UNIVERSITY OF APPLIED SCIENCES

Department of Computer Science

WS 2015/2016

Seminar
IT Sicherheit

Bitmessage

Autor: Meik Jejkal
Betreuer: Prof. Dr. Gerd Beuster

abgegeben am: 15. Februar 2016

Inhalt

Abbildungsverzeichnis

1 Einleitung

E-Mails sind aus der heutigen Kommunikation sowohl privat als auch geschäftlich nicht mehr wegzudenken. Für gewöhnlich werden sie in Klartext verschickt. So können sie leicht eingesehen, oder manipuliert werden. Um diesen Umstand zu vermeiden, wurden Verschlüsselungsverfahren wie PGP oder S/MIME entwickelt. Die Schutzziele der Vertraulichkeit und der Integrität sollen so gesichert werden. Damit diese Verschlüsselungsverfahren sicher verwendet werden können, wird jedoch eine gewisse Fachkenntnis vorausgesetzt. Es müssen asymmetrische Schlüssel erzeugt und über einen sicheren Kanal ausgetauscht werden. Im Anschluss daran können E-Mails verschlüsselt und signiert werden. Dabei geht es nur um den Inhalt der Nachrichten. Es fallen allerdings auch Metadaten wie Absender, Empfänger, Zeitstempel, oder IP-Adressen beim Mailserver an. Derartige Informationen können genutzt werden, um ein Profil über das Kommunikationsverhalten zu erstellen. Metadaten werden häufig unterschätzt. Nehmen wir eine regelmäßige Kommunikation mit einem Inkasso Unternehmen, oder einer Beratungsstelle für Alkohol. Der Inhalt der Nachrichten ist dabei unerheblich. Viele Institutionen, wie beispielweise Krankenkassen oder Versicherungen, entwickeln ein großes Interesse an derartigen Metadaten, um diese mit bestehenden Datensätzen zu verknüpfen. Mit derartigen Profilen, ergeben sich neue wirtschaftliche Möglichkeiten für beteiligte Unternehmen.

Sichere Kommunikation ist keine Luxus-Funktion, sie dient zur Wahrung der Privatsphäre. Von politisch Verfolgten, bis hin zu Journalisten oder Geheimnisträgern können sogar Leben davon abhängen. In der Wirtschaft bedeuten Firmengeheimnisse mitunter ganze Existenzen. Bitmessage ist ein Lösungsvorschlag für sichere Kommunikation. Es soll sich ohne weitergehende Kenntnisse nutzen lassen und die Wahrung von Vertraulichkeit und Integrität sicher stellen. [Wara]

2 Bitmessage

Bitmessage ist ein Protokoll von Jonathan Warren [Ward]. Es soll dazu dienen, Textnachrichten zwischen verschiedenen Benutzern auszutauschen. Diese werden mit Hilfe eines Public-Key-Verschlüsselungsverfahrens chiffriert, um die Vertraulichkeit sowie Integrität der Nachrichten sicher zu stellen. Parallel zum einfachen Nachrichtenaustausch wird auch Broadcasting unterstützt. Dabei können Nachrichten von bestimmten Absendern abonniert werden, ähnlich den News-Feeds innerhalb des regulären E-Mail-Protokolls.

Das Bitmessage-Protokoll arbeitet dezentral über ein P2P-Netzwerk und übernimmt das Schlüsselmanagement. Es ist somit nicht nötig asymmetrische Schlüsselpaare selbstständig zu generieren oder weiterzuleiten. Durch die P2P-Architektur entfällt ein Mailserver als zentrale Datensenke für Metadaten. Postfächer werden ebenfalls nicht serverseitig bereitgestellt, sondern von den einzelnen Nutzern lokal verwaltet. Die Nachrichten werden mit Hilfe einer Blockchain übertragen, vergleichbar mit den Transaktionen bei der Cryptowährung Bitcoin [Proa]. Aus Performance Gründen wird diese in verschiedene sog. Streams aufgeteilt, sodass nicht jeder Teilnehmer des Netzwerkes die gesamte Blockchain vorhalten muss. Jede Nachricht besitzt eine Time-to-Live, die bestimmt, wann sie aus der Blockchain entfernt wird. Meldet sich ein Teilnehmer innerhalb dieser Zeit nicht am Netzwerk an, so wird die Nachricht aus der Blockchain entfernt und ist damit verworfen. Jeder Teilnehmer des Netzwerks kann nur die Nachrichten entschlüsseln, die mit seinem öffentlichen Schlüssel verschlüsselt wurden.

Anhänge werden nicht unterstützt, da diese die Sicherheit beeinträchtigen könnten. Bilder mit Tracking-Pixeln könnten zum Beispiel dazu verwendet werden, Nutzer des Netzwerkes zu deanonymisieren. Eine Bitmessage-Adresse lässt sich wechseln, ohne dabei den Posteingang zu beeinträchtigen. Bisher wurde eine plattformübergreifende Referenz Implementierung namens pyBitmessage für Windows, OSX, und Linux auf GitHub veröffentlicht[Bit]. Dieser Client wird in Kapitel 4 näher behandelt.

2.1 Adressen

Analog zur regulären E-Mail, werden zur Kommunikation mit anderen Bitmessage Nutzern ebenfalls Adressen genutzt. Die Verwaltung der Adressen übernimmt der Bitmessage Client selbst. Es können beliebig viele Adressen generiert werden. Der Ablauf ist dabei immer folgender:
Der Client erzeugt zwei secp256k1 asymmetrische Public Key Schlüsselpaare. Ein Schlüsselpaar dient dabei der Verschlüsselung, das andere der Signatur von Nachrichten. Die jeweiligen Public-Keys werden unkomprimiert im X9.62-Format zusammengefügt. Nun wird ein SHA512-Hash über die beiden Public Keys gebildet. Das Ergebnis wird mittels RIPEMD-160 erneut gehashed. Die beiden Hashing Schritte können so lange wiederholt werden, bis das Ergebnis mit einer oder zwei Nullen beginnt. Diese können an dieser Position aus der Adresse entfernt werden, um sie bis zu 2 Zeichen kürzer zu machen [Prob]. Da dieser Vorgang extra Rechenzeit benötigt, ist er rein optional. Die Adressverkürzung muss beim Client pyBitmessage beispielsweise erst per Checkbox aktiviert werden. Der Ergebnis Hash umfasst nun noch 20 Byte von den ursprünglichen 128 Byte Ausgangsdaten. Zum Abschluss wird noch das Präfix BM- vorangestellt, gefolgt von der Nummer des Streams [Kapitel 3.3] auf dem primär Nachrichten verarbeitet werden. So könnte eine Adresse beispielsweise wie folgt aussehen:

