Aktueller Überblick

§0 Einführende Worte (✓)

§1 Geschichtlicher Überblick (✓)

§2 Zufall (✓)

§3 Perfekte Sicherheit und ihre Grenzen (✓)

§4 Angriffsszenarien (✓)

§5 Der komplexitätstheoretische Ansatz(✓)

§6 Pseudozufallsgeneratoren und Stromchiffren

§7 Pseudozufallsfunktionen und Blockchiffren

§8 ...

§6 Pseudozufallsgeneratoren und Stromchiffren

(IND-EAV-sichere Verschlüsselungssysteme)

Pseudozufallsgeneratoren

Bit-Generatoren

Definition. (C.D.) Ein (deterministischer) Bit-Generator (mit Sicherheitsparameter) ist ein deterministischer Algorithmus $\mathcal{G}$, der unter Eingabe von $s \in \{0,1\}^n$

- entweder einen String einer festen Länge $\ell = \ell(n) > n$ ausgibt und terminiert
- oder ein Bit nach dem anderen ausgibt und nicht terminiert ($\ell = \infty$)

wobei die Laufzeit immer polynomiell in der aktuellen Ausgabelänge ist, sobald diese mindestens n ist.

Pseudozufallsgeneratoren

Wir wollen:

Die Ausgabe so eines Generators ist in einem komplexitätstheoretischen Sinn ununterscheidbar von einem uniform zufälligen String.

Einen Algorithmus mit dem Zweck einer Unterscheidung heißt Unterscheider (distinguisher).

Typische Bezeichnung: $\mathcal{D}$

Ausgabe: 0/1 für "uniform zufällig" / "nicht uniform zufällig" = "vom Generator erzeugt"

Pseudozufallsgeneratoren

Es sei $\mathcal{G}$ ein Bit-Generator.

Sei zunächst die Ausgabelänge $\ell = \ell(n)$ endlich.

Es sei $\mathcal{D}$ ein Unterscheider.

Wir definieren den Vorteil oder Erfolg von $\mathcal{D}$ für Sicherheitsparameter n als

$$\mathbf{P}[\mathcal{D}(1^n, \mathcal{G}(s)) = 1] - \mathbf{P}[\mathcal{D}(1^n, r) = 1] \,,$$

wobei s in $\{0,1\}^n$ und r in $\{0,1\}^\ell$ unform sind.

Pseudozufallsspiel

Eingabe: Sicherheitsparameter 1^n

1. Unterscheider und Herausforderer erhalten 1^n.
2. Der Herausforder wählt $i \in \{0,1\}$ unform zufällig
   - Wenn $i = 0$, wählt der Herausforderer einen uniform zufälligen String $w \in \{0,1\}^\ell$ uniform zufällig.
   - Wenn $i = 1$, wählt er $s \in \{0,1\}^n$ uniform zufällig und berechnet $w := \mathcal{G}(s)$.

   Er schickt w an den Unterscheider.
3. Der Unterscheider entscheidet sich für $j \in \{1,2\}$.

Der Unterscheider hat gewonnen, wenn $i = j$ ist.

Wir sollten erhalten: Der Erfolg / Vorteil von $\mathcal{D}$ (wie zuvor definiert) ist gleich dem im Spiel:

$$2 \cdot (\mathbf{P}[\mathcal{D} \text{ gewinnt}] - \frac{1}{2}) \, .$$

Zum Pseudozufallsspiel

Es ist

$$
\begin{aligned}
& \text{Erfolg von } \mathcal{D} \text{ im Spiel} \\
= \;& 2 \cdot (\mathbf{P}[\mathcal{D} \text{ gewinnt}] - \tfrac{1}{2}) \\
= \;& 2 \cdot (\mathbf{P}[\mathcal{D} \text{ gewinnt} \mid i = 1] \cdot \tfrac{1}{2} + \mathbf{P}[\mathcal{D} \text{ gewinnt} \mid i = 0] \cdot \tfrac{1}{2} - \tfrac{1}{2}) \\
= \;& 2 \cdot (\mathbf{P}[\, j = 1 \mid i = 1] \cdot \tfrac{1}{2} + \mathbf{P}[\, j = 0 \mid i = 0] \cdot \tfrac{1}{2} - \tfrac{1}{2}) \\
= \;& \mathbf{P}[\, j = 1 \mid i = 1] + \mathbf{P}[\, j = 0 \mid i = 0] - 1 \\
= \;& \mathbf{P}[\, j = 1 \mid i = 1] - \mathbf{P}[\, j = 1 \mid i = 0] \\
= \;& \mathbf{P}[\mathcal{D}(\mathcal{G}(s)) = 1] - \mathbf{P}[\mathcal{D}(r) = 1] \\
= \;& \text{Erfolg von } \mathcal{D} \text{ wie definiert}
\end{aligned}
$$

Stimmt also!

Zum Pseudozufallsspiel

Man kann den Erfolg von $\mathcal{D}$ auch so formulieren:

$$
\begin{aligned}
& \text{Erfolg von } \mathcal{D} \\
= \;& \text{Erfolg von } \mathcal{D} \text{ im Spiel} \\
= \;& 2 \cdot (\mathbf{P}[\mathcal{D} \text{ gewinnt}] - \tfrac{1}{2}) \\
= \;& 2 \cdot (\mathbf{P}[\mathcal{D} \text{ gewinnt} \mid i = 1] \cdot \tfrac{1}{2} + \mathbf{P}[\mathcal{D} \text{ gewinnt} \mid i = 0] \cdot \tfrac{1}{2} - \tfrac{1}{2}) \\
= \;& 2 \cdot (\mathbf{P}[\mathcal{D} \text{ gewinnt} \mid i = 1] - \tfrac{1}{2}) \cdot \tfrac{1}{2} \\
+ \;& 2 \cdot (\mathbf{P}[\mathcal{D} \text{ gewinnt} \mid i = 0] - \tfrac{1}{2}) \cdot \tfrac{1}{2} \\
= \;& \big(\text{Erfolg von } \mathcal{D} \text{ unter der Bedingung,} \\
& \quad \text{dass der Herausforderer } i = 0 \text{ (“uniform zufällig”) wählt}\big) \cdot \tfrac{1}{2} \\
+ \;& \big(\text{Erfolg von } \mathcal{D} \text{ unter der Bedingung,} \\
& \quad \text{dass der Herausforderer } i = 1 \text{ (“nicht u. z.”) wählt}\big) \cdot \tfrac{1}{2}
\end{aligned}
$$

Pseudozufallsgeneratoren

Standarddefinition

Ein Pseudozufallsgenerator ist ein Bit-Generator mit fester Ausgabelänge $\ell = \ell(n) = \text{Poly}(n)$ derart, dass für alle PPT-Unterscheider $\mathcal{D}$ der Erfolg vernachlässigbar in n ist.

