Das Sticker-Modell für DNA-Computing

1. Repräsentation von Information

Sei s ein Speicher-Strang (single-stranded) der Länge N, $s \in \{A, T, C, G\}^*$

s unterteilt sich in K nichtüberlappende Regionen $r_1, \ldots, r_K$ mit $|r_i| = M$ (insbesondere: $N \geq K \cdot M$)

$$s = t_0 r_1 t_1 r_2 \cdots t_{K-1} r_K t_K$$

(es kann sein, dass $t_0 = t_1 = \ldots = t_K = \lambda$)

Annahme: $s = \alpha r_i \beta \rightarrow$

$\alpha = t_0 r_1 t_1 \cdots r_{i-1} t_{i-1}$ und

$\beta = t_i r_{i+1} \cdots t_{K-1} r_K t_K$

Die zu den Regionen $r_1, \ldots, r_K$ komplementären Sequenzen heißen Sticker $\bar{r}_1, \ldots, \bar{r}_K$

Idee: Ein Speicherstrang, bei dem an einige der Regionen r_i der entsprechende Sticker $\bar{r}_i$ gebunden ist, repräsentiert einen Bitstring der Länge K.

Beispiel

$$\overbrace{A-T-C-G-G}^{r_1}-\overbrace{T-C-A-T-A}^{r_2}-\overbrace{G-C-A-C-T}^{r_3}$$

$$\bar{r}_1 = T-A-G-C-C \qquad\qquad\qquad C-G-T-G-A = \bar{r}_3$$

$$1 \qquad\qquad 0 \qquad\qquad 1$$

Sticker: $\bar{r}_1 = T-A-G-C-C$

$\bar{r}_2 = A-G-T-A-T$

$\bar{r}_3 = C-G-T-G-A$

- Ein Speicherstrang, bei dem an einige der Regionen der passende Sticker gebunden ist, heißt auch Speicherkomplex.

Wir identifizieren im folgenden einen Speicherkomplex C mit dem entsprechenden Bitstring.

- Eine Multimenge von Speicherkomplexen ist eine DNA-Lösung $L \subseteq \{0,1\}^K$

- Informationsdichte im Sticker-Modell: $\frac{1}{M}$ bits/base

2. Operationen im Sticker-Modell:

a) Vereinigen zweier DNA-Lösungen
$L := L_1 \cup L_2$

b) Separieren einer DNA-Lösung L an der Position $i \in \{1, \ldots, k\}$

$(L_0, L_1) := separate(L, i)$

Danach enthält L_j alle Bitstrings aus L, wo an der i-ten Position eine 1 steht.

c) Setzen des i-ten Bits in einer DNA-Lösung L auf $b \in \{0,1\}$

$L' := set(L, i, b)$

Danach gilt:

$$L' = \{ ubv \mid |u| = i-1, |v| = K-i, \exists c \in \{0,1\} : ucv \in L \}$$

Es wird sich später zeigen, dass set$(L,i,1)$ leichter als set$(L,i,0)$ zu realisieren ist.

Deshalb betrachten wir auch noch die Operation

$L' := \text{clear}(L)$

Danach gilt:

$$L' = \{ u0^{K-\ell} \mid |u| = \ell, \ \exists v \in \{0,1\}^{K-\ell} : uv \in L \}$$

Hierbei ist ℓ eine feste Zahl, d.h. kein Eingabeparameter.

d) Initialisierung einer "Mutter-Lösung":

$L = \text{init}(K,B)$

wobei $K \in \mathbb{N}$, $0 \leq B \leq K$.

Danach gilt:

$$L = \{ u0^{K-B} \mid u \in \{0,1\}^B \}$$

L heißt auch eine (K,B)-Library

Beispiel für einen Algorithmus im Sticker-Modell:

Lösung von Minimal Set Cover

Minimal Set Cover ist das folgende Berechnungsproblem:

EINGABE: Mengen $\{1,2,\ldots,A\}$, $\{C_1, C_2, \ldots, C_B\}$ mit $C_i \subseteq \{1,2,\ldots,A\}$

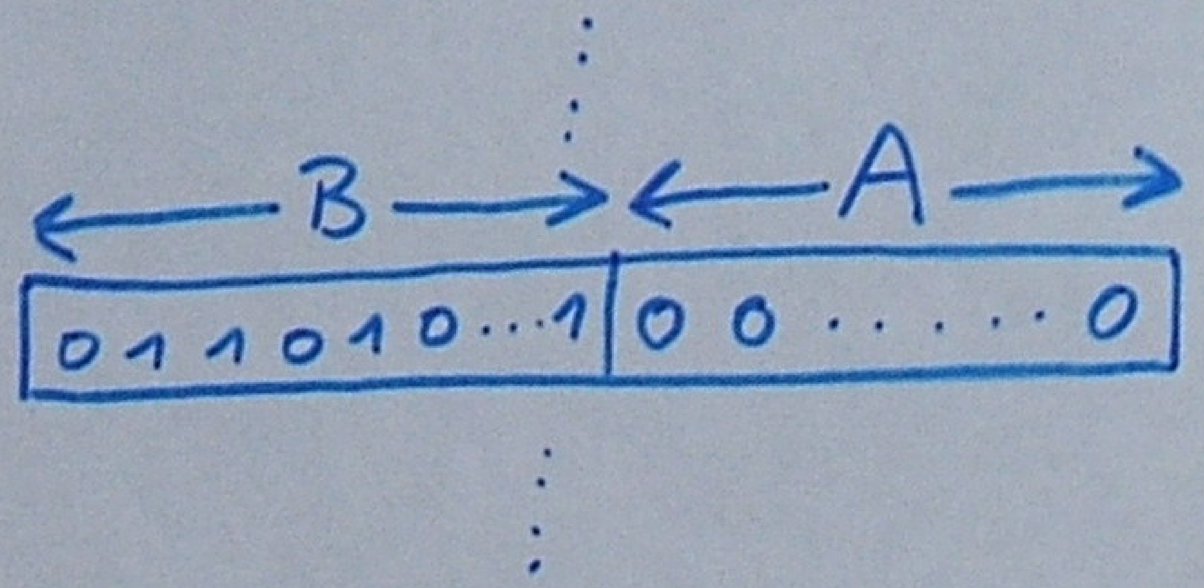
Zahl $m \leq B$

FRAGE: Gibt es eine Teilmenge $\mathcal{U} \subseteq \{C_1, C_2, \ldots, C_B\}$ mit $|\mathcal{U}| \leq m$, $\bigcup_{C \in \mathcal{U}} C = \{1,2,\ldots,A\}$

Minimal Set Cover ist NP-vollständig!

