1 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“).

1.1 Klassifi kation 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 1 on page 120).

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****odifi cation* ***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 **Authentikationscode** bezeichnet. Daneben gibt es auch Hashverfahren mit asymmetrischen Schlüsseln. Diese werden jedoch der Rubrik der Signaturverfahren zugeordnet, da mit ihnen ausschließlich digitale Unterschriften gebildet werden.

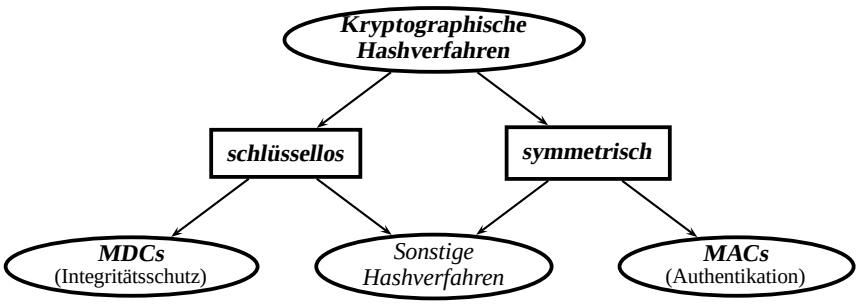

Abbildung 1 Eine grobe Einteilung von kryptographischen Hashverfahren.

1.2 Nachrichten-Authentikationscodes (MACs,)

Problemstellung:

- „Nachrichtenauthentikation“ (message authentication)
  - Wie kann ich sicherstellen, dass eine Nachricht (Datei) nicht verändert wurde?
  - Wie kann ich mich (und evtl. andere) davon überzeugen, dass eine Nachricht vom angegebenen Sender stammt?
- „Teilnehmerauthentikation“ (entity authentication, identification)
  - Wie kann ich mich anderen gegenüber zweifelsfrei ausweisen?

Definition 1 (**Authentikationscode**)

Ein **Authentikationscode (AC oder MAC)** besteht aus

- einer endlichen Menge M von möglichen Nachrichten
- einer endlichen Menge T von möglichen Authentikationsnummern (tags)
- einer endlichen Menge K von möglichen Schlüsseln
- einer Authentikationsvorschrift

$$\mathcal{A} : K \times M \to T$$

Wie sich Nachrichten mit einem MAC authentisieren lassen, ist in Abbildung 2 dargestellt.

Möchte Bob eine Nachricht x an Alice übermitteln, so berechnet er den zugehörigen *MAC*-Hashwert $y = \mathcal{A}(k, x)$ und fügt diesen der Nachricht

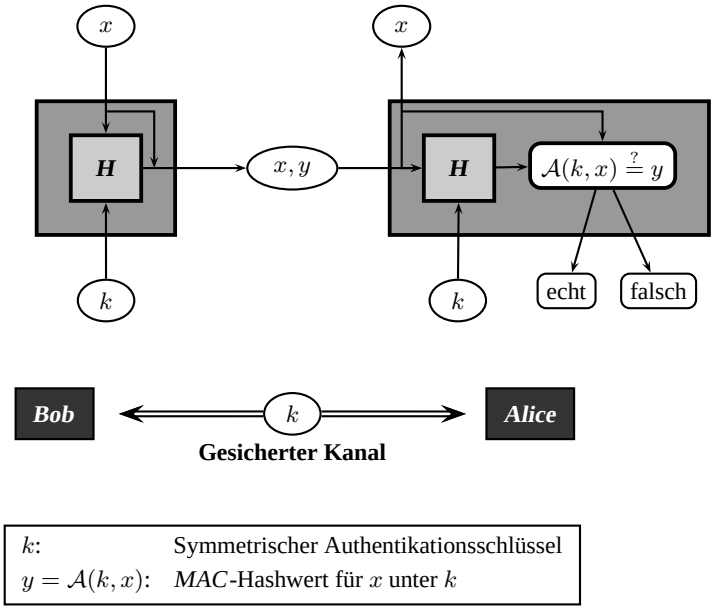

Abbildung 2 Ein MAC.

x hinzu. Alice überprüft die Echtheit der empfangenen Nachricht (x', y'), indem sie ihrerseits den zu x' gehörigen Hashwert $\mathcal{A}(k, x')$ 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.

Indem Bob seine Nachricht x um den Hashwert $y = \mathcal{A}(k, x)$ 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 folgende Bedingung erfüllt sein.

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

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 $y_1 = \mathcal{A}(k, x_1), \ldots, y_n = \mathcal{A}(k, x_n)$ 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 $\mathcal{A}(k, x_i)$ 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.

1.3 Angriffe gegen symmetrische Hashfunktionen

Ein Angriff gegen einen MAC hat die unbefugte Berechnung von Hashwerten zum Ziel. Das heißt, der Gegner versucht, Hashwerte $\mathcal{A}(k, x)$ 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 den benutzten Authentikationscode und versucht ein Paar (x, y) mit $\mathcal{A}(k, x) = y$ zu generieren, wobei k der (dem Gegner unbekannte) Schlüssel ist.

Substitution
: Der Gegner versucht in Kenntnis eines Paares $(x, \mathcal{A}(k, x))$ ein Paar (x', y') mit $\mathcal{A}(k, x') = y'$ 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 $\mathcal{A}(k, x_1), \ldots, \mathcal{A}(k, x_n)$ und versucht, ein Paar (x', y') mit $\mathcal{A}(k, x') = y'$ 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 $\mathcal{A}(k, x_j)$, $j < i$, abhängig machen.

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

1.4 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

$\boldsymbol{\alpha}$: 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 $K(x, y) = \{k \in K \mid \mathcal{A}(k, x) = y\}$ 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

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

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

$$\mathcal{A}(k, x) = ax + b \bmod 3.$$

Die zugehörige Authentikationsmatrix 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 $\mathcal{A}(k,x)$ 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 α ebenfalls den Wert $\alpha = 1/3$ ergibt. ◁

Satz 3 *Für jeden Authentikationscode $(M,T,K,\mathcal{A})$ gilt $\alpha \geq \frac{1}{\|T\|}$.*

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 $p(x \mapsto y) \geq \frac{1}{\|T\|}$ und dies impliziert

$$\alpha = \max_{x,y} p(x \mapsto y) \geq \frac{1}{\|T\|}.$$

■

Bemerkung 4 *Wie der Beweis zeigt, gilt $\alpha = \frac{1}{\|T\|}$ genau dann, wenn für alle Paare $(x,y) \in X \times Y$ gilt,*

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

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

Beispiel 5 *Sei $K = \{1,2,3\}$, $M = \{a,b,c,d\}$ und $T = \{0,1\}$.*

		0,1	0,2	0,3	0,4
	$\mathcal{A}(k,m)$	a	b	c	d
0,25	1	0	0	0	1
0,30	2	1	1	0	1
0,45	3	0	1	1	0

Annahme: $\mathbf{K}$ *hat Wahrscheinlichkeitsverteilung* $(0,25; 0,3; 0,45)$ *und* $\mathbf{M}$ *hat Wahrscheinlichkeitsverteilung* $(0,1; 0,2; 0,3; 0,4)$.

Dann hat der Gegner folgende Erfolgsaussichten, falls er das Paar (x, y) *an* $Alice$ *sendet:*

$\alpha(x,y)$	*0*	*1*
a	*0,7*	*0,3*
b	*0,25*	*0,75*
c	*0,55*	*0,45*
d	*0,45*	*0,55*

$\Rightarrow \alpha = 0,75.$

Erfolgswahrscheinlichkeit für Substitution

β: 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

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

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

$$p(x, y) = p(x)p(y|x) = p(x)p(x \mapsto y)$$

gesendet wird. Somit ergibt sich β zu

$$\beta = \sum_{x \in X, y \in Y} p(x, y)\beta(x, y) = \sum_{x \in X} p(x) \sum_{y \in Y} \beta'(x, y),$$

wobei

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

ist.

Beispiel 5 (Fortsetzung)

(x,y)	$p(x\mapsto y,x'\mapsto y')$								$\beta'(x,y)$	$\beta(x,y)$
	$(a,0)$	$(a,1)$	$(b,0)$	$(b,1)$	$(c,0)$	$(c,1)$	$(d,0)$	$(d,1)$		
$(a,0)$			*0,25*	***0,45***	*0,25*	***0,45***	***0,45***	*0,25*	*0,45*	*0,643*
$(a,1)$			*0*	***0,3***	***0,3***	*0*	*0*	***0,3***	*0,3*	*1*
$(b,0)$	***0,25***	*0*			***0,25***	*0*	*0*	***0,25***	*0,25*	*1*
$(b,1)$	***0,45***	*0,3*			*0,3*	***0,45***	***0,45***	*0,3*	*0,45*	*0,6*
$(c,0)$	*0,25*	*0,3*	*0,25*	*0,3*			*0*	***0,55***	*0,55*	*1*
$(c,1)$	***0,45***	*0*	*0*	***0,45***			***0,45***	*0*	*0,45*	*1*
$(d,0)$	***0,45***	*0*	*0*	***0,45***	*0*	***0,45***			*0,45*	*1*
$(d,1)$	*0,25*	*0,3*	*0,25*	*0,3*	***0,55***	*0*			*0,55*	*1*

Für β erhalten wir also den Wert

$$\begin{aligned}\beta &= 0,1\cdot(0,45+0,3)+0,2\cdot(0,25+0,45)+0,3\cdot(0,55+0,45)+0,4\cdot(0,45+0,55)\\ &= 0,915.\end{aligned}$$

Satz 6 *Für jeden Authentikationscode $(M,T,K,\mathcal{A})$ gilt $\beta \geq \frac{1}{\|T\|}$.*

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 $p(x' \mapsto y' \,|\, x \mapsto y) \geq \frac{1}{\|T\|}$ und dies impliziert für alle (x,y) mit $p(x,y) > 0$,

$$\beta(x,y) \;=\; \max\{p(x' \mapsto y' \,|\, x \mapsto y) \mid x' \in X - \{x\}, y' \in Y\} \;\geq\; \frac{1}{\|T\|}, \quad (1)$$

was wiederum

$$\beta = \sum_{x\in X, y\in Y} p(x,y)\beta(x,y) \geq \frac{1}{\|T\|}\sum_{x\in X, y\in Y} p(x,y) = \frac{1}{\|T\|}$$

impliziert. ■

Lemma 7 *Sei $(M,T,K,\mathcal{A})$ ein Authentikationscode mit $\beta = \frac{1}{\|T\|}$. Dann gilt*

$$p(x' \mapsto y' \,|\, x \mapsto y) = 1/\|T\|$$

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

Beweis: Wir zeigen zuerst, dass im Fall

$$\beta = \frac{1}{\|T\|}$$

für alle Paare $(x, y) \in X \times Y$

$$p(x \mapsto y) > 0$$

ist.
Ist nämlich

$$p(w \mapsto z) = 0,$$

so ist auch

$$p(w \mapsto z \,|\, u \mapsto v) = 0,$$

wobei $(u, v) \in X \times Y$ ein beliebiges Paar mit

$$p(u \mapsto v) > 0$$

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

$$p(w \mapsto z' \,|\, u \mapsto v) \geq 1/(\|T\| - 1) > 1/\|T\|.$$

Dann ist aber auch

$$\beta(u, v) = \max\{p(u' \mapsto v' \,|\, u \mapsto v) \mid u' \in X - \{u\}, v' \in Y\} > 1/\|T\|.$$

Da

$$\beta(x, y) \geq 1/\|T\|$$

für alle Paare (x, y) gilt (siehe (8)) und da

$$p(u, v) = p(u)p(u \mapsto v) > 0$$

ist, folgt

$$\beta = \sum_{x \in X, y \in Y} p(x, y)\beta(x, y) > 1/\|T\|.$$

Ist nun

$$p(x' \mapsto y' \,|\, x \mapsto y) \neq 1/\|T\|$$

für ein Doppelpaar (x, y, x', y') mit $x \neq x'$, so muss wegen

$$\sum_{z' \in Y} p(x' \mapsto z' \,|\, x \mapsto y) = 1$$

auch ein Doppelpaar (x, z', x', y') mit

$$p(x' \mapsto z' \,|\, x \mapsto y) > 1/\|T\|$$

existieren, was genau wie im ersten Teil des Beweises zu einem Widerspruch führt. ■

Satz 8 *Ein Authentikationscode $(M, T, K, \mathcal{A})$ erfüllt $\beta = \frac{1}{\|T\|}$ genau dann, wenn*

$$p(x \mapsto y, x' \mapsto y') = 1/\|T\|^2$$

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

Beweis: Sei $(M, T, K, \mathcal{A})$ ein MAC mit $\beta = \frac{1}{\|T\|}$.
Nach obigem Lemma impliziert dies, dass

$$p(x' \mapsto y' \,|\, x \mapsto y) = 1/\|T\|$$

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

$$p(x' \mapsto y') = \sum_y p(x \mapsto y) p(x' \mapsto y' \,|\, x \mapsto y) = 1/\|T\|$$

und daher

$$p(x \mapsto y, x' \mapsto y') = p(x' \mapsto y') p(x \mapsto y \,|\, x' \mapsto y') = 1/\|T\|^2.$$

Umgekehrt rechnet man leicht nach, dass $\mathcal{H}$ tatsächlich die Bedingung

$$\beta = \frac{1}{\|T\|}$$

erfüllt, wenn

$$p(x \mapsto y, x' \mapsto y') = 1/\|T\|^2$$

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

Bemerkung 9 *Nach obigem Satz gilt $\beta = \frac{1}{\|T\|}$ genau dann, wenn für alle Doppelpaare (x, y, x', y') mit $x \neq x'$ gilt,*

$$p(x \mapsto y, x' \mapsto y') = \sum_{k \in K(x,y,x',y')} p(k) = \frac{1}{\|T\|^2}.$$

D.h. bei Gleichverteilung der Schlüssel gilt $\beta = \frac{1}{\|T\|}$ 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 10 (stark universale Hashfamilien)

Ein MAC $(M, T, K, \mathcal{A})$ heißt stark universal, falls für alle $x, x' \in M$ mit $x \neq x'$ und alle $y, y' \in T$ gilt:

$$\|K(x, y, x', y')\| = \frac{\|K\|}{\|T\|^2}.$$

Bemerkung 11 *Bei der Konstruktion von stark universalen Hashfamilien spielt der Parameter $\lambda = \frac{\|K\|}{\|T\|^2}$ eine wichtige Rolle. Da λ notwendigerweise positiv und ganzzahlig ist, muss insbesondere $\|K\| \geq \|T\|^2$ gelten.*

Beispiel 12 *Betrachten wir den MAC $(M, T, K, \mathcal{A})$ mit $M = \{0, 1, 2, 3\}$, $T = \{0, 1, 2\}$, $K = \{0, 1, \ldots, 8\}$ 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 170 ist klar, dass ein MAC bei gleichverteilten Schlüsseln genau dann die Bedingung $\beta = \frac{1}{\|T\|}$ erfüllt, wenn er stark universal ist. Auf Grund von Bemerkung 165 nimmt in diesem Fall auch α den optimalen Wert $\frac{1}{\|T\|}$ an.

