Inhalt

Materialien zum Buch 16

1 Vorwort 17

1.1 Vorwort der Autoren 17

1.2 Geleitwort 19

2 Einführung 21

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

3 Die Basics: So funktioniert eine Blockchain 67

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

3.2 Die Blockchain ... 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

3.3 Die Blockchain 2.0 ... 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

3.4 Alternative Konsensmodelle ... 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

3.5 Sicherheit der Blockchain ... 153
3.5.1 Blockchain und Informationssicherheit ... 153
3.5.2 Angriffsszenarien ... 156

3.6 Zusammenfassung ... 164

4 Eine eigene Blockchain erstellen – Grundfunktionen ... 167

4.1 Transaktionen – die kleinste Einheit ... 169

4.2 Blockheader – der Inhalt der Block-ID ... 171

4.3 Die Blöcke verketten ... 173

4.4 Die Blockchain auf die Festplatte speichern ... 175

4.5 Der Genesis Block – die Entstehung einer Blockchain ... 177
4.6 Ausstehende Transaktionen ... 178
4.7 Die Difficulty einer Blockchain ... 180
4.8 Zeit zu schürfen – der Miner Thread ... 182
4.9 Zusammenfassung und Ausblick ... 186

5 Die Blockchain an eine Web-API anbinden 189

5.1 Die Service-Endpunkte der Web-API ... 190
5.1.1 Die Service-Klasse für Blöcke implementieren ... 192
5.1.2 Die Service-Klasse für Transaktionen implementieren ... 194
5.2 Deployment der Web-API ... 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
5.3 Transaktionen per Webinterface versenden ... 200
5.4 Einen eigenen Block-Explorer implementieren ... 204
5.4.1 Transaktionen erkunden ... 204
5.4.2 Blöcke erkunden ... 206
5.4.3 Startseite und Suchfunktion implementieren ... 209
5.5 Zusammenfassung und Ausblick ... 211

6 Ein Peer-to-Peer-Netzwerk aufbauen 215

6.1 Das Peer-to-Peer-Framework konfigurieren ... 216
6.2 Transaktionen im Netzwerk verteilen ... 219
6.3 Blöcke im Netzwerk verteilen ... 222
6.4 Mehrere Chains parallel verarbeiten ... 224
6.4.1 Chains aufbewahren und bei Bedarf wechseln ... 224
6.4.2 Ausstehende Transaktionen beim Chain-Wechsel aktualisieren ... 227
6.5 Neue Knoten im Netzwerk aufnehmen ... 228
6.6 Zusammenfassung und Ausblick ... 231

7 Accounts und Guthaben einführen 233

7.1 Die Miner belohnen 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

7.2 Die Accounts verwalten 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

7.3 Die Accounts integrieren 242

7.4 Die Accounts im Block-Explorer einbinden 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

7.5 Zusammenfassung und Ausblick 249

8 Verifikation und Optimierungen umsetzen 251

8.1 Transaktionen signieren 251
8.1.1 Digitale Signaturen im Webclient einführen 252
8.1.2 Digitale Signaturen im Backend unterstützen 253

8.2 Die Rahmenbedingungen erzwingen 255
8.2.1 Transaktionen verifizieren 255
8.2.2 Blöcke verifizieren 256

8.3 Guthaben sperren und entsperren 257

8.4 Mit dem Merkle-Baum die Performance optimieren 260
8.4.1 Die Struktur des Merkle-Baums erstellen 260
8.4.2 Den Merkle-Baum über die Web-API nutzen 262

8.5 Den Public Key verkürzen zum Sparen von Speicher 263

8.6 Startguthaben über den Genesis Block ermöglichen 264

8.7 Weitere Optimierungen bedenken 265

8.8 Zusammenfassung und Ausblick 267

9 Smart Contract Development 269

9.1 Einführung ... 270

9.2 Einfache Smart Contracts bei Bitcoin ... 272

9.2.1 Bitcoin Script ... 272

9.2.2 Smart Contracts mit Bitcoin Script ... 275

9.2.3 Höhere Programmiersprachen für Bitcoin ... 279

9.3 Anspruchsvolle Smart Contracts ... 280

9.3.1 Bitcoin-Erweiterungen ... 280

9.3.2 Smart Contracts mit Ethereum ... 282

9.4 Zusammenfassung ... 283

10 Solidity – Die Grundlagen verstehen 285

10.1 Was ist Solidity? ... 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

10.2 Elemente und Speicherbereiche eines Contracts ... 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

10.3 Verfügbare Datentypen ... 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

10.4 Zusätzliche Features von Solidity ... 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

10.5 Vererbungshierarchien von Smart Contracts erstellen ... 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

10.6 Libraries erstellen und verwenden ... 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

10.7 Zusammenfassung und Ausblick ... 321

11 Solidity – Details und Herausforderungen 325

11.1 Wichtige Details zu Funktionen ... 326
11.1.1 Polymorphismus richtig anwenden ... 326
11.1.2 Funktionen überladen ... 327
11.1.3 Ether empfangen mit der Fallback-Funktion ... 328

11.2 Gas verstehen und optimieren ... 330

11.3 Den richtigen Exception-Mechanismus wählen ... 333

11.4 Solidity mit Assembly erweitern ... 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

11.5 Leicht verständliche Contracts entwickeln ... 341

11.6 Updatefähige Contracts entwickeln ... 343
11.6.1 Separation Pattern anwenden ... 344
11.6.2 Proxy Pattern anwenden ... 349

11.7 Warum kein Zufallsgenerator sicher ist ... 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

11.8 Daten von außerhalb der Blockchain vertrauen ... 356

11.9 Zeitabhängigkeiten einbauen ... 357
11.9.1 Zeitabhängigkeiten über die Blockzeit prüfen ... 358
11.9.2 Externe Services verwenden ... 359

