5. Netzwerke

Das Internet, wenn man es als Einheit betrachtet, ist wahrscheinlich das komplizierteste von Menschenhand geschaffene Artefakt. Milliarden Computer, in denen Chips aus Milliarden Transistoren verbaut sind, arbeiten zusammen um die Elektronen in Kupferkabeln gerade so anzuschubsen, dass die Informationen, die man benötigt um die einzelnen Pixel eines Bildschirms die richtigen Farben zeigen zu lassen, damit sie ein Photo einer weit entfernt lebenden Katze bilden, in Sekundenbruchteilen quer über den Globus übertragen werden.

Versucht man sich vorzustellen, wie sich die Elektronen im Telefonkabel bewegen wenn wir eine Webseite aufrufen, ist man von der Komplexität sofort und vollkommen überwältigt. Wie man ein solches System überhaupt bauen kann, und wie es so zuverlässig funktionieren kann scheint unverständlich. In diesem Kapitel werden wir ein extrem wichtiges Werkzeug kennen lernen**FiXme Note: haben wir vielleicht schon vorher kennen gelernt**, um die Komplexität technischer Systeme wie dem Internet zu bändigen.

Um Daten z.B. über Telefonkabel zu übertragen, muss sich natürlich jemand irgendwann Gedanken darüber gemacht haben, wie die Elektronen sich bewegen müssen, um einzelne Bits zu übertragen. Der kritische Schritt ist, die Hardware so zu bauen, dass sich Benutzer nicht darum kümmern müssen, wie genau die Spannung auf dem Kabel sein muss um eine 1 zu senden und wie lange man das Signal aufrechterhalten muss, um sicherzugehen, dass sie auf der anderen Seite gelesen werden kann.

Man *abstrahiert* von den technischen Details und präsentiert dem Nutzer ein abstraktes Modell der Datenübertragung, das einfache Methoden zum Senden einzelner Bits bietet. Auf die gleiche Art möchte man sich als Programmierer eines Webbrowsers nicht darum kümmern, welche Bits man über die Leitung schicken muss, um den Computer des Internetdienstanbieters darum zu bitten, sie an den Betreiber der Webseite, die man anzeigen möchte, weiterzuleiten. Deswegen führt man eine weitere Abstraktionsschicht ein. Man benutzt das Modell des Bittransfers, um paketbasierte Datenübertragung zu realisieren. Statt über einzelne Bits denkt man jetzt mit Paketen, die Adressen und Daten enthalten und automatisch von Rechner zu Rechner ans Ziel weitergeleitet werden.

Manchmal ist die Übertragung eines Pakets nicht erfolgreich, weil zum Beispiel gerade die Mikrowelle angegangen ist und die Funkverbindung kurz gestört war. Man möchte sich nicht darum kümmern müssen festzustellen ob ein Paket angekommen ist und ob der Rechner, mit dem man eigentlich sprechen möchte, einen überhaupt gehört hat. Man benutzt also das Paketmodell der Datenübertragung um störungsunempfliche

Verbindungen zu bauen, über die nicht Pakete, sondern Datenströme ausgetauscht werden.

Auf diese Art baut man nach und nach mehrere Abstraktionsschichten, die mehr und mehr Komfort für die Benutzung bieten. Dabei benutzt man beim Erstellen einer neuen Schicht stets nur die von direkt darunter liegenden Schicht angebotenen Werkzeuge. Der Programmierer der Datenstromabstraktion braucht also nicht zu wissen, wie einzelne Bits übertragen werden, weil er nur die bereits vorhandene Paketabstraktion benutzt. Dadurch kann man einzelne Teile auch leicht austauschen. Für den Programmierer macht es so keinen Unterschied, ob man per Kabel, per WLAN oder über eine Satellitenverbindung an das Netzwerk angeschlossen ist, weil sich immer nur die unterste Schicht ändert.

In diesem Kapitel werden wir einen sehr groben Überblick über die einzelnen Schichten geben und uns in einem zweiten Teil darüber Gedanken machen, wie man als Nutzer des Internets anonym bleiben kann.

