Concordium White Paper

An Introduction to the Technical Specifications and Features of the Concordium Platform

Ivan Damgård[1], Hans Gersbach[2], Ueli Maurer[2], Jesper Buus Nielsen[1],
Claudio Orlandi[1], Torben Pryds Pedersen[3]

[1]COBRA, Aarhus University, Denmark
[2]ETH Zurich, Switzerland
[3]Concordium AG, Switzerland

Concordium is a privacy-focused, public and permissionless blockchain architecture. This paper introduces the technical specifications of 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 payments 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 designed new languages for writing smart contract code that make development easier and smart contracts more reliable. The Concordium Platform also features a transparent incentive structure with cost-effective transactions and predictable fees.

Vol 1.0, April 2020

Editors:

Thomas Graham, Christian Matt, Mario Rogic, Daniel Tschudi

Acknowledgements:

We would like to thank everyone from the team at Concordium that has contributed to the project, as well as the following external contributors: Diego Aranha, Chaya Ganesh and Bas Spitters.

Table of Contents

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.

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 any business application 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. We achieve significant speedups and efficiencies under normal conditions, where less than 33% of all stake is controlled by malicious parties. Unlike some platforms, Concordium also remains available and secure as long as less than 50% of all stake is controlled by malicious parties. Furthermore, the safety of our finality layer holds even under catastrophic failures to the network that cause 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 has developed a novel sharding mechanism that operates in tandem with our finality layer. Concordium can run and coordinate several sub-blockchains in parallel, each much faster than standalone blockchains. Transfers and communication across shards are also supported by a novel design for intershard signaling.

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 a 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.

Easy to use smart contracts fuel business applications

For businesses that rely on smart contracts, it is critical that they operate as intended and without bugs or security issues. Concordium has developed two novel programming languages, Midlang and Retlang, that make smart contracts easier to develop and more reliable to deploy. Midlang, built on Erlang and Elm, models problems clearly and precisely, while also being simple enough to allow for easy on-boarding. Midlang code compiles to Retlang, a new low-level language that allows for formal verification and static analysis of smart contracts. Concordium is also developing tools that will allow formal verification of smart contracts [5,8].

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

The Concordium Foundation will initially be responsible for all governance and development decisions regarding the platform. Over time, as the platform matures, governance will be decentralized across network participants. In the future, the protocol specifications and codebase for all components of the Concordium Platform will be open-source.

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 a 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.

The **identity layer** defines how users' identities are processed. It specifies how identity providers and anonymity revokers interact with users to create identity objects and revoke the anonymity of identity objects 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.

