Unit 1

1. Explain with a neat diagram the key concepts of blockchain for business.

Ans:

Blockchain is a decentralized digital ledger that records transactions across a network of computers. It is used to record transactions for various types of assets, including financial transactions, property ownership, and supply chain management.

Some key concepts of blockchain for business include:

- Decentralization: Blockchain technology allows for a distributed network of computers to record and validate transactions, rather than relying on a central authority.
- Immutability: Once data is recorded on a blockchain, it cannot be altered or deleted. This creates a permanent and tamper-proof record of all transactions.
- Smart contracts: These are self-executing contracts with the terms of the agreement written directly into code. They can be used to automate processes and enforce the negotiation or performance of a contract.
- Transparency: Blockchain provides a transparent and publicly accessible ledger of all transactions, which can increase trust and accountability.
- Security: Blockchain uses cryptography to secure and validate transactions, making it difficult for unauthorized parties to tamper with the data.

Overall, the key concepts of blockchain for business are its decentralization, immutability, smart contract functionality, transparency and security that can be used to improve transparency, efficiency and trust in various business processes.

2. Define Blockchain, and explain in brief the limitation of current transaction systems that do not use the Blockchain?

Ans:

Blockchain is a shared, distributed ledger that facilitates the process of recording transactions and tracking assets in a business network.

The shortcomings of current transaction systems:

- Throughout history, instruments of trust, such as minted coins, paper money, letters of credit, and banking systems, have emerged to facilitate the exchange of value and protect buyers and sellers.
- Still, many business transactions remain inefficient, expensive, and vulnerable, suffering from the following limitations:
    - Cash is useful only in local transactions and in relatively small amounts.

- The time between transaction and settlement can be long.
- Duplication of effort and the need for third-party validation and/or the presence of intermediaries add to the inefficiencies.
- Fraud, cyberattacks, and even simple mistakes add to the cost and complexity of doing business, and they expose all participants in the network to risk if a central system, such as a bank, is compromised.
- Credit card organizations have essentially created walled gardens with a high price of entry.
- Half of the people in the world don't have access to a bank account and have had to develop parallel payment systems to conduct transactions.

3. Describe in detail the steps required to build your first blockchain application.

Ans:

Building a blockchain application involves several steps, including:

1. Define the problem or use case: Determine the specific problem or need that your blockchain application will address. This will help you to understand the requirements and limitations of the application.
2. Choose a blockchain platform: Decide on the blockchain platform that you will use to build your application. Some popular options include Ethereum, EOS, and Hyperledger. Each platform has its own set of features and capabilities, and it's important to choose the one that best suits your needs.
3. Design the architecture: Once you have chosen a blockchain platform, you can start designing the architecture of your application. This includes defining the data structure, the consensus mechanism, and the smart contract logic.
4. Develop the smart contracts: Smart contracts are the backbone of blockchain applications and are used to execute the logic of the application. They can be written in various programming languages such as Solidity (for Ethereum), C++ (for EOS) and Go, Java etc. (for Hyperledger)
5. Create the user interface: Develop the user interface of your application, which will allow users to interact with the blockchain and perform various actions, such as sending and receiving transactions.
6. Test and deploy the application: Before deploying your blockchain application to the main network, it is important to test it thoroughly to ensure that it is functioning as intended.

7. Monitor and maintain the application: Once your blockchain application is live, it's important to monitor its performance and maintain it to ensure its security and stability.
8. Keep in mind that Blockchain technology is still in its early stages of evolution, so it's important to stay up-to-date with the latest developments in the field, and be prepared to adapt your application as the technology evolves.

Note: The steps above are a general guideline, depending on the complexity of the project, additional steps may be required, and some steps might be combined or omitted.

4. Illustrate in brief the various participants on a blockchain network play.

Ans:

On a blockchain network, there are several types of participants who play different roles:

1. Users: These are individuals or organizations that use the blockchain network to send and receive transactions, create smart contracts, and access the decentralized applications (dapps) built on the network.
2. Miners/Validators: These are the participants who perform the computational work required to validate transactions and add them to the blockchain. They are incentivized for their work through rewards and transaction fees.
3. Developers: These are individuals or organizations that create decentralized applications (dapps) and smart contracts on the blockchain network. They use the blockchain platform's programming languages and tools to build and deploy their applications.
4. Node operators: These are the individuals or organizations that run a full node, which maintains a copy of the blockchain ledger and participates in the consensus process.
5. Consortium or Federated Blockchain: Participants in this type of blockchain network are pre-selected or authorized by a central authority, and the network is maintained by a group of pre-approved organizations.
6. Regulators: These are government or non-government entities that monitor and regulate the blockchain network for compliance with laws and regulations.
7. Auditors: These are third-party organizations that audit the blockchain network to ensure its security, integrity, and compliance with industry standards.

These participants work together to maintain the integrity and functionality of the blockchain network, and they play different roles depending on the type of blockchain network.

5. Explain how can we ease interaction friction and innovation friction.

Ans:

Easing interaction friction

Blockchain is particularly well-equipped to reduce interaction friction because it removes the barriers between participants in a transaction. Blockchain properties that reduce interaction friction include the following:

»Shared ledger: Asset ownership can be transferred between any two participants on the network, and the transaction recorded to the shared ledger.

» State-based communication: Today, banks communicate via secure messaging architecture, such as SWIFT, to accomplish tasks, with each bank maintaining its state of the task locally. With blockchain, banks can send messages that represent the shared state of the task on the blockchain, with each message moving the task to the next state in its life cycle.

» Peer-to-peer (P2P) transactions: On a blockchain for business network, participants exchange assets directly, without having to process the transaction through intermediaries or a central point of control, thus reducing the costs and delays associated with the use of intermediaries.

» Consensus: In place of intermediaries, blockchain uses consensus algorithms to validate and authorize transactions. Participants can conduct business at a pace that is more in-line with the pace of their business decisions.

» Smart contracts: Smart contracts eliminate the hassles and delays inherent in contracts by building the contract into the transaction. Through smart contracts, the blockchain establishes the conditions under which a transaction or asset exchange can occur. No more faxing or emailing documents back and forth for review, revision, and signatures.

Easing innovation friction

Innovation friction is possibly the most difficult to overcome through technology alone, but blockchain can help in the following ways:

» Eliminate the cost of complexity: As an organization's operations become increasingly complex, its growth results in diminishing returns. Blockchains have the potential to eradicate the cost of complexity and ultimately redefine the traditional boundaries of an organization.

» Reduce costs and delays of regulatory processes: Automation can't entirely eliminate governance through regulation, but it can lower the costs and reduce delays inherent in regulatory processes.

» Expand opportunities: Blockchain can be both good and bad for businesses by providing the technology that enables businesses to develop new competitive business models. Some businesses will fail, while others redefine entire industries.

6. What is Hyperledger and Hyperledger Fabric, discuss in details?

Ans:

Hyperledger is an open-source, collaborative effort to create blockchain technology suitable for the enterprise. Hyperledger Fabric is an open-source platform for developing blockchain solutions with a modular architecture and pluggable, interchangeable services using container technology.

Hyperledger Fabric provides a framework for developing blockchain solutions with a modular architecture, pluggable implementations, and container technology. While leveraging open-source best practices, Hyperledger Fabric enables confidentiality, scalability, and security in business environments.

Unlike other blockchain implementations like Bitcoin or Ethereum, Hyperledger Fabric is the only one that fulfils all four key elements of a blockchain for business:

» Permissioned network: Collectively defined membership and access rights within your business network

» Confidential transactions: Gives businesses the flexibility and security to make transactions visible to select parties with the correct encryption keys

» Doesn't rely on cryptocurrencies: Doesn't require mining and expensive computations to assure transactions

» Programmable: Leverages the embedded logic in smart contracts to automate business processes across your network

7. What are the primary goals of Hyperledger Fabric?

Ans:

The primary goals of Hyperledger Fabric are to

- Support a wide variety of industry use cases with different requirements
- Comply with statutes and regulations that exist today
- Support verified identities and private and confidential transactions
- Support permissioned, shared ledgers
- Support performance, scaling, auditability, identity, security, and privacy
- Reduce costly computations involved in proof of work

8. Explain with a neat diagram how business networks operated before and after the use of Blockchain.

Ans:

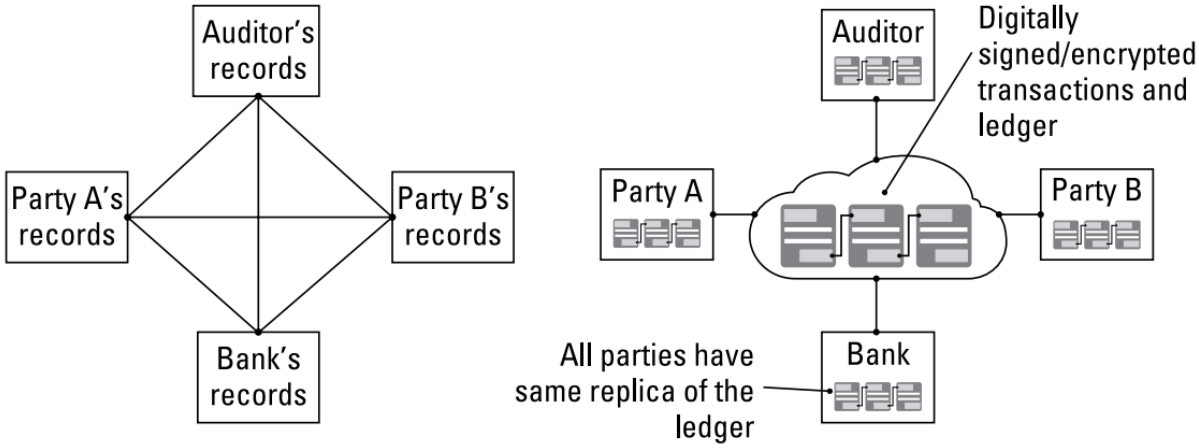


FIGURE 1-1: Business networks before and after blockchain.

- With traditional methods for recording transactions and tracking assets, participants on a network keep their own ledgers and other records.
- This traditional method can be expensive, partially because it involves intermediaries that charge fees for their services.
- It's clearly inefficient due to delays in executing agreements and the duplication of effort required to maintain numerous ledgers.
- It's also vulnerable because if a central system (for example, a bank) is compromised, due to fraud, cyberattack, or a simple mistake, the entire business network is affected.
- With traditional methods for recording transactions and tracking assets, participants on a network keep their own ledgers and other records.
- The blockchain architecture gives participants the ability to share a ledger that is updated, through peer-to-peer replication, every time a transaction occurs.
- Peer-to-peer replication means that each participant (node) in the network acts as both a publisher and a subscriber.
- Each node can receive or send transactions to other nodes, and the data is synchronized across the network as it is transferred.

9. Analyse the benefits for an Airline system after a blockchain is implemented.

Ans:

- Saves time, effort, and cost for the customers as well as the airlines
- Better customer experience
- Efficient use of airline resources (seats, flights)
- Opportunities for collaboration among airlines, benefiting customers as well as the airlines
- Seamless payment settlement system
- Better management of emergencies
- Potential for token-based airline-to-airline business model 8. An innovative operational model for the airline industry as a whole

10. Discuss the key elements of a blockchain that Hyperledger Fabric fulfils for any business?

Ans:

Hyperledger Fabric provides a framework for developing blockchain solutions with a modular architecture, pluggable implementations, and container technology. While leveraging open-source best practices, Hyperledger Fabric enables confidentiality, scalability, and security in business environments.

Unlike other blockchain implementations like Bitcoin or Ethereum, Hyperledger Fabric is the only one that fulfils all four key elements of a blockchain for business:

a. Permissioned network: Collectively defined membership and access rights within your business network
b. Confidential transactions: Gives businesses the flexibility and security to make transactions visible to select parties with the correct encryption keys
c. Doesn't rely on cryptocurrencies: Doesn't require mining and expensive computations to assure transactions
d. Programmable: Leverages the embedded logic in smart contracts to automate business processes across your network

11. Analyse the key business benefits of blockchain also explain the attributes of how Blockchain builds trust?

Ans:

A. Key business benefits of blockchain:

- Time savings: Transaction times for complex, multi-party interactions are slashed from days to minutes. Transaction settlement is faster, because it doesn't require verification by a central authority.
- Cost savings: A blockchain network reduces expenses in several ways:
    - Less oversight is needed because the network is selfpoliced by network participants, all of whom are known on the network.
    - Intermediaries are reduced because participants can exchange items of value directly.
    - Duplication of effort is eliminated because all participants have access to the shared ledger.
- Tighter security: Blockchain's security features protect against tampering, fraud, and cybercrime. If a network is permissioned, it enables the creation of a members-only network with proof that members are who they say they are and that goods or assets traded are exactly as represented.

Not all blockchains are built for business. Some are permissioned while others aren't. A permissioned network is critical for a blockchain for business, especially within a regulated industry. It offers

- Enhanced privacy: Through the use of IDs and permissions, users can specify which transaction details they want other participants to be permitted to view. Permissions can be expanded for special users, such as auditors, who may need access to more transaction detail.
- Improved auditability: Having a shared ledger that serves as a single source of truth improves the ability to monitor and audit transactions.
- Increased operational efficiency: Pure digitization of assets streamlines transfer of ownership, so transactions can be conducted at a speed more in line with the pace of doing business.

B. Attributes of how Blockchain builds trust:

- Distributed and sustainable: The ledger is shared, updated with every transaction, and selectively replicated among participants in near real time. Because it's not owned or controlled by any single organization, the blockchain platform's continued existence isn't dependent on any individual entity.
- Secure, private, and indelible: Permissions and cryptography prevent unauthorized access to the network and ensure that participants are who they claim to be. Privacy is maintained through cryptographic techniques and/or data partitioning techniques to give participants selective visibility into the ledger; both transactions and the identity of transacting parties can be masked. After conditions are agreed to, participants can't tamper with a record of the transaction; errors can be reversed only with new transactions.

- Transparent and auditable: Because participants in a transaction have access to the same records, they can validate transactions and verify identities or ownership without the need for third-party intermediaries. Transactions are time-stamped and can be verified in near real time.
- Consensus-based and transactional: All relevant network participants must agree that a transaction is valid. This is achieved through the use of consensus algorithms. Each blockchain network can establish the conditions under which a transaction or asset exchange can occur.
- Orchestrated and flexible: Because business rules and smart contracts (that execute based on one or more conditions) can be built into the platform, blockchain business networks can evolve as they mature to support end-to-end business processes and a wide range of activities.

