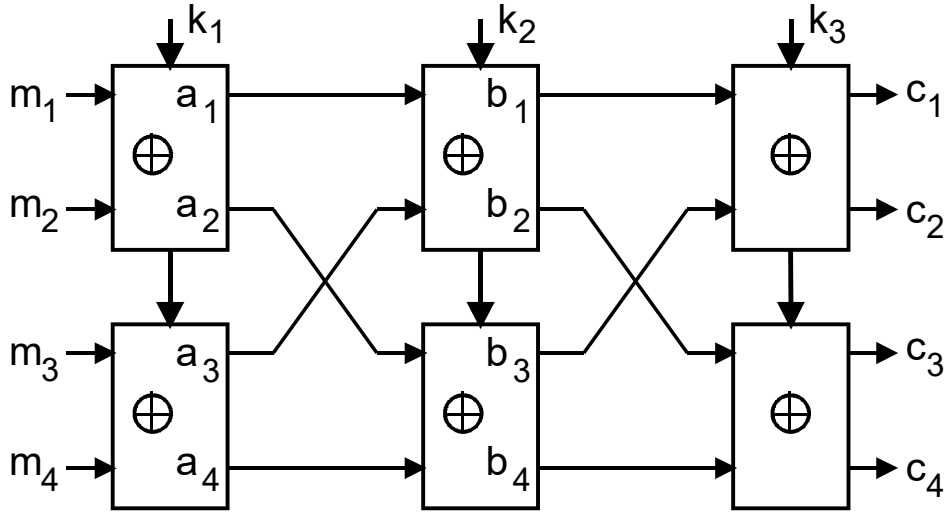
Sommersemester 2022
(LV 4120 und 7240)

## 8. Aufgabenblatt

Analyse und Entzifferung von einfachen Block- und Stromchiffren (Ver- und Entschlüsselung). Diskussion der Schlüsselwahl und -reihenfolge.

### Aufgabe 8.1

In der folgenden Abbildung ist eine einfache Blockschiffre $E_k : \{0, 1\}^{16} \times \{0, 1\}^{24} \rightarrow \{0, 1\}^{16}$ mit $c = (c_1, c_2, c_3, c_4) = E_k(m) = E_k(m_1, m_2, m_3, m_4)$ dargestellt, wobei der Schlüssel $k = (k_1, k_2, k_3)$ 24 Bit lang ist. Die Komponenten $m_i$ und $c_i$ , $1 \leq i \leq 4$ , sind jeweils 4 Bit lang. Die einzelnen Schlüsselkomponenten $k_1$, $k_2$ und $k_3$ besitzen jeweils eine Länge von 8 Bit.




Wird durch x II y die Konkatenation der Bitfolgen x und y dargestellt, und sind ein Klartext $m = (m_1, m_2, m_3, m_4)$ und ein Schlüssel $k = (k_1, k_2, k_3)$ gegeben, so ergibt sich der Chiffretext $c = (c_1, c_2, c_3, c_4)$ folgendermaßen:

$a_1 \text{ II } a_2 = k_1 \oplus (m_1 \text{ II } m_2)$ $\quad$ $a_3 \text{ II } a_4 = k_1 \oplus (m_3 \text{ II } m_4)$

$b_1 \text{ II } b_2 = k_2 \oplus (a_1 \text{ II } a_3)$ $\quad$ $b_3 \text{ II } b_4 = k_2 \oplus (a_2 \text{ II } a_4)$

$c_1 \text{ II } c_2 = k_3 \oplus (b_1 \text{ II } b_3)$ $\quad$ $c_3 \text{ II } c_4 = k_3 \oplus (b_2 \text{ II } b_4)$

a) Programmieren Sie für die Blockchiffre eine C/C++ Anwendung, die sowohl den 24 Bit Schlüssel k als auch den 16 Bit Klartext m von einer Eingabetextdatei einliest und den dazugehörigen 16 Bit Chiffretext c in eine Ausgabetextdatei schreibt.

   Testen Sie die Funktion der Blockchiffre mit Hilfe folgender Beispieldaten:

   Eingabedatei:

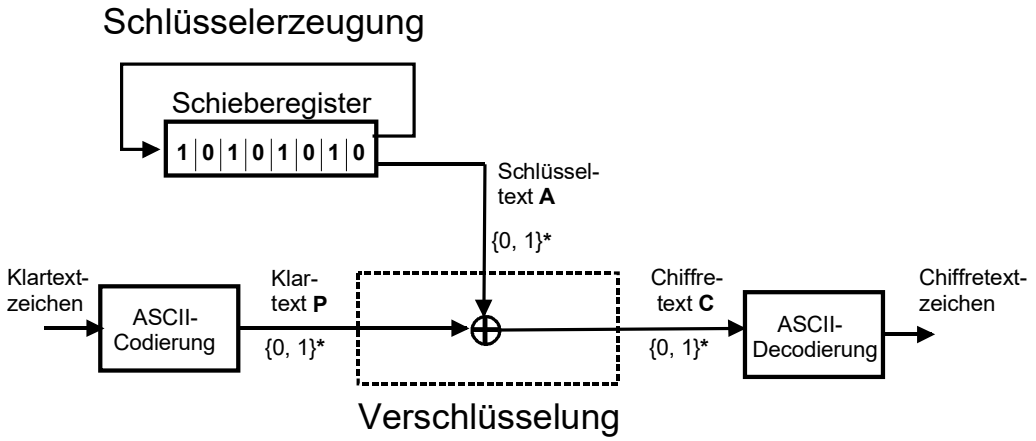
```
k = 10101010  11001100  00110011
m = 1001  0110  1100  0011
```

b) Wie lässt sich mit der angegebenen Blockchiffre aus dem Chiffretext c der Klartext m wieder rekonstruieren?