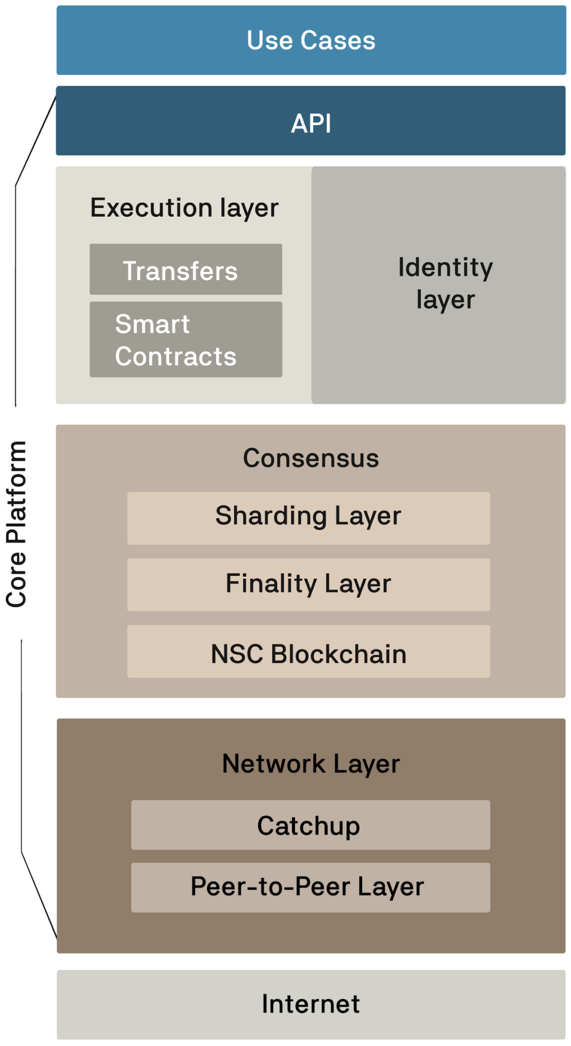


Figure 1: Overview of the Concordium stack.

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 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 and always available.

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 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 are 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 are 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

Here we describe Concordium's consensus mechanism. 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 run their own node or delegate their GTU to a baker. Across the network, the more stake that participates, the harder it becomes for malicious parties to control a majority of that stake. To maximize the amount of participating stake, we allow for the delegation of stake by stakeholders that do not want to run a node. The effective stake of a node equals the stake of that node plus the stake delegated to it. Delegation and baking can reveal information about a user's stake, as such we have investigated methods to increase user privacy in PoS systems, see Ganesh et al. [GOT19].

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 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 in the chain below it.

NSC blockchains are secure where corruption is below ½, where 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 stake.

Our two-layer approach

Concordium's finality layer can be added on top of NSC blockchains [DMMNT19]. 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 units called a **slot**, and in every slot, each baker checks locally whether they won the lottery. 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** here is a random value that is periodically updated 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 roughly proportional to α, and higher difficulty parameters decrease 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 difficulty parameter f needs to be set such that these collisions do not happen too often and winners are spread out sufficiently to accommodate for network delays.

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.

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 below the last finalized block and finalized blocks are never rolled back.

Finalization committees

Finalization is run by a **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. For example, the requirement that finalizers hold at least 0.1% of the total stake ensures that there will be at most 1000 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. When a finalizer has a chain that has reached depth $d + 1$, it 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 the ith retry, they wait until they are at depth $d + 2^i$ and vote for the block they see at depth d on their chain. Eventually 2^i 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 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 i. After a 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.

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. [DMMNT19], we obtain the following guarantees from our finality layer:

- **Chain-Forming:** that finalized blocks form a chain;
- **Agreement:** that all parties agree on the finalized blocks;
- **Updated:** that the last finalized block does not fall too far behind the last block in the underlying blockchain;
- **⅓-Support:** that 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 run 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

We will use a **control chain** that can spawn shards. The control chain runs a full-fledged 2-layer consensus mechanism as described above. Each shard runs an individual blockchain, and uses a **finality-as-a-service** (**FaaS**) component of the control chain to finalize. This means finalization on shards is performed by the nodes in the shards together with the finality committee of the control chain, and the control chain keeps track of finalized blocks on all shards.

Obtaining security and efficiency

For optimal efficiency, there should be many shards, each consisting only of a few nodes. However, the risk associated with sampling only a small number of nodes is that a large fraction of them are corrupted. For the blockchain 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 protocols such that safety holds even with high corruption levels, whereas the system may come to 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 while having safety. 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. With safety and liveness in place, we have security.

Intershard signaling

Intershard signaling 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 private shards. In a private shard the control chain cannot see the transactions on the shard, it only provides 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. Private shards benefit from the use of GTU and tooling provided by Concordium as well as the control chain that acts as a fallback mechanism.

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.

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, who will also be the only parties that can see the actual amount of a transaction, if the transaction is an encrypted transfer. On the other hand, where a suspicious transaction or set of transactions have been detected, the real-world identity of the user can be revealed to qualified authorities with the help of anonymity revokers and identity providers.

Moreover, if a specific real-world identity is suspected of malicious behavior, anonymity revokers and identity providers can help trace all accounts of that user.