Sei $K := A + B$

$L_0 := init(K, B)$




```
• for i := 1 to B do
      (L_on, L_off) := separate(L_0, i)
      for all j ∈ C_i do
          L_on := set(L_on, B+j, 1)
      endfor
      L_0 := L_on ∪ L_off
  endfor
• for i := B+1 to B+A do
      (L_0, L_bad) := separate(L_0, i)
  endfor
• for i := 0 to B-1 do
      for j := i downto 0 do
          (L'_{j+1}, L_j) := separate(L_j, i+1)
          L_{j+1} := L'_{j+1} ∪ L_{j+1}
      endfor
  endfor
```

(Nach diesem Schritt enthält L_i genau die Bitstrings wo genau i viele der ersten B bits 1 sind, und alle A viele folgenden bits 1 sind.)

```
for i := 1 to m do
    if L_i ≠ ∅ then return(YES)
endfor
return(NO)
```

<u>Beachte</u>: Dieser Algorithmus verwendet nicht die schwer realisierbare Operation $set(L, i, 0)$.

Biotechnologische Realisierung der Operationen im Sticker-Modell

a) Vereinigung zweier DNA-Lösungen

$L := L_0 \cup L_1$

↳ Zusammenschütten und Mischen zweier Reagenzgläser

Probleme:

- Ab einer Länge von ca 15 000 Basen werden die Speicherkomplexe aufgrund von Scherkräften beim Mischen fragmentiert
- Speicherkomplexe bleiben an den Wänden der Reagenzgläser haften.

b) Separieren

$(L_0, L_1) := \text{separate}(L, i)$

Idee: Jeder Region r_i auf dem Speicherstrang s wird neben dem Sticker $\bar{r}_i$ noch eine weitere einsträngige Sequenz p_i (Filter) zugeordnet, die sich über Wasserstoffbrückenbindungen nur an die Region r_i binden kann.

Jeder Filter p_i ist z.B. magnetisch markiert.

(1) Füge der DNA-Lösung L eine große Menge des Filters p_i hinzu. Dieser wird sich an die Region r_i binden, falls auf dem entsprechenden Speicherkomplex das i-te Bit 0 ist (d.h. Region

r_i ist nicht bereits durch den Sticker $\bar{r}_i$ besetzt.

(2) Wasche ungebundene Filtersequenzen p_i aus der Lösung heraus.

(3) Aufgrund der magnetischen Markierung von p_i können nun alle Speicherkomplexe aus L herausgefiltert werden, an die sich der Filter p_i gebunden hat (→ Lösung L_0). Übrig bleibt die Lösung L_1

(4) Entferne von allen Speicherkomplexen aus L_0 den Filter p_i, z.B durch Erhitzen.

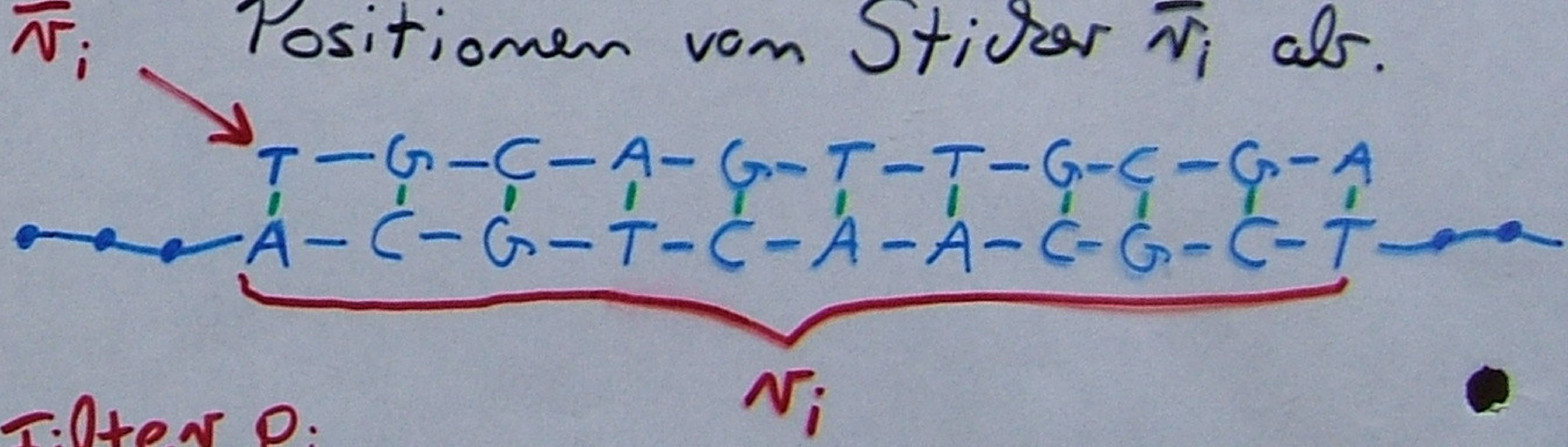
Problem beim letzten Schritt:

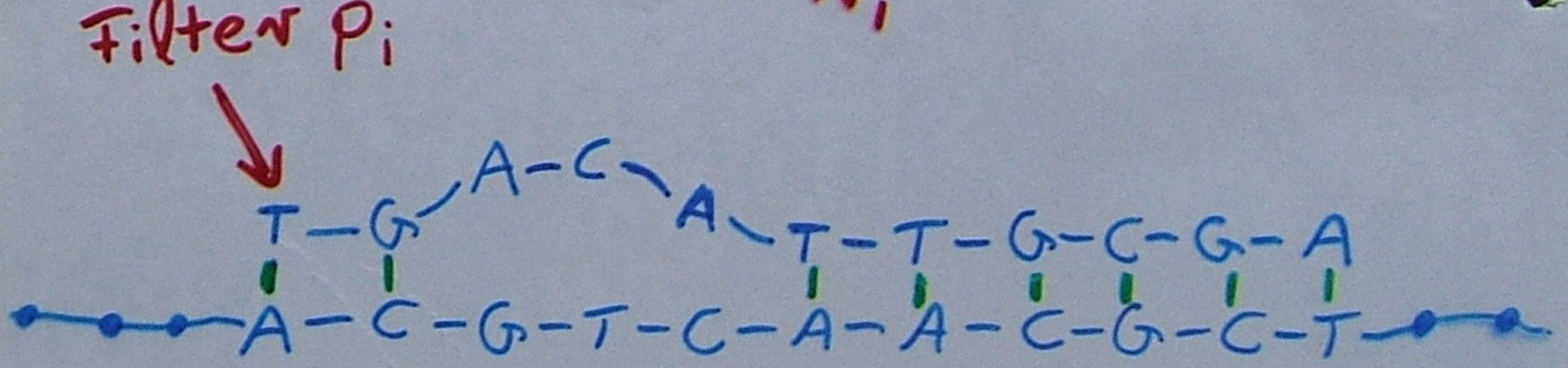
Beim Erhitzen soll sich nur der Filter p_i von einem Speicherkomplex ablösen, ein Sticker $\bar{r}_j$ ($i \neq j$) soll nicht abgelöst werden.

Lösung: Filter p_i muß sich bei einer niedrigeren Temperatur von der Region r_i ablösen, als es Sticker $\bar{r}_i$ tut.

Dies kann z.B. wie folgt erreicht werden:

- Filter p_i weicht an einigen Positionen von Sticker $\bar{r}_i$ ab.





- Filter p_i ist ein echter Teilstrang von $\bar{r}_i$

- Filter p_i und Sticker $\bar{r}_i$ haben identische Basenabfolge, aber im Sticker $\bar{r}_i$ wird anstatt Desoxyribose ein anderes Molekül verwendet (→ PNA / DNG), welches zu stärkeren Wasserstoffbrückenbindungen zur Region r_i führt.

c) i-te Bit auf 1 setzen

$L' := set(L, i, 1)$

Füge zur Lösung L große Menge des Stickers $\bar{r}_i$ hinzu; ungebundene Sticker werden aus der DNA-Lösung wieder herausgewaschen.

d) i-te Bit auf 0 setzen

$L' := set(L, i, 0)$

schwierig zu realisieren!

e) Clear-Operation:

$L' := clear(L)$

Setze alle Bits in einem festen Bereich des Speicherstrangs auf 0.

Sticker $\bar{r}_i$ für ein i aus diesem Bereich muß sich von der Region r_i bereits bei einer niedrigeren Temperatur ablösen als dies ein Sticker $\bar{r}_j$ für ein j außerhalb des Bereichs tut.

Kann mit gleichen Methoden wie bei der separate-Operation erreicht werden.

d) Initialisierung

$L := init(K, B)$

Erstellen einer (K, B)-library.

(1) Synthetisiere mittels mittels Polymerase-Ketten-Reaktion (PCR) ca 2^B Kopien des leeren Speicherstrangs s (alle Bits $= 0$)
↳ Lösung L

(2) Halbiere L in zwei Teillösungen L_0 und L_1.

(3) Füge zu L_1 eine große Menge aller Sticker $\overline{r}_1, \overline{r}_2, \ldots, \overline{r}_B$ hinzu.
↳ Bit 1 bis Bit m wird auf allen Speicherkomplexen in L_1 auf 1 gesetzt.

(4) Wasche ungebundene Sticker aus L_1 heraus.

(5) Mische L_1 wieder mit L_0
↳ DNA-Lösung L

(6) Erhitze L, so dass sich die Sticker $\overline{r}_1, \ldots, \overline{r}_B$ wieder von ihren Regionen ablösen.

(7) Kühle langsam ab
→ Sticker werden sich zufällig wieder an Speicherkomplexe binden.

Beachte: Nach dem Erhitzen sind $\frac{1}{2} \cdot 2^B = 2^{B-1}$ Kopien von jedem Sticker $\overline{r}_i$ $(1 \leq i \leq B)$ in der Lösung vorhanden, während 2^B viele Speicherstränge s existieren.

→ Für eine bestimmte Kopie des Speicherstrangs s und eine bestimmte Region r_i auf dieser Kopie wird sich mit Wahrscheinlichkeit $\frac{1}{2}$ ein Sticker $\overline{r}_i$ an diese

Region finden.

- Wir generieren also 2^B viele zufällige Bitstrings der Länge B.

- Für jeden festen Bitstring $\alpha \in \{0,1\}^B$ gilt:

Prob[α ist nach dem Abkühlen nicht in der Lösung vorhanden]

$$\approx \left(1 - \frac{1}{2^B}\right)^{2^B} \approx \frac{1}{e}$$

- $\Rightarrow$ $\approx 63\%$ aller 2^B vielen Bitstrings der Länge B werden am Ende in der Lösung vorhanden sein.

