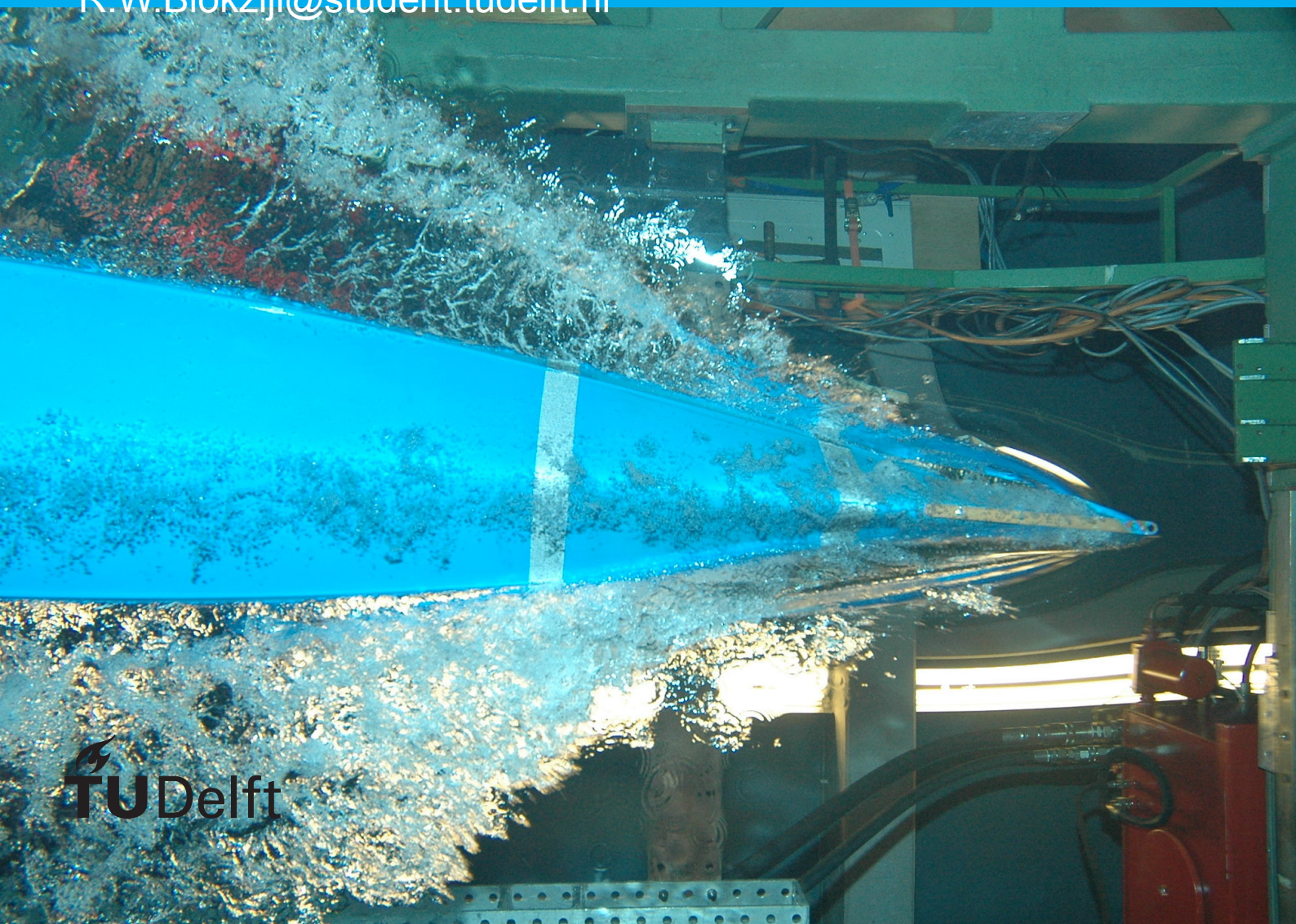
EuroToken

A Central Bank Digital Currency (CBDC) with off-line transfer capapabilities

R. W. Blokzijl

- Stablecoin
- Blockchain
- Cryptocurrencies
- TrustChain
- CBDC

R.W.Blokzijl@student.tudelft.nl



TU Delft

EuroToken

A Central Bank Digital Currency (CBDC) with off-line transfer capapabilities

by

R. W. Blokzijl

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on TODO.

Student number:	4269519	
Project duration:	November 11, 2020 – TODO	
Thesis committee:	Dr.ir. J.A. Pouwelse,	TU Delft, supervisor
	Member 1,	TU Delft

This thesis is confidential and cannot be made public until TODO.

