Redeemeum

decentralised commerce protocol

Connect onchain value with real world products and customer data.

Redeemeum is a modular Web3 protocol that connects onchain and offchain commerce. Use Redeemeum smart vouchers to build applications that connect decentralised finance and loyalty with real world products and customer data.

Authors: Justin Banon, Gregor Boroša

Whitepaper Version: v1.4, June 2019

Abstract

We present Redeemeum decentralised commerce protocol. Redeemeum is a modular Web3 protocol for building decentralised commerce applications using smart vouchers that connect decentralised finance and loyalty with real world products and customer data. We introduce the concept of a smart voucher as an interoperable, programmable and composable financial primitive which represents the right to claim goods or services. The vision of Redeemeum is to support the growth of an ecosystem of decentralised commerce applications, by solving two basic problems. First, the protocol solves the problem of redeeming onchain value for real world products by enabling the issue, trade and redemption of smart voucher tokens using blockchain technology, smart contracts, keeper marketplaces and standardized voucher specifications and interfaces. Second, the protocol solves the problem of smart contracts accessing customer and usage data in a decentralised and self-sovereign way. Each smart voucher instance constitutes a secure, data channel for consumers to share product and usage data or receive advertising or promotional messaging. Whilst the protocol provides incentives to share data, sharing is entirely voluntary and consumers retain sovereignty over their data at all times. Redeemeum employs a work token, embedded within a multi-sided business model comprising a nearly free side to drive adoption, a subsidised side to incentivize target behaviours such as data sharing, and a paid side providing businesses with access to data, marketing services and applications. Redeemeum enables the scalable coordination of commerce, whilst reducing transaction costs associated with triangulation, transfer and trust on a market-wide basis. In addition, the protocol fosters an open, competitive market via a decentralized public utility that protects participants from market power abuses. The resulting decentralised commerce ecosystem has the potential to disrupt centralised intermediaries and drive mass adoption of crypto within consumer commerce.

Contents

1 Background

The vision of a decentralized web, where data is liberated from monopolies and markets are freed from centralised control, is being realised in the form of the emerging decentralized stack.

Within the decentralised finance space, many components are already available, including cryptocurrencies for payments, and decentralised lending protocols for credit. Protocols such as Dharma, Augur, 0x and Maker represent new interoperable, composable and programmable financial primitives, upon which an ecosystem of decentralised financial applications is being built. For example, BlitzPredict is a trust-minimized sports betting DApp that combines Augur to create markets, 0x to execute trades, and Maker to denominate value. However, outside of decentralised finance, a wider decentralised commerce ecosystem has been slower to emerge.

Within the offchain commerce ecosystem, competition drives consumer commerce applications to integrate multiple value types including: *finance, loyalty products and services* and *consumer data.* For example, travel co-brand credit cards combine payments with credit finance for purchasing goods and services. In addition, co-brands operate loyalty programs that issue points and rewards, and capture valuable customer data. This combination of value types enables credit card issuers to differentiate otherwise commoditised products without competing directly on fees, whilst driving brand loyalty and customer engagement.

> *The vision of Redeemeum is to support the growth of an ecosystem of decentralised commerce applications that combine decentralised finance and loyalty with real world products and rich customer data.*

Problem statement

The primary problem that Redeemeum solves is that o*nchain and offchain commerce are disconnected.* This breaks down into three basic problems preventing the development of this decentralised commerce ecosystem.

1.1 Problem 1: redeeming onchain value for real world assets

S*mart contracts are unable to redeem onchain forms of value such as decentralised finance and loyalty for real world goods and services.*

Due to the consensus mechanisms that control them, smart contracts cannot directly interact with external systems in order to exchange onchain value with real world assets. As a consequence decentralized applications are either siloed from offchain commerce or they must rely on trusted intermediaries in order to connect with real world goods and services.

1.2 Problem 2: connecting offchain customer and usage data

Smart contracts are unable to gather rich customer and usage data on real world products and services in a decentralised and self-sovereign way.

Whilst oracles are designed to connect smart contracts to external data sources, connecting decentralised systems to unreliable centralised data sources simply replicates the problem that they were designed to solve. Centralised systems collect huge quantities of customer and usage data, but they suffer from a number of challenges. *Firstly,* data is hoarded within insecure centralised database 'honeypots' which are frequently hacked. *Secondly,* data can be easily tampered with, censored or withheld from its rightful owners. *Thirdly,* there is no incentive for consumers to share data, instead they are separated from the value that they create.

1.3 Problem 3: building decentralised commerce applications

There is no easy way to build decentralised commerce applications that integrate multiple components of decentralised finance and loyalty with real world products and customer data.

This has several consequences. *Firstly*, single component decentralised payments applications compete and lose against existing payments that are just 'good enough'. *Secondly*, decentralised finance has no way to differentiate other than by competing on fees. *Thirdly*, blockchain loyalty programs are mostly indistinguishable from their traditional counterparts.

2 Existing solutions

2.1 Centralised voucher systems

Centralised systems address the problem of redeeming value for real world assets by enabling stored value (such as electronic money, credits or loyalty points) to be exchanged for a redeemable promise in the form of a *voucher,* which can be redeemed at a later date for physical or digital goods and services. Such redeemable promises can take various forms including *vouchers, gift certificates* and *cards, discount coupons, membership cards and loyalty points;* or simply a proof of purchase which is used to collect goods or services. We borrow the following generic definition of a *voucher*:

> “A *‘voucher’* represents a redeemable promise *or “a digital representation of the right to claim services or goods” Fujimura*[1]

Electronic vouchers require a centralised intermediary in order to operate. Such voucher scheme operators are typically firms in competition. Rather than adopt a common voucher standard, rival firms are incentivized to build siloed, proprietary platforms and issue incompatible vouchers in order to lock-out competitors and lock-in consumers and producers. As a consequence, many incompatible, proprietary voucher systems fragment the market. This fragmentation multiplies implementation costs and reduces adoption:

> *“If a different issuing or collecting system to handle such points or coupons must be developed for each individual application, the implementation cost will be excessive, inhibiting the use of such mechanisms.”*[3] *Fujimura*

Centralised voucher schemes reduce transaction costs, albeit suboptimally, across the three dimensions of: *triangulation* (searching and measuring quality of opportunities for exchange), *transfer* (negotiating terms and exchanging goods, services, data and value), and *trust* (contracting and enforcing)[2]. *Firstly* triangulation, search is restricted to a proprietary network, and quality measurement data are asymmetric and at risk of tampering. *Secondly* transfer, redemption is not interoperable across networks and voucher terms are not programmable. *Thirdly* trust, transactions may be censored, payment is via the trust model and rent-seeking is endemic. Consequently, the transaction cost efficiencies enabled by centralised intermediaries are mostly limited in scale, local in scope, and incompatible with decentralised systems.

3 The Redeemeum Protocol Solution

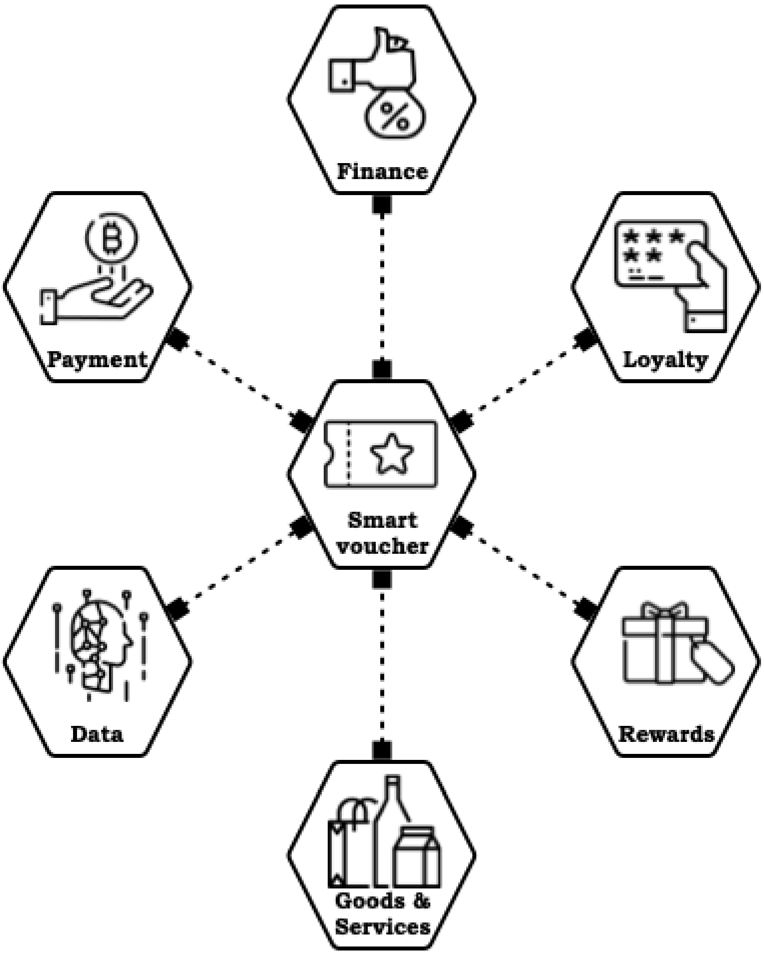
Redeemeum is a modular Web3 protocol for building decentralised commerce applications using smart vouchers that connect decentralised finance and loyalty with real world products and customer data.

We introduce the concept of a *smart voucher* as an interoperable, programmable and composable building block of the decentralized stack. We define a smart voucher as follows:

> *A smart voucher is a financial primitive which represents the right to claim goods or services in accordance with terms which are programmed within and enforced by smart contracts.*

[1] "Requirements and Design for Voucher Trading System (VTS)." https://www.rfc-editor.org/pdfrfc/rfc3506.txt.pdf.

[2] "Tomorrow 3.0 by Michael C. Munger - Cambridge University Press." https://www.cambridge.org/core/books/tomorrow-30/FC3C681277A6AEC527DB5230AA23FB16. Accessed 29 May. 2019.



The role of smart vouchers for enabling decentralised commerce

3.1 Summary of protocol architecture

Redeemeum leverages a number of actors distributed across two distinct categories. *Firstly,* Agents who transact with each other. This category comprises Producers who sell assets as smart vouchers, Consumers who purchase and redeem smart vouchers, and Collectors who implement smart voucher promises. *Secondly,* Keepers who are paid to perform important maintenance roles for the network. This category comprises Aggregators who source the supply and maintain the quality of smart vouchers, Relayers who host order books of smart vouchers for sale to Consumers, and Resellers who purchase smart vouchers and then exchange or resell them to Consumers.

[Note: a more formal description of Redeemeum's architecture is provided within the Protocol Specification section.]

3.2 Comparison with centralised solutions

Redeemeum significantly improves transaction cost efficiency versus centralised systems, across the following key dimensions:

3.2.1 Triangulation

Finding information and locating opportunities for mutually beneficial exchange.

