Universal Credit System

a next generation crypto currency

Welcome!

In this document you will find everything you need to know about the new and innovative crypto currency of the next generation - **the Universal Credit System**. It contains all the information you need to understand and trust the Universal Credit System.

This document is some kind of white paper, which contains a precise specification of the crypto currency. Furthermore, the logical processes shall be presented in an understandable manner for everyone in this document.

The Universal Credit Standard

The ***Universal Credit Standard*** is an independent digital crypto currency standard. The standard shall enable all users to participate in trade and build up wealth. The ***Universal Credit System*** is the underlying software for managing the currency (wallet) and the currency paid out in this software is called ***Universal Credit Coins (UCC)***.

However, the *Universal Credit Standard* and the principles behind it are fundamentally different compared to other cryptocurrencies. Instead of *mining* new blocks in whatever form and getting a *reward*, all participants are simply *credited* with a fixed number of Universal Credit Coins every day.

These Universal Credit Coins are not credited or granted by a central authority; the participants manage themselves based on an algorithm. The Universal Credit Coins granted are valid indefinitely and can be used for whatever by the recipient.

The payout per day follows a very simple rule:

1.000000000 UCC per user per day

With this rule it is very easy to determine the value of a thing, but also the value of a credit coin at any time. If something has a price of 200 UCC, its actually the payouts of 200 days. And it doesn't matter if today, in 3 years or even 10 years. 200 Coins are the payout of 200 days.

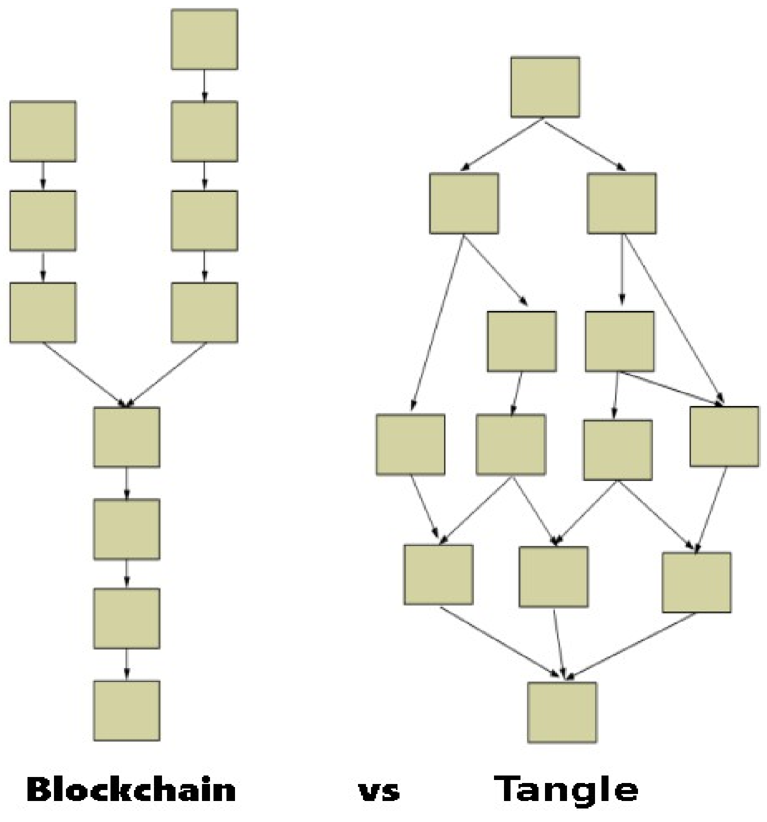
Due to the fact, that the payout / reward is linked to the time that has passed and is consistent, the Universal Credit Coins correspond to the definition of a stable coin.

The proof-of-Trust algorithm [poT]

While most cryptocurrencies use either *proof-of-work* or *proof-of-stake / delegated-proof-of-stake* algorithms to reach consensus, the technology used in the Universal Credit System is fundamentally different. The algorithm used in the UCS is called the **proof-of-Trust algorithm** and has been completely redeveloped from the scratch.

Instead of building the consensus from a monolithic and central ledger (blockchain) like Bitcoin, the architecture of the Universal Credit System is based on an **directed acyclic graph** (DAG) in which each user has his own blockchain in the sense of a self managed block grid (a so called „true distributed ledger“). Each block of a user corresponds to a transaction and the blocks are **not linked** to one another.