Pseudozufallsspiel allgemein (C.D.)

Es sei nun ein Generator mit beliebiger Ausgabelänge gegeben.

Eingabe: Sicherheitsparameter 1^n

1. Unterscheider und Herausforderer erhalten 1^n.
2. Der Herausforder wählt $i \in \{0, 1\}$ unform zufällig
   Es sei $w = w_1 w_2 \ldots$ ein Bit-Strom der Länge ℓ (vielleicht ∞) mit:
   - Wenn $i = 0$ sind die w_i uniform und unabhängig zufällig
   - Wenn $i = 1$, ist $w = \mathcal{G}(s)$
3. Der Unterscheider kann vom Herausforderer immer wieder ein "nächstes Bit" von w anfordern (bis w verbraucht ist).
   Der Unterscheider entscheidet sich für $j \in \{1, 2\}$.

Der Unterscheider hat gewonnen, wenn $i = j$ ist.

Pseudozufallsgeneratoren

Definition. (C.D.) Es sei $\mathcal{G}$ ein Bit-Generator mit beliebiger Ausgabelänge (fest oder potentiell unendlich).

Dann heißt $\mathcal{G}$ Pseudozufallsgenerator, falls für alle PPT-Unterscheider im Pseudozufallsspiel der Erfolg vernachlässigbar in n ist.

Alternative Beschreibung. Für alle Algorithmen $\mathcal{D}$ mit Laufzeit polynomiell in n ist

$$\mathbf{P}[\mathcal{D}(1^n, \mathcal{G}(s)) = 1] - \mathbf{P}[\mathcal{D}(1^n, r) = 1] \,,$$

wobei s in $\{0,1\}^n$ uniform zufällig ist und r gleiche Länge wie $\mathcal{G}(s)$ hat mit r_i unabhängig unform zufällig in $\{0,1\}$, vernachlässigbar (in n).

[$\mathcal{G}(s)$ und r müssen nicht vollständig gelesen werden!]

Stromchiffren zu Pseudozufallsgeneratoren

Eine einfache Konstruktion

Es sei $\mathcal{G}$ ein Bit-Generator mit Ausgabelänge $\ell = \ell(n)$.

(Einfacher) Stromchffrierer zu $\mathcal{G}$:

- Schlüsselgenerator $\mathcal{G}en$
  Eingabe: 1^n mit n Sicherheitsparameter
  Ausgabe: $k = s \in \{0,1\}^n$ unform
- Verschlüsselungsalgorithmus $\mathcal{E}nc$
  Eingabe: Schlüssel k, Nachricht m mit Länge $\leq \ell$.
  Ausgabe: Chiffriertext c wie folgt:
  Sei $\mathcal{G}(s) = o = o_1 o_2 \ldots$.
  Dann: $c_i = o_i \oplus m_i$.
- Entschlüsselungsalgorithmus $\mathcal{D}ec$:
  Ebenso

Pseudozufallsgeneratoren und ihre Stromchiffren

Satz. Es sei $\mathcal{G}$ ein Bit-Generator. Dann ist die Stromchiffre zu $\mathcal{G}$ genau dann IND-EAV-sicher, wenn $\mathcal{G}$ ein Pseudozufallsgenerator ist.

Dann kann man von einem pseudo-one-time-Pad sprechen ...

Transformation "Angreifer $\rightarrow$ Unterscheider"

Es sei $\mathcal{A}$ ein Angreifer im Ununterscheidbarkeitsspiel für Lauscher für die Stromchiffre.

Es sei $\epsilon = \epsilon(n)$ der Erfolg.

Beschreibung eines Unterscheiders $\mathcal{D}$:

Eingabe: $1^n, w = w_1 w_2 \ldots$.

[Hier sind entweder die w_i unabhängig uniform zufällig in $\{0,1\}$ oder $w = \mathcal{G}(s)$, s uniform zufällig in $\{0,1\}^n$.]

Führe den Test für den Lauscher-Angriff für Variante des one-time-Pad mit String w (statt uniform zufälligem String) durch.

Das heißt:

1. Führe das Spiel durch, wobei kein Schlüssel gewählt wird, sondern stattdessen w benutzt wird.
2. - Wenn $\mathcal{A}$ gewonnen hat, gib 1 aus.
   - Wenn $\mathcal{A}$ verloren hat, gib 0 aus.

Transformation "Angreifer $\rightarrow$ Unterscheider"

Unterscheider $\mathcal{D}$:

Eingabe: $1^n, w_1 w_2 \ldots$

1. Gib $w_1 w_2 \ldots$ als Schlüssel an den Angreifer.
2. Wähle "Nachrichten" m_1, m_2 durch Angreifer $\mathcal{A}$
3. Wähle "Chiffriertext" wie der Herausforderer:
   Wähle $i \in \{1, 2\}$ uniform zufällig,
   setze $m := m_i$,
   berechne "Chiffriertext" c mit $c_{i'} = m_{i'} \oplus w_{i'}$.
   Gib c an den Angreifer.
4. Berechne die "Wahl" j von $\mathcal{A}$.
5. Wenn $i = j$ ist ($\mathcal{A}$ hat gewonnen), gib 1 aus,
   wenn $i = 1 - j$ ist ($\mathcal{A}$ hat verloren), gib 0 aus.

Transformation "Angreifer $\rightarrow$ Unterscheider"

Wir betrachten den Erfolg von $\mathcal{D}$ im Pseudozufallsspiel:

Wenn $i = 0$, d.h. $w = r$ ist "wirklich zufällig":

- Man kann sagen: Es wird ein Angriff gegen das one-time-Pad durchgeführt.
- $\mathbf{P}[\mathcal{D}(1^n, w) = 1] =$
  $\mathbf{P}[\mathcal{A}$ gewinnt im Spiel für das one-time-Pad$]$
- Erfolg von $\mathcal{D}$ (unter der geg. Bed.) $=$
  Erfolg von $\mathcal{A}$ im Spiel für das one-time-Pad
  $= 0$.

