Der komplexitätstheoretische Zugang zur Kryptographie

Claus Diem

Im Wintersemester 2017 / 18

Literatur

Oded Goldreich: Foundations of Cryptography

Jonathan Katz & Yeduda Lindell: Intoduction to Modern Cryptography

Claus Diem: Kryptologie – Methoden, Anwendungen und Herausforderungen

Motivation

Kryptographie

Klassische Bedeutung: Lehre der Geheimschrift

Kryptographie

Klassische Bedeutung: Lehre der Geheimschrift

Oder Kryptologie?

Kryptographie

Klassische Bedeutung: Lehre der Geheimschrift

Oder Kryptologie?

Moderne Bedeutung:

Die Ziele der Kryptographie / Kryptologie umfassen alle sicherheitsrelevanten Aspekte des Verarbeitens, Übertragens und Benutzens von Informationen in Anwesenheit eines Gegners.

Kryptographie

Klassische Bedeutung: Lehre der Geheimschrift

Oder Kryptologie?

Moderne Bedeutung:

Die Ziele der Kryptographie / Kryptologie umfassen alle sicherheitsrelevanten Aspekte des Verarbeitens, Übertragens und Benutzens von Informationen in Anwesenheit eines Gegners.

- Vertraulichkeit
- Authentisierung
- Verbindlichkeit
- Integrität

Sicher?

Grundlegende Frage: Ist ein verwendetes Verfahren sicher?

Sicher?

Grundlegende Frage: Ist ein verwendetes Verfahren sicher?

Kritik: Hierzu muss man erstmal im Einzelnen definieren, was man mit “sicher” meint.

Sicher?

Grundlegende Frage: Ist ein verwendetes Verfahren sicher?

Kritik: Hierzu muss man erstmal im Einzelnen definieren, was man mit “sicher” meint.

Moderne / wissenschaftliche Kryptographie:

- Man studiert die verwendeten Verfahren von einem mathematischen Gesichtspunkt aus.
- Man definiert die Verfahren klar.
- Man gibt sich eine Klasse von Angriffen vor und definiert auf Grundlage dieser Angriffe, wann ein Verfahren als “sicher” gelten soll.

Sicher?

Beispiel:

- Man definiert, was ein Verschlüsselungsverfahren ist.
- Man definiert, wann dieses perfekt sicher ist.

Sicher?

Beispiel:

- Man definiert, was ein Verschlüsselungsverfahren ist.
- Man definiert, wann dieses perfekt sicher ist.

Dann kann man zeigen:

Sicher?

Beispiel:

- Man definiert, was ein Verschlüsselungsverfahren ist.
- Man definiert, wann dieses perfekt sicher ist.

Dann kann man zeigen:

- Das one-time pad ist perfekt sicher.
- Jedes perfekt sichere Verfahren muss Schlüssel verwenden, die so lang wie die Nachricht sind (wenn die Alphabete gleich sind.)

Der komplexitätstheoretische Zugang

Ab jetzt: Verschlüsselungsverfahren = mathematische Methode zur Verschlüsselung

Der komplexitätstheoretische Zugang

Ab jetzt: Verschlüsselungsverfahren = mathematische Methode zur Verschlüsselung

Frage: Kann man eine geeignete abgeschwächte Definition von “sicher” angeben?

Der komplexitätstheoretische Zugang

Ab jetzt: Verschlüsselungsverfahren = mathematische Methode zur Verschlüsselung

Frage: Kann man eine geeignete abgeschwächte Definition von "sicher" angeben?

Erste Antwort: Man benutze einen komplexitätstheoretischen Zugang.

Der komplexitätstheoretische Zugang

Ab jetzt: Verschlüsselungsverfahren = mathematische Methode zur Verschlüsselung

Frage: Kann man eine geeignete abgeschwächte Definition von "sicher" angeben?

Erste Antwort: Man benutze einen komplexitätstheoretischen Zugang.

Dann wird's kompliziert ...

Komplexitätstheoretischer Zugang

Auf Grundlage der Komplexitätstheorie kann man eine Theorie der Kryptographie entwickeln.

Komplexitätstheorischer Zugang

Auf Grundlage der Komplexitätstheorie kann man eine Theorie der Kryptographie entwickeln.

Diese stellt grundlegende Paradigma und Methoden für verschiedene Aspekte der Kryptographie bereit.

Komplexitätstheorischer Zugang

Auf Grundlage der Komplexitätstheorie kann man eine Theorie der Kryptographie entwickeln.

Diese stellt grundlegende Paradigma und Methoden für verschiedene Aspekte der Kryptographie bereit.

Achtung: Es gibt immer eine Lücke zwischen Theorie & Praxis!

P, NP etc.

Wir benutzen $\{0,1\}^*$ oder $\mathbf{N}$:

$$100110 \longleftrightarrow 011001 \longleftrightarrow 1011001 \longleftrightarrow (1011001)_2$$

Wir benutzen $\{0,1\}^*$ oder $\mathbf{N}$:

$$100110 \longleftrightarrow 011001 \longleftrightarrow 1011001 \longleftrightarrow (1011001)_2$$

Im Folgenden: $\{0,1\}^*$. Für einen String x sei $|x|$ die Länge.

Grundlegendes Paradigma:

- Es werden nur asymptotische Aussagen gemacht.
- Ein qualitativ schneller Algorithmus ist ein solcher, dessen Laufzeit polynomiell (beschränkt) in der Eingabelänge ist.

Komplexitätstheorie

Grundlegendes Paradigma:

- Es werden nur asymptotische Aussagen gemacht.
- Ein qualitativ schneller Algorithmus ist ein solcher, dessen Laufzeit polynomiell (beschränkt) in der Eingabelänge ist.

Sprache: $L \subseteq \{0,1\}^*$, entspricht Funktion $f : \{0,1\}^* \to \{0,1\}$.

Grundlegendes Paradigma:

- Es werden nur asymptotische Aussagen gemacht.
- Ein qualitativ schneller Algorithmus ist ein solcher, dessen Laufzeit polynomiell (beschränkt) in der Eingabelänge ist.

Sprache: $L \subseteq \{0,1\}^*$, entspricht Funktion $f : \{0,1\}^* \to \{0,1\}$.

Entscheidungsproblem zu $L \subseteq \{0,1\}^*$ / $f : \{0,1\}^* \to \{0,1\}$:

Komplexitätstheorie

Grundlegendes Paradigma:

- Es werden nur asymptotische Aussagen gemacht.
- Ein qualitativ schneller Algorithmus ist ein solcher, dessen Laufzeit polynomiell (beschränkt) in der Eingabelänge ist.

Sprache: $L \subseteq \{0,1\}^*$, entspricht Funktion $f : \{0,1\}^* \to \{0,1\}$.

Entscheidungsproblem zu $L \subseteq \{0,1\}^*$ / $f : \{0,1\}^* \to \{0,1\}$:
Entscheide, ob eine Eingabe x in L liegt / ob $f(x) = 1$ ist.

Komplexitätsklassen:

Komplexitätsklassen:

P. Menge der in Polynomzeit mit einer DTM entscheidbaren Sprachen / Funktionen / Probleme

Komplexitätsklassen:

P. Menge der in Polynomzeit mit einer DTM entscheidbaren Sprachen / Funktionen / Probleme

NP. Menge der in Polynomzeit mit einer (möglicherweise) nichtdeterminstischen TM entscheidbaren Sprachen / Funktionen / Probleme

Komplexitätsklassen:

P. Menge der in Polynomzeit mit einer DTM entscheidbaren Sprachen / Funktionen / Probleme

NP. Menge der in Polynomzeit mit einer (möglicherweise) nichtdeterminstischen TM entscheidbaren Sprachen / Funktionen / Probleme

BPP. Menge der mit einer randomizierten TM mit beschränktem Fehler in Polynomzeit entscheidbaren Sprachen / Funktionen / Probleme

Mögliche Definitionen für $L \in \mathrm{NP}$:

- Es gibt eine nicht-deterministische TM T mit:
  - T terminiert in Polynomzeit.
  - Für Eingabe x sind äquivalent:
    - $x \in L$.
    - Mindestens eine Ausgabe von T angewandt auf x ist 1.

Komplexitätsklassen

Mögliche Definitionen für $L \in \mathrm{NP}$:

- Es gibt eine nicht-deterministische TM T mit:
  - T terminiert in Polynomzeit.
  - Für Eingabe x sind äquivalent:
    - $x \in L$.
    - Mindestens eine Ausgabe von T angewandt auf x ist 1.
- Es gibt eine Relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, eine DTM T und ein positives Polynom $p(n)$ mit:
  - T berechnet R.
  - $x \in L$ genau dann, wenn es ein y mit $|y| \leq p(|x|)$ und $(x, y) \in R$ gibt.
  - T terminiert in Polynomzeit.

Mögliche Definitionen für $L \in \mathrm{NP}$:

- Es gibt eine nicht-deterministische TM T mit:
  - T terminiert in Polynomzeit.
  - Für Eingabe x sind äquivalent:
    - $x \in L$.
    - Mindestens eine Ausgabe von T angewandt auf x ist 1.

- Es gibt eine Relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, eine DTM T und ein positives Polynom $p(n)$ mit:
  - T berechnet R.
  - $x \in L$ genau dann, wenn es ein y mit $|y| \leq p(|x|)$ und $(x, y) \in R$ gibt.
  - T terminiert in Polynomzeit.

So ein y heißt dann *Zeuge* oder *Beweis.*

Komplexitätsklassen

$L \in$BPP:

$L \in$BPP:

Es gibt eine probabilistische TM T mit:

- T terminiert in Polynomzeit.
- Für $x \in L$ gibt T mit Wahrscheinlichkeit $\geq \frac{2}{3}$ die 1 aus.
- Für $x \notin L$ gibt T mit Wahrscheinlichkeit $\geq \frac{2}{3}$ die 0 aus.

Komplexitätsklassen

$L \in$BPP:

Es gibt eine probabilistische TM T mit:

- T terminiert in Polynomzeit.
- Für $x \in L$ gibt T mit Wahrscheinlichkeit $\geq \frac{2}{3}$ die 1 aus.
- Für $x \notin L$ gibt T mit Wahrscheinlichkeit $\geq \frac{2}{3}$ die 0 aus.

Beachte: Das muss für alle x gelten!

Komplexitätsklassen

$L \in$BPP:

Es gibt eine probabilistische TM T mit:

- T terminiert in Polynomzeit.
- Für $x \in L$ gibt T mit Wahrscheinlichkeit $\geq \frac{2}{3}$ die 1 aus.
- Für $x \notin L$ gibt T mit Wahrscheinlichkeit $\geq \frac{2}{3}$ die 0 aus.

Beachte: Das muss für alle x gelten!

