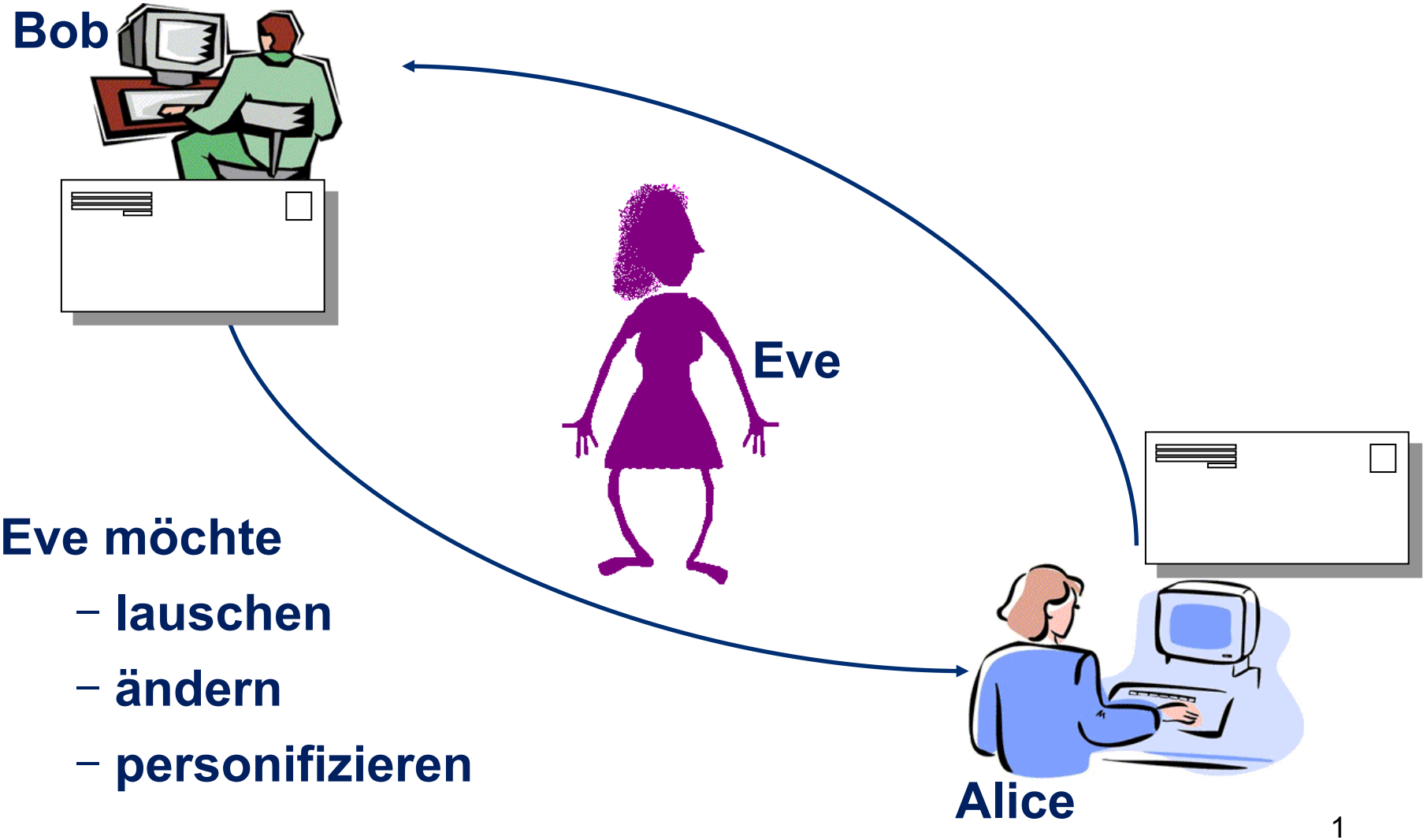
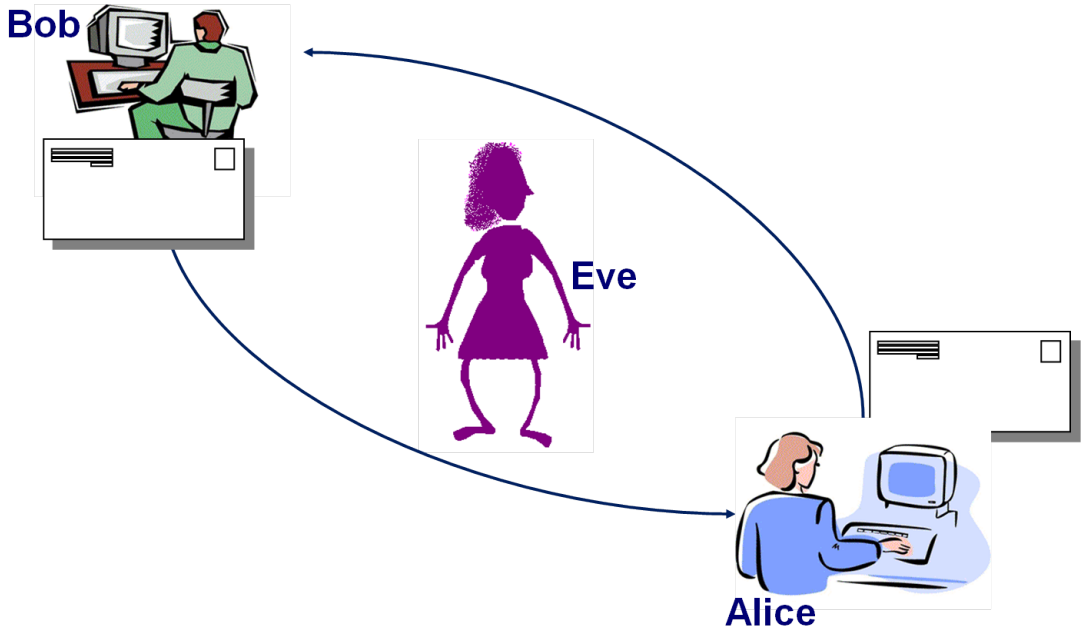
VII. Hashfunktionen und Authentifizierungscodes



Eve möchte

- lauschen
- ändern
- personifizieren

Aufgaben

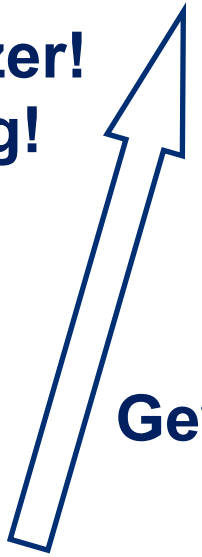


- Vertraulichkeit – Lauschen
- Authentizität – Tauschen des Datenursprungs
- Integrität – Änderung der Daten
- Zurechenbarkeit – Leugnen des Datenursprungs