BM-2nTX1KchxgnmHvy9ntCN9r7sgKTraxczzyE

Die Adresse wird in base58 kodiert, um den Zeichensatz auf relevante Zeichen einzuschränken. Eine Verwechslung von Zeichen wie I und 1 oder 0 und O wird damit ebenfalls ausgeschlossen. In der Standardeinstellung werden Zufallszahlen zur Adressgenerierung verwendet. Es ist allerdings auch möglich Adressen deterministisch zu generieren. Dabei werden die Adressen aus einem Kennwort generiert, dass als Initialwert für den Zufallsgenerator verwendet wird. Dies hat den Vorteil, dass der Adresssatz, mit Hilfe des Kennworts, auf beliebigen Clients wiederhergestellt werden kann. Zusätzlich wird hierzu noch die Stream-Nummer der Adressen benötigt, da diese außerhalb des Zufallsgenerators bestimmt wird. Das Kennwort muss besonders gut geschützt werden, denn mit seiner Hilfe können beliebige Personen den Adresssatz inklusive zugehöriger Schlüssel erzeugen. Es empfiehlt sich deshalb gleich ein ganzer Kennwortsatz, um Brute-Force-Angriffe zu erschweren. Bitmessage Adressen können auf verschiedene Weisen ausgetauscht werden. Zum Beispiel können diese in einen QR-Code (Abbildung 2.1) konvertiert, oder auch auf RFID Tags gespeichert werden.

BM-NBZ34dF2A31BCEcSKDmqtZ7mo8GMVqSh

Abbildung 2.1: Bitmessage-Adresse als QR-Code

2.2 Netzwerk

Die Nachrichten werden über ein P2P Netzwerk innerhalb einer Blockchain an die einzelnen Clients verteilt. Ein erster Verbindungsaufbau eines neuen Clients ist in Abbildung 2.2 dargestellt:

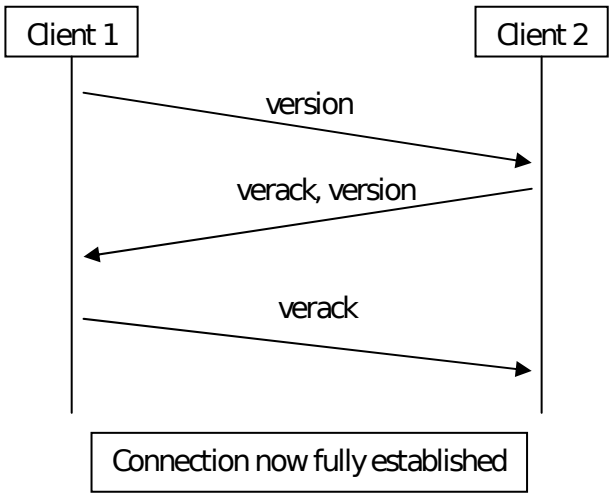

Abbildung 2.2: Bitmessage Handshake [Warc]

Verbindet sich ein Client das erste Mal mit dem Netzwerk, erhält er zunächst von einem festgelegten Server eine Liste mit bekannten Netzwerkteilnehmern. Bevor nun eine Kommunikation stattfindet, wird geprüft, ob die einzelnen Clients die selbe Protokoll Version ausführen. Dazu sendet er eine leere Nachricht mit der Versionsnummer und einigen anderen Parametern im Header an den Client. Darunter auch das Feld „services“ welches den Wert „NODE_NETWORK“ oder „NODE_SSL“ annehmen kann. Die erste Option steht für einen normalen Client, die zweite für die Unterstützung einer verschlüsselten Verbindung über SSL/TLS. Liegt eine Übereinstimmung der Version in den Headern vor, so wird dies durch eine ebenfalls leere „verack“ Nachricht bestätigt. Erfolgt keine Antwort auf die Nachricht, wird der Vorgang nach 20 Sekunden mit einem Timeout abgebrochen. Im Anschluss wird dieser Prozess vom anderen Client wiederholt. Stellen nun beide Clients über den „services“ Parameter fest, dass sie SSL/TLS unterstützen, so wird ein SSL/TLS Handshake ausgeführt. Dieser muss nun innerhalb von 10 Minuten erfolgen, da ansonsten wieder ein Timeout erfolgt. Nach erfolgreichem Handshake, findet die weitere Kommunikation zwischen den beteiligten Clients verschlüsselt statt. Vorraussetzung hierfür ist allerdings die Übereinstimmung der Version in den Nachrichten-Headern. Momentan liegt die aktuelle Versionsnummer des Bitmessage-Protokolls bei 3.0 (Stand: 15. Februar 2016).

Die Verbindung ins P2P-Netzwerk kann dabei auch über weitere Dienste wie Tor [Proc] aufgebaut werden. Hierzu ist allerdings ein SOCKS-Proxy erforderlich, wie ihn zum Beispiel das Tor-Browser-Bundle [Prod] mitbringt. Die Anzahl der Verbindungen wird durch Firewalls auf 8 beschränkt. Diese sind bei Privatanschlüssen bereits fest in Routern integriert. Die Verbindungsanzahl lässt sich allerdings erhöhen, wenn Port 8444 als NAT/Firewall Regel auf das System des Clients weitergeleitet wird.