Transformation "Angreifer $\rightarrow$ Unterscheider"

Wir betrachten den Erfolg von $\mathcal{D}$ im Pseudozufallsspiel:

Wenn $i = 1$, d.h. $w = \mathcal{G}(s)$:

- Man kann sagen: Es wird ein Angriff gegen das Verschlüsselungsverfahren zu $\mathcal{G}$ durchgeführt
- $\mathbf{P}[\mathcal{D}(1^n, w) = 1] =$
  $\mathbf{P}[\mathcal{A}$ gewinnt im Spiel für das Verschlüsselungsverfahren$]$
- Erfolg von $\mathcal{D}$ (unter der geg. Bed.) $=$
  Erfolg von $\mathcal{A}$ im Spiel für das Verschlüsselungsverfahren zu $\mathcal{G}$
  $= \epsilon$.

Insgesamt:

$\mathcal{D}$ hat Erfolg $0 \cdot \frac{1}{2} + \epsilon \cdot \frac{1}{2} = \frac{\epsilon}{2}$.

Transformation "Unterscheider $\rightarrow$ Angreifer"

[Für Sicherheitsresultat irrelevant.]

Es sei $\mathcal{D}$ ein Unterscheider.

Sei ℓ die durch $\mathcal{D}$ maximal angeforderte Anzahl von Bits.

($\ell = \text{Poly}(n)$)

Beschreibung eines Angreifer $\mathcal{A}$:

- Wähle m_1 unform von Länge ℓ.
- Wähle $m_2 = 0^\ell$ wie im Unterscheider.

Nach Interaktion mit Herausforderer:

- Wenn $\mathcal{D}(c) = 0$ gib $j = 1$ aus.
- Wenn $\mathcal{D}(c) = 1$, gib $j = 2$ aus.

Transformation "Unterscheider $\rightarrow$ Angreifer"

Beschreibung eines Angreifer $\mathcal{A}$:

- Wähle m_1 unform von Länge ℓ.
- Wähle $m_2 = 0^\ell$ wie im Unterscheider.

Nach Interaktion mit Herausforderer:

- Wenn $\mathcal{D}(c) = 0$ gib $j = 1$ aus.
- Wenn $\mathcal{D}(c) = 1$, gib $j = 2$ aus.

Beachte:

Der Angreifer erhält von Herausforderer:

- Wenn $i = 1$: c in $\{0,1\}^\ell$ uniform
- Wenn $i = 2$: $c =$ die ersten ℓ Bits von $\mathcal{G}(s)$.

Transformation "Unterscheider $\rightarrow$ Angreifer"

Analyse des Angreifers:

Der Erfolg von $\mathcal{A}$ ist

$$
\begin{aligned}
& 2 \cdot (\mathbf{P}[j = 1 \mid i = 1] \cdot \tfrac{1}{2} + \mathbf{P}[j = 2 \mid i = 2] \cdot \tfrac{1}{2} - \tfrac{1}{2}) \\
= \quad & \mathbf{P}[j = 1 \mid i = 1] + \mathbf{P}[j = 2 \mid i = 2] - 1 \\
= \quad & \mathbf{P}[j = 2 \mid i = 2] - \mathbf{P}[j = 2 \mid i = 1] \\
= \quad & \mathbf{P}[\mathcal{D}(1^n, \mathcal{G}(s)) = 1] - \mathbf{P}[\mathcal{D}(1^n, r) = 1] \\
= \quad & \text{Erfolg von } \mathcal{D}
\end{aligned}
$$

Konkretes Sicherheitsresultat

Es sei $\mathcal{G}$ ein Bit-Generator.

[Vielleicht auch einer ohne Sicherheitsparameter – wie in der Praxis.]

- Ein Angreifer gegen die Stromchiffre zu $\mathcal{G}$ im Ununterscheidbarkeitsspiel für den Lauscherangriff mit Erfolg ϵ kann transformiert werden
  in einen Unterscheider für $\mathcal{G}$ mit "vergleichbarer" Laufzeit und Erfolg $\epsilon/2$.
- Ein Unterscheider für $\mathcal{G}$
  kann transformiert werden
  in einen Angreifer gegen die Stromchiffre zu $\mathcal{G}$ im Ununterscheidbarkeitsspiel für den Lauscherangriff mit "vergleichbarer" Laufzeit und gleichem Erfolg.

[Das ist natürlich etwas ungenau; man könnte versuchen, das zu verbessern.]

Exansion der Ausgabe

Exansion der Ausgabe

Man kann aus einem “minimalen” Pseudozufallsgenerator einen mit beliebig langer Ausgabe erhalten:

Es sei $\mathcal{G}$ ein Bit-Generator mit Ausgabelänge $\ell = n + 1$.

Wir setzen $\mathcal{G}(s) = (\mathcal{O}(s) \,|\, \mathcal{I}(s))$ mit $|\mathcal{I}(s)| = n$, $\mathcal{O}(s)$ ein Bit.

Hiermit betrachten wir:

Generator $\mathcal{G}^*$:

Eingabe: $s \in \{0,1\}^*$.

Wiederhole:

Gib $\mathcal{O}(s)$ aus.
Setze $s := \mathcal{I}(s)$.

Expansion der Ausgabe

Satz. Es sei $\mathcal{G}$ ein Bit-Generator mit Ausgabelänge $n+1$ (wobei n der Sicherheitsparameter ist).

Wenn $\mathcal{G}$ ein Pseudozufallsgenerator ist, dann ist auch $\mathcal{G}^*$ einer.

Beginn des Beweises

Transformation "Unterscheider für $\mathcal{G}^*$ $\rightarrow$ Unterscheider für $\mathcal{G}$"

Es sei $\mathcal{D}$ ein Unterscheider für $\mathcal{G}^*$.

Es sei $\epsilon = \epsilon(n)$ der Erfolg / Vorteil von $\mathcal{D}$.

Es ist also

$$\mathbf{P}[\mathcal{D}(\mathcal{G}^*(s)) = 1] - \mathbf{P}[\mathcal{D}(r) = 1] = \epsilon$$

(mit s und r wie üblich.)

Es sei $\mathcal{G}(s) = o = o_1 o_2 \ldots$ die Ausgabe von $\mathcal{G}^*$ (immer noch s in $\{0,1\}$ uniform).

“Interpolation” / “Hybrid-Argument”

Wir haben also:

$$\mathbf{P}[\mathcal{D}(o_1 o_2 \ldots) = 1] - \mathbf{P}[\mathcal{D}(r_1 r_2 \ldots) = 1] = \epsilon$$

Idee

Sei z das Maximum der von $\mathcal{D}$ betrachteten Bits der Eingabe.