Der nächste Satz zeigt für primes p eine Konstruktionsmöglichkeit von stark universalen MACs mit dem Parameterwert $\lambda = 1$.

Satz 13 *Sei p prim und für $a, b, x \in \mathbb{Z}_p$ sei*

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

Dann ist (M, T, K, A) mit $M = T = \mathbb{Z}_p$ und $K = \mathbb{Z}_p \times \mathbb{Z}_p$ stark universal.

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 Hashfunktionen des vorigen Satzes erfüllen wegen $\|M\| = \|T\| = p$ nicht die Kompressionseigenschaft. Zwar lässt sich $\|M\|$ noch geringfügig von p auf $p+1$ vergrößern, ohne K und T (und damit λ) zu verändern (siehe Übungen), aber eine stärkere Kompression ist mit dem Parameterwert $\lambda = 1$ nicht realisierbar.

Satz 14 *Für jeden stark universalen MAC (M, T, K, A) mit $\lambda = \|K\|/\|T\|^2 = 1$ gilt $\|M\| \leq \|T\| + 1$.*

Beweis: O.B.d.A. sei $T = \{1, \ldots, \|T\|\}$.
Es ist leicht zu sehen, dass eine (bijektive) Umbenennung $\pi : T \to T$ der Hashwerte in einer einzelnen Spalte der Authentikationsmatrix A wieder auf einen stark universalen MAC führt.
Also können wir o.B.d.A. annehmen, dass die erste Zeile von A nur Einsen enthält.
Da (M, T, K, A) stark universal ist, gilt:

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

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

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

was $\|T\|^2 - 1 \geq \|M\|(\|T\| - 1)$ bzw. $\|M\| \leq \|T\| + 1$ impliziert. ■

Der nächste Satz liefert stark universale MACs mit beliebig großem Kompressionsfaktor. Für den Beweis benötigen wir das folgende Lemma.

Lemma 15 *Sei A eine $m \times l$-Matrix über einem endlichen Körper K, deren Zeilen linear unabhängig sind. Dann besitzt das lineare Gleichungssystem*

$$Ax = y$$

für jedes $y \in K^m$ genau $\|K\|^{l-m}$ Lösungen $x \in K^l$.

Beweis: Siehe Übungen. ■

Satz 16 *Sei p prim und für $x = (x_1, \ldots, x_l) \in \{0,1\}^l$ und $k = (k_1, \ldots, k_l) \in \mathbb{Z}_p^l$ sei*

$$\mathcal{A}(k, x) = kx = \sum_{i=1}^{l} k_i x_i \bmod p.$$

Dann ist (M, T, K, A) mit $M = \{0,1\}^l - \{0^l\}$, $T = \mathbb{Z}_p$ und $K = \mathbb{Z}_p^l$ stark universal.

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. Es gilt

$$\begin{aligned} k \in K(x, y, x', y') &\Leftrightarrow \mathcal{A}(k, x) = y \wedge \mathcal{A}(k, x') = y' \\ &\Leftrightarrow k \cdot x = y \wedge k \cdot x' = y'. \end{aligned}$$

Fassen wir $x = x_1 \cdots x_l$ und $x' = x'_1 \cdots x'_l$ zu einer Matrix A zusammen, so ist dies äquivalent zu

$$\begin{pmatrix} x_1 \cdots x_l \\ x'_1 \cdots x'_l \end{pmatrix} \cdot \begin{pmatrix} k_1 \\ \vdots \\ k_l \end{pmatrix} = \begin{pmatrix} y \\ y' \end{pmatrix}.$$

Da die beiden Zeilen von A verschieden und damit linear unabhängig sind, folgt mit obigem Lemma, dass genau $\|K(x, y, x', y')\| = p^{l-2}$ Schlüssel $k = (k_1, \ldots, k_l)$ mit dieser Eigenschaft existieren. ■

Bemerkung 17 *Obige Konstruktion liefert einen λ-Wert von $\frac{\|K\|}{\|T\|^2} = p^{l-2}$. Durch Erweiterung von M auf eine geeignete Teilmenge $M' \subseteq \mathbb{Z}_p^l$ lässt sich $\|M\|$ von $2^l - 1$ auf $\frac{p^l-1}{p-1}$ vergrößern (siehe Übungen). Wie der nächste Satz zeigt, kann der Kompressionsfaktor nicht weiter verbessert werden, ohne einen größeren Wert für λ in Kauf zu nehmen.*

Im Beweis des nächsten Satzes benötigen wir folgendes Lemma.

Lemma 18 *Seien $b_1, \ldots, b_m \in \mathbb{R}^m$. Dann gilt $\left(\sum_{i=1}^m b_i\right)^2 \leq m \sum_{i=1}^m b_i^2$.*

Beweis: Siehe Übungen. ■

Satz 19 *Sei $(M, T, K, \mathcal{A})$ stark universal mit $\|M\| = n$, $\|T\| = m$ und $\|K\| = l$. Dann gilt*

$$l \geq n(m-1) + 1,$$

d.h. $\lambda = l/m^2 \geq \frac{n(m-1)+1}{m^2}$.

Beweis: O.B.d.A. können wir annehmen, dass $T = \{1, \ldots, m\}$ ist und die 1.Zeile der Authentikationsmatrix von $\mathcal{A}$ nur aus Einsen besteht. Für jede Zeile i, $1 \leq i \leq l$, bezeichnet x_i die Anzahl der Einsen in Zeile i (also $x_1 = n$). Da in jeder Spalte jeder Hashwert genau λm-mal vorkommt, gilt

$$\sum_{i=1}^{l} x_i = \lambda n m$$

und

$$\sum_{i=2}^{l} x_i = \lambda n m - n = n(\lambda m - 1).$$

Nun ist die Anzahl der Vorkommen von Paaren (1,1) in den Zeilen $2, \ldots, l$ gleich

$$\sum_{i=2}^{l} x_i(x_i - 1) = \sum_{i=2}^{l} x_i^2 - \sum_{i=2}^{l} x - i = \sum_{i=2}^{l} x_i^2 - n(\lambda m - 1).$$

Mit obigem Lemma ergibt sich

$$\sum_{i=2}^{l} x_i^2 \geq \frac{(n(\lambda m-1))^2}{\lambda m^2 - 1} = \frac{(n(\lambda m-1))^2}{l-1} \Rightarrow \sum_{i=2}^{l} x_i(x_i - 1) \geq \frac{(n(\lambda m-1))^2}{l-1} - n(\lambda m - 1).$$

Da andererseits in jedem Spaltenpaar das Hashwert-Paar (1,1) in genau $\lambda - 1$ Zeilen $i \geq 2$ vorkommt, und da n(n-1) solche Spaltenpaare existieren, folgt

$$\sharp \text{ Vorkommen des HW-Paars (1,1) in Zeilen } i \geq 2 = (\lambda - 1)n(n-1) = \sum_{i=2}^{l} x_i(x_i - 1)$$

$$\begin{aligned}
&\Rightarrow \quad (\lambda-1)n(n-1) = \frac{(n(\lambda m-1))^2}{l-1} - n(\lambda m - 1) \\
&\Rightarrow \quad ((\lambda-1)n(n-1) + n(\lambda m - 1))(\lambda m^2 - 1) \geq (n(\lambda m - 1))^2 \\
&\Rightarrow \quad (\lambda n - n - \lambda + \lambda m)(\lambda m^2 - 1) \geq n(\lambda m - 1)^2 \\
&\Rightarrow \quad -\lambda^2 m^2 + \lambda^2 m^3 \geq \lambda n m^2 + \lambda n - \lambda + \lambda m - 2\lambda n m \\
&\Rightarrow \quad \lambda^2(m^3 - m^2) \geq \lambda(n(m-1)^2 + m - 1) \\
&\Rightarrow \quad \lambda m^2 \geq n(m-1) + 1 \\
&\Rightarrow \quad l \geq n(m-1) + 1
\end{aligned}$$

■

Für den Beweis des nächsten Satzes benötigen wir folgendes Lemma.

Lemma 20 *Sei $\mathcal{X}$ eine Zufallsvariable mit endlichen Wertebereich $W(\mathcal{X}) \supseteq \mathbb{R}^+$. Dann gilt* $\log E(\mathcal{X}) \geq E(\log \mathcal{X})$.

Satz 21 *Für jeden MAC $(M, T, K, \mathcal{A})$ gilt:*

$$\alpha \geq \frac{1}{2^{H(\mathcal{K}) - H(\mathcal{K}|\mathcal{X},\mathcal{Y})}}.$$

Hierbei sind $\mathcal{X}, \mathcal{Y}, \mathcal{K}$ Zufallsvariablen, die die Verteilungen der Nachrichten, der Hashwerte und der Schlüssel beschreiben.

Beweis: Wir zeigen: $\log \alpha \geq H(\mathcal{K}|\mathcal{X}, \mathcal{Y}) - H(\mathcal{K})$.

Es gilt: $\alpha = \max_{x,y} p(x \mapsto y)$, wobei

$$\begin{aligned}
p(x \mapsto y) &= Prob_k[\mathcal{A}(k,x) = y] \\
&= Prob[\mathcal{Y} = y | \mathcal{X} = x] \\
&=: p_{y|x} \\
\Rightarrow \alpha &\geq \sum_{x,y} Prob[\mathcal{X} = x, \mathcal{Y} = y] \cdot p(x \mapsto y) \\
&= E(\alpha(\mathcal{X}, \mathcal{Y})) \\
\Rightarrow \log \alpha &\geq \log E(\alpha(\mathcal{X}, \mathcal{Y})) \\
&\geq E(\log \alpha(\mathcal{X}, \mathcal{Y}))(*) \\
&= \sum_{x,y} p_{x,y} \cdot \log p_{y|x} \\
&= \sum_{x,y} p_x \cdot p_{y|x} \cdot \log p_{y|x} \\
&= -H(\mathcal{Y}|\mathcal{X}) \\
&\geq H(\mathcal{K}|\mathcal{X}, \mathcal{Y}) - H(\mathcal{K})(**)
\end{aligned}$$

Hierbei gilt (*) wegen obigem Lemma und (**) ergibt sich aus

$$\begin{aligned}
H(\mathcal{K}, \mathcal{Y}, \mathcal{X}) &= H(\mathcal{X}) + H(\mathcal{Y}|\mathcal{X}) + H(\mathcal{K}|\mathcal{X}, \mathcal{Y}) \\
&= \underbrace{H(\mathcal{K}, \mathcal{X})}_{=H(\mathcal{K})+H(\mathcal{X})} + \underbrace{H(\mathcal{Y}|\mathcal{K}, \mathcal{X})}_{=0}.
\end{aligned}$$

■

1.5 Codes zur Erkennung von Manipulationen (MDCs,)

Definition 22 (**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 der Schlüsselraum 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 meist, dass $n \geq 2m$ ist, und wir nennen h auch **Kompressionsfunktion** (*compression function*).

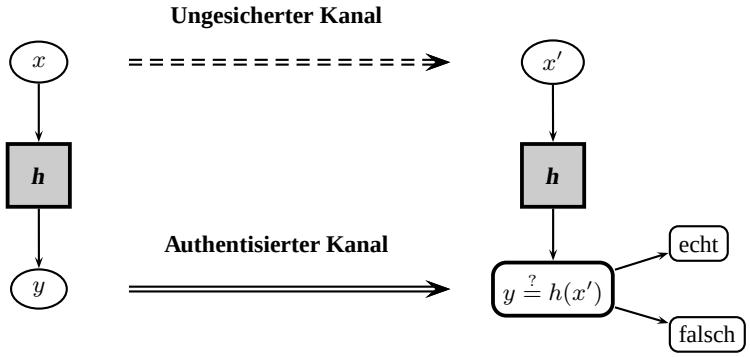

Abbildung 3 Einsatz eines MDC h zur Überprüfung der Integrität eines Datensatzes x.

1.6 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*).

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*).

Obwohl die schwache Kollisionsresistenz eine gewisse Ähnlichkeit mit der Einweg-Eigenschaft aufweist, 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.

1.7 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. Die Möglichkeit, außer durch derartige Orakelbefragungen Informationen über h zu erhalten, besteht im ZOM nicht.

Die gute Eignung einer Zufallsfunktion h für kryptografische Zwecke liegt darin begründet, dass der Hashwert $h(x)$ auf Grund der Gleichverteilung nur schwer vorhersagbar ist, auch wenn bereits eine Reihe von Werten $h(x_i)$ bekannt ist.

Proposition 1 *prop:zom 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 23 FINDPREIMAGE(h, y, q)

```
1  wähle eine Menge X_0 ⊆ X mit ||X_0|| = q
2  for each x ∈ X_0 do
3    if h(x) = y then
4      output x
5    end
```

6 **end**
7 **output** "?"

