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

§5 Der komplexitätstheoretische Ansatz

Shafi Goldwasser

Silvio Micali

Quellen. Wikipedia / Wikipedia, Benutzer Rama

Probabilistic encryption (1983)

Turing Award 2012

Das “polynomiell ist schnell-Paradigma”

Polynomzeit-Paradigma

Paradigma. Algorithmen durch eine Polynomfunktion beschränkter Laufzeit gelten als "qualitativ schnell".

Frage. Warum gerade durch Polynomfunktionen beschränkt?

Partielle (gute) Antworten.

- Wir wollen Unabhängigkeit vom Rechnermodell, solange die Modelle einigermaßen vernünftig sind.
- Von RAM (mit logarithmischem Maß) zu Turing-Maschine: Laufzeit(TM) $\leq C \cdot$ Laufzeit(RAM)^2.
- Abgeschlossenheit unter $(\ldots)^e$ ist sehr praktisch.

Polynomzeit-Paradigma

Weitere Antwort. Es wird "alles immer schneller", somit wird auch jede "polynomielle Schranke" überwunden werden.

Problem.

"Polynomiell" sagt wenig aus. Was sagt z.B. aus

$$\text{Laufzeit}(\mathcal{A}) \leq 2^{1000} \cdot n^{100} \text{ ?}$$

$n =$ Eingabelänge.

Polynomzeit-Paradigma

Notation. n = Eingabelänge.

Polynomielle Laufzeit = Laufzeit beschränkt durch eine Polynomfunktion in n

Es gibt ein $e \in \mathbf{N}$, ein $C > 0$:

$$\text{Laufzeit} \leq C \cdot n^e$$

Oder: Es gibt ein $e \in \mathbf{N}$:

$$\text{Laufzeit} \leq n^e$$

für große n.

Polynomzeit-Paradigma

Definition. Eine Funktion $f : \mathbf{N} \longrightarrow \mathbf{R}$ heißt vernachlässigbar, falls zu jedem $e \in \mathbf{N}$ ein $N > 0$ mit

$$|f(n)| \leq \frac{1}{n^e}$$

für alle $n \geq N$ gibt.

So eine Funktion ist (für $n \longrightarrow \infty$) gegen 0 konvergent.

Dies heißt:

Für alle $\epsilon > 0$ gibt es ein $N > 0$ mit

$$|f(n)| \leq \epsilon$$

für alle $n \geq N$.

(Oder: $(f(n))_{n \in \mathbf{N}}$ ist eine Nullfolge.)

Exkurs: Funktionen und Folgen

Formal. Eine Funktion $f : \mathbf{N} \longrightarrow \mathbf{R}$ ist identisch zur Folge $(f(n))_{n \in \mathbf{N}}$.

Z.B.

$$\mathbf{N} \longrightarrow \mathbf{R}\ ,\ n \mapsto \sin(n^2)$$

ist identisch zu

$$(\sin(n^2))_{n \in \mathbf{N}}$$

Beispiele.

- $\frac{1}{n^5}$ ist eine Nullfolge aber nicht vernachlässigbar
- $\frac{1}{5^n}$ ist vernachlässigbar

Polynomzeit-Paradigma

Wir schreiben

- Poly(n) für (in n) polynomiell beschränkte Funktionen (z.B. Laufzeit)
- negl(n) für (in n) vernachlässigbare Funktionen (z.B. Vorteil / Erfolg)

Außerdem:

PPT = Probabilistic polynomial time

$\rightarrow$ PPT-Algorithmus

Die üblichen komplexitätstheoretischen Sicherheitsniveaus

Verschlüsselungssystem

Wir wollen ein Verschlüsselungssystem wieder als ein Tripel von drei Algorithmen definieren:

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

mit $\mathcal{D}ec_k(\mathcal{E}nc_k(m)) = m$ (mit Wahrscheinlichkeit 1) für alle relevanten k, m.

Die Verfahren sollen wieder zustandslos sein.

Es soll nun einen Sicherheitsparameter geben.

Verschlüsselungssystem

Es gibt nun einen Sicherheitsparameter n.

Der Schlüsselerzeugungsalgorithmus erhält diesen Parameter.

Die Längen der Ein- und Ausgaben sollen $\geq n$ und polynomiell in n sein.

Alle Algorithmen sollen Polynomzeit-Algorithmen (PPT-Algorithmen) sein.

Mit anderen Worten: Die Laufzeiten sollen polynomiell in n sein.

Aber:

- Wir übergeben n an $\mathcal{G}en$.
- n hat selbst nur Länge $\sim \log_2(n)$.
- Die Eingabelänge soll aber etwa n sein.