12. Illustrate in detail the common types of market friction that blockchain is capable of alleviating.

Ans:

- Information frictions:
  Information frictions result from the following limitations:
  - Imperfect information: Participants in a transaction don't have access to the same information, giving one party an unfair advantage. Information may also be incorrect or inconsistent, leading to bad decisions or delays while reconciling it.
  - Inaccessible information: The potential value of abundant data and information is greatly constrained by the technical challenges of storing, processing, sharing and analyzing it. As a result, much information is not collected or remains inaccessible.
  - Information risks: Technological risks to information, from hacking to cybercrime and privacy concerns to identity theft are on the rise. These incur growing costs, as well as damage to brand reputations.
- Interaction frictions:
  - Interaction frictions arise when either the cost of transaction is too high or the degree of separation (physical or otherwise) between parties is too great.
  - Business transactions that take days and are costly to manage via intermediaries are prime candidates for disruption by nimbler competitors.
  - Interaction frictions are often magnified by the number of interactions required.
  - Blockchain's peer-to-peer architecture can often reduce the number of interactions or parties required to execute a transaction, thus reducing the number of potential sources of interaction friction.
- Innovation frictions:
  - Innovation frictions are any conditions, internal or external, that compromise an organization's ability to respond to market changes, such as the following:
  - Institutional inertia: Internal bureaucracy and legacy systems along with the natural human resistance to change can impede a company's responsiveness.
  - Restrictive regulations: While regulations may be required to control industry behaviour, they have the side effect of introducing costs and delays.

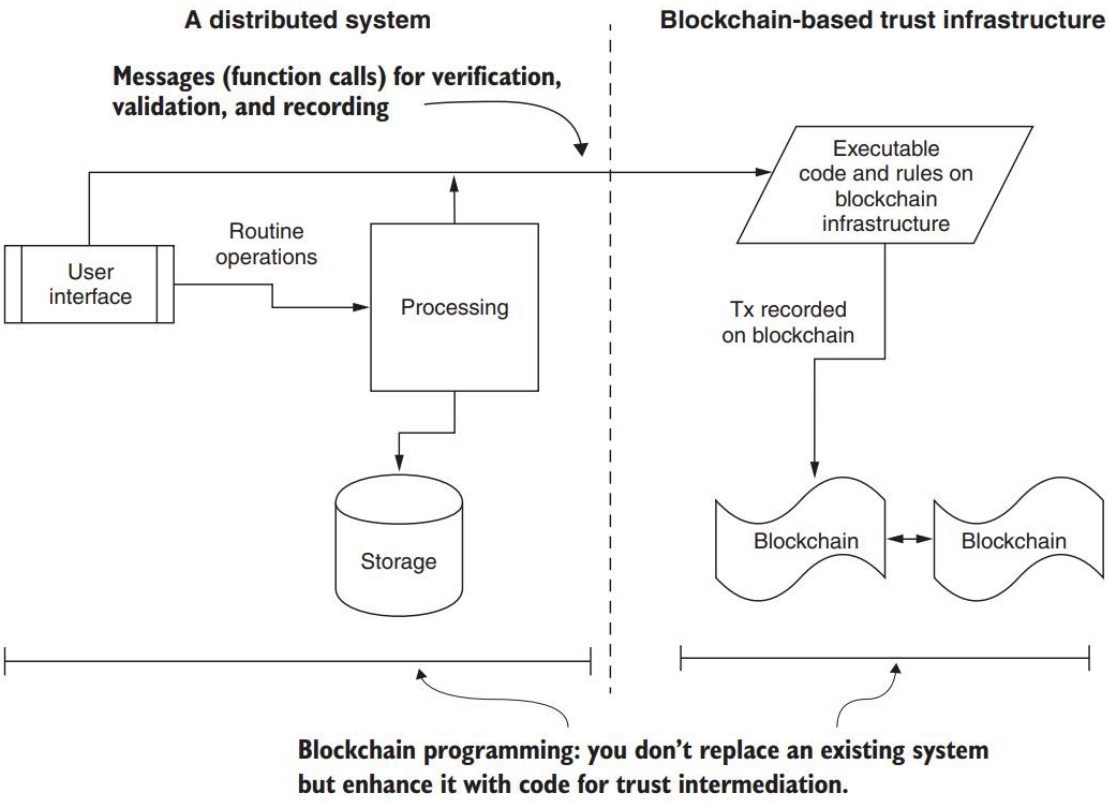
- Invisible threats: New competitive business models made possible by new technologies are threats for which organizations can’t plan. For many, this growing uncertainty will disrupt continued business success. Both small organizations and nimble larger ones will try new approaches, and though many will fail, some will redefine entire industries.

Unit 2

1. Illustrate with a neat diagram a Blockchain-based trust infrastructure within a large system.

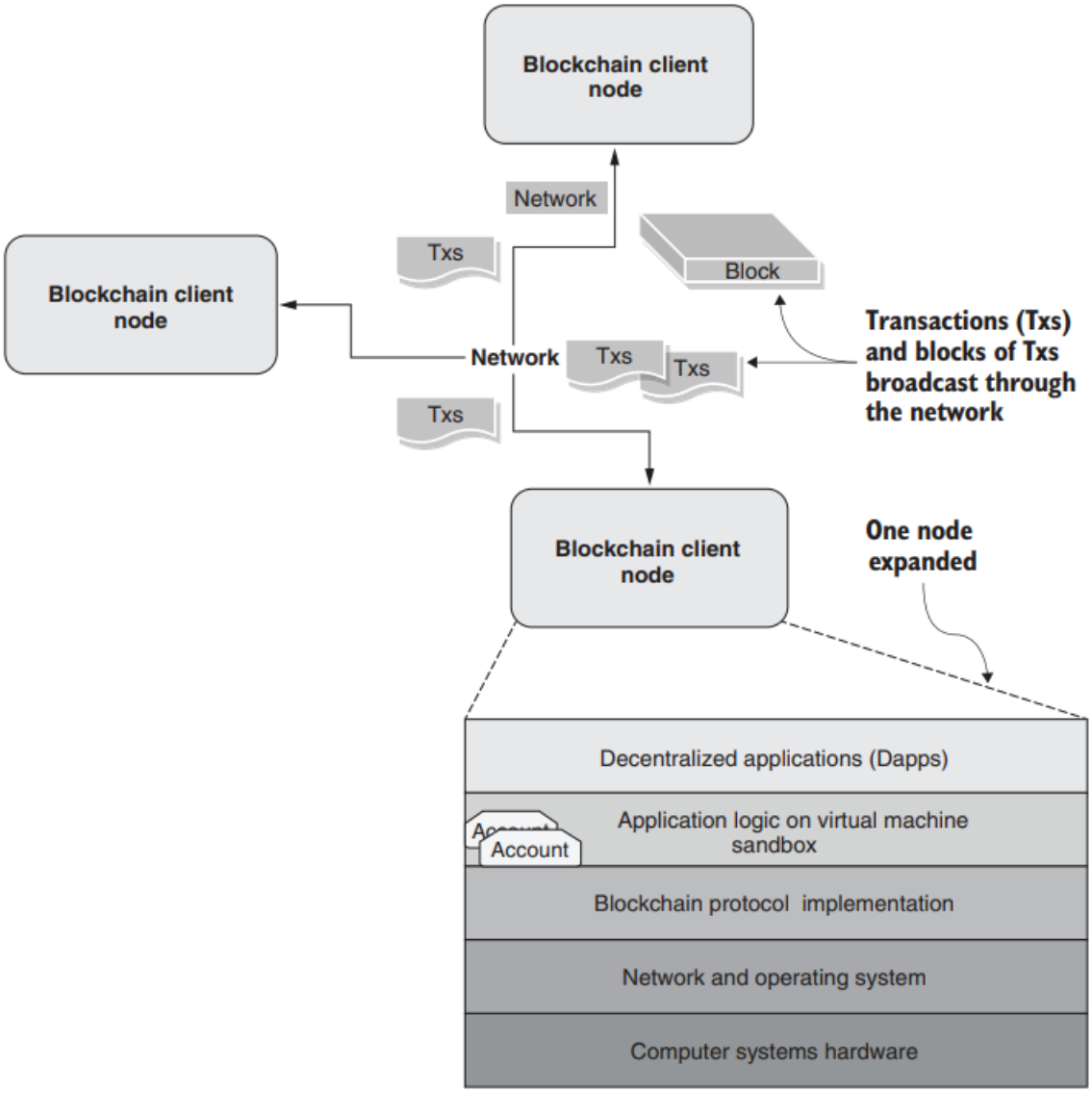
Ans:

Blockchain-based trust infrastructure uses blockchain technology to establish trust within a large system. It creates a secure and transparent record of all transactions and ensures secure and transparent data sharing. Additionally, smart contracts can automate the execution of agreements between parties, increasing the efficiency and security of the system. Overall, it makes the system more transparent, secure, and efficient by providing a decentralized, tamper-proof system for recording and sharing data.



2. Assume a bank has three servers (nodes), then show a network of three nodes connected by a blockchain network having various transactions and blocks.

Ans:



3. What is a transaction in Blockchain, and where and how do these transactions play an important role in Blockchain?

Ans:

A transaction in Blockchain refers to the transfer of value, or the exchange of digital assets, from one party to another. Transactions are recorded on the blockchain, which is a decentralized and distributed digital ledger that contains a record of all the transactions that have taken place on the network.

Transactions play an important role in Blockchain in several ways:

a. Recording of Transactions: Transactions are the building blocks of the blockchain, and they are used to record all the activities that take place on the network. The transactions are grouped together in blocks, which are then added to the blockchain in a chronological order.

b. Validation of Transactions: Transactions are validated by the network participants, called nodes, using a consensus algorithm before they are added to the blockchain. This ensures the integrity and security of the transaction records on the blockchain.

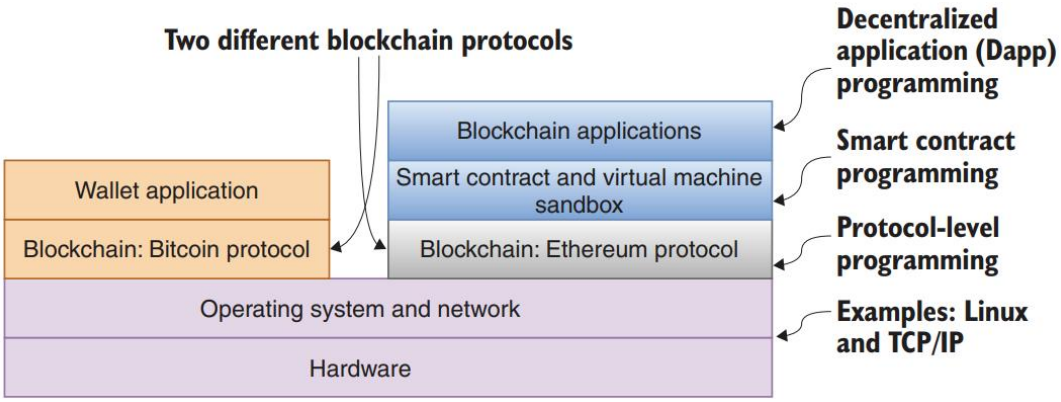
c. Smart Contracts: Transactions are the mechanism that triggers the execution of smart contracts, which are self-executing contracts with the terms of the agreement directly written into the code. Smart contracts allow for trustless and automated execution of agreements between parties.

d. Facilitating Digital Transactions: Blockchain-based transactions enable the exchange of digital assets, such as cryptocurrency, without the need for intermediaries. This allows for faster and more secure digital transactions.

In summary, Transactions play an important role in blockchain by recording and validating activities on the network, triggering smart contracts, and facilitating digital transactions.

4. Explain with a neat diagram Blockchain stacks and types of programming used.

Ans:



Blockchain stacks refer to the various layers and components that make up a blockchain system. These stacks typically include the following layers:

i. Application layer (Dapp): This is the topmost layer and it is used to interact with the blockchain network. It includes the user interface and the smart contracts that define the rules for the blockchain network.

ii. Middleware layer (Smart contract): This layer sits between the application layer and the underlying blockchain network. It provides the necessary infrastructure for the application layer to interact with the blockchain network, such as libraries and APIs.

iii. Protocol layer (Protocol-level): This is the underlying layer that defines the rules and protocols for the blockchain network. It includes the consensus algorithm, the peer-to-peer network, and the cryptographic functions used to secure the network.

The types of programming languages used in blockchain stacks depends on the specific blockchain platform that is used.

a) Bitcoin: C++.

b) Ethereum: Solidity.

c) Hyperledger: Go, Java, and JavaScript.

d) EOS: C.

e) Corda: Java.

f) Cosmos: Go.

5. Explain what does a Finite State Machine (FSM) diagram is composed of?

Ans:

A Finite State Machine (FSM) diagram is a visual representation of a system that can be in one of a finite number of states, and that can transition from one state to another in response to certain inputs or events. An FSM diagram is composed of the following elements:

1. States: A state represents a particular condition or phase of the system. The FSM diagram shows all the possible states that the system can be in.
2. Transitions: Transitions represent the movement of the system from one state to another in response to certain inputs or events. The FSM diagram shows the possible transitions between states, and the inputs or events that trigger each transition.
3. Inputs or Events: The inputs or events are the conditions or triggers that cause the system to transition from one state to another. These inputs or events are usually represented by arrows pointing from the input or event to the transition it triggers.
4. Outputs: The outputs are the effects or actions that the system performs when it is in a particular state. The FSM diagram can also show the outputs of the system for each state.
5. Initial State: The initial state is the state in which the system starts. It's represented by a filled-in circle or a double circle.
6. Final State: The final state is the state in which the system ends. It's represented by a filled-in circle with a dot in the middle.

The FSM diagram is a useful tool for modeling and understanding the behavior of a system, as it provides a clear and visual representation of the possible states, transitions,

and actions of the system. It's widely used in Computer Science, Automation, and Control Systems.

6. Construct an FSM diagram for a juice vending machine application.

Ans:

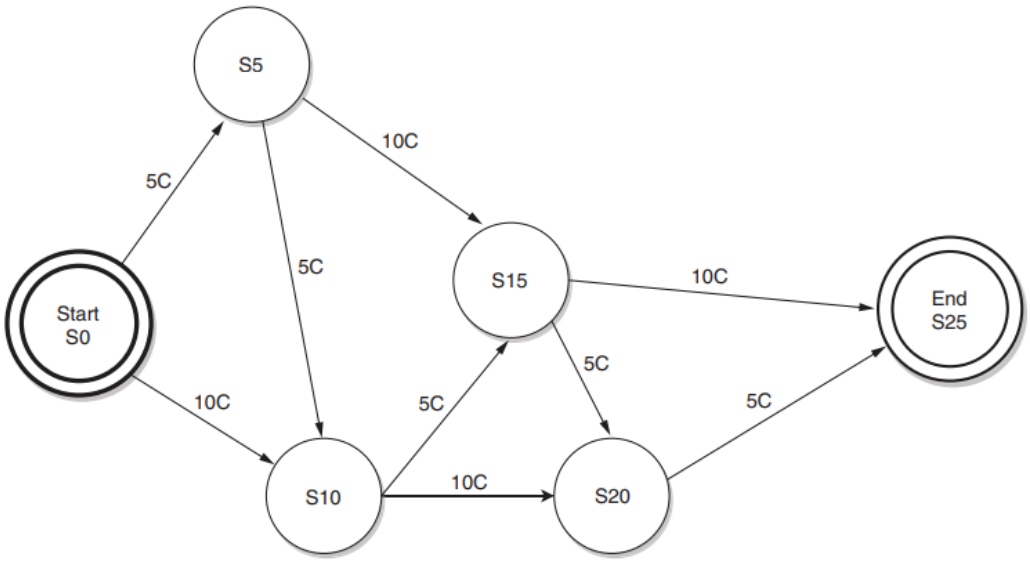


Figure A.3 FSM for counting exact change (25 cents)

- For example, we'll use the count coins use case from the vending machine.
- For simplicity, let's assume that the vending machine counts up to 25 cents and takes 5- and 10- cent coins as the only inputs. The FSM for counting to 25 cents is shown in figure A.3.
- The starting state is S0, and the ending state of the FSM design diagram is S25.
- That is, beginning at the value 0, using 5C and 10C coins, you would like to reach the S25 state. The states possible are S5, S10, S15, and S20.
- You can see the transitions brought about by the customer inserting 5C and 10C coins, ending up with S25. The FSM in figure A.3 exhaustively enumerates all the logical possibilities.
- Assume that the customer is aware of the requirement to insert exact change.

