

<b>Vorwort</b> .....	<b>XV</b>
<b>Glossar</b> .....	<b>XXV</b>
<b>1 Was ist Ethereum?</b> .....	<b>1</b>
Vergleich mit Bitcoin .....	1
Komponenten einer Blockchain .....	2
Die Geburt von Ethereum .....	3
Die vier Entwicklungsstufen von Ethereum .....	5
Ethereum: eine Allzweck-Blockchain .....	6
Die Komponenten von Ethereum .....	7
Weiterführende Literatur .....	7
Ethereum und Turing-Vollständigkeit .....	8
Turing-Vollständigkeit als »Feature« .....	9
Auswirkungen der Turing-Vollständigkeit .....	9
Von Allzweck-Blockchains zu dezentralisierten Anwendungen (Decentralized Applications, DApps) .....	10
Das dritte Internetzeitalter .....	11
Ethereums Entwicklungskultur .....	11
Warum Ethereum lernen? .....	12
Was Sie in diesem Buch lernen .....	13
<b>2 Ethereum-Grundlagen</b> .....	<b>15</b>
Ether-Währungseinheiten .....	15
Eine Ethereum-Wallet wählen .....	16
Kontrolle und Verantwortung .....	17
Einführung in MetaMask .....	19
Eine Wallet anlegen .....	20
Netzwerke wechseln .....	22
Test-Ether beschaffen .....	23

Ether aus MetaMask senden . . . . .	24
Die Transaktionshistorie einer Adresse untersuchen. . . . .	26
Der Weltcomputer . . . . .	28
Externally Owned Accounts (EOAs) und Kontrakte . . . . .	28
Ein einfacher Kontrakt: ein Test-Ether-Faucet . . . . .	29
Den Faucet-Kontrakt kompilieren . . . . .	32
Den Kontrakt in der Blockchain registrieren . . . . .	33
Interaktion mit dem Kontrakt. . . . .	35
Die Kontraktadresse in einem Block-Explorer ansehen. . . . .	36
Den Kontrakt finanzieren. . . . .	37
Abhebungen aus unserem Kontrakt. . . . .	38
Fazit . . . . .	41
<b>3 Ethereum-Clients . . . . .</b>	<b>43</b>
Ethereum-Netzwerke . . . . .	44
Soll ich einen Full Node betreiben? . . . . .	44
Vor- und Nachteile eines Full Node. . . . .	45
Vor- und Nachteile eines öffentlichen Testnetzwerks . . . . .	46
Vor- und Nachteile der Simulation einer lokalen Blockchain . . . . .	47
Einen Ethereum-Client betreiben . . . . .	47
Hardwareanforderungen für einen Full Node . . . . .	47
Softwareanforderungen für die Kompilierung und den Betrieb eines Clients (Node). . . . .	49
Parity . . . . .	50
Go-Ethereum (Geth) . . . . .	51
Die erste Synchronisation Ethereum-basierter Blockchains . . . . .	53
Geth oder Parity ausführen . . . . .	54
Die JSON-RPC-Schnittstelle . . . . .	54
Entfernte Ethereum-Clients . . . . .	57
Mobile (Smartphone) Wallets . . . . .	57
Browser-Wallets. . . . .	58
Fazit . . . . .	60
<b>4 Kryptografie . . . . .</b>	<b>61</b>
Schlüssel und Adressen. . . . .	61
Public-Key-Kryptografie und Kryptowährungen . . . . .	63
Private Schlüssel . . . . .	65
Generierung eines privaten Schlüssels aus einer Zufallszahl . . . . .	65
Öffentliche Schlüssel. . . . .	66
Kryptografie mit elliptischen Kurven . . . . .	68
Arithmetische Operationen bei elliptischen Kurven . . . . .	70

