Vorlesung Sicherheit

Dennis Hofheinz

ITI, KIT

27.04.2017

Organisatorisches

- Mailingliste (Informationen auf Webseite)
  - Anmeldung optional, erleichtert aber Kommunikation

Erinnerung

- Sicherheit durch sichere Bausteine
- Erstes Thema: (symmetrische) Verschlüsselung $(\mathrm{Enc}, \mathrm{Dec})$

$$\texttt{Alice}_K \xleftarrow{C:=\mathrm{Enc}(K,M)} \texttt{Bob}_K$$

- Beispiele: Cäsar, Vigenère, One-Time-Pad
- OTP: $C = M \oplus K$ (unhandlich, veränderbar)
- Stromchiffren: „Simulation“ von OTP (veränderbar)

Überblick

Überblick

Struktur von Blockchiffren (I)

Eine Blockchiffre besteht aus zwei Funktionen E & D

Verschlüsselung:

- Funktion $\mathrm{E} : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$
  - k: Schlüssellänge, Schlüssel sind also Bitstrings der Länge k
  - ℓ: Blocklänge für Klartexte und Chiffrate
  - E bildet Schlüssel und Klartextblock auf Chiffratblock ab

Entschlüsselung:

- Funktion $\mathrm{D} : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$
- **Korrektheit:**

$$\forall K, M: \quad \mathrm{D}(K, \mathrm{E}(K, M)) \stackrel{!}{=} M$$

Struktur von Blockchiffren (II)

Schlüssel: *Symmetrisches* Verfahren, d.h.

- Ein (geheimer) Schlüssel für Ver- und Entschlüsselung!
- Muss zwischen Sender und Empfänger ausgetauscht werden.

Anwendung:

- Klartexte bestehen aus mehreren Blöcken.
- E und D verarbeiten jeweils nur einen Block.
- Verschiedene Wege, E & D zu benutzen (Betriebsmodi).

Überblick

Electronic Codebook (ECB) Mode

- Erinnerung: $\mathrm{E}, \mathrm{D} : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$
- Einfachster Weg, zu verschlüsseln:
  - Teile M in ℓ-Bit-Blöcke $M_1, \ldots \in \{0,1\}^\ell$ auf
  - Setze $C := (C_1, \ldots)$ mit $C_i := \mathrm{E}(K, M_i) \in \{0,1\}^\ell$
  - Entschlüsselung funktioniert genauso, nur mit D
- **Frage:** Vorteile/Nachteile?

