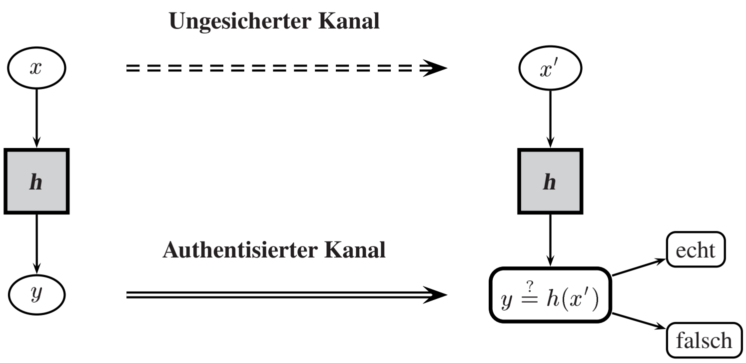


Abbildung 1 Einsatz eines MDC h zur Überprüfung der Integrität eines Datensatzes x.

5 Kryptographische Hashverfahren

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

Den überaus meisten Anwendungen von kryptographischen 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 daher gewissermaßen darauf abzielen, „charakteristische Merkmale" von x in den Hashwert $h(x)$ einfließen zu 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 **kryptographische Prüfsumme** oder *message digest* (englische Bezeichnung für „Nachrichtenextrakt").

Definition 153 (**Hashfamilie**)

Eine **Hashfamilie** $\mathcal{H}$ wird durch folgende Komponenten beschrieben:

- X, eine endliche oder unendliche Menge von Texten,
- Y, endliche Menge aller möglichen **Hashwerte**, $|Y| \leq |X|$,
- K, endlicher **Schlüsselraum** (*key space*),
- $H = \{h_k \mid k \in K\}$, endliche Menge von Hashfunktionen $h_k : X \to Y$.

Ein Paar $(x, y) \in X \times Y$ heißt **gültig** für h_k, falls $h_k(x) = y$ ist. Ein Paar (x, x') mit $h(x) = h(x')$ heißt **Kollisionspaar** für h. Ist K einelementig, so spricht man von einer **schlüssellosen** Hashfunktion.

Die Anzahl $|Y|$ der Hashwerte wird mit M bezeichnet. Ist auch der Textraum X endlich, $|X| = N$, so heißt $\mathcal{H}$ eine (N, M)-**Hashfamilie**. In diesem Fall verlangen wir, dass $N \geq 2M$ ist, und wir nennen h auch **Kompressionsfunktion** (*compression function*).

5.1 Sicherheit von Hashfunktionen

Sei $h : X \to Y$ eine Hashfunktion. Die einfachste Möglichkeit, ein gültiges Paar (x, y) für h zu erzeugen, ist, zuerst x zu wählen und dann $y = h(x)$ zu berechnen. In vielen kryptografischen Anwendungen ist es wichtig, dass dies die einzige effiziente Methode ist. Abhängig von der konkreten Anwendung, ist ein potentieller Gegner mit folgenden Problemen konfrontiert.

1. Problem (P1): Bestimmung eines Urbilds

Gegeben: Eine Hashfkt. $h : X \to Y$ und ein Hashwert $y \in Y$.

Gesucht: Ein Text $x \in X$ mit $h(x) = y$.

Falls es einen immensen Aufwand erfordert, für einen *vorgegebenen* 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*).

2. Problem (P2): Bestimmung eines zweiten Urbilds

Gegeben: Eine Hashfkt. $h : X \to Y$ und ein Text $x \in X$.

Gesucht: Ein Text $x' \in X \setminus \{x\}$ mit $h(x') = h(x)$.

Falls es einen immensen Aufwand erfordert, für einen *vorgegebenen* Text x einen weiteren Text $x' \neq x$ zu finden, der auf den gleichen Hashwert $h(x') = h(x)$ führt, so heißt h **schwach kollisionsresistent** (*weakly collision resistant* bzw. *second preimage resistant*).

Offensichtlich weist diese Art der Kollisionsresistenz eine gewisse Ähnlichkeit mit der Einweg-Eigenschaft auf. Trotz dieser Ähnlichkeit sind die beiden Eigenschaften im allgemeinen unvergleichbar. So muß eine schwach kollisionsresistente Funktion 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.

3. Problem (P3): Bestimmung einer Kollision

Gegeben: Eine Hashfkt. $h : X \to Y$.

Gesucht: Texte $x \neq x' \in X$ mit $h(x') = h(x)$.

Falls es einen immensen Aufwand erfordert, zwei verschiedene Texte $x \neq x'$ zu finden, die auf den gleichen Hashwert $h(x') = h(x)$ führen, so heißt h **(stark) kollisionsresistent** (*collision resistant*).

5.2 Das Zufallsorakelmodell (ZOM)

Das ZOM dient dazu, die Effizienz verschiedener Angriffe auf Hashfunktionen nach oben abzuschätzen. Liegen X und Y fest, so können wir eine Hashfunktion $h : X \to Y$ 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 N^M Funktionen von X nach Y zu wählen. Dieses Verfahren ist auf Grund des hohen Aufwands zwar nicht praktikabel. Es liefert uns aber ein theoretisches Modell für eine Hashfunktion mit "optimalen" kryptografischen Eigenschaften. Dabei entspricht die Auswertung von h an einer noch unbekannten Stelle x der Befragung eines (funktionalen) Zufallsorakels. Ausser derartigen Orakelbefragungen gibt es im ZOM keine Möglichkeit, Informationen über h zu erhalten.

Dass eine Zufallsfunktion h gute kryptografische Eigenschaften besitzt, liegt daran, dass alle Hashwerte gleichwahrscheinlich sind, auch dann, wenn bereits viele Werte $h(x_i)$ bekannt sind.

Lemma 154 *Sei h eine zufällig aus $F(X, Y)$ gewählte Funktion, sei $X_0 = \{x_1, \ldots, x_n\}$ eine beliebige Teilmenge von X und seien $y_i = h(x_i)$ die Werte, die h auf X_0 annimmt. Dann gilt für jedes $x \in X \setminus X_0$ und jedes $y \in Y$,*

$$Prob[h(x) = y \,|\, h(x_i) = y_i \text{ für } i = 1, \ldots, n] = 1/M.$$

Um eine obere Komplexitätsschranke für das Urbildproblem im ZOM zu erhalten, betrachten wir folgenden Algorithmus.

Algorithmus 155 FINDPREIMAGE(h, y, q)

wähle eine Menge $X_0 \subseteq X$ mit $\|X_0\| = q$
for each $x \in X_0$ **do**
  if $h(x) = y$ **then**
    output x
  end
end
output "?"