Dieser Prozentsatz kann erhöht werden, indem man am Anfang mit $M > 2^B$ vielen Kopien von s beginnt.

g) Am Ende der Berechnung muß überprüft werden, ob $L \neq \emptyset$ für eine DNA-Lösung L gilt

Lösung: Fluoreszente Markierung der Speicherstränge

Um einen Binärstring in der Output-DNA-Lösung zu ermitteln, kann eine binäre Baumdekodierung verwendet werden:

Sei L = ouput-Lösung

```
if L = ∅ then "Es gibt keine Lösung"
else
  w := λ
  for i := 1 to B do
     (L1, L0) = separate(L, i);
     if L0 ≠ ∅ then
        w := w0
        L := L0
     else (L1 ≠ ∅ muß dann gelten)
        w := w1
        L := L1
     endif
  endfor
endif
output(w)
```

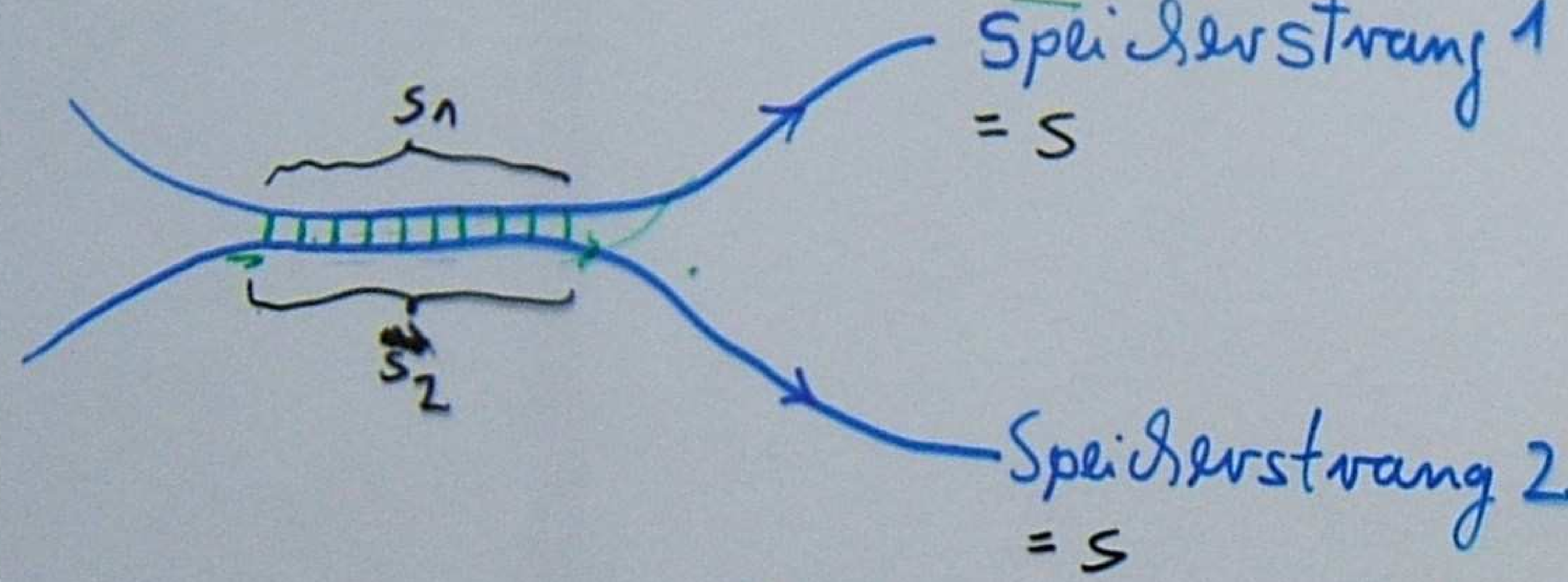
- Wie wählt man den Speicherstrang $s = t_0 v_1 t_1 v_2 \cdots t_{K-1} v_K t_K$ und die Regionen $v_1, \ldots, v_K$?

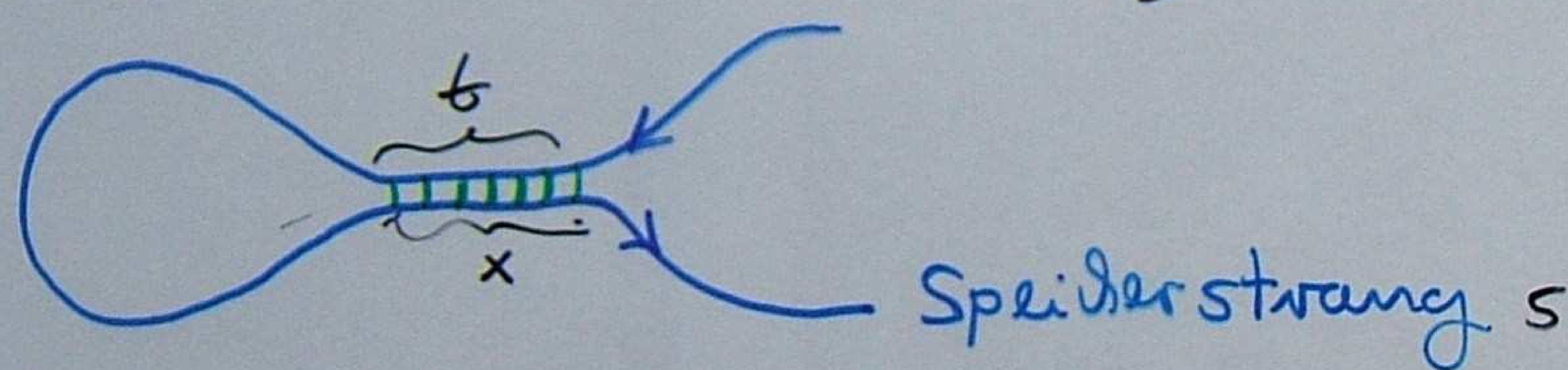
<u>Bedingung 1</u>: Sticker $\bar{v}_i$ darf sich nur an seine spezifische Region binden:

$s = \alpha \beta \gamma$ mit $|\beta| = M = |v_1| = \cdots = |v_K|$ und $\alpha \neq t_0 v_1 t_1 \cdots v_{i-1} t_{i-1}$

$\Rightarrow$ v_i und β unterscheiden sich an mindestens D_1 vielen Positionen.