3 Nachrichtenaustausch

Es gibt 4 verschiedene Kommunikations-Typen innhalb von Blockchain-Streams. Sie lauten „getpubkey“, „pubkey“, „msg“, und „broadcast“. Darüber hinaus tauschen die einzelnen Clients zusätzlich Nachrichten aus, um Daten abzugleichen. Mit einer „getpubkey“ Nachricht wird der zu einer Bitmessage Adresse gehörige Public Key angefragt. Als Antwort dient die „pubkey“ Nachricht die diesen enthält. Die klassische person-to-person Nachricht wird mit Hilfe von „msg“ realisiert. Die „broadcast“ Nachricht versendet eine Nachricht an jeden Abonnenten, der sich zuvor für Nachrichten von diesem Absender angemeldet hat. Folgende Abbildung 3.1 zeigt schematisch den Datenaustausch zwischen einzelnen Clients:

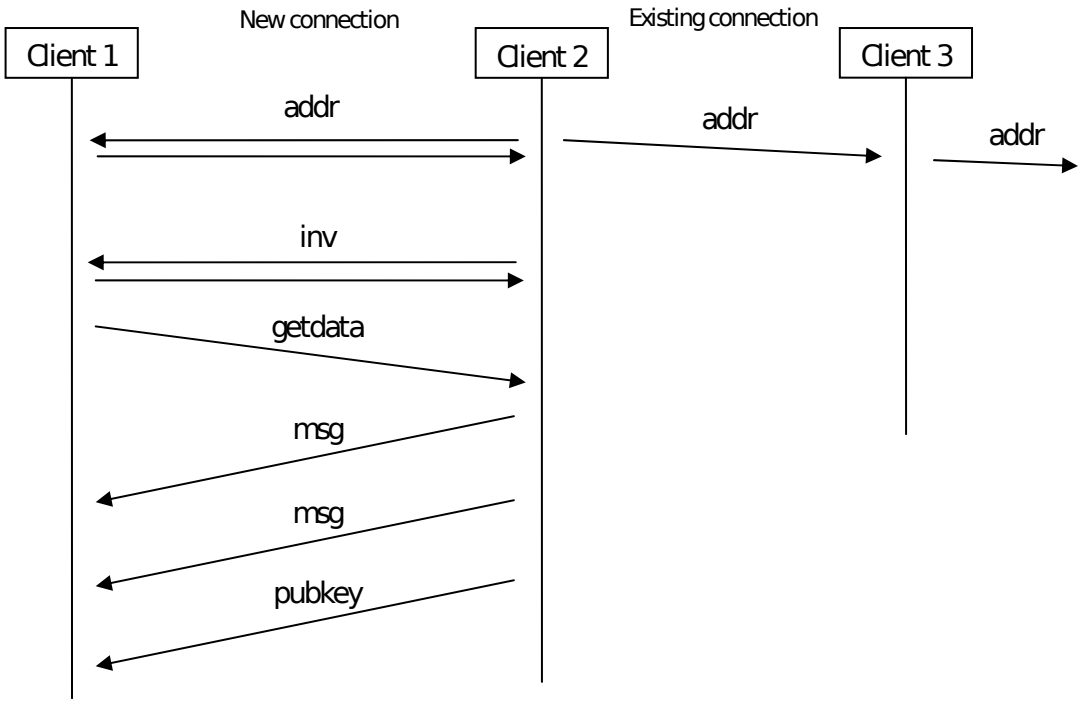

Abbildung 3.1: Bitmessage Nachrichtenaustausch [Warc]

Nach erfolgreichem Verbindungsaufbau tauschen die einzelnen Clients durch „addr“ Nachrichten ihre Adressen aus, um sich im Netzwerk bekannt zu machen. Der nächste Schritt ist der Austausch von „inv“ Nachrichten. Hierbei handelt es sich um Listen, mit denen anderen Clients mitgeteilt wird, welche Daten vorhanden sind und welche angefragt wurden. Befinden sich in dieser sog. Inventarliste Datensätze die vom Client angefordert werden, oder die er selbst benötigt, so werden „getdata“ Nachrichten verschickt mit dem entsprechenden Index der Liste. Diese Nachrichten fordern Clients auf, den gewünschten Datensatz zu übertragen. In diesem Fall handelt es sich um zwei person-to-person Nachrichten und einem Public Key.

3.1 Senden

Bevor eine Nachricht versendet werden kann, ist ein proof-of-work [Warb] zu leisten. Dieser soll Spam im Netzwerk vorbeugen. Dabei wird die Berechnung einer partielle Kollision einer SHA512-Funktion gefordert. Die Bit-Stärke der kryptographische Hashfunktion wurde bewusst größer angesetzt, um zu verhindern, dass Bitcoin Mining Hardware zur Berechnung eingesetzt wird. Abbildung 3.1 zeigt wie Nachrichten sich im Netzwerk verbreiten.

