17
Scalability

Even though multiple use cases and Proof-of-Concept (PoC) systems have been developed using blockchains, and the technology works well for many scenarios, there is still a need to address some fundamental limitations present in blockchains to make the technology more adoptable.

At the top of the list of these issues comes **scalability**, which is a significant limitation. The lack of scalability is a general concern, where blockchains do not meet the adequate performance levels expected by users when the chain is used on a large scale.

There are several techniques to solve the scalability issue in blockchains, which will be discussed in detail in the following sections. Along the way, we'll cover the following topics:

- What is scalability?
- Blockchain scalability trilemma
- Methods to improve blockchain scalability
- Layer 0, 1, 2, and beyond

What is scalability?

This is the single most important problem in blockchain, which could mean the difference between the wider adaptability of blockchains or limited private use only by consortiums.

We can define scalability as the ability of a system to preserve or readjust its attributes to adapt to the increased demand for its resources as its utilization increases.

As a result of substantial research in this area, many solutions have been proposed, which are discussed in the following section.

From a theoretical perspective, the general approach toward tackling the scalability issue generally revolves around protocol-level enhancements. For example, a commonly mentioned solution to Bitcoin scalability is to increase its block size. This would mean that a larger number of transactions can be batched in a block, resulting in increased scalability.

Other proposals include solutions that offload certain processing to off-chain networks; for example, off-chain state networks.

Based on the solutions mentioned above, generally, the proposals can be divided into two categories: **on-chain solutions**, which are based on the idea of changing fundamental protocols on which the blockchain operates, and **off-chain solutions**, which make use of off-chain network resources to enhance the blockchain.

While there are many solutions to the scalability problem now, and the situation is not as bad as the first decade after Bitcoin's introduction, note that scalability is not easy to achieve. Some tradeoffs need to be made for a scalability solution to work. It has been conjectured that blockchain's three objectives, decentralization, scalability, and security, cannot be achieved simultaneously – the so-called blockchain trilemma.

Blockchain trilemma

Based on the problem presented by Vitalik Buterin, an Ethereum co-founder, it is understood that only two of the three main core properties of a blockchain can be utilized at a time. These three core properties are the following:

- **Decentralization** – This means that the system runs for participants with access only to normal resources, e.g., an entry-level computer with the usual hardware. In other words, no specialized hardware is required to participate in the network, a requirement that could tilt the scales in favor of those with greater resources.
- **Scalability** – This means that the overall system is able to process a larger number of transactions than the number of transactions an individual participant can process. In simpler words, the system is able to perform a high number of transactions.

- **Security (or consistency)** – This means that the system is secure against adversaries even with high levels of resources (though not unlimited).

The trilemma means that the three key goals of a scalable blockchain – security, scalability, and decentralization – cannot be fully achieved; we must sacrifice at least one of them to achieve the other two. In other words, the blockchain trilemma states that a simple blockchain architecture can only achieve two out of three properties, e.g. if you need a secure and decentralized blockchain, you need to sacrifice scalability. Similarly, if you want to achieve scalability and security, then decentralization must be sacrificed.

This is known as the **blockchain trilemma** and is seen as a fundamental problem that needs to be addressed before the global adoption of Ethereum and other blockchains can be achieved.

This concept is visualized in the following diagram:

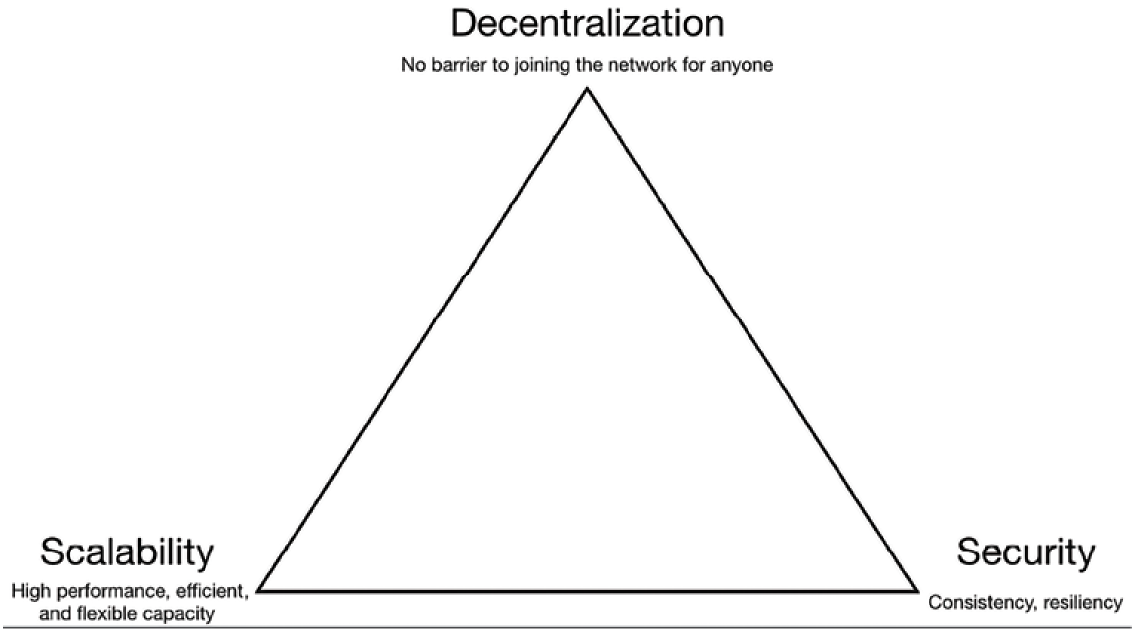
Decentralization
No barrier to joining the network for anyone
Scalability
High performance, efficient, and flexible capacity
Security
Consistency, resiliency

Figure 17.1: Blockchain trilemma

Traditional monolithic blockchains like Bitcoin perform all blockchain operations including data availability, consensus, settlement, and execution on-chain. Data availability ensures that block header data is publicly available so that anyone can recreate the state for verification purposes. Consensus, as we saw in *Chapter 5*, *Consensus Algorithms*, ensures consistency of the chain by achieving agreement between participants on the inclusion and ordering of the transactions. Settlement means finalization of the transactions on-chain. Finally, we have execution, which means the running (computation) of a transaction to transition it from the existing state to the new state.

There are three bottlenecks that adversely affect blockchain performance: execution, verification, and communication. By design, on monolithic blockchains, every participant has to accept, process, forward, and execute every transaction, which makes the network fundamentally slow.

Intensive research is underway and significant advancement has been made in this regard in the blockchain research community. With solutions such as layer 2, multichain networks, and layer 1 improvements, the scalability issue has been addressed to some extent, although there are still some tradeoffs to be made. Even with all the aforementioned advances, blockchain scaling is not straightforward. There is no one-size-fits-all solution due to differing requirements and use cases. In future, it is likely that multiple scalability solutions will coexist. Multiple solutions also lead to disparate heterogeneous architectures, which makes developer adoption somewhat challenging. Also, although good progress is being made, the layer 2 landscape undergoes constant evolution, which makes it difficult to adopt a single solution for the long term.

An important thing to understand is that due to the blockchain trilemma, when high performance and efficiency are achieved, either security or decentralization must be given up to some extent. All three properties cannot be achieved at the same time.

This is, however, somewhat debatable, as proponents of layer 1 scaling such as Solana are of the opinion that all three properties can be achieved, and Solana in particular claims to have demonstrated this with the Solana blockchain. However, there is some concern that this solution is not as decentralized as PoW networks, because the number of validators in the former is not enough to enable full decentralization. Roughly speaking, as long as more than 50% of the validator network is under the control of an honest majority, the network is expected to remain secure.

Note that in order to achieve performance and speed, it might be acceptable to give up some level of decentralization as a tradeoff, but it should not be so much that it compromises security and consequently results in transaction censorship.

> We discussed the Nakamoto coefficient in *Chapter 2, Decentralization*, on decentralization. It turns out that chains like Solana have a good Nakamoto coefficient (around 31), but perhaps not enough to thwart cartel formation attacks. Other chains like Polygon have an even lower Nakamoto coefficient (about 4), which is not suitable. Here, too much decentralization is given up in favor of efficiency (speed), which is not ideal. Solana appears to be a better choice, with a Nakamoto coefficient of 31. These stats are tracked online at https://nakaflow.io.

There's a lot of research in this area, and the aim is to address this trilemma and find the right balance between all three properties instead of compromising and only choosing two of the three. In this regard, Algorand has proposed a key solution that claims to have solved the blockchain trilemma without sacrificing any of the three objectives. The trick is to enable random validator selection to pick up the next set of nodes to add blocks. The algorithm can choose anyone randomly to become the next block-adding validator, thus not giving up decentralization. The algorithm achieves scalability by randomly selecting a small set of representatives (committee) that run the protocol instead of using all nodes. This small set of block proposers and verifiers allows users to only have to receive a small, fixed number of messages to achieve consensus for the next block, thus achieving speed and scalability. This is called a pure proof-of-stake algorithm.

> Algorand is a very important development and has in fact refuted the blockchain trilemma. With Algorand it is possible to achieve all three properties, i.e., scalability, decentral-

> ization, and security. We'll introduce Algorand in *Chapter 23, Alternative Blockchains*.

With the advent of **layer 2 protocols**, **innovative layer 1 enhancements**, **faster consensus mechanisms**, and other techniques like **sharding** and **parallelization**, a somewhat balanced combination of all three properties of decentralization, scalability, and security (consistency) can be achieved.

Now, we discuss some methods for improving scalability.

Methods for improving scalability

As this is a very active area of research, over the years many techniques and proposals have been made to address the blockchain scalability problem. In this section, we'll introduce many of these techniques.

We can divide approaches to solving the scalability issue into four main categories based on the *layer* in the blockchain stack they operate on.

We describe these categories here:

- **Layer 0** methods (also called *multichain methods*, *modular blockchains*, or *polylithic blockchain architecture*) involve a multi-chain ecosystem, enabling interoperability between chains and allowing developers to create custom chains.
- **Layer 1** methods are also called on-chain methods, where the blockchain and network protocol themselves are enhanced to improve scalability. This layer represents the blockchain and the network.
- **Layer 2** methods, or off-chain methods, are where mechanisms that are not part of the blockchain and exist outside of the main blockchain are used to improve the scalability of the blockchain.
- **Layer 3** methods (also called *multilayer scaling* or *hyperscaling*) are based on the fundamental observation that if layer 2, using rollups, can compress data up to n times, then it could be possible to add another layer on top and achieve data compression multiple (up to n x n) times. Note that there are some other views on what's beyond layer 2,

such as proposals made by StarkWare – so-called fractal scaling, which introduces application-specific layer 3s that are built recursively over layer 2.

Scalability has two facets. One is increasing the processing speed of transactions to achieve better **transactions per second** (**TPS**), and the other is the increase in the number of nodes on the network. Both are desirable in many situations; however, transaction speed is more sought after on public networks, as node scalability is not an issue in public blockchain networks. This is evident from blockchain networks like Ethereum and Bitcoin where thousands of nodes operate on the network, but the transaction throughput is roughly around 7 and 15 TPS respectively.

Layer 0 – multichain solutions

We'll describe some of the layer 0, or network layer, solutions in the following sections.

Layer 0 solutions emerged because of multichain architectures (sometimes called a "blockchain of blockchains") where a central chain acts as a relay between multiple sidechains. Some prime examples of layer 0 chains are **Polkadot**, **Avalanche**, and **Cosmos**.

A brief introduction to Polkadot is given below.

