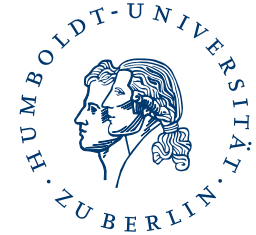
Kryptologie

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2020/21

Organisatorisches

Aktuelle Infos auf der VL-Webseite unter

- *https://hu.berlin/vlkrypto*

bzw.

- *https://www.informatik.hu-berlin.de/de/forschung/gebiete/algorithmenII/Lehre/ws20/krypto*

Ablauf

Skript, Folien und Aufgabenblätter

- Skript, Folien und Aufzeichnung werden jeweils nach der Vorlesung ins Netz (Webseite bzw. Moodle) gestellt
- Übungsblätter werden in der Regel dienstags veröffentlicht
- Die Besprechung der mündlichen Aufgaben erfolgt am Freitag der Folgewoche. Lösungen dazu können bis zum Tag davor in Moodle hochgeladen werden, Details siehe dort
- Die schriftlichen Aufgaben sind bis Dienstag zwei Wochen nach Ausgabe um 23:59 Uhr abzugeben
- Fragen zu Übung und Vorlesung können im Moodle-Forum auch asynchron gestellt und diskutiert werden

Übungen

Anmeldung

- über Agnes
- und bei Moodle (wegen Punktevergabe und Bildung von Abgabegruppen)
- Mails von Agnes und von Moodle werden standardmäßig an den HU-Account gesendet (bitte regelmäßig checken)

Ausgabe der Aufgabenblätter

- über Moodle und auf der VL-Webseite

Abgabe von Lösungen

- digital über Moodle

Bearbeitung der Übungsaufgaben

- in Gruppen von bis zu drei Teilnehmern
- Lösungen für die schriftlichen Aufgaben sollten als PDF abgegeben werden
- die Abgabe von Lösungsvorschlägen für die mündlichen Aufgaben ist freiwillig und geht nicht in die Punktewertung ein
- Lösungsvorschläge für die mündlichen Aufgaben können auch per Texteingabe gemacht werden
- besonders gut gelungene Lösungen werden mit Zustimmung der/des Abgebenden im Forum veröffentlicht

Übungsschein und Prüfung

Scheinkriterien

- Lösen von mindestens 50% der schriftlichen Aufgaben

Prüfungsform

- voraussichtlich mündlich
- Der Übungsschein ist ***nicht*** Prüfungsvoraussetzung

Gibt es zum organisatorischen Ablauf noch Fragen?

Lernziele

- Kryptografische Verfahren schaffen Vertrauen in ungeschützten Umgebungen
- Sie ermöglichen sichere Kommunikation über unsichere Kanäle und können verhindern, dass sich ein Kommunikationspartner unfair verhält
- In unsicheren Umgebungen wie dem Internet können sie die aus direkter Interaktion gewohnte Sicherheit herstellen
- Und auch die Interaktion in sicheren Umgebungen wird um Möglichkeiten erweitert, die ohne Kryptografie nicht denkbar wären
- Im Bachelormodul ***Einführung in die Kryptologie*** haben wir uns mit den mathematischen Grundlagen von kryptografischen Verfahren beschäftigen, wobei (symmetrische und asymmetrische) Verschlüsselungsverfahren im Vordergrund standen
- Im aktuellen Mastermodul ***Kryptologie*** werden wir dagegen kryptografische Verfahren und Protokolle für andere Schutzziele betrachten wie z.B. Hashverfahren und digitale Signaturen sowie Pseudozufallsgeneratoren

Kryptosysteme

- Kryptosysteme (Verschlüsselungsverfahren) dienen der Geheimhaltung von Nachrichten bzw. Daten
- Hierzu gibt es auch andere Methoden wie z.B.
  - Physikalische Maßnahmen: Tresor etc.
  - Organisatorische Maßnahmen: einsamer Waldspaziergang etc.
  - Steganografische Maßnahmen: unsichtbare Tinte etc.

Überblick weiterer Schutzziele

Andererseits können durch kryptografische Verfahren weitere Schutzziele realisiert werden wie z.B.

- Vertraulichkeit
  - Geheimhaltung
  - Anonymität (z.B. Mobiltelefon)
  - Unbeobachtbarkeit (von Transaktionen)
- Integrität
  - von Nachrichten und Daten
- Zurechenbarkeit
  - Authentikation
  - Unabstreitbarkeit
  - Identifizierung
- Verfügbarkeit
  - von Daten
  - von Rechenressourcen
  - von Informationsdienstleistungen

Kryptologie

In das Umfeld der Kryptologie fallen die folgenden Begriffe

- Kryptografie:
  Lehre von der Geheimhaltung von Informationen durch Verschlüsselung
  Im weiteren Sinne: Wissenschaft von der Übermittlung, Speicherung und Verarbeitung von Daten in einer von potentiellen Gegnern bedrohten Umgebung
- Kryptoanalysis:
  Erforschung der Methoden eines unbefugten Angriffs gegen ein Kryptoverfahren
  Zweck: Vereitelung der mit seinem Einsatz verfolgten Ziele
- Kryptoanalyse:
  Analyse eines Kryptoverfahrens zum Zweck der Bewertung seiner kryptografischen Stärken und Schwächen
- Kryptologie:
  Wissenschaft vom Entwurf, der Anwendung und der Analyse von kryptografischen Verfahren (umfasst Kryptografie und Kryptoanalyse)