Die Fehlerwahrscheinlichkeit ist also stets $\leq \frac{1}{3}$. Die kann durch jede Konstante $< \frac{1}{2}$ ersetzt werden.

Komplexitätsklassen

Es ist offenbar $\mathsf{P} \subseteq \mathsf{NP}$, $\mathsf{P} \subseteq \mathsf{BPP}$.

Komplexitätsklassen

Es ist offenbar $\mathsf{P} \subseteq \mathsf{NP}$, $\mathsf{P} \subseteq \mathsf{BPP}$.

Alle anderen denkbaren Beziehungen sind unbekannt.

Es ist offenbar $\mathsf{P} \subseteq \mathsf{NP}$, $\mathsf{P} \subseteq \mathsf{BPP}$.

Alle anderen denkbaren Beziehungen sind unbekannt.

Ist $\mathsf{P} = \mathsf{NP}$, $\mathsf{P} = \mathsf{BPP}$, $\mathsf{NP} \subseteq \mathsf{BPP}$, $\mathsf{BPP} \subseteq \mathsf{NP}$?

Vernachlässigbar

Im folgenden:

$$\begin{aligned}\text{Algorithmus} &= \text{Turingmachine oder}\\ &= \text{informelle Beschreibung einer Berechnung.}\end{aligned}$$

Angreifer werden immer mittels randomisierter Algorithmen modelliert. Im quanitativen Zugang sind dies Polynomzeitalgorithmen.

Aber: Wir wollen oftmals eine wesentlich geringere Erfolgswahrscheinlichkeit.

Das Polynomzeit-Paradigma motiviert:

Eine Funktion $\epsilon : \mathbf{N} \to \mathbf{R}_{\geq 0}$ heißt vernachlässigbar, falls für jedes postive Polynom $p(n)$ gilt:

$$\epsilon(n) \leq \frac{1}{p(n)}$$

für alle n mit $n \gg 0$.

Einwegfunktionen

Einwegfunktionen

Es sei $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ effizient berechenbar.

Einwegfunktionen

Es sei $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ effizient berechenbar.

Wir wollen, dass Urbilder schwer berechenbar sind.

Einwegfunktionen

Es sei $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ effizient berechenbar.

Wir wollen, dass Urbilder schwer berechenbar sind.

Es sei n die Eingabelänge.

Einwegfunktionen

Es sei $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ effizient berechenbar.

Wir wollen, dass Urbilder schwer berechenbar sind.

Es sei n die Eingabelänge.

Genauer wollen wir: Der Anteil der $x \in \{0,1\}^n$, für welche man aus $f(x)$ ein x' mit $f(x') = f(x)$ effizient berechnen kann, soll vernachlässigbar sein.

Einwegfunktionen

Es sei $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ effizient berechenbar.

Wir wollen, dass Urbilder schwer berechenbar sind.

Es sei n die Eingabelänge.

Genauer wollen wir: Der Anteil der $x \in \{0,1\}^n$, für welche man aus $f(x)$ ein x' mit $f(x') = f(x)$ effizient berechnen kann, soll vernachlässigbar sein.

Wir lassen hier randomisierte Algorithmen zu.

Einwegfunktionen

Idee einer Definition. Eine Einwegfunktion ist eine effizient berechenbare Funktion $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ mit:

Einwegfunktionen

Idee einer Definition. Eine Einwegfunktion ist eine effizient berechenbare Funktion $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ mit:

Für alle PPT-Algorithmen $\mathcal{A}$ ist

$$\mathbf{P}[f(\mathcal{A}(f(x))) = f(x)] ,$$

wobei $x \in \{0,1\}^n$ uniform ist, vernachlässigbar in $n = |x|$.

Einwegfunktionen

Idee einer Definition. Eine Einwegfunktion ist eine effizient berechenbare Funktion $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ mit:

Für alle PPT-Algorithmen $\mathcal{A}$ ist

$$\mathbf{P}[f(\mathcal{A}(f(x))) = f(x)] \; ,$$

wobei $x \in \{0,1\}^n$ uniform ist, vernachlässigbar in $n = |x|$.

Problem. Die Funktion $f : x \mapsto |x|$ ist hiernach keine Einwegfunktion.

Idee einer Definition. Eine Einwegfunktion ist eine effizient berechenbare Funktion $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ mit:

Für alle PPT-Algorithmen $\mathcal{A}$ ist

$$\mathbf{P}[f(\mathcal{A}(f(x))) = f(x)] \ ,$$

wobei $x \in \{0,1\}^n$ uniform ist, vernachlässigbar in $n = |x|$.

Problem. Die Funktion $f : x \mapsto |x|$ ist hiernach keine Einwegfunktion.

Aus der Länge von x kann man nicht effizient x berechnen, weil x einfach zu groß ist.

Einwegfunktionen

Idee einer Definition. Eine Einwegfunktion ist eine effizient berechenbare Funktion $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ mit:

Für alle PPT-Algorithmen $\mathcal{A}$ ist

$$\mathbf{P}[f(\mathcal{A}(f(x))) = f(x)] \; ,$$

wobei $x \in \{0,1\}^n$ uniform ist, vernachlässigbar in $n = |x|$.

Problem. Die Funktion $f : x \mapsto |x|$ ist hiernach keine Einwegfunktion.

Aus der Länge von x kann man nicht effizient x berechnen, weil x einfach zu groß ist.

Das ist nicht sinnvoll!

Einwegfunktionen

Idee einer Definition. Eine Einwegfunktion ist eine effizient berechenbare Funktion $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ mit:

Für alle PPT-Algorithmen $\mathcal{A}$ ist

$$\mathbf{P}[f(\mathcal{A}(f(x))) = f(x)] \,,$$

wobei $x \in \{0,1\}^n$ uniform ist, vernachlässigbar in $n = |x|$.

Problem. Die Funktion $f : x \mapsto |x|$ ist hiernach keine Einwegfunktion.

Aus der Länge von x kann man nicht effizient x berechnen, weil x einfach zu groß ist.

Das ist nicht sinnvoll!

Lösung. Wir geben auch $1^n = \overbrace{1 \cdots 1}^{n-\text{mal}}$ (Wort) als Eingabe.

Einwegfunktionen

Definition. Eine (starke) Einwegfunktion ist eine in Polynomzeit berechenbare Funktion $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ mit:

Definition. Eine (starke) Einwegfunktion ist eine in Polynomzeit berechenbare Funktion $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ mit:

Für alle PPT-Algorithmen $\mathcal{A}$ ist

$$\mathbf{P}[f(\mathcal{A}(1^n, f(x))) = f(x)] \, ,$$

wobei $x \in \{0,1\}^n$ uniform ist, vernachlässigbar in n.

Definition. Eine (starke) Einwegfunktion ist eine in Polynomzeit berechenbare Funktion $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ mit:

Für alle PPT-Algorithmen $\mathcal{A}$ ist

$$\mathbf{P}[f(\mathcal{A}(1^n, f(x))) = f(x)] ,$$

wobei $x \in \{0,1\}^n$ uniform ist, vernachlässigbar in n.

Man sieht:

Einwegfunktionen

Definition. Eine (starke) Einwegfunktion ist eine in Polynomzeit berechenbare Funktion $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ mit:

Für alle PPT-Algorithmen $\mathcal{A}$ ist

$$\mathbf{P}[f(\mathcal{A}(1^n, f(x))) = f(x)] \, ,$$

wobei $x \in \{0,1\}^n$ uniform ist, vernachlässigbar in n.

Man sieht:

Wenn es eine Einwegfunktion gibt, ist NP $\subsetneq$BPP.

Vermutung. Die Funktion

$$\{\, (m, n) \in \mathbf{N} \times \mathbf{N} \mid \lceil \log_2(m) \rceil = \lceil \log_2(n) \rceil \,\} \longrightarrow \mathbf{N}\ ,$$

$$(m, n) \mapsto m \cdot n$$

ist / führt zu einer Einwegfunktion.

Familien von Einwegfunktionen

Idee. Gegeben der Sicherheitsparameter wählt zunächst einen Parameter. Dazu betrachtet man dann eine Funktion mit endlicher Ein- und Ausgabelänge.

Idee. Gegeben der Sicherheitsparameter wählt zunächst einen Parameter. Dazu betrachtet man dann eine Funktion mit endlicher Ein- und Ausgabelänge.

Wichtiges vermutetes Beispiel: Modulo-Exponentieren

- Parameter: Eine Primzahl p und ein Erzeuger g von $(\mathbf{Z}/p\mathbf{Z})^\times$.
- Funktion: $\{0, \dots, p-2\} \longrightarrow (\mathbf{Z}/p\mathbf{Z})^\times, \; x \mapsto g^x$

Familien von Einwegfunktionen

Idee. Gegeben der Sicherheitsparameter wählt zunächst einen Parameter. Dazu betrachtet man dann eine Funktion mit endlicher Ein- und Ausgabelänge.

Wichtiges vermutetes Beispiel: Modulo-Exponentieren

- Parameter: Eine Primzahl p und ein Erzeuger g von $(\mathbf{Z}/p\mathbf{Z})^\times$.
- Funktion: $\{0, \ldots, p-2\} \longrightarrow (\mathbf{Z}/p\mathbf{Z})^\times, \; x \mapsto g^x$
- Umkehrung: Berechnung von x
  $=$ diskretes Logarithmusproblem

Familien von Einwegfunktionen

Allgemeiner, z.B. für $G = E(\mathbf{F}_q)$:

Familien von Einwegfunktionen

Allgemeiner, z.B. für $G = E(\mathbf{F}_q)$:

- Parameter: Eine endliche Gruppe $G = (G, \cdot)$ mit effizienter Arithemtik, $a \in G$.
- Funktion: $G \longrightarrow G \ , \ x \mapsto a^x$

Familien von Einwegfunktionen

Allgemeiner, z.B. für $G = E(\mathbf{F}_q)$:

- Parameter: Eine endliche Gruppe $G = (G, \cdot)$ mit effizienter Arithemtik, $a \in G$.
- Funktion: $G \longrightarrow G \ , \ x \mapsto a^x$

- Parameter: Eine endliche abelsche Gruppe $G = (G, +), a \in G$.
- Funktion: $G \longrightarrow G \ , \ x \mapsto x \cdot a$

Hardcore-Bits

Hardcore-Bits

Idee. Es sei f eine Einwegfunktion. Dann kann man höchstens für vernachlässigbar viele x in effizienter Weise aus $f(x)$ ein Urbild von $f(x)$ berechnen.

Hardcore-Bits

Idee. Es sei f eine Einwegfunktion. Dann kann man höchstens für vernachlässigbar viele x in effizienter Weise aus $f(x)$ ein Urbild von $f(x)$ berechnen.

Aber: Es könnte sein, dass man trotzdem aus $f(x)$ Informationen über x erhalten kann.

