

International Handbook of Blockchain Law

A Guide to Navigating Legal and Regulatory
Challenges of Blockchain Technology and Crypto Assets

Second Edition

EDITED BY
MATTHIAS ARTZT
THOMAS RICHTER



Wolters Kluwer

Summary of Contents

Editors	v
Contributors	vii
List of Abbreviations	xxxv
Preface	xliv
CHAPTER 1	
Understanding Blockchain	
<i>Jake van der Laan</i>	1
CHAPTER 2	
Regulatory Aspects of Blockchains	
<i>Thomas Richter</i>	91
CHAPTER 3	
Blockchain and Smart Contracts	
<i>Philip Trillmich, Matthias Goetz & Chris Ewing</i>	127
CHAPTER 4	
Blockchain and Data Privacy	
<i>Lothar Determann, William Long & Matthias Artzt</i>	161
CHAPTER 5	
Capital Markets	
<i>Michael Jünemann et al.</i>	243
CHAPTER 6	
Crypto Asset Regulation in Europe and in United Kingdom	
<i>Laura Douglas</i>	447

Summary of Contents

CHAPTER 7	
Regulation of Crypto Assets in the United States	
<i>Kayvan B. Sadeghi</i>	485
CHAPTER 8	
Blockchain and Intellectual Property	
<i>Andreas Holzwarth-Rochford</i>	517
CHAPTER 9	
Blockchain and Antitrust	
<i>Jay Modrall</i>	567
Bibliography	599
Table of Cases	625
Index	629

Table of Contents

Editors	v
Contributors	vii
List of Abbreviations	xxxv
Preface	xlv
CHAPTER 1	
Understanding Blockchain	
<i>Jake van der Laan</i>	1
§1.01 Introduction	1
§1.02 Core Concepts	2
[A] Record-Keeping	2
[B] How Computers Encode Data	4
[C] How Computers Store Large Numbers	5
[D] Computer Programming	6
[1] Variables	7
[2] Control Structures	7
[3] Data Structures	9
[a] The Array	9
[b] The Linked List	10
[4] Syntax	10
[5] From Program Code to Machine Code	11
[E] Computer Architecture	12
[1] Optimizing Computer Architecture	13
§1.03 The Mathematics of Blockchain	13
[A] The Hash Function	13
[B] The Cryptographic Hash Function	14
[1] The 'Always the Same' Property	14

	[2] The 'Digital Signature' Property	15
	[3] The 'No Reverse Engineering' Property	16
	[C] The Hash List	16
	[D] The Merkle Tree	17
	[E] Public-Private Key Encryption	18
§1.04	Database Concepts	22
	[A] Database Operations	23
	[B] A Transaction Log Distinguished from a Database	24
	[C] Database Transactions: The ACID Properties	24
§1.05	Networking Fundamentals	25
	[A] P2P Networks	26
	[1] The Consensus Problem	27
§1.06	Core Blockchain Functionality	27
	[A] Tying Blocks Together to Prevent Tampering	28
	[B] Adding a Time Cost	29
	[1] The Nonce	29
	[2] The Probabilities of Generating a Particular Hash Value	30
	[a] Randomness	30
	[b] Sample Space	30
	[3] Not Replacing Generated Events	30
	[4] A Cryptographic Hash Function Behaves Like a Random Variable	31
	[5] Requiring a Hash Value Below a Certain Value	33
	[C] The Proof of Work Process	33
	[1] A Note on Nonces	34
§1.07	Transactions on the Bitcoin Blockchain	35
	[A] Bitcoin Scripting	35
	[B] Entitlement Tracking	36
	[C] The UTXO Database	36
	[D] Wallets	36
§1.08	The Distributed Blockchain	38
	[A] The Bitcoin Network	38
	[B] The Memory Pool	40
	[C] Adding a Mined Block	40
	[D] The Double Spending Problem	41
	[E] The Mining Reward	41
	[F] Mining Pools	42
	[G] The 51% Attack	43
§1.09	Blockchain Forks	43
	[A] The Soft Fork	44
	[B] The Hard Fork	44
§1.10	Blockchain as a Platform	45
	[A] Ethereum	45
	[1] Gas: Limiting How Long a Program Can Run	46

	[2] An Ethereum Transaction	47
	[3] Ethereum Transaction Account Types	47
	[a] The EOA	48
	[b] The Contract Account	48
	[4] The Ethereum Virtual Machine	48
	[B] Other Platforms	48
§1.11	Smart Contracts	49
	[A] Programming Smart Contracts	49
	[B] Oracles	51
	[C] Decentralized Applications	51
§1.12	Tokens on the Blockchain	52
	[A] The ERC-20 Token Standard	53
	[B] The ERC-721 Token Standard (NFTs)	54
	[C] Crowdfunding on the Blockchain	54
	[1] Initial Coin Offerings	55
	[2] Token Generation Events	56
	[3] Initial Exchange Offerings	57
	[4] Security Token Offerings	57
	[5] A Note on Token Fungibility	58
§1.13	Bitcoin and Ethereum Governance	58
	[A] The Bitcoin Governance Process	58
	[1] Bitcoin Improvement Proposals	59
	[2] Segwit	60
	[3] Taproot	61
	[B] Ethereum Governance	62
	[C] Off-Chain and On-Chain Governance	62
§1.14	Moving Beyond Ethereum	62
	[A] Public (Permissionless) Blockchains	63
	[B] Private (Permissioned) Blockchains	63
	[C] Distributed Ledger Technologies	63
§1.15	Distributed System Consensus Theory	64
	[A] Synchronous Versus Asynchronous Processes	64
	[B] The FLP Impossibility Result	64
	[C] Consensus Mechanism Properties	65
	[1] The CAP Theorem	65
§1.16	Other DLT and Consensus Approaches	66
	[A] Byzantine Fault Tolerance (BFT)-Based Networks	66
	[1] Ripple	67
	[2] Stellar	68
	[B] Proof-of-Stake-Based Networks	69
	[1] Peercoin	69
	[C] Delegated Proof-of-Stake-Based Networks	71
	[1] BitShares	71
	[D] Proof-of-History-Based Networks	72

[E] Directed Acyclic Graph-Based Networks	72
[1] Graph Theory Basics	73
[2] Using a DAG as a Ledger	74
[3] IOTA	74
§1.17 Enterprise Blockchain Platforms	76
[A] Hyperledger	76
[1] Fabric	76
[2] Sawtooth	76
[3] Proof of Elapsed Time Consensus	77
[4] Transaction Families	78
§1.18 Scaling the Blockchain	78
[A] Sharding	78
[B] Sidechains	80
[1] The Lightning Network	81
[C] Multilayer Blockchains	82
[D] Rollups	83
[E] Cross-Chain Bridges	84
§1.19 Blockchain as a Service	84
§1.20 Decentralized Finance	84
[A] Introduction	84
[B] Decentralized Exchanges (DEXs)	86
[1] Initial DEX Offerings (IDOs)	87
[C] Staking	88
[D] Stablecoins	88
[1] Centralized Stablecoins	88
[2] Decentralized Stablecoins	88
[3] Central Bank Digital Currency	89
[E] Borrowing	89
Further Reading	90
CHAPTER 2	
Regulatory Aspects of Blockchains	
<i>Thomas Richter</i>	
§2.01 Blockchain as a Regulatory Subject	91
[A] The 'Legal Factor'	92
[B] The Meaning of 'Blockchain' in the Context of Regulation	93
[1] The Relevance of a Legal Definition	93
[2] Various Organizational Forms of Blockchains	94
[3] Legal Relevance of the Blockchain Protocol	97
[4] Inherent Characteristics of Blockchains and Their Legal Implications	99
[5] Summary	101
[C] The Meaning of 'Regulation' in the Context of Blockchain	101

