

## Auf einen Blick

1	Vorwort .....	17
2	Einführung .....	21
3	Die Basics: So funktioniert eine Blockchain .....	67
4	Eine eigene Blockchain erstellen – Grundfunktionen .....	167
5	Die Blockchain an eine Web-API anbinden .....	189
6	Ein Peer-to-Peer-Netzwerk aufbauen .....	215
7	Accounts und Guthaben einführen .....	233
8	Verifikation und Optimierungen umsetzen .....	251
9	Smart Contract Development .....	269
10	Solidity – Die Grundlagen verstehen .....	285
11	Solidity – Details und Herausforderungen .....	325
12	Smart Contracts testen und debuggen .....	363
13	Smart Contracts schützen und absichern .....	385
14	Smart Contracts deployen und managen .....	407
15	Standards, Libraries und Design Patterns .....	429
16	Decentralized Applications entwickeln .....	453
17	Ihre erste DApp mit Drizzle zur DAO erweitern .....	475
18	Dezentrale Autonome Initiale Coin Offerings umsetzen .....	497
19	Supply Chains mit Smart Contracts ergänzen .....	519
20	Ausblick: Zukunftstechnologien .....	535

# Inhalt

Materialien zum Buch .....	16
<b>1 Vorwort</b> .....	<b>17</b>
1.1 Vorwort der Autoren .....	17
1.2 Geleitwort .....	19
<b>2 Einführung</b> .....	<b>21</b>
2.1 Was ist die Blockchain? .....	21
2.1.1 Herausforderungen des Internets .....	21
2.1.2 Die Blockchain .....	24
2.1.3 Die Blockchain als Problemlöser .....	28
2.2 Geschichte der Blockchain .....	29
2.2.1 Pioniere der Blockchain .....	29
2.2.2 Bitcoin .....	30
2.2.3 Altcoins .....	35
2.2.4 Blockchain 2.0 .....	36
2.2.5 Gegenwart und Zukunft .....	41
2.3 Anwendung der Blockchain-Technologie .....	43
2.3.1 Entscheidungskriterien für die Blockchain .....	43
2.3.2 Blockchain-Varianten .....	45
2.3.3 Branchen mit Blockchain-Potenzial .....	48
2.3.4 Realbeispiele für Blockchain-Anwendungen .....	54
2.4 Zusammenfassung .....	65
<b>3 Die Basics: So funktioniert eine Blockchain</b> .....	<b>67</b>
3.1 Kryptografische Grundlagen .....	67
3.1.1 Einführung in die Kryptografie .....	68
3.1.2 Elliptic Curve Cryptography (ECC) .....	72
3.1.3 Kryptografische Hashfunktionen .....	75

<b>3.2 Die Blockchain</b>	80
3.2.1 Transaktionen	81
3.2.2 Vom Block zur Blockchain	86
3.2.3 Das Blockchain-System	90
3.2.4 Weiterentwicklungen der Blockchain	105
<b>3.3 Die Blockchain 2.0</b>	112
3.3.1 Einführung und Grundlagen	113
3.3.2 Accounts und State Trie	117
3.3.3 Transaktionen und Transaction Trie	120
3.3.4 Receipts und Receipts Trie	123
3.3.5 Vom Block zur Blockchain 2.0	125
3.3.6 Das Blockchain-System 2.0	129
3.3.7 Weiterentwicklung der Ethereum-Plattform	137
<b>3.4 Alternative Konsensmodelle</b>	144
3.4.1 Proof-of-Stake (PoS)	145
3.4.2 Delegated Byzantine Fault Tolerance (dBFT)	147
3.4.3 Proof-of-Activity	148
3.4.4 Proof-of-Importance	149
3.4.5 Proof-of-Authority	149
3.4.6 Proof-of-Reputation	150
3.4.7 Proof-of-Capacity/Proof-of-Space	151
3.4.8 Proof-of-Elapsed-Time	152
3.4.9 Proof-of-Burn	152
<b>3.5 Sicherheit der Blockchain</b>	153
3.5.1 Blockchain und Informationssicherheit	153
3.5.2 Angriffsszenarien	156
<b>3.6 Zusammenfassung</b>	164
<b>4 Eine eigene Blockchain erstellen – Grundfunktionen</b>	167
<hr/>	
<b>4.1 Transaktionen – die kleinste Einheit</b>	169
<b>4.2 Blockheader – der Inhalt der Block-ID</b>	171
<b>4.3 Die Blöcke verketteten</b>	173
<b>4.4 Die Blockchain auf die Festplatte speichern</b>	175