Hardcore-Bits

Idee. Es sei f eine Einwegfunktion. Dann kann man höchstens für vernachlässigbar viele x in effizienter Weise aus $f(x)$ ein Urbild von $f(x)$ berechnen.

Aber: Es könnte sein, dass man trotzdem aus $f(x)$ Informationen über x erhalten kann.

Zum Beispiel: Das erste Bit von x könnte in $f(x)$ kodiert sein.

Hardcore-Bits

Idee. Es sei f eine Einwegfunktion. Dann kann man höchstens für vernachlässigbar viele x in effizienter Weise aus $f(x)$ ein Urbild von $f(x)$ berechnen.

Aber: Es könnte sein, dass man trotzdem aus $f(x)$ Informationen über x erhalten kann.

Zum Beispiel: Das erste Bit von x könnte in $f(x)$ kodiert sein.

Dann wäre das erste Bit kein Hardcore-Bit.

Hardcore-Bits

Definition. Es sei $f : \{0,1\}^* \longrightarrow \{0,1\}$ eine Einwegfunktion. Ein Hardcore-Bit ist eine Funktion $b : \{0,1\}^* \longrightarrow \{0,1\}$ mit:

Für alle PPT-Algorithmen $\mathcal{A}$ ist der Erfolg

$$\mid \mathbf{P}[\mathcal{A}(1^n, f(x)) = b(x)] - \frac{1}{2} \mid ,$$

wobei $x \in \{0,1\}^n$ uniform ist, vernachlässigbar in n.

Hardcore-Bits

Definition. Es sei $f : \{0,1\}^* \longrightarrow \{0,1\}$ eine Einwegfunktion. Ein Hardcore-Bit ist eine Funktion $b : \{0,1\}^* \longrightarrow \{0,1\}$ mit:

Für alle PPT-Algorithmen $\mathcal{A}$ ist der Erfolg

$$\mid \mathbf{P}[\mathcal{A}(1^n, f(x)) = b(x)] - \frac{1}{2} \mid ,$$

wobei $x \in \{0,1\}^n$ uniform ist, vernachlässigbar in n.

$$\begin{tikzcd} x \arrow[r, mapsto] \arrow[d, mapsto] & b(x) \\ f(x) \arrow[ur, mapsto, dashed, "?"] & \end{tikzcd}$$

Hardcore-Bits

$$\begin{tikzcd} x \arrow[r, mapsto] \arrow[d, mapsto] & b(x) \\ f(x) \arrow[ur, mapsto, dashed, "?"] & \end{tikzcd}$$

Hardcore-Bits

$$\begin{tikzcd}
x \arrow[r, mapsto] \arrow[d, mapsto] & b(x) \\
f(x) \arrow[ur, mapsto, dashed, "?"] &
\end{tikzcd}$$

Beispiel. Es sei $f(x_1 \cdots x_n) := x_2 \cdots x_n, \quad b(x_1 \cdots x_n) := x_1$.

Dann ist b ein Hardcore-Bit zu f.

Hardcore-Bits

$$\begin{tikzcd} x \arrow[r, mapsto] \arrow[d, mapsto] & b(x) \\ f(x) \arrow[ur, mapsto, dashed, "?"] & \end{tikzcd}$$

Beispiel. Es sei $f(x_1 \cdots x_n) := x_2 \cdots x_n, \quad b(x_1 \cdots x_n) := x_1$.
Dann ist b ein Hardcore-Bit zu f.

Allerdings: f ist nicht injektiv.

Hardcore-Bits

$$\begin{tikzcd} x \arrow[r, mapsto] \arrow[d, mapsto] & b(x) \\ f(x) \arrow[ur, mapsto, dashed, "?"] & \end{tikzcd}$$

Beispiel. Es sei $f(x_1 \cdots x_n) := x_2 \cdots x_n, \quad b(x_1 \cdots x_n) := x_1$.

Dann ist b ein Hardcore-Bit zu f.

Allerdings: f ist nicht injektiv.

Lemma. Es sei f injektiv. Wenn nun f ein Hardcore-Bit hat, dann ist f eine Einwegfunktion.

Hardcore-Bits

$$\begin{tikzcd} x \arrow[r, mapsto] \arrow[d, mapsto] & b(x) \\ f(x) \arrow[ur, mapsto, dashed, "?"] & \end{tikzcd}$$

Beispiel. Es sei $f(x_1 \cdots x_n) := x_2 \cdots x_n, \quad b(x_1 \cdots x_n) := x_1$.
Dann ist b ein Hardcore-Bit zu f.

Allerdings: f ist nicht injektiv.

Lemma. Es sei f injektiv. Wenn nun f ein Hardcore-Bit hat, dann ist f eine Einwegfunktion.

Anders ausgedrückt: Wenn f injektiv und keine Einwegfunktion ist, dann hat f kein Hardcore-Bit.

Hardcore-Bits

Satz. (Blum & Micali, 1984) Es sei g ein Generator von $(\mathbf{Z}/p\mathbf{Z})^{\times}$.

Hardcore-Bits

Satz. (Blum & Micali, 1984) Es sei g ein Generator von $(\mathbf{Z}/p\mathbf{Z})^\times$.
Wenn $\{1,\ldots,p-1\} \longrightarrow (\mathbf{Z}/p\mathbf{Z})^\times \simeq \{1,\ldots,p-1\},\ x \mapsto g^x$ eine Einweg-Funktion ist, dann ist

$$b:\ x \mapsto \begin{cases} 0, & \text{falls } x \leq \frac{p-1}{2} \\ 1, & \text{falls } x > \frac{p-1}{2} \end{cases}$$

ein Hardcore-Bit hiervon.

Hardcore-Bits

Satz. (Blum & Micali, 1984) Es sei g ein Generator von $(\mathbf{Z}/p\mathbf{Z})^\times$.
Wenn $\{1, \ldots, p-1\} \longrightarrow (\mathbf{Z}/p\mathbf{Z})^\times \simeq \{1, \ldots, p-1\}$, $x \mapsto g^x$ eine Einweg-Funktion ist, dann ist

$$b: \ x \mapsto \begin{cases} 0, & \text{falls } x \leq \frac{p-1}{2} \\ 1, & \text{falls } x > \frac{p-1}{2} \end{cases}$$

ein Hardcore-Bit hiervon.

Satz. (Goldreich & Levin, 1989) Es sei f eine Einwegfunktion. Dann ist eine uniform zufällige Linearkombination von x ein Hardcore-Bit von f.

Hardcore-Bits

Satz. (Blum & Micali, 1984) Es sei g ein Generator von $(\mathbf{Z}/p\mathbf{Z})^\times$.

Wenn $\{1, \ldots, p-1\} \longrightarrow (\mathbf{Z}/p\mathbf{Z})^\times \simeq \{1, \ldots, p-1\}$, $x \mapsto g^x$ eine Einweg-Funktion ist, dann ist

$$b: \; x \mapsto \begin{cases} 0, & \text{falls } x \leq \frac{p-1}{2} \\ 1, & \text{falls } x > \frac{p-1}{2} \end{cases}$$

ein Hardcore-Bit hiervon.

Satz. (Goldreich & Levin, 1989) Es sei f eine Einwegfunktion. Dann ist eine uniform zufällige Linearkombination von x ein Hardcore-Bit von f.

Das heißt: $(x, u) \mapsto (f(x), u)$ mit $|x| = |u|$ ist eine Einwegfunktion und $b : (x, u) \mapsto x_1 u_1 + \cdots + x_n u_n$ ist ein Hardcore-Bit hiervon.

Zum Beweis.

Hardcore-Bits

Zum Beweis.

Wir müssen zeigen:

Wenn b kein Hardcore-Bit ist, dann ist f keine Einwegfunktion.

Zum Beweis.

Wir müssen zeigen:

Wenn b kein Hardcore-Bit ist, dann ist f keine Einwegfunktion.

Wir setzen voraus, dass b kein Hardcore-Bit ist.

Zum Beweis.

Wir müssen zeigen:

Wenn b kein Hardcore-Bit ist, dann ist f keine Einwegfunktion.

Wir setzen voraus, dass b kein Hardcore-Bit ist.

b kein Hardcore-Bit heißt: Es gibt einen PTT-Algorithmus $\mathcal{A}$ für Berechnung aus $b(x, u)$ aus $(f(x), u)$ mit nicht-vernachlässigbarem Erfolg.

Zum Beweis.

Wir müssen zeigen:

Wenn b kein Hardcore-Bit ist, dann ist f keine Einwegfunktion.

Wir setzen voraus, dass b kein Hardcore-Bit ist.

b kein Hardcore-Bit heißt: Es gibt einen PTT-Algorithmus $\mathcal{A}$ für Berechnung aus $b(x, u)$ aus $(f(x), u)$ mit nicht-vernachlässigbarem Erfolg.

Es sei

$$\epsilon(n) := \mathbf{P}_{x,u}[\mathcal{A}(f(x), u) = b(x, u)] - \frac{1}{2}$$

der signierte Erfolg.

Hardcore-Bits

Zum Beweis.

Wir müssen zeigen:

Wenn b kein Hardcore-Bit ist, dann ist f keine Einwegfunktion.

Wir setzen voraus, dass b kein Hardcore-Bit ist.

b kein Hardcore-Bit heißt: Es gibt einen PTT-Algorithmus $\mathcal{A}$ für Berechnung aus $b(x,u)$ aus $(f(x),u)$ mit nicht-vernachlässigbarem Erfolg.

Es sei

$$\epsilon(n) := \mathbf{P}_{x,u}[\mathcal{A}(f(x),u) = b(x,u)] - \frac{1}{2}$$

der signierte Erfolg.

OE gibt es ein positives Polynom $p(n)$ mit $\epsilon(n) \geq \frac{1}{p(n)}$ für unendlich viele n.

Zum Beweis.

Wir müssen zeigen:

Wenn b kein Hardcore-Bit ist, dann ist f keine Einwegfunktion.

Wir setzen voraus, dass b kein Hardcore-Bit ist.

b kein Hardcore-Bit heißt: Es gibt einen PTT-Algorithmus $\mathcal{A}$ für Berechnung aus $b(x, u)$ aus $(f(x), u)$ mit nicht-vernachlässigbarem Erfolg.

Es sei

$$\epsilon(n) := \mathbf{P}_{x,u}[\mathcal{A}(f(x), u) = b(x, u)] - \frac{1}{2}$$

der signierte Erfolg.

OE gibt es ein positives Polynom $p(n)$ mit $\epsilon(n) \geq \frac{1}{p(n)}$ für unendlich viele n.

z.z.: Es gibt einen PTT-Algorithmus $\mathcal{B}$ für's Invertieren von f mit Erfolg $\geq \frac{1}{q(n)}$ für die (oder: fast alle der) genannten n ($q(n)$ ein positives Polynom).