Satz 24 *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 $X_0 = \{x_1, \ldots, x_q\}$. Für $i = 1, \ldots, q$ 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 $i = 1, \ldots, q$ 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 Komplezität des Problems, ein zweites Urbild für $h(x)$ zu bestimmen:

Algorithmus 25 FINDSECONDPREIMAGE(h, x, q)

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

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

Satz 26 *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

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

eine Kugel zweimal gezogen wird. Für $m = 365$ und $q = 23$ ergibt dies einen Wert von ungefähr $0,507$.

Im nächsten Satz analysieren wir folgenden einfachen Algorithmus zur Kollisionsbestimmung.

Algorithmus 27 COLLISION(h, q)

```
1  wähle eine Menge X_0 ⊆ X mit ||X_0|| = q
2  for each x ∈ X_0 do
3    y_x ← h(x)
4  end
5  if y_x = y_{x'} für zwei Texte x ≠ x' in X_0 then
6    output (x, x')
7  else
8    output "?"
9  end
```

Bei einer naiven Vorgehensweise 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 (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 28 *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.

1.8 Vergleich von Sicherheitsanforderungen

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

Satz 29 *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

```
1  wähle zufällig x ∈ X
2  x' ← A(x)
3  if x' ≠ "?" then
4    output (x, x')
5  else
6    output "?"
7  end
```

mit Wahrscheinlichkeit ε ein Kollisionspaar ausgibt. ■

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

Satz 30 *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:

```
1  wähle zufällig x ∈ X
2  y ← h(x)
3  x' ← A(y)
4  if x ≠ x' then
5    output (x, x')
6  else
7    output "?"
8  end
```

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

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

■

1.9 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 $y : \{0,1\}^{\geq m+t} \rightarrow \{0,1\}^*$ 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 $g : \{0,1\}^m \rightarrow \{0,1\}^l$ 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 $y : \{0,1\}^{\geq m+t} \rightarrow \{0,1\}^{\geq m+t}$ definiert durch $y(x) = y_1 \cdots y_{k+1}$, wobei

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

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 31 *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: $|x| \not\equiv |\tilde{x}|, \quad |x| \equiv |x'| \pmod{t-1}$. 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 $y : \{0,1\}^{\geq m+2} \to \{0,1\}^*$ die Funktion $y(x) := 11 f(x)$. 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 $y(x) = z y(x')$ 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 $a_0a_1a_2a_3$, $0 \leq a_i \leq 255$ 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. $a_0a_1a_2a_3$, $0 \leq a_i \leq 255$ 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 32 MD4(x)

Eingabe: $x \in \{0,1\}^*, |x| = n$
  $y \leftarrow x10^k \operatorname{bin}_{64}(n)$, $k \in \{0,1,\ldots,511\}$ mit $n+1+k+64 \equiv 0 \pmod{512}$
  $(H_1, H_2, H_3, H_4) \leftarrow (67452301, efcdab89, 98badcfe, 10325476)$
  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: $H_1H_2H_3H_4$

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 $\text{abs}(\sin(j+1))$, $0 \leq j \leq 63$, 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 33 MD5(x)

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

4 sei $y = M_1 \cdots M_r$, $r = (n + 1 + k + 64)/512$
5 **for** $i \leftarrow 1$ **to** r **do**
6 sei $M_i = X[0] \cdots X[15]$
7 $(A, B, C, D) \leftarrow (H_1, H_2, H_3, H_4)$
8 **for** $j \leftarrow 0$ **to** 63 **do**
9 $(A, B, C, D) \leftarrow (D, (A + f_j(B, C, D) + X[z_j] + y_j) \hookleftarrow s_j, B, C)$
10 **end**
11 $(H_1, H_2, H_3, H_4) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D)$
12 **end**
13 **Ausgabe:** $H_1H_2H_3H_4$

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 34 SHA-1(x)

1 **Eingabe:** $x \in \{0,1\}^*, |x| = n$
2 $y \leftarrow x10^k \operatorname{bin}_{64}(n)$, $k \in \{0, 1, \ldots, 511\}$ mit $n + 1 + k + 64 \equiv 0 \pmod{512}$
3 $(H_0, H_1, H_2, H_3, H_4) \leftarrow (67452301, efcdab89, 98badcfe, 10325476, c3d2e1f0)$
4 sei $y = M_1 \cdots M_r$, $r = (n + 1 + k + 64)/512$
5 **for** $i \leftarrow 1$ **to** r **do**
6 sei $M_i = X[0] \cdots X[15]$
7 **for** $t \leftarrow 16$ **to** 79 **do**
8 $X[t] \leftarrow (X[t-3] \oplus X[t-8] \oplus X[t-14] \oplus X[t-16]) \hookleftarrow 1$
9 **end**

10 $(A, B, C, D, E) \leftarrow (H_0, H_1, H_2, H_3, H_4)$
11 **for** $j \leftarrow 0$ **to** 79 **do**
12 $temp \leftarrow (A \hookleftarrow 5) + f_j(B, C, D) + E + X[t] + K_t$
13 $(A, B, C, D, E) \leftarrow (temp, A, B \hookleftarrow 30, C, D)$
14 **end**
15 $(H_0, H_1, H_2, H_3, H_4) \leftarrow (H_0 + A, H_1 + B, H_2 + C, H_3 + D, H_4 + E)$
16 **end**
17 **Ausgabe:** $H_0H_1H_2H_3H_4$

2 Digitale Signaturverfahren

Handschriftliche Signaturen

- Die durch die Unterschrift gekennzeichnete Person hat überprüfbar die Unterschrift geleistet.
- Die Unterschrift ist nicht auf ein anderes Dokument übertragbar.
- Das signierte Dokument kann nachträglich nicht unbemerkt verändert werden.

Eine direkte Übertragung dieser Eigenschaften in die digitale Welt ist nicht möglich.

Bemerkung 35 *Lösung:*
Die digitale Unterschrift wird nicht physikalisch, sondern logisch (inhaltlich) an das elektronische (digitale) Dokument gebunden.

Digitale Signaturen

Definition 36 (Digitales Signaturverfahren)

Ein digitales Signaturverfahren besteht aus:

- endlicher Menge X von Dokumenten
- endlicher Menge Y von Unterschriften
- endlicher Menge K von Schlüsseln
- $S \subseteq K \times K$ von Schlüsselpaaren $(\overline{K}, K)$
- Signaturalgorithmus $sig : K \times X \to Y$
- Verifikationsalgorithmus $ver : K \times X \times Y \to \{0, 1\}$

mit

$$ver(k, x, y) = \begin{cases} 1, & sig(\overline{k}, x) = y \\ 0, & sonst \end{cases}$$

für alle $(\overline{k}, k) \in S$.

Erfolgskriterien für die Fälschung digitaler Signaturen

• uneingeschränktes Fälschungsvermögen (total break):
d.h. der Gegner hat einen Weg gefunden, die Funktion $x \mapsto sig(\overline{k}, x)$, effizient zu berechnen ohne $\overline{k}$ als Eingabe zu benutzen. (k ist bekannt).

• selektives Fälschungsvermögen (selective forgery):
d.h. der Gegner kann für bestimmte von ihm selbst ausgewählte Dokumente die zugehörige Signatur bestimmen (eventuell mit Hilfe des legalen Unterzeichners).

• nichtselektives (existentielles) Fälschungsvermögen:
d.h. der Gegner kann für irgendein Dokument x die zugehörige digitale Signatur bestimmen.

2.1 Klassifikation von Angriffen gegen Signaturverfahren

• Angriff bei bekanntem Verifikationsschlüssel (key-only attack)

• Angriff bei bekannter Signatur (known signature attack):
d.h. für eine Reihe von Dokumenten x ist die zugehörige Signatur $y = sig(\overline{k}, x)$ bekannt, auf deren Auswahl der Gegner keinen Einfluß hat.

• Angriff bei frei wählbaren Dokumenten (chosen document attack):
d.h. der Gegner war für eine gewisse Zeit in der Lage, für von ihm gewählte Dokumente die zugehörige Signatur in Erfahrung zu bringen und versucht nun für ein "neues" Dokument die Unterschrift zu bestimmen.

• adaptiver Angriff bei frei wählbaren Dokumenten:
d.h. der Gegner wählt jeweils das nächste Dokument in Abhängigkeit von der Signatur des vorigen.

Beispiel 37 *(RSA-Signaturverfahren)*
$K = \{(a, n) | n = pq$ *für Primzahlen* p, q *und* $a \in \mathcal{Z}^*_{\varphi(n)}\}$

$S = \{(d, n, e, n) \in K \times K | de \equiv_{\varphi(n)} 1\}$

$sig(d, n, x) := x^d \bmod n$, $x \in X = \mathcal{Z}_n$

$$ver(e, n, x, y) := \begin{cases} 1, & y^e \equiv_n x \quad \forall x, y \in Y = X \\ 0, & sonst \end{cases}$$ ◁

Satz 38 $\forall (d, n, e, n) \in S$ *und* $x, y \in \mathcal{Z}_n$ *gilt:*

$$ver(e, n, x, y) = \begin{cases} 1, & sig(d, n, x) = y \\ 0, & sonst \end{cases}$$

Beweis: siehe RSA-Kryptosystem ■

Nichtselektive Fälschung bei Angriff mit bekanntem Verifikationsschlüssel: .

Geg: $k = (e, n)$
Ges: (x, y) mit $ver(k, x, y) = 1$ (Schliessen 0,1 als Dokument aus)

Zu beliebigem $y \in Y$ kann durch $x = y^e \bmod n$ leicht ein Dokument x bestimmt

werden mit $ver(k, x, y) = 1$.
1. Lösungsweg: Verwendung einer Hashfunktion H.
2. Lösungsweg: Dokument x wird mit Redundanz versehen;
z.B. $x \rightarrow xx$ (sehr ineffizient für die Funktion)

Von H benötigte Eigenschaften

• H sollte eine Einweg-Hashfunktion sein, da sonst der Gegner zu $H(x)$ ein passendes x bestimmen kann (zumindest wenn das Signaturverfahren anfällig gegen eine nicht-selektive Fälschung ist, wie etwa RSA)

• Angenommen der Gegner kennt bereits ein Paar (x, y) mit $ver(k, H(x), y) = 1$. Dann sollte H schwach kollisionsresistent sein, da sonst der Gegner ein x' mit $H(x') = H(x)$ berechnen und das Paar (x', y) bestimmen könnte.

• Falls sich der Gegner für ein selbst gewähltes Dokument x die zugehörige Signatur y beschaffen kann, so sollte H kollisionsresistent sein, da sonst der Gegner ein Kollisonspaar (x, x') für H berechnen kann (und sich zu x die zugehörige Signatur beschaffen). Dann gilt $ver(k, x', y) = 1$.

Weitere Angriffe gegen das RSA-Verfahren (ohne Hash-Funktion)

• Angenommen der Gegner kennt zwei Paare $(x_1, y_1), (x_2, y_2)$ mit $ver(k, x_i, y_i) = 1$, dh. $y_i^e \equiv_n x_i \; i = 1, 2$.
$\Rightarrow$ Dann gilt auch $ver(k, x_1x_2 \bmod n, y_1y_2 \bmod n) = 1$, da $(y_1y_2)^e \equiv_n y_1^e y_2^e \equiv_n x_1x_2$ ist. (Nicht-selektive Fälschung bei bekannten Signaturen)

• Selektive Fälschung bei frei wählbaren Dokumenten
Geg: $x \in X, k$
Ges: $y \in Y$ mit $ver(k, x, y) = 1$,
wobei für beliebige Dokumente $x' \neq x$ die Signatur erfragt werden kann.

Der Gegner kann einfach $x_1 \in \mathcal{Z}_n^*$ wählen und sich die Signatur zu $x \cdot x_1 \bmod n$ und $x_1^{-1} \bmod n$ beschaffen.

2.2 Der diskrete Logarithmus

• Sei $(G, *)$ eine Gruppe und sei $\alpha \in G$.

• Sei $ord_G(\alpha) = n$, dann ist $< \alpha >=_{def} \{\alpha^i | i = 0 \cdots n-1\}$ die von α in G erzeugte

Untergruppe.

• $| < \alpha > | = n$

Beispiel 39 *Untergruppen-Ordnung*

1. *Sei* $G = (\mathcal{Z}_p^*, *)$, p *prim und* α *Primitiv-Wurzel* $\mathrm{mod} p$.
   Dann ist $< \alpha >= \mathcal{Z}_p^*$.
2. *Sei* $G = (\mathcal{Z}_p^*, *)$ *und sei* $q|p-1$.
   Sei α *die Primtiv-wurzel* $\mathrm{mod} p$.
   Setze $\beta = \alpha^{\frac{p-1}{q}}$.
   Dann ist $< \beta >$ *echte Untergruppe von* $\mathcal{Z}_p^*$ *und* $\| < \beta > \| = q$.

◁

Das Problem des diskreten Logarithmus

Eingabe: Gruppe $(G, *)$ $\alpha \in G$, $ord_G(\alpha) = n$, $\beta \in < \alpha >$

Gesucht: Die eindeutig bestimmte Zahl a mit $0 \leq a \leq n-1$, so daß $\beta = \alpha^a$.

Bezeichnung: $a = \log_\alpha \beta$.

Es ist kein effizienter Algorithmus zur Bestimmung von $\log_\alpha$ bekannt, jedoch ist die Umkehrfunktion effizient berechenbar, mittels wiederholtem Quadieren.
Damit ist α^n Kandidat einer Einwegfunktion.

Das El Gamal-Signaturverfahren

Sei p Primzahl und $\alpha \in \mathcal{Z}_p^*$ sei Primitiv-Wurzel.
$\mathcal{P}$= Menge der Nachrichten = $\mathcal{Z}_p^*$
$\mathcal{A}$= Menge der Signaturen = $\mathcal{Z}_p^* \times \mathcal{Z}_{p-1}$
$\mathcal{K}$= Menge der Schlüssel = $\{(p, \alpha, \beta, a) | \beta \equiv \alpha^a \mod p\}$

öffentlich: p, α, β
geheim: a

Signatur: Wähle zufällig $k \in \mathcal{Z}_{p-1}^*$ und setzen $sig_K(x, k) = (\gamma, \delta)$ mit $\gamma \equiv \alpha^k \mod p$ und $\delta = (x - a\gamma)k^{-1} \mod p - 1$.