Beispiel WLAN

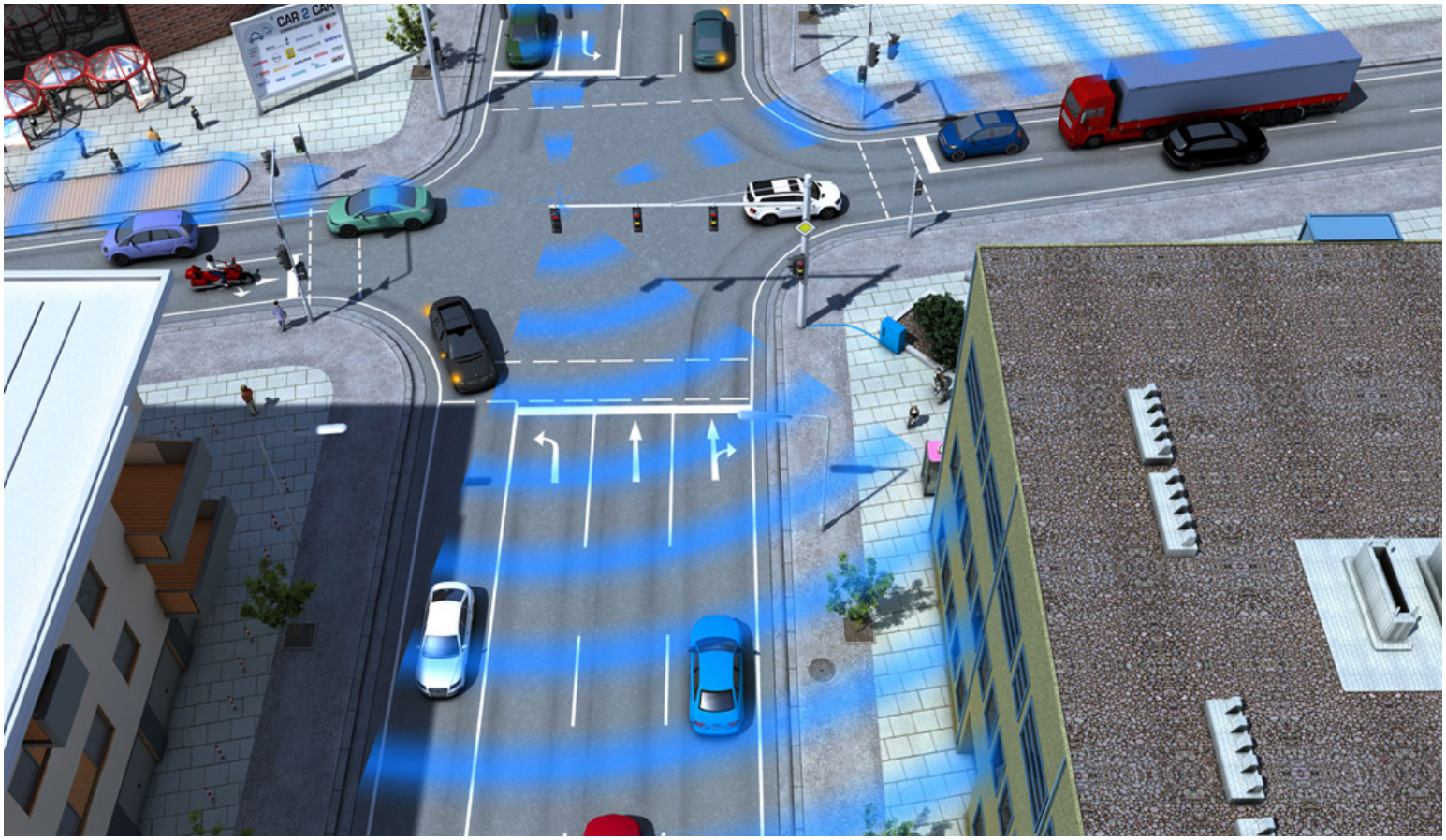
Ich bin’s, dein Besitzer!
Gewähr mir Zugang!

Authentifizierung!



Ich bin’s,
dein Besitzer!
Gewähr mir Zugang!

Beispiel Car2Car



Asymmetrische Verschlüsselung

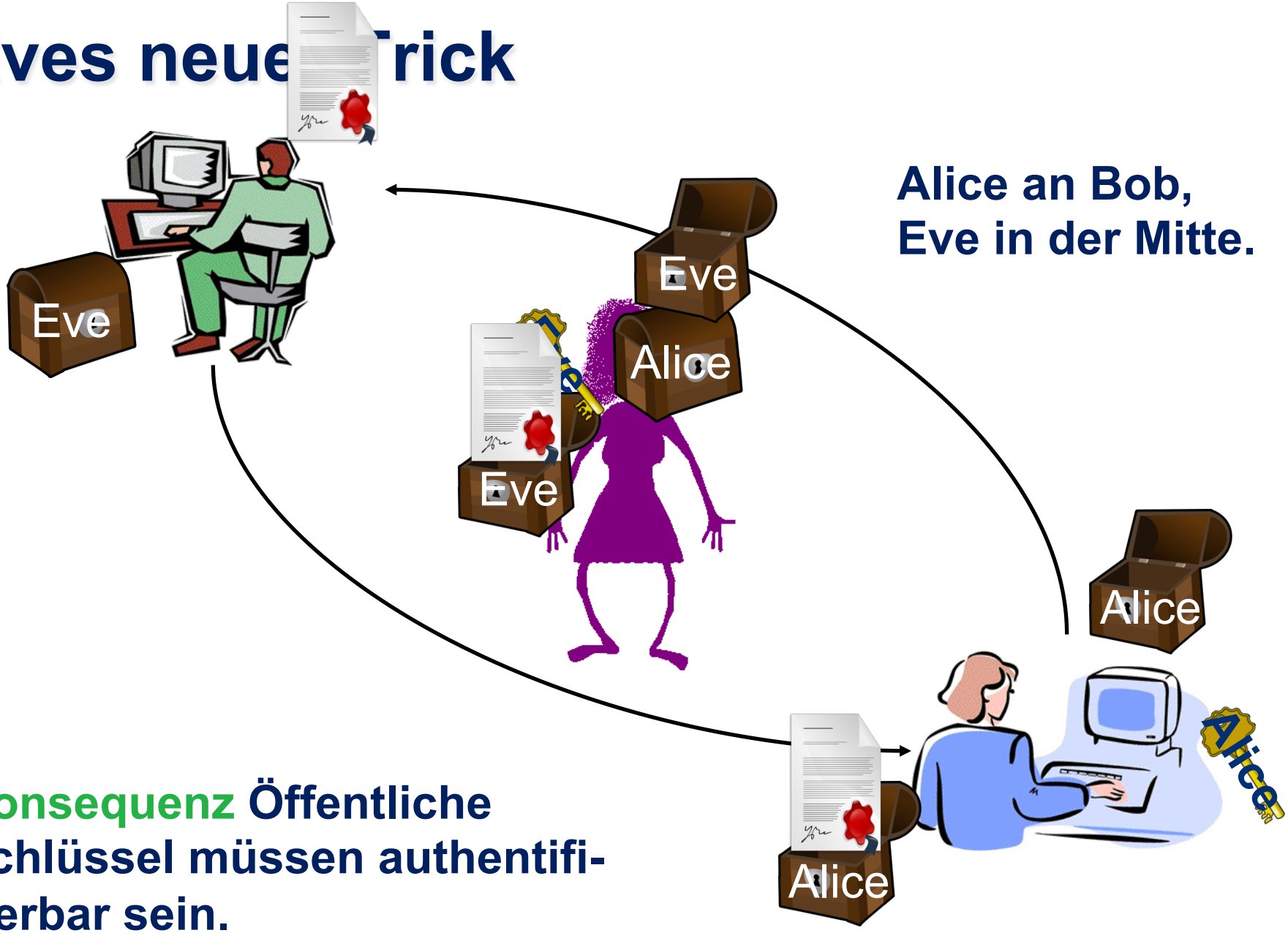
Bob an Alice:

Alice an Bob:
Rollen werden vertauscht!

öffentlicher Schlüssel

geheimer Schlüssel

Eves neue Trick



Alice an Bob,
Eve in der Mitte.