TU Delft

Preface

TODO: Add preface

R. W. Blokzijl
Delft, TODO

Contents

1 Introduction

Since the Bitcoin [TODO cite] white paper was published in 2008, the world has been speculating on how decentralized ledger technologies could be used to restructure the financial infrastructure of the world to redistribute power towards the masses and away from large opaque organisations. 9 years later, Facebook announced a private currency controlled by a group of corporations [TODO cite]. 3 years after that the Chinese government announced that they had reached 92,771 transactions per second with their new Central Bank Digital Currency (CBDC) [TODO cite]. And the Eurosystem is set to make a decision on whether to start a digital euro project in mid 2021.

The direction of crypto currencies is no longer determined by eccentric visionaries imagining a financial system that gave power back to the people and makes money open and programmable by anyone. Governments and large corporations are now competing to create the worlds main digital coin. The winner will be left controlling and overseeing all the worlds transactions, either for profit, or their national interests.

Currently very few decentralized currencies are in a position to challenge the upcoming central coins. The most well known crypto currencies, including Bitcoin and Ethereum, lack the price stability necessary to be a reliable store of value. There have been attempts to create fully decentralized stablecoins, but none have been proven to work in practice on a large scale just yet.

If fully decentralized currencies don't come up with a solution to scalability and stability soon. The future of the financial system might come down to a competition between the large governments of the world and the private sector.

Whether and how these parties succeed will have large implications for the future of financial markets of the world, and might determine the level of freedom of the societies of the future. Where China and Facebook are making rapid progress, the Eurozone is still deliberating while they might have a vital role in including the democratic process in the running of the future financial markets.

This thesis aims to provide a design for a digital euro that utilizes mechanisms used by todays stablecoin in order to create a digital euro analog called EuroToken. EuroToken is a design for a scalable system that allows transfer between parties in a scalable and peer to peer way, while maintaining price stability through maintaining collateral euros in a bank account.

We show how the EuroToken system can be used to create a scalable CBDC as well as serve as a private money alternative to current banks and provide all the benefits of programmable money with the price stability of the euro.

2

Problem description

3

Design

Any CBDC that aims to replace public money while being able to operate at the scale of the euro system needs to conform to a number of requirements. Such a system needs to be scalable, privacy aware, allow peer to peer transactions off-line. In needs to be price stable, exchangeable for euros, and most importantly needs to be secure and cheating resistant. In this chapter we explain how a distributed personal blockchain provides a good basis for a scalable, private, and off-line friendly transaction system. We then explain how we position the system in relation to the euro, how the price remains stable, and how the system mimics the properties of cash. We then go in to the details of how the system is secured, and how we prevent double spending while still remaining scalable and allowing off-line transactions. Finally, we explain how the system could be expanded upon by legal frameworks that would provide different risk vs privacy trade-offs and how they could be enforced in the system.

3.1. Distributed accounting and networking

The possibilities and limitations of any virtual currency are dependent on its system of accounting. In order to conform to the off-line, scalability and transparency requirements a system of distributed accounting is chosen. As the fundamental building block for the EuroToken system we use a Hyper-Sharded personal blockchain that keeps track of every users transaction history on their own edge device. By storing all information required for transacting at the physical point of transaction, we create the possibility of off-line transaction between users, without any link to the outside world.

3.1.1. TrustChain

As a technology for this “personal blockchain”, we use the TrustChain system. As illustrated in figure 3.1 every users personal blockchain is structured as a chronological, one-dimentional string of “blocks”. Every block will include a cryptographically secure hash identifying what block preceded it. Because of the trapdoor effect of the hash, this has the effect that any block will uniquely identify all blocks that come before it. This allows anyone to verify the validity of entire history of another user, given the last block in this history.

Every block can contain a “declaration’ by the user, or a reference to the declaration of another party. These declarations are digitally signed by the declaring party and form the base of any transaction. To do a transaction the sending user (Alice) will create a new block with a declaration stating”I transfer 1 EuroToken to Bob“. In order to pre-