Polkadot

Polkadot is a modern blockchain protocol that connects a network of purpose-built blockchains and allows them to operate together. It is a heterogenous multichain ecosystem with shared consensus and shared state.

Polkadot has a central main chain called the Relay Chain. This Relay Chain manages the parachains – the heterogenous shards connected to the Relay Chain. The Relay Chain holds the states of all parachains. All these parachains can communicate and share security, leading to a better and more robust ecosystem.

As the parachains are heterogenous, they can serve different purposes. For example, one chain can be specifically for smart contracts, another for gaming, another for providing some public service, and so on. The Relay Chain is secured by nominated proof of stake.

This Relay Chain and parachain-based sharding architecture with many blockchains running in parallel is how Polkadot achieves scalability. Sharding allows many computations to be executed in parallel. Polkadot can also connect with private chains, other public chains, consortium chains, and oracles, which enables interoperability and results in increased network scalability, improved “network effect,” and overall throughput.

The Relay Chain’s validators produce blocks, communicate with parachains, and finalize blocks. On-chain governance through stake-based voting schemes decides what the ideal number of validators should be:

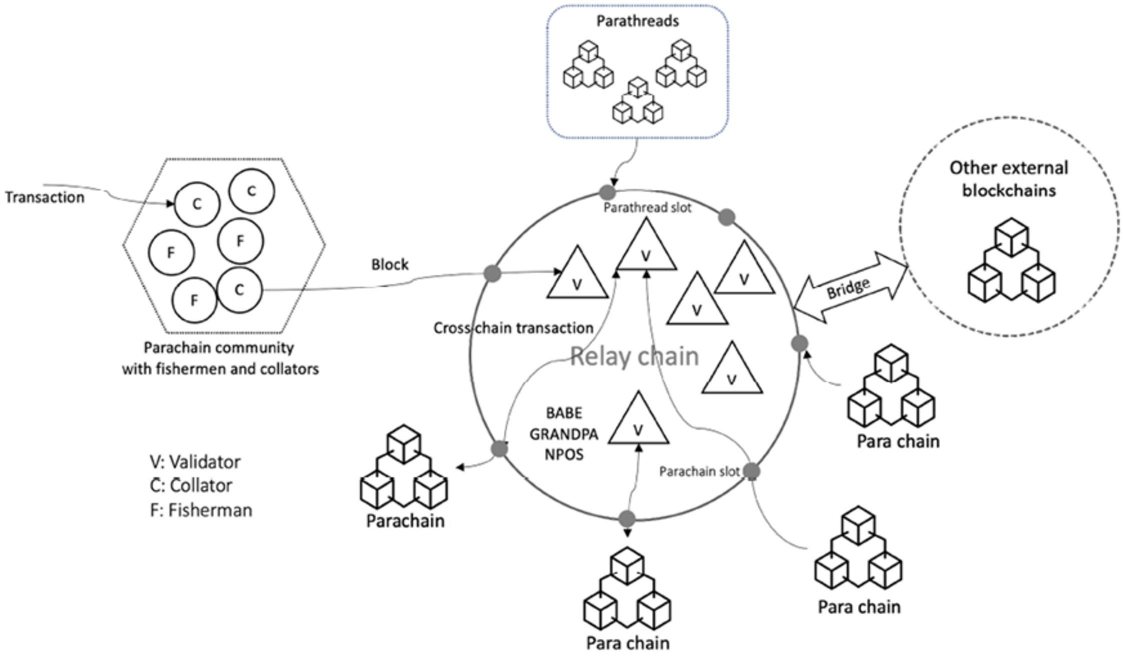

Figure 17.2: A depiction of the Polkadot network

Polkadot aims to be able to communicate with other blockchains as well. For this purpose, bridges are used that connect parachains to external blockchains such as Bitcoin and Ethereum.

There are several components in Polkadot. The Relay Chain is the main chain responsible for managing parachains, cross-chain interoperability, interchain messaging, consensus, and security.

As shown in *Figure 17.2*, the network consists of nodes and roles. Nodes can be light clients, full nodes, archive nodes, or sentry nodes. Light clients consist of only runtime and state. Full nodes are pruned at configurable intervals. Archive nodes keep the entire history of blocks, and sentry nodes protect validators and thwart DDoS attacks to provide security to the Relay Chain.

There are several roles that nodes can perform: validator, nominator, collator, and fisherman. Validators are the highest level in charge of the system. They are block producers, and to become block producers, they need to provide a sufficient bond deposit. They produce and finalize blocks and communicate with parachains. Nominators are stakeholders and contribute to the validators' security bond. They trust the validators to "be good" and produce blocks. Collators are responsible for transaction execution. They create unsealed but valid blocks for validators. Fishermen nodes are used to detect malicious behavior. Fishermen are rewarded for providing proof of the misbehavior of participants.

Parachains are heterogenous blockchains connected to the Relay Chain. These are fundamentally the execution core of Polkadot. Parachains can exist with their own runtimes, called application-specific blockchains. Another component called a parathread is a blockchain that works within the Polkadot host and connects to the Relay Chain. They can be thought of as pay-as-you-go chains. A parathread can become a parachain via an auction mechanism. Bridges are used to connect parachains with external blockchain networks like Bitcoin and Ethereum.

Next, let's move on to some core on-chain solutions, or *layer 1* solutions.

Layer 1 – on-chain scaling solutions

In this section, we'll describe layer 1 (network-level; on-chain), solutions, which target core blockchain elements such as blocks, transactions, and other on-chain data structures to address the scalability problem.

Kadcast

This is a new protocol that enables fast, efficient, and secure block propagation for the Bitcoin blockchain network.

> More information is available in the paper: Rohrer, E. and Tschorsch, F., 2019, October. *Kadcast: A structured approach to broadcast in blockchain networks*. In Proceedings of the 1st ACM Conference on Advances in Financial Technologies (pp. 199-213).
>
> https://dl.acm.org/doi/pdf/10.1145/3318041.3355469

---

bloXroute

Another network layer solution is bloXroute, which aims to address scalability problems by creating a trustless blockchain distribution network.

> More information about bloXroute is available at https://bloxroute.com and in the following whitepaper:
>
> https://bloxroute.com/wp-content/uploads/2019/11/bloXrouteWhitepaper.pdf

---

Another option that has been proposed to improve scalability is transaction parallelization, which we introduce next.

Transaction parallelization

Usually, in blockchain designs, transactions are executed sequentially. For example, in Ethereum, all transaction execution is sequential, which allows it to be safe and consistent. This also raises the question that if, somehow, transaction executions can be done in parallel without compromising the consistency and security of the blockchain, it would result in much better performance.

Some suggestions have already been made for Ethereum, for example:

- Easy parallelizability, which introduces a new type of transaction: https://github.com/ethereum/EIPs/issues/648.

- Transaction parallelizability in Ethereum, which proposes a mechanism to enable parallel transaction execution in Ethereum. The paper is available here: https://arxiv.org/pdf/1901.09942.pdf.

Parallel transactions are supported on several blockchain platforms including Hyperledger Sawtooth and Solana. Solana allows parallel execution of smart contracts using a runtime named *Sealevel*.

Increase in block size

This is the most debated proposal for increasing blockchain performance (transaction processing throughput). Currently, Bitcoin can process only about three to seven transactions per second, which is a major inhibiting factor in adapting the Bitcoin blockchain for processing microtransactions. Block size in Bitcoin is hardcoded to be 1 MB, but if the block size is increased, it can hold more transactions and can result in faster confirmation time. There are several **Bitcoin Improvement Proposals** (**BIPs**) made in favor of block size increase. These include `BIP 100`, `BIP 101`, `BIP 102`, `BIP 103`, and `BIP 109`.

> An interesting account of historic references and discussion is available at https://en.bitcoin.it/wiki/Block_size_limit_controversy.

In Ethereum, the block size is not limited by hardcoding; instead, it is controlled by a gas limit. In theory, there is no limit on the size of a block in Ethereum because it's dependent on the amount of gas, which can increase over time. This is possible because miners are allowed to increase the gas limit for subsequent blocks if the limit has been reached in the previous block. Bitcoin **Segregated Witness** (**SegWit**) has addressed this issue by separating witness data from transaction data, which resulted in more space for transactions. Other proposals for Bitcoin include Bitcoin Unlimited, Bitcoin XT, and Bitcoin Cash. You can refer to *Chapter 6*, *Bitcoin Architecture*, for more details.

For more information on Bitcoin proposals, refer to the following addresses:

`https://www.bitcoinunlimited.info`

`https://www.bitcoincash.org`

Block interval reduction

This is another proposal about reducing the time between each block generation. The time between blocks can be decreased to achieve faster finalization of blocks, but it may result in less security due to the increased number of forks. Ethereum has achieved a block time of approximately 14 seconds.

This is a significant improvement from the Bitcoin blockchain, which takes 10 minutes to generate a new block. In Ethereum, the issue of high orphaned blocks resulting from shorter times between blocks is mitigated by using the **Greedy Heaviest Observed Subtree** (**GHOST**) protocol, whereby orphaned or stale blocks (also called uncles in the Ethereum chain) are also included in determining the valid chain. Once Ethereum moves to **Proof of Stake** (**PoS**), this will become irrelevant as no mining will be required and almost immediate finality of transactions can be achieved.

Invertible Bloom Lookup Tables

This is another approach that has been proposed to reduce the amount of data required to be transferred between Bitcoin nodes. **Invertible Bloom Lookup Tables** (**IBLTs**) were originally proposed by Gavin Andresen, and the key attraction of this approach is that it does not result in a hard fork of Bitcoin if implemented. The key idea is based on the fact that there is no need to transfer all transactions between nodes; instead, only those that are not already available in the transaction pool of the syncing node are transferred. This allows quicker transaction pool synchronization between nodes, thus increasing the overall scalability and speed of the Bitcoin network.

Sharding

Sharding is not a new technique and has long been used in distributed databases such as MongoDB and MySQL for scalability. The key idea behind sharding is to split up the tasks into multiple chunks that are then processed by multiple nodes. This results in improved throughput and reduced storage requirements. In blockchains, a similar scheme is employed, whereby the state of the network is partitioned into multiple shards. The state usually includes balances, code, nonce, and storage. Shards are loosely coupled partitions of a blockchain that run on the same network. There are a few challenges related to inter-shard communication and consensus on the history of each shard. This is an area of active research and has been extensively studied in the context of scaling Ethereum.

Private blockchains

Most private blockchains are inherently faster because no real decentralization is required and participants on the network do not need to mine using PoW; instead, they can only validate transactions. This can be considered as a workaround to the scalability issue in public blockchains; however, this is not the solution to the scalability problem. Also, it should be noted that private blockchains are only suitable in specific areas and setups such as enterprise environments, where all participants are known.

Block propagation

In addition to the preceding proposal, pipelining of block propagation has also been suggested, which is based on the idea of anticipating the availability of a block. In this scheme, the availability of a block is already announced without waiting for actual block availability, thus reducing the round-trip time between nodes.

Finally, the problem of long distances between the transaction originator and nodes also contributes toward the slowdown of block propagation. It has been shown in research conducted by Christian Decker et al. that connectivity increases can reduce the propagation delay of blocks and transactions. This is possible because, if at any one time the Bitcoin node is connected to many other nodes, it can speed up the information propagation on the network.

An elegant solution to scalability issues will most likely be a combination of some or all of the aforementioned general approaches. A few initiatives undertaken in order to address scalability and security issues in blockchains are now almost ready for implementation or have already been implemented. For example, Bitcoin SegWit is a proposal that can help massively with scalability and only needs a soft fork in order for it to be implemented. The key idea behind so-called SegWit is to separate signature data from the transactions, which resolves the transaction malleability issue and allows block size increase, thus resulting in increased throughput.