Verifikation: $ver_K(x, (\gamma, \delta)) = ja \Leftrightarrow_{def} \beta^\gamma \gamma^\delta \equiv \alpha^x \mod p$ mit $K \in \mathcal{K}$,
d.h. $K = (p, \alpha, \beta, a)$.

Bemerkung 40 *Ist* $sig_K(x,k) = (\gamma, \delta)$, *so gilt* $ver_K(x,(\gamma,\delta)) = ja$.

Beweis: $ver_K(x,(\gamma,\delta)) = ja \Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^X \bmod p$.
$\beta^\gamma \gamma^\delta \equiv \alpha^{a\gamma}\alpha^{k\delta} \equiv \alpha^{a\gamma+k\delta} \equiv_{(*)} \alpha^{x+n(p-1)} \equiv \alpha^x \alpha^{n(p-1)} \equiv \alpha^x \bmod p$

(*) $a\gamma + k\delta \equiv x \bmod p-1 \Rightarrow a\gamma + k\delta = x + n(p-1)$ für ein $n \in IN$. ■

Bemerkung 41 *Das Signaturverfahren erhält man wie folgt:*

Beginne mit $\beta^\gamma \gamma^\delta \equiv \alpha^x \bmod p$
Setze $\gamma = a^k \bmod p$ *und* $\beta = \alpha^a \bmod p \Rightarrow \alpha^x \equiv \alpha^{a\gamma}\alpha^{k\delta}$ *(*)*

(*) gilt genau dann, wenn $x \equiv a\gamma + k\delta \bmod p-1$ *weil* α *Primitiv-Wurzel* $\bmod p$.
$\Rightarrow$ *(*) gilt genau dann, wenn* $\delta \equiv (x - a\gamma)k^{-1} \bmod p-1$

Sicherheit des El Gamal-Systems

1. Falls Oskar den diskreten Logarithmus bestimmen kann, so kann er den geheimen Schlüssel $a = \log_\alpha \beta$ bestimmen.
2. Selektiver Angriff für gegebenen Klartext x:
   (a) Oskar wählt zuerst γ und versucht ein passendes δ zu finden.
   Mit $\alpha^x \equiv \beta^\gamma \gamma^\delta \bmod p$ folgt $\delta = \log_\gamma \alpha^x \beta^{-\gamma}$.
   D.h. die Bestimmung von δ ist Instanz des dikreten Logarithmus.
   (b) Oscar wählt zuerst δ und versucht dann γ zu bestimmen aus $\alpha^x \equiv \beta^\gamma \gamma^\delta \bmod p$.
   Dazu ist kein effizientes Verfahren bekannt.
   (c) Oscar wählt γ und δ gleichzeitig.
   Auch dazu ist kein effizientes Verfahren bekannt.
3. Oskar wählt γ und δ und versucht x zu bestimmen.
   Dann muss Oscar $x = \log_\alpha \beta^\gamma \gamma^\delta$ bestimmen.
4. Oskar kann jedoch 'zufällige' Nachrichten x gültig signieren, und zwar wie folgt:
   (a) Oskar wählt $i \in \mathcal{Z}_{p-1}$, $j \in \mathcal{Z}^*_{p-1}$ und setzt $\gamma = \alpha^i \beta^j \bmod p$.
   (b) • $(x,(\gamma,\delta))$ ist gültige Signatur $\Leftrightarrow \alpha^x \equiv \beta^\gamma (\alpha^i \beta^j)^\delta \bmod p$.
   • Dies gilt genau dann, wenn $\alpha^{x-i\delta} \equiv \beta^{\gamma+j\delta} \bmod p$ (*)
   • (*) gilt, falls $x - i\delta \equiv 0 \bmod p-1 \wedge \gamma + j\delta \equiv 0 \bmod p-1$

   Dies ergibt: $\delta \equiv -\gamma j^{-1} \bmod p-1 \wedge x \equiv -i\gamma j^{-1} \bmod p-1$

   Für beliebig gewählte $i \in \mathcal{Z}_{p-1}$, $j \in \mathcal{Z}^*_{p-1}$ ist also folgendes eine gültige Signatur:

$$x \equiv -ij^{-1}\alpha^i\beta^j \bmod p-1$$
$$\gamma \equiv \alpha^i\beta^j \bmod p$$
$$\delta \equiv -j^{-1}\alpha^i\beta^j \bmod p-1$$

Bemerkung 42 *Bei der Benutzung des El Gamal-Signaturverfahrens ist zu beachten:*

1. *Die Zufallszahl k muss geheim gehalten werden*
2. *Zufallszahlen dürfen nicht mehrfach verwendet werden*

Beweis: (der Bemerkung)
zu 1.

Angenommen Oskar kennt zu einer Signatur $(x,(\gamma,\delta))$ mittels p,α,β auch k.
Dann gilt: $\delta \equiv (x-a\gamma)k^{-1} \bmod p-1 \Rightarrow a \equiv (-k\delta+x)\gamma^{-1} \bmod p-1$

zu 2.

Angenommen $(x_1,(\gamma,\delta_1))$ und $(x_2,(\gamma,\delta_2))$ sind mit demselben k signiert.
Dann ist $\beta^\gamma\gamma^{\delta_1} \equiv \alpha^{x_1} \bmod p \wedge \beta^\gamma\gamma^{\delta_2} \equiv \alpha^{x_2} \bmod p$.

$$\begin{aligned} &\Rightarrow \quad \gamma^{\delta_1-\delta_2} \equiv \alpha^{x_1-x_2} \bmod p \\ &\Rightarrow \quad \alpha^{k(\delta_1-\delta_2)} \equiv \alpha^{x_1-x_2} \bmod p \\ &\Rightarrow \quad k(\delta_1-\delta_2) \equiv x_1-x_2 \bmod p-1 (*) \end{aligned}$$

Sei $d = ggT(\delta_1-\delta_2, p-1) \Rightarrow d|x_1-x_2$

Setze $x' = \frac{x_1-x_2}{d}$, $\delta' = \frac{\delta_1-\delta_2}{d}$, $p' = \frac{p-1}{d}$.

$$\begin{aligned} &\Rightarrow \quad k\cdot\delta' \equiv x' \bmod p' \text{ wegen (*)} \\ &\Rightarrow \quad ggT(\delta',p') = 1 \Rightarrow k \equiv x'(\delta')^{-1} \bmod p' \\ &\Rightarrow \quad k = x'(\delta')^{-1} + ip' \text{ für } i = 0,\cdots,d-1 \text{ sind Lösungen für (*).} \end{aligned}$$

Bestimme das korrekte k mittels Überprüfen von $\gamma \equiv \alpha^k \bmod p$
$\Rightarrow$ Kenntnis von k $\Rightarrow$ Kenntnis von a (wie bei 1.)

■

2.3 Varianten des ElGamal-Signaturverfahrens

Das Schnoor-Signaturverfahren

Da p im El Gamal-Signaturverfahren mindestens eine 512-Bit-Zahl (besser 1024-Bit-Zahl) sein sollte, beträgt die Signaturlänge 1024 bzw 2048 Bit.

Idee:
Indem wir für α ein Element der Ordnung q mit $q \approx 2^{160}$ wählen, reduziert sich die Signaturlänge auf $2 \cdot 160 = 320$ Bit.
Aber die Berechnungen werden modulo p mit $p \approx 2^{1024}$ ausgeführt, so dass das Problem des diskreten Logarithmus hart bleibt.

Sei g ein Erzeuger von $\mathcal{Z}_p^*$, wobei p die Bauart $p - 1 = mq$ für q Prim und $q = \frac{p-1}{m} \approx 2^{160}$ hat.

Definition 43 ()

Sei $p = mq + 1$ prim, q ebenfalls prim und sei $\alpha \in \mathcal{Z}_p^*$ mit $ord_p(\alpha) = q$.
Weiter sei $h : \{0,1\}^* \to \mathcal{Z}_q$ eine Hashfunktion.
$X := \{0,1\}^*$
$Y := \mathcal{Z}_q \times \mathcal{Z}_q$
$S := \{(p, q, \alpha, \beta, a) | \beta \equiv_p \alpha^a\}$

Für eine geheime Zufallszahl $r \in \mathcal{Z}_q^*$ definiere

$$sig(k, \overline{k}, r, x) := (\gamma, \delta),$$

wobei

$\overline{k} = (p, q, \cdots, a)$
$\gamma = h(x \text{ bin}(\alpha^r \bmod p))$
$\delta = (r + a\gamma) \bmod q$
$ver(k, \gamma, \delta) = \begin{cases} 1, & h(x \text{ bin}(\alpha^\delta \beta^{-\gamma})) = \gamma \\ 0, & sonst \end{cases}$

Leicht zu sehen:

$$ver(k, \gamma, \delta) = 1 \Leftrightarrow h(x \text{ bin}(\alpha^\delta \beta^\gamma)) = \gamma$$

Der Digital Signature Algorithm (DSA)

(Standard in USA seit 1994)

Hierbei wird das El Gamal-Verfahren wie folgt modifiziert:

1. z als Lösung von $rz - ay \equiv_{p-1} x$ (d.h. $z = (x + ay)r^{-1}$)
   $\rightsquigarrow$ Verifikationsbedingung: $g^x b^y \equiv_p y^z$ ($g^x g^{ay} \equiv_p g^{r(x+ay)r^{-1}}$)
2. Ist $x + ay \in \mathcal{Z}_{p-1}^*$, dann existiert $z^{-1} = (x + ay)^{-1} r \bmod p - 1$
   $\rightsquigarrow$ Verifikation durch: $g^{xz^{-1}} b^{yz^{-1}} \equiv_p y$
3. Sei q ein Primteiler von $p - 1$ mit $q \approx 2^{160}$ und sei $\alpha \in \mathcal{Z}_p^*$ mit $ord_p(\alpha) = q$.
   $\rightsquigarrow$ Bei der Verifikation von $g^{xz^{-1}} b^{yz^{-1}} \equiv_p y$ kann auf der Exponentenebene

$modulo\ q$ gerechnet werden, wenn wir g durch α und $b = g^a$ durch $b = \alpha^a$ ersetzen.
Da y jedoch rechts nicht als Exponent, sondern als Basiszahl, vorkommt, muss auch die linke Seite $modulo\ q$ reduziert werden.

Definition 44 (Digital Signature Algorithm)

p prim, 512-1024 Bit
q Primteiler von $p-1$, 160 Bit
$\alpha \in \mathcal{Z}_p^*$ mit $ord_p(\alpha) = q$
$X = \mathcal{Z}_p^*$, $Y = \mathcal{Z}_q \times \mathcal{Z}_q^*$
Signierschlüssel: $\overline{k} = (p, q, \alpha, a)$, wobei $a \in \mathcal{Z}_p^*$
Verifikationsschlüssel: $k = (p, q, \alpha, \beta)$ mit $\beta \equiv_p \alpha^a$

Zu gegebenem $x \in X$ wird zufällig eine geheime Zahl $r \in \mathcal{Z}_p^*$ gewählt.

$$sig(\overline{K}, r, x) = (y, z), \text{ wobei } \begin{cases} y = (\alpha^r \bmod p) \bmod q \\ z = (x + ay)r^{-1} \bmod q \in \mathcal{Z}_q^* \end{cases}$$

(falls $z \equiv_q 0$ muss ein neues r gewählt werden)

$$ver(k, x, y, z) = \begin{cases} 1, & (\alpha^e \beta^d \bmod p) \bmod q = y \\ 0, & sonst \end{cases}$$

wobei $e = xz^{-1} \bmod q$, $d = yz^{-1} \bmod q$

Im Fall $sig(\overline{k}, r, x) = (y, z)$ gilt:

$$\alpha^e \beta^d \equiv_p \alpha^{xz^{-1}} \alpha^{ayz^{-1}} \equiv_p \alpha^{z^{-1}(x+ay)} \equiv_p \alpha^{(x+ay)^{-1} r(x+ay)} \equiv_p \alpha^r$$

$$\rightsquigarrow (\alpha^e \beta^d \bmod p) \bmod q = (\alpha^r \bmod p) \bmod q = y$$

Beispiel 45 $q = 101, p = 78q + 1 = 7879, g = 3$ *(*$ord_p(3) = p - 1$*)*

$$\rightsquigarrow \alpha = 3^{78} \bmod p = 170 \textit{ hat Ordnung } q$$

Wir wählen $a = 75 \in \mathcal{Z}_q^*$*, d.h.* $\beta = \alpha^a \bmod p = 170^{75} \bmod p = 4547$
Um das Dokument $x = 1234 \in \mathcal{Z}_p^*$ *zu signieren, wählen wir die geheime Zufallszahl* $r = 50 \in \mathcal{Z}_p^*$ *(*$\rightsquigarrow r^{-1} = 99$*) und erhalten dann*

$$\begin{aligned} y &= (170^{50} \bmod 7879) \bmod 101 \\ &= 2518 \bmod 101 \\ &= 94 \\ z &= (1234 + 75 \cdot 94) \cdot 99 \bmod 101 \\ &= 97 \ (\rightsquigarrow z^{-1} = 25) \end{aligned}$$

d.h. $sig(p, q, \alpha, r, x) = (94, 97)$*, wobei* $\overline{k} = (p, q, \alpha, a)$

Um diese Signatur zu prüfen berechnen wir:

$$\begin{aligned} e &= xz^{-1} \bmod q \\ &= 1234 \cdot 25 \bmod 101 \\ &= 45 \\ d &= yz^{-1} \bmod q \\ &= 94 \cdot 25 \bmod 101 \\ &= 27 \end{aligned}$$

$\rightsquigarrow (\alpha^e \beta^d \bmod p) \bmod q = (170^{45} 4547^{27} \bmod 7879) \bmod 101 = 94$

◁

2.4 Elliptische Kurven

Definition 46 (nicht-singuläre elliptische Kurve)

Seien $a, b \in IR$ mit $4a^3 + 27b^2 \neq 0$.
Eine nicht-singuläre elliptische Kurve E enthält alle Lösungen $(x, y) \in IR^2$ der Gleichung $y^2 = x^3 + ax + b$ und zusätzlich der Punkt ∞.