7. Explain with a neat diagram the Evolution of the internet and the blockchain-based trust layer

Ans:

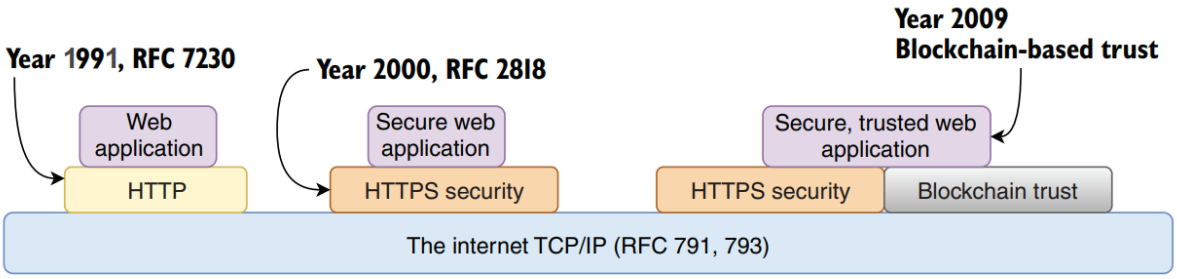


Figure 1.11 Evolution of the internet and the blockchain-based trust layer

- The internet was created for sharing research among scientists. It enabled connectivity among computing machines and internetworking.
- Later, Hypertext Transfer Protocol (HTTP) was introduced as the underlying protocol for the web.
- It became a standard around 1991 and opened many commercial activities through web applications. Note that security was not part of the standard at that time.
- With increased digitization and adoption of online activities came rampant online fraud and security breaches.
- Security became critical for web applications, and it was retrofitted into HTTP as a standard (HTTPS) around 2000. This addition enabled secure web applications.
- Global standards were established with formal Request for Comments (RFC) documents from the Internet Engineering Task Force (IETF)—RFCs 7230, 2818, and so on.
- Blockchain, introduced in 2009, established a trust layer alongside the security layer of the internet.
- Trust is currently realized in centralized systems by ad hoc means (such as verifying credentials, recommendation systems, and reviews/ratings) and by human involvement in other situations, such as at airports and grocery-store checkouts.
- Blockchain enables the trust layer for Dapps through software-based verification, validation, and immutable recording of transactions and facts.

8. Compare with a neat diagram, Bitcoin transactions versus smart contract transactions.

Ans:

Bitcoin transactions are used to transfer the ownership of bitcoins, while smart contract transactions are used to execute the terms of a smart contract on the blockchain. A smart contract is a program that runs on the blockchain and can be used to facilitate, verify, and enforce the negotiation or performance of a contract. Smart contracts allow for a wide range of applications, such as decentralized exchanges, lending platforms, and prediction markets. Bitcoin transactions are simpler and more limited in functionality compared to

smart contract transactions, but they are also more widely used and have a longer track record of being secure.

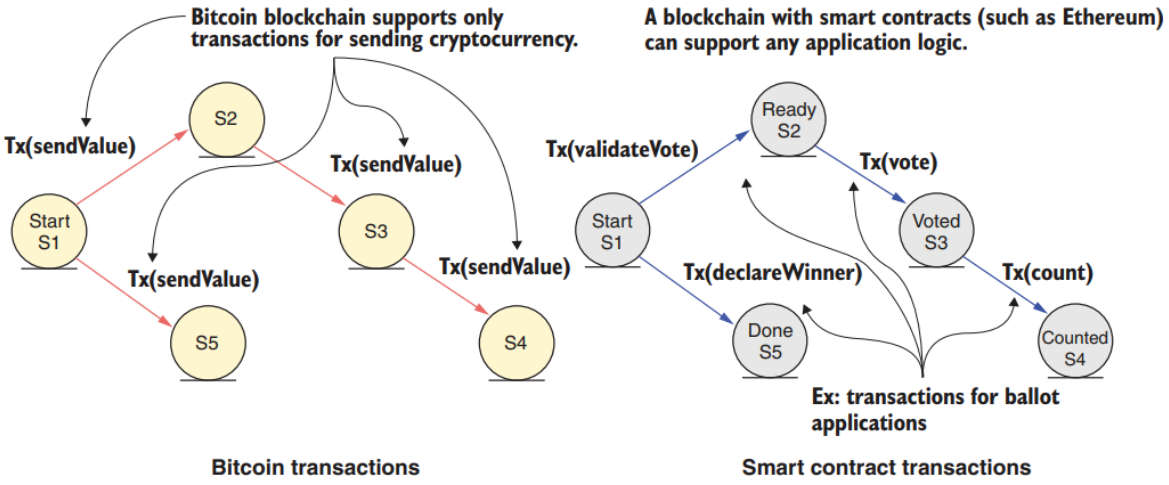


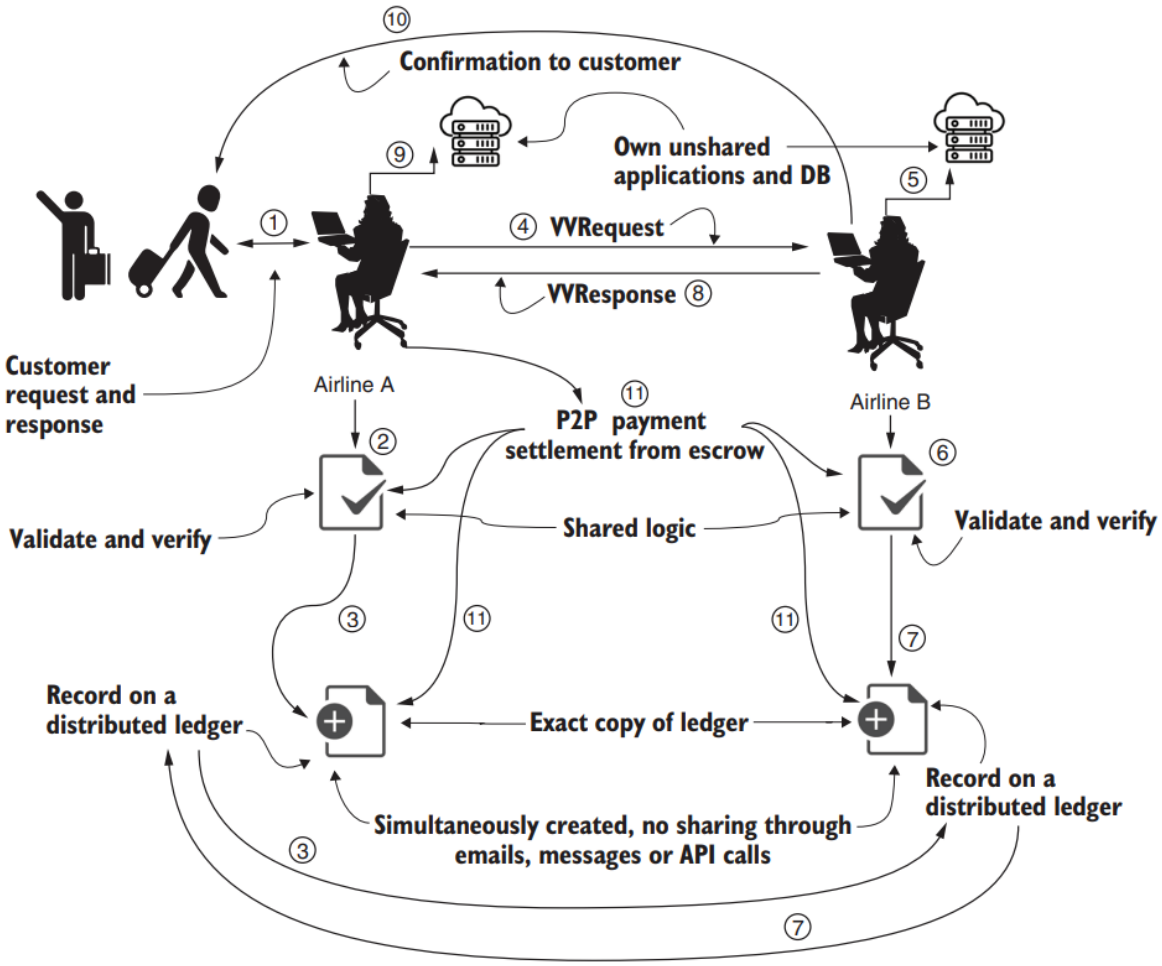
Figure 2.2 Cryptocurrency transactions versus smart contract transactions

Let's compare a Bitcoin transaction and a smart contract transaction, as shown in figure 2.2, to give you an idea of the difference between currency transactions and noncurrency, application-dependent function calls. As you can see, in Bitcoin, all the transactions are about sending value (Tx(sendValue)). In the case of a blockchain that supports smart contracts, a transaction embeds a function implemented by the smart contract. In figure 2.2, this function is a voting smart contract. The functions are validateVoter(), vote(), count(), and declareWinner(). The invocation of these functions results in the transactions that will be recorded on the blockchain (Tx(validateVoter), Tx(vote), and so on). This ability to deploy an arbitrary logic on a blockchain significantly enhances its applicability beyond simple cryptocurrency transfers.

9. Demonstrate with a diagram the operations of various participants in a decentralized Airline System Consortium (ASK instead of ASC) blockchain.

Ans:

- A customer initiates a change of flight seat that they hold on airline A.
- An agent or application at airline A verifies and validates the request through smart contract logic shared among the ASK consortium members.
- Once verified, the request Tx is confirmed and recorded in a distributed immutable ledger. Now everyone in the consortium knows that a legitimate request has been made.
- In the simplest design, an agent at airline A sends the verified and validated request (VVRequest) to airline B. (Alternatively, we could use a broadcast model in which many airlines get the request, and any one of them could respond.)
- An agent or application at airline B checks the airline's database to check for availability.
- An agent at airline B responds through shared smart contract logic that verifies and validates the common interests and shared rules of the consortium.



- Once verified, the response Tx is confirmed and recorded in a distributed immutable ledger. Now everyone in the consortium knows that a response has been sent.
- Airline B sends the response (indicated by VVResponse) to the agent at airline A.
- Airline A updates its database, noting that a change has been made.
- An agent at airline B sends the customer the information for the flight seats and other details.
- Payments are settled through peer-to-peer Txs, using the escrow or deposit that participating airlines hold in their shared smart contract. The payment settlement can be embedded in other suitable operations in the system but will be handled by the shared smart contract and recorded in the ledger. This settlement is automatically carried out by the smart contract logic.

10. What makes a blockchain contract smart?

Ans:

A smart contract is a type of blockchain contract that is self-executing and self-enforcing. It is "smart" because it contains code that can automatically perform certain actions when certain conditions are met. This code is executed by the nodes of the blockchain network, so it is transparent, tamper-proof and decentralized.

A smart contract typically includes the following features:

- The terms of the contract, which are encoded in the form of code and stored on the blockchain
- The ability to automatically trigger actions, such as transferring assets or updating a database, based on the terms of the contract
- The ability to automatically verify the fulfillment of the contract's conditions
- The ability to automatically enforce the contract's rules and penalties in case of non-compliance

A smart contract also has the capability to interact with other smart contracts, making them building blocks to larger decentralized applications.

Overall, the "smart" in smart contract refers to the programmability and automation that allows the contract to execute automatically when certain conditions are met, without the need for intermediaries.

11. Explain with a neat diagram, the process how transactions are created into blocks and how these blocks turn in to chains.

Ans:

A set of transactions makes a block, and a set of blocks make a blockchain, as shown in figure 1.8. The process is as follows:

- Transactions on the network are verified, gathered, and pooled. Nodes select a set of transactions from the pool to create a block.
- Participant nodes use a consensus algorithm to collectively agree or come to a consensus on a single consistent block of transactions to be appended to the existing chain.
- A hash or representative value of the current lead block of the chain is added to the newly appended block, creating a chain link.

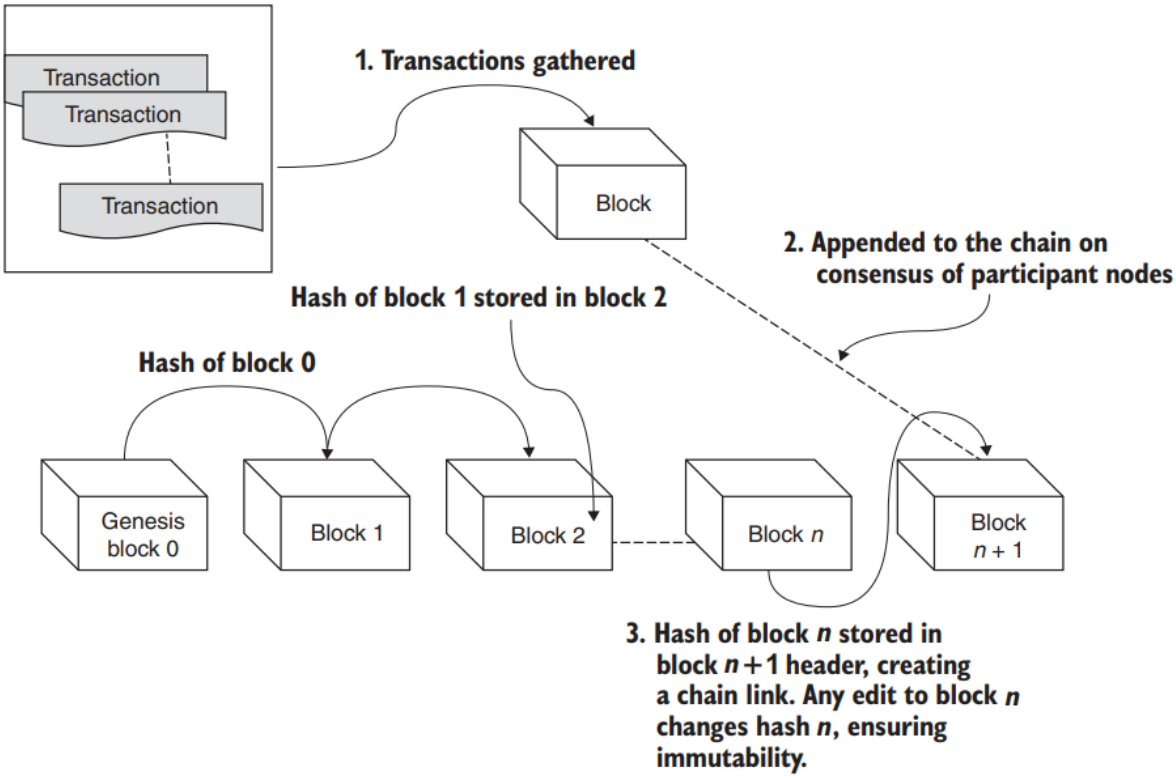


Figure 1.8 Transactions to blocks and blocks to the blockchain

As figure 1.8 demonstrates, a blockchain is an append-only distributed immutable ledger. Its creation begins with a single block called the genesis block. Every node of a stakeholder on the blockchain has an identical copy of the blockchain, starting with the genesis node. A blockchain DLT, therefore, is

- Distributed, because the blockchain protocol ensures that every distributed node involved has an identical copy of the chain of blocks.
- Immutable, because each newly created block is linked to the existing blockchain by the hash value of the current head of the blockchain, as shown in figure 1.8.

