Seminararbeit

Time-Lock Encryption mit Bitcoin

31. August 2018

Eingereicht von:
Niklas Kühl
inf102861@fh-wedel.de

Betreuer:
Prof. Dr. Gerd Beuster
gb@fh-wedel.de

Inhaltsverzeichnis

1 Einleitung

Die folgende Ausarbeitung beschäftigt sich mit der zeitabhängigen Verschlüsselung von Daten, der sogenannten Time-Lock Encryption. Damit ist es möglich „Nachrichten in die Zukunft zu schicken“. Das Verfahren sieht vor, dass einmal verschlüsselte Daten bis zu einem bestimmten Zeitpunkt nicht entschlüsselt werden können, danach die Entschlüsselung aber trivial ist.

In Kapitel 2 werden die Grundlagen für die Time-Lock Encryption beschrieben. Zunächst wird die Time-Lock Encryption genau definiert. Zur Umsetzung der Time-Lock Encryption wird die Witness Encryption verwendet, welche im Anschluss definiert wird. Da hierfür öffentliche Berechnungen genutzt werden sollen, werden einige Eigenschaften derartiger Berechnungen kurz vorgestellt.

In Kapitel 3 wird die Witness Encryption detailliert erklärt. Als Erstes werden die dafür nötigen Objekte vorgestellt. Dies umfasst das Konzept der multilinearen Abbildungen sowie das NP-vollständige Problem Exact Cover. Auf Basis dieser Definitionen wird dann das Verfahren für Ver- und Entschlüsselung erläutert.

In Kapitel 4 wird dann das vorgestellte Verschlüsselungsschema mit Bitcoin praktisch umgesetzt. Zunächst werden die Grundlagen von Bitcoin erklärt. Anschließend wird die δ-Funktion definiert, welche für die Sicherheit des Verfahrens notwendig ist. Weiterhin wird die praktische Umsetzung des Schemas diskutiert.

2 Grundlagen

Im folgenden Kapitel wird das grundlegende Vorgehen erklärt, um die Time-Lock Encryption zu konstruieren. Hierzu gehören die Definitionen der Time-Lock Encryption und der Witness Encryption.

2.1 Time-Lock Encryption

Time-Lock Encryption [Jag15] ist ein Verschlüsselungsschema, bei dem die Verschlüsselung unter Angabe eines bestimmten in der Zukunft liegenden Zeitpunktes ausgeführt wird. Bis zu diesem Zeitpunkt soll die Entschlüsselung unmöglich sein. Nach dem Zeitpunkt hingegen soll die Entschlüsselung trivial sein, also ohne aufwendige Berechnungen und ohne Angabe eines Schlüssels erfolgen können.

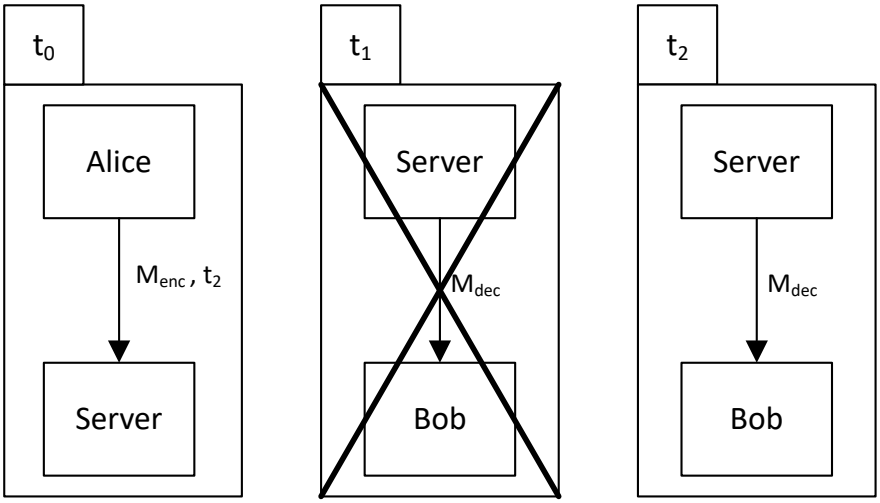

Abbildung 2.1: Graphische Darstellung Time-Lock Encryption

Es ist zu beachten, dass eine Nachricht nicht für eine bestimmte Zeitspanne, sondern bis zu einem bestimmten Zeitpunkt verschlüsselt wird. Daher ist es bei der Entschlüsselung irrelevant, zu welchem Zeitpunkt die Nachricht verschlüsselt wurde. Nur der Zeitpunkt der Entschlüsselung ist wichtig.

Wenn eine vertrauenswürdige dritte Partei existiert, ist dieser Vorgang trivial. Dann kann die Nachricht mit einem sicheren symmetrischen Verfahren verschlüsselt werden und der Schlüssel zum gewünschten Zeitpunkt von dieser Partei veröffentlicht werden. Ein Nachteil dieser Variante ist, dass es sehr schwer ist die Vertrauenswürdigkeit einer dritten Partei sicherzustellen. Außerdem muss die Existenz der dritten Partei gewährleistet sein, bis der Schlüssel veröffentlicht wird. Daher soll im Folgenden ein von Drittinstanzen unabhängiges Verschlüsselungsschema vorgestellt werden.