Definition 47 (Gruppenoperation $+$ auf E)

• Neutrales Element ∞, d.h. $P + \infty = P \; \forall P \in E$
• Für $P = \{x_1, y_1\}, Q = \{x_2, y_2\} \in E - \{\infty\}$ betrachten wir folgende Fälle:

1. $x_1 \neq x_2$

   Sei $G = \{(x, y) \in IR^2 | y = \lambda x + \mu\}$, wobei $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ und $\mu = y_1 - \lambda x_1 = y_2 - \lambda x_2$ ist.

   Behauptung:

   Für $R = (x_3, y_3)$ mit

   $$x_3 = \lambda^2 - x_1 - x_2$$

   und

   $$y_3 = \lambda(x_3 - x_1) + y_1,$$

   wobei $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$,
   gilt

   $$E \cap G = \{P, Q, R\}.$$

Beweis: der Behauptung
Für alle $(x, y) \in E \cap G$ gilt:

$$(\lambda x + \mu)^2 = x^3 + ax + b$$

$$\rightsquigarrow \underbrace{x^3 - \lambda^2 x^2 + (a - 2\mu\lambda)x + b - \mu^2}_{p(x)} = 0.$$

p läßt sich in $\mathcal{C}$ vollständig in Linearfaktoren zerlegen:

$$p(x) = (x - x_1)(x - x_2)(x - x_3)$$

Da x_1, x_2 und alle Koeffizienten von $p \in IR$ sind, muss auch $x_3 \in IR$ sein.
Der Koeffizient $-\lambda^2$ von x^2 berechnet sich aus der linearen Zerlegung von $p(x)$ zu

$$-\lambda^2 = -x_1 - x_2 - x_3 \rightsquigarrow x_3 = \lambda^2 - x_1 - x_2$$

Wegen $\lambda = \frac{y_3 - y_1}{x_3 - x_1}$ erhalten wir dann $y_3 = \lambda(x_3 - x_1) + y_1$. ■

Wir definieren $P + Q := \overline{R} = (x_3, -y_3)$.

2. $x_1 = x_2, y_1 = -y_2$

   Wir definieren $P + Q := \infty$

3. $x_1 = x_2, y_1 = y_2$, d.h. $P = Q$, $y_1 \neq 0$

   Sei T die Tangente durch P an E.

   Behauptung:

   Es gibt ein $R = (x_3, y_3) \in IR^2$ mit

   $$x_3 = \lambda^2 - 2x_1$$

   und

   $$y_3 = \lambda(x_3 - x_1) + y_1,$$

   so daß gilt

   $$T \cap E = \{P, R\}.$$

   Die Steigung λ von T erhalten wir durch implizites Differenzieren:

   $$\lambda = \frac{dy}{dx} = \frac{-\frac{\delta F}{\delta x}(x_1, y_1)}{\frac{\delta F}{\delta y}(x_1, y_1)} = \frac{3x_1^2 + a}{2y_1}$$

   wobei $F(x, y) = y^2 - x^3 - ax - b$ ist.

   Begründung:

Sei

$$E(x,y) = ax + by + c$$

die Tangentialebene an $F(x,y)$ im Punkt (x_1, y_1).
Dann gilt

$$a = \frac{\delta F}{\delta x}(x_1, y_1)$$

und

$$b = \frac{\delta F}{\delta y}(x_1, y_1).$$

T ist dann der Schnitt von E mit der x,y-Ebene, d.h.

$$\begin{aligned}(x,y) \in T &\Leftrightarrow E(x,y) = 0 \\ &\Leftrightarrow y = -\frac{a}{b}x - \frac{c}{b}\end{aligned}$$

Genau wie im 1.Fall erhalten wir nun:

$$\begin{aligned}x_3 &= \lambda^2 - x_1 - x_2 = \lambda^2 - 2x_1 \\ y_3 &= \lambda(x_3 - x_1) + y_1\end{aligned}$$

Wir definieren $P + Q = P + P = 2P := \overline{R} = (x_3, -y_3)$

Satz 48 *E bildet mit ∞ als neutralem Element und $+$ als Addition eine abelsche Gruppe, d.h.*

- *$+$ ist abgeschlossen auf E*
- *$+$ ist kommutativ*
- *jeder Punkt hat ein Inverses $-P$*
  ($P = -P \Leftrightarrow P = \infty \vee P = (x, 0)$)
- *$+$ ist assoziativ (ohne Beweis!)*

Additive Version des DLP

geg: $\alpha, \beta \in G = (\mathcal{Z}_m^*, 0, +)$
ges: $e \in \mathcal{Z}^+$ mit $e \cdot \alpha = \beta$
Lösbar durch Division von α durch β

geg: $\alpha, \beta \in G = (E_{a,b}, \infty, +)$
ges: $e \in \mathcal{Z}^+$ mit $e \cdot \alpha = \beta$

Definition 49 (Addition auf E)

Sei $p \geq 3$ prim und seien $a, b \in \mathcal{Z}_p$ mit $4a^3 + 27p^2 \not\equiv_p 0$.
Dann ist

$$E = \{(x, y) \in \mathcal{Z}_p^2 | y^2 \equiv_p x^3 + ax + b\} \cup \{\infty\}$$

nicht singuläre elliptische Kurve über $\mathcal{Z}_p$.
Wir definieren eine Addition $+$ auf E wie folgt:

- ∞ ist neutrales Element, d.h. $\forall P \in E : P + \infty = \infty + P = P$
- $\forall P, Q \in E - \{\infty\}$ ist

$$P + Q = \begin{cases} \infty, & P = \overline{Q} \\ R, & sonst \end{cases}$$

wobei sich $R = (x_3, y_3)$ wie folgt aus $P = (x_1, y_1)$ und $Q = (x_2, y_2)$ berechnet:

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned}$$

wobei $\lambda = \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1}, & P \neq Q \\ (3x_1^2 + a)(2y_1)^{-1}, & P = Q \end{cases}$

Satz 50 *$(E, \infty, +)$ bildet eine abelsche Gruppe*

Beweis: ohne Beweis ■

Bemerkung 51 *$p = 11$, E definiert durch $y^2 = x^3 + x + 6$*
Zur Erinnerung: Im Fall $p \equiv_4 3$ lassen sich für $z \in QR_p$ die Wurzeln y durch $\pm z^{\frac{p+1}{4}}$ bestimmen.

x	0	1	2	3	4	5	6	7	8	9	10
$z = x^3 + x + 6$	6	8	5	3	8	4	8	4	9	7	4
$z \in QR_{11}$	-	-	x	x	-	x	-	x	x	-	x
y	-	-	4;7	5;6	-	2;9	-	2;9	3;8	-	2;9

$\sharp E = \|E\| = 13$
$\rightarrow (E, \infty, +)$ *ist zyklisch (sogar jedes Element $p \in E - \{\infty\}$ ist Erzeuger von E)*
$\rightarrow E$ *ist isomorph zu $\mathcal{Z}_{13}$; genauer* $(E, \infty, +) \approx (\mathcal{Z}_{13}, 0, +)$

Berechnung von $2g = (2, 7) + (2, 7)$*:*

$$\begin{aligned} \lambda &= (3 \cdot 2^2 + 1)(2 \cdot 7)^{-1} \bmod 11 \\ &= 2 \cdot 3^{-1} \\ &= 2 \cdot 4 = 8 \\ x_3 &= 8^2 - 2 - 2 \bmod 11 = 5 \\ y_3 &= 8(2 - 5) - 7 \bmod 11 = 2 \end{aligned}$$

$\Rightarrow 2g = (5, 2)$

Berechnung von $3g = 2g + g = (5,2) + (2,7)$*:*

$$\begin{aligned} \lambda &= (7-2)(2-5)^{-1} \bmod 11 \\ &= 5 \cdot (-3)^{-1} \\ &= 2 \\ x_3 &= 2^2 - 5 - 2 \bmod 11 = 8 \\ y_3 &= 2 \cdot (5-8) - 2 \bmod 11 = 3 \end{aligned}$$

$\Rightarrow 3g = (8,3)$

k	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*
k·g	*(2,7)*	*(5,2)*	*(8,3)*	*(10,2)*	*(3,6)*	*(7,9)*	*(7,2)*	*(3,5)*	*(10,9)*	*(8,8)*	*(5,9)*	*(2,4)*	∞

Satz 52 *(Hasse)*
Für die Anzahl $\sharp E$ *von Punkten einer elliptischen Kurve über* $\mathcal{Z}_p$ *(*$p \geq 3$ *prim) gilt* $p + 1 - 2\sqrt{p} \leq \sharp E \leq p + 1 + 2\sqrt{p}$

Beweis: (ohne Beweis)

■

PointCompress: $E - \{\infty\} \rightarrow \mathcal{Z}_p \times \mathcal{Z}_2$: $PointCompress(x,y) := (x, y \bmod 2)$

Algorithmus 53 POINTDECOMPRESS(x,i)

```
1  z ← x^3 + ax + b mod p
2  if z ∈ QR_p then
3    y ← √z mod p
4    if y ≢_2 i then
5      y ← p − y
6    output (x, y)else
7  output "error" end
```

Bemerkung 54 *Es gibt einen effektiven Algorithmus (von Schoof) mit Zeitkomplexität* $O((\log p)^8)$*, der* $\sharp E$ *bei Eingabe von* a, b *und* p *berechnet.*

Satz 55 *Sei* E *eine elliptische Kurve über* $\mathcal{Z}_p, p > 3$ *prim.*
Dann ist $(E, \infty, +)$ *isomorph zu* $\mathcal{Z}_{n_1} \times \mathcal{Z}_{n_2}$*, wobei* $n_1, n_2 \in IN^+$ *sind und* n_2 *Teiler von* n_1 *und von* $p - 1$ *ist.*

Bemerkung 56 *Zyklische (Unter)-Gruppe*
• *Wegen* $\sharp E = n_1 \cdot n_2$ *und da* n_2 *Teiler von* n_1 *ist, muss* E *im Fall, daß* $\sharp E$ *prim oder das Produkt von zwei oder verschiedenen Primzahlen ist, zyklisch sein (d.h.* $n_2 = 1$*)*
• *Im Fall* $n_2 > 1$ *hat* E *eine nicht-triviale zyklische Untergruppe, die zu* $\mathcal{Z}_{n_1}$ *isomorph ist und im DSA benutzt werden kann.*

ECDSA (Elliptic Curve DSA)

(im Jahr 2000 als FIPS 186-2 als Standard deklariert)

Definition 57 (ECDSA)

Sei p eine Primzahl oder eine Zweier-Potenz und sei E eine elliptische Kurve über $\mathcal{Z}_p$.
Sei $A \in E$ ein Punkt der Ordnung q (q prim), so daß das Diskrete-Logarithmus-Problem zur Basis A in E schwierig ist.

$X = \{0,1\}^*$
$Y = \mathcal{Z}_q^* \times \mathcal{Z}_q^*$

öffentlicher Verifikationsschlüssel: (p, q, E, A, B), wobei $B = m \cdot A$
geheimer Signierschlüssel: $(p, q, E, A, m), 0 \leq m \leq q-1$

$sig(\overline{k}, r, x) = (y, z)$, wobei

$$\begin{aligned} (u,v) &:= r \cdot A \\ y &:= u \bmod q \\ z &:= (SHA-1(x) + my) r^{-1} \bmod q \end{aligned}$$

$$ver(k,x,y,z) = \begin{cases} 1, & n \bmod q = y \\ 0, & sonst \end{cases} \quad \text{wobei}$$

$$\begin{aligned} (u,v) &:= eA + dB \\ e &:= SHA-1(x) z^{-1} \bmod q \\ d &:= yz^{-1} \bmod q \end{aligned}$$

Verifikation bei ECDSA:

$$\begin{aligned} (u,v) &= eA + dB \\ &= (x' z^{-1})A + (yz^{-1})mA \\ &= (x' + my) z^{-1} A \\ &= rA \text{ (da } (x' + my) z^{-1} \equiv_q r) \end{aligned}$$

Beispiel 58 *Signieren und Verifizieren:*
Sei E über $\mathcal{Z}_{11}$ definiert durch $y^2 = x^3 + x + 6$
Wir wählen $A = (2,7)$, $m = 7 \rightarrow p = 11, q = 13, B = 7A = (7,2)$

Annahme: Wollen Dokument x mit $SHA-1(x) = 4$

mit dem Signierschlüssel $\overline{k} = (p, q, E, A, m)$ *und der Zufallszahl* $r = 3$ *signieren.*

$$\begin{aligned} (u, v) &:= rA = 3 \cdot (2, 7) = (8, 3) \\ y &:= n \bmod q = 8, z = (4 + 7 \cdot 8)3^{-1} \bmod 13 = 7 \\ sig(\overline{k}, r, x) &= (8, 7) \end{aligned}$$

Verifikation von $(y, z) = (8, 7)$ *unter* $k = (p, q, E, A, B)$*:*

$$\begin{aligned} e &:= x'z^{-1} \bmod q = 4 \cdot 7^{-1} \bmod 13 = 4 \cdot 2 \bmod 13 = 8 \\ d &:= yz^{-1} \bmod q = 8 \cdot 2 \bmod 13 = 3 \\ (u, v) &:= eA + dB = 8 \cdot (2, 7) + 3 \cdot (7, 2) = (8, 3) \end{aligned}$$

$\rightsquigarrow u \bmod q = 8 = y$ ◁

Effi ziente Berechnung von Vielfachen von Punkten auf E

In $\mathcal{Z}_m^*$ berechnen wir Potenzen $a^e \bmod m$ durch 'wiederholtes Quadieren und Multiplizieren'.
Ähnlich können wir in einer elliptischen Kurve E die Vielfachen mP eines Punktes P durch 'wiederholtes Verdoppeln und Addieren' berechnen.
Da in E additiv Inverse sehr leicht zu berechnen sind, kann mP durch 'wiederholtes Verdoppeln, Addieren und Subtrahieren' noch effizienter berechnen werden.
Hierzu stellen wir m in NAF (non adjacent form) dar.

