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

Security

Sommersemester 2021
(LV 4121 und 4241)

9. Aufgabenblatt

Ziel des folgenden Aufgabenblatts ist es, eine Hillchiffre zu implementieren und den zugrundliegenden Algorithmus zu analysieren.

Aufgabe 9.1

a) Entwickeln und implementieren Sie in der Programmiersprache C eine Applikation für eine (3x3)-Matrix Hill-Chiffre in Form einer multiplikativen Tauschchiffre $E : \{0, 1\}^{64} \times \{0, 1\}^{64} \rightarrow \{0, 1\}^{72}$ mit $\mathbf{C} = (\mathbf{K} \cdot \mathbf{P}) \bmod n$ gemäß dem in Vorlesung betrachteten Verfahren. Realisieren Sie sowohl die Verschlüsselungs- als auch die Entschlüsselungsfunktion `hillverH33()` bzw. `hillentH33()`. Für die Invertierbarkeit der Chiffre wird die Bedingung ggT(det **K**, n) = 1 vorausgesetzt. Die Matrizen **C**, **K** und **P** betrachten wir jeweils als 3x3-Matrix mit jeweils 9 Elementen. Jedes Element hat eine Länge von einem Byte.

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,1} & x_{3,3} \end{pmatrix}$$

*Matrix-Form **X** für **C**, **K**, **P***

Das Feldelement $x_{2,2}$ wählen wir sowohl bei der Schlüsseltextmatrix **K** als auch bei der Klartextmatrix **P** als Prüffeld (Addition mod n) zur Absicherung der übrigen Elemente der Matrix gegenüber eventuellen Übertragungs- oder Berechnungsfehler, so dass sich mit diesem Verfahren eine Blocklänge von 8 x 8 Bit = 64 Bit erzielen lässt. Für die Berechnung des Feldelements $x_{2,2}$ biete sich an:

$$x_{2,2} = (x_{1,1} \oplus x_{1,2} \oplus x_{1,3} \oplus x_{2,1} \oplus x_{2,3} \oplus x_{3,1} \oplus x_{3,2} \oplus x_{3,3}) \bmod n;$$

Der verwendete Zeichensatz sei { a..z, |, !, :, -, ? } und alle Berechnungen seien im Restklassenring $\mathbf{Z}_{31}$ auszuführen.

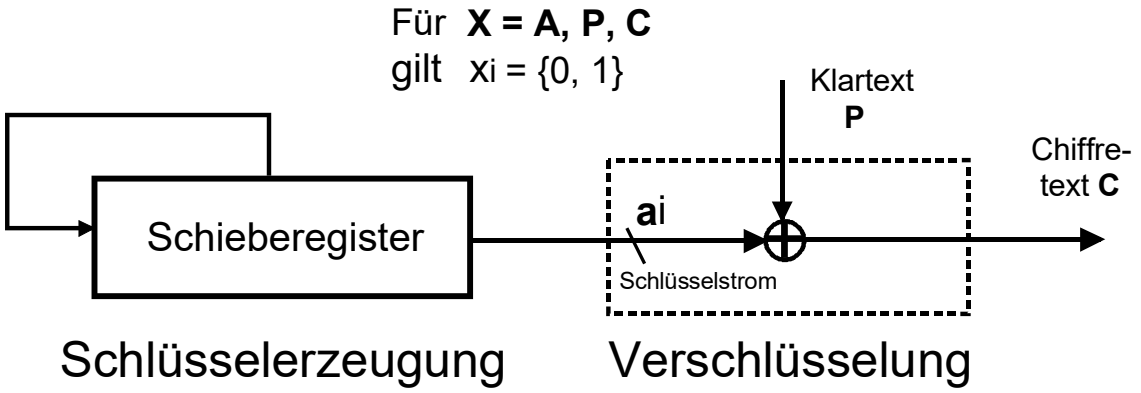
b) Ermitteln Sie unter Verwendung Ihres Programms für das 24 Zeichen lange Klartextpasswort `lestershlestersh:pass-rm` und den 8 Zeichen langen Schlüsseltext `hillkey!` den entsprechenden 24 Zeichen langen Ciphertext.

c) Wie lautet die entsprechende Dechiffrierfunktion und welche Matrizen müssen zum Entschlüsseln verwendet werden?

d) Ist durch die Wahl des Schlüsseltextes die Umkehrbarkeit der Chiffre gewährleistet? Begründen Sie Ihre Antwort.