[1] Which Kind of Regulation Can Be Relevant for Blockchains?	101
[2] Which Legal Subjects Can Be Relevant for Blockchains?	103
[3] 'Code Is Law' and 'Code as Law'	104
[4] Summary	104
§2.02 Regulatory Challenges	106
[A] Decentralization as Key Challenge	107
[B] Jurisdiction	107
[C] Anonymity	108
§2.03 Regulatory Concepts	109
[A] The Regulatory Principle of Technology Neutrality	110
[B] Regulating the Use Case Versus Regulating the Technology	112
[C] Regulatory Sandboxing	113
§2.04 Addressees of Regulation	114
[A] Addressing 'the Blockchain' as Subject of Regulation	115
[1] Legal Personality of the Blockchain	115
[2] The Malta Innovative Technology Arrangements and Services Act 2018	115
[3] Decentralized Autonomous Organizations as a New Corporate Form	116
[4] Summary	117
[B] Addressing Blockchain Actors as Subjects of Regulation	118
[1] Validation Nodes ('Nodes')	118
[2] Mining Nodes ('Miners')	120
[3] Blockchain Users	120
[4] Coders	120
[5] Intermediaries and Other Parties	121
§2.05 Financial Services Prudential Regulation	121
§2.06 Outlook: Blockchain, Regulation and AI	122
§2.07 Summary	125
Further Reading	125
CHAPTER 3	
Blockchain and Smart Contracts	
<i>Philip Trillmich, Matthias Goetz & Chris Ewing</i>	
§3.01 Terminology and Characteristics: What Is a Smart Contract?	127
[A] History of the Term 'Smart Contract'	127
[B] Technical Characteristics of a 'Smart Contract'	128
[C] Terminology	129
[1] The Meaning of the Term 'Smart' in Smart Contract	129
[2] What Does the Term 'Contract' Mean in Smart Contract?	129
[3] The Attempt to Develop a Definition	131
[D] Characteristics of 'Smart Contracts' as an Application of a Blockchain	132

Table of Contents

§3.02	Fields of Application	133
	[A] Legal Profession	133
	[B] Insurance Industry	134
	[C] Music Industry/Media Industry	136
	[D] Energy Supply Industry	138
	[E] Internet of Things	139
	[F] Sharing Economy	140
§3.03	Contract Law Issues	141
	[A] Applicable Law Versus 'Code in Law'	141
	[B] Formation of a Smart Contract	142
	[1] Contracting Parties	142
	[2] Form Requirements	143
	[3] Offer and Acceptance	146
	[C] Regulatory Issues and Smart Contracts	147
	[D] Consumer-Protection Laws and Smart Contracts	147
	[1] EU Directive 2011/83/EU	147
	[2] Unfair Standard Terms	148
	[E] Interpretation of Smart Contracts and Their Terms	150
	[F] Performance and Remedies	152
§3.04	Outlook	154
	[A] Benefits, Opportunities and Limitations of Smart Contracts	154
	[B] Use of Smart Contracts in the Future	157
	Further Reading	158
CHAPTER 4		
Blockchain and Data Privacy		
<i>Lothar Determann, William Long & Matthias Artzt</i>		
§4.01	Introduction and Executive Summary	161
§4.02	Personal Data in the Context of the GDPR	162
	[A] GDPR Definitions in the Context of a Blockchain	162
	[1] Personal Data	162
	[a] Public Keys	163
	[b] Transactional Data	165
	[2] Processing	167
	[B] Territorial Scope of the GDPR to the Blockchain	167
§4.03	Identifying Controllers and Processors in a Blockchain Environment under the GDPR	169
	[A] Definition of Controller and Processor	169
	[1] Controllers	169
	[2] Processors	170
	[B] Legal Status of Participants of a Blockchain Network	170
	[1] Static Number of Roles and Responsibilities	170
	[2] Miners	172
	[3] Nodes	174

Table of Contents

	[4] Wallets	
	[5] Users of a Blockchain	175
	[6] Developer of Smart Contracts	175
	[7] Oracles	177
	[8] Governance Bodies and Joint-/Co-controllership	178
	[C] Frictions Between Controllership and Obligations under the GDPR	179
§4.04	Legal Basis and Consent under the GDPR	180
	[A] Contractual Necessity	181
	[B] Consent	183
	[C] Legitimate Interest	183
	[D] Compliance with a Legal Obligation	185
	[E] Special Categories of Personal Data	186
§4.05	Security of Data Processing on a Blockchain in the Context of the GDPR	186
	[A] Personal Data Versus Anonymous Data	187
	[B] Particular Security Techniques	187
	[1] Encryption	187
	[2] Hashing	188
	[3] Multi-layered Blockchains	189
	[4] Storing Personal Data Off-Chain	189
	[C] Principles of Purpose Limitation and Data Minimization	190
	[1] Purpose Limitation	190
	[2] Data Minimization	192
	[3] Evaluation in the Light of the Principles of Purpose Limitation and Data Minimization: Multi-layered Blockchains Versus Off-Chain Storage	192
	[D] Recommendation for Security Measures in a Blockchain Environment	195
	[E] Implications in the Case of Security Breaches	196
§4.06	Data Subject Rights under the GDPR	196
	[A] How Do Data Subject Rights Apply to the Blockchain?	196
	[B] Right to Access Personal Data	197
	[C] Right to Rectification	198
	[D] Right to Erasure	199
	[1] What Does Erasure Mean?	199
	[2] At a Glance: The Techniques to Erase Data	200
	[3] Public Keys/Identifiers	202
§4.07	Accountability Principles under the GDPR	203
	[A] Lawfulness, Fairness and Transparency	204
	[B] Purpose Limitation, Data Minimization and Storage Limitation	205
	[C] Use of Data Privacy Impact Assessment	206
	[D] Data Protection by Design and Default	207
	[E] Record of Processing Activities	209

[F]	Appointment of a DPO	210
§4.08	International Data Transfers on a Blockchain under the GDPR	211
§4.09	Outlook	213
§4.10	Blockchain and US Privacy Law	214
[A]	US Privacy Law Versus EU Data Protection Regulation	214
[B]	Federal and State Law	220
[C]	Diverse Terminology	221
[D]	General and Specific US Privacy Laws Applied to Blockchain	222
[1]	General US Privacy Laws	223
[2]	Specific US Privacy Laws	225
[E]	Blockchain and CCPA	227
[1]	Scope of CCPA	229
[2]	Which Blockchain Participants Must Comply with CCPA?	230
[a]	Doing Business in California	230
[b]	Operating for Profit or Financial Benefit of Owners	231
[c]	Determining Means and Purposes of Processing	232
[d]	Thresholds	232
[i]	Annual Gross Revenue Threshold: USD 25 Million	232
[ii]	Number of Data Subjects Threshold: 100,000 Annually	233
[iii]	Revenue from Selling or Sharing Personal Information Threshold: 50%	233
[e]	Affiliates	234
[f]	Exempted Businesses	235
[3]	CCPA Compliance Obligations	236
[4]	Data Access and Deletion Rights	238
[5]	Sanctions and Remedies	239
	Further Reading	241
CHAPTER 5		
Capital Markets		
<i>Michael Jünemann et al.</i>		
§5.01	Capital Markets and Blockchain: International Chapter	243
[A]	What Makes a Token a Security?	243
[B]	Blockchain Finality	248
[C]	International Private Law	251
[D]	Special Requirements for Prospectuses	253
[E]	Regulatory Specifics for Organized Trade	255
§5.02	Capital Markets and Blockchain: Country Report – Austria	
<i>Johannes Frank & Philipp Kinsky</i>		
[A]	What Makes a Token a Security?	258
[B]	Blockchain Finality	259
[C]	Special Requirements for Prospectus	260