<b>4.5 Der Genesis Block – die Entstehung einer Blockchain</b>	177
<b>4.6 Ausstehende Transaktionen</b>	178
<b>4.7 Die Difficulty einer Blockchain</b>	180
<b>4.8 Zeit zu schürfen – der Miner Thread</b>	182
<b>4.9 Zusammenfassung und Ausblick</b>	186
<b>5 Die Blockchain an eine Web-API anbinden</b>	189
<hr/>	
<b>5.1 Die Service-Endpunkte der Web-API</b>	190
5.1.1 Die Service-Klasse für Blöcke implementieren	192
5.1.2 Die Service-Klasse für Transaktionen implementieren	194
<b>5.2 Deployment der Web-API</b>	196
5.2.1 Erstellen einer Ressourcenkonfiguration	196
5.2.2 Einrichten eines embedded Tomcat	198
5.2.3 Die ausgelieferten JSON-Repräsentationen überprüfen	199
<b>5.3 Transaktionen per Webinterface versenden</b>	200
<b>5.4 Einen eigenen Block-Explorer implementieren</b>	204
5.4.1 Transaktionen erkunden	204
5.4.2 Blöcke erkunden	206
5.4.3 Startseite und Suchfunktion implementieren	209
<b>5.5 Zusammenfassung und Ausblick</b>	211
<b>6 Ein Peer-to-Peer-Netzwerk aufbauen</b>	215
<hr/>	
<b>6.1 Das Peer-to-Peer-Framework konfigurieren</b>	216
<b>6.2 Transaktionen im Netzwerk verteilen</b>	219
<b>6.3 Blöcke im Netzwerk verteilen</b>	222
<b>6.4 Mehrere Chains parallel verarbeiten</b>	224
6.4.1 Chains aufbewahren und bei Bedarf wechseln	224
6.4.2 Ausstehende Transaktionen beim Chain-Wechsel aktualisieren	227
<b>6.5 Neue Knoten im Netzwerk aufnehmen</b>	228
<b>6.6 Zusammenfassung und Ausblick</b>	231

<b>7</b>	<b>Accounts und Guthaben einführen</b>	233
<b>7.1</b>	<b>Die Miner belohnen</b>	234
7.1.1	Den Minern einen Account zuweisen	234
7.1.2	Accounts persistent speichern	235
7.1.3	Den Blöcken einen Miner zuweisen	237
<b>7.2</b>	<b>Die Accounts verwalten</b>	238
7.2.1	Accounts speichern	238
7.2.2	Accounts initialisieren und aktualisieren	240
7.2.3	Account-Informationen über die Web-API bereitstellen	241
<b>7.3</b>	<b>Die Accounts integrieren</b>	242
<b>7.4</b>	<b>Die Accounts im Block-Explorer einbinden</b>	243
7.4.1	Accounts mit dem Block-Explorer einsehen	243
7.4.2	Accounts im Webclient generieren	246
7.4.3	Accounts im Block-Explorer verlinken und suchen	248
<b>7.5</b>	<b>Zusammenfassung und Ausblick</b>	249
<b>8</b>	<b>Verifikation und Optimierungen umsetzen</b>	251
<b>8.1</b>	<b>Transaktionen signieren</b>	251
8.1.1	Digitale Signaturen im Webclient einführen	252
8.1.2	Digitale Signaturen im Backend unterstützen	253
<b>8.2</b>	<b>Die Rahmenbedingungen erzwingen</b>	255
8.2.1	Transaktionen verifizieren	255
8.2.2	Blöcke verifizieren	256
<b>8.3</b>	<b>Guthaben sperren und entsperren</b>	257
<b>8.4</b>	<b>Mit dem Merkle-Baum die Performance optimieren</b>	260
8.4.1	Die Struktur des Merkle-Baums erstellen	260
8.4.2	Den Merkle-Baum über die Web-API nutzen	262
<b>8.5</b>	<b>Den Public Key verkürzen zum Sparen von Speicher</b>	263
<b>8.6</b>	<b>Startguthaben über den Genesis Block ermöglichen</b>	264
<b>8.7</b>	<b>Weitere Optimierungen bedenken</b>	265
<b>8.8</b>	<b>Zusammenfassung und Ausblick</b>	267