5.1. Einzelne Bits Austauschen

Im einfachsten Fall wollen wir Bits zwischen zwei Computern austauschen. Zu diesem Zweck verbinden wir die beiden Computer mit einem Kabel. Jetzt können wir verschiedene Spannungen an das Kabel anlegen, um Signale auszutauschen. Zum Beispiel können wir uns darauf einigen, dass wir $+5$V anlegen, wenn wir eine 1 übertragen wollen und -5V, wenn wir eine 0 übertragen. In modernen Kabelnetzwerken benutzt man mehr Spannungsstufen und mehrere Kabel um mehr als nur ein Bit gleichzeitig übertragen zu können. Außerdem wechselt man nicht für jedes Bit die Spannung, um die Frequenz auf der Leitung gering zu halten. Das verringert das durch die Kabel hervorgerufene Störsignal und vereinfacht zusätzlich die Messung der Spannung, weil sie für längere Zeit anliegt.

Was für Probleme treten bei dieser Art der Datenübertragung auf? Zunächst müssen sich beide Computer darauf einigen, wann die Spannung auf der Leitung gemessen wird. Das ist nicht trivial. Bei hohen Übertragungsraten liegt jede Spannung nur sehr kurz an, typischerweise in der Größenordnung von 10 Nanosekunden. Bei so kurzen Zeiten ist es nicht ohne weiteres möglich die in den beiden Computern eingebauten Uhren zu benutzen, um die Signalmessung zu synchronisieren. Weil die Uhren mit leicht unterschiedlicher Geschwindigkeit laufen, würde man so schnell aus dem Takt kommen und keine ordentlichen Werte mehr messen.

Dieser Uhrenfehler kommt besonders dann zum Tragen, wenn viele gleiche Bits übertragen werden. Wenn sich das Signal oft genug ändert, kann man es selbst zur Synchronisierung verwenden. Um zu verhindern, dass wir lange Folgen von Einsen oder Nullen senden, kodieren wir Folgen von Bits. Wir teilen die Bitfolge, die wir senden wollen, in

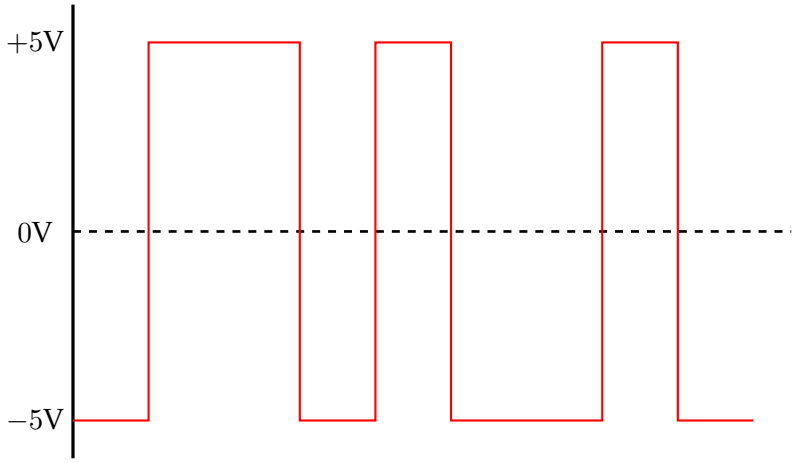

Abbildung 5.1.: Übertragen der Bitfolge `011010010` durch Spannung auf einem Kabel. Die Darstellung ist idealisiert, ein reales Rechtecksignal ist in Abbildung 5.2 gezeigt.