Satz 156 *Für jede Menge $X_0 \subseteq X$ mit $\|X_0\| = q$ gibt FindPreimage(h, y, q) mit Erfolgswahrscheinlichkeit $\varepsilon = 1 - (1 - 1/M)^q$ ein Urbild von y aus.*

Beweis: Sei $y \in Y$ fest und sei $X_0 = \{x_1, \dots, x_q\}$. Für $i = 1, \dots, q$ bezeichne E_i das Ereignis "$h(x_i) = y$". Dann ist klar, dass diese Ereignisse stochastisch unabhängig sind, und $Prob[E_i] = 1/M$ für $i = 1, \dots, q$ ist. Folglich ist

$$Prob[E_1 \cup \cdots \cup E_q] = 1 - Prob[\overline{E}_1 \cap \cdots \cap \overline{E}_q] = 1 - (1 - 1/M)^q.$$

■

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

Algorithmus 157 FINDSECONDPREIMAGE(h, x, q)

$y \leftarrow h(x)$
wähle eine Menge $X_0 \subseteq X \setminus \{x\}$ mit $\|X_0\| = q - 1$
for each $x_0 \in X_0$ **do**
  if $h(x_0) = y$ **then**
    output x_0
  end
end
output "?"

Vollkommen analog zum vorherigen Satz ergibt sich die folgende Erfolgswahrscheinlichkeit für diesen Algorithmus.

Satz 158 *Für jede Menge $X_0 \subseteq X$ mit $\|X_0\| = q - 1$ gibt FindSecondPreimage(h, x, q) mit Erfolgswahrscheinlichkeit $\varepsilon = 1 - (1 - 1/M)^{q-1}$ ein zweites Urbild $x_0 \neq x$ von $y = h(x)$ aus.*

Ist q klein im Vergleich zu M, 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 ist deutlich zeiteffizienter zu realisieren. Die auf den ersten Blick etwas verwirrende Namensgebung rührt daher, daß dieser Angriff auf dem sogenannten **Geburtstagsparadoxon** basiert, welches in seiner einfachsten Form folgendes besagt.

Geburtstagsparadoxon: Bereits in einer Schulklasse mit 23 Schulkindern haben mit einer Wahrscheinlichkeit größer $1/2$ mindestens zwei Kinder am gleichen Tag Geburtstag (dies erscheint zwar verblüffend, wird aber durch die Praxis mehr als bestätigt).

Tatsächlich zeigt der nächste Satz, dass bei q-maligem Ziehen (mit Zurücklegen) aus einer Urne mit M Kugeln mit einer Wahrscheinlichkeit von

$$\frac{(M-1)(M-2)\cdots(M-q+1)}{M^{q-1}}$$

keine Kugel zweimal gezogen wird. Für $M = 365$ und $q = 23$ ergibt dies einen Wert von ungefähr $0,493$.

Im nächsten Satz analysieren wir folgenden einfachen Algorithmus zur Kollisionsbestimmung.

Algorithmus 159 COLLISION(h, q)

wähle eine Menge $X_0 \subseteq X$ mit $\|X_0\| = q$
for each $x \in X_0$ **do**
  $y_x \leftarrow h(x)$
end
if $y_x = y_{x'}$ für zwei Texte $x \neq x'$ in X_0 **then**
  output (x, x')
else
  output "?"
end

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

Satz 160 *Für jede Menge $X_0 \subseteq X$ mit $\|X_0\| = q$ gibt Collision(h, q) mit Erfolgswahrscheinlichkeit*

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

ein Kollisionspaar (x, x') für h aus.

Beweis: Sei $X_0 = \{x_1, \ldots, x_q\}$. Für $i = 1, \ldots, q$ bezeichne E_i das Ereignis

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

Dann beschreibt $E_1 \cap \cdots \cap E_q$ das Ereignis "Collision(h, q) gibt ? aus" und für $i = 1, \ldots, q$ gilt

$$Prob[E_i \,|\, E_1 \cap \cdots \cap E_{i-1}] = \frac{M-i+1}{M}.$$

Dies führt auf die Erfolgswahrscheinlichkeit

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

■

Mit $1 - x \approx e^{-x}$ folgt

$$\varepsilon = 1 - \prod_{i=1}^{q-1}\left(1 - \frac{i}{M}\right) \approx 1 - \prod_{i=1}^{q-1} e^{\frac{-i}{M}} = 1 - e^{-\frac{1}{M}\sum_{i=1}^{q-1} i} = 1 - e^{-\frac{q(q-1)}{2M}}.$$

Dies lässt sich umformen zu

$$e^{\frac{q(q-1)}{2M}} \approx \frac{1}{1-\varepsilon},$$

beziehungsweise zu

$$\underbrace{q(q-1)}_{(q-\frac{1}{2})^2 - \frac{1}{4}} \approx 2M \ln \frac{1}{1-\varepsilon}.$$

Somit erhalten wir die Abschätzung

$$\begin{aligned} q &\approx \frac{1}{2} + \sqrt{\frac{1}{4} + 2M \ln \frac{1}{1-\varepsilon}} \\ &\approx c_\varepsilon \sqrt{M} \end{aligned}$$

mit $c_\varepsilon = \sqrt{2 \ln \frac{1}{1-\varepsilon}}$. Für $\varepsilon = 1/2$ ergibt sich also

$$q \approx 1{,}17\sqrt{M}.$$

Besitzt also eine binäre Hashfunktion $h : \{0,1\}^n \to \{0,1\}^m$ 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.

5.3 Vergleich von Sicherheitsanforderungen

In diesem Abschnitt zeigen wir, dass kollisionsfreie Hashfunktionen sowohl schwach kollisionsfrei als auch Einweghashfunktionen sein müssen.

Satz 161 *Sei $h : X \to Y$ 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.*

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. Dann ist klar, dass der Las-Vegas Algorithmus

wähle zufällig $x \in X$
$x' \leftarrow A(x)$
if $x' \neq$ "?" **then**
  output (x, x')
else
  output "?"
end

mit Wahrscheinlichkeit ε ein Kollisionspaar ausgibt. ∎

Als nächstes zeigen wir, wie sich das Kollisionsproblem auf das Urbildproblem reduzieren lässt.

Satz 162 *Sei $h : X \to Y$ 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:

wähle zufällig $x \in X$
$y \leftarrow h(x)$
$x' \leftarrow A(y)$
if $x \neq x'$ **then**
  output (x, x')
else
  output "?"
end

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

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

∎

5.4 Iterierte Hashfunktionen

In diesem Abschnitt beschäftigen wir uns mit der Frage, wie sich aus einer kollisionsresistenten Kompressionsfunktion

$$h:\ \{0,1\}^{m+t} \rightarrow \{0,1\}^m$$

eine kollisionsresistente Hashfunktion

$$\hat{h}:\ \{0,1\}^{\geq m+t} \rightarrow \{0,1\}^l$$

konstruieren lässt. Hierzu betrachten wir folgende kanonische Konstruktionsmethode.

Preprocessing: Transformiere $x \in \{0,1\}^{\geq m+t}$ mittels einer injektiven Funktion $y : \{0,1\}^{\geq m+t} \to \{0,1\}^*$ 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:

1 $z_0 \leftarrow IV$
2 $z_{i+1} \leftarrow h(z_i y_{i+1})$ für $i = 1, \ldots, r$.

