Das DC-Netz
Anonymes Senden und Empfangen

Inhaltsverzeichnis

1 Das DC-Netz

Das DC-Netz wurde 1984 von David Chaum erfunden [Chau_88]. Wie das MIX-Netz (vgl. Versuch "Das Mix-Netz") soll auch das DC-Netz Anonymität in Kommunikationsnetzen ermöglichen. Im Unterschied zum MIX-Netz, das in seiner Grundform nur die Kommunikationsbeziehung verbirgt, soll im DC-Netz Senden und Empfangen selbst anonym sein.

Unbeobachtbares Empfangen (und zwar in einem informationstheoretischen Sinn!) wird dabei durch **Verteilung der Nachrichten** erreicht.

Anonymes Senden

Wie kann man anonymes Senden erreichen?

Natürlich könnte man versuchen, die **Netzleitungen abzuschirmen**, um Abhören zu erschweren; dies wäre wohl zum einen teuer, zum anderen aber hilft es (wie Verbindungsverschlüsselung) nicht gegen Netzteilnehmer selbst.

Das physische Senden bedeutungsloser Leernachrichten (**dummy traffic**) kann das Senden von Botschaften auch vor anderen Netzteilnehmern, nicht jedoch vor dem Empfänger der Botschaft verbergen [Pfit_89, S. 90].

Da das physische Senden von Nachrichten auf dem Netz immer beobachtbar ist, darf nicht jede physisch gesendete Nachricht wirklich eine logische Botschaft beinhalten; es muß physische **Leernachrichten** geben, die nicht zum Senden von übergeordneten Inhalten verwendet werden. Im folgenden wird die Ausgabe eines Knotens ins Netz als (physische) **Nachricht**, die evtl. enthaltenen Inhalte als (logische) **Botschaft** bezeichnet.

Damit ein Angreifer nicht aus dem Senden einer Nachricht auf das Senden einer Botschaft schließen kann, sollten, wann immer ein Teilnehmer eine Botschaft senden will, alle Teilnehmer eine Nachricht senden; in Analogie zum Empfänger bei Verteilung ist damit der Sender anonym unter allen Teilnehmern, außer vor dem Empfänger der Botschaft.

Um auch das zu erreichen, darf der Empfänger die Botschaft, die er erhält, nicht in einer der Nachrichten der Teilnehmer erkennen können. Man hat also hier eine Art Umcodierungsproblem wie bei den MIXen.

Ein Beispiel

Petra will von einer (etwa Bildschirmtext-ähnlichen) Datenbank Informationen abfragen, etwa Krankheitsverläufe von einer medizinischen Datenbank, und legt Wert darauf, bei dieser Anfrage anonym zu bleiben.

Dazu muß sie die Anfrage anonym an die Datenbank senden und anonym die Antwort empfangen. Zum anonymen Empfangen wird Verteilung mit verdeckter impliziter Adressierung verwendet. Petra teilt dazu (anonym) der Datenbank ein Pseudonym (Zufallszahl als Adresse) mit; die Datenbank verteilt ihre Antwort mit diesem Pseudonym als Adresskennzeichnung.

Alle Empfänger suchen in allen verteilten Nachrichten nach ihren Pseudonymen und werfen fremde Nachrichten weg. (Es kann auch Ende-zu-Ende verschlüsselt werden. Auch zum anonymen Bezahlen von anonymen Dienstleistungen gibt es geeignete Verfahren [PWP_90].)

Petra kann also die Antwort anonym empfangen. Zum anonymen Senden (anonym auch gegenüber der Datenbank als Empfänger und dem Netzbetreiber gemeinsam!) nützen ihr weder abgeschirmte Leitungen noch Verbindungs-Verschlüsselung noch das Senden von bedeutungslosen Nachrichten.

Petras Senden kann anonym sein immer nur unter all denen, die als Sender ebenfalls in Frage kommen, die also ebenfalls physisch gesendet haben (bei den MIXen ist das jeweils der entsprechende Schub Nachrichten). Unter allen Teilnehmern anonymes Senden von Botschaften ist daher nur möglich, wenn auch alle Teilnehmer Nachrichten senden.

Der Empfänger (die Datenbank) darf die Anfrage natürlich nicht in diesen Ausgaben der Teilnehmer erkennen können, sonst könnte sie durch Abhören der Leitungen und Vergleich

der Nachrichten mit der Anfrage den Sender ausfindig machen (Umcodierungsproblem). David Chaum hat für dieses Umcodierungsproblem eine einfache und wirkungsvolle Lösung gefunden: das DC-Netz.

Das DC-Netz

Das DC-Netz geht von Botschaften stets gleicher und fester Länge aus.

Die Teilnehmer verwenden **Schlüssel** (Zufallszahlen, die den Botschaften etwa byteweise überlagert werden; daher auch der Name **Überlagerndes Senden**). Der Sender addiert dabei etwa byteweise Schlüssel auf seine Botschaft und verschickt als Nachricht das Ergebnis (**lokale Summe**); jeder, der diese Schlüssel nicht kennt, kann die Botschaft in der Nachricht nicht erkennen (vgl. one-time pad). Hat ein Teilnehmer nichts zu senden, überlagert er die Schlüssel einer **Leerbotschaft**.

> Zum Überlagern einer k Bytes langen Botschaft sind natürlich k Schlüsselbytes nötig.

Die Botschaft muß, nachdem der Teilnehmer sein Ergebnis und die übrigen Teilnehmer ihre Nachrichten ebenfalls verschickt haben, wieder „entschlüsselt“ werden. Dazu teilt der Sender je einen der addierten Schlüssel einem anderen Teilnehmer mit, worauf jener diesen Schlüssel von seiner Botschaft bzw. Leerbotschaft subtrahiert.

Addiert man nun alle Nachrichten, die in dieser „Runde“ verschickt wurden, wurde jeder Schlüssel genau **einmal addiert und einmal subtrahiert**, sie heben sich daher alle weg. Nach jeder Runde tauchen also die gesendeten Botschaften „auf einen Schlag“ bei der globalen Addition wieder auf (**globale Summe**).