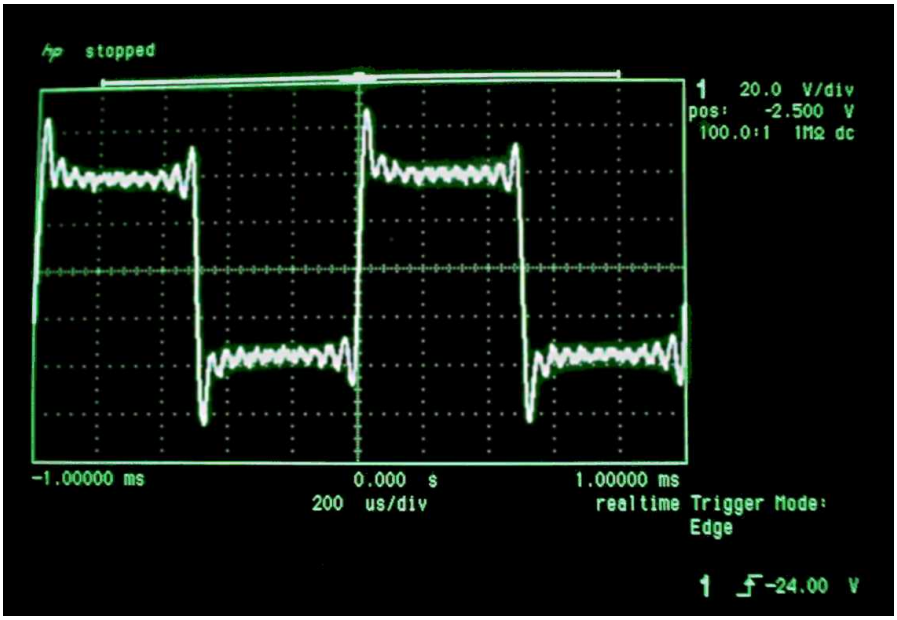

Abbildung 5.2.: Ein reales Rechtecksignal. Jedes Rechteck ist etwa 500μs**FiXme Note: upshape mu** groß, also ungefähr 50 000 Mal länger als ein Signal auf einem Netzwerkkabel.

Blöcke von vier Bits auf und ersetzen jeden Block durch einen Block der Länge fünf. Das nennt man 4B5B Kodierung.

Insbesondere ersetzen wir `0000` durch `11110` und `1111` durch `11101`. Da es mehr fünfstellige Bitfolgen gibt als vierstellige, bleiben ein paar Kombinationen übrig. Diese können wir Statusmeldungen verwendet werden, zum Beispiel um den Beginn oder das Ende der Datenübertragung zu signalisieren.

Ein weiteres Problem sind Fehler bei der Datenübertragung. Selbst wenn wir von ausgezeichneten Bedingungen ausgehen, bei denen nur 0.00001% der Bits falsch übertragen werden, treten bei den üblichen Netzwerken, die ein Gigabit pro Sekunde übertragen, zehn Fehler pro Sekunde auf.

Um diese Fehler zu erkennen, teilen wir die Bits, die wir senden wollen, in Pakete einer festen Länge auf und fügen eine sogenannte Prüfsumme an jedes Paket an. Eine sehr einfache Prüfsumme besteht aus einem einzelnen Bit, das wir auf Eins setzen, wenn die Anzahl der Einsen im Paket gerade war und sonst auf Null. Der Empfänger kann dann einfach nachzählen und erkennt immer, wenn sich ein einzelnes Bit unterwegs geändert hat. Wenn sich zwei Bits in einem Paket ändern, kann dieser Test aber schon fehlschlagen.

Weil fehlertolerante Datenübertragung in vielen Bereichen von großer Bedeutung ist, gibt es viele raffinierte Möglichkeiten Prüfsummen zu berechnen. Man möchte mit möglichst wenig zusätzlichen Bits möglichst viele falsch übertragene Bits erkennen können. Das Verhältnis von Gesamtlänge zu Datenlänge nennt man die *Effizienz* eines Kodierungsschemas.

Mit der Überprüfung der Parität, wie wir sie oben beschrieben haben, fügen wir ein Bit hinzu und können ein falsches Bit erkennen. Da wir nur ein Bit hinzufügen, ist dieses Schema sehr effizient, aber dafür können wir auch nur sehr wenige Fehler erkennen. Richard Hamming hat 1950 die nach ihm benannten *Hamming Codes* beschrieben, mit denen man einen Fehler nicht nur korrigieren, sondern sogar berichtigen und zwei Fehler erkennen kann.

Es gibt verschiedene Hamming Codes, abhängig von der Länge der Blöcke, in die man seine Bits teilt. Allgemein gilt, dass die Effizienz mit der Länge der Blöcke steigt. Mit m Bits können wir eine Prüfsumme für Blöcke der Länge $n = 2^m - m - 1$ wie folgt berechnen.

Wir nummerieren die Bitpositionen von 1 bin $n + m$ durch. Die Bits der Prüfsumme schreiben wir an die Positionen $1, 2, 4, 8, \ldots$, also an Positionen, die mit einer Zweierpotenz nummeriert sind. Jedes Prüfsummenbit kodiert die Parität der Einsen in einer Teilmenge der Bits. Das i-te Prüfsummenbit speichert die Parität der Positionen, die eine Eins an der i-ten Stelle ihrer Binärschreibweise haben (vgl. Abschnitt 1 im Anhang, Seite 89), sich selbst ausgenommen. Das bedeutet, dass das erste Bit die Parität an den ungeraden Positionen kodiert. Das zweite Bit speichert die Parität der Positionen $3, 6,$

Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p_1															
p_2															
p_4															
p_8															

Tabelle 5.1.: Ein Hamming Code für 11 Datenbits.