<b>9</b>	<b>Smart Contract Development</b>	269
<b>9.1</b>	<b>Einführung</b>	270
<b>9.2</b>	<b>Einfache Smart Contracts bei Bitcoin</b>	272
9.2.1	Bitcoin Script	272
9.2.2	Smart Contracts mit Bitcoin Script	275
9.2.3	Höhere Programmiersprachen für Bitcoin	279
<b>9.3</b>	<b>Anspruchsvolle Smart Contracts</b>	280
9.3.1	Bitcoin-Erweiterungen	280
9.3.2	Smart Contracts mit Ethereum	282
<b>9.4</b>	<b>Zusammenfassung</b>	283
<b>10</b>	<b>Solidity – Die Grundlagen verstehen</b>	285
<b>10.1</b>	<b>Was ist Solidity?</b>	285
10.1.1	Die offizielle Entwicklungsumgebung Remix	286
10.1.2	Aufbau eines Sourcefiles	287
10.1.3	Den ersten Smart Contract erstellen	288
10.1.4	Den ersten Smart Contract lokal deployen	288
<b>10.2</b>	<b>Elemente und Speicherbereiche eines Contracts</b>	291
10.2.1	Die Speicherbereiche verstehen	291
10.2.2	Sichtbarkeiten von Solidity korrekt nutzen	293
10.2.3	Modifier verwenden und selbst definieren	294
10.2.4	Zustandsvariablen deklarieren und initialisieren	296
10.2.5	Contracts erzeugen und zerstören	297
10.2.6	Funktionen implementieren	298
10.2.7	Events definieren und zum Loggen verwenden	299
<b>10.3</b>	<b>Verfügbare Datentypen</b>	300
10.3.1	Primitive Datentypen verwenden	301
10.3.2	Adressen definieren	302
10.3.3	Arrays anlegen und nutzen	304
10.3.4	Mehrdimensionale Arrays und ihre Einschränkungen berücksichtigen	306
10.3.5	Structs und Enums definieren	308
10.3.6	Mappings und ihre Besonderheiten verstehen	310
10.3.7	Storage Pointer als Funktionsparameter definieren	310
10.3.8	Funktionen als Variablen verwenden	310

<b>10.4</b>	<b>Zusätzliche Features von Solidity</b>	312
10.4.1	LValues verstehen	312
10.4.2	Variablen löschen und Storage freigeben	312
10.4.3	Elementare Datentypen ineinander umwandeln	313
10.4.4	Typherleitung nutzen	313
<b>10.5</b>	<b>Vererbungshierarchien von Smart Contracts erstellen</b>	314
10.5.1	Wie funktioniert die Vererbung von Contracts?	314
10.5.2	Abstrakte Contracts verwenden	316
10.5.3	Interfaces in Solidity definieren	316
<b>10.6</b>	<b>Libraries erstellen und verwenden</b>	317
10.6.1	Eine eigene Library implementieren	317
10.6.2	Libraries in Contracts verwenden	319
10.6.3	Datentypen mit Libraries erweitern	320
<b>10.7</b>	<b>Zusammenfassung und Ausblick</b>	321
<b>11</b>	<b>Solidity – Details und Herausforderungen</b>	325
<b>11.1</b>	<b>Wichtige Details zu Funktionen</b>	326
11.1.1	Polymorphismus richtig anwenden	326
11.1.2	Funktionen überladen	327
11.1.3	Ether empfangen mit der Fallback-Funktion	328
<b>11.2</b>	<b>Gas verstehen und optimieren</b>	330
<b>11.3</b>	<b>Den richtigen Exception-Mechanismus wählen</b>	333
<b>11.4</b>	<b>Solidity mit Assembly erweitern</b>	335
11.4.1	Inline Assembly in Solidity anwenden	335
11.4.2	Inline Assembly mit dem funktionalen Stil schreiben	337
11.4.3	Inline Assembly über Instruktionen schreiben	338
11.4.4	Standalone Assembly als Alternative zu Solidity verwenden	340
11.4.5	Joyfully Universal Language for (Inline) Assembly	340
<b>11.5</b>	<b>Leicht verständliche Contracts entwickeln</b>	341
<b>11.6</b>	<b>Updatefähige Contracts entwickeln</b>	343
11.6.1	Separation Pattern anwenden	344
11.6.2	Proxy Pattern anwenden	349
<b>11.7</b>	<b>Warum kein Zufallsgenerator sicher ist</b>	354
11.7.1	Blockvariablen verwenden	354
11.7.2	Fortlaufende Nummern verwenden	355