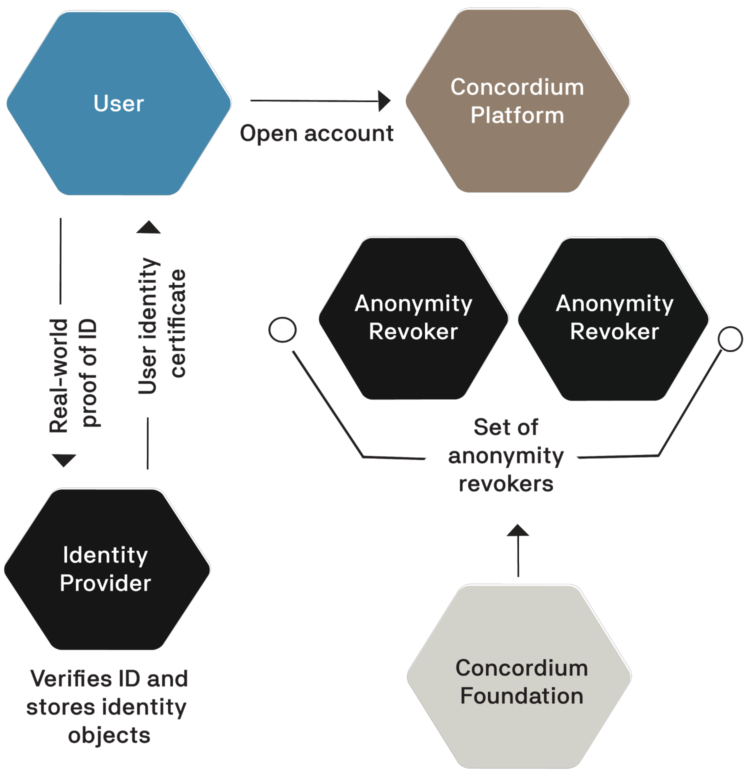


Figure 2: Concordium's identity layer

Entities in the Identity Layer

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

Users

A **user** is an 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.

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**.

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 are required to obtain an identity object from an 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 reveal the real-world identity of the suspicious user.

Processes

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

Identification of a user

Before an individual or entity can open an account on the Concordium Platform, their real-world identity must be verified and recorded by an identity provider.

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, which will ultimately be associated with any account that the user creates. 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. At the end of the identity verification process, the user receives a **user identity certificate** from the identity provider, which allows them to open an account on the Concordium Platform. User identity certificates are valid for a given number of years, after which they can be renewed in connection with updated identity verification by an identity provider. Users can elect which attributes they would like to associate with specific accounts.

Once the identity verification process is complete and an identity object for a user has been created, the identity provider cannot access the user identity information in the identity object or associate it with a specific user, account or transaction.

As set out below, the user identity information for a specific user can only be revealed where a qualified authority initiates a legal process, and works together with both the identity provider and the anonymity revokers associated with the identity object that belongs to that specific user.

Creating an account on the Concordium Platform

Once a user has acquired a user identity certificate from an identity provider, they can create one or more accounts on the Concordium Platform. This process can be automated through 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 generate a set of **account creation information**, which will be published on the blockchain. The platform also allows the user to create a set of account keys, which will be stored privately by the user. The account creation information contains: public account keys, a subset of the user identity information (the specific set of attributes to be included for a given account can be determined when creating the account), and information about the identity provider and anonymity revokers associated with the identity object. The account creation information does not allow any party to publicly identify the user. Further, accounts created with the same user identity certificate cannot be publicly linked.

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 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 will all 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 layer offers a compliance-centric solution that can be tailored to specific business needs.

