Quantencomputer

Rupert Hölzl
Universität der Bundeswehr München

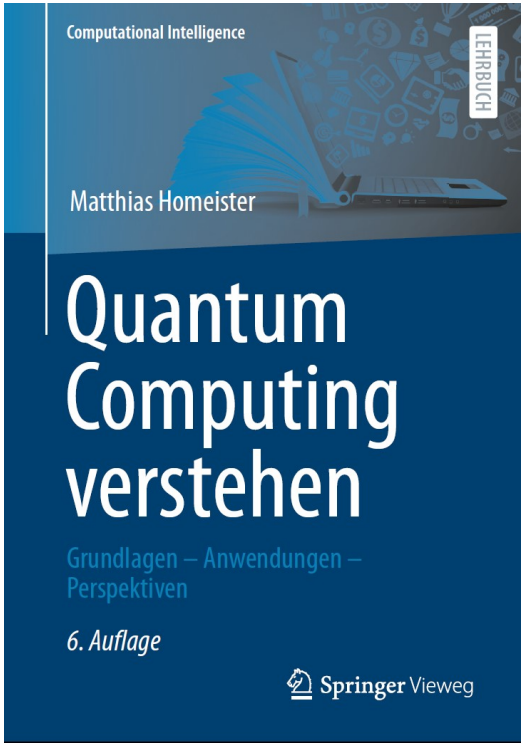
Zuletzt überarbeitet am 2. Dezember 2024

1 Vorbemerkungen

Caveat & Ziele

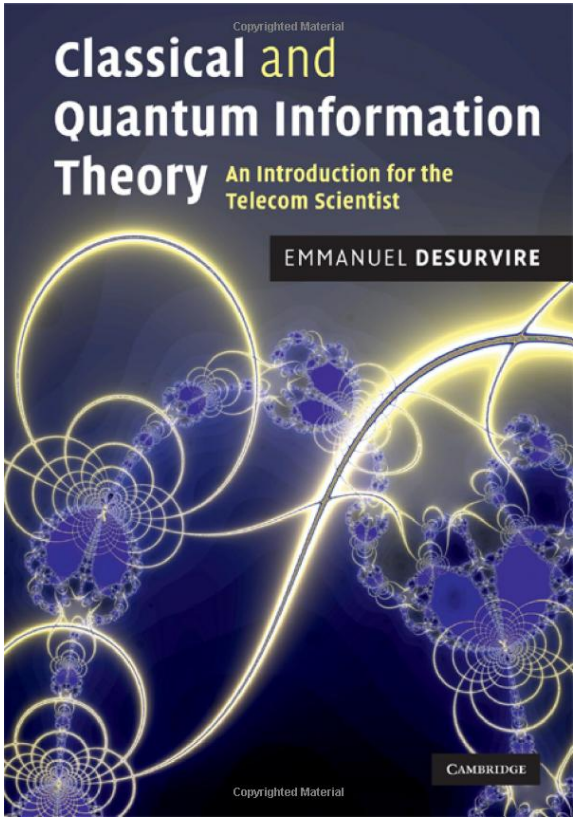
1. Wegen des geringen Stundenumfangs beschränken wir uns auf das Wesentliche, bei minimalem mathematischem Formalismus.
2. **Hauptziel:** Der Shor-Algorithmus,
   - mit dem ein Quantencomputer Zahlen wesentlich schneller faktorisieren kann als ein klassischer Computer, was ihn wichtig macht für die kryptographische Praxis.
3. **Hauptziel:** Der Grover-Algorithmus,
   - ein Suchalgorithmus für Datenbanken, der zum Finden eines bestimmten Elements viel weniger oft in der Datenbank nachsehen muss, als ein klassisches Verfahren dies müsste.

Literatur



1. Wir orientieren uns an diesem Buch, da es
   - kompakt, verständlich und deutschsprachig ist, und
   - immer nur so viel Mathematik einführt, wie gerade erforderlich.
2. **Wir behandeln:** Kapitel 1–3, 4.4–4.5, 8, 7, 6.1–6.3
3. **E-Book:** *opac.unibw.de/search?bvnr=BV042788236*

Literatur



1. Dieses Buch ist
   - umfangreich und detailliert,
   - sehr klar geschrieben,
   - gut zum Nachschlagen von Details.
2. **Ausleihbar:** *opac.unibw.de/search?bvnr=BV035293323*

Theorie...

1. Quantencomputer sind zunächst nur ein theoretisches Modell von Rechenmaschinen.
2. Thema der Vorlesung ist, was sich mit diesen theoretischen Maschinen anfangen lässt, besonders im Vergleich mit klassischen theoretischen Rechenmaschinen (Turingmaschinen, Registermaschinen, Schaltkreise,...).
3. Thema wird *nicht* sein, ob und wie sich dieses theoretische Rechenmodell in praktischen Maschinen implementieren lässt.
4. Aber dennoch es gibt praktische Gründe, warum uns Quantencomputer interessieren können.

... und Praxis

1. So haben klassische theoretische Rechenmaschinen wie Registermaschinen, Schaltkreise, etc. eine Entsprechung in der Praxis: Sie bilden theoretisch ab, was reale Computer können.
2. Angenommen, wir könnten zeigen,
   - dass das theoretische Modell der Quantencomputer *ebenfalls* eine Entsprechung in der Praxis hat, und
   - dass theoretische Quantencomputer *tatsächlich mehr* leisten können als klassische theoretische Rechenmaschinen,

   dann wäre dies auch für die Praxis relevant.
3. Der erste Punkt ist eine Aufgabe für Physiker und Ingenieure.
   - Es gibt physikalische Experimente, die zeigen, dass sich zumindest *im Prinzip* auch in der Praxis Maschinen bauen lassen, die sich wie die theoretischen Quantencomputer verhalten.
   - Ob es aber jemals technisch möglich sein wird, sie in so großem Maßstab zu bauen, dass sie praktisch nützlich werden, ist offen.
4. Der zweite Punkt ist Thema dieser Vorlesung.

2 Die Quantenwelt

Schrödingers Katze

1. In einer von der Außenwelt isolierten Kiste befinden sich
   - eine Katze,
   - ein Atom eines radioaktiven Isotops, das während des Experiments mit 50% Wahrscheinlichkeit zerfällt, und
   - ein Geigerzähler, der bei Zerfall ein tödliches Gift freisetzt.
2. Ob die Katze lebt, hängt davon ab, ob es einen Zerfall gab.
3. Hier spielen Quanteneffekte mit hinein: *Von außen betrachtet* befindet sich das Atom in einer Überlagerung der Zustände „zerfallen“ und „nicht zerfallen“.
4. Der Mechanismus übertragt dies vom Atom auf die makroskopische Katze: *von außen betrachtet* befindet sie sich in einer Überlagerung der Zustände „tot“ und „lebendig“.

Schrödingers Katze

1. Sobald wir die Kiste öffnen, „muss sich die Welt entscheiden“: das Atom muss sich darauf festlegen, ob es zerfallen ist, und in der Konsequenz die Katze, ob sie lebt.
2. Durch Aufhebung der Isolation haben wir also die Überlagerung der Zustände zerstört.
3. Der für uns wichtigste Aspekt des Gedankenexperiments: Obwohl die Zustände des Systems nicht eindeutig bestimmt sind, verhalten sich Kausalzusammenhänge wie erwartet.

Von der Katze zum Bit

1. Genauso werden sich Bits in Quantencomputern verhalten: Ihr Zustand ist eine Überlagerung der Zustände 0 und 1.
2. Auch ohne deren Wert zu kennen (sie müssen isoliert sein, wie die Kiste), können wir die Bits manipulieren (analog dazu, wie bei Schrödingers Katze der mörderische Mechanismus arbeitet) und mit ihnen rechnen.
3. Am Schluss wollen wir das Ergebnis der Rechnung natürlich auslesen. Dabei zerstören wir die Isolation. Die Überlagerung der möglichen Ergebnisse bricht zusammen. Die Rechnung muss sich für eines der möglichen Ergebnisse entscheiden. *Nur dieses bekommen wir zu sehen!*

Die Herausforderung

1. Die zentrale Herausforderung ist es, die Manipulationen so geschickt zu wählen, dass wir, obwohl wir nur einen zufällig ausgewählten Teilaspekt des Ergebnisses zu sehen bekommen, dennoch eine gewünschte (Teil-)Information erhalten.
2. Für einige wenige algorithmische Probleme hatte jemand die richtige Idee, wie dies bewerkstelligt werden kann. In diesen Fällen könnten sich Quantencomputer als nützlich erweisen.
3. Quantencomputer sind also nicht einfach besonders leistungsfähige Computer.

Die formale Version

1. **ket-Notation:** Quantenbits werden als „$|x\rangle$“ geschrieben.
2. Ein Quantenbit $|x\rangle$ kann dabei wie ein normales Bit einen von zwei *Grundzuständen* $|0\rangle$ und $|1\rangle$ annehmen.
3. Zusätzlich kann, wie auf der letzten Folie diskutiert, ein Quantenbit auch eine Überlagerung dieser beiden Grundzustände annehmen, nämlich

$$|x\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle\,,$$

   wobei $\alpha, \beta \in \mathbb{C}$ mit $|\alpha|^2 + |\beta|^2 = 1$ die *Amplituden* sind.
4. Offensichtlich kann ein Quantenbit überabzählbar viele verschiedene Zustände annehmen.

Messen eines Quantenbits

1. Sobald wir den Zustand eines Quantenbit $|x\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle$ messen, muss eines der Ergebnisse $|0\rangle$ oder $|1\rangle$ herauskommen — die Überlagerung der beiden Grundzustände kollabiert, wie bei der Katze.
2. **Frage.** Welcher der Grundzustände herauskommt ist zufällig, aber was ist die Wahrscheinlichkeitsverteilung?
3. **Antwort.** Das Ergebnis 0 kommt mit Wahrscheinlichkeit $|\alpha|^2$ heraus; das Ergebnis 1 mit Wahrscheinlichkeit $|\beta|^2$.

Mögliche Werte für α und β

1. **Beispiel.** $\alpha = \beta = 1/\sqrt{2}$.
   Dann ist die Wahrscheinlichkeit 0 und 1 zu messen jeweils $1/2$.
2. **Beispiel.** $\alpha = 1/2 = \sqrt{1/4}$ und $\beta = \sqrt{3/4}$.
   Dann ist die Wahrscheinlichkeit, 0 zu messen $1/4$,
   und die Wahrscheinlichkeit für 1 ist $3/4$.
3. **Beispiel.** $\alpha = 0$ und $\beta = 1$.
   Dann messen wir mit Sicherheit 1.
4. **Verständniskontrolle.** Neunzig unabhängige Quantenbits sind im Zustand $\frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$. Wir messen sie, und zählen die 0en und 1en. Welche Anzahlen sind am wahrscheinlichsten?

Quantenbits als Vektoren

1. Wir können $|x\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle$ auch als Vektor darstellen:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

2. $|\alpha|^2 + |\beta|^2 = 1$ erzwingt Punkte auf dem Einheits„kreis“.
3. **Achtung.** α und β sind komplexe Zahlen!

Rechnen mit Quantenbits

1. Da die Bits während der Rechnung als Überlagerungen von Grundzuständen vorliegen (können), müssen auch Rechenschritte dazu passend definiert sein:
   - Ein Grundzustand kann durch einen Rechenschritt in eine Überlagerung übergehen.
   - Falls der Zustand bereits vor Ausführen des Rechenschritts eine Überlagerung ist, so muss der Rechenschritt auf jeden der darin „enthaltenen“ Grundzustände angewendet werden.
   - Jeder einzelne davon kann wieder selbst in eine Überlagerung übergehen, usw.
2. Nach dem Ausführen mehrerer Rechenschritte kann der Zustand also die Überlagerung von Überlagerungen von Überlagerungen … von Grundzuständen sein.
3. In manchen Fällen können wir die Algorithmen aber so geschickt wählen, dass am Ende doch Grundzustände herauskommen.

Formalisierung von Rechenschritten

1. Formal sind Rechenschritte in diesem Rechenmodell *unitäre Matrizen*, d.h. Matrizen A mit der Eigenschaft

$$A^\dagger := (A^*)^T = A^{-1},$$

   mit „*“ komplexer Konjugation und „^{T}“ Transponierung.
2. **Nebeneffekt.** Jeder Rechenschritt ist umkehrbar. Ein entscheidender Unterschied zu klassischen Rechnern!
3. **Fakt.** Eine reelle Matrix A ist unitär g.d.w. $A^T = A^{-1}$.
4. **Beispiel.** Die Einheitsmatrix $I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.
5. **Verständniskontrolle.** Zeigen Sie, dass die *Hadamardmatrix* H unitär ist, wobei

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Anwendung von Rechenschritten

1. Einen Rechenschritt auf ein Quantenbit anzuwenden, bedeutet einfach, die Matrix auf die Vektordarstellung anzuwenden, also

$$|x\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle \quad \text{wird durch} \quad A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

überführt nach

$$A \cdot |x\rangle = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha a + \beta b \\ \alpha c + \beta d \end{pmatrix}.$$

2. Oft werden wir auch die ket-Notation und die Linearität verwenden, zum Beispiel:

$$\begin{aligned} A\,|x\rangle &= A(\alpha\,|0\rangle + \beta\,|1\rangle) &&= \alpha A\,|0\rangle + \beta A\,|1\rangle \\ &= \begin{pmatrix} \alpha a \\ \alpha c \end{pmatrix} + \begin{pmatrix} \beta b \\ \beta d \end{pmatrix} &&= \begin{pmatrix} \alpha a + \beta b \\ \alpha c + \beta d \end{pmatrix} \\ &= (\alpha a + \beta b)|0\rangle &&+ (\alpha c + \beta d)|1\rangle\,. \end{aligned}$$

Echter und falscher Zufall

1. Für interessante Probleme benötigen wir mehrere Bits.
2. Trotzdem können wir bereits jetzt einen Algorithmus vorstellen, der mehr kann, als ein klassischer Rechner.
3. Das Verhalten klassischer Rechner ist voll determiniert. Sie können *aus Prinzip niemals* Zufallszahlen produzieren.
4. Sie können *Pseudozufallszahlen* produzieren, das heißt Zahlen, die zufällig aussehen, wenn man nicht zu genau hinsieht.
5. Aber für praktische Anwendungen, beispielsweise bei randomisierten Algorithmen, reicht das nicht.
   - Die Beweise, das ein solcher Algorithmen seinen Zweck erfüllt, erfordern probabilistische Argumente, also „echte“ Zufälligkeit.
   - Würden Pseudozufallszahlen genügen, so könnte man die Prozedur, die diese generiert, in den Algorithmus miteinbauen, und hätte einen deterministischen Algorithmus.
6. Quantencomputer *können* echte Zufallsbits produzieren!

Quantenzufallsgenerator

1. **Algorithmus:**
   - $|x\rangle \leftarrow |0\rangle$ (Weise dem Bit $|x\rangle$ den Grundzustand $|0\rangle$ zu.)
   - $|x\rangle \leftarrow H\,|x\rangle$ (Wende die Hadamardtransformation auf $|x\rangle$ an.)
   - Miss $|x\rangle$ (Miss den Wert des Quantenbits $|x\rangle$.)
2. Der Algorithmus tut, was er soll, weil
   - die Anwendung von H auf $|0\rangle$ das Ergebnis

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\cdot\begin{pmatrix}1\\ 0\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix}=\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle$$

     produziert;
   - dies ist gerade die „50/50-Überlagerung“, das heißt mit Wahrscheinlichkeit $\left|\frac{1}{\sqrt{2}}\right|^2 = 1/2$ kommt beim Messen 0 heraus, und ebenfalls mit $\left|\frac{1}{\sqrt{2}}\right|^2 = 1/2$ das Ergebnis 1.
3. **Verständniskontrolle.** Was tut der Algorithmus, wenn wir die erste Anweisung durch „$|x\rangle \leftarrow |1\rangle$“ ersetzen?

3 Quantenregister

Quantenregister

1. Quantenregister speichern mehrere Quantenbits.
2. **Beispiel.**
   - Es seien zwei Bits gegeben:

$$\begin{aligned} |x_1\rangle &= \alpha_1 |0\rangle + \beta_1 |1\rangle \\ |x_0\rangle &= \alpha_0 |0\rangle + \beta_0 |1\rangle \end{aligned}$$

   - Dann hat das Quantenregister $|x_1\rangle |x_0\rangle$ den Zustand

$$|x_1\rangle |x_0\rangle = \alpha_1\alpha_0 |0\rangle |0\rangle + \alpha_1\beta_0 |0\rangle |1\rangle + \beta_1\alpha_0 |1\rangle |0\rangle + \beta_1\beta_0 |1\rangle |1\rangle .$$

   - Wir vereinfachen die Schreibweise und erhalten

$$|x_1x_0\rangle = \gamma_{00} |00\rangle + \gamma_{01} |01\rangle + \gamma_{10} |10\rangle + \gamma_{11} |11\rangle$$

   für geeignete $\gamma_{ij} \in \mathbb{C}$.

Quantenregister

1. **Definition.** Ein n-Bit Quantenregister $\left|x_{n-1}\ldots x_0\right\rangle$ befindet sich in überlagerten Zuständen der Form

$$\sum_{\sigma\in\{0,1\}^n} \alpha_\sigma \left|\sigma\right\rangle \quad \text{wobei} \quad \sum_{\sigma\in\{0,1\}^n} |\alpha_\sigma|^2 = 1.$$

2. Statt eines binären Strings σ schreiben wir oft die natürliche Zahl n, die von σ kodiert wird. Dann erhalten wir alternativ

$$\sum_{i=0}^{2^n-1} \alpha_i \left|i\right\rangle \quad \text{wobei} \quad \sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1.$$

3. **Verständniskontrolle.** Kann es Missverständnisse geben, wenn man „$|0\rangle$“ oder „$|1\rangle$“ schreibt, d.h. ist die Bedeutung eindeutig?
4. Wenn wir $\left|x_{n-1}\ldots x_0\right\rangle$ messen, so erhalten wir mit Wahrscheinlichkeit $|\alpha_i|^2$ das Messergebnis i.

Caveat

1. Zustände eines n-Bit-Quantenregisters sind also Vektoren in einem 2^n-dimensionalen Vektorraum. Eine Basis bei $n = 2$ ist

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \ |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \ |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \ |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

   Man muss also beim Umwandeln zwischen der ket-Darstellung und der Vektordarstellung etwas aufpassen.

2. **Beachte:** Zustände eines *klassischen* n-Bit-Registers sind Vektoren in einem Vektorraum mit nur n Dimensionen.

Rechenoperationen auf Registern

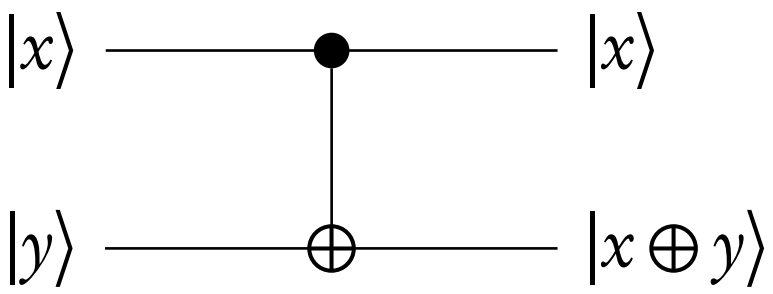
1. Um Matrizen auf Vektoren anwenden zu können, müssen die Matrizen natürlich die zu den Vektoren passende Größe haben.
2. **Also:** Rechenoperationen auf n-Bit-Quantenregistern sind $2^n \times 2^n$-Matrizen, die unitär sind.

Rechenoperationen auf Registern

1. **Beispiel.** CNOT: $|xy\rangle \mapsto |x(x \oplus y)\rangle$ mit der Matrixdarstellung

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

2. **Verständniskontrolle.** Man überzeuge sich, dass die obige Matrix tatsächlich CNOT berechnet.
3. **Darstellung als Quantenschaltkreis:**



Permutationsmatrizen

1. **Definition.** Eine Matrix, die in jeder Zeile und Spalte genau eine 1 und sonst nur 0en stehen hat, heißt Permutationsmatrix.
2. **Beispiel.** CNOT ist eine Permutationsmatrix.

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

3. **Verständniskontrolle.** Überzeugen Sie sich, dass Permutationsmatrizen gerade die Amplituden des Zustandes eines Quantenregisters *vertauschen*.
4. **Verständniskontrolle.** Zeigen Sie, dass Permutationsmatrizen unitär sind.

Das Problem von Deutsch

1. Eine Funktion $f : \{0,1\} \to \{0,1\}$ ist gegeben.
2. Wir sollen die folgenden Fälle unterscheiden:
   - $f(0) = f(1)$ „*konstant*“
   - $f(0) \neq f(1)$ „*balanciert*“

   dabei aber f nur an einer Stelle abfragen.
3. Klassisch wäre das natürlich unmöglich.
4. **Idee.**
   - Erzeuge ein Bit $|x\rangle$ als Überlagerung von $|0\rangle$ und $|1\rangle$.
   - Frage (informell gesprochen) $f(|x\rangle)$ ab.
   - Da $|x\rangle$ keinen eindeutigen Wert hat, ist uneindeutig, an welcher Stelle der Wert von f eigentlich abgefragt wird.
   - Also wird f an beiden Stellen gleichzeitig abgefragt, und das Ergebnis ist eine Überlagerung von $|f(0)\rangle$ und $|f(1)\rangle$.
   - Wir müssen noch überlegen, wie wir die Lösung auslesen.

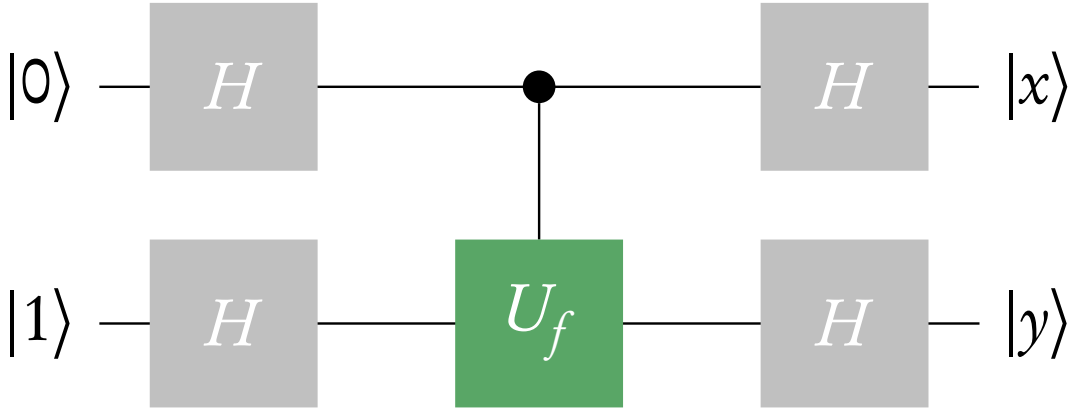
Das Problem von Deutsch

1. Zunächst müssen wir das Abfragen von f formalisieren.
2. Alle Operationen müssen unitär, also umkehrbar sein.
3. Es gibt also keine Operation, die einfach $f(x)$ in ein Bit schreibt.
   (Denn dabei würde das vorher dort stehende Bit überschrieben; die Information über seinen alten Wert wäre verloren, was die Invertierung unmöglich macht.)
4. Wir brauchen deshalb ein *invertierbares Orakel* wie folgt:

$$U_f \colon |xy\rangle \mapsto |x(y \oplus f(x))\rangle$$

5. **Verständniskontrolle.** Welche Größe hat die Matrix zu U_f?
6. **Verständniskontrolle.** Überzeugen Sie sich, dass U_f für beliebiges f invertierbar ist.

Algorithmus



1. $|xy\rangle \leftarrow |01\rangle$
2. $|xy\rangle \leftarrow H\,|x\rangle\, H\,|y\rangle$
3. $|xy\rangle \leftarrow U_f\,|xy\rangle$
4. $|xy\rangle \leftarrow H\,|x\rangle\, H\,|y\rangle$
5. Miss $|xy\rangle$;
   falls $xy = 01$, gib aus „konstant“,
   falls $xy = 11$, gib aus „balanciert“.

Analyse des Algorithmus

$$|01\rangle$$

$$\Big\downarrow H$$

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \cdot \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

$$\Big\downarrow U_f$$

$$\frac{1}{2}(|0(0 \oplus f(0))\rangle - |0(1 \oplus f(0))\rangle + |1(0 \oplus f(1))\rangle - |1(1 \oplus f(1))\rangle)$$
$$= \frac{1}{2}(|0\rangle(|f(0)\rangle - |1 \oplus f(0)\rangle) + |1\rangle(|f(1)\rangle - |1 \oplus f(1)\rangle))$$
$$= \frac{1}{2}((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle)(|0\rangle - |1\rangle)$$

Vorsicht: Amplituden dürfen wir kommutieren, die Reihenfolgen der Bits natürlich nicht!

$$\begin{array}{ll} \text{Falls } f \text{ konstant:} & \pm\frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \\ \text{Falls } f \text{ balanciert:} & \pm\frac{1}{2}(|0\rangle - |1\rangle)(|0\rangle - |1\rangle) \end{array}$$

$$\Big\downarrow H$$

$$\begin{array}{ll} \text{Falls } f \text{ konstant:} & \pm|0\rangle|1\rangle \\ \text{Falls } f \text{ balanciert:} & \pm|1\rangle|1\rangle \end{array}$$

Das Tensorprodukt

1. Auf den bisherigen Folien haben wir bereits informell Quantenschaltkreise gebildet und benutzt.
2. Hierbei haben wir Matrizen auf Vektoren angewendet, die eigentlich „zu klein“ sind, zum Beispiel die 2×2-Matrix H auf das erste Bit eines Quantenregisters, dessen Zustände eigentlich von 4-dimensionalen Vektoren repräsentiert werden.
3. Wir erklären nun, warum wir dies dürfen.

Das Tensorprodukt

1. Seien V_1, V_2 Vektorräume über dem selben Körper mit Basen
$$e_0, \ldots, e_{m-1} \qquad \text{bzw.} \qquad f_0, \ldots, f_{n-1}.$$
2. **Definition.** Das Tensorprodukt $V_1 \otimes V_2$ ist ein (ganz bestimmter) $(m \cdot n)$-dimensionaler Vektorraum; seine Basis schreiben wir
$$\begin{matrix} e_0 \otimes f_0 & \cdots & e_0 \otimes f_{n-1} \\ \vdots & \ddots & \vdots \\ e_{m-1} \otimes f_0 & \cdots & e_{m-1} \otimes f_{n-1}. \end{matrix}$$
3. **Bemerkung.** „$\otimes$“ ist keine Rechenoperation, sondern nur eine Schreibweise. Hier werden ganz neue Elemente $e_i \otimes f_j$ definiert, und so benannt, weil sie aus e_i und f_j hervorgegangen sind.
4. Das Tensorprodukt übertragt sich von Basiselementen auf Vektoren. Für $v_1 = \sum_{i=0}^{m-1} \alpha_i e_i$ und $v_2 = \sum_{j=0}^{n-1} \beta_i f_i$ gelte
$$v_1 \otimes v_2 = \left(\sum_{i=0}^{m-1} \alpha_i e_i \right) \otimes \left(\sum_{j=0}^{n-1} \beta_i f_i \right) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \alpha_i \beta_j (e_i \otimes f_j).$$

Das Tensorprodukt

1. Die Darstellung $\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \alpha_i \beta_j (e_i \otimes f_j)$ sieht unseren Quantenregistern sehr ähnlich. Zum Beispiel gilt

$$|0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle = |0\rangle\,|1\rangle\,.$$

2. **Folgerung.** Was wir bisher durch „Hintereinanderschreiben“ von Quantenbits dargestellt haben, ist in Wahrheit ein Tensorprodukt aus den Vektoren, die diesen Bits entsprechen.
3. **Verständniskontrolle.** Überzeugen Sie sich, dass das Tensorprodukt nicht kommutativ ist.

   (Dies erklärt auch, warum wir die Bitreihenfolge auf Folie 32 nicht vertauschen durften, die Positionen der Amplituden der Zustände aber schon.)

Das Tensorprodukt

1. Passend zum Tensorprodukt auf Vektorräumen und Vektoren kann man auch ein Tensorprodukt auf Matrizen definieren.
2. **Definition.** Sei B eine beliebige Matrix und

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$$

   Dann ist das Tensorprodukt von A und B definiert als

$$A \otimes B = \begin{pmatrix} a_{11} \cdot B & \dots & a_{1n} \cdot B \\ \vdots & \ddots & \vdots \\ a_{m1} \cdot B & \dots & a_{mn} \cdot B \end{pmatrix}$$

3. **Allgemein gilt:** $(A \otimes B)(u \otimes v) = Au \otimes Bv$ ($\rightarrow$ Übungsaufgabe).

Das Tensorprodukt

1. **Beispiel.** $I_2 \otimes I_2 = \begin{pmatrix} I_2 & 0_2 \\ 0_2 & I_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = I_4$

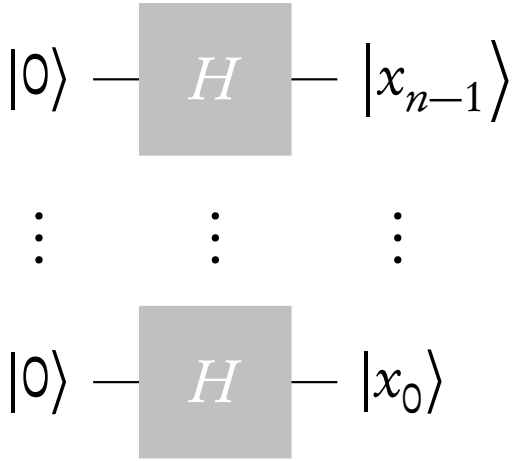
2. **Beispiel.** $H \otimes H = \frac{1}{\sqrt{2}} \begin{pmatrix} H & H \\ H & -H \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$

   Verständniskontrolle. Dies ist äquivalent zur einzelnen Anwendung von H auf welche Bits eines Registers $|xy\rangle$?