- *Choice and discovery* - is extended via a transparent, open market.
- *Customer and usage data* - Consumers are incentivised to share usage and feedback data via a secure channel and receive payment when a third party accesses, whilst retaining full sovereignty over their data.
- *Open market structure* - enables access to customers and peers.

3.2.2 Transfer

Programming price and terms, facilitating payment and enabling redemption

- *Multiple value and asset types* - enables the redemption of decentralised payments, finance and loyalty for real world products, services and data.
- *Interoperable* - as an open, generic protocol, redemptions of goods and services are enabled by smart vouchers issued as non-fungible tokens (NFTs) on the Ethereum blockchain under the ERC721 standard.
- *Programmable* - programmable terms for each asset instance.
- *Composable* - Redeemeum can be recombined with other modular protocols to create entirely new trust-minimised applications.
- *Implementation costs* - as a universal protocol, Redeemeum reduces the costs of implementing and integrating multiple proprietary systems.
- *Payment* is made via secure, low-cost transactions with dispute arbitration via an intermediary who is game-theoretically incentivized to behave fairly.

3.2.3 Trust

Writing and enforcing contracts.

- *Transaction regulation* - trust is increased by cryptographic enforcement of on-chain terms; and reduction of opportunism for off-chain terms through public transparency, decentralized reputation, and arbitrated escrow.
- *Rent-seeking* - Redeemeum Protocol removes the need for centralised intermediaries, whilst retaining the option of trusted keepers who perform valuable jobs for the network and compete for compensatory fees[3].
- *Market structure* - the protocol's decentralized governance structure provides cryptographically-secured protection against monopolistic rent extraction and promotes an open, competitive market.

[3] "Keepers — Workers that Maintain Blockchain Networks - Medium." 5 Aug. 2017, https://medium.com/@rzurrer/keepers-workers-that-maintain-blockchain-networks-a40182615b66. Accessed 1 May. 2019.

3.3 Solution 1: redeeming onchain value for real world assets

Redeemeum protocol solves the problem of redeeming onchain value for real world products by enabling the issuing, administering and trading of smart voucher tokens using blockchain technology, smart contracts, keeper marketplaces and standardized voucher specifications and interfaces.

For example, Redeemeum enables smart contracts to:

- exchange cryptoassets for real world products and services
- redeem blockchain loyalty point tokens for real world rewards
- purchase goods and services, using decentralised finance
- donate cryptocurrency in exchange for tokenised charitable deeds.

3.4 Solution 2: connecting offchain customer and usage data

Redeemeum protocol solves the problem of smart contracts accessing customer and usage data in a decentralised and self-sovereign way. Each smart voucher instance constitutes a secure, data channel for consumers to share product and usage data or receive advertising or promotional messaging. Whilst the protocol provides incentives to share data, sharing is entirely voluntary and consumers retain sovereignty over their data at all times.

3.5 Solution 3: building decentralised commerce applications

As a modular Web3 protocol, Redeemeum enables developers to build decentralised commerce applications that combine decentralised finance, credit and loyalty with real world products, services and customer data. Redeemeum enables the development of an open ecosystem for decentralised commerce, with the potential to disrupt centralised intermediaries and drive mass adoption of crypto within consumer commerce.

4 Use cases

Use cases for Redeemeum follow, starting with basic transactions, followed by loyalty transactions and finally decentralised finance examples.

4.1 Basic transactions

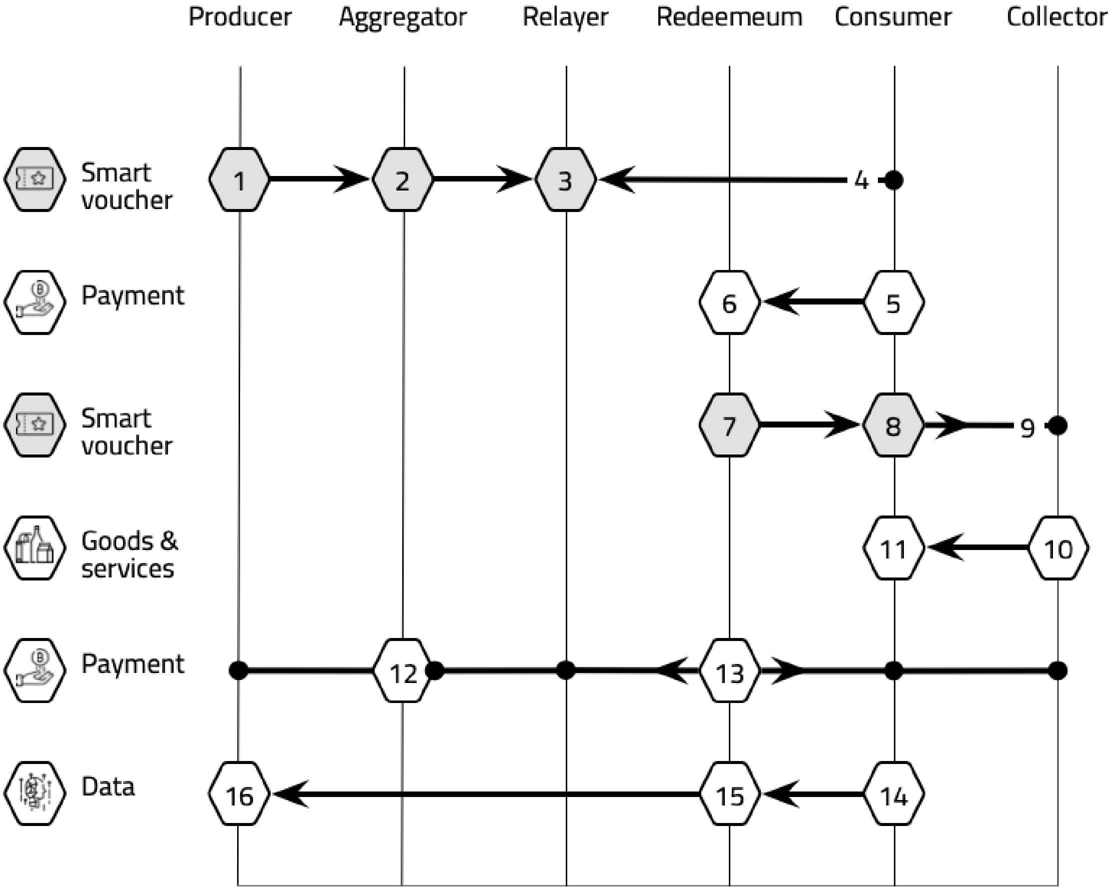
4.1.1 Smart vouchers

Redeemeum Protocol's core functionality enables applications to issue, trade and redeem goods and services represented as smart voucher tokens. For example, a producer can issue a smart voucher token for a product, trade the smart voucher for an agreed price, and then the funds are held in escrow until redemption by the consumer. Payment for smart vouchers can be made trustlessly from a whitelist of cryptocurrencies and tokens, or indirectly from offchain funding sources including bank accounts and payment cards via integrations with oracle services.

4.1.2 E-commerce transactions

Current e-commerce transactions require trust between multiple counterparties. From providing advance payment, through receiving a redeemable promise for goods, to receiving goods. Redeemeum provides a trust-minimised protocol for conducting e-commerce transactions which can be used by existing centralised retailers, hybrid ecommerce platforms such as Public Market or fully decentralised platforms where buyers and sellers are anonymous.

4.1.3 Basic transaction value exchange



1. Producer commits to a redeemable promise for goods or services.
2. Aggregator creates voucher order.
3. Relayer lists voucher order.
4. Consumer selects voucher order representing goods or services.
5. Consumer purchases voucher by transferring onchain funds.
6. Onchain funds deposited into Redeemeum escrow.
7. Redeemeum transfers smart voucher to Consumer.
8. Consumer redeems smart voucher by presenting to Collector.
9. Collector checks Consumer eligibility.
10. Collector implements promise by delivering goods or services.
11. Consumer receives goods or services.
12. Aggregator arbitrates any disputes and disbursement.
13. Redeemeum protocol disburses fees.
14. Consumer optionally shares data.
15. Redeemeum provides secure channel for data exchange.
16. Producer receives customer and usage data.

4.2 Loyalty and rewards transactions

4.2.1 Blockchain loyalty rewards

Loyalty and rewards programs enable customers to redeem loyalty points or tokens into tangible rewards in order to incentivise target customer behaviours, drive brand loyalty and increase customer engagement. Redeemeum enables the onchain exchange of loyalty and rewards tokens into smart vouchers which are redeemable for real world goods and services.

For example, dexFreight is a decentralised logistics platform that incentivizes users with dexMiles rewards tokens. Redeemeum could enable dexFreight users, such as truck drivers, to exchange dexMiles tokens into smart vouchers which could then be redeemed for goods and services such as fuel, hotel stays and decentralised insurance. By providing tangible rewards as tradeable smart vouchers, Redeemeum increases customer perceived value, applies upward pressure on token price, and provides rich customer and usage data to businesses.

Additional use case for issuing smart vouchers as rewards include:

- *Crypto exchanges* enabling exchange tokens (e.g. Binance Coin) to be redeemed for real world rewards in order to differentiate on rewards rather than compete on fees, whilst increasing the value of the token..
- *Blockchain games* such as MyCryptoHeroes enabling game items and points to be redeemed for real world items such as t-shirts, stickers and trophies.
- *Blockchain gaming* applications such as onchain bingo enabling prizes to be paid out in smart vouchers redeemable for real world products.

4.2.2 Consumer credit rewards redemption

Consumer credit products such as credit cards, offer benefits and rewards as a means of differentiating without discounting pricing, whilst also driving brand loyalty and card spend. For example, a premium credit card issuer may offer a bundle of travel benefits such as (airport lounge, spas) and insurance (trip, flight delay) to support annual card fees and higher interest rates.