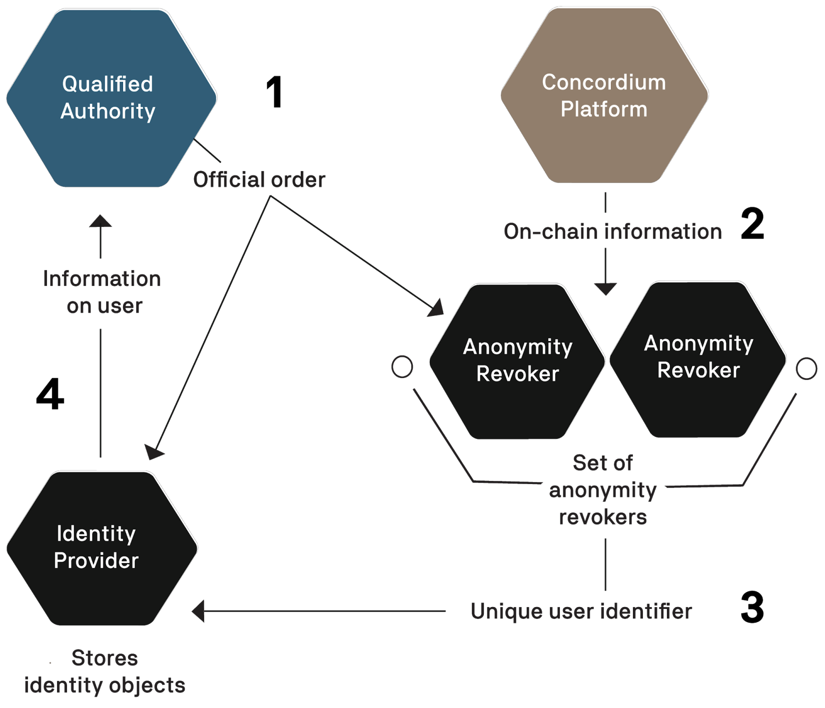
After the authorities have identified an on-chain transaction or account they would like to investigate, in order to reveal the real-world identity of a user, the following process must be followed:

- The qualified authority must identify the anonymity revokers and identity provider associated with the account they would like to deanonymize and present them with an official order.
- Per the terms of the official order, the anonymity revokers will extract the unique user identifier for the user by inspecting and decrypting the available on-chain data.
- With this unique user identifier, the qualified authority can work with the relevant identity provider to retrieve the real-world identity of the user.

If required and per an official order, all accounts which were created by a specific user can be uncovered in this 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 when acting alone neither identity providers nor anonymity revokers can reveal a user's real-world identity.

Figure 3: The anonymity revocation process.



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. In such a case, the transaction will still be added to a block and the transaction fee will be charged but the transaction will not be executed. 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*). A user can renew the user identity certificates of an account by updating the account using an **account-update transaction**.

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 (zk-SNARK) 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.

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

Anonymous intra-user transfers via a mixnet

One user can have multiple accounts on the Concordium Platform. In the future, to prevent the linking of accounts, Concordium will use a mixnet that works as follows. Users send an encrypted amount to a pool account. Bakers repeatedly shuffle all pool accounts so that they cannot be linked to the originating account. The user who made a transfer to the pool account can later retrieve that amount from the pool. Zero-knowledge proofs are used to ensure that the amounts match and that the user retrieving money has the same identity as the one who sent it to the pool in the first place. This allows users to make transfers between their accounts without others being able to link them to the same user. Concordium's mixnet design is based on the work of Fauzi et al. on Quisquis [FMMO19]. To make this process easy-to-use from a user perspective, the above functionality will be automated by the wallet used to initiate the transfer.

Anonymous inter-user transfers

A user, *User Send*, can also elect to make an anonymous transfer to another user, *User Receive*. This is implemented via the above transactions, performed automatically in the background, using dedicated single-use accounts for each transaction and user.

Before the transfer, *User Receive* needs to inform *User Send* of the details of the fresh account to be used to receive the transfer. The wallet of *User Send* will create a fresh account and then make an anonymous intra-user transfer to this fresh account. Afterwards, an encrypted transfer is made from *User Send's* fresh account to the fresh account of *User Receive*.

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. A user can use an **initialization transaction** to get an instance of a deployed smart contract. Each initialized smart contract is associated with an account. That also means that the validity of this contract depends on the validity of the associated 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. This account is used to receive the baking reward. The on-chain information of a baker can be updated by a transaction. This allows the baker to update the baking reward recipient account or the signature key. Finally, there is a transaction to deregister as a baker.

