Erinnerung Blockchiffre

Definition schlüsselabhängige Permutation

Seien F, F^{-1} pt Algorithmen. F heißt *schlüsselabhängige Permutation* auf ℓ Bits falls

1. F berechnet eine Funktion $\{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^\ell$, so dass für alle $k \in \{0,1\}^n$ die Funktion $F_k(\cdot)$ eine Bijektion ist.
2. $F_k^{-1}(\cdot)$ berechnet die Umkehrfunktion von $F_k(\cdot)$.

Definition Starke Pseudozufallspermutation (Blockchiffre)

Sei F eine schlüsselabhängige Permutation auf ℓ Bits. Wir bezeichnen F als *starke Pseudozufallspermutation (Blockchiffre)*, falls für alle ppt D gilt

$$\left|\mathrm{Ws}[D^{F_k(\cdot),F_k^{-1}(\cdot)}(1^n)=1] - \mathrm{Ws}[D^{f(\cdot),f^{-1}(\cdot)}(1^n)=1]\right| \leq \mathrm{negl}(n),$$

mit $k \in_R \{0,1\}^n$ und $f \in_R Perm_\ell$.

Angriffe auf Blockchiffren

Angriffe: in aufsteigender Stärke

1. Ciphertext-only: $\mathcal{A}$ erhält $F_k(x_i)$ für unbekannte x_i.
2. Known plaintext: $\mathcal{A}$ erhält Paare $(x_i, F_k(x_i))$
3. Chosen plaintext: $\mathcal{A}$ wählt x_i und erhält $F_k(x_i)$.
4. Chosen ciphertext: $\mathcal{A}$ wählt x_i, y_i und erhält $F_k(x_i), F_k^{-1}(y_i)$.

Sicherheit:

- Jede Pseudozufallspermutation F ist CPA-sicher.
- Jede starke Pseudozufallspermutation F ist CCA-sicher.

Warnung:
Blockchiffren selbst sind **kein sicheres** Verschlüsselungsschema.

Design Prinzip: Konfusion und Diffusion

Ziel: Kleine Eingabedifferenzen erzeugen pseudozufällige Ausgaben.

Paradigma Konfusion und Diffusion

Rundeniterierte Vorgehensweise zur Konstruktion einer Blockchiffre

1. **Konfusion:** Permutiere kleine Bitblöcke schlüsselabhängig.
2. **Diffusion:** Permutiere alle Bits.

Bsp: F soll Blocklänge 128 Bits besitzen.

- Konfusion: Definiere schlüsselabhängige Permutation $S_1, \ldots, S_{16}$ auf 8 Bits. Sei $x = x_1 \ldots x_{16} \in (\{0,1\}^8)^{16}$. Definiere
$$F_k(x) = S_1(x_1) \ldots S_{16}(x_{16}).$$
- Diffusion: Permutiere die Bits von $F_k(x)$.
- Iteriere die obigen beiden Schritte hinreichend oft, damit kleine Eingabedifferenzen sich auf alle Ausgabebits auswirken.
- Beschreibungslänge von S_i: $8 \cdot 2^8$ Bits, F : $16 \cdot 8 \cdot 2^8 = 2^{15}$ Bits.
- Länge einer echten Zufallspermutation: $128 \cdot 2^{128} = 2^{135}$ Bits.

Substitutions-Permutations Netzwerk (SPN)

Szenario: Verwende einen Masterschlüssel k.

- Berechne aus dem Masterschlüssel k Rundenschlüssel $k_1, \ldots k_r$ mittels eines sogenannten Keyschedule-Algorithmus.
- Die Permutationsfunktionen $S_1, \ldots, S_m$ werden fest und schlüsselunabhängig gewählt (sogenannte S-Boxen).

Beschreibung Substitutions-Permutations Netzwerk (SPN)

EINGABE: $S_1, \ldots, S_m$, $k \in \{0,1\}^n$, x, ℓ, r

1. Berechne $k_1, \ldots, k_r \in \{0,1\}^\ell$ aus k. $y \leftarrow x$.
2. For $i \leftarrow 1$ to r
   1. **Schlüsseladdition:** $y \leftarrow y \oplus k_i$. Schreibe $y = y_1 \ldots y_m$.
   2. **Substitution per S-Boxen:** $y \leftarrow S_1(y_1) \ldots, S_m(y_m)$
   3. **Permutation:** $y \leftarrow$ Permutation der Bits von y.