Bitcoin-NG

Another proposal, Bitcoin-NG, based on the idea of microblocks and leader election, has gained some attention recently. The core idea is to split blocks into two types, namely leader blocks (also called key blocks) and microblocks:

- **Leader blocks**: These are responsible for PoW, whereas microblocks contain actual transactions.
- **Microblocks**: These do not require any PoW and are generated by the elected leader every block-generation cycle. This block-generation cycle is initiated by a leader block. The only requirement is to sign the microblocks with the elected leader's private key. The microblocks can be generated at a very high speed by the elected leader (miner), thus resulting in increased performance and transaction speed.

On the other hand, the *Ethereum 2.0 Mauve Paper*, written by Vitalik Buterin and presented at Ethereum Devcon2 in Shanghai, describes a different vision of a scalable blockchain.

The Mauve Paper is available at https://docs.google.com/document/d/1maFT3cpHvwn29gLvt

Y4WcQiI6kRbN_nbCf3JlgR3m_8/edit#.

This proposal is based on a combination of sharding and an implementation of the PoS consensus algorithm. The paper defined certain goals such as gaining efficiency via PoS, minimizing block time, and ensuring economic finality, scalability, cross-shard communication, and censorship resistance. Some of the vision presented in the Mauve Paper has been implemented in Ethereum as “the Merge” upgrade. We discussed the Merge and the future of Ethereum in *Chapter 13, The Merge and Beyond*.

DAG-based chains

DAG stands for **Directed Acyclic Graph**. It is seen as an alternative to linear chain-based blockchain technology. A blockchain is fundamentally a linked list, whereas a DAG is an acyclic graph where links between nodes have only one direction. In other words, DAG-based blockchains do not look like a linear chain, but more like a graph that resembles a tree. A DAG consists of vertices and edges.

Directed means the graph moves only in one direction and acyclic means that there is the possibility of moving back to a node from the current node.

The DAG structure allows for parallel creation and confirmation of transactions and blocks, thus achieving high transaction throughput. The key idea behind DAG-based chains is that, as we know in normal blockchains, there is a linear sequence of blocks one after another and in the event of forks, only one fork survives due to the fork selection rule (e.g. the longest chain) where the rest of the forks are ignored (destroyed). This means that blocks/transactions can be generated in a limited fashion one after another. Now if somehow, we are able to keep those forks without compromising security, then it means that we can produce more blocks and much faster, even almost in parallel. So instead of only one child and parent in a linear sequence of blocks, in DAGs there are blocks with multiple children and multiple parents, thus improving the block/transaction production speed.

There are two types of DAG-based distributed ledgers:

- **Block-less DAGs**: Here, vertices are transactions, and no blocks exist. Examples of this type of chain are IOTA (`https://www.iota.org`) and Obyte (`https://obyte.org/`).
- **Block-based DAGs**: Here, vertices are blocks and blocks can refer to several predecessor blocks. Examples of this type of chain are the FANTOM, Ghost, and Spectre protocols. See more at `https://eprint.iacr.org/2018/104.pdf`.

Faster consensus mechanisms

Traditionally, blockchains are based on PoW or PoS algorithms, which are inherently slow in terms of performance. PoW especially can process only a few transactions per second. If a different consensus mechanism such as Raft or PBFT is used, then the processing speed can increase significantly. We examined these concepts in greater detail in *Chapter 5*, *Consensus Algorithms*.

PoS algorithm-based blockchains are fundamentally faster because PoS algorithms do not require the completion of time- and energy-consuming PoW.

So far, we have discussed layer 0 (multichain) and layer 1 (on-chain) solutions to the scalability issue. In the next section, we'll introduce a more popular approach that aims to solve the scalability problem by using off-chain or layer 2 components.

Layer 2 – off-chain solutions

Layer 2 solutions are based on the idea that instead of modifying the main chain to achieve scalability, the objective should be to offload some of the processing to faster mechanisms outside of the main underlying chain, do the processing there, and then write the result back on the main chain as an integrity guarantee. Several such techniques are described here.

Layer 2

Layer 2 is the name given to technologies that help to scale Ethereum using off-chain techniques. Blockchains such as Bitcoin and Ethereum currently can do up to 7 and 15 transactions per second respectively. This is much slower than traditional centralized methods of payment, such as Visa, which can process 100,000 transactions per second.

Unless blockchain transaction throughput can be significantly improved, it cannot be used for day-to-day business. A lot of research has been done in this regard. There are two ways to address this issue, either by improving the base layer (i.e., layer 1), or offloading some of the work to another layer (an off-chain system).

Note that there are some complex layer 1 chains that are trying to achieve all three properties at the same time with apparently good results, e.g. Solana. However, there are a few hurdles along the way and only time will tell if it succeeds – so far it's looking reasonable. So, if we cannot scale layer 1, then is there any other option? Yes – we offload the work to another off-chain system, called layer 2.

Layer 2 solutions are off-chain tools, mechanisms, and protocols that allow layer 1 to scale without requiring any changes in layer 1. Some changes can be made to layer 1 to make it more layer 2 friendly, however no changes are necessary. In a way, layer 2 complements layer 1 and results in a more scalable blockchain system. The L2 state can be verified by L1 through either validity proofs or fraud proofs. This mechanism is most important as it ensures that L2 validators cannot cheat and include invalid transactions in an L2 block, e.g., mint coins out of thin air or steal your coins. The second use of L1 is as a data availability layer (state validation) for L2 transactions so that, if there is a dispute, users can either independently re-create the L2 state and ensure data validity and continued system operation, or they can trustlessly exit to L1. Key advantages of layer 2 include lower fees, benefitting from L1 security while being able to execute transactions quicker, and the fact that more use cases are possible than on L1. Layer two solutions aim to achieve similar security and decentralization guarantees as those of the layer 1 chain by inheriting these security guarantees from layer 1.

State channels

Also called payment channels, state channels are another proposal for speeding up the transaction on a blockchain network. The basic idea is to use side channels for state updating and processing transactions off the main chain; once the state is finalized, it is written back to the main chain, thus offloading the time-consuming operations from the main blockchain.

State channels work by performing the following three steps:

1. First, a part of the blockchain state is locked under a smart contract, ensuring the agreement and business logic between participants.
2. Now, off-chain transaction processing and interaction are started between the participants that update the state (only between themselves for now). In this step, almost any number of transactions can be performed without requiring the blockchain. This is what makes the process so fast and arguably the best candidate for solving blockchain scalability issues. It could be argued that this is not a real on-blockchain solution such as, for example, sharding, but the end result is a faster, lighter, and more robust network that can prove very useful in micropayment networks, IoT networks, and many other applications.
3. Once the final state is achieved, the state channel is closed, and the final state is written back to the main blockchain. At this stage, the locked part of the blockchain is also unlocked.

This process is shown in the following diagram:

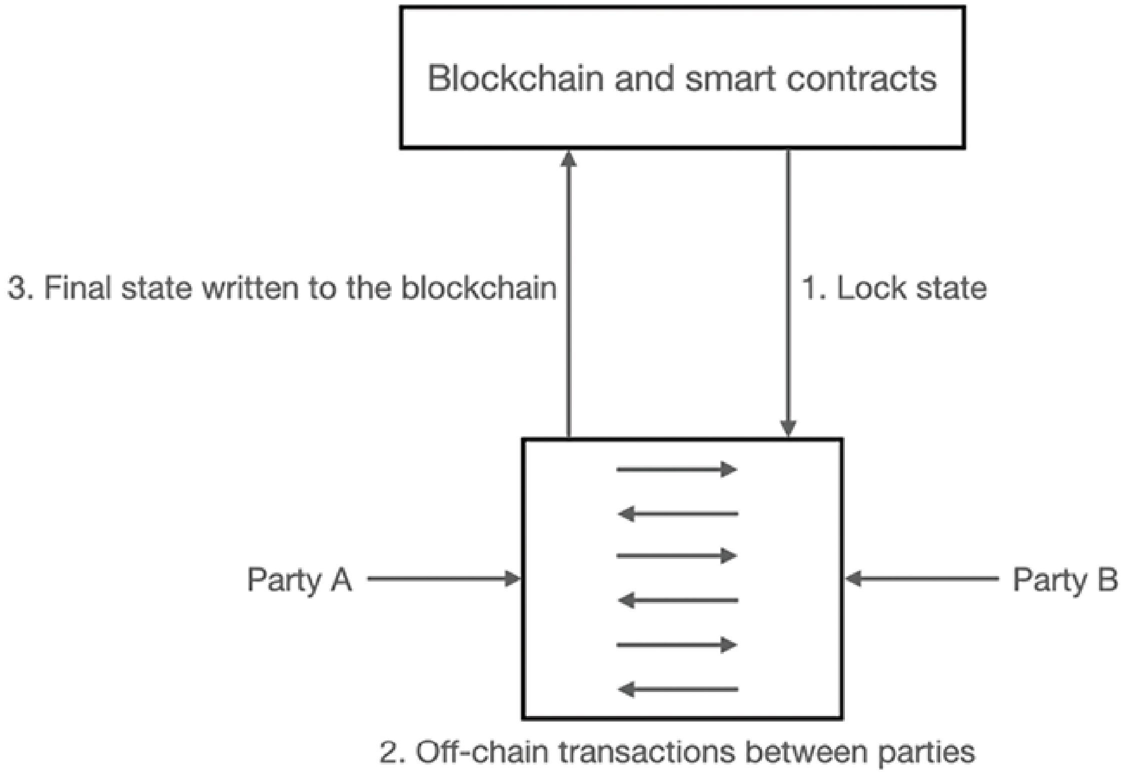
Blockchain and smart contracts
3. Final state written to the blockchain
1. Lock state
Party A
Party B
2. Off-chain transactions between parties

Figure 17.3: State channels

This technique has been used in the **Bitcoin Lightning** network and Ethereum's **Raiden**. The key difference between Lightning and Raiden is that Lightning only works for Bitcoin transactions, whereas Raiden supports all ERC20-compliant tokens, making the Raiden network a more flexible option.

Sidechains

Sidechains can improve scalability indirectly by allowing many sidechains to run along with the main blockchain, while allowing usage of perhaps comparatively less secure and faster sidechains to perform transactions but that are still pegged with the main blockchain. The core idea of sidechains is called a **two-way peg**, which allows the transfer of coins from a parent chain to a sidechain and vice versa.

Sub-chains

This is a relatively new technique recently proposed by Peter R. Rizun and is based on the idea of weak blocks, which are created in layers until a strong block is found.

Rizun's sub-chains research paper is available at https://ledger.pitt.edu/ojs/ledger/article/view/40/55 . Rizun, P. R. (2016). *Subchains: A Technique to Scale Bitcoin and Improve the User Experience*. Ledger, 1, 38-52.

Weak blocks can be defined as those blocks that have not been able to be mined by meeting the standard network difficulty criteria but have done enough work to meet another, weaker, difficulty target. Miners can build **sub-chains** by layering weak blocks on top of each other unless a block is found that meets the standard difficulty target.

At this point, the sub-chain is closed and becomes the strong block. Advantages of this approach include a reduced waiting time for the first verification of a transaction. This technique also results in a reduced chance of orphaning blocks and a speeding up of transaction processing. This is also an indirect way of addressing the scalability issue. Sub-chains do not require any soft or hard fork to implement but do need acceptance by the community.