Definition 59 (Non adjacent form)

- $(c_{l-1}, \ldots, c_0) \in \{-1, 0, 1\}^l$ heißt SBR-Darstellung (signial binary representation) einer Zahl $c \in \mathcal{Z}$, falls

$$\sum_{i=0}^{l-1} c_i z^i = c$$

ist.
- Ist von je zwei benachbarten c_i's mindestens eines 0, so heißt $(c_{l-1}, \ldots, c_0)$ NAF-Darstellung von c.

Beispiel 60 *Sowohl* 01011 *als auch* $10 - 10 - 1$ *sind SBR-Darstellungen von* $c = 1 + 2 + 8 = 11 = -1 - 4 + 16$. ◁

Satz 61 *Jede Zahl* $c \in \mathcal{Z}$ *hat eine eindeutige NAF-Darstellung.*

Beweis: (siehe Übungen) ■

Berechnung einer NAF-Darstellung aus der Binärdarstellung:
Ersetzen Teilstrings der Form $01\ldots1$ von rechts beginnend durch den Teilstring $10\ldots01$.

Algorithmen zur Berechnung von Vielfachen von Punkten auf E:

Algorithmus 62 DOUBLEADD(P, $\underbrace{(c_{l-1}\ldots c_0)}_{Binaer-Darstellung}$)

```
1  Q ← ∞
2  for i := l − 1 to 0 do
3    Q ← 2 · Q
4    if c_i = 1 then
5      Q ← Q + P
6    end
7  end
8  output Q
```

Algorithmus 63 DOUBLEADDSUB(P, $\underbrace{(c_{l-1},\ldots,c_0)}_{SBK-Darstellung}$)

```
1  Q ← ∞
2  for i := l − 1 to 0 do
3    Q ← 2 · Q
4    if c_i = 1 then
5      Q ← P + Qelse
6    if c_i = −1 then
7      Q ← Q + (−P)
8    endend
9  end
10 output Q
```

Da eine l-Bitzahl im Durchschnitt $\frac{l}{2}$-Nullen in Binärdarstellung und $\frac{2l}{3}$-Nullen in NAF-Darstellung enthält, ist DoubleAddSub um 11 Prozent effizienter als DoubleAdd.

2.5 One-time Signatur (Lamport)

Definition 64 (One-time Signatur)

$X := \{0,1\}^n$; $Y = \mathcal{Y}^n$
Sei $f : \mathcal{Y} \to Z$ eine Einwegfunktion,
d.h. bei Eingabe $y \in Y$ kann $f(y)$ effizient (in polynomieller Zeit) berechnet werden, aber für gegebenes $z \in Z$ ist es 'schwer', ein y mit $f(y) = z$ zu finden.
Für $(i,b) \in \{1,\ldots,n\} \times \{0,1\}$ wähle zufällig $y_{i,b} \in Y$ und berechne $z_{i,b} = f(y_{i,b})$.

Signierschlüssel: $\overline{k} = (y_{i,b})_{i=1,\ldots,n;b=0,1}$
Verifikationsschlüssel: $k = (z_{i,b})_{i=1,\ldots,n;b=0,1}$

Für $x = x_1 \ldots x_n \in X$ ist dann

$$sig(\overline{k}, x) := y_{1,x_1} \ldots y_{n,x_n}$$

$$ver(k,x,y) := \begin{cases} 1, & f(y_i) = z_{i,x_i}; i = 1,\ldots,n \\ 0, & sonst \end{cases}$$

Beispiel 65 *Wir wählen als Einwegfunktion für $f : \mathcal{Z}_p^* \to \mathcal{Z}_p^*$ mit $f(x) := g^x \bmod p$, wobei g ein Erzeuger von $\mathcal{Z}_p^*$ ist.*

$p = 7879$, $g = 3$, $f(x) = 3^x \bmod 7879$
$n = 3$
$\overline{k} = (5831, 803, 4285, 735, 2467, 6449)$
$k = (2009, 4672, 268, 3810, 4721, 5731)$

$sig(\overline{k}, 110) = (803, 735, 2467)$
$ver(k, 110, (803, 735, 2467)) = ?$

Zu prüfen ist:

$i = 1 : f(y_1) = z_{1,x_1} \to f(803) = 4672 \to 3^{803} = 4672 \bmod 7879$ *?*
$i = 2 : f(y_2) = z_{2,x_2} \to f(735) = 3810 \to 3^{735} = 3810 \bmod 7879$ *?*
$i = 3 : f(y_3) = z_{3,x_3} \to f(2467) = 4721 \to 3^{2467} = 4721 \bmod 7879$ *?*

◁

Sei Lamport-Fälschung (k) ein Algorithmus, der bei Eingabe eines Verifikationsschlüssel k eine nicht-selektive Fälschung (x, y) mit $ver(k, x, y) = 1$ berechnet.
Sei $f : \mathcal{Y} \to Z$ eine Bijektion.
Betrachte folgenden probabilistischen Algorithmus:

Algorithmus 66 LAMPORT-URBILD(Z)

1 **Konstruiere zuf. Verifikationsschlüssel** $k = (z_{i,b})_{i=1,\ldots,n;b=0,1}$ mit $z_{i_0,b_0} = z$, wobei(i_0, b_0) zufällig aus $\{1, \ldots, n\} \times \{0, 1\}$ gewählt ist.
2 $(x_1, \ldots, x_n, y_1, \ldots, y_n) \leftarrow$ Lamport-Fälschung(k)
3 **if** $x_{i_0} = b_0$ **then**
4 **output** y_{i_0}**else**
5 **output** '?' **end**

Satz 67 *Unter den genannten Voraussetzungen gibt Lamport-Urbild(z) für ein zufällig aus Z gewähltes z mit Wahrscheinlichkeit $\frac{1}{2}$ ein Urbild y von z aus.*

Beweis: Im Fall $x_{i_0} = b_{i_0}$ gibt der Algorithmus Lamport-Urbild ein $y = y_{i_0}$ aus mit

$$f(y_{i_0}) = z_{i_0}, x_{i_0} = z_{i_0,b_0} = z$$

Es reicht zu zeigen:

$$Prob_{z \in_R Z}[Lamport - Urbild(z) = \text{'?'}] = \frac{1}{2}$$

Sei $\mathcal{S}$ die Menge aller möglichen Verifikationsschlüssel k und für $z \in Z$ sei $\mathcal{S}_z$ die Menge aller $k \in \mathcal{S}$, die z enthalten.
$\mathcal{T}_z$ bezeichne die Menge aller $k \in \mathcal{S}_z$, bei deren Wahl Lamport-Urbild(z) Erfolg hat.

Behauptung1:

Sei $t_z = \|\mathcal{T}_z\|, s_z = \|\mathcal{S}_z\|$ und $s = \|\mathcal{S}\|$.
Dann gilt:

$$\sum_{z \in Z} t_z = n \cdot s$$

Für jeden der s möglichen Verifikationsschlüssel $k \in \mathcal{S}$ findet Lamport-Fälschung genau n Urbilder, was der Gesamtzahl $\sum t_z$ aller von Lamport-Fälschung berechneten Urbilder entspricht.

Behauptung2:

$$s_z \cdot \|z\| = 2ns$$

Jeder der s möglichen Verifikationsschlüssel $k \in \mathcal{S}$ enthält 2n Elemente von Z, also ist

$$2ns = \sum_{z \in Z} s_z = s_z \sum_{z \in Z} 1 = s_z \cdot \|Z\|$$

Sei nun p_z die Erfolgswahrscheinlichkeit von Lamport-Urbild(z), d.h.

$$p_z = \frac{\|\mathcal{T}_z\|}{\|\mathcal{S}_z\|} = \frac{t_z}{s_z}.$$

Dann gilt für die durchschnittliche Erfolgswahrscheinlichkeit

$$\overline{p} = \frac{\sum p_z}{\|Z\|} = \frac{1}{s_z \cdot \|Z\|} \cdot \sum t_z = \frac{ns}{2ns} = \frac{1}{2}$$

■

2.6 Full Domain Hash (FDH) Signaturen

Sei $\mathcal{F} = \{f_k | k \in \mathcal{K}\}$ eine Familie von Falltür-Permutationen auf $\{0,1\}^n$, d.h. für jedes $k \in \mathcal{K}$ gilt:

- f_k ist Einweg-Permutation auf $\{0,1\}^n$
- es existiert ein $\overline{k} \in \mathcal{K}$ mit $f_{\overline{k}}(f_k(x)) = x \forall x \in \{0,1\}^n$

Weiter sei $G : \{0,1\}^* \to \{0,1\}^n$ eine Zufallsfunktion,
d.h.

$$Prob[G(x) = y] = 2^{-n} \; \forall x \in \{0,1\}^* \text{ und } y \in \{0,1\}^n.$$

G modelliert eine Hashfunktion $H : \{0,1\}^* \to \{0,1\}^n$ mit optimalen kryptographischen Eigenschaften (vgl. Zufalls-Orakel-Modell (ZOM)), deren Wertebereich den gesamten Definitionsbereich der f_k ausfüllt (full domain hash).

In der Praxis wird für G eine konkrete Hashfunktion (etwa $SHA-1$) eingesetzt.

FDH-Signaturverfahren (basierend auf $\mathcal{F}$ und G)

Definition 68 ()

$X = \{0,1\}^*$
$Y = \{0,1\}^n$
Signierschlüssel: $\overline{k}$
Verifikationsschlüssel: k

$$sig(\overline{k}, x) := f_{\overline{k}}(G(x))$$

$$ver(k, x, y) := \begin{cases} 1, & f_k(y) = G(x) \\ 0, & sonst \end{cases}$$

Z.B. beruht das RSA-Signatur-Verfahren in Verbindung mit einer Hash-Funktion auf dem FDH-Prinzip.
Ein Problem hierbei ist allerdings, daß der Wertebereich von in der Praxis verwendeten Hashfunktionen die Menge $\{0,1\}^{160}$ ist und für die RSA-Falltür-Permutationen ein Defintionsbereich von $\{0,1\}^n$ mit $n \approx 1024$ zu wählen ist, um eine ausreichend große Sicherheit zu erreichen. In der Praxis behilft man sich damit, daß man die 160-Bit-Hashwerte durch eine deterministische Padding-Funktion auf 1024-Bit aufbläht.

Sicherheitsanalyse der FDH-Signatur im ZOM

Sei *FDH-Fälschung* ein probabilistischer Algorithmus, der bei Eingabe des öffentlichen Verifikationsschlüssels k mit Wahrscheinlichkeit ϵ eine nicht-selektive Fälschung (x,y) mit $ver(x,y) = 1$ ausgibt und sei q die Anzahl der verschiedenen Orakelfragen $x_1, \ldots, x_q$ von *FDH-Fälschung* an G.
Wir nehmen an, daß $\epsilon > 2^{-n}$ ist, da für beliebiges Dokument $x \in \{0,1\}^*$ ein zufällig gewähltes $y \in \{0,1\}^n$ mit Wahrscheinlichkeit 2^{-n} eine gültige Signatur liefert.
Betrachte folgenden Invertierungsalgorithmus für f_k bei Eingabe eines beliebigen Verifikationsschlüssels $k \in \mathcal{K}$ und eines zufällig gewählten $z_0 \in \{0,1\}^n$.

Algorithmus 69 $FDH - Invert(k, z_0)$

1 **wähle zufällig** $i_0 \in (1, \ldots, q)$
2 **simuliere *FDH-Fälschung(k),*** wobei die i-te Orakel-Frage $x_i, 1 \leq i \leq q$, im Fall $i = i_0$ durch z_0 und sonst durch ein zufällig gewähltes $z \in \{0,1\}^n$ beantwortet wird.
3 **if** *FDH-Fälschung*$(k) = (x,y)$ **then**
4 **if** $f_k(y) = z_0$ **then**
5 **output** y**else**
6 **output** ? **end else**
7 **output** ? **end**

Da die Eingabe z_0 zufällig gewählt wird, erhält *FDH-Fälschung* als Antwort auf seine Orakel-Fragen $x_1, \ldots, x_q$ zufällig gewählte Strings z, was dem ZOM entspricht.

$$\rightsquigarrow \underbrace{Pr[\text{ FDH-Fälschung } (k) = (x,y) \text{ mit } ver(k,x,y) = 1]}_{Pr[\text{ FDH-Fälschung hat Erfolg}]} = \epsilon$$

Weiter gilt:

$$\begin{aligned} \epsilon &= Pr[\text{FDH-Fälschung hat Erfolg} \wedge x \in \{x_1, \ldots, x_n\}] \\ &+ \underbrace{Pr[\text{FDH-Fälschung hat Erfolg} \wedge x \notin \{x_1, \ldots, x_n\}]}_{\leq 2^{-n}, \text{ da über } G(x) \text{ in diesem Fall nichts bekannt ist und } G(x) \text{ jeden Wert in } \{0,1\}^n \text{ mit gleicher Wahrscheinlichkeit } 2^{-n} \text{ annimmt.}} \end{aligned}$$

$$\leadsto Pr[\textit{FDH-Fälschung}\text{hat Erfolg} \wedge x \in \{x_1, \ldots, x_n\}] \geq \epsilon - 2^{-n}$$

Zudem gilt:

$$\begin{aligned} Pr[\textit{FDH-Invert hat Erfolg}] &\geq Pr[\textit{FDH-Fälschung hat Erfolg } \wedge x = x_{i_0}] \\ &= \frac{Pr[\textit{FDH-Fälschung hat Erfolg } \wedge x \in \{x_1, \ldots, x_q\}]}{q} \\ &\quad \textit{(Da unter der Bedingung, daß } x \in \{x_1, \ldots, x_q\} \textit{ ist,} \\ &\quad \textit{die Wahrscheinlichkeit für } x = x_{i_0} \textit{ gleich } \frac{1}{q} \textit{ ist.)} \\ &\geq \frac{\epsilon - 2^{-n}}{q} \end{aligned}$$

Satz 70 *Falls FDH-Fälschung bei Eingabe k nach genau q Fragen an G eine gültige Fälschung (x, y) mit Wahrscheinlichkeit $\epsilon > 2^{-n}$ ausgibt, findet $FDH - Invert$ bei Eingabe von k und einem zufällig gewählten String $z_0 \in \{0,1\}^n$ mit Wahrscheinlichkeit*

$$\epsilon' \geq \frac{\epsilon - 2^{-n}}{q}$$

ein Urbild y von z_0 für die Funktion f_k.