Lösung. Wir übergeben n in unärer Darstellung:

Wir übergeben den String $\overbrace{1 \cdots 1}^{n \text{ mal}} =: 1^n$.

Verschlüsselungssystem

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

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

  Eingabe: 1^n mit $n =$ Sicherheitsparameter

  Ausgabe: Schlüssel $k = \mathcal{G}en(1^n)$ mit $|k| \geq n$
  (Standardgenerator : Ausgabe: k in $\{0,1\}^n$ uniform.)
- Einem Verschlüsselungsalgorithmus $\mathcal{E}nc$.

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

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

  Eingabe: Schlüssel k, Chiffriertext c

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

mit $\mathcal{D}ec_k(\mathcal{E}nc_k(m)) = m$ (mit Wahrscheinlichkeit 1) für alle relevanten k, m.

Das Ununterscheidbarkeits-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. Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.

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

Das Ununterscheidbarkeits-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. Er berechnet $c := \mathcal{E}nc_k(m_i)$ und schickt c an den Angreifer.
5. Der Angreifer entscheidet sich für $j \in \{1, 2\}$.

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

Wir definieren den Vorteil oder Erfolg von $\mathcal{A}$ als

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

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

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

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

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

Das Spiel zum adaptiven CCA (CCA2)

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 entscheidet sich für $j \in \{1, 2\}$.

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

Definition

Ein Verschlüsselungssystem mit öffentlichen Schlüsseln besteht aus drei 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: $\mathcal{D}ec_{sk}(m)$ deterministisch

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.

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

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

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.

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

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

Das Spiel zum adaptiven CCA (CCA2)

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

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

Semantische Sicherheit

Perfekte semantische Sicherheit (Wiederholung)

Es sei ein Verschlüsselungssystem ohne Sicherheitsparameter gegeben.

Idee. Alles, was man aus einem Chiffriertext einer Nachricht und weiteren Daten über die Nachricht selbst erfahren kann, kann man auch ohne den Chiffriertext berechnen.

Was bedeutet "alles"?

Das könnte z.B. ein bestimmtes Bit sein.

Oder die Summe von zwei bestimmten Bits.

Oder (1. Bit + 2. Bit) $\cdot$ 5. Bit

$\vdots$

Also: Für irgendeine Funktion $f : \mathcal{M} \longrightarrow \{0,1\}^*$: $f(m)$

"Daten über die Nachricht" entsprechend:

Für irgendeine Funktion $h : \mathcal{M} \longrightarrow \{0,1\}^*$: $h(m)$.

Perfekte semantische Sicherheit

Es sei ein Chiffierverfahren mit $\mathcal{M}, \mathcal{C}, \mathcal{K}; \mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec$ gegeben.

Definition. Das Verfahren ist perfekt semantisch sicher, wenn gilt:

Für alle Algorithmen $\mathcal{A}$ mit Eingabe $(c, \textbf{Zusatz}) \in \mathcal{C} \times \{0,1\}^*$ gibt es einen Algorithmus $\mathcal{A}'$ mit Eingabe $\textbf{Zusatz} \in \{0,1\}^*$ mit:

Für alle Funktionen $h : \mathcal{M} \longrightarrow \{0,1\}^*$ und $f : \mathcal{M} \longrightarrow \{0,1\}^*$ und alle zufälligen Nachrichten m:

$$\mathbf{P}[\mathcal{A}(\mathcal{E}nc_k(m), h(m)) = f(m)] = \mathbf{P}[\mathcal{A}'(h(m)) = f(m)]$$

Satz. Das Verfahren ist genau dann perfekt sicher, wenn es perfekt semantisch sicher ist.

(Das ist von mir und dient nur dazu, das Folgende klarer zu machen.)

Semantische Sicherheit

Gegeben sei ein Verschlüsselungssystem mit privaten Schlüsseln.

Änderungen:

- Sicherheitsparameter
- Polynomzeitparadigma
- Es wird nicht mehr jede Verteilung von Nachrichten betrachtet, sondern jedes effiziente Sampling der Nachrichten.

Definition. Das System ist semantisch sicher in Gegenwart eines Lauschers, wenn für jeden PPT-Algorithmus $\mathcal{A}$ ein PPT-Algorithmus $\mathcal{A}'$ existiert derart, dass für je zwei in Polynomzeit berechenbare Funktionen f, h und jeden PPT-Algorithmus $\mathcal{S}amp$

$$\mathbf{P}[\mathcal{A}(1^n, \mathcal{E}nc_k(m), h(m)) = f(m)] - \mathbf{P}[\mathcal{A}'(1^n, |m|, h(m)) = f(m)] \ ,$$

wobei $k := \mathcal{G}en(1^n)$ und $m := \mathcal{S}amp(1^n)$ ist, vernachlässigbar ist.