Konsequenz Öffentliche Schlüssel müssen authentifizierbar sein.

VII.1 Integrität und Hashfunktionen

Definition 7.1 Eine Hashfunktion ist eine Funktion h der Form $h : \Sigma^* \to \Sigma^n$, wobei Σ eine endliche Menge und $n \in \mathbb{N}$ ist.

Eine Kompressionsfunktion ist eine Funktion h der Form $h : \Sigma^m \to \Sigma^n$, wobei Σ eine endliche Menge und $n,m \in \mathbb{N}$ mit $m > n$ sind.

Hashfunktionen

Bemerkungen

- h(x) heißt Hashwert oder Hash oder Fingerabdruck von x.
- h kann nicht injektiv sein.
- Nehmen immer an, dass h surjektiv ist.
- Hashfuktionen auch in Datenstrukturen und Algorithmen.
- An kryptographische Hashfunktionen stärkere Anforderungen, da böswillige Angreifer angenommen werden müssen.
- Hashfunktionen müssen sehr effizient berechenbar sein.

Beispiele

Beispiel 1

$$h: \{0,1\}^* \to \{0,1\}$$
$$x = x_1 \dots x_k \mapsto x_1 \oplus \dots \oplus x_k$$

ist eine Hashfunktion.

Beispiel 2

$$h: \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n$$
$$a = a_1 a_2 \mapsto a_1 + a_2 \bmod n$$

ist eine Kompressionsfunktion.

Einwegfunktionen

Einwegfunktionen Sei $D = \Sigma^*$ oder $D = \Sigma^m$ und $h : D \to \Sigma^n$ eine Hash- oder Kompressionsfunktion. h heißt Einweg-funktion, wenn kein Angreifer für zufällig gewähltes $s \in \Sigma^n$ mit hoher Wahrscheinlichkeit und mit vertretbarem Aufwand ein $x \in D$ mit $h(x) = s$ berechnen kann.

Bemerkungen

- Präzise mathematische Definition mit Komplexitätstheorie und für parametrisierte Funktionen möglich.
- Können Einweg-Eigenschaft für Funktionen $f : D \to R$ definieren, $|D|, |R| < \infty$.
- Einwegfunktionen Grundlage symmetrischer Kryptographie.

Beispiele

Beispiel 1

$$h: \{0,1\}^* \to \{0,1\}$$
$$x = x_1 \ldots x_k \mapsto x_1 \oplus \cdots \oplus x_k$$

ist eine Hashfunktion, aber nicht Einwegfunktion

Beispiel 2

$$h: \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n$$
$$a = a_1 a_2 \mapsto a_1 + a_2 \bmod n$$

ist eine Kompressionsfunktion, aber nicht Einwegfunktion.

Beispiele

Beispiel 3 Sei p eine Primzahl und g ein Generator von $\mathbb{Z}_p^*$.

$$h: \{0,1,\ldots,p-2\} \rightarrow \mathbb{Z}_p^*$$
$$x \mapsto g^x \bmod p$$

ist eine Einwegfunktionfunktion, falls der diskrete Logarithmus in $\mathbb{Z}_p^*$ schwer zu berechnen ist.

Kollisionen und Kollisionsresistenz

Kollisionen Sei $D = \Sigma^*$ oder $D = \Sigma^m$ und $h : D \to \Sigma^n$ eine Hash- oder Kompressionsfunktion. Eine Kollison in h ist ein Paar $(x, y) \in D \times D$ mit $x \neq y$ und $h(x) = h(y)$.

Kollisionsresistente Funktionen Sei $D = \Sigma^*$ oder $D = \Sigma^m$ und $h : D \to \Sigma^n$ eine Hash- oder Kompressionsfunktion. h heißt kollisionsresistent, wenn mit vertretbarem Aufwand keine Kollisionen in h gefunden werden können.

Bemerkung Präzise mathematische Definition mit Komplexitätstheorie und für parametrisierte Funktionen möglich.

Beispiele

Beispiel 1

$$h: \{0,1\}^* \to \{0,1\}$$
$$x = x_1 \ldots x_k \mapsto x_1 \oplus \cdots \oplus x_k$$

ist nicht kollisionsresistent.

Beispiel 2

$$h: \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n$$
$$a = a_1 a_2 \mapsto a_1 + a_2 \bmod n$$

ist nicht kollisionsresistent.

Einweg-Eigenschaft und Kollisionsresistenz

Beobachtung Sei $D = \Sigma^*$ oder $D = \Sigma^m$ und $h : D \to \Sigma^n$ eine Hash- oder Kompressionsfunktion. Ist h keine Einweg-funktion, so ist h auch nicht kollisionsresistent.

Idee der Reduktion Sei A Angreifer, der effizient Urbilder in h berechnet.

1. Wähle $x \in D$ zufällig und berechne $z = h(x)$.
2. Starte A mit Eingabe z. Falls $A(z) = y$ mit $x \neq y$, dann Ausgabe (x, y), sonst Abbruch.

Anwendung von Hashfunktionen

- Alice möchte Daten M so speichern, dass sie vor Veränderungen geschützt sind (Integrität).
- Alice besitzt PC mit großem, aber unsicherem Speicher und Smartcard mit kleinem, aber sicherem Speicher.
- Alice wählt kollisionsresistente Hashfunktion h.
- Alice speichert M auf dem PC und h(M) auf der Smartcard.

Speicherung von Passwörtern In der Regel werden nur Hashwerte von Passwörtern gespeichert. Bei Anmeldung wird dann Hashwert des eingegebenen Wortes mit gespeichertem Hashwert verglichen.

Eigenschaft kollisionsresistenter Funktionen

Beobachtung Sei $D = \Sigma^*$ oder $D = \Sigma^m$ und $h : D \to \Sigma^n$ eine Hash- oder Kompressionsfunktion. Dann kann in Zeit $\approx |\Sigma|^{n/2}$ eine Kollision in h gefunden werden.

Beweisidee Wähle $x_j \in D$ so lange, bis x_k, x_l mit $x_k \neq x_l$ und $h(x_k) = h(x_l)$ gefunden wurden.

Geburtstagparadox Mit $\approx |\Sigma|^{n/2}$ vielen x_j wird Kollision gefunden.

Konstruktion von Hashfunktionen

Konstruktion guter Hashfunktionen häufig in zwei Schritten

a. **Konstruiere gute Kompressionsfunktion h.**
b. **Durch Anwendung allgemeiner Konstruktionsmethode (Merkle-Damgård) erhalte daraus Hashfunktion H.**

VII.2 Konstruktion von Kompressionsfunktionen

Allgemeine Konstruktionen

- aus Blockchiffren
- aus Blockchiffren und CBC-Modus
- ...