Users can delegate stake to a baker using a **delegation transaction**. There is also a transaction which allows users to un-delegate stake.

Smart Contract Languages

Concordium has developed a new language called Midlang for writing smart contract code. Midlang code is compiled into a second new language called Retlang, which then can be deployed on the chain. We briefly discuss these languages and their features below. In the future, support for additional high-level languages can be added.

Midlang® high-level language

Midlang is a high-level language for writing smart contracts. The design of Midlang builds on Erlang and Elm. This makes the language a great tool for modeling problems clearly and precisely, while also being simple enough to allow for easy on-boarding. In particular, Midlang provides the following features:

- A mathematically grounded type system which allows the compiler to eliminate whole categories of bugs.
- Type-inference, yielding the benefits of the type system without the overhead of writing out explicit type annotations every time.
- Exhaustiveness analysis, ensuring no edge cases can be ignored by the developer.
- Powerful records and custom types, which allows modeling of the contract domain in only a few lines of code.
- No side-effects, no mutable state and no escape hatches, which means we can rule out many of the most complex, subtle, and surprising bugs.
- High-quality error messages that explain any problems the compiler detects and guides the developer in the right direction making on-boarding easier, improving retention of newcomers and lowering the cost of understanding and fixing mistakes in the code.

Retlang® low-level language

Retlang is the low-level language used on the Concordium Platform. All deployed smart contracts are compiled into Retlang. The language reduces the number of language constructs making it simpler and possible to define the language and its static and execution semantics.

The Retlang typechecker is used on the chain to verify the validity of smart-contracts. Well-typed smart-contracts cannot raise runtime errors. The Retlang interpreter executes smart contracts and ensures their execution stays within the ENERGY limits imposed by the sender of the transaction.

Interoperability

Blockchain interoperability can be considered at different levels. One level is the APIs and data formats used by relying applications. Interoperability at this level will allow applications to run on different blockchains and could potentially make the application independent of a specific blockchain. Another aspect is to define a small number of common smart contract languages so that smart contracts written in these can be run on different blockchains.

Often, interoperability refers to the ability to transfer data between blockchains, typically allowing one blockchain to request data from another. Concordium is designed to support such requests as the finalization committee can sign outgoing messages on behalf of the Concordium Platform. For this to work, the receiver only needs to know the verification keys of the current finalization committee. The verification keys of the initial finality committee are in the genesis block. Using a chain of signatures we can link the current finalization committee back to the genesis block. This will allow the Concordium Platform to send short authenticated messages to other chains and other entities, such as exchanges, without the receiver running the Concordium Platform. The format of these outgoing messages is independent of the block structure on the chain and, to achieve full interoperability, a standardized format should be used. In the case that such a standard is defined and widely supported, the Concordium Platform could also be enhanced with a mechanism to request information from other blockchains obtained using this format.

In any case, whether we are talking about interoperability of APIs, smart contracts or the exchange of data, a prerequisite for a high level of interoperability is that appropriate standards are defined and applied across many blockchains. Concordium welcomes and looks forward to participating in initiatives that move in this direction.

Tokenomics and On-chain Incentivization

The Concordium Platform creates a specific 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 the creation of value on top of the Concordium Platform. It is the objective of the Concordium Foundation to guide the creation of a sustainable economy that rewards all participants for their effort in developing the network.

Economic Roles in the Concordium Platform Economy

The Concordium Foundation is responsible for maintaining the health of the economy through close 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 4* below for a visualization of the various roles, accounts and transactions in the Concordium economy.