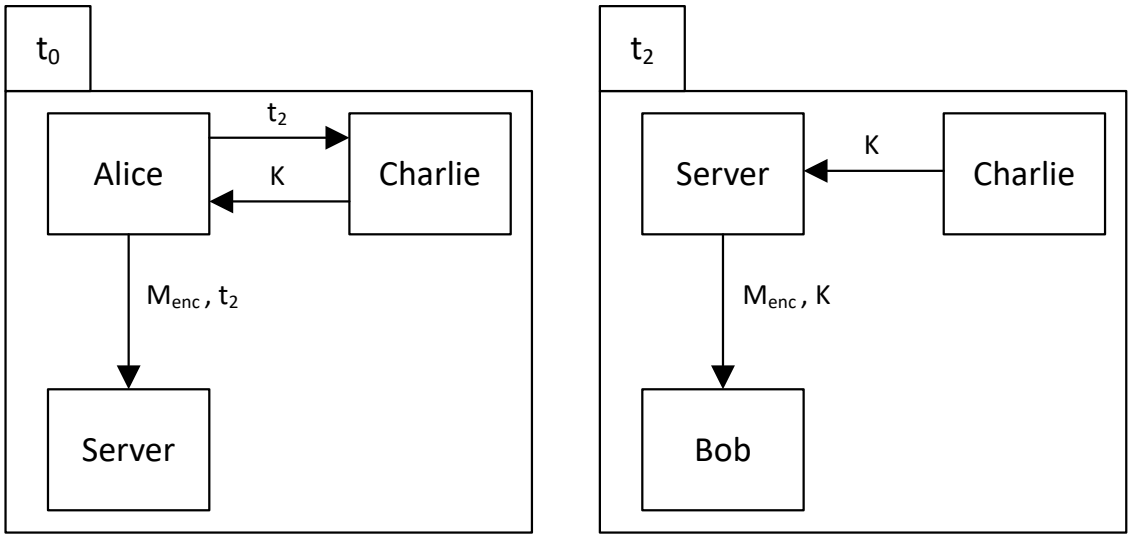

Abbildung 2.2: Time-Lock Encryption mit dritter Partei

2.2 Witness Encryption

Witness Encryption [GGSW13] ist ein asymmetrisches Verschlüsselungsschema, welches Problemstellungen zur Verschlüsselung und Problemlösungen zur Entschlüsselung nutzt. Ein bestimmtes Witness Encryption Schema wird dabei für eine bestimmte Sprache aus NP definiert, wodurch die möglichen Problemstellungen festgelegt werden.

Die Verschlüsselung einer Nachricht erfolgt mit einer bestimmten Instanz des zuvor gewählten Problems. Die Entschlüsselung soll nur möglich sein, wenn eine gültige Lösung zu dieser Probleminstanz bekannt ist. Eine solche gültige Lösung wird auch als *witness* bezeichnet. Wenn keine derartige Lösung existiert, ist die Entschlüsselung nicht möglich.

2.3 Time-Lock Encryption mit Witness Encryption

Zur Realisierung einer Time-Lock Encryption eignet sich die Witness Encryption als Grundlage, denn eine mit einer konkreten Probleminstanz verschlüsselte Nachricht kann von jedem entschlüsselt werden, sofern eine gültige Lösung vorliegt. Das verwendete Problem muss dabei folgende Eigenschaften erfüllen:

1. Für alle Probleminstanzen existiert eine gültige Lösung
2. Für jede Probleminstanz kann eindeutig bestimmt werden, wie lange die Berechnung der Lösung dauert

Da Punkt 1 von vielen Problemen bereits erfüllt wird, ist es relativ einfach, entsprechende Probleme zu finden.

Punkt 2 ist hingegen sehr viel schwieriger zu realisieren. Da nicht bekannt ist, welche Ressourcen zur Berechnung benutzt werden, kann auch keine Abschätzung zur Rechenzeit gegeben werden. Daher muss die Definition aus 2.1 etwas abgeschwächt werden: Die Entschlüsselung einer Nachricht vor dem Zeitpunkt ist nun nicht mehr komplett unmöglich, sondern nur bestimmte Angreifer. Dazu wird eine Obergrenze für die Rechenleistung eines Angreifers definiert, woraus sich automatisch eine minimale Rechenzeit zur Lösung des Problems ergibt. Wird die Obergrenze sinnvoll gewählt, so dass es praktisch unmöglich ist, dass ein Angreifer eine derart hohe Rechenleistung besitzt, ist das Verfahren weiterhin sicher.

2.4 Iterative öffentliche Berechnungen

Als mögliches Problem zur Verwendung in der Time-Lock Encryption eignet sich eine iterative öffentliche Berechnung. Eine solche Berechnung findet in der Regel in einem verteiltem Netzwerk aus vielen einzelnen Rechnern statt, zu dem jeder beitragen kann. Wenn das Netzwerk ausreichend groß ist, verfügt es über eine entsprechend große Rechenleistung, welche sich nicht schlagartig ändert. Somit kann abgeschätzt werden, wie lange die Berechnung einer bestimmten Probleminstanz durch das Netzwerk dauert.

Die Rechenleistung eines entsprechend großen Netzwerks übersteigt die mögliche Rechenleistung eines Angreifers sehr stark. Somit entsteht eine Obergrenze für die Rechenleistung eines Angreifers und es ist keinem Angreifer möglich, die Berechnung schneller als das Netzwerk auszuführen.

Eine weitere Angriffsmöglichkeit besteht darin, dass ein Angreifer seine Ressourcen zum Netzwerk hinzufügt, um die Berechnung zu beschleunigen. Diese Beschleunigung würde allerdings minimal ausfallen, da die Rechenleistung des Angreifers sehr viel geringer als die des Netzwerks ist.

3 Witness Encryption im Detail

Im folgenden Kapitel wird beschrieben, wie die Witness Encryption funktioniert, welche die Grundlage für die Time-Lock Encryption bildet. Es wird zunächst das mathematische Konzept der multilinearen Abbildung eingeführt. Anschließend wird anhand des NP-vollständigen Problems Exact Cover das Verschlüsselungsschema konstruiert.

3.1 Multilineare Abbildungen

3.1.1 Bilineare Abbildungen

Eine bilineare Abbildung [Haz94] hat die Form $e : V \times W \mapsto X$, wobei V, W und X Vektorräume sind. Eine bilineare Abbildung muss außerdem die Rechenregeln der Addition und der skalaren Multiplikation erhalten:

$e(v + v', w) = e(v, w) + e(v', w)$

$e(v', w + w') = e(v, w) + e(v, w')$

$e(av, w) = ae(v, w)$

$e(v, wb) = e(v, w)b$

Ein Beispiel für eine bilineare Abbildung ist die Matrixmultiplikation.

3.1.2 Bilineare Abbildungen in der Kryptographie

In der Kryptografie werden bilineare Abbildung auf zyklischen Gruppen definiert. Daher hat sie die Form $e : G_1 \times G_2 \mapsto G_t$, wobei ein der Praxis meist $G_1 = G_2$

Weiterhin muss für eine bilineare Abbildung gelten: $e(a^x, b^y) = e(a, b)^{xy}$.

Solche bilinearen Abbildungen erlauben die Konstruktion von kryptografischen Protokollen mit 3 Parteien. Ein mögliches Anwendungsbeispiel ist die *Identity-based Encryption* zum sicheren Nachrichtenaustausch, bei der es zusätzlich zu Sender und Empfänger einen vertrauenswürdigen zentralen Dienst zur Schlüsselgenerierung gibt.

Bei der *Identity-based Encryption* dient eine ID, wie etwa die Empfänger-Mail-Adresse als öffentlicher Schlüssel. Somit ist zum Senden einer Nachricht keine Aktion des Empfängers erforderlich, wie etwa die Bereitstellung des öffentlichen Schlüssels.