[D]	International Private Law	
[1]	Applicable Law: Contractual Obligations	261
[2]	Applicable Property Law	261
[E]	Regulatory Specifics for Organized Trade	263
§5.03	Capital Markets and Blockchain: Country Report – Belarus	264
<i>Klim Stashevsky & Mikhail Khodosevich</i>		
[A]	What Makes a Token a Security?	265
[B]	Blockchain Finality	265
[C]	International Private Law	266
[D]	Special Requirements for Prospectus	267
[E]	Regulatory Specifics for Organized Trade	268
§5.04	Capital Markets and Blockchain: Country Report – Estonia	270
<i>Kirsti Pent</i>		
[A]	What Makes a Token a Security?	274
[B]	Blockchain Finality	274
[C]	International Private Law	278
[D]	Special Requirements for Prospectus	279
[E]	Regulatory Specifics for Organized Trade	280
§5.05	Capital Markets and Blockchain: Country Report – Finland	281
<i>Mika Puurunen</i>		
[A]	What Makes a Token a Security?	282
[B]	Blockchain Finality	284
[C]	International Private Law	285
[D]	Special Requirements for Prospectus	287
[E]	Regulatory Specifics For Organized Trade	288
[F]	Conclusion	290
§5.06	Capital Markets and Blockchain: Country Report – France	291
<i>Bertrand Levy</i>		
[A]	What Makes a Token a Security?	292
[B]	Blockchain Finality	293
[C]	Private International Law and Blockchain	294
[D]	Special Requirements for Prospectus	296
[E]	Regulatory Specifics for Organized Trade	298
§5.07	Capital Markets and Blockchain: Country Report – Germany	
<i>Dr Michael Jünemann & Johannes Wirtz</i>		
[A]	What Makes a Token a Security?	299
[B]	Blockchain Finality	303
[C]	International Private Law	305
[D]	Special Requirements for Prospectus	307
[E]	Regulatory Specifics for Organized Trade	309
§5.08	Capital Markets and Blockchain: Country Report – Ireland	
<i>Niall Esler</i>		
[A]	What Makes a Token a Security?	312
[B]	Blockchain Finality	314

[C] International Private Law	315
[D] Special Requirements for Prospectus	316
[E] Regulatory Specifics for Organized Trade	318
§5.09 Capital Markets and Blockchain: Country Report – Italy	
<i>Giuseppe D'Agostino & Stefano Febbi</i>	319
[A] What Makes a Token a Security?	320
[B] Blockchain Finality	322
[C] International Private Law	324
[D] Special Requirements for Prospectus	327
[E] Regulatory Specifics for Organized Trade	329
§5.10 Capital Markets and Blockchain: Country Report – Liechtenstein	
<i>Johannes Dür</i>	334
[A] What Makes a Token a Security?	336
[B] Blockchain Finality	337
[C] International Private Law	340
[D] Special Requirements for a Prospectus	341
[E] Regulatory Specifics for Organized Trade	342
§5.11 Capital Markets and Blockchain: Country Report – Malta	
<i>Nicole Debrincat & Omar Zerafa</i>	343
[A] What Makes a Token a Security?	343
[1] Terminology	344
[2] Concept of Securities and Transferable Securities	344
[3] Regulatory Classification of Tokens	345
[B] Blockchain Finality	347
[1] The Blockchain	347
[2] European and Maltese Law	347
[C] Special Requirement for Prospectus	348
[1] The Prospects Regulation	348
[2] Prospectuses in Maltese Law	348
[3] The Prospectus	349
[D] International Private Law	350
[1] International Private Law in Malta	350
[2] Blockchain in Maltese Law	351
[3] Blockchain under English Law	351
[4] EU Law	352
[E] Regulatory Specifics for Organized Trade	353
§5.12 Capital Markets and Blockchain: Country Report – Poland	
<i>Aleksandra Widziewicz & Piotr Lesiński</i>	354
[A] What Makes a Token a Security?	354
[B] Blockchain Finality	359
[C] International Private Law	359
[D] Special Requirements for Prospectus	361
[E] Regulatory Specifics for Organized Trade	361

§5.13 Capital Markets and Blockchain: Country Report – Spain	
<i>Jose Luis Lorente Howell</i>	
[A] What Makes a Token a Security?	363
[B] Blockchain Finality	363
[C] Special Requirements for Prospectuses	367
[D] International Private Law	368
[E] Considerations in Relation to Organized Trade and Platforms	369
§5.14 Capital Markets and Blockchain: Country Report – Switzerland	
<i>Olivier Favre & Fabio Elsener</i>	
[A] What Makes a Token a Security?	371
[B] Blockchain Finality	372
[1] Payment Tokens and Utility Tokens Without Claims	373
[2] Asset Tokens and Utility Tokens Conferring Claims Issued in the Form of DLT Rights	373
[3] Asset Tokens and Utility Tokens Conferring Claims Issued in Other Form Than DLT Rights	374
[C] International Private Law	375
[D] Special Requirements for Prospectuses	376
[E] Regulatory Specifics for Organized Trade	377
[1] Payment Tokens and Utility Tokens	378
[2] Asset Tokens	378
[3] Travel Rules	380
§5.15 Capital Markets and Blockchain: Country Report – British Virgin Islands	
<i>Sara Hall</i>	380
[A] Introduction	380
[B] What Makes a Token a Security?	381
[C] Blockchain Finality	385
[D] International Private Law	385
[E] Special Requirements for Prospectus	387
[F] Regulatory Specifics for Organized Trade	387
[1] Virtual Assets	388
[2] Definition of a Virtual Assets Service	388
[3] On the Horizon	389
§5.16 Capital Markets and Blockchain: Country Report – Canada	
<i>Daniel Fuke & Mike Stephens</i>	390
[A] What Makes a Token a Security	391
[B] Crypto Winter(s)	393
[C] Special Requirements for Prospectus	394
[D] Regulatory Specifics for Organized Trade	394
[1] Regulatory Environment and Oversight	394
[2] Enforcement	395
[3] Digital Asset Trading	396
[4] Crypto Dealer Registration	398