Tree chains

There are also other proposals to increase Bitcoin scalability, such as **tree chains**, which change the blockchain layout from a linearly sequential model to a tree. This tree is basically a binary tree that descends from the main Bitcoin chain. This approach is similar to sidechain implementation, eliminating the need for major protocol changes or block size increases. It allows improved transaction throughput. In this scheme, the blockchains themselves are fragmented and distributed across the network to achieve scalability.

Moreover, mining is not required to validate the blocks on the tree chains; instead, users can independently verify the block header. However, this idea is not ready for production yet and further research is required in order to make it practical.

The original idea was proposed in the following research paper: https://eprint.iacr.org/2016/545.pdf.

In addition to the aforementioned general techniques, some Bitcoin-specific improvements have also been proposed by Christian Decker in his book *On the Scalability and Security of Bitcoin*. This proposal is based on the idea of speeding up propagation time as the current information propagation mechanism results in blockchain forks. These techniques include minimization of verification, pipelining of block propagation, and connectivity increase. These changes do not require fundamental protocol-level changes; instead, these changes can be implemented independently in the Bitcoin node software.

With regard to verification minimization, it has been noted that the block verification process contributes toward propagation delay. The reason behind this is that a node takes a long time to verify the uniqueness of the block and transactions within the block. It has been suggested that a node can send the inventory message as soon as the initial PoW and block validation checks are completed. This way, propagation can be improved by just performing the first *difficulty check* and not waiting for transaction validation to finish.

Plasma

Another scalability proposal is **Plasma**, which has been proposed by Joseph Poon and Vitalik Buterin. This proposal describes the idea of running smart contracts on the root blockchain (Ethereum mainnet) and having child blockchains that perform high numbers of transactions to feed back small amounts of commitments to the parent chain. In this scheme, blockchains are arranged in a tree hierarchy with mining performed only on the root (main) blockchain, which feeds the proofs of security down to child chains. This is also a layer 2 system since, like state channels, Plasma also operates off-chain.

The research paper on Plasma contracts is available at http://plasma.io.

Plasma vs sidechains

Plasma chains are almost like sidechains, however, they sacrifice some functionality for extra security. Plasma chains can be seen as non-custodial sidechains.

A side chain is an alternate chain to the main parent chain, whereas Plasma is a framework for child chains. Sidechains run as a separate blockchain in parallel to a layer 1 blockchain such as Ethereum. These two chains can communicate with each other so that assets can be moved between the chains. Sidechains also have a consensus mechanism. Plasma chains do have a consensus mechanism used to produce blocks, however, unlike sidechains the Merkle root of each block of the Plasma chain is submitted to Ethereum. Block Merkle roots are used to prove the correctness of the blocks.

Trusted hardware-assisted scalability

This technique is based on the idea that complex and heavy computing can be offloaded to off-chain resources in a verifiably secure manner. Once the computations are complete, the verified results are sent back to the blockchain. An example of such a solution is Truebit: https://truebit.io/.

Commit chains

Commit chains are a more generic term for Vitalik Buterin's Plasma proposal. They are also referred to as non-custodial sidechains but without a new consensus mechanism, as is the case with sidechains. They rely on the main chain consensus mechanism, so they can be considered as safe as the main chain. The operator of the commit chain is responsible for facilitating communication between transacting participants and sending regular updates to the main chain.

> More information on commit chains is available here: Khalil, R., Zamyatin, A., Felley, G., Moreno-Sanchez, P. and Gervais, A., 2018. *Commit-Chains: Secure, Scalable Off-Chain Payments*. Cryptology ePrint Archive, Report 2018/642, 2018. https://eprint.iacr.org/2018/642.pdf.

Rollups

In rollup solutions, transactions are submitted directly to layer 2 instead of layer 1. Submitted transactions are batched and eventually submitted to layer 1. Layer 2s are independent blockchains with nodes that are Ethereum-compatible. All states and executions are processed on layer 2, including signature verification and contract execution. L1 only stores the transaction data, hence the performance boost.

In other words, rollups provide an execution environment outside layer 1, i.e., layer 2, which results in faster execution due to the absence of layer 1 limitations. Once the execution is completed the proof and summary of data is posted to layer 1, where consensus is reached. Therefore, layer 1 is usually called the settlement layer.

This proof is a proof of computational integrity, proving that the state transition is valid after transaction execution. These proofs provide evidence of transaction validity in the batch, that internal application logic of the rollup is correctly followed, and of state transition.

Rollups are now quite commonly used and are the main scalability solution for blockchains.

Rollups can be divided into two categories based on their usage. First, we have application-specific rollups where a resource-hungry (expensive) part of an application (i.e. execution) is bundled up in a rollup. Secondly, we have general rollups, which help to scale EVM-based blockchain networks by their ability to generate validity proofs for any state transition on the EVM.

Layer 2 can improve TPS to tens of thousands of TPS without sacrificing decentralization and security because its security is inherited from the layer 1 chain, e.g., Ethereum.

While achieving scalability, it is also important to ensure the integrity of the chains and relevant data. There are two ways layer 1 can provide security (data integrity) for layer 2, *data validity* and *data availability*.

Data validity

Data validity refers to the requirement that the layer 2 state can be verified by layer 1 by using validity proofs or fraud proofs. This mechanism ensures that layer 2 validators cannot do malicious things such as including invalid transactions in layer 2 blocks, censoring transactions, creating money out of thin air, or stealing funds. Colloquially speaking, data validity ensures that “no one can spend funds that do not belong to them.” A layer 2 blockchain can publish its state periodically to the layer 1 chain by writing the hash of its latest state root. This state root is provided as a cryptographic validity proof using zero knowledge and verified at layer 1 by a smart contract. Another technique is the use of fraud proofs, which are based on the paradigm where honest observers monitor the layer 2 chain and in the case of any suspected incorrect state root submission to layer 1, they can raise an alarm and provide the fraud proof, which will result in automatic chain rollback.

Data availability

Data availability refers to a requirement where in the case of disputes on a transaction, the users must be able to independently recreate the layer 2 state and can do a graceful trustless exit to layer 1. Colloquially speaking, data availability ensures that “anyone can spend their own funds.” Data availability is useful when users need to prove to a layer 1 chain that they own the funds they are trying to withdraw or spend.

For this to happen, layer 1 needs to have access to all transactions on layer 2, or its latest (current) state. One common technique to achieve this is to record layer 2 transactions on layer 1 using *calldata*. Another technique is to store records on a separate external data availability layer where the provider guarantees the data availability via cryptoeconomic or cryptographic mechanisms. Note that posting data to Ethereum using calldata is expensive, however, this is the only practical technique available to post data easily on L1 Ethereum. Any other technique could be too expensive. Another technique is to store data totally off-chain, where another provider becomes the custodian of the data.

A simple way to solve the data availability problem is to download the full data on each entity on the network, i.e., every node keeps a full copy of the data, which is of course not sustainable due to the amounts of data involved and the replication overhead. The other more commonly used and practical option is data availability proofs, which allow clients to check that full data for a block has been published by only downloading a small part of the block.

Computation and data are the two main bottlenecks on the blockchain that are addressed by rollups. Rollups compress data so that only a small amount of data is posted onto layer 1, thus reducing the data footprint. Rollups use several techniques for data compression including signature aggregation (one signature per batch, instead of one signature per transaction).

Computation and execution are addressed by performing executions off-chain and submitting fraud proofs or validity proofs to ensure data validity.

Generally, we can say the key security goals of rollups are data availability, state transition integrity, and censorship resistance. Data availability can be summarized as the question of whether all state updates made to the layer 2 database are publicly available. State transition integrity questions whether all state updates made to the database are valid. Censorship resistance asks the question of whether a user is able to ensure that once the transaction is submitted, it will eventually execute and not be censored by an adversary.

How rollups work

Rollups operate using a smart contract (i.e., rollup contract) that exists on-chain. This smart contract is responsible for verifying and maintaining the state root. State root or batch root is in fact the Merkle root of the entire state of the rollup including account balances, smart contract code, transactions, etc.

A collection of transactions in a compressed format, usually called a batch, can be posted by anyone. The previous and new state root are published with the batch. The on-chain rollup contract checks the previous state root in the batch and matches it with its current state root and if both match, then it shifts the state root to the new state root.

A rollup system handles deposits, transfers, and withdrawals. We can visualize a generic rollup system in *Figure 17.4*:

Figure 17.4: A rollup system

The core idea is as simple as described, but the key problem is to how make sure that the new states are valid and correct. This is where *fraud proofs* and *validity proofs* come in.

If someone observes that a batch had an incorrect post state root (new root), then a fraud proof is submitted as evidence to the chain via the rollup contract to refute the batch. The rollup contract maintains the complete history of state roots and hash of each batch of transactions. The rollup contract verifies the fraud proof, and if valid, reverts the batch in question and all batches that come after it.

Validity proofs are used by ZK-rollups, which are cryptographic proofs included with every batch that prove that the new state root is the correct result of executing the batch of transactions.

A transaction batch can be submitted by many types of users, depending on the security and architecture of the system. The core security of the system revolves around the rule that anyone wishing to submit batches should have a large stake/deposit in the system, which acts as a guarantee against fraudulent (malicious) behaviours. If the user ever submits a fraudulent batch that is proven to be invalid via a fraud proof, then the stake/deposit is burned according to the system rules. Anyone can submit a batch, but such an open approach is not efficient and secure. There are other options too, for example, a sequencer auction can be held at regular intervals to determine the next sequencer for a fixed period of time. The sequencer can also be randomly chosen from a set of stakeholders that are holding a stake in a PoS mechanism. An inadequately performing sequencer can be voted out by the token holders on the network, and a new auction can be held again.

Types of rollups

There are two types of rollups that vary depending on the security model they are based on: optimistic rollups and **zero-knowledge** (**ZK**) rollups.

Optimistic rollups

Under this model it is assumed that transactions are valid by default unless challenged by a fraud proof. The fundamental idea is that the rollup provider posts the transaction data on the chain and waits for a few days. If someone (an external validator) complains and proves via a fraud proof that the posted transaction is invalid, the rollup provider gets “slashed” (loses its stake) and the transactions are rolled back.

Optimistic rollups use a sidechain that runs in parallel to the main Ethereum chain. The key idea is to move computation and storage off layer 1. After a transaction is executed on L2, the optimistic rollup proposes a new state to the L1 or notarizes the transaction. The transaction data from L2 is written to L1 as *calldata*. The rollup system compresses many transactions into a batch to reduce the amount of data posted on the main L1 chain and thus the cost (in fees).

”Calldata” is a data area in Ethereum smart contracts used to pass arguments to a function. Its behavior is similar to memory. It persists on the Ethereum blockchain as history logs but is not part of Ethereum’s state and is therefore an inexpensive method of storing data on-chain.

The system utilizes fraud proofs, meaning that if a verifier (someone on the network) notices a fraudulent transaction, the optimistic rollup network will execute a fraud proof and run the transaction’s computation using the available state data (data validity). In this case, the gas consumed to run a fraud proof is reimbursed to the verifier.

Optimistic rollups are more suitable for general-purpose EVM computations because of their ready compatibility with EVM and low cost.

Transactions are processed off-chain but are posted on-chain to the rollup contract. The funds in an optimistic rollup system are stored in a smart contract on the Ethereum L1 main chain. This smart contract is responsible for handling fund deposits and withdrawals, signing up aggregators, and committing fraud proofs. Anyone can sign up as an aggregator by staking a bond in the smart contract.