Einen öffentlichen Schlüssel generieren . . . . .	71
Bibliotheken für elliptische Kurven . . . . .	72
Kryptografische Hashfunktionen. . . . .	73
Ethereums kryptografische Hashfunktion: Keccak-256 . . . . .	74
Welche Hashfunktion nutze ich? . . . . .	75
Ethereum-Adressen . . . . .	76
Ethereum-Adressformate . . . . .	76
Inter Exchange Client Address Protocol . . . . .	77
Hex-Codierung mit Prüfsumme durch Großschreibung (EIP-55) . . . . .	78
Fazit . . . . .	80
<b>5 Wallets . . . . .</b>	<b>81</b>
Übersicht über die Wallet-Technologie. . . . .	81
Nichtdeterministische (zufallsbasierte) Wallets . . . . .	83
Deterministische (Seed-basierte) Wallets . . . . .	84
Hierarchisch-deterministische Wallets (BIP-32/BIP-44) . . . . .	85
Seeds und mnemonische Codes (BIP-39) . . . . .	86
Die Wallet-Best-Practices . . . . .	86
Mnemonische Codewörter (BIP-39) . . . . .	87
Eine HD-Wallet aus dem Seed-Wert erzeugen . . . . .	93
HD-Wallets (BIP-32) und Pfade (BIP-43/44) . . . . .	94
Fazit . . . . .	98
<b>6 Transaktionen . . . . .</b>	<b>99</b>
Die Struktur einer Transaktion . . . . .	99
Die Transaktions-Nonce . . . . .	100
Die Nonces nachhalten. . . . .	102
Lücken in Nonces, doppelte Nonces und Bestätigungen . . . . .	104
Nebenläufigkeit, Transaktionsursprung und Nonces. . . . .	104
Transaktionsgas. . . . .	105
Transaktionsempfänger . . . . .	107
Transaktionswert und -daten . . . . .	108
Mittel an EOAs und Kontrakte senden. . . . .	110
Nutzdaten an EOAs oder Kontrakte senden . . . . .	110
Spezielle Transaktion: Kontrakterzeugung . . . . .	112
Digitale Signaturen . . . . .	114
Elliptic Curve Digital Signature Algorithm . . . . .	115
Wie digitale Signaturen funktionieren . . . . .	115
Die Signatur verifizieren . . . . .	116
Die Mathematik hinter ECDSA . . . . .	116

Signieren von Transaktionen in der Praxis. . . . .	117
Erzeugen/Signieren einer Rohtransaktion . . . . .	118
Eine Rohtransaktion generieren mit EIP-155. . . . .	119
Der Signaturpräfixwert (v) und die Public-Key-Recovery. . . . .	120
Trennung von Signierung und Übertragung (Offlinesignierung) . . . . .	121
Propagation von Transaktionen . . . . .	123
Aufzeichnen in der Blockchain . . . . .	123
Multisignatur-Transaktionen (Multisig) . . . . .	124
Fazit . . . . .	125
<b>7 Smart Contracts und Solidity . . . . .</b>	<b>127</b>
Was ist ein Smart Contract? . . . . .	127
Lebenszyklus eines Smart Contract . . . . .	128
Einführung in Ethereum-Hochsprachen. . . . .	129
Smart Contracts mit Solidity entwickeln . . . . .	131
Eine Solidity-Version wählen . . . . .	132
Download und Installation . . . . .	132
Entwicklungsumgebung. . . . .	133
Ein einfaches Solidity-Programm entwickeln. . . . .	133
Kompilieren mit dem Solidity-Compiler (solc) . . . . .	134
Das Ethereum-Kontrakt-ABI . . . . .	134
Wahl einer Solidity-Compiler- und Sprachversion . . . . .	135
Programmieren mit Solidity . . . . .	136
Datentypen. . . . .	137
Vordefinierte globale Variablen und Funktionen. . . . .	138
Kontraktdefinition . . . . .	141
Funktionen. . . . .	141
Kontraktkonstruktor und selfdestruct . . . . .	143
Unser Faucet-Beispiel um einen Konstruktor und selfdestruct erweitern . . . . .	144
Funktionsmodifikatoren . . . . .	145
Kontraktvererbung. . . . .	147
Fehlerbehandlung (assert, require, revert) . . . . .	148
Events. . . . .	149
Andere Kontrakte aufrufen (send, call, callcode, delegatecall) . . . . .	152
Überlegungen zum Gasverbrauch. . . . .	158
Dynamisch dimensionierte Arrays vermeiden . . . . .	158
Aufrufe anderer Kontrakte vermeiden . . . . .	159
Die Gaskosten kalkulieren . . . . .	159
Fazit . . . . .	160