$7, 10, 11, 14, 15, \ldots$ und das dritte Bit speichert die Parität der Positionen $5 - 7$, $12 - 15$, $20 - 23, \ldots$ usw.

Jedes Bit fließt also in mehrere Prüfsummenbits ein aber keine zwei Datenbits beieinflussen genau die selben Prüfsummenbits. Das erlaubt es festzustellen welches Bit sich bei der Übertragung geändert hat. In Tabelle 5.1 sieht man den Hamming Code für 11 Datenbits.

Eine grüne Zelle gibt an, dass das Bit an dieser Stelle in die Berechnung des Prüfbits einfließt. Wie man sieht ist jedes Datenbit von mindestens zwei Prüfbits abgedeckt. Wenn also nur ein Prüfbit einen Fehler anzeigt, ist das Prüfbit selber falsch. Ansonsten kann man an Hand des Musters der falschen Prüfbits die Position des Fehlers eindeutig feststellen und ihn beheben.

Wir übertragen Bits zwischen zwei Computern also wie folgt:

1. Bits in Pakete einteilen.
2. Pakete mit Prüfsummen ausstatten.
3. Erneut in Blöcke aufteilen.
4. Blöcke mit 4B5B kodieren.
5. Bits in Spannungen übersetzen.

Wenn mehr als zwei Computer miteinander kommunizieren sollen treten weitere Probleme auf. Zunächst muss man sich Gedanken machen, wie man sie miteinander verbindet. Man kann zum Beispiel zwischen je zwei Computern ein Kabel legen. Diese Lösung ist nicht sehr praktikabel, weil man schon für wenige Computer viele Kabel braucht. Alternativ hängen mehrere Computer an einem Kabel, dann muss aber darauf geachtet werden, dass nicht zwei Computer gleichzeitig versuchen Daten zu übertragen, sonst stören sie ihre Signale gegenseitig.

In der Praxis baut man Netzwerke im wesentlichen baumförmig auf[1]. Dadurch hat man nicht das Problem, dass mehrere Computer gleichzeitig auf dem selben Kabel Daten übertragen möchten, und man braucht nicht so viele Kabel. Dafür gibt es jetzt keinen

[1] Weil Hardware auch manchmal kaputt geht, legt man oft mehr Kabel als nötig, so dass die Verkabelung nicht mehr baumförmig aussieht. Spezielle Software auf den Computern bemerkt aber die Existenz von Kreisen und schaltet selektiv manche Verbindungen ab

direkten Draht mehr zwischen jedem Paar Computern. Die inneren Knoten des Baums haben dann die besondere Aufgabe Daten weiterzuleiten, die nicht für sie bestimmt sind, damit auch zwei Blätter miteinander reden können.

Zu diesem Zweck muss natürlich an den Daten stehen für welchen Computer sie bestimmt sind. Jeder Computer bekommt also eine sogenannte *physikalische Adresse*. An jedes Datenpaket werden sowohl Ziel- als auch Absenderadresse vorne angehängt.

Damit nicht ein Mensch mühsam für jeden innen Knoten im Netzwerk angeben muss über welches seiner angeschlossenen Kabel er Pakete in Richtung Adresse X weiterleiten muss, schaut jeder innere Knoten auf die Absenderadresse eingehender Pakete und merkt sich von welchem Kabel her sie kommen. Wenn er jetzt ein Paket weiterleiten soll, weiß er entweder schon das richtige Kabel, oder er sendet es einfach über alle ausgehenden Kabel weiter.

5.2. Pakete Routen

Die Beschreibung, die wir bisher gegeben haben, beschränkt sich auf homogene Netzwerke, bei denen alle Computer über die gleiche Technik miteinander verbunden sind. Die normale Netzwerktechnik, die man in einem Gebäude einsetzen kann, ist aber ungeeignet, wenn man Daten quer über den Atlantik übertragen möchte, oder eine Videokonferenz mit der ISS organisieren muss.