11.7.3	Zweistufige Lotterien verwenden	355
11.7.4	Zufall außerhalb der Blockchain ermitteln	356
<b>11.8</b>	<b>Daten von außerhalb der Blockchain vertrauen</b>	356
<b>11.9</b>	<b>Zeitabhängigkeiten einbauen</b>	357
11.9.1	Zeitabhängigkeiten über die Blockzeit prüfen	358
11.9.2	Externe Services verwenden	359
<b>11.10</b>	<b>Zusammenfassung und Ausblick</b>	359
<b>12</b>	<b>Smart Contracts testen und debuggen</b>	363
<b>12.1</b>	<b>Contracts mit Remix testen</b>	364
12.1.1	Unittests innerhalb von Remix ausführen	364
12.1.2	Remix-Tests in Continuous-Integration-Systeme integrieren	366
<b>12.2</b>	<b>Debugging mit Remix</b>	366
<b>12.3</b>	<b>Einführung in das Truffle Framework</b>	368
12.3.1	Das Truffle Framework aufsetzen	368
12.3.2	Migrationen in Truffle verwenden	370
<b>12.4</b>	<b>Unittests mit Truffle implementieren</b>	372
12.4.1	Unittests in Solidity mit den Truffle-Funktionen implementieren	372
12.4.2	Exceptions mit dem Truffle Framework testen	373
12.4.3	Unittests für Ether-Transaktionen implementieren	377
<b>12.5</b>	<b>Integrationstests mit Truffle implementieren</b>	377
12.5.1	Mit Contracts im JavaScript-Code interagieren	378
12.5.2	Unterschiedliche Schreibweisen verstehen	379
12.5.3	Testläufe starten und auswerten	379
12.5.4	Events in JavaScript überprüfen	380
<b>12.6</b>	<b>Mit dem Truffle Framework debuggen</b>	381
<b>12.7</b>	<b>Zusammenfassung und Ausblick</b>	382
<b>13</b>	<b>Smart Contracts schützen und absichern</b>	385
<b>13.1</b>	<b>Allgemeine Sicherheitsempfehlungen</b>	385
13.1.1	Sichtbarkeiten explizit angeben	386
13.1.2	Konstruktoren nur über das Schlüsselwort definieren	386
13.1.3	Storage Pointer immer initialisieren	387

13.1.4	Race Conditions im Hinterkopf behalten .....	387
13.1.5	Rückgabewerte von Low-Level-Funktionen überprüfen .....	387
13.1.6	Manipulationen durch Miner berücksichtigen .....	388
<b>13.2</b>	<b>Ether in Contracts schmuggeln .....</b>	<b>388</b>
<b>13.3</b>	<b>Arithmetische Overflows und Underflows .....</b>	<b>391</b>
<b>13.4</b>	<b>Mit DelegateCalls den Zustand manipulieren .....</b>	<b>393</b>
<b>13.5</b>	<b>Reentrancy-Angriffe durchführen .....</b>	<b>396</b>
<b>13.6</b>	<b>Denial-of-Service-Angriffe durchführen .....</b>	<b>399</b>
<b>13.7</b>	<b>Gas-Siphoning-Angriffe beachten .....</b>	<b>401</b>
<b>13.8</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>404</b>
<b>14</b>	<b>Smart Contracts deployen und managen .....</b>	<b>407</b>
<b>14.1</b>	<b>MetaMask einrichten und Accounts verwenden .....</b>	<b>408</b>
<b>14.2</b>	<b>Contracts mit Remix und MetaMask deployen .....</b>	<b>409</b>
<b>14.3</b>	<b>Contracts mit dem Truffle Framework deployen .....</b>	<b>411</b>
14.3.1	Contracts auf Ganache deployen .....	412
14.3.2	Infura für das Deployen auf Live-Blockchains verwenden .....	415
14.3.3	Die Mnemonics verschlüsselt im Projekt verwenden .....	417
14.3.4	Tipps für die Nutzung von Truffle .....	418
<b>14.4</b>	<b>Code auf Etherscan veröffentlichen .....</b>	<b>418</b>
<b>14.5</b>	<b>Einen eigenen Knoten aufsetzen und verwenden .....</b>	<b>420</b>
<b>14.6</b>	<b>Contracts manuell kompilieren, linken und deployen .....</b>	<b>422</b>
<b>14.7</b>	<b>Contracts nach dem Deployment managen .....</b>	<b>424</b>
14.7.1	Contracts über Remix managen .....	425
14.7.2	Contracts über die Truffle-Konsole managen .....	425
<b>14.8</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>426</b>
<b>15</b>	<b>Standards, Libraries und Design Patterns .....</b>	<b>429</b>
<b>15.1</b>	<b>ERC-173 Contract Ownership Standard .....</b>	<b>429</b>
<b>15.2</b>	<b>ERC-165 Standard Interface Detection .....</b>	<b>432</b>
<b>15.3</b>	<b>ERC-20 Token Standard .....</b>	<b>436</b>