Spezielle Konstruktionen

- MD4, MD5
- RIPEMD
- SHA1, SHA2
- Keccak – SHA3

Konstruktion von Kompressionsfunktionen

- MD4 – SHA1 gelten als nicht mehr sicher genug.
- Für SHA2 (noch) keine Schwächen bekannt
- NIST startete 2007 Auswahlprozess für alternativen Standard.
- Oktober 2012 wurde Keccak als Gewinner gewählt.
- Keccak unmittelbar Hashfunktion
- beruht auf Schwammfunktionen
- Schwammfunktionen liefern viele andere kryptographischen Primitiven

Schwammfunktionen

Schwammfunktion definiert durch

1. Weite b, Kapazität c, Blocklänge r, wobei $b = r + c$
2. Ausgabelänge l
3. Bijektion $f : \{0,1\}^b \to \{0,1\}^b$
4. Padding $pad : \{0,1\}^* \to \left(\{0,1\}^r\right)^*$

   (Folgen mit Länge $h \cdot r, h \in \mathbb{N}$)
5. zwei Phasen
   a) Aufnahmephase (absorbing phase)
   b) Pressphase (squeezing phase)

Schwammfunktionen - Zustände

Zustände

- $s \in \{0,1\}^b$ Zustand
- $s = \bar{s} \parallel \hat{s}, \bar{s} \in \{0,1\}^r, \hat{s} \in \{0,1\}^c$
- $\bar{s}$ äußerer Zustand, $\hat{s}$ innerer Zustand

Schwammfunktionen - Aufnahmephase

Eingabe $m \in \{0,1\}^*$

$$pad(m) := p_0 \| \cdots \| p_{n-1}, p_i \in \{0,1\}^r$$

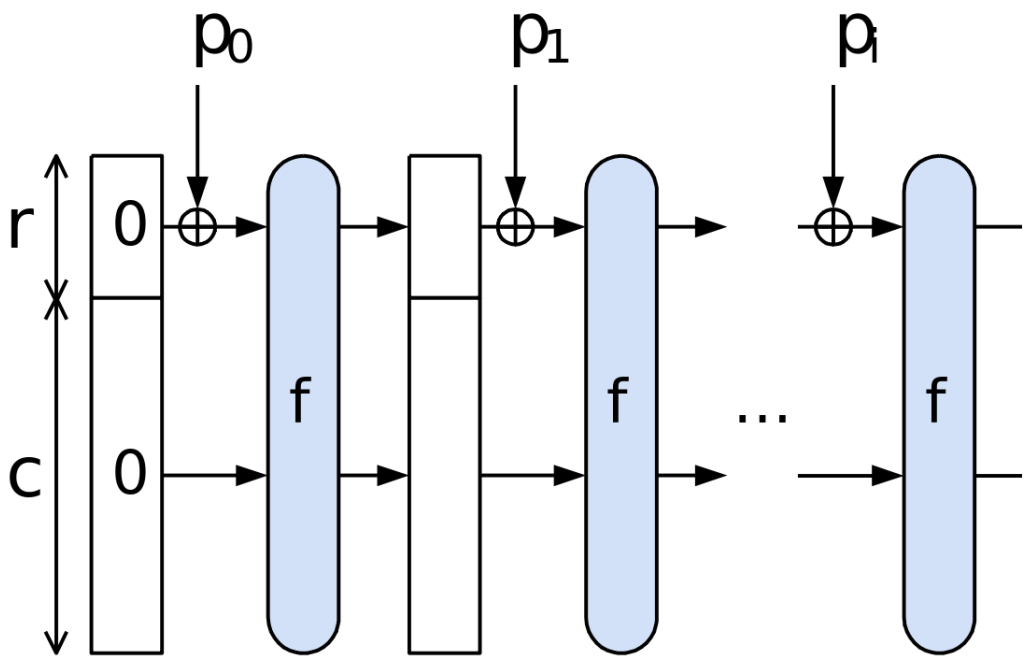
$$s_0 = 0^b, s_0 = \bar{s}_0 \| \hat{s}_0$$

Für $i = 1, \ldots, n$ **definieren**

$$s_i := f(\bar{s}_{i-1} \oplus p_{i-1} \| \hat{s}_{i-1}),$$

$$s_i = \bar{s}_i \| \hat{s}_i, \bar{s}_i \in \{0,1\}^r, \hat{s}_i \in \{0,1\}^c$$

Schwammfunktionen und Keccak



Schwammfunktionen - Pressphase

Ausgabe (Länge Vielfaches von r)

$$z \in \{0,1\}^*, z = z_0 \| \cdots \| z_{h-1}, z_j \in \{0,1\}^r, h := l/r$$

Für $j = 0, \ldots, h-1$ definieren

$$\begin{aligned} z_j &:= \bar{s}_{n+j} \\ s_{n+j+1} &:= f\left(s_{n+j}\right), \\ s_{n+j+1} &= \bar{s}_{n+j+1} \| \hat{s}_{n+j+1}, \bar{s}_{n+j+1} \in \{0,1\}^r, \hat{s}_{n+j+1} \in \{0,1\}^c \end{aligned}$$