At this point, it is sufficient to know that a representative signature value of the block n is stored in the block n+1 to ensure immutability.

12. Illustrate with a use case diagram representing a decentralized application that will use the counter and the functions involved.

Ans:

The UML use case diagram helps you think through the problem and decide how the smart contract—and, more specifically, its functions—will be used. Figure 2.3 shows just one actor: a stick figure representing a decentralized application that will use the counter.

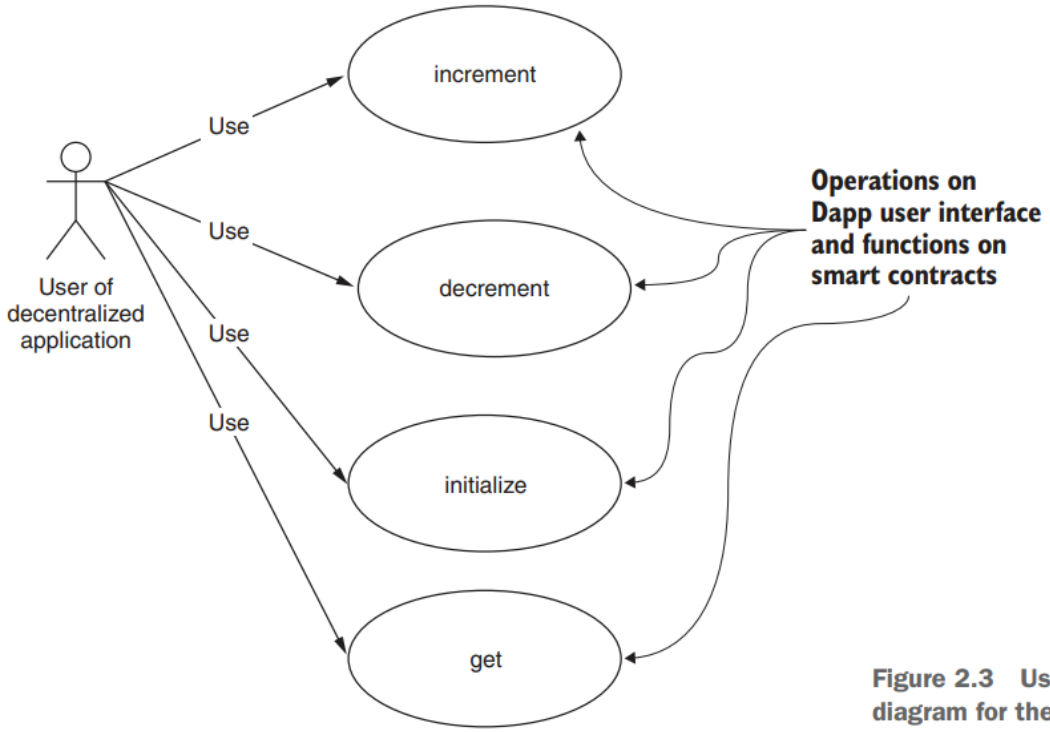


Figure 2.3 Use case diagram for the counter

First, let's think about the functions of a counter:

- ⇨ initialize() to a value.
- ⇨ increment() by a value.
- ⇨ decrement() by a value.
- ⇨ get() to access the value of the counter.

This diagram clearly articulates the intent of the smart contract. This diagram is a good starting point for the design process, providing an artifact for discussion among your team members and stakeholders who are interested in the problem. It also provides a systematic lead-in to the next steps in the design process. Note that this step of use case design representation is dependent on the problem specification; it does not require you to specify any coding or system dependencies.

13. Compare with a neat diagram, Bitcoin versus Ethereum protocol stacks.

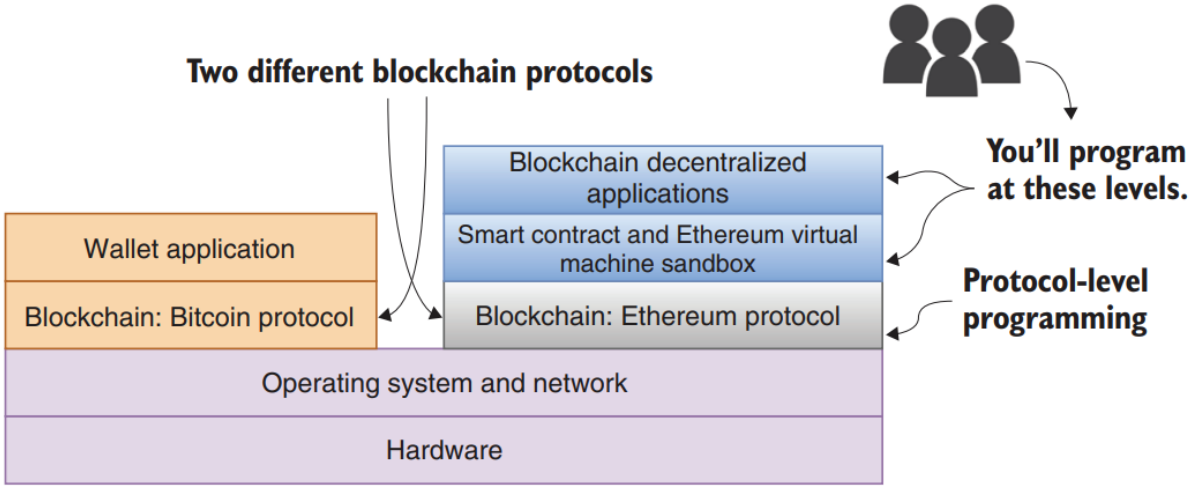
Ans:

Bitcoin and Ethereum are both blockchain platforms, but they have some key differences in their protocol stacks.

Bitcoin is primarily a cryptocurrency, and its protocol stack is focused on facilitating the transfer of bitcoins between users. The Bitcoin protocol includes a consensus mechanism (Proof-of-Work), a network protocol, and a transaction format. The consensus mechanism is used to confirm transactions and add them to the blockchain, the network protocol is used to propagate transactions and blocks across the network, and the transaction format is used to specify how bitcoins are transferred between addresses.

Ethereum, on the other hand, is a blockchain platform that enables the development of decentralized applications (dApps) and smart contracts. Ethereum's protocol stack includes a consensus mechanism (Proof-of-Work, currently transitioning to Proof-of-Stake), a network protocol, a transaction format, and a virtual machine (Ethereum Virtual Machine, EVM). The consensus mechanism is used to confirm transactions and add them to the blockchain, the network protocol is used to propagate transactions and blocks across the network, the transaction format is used to specify how ether (the native cryptocurrency of Ethereum) and other assets are transferred between addresses, and the virtual machine is used to execute smart contracts.

In summary, Bitcoin is mainly used for digital transactions and the transfer of bitcoins, while Ethereum is used for building decentralized applications and smart contracts with its own cryptocurrency Ether.



UNIT-3

1.Demonstrate how blockchain-based decentralized systems constituents of trust and integrity are represented.

2.Identify are the stages for a Ballot contract diagram.

3.Illustrate the requirement for the develop and configure of the web application for the front end?

4.Explain the ballot-contract and ballot-app directory structure with a necessary diagram

HTML, JavaScript, and CSS for rendering the server contents for user interaction

ϒ A server to host the base entry script defined in index.js

ϒ Server code (app.js) linking the web server and web client

ϒ Additional wrappers and plugins for any frameworks such as Bootstrap and the web3 API

ϒ A package configuration file, package.json

These items are organized in a standard project directory structure, as shown in figure 4.6. On the left is the ballot-contract directory, and on the right, you see the contents of ballot-app. ballot-contract was covered in chapter 3 and also in section 4.3. Now let's get started on the web app (ballot-app) branch of the Dapp project. Your objective in this section is to understand the various components of the web application for the Ballot-Dapp. With this objective in mind, I've provided the complete code elements needed for the ballot-app to explore

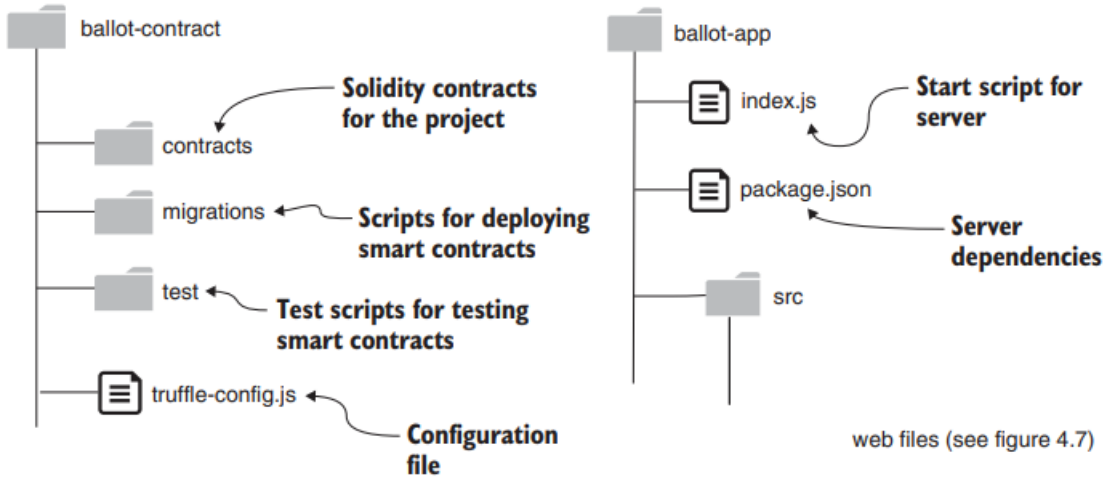


Figure 4.6 ballot-contract and ballot-app directory structure

5.Demonstrate the Dapp Stack with a neat diagram and also what are the Dapp development features using Truffle environment?

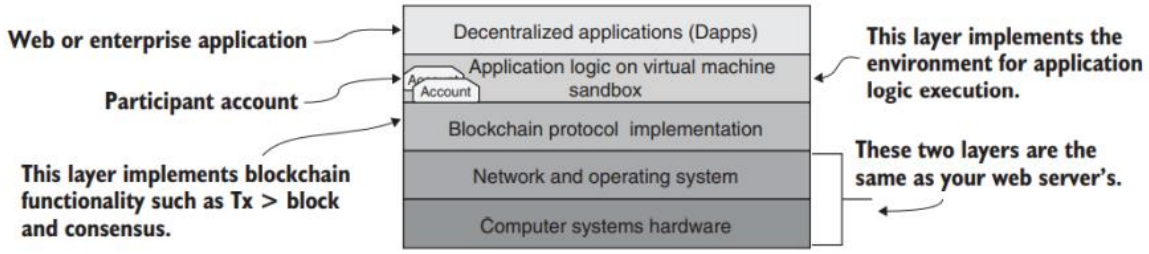


Figure 2.1 Blockchain application stack and layers

Let's start from the bottom and move up. The lower two levels are the standard hard ware and software of most computing systems. The next level up is the blockchain pro tocol level: it houses the components of the blockchain, but you won't program at this level. The next layer hosts the application logic. This layer is where you solve problems like access control to data and code functions for validation, verification, and record ing. The top layer is the user-facing interface where web (or enterprise) programming is done, such as with HTML, JavaScript, and associated frameworks. These elements form the Dapp and its user interface (UI) layer.

6.What are the items for the Ballot smart contract artifacts and ballot-contract directory structure with a neat diagram with respect to Files and folders.

HTML, JavaScript, and CSS for rendering the server contents for user interaction

ϒ A server to host the base entry script defined in index.js

ϒ Server code (app.js) linking the web server and web client

ϒ Additional wrappers and plugins for any frameworks such as Bootstrap and the web3 API

ϒ A package configuration file, package.json

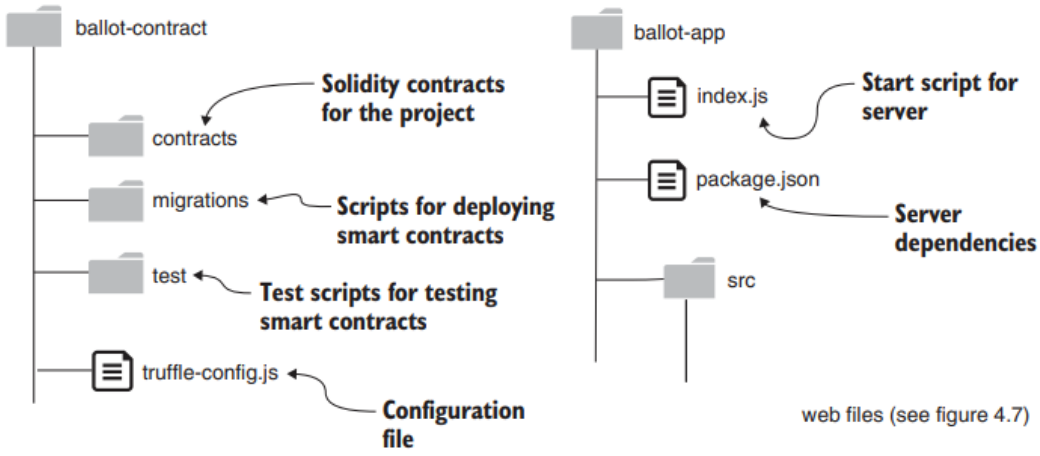


Figure 4.6 ballot-contract and ballot-app directory structure

7.Explain the Dapp development layers with a neat diagram.