Aufgabe 9.2

Bei einer binären Strom-Chiffre werden Klartext und Schlüssel modulo 2 addiert, um den Chiffretext zu erhalten. Umgekehrt werden zur Entschlüsselung Chiffretext und Schlüssel modulo 2 addiert. Eine besonders einfache Realisierung dieses Verfahrens (der "heiße Draht" zwischen Washington und Moskau soll auf diesem Prinzip basieren) mit Schieberegisterschaltungen und XOR-Schaltkreisen zeigt das folgende Prinzipschaltbild:



Bei der weiteren Betrachtung nehmen wir an, dass es sich um ein 4stufiges lineares Schieberegister handelt.

a) Welcher Schlüsselstrom ist erforderlich, um die Klartextfolge {1, 0, 1, 1, 0, 1, 1, 1} in die Chiffrefolge {1, 0, 1, 0, 0, 1, 0, 0} abzubilden?

b) Wie lautet der Startvektor des verwendeten Schieberegisters?

c) Ermitteln Sie die zugrundeliegende Rückkopplungsstruktur des Schieberegisters.

d) Skizzieren Sie die vollständige Schaltung des Verschlüsslers sowie des entsprechenden Entschlüsslers.

e) Was gilt es in bezug auf Synchronisation zu beachten?

Aufgabe 9.3

Gegeben sei eine Hash-Funktion H, die 10^8 unterschiedliche Hash-Werte erzeugen kann, z. B. Zahlen aus dem Intervall von 0 bis 99 999 999.

a) Weiterhin sei eine Nachricht M mit Hash-Wert H(M) gegeben. Wie viele Nachrichten müssen Sie erzeugen, um mit einer Wahrscheinlichkeit größer als 1/2 eine Nachricht mit demselben Hash-Wert H(M) zu erhalten?

b) Was ist eine Kollision bei Hash-Werten und wieso gibt es überhaupt Kollisionen?

Aufgabe 9.4

a) Entwickeln und implementieren Sie eine C-Funktion `hash(m)`, die zu einer einzulesenden Textdatei bzw. einer beliebig langen Nachricht m einen Fingerabdruck in Form eines Hashwertes h(m) erzeugt, der eine feste Länge von 56 Bit ausweist. Bei der Konstruktion des Hashalgorithmus werden jeweils 8 Zeichen der eingelesenen Nachricht m zu einem Block m_i (i = 0, 1, ..., M) mit der Länge von 8 Byte zusammengefasst und gemäß folgendem Algorithmus verarbeitet:

```
Wiederhole für alle Eingabetextblöcke i mit 0 <= i < M
    mi := block(z0 z1 z2 … z7);
mM := block(z0 z1 z2 … z5 '0' '0');
c0 = (m0)^k mod n ;
Wiederhole für alle Eingabetextblöcke i mit 0 < i <= M
    ci = (mi ⊕ ci-1)^k mod n ;
Output h(m) = cM ;
```

Dabei wird der zuletzt berechnete Block c_M als Hashwert der Nachricht m aufgefasst. Die beiden Sonderzeichen der Textdatei LF (\n) und EOF werden mit dem Wert 10 für \n bzw. 127 für EOF in die Berechnung des Hashwertes miteinbezogen. Falls ein Auffüllen (Pad) des letzten Blockes m_M erforderlich ist, um auf eine Blocklänge von 8 Zeichen zu kommen, erfolgt dies mit dem Zeichen '0' bzw. ASCII-Wert 48 von rechts (LSB) nach links (MSB). Die Parameter dieses Hash-Algorithmus sind n und k. Sie sind für die Kollisionsresistenz von entscheidender Bedeutung.

Wir interpretieren die Zeichen der Textdatei als 7-Bit-ASCII im Wertebereich von 0 bis 127 (dezimal) bzw. 0 bis 7F (hexadezimal) und wenden folgende Zahleninterpretation für den Wert eines Blockes an.

$$Wert(m_i) = z_0 \cdot 2^{49} + z_1 \cdot 2^{42} + \ldots + z_6 \cdot 2^7 + z_7$$

bzw. für $0 \le i < M$

$$Wert(m_i) = z_0 <<<_{49} + z_1 <<<_{42} + \ldots + z_6 <<<_7 + z_7$$

und für den letzten Block m_M:

$$Wert(m_M) = z_0 \cdot 2^{49} + z_1 \cdot 2^{42} + \ldots + \backslash n \cdot 2^{21} + EOF \cdot 2^{14} + '0' \cdot 2^7 + '0'$$

Zwischenräume (Leerzeichen) werden mit dem Wert 32 berücksichtigt.

b) Ermitteln Sie für den Wert der folgenden Eingabetexte im UTF-8-Format den 56 Bit Hashwert als Dezimal- und Hexadezimalwert (einschließlich der beiden Steuerzeichen LF und EOL). Parameter sind k = 17 und n = 72.057.594.037.927.935.

Eingabetext	**56 Bit Hashwert (dezimal und hexadezimal)**	**AZ**
12345678		
123456789		
01234567		

AZ: *Anzahl Zeichen (einschließlich \n und EoF)*