Optionale Ausgabetransformation: Berechne den Hashwert

$$\hat{h}(x) = g(z_r),$$

wobei $g : \{0,1\}^m \to \{0,1\}^l$ eine öffentlich bekannte Funktion ist. (Meist wird für g die Identität verwendet.)

Unter Benutzung dieses Schemas haben Merkle und Damgard folgende Konstruktion vorgeschlagen. Als Initialisierungsvektors wird der Nullvektor $IV = 0^m$ benutzt, die optionale Ausgabetransformation entfällt, und für $y(x)$ wird im Fall $t \geq 2$ die folgende Funktion verwendet. (Den Fall $t = 1$ betrachten wir später.)

Sei $x = x_1 x_2 \ldots x_{k-1} x_k \in \{0,1\}^n$ mit $k = \lceil \frac{n}{t-1} \rceil$ und $|x_1| = |x_2| = \ldots = |x_{k-1}| = t-1$ sowie $|x_k| = t - 1 - d$, wobei $0 \leq d < t - 1$.

Dann ist $y : \{0,1\}^{\geq m+t} \to \{0,1\}^{\geq m+t}$ definiert durch $y(x) = y_1 \cdots y_{k+1}$, wobei

$$y_i = \begin{cases} 0x_1, & i = 1, \\ 1x_i, & 2 \leq i < k, \\ 1x_k0^d, & i = k, \\ 1\,\mathrm{bin}_{t-1}(d), & i = k+1 \end{cases}$$

und $\mathrm{bin}_{t-1}(d)$ die durch führende Nullen auf die Länge $t-1$ aufgefüllte Binärdarstellung von d ist. Um $\hat{h}(x)$ zu berechnen, muss also die Kompressionsfunktion h genau $(k+1)$-mal aufgerufen werden.

Satz 163 *Mit h ist auch $\hat{h}$ kollisionsresistent.*

Beweis: Angenommen, es gelingt, ein Kollisionspaar $x, \tilde{x}$ für $\hat{h}$ zu finden, wobei $x = x_1 x_2 \dots x_{k-1} x_k$ und $\tilde{x} = \tilde{x}_1 \tilde{x}_2 \dots \tilde{x}_{l-1} \tilde{x}_l$ ist. Wir zeigen, wie sich daraus in Polynomialzeit ein Kollisionspaar für h gewinnen lässt.

1. Fall: $|x| \not\equiv |\tilde{x}| \pmod{t-1}$.

$$\rightsquigarrow d \neq \tilde{d} \rightsquigarrow y_{k+1} \neq \tilde{y}_{l+1} \rightsquigarrow z_k y_{k+1} \neq \tilde{z}_l \tilde{y}_{l+1}$$

aber

$$h(z_k y_{k+1}) = z_{k+1} = \hat{h}(x) = \hat{h}(\tilde{x}) = \tilde{z}_{l+1} = h(\tilde{z}_l \tilde{y}_{l+1}),$$

d.h. $(z_k y_{k+1}, \tilde{z}_l \tilde{y}_{l+1})$ ist ein Kollisionspaar für h.

2. Fall: $|x| = |\tilde{x}|$. Dann gilt wegen $z_{k+1} = \hat{h}(x) = \hat{h}(\tilde{x}) = \tilde{z}_{k+1}$ für $i = k$

$$h(z_i y_{i+1}) = z_{i+1} = \tilde{z}_{i+1} = h(\tilde{z}_i \tilde{y}_{i+1}).$$

Ist nun $y_{i+1} \neq \tilde{y}_{i+1}$ oder $z_i \neq \tilde{z}_i$, so haben wir ein Kollisionspaar für h. Andernfalls können wir in obiger Gleichung i um eins erniedrigen. Finden wir auf diese Weise für $i = k, \dots, 0$ kein Kollisionspaar, so ist $y_{i+1} = \tilde{y}_{i+1}$ für $i = k, \dots, 0$, was auf Grund der Injektivität von y der Ungleichheit $x \neq \tilde{x}$ widerspricht.

3. Fall: $|x| \neq |\tilde{x}|$, $|x| \equiv |x^{'}| \pmod{t-1}$. Sei o.B.d.A. $k < l$. Dann gilt wegen $z_{k+1} = \hat{h}(x) = \hat{h}(\tilde{x}) = \tilde{z}_{l+1}$ für $i = 0$

$$h(z_{k-i} y_{k+1-i}) = z_{k+1-i} = \tilde{z}_{l+1-i} = h(\tilde{z}_{l-i} \tilde{y}_{l+1-i}).$$

Ist nun $y_{i+1-i} \neq \tilde{y}_{i+1-i}$ oder $z_{k-i} \neq \tilde{z}_{k-i}$, so haben wir ein Kollisionspaar für h. Andernfalls können wir in obiger Gleichung i um eins erhöhen. Wegen $y_1 \neq \tilde{y}_j$ für $j > 1$ finden wir auf diese Weise spätestens für $i = k$ ein Kollisionspaar.

∎

Nun kommen wir zum Fall $t = 1$. Sei die Funktion f definiert durch

$$f(x_1, \dots, x_n) = f(x_1) \dots f(x_2), \text{ wobei } f(0) = 0, f(1) = 01$$

und sei $y : \{0,1\}^{\geq m+2} \rightarrow \{0,1\}^*$ die Funktion $y(x) := 11f(x)$. Wie im Fall $t > 1$ hat auch diese Funktion die beiden folgenden Eigenschaften.

1. y ist injektiv und
2. es gibt keine Texte $x \neq x^{'}$ mit $y(x) = zy(x^{'})$ für ein $z \in \{0,1\}^*$ (d.h. kein y-Wert ist Suffix eines anderen y-Werts).

Schauen wir uns den Beweis des vorigen Satzes nochmals an, so stellen wir fest, dass nur diese beiden Eigenschaften benutzt wurden. Folglich gilt der Satz auch für diese Konstruktion. Da die Kompressionsfunktion h bei der Berechnung von $\hat{h}(x)$ für jedes Bit von $y(x)$ einmal aufgerufen wird, muss h höchstens $|y(x)| \leq 2(|x| + 1)$-mal berechnet werden.

Die MD4-Hashfunktion

Die MD4-Hashfunktion (***M**essage **D**igest*) wurde 1990 von Rivest vorgeschlagen. Eine verbesserte Version (MD5) wurde 1991 präsentiert. Die Bitlängen von MD4 und MD5 betragen $l = 128$ Bit. Der *Secure Hash Algorithm* (SHA-1) ist eine Weiterentwicklung des MD4 bzw. MD5 Algorithmus. Er gilt in den USA als Standard und ist Bestandteil des DSS (Digital Signature Standard). Die Bitlänge von SHA-1 beträgt $l = 160$ Bit. Bei einer Wortlänge von 32 Bit entspricht dies 5 Wörtern. MD4, MD5 und SHA-1 benutzen folgende Operationen auf Wörtern.