Kryptografische Hashverfahren

- sind ein wirksames Werkzeug zur Sicherstellung der Integrität von Nachrichten oder generell von digitalisierten Daten
- Sie nehmen somit beim Schutz der Datenintegrität eine ähnlich herausragende Stellung ein wie sie Kryptosystemen bei der Wahrung der Vertraulichkeit zukommt
- Daneben finden kryptografische Hashfunktionen aber auch vielfach als Bausteine von komplexeren Systemen Verwendung
- Wie wir noch sehen werden, sind kryptografische Hashfunktionen etwa bei der Erstellung von digitalen Signaturen sehr nützlich
- Auf weitere Anwendungsmöglichkeiten werden wir später eingehen

Kryptografische Hashverfahren

- Vielen Anwendungen von kryptografischen Hashfunktionen h liegt die Idee zugrunde, dass sie zu einem vorgegebenen Text x eine zwar kompakte aber dennoch repräsentative Darstellung $h(x)$ liefern, die unter praktischen Gesichtspunkten als eine eindeutige Identifikationsnummer von x fungieren kann
- Die Berechnungsvorschrift für h muss somit „charakteristische Merkmale“ von x in den Hashwert $h(x)$ einfließen lassen
- Da der Fingerabdruck eines Menschen ganz ähnliche Eigenschaften besitzt (was ihn für Kriminalisten bekanntlich so wertvoll macht), wird der Hashwert $h(x)$ auch oft als ein ***digitaler Fingerabdruck*** von x bezeichnet
- Gebräuchlich sind auch die Bezeichnungen kryptografische Prüfsumme oder message digest (englische Bezeichnung für „Nachrichtenextrakt“)

Kryptografische Hashverfahren

Typische Schutzziele, die sich mittels Hashfunktionen realisieren lassen:

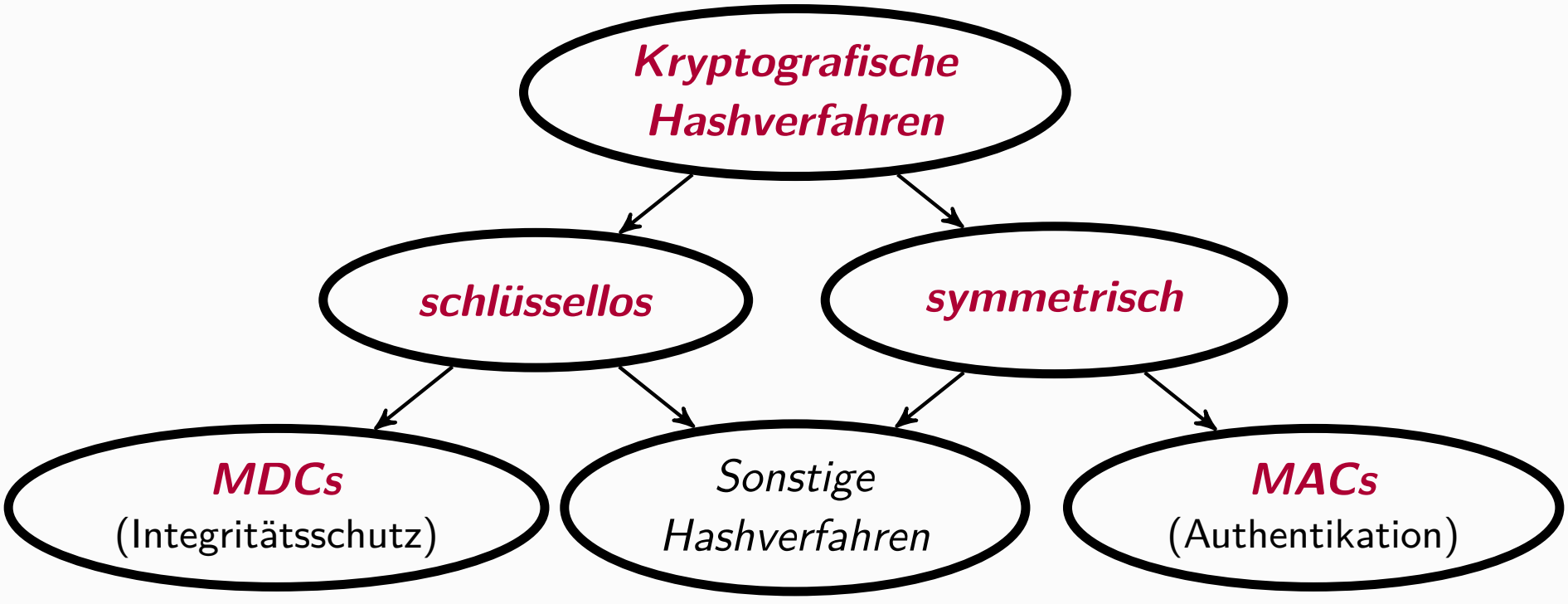
Nachrichtenauthentikation (message authentication)

- Wie lässt sich sicherstellen, dass eine Nachricht (oder eine Datei) während einer (räumlichen oder auch zeitlichen) Übertragung nicht verändert wurde?
- Wie lässt sich der Urheber (oder Absender) einer Nachricht zweifelsfrei feststellen?