The structure used here can be compared to the *tangle* of the cryptocurrency IOTA. But compared to IOTA the common consensus within the network is reached in a totally different way. The use of a directed acyclic graph makes the cryptocurrency a lot different than other, blockchain-based architectures:



In a certain sense, users in the UCS provide proof of work (pow) by signing their own transactions and verifying the transactions of other users, but the consensus on an account balance is only partially determined using this proof of work. The coins paid out daily, which are determined by the certified creation date, have a far greater influence on an account balance. The certified creation date and the transactions signed by the user as well as the transactions addressed to him, together with the confirmations from other users, form the *proof-of-Trust*.

Since the participants manage themselves, no permanent online connection is required. Each user manages his own version of the blockgrid. With each action, users provide each other with proofs of trust, which mathematically proves the validity of their account balance to other participants.

Users and the certification of the creation date by the TSA (Time Stamping Authority)

Each entity, henceforth referred to as a neutral “user” (a user can be anything), is allowed to create a pair of 4096bit RSA keys. The user must immediately disclose the public key to an RFC-3161 compliant Time Stamping Authority (TSA), which acts as an independent notary and certifies the date the profile was created **within 120 seconds** after the creation date.

The TSA certifies the user that his key is available at the time of creation. This certified creation time serves as the basis for calculating the daily payment.

The certification of a new user after its creation is the only point at which a connection to the internet is required. Any other action can be done without being connected to the internet.

Transactions

A user can create a transaction that contains his address as the sender, the amount, the recipient and the current time stamp and then signs this transaction with his private key. The user then creates the transaction file, which he can send to the payee in a way of his choice. There is no need to be connected to the world wide web for this, the user can use a medium of his choice to exchange the transaction file. The recipient can check the authenticity of the transaction with the help of the public key and verify the transaction and the associated credit with the help of the proof of trust.

Strictly speaking, the transaction file is an archive that contains the sender's public key, the authentication of the TSA and all transactions previously sent and received by the sender to process the transaction. It also contains all keys collected so far, their TSA authentications and transactions that were in the sender's possession at that time. This is used to build up a common knowledge of transactions.

This common transaction knowledge is used to determine dependencies on transactions and to be able to check the associated balances. This is particularly important to determine balances, since a user may have been dependent on a transaction sent to him.

Here is an example of a transaction that depends on other transactions:

User A balance	Transaction
0	
100	+100 granted by system
200	+100 received from User B
50	-150 sent to User C

As you can see, *User A* was only able to send the 150 credit coin transaction because he had previously received 100 credit coins from *User B*. This last transaction (150 credit coins) is dependent on a previous transaction. Such dependencies can be determined and checked through the collected proof-of-trusts of the other users.

Transactions and Confirmations

If a transaction is plausible, it is always immediately valid for the sender. So if a transaction was successfully created, the sender's account balance is immediately reduced. In return, the recipient's account balance is only increased if he has collected the required number of confirmations from other users. Through synchronization and future transactions the knowledge about this transaction is spread and other users confirm / authenticate it.

A transaction requires a certain number of confirmations other users in order to be acknowledged/recognized by previously uninvolved users in the future for a transaction. The problem is particularly important when checking dependent payments, since the payment on which the sender is dependent may not have enough confirmations to be considered valid yet. But also so called s*tale blocks* can be identified that occur when someone tries to double spend.

In order for a transaction to be taken into account when calculating an account balance, it must be confirmed **by at least 2 users** so that it is as final as possible and the sender can no longer deny this transaction in the future ("double-spending" and "0 race" attacks).

The process of a changing payment status (from *pending* to *valid* to *final*) is a never ending process. So a transaction is never totally final, but a transaction becomes more and more final. At some point, there is enough confirmation that it can be considered valid.

The acknowledgments of the other users about the existence of a transaction is called **confirmation**. It depends on the number of confirmations of a transaction if it can be considered valid. A transaction is considered to be valid by default if it has at least 2 **confirmations from other users**. The transaction must be confirmed by users that were not involved in the transaction (neither as sender, nor as receiver). These users act as some kind of notary and certify / confirm / acknowledge this transaction by listing the transaction file in their index file.

The fact that the receiver of 1 transaction needs at least 2 confirmations to process it ensures the exponential spread of the transaction-knowledge.