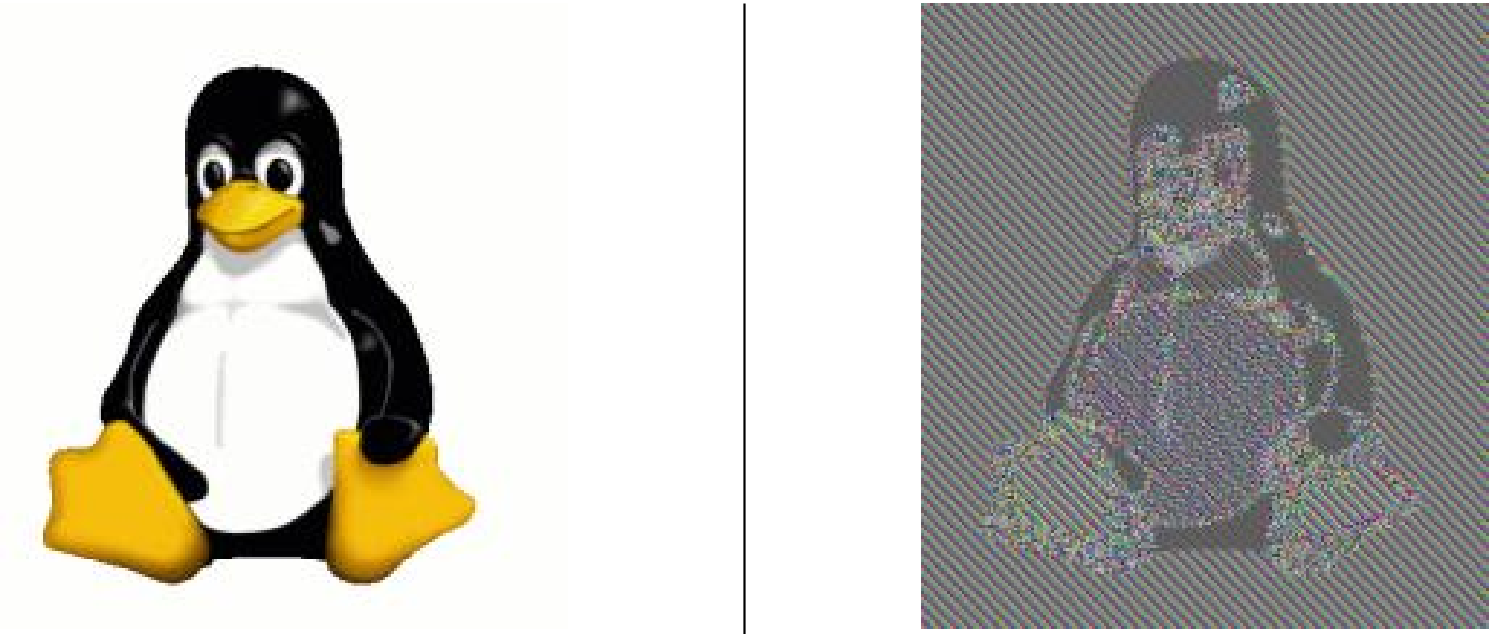
Eigenschaften des ECB

Erinnerung: $C := (C_1, \dots)$ mit $C_i := \mathrm{E}(K, M_i)$

- Vorteile:
  - Einfach zu implementieren
  - Kein Zustands-Update, keine Synchronisation nötig
- Nachteile:
  - Gleiche Nachricht $\Rightarrow$ gleiches Chiffrat
  - Einfügen/Umsortieren von Chiffratblöcken möglich
- Bitfehler in C_i: Block M_i zerstört

- **Fun fact:** Bundestrojaner nutzt AES (gängige Blockchiffre) im ECB-Modus (mit festkodiertem Schlüssel)

Eigenschaften des ECB (Beispiel)



ECB-Verschlüsselung (links Nachricht, rechts Chiffrat) (Wikipedia)

- **Frage:** Wie können Nachteile behoben werden?

Cipher Block Chaining (CBC) Mode

- Erinnerung: $\mathrm{E}, \mathrm{D} : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$
- Problem des ECB: Chiffratblöcke „unabhängig“
- Idee des CBC: Chiffratblöcke verketten:
  - Teile M in ℓ-Bit-Blöcke $M_1, \ldots \in \{0,1\}^\ell$ auf
  - Setze $C_0 := IV$ (Initialisierungsvektor)
  - Setze $C_i := \mathrm{E}(K, M_i \oplus C_{i-1})$
  - Entschlüsselung: $M_i := \mathrm{D}(K, C_i) \oplus C_{i-1}$
- IV muss mit übertragen werden (oder konstant sein)
- **Frage:** Vorteile/Nachteile?

Eigenschaften des CBC (I)

Erinnerung: $C_i := \mathrm{E}(K, M_i \oplus C_{i-1})$

- CBC behebt Nachteile des ECB:
  - Gleiche Nachricht $\Rightarrow$ unterschiedliche Chiffrate
    (bei unterschiedlichen vorherigen Chiffraten)
  - Umsortierung von Chiffratblöcken führt zu fehlerhaften Blöcken
  - **Frage:** Welche Blöcke werden genau zerstört?
- Vorteile erkauft mit neuen Nachteilen:
  - *Ver*schlüsselung nicht parallelisierbar (C_{i-1} muss bekannt sein)
  - **Aber:** *Ent*schlüsselung parallelisierbar, (fast) wahlfreier Zugriff (**Frage:** wie?)
  - Chiffrate veränderbar (annähernd XOR-homomorph)

Eigenschaften des CBC (II)

- Bitfehler in C_i an Stelle j: Block M_i zerstört und Bit j in M_{i+1} negiert.

- Hauptproblem des CBC: „annähernde XOR-Homomorphie“
  - Ändern von C_i ändert entschlüsseltes M_{i+1}
  - Kann gewisse Informationen über M_{i+1} liefern
    (etwa: $M_{i+1} \oplus X$ noch „gültig“)
- Beispiele für konkrete Probleme, die hierdurch entstehen:
  - Angriffe auf TLS (wird noch besprochen)
  - Angriffe auf Linux-Festplattenverschlüsselung
    `http://www.jakoblell.com/blog/2013/12/22/practical-malleability-attack-against-cbc-encrypted-luks-partitions/`