3. **Beispiel.** $I_2 \otimes H = \begin{pmatrix} H & 0_2 \\ 0_2 & H \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}$

   Verständniskontrolle. Dies ist äquivalent zur einzelnen Anwendung von H auf welche Bits eines Registers $|xy\rangle$?

n-Bit-Quantenzufallsgenerator



1. **Definition.** Schreibe H_n für $\bigotimes_{i=1}^{n} H$.
2. **Algorithmus für n-Bit-Quantenzufallsgenerator:**
   - $|x_{n-1} \dots x_0\rangle \leftarrow |0 \dots 0\rangle$
   - $|x_{n-1} \dots x_0\rangle \leftarrow H_n |x_{x-1} \dots x_0\rangle$
   - Miss $|x\rangle$
3. **Verständniskontrolle.** Welcher Zustand wurde nach dem zweiten Schritt erzeugt?
4. **Verständniskontrolle.** Welche Messergebnisse sind möglich, und mit welchen Wahrscheinlichkeiten treten sie auf?

4 Messungen

Allgemeinere Messungen

1. Bisher haben wir nur bezüglich der Standardbasen der Form $\{|\sigma\rangle : \sigma \in \{0,1\}^n\}$ und immer alle Bits auf einmal gemessen.
2. Man kann auch bezüglich anderer Orthonormalbasen messen, beispielsweise bezüglich der *Hadamardbasis*

$$|+\rangle := H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |-\rangle := H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

3. Man kann die Bits eines Registers auch nur teilweise messen.
   - Die Überlagerung der Zustände bricht dann nur für die gemessenen Bits zusammen.
   - Wir erfahren dann nur den Wert dieser Bits.
   - Die ungemessenen Bits bleiben (im Allgemeinen) überlagert, und wir können mit ihnen weiterrechnen, und sie später messen.
4. **Verständniskontrolle.** Wie lässt sich der Algorithmus auf Folie 31 durch Messung bezüglich $\{|+\rangle, |-\rangle\}$ vereinfachen?

Noch allgemeiner: Messungen als Projektionen

1. Sei $E_1, \ldots, E_k$ eine disjunkte Zerlegung einer Orthonormalbasis $\{e_0, \ldots, e_{n-1}\}$ des Zustandsraumes eines Quantenregisters.
2. Die Folge von Unterräumen $\mathrm{Span}(E_1), \ldots, \mathrm{Span}(E_k)$ heißt dann eine *Observable*. Für $1 \leq i \leq k$ sei $|\varphi_i\rangle$ die Projektion eines Zustandes $|\varphi\rangle$ auf $\mathrm{Span}(E_i)$. Dann gilt $|\varphi\rangle = \sum_{i=1}^{k} |\varphi_i\rangle$.
3. Sei $\| \, |\psi\rangle \, \|$ die *Norm* $\sqrt{\sum_{i=0}^{n-1} |\alpha_i|^2}$ von $|\psi\rangle = \sum_{i=0}^{n-1} \alpha_i |i\rangle$.
4. Messen wir $|\varphi\rangle$ bezüglich obiger Observable, so geht das Quantenregister mit Wahrscheinlichkeit $\| \, |\varphi_i\rangle \, \|^2$ in den Zustand $\frac{|\varphi_i\rangle}{\| \, |\varphi_i\rangle \, \|}$ über, und dann erhalten wir die Information i.
5. Die Normierung mit $\frac{1}{\| \, |\varphi_i\rangle \, \|}$ sorgt dafür, dass für den neuen Zustand die Summe der Quadrate der Amplituden wieder 1 ist.

Noch allgemeiner: Messungen als Projektionen

1. Die Arten, wie wir bisher gemessen haben, sind nur Spezialfälle des Begriffs der Observablen.
2. **Verständniskontrolle.** Geben Sie die Observablen an für
   - die Messung eines Quantenbits bezüglich der Standardbasis,
   - die Messung eines Quantenbits bezüglich der Hadamardbasis,
   - die Messung aller Bits eines 2-Bit-Registers mit der Standardbasis,
   - die Messung des zweiten Bits eines 2-Bit-Registers.
3. **Achtung!**
   - Wenn wir beispielsweise ein Bit bezüglich der Hadamardbasis messen, so geht es in einen der Zustände $|+\rangle$ und $|-\rangle$ über. Diese sind selbst Überlagerungen der Grundzustände $|0\rangle$ und $|1\rangle$.
   - Dennoch ist hier eine Überlagerung zerstört worden: Während das Bit sich vor der Messung in einem 2-dimensionalen Vektorraum bewegen konnte, ist es durch die Projektion auf einen 1-dimensionalen eingeschränkt worden.

5 Verschränkung

Verschränkte Zustände

1. Betrachte folgenden Programmablauf:

$$
\begin{array}{c}
|00\rangle \\
\Big\downarrow H \otimes I_2 \\
\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \\
\Big\downarrow \text{CNOT} \\
\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)
\end{array}
$$

2. Eine Messung des ersten Bits liefert je mit Wahrscheinlichkeit $1/2$ die Ergebnisse 0 und 1.
3. Angenommen wir messen 0, dann ist der Folgezustand $|00\rangle$.
   (Denn wenn man $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ auf $\mathrm{Span}(\{|00\rangle, |01\rangle\})$ projiziert, erhält man $\frac{1}{\sqrt{2}}|00\rangle$, was durch Normierung mit dem Faktor $1/||\frac{1}{\sqrt{2}}|00\rangle|| = \sqrt{2}$ zu $|00\rangle$ wird.)
4. Analog, falls wir 1 messen, ist der Folgezustand $|11\rangle$.
5. **Also:** Auch das Ergebnis der Messung des *zweiten* Bits ist durch die erste Messung bereits festgelegt worden.

Verschränkte Zustände

1. Nun stelle man sich vor, man produziere zwei Bits wie auf der letzten Folie, messe sie aber nicht, sondern teile sie zwischen zwei Personen Alice und Bob auf.
2. Solange keiner von ihnen sein Bit gemessen hat, beträgt für jedes davon die Wahrscheinlichkeit, dass es den Wert $|0\rangle$ hat, $\frac{1}{2}$.
3. Sobald aber Alice ihr Bit misst, ist sofort auch der Wert von Bobs Bit festgelegt, und umgekehrt.
4. Die Überlagerung bricht durch Alices Messung also nicht nur bei ihrem eigenen Bit zusammen, sondern auch bei Bobs.
5. Dabei können Alice und Bob beliebig weit voneinander entfernt sein, der Effekt tritt trotzdem ein.
6. Es tritt also eine „spukhafte Fernwirkung“ (Einstein) ein, ohne Medium, das die Information über den Wert des Bits überträgt.

Verschränkte Zustände

1. Das gleiche gilt natürlich auch für alle anderen Zustände, bei denen ein Bit das andere eindeutig bestimmt.
2. Diese Zustände heißen *Bell-Zustände* und sind

$$\begin{aligned}
\Phi^+ &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \\
\Phi^- &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle), \\
\Psi^+ &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), \\
\Psi^- &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle).
\end{aligned}$$

3. **Verständniskontrolle.** Man erzeugt Ψ^-. Alice erhält das erste Bit, Bob das zweite. Bob misst sein Bit als erstes und erhält 0. Dann misst Alice ihr Bit. Welches Ergebnis erhält sie?

Verschränkte Zustände

1. Mathematisch beschreibt man dieses Phänomen so:

   Definition. Sei $|\varphi\rangle$ der Zustand eines n-Bit-Quantenregisters. $|\varphi\rangle$ heißt *unverschränkt* falls es ein Tensorprodukt der Zustände der einzelnen Bits ist, also $|\varphi\rangle = |\varphi_{n-1}\rangle \otimes \cdots \otimes |\varphi_0\rangle$. Falls es keine solche Darstellung gibt heißt $|\varphi\rangle$ *verschränkt*.

2. **Verständniskontrolle.** Überlegen Sie, wie man zwei ganze n-Bit-Register miteinander verschränken kann. Benutzen Sie eine bijektive Funktion $f\colon \{0,1\}^n \to \{0,1\}^n$ und den Operator

$$\begin{array}{rccl} U\colon & \{0,1\}^{2n} & \longrightarrow & \{0,1\}^{2n}, \\ & |xy\rangle & \longmapsto & |x(y \oplus f(x))\rangle\,. \end{array}$$

6 Das Problem von Deutsch-Josza

Die Hadamard-Transformation mehrerer Bits

$$\text{Sei } |x\rangle := \left|x_{n-1}\dots x_0\right\rangle \text{ ein Grundzustand.}$$

$$\Big\downarrow H_n$$

$$\frac{1}{\sqrt{2^n}}\left(|0\rangle + (-1)^{x_{n-1}}|1\rangle\right)\cdots\left(|0\rangle + (-1)^{x_0}|1\rangle\right)$$

$$= \frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{x\odot y}|y\rangle$$

(Hier bezeichne „$\odot$“ das binäre Skalarprodukt.)

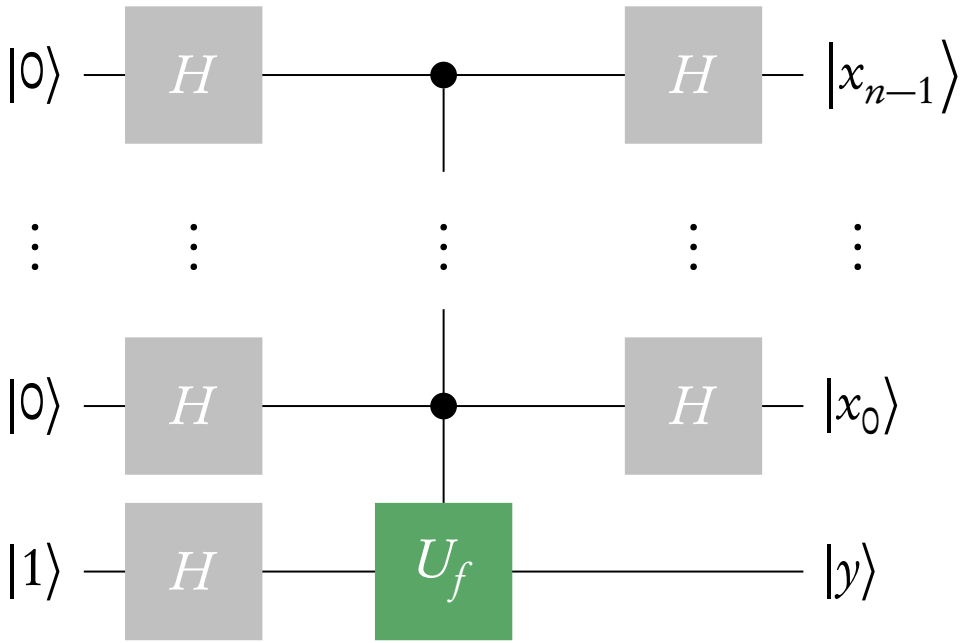
1. Die Gleichheit gilt, weil, damit ein $|y\rangle$ ein negatives Vorzeichen bekommen kann, zwei Voraussetzungen erforderlich sind:
   - y muss 1en enthalten, denn nur dann entsteht y durch das Ausmultiplizieren von Faktoren mit negativen Vorzeichen; und
   - von den Positionen i dieser 1en müssen *ungerade viele* die Eigenschaft $x_i = 1$ haben; sonst heben sich die (-1)en auf.
2. **Verständniskontrolle.** Wieviele Minusse gibt es in $H_6\,|001000\rangle$?

Das Problem von Deutsch-Josza

1. Eine Funktion $f : \{0,1\}^n \to \{0,1\}$ ist gegeben.
2. Wir sollen die folgenden Fälle unterscheiden:
   - $f(x) = f(y)$ für alle $x, y \in \{0,1\}^n$ „*konstant*“
   - $|f^{-1}(0)| = |f^{-1}(1)| = 2^{n-1}$ „*balanciert*“

   dabei aber ein reversibles Orakel für f nur einmal abfragen.
3. Klassisch wäre das natürlich unmöglich.
4. Es handelt sich bei diesem Problem offensichtlich um eine Verallgemeinerung des Problems von Deutsch.

Algorithmus



1. $|x_{n-1}\ldots x_0 y\rangle \leftarrow |0\ldots 01\rangle$
2. $|xy\rangle \leftarrow H_{n+1}|xy\rangle$
3. $|xy\rangle \leftarrow U_f|xy\rangle$
4. $|xy\rangle \leftarrow (H_n \otimes I_2)|x\rangle|y\rangle$
5. Miss; falls $x = 0\ldots 0$, gib „konstant“ aus, sonst „balanciert“.

Analyse des Algorithmus

$$|0\ldots01\rangle$$

$$\Big\downarrow H_{n+1}$$

$$\left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\right)\cdot\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$$

$$\Big\downarrow U_f$$

$$\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}\left(|x\rangle\cdot\frac{1}{\sqrt{2}}(|0\oplus f(x)\rangle-|1\oplus f(x)\rangle)\right)$$
$$=\left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle\right)\cdot\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$$

$$\Big\downarrow H_n\otimes I_2$$

$$\sum_{y\in\{0,1\}^n}\left(\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot y}|y\rangle\right)\cdot\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle),$$

wobei wir die Betrachtung von Folie 49 benutzt haben.

Analyse des Algorithmus

$$|0\ldots01\rangle$$

$$\Big\downarrow H_{n+1}$$

$$\left(\tfrac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\right)\cdot\tfrac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$$

$$\Big\downarrow U_f$$

$$\tfrac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}\left(|x\rangle\cdot\tfrac{1}{\sqrt{2}}(|0\oplus f(x)\rangle-|1\oplus f(x)\rangle)\right)$$
$$=\left(\tfrac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle\right)\cdot\tfrac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$$

$$\Big\downarrow H_n\otimes I_2$$

$$\sum_{y\in\{0,1\}^n}\left(\tfrac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot y}|y\rangle\right)\cdot\tfrac{1}{\sqrt{2}}(|0\rangle-|1\rangle),$$

wobei wir die Betrachtung von Folie 49 benutzt haben.

Wir wollen zeigen, dass der Algorithmus korrekt ist, weil,
falls f konstant ist, $|0\ldots0\rangle$ die Amplitude ±1 und,
falls f balanciert ist, $|0\ldots0\rangle$ die Amplitude 0 hat.

Analyse des Algorithmus

$$|0\ldots01\rangle$$

$$\Big\downarrow H_{n+1}$$

$$\left(\tfrac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\right)\cdot\tfrac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$$

$$\Big\downarrow U_f$$

$$\tfrac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}\left(|x\rangle\cdot\tfrac{1}{\sqrt{2}}(|0\oplus f(x)\rangle-|1\oplus f(x)\rangle)\right)$$
$$=\left(\tfrac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle\right)\cdot\tfrac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$$

$$\Big\downarrow H_n\otimes I_2$$

$$\sum_{y\in\{0,1\}^n}\left(\tfrac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot y}|y\rangle\right)\cdot\tfrac{1}{\sqrt{2}}(|0\rangle-|1\rangle),$$

wobei wir die Betrachtung von Folie 49 benutzt haben.

Wir wollen zeigen, dass der Algorithmus korrekt ist, weil,
falls f konstant ist, $|0\ldots0\rangle$ die Amplitude ±1 und,
falls f balanciert ist, $|0\ldots0\rangle$ die Amplitude 0 hat.
Dazu betrachten wir den grünen Term für $y=0\ldots0$.

Analyse des Algorithmus

Falls f konstant: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}\,|0\ldots 0\rangle$

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}\,|0\ldots 0\rangle$

Analyse des Algorithmus

Falls f konstant: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}\,|0\ldots 0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1.$

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}\,|0\ldots 0\rangle$

Analyse des Algorithmus

Falls f konstant: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\,|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1.$

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}\,|0\ldots0\rangle$

Analyse des Algorithmus

Falls f konstant: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\,|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}\,|0\ldots0\rangle$

Analyse des Algorithmus

Falls f konstant: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}|0\ldots0\rangle$

Analyse des Algorithmus

Falls f konstant: $\pm\frac{1}{2^n}\sum_{x\in\{0,1\}^n}|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0}=1$.
2. $f(x)$ hat den gleichen Wert für alle x.

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}|0\ldots0\rangle$

Analyse des Algorithmus

Falls f konstant: $\pm\frac{1}{2^n}\sum_{x\in\{0,1\}^n}|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0}=1$.
2. $f(x)$ hat den gleichen Wert für alle x.

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}|0\ldots0\rangle$

Analyse des Algorithmus

Falls f konstant: $\pm\frac{1}{2^n}\sum_{x\in\{0,1\}^n}|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0}=1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n|=2^n$.

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}|0\ldots0\rangle$

Analyse des Algorithmus

Falls f konstant: $\pm\frac{1}{2^n}2^n\,|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n| = 2^n$.

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}\,|0\ldots0\rangle$

Analyse des Algorithmus

Falls f konstant: $\pm\frac{1}{2^n}2^n\,|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n| = 2^n$.

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}\,|0\ldots0\rangle$

Analyse des Algorithmus

Falls f konstant: $\pm|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n| = 2^n$.

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}|0\ldots0\rangle$

Analyse des Algorithmus

Falls f konstant: $\pm|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n| = 2^n$.

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\odot 0\ldots 0}|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.

Analyse des Algorithmus

Falls f konstant: $\pm|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n| = 2^n$.

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.

Analyse des Algorithmus

Falls f konstant: $\pm|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n| = 2^n$.

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x) = 0$ für eine Hälfte der x, und $f(x) = 1$ für die andere.

Analyse des Algorithmus

Falls f konstant: $\pm|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n| = 2^n$.

Falls f balanciert: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x) = 0$ für eine Hälfte der x, und $f(x) = 1$ für die andere.

Analyse des Algorithmus

Falls f konstant: $\pm|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n| = 2^n$.

Falls f balanciert: $\frac{1}{2^n}\cdot 0\cdot|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x) = 0$ für eine Hälfte der x, und $f(x) = 1$ für die andere.

Analyse des Algorithmus

Falls f konstant: $\pm|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n| = 2^n$.

Falls f balanciert: $\frac{1}{2^n}\cdot 0\cdot|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x) = 0$ für eine Hälfte der x, und $f(x) = 1$ für die andere.

Analyse des Algorithmus

Falls f konstant: $\pm|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n| = 2^n$.

Falls f balanciert: $0\cdot|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x) = 0$ für eine Hälfte der x, und $f(x) = 1$ für die andere.

Analyse des Algorithmus

Falls f konstant: $\pm|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n| = 2^n$.

Falls f balanciert: $0\cdot|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x) = 0$ für eine Hälfte der x, und $f(x) = 1$ für die andere.

Analyse des Algorithmus

Falls f konstant: $\pm|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x)$ hat den gleichen Wert für alle x.
3. $|\{0,1\}^n| = 2^n$.

Falls f balanciert: $0\cdot|0\ldots0\rangle$

1. $(-1)^{x\odot 0\ldots 0} = 1$.
2. $f(x) = 0$ für eine Hälfte der x, und $f(x) = 1$ für die andere. $\square$

7 Komplexität von Quantenalgorithmen

Erinnerung: Uniformität und Nichtuniformität

1. Das Konzept der Uniformität spielt eine zentrale Rolle in der klassischen Komplexitäts- und Berechenbarkeitstheorie.
2. Uniforme Berechnungen werden meist durch Turingmaschinen dargestellt, nichtuniforme durch Schaltkreisfamilien.
3. Nichtuniformität kann beliebige Information kodieren:

   Verständniskontrolle. Beschreiben Sie eine Schaltkreisfamilie, die Wörter x akzeptiert g.d.w. $|x|$ im Halteproblem ist.
4. Sobald wir aber uniforme Generierbarkeit der Schaltkreisfamilie fordern, verschwindet die Unterscheidung, und es gilt sogar:

   Eine Funktion ist in Polynomialzeit berechenbar g.d.w. sie von polynomiell uniformen Schaltkreisen berechnet wird.
5. Wir werden uniforme Familien von *Quanten*schaltkreisen betrachten. Dann wird eine analoge Aussage gelten.

Komplexität von Familien von Quantenschaltkreisen

1. Sei $(C_i)_{i\in\mathbb{N}}$ eine Familie von Quantenschaltkreisen, das heißt für alle i ist C_i ein Quantenschaltkreis mit i Eingabebits.
2. **Definition.** Wir sagen $(C_i)_{i\in\mathbb{N}}$ ist *polynomiell uniform* wenn es einen klassischen Polynomialzeit-Algorithmus gibt, der bei Eingabe i eine polynomiell große Darstellung von C_i ausgibt.
3. Wir werden den Algorithmus nie explizit angeben, sondern stattdessen das gleichförmige Schema beschreiben, nach dem Schaltkreise beliebiger Größe in der Familie aufgebaut sind.
4. **Dann gilt:**
   Eine Funktion ist mit Quantenalgorithmen genau dann in Polynomialzeit berechenbar, wenn sie von polynomiell uniformen Quantenschaltkreisen berechnet wird.
5. **Beachte:** Quantenpolynomialzeit müssten wir eigentlich erst über Quanten-Turingmaschinen definieren. Wir ersparen uns dies, und werden nur mit Quantenschaltkreisen arbeiten.

Die Größe von Quantenschaltkreisen

1. Produkte von unitären Matrizen sind unitär.
2. Dass heißt, auch eine lange Folge von Rechenoperationen lässt sich **als eine einzige Rechenoperation zusammenfassen.**
3. So macht es keinen Sinn, von der Größe eines Quantenschaltkreises zu sprechen.
4. Also definieren wir „einfache Rechenschritte“ als solche an denen höchstens 3 Bits beteiligt sind.
5. Die *Größe* eines Quantenschaltkreises ist dann die Anzahl der einfachen Rechenschritten, die er beinhaltet.

8 Das No-Cloning-Theorem

Das komplexe Skalarprodukt

1. **Definition.** Für zwei Zustandsvektoren

$$|x\rangle = \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{n-1} \end{pmatrix} \qquad \text{und} \qquad |y\rangle = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_{n-1} \end{pmatrix}$$

   definieren wir das komplexe Skalarprodukt als

$$\langle x|y\rangle = \alpha_0^* \beta_0 + \cdots + \alpha_{n-1}^* \beta_{n-1}.$$

2. **Achtung:** Nicht verwechseln mit „$\odot$“, dem binären Skalarprodukt auf $\{0,1\}^n$, das wir bereits benutzt haben.

Das komplexe Skalarprodukt

1. **Proposition.** Das komplexe Skalarprodukt ist invariant unter unitären Transformationen U, das heißt

$$\langle Ux|Uy\rangle = \langle x|y\rangle$$

für alle Zustandsvektoren $|x\rangle$ und $|y\rangle$.

Beweis. → Übungsaufgabe.

2. **Proposition.** Das komplexe Skalarprodukt verträgt sich mit dem Tensorprodukt, das heißt

$$\langle x \otimes y|z \otimes w\rangle = \langle x|z\rangle \cdot \langle y|w\rangle$$

für alle Zustandsvektoren $|x\rangle$, $|y\rangle$, $|z\rangle$ und $|w\rangle$.

Beweis. → Übungsaufgabe.

Das No-Cloning-Theorem

1. **Satz.** Sei $|s\rangle \in \mathbb{C}^n$ mit $||s|| = 1$ fest gewählt. Es existiert kein unitäres U so dass für alle $|\varphi\rangle \in \mathbb{C}^n$ gilt $U(|\varphi\rangle\,|s\rangle) = |\varphi\rangle\,|\varphi\rangle$.
2. **Intuition.** Quantenbits lassen sich nicht fehlerfrei kopieren.
3. **Beweis.** Nimm an, ein solches U existiere. Da das Skalarprodukt invariant unter unitären Operationen ist, gilt für $|\varphi\rangle, |\psi\rangle \in \mathbb{C}^n$

$$\langle \varphi \otimes \varphi | \psi \otimes \psi \rangle = \langle U(\varphi \otimes s) | U(\psi \otimes s) \rangle = \langle \varphi \otimes s | \psi \otimes s \rangle .$$

   Da sich Skalar- und Tensorprodukt vertragen folgt

$$\langle \varphi | \psi \rangle \langle \varphi | \psi \rangle = \langle \varphi | \psi \rangle \langle s | s \rangle = \langle \varphi | \psi \rangle \, ||s||^2 = \langle \varphi | \psi \rangle .$$

   Also ist $\langle \varphi | \psi \rangle \in \{0, 1\}$, das heißt entweder $\varphi = \psi$ oder $\varphi \perp \psi$. □
4. **Intuition.** Nur für orthogonale Zustände kann gelten, dass sie sich von ein und der selben Prozedur korrekt kopieren lassen.

9 Simulation klassischer Berechnungen

Klassische und Quantenberechnungen

1. Berechnungen auf klassischen Schaltkreisen müssen nicht umkehrbar sein, solche auf Quantenschaltkreisen aber schon.
2. Bei den bis jetzt betrachteten Operationen wie U_f haben wir Hilfsbits benutzt, um die Ausgangsbits wiederherzustellen.
3. Nun zeigen wir allgemein, dass sich jede beliebige klassische Berechnung durch einen Quantenschaltkreis simulieren lässt.
4. Die Schaltkreise werden dabei nur linear größer werden.
5. Wir gehen in zwei Schritten vor:
   - Wir zeigen, dass *umkehrbare* klassische Schaltkreise durch Quantenschaltkreise simulierbar sind.
   - Wir zeigen, wie man jeden beliebigen klassischen Schaltkreis in einen umkehrbaren *klassischen* Schaltkreis umformen kann.

Simulation umkehrbarer klassische Berechnungen

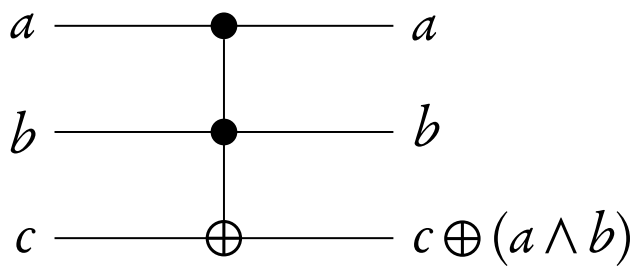
1. Eine Abbildung, die n-Bit-Binärwörter auf n-Bit-Binärwörter abbildet, ist offensichtlich genau dann umkehrbar, wenn sie eine Permutation der n-Bit-Binärwörter ist.
2. Sie wird also durch eine Permutationsmatrix beschrieben.
3. Wir haben gesehen, dass Permutationsmatrizen unitär sind.
4. Also lässt sich jede umkehrbare klassische Berechnung durch Quantenschaltkreise simulieren, indem man geeignete Permutationsmatrizen *auf Grundzustände* anwendet.

Nichtumkehrbare Berechnungen umkehrbar machen

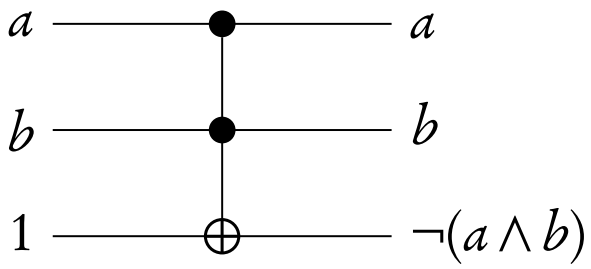
1. **Definition.** Ein klassisches Gatter G heißt *universell*, wenn sich jede Funktion von $\{0,1\}^n$ nach $\{0,1\}^m$ durch einen Schaltkreis berechnen lässt, der nur aus Kopien von G besteht.
2. **Beispiel.** Es ist allgemein bekannt, dass NAND universell ist.
3. Wir geben nun das *umkehrbare* Toffoli-Gatter an, welches in folgendem Sinne universell ist:
   - Durch geeignete Belegung der Eingangsbits kann bestimmt werden, welches logische Gatter es simulieren soll.
   - Das Ergebnis liegt dann an einem der Ausgangsbits an.
   - Die anderen Ausgangsbits sind Hilfsbits für die Umkehrbarkeit.

Das Toffoli-Gatter

1. **Definition.** Das *Toffoli-Gatter* ist wie folgt definiert:



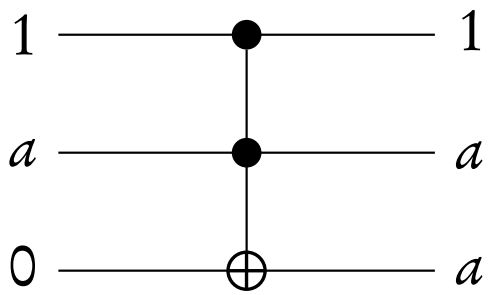
2. **Verständniskontrolle.** Warum ist es umkehrbar?
3. Das Toffoli-Gatter kann NAND-Gatter simulieren:



4. **Verständniskontrolle.** Wie lassen sich NOT, AND und XOR durch *ein einziges* Toffoli-Gatters darstellen?

Das Toffoli-Gatter

1. Das Toffoli-Gatter kann Bits kopieren:



2. Dies ist wichtig, da in Quantenschaltkreisen Leitungen sich nicht verzweigen dürfen, anders als in klassischen.
3. Wegen des No-Cloning-Theorems funktioniert das Kopieren natürlich nur für Grundzustände, aber das genügt hier.

Nichtumkehrbare Berechnungen umkehrbar machen

1. Beginne mit einem klassischen Schaltkreis C.
2. Baue einen Schaltkreis C' wie folgt:
   - Alle Eingabeleitungen von C gibt es auch in C'.
   - Alle Gatter werden durch geeignet belegte Toffoli-Gatter ersetzt.
   - Als Eingaben dienen dabei frische Kopien der jeweiligen Leitungen auf bisher unbenutzte Leitungen.
   - Diese Kopien werden ebenfalls mit Toffoli-Gattern erzeugt.
   - Beide Verwendungen von Toffoli-Gattern brauchen Hilfsbits; C' wird entsprechend um fest belegte Eingabeleitungen ergänzt.
3. Offensichtlich ist die Anzahl der Gatter in C' nur um einen konstanten multiplikativen Faktor größer als die in C.