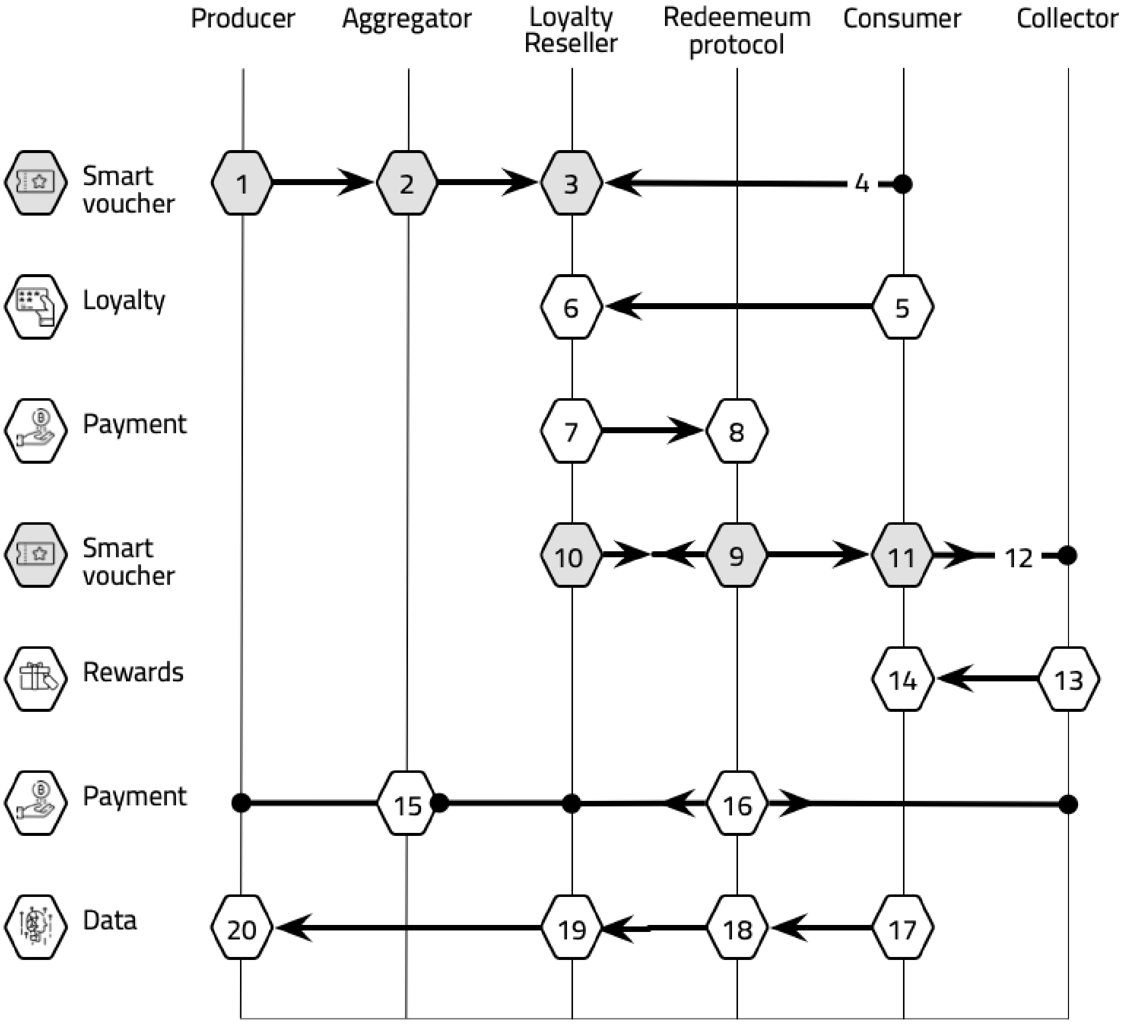
The current market for such benefits is highly intermediated within a few monopolistic players, who provide limited transaction cost efficiencies, extract significant rents and provide minimal customer and usage data. Redeemeum enables centralised purchasers to access an open market of interoperable benefits, which provides a wider range of benefits, at reduced cost and with rich customer and usage data.

4.2.3 Goods, services and deeds

Charitable organisations typically accept donations as an input, and then output charitable deeds. Charities need to be able to provide more transparency to donors and also track and report on their charitable deeds.

By using Redeemeum smart vouchers, charities could enable donors to make donations for specific charitable deeds such as planting a tree in a rainforest or educating a child for a day. For example, the South American Initiative (SAI), provides food and supplies to the people of Venezuela. By issuing food stamps as ERC-20 tokens and representing inventory items as Redeemeum smart vouchers, SAI could gift food whilst providing beneficiaries with increased food security and SAI with traceability and auditability.

4.2.4 Loyalty transaction value exchange



1. Producer commits to a promise for rewards (goods, services, discounts, offers).
2. Aggregator creates voucher order.
3. Relayer lists voucher order.

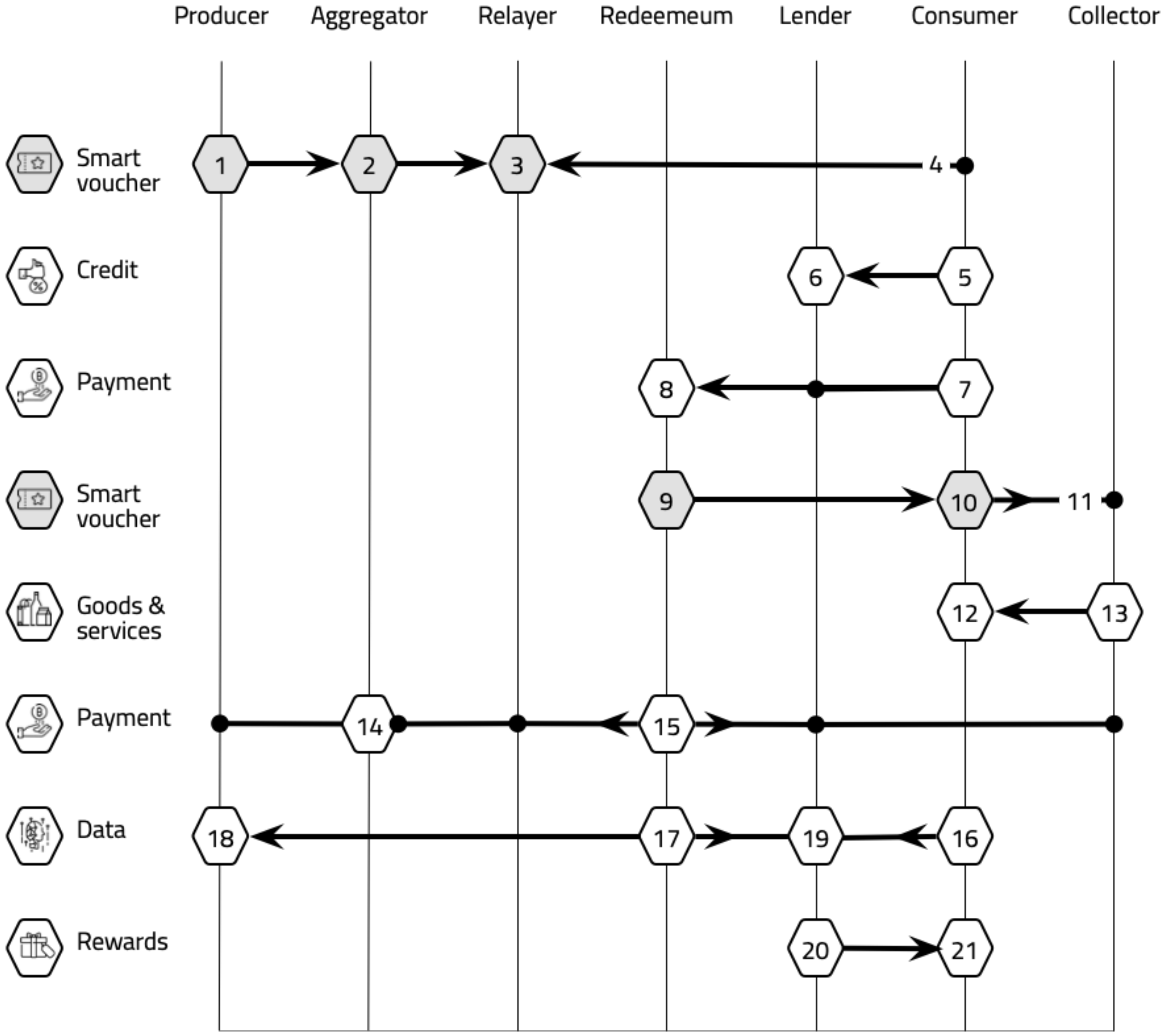
4. Consumer selects voucher order representing rewards.
5. Consumer transfers loyalty tokens, points or credits to Reseller.
6. Reseller accepts redemption of loyalty value for rewards
7. Reseller purchases voucher by transferring onchain funds.
8. Onchain funds deposited into Redeemeum escrow.
9. Redeemeum transfers smart voucher to Reseller
10. Reseller transfers smart voucher to Consumer.
11. Consumer redeems smart voucher by presenting to Collector.
12. Collector checks Consumer eligibility.
13. Collector implements promise by delivering rewards.
14. Consumer receives rewards.
15. Aggregator arbitrates any disputes and disbursement.
16. Redeemeum protocol disburses fees.
17. Consumer optionally shares data.
18. Redeemeum provides secure channel for data exchange.
19. Producer and (20) Reseller receive customer and usage data.

4.3 Decentralised finance transactions

4.3.1 Decentralised Finance

Centralised consumer credit products such as credit cards, store cards and retail finance, enable buyers to purchase products now and pay later. By representing goods and services as Redeemeum smart vouchers, sellers could provide buyers with access to decentralised lending pools that are open, decentralised and trustless. For example, DApps could enable payment to be made for smart vouchers using funds borrowed via integrations with DeFi credit protocols such as Dharma, Maker and Compound.

4.3.2 Decentralised finance value exchange



1. Producer commits to a redeemable promise for goods or services.
2. Aggregator creates voucher order.
3. Relayer lists voucher order.
4. Consumer selects voucher order representing goods or services.
5. Consumer locks-up crypto funds as collateral.
6. Lender issues DeFi credit against collateral.
7. Consumer purchases voucher by authorising transfer of credit funds.
8. Onchain funds deposited into Redeemeum escrow.
9. Redeemeum transfers smart voucher to Consumer.
10. Consumer redeems smart voucher by presenting to Collector.
11. Collector checks Consumer eligibility.
12. Collector implements promise by delivering goods or services.
13. Consumer receives goods or services.
14. Aggregator arbitrates any disputes and disbursement.
15. Redeemeum protocol disburses fees.

16. Consumer optionally shares data.
17. Redeemeum provides secure channel for data exchange.
18. Producer (19) Lender receive customer and usage data.
20. Lender gives rewards to Consumer (21).

4.3.3 Convergent commerce

The dexFreight example can be extended to illustrate how Redeemeum enables convergent commerce by connecting multiple forms of value. dexFreight's business model is to connect the billions of dollars of capital locked-up in freight invoices, with the decentralised liquidity of DeFi. One initiative is to tokenise invoices and use them as collateral within CDPs (collateralised debt positions) for Maker's multi-collateral Dai. In essence, this process inputs illiquid freight invoices and outputs liquid Dai stablecoin. This Dai could be used to fund a decentralised debit card similar to the Dai card[4]. Instead of incurring the friction and fees of converting Dai to fiat for all real world purchases, users could make purchases at participating retailers by converting Dai directly into Redeemeum smart vouchers which would then be redeemed for products and services. Such purchases would be incentivised with dexMiles tokens and provide the retailer with rich customer and usage data.

[4] "Introducing the Dai Card – Arjun Bhuptani – Medium." 12 Mar. 2019, https://medium.com/@arjunbhuptani/introducing-the-dai-card-fc46520078d3. Accessed 30 May. 2019.