Teilnehmerauthentikation (entity authentication, identification)

- Wie kann sich eine Person (oder ein Gerät) anderen gegenüber zweifelsfrei ausweisen?

Klassifikation von Hashverfahren

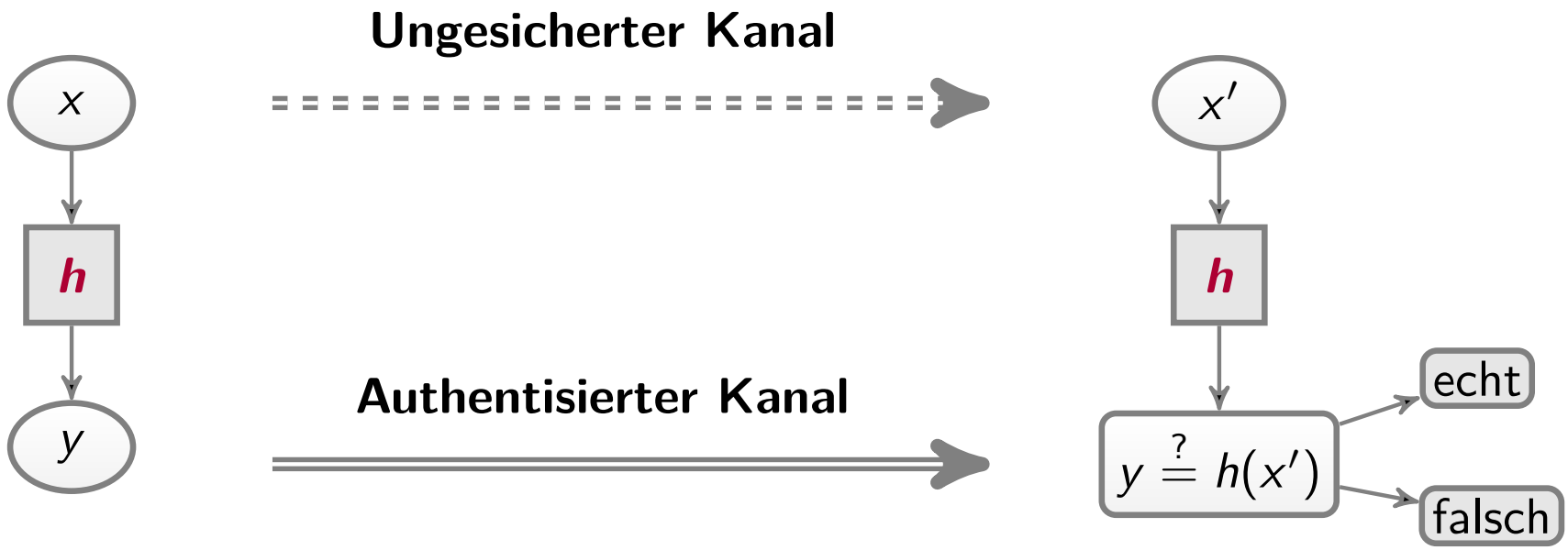


Kryptografische Hashverfahren lassen sich grob danach klassifizieren, ob der Hashwert lediglich in Abhängigkeit vom Eingabetext berechnet wird oder zusätzlich von einem symmetrischen Schlüssel abhängt

Schlüssellose Hashfunktionen

- Kryptografische Hashfunktionen, bei deren Berechnung keine Schlüssel benutzt werden, dienen vornehmlich der Erkennung von unbefugt vorgenommenen Manipulationen an Dateien oder Nachrichten
- Daher werden sie auch als ***MDC*** (***M****anipulation* ***D****etection* ***C****ode*) bezeichnet
- Zuweilen wird das Kürzel ***MDC*** auch als eine Abkürzung für ***M****odification* ***D****etection* ***C****ode* verwendet
- Seltener ist dagegen die Bezeichnung ***MIC*** (***m****essage* ***i****ntegrity* ***c****odes*)

Manipulation Detection Codes



Um die Integrität eines Datensatzes x sicherzustellen, der über einen ungesicherten Kanal gesendet (bzw. auf einem vor Manipulationen nicht sicheren Webserver abgelegt) wird, kann man wie folgt verfahren

- Der MDC-Hashwert $y = h(x)$ von x wird auf einem authentisierten Kanal übertragen
- Nach der Übertragung wird geprüft, ob der Datensatz noch den Hashwert y liefert