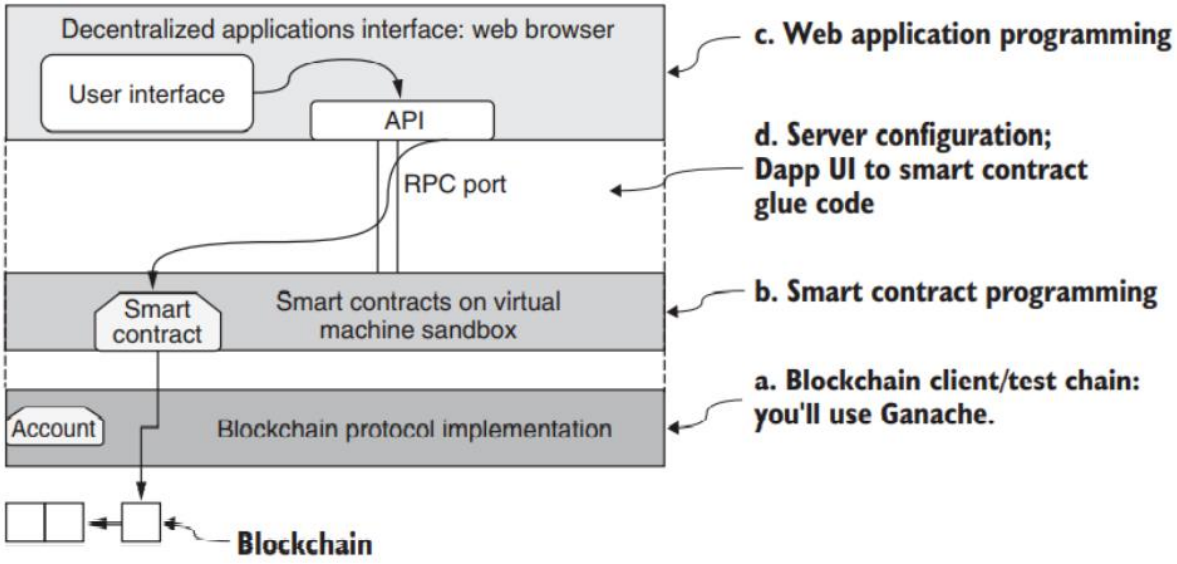
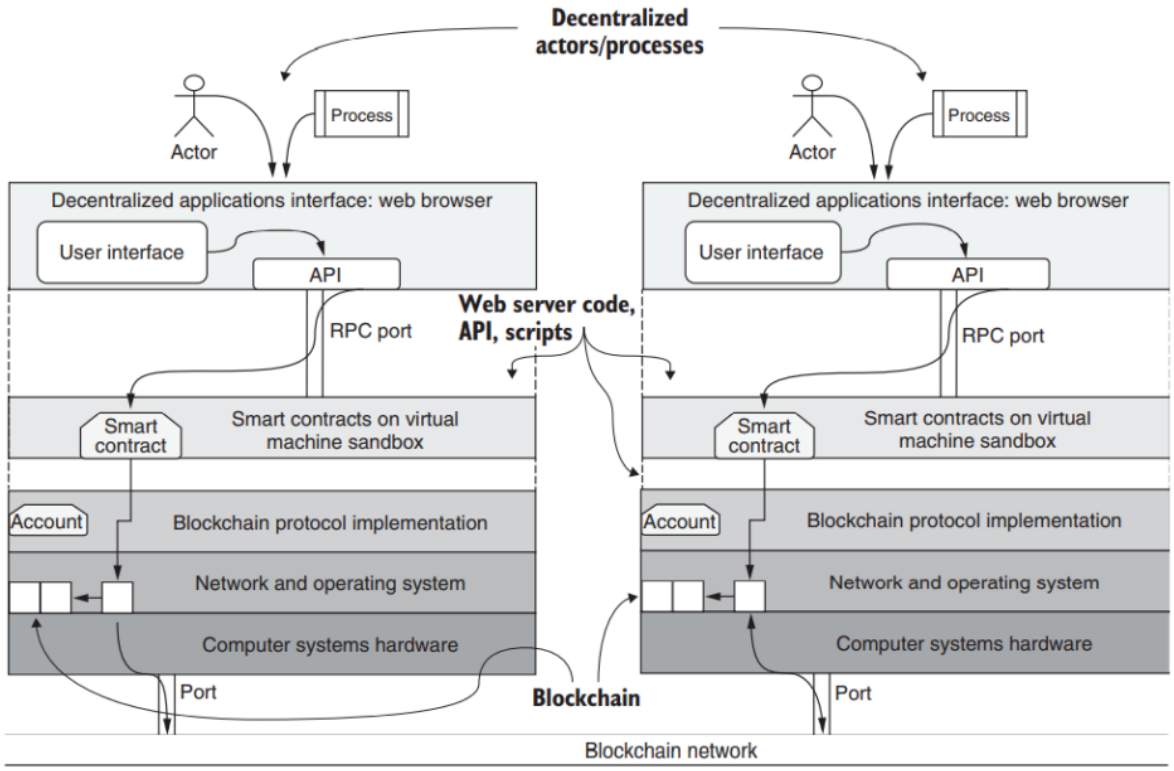


Figure 4.3 Dapp development layers

8.Explain with a neat diagram the architectural model of a blockchain network where two nodes connected by a blockchain network.



Starting at the top, the users (actors) or processes acting on behalf of users invoke the UI functions. These functions use web application software and blockchain APIs to connect to smart contract functions. Txs representing the smart contract function invocations are

recorded on the blockchain. (Note that only some of the necessary Txs will be recorded.) You can follow the operational flow in a node from an actor to the consis tent blockchain recording on both nodes via the blockchain network. This figure also illustrates how a blockchain-based Dapp is not a standalone application but is dependent on its host operating system's file system, ports, and network capabilities.

10.Illustrate the Dapp development process steps.

1 Analyze the problem and its requirements. The problem definition always comes before the solution.

2 Design the solution, using UML contract diagrams.

3 Develop and test the smart contract, using the Remix IDE.

4 Develop the -contract module, using the Truffle IDE.

5 Deploy the smart contract on the Ganache test chain.

6 Develop the -app module, using Node.js and related software modules.

7 Design the web UI and the web component of the application.

8 Test the Dapp you've developed via the web UI, using a browser with the MetaMask plugin enabled. This step connects the web UI to the contracts deployed on the blockchain (in this case, Ganache)

Unit 4:

1)Demonstrate with a neat diagram, the different types of data to 256-bit hashes.

Hashing is the process of mapping data of an arbitrary length to a fixed size by using a specially defined function called a hash function. Even a single bit change in the data elements changes the hash value of the data elements significantly. Any type of data, including a database or an image, can be succinctly represented by a hash of fixed length, as shown in figure 5.9. A 256-bit data item and a strong hash function together provide a large, collision-free account address space. Collision-free means there is a high probability that no two values generated by the hash function will be the same and that you'll get a unique hash value when you apply the hashing function to the same data elements. This property is an important requirement of a hash function: you don't want to have the same identification number as your friend!

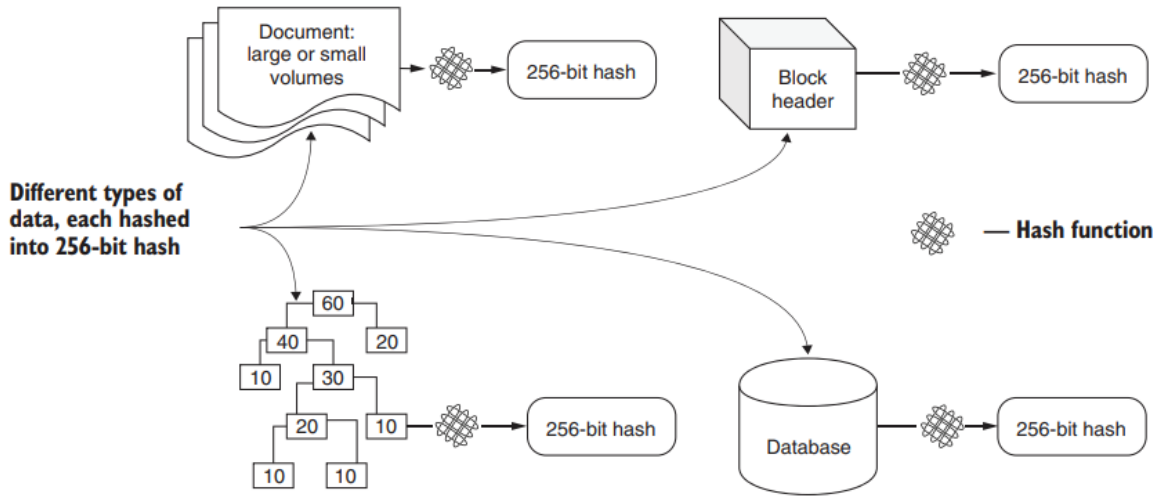
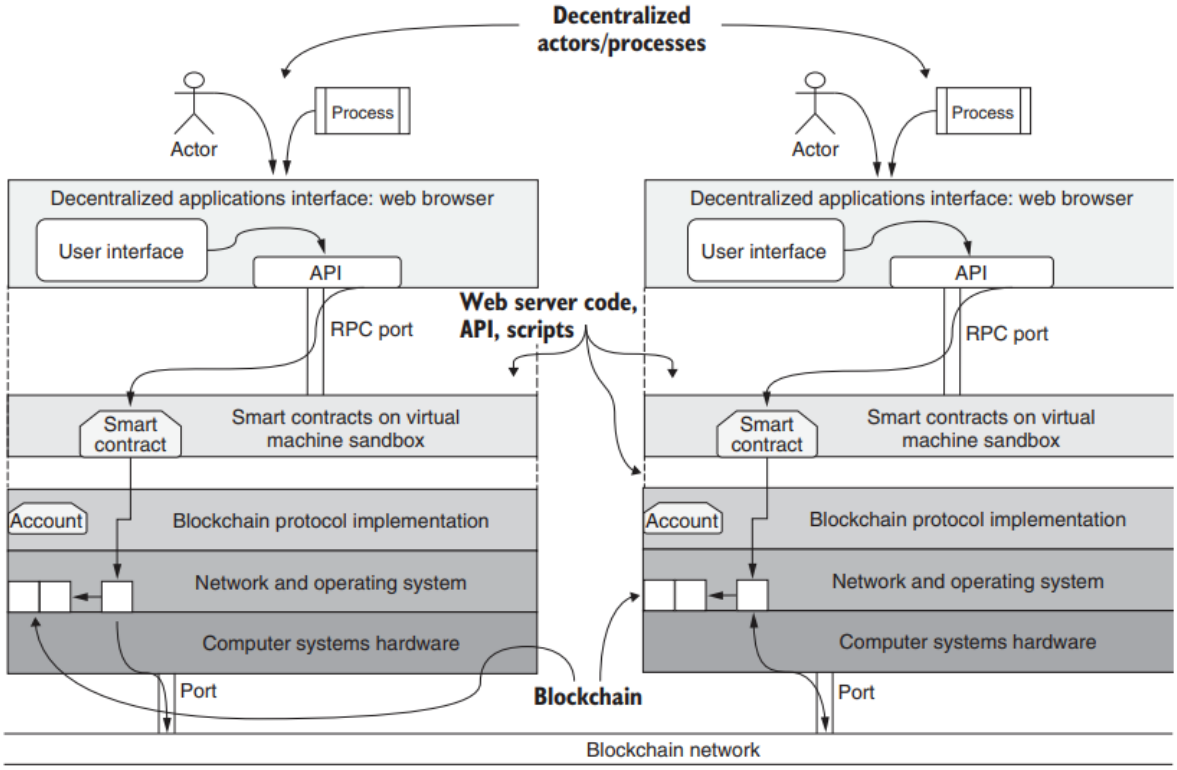


Figure 5.9 Transforming different types of data to 256-bit hashes

2) Explain with a neat diagram the architectural model of a blockchain network where two nodes connected by a blockchain network.



Starting at the top, the users (actors) or processes acting on behalf of users invoke the UI functions. These functions use web application software and blockchain APIs to connect to smart contract functions. Txs representing the smart contract function invocations are recorded on the blockchain. (Note that only some of the necessary Txs will be recorded.) You can follow the operational flow in a node from an actor to the consis tent blockchain recording on both nodes via the blockchain network. This figure also illustrates how a blockchain-based Dapp is not a standalone application but is dependent on its host operating system's file system, ports, and network capabilities.

3) Illustrate with a neat diagram the Web files and folders in the src directory.

The src directory contains

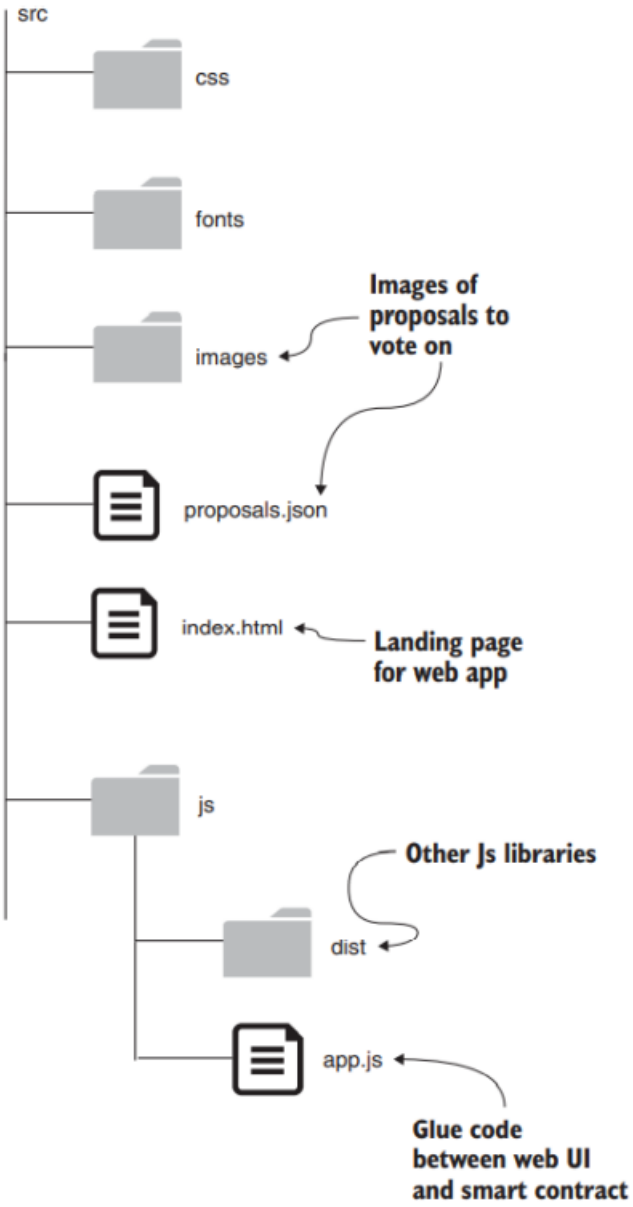
ϒ The usual artifacts for a web page (CSS, fonts, images, JavaScript).

ϒ The landing page for the web application (index.html).

ϒ proposals.json, which holds details about the proposals being voted on. (Images for the proposals are in the images subdirectory.)

ϒ app.js, the glue code connecting the web server layer and the smart contract layer. You can review the files for the web app by unzipping src.zip in the codebase for this chapter

and cloning it into the ballot-app folder, which contains index.js package.json src. The src directory contains the source for the web application part. Communication from the web client to the blockchain server is through JSON over RPC. Let's focus on analyzing app.js, which contains the handlers for the stimuli invoking the smart contract functions. I'll discuss this app.js code later for two reasons: this part will be different for different smart contracts and Dapp, and you'll have a better idea of the role of app.js after you run the Dapp and interact with it.



4) Explain with a neat diagram the Traditional application vs. blockchain Dapp with on-chain and off-chain data.

Figure 6.1 compares a traditional application and a blockchain-based application. The traditional system on the left has data stored in a filesystem or database. On the right is a blockchain application, with its familiar -app and -contract parts. In general, Dapp contract–generated data is stored on-chain, and data created and used by the Dapp app is stored off-chain. The function calls from a Dapp invoke a smart contract, the Txs are generated, and related artifacts are recorded on the blockchain. In figure 6.1, follow the arrows from the -app to understand the on-chain data. In this chapter, you'll explore this relationship between the on-chain and off-chain data.

