CONCORDIUM

Concordium White Paper

An introduction to the features of the Concordium Platform

Concordium is a privacy-focused, public and permissionless blockchain architecture. This paper describes the Concordium Platform along with a range of novel features that allow individuals, businesses and public institutions to use permissionless blockchain technology in a way that is private, trusted, scalable and compliant with regulations.

The Concordium Platform is designed to be fast, secure and cost-effective. Concordium's innovative identity layer provides on-chain identity, compliance-centric transactions and enhanced privacy for users, while also allowing for the de-anonymization of network participants. Concordium's two-layer consensus protocol consists of a Nakamoto-style consensus blockchain and a finality layer for fast confirmation of transactions. Our sharding design enables high transaction throughput and private shards for business use cases and sensitive data. Concordium also enables interoperability and communication between shards and between Concordium and other blockchains. Concordium has a standards-based smart contract core with multi-language support. The Concordium Platform also features a transparent incentive structure with cost-effective transactions and predictable fees.

An Introduction to Concordium
Version 1.4, June 2021

Concordium: An Overview

Regulatory compliance by design

Concordium is designed to integrate with current financial and business systems that require knowledge of a user's identity. Through the development of unique protocol-level identity primitives, Concordium helps application developers, individuals and businesses build products that comply with local regulations, while retaining the benefits of a privacy-focused, public and permissionless blockchain.

By **user** we mean any entity that holds an account on the Concordium Platform. These can be individuals or legal entities, such as businesses, and they require a valid form of identification to facilitate the off-chain identification process.

Both privacy and verification of the identity of users

Concordium's innovative identity layer provides a compliance-centric balance between anonymity and accountability. A user's identity is anonymous on-chain, however this anonymity can be revoked and their real-world identity revealed in response to a valid request from a government authority via established legal channels. From the user's perspective, anonymity with respect to the general public is maintained and Concordium's identity layer can accommodate identity providers and anonymity revokers based in different jurisdictions around the world. As such, the Concordium Platform offers a global, multi-jurisdictional solution to the adoption of blockchain technologies across regulatory regimes.

Fast transactions at scale

The Concordium Platform is designed to be fast enough, in terms of transactions per second and the time it takes to finalize a transaction, to meet the needs of business applications on a global scale. This is a major development compared to previous generations of blockchain technology.

Provable and fast finality

Concordium has developed the first provably secure and fast finality layer to run on top of a Nakamoto-Style (NSC) blockchain. This means that a transaction on the Concordium Platform is confirmed and immutable within a short period of time. This is a major advantage over other NSC blockchains, where the finality of a block is only assumed after a large number of subsequent blocks have been produced.

Reliable uptime

Concordium is designed for demanding business use cases with strict uptime requirements. Our two-layer consensus design ensures that the platform remains available and secure in the most adverse conditions. Blocks are correctly added to the longest chain when less than 50% of all stake is controlled by malicious parties, and we achieve significant speedups and efficiencies under normal conditions, when less than 33% of all stake is controlled by malicious parties. Furthermore, the safety of our finality layer holds even under catastrophic failures to the network that causes messages to be delayed much longer than under normal conditions.

High throughput for global scale

Many real-world business applications of blockchain technology have high throughput requirements. To meet these needs, Concordium is developing a novel sharding mechanism that operates in tandem with our finality layer. Concordium can run and coordinate several sub-blockchains (shards) in parallel, each much faster than standalone blockchains. Transfers and communication across shards are also supported by a novel design for intershard signalling.

Build private networks on top of Concordium

Concordium understands that some mission-critical applications require dedicated resources and data security. Our platform provides a cost-effective and easy mechanism for businesses, countries or individuals to create their own blockchains using private shards within the Concordium Platform. In the case of a private shard, the Concordium Platform acts as a notary service that confirms transactions without inspecting content or data.

Future-proof via sophisticated interoperability

Interoperability allows transferring data between different blockchains. This is a requirement for wide adoption of blockchain technology, where businesses are not locked into a specific platform. Concordium has developed a novel design for interoperability that allows our platform to send short authenticated messages to other chains and entities without the recipient being required to operate the Concordium Platform.

A standards-based smart contract core with multi-language support

Concordium’s core on-chain language is WebAssembly (Wasm), a portable, well-defined assembly-like language. Wasm is an internet standard which is gaining a lot of traction in recent years and is already supported in the major web browsers. Many programming languages can already be compiled to Wasm, which potentially allows us to support a large range of smart contract languages. Concordium uses Rust as our first high-level smart contract language, a safe language that also allows for low-level resource control.

A single native token that is easy to model

Global Transaction Unit (GTU) is the native token of the Concordium Platform and the medium of incentivization that ensures network participants are rewarded for their efforts. GTU can be used for a variety of purposes, including as payment for the execution of smart contracts, payment via transactions between users, and as a store of value.

Cost-effective transactions

To accommodate businesses that operate at scale, transactions on the Concordium Platform are designed to be cost-effective as well as fast. Low costs are a function of our proof-of-stake, finalization and sharding design along with incentive mechanisms that prevent excessive charging.

Know transaction costs ahead of time

Transaction costs need to be understood ahead of time in order to build sustainable business models. Concordium uses an innovative price stability mechanism to ensure that transaction costs are fixed in real-world fiat terms despite potential volatility in the price of GTU on the open market.

Transparent token economics and incentives

The economy and incentive structure within the Concordium Platform is transparent and easy to understand. All parameters that control the distribution of rewards, the rate of inflation and the pricing of transactions will be publicly available. The Concordium Foundation will actively monitor and manage the health of the Concordium economy.

Responsible governance on the road to decentralization

While the Concordium Foundation will act as a guarantor that the central principles of the Concordium Platform are adhered to, including privacy with accountability, key functions will be delegated to the Governance Committee. Over time the Governance Committee will evolve to have more responsibilities and governance will be decentralized across network participants.

Open Source

The Concordium Platform has been developed in a closed environment, but before the launch of Mainnet the codebase for all central components of the Concordium Platform has been made open-source at https://github.com/Concordium.

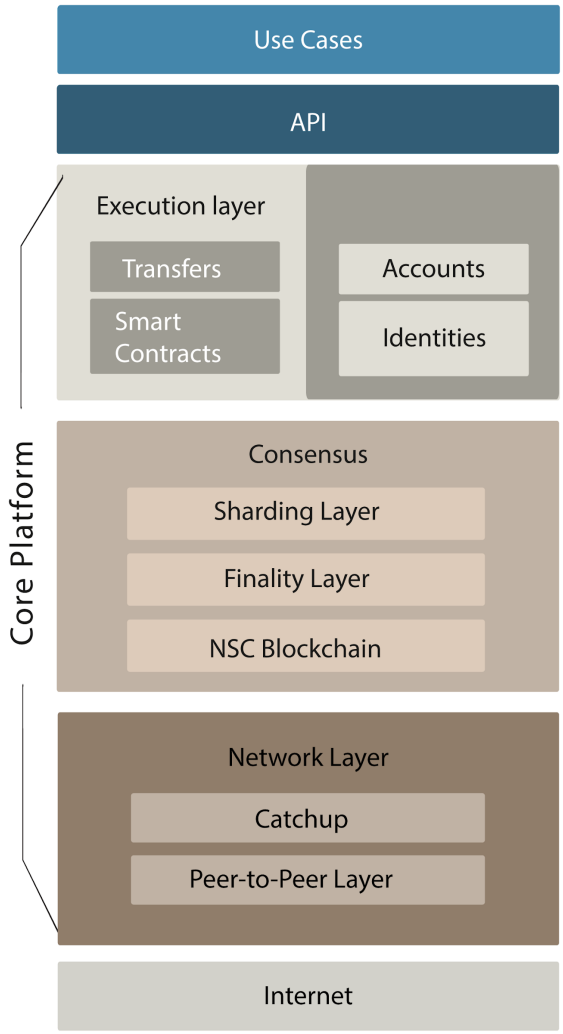
Design Overview

The Concordium protocols can be divided into the following layers:

The **network layer** consists of a **peer-to-peer layer** and a **catchup layer.** The peer-to-peer layer consists of a public, permissionless, high-speed protocol for broadcasting messages to all available nodes. The catchup layer sits between the peer-to-peer layer and the consensus layer and ensures that nodes receive all relevant messages.