<b>8</b>	<b>Smart Contracts und Vyper</b> .....	<b>161</b>
	Sicherheitslücken und Vyper .....	161
	Vergleich mit Solidity .....	162
	Modifikatoren .....	162
	Klassenvererbung .....	163
	Inline-Assembler .....	164
	Funktionsüberladung .....	164
	Variablen-Typecasting .....	164
	Vor- und Nachbedingungen .....	166
	Dekoratoren .....	166
	Funktions- und Variablenanordnung .....	167
	Kompilierung .....	168
	Schutz vor Überlauffehlern auf Compilerenebene .....	169
	Daten lesen und schreiben .....	169
	Fazit .....	170
<b>9</b>	<b>Sicherheit von Smart Contracts</b> .....	<b>171</b>
	Best Practices .....	171
	Sicherheitsrisiken und Anti-Pattern .....	172
	Reentrancy-Angriffe .....	173
	Die Sicherheitslücke .....	173
	Präventive Techniken .....	176
	Reales Beispiel: DAO .....	177
	Arithmetischer Über-/Unterlauf .....	177
	Die Sicherheitslücke .....	177
	Präventive Techniken .....	180
	Reale Beispiele: PoWHC und Batch Transfer Overflow (CVE-2018–10299) .....	181
	Unerwartete Ether .....	181
	Die Sicherheitslücke .....	182
	Präventive Techniken .....	184
	Weitere Beispiele .....	185
	DELEGATECALL .....	185
	Die Sicherheitslücke .....	186
	Präventive Techniken .....	190
	Reales Beispiel: Parity Multisig Wallet (zweiter Hack) .....	190
	Standardsichtbarkeit .....	192
	Die Sicherheitslücke .....	192
	Präventive Techniken .....	192
	Reales Beispiel: Parity Multisig Wallet (erster Hack) .....	193

Entropie-Illusion . . . . .	194
Die Sicherheitslücke . . . . .	194
Präventive Techniken . . . . .	195
Reales Beispiel: PRNG-Kontrakte . . . . .	195
Referenzierung externer Kontrakte . . . . .	195
Die Sicherheitslücke . . . . .	195
Präventive Techniken . . . . .	198
Reales Beispiel: Reentrancy-Honeypot . . . . .	199
Kurze Adressen/Parameter . . . . .	200
Die Sicherheitslücke . . . . .	201
Präventive Techniken . . . . .	202
Ungeprüfte CALL-Rückgabewerte . . . . .	202
Die Sicherheitslücke . . . . .	202
Präventive Techniken . . . . .	203
Reales Beispiel: Etherpot und King of the Ether . . . . .	203
Race Conditions/Front Running . . . . .	204
Die Sicherheitslücke . . . . .	205
Präventive Techniken . . . . .	206
Reale Beispiele: ERC20 und Bancor . . . . .	207
Denial of Service (DoS) . . . . .	207
Die Sicherheitslücke . . . . .	207
Präventive Techniken . . . . .	209
Reales Beispiel: GovernMental . . . . .	210
Manipulation der Blockzeitstempel . . . . .	210
Die Sicherheitslücke . . . . .	210
Präventive Techniken . . . . .	211
Reales Beispiel: GovernMental . . . . .	212
Konstruktoren . . . . .	212
Die Sicherheitslücke . . . . .	212
Präventive Techniken . . . . .	213
Reales Beispiel: Rubixi . . . . .	213
Nicht initialisierte Zeiger auf Speicher . . . . .	213
Die Sicherheitslücke . . . . .	214
Präventive Techniken . . . . .	215
Reale Beispiele: OpenAddressLottery- und CryptoRoulette-Honeypot . . . . .	215
Fließkomma und Genauigkeit . . . . .	216
Die Sicherheitslücke . . . . .	216
Präventive Techniken . . . . .	217
Reales Beispiel: Ethstick . . . . .	217