1.Schritt.

Es sei n fixiert.

z.z.: Mit genügend großer Wahrscheinlichkeit an $x \in \{0,1\}^n$ ist

$$\mathbf{P}_u[\mathcal{A}(f(x),u) = b(x,u)] - \frac{1}{2}$$

nicht zu klein.

Hardcore-Bits

Es sei $s(x) := \mathbf{P}_u[\mathcal{A}(f(x), u) = b(x, u)]$.

Es sei $s(x) := \mathbf{P}_u[\mathcal{A}(f(x), u) = b(x, u)]$.

Wir wollen eine Aussage der Form

Mit einer Wahrscheinlichkeit von ... an x gilt $s(s) \geq \dots$

beweisen.

Hardcore-Bits

$$s(x) = \mathbf{P}_u[\mathcal{A}(f(x), u) = b(x, u)].$$

Hardcore-Bits

$$s(x) = \mathbf{P}_u[\mathcal{A}(f(x), u) = b(x, u)].$$

Es ist

$$\mathbf{P}_{x,u}[\mathcal{A}(f(x), u) = b(x, u)]$$

Hardcore-Bits

$$s(x) = \mathbf{P}_u[\mathcal{A}(f(x), u) = b(x, u)].$$

Es ist

$$\mathbf{P}_{x,u}[\mathcal{A}(f(x), u) = b(x, u)] =$$

$$\sum_{x_0 \in \{0,1\}^n} \mathbf{P}_u[\mathcal{A}(f(x_0), u) = b(x_0, u)] \cdot \mathbf{P}[x = x_0]$$

Hardcore-Bits

$$s(x) = \mathbf{P}_u[\mathcal{A}(f(x), u) = b(x, u)].$$

Es ist

$$\mathbf{P}_{x,u}[\mathcal{A}(f(x), u) = b(x, u)] =$$

$$\sum_{x_0 \in \{0,1\}^n} \mathbf{P}_u[\mathcal{A}(f(x_0), u) = b(x_0, u)] \cdot \mathbf{P}[x = x_0] =$$

$$\sum_{x_0 \in \{0,1\}^n} s(x_0) \cdot \mathbf{P}[x = x_0] = \mathbf{E}[s(x)]$$

Hardcore-Bits

$s(x) = \mathbf{P}_u[\mathcal{A}(f(x), u) = b(x, u)].$

Es ist

$$\mathbf{P}_{x,u}[\mathcal{A}(f(x), u) = b(x, u)] =$$

$$\sum_{x_0 \in \{0,1\}^n} \mathbf{P}_u[\mathcal{A}(f(x_0), u) = b(x_0, u)] \cdot \mathbf{P}[x = x_0] =$$

$$\sum_{x_0 \in \{0,1\}^n} s(x_0) \cdot \mathbf{P}[x = x_0] = \mathbf{E}[s(x)]$$

Also:

$$\mathbf{E}[s(x)] = \frac{1}{2} + \epsilon(n)$$

Hardcore-Bits

$s(x) = \mathbf{P}_u[\mathcal{A}(f(x), u) = b(x, u)]$.

Es ist

$$\mathbf{P}_{x,u}[\mathcal{A}(f(x), u) = b(x, u)] =$$

$$\sum_{x_0 \in \{0,1\}^n} \mathbf{P}_u[\mathcal{A}(f(x_0), u) = b(x_0, u)] \cdot \mathbf{P}[x = x_0] =$$

$$\sum_{x_0 \in \{0,1\}^n} s(x_0) \cdot \mathbf{P}[x = x_0] = \mathbf{E}[s(x)]$$

Also:

$$\mathbf{E}[s(x)] = \frac{1}{2} + \epsilon(n)$$

$s(x)$ nimmt Werte in $[0, 1]$ an.

Hardcore-Bits

$s(x) = \mathbf{P}_u[\mathcal{A}(f(x), u) = b(x, u)]$.

Es ist

$$\mathbf{P}_{x,u}[\mathcal{A}(f(x), u) = b(x, u)] =$$

$$\sum_{x_0 \in \{0,1\}^n} \mathbf{P}_u[\mathcal{A}(f(x_0), u) = b(x_0, u)] \cdot \mathbf{P}[x = x_0] =$$

$$\sum_{x_0 \in \{0,1\}^n} s(x_0) \cdot \mathbf{P}[x = x_0] = \mathbf{E}[s(x)]$$

Also:

$$\mathbf{E}[s(x)] = \frac{1}{2} + \epsilon(n)$$

$s(x)$ nimmt Werte in $[0, 1]$ an.

$\longrightarrow$ Informationen über die Verteilung von s.

Satz (Markov-Schranke). Es sei X eine Zufallsvariable auf $\mathbf{R}_{\geq 0}$ und $\alpha > 1$. Dann ist

$$\mathbf{P}[\,X \geq \alpha\,\mathbf{E}[X]\,] \leq \frac{1}{\alpha}\;.$$

Satz (Markov-Schranke). Es sei X eine Zufallsvariable auf $\mathbf{R}_{\geq 0}$ und $\alpha > 1$. Dann ist

$$\mathbf{P}[\,X \geq \alpha\,\mathbf{E}[X]\,] \leq \frac{1}{\alpha}\ .$$

Anwendung mit $\mathbf{E}[s(x)] = \frac{1}{2} + \epsilon(n)$:

Satz (Markov-Schranke). Es sei X eine Zufallsvariable auf $\mathbf{R}_{\geq 0}$ und $\alpha > 1$. Dann ist

$$\mathbf{P}[\, X \geq \alpha\, \mathbf{E}[X]\,] \leq \frac{1}{\alpha}\ .$$

Anwendung mit $\mathbf{E}[s(x)] = \frac{1}{2} + \epsilon(n)$:

Mit einer Wahrscheinlichkeit von $\leq$ ... gilt $s(x) \geq$ ...

Hardcore-Bits

Satz (Markov-Schranke). Es sei X eine Zufallsvariable auf $\mathbf{R}_{\geq 0}$ und $\alpha > 1$. Dann ist

$$\mathbf{P}[\, X \geq \alpha\, \mathbf{E}[X]\,] \leq \frac{1}{\alpha}\,.$$

Anwendung mit $\mathbf{E}[s(x)] = \frac{1}{2} + \epsilon(n)$:

Mit einer Wahrscheinlichkeit von $\leq$ ... gilt $s(x) \geq$ ...

Das geht in die falsche Richtung!

Es sei X eine Zufallsvariable auf $(-\infty, 1]$, $\alpha > 1$.

Dann ist

$$\mathbf{P}[\,(1 - X) \geq \alpha \mathbf{E}[1 - X]\,] \leq \frac{1}{\alpha}$$

$$\ldots = \mathbf{P}[\,1 + \alpha(\mathbf{E}[X] - 1) \geq X] = \mathbf{P}[\,X \leq 1 + \alpha(\mathbf{E}[X] - 1)\,]$$

Hardcore-Bits

Für X auf $(-\infty, 1]$, $\alpha > 1$:

$$\mathbf{P}[\, X \leq 1 + \alpha(\mathbf{E}[X] - 1)\,] \leq \frac{1}{\alpha}$$

Hardcore-Bits

Für X auf $(-\infty, 1]$, $\alpha > 1$:

$$\mathbf{P}[\, X \leq 1 + \alpha(\mathbf{E}[X] - 1)\,] \leq \frac{1}{\alpha}$$

$$\mathbf{P}[\, X > 1 + \alpha(\mathbf{E}[X] - 1)\,] > 1 - \frac{1}{\alpha}$$

Hardcore-Bits

Für X auf $(-\infty, 1]$, $\alpha > 1$:

$$\mathbf{P}[\, X \leq 1 + \alpha(\mathbf{E}[X] - 1) \,] \leq \frac{1}{\alpha}$$

$$\mathbf{P}[\, X > 1 + \alpha(\mathbf{E}[X] - 1) \,] > 1 - \frac{1}{\alpha}$$

Für $\beta := 1 - \frac{1}{\alpha}$, $\alpha = \frac{1}{1-\beta}$:

$$\mathbf{P}[\, X > 1 + \frac{1}{1-\beta}(\mathbf{E}[X] - 1) \,] > \beta$$

Hardcore-Bits

Für X auf $(-\infty, 1]$, $\alpha > 1$:

$$\mathbf{P}[\, X \leq 1 + \alpha(\mathbf{E}[X] - 1)\,] \leq \frac{1}{\alpha}$$

$$\mathbf{P}[\, X > 1 + \alpha(\mathbf{E}[X] - 1)\,] > 1 - \frac{1}{\alpha}$$

Für $\beta := 1 - \frac{1}{\alpha}$, $\alpha = \frac{1}{1-\beta}$:

$$\mathbf{P}[\, X > 1 + \frac{1}{1-\beta}(\mathbf{E}[X] - 1)\,] > \beta$$

Hardcore-Bits

Für X auf $(-\infty, 1]$, $\alpha > 1$:

$$\mathbf{P}[\, X \leq 1 + \alpha(\mathbf{E}[X] - 1)\,] \leq \frac{1}{\alpha}$$

$$\mathbf{P}[\, X > 1 + \alpha(\mathbf{E}[X] - 1)\,] > 1 - \frac{1}{\alpha}$$

Für $\beta := 1 - \frac{1}{\alpha}$, $\alpha = \frac{1}{1-\beta}$:

$$\mathbf{P}[\, X > 1 + \frac{1}{1-\beta}(\mathbf{E}[X] - 1)\,] > \beta$$

Anwendung auf $s(x)$:

$$\mathbf{E}[s(x)] = \frac{1}{2} + \epsilon(n)\ .$$

$$\mathbf{P}[\, s(x) > 1 + \frac{1}{1-\beta}(\epsilon(n) - \frac{1}{2})\,] > \beta$$

Hardcore-Bits

$$\mathbf{P}[\, s(x) > 1 + \frac{\epsilon(n) - \frac{1}{2}}{1-\beta}\,] > \beta$$

Hardcore-Bits

$$\mathbf{P}[\, s(x) > 1 + \frac{\epsilon(n) - \frac{1}{2}}{1 - \beta}\,] > \beta$$

Für $\beta = \frac{\epsilon(n)}{2}$:

$$1 + \frac{\epsilon(n) - \frac{1}{2}}{1 - \frac{\epsilon(n)}{2}} = 1 + \frac{2\epsilon(n) - 1}{2 - \epsilon(n)} = \frac{2 - \epsilon(n) + 2\epsilon(n) - 1}{2 - \epsilon(n)} = \frac{1 + \epsilon(n)}{2 - \epsilon(n)}$$

Hardcore-Bits

$$\mathbf{P}[\, s(x) > 1 + \frac{\epsilon(n) - \frac{1}{2}}{1-\beta}\,] > \beta$$