Semantische Sicherheit

Satz. Ein Verschlüsselungsverfahren mit privaten Schlüsseln und Standardgenerator ist genau dann IND-EAV sicher, wenn es semantisch sicher in Gegenwart eines Lauschers ist.

Änliche Definitionen und Resultate gibt es für die anderen Angriffsszenarien.

Das war's zu semantischer Sicherheit.

Grenzen komplexitätstheoretischer Aussagen

Grenzen komplexitätstheoretischer Aussagen

So gut wie alle komplexitätstheoretischen kryptographischen Aussagen beruhen auf Annahmen.

Zentral ist insbesondere: $P \neq NP$.

Frage (mehr als eine Übungsaufgabe). Gibt es IND-CPA-sichere Verfahren, wenn $P = NP$ ist?

Grenzen der komplexitätstheoretischen Aussagen

Komplexitätstheoretische kryptographische Aussagen sind "eigentlich immer" wenn-dann-Aussagen.

So wie:

- Wenn es keinen Polynomzeit-Algorithmus für Problem A gibt, ist Verfahren B IND-CPA-sicher.
- Wenn Verfahren B IND-CPA-sicher ist, ist Verfahren C IND-CCA2-sicher.
- Wenn die Hash-Funktion kollisionsresistent ist, ist das Verfahren sicher im Sinne von ...
- Wenn ... und ... und ..., ist das Protokoll für ... sicher im Sinne von ...

Theorie und Praxis

Ansätze für “konkrete Sicherheit”

Theorie und Praxis

Was sagt eine Aussage der Form

> *"Wenn es keinen PPT-Algorithmus für Problem XY gibt, dann gibt keinen PPT-Angreifer mit nicht-vernachlässigbarem Vorteil im CPA-Spiel Verfahren Z"*

über die Sicherheit konkreter kryptographischer Systeme aus?

Man hätte für konkrete Parameter gerne eine Aussage der Form:

> *Wir sind uns ziemlich sicher, dass man in der Praxis keinen CPA-Angriff durchführen kann, denn so ein Angriff würde auch bedeuten, dass man Problem XY mit Parametern ... lösen kann, und das scheint ausgeschlossen.*

Man muss hier natürlich immer beachten, was wirklich gesagt und was nicht gesagt wird, aber hier soll es um etwas anderes gehen:

Es gibt erstmal keinen Zusammenhang zwischen asymptotischen ("Polynomzeit") Aussagen) und konkreten Komplexitätsaussagen.

Definitionsansversuch für "konkretere Sicherheit"

Es sei ein System mit konkreten Parametern gegeben.

Z.B. AES oder ein System basierend auf Rechnen in einem konkreten endlichen Körper)

Es sei ein Angriffsszenario fixiert (z.B. IND-CPA).

Wir geben uns $t > 0$ und $\epsilon > 0$ vor und definieren:

Das Verfahren ist (t, ϵ)-sicher bezüglich ..., wenn es in Szenario ... keinen Angreifer mit Laufzeit $\leq t$ und Erfolg $\geq \epsilon$ gibt.

Man sollte hier ein Rechenmodell (mit Komplexitätsfunktion) fixieren, aber das ist nicht das Hauptproblem ...

“Es existiert”

“Es existiert ein Angriff” bedeutet nicht dasselbe wie

“Ich habe einen Angriff”.

Und das erste ist erstmal irrelevant.

Oftmals kann man sagen:

Es gibt einen Angriff. Der schaut in einer riesigen Tabelle nach.

Das könnte man ausschließen, indem man die Größe der Algorithmen beschränkt.

Aber das reicht nicht ...

Ein Extremfall

Eine Hashfunktion ist erstmal nur eine Funktion $\{0,1\}^* \longrightarrow \{0,1\}^\ell$.

Von einer kryptogrphischen Hashfunktionen H möchte man Kollisionsfreiheit:

Es können keine zwei verschiedene Strings x_1, x_2 mit $H(x_1) = H(x_2)$ gefunden werden.

Was soll das bedeuten?

Sicherlich:

Es sind keine zwei solchen Strings bekannt.

Sicherlicht nicht:

Es gibt keine zwei solchen Strings.

Vielleicht:

Nach heutigem Wissen wäre es viel zu aufwendig, zwei solche Strings zu berechnen.

Ein Extremfall

> *Nach heutigem Wissen wäre es viel zu aufwendig, zwei solche Strings zu berechnen.*

Das kann man schwer "mathematisch" formulieren.

Wir wollen eine Aussage der Form "Es gibt keinen Algorithmus, der ... effizient kann".

Das geht hier aber nicht.

