LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Praktikum Rechnerarchitektur

Gruppe 155 – Abgabe zu Aufgabe A501

Sommersemester 2019

Mete Polat Jonas Hübotter Simon Martin Bohnen

1 Einleitung

2 Problemstellung und Spezifikation

3 Lösungsfindung

3.1 Initialisierungsvektor

Eine weitere Herausforderung war die Erzeugung und Speicherung des Initialisierungsvektors, der beim Cipher Block Chaining Mode benötigt wird. Es ist einerseits entscheidend, dass dieser nicht aus zuvor bekannten Informationen erzeugt wird, wie es zum Beispiel bei SSL 2.0 der Fall war. Andererseits ist eine ausreichende Länge wichtig, um einem Related-Key-Attack vorzubeugen, der zum Beispiel das WEP-Protokoll betraf.

Aufgrund der Blocklänge bot sich für uns nur ein 32-Bit-Initialisierungsvektor an, den wir pseudozufällig generieren. Eine Geheimhaltung des Initialisierungsvektors ist nicht erforderlich, weshalb der Vektor am Ende der verschlüsselten Datei gespeichert und dort bei der Entschlüsselung wieder ausgelesen wird.

3.2 Einmaliges Ausführen der Schlüsselexpansion

Ein bei gleicher Rundenanzahl und Blockgröße stets gleich bleibender Schritt bei der Ver- und Entschlüsselung ist die Schlüsselexpansion. Diese hängt nur von den Nothing-Up-My-Sleeve-Zahlen P und Q ab und muss daher im Voraus nur einmal berechnet werden. Die resultierenden Rundenschlüssel speichern wir in der data-Section unseres Assemblyprogramms, um sie beim Key-Mixing weiter zu verwenden. Der Code zur Schlüsselexpansion ist seperat in der Datei TODO zu finden.

3.3 Optimierung durch SIMD

Eine Optimierung durch SIMD ist möglich und sinnvoll, wenn auf mehreren Datenblöcken die selbe Operation ohne Abhängigkeiten zwischen Blöcken ausgeführt wird. Bei RC5 und dem CBC-Mode werden jedoch häufig Abhängigkeiten verwendet, um statistischen Analysen, wie sie zum Beispiel beim ECB-Mode möglich sind, vorzubeugen. Beim Key-Mixing hängt der nächste Rundenschlüssel beispielsweise direkt vom vorherigen ab, wodurch eine Parallelisierung unmöglich wird. Ähnliches gilt für die Ver- und Entschlüsselung, da dort Da beim Cipher Block Chaining Mode der folgende

Ciphertextblock stets vom aktuellen abhängt, ist eine Optimierung durch SIMD bei einer Blockgröße von 32 Bit nicht möglich. -IV am Dateiende -andere Mögl: aus Key generieren, Nutzer eingeben lassen -Kein SIMD -Entscheidung Schlüsselexpansion hardgecodet

4 Dokumentation der Implementierung

5 Ergebnisse

5.1 RC5-16/16/b

5.1.1 Schlüsselerweiterung mit P und Q

Für den Schritt der Schlüsselexpansion von RC5 werden die beiden ungeraden Ganzzahlen P und Q benötigt. Diese sind jeweils für eine gegebene Blockgröße von RC5 konstant. Im Allgemeinen gilt

$$P = Odd((e-2) \cdot 2^w) \tag{1}$$

$$Q = Odd((\phi-1) \cdot 2^w) \tag{2}$$

mit w als der Größe eines Halbblocks in Bits, e als der Eulerschen Zahl und ϕ als dem Goldenen Schnitt wobei $Odd(x)$ die jeweils näheste ungerade Zahl bezeichnet.