The **consensus layer** ensures agreement on all transactions and their order in the ledger. The consensus layer has at its lowest level a **proof-of-stake Nakamoto-style consensus blockchain** that uses the longest chain rule. On top of the NSC blockchain is a fast BFT **finality layer**. On top of the **finality layer** is a **sharding layer**, which adds throughput.

Accounts and identities define how users' identities are processed and used during the creation of new accounts. It specifies how identity providers and anonymity revokers interact with users to create identity objects and revoke the anonymity of users if validly ordered by a qualified authority.



The **execution layer** allows users to interact with the platform. Using the **API**, users can submit transactions, including, for example, transfers between user accounts and deployment and execution of smart contracts.

Figure 1: Overview of the Concordium stack.

Network Layer

The Concordium Platform is a distributed system. It consists of multiple nodes which maintain the blockchain by baking and finalizing blocks. The network layer establishes and manages the communication between nodes.

The network layer provides an abstract interface for communication such that protocols sitting on top of it do not have to know details of the actual communication protocol. For

example, the interface for the consensus layer is a broadcast channel that allows parties to send messages to all other parties.

The network layer consists of the peer-to-peer layer that manages communication and the catchup layer that ensures all nodes, including those having been temporarily offline, receive all necessary messages.

Peer-to-Peer Layer

The peer-to-peer layer establishes a node's connection with peers and manages bi-directional communication between them. Its role can be divided into 3 parts:

1. To set up and maintain a node's presence on the network. This includes finding and managing peer nodes and collecting information about the network to ensure optimal peer-lists.
2. To manage one-to-one communication with each peer.
3. To broadcast messages, including transactions, blocks and finalization messages, to all nodes in the network.

When a node starts up, the peer-to-peer layer is the first module that boots up. The node resolves a DNS record, decodes the list of bootstrap servers, connects and receives a list of peers. Initially, the bootstrap servers will be managed by the Concordium Foundation.

All communication on the peer-to-peer layer is done through network messages. For efficiency, network messages are serialized using FlatBuffer which enables fast implementation in various programming languages.

The peer-to-peer layer protocol is heavily secured against wiretapping by making use of the Verifiable Noise Protocol [Per16] and using formally verified encryption implementations. This enables a very robust guarantee against wiretapping and replay attacks on the peer-to-peer layer.

Any node can initiate a broadcast of a message, *m*, by sending *m* to its peers. When a peer receives *m* for the first time, it forwards *m* to all its peers. This ensures that the message propagates through the whole network. To avoid resending the same message multiple times, nodes buffer hashes of previously broadcast messages. A received message is then only forwarded to all peers if it is not already in the buffer.

Catchup

The peer-to-peer layer allows sending messages to all nodes that are online when the message is sent. The catchup layer ensures that nodes that were offline when the message was sent, or that missed the message for other reasons, also receive the message.

There are two types of messages that are handled differently by the catchup layer. The first type is messages where nodes can determine by themselves that they missed the message. This includes blocks sent in the consensus layer. For example, when a node receives a block with a parent-pointer to an unknown block, this shows that the message containing the parent block is missing. For such messages, the catchup layer uses a pull-mechanism, i.e. nodes pull messages from their peers by explicitly requesting them.

The second type of messages is those that cannot easily be recognized as missing. For instance, this is the case for messages sent between finalizers in our finality layer. For these messages, the catchup layer uses a push mechanism in which the sender periodically resends these messages as long as they are relevant. In this case, finalization messages for a particular block are relevant and replayed until the finalization for this block is complete.

Consensus Layer

A major innovation of our design is the two-layer consensus approach, which combines a Nakamoto-style consensus blockchain with a novel finalization method, providing fast finality.

Proof of Stake and Delegation

The Concordium Platform uses a proof of stake (**PoS**) mechanism to ensure resource-efficient operation of the network along with enhanced security among participants. Users that hold GTU in their account can either stake some of their GTU and run their own node or delegate their GTU to a so-called delegation pool. Across the network, the more stake that participates, the harder it becomes for malicious parties to control a majority of the stake. The effective stake of a node equals the stake of that node plus the node's part of the delegation pool.

Concordium's Two-Layer Consensus Mechanism

Nakamoto-Style consensus

We define Nakamoto-style consensus (**NSC**) blockchains as systems where parties participate in a form of a lottery to win the right to append blocks to the chain. The probability that a certain party wins the lottery depends on how much of the required resource they have, for example computing power for proof of work systems or stake for PoS systems.

In NSC blockchains it is possible that, due to network delays, one party adds a new block without having received information about the previous block. Furthermore, malicious parties can purposefully ignore existing blocks. Both of these situations result in a fork in the chain, turning the blockchain into a tree. One way to deal with this issue is for parties to have a chain-selection rule, such as the longest-chain rule, which determines which chain in the tree parties consider as their current chain and where new blocks should be added. The chain selected by any given party can change over time, causing rollbacks and invalidating transactions on the previously selected chain. Since very long rollbacks are unlikely, blocks can be considered 'final' when there are 'sufficiently many' blocks after it in the chain.

NSC blockchains are secure when corruption is below ½. In PoS systems this threshold refers to the fraction of the stake controlled by malicious parties.

Committee-based Byzantine fault-tolerant consensus

Aiming to provide an alternative, there are committee-based Byzantine fault-tolerant (**CBFT**) consensus designs such as those employed by Tendermint [Kwo14] and Algorand [CM19]. They provide immediate finality in that every block included in the chain can be considered final. This mechanism offers better consistency when compared with NSC blockchains, but this comes at a cost. Namely, CBFT consensus designs only tolerate corruption below ⅓ of the stake.

Our two-layer approach

Concordium's finality layer can be added on top of NSC blockchains [DMMNT20]. Our finality layer allows us to dynamically 'checkpoint' the blockchain by using Byzantine agreement to identify and then mark common blocks in the chains of honest users as final. This two-layer approach provides the best of both worlds. In particular, we achieve the following:

- As long as corruption is below ⅓, our finality layer declares blocks as final faster than the finality rule in a pure NSC blockchain, which is required to wait for 'sufficiently many' blocks.
- When corruption is between ⅓ and ½, we can still rely on our NSC blockchain and obtain the same guarantees as a pure NSC blockchain. Note that pure CBFT designs fail completely under these conditions.

Concordium's Nakamoto-Style Proof-of-Stake Blockchain

Blockchain protocol

While Concordium is currently undertaking research to improve the efficiency and security of NSC blockchains [KMM+20], our proof-of-stake mechanism is a simplified variation of Ouroboros Praos [DGKR18]. The simplification is made possible due to our innovative finality layer, which prevents long-range attacks.

The party that participates in the production of blocks is called a **baker**. Time is divided into equally sized periods called **epochs** each consisting of the same number of **slots** (currently an epoch is set to one hour and a slot is 250 millsec.). In every slot, each baker checks locally whether they won the lottery using. To this end, every baker has a secret key for a verifiable random function (**VRF**). This allows the baker to evaluate the function on given inputs such that other parties (without knowing the secret key) can verify that a certain output was computed correctly without being able to predict outputs themselves.