This is how the process generally works:

1. A user deploys a smart contract/transaction off-chain to an aggregator (block producer or rollup provider).
2. The new smart contract/transaction is created locally at an aggregator. At this point, the local state of the aggregator has advanced to the new state.
3. The aggregator calculates the new state Merkle root.
4. The aggregator creates a new Ethereum transaction containing the new state root. The new state contains elements such as accounts, balances, and contract code. The operator submits both the old and new state root and if the old state root matches the existing state root, the existing one is discarded, and the new state is applied.
5. If an aggregator deploys an invalid transaction, an observer can challenge this by posting the valid state root and the Merkle state root as a proof (i.e., a fraud proof). If the fraud proof is correct, the aggregator is penalized along with any others that accepted the block or earned rewards due to the fraudulent transaction. This penalization can take the form of stake removal (slashing) and/or aggregator/node removal.
6. If the fraud proof is accepted, the layer 2 chain is rolled back to the last good configuration.

Optimistic rollup systems are composed of on-chain (layer 1) smart contracts and off-chain (layer 2) virtual machines. As the transaction data is posted and available on-chain, anyone can verify the correctness of state transitions and challenge via fraud proofs if they are found to be inaccurate.

As rollup providers are centralized due to aggregators/sequencers being standalone entities, there is some risk of transaction censorship and denial of service by withholding state data. This risk is, however, somewhat mitigated by layer 1's security guarantees and the fact that transaction data (state data) is available on the main chain. This means that if a rogue or otherwise faulty rollup provider goes offline, another node can take over that role and produce the rollup's last state to continue block production. On-chain data availability also enables users to construct Merkle proofs proving their ownership of the funds, which allows them to withdraw their funds from the rollup. Moreover, users have the option of submitting transactions (usually for a higher fee) directly to the L1 rollup contract instead of a sequencer, which forces the sequencer (aggregator) to include the transaction within a time limit to keep producing blocks. As L1 is the final settlement layer where a transaction is only considered final if the rollup block is accepted on the Ethereum main layer, users can be confident that L1 provides sufficient security guarantees for the safety of their funds.

Advantages of optimistic rollups include the fact that anything you can do on L1, you can do with optimistic rollups because optimistic rollups are EVM and Solidity compatible. Optimistic rollups are also secure because all transaction data is stored on the L1 main chain, which means that they are secure and decentralized due to the security properties of the main chain.

Disadvantages include long wait times for on-chain transactions due to potential fraud challenges. It can also be vulnerable to attacks if the value in an optimistic rollup exceeds the amount in an operator's deposited bond.

ZK-rollups

The key idea behind ZK-rollups is that a node on layer 2 processes and compresses transactions and produces a proof of correctness that is submitted to the blockchain. Verification of this proof is much faster than running and verifying all transactions. So, the insight here is that it doesn't matter if the main chain is slow and can handle only seven transactions per second. If each transaction contains thousands of transactions within it, then processing a single transaction on the main chain actually means processing thousands of transactions. This is how scalability is achieved.

The main difference between optimistic rollups and ZK-rollups is that ZK-rollups use zero-knowledge validity proofs whereas optimistic rollups use fraud proofs. The key idea is similar to optimistic rollups in that the rollup provider posts transaction data on the layer 1 chain, but with a zero-knowledge proof that proves the correct execution of the transactions. This zero-knowledge proof guarantees data validity. ZK-rollups compress a large amount of computation and verification into a single operation. ZK-rollups use this technique to bundle many transactions that are executed on the L2, into a single L1 mainnet transaction via a smart contract located on L1. From the data submitted, the smart contract verifies all the transfers that are included in the rollup. The key advantage here is that verifying the transactions doesn't require all of the relevant data, just the zero-knowledge proof. Transactions are written to Ethereum as *calldata* to reduce gas fees.

ZK-rollups consist of two types of actors. The transactors, which create and broadcast the transactions, and the relayers (validators), which collect transactions and create rollups. The relayers create the ZK-ZNARK or ZK-STARK proofs, which are basically a hash that represents the difference between the old and new state. Anyone can become a relayer as long as they meet the rollup-contract-specific requirements of the stake amount.

Zk-rollups are usually better for simple payments (token transfers) and simple exchanges because they do not perform general-purpose computations efficiently. This is because they are not readily compatible with the standard EVM and thus cannot perform general-purpose (Turing-complete) computations efficiently. However, great efforts are being made to achieve this, such as the development of ZK-EVM. Several projects working on building ZK-EVMs are zkSync, Applied ZKP, Scroll, Polygon Hermez, and Polygon Miden.

The key idea behind ZK-rollups is depicted in *Figure 17.5*. Instead of users sending their transactions to the layer 1 main blockchain, they send them to an entity called a rollup provider or rollup coordinator:

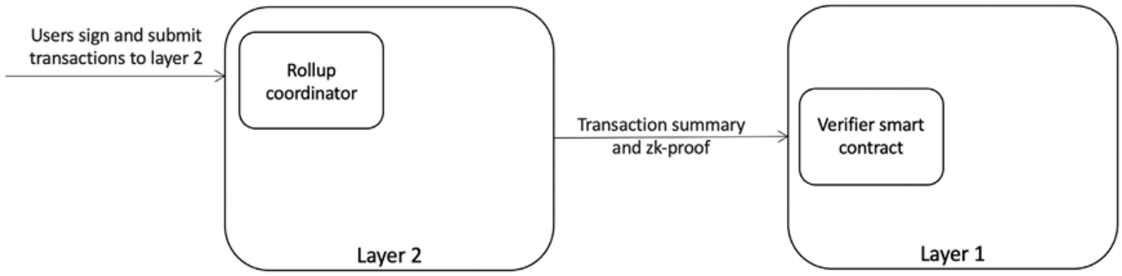
Users sign and submit transactions to layer 2
Rollup coordinator
Layer 2
Transaction summary and zk-proof
Verifier smart contract
Layer 1

Figure 17.5: The idea behind the ZK-rollup-based scalability solution

The rollup coordinator verifies, executes, and compresses many (usually thousands of) transactions into a single proof that is published on-chain and is verified by the verifiers (miners). This proof is much quicker to verify compared to running the execution itself on-chain. The proof is computed using techniques such as SNARKs and STARKs.

Similar to optimistic rollups, the ZK-rollup system is composed of on-chain L1 rollup smart contracts and off-chain virtual machines. The transactions are finalized and settled on Ethereum layer 1 and are only settled if the L1 rollup contract accepts the validity proof submitted by the layer 2 coordinator (prover). As this proof proves the integrity of the transaction and all transactions must be approved by layer 1, this removes the risk of rogue operators trying to submit fraudulent transactions.

Technologies used for building ZK-rollups

There are two technologies that are used to build ZK-rollups, including ZK-SNARKs and ZK-STARKs.

ZK-STARKs are transparent, meaning no trusted setup is required. They are scalable, as proving time is quasi-linear and verifying time is logarithmic. The setup is succinct, requiring at most a logarithmic size. The proof size is somewhat large. It is also post-quantum secure. Key examples of STARK-based layer 2s are StarkNet and Polygon Miden. STARKs generate proofs faster and are post-quantum secure. As the proof is larger in size, they require more calldata space – roughly around 50KB. Thus, they also require more gas to verify and are more expensive for users. Also note that STARKs are based on 80-bit security, meaning a 1/1000 chance of proof forging, which can be a big problem, given that these systems are expected to be used for financial dApps potentially handling millions and billions of dollars. Therefore, this needs to be looked at carefully before choosing a STARK-based rollup solution.

ZK-SNARKs are non-interactive, meaning there is only one message posted by the verifier. They are succinct, meaning logarithmic verifying times. They require a trusted setup, which can take linear time or more to do. The proof size is smaller. They are not quantum secure. Some key examples of ZK-SNARK-based systems are Aztec, Loopring, zkSync, Hermez, and Polygon Zero.

The following table shows a comparison between SNARKs and STARKs:

	Proof size	**Prover time**	**Verification time**
SNARKs	Small	Medium	Small
STARKs	Large	Small	Medium

Let's now see how SNARKs and STARKs are used to create proofs.

Think about a program that always terminates – let's call it *c*. This program *c* takes a public input called *x*, and a private input called *w*. The SNARK/STARK system is composed of a prover, a verifier, the program *c*, and a proof π. The prover knows the program *c*, the public input *x*, and the private input *w*. The verifier only knows the program *c* and public input *x*. The prover produces a short proof that proves that it knows a witness (private input) *w* that makes the program output some expected value. The verifier runs a verification process on the proof, which convinces the verifier that the prover indeed knows a witness (private input) *w* that, when input into the program *c* with public input *x* and executed, outputs the expected value. The key point here from the scalability point of view is that the verifier's running time to verify the proof is a lot less than the execution of the program itself.

In other words, the prover time is linear to the program whereas the verifier time is sublinear:

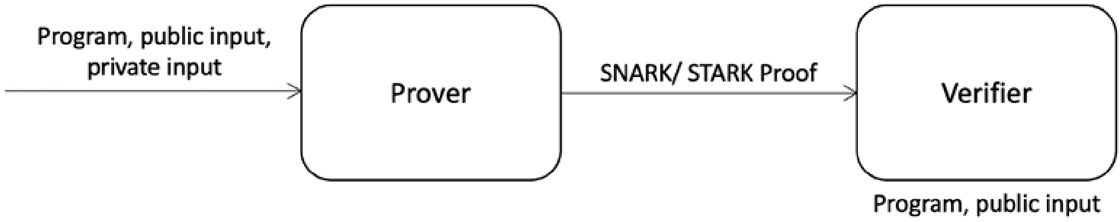

Figure 17.6: SNARK/STARK fundamental idea

What does a rollup provider do? Fundamentally, rollup providers maintain Merkle trees, which are comprised of accounts and consequently represent a state with a Merkle root. This Merkle root is posted onto the layer 1 main blockchain verified by the verifiers. When the rollup provider executes all the transactions submitted to it, it creates a new Merkle root as a new state. At this stage, the old Merkle root is no longer effective. The rollup provider creates a SNARK proof and submits that to the layer 1 blockchain along with the transaction summary. On layer 1, the verifiers (miners) verify the proof.

The new Merkle root is the result of applying a sequence of valid transactions to the old Merkle root. Here we can write program c (recall program c!) that checks for that. Here the public input x is the old Merkle root, new Merkle root, and the transaction summaries. The private input to program c is the account balances (state) of all the accounts that changed due to the transaction execution, the old account balances , and the digital signatures that signed those transactions. Program c will run all the checks, including signature validations, account balance checks, and transaction verification checks and ensure that the executions are correct. Basically, this checks that the new root is valid and consistent with the old root after the transaction execution was completed. Next, the rollup provider posts the updated new Merkle root to the layer 1 blockchain along with the SNARK proof that the new root is consistent with the old root according to the transaction the rollup provider received and executed. The data posted to the blockchain includes the Merkle root, SNARK proof, and a summary of the transaction data excluding transaction signatures. Miners now simply record the Merkle root and verify that the ZK-SNARK proof is valid. Now the verifiers have basically verified the new state root, which was created as a result of executing many (usually thousands of) transactions on layer 2, i.e., the rollup provider. This is how the massive scaling in transaction throughput is achieved.

ZK-rollups' architecture is largely based on two key components: on-chain contracts and off-chain virtual machines. The ZK-rollup protocol is managed by smart contracts running on the layer 1 Ethereum main chain. The ZK-rollup contracts are responsible for rollup block storage, tracking deposits and withdrawals, monitoring state updates, and verification of the zero-knowledge proofs submitted by the block producers from layer 2. The off-chain virtual machine on layer 2 is responsible for transaction execution and state storage. Here, the validity proofs verified on the Ethereum main chain provide a cryptographic guarantee of the correctness of the state transitions on the off-chain layer 2 VM.