Das Protokoll arbeitet folgendermaßen:

1. Alice fragt einmalig den MPK (Master Public Key) vom Dienst zur Schlüsselgenerierung ab
2. Alice kann aus dem MPK und Bobs E-Mail-Adresse dessen öffentlichen Schlüssel berechnen
3. Alice sendet Bob eine verschlüsselte Nachricht
4. Bob muss einmalig seinen privaten Schlüssel vom Dienst generieren lassen und kann dann alle an ihn gesendeten Nachrichten entschlüsseln

Ein Nachteil dieses Verfahrens ist natürlich, dass der zentrale Dienst alle privaten Schlüssel kennt. Wenn allerdings alle Teilnehmer von Beginn an bekannt sind, können einmal alle privaten Schlüssel generiert werden und den Teilnehmern zugänglich gemacht werden. Dann kann der zentrale Dienst gelöscht werden und existiert danach nur noch „mathematisch“.

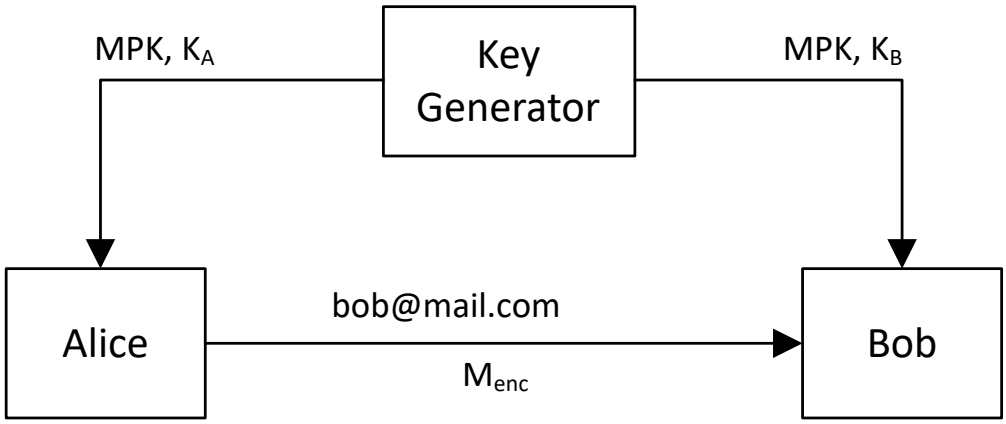

Abbildung 3.1: Identity-based Encryption

3.1.3 Multilineare Abbildungen in der Kryptographie

Eine multilineare Abbildung ist eine Erweiterung der bilinearen Abbildung auf n Dimensionen. Sie hat die Form $e : G_1 \times G_2 \times ... \times G_n \mapsto G_t$. Für die multilineare Abbildung muss gelten: $e(a_1^{x_1}, a_2^{x_2}, ..., a_n^{x_n}) = e(a_1, a_2, ..., a_n)^{\prod_{i=1}^{n} x_i}$.

Solche multilinearen Abbildungen erlauben die Konstruktion von Protokollen mit $n + 1$ Parteien.

3.2 Exact Cover

3.2.1 Definition

Das Problem der exakten Überdeckung (Exact Cover) ist eines der klassischen NP-vollständigen Probleme von Richard Karp [Kar72]. Es ist folgendermaßen definiert:

Gegeben:

- Eine Grundmenge $X = \{x_1, x_2, ..., x_n\}$
- Eine Menge $S = \{T_1, T_2, ..., T_m\}$; jedes T_i ist dabei eine Teilmenge von X

Gesucht:

Eine Menge $S^* \subseteq S$ für die gilt:

- Die Vereinigungsmenge der Elemente von S^* ergibt X
- Die Schnittmenge von je zwei Elementen aus S^* ist leer

Somit ist S^* eine Partition von X.

Für eine gegebene Probleminstanz von Exact Cover kann es daher mehrere gültige Lösungen geben. Es ist allerdings nicht garantiert, dass überhaupt eine Lösung existiert.

Bildlich gesprochen ist S^* eine Menge von Puzzleteilen, aus denen wieder die Menge X gebildet werden soll. Die Puzzleteile dürfen sich dabei nicht überlappen.

3.2.2 Beispiel

Es soll eine FH-Übung stattfinden. Die Bearbeitung soll ausschließlich in Gruppen erfolgen. Es sind allerdings nur bestimmte Gruppen möglich (z. B. aufgrund unterschiedlicher Studiengänge). Jeder Übungsteilnehmer soll einer Gruppe zugeteilt werden.

X ist somit die Menge der Studenten, die an der Übung teilnehmen

S ist dann die Menge aller möglichen Gruppen.

S^* ist eine gültige Zuordnung aller Studenten zu einer Gruppe.

Weitere Anwendungen von Exact Cover:

- Sudoku
- N-Damen-Problem

3.3 Definition der Witness Encryption

Basierend auf Exact Cover wird nun die Witness Encryption definiert. Für jedes Element x_i der Menge X wird dabei ein Zufallswert generiert. Für jedes Element T_i der Menge S wird ein Teilschlüssel C_i definiert.

3.3.1 Definition der genutzten multilinearen Abbildung

Die hier genutzte multilineare Abbildung ist eine Funktion, welche auf den Gruppen $G_1, G_2, ..., G_n$ operiert. Für jedes Element von $X = \{x_1, x_2, ..., x_n\}$ gibt es somit also eine Gruppe. Alle Gruppen müssen die gleiche Ordnung p besitzen, wobei p eine Primzahl sein muss. Für jede Gruppe G_i ist ihr erzeugendes Element gen_i bekannt.

Diese Gruppen können durch die Abbildung $e_{ij} : G_i \times G_j \mapsto G_{i+j}$ miteinander kombiniert werden: $e_{ij}(gen_i, gen_j) = gen_{i+j}$, wobei $i + j \leq n$. Es werden also immer 2 Elemente aus Gruppen mit niedrigem Index auf ein Element einer Gruppe mit höherem Index abgebildet. Diese Definition von e erfüllt auch die in 3.1.2 geforderte Bedingung für bilineare Abbildungen.

Nun werden die Funktionen e_{ij} zu einer Funktion $e : G_{i_1} \times G_{i_2} \times ... \times G_{i_t} \mapsto G_{i_x}$ zusammengefasst, wobei $x = i_1 + i_2 + ... + i_t$ und $x \leq n$. Dies ist dann die multilineare Abbildung, welche bei dem Verfahren genutzt wird.

Die multilineare Abbildung erlaubt effektiv eine Multiplikation der Exponenten von Gruppenelementen. Eine Division ist nicht möglich.