2.7 Verbindliche Signaturen (undeniable signatures)

In manchen Fällen ist es für den Unterzeichner eines Dokumentes nicht wünschenswert, daß jeder die von ihm geleistete Unterschrift verifizieren kann.

Beispiel 71 *Eine Software-Firma möchte sicherstellen, daß nur rechtmässige Käufer (keine SW-Piraten) ihre Signatur, die u.a. Virus-Freiheit garantiert, verifizieren können.*
Lösung:
Zur Verifikation wird die Kooperation des Unterzeichners Alice benötigt.

Wie kann man verhindern, daß sich Alice absichtlich unkooperativ verhält, nur damit eine von ihr geleistete Unterschrift als falsch eingestuft wird?
Lösung:
Es gibt ein Ableugnungsprotokoll (disavowal protocol) mit dem Alice gefälschte Unterschriften als solche entlarven kann. Scheitert auch dieses Protokoll so liegt der Verdacht nahe, daß die fragliche Unterschrift doch von Alice stammt. ◁

Das Chaum-van Antwerpen Signaturverfahren

Definition 72 (Chaum-van Antwerpen Signaturverfahren)

Sei $p = 2q + 1$ mit p prim, so daß auch q prim und das Diskrete Logarithmus-Problem in $\mathcal{Z}_p$ hart ist.
Sei $\alpha \in \mathcal{Z}_p^*$ ein Element der Ordnung q und sei $G = \{\alpha^a | a \in \mathcal{Z}_q\}$, die von α in $\mathcal{Z}_p^*$ erzeugte Untergruppe.

X=Y=G
Signierschlüssel: $\overline{k} = (P, \alpha, a), a \in \mathcal{Z}_q^*$
Verifikationsschlüssel:$k = (P, \alpha, \beta), \beta = \alpha^a \bmod p$
Signieralgorithmus: $sig(\overline{k}, x) = x^a \bmod p$

Verifikationsprotokoll:
Bob verifiziert eine von Alice geleistete Unterschrift $y \in G$ für ein Dokument $x \in G$ wie folgt:

1. Bob wählt zufällig $e_1, e_2 \in \mathcal{Z}_q$ und und sendet $c = y^{e_1}\beta^{e_2} \bmod p$ an Alice.
2. Alice sendet $d = c^{a^{-1} \bmod q} \bmod p$ zurück an Bob.
3. Bob akzeptiert y als echt, falls $d \equiv_p x^{e_1}\alpha^{e_2}$ ist.

Bemerkung 73 *Die Wahl der Primzahl p von der Form $p = 2q + 1$ mit q prim dient dazu, folgende beiden Ziele zu erreichen:*

- *$\|G\|$ ist prim (erlaubt Berechnung von $a^{-1} \bmod \|G\|$)*
- *$\|G\|$ ist für vorgegebenes p möglichst groß.*

Es ist leicht zu sehen, daß Bob eine echte Signatur y akzeptiert, falls Alice kooperiert: Wegen

$$\beta \equiv_p \alpha^a$$

folgt

$$\beta^{a^{-1}} \equiv_p \alpha^{a \cdot a^{-1}} \equiv_p \alpha$$

und wegen

$$y \equiv_p x^a$$

folgt

$$y^{a^{-1}} \equiv_p x^{a \cdot a^{-1}} \equiv_p x.$$

Somit

$$d = c^{a^{-1}} = (y^{e_1}\beta^{e_2})^{a^{-1}} = y^{a^{-1}e_1}\beta^{a^{-1}e_2} = x^{e_1}\alpha^{e_2}.$$

Beispiel 74 $p = 467 = 2 \cdot 233 + 1, q = 233$
Da $g = 2$ ein Erzeuger von $\mathcal{Z}_p^*$ ist, hat $\alpha = g^2 = 4$ die gewünschte Ordnung $q = \frac{p-1}{2}$.
(Da α QR_p ist, ist $G = QR_p$)
Signierschlüssel: $\overline{k} = (p, \alpha, a) = (467, 4, 101)$
Verifikationsschlüssel: $k = (p, \alpha, \beta) = (467, 4, 449)$
Signaturberechnung: für $x = 119 \in G$

$$sig(\overline{k}, x) = x^a \bmod p = 119^{101} \bmod 467 = 129 = y$$

Verifikation: von $y = 129$ *für* $x = 119$ *unter* k*:*

1. *Bob wählt* $e_1, e_2 \in \mathcal{Z}_q$ *(*$e_1 = 38, e_2 = 397 = 164$*) und sendet* $c = y^{e_1}\beta^{e_2} \bmod p = 129^{38}449^{164} \bmod 467 = 13$ *an Alice.*
2. *Alice sendet* $d = c^{a^{-1} \bmod q} \bmod p = 9$ *an Bob zurück.*
3. *Bob akzeptiert, da* $d = x^{e_1}\alpha^{e_2} = 119^{38}4^{164} \bmod 467 = 9$ *gilt.*

◁

Satz 75 *Im Fall* $y \not\equiv_p x^a$ *akzeptiert Bob* y *mit Wahrscheinlichkeit* $\frac{1}{q}$

Beweis: Da zu $y, \beta, c \in G$ und zu $e_1 \in \mathcal{Z}_q$ genau ein $e_2 \in \mathcal{Z}_q$ mit $c \equiv_p y^{e_1}\beta^{e_2}$ existiert, führen je q Paare $(e_1, e_2) \in \mathcal{Z}_q \times \mathcal{Z}_q$ auf dasselbe c.
Beh:
Im Fall $y \not\equiv_p x^a$ erfüllt für jedes $d \in G$ genau ein Paar $(e_1, e_2) \in \mathcal{Z}_q \times \mathcal{Z}_q$ die Kongruenzen $\begin{cases} c \equiv_p y^{e_1}\beta^{e_2} \\ d \equiv_p x^{e_1}\alpha^{e_2} \end{cases}$ (*)

Beweis: Da $c, d, x, y \in G$, existieren eindeutig bestimmte Exponenten $i, j, k, l \in \mathcal{Z}_q$ mit $c \equiv_p \alpha^i, \ldots, y \equiv_p \alpha^l$.
(*) ist äquivalent zu $\begin{cases} \alpha^i \equiv_p \alpha^{le_1} \cdot \alpha^{ae_2} \\ \alpha^j \equiv_p \alpha^{ke_1} \cdot \alpha^{e_2} \end{cases}$ (**) bzw. zu $\begin{cases} i \equiv_q le_1 + ae_2 \\ j \equiv_q ke_1 + e_2 \end{cases}$ (***).
Da $y \not\equiv_p x^a$ ist, folgt

$$l \not\equiv_q ka \Rightarrow det \begin{pmatrix} l & a \\ k & 1 \end{pmatrix} \not\equiv_q 0,$$

weshalb (***) und damit auch (*) eindeutig lösbar ist.

Daher gilt unabhängig davon welches $d \in G$ Alice an Bob zurückschickt,

$$Pr[\text{Bob akzeptiert}] = \frac{\sharp \text{ günstige Fälle}}{\sharp \text{ mögliche Fälle}} = \frac{1}{q}.$$

■

■

Algorithmus 76 ABLEUGNUNGSPROTOKOLL

1 **Bob wählt zufällig** $e_1, e_2 \in \mathcal{Z}_q$ **und sendet** $c = y^{e_1}\beta^{e_2} \bmod p$ **an Alice.**

2 **Alice sendet** $d = c^{a^{-1}} \bmod p$ **zurück.**

3 **Bob testet, ob** $d \not\equiv_p x^{e_1}\alpha^{e_2}$ **ist.**

4 **Bob wählt zufällig** $f_1, f_2 \in \mathcal{Z}_q$ **und sendet** $C = y^{f_1}\beta^{f_2} \bmod p$ **an Alice.**

5 **Alice sendet** $D = C^{a^{-1}} \bmod p$ **zurück.**

6 **Bob testet, ob** $D \not\equiv_p x^{f_1}\alpha^{f_2}$ **ist.**

7 **Bob erkennt** y **als gefälscht an, falls mindestens einer der Tests in Schritt 3) oder 6) erfolgreich war und** $(d\alpha^{-e_2})^{f_1} \equiv_p (D\alpha^{-f_2})^{e_1}$ **gilt.**

Bei den Schritten 1.-3. und 4.-6. handelt es sich jeweils um eine fehlgeschlagene Verifikation der Unterschrift y (bei positivem Testausgang).
In Schritt 7. führt Bob einen Konsistenztest aus, um sich davon zu überzeugen, daß sich Alice bei der Berechnung der Zahlen d und $\tilde{d}$ an das Protokoll gehalten hat.

Beispiel 77 $p = 467, q = 233, \alpha = 4, a = 101, \beta = 449$
Ann: Dokument $x = 286$ *ist mit der Alice zugeschriebenen Signatur* $y = 81$ *unterschrieben und Alice möchte Bob überzeugen, daß* y *gefälscht ist.*

1. *Bob wählt* $e_1 = 45, e_2 = 237$ *und sendet* $c = 305$ *an Alice.*
2. *Alice antwortet mit* $d = c^{a^{-1}} = 109$
3. *Bob verifiziert, daß* $286^{45}4^{237} \equiv_p 149 \not\equiv_p 109$ *gilt.*
4. *Bob wählt* $f_1 = 125, f_2 = 9$ *und sendet* $C = 72$ *an Alice.*
5. *Alice antwortet mit* $D = C^{a^{-1}} = 68$
6. *Bob verifiziert, daß* $286^{125}4^{9} \equiv_p 25 \not\equiv_p 109$ *gilt.*
7. *Bon erkennt* y *also gefälscht an, da* $(109 \cdot 4^{-237})^{125} \equiv_p 188 \equiv_p (68 \cdot 4^{-9})^{45}$ *ist, also die Konsistenzbedingung erfüllt ist.*

◁

Noch zu zeigen:
• Alice kann Bob mit hoher Wahrscheinlichkeit von der Falschheit

einer Signatur $y \not\equiv_p x^a$ überzeugen.

• Alice kann Bob nur mit geringer Wahrscheinlichkeit von der Falschheit einer echten Signatur $y \equiv_p x^a$ überzeugen.

Satz 78 *Im Fall $y \not\equiv_p x^a$ erkennt Bob y mit Wahrscheinlichkeit $1 - \frac{1}{q^2}$ als gefälscht an, falls sich beide an das Protokoll halten.*

Beweis: Nach vorigem Satz beträgt die Wahrscheinlichkeit, daß beide Tests in Schritt 3. und 6. fehlschlagen genau $\frac{1}{q^2}$.
Wegen

$$d \equiv_p c^{a^{-1}}, c \equiv_p y^{e_1}\beta^{e_2}, \beta \equiv_p \alpha^a$$

folgt

$$\begin{aligned}(d\alpha^{-e_2})^{f_1} &\equiv_p ((y^{e_1}\beta^{e_2})^{a^{-1}}\alpha^{-e_2})^{f_1} \\ &\equiv_p y^{e_1 a^{-1} f_1}\beta^{e_2 a^{-1} f_1}\alpha^{-e_2 f_1} \\ &\equiv_p y^{e_1 a^{-1} f_1}\alpha^{e_2 f_1}\alpha^{-e_2 f_1} \\ &\equiv_p y^{e_1 a^{-1} f_1}\end{aligned}$$

Analog ergibt sich aus

$$D \equiv_p C^{a^{-1}}, C \equiv_p y^{f_1}\beta^{f_2}, \beta \equiv_p \alpha^a$$

$$\begin{aligned}(D\alpha^{-f_2})^{e_1} &\equiv_p ((y^{f_1}\beta^{f_2})^{a^{-1}}\alpha^{-f_2})^{e_1} \\ &\equiv_p y^{f_1 a^{-1} e_1} \\ &\equiv_p (d\alpha^{-e_2})^{f_1}\end{aligned}$$

d.h. die Konsistenzbedingung wird mit Wahrscheinlichkeit 1 erfüllt.

■

Satz 79 *Im Fall $y \equiv_p x^a$ erkennt Bob y mit Wahrscheinlichkeit $\leq \frac{1}{q}$ als gefälscht an, auch wenn sich Alice nicht an das Protokoll hält.*

Beweis: Bob erkennt y nur dann als gefälscht an, wenn

$$(d \not\equiv_p x^{e_1}\alpha^{e_2} \text{ oder } D \not\equiv_p x^{f_1}\alpha^{f_2}) \text{ und } (d\alpha^{-e_2})^{f_1} \equiv_p (D\alpha^{-f_2})^{e_1}$$

gilt.

1.Fall: (2.Fall symmetrisch)

$$d \not\equiv_p x^{e_1}\alpha^{e_2} \text{ (*)}$$

Seien $e_1, e_2 \in \mathcal{Z}_q$ gewählt ,mit (*)
Durch C wird jedem f_1 genau ein f_2 mit $C \equiv_p y^{f_1}\beta^{f_2}$ zugeordnet.