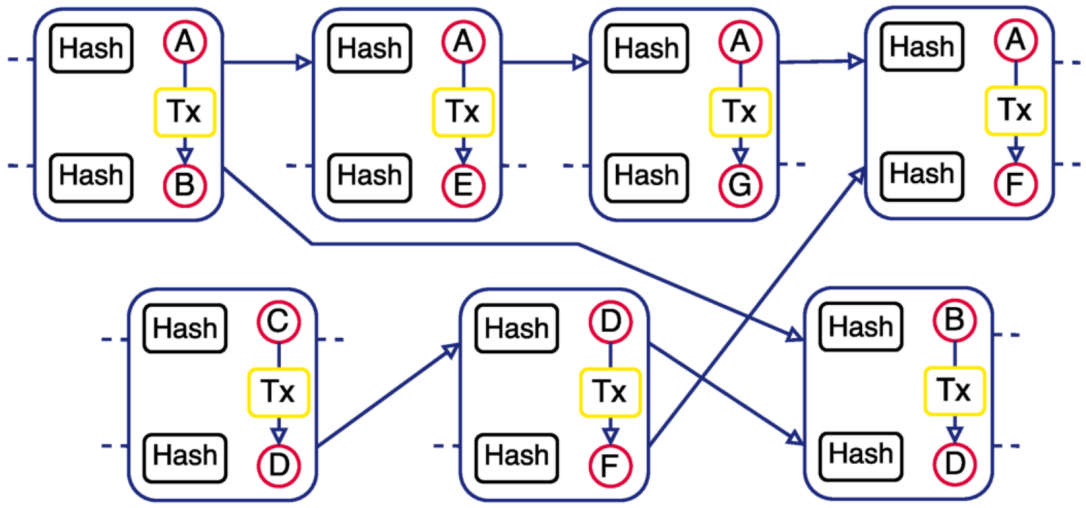


Figure 3.1: TrustChain (Vos and Pouwelse 2018)

vent fraudulent transactions every transaction is registered both Alice's chain as well as Bob's.

When Bob receives the block from Alice, he can accept it by creating a block in his own chain and returning it Alice. Before Bob accepts the block, he first validates the history of Alice by requesting enough of her chain make sure that Alice doesn't validate any of the network rules that would invalidate Bob's receiving of the money. Once Bob is satisfied with the correctness of Alice's chain he incorporates a new block declaring the acceptance of Alice's transaction. This block includes the hash of Alice's block, thus entangling the chains of Alice and Bob together. Bob now has a signed proof by Alice that the transaction happened. He can use this to prove the transaction happened at any point in the future.

3.1.2. IPv8

The TrustChain implementation comes as a part of the IPv8 distributed networking suite. While the EuroToken system design is independent of the underlying communication technology, IPv8 gives us a number of features that are a useful addition to our design.

Peer to Peer overlay networking. Firstly it provides a mechanism to discover the network location of users based on the same public key that is used to identify a TrustChain user. This allows us to almost completely abstract away from locating users using IP addresses and ports. As a result we only have to worry about maintaining a users public key to identify and communicate with them across time. IPv8 does not only abstract away from IP addresses, bus also from the IP network completely. Namely, it provides communication over Bluetooth without the need for any internet connection. This becomes very useful for demonstrating the off-line capabilities of the EuroToken system.

The final aspect of IPv8 that is useful is the fact that it has an implementation in Kotlin. This allows us to create a mobile app to run allow users to maintain their wallet on a portable device and interact with the system in a modern way.

3.2. Gateways: Euro to EuroToken exchange

The viability of any currency as a store of value over a given time frame is dependent on the stability of its price over that time frame. This is an issue that has plagued de-

centralised crypto currencies from the very beginning. The hope is that the currency will stabilise itself when it reaches a critical adoption level. However even currencies like the euro and US dollar don't remain stable without periodic interventions of their respective central banks.

The euro has long served as the core of the financial infrastructure of the European economy. It has essentially done this using two consumer facing versions of money: the euro as a publicly accepted, physical item of value (the public euro), and the euro as a digital, privately managed, unit of account (the private euro). These public, and private types of money serve citizens in different ways. The public euro is the most stable store of value since its guaranteed by the central bank, it also has the advantaged of requiring no internet connection to use. While the private euro has digital advantages in usability and security, but derive their value from the "reliability" of private banks, and are only insured by governments up to 100.000 euros [CITE]. With the declining usage of public money in favor of digital money, the need for a new type of euro to fill the gap of public money is getting stronger.

For these reasons we present the EuroToken system as a 3rd type of money. Instead of reinventing the wheel of "stability" we connect the EuroToken system directly to the euro system, while providing extra features on top of the current euro system.

In order to properly connect EuroTokens to the euro system, an easy and value-transparent method of exchange is required. Just like private and public euros are exchangeable though local banks, a mechanism is needed to exchange between euros and EuroTokens at a 1:1 ratio. To do this, we implement a "gateway" between the private euro system and the digital euro. This gateway implements the EuroToken protocol on the one hand, and interfaces with banks on the other.

In our current design, the gateways are designed to be run by public parties associated with the central bank. Any detailed speculation on the best way to connect such a system to the established euro is best left to economists. However, we envision a possible future where multiple gateways are run by existing private money institutions who perform the heavy lifting of day to day exchange. In such a system private banks would act as an accounting system for the EuroToken exchange without being allowed to leverage their EuroToken position. The Central bank would allow these private institutions to conform to reserve requirements in the form of EuroToken holdings rather than only cash. By not allowing private banks to mint new EuroTokens, but only exchange them, the central bank can control the amount of EuroToken in circulation in a similar way to current public money. This would insulate the EuroToken from the impact of a failing euro or bank, in the same way as physical public money is currently insulated from such failing.

This way of connecting to the euro could allow for a smooth transition to a digital form of public money, while the established and regulated financial institutions are still positioned properly in a place where financial services can be provided.