Für $\beta = \frac{\epsilon(n)}{2}$:

$$1 + \frac{\epsilon(n) - \frac{1}{2}}{1 - \frac{\epsilon(n)}{2}} = 1 + \frac{2\epsilon(n) - 1}{2 - \epsilon(n)} = \frac{2 - \epsilon(n) + 2\epsilon(n) - 1}{2 - \epsilon(n)} = \frac{1 + \epsilon(n)}{2 - \epsilon(n)}$$

Also:

$$\mathbf{P}[\, s(x) > \frac{1 + \epsilon(n)}{2 - \epsilon(n)}\,] > \frac{\epsilon(n)}{2}$$

Aus $\frac{1}{2-\epsilon(n)} \leq \frac{1}{2}$ folgt: $\frac{1+\epsilon(n)}{2-\epsilon(n)} \leq \frac{1}{2} + \frac{\epsilon(n)}{2}$, also:

Hardcore-Bits

$$\mathbf{P}[\, s(x) > 1 + \frac{\epsilon(n) - \frac{1}{2}}{1 - \beta}\,] > \beta$$

Für $\beta = \frac{\epsilon(n)}{2}$:

$$1 + \frac{\epsilon(n) - \frac{1}{2}}{1 - \frac{\epsilon(n)}{2}} = 1 + \frac{2\epsilon(n) - 1}{2 - \epsilon(n)} = \frac{2 - \epsilon(n) + 2\epsilon(n) - 1}{2 - \epsilon(n)} = \frac{1 + \epsilon(n)}{2 - \epsilon(n)}$$

Also:

$$\mathbf{P}[\, s(x) > \frac{1 + \epsilon(n)}{2 - \epsilon(n)}\,] > \frac{\epsilon(n)}{2}$$

Aus $\frac{1}{2-\epsilon(n)} \leq \frac{1}{2}$ folgt: $\frac{1+\epsilon(n)}{2-\epsilon(n)} \leq \frac{1}{2} + \frac{\epsilon(n)}{2}$, also:

$$\mathbf{P}[\, s(x) > \frac{1}{2} + \frac{\epsilon(n)}{2}\,] > \frac{\epsilon(n)}{2}$$

Hardcore-Bits

Wir haben nun genug “gute” x.

Hardcore-Bits

Wir haben nun genug “gute” x.

2.Schritt.

Bestimmen von x aus $y = f(x)$:

Wir haben nun genug “gute” x.

2.Schritt.

Bestimmen von x aus $y = f(x)$:

Idee. Angonommen, wir kennen alle $b(x, u)$. Dann:

Hardcore-Bits

Wir haben nun genug “gute” x.

2.Schritt.

Bestimmen von x aus $y = f(x)$:

Idee. Angonommen, wir kennen alle $b(x, u)$. Dann:

Für $j = 1, \ldots, n$:

1. Wähle $u \in \{0, 1\}^n$ unform.
2. Berechne $b(x, r) = \sum_i x_i u_i$, $b(x, u + e_j) = \sum_i x_i u_i + x_j$, erhalte:
$$b(x, u) + b(x, u + e_j) = x_j$$

Wir haben nun genug "gute" x.

2.Schritt.

Bestimmen von x aus $y = f(x)$:

Idee. Angonommen, wir kennen alle $b(x, u)$. Dann:

Für $j = 1, \ldots, n$:

1. Wähle $u \in \{0,1\}^n$ unform.
2. Berechne $b(x, r) = \sum_i x_i u_i$, $b(x, u + e_j) = \sum_i x_i u_i + x_j$, erhalte:
$$b(x, u) + b(x, u + e_j) = x_j$$

Problem. Wir können nicht $b(x, r), b(x, u + e_j)$ berechnen, sondern nur $\mathcal{A}(y, u)$.

Hardcore-Bits

Wir haben nun genug "gute" x.

2.Schritt.

Bestimmen von x aus $y = f(x)$:

Idee. Angonommen, wir kennen alle $b(x, u)$. Dann:

Für $j = 1, \ldots, n$:

1. Wähle $u \in \{0, 1\}^n$ unform.
2. Berechne $b(x, r) = \sum_i x_i u_i$, $b(x, u + e_j) = \sum_i x_i u_i + x_j$, erhalte:
$$b(x, u) + b(x, u + e_j) = x_j$$

Problem. Wir können nicht $b(x, r), b(x, u + e_j)$ berechnen, sondern nur $\mathcal{A}(y, u)$.

Die Wahrscheinlichkeit, dass $\mathcal{A}$ beide Ergebnisse richtig berechnet, ist $\geq \epsilon(n)$.

Wir haben nun genug "gute" x.

2.Schritt.

Bestimmen von x aus $y = f(x)$:

Idee. Angonommen, wir kennen alle $b(x, u)$. Dann:

Für $j = 1, \ldots, n$:

1. Wähle $u \in \{0, 1\}^n$ unform.
2. Berechne $b(x, r) = \sum_i x_i u_i$, $b(x, u + e_j) = \sum_i x_i u_i + x_j$, erhalte:
$$b(x, u) + b(x, u + e_j) = x_j$$

Problem. Wir können nicht $b(x, r), b(x, u + e_j)$ berechnen, sondern nur $\mathcal{A}(y, u)$.

Die Wahrscheinlichkeit, dass $\mathcal{A}$ beide Ergebnisse richtig berechnet, ist $\geq \epsilon(n)$.

Das ist absurd niedrig!

Weitere Idee. Die Anzahl der u, für welche $b(x, u)$ berechnet werden muss, reduzieren durch Verwendung von Linearkombinationen:

Weitere Idee. Die Anzahl der u, für welche $b(x, u)$ berechnet werden muss, reduzieren durch Verwendung von Linearkombinationen:

1. Wähle ein linear unabhängiges System $u_1, \ldots, u_\ell$ mit ℓ polynomiell in $\log_2(n)$.
2. Rate die $b(x, u_i)$.

Weitere Idee. Die Anzahl der u, für welche $b(x, u)$ berechnet werden muss, reduzieren durch Verwendung von Linearkombinationen:

1. Wähle ein linear unabhängiges System $u_1, \ldots, u_\ell$ mit ℓ polynomiell in $\log_2(n)$.
2. Rate die $b(x, u_i)$. (Seien σ_i die geratenen Werte).

Weitere Idee. Die Anzahl der u, für welche $b(x, u)$ berechnet werden muss, reduzieren durch Verwendung von Linearkombinationen:

1. Wähle ein linear unabhängiges System $u_1, \ldots, u_\ell$ mit ℓ polynomiell in $\log_2(n)$.
2. Rate die $b(x, u_i)$. (Seien σ_i die geratenen Werte).

Um x_j zu berechnen:

Weitere Idee. Die Anzahl der u, für welche $b(x, u)$ berechnet werden muss, reduzieren durch Verwendung von Linearkombinationen:

1. Wähle ein linear unabhängiges System $u_1, \ldots, u_\ell$ mit ℓ polynomiell in $\log_2(n)$.
2. Rate die $b(x, u_i)$. (Seien σ_i die geratenen Werte).

Um x_j zu berechnen:

1. Wähle Linearkombination $u = a_1 u_1 + \cdots + a_\ell u_\ell$.

Weitere Idee. Die Anzahl der u, für welche $b(x, u)$ berechnet werden muss, reduzieren durch Verwendung von Linearkombinationen:

1. Wähle ein linear unabhängiges System $u_1, \ldots, u_\ell$ mit ℓ polynomiell in $\log_2(n)$.
2. Rate die $b(x, u_i)$. (Seien σ_i die geratenen Werte).

Um x_j zu berechnen:

1. Wähle Linearkombination $u = a_1 u_1 + \cdots + a_\ell u_\ell$.
2. Gib die Antwort $(a_1\sigma_1 + \cdots + a_\ell\sigma_\ell) + \mathcal{A}(y, u + e_j)$.

Hardcore-Bits

Die verwendeten u's sind nicht stochastisch unabhängig ...

Hardcore-Bits

Die verwendeten u's sind nicht stochastisch unabhängig ...

... aber mit der Tschebyscheff-Ungleichung kann man die Analyse beenden.

Pseudozufallsgeneratoren

Pseudozufallsgeneratoren

Idee. Wir wollen echten Zufall verstärken,

Pseudozufallsgeneratoren

Idee. Wir wollen echten Zufall verstärken,

aus einem kleinen "echt" zufälligen String einen größeren machen, der nicht effizient unterscheidbar von einem "echt" zufälligen ist.

Pseudozufallsgeneratoren

Idee. Wir wollen echten Zufall verstärken,

aus einem kleinen "echt" zufälligen String einen größeren machen, der nicht effizient unterscheidbar von einem "echt" zufälligen ist.

"echt zufällig" ist ungenau. Genau: uniform zufällig.

Pseudozufallsgeneratoren

Definition. Ein Pseudozufallsgenerator (mit fester Ausgabelänge) ist ein deterministischer Polynomzeit-Algorithmus $\mathcal{G}$, der

- unter Eingabe eines Strings $s \in \{0,1\}^n$ einen String einer festen Länge $\ell(n) > n$ ausgibt,
- wobei der String nicht effizient unterscheidbar von einem echt zufälligen String der Länge $\ell(n)$ ist

Pseudozufallsgeneratoren

Definition. Ein Pseudozufallsgenerator (mit fester Ausgabelänge) ist ein deterministischer Polynomzeit-Algorithmus $\mathcal{G}$, der

- unter Eingabe eines Strings $s \in \{0,1\}^n$ einen String einer festen Länge $\ell(n) > n$ ausgibt,
- wobei der String nicht effizient unterscheidbar von einem echt zufälligen String der Länge $\ell(n)$ ist:

Für alle PPT-Algorithmen (genannt Unterscheider (=distinguisher)) $\mathcal{D}$ ist der Erfolg von $\mathcal{D}$

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

vernachlässigbar

Definition. Ein Pseudozufallsgenerator (mit fester Ausgabelänge) ist ein deterministischer Polynomzeit-Algorithmus $\mathcal{G}$, der

- unter Eingabe eines Strings $s \in \{0,1\}^n$ einen String einer festen Länge $\ell(n) > n$ ausgibt,
- wobei der String nicht effizient unterscheidbar von einem echt zufälligen String der Länge $\ell(n)$ ist:

Für alle PPT-Algorithmen (genannt Unterscheider (=distinguisher)) $\mathcal{D}$ ist der Erfolg von $\mathcal{D}$

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

vernachlässigbar,

wobei $s \in \{0,1\}^n$ und $r \in \{0,1\}^{\ell(n)}$ uniform zufällig sind.

Pseudozufallsgeneratoren

Wir fixieren nun eine Einwegpermutation f.