5 Technical Requirements

Voucher Trading System should meet the following requirements[5]:

1. It MUST handle diverse types of vouchers issued by different issuers.
2. It MUST prevent illegal acts, like alteration, forgery, and ensure privacy.
3. It MUST be practical in terms of implementation/operation cost and efficiency.

Handling diversity

Redeemeum supports the issuance of vouchers by multiple actors on a universal protocol and handles multiple voucher types, including: coupons, gift certificates, loyalty points, membership cards, exchangeable tokens for goods or services etc.

Security guarantees

Preventing forgery: it is impossible to forge a Voucher Token as long as standard cryptographic primitives remain secure. While traditional centralized voucher systems are typically limited in ways they can validate the redemption of a voucher, Redeemeum offers significantly more powerful verification capabilities.

Preventing alteration: vouchers cannot be altered after issuance. Voucher orders can only be cancelled by Producers.

Preventing duplicate redemption: a voucher can only be redeemed by the voucher holder. Once consumed, it cannot be redeemed again; if the voucher can be used repeatedly, for example a membership card, it is bound to an expiration period.

Non-repudiation: it is not possible to repudiate issuance, trade or redemption by actors, as their digital signatures bind them to the commitments.

Ensuring privacy

Despite the open nature of Redeemeum, it is possible to shield some data, esp. personally identifiable information, following a privacy by design principle.

Operational efficiency

Redeemeum enables discovery of globally available vouchers to anyone, which previously was impractical due to scattered and heterogeneous sources. The resilience of its operation rests in the distributed nature of the protocol. The mature, fully-developed stage of the protocol will be horizontally scalable to meet the global demand.

[5] "RFC 4154 - Voucher Trading System Application ... - IETF Tools." https://tools.ietf.org/html/4154. Accessed 13 Dec. 2018.

6 Protocol Specification

Redeemeum is a protocol that enables issuing, administering and trading of programmable voucher tokens using smart contracts, keeper marketplaces and standardized voucher specifications and interfaces.

Reedeemum leverages the architecture and methodology of Dharma protocol[6], which itself leverages 0x[7], in combination with the VTS (Voucher Trading System[8]) and Generic Voucher Language[9] specifications. With Redeemeum actors, namely Producers, Aggregators, Relayers and Consumers being analogous to Dharma Debtors, Underwriters, Relayers and Creditors.

6.1 Agents

Agents are the platform participants between which Redeemeum Protocol seeks to facilitate interactions.

6.1.1 Producers

The Agent who is selling an asset to a Consumer for an agreed value.
Producer fees:

- Producers may be required to stake fees which may be slashed for non-performance.
- Producers may receive fees for performance.

6.1.2 Consumers

The Agent who is buying an asset from a Producer for an agreed value.

6.1.3 Collectors

Collectors are agents who verify and collect the voucher and implement voucher's promise.

Collectors perform the following functions:

- Commits to a Producer's offer to collect a certain category of vouchers.
- If the Collector wishes to discontinue collecting vouchers, then they will need to cancel their agreement with the Producer.
- Verifies the voucher validity by triggering on-chain validation.

[6] "Dharma White Paper · GitBook." https://whitepaper.dharma.io/. Accessed 8 Dec. 2018.
[7] "White Paper - 0xProject." 21 Feb. 2017, https://0xproject.com/pdfs/0x_white_paper.pdf. Accessed 8 Dec. 2018.
[8] "RFC 4154 - Voucher Trading System Application Programming ...." https://tools.ietf.org/html/4154. Accessed 8 Dec. 2018.
[9] "RFC 4153 - XML Voucher: Generic Voucher Language - IETF Tools." https://tools.ietf.org/html/4153. Accessed 8 Dec. 2018.

- Attests to the consumer's conformity with the terms e.g. by *checking and inputting identity, age, eligibility.*
- Collects the voucher.
- Performs the voucher promise, typically by rendering goods or services.

Note: in case the voucher is issued as a promise for a natively on-chain asset, such as a token or similar crypto-asset, then the collection can be completely automated and the Collector replaced with a zero-fee smart contract by following a hash commitment scheme.

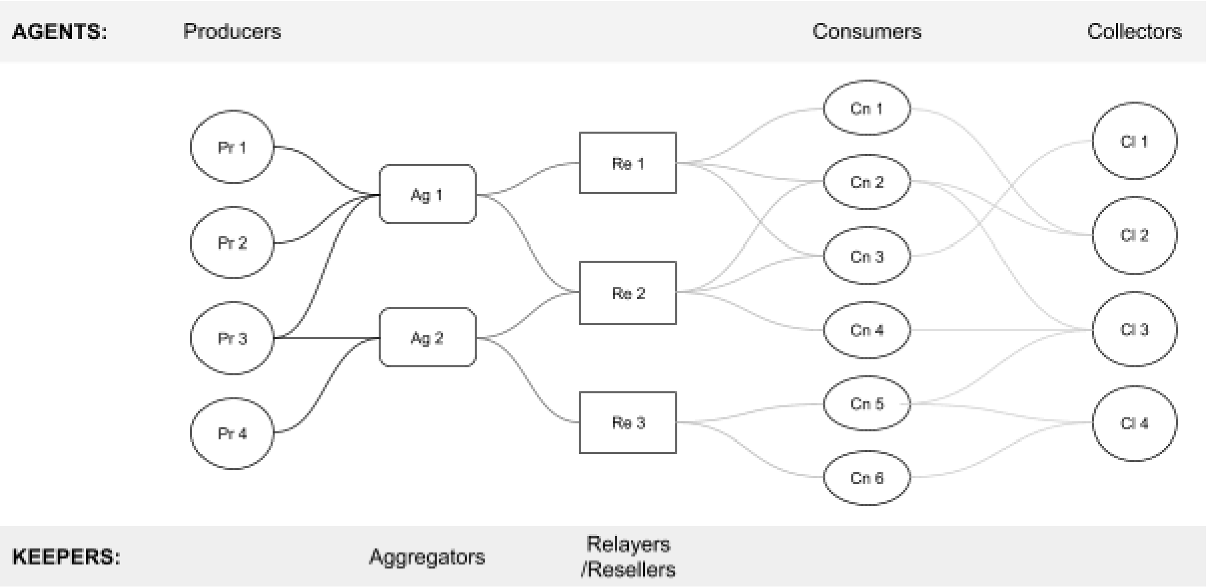
Collector fees:

- Collectors may be required to stake fees which may be slashed for non-performance.
- Collectors may receive fees for performance.

6.2 Keepers

In common with Dharma, we implement a marketplace comprising two Keepers[10], for which we use Ryan Zurrer's definition:

> *"a catchall term for the different utility players in distributed networks that maintain stability and perform crucial jobs in the crypto-economic model"*



Agents and Keepers - platform interaction diagram

[10] "Keepers — Workers that Maintain Blockchain Networks - Medium." 5 Aug. 2017, https://medium.com/@rzurrer/keepers-workers-that-maintain-blockchain-networks-a40182615b66. Accessed 8 Dec. 2018.

6.2.1 Aggregators

An Aggregator is a trusted entity that receives compensatory fees for performing the following functions:

- Originating a Voucher Order from a Producer
- Determining and negotiating the terms of the Voucher (i.e. value/items redeemable, conditions/restrictions) with the Producer
- Committing to the redemption likelihood.
- Committing to the redemption quality.
- Administering the Voucher Order's purchase by forwarding it to any number of relayers.
- Servicing the Voucher -- i.e. ensuring redemption to terms, Consumer satisfaction.

  In the case of defaults or complaints, arbitrating and deciding whether escrowed fees are forwarded to Producer or returned to Consumer.
- Aggregators are rated by:
  - Consumers - Consumer satisfaction, Voucher rating.
  - Producers - fairly handling disputes

  Therefore there is an incentive for Aggregators to curate high-quality Producers, resolve challenges fairly and ensure redemptions are made as agreed.

Aggregator fees:

- Aggregators may be required to stake fees which may be slashed for non-performance.
- Aggregators may receive fees for performance.

6.2.2 Relayers

At a basic level, relayers in Redeemeum Protocol perform an analogous function to relayers in 0x Protocol and Dharma Protocol.

Relayers perform the following functions:

- Relayers aggregate Voucher Orders from any number of Aggregators,
- for an agreed upon fee, host the messages in a centralized order book,
- provide Consumers with the ability to purchase the requested Vouchers.
- Relayers need not hold any agent's tokens, but may do so (see Resellers below)

Note:

1. Relayers provide Consumers with signed voucher-specific metadata associated with the accompanying Aggregator so that they can make informed purchase decisions about the quality of a given Voucher order.
2. Relayers do not freely allow any anonymous party to publish signed Voucher Orders on to their order book, and use their discretion to only accept Voucher Orders from known, trusted Aggregators.

We define two types of Relayers within Redeemeum Protocol:

- **Relayers** - these will provide simple order book functionality, requiring Consumers to take custody of and manage Voucher Tokens themselves. Radar Relay is an example of a typical Simple Relayer.
- **Resellers** - in addition to hosting an order book, Resellers will abstract away the underlying technology by purchasing as B2B buyers and maintaining custody of vouchers on behalf of Consumers. Resellers will then present a standardized UX to Consumers within wallets and consumer-facing apps. Examples of potential Resellers include: OTAs, Loyalty Programs and Payment Card Networks.

6.3 Voucher Specification Overview

We use the Generic Voucher Language definition of a voucher:

> *A voucher is a logical entity that represents a right to claim goods or services. A voucher can be used to transfer a wide range of electronic values, including coupons, tickets, loyalty points, and gift certificates, which often have to be processed in the course of payment and/or delivery transactions.*

Asset is claimed at the redemption of a voucher, i.e. the goods or services delivered. It is offered by the Producer to the Consumer and is the basis that a voucher token is referencing.

Promise denotes the promise of the Producer to deliver the Asset to the Consumer under programmable restrictions and funds allocation. Promises are a core construct that enable reusability across a multitude of participants, while still being anchored to the same Producer.

Promise-Collection construct defines the agreements between Producers and Collectors, for example, which Collectors are authorized for a set of vouchers and the fees they expect in return. If there are no restrictions, it can be omitted.

Voucher Offer then wraps a Promise, adding processing details, such as groupings, validity period etc. It is a base building block for the issuing of vouchers. Offers are uniformly addressable across all Aggregators and Relayers.

Voucher Order is a construct in the hands of the keepers. There can be multiple Voucher Orders created from a single Voucher Offer as each derived Order is assigned to a different Aggregator-Relayer pair, committing to their fees, Aggregator's rating of the voucher etc. It is up to the Relayer to display one order per group or as many orders as are vouchers in the offer's group.

Smart Voucher, or synonymously, **Voucher Token** is created when a Voucher Order is filled - by transferring the requested funds from either the Consumer or from the Reseller. Funds remain locked until redemption or dispute resolution. The holder can then redeem this token for the promised asset.

Note: the process is sequential, but certain types of vouchers and certain use-cases do not require all actors to be involved. The full set of objects maximizes reusability in complex paths.