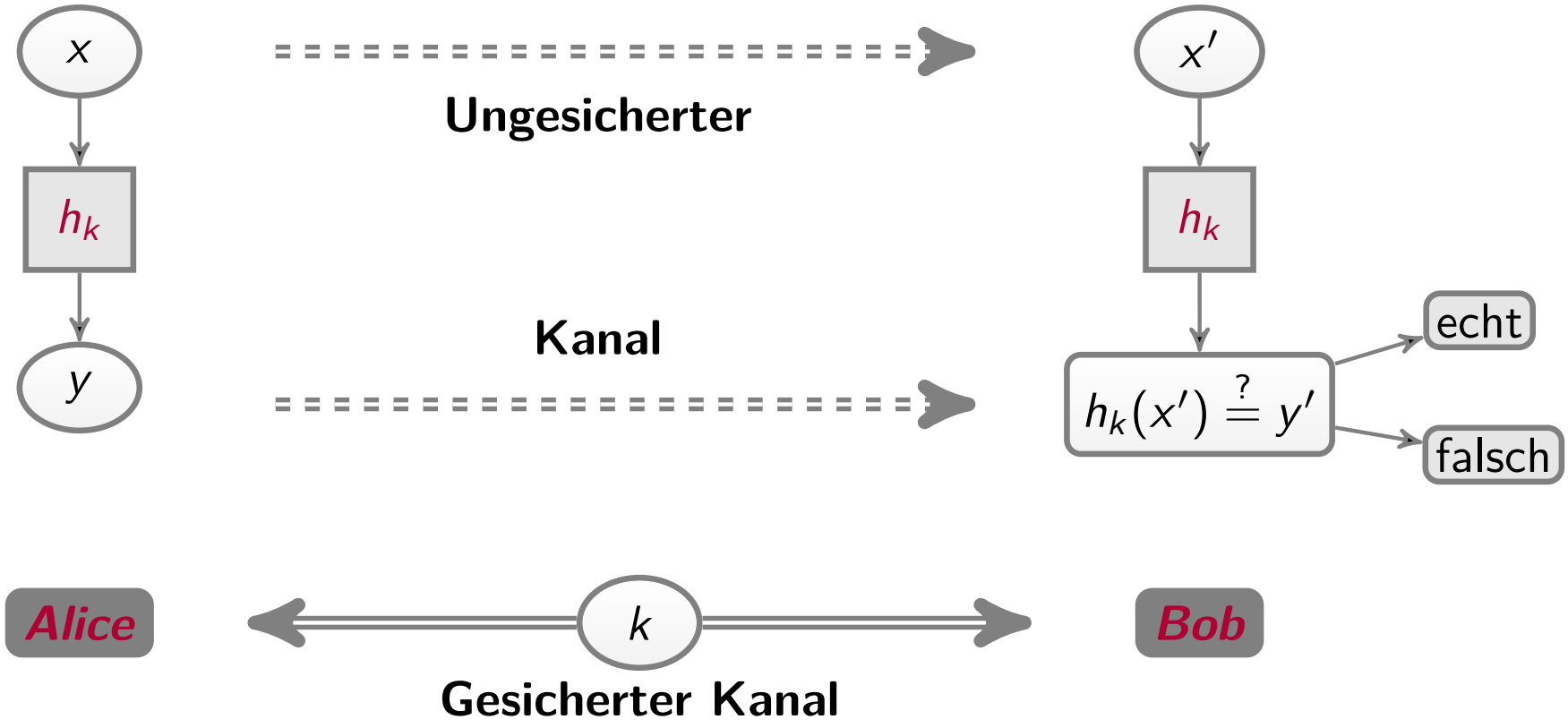
Hashverfahren mit symmetrischen Schlüsseln

- Kryptografische Hashverfahren mit symmetrischen Schlüsseln finden hauptsächlich bei der Authentifizierung von Nachrichten Verwendung
- Diese werden daher auch als ***MAC*** (***m****essage* ***a****uthentication* ***c****ode*) oder als ***Authentikationscode*** bezeichnet
- Daneben gibt es auch Hashverfahren mit asymmetrischen Schlüsseln
- Diese werden jedoch der Rubrik der Signaturverfahren zugeordnet, da mit ihnen ausschließlich digitale Signaturen gebildet werden

Hashverfahren mit symmetrischen Schlüsseln

- Die Abbildung auf der nächsten Folie zeigt, wie sich Nachrichten mit einem MAC authentisieren lassen
- Man beachte, dass nun auch der Hashwert über den unsicheren Kanal gesendet wird
- Möchte Alice eine Nachricht x an Bob übermitteln, so berechnet sie den zugehörigen MAC-Wert $y = h_k(x)$ und fügt diesen der Nachricht x hinzu
- Bob überprüft die Echtheit der empfangenen Nachricht (x', y'), indem er seinerseits den zu x' gehörigen Hashwert $h_k(x')$ berechnet und das Ergebnis mit y' vergleicht
- Der geheime Authentikationsschlüssel k muss hierbei genau wie bei einem symmetrischen Kryptosystem über einen gesicherten Kanal vereinbart werden

Verwendung eines MAC zur Nachrichtenauthentikation



- Hierbei ist k der symmetrische Authentikationsschlüssel und $y = h_k(x)$ der MAC-Wert für x unter k
- Indem Alice ihre Nachricht x um den Hashwert $y = h_k(x)$ ergänzt, hat Bob nicht nur die Möglichkeit, anhand von y die empfangene Nachricht x' auf Manipulationen, sondern auch ihre Herkunft zu überprüfen

Schlüssellose Hashfunktionen (MDCs)

- Wir betrachten nun verschiedene Sicherheitsanforderungen an MDCs h
- Dabei nehmen wir an, dass $h\colon X \to Y$ öffentlich bekannt ist
- Ein Paar $(x, y) \in X \times Y$ heißt ***gültig*** für h, falls $h(x) = y$ ist
- Ein Paar (x, x') mit $x \neq x'$ und $h(x) = h(x')$ heißt ***Kollisionspaar*** für h
- Die Anzahl $\|Y\|$ der Hashwerte bezeichnen wir mit m
- Ist auch der Textraum X endlich, $\|X\| = n$, so heißt h eine (n, m)-***Hashfunktion***
- In diesem Fall verlangen wir meist, dass $n \geq 2m$ ist, und wir nennen h dann eine ***Kompressionsfunktion*** (compression function)