3.3.2 Verschlüsselung

Gegeben:

- Eine Instanz von Exact Cover[1]: (X, S)
- Die multilineare Abbildung e
- Klartext

Berechnung:

- Ciphertext
- C: Eine Menge, welches für jedes Element aus S einen entsprechenden Teilschlüssel enthält

Folgender Algorithmus wird zur Verschlüsselung benutzt. Es wird davon ausgegangen, dass der Klartext als Gruppenelement codiert werden kann:

1. Bestimme die Gruppen $G_1, G_2, ..., G_n$ mit der Ordnung p für die Elemente von X

[1] Da sich X einfach aus S durch Vereinigung bestimmen lässt, kann die Instanz auch durch $(|X|, S)$ dargestellt werden

2. $randomVals =$ Wähle n zufällige Zahlen aus dem Bereich 1 bis $p-1$
3. $rp = \prod_{i=1}^{n} randomVals$
4. $secret = (gen_n)^{rp}$
5. $ciphertext = plaintext \cdot secret$
6. for i in 1 .. $|S|$
   - $randomVals'_i =$ Wähle für jedes Element aus T_i das entsprechende Element aus $randomVals$ aus
   - $rp'_i = \prod_{j=1}^{|T_i|} randomVals'_i$
   - $C_i = (gen_{|T_i|})^{rp'_i}$
7. return $(ciphertext, C_1..C_m)$

Die Idee bei der Verschlüsselung ist folgende: Jedem Element aus X wird ein Zufallswert zugewiesen. Der Klartext wird mit dem Produkt aus allen Zufallswerten verschlüsselt. Für die Menge S wird für jedes Element ein Produkt von Zufallswerten entsprechend der enthaltenen Elemente gebildet, wodurch die Menge C entsteht.

Wenn nun eine Lösung der Exact-Cover-Instanz bekannt ist, können die der Lösung entsprechenden Elemente aus C ausgewählt werden und mit diesen kann dann das Produkt aus allen Zufallswerten, also der Schlüssel, rekonstruiert werden.

3.3.3 Entschlüsselung

Folgender Algorithmus wird zur Entschlüsselung benutzt:

1. $C' =$ Wähle aus C die Elemente aus, die der Lösung der Exact-Cover-Instanz entsprechen
2. $e(C'_1, ..., C'_m)$
   $= e(C_a, ..., C_z)$
   $= e((gen_{|T_a|})^{rp'_a}, ..., (gen_{|T_z|})^{rp'_z})$
   $= e(gen_{|T_a|}, ..., gen_{|T_z|})^{\prod_{i=a}^{z} rp'}$ (laut 3.1.3)
   $= (gen_{|T_a|}, ..., gen_{|T_z|})^{rp}$ (sofern Exact-Cover-Lösung korrekt)
   $= (gen_n)^{rp}$ (laut 3.3.1, sofern Exact-Cover-Lösung korrekt)
   $= secret$
3. $plaintext = ciphertext / secret$

Laut der Eigenschaft der multilinearen Abbildung können die Zufallsprodukte rp'_i als Exponent der Abbildung geschrieben werden. Wenn die Exact-Cover-Lösung korrekt ist, ergibt das Produkt der gewählten Zufallswerte genau das Produkt aller Zufallswerte rp. Dies ist der Fall, da die Zufallswerte mit den Elementen von X korrelieren.

Für die Generatoren gen_i wird die Definition der genutzten multilinearen Abbildung genutzt. Es können beliebig viele Generatoren mit niedrigem Index zu einem Generator

mit höherem Index verknüpft werden. Die Indizes der Generatoren für C wurden so gewählt, dass sie $|T_i|$ entsprechen. Die Generatoren werden also genutzt, um die Anzahl der Elemente der Lösung zu zählen. Wenn diese Anzahl gleich $|X|$ ist, ist das Ergebnis von $e(gen_{|T_a|}, ..., gen_{|T_z|}) = gen_n$.

3.4 Beispiel

3.4.1 Exact-Cover-Instanz

Folgende Exact-Cover-Instanz soll zur Verschlüsselung genutzt werden:

$X = \{11, 12, 13, 14, 15, 16\}$
$|X| = n = 6$
$S = \{T_1, T_2, T_3, T_4, T_5\} = \{\{11\}, \{12, 13\}, \{14, 15, 16\}, \{11, 16\}, \{14, 15\}\}$

Gültige Lösungen:
$S_0^* = \{\{11\}, \{12, 13\}, \{14, 15, 16\}\}$
$S_1^* = \{\{12, 13\}, \{11, 16\}, \{14, 15\}\}$

3.4.2 Definition der Gruppen

Nun wird für jedes Element aus X eine Gruppe definiert. In diesem Beispiel wird nur eine einzige Gruppe benutzt:

$\mathbb{Z}_7$:

	1	a	a^2	a^3	a^4	a^5	a^6
1	1	a	a^2	a^3	a^4	a^5	a^6
a	a	a^2	a^3	a^4	a^5	a^6	1
a^2	a^2	a^3	a^4	a^5	a^6	1	a
a^3	a^3	a^4	a^5	a^6	1	a	a^2
a^4	a^4	a^5	a^6	1	a	a^2	a^3
a^5	a^5	a^6	1	a	a^2	a^3	a^4
a^6	a^6	1	a	a^2	a^3	a^4	a^5

Generator: a

Somit ist $G_1 = G_2 = ... = G_6 = \mathbb{Z}_7$.

3.4.3 Zu verschlüsselnde Nachricht

Der Einfachheit halber wird angenommen, dass die Nachricht als Element der Gruppe kodiert werden kann:
$plaintext = a^6$

3.4.4 Definition der multilinearen Abbildung

Es gibt eine Menge von bilinearen Abbildungen der Form $G_i \times G_j \mapsto G_{i+j}$, wobei $i + j \leq n$:

$e_1 : G_1 \times G_2 \mapsto G_3$
$e_1(a^x, a^y) = a^{xy}$

$e_2 : G_3 \times G_3 \mapsto G_6$
$e_2(a^x, a^y) = a^{xy}$

...

Da es im allgemeinen Fall nicht trivial ist, zu einem beliebigem Element aus einer Gruppe den Exponenten des Generators zu ermitteln, stellt obige Definition noch keine Implementierung der bilinearen Abbildungen dar. Die konkrete Implementierung wird hier nicht behandelt.