Bedingung 2: Speicherstränge sollten keine Sekundärstrukturen bilden.





- Sei $s = u_1 s_1 v_1 = u_2 s_2 v_2$ mit $|s_1| = |s_2| \geq d$
  → s_1 und $\bar{s}_2$ unterscheiden sich an mindestens D_2 vielen Positionen
- Sei $s = u t v x y$ mit $|t| = |x| \geq d$
  → t und $\bar{x}^{rev}$ unterscheiden sich an mindestens D_2 vielen Positionen

Eine Attacke auf DES mittels des Sticker-Modells

- Für einen ~~Geh~~ Geheimtext Quelltext (plaintext) $w \in \{0,1\}^{64}$ und einen Schlüssel $K \in \{0,1\}^{56}$ bezeichnen wir mit $C_K(w)$ den mittels DES generierten ~~Geheimtext~~ ciphertext.
- Wir wollen im folgenden eine plaintext-ciphertext Attacke auf DES durchführen:

Gegeben: Ein Paar $(u,v) \in \{0,1\}^{64} \times \{0,1\}^{64}$
(u = plaintext, v = zugehöriger ciphertext)

Gesucht: Ein Schlüssel $K \in \{0,1\}^{56}$ mit $v = C_K(u)$

Beachte: Es wird i.A. mehrere (wenn auch nicht sehr viele) Schlüssel K mit $v = C_K(m)$ geben.

Vorgehen im Sticker-Modell

Ein Speicherstrang s besteht aus 11560 Basen und unterteilt sich in 578 Regionen $r_1, \ldots, r_{578}$ der Länge 20 (keine Zwischenräume) zwischen den Regionen.

$\Rightarrow$ Ein Speicherkomplex repräsentiert einen Bitstring aus $\{0,1\}^{578}$

- Regionen $r_1, \ldots, r_{56}$ speichern einen Schlüssel
- Regionen $r_{57}, \ldots, r_{578}$ speichern Zwischenergebnisse.
- Am Ende der Berechnung wird Region $r_{57} \ldots r_{120}$ (repräsentiert 64 Bits) den ciphertext $C_K(m)$ enthalten, wobei K der Schlüssel ist, der durch $r_1 \ldots r_{56}$ repräsentiert wird und m unser fester plaintext ist.

Gesamtstrategie

(1) $L := \text{init}(578, 56)$
(L enthält danach 2^{56} Speicherkomplexe)

(2) Auf jedem Speicherkomplex $m \in L$ berechne $C_K(m)$, wobei K der Schlüssel ist, der in den Regionen $r_1, \ldots, r_{56}$ von m gespeichert ist.

(3) Filtere aus L diejenigen Speicherkomplexe heraus, die in den Regionen $v_{57} \dots v_{120}$ den gegebenen ciphertext v kodieren.
Jeder der so herausgefilterten Speicherkomplexe speichert in Region $v_1 \dots v_{56}$ einen Schlüssel K mit $v = c_K(m)$.

Die Hauptarbeit liegt in Schritt (2):

- In jeder der 16 Runden von DES wird ein neuer 32-Bit Block $b_1, \dots, b_{16}$ generiert (der zweite 32-Bit Block wird lediglich kopiert).

- $b_1, \dots, b_{14}$ werden in den Regionen $v_{121}, \dots, v_{568}$ gespeichert, während b_{15} und b_{16} in den Regionen $v_{57}, \dots, v_{120}$ gespeichert werden (b_{15}, b_{16} bilden zusammen den ciphertext)

$v_{569}, \dots, v_{578}$ werden als Zwischenspeicher genutzt. Dieser muß während der Berechnung wiederholt auf 0…0 gesetzt werden.
↪ clear-Operation

Die Regionen $v_1, \dots, v_{568}$ werden nie auf 0 gesetzt.

- Berechnung des Blocks b_i:

$$b_i = b_{i-2} \oplus \underbrace{S\Big(\underbrace{E(\overset{32\text{ Bit}}{b_{i-1}})}_{48\text{ Bit}} \oplus k_i\Big)}_{32\text{ Bit}}$$

Ein 4-Bit Block von b_i berechnet sich aus

- 6 Bits vom b_{i-1}
- 6 Bits vom K_i
- 4 Bits vom b_{i-2}

(1) 6 Bits vom b_{i-1} werden $\oplus$-verknüpft mit 6 Bits vom K_i

↳ Resultat wird in den Regionen $r_{569}, \ldots, r_{574}$ gespeichert.

(2) Eine S-Box wird auf die 6 in $r_{569}, \ldots, r_{574}$ gespeicherten Bits angewendet

↳ Resultat wird in den Regionen $r_{575}, \ldots, r_{578}$ gespeichert

(3) Die in $r_{575}, \ldots, r_{578}$ gespeicherten 4 Bits werden $\oplus$-verknüpft mit 4 Bits aus b_{i-2}

↳ Resultat = 4-Bit Block von b_i

<u>Beachte</u>:

Welche Bits vom b_{i-1}, K_i (in Schritt (1)) und b_{i-2} (in Schritt (3)) gelesen werden müssen, d.h. welche Regionen der Speicherstränge gelesen werden (mittels separate-Operation), hängt nur ab, von

- der aktuellen Rundenzahl i und
- der Nummer des 4-Bit Blocks von b_i (zwischen 1 .... 8) der gerade berechnet wird.