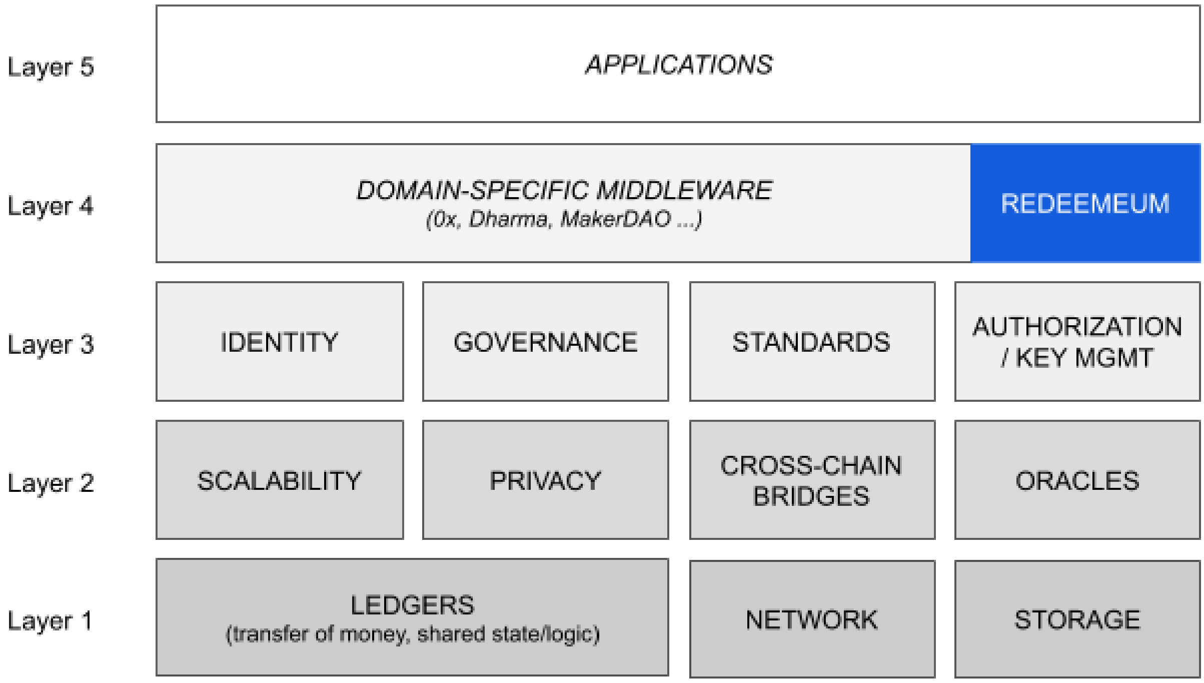
A more comprehensive specification is in: *Appendix - Detailed Voucher Specification*.

7 Technology and Architecture

7.1 Introduction to Technology

The basis of Redeemeum Protocol lies in blockchain technology. It includes a set of basic constructs (a promise, an offer etc.) and logic for matching the supply and demand side of the market.

In the wider ecosystem, Redeemeum Protocol sits on top of domain-neutral, predominantly technocratic platforms and just below the application layer. It is at the lowest contact with a specific usage domain (in this case, vouchers) and is to be used by applications built on top of it. Thus it can be considered as **middleware**, as shown in the diagram.



The position of Redeemeum in the wider stack

As middleware, Redeemeum builds on sub-systems from the lower layers: decentralized identity platforms to address its users (e.g. 3box), relies on governance protocols to manage its operation (e.g. DAOstack), conforms to the standards for its inputs-outputs (e.g. ERC-721) and uses proven key management services (e.g. Clef). This also marks the scope of Redeemeum.

7.2 Emergent Design

Operating in a nascent space, there are several parts impractical to be implemented with the current state-of-the-art permissionless blockchains, facing critical issues with the protection of private data, limited throughput and unpredictable costs of transactions. While these keep on improving, we acknowledge that there are certain fundamental limitations.

As we've learned from the past years, the blockchain part of the layer 1 converges towards global settlement. The layers don't share the same context, so some design assumptions can be changed. This is manifested in several emerging approaches, two of the most relevant for Redeemeum are:

1. *application-specific blockchains*, that are highly optimized and customized to a particular domain, but still maintain a full-blown ledger on their own;
2. *off-chain techniques*, esp. state-channels, that perform the majority of operations off the main blockchain ledger, relying on transient, off-chain transactions that only settle on-chain.

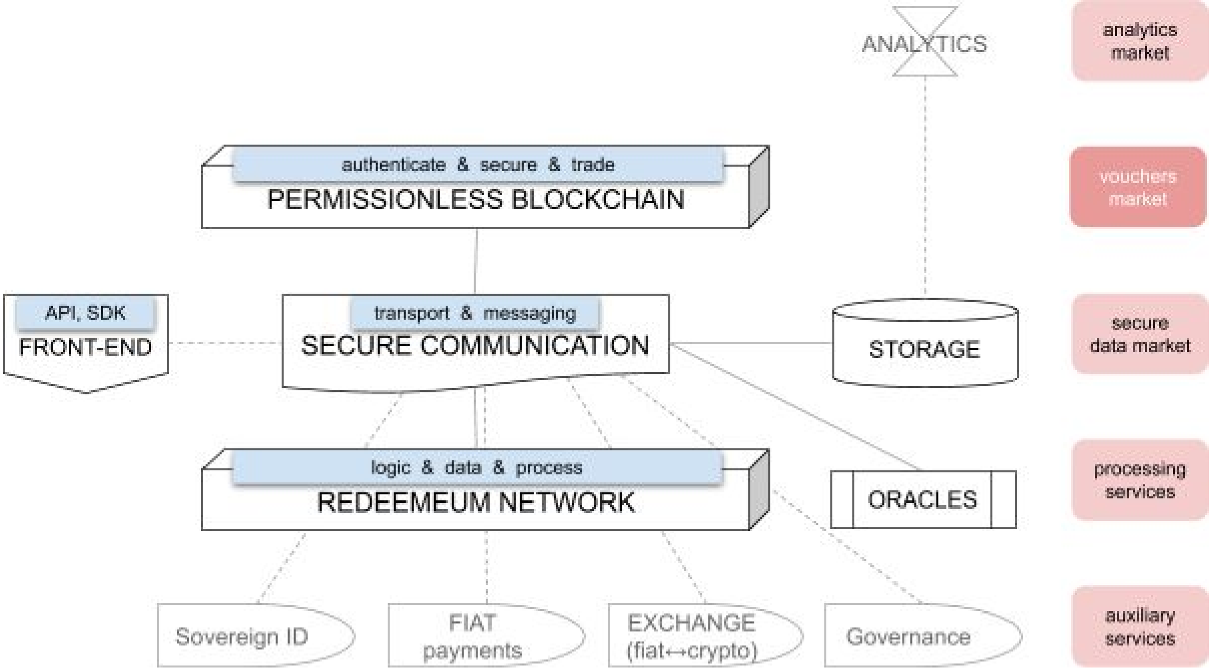
Redeemeum as middleware is designed to be a blockchain-based system on its own, but always coupled with layer 1 blockchain networks, such as Ethereum, for two specific reasons. *Firstly, token standards* on Ethereum have the largest adoption by far and they are only getting stronger, so much so that even corporate blockchain initiatives are taking steps towards interoperability[11]. *Secondly, the network effects* of Ethereum and its *superior development* community make it possible to start building Redeemeum the soonest and contribute to the adoption of the disruptive blockchain technology.

For the reasons described in *Appendix - Development Roadmap*, Redeemeum is starting with a dual-chain architecture and is later expected to morph according to the maturity of its wider ecosystem. Currently, it is a work in progress and locking-in to a particular architecture remains open.

[11] "Enterprise Ethereum Alliance Launches Blockchain-Neutral Token Taxonomy Initiative to Accelerate a Token-Powered Blockchain Future – Enterprise Ethereum Alliance." 17 Apr. 2019, https://entethalliance.org/enterprise-ethereum-alliance-launches-blockchain-neutral-token-taxonomy-initiative-to-accelerate-a-token-powered-blockchain-future/. Accessed 26 Apr. 2019.

7.3 Architecture Overview

The following diagram represents an overview of components in Redeemeum and interoperability with (semi-)external service providers.



Redeemeum technological overview

7.3.1 Vouchers market on Permissionless Blockchain

Issuance of voucher tokens, trading and redemption takes place on public, permissionless blockchain (e.g. Ethereum).

- **Token issuance** is triggered by accepting the funds from the Consumer, which in turn mints a voucher token, with the terms of redemption, costs and the ownership of the voucher explicitly and publicly defined.
- **Trading** is made possible with the voucher token being a standard non-fungible token (i.e. ERC-721 on Ethereum), therefore supported by the majority of token wallets. The protocol is not limiting secondary trading, unless vouchers are so configured.
- **Redemption** of the voucher is recorded in a secure and immutable manner, leaving no room for ambiguity. Additionally, it is uncovering the operational insights of the protocol.

The goal is not to place all functionalities on the permissionless blockchain - on the contrary, it is to be used as little as possible, but to leverage its main advantages: authentication of participants, secure creation of vouchers, secure transfers of funds and trading of tokens.

7.3.2 Processing on the Redeemeum network

Before a voucher token can be issued, its nature must be defined and made available by the keepers. This being a more intensive process in terms of computation and communication resources, it is implemented on a separate network, all the while closely tied to a permissionless blockchain on tier 1.