3.3. Transaction finality and Double-spending

In order to remain a viable store of value, a currency needs to provide protection against any non-sanctioned creation of that currency. If a network allows its users to "create" new money in any significant way, the value of the coin will drop as the supply increases, thus undermining one of the most fundamental function of the currency. The structure of the blockchain provides an immutable and signed history of any transactions, thus enabling users to prove that the funds they are attempting to send actually exist. However the blockchain does not inherently allow users to prove that they have not spent,

and will not spend, the same balance again.

In this section we explain how the network prevents unsanctioned creation of currency.

3.3.1. The double spending problem

In order to spend their money twice, a user has to create 2 blocks that are positioned in the same place in their blockchain. This is what is called a “double-spend attack”. This attack is only detectable if both of the conflicting blocks are found. Since we have opted for a distributed blockchain this detection becomes a non-trivial problem to solve. The transactions of 2 conflicting blocks might be re-spent many times by the time anyone sees the 2 conflicting blocks and notices that a double spend happened.

Bitcoin and similar currencies solve this problem using a global blockchain that everyone has access to. This allows users to check whether a given balance has already been spent by inspecting the global database of transactions. However, the global knowledge of the Bitcoin chain is inherently un-scalable. Additionally, the details of the Proof of Work method of block generation leaves a certain measure of uncertainty with regards to the “finality” of any transaction in the newest blocks. This often requires users to wait up to an hour to be sufficiently confident their transaction really happened.

A solution to this problem in a network with distributed blockchains, starts with the realisation that the issue of detecting double-spending can be reduced to the issue of detecting “chain forking” in our network. The usage of the blockchain allows us to make sure that all transactions are ordered and consistent, this means that double-spend needs to be in 2 separate versions of that history. Thus requiring 2 blocks that refer back to the same historic block. This is a fork in the chain. We cannot “prevent” a user from creating 2 conflicting blocks in their chain as their chain is stored on their own device. But we can make sure that the rest of the network only accepts one of the 2 blocks, thus only accepting 1 “spending” of the balance. This choice between 2 conflicting blocks needs to be consistent so anyone in the network is working with the “same history”. Additionally, forks need to be detected and resolved before the balance is spent again by any of the 2 receiving parties. This way a double-spend will not propagate into the network and is limited to the users involved in the 2 transactions. To resolve the conflict between blocks we define the concept of “transaction finality”. For a transaction to be final, it needs to be “validated” and “stored” in the network, while any conflicting transaction will be rejected by the network. Transaction finality the guarantee that a merchant needs before they can send their goods to a paying customer.

The transaction finality problem in our network has several possible solutions. In (Brouwer 2020) Brouwer presents a method of distributing blocks to a randomly and fairly selected list of witnesses that would probabilistically detect any conflicting block before the receiver would accept them. In (Guerraoui et al. 2019) Guerraoui et. al present a more theoretical method of block broadcast. These might be good candidates for future research. However since these solutions are inherently probabilistic, there is no hard guarantee that any double-spend will be detected in time.

3.3.2. Balance vs spendable balance

Currently lacking a good exact and distributed solution, we choose to utilize a decentralized network of trusted validators. These validators maintain the last transaction of users that register with them. Any user who receives money, can verify the non-existence of a conflicting block with the associated validator of the sender.

In the rest of this section, we define the concepts of “spendable balance” and specify

the information requirements for marking a transaction as finalised.

In order for Alice verify if Bob is able to send her the money he is sending, she needs to know that Bob has sufficient funds. For this reason a rolling a balance across all transactions could be maintained across all blocks. Where the balance B for a given block with sequence i (B_i) is:

$$B_i = B_{i-1} + C_i$$

.

Where C_i is the change in balance for the block with sequence number i. This is negative when sending money. However the balance of a user does not take into account the concept of transaction finality. So instead we maintain the total "spendable balance" instead.

3.3.3. Finality statements

Before Alice can add the output of a block she received from Bob to her "spendable balance", the transaction from Bob first has to be finalised. To achieve this a validation is performed with Bob's associated validator. This is done by sending the validator a finality proposal.

The finality proposal block includes notes a list of hashes that point to transactions from Bob. Together with this block for the validator to sign, Alice will send all of Bobs blocks from the last transaction to validate to the last block the validator knows about. The way for Alice to determine what information this is, is explained in the section on checkpointing later in this chapter. In addition to Bob's blocks, she will also send her "accepting blocks" that include the transaction in her chain. This is to make sure she can only claim a transaction from Bob once. Bob's validator will then verify:

1. That there are no other transactions that conflict with the one to Alice.
2. That there are no other "accpeting blocks" already linked to this transaction.
3. That Bob's chain is valid up to the last transaction to verify.