Für RC5-16/16/b — RC5 mit der Wortgröße 16, der Rundenanzahl 16 und der Schlüsselgröße b — gilt damit:

$$\begin{aligned}
P &= Odd((2{,}71828-2) \cdot 2^{16}) \\
&= Odd(0{,}71828 \cdot 65.536) \\
&= Odd(47.073{,}19808) \\
&= 47.073 \\
&= 0xb7e1 \\
Q &= Odd((1{,}61803-1) \cdot 2^{16}) \\
&= Odd(0{,}61803 \cdot 65.536) \\
&= Odd(40.503{,}21408) \\
&= 40.503 \\
&= 0x9e37
\end{aligned}$$

5.1.2 Sicherheit

Durch die Parametrisierung von RC5[6, p.2] unterstützt die Chiffre sowohl unterschiedliche Blockgrößen, unterschiedliche Schlüssellängen als auch eine variable Anzahl von Runden. Hinter dieser Parametrisierung stehen zwei Intentionen:

1. Durch die variable Blockgröße soll die Performance von RC5 von neueren 64-Bit Architekturen profitieren — allerdings nicht auf diese beschränkt sein.[6, p.1]

2. Andererseits sollen die frei wählbaren Parameter r und b dem Nutzer der Chiffre die Entscheidung überlassen, wie viel Sicherheit und welche Performance seine Applikation benötigt.[6, p.1]

Damit sind zur Beurteilung der Sicherheit von RC5 insbesondere die Rundenanzahl und die Schlüssellänge zu betrachten.

Schlüssellänge So wie im Allgemeinen bei Blockchiffren ist auch bei RC5 die Sicherheit der Chiffre stark von der gewählten Schlüssellänge abhängig. RC5 hat den Parameter b mit $b \in \{k \in \mathbb{N}_0 \colon k \leq 255\}$, der die Länge des Schlüssels in Bytes angibt.[6, p.3] Die Länge der erweiterten Schlüsseltabelle in Bits ergibt sich durch $2^{(2r+2)w}$.[6, p.2] Der Aufwand für eine *erschöpfende Suche* ist damit $min\{2^{8b}, 2^{(2r+2)w}\}$.[9, p.29] Für RC5-16/16/b ist damit der Aufwand einer erschöpfenden Suche allein von b abhängig, solange $b < 68$ gilt.

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) schlägt für Blockchiffren wie RC5 eine minimale Schlüssellänge von 128 Bits vor.[7, p.21]

Mode of Operation In der Praxis werden Blockchiffren in der Regel mit einem Mode of Operation umgesetzt, mithilfe dessen auch Nachrichten variabler Länge verschlüsselt werden können. Entscheidend ist für deren Sicherheit, dass der entstehende Ciphertext *pseudorandom* — ohne Wissen über Plaintext oder Schlüssel stochastisch zufällig[5] — ist.

Ist dies nicht der Fall — resultieren äquivalente Plaintext-Blöcke beispielsweise in äquivalenten Ciphertext-Blöcken —, dann enthält der Ciphertext Informationen zur Struktur des Plaintextes[7, p.22]. In diesem Fall können Teile des Plaintextes durch *Häufigkeitsanalyse* rekonstruiert werden[7, p.22], oder sogar der verwendete Schlüssel durch einen *Codebook Attack* gewonnen werden[1, p.2].

Sichere Operationsmodi sind beispielsweise *Counter Block Chaining (CBC)* und der *Counter Mode (CTR)*, da dort der n-te Chiphertext-Block nicht nur von dem n-ten Plaintext-Block und dem genutzten Schlüssel abhängt, sondern zudem noch von einem weiteren Wert, wie dem $(n-1)$-ten Ciphertext-Block oder einem Zähler.[7, p.22]

Rundenanzahl Nach Kaliski und Yin werden für eine *differenzielle Kryptoanalyse* von RC5-32/16/b entweder 2^{61} selbst gewählte Plaintexte oder 2^{63} bekannte Plaintexte benötigt. Ein solcher Angriff auf ein 16-rundiges RC5 ist damit überaus unwahrscheinlich. Da die Anzahl der möglichen Plaintexte bei dieser Konfiguration von RC5 jedoch bei 2^{64} liegt, kann ein solcher Angriff nicht theoretisch ausgeschlossen werden.[9, p.6] Weiterhin sei eine *lineare Kryptoanalyse von RC5* nur bei einer sehr geringen Rundenzahl von RC5 effektiv.[9, p.28]