10 Quantenkomplexitätstheorie

Bounded error quantum polynomial time

1. Mit Quantencomputern lassen sich die *selben* Funktionen berechnen wie mit klassischen Rechnern, z.B. Turingmaschinen.
2. Offen ist aber, ob sie bestimmte Funktionen *schneller* berechnen können, z.B. mehr Funktionen in Polynomialzeit.
3. **Definition.** BQP sind die Entscheidungsprobleme E für die es polynomiell uniforme Quantenschaltkreise $(C_i)_{i\in\mathbb{N}}$ gibt, so dass
   - wenn $E(x) = 0$, dann $C_{|x|}(x) = 0$ mit Wahrscheinlichkeit $\geq 3/4$,
   - wenn $E(x) = 1$, dann $C_{|x|}(x) = 1$ mit Wahrscheinlichkeit $\geq 3/4$.
4. Wir haben gesehen, dass sich klassische Rechnungen durch Quantenschaltkreise simulieren lassen. Deshalb gilt $\mathrm{P} \subseteq \mathrm{BQP}$.

Bounded error quantum polynomial time

1. Durch das Erzeugen eines überlagerten Bits und der Messung nur dieses Bits kann ein Quantenschaltkreis ein Zufallsbit erzeugen. Deshalb gilt BPP $\subseteq$ BQP.
2. Ein wichtiger Unterschied zwischen randomisierten und Quantenalgorithmen:
   - Gibt es bei einem randomisierten Algorithmus mehrere Rechenpfade, die zum gleichen Ergebnis führen, so erhöht sich dessen Eintrittswahrscheinlichkeit.
   - Bei Quantenalgorithmen kann es beispielsweise auch zwei Rechenpfade geben, die zum gleichen Ergebnis führen, aber mit unterschiedlichen Amplituden. Dann heben sich ihre Amplituden teilweise auf, was die Eintrittswahrscheinlichkeit des Ergebnisses senken kann *(destruktive Interferenz)*.

Zusammenfassung

1. **Satz.** $\mathrm{P} \subseteq \mathrm{BPP} \subseteq \mathrm{BQP}$.
2. Bei allen Inklusionen ist unbekannt, ob sie strikt sind.

RSA überwinden mit Quantencomputern

Verschlüsselung mit RSA

1. RSA ist ein asymmetrisches Verschlüsselungsverfahren.
   - Bob hat einen öffentlichen und einen geheimen Schlüssel.
   - Er gibt den öffentlichen Schlüssel allgemein bekannt.
   - Alice kann Bobs öffentlichen Schlüssel benutzen, wenn sie Bob eine Nachricht senden möchte.
   - Ebenso Charlotte mit dem *selben* öffentlichen Schlüssel.
   - Aber obwohl Alice den selben öffentlichen Schlüssel verwendet wie Charlotte, kann sie nicht die Nachrichten entschlüsseln, die Charlotte an Bob sendet; und umgekehrt.
   - Denn zum Entschlüsseln reicht der öffentliche Schlüssel nicht, man braucht den geheimen, den Bob für sich behalten hat.
2. Vorteile:
   - Es muss kein riskanter Austausch geheimer Schlüssel stattfinden.
   - Der selbe öffentliche Schlüssel kann für Nachrichten von Alice an Bob *und* für die von Charlotte an Bob benutzt werden.

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28? Er ist 2.

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28? Er ist 2.
2. **Algorithmus.** „Ziehe immer das Kleinere vom Größeren ab!“

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28? Er ist 2.
2. **Algorithmus.** „Ziehe immer das Kleinere vom Größeren ab!“
3. **Verständniskontrolle.** Warum ergibt das den g.g.T.?

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28? Er ist 2.
2. **Algorithmus.** „Ziehe immer das Kleinere vom Größeren ab!“
3. **Verständniskontrolle.** Warum ergibt das den g.g.T.?
4. **Abgekürzte rekursive Formulierung.** O.B.d.A. sei $a > b$.

   Euklid(a, b)
   {
   Falls $b = 0$, gib a zurück;
   sonst gib Euklid$(b, a \bmod b)$ zurück.
   }

Erinnerung: Euklidischer Algorithmus

1. Was ist der größte gemeinsame Teiler von 6 und 28? Er ist 2.
2. **Algorithmus.** „Ziehe immer das Kleinere vom Größeren ab!“
3. **Verständniskontrolle.** Warum ergibt das den g.g.T.?
4. **Abgekürzte rekursive Formulierung.** O.B.d.A. sei $a > b$.

   Euklid(a, b)
   {
   Falls $b = 0$, gib a zurück;
   sonst gib Euklid$(b, a \bmod b)$ zurück.
   }
5. Wegen $a \bmod b \leq a/2$ finden $O(\log a)$ rekursive Aufrufe statt.
6. Division erfordert $O((\log a)^2)$, die Gesamtlaufzeit ist $O((\log a)^3)$.

Erweiterter Euklidischer Algorithmus

1. Zusätzlich wollen wir x und y, so dass $ax + by = \mathrm{ggT}(a,b)$ gilt.
2. **Algorithmus.** O.B.d.A. sei $a > b$.

   ```
   ErweiterterEuklid(a, b)
   {
      Falls b = 0, gib (a, 1, 0) zurück; sonst
      {
         (d', x', y') := ErweiterterEuklid(b, a mod b);
         (d, x, y) := (d', y', x' − (a div b) · y');
         gib (d, x, y) zurück.
      }
   }
   ```

3. **Korrektheit per Induktion.** Induktionsanfang $b = 0$ ist klar.

   Nimm an, der Algorithmus ist korrekt für $(b, a \bmod b)$. Wegen

   $$d' = bx' + (\underbrace{a - b \cdot (a \operatorname{div} b)}_{a \bmod b})y' = ay' + b(x' - (a \operatorname{div} b)y') = ax + by = d$$

   und $\mathrm{ggT}(a,b) = \mathrm{ggT}(b, a \bmod b)$ folgt $ax + by = \mathrm{ggT}(a,b)$. $\square$

Kleiner Satz von Fermat

1. **Satz.** Sei p prim und $a \neq 0$ eine natürliche Zahl mit $p \nmid a$. Dann

$$a^{p-1} \equiv 1 \bmod p.$$

Chinesischer Restsatz

1. **Satz.** Seien $n_1, \ldots, n_k$ paarweise teilerfremde natürliche Zahlen. Seien $a_1, \ldots, a_k$ ganze Zahlen. Dann gibt es eine ganze Zahl x mit

$$x \equiv a_1 \bmod n_1, \ \ldots, \ x \equiv a_k \bmod n_k,$$

und für jede andere Zahl x' mit dieser Eigenschaft gilt

$$x \equiv x' \bmod (n_1 \cdot \ldots \cdot n_k).$$

2. **Korollar.** $a \equiv b \bmod n_1 \ \wedge \ a \equiv b \bmod n_2 \ \Longrightarrow \ a \equiv b \bmod n_1 n_2$.
Verständniskontrolle. Wie folgt das aus dem Satz?

Chinesischer Restsatz

1. **Satz.** Seien $n_1, \ldots, n_k$ paarweise teilerfremde natürliche Zahlen. Seien $a_1, \ldots, a_k$ ganze Zahlen. Dann gibt es eine ganze Zahl x mit

$$x \equiv a_1 \bmod n_1, \ \ldots, \ x \equiv a_k \bmod n_k,$$

und für jede andere Zahl x' mit dieser Eigenschaft gilt

$$x \equiv x' \bmod (n_1 \cdot \ldots \cdot n_k).$$

2. **Korollar.** $a \equiv b \bmod n_1 \ \wedge \ a \equiv b \bmod n_2 \implies a \equiv b \bmod n_1 n_2$.
Beweis. Die Gleichungen

$$x \equiv b \bmod n_1 \quad \text{und} \quad x \equiv b \bmod n_2$$

sind für $x = b$ trivialerweise und für $x = a$ nach Annahme erfüllt. Aus dem Satz folgt $a \equiv b \bmod n_1 n_2$. □

Bobs Schlüsselgenerierung

1. Wähle zwei sehr große, zufällige Primzahlen $q_1 \neq q_2$.
   Die Zahlen werden zufällig gewählt und dann auf Primheit überprüft.
2. Berechne $n = q_1 \cdot q_2$ und $\varphi(n) = (q_1 - 1)(q_2 - 1)$.
3. Wähle ein e mit $\mathrm{ggT}(\varphi(n), e) = 1$.
   Verständniskontrolle. Warum muss e ungerade sein?
4. Löse die Gleichung $e \cdot d \equiv 1 \bmod \varphi(n)$ nach d.
   Verwende den erweiterten Euklidischen Algorithmus mit Laufzeit $O((\log n)^3)$ um Zahlen k und d zu finden, so dass $e \cdot d + \varphi(n) \cdot k = 1 = \mathrm{ggT}(e, \varphi(n))$ gilt.
5. Der öffentliche Schlüssel ist (e, n) und der geheime ist d.

Verschlüsselung und Entschlüsselung

1. Um eine Nachricht m an Bob zu verschlüsseln, berechnet Alice

$$\widetilde{m} = m^e \bmod n.$$

2. Um die Nachricht wieder zu entschlüsseln, berechnet Bob

$$\widetilde{m}^d \bmod n.$$

3. Das liefert wieder m, denn wegen $ed \equiv 1 \bmod (q_1-1)(q_2-1)$ gibt es ein k mit $ed = 1 + k(q_1-1)(q_2-1)$ und dann gilt

$$\begin{aligned} \widetilde{m}^d &= m^{1+k(q_1-1)(q_2-1)} &&= m \cdot (m^{q_1-1})^{k(q_2-1)} \\ &\equiv m \cdot 1^{k(q_2-1)} \bmod q_1 &&\equiv m \bmod q_1 \end{aligned}$$

nach den kleinen Satz von Fermat.

4. Genauso gilt $\widetilde{m}^d \equiv m \bmod q_2$ und mit dem Korollar zum Chinesischen Restsatz folgt $\widetilde{m}^d \equiv m \bmod n$.

Sicherheit der Verschlüsselung

1. Die Sicherheit von RSA ist nicht mathematisch bewiesen.
2. Aber man geht davon aus, dass es gegenüber klassischen Berechnungsverfahren sicher ist.
3. Wir diskutieren einige mögliche Angriffswege durch eine Lauscherin Eve, und woran sie scheitern.

Angriff durch Faktorisierung

1. Kennt Eve die Faktoren q_1 und q_2 von n, kann sie die Schlüsselgenerierung durch Bob nachahmen und kennt d.
2. Festzustellen, ob eine Zahl prim ist, liegt in P.
3. Faktorisierung liegt trivialerweise in NP.

   (Damit diese Aussage Sinn ergibt, muss aus Faktorisierung streng genommen zunächst einmal ein Entscheidungsproblem definiert werden; siehe z.B. Lance Fortnows Blog.)
4. Man geht davon aus, dass Faktorisierung nicht in P liegt.
5. Man glaubt aber auch nicht, dass sie NP-vollständig ist.
6. Solche Probleme heißen NP-*intermediate*.
7. Natürlich ist das unbewiesen; sogar ob $\mathrm{P} \neq \mathrm{NP}$ ist unbekannt.

Angriff über $\varphi(n)$

1. $\varphi(n)$ genügt, um die Schlüsselgenerierung nachzuahmen.
2. Es ist aber nicht leichter, $\varphi(n)$ zu bestimmen, als q_1 und q_2:
   - Wegen $\varphi(n) = (q_1 - 1)(q_2 - 1) = n - (q_1 + q_2) + 1$ kennt man $q_1 + q_2$ wenn man n und $\varphi(n)$ kennt.
   - Dann kennt man auch $q_1 - q_2$ wegen

$$\begin{aligned}(q_1 - q_2)^2 &= q_1^2 - 2q_1q_2 + q_2^2 \\ &= (q_1 + q_2)^2 - 4q_1q_2 \\ &= (q_1 + q_2)^2 - 4n.\end{aligned}$$

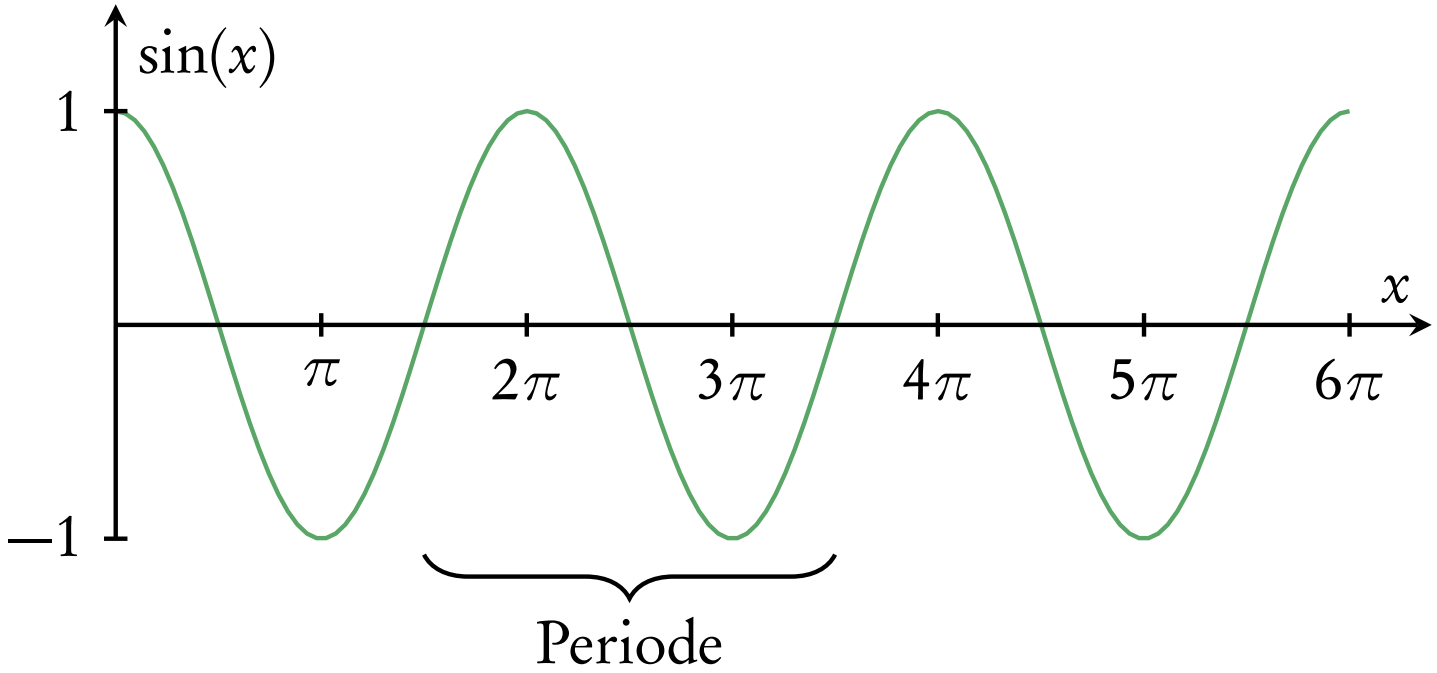
   - Dann kennt man q_2 wegen $2q_2 = (q_1 + q_2) - (q_1 - q_2)$.
   - Dann kennt man auch q_1.
3. $\varphi(n)$ zu finden ist also gleich schwer, wie q_1q_2 zu faktorisieren.

Andere Angriffe

1. Es könnte auch ganz andere Angriffe geben, die ohne Faktorisierung auskommen, jedoch wurde trotz jahrzehntelanger Überlegungen keiner gefunden.
2. Man geht also davon aus, dass RSA sicher ist, wenn der Angreifer klassische Rechner benutzt.
3. Wir werden aber zeigen, dass es randomisierte Quantenalgorithmen gibt, die schnell und mit hoher Wahrscheinlichkeit Zahlen faktorisieren können.
4. RSA würde also unsicher sobald Quantencomputer in praxistauglichem Maßstab gebaut werden können.

12 Faktorisierung durch Periodenbestimmung

Perioden von Funktionen



1. **Definition.** Eine Funktion f heißt *periodisch* falls es ein $p > 0$ gibt mit $f(x+p) = f(x)$ für alle x. Das kleinste solche p heißt *Periode* von f.
2. **Beispiel.** Die Periode von $x \mapsto \sin x$ ist 2π.
3. **Verständniskontrolle.** Was ist die Periode von $x \mapsto 3^x \bmod 10$?
4. Wir werden sehen, dass die Kenntnis von Perioden bestimmter Funktionen uns bei der Faktorisierung von Zahlen hilft.

Faktorisierung von n durch Periodenbestimmung

1. Falls n gerade ist, sehen wir das sofort an der Binärdarstellung und haben 2 als Teiler. Nimm also o.B.d.A. an, n sei ungerade.
2. Wir bestimmen zufällig ein $1 < a < n$ und berechnen mit dem Euklidschen Algorithmus den $\mathrm{ggT}(a, n)$.
3. Falls $\mathrm{ggT}(a, n) \neq 1$ haben wir sofort einen Teiler von n.
4. In unserem Fall $n = q_1 q_2$ ist dies aber extrem unwahrscheinlich.

   Verständniskontrolle. Warum? Wann tritt dieser Fall ein?
5. Wie gehen wir also im Fall $\mathrm{ggT}(a, n) = 1$ weiter vor?

Faktorisierung von n durch Periodenbestimmung

1. Nimm an, wir hätten ein Verfahren, das für ein $1 < a < n$ mit $\mathrm{ggT}(a,n) = 1$ die Periode p von $x \mapsto a^x \bmod n$ bestimmt.
2. Dann gilt $a^p \equiv 1 \bmod n$, also $a^p = 1 + kn$ für geeignetes k.
3. Nimm an, p sei gerade. Dann gilt

$$kn = a^p - 1 = (a^{p/2} - 1)(a^{p/2} + 1),$$

   also ist n mit mindestens einem der Faktoren nicht teilerfremd.
4. **Hoffnung:** $\mathrm{ggT}(a^{p/2} - 1, n)$ oder $\mathrm{ggT}(a^{p/2} + 1, n)$ teilt n.
5. **Problem:** Wir müssen ausschließen, dass n die Faktoren teilt.
6. **Verständniskontrolle.** Warum gilt $n \nmid a^{p/2} - 1$?
7. Es bleibt also der problematische Fall $n \mid a^{p/2} + 1$.

Faktorisierung von n durch Periodenbestimmung

1. **Tatsache aus der Zahlentheorie.** Sei n ungerade und keine Primpotenz. Dann gilt mindestens für die Hälfte der Zahlen $0 \leq a < n$ mit der Eigenschaft $\mathrm{ggT}(a, n) = 1$, dass
   - die Periode p der Funktion $x \mapsto a^x \bmod n$ gerade ist, und
   - $n \nmid a^{p/2} + 1$.
2. **Intuition.** $n = q_1 q_2$ erfüllt die Voraussetzungen; der problematische Fall $n \mid a^{p/2} + 1$ tritt nicht allzu oft auf; und p ist relativ oft gerade, wie auf der letzten Folie angenommen.
3. Ein randomisierter Ansatz sieht also vielversprechend aus.

Laufzeitoptimierung

1. Da $p \approx n$ sein kann, kann $a^{p/2}$ relativ zur Länge $\log(n)$ einer Binärdarstellung von n sehr groß sein. Dann ist die Berechnung von $\mathrm{ggT}(a^{p/2}-1, n)$ und $\mathrm{ggT}(a^{p/2}+1, n)$ aufwändig.
2. Für ganze Zahlen a und b gilt $\mathrm{ggT}(a,b) = \mathrm{ggT}(a \bmod b, b)$.
3. Wir rechnen also statt mit $a^{p/2}$ mit $a^{p/2} \bmod n$.
4. **Algorithmus zur Berechnung von a^b mod n:**

   $x := 1;$
   solange $b \neq 0$
   {
   falls b ungerade, $x := x \cdot a \bmod n;$
   $b := b \operatorname{div} 2;$
   $a := a \cdot a \bmod n;$
   }
   gib x aus.

5. Da $a, b, x \leq n$, sind arithmetische Operationen in $O((\log n)^2)$. Mit $O(\log b)$ Schleifendurchläufen ist die Laufzeit $O((\log n)^3)$.

Zusammenfassung: Algorithmus

1. Wähle zufällig eine Zahl $1 < a < n$.
2. Falls $\mathrm{ggT}(a,n) \neq 1$, gib $\mathrm{ggT}(a,n)$ aus und brich ab.
3. Ermittle die Periode p von $x \mapsto a^x \bmod n$.
4. Falls p ungerade, gehe zu 1.
5. Berechne $y := \mathrm{ggT}((a^{p/2} - 1) \bmod n, n)$
   und $z := \mathrm{ggT}((a^{p/2} + 1) \bmod n, n)$.
6. Falls $\{y,z\} \subseteq \{1,n\}$ gehe zu 1.
7. Gib ein Element von $\{y,z\} \setminus \{1,n\}$ aus.

Analyse

1. In jedem Schleifendurchlauf hat der Algorithmus eine Erfolgswahrscheinlichkeit größer als $1/2$. Durch wiederholte Durchläufe kann sie beliebig erhöht werden.
2. Das Verfahren zur Periodenbestimmung haben wir noch nicht diskutiert; hierfür werden wir Quantenschaltkreise einsetzen.
3. Die restliche Laufzeit eines Durchlaufs ist bestimmt durch
   - die Berechnungen der größten gemeinsamen Teiler und
   - die Berechnungen von $a^{p/2} \bmod n$,

   jeweils mit Laufzeit $O((\log n)^3)$.

Zwischenfazit

1. **Satz.** Sei P ein Orakel, das zu (n,a) mit $1 < a < n$ und $\mathrm{ggT}(a,n) = 1$ die Periode der Funktion $x \mapsto a^x \bmod n$ ausgibt.

   Sei $A = \{n \in \mathbb{N} \colon n \text{ ist ungerade und keine Primpotenz}\}$.

   Dann gibt es einen randomisierten Algorithmus mit Orakel P, der bei Eingabe $n \in A$ mit Wahrscheinlichkeit $1 - \frac{1}{2^k}$ in Laufzeit $O(k \cdot (\log n)^3)$ einen echten Teiler von n ausgibt.

Zwischenfazit

1. **Satz.** Sei P ein Orakel, das zu (n,a) mit $1 < a < n$ und $\mathrm{ggT}(a,n) = 1$ die Periode der Funktion $x \mapsto a^x \bmod n$ ausgibt.

   Sei $A = \{n \in \mathbb{N} \colon n \text{ ist ungerade und keine Primpotenz}\}$.

   Dann gibt es einen randomisierten Algorithmus mit Orakel P, der bei Eingabe $n \in A$ mit Wahrscheinlichkeit $1 - \frac{1}{2^k}$ in Laufzeit $O(k \cdot (\log n)^3)$ einen echten Teiler von n ausgibt.

2. **Ziel.** Es bleibt zu zeigen, dass sich das Orakel P durch polynomiell uniforme Quantenschaltkreise berechnen lässt.

13 Fouriertransformationen

Polynommultiplikation

1. Es seien zwei Polynome

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{und} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i$$

gegeben. Wir wollen

$$C(x) := A(x) \cdot B(x) = \sum_{i=0}^{2n-2} c_i x^i \quad \text{mit} \quad c_i = \sum_{j=0}^{i} a_j b_{i-j}$$

bestimmen.

2. Die naive Berechnung der c_i's braucht $\Theta(n^2)$ Multiplikationen.

   In diesem Abschnitt messen wir Komplexität in Anzahl der Multiplikationen. Da wir den Aufwand später mit der Größe von Quantenschaltkreisen vergleichen wollen, und die Kodierungslänge von komplexen Zahlen dort keine Rolle spielt, ist dies sinnvoll.

3. **Ziel.** Ein schnelleres Verfahren.

Stützstellendarstellung

1. **Satz.** Zu $(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})$ mit $x_i \neq x_j$ für $i \neq j$ gibt es genau ein Polynom P vom Grad $n-1$ mit $P(x_i) = y_i$ für alle i.
2. **Beweis.** Für die Koeffizienten $p_0, \ldots, p_{n-1}$ von P muss gelten

$$\underbrace{\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix}}_{\text{Vandermonde-Matrix } V(x_0,\ldots,x_{n-1})} \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}.$$

   Da die x_i's paarweise verschieden sind, ist $V(x_0, \ldots, x_{n-1})$ invertierbar, also sind die $p_0, \ldots, p_{n-1}$ eindeutig bestimmt. □
3. **Also:** Kennen wir $2n-1$ gemeinsame Stützstellen von A und B, so können wir eine Stützstellendarstellung von C durch punktweise Multiplikation in $\Theta(n)$ Schritten bestimmen.

Umwandlung der Darstellungen

1. Es bleibt also zu klären, wie wir *schnell*
   - die Koeffizientendarstellungen von A und B in Stützstellendarstellungen,
   - und die Stützstellendarstellung von C in eine Koeffizientendarstellung

   umwandeln können.
2. **Verständniskontrolle.** Wie geht ersteres in Laufzeit $\Theta(n^2)$?
3. Natürlich haben wir mit Laufzeit $\Theta(n^2)$ nichts gewonnen, da das nicht schneller ist, als die naive Berechnung der c_i's.
4. Wir müssen also schnellere Umwandlungsverfahren finden.
5. Wir können dabei die Stützstellen frei wählen; falls wir also solche finden können, mit denen sich besonders schnell rechnen lässt, so können wir diese nutzen.

Einheitswurzeln & diskrete Fouriertransformation

1. **Definition.** Sei $\omega_n = e^{\frac{2\pi i}{n}}$. Die Zahlen $(\omega_n)^j$ für $0 \leq j \leq n-1$ heißen *n-te Einheitswurzeln*.
2. **Definition.** Sei $A(x) = \sum_{i=0}^{n-1} a_i x^i$. Dann definieren wir die *diskrete Fouriertransformation* des Vektors $(a_0, \ldots, a_{n-1})^T$ als

$$\mathrm{DFT}\left(\begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix}\right) = \begin{pmatrix} A(\omega_n^0) \\ \vdots \\ A(\omega_n^{n-1}) \end{pmatrix}.$$

3. **Intuition.** Man kann das als eine Stützstellendarstellung von A mit den n-ten Einheitswurzeln als Stützstellen verstehen.
4. In Kürze zeigen wir, wie wir diese spezielle Stützstellendarstellung schnell in eine Koeffizientendarstellung umwandeln können, und umgekehrt.

Diskrete Fouriertransformation

1. DFT_n sei die Einschränkung von DFT auf $\mathbb{C}^n$.
2. **Verständniskontrolle.** Überzeugen Sie sich, dass DFT_n die Matrix mit den Einträgen $\omega_n^{(j-1)(k-1)}$ an der Stelle (j,k) ist.
3. **Fakt.** DFT_n^{-1} ist die Matrix mit Einträgen $\frac{\omega_n^{-(j-1)(k-1)}}{n}$ bei (j,k).

   Beweis. $\rightarrow$ Übungsaufgabe.

Fast Fourier Transformation (FFT)

1. **Fakt.** Sei n gerade und

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1}.$$

Fast Fourier Transformation (FFT)

1. **Fakt.** Sei n gerade und

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1}.$$

Definiere

$$A_0(x) \quad = \quad a_0 + a_2 x + \cdots + a_{n-2} x^{n/2-1}$$

Fast Fourier Transformation (FFT)

1. **Fakt.** Sei n gerade und

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1}.$$

Definiere

$$\begin{aligned} A_0(x) &= a_0 + a_2 x + \cdots + a_{n-2} x^{n/2-1} \text{ und} \\ A_1(x) &= a_1 + a_3 x + \cdots + a_{n-1} x^{n/2-1}. \end{aligned}$$

Fast Fourier Transformation (FFT)

1. **Fakt.** Sei n gerade und

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1}.$$

Definiere

$$\begin{aligned} A_0(x) &= a_0 + a_2 x + \cdots + a_{n-2} x^{n/2-1} \text{ und} \\ A_1(x) &= a_1 + a_3 x + \cdots + a_{n-1} x^{n/2-1}. \end{aligned}$$

Dann gilt $A(x) = A_0(x^2) + x \cdot A_1(x^2)$.

Fast Fourier Transformation (FFT)

1. **Fakt.** Sei n gerade und

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1}.$$

   Definiere

$$\begin{aligned} A_0(x) &= a_0 + a_2 x + \cdots + a_{n-2} x^{n/2-1} \text{ und} \\ A_1(x) &= a_1 + a_3 x + \cdots + a_{n-1} x^{n/2-1}. \end{aligned}$$

   Dann gilt $A(x) = A_0(x^2) + x \cdot A_1(x^2)$.

2. **Idee.** Wir berechnen DFT_n rekursiv aus $\mathrm{DFT}_{n/2}$.

3. **Frage.** Nun evaluiert aber DFT_n ein Polynom an den n-ten Einheitswurzeln, und $\mathrm{DFT}_{n/2}$ an den $n/2$-ten Einheitswurzeln. Wie kann also $\mathrm{DFT}_{n/2}$ bei der Berechnung von DFT_n helfen?

Fast Fourier Transformation (FFT)

1. **Fakt.** Sei n gerade und

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1}.$$

   Definiere

$$\begin{aligned} A_0(x) &= a_0 + a_2 x + \cdots + a_{n-2} x^{n/2-1} \text{ und} \\ A_1(x) &= a_1 + a_3 x + \cdots + a_{n-1} x^{n/2-1}. \end{aligned}$$

   Dann gilt $A(x) = A_0(x^2) + x \cdot A_1(x^2)$.

2. **Idee.** Wir berechnen DFT_n rekursiv aus $\mathrm{DFT}_{n/2}$.

3. **Frage.** Nun evaluiert aber DFT_n ein Polynom an den n-ten Einheitswurzeln, und $\mathrm{DFT}_{n/2}$ an den $n/2$-ten Einheitswurzeln. Wie kann also $\mathrm{DFT}_{n/2}$ bei der Berechnung von DFT_n helfen?

4. **Fakt.** Sei n gerade. Dann liefert Quadrieren der n-ten Einheitswurzeln die $n/2$-ten Einheitswurzeln.