In jeder dieser Runden führt damit jeder Knoten das folgende Protokoll durch:

Protokoll DC-Runde:
Eingabe: Botschaft; {default: Leerbotschaft}
lokale_Summe := Überlagerung (Schlüssel, Botschaft); {addiert bzw. subtrahiert}
verteile (lokale_Summe); {an alle Teilnehmer}
empfange (Ausgaben); {lokale Summen der anderen Teilnehmer}
globale_Summe := Überlagerung (Ausgaben, lokale_Summe); {addiert}
Ausgabe: globale_Summe.

Abbildung 1: Das DC-Rundenprotokoll

Die Botschaften und Schlüssel müssen dazu als Elemente einer abelschen (kommutativen) Gruppe interpretiert werden können; die Leernachricht ist dabei das neutrale Element. Die Schlüssel sind genauso lang wie die Botschaften.

Natürlich müssen in jeder DC-Runde vollständig **neue Schlüssel** verwendet werden, da die lokale Summe von einem Knoten, der keine Botschaft (d.h. die Leerbotschaft) sendet, sonst in jeder Runde gleich wäre. Für jedes Byte einer Botschaft, die *ein* Teilnehmer im Netz anonym

senden will, benötigt *jeder* Teilnehmer im Netz also (mindestens) ein Schlüsselbyte, das er zuvor mit einem anderen Teilnehmer ausgetauscht haben muß.

Um nicht jedesmal die benötigten neuen Schlüssel austauschen zu müssen, kann man auch **Pseudozufallsgeneratoren** (**PZG**, etwa [BlBS_86, VaVa_85]) verwenden: Die Teilnehmer tauschen dann nur einmal zu Beginn Startwerte der PZGen aus. Um Gefährdung der Anonymität durch gebrochene Pseudozufallszahlenfolgen zu verhindern, müssen die PZGen kryptographisch sicher sein; die Anonymität im DC-Netz ist dann ebenfalls nur noch kryptographisch sicher.

Im DC-Netz erfährt automatisch jeder Teilnehmer die globalen Summen. Daher würde man normalerweise die üblichen Verfahren (Ende-zu-Ende-Schutz, implizite Adressierung) einsetzen. Der Einfachheit und Übersichtlichkeit halber werden wir in diesem Praktikumsversuch ohne diese Verfahren arbeiten.

Dieses DC-Netz-Schema soll im folgenden genauer untersucht und verallgemeinert werden. Dabei wird insbesondere die Sicherheit angesprochen und Algorithmen zur anonymen Auflösung von Kollisionen erläutert.

Allgemein zum DC-Netz siehe [Pfit_89, S. 92ff und 119ff], zur Diensterbringung [WaPf_89] [Pfit_89, S. 267ff][PfWa1_91].

2 Grau ist alle Theorie...

... man glaubt es erst, wenn man ein Beispiel gesehen hat. Im folgenden werden wir ein DC-Netz mit vier Knoten und einer Zentrale betrachten:

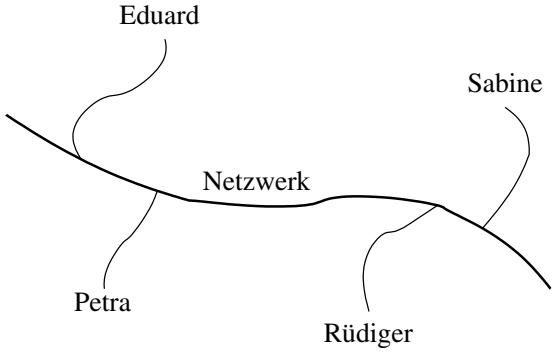

Abbildung 2: Netztopologie des betrachteten Beispiels

In jeder DC-Netz-Runde sendet jeder Knoten seine Nachricht (=Überlagerung von Botschaft bzw. Leerbotschaft und Schlüsseln) an alle anderen Knoten.

Da die Knoten paarweise dieselben Schlüssel verwenden müssen, müssen die Knoten diese Schlüssel vorher ausgetauscht haben. In unserem Beispiel sollen Eduard mit Sabine, Sabine mit Rüdiger und Rüdiger mit Petra Schlüssel ausgetauscht haben, was durch die Linien im **Schlüsselgraphen** (Bild 3) angedeutet wird. An den Linien ist jeweils vermerkt, wer den Schlüssel positiv und wer ihn negativ verwendet (z.B. addiert oder subtrahiert).

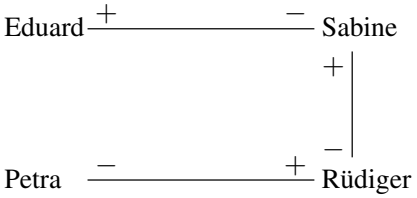

Abbildung 3: Schlüsselgraph des betrachteten DC-Netzes

Achtung: Der Schlüsselgraph gibt *nicht* an, wer mit wem (anonym) kommunizieren kann; da alle Nachrichten verteilt werden, kann natürlich jeder an jeden senden. Der Schlüsselgraph gibt nur an, wer mit wem Schlüssel ausgetauscht hat. Was das mit Anonymität zu tun hat, werden wir in den folgenden Aufgaben sehen.

Damit Rechnungen von Hand einfach durchführbar sind und alle Zeichen auf dem Bildschirm eindeutige Darstellungen besitzen, wird folgende Zeichencodierung verwendet:

_	A	B	C	D	E	F	G	H	I	J	K	L	M	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

Gerechnet wird hier also im 30er-Zahlensystem (zeichenweise modulo 30). Normalerweise könnte man auch die normale ASCII-Codierung verwenden oder sich die Nachrichten als Zahlen vorstellen. In beiden Fällen ist aber keine anschauliche Bildschirmein- und -ausgabe möglich. Für die Berechnungen sind vorgefertigte Blätter zum Ausfüllen in Anhang A beigefügt.