11.10 Zusammenfassung und Ausblick ... 359

12 Smart Contracts testen und debuggen 363

12.1 Contracts mit Remix testen ... 364
12.1.1 Unittests innerhalb von Remix ausführen ... 364
12.1.2 Remix-Tests in Continuous-Integration-Systeme integrieren ... 366

12.2 Debugging mit Remix ... 366

12.3 Einführung in das Truffle Framework ... 368
12.3.1 Das Truffle Framework aufsetzen ... 368
12.3.2 Migrationen in Truffle verwenden ... 370

12.4 Unittests mit Truffle implementieren ... 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

12.5 Integrationstests mit Truffle implementieren ... 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

12.6 Mit dem Truffle Framework debuggen ... 381

12.7 Zusammenfassung und Ausblick ... 382

13 Smart Contracts schützen und absichern 385

13.1 Allgemeine Sicherheitsempfehlungen ... 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
13.2 Ether in Contracts schmuggeln ... 388
13.3 Arithmetische Overflows und Underflows ... 391
13.4 Mit DelegateCalls den Zustand manipulieren ... 393
13.5 Reentrancy-Angriffe durchführen ... 396
13.6 Denial-of-Service-Angriffe durchführen ... 399
13.7 Gas-Siphoning-Angriffe beachten ... 401
13.8 Zusammenfassung und Ausblick ... 404

14 Smart Contracts deployen und managen 407

14.1 MetaMask einrichten und Accounts verwenden ... 408
14.2 Contracts mit Remix und MetaMask deployen ... 409
14.3 Contracts mit dem Truffle Framework deployen ... 411
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
14.4 Code auf Etherscan veröffentlichen ... 418
14.5 Einen eigenen Knoten aufsetzen und verwenden ... 420
14.6 Contracts manuell kompilieren, linken und deployen ... 422
14.7 Contracts nach dem Deployment managen ... 424
14.7.1 Contracts über Remix managen ... 425
14.7.2 Contracts über die Truffle-Konsole managen ... 425
14.8 Zusammenfassung und Ausblick ... 426

15 Standards, Libraries und Design Patterns 429

15.1 ERC-173 Contract Ownership Standard ... 429
15.2 ERC-165 Standard Interface Detection ... 432
15.3 ERC-20 Token Standard ... 436

15.4 ERC-721 Non-Fungible Token Standard 440

15.5 Die Libraries von OpenZeppelin nutzen 445

15.6 Design Patterns verwenden 446

15.6.1 Die Struktur des PubSub Patterns verstehen 446

15.6.2 Das PubSub Pattern implementieren 447

15.7 Zusammenfassung und Ausblick 450

16 Decentralized Applications entwickeln 453

16.1 Was ist eine Decentralized Application? 453

16.2 Der Entwicklungsprozess einer DApp 455

16.3 Das Backend Ihrer ersten DApp entwickeln 457

16.4 Das Frontend Ihrer ersten DApp entwickeln 460

16.5 Ihre erste DApp auf Swarm deployen 463

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

16.6 ENS-Domains für die eigene DApp einrichten 467

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

16.7 Zusammenfassung und Ausblick 471

17 Ihre erste DApp mit Drizzle zur DAO erweitern 475

17.1 Was ist Drizzle? 476

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

17.2 Was ist eine DAO, und was wird benötigt? 478

17.3 Die DApp, um einen Governance Contract erweitern 479

17.4 Das Frontend mit Drizzle und React umsetzen 480

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

17.5 Die Drizzle-DApp zentral oder dezentral deployen ... 492
17.5.1 Die DApp mit einem lokalen Server testen ... 493
17.5.2 Die Drizzle-DApp dezentral deployen ... 494

17.6 Zusammenfassung und Ausblick ... 494

18 Dezentrale Autonome Initiale Coin Offerings umsetzen 497

18.1 Was ist ein DAICO? ... 497
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

18.2 Die Contracts eines DAICO implementieren ... 501
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

18.3 Die passende DApp zu Ihrem DAICO entwerfen ... 510
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

18.4 Zusammenfassung und Ausblick ... 517

19 Supply Chains mit Smart Contracts ergänzen 519

19.1 Ideen, Visionen und Problemlösungen ... 520

19.2 Assets mit dem ERC-721 Token Standard tracken ... 521
19.2.1 Proof-of-Concept: der mobile Nadelmarkierer ... 521
19.2.2 Ein Asset-Token mit OpenZeppelin implementieren ... 522

19.3 web3j: die Web3 Java Ethereum DApp API ... 522
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

19.4 Ein Command Line Interface für Asset Tracker bauen ... 527
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

19.5 Zusammenfassung und Ausblick ... 532

20 Ausblick: Zukunftstechnologien 535

20.1 Vyper – Smart Contracts für jedermann? ... 535
20.1.1 Die Ziele von Vyper ... 536
20.1.2 Einschränkungen von Vyper ... 536
20.1.3 Die Syntax von Vyper ... 537

20.2 NEO – der chinesische Klon von Ethereum ... 539
20.2.1 Die Idee hinter NEO ... 539
20.2.2 Smart Contracts in NEO ... 540

20.3 EOS – der stärkste Wettbewerber zu Ethereum? ... 541
20.3.1 Die Vision von EOS ... 541
20.3.2 Die Architektur von EOS ... 542
20.3.3 Smart Contracts in EOS ... 543

20.4 Ripple – die Zukunft der Banken? ... 544
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

20.5 IOTA – das dezentrale Internet der Dinge ... 548
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

20.6 Zusammenfassung ... 554

Literaturverzeichnis ... 557
Index ... 559