Schlüssellose Hashfunktionen (MDCs)

- Da h öffentlich bekannt ist, ist es sehr einfach, für einen vorgegebenen Text x ein gültiges Paar (x, y) zu erzeugen
- Für bestimmte kryptografische Anwendungen ist es wichtig, dass dies bei vorgegebenem Hashwert y dagegen nicht möglich ist

Problem P1 (Bestimmung eines Urbilds)
Gegeben: Eine Hashfunktion $h\colon X \to Y$ und ein Hashwert $y \in Y$
Gesucht: Ein Text $x \in X$ mit $h(x) = y$

- Falls es einen immensen Aufwand erfordert, bei gegebenem Hashwert y einen Text x mit $h(x) = y$ zu finden, so heißt h ***Einweg-Hashfunktion*** *(one-way hash function bzw. preimage resistant hash function)*
- Diese Eigenschaft wird beispielsweise benötigt, wenn die Hashwerte der Benutzerpasswörter in einer öffentlich zugänglichen Datei abgespeichert werden, wie es bei manchen Unix-Systemen der Fall ist

Schlüssellose Hashfunktionen (MDCs)

- Für andere Anwendungen ist es dagegen wichtig, dass es für einen gebenen Text x praktisch unmöglich ist, einen weiteren Text $x' \neq x$ mit dem gleichen Hashwert $h(x') = h(x)$ zu finden

Problem P2 (Bestimmung eines zweiten Urbilds)
Gegeben: Eine Hashfunktion $h\colon X \to Y$ und ein Text $x \in X$
Gesucht: Ein Text $x' \in X \setminus \{x\}$ mit $h(x') = h(x)$

- Falls Problem P2 einen immensen Aufwand erfordert, heißt h ***schwach kollisionsresistent*** *(weakly collision resistant bzw. second preimage resistant)*
- Diese Eigenschaft wird beim Integritätsschutz durch einen MDC benötigt

Schlüssellose Hashfunktionen (MDCs)

- Für bestimmte Anwendungen ist es sogar nötig, dass sich überhaupt kein Kollisionspaar finden lässt
- Diese Eigenschaft ist bspw. beim Einsatz von MDCs bei der Erstellung von digitalen Signaturen erforderlich

Problem P3 (Bestimmung einer Kollision)
Gegeben: Eine Hashfunktion $h\colon X \to Y$
Gesucht: Zwei Texte $x \neq x' \in X$ mit $h(x') = h(x)$

- Falls Problem P3 einen immensen Aufwand erfordert, heißt h ***(stark) kollisionsresistent*** *(collision resistant)*

Schlüssellose Hashfunktionen (MDCs)

- Falls Problem P3 einen immensen Aufwand erfordert, heißt h ***(stark) kollisionsresistent*** *(collision resistant)*
- Obwohl die schwache Kollisionsresistenz eine gewisse Ähnlichkeit mit der Einweg-Eigenschaft aufweist, sind diese beiden Eigenschaften im allgemeinen unvergleichbar:
  - Eine schwach kollisionsresistente Funktion muss nicht notwendigerweise eine Einwegfunktion sein, da die Bestimmung eines Urbildes gerade für diejenigen Funktionswerte einfach sein kann, die nur ein einziges Urbild besitzen
  - Umgekehrt impliziert die Einweg-Eigenschaft auch nicht die schwache Kollisionsresistenz, da die Kenntnis eines Urbildes das Auffinden weiterer Urbilder sehr stark erleichtern kann

Vergleich von Sicherheitsanforderungen

- Wir zeigen nun, dass stark kollisionsresistente Hashfunktionen sowohl schwach kollisionsresistent als auch Einweghashfunktionen sind
- Hierzu reduzieren wir das Kollisionsroblem auf das Problem, ein zweites Urbild zu bestimmen

Satz

- Sei $h\colon X \to Y$ eine (n, m)-Hashfunktion
- Dann ist das Problem P3, ein Kollisionspaar für h zu bestimmen, auf das Problem P2, ein zweites Urbild zu bestimmen, reduzierbar
- Folglich sind stark kollisionsresistente Hashfunktionen auch schwach kollisionsresistent

Vergleich von Sicherheitsanforderungen

Satz

- Sei $h\colon X \to Y$ eine (n, m)-Hashfunktion
- Dann ist das Problem P3, ein Kollisionspaar für h zu bestimmen, auf das Problem P2, ein zweites Urbild zu bestimmen, reduzierbar
- Folglich sind stark kollisionsresistente Hashfunktionen auch schwach kollisionsresistent

Beweis.

- Sei A ein Las-Vegas Algorithmus, der für ein zufällig aus X gewähltes x mit Erfolgswahrscheinlichkeit ε ein zweites Urbild x' für h liefert und andernfalls ? ausgibt
- Dann ist klar, dass folgender Las-Vegas Algorithmus mit Wahrscheinlichkeit ε ein Kollisionspaar findet:

```
1 wähle zufällig x ∈ X
2 x' := A(x)
3 if x' ≠ ? then return (x, x') else return ?
```

□

Vergleich von Sicherheitsanforderungen

Als nächstes reduzieren wir das Kollisionsproblem auf das Urbildproblem

Satz

- Sei $h\colon X \to Y$ eine (n, m)-Hashfunktion mit $n \geq 2m$
- Dann ist das Problem P3, ein Kollisionspaar für h zu bestimmen, auf das Problem P1, ein Urbild zu bestimmen, reduzierbar

Beweis.