If this is the case it will sign the proposal. If a later transaction from Bob is received that marks a fork in his chain, the fork from Alice becomes the only accepted fork, and the other one is rejected. Using this finality statements as proof of this, Alice is now allowed to spend the output of the transaction.

In the case that a different fork from Bob has arrived at the validator first, the fork where Alice receives money is rejected. Since Alice has already accepted the transaction in her chain and may have built other transactions after it (though not spent the output), she could be requested to submit a new finality proposal without this block. Since Alice is not permitted to spend the funds from Bob until it has been finalised this is the point where double spending is handled.

Note that the specific handling of this event might not involve the forfeiture of a transaction. We discuss this further in the section on off-line payments and conflict resolution.

3.3.4. Verification

For a block to be considered valid:

1. All standard TrustChain invariants are maintained.
2. All blocks preceding it are verified to be valid
3. The total spent amount is to be less than the spendable balance.

For a transaction of a receiving block to be considered final:

1. A checkpoint from from the validator of the sender has to be exist in the chain of the user AFTER the transaction.

By introducing checkpoints, the required information at the point of transactions is reduced. When Alice and Bob set transact between them, Alice can determine the validity of Bob's transaction by inspecting only Bob's chain, down do his last checkpoint. However, Alice must also request all Bob's information down to the last Full checkpoint, in order to

3.3.5. Spendable balance

Once a transaction if finalised, "spendable balance" of Alice can be calculated. The spendable balance changes at two events, the finalisation of an earlier receiving transaction and when Alice spends her money. As such the spendable balance SB_i for a given block with sequence number i is:

$$SB_i = SB_{i-1} + F_i - S_i$$

Where S_i is the total amount spent in the block with sequence number i, F_i is the total amount finalised in the block with sequence number i.

3.3.6. Conclusion

In the future we envision the system to take one of three routes regarding transaction finality. First, system could be built on a future breakthrough in distributed transaction finality. Second the system could be built on a probabilistic but bounded transaction finality, where the rare double-spend is eventually detected and settled through the legal system. Or third, like in our solution, the system is build on trusted nodes that verify transactions for user. Like the gateways, these validators could be run by regulated financial institutions. Such a system would most resemble the current financial system, with the added benefits of off-line transactions, programmable money, a standardised system of accounting, instantaneous international transactions, etc.

3.4. Checkpointing

Because of transaction finality, when Alice receives the transaction from Bob, she can rely on the finality statements, rather than having to validate the chain of everyone he received money from. This reduces the validation load to only Bob's chain. However this still has some issues. First, Bob's chain will grow larger over time, thus slowly increasing the validation load. Second, all this information needs to be stored by Alice until it can be delivered to Bob's validator.

The way this problem has been solved in traditional blockchain systems is through the global blockchain and limited transactions per second. By having only miners or stakers being required to maintain the whole blockchain, only a few machines have to be able to know the entire chain and store all that data. But this is still inherently unscalable.

A second issue is one of privacy, when Bob has to send Alice all of his chain for verification, Alice can derive much from this information. Though we would like to see methods of privatization added to perhaps conceal transfered amounts, we still need a way to minimize the information leakage to 3rd parties.

To solve this issue of validation scalability, we define a form of checkpointing. We periodically create a checkpoint block in a users chain that , that includes a summary of the entire chain before it. This information is:

1. The total “spendable balance” at that point in the chain
2. The public key of the validator who is responsible for this wallet.
3. A statement that the validator has received all blocks before this point

Alice now knows the blocks that are already stored by the validator. When Alice is receiving money from Bob, she only requires Bob’s blocks down to the his last checkpoint.

3.5. Deferred validation, conflict resolution and off-line transactions

The EuroToken system has the intentional distinction between transactions and their finalisation. Because of this, the first step of transactions only require a direct connection between users. In theory, this allows to transact off-line, if they’re willing to risk that a conflicting block already exists in the validator. Of course, in this case, the transfer of funds depends on the trustworthiness of the sending party.

In this section we discuss a few ways of interacting with the system that allows for different risk exposure to the parties.

3.5.1. On-line transactions

When users are connected to the internet, a real life interaction can easily combine the finalisation step with the transaction, only transferring goods or services once the transaction is finalised. We envision this as the default way for users to interact, especially for large transactions, and transactions with strangers, since this reduces the risk to either party to zero.

3.5.2. Off-line transactions

Since money only becomes spendable after finalisation, the receiving user is the one that will lose funds when a double spend happens. To lower the risk and damage of this, certain systems might be put in place. For this, we build on the fact that transactions are always signed by both parties. This makes sure that a proof of double-spending always exists, and is obtained no later than the finalisation attempt.

A way to ensure a user that they will receive the funds is by allowing senders to register their identity with their validator. The validator would sign a statement that the identity of the sender is known and that they will take legal action in the event of a double spend. This then optionally allows the validator to accept the risk of double spending. In the case of a double spend the validator would sign a special statement with the receiver, that invalidates the double-spent transaction, but transfers and finalises the funds from the validator instead. The validator will then pursue legal action against the sender for fraud.