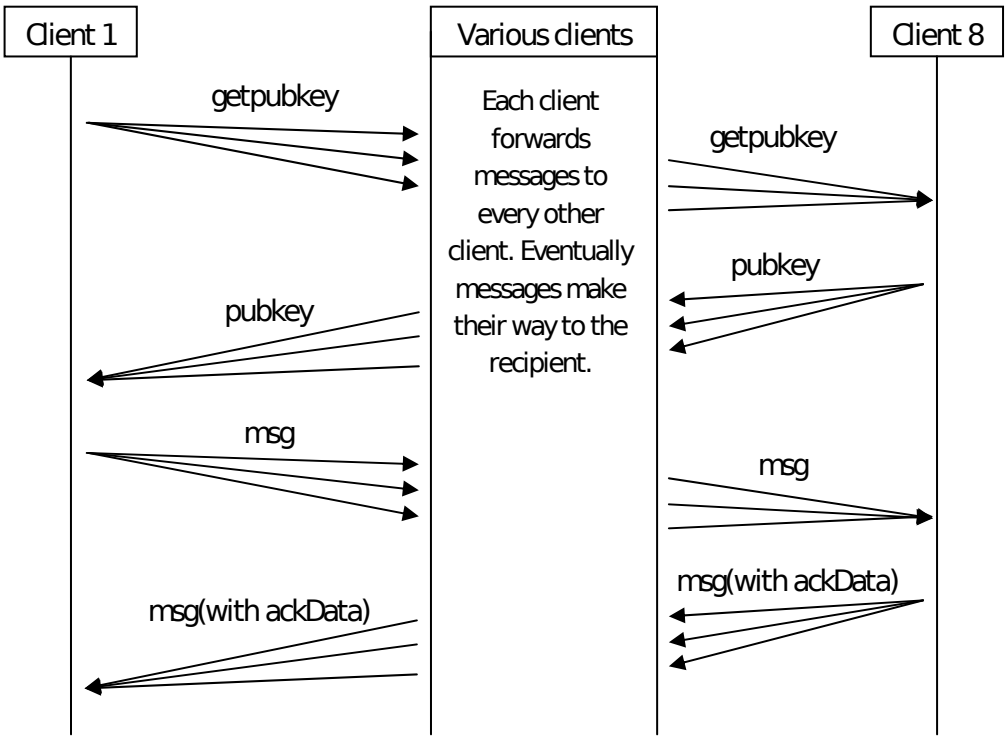

Abbildung 3.2: Nachrichtenaustausch [Warc]

Zunächst wird durch eine „getpubkey“ Nachricht der entsprechende Public-Key zur Bitmessage-Adresse angefordert. Die Anfrage wird so lange im P2P-Netz verbreitet, bis schließlich eine ein Teilnehmer den gewünschten Schlüssel bereitstellen kann. Mit diesem Schlüssel kann die Nachricht nun verschlüsselt werden. Gleichzeitig wird die Nachricht vom Absender mit seinem Signatur-Schlüsselpaar signiert, um die Herkunft sicher zu stellen. Der letzte Schritt ist nun die Verbreitung im Netzwerk in der Blockchain über den „msg“ Nachrichtentyp. Dabei bestimmt die Stream-Nummer des Empfängers die Richtung der Verbreitung. Der Sender wartet dann auf eine Bestätigung des Empfänger. Aus Gründen der Performance bleibt eine Nachricht nur eine begrenzte Zeit im Netz, bis sie von den Clients gelöscht wird. Diese Zeit ist in der Standardeinstellung auf vier Tage gesetzt, kann aber bis auf 28 Tage erhöht werden. Dabei erhöht sich die Arbeit die für den proof-of-work verrichtet werden muss entsprechend mit. Meldet sich der Empfänger in dieser Zeit nicht am Netzwerk an, so erhält er die Nachricht nicht. In einem solchen Fall, erhält der Sender keine Bestätigung und sendet die Nachricht erneut. Hierzu muss allerdings der proof-of-work erneut geleistet werden, um den Zeitstempel der Nachricht zu aktualisieren. Dies soll Replay-Angriffe verhindern, bei denen Nachrichten mitgeschnitten und 1:1 wieder ins Netzwerk eingespielt werden. Die Wartezeit für ein erneutes Senden wird automatisch exponentiell erhöht.

3.2 Empfangen

Der Bitmessage-Client versucht ständig die Nachrichten innerhalb seines Streams mit seinem Private-Key zu entschlüsseln. Schafft er dies, so wird die Nachricht dauerhaft im Posteingang des Clients angezeigt. Dazu wird sie in pyBitmessage in einer Datenbank-Datei namens messages.dat abgespeichert. Public Keys werden in der Datei key.dat lokal abgelegt. In der Regel wird dann eine Bestätigung an den Absender gesendet. Die Bestätigung besteht aus der Nachricht selbst, an diese „ackData“[Warc] gehangen wurden. Da Bestätigungen auf Nachrichten unter Umständen dazu missbraucht werden können einen Client zu deanonymisieren [Kapitel 4.2], gibt es einen sog. passiven Betriebsmodus. Ist dieser aktiviert, werden keine Bestätigungen mehr von diesem Client verschickt. Dieser Modus erzeugt allerdings das Problem, dass Nachrichten immer wieder gesendet werden, da der Sender keine Bestätigung erhält. Die Lösung hierfür ist, die Bestätigung an eine andere Nachricht anzuhängen und diese an einen unbeteiligten Client zu schicken. Bestätigt dieser seine Nachricht, so wird die Bestätigung für die eigentliche Nachricht gleichzeitig mit veröffentlicht. Abbildung 3.3 zeigt den Posteingang von pyBitmessage:

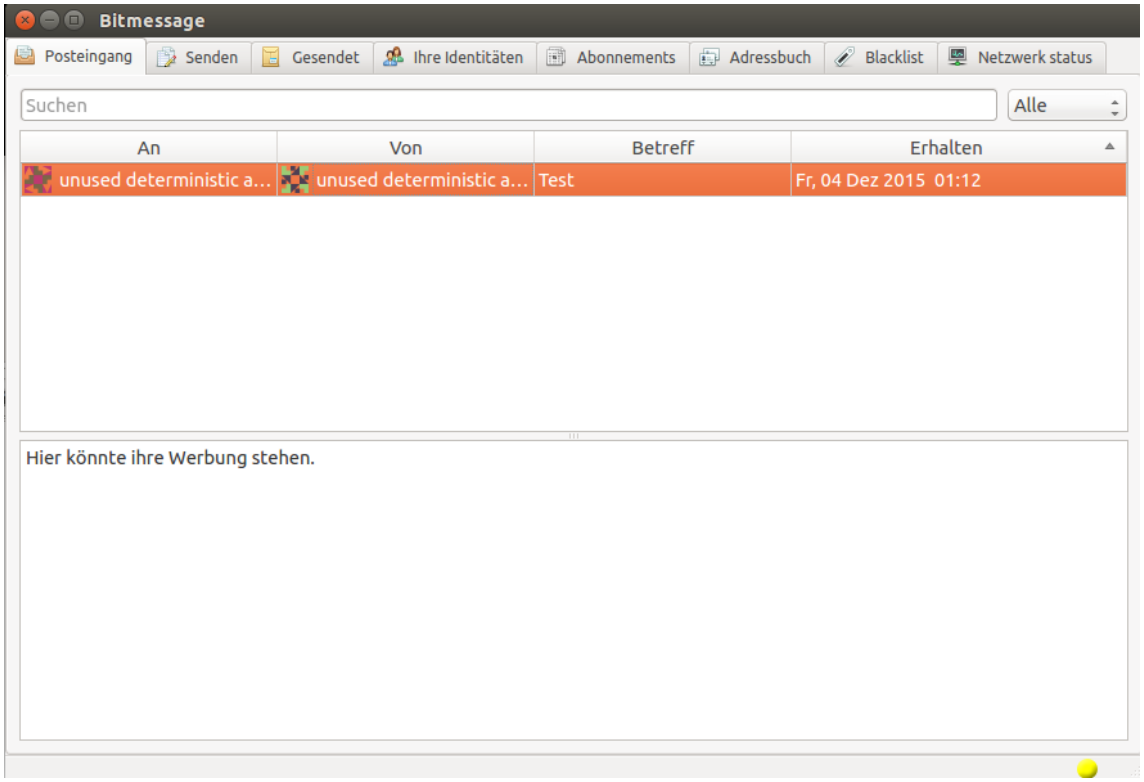

Abbildung 3.3: pyBitmessage Posteingang

3.3 Blockchain

In der Blockchain werden alle gesendeten Nachrichten gespeichert. Damit nicht jeder Client die gesamte Blockchain herunterladen muss, teilt sie sich ab einer bestimmten Größe in sog. Streams auf. Abbildung 3.4 zeigt dieses Verhalten:

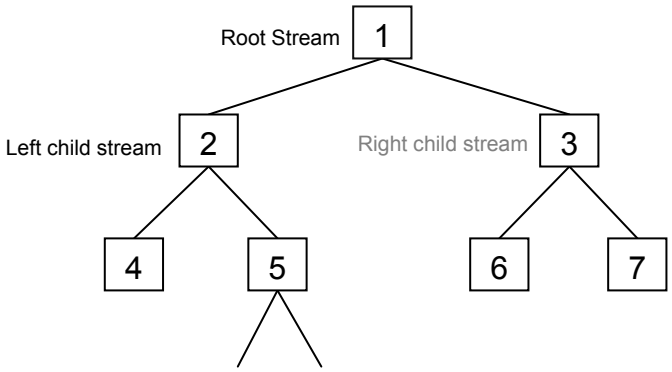

Abbildung 3.4: Streams in der Blockchain [Wara]

Die Knoten müssen dabei jeweils die Adressen von Clients in den beiden folgenden Streams, sowie dem vorherigen vorhalten. Das Prinzip entspricht einer baumartigen Datenstruktur. Bitmessage Clients mit Adressen innerhalb eines Streams müssen keine Informationen über parallele Zweige vorhalten. Erst wenn sie mit Bitmessage Adressen aus anderen Streams kommunizieren müssen, verbinden sie sich zu höheren Ebenen, um von dort zum gewünschten Stream abzusteigen. Dies ist nicht notwendig, wenn ein anderer Client die benötigte Information schon gespeichert hat. In einem solchen Fall endet die Suche vorher.

Im Gegensatz zur Bitcoin Blockchain in Abbildung 3.5, ist es an dieser Stelle gewollt, dass es Abzweigungen gibt.

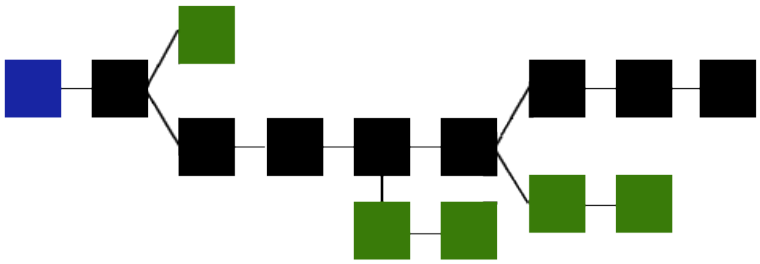

Abbildung 3.5: Bitcoin Blockchain Schema

Die Bitcoin Blockchain besitzt nur einen gültigen Pfad. Sobald Abzweigungen entstehen, werden diese so bald es geht verworfen. Bitcoins würden ansonsten doppelt existieren. Dieser Fall darf nicht eintreten. Somit existieren Nebenpfade höchstens temporär, als Resultat von Race-Conditions bei der Generierung einzelner Blöcke. Bitmessage nutzt das Aufteilen der Blockchain bewusst, um die Performance zu verbessern.

4 Anwendung

Die Entwickler haben neben einem Whitepaper auch ein technisches Datenblatt[Warc] veröffentlicht, welches alle Informationen enthält um das Bitmessage Protokoll zu implementieren. Es gibt bisher eine Referenz Implementierung namens pyBitmessage. Das py steht in diesem Fall für Phyton, da die Hauptkomponenten mit dieser Scriptsprache realisiert wurden. Der Client ist sowohl für Windows, als auch OSX und Linux verfügbar. Eine mobile Implementierung als App gibt es bisher nicht. Bitmessage wird im Darknet häufig dazu verwendet, die reguläre Nachrichtenfunktion von Foren zu ersetzen. Die Mitglieder des Forums geben hierzu ihre Bitmessage Adresse in ihrem Profil an. Möchte nun ein anderes Mitglied Kontakt aufnehmen, klickt er den entsprechenden Link im Profil des Empfängers an und kann so über das Forum eine Bitmessage versenden. Dabei wird die Adresse des Mitglieds nicht angezeigt.