Man führt also eine weitere Abstraktionsebene ein, die die genauen physikalischen Gegebenheiten des Netzwerks versteckt und die Kommunikation zwischen verschiedenen Netzwerken ermöglicht.

Computer bekommen eine weitere Adresse, die sogenannte *Internet Protocol* (IP) Adresse. IP-Adressen bestehen aus vier Blöcken von Zahlen zwischen 0 und 255. Mein Computer hat zum Beispiel gerade die IP-Adresse `95.90.165.155`. Diese Adressen sind hierarchisch, Computer im gleichen Netzwerk teilen sind den Anfang ihrer IP-Adresse. Kunden von Kabel Deutschland befinden sich zum Beispiel im Adressraum zwischen `95.90.0.0` und `95.91.255.255`.

Jeder Computer hat eine Tabelle, in der steht an welchen Computer man am besten Pakete mit einer bestimmten Adresse schickt. Für die meisten Computer ist diese Tabelle sehr einfach. Mein Computer schickt zum Beispiel alle Pakete, die nicht an einem anderen Computer in meinem lokalen Netzwerk gehen, weiter an das Modem des Internetdienstleisters, weil das ja der Verbindungspunkt zum gesamten Internet ist. Für Adressen im lokalen Netzwerk stehen in dieser Tabelle direkt die physikalischen Adressen.

Die Internetbetreiber haben natürlich eine viel schwierigere Aufgabe. Sie müssen Pakete auf einem möglichst kurzen Weg zum Ziel transportieren. Dazu berechnen die Computer kontuinierlich kürzeste Wege (vgl. **??**), indem sie die ihnen bekannten Adressen und die

Länge des Weges von ihnen aus bis zum Ziel an ihre direkten Nachbarn im Netzwerk schicken.

Das normale Internet-Protokoll macht keine Garantien zur Zuverlässigkeit der Datenübertragung. Pakete können mehrfach ankommen, oder gar nicht, und ihre Reihenfolge kann sich ändern (zum Beispiel weil sie verschiedene Routen im Netzwerk nehmen). Typische Verbindungen zwischen Computern benutzen deswegen nicht direkt das Internet-Protokoll, sondern eine weiter Abstraktion, die darauf aufbaut.

Das Transport Control Protocol (TCP) benutzt IP-Pakete um zuverlässige Datenübertragung zu realisieren. Dazu erhält jedes Paket eine Nummer und der Empfänger des Pakets muss eine Quittung zurück schicken. Geht entweder das Paket oder die Quittung verloren, wird es automatisch erneut gesendet. An Hand der Nummern kann auf die eventuelle Umordnung der Pakete erkannt werden.

5.3. Anonymität

An Hand der IP Adressen, die in jedem Paket stehen, kann jemand, der den Internetverkehr abhören kann, rekonstruieren wer mit wem geredet hat. Die Benutzer können zwar die Inhalte ihrer Kommunktation verschlüsseln (vgl. **??**), aber wenn die Adressen verschlüsselt sind, können die Pakete nicht ans Ziel kommen.

Es scheint nicht so einfach, diese Verbindungsdaten vor einem Beobachter zu verstecken. Es ist aber nicht unmöglich. Die Idee ist es, Pakete nicht direkt ans Ziel zu schicken, sondern ein paar Zwischenstationen zu benutzen. Der Trick ist es dabei, es unmöglich zu machen zu unterscheiden, ob eine Zwischenstation ein Paket nur weiterleitet, oder ob es von ihr ausgeht. Irgendwann muss natürlich jedes Paket diese Gruppe von Zwischenstationen verlassen, aber man kann dann nicht mehr feststellen, von welchem Computer es ursprünglich ausgegangen ist.

Um an einem solchen Weiterleitungsnetz teilzunehmen, lädt sich mein Computer zunächst von einer öffentlichen Stelle eine Liste aller Transitcomputer runter. Dann sucht er sich zufällig eine Route von Transitknoten aus, die er benutzen möchte.