[5] Money Services Businesses and Anti-money Laundering Laws	398
[E] Conclusion	399
§5.17 Capital Markets and Blockchain: Country Report – United States <i>James Gatto</i>	400
[A] What Makes a Token a Security?	400
[1] US SEC	400
[2] The Commodities Futures Trading Commission	402
[3] The Financial Crimes Enforcement Network	403
[4] FINRA	405
[5] The IRS	405
[6] State Laws	405
§5.18 Capital Markets and Blockchain: Country Report – Japan <i>Takashi Itokawa</i>	406
[A] What Makes a Token a Security?	407
[B] Blockchain Finality	407
[1] Corporate Bonds	408
[2] Securitization of Real Estate	408
[a] A Trust Issuing Beneficiary Certificates with Stipulating in the Trust Agreement That the Beneficiary Certificates Shall Not Be Issued	408
[b] Investor's Right in the Silent Partnership (Tokumei Kumiai Shusshi Mochibun Kenri)	409
[c] Article 11-2 of the Act on Strengthening Industrial Competitiveness	409
[C] Special Requirements for Prospectuses	410
[1] Real Estate Security Tokens	410
[2] Corporate Bond Tokens	411
[3] Self-regulation Guidelines	411
[a] Technologies Used for Tokenized Securities	411
[b] Risks of Technology Used in Tokenized Securities	411
[c] The Application of Regulations and the Like According to the Type of Securities and the Transaction	412
[D] Regulatory Specifics for Organized Trade	412
[1] Reclassification as Securities under the FIEA	412
[2] Limitations for Improving the Liquidity	413
[3] Other Regulations under the FIEA	413
§5.19 Capital Markets and Blockchain: Country Report – Singapore <i>Kim Kit Ow & Gerard Ng</i>	414
[A] What Makes a Token a Security?	414
[B] Blockchain Finality	416
[C] Private International Law	418
[D] Special Requirements for Prospectus	419

[E] Regulatory Specifics for Organized Trade	422
[F] Conclusion	424
§5.20 Capital Markets and Blockchain: Country Report – South Korea <i>Jongbaek Park</i>	424
[A] What Makes a Token a Security?	424
[B] Blockchain Finality	424
[C] Private International Law	427
[D] Special Requirements for Prospectus	429
[E] Regulatory Specifics for Organized Trade	431
§5.21 Capital Markets and Blockchain: Country Report – United Arab Emirates <i>Gregory Man & Jessica White</i>	433
[A] What Makes a Token a Security?	436
[1] Federal Jurisdiction	436
[2] DIFC Jurisdiction	437
[3] ADGM Jurisdiction	438
[B] Blockchain Finality	438
[1] DIFC Jurisdiction	439
[2] ADGM Jurisdiction	439
[C] International Private Law	440
[D] Special Requirements for Prospectus	440
[1] DIFC Jurisdiction	442
[2] ADGM Jurisdiction	442
[E] Regulatory Specifics for Organized Trade	443
[1] DIFC Jurisdiction	444
[2] ADGM Jurisdiction	445
CHAPTER 6	
Crypto Asset Regulation in Europe and in United Kingdom <i>Laura Douglas</i>	447
§6.01 The EU Markets in Crypto Assets Regulation	447
[A] Scope of MiCA	448
[1] Types of Crypto Assets Regulated under MiCA	448
[a] Definition of 'Crypto Assets'	448
[b] Exclusions from Scope	449
[c] Categories of Crypto Assets and Stablecoins	450
[2] Activities and Persons Regulated under MiCA	450
[B] Issuance and Offering of Crypto Assets (Other Than EMTs and ARTs)	450
[1] Scope: Offerors and Persons Seeking Admission to Trading	451
[2] Requirements for Public Offers or Admission to Trading of Crypto Assets (Other Than EMTs and ARTs)	451
[a] Overview	451

[b]	Exemptions	452
[c]	Form and Content of the White Paper	453
[d]	Notification and Publication of the White Paper	453
[e]	Marketing Communications	454
[f]	Changes to White Papers and Marketing Communications	454
[g]	Liability for Content of the White Paper	455
[h]	Application to Crypto Assets Already Traded on EU Venues	455
[i]	General Obligations on Issuers and Offerors	456
[j]	Withdrawal and Reimbursement Rights	456
[C]	Regulation of Stablecoins: EMTs and ARTs	457
[1]	Asset-Referenced Tokens (ARTs)	457
[a]	Requirements for Public Offering or Admission of ARTs to Trading	457
[b]	Authorisation Regime for ART Issuers	457
[c]	ART Redemption Rights	458
[d]	ART White Paper and Marketing Requirements	458
[e]	Reporting Requirements and Restrictions on Issuance of Widely Used ARTs	458
[f]	Significant ARTs	459
[2]	E-Money Tokens (EMTs)	460
[a]	Requirements for Public Offering or Admission of EMTs to Trading	460
[b]	Requirements for EMTs and Issuers	460
[c]	EMT White Paper and Marketing Requirements	461
[d]	Significant EMTs	461
[D]	Regulation of Crypto Asset Service Providers	462
[1]	Regulated Crypto Asset Services	462
[2]	Authorisation Regime for CASPs	464
[3]	Provision of Crypto Asset Services by Licensed Financial Entities	465
[4]	Cross-Border Provision of Crypto Asset Services	466
[5]	Ongoing Obligations for Provision of Crypto Asset Services	466
[a]	Provision of Crypto Asset Custody Services	466
[b]	Operation of a Trading Platform for Crypto Assets	467
[c]	Exchange of Crypto Assets for Funds or for Other Crypto Assets	468
[d]	Execution of Orders for Crypto Assets on Behalf of Clients	468
[e]	Placing of Crypto Assets	468
[f]	Reception and Transmission of Orders for Crypto Assets on Behalf of Clients	468

[g]	Advising on and Providing Portfolio Management of Crypto Assets	469
[h]	Providing Transfer Services for Crypto Assets on Behalf of Clients	469
[E]	Market Abuse Regime for Crypto Assets	469
[1]	Inside Information Relating to Crypto Assets	470
[2]	Obligation to Disclose Inside Information to the Public	470
[3]	Prohibitions Relating to Market Abuse	471
[a]	Prohibition on Insider Dealing	471
[b]	Unlawful Disclosure of Inside Information	471
[c]	Prohibition on Market Manipulation	471
[4]	Prevention and Detection of Market Abuse	472
[F]	Status and Future Developments	472
§6.02	AML Regulation Relating to Crypto Assets in the EU	473
[A]	The MLD5 Registration Regime for Virtual Currency Service Providers	473
[1]	Virtual Currencies	473
[2]	Requirements on Virtual Currency Exchange Providers and Custodians	474
[B]	The EU 'Travel Rule' for Crypto Asset Transfers	474
[1]	Scope and Definitions	474
[2]	Travel Rule Requirements for Crypto Asset Transfers	475
§6.03	UK Regulation of Crypto Assets	476
[A]	Scope of UK Regulation of Crypto Assets	476
[1]	Regulatory Characterisation of Crypto Assets	476
[2]	Extending the Authorisation Regime under FSMA to Stablecoins and Other Crypto Assets	477
[a]	Regulation of Fiat-Backed Stablecoins	478
[b]	Regulation of Other Crypto Assets	479
[B]	Financial Promotions Relating to Crypto Assets	479
[1]	The UK Financial Promotion Regime	479
[2]	Extension of the Regime to Crypto Asset Financial Promotions	480
[3]	FCA Rules and Guidance for Crypto Asset Financial Promotions	480
§6.04	AML Regulation Relating to Crypto Assets in the UK	481
[A]	Registration Regime for Crypto Asset Service Providers	481
[1]	Scope and Definitions	481
[2]	Requirements on Crypto Asset Exchange Providers and Custodian Wallet Providers	482
[B]	The UK 'Travel Rule' for Crypto Asset Transfers	482
[1]	Scope and Definitions	482
[2]	UK Travel Rule Requirements for Crypto Asset Transfers	482