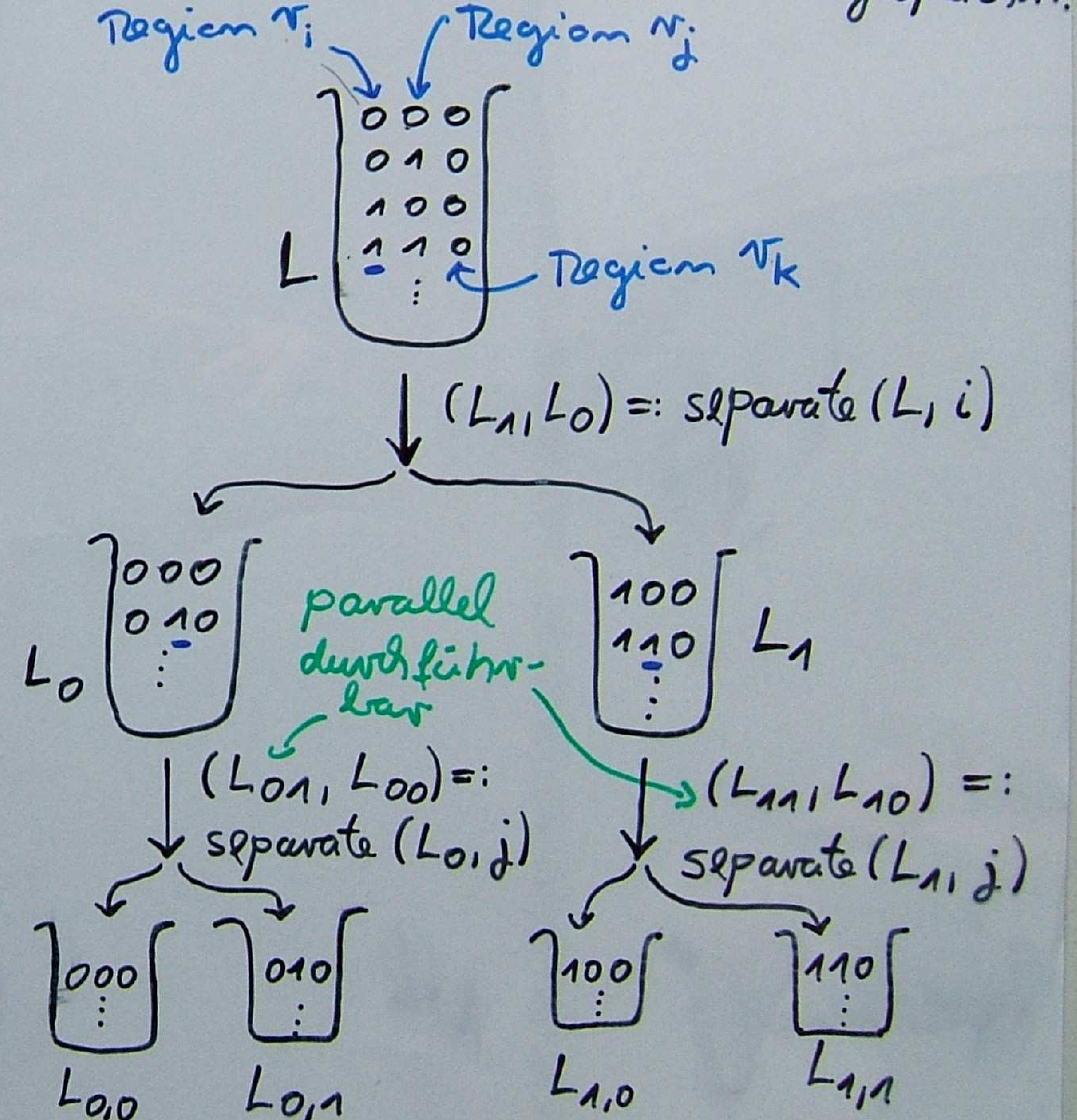
(4) Regionen $r_{569}, \ldots, r_{578}$ werden mittels clear-Operation auf 0 gesetzt.