ZK-rollups rely on Ethereum's main chain for data availability, transaction finality, and censorship resistance. Note that only the execution has been offloaded to layer 2 – decentralization and security are still based on the Ethereum main chain.

What happens if the rollup provider fails? As rollup providers are trustless entities, users can find new rollup providers. However, the issue is that these new rollup providers don't know anything about the state information (account balances, etc.) held by the previous rollup provider. So, how can the new rollup provider get the state of the old provider? More generally, the problem domain concerns how we transfer assets/state/account data to and from layer 2. If the old rollup provider (also called the coordinator) is no longer available, how do we make the old rollup provider's data available to the new rollup provider? For this purpose, there are two solutions, on-chain data availability and off- chain data availability. An example of an on-chain data availability solution is zkSync and an off-chain data availability solution is zkPorter. We introduce these solutions next.

In zkSync, all transaction summary data is stored on the layer 1 blockchain. When the new rollup provider starts up, it can read all transaction information from layer 1 that the old rollup provider processed before failing and can reconstruct the entire state (world state) from this data. Once the new rollup provider has the latest state, it can start working as the new rollup provider just as the old one was. This is a safe way to make data available again – a safe solution (due to the L1 security guarantees) to the data availability problem, however, the disadvantage here is that rollup providers have to write large amounts of transaction data to the layer 1 chain, resulting in higher transaction fees. As it's a safe but expensive method, it is recommended for high-value crypto digital assets.

In zkPorter, the transaction summary data is not posted to layer 1, but on a separate new blockchain (off-chain). This mechanism is much cheaper than zkSync. This seperate blockchain is there for storing transaction data only and is managed by coordinators under a proof-of-stake mechanism or some other cryptoeconomic incentive mechanism. This mechanism is less safe than zkSync because it doesn't have layer 1 security guarantees, but is cheap because it doesn't require posting and storing data on the layer 1 chain. This means that when we need transaction summary data, it may not be available. Even though the chances of this happening are minimal, the data availability guarantee is still not as strong as that of layer 1. When there is a need to reconstruct state data/world state/account state again, these coordinators on the network provide the transaction summaries, which are then used to reconstruct the state from scratch e.g., on a new coordinator, thus effectively solving the data availability problem.

There is a choice for the user here: they can use either on-chain data availability or off-chain data availability depending on the security and use case requirements.

In summary, ZK-rollups are tools of choice to enable high transaction throughput by executing the computations off-chain and using zero-knowledge proofs to verify these executions on-chain. As verification of transactions is faster than execution, such techniques provide a considerable increase in transaction throughput as measured in transactions per second (TPS). The future of Ethereum is rollup-centric and several upgrades are in the pipeline to remove performance bottlenecks and high costs and make Ethereum more rollup-friendly. We covered this rollup-centric roadmap in *Chapter 13*, *The Merge and Beyond*.

> Technically, ZK-rollups could be (should be) called validity rollups because they do not use ZK for privacy, only for scalability, and the ZK acronym could give the incorrect impression. Rollups that provide privacy should be called ZK-rollups or private rollups. Remember that ZK-rollups use ZK for data integrity rather than for data confidentiality.

Pros of ZK-rollups: No delay, efficient, faster finality, cryptographically secure, and less vulnerable to economic attacks. Due to cryptographic validity proofs, the correctness of off-chain transactions can be guaranteed, in turn guaranteeing security, censorship resistance, and decentralization because it stores the data needed to recover the off-chain state on layer 1. Faster withdrawal time.

Cons of ZK-rollups: ZK-rollups are limited only to simple transfers as they are not compatible with EVM. This is due to the reason that validity proofs are computationally heavy and in order to efficiently process them, ZK-rollups have to build their own programming language. The introduction of a new custom language means ZK-rollups are not Solidity-compatible. However, to alleviate this issue, some work on building Solidity to ZKP language compilers (transpilers) is already in progress. For example Warp, which is a Solidity to Cairo (StarkNet ZKP language) transpiler. ZK-rollups are not efficient enough to run smart contracts. It is better to use ZK-rollups for applications with a simple, high-volume transaction rate such as exchanges. They are suitable for applications with little on-chain activity. Computation and verification of validity proof is costly, which can result in higher fees. EVM compatibility is not easy to achieve. Producing validity proofs can need special high-end hardware, which could result in centralization due to resource requirements, where only a few parties are able to afford such expensive hardware. Centralized operators could possibly censor or reorder transactions to their advantage. SNARKs require a trusted setup, which can raise concerns about security.

The key difference between rollups and sidechains is that sidechains are based on their own assumption of security and run their own validators, whereas rollups inherit their security and trust from layer 1. Sidechains can be seen as another layer 1 chain, thus suffering from the blockchain trilemma, trading off decentralization for scalability. Moreover, sidechains rely on trusted bridges (usually multisig) to connect with the main chain (Ethereum). Sometimes these bridges are also somewhat centralized where a majority of multisig signatures are controlled by members of the same group, e.g., foundational members. Rollups use trustless bridges with better trust assumptions.

ZK-rollups are good for basic transactions but are not general-purpose. They are good enough to do specific basic transactions within the rollup system and post proofs for verification on the main chain, but what about running general-purpose programs like Solidity smart contracts inside the rollup? It is possible, by using Zero-Knowledge **EVMs** (**ZK-EVMs**). This means that the rollup provider can produce proof of correct execution of a Solidity program. This is done the same way that standard Ethereum proof of balance updates are created, meaning that proof of state transitions that occurred due to a Solidity program execution resides on a leaf in the Merkle tree.

Proving off-chain deposits, transfers, and withdrawals is very useful, but it is not general-purpose. We still cannot prove an arbitrary execution like a smart contract. If we can somehow prove an off-chain EVM execution's correctness, then you can imagine how many different use cases it would enable. This is exactly what ZK-EVMs try to do, which we'll explore next.

ZK-EVM

A **ZK-EVM** is a specific type of ZK-rollup. Instead of generating proof of some application-specific logic, as is the case with normal ZK-rollups, ZK-EVMs produce proof of the correctness of an EVM bytecode execution. This means that standard Ethereum layer 1 EVM programs can run on a ZK-EVM and the ZK-EVM will produce the proof of correct execution of

EVM code. A ZK proof of valid state transition on layer 2 is published on layer 1.

We can define a ZK-EVM as a layer 2 VM that executes smart contracts as zero-knowledge proof computations. It aims to replicate the Ethereum system as a ZK-rollup to scale Ethereum without requiring any changes to code or the loss of compatibility with existing Ethereum development tools.

The general high-level architecture of a ZK-EVM is shown in *Figure 17.7*:

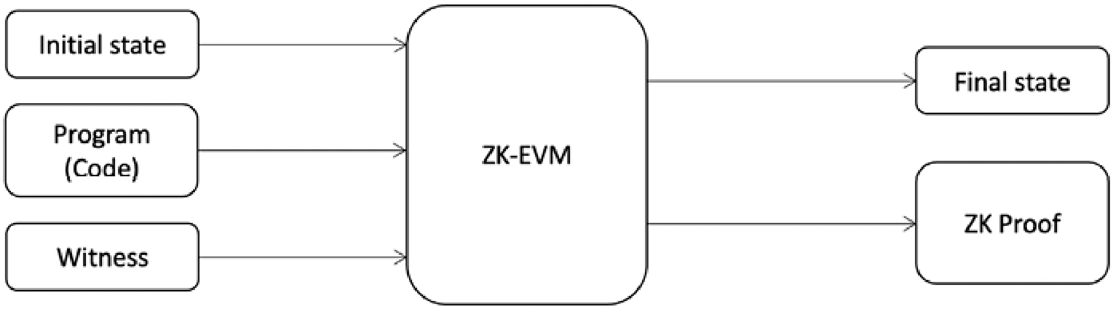

Figure 17.7: ZK-EVM high-level architecture

Remember, the key idea here is to prove EVM bytecode execution. The following are the steps involved:

1. First, layer 2 executes the transactions and as a result state changes occur.
2. The ZK-EVM extracts the previous state, executions, and new state and calculates the difference between the old and new state.
3. The ZK-EVM runs and produces a trace of execution. The trace of execution usually contains the execution steps, block header, transaction data, smart contract bytecode, and Merkle proofs.
4. The prover takes the trace and produces a proof. This is where arithmetic circuits are built, such as the EVM circuit, RAM circuit, and storage circuit. Each circuit produces a proof and is then aggregated into a smaller circuit. Eventually a final proof is produced.
5. This proof is published on the main chain.

This process allows all layer 2 transactions (possibly thousands) to be represented as a single transaction on the main chain.

> The key idea here is that ZK-EVMs scale the transaction throughput by multiplexing (consolidating) layer 2 blocks and transactions into layer 1 as a transaction.

Designing a ZK-EVM is difficult. It requires not only refactoring of the EVM but also refactoring of the Ethereum state transition mechanism to make it zero-knowledge friendly.

There are two ways to do this. The first is **native ZK-EVM** or **bytecode compatible**, which creates circuits for all EVM opcodes. The other option is to build a new customized **Virtual Machine** (**VM**) and a programming language for the new VM t. It’s not easy to build either the native ZK-EVM or a customized virtual machine, however, the native ZK-EVM is more difficult to build. The native ZK-EVM is more developer-friendly due to compatibility with Ethereum tools that already exist and that developers are familiar with, while customized EVMs require learning new programming languages and tools, which is not as much of a developer-friendly option. As native ZK-EVMs inherit the security model of the standard EVM as well as Solidity, a known, comparatively mature language, they are more secure, whereas customized EVMs may not offer the same level of security.

Scroll, ConsenSys, and Applied ZKP are examples of native (bytecode-compatible) EVMs whereas StarkNet and zkSync are two prime examples of customized VMs. In all these options, the key requirement is that proofs must be generated quickly and at minimal cost. The key benefit of ZK-EVMs is that we can reuse the already deployed bytecode on layer 1 without changing any code, which makes already deployed dApps immediately more scalable simply by redeploying them on layer 2. Moreover, we can use existing tools and languages like MetaMask, Truffle, and Solidity for writing smart contracts on layer 2, just as if we were on layer 1.

ZK-EVMs allow developers to write smart contracts using the same tools and languages used on layer 1, e.g., Truffle and Solidity, and also allow reuse of already deployed bytecode. Both features are allowed while still guaranteeing state integrity.