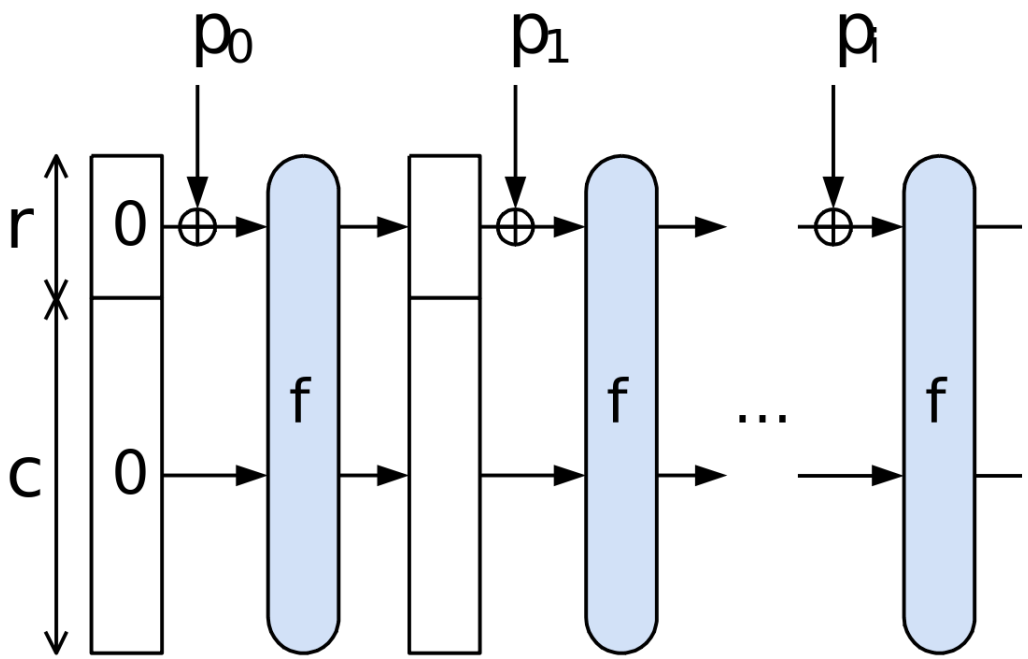
beliebige Ausgabelänge l

$$z = z_0 \| \cdots \| z'_{h-1}, z_j \in \{0,1\}^r,$$

$$h := \lceil l/r \rceil, k := l - (h-1) \cdot r$$

$z_{h-1} := \lfloor z'_{h-1} \rfloor_k$ = ersten k Bits von z'_{h-1}

Schwammfunktionen - Pressphase



Schwammfunktionen und Keccak

Default Paramter

$b := 1600$
$r := 1024$
$c := 576$
$l := 512$

zusätzliche Paramter

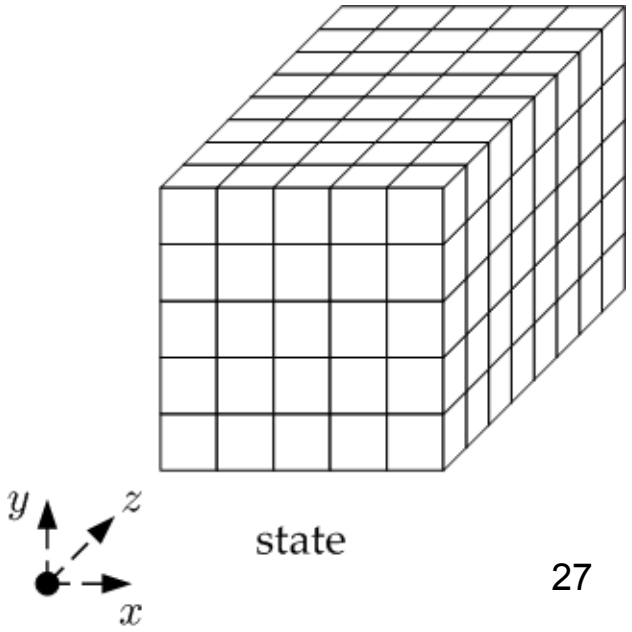
$w := 64$
$\ell := 6$

$$b = 25 \cdot w = 25 \cdot 2^{\ell}$$

Keccak-Zustände

$$\{0,1\}^{1600} = \left(\{0,1\}^{5\times 5}\right)^{64}$$

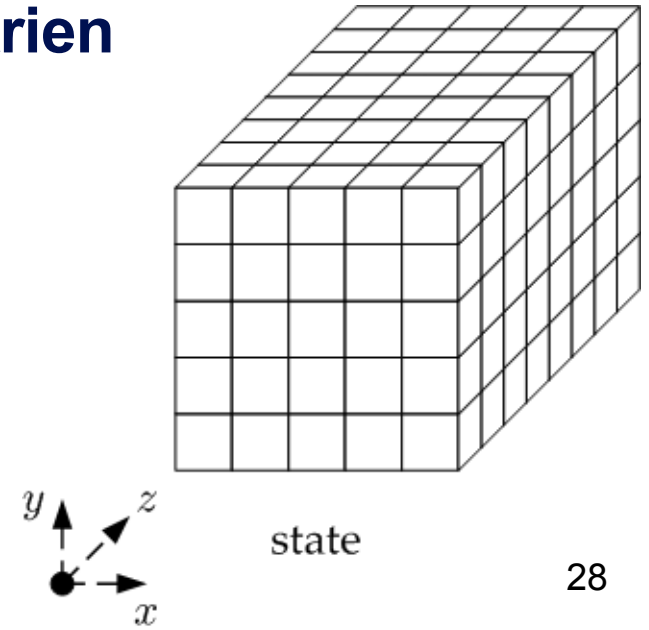
$$s = \left(s(x,y,z) : x,y \in \mathbb{Z}_5, z \in \mathbb{Z}_w\right)$$



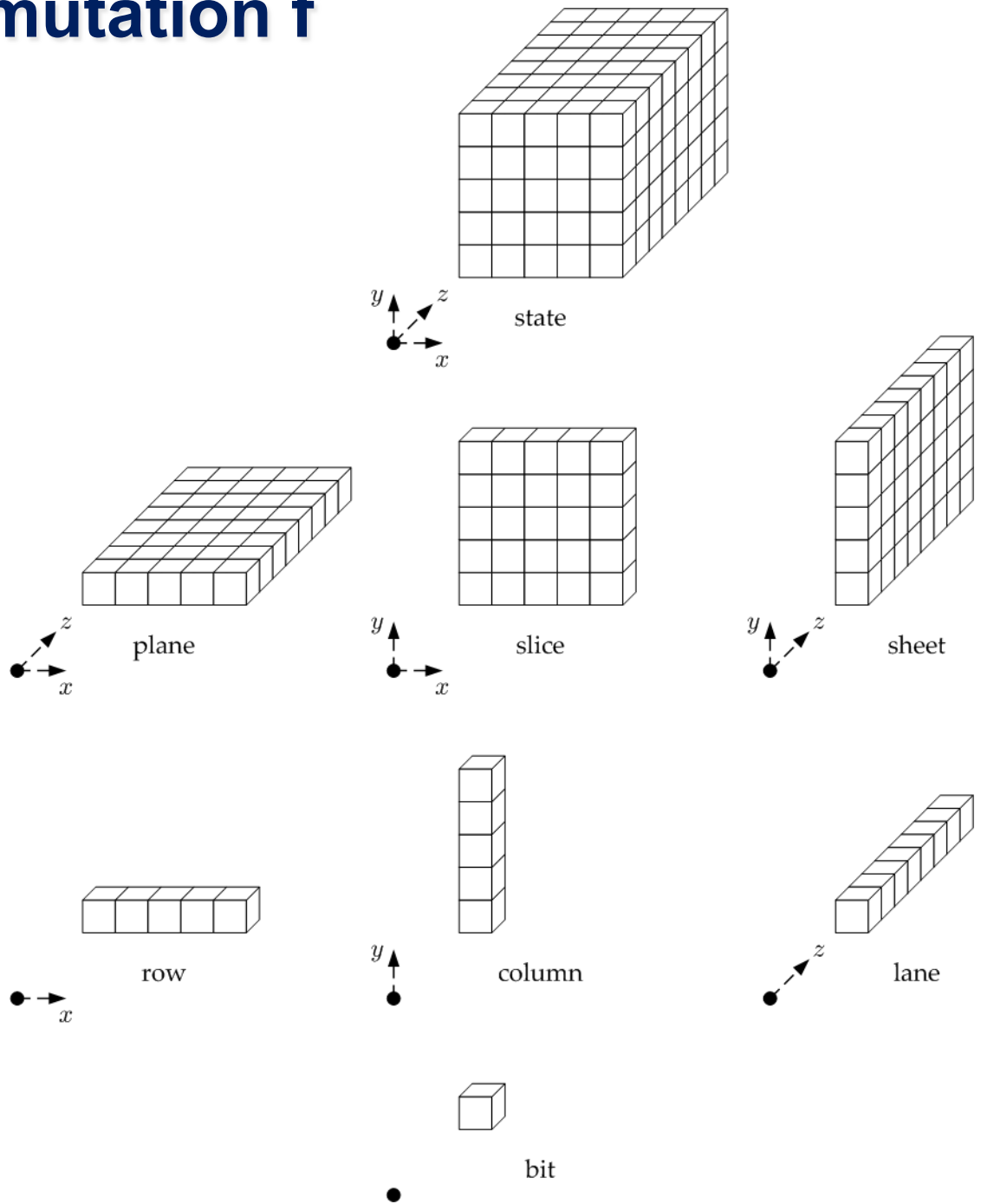
Keccak-Permutation f

- Ziel sind Konfusion und Diffusion
- Funktion f besteht aus $n_r := 12 + 2 \cdot l = 24$ Runden
- jede Runde setzt sich aus 5 Funktionen zusammen:

  $\theta, \rho, \pi, \chi, \iota$
- $R := \iota \circ \chi \circ \pi \circ \rho \circ \theta$
- $\theta, \rho, \pi, \chi, \iota$ sorgen für Vermischung innerhalb der drei Dimensionen und brechen Symmetrien