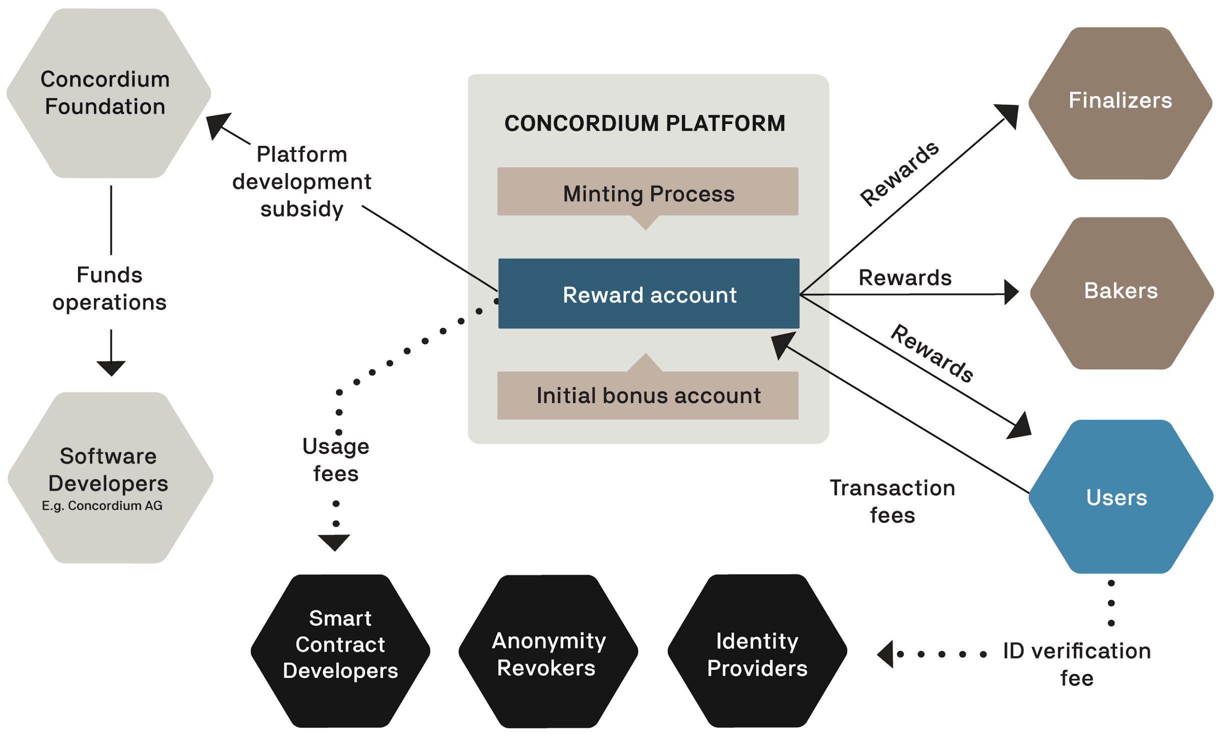


Figure 4: The Concordium Platform economy.

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 in 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 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 and legal entities, such as businesses, that create and control accounts within the Concordium Platform.

Users participate in the Concordium Platform economy as follows:

Creating an Account:

- Account-opening and account-update transactions allow a user to create an account on the Concordium Platform.
- Before a user can create an account, they must first work with an identity provider to create the identity object required to create an account. Initially, there will be no payment required to the identity provider for the creation of a user's identity object. However, in the future, users may have to pay a fee to identity providers.

Transactions:

- Users, via an associated account, can initiate a number of transactions, including plain transfers of GTU to another account, writing data records or registering information to the chain, or transferring other native tokens that exist on the Concordium Platform. It is Concordium's goal to add transaction types that will help solve general problems, whereas specific functionality will likely be handled using smart contracts.
- Users will receive GTU or other tokens into their account when their address has been specified as the recipient in the transaction. It will be possible to opt out for the receipt of GTU or other tokens by adding a specific smart contract.
- Users will be able to initiate encrypted transfers along with anonymous intra-user and inter-user transfers.

Delegation of stake:

- A user can initiate a delegation transaction to delegate stake to a baker (**user delegated stake**). Likewise, a user can initiate a similar transaction to un-delegate stake.
- Rewards associated with a user's delegated stake will be automatically transferred from the reward account to the relevant accounts.