Below you can find an overview of the number of confirmations of a transaction and the validity (as per default setting):

Confirmations	Validity
0	not valid (pending)
1	not valid (pending)
2	valid
3	valid
...	...
∞	final

The current number of *confirmations* for a transaction can be viewed in the transaction history of the *Universal Credit System*. To ensure that all transactions that have been made or received are sufficiently confirmed, it makes sense to synchronize with other users as soon as possible.

Scoring

In order to prevent a user from creating any number of additional accounts, so-called scoring of the user takes place. This means that the transactions sent, the transactions received and the user's balance are weighed against each other. The score value of a user is calculated according to the following 5 rules:

- Every day the score of all users increases by the payout
- Immediately after creating a transaction, the score is reduced by the transaction amount
- If the transaction has received enough confirmations, the sender's score increases by the original amount
- The score cannot be lower than 0
- The score cannot be higher than the users balance

The scoring thus ensures that a user cannot collect an infinite number of credits from other users and spend them again immediately. Instead, more and more credits are unlocked over time. When a user does not have enough funds to buy something, it is impossible to get around it by creating multiple other accounts and collecting the withdrawals as the user cannot send them all at once.

Here is an example <u>without</u> scoring:

- Account A has a balance of 5 UCC Account A wants to buy something for the price of 10 UCC

- Account A creates accounts B, C, D, E and waits 1 day for the first payout

- Account A has a balance of 6 UCC on the next day, Accounts B, C, D and E each have 1 UCC balance

- accounts B, C, D, E each send 1 UCC to account A and increase the balance from 6 UCC to 10 UCC

- Account A now has a balance of 10 UCC

In this example without scoring, account A now has 10 UCC and could buy the item.

Now an example <u>with</u> scoring:

- Account A has a balance of 5 UCC and a score of 5 UCC
- Account A wants to buy something for the price of 10 UCC
- Account A creates accounts B, C, D, E and waits 1 day for the first payout
- Account A has a balance of 6 UCC on the next day, Accounts B, C, D and E each have 1 UCC balance
- Accounts B, C, D, E each send 1 UCC to account A, increasing the balance from 6 UCC to 10 UCC
- Account A now has a balance of 10 UCC, but only a score of 6 UCC

A now has a balance of 10 UCC, but due to the scoring (balance is equal to the score so the score is not affected) only 6 UCC (the initial 5 UCC + 1 UCC payout on the day he waited) unlocked and can be spent. When you spend the 6 UCC, the rest of the credit will be unlocked. With scoring, a user can only spend the daily payouts without limit and must activate the credits received. In the example without scoring, Account A can spend the entire amount in one transaction, while Account A with scoring cannot spend the entire amount.

<u>It is very important that the entire amount is sent in a single transaction if you cannot trust the sender.</u> This ensures that attackers cannot draw in funds from a potentially infinite number of accounts they have created. This scoring based evaluation process will protect you and all other participants from fraudulent users.

Determination of the account balance

The balance of an account is determined by the users themselves, the users checking each other again and again.

In order to determine the account balance of a user, the resulting *creditload* for this first day is first calculated starting with the certified creation date of the key and then any transactions sent or received are subtracted or added up. The result is the account balance at the end of this day and serves as the basic account balance for the following day, to which the *creditload* is first added up again and then any outgoing or incoming transactions are subtracted or added up again.

This calculation is continued day by day until the current day, whereby the account balance must never fall below the 0-balance limit. The *Universal Credit System* is programmed in a way that a user can never spend more coins than he has balance. Conversely, transactions from accounts that according to general consensus have no funds cannot be credited as they would get no confirmations.

Community consensus

Users verify each other during a transaction or after synchronization by checking each other's proofs of trust. To make it simple:

If a user wants to send a transaction, he has to prove to the other participants, and in particular to the recipient, that there are enough credits to carry out this transaction.

He provides this proof through the collected proof-of-trusts. The validation of a transaction is done over and over again by the other participants. By calculating the account balances and checking the transactions while taking into account the confirmations to date and the score, a common consensus is achieved.

Synchronization with other users

A user can synchronize with other users independently from previously exchanged transactions to build up trust and knowledge. The user should also do this on a regular basis.

A user who synchronizes with another user helps to build up common transaction knowledge and helps spreading the transactions of other users. At the same time, he protects himself from double-spending attacks by having knowledge about other transactions that were previously unknown to him.

Common transaction knowledge can also significantly reduce the size of a transaction file, since in the best case user A no longer has to send proof-of-Trusts required for the transaction to user B because its already there.