Fast Fourier Transformation (FFT)

1. Eingabe: $(a_0, \ldots, a_{n-1})$ wobei $n = 2^\ell$ für ein $\ell \in \mathbb{N}$.
2. Ausgabe: $\mathrm{DFT}_n(a_0, \ldots, a_{n-1})$
3. Algorithmus FFT_n:
   - Falls $n - 1 = 0$, gib a_0 aus und brich ab.
   - $y_0 := \mathrm{FFT}_{n/2}(a_0, a_2, \ldots, a_{n-2})$
   - $y_1 := \mathrm{FFT}_{n/2}(a_1, a_3, \ldots, a_{n-1})$
   - Setze y_0 und y_1 zusammen und gib $\mathrm{DFT}_n(a_0, \ldots, a_{n-1})$ aus.

   (Wie das Zusammensetzen funktioniert, sehen wir auf der nächsten Folie.)

Zusammensetzen

1. **Wir wollen:** $(A(1), A(\omega_n), A(\omega_n^2), \ldots, A(\omega_n^{n-1}))^T$.
2. **Es gilt:**

$$
\begin{array}{rclcl}
A(1) & = & A_0(1) & + & 1 \cdot A_1(1) \\
A(\omega_n) & = & A_0(\omega_n^2) & + & \omega_n \cdot A_1(\omega_n^2) \\
\vdots & & \vdots & & \vdots \\
A(\omega_n^{n/2-1}) & = & A_0(\omega_n^{n-2}) & + & \omega_n^{n/2-1} \cdot A_1(\omega_n^{n-2}) \\
A(\omega_n^{n/2}) & = & A_0(\omega_n^n) & + & \omega_n^{n/2} \cdot A_1(\omega_n^n) \\
A(\omega_n^{n/2+1}) & = & A_0(\omega_n^{n+2}) & + & \omega_n^{n/2+1} \cdot A_1(\omega_n^{n+2}) \\
\vdots & & \vdots & & \vdots \\
A(\omega_n^{n-1}) & = & A_0(\omega_n^{2n-2}) & + & \omega_n^{n-1} \cdot A_1(\omega_n^{2n-2})
\end{array}
\begin{array}{rclcl}
= & A_0(1) & + & \omega_n^{n/2} \cdot A_1(1) \\
= & A_0(\omega_n^2) & + & \omega_n^{n/2+1} \cdot A_1(\omega_n^2) \\
& \vdots & & \vdots \\
= & A_0(\omega_n^{n-2}) & + & \omega_n^{n-1} \cdot A_1(\omega_n^{n-2})
\end{array}
$$

3. **Durch die rekursiven Aufrufe erhalten wir:**

$$
\begin{array}{l}
(A_0(1), A_0(\omega_n^2), \ldots, A_0(\omega_n^{n-2}))^T, \\
(A_1(1), A_1(\omega_n^2), \ldots, A_1(\omega_n^{n-2}))^T.
\end{array}
$$

Zusammensetzen

1. **Wir wollen:** $(A(1), A(\omega_n), A(\omega_n^2), \ldots, A(\omega_n^{n-1}))^T$.
2. **Es gilt:**

$$
\begin{array}{rclcl}
A(1) & = & A_0(1) & + & 1 \cdot A_1(1) \\
A(\omega_n) & = & A_0(\omega_n^2) & + & \omega_n \cdot A_1(\omega_n^2) \\
\vdots & & \vdots & & \vdots \\
A(\omega_n^{n/2-1}) & = & A_0(\omega_n^{n-2}) & + & \omega_n^{n/2-1} \cdot A_1(\omega_n^{n-2}) \\
A(\omega_n^{n/2}) & = & A_0(\omega_n^n) & + & \omega_n^{n/2} \cdot A_1(\omega_n^n) \\
A(\omega_n^{n/2+1}) & = & A_0(\omega_n^{n+2}) & + & \omega_n^{n/2+1} \cdot A_1(\omega_n^{n+2}) \\
\vdots & & \vdots & & \vdots \\
A(\omega_n^{n-1}) & = & A_0(\omega_n^{2n-2}) & + & \omega_n^{n-1} \cdot A_1(\omega_n^{2n-2})
\end{array}
\quad
\begin{array}{rclcl}
= & A_0(1) & + & \omega_n^{n/2} \cdot A_1(1) \\
= & A_0(\omega_n^2) & + & \omega_n^{n/2+1} \cdot A_1(\omega_n^2) \\
\vdots & \vdots & & \vdots \\
= & A_0(\omega_n^{n-2}) & + & \omega_n^{n-1} \cdot A_1(\omega_n^{n-2})
\end{array}
$$

3. **Durch die rekursiven Aufrufe erhalten wir:**

$$
\begin{array}{l}
(A_0(1), A_0(\omega_n^2), \ldots, A_0(\omega_n^{n-2}))^T, \\
(A_1(1), A_1(\omega_n^2), \ldots, A_1(\omega_n^{n-2}))^T.
\end{array}
$$

Zusammensetzen

1. **Wir wollen:** $(A(1), A(\omega_n), A(\omega_n^2), \ldots, A(\omega_n^{n-1}))^T$.
2. **Es gilt:**

$$
\begin{array}{rclcl}
A(1) & = & A_0(1) & + & 1 \cdot A_1(1) \\
A(\omega_n) & = & A_0(\omega_n^2) & + & \omega_n \cdot A_1(\omega_n^2) \\
\vdots & & \vdots & & \vdots \\
A(\omega_n^{n/2-1}) & = & A_0(\omega_n^{n-2}) & + & \omega_n^{n/2-1} \cdot A_1(\omega_n^{n-2}) \\
A(\omega_n^{n/2}) & = & A_0(\omega_n^{n}) & + & \omega_n^{n/2} \cdot A_1(\omega_n^{n}) \\
A(\omega_n^{n/2+1}) & = & A_0(\omega_n^{n+2}) & + & \omega_n^{n/2+1} \cdot A_1(\omega_n^{n+2}) \\
\vdots & & \vdots & & \vdots \\
A(\omega_n^{n-1}) & = & A_0(\omega_n^{2n-2}) & + & \omega_n^{n-1} \cdot A_1(\omega_n^{2n-2})
\end{array}
\quad
\begin{array}{rclcl}
= & A_0(1) & + & \omega_n^{n/2} \cdot A_1(1) \\
= & A_0(\omega_n^2) & + & \omega_n^{n/2+1} \cdot A_1(\omega_n^2) \\
& \vdots & & \vdots \\
= & A_0(\omega_n^{n-2}) & + & \omega_n^{n-1} \cdot A_1(\omega_n^{n-2})
\end{array}
$$

3. **Durch die rekursiven Aufrufe erhalten wir:**

$$
\begin{array}{l}
(A_0(1), A_0(\omega_n^2), \ldots, A_0(\omega_n^{n-2}))^T, \\
(A_1(1), A_1(\omega_n^2), \ldots, A_1(\omega_n^{n-2}))^T.
\end{array}
$$

Zusammensetzen

1. **Wir wollen:** $(A(1), A(\omega_n), A(\omega_n^2), \ldots, A(\omega_n^{n-1}))^T$.
2. **Es gilt:**

$$
\begin{array}{rclcl}
A(1) & = & A_0(1) & + & 1 \cdot A_1(1) \\
A(\omega_n) & = & A_0(\omega_n^2) & + & \omega_n \cdot A_1(\omega_n^2) \\
\vdots & & \vdots & & \vdots \\
A(\omega_n^{n/2-1}) & = & A_0(\omega_n^{n-2}) & + & \omega_n^{n/2-1} \cdot A_1(\omega_n^{n-2}) \\
A(\omega_n^{n/2}) & = & A_0(\omega_n^{n}) & + & \omega_n^{n/2} \cdot A_1(\omega_n^{n}) \\
A(\omega_n^{n/2+1}) & = & A_0(\omega_n^{n+2}) & + & \omega_n^{n/2+1} \cdot A_1(\omega_n^{n+2}) \\
\vdots & & \vdots & & \vdots \\
A(\omega_n^{n-1}) & = & A_0(\omega_n^{2n-2}) & + & \omega_n^{n-1} \cdot A_1(\omega_n^{2n-2})
\end{array}
\quad
\begin{array}{rclcl}
= & A_0(1) & + & \omega_n^{n/2} \cdot A_1(1) \\
= & A_0(\omega_n^2) & + & \omega_n^{n/2+1} \cdot A_1(\omega_n^2) \\
 & \vdots & & \vdots \\
= & A_0(\omega_n^{n-2}) & + & \omega_n^{n-1} \cdot A_1(\omega_n^{n-2})
\end{array}
$$

3. **Durch die rekursiven Aufrufe erhalten wir:**

$$
\begin{array}{l}
(A_0(1), A_0(\omega_n^2), \ldots, A_0(\omega_n^{n-2}))^T, \\
(A_1(1), A_1(\omega_n^2), \ldots, A_1(\omega_n^{n-2}))^T.
\end{array}
$$

Zusammensetzen

1. **Wir wollen:** $(A(1), A(\omega_n), A(\omega_n^2), \ldots, A(\omega_n^{n-1}))^T$.
2. **Es gilt:**

$$
\begin{array}{rclcl}
A(1) & = & A_0(1) & + & 1 \cdot A_1(1) \\
A(\omega_n) & = & A_0(\omega_n^2) & + & \omega_n \cdot A_1(\omega_n^2) \\
\vdots & & \vdots & & \vdots \\
A(\omega_n^{n/2-1}) & = & A_0(\omega_n^{n-2}) & + & \omega_n^{n/2-1} \cdot A_1(\omega_n^{n-2}) \\
A(\omega_n^{n/2}) & = & A_0(\omega_n^n) & + & \omega_n^{n/2} \cdot A_1(\omega_n^n) \\
A(\omega_n^{n/2+1}) & = & A_0(\omega_n^{n+2}) & + & \omega_n^{n/2+1} \cdot A_1(\omega_n^{n+2}) \\
\vdots & & \vdots & & \vdots \\
A(\omega_n^{n-1}) & = & A_0(\omega_n^{2n-2}) & + & \omega_n^{n-1} \cdot A_1(\omega_n^{2n-2})
\end{array}
\begin{array}{rclcl}
 & & & & \\
 & & & & \\
 & & & & \\
 & & & & \\
 & = & A_0(1) & + & \omega_n^{n/2} \cdot A_1(1) \\
 & = & A_0(\omega_n^2) & + & \omega_n^{n/2+1} \cdot A_1(\omega_n^2) \\
 & & \vdots & & \vdots \\
 & = & A_0(\omega_n^{n-2}) & + & \omega_n^{n-1} \cdot A_1(\omega_n^{n-2})
\end{array}
$$

3. **Durch die rekursiven Aufrufe erhalten wir:**

$$
\begin{array}{l}
(A_0(1), A_0(\omega_n^2), \ldots, A_0(\omega_n^{n-2}))^T, \\
(A_1(1), A_1(\omega_n^2), \ldots, A_1(\omega_n^{n-2}))^T.
\end{array}
$$

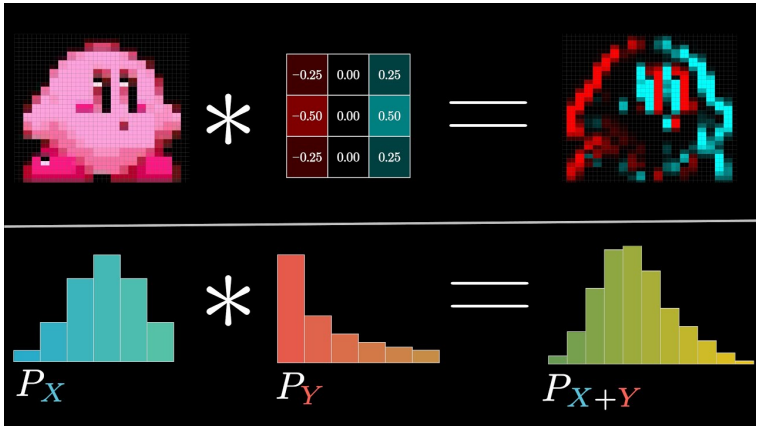
Laufzeitanalyse

1. Die ω_n^k für $k = 0, \ldots, n-1$ kann man vorausberechnen in $O(n)$.
2. Ohne Rekursion ist die Laufzeit pro Aufruf dann $\Theta(n)$.
3. Mit Rekursion ergibt sich $T(n) = 2T(n/2) + \Theta(n)$.
4. Also ist für jede Stufe der Rekursion die Laufzeit $\Theta(n)$, und es gibt $O(\log n)$ solche Stufen.
5. Die Gesamtlaufzeit ist also $O(n \log n)$.

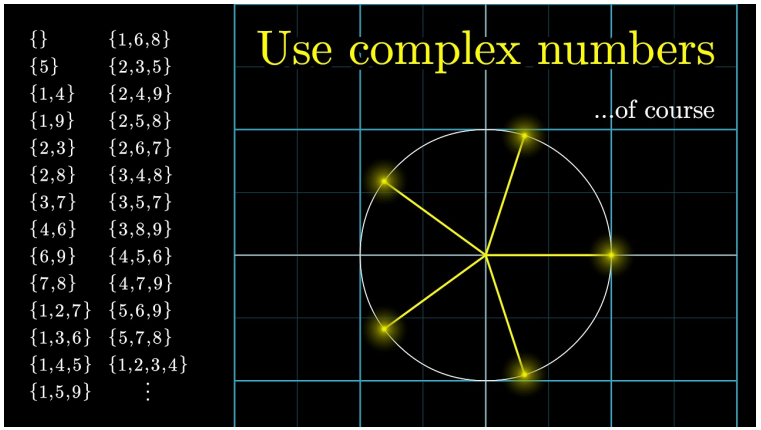
6. Aus dem Fakt auf Folie 99 folgt, dass man mit minimalen Änderungen am Algorithmus auch DFT^{-1} in $O(n \log n)$ berechnen kann.

3Blue1Brown-Videoempfehlungen

1. Zum Thema Fast Fourier Transform:



2. Hier geht es zwar nicht um Quantencomputer, aber die Ideen sind verwandt mit denen aus diesem und dem nächsten Kapitel:



14 Quanten-Fouriertransformation

Quanten-Fouriertransformation (QFT)

1. **Definition.** Schreibe $N = 2^n$. Dann ist $\mathrm{QFT}_N = \frac{1}{\sqrt{N}}\mathrm{DFT}_N$.
2. QFT_N ist also die $N \times N$-Matrix mit $\frac{1}{\sqrt{N}} \cdot \omega_N^{(j-1)(k-1)}$ bei (j,k).
3. QFT_N operiert auf den Vektoren $|0\rangle, \ldots, |N-1\rangle$ via

$$\mathrm{QFT}_N |j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{j \cdot k} |k\rangle .$$

4. QFT ist also eine Version von DFT, die so normiert ist, dass man sie auf Zustände von Quantenregistern anwenden kann.
5. Ähnlich zur Hadamardtransformation, denn $\omega_2 = -1$, aber beachte: $j \cdot k$ ist das Produkt in $\mathbb{N}$, kein Skalarprodukt.
6. **Verständniskontrolle.** Zeigen Sie $\mathrm{QFT}_N |0^n\rangle = H_n |0^n\rangle$.

Darstellung durch Quantengatter

1. **Beispiel.** Sei $x \in \{0,1\}^2$.

$$\begin{aligned}
\mathrm{QFT}_4 |x\rangle &= \tfrac{1}{2} \textstyle\sum_{y=0}^{3} \omega_4^{xy} |y\rangle \\
&= \tfrac{1}{2}(\omega_4^0 |00\rangle + \omega_4^x |01\rangle + \omega_4^{2x} |10\rangle + \omega_4^{3x} |11\rangle) \\
&= \tfrac{1}{2}(|0\rangle + \omega_4^{2x} |1\rangle) \otimes (|0\rangle + \omega_4^x |1\rangle) \\
&= \tfrac{1}{2}(|0\rangle + \omega_2^x |1\rangle) \otimes (|0\rangle + \omega_4^x |1\rangle).
\end{aligned}$$

2. **Allgemeiner:** Sei $x \in \{0,1\}^n$.

$$\begin{aligned}
\mathrm{QFT}_N |x\rangle &= \tfrac{1}{\sqrt{N}} \textstyle\sum_{y=0}^{N-1} \omega_N^{xy} |y\rangle \\
&= \tfrac{1}{\sqrt{N}} \textstyle\bigotimes_{i=1}^{n} (|0\rangle + \omega_{2^n}^{2^{n-i}x} |1\rangle) \\
&= \tfrac{1}{\sqrt{N}} \textstyle\bigotimes_{i=1}^{n} (|0\rangle + \omega_{2^i}^{x} |1\rangle).
\end{aligned}$$

3. **Verständniskontrolle.** Warum gilt die zweite Gleichheit?

Darstellung durch Quantengatter

1. **Beispiel.** Sei $x \in \{0,1\}^2$.

$$\begin{aligned}
\mathrm{QFT}_4 |x\rangle &= \tfrac{1}{2} \textstyle\sum_{y=0}^{3} \omega_4^{xy} |y\rangle \\
&= \tfrac{1}{2}(\omega_4^0 |00\rangle + \omega_4^x |01\rangle + \omega_4^{2x} |10\rangle + \omega_4^{3x} |11\rangle) \\
&= \tfrac{1}{2}(|0\rangle + \omega_4^{2x} |1\rangle) \otimes (|0\rangle + \omega_4^x |1\rangle) \\
&= \tfrac{1}{2}(|0\rangle + \omega_2^x |1\rangle) \otimes (|0\rangle + \omega_4^x |1\rangle).
\end{aligned}$$

2. **Allgemeiner:** Sei $x \in \{0,1\}^n$.

$$\begin{aligned}
\mathrm{QFT}_N |x\rangle &= \tfrac{1}{\sqrt{N}} \textstyle\sum_{y=0}^{N-1} \omega_N^{xy} |y\rangle \\
&= \tfrac{1}{\sqrt{N}} \textstyle\bigotimes_{i=1}^{n} (|0\rangle + \omega_{2^n}^{2^{n-i}x} |1\rangle) \\
&= \tfrac{1}{\sqrt{N}} \textstyle\bigotimes_{i=1}^{n} (|0\rangle + \omega_{2^i}^{x} |1\rangle).
\end{aligned}$$

3. **Verständniskontrolle.** Warum gilt die zweite Gleichheit?
4. **Antwort.** Sei $y = y_{n-1} \dots y_0$. Dann $xy = x \cdot \sum_{j:\, y_j = 1} 2^j$. Somit trägt jedes j mit $y_j = 1$ einen Faktor $\omega_N^{x \cdot 2^j}$ zur Amplitude von $|y\rangle$ bei.

Darstellung durch Quantengatter

1. Wir müssen also einen Quantenschaltkreis konstruieren, der jeden Grundzustand $|x\rangle$ in den Zustand

$$\mathrm{QFT}_N |x\rangle = \frac{1}{\sqrt{N}} \bigotimes_{i=1}^{n} (|0\rangle + \omega_{2^i}^{x} |1\rangle)$$

   überführt.

2. Einen Teil der Aufgabe erfüllt die Hadamardtransformation:

$$H|x_i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_i}|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + \omega_2^{x_i}|1\rangle)$$

3. Für die höheren Einheitswurzeln benutzen wir die Operation

$$R_m = \begin{pmatrix} 1 & 0 \\ 0 & \omega_m \end{pmatrix}.$$

4. **Verständniskontrolle.** Warum ist R_m unitär?

Darstellung durch Quantengatter

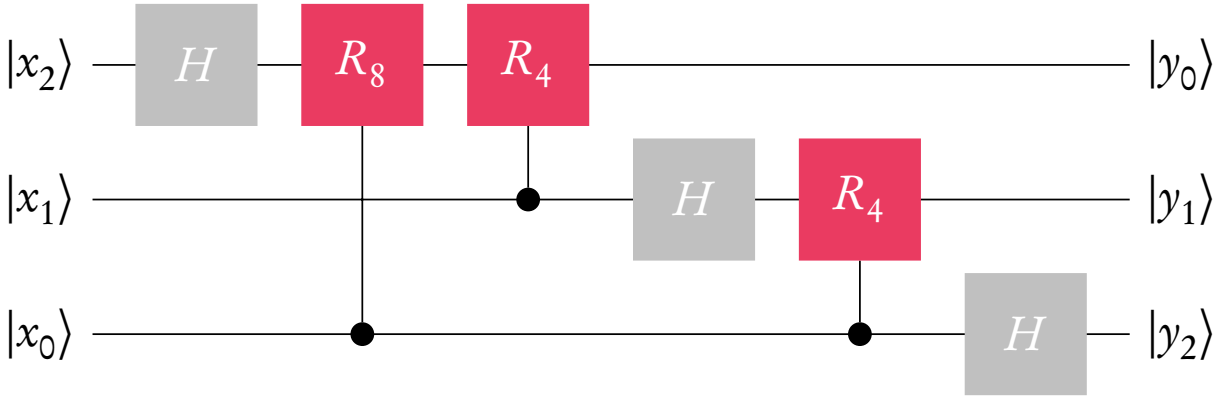
1. Wir müssen Zustände der Form $(|0\rangle + \omega_{2^i}^x |1\rangle)$ erzeugen.
2. Beachte, dass R_{2^i} auf $|0\rangle$ keinen Effekt hat, aber aus $|1\rangle$ das Ergebnis $\omega_{2^i} |1\rangle$ erzeugt — genau das, was wir brauchen.
3. Nun müssen wir R_{2^i} noch x mal anwenden. Wegen $\omega_{2^i}^{2^i+k} = \omega_{2^i}^k$ reicht es, R_{2^i} für jedes i nur $x \bmod 2^i$ mal auf Bit i anzuwenden.
4. **Genauer:** Schreibe $\left|y_{n-1} \dots y_0\right\rangle = \mathrm{QFT}_N(\left|x_{n-1} \dots x_0\right\rangle)$. Dann soll $\left|y_{n-i}\right\rangle$ am Ende den folgenden Zustand haben:

$$\begin{aligned}
& \tfrac{1}{\sqrt{2}}(|0\rangle + \omega_{2^i}^{x \bmod 2^i} |1\rangle) \\
= \; & \tfrac{1}{\sqrt{2}}(|0\rangle + \omega_{2^i}^{\sum_{j=0}^{i-1} 2^j \cdot x_j} |1\rangle) \\
= \; & \tfrac{1}{\sqrt{2}}(|0\rangle + \textstyle\prod_{0 \le j \le i-1 \colon x_j \neq 0} \omega_{2^i}^{2^j} |1\rangle) \\
= \; & \tfrac{1}{\sqrt{2}}(|0\rangle + \textstyle\prod_{0 \le j \le i-1 \colon x_j \neq 0} \omega_{2^{i-j}} |1\rangle).
\end{aligned}$$

5. Das wird erreicht, indem man zunächst H und dann für alle $0 \le j < i-1$, falls $x_j = 1$, $R_{2^{i-j}}$ anwendet.

Darstellung durch Quantengatter

1. **Beispiel.** Das entsprechende Quantengatter im Fall $n = 3$:



2. Die „•“ stellen *kontrollierte Operationen* (→ Übungsaufgabe) dar; die R_m sollen ja nur abhängig von x’ Bits angewandt werden.
3. Die Anordnung ist so gewählt, dass Bits erst modifiziert werden, sobald sie keine anderen Operationen mehr kontrollieren.
4. **Beachte:** $|y\rangle$ wird rückwärts ausgegeben. Zum Umsortieren können Crossover-Gatter (→ Übungsaufgabe) benutzt werden.
5. **Verständniskontrolle.** Überzeugen Sie sich, dass QFT_N nur $O(n^2) = O((\log N)^2)$ lokale Quantengatter benötigt.

Zusammenfassung

1. **Satz.** QFT_N ist mit $O((\log N)^2)$ lokalen Quantengattern darstellbar.
2. Das klassische Verfahren FFT_N benötigt Laufzeit $\Theta(N \log N)$.
3. **Wir werden sehen:** Dieser exponentielle Vorteil ermöglicht Shor's schnellen Faktorisierungsalgorithmus.

15 Simons Algorithmus

Simons Algorithmus

1. Ein für das Verständnis sinnvoller Zwischenschritt vom Deutsch-Josza-Algorithmus zu Shors Algorithmus.
2. Eine Funktion $f\colon \{0,1\}^n \to \{0,1\}^n$ ist gegeben, so dass es ein $s \in \{0,1\}^n$ gibt, so dass für alle $x \in \{0,1\}^n$ gilt $f(x) = f(x \oplus s)$. Für $x \neq y \neq x \oplus s$ gilt $f(x) \neq f(y)$.
3. Der Fall $s = 0^n$ ist erlaubt.

   Verständniskontrolle. Welche Eigenschaft hat dann f?
4. Wir sollen s bestimmen.

Simons Algorithmus

Quantenteil:

1. $|a\rangle|b\rangle \leftarrow |0^n\rangle|0^n\rangle$.
2. $|a\rangle|b\rangle \leftarrow (H_n|a\rangle)|b\rangle = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle|0^n\rangle$.
3. $|a\rangle|b\rangle \leftarrow U_f(|a\rangle|b\rangle) = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle|f(x)\rangle$.
4. Miss $|b\rangle$.
5. $|a\rangle|b\rangle \leftarrow (H_n|a\rangle)|b\rangle$.
6. Miss $|a\rangle$ und speichere das Ergebnis z.
7. Wiederhole Schritte 1–6 solange, bis $n-1$ linear unabhängige z's gefunden wurden; wir bezeichnen diese als $z_1,\ldots,z_{n-1}$.

Klassischer Teil:

1. Finde eine Lösung $t\in\{0,1\}^n$, $t\neq 0^n$, des Gleichungssystems

$$z_1\odot t\equiv 0 \bmod 2,\ \ldots,\ z_{n-1}\odot t\equiv 0 \bmod 2.$$

2. Falls $f(0)=f(t)$, gib t aus; sonst gib 0^n aus.

Analyse des Quantenteils

1. Nach Schritt 3 wird $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle$ erreicht.
2. Sei $f(x)$ das Ergebnis der Messung in Schritt 4.
   - Falls $s = 0^n$, so gibt es nur ein Urbild x von $f(x)$, also wurde durch das Messen ein zufällig ausgewählter Zustand der Form $|x\rangle |f(x)\rangle$ erreicht.
   - Falls $s \neq 0^n$, so gibt es genau zwei Urbilder x und $x \oplus s$ von $f(x)$, also wurde ein Zustand $\frac{1}{\sqrt{2}}(|x\rangle + |x \oplus s\rangle) |f(x)\rangle$ erreicht.
3. **Verständniskontrolle.** Wäre es sinnvoll, an dieser Stelle das erste Register zu messen? Warum, bzw. warum nicht?

Analyse des Quantenteils

1. Nimm an, $s \neq 0^n$. Nach Schritt 5 erreichen wir dann

$$\begin{aligned}
& \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} \frac{1}{\sqrt{2}} \left((-1)^{z \odot x} + (-1)^{z \odot (x \oplus s)} \right) |z\rangle \, |f(x)\rangle \\
= \; & \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} \frac{1}{\sqrt{2}} \left((-1)^{z \odot x} + (-1)^{z \odot x} (-1)^{z \odot s} \right) |z\rangle \, |f(x)\rangle \\
= \; & \frac{1}{\sqrt{2^{n-1}}} \left(\sum_{\substack{z \in \{0,1\}^n : \, z \odot s \text{ gerade} \\ z \odot x \text{ gerade}}} |z\rangle - \sum_{\substack{z \in \{0,1\}^n : \, z \odot s \text{ gerade} \\ z \odot x \text{ ungerade}}} |z\rangle \right) |f(x)\rangle \, .
\end{aligned}$$

2. Für alle z_i, $1 \leq i \leq n-1$, gilt also $z_i \odot s \equiv 0 \bmod 2$.

Analyse des klassischen Teils

1. **Erinnerung.** Wir lösen das Gleichungssystem

$$z_1 \odot t \equiv 0 \bmod 2, \ \ldots, \ z_{n-1} \odot t \equiv 0 \bmod 2.$$

   nach $t \in \{0,1\}^n$, $t \neq 0^n$ auf.
2. Für $v \neq 0^n$ formen die $w \in \{0,1\}^n$ mit $v \odot w \equiv 0 \bmod 2$ einen $(n-1)$-dimensionalen Vektorraum. Also ist t durch die $n-1$ Gleichungen eindeutig bestimmt.
3. Falls $s \neq 0^n$, gilt $z_i \odot s \equiv 0 \bmod 2$ für alle z_i, also $t = s$.
4. Falls $s = 0^n$ so werden die z_i zufällig aus $\{0,1\}^n$ gezogen; dann ist t bedeutungslos. Aber da f dann bijektiv ist, gilt $f(0) \neq f(t)$.
5. In beiden Fällen arbeitet der Algorithmus korrekt.

Laufzeitanalyse

1. **Beispiel.** Falls $s \neq 0^n$, wird in Durchlauf i ein z_i aus 2^{n-1} Vektoren zufällig ausgewählt.
   - z_1 ist linear abhängig falls $z_1 = 0^n$.
     Die Wahrscheinlichkeit dafür beträgt $2^{-(n-1)}$.
   - z_2 ist linear abhängig von z_1 falls $z_2 \in \{0^n, z_1\}$.
     Die Wahrscheinlichkeit dafür beträgt $2 \cdot 2^{-(n-1)}$.
   - z_3 ist linear abhängig von z_1, z_2 falls $z_3 \in \{0^n, z_1, z_2, z_1 \oplus z_2\}$.
     Die Wahrscheinlichkeit dafür beträgt $4 \cdot 2^{-(n-1)}$.
2. **Fakt.** $\mathbb{P}(z_i \text{ ist linear abhängig von } z_1, \ldots, z_{i-1}) = \frac{2^{i-1}}{2^{n-1}} = 2^{i-n}$.