Weitere Betriebsmodi

- Counter (CTR) Mode (ähnelt Stromchiffre)

$$C_0 := IV$$
$$C_i := \mathrm{E}(K, IV + i) \oplus M_i$$

  - Ähnliche Eigenschaften wie CBC (aber besser parallelisierbar)
  - **Allerdings:** wie CBC auch homomorph veränderbar
- **Deshalb:** Galois Counter Mode (GCM)
  - Authentifizierter CTR Mode (mit „Prüfsumme“)
  - Schützt gegen Manipulation der Chiffrate
  - **Allerdings:** Kompromiss (kleine Authentifikationstags), besser: „richtigen“ MAC zum Authentifizieren benutzen

Zusammenfassung Betriebsmodi

- Blockchiffre benutzt blockweise Funktion E in Betriebsmodus
- ECB: „rohe“ Funktion E, **nicht benutzen!**
- CBC,CTR: besser, schützt aber nur gegen Lauschangriffe
- GCM: Betriebsmodus der Wahl
- **Später:** Formalisierung der Sicherheit von CBC/CTR
  - Klärt auch die Wahl von *IV*
- Mehr in Vorlesung „Symmetrische Verschlüsselungsverfahren“

Überblick

Data Encryption Standard (DES)

- Erinnerung: $\mathrm{E} : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$ zentral
- DES: Beispiel für E mit $k = 56$ und $\ell = 64$
- Mittlerweile veraltet (zu kurzer Schlüssel)
- Aber: historisch und technisch interessant
  - Verwendete Feistel-Netzwerke interessante Struktur
  - Rundenfunktion F ohne Falltür $\to$ Falltürfunktion E
    - D.h. selbst F wenn nicht invertierbar, können wir trotzdem entschlüsseln
  - Strukturell ungebrochen[1]

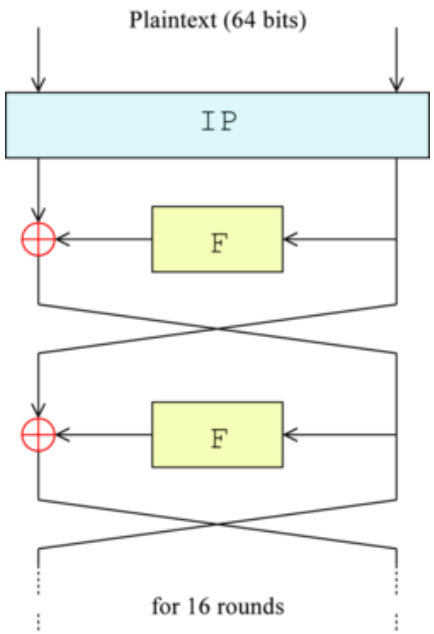
[1] lineare Kryptoanalyse besser als vollständige Suche, aber nicht praktikabel

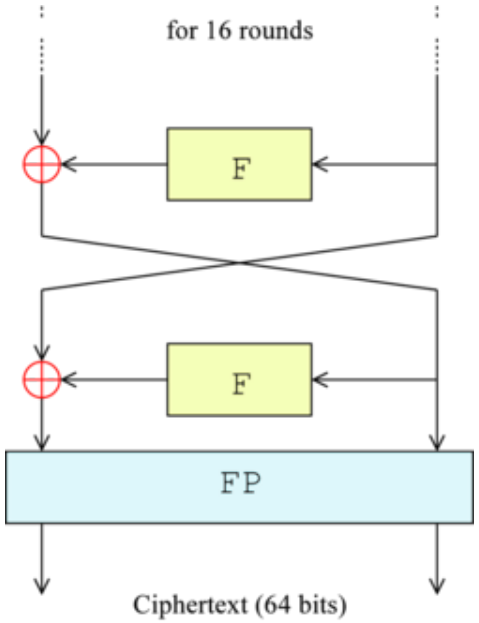
DES: Allgemeines

- Eingangs- und Ausgangspermutationen
- Struktur: 16 Runden Feistel-Struktur
- Verschlüsselt 64-Bit Blöcke, die in zwei 32-Bit Blöcke aufgeteilt werden
- Rundenfunktion F
- 56 Bit Schlüssel wird auf 16 Rundenschlüssel mit je 48 Bit „erweitert“