Wir “interpolieren” zwischen den Strings

$o_1 o_2 \ldots o_z$ und $r_1 r_2 \ldots r_z$:

$$\begin{array}{lllllll}
q_0 & := & o_1 & o_2 & \cdots & \cdots & o_z \\
q_1 & := & r_1 & o_1 & \cdots & \cdots & o_{z-1} \\
q_2 & := & r_1 & r_2 & o_1 & \cdots & o_{z-2} \\
\cdots & & \cdots & \cdots & \cdots & \cdots & \cdots \\
q_{z-1} & := & r_1 & r_2 & \cdots & \cdots & o_1 \\
q_z & := & r_1 & r_2 & \cdots & \cdots & r_z
\end{array}$$

"Interpolation" / "Hybrid-Argument"

Es ist

$$
\begin{aligned}
\epsilon &= \mathbf{P}[\mathcal{D}(o_1 o_2 \dots o_z) = 1] - \mathbf{P}[\mathcal{D}(r_1 r_2 \dots r_z) = 1] \\
&= \mathbf{P}[\mathcal{D}(q_0) = 1] - \mathbf{P}[\mathcal{D}(q_z) = 1] \\
&= \mathbf{P}[\mathcal{D}(q_0) = 1] - \mathbf{P}[\mathcal{D}(q_1) = 1] \\
&+ \mathbf{P}[\mathcal{D}(q_1) = 1] - \mathbf{P}[\mathcal{D}(q_2) = 1] \\
&+ \mathbf{P}[\mathcal{D}(q_2) = 1] - \mathbf{P}[\mathcal{D}(q_3) = 1] \\
&\qquad \dots \qquad\qquad \dots \\
&+ \mathbf{P}[\mathcal{D}(q_{z-1}) = 1] - \mathbf{P}[\mathcal{D}(q_z) = 1] \\
&= \sum_{k=0}^{z-1} \mathbf{P}[\mathcal{D}(q_k) = 1] - \sum_{k=0}^{z-1} \mathbf{P}[\mathcal{D}(q_{k+1}) = 1]
\end{aligned}
$$

Betrachte nun k als uniform zufällig in $\{0, z-1\}$ (und unabhängig vom sonstigen Zufall).

Zufälliger "Zwischenstring"

$$\begin{aligned}
\epsilon &= \sum_{k^{(0)}=0}^{z-1} \mathbf{P}[\mathcal{D}(q_{k^{(0)}}) = 1] - \sum_{k^{(0)}=0}^{z-1} \mathbf{P}[\mathcal{D}(q_{k^{(0)}+1}) = 1] \\
&= \sum_{k^{(0)}=0}^{z-1} \mathbf{P}[\mathcal{D}(q^{(0)}) = 1 \mid k = k^{(0)}] - \\
&\qquad \sum_{k^{(0)}=0}^{z-1} \mathbf{P}[\mathcal{D}(q_{k+1}) = 1 \mid k = k^{(0)}]
\end{aligned}$$

Multiplikation mit $\mathbf{P}[k = k^{(0)}] = \frac{1}{z}$ ...

Zufälliger “Zwischenstring”

$$\begin{aligned}\frac{\epsilon}{z} &= \sum_{k^{(0)}=0}^{z-1} \mathbf{P}[\mathcal{D}(q^{(0)}) = 1 \mid k = k^{(0)}] \cdot \mathbf{P}[k = k^{(0)}] - \\ &\qquad \sum_{k^{(0)}=0}^{z-1} \mathbf{P}[\mathcal{D}(q_{k+1} = 1 \mid k = k^{(0)}] \cdot \mathbf{P}[k = k^{(0)}] \\ &= \mathbf{P}[\mathcal{D}(q_k) = 1] - \mathbf{P}[\mathcal{D}(q_{k+1}) = 1]\end{aligned}$$

mit k uniform zufällig in $\{0, z-1\}$ (und unabhängig vom sonstigen Zufall).

Unterscheider für $\mathcal{G}$

Es sei wieder $\mathcal{D}$ der Unterscheider für $\mathcal{G}^*$.

Es sei z eine obere Schranke an die Anzahl der gewünschten Bit.

Der Unterscheider $\mathcal{D}'$ für $\mathcal{G}$:

Unterscheider für $\mathcal{G}$

Eingabe: $w \in \{0,1\}^{n+1}$

[w soll entweder uniform verteilt sein oder es soll $w = \mathcal{G}(s)$ mit s uniform in $\{0,1\}^n$ sein.]

1. Wähle k in $\{0, z-1\}$ uniform.
2. Wähle $u_1, \ldots, u_k \in \{0,1\}^k$ uniform zufällig.
3. Lasse $\mathcal{G}^*$ auf Eingabe $w_2 \ldots w_{k-1}$ laufen; berechne den Ausgabestrom bis zum $(z-k-1)$-ten Bit: $a_1 \ldots a_{z-k-1}$.
4. Gib $\mathcal{D}(u_1 \ldots u_k w_1 a_1 \ldots a_{z-k-1})$ aus.

$$\mathcal{D}'(w_1 \ldots w_{n+1}) = \mathcal{D}(u_1 \ldots u_k w_1 a_1 \ldots a_{z-k-1})$$

mit k in $\{0, \ldots, z-1\}$ unform.

Analyse des Unterscheiders für $\mathcal{G}$

Beschreibung des Strings $u_1 \dots u_k w_1 a_1 \dots a_{z-k-1}$:

- Es sei $w = r$ uniform in $\{0,1\}^{n+1}$.
  $\mathcal{G}^*$ geht mit uniformem Seed los;
  $w_1 = r_1$ ist unform und unabhängig von den anderen Variablen.
  String kann so beschrieben werden:

$$u_1 \dots u_{k+1} o_1 \dots o_{z-k-1} = q_{k+1}$$

- Es sei $w = \mathcal{G}(s)$, s uniform in $\{0,1\}^n$.
  $\mathcal{G}^*$ geht mit Zustand $\mathcal{G}(s)$ los,
  das ist der Zustand nach einer Interation angewandt auf s;
  $w_1 = o_1$.
  String kann so beschrieben werden:

$$u_1 \dots u_k o_1 o_2 \dots o_{z-k} = q_k \ .$$

Analyse des Unterscheiders für $\mathcal{G}$

Beschreibung des Strings $u_1 \dots u_k w_1 a_1 \dots a_{z-k-1}$:

- Es sei $w = r$ uniform in $\{0,1\}^{n+1}$.
  String kann so beschrieben werden:

$$u_1 \dots u_{k+1} o_1 \dots o_{z-k-1} = q_{k+1}$$

- Es sei $w = \mathcal{G}(s)$, s uniform in $\{0,1\}^n$.
  String kann so beschrieben werden:

$$u_1 \dots u_k o_1 o_2 \dots o_{z-k} = q_k \ .$$

Somit:

$$\text{Erfolg von } \mathcal{D}' =$$

$$\mathbf{P}[\mathcal{D}'(1^n, \mathcal{G}(s)) = 1] - \mathbf{P}[\mathcal{D}(1^n, r) = 1] =$$

$$\mathbf{P}[\mathcal{D}(1^n, q_k) = 1] - \mathbf{P}[\mathcal{D}(1^n, q_{k+1}) = 1] = \frac{\epsilon}{z}$$

Sicherheitsresultate

Ein Unterscheider $\mathcal{D}$ für $\mathcal{G}^*$ mit Erfolg ϵ, der bis zu z Bits verlangt.

kann transformiert werden

in einen Unterscheider $\mathcal{D}'$ von $\mathcal{G}$ mit "vergleichbarer" Laufzeit und Erfolg

$$\frac{\epsilon}{z} .$$

Dies kann wieder als praktisches Resultat interpretiert werden, auch ohne Sicherheitsparameter.

Ende des Beweises

Alles im Polynomzeit-Paradigma:

Es werden PPT-Unterscheider betrachtet. Die Laufzeiten sind polynomiell in n.

Was wir haben

Wenn es einen PPT-Unterscheider gegen $\mathcal{G}^*$ mit Laufzeit $t = t(n)$ und Erfolg $\epsilon = \epsilon(n)$ gibt,

dann gibt es auch einen PPT-Unterscheider gegen $\mathcal{G}$ mit Laufzeit Poly(t) und Erfolg $\frac{\epsilon}{t}$.

Wenn ϵ nicht-vernachlässigbar ist, dann ist es $\frac{\epsilon}{t}$ auch nicht.

Folgerungen

Wenn $\mathcal{G}^*$ kein Pseudozufallsgenerator ist, dann ist es $\mathcal{G}$ auch nicht.

Wenn $\mathcal{G}$ ein Pseudozufallsgenerator ist, dann $\mathcal{G}^*$ auch.

Vorwärts-Sicherheit

Der Generator $\mathcal{G}^*$ kann als Bit-Generator mit Zustand aufgefasst werden.

[Man kann formal definieren, was ein Bit-Generator mit Zustand sein soll ...]

Nach einer Eingabe $s \in \{0,1\}^n$ uniform haben wir den Bit-Strom $o_1 o_2 \ldots$ der Ausgabe, d.h. die Folge

$$(o_1, o_2, \ldots)$$

und die Folge der Zustände

$$(s_0, s_1, \ldots) \quad (s_0 = s)$$

Vorwärts-Sicherheit

Vorwärts-Sicherheit bedeutet:

Für z in Polynomzeit berechenbar aus 1^n kann in keiner effizienten Weise

$$(o_1, o_2, \ldots o_z, s_z)$$

nicht von einem uniform zufälligen String gleicher Länge unterschieden werden.

Vorwärts-Sicherheit

Das bedeutet ganz genau:

Alle Bit-Generatoren, die der folgenden Beschreibung genügen, sind Pseudozufallsgeneratoren:

Eingabe: $s \in \{0,1\}^*$.

[Sei $n = |s|$.]

1. Berechne $z \in \mathbf{N}$ [in polynomieller Zeit (in n)].
2. Für $i = 1, \ldots, z$:
   Gib $\mathcal{O}(s)$ aus.
   Setze $s := \mathcal{I}(s)$.
3. Gib s aus.

Vorwärts-Sicherheit

Satz. Es sei $\mathcal{G}$ ein Pseudozufallsgenerator mit Ausgabelänge $\ell = n + 1$. Dann ist $\mathcal{G}^*$ als Bit-Generator mit Zustand vorwärts-sicher.

Das "Hybrid-Argument" kann (leicht) hierauf übertragen werden.

(Hausaufgabe!)

Blocklänge

Wir sind bisher immer von einem Bit-Generator $\mathcal{G}$ mit Ausgabelänge $n+1$ ausgegangen.

Man kann "alles übertragen" auf Ausgabelängen $n+m$.

Dann gibt $\mathcal{G}^*$ m Bits "auf einmal" aus.

Definition von "Generator mit Zustand" sollte so verallgemeinert werden.

Dann gilt immer noch:

Wenn $\mathcal{G}$ ein Pseudozufallsgenerator mit endlicher Ausgabelänge ist, ist $\mathcal{G}^*$ ein vorwärts-sicherer Pseudozufallsgenerator.

(Hausaufgabe!)

So gut wie echter Zufall

So gut wie echter Zufall

Es sei $\mathcal{G}$ ein Pseudozufallsgenerator mit Ausgabelänge $\ell = \text{Poly}(n)$.

Dann ist also $\mathcal{G}(s)$ für s unform in $\{0,1\}^n$ (komplexitätstheoretisch) ununterscheidbar von r unform in $\{0,1\}^\ell$.

D.h. für alle PPT-Algorithmen $\mathcal{D}$ ist

$$\mathbf{P}[\mathcal{D}(\mathcal{G}(s)) = 1] - \mathbf{P}[\mathcal{D}(r) = 1]$$

vernachlässigbar (in n).

So gut wie echter Zufall

Es sei $\mathcal{B}$ ein PPT-Algorithmus. Dann ist auch $\mathcal{A}(\mathcal{G}(s))$ (komplexitätstheoretisch) unbunterscheidbar von r unform in $\{0,1\}^\ell$.

Denn:

Für alle PPT-Algorithmen $\mathcal{D}$ ist auch

$$\mathbf{P}[\mathcal{D}(\mathcal{A}(\mathcal{G}(s))) = 1] - \mathbf{P}[\mathcal{D}(\mathcal{A}(r)) = 1]$$

vernachlässigbar (in n).

So gut wie echter Zufall

Prinzip. Man kann getrost "echten Zufall" durch Pseudozufall ersetzen.

Z.B. kann man dies bei beliebigen Verschlüsselungssystemen machen.

Für Generatoren mit beliebiger Eingeabelänge (d.h. für festen Sicherheitsparameter beliebig) sollte man einen Generator mit beliebig langer Ausgabe verwenden.

Im Spiel für CPA-Sicherheit (und CCA / CCA2) auch.

Das heißt: Der Herausforderer

hat so einen Generator,

erhält zu Beginn je einen uniform zufälligen String in $\{0,1\}^n$,

ersetzen "echten Zufall" durch Ausgabe des Generators.