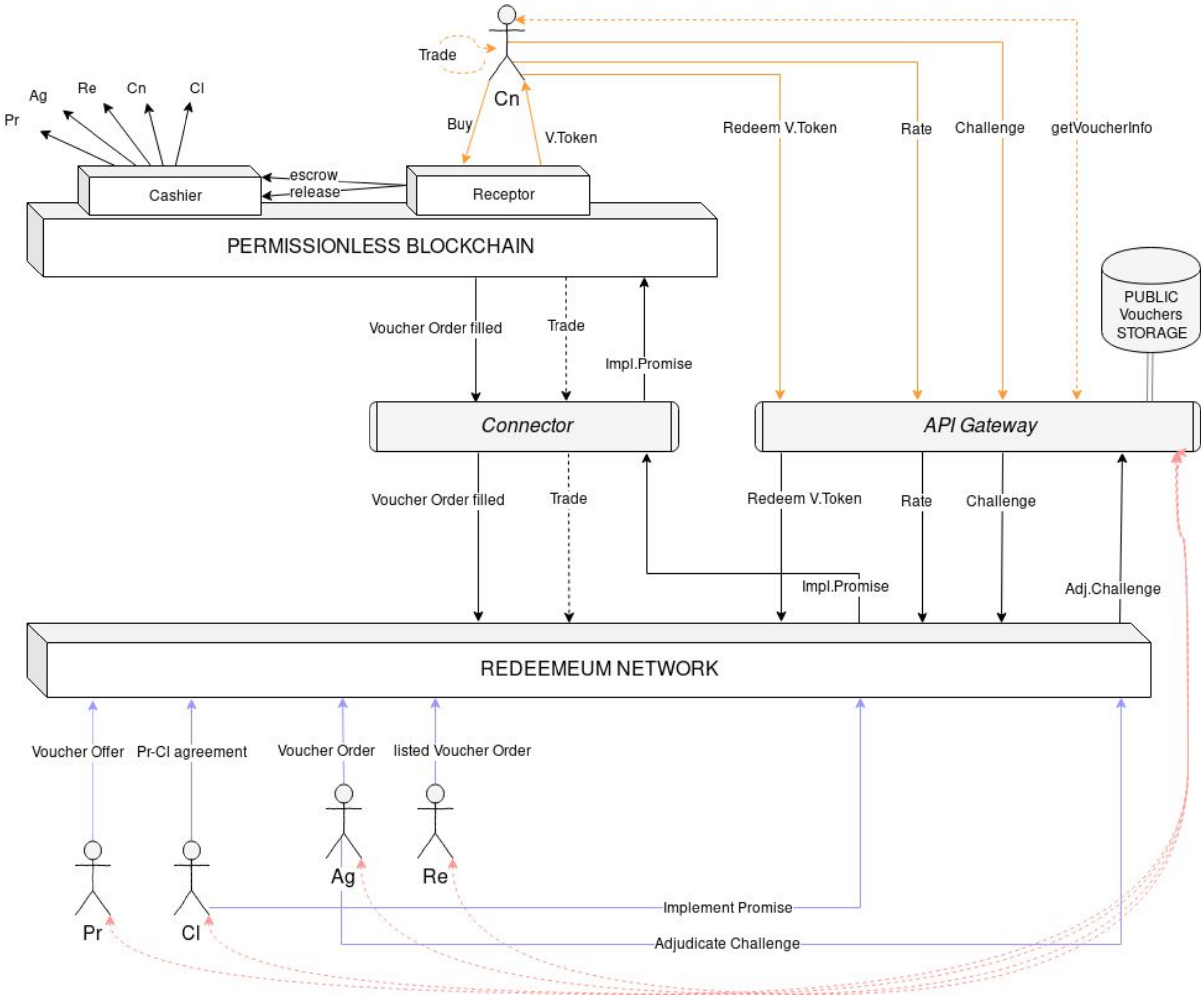
Data: the data on this shared ledger is generally open, with exceptions that protect personal data and potential voucher-specific business secrets. Off-chain data can be made private using standard cryptographic methods, while on-chain privacy benefits from lower transaction costs on Redeemeum network and could eventually support a wider range of pre-compiled cryptographic operations.

Funds: vouchers are programmed to support simple distribution of funds (fees) as per the Terms Contract. The unavoidable technological fees, such as blockchain gas, are paid by the transaction initiator or by a predefined actor.

Processing consists of several actions, that are executed and recorded on-chain:

- the definition of the underlying asset and the promise for its redemption,
- the fee and delivery negotiating between the keepers,
- upkeep of the voucher order set and
- final steps at redemption: validation, execution, ratings and challenges.

The diagram below describes high-level interactions.



Overview of components and interactions in Redeemeum

All participants can use a classic API gateway as a convenience, but notably the Consumer only interacts with the permissionless blockchain, while all others mainly interact with the Redeemeum network and access permissionless chain mainly for payments. From the Consumer's perspective, the protocol is comparatively flat.

7.3.3 Communications

Communications protocol is serving as transport rails for data as well as messaging between participants. It uses end-to-end encryption - since parties are aware of each other's addresses, asymmetric encryption can be used and form private *per-voucher channels*. While the finalized agreements are visible to others, the prior negotiation is private. Communications are also based on a decentralized model, e.g. Postal Service over Swarm and Swarm Feeds.

7.3.4 Storage & Data

To minimize on-chain data for reasons of costs and ease of access, space-consuming data is stored on a specialized decentralized storage system like Swarm, including additional descriptions of vouchers, rich content, voucher ratings, keepers reputation scores etc.

Attention is given to the data and metadata that unlocks valuable insights into the operation and usage of the protocol, such as good data, routing metadata, quantitative & qualitative ratings, value for money etc. All processing and data exposure can be automated through external platforms such as Ocean Protocol, prepared for further use in data marketplaces, analytics etc. Where appropriate, it is consent-driven and processed to protect the privacy and anonymity[12].

The data is a valuable part of the platform, but not all of it can be shared openly, because the platform doesn't own it. Instead, it is the users who control their data and should they agree to share it selectively, that might be against reimbursement and under privacy-preserving guarantees.

7.3.5 Front-end & Developers support

Enforcing compliance with the protocol is a given using blockchain technology, but that is covering mainly the back-end. The front-end is often more chatty, ephemeral and as such implies a less strict approach, covered by providing a standardized API (Application Programming Interface) and an SDK (Software Development Kit) for keepers and consumer-facing applications.

The negotiation part of the process, where the details of vouchers get defined, does not require to be managed through the use of blockchain technology, as standard cryptographic methods suffice. Redeemeum Protocol does not impose restrictions

[12] "Elastic Sensitivity - Noah Johnson, Joseph P. Near, Dawn Song." 4 Sep. 2018 https://github.com/uber/sql-differential-privacy . Accessed 10 Mar. 2019.

about how participants communicate in any way, but it encourages the use of standard APIs to streamline the process and minimize the development time for users. Nonetheless, once the commitments are recorded on-chain, the smart contracts will only allow the process flow as it was agreed upon.

7.3.6 Interoperability with the wider ecosystem

Rather than reinventing the wheel for each auxiliary component, the protocol evolves best when it is in sync with the wider ecosystem, by plugging-in the best currently available solutions. “*The most successful networks, projects, and organisations within the Convergence Ecosystem will be those that have inclusive and aligned communities. The Convergence Ecosystem drives collaboration rather than competition.*”[13]

For these reasons, Redeemeum Protocol is working hand-in-hand with platforms that provide services such as decentralized identity - a long awaited tech that is finally within reach due to blockchain technology. Self-sovereign identity, as its subset, is giving users a true control over their digital identity, because it is consent-based and following the principle of least disclosure. The DID provider will be chosen based on several requirements, such as the ability to use multiple addresses linked to a single identifier, key recovery mechanism to ensure an uninterrupted user experience and compatibility with Redeemeum stack.

Converting between currencies, loans and refactoring are auxiliary services that are naturally left to specialized providers. Likewise, accessing off-chain data and triggering of on-chain transactions are some of the tasks for oracles.

Interoperability with existing, legacy systems is important for the adoption and even though smart contracts can’t access the off-chain world, Redeemeum has placeholders to store passive data which the legacy software can interpret and trigger further actions. For example, it is possible to attach arbitrary data to the voucher, that is needed to verify the redemption conditions, such as a special code to redeem the voucher for a rent-a-car service.

Redeemeum, a web3 component for vouchers. The capability of blockchain to empower the edges by unlocking the decentralized internet with read/write/execute operations, also re-defines how vouchers can be used. As a web3 component, Redeemeum provides a uniform way to issue, trade and redeem a diverse set of voucher types. It can be embedded into other DApps, for example to issue exchangeable tickets or to reward their users with redeemable tokens for goods/services, a.k.a. smart vouchers.

[13] "The Convergence Ecosystem, Building the Decentralised Future - Outlier Ventures." https://outlierventures.io/wp-content/uploads/2018/03/The_Convergence_Ecosystem_Report_Outlier_Ventures_2018.pdf . Accessed 15 Feb. 2019.

8 Voucher Issuance and Redemption Process

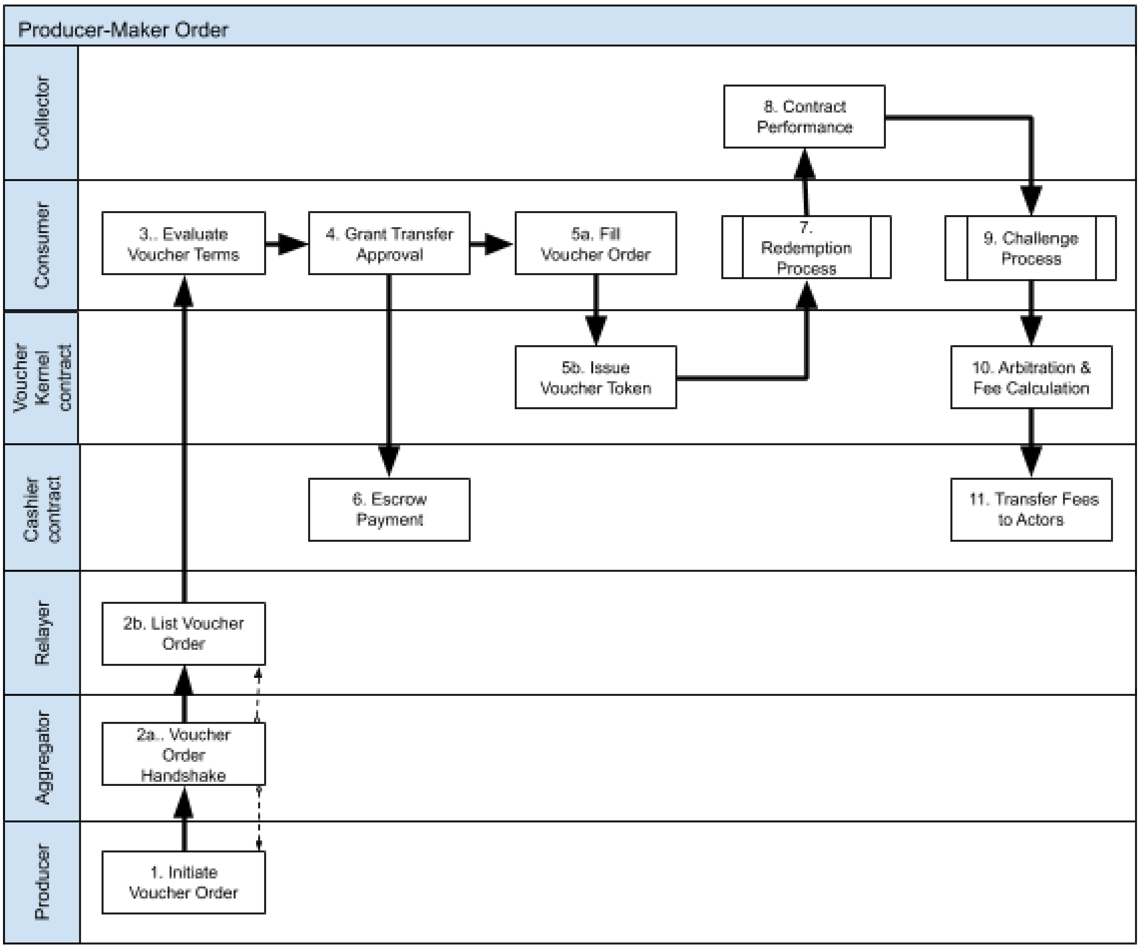
A voucher token is issued when a voucher order is filled, passing all checks and payment forwarded to escrow. Zero-pay vouchers are also possible when all parties relinquish their fees.