- Sei A ein Invertierungsalgorithmus für h, d.h. A berechnet für jeden Hashwert y in $W(h) = \{h(x) \mid x \in X\}$ ein Urbild x mit $h(x) = y$
- Betrachte folgenden Las-Vegas Algorithmus B:

```
1 wähle zufällig x ∈ X
2 y := h(x)
3 x' := A(y)
4 if x ≠ x' then return (x, x') else return ?
```

Vergleich von Sicherheitsanforderungen

Beweis.

- Sei A ein Invertierungsalgorithmus für h, d.h. A berechnet für jeden Hashwert y in $W(h) = \{h(x) \mid x \in X\}$ ein Urbild x mit $h(x) = y$
- Betrachte folgenden Las-Vegas Algorithmus B:

```
1 wähle zufällig x ∈ X
2 y := h(x)
3 x' := A(y)
4 if x ≠ x' then return (x, x') else return ?
```

- Sei $\mathcal{C} = \{h^{-1}(y) \mid y \in W(X)\}$
- Dann hat B eine Erfolgswahrscheinlichkeit von

$$\sum_{C \in \mathcal{C}} \frac{\|C\|}{\|X\|} \cdot \frac{\|C\| - 1}{\|C\|} = \frac{1}{n} \sum_{C \in \mathcal{C}} (\|C\| - 1) = (n - m)/n \geq \frac{1}{2}$$

□

Das Zufallsorakelmodell (ZOM)

- Das ZOM dient dazu, den Aufwand verschiedener Angriffe auf eine Hashfunktion $h\colon X \to Y$ nach oben abzuschätzen
- Sind X und Y vorgegeben, so können wir eine Hashfunktion $h\colon X \to Y$ dadurch „konstruieren“, dass wir für jedes $x \in X$ zufällig ein $y \in Y$ wählen und $h(x) = y$ setzen
- Äquivalent hierzu ist, für h eine zufällige Funktion aus der Klasse $F(X, Y)$ aller m^n Funktionen von X nach Y zu wählen
- Dieses Verfahren ist auf Grund des hohen Aufwands zwar nicht mehr praktikabel, wenn $n = \|X\|$ eine bestimmte Größe übersteigt
- Es liefert uns aber ein theoretisches Modell für eine Hashfunktion mit „idealen“ kryptografischen Eigenschaften
- Offensichtlich kann ein Angreifer nur dadurch Informationen über h erhalten, dass er für eine Reihe von Texten x_i die zugehörigen Hashwerte $h(x_i)$ berechnet (was der Befragung eines funktionalen Zufallsorakels entspricht)

Das Zufallsorakelmodell (ZOM)

Eine Zufallsfunktion h eignet sich deshalb gut als kryptografische Hashfunktion, weil der Hashwert $h(x)$ für einen Text x auch dann noch schwer vorhersagbar ist, wenn der Angreifer bereits die Hashwerte einer beliebigen Zahl von anderen Texten $x_i \neq x$ kennt

Proposition

- Sei $X_0 = \{x_1, \ldots, x_k\}$ eine beliebige Menge von k verschiedenen Texten $x_i \in X$ und seien $y_1, \ldots, y_k \in Y$
- Dann gilt für eine zufällig aus $F(X, Y)$ gewählte Funktion h und für jedes Paar $(x, y) \in (X - X_0) \times Y$,

$$\Pr[h(x) = y \,|\, h(x_i) = y_i \text{ für } i = 1, \ldots, k] = 1/m$$

Das Zufallsorakelmodell (ZOM)

- Um eine obere Komplexitätsschranke für das Urbildproblem P1 im ZOM zu erhalten, betrachten wir folgenden Algorithmus
- Hierbei gibt der Parameter q die Anzahl der Hashwertberechnungen (also die Anzahl der gestellten Orakelfragen an das Zufallsorakel h) an
- Die Laufzeit des Algorithmus ist also proportional zu q

Prozedur `FindPreimage`(h, y, q)

```
1 wähle eine beliebige Menge X_0 = {x_1, ..., x_q} ⊆ X
2 for each x_i ∈ X_0 do
3    if h(x_i) = y then return(x_i)
4 return ?
```

Das Zufallsorakelmodell (ZOM)

Satz

FindPreimage(h, y, q) gibt im ZOM mit Wahrscheinlichkeit $\varepsilon = 1 - (1 - 1/m)^q$ ein Urbild von y aus (unabhängig von der Wahl der Menge X_0)

Beweis.

- Sei $y \in Y$ fest und sei $X_0 = \{x_1, \ldots, x_q\}$
- Für $i = 1, \ldots, q$ bezeichne E_i das Ereignis "$h(x_i) = y$"
- Nach obiger Proposition sind diese Ereignisse stochastisch unabhängig und ihre Wahrscheinlichkeit ist

$$\Pr[E_i] = 1/m \text{ für } i = 1, \ldots, q$$

- Also folgt

$$\Pr[E_1 \cup \ldots \cup E_q] = 1 - \Pr[\overline{E}_1 \cap \ldots \cap \overline{E}_q] = 1 - (1 - 1/m)^q$$

□

Das Zufallsorakelmodell (ZOM)

Folgender Algorithmus liefert uns eine obere Schranke für die Komplexität des Problems P2, ein zweites Urbild für $h(x)$ zu bestimmen

Prozedur `FindSecondPreimage`(h, x, q)

```
1   y := h(x)
2   wähle eine beliebige Menge X_0 = {x_1, ..., x_{q-1}} ⊆ X − {x}
3   for each x_i ∈ X_0 do
4     if h(x_i) = y then return(x_i)
5   return ?
```

Satz

FindSecondPreimage(h, x, q) gibt im ZOM mit Wahrscheinlichkeit $\varepsilon = 1 - (1 - 1/m)^{q-1}$ ein zweites Urbild $x_0 \neq x$ von $y = h(x)$ aus.