Wir fixieren nun eine Einwegpermutation f.

Vermutetes Beispiel. (Für Familie von Einwegpermutationen)

Parameter: Eine Primzahl p und ein Erzeuger g von $(\mathbf{Z}/p\mathbf{Z})^{\times}$.

$$\{1, \ldots, p-1\} \longrightarrow (\mathbf{Z}/p\mathbf{Z})^{\times} \simeq \{1, \ldots, p-1\}\ ,\ x \mapsto g^x$$

Es sei nun f eine Einwegpermutation mit einem Hardcore-Bit b.

Aussage. Die Funktion $s \mapsto (f(s), b(s))$ definiert einen Pseudozufallsgenerator (mit $\ell(n) = n + 1$).

Es sei nun f eine Einwegpermutation mit einem Hardcore-Bit b.

Aussage. Die Funktion $s \mapsto (f(s), b(s))$ definiert einen Pseudozufallsgenerator (mit $\ell(n) = n + 1$).

Beweisidee. Für $s \in \{0,1\}^n$ uniform zufällig ist auch $f(s) \in \{0,1\}^n$ uniform zufällig.

Also ist "höchstens das letzte Bit problematisch".

Pseudozufallsgeneratoren

Es sei nun f eine Einwegpermutation mit einem Hardcore-Bit b.

Aussage. Die Funktion $s \mapsto (f(s), b(s))$ definiert einen Pseudozufallsgenerator (mit $\ell(n) = n + 1$).

Beweisidee. Für $s \in \{0,1\}^n$ uniform zufällig ist auch $f(s) \in \{0,1\}^n$ uniform zufällig.

Also ist "höchstens das letzte Bit problematisch".

Angenommen, man kann $(f(s), b(s))$ von $(r_1, \ldots, r_n, r_{n+1})$ effizient unterscheiden.

Es sei nun f eine Einwegpermutation mit einem Hardcore-Bit b.

Aussage. Die Funktion $s \mapsto (f(s), b(s))$ definiert einen Pseudozufallsgenerator (mit $\ell(n) = n + 1$).

Beweisidee. Für $s \in \{0,1\}^n$ uniform zufällig ist auch $f(s) \in \{0,1\}^n$ uniform zufällig.

Also ist "höchstens das letzte Bit problematisch".

Angenommen, man kann $(f(s), b(s))$ von $(r_1, \ldots, r_n, r_{n+1})$ effizient unterscheiden.

Dann kann man $(f(s), b(s))$ effizient von $(f(s), r)$ unterscheiden.

Es sei nun f eine Einwegpermutation mit einem Hardcore-Bit b.

Aussage. Die Funktion $s \mapsto (f(s), b(s))$ definiert einen Pseudozufallsgenerator (mit $\ell(n) = n + 1$).

Beweisidee. Für $s \in \{0,1\}^n$ uniform zufällig ist auch $f(s) \in \{0,1\}^n$ uniform zufällig.

Also ist "höchstens das letzte Bit problematisch".

Angenommen, man kann $(f(s), b(s))$ von $(r_1, \ldots, r_n, r_{n+1})$ effizient unterscheiden.

Dann kann man $(f(s), b(s))$ effizient von $(f(s), r)$ unterscheiden. Man kann $b(s)$ effizient aus $f(s)$ berechnen.

Es sei nun f eine Einwegpermutation mit einem Hardcore-Bit b.

Aussage. Die Funktion $s \mapsto (f(s), b(s))$ definiert einen Pseudozufallsgenerator (mit $\ell(n) = n + 1$).

Beweisidee. Für $s \in \{0,1\}^n$ uniform zufällig ist auch $f(s) \in \{0,1\}^n$ uniform zufällig.

Also ist "höchstens das letzte Bit problematisch".

Angenommen, man kann $(f(s), b(s))$ von $(r_1, \ldots, r_n, r_{n+1})$ effizient unterscheiden.

Dann kann man $(f(s), b(s))$ effizient von $(f(s), r)$ unterscheiden.
Man kann $b(s)$ effizient aus $f(s)$ berechnen.

Das ist aber ausgeschlossen.

Pseudozufallsgeneratoren

Es sei $\mathcal{G}$ ein Pseudozufallsgenator. Es sei

$$\mathcal{G}(s_1, \ldots, s_n) = (\underbrace{\mathcal{I}(s)}_{n \text{ bit}}, \underbrace{\mathcal{O}(s)}_{\ell - n \text{ bit}}) \; .$$

Pseudozufallsgeneratoren

Es sei $\mathcal{G}$ ein Pseudozufallsgenator. Es sei

$$\mathcal{G}(s_1, \ldots, s_n) = (\underbrace{\mathcal{I}(s)}_{n \text{ bit}}, \underbrace{\mathcal{O}(s)}_{\ell - n \text{ bit}}) \ .$$

Wir betrachten diesen Algorithmus:

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

Wiederhole:

1. Ausgabe $\mathcal{O}(s)$
2. $s := \mathcal{I}(s)$

(Abbruch nach polynomiell vielen Schritten.)

Es sei $\mathcal{G}$ ein Pseudozufallsgenator. Es sei

$$\mathcal{G}(s_1, \ldots, s_n) = (\underbrace{\mathcal{I}(s)}_{n \text{ bit}}, \underbrace{\mathcal{O}(s)}_{\ell - n \text{ bit}}) \ .$$

Wir betrachten diesen Algorithmus:

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

Wiederhole:

1. Ausgabe $\mathcal{O}(s)$
2. $s := \mathcal{I}(s)$

(Abbruch nach polynomiell vielen Schritten.)

Satz. Der soeben definierte Algorithmus ist ein Pseudozufallsgenerator.

Definitionen für Verschlüsselungssysteme mittels Spielen

Definition

Ein Verschlüsselungssystem mit nicht-öffentlichen Schlüsseln besteht aus drei randomisierten Algorithmen:

Definition

Ein Verschlüsselungssystem mit nicht-öffentlichen Schlüsseln besteht aus drei randomisierten Algorithmen:

- Einem Schlüsselerzeugungsalgorithmus $\mathcal{G}en$.

  Eingabe: 1^n mit $n =$ Sicherheitsparameter

  Ausgabe: Schlüssel $k = \mathcal{G}en(1^n)$

Definition

Ein Verschlüsselungssystem mit nicht-öffentlichen Schlüsseln besteht aus drei randomisierten Algorithmen:

- Einem Schlüsselerzeugungsalgorithmus $\mathcal{G}en$.

  Eingabe: 1^n mit $n =$ Sicherheitsparameter

  Ausgabe: Schlüssel $k = \mathcal{G}en(1^n)$

- Einem Verschlüsselungsalgorithmus $\mathcal{E}nc$.

  Eingabe: Schlüssel k,
  Nachricht $m \in \{0,1\}^*$ oder $m \in \{0,1\}^{\ell(n)}$.

  Ausgabe: Chiffriertext $c = \mathcal{E}nc_k(m)$

Definition

Ein Verschlüsselungssystem mit nicht-öffentlichen Schlüsseln besteht aus drei randomisierten Algorithmen:

- Einem Schlüsselerzeugungsalgorithmus $\mathcal{G}en$.

  Eingabe: 1^n mit $n =$ Sicherheitsparameter

  Ausgabe: Schlüssel $k = \mathcal{G}en(1^n)$
- Einem Verschlüsselungsalgorithmus $\mathcal{E}nc$.

  Eingabe: Schlüssel k,
  Nachricht $m \in \{0,1\}^*$ oder $m \in \{0,1\}^{\ell(n)}$.

  Ausgabe: Chiffriertext $c = \mathcal{E}nc_k(m)$
- Einem Entschlüsselungsalgorithmus $\mathcal{D}ec$.

  Eingabe: Schlüssel k, Chiffriertext c

  Ausgabe: $\mathcal{D}ec_k(c)$

Definition

Ein Verschlüsselungssystem mit nicht-öffentlichen Schlüsseln besteht aus drei randomisierten Algorithmen:

- Einem Schlüsselerzeugungsalgorithmus $\mathcal{G}en$.

  Eingabe: 1^n mit $n =$ Sicherheitsparameter

  Ausgabe: Schlüssel $k = \mathcal{G}en(1^n)$
- Einem Verschlüsselungsalgorithmus $\mathcal{E}nc$.

  Eingabe: Schlüssel k,
  Nachricht $m \in \{0,1\}^*$ oder $m \in \{0,1\}^{\ell(n)}$.

  Ausgabe: Chiffriertext $c = \mathcal{E}nc_k(m)$
- Einem Entschlüsselungsalgorithmus $\mathcal{D}ec$.

  Eingabe: Schlüssel k, Chiffriertext c

  Ausgabe: $\mathcal{D}ec_k(c)$

mit $\mathcal{D}ec_k(\mathcal{E}nc_k(m)) = m$ für alle relevanten k, m.

Die Laufzeiten sollen polynomiell sein.

Ideen.

Ein erfolgreicher Angreifer sollte “was rauskriegen” können, was über reines Raten hinausgeht.

Ideen.

Ein erfolgreicher Angreifer sollte “was rauskriegen” können, was über reines Raten hinausgeht.

Für ein wirklich hohes Niveau von Sicherheit sagen wir, dass der Angreifer selbst entscheiden kann, was er “rauskriegen” will.

Ideen.

Ein erfolgreicher Angreifer sollte “was rauskriegen” können, was über reines Raten hinausgeht.

Für ein wirklich hohes Niveau von Sicherheit sagen wir, dass der Angreifer selbst entscheiden kann, was er “rauskriegen” will.

Ein Herausforder testet, ob dies der Fall ist.

Ein einfacher Test / ein einfaches Spiel

- Der Angreifer wählt zwei Nachrichten m_1, m_2,
  schickt diese an den Herausforderer.
- Dieser wählt eine davon aus,
  berechnet den Chiffiertext,
  schickt diesen an den Angreifer.

Ein einfacher Test / ein einfaches Spiel

- Der Angreifer wählt zwei Nachrichten m_1, m_2,
  schickt diese an den Herausforderer.
- Dieser wählt eine davon aus,
  berechnet den Chiffiertext,
  schickt diesen an den Angreifer.
- Der Angreifer tippt, welche Nachricht verschlüsselt worden ist.
- Der Angreifer hat gewonnen, wenn er richtig getippt hat.

Ein einfacher Test / ein einfaches Spiel

- Der Angreifer wählt zwei Nachrichten m_1, m_2,
  schickt diese an den Herausforderer.
- Dieser wählt eine davon aus,
  berechnet den Chiffiertext,
  schickt diesen an den Angreifer.
- Der Angreifer tippt, welche Nachricht verschlüsselt worden ist.
- Der Angreifer hat gewonnen, wenn er richtig getippt hat.