CHAPTER 7

Regulation of Crypto Assets in the United States

Kayvan B. Sadeghi

§7.01	Introduction	485
§7.02	Federal Securities Laws	485
	[A] Defining "Securities"	487
	[1] "Investment Contracts" and the <i>Howey</i> Test	488
	[a] "Investment of Money"	489
	[b] "Common Enterprise"	489
	[i] Continuing Contractual Obligation	490
	[ii] Horizontal Commonality	490
	[iii] Vertical Commonality	491
	[c] "Reasonable Expectation of Profit"	491
	[d] "Efforts of Others"	492
	[2] "Notes" and the <i>Reves</i> Test	492
	[B] The SEC's Approach to Digital Assets	493
	[1] Issuers and Token Offerings	495
	[a] Telegram and Kik	497
	[b] <i>SEC v. Ripple Labs</i>	497
	[c] <i>SEC v. Terraform Labs</i>	499
	[2] Trading Platforms	500
	[a] Exchanges	500
	[b] Clearing Agencies	501
	[3] Brokers & Dealers	501
	[a] Brokers	502
	[b] Dealers	502
	[4] Crypto Asset Lending	504
§7.03	Federal Commodities Laws	505
	[A] Scope of CFTC's Authority	505
	[B] Registration Requirements	506
	[C] Enforcement Actions	507
§7.04	Money Services Businesses & Money Transmitting	509
§7.05	Sanctions	513
§7.06	State Regulations	515
	[A] New York	515
	[B] California	516

CHAPTER 8

Blockchain and Intellectual Property

Andreas Holzwarth-Rochford

§8.01	Introduction	517
§8.02	Trademark	517
	[A] Definition	519
	[B] Duration	519

	[C] Territory	519
	[D] Goods and Services	520
	[E] Types of Trademarks	520
	[F] Registration Process	521
	[G] Post-registration	521
	[H] Enforcement	522
	[I] Examples and Practical Hints	523
§8.03	Designs	523
	[A] Definition	525
	[B] Duration/Territory	525
	[C] Registration/Invalidity Procedures/Protection Requirements	525
	[D] Enforcement	526
	[E] Examples and Practical Hints	528
§8.04	Copyright	528
	[A] Definition	530
	[B] Rights Given by Copyright, Duration and Practical Hints	530
§8.05	Open Source	530
§8.06	Patents/Utility Models	531
	[A] Definition/Content of Patent Application	533
	[B] Rights Provided by a Patent	533
	[C] Territory	534
	[1] European Patents/Newly Implemented European Unitary Patent	535
	[2] Further Regional Patents European Patents/Planned European Unitary Patent	538
	[D] Protection Requirements EPO	538
	[1] Novelty/Grace Period	538
	[2] Priority	539
	[3] International Patent Applications	540
	[4] Exclusion from Patentability	541
	[5] Examples of Blockchain-Related EP Patents	542
	[6] Opposition Procedure	545
	[7] Enablement	546
	[8] Invalidity Proceedings at National Courts/UPC for EP Patents and Double Patenting	548
	[E] Protection Requirements USPTO: Examples of Blockchain-Related US Patents	549
	[F] Practical Hints Protection by Patents	552
	[1] Strategic Considerations	552
	[2] Observation/Surveillance of Third-Party Rights	553
	[3] Transfer of Rights	554
	[a] German Act on Employee Inventions	555
	[G] Utility Models	556
	[1] Background	556

Table of Contents

	[2] Registration Procedure: Branching Off	557
	[3] Protection Requirements/Duration	558
	[4] Protectable Subject Matter	559
	[5] Summary of Pros and Cons of Utility Models	559
§8.07	Trade Secret	560
	[A] General Background	560
	[B] Legal Basis: Definition	561
	[C] Practical Aspects	563
	Further Reading	565
CHAPTER 9		
Blockchain and Antitrust		
<i>Jay Modrall</i>		
§9.01	Introduction	567
§9.02	Anticompetitive Agreements, Decisions and Concerted Practices	567
	[A] Introduction to Concepts	568
	[B] Horizontal Agreements and Blockchain	568
	[1] Information Exchange	570
	[2] Standardization	571
	[C] Vertical Agreements and Blockchain	574
	[D] Appropriate Safeguards	575
§9.03	Abuse of Dominance	577
	[A] Introduction of Concepts	577
	[B] Big Data	577
	[1] Mandating Data Access	579
	[2] Policing Data Sharing and Pooling	580
	[C] Online Platforms	583
	[1] Most Favored Nations	583
	[2] Multi-homing	584
	[3] Interoperability	584
	[4] Transparency	584
	[5] Leveraging	585
	[6] Self-preferencing	585
	[D] Article 102 TFEU and Blockchain	585
	[E] Appropriate Safeguards	586
§9.04	Merger Control	587
	[A] Introduction to Concepts	587
	[B] Full-Function Joint Ventures	587
	[C] Gun-Jumping	589
	[D] Merger Control and Blockchain	589
	[1] Horizontal Assessment: Combining Datasets	590
	[2] Vertical Concerns: The Role of Data in Foreclosure Assessment	591
		592

Table of Contents

	[E] Appropriate Safeguards	
	Further Reading	596
	Bibliography	596
	Table of Cases	599
	Index	625
		629

Figure 1.4 An Example of an If ... Else Statement

```

if (age < 20 and age > 12)
{
    print "this person is a teenager"
}
else
{
    print "this person is not a teenager"
}

```

Another often used control structure is the *while loop* which allows the repeated execution of a section of logic for as long as a condition is *true*. Here is a *while loop* example which displays an integer variable as long as it is not 11, as it is being incremented by one, starting from 1 (Figure 1.5).

Figure 1.5 An Example of a While Statement

```

variable = 1
while (variable < 11)
{
    print variable
    variable = variable + 1
}

```

Control structures can be used to build ever more complex functionality from very basic building blocks,²⁵ which can then be stored as *functions* or routines and re-used in other computer programs. For example, it is very easy to build a multiplication function (seeking to multiply *x* and *y*) given an existing simple addition function by simply adding *y* to *y*, *x* number of times. Once such a multiplication function is created and can be accessed as a function (or as a *callable* set of code), it can then be used to create another yet more intricate function to calculate the *power* of a number, and so on.

All programming languages have developed extensive sets of such complex functions, procedures and reusable code called *libraries*. Libraries can be accessed, or called, by any computer program written in the applicable programming language without having to re-invent a large set of functions and procedures. This is one of the

25. Which build on the very basic capacities of the core circuitry of a computer's Central Processing Unit or CPU. A CPU has only a limited set of core functionalities, including addition and multiplication. The power of a CPU is in being able to do billions of these basic operations per second.

most important features of modern programming - the ability to re-use and build on existing code.

In many cases, sets of libraries for a particular program also include functionality known as an *Application Program Interface* or API, which allows other programming languages, and in some cases devices with programmatic functionality, to interact with programs written in the language for which the API is available.²⁶

[3] Data Structures

Data structures are methods of storing data in computer memory in a particular way to enable certain capabilities for manipulating that data, which would be (much) more difficult or not possible without that data structure.

In many instances, data structures incorporate or enable certain relationships between data elements which then facilitate indexing, sorting, editing, or other functionality, including certain kinds of calculations. Data structures in many cases also help with the efficient access and manipulation of data stored in memory.

[a] The Array

An *array* is an example of a simple data structure, which is simply a list of items in order. Here is an example of an array containing the months of the year (Figure 1.6).