Der Beweis ist analog zum Beweis des vorherigen Satzes

Der Geburtstagsangriff

- Ist q vergleichsweise klein, so ist bei beiden bisher betrachteten Angriffen $\varepsilon \approx q/m$
- Um also auf eine Erfolgswahrscheinlichkeit von $1/2$ zu kommen, ist $q \approx m/2$ zu wählen
- Geht es lediglich darum, *irgendein* Kollisionspaar (x, x') aufzuspüren, so bietet sich ein sogenannter ***Geburtstagsangriff*** an
- Dieser lässt sich deutlich zeiteffizienter realisieren
- Wie der Name schon andeutet, basiert dieser Angriff auf dem sog. Geburtstagsparadoxon, welches in seiner einfachsten Form folgendes besagt

Geburtstagsparadoxon

Bereits in einer Klasse mit 23 Kindern ist die Wahrscheinlichkeit größer $1/2$, dass mindestens zwei Kinder am gleichen Tag Geburtstag haben

Der Geburtstagsangriff

- Der nächste Satz besagt, dass bei q-maligem Ziehen (mit Zurücklegen) aus einer Urne mit m Kugeln mit einer Wahrscheinlichkeit von

  $$1 - (m-1)(m-2)\cdots(m-q+1)/m^{q-1}$$

  mindestens eine Kugel mehrmals gezogen wird
- Für $m = 365$ und $q = 23$ ergibt dies einen Wert von ungefähr 0,507
- Da die Häufigkeiten der Geburtstage in einer Klasse nicht gleichverteilt sind, ist die Wahrscheinlichkeit, dass 2 Kinder am gleichen Tag Geburtstag haben, sogar noch etwas höher
- Zur Kollisionsbestimmung verwenden wir folgenden Algorithmus

Prozedur `Collision`(h, q)

1 wähle eine beliebige Menge $X_0 = \{x_1, \ldots, x_q\} \subseteq X$
2 **for** each $x_i \in X_0$ **do** $y_i := h(x_i)$
3 **if** $\exists i \neq j : y_i = y_j$ **then return** (x_i, x_j) **else return** ?

Der Geburtstagsangriff

Prozedur `Collision`(h, q)

1 wähle eine beliebige Menge $X_0 = \{x_1, \ldots, x_q\} \subseteq X$
2 **for** each $x_i \in X_0$ **do** $y_i := h(x_i)$
3 **if** $\exists i \neq j : y_i = y_j$ **then return** (x_i, x_j) **else return** ?

- Bei einer naiven Implementierung würde zwar der Zeitaufwand für die Auswertung der if-Bedingung quadratisch von q abhängen
- Trägt man aber jeden Text x unter dem Suchwort $h(x)$ in eine Hashtabelle der Größe q ein, so wird der Zeitaufwand für jeden einzelnen Text x im wesentlichen durch die Berechnung von $h(x)$ bestimmt

Satz

$\textsc{Collision}(h, q)$ gibt im ZOM mit Erfolgswahrscheinlichkeit

$$\varepsilon = 1 - \frac{(m-1)(m-2)\cdots(m-q+1)}{m^{q-1}}$$

ein Kollisionspaar (x, x') für h aus

Der Geburtstagsangriff

Beweis.

- Sei $X_0 = \{x_1, \ldots, x_q\}$ und für $i = 1, \ldots, q$ bezeichne E_i das Ereignis

$$h(x_i) \notin \{h(x_1), \ldots, h(x_{i-1})\}$$

- Dann ist $E_1 \cap \ldots \cap E_q$ das Ereignis "$\textsc{Collision}(h, q)$ gibt ? aus"
- Für $i = 1, \ldots, q$ gilt nun

$$\Pr[E_i \,|\, E_1 \cap \ldots \cap E_{i-1}] = \frac{m - i + 1}{m}$$

- Dies führt auf die Erfolgswahrscheinlichkeit

$$\begin{aligned} \varepsilon &= 1 - \Pr[E_1 \cap \ldots \cap E_q] \\ &= 1 - \Pr[E_1]\Pr[E_2 \,|\, E_1] \cdots \Pr[E_q \,|\, E_1 \cap \ldots \cap E_{q-1}] \\ &= 1 - \left(\frac{m-1}{m}\right)\left(\frac{m-2}{m}\right)\cdots\left(\frac{m-q+1}{m}\right) \end{aligned}$$

□

Der Geburtstagsangriff

- Mit der Approximation $1 - x \approx e^{-x}$ erhalten wir folgende Abschätzung für ε:

$$\begin{aligned} \varepsilon &= 1 - \prod_{i=1}^{q-1} \left(1 - \frac{i}{m}\right) \\ &\approx 1 - \prod_{i=1}^{q-1} e^{\frac{-i}{m}} = 1 - e^{-\frac{1}{m}\sum_{i=1}^{q-1} i} = 1 - e^{-\frac{q(q-1)}{2m}} \\ &\approx 1 - e^{-\frac{q^2}{2m}} \approx q^2/2m \end{aligned}$$