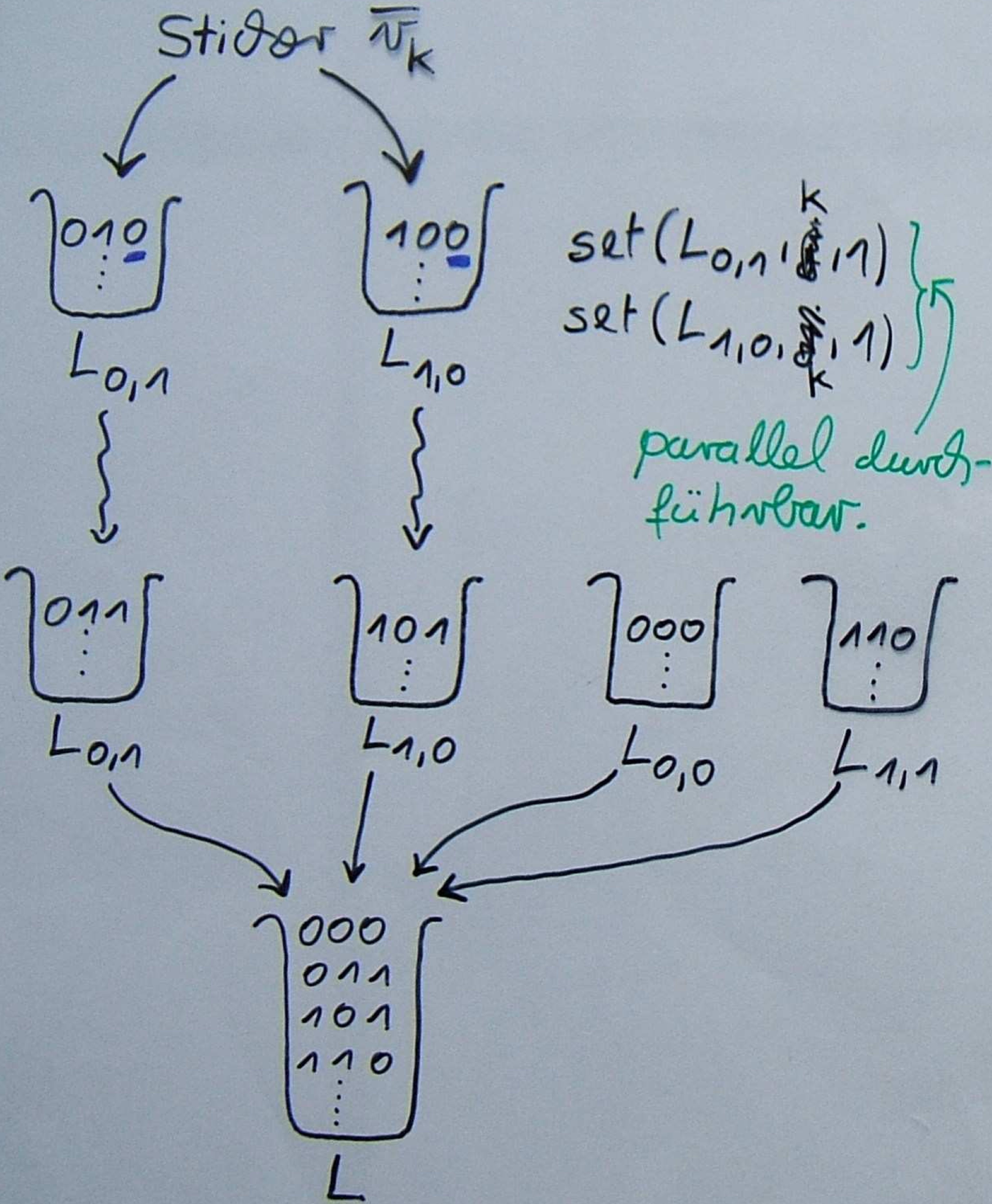
Gesamte Berechnung des ciphertexts $c_K(m)$ kann somit als eine sequentielle Abfolge folgender Elementarschritte angesehen werden:

(1) Berechnung des XOR von zwei Bits, Resultat wird in bestimmter Region gespeichert.

(2) Anwendung einer S-Box auf 6 Bits, Resultat wird in 4 Regionen gespeichert.

- Berechnung des XOR von zwei Bits, welche in den Regionen r_i und r_j gespeichert sind, Resultat wird in r_k abgespeichert.





Insgesamt: 4 Schritte

- <u>Allgemein</u>: Die Berechnung einer Booleschen Funktion $f: \{0,1\}^n \to \{0,1\}^m$ benötigt $n+m+1$ Schritte:
  - n (parallele) separate-Operationen
  - m (parallele) set-Operationen
  - Am Ende wird alles in einem Schritt zusammengeschüttet.

↪ Berechnung einer S-Box benötigt 11 Schritte, hierfür müssen 32 separate-Operationen parallel durchführbar sein.

- Insgesamt sind zur Berechnung des ciphertexts $c_K(m)$
  - 6655 parallele Schritte (siehe Übung)
  - und 1271 Reagenzgläser

notwendig.

- Nachdem auf jedem der 2^{56} initialen Speicherkomplexe der ciphertext $c_K(m)$ berechnet wurde, müssen diejenigen Speicherkomplexe herausgefiltert werden, die in den Regionen $r_{57}, \dots, r_{120}$ den gegebenen ciphertext v enthalten.

↳ 64 (~~parallele~~) separate-Operationen

- Jeder der so herausgefilterten Speicherkomplexe speichert in den Regionen $r_1, \dots, r_{56}$ einen Schlüssel K mit $r_K(m) = v$.

Imsgesamt: 6655 + 64 = 6719 Schritte.

Zeitbedarf für eine parallele Operation	Gesamte Laufzeit
1 Tag	18 Jahre
1 Stunde	9 Monate
1 Minute	5 Tage
1 Sekunde	2 Stunden

Fehleranalyse

Fehlerquellen:

- Speicherstränge können brechen
- Sticker können sich ablösen
- Speicherkomplexe können an den Wänden vom Reagenzgläsern haften bleiben.

⋮

- Für jeden der 6719 Arbeitsschritte kann eine <u>Fehlerrate</u> definiert werden:

$$\frac{\#\text{Speicherkomplexe, die im Arbeitsschritt fehlerhaft behandelt werden}}{\text{Gesamtzahl an Speicherkomplexen}}$$

- Sei E = Maximale Fehlerrate aller 6719 Arbeitsschritte.

- Für unser gegebenes Paar (u,v) (u = plaintext, v = ciphertext) heißt jeder Schlüssel K mit $C_K(u) = v$ ein <u>Gewinnschlüssel</u>

Beachte: Es kann mehrere Gewinnschlüssel geben.
Sei K_W im folgenden ein fester Gewinnschlüssel

- Angenommen wir beginnen die Berechnung mit $X \cdot 2^{56}$ (leeren) Speichersträngen $s_1, s_2, \ldots, s_{X \cdot 2^{56}}$.