To check whether a given baker has won in a given slot, it takes the values *slot* and a *nonce* as inputs to the VRF and computes a random value r. The *nonce* is a random value that is updated for each epoch to prevent parties from predicting too far into the future when they will win. The party wins if the value r is below a threshold T, which depends on the party's relative stake α and a common difficulty parameter *f*. The winning probability is

$$\text{Prob(Baker with relative stake } \alpha \text{ wins)} = 1 - (1 - f)^{\alpha}$$

The winning probability is roughly proportional to α, and higher difficulty parameters increase the winning probability for all parties.

A winning party then extends the current longest chain by a fresh block. Since parties compute their VRFs independently, there can be zero or multiple winners for a given slot. The slot time and the difficulty parameter f need to be set to ensure that blocks are produced with the required frequency (i.e., with the expected blocktime) and to reduce the likelihood of having multiple winners.

The lottery uses a fixed relative stake for each baker per epoch. Under normal network conditions, the stake of a baker for epoch *i* is determined by the first block of epoch *i*-1.

Obtained guarantees

Running our NSC blockchain protocol described above, participating bakers grow a tree with the genesis block at its root. The consistency attribute we obtain is called the common-prefix property. It says that if the majority of the stake is controlled by honest parties who follow the protocol, the longest chains (in the view of all honest parties) have a long common-prefix. That is, all honest parties agree on all blocks except for the freshest blocks. We also obtain the properties of chain quality, such that a sufficient fraction of blocks are generated by honest bakers, and chain growth, such that the longest chain grows at a sufficiently high rate. Thus the protocol is both safe and live. These properties of the Concordium protocol have been formally verified in [TS20].

Concordium's Finality Layer

Overview and guarantees

As the tree of blocks described above grows, a finalization committee is responsible for periodically marking blocks as final. Bakers only extend the chain past the last finalized block and finalized blocks are never rolled back.

Finalization committees

Finalization is run by a subset of the bakers in the **finalization committee** whose members we call **finalizers**. For finalization to work properly, we need honest finalizers to hold more than ⅔ of the total stake, including delegated stake. It is therefore important to select the finalization committee such that sufficient honest stake participates in finalization. On the other hand, committees that are too large will slow down finalization considerably. We balance these two conditions by allowing all bakers that hold a minimum required fraction of stake to be included in the finalization committee and act as finalizers. Currently finalizers are required to hold at least 0.1% of the total stake, which ensures that there will be at most 1,000 finalizers in the committee and that all nodes with substantial stake can participate.

Sketch of our finalization protocol

Recall that our NSC blockchain produces a growing tree of blocks. The finalization committee is repeatedly finalizing blocks in this tree. In each iteration, finalizers run a protocol to agree on a unique block at a given depth d (i.e. with distance d from the genesis block). Finalization for a given d proceeds as follows. Each finalizer waits until its chain has reached depth $d + \Delta$, where Δ is a parameter that is set to 0 at genesis. The finalizer then votes on the block it sees at depth d on its chain using the Concordium CBFT consensus protocol. This protocol is designed such that it succeeds if all finalizers vote for the same

block, otherwise it might fail. If the consensus protocol is successful, the block it outputs is defined to be final. If the consensus protocol fails, the finalizers iteratively retry until it succeeds. In every retry, the value of Δ is doubled (or set to 1 if it was 0) and the finalizers wait until they are at depth $d + \Delta$. They then vote for the block they see at depth d on their chain. Eventually Δ will be large enough that the block at depth d is in the common-prefix of all finalizers. In that case, all finalizers vote for the same block, and the consensus protocol will succeed. Increasing the offset Δ exponentially can be seen as a binary search for the common prefix. Note that honest parties typically only deviate for a few blocks, so the algorithm will usually succeed with small Δ. After successful finalization, the finalizers produce signatures on the finalized block, which count as a finalization proof. The finalization proof will be part of a subsequent block. Finalization is then repeated for a larger depth d, and the value Δ is divided by 2 (or decreased from 1 to 0). Decreasing Δ ensures that a good value is found over time.

Our finality layer works if there is *some* common-prefix; it does not need to know how long it is. The rationale behind this is two-fold. It gives responsive finality, which means whenever blocks are included in the common prefix of all finalizers and subsequently agreed upon quickly, we also finalize quickly. Furthermore, not relying on a fixed bound for the common-prefix length makes the finality layer work as a hedge against catastrophic events that cause long forks (e.g. partitions of the internet).

Obtained Guarantees

As proven by Dinsdale-Young et al. [DMMNT20], we obtain the following guarantees from our finality layer:

- **Chain-Forming:** finalized blocks form a chain;
- **Agreement:** all parties agree on the finalized blocks;
- **Updated:** the last finalized block does not fall too far behind the last block in the underlying blockchain;
- **⅓-Support:** all finalized blocks are ‘supported’ by honest parties holding at least ⅓ of the total stake. This means that honest parties had these blocks on their chains before finalization and they are not required to adopt a new chain after finalization, limiting potential rollbacks.

To further ensure the reliability of our platform we are investigating formal verification methods to formally prove the security of our finality layer [DSTT19].

Sharding

The main goal of sharding is to overcome scalability issues. Without sharding, every node in the network has to process all transactions and execute all smart contracts. The basic idea of sharding is to parallelize execution by dividing the network into smaller components or shards. Nodes are then assigned to different shards with separate account balances. Each shard essentially corresponds to a separate blockchain that can be running almost independently of the other shards. This means that transactions on one shard are only processed by the nodes on that shard and, consequently, more transactions can be processed overall.

Overall sharding architecture

Concordium's blockchain has a two-level sharded design with a very robust control chain and light-weight shards. Our sharding mechanism has been described by David et al. [DMMNT21].

The control chain manages shards, provides a finalisation service to the shards and gives a vehicle for cross-shard transactions. Each shard runs an individual blockchain and uses the control chain to coordinate the individual shards.

A shard defines an ordering of all transactions within the shard. As the control chain provides ordering/synchronisation across all shards, the entirety of shards can be considered a single totally ordered blockchain.

As shards allow for efficient reading of only the parts of this global ledger needed for a given application, this architecture makes it possible to create shards for specific purposes (including private shards as described below).

Obtaining security and efficiency

For optimal efficiency, there should be many shards. Initially, Concordium runs an optimistic consensus algoríthm on the shards but shards may use different algorithms, and Concordium will support more consensus algorithms to meet the specific requirements on a shard. In order to optimise the use of the resources on the blockchain, a shard is run by a small committee. However, the risk associated with sampling only a small number of nodes per shard is that a large fraction of them are corrupted. Normally, for a consensus protocol (as run by the shard committee) to be secure, only a small fraction of participants can be corrupted.

The important insight allowing us to circumvent this issue is that security consists of two parts: safety and liveness. Safety means the system does not make mistakes. Liveness means the system does not come to a halt. These two properties can be balanced such that the level of corruption one can tolerate is different for both. This means one can set the parameters of the (shard consensus) protocols such that safety holds, i.e., that the system makes no mistakes, even with high corruption levels, whereas the system may come to a halt if only a few nodes are corrupted.

We will utilize this insight as follows. The control chain consists of many nodes with broad decentralization. The shards run with few nodes per shard, tolerating relatively high corruption for safety. This gives us scalability and safety but with limited liveness in the shard. To improve the latter, the control chain monitors the shards and if any come to a halt because of too many corrupted nodes, it resamples the set of nodes running that shard. This re-establishes liveness and with safety and liveness in place, we have security.

Intershard signalling

Intershard signalling allows transactions between shards and communication of smart contracts on different shards. Our protocol for this operates as follows. When a block finalizes on a shard, it contains a list of outgoing messages for other shards. The nodes of the sending shard sign the list of outgoing messages. The nodes in the receiving shard can

obtain the list of nodes running on the sending shard together with their public keys from the control chain, which allows them to verify the signed messages from the sending shard. Once a message is verified, it is executed on the receiving shard.