Ein “guter Angreifer” sollte deutlich häufiger als jedes zweite Mal gewinnen.

Das Spiel zum Lauscher-Angriff

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

Das Spiel zum Lauscher-Angriff

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

Eingabe: Sicherheitsparameter 1^n.

Das Spiel zum Lauscher-Angriff

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.

Das Spiel zum Lauscher-Angriff

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 $k := \mathcal{G}en(1^n)$.

Das Spiel zum Lauscher-Angriff

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 wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.

Das Spiel zum Lauscher-Angriff

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 wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{1, 2\}$ uniform (und unabhängig vom Bisherigen). Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.

Das Spiel zum Lauscher-Angriff

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 wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{1, 2\}$ uniform (und unabhängig vom Bisherigen). Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.

Das Spiel zum Lauscher-Angriff

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 wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{1, 2\}$ uniform (und unabhängig vom Bisherigen). Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.
6. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Das Spiel zum Lauscher-Angriff

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 wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{1, 2\}$ uniform (und unabhängig vom Bisherigen). Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.
6. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Der Erfolg eines Angreifers $\mathcal{A}$ ist definiert als

$$|\mathbf{P}[\mathcal{A} \text{ gewinnt}] - \frac{1}{2}| \, .$$

Das Spiel zum Lauscher-Angriff

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 wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{1, 2\}$ uniform (und unabhängig vom Bisherigen). Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.
6. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Wenn für jeden Angreifer der Erfolg 0 ist, heißt das Verfahren perfekt sicher.

Das Spiel zum Lauscher-Angriff

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 wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{1, 2\}$ uniform (und unabhängig vom Bisherigen). Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.
6. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Wenn für jeden PPT-Angreifer der Erfolg vernachlässigbar ist, heißt das Verfahren IND-EAV-sicher.

Das Spiel zum CPA

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

Das Spiel zum CPA

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

Eingabe: Sicherheitsparameter 1^n.

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.

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 $k := \mathcal{G}en(1^n)$.

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).

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.

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.

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.

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 $j \in \{1, 2\}$.

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 $j \in \{1, 2\}$.
8. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Das Spiel zum nicht-adaptiven CCA

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

Das Spiel zum nicht-adaptiven CCA

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

Eingabe: Sicherheitsparameter 1^n.

Das Spiel zum nicht-adaptiven CCA

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.

Das Spiel zum nicht-adaptiven CCA

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 $k := \mathcal{G}en(1^n)$.

Das Spiel zum nicht-adaptiven CCA

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 Texte ver- und entschlüsseln lassen.

Das Spiel zum nicht-adaptiven CCA

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 Texte ver- und entschlüsseln lassen.
4. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.

Das Spiel zum nicht-adaptiven CCA

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 Texte ver- und entschlüsseln 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.

Das Spiel zum nicht-adaptiven CCA

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 Texte ver- und entschlüsseln 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 verschlüsseln lassen.

Das Spiel zum nicht-adaptiven CCA

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 Texte ver- und entschlüsseln 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 verschlüsseln lassen.
7. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.

Das Spiel zum nicht-adaptiven CCA

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 Texte ver- und entschlüsseln 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 verschlüsseln lassen.
7. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.
8. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Das Spiel zum adaptiven CCA (CCA2 / CCA)

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 Texte ver- und entschlüsseln 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.

Das Spiel zum adaptiven CCA (CCA2 / CCA)

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 Texte ver- und entschlüsseln 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 verschlüsseln lassen

Das Spiel zum adaptiven CCA (CCA2 / CCA)

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 Texte ver- und entschlüsseln 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 verschlüsseln lassen und Chiffriertexte verschieden von c entschlüsseln lassen.

Das Spiel zum adaptiven CCA (CCA2 / CCA)

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 Texte ver- und entschlüsseln 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 verschlüsseln lassen und Chiffriertexte verschieden von c entschlüsseln lassen.
7. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Definition

Ein Verschlüsselungssystem mit öffentlichen Schlüsseln besteht aus drei randomisierten Algorithmen:

Definition

Ein Verschlüsselungssystem mit öffentlichen Schlüsseln besteht aus drei randomisierten Algorithmen:

- Einem Schlüsselerzeugungsalgorithmus $\mathcal{G}en$.

  Eingabe: Sicherheitsparameter 1^n

  Ausgabe: Schlüsselpaar $(pk, sk) = \mathcal{G}en(1^n)$

Definition

Ein Verschlüsselungssystem mit öffentlichen Schlüsseln besteht aus drei randomisierten Algorithmen:

- Einem Schlüsselerzeugungsalgorithmus $\mathcal{G}en$.

  Eingabe: Sicherheitsparameter 1^n

  Ausgabe: Schlüsselpaar $(pk, sk) = \mathcal{G}en(1^n)$
- Einem Verschlüsselungsalgorithmus $\mathcal{E}nc$.

  Eingabe: öffentlicher Schlüssel pk,
  Nachricht $m \in \{0,1\}^*$ oder $m \in \{0,1\}^{\ell(n)}$.

  Ausgabe: Chiffriertext $c = \mathcal{E}nc_{pk}(m)$

Definition

Ein Verschlüsselungssystem mit öffentlichen Schlüsseln besteht aus drei randomisierten Algorithmen:

- Einem Schlüsselerzeugungsalgorithmus $\mathcal{G}en$.

  Eingabe: Sicherheitsparameter 1^n

  Ausgabe: Schlüsselpaar $(pk, sk) = \mathcal{G}en(1^n)$
- Einem Verschlüsselungsalgorithmus $\mathcal{E}nc$.

  Eingabe: öffentlicher Schlüssel pk,
  Nachricht $m \in \{0,1\}^*$ oder $m \in \{0,1\}^{\ell(n)}$.

  Ausgabe: Chiffriertext $c = \mathcal{E}nc_{pk}(m)$
- Einem Entschlüsselungsalgorithmus $\mathcal{D}ec$.

  Eingabe: Privater Schlüssel sk, Chiffriertext c

  Ausgabe: $\mathcal{D}ec_{sk}(m)$

Definition

Ein Verschlüsselungssystem mit öffentlichen Schlüsseln besteht aus drei randomisierten Algorithmen:

- Einem Schlüsselerzeugungsalgorithmus $\mathcal{G}en$.

  Eingabe: Sicherheitsparameter 1^n

  Ausgabe: Schlüsselpaar $(pk, sk) = \mathcal{G}en(1^n)$
- Einem Verschlüsselungsalgorithmus $\mathcal{E}nc$.

  Eingabe: öffentlicher Schlüssel pk,
  Nachricht $m \in \{0,1\}^*$ oder $m \in \{0,1\}^{\ell(n)}$.

  Ausgabe: Chiffriertext $c = \mathcal{E}nc_{pk}(m)$
- Einem Entschlüsselungsalgorithmus $\mathcal{D}ec$.

  Eingabe: Privater Schlüssel sk, Chiffriertext c

  Ausgabe: $\mathcal{D}ec_{sk}(m)$

mit $\mathcal{D}ec_{sk}(\mathcal{E}nc_{pk}(m)) = m$ für (fast) alle relevanten Schlüsselpaare (pk, sk) und Nachrichten m.

Ein einfaches Ununterscheidbarkeits-Spiel

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

Ein einfaches Ununterscheidbarkeits-Spiel

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

Eingabe: Sicherheitsparameter 1^n.

Ein einfaches Ununterscheidbarkeits-Spiel

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.

Ein einfaches Ununterscheidbarkeits-Spiel

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.

Ein einfaches Ununterscheidbarkeits-Spiel

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.

Ein einfaches Ununterscheidbarkeits-Spiel

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{1, 2\}$ uniform. Er berechnet $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.

Ein einfaches Ununterscheidbarkeits-Spiel

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{1, 2\}$ uniform. Er berechnet $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.
6. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Das Ununterscheidbarkeits-Spiel zu EAV und CPA

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{1, 2\}$ uniform. Er berechnet $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.
6. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Das Ununterscheidbarkeits-Spiel zu EAV und CPA

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer wählt zwei verschiedene Nachrichten m_1, m_2 gleicher Länge. Er schickt diese an den Herausforderer.
4. Der Herausforderer wählt $i \in \{1, 2\}$ uniform. Er berechnet $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.
6. Der Angreifer hat gewonnen, wenn $i = j$ ist.

IND-EAV = IND-CPA

Das Spiel zum nicht-adaptiven CCA

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

Das Spiel zum nicht-adaptiven CCA

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer kann beliebige Chiffriertext entschlüsseln 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 $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.

Das Spiel zum nicht-adaptiven CCA

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt einen Schlüssel $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer kann beliebige Chiffriertext entschlüsseln 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 $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.
6. Der Angreifer entscheidet sich für $j \in \{0, 1\}$.
7. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Das Spiel zum adaptiven CCA (CCA2 / CCA)

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer kann beliebige Texte entschlüsseln 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 $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.

Das Spiel zum adaptiven CCA (CCA2 / CCA)

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer kann beliebige Texte entschlüsseln 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 $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.
6. Der Angreifer kann beliebige Chiffriertexte verschieden von c entschlüsseln lassen.

Das Spiel zum adaptiven CCA (CCA2 / CCA)

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer kann beliebige Texte entschlüsseln 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 $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.
6. Der Angreifer kann beliebige Chiffriertexte verschieden von c entschlüsseln lassen.
7. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.

Das Spiel zum adaptiven CCA (CCA2 / CCA)

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer kann beliebige Texte entschlüsseln 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 $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.
6. Der Angreifer kann beliebige Chiffriertexte verschieden von c entschlüsseln lassen.
7. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.
8. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Das Spiel zum adaptiven CCA (CCA2 / CCA)

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

Eingabe: Sicherheitsparameter 1^n.

1. Angreifer und Herausforderer erhalten 1^n.
2. Der Herausforderer erzeugt ein Schlüsselpaar $(pk, sk) := \mathcal{G}en(1^n)$ und schickt pk an den Angreifer.
3. Der Angreifer kann beliebige Texte entschlüsseln 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 $c := \mathcal{E}nc_{pk}(m_i)$ und schickt c an den Angreifer.
6. Der Angreifer kann beliebige Chiffriertexte verschieden von c entschlüsseln lassen.
7. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.
8. Der Angreifer hat gewonnen, wenn $i = j$ ist.

Ein IND-EAV-sicheres Verfahren

Ein IND-EAV-sicheres Verfahren

... mit nicht-öffentlichen Schlüsseln.

Ein IND-EAV-sicheres Verfahren

... mit nicht-öffentlichen Schlüsseln.

Das one-time pad

$\mathcal{G}en$: Wähle $k \in \{0,1\}^n$ uniform zufällig

Ein IND-EAV-sicheres Verfahren

... mit nicht-öffentlichen Schlüsseln.