AUSGABE: $F_k(x) := y$

Beobachtung: F ist invertierbar, da jeder Schritt invertierbar ist.

Lawineneffekt

Ziel: Veränderung in Eingabebit wirkt sich auf alle Ausgabebits aus.

Beobachtung Notwendige Eigenschaften für Lawineneffekt

1. **S-Box:** Ändern eines Eingabebits verändert ≥ 2 Ausgabebits.
2. **Permutation:** Ausgabebits einer S-Box werden zu Eingabebits verschiedener S-Boxen.

Beobachtung: Lawineneffekt

- Betrachten ein SPN mit 4 Bit S-Boxen und Blocklänge 128 Bit.
- 1-Bit Eingabedifferenz erzeugt mindestens eine 2-Bit Differenz.
- Eine 2-Bit Differenz resultiert in zwei 1-Bit Differenzen an verschiedenen S-Boxen in der nächsten Runde.
- Diese sorgen für mindestens 4-Bit Differenz, usw.
- D.h. jede Runde verdoppelt potentiell die beeinträchtigten Bits.
- Nach 7 Runden sind alle $2^7 = 128$ Bits von der Veränderung eines Eingabebits beeinträchtigt.

Angriff auf eine Runde eines SPN

Algorithmus Angriff auf eine Runde eines SPN

EINGABE: x, $y = F_k(x)$

1. $y :=$ Invertiere auf y die Permutation und die S-Boxen.
2. Berechne $k := x \oplus y$.

AUSGABE: k

Anmerkungen:

- Die Invertierung in Schritt 1 ist möglich, da sowohl die Permutation als auch die S-Boxen öffentlich sind.
- Nach Invertierung erhält man den Wert $x \oplus k$.

Distinguisher für 2 Runden

Bsp: Distinguisher für 2 Runden

- Wir betrachten Blocklänge 80 Bit und 4 Bit S-Boxen.
- Wähle x_i, die sich nur im ersten 4-Bit Block unterscheiden.
- Nach 1. Runde: Ausgaben unterscheiden sich in ≤ 4 Blöcken.
- Nach 2. Runde: Ausgaben unterscheiden sich in ≤ 16 Blöcken.
- D.h. nicht alle der 20 Ausgabeblöcke werden verändert.
- Können SPN leicht von Pseudozufallspermutation entscheiden.

DES - Data Encryption Standard

Beschreibung von DES:

- Entwickelt 1973 von IBM, standardisiert 1976.
- DES besitzt Schlüssellänge 56 Bit und Blocklänge 64 Bit.
- Besteht aus Feistelnetzwerk (siehe Vorlesung 8) mit 16 Runden.
- Aus Bits von k werden 48-Bit Schlüssel $k_1, \ldots, k_{16}$ ausgewählt.
- Rundenfunktionen f_i sind SPNs mit nicht invertierbaren S-Boxen.

Algorithmus Rundenfunktion f_i

EINGABE: $k_i, R_{i-1} \in \{0,1\}^{32}$

1. $y :=$ Erweitere R_{i-1} auf 48 Bit durch Verdopplung von 16 Bits.
2. $y := y \oplus k_i$
3. $y :=$ Splitte y in 6-Bit Blöcke $y_1 \ldots y_8$ auf. Wende auf jedes y_i eine S-Box $S_i : \{0,1\}^6 \to \{0,1\}^4$ an. Permutiere das Ergebnis.

AUSGABE: $f_i(R_{i-1}) := y$

Die DES S-Boxen

DES S-Boxen:

- Alle 8 S-Boxen realisieren verschiedene Abb. $\{0,1\}^6 \to \{0,1\}^4$.
- Jede S-Box ist eine 4:1-Abbildung.
- D.h. jede S-Box sendet genau 4 Eingaben auf eine Ausgabe.
- Wechsel eines Eingabebits ändert mindestens zwei Ausgabebits.