- Für $1 \leq i \leq X \cdot 2^{56}$ sei $\mathcal{E}_i$ das folgende Ereignis:
  - Auf Speicherstrang s_i wird nach der init-Operation in den Regionen $r_1, \ldots, r_{56}$ der Gewinnschlüssel K_W repräsentiert, und
  - der resultierende Speicherkomplex durchläuft alle 6719 Arbeitsschritte fehlerfrei.

$$\longrightarrow \operatorname{Prob}[\mathcal{E}_i] = \underbrace{\frac{1}{2^{56}} \cdot \underbrace{(1-E)^{6719}}_{S}}_{=:\,p}$$

Sei Zufallsvariable W = Anzahl der Speicherkomplexe, auf denen am Ende der Geheimschlüssel K_W repräsentiert wird, und die keinen Fehler durchlaufen

$$\hookrightarrow \text{Prob}[W=m] = \binom{2^{56}\cdot X}{m} \cdot p^m \cdot (1-p)^{2^{56}\cdot X - m}$$

(Binomialverteilung)

$$p = \frac{S}{2^{56}}$$

$$\rightarrow \text{Prob}[W=0] = (1-p)^{2^{56}\cdot X} =$$

$$= \left(1 - \frac{S}{2^{56}}\right)^{\overset{\uparrow\ \text{sehr groß}}{2^{56}/S}\ \cdot X\cdot S} \approx e^{-X\cdot S}$$

$\longrightarrow$ Prob[mind. ein Speicherkomplex repräsentiert am Ende K_W und hat keinen Fehler durchlaufen]

$$\approx \underline{1 - e^{-X\cdot S}}$$

- Ein <u>Störkomplex</u> ist ein Speicherkomplex, auf dem kein Geheimschlüssel gespeichert ist, der aber trotzdem (auf Grund von Fehlern) am Ende in der Lösung auftaucht.

Wieviele Störkomplexe hat man am Ende?

(*) <u>Annahme</u>: Zu Beginn wird auf jedem Speicherstring s_i $(1 \leq i \leq X\cdot 2^{56})$ jeder Schlüssel K mit Wahrscheinlichkeit $\frac{1}{2^{56}}$ repräsentiert.

stimmt eigentlich nicht.

$\longrightarrow$

Nach der Berechnung des ciphertexts $C_K(m)$ (für unseren festen plaintext m) wird jeder mögliche ciphertext mit Wahrscheinlichkeit $\frac{1}{2^{64}}$ in den Regionen

$r_{57}, \dots, r_{120}$ des Speicherstrangs S_i repräsentiert:

$$\overbrace{\boxed{r_1} \boxed{\dots} \boxed{r_{56}}}^{\text{zufälliger Schlüssel } K} \boxed{r_{57}} \boxed{\dots} \boxed{r_{120}}$$

$\Big\}$ DES-Berechnung mittels des (fehlerbehafteten) Sticker-Algorithmus

$$\boxed{r_1} \boxed{\dots} \boxed{r_{56}} \underbrace{\boxed{r_{57}} \boxed{\dots} \boxed{r_{120}}}_{\text{zufälliger ciphertext}}$$

Stimmt nicht wirklich: Für einen festen plaintext m ist der ciphertext $C_K(m)$ kein Zufallsstring, falls K zufällig gewählt wird (hätte man aber gerne bei kryptographischen Verfahren).

$(1 \leq i \leq x \cdot 2^{56})$

Für einen Speicherkomplex M_i sei

$H(M_i)$ = Hamming-Distanz zwischen unserem festen Input-ciphertext v und dem auf M_i (in Regionen $r_{57}, \dots, r_{120}$) repräsentierten ciphertext.

Annahme (*) $\longrightarrow$

$$\text{Prob}\left[H(M_i) = l\right] = \binom{64}{l} \cdot \left(\frac{1}{2}\right)^{64-l} \cdot \left(\frac{1}{2}\right)^{l}$$

$$= \binom{64}{l} \cdot \frac{1}{2^{64}}$$

Unter der Voraussetzung $H(M_i) = l$ gilt:

Prob[M_i überlebt die finalen 64 separate-Operationen]

$$= (1-E)^{64-l} \cdot E^{l}$$

Für die Bit-Positionen, wo v mit dem ciphertext auf M_i übereinstimmt, muß die entsprechende separate-Operation M_i korrekt behandeln.

Für die Bit-Positionen, wo v sich vom ciphertext auf M_i unterscheidet, muß die entsprechende separate-Operation M_i fehlerhaft behandeln.

→ Prob[M_i ist am Ende ein Störkomplex]

$$\leq \sum_{l=0}^{64} \text{Prob}[H(M_i)=l] \cdot (1-E)^{64-l} \cdot E^{l}$$

Erwartungswert von W:

$$W_i = \begin{cases} 1 & \text{falls am Ende auf Speicherstrang } s_i \text{ der Gewinnschlüssel } K_W \text{ repräsentiert wird. und } s_i \text{ keinen Fehler durchläuft} \\ 0 & \text{sonst} \end{cases}$$

Erwartungswert

$$\rightarrow E[W_i] = p = \frac{S}{2^{56}} \qquad W = \sum_{i=1}^{X \cdot 2^{56}} W_i$$

$$\underline{E[W]} = \sum_{i=1}^{X \cdot 2^{56}} E[W_i] = X \cdot 2^{56} \cdot \frac{S}{2^{56}}$$

$$= \underline{X \cdot S}$$

Beispiel: $S = (1-E)^{6719} = \frac{1}{X}$

d.h $X = \frac{1}{S}$

$$\longrightarrow \text{Prob}[W > 0] \approx 1 - e^{-X \cdot S} = 1 - \frac{1}{e}$$

$$\approx 0{,}63$$

$$E[W] = 1$$

$$= \sum_{l=0}^{64} \binom{64}{l} \cdot \frac{1}{2^{64}} \cdot (1-E)^{64-l} \cdot E^{l}$$

$$= \frac{1}{2^{64}}$$

→ E[Anzahl der Störkomplexe] ≤

$$X \cdot 2^{56} \cdot \frac{1}{2^{64}} = \frac{X}{256}$$

Fehlerrate E	0	10^{-4}	10^{-3}	10^{-2}
Faktor X um Prob[W>0] ≥ 0,63 zu erreichen	1.	2	830	$2{,}1 \cdot 10^{29}$
Notwendige DNA-Menge in Gramm	0,7	1,4	580	$1{,}5 \cdot 10^{29}$ (≈ 23 Erdmassen)
Anzahl der Störkomplexe	0,004	0,008	3,2	$8{,}3 \cdot 10^{26}$