Aufgabe 1: Das Funktionsprinzip

Rüdiger hat mit Petra und Sabine folgende Schlüssel ausgetauscht (vgl. Bild 3; das Leerzeichen ist im folgenden oft als '_' dargestellt):

Schlüssel Rüdiger-Petra: `'V ? ! Y C P G C P S ! U N Y . P . N S V'`
Schlüssel Rüdiger-Sabine: `'_ M W X Q S C I X L H H D . Z E _ H P R'`

Rüdiger will im Moment keine Botschaft schicken; er verwendet in dieser Runde daher die Leerbotschaft. Nach dem DC-Runden-Protokoll aus Bild 1 muß er dieser Botschaft nun die beiden Schlüssel überlagern (addieren bzw. subtrahieren wie im Schlüsselgraphen, Bild 3, vorgeschrieben). Das Ergebnis, seine lokale Summe, verteilt er dann an die anderen Teilnehmer.

Aufgabe 1 a): Berechnen Sie von Hand (in der angegebenen Zeichencodierung zeichenweise modulo 30) die Nachricht (=Überlagerung), die Rüdiger zu verteilen hat. Vorgefertigte Blätter zum Ausfüllen und eine Additionstabelle sind beigefügt.

Umgekehrt erhält er in dieser DC-Runde von den anderen Teilnehmern die folgenden Nachrichten (deren lokale Summen):

Nachricht von Petra:	`'H A B E . N W . N K B I P E C N C P K H'`
Nachricht von Eduard:	`'O L Q G I A H K _ O I C L V I R I B C K'`
Nachricht von Sabine:	`'S ! F T P S P K X . ? E V E Q Q U F M G'`

Aufgabe 1 b): Überlagern Sie diese Nachrichten Rüdigers eigener lokalen Summe, wie es das DC-Runden-Protokoll aus Bild 1 angibt! Welche Botschaft wurde in dieser DC-Runde verschickt? (Hilfe: Die anonyme Botschaft hat die Quersumme 77.) ♢

Für die nächsten Aufgaben wird das DC-Netz-Programm benötigt. Starten Sie nun den DC-Client (z.B. `client.bat`).

3 Mit Sicherheit anonym?

Nun stellt sich natürlich die Frage, ob dieses Verfahren wirklich unbeobachtbares Senden und Empfangen garantiert, und insbesondere, wie die Sicherheit des Verfahrens vom **Schlüsselgraphen** abhängt.

Natürlich könnten **alle** Teilnehmer untereinander Schlüssel austauschen (vollvermaschter Schlüsselgraph). In diesem Fall benötigen die DC-Netz-Teilnehmer in jeder Runde $\frac{1}{2} \cdot n \cdot (n-1)$ Schlüssel (d.h. $\frac{1}{2} \cdot n \cdot (n-1)$ Schlüsselbits pro übertragenem Bit).

Die nächsten Aufgaben finden am Rechner statt. Für die Aufgaben 2 bis 4 wird die lokale DC-Netz-Demo mit den vier Teilnehmern Eduard, Sabine, Rüdiger und Petra benötigt. Wenn der Client gestartet wird, ist automatisch diese Demo eingestellt. Es werden für die einzelnen Aufgaben jeweils die lokalen Summen der Teilnehmer sowie die globale Summe angezeigt. Mit der Taste „`Nächster Schritt`“ wird jeweils auf die DC-Netz-Runde für die nächste Aufgabe umgeschalten.

Auch in den folgenden Aufgaben wird das DC-Netz mit dem Schlüsselgraphen aus Abschnitt 2 verwendet. Die Schlüssel wurden mit PZGen erneuert, die Schlüssel aus Aufgabe 1 haben sich also geändert. Zur Erleichterung des Rechnens mit dem Praktikumsalphabet sind Additions- und Invertierungstabellen im Anhang beigefügt; es kann auch ein „DC-Rechner“ verwendet werden, der im Clientprogramm unter `Datei⇒DC-Rechner` zu finden ist. Er erlaubt Addition und Subtraktion ganzer Nachrichten und Schlüssel.

Aufgabe 2: Isolierter Teilnehmer

Das gestartete DC-Netz arbeitet mit den Teilnehmern und dem Schlüsselgraphen aus Aufgabe 1, aber natürlich mit (von PZGen erzeugten) *neuen* Schlüsseln. **Felix**, einem Angreifer, der mit Eduard und Rüdiger befreundet ist, gelingt es, die Schlüssel in Erfahrung zu bringen, die beide mit Sabine ausgetauscht haben:

Schlüssel Rüdiger-Sabine:	`'G _ S H _ L V B M L I _ T O D . Y M Z V'`
Schlüssel Eduard-Sabine:	`'H . A ? C . ? L T W J _ Z H H Q J Y A A'`

In der nächsten DC-Runde belauscht er die Netzleitungen (Monitor) und erfährt so die lokale Summe von Sabine:

Nachricht von Sabine: `'V H F I F D M T A X Q R X H H _ N R Y U'`

(Diese Nachricht erscheint auch auf dem Bildschirm als lokale Summe von Sabine. Zusätzlich sind die Bildschirmausgaben für die Aufgaben 2 bis 4 auch im Anhang C zu finden.)

Aus welchen Schlüsseln und Botschaften setzt sich Sabines lokale Summe zusammen?

Sabines lokale Summe = Sabines Botschaft

\+ Schlüssel Sabine- _____

− Schlüssel Sabine- _____

Aufgabe 2: Rechnen Sie „rückwärts“ aus der abgehörten lokalen Summe die Schlüssel wieder ab! Hat Sabine eine Botschaft geschickt, und wenn ja, welche? ♢

Aufgabe 3: Geteilter Schlüsselgraph

In der nächsten DC-Runde werden wieder neue Schlüssel der PZGen verwendet, so daß Felix Sabine nicht mehr isolieren kann. In dieser Runde antwortet ein Teilnehmer mit „Ich!“, wie am Monitor ersichtlich ist. Felix wüßte nun gerne, wer auf Sabines Botschaft aus Aufgabe 2 geantwortet hat (natürlich kommen als Antwortende, als „Herr Alt“, nur Eduard und Rüdiger in Frage).