Private shards

The sharding mechanism also allows for private shards. In a private shard, the control chain cannot see the transactions on the shard, it only provides Finalisation-as-a-Service (FaaS) and coordination to relaunch deadlocked shards. The private shard can ultimately run its own consensus algorithm and use its own identity providers and anonymity revokers. Private shards provide a cheap way for an individual, country or corporation to launch their own blockchain while having the benefit of the strong finalisation service provided by the control chain.

Identity Layer

This section describes Concordium's innovative identity layer that allows users to create a verifiable identity off-chain to facilitate compliance with relevant regulations, while also allowing that identity to be represented on-chain in a way that protects the user's privacy. Concordium's identity layer has been proven secure in [DGKOS21].

Previous blockchains have chosen extreme balances between user privacy and accountability. Some blockchains allow fully anonymous transactions without any accountability, making them vulnerable to illegal activity. Equally troubling is that while some blockchains do not provide true anonymity for transactions, allowing for transactions and accounts to be tracked, they offer no systematic way to discover the real-world identity of suspicious users.

Concordium offers a workable solution by providing transactional privacy for users, along with a mechanism that allows accountability to local regulations. This means that transactions are processed without exposing the identity of the sender or receiver. In case of encrypted transfers (explained later), sender and receiver will also be the only parties that can see the actual amount of a transaction. On the other hand, where a suspicious transaction or set of transactions have been detected, the real-world identity of the user can be obtained by qualified authorities with the help of anonymity revokers and identity providers. Moreover, if a specific real-world identity is the subject of an authorized investigation, anonymity revokers and identity providers can help trace all accounts and transactions of that user.

Entities in the Identity Layer

This section provides an overview of the entities involved in the identity layer.

Identity provider

An **identity provider** is a person or organization that performs off-chain identification of users. For each identity issued for a user, the identity provider stores a record off-chain

called an **identity object**, and the user gets a corresponding private part of the identity object, called **user identity certificate**[1], which is known only to the user.

The primary functions of an identity provider are to:

- Verify the identity of users;
- Issue user identity certificates to users;
- Create and store identity objects and relevant attributes for record-keeping purposes; and
- Participate in the anonymity revocation process.

Information about the organizations that act as identity providers, such as their name, location or public key, is found in an on-chain registry. Initially, the registration of identity providers will be managed by the Concordium Foundation. Users must go through the identification process with a registered identity provider in order to open and operate an account on the Concordium Platform.

Anonymity revokers

An **anonymity revoker** is a person or organization that is trusted by the Concordium Platform to help identify a user that owns an account should the need arise. Initially, anonymity revokers will be appointed by the Concordium Foundation.

All accounts on the Concordium Platform are associated with a real-world identity, which is linked to an identity object stored by an identity provider. Identity objects are also linked to a set of anonymity revokers. Anonymity revokers play a critical role in revealing the real-world identity of a suspicious user by decrypting the **unique user identifier** that is stored on-chain for each account. When a unique user identifier has been decrypted following service of an official order (as described below), it can be combined with information stored by the relevant identity provider to allow the qualified authorities to obtain the real-world identity of the user.

Processes

In this section we describe the processes related to the identity layer.

Opening of initial account and identification

Before an individual or entity can use the Concordium Platform, their real-world identity must be verified and recorded by an identity provider. This identification is performed during the initial account creation.

Before an identity provider can verify a user's identity, the user must first complete the process of creating **user identity information** via a purpose-built wallet or app. The user identity information includes attributes of the user, such as age or nationality. The user may choose to associate these or a function of these (e.g. "age > 21") with any account that the user creates. The user can also decide to keep these attributes private in which case they are not visible to the public. However, they can be revealed during anonymity revocation.

---

[1] User identity certificates should not be confused with other types of certificates such as X.509, which are public certificates. The user identity certificate is private to the user and never shown on the chain.

The user additionally creates account keys for an initial account, which the user stores privately. The identity provider then verifies that the attributes in the user identity information are valid for the user and stores them locally in an **identity object** that is specific to the user. Identity objects are only held by identity providers. The identity provider then opens an account, the initial account, on behalf of the user. At the end of the identity verification process, the user receives a **user identity certificate** that can be used for creating additional accounts and the user gets access to the initial account on the Concordium Platform. These certificates are valid for a given period and users can obtain new certificates in connection with updated identity verification by an identity provider.

While the identity provider can link the initial account to the user, nobody else can. Based on the user identity certificate the user can subsequently create other accounts (see below) that can only be linked to the user if the anonymity revokers and the identity provider are involved. This gives a user a way to create accounts with an additional layer of privacy protection compared to that in the initial account.

Creating additional accounts on the Concordium Platform

Once a user has acquired a user identity certificate from an identity provider, they can create more accounts on the Concordium Platform. This is typically done using an app or wallet that guides users through the account creation process.

Based on the user identity certificate, the Concordium Platform allows a user to create additional accounts. While the private account keys are stored by the user, public **account creation information** is published on the blockchain. The latter contains public account keys, a subset of the user identity information (the specific set of attributes to be publicly revealed for a given account can be chosen when creating the account), and information about the identity provider and anonymity revokers. While this allows the relevant anonymity revokers and the identity provider, when working together, to link the account to the user, the account creation information does not allow other parties or any single party to identify the user. Further, accounts created with the same user identity certificate cannot be publicly linked.

Multi-user accounts on the Concordium Platform

The Concordium Platform also allows users to create jointly owned accounts. For example, three users can have a joint account where two of them are needed to authorize a transaction. The total number of users and the authorization threshold can be configured freely by the users. To create a multi-user account, one user creates a normal account and the other users generate account credentials that can then be added to the account by the initial owner. All added credentials contain identity information on the users, which, similar to a normal account, allows the relevant anonymity revokers and the identity provider, when working together, to link the account to the users.

Anonymity revocation

The identity of a user can only be revealed to a qualified authority as part of a valid legal process. A **qualified authority** is a governmental body that has the authority to act in a

relevant jurisdiction. For example, a local police force, a local court or an investigatory division of a local authority that regulates financial conduct may have authority to act in their relevant jurisdictions. These authorities are qualified to begin the process of revoking the anonymity of a user when they proceed through established legal channels and make a formal request. The outcome of such a request is likely to be that a qualified authority obtains an **official order**, which may be in the form of a warrant, court order, or similar instrument. Only after a qualified authority validly serves an official order upon the relevant anonymity revokers and identity provider, can the real-world identity of a user be revealed and only to the extent set out in the order.

Concordium's identity layer is flexible and sensitive to the evolving nature of financial regulation and its impact on the blockchain space. Where new legislation or rules emerge, for instance the application of the so-called "travel rule" to blockchain transactions, the Concordium identity framework offers a compliance-centric solution that can be tailored to specific business needs.

After the authorities have identified an on-chain transaction or account they would like to investigate in order to reveal the real-world identity of a user, one of the following two processes must be followed.

If the account is an initial account the authority can work directly with the identity provider to get the real-world identity of the user. For other accounts the anonymity revokers must be involved as well:

- The qualified authority must identify the anonymity revokers and identity provider associated with the account under investigation and present them with an official order. This information is available on-chain as part of the account creation data.
- Per the terms of the official order, the anonymity revokers will extract parts of[2] the unique user identifier for the user by inspecting and decrypting the available on-chain data.
- The qualified authority can now combine the parts received from the anonymity revokers to reconstruct the unique user identifier,
- With this unique user identifier, the qualified authority can work with the relevant identity provider to retrieve the real-world identity of the user.