Das Spiel zum CPA

Gegeben: $\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec$.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $k := \mathcal{G}en(1^n)$.
3. Der Angreifer kann beliebige Nachrichten m mit dem Schlüssel k verschlüsseln lassen (d.h. $\mathcal{E}nc_k(m)$ berechnen lassen).
4. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
5. Der Herausforderer wählt $i \in \{1, 2\}$ uniform. Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
6. Der Angreifer kann beliebige Nachrichten mit dem Schlüssel k verschlüsseln lassen.
7. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.

Der Angreifer hat gewonnen, wenn $i = j$ ist.

Variante des Spiels zum CPA

Gegeben: $\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec, \mathcal{G}$.

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $k := \mathcal{G}en(1^n)$.
   Er wählt einen Bit-String $s \in \{0,1\}^n$ uniform zufällig.
   Der "interne Zufall" beim Verschlüsseln wird durch fortlaufende Ausgaben des Generators $\mathcal{G}$ ersetzt.
3. Der Angreifer kann beliebige Nachrichten m mit dem Schlüssel k verschlüsseln lassen (d.h. $\mathcal{E}nc_k(m)$ berechnen lassen).
4. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
5. Der Herausforderer wählt $i \in \{1,2\}$ uniform. Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
6. Der Angreifer kann beliebige Nachrichten mit dem Schlüssel k verschlüsseln lassen.
7. Der Angreifer entscheidet sich für $j \in \{1,2\}$.

So gut wie echter Zufall

Das macht keinen Unterschied, denn:

Es sei $\mathcal{A}$ ein Angreifer.

Wir haben einen PPT-Algorithmus ("Test") $\mathcal{T}$, der

- unter Eingabe von 1^n das Spiel mit Angreifer $\mathcal{A}$ spielt,
- ausgibt, ob $\mathcal{A}$ gewonnen hat (Ausgabe 1) oder verloren hat (Ausgabe 0).

Mit der Variante $\mathcal{A}^{ps}$ wird ein Teil des Zufalls von $\mathcal{A}$ durch $\mathcal{G}$ (angewandt aufen kürzen String) erzeugt.

Ergebnis: Die Ausgaben von $\mathcal{A}^{ps}$ sind ununterscheidbar von denen von $\mathcal{A}$ (bezüglich Sicherheitsparameter n).

So gut wie echter Zufall

Aufgabe 2

Es sei $\mathcal{G}$ ein Pseudozufallsgenerator, k polynomiell in n und $\leq \ell$ (=Ausgabelänge)

Zeigen Sie: Für alle PPT-Algorithmen $\mathcal{A}$ gilt mit $\mathcal{G}(s) = w_1 w_2 \dots w_k \dots$ (und s in $\{0,1\}^n$ uniform):

$$\mathbf{P}[\mathcal{A}(1^n, w_1 \dots w_{k-1}) = w_k] - \frac{1}{2}$$

ist vernachlässigbar (in n).

Also inhaltlich: Man kann das nächste Bit nicht effizient vorherberechnen.

Geben Sie auch ein konkretes Sicherheitsresultat an.

Strom- und Blockchiffren (in der Praxis)

Strom- versus Blockchiffren

"Stromchiffre" und "Blockchiffre" sind normalerweise praxisorientierte "ingenieurtechnische" Begriffe.

Grobe Idee

Stromchiffren verarbeiten Bit für Bit und haben einen Zustand

Blockchiffren verarbeiten Blöcke (jeden für sich).

(Aber ...)

Stromchiffren

Stromchiffren werden unterteilt in:

synchrone und selbst-synchronisierende Stromchiffren

Bei synchronen Stromchiffren wird der Schlüsselstrom unabhängig vom Nachrichtenstrom erzeugt,

üblicherweise mit XOR verknüpft
(pseudo-one-time-pad-Konstruktion).

Bei selbst-synchronisierenden Stromchiffren wird ein Schlüssel vereinbart, der Schlüsselstrom mittels vorheriger Ausgabebits und dem vereinbarten Schlüssel errechnet.

Vorteil von selbst-synchronisierenden Chiffren: Ein "Verschieben" des Schlüsselstroms gegen den Chiffriertextstrom wirkt sich nur kurz aus.

Möglicher Vorteil von synchronen Chiffren: Man kann den Schlüsselstrom in einer "verschlossenen Box" erzeugen.

Aber ...

1. Stromchiffren können auch mehr als ein Bit in einem "Takt" ausgeben.
2. Was macht man, wenn man mit einer Blockchiffre mehr als einen Block verschlüsseln will?

$$m_1 \dots m_r \mapsto \mathcal{E}nc_k(m_1) \dots \mathcal{E}nc_k(m_r) \quad ?$$

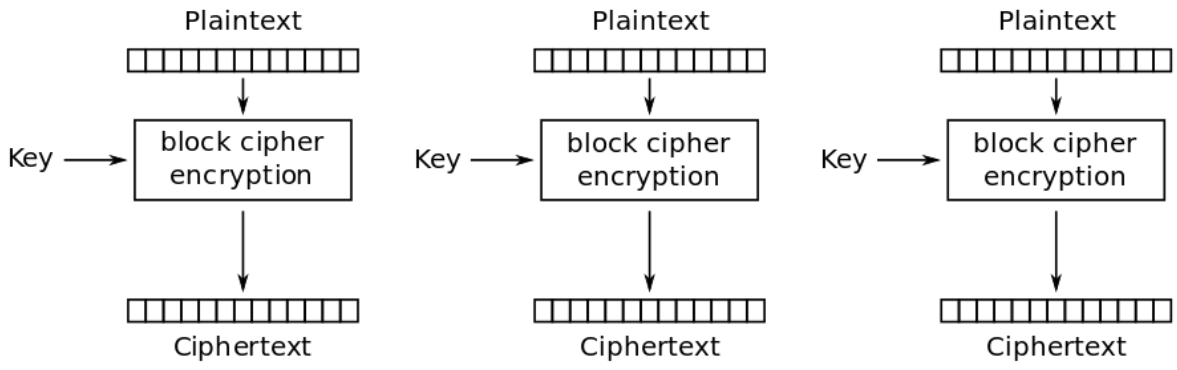
Ganz schlechte Idee!

Man benutzt sogenannte Modi.