   Verständniskontrolle. Warum gilt das?
3. **Also:** $\mathbb{P}(z_1, \ldots, z_{n-1} \text{ linear unabhängig}) = \prod_{i=1}^{n-1}(1 - 2^{i-n}) > 1/4$.
4. Falls $s = 0^n$ werden die z_i sogar aus 2^n Vektoren gewählt.
5. Mit positiver Wahrscheinlichkeit sind die ersten $n-1$ gefundenen Vektoren also bereits linear unabhängig.
6. Die Gleichungssysteme werden mit Gaußverfahren in $O(n^3)$ gelöst; dabei wird auf lineare Unabhängigkeit der z_i geprüft.

16 Shors Algorithmus

Übersicht

1. Wir haben gesehen, dass Simons Algorithmus die Periode s einer Funktion von $\{0,1\}^n$ nach $\{0,1\}^n$ finden kann.
2. Nun werden wir sehen, wie Shors Algorithmus die Periode einer Funktion $x \mapsto a^x \bmod n$ finden kann.
3. Wir haben bereits gesehen, dass wir dann RSA knacken können.

Quantenteil von Shors Algorithmus

Seien n und N Zweierpotenzen mit $N = c \cdot n^2$ für ein $c \in \mathbb{N}^+$.

Gegeben: Ein $f : \{0, \ldots, N-1\} \to \{0, \ldots, n-1\}$ mit Periode p.

Gesucht: Falls $p \mid N$ ein y mit $y = {}^{jN}/_{p}$; sonst ein y mit $y \approx {}^{jN}/_{p}$.

1. $|a\rangle |b\rangle \leftarrow |0^{\log N}\rangle |0^{\log n}\rangle$.
2. $|a\rangle |b\rangle \leftarrow (H_{\log N} |a\rangle) |b\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle |0^{\log n}\rangle$.
3. $|a\rangle |b\rangle \leftarrow U_f(|a\rangle |b\rangle) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle |f(x)\rangle$.
4. Miss $|b\rangle$.
5. $|a\rangle |b\rangle \leftarrow (\mathrm{QFT}_N |a\rangle) |b\rangle$.
6. Miss $|a\rangle$ und gib das Ergebnis aus.

Analyse im Fall $p \mid N$

1. Nach Schritt 3 wurde $\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle |f(x)\rangle$ erreicht.
2. Durch Schritt 4 geht das zweite Register in einen zufällig ausgewählten eindeutigen Zustand $|f(x)\rangle$ über.
3. Das erste Register ist dann eine Überlagerung aller Urbilder von $f(x)$, also für $A = N/p$ von der Form

$$\frac{1}{\sqrt{A}}(|x\rangle + |(x+p)\rangle + \cdots + |(x+(A-1)p)\rangle)\,|f(x)\rangle\,,$$

   wenn x das kleinste Urbild von $f(x)$ ist.
4. **Verständniskontrolle.** Wäre es sinnvoll, an dieser Stelle das erste Register zu messen? Warum, bzw. warum nicht?

Analyse im Fall $p \mid N$

1. Nach Schritt 5 erreichen wir

$$\begin{aligned} & \frac{1}{\sqrt{NA}} \sum_{y=0}^{N-1} \sum_{j=0}^{A-1} \omega_N^{(x+j\cdot p)\cdot y} \left|y\right\rangle \left|f(x)\right\rangle \\ = \quad & \frac{1}{\sqrt{NA}} \sum_{y=0}^{N-1} \omega_N^{x\cdot y} \sum_{j=0}^{A-1} \omega_N^{j\cdot p\cdot y} \left|y\right\rangle \left|f(x)\right\rangle . \end{aligned}$$

2. Wegen $\left|\omega_N^{x\cdot y}\right|^2 = 1$ hat der Faktor $\omega_N^{x\cdot y}$ keinen Einfluss auf die Wahrscheinlichkeit, mit der ein bestimmtes y gemessen wird.
3. Die Wahrscheinlichkeit, bei Messung des ersten Registers das Ergebnis y zu erhalten beträgt also

$$\mathbb{P}(y) = \frac{1}{NA} \left|\omega_N^0 + \omega_N^{py} + \cdots + \omega_N^{(A-1)py}\right|^2 .$$

Analyse im Fall $p \mid N$

1. Wegen $\omega_N^{jpy} = \omega_A^{jy}$ ist $\mathbb{P}(y) = \frac{1}{NA}\left|\omega_A^0 + \omega_A^y + \cdots + \omega_A^{(A-1)y}\right|^2$.
2. **Summationslemma.** Falls $A \nmid \ell$ so gilt $\sum_{i=0}^{A-1}(\omega_A^\ell)^i = 0$.
   Beweis. $\rightarrow$ Übungsaufgabe.
3. Falls $A \nmid y$ so gilt also $\mathbb{P}(y) = 0$.
4. Falls $A \cdot k = y$, dann $\omega_A^{jy} = (\omega_A^A)^{kj} = 1$, also $\mathbb{P}(y) = \frac{A^2}{NA} = \frac{A}{N} = \frac{1}{p}$.
5. Wir messen also wie gefordert einen der p verschiedenen Werte

$$\{A \cdot 0, \ldots, A \cdot (p-1)\} = \{N/p \cdot 0, \ldots, N/p \cdot (p-1)\}.$$

6. Wir erhalten ein $y = j \cdot N/p$ für ein $j \in \{0, \ldots, (p-1)\}$.

Analyse im Fall $p \nmid N$

1. Nach der Messung in Schritt 4 steht der Wert des zweiten Registers fest, und das erste Register enthält weiterhin eine Überlagerung aller passender Urbilder.
2. Es ist aber komplizierter, die Wahrscheinlichkeit anzugeben, mit der nach Anwendung der QFT ein bestimmtes y gemessen wird, da ω_N^{jpy} keine A-te Einheitswurzel mehr ist.
3. Wir überspringen die Details, aber da $A \approx N/p$ gilt, erhalten wir mit hoher Wahrscheinlichkeit ein $y \in \mathbb{N}$ nahe an einem $j \cdot N/p$.
4. Je größer c ist, desto geringer wird die Abweichung, da c die „Auflösung“ des ersten Registers bestimmt.

Nachbereitung

1. Egal ob $p \mid N$ gilt oder nicht, finden wir also mit großer Wahrscheinlichkeit ein y nahe an einem $j \cdot {}^{N}/_{p}$.
2. Da wir N kennen, kennen wir Zähler und Nenner von $\frac{y}{N}$.
3. Um daraus (mit hoher Wahrscheinlichkeit) p bestimmen zu können, wenden wir das *Kettenbruch*-Vefahren an.

Wiederholte Kettenbrüche

1 **Beispiel.** Die Kettenbruchdarstellung von $\frac{31}{13}$ ist

$$\begin{aligned}
\frac{31}{13} &= 2+\frac{5}{13} = 2+\frac{1}{\frac{13}{5}} = 2+\frac{1}{2+\frac{3}{5}} \\
&= 2+\frac{1}{2+\frac{1}{\frac{5}{3}}} = 2+\frac{1}{2+\frac{1}{1+\frac{2}{3}}} = 2+\frac{1}{2+\frac{1}{1+\frac{1}{\frac{3}{2}}}} \\
&= 2+\frac{1}{2+\frac{1}{1+\frac{1}{1+\frac{1}{2}}}}.
\end{aligned}$$

Wiederholte Kettenbrüche

1. **Beispiel.** Die Kettenbruchdarstellung von $\frac{31}{13}$ ist

$$\begin{aligned}
\frac{31}{13} &= 2+\frac{5}{13} &&= 2+\frac{1}{\frac{13}{5}} &&= 2+\frac{1}{2+\frac{3}{5}} \\
&= 2+\frac{1}{2+\frac{1}{\frac{5}{3}}} &&= 2+\frac{1}{2+\frac{1}{1+\frac{2}{3}}} &&= 2+\frac{1}{2+\frac{1}{1+\frac{1}{\frac{3}{2}}}} \\
&= 2+\frac{1}{2+\frac{1}{1+\frac{1}{1+\frac{1}{2}}}}.
\end{aligned}$$

2. **Definition.** Die grün markierten Terme, sowie der vollständige letzte Term, heißen die *Konvergenten* von $\frac{31}{13}$.

Wiederholte Kettenbrüche

1. **Satz.** Ein gekürzter Bruch a/b hat $O(\log a)$ Konvergenten.
2. **Satz.** Seien $s/r, a/b \in \mathbb{Q}$ und es gelte

$$\left|\frac{s}{r} - \frac{a}{b}\right| \leq \frac{1}{2r^2}.$$

   Dann ist s/r eine der Konvergenten von a/b.

3. Wir haben $N = c \cdot n^2$ und $p \leq n$, und mit großer Wahrscheinlichkeit auch, dass y/N nur wenig vom korrekten Wert j/p abweicht. Also ist der Satz mit großer Wahrscheinlichkeit für

$$s/r = j/p \quad \text{und} \quad a/b = y/N$$

   anwendbar.

   (Wir müssen c ausreichend groß wählen, um die nötige Präzision $\frac{1}{2n^2}$ zu erhalten.)

Wiederholte Kettenbrüche

1. Wir bestimmen also die

$$O(\log y) \leq O(\log N) = O(\log n)$$

   Konvergenten von y/N.
2. Jede Konvergente rechnen wir zurück auf einen gekürzten Bruch k/ℓ und prüfen, ob $a^\ell \bmod n = 1$.
3. Falls ja, so geben wir das kleinste solche ℓ aus.
4. Die arithmetischen Operationen auf den $\log N$-Bit-Zahlen benötigen dabei Laufzeit $O((\log n)^2)$.
5. Falls wir kein geeignetes ℓ gefunden haben, so ist das Kettenbruch-Verfahren gescheitert.
6. Mögliche Gründe für ein Scheitern sind $\mathrm{ggT}(j,p) > 1$ oder im Fall $p \nmid N$, dass wir ein schlechtes y gemessen haben.

Ressourcenabschätzung

1. Die Hadamardtransformation wird auf $O(\log n)$ Bits angewendet und erfordert entsprechend viele lokale Quantengatter.
2. Die Quantenfouriertransformation wird auf $O(\log n)$ Bits angewendet, und erfordert $O((\log n)^2)$ lokale Quantengatter.
3. Das Orakel für $x \mapsto a^x \bmod n$ wird auf $O(\log n)$ Bits angewendet.
   - Es gibt klassische Schaltkreise, die solche modularen Potenzen mit $O((\log n)^3)$ Gattern berechnen.
   - Wir haben gesehen, dass wir diese durch Quantenschaltkreise simulieren können, die nur linear größer sind.

Ressourcenabschätzung

1. Das Kettenbruchverfahren läuft in $O((\log n)^3)$ Schritten und hat aus zahlentheoretischen Gründen eine Erfolgswahrscheinlichkeit von etwa $1/(20 \log\log n)$.
2. Scheitert es, so muss der Quantenteil erneut ausgeführt werden.
3. Nach $O(\log\log n)$ Wiederholungen ist p mit hoher Wahrscheinlichkeit gefunden.
4. **Zusammenfassung Quantenteil.** Der Quantenteil hat eine erwartete Laufzeit von $O(\log\log n \cdot (\log n)^3)$ und erfordert $O((\log n)^3)$ lokale Quantengatter.

Zusammenfassung

1. **Satz.** Shors Algorithmus ermittelt mit erwarteter Laufzeit

$$O(\log\log n \cdot (\log n)^3)$$

einen echten Teiler einer natürlichen Zahl n.

2. **Bemerkung.** Bei Verwendung des besten bekannten Multiplikationsverfahrens beträgt die erwartete Laufzeit

$$O((\log n)^2 \cdot \log\log n \cdot \log\log\log n).$$

17 Grovers Algorithmus

Szenario

1. Ein Funktion $f\colon \{0,1\}^n \to \{0,1\}$ ist gegeben, so dass genau ein $\widehat{x} \in \{0,1\}^n$ mit $f(\widehat{x}) = 1$ existiert.
2. Wir sollen $\widehat{x}$ finden, und dazu f möglichst selten abfragen.
3. Dazu steht uns ein invertierbares Orakel U_f zur Verfügung.
4. Schreibe $N = 2^n$.
5. **Verständniskontrolle.** Wieviele Anfragen benötigt ein klassischer Computer höchstens? Was ist der Erwartungswert der Anzahl der benötigten Abfragen?
6. **Wir werden sehen:** Einem Quantencomputer genügen $O(\sqrt{N})$.

Szenario

1. Ein Funktion $f\colon \{0,1\}^n \to \{0,1\}$ ist gegeben, so dass genau ein $\widehat{x} \in \{0,1\}^n$ mit $f(\widehat{x}) = 1$ existiert.
2. Wir sollen $\widehat{x}$ finden, und dazu f möglichst selten abfragen.
3. Dazu steht uns ein invertierbares Orakel U_f zur Verfügung.
4. Schreibe $N = 2^n$.
5. **Verständniskontrolle.** Wieviele Anfragen benötigt ein klassischer Computer höchstens? Was ist der Erwartungswert der Anzahl der benötigten Abfragen?
6. **Wir werden sehen:** Einem Quantencomputer genügen $O(\sqrt{N})$.
7. **Verständniskontrolle.** Welche Relevanz hat das für **NP**-vollständige Probleme? Welche für symmetrische Verschlüsselung?

Grundidee

1. Wir versetzen ein Register in einen einfach herzustellenden Startzustand $|s\rangle$.
2. Wir wenden eine Reihe von Operationen auf das Register an; dabei wird unter anderem U_f benutzt.
3. Dadurch erhält das Register (ungefähr) den Zustand $|\widehat{x}\rangle$.
4. Eine Messung des Registers liefert dann mit sehr hoher Wahrscheinlichkeit $\widehat{x}$.

Die Operation $U_{\widehat{x}}$

1. Sei $|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$.
2. **Verständniskontrolle.** Wie können wir $|s\rangle$ herstellen?
3. Sei $U_{\widehat{x}} = I_N - 2\,|\widehat{x}\rangle\,\langle\widehat{x}|$.
   In den Übungen haben wir gezeigt, dass $|\widehat{x}\rangle\,\langle\widehat{x}|$ die Projektion auf $|\widehat{x}\rangle$ ist.
4. **Verständniskontrolle.** Wie verändert $U_{\widehat{x}}$ die $|\widehat{x}\rangle$-Komponente eines Vektors? Wie alle anderen Komponenten?
5. **Verständniskontrolle.** Sind Projektionen unitäre Operationen?
6. **Verständniskontrolle.** Warum ist $U_{\widehat{x}}$ dennoch unitär?

Die Operation $U_{\widehat{x}}$

1. Anwendung von $U_{\widehat{x}}$ auf $|s\rangle$ liefert

$$|\psi\rangle := U_{\widehat{x}}|s\rangle = \frac{1}{\sqrt{N}}\left(\sum_{x=0,\,x\neq\widehat{x}}^{N-1} U_{\widehat{x}}|x\rangle + U_{\widehat{x}}|\widehat{x}\rangle\right)$$

$$= \frac{1}{\sqrt{N}}\sum_{x=0,\,x\neq\widehat{x}}^{N-1}|x\rangle - \frac{1}{\sqrt{N}}|\widehat{x}\rangle = |s\rangle - \frac{2}{\sqrt{N}}|\widehat{x}\rangle\,.$$

2. $U_{\widehat{x}}$ führt also eine Drehung von $|s\rangle$ um Winkel θ aus, mit

$$\cos\theta = \langle s|\psi\rangle = \langle s|s\rangle - \frac{2}{\sqrt{N}}\langle s|\widehat{x}\rangle = 1 - \frac{2}{\sqrt{N}}\frac{1}{\sqrt{N}} = 1 - \frac{2}{N}.$$

Achtung: $U_{\widehat{x}}$ ist natürlich eine Spiegelung, keine Drehung. Man kann aber dennoch bestimmen, um welchen Winkel man drehen *müsste*, um *auf den einen festen* Vektor $|s\rangle$ den gleichen Effekt zu haben wie $U_{\widehat{x}}$.

3. θ ist also für typische Werte von N recht klein.

Die Grover-Iteration

1. Sei $U_s = 2\,|s\rangle\langle s| - I_N$.
2. **Definition.** $G = U_s U_{\widehat{x}}$ heißt *Grover-Iteration*.
3. Die Anwendung von G auf $|s\rangle$ liefert

$$|\psi_1\rangle := G|s\rangle = U_s U_{\widehat{x}}|s\rangle = U_s|\psi\rangle = U_s\left(|s\rangle - \tfrac{2}{\sqrt{N}}|\widehat{x}\rangle\right)$$

$$= |s\rangle - \tfrac{2}{\sqrt{N}}(2\,|s\rangle\langle s| - I_N)|\widehat{x}\rangle = |s\rangle - \tfrac{4}{\sqrt{N}}|s\rangle\langle s|\widehat{x}\rangle + \tfrac{2}{\sqrt{N}}|\widehat{x}\rangle$$

$$= |s\rangle - \tfrac{4}{\sqrt{N}}\tfrac{1}{\sqrt{N}}|s\rangle + \tfrac{2}{\sqrt{N}}|\widehat{x}\rangle = \left(1 - \tfrac{4}{N}\right)|s\rangle + \tfrac{2}{\sqrt{N}}|\widehat{x}\rangle\,.$$

4. Wir sehen, dass G die Amplitude von $|\widehat{x}\rangle$ auf Kosten der Amplitude von $|s\rangle$ steigert — genau, was wir brauchen.
5. **Verständniskontrolle.** Überzeugen Sie sich, dass auch bei *wiederholter* Anwendung von $U_{\widehat{x}}$ und/oder U_s auf $|s\rangle$ die von $|s\rangle$ und $|\widehat{x}\rangle$ aufgespannte Ebene E nie verlassen wird.

Die Grover-Iteration

1. G bildet also $|s\rangle$ auf $|\psi_1\rangle$ ab, dazwischen liegt der Winkel

$$\begin{aligned}\langle s|\psi_1\rangle &= \left\langle s\middle|\left[\left(1-\tfrac{4}{N}\right)|s\rangle + \tfrac{2}{\sqrt{N}}|\widehat{x}\rangle\right]\right\rangle \\ &= \left(1-\tfrac{4}{N}\right)\langle s|s\rangle + \tfrac{2}{\sqrt{N}}\langle s|\widehat{x}\rangle \\ &= 1-\tfrac{4}{N}+\tfrac{2}{\sqrt{N}}\tfrac{1}{\sqrt{N}} \\ &= 1-\tfrac{2}{N} \\ &= \cos\theta.\end{aligned}$$

2. Es gilt $\langle\psi|\psi_1\rangle = \cos 2\theta$ ($\rightarrow$ Übungsaufgabe).
   Da $U_{\widehat{x}}$ und G die Ebene E auf sich selbst abbilden, wird $|s\rangle$ also von $U_{\widehat{x}}$ und G in entgegengesetzte Richtungen gedreht.

Die Grover-Iteration

1. Wir müssen noch sicherstellen, dass wiederholte Anwendung von G auf $|s\rangle$ die dabei auftretenden Vektoren immer weiter in Richtung $|\widehat{x}\rangle$ dreht.
2. Da G unitär ist, gilt zunächst, dass

$$\left\langle G^k s \,\middle|\, G^{k-1} s \right\rangle = \left\langle G^{k-1} s \,\middle|\, G^{k-2} s \right\rangle = \ldots = \langle Gs \,|\, s \rangle = \cos\theta.$$

3. Die Winkel sind also schon mal so, wie wir sie brauchen.
4. Wir müssen aber noch die Möglichkeit ausschließen, dass auf E G eine Spiegelung ist. Denn dann würde sich G selbst aufheben, und wir kämen $|\widehat{x}\rangle$ durch mehrfache Anwendung nicht näher.

Die Grover-Iteration

1. G ist aber keine Spiegelung; verglichen mit $|\psi_1\rangle$ steigert eine zweite Anwendung den „$|\widehat{x}\rangle$-Anteil“ weiter auf Kosten von $|s\rangle$:

$$
\begin{aligned}
& G\,|\psi_1\rangle \\
= \; & G\Big(\Big(1-\tfrac{4}{N}\Big)|s\rangle + \tfrac{2}{\sqrt{N}}\,|\widehat{x}\rangle\Big) \\
= \; & \Big(1-\tfrac{4}{N}\Big)G|s\rangle + \tfrac{2}{\sqrt{N}}U_s U_{\widehat{x}}\,|\widehat{x}\rangle \\
= \; & \Big(1-\tfrac{4}{N}\Big)^2|s\rangle + \Big(1-\tfrac{4}{N}\Big)\tfrac{2}{\sqrt{N}}\,|\widehat{x}\rangle + \tfrac{2}{\sqrt{N}}U_s(-|\widehat{x}\rangle) \\
= \; & \Big(1-\tfrac{4}{N}\Big)^2|s\rangle + \Big(1-\tfrac{4}{N}\Big)\tfrac{2}{\sqrt{N}}\,|\widehat{x}\rangle + \tfrac{2}{\sqrt{N}}(-U_s)\,|\widehat{x}\rangle \\
= \; & \Big(1-\tfrac{4}{N}\Big)^2|s\rangle + \Big(1-\tfrac{4}{N}\Big)\tfrac{2}{\sqrt{N}}\,|\widehat{x}\rangle + \tfrac{2}{\sqrt{N}}(I_N\,|\widehat{x}\rangle - 2\,|s\rangle\,\langle s|\widehat{x}\rangle) \\
= \; & \Big(1-\tfrac{4}{N}\Big)^2|s\rangle + \Big(1-\tfrac{4}{N}\Big)\tfrac{2}{\sqrt{N}}\,|\widehat{x}\rangle + \tfrac{2}{\sqrt{N}}(|\widehat{x}\rangle - 2\,|s\rangle\,\tfrac{1}{\sqrt{N}}) \\
= \; & \Big(\Big(1-\tfrac{4}{N}\Big)^2 - \tfrac{4}{N}\Big)|s\rangle + \Big(2-\tfrac{4}{N}\Big)\tfrac{2}{\sqrt{N}}\,|\widehat{x}\rangle\,.
\end{aligned}
$$

2. **Zwischenfazit:** G^k dreht $|s\rangle$ um $k\cdot\theta$ in Richtung $|\widehat{x}\rangle$.

Die Grover-Iteration

1. Unser Ziel ist, $|s\rangle$ mit G^k möglichst nahe an $|\widehat{x}\rangle$ heran zu drehen.
2. Also müssen wir das (kleinste) k bestimmen mit $G^k\,|s\rangle \approx |\widehat{x}\rangle$.
3. Im Wesentlichen benötigen wir also den Startwinkel zwischen $|s\rangle$ und $|\widehat{x}\rangle$, da er bestimmt, wie weit wir drehen müssen.
4. **Verständniskontrolle.** Wir kennen $|\widehat{x}\rangle$ doch gar nicht! Warum können wir dennoch den Startwinkel bestimmen?
   (Das erklärt übrigens auch unsere Wahl des Startzustandes $|s\rangle$.)

Anzahl der Iterationen bestimmen

1. Sei $|u\rangle = \frac{1}{\sqrt{N-1}} \sum_{x=0,\ x\neq\widehat{x}}^{N-1} |x\rangle = \sqrt{\frac{N}{N-1}}\,|s\rangle - \frac{1}{\sqrt{N-1}}\,|\widehat{x}\rangle$.
2. Offensichtlich liegt $|u\rangle$ in E.
3. Da die $|\widehat{x}\rangle$-Komponente von $|u\rangle$ 0 ist, ist $|u\rangle$ orthogonal zu $|\widehat{x}\rangle$.
4. $|\widehat{x}\rangle$ und $|u\rangle$ spannen also E auf, und $|\widehat{x}\rangle$ entsteht durch eine 90°-Drehung von $|u\rangle$.
5. $|s\rangle$ entsteht ebenfalls durch eine Drehung von $|u\rangle$ in Richtung $|\widehat{x}\rangle$ um einen bestimmten Winkel. Wir müssen diesen bestimmen, dann können wir k berechnen.

Anzahl der Iterationen bestimmen

1. Es gilt $|s\rangle = \sqrt{1-\frac{1}{N}}\,|u\rangle + \frac{1}{\sqrt{N}}\,|\widehat{x}\rangle = \cos\frac{\theta}{2}\,|u\rangle + \sin\frac{\theta}{2}\,|\widehat{x}\rangle$, unter Verwendung
   - der Definition $\cos\theta = 1 - {}^{2}/_{N}$,
   - der *Halbwinkelformel* $\cos{}^{\alpha}/_{2} = \sqrt{\frac{1+\cos\alpha}{2}}$,
   - und der Gleichung $(\sin\alpha)^2 = 1-(\cos\alpha)^2$.
2. $|s\rangle$ entsteht also durch eine Drehung um ${}^{\theta}/_{2}$ aus $|u\rangle$.
   Um $|s\rangle$ auf $|\widehat{x}\rangle$ zu drehen, fehlen also noch 90° abzüglich ${}^{\theta}/_{2}$.
3. Die Zahl k ergibt sich also aus ${}^{\pi}/_{2} - {}^{\theta}/_{2} \overset{!}{\approx} k\cdot\theta$ zu $k \approx \frac{\pi-\theta}{2\theta}$.
4. Da $\sin\frac{\theta}{2} = \frac{1}{\sqrt{N}}$, und weil N typischerweise groß ist, kann man die Approximation $\sin x \approx x$ für kleine x verwenden, und erhält

$$G(N) := k \approx \frac{\pi}{4}\sqrt{N} - \frac{1}{2} \approx \frac{\pi}{4}\sqrt{N}.$$

5. Man benötigt also $O(\sqrt{N})$ Iterationen, und jede davon greift einmal auf die Datenbank U_f zu, wie wir gleich sehen werden.

Implementierung durch Quantenschaltkreise

1. **Satz.** Sei G eine Menge von ℓ Grundzuständen eines n-Bit-Quantenregisters und V_G die lineare Abbildung, die auf Grundzuständen $|x\rangle$ operiert gemäß

$$|x\rangle \mapsto \begin{cases} -|x\rangle & \text{falls } |x\rangle \in G, \\ |x\rangle & \text{sonst.} \end{cases}$$

   Dann ist V_G mit $O(\ell \cdot n)$ lokalen Quantengattern darstellbar. Alternativ kann V_G alleine durch U_g dargestellt werden, wenn g die Funktion ist mit $g(x) = 1$ für $|x\rangle \in G$ und $g(x) = 0$ sonst.

   Beweis. → Übungsaufgabe.

2. Wir wählen also $U_{\hat{x}} = V_{\{\hat{x}\}}$ und $U_s = -H_n \cdot V_{\{0^n\}} \cdot H_n$.
3. **Verständniskontrolle.** Warum gilt $U_s = -H_n \cdot V_{\{0^n\}} \cdot H_n$?
4. Damit ist der Grover-Algorithmus vollständig beschrieben.

Grover-Algorithmus für mehrere Lösungen

1. Nimm an, statt einer Lösung $\widehat{x}$ gebe es Lösungen $\widehat{x}_1, \ldots, \widehat{x}_\ell$, wobei gelten möge, dass $\ell < {}^3/_4 N$.
2. **Verständniskontrolle.** Warum ist $\ell \geq {}^3/_4 N$ uninteressant?
3. Sei $|\widetilde{x}\rangle = \frac{1}{\sqrt{\ell}} \sum_{i=1}^{\ell} |\widehat{x}_i\rangle$.
4. Wir wollen ähnlich vorgehen, wie im Fall mit einer Lösung und den Startzustand $|s\rangle$ so lange drehen, bis eine Messung mit hoher Wahrscheinlichkeit eines der $\widehat{x}_i$ liefert.
5. Dazu drehen wir den Zustand $|s\rangle$ bis er sehr nahe an $|\widetilde{x}\rangle$ liegt, und somit eine Messung mit sehr hoher Wahrscheinlichkeit eines der $\widehat{x}_1, \ldots, \widehat{x}_\ell$ liefern muss.
6. **Verständniskontrolle.** Warum können wir $|s\rangle$ nicht einfach auf einen bestimmten, willkürlich ausgewählten Punkt $|\widehat{x}_i\rangle$ drehen?

Grover-Algorithmus für mehrere Lösungen

1. Um $|s\rangle$ in Richtung $|\widetilde{x}\rangle$ zu drehen, gehen wir genauso vor, wie im Fall mit einer Lösung, nur mit $|\widetilde{x}\rangle$ in der Rolle von $|\widehat{x}\rangle$.
2. Also ersetzen wir $U_{\widehat{x}}$ durch $U_{\widetilde{x}}$, der Spiegelung an der Hyperebene, die auf der gleichmässigen Überlägerung der $|\widehat{x}_i\rangle$ senkrecht steht.
3. **Verständniskontrolle.** Wieso verfügen wir über einen Quantenschaltkreis, der $U_{\widetilde{x}}$ ausführt?
4. **Verständniskontrolle.** Falls ℓ groß ist, kann dieser Schaltkreis groß werden. Warum ist das kein Problem, zumindest nicht für die interessanten Anwendungen des Grover-Algorithmus'?

Grover-Algorithmus für mehrere Lösungen

1. Wieviele Grover-Iterationen werden bei ℓ Lösungen benötigt?
2. Sei analog zu oben

$$|u\rangle = \frac{1}{\sqrt{N-\ell}} \sum_{\substack{x=0,\\ x\neq\widehat{x}_1,\ldots,\widehat{x}_\ell}}^{N-1} |x\rangle = \sqrt{\frac{N}{N-\ell}}\,|s\rangle - \sqrt{\frac{\ell}{N-\ell}}\,|\widetilde{x}\rangle\,.$$

3. Dann gilt analog

$$|s\rangle = \sqrt{1-\frac{\ell}{N}}\,|u\rangle + \sqrt{\frac{\ell}{N}}\,|\widetilde{x}\rangle = \cos\frac{\theta}{2}\,|u\rangle + \sin\frac{\theta}{2}\,|\widetilde{x}\rangle\,.$$

4. $k \approx \frac{\pi-\theta}{2\theta}$ gilt noch, aber mit $\sin\frac{\theta}{2} = \sqrt{\frac{\ell}{N}}$.
5. Falls ℓ/N klein ist, folgt $G(N,\ell) := k \approx \frac{\pi}{4}\sqrt{\frac{N}{\ell}}$.