The Concordium Foundation controls a set of special accounts (described below) and can perform the same transactions as a user. The Concordium Foundation is the recipient of the **platform development subsidy** that is paid from the reward account.

Bakers are created when users issue special transactions to register as a baker. In the future, a baker may be required to commit a certain percentage of its GTU stake as incentive against undesirable behavior that damages the Concordium Platform.

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 and make adjustments to increase or decrease the number of finalizers with the view of securing the network without compromising speed or efficiency. In the future, bakers will be able to opt out of being a finalizer.

Smart Contract Developers: Any user will be able to write, deploy, or use a smart contract on the Concordium Platform. We propose that 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 is valuable as it allows 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. In the future, identity providers may be incentivized on-chain for the process of creating identity objects.

Anonymity Revokers do not interact directly with the blockchain and do not need an account.

Special Accounts

There are a set of special accounts that exist within the Concordium Platform. These facilitate specific transactions, including token reward schemes, inflation and the role of the Concordium Foundation.

The **Foundation accounts** are a set of accounts that hold the GTU allocated to the Concordium Foundation in the initial token distribution as per the genesis block. The Concordium Foundation controls the key to these accounts. The GTU in the Foundation accounts legally belong to the Concordium Foundation and will be used for the exclusive purpose of facilitating the development of the Concordium Platform and the proliferation of the network.

The **reward account** holds the pool of GTU from which rewards are distributed to bakers, finalizers and users that have delegated stake. There are a number of mechanisms via which GTU are deposited into the reward account:

1. The minting of new GTU as set out below.
2. Transaction fees paid by users.
3. GTU from the initial bonus account.

The **initial bonus account** is created during the genesis block generation and contains a fixed number of tokens that are released to the reward account over time. Initially, the Concordium Foundation will control the rate at which tokens move from the initial bonus account to the reward account with the purpose of incentivizing user acquisition on the network and providing extra incentives to bakers and finalizers. The distribution schedule for the release of GTU from the initial bonus account will support the development of the Concordium Platform over a long period of time from the instantiation of the genesis block.

Transaction Fees

There is a compute and infrastructure cost associated with processing a transaction. As such, the entity that initiates a transaction must pay a fee.

Transaction fees are calculated as follows:

- There is a flat fee for each transaction. This is designed to be as low as possible.
- There is a variable cost on top of the flat fee that scales up with the amount of resources required to bake and finalize a particular transaction.
- For smart contracts, there can be an extra fee defined by the instructions in the contract to be executed.

Stability of fees

In order to help businesses and individuals understand the actual costs of processing transactions and building business models on top of the Concordium Platform, we have implemented an internal unit to help price transactions within the platform. This unit is called **ENERGY** and it is used to denominate transaction fees and other on-chain incentives.

While the price of GTU may fluctuate on the open market, the ENERGY required in fees for any specific transaction will remain the same. The exchange rate between GTU and ENERGY is called **GTU/ENERGY** and fluctuates in response to movements in the market price of GTU. For example, if the market price of GTU goes up, then in order to ensure price certainty from the user perspective, the exchange rate of GTU/ENERGY will go down. Thus, less GTU is required to cover the fixed amount of ENERGY for a specific transaction fee and the total cost of the transaction will remain consistent in real-world terms. From the perspective of the Concordium Platform, the cost of a transaction is denominated in ENERGY and the platform will automatically deduct the correct amount of GTU from a user's account according to the GTU/ENERGY exchange rate. The user must have enough GTU in their account to process a transaction. We are currently investigating the stability mechanism that will be deployed to inform fluctuations in the GTU/ENERGY exchange rate.

Minting

The process of minting new GTU is an automatic feature of the Concordium Platform whereby newly minted tokens are transferred directly to the reward account for distribution as set out above. Minting is the only source of inflation inside the Concordium Platform and, initially, the inflation rate will be controlled by the Foundation.