Operatoren auf $\{0,1\}^{32}$	
$X \wedge Y$	bitweises „Und“ von X und Y
$X \vee Y$	bitweises „Oder“ von X und Y
$X \oplus Y$	bitweises „exklusives Oder“ von X und Y
$\neg X$	bitweises Komplement von X
$X + Y$	Ganzzahl-Addition modulo 2^{32}
$X \hookleftarrow s$	zirkulärer Linksshift um s Stellen

Während die Ganzzahl-Addition bei MD4 und MD5 in *little endian* Architektur (d.h. ein aus 4 Bytes $a_0a_1a_2a_3$, $0 \le a_i \le 255$ zusammengesetztes Wort repräsentiert die Zahl $a_3 2^{24} + a_2 2^{16} + a_1 2^8 + a_0$) ausgeführt wird, verwendet SHA-1 eine *big endian* Architektur (d.h. $a_0a_1a_2a_3$, $0 \le a_i \le 255$ repräsentiert die Zahl $a_0 2^{24} + a_1 2^{16} + a_2 2^8 + a_3$).

Der MD4-Algorithmus benutzt die folgenden Konstanten $y_j, z_j, s_j, j = 0, \ldots, 47$

	y_j (in Hexadezimaldarstellung)
$j = 0, \ldots, 15$	0
$j = 16, \ldots, 31$	$5a827999$
$j = 32, \ldots, 47$	$6ed9eba1$

	z_j
$j = 0, \ldots, 15$	$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15$
$j = 16, \ldots, 31$	$0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15$
$j = 32, \ldots, 47$	$0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15$

	s_j
$j = 0, \ldots, 15$	$3, 7, 11, 19, 3, 7, 11, 19, 3, 7, 11, 19, 3, 7, 11, 19$
$j = 16, \ldots, 31$	$3, 5, 9, 13, 3, 5, 9, 13, 3, 5, 9, 13, 3, 5, 9, 13$
$j = 32, \ldots, 47$	$3, 9, 11, 15, 3, 9, 11, 15, 3, 9, 11, 15, 3, 9, 11, 15$

und folgende Funktionen $f_j, j = 0, \ldots, 47$

$$f_j(X, Y, Z) \quad := \quad \begin{cases} (X \wedge Y) \vee (\neg X \wedge Z), & j = \ \ 0, \ldots, 15, \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), & j = 16, \ldots, 31, \\ X \oplus Y \oplus Z, & j = 32, \ldots, 47. \end{cases}$$

Für MD4 konnten nach ca. 2^{20} Hashwertberechnungen Kollisionen aufgespürt werden. Deshalb gilt MD4 nicht mehr als kollisionsresistent.

Algorithmus 164 MD4(X)

Eingabe: $x \in \{0,1\}^*, |x| = n$
  $y \leftarrow x10^k\,\mathrm{bin}_{64}(n)$, $k \in \{0,1,\ldots,511\}$ mit $n + 1 + k + 64 \equiv 0 \pmod{512}$
  $(H_1, H_2, H_3, H_4) \leftarrow (67452301, efcdab89, 98badcfe, 10325476)$
  sei $y = M_1 \cdots M_r$, $r = (n + 1 + k + 64)/512$
  for $i \leftarrow 1$ **to** r **do**
    sei $M_i = X[0] \cdots X[15]$
    $(A, B, C, D) \leftarrow (H_1, H_2, H_3, H_4)$
    for $j \leftarrow 0$ **to** 47 **do**
      $(A, B, C, D) \leftarrow (D, (A + f_j(B, C, D) + X[z_j] + y_j) \hookleftarrow s_j, B, C)$
    end
    $(H_1, H_2, H_3, H_4) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D)$
  end
Ausgabe: $H_1 H_2 H_3 H_4$

Die MD5-Hashfunktion

In MD5 werden teilweise andere Konstanten als in MD4 verwendet. Zudem besitzt MD5 eine zusätzliche 4. Runde ($j = 48, \ldots, 63$), in der die Funktion $f_j(X, Y, Z) = Y \oplus (X \vee \neg Z)$ verwendet wird. Außerdem wurde die in Runde 2 von MD4 verwendete Funktion durch $f_j(X, Y, Z) := (X \wedge Z) \vee (Y \wedge \neg Z)$, $j = 16 \ldots 31$, ersetzt. Die y-Konstanten sind definiert als $y_j :=$ die ersten 32 Bit der Binärdarstellung von $\mathrm{abs}(\sin(j + 1))$, $0 \leq j \leq 63$, und für z_j und s_j werden folgende Konstanten benutzt.

	z_j
$j = 0, \ldots, 15$	$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15$
$j = 16, \ldots, 31$	$1, 6, 11, 0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12$
$j = 32, \ldots, 47$	$5, 8, 11, 14, 1, 4, 7, 10, 13, 0, 3, 6, 9, 12, 15, 2$
$j = 48, \ldots, 63$	$0, 7, 14, 5, 12, 3, 10, 1, 8, 15, 6, 13, 4, 11, 2, 9$

	s_j
$j = 0, \ldots, 15$	$7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22$
$j = 16, \ldots, 31$	$5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20$
$j = 32, \ldots, 47$	$4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23$
$j = 48, \ldots, 63$	$6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21$

Für MD5 konnten bisher keine Kollisionspaare gefunden werden (allerdings gelang dies für die Kompressionsfunktion von MD5).

Algorithmus 165 MD5(X)

Eingabe: $x \in \{0,1\}^*, |x| = n$
  $y \leftarrow x10^k\,\mathrm{bin}_{64}(n)$, $k \in \{0,1,\ldots,511\}$ mit $n + 1 + k + 64 \equiv 0 \pmod{512}$
  $(H_1, H_2, H_3, H_4) \leftarrow (67452301, efcdab89, 98badcfe, 10325476)$

sei $y = M_1 \cdots M_r$, $r = (n + 1 + k + 64)/512$
for $i \leftarrow 1$ **to** r **do**
  sei $M_i = X[0] \cdots X[15]$
  $(A, B, C, D) \leftarrow (H_1, H_2, H_3, H_4)$
  for $j \leftarrow 0$ **to** 63 **do**
    $(A, B, C, D) \leftarrow (D, (A + f_j(B, C, D) + X[z_j] + y_j) \hookleftarrow s_j, B, C)$
  end
  $(H_1, H_2, H_3, H_4) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D)$
end
Ausgabe: $H_1 H_2 H_3 H_4$

Die SHA-1-Hashfunktion

SHA-1 unterscheidet sich nur geringfügig von der SHA-Hashfunktion, in der eine Schwachstelle dazu führt, dass nach Berechnung von ca. 2^{61} Hashwerten ein Kollisionspaar gefunden werden kann (obwohl bei einem Geburtstagsangriff auf Grund der Hashwertlänge von 160 Bit ca. 2^{80} Berechnungen erforderlich sein müssten). Diese potentielle Schwäche wurde in SHA-1 entfernt.