Figure 1.6 An Array of the Months of the Year

Jan.	Feb.	Mar.	Apr.	May	Jun.	Jul.	Aug.	Sep.	Oct.	Nov.	Dec.
0	1	2	3	4	5	6	7	8	9	10	11

The location of an element in an array is referred to as its *index*, displayed above below each month.²⁷

An array is useful if one wants to relate a specific location in that array with the ranked order of the data items stored. In the above example, the months of the year are thus listed in the order in which they occur.

Another example is the list of ingredients in packaged food, usually found on its label. This array is usually required to be listed in the order of prevalence of the ingredients listed. The ingredient with the highest percentage of content goes first, the ingredient with the next highest goes next, and so on. The structure of the array thus embodies a sorted list without needing to do anything further.

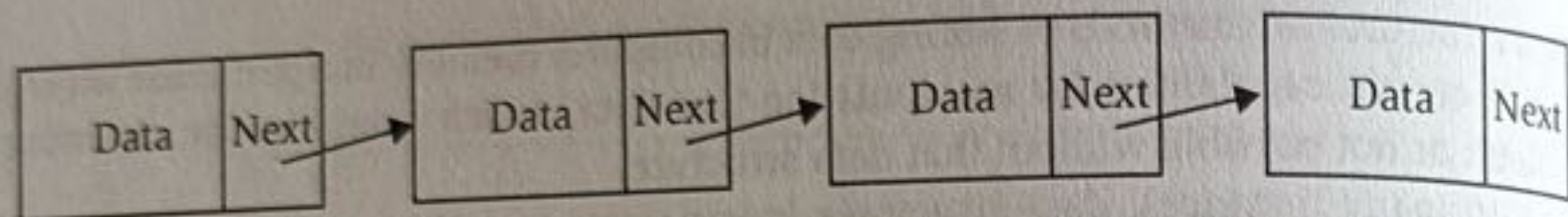
26. See, for instance, the API for the Java programming language at: <https://docs.oracle.com/en/java/javase/21/index.html> (accessed 10 Jan. 2024).

27. Note that many programming languages start the index at 0, which is perhaps a bit counterintuitive but makes it easier for the computer, which deems 0 to be just as good a number as 1.

[b] The Linked List

The *linked list* is another example of a data structure. A linked list is a set of data objects where each data object, in addition to storing some data value, keeps track of the location of the next item in the list by way of a pointer (usually pointing to the memory address of the next item in the list) (Figure 1.7).

Figure 1.7 A Linked List



Linked lists are very useful for maintaining an ordered list because it is very easy to insert or remove elements from the list. For example, to insert a new element in an ordered list, one only needs to update the pointer of the previously ordered item in the list. Compare this to an array where one would need to shift all subsequent list members over one location to the right.

The core idea behind the linked list (tying data elements together with a pointer) is also used in a slightly different way in a blockchain.

[4] Syntax

The syntax of a programming language is the set of rules for translating a logical process or workflow into a sequence of instructions which the computer can understand and execute.

When writing a program in a specific language, it is very important to fully understand that language's syntax. Any misunderstanding of the syntax can lead to errors in the instructions which may go unnoticed in the final implementation of a program until a set of conditions arises which causes the execution of the program to go awry. These types of errors are often referred to as 'bugs', and no computer program is without them.²⁸

It should be noted at this point that bugs or errors in how computer programs run can occur not only from syntax errors in the code but also from such things as:

- errors in the operating system software of the computer, for instance, how the operating manages its memory;
- errors in the hardware configuration of the computer, for instance, how data traffic is managed on the motherboard of the computer;

28. With the exception perhaps of the code written to power the lunar missions in the late 1960s and early 1970s.

- logical errors in the code which fail to contemplate the full range of possible values for input variables to programmatic control structures and may thus cause unintended program flow and/or memory overflows²⁹ as a result.

No computer program of any complexity is without some bugs.

[5] From Program Code to Machine Code

A computer language, such as Java, Python or Visual Basic, is actually a large computer program written to make programming of computers easier for humans. These higher-level languages use simple English words to represent their syntax, and many have complex integrated editing tools with a great deal of features.³⁰

The product of these high-level programming languages is a relatively easy-to-read text file, compliant with the structural formatting and syntax requirements of the specific programming language used. This file is not directly usable by the computer for execution, however, and must first be converted to *machine code* interpretable by the Central Processing Unit (CPU)³¹ of the computer on which the code is to be executed.

The type of machine code required depends on the architecture of the CPU in question. Machine code for an Intel processor³² is different from that for an ARM processor,³³ for example. The required machine code is determined by the *instruction set* for the CPU, which can be thought of as the complete list of available instructions which can be issued to the CPU by a computer program.

The program which translates the program language specific text file into machine code is called a *compiler*.³⁴ The product of a compiler is a long string of zeroes and ones called a *bitstream*, which in the case of a computer program is usually referred to as a *binary*.

In some cases, an intermediary code interpreter called a *virtual machine*, *process virtual machine*, or *runtime environment* sits between the programming language and the CPU.

Such a virtual machine is used to create the ability to execute a particular program on any CPU for which a virtual machine exists.³⁵ The Java programming language uses the *Java Virtual Machine*, or JVM, exactly for this purpose, and it is one of the reasons why this language is so popular: it can run on many different types of hardware.

29. For instance, by placing a value in a memory location which is too large for the reserved space, which can then result in overwriting the next memory location.

30. Referred to as Integrated Development Environments, or IDEs.

31. The 'brains' of the computer, which contain hardwired circuitry which can execute basic computer instructions at a very high rate.

32. Found in most desktop computers.

33. Advanced RISC Machines, which is used in many mobile and embedded devices.

34. This is a dramatic oversimplification of what a compiler does, but it is sufficient for our limited purposes.

35. A virtual machine thus allows a programming language to become *multi-platform*.

In the case of the JVM, the compiler does not generate binary code but rather a *bytecode* file, which the JVM then converts to machine code usable by the specific hardware being used.

In many cases, the binary or bytecode is transmitted to a computer or virtual machine for execution using hex encoding. As we will see, this is also how smart contracts are encoded and transmitted to the blockchain.

[E] Computer Architecture

Given all the above functionalities, a computer is physically organized with the following key components and associated functionalities:

- (1) The Central Processing Unit (CPU), which executes instructions stored in memory.
- (2) Memory, primarily of two types: Random-Access Memory (RAM), which is volatile³⁶ memory used for temporary storage of data and program instructions that the CPU is currently using, and secondary memory, which is non-volatile memory used for long-term storage, such as a hard drive.
- (3) Registers, which are small, high-speed storage locations within the CPU used for temporarily holding data during processing.
- (4) The Instruction Set Architecture (ISA), which is the set of commands or instructions that a CPU understands and can execute. These are pre-programmed into the CPU.
- (5) Cache Memory, which is high-speed memory located between the CPU and RAM and which is used to store frequently accessed data and instructions for faster retrieval.
- (6) The Bus System, which is a set of communication pathways that transfer data between components. They include the address bus (identifying memory locations), data bus (transferring actual data), and control bus (managing the operation of components).
- (7) The Motherboard, which is the main circuit board that connects and facilitates communication between the CPU, memory, storage, and peripheral devices.
- (8) A System Clock, which synchronizes the operations of the CPU and other components, ensuring that computational work is properly coordinated and sequenced.

In addition to these, a computer will also have a set of input devices, such as a keyboard, mouse, touch screen, or scanner, which allow users to interact with the computer by providing data or commands, as well as output devices, such as a monitor, printer, and speakers.

