

Algorithmus Stromchiffre

Sei G ein Pseudozufallsgenerator mit Expansionsfaktor $\ell(n)$. Wir definieren $\Pi_S = (Gen, Enc, Dec)$ mit Sicherheitsparameter n für Nachrichten der Länge $\ell(n)$.

- 1 **Gen:** Wähle $k \in_R \{0, 1\}^n$.
- 2 **Enc:** Bei Eingabe $k \in \{0, 1\}^n$ und $m \in \{0, 1\}^{\ell(n)}$, berechne
$$c := G(k) \oplus m.$$
- 3 **Dec:** Bei Eingabe $k \in \{0, 1\}^n$ und $c \in \{0, 1\}^{\ell(n)}$, berechne
$$m := G(k) \oplus c.$$

Anmerkung:

- Π_S verwendet $G(k)$ anstatt $r \in \{0, 1\}^{\ell(n)}$ wie im One-Time Pad.
- D.h. wir benötigen nur n statt $\ell(n)$ echte Zufallsbits.
(Bsp: n 128 Bit, $\ell(n)$ mehrere Megabyte)

Sicherheit unserer Stromchiffre

Satz Sicherheit von Π_s

Sei G ein Pseudozufallsgenerator. Dann ist Π_s KPA-sicher.

Beweis:

- Idee: Erfolgreicher Angreifer \mathcal{A} liefert Unterscheider für G .
- Sei \mathcal{A} ein KPA-Angreifer auf Π_s mit Vorteil $\epsilon(n)$.
- Wir konstruieren mittels \mathcal{A} folgenden Unterscheider D für G .

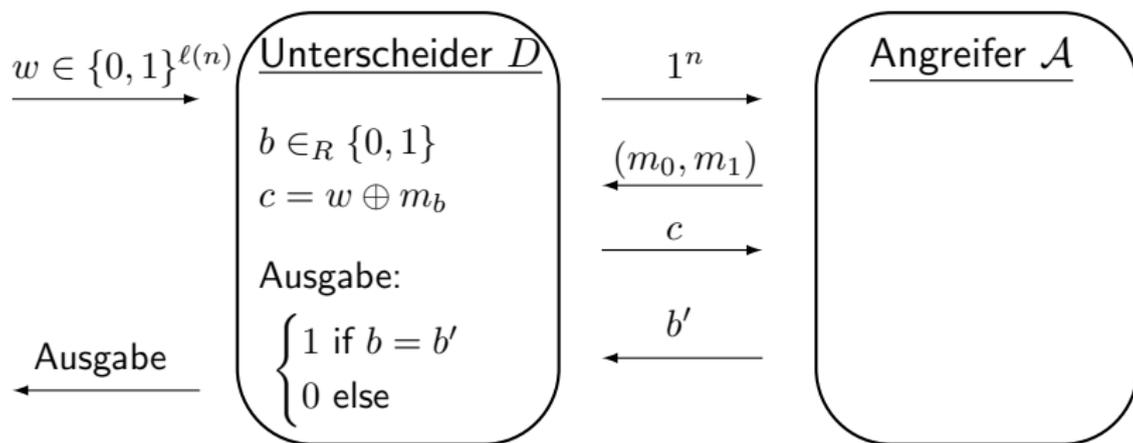
Algorithmus Unterscheider D

EINGABE: $w \in \{0, 1\}^{\ell(n)}$

- 1 Erhalte $(m_0, m_1) \leftarrow \mathcal{A}(1^n)$
- 2 Wähle $b \in_R \{0, 1\}$ und berechne $c := w \oplus m_b$.
- 3 Erhalte $b' \leftarrow \mathcal{A}(c)$.

AUSGABE: $= \begin{cases} 1 & \text{falls } b' = b, \text{ Interpretation: } w = G(k), k \in_R \{0, 1\}^n \\ 0 & \text{sonst, Interpretation: } w \in_R \{0, 1\}^{\ell(n)} \end{cases}$.