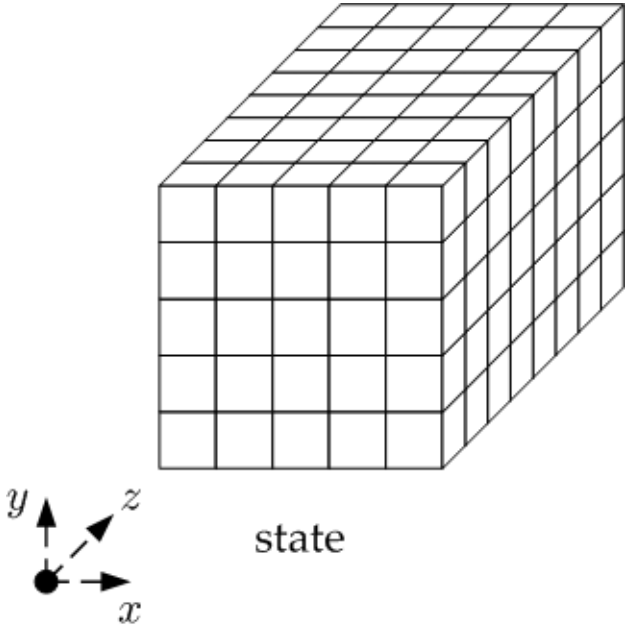


Keccak-Permutation f



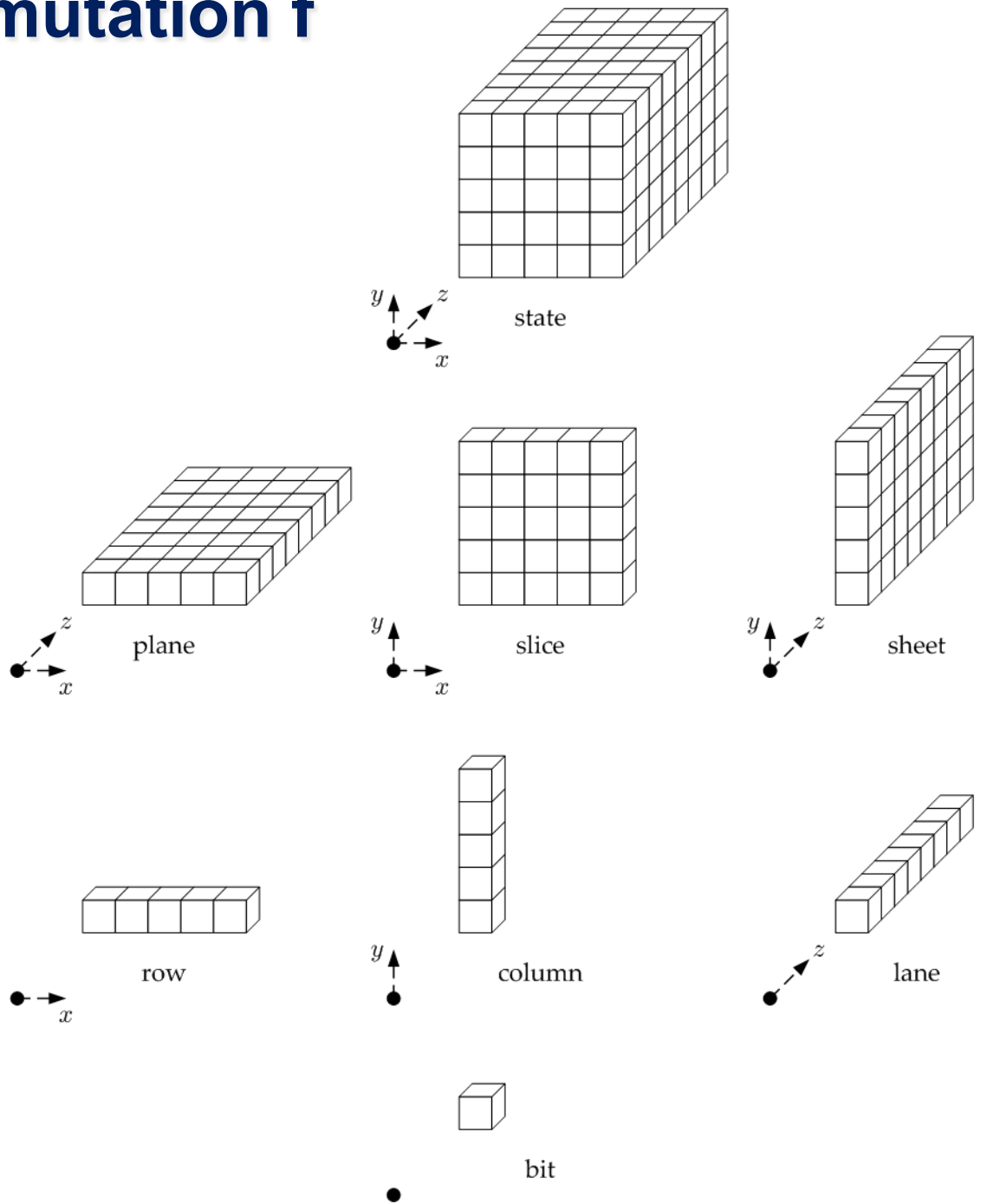
Keccak-Permutation f

$$\theta: \quad s(x,y,z) \leftarrow s(x,y,z) + \sum_{y'=0}^{4} s(x+1,y',z-1) + \sum_{y'=0}^{4} s(x-1,y',z)$$