DES-Feistelstruktur

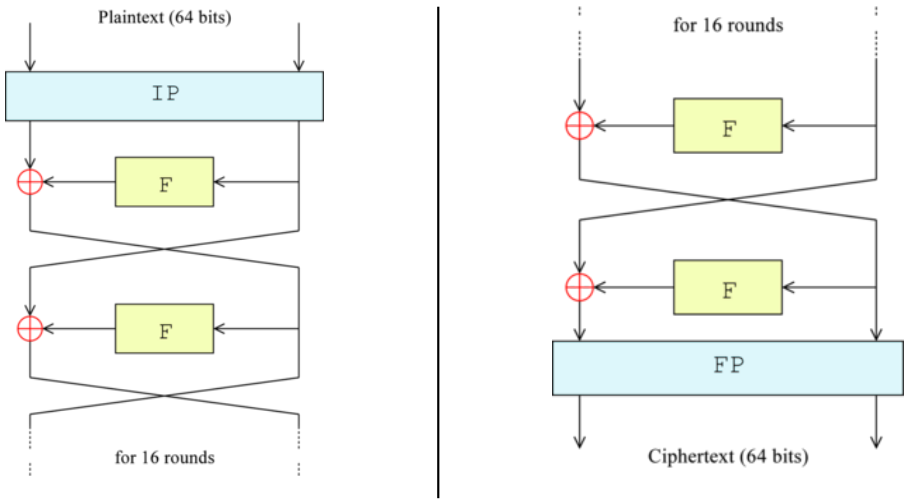
DES-Feistelstruktur (links Anfang, rechts Ende) (Wikipedia)





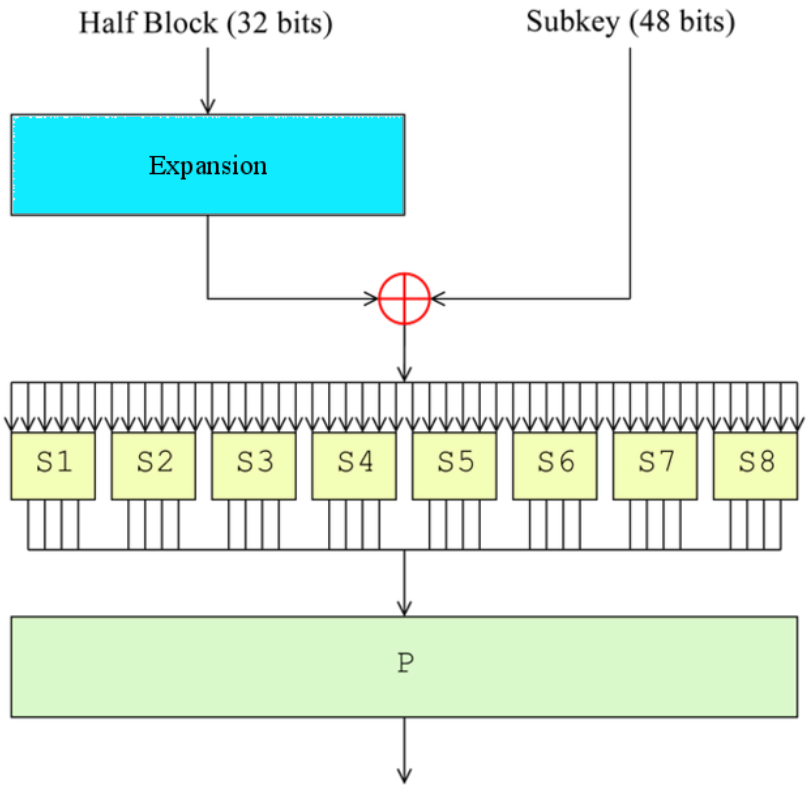
Wichtig: F erhält auch 48-Bit Rundenschlüssel K_i als Eingabe!

DES-Feistelstruktur



- Eingangs- und Ausgangspermutation historisch bedingt
- IP und FP sind invers, d.h. $\mathrm{IP} = \mathrm{FP}^{-1}$
- **Wichtig:** F muss für Entschlüsselung nicht invertierbar sein!
- Entschlüsselung hat Feistel-Struktur wie Verschlüsselung
  (nur mit F-Teilschlüsseln in umgekehrter Reihenfolge)

DES-Rundenfunktion F



DES-Rundenfunktion $\mathrm{F} : \{0,1\}^{48} \times \{0,1\}^{32} \to \{0,1\}^{32}$ (Wikipedia)

Überblick

2DES

- DES-Schlüssel zu kurz (56 Bits)
- Naive Verbesserung: 2DES
  - $K := (K_1, K_2) \in (\{0,1\}^{56})^2$
  - $\mathrm{E}_{\mathrm{2DES}}(K, M) := \mathrm{E}_{\mathrm{DES}}(K_2, \mathrm{E}_{\mathrm{DES}}(K_1, M))$
  - Erst mit K_1, dann mit K_2 DES-verschlüsseln
- Problem: 2DES nicht wesentlich sicherer als DES