Wessen Schlüssel müßte Felix kennen, um diese Frage entscheiden zu können? Was könnte er etwa mit dem Schlüssel Rüdiger-Sabine anfangen:

Schlüssel Rüdiger-Sabine: `'B H V F K H C _ ! P N P Q ? K Q R O Q P'`

(Wie im Schlüsselgraphen ersichtlich, verwendet Rüdiger den Schlüssel negativ, Sabine positiv.) Tragen Sie im folgenden Bild den Schlüsselgraphen ein, der entsteht, wenn man die Schlüssel wegläßt, die Felix kennt. Was fällt auf?

Eduard Sabine

Petra Rüdiger

Abbildung 4: Schlüsselgraph des geteilten DC-Netzes

Aufgabe 3 a): Berechnen Sie die Teilsummen „Eduard+Sabine“ und „Petra+Rüdiger“!

Aus welchen Botschaften und Schlüsseln setzen sich die errechneten Teilsummen jeweils zusammen?

Teilsumme Eduard+Sabine	=	Sabines Botschaft
		+ Eduards Botschaft
		+ Schlüssel Eduard- _____
		− Schlüssel Sabine- _____
		+ Schlüssel Sabine- _____
Teilsumme Petra+Rüdiger	=	Petras Botschaft
		+ Rüdigers Botschaft
		− Schlüssel Petra- _____
		+ Schlüssel Rüdiger- _____
		− Schlüssel Rüdiger- _____

Welche Schlüssel fallen bei der Bildung der Teilsummen weg, welche bleiben übrig?

Aufgabe 3 b): Trennen Sie das Gesamtnetz in zwei Teil-DC-Netze auf! Rechnen Sie dazu die übriggebliebenen Schlüssel jeweils von den Teilsummen ab! (Anmerkung: Da die resultierende Botschaft schon vorher bekannt ist und nur aus genau einem der Teilnetze kommt, genügt hier die Berechnung etwa des ersten Zeichens.)

Durch diese Aufteilung des DC-Netzes kann Felix erfahren, aus welchem Teilnetz die Nachricht kam (und in diesem Fall sogar von wem)! ◊

Resultat:
Ein DC-Netz-Teilnehmer ist anonym (höchstens) in der Gruppe von Teilnehmern, die einen zusammenhängenden Teilgraphen des Schlüsselgraphen bilden. Schlüssel, die der Angreifer kennt, sind für diesen natürlich kein Hindernis, er braucht sie in seiner Sicht des Schlüsselgraphen nicht berücksichtigen.
Auch äußere Zusatzinformationen, hier Nachrichteninhalte, können die Anonymität weiter einschränken. Wie man sich leicht überlegt, gilt dies etwa auch für den Sendezeitpunkt.

Aufgabe 4: Der Sicherheitsbeweis

Wie am Monitor ersichtlich, wird in der dritten Runde die Botschaft „WAS SOLL DAS?“ geschickt. Sabine möchte nun gerne wissen, von wem die Botschaft stammt. Sie kennt natürlich ihre Schlüssel:

Schlüssel Rüdiger-Sabine: `'C R I _ N Z N _ G T M Z R G P A M C L P'`
Schlüssel Eduard-Sabine: `'D Y A M G G . F F K B H P X F Y X Y E G'`

Sie hat auf dem Netz die lokalen Summen abgehört und kann nun ihre Schlüssel von den lokalen Summen abrechnen. Tragen Sie im folgenden Bild den Schlüsselgraphen ein, der entsteht, wenn man die Schlüssel wegläßt, die Sabine kennt. Was fällt auf?

Eduard Sabine

Petra Rüdiger

Abbildung 5: Schlüsselgraph des DC-Netzes aus Sabines Sicht

Aufgabe 4 a): Berechnen Sie die Botschaft, die Eduard geschickt hat!

Man stellt fest: Eduard ist nicht anonym vor Sabine, da er keinen Schlüssel ausgetauscht hat, den Sabine nicht kennt. Die Botschaft kann also nur von Petra oder Rüdiger gekommen sein. Sabine trennt Petra und Rüdiger als Teilnetz ab, indem sie wie in Aufgabe 3 von der Teilsumme „Petra + Rüdiger“ den ihr bekannten Schlüssel abrechnet.

Aufgabe 4 b): Berechnen Sie diese globale Summe des Teilnetzes „Petra+Rüdiger“!

Aus welchen Botschaften und Schlüsseln setzt sich die errechnete Summe zusammen?

Teilnetz „Petra+Rüdiger“ = Botschaft „WAS SOLL DAS?“ (von Petra oder Rüdiger)
+ Leerbotschaft (vom jeweils anderen)

− Schlüssel Petra- _____

+ Schlüssel Rüdiger- _____

Aufgabe 4 c): Angenommen, Petra hätte die Botschaft geschickt. Berechnen Sie für diesen Fall den Schlüssel Petra-Rüdiger!

Aufgabe 4 d): Und wenn Rüdiger die Botschaft geschickt hätte? Berechnen Sie auch für diesen Fall den Schlüssel Petra-Rüdiger! ◊

Resultat:
Wer auch die Botschaft geschickt hat — in beiden Fällen existiert ein Schlüssel, den die beiden ausgetauscht haben könnten, so daß sich genau diese Nachrichten auf dem Netz ergeben: Diese Botschaft wurde anonym gesendet (anonym vor allen, die diesen Schlüssel nicht kennen)!
Dies gilt für je zwei Teilnehmer, die einen geheimen Schlüssel ausgetauscht haben. Würde Eduard diese Berechnungen anstellen, wüßte er nicht, ob Sabine, Rüdiger oder Petra die Botschaft geschickt hat, da er noch den Schlüssel Sabine-Rüdiger nicht kennt. Dies „ist" der Beweis zur Senderanonymität (vollst. Induktion über die Anzahl der Teilnehmer im Schlüsselgraphen [Pfit_89, S. 96]).
Ein Sender einer Botschaft ist anonym unter all denjenigen DC-Netz-Teilnehmern, die zusammen mit ihm einen zusammenhängenden Teilgraphen im Schlüsselgraphen bilden. Die Schlüssel, die der Angreifer kennt, sind dabei im Schlüsselgraphen wegzulassen.