Lawineneffekt bei DES:

- Wähle (L_0, R_0) und (L'_0, R_0) mit 1-Bit Differenz in L_0, L'_0.
- (L_1, R_1) und (L'_1, R'_1) besitzen 1-Bit Differenz in R_1, R'_1.
- Durch f_2 erhält man mindestens eine 2-Bit Differenz in R_2, R'_2.
- D.h. (L_2, R_2) und (L'_2, R'_2) besitzen mind. eine 3-Bit Differenz.
- f_3 angewendet auf R_2, R'_2 liefert mind. eine 4-Bit Differenz, usw.
- Nach 8 Runden erreicht man volle Diffusion auf alle Ausgabebits.

Die (Un-)Sicherheit von DES

Sicherheit von DES:

- Bester praktischer Angriff ist noch immer die Brute-Force Suche.
- Die folgende Tabelle gibt eine Übersicht über DES Kryptanalysen.

Jahr	Projekt	Zeit
1997	DESCHALL, Internet	96 Tage
1998	distributed.net, Internet	41 Tage
1998	Deep Crack, 250.000 Dollar Maschine	2 Tage
2008	COPACOBANA, 10.000 Euro FPGAs	1 Tag

- Das Design von DES ist gut, nur die Schlüssellänge ist zu kurz.
- Die Blocklänge von 64 Bits von DES gilt als zu kurz.

Doppelte Verschlüsselung bringt wenig

Szenario: doppelte Verschlüsselung

- Sei F eine Blockchiffre mit Schlüssellänge n wie z.B. DES.
- Dann besitzt $F'_{k_1,k_2}(x) = F_{k_2}(F_{k_1}(x))$ Schlüssellänge $2n$.
- Leider liefert F' kein Sicherheitsniveau von 2^{2n}.

Algorithmus Meet-in-the-Middle Angriff auf doppelte Verschl.

EINGABE: (x_1, y_1), (x_2, y_2)

1. Für alle $k_1 \in \{0,1\}^n$, berechne $z := F_{k_1}(x_1)$. Speichere (z, k_1) in einer nach der ersten Komponente sortierten Liste L_1.
2. Für alle $k_2 \in \{0,1\}^n$, berechne $z := F^{-1}_{k_2}(y_1)$. Speichere (z, k_2) in einer nach der ersten Komponente sortierten Liste L_2.
3. Für alle z mit $(z, k_1) \in L_1$ und $(z, k_2) \in L_2$, speichere (k_1, k_2) in S.
4. Für alle $(k_1, k_2) \in S$: Verifiziere Korrektheit mittels (x_2, y_2).

AUSGABE: $k = (k_1, k_2)$

Doppelte Verschlüsselung bringt wenig

Korrektheit:

- Für korrektes (k_1, k_2) gilt $F_{k_2}(F_{k_1}(x)) = y$, d.h. $F_{k_1}(x) = F_{k_2}^{-1}(y)$.
- Ein falsches (k_1, k_2) erfüllt diese Identität mit Ws etwa 2^{-n}.
- D.h. wir erwarten $2^{2n} \cdot 2^{-n} = 2^n$ Elemente in der Menge S.
- Verifizieren mit (x_2, y_2) liefert erwartet den korrekten Schlüssel.

Laufzeit: Operationen auf einzelnen Schlüsseln zählen Zeit $\mathcal{O}(1)$.

- Schritt 1 und 2: jeweils Zeit und Platz $\mathcal{O}(n \cdot 2^n)$.
- Schritt 3: Laufzeit $\mathcal{O}(2^n)$ und Platz $\mathcal{O}(n \cdot 2^n)$.
- Schritt 4 lässt sich in Laufzeit $\mathcal{O}(2^n)$ realisieren.
- D.h. wir erhalten insgesamt Laufzeit und Platz $\mathcal{O}(n \cdot 2^n)$.
- Damit erhöht sich die Laufzeit gegenüber einem Brute-Force Angriff bei einfacher Verschlüsselung nicht wesentlich.

Dreifache Verschlüsselung

Szenario: dreifache Verschlüsselung

1. Variante 1: $F'_{k_1,k_2,k_3}(x) := F_{k_3}(F^{-1}_{k_2}(F_{k_1}(x)))$
2. Variante 2: $F'_{k_1,k_2}(x) := F_{k_1}(F^{-1}_{k_2}(F_{k_1}(x)))$

Grund des Alternierens von F, F^{-1}, F: Für die Wahl von $k_1 = k_2 = k_3$ erhalten wir eine einfache Anwendung von $F_{k_1}(x)$.