The more a user knows about this user and the transactions that are linked to this user, the more credibly a user and his balance can be confirmed. The more users are aware of a transaction and confirm it, the more final is a transaction. A user who has made a transaction and subsequently deletes it from his history cannot prevent the rapid spread of this knowledge by other users. Furthermore, the attacking user no longer has influence on how quickly the knowledge about the transaction spreads. In the best case, the next transaction partner has already been informed of the last transaction by the synchronization or by a payment from a previous business partner / friend, so that the attacking user cannot deny it.
In addition to synchronization with UCAs/users, **sync-as-a-service models** are also conceivable in this context, in which websites act as a meeting point for users who want to synchronize and act as a notary for a number of coins.

FAQ

Why is there no central blockchain?

The architecture used in UCS is based on a decentralized block grid structure instead of a central monolithic block chain.
A central blockchain is not required to form a common consensus or to check a specific transaction or account. The proofs-of-Trust (TSA files and transaction files) are enough. As a result, a user can only confirm transactions from the trusted network in which he is located. Only networks and participants connected to this trust network are visible to the user. It's not like Bitcoin where you can see the entire transaction history of all users due to the monolithic blockchain. Of course, the trust networks that are formed will grow and connect through trade and synchronization. This creates more or less central transaction knowledge over time, but the basic concept is not central. The blockchain of a user consists of all sent and received transactions, which are managed as part of a distributed ledger concept.

Which TSAs are supported?

Only FreeTSA is currently supported as a TSA. Concepts for further TSAs are currently being discussed.
.
NOTE: It has already been discussed to extend the selection of TSAs to OpenTimestamp or to give users the opportunity to define TSAs themselves. The former would, however, mean that one would have to integrate the OpenTimestamp client, which in turn accesses the blockchain of Bitcoin, which either entails further dependency on an online service or the need to install the client including the Bitcoin client locally. The ability to define TSAs yourself would allow users to use compromised TSAs. Both solutions are not good!

What is the proof-of-Trust algorithm?

The proof-of-Trust algorithm ensures that everything is correct and users are not able to cheat. A proof of Trust consists of public key of the sender, which have been certified by the TSA (and maybe also the keys of users), including all incoming and outgoing transactions of a user and his contacts. This is enough to determine the current balance of an account and / or to determine whether a transaction concerning this user is plausible or not.

What contains the proof-of-trust?

A proof-of-trust is always made up of several files. A user's proof-of-trust consists of at least:

- public key of the user in /keys-folder

  optional: public keys of other users if transactions are dependant

- freeTSA query file (*.tsq) and response file (*.tsr) of public keys in /proofs-folder

  optional: query and response files of other public keys if transactions are dependant

- transactions of the user in /trx-folder

  optional: transactions of other users if transactions are dependant

- index-file of each user in /proofs-folder

  Contains a list of all plausible transactions and is used to determine the confirmations

With the help of these files it is possible to check a credit and related transactions at any time.

What happens if a user manually removes a transaction from the history – can the user spend the same amount again?

If you have followed all the rules: No! If a user sends a transaction to another user, the recipient will distribute that transaction in the future. If the sending user simply removes a transaction from its history, the users who already know about the transaction would still spread knowledge of this transaction in the future. How fast this happens can no longer be influenced by the sender. Synchronization with other participants significantly reduces the risk of such an attack since one participant logically cannot withhold a transaction other users are already aware of in the trust network.

Why universal credit system?

No energy-intensive calculations are required to get credit coins. Credit coins are given to all entities that have successfully created an account. It is independent of a central authority, a state or a country. It is independent of the place where it is used. Nobody can prohibit it, nobody can close it and nobody can change the consensus. On the one hand it can act as a monetary medium and on the other hand it can be used for speculation that is stimulated by the artificial shortage. The artificial lack of credits offers potential users an incentive to participate as quickly as possible and at the same time does not exclude the late participants. It is a kind of democratic currency, because everyone receives the same amount and nobody is favored or disadvantaged. There are no centralization tendencies in terms of the creation of money like in existing cryptocurrencies. Someone with a lot of money gets the same number of credits as someone with little money. It's a new start from scratch and every user starts with balance 0 and the same opportunities. It is a universal currency standard based on math and decentralized trust. It is transparent where it is necessary for traceability and to reach consensus, but at the same time, participants outside of the established trust network do not need to be informed about transactions, etc. This makes it less prone to fraud and overall much more attractive than existing cryptocurrencies.