Knudsen und Meier zeigen zwar, dass die Komplexität des von Kaliski und Yin vorgeschlagenen differenziellen Angriffs um einen Faktor von bis zu 512 reduziert werden kann.[8, p.2] Allerdings bleibt ein solcher Angriff damit weiterhin sehr unwahrscheinlich. Zudem wurde gezeigt, dass für bestimmte Teile des Schlüsselraums die differentiellen Kryptoanalysen weiter verbessert werden können.[8, p.13] Allerdings sind für einen effektiven Angriff entweder zu wenige Schlüssel betroffen oder die Anzahl der benötigten Plaintexte weiterhin zu hoch. Heys konnte Ähnliches für lineare Kryptoanalyse zeigen.[2, p.5]

Damit kann RC5-16/16/b bei $b >= 16$ und Nutzung eines geeigneten Operationsmodus als sicher gelten.

5.2 Feistelchiffren

Die folgende allgemeine Darstellung von Feistelchiffren soll auf klassische (auch ausgewogene) Feistelchiffren begrenzt werden. Wie für RC5, gilt für klassische Feistelchiffren, dass die Längen der beiden Halbblöcke eines Blocks gleich sein müssen. Zudem wird sich auf das für die umkehrbare Verknüpfung von zwei Halbblöcken übliche $\oplus$ (XOR) beschränkt.

5.2.1 Einrundige Feistelnetzwerke

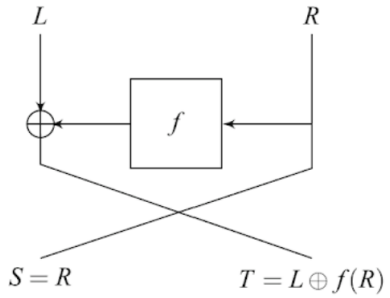
Eine Feistelchiffre ist eine rundenbasierte Blockchiffre, die nach der Art eines Feistelnetzwerks aufgebaut ist. Sei

$$F_n := \{f \mid f\colon \{0,1\}^n \to \{0,1\}^n\}$$

die Famile der Rundenfunktionen. Zunächst soll ein klassisches einrundiges Feistelnetzwerk Ψ betrachtet werden. Dieses wird definiert durch eine beliebige Abbildung $f \in F_n$ und eine umkehrbare Bitoperation — durch obige Einschränkung der Allgemeinheit $\oplus$.

$$\Psi(f)\colon \{0,1\}^{2n} \to \{0,1\}^{2n}\colon [L,R] \mapsto [S,T] \Leftrightarrow \begin{cases} S = R \\ T = L \oplus f(R) \end{cases}$$

für $\forall (L,R) \in (\{0,1\}^n)^2$.[4, p.11]



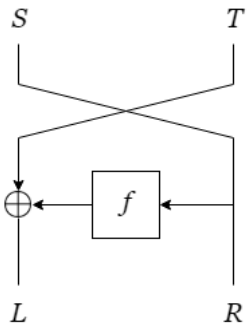
[4, Fig. 2.1]

Wichtig für jede Verschlüsselung ist Bijektivität, damit jedem Codewort eine eindeutige Plaintext-Nachricht zugeordnet werden kann. $\Psi(f)$ ist unabhängig von $f \in F_n$ eine Permutation, d.h. f selbst muss nicht bijektiv sein.[4, p.12]

Aus der Definition von $\Psi(f)$ ergibt sich ihr Inverses als

$$\Psi(f)^{-1} = \sigma \circ \Psi(f) \circ \sigma$$

mit σ definiert als $\sigma([L, R]) = [R, L]$ für $L, R \in \{0, 1\}^n$, der Vertauschung beider Halbblöcke.[4, p.12]