Sicherheit von Π_s



Fall 1: $w = r \in_R \{0, 1\}^{\ell(n)}$, d.h. w ist ein echter Zufallsstring.

- Dann ist die Verteilung von c identisch zur Verteilung beim One-Time Pad Π_{otp} .
- Damit folgt aus der perfekten Sicherheit des One-Time Pads

$$\text{Ws}[D(w) = 1] = \text{Ws}[\text{PrivK}_{\mathcal{A}, \Pi_{\text{otp}}}^{\text{eav}}(n) = 1] = \frac{1}{2}.$$

Sicherheit von Π_s

Fall 2: $w = G(k)$ für $k \in_R \{0, 1\}^n$, d.h. w wurde mittels G generiert.

- Damit ist die Verteilung von c identisch zur Verteilung in Π_s .
- Es folgt $\mathbb{W}_s[D(w) = 1] = \mathbb{W}_s[\text{PrivK}_{\mathcal{A}, \Pi_s}^{\text{eav}}(n) = 1] = \frac{1}{2} + \epsilon(n)$.

Aus der Pseudozufälligkeit von G folgt insgesamt

$$\text{negl}(n) \geq \left| \underbrace{\mathbb{W}_s[D(r) = 1]}_{\frac{1}{2}} - \underbrace{\mathbb{W}_s[D(G(k)) = 1]}_{\frac{1}{2} + \epsilon(n)} \right| = \epsilon(n).$$

Damit ist der Vorteil jedes Angreifers \mathcal{A} vernachlässigbar.

Generator mit variabler Ausgabelänge

Ziel: Um Nachricht beliebiger Länge $\ell(n)$ mit Algorithmus Π_s zu verschlüsseln, benötigen wir ein G mit variabler Ausgabelänge $\ell(n)$.

Definition Pseudozufallsgenerator mit variabler Ausgabelänge

Ein pt Algorithmus G heißt *Pseudozufallsgenerator mit variabler Ausgabelänge* falls

- 1 Für eine Saat $s \in \{0, 1\}^n$ und eine Länge $\ell \in N$ berechnet $G(s, 1^\ell)$ einen String der Länge ℓ .
- 2 Für jedes Polynom $\ell(\cdot)$ ist $G_{\ell}(s) := G(s, 1^{\ell(n)})$, $s \in \{0, 1\}^n$ ein Pseudozufallsgenerator mit Expansionsfaktor $\ell(n)$.
- 3 Für alle s, ℓ, ℓ' mit $\ell \leq \ell'$ ist $G(s, 1^\ell)$ ein Präfix von $G(s, 1^{\ell'})$.

Anmerkungen:

- Für Nachricht m erzeugen wir Chiffretext $c := G(k, 1^{|m|}) \oplus m$.
- Bedingung 3 ist technischer Natur, um im KPA-Spiel Verschlüsselungen von m_0, m_1 beliebiger Länge zuzulassen.

Existenz Zufallsgenerator mit/ohne variable Länge

Fakt Existenz von Zufallsgeneratoren

- 1 Die Existenz von Pseudozufallsgeneratoren folgt unter der Annahme der Existenz von sogenannten Einwegfunktionen.
- 2 Pseudozufallsgeneratoren variabler Ausgabelänge können aus jedem Pseudozufallsgenerator fixer Länge konstruiert werden.

Vereinfacht:

Einwegfunktion \Rightarrow Pseudozufallsgenerator fixer Länge
 \Rightarrow Pseudozufallsgenerator variabler Länge
(\Rightarrow Einwegfunktion)

(mehr dazu im Verlauf der Vorlesung)

Diskussion Stromchiffren

Stromchiffre:

- Pseudozufallsgeneratoren mit variabler Ausgabelänge liefern Strom von Zufallsbits.
- Wir nennen diese Stromgeneratoren auch Stromchiffren.