Aus den bilinearen Abbildungen wird nun die multilineare Abbildung e zusammengebaut:

Beispiel:
$e : G_1 \times G_2 \times G_3 \mapsto G_6$
$e(a^x, a^y, a^z) = e_2(e1(a^x, a^y), a^z)$

$e : G_1 \times G_2 \mapsto G_3$
$e(a^x, a^y) = e_1(a^x, a^y)$

...

Die multilineare Abbildung e erlaubt es, bis zu 6 Elemente miteinander zu verknüpfen.

3.4.5 Verschlüsselung

Nun werden 6 Zufallswerte aus dem Bereich 1 bis 6 bestimmt: $randomVals = \{5, 2, 4, 6, 3, 4\}$

Nun wird das Produkt aller Zufallswerte gebildet:
$rp = 5 \cdot 2 \cdot 4 \cdot 6 \cdot 3 \cdot 4 \pmod 7 = 3$

Das *secret* wird bestimmt:
$secret = a^3$

$ciphertext = plaintext \cdot secret = a^6 \cdot a^3 = a^2$

Die 6 Werte aus $randomVals$ werden den 6 Werten aus X zugeordnet:

Element aus X	11	12	13	14	15	16
Zufallswert	5	2	4	6	3	4

Tabelle 3.1: Zufallswerte und Zuordnung zu X

Nun wird für jede der Teilmengen T_i aus S der Teilschlüssel C_i entsprechend der Zuordnung bestimmt:

i	T_i	$randomVals'_i$	rp'_i	C_i
1	$\{11\}$	$\{5\}$	5	a^5
2	$\{12, 13\}$	$\{2, 4\}$	1	a^1
3	$\{14, 15, 16\}$	$\{6, 3, 4\}$	2	a^2
4	$\{11, 16\}$	$\{5, 4\}$	6	a^6
5	$\{14, 15\}$	$\{6, 3\}$	4	a^4

Tabelle 3.2: Berechnung der Teilschlüssel C_i

Berechnungsbeispiel für $i = 3$:

$T_3 = \{14, 15, 16\}$
$randomVals'_3 = \{6, 3, 4\}$ (Den Elementen 14, 15, 16 zugeordnete Zufallswerte)
$rp'_3 = 6 \cdot 3 \cdot 4 \pmod 7 = 2$
$C_3 = a^{rp'_3} = a^2$

3.4.6 Entschlüsselung

1. Fall:
Lösung: $S_0^* = \{\{T_1\}, \{T_2\}, \{T_3\}\} = \{\{11\}, \{12, 13\}, \{14, 15, 16\}\}$
Somit muss $e(C_1, C_2, C_3)$ berechnet werden:
$e(C_1, C_2, C_3)$
$= e(a^5, a^1, a^2)$
$= e(a, a, a)^{5 \cdot 1 \cdot 2}$
$= e(a, a, a)^3$
$= a^3$

2. Fall:
Lösung: $S_1^* = \{\{T_2\}, \{T_4\}, \{T_5\}\} = \{\{12, 13\}, \{11, 16\}, \{14, 15\}\}$
Somit muss $e(C_2, C_4, C_5)$ berechnet werden:
$e(C_2, C_4, C_5)$
$= e(a^1, a^6, a^4)$
$= e(a, a, a)^{1 \cdot 6 \cdot 4}$
$= e(a, a, a)^3$
$= a^3$

Es kann also in beiden Fällen das *secret* berechnet werden. Daher kann die ursprüngliche Nachricht rekonstruiert werden:

$plaintext = ciphertext / secret = a^2 / a^3 = a^2 \cdot a^4 = a^6$

3.5 Anwendung der Witness Encryption

Die Witness Encryption wurde anhand des Problems Exact Cover erläutert. Da dieses Problem NP-vollständig ist, lässt sich das Verfahren auf beliebige andere Probleme in NP übertragen. Im folgenden Kapitel wird das Problem der Berechnung einer Blockchain verwendet.

4 Bitcoin Time-Lock Encryption

Im folgenden Kapitel wird am Beispiel von Bitcoin gezeigt, wie die vorgestellte Time-Lock Encryption mit dezentralen Kryptowährungen realisiert werden kann. Die Konstruktion einer Blockchain bildet dabei das grundlegende Problem, welche für die Witness Encryption genutzt wird.

4.1 Bitcoin & Blockchain

Bitcoin bildet wie andere Kryptowährungen auch eine Alternative zu den klassischen Währungen. Im Gegensatz zu diesen werden Transaktionen nicht von einer zentralen Instanz, sondern von einem dezentralen System abgewickelt. Der gesamte Transaktionsverlauf wird in einer Datenbank, der Blockchain, abgelegt. Diese Blockchain wird von jedem Bitcoin-Teilnehmer gespeichert. In einem einzelnen Block werden dabei eine oder mehrere Transaktionen gespeichert. Durch diesen so gespeicherten Transaktionsverlauf wird verhindert, dass Bitcoins doppelt ausgegeben werden. Durch kryptographische Schlüssel wird sichergestellt, dass jeder Teilnehmer nur von seinem eigenen Konto Bitcoins übertragen kann.

Wenn eine Transaktion ausgeführt werden soll, muss zur Bestätigung also ein neuer Block erzeugt werden, was in etwa 10 min dauert. Die Erzeugung eines neuen Blocks (Mining) ist mit einem erheblichen Rechenaufwand verbunden. Während das Mining in den Anfangszeiten von Bitcoin noch auf handelsüblichen Rechnern erfolgen konnte, ist es mittlerweile nur auf speziell für diese Aufgabe angepassten ASICs praktikabel. Durch diesen notwendigen Rechenaufwand soll verhindert werden, dass manipulierte Blöcke zur Blockchain hinzugefügt werden. Da das Bitcoin-System stets nur die längste Blockchain als gültig ansieht, müsste ein Angreifer zur Kompromittierung des Systems mehr Rechenleistung besitzen als alle ehrlichen Bitcoin-Miner zusammen. Einen solchen Angriff bezeichnet man als 51%-Angriff.

4.2 Mining

Als Mining wird die Erzeugung neuer Blöcke für die Blockchain bezeichnet. Ein einzelner Block enthält dabei folgende Felder [Jag15]:

- T: Liste von Transaktionen
- r: Nonce (Variabler Zähler zur Prüfsummenberechnung)
- D: Target (bestimmt Schwierigkeit des Mining)
- B: Prüfsumme

Ein einzelner Block ist also ein Tupel (T_i, r_i, D_i, B_i). Die Blockchain ist daher eine Liste von Tupeln: $(T_1, r_1, D_1, B_1), ..., (T_n, r_n, D_n, B_n)$.