4.1 Client

Im folgenden wird die Linux Version von pyBitmessage näher betrachtet. Nachdem man den Client heruntergeladen und gestartet hat, bietet sich die in Abbildung 4.1 gezeigte Oberfläche:

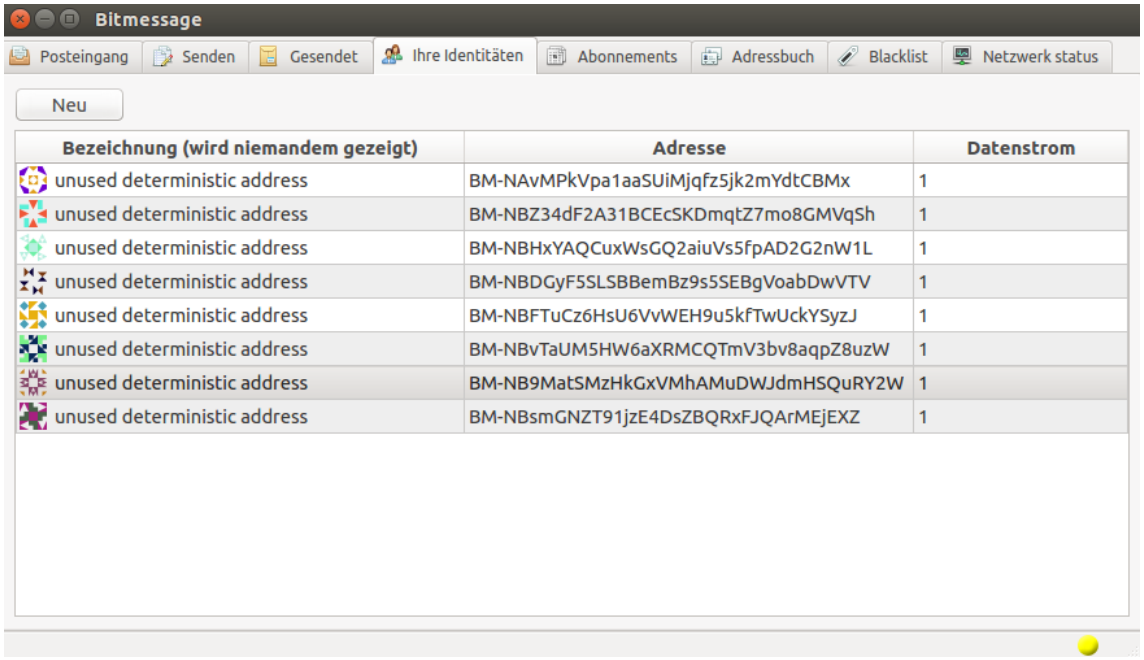

Abbildung 4.1: pyBitmessage Client unter Linux

Der Client ist sehr simpel aufgebaut. Der Posteingang zeigt die Nachrichten der Blockchain die sich mit dem eigenen privaten Schlüssel entschlüsseln lassen. Hinter dem Menüpunkt Senden (Abbildung 4.2) steht die klassische Mailoberfläche, mit Angabe des Empfängers, der Nachricht und dem Nachrichtentyp (Broadcast oder person-to-person).

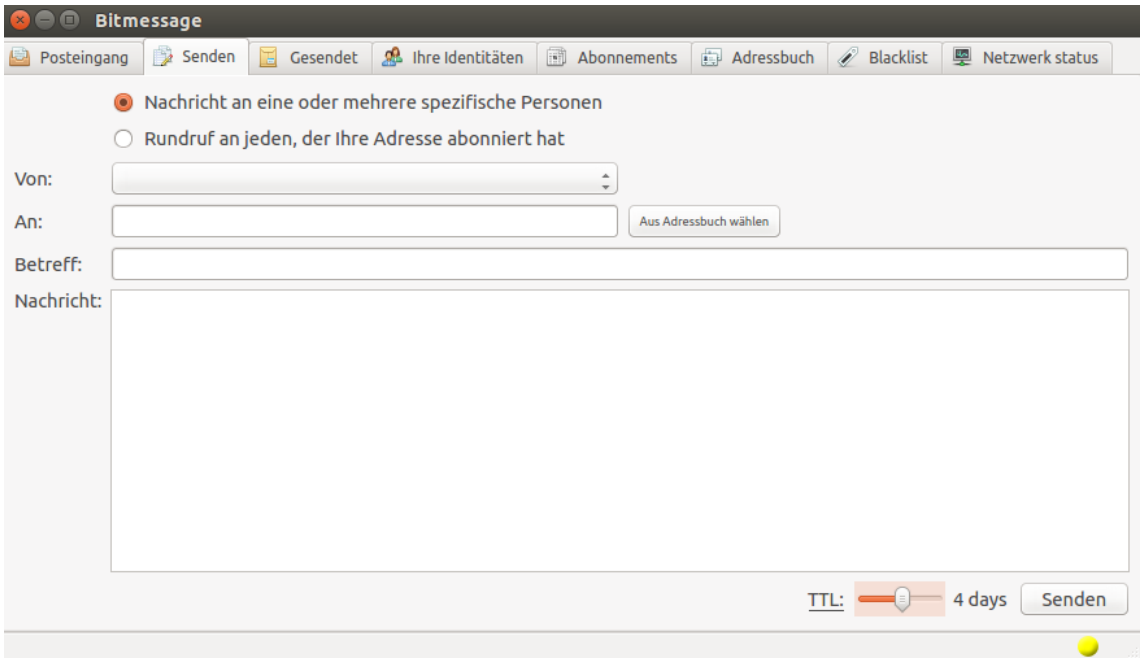

Abbildung 4.2: Nachricht senden

Die Time-to-Live einer Nachricht wird im unteren Bereich eingestellt und nimmt Einfluss auf die Berechnung des proof-of-work Wertes. Gesendete Nachrichten werden archiviert. Das Menü Ihre Identitäten, enthält alle Bitmessage Adressen die bereits mit diesem Client generiert wurden. Adressen können nicht verfallen, sie werden bei längerer Inaktivität lediglich als „unused“ markiert. Werden diese anschließend wieder genutzt, verschwindet auch die Markierung wieder. Aktive Broadcast Abonnements lassen sich ebenfalls über den Client verwalten. In einem eigenen Adressbuch werden die Bitmessage Adressen von Nutzern gespeichert mit denen eine Kommunikation statt fand. In der Blacklist können Bitmessage Adressen eingetragen werden, von denen Nachrichten ignoriert werden sollen. Sie kann wahlweise auch als Whitelist verwendet werden die dann entsprechend umgekehrt funktioniert. Das Icon in der unteren rechten Ecke, sowie der entsprechende Menüpunkt zeigen den Status der Netzwerkverbindung. Gelb ist die normale Betriebsfarbe, wenn keine Port-Weiterleitung für zusätzliche Verbindungen eingerichtet wurde. Mit diesem Status kann der Client lediglich 8 gleichzeitige Verbindungen aufbauen.