Der Schlüsselgraph beschreibt also nur den Grad der Anonymität, *nicht* die möglichen Kommunikationsbeziehungen; jeder DC-Netz-Teilnehmer kann an jeden senden (Verteilung!), anonym ist er **unabhängig vom Empfänger** nur innerhalb der Teilnehmer, die mit ihm im Schlüsselgraphen zusammenhängen.

Insbesondere kann in unserem Beispiel Eduard nicht entscheiden, ob Sabine, Rüdiger oder Petra senden: Obwohl Sabine und Petra direkt keine Schlüssel untereinander ausgetauscht haben, sind sie anonym vor Eduard.

4 Kollisionen und Auflösungsmechanismen

Hinweis zum Programm:
Nun können die Knoten (DC-Clients) beim DC-Netz-Server (der die Bildung der globalen Summe übernimmt) angemeldet werden. Dafür muß zunächst die Netzwerkkonfiguration angepaßt werden. (`Optionen`⇒`Netzwerk-Konfiguration`) Es muß die IP-Adresse des Servers und dessen Empfangsport eingetragen werden. Dem Client selbst sollte am besten der eigene Name gegeben werden. (Die Standardwerte für Portnummer und Hostname sollten dort funktionieren).

Nun können die Clients am DC-Netz angemeldet werden (Taste „`DC-Netz`"). Die Knoten erhalten von der Zentrale (Server) Informationen über die bereits angemeldeten Teilnehmer und tauschen mit diesen Schlüssel aus. (Dies kann kurze Zeit dauern, da der kryptographisch starke Zufallsgenerator etwas Zeit benötigt, um genügend zufällige Systemereignisse zu finden und sich damit selbst zu initialisieren.) Die Botschaften der Teilnehmer werden dann mit diesen Schlüsseln überlagert. („`inv`" in der Teilnehmeranzeige gibt an, daß dieser Schlüssel invertiert überlagert wird.)

Im normalen Modus sendet jeder Client jeweils nach 20 Sekunden eine Leernachricht, falls keine Botschaft eingegeben wurde (Taste „`neue Nachricht eingeben`"). Im Einzelschrittmodus muß jeder Teilnehmer in jeder Runde eine Botschaft explizit angeben. Dieser Modus eignet

sich besonders, um Kollisionen von Nachrichten zu beobachten. Zwischen diesen Modi kann am Server umgeschaltet werden, und sie gelten jeweils ab der nächsten DC-Netz-Runde.

Aufgabe 5: Paarweises Überlagerndes Empfangen

Was passiert, wenn in einer DC-Runde zwei Knoten eine echte Botschaft (und keine Leerbotschaft) schicken?

Aufgabe 5: Wählen Sie einen Partnerknoten, erzeugen Sie eine Kollision von genau zwei Botschaften und betrachten Sie das Kollisionsergebnis. Aus welchen Bestandteilen besteht das Ergebnis? Wer kennt welche Bestandteile? Wer kann die ursprünglichen Botschaften berechnen?

Kein Außenstehender kann die einzelnen Botschaften bestimmen! Dieses Verfahren heißt **Paarweises Überlagerndes Empfangen** [Pfit_89, S. 98]. ◊

Resultat:
Paarweises Überlagerndes Empfangen funktioniert, wenn zwei Teilnehmer sich gegenseitig Botschaften zuschicken wollen, die genaue Runde vorher ausgemacht haben und kein anderer Teilnehmer dazwischenfunkt.

Aufgabe 6: Kollisionsauflösung durch wiederholtes Senden

Was würde geschehen, wenn genau eine der beiden kollidierten Botschaften in der nächsten Runde nochmal gesendet würde und nun nicht mehr kollidiert?

Aufgabe 6 a): Probieren Sie das mit dem Partner aus Aufgabe 5 aus! Berechnen Sie die beiden Botschaften!

Dies liegt daran, daß diese „logischen“ (additiven) Überlagerungs-Kollisionen im DC-Netz (im Gegensatz zu üblichen Übertragungs-Kollisionen) durchaus noch ihren maximalen Informationsgehalt besitzen. Diese Eigenschaft des DC-Netzes kann man auch bei mehreren Kollisionen ausnutzen.

Aufgabe 6 b): Probieren Sie das für 4 kollidierte Botschaften aus: (Sie sollten sich die Botschaften für die Wiederholungen merken!): Wieviele DC-Netz-Runden wurden für diese vier Botschaften benötigt? ◊

In diesem Verfahren mußte jeder wissen, wann wer senden wird, damit sie sich nicht in die Quere kommen; das ist natürlich ganz und gar nicht anonym. Man könnte dann gleich von vornherein festlegen, wann wer senden darf (Token), und auf die Anonymität verzichten.

Es gibt aber auch Verfahren, die **anonym** festlegen, wann wer sendet. Zwei davon werden wir nun kennenlernen.

Aufgabe 7: Reservierungsverfahren

Eine Möglichkeit ist, daß sich jeder DC-Netz-Teilnehmer anonym Runden reservieren lassen kann [Pfit_89, S. 137]. In unserem Nachrichtenformat ginge das etwa so:

Es gibt festgelegte Reservierungsrunden im DC-Netz. In diesen Runden wird jedes Zeichen der Nachrichten für sich betrachtet; es steht für eine der folgenden DC-Netz-Runden. In unserem Nachrichtenformat mit 20 Zeichen werden die folgenden 20 DC-Runden reserviert. Jeder DC-Netz-Teilnehmer, der etwas senden will, sucht sich eine dieser 20 Runden aus und sendet als Zeichen im entsprechenden Feld der Reservierungsrunde eine '1' (da es in unserem Format keine Zahlen gibt, tut es dort das 'A'), in den übrigen eine '0' (bzw '_').