18 Quantenkryptographie

Quantenschlüsselaustausch

1. Es gibt gute klassische Verschlüsselungsverfahren, aber bei diesen gibt es immer das Problem des Schlüsselaustauschs.
2. Und genau dafür können Quantenphänomene genutzt werden.

Das BB84-Protokoll: Unsichere Version

1. Alice erzeugt Zufallsbits $a_1, \ldots, a_m$ und $a'_1, \ldots, a'_m$.
2. Alice sendet $|x_1 \ldots x_m\rangle$ an Bob, wobei

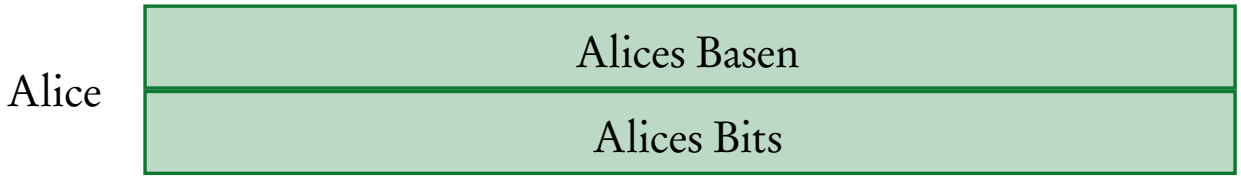
$$|x_i\rangle = \begin{cases} |0\rangle & \text{falls } (a_i, a'_i) = (0,0), \\ |1\rangle & \text{falls } (a_i, a'_i) = (1,0), \\ |+\rangle & \text{falls } (a_i, a'_i) = (0,1), \\ |-\rangle & \text{falls } (a_i, a'_i) = (1,1), \end{cases} \quad \text{for } 1 \leq i \leq m.$$

3. Bob erzeugt Zufallsbits $b'_1, \ldots, b'_m$.
4. Für $1 \leq i \leq m$, falls $b'_i = 0$, misst Bob $|x_i\rangle$ in Basis $\{|0\rangle, |1\rangle\}$,
   sonst misst Bob $|x_i\rangle$ in Basis $\{|+\rangle, |-\rangle\}$,
   und speichert das jeweilige Ergebnis als b_i.
5. Alice und Bob tauschen $a'_1, \ldots, a'_m$ und $b'_1, \ldots, b'_m$ aus.
6. Der Schlüssel ist $a_{i_1} \ldots a_{i_\ell}$ mit $\{i_1 < \ldots < i_\ell\} = \{i \colon a'_i = b'_i\}$.

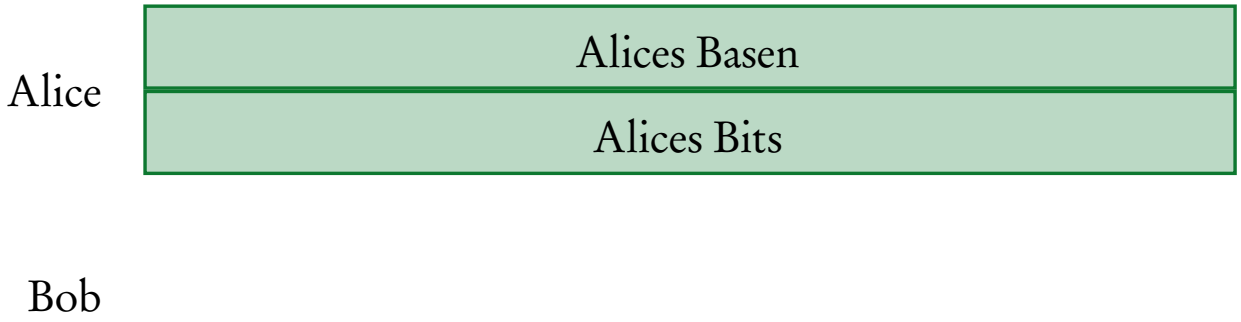
Zu erwartender Ablauf

Alice

Zu erwartender Ablauf

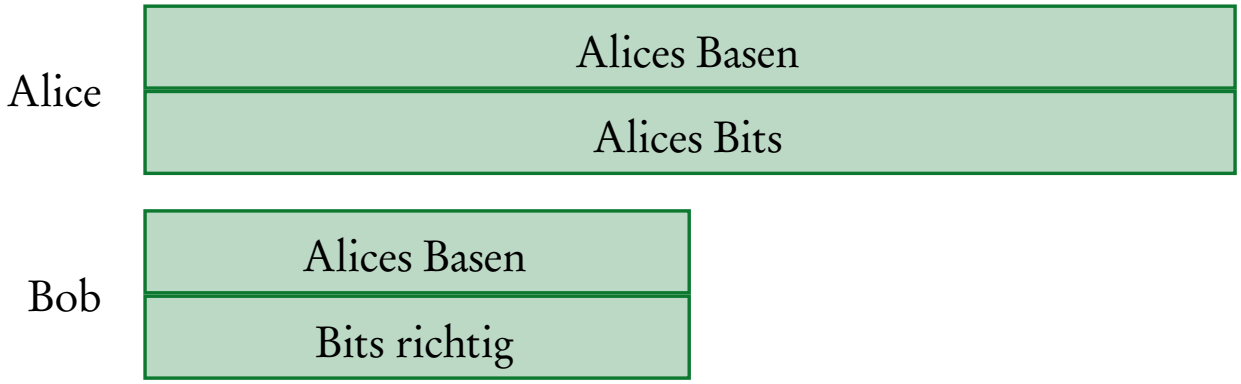


Zu erwartender Ablauf

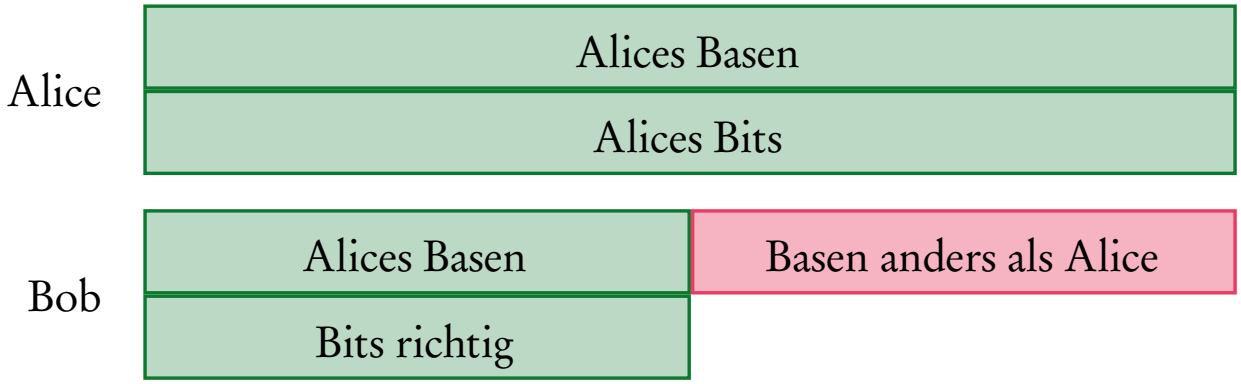


Zu erwartender Ablauf

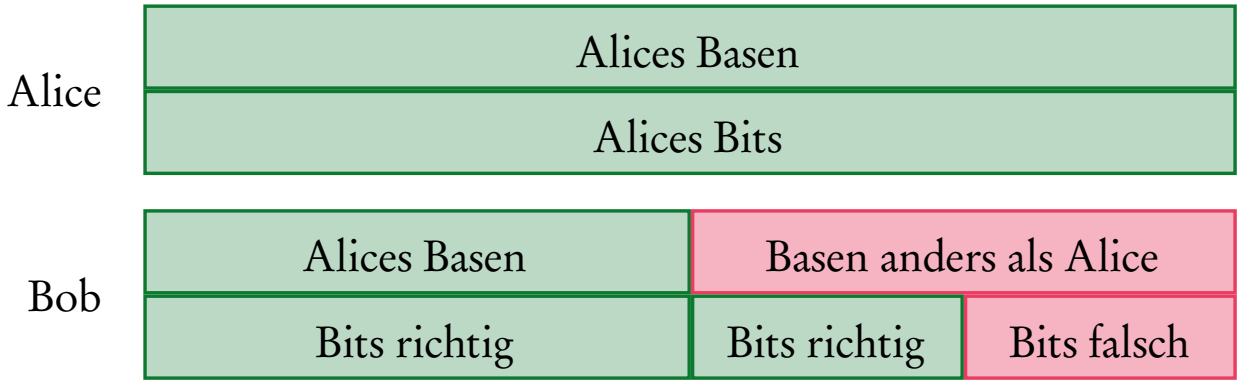
Zu erwartender Ablauf



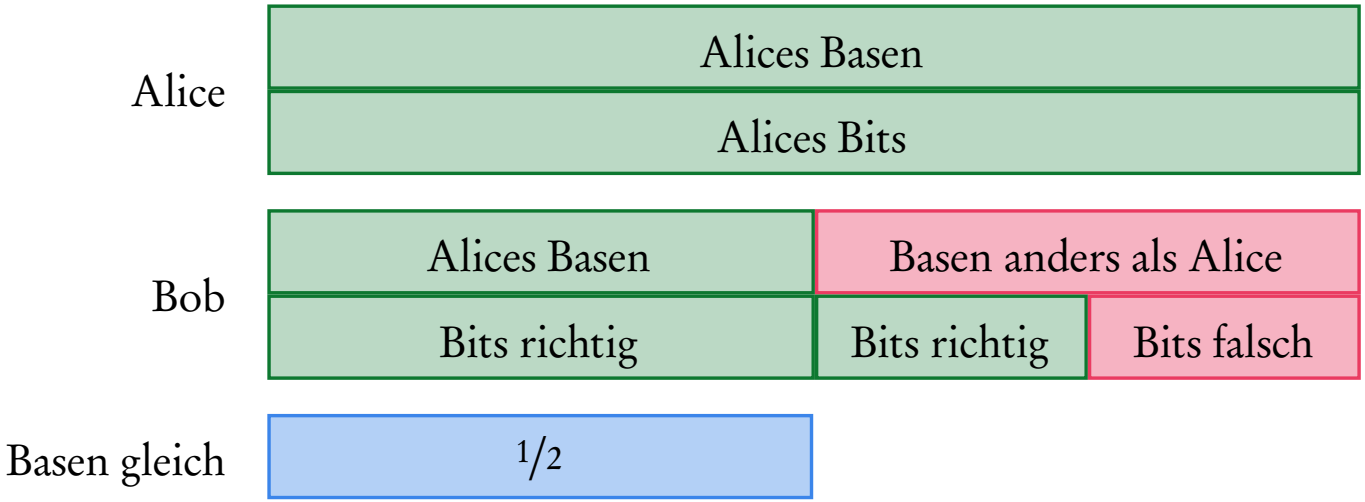
Zu erwartender Ablauf



Zu erwartender Ablauf



Zu erwartender Ablauf



Grundidee

1. Eve kann die Übermittlung der Quantenbits vielleicht belauschen, weiß aber nicht, in welcher Basis sie messen muss.
2. Erst sobald es für Eve zu spät ist, tauschen Alice und Bob aus, welche Basen sie benutzt haben. Immer wenn sie die gleiche Basis benutzt haben, liefert Bobs Messung das versandte Bit.
3. Diese Bits können also als Schlüssel genutzt werden; im Schnitt ist es die Hälfte der versandten Bits.
4. Startet Alice mit $m = 2n + \Delta$ Bits, mit Δ groß genug, so hat der Schlüssel mit hoher Wahrscheinlichkeit Länge $\geq n$.

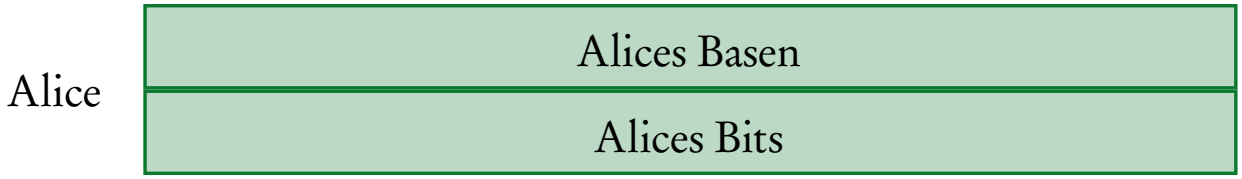
Angriff

1. Wie könnte Eve dieses Protokoll angreifen?
2. Sie könnte die Bits abfangen, messen, dann weiterschicken.
   (Oder äquivalent, neue Bits gemäß ihrer Messwerte erzeugen und verschicken.)
3. Da sie aber nicht weiß, mit welcher Basis sie die Bits messen muss, wird sie oft in der falschen Basis messen, und damit nur einen Teil des Schlüssels erhalten.
4. Wie viel kann sie über den Schlüssel lernen? Und wie wirkt sich das auf das Protokoll aus?