Der SHA-1-Algorithmus benutzt die folgenden Konstanten K_j, $j = 0, \ldots, 79$

	K_j (in Hexadezimaldarstellung)
$j = 0, \ldots, 19$	$5a827999$
$j = 20, \ldots, 39$	$6ed9eba1$
$j = 40, \ldots, 59$	$8f1bbcdc$
$j = 60, \ldots, 79$	$ca62c1d6$

und folgende Funktionen f_j, $j = 0, \ldots, 79$

$$f_j(X, Y, Z) \quad := \quad \begin{cases} (X \wedge Y) \vee (\neg X \wedge Z), & j = 0, \ldots, 19, \\ X \oplus Y \oplus Z, & j = 20, \ldots, 39, \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), & j = 40, \ldots, 59, \\ X \oplus Y \oplus Z, & j = 60, \ldots, 79. \end{cases}$$

Algorithmus 166 SHA-1(X)

Eingabe: $x \in \{0, 1\}^*$, $|x| = n$
  $y \leftarrow x10^k \operatorname{bin}_{64}(n)$, $k \in \{0, 1, \ldots, 511\}$ mit $n + 1 + k + 64 \equiv 0 \pmod{512}$
  $(H_0, H_1, H_2, H_3, H_4) \leftarrow (67452301, efcdab89, 98badcfe, 10325476, c3d2e1f0)$
  sei $y = M_1 \cdots M_r$, $r = (n + 1 + k + 64)/512$
  for $i \leftarrow 1$ **to** r **do**
    sei $M_i = X[0] \cdots X[15]$
    for $t \leftarrow 16$ **to** 79 **do**
      $X[t] \leftarrow (X[t-3] \oplus X[t-8] \oplus X[t-14] \oplus X[t-16]) \hookleftarrow 1$
    end

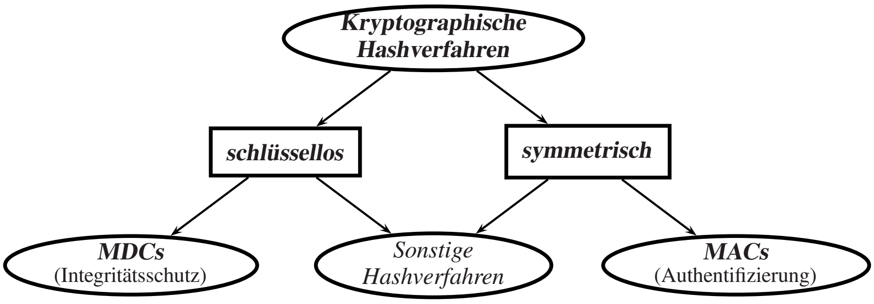


Abbildung 2 Eine grobe Einteilung von kryptographischen Hashverfahren.

    $(A, B, C, D, E) \leftarrow (H_0, H_1, H_2, H_3, H_4)$
    for $j \leftarrow 0$ **to** 79 **do**
      $temp \leftarrow (A \hookleftarrow 5) + f_j(B, C, D) + E + X[t] + K_t$
      $(A, B, C, D, E) \leftarrow (temp, A, B \hookleftarrow 30, C, D)$
    end
    $(H_0, H_1, H_2, H_3, H_4) \leftarrow (H_0 + A, H_1 + B, H_2 + C, H_3 + D, H_4 + E)$
  end
Ausgabe: $H_0 H_1 H_2 H_3 H_4$

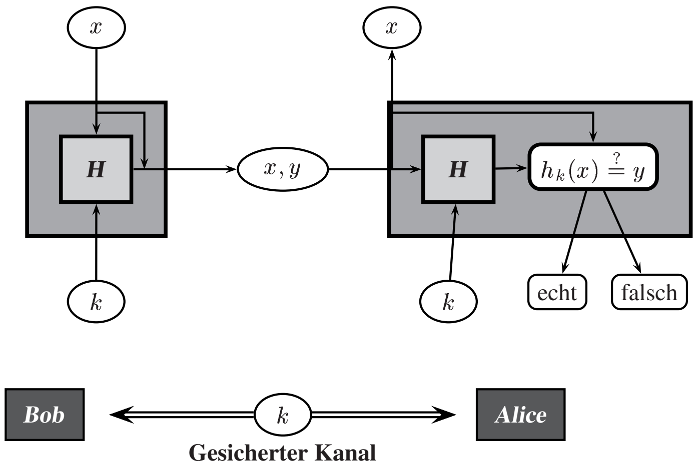
5.5 Klassifikation von Hashverfahren

Kryptographische Hashverfahren lassen sich grob danach klassifizieren, ob der Hashwert lediglich in Abhängigkeit vom Eingabetext berechnet wird oder zusätzlich von einem (symmetrischen) Schlüssel abhängt (vergleiche mit Abbildung 2).

Kryptographische 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*** bezeichnet (***M**anipulation* ***D**etection* ***C**ode* [englisch] = Code zur Erkennung von Manipulationen). 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*).

Kryptographische Hashverfahren mit symmetrischen Schlüsseln finden hauptsächlich bei der Authentifizierung von Nachrichten Verwendung. Diese werden daher auch als ***MAC*** (***m**essage* ***a**uthentication* ***c**ode* [englisch] = Code zur Nachrichtenauthentifizierung) oder als **Authentifizierungscode** bezeichnet.

Daneben gibt es auch Hashverfahren mit asymmetrischen Schlüsseln. Diese werden jedoch der Rubrik der Signaturverfahren zugeordnet, da mit ihnen ausschließlich digitale



k:	Symmetrischer Authentifizierungsschlüssel
$y = h_k(x)$:	*MAC*-Hashwert für x unter k

Abbildung 3 Ein MAC.

Unterschriften gebildet werden.

5.6 Nachrichten-Authentifizierungscodes (MACs,)

Wie sich Nachrichten mit einer Hashfamilie $\mathcal{H}$ authentisieren lassen, ist in Abbildung 3 dargestellt.

> Möchte Bob eine Nachricht x an Alice übermitteln, so berechnet er den zugehörigen *MAC*-Hashwert $y = h_k(x)$ und fügt diesen der Nachricht x hinzu. Alice überprüft die Echtheit der empfangenen Nachricht x', y', indem sie ihrerseits den zu x' gehörigen Hashwert $h_k(x')$ berechnet und das Ergebnis mit y' vergleicht. Der geheime Authentifizierungsschlüssel k muss hierbei genau wie bei einem symmetrischen Kryptosystem über einen gesicherten Kanal vereinbart werden.

Indem Bob seine Nachricht x um den Hashwert $y = h_k(x)$ ergänzt, gibt er Alice nicht nur die Möglichkeit, anhand von y die empfangene Nachricht auf Manipulationen zu überprüfen. Die Benutzung des geheimen Schlüssels k erlaubt zudem eine Überprüfung der Herkunft der Nachricht.

Sicherheitseigenschaften von *MACs*

Damit ein geheimer Schlüssel k für die Authentifizierung mehrerer Nachrichten benutzt werden kann, ohne dass dies einem potentiellen Gegner zur nichtautorisierten Berechnung von gültigen *MAC*-Werten verhilft, sollte die Hashfunktion H für jeden Schlüssel k die folgende Bedingung erfüllen.