<b>15.4</b>	<b>ERC-721 Non-Fungible Token Standard .....</b>	<b>440</b>
<b>15.5</b>	<b>Die Libraries von OpenZeppelin nutzen .....</b>	<b>445</b>
<b>15.6</b>	<b>Design Patterns verwenden .....</b>	<b>446</b>
15.6.1	Die Struktur des PubSub Patterns verstehen .....	446
15.6.2	Das PubSub Pattern implementieren .....	447
<b>15.7</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>450</b>
<b>16</b>	<b>Decentralized Applications entwickeln .....</b>	<b>453</b>
<b>16.1</b>	<b>Was ist eine Decentralized Application? .....</b>	<b>453</b>
<b>16.2</b>	<b>Der Entwicklungsprozess einer DApp .....</b>	<b>455</b>
<b>16.3</b>	<b>Das Backend Ihrer ersten DApp entwickeln .....</b>	<b>457</b>
<b>16.4</b>	<b>Das Frontend Ihrer ersten DApp entwickeln .....</b>	<b>460</b>
<b>16.5</b>	<b>Ihre erste DApp auf Swarm deployen .....</b>	<b>463</b>
16.5.1	Einen Swarm-Knoten betreiben .....	463
16.5.2	Die DApp vorbereiten und deployen .....	464
16.5.3	Alternativen zum eigenen Knoten .....	466
<b>16.6</b>	<b>ENS-Domains für die eigene DApp einrichten .....</b>	<b>467</b>
16.6.1	Eine ENS-Domain über Onlinetools erwerben .....	467
16.6.2	Eine ENS-Domain über einen eigenen Geth-Knoten erwerben .....	468
16.6.3	Domains mit dem ENS Manager verwalten .....	469
<b>16.7</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>471</b>
<b>17</b>	<b>Ihre erste DApp mit Drizzle zur DAO erweitern .....</b>	<b>475</b>
<b>17.1</b>	<b>Was ist Drizzle? .....</b>	<b>476</b>
17.1.1	Die Architektur von Drizzle .....	476
17.1.2	Die Synchronisation der Blockchain-Daten .....	477
17.1.3	Weitere Funktionen von Drizzle .....	477
<b>17.2</b>	<b>Was ist eine DAO, und was wird benötigt? .....</b>	<b>478</b>
<b>17.3</b>	<b>Die DApp, um einen Governance Contract erweitern .....</b>	<b>479</b>
<b>17.4</b>	<b>Das Frontend mit Drizzle und React umsetzen .....</b>	<b>480</b>
17.4.1	Das Frontend-Projekt aufsetzen .....	481
17.4.2	Drizzle korrekt konfigurieren .....	482
17.4.3	Drizzle in das Frontend über React einbinden .....	483

17.4.4	Den Einstiegspunkt Ihrer DApp vorbereiten .....	484
17.4.5	Das Frontend für den Governance Contract implementieren .....	486
17.4.6	Das Frontend für die Abstimmung implementieren .....	490
<b>17.5</b>	<b>Die Drizzle-DApp zentral oder dezentral deployen .....</b>	<b>492</b>
17.5.1	Die DApp mit einem lokalen Server testen .....	493
17.5.2	Die Drizzle-DApp dezentral deployen .....	494
<b>17.6</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>494</b>