Existenz eines passenden Algorithmus:

Da H nicht injektiv ist, gibt es $x_1 \neq x_2 \in \{0,1\}^*$ mit $H(x_1) = H(x_2)$.

Algorithmus: Gib ein solches Tupel (x_1, x_2) aus.

Es existiert ein Algorithmus, aber ...

Ein Extremfall

Standarddefinition für kryptographische Hash-Funktionen:

Wir haben zwei Algorithmen, $\mathcal{G}en$ und $\mathcal{H}$.

- Ein Schlüsselerzeugungsalgorithmus $\mathcal{G}en$.
  Eingabe: 1^n
  Ausgabe: Ein Schlüssel k
- Ein Hash-Algorithmus $\mathcal{H}$.
  Eingabe: Ein Schlüssel k, ein String x
  Ausgabe: Ein String von Länge $\ell = \ell(n)$ (deterministisch).

Ein Extremfall

Wir betrachten Algorithmen (Angreifer) $\mathcal{A}$ mit

Eingabe: Schlüssel k

Ausgabe: (x_1, x_2) mit $x_1 \neq x_2$ und $\mathcal{H}_k(x_1) = \mathcal{H}_k(x_2)$ oder sonstwas.

Für $k = \mathcal{G}en(1^n)$ definieren wir den Erfolg eines solchen Algorithmus als $\mathbf{P}$[gewünschte Ausgabe].

Bedingung: Jeder Angreifer $\mathcal{A}$ hat nur vernachlässigbaren Erfolg.

Das hat aber nichts mit der Praxis zu tun ....

Lösungsversuch

Wir betrachten wenn-dann-Aussagen.

Nehmen wir mal an, wir haben ein Signaturverfahren, dessen Sicherheit im folgendne Sinne auf Kollisionsresistenz einer Hash-Funktion beruht:

Wenn die Hash-Funktion kollisionsresistent ist, gibt es keinen PPT-Angreifer gegen das Signatur-Verfahren, der mit nicht-vernachlässigbaren Erfolg die Signatur einer Nachricht fälschen kann.

(Man braucht hier noch ein paar Definitionen; diese seien mal fixiert.)

Starke Variante mit konkreter Hash-Funktion und Parametern:

Wenn die Hashfunktion SHA3 kollisionsresistent ist, dann kann kein Angreifer gegen das Signatur-Verfahren eine Signatur fälschen.

Lösungsversuch

Wenn die Hashfunktion SHA3 kollisionsresistent ist, dann kann kein Angreifer gegen das Signatur-Verfahren eine Signatur fälschen.

Das ist "mathematisch" eine sinnlose Aussage. Wir betrachten stattdessen konkrete Varianten:

- Jeder Algorithmus zum Fälschen einer Signatur kann (recht leicht) in einen Algorithmus zum Finden einer Kollision transformiert werden. Die Laufzeiten sind vergleichbar.
- Wenn eine Signatur gefälscht werden kann, dann kann der Angreifer auch "ganz einfach" eine Kollision von SHA3 berechnen.
- Solange in SHA3 keine Kollision gefunden ist, kann auch keine Signatur gefälscht werden.

Sind diese Aussagen äquivalent? Was sollte "solange" bedeuten? Vielleicht "approximativ" ...

Zurück zu Theorie und Praxis

Definitionsversuche für konkrete Sicherheit

1. (konret und endlich) Wir betrachten ein System mit konkreten Parametern.
   Für $t > 0$ und $\epsilon > 0$ definieren wir:
   Das Verfahren ist (t, ϵ)-sicher bezüglich ..., wenn es in Szenario ... keinen Angreifer mit Laufzeit $\leq t$ und Erfolg $\geq \epsilon$ gibt.
2. (konkret mit Sicherheitsniveau) Man betrachtet ein System mit Sicherheitsparameter.
   Man definiert (t, ϵ)-sicher bezüglich ... für $t = t(n), \epsilon = \epsilon(n)$.
3. (konkret für großes Sicherheitsniveau) Wie ii) für "großes n".
   (D.h. es werden Aussagen der Form "Es gibt ein $N > 0$ derart, dass für alle $n \geq N$ gilt: ..." gemacht.)

Beachte: Wir haben eigentlich immer wenn-dann-Aussagen.

Ansatz iii) geht gut, Ansatz ii) vielleicht auch.

Ansatz i) vielleicht mit konstruktiven wenn-dann-Aussagen.

In der echten Praxis

Siehe zum Beispiel hier:

NIST: Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process

Für die Klausur ...

... sind die Sicherheits-Definitionen ganz zentral.

Fragen könnten sein:

- Wofür ist “IND-CPA-sicher” die Abkürzung?
- Wie lautet die formale Defiinition davon, dass ein kryptographisches System mit privatem Schlüssel IND-CPA-sicher ist?