- vermischt aufeinanderfolgende Slices

Keccak-Permutation f



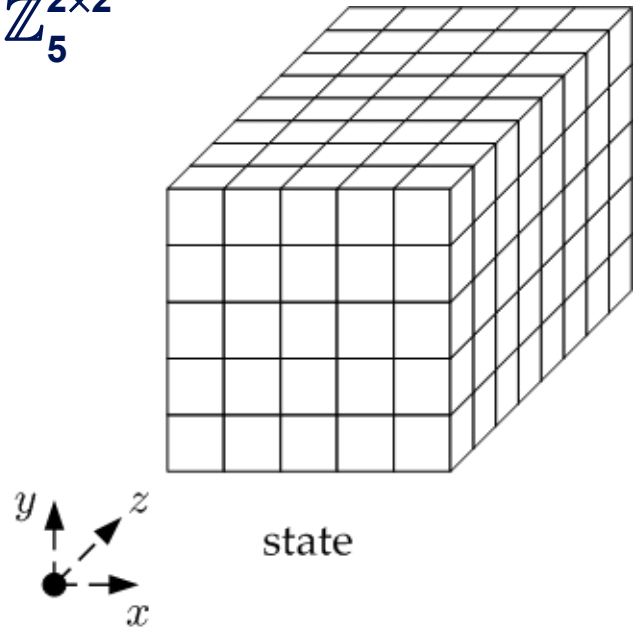
Keccak-Permutation f

$$\rho: \quad s(x,y,z) \leftarrow s(x,y,z-(t-1)(t+1)/2),$$

wobei $0 \le t \le 24$ und

$$M^t \cdot (1,0)^T = (x,y)^T \text{ in } \mathbb{Z}_5^{2\times 2}$$

$$M = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}$$



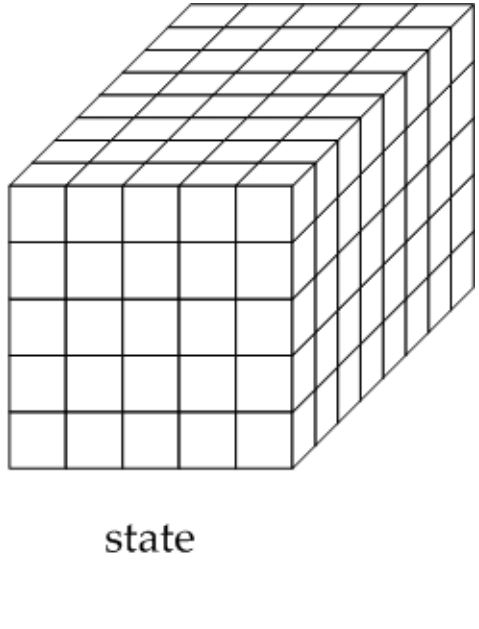
- permutiert innerhalb einer Lane

Keccak-Permutation f

$$\pi: \quad s(x,y,-) \leftarrow s(x',y',-),$$

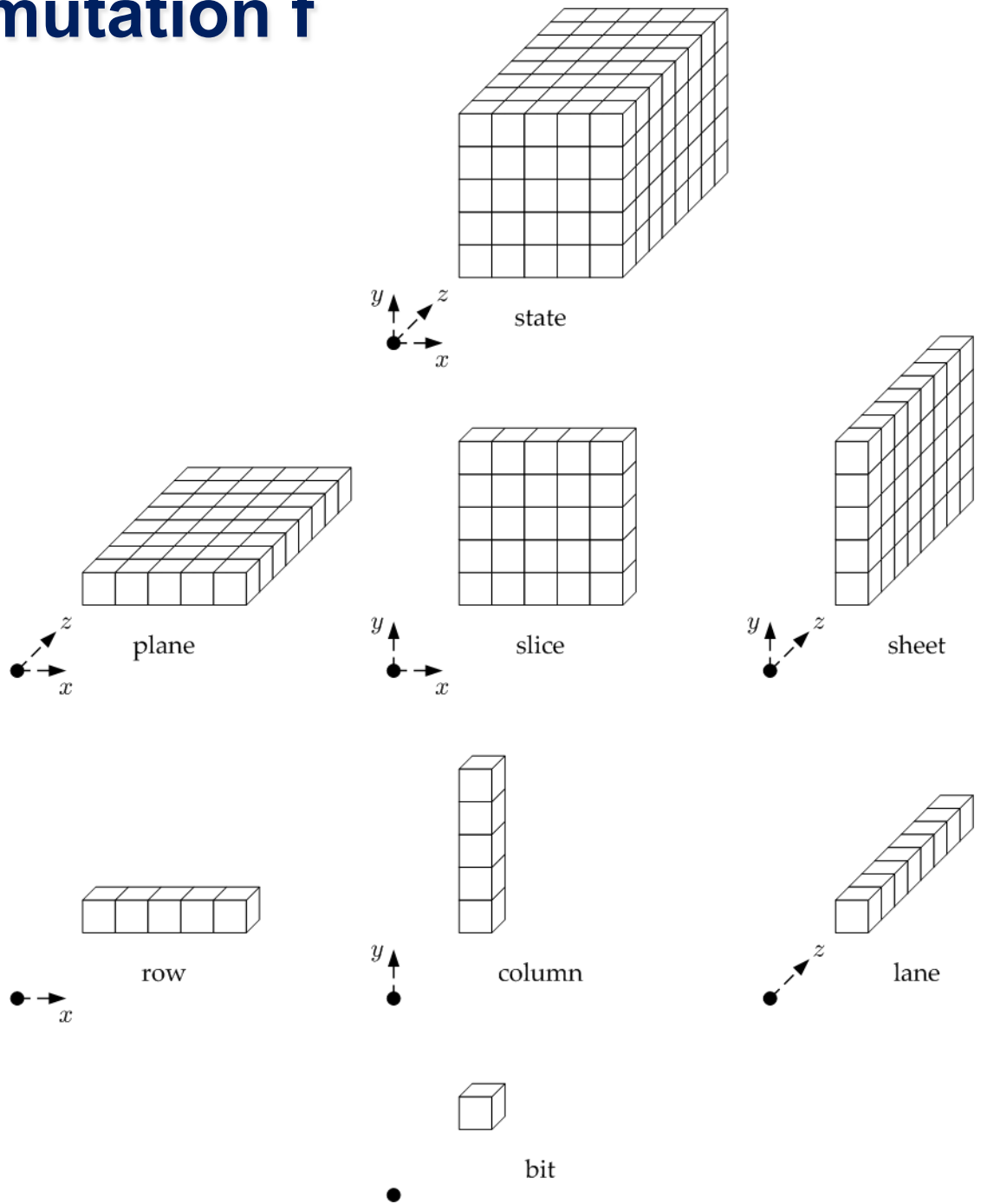
$$\text{wobei } M^t \cdot (x',y')^T = (x,y)^T$$

$$M = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}$$



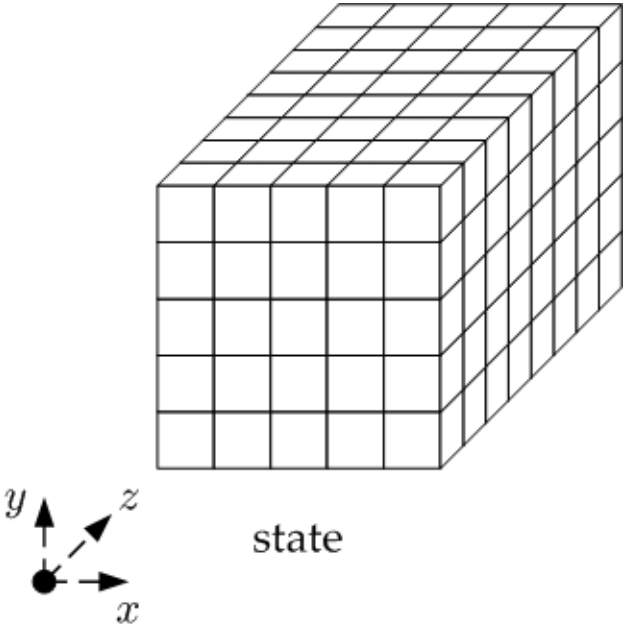
- permutiert innerhalb eines Slices

Keccak-Permutation f



Keccak-Permutation f

$$\chi: \quad s(x,-,-) \leftarrow s(x,-,-) + \left(s(x+1,-,-)+1\right) \cdot s(x+2,-,-)$$