The diagram in *Figure 17.8* shows various approaches that native EVMs and customized EVM projects are taking to build ZK-EVMs:

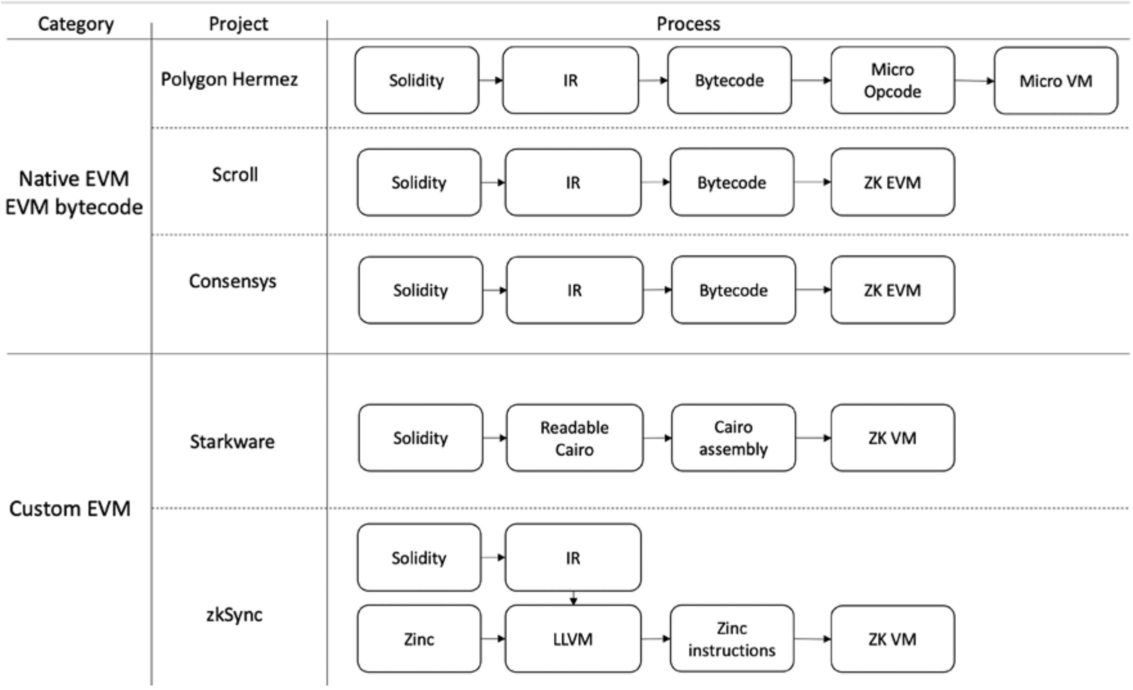

Figure 17.8: Different approaches to building ZK-EVMs

ZK-EVMs can be divided into three broad categories based on the technique they use to interpret the code and produce proofs:

- **Transpiler based** – In this type, an EVM language such as Solidity is trans-compiled (transpiled) into a SNARK-compatible VM. StarkWare and Matter Labs use this approach.
- **Direct interpretation based** – In this approach, EVM bytecode is interpreted directly to produce a proof of correct execution of bytecode. This technique is used by Hermez, Scroll, and ConsenSys ZK-EVMs.
- **Full EVM equivalence** – This approach aims for full EVM equivalence at the Ethereum layer 1 consensus level. This proves the validity of layer 1 Ethereum state roots. There is no concrete example of this variant yet.

There are also various types of ZK-EVMs based on different levels of performance and compatibility with the EVM. These types were proposed by Vitalik Buterin at `https://vitalik.ca/general/2022/08/04/zkevm.html`.

There is a dilemma here in terms of how much a ZK-EVM can be compatible with the EVM. We can call it the ZK-EVM tradeoff dilemma. The dilemma is that the more compatible a ZK-EVM is with the standard EVM, the less efficient it becomes. In other words, better performance (speed) means less compatibility with the standard EVM. This dilemma will become clearer by examining the types of ZK-EVM described as follows:

- **Type 1**: This type allows verification of Ethereum blocks natively. This type is fully equivalent (compatible) to Ethereum without any changes being made to any component of the Ethereum system, including state trees, any relevant data structures, compute logic, consensus logic, precompiles, or any other mechanism. EVM equivalence means that if the same set of transactions were input into both the EVM and ZK-EVM, the resultant states produced by both would be exactly the same, i.e., the same state root, storage, account state, balances, and any other data structures. As such, this type is compatible with all native Ethereum applications. However, it is not efficient, as proof generation takes a long time with this type.
- **Type 2**: This type is EVM-compatible but not Ethereum-compatible, meaning that it is equivalent to the EVM but differs from Ethereum core data structures like block structure, state tries, consensus, and other relevant mechanisms. It is fully compatible with existing applications but gives up Ethereum compatibility somewhat by modifying Ethereum's data structures to make proof generation faster and to make development easier. For example, Ethereum uses **Merkle Patricia Tree** (**MPT**), whereas a ZK-EVM could use sparse Merkle trees, which are more efficient data structures. Another example is that Ethereum uses the Keccak hash function, while the ZK-EVM could use the Poseidon hash function, which is more efficient. Another improvement could be to use only a single trie in the ZK-EVM system, instead of the multiple types of tries used in Ethereum. It is fully compatible with almost all Ethereum applications and can share most infrastructure, however proof generation is still slow.

- **Type 2.5**: This type modifies the EVM only by changing gas costs. It has fast proof generation times but introduces some incompatibilities.
- **Type 3**: This type is most equivalent to the EVM, but some tradeoffs are made to improve proof generation times and make development easier. These tradeoffs include some application incompatibility – for example, if a dApp uses precompiles, it would have to be rewritten because type 3 doesn't support these. Scroll and Polygon are two key examples of type 3 ZK-EVMs. Proof generation is fast in this type and is fully compatible with most Ethereum dApp.
- **Type 4**: This type of ZK-EVM compiles high-level language contracts written in Solidity or other languages into a specialized VM. It is not compatible with some Ethereum dApp and cannot share most of their infrastructure. However, this type has the fastest proof generation times, thus reducing costs and centralization risk.

Some key challenges that ZK-EVM implementations must solve are complexity and performance. ZK-EVM implementations are complex because creating and maintaining circuits (arithmetization) of EVM operations, i.e. opcodes can be challenging. Also, proofs must be generated as fast as possible with minimal cost, making achieving a high level of performance difficult. However, it is doable, and many reasonable examples are available. Proof generation time is usually linear to the program size, whereas verifier time is much lower than the program execution itself.

With ZK-EVMs, it is easy to migrate dApps from layer 1 to layer 2 zk-rollups. It also results in a reduction of transaction fees and increased program execution throughput because programs are now running on layer 2.

There are different ZK-EVM projects including Scroll ZK-EVM (`https://scroll.io/`), ConsenSys ZK-EVM (`https://consensys.net/blog/news/consensys-launches-private-beta-zkevm-testnet-to-scale-ethereum/`), zkSync (`https://zksync.io`), and Polygon ZK-EVM.

ZK-ZK-rollups

This is another type of rollup, fundamentally still a ZK-rollup, but providing privacy along with scalability. A prime example of this type is Aztec, who introduced the term ZK-ZK-rollup, meaning fully private rollups. .

Optimistic rollups vs ZK-rollups

The following table summarizes the differences between optimistic rollups and ZK-rollups:

Property	Optimistic	ZK-rollups	ZK-ZK-rollups
Confidentiality	No	Not inherently; ZK only used for scaling	Yes
Anonymity	No	Not inherently; ZK only used for scaling	Yes
Complexity	Low	High	High
General-purpose execution–EVM compatibility	Yes, easier to do	Hard to do	Hard to do
Withdrawal time	Approx. 1 week as network participants must be given time to verify transactions/batches	Fast, as soon as the next batch is submitted by the rollup provider, usually less than 10 mins	Fast, as soon as the next batch is submitted by the rollup provider, usually less than 10 mins

Performance / efficiency	High	Lower comparatively	Lower comparatively
Verification efficiency	Fraud proof is quick to verify once submitted, however, the system relies on verifiers to monitor the chain to detect fraud and submit fraud proofs, which can be a slow process.	Quick to verify	Quick to verify
Security	Crypto-economic incentives	Cryptographic proof	Cryptographic proof
Finality	Days (single confirmation)	Minutes	Minutes
Usability	Easy to migrate apps from L1 to L2	Not easy to build apps on due to mathematical complexities, but compilers and DSLs are available to create circuits, e.g., Cairo	Not easy to build apps on due to mathematical complexities, but compilers and DSLs are available to create circuits, e.g., Noir
Privacy	No	No	Full

Transaction costs	Lower as no high-end hardware is usually required, and limited data is posted on-chain. However, note that at scale, e.g. more than 100 TPS, optimistic rollups tend to start to cost more and ZK-rollups start to cost less.	Higher due to on-chain proof verification and high-end hardware required to generate proofs	Higher due to on-chain proof verification and high-end hardware required to generate proofs
Trusted setup	No	Yes	Yes

Fraud and validity proof-based classification of rollups

From another angle, depending on whether validity proofs or fraud proofs are used and what mechanism is used for data availability, we can broadly divide L2s into the following four categories:

- **ZK-rollups** – These use validity proofs with data kept on the L1 Ethereum chain and utilize ZK-SNARKs/STARKs.
- **Optimistic rollups** – These use fraud proofs with data posted on the L1 Ethereum chain.
- **Validium (validia)** – Validia (validiums) are scaling solutions that use validity proofs such as ZK-rollups utilizing ZK-SNARKs or ZK-STARKs to guarantee transaction integrity and do not store transaction data on the Ethereum L1 chain. If data is kept off-chain, it can result in data availability issues, but this is seen as a tradeoff to achieve a high level of scalability – usually around 10K transactions per second. Moreover, multiple chains can run in parallel to achieve further increases in throughput and scalability.
- **Plasma** – This is a separate blockchain anchored with the layer 1 Ethereum chain. It executes transactions off-chain using its own block

validation mechanism and transaction data is kept off-chain. It uses fraud proofs to mediate disputes.

> In the future, we expect to see hybrid solutions using a mix of the above techniques. Some are good for certain specific use cases, and some for others.

We can visualize this classification in the following table:

Where is data	SNARK/STARK	Fraud proofs
Data on-chain	ZK-rollup	Optimistic rollup
Data off-chain	Validium	Plasma

> Even though **ZK-ZK-rollups** are basically **ZK-rollups**, they also provide confidentiality, which is why they are categorized separately here.

Remember: the key difference between the security models of rollups and sidechains is that rollups inherit security guarantees from Ethereum by posting state changes and transactions to the main Ethereum chain, whereas sidechains are independent and do not post state changes and transaction data to the Ethereum chain.

There are many solutions that can be categorized as layer 2. A selection is given in the following list, with a brief description of each:

- **Arbitrum** – An optimistic rollup aiming to feel exactly like interacting with Ethereum, but with a fraction of the L1 transaction cost.
- **Aztec** – An open source L2 network offering scalability and privacy for Ethereum. It enables inexpensive and fully private crypto payments using zero-knowledge proofs.
- **Base** – A layer 2 chain introduced by Coinbase based on the Optimism stack (OP stack).

- **Boba Network** – An L2 Ethereum scaling and augmenting solution based on optimistic rollups, forked from Optimism.
- **DeversiFi** – Enables access to DeFi on Ethereum. Users can invest, trade, and send tokens without paying gas fees.
- **Fuel v1** – An optimistic rollup with low transaction costs, high speed, and high throughput.
- **Gluon** – A layer 2 scalable trading engine built on top of Ethereum and offering low fees and high-frequency trading.
- **Immutable X** – A layer 2 solution for NFTs on Ethereum with zero gas fees, instant trades, and scalability for games, applications, and marketplaces.
- **Layer2 Finance** – Enables access to DeFi protocols for everyone, where users can aggregate their DeFi usage and save on Ethereum fees.
- **Loopring** – The same security guarantees as L1 Ethereum, but with scalability and throughput increasing by 1000x and transaction costs reduced to 0.1% of L1.
- **Metis Andromeda** – An EVM-equivalent scaling solution originally forked from Optimism.
- **OMG Network** – A value-transfer network for ETH and ERC20 tokens. Scales through centralized transaction processing and guarantees safety by decentralizing security.
- **Optimism** – Optimistic Ethereum is an EVM-compatible optimistic rollup chain. It aims to be fast, simple, and secure.
- **Polygon** – This is a layer 2 solution consisting of various sidechains that run concurrently with the main Ethereum chain. Several techniques are utilized by these sidechains to provide scalability, including ZK-rollups, Plasma chains, and optimistic rollups. Polygon has a native token called MATIC, which is an ERC-20 token and is used for governance, staking, and fees. Polygon is a suite consisting of the following scalability solutions:
    - **Polygon PoS** – A public PoS network achieving high transaction throughput and low fees by running transactions on sidechains. PoS ensures security through the Plasma bridging framework and decentralization through a PoS validator network.
    - **Polygon Edge** – Polygon Edge is a framework that allows the building of Ethereum-compatible public and private networks.