36. 'Volatile' in this context means that the storage is not persistent if the storage medium loses power.

[I] Optimizing Computer Architecture

Over the last number of decades, there has been a great deal of innovation in making computer architectures perform at ever faster rates. Among these, the most effective improvements have resulted from technologies which enable simultaneous processing of multiple instructions by the computer. These include:

- Pipelining or Parallel Processing, which usually involves breaking down the execution of instructions into multiple stages, allowing the CPU to work on different instructions at the same time.
- Multi-Core Processors, which is the installation of multiple integrated processor cores on a single computer chip. Such chips are designed to optimize parallel processing.

These parallel processing features are integral to the capabilities of today's computing hardware.

§1.03 THE MATHEMATICS OF BLOCKCHAIN

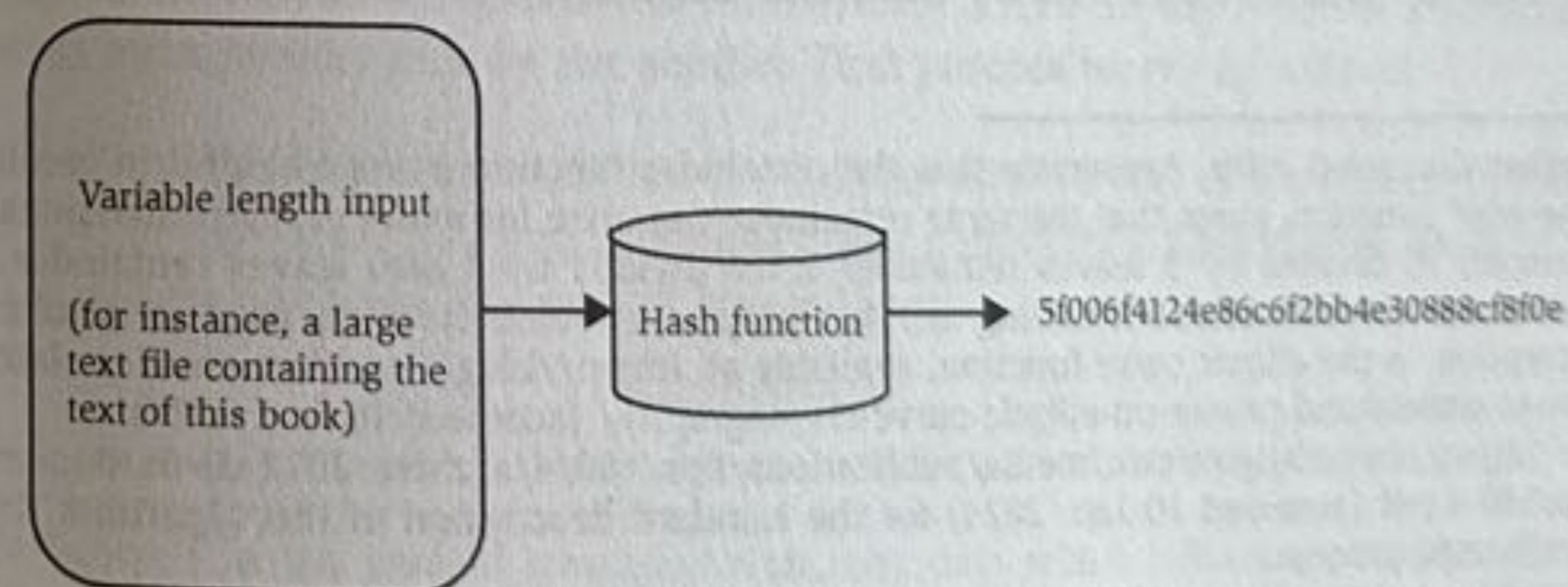
[A] The Hash Function

A function is a mathematical recipe for turning an input into a desired output based on a clearly articulated formula or algorithm. For example, the addition function ' $C = A + B$ ' takes the variables A and B as inputs and generates the desired output C by adding A and B together.

One group of functions relevant to understanding blockchain is the set of *hash functions*. Hash functions are used to convert a usually large input of data into a much smaller, fixed-length output to create a unique 'signature' of the larger input.

A hash function thus maps the input data to a numerical key. This key is usually referred to as the *hash value* or *hash digest*, or just the *hash* (Figure 1.8).

Figure 1.8 A Hash Function



Most hash functions work by taking the input data, chopping it up into smaller sections and then applying certain functions to those smaller chunks. The division and

remainder - or *modulo* - function is often used in this process, which gives a hash function a 'one-way' property: it is not possible to recreate the original input from the eventual hash generated by the function.³⁷

These hash functions then shuffle and/or combine the outputs from these smaller chunks in a predefined manner and reapply the modulo, or some variant, again. This process is continued until a single output of the desired length is achieved.

The output of a hash function is often written in hex. For example, the very popular hashing algorithm SHA256³⁸ (which is used in the Bitcoin blockchain) will output a 64-bit hash digest which might look like this:³⁹

5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8

Note that this hex value is the following very large number in base-10:

42,758,200,014,260,304,871,795,218,010,112,217,441,437,394,933,425,264,440,882,844,866,153,524,052,696⁴⁰

Hash functions are used in many applications, including storing passwords, digital certificates to support encryption, and indexing of large data sets. More refined hashing functions are used to assist in ensuring that data is not corrupted or has not been changed.

[B] The Cryptographic Hash Function

One specific type of hash function is particularly relevant to our understanding of blockchain: the *cryptographic hash function*. Like a regular hash function, the cryptographic hash function takes a data input of variable length and generates a fixed-length hash digest as its output.

A cryptographic hash function has some specific and very useful properties which the blockchain takes advantage of.

[1] The 'Always the Same' Property

A cryptographic hash function will always generate the same hash value for the same input. This is usually described by computer scientists and mathematicians as the

37. Further discussed *infra*. Appreciate that the remainder function is very helpful in creating a 'one-way' function, given that the same remainder can arise for many different divisions. For example, 10 divided by 3 leaves remainder 1. 16 divided by 5 also leaves remainder 1. A specialized implementation building on this property, which is often used in public key encryption, is the *elliptic curve* function, available at: <https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/> (accessed 10 Jan. 2024).

38. See <https://csrc.nist.gov/csrc/media/publications/fips/180/4/archive/2012-03-06/documents/fips180-4.pdf> (accessed 10 Jan. 2024) for the standard description of this algorithm. It is a complicated process.

39. Available at: <https://www.fileformat.info/tool/hash.htm> (accessed 10 Jan. 2024).

40. This is a 78-digit number and one of the 2^{256} possible outputs from the SHA256 hashing algorithm. Note that the number of digits of that number can be determined by calculating $n \log 2$, in this case $256 \log 2$ which equals 78.

'deterministic' property. A function which is non-deterministic will generate different outputs because a certain amount of randomness exists in how the function works. An example of a non-deterministic function is the rolling of a set of dice.

[2] The 'Digital Signature' Property

A cryptographic hash function also uses a formula which ensures (with an extremely high level of confidence⁴¹) that the hash digest will be *unique*. A hash digest thus becomes a *digital signature* of the input data and can be used to determine whether an input has been modified or not.

In the use of a cryptographic hash function, the smallest possible change (i.e., one bit or character) in the input data will generate a completely different output digest (Figure 1.9).