[2] More precisely the unique user identifier is "secret shared" and after decryption each anonymity revoker can return its share of the identifier.

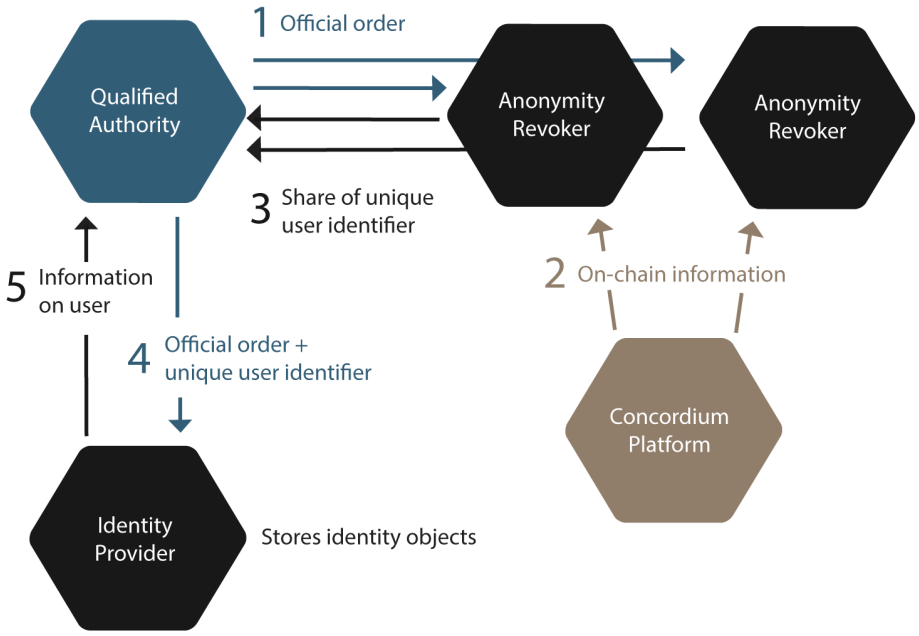


Figure 2: The anonymity revocation process.

If required and per an official order, all accounts and transactions related to a specific user can be uncovered in a similar process.

It is important to note that this process intends to dovetail into contemporary legal systems that have established checks, balances and controls to prevent overreach by qualified authorities. It also provides that a user can create accounts for which neither identity providers nor anonymity revokers acting alone can reveal an on-chain user's real-world identity on their own.

Execution Layer

Users interact with the Concordium Platform via different types of transactions. Once transactions are submitted, they are added to the transaction pool. Bakers check the validity of transactions and include valid transactions into the next block. Transaction validity and the manner in which transactions are handled is dependent on the type of transaction. We discuss the most important types below.

All transactions have timeouts. The **timeout** is set by the creator of the transaction and a transaction cannot be executed after its timeout time has expired. This prevents the situation where an old payment can suddenly execute days later.

Transaction Types

Account-related transactions

A user opens a new account by publishing an **account-opening transaction**. This transaction contains account creation information (see the above section, *Identity Layer*).

Plain transfers

The most basic type of transaction is a **plain transfer** of *x* GTU from *Account A* to *Account B*. Such a transaction is valid if the public balance of *Account A* is at least *x + transaction fee*. The effect of the transaction is that the public balance of *Account A* is reduced by *x + transaction fee* and the public balance of *Account B* is increased by *x*. Both accounts and the value *x* are publicly visible for this type of transaction.

Encrypted transfers

The Concordium Platform supports **encrypted transfers**. To allow for encrypted transfers, accounts have an **encrypted balance** in addition to their public balance. An encrypted transfer has the same functionality as a plain transfer except that it operates on the encrypted balances and the transferred amount is hidden and only known to the sender and receiver. To ensure that the sender has a sufficient encrypted balance, the transaction contains a zero-knowledge proof that allows everyone to verify that the amount in the transfer does not exceed the sender's encrypted balance, without revealing any of these values.

Authorities are able to reveal encrypted amounts in a process similar to that used for anonymity revocation.

Anonymous transfers

While encrypted transfers hide the amount of GTU it is possible to see which accounts are involved in the transfer. An additional level of privacy is added with **anonymous transfers**. An anonymous transfer has the same functionality as an encrypted transfer with the addition that the sender and receiver of a transaction cannot be linked.

Concordium plans to add, in a future release, support for anonymous transfer using various new and well-known techniques. Note that the anonymity revocation process allows authorities to obtain information about the transaction hidden in an anonymous transfer.

Smart contract-related transactions

The life of a smart contract starts with the deployment of the smart contract code. Any user can deploy smart contract code using a special transaction and can get an instance of a deployed smart contract using an **initialization transaction**. Each initialized smart contract is associated with an account. An instance of a smart contract can receive inputs in the form of **input transactions**. These transactions specify an amount of ENERGY that is allowed for the execution of the input. The order in which different inputs are executed depends on the order in which the input transactions are added to blocks.

Consensus-related transactions

Users can become bakers by publishing a special transaction. The newly created baker is associated with one of the user's accounts, which is used for holding the staked amount of GTU and for receiving rewards. Furthermore, there are transactions to change the stake of the baker and to deregister as a baker.

Smart Contract Languages

WebAssembly low-level language

Concordium's core on-chain language is WebAssembly (Wasm), a portable, well-defined assembly-like language. Wasm is an internet standard which is gaining a lot of traction in recent years and is already supported in the major web browsers.

Many programming languages can already be compiled to Wasm, which potentially enables support of a large range of smart contract languages. Wasm allows for low-level control of the on-chain code, which helps with optimisations when adding support for cryptography in smart contracts.

Among permissionless blockchain platforms, there are very few common standards for smart contracts, however, Wasm is one of the few that is seeing adoption by multiple platforms.

Rust high-level language

Concordium plans to support a number of smart contract languages and has chosen Rust as the first high-level smart contract language. The Rust ecosystem is quite friendly, with good documentation, and good support for WebAssembly.

Rust is a safe language, but it also allows for low-level resource control. This can help reduce the cost of contracts and makes it very well-suited for development of cryptographic primitives and protocols. Many high-quality libraries exist that can be used off-the-shelf and compiled to web-assembly.

Concordium also provides additional validation of Rust code so that generated modules conform to on-chain requirements: for instance, that smart contract entry-points have appropriate types.

Ultimately, however, any language that is able to compile to WebAssembly will be able to target the Concordium chain.

Interoperability

Blockchain interoperability can be considered at different levels such as

- Inter blockchain ordering of events
- Interfacing to external applications. This could be for retrieving external information to be used in in the blockchain ("oracles") or for emitting verifiable information from the blockchain
- Interoperability at smart contract level making it easier to develop decentralized applications based on smart contracts for different blockchain platforms.

Ordering of events across different blockchains can be handled by submitting transactions to the blockchains. However, in practice, such synchronisation between blockchains is only as precise as the latency on the different blockchains. The design of the Concordium Consensus layer with almost immediate finalisation and thereby protection against roll-back

is well suited for such inter blockchain ordering. The Finalisation-as-a Service functionality used for handing shards as described earlier can in the future be extended to other blockchains to allow an efficient and precise ordering between different blockchains.

Regarding interfacing to other applications, the finalisation committee in the Concordium Platform is in principle able to create messages authenticated by the blockchain. The format of such outgoing messages is independent of the block structure on the chain and will follow a standardised format. The main hurdle is to create these authenticated messages in such a way that the recipient does not have to travel through the complete chain in order to reach a point of trust in the form of the Genesis Block. Concordium is working on solutions to this.

Another aspect of the external interface is providing external data to the blockchain (decentralized oracles). Concordium will implement one such oracle shortly after the launch of Mainnet as it will be needed to dynamically update exchange rates in our tokenomics model.