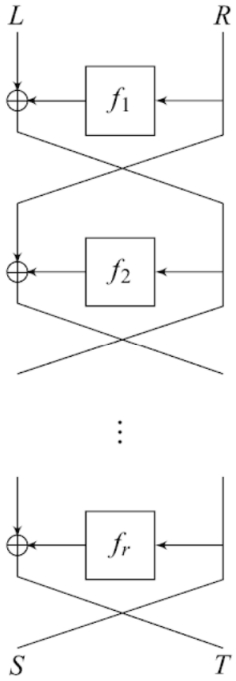


5.2.2 r-rundige Feistelnetzwerke

Üblicherweise werden Feistelnetzwerke in mehreren Runden angewendet. Im Allgemeinen ist ein klassisches Feistelnetzwerk mit $r \geq 1$ Runden und $f_1, f_2, ..., f_r \in F_n$ Rundenfunktionen definiert durch

$$\Psi^r(f_1, ..., f_r) = \Psi(f_r) \circ ... \circ \Psi(f_2) \circ \Psi(f_1).$$

[4, p.12]



[4, Fig. 2.2]

Da ein einrundiges Feistelnetzwerk eine Permutation über $\{0,1\}^{2n}$ ist, sind auch r-rundige Feistelnetzwerke Permutationen. Weiterhin ist das Inverse eines r-rundigen Feistelnetzwerks die Komposition der Inversen der einzelnen Runden.

$$\begin{aligned}(\Psi^r(f_1, ..., f_r))^{-1} &= \sigma \circ \Psi(f_1) \circ \sigma \circ ... \circ \sigma \circ \Psi(f_r) \circ \sigma \\ &= \sigma \circ \Psi^r(f_r, ..., f_1) \circ \sigma\end{aligned}$$

[4, p.13]

Eine Feistelchiffre ist nun ein spezielles Feistelnetzwerk, dessen Rundenfunktionen von einem Rundenschlüssel aus dem Schlüsselraum K abhängen. Seien die Rundenschlüssel $(k_1, ..., k_r) \in K^r$ und die Familie der Rundenfunktionen

$$F_{n,K} := \{f_k \mid k \in K, f_k \colon \{0,1\}^n \to \{0,1\}^n\}.$$

Dann ist eine Feistelchiffre das Feistelnetzwerk $\Psi^r(f_{k_1},...,f_{k_r})$. Also die r-rundige Permutation von der Nachricht $\{0,1\}^{2n}$ in Abhängigkeit vom Schlüssel $(k_1, ..., k_r)$.[4, p.14]

5.2.3 RC5 als Feistelchiffre

RC5 ist eine symmetrische Blockchiffre, deren Aufbau dem einer Feistelchiffre gleicht. RC5 hat die Parameter:[6, p.2f]

- w ist die Wortgröße in Bits. Ein durch RC5 verschlüsselbarer Block besteht aus zwei Wörtern.
- r ist die Anzahl der Runden in denen RC5 Operationen auf einem Block ausführt. Jede Runde besteht aus zwei Halbrunden, in denen ein Wort aus dem Block alteriert wird.
- b ist die Anzahl der Bytes in dem privaten Schlüssel K.