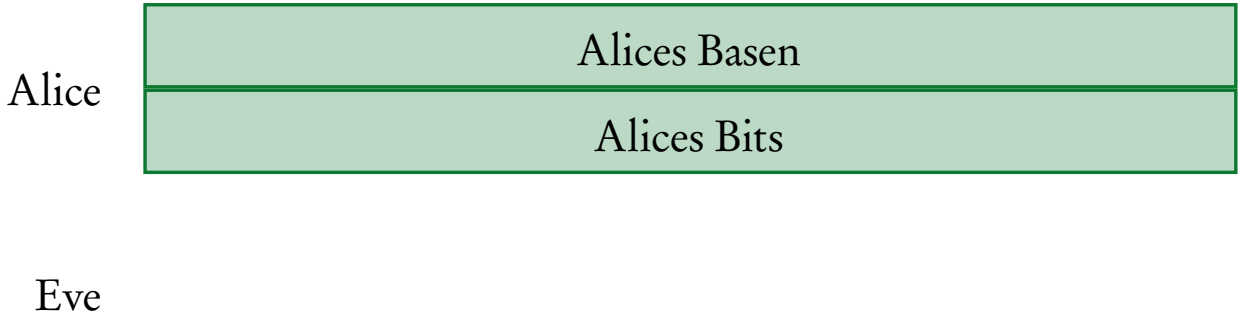
Zu erwartender Ablauf mit Lauscher

Alice

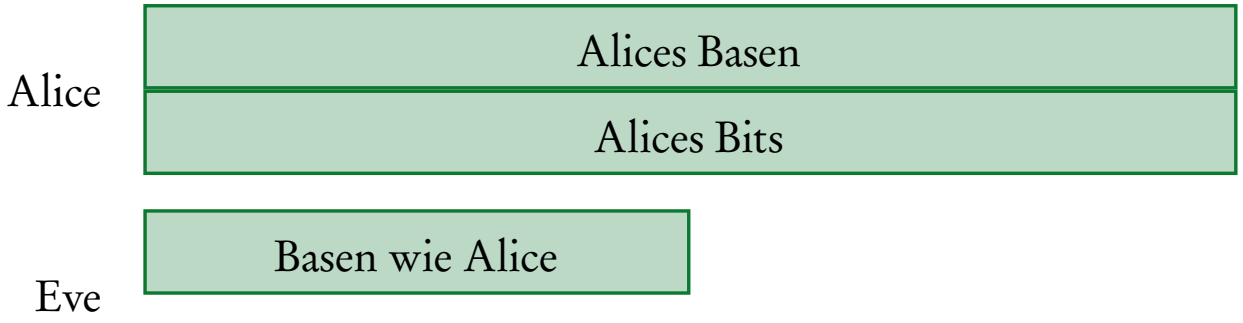
Zu erwartender Ablauf mit Lauscher



Zu erwartender Ablauf mit Lauscher

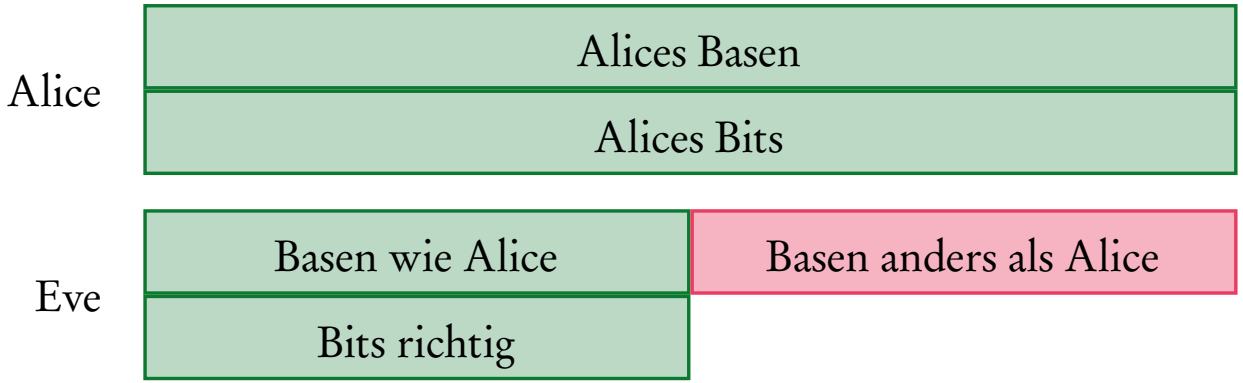


Zu erwartender Ablauf mit Lauscher

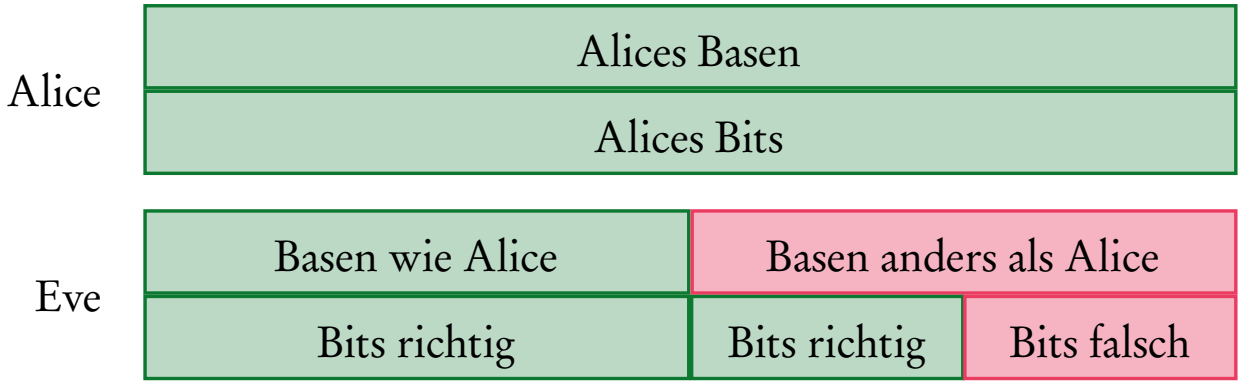


Zu erwartender Ablauf mit Lauscher

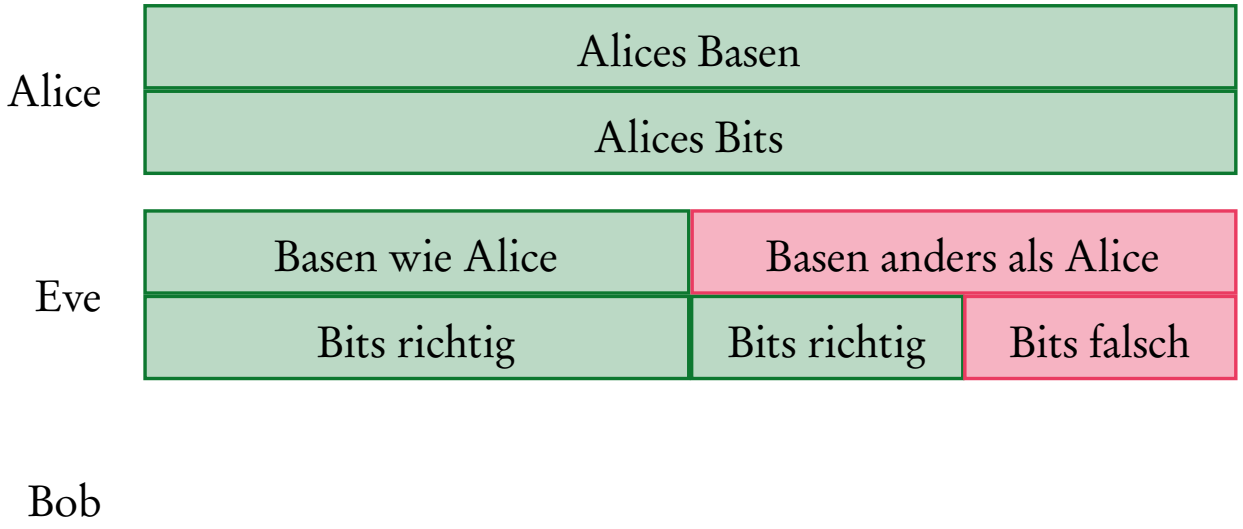
Zu erwartender Ablauf mit Lauscher



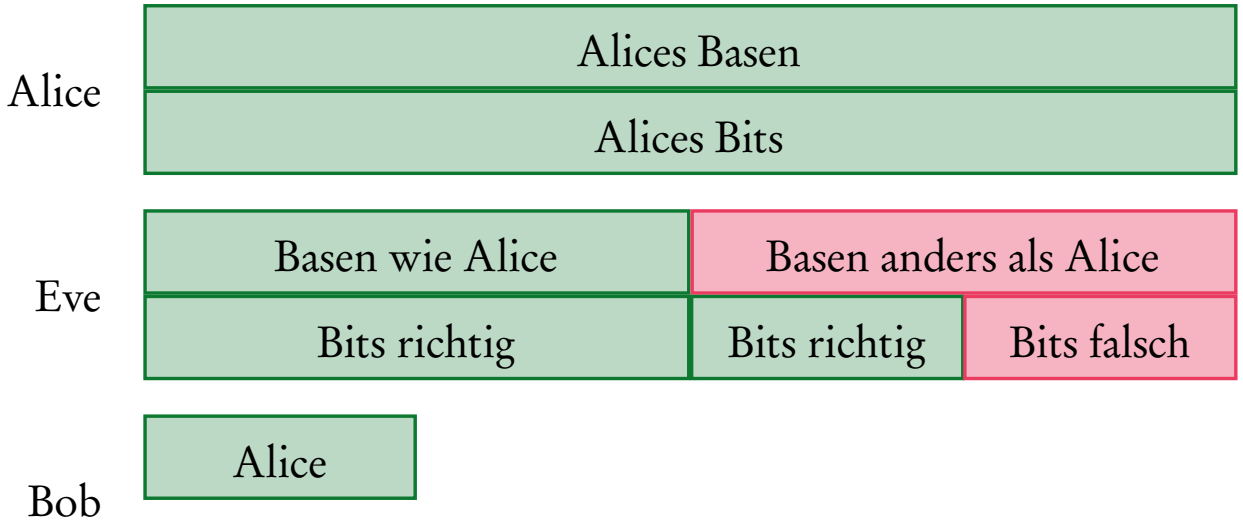
Zu erwartender Ablauf mit Lauscher



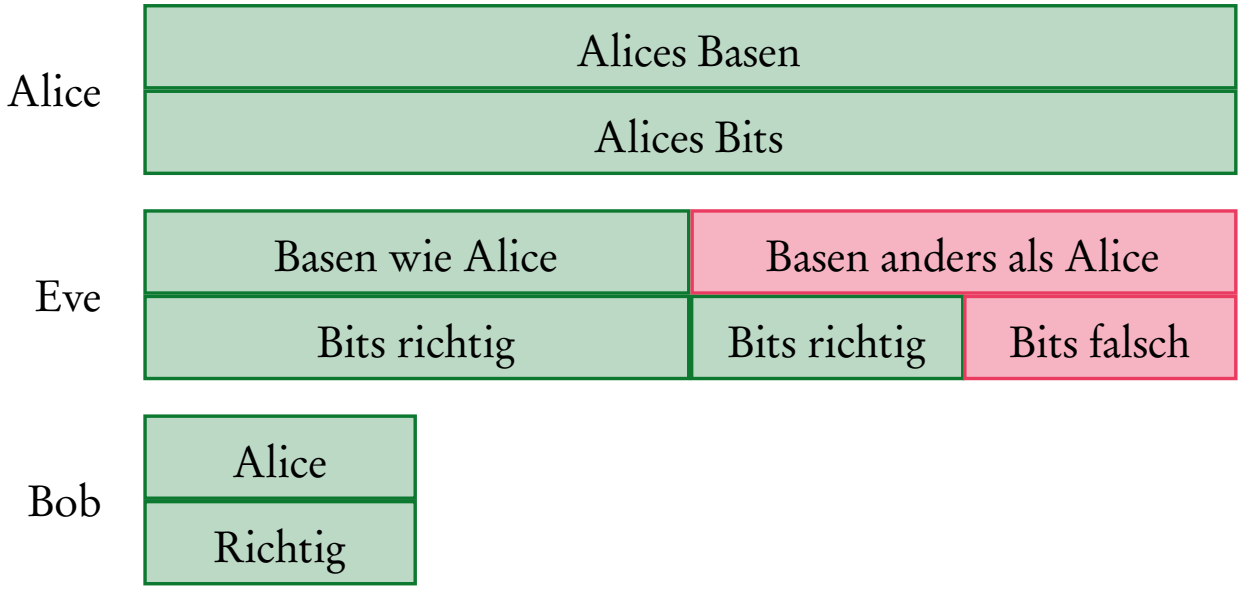
Zu erwartender Ablauf mit Lauscher



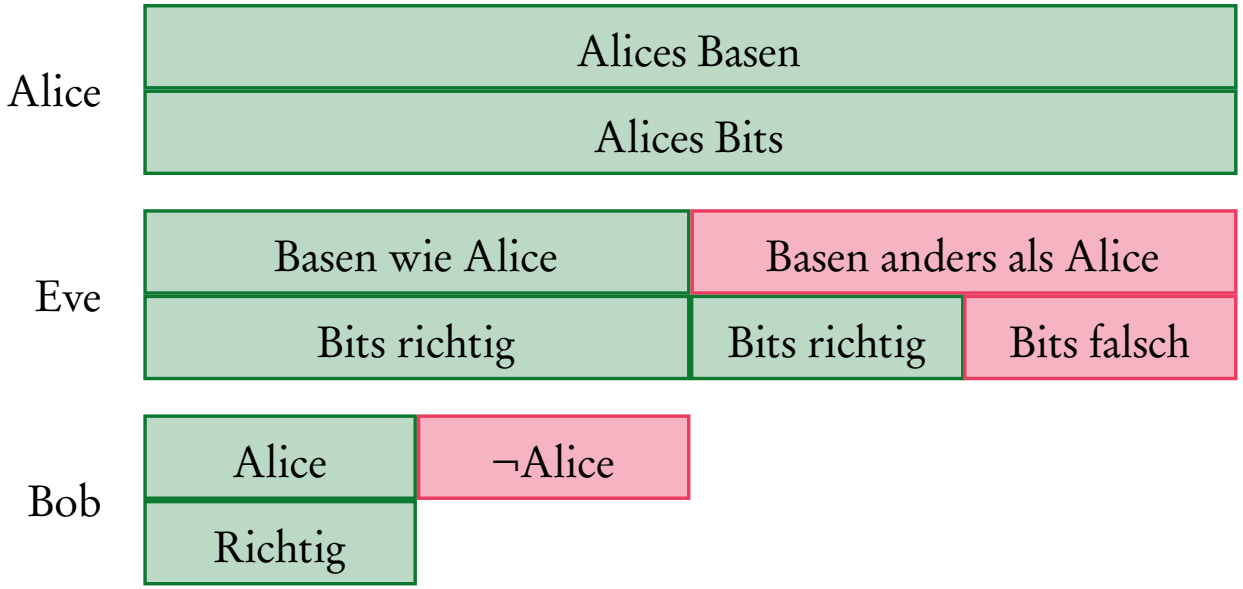
Zu erwartender Ablauf mit Lauscher



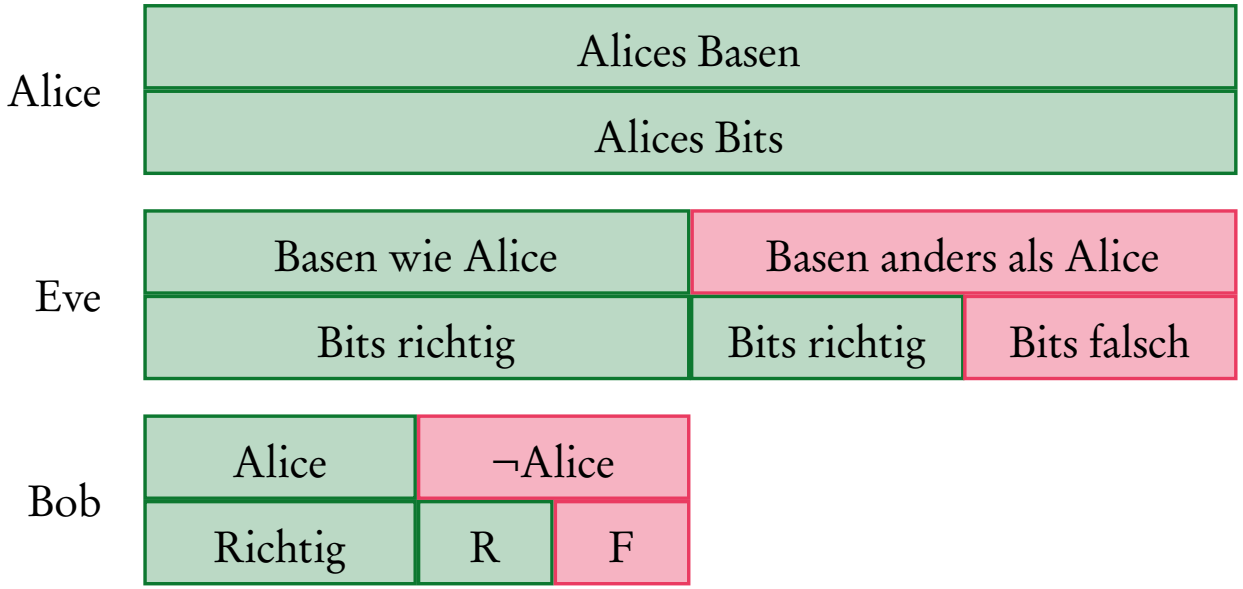
Zu erwartender Ablauf mit Lauscher



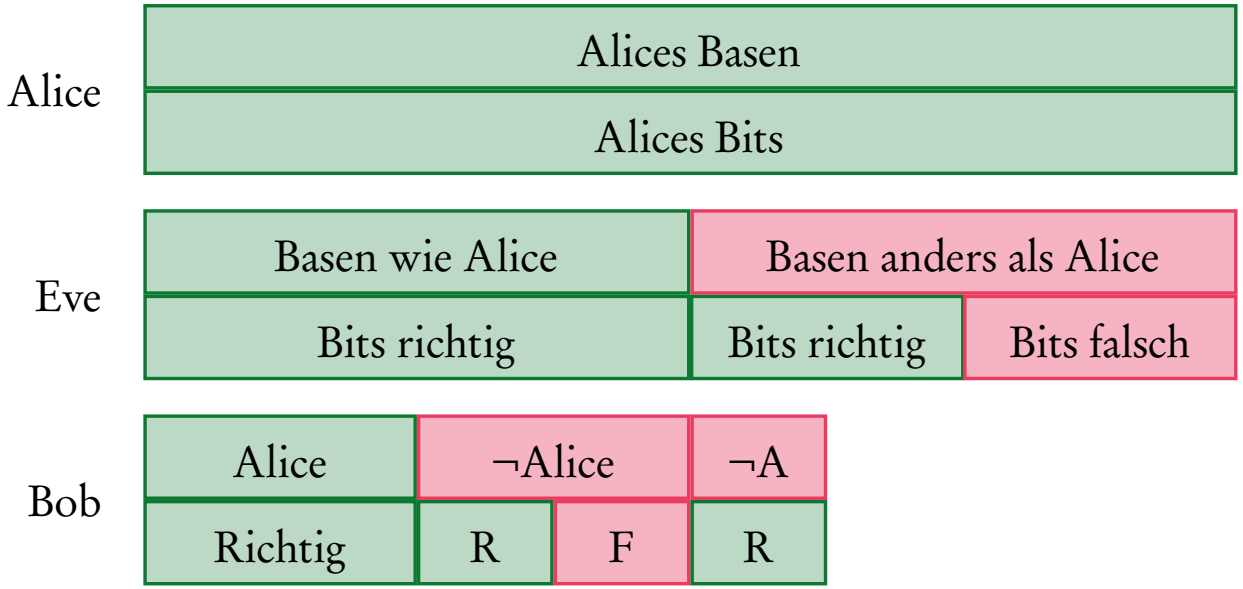
Zu erwartender Ablauf mit Lauscher



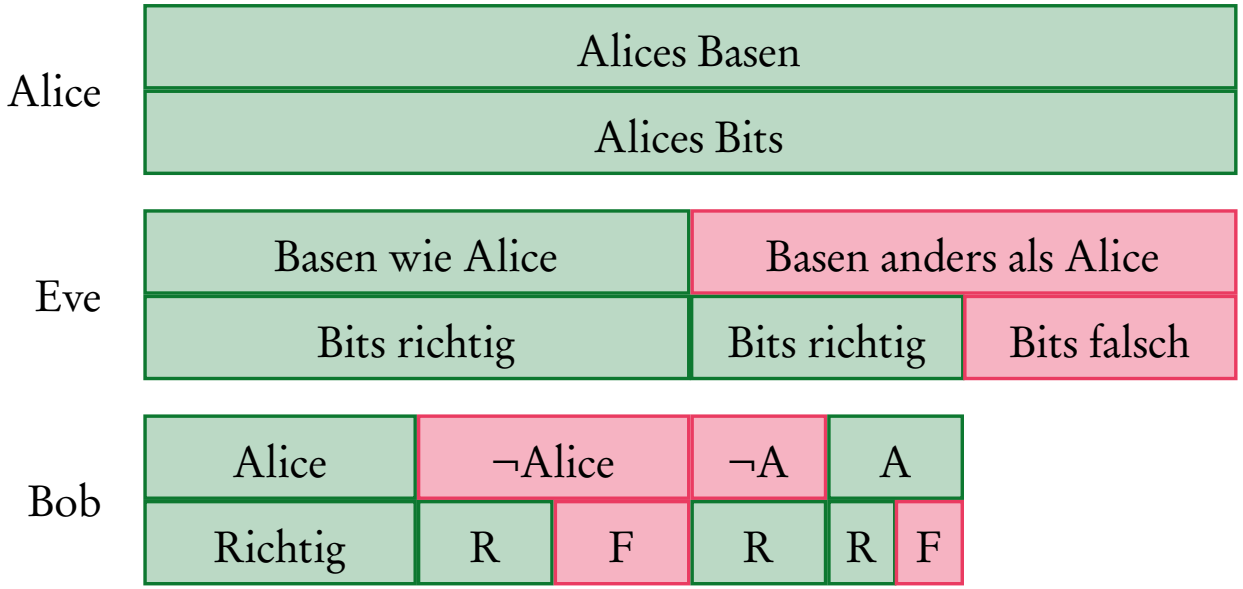
Zu erwartender Ablauf mit Lauscher



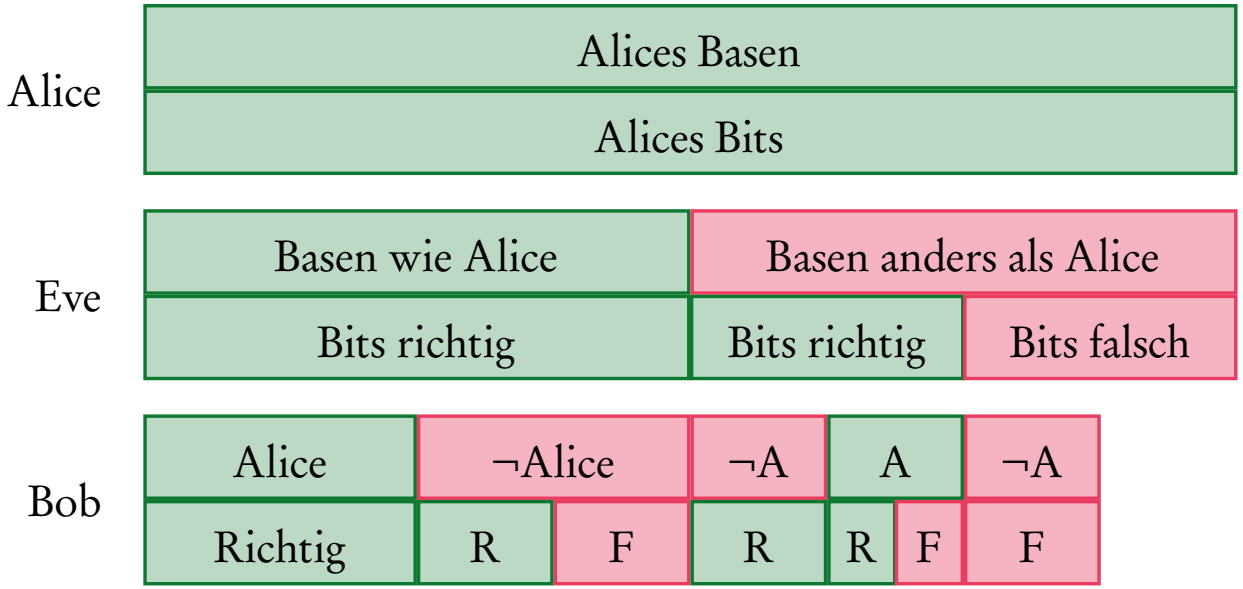
Zu erwartender Ablauf mit Lauscher



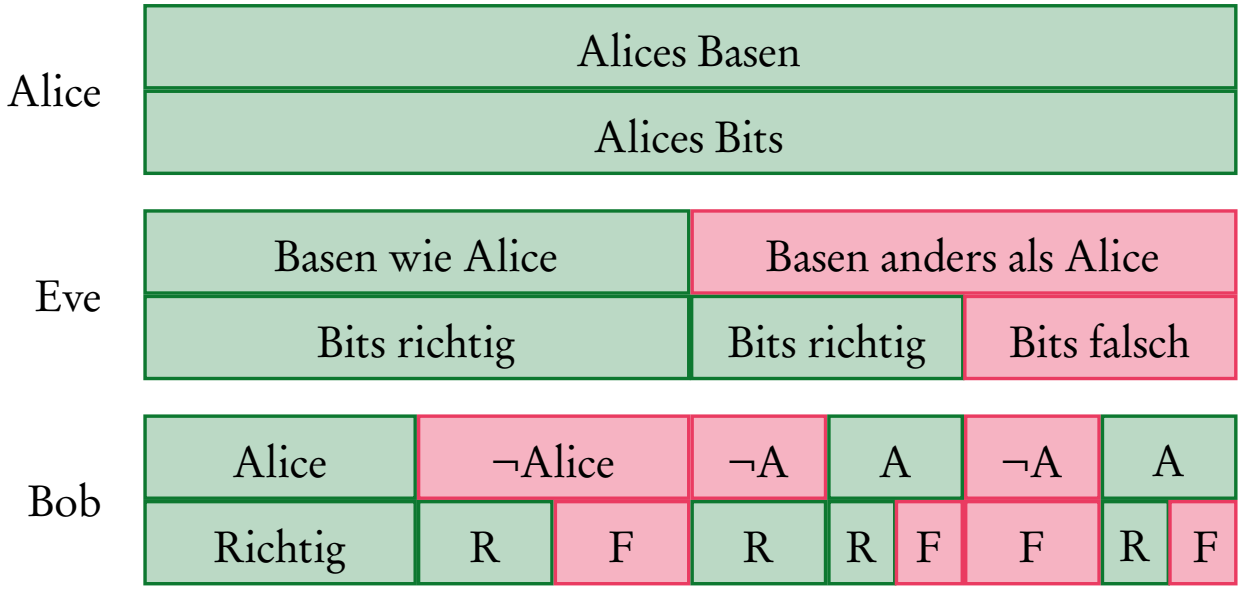
Zu erwartender Ablauf mit Lauscher



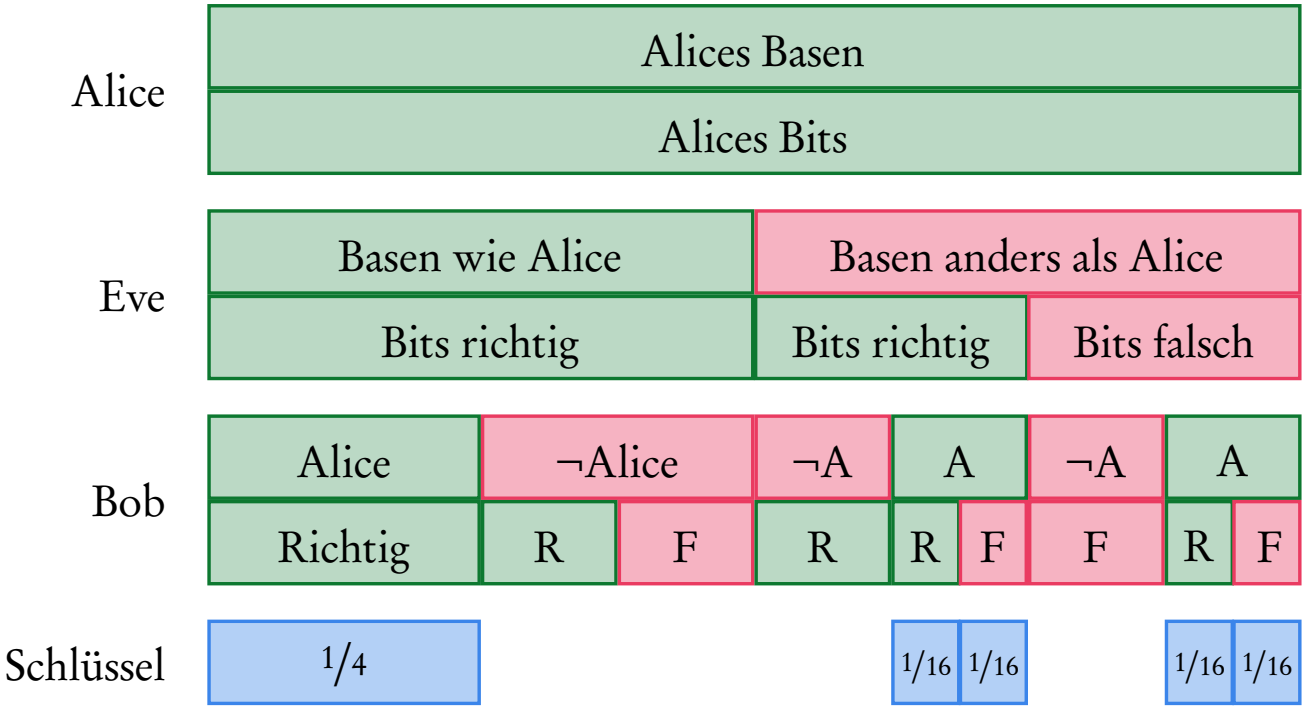
Zu erwartender Ablauf mit Lauscher



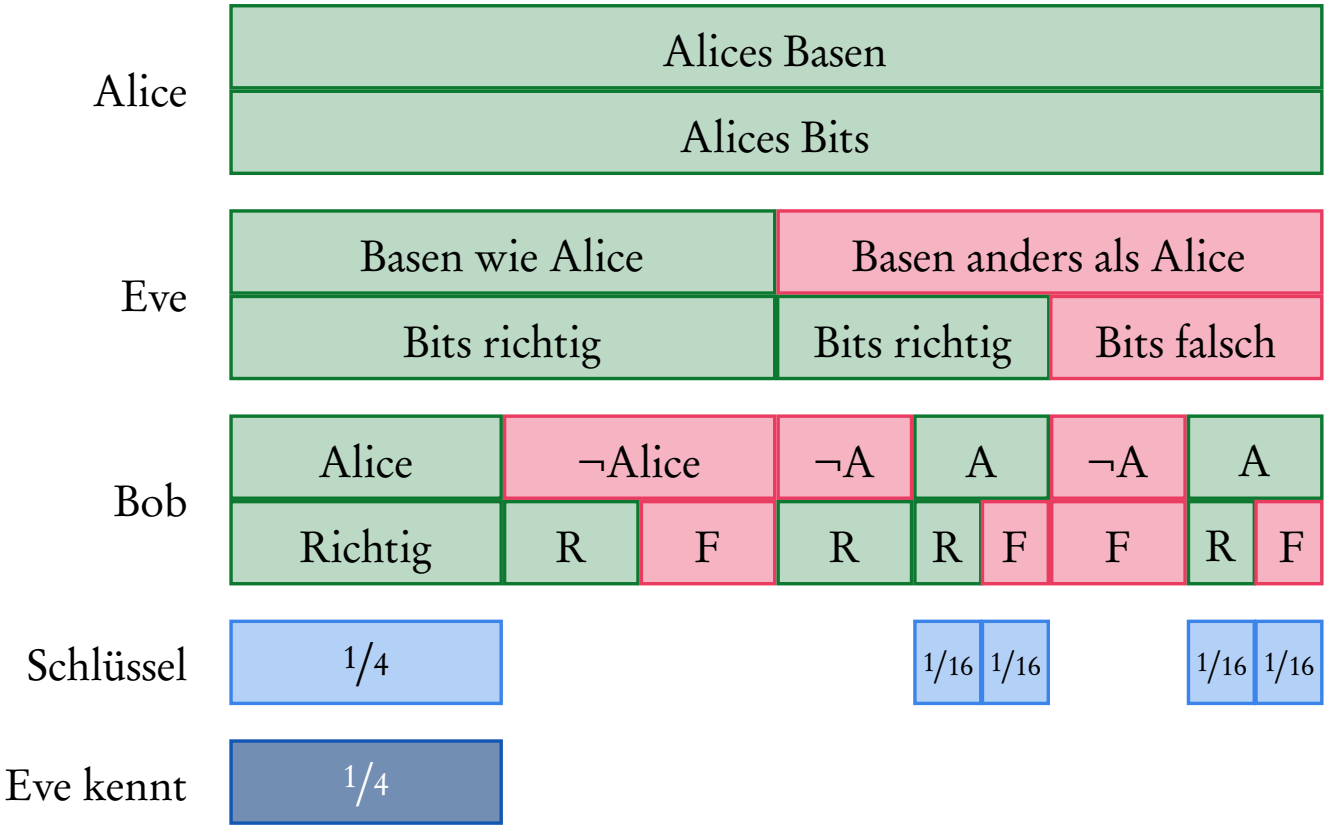
Zu erwartender Ablauf mit Lauscher



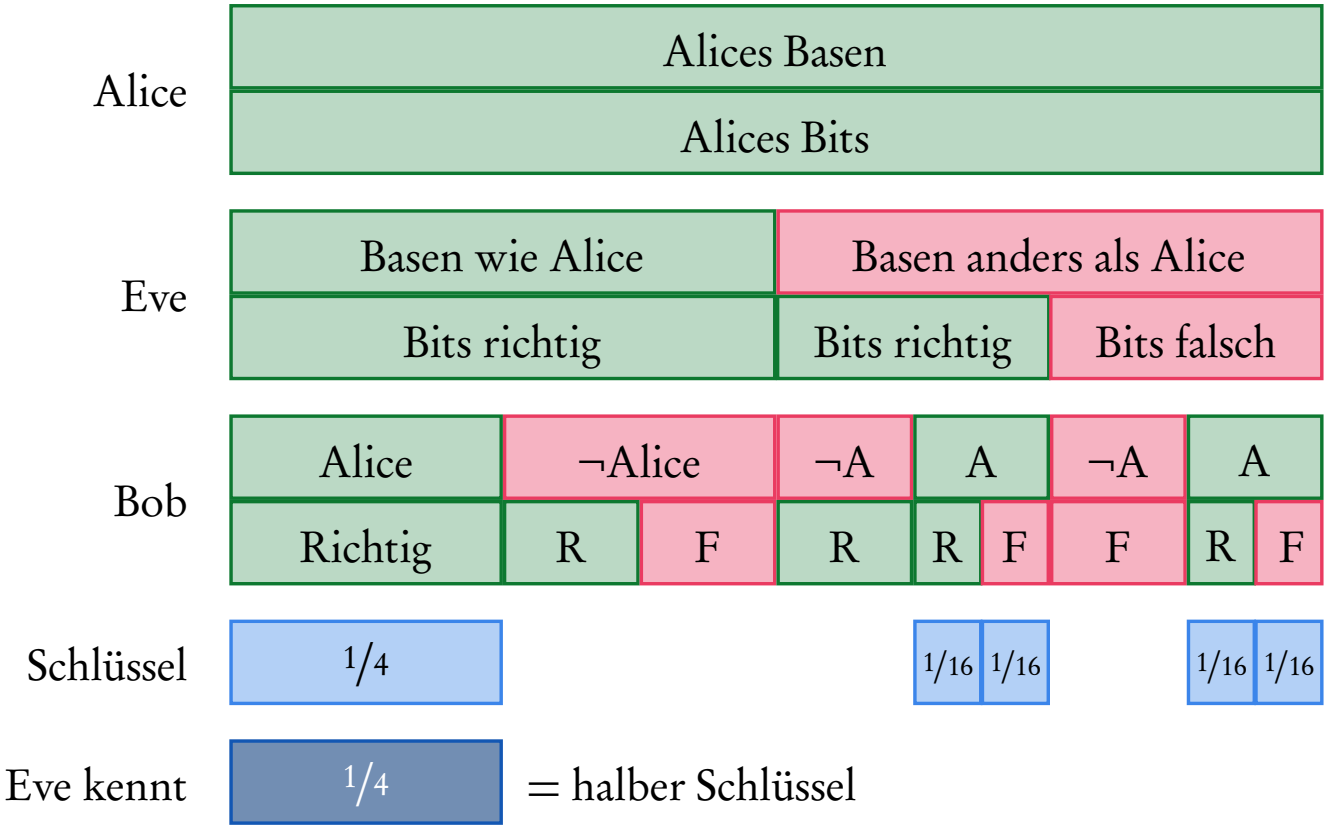
Zu erwartender Ablauf mit Lauscher



Zu erwartender Ablauf mit Lauscher



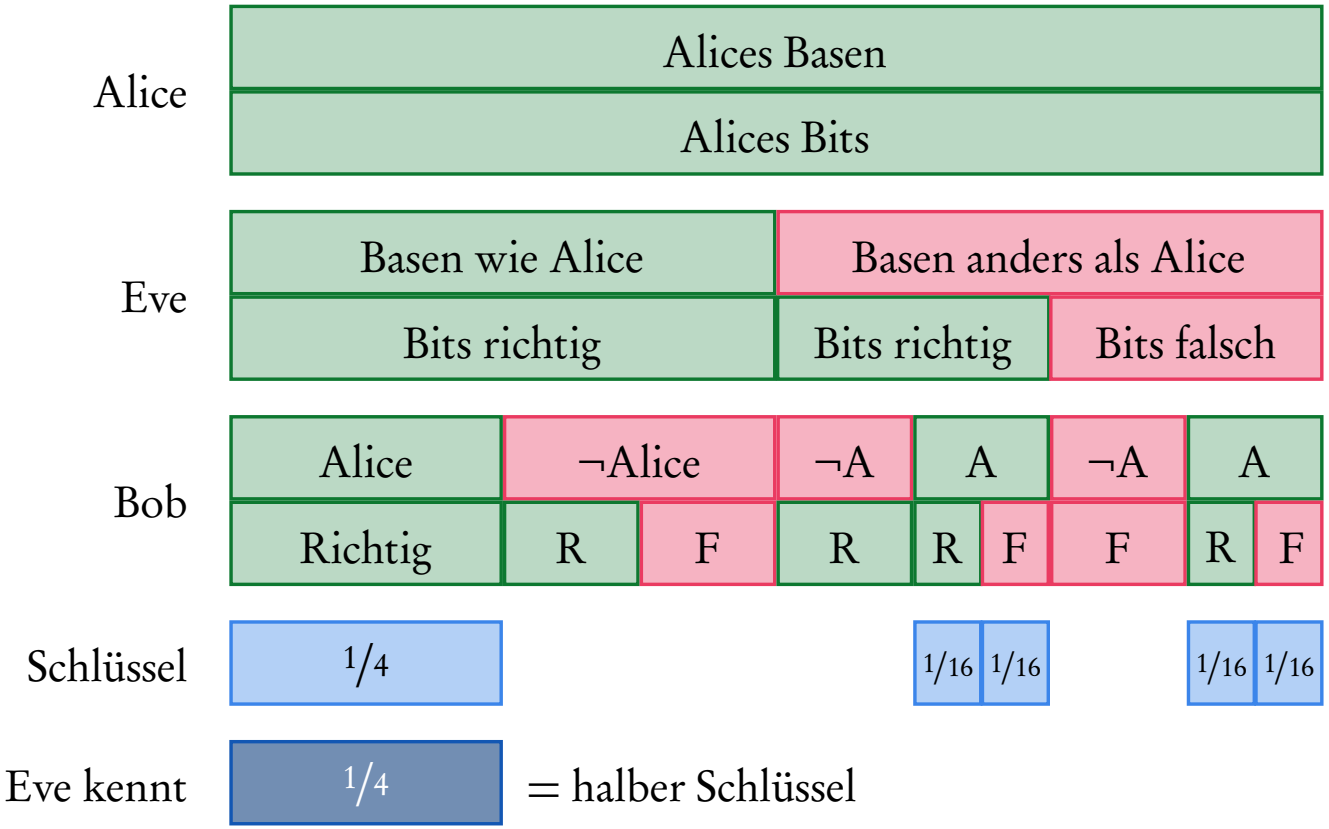
Zu erwartender Ablauf mit Lauscher



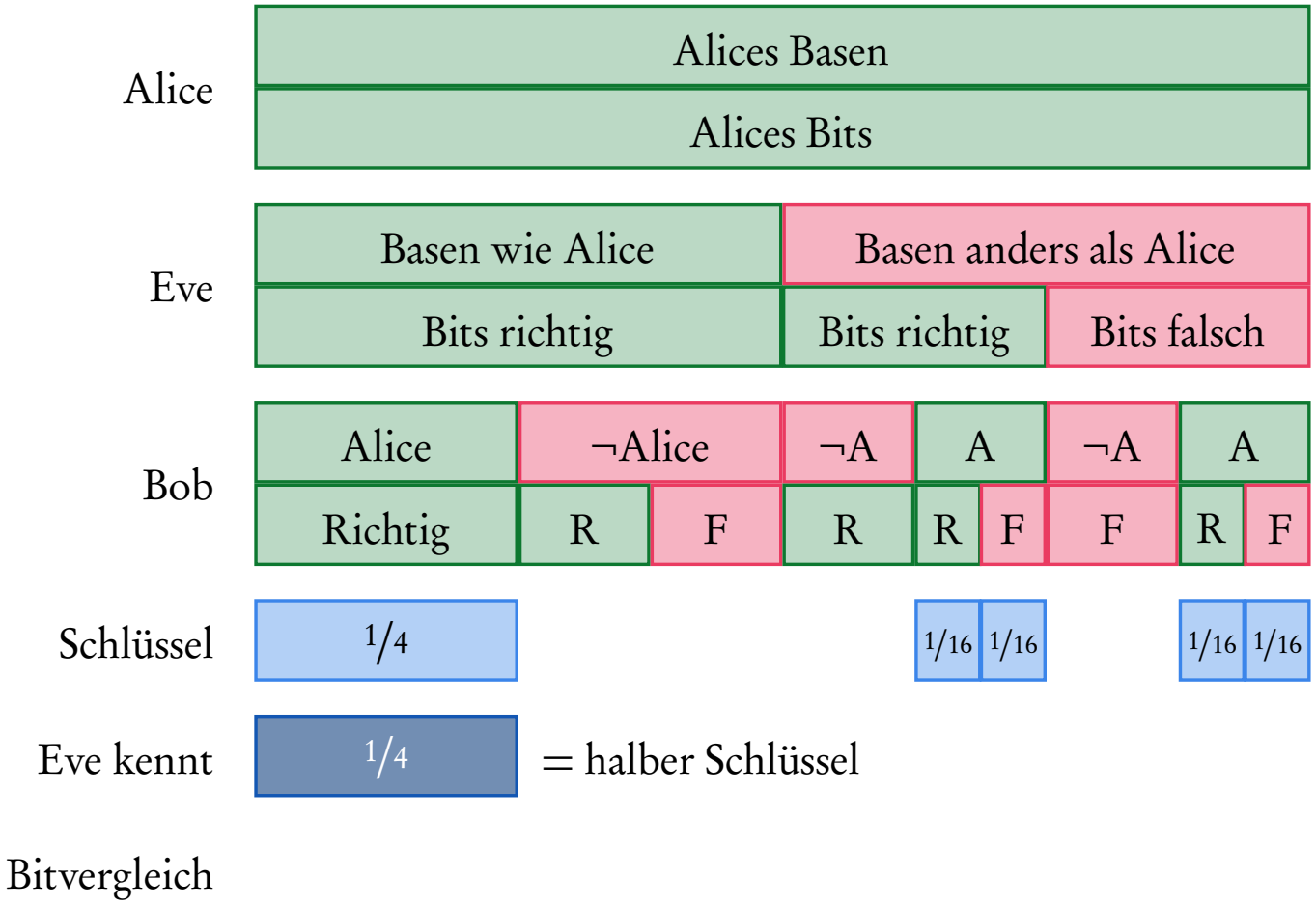
Sicherheit herstellen durch Bitvergleich

1. Eve erhält also die Hälfte des Schlüssels; das ist zu viel. Was können Alice und Bob dagegen tun?
2. Das Abhören hat die von Bob gemessenen Bits verfälscht. Ohne dass Alice und Bob es wissen, haben sie gar nicht den gleichen Schlüssel bestimmt!
3. Wie können sie dies feststellen, ohne die geheimen Bits über einen möglicherweise unsicheren Kanal aneinander zu schicken?
4. Die Idee ist, zufällig ausgewählte Bits zu opfern, und diese über den unsicheren Kanal zu vergleichen. Diese Bits dürfen natürlich dann nicht mehr zur Verschlüsselung verwendet werden.
5. Falls sich die Bits unterscheiden, gab es einen Angriff.