Es muss für jeden Block gelten: $B_i = Hash(T_i, r_i, D_i, B_{i-1})$. Das bedeutet, dass die Prüfsumme (Hash) eines Blocks über alle Daten des Blocks sowie über die Prüfsumme des vorangehenden Blocks gebildet wird. Dadurch hängt jeder Block von seinem Vorgänger ab, und es wird verhindert, dass einzelne Blöcke der Blockchain geändert werden.

Weiterhin muss gelten: $B_i \leq D_i$. Also darf als Prüfsumme eines Blocks kein beliebiger Hashwert verwendet werden, sondern der Hashwert muss kleiner oder gleich dem Target sein. Ist ein berechneter Hashwert größer als das Target, so muss die Nonce inkrementiert werden und der neue Hashwert berechnet werden.

Die Schwierigkeit des Minings hängt also vom Target ab. Dieses Target ist variabel und wird alle 2016 Blöcke angepasst [BWi]. Im Idealfall soll alle 10 Minuten ein neuer Block erzeugt werden, wodurch es genau 2 Wochen benötigen würden, 2016 Blöcke zu erzeugen. Wenn das Target angepasst werden soll, wird überprüft, wie lange es tatsächlich gedauert hat, die letzten 2016 Blöcke zu erzeugen. Hat es länger gedauert, wird das Target erhöht, wodurch die Schwierigkeit des Minings sinkt. Die Rate, um die das Target erhöht wird, ist dabei proportional zu der zusätzlich benötigten Zeit. Analog dazu wird das Target verringert, wenn die Blöcke in weniger als 2 Wochen erzeugt wurden.

4.3 Verschlüsselungsschema

Für eine konkrete Instanziierung der Time-Lock Encryption wird nun die Bitcoin-Blockchain verwendet. Da die Berechnung der Blockchain eine iterative öffentliche Berechnung ist, eignet diese sich für das in 2.3 vorgestellte Modell. Das zur Verschlüsselung der Nachrichten genutzte Problem liegt darin, eine Blockchain einer bestimmten Länge zu finden.

Ein *witness* zur Entschlüsselung muss folgende Eigenschaften besitzen:

1. Es ist eine gültige Blockchain mit mindestens n Blöcken
2. Der erste Block dieser Blockchain ist β
3. Für jeden Block gilt: $B_i \leq \delta(i)$

Die Blockchain muss eine gültige Bitcoin-Blockchain sein. Alle Prüfsummen müssen korrekt und kleiner als das jeweilige Target sein.

Der erste Block des witness muss β sein. β entspricht dem Bitcoin Genesis Block, also dem ersten Block der Blockchain. Durch diese Beschränkung wird verhindert, dass ein Angreifer eine komplett neue Blockchain erstellt.

Es wird eine Funktion δ definiert. Diese Funktion legt für jeden Block zusätzlich zum Bitcoin-Target einen Maximalwert für die Prüfsumme fest. Dieser Maximalwert darf von keiner Prüfsumme überschritten werden. Somit kann es passieren, dass eine für das Bitcoin-System gültige Blockchain kein witness für eine bestimmte Nachricht ist. Dies ist der Fall wenn $\delta(i) < B_i \leq D_i$. Um die Entschlüsselbarkeit zu gewährleisten, muss also $\delta(i) \geq D_i$ gelten.

4.4 δ-Funktion

Die δ-Funktion [Jag15] legt für jeden Block der Blockchain zusätzlich zum Target einen Maximalwert fest: $B_i \leq \delta(i)$. Es ist extrem wichtig, dass die δ-Funktion richtig gewählt wird. Sowohl zu große als auch zu kleine Werte können kritisch sein.

Sind die Werte der δ-Funktion zu klein führt dies dazu, dass eine verschlüsselte Nachricht nicht mehr entschlüsselt werden kann. Hierzu genügt es, wenn nur für einen einzelnen Block $\delta(i) < B_i$ gilt.

Sind die Werte der δ-Funktion zu groß, wird das Verschlüsselungsschema zu unsicher. Dies kann von einem Angreifer ausgenutzt werden, um die fehlenden Blöcke schneller zu berechnen und so vor dem Zeitpunkt die Nachricht zu entschlüsseln. Ein mögliches Angriffsszenario wäre dabei:

- Ein Angreifer berechnet eigene Blöcke
- Diese hängt er an die offizielle Blockchain an, veröffentlicht diese jedoch nicht
- Die Zeitstempel der Blöcke werden manipuliert, sodass das Target bei der nächsten Anpassung sehr viel größer wird
- Der Angreifer kann schnell viele Blöcke berechnen (deren Target sehr viel größer ist als das offizielle Target)

Im Idealfall entspricht also die δ-Funktion dem Target. Für alle Blöcke die bereits erzeugt wurden sowie für alle Blöcke die bis zur nächsten Target-Anpassung berechnet werden, wird dieser ideale Wert verwendet: $\delta(i) = D_i$.

Für alle Blöcke, die nach der nächsten Target-Anpassung berechnet werden muss ein Wert $\delta(i)$ geschätzt werden. Da eine erfolgreiche Entschlüsselung nur gewährleistet ist wenn $\delta(i) \geq D_i$, muss somit D_i geschätzt werden. Die Möglichkeit der Abschätzung des Target-Wertes wird im folgenden Abschnitt untersucht.

4.5 Abschätzung der Entwicklung des Targets

Da die zukünftige Entwicklung des Targets nicht bekannt ist, stellt dies eine nicht triviale Aufgabe dar. Es muss abgeschätzt werden, wie sich das Target vom Zeitpunkt der Verschlüsselung bis zum Zeitpunkt ab dem eine Entschlüsselung möglich sein soll, entwickeln wird.

Das Target wird alle 2016 Blöcke basierend Zeit, in der die Blöcke erzeugt wurden, angepasst. Somit hängt die Anpassung des Targets von der Rechenleistung ab, die für den Mining-Prozess zur Verfügung steht. Zur Abschätzung der Target-Entwicklung muss also abgeschätzt werden, wie sich die Mining-Rechenleistung entwickelt. Es gibt folgende Fälle:

- Die Rechenleistung bleibt gleich
- Die Rechenleistung erhöht sich
- Die Rechenleistung verringert sich

Bleibt die Rechenleistung in etwa gleich, verändert sich auch das Target nur wenig. In diesem Fall kann ein konstanter Wert für die δ-Funktion verwendet werden. Dieser sollte nicht zu klein gewählt werden, es kann z. B. das maximale erwartete Target verwendet werden.