Nach der Reservierungsrunde begutachtet jeder DC-Netz-Teilnehmer das Überlagerungsergebnis. Da unser DC-Netz zeichenweise addiert hat, ist für jede der folgenden 20 DC-Runden ablesbar, wieviele DC-Netz-Teilnehmer sich eine Runde zu reservieren versuchten:

- Eine '0' (bzw. '_') bedeutet, daß keine Reservierung erfolgt ist.
- Eine '1' (bzw. 'A') bedeutet, daß genau ein DC-Netz-Teilnehmer in dieser Runde senden will.
- Größere Werte bedeuten, daß mehrere DC-Netz-Teilnehmer eine Reservierung versuchten und deshalb keiner senden kann.

Natürlich können alle DC-Runden, für die keine oder keine eindeutige Reservierung vorliegt, weggelassen werden. DC-Netz-Teilnehmer, deren Reservierung kollidierte, müssen es eben in der nächsten Reservierungsrunde nochmal probieren.

Aufgabe 7: Was passiert, wenn es mehr DC-Netz-Teilnehmer gibt als Zeichen im verwendeten Reservierungsalphabet? Was passiert, wenn ein DC-Netz-Teilnehmer in **jeder** Reservierungsrunde in **jedem** Reservierungsfeld eine '1' (oder eine beliebige andere Zahl >0) sendet? ♢

Aufgabe 8: Globales Überlagerndes Empfangen

Sehen Sie sich das Beispiel aus Aufgabe 6 noch einmal an. Damit das Senden anonym bleibt, müßte die Reihenfolge des wiederholten Sendens natürlich wiederum anonym festgelegt werden. Man erhält dann ein Verfahren, das **Globales Überlagerndes Empfangen** genannt wird. Zu lösen sind dazu die beiden folgenden Probleme:

erstes Problem: Woher wissen die Teilnehmer, wieviele Botschaften überlagert sind?

zweites Problem: Wann muß wer seine Botschaft nocheinmal schicken?

Das erste Problem wird dadurch gelöst, daß ein Zähler parallel zur Nachricht mitgeführt und natürlich auch durch Schlüssel überlagert wird (wie im eigentlichen DC-Netz). Jeder Teilnehmer, der eine Botschaft sendet, sendet als Zählerbotschaft (ebenso anonym) eine '1', für die Leerbotschaft eine '0'. Diese Zählernachrichten werden getrennt von den eigentlichen Nachrichten, aber genauso wie diese global überlagert. Wählt man dabei die für die Zähler

verwendete abelsche Gruppe als zyklische Gruppe (z.B. Addition modulo einer festen Zahl) und genügend groß (wie groß nämlich?), so erscheint im Zählerfeld der globalen Summe genau die Anzahl der kollidierten Botschaften!

Trick:
Werden auch die Botschaften in einer genügend großen zyklischen Gruppe addiert und interpretiert man die kollidierten Botschaften als (große) Zahl und teilt diese durch die Anzahl der Kollisionen, so erhält man die **„durchschnittliche Botschaft“**, die in dieser Runde gesendet wurde!

Dadurch kann jeder für sich seine eigene, gesendete Botschaft mit dieser Durchschnittsbotschaft vergleichen und das Resultat zur Entscheidung benutzen, wann er die Botschaft nocheinmal senden muß: Ist seine Botschaft kleiner als die durchschnittliche oder gleich, sendet er sie gleich nochmal. Die Summe aller größeren läßt sich dann aus den Ergebnissen der ersten beiden Runden berechnen!

Sollten die kleineren und/oder größeren Botschaften wieder Kollisionen sein, kann man sie (rekursiv) genauso auflösen wie die ursprüngliche.

Das zweite Problem löst man also mit diesem Trick rekursiv durch Mittelwertbildung.

Aufgabe 8.1 Entwerfen Sie einen Algorithmus, der das leistet! *Hinweis:* Zum besseren Verständnis können sie das folgende Beispiel auf Papier rechnen (Zeichnen sie einen Baum!): A sendet 15, B 10, C 20, D und E senden beide 25. Was ist bei der Abarbeitung des Algorithmus wichtig bezüglich der Reihenfolge?

Aufgabe 8.2 Implementieren Sie diesen Algorithmus in der Methode
`resolveCollisions(DCNet dc, Message b)`
der Klasse `CollisionResolution` so, dass er die im Programmrahmen vorgegebenen Testfälle erfüllt. (Es geht dabei *nur* um die Kollisionsauflösung. Schlüssel werden dabei gänzlich außer acht gelassen.) Das DC-Netz an sich und die anderen Teilnehmer werden dabei von einem Stub simuliert, der im Programmrahmen `CollisionResolution` als Objekt `dc` vorliegt. Mit der Methode `dc.nextRound(Message b)` wird eine Botschaft ins Netz gesendet. Als Returnwert erhält man die Antwort des Netzes.

Was passiert, wenn zwei Teilnehmer innerhalb einer Kollision beide die gleiche Botschaft geschickt haben? ◊

Wichtige Hinweise: In DC-Netzen müssen alle Teilnehmer exakt das gleiche tun. Damit die Botschaftenreihenfolge im Kollisionsauflösungsalgorithmus eindeutig ist, müssen folgende Festlegungen beachtet werden:

- Nachdem eine erhaltene Botschaft b Kollisionen enthielt, wird die eigene Botschaft genau dann erneut gesendet, wenn sie in der Kollision in b enthalten ist[1] und *kleiner oder gleich* der Durchschnittsbotschaft ist. Anderenfalls wird eine Leernachricht gesendet.
- An einer Verzweigung im „Kollisionsauflösungsbaum“ wird immer zuerst der Zweig der empfangenen Botschaft verfolgt.

[1] „in der Kollision enthalten“ bedeutet hierbei, daß in der Runde, deren Resultat die Botschaft b ist, die eigene Botschaft (d.h. nicht die Leerbotschaft) gesendet wurde.