Minting is governed by a parameter which controls how many GTU are created per slot, where a slot is an arbitrary, fixed unit of time. This ensures that the inflation rate is linked to time and does not depend on fluctuations in the rate that blocks are created. Initially, the Concordium Foundation will control the GTU/slot parameter and monitor inflation to ensure the sustainability of the economy. In the future, we will propose solutions to the issues created by the distribution of minted GTU across shards.

Rewards

Rewards are distributed to bakers, finalizers and users that have delegated stake from the reward account on a regular basis.

Rewards for bakers

The reward distributed to a baker for creating a block is a combination of:

- A fixed percentage of the current balance of the reward account.
- A percentage of the transaction fees paid to process transactions in that block. The remaining transaction fees are deposited in the reward account.

Rewards for finalizers

For every finalized block the finalization committee is rewarded a fixed percentage of the balance of the reward account. The reward is split among the individual finalizers participating in that finalization proportional to their voting power in the finalization committee, i.e. their stake, including all stake delegated to them.

Rewards for delegators

Users that have delegated stake to a baker receive a fixed percentage of the reward for baking a block, where this percentage is determined by the Concordium Foundation. Initially, the Concordium Foundation will adjust this parameter in order to achieve the right balance of incentives between bakers and users that delegate stake. In order to prevent vote buying and promote delegation to the most reliable bakers, this rate will be the same for all users and bakers, regardless of the size of the user delegated stake. Where multiple users delegate stake to a single baker, the portion of the baker reward that is passed on to users will be divided among users based on the size of their delegated stake.

Platform Development Subsidy

The Concordium Foundation is the current recipient of the **platform development subsidy** which is equal to a proportion of the total GTU minted per slot plus a proportion of transaction fees. The Concordium Foundation receives this payment in GTU from the reward account. The Concordium Foundation will use the proceeds from the platform development subsidy to develop the Concordium Platform and incentivize user and business case adoption.

Governance

We intend governance of the Concordium Platform to eventually become decentralized. Until this is achieved, governance, development and decision making will reside with the Concordium Foundation. During this period, the Concordium Foundation will be responsible for all decisions concerning the Concordium Platform, including the parameterization of incentives, the development roadmap and timeline, and off-chain activities to promote adoption of the Concordium Platform among individuals, businesses and the community. Concordium AG, a wholly-owned subsidiary of the Concordium Foundation, is mandated to undertake the practical development of Concordium's technology along with various business development and marketing activities.

Decentralized Governance

In the future, governance of the Concordium Platform will be fully delegated to network participants and stakeholders where the voting power of stakeholders is proportionate to their stake. It is difficult to predict the best mode of decentralized governance and we believe that design parameters will evolve along with the community that grows up around the Concordium Platform.

At the stage of full decentralization, the following mechanisms are envisaged:

- On-chain voting that allows stakeholders to vote on proposals posted on the blockchain. This could involve anonymous delegation of stake to avoid vote buying.
- An on-chain mechanism for paying off-chain entities to perform work for the benefit of the platform. For example, a variety of organizations could be compensated for developing new technology or onboarding corporate users on behalf of the platform.
- An effective update mechanism for the blockchain.

As we move towards decentralization, we envisage that special procedures, keys and accounts will be created in order to facilitate practical governance, updates to the platform and payments for off-chain services after the Concordium Foundation no longer provides such functions.

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.

[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.

[DMMNT19] Dinsdale-Young, T., Magri, B., Matt, C., Nielsen J., Tschudi, D. Afgjort: A partially synchronous finality layer for blockchains. Cryptology ePrint Archive, Report 2019/504, 2019. https://eprint.iacr.org/2019/504.

[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.

[FMMO19] Fauzi P., Meiklejohn S., Mercer R., Orlandi C. (2019) Quisquis: A New design for anonymous cryptocurrencies. Advances in Cryptology – ASIACRYPT 2019. Lecture Notes in Computer Science, vol 11921. Springer, Cham, 2019. https://doi.org/10.1007/978-3-030-34578-5_23.

[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.