Wenn sich die Rechenleistung erhöht, kann dies mehrere Ursachen haben:

- Es stehen mehr Ressourcen zur Verfügung
- Ein technologischer Fortschritt

Wenn mehr Ressourcen zur Verfügung stehen (z. B. weil sich mehr Personen am Mining beteiligen wollen), muss dies nicht zwangsläufig korrekt abgeschätzt werden können. Es ist schon bei der jetzigen Rechenleistung für einen Angreifer praktisch unmöglich, diese zu übertreffen. Daher ist das Schema auch sicher, wenn der aktuelle Target-Wert als minimaler Wert der δ-Funktion genutzt wird.

Wenn die Erhöhung der Rechenleistung in einem technischen Fortschritt (z. B. Verwendung von Quantencomputern) begründet ist, muss dies in der δ-Funktion berücksichtigt werden. Denn die neuen technischen Maßnahmen stehen auch einem potenziellen Angreifer zur Verfügung.

Allgemein funktioniert das Schema bei einer Erhöhung der Rechenleistung jedoch weiterhin, da das Target kontinuierlich vom Bitcoin-System angepasst wird. Die Blockchain der Länge n, welcher als witness für eine Nachricht fungiert, wird somit nicht schneller berechnet.

Wenn sich die Rechenleistung geringfügig verringert (z. B. aufgrund von geringerem Interesse) muss dies in der δ-Funktion berücksichtigt werden, da die Nachricht ansonsten nicht mehr entschlüsselt werden könnte.

Wenn sich die Rechenleistung sehr stark verringert (z. B. durch einen Marktcrash von Bitcoin) ist dies fatal für das Verschlüsselungsschema. Selbst wenn eine solche Entwicklung

korrekt vorhergesagt werden kann, ist es für einen Angreifer dann einfach die öffentliche Mining-Rechenleistung zu übertreffen.

Insgesamt kann die Entwicklung des Targets für kürzere Zeitspannen (z. B. einige Wochen) gut abgeschätzt werden, da sich die Mining-Rechenleistung in diesem Zeitraum nicht stark verändert. Die große Rechenleistung bietet auch genug Spielraum für Fehler. Außerdem wird das Target im Mittel nur alle 2 Wochen angepasst, bleibt also in dieser Zeit konstant.

Für längere Zeitspannen wird es schwieriger, den Verlauf des Targets korrekt abzuschätzen. Es müssen sowohl potenzielle technische Fortschritte als auch sinkendes Interesse vorhergesagt werden. Um dem Problem vorzubeugen, dass ein einzelner zu kleiner δ-Wert eine Verschlüsselung verhindert, kann das zugrunde liegende Problem angepasst werden. Ein Beispiel hierfür wäre, dass nur ein bestimmter Prozentsatz der Blöcke die Bedingung $B_i \leq \delta(i)$ erfüllen muss.

4.6 Wahl der richtigen δ-Funktion

Die δ-Funktion muss richtig gewählt werden. Ist sie zu klein, können Nachrichten nicht mehr entschlüsselt werden. Ist sie zu groß, wird es zu einfach das Verschlüsselungsschema anzugreifen. Da keine allgemeine Aussage darüber getroffen werden kann, welche von beiden Optionen besser ist, ergibt es daher Sinn die δ-Funktion anwendungsspezifisch oder sogar nachrichtenspezifisch festzulegen.

Wenn für eine Nachricht die Sicherheit wichtiger als die Entschlüsselbarkeit ist, sollt die δ-Funktion relativ klein gewählt werden. In diesem Fall gibt es keinen Spielraum, falls die Mining-Rechenleistung sinkt und die Nachricht wäre in diesem Fall nicht entschlüsselbar.

Wenn für eine Nachricht hingegen die Entschlüsselbarkeit wichtiger ist, sollte die δ-Funktion großzügig gewählt werden. Es muss genug Spielraum für ein Sinken der Rechenleistung geben.

4.7 Beispiel

4.7.1 Ausgangssituation

Es folgt ein Beispiel um das Vorgehen zu veranschaulichen. Zur besseren Verständlichkeit werden folgende Vereinfachungen getroffen:

- Es wird eine fiktive Blockchain mit wenigen Blöcken genutzt
- Für die Target-Werte werden fiktive Platzhalter genutzt

Das Ausgangsszenario sieht wie folgt aus:

- Es wurde am 1.1.2018 damit begonnen, die fiktive Blockchain zu berechnen
- Inzwischen wurden 5040 Blöcke berechnet: $B_0, ..., B_{5039}$
- Zeitstempel von $B_{4031} = 28.1.2018$
- Aktuelles Target $D_{5040} = 64$
- Aktuelles Datum: 5.2.2018

Von allen bisher erzeugten Blöcken ist das entsprechende Target und der Zeitstempel bekannt. Hierbei ist insbesondere der Zeitstempel von Block B_{4031} relevant, da zu diesem Datum zuletzt das Target neu berechnet wurde. Seit diesem Datum ist eine Woche vergangen, in diesem Zeitraum wurden 1008 Blöcke berechnet, was dem Idealfall entspricht. Es müssen also weitere 1008 Blöcke berechnet werden, bis das Target wieder angepasst wird.

Nun soll eine Nachricht bis zum 16.4.2019 verschlüsselt werden. Da etwa alle 10 Minuten ein Block berechnet wird, soll zur Entschlüsselung also eine Blockchain der Länge 15120 erforderlich sein. Bis diese Blockchain berechnet wird, muss das Target 5 mal angepasst werden.

Nun wird die δ-Funktion definiert: Die Werte D_0 bis D_{5039} sind bekannt. Bis zum Block B_{6047} wird sich das Target nicht ändern, also kann die Funktion bis zu diesem Block ohne Vermutungen definiert werden. Der weitere Verlauf der Funktion muss geschätzt werden. Es wird eine Erhöhung des Targets geschätzt, damit die Nachricht auch in einem solchen Fall entschlüsselt werden kann. Somit ergibt sich folgende Funktion:

$\delta(i)$	Funktionswert
$\delta(0)$ bis $\delta(6047)$	Bereits bekannte Target-Werte D_0 bis D_{6047}
$\delta(6048)$ bis $\delta(8063)$	76
$\delta(8064)$ bis $\delta(10079)$	92
$\delta(10080)$ bis $\delta(12095)$	110
$\delta(12096)$ bis $\delta(14111)$	132
$\delta(14112)$ bis $\delta(15119)$	159

Tabelle 4.1: Gewählte δ-Funktion

Im folgenden werden nun verschiedene Szenarien betrachtet.