In the meantime the validator could block the sender to perform online transactions and checkpoints until they first settle the double spent funds. The details of what is both technically and legally possible here is a good subject for future research.

3.6. Regulation of validators

One could argue that system hasn't solved the issues of transaction finality and double-spending, and that we only defer the problem to a different point. It is entirely conceivable that trusted validators could cheat by allowing certain wallets to double-spend. To add to this, using the checkpoint functionality a validator can specify a higher spendable balance than is actually logged in the chain.

However, when comparing our system to the current way private institutions are regulated, the blockchain structure of transaction can provide a powerful method of maintaining the integrity of the institutions. While we cannot prevent fraud at the institutional level, we do provide an option for detection to allow for regulation.

In order for a regulator to check that a validator has done their job with integrity, they need to be sure of 2 things:

1. That all "statements" have been made consistently with the rules of the network.
2. That no other "statements" have been hidden from the regulator.

"Statements" in this context, describe anything that the rest of the network puts their trust in. These are:

1. Finality statements
2. Checkpoints

Both of these statements are created in the form of "accepting blocks" and are stored by users and their validators with an associated hash. We now propose a two round system for validating all transactions within a given time period. In round one, we validate that all statements have been made correctly and publicly store the hashes. In the second round, once we have the hashes of all statements available, we validate that all statements from other validators exist.

In the first round all information in the database of the validator is processed for consistency. Since all statements by the validator are made in the form of blocks in their personal blockchain, they have an explicit order. The blocks of the validator, together with the blocks of all the users the validator is responsible for, are processed in the same way as the validator was responsible for processing them. This step in the process ensures that all statements are made correctly.

In the second round, we ensure that there is no statement withheld by the validator. This is done by publicly publishing a signed list of the hashes of all statements made by the validator. This allows regulators to cross-check that all inter-validator statements have been reviewed by a validator. To make this step more efficient, we propose that when checking a validators consistency, regulators generate a list of statements for each distinct validator to increase the efficiency of distributing these hashes to relevant parties.

A possibility also exists to allow the public access to these records to ensure the integrity of their institutions.

4

Implementation

The implementation of the stablecoin system consists of 2 main elements: the wallet Android app, and the gateway REST API. A web front end for the rest API has also been created.

The wallet demonstrates the ability of TrustChain to handle the transfer of the EuroTokens peer to peer without a central entity.

The Gateway demonstrates how a bridge can be created between the traditional euro system and a blockchain based analog.

4.1. Gateway (Central Bank API)

The only way tokens are created is when a central bank creates them. In our implementation this only happens when a user has transfered an equal amount of euro into the central bank account.

The gateway is responsible for the exchange of euro for tokens and vice versa. This involves taking payments in both tokens and euros, and payments in both currencies. This means the gateway needs to interface with the bank to allow a user to make payments in euro when creating EuroTokens, as well as a mechanism for paying out euro to the user when they trade in EuroTokens. On the other side of the gate the system needs to be able to create/send, and destroy/receive tokens on TrustChain.

The gateway aims to automate and link all of this interaction, so EuroTokens can be bought and sold at any time by anyone.

4.1.1. EuroToken Creation

When a user wants to convert a euro to a EuroToken, a creation event is initiated with the gateway API. The user sends their TrustChain wallet address and amount to convert with the request.

The API will then create a payment request with the associated bank for the specified amount, and store the information in its database. The payment link is returned to the user.

When the user has paid the request, a transaction for the EuroTokens will be created using TrustChain. The gateway will create a proposal half-block which will be sent to the user, who will create an accepting half-block registering the transaction on both chains.

The user is now free to send the EuroTokens to anyone they like, requiring only a TrustChain transaction.

4.1.2. EuroToken Destruction

When a user wants to trade in a EuroToken for a euro the process happens in reverse. For the demo the user does a request to the API with the desired amount, their TrustChain address and an IBAN.

The system creates a TrustChain transaction for negative the amount. This transaction is sent for the user to accept.

When the user has then signed the accepting half-block. The system will pay out the amount to the specified IBAN.

4.1.3. Frontend

To aid everyday users in the purchase and sale of EuroTokens a web frontend is created where the user can interact with the API. It demonstrates the ease of use of the system.

[Screenshots]

4.1.4. Implementation considerations

The design specified a general architecture for the EuroToken system. However in order to make an implementation possible within the constraints of the project some implementation trade-offs have been made.

Bank support

The EuroToken is designed to work with any bank account for euro collateral. However in this implementation we only implemented the API for ABN AMRO. Adding other banks is a simple as implementing the `Bank` class.