Electronic Codebook Mode (ECB)

Trivialer Modus:

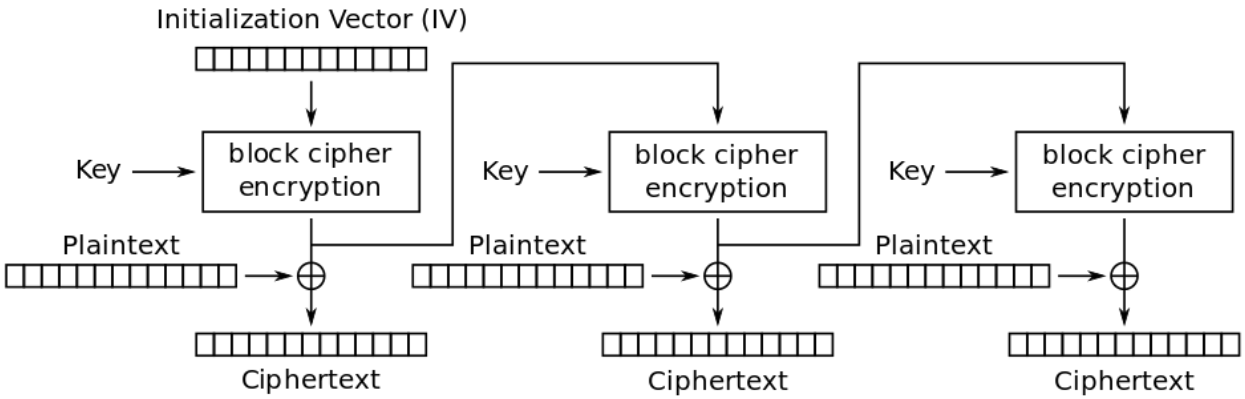
Es wird immer wieder “ganz einfach” derselbe Schlüssel verwendet.



Electronic Codebook (ECB) mode encryption

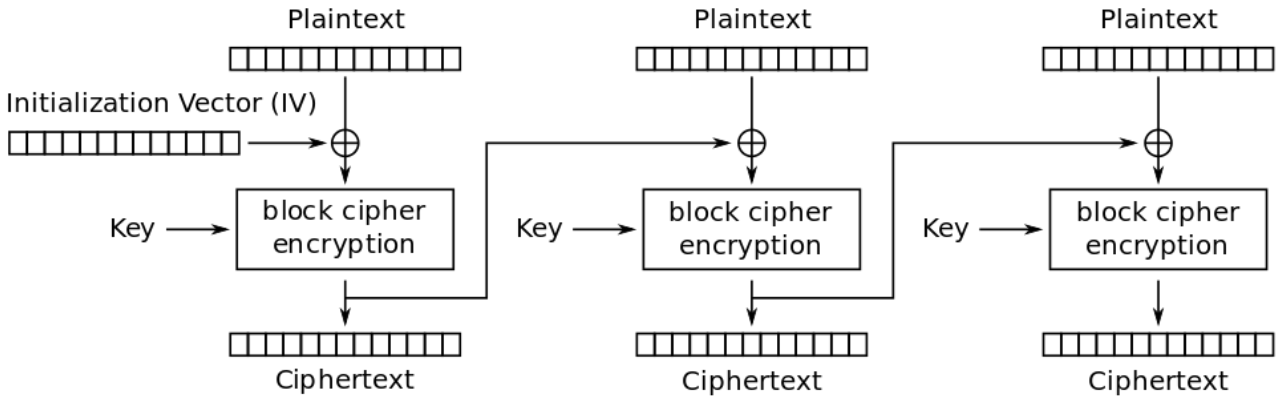
Total unsicher!

Output Feedback Mode (OFB)



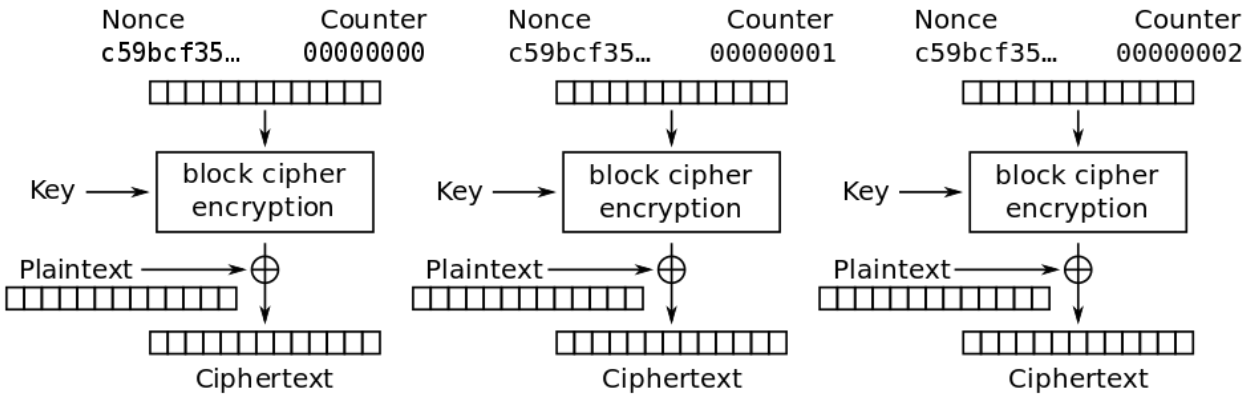
Output Feedback (OFB) mode encryption

Cipher Block Chaining Mode (CBC)



Cipher Block Chaining (CBC) mode encryption

Counter Mode (CTR)



Counter (CTR) mode encryption

Graphiken. Wikipedia, Benutzer WhiteTimberwolf

Strom- versus Blockchiffren

Idee

Stromchiffren verarbeiten Bit für Bit mit Zustand

Blockchiffren verarbeiten Blöcke (ohne Zustand)

Aber:

Stromchiffren können auch mehr als ein Bit in einem Takt verarbeiten

Wenn man mit einer Blockchiffre längere Nachrichten verarbeiten will, benutzt man Modi, z.B. OFB oder CRT, erhält Stromchiffren auf Blockbasis.

Strom- versus Blockchiffren

In der Praxis gibt es nichtsdestoweniger einen erheblichen Unterschied zwischen Strom- und Blockchiffren.