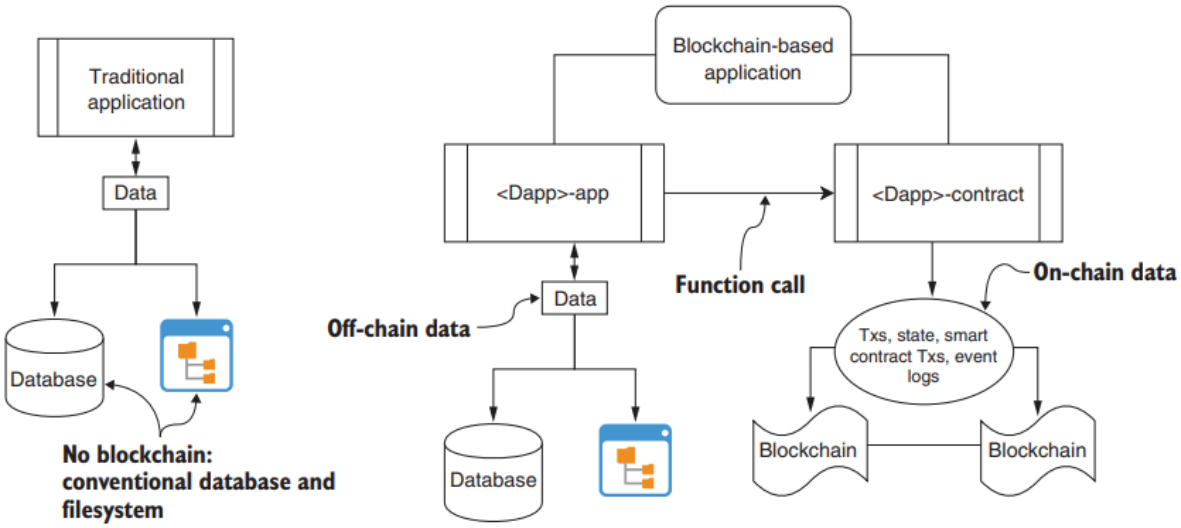


Figure 6.1 Traditional application vs. blockchain Dapp with on-chain and off-chain data

5) What is the importance of Security and privacy in blockchain.

6) Explain the various elements of trust and integrity with necessary diagram.

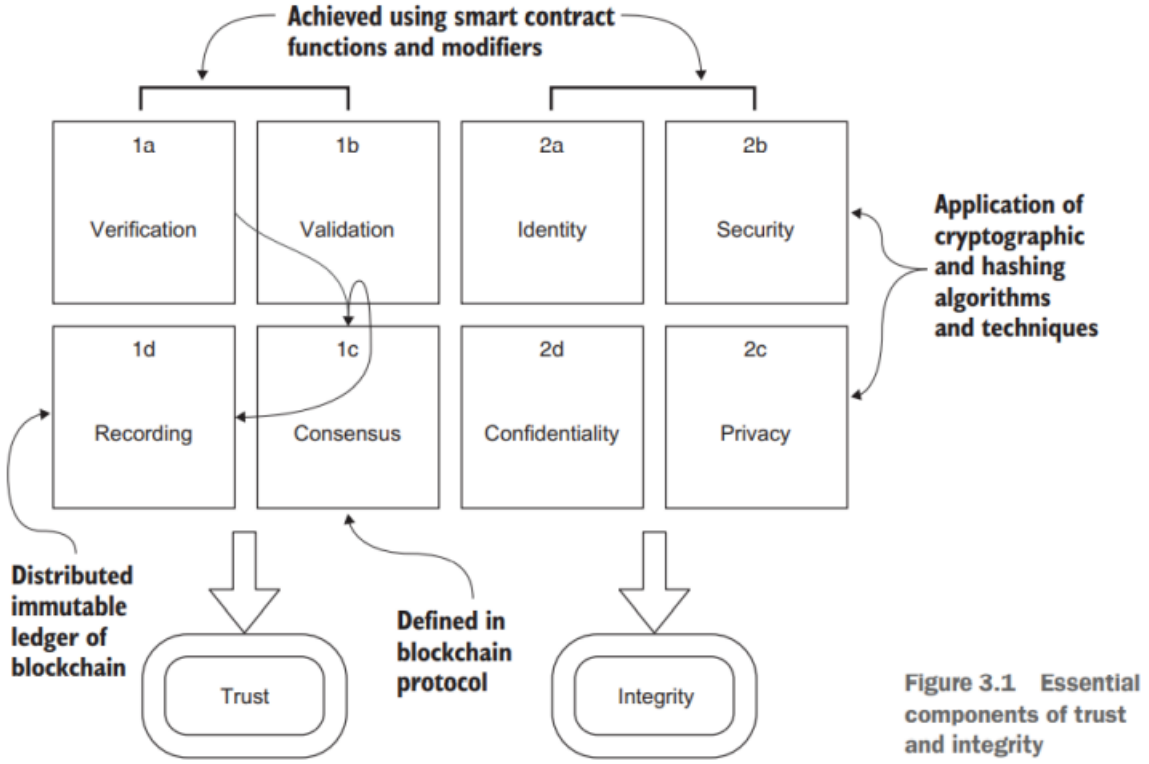


Figure 3.1 Essential components of trust and integrity

Trust is a measure of confidence in the credibility of a peer partic ipant in a system. Trust in a blockchain-based system is established by verifica tion and validation of relevant participant data and transactions, and by immutable recording of appropriate information done with the consensus of the stakeholders. Integrity, in the context of blockchain, means ensuring the security and privacy of data and confidentiality of transaction.

7) What is On-chain data in blockchain? Explain with a neat diagram the Elements of on-chain data.

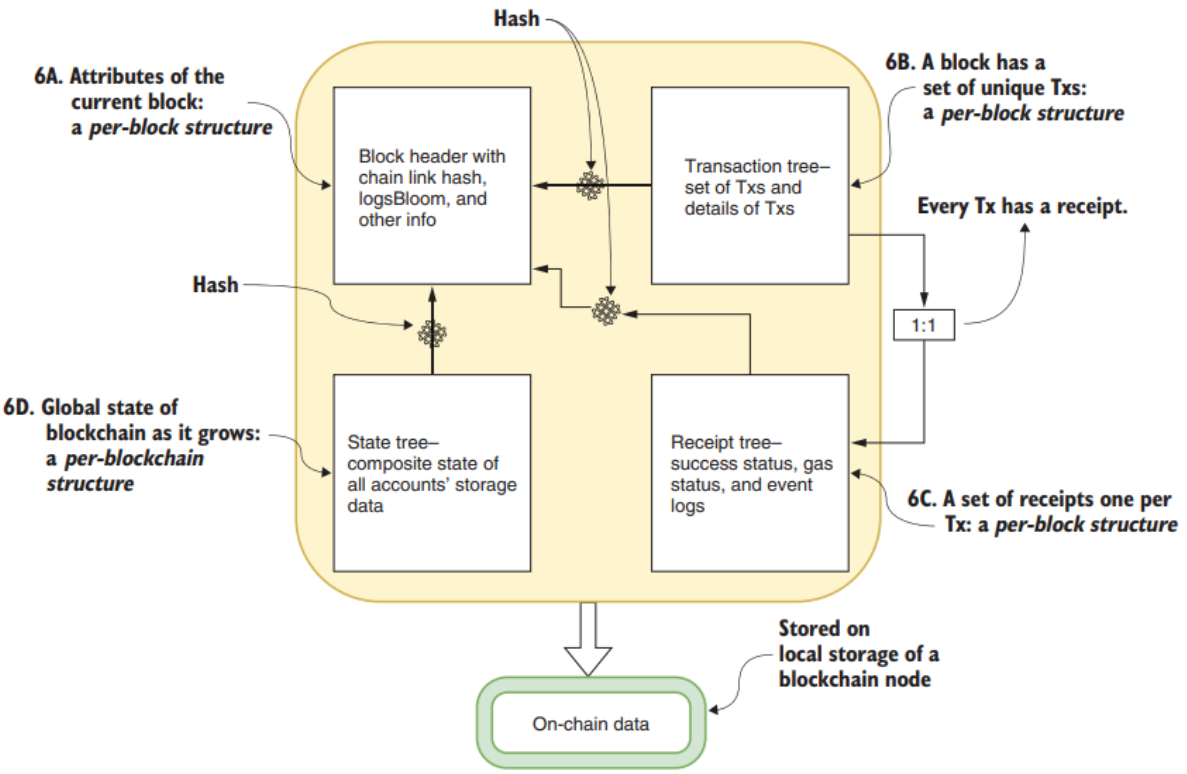
Transactions are not the only data stored on the blockchain. A blockchain protocol determines the different types of on-chain data. In Ethereum protocol, as shown in figure 6.2, a block is made up of several elements, each serving a specific purpose:

ϒ The blockchain header (6A) stores the attributes of the block.

ϒ Transactions (6B) store the details of the Txs recorded in the block.

ϒ Receipts (6C) store the execution results of Txs recorded in the block.

ϒ A composite global state (6D) stores all the data values or current state of the smart contract accounts.



8) Explain the basic concepts of Hashing with a proper example in Blockchain.

Hashing is a transformation that maps data of arbitrary sizes to a standard fixed-size value. The hash of data elements are computed by using a hash function, as shown here:

hash = hashFunction(one or more data items)

Using a logical XOR (exclusive OR) function as a simple hash function and two data items of a =1010 binary, b= 1100 binary, you get the hashed value of the two data items as 0110: hash value = xor(a=1010, b=1100) = 0110

Hashing is the process of mapping data of an arbitrary length to a fixed size by using a specially defined function called a hash function. Even a single bit change in the data elements changes the hash value of the data elements significantly. Any type of data, including a database or an image, can be succinctly represented by a hash of fixed length, as shown in figure 5.9.

A 256-bit data item and a strong hash function together provide a large, collision-free account address space. Collision-free means there is a high probability that no two values generated by the hash function will be the same and that you'll get a unique hash value when you apply the hashing function to the same data elements. This property is an important requirement of a hash function: you don't want to have the same identification number as your friend!

9) Demonstrate how Hashing can be used in signing document, Distributed ledger and how Hashes are used in Ethereum block header with a diagram.

Digital signing of documents:

For the digital signing of a document, a hash function is used to compute a hash of the document. This hash is used as the digital signature for the document and attached to the document by the sender. This signature can be verified later by the receiver of the document by recomputing the hash of the document and comparing it with the attached digital signature (hash)

Hashed data on distributed ledger:

Blockchain is not your regular database; it stores only the minimal data needed in its distributed ledger. Hashing helps here too! The blockchain isn't overloaded by a large document because only the hash value (representation) of the document can be stored on the chain. You'll learn more about the versatility of hashing when we develop our decentralized system model further with on-chain and off-chain data in chapter 6

Hashes in Ethereum block header:

Recall from chapter 1 that the blockchain is a tamperproof immutable ledger consisting of blocks that contain records of transactions, mutable state, logs, return values (receipts), and many other details, as shown in the Ethereum block header diagrams in figure 5.10. Txs, state, logs, and receipts are stored in a Merkle tree (trie) data structure, and the hash of this tree is stored in the header. The header also stores a hash of the previous block's header, forming a link to the previous block, constructing the chain, and enforcing immutability. Even a single bit change in the block's contents will change its hash significantly, thus breaking the chain; so as you can see, the block hashes are instrumental in realizing the immutability and integrity of the chain!

10)Construct a State transition FSM diagram and contract diagram for the various stages for an online Ballot system.

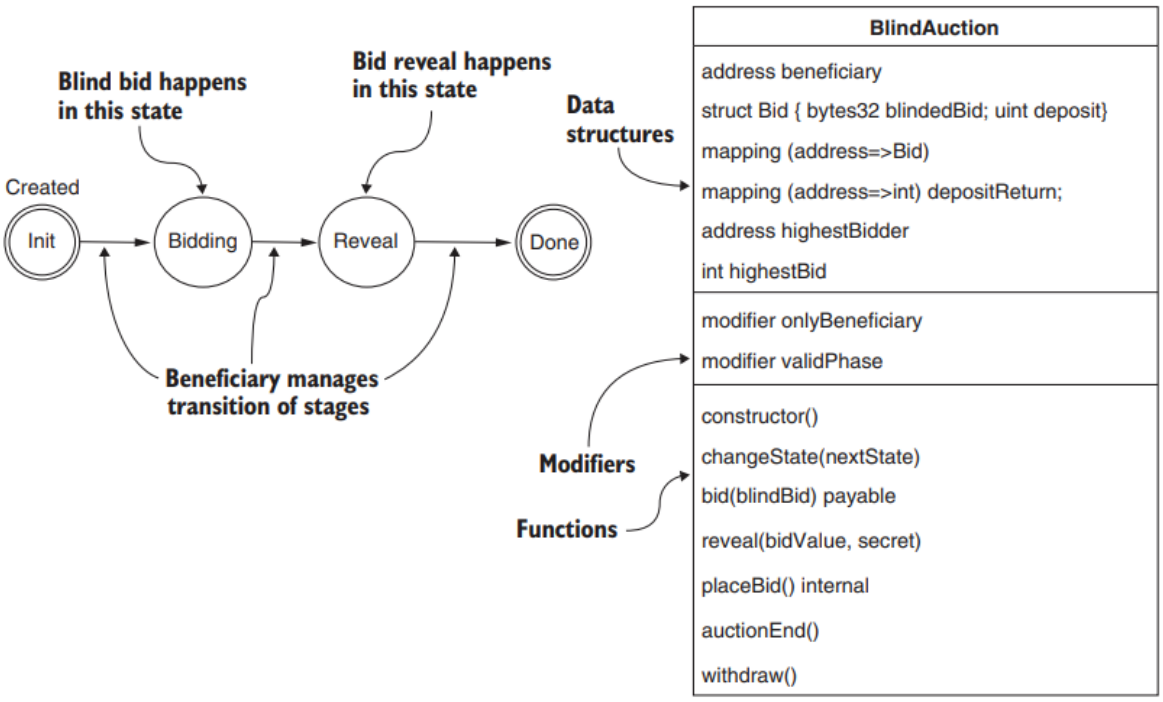


Figure 5.11 State transition FSM and contract diagram for a blind auction

11) Explain with a neat diagram how Off-chain data is stored on a variety of data sources.

Off-chain data is stored on a variety of data sources, some of which are shown in figure 6.10. A significant difference from on-chain data is that the types and uses of off-chain data are not determined by the blockchain protocol; this data is used by the non blockchain part of the larger system. Off-chain data can be anything from the output of a medical device to data in cloud storage. The types and formats of data sources are limitless and application-dependent. A typical scenario, however, is a regular database that works in tandem with the blockchain trust layer that is needed in a decentralized system of unknown peers.