With respect to smart contracts, Concordium has a strategy to use existing programming languages with a strong community of developers in order to lower the threshold for many developers to start working with smart contracts. Furthermore, we expect that the choices of Rust as the initial smart contract languages and Wasm as on-chain languages will make it easier to port smart contracts between different blockchain.

Tokenomics and On-chain Incentivization

The Concordium Platform comprises a set of transactions and economic roles that interact within the economy. An economic role, such as a baker or identity provider, exists either off-chain or is represented by an account on the Concordium Platform.

The flow of GTU between accounts via transactions creates an economy that is designed to incentivize participation in the network and counter dishonest behaviour. It is the objective of the Concordium Foundation to guide the creation of a sustainable economy that rewards participants for their efforts in developing the network.

Overview of the Concordium Platform Economy

The Concordium Foundation is responsible for maintaining the health of the economy through monitoring of internal dynamics and scrutiny of the impact of external market conditions. The Concordium Foundation will adopt a flexible approach to nurturing the Concordium Platform economy as it evolves to include more complex dynamics and transactions.

See *Figure 3* below for a visualization of the various roles, accounts and transactions in the Concordium economy, both on-chain and off-chain. In this figure, the blue lines indicate the core flow supported at the time of Mainnet launch while the dotted lines indicate flows planned for future updates.

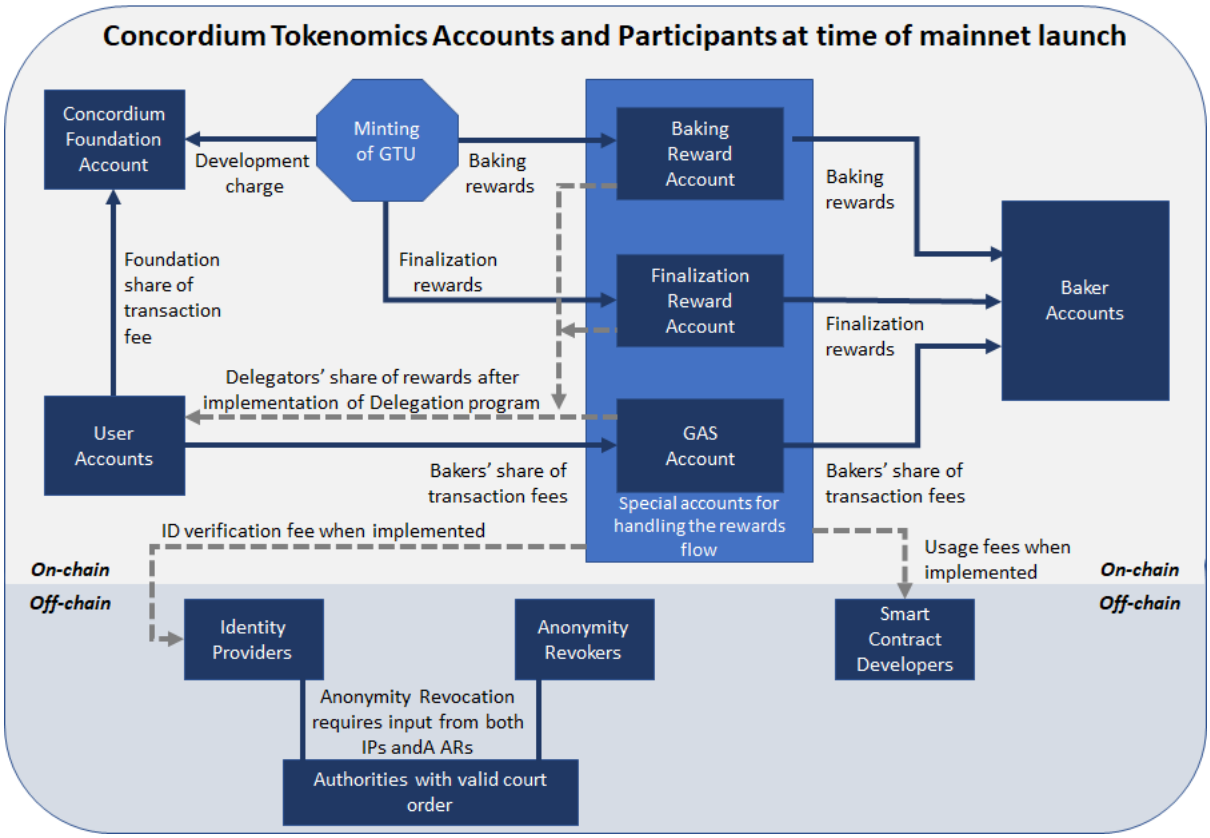


Figure 3: The Concordium Platform economy and roles.

GTU - Concordium's Native Token

Global Transaction Unit (**GTU**) is the native token on Concordium Platform. GTU is a payment token, which can be used for a variety of purposes, including as payment for execution of smart contracts, payments between users, payments for commercial transactions, and as a store of value. A specific number of GTU is created in the genesis block. After this, the only mechanism to create more GTU is the minting process. The number of GTU that exists on the platform at any time is defined and publicly known.

Roles and Participation in the Economy

There are a number of transactions that allow users and other entities to interact with the economy and each other within the Concordium Platform. Many of these transactions are discussed at length in the section *Execution Layer* above. Specific roles within the Concordium Platform are:

Users can be individuals, businesses, and other identifiable legal entities that create and control accounts within the Concordium Platform.

Users participate in the Concordium Platform economy as follows:

Creating an Account:

1. Account-opening and account-update transactions allow a user to create an account on the Concordium Platform.
2. Initially, there will be no payment required from the user to the identity provider for the creation of a user's identity object, the costs being covered by the Concordium Foundation to stimulate the ecosystem. However, in the future, users may have to pay a fee to identity providers, or the identity providers may get a fee as part of the overall tokenomics flow as indicated in Figure 3.

Making transactions:

1. Users, via an associated account, can initiate transactions, including plain transfers of GTU to other accounts, registering information on the chain, or transferring other tokens that exist on the Concordium Platform. In addition to the transactions enabled on the blockchain, smart contracts provide powerful programming functionality to handle specific applications.
2. Users can receive GTU into their account when their address has been specified as the recipient in the transaction. It will later be possible to opt-out for the receipt of GTU from other users that meet certain criteria.
3. Users will be able to make encrypted transfers as well as anonymous transfers.

Staking:

1. A user can stake part of the GTU on the user's account. This is required if the user wants to operate as a baker.
2. Rewards associated with a user's stake will be automatically transferred to the user's account.

Delegation:

1. In a future update of the Concordium Platform, a user can initiate a delegation transaction to delegate an amount of GTU to a pool of bakers (the Delegation To Pool feature). Likewise, a user can initiate a similar transaction to un-delegate GTU, i.e. withdraw GTUs from the Pool.
2. Rewards associated with a user's delegated GTU will be automatically transferred to the user's account.

Similar to other users, **the Concordium Foundation** has a number of accounts and runs a number of nodes as bakers. Furthermore, the Concordium Foundation plays a central role in the governance of the chain including the management of tokenomics parameters as described in the section on Governance. The Concordium Foundation is the recipient of the Platform Development Charge that comprises part of newly minted GTUs and part of the transaction fees.

Bakers are created when users issue special transactions to register as a baker.

Finalizers. Bakers automatically become finalizers once they reach a certain threshold of GTU stake. The Concordium Foundation will control the threshold for how much GTU must be staked (initially set to 0.1% of all staked GTU) and make adjustments to increase or decrease the number of finalizers with the view of securing the network without compromising speed or efficiency.