Stromchiffren in der Praxis:

- Bsp: Linear Feedback Shift Register (LFSR), RC4, und A5/1.
- Viele Stromchiffren in der Praxis sind sehr schnell, allerdings sind die meisten leider ad hoc Lösungen ohne Sicherheitsbeweis.
- Schwächen in RC4 führten zum Brechen des WEP Protokolls.
- LFSRs sind kryptographisch vollständig gebrochen worden.
- Seit 2004: Ecrypt-Projekt eStream zur Etablierung sicherer Standard-Stromchiffren. Derzeitige Kandidaten:
 - ▶ Software: HC-128, Rabbit, Salsa20/12 und SOSEMANUK.
 - ▶ Hardware: Grain v1, MICKEY v2 und Trivium.

Sicherheit mehrfacher Verschlüsselung

Bisher: Angreifer \mathcal{A} erhält nur *eine* Verschlüsselung. Nachrichten müssen aber sicher bleiben, falls \mathcal{A} mehrere Chiffretexte erhält.

Spiel Mehrfache Verschlüsselung $\text{PrivK}_{\mathcal{A},\Pi}^{\text{mult}}(n)$

Sei Π ein Verschlüsselungsverfahren und \mathcal{A} ein Angreifer.

- 1 $(M_0, M_1) \leftarrow \mathcal{A}(1^n)$ mit $M_0 = (m_0^1, \dots, m_0^t)$, $M_1 = (m_1^1, \dots, m_1^t)$ und $|m_0^i| = |m_1^i|$ für alle $i \in [t]$.
- 2 $k \leftarrow \text{Gen}(1^n)$.
- 3 Wähle $b \in_R \{0, 1\}$. $b' \leftarrow \mathcal{A}((\text{Enc}_k(m_b^1), \dots, \text{Enc}_k(m_b^t)))$.
- 4 $\text{PrivK}_{\mathcal{A},\Pi}^{\text{mult}}(n) = \begin{cases} 1 & \text{für } b = b' \\ 0 & \text{sonst} \end{cases}$.

Mult-KPA Spiel

PrivK $_{\mathcal{A},\Pi}^{mult}(n)$

$k \leftarrow \text{Gen}(1^n)$

$b \in_R \{0, 1\}$

$c^j = \text{Enc}_k(m_b^j)$

$C = (c^1, \dots, c^t)$

Ausgabe:

$= \begin{cases} 1 & \text{falls } b = b' \\ 0 & \text{sonst} \end{cases}$

1^n

(M_0, M_1)

C

b'

Angreifer \mathcal{A}

$M_i = (m_i^1, \dots, m_i^t), i = 0, 1$

$|m_0^j| = |m_1^j| \quad \forall j, m_i^j \in \mathcal{M}$

$b' \in \{0, 1\}$

Multi-KPA Sicherheit

Definition Multi-KPA Sicherheit

Ein Verschlüsselungsschema $\Pi = (Gen, Enc, Dec)$ besitzt *ununterscheidbare mehrfache Chiffretexte gegenüber KPA* falls für alle ppt \mathcal{A} :

$$\text{Ws}[PrivK_{\mathcal{A},\Pi}^{mult}(n) = 1] \leq \frac{1}{2} + \text{negl}(n).$$

Der Wsraum ist definiert über die Münzwürfe von \mathcal{A} und $PrivK_{\mathcal{A},\Pi}^{mult}$.

Notation: Wir bezeichnen Π als *mult-KPA sicher*.

KPA Sicherheit vs. mult-KPA Sicherheit

Satz KPA Sicherheit vs. mult-KPA Sicherheit

KPA Sicherheit impliziert **nicht** mult-KPA Sicherheit.

Beweis:

- Π_S ist KPA-sicher. Wir betrachten folgendes mult-KPA Spiel.