Euro Payment Initiation

The design specifies a requirement of automatic euro payout on EuroToken destruction. In order to automate this, most banks (including ABN) requires registration and use of the PSD2 payment initiation API. This API requires a Payment Initiation Service Provider (PISP) licence, which in turn requires a banking licence. Since both of these licences require you to be a fully functioning bank, the payment initiation part of the ABN API has not been implemented and is done manually in the field trial.

TrustChain

Since the main implementation if the TrustChain software (Tribler, n.d.) is build on python so is the gateway API. The server is provided as a single docker container that also provides the frontend.

4.2. Android Wallet

In order to use the EuroToken system on a daily basis, users need a way to send and receive the token. Because the added value of the system is its distributed nature, a way to send and receive the asset in a convenient and peer to peer way is needed. The TrustChain team has recently come out with an

Android super-app[TODO, cite] that showcases some of the IPv8 (Tribler, n.d.) and TrustChain[TODO CITE] capabilities. This app provides the perfect platform to showcase the EuroToken capabilities.

4.2.1. PeerChat Extension

The super-app already includes a number of applications, including PeerChat. A chat application that uses IPv8s peer to peer capabilities to communicate. In order to show

that the EuroToken can be used in a modern context, the PeerChat app has been expanded to include the capacity to send money attached to a message.

To send money, the user simply selects the option to send money, and is taken to a screen where a transaction can be created. The message is then sent to the receiver who within a few moments sees the transaction appear as a message in their shared chat. The transaction amount is also added to their balance.

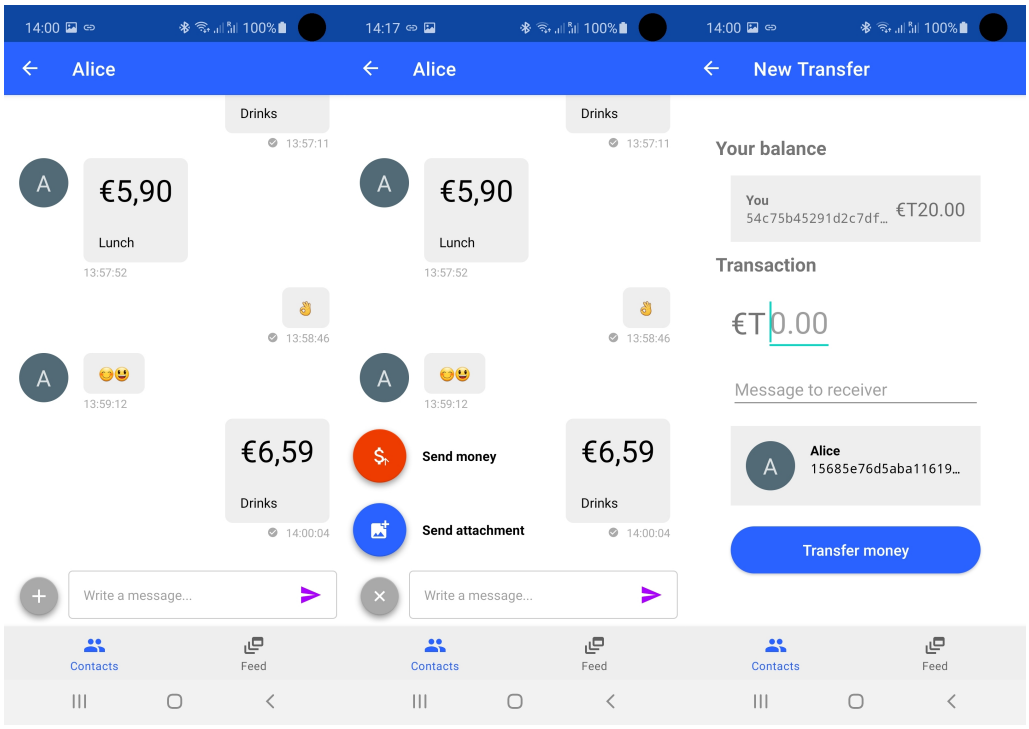


Figure 4.1: Attach money in PeerChat

The capacity to send transactions as shown in Figure 4.1 is not tied to PeerChat messaging. When money is sent, a transaction is created and transferred to the receiver using the TrustChain main community. The transaction hash is then sent as part of the PeerChat message. The receiver then fetches the transaction it received earlier via TrustChain.

- [TODO diagram of TrustChain and PeerChat interaction]

This implementation demonstrates the simple way in which EuroToken allows monetary transactions to be seamlessly and programmatically inserted into any application.

4.2.2. EuroToken app

The PeerChat app is one specific use case. In reality different applications would simultaneously use the EuroToken system. This would leave the user with a splintered record of their financial life.

In order to solve this, a EuroToken accounting app has been added to the super-app. The purpose of this is to show that the systems data can be reorganized in whatever way. The EuroToken app shows a history of all transactions, and provides another interface to the gateway.