Smart Contract Developers: Any user will be able to write, deploy, or use a smart contract on the Concordium Platform. Users that publish a smart contract may also be rewarded for the use of that smart contract by other users. In the future, this process would entail an *App Store*-like library of certified smart contracts. This will allow users to use high-quality code without having to develop it themselves.

Identity Providers perform off-chain identification and create identity objects. Identity providers do not need an account. Identity providers are initially paid in fiat money off-chain but may in the future be incentivized with GTU on-chain for the process of creating identity objects.

Anonymity Revokers perform decryption of encrypted user identifier numbers related to accounts and disclose such numbers to relevant authorities if presented with a valid court order. Anonymity Revokers do not need an account.

Special Accounts

As depicted in Figure 3, there are initially three special accounts within the Concordium Platform. These hold the GTU that are used for fees and rewards to incentivise participants in the network.

The **Baking Reward Account** holds the pool of GTU from which rewards are distributed to bakers and users that have delegated stake. GTU are deposited into the Baking Reward Account from the minting of new GTU.

The **Finalization Reward Account** holds the pool of GTU from which rewards are distributed to Bakers who are also Finalizers and users that have delegated stake. GTU's are deposited into the Finalization Reward Account from the minting of new GTU.

The **GAS Account** holds the pool of GTU from which transaction fees are distributed to bakers. GTU are deposited into the GAS Account from the users by way of payment of transaction fees.

Minting of GTU

The process of minting new GTU is an automatic feature of the Concordium Platform whereby newly minted tokens are transferred partly to reward accounts for distribution as incentives and partly to Concordium Foundation. Minting is the only source of growth in the number of GTU in existence inside the Concordium Platform and, initially, the growth rate will be controlled by the Concordium Foundation.

Minting is governed by a parameter which controls how many GTU are created per slot. This ensures that the GTU growth rate is linked to time and does not depend on fluctuations in the rate by which blocks are created. Initially, the Concordium Foundation will control the GTU/slot parameter and monitor the impact of the GTU growth on the overall tokenomics to ensure the sustainability of the economy.

A Platform Development Charge of 10% of newly minted GTU is transferred to Concordium Foundation. The remaining 90% of the minted GTU is transferred to the Baker Reward Account (currently 85%) and the Finalization Reward Account (currently 5%).

Transaction costs for users

Users of the blockchain pay a Transaction Fee (also referred to as a payment of GAS) for each transaction they make. The fee is paid in GTU and for that purpose, users will need to acquire and hold GTUs.

Transaction costs are designed to be relatively stable in EUR terms, thereby enabling businesses and other users to predict and plan with fixed predictable EUR-costs.

With the ENERGY principle described below, Concordium combines the freely fluctuating value of the GTU with a transaction cost that remains stable versus EUR. This is achieved by making the number of GTU that shall be paid for transactions vary inversely with the price of the GTU when measured against EUR. So if the value of the GTU goes up, a user needs fewer GTUs to pay for transactions, thereby maintaining a stable cost in EUR-terms. The technical implementation of the above principle is discussed below.

ENERGY, which is an internal measure of transaction cost per transaction, comprises three elements:

Transaction Base Cost (same amount for all transaction)
\+ Complexity Premium (dependent on the computational complexity in relation to verification of the transaction and the amount of time it takes to execute)
\+ Size Premium for transactions that contain a large amount of data
= Total Transaction Cost expressed as a number of ENERGY

The calculated ENERGY is converted into a GTU-cost by applying a fixed EUR/ENERGY conversion rate and a variable GTU/EUR exchange rate which is adjusted dynamically to ensure that the cost remains stable when measured in EUR.

The EUR-cost of transactions may be changed by the Governance Committee (see the section on Governance) in response to significant changes in market conditions or changes in the costs related to processing transactions. It is envisaged that over time, the transaction costs will decrease considerably, ultimately approaching zero. Any changes to the EUR-cost of transactions will be announced to users and the public with a notice period of no less than 1 month, unless exceptional circumstances occur, thereby enabling users to make necessary adjustments to their use of the blockchain.

The GAS-payment in GTU-terms is calculated by applying the following calculation:

GAS (GTU) = ENERGY x (EUR/ENERGY conversion) x (GTU/EUR exchange rate)

Prior to the launch of the blockchain, the precise calculation principles for size and complexity will be prepared and made public.

The GTU/EUR conversion rate required to perform the conversion will be supplied from an external source. Initially, the Concordium Foundation will provide this data. Subsequently,

the data will be provided using an Oracle to pull data from exchanges with sufficient liquidity and volume, scrutinising for Market Manipulation.

The GTU paid as fees for transactions are partially deposited in the GAS account (45%) and partially paid to the baker of the corresponding block (45%) while the remaining 10% are transferred to the Concordium Foundation.

Rewards from Special Accounts

The rewards are paid to bakers, finalizers (i.e., bakers who are also finalizing) and future delegators from the special accounts. The Tokenomics System comprises the following rewards:

- Baking rewards paid with newly minted GTU to bakers from the Baking Reward Account.
- Finalization rewards paid with newly minted GTU to finalizers from the Finalization Reward Account.
- Transaction rewards to bakers.

The mechanisms for these rewards are described below. It should be noted that upon implementation of the Delegation To Pool feature, expected within nine months after mainnet launch, users who delegate their GTU (stake) to the delegation pool will also participate in the reward scheme and will receive part of the rewards paid from the Baking Reward Account, the Finalization Reward Account and the GAS Account.

Furthermore, additional on-chain incentive rewards to Smart Contract Developers will be implemented later but are not part of the mainnet solution. At the time of mainnet launch, incentive models for Smart Contract Developers are provided through a Community Endowment Programme.

Baking Rewards

Baking Rewards are paid to the bakers from the Baking Reward Account. At the end of an epoch, the current balance of the Baking Reward Account (defined as the balance when the last block of the epoch is created) is distributed to bakers with the allocation between bakers being based on how many blocks each baker produced in that epoch. I.e, the total baking reward paid to a baker is calculated as

> No. of blocks baked on the longest chain during the epoch by the baker
> divided by
> Total no. of blocks baked by all bakers on the longest chain during the epoch
> multiplied by
> Balance of the Baking Reward Account at the end of the epoch

The baking rewards are distributed to the bakers as part of the first block in the following epoch.

Finalization Rewards

Finalization rewards are paid from the Finalization Reward Account to bakers who are also finalizers.

The rewards are paid out once a block has been finalized and a baker has added the corresponding finalization proof to a block. The payout is equal to the full current balance in the Finalization Reward Account.

The rewards to finalizers are allocated to all members of the finalization committee and are split among the individual finalizers proportional to their stake. That is, a finalizer with stake S gets (S / sum of stakes of all finalizers) of the current balance of the Finalization Reward Account.

For the block that includes the finalization proof of a previous block, an additional transaction fee is taken from the GAS account and added to the transaction fees in the block (in order to incentivize bakers to include these proofs).

Transaction Rewards

The transaction reward for a block is distributed to the baker when the block is created. The size of the transaction reward depends on the fees paid for the transactions in the block and the amount in the GAS Account. More precisely, the transaction reward to the winning baker is computed as:

```
    A fraction, initially set at 45%, of the fees from the transactions in that block
+   A fraction, initially set at 25%, of the balance of the GAS Account before this block
```

Recalling that 10% of the transaction fees are distributed to the Concordium Foundation, the remaining 45% of the fees from the transactions in the block are added to the GAS Account. Thus the value of the GAS Account is changed to:

```
    75% of the old balance of the GAS account
+   45% of the transaction fees in the new block
```

Using the GAS Account as a buffer for some of the transaction fees makes the transaction rewards more stable despite fluctuations in the transaction fees for different blocks.

Relationship between staking, GTU growth, and return on staking

Because the number of minted GTU per block (and thus the rewards) are linked to the GTU growth while the amount of staked GTU is the result of each individual baker's staking decisions, the Return on Staking (RoS) for bakers and finalizers will fluctuate over time.