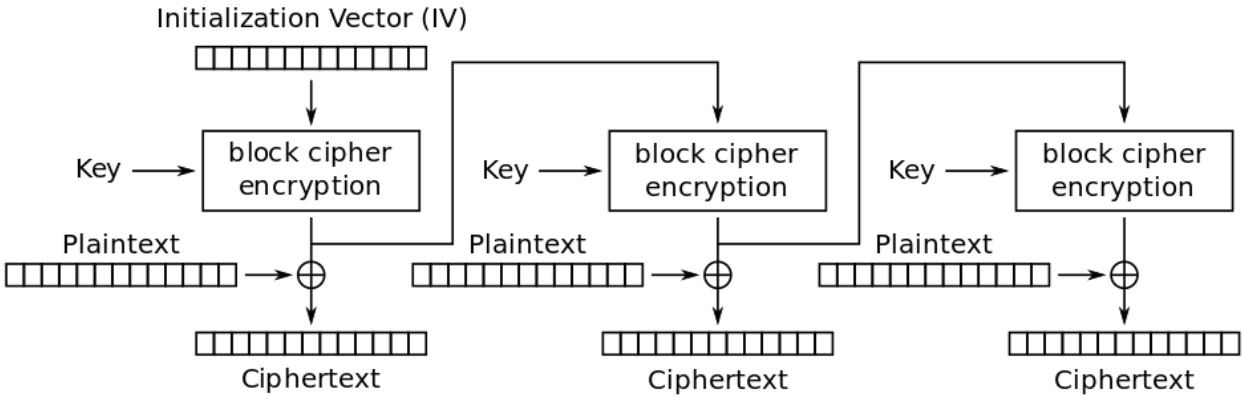
Stromchiffren sollten besonders schnell sein, haben Vorteil wenn weniger als ein Block gesendet werden soll.

Bauteile von Stromchiffren sind typischerweise:

- Lineare Feedback-Shift-Register
- Nicht-lineare Kombinierer
- “Etwas tacktet etwas anderes”

Ausblick auf Blockchiffren

Output Feedback Mode (OFB)



Output Feedback (OFB) mode encryption

Man erhält eine synchrone Stromchiffre (mit Takt ein Block).

Warum der Initialisierungevektor?

Das sollte das Verfahren sicher gegen gewählte Klartextangriffe (IND-CPA-sicher) machen.

Output Feedback Mode (OFB)

Was bedeutet hier IND-CPA-sicher im Vergleich zu IND-EAV-sicher?

Wir haben gesehen:

Das one-time-Pad ist IND-EAV-sicher.

Es ist aber nicht IND-CPA-sicher.

Aber: Es als Verfahren mit Zustand ist es sicher gegen gewählte Klartextangriffe, sagen wir: IND-CPA-mit-Zustand-sicher.

(Es ist auch IND-CCA-mit-Zustand-sicher.)

"Mit Zustand" heißt: Hier wird "sich was gemerkt" von Anwendung zu Anwendung.

Achtung nochmal: IND-CPA-mit-Zustand-sicher ist was anderes als IND-CPA-sicher!

Das gilt auch für das pseudo-one-time-pad.

Output Feedback Mode (OFB)

Hier wird gesagt:

Für eine “gute Blockchiffre” ist der OFB IND-CPA-sicher.

D.h. ohne Zustand, es wird sich nichts gemerkt von Anwendung zu Anwendung.

Innerhalb jeder einzelnen Anwendung sieht es so aus wie eine Stromchiffre, d.h. “mit Zustand”, aber der wird dann vergessen.

(Wie wir es auch machen, wenn wir ein peudo-one-time-pad im CPA-Spiel untersuchen.)

Eine idealisierte Konstruktion

Beachte: Im OFB muss die Blockchiffre nicht invertierbar sein.

Wir nehmen an, die Kommunikationspartner haben eine Maschine M,

die unter Eingabe von $x \in \{0,1\}^n$

einen uniform zufälligen String in $\{0,1\}^n$ ausgibt,

außer x war schon mal dran, dann wird das Ergebnis nochmal ausgegeben.

Sonst keiner hat Zugriff auf diese Maschine.

So was kann man auch modellieren. Dann heißt M Zufallsorakel.

Für unsere theoretischen Überlegungen ist hier n wieder ein Parameter.

Eine idealisierte Konstruktion

Wir betrachten diese Verschlüsselung / Entschlüsselung:

Verschlüsselung:

Wähle $r \in \{0,1\}^n$ unform zufällig.

Sende $c := (r, M(r) \oplus m)$.

Entschlüsselung:

Gegeben $c = (r, a)$, berechne $M(r) \oplus a = m$.

Vergleiche OFB!

CPA-Sicherheit

Das ist CPA-sicher [im Zufallsorakelmodell].

Begründung

Ein Angreifer kann beliebige Nachrichten $m \in \{0,1\}^n$ verschlüsseln lassen.

Er erhält: $c = (r, M(r) \oplus m)$.

Das ist "genauso gut" wie $(r, M(r))$ zu erhalten.

Man kann das Spiel so umformulieren:

Wir gehen vom IND-EAV-Spiel aus.

Zusätzlich kann der Angreifer vom Herausforderer stets Tupel $(r, M(r))$ erhalten (r uniform zufällig).

CPA-Sicherheit

Also: Der Angreifer kann $(r, M(r))$ erhalten, wenn er will.

- r ist uniform zufällig.
- $M(r)$ ist auch uniform zufällig und unabhängig von r, außer r wurde schon mal gewählt.

Nur wenn er "das richtige r" trifft, bringt das was.

Ansonsten könnte er r und $M(r)$ auch selbst erzeugen.

Die Frage nach $(r, M(r))$ ist recht sinnlos ...

RC4

RC4

RC4 ist ein Pseudozufallsgenerator / eine synchrone Stromchiffre

wurde 1987 von Ron Rivest designed,

war zunächst ein Geschäftsgeheimnis der Firma RSA,

wurde 1994 öffentlich gemacht,

wurde insbesondere in WEP und WPA benutzt,

wurde 2004 abgelöst von WPA2 mit AES,

ist ziemlich unsicher.

RC4

In dem Verfahren wird aus dem Schlüssel k zunächst eine Substitutionstabelle S mit 256 Bits erstellt.

Dann wird diese beständig geändert.

Initialisierungsalgorithmus (KSA)

Eingabe: Ein Schlüssel k bestehend aus 5 bis 256 Bites (Länge $= SL$).

Für i von 0 bis 255
  $S[i] := i$
$j := 0$
Für i von 0 bis 255
  $j := (j + S[i] + k[i \mod SL]) \mod 256$
  Vertausche $S[i]$ und $S[j]$

Pseudozufallsgenerator

$i := 0$
$j := 0$
Wiederhole:
  $i := (i + 1) \mod 256$
  $j := (j + S[i]) \mod 256$
  Vertausche Werte $S[i]$ und $S[j]$
  $t := (S[i] + S[j]) \mod 256$
  Gib $S[t]$ aus

Ernstes Problem. Der Anfang des Schlüsselstroms ist erkennbar nicht unabhängig von Schlüssel, und das kann man wirklich benutzen.