Note: henceforth, complex paths are described. Redeemeum Protocol does, however, not prevent simpler flows, e.g. where the Producer is also the Collector and/or Aggregator, or Relayers are omitted, or there are multiple Aggregators and/or Relayers for a single voucher offer.

- **Producer-maker Orders** - where the Producer is offering assets for sale, typical for voucher type of exchangeable tokens; *the focus of this chapter.*
- **Consumer-maker Orders** - where Consumer requests a voucher, typical for voucher types such as membership cards, gift certificates etc.

8.1 Producer-Maker Orders

The following diagram describes voucher lifecycle from the offer till redemption.

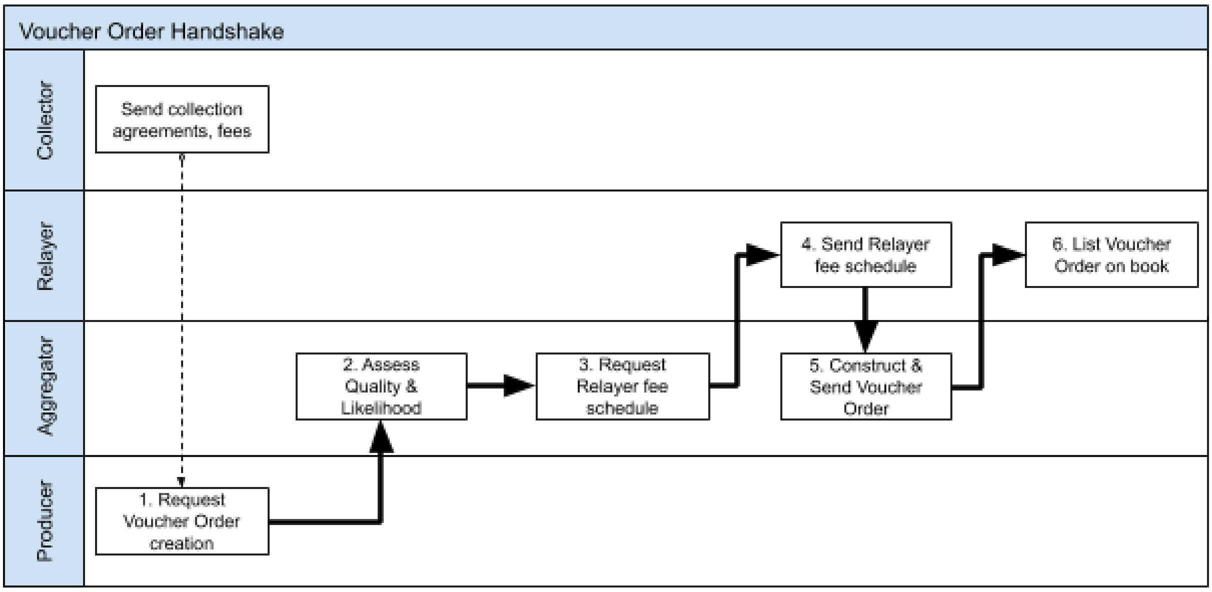


1. Producer commits to a promise of delivering a good or service. This is an offer to an Aggregator, who then assembles it into a voucher order. Prior to that, Producer can negotiate the delivery with a set of Collectors.
2. (a) Voucher Order handshake (see details below) is performed between Producer, Aggregator and Relayers, (b) resulting in the Relayer listing a complete Voucher Order.
3. Consumer evaluates terms of Voucher Order on Relayer's public order book.
4. If Consumer wants to buy a voucher, i.e. fill an order, Consumer grants approval for transferring sufficient funds. *Note that a single transfer approval may be sufficient for multiple vouchers, potentially bought later.*
5. The Consumer then (a) fills the Voucher Order. Voucher Kernel contract then (b) issues to the Consumer a non-fungible, non-divisible token representing the Consumer's agreement to the terms contract and the right to redemption.
6. The Cashier contract transfers the payment amount into escrow.
7. Later on, the Consumer initiates the Redemption Process (detailed below).
8. The Collector performs the contract (this could be providing a service, delivering a good or applying a discount or promotion).
9. The Consumer may challenge the redemption during the challenge period, in which case the Aggregator will perform arbitration (or invoke an external service) and may recalculate fees and/or slash deposits.
10. The Cashier contract transfers calculated fees to actors.

8.2 Voucher Order Handshake

The following diagram and text describes the Voucher Order Handshake process, where Producers, Aggregators and Relayers determine and agree on voucher's qualitative rating and distribution of fees.

Note: the agreements between Producer and Collectors, authorized to collect a set of vouchers, are arranged beforehand and need not be repeated.

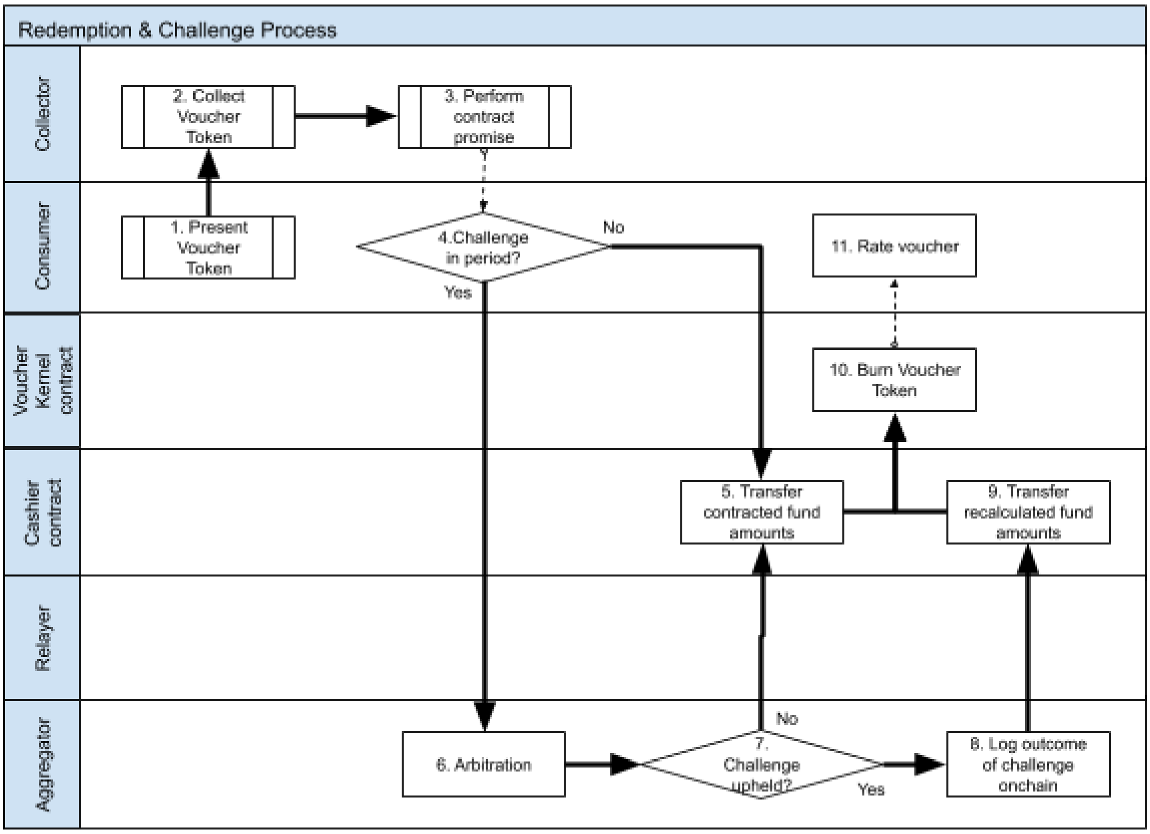

Voucher Order Handshake
Collector
Relayer
Aggregator
Producer
Send collection agreements, fees
1. Request Voucher Order creation
2. Assess Quality & Likelihood
3. Request Relayer fee schedule
4. Send Relayer fee schedule
5. Construct & Send Voucher Order
6. List Voucher Order on book

1. Producer requests creation of a Voucher Order from an Aggregator by sending a Voucher Offer, with defined promise of goods/services to be delivered under defined conditions by contracted Collectors.
2. Aggregator assesses the Likelihood of Redemption and Quality Rating of the offer.
3. If Aggregator wants to use Relayer(s), then Aggregator requests their fee schedule.
4. Relayer sends fee schedule and receiving address to Aggregator.
5. If fees comply with Aggregator parameters, then Aggregator constructs a complete Voucher Order using inputs from Aggregator and Relayer, and assigns Voucher Order to Relayer.
6. Relayer lists completed Voucher Order on their order book.

8.3 Redemption & Challenge Process

The following diagram and text describes the redemption and challenge process, which assumes initial conditions where Voucher Kernel contract has transferred newly minted Voucher to Consumer and escrowed payment and any deposits.

Note: challenging is possible immediately after the minting of a Voucher Token, up until token expiration delayed by the full challenge period. E.g. a voucher token is minted at time T, expires at $T+10$ with a challenge period of 3, then a challenge is possible in the interval $[T,\ T+10+3)$.



1. Consumer presents voucher token to Collector.
2. Collector validates and collects voucher token.
3. Collector performs the promise (typically renders goods, services or promotional offer).
4. The challenge period starts ticking upon redemption. If the challenge period is undefined (i.e. equals 0), the voucher does not support challenges.
5. If the Consumer does not challenge before the end of the challenge period then Voucher Kernel contract sets the *r-value* to 1 (redemption promise delivered), then it calculates originally contracted fund amounts and transfers them via Cashier contract.
6. If Consumer challenges before the end of the challenge period then the Aggregator performs off-chain arbitration.
7. If the challenge is not upheld, the process continues as in step 5. If the challenge is upheld, it is followed by step 8.
8. Aggregator registers outcome of challenge onchain by setting the *r-value* between 0 (redemption promise not delivered) and 1 (promise delivered).
9. Cashier contract calculates and transfers fund amounts.
10. Voucher Token is finally burned when the challenge period expires or the challenge is resolved.
11. Consumer can optionally rate the voucher after the redemption or when potential challenge is resolved.

8.4 Funds

In order to give the system maximum security, the management of funds is defined in detail by the Terms Contract and controlled by the Voucher Kernel:

- input funds are in accordance with input code, e.g. Consumer buys the voucher and funds are transferred into escrow until redemption;
- participants commit to smart contracted terms, e.g. by listing their fees;
- commitment to the authority of the Aggregator to arbitrate a challenge;
- receive output funds in accordance with output code.