Berechnungsresistenz: Auch wenn eine Reihe von unter dem Schlüssel k generierten Text-Hashwert-Paaren $(x_1, h_k(x_1)), \ldots, (x_n, h_k(x_n))$ bekannt ist, erfordert es einen immensen Aufwand, in Unkenntnis von k ein weiteres Paar (x, y) mit $y = h_k(x)$ zu finden.

Bei Verwendung einer berechnungsresistenten Hashfunktion ist es einem Gegner nicht möglich, an Alice eine Nachricht x zu schicken, die Alice als von Bob stammend anerkennt. Zu beachten ist allerdings, dass die Berechnungsresistenz nichts für den Fall aussagt, dass der Schlüssel k bekannt ist. So kann nicht davon ausgegangen werden, dass die Funktion $x \mapsto h_k(x)$ bei bekanntem k die Einweg-Eigenschaft besitzt oder schwach (beziehungsweise stark) kollisionsresistent ist. Es ist jedoch leicht zu sehen, dass diese Funktion bei geheimgehaltenem* k zumindest schwach kollisionsresistent ist.

Verwendung eines *MAC* zur Versiegelung von Software

Mithilfe einer berechnungsresistenten Hashfunktion kann der Integritätsschutz für mehrere Datensätze auf die Geheimhaltung eines Schlüssels k zurückgeführt werden.

> Um die Datensätze $x_1, \ldots, x_n$ gegen unbefugt vorgenommene Veränderungen zu schützen, legt man sie zusammen mit ihren Hashwerten $y_1 = h_k(x_1), \ldots, y_n = h_k(x_n)$ auf einem unsicheren Speichermedium ab und bewahrt den geheimen Schlüssel k an einem sicheren Ort auf. Bei einem späteren Zugriff auf einen Datensatz x_i lässt sich dessen Unversehrtheit durch einen Vergleich von y_i mit dem Ergebnis $h_k(x_i)$ einer erneuten *MAC*-Berechnung überprüfen.

Da auf diese Weise ein wirksamer Schutz der Datensätze gegen Viren und andere Manipulationen erreicht wird, spricht man von einer Versiegelung der gespeicherten Datensätze.

---

* Dies ist etwa der Fall, wenn k im Speicher eines ausforschungssicheren Chips abgelegt wird.

5.7 Angriffe gegen symmetrische Hashfunktionen

Ein Angriff gegen einen MAC hat die unbefugte Berechnung von Hashwerten zum Ziel. Das heißt, der Gegner versucht, Hashwerte $h_k(x)$ ohne Kenntnis des geheimen Schlüssels k zu berechnen. Entsprechend der Art des zur Verfügung stehenden Textmaterials lassen sich die Angriffe gegen einen MAC wie folgt klassifizieren.

Impersonation
Der Gegner kennt nur die benutzte Hashfamilie und versucht ein Paar (x, y) mit $h_k(x) = y$ zu generieren, wobei k der (dem Gegner unbekannte) Schlüssel ist.

Substitution
Der Gegner versucht in Kenntnis eines Paares $(x, h_k(x))$ ein Paar (x', y') mit $h_k(x') = y'$ zu generieren.

Angriff bei bekanntem Text (*known-text attack*)
Der Gegner kennt für eine Reihe von Texten $x_1, \ldots, x_n$ (die er nicht selbst wählen konnte) die zugehörigen *MAC*-Werte $h_k(x_1), \ldots, h_k(x_n)$ und versucht, ein Paar (x', y') mit $h_k(x') = y'$ und $x' \notin \{x_1, \ldots, x_n\}$ zu generieren.

Angriff bei frei wählbarem Text (*chosen-text attack*)
Der Gegner kann die Texte x_i selbst wählen.

Angriff bei adaptiv wählbarem Text (*adaptive chosen-text attack*)
Der Gegner kann die Wahl des Textes x_i von den zuvor erhaltenen *MAC*-Werten $h_k(x_j)$, $j < i$, abhängig machen.

Wechseln die Anwender nach jeder Hashwertberechnung den Schlüssel, so genügt es, dass $\mathcal{H}$ einem Substitutionsangriff widersteht.

5.8 Informationstheoretische Sicherheit von MACs

Modell: Schlüssel k und Nachrichten x werden unabhängig gemäß einer Wahrscheinlichkeitsverteilung $p(k, x) = p(k)p(x)$ generiert, welche dem Gegner (im Folgenden auch Oskar genannt) bekannt ist. Wir nehmen o.B.d.A. an, dass $p(x) > 0$ und $p(k) > 0$ für alle $x \in X$ und alle $k \in K$ gilt.

Erfolgswahrscheinlichkeit für Impersonation

p_{imp} : Wahrscheinlichkeit mit der sich ein Gegner bei optimaler Strategie als Bob ausgeben kann, ohne dass Alice dies bemerkt.

Für ein Paar (x, y) sei $p(x \mapsto y)$ die Wahrscheinlichkeit, dass ein zufällig gewählter Schlüssel den Text x auf den Hashwert y abbildet:

$$p(x \mapsto y) = \sum_{k \in K(x,y)} p(k).$$

wobei $K(x, y) = \{k \in K \mid h_k(x) = y\}$ alle Schlüssel enthält, die x auf y abbilden. D.h. $p(x \mapsto y)$ ist die Wahrscheinlichkeit, dass Alice das (vom Gegner gewählte) Paar (x, y) als echt akzeptiert. Dann gilt

$$p_{imp} = \max\{p(x \mapsto y) \mid x \in X, y \in Y\}.$$

Beispiel 167 *Sei $X = Y = \{0, 1, 2\} = \mathbb{Z}_3$ und sei $K = \mathbb{Z}_3 \times \mathbb{Z}_3$. Für $k = (a, b) \in K$ sei h_k die Hashfunktion*

$$h_k(x) = ax + b \bmod 3, x \in X.$$

Die zu $\mathcal{H} = (X, Y, K, H)$ gehörige Authentikationsmatrix A erhalten wir, indem wir die Zeilen mit den Schlüsseln $k \in K$ und die Spalten mit den Texten $x \in X$ indizieren und in Zeile k und Spalte x den Hashwert $h_k(x)$ eintragen.