Sicherheit der 1. Variante:

- Meet-in-the-Middle Angriff wie zuvor in Zeit und Platz $\mathcal{O}(n \cdot 2^{2n})$.

Sicherheit der 2. Variante:

- Bekannter CPA-Angriff mit $\mathcal{O}(2^n)$ gewählten Paaren.
- Zeitkomplexität beträgt ebenfalls $\mathcal{O}(2^n)$.

Triple-DES:

- Beide Varianten von Triple-DES finden in der Praxis Verwendung.
- Löste 1999 DES als Standard ab. Trotz Standardisierung von AES im Jahr 2002 ist Triple-DES auch heute noch weitverbreitet.

AES - Advanced Encryption Standard

NIST Wettbewerb: (National Institute of Standard and Technology)

- Jan 1997: Aufruf zum Konstruktions-Wettbewerb einer Blockchiffre
- Ursprünglich 15 Kandidaten eingereicht.
- Aug 1999: Auswahl von fünf AES-Finalisten

  MARS, RC6, Rijndael, Serpent und Twofish.
- Okt 2000: Auswahl von Rijndael der Autoren Rijmen und Daemen.

Struktur von AES (Rijndael):

- AES ist ein SPN und besitzt Blocklänge 128.
- Schlüssel mit 128, 192 und 256 Bit können verwendet werden.
- Eingaben $x \in \{0,1\}^{128}$ werden rundenweise in einer 4×4-Byte Matrix, der sogenannten Zustandsmatrix, modifiziert.
- Anzahl Runden: 10 für 128-Bit k, 12 für 192-Bit und 14 für 256-Bit.

Die vier Rundenoperationen von AES

Operation 1: AddRoundKey

- Leite aus k einen Rundenschlüssel $k_i \in \{0,1\}^{128}$ ab.
- XOR der Zustandsmatrix mit k_i.

Operation 2: SubByte

- Interpretiere jedes Byte der Zustandsmatrix als Element $x \in \mathbb{F}_8$.
- Ersetze x durch x^{-1} in $\mathbb{F}_8$ und 0^8 durch 0^8.
- Wende eine affine Transformation auf die Zustandsbytes an.
- Man beachte: Dieselbe S-Box wird für alle Bytes verwendet.

Operation 3: ShiftRow

- Verschiebe die 4 Zeilen der 4×4-Zustandsmatrix zyklisch.
- Lasse die 1. Zeile unverändert.
- Verschiebe die 2. Zeile um eine Position nach links, die 3. Zeile um 2 nach links und die 3. Zeile um 3 Positionen nach links.

Die vier Rundenoperationen von AES

Operation 4: MixColumn

- Sei $a_{0,j}, a_{1,j}, a_{2,j}, a_{3,j}$ eine Spalte der Zustandsmatrix.
- Betrachte die Spalte als Element aus $\mathbb{F}_8/(x^4+1)$, d.h.

$$a_{0,j} + a_{1,j}x + a_{2,j}x^2 + a_{3,j}x^3 \text{ mit } a_{i,j} \in \mathbb{F}_8.$$

- Multipliziere mit $c(x) = 2 + x + x^2 + 3x^3 \in \mathbb{F}_8/(x^4+1)$.
- Kodieren $a \in \mathbb{F}_{2^8} = \mathbb{F}_2[y]/\pi$, π irreduzibel mit Grad 8, wie folgt: Sei z.B. $a = y^5 + y + 1$. Wir schreiben $a = (00100011) = 35$.
- MixColumn entspricht Multiplikation mit einer Matrix $C \in \mathbb{F}_8^{4\times 4}$.
- D.h. eine Spalte x wird mittels $x \to Cx$ linear abgebildet.
- Menge aller (x, Cx) definiert einen linearen Code mit Distanz 5.
- D.h. unterscheiden sich zwei Spalten x, x' in nur einer Position, so unterscheiden sie sich nach MixColumn in allen 4 Positionen.
- Diese Eigenschaft führt zu einer schnellen Diffusion bei AES.