RC5 baut zu Beginn die erweiterte Schlüsseltabelle S auf, die aus $2r + 2$ Schlüsseln besteht und von K abhängt. Seien $\Sigma := (S_2, S_3, ..., S_{2r+1}) = (\Sigma_0, \Sigma_1, ..., \Sigma_{2r-1})$ mit $|\Sigma| = 2r$ die Schlüssel aus der erweiterten Schlüsseltabelle, die während der Runden von RC5 zum Verschlüsseln benutzt werden — S_0 und S_1 werden für das Key-Whitening genutzt. Zudem sei $(g_{\Sigma_0}, g_{\Sigma_1}, ..., g_{\Sigma_{2r-1}})$ definiert durch

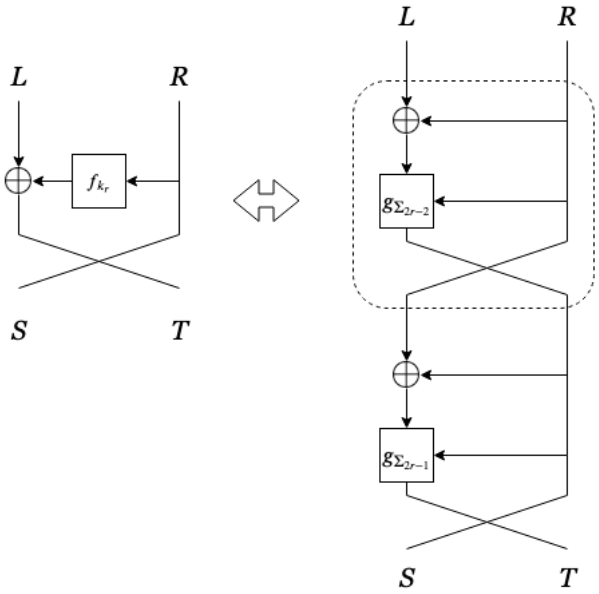
$$g_k \colon \{0,1\}^w \times \{0,1\}^w \to \{0,1\}^w \colon$$
$$(\tau, R) \mapsto (\tau \lll R) + k$$

mit $\tau = L \oplus R$, $k \in \Sigma$ und $L, R \in \{0,1\}^w$ wobei $x \lll y$ die Linksrotation von x um y Bits angibt. Dann zeigt die folgende Tabelle die Zusammenhänge von RC5 und Feistelchiffren.

RC5	Feistelchiffre
r	$2r$
w	n
Σ	K^{2r}
$(g_{\Sigma_0}, g_{\Sigma_1}, ..., g_{\Sigma_{2r-1}})$	$(f_{k_1}, ..., f_{k_{2r}}) \in F^{2r}_{n,K}$

Die Reihenfolge der Anwendung der umkehrbaren Bitoperation ($\oplus$) und der Rundenfunktion unterscheidet sich leicht zwischen RC5 und einer allgemeinen klassischen

Feistelchiffre. Dieser Unterschied soll in der folgenden Abbildung skizziert werden.



Links eine Runde einer Feistelchiffre, rechts eine Runde (zwei Halbrunden) von RC5.

wobei r die aktuelle Runde angibt. Wie dargestellt, ist eine Halbrunde von RC5 im Aufbau ähnlich zu einer Runde einer Feistelchiffre.

Durch den leicht modifizierten Aufbau einer Feistelchiffre in RC5, verändert sich bei RC5 die Berechnung der Inversen. Für eine RC5-Runde gilt

$$RC5_{r,\Sigma}\colon \{0,1\}^{2w} \to \{0,1\}^{2w}\colon [L,R] \mapsto [S,T] \Leftrightarrow \begin{cases} S = ((L \oplus R) \lll R) + \Sigma_{2r-2} \\ T = ((R \oplus S) \lll S) + \Sigma_{2r-1} \end{cases}.$$

Damit gilt für die Berechnung der Inversen von einer RC5-Runde

$$RC5^{-1}_{r,\Sigma}\colon \{0,1\}^{2w} \to \{0,1\}^{2w}\colon [S,T] \mapsto [L,R] \Leftrightarrow \begin{cases} L = ((S - \Sigma_{2r-2}) \ggg R) \oplus R \\ R = ((T - \Sigma_{2r-1}) \ggg S) \oplus S \end{cases}$$

für $\forall (S,T) \in (\{0,1\}^w)^2$ wobei $x \ggg y$ die Rechtsrotation von x um y Bits angibt.