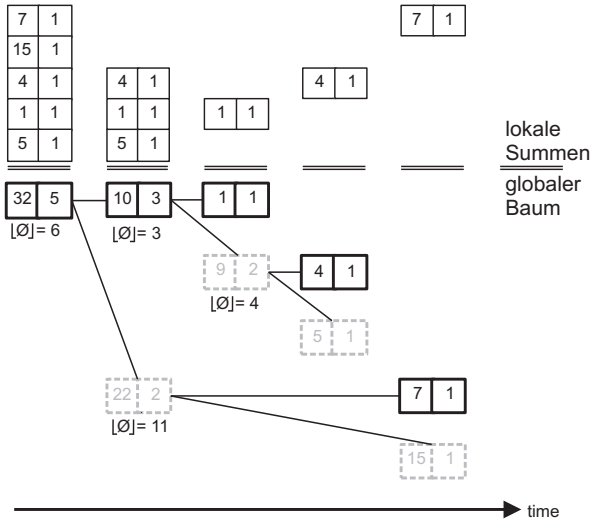

Abbildung 6: Abarbeitungsbaum bei der Kollisionsauflösung

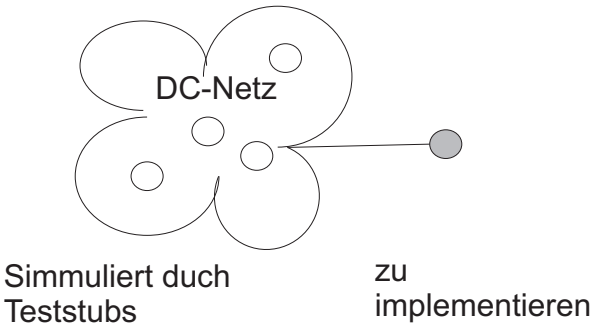

Abbildung 7: Prinzipplan

Abbildung 6 soll nochmals verdeutlichen wie der Algorithmus funktioniert. Dabei sind oben die Aktivitäten der Teilnehmer dargestellt und im unteren Bereich der globale (von jedem einzelnen Teilnehmer zu berechnende) Baum.

Die Aufgabe besteht nur darin die Funktionalität *eines einzelnen Teilnehmers* zu implementieren (dargestellt in Abbildung 7). Die Netzfunktionalität und die der anderen Teilnehmer wird durch die Unittests simuliert.

Wie oben schon beschrieben bietet sich für die Kollisionsauflösung ein rekursiver Algorithmus an. Um die Idee rekursiver Funktionen nochmal ins Gedächtnis zu rufen hier der Pseudocode einer Java-Klasse, die eine Liste implementiert:

```
public class List {

  public int kopf;
  public List rest;
```

```
  public List(int n, List r){
    kopf=n;
    rest=r;
  };

  public int sum(List l){
    if (l==null) return 0;
    return l.kopf + sum(l.rest);
  }
}
```

Mit der Klasse kann man Listen konstruieren, die Integerzahlen darstellen. Diese kann man mit der Methode sum(int,List) aufsummieren.

Aufgabe 9: Durchsatz mit Globalem Überlagerndem Empfangen

Am Server wird jetzt auf Kollisionsauflösung mit Mittelwertbildung umgeschaltet. Spielen Sie ein bißchen mit dem DC-Netz mit Globalem Überlagerndem Empfangen!

Wieviele DC-Netz-Runden werden jeweils benötigt, um die Kollisionen aufzulösen?

Kollidierte Botschaften	benötigte Runden

Wieviele Runden werden allgemein zur Übertragung von k kollidierten Botschaften benötigt?

◊

Resultat:
Globales Überlagerndes Empfangen ist ein sehr effizientes Verfahren, um Kollisionen aufzulösen. Es nutzt den Informationsgehalt der logischen Kollisionen auf dem DC-Netz aus [Pfit_89, S. 125ff].

5 Diensterbringung

Aufgabe 10: Diensterbringung

Was passiert, wenn ein Knoten Schlüssel verwechselt oder sich nicht an das Protokoll hält?

◊

Resultat:
Die Diensterbringung ist der Schwachpunkt des DC-Netzes. Zur Lösung des Problems sind viele Verfahren aus der Fehlertoleranz anwendbar [Pfit_89, S. 254], zur Verhinderung von Angriffen auf die Diensterbringung gibt es etwa spezielle Fallen- oder Schlüsselerzeugungsprotokolle. Eine ausführliche Behandlung dieses Themas findet man in [WaPf_89, PfWa1_91].

6 Bemerkungen zur Anwendung

Natürlich würde man zum Verbergen der Inhalte der Botschaften diese vor der Überlagerung mit Schlüsseln Ende-zu-Ende verschlüsseln und etwa verdeckte implizite Adressen verwenden. Für den Austausch der Schlüsselstartwerte für die Pseudozufallsgeneratoren können die üblichen Schlüsselaustauschverfahren etwa mit Schlüsselzentralen angewendet werden.

Zum Schluß noch ein echtes Anwendungsbeispiel:

Aufgabe 11: Drinking Cryptographers

(frei nach David Chaum; dort sind es „Dining Cryptographers“ in einem 3-Sterne-Restaurant.)

Angenommen, Sie sitzen mit zwei weiteren Kryptographen (beispielsweise nach dem Praktikum) in einer Studentenkneipe. Als es an's Bezahlen geht, erklärt der Wirt, es sei bereits alles bezahlt. Der Zahlende habe aber ausdrücklich gebeten, nicht genannt zu werden. Es ist aus dem Wirt nur soviel herauszubekommen, daß entweder der Chef (beispielsweise der Praktikumsleiter) oder einer der drei Kryptographen bezahlt hat.

Sie wollen nun herausfinden, ob der Chef bezahlt hat.

Ein zahlender Kryptograph würde natürlich nicht zugeben, daß er es war; eine „öffentliche“ Befragung innerhalb der drei Kryptographen nützt daher nichts.

Er müßte die Möglichkeit haben, sein Zahlen gewissermaßen anonym...

Wie könnte man das bewerkstelligen?