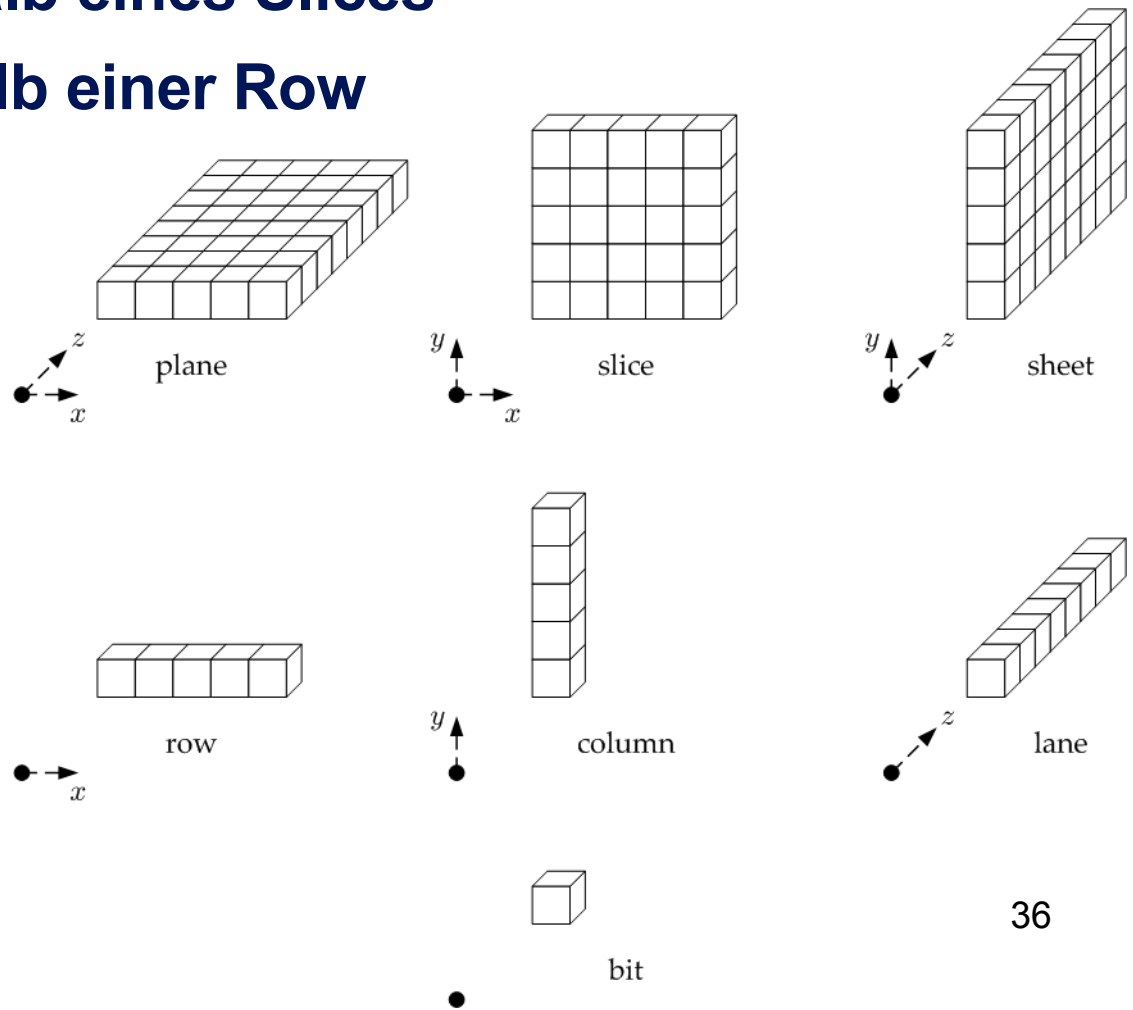
- vermischt innerhalb einer Row

$$\iota: \quad s \leftarrow s + RC_i$$

- addiert Konstanten, bricht Symmetrien

Keccak-Permutation f

- θ : vermischt aufeinanderfolgend Slices
- ρ : permutiert innerhalb einer Lane
- π : permutiert innerhalb eines Slices
- χ : vermischt innerhalb einer Row
- ι : bricht Symmetrien



Keccak-Paddings

pad10* erweitert Nachricht m um eine 1 gefolgt von der Mindestanzahl von 0, um Vielfaches der Weite b zu erhalten.

pad10*1 erweitert Nachricht m um eine 1 gefolgt von der Mindestanzahl von 0 und einer 1, um Vielfaches der Weite b zu erhalten.

Zahlentheoretische Konstruktion

p Primzahl, so dass $q = (p-1)/2$ ebenfalls Primzahl,

a Generator von $\mathbb{Z}_p^*$, $b \in \mathbb{Z}_p^*$ beliebig

$$h: \{0,1,\ldots,q-1\}^2 \to \{1,2,\ldots,p-1\}$$
$$(x_1, x_2) \mapsto a^{x_1} b^{x_2} \bmod p$$

Formal keine Kompressionsfunktion, aber hinreichend nahe,

da $\left|\{0,1,\ldots,q-1\}^2\right| = q^2 \gg p-1 = \left|\{1,2,\ldots,p-1\}\right|$.

Satz 7.2 Gegeben eine Kollision (x, x') in h, kann der diskrete Logarithmus von b zur Basis a in Zeit $\mathcal{O}\left(\log(p)^3\right)$ berechnet werden.

VII.3 Von K.-funktionen zu H.-funktionen

Merkle-Damgård Konstruktion (MD)

a) Konstruiert Hashfunktion $H: \{0,1\}^* \to \{0,1\}^n$ aus beliebiger Kompressionsfunktion $h: \{0,1\}^l \to \{0,1\}^n$ mit $l > n$.

b) Zeigen, wie aus Kompressionsfunktion $h: \{0,1\}^{2n} \to \{0,1\}^n$ Kompressionsfunktions $H: \{0,1\}^{<2^n} \to \{0,1\}^n$ konstruiert werden kann.

c) Techniken identisch, in b) etwas klarer.

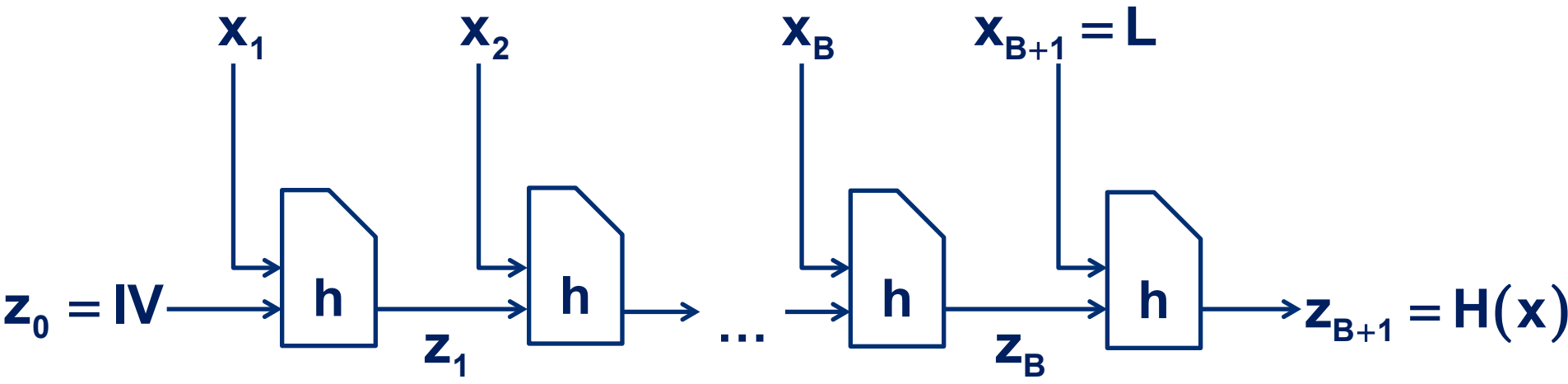
Merkle-Damgård Konstruktion

$$h:\{0,1\}^{2n} \to \{0,1\}^n, H:\{0,1\}^{<2^n} \to \{0,1\}^n$$

MD bei Eingabe $x \in \{0,1\}^L, L < 2^n$

1. Setze $B := \lceil L/n \rceil$ und ergänze x mit 0en, so dass Länge von x ein Vielfaches von n ist.
2. Schreibe ergänztes x als $x_1 x_2 \cdots x_B, x_j \in \{0,1\}^n$, setze $x_{B+1} := L$, wobei L mit genau n Bits dargestellt wird.
3. Setze $z_0 := 0^n$.
4. Für $j = 1,\ldots,B+1$, setze $z_j := h(z_{j-1} \| x_j)$.
5. Setze $H(x) := z_{B+1}$.

Merkle-Damgård Konstruktion



Analyse der Merkle-Damgård Konstruktion

Satz 7.3 **Ist h kollisionsresistent, so ist die Funktion H, die aus Anwendung der Merkle-Damgård Konstruktion auf h entsteht, ebenfalls kollisionsresistent.**