c) Testen Sie die Korrektheit Ihrer Implementierung, indem Sie den für die aufgeführten Beispieldaten erzielten Chiffretext c wieder in den Klartext m überführen.

**Aufgabe 8.2**

Wir betrachten die nachfolgend skizzierte binäre Stromchiffre, bei der der Chiffretext **C** aus der bitweisen XOR-Verknüpfung von Klartext **P** und Schlüsseltext **A** generiert wird. Der Einfachheit halber möge der verwendete Schlüsseltext mittels eines einfach rückgekoppelten Schieberegisters erzeugt werden, welches mit dem Cosetmuster 10101010 initialisiert wurde.




Die Codierung sowohl des Klartextes als auch des Chiffretextes erfolge mit Hilfe des 8-Bit ASCII-Zeichensatzes.

Programmieren Sie für die Stromchiffre eine C/C++ Anwendung, die den Klartext P von einer Eingabetextdatei einliest und den dazugehörigen Chiffretext C in eine Ausgabetextdatei schreibt.

Programmieren Sie für die Stromchiffre eine C/C++ Anwendung, die den Klartext P von einer Eingabetextdatei einliest und den dazugehörigen Chiffretext C in eine Ausgabetextdatei schreibt.

a) Testen Sie die Funktion der Stromchiffre mit Hilfe folgender Beispieldaten:

```
„Jede Sicherheitslücke ist zunächst auf ihre
Ausnutzbarkeit hin zu untersuchen.”
```

   Wie lautet der dazugehörige Chiffretext?

b) Wie lässt sich aus einem erhaltenen Chiffretext C der dazugehörige Plaintext P ermitteln?

c) Testen Sie auch die Umkehrfunktion. Was verbirgt sich hinter folgender Chiffre?

```
„ïØÙÞŠÝÏÄÄŠÏÃÄÏŠùÃÉÂÏØÂÏÃÞÙÆVÉÁÏŠËßÙÄßÞÐÈËØŠÃÙÞ†
ÝÃØÎŠÙÃÏŠÐßŠÏÃÄÏÇŠÚÅÞÏÄÞÃÏÆÆÏÄŠøÃÙÃÁÅ„ŠãÄŠÎÃÏÙÏÇ
ìËÆÆŠÙÃÄÎŠùÃÉÂÏØÂÏÃÞÙÜÅØÁÏÂØßÄÍÏÄŠÏØÌÅØÎÏØÆÃÉÂ†
ÎÃÏŠÏÄÞÝÏÎÏØŠÎÃÏŠùÃÉÂÏØÂÏÃÞÙÆVÉÁÏŠÙÉÂÆÃÏuÏÄŠÅÎÏØ
ËÈÏØŠÎËÙŠùÃÉÂÏØÂÏÃÞÙØÃÙÃÁÅŠËßÌŠÏÃÄŠÞÅÆÏØÃÏØÈËØÏÙ
ÇËuŠÈÏÍØÏÄÐÏÄŠ‡ŠÙÅÍÏÄËÄÄÞÏÙŠøÏÙÞØÃÙÃÁÅ„"
```

**Aufgabe 8.3**

Für die Zahl e = 9 ergibt sich in $\mathbf{Z}_{31}$ nach einer Multiplikation mit der Zahl 17 der Wert a = 29. Mit welcher Zahl $z \in \mathbf{Z}_{31}$ müsste man die Zahl a multiplizieren, um wieder in $\mathbf{Z}_{31}$ als Ergebnis der Multiplikation die Zahl e zu erhalten?

**Aufgabe 8.4**

Entwerfen und implementieren Sie eine kryptographische Funktion in C, welche im Restklassenring $\mathbf{Z}_n$ die Berechnung der modulare Exponentiation $a^b$ mod n mittels Square and Multiply Algorithmus ermöglicht.

a) Berechnung der modularen Exponentiation `ModExp(a, b, n)` für die natürlichen Zahlen a = 1.234.567.891.234.567, b = 1.234.567 und n = 543.222.266.

b) Vervollständigen Sie die nachstehende Tabelle mit Ihren Rechenergebnissen in der vierten Spalte.

| a | b | n | | $a^b$ mod n |
|---:|---:|---:|---|---|
| 4294967295 | 17 | 2147483647 | | |
| 4294967292 | 19 | 4294967295 | | |
| 4294967289 | 31 | 8589934591 | | |
| 4611686018427387909 | 64 | 4611686018427387903 | | |
| 9223372036854775813 | 64 | 9223372036854775807 | | |
| 18446744073709551615 | 64 | 18446744073709551611 | | |