Das one-time pad

$\mathcal{G}en$: Wähle $k \in \{0,1\}^n$ uniform zufällig

$\mathcal{E}nc$: Gegeben $m \in \{0,1\}^n$, gib $c := m \oplus k \in \{0,1\}^n$ aus.

Ein IND-EAV-sicheres Verfahren

... mit nicht-öffentlichen Schlüsseln.

Das one-time pad

$\mathcal{G}en$: Wähle $k \in \{0,1\}^n$ uniform zufällig

$\mathcal{E}nc$: Gegeben $m \in \{0,1\}^n$, gib $c := m \oplus k \in \{0,1\}^n$ aus.

$\mathcal{D}ec$: Gegeben $c \in \{0,1\}^n$, gib $m := c \oplus k \in \{0,1\}^n$ aus.

Ein IND-EAV-sicheres Verfahren

Das pseudo-one time pad

Es sei $\mathcal{G}$ ein Pseudozufallsgenerator.

$\mathcal{G}en$: Wähle $k \in \{0,1\}^n$ uniform zufällig

$\mathcal{E}nc$: Gegeben $m \in \{0,1\}^{\ell(n)}$, gib $c := m \oplus \mathcal{G}(k) \in \{0,1\}^n$ aus.

$\mathcal{D}ec$: Gegeben $c \in \{0,1\}^{\ell(n)}$, gib $m := c \oplus \mathcal{G}(k) \in \{0,1\}^n$ aus.

Ein IND-EAV-sicheres Verfahren

Das pseudo-one time pad

Es sei $\mathcal{G}$ ein Pseudozufallsgenerator.

$\mathcal{G}en$: Wähle $k \in \{0,1\}^n$ uniform zufällig

$\mathcal{E}nc$: Gegeben $m \in \{0,1\}^{\ell(n)}$, gib $c := m \oplus \mathcal{G}(k) \in \{0,1\}^n$ aus.

$\mathcal{D}ec$: Gegeben $c \in \{0,1\}^{\ell(n)}$, gib $m := c \oplus \mathcal{G}(k) \in \{0,1\}^n$ aus.

Satz. Wenn $\mathcal{G}$ ein Pseudozufallsgenerator ist, ist das Verfahren IND-EAV-sicher.

Beweis.
Wir müssen zeigen:

Wenn das Verfahren nicht IND-EAV sicher ist, dann ist $\mathcal{G}$ kein Pseudozufallsgenerator.

Ein IND-EAV-sicheres Verfahren

Beweis.
Wir müssen zeigen:

Wenn das Verfahren nicht IND-EAV sicher ist, dann ist $\mathcal{G}$ kein Pseudozufallsgenerator.

Das heißt: Wenn es einen PPT-Angreifer mit nicht-vernachlässigbarem Erfolg gibt, dann gibt es einen PPT-Unterscheider zu $\mathcal{G}$ mit nicht-vernachlässigbarem Erfolg.

Ein IND-EAV-sicheres Verfahren

Beweis.
Wir müssen zeigen:

Wenn das Verfahren nicht IND-EAV sicher ist, dann ist $\mathcal{G}$ kein Pseudozufallsgenerator.

Das heißt: Wenn es einen PPT-Angreifer mit nicht-vernachlässigbarem Erfolg gibt, dann gibt es einen PPT-Unterscheider zu $\mathcal{G}$ mit nicht-vernachlässigbarem Erfolg.

Strategie:

Wir transformieren jeden Angreifer zu einem Unterscheider mit demselben Erfolg und vergleichbarer Laufzeit.

Ein IND-EAV-sicheres Verfahren

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

Ein IND-EAV-sicheres Verfahren

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

Unser Unterscheider $\mathcal{D}$:

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

Ein IND-EAV-sicheres Verfahren

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

Unser Unterscheider $\mathcal{D}$:

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

1. Wähle $m_1, m_2 \in \{0,1\}^{\ell(n)}$ mittels $\mathcal{A}$.

Ein IND-EAV-sicheres Verfahren

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

Unser Unterscheider $\mathcal{D}$:

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

1. Wähle $m_1, m_2 \in \{0,1\}^{\ell(n)}$ mittels $\mathcal{A}$.
2. Wähle $i = 1, 2$ uniform zufällig, setze $m := m_i$.

Ein IND-EAV-sicheres Verfahren

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

Unser Unterscheider $\mathcal{D}$:

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

1. Wähle $m_1, m_2 \in \{0,1\}^{\ell(n)}$ mittels $\mathcal{A}$.
2. Wähle $i = 1, 2$ uniform zufällig, setze $m := m_i$.
3. Gib $c := m \oplus w$ an $\mathcal{A}$.

Ein IND-EAV-sicheres Verfahren

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

Unser Unterscheider $\mathcal{D}$:

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

1. Wähle $m_1, m_2 \in \{0,1\}^{\ell(n)}$ mittels $\mathcal{A}$.
2. Wähle $i = 1, 2$ uniform zufällig, setze $m := m_i$.
3. Gib $c := m \oplus w$ an $\mathcal{A}$.
4. $\mathcal{A}$ entscheidet sich für $j = 1, 2$.

Ein IND-EAV-sicheres Verfahren

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

Unser Unterscheider $\mathcal{D}$:

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

1. Wähle $m_1, m_2 \in \{0,1\}^{\ell(n)}$ mittels $\mathcal{A}$.
2. Wähle $i = 1, 2$ uniform zufällig, setze $m := m_i$.
3. Gib $c := m \oplus w$ an $\mathcal{A}$.
4. $\mathcal{A}$ entscheidet sich für $j = 1, 2$.
5. Wenn $j = i$ ($\mathcal{A}$ hat gewonnen), gib 1 (“nicht zufällig”) aus, ansonsten 0 (“zufällig”).

Ein IND-EAV-sicheres Verfahren

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

Unser Unterscheider $\mathcal{D}$:

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

1. Wähle $m_1, m_2 \in \{0,1\}^{\ell(n)}$ mittels $\mathcal{A}$.
2. Wähle $i = 1, 2$ uniform zufällig, setze $m := m_i$.
3. Gib $c := m \oplus w$ an $\mathcal{A}$.
4. $\mathcal{A}$ entscheidet sich für $j = 1, 2$.
5. Wenn $j = i$ ($\mathcal{A}$ hat gewonnen), gib 1 ("nicht zufällig") aus, ansonsten 0 ("zufällig").

- Wenn $w = r$ uniform ist:

  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei one-time-pad}] = \frac{1}{2}$

Ein IND-EAV-sicheres Verfahren

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

Unser Unterscheider $\mathcal{D}$:

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

1. Wähle $m_1, m_2 \in \{0,1\}^{\ell(n)}$ mittels $\mathcal{A}$.
2. Wähle $i = 1, 2$ uniform zufällig, setze $m := m_i$.
3. Gib $c := m \oplus w$ an $\mathcal{A}$.
4. $\mathcal{A}$ entscheidet sich für $j = 1, 2$.
5. Wenn $j = i$ ($\mathcal{A}$ hat gewonnen), gib 1 ("nicht zufällig") aus, ansonsten 0 ("zufällig").

- Wenn $w = r$ uniform ist:

  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei one-time-pad}] = \frac{1}{2}$
- Wenn $w = \mathcal{G}(s), s$ uniform ist:

  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei pseudo-one-time-pad}]$

Ein IND-EAV-sicheres Verfahren

- Wenn $w = r$ uniform ist:
  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei one-time-pad}] = \frac{1}{2}$
- Wenn $w = \mathcal{G}(s)$ ist:
  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei pseudo-one-time-pad}]$

Ein IND-EAV-sicheres Verfahren

- Wenn $w = r$ uniform ist:
  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei one-time-pad}] = \frac{1}{2}$
- Wenn $w = \mathcal{G}(s)$ ist:
  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei pseudo-one-time-pad}]$

Erfolg von $\mathcal{D}$

Ein IND-EAV-sicheres Verfahren

- Wenn $w = r$ uniform ist:

  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei one-time-pad}] = \frac{1}{2}$
- Wenn $w = \mathcal{G}(s)$ ist:

  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei pseudo-one-time-pad}]$

$$
\begin{aligned}
& \text{Erfolg von } \mathcal{D} \\
= \quad & |\, \mathbf{P}[\mathcal{D}(\mathcal{G}(s)) = 1] - \mathbf{P}[\mathcal{D}(r) = 1] \,|
\end{aligned}
$$

Ein IND-EAV-sicheres Verfahren

- Wenn $w = r$ uniform ist:

  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei one-time-pad}] = \frac{1}{2}$
- Wenn $w = \mathcal{G}(s)$ ist:

  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei pseudo-one-time-pad}]$

$$
\begin{aligned}
& \text{Erfolg von } \mathcal{D} \\
= \quad & |\, \mathbf{P}[\mathcal{D}(\mathcal{G}(s)) = 1] - \mathbf{P}[\mathcal{D}(r) = 1] \,| \\
= \quad & |\, \mathbf{P}[\mathcal{A} \text{ gewinnt bei pseudo-one-time-pad}] - \tfrac{1}{2} \,|
\end{aligned}
$$

Ein IND-EAV-sicheres Verfahren

- Wenn $w = r$ uniform ist:
  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei one-time-pad}] = \frac{1}{2}$
- Wenn $w = \mathcal{G}(s)$ ist:
  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei pseudo-one-time-pad}]$

$$
\begin{aligned}
& \text{Erfolg von } \mathcal{D} \\
= \quad & |\ \mathbf{P}[\mathcal{D}(\mathcal{G}(s)) = 1] - \mathbf{P}[\mathcal{D}(r) = 1]\ | \\
= \quad & |\ \mathbf{P}[\mathcal{A} \text{ gewinnt bei pseudo-one-time-pad}] - \tfrac{1}{2}\ | \\
= \quad & \text{Erfolg von } \mathcal{A}.
\end{aligned}
$$

Ein IND-EAV-sicheres Verfahren

- Wenn $w = r$ uniform ist:

  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei one-time-pad}] = \frac{1}{2}$
- Wenn $w = \mathcal{G}(s)$ ist:

  $\mathbf{P}[\text{Ausgabe} = 1] = \mathbf{P}[\mathcal{A} \text{ gewinnt bei pseudo-one-time-pad}]$

$$
\begin{aligned}
& \text{Erfolg von } \mathcal{D} \\
= \quad & |\, \mathbf{P}[\mathcal{D}(\mathcal{G}(s)) = 1] - \mathbf{P}[\mathcal{D}(r) = 1] \,| \\
= \quad & |\, \mathbf{P}[\mathcal{A} \text{ gewinnt bei pseudo-one-time-pad}] - \tfrac{1}{2} \,| \\
= \quad & \text{Erfolg von } \mathcal{A}.
\end{aligned}
$$

$\square$