4.7.2 Target erhöht sich leicht

Datum	Blockchain-Länge	Target D_i	$\delta(i)$
5.2.2018	5040	64	64
14.2.2018	6048	73	76
2.3.2018	8064	83	92
17.3.2018	10080	89	110
31.3.2018	12096	89	132
15.4.2018	14112	95	159
22.4.2018	15120	95	159

Tabelle 4.2: Blockchain-Entwicklung (Leicht steigendes Target)

Durch die kontinuierliche Verringerung der Rechenleistung wird die Blockchain langsamer berechnet. Durch die Anpassung des Targets wird dieser Effekt abgeschwächt. Die δ-Funktion wurde passend gewählt, es gilt stets $D_i \leq \delta(i)$, somit gilt auch immer $B_i \leq \delta(i)$. Daher kann die Nachricht entschlüsselt werden. Durch die Verringerung der Rechenleistung liegt das Entschlüsselungsdatum 6 Tage nach dem gewünschtem Datum.

4.7.3 Target erhöht sich stark

Datum	Blockchain-Länge	Target D_i	$\delta(i)$
5.2.2018	5040	64	64
16.2.2018	6048	82	76
6.3.2018	8064	106	92
21.3.2018	10080	113	110
5.4.2018	12096	121	132
19.4.2018	14112	121	159
26.4.2018	15120	121	159

Tabelle 4.3: Blockchain-Entwicklung (Stark steigendes Target)

In diesem Szenario übersteigen einige Target-Werte den abgeschätzten Wert der δ-Funktion (Blöcke 6048 bis 12095). Die Prüfsummen der entsprechenden Blöcke könnten somit höher als δ sein. Daher ist nicht garantiert, dass die Nachricht entschlüsselt werden kann. Für diesen Fall hätte also die δ-Funktion noch pessimistischer abgeschätzt werden sollen.

4.7.4 Target verringert sich

Datum	Blockchain-Länge	Target D_i	$\delta(i)$
5.2.2018	5040	64	64
10.2.2018	6048	55	76
23.2.2018	8064	51	92
9.3.2018	10080	51	110
21.3.2018	12096	44	132
3.4.2018	14112	41	159
10.4.2018	15120	41	159

Tabelle 4.4: Blockchain-Entwicklung (Fallendes Target)

Durch die kontinuierliche Erhöhung der Rechenleistung wird die Blockchain schneller berechnet. Da bei jedem Block $B_i \leq \delta(i)$ gilt, kann die Nachricht entschlüsselt werden. Durch die Erhöhung der Rechenleistung liegt das Entschlüsselungsdatum 6 Tage vor dem gewünschtem Datum.

4.7.5 Angriffsversuch

Datum	Manipulierter Zeitstempel der Blöcke	Blockchain-Länge	Target D_i	$\delta(i)$
5.2.2018	-	5040	64	64
26.2.2018	12.09.2018	6048	1000	76
26.2.2018	26.09.2018	8064	1000	92
26.2.2018	10.10.2018	10080	1000	110
26.2.2018	24.10.2018	12096	1000	132
26.2.2018	7.11.2018	14112	1000	159
26.2.2018	14.11.2018	15120	1000	159

Tabelle 4.5: Entwicklung der privaten Blockchain des Angreifers

Der Angreifer berechnet eine inoffizielle Blockchain. Da noch 1008 Blöcke bis zur nächsten Target-Berechnung berechnet werden müssen, können diese Blöcke vom Angreifer nur langsam berechnet werden (Blöcke 5040 bis 6047). Der Angreifer setzt aber die Zeitstempel der Blöcke auf ein in der Zukunft liegendes Datum. Daher wirkt es für die private Bitcoin-Instanz des Angreifers, als ob sich die Mining-Rechenleistung verschlechtert. Das Target wird erhöht und der Angreifer kann sehr schnell viele Blöcke berechnen.

Wäre die δ-Funktion schlecht gewählt, so könnte der Angreifer die Nachricht schon lange vor dem gewähltem Datum entschlüsseln (7 Wochen vorher). Da das angepasste Target jedoch über der gewählten δ-Funktion liegt, kann die Nachricht so nicht entschlüsselt werden.

5 Fazit

In dieser Ausarbeitung wurde eine Möglichkeit zur zeitbasierten Verschlüsselung von Nachrichten vorgestellt. Da die Verschlüsselung unabhängig von dritten Parteien sein sollte, wurde ein Schema entwickelt, welches auf öffentlichen Berechnungen basiert. Die Grundlage, um derartige Berechnungen zur Verschlüsselung zu nutzen, bildet die Witness Encryption. Daher wurde diese näher erläutert und basierend auf dem NP-vollständigen Problem Exact Cover die Verfahren zur Ver- und Entschlüsselung beschrieben.

Weiterhin wurde die praktische Umsetzung des Verfahrens mit Bitcoin diskutiert. Das konkrete Problem zur Verschlüsselung war dabei, eine gültige Blockchain der Länge n zu finden. Um die Sicherheit des Verschlüsselungsschemas zu erhalten, wurde die δ-Funktion als zusätzliche Restriktion für die Blockchain eingeführt.

Um die Werte der δ-Funktion passend zu wählen, müssen annähernd präzise Aussagen zur zukünftigen Entwicklung des Bitcoin-Targets gemacht werden, was insbesondere bei längeren Zeiträumen nicht möglich ist. Weiterhin ist die Sicherheit der Time-Lock Encryption an die Integrität von Bitcoin gebunden. Falls das Interesse an Bitcoin stark sinkt, sodass 51%-Angriffe möglich sind, ist auch die Verschlüsselung nicht mehr sicher.

Anstatt von Bitcoin können auch beliebige andere öffentlich Berechnungen als Grundlage für die Time-Lock Encryption verwendet werden. Allerdings gilt auch für diese, dass ein hohes Interesse bis zum gewählten Zeitpunkt gewährleistet sein muss, um 51%-Angriffe zu unterbinden. Das macht es schwer, Nachrichten für einen weit in der Zukunft liegenden Zeitpunkt (z. B. 10 Jahre) sicher zu verschlüsseln. Für kürzere Zeiträume eignet sich das Verfahren hingegen gut, da hier genauere Prognosen möglich sind.

Literaturverzeichnis

[BWi] Bitcoin Wiki (https://en.bitcoin.it). Abgerufen am 06.04.2018.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, 2013.

[Haz94] Michiel Hazewinkel. *Encyclopedia of mathematics*. Kluwer, 1994.

[Jag15] Tibor Jager. How to Build Time-Lock Encryption. *IACR Cryptology ePrint Archive*, 2015:478, 2015.

[Kar72] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.