- Für q erhalten wir daraus die Abschätzung

$$q \approx c_\varepsilon \sqrt{m}$$

  mit einer von ε abhängigen Konstante $c_\varepsilon = \sqrt{2\varepsilon}$
- Diese Abschätzung ist nur für ε-Werte nahe Null hinreichend genau

Der Geburtstagsangriff

- Aus der Abschätzung $\varepsilon \approx 1 - e^{-\frac{q^2}{2m}}$ für ε (siehe vorige Folie) erhalten wir insbesondere für größere Werte von ε eine bessere Abschätzung für q:

  $$q \approx c'_\varepsilon \sqrt{m}$$

  mit der Konstanten $c'_\varepsilon = \sqrt{2 \ln \frac{1}{1-\varepsilon}}$
- Für $\varepsilon = 1/2$ ergibt sich somit $q \approx \sqrt{(2 \ln 2)m} \approx 1{,}17\sqrt{m}$
- Besitzt also eine binäre Hashfunktion $h\colon \{0,1\}^n \to \{0,1\}^m$ die Hashwertlänge $m = 128$ Bit, so müssen im ZOM $q \approx 1{,}17 \cdot 2^{64}$ Texte gehasht werden, um mit einer Wahrscheinlichkeit von $1/2$ eine Kollision zu finden
- Um einem Geburtstagsangriff widerstehen zu können, sollte eine Hashfunktion mindestens eine Hashwertlänge von 128 oder besser 160 Bit haben

Iterierte Hashfunktionen

- Im Folgenden beschäftigen wir uns mit der Frage, wie sich aus einer kollisionsresistenten Kompressionsfunktion

  $$h\colon \{0,1\}^{m+t} \to \{0,1\}^{m}$$

  eine kollisionsresistente Hashfunktion

  $$\hat{h}\colon \{0,1\}^{*} \to \{0,1\}^{l}$$

  konstruieren lässt
- Hierzu betrachten wir folgende kanonische Konstruktionsmethode:

Iterierte Hashfunktionen

Preprocessing: Transformiere $x \in \{0,1\}^*$ mittels einer Funktion

$$y : \{0,1\}^* \to \bigcup_{r \geq 1} \{0,1\}^{rt}$$

zu einem String $y(x)$ mit der Eigenschaft $|y(x)| \equiv_t 0$

Processing: Sei $IV \in \{0,1\}^m$ ein öffentlich bekannter Initialisierungsvektor und sei $y(x) = y_1 \cdots y_r$ mit $|y_i| = t$ für $i = 1, \ldots, r$. Berechne eine Folge $z_0, \ldots, z_r$ von Strings $z_i \in \{0,1\}^m$ wie folgt:

$$z_i = \begin{cases} IV, & i = 0, \\ h(z_{i-1} y_i), & i = 1, \ldots, r \end{cases}$$

Optionale Ausgabetransformation: Berechne den Wert $\hat{h}(x) = g(z_r)$, wobei $g : \{0,1\}^m \to \{0,1\}^l$ eine öffentlich bekannte Funktion ist (meist wird für g die Identität verwendet)

Zur Berechnung von $\hat{h}(x)$ wird also die Funktion h genau r-mal aufgerufen

Iterierte Hashfunktionen

Wir formulieren nun eine für Preprocessing-Funktionen wünschenswerte Eigenschaft

Definition

- Eine Funktion $y\colon \{0,1\}^* \to \{0,1\}^*$ heißt ***suffixfrei***, falls es keine Strings $x \neq \tilde{x}$ und z in $\{0,1\}^*$ mit $y(\tilde{x}) = zy(x)$ gibt
- Mit anderen Worten: kein Funktionswert $y(x)$ ist Suffix eines Funktionswertes $y(\tilde{x})$ an einer Stelle $\tilde{x} \neq x$

Man beachte, dass jede suffixfreie Funktion insbesondere injektiv ist

Iterierte Hashfunktionen

Satz

Falls die Preprocessing-Funktion y suffixfrei und die Ausgabetransformation g injektiv ist, so ist mit h auch $\hat{h}$ kollisionsresistent

Beweis.

- Wir nehmen an, dass es gelingt, ein Kollisionspaar $(x, \tilde{x})$ für $\hat{h}$ zu finden (d.h. $\hat{h}(x) = \hat{h}(\tilde{x})$ und $x \neq \tilde{x}$)
- Seien $y(x) = y_1 \dots y_r$ und $y(\tilde{x}) = \tilde{y}_1 \dots \tilde{y}_s$ mit $r \leq s$
- Da y suffixfrei ist, muss ein Index $i \in \{1, \dots, r\}$ mit $y_i \neq \tilde{y}_{s-r+i}$ existieren
- Weiter seien z_i $(i = 0, \dots, r)$ und $\tilde{z}_j$ $(j = 0, \dots, s)$ die in der Processing-Phase berechneten Hashwerte
- Da g injektiv ist, muss mit $g(z_r) = \hat{h}(x) = \hat{h}(\tilde{x}) = g(\tilde{z}_s)$ auch $z_r = \tilde{z}_s$ gelten