	0	1	2
$(0,0)$	0	0	0
$(0,1)$	1	1	1
$(0,2)$	2	2	2
$(1,0)$	0	1	2
$(1,1)$	1	2	0
$(1,2)$	2	0	1
$(2,0)$	0	2	1
$(2,1)$	1	0	2
$(2,2)$	2	1	0

Angenommen, jeder Schlüssel (a, b) hat die gleiche Wk $p(a, b) = 1/9$. Versucht der Gegner dann eine Impersonation mit dem Paar (x, y), so akzeptieren genau 3 der 9 möglichen Schlüssel dieses Paar. Dies liegt daran, dass in jeder Spalte jeder Hashwert genau dreimal vorkommt. Also gilt $p(x \mapsto y) = 3/9 = 1/3$ für alle Paare $(x, y) \in X \times Y$, was für p_{imp} ebenfalls den Wert $p_{imp} = 1/3$ ergibt. ◁

Satz 168 *Für jede (N, M)-Hashfamilie $\mathcal{H}$ gilt $p_{imp} \geq \frac{1}{M}$.*

Beweis: Sei $x \in X$ beliebig. Dann gilt

$$\sum_{y \in Y} p(x \mapsto y) = \sum_{y \in Y} \sum_{k \in K(x,y)} p(k) = \sum_{k \in K} p(k) = 1.$$

Somit existiert für jedes $x \in X$ ein $y \in Y$ mit $p(x \mapsto y) \geq \frac{1}{M}$ und dies impliziert

$$p_{imp} = \max_{x,y} p(x \mapsto y) \geq \frac{1}{M}.$$

■

Bemerkung 169 *Wie der Beweis zeigt, gilt* $p_{imp} = \frac{1}{M}$ *genau dann, wenn für alle Paare* $(x, y) \in X \times Y$ *gilt,*

$$\sum_{k \in K(x,y)} p(k) = \frac{1}{M}.$$

D.h. bei Gleichverteilung der Schlüssel muss in jeder Spalte jeder Hashwert gleich oft vorkommen.

Erfolgswahrscheinlichkeit für Substitution

p_{sub} : Wahrscheinlichkeit mit der ein Gegner bei optimaler Strategie eine von Bob gesendete Nachricht (x, y) durch eine andere Nachricht (x', y') ersetzen kann, ohne dass Alice dies bemerkt.

Angenommen, Bob sendet die Nachricht (x, y) und der Gegner ersetzt diese durch (x', y'). Dann ist die Erfolgswahrscheinlichkeit des Gegners gleich der bedingten Wk

$$p(x' \mapsto y' \,|\, x \mapsto y) = \frac{p(x \mapsto y, x' \mapsto y')}{p(x \mapsto y)} = \frac{\sum_{k \in K(x,y,x',y')} p(k)}{\sum_{k \in K(x,y)} p(k)},$$

dass ein zufällig gewählter Schlüssel k den Text x' auf y' abbildet, wenn bereits bekannt ist, dass er x auf y abbildet. Falls Bob also das Paar (x, y) sendet, so kann der Gegner bestenfalls die Erfolgswahrscheinlichkeit

$$p_{sub}(x, y) = \max\{p(x' \mapsto y' \,|\, x \mapsto y) \mid x' \in X - \{x\}, y' \in Y\}$$

erzielen. Da Bob auf die Wahl von (x, y) keinen Einfluss hat, berechnet sich p_{sub} als der erwartete Wert von $p_{sub}(x, y)$, wobei das Paar (x, y) von Bob mit Wk

$$p(x, y) = p(x)p(y|x) = p(x) \sum_{k \in K(x,y)} p(k) = p(x)p(x \mapsto y)$$

gesendet wird. Somit ergibt sich p_{sub} zu

$$p_{sub} = \sum_{x \in X, y \in Y} p(x, y) p_{sub}(x, y) = \sum_{x \in X} p(x) p_{sub}(x),$$

wobei

$$p_{sub}(x) = \sum_{y \in Y} \max\{p(x \mapsto y, x' \mapsto y') \mid x' \in X - \{x\}, y' \in Y\}$$

die (erwartete) Erfolgswahrscheinlichkeit des Gegners bei optimaler Strategie ist, falls Bob den Text x wählt.

Satz 170 *Für jede* (N, M)*-Hashfamilie* $\mathcal{H}$ *gilt* $p_{sub} \geq \frac{1}{M}$.

Beweis: Sei $(x,y) \in X \times Y$ ein Paar mit $p(x,y) > 0$. Dann gilt für beliebige $x' \in X - \{x\}$,

$$\sum_{y' \in Y} p(x' \mapsto y' \,|\, x \mapsto y) = \frac{\sum_{y' \in Y} \sum_{k \in K(x',y';x,y)} p(k)}{\sum_{k \in K(x,y)} p(k)} = 1.$$

Somit existiert ein $y' \in Y$ mit $p(x' \mapsto y' \mid x \mapsto y) \geq \frac{1}{M}$ und dies impliziert für alle (x,y) mit $p(x,y) > 0$,

$$p_{sub}(x,y) \quad = \quad \max\{p(x' \mapsto y' \,|\, x \mapsto y) \mid x' \in X - \{x\}, y' \in Y\} \;\geq\; \frac{1}{M}, \quad (8)$$

was wiederum

$$p_{sub} = \sum_{x \in X, y \in Y} p(x,y) p_{sub}(x,y) \geq \frac{1}{M} \sum_{x \in X, y \in Y} p(x,y) = \frac{1}{M}$$

impliziert. ■

Lemma 171 *Sei $\mathcal{H}$ eine (N,M)-Hashfamilie mit $p_{sub} = \frac{1}{M}$. Dann gilt $p(x,y) > 0$ für alle Paare $(x,y) \in X \times Y$ und $p(x' \mapsto y' \mid x \mapsto y) = 1/M$ für alle Doppelpaare (x,y,x',y') mit $x \neq x'$.*

Beweis: Falls ein Paar $(w,z) \in X \times Y$ existiert mit $p(w,z) = 0$, so ist auch $p(w \mapsto z \,|\, u \mapsto v) = 0$, wobei $(u,v) \in X \times Y$ ein beliebiges Paar mit $p(u,v) > 0$ ist. Wegen

$$1 = \sum_{z' \in Y} p(w \mapsto z' \,|\, u \mapsto v) = \sum_{z' \in Y - \{z\}} p(w \mapsto z' \,|\, u \mapsto v)$$

impliziert dies die Existenz eines Hashwertes z' mit $p(w \mapsto z' \,|\, u \mapsto v) \geq 1/(M-1) > 1/M$. Dann ist aber auch $p_{sub}(u,v) = \max\{p(u' \mapsto v' \,|\, u \mapsto v) \mid u' \in X - \{u\}, v' \in Y\} > 1/M$, was wegen (8) $p_{sub} = \sum_{x \in X, y \in Y} p(x,y) p_{sub}(x,y) > 1/M$ impliziert.

Ist nun $p(x' \mapsto y' \mid x \mapsto y) \neq 1/M$ für ein Doppelpaar (x,y,x',y') mit $x \neq x'$, so muss wegen $\sum_{z' \in Y} p(x' \mapsto z' \mid x \mapsto y) = 1$ auch ein Doppelpaar (x,y,x',y') mit $p(x' \mapsto y' \mid x \mapsto y) > 1/M$ und $x \neq x'$ existieren, was genau wie oben zu einem Widerspruch führt. ■

Satz 172 *Eine (N,M)-Hashfamilie $\mathcal{H}$ erfüllt $p_{sub} = \frac{1}{M}$ genau dann, wenn*

$$p(x \mapsto y, x' \mapsto y') = 1/M^2$$

für alle Doppelpaare (x,y,x',y') mit $x \neq x'$ gilt.