Mein Computer schickt zwei Arten Mitteilungen. Entweder ein „Hallo“ (H), oder ein „Bitte schick M weiter an X“ ($W_X(M)$). Ein H enthält meinen öffentlichen Schlüssel und der Adressat antwortet mit dem seinigen.[2] Nachdem ich ein H an einen Computer geschickt hab, kann ich also verschlüsselt mit ihm kommunizieren. Ein W ist eine Kiste, in die man weitere Pakete stecken kann. Der Inhalt und die Adresse an der er weitergeleitet werden soll ist nur für den Empfänger lesbar, man muss also zuerst ein H

[2] Das ist eine vereinfachte Darstellung. Tatsächlich einigen sich die Computer auf einen gemeinsamen Schlüssel für ein symmetrisches Verschlüsselungsverfahren. Dazu kann man beispielsweise den Diffie-Hellman-Schlüsselaustausch benutzen. Wenn wirklich öffentliche Schlüssel ausgetauscht würden, könnte man Computer durch ihre Schlüssel identifizieren.

schicken, bevor W's verschickt werden können. Wenn man ein W empfängt, packt man den Inhalt aus, entschlüsselt ihn und schickt ihn an den Empfänger des Inhalts weiter.

Damit man auch antworten auf die weitergeleiteten Pakete erhalten kann, merkt sich die Transitcomputer von wem er Pakete wohin weitergeleitet hat und kann so Antworten zurückschicken.

Nehmen wir an ich wähle die Route A, B, C. Um alle Computer auf der auf mir ausgewählten Route zu rekrutieren, schicke ich Computer A ein H. Anschließend schicke ich ihm ein $W_B(H)$ mit einem H für den zweiten Computer darin. Für B sieht es so aus als würde A ihm ein H schicken[3], aber tatsächlich werden seine Antworten an mich zurückgeschickt. Anschließend kann ich also mit B verschlüsselt kommunizieren. Um mit C zu reden, schicke ich A ein $W_B(W_C(H))$, also ein W mit einem W mit einem H darin zu. Er packt ein W aus und schickt $W_C(H)$ an B. Da das innere $W_C(H)$ mit B's Schlüssel verschlüsselt ist, weiß A nicht worum es sich beim Inhalt des Pakets handelt. Dann packt B das zweite W aus und leitet das H an C weiter. Die Antwort von C kommt dann über zwei Zwischenstationen an mich zurück, aber für ihn sieht es so aus als hätte B alles initialisiert.

Auf einer Route kennt jeder Computer also nur seinen direkten Vorgänger und seinen direkten Nachfolger und für jeden Computer auf der Route sieht es so aus, als hätte sein Vorgänger die Route initialisiert. Damit ein Angreifer die Anonymisierung brechen kann, muss er alle Computer auf der Route kontrollieren. Dann kennt er alle Schlüssel, kann die Pakete entschlüsseln und den Absender ausfindig machen.

Um das System tatsächlich anonym zu machen muss man auch noch auf ein paar andere Probleme achten. Zum Beispiel kann ein Angreifer, der die Computer nicht kontrolliert, aber ihren Netzwerkverkehr beobachten kann anhand der Zeitpunkte zu denen Pakete verschickt werden die Routen rekonstruieren. Wenn immer zuerst A, dann B, und dann C ein Paket schicken, nachdem sie gerade ein Paket vom jeweiligen Vorgänger erhalten haben, kann man davon ausgehen, dass sie zu einer Route gehören. Hat man einmal der Ursprung von verdächtigem Verkehr lokalisiert, kann man den Computer direkt angreifen. Die beste Verschlüsselungs- und Anonymisierungssoftware hilft wenig, wenn nebenher noch ein Programm läuft, dass den Datenverkehr direkt aus dem Speicher des Computers abgreift.

Außerdem kann man mit diesem System nur die IP-Adresse verbergen. Das reicht natürlich nicht aus, wenn man mit einer verborgenen IP-Adresse Dinge tut, die einen sofort identifizieren. Zum Beispiel indem man sich in ein soziales Netzwerk eingloggt, oder seine normale E-Mailadresse benutzt. Tatsächlich anonym zu bleiben ist sehr schwierig. Schon recht unverfängliche Daten reichen aus, um einen eindeutig zu identifizieren. Zum Beispiel reicht es aus zu wissen, in welcher Postleitzahl man wohnt und in welcher man arbeitet um identifiziert zu werden [3]. Auch wenn man einen markanten Schreibstil hat, kann man erkannt werden. Technische Lösungen können es daher die

[3] Wenn öffentliche Schlüssel ausgetauscht würden, könnte B sehen, dass es sich um meinen öffentlichen Schlüssel handelt, nicht um den von A.

Rückverfolgung nur erschweren, gänzlich verhindern kann man sie wahrscheinlich nicht.