Tx.Origin-Authentifizierung . . . . .	218
Die Sicherheitslücke . . . . .	218
Präventive Techniken . . . . .	219
Kontraktbibliotheken . . . . .	219
Fazit . . . . .	220
<b>10 Tokens . . . . .</b>	<b>221</b>
Wie Tokens genutzt werden . . . . .	221
Tokens und Fungibilität . . . . .	223
Kontrahentenrisiko . . . . .	223
Tokens und »Wesenhaftigkeit« . . . . .	224
Tokens nutzen: Utility oder Equity . . . . .	224
Ententest . . . . .	225
Utility-Tokens: Wer braucht sie? . . . . .	225
Tokens bei Ethereum. . . . .	227
Der ERC20-Token-Standard. . . . .	227
Ein eigenes ERC20-Token ausgeben . . . . .	231
Probleme mit ERC20-Tokens . . . . .	242
ERC223: ein vorgeschlagener Schnittstellenstandard für Token-Kontrakte . . . . .	243
ERC777: ein vorgeschlagener Schnittstellenstandard für Token-Kontrakte . . . . .	244
ERC721: Standard für nicht fungible Tokens (Deeds) . . . . .	247
Token-Standards nutzen . . . . .	248
Was sind Token-Standards? Was ist deren Zweck? . . . . .	248
Sollte man diese Standards nutzen? . . . . .	249
Sicherheit durch Reife. . . . .	250
Erweiterungen von Token-Interface-Standards. . . . .	250
Tokens und ICOs . . . . .	251
Fazit . . . . .	252
<b>11 Orakel . . . . .</b>	<b>253</b>
Warum Orakel benötigt werden . . . . .	253
Anwendungsfälle und Beispiele. . . . .	254
Entwurfsmuster für Orakel . . . . .	256
Datenauthentifizierung . . . . .	259
Rechnende Orakel . . . . .	260
Dezentralisierte Orakel . . . . .	262
Orakel-Clientschnittstellen in Solidity . . . . .	263
Fazit . . . . .	266

<b>12</b>	<b>Dezentralisierte Anwendungen (DApps)</b> .....	<b>267</b>
	Was ist eine DApp? .....	268
	Backend (Smart Contract) .....	269
	Frontend (Web-Nutzerschnittstelle) .....	269
	Datenspeicher .....	270
	Dezentralisierte Nachrichtenkommunikationsprotokolle .....	271
	Ein einfaches DApp-Beispiel: Auktions-DApp .....	271
	Auktions-DApp: Backend-Smart-Contracts .....	272
	Auktions-DApp: Frontend-Nutzerschnittstelle .....	275
	Weitere Dezentralisierung der Auktions-DApp .....	277
	Die Auktions-DApp in Swarm speichern .....	278
	Swarm vorbereiten .....	278
	Dateien in Swarm hochladen .....	279
	Der Ethereum-Nameservice (ENS) .....	281
	Geschichte des Ethereum-Nameservice .....	281
	Die ENS-Spezifikation .....	282
	Untere Schicht: Namenseigner und Resolver .....	282
	Mittlere Schicht: die .eth-Nodes .....	284
	Oberste Schicht: die Deeds .....	286
	Einen Namen registrieren .....	286
	Den ENS-Namen verwalten .....	290
	ENS-Resolver .....	291
	Einen Namen in einen Swarm-Hash (Inhalt) auflösen .....	292
	Von der App zur DApp .....	294
	Fazit .....	295
<b>13</b>	<b>Die Ethereum Virtual Machine</b> .....	<b>297</b>
	Was ist die EVM? .....	297
	Vergleich mit existierender Technologie .....	299
	Der EVM-Befehlssatz (Bytecode-Operationen) .....	299
	Ethereum-Zustand .....	302
	Solidity in EVM-Bytecode kompilieren .....	303
	Kontrakt-Deployment .....	306
	Disassemblierung des Bytecodes .....	307
	Turing-Vollständigkeit und Gas .....	312
	Gas .....	313
	Gasberechnung während der Ausführung .....	314
	Erwägungen zur Gasberechnung .....	314
	Gaskosten versus Gaspreis .....	315
	Block-Gaslimit .....	316
	Fazit .....	316



<b>14</b>	<b>Konsens</b> .....	<b>317</b>
	Konsens über Proof of Work .....	318
	Konsens über Proof of Stake (PoS) .....	318
	Ethash: Ethernets Proof-of-Work-Algorithmus .....	319
	Casper: Ethernets Proof-of-Stake-Algorithmus .....	320
	Konsensgrundsätze .....	321
	Kontroverse und Wettbewerb .....	321
	Fazit .....	322
<b>A</b>	<b>Ethereum-Fork-Historie</b> .....	<b>323</b>
<b>B</b>	<b>Ethereum-Standards</b> .....	<b>331</b>
<b>C</b>	<b>Ethereum EVM-Opcodes und Gasverbrauch</b> .....	<b>339</b>
<b>D</b>	<b>Entwicklungswerkzeuge, Frameworks und Bibliotheken</b> .....	<b>347</b>
<b>E</b>	<b>Einführung in web3.js</b> .....	<b>367</b>
<b>F</b>	<b>Kurzlink-Referenz</b> .....	<b>371</b>
	<b>Index</b> .....	<b>373</b>