Beweis: Sei $\mathcal{H}$ eine (N,M)-Hashfamilie mit $p_{sub} = \frac{1}{M}$. Nach obigem Lemma impliziert dies, dass $p(x,y) > 0$ für alle Paare $(x,y) \in X \times Y$ und $p(x' \mapsto y' \,|\, x \mapsto y) = 1/M$ für alle Doppelpaare (x,y,x',y') mit $x \neq x'$ gilt. Dies impliziert nun

$$p(x' \mapsto y') = \sum_{y} p(x \mapsto y) p(x' \mapsto y' \,|\, x \mapsto y) = 1/M$$

und daher

$$p(x \mapsto y, x' \mapsto y') = p(x' \mapsto y')p(x \mapsto y \,|\, x' \mapsto y') = 1/M^2.$$

Umgekehrt rechnet man leicht nach, dass $\mathcal{H}$ tatsächlich die Bedingung $p_{sub} = \frac{1}{M}$ erfüllt, wenn $p(x \mapsto y, x' \mapsto y') = 1/M^2$ für alle Doppelpaare (x, y, x', y') mit $x \neq x'$ gilt. ■

Bemerkung 173 *Nach obigem Satz gilt $p_{sub} = \frac{1}{M}$ genau dann, wenn für alle Doppelpaare (x, y, x', y') mit $x \neq x'$ gilt,*

$$p(x \mapsto y, x' \mapsto y') = \sum_{k \in K(x,y,x',y')} p(k) = \frac{1}{M^2}.$$

D.h. bei Gleichverteilung der Schlüssel gilt $p_{sub} = \frac{1}{M}$ genau dann, wenn in je zwei Spalten der Authentikationsmatrix jedes Hashwertpaar gleich oft vorkommt.

Ab jetzt setzen wir voraus, dass der Schlüssel unter Gleichverteilung gewählt wird, d.h. es gilt $p(k) = \frac{1}{\|K\|}$ für alle $k \in K$.

Definition 174 (stark universale Hashfamilien)

Eine (N, M)-Hashfamilie $\mathcal{H} = (X, Y, K, H)$ heißt stark universal, falls für alle $x, x' \in X$ mit $x \neq x'$ und alle $y, y' \in Y$ gilt:

$$\|K(x, y, x', y')\| = \frac{\|K\|}{M^2}.$$

Bemerkung 175 *Bei der Konstruktion von stark universalen Hashfamilien spielt der Parameter $\lambda = \frac{\|K\|}{M^2} = 1$ eine wichtige Rolle. Da λ notwendigerweise positiv und ganzzahlig ist, muss insbesondere $\|K\| \geq M^2$ gelten.*

Beispiel 176 *Betrachten wir die $(4, 3)$-Hashfamilie $\mathcal{H} = (X, Y, K, H)$ mit $X = \{0, 1, 2, 3\}$, $Y = \{0, 1, 2\}$, $K = \{0, 1, \ldots, 8\}$ und folgender Authentikationsmatrix,*

	0	1	2	3
0	0	0	0	0
1	1	1	1	0
2	2	2	2	0
3	0	1	2	1
4	1	2	0	1
5	2	0	1	1
6	0	2	1	2
7	1	0	2	2
8	2	1	0	2

so sehen wir, dass in je zwei Spalten jedes Hashwertpaar genau einmal vorkommt (also $\lambda = 1$ ist). ◁

Auf Grund von Bemerkung 173 ist klar dass eine (N, M)-Hashfamilie $\mathcal{H}$ bei gleichverteilten Schlüsseln genau dann die Bedingung $p_{sub} = \frac{1}{M}$ erfüllt, wenn $\mathcal{H}$ stark universal ist. Auf Grund von Bemerkung 169 nimmt in diesem Fall auch p_{imp} den optimalen Wert $\frac{1}{M}$ an.

Satz 177 *Sei p prim und für* $a, b \in \mathbb{Z}_p$ *sei*

$$h_{(a,b)}(x) = ax + b \bmod p.$$

Dann ist $\mathcal{H} = (X, Y, K, H)$ *mit* $X = Y = \mathbb{Z}_p, K = \mathbb{Z}_p \times \mathbb{Z}_p$ *und* $H = \{h_k | k \in K\}$ *eine stark universale* (p, p)*-Hashfamilie (wegen* $N = M = p$ *handelt es sich streng genommen nicht um "Hash"funktionen).*

Beweis: Wir müssen zeigen, dass die Größe von $K(x, y, x', y')$ für alle Doppelpaare (x, y, x', y') mit $x \neq x'$ konstant ist. Ein Schlüssel (a, b) gehört genau dann zu dieser Menge, wenn er die beiden Kongruenzen

$$\begin{aligned} ax + b &\equiv_p y, \\ ax' + b &\equiv_p y' \end{aligned}$$

erfüllt. Da dies jedoch nur auf den Schlüssel (a, b) mit

$$\begin{aligned} a &= (y' - y)(x' - x)^{-1} \bmod p, \\ b &= y - x(y' - y)(x' - x)^{-1} \bmod p \end{aligned}$$

zutrifft, folgt $\|K(x', y', x, y)\| = 1$. ∎

Die Konstruktion in obigem Beweis liefert "Hash"funktionen, die wegen $N = M = p$ nicht die Kompressionseigenschaft erfüllen. Eine Vergrößerung von N auf den Wert $p + 1$ ist allerdings möglich (siehe Übungen). Wie der folgende Satz zeigt, ist eine weiter gehende Kompression im Fall $\lambda = 1$ nicht möglich.

Satz 178 *Für jede stark universale* (N, M)*-Hashfamilie* $\mathcal{H}$ *mit* $\lambda = \|K\|/M^2 = 1$ *gilt* $N \leq M + 1$.

Beweis: Sei A die $M^2 \times N$-Authentikationsmatrix von $\mathcal{H}$ und o.B.d.A. sei $Y = \{1, \ldots, M\}$ angenommen. Es ist leicht zu sehen, dass eine (bijektive) Umbenennung $\pi : Y \to Y$ der Hashwerte in einer einzelnen Spalte von A wieder auf eine stark universale Hashfamilie führt. Also können wir o.B.d.A. annehmen, dass die erste Zeile von A nur Einsen enthält. Da $\mathcal{H}$ stark universal ist, gilt:

- In jeder der Zeilen $i = 2, \ldots, M^2$ kommt höchstens eine Eins vor.
- Jede der N Spalten enthält eine Eins in Zeile 1 und $M - 1$ Einsen in den übrigen Zeilen.

Da in den Zeilen $i = 2, \ldots, M^2$ insgesamt genau $N(M - 1)$ Einsen vorkommen, folgt

$$\underbrace{\text{Anzahl der Zeilen}}_{M^2} \geq \underbrace{\text{Anzahl der Zeilen mit einer Eins}}_{1+N(M-1)},$$

was $M^2 - 1 \geq N(M - 1)$ bzw. $N \leq M + 1$ impliziert. ∎