Figure 1.9 An Example of How a Minimal Change in Input Generates a Completely Different Hash Digest

Input text	SHA256 hash digest
This is sample text submitted to the SHA256 hashing algorithm. We will change it just a little bit by adding a period to the end of this sentence below	bbadb8c5138b8c9ced5c5197082ff5b36c9128f009bbb4cb450389c626c0214
This is sample text submitted to the SHA256 hashing algorithm. We will change it just a little bit by adding a period to the end of this sentence below	9d2f5d3b509a0334e752a0aaa0fe468f34affdddcc1912ae75321117514

This is a very valuable feature which enables a cryptographic hash function to serve several useful purposes. For example, this property is exploited when transmitting digital files over the internet to ensure they arrive on the receiver's computer in identical form to that sent by the sender. That process works as follows:

- The sender feeds the digital file (of any size) to one of a number of available cryptographic hashing functions⁴² and generates a hash digest, usually somewhere between 32 and 128 bits in length.

41. Particularly in the case of *structured data*, i.e., data which follows a pattern, such as the grammar of a language. This property is sometimes also referred to as the function having 'strong collision resistance'.

42. For further review and testing with a number of these hashing algorithms: https://www.tools4noobs.com/online_tools/hash/ (accessed 10 Jan. 2024).

- The sender transmits the digital file to the recipient over the internet and also sends particulars of the cryptographic hash function used, together with the generated hash digest.
- The receiver feeds the received digital file to the cryptographic hash function, generating a hash digest.
- If the receiver's hash digest matches the hash digest sent by the sender, then we must conclude that the received file is identical to the one transmitted by the sender.
- If the hash digests do not match, then there was an error in transmission, and the file needs to be resent.

[3] The 'No Reverse Engineering' Property

A cryptographic hash function is also a 'one-way function' which means it is not possible to recreate or 'reverse engineer' the original file from the hash digest.

One might think that calculating all possible permutations of the input data and then looking for the digest in that list to find the original input data is an option. This is, however, computationally prohibitive - it would require computational power, and time, well beyond the capability of current computer technology - particularly with large input data sets of different sizes.⁴³

One-way functions are very useful in many applications, including public-private key encryption.⁴⁴

[C] The Hash List

One application of a cryptographic hash function is the *hash list*. The hash list is used a lot in peer-to-peer (P2P) file sharing systems (see the P2P Networks section *infra*) to ensure that the various distributed pieces of a file come together to reconstitute the original file without errors. In that context, a hash list (Figure 1.10) functions as follows:

- The file being shared is broken up into several smaller pieces, called data blocks, which are sufficiently small to enable easy transmission over the internet as packets⁴⁵ or groups of packets.
- Each block is hashed, and the hash value is added to a separate hash list.
- Once all block hashes have been added, the hash list file is itself hashed and a *top hash* is generated.
- The top hash file is kept by a trusted authority.

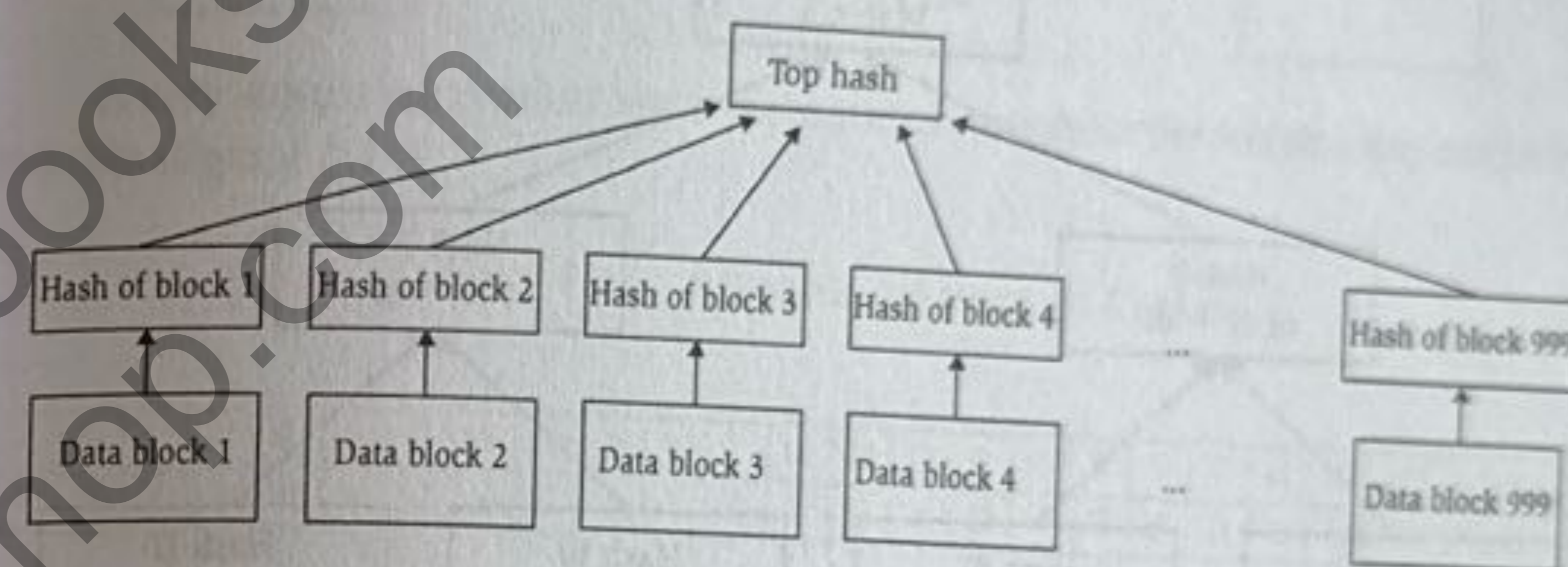
43. These types of problems are called *intractable* which means they cannot be solved within the bounds of our foreseeable computational capacity, see https://en.wikipedia.org/wiki/Computational_complexity_theory#Intractability (accessed 10 Jan. 2024).

44. See the discussion *infra*.

45. A packet is a small block of data, easily transmissible between computers.

- Anyone wishing to obtain the file from the P2P network will first obtain the top hash file from the trusted source.⁴⁶
- With that file in hand, the hash list can then be obtained from any other source (i.e., any other sharing user on the P2P network) for comparison with the top hash.
- If the hash of the hash list file matches the top hash, the user can proceed to download the various data blocks from any of the other users on the network who have those blocks available.
- Every block's hash is compared with the hash value for that block in the confirmed hash list file, and if it matches, it is accepted as genuine.

Figure 1.10 A Hash List



Given the properties of the cryptographic hash function, one is assured that if the hash of a received data block is not the same as the hash found in the hash list, a change or corruption is present in that block of data, necessitating a retransmission or obtaining the block from another source.

Using a hash list to verify the integrity of a file in blocks has another advantage in that it is much easier to determine where data errors exist than if one simply hashed the whole file. Fixing 'bad data' is thus much easier and faster.

[D] The Merkle Tree

A Merkle tree (also sometimes referred to as a *binary hash tree* or just *hash tree*) takes the ideas behind the hash list a step further for even better data integrity.

Rather than creating a simple list of hashes of data blocks, a Merkle tree incorporates data hashes into an upside-down tree-type structure, much like one might organize a family tree for a genealogy project. The individual elements in the tree are referred to as *nodes*, and every node has two children and one parent.

46. In the case of BitTorrent, this file actually contains both the top hash and the hash values for all the data blocks for the file being shared.