Als Hilfe: Es geht um ein einziges Bit an Information: „habe bezahlt“ bzw. „habe nicht bezahlt“. Einzelne Schlüsselbits sind durch Münzwürfe sehr leicht zu erzeugen... ◊

Ich glaube aber, in gewisser Hinsicht ist dieses Beispiel doch sehr unrealistisch!

Literatur

[BlBS_86] L. Blum, M. Blum, M. Shub: A Simple Unpredictable Pseudo-Random Number Generator. SIAM J. Comput., Band 15(2) (1986), 364–383.

[Chau_88] David Chaum: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. Journal of Cryptology, Band 1(1) (1988), 65–75.

[Pfit_89] Andreas Pfitzmann: Diensteintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz; Universität Karlsruhe, Fakultät für Informatik, Dissertation, Feb. 1989. IFB 234, Springer-Verlag, Heidelberg (1990).

[PfWa1_91] Birgit Pfitzmann, Michael Waidner: Unbedingte Unbeobachtbarkeit mit kryptographischer Robustheit. In: Proc. Verläßliche Informationssysteme (VIS'91) März 1991 Darmstadt, Band 271 von *Informatik-Fachberichte*, Springer-Verlag, Heidelberg (1991), 302–320.

[PWP_90] Birgit Pfitzmann, Michael Waidner, Andreas Pfitzmann: Rechtssicherheit trotz Anonymität in offenen digitalen Systemen. Datenschutz und Datensicherung DuD (1990).

[VaVa_85] Umesh V. Vazirani, Vijay V. Vazirani: Efficient and Secure Pseudo-Random Number Generation (extended abstract). In: Crypto '84, Band 196 von *LNCS*, Springer-Verlag, Berlin (1985), 193–202.

[WaPf_89] Michael Waidner, Birgit Pfitzmann: Unconditional Sender and Recipient Untraceability in spite of Active Attacks - Some Remarks. Fakultät für Informatik, Universität Karlsruhe (März 1989), Interner Bericht 5/89.

A Lösungsblätter

Aufgabe 1a)

Leerbotschaft	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
Schlüssel Rüdiger-Petra	22 29 28 25 3	16 7 3 16 19	28 21 14 25 27	16 27 14 19 22
Schlüssel Rüdiger-Sabine				
Rüdigers lokale Summe				
in Zeichen				

Aufgabe 1b)

lokale Summe von Petra				
lokale Summe von Eduard				
lokale Summe von Sabine				
Rüdigers lokale Summe				
globale Summe				
in Zeichen				

Aufgabe 2

lokale Summe von Sabine				
Schlüssel				
Zwischenergebnis				
Schlüssel				
Sabines Botschaft				

Aufgabe 3a)

lokale Summe von Eduard				
lokale Summe von Sabine				
Summe Eduard + Sabine				

lokale Summe von Petra				
lokale Summe von Rüdiger				
Summe Petra + Rüdiger				

Aufgabe 3b)

Summe Eduard + Sabine				
Schlüssel				
Ergebnis Teilnetz 1				

Summe Petra + Rüdiger				
Schlüssel				
Ergebnis Teilnetz 2				

Aufgabe 4a)

lokale Summe von Eduard				
Schlüssel				
Eduards Botschaft				

Aufgabe 4b)

lokale Summe von Petra				
lokale Summe von Rüdiger				
Summe Petra + Rüdiger				
Schlüssel				
Teilnetz Petra + Rüdiger				

Aufgabe 4c)

Schlüssel Petra-Rüdiger				

Aufgabe 4d)

Schlüssel Petra-Rüdiger				

Aufgabe 5

Kollisionsergebnis				

Aufgabe 6a)

Kollisionsergebnis				
wiederholte Botschaft				
zweite Botschaft				

Aufgabe 6b)

1. DC-Runde (4 Botsch.)				
2. DC-Runde (1. Botsch.)				
drei Restkollisionen				

Auflösung der drei Restkollisionen:

drei Restkollisionen				
3. DC-Runde (2 Botsch.)				
zweite Botschaft				

Kollision der 3. DC-Runde				
4. DC-Runde (3. Botsch.)				
vierte Botschaft (berechnet)				

B Rechentabellen für das Praktikumsalphabet

Zeichencodierung des Praktikumsalphabets

_	A	B	C	D	E	F	G	H	I	J	K	L	M	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

Invertierungstabelle

_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
_	?	!	.	Z	Y	X	W	V	U	T	S	R	Q	P	O

Die übereinanderstehenden Zeichen sind jeweils zueinander invers.

Additionstabelle

+	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?
_	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
.	.	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
!	!	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.
?	?	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	!

C Nachrichten der Aufgaben 2 bis 4

Teilnehmer: Petra, Rüdiger, Sabine, Eduard.

Runde 1: **{Aufgabe 2}**

„YMGECG LNPVJNQMQXXNN“	lokale Summe ⇐ Petra
„!QDQ.KHPCB?TZ!MPKWTX“	lokale Summe ⇐ Rüdiger
„VHFIFDMTAXQRXHH NRYU“	lokale Summe ⇐ Sabine
„H.A?C.?LTWJ ZHHQJYAA“	lokale Summe ⇐ Eduard
„WER IST HERR ALT? “	globale Summe

Runde 2: **{Aufgabe 3}**

„RSPWLAYN.ZCREGHIUNSN“	lokale Summe ⇐ Petra
„SF ?GUBPERMZHXKDUAX “	lokale Summe ⇐ Rüdiger
„MEVSK?AZAI.LB?FD?FUM“	lokale Summe ⇐ Sabine
„SC Q IBD.GQDO EMSIZC“	lokale Summe ⇐ Eduard
„ICH! “	globale Summe

Runde 3: **{Aufgabe 4}**

„AWMGEM!Z RRJTMC?ZJFH“	lokale Summe ⇐ Petra
„ST.W F PWZ MUJK UQLF“	lokale Summe ⇐ Rüdiger
„?WHQGSQXAIKRBMJFSHGI“	lokale Summe ⇐ Sabine
„DYAMGG.FFKBHPXFYXYEG“	lokale Summe ⇐ Eduard
„WAS SOLL DAS? “	globale Summe