    - **Polygon zkEVM** – This is an EVM-compatible ZK-rollup.
    - **Polygon Avail** – This is a data availability solution that allows the ordering and storing of transaction data for other blockchains.
    - **Polygon Miden** – This is a STARK-based ZK-rollup.
    - **Polygon Zero** – This is an Ethereum-compatible ZK-rollup based on recursive proofs.
    - **Polygon Nightfall** – This is an optimistic rollup utilizing zero-knowledge proofs.
    - **Polygon Hermez** – This is a ZK-rollup optimized for secure, low-cost, and usable token transfers on Ethereum.
    - **Polygon supernets** – Allow the creation of custom, high-performance blockchains.
- **STARKs, StarkNet, and StarkEx** – StarkNet and StarkEx are two solutions provided by StarkWare to scale Ethereum. StarkNet is a general-purpose, permissionless, and decentralized layer 2 based on STARK-based ZK-rollups. StarkEx is an application-specific scaling engine developed using Cairo and SHARP. StarkNet, StarkEx, and related technologies are developed by StarkWare. These scaling technologies use STARKs to enable blockchain scaling by proving the computational integrity of the computations performed off-chain on layer 2.
- **ZKSpace** – A platform consisting of three key parts: 1) a layer 2 Automated Market Maker Decentralized Exchange – AMM DEX based on ZK-rollups called ZKSwap v3; 2) a payment service called ZKSquare; 3) an NFT marketplace called ZKSea
- **ZKSwap** – It is a fork of zkSync with added AMM functionality.
- **zkSync** – A user-centric ZK-rollup platform providing scaling for Ethereum. It supports payments, token swaps, and NFT minting.
- **zkSync 2.0** – This is a new version of zkSync that combines zk-rollup and zkporter.

In summary, rollups are the most promising and actively developed and used technology for scaling layer 1 blockchains.

Example

In this section, we'll see how Polygon works and how can we deploy a contract to the Polygon layer 2 network.

Polygon PoS

Polygon is a suite of scalability solutions. Polygon PoS is a proof-of-stake commit chain anchored to Ethereum layer 1 through checkpoints.

Polygon PoS is composed of two core components, a Heimdall chain and Bor chain. The Heimdall chain works in combination with the stake manager contract deployed on Ethereum layer 1. The stake manager contract is responsible for validator and stake management. By having a stake manager contract on Ethereum, the security of the mechanism improves, because now there is no need to trust the validator running on the Heimdall chain, and you can rather rely on the security of layer 1. The Heimdall chain creates checkpoints on the Ethereum mainchain periodically. This checkpointing ensures finality on the Ethereum main chain.

The Bor chain component is responsible for block production by collecting transactions into each block. Block production is carried out by a subset of PoS validators from the Heimdall network. A subset of validators is selected for producing blocks for a set period of time or number of blocks, called a span. After the span elapses, the validator selection process starts again. Validators are selected based on the amount of MATIC tokens they have staked. The greater the stake, the greater the number of slots the validator is allocated to. However, the shuffling of the validator set coupled with a Tendermint-based proposer selection mechanism allows the algorithm to select validators in such a way that everyone gets a chance in a span as block producer, even if their stake amount is lower than others.

Once the blocks are produced by the Bor layer, the Heimdall layer aggregates them into a single Merkle root and posts it onto the Ethereum layer 1 chain at regular intervals. The proposers for checkpoints are selected via Tendermint in a weighted round-robin manner.

Polygon is thus secured by its own proof-of-stake mechanism and with its own native token called MATIC.

A high-level architecture of Polygon PoS is shown in *Figure 17.9*:

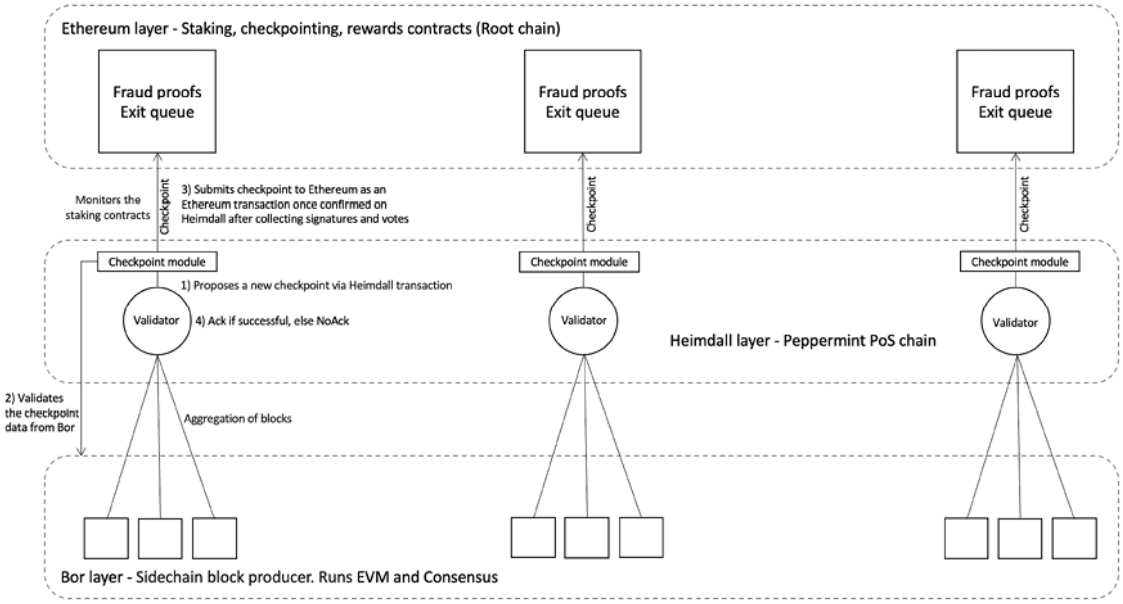
Ethereum layer - Staking, checkpointing, rewards contracts (Root chain)
Fraud proofs
Exit queue
Fraud proofs
Exit queue
Fraud proofs
Exit queue
Monitors the staking contracts
Checkpoint
3) Submits checkpoint to Ethereum as an Ethereum transaction once confirmed on Heimdall after collecting signatures and votes
Checkpoint
Checkpoint
Checkpoint module
Checkpoint module
Checkpoint module
1) Proposes a new checkpoint via Heimdall transaction
Validator
4) Ack if successful, else NoAck
Validator
Heimdall layer - Peppermint PoS chain
Validator
2) Validates the checkpoint data from Bor
Aggregation of blocks
Bor layer - Sidechain block producer. Runs EVM and Consensus

Figure 17.9: High-level Polygon architecture

The preceding diagram shows the Ethereum main chain layer 1 at the top with PoS nodes in the middle and the Polygon sidechain at the lowest layer.

As an example, we can implement on Polygon the ERC-20 token called MET that we built in *Chapter 15*, *Tokenization*. As the Ethereum standard toolchain works with Polygon, all we need to do is simply connect MetaMask to the Polygon test network, and then deploy it through the Remix IDE. I won't go through these instructions step by step, as it is quite straightforward and we've seen already how to set up, configure, and use MetaMask in earlier chapters. All we need to do is to add the Polygon network details under the custom RPC option in MetaMask and deploy the contract as you would normally on Ethereum. The details of the Polygon test network are shown below:

```
Network name: Mumbai
Network URL: https://matic-mumbai.chainstacklabs.com
Chain ID: 80001
Currency symbol: MATIC
```

As an alternative, you can browse to https://chainlist.org/, which hosts a list of EVM networks and enables users to add networks easily to their MetaMask browser, without requiring these details to be added manually via a custom RPC in MetaMask.

You will also need some test tokens to be able to deploy the smart contracts. In order to get test MATIC tokens, visit `https://faucet.polygon.technology/` and enter your MetaMask wallet account address.

Once MetaMask is configured and connected and you have sufficient funds from Faucet, open the Remix IDE from `https://remix-project.org/` and choose **Injected provider – MetaMask** as the environment under the **DEPLOY & RUN TRANSACTIONS** item. Once Remix is connected to MetaMask, make sure the account is set up correctly. Once verified, simply compile and deploy the same way you did in *Chapter 15*, *Tokenization*, using the Remix IDE. As MetaMask is now connected to Polygon, deployment will be made on the Polygon Mumbai test net. Once deployed, you can explore further by using the block explorer at `https://mumbai.polygonscan.com`. With this example, we complete our introduction to layer 2 technologies.

A question arises here – with all these scalability solutions of various types, what will the future of blockchain scalability look like? Is it going to be rollup-centric, or multichain-based, or is layer 1 scaling the future? Only time will tell, but for now rollups are ahead in the race! At the same time, remember that there is no single best solution that is suitable for all use cases. It is likely that the future will contain all these scalability solutions working together to form a high-performance and efficient global ecosystem of blockchains.

Layer 3 and beyond

Multilayer solutions are expected to exist in future where there is a public layer 1 chain at the core of the ecosystem and a public layer 2 providing scalability. There is the possibility of another layer on top, the so-called layer 3, where multiple application-specific chains can run that inherit the security of the “core chain” (i.e., layer 1) through layer 2 – for example, one chain that provides privacy, another chain that’s another rollup, another chain that’s used for payments, and so forth.

Beyond that, we can imagine another layer, also an application-specific layer or another rollup layer, and this ecosystem can grow layer by layer. Layer 3 is conceived as an additional off-chain execution layer on top of existing layer 2 networks. There can even be an additional layer 4 and further layers to provide even more scalability, dubbed "hyperscalability."

Such a configuration could enable an interoperable ecosystem of chains and layers that could give rise to a heterogenous multichain layered network of networks, which would create an interoperable inclusive environment of services and economy. Therefore, layer 3 can also be imagined as an interoperable layer between different blockchains.

Ideas such as fractal scaling or hyperscaling have been proposed by StarkWare and others with different variations, but the general consensus is that it is possible to scale beyond layer 2 and the future of the blockchain ecosystem will largely be based on this vision.

We can visualize this concept as in *Figure 17.10*:

Figure 17.10: A vision of a scalability ecosystem

With this, we have completed our discussion of some of the many existing and potential scalability improvements for blockchain. In the next chapter, we'll discuss privacy and some techniques to achieve privacy in blockchain.

Summary

In this chapter, we introduced a major challenge that blockchain faces: limited scalability, along with several methods available to achieve greater scalability. Fundamentally, there exist layer 0, layer 1, and layer 2 solutions. Layer 2 can be divided into Plasma, state channels, rollups, and validiums. Layer 1 solutions are on-chain solutions such as DAG-based chains and blockchains based on faster consensus mechanisms such as Solana. Layer 0 solutions are multichain solutions such as Polkadot.

Note that while layer 2 solutions are quite promising and a lot of research and development is going on to make these even better, there are several security issues and relevant challenges that need to be addressed to ensure that these protocols are safe. We'll cover some key security challenges of layer 2 protocols in *Chapter 19, Blockchain Security*. In the next chapter, we will explore another very interesting topic – blockchain privacy.

Join us on Discord!

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

`https://packt.link/ips2H`