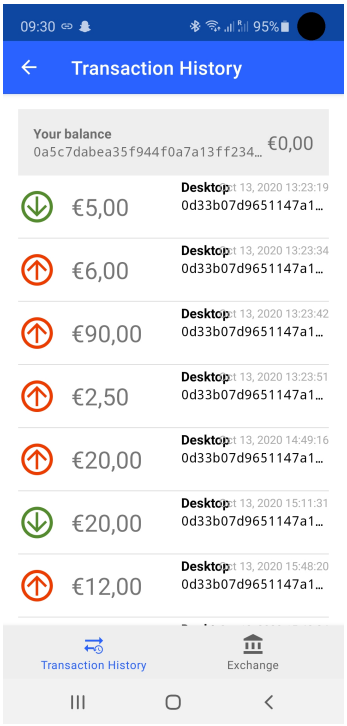


Figure 4.2: EuroToken Transaction History

4.2.3. EuroToken transactions in depth

- Validation (TODO)
  - Creation/destruction:
    - Trust Central Bank only
  - Transactions (prevent double spend)
    - Trusted users (based on identity later)
    - Validated by bank
    - Bami double-spend protection

4.2.4. EuroToken Settings

In addition to providing convenient services to the user, the EuroToken app has some configuration options to give the user control over their role in the EuroToken network.

Trusted Minters - Since the network has a central component that regulates the creation and destruction of the tokens, a demo requires a running server. Since there is no party to maintain such a server indefinitely right now, an option is added to allow the user to specify public keys of trusted “central banks”. If this option is enabled, the wallet in the super-app does only accepts blocks signed by the configured public keys.

- [TODO: image of minter config]

Trusted validators - Validation of transactions and prevention of double spending is unsolved in TrustChain but is an important part of any currency. Solving this problem in general is being worked on [TODO CITE bami] and is out of scope for this project. However the issue of transaction finality being important to a EuroToken system, a way to prevent double spending has been added. A transaction is not considered final until a trusted entity has signed a block in the senders chain that comes after the send block.

This means that the trusted entity is responsible for the validation of the block and its dependencies. These validators can be configured in the app in order to make de demo repeatable.

- [TODO: image of validator config]

5

Experiments and performance analisys

6

Discussion

6.1. System dangers

6.1.1. Under-collateralization

Causes:

- By central bank printing without collateral
- Licenced gateway banks going bust, taking collateral with them

Effects:

Future bank runs could leave some token holders without their collateral, this makes token holders less confident in tokens. This would lower their value, but the direct exchange peg maintains the price. This hides the problem while undermining trust in the value of the tokens.

Solution:

- Don't print without collateral.
- Short term:
  - Keep collateral liquid at all times (also stops inflation)
- long term:
  - see system future

6.2. System future

- euros are deleted by banks on euro2token exchange, and created on token2euro exchange.
- Banks don't manange the collateral, only the CBDC exchange.
- Banks get a place in trust instead of investment.

7

Conclusion

Related Work

Brouwer, Jetse. 2020. “Consensus-Less Security.” http://resolver.tudelft.nl/uuid:d3d56dd8-60ee-47f7-b23a-cdc6c2650e14.

Guerraoui, Rachid, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, Dragos-Adrian Seredinschi, and Yann Vonlanthen. 2019. “Scalable Byzantine Reliable Broadcast (Extended Version).” doi:10.4230/LIPIcs.DISC.2019.22.

Tribler. n.d. “Tribler/Py-Ipv8: Python Implementation of the Ipv8 Layer.” Accessed: June 13, 2020. https://github.com/Tribler/py-ipv8.

Vos, Martijn de, and Johan Pouwelse. 2018. “Real-Time Money Routing by Trusting Strangers with Your Funds.” https://repository.tudelft.nl/islandora/object/uuid:c51ac99d-3013-44b3-8ddd-fbd951a2454a.

Bibliography

[1] Jetse Brouwer. Consensus-less security, 2020. URL http://resolver.tudelft.nl/uuid:d3d56dd8-60ee-47f7-b23a-cdc6c2650e14.

[2] Martijn de Vos and Johan Pouwelse. Real-time money routing by trusting strangers with your funds. https://repository.tudelft.nl/islandora/object/uuid:c51ac99d-3013-44b3-8ddd-fbd951a2454a, 2018.

[3] Martijn de Vos, Can Umut Ileri, and Johan Pouwelse. Xchange: A blockchain-based mechanism for generic asset trading in resource-constrained environments, 2020.

[4] Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, Dragos-Adrian Seredinschi, and Yann Vonlanthen. Scalable byzantine reliable broadcast (extended version). 2019. doi: 10.4230/LIPIcs.DISC.2019.22.

[5] Tribler. Tribler/py-ipv8: Python implementation of the ipv8 layer. Accessed: June 13, 2020. URL https://github.com/Tribler/py-ipv8.