5.3 PKCS#7-Padding

Da eine Blockchiffre nur Nachrichten vollständig verschlüsseln kann, die restfrei in Blöcke geteilt werden können, muss die Länge dieser Nachrichten zunächst auf ein Vielfaches der Blockgröße erweitert werden. Diese Erweiterung wird im Allgemeinen als Padding bezeichnet.

Das PKCS#7-Padding ist eine Form der Erweiterung des Plaintextes auf ein Vielfaches der Blocklänge und soll im Folgenden erläutert werden. Es sei Δ definiert als

$$\Delta = b - (l \bmod b)$$

mit b als der Länge eines Blocks und l als der Länge des Plaintextes in Byte. Vor der Anwendung eines Verschlüsselungsalgorithmus, der als Länge des Inputs ein Vielfaches von b Bytes erwartet, werden Δ Bytes jeweils mit dem Wert Δ an den Plaintext angefügt.[3, p.28]

Das heißt, dass der Input in Abhängigkeit von b und l um eine der folgenden Byte-Sequenzen erweitert wird:

```
         01 -- if l mod b = b-1
      02 02 -- if l mod b = b-2
          .
          .
          .
b b ... b b -- if l mod b = 0
```

Nach dem Entschlüsseln des Codewortes, kann das Padding auf eindeutige Weise entfernt werden, da jeder Plaintext — einschließlich jener, deren Länge selbst ein Vielfaches der Blockgröße ist — vor der Verschlüsselung mit PKCS#7-Padding erweitert wurde. Die Anzahl der zu entfernenden Bytes wird durch das letzte Byte des letzten Blocks angegeben. PKCS#7-Padding ist wohldefiniert für $b < 256$.[3, p.28]

5.4 Performance

6 Zusammenfassung

Literatur

[1] Limor Elbaz und Hagai Bar-El. „Strength Assessment of Encryption Algorithms“. In: (Okt. 2000). URL: https://pdfs.semanticscholar.org/03cb/17aad62a46d0fad1133f9656ff0f8a4e39dc.pdf. (aufgerufen: 06.07.2019).

[2] H. M. Heys. „Linearly Weak Keys of RC5“. In: (Mai 1997). URL: https://pdfs.semanticscholar.org/f4cb/0acab4eb24a74e49fce909496b10d62b266c.pdf. (aufgerufen: 06.07.2019).

[3] R. Housley. *[RFC5652] Cryptographic Message Syntax (CMS)*. URL: https://tools.ietf.org/html/rfc5652. (aufgerufen: 29.06.2019).

[4] Valerie Nachef und Jacques Patarin und Emmanuel Volte. *Feistel Ciphers: Security Proofs and Cryptanalysis*. Springer, 2017. ISBN: 9783319495309.

[5] *Pseudorandomness*. URL: https://en.wikipedia.org/wiki/Pseudorandomness. (aufgerufen: 06.07.2019).

[6] Ronald L. Rivest. „The RC5 Encryption Algorithm*“. In: (März 1997). URL: http://people.csail.mit.edu/rivest/Rivest-rc5rev.pdf. (aufgerufen: 06.07.2019).

[7] Bundesamt für Sicherheit und Informationstechnik. *Cryptographic Mechanisms: Recommendations and Key Lengths*. Technical Guideline TR-02102-1. Jan. 2019. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf?__blob=publicationFile&v=9. (aufgerufen: 06.07.2019).

[8] Lars R. Knudsen und Willi Meier. „Improved Differential Attacks on RC5“. In: Nov. 1998. URL: https://link.springer.com/content/pdf/10.1007%2F3-540-68697-5_17.pdf. (aufgerufen: 06.07.2019).

[9] Burton S. Kaliski Jr. und Yiqun Lisa Yin. *On the Security of the RC5 Encryption Algorithm*. Technical Report TR-602. Version 1.0. RSA Laboratories, Sep. 1998.