4.2 Sicherheit

Gelingt es einem Angreifer Zugang zu einem privaten Schlüssel zu erhalten, so kann er nachträglich alle bisher mit der zugehörigen Bitmessage-Adresse empfangenen Nachrichten entschlüsseln. Selbst wenn die Nachrichten lokal gelöscht wurden, muss davon ausgegangen werden, dass ein Angreifer sie während der Übertragung aufgezeichnet hat. Innerhalb der Nachrichten können zudem Signaturen vorhanden sein, die einzelne Nutzer deanonymisieren könnten. Bitmessage unterstützt keine Off-the-Record-Nachrichten, deren Existenz die Beteiligten nachträglich abstreiten können. Diese Schwäche teilt Bitmessage allerdings mit S/MIME, PGP und anderen Anwendungen asymmetrischer Verschlüsselung. Wenn die lokale Nachrichten-Datei regelmäßig gelöscht wird, erhöht dies die Sicherheit.

Abstreitbarkeit kann in vielen Fällen von Nutzern gewollt sein. Sie kann dadurch erreicht werden, dass der Nutzer den gesamten Adresssatz öffentlich macht. Sollte von mindestens einer anderen Person der Adressblock in die Schlüsseldatei keys.dat aufgenommen werden, so kann der ursprüngliche Verfasser der Nachrichten nicht eindeutig ermittelt werden, da theoretisch jeder die Nachricht hätte verfassen können. Durch diesen Schritt werden jedoch auch alle existierenden Nachrichten für jeden einsehbar.

Laut Jonathan Warren soll ein Bitmessage Client verwundbar für einen Timing Angriff sein. Um Nutzer A zu identifizieren, wird eine sehr hohe Zahl an Nachrichten von Nutzer B an Knotenpunkt X gesendet. Angenommen es dauert 1 Sekunde eine nicht an Nutzer A gerichtete Nachricht weiterzuleiten und 2 Sekunden, damit Nutzer A seine Nachricht entschlüsselt und anschaut. Bei ca. 100 gesendeten Nachrichten würde es 100 Sekunden dauern, falls A den Knotenpunkt nicht nutzt, jedoch doppelt so lange, falls A den Knotenpunkt nutzt. Hierdurch könnte A identifiziert werden. [Wara] Ein lokaler Angreifer kann zudem feststellen, dass eine Bestätigung auf eine Nachricht von einem Client aus dem Netzwerk stammt. Dies setzt allerdings zusätzliches Wissen voraus. Dieses Risiko lässt sich durch den passiven Modus ausschließen, indem die Bestätigungen als Teil einer neuen Nachricht indirekt von einem anderen Client veröffentlicht werden. Bisher gab es keine protokollierten Angriffe auf das Bitmessage Netzwerk, lediglich theoretische Konzepte die auf Netzwerkanalysen beruhen, oder die Kontrolle über einen Großteil der Knoten voraussetzen.

Nachrichten die erfolgreich aus der Blockchain empfangen wurden, werden im lokalen Posteingang des Bitmessage-Clients gespeichert. Im Falle von pyBitmessage kann der Client nicht zusätzlich mit einer Authentifizierung gesichert werden. Somit kann ein Angreifer nach einer physikalischen Übername des Systems (Evil-Maid-Angriff) in vollem Umfang auf den Posteingang zugreifen. Für zusätzliche Sicherheit, können die Nachrichten nochmals manuell per PGP verschlüsselt werden. Der externe Public Key des entsprechenden Schlüsselpaares würde beispielsweise entsprechend mit der Bitmessage-Adresse in einem virtuellen Profil hinterlegt werden.

4.3 Mail Gateway

Das Schweizer Projekt Bitmessage Mail Gateway [V] bietet einen Service zur Verbindung von Bitmessage und regulärem E-Mail Protokoll. Nach einer Registrierung erhält man eine reguläre user@bitmessage.ch E-Mail Adresse. Im Benutzeraccount lässt sich im Anschluss eine Bitmessage Adresse hinterlegen. Der Nachrichtenaustausch funktioniert daraufhin in beide Richtungen. Verwendet man die Bitmessage Adresse als @-Präfix, so wird die Nachricht über das Bitmessage Netzwerk verschickt. Somit wird auch der proof-of-work vom Server geleistet. Der Dienst erkennt dabei, ob einer Bitmessage Adresse schon ein bitmessage.ch Account zugeordnet ist. Ist das der Fall, wird die E-Mail sofort an das E-Mail Postfach des Empfängers weitergeleitet, ohne dabei das Bitmessage Netzwerk zu passieren. Somit entfällt die Berechnung des proof-of-work.

Die Kommunikation findet via TLS statt. Auf die Postfächer kann per IMAP, POP3, sowie SMTP zugegriffen werden. Der Zugriff auf die Website ist auch aus dem Tor Netzwerk möglich. Mit Hilfe dieses Dienstes, lassen sich die Funktionen von regulären Mailclients in Kombination mit Bitmessage nutzen. Hierunter fallen zum Beispiel Wiedervorlage, Gruppieren, Sortieren, Filter, Autoresponder, sowie Spamfilter. Die Finanzierung des Projekts läuft über Bitcoin oder Dogecoin Spenden. Ein weiterer E-Mail Gateway ist Mailchuck [Mai]. Dieser funktioniert auf ähnliche Art und Weise wie Bitmessage Mail Gateway. Das Empfangen von Nachrichten ist hier grundsätzlich kostenlos. Sollen auch Nachrichten versendet werden, so muss der Account gegen Bezahlung in einen Premium-Status versetzt werden.