As described above, rewards being paid out to bakers and finalizers will be 90% of new minting. If the GTU growth rate is 2%, and 50% of all GTUs are staked, then the RoS will be:

2% x 90% / 50% = 3.6%

If the number of staked GTUs drops to 25%, the RoS will be:

2% x 90% / 25% = 7.2%

The initial GTU growth rate after mainnet launch is set to 10%. If the GTU growth rate is 10%, and 50% of all GTUs are staked, then the RoS will be:

10% x 90% / 50% = 18.0%

If the number of staked GTUs drops to 25%, the RoS will be:

10% x 90% / 25% = 36.0%

When the amount of staked GTU goes up, the Return on Staking goes down, and vice versa. This relationship is seen as having a stabilizing effect as it makes it financially more attractive to stake when there is a low amount of GTU being staked and vice versa.

Managing baker status and staking

If a user wants to bake/finalize in epoch i, they must register as a baker before the start of epoch i-1. A longer delay may be implemented at a later stage.
The baker registration includes the following information:

- An account number. This account will be used for reward payout and will also be where the stake is stored.
- Stake limit. The amount to stake. This amount must be less than the balance on the account at that point in time.

By default (may be changed in the future), rewards are added to the stake immediately. This means that the stake limit is increased automatically when rewards are received. The account owner is given the option to not add rewards to the stake, meaning that the stake limit is not increased when a reward is received and the amount can be transferred freely from the account.

As long as the baker/finalizer is active, the stake limit GTU amount is locked in the account and cannot be moved. Any amount above the stake limit can freely be transferred to another account or used for paying transaction fees.

If a baker wishes to change his/her stake it can be done by re-registering as a baker with a higher or lower stake limit. The stake limit GTU has to be available on the account for the transaction to succeed, otherwise, the baker will need to transfer extra GTU to the account before re-registration.

If the baker wants to increase his stake for epoch i, the re-registration must be done before the start of epoch i-1.

If the baker wants to decrease the stake, the change will take effect after 168 epochs corresponding to 7 days. During this grace period the baker will continue to bake with the original stake.

If a baker wants to stop baking in epoch j, the baker has to deregister as a baker before the start of epoch j-168.

Governance

The Concordium Foundation is supervised by the Swiss authorities, with a clear purpose defined in its Public Deed. The Foundation Board is tasked with ensuring that the Concordium protocol continues to develop and remain relevant to the needs of users within the parameters of the Foundation's purpose.

GTU holders will assume a central role in suggesting and determining the priorities for the Concordium platform development through the Governance Committee, whilst the Foundation Board will act as a guarantor that the fundamental Concordium principles of privacy with accountability and the Public Deed of the Concordium Foundation are adhered to. The Foundation Board will retain the role as the supreme body of the Foundation after full decentralization, but key functions will be delegated to the Governance Committee.

Decentralisation will be achieved in four steps.

After the launch of the Mainnet
The Concordium Foundation Board appoints a Governance Committee with five members to oversee the governance decentralization process, the on-chain governance mechanisms, and the tokenomics, including the GTU growth rate.

At this stage, the Governance Committee is 100% appointed by the Concordium Foundation Board.

Mainnet + 2 years
GTU holders appoint two members of the Governance Committee by on-chain vote, expanding the Committee from five to seven members. The Governance Committee can propose changes to the protocol on-chain.

At this stage, 28.6% of the Governance Committee is appointed by the GTU holders.

Mainnet +4 years
GTU holders appoint two further members of the Governance Committee by on-chain vote, expanding the Committee from seven to nine members. holders can propose changes to the protocol and vote on the changes on-chain. The Governance Committee will oversee the implementation and ordering of decisions made by the holders.

At this stage, 44.4% of the Governance Committee is appointed by the GTU holders.

Mainnet +6 years
GTU holders appoint two further members of the Governance Committee by on-chain vote, replacing two of the initial five Committee seats appointed by the Foundation Board. The Committee’s total membership remains at nine members, with two seats up for election every year. GTU holders can implement changes to the protocol. The Governance Committee acts as the executor of decisions passed by the GTU holders.

At this stage, 66.6% of the Governance Committee is appointed by the GTU holders.

Mainnet +8 years
At this stage, 100% of the Governance Committee will be appointed by the GTU holders.

References

An up-to-date list of research papers from the Concordium Blockchain Research Center Aarhus (COBRA) can be found under https://cs.au.dk/research/centers/concordium/publications/.

[ANS19] Annenkov, D., Nielsen, J., Spitters, B. “ConCert: A smart contract certification framework in Coq.” Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, 2019. https://doi.org/10.1145/3372885.3373829.

[CM19] Chen, J., Micali, S. Algorand: A secure and efficient distributed ledger, Theoretical Computer Science, Volume 777, 2019. https://doi.org/10.1016/j.tcs.2019.02.001.

[DGKOS21] Damgård I., Ganesh C., Khoshakhlagh H., Orlandi C., Siniscalchi L. (2021) Balancing Privacy and Accountability in Blockchain Identity Management. Topics in Cryptology – CT-RSA 2021. Lecture Notes in Computer Science, vol 12704. Springer, Cham. https://doi.org/10.1007/978-3-030-75539-3_23

[DGKR18] David B., Gaži P., Kiayias A., Russell A. Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain. Advances in Cryptology – EUROCRYPT 2018. Lecture Notes in Computer Science, vol 10821. Springer, Cham, 2018. https://doi.org/10.1007/978-3-319-78375-8_3.

[DMMNT20] Dinsdale-Young T., Magri B., Matt C., Nielsen J.B., Tschudi D. "Afgjort: A Partially Synchronous Finality Layer for Blockchains." Security and Cryptography for Networks. SCN 2020. Lecture Notes in Computer Science, vol 12238. Springer, Cham. https://doi.org/10.1007/978-3-030-57990-6_2

[DMMNT21] David, B., Magri, B., Matt, C., Nielsen, J.B., Tschudi, D. “GearBox: An Efficient UC Sharded Ledger Leveraging the Safety-Liveness Dichotomy” IACR Cryptology ePrint Archive, Report 2021/211, 2021. https://eprint.iacr.org/2021/211

[DSTT19] Dinsdale-Young, T., Spitters, B., Thomsen, S., Tschudi, D.: WIP: Formalizing the Concordium consensus protocol in Coq. CoqPL 2019. https://cs.au.dk/~sethomsen/coqpl19.pdf.

[GOT19] Ganesh C., Orlandi C., Tschudi D. (2019) Proof-of-stake protocols for privacy-aware blockchains. Advances in Cryptology – EUROCRYPT 2019. Lecture Notes in Computer Science, vol 11476. Springer, Cham, 2019. https://doi.org/10.1007/978-3-030-17653-2_23.

[Kwo14] Kwon, J. Tendermint: Consensus without mining. Manuscript, 2014. https://tendermint.com/static/docs/tendermint.pdf.

[KMM+20] Kamp, S., Magri, B., Matt, C., Nielsen, J., Thomsen, S., Tschudi, D.: Leveraging weight functions for optimistic responsiveness in blockchains. IACR Cryptology ePrint Archive, Report 2020/328, 2020. https://eprint.iacr.org/2020/328.

[NS19] Nielsen, J., Spitters, B.: Smart contract interactions in Coq. CoRR abs/1911.04732, 2019. https://arxiv.org/abs/1911.04732.

[Per16] Perrin, T. The noise protocol framework, 2016. http://noiseprotocol.org/noise.pdf.

[TS20] Thompson S. E., Spitters B.: Formalizing Nakamoto-Style Proof of Stake, 2020. ePrint.