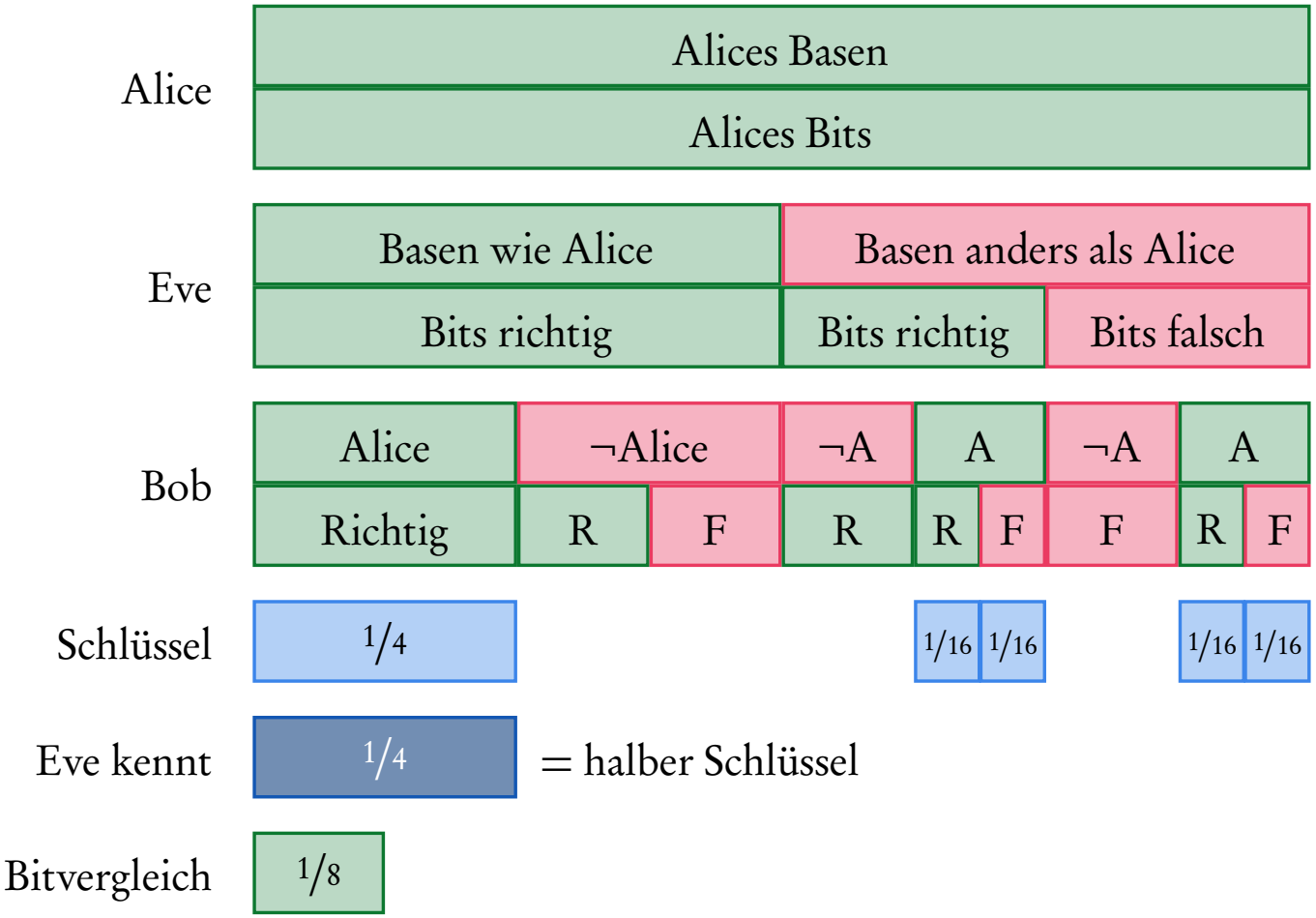
Sicherheit herstellen durch Bitvergleich



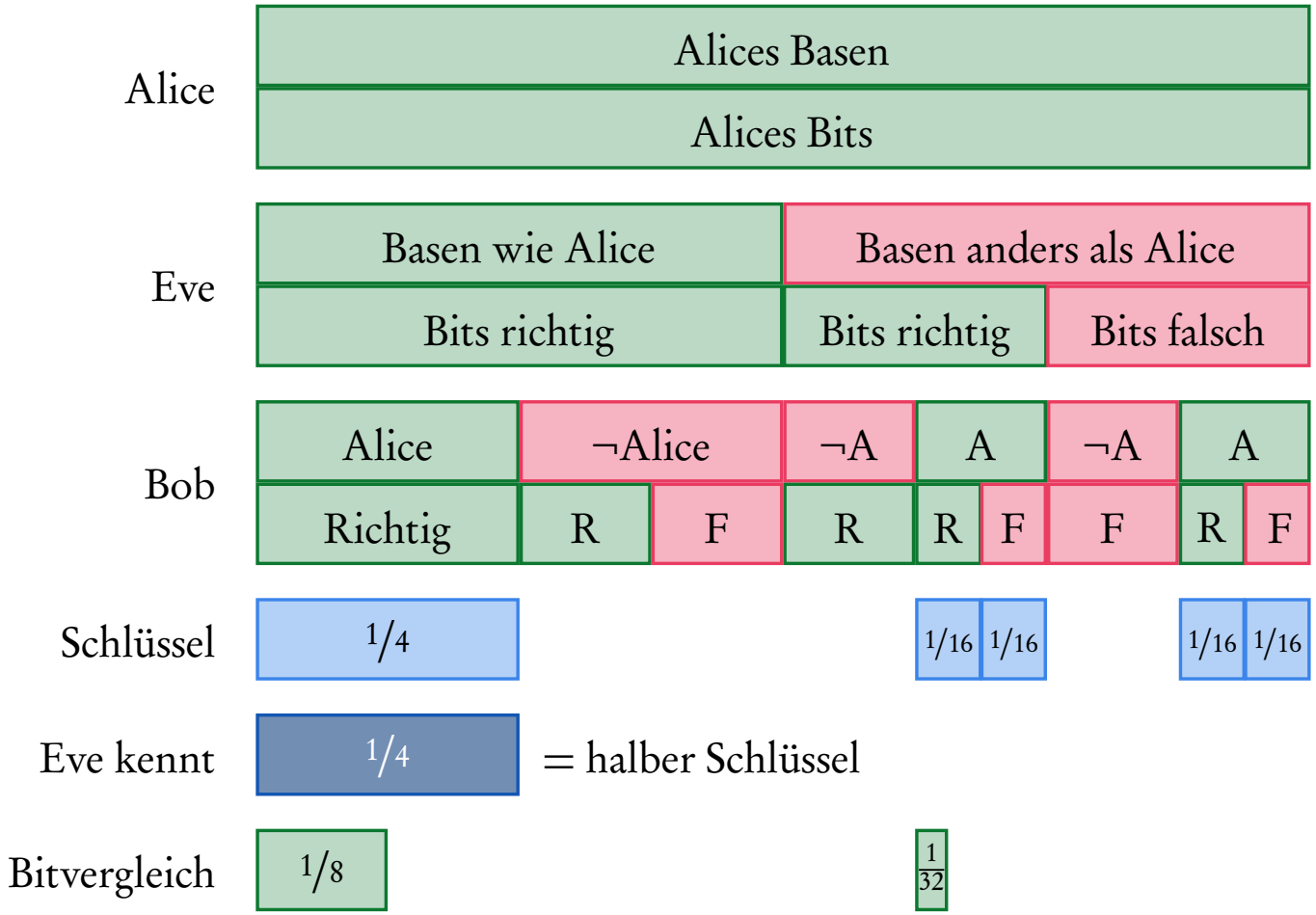
Sicherheit herstellen durch Bitvergleich



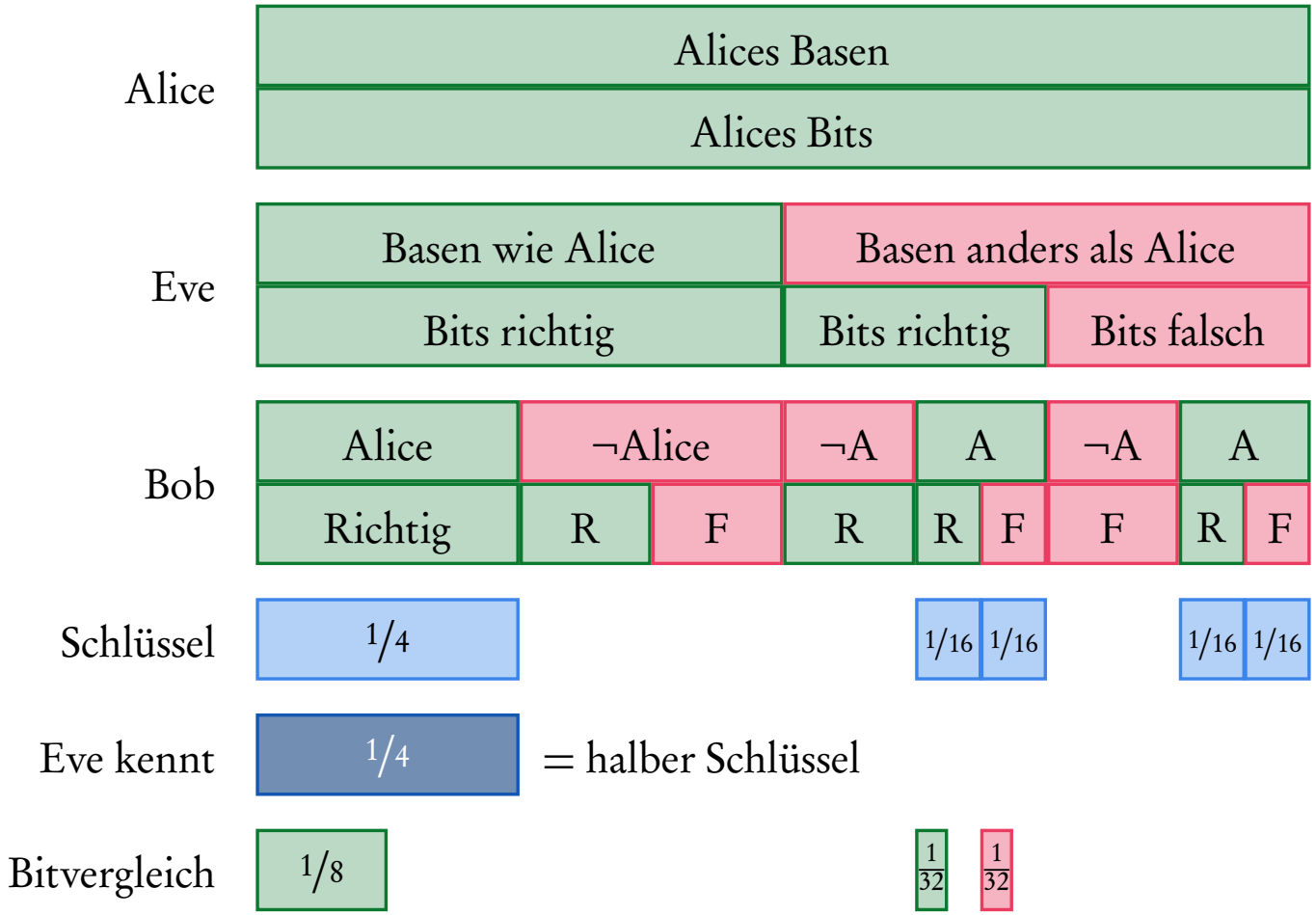
Sicherheit herstellen durch Bitvergleich



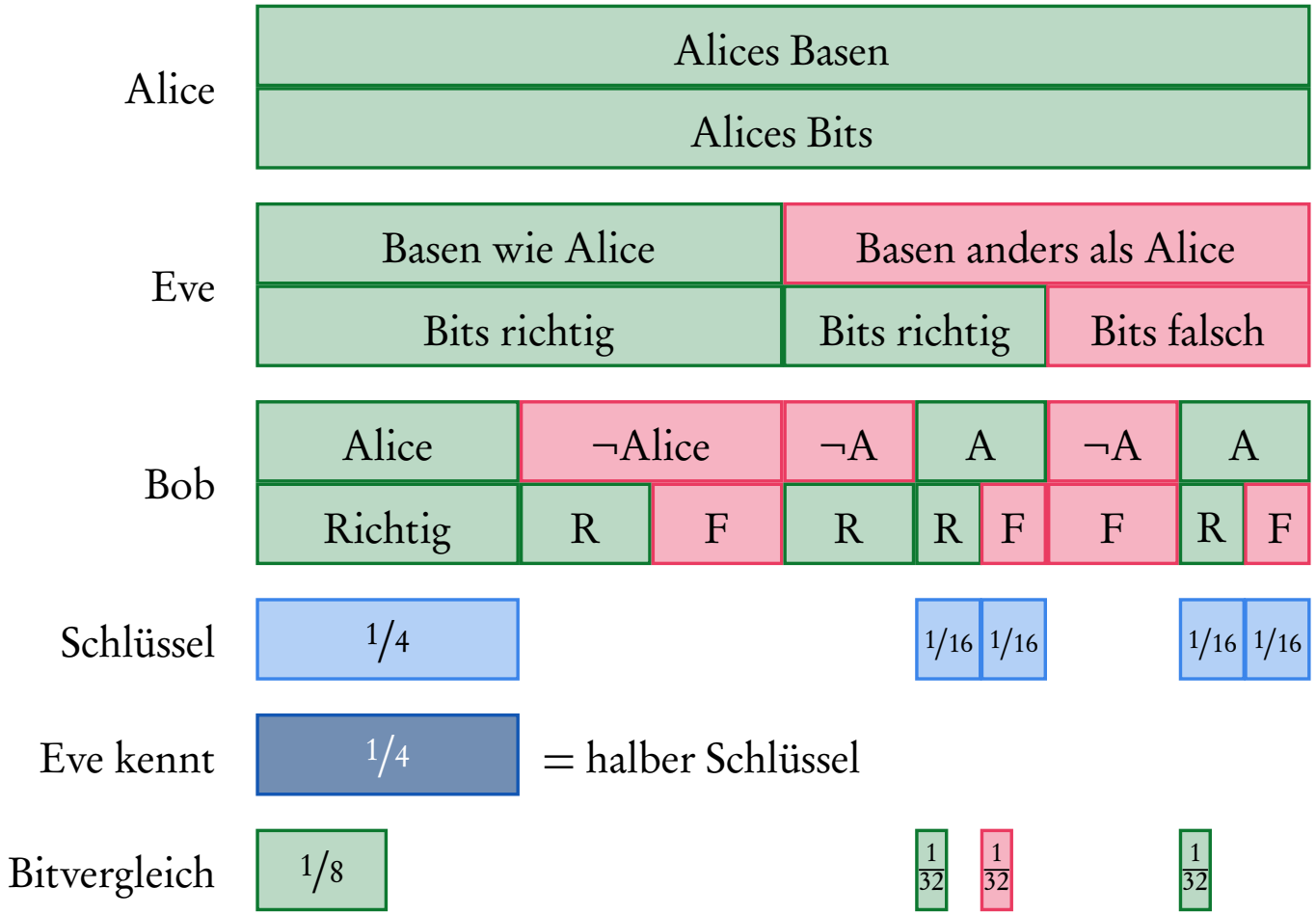
Sicherheit herstellen durch Bitvergleich



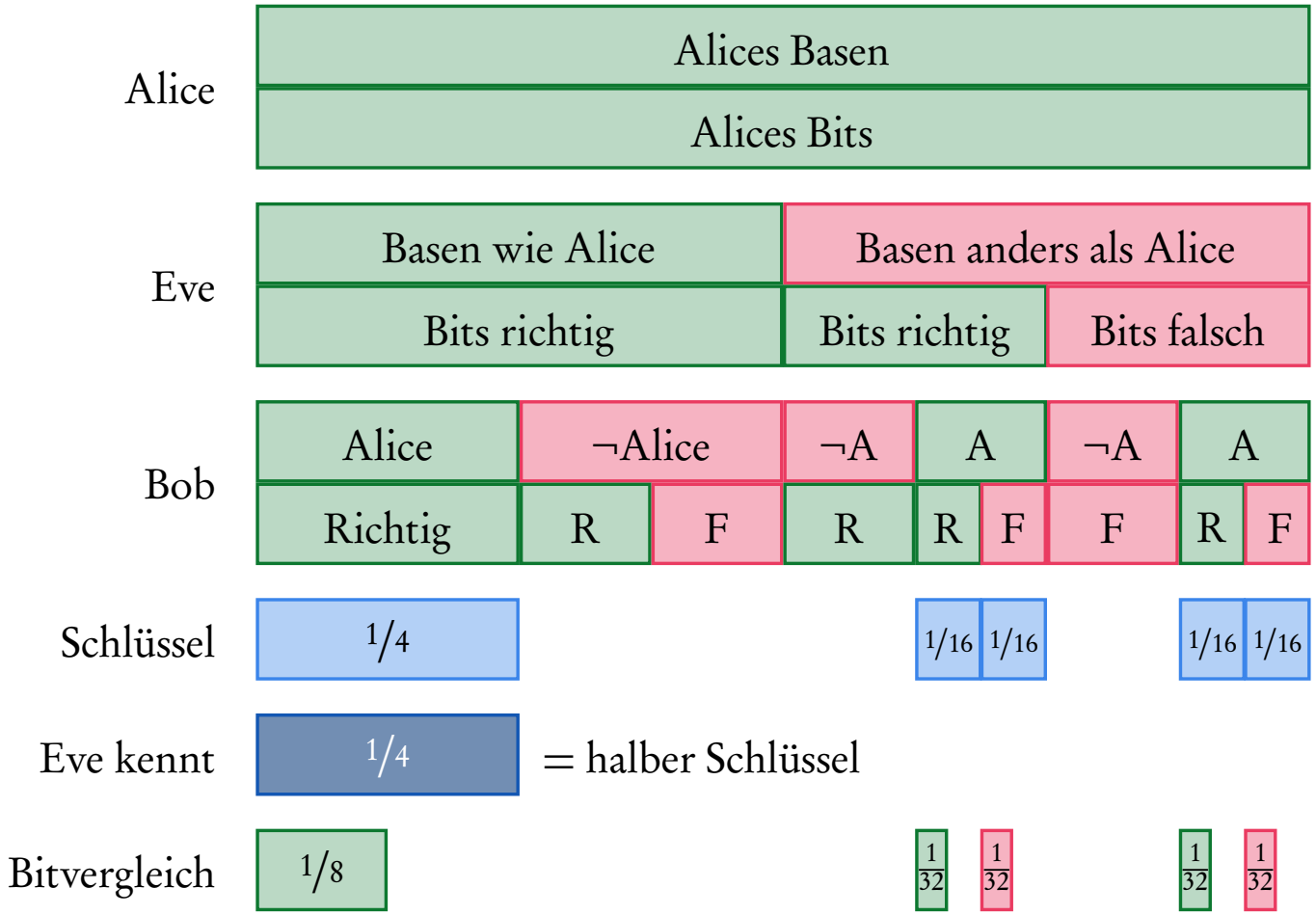
Sicherheit herstellen durch Bitvergleich



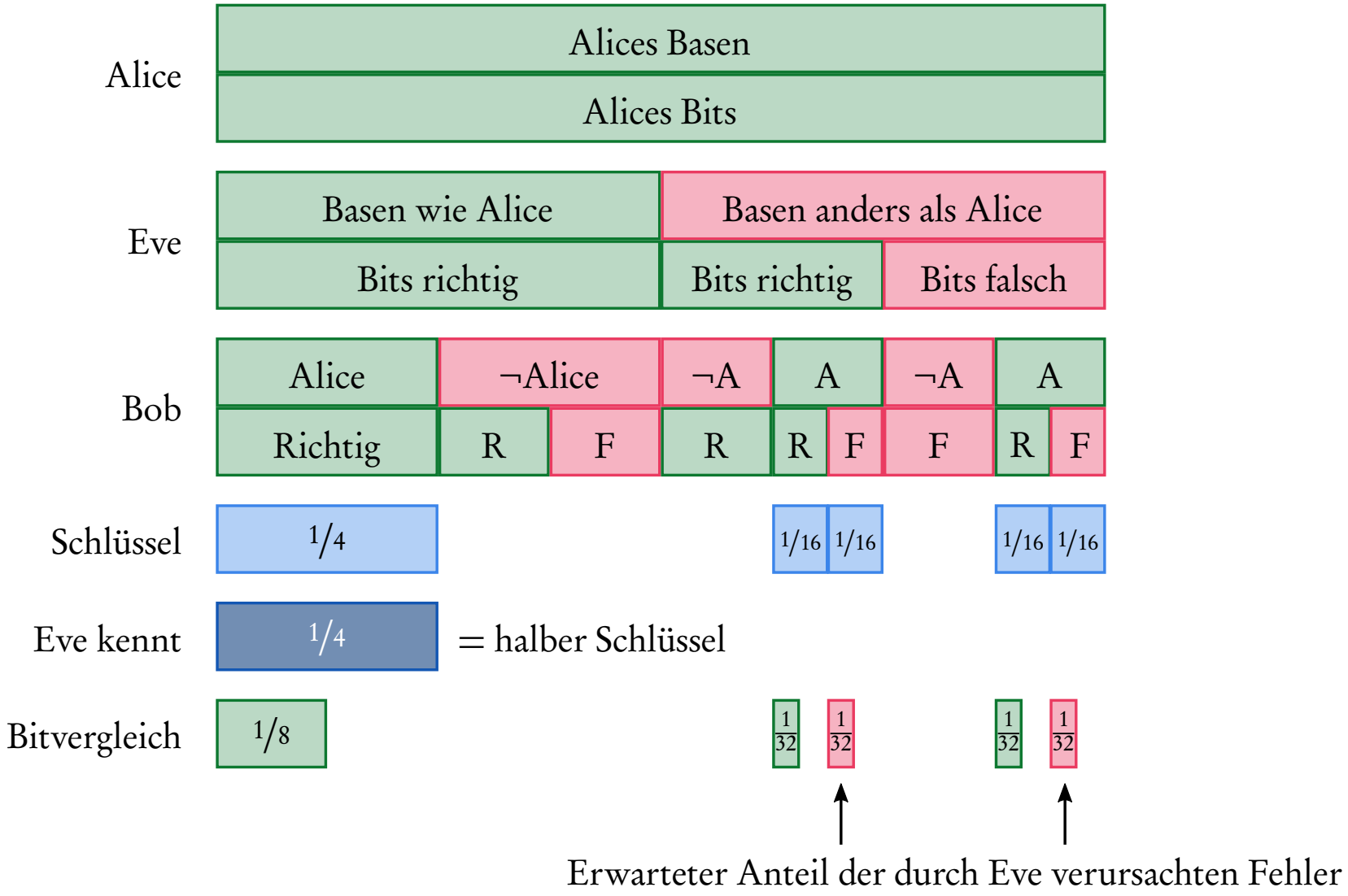
Sicherheit herstellen durch Bitvergleich



Sicherheit herstellen durch Bitvergleich



Sicherheit herstellen durch Bitvergleich



Vorsichtigere Angriffe

1. Eve kann vorsichtig sein, und nur manche Bits belauschen.
2. Die Zahlenverhältnisse ändern sich dadurch aber nicht.
3. Das Risiko sinkt, erwischt zu werden, aber dafür erhält Eve auch entsprechend weniger Information über den geheimen Schlüssel.
4. Angenommen die Quantenleitung ist völlig rauschfrei, dann reicht sogar ein einziger Fehler, damit Eve überführt wird.
5. In der Realität wird dem nicht so sein. Doch es kann eine ausreichend gute Qualität erreicht werden, dass Eve sich nur sehr wenige abgehörte Bits erlauben kann.
6. Damit Alice und Bob den gleichen Schlüssel erhalten, werden fehlerkorrigierende Codes eingesetzt.
7. Alice und Bob können die Sicherheit noch weiter erhöhen.

Privacy Amplification

1. Nimm an, Alice und Bob haben zwei Bits b_1 und b_2.
2. Sie wollen sie als Schlüssel benutzen, fürchten aber, dass Eve eines davon belauscht haben könnte. Sie wissen nicht, welches.
3. Alice und Bob benutzen stattdessen $b_3 = b_1 \oplus b_2$.
4. Angenommen Eve kennt b_1, aber nicht b_2. Der Wert von b_3 ist völlig unabhängig von b_1. Eves Wissen hilft ihr nicht.
5. Analog nützt es Eve nichts, wenn sie nur b_2 kennt.
6. Sie müsste für Rückschlüsse beide Bits kennen.

Privacy Amplification

1. Analog weiß Eve *nichts* über $b_1 \oplus b_2 \oplus \ldots \oplus b_k$, wenn sie *ein einziges* b_i, $1 \leq i \leq k$, nicht kennt.
2. Um aus $A = \{b_1, \ldots, b_m\}$ einen kürzeren Schlüssel mit n Bits zu erzeugen, wählt man n zufällige Teilmengen $A_1, \ldots, A_n \subset A$ aus. Der neue Schlüssel ist dann $\{\bigoplus_{b \in A_1} b, \ldots, \bigoplus_{b \in A_n} b\}$.
3. Das kann über einen öffentlichen Kanal koordiniert werden.
4. **Verständniskontrolle.** Warum ist die Zufälligkeit erforderlich?

Privacy Amplification

1. Beim Bitvergleich stellen Alice und Bob die Fehlerrate fest.
2. Sicherheitshalber müssen sie davon ausgehen, dass alle Fehler durch Eve verursacht wurden (und nicht durch Rauschen).
3. Je höher die Fehlerrate, desto mehr Bits könnte Eve kennen.
4. Entsprechend stark muss der Schlüssel gekürzt werden.
5. So kann die Wahrscheinlichkeit, dass Eve viele Bits des endgültigen Schlüssels kennt, erheblich reduziert werden.
6. **Verständniskontrolle.** Warum kann 0 nicht erreicht werden?
7. Wir ersparen uns die genauen Abschätzungen.

Allgemeinere Angriffe

1. Wir haben angenommen, dass Eve ein Bit misst und weiterleitet (oder äquivalent, ein neues Bit gemäß der Messung erzeugt und verschickt).
2. Da Alice und Bob sich erst danach über die Messbasen austauschen, konnte Eve nicht vermeiden, in der Hälfte der Fälle in der falschen Basis zu messen.
3. **Verständniskontrolle.** Angenommen Eve möchte erst nach dem Austausch messen; was könnte sie versuchen?

Allgemeinere Angriffe

1. Wir haben angenommen, dass Eve ein Bit misst und weiterleitet (oder äquivalent, ein neues Bit gemäß der Messung erzeugt und verschickt).
2. Da Alice und Bob sich erst danach über die Messbasen austauschen, konnte Eve nicht vermeiden, in der Hälfte der Fälle in der falschen Basis zu messen.
3. **Verständniskontrolle.** Angenommen Eve möchte erst nach dem Austausch messen; was könnte sie versuchen?
4. **Antwort.** Eve könnte versuchen, mit einem CNOT-Gatter ein zusätzliches Bit $|e\rangle = |0\rangle$ mit dem übertragenen Bit $|x\rangle$ zu verschränken, und dann später $|e\rangle$ zu messen.

Angriff durch Verschränkung

1. Falls Alice $|x\rangle = |1\rangle$ sendet, dann $\mathrm{CNOT}(|x\rangle\,|e\rangle) = |1\rangle\,|1\rangle$.
2. Eve belauscht den Austausch über die verwendeten Basen.
3. Hat Bob in der richtigen Basis $\{|0\rangle, |1\rangle\}$ gemessen, weiß Eve das nun, und kann $|e\rangle$ ebenfalls in $\{|0\rangle, |1\rangle\}$ messen. Sie erhält das korrekte Ergebnis 1.
4. Sendet Alice dagegen $|x\rangle = |-\rangle$, so gilt

$$\mathrm{CNOT}(|x\rangle\,|e\rangle) = \mathrm{CNOT}(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\,|0\rangle) = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle).$$

5. Nimm an, Bob errät die Basis $\{|+\rangle, |-\rangle\}$ und misst. Äquivalent wendet er H an und misst in Basis $\{|0\rangle, |1\rangle\}$. Wegen

$$H(|x\rangle)\,|e\rangle = \frac{1}{\sqrt{2}}((H\,|0\rangle)\,|0\rangle - (H\,|1\rangle)\,|1\rangle) = \frac{1}{2}(|00\rangle + |10\rangle - |01\rangle + |11\rangle)$$

   erhält er 0 oder 1 jeweils mit Wahrscheinlichkeit 50%.
6. Wegen $H(|-\rangle) = |1\rangle$ müsste er aber immer 1 erhalten.
7. Durch einen Bitvergleich kann Eve also enttarnt werden.

Angriff durch beliebige Transformationen

1. Statt CNOT könnte Eve andere unitäre U benutzen.
2. **Übungsaufgabe.** Zeigen Sie, dass ein beliebiges U, welches das erste Bit eines 2-Bit-Registers nicht verändert, nicht geeignet ist, um die Zustände $|0\rangle, |1\rangle, |+\rangle, |-\rangle$ zu unterscheiden.
3. **Übungsaufgabe.** Zeigen Sie, dass, falls U das erste Bit verändern darf, die Veränderung umso stärker ausfallen muss, je besser Eve $|0\rangle, |1\rangle, |+\rangle, |-\rangle$ unterscheiden können möchte.

(Die ausführlichen Aufgabenstellungen finden sich auf dem Übungsblatt.)

Das B92-Protokoll

1. Alice erzeugt Zufallsbits $a_1, \ldots, a_m$
2. Alice sendet $|x_1 \ldots x_m\rangle$ an Bob, wobei
$$|x_i\rangle = \begin{cases} |0\rangle & \text{falls } a_i, = 0, \\ |+\rangle & \text{falls } a_i = 1, \end{cases} \quad \text{for } 1 \leq i \leq m.$$
3. Bob erzeugt Zufallsbits $a'_1, \ldots, a'_m$.
4. Für $1 \leq i \leq m$, falls $a'_i = 0$, misst Bob $|x_i\rangle$ in Basis $\{|0\rangle, |1\rangle\}$,
sonst misst Bob $H(|x_i\rangle)$ in Basis $\{|0\rangle, |1\rangle\}$,
und speichert das jeweilige Ergebnis als b_i.
5. Bob schickt $b_1, \ldots, b_m$ an Alice.
6. Alice's Schlüssel ist $a_{i_1} \ldots a_{i_\ell}$ mit $\{i_1 < \ldots < i_\ell\} = \{i \colon b_i = 1\}$.
7. Bob's Schlüssel ist $(1 - a'_{i_1}) \ldots (1 - a'_{i_\ell})$.
8. **Verständniskontrolle.** Wie lang wird der Schlüssel?
9. **Verständniskontrolle.** Warum funktioniert das Protokoll?

Ganz andere Angriffsszenarien

1. **Man in the Middle.** Eve gibt sich Alice gegenüber als Bob aus und umgekehrt; sie sieht die unverschlüsselte Nachricht.
2. **Denial of Service.** Eve stört den Quantenkanal; Alice und Bob müssen auf einen klassisch angreifbaren Kanal ausweichen.
3. **Physikalische Angriffe.** Eve schickt zum Beispiel einen starken Lichtpuls an Alices und Bobs Sender und Empfänger. Einige der Photonen werden reflektiert und geben möglicherweise Auskunft über die eingestellten Polarisierungen (= Messbasen). Eve nutzt dafür eine Wellenlänge, die Alice und Bob nicht beobachten („large pulse attack“).

19 Schlussbemerkungen

Zitat

„Wenn man nicht zunächst über die Quantentheorie entsetzt ist, kann man sie doch unmöglich verstanden haben.“

— Niels Bohr

Frohe Weihnachten!