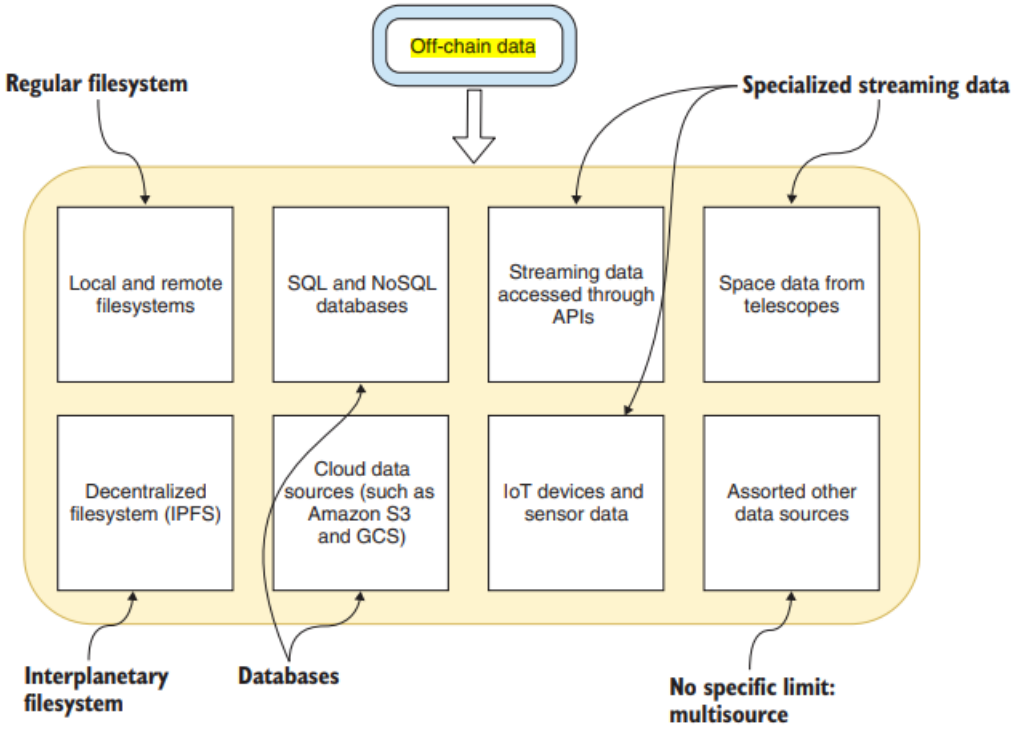


Figure 6.10 Different types of off-chain data

UNIT 5:

1.Explain with a neat diagram the role of web3 in the blockchain-based Dapp stack, and explain the layers in brief.

Where does web3 appear in your Dapp stack? Functions supported by web3 can be placed in two categories: those that support the core operations of the blockchain node and those that enable the decentralized application stack on the blockchain. Figure 7.1 illustrates the essential role played by web3 in these two modules, followed by a detailed explanation of each module.

The top part of figure 7.1 shows the application module, which has a web server and application code specified in app.js. The bottom part of figure 7.1 is the blockchain client node module, which provides the core blockchain services. Let's further analyze the layers of the stack in figure 7.1:

ϒ The top layer of the stack is a web client, but it could be any client—mobile or enterprise—that requires the services of the blockchain.

ϒ In the next layer, the web application's app.js uses the web3.js library to access the blockchain services.

ϒ The layer below is a traditional web server, listening to a port for client requests, and in this case implemented by the Node.js server.

ϒ Web3.js enables app.js application logic to connect to the underlying web3 provider in the blockchain node.

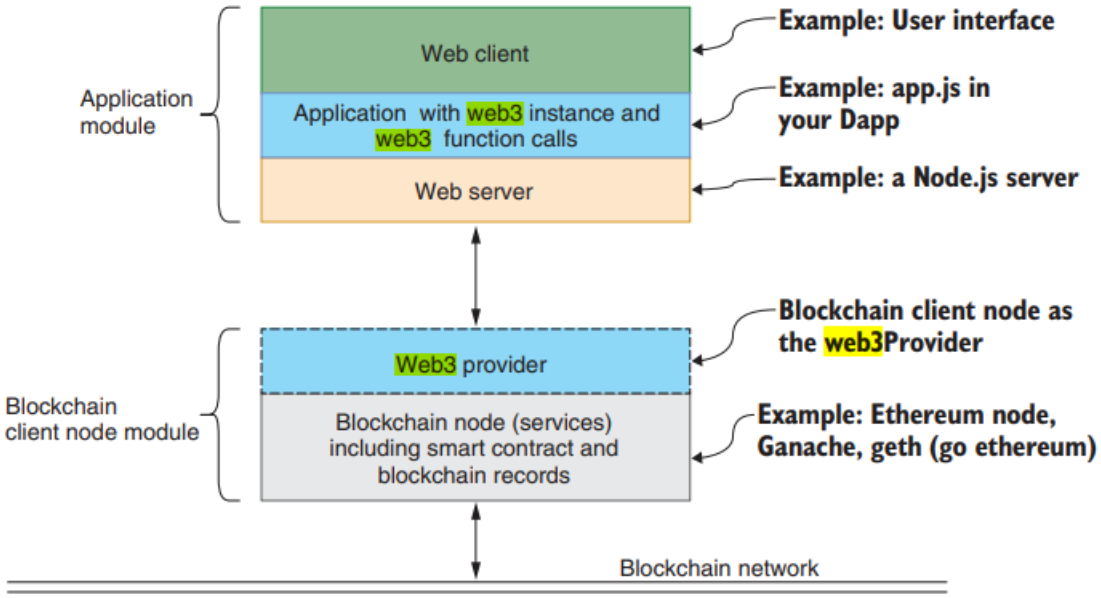


Figure 7.1 Role of web3 in the blockchain-based Dapp stack

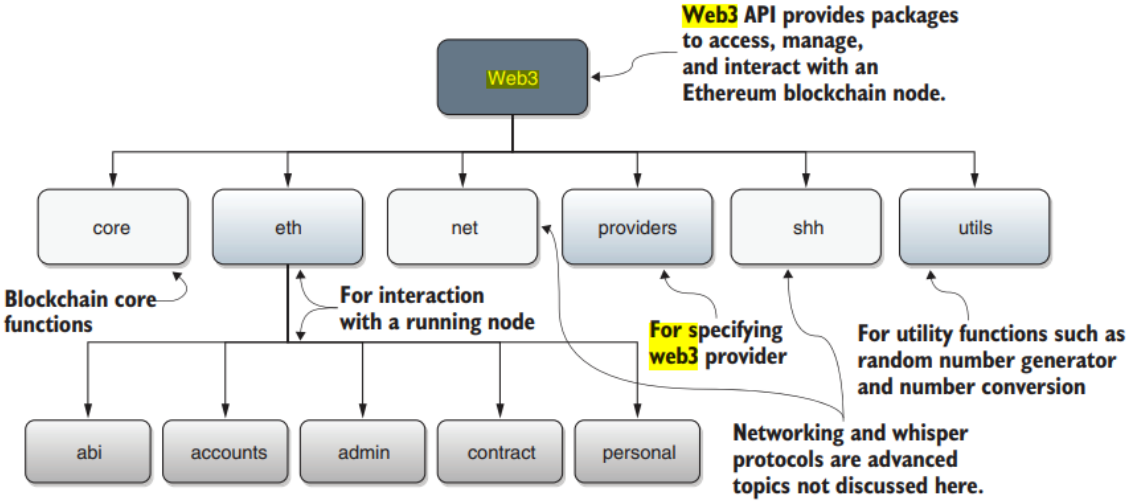
2. Illustrate with diagram Web3 API as a large unit with many packages representing various functions.

7.1.2 Web3 packages

Web3 API is a large unit with many packages representing various functions. It consists of six packages: `core`, `eth`, `net`, `providers`, `shh`, and `utils` (figure 7.2). You'll use `web3.eth`, `web3.providers`, and `web3.utils` in this chapter:

- The `eth` package and its subpackages enable an application to interact with accounts and smart contracts.
- The `providers` package lets you set a specific web3 provider, such as Ganache.
- The `utils` package provides a standard implementation of common utility functions for their uniform use by the Dapps.

Among the other packages, `web3.core` implements the core protocol for the operation of the blockchain, `web3.net` implements the networking aspects of transaction broadcasting and receiving, and `web3.shh` is for an advanced concept called *whisper protocol* that allows Dapps to communicate with one another.



3. What are the basic facts about a micropayment channel?

7.3 Micropayment channel

Micropayments are an age-old practice all over the world. Many local mom-and-pop economies depend on micropayments for daily living as well as for sustaining the local economy. These payments typically do not involve conventional financial institutions such as banks. With the advent of the digital age, efforts were focused on digitizing these micropayment methods, but they met limited success. The Bitcoin blockchain changed all that by proving the feasibility of online payments among unknown peers. With that breakthrough, interest in micropayments has been revived, and rightly so.

Here are some basic facts about a micropayment channel:

- It is defined by endpoints identified by the sender and receiver account addresses.
- It facilitates small (micro) and frequent payments between sender(s) and receiver(s).
- Payment values are less than the transaction fees charged on transactions on the main channel. (This characteristic is understandable.)
- The relationship between sender and receiver is temporary, typically terminated after payment is settled and synchronized with the main channel.

4. Explain a neat diagram the Role of web3 in the blockchain-based Dapp stack.

ANS IS SAME AS 1.

5. Explain with a neat diagram Relationship between the main channel and micropayment channel.

Figure 7.4 shows these concepts and the relationship between the on-chain main channel and off-chain side channel between two accounts. Anybody can join and leave the main channel, and any account can transact with any other account. Every transaction on the main channel is recorded on the blockchain. The main channel is permanent, as in Bitcoin's and Ethereum's main chains. Now look at the side channel in figure 7.4. The micropayment channel is an example of a side channel between selected accounts—in this case, between two accounts— and is temporary. The transactions between the side channel accounts are off-chain and not recorded on the chain until the side channel synchronizes with the main channel. This synchronization happens when one of the participating accounts sends a transaction on the main channel, capturing and summarizing the details of the offchain transactions. The side channel may be dissolved after synchronization with the main channel. The micropayment channel concept is discussed in the Solidity documentation and in many online publications.

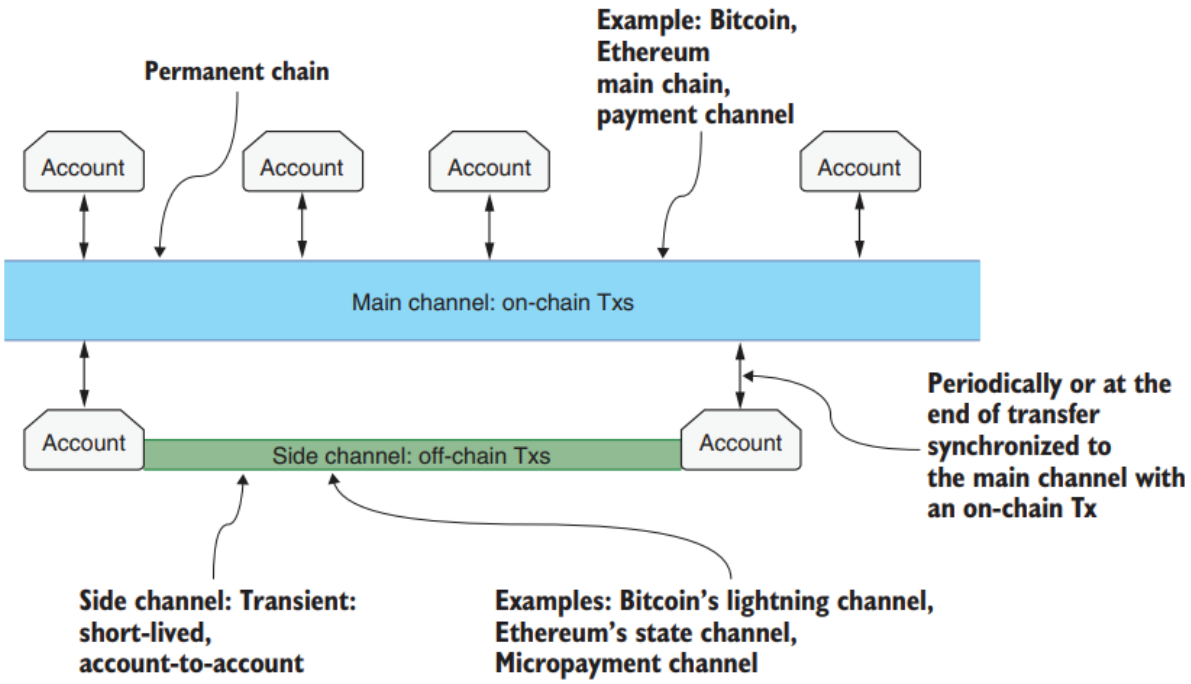


Figure 7.4 Relationship between the main channel and micropayment channel

6. Explain in details the blockchain use cases (i) Supply chain, (ii) Health care. Also address advantage and disadvantages.

7. Illustrate with a neat diagram the End-to-end process for public deployment steps in deploying a Dapp on Infura.

To deploy a smart contract on Infura and a public network such as Ropsten, you need a few items, shown in figure 8.5. This roadmap begins from account generation with a private-public key pair and ends with decentralized end-user interaction. In earlier chapters, the accounts were precreated and made available through a test chain such as Ganache. In this chapter, however, you are going to start from the beginning: generating the private key.

Figure 8.5 shows the steps that will be described in detail in this chapter, using two of the familiar Dapps: blind auction and MPC. I'll use the blind auction to introduce the public deployment process and a second Dapp MPC to repeat and reinforce the steps learned with the blind auction Dapp. The figure illustrates the process of deploying a Dapp on Infura-provisioned Ethereum blockchain nodes that are networked via the Ropsten test network. Although it's possible to deploy the smart contracts of the two Dapps (blind auction and MPC) on a single Infura project, I've chosen to use two projects offered by Infura to illustrate the separation of unrelated projects.