Algorithmus Angreifer \mathcal{A} für Π_S

EINGABE: Sicherheitsparameter n

- 1 $(M_0, M_1) \leftarrow \mathcal{A}(1^n)$ mit $M_0 = (0^{\ell(n)}, 0^{\ell(n)})$, $M_1 = (0^{\ell(n)}, 1^{\ell(n)})$.
- 2 Erhalte $C = (c_1, c_2)$.

AUSGABE: $b' = \begin{cases} 1 & \text{falls } c_1 = c_2 \\ 0 & \text{sonst} \end{cases}$.

- Da Enc von Π_S deterministisch ist, gilt $\text{Ws}[PrivK_{\mathcal{A}, \Pi_S}^{mult}(n) = 1] = 1$.

Multi-KPA Angreifer auf Π_s

PrivK $_{\mathcal{A}, \Pi_s}^{mult}(n)$

$k \leftarrow \text{Gen}(1^n)$

$b \in_R \{0, 1\}$

$c^j = \text{Enc}_k(m_b^j)$

Ausgabe:

$= \begin{cases} 1 & \text{falls } b = b' \\ 0 & \text{sonst} \end{cases}$

1^n

(M_0, M_1)

(c^1, c^2)

b'

Angreifer \mathcal{A}

$M_0 = (0^{\ell(n)}, 0^{\ell(n)})$

$M_1 = (0^{\ell(n)}, 1^{\ell(n)})$

Falls $c^1 = c^2$, setze $b = 0$.

Falls $c^1 \neq c^2$, setze $b = 1$.

Unsicherheit deterministischer Verschlüsselung

Korollar Unsicherheit deterministischer Verschlüsselung

Sei $\Pi = (Gen, Enc, Dec)$ mit deterministischer Verschlüsselung Enc .
Dann ist Π unsicher gegenüber mult-KPA Angriffen.

- Voriger Angreifer \mathcal{A} nutzt lediglich, dass für zwei identische Nachrichten $m_0 = m_1$ auch die Chiffretexte identisch sind.

Notwendig: Wir benötigen randomisiertes Enc , das identische Nachrichten auf unterschiedliche (exponentiell viele) Chiffretexte abbildet.

Synchronisierte sichere mehrfache Verschlüsselung

Synchronisierter Modus für Stromchiffren:

- Nutze für Nachrichten m_1, m_2, \dots, m_n sukzessive Teil des Bitstroms $G(s) = s_1 s_2 \dots s_n$ mit $|s_i| = |m_i|$.
- D.h. es werden nie Teile des Bitstroms wiederverwendet.
- Ermöglicht einfaches Protokoll zur Kommunikation von A und B :
 - ▶ A verschlüsselt mit s_1 , B entschlüsselt mit s_1 .
 - ▶ Danach verschlüsselt B mit s_2 , mit dem auch A entschlüsselt, usw.
- Erfordert, dass A und B die Position im Bitstrom *synchronisieren*.
- Verfahren ist sicher, da die Gesamtheit der Nachrichten als einzelne Nachricht $m = m_1 \dots m_n$ aufgefasst werden kann.

Nicht-synchronisierte mehrfache Verschlüsselung

Nicht-synchronisierter Modus für Stromchiffren:

- Erweitern Funktionalität von Pseudozufallsgeneratoren G :
 - 1 G erhält zwei Eingaben: Schlüssel k und Initialisierungsvektor IV .
 - 2 $G(k, IV)$ ist pseudozufällig selbst für bekanntes IV .
- Verschlüsselung von m mit erweiterten Pseudozufallsgeneratoren:

$$Enc_k(m) := (IV, G(k, IV) \oplus m) \text{ für } IV \in_R \{0, 1\}^n.$$

- Entschlüsselung möglich, da IV im Klartext mitgesendet wird.
- D.h. eine Nachricht m besitzt 2^n mögliche Verschlüsselungen.
- **Warnung:** Konstruktion solch erweiterter G ist nicht-trivial.