Beh.: Für jedes $D \in G$

existiert genau ein Paar (f_1, f_2) mit $\begin{cases} C \equiv_p y^{f_1}\beta^{f_2} \\ \underbrace{(d\alpha^{-e_2})^{f_1}}_{u} \equiv_p (D\alpha^{-f_2})^{e_1} \end{cases}$

Beweis: Zuordnung der Exponenten zur Basis α:

β	x	y	C	D	$u = d\alpha^{-e2}$
a	k	ka	i	j	$l \not\equiv_q e_1 k$

Daraus folgt:

$$\begin{aligned} i &\equiv_q ka f_1 + a f_2 \\ l f_1 &\equiv_q j e_1 - f_2 e_1 \Rightarrow j e_1 \equiv_q l f_1 + e_1 f_2 \end{aligned}$$

Dieses Kongruenzgleichungssystem führt auf die Matrix

$$A = \begin{pmatrix} ka & a \\ l & e_1 \end{pmatrix}$$

mit

$$\det A = kae_1 - al = \underbrace{a}_{\not\equiv_q 0}\underbrace{(ke_1 - l)}_{\not\equiv_q 0} \not\equiv_q 0.$$

■

■

2.8 Fail-Stop-Signaturen

Diese Signaturen erlauben dem Unterschreiber Alice im Fall, daß ihr Signierschlüssel $\overline{k}$ geknackt wird (“fail”), dies zu beweisen und damit alle von ihr mit $\overline{k}$ geleisteten Unterschriften zu widerrufen (“stop”).

Genauer:

> Alice kann mit hoher Wahrscheinlichkeit beweisen, daß eine von einem Gegner erzeugte gültige Signatur y für ein Dokument x nicht von ihr stammt.

Das van Heyst-Pedersen Signaturverfahren

Definition 80 (van Heyst-Pedersen-Signaturverfahren)

Sei $q = 2p + 1$ prim, q prim und sei $\alpha \in \mathcal{Z}_q^*, \beta = \alpha^{a_0} \bmod p$.
$G = \{\alpha^k | k \in \mathcal{Z}_q\}$, alles wie bei der Chaum-van Antwerpen Signatur.
p, q, α, β werden von einer vertrauenswürdigen Instanz generiert und bekannt gegeben, a_0 wird jedoch vor allen Teilnehmern geheim gehalten.

$X = \mathcal{Z}_\mathrm{q}$
$Y = \mathcal{Z}_\mathrm{q} \times \mathcal{Z}_\mathrm{q}$
Signaturschlüssel: $\overline{k} := (a_1, a_2.b_1, b_2) \in \mathcal{Z}_q^4$
Verifikationsschlüssel: $k := (\gamma_1, \gamma_2) = (\alpha^{a_1}\beta^{a_2}, \alpha^{b_1}\beta^{b_2}) \in G^2$
Signieralgorithmus:

$$sig(\overline{k}, x) = (y_1, y_2) = (a_1 + xb_1 \bmod q, a_2 + xb_2 \bmod q)$$

Verifikationsalgorithmus:

$$ver(k, x, y_1, y_2) = \begin{cases} 1 & \gamma_1\gamma_2^x \equiv_p \alpha^{y_1}\beta^{y_2} \\ 0 & sonst \end{cases}$$

Beh: Im Fall: $sig(\overline{k}, x) = (y_1, y_2)$ gilt $ver(k, x, y_1, y_2) = 1$, da

$$\begin{aligned} \gamma_1\gamma_2^x &= \alpha^{a_1}\beta^{a_2}(\alpha^{b_1}\beta^{b_2})^x \\ &= \alpha^{a_1+xb_1}\beta^{a_2+xb_2} \\ &= \alpha^{y_1}\beta^{y_2} \end{aligned}$$

Sei S die Menge aller Paare $(\overline{k}, k) \in \mathcal{Z}_q^4 \times G^2$ mit $\overline{k} = (a_1, a_2, b_1, b_2)$ und $k = (\alpha^{a_1}\beta^{a_2}, \alpha^{b_1}\beta^{b_2})$ für $a_1, a_2, b_1, b_2 \in \mathcal{Z}_q$.
Weiter sei $S(k) = \{\overline{k} | (\overline{k}, k) \in S\}$.

Lemma 81 $\|S(k)\| = q^2$

Lemma 82 $\overline{k}, \overline{k'} \in S(k), y = sig(\overline{k}, x), y' = sig(\overline{k'}, x)$
$\Rightarrow ver(k, x, y) = ver(k, x, y') = 1$

Lemma 83 *Sei* $= (y_1, y_2) = sig(\overline{k}, x)$ *und sei* $\overline{k} \in S(k)$.
Dann gilt

$$S(k, x, y) :=\mid \{\overline{k'} \in S(k) | sig(\overline{k'}, x) = y\} \mid = q$$

Beweis: Sei $k = (\gamma_1, \gamma_2)$. Dann ist $\overline{k'} = (a_1, a_2, b_1, b_2) \in S(k, x, y)$

$$\Leftrightarrow (*) \left\{ \begin{array}{c} \left.\begin{array}{c} \gamma_1 \equiv_p \alpha^{a_1}\beta^{a_2} \\ \gamma_2 \equiv_p \alpha^{b_1}\beta^{b_2} \end{array}\right\} \overline{k'} \in S(k) \\ \left.\begin{array}{c} y_1 \equiv_q a_1 + xb_1 \\ y_2 \equiv_q a_2 + xb_2 \end{array}\right\} sig(\overline{k'}, x) = y \end{array} \right.$$

Seien $c_1, c_2 \in \mathcal{Z}_q$ eindeutig bestimmte Exponenten mit

$$\gamma_1 \equiv_p \alpha^{c_1} \text{ und } \gamma_2 \equiv_p \alpha^{c_2}$$

Dann ist (*) äquivalent zu $\left.\begin{array}{c} c_1 \equiv_q a_1 + a_0 a_2 \\ c_2 \equiv_q b_1 + a_0 b_2 \\ y_1 \equiv_q a_1 + xb_1 \\ y_2 \equiv_q a_2 + xb_2 \end{array}\right\} (**)$

oder in Matrixform $\underbrace{\begin{pmatrix} 1 & a_0 & 0 & 0 \\ 0 & 0 & 1 & a_0 \\ 1 & 0 & x & 0 \\ 0 & 1 & 0 & x \end{pmatrix}}_{A=\begin{pmatrix} r_1 \\ \vdots \\ r_4 \end{pmatrix}} \begin{pmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ y_1 \\ y_2 \end{pmatrix}$

Es genügt zu zeigen: $rang(A) = 3$

$rang(A) \geq 3$, da r_2, r_3, r_4 linear unabhängig
$rang(A) \leq 3$, da $r_1 = r_3 + a_0 r_4 - x r_2$

■

Lemma 84 *Sei $\overline{k} \in S(k)$ und gelte $sig(\overline{k}, x) = y$.*
Weiter sei (x', y') ein Paar mit $x' \neq x$ und $ver(k, x', y') = 1$.
Dann ist

$$S(k, x, y, x', y') = \|\{\overline{k} \in S(k) | sig(\overline{k'}, x) = y; sig(\overline{k'}, x') = y'\} = 1$$

Beweis: Die Bedingung $\overline{k'} = (a_1, a_2, b_1, b_2) \in S(k, x, y, x', y')$ ist äquivalent zu

$$\begin{pmatrix} 1 & a_0 & 0 & 0 \\ 0 & 0 & 1 & a_0 \\ 1 & 0 & x & 0 \\ 0 & 1 & 0 & x \\ 1 & 0 & x' & 0 \\ 0 & 1 & 0 & x' \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ y_1 \\ y_2 \\ y'_1 \\ y'_2 \end{pmatrix} (*)$$

wobei $y' = (y'_1, y'_2), y = (y_1, y_2), \gamma_1 \equiv_p \alpha^{c_1}, \gamma_2 \equiv_p \alpha^{c_2}$ ist.

Behauptung:

$rang(A) = 4$, da $r_3, \ldots, r_6$ linear unabhängig

z.z:

$$\sum_{i=3}^{6} l_i r_i = \vec{0} \Rightarrow l_3 = \ldots = l_6 = 0$$

$$l_3 + l_5 = 0 \rightsquigarrow l_3 = -l_5 \rightsquigarrow l_5 = 0$$

$$x l_3 + x' l_5 = 0 \rightsquigarrow l_3 \underbrace{(x - x')}_{\neq 0} = 0 \rightsquigarrow l_3 = 0$$

Bemerkung 85 *Es handelt sich also um ein One-time-Signaturverfahren, da sich $\overline{k}$ bei Kenntnis zweier Signaturen y, y' für zwei Dokumente x, x' leicht bestimmen läßt.*

noch zu zeigen:

(*) ist lösbar.

Da $ver(k, x', y_1', y_2') = 1$, folgt $\gamma_1 \gamma_2^{x'} \equiv_p \alpha^{y_1'} \beta^{y_2'}$

$$\begin{aligned} \Rightarrow c_1 + x' c_2 &\equiv_q y_1' + a_0 y_2' \\ \Rightarrow y_1' &\equiv c_1 + x' c_2 - a_0 y_2' \end{aligned}$$

Analog folgt aus $ver(k, x, y) = 1$

$$y_1 \equiv_q c_1 + x c_2 - a_0 y_2$$

Wegen

$$r_1 + x' r_2 - a_0 r_6 = r_5$$

und

$$r_1 + x r_2 - a_0 r_4 = r_3$$

folgt das jede Lösung von $\begin{pmatrix} r_1 \\ r_2 \\ r_4 \\ r_6 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ y_2 \\ y_2' \end{pmatrix}$ auch Lösung von (*) ist. ■

Korollar 86 *Sei $x \neq x', ver(k, x', y') = 1$.*
Dann ist

$$Prob_{\overline{k'} \in_R \mathcal{Z}_q^4}[\underbrace{sig(\overline{k'}, x') = \underbrace{y'}_{sig(\overline{k}, x)}}_{A} \mid \underbrace{\underbrace{sig(\overline{k'}, x) = y, \overline{k'} \in S(k)}_{\overline{k'} \in S(k,x,y)}}_{B}] = \frac{1}{q}$$

d.h. für jedes von einem Gegner , bei Kenntnis von $k, x, y = sig(\overline{k}, x)$ generierte Paar $(x', y') \in X \times Y$ mit $x \neq x'$ und $ver(k, x', y') = 1$ ist die Wahrscheinlichkeit dafür das der 'richtige' Schlüssel $\overline{k}$ auch $sig(\overline{k}, x') = y'$ liefert, $\frac{1}{q}$.

Beweis: Klar, da $Prob[A|B] = \frac{Prob[A \wedge B]}{Prob[B]} = \frac{\|A \wedge B\|}{\|B\|} = \frac{1}{q}$. ■

Korollar 87 *Sei* $S(k, x, y) = \{\overline{k_1}, \ldots, \overline{k_q}\}$.
Dann gilt $\|\{sig(\overline{k_i}, x') | i = 1, \ldots, q\}\| = q$ *falls* $x' \neq x$ *ist.*

Beweis: Klar, da im Fall $sig(\overline{k_i}, x') = sig(\overline{k_j}, x') = y'$ und $i \neq j$ mindestens zwei Schlüssel in $S(k, x, y, x', y')$ existieren würden.⇒ Widerspruch zum Lemma. ■

Frage: Wie funktioniert der *Fail-Stop-Mechanismus*?

> D.h. wie kann Alice bei Vorlage eines Paares (x', y'') mit $ver(k, x', y'') = 1$ und $y'' \neq sig(\overline{k}, x') = y'$ beweisen, dass y'' nicht von ihr erzeugt wurde?

Antwort: Sie benutzt (k, x', y'') um a_0 zu berechnen.

> Wegen
> $$ver(k, x', y'') = 1 = ver(k, x', y')$$
> folgt
> $$\begin{aligned} \alpha^{y_1'}\beta^{y_2'} \equiv_p \gamma_1\gamma_2^{x'} &\equiv_p \alpha^{y_1''}\beta^{y_2''} \\ \Rightarrow y_1' + a_0 y_2' &\equiv_q y_1'' + a_0 y_2'' \\ \Rightarrow a_0 &\equiv_q \frac{y_1'' - y_1'}{y_2' - y_2''} \end{aligned}$$

Beispiel 88 *(zur van Heyst-Pedersen-Fail-Stop-Signatur)*
$p = 2q + 1, p = 3467 = 2 \cdot \underbrace{1733}_{q} + 1$
$\alpha \in \mathcal{Z}_p^*$ *mit* $ord_p(\alpha) = q, \alpha = 4$
$a_0 \in \mathcal{Z}_q^*, a_0 = 1567$
$\rightsquigarrow \beta = \alpha^{a_0} = 4^{1567} = 514$

Die vertrauenswürdige Instanz (TTP,trusted third party) gibt p, q, α, β *bekannt und hält* a_0 *geheim.*

Angenommen Alice wählt

$$\overline{k} = (\underbrace{888}_{a_1}, \underbrace{1024}_{a_2}, \underbrace{786}_{b_1}, \underbrace{999}_{b_2})$$

so berechnet sich k *zu*

$$k = (\gamma_1, \gamma_2), \text{ wobei}$$

$$\gamma_1 = \alpha^{a_1}\beta^{a_2} = 4^{888}514^{1024} = 3405$$

$$\gamma_2 = \alpha^{b_1}\beta^{b_2} = 4^{786}514^{999} = 2281$$

Wird nun Alice mit dem Paar $(x', y'') = (x', y_1'', y_2'') = (3383, 822, 55)$ *konfrontiert, das wegen*

$$\begin{aligned} \gamma_1\gamma_2^{x'} &= 3405 \cdot 2281^{3383} \equiv_p 2282 \\ \text{und } \alpha^{y_1''}\beta^{y_2''} &= 4^{822}515^{55} \equiv_p 2282 \end{aligned}$$

die Verifikationsbedingung $ver(k, x', y'') = 1$ *erfüllt, so berechnet Alice zunächst*

$$sig(k, x') = y' = (y_1', y_2') \text{ mit}$$

$$\begin{aligned} y_1' &= a_1 + x'b_1 \bmod q = 888 + 3383 \cdot 786 \equiv_p 1504 \\ y_2' &= a_2 + x'b_2 \bmod q = 1024 + 3383 \cdot 999 \equiv_p 1291 \end{aligned}$$

um sich zu vergewissern, daß $y' \neq y''$ *ist.*
Hieraus erhält sie dann a_0 *zu*

$$a_0 = \frac{y_1' - y_1''}{y_2' - y_2''} \bmod q = \frac{1504 - 822}{55 - 1291} \equiv_p 1567$$

◁