5 Zusammenfassung

Das Bitmessage Protokoll wurde bisher noch nicht von einer unabhänigen Stelle analysiert. Besonders interessant wäre hier eine Sicherheitsanalyse, um vor allem die kryptographische Implementierung zu testen. Ungeachtet dessen ist Bitmessage bereits weit verbreitet im Darknet und ähnlichen Netzwerken. Es ist kein zentraler Server von Nöten, dessen X.509 Zertifikaten man vertrauen müsste. Windows-Betriebssysteme oder Browser wie Firefox stufen über 1000 X.509 Zertifikate als vertrauenswürdig ein. Eine einzelne Firma, die Teil dieser Liste ist und fragwürdige Absichten verfolgt, genügt um man-in-the-middle-Angriffe zu ermöglichen[**X.509**]. National Security Letter könnten jede Firma mit Sitz in den USA zu einer Herausgabe von Daten zwingen, ohne das in der Öffentlichkeit Hinweise darauf auftreten [Hol]. Im Fall von Dell wurden sogar ab Werk eigene Root Zertifikate installiert[Böc]. Gleichzeitig wurde hier der private Schlüssel mit im Zertifikatsspeicher abgelegt. Aus diesem lassen sich Zertifikate problemlos wieder extrahieren. So ist es einem Angreifer möglich, mit Hilfe des privaten Schlüssels, HTTPS Verbindungen aufzubauen, die einem Benutzer als sicher erscheinen. Die Daten die über eine solche Verbindung übertragen werden, sind für Angreifer les- sowie manipulierbar. Metadaten über Kommunikation, die beispielsweise von der NSA an vielen Stellen in großem Stil gesammelt und ausgewertet werden, sollten genauso gut geschützt werden wie der Inhalt von Nachrichten selbst. Zeitstempel von Nachrichten, sowie Absender und Empfänger sind für sich bereits wertvolle Informationen, um Kommunikation rekonstruieren zu können. Im Hinblick auf QR-Codes oder NFC-Tags für Smartphones, ist eine App für mobile Betriebssysteme eine sinnvolle Weiterentwicklung. Eine erste Implementierung gibt es bereits in Form von Bitseal [Coe]. Diese APP unterstützt QR-Codes und gestaltet die Handhabung der Adressen somit einfacher. Die Prozessoren in modernen mobilen Geräten sind mitlerweile so leistungsfähig, dass sie auch komplexere Verschlüsselungsverfahren in moderater Zeit verarbeiten können. Die Sicherheit der Nachrichten lässt sich über eine manuelle externe PGP-Verschlüsselung noch weiter ausbauen. Ein entsprechender Verbesserungsvorschlag wurde bereits eingereicht, bei dem es um die Verschlüsselung des lokalen Nachrichtenspeichers sowie der Schlüsselpaare geht.

Literaturverzeichnis

[Bit] Bitmessage. *PyBitmessage*. 01.12.2015. URL: https://github.com/Bitmessage/PyBitmessage.

[Böc] Hanno Böck. *HTTPS-Verschlüsselung von Dell-Nutzern gefährdet*. 01.12.2015. URL: http://www.golem.de/news/gefaehrliches-root-zertifikat-https-verschluesselung-von-dell-nutzern-gefaehrdet-1511-117585.html.

[Coe] Jonathan Coe. *Bitseal*. 01.12.2015. URL: https://github.com/JonathanCoe/bitseal.

[Cor] Creative Commons Corporation. *Attribution 3.0 Unported*. 15.02.2016. URL: http://creativecommons.org/licenses/by/3.0/legalcode.

[Hol] Martin Holland. *US-Überwachung: Erster National Security Letter vollständig öffentlich*. 01.12.2015. URL: http://www.heise.de/newsticker/meldung/US-Ueberwachung-Erster-National-Security-Letter-vollstaendig-oeffentlich-3028076.html.

[Mai] Mailchuck. *Mailchuck: anonymous email*. 01.12.2015. URL: https://mailchuck.com.

[Man] Manzanamedanica. *Bitmessage Image*. 01.12.2015. URL: http://manzanamecanica.org/2013/07/mensajes_y_foros_seguros_privados_y_a_prueba_de_gobiernos_bitmessage.html.

[MIT] MIT. *The MIT License (MIT)*. 15.02.2016. URL: https://opensource.org/licenses/mit-license.php.

[PEc] J.Burns P.Eckersley. *An Observatory for the SSLiverse*. 01.12.2015. URL: https://www.eff.org/files/DefconSSLiverse.pdf.

[Proa] Bitcoin Project. *Bitcoin*. 01.12.2015. URL: https://bitcoin.org/de/.

[Prob] Bitcoin Project. *Public key to bitmessage address*. 01.12.2015. URL: https://bitmessage.org/wiki/Public_key_to_bitmessage_address.

[Proc] The Tor Project. *Anonymity Online*. 01.12.2015. URL: https://www.torproject.org/index.html.en.

[Prod] The Tor Project. *Tor Browser*. 01.12.2015. URL: https://www.torproject.org/projects/torbrowser.html.en.

[V] o. V. *Bitmessage Mail Gateway*. 01.12.2015. URL: https://bitmessage.ch.

[Wara] Jonathan Warren. *Bitmessage: A Peer-to-Peer Message Authentication and Delivery System*. 01.12.2015. URL: https://bitmessage.org/bitmessage.pdf.

[Warb] Jonathan Warren. *Proof of work*. 01.12.2015. URL: https://bitmessage.org/wiki/Proof_of_work.

[Warc] Jonathan Warren. *Proposed Bitmessage Protocol Technical Paper*. 01.12.2015. URL: https://bitmessage.org/Bitmessage%20Technical%20Paper.pdf.

[Ward] Jonathan Warren. *Protocol specifications*. 01.12.2015. URL: https://bitmessage.org/wiki/Protocol_specifications.