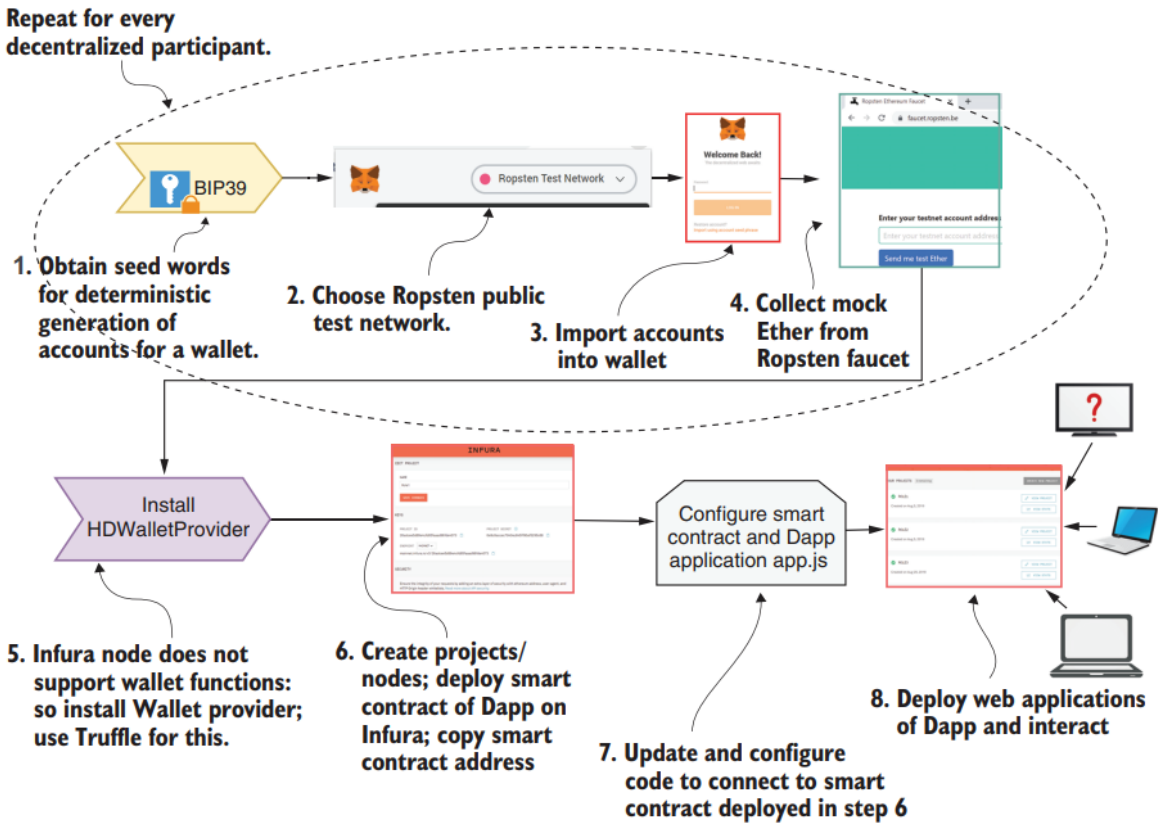


Figure 8.5 Steps in deploying a Dapp on Infura

8. Explain with a neat diagram the network-of-nodes concept also knows as a network of Ethereum nodes.

Nodes and networks

When we consider applications such as email and messaging, most of us see only the user interface of the client (the email client, for example). Behind most applications are servers—application servers that manage the emails, store them, format, and filter them, and so on. Similarly, the nodes are servers for blockchain services, as you learned in chapter 6. The nodes manage blockchain-related operations. A network connects nodes. The operations on the network of nodes (chapter 1) are controlled by a protocol or a set of rules. Figure 8.1 is replicated here from chapter 1 to refresh the network-of-nodes concept.

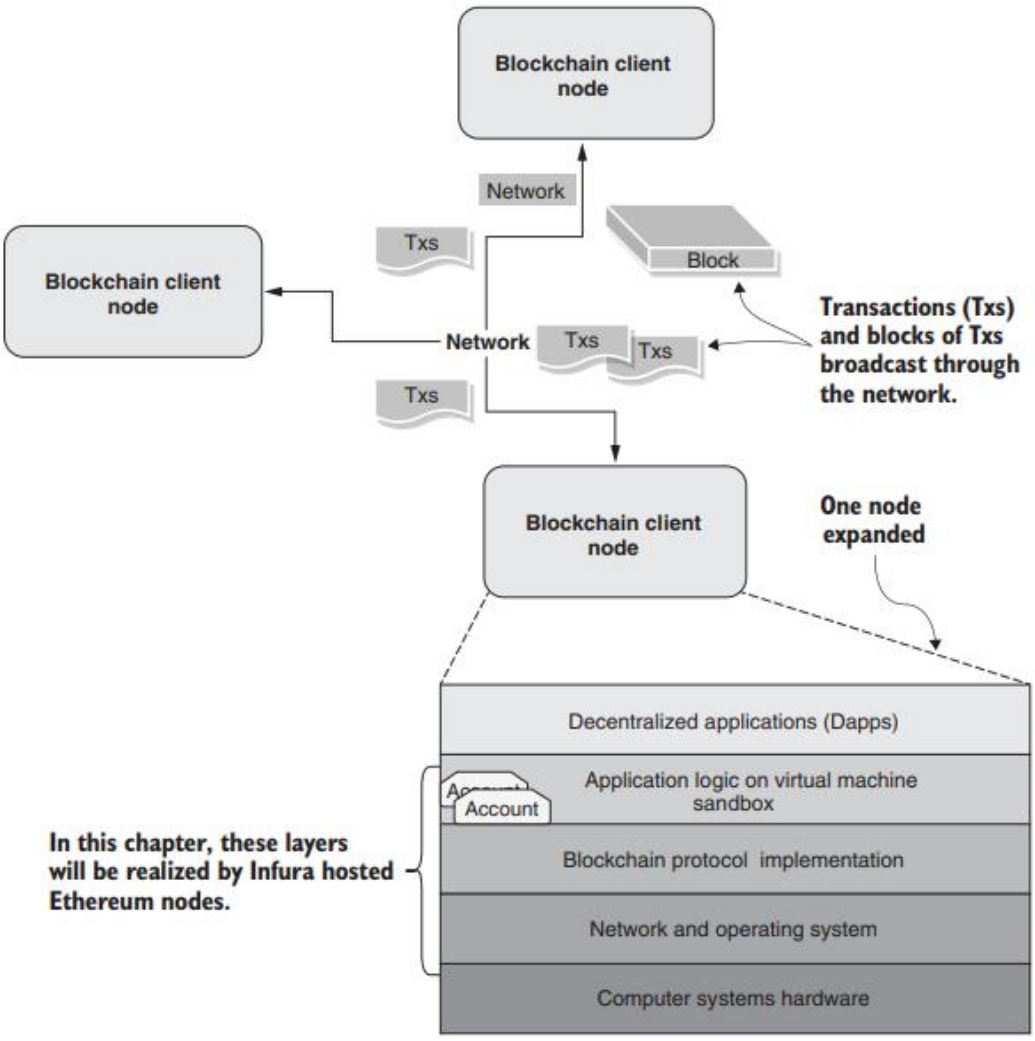


Figure 8.1 A network of Ethereum nodes (adapted from figure 1.6)

9. Explain with a neat diagram the various services offered by Infura in support of the expanding ecosystem of blockchain-based Dapps.

Figure 8.2 provides an overview of the various services offered by Infura in support of the expanding ecosystem of blockchain-based Dapps. In the bottom-left corner is the familiar Ganache that you used as a test node for the Dapps in earlier chapters—your blockchain development on training wheels. You can take the training wheels off with Infura. Most of figure 8.2 is about services offered by the Infura infrastructure, the primary function of which is to provision Ethereum blockchain nodes. It also makes available the endpoints and the API to connect to the nodes. A public network connects these nodes. You'll use the Ropsten network, as shown in figure 8.1. Infura also provides other services, such as a gateway to connect to IPFS (Interplanetary File System) that can serve as the decentralized store of some of the off-chain data. In this chapter, you'll focus on the endpoint for the Ethereum network and developing with the API to access it.

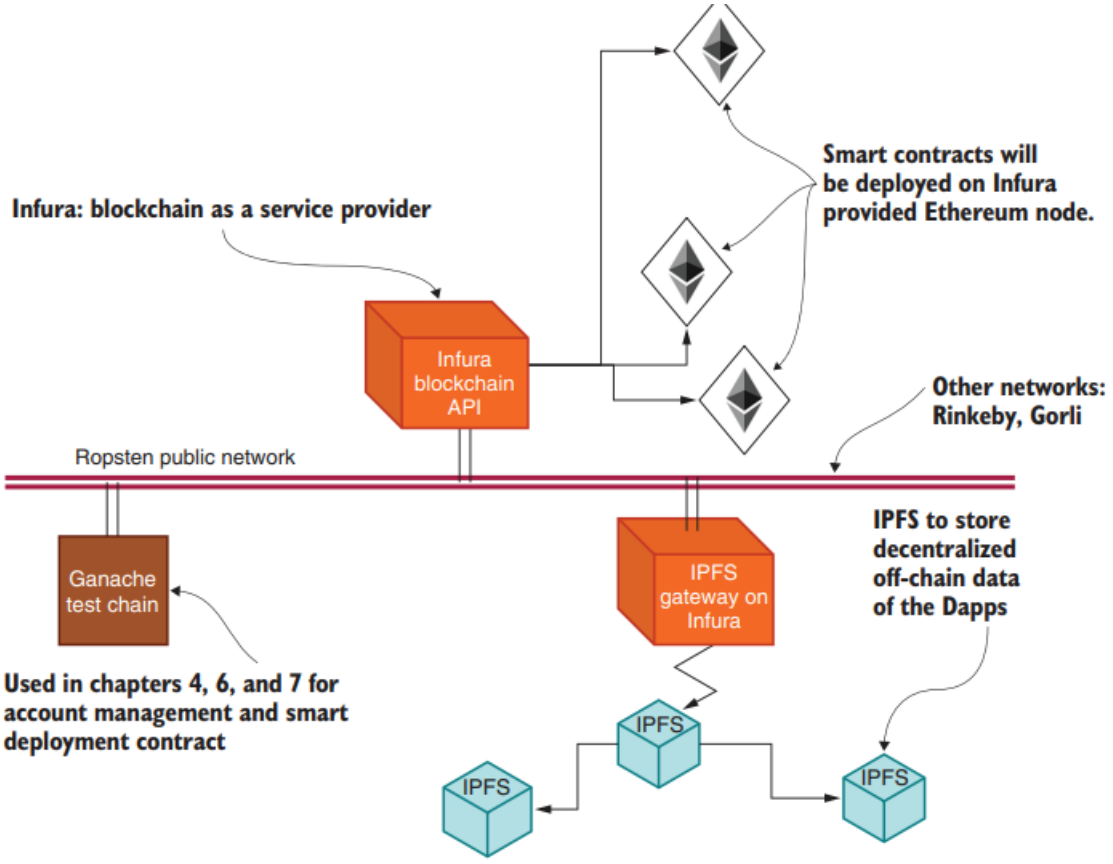


Figure 8.2 Expanding blockchain ecosystem: Infura, Ropsten, and IPFS

10. What is Micropayment channel? Compare Traditional banks vs. blockchain payment channels in details.

Micropayment channel

Micropayments are an age-old practice all over the world. Many local mom-and-pop economies depend on micropayments for daily living as well as for sustaining the local economy. These payments typically do not involve conventional financial institutions such as banks. With the advent of the digital age, efforts were focused on digitizing these micropayment methods, but they met limited success. The Bitcoin blockchain changed all that by proving the feasibility of online payments among unknown peers. With that breakthrough, interest in micropayments has been revived, and rightly so.

Here are some basic facts about a micropayment channel:

- It is defined by endpoints identified by the sender and receiver account addresses.
- It facilitates small (micro) and frequent payments between sender(s) and receiver(s).
- Payment values are less than the transaction fees charged on transactions on the main channel. (This characteristic is understandable.)
- The relationship between sender and receiver is temporary, typically terminated after payment is settled and synchronized with the main channel.

Table 7.1 Traditional banks vs. blockchain payment channels

Traditional banking system	Blockchain payment channel
Account creation—For millions of people, it is impossible to open an account in the traditional banking system due to a lack of credentials, such as a job or a home address.	Blockchain is built on this very concept of account-based identity and peer-to-peer interaction between unknown participants. It can facilitate quick account (digital identity) creation.
Small payments—Payment amounts involved may be too small to warrant account creation.	Blockchain by design naturally supports online digital micropayments.
Check-cashing fees—Payment in regular checks for every collected bin of plastics may generate numerous checks, and check-cashing fees may be higher than the payments.	The blockchain approach uses a cumulative check-payment approach to address cashing of online digital checks, which helps minimize fees.
Check verification process—Signing checks and signature verification in a traditional system can be cumbersome for a large number of checks.	Blockchain uses hashing and cryptographic functions for automatic digital signature verification at scale.

11. Explain with a neat diagram the Traditional vs. blockchain-based system with differences highlighted.

Let's compare the two approaches. Figure 7.6 shows a modified version of figure 7.5, but the solution depicted is that of a blockchain-based micropayment. The differences between the blockchain version and the traditional system are highlighted in figure 7.6. Take a few minutes to review the figures carefully, noting the differences in the blockchain application for MPC.

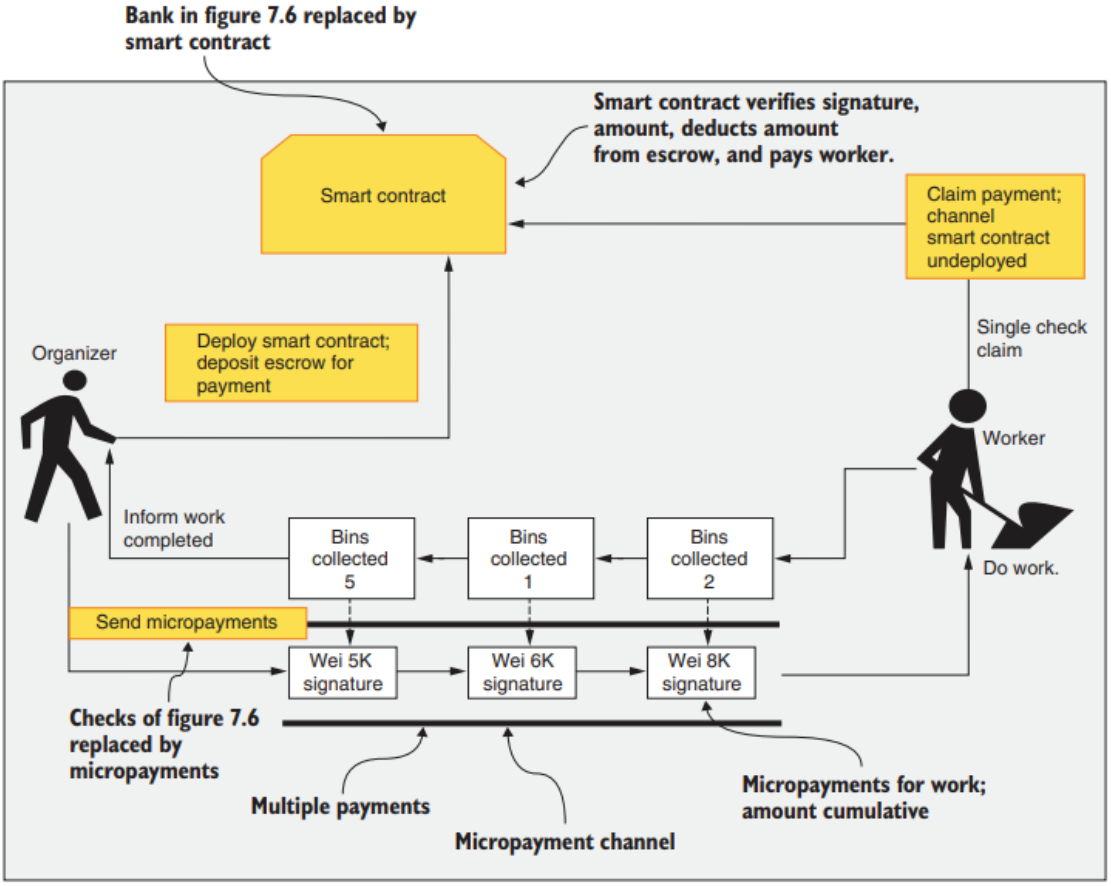


Figure 7.6 Traditional vs. blockchain-based system with differences highlighted