## **18 Dezentrale Autonome Initiale Coin Offerings umsetzen** 497

<b>18.1</b>	<b>Was ist ein DAICO? .....</b>	<b>497</b>
18.1.1	Welche Komponenten benötigt ein DAICO? .....	499
18.1.2	Theoretische Analyse der Sicherheitsprobleme .....	499
18.1.3	Der zeitliche Ablauf eines DAICO .....	500
<b>18.2</b>	<b>Die Contracts eines DAICO implementieren .....</b>	<b>501</b>
18.2.1	Den Crowdsale und das Token mit OpenZeppelin umsetzen .....	501
18.2.2	Den bisherigen Contract für Abstimmungen anpassen .....	503
18.2.3	Den Governance Contract eines DAICO als Escrow einsetzen .....	504
18.2.4	Eine Migration für das gesamte DAICO konfigurieren .....	509
<b>18.3</b>	<b>Die passende DApp zu Ihrem DAICO entwerfen .....</b>	<b>510</b>
18.3.1	Mehr Routen für den Einstiegspunkt definieren .....	510
18.3.2	Das Frontend der DAO zum Escrow umgestalten .....	511
18.3.3	Den Token Contract per DApp verwalten .....	514
18.3.4	Am Crowdsale über Ihre DApp teilnehmen .....	516
<b>18.4</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>517</b>

## **19 Supply Chains mit Smart Contracts ergänzen** 519

<b>19.1</b>	<b>Ideen, Visionen und Problemlösungen .....</b>	<b>520</b>
<b>19.2</b>	<b>Assets mit dem ERC-721 Token Standard tracken .....</b>	<b>521</b>
19.2.1	Proof-of-Concept: der mobile Nadelmarkierer .....	521
19.2.2	Ein Asset-Token mit OpenZeppelin implementieren .....	522
<b>19.3</b>	<b>web3j: die Web3 Java Ethereum DApp API .....</b>	<b>522</b>
19.3.1	Contract Wrapper für Java generieren .....	523
19.3.2	Eine Web3-Instanz initialisieren und konfigurieren .....	523

19.3.3	Transaktionen und Calls auslösen .....	524
19.3.4	Gas mit dem GasProvider regulieren .....	525
19.3.5	Events filtern und überwachen .....	526
<b>19.4</b>	<b>Ein Command Line Interface für Asset Tracker bauen .....</b>	<b>527</b>
19.4.1	Das Maven-Projekt vorbereiten .....	527
19.4.2	Die Implementierung eines Apache Commons CLI .....	528
19.4.3	Den Asset Contract deployen und laden .....	529
19.4.4	Neue Assets per CLI erzeugen .....	530
19.4.5	Die Metadaten automatisiert auf Swarm dezentral deployen .....	531
<b>19.5</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>532</b>

## **20 Ausblick: Zukunftstechnologien** 535

<b>20.1</b>	<b>Vyper – Smart Contracts für jedermann? .....</b>	<b>535</b>
20.1.1	Die Ziele von Vyper .....	536
20.1.2	Einschränkungen von Vyper .....	536
20.1.3	Die Syntax von Vyper .....	537
<b>20.2</b>	<b>NEO – der chinesische Klon von Ethereum .....</b>	<b>539</b>
20.2.1	Die Idee hinter NEO .....	539
20.2.2	Smart Contracts in NEO .....	540
<b>20.3</b>	<b>EOS – der stärkste Wettbewerber zu Ethereum? .....</b>	<b>541</b>
20.3.1	Die Vision von EOS .....	541
20.3.2	Die Architektur von EOS .....	542
20.3.3	Smart Contracts in EOS .....	543
<b>20.4</b>	<b>Ripple – die Zukunft der Banken? .....</b>	<b>544</b>
20.4.1	Die Idee von Ripple .....	544
20.4.2	Der Ledger und das Netzwerk .....	546
20.4.3	Smart Contracts in Ripple mit Codius .....	546
<b>20.5</b>	<b>IOTA – das dezentrale Internet der Dinge .....</b>	<b>548</b>
20.5.1	Das Projekt .....	548
20.5.2	Adressen und Transaktionen .....	549
20.5.3	Der Tangle .....	551
20.5.4	Die Knoten und das Netzwerk .....	553
20.5.5	Die Zukunft wird smart: Smart Contracts in IOTA .....	553
<b>20.6</b>	<b>Zusammenfassung .....</b>	<b>554</b>
	Literaturverzeichnis .....	557
	Index .....	559