Meet-in-the-Middle-Angriff auf 2DES

- Erinnerung: $\mathrm{E}_{\mathrm{2DES}}(K, M) := \mathrm{E}_{\mathrm{DES}}(K_2, \mathrm{E}_{\mathrm{DES}}(K_1, M))$
- Gegeben: M, $C = \mathrm{E}_{\mathrm{2DES}}(K, M)$, gesucht: $K = (K_1, K_2)$
  1. Berechne alle Paare $(K_1', C_{K_1'} := \mathrm{E}_{\mathrm{DES}}(K_1', M))$ (für alle K_1')
  2. Sortiere Folge nach $C_{K_1'}$ lexikographisch ($\rightarrow$ binäre Suche)
  3. Berechne nacheinander $C_{K_2'} := \mathrm{D}_{\mathrm{DES}}(K_2', C)$
  4. Wenn $C_{K_2'} = C_{K_1'}$ für ein K_1', gib (K_1', K_2') aus
- Bei mehreren Kandidaten (K_1', K_2'): Suche mit neuen M, C
- Zeitaufwand $\mathbf{O}(56 \cdot 2^{56})$, Platzbedarf $64 \cdot 2^{56} + \varepsilon$ Bits

3DES

- DES zu unsicher, 2DES nicht so sicher wie erhofft
- Triple-DES (3DES)
  - $K := (K_1, K_2, K_3) \in (\{0,1\}^{56})^3$
  - $\mathrm{E}_{\mathrm{3DES}}(K, M) := \mathrm{E}_{\mathrm{DES}}(K_3, \mathrm{D}_{\mathrm{DES}}(K_2, \mathrm{E}_{\mathrm{DES}}(K_1, M)))$
  - Mit K_1 ver-, mit K_2 ent-, dann mit K_3 verschlüsseln
- Meet-in-the-Middle anwendbar, Aufwand $\mathbf{O}(2^{112})$
- Bessere (aber unpraktikable) Angriffe existieren
- Hauptgrund für Verwendung: benutzt DES als black box

Überblick

Advanced Encryption Standard (AES)

- Erinnerung: $\mathrm{E} : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$ zentral
- AES: Beispiel für E mit $k \in \{128, 192, 256\}$ und $\ell = 128$
- Entwickelt von Daemen und Rijmen, standardisiert 2000
- Keine Feistel-Struktur
- Nach heutigem Kenntnisstand sicher[2]

[2] Strukturelle, aber impraktikable Angriffe existieren

Überblick

Lineare Kryptoanalyse

- Grundidee: finde $\mathbb{F}_2$-lineare Abhängigkeiten zwischen den Bits von X und $Y := \mathrm{E}(K, X)$
  - Beispiel: $X_1 + X_7 + Y_3 + Y_8 + 1 = K_3 + K_{17} \bmod 2$
- Idealer Fall: K aus bekannten (X, Y)-Paaren herleitbar
- Bei Feistel-Verfahren (n Runden) indirekter Angriff möglich:
  1. Finde lineare Abhängigkeiten zwischen F-Ein- und -Ausgabe
  2. Erweitere Abhängigkeiten auf die ersten $n - 1$ Feistel-Runden
  3. Vollständige Suche über letzten Rundenschlüssel $K^{(n)}$...
  4. ...überprüfe $K^{(n)}$-Kandidaten mittels linearer Abhängigkeit
  5. Wenn $K^{(n)}$ gefunden, suche nach $K^{(n-1)}$, danach $K^{(n-2)}$, usw.
- Bricht FEAL, bei DES besser als vollständige Suche

  (benötigt aber riesige Anzahl an Klartext-Chiffrat-Paaren)

Differentielle Kryptoanalyse

- Grundidee: betrachte Ausgabedifferenzen $\Delta_{\text{out}} := Y \oplus Y'$ in Abhängigkeit von Eingabedifferenzen $\Delta_{\text{in}} := X \oplus X'$
- Bei bestimmten Eingabedifferenzen (z.B. von S-Boxen) manche Ausgabedifferenzen wahrscheinlicher als andere
- Bei Feistel-Verfahren Angriff ähnlich wie bei linearer Analyse:
  1. Finde wahrscheinliche Paare $\Delta_{\text{in}} \Rightarrow \Delta_{\text{out}}$ zwischen Eingabe und Ausgabe von vorletzter Runde
  2. Vollständige Suche über letzten Rundenschlüssel $K^{(n)}$...
  3. ... überprüfe $K^{(n)}$-Kandidaten auf $\Delta_{\text{in}} \Rightarrow \Delta_{\text{out}}$-Konsistenz
- DES resistent gegen differentielle Analyse, FEAL nicht