Redeemeum has a defined schema for the escrow and distribution of input funds. Alternative schemes can be defined and linked to the Voucher Kernel contract.

See detailed description in *Appendix - Details of Funds Distribution*.

9 Token Model

When designing complex systems such as token models, it is beneficial for designers to be able to draw upon the results of previous experiments. Results from early token experiments are only just beginning to augment cryptoeconomic theory with empirical evidence. What follows then, is the starting point for the evolution of Redeemeum's token model.

9.1 Protocol objectives

Redeemeum protocol's high-level objectives are to:

- Incentivise participation in the network
- Enable the origination and creation of smart vouchers.
- Set accurate expectations of quality- by accurately report the quality of the goods and services underlying smart vouchers to enable Consumers to make an informed purchase decision.
- Be an honest broker - by fairly arbitrate disputes against the standard of the quality rating.
- Enable participants to contribute to the operation of the protocol and be compensated accordingly.

Protocol challenges that need to be mitigated are:

- How to verify that smart vouchers were redeemed?
- How to assess the quality of smart voucher redemptions?

9.2 Redeemeum Token

Redeemeum (RDM) is the native ERC-20 compatible work token of the Redeemeum Protocol that game-theoretically incentivises protocol actors to participate in a secure and economically rational way, without the need for a centralised third party.

Redeemeum tokens are necessary in order to enable the protocol to function as follows:

- To incentivise valuable work and transaction validation through staking, and to deter poor work and protocol violations through slashing.
- For coordinating the assignment of work in accordance with the value of staked and delegated tokens.
- As a unit of account for protocol fees and rewards.

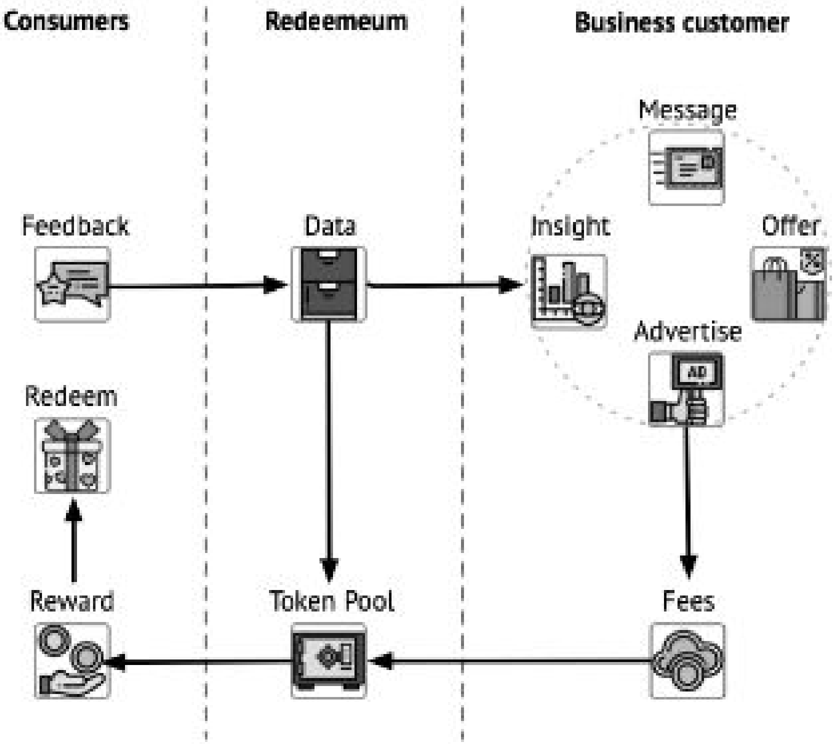
RDM is not a medium of exchange token, instead smart vouchers may be purchased using any currency denominated within the smart voucher contract. Protocol fees may be paid by any token that is contained within a whitelist which is curated by the governance mechanism. The expectation being that suitable currencies would be stable, economical and liquid.

9.3 Token model function

The Redeemeum token model functions as follows:

- *Validate transactions*
  - Aggregators perform the role of validators on the Redeemeum network.
  - Validators stake Redeemeum tokens and receive rewards every block.
  - Redeemeum token holders can delegate their tokens to Aggregators to earn a share of block rewards.
  - Validators and Delegators can have their stakes slashed for protocol violations.
- *Perform work*
  - Aggregators perform the work of onboarding Producers, originating voucher orders, accurately reporting the quality rating of smart vouchers and arbitrating fairly.
  - If Consumers rate redemption quality equal or above the Aggregator's rating, Aggregator and Delegators are rewarded. If consumers rate below, stakes are slashed.
  - If Producer or Consumer deems Aggregators dispute resolution unfair, then stakes are proportionally slashed.
  - Therefore Aggregators are incentivised to set reasonable expectations of quality and be an honest broker in arbitrating disputes.
  - The demand and therefore pricing for smart vouchers is driven by quality ratings which incentivises Producers and Collectors to maintain high-quality.
- *Assign work*
  - Aggregator work (originating voucher orders, accurately reporting the quality rating of smart vouchers and arbitrating fairly) is probabilistically assigned based on an Aggregator's staked and delegated tokens.
- *Verify work*
  - Verification of redemption will be performed onchain by signing of Voucher Tokens by Consumers and Collectors. However, verification of quality is described within 'Perform work' section.
- *Payment of fees*
  - Redeemeum tokens are used for the payment of protocol fees.
- *Rewarding target behaviours*
  - Target behaviours including data sharing, and consenting to receive messages and advertising will be rewarded with Redeemeum tokens. The size of rewards will be a function of e.g data shared, third party usage of shared data and Redeemeum tokens staked by the Consumer. Redeemeum tokens earned can then be staked, sold or redeemed for smart vouchers within the system. At all times participants retain sovereignty over their data, and participation is entirely voluntary.

- *DApp store business model*
    - The paid side of Redeemeum's business model provides businesses with access to data and insight and also access to customers via advertising, offers and messaging. Fees are paid into the token pool using Redeemeum tokens, with fee levels set via the governance mechanism. A dapp store business model enables third party businesses to consume native services and charge additional fees for their own services via the Redeemeum token, whilst capturing some of the value within the Redeemeum token itself.



Redeemeum model value flow

9.4 Token issuance

Network participation is important in order to ensure the quality and security of the network. Therefore, the Redeemeum token issuance model will algorithmically adjust the inflation rate in order to meet a specific participation target which is set and adjusted by the governance mechanism. This means that if the participation rate is below the target, the inflation rate will increase thus increasing rewards and incentives to participate. Conversely, should the participation rate be exceeded, the inflation rate will decrease- reducing the incentives to participate.

10 Governance

Redeemeum protocol will be governed by a DAO which will enable Redeemeum token holders to:

- make decisions regarding updates to the protocol to ensure that the protocol thrives
- set protocol parameters such as participation targets, fees and rewards
- assign protocol funds to development initiatives, grants.

The precise mechanism for governance will be developed in tandem with the development of the protocol and with reference to the ongoing developments in this space. At time of writing, the projects with the most influence on the authors' thinking are the governance mechanisms of Polkadot and dxDAO, and the governance platforms of Aragon and DAOstack.

11 APPENDIX

11.1 Detailed Voucher Specification

Let $Asset$ be claimed at the redemption of a voucher, i.e. the goods or services delivered. It is offered by the Producer Pr to the Consumer Cn. Versioning enables continuous operation of older assets to coexist with new ones. Categorization is used for narrowing the discovery of vouchers and can include prefixed namespaces for more detailed categorization. An asset can be further described with a pointer to additional information, URI (Uniform Resource Identifier).

$Asset \equiv \{ID_{asset}, Pr, version, title, description, category, URI\}$

where ID_{asset} is asset's identification, calculated as:

$ID_{asset} \equiv hash(Pr, version, title)$

Let $Promise$ be the promise of the Producer to deliver the Asset to the Consumer under programmable restrictions on redemption, collection, monetary allocations and human-readable conditions. Promises are a core construct that enable reusability across multitude of participants, while still being anchored to the same Producer.

$Promise \equiv \{ID_{promise}, ID_{asset}, value, merchandise, conditions_{TXT}, challenge_{period}, terms\}$

where $ID_{promise}$ is calculated as a hash of its components;

$value$ is an encoded value of the voucher, defined with the type of the voucher, the type of the representation of value which can be either amount or percentage, currency denomination, and the number of vouchers needed for redemption (zero, if the voucher can be used repeatedly and is therefore not consumed).

$value \equiv \{type_{voucher}, type_{value}, quantity, currency, spend\}$

where:
$type_{voucher} \in \{exchange\ if\ 0,\ discount\ if\ 1,\ monetary\ if\ 2\}$
$type_{value} \in \{fixed\ if\ 0,\ ratio\ if\ 1 \wedge type_{voucher} = discount\}$
$quantity \in \{amount\ if\ type_{value} = 0,\ percentage\ if\ type_{value} = 1\}$
$spend \in \{n_{consume}\ if > 0,\ 0_{present}\ if\ 0\}$

merchandise is an optional, collector-specific meaning of the voucher, important for the Collector's interpretation, such as a voucher identification by the Collector or a pointer to an external restrictions object. Note that if *merchandise* is specified, then $conditions_{TXT}$ must also be specified in order to enable understanding of restrictions in natural language.
$merchandise \equiv ID^{Cl}_{voucher} \vee ext.restrictions^{Cl}_{voucher}$

$challenge_{period}$ is optional. The redemption process can support a challenge by the Consumer when $challenge_{period} > 0$, in which case the Aggregator is engaged for dispute resolution.

terms define the collection and allocation of funds, defined within an immutable contract, that is instantiated with parameters corresponding to a particular voucher set; terms are a separate construct, enabling common calculation templates:
$terms \equiv \{terms_{contract}, terms_{parameters}\}$

where the $terms_{contract}$ is deployed at a particular address, defining the calculation of input/output funds (such as the fees per participant); and $terms_{parameters}$ instantiating the terms contract with specific parameters, encoded in a predefined format.

Let $Promise_{collection}$ define the agreements between Producers and Collectors, for example, which Collectors are authorized for that voucher and their fees. If there are no restrictions, it can be omitted. The redemption process can support a challenge by the Consumer (when $challenge_{period} > 0$), in which case the Aggregator is engaged for dispute resolution. It is also used to register redemption attempts at a particular Collector.
$Promise_{collection} \equiv \{agreement^{Cl}_{voucher}, redemption^{Cl}_{i}\}$

Let voucher *Offer* then wrap a Promise, adding processing details such as groupings and validity period. It is a base building block for the issuing of vouchers, uniformly addressable across all Relayers, which is required to manage the number of available voucher offers.
$Offer \equiv \{ID_{offer}, ID_{promise}, quantity_{group}, quantity_{remaining}, Pr, validity_{period}\}$

where:
ID_{offer} is voucher offer's identification, calculated at the time of generation as:
$ID_{offer} \equiv hash(ID_{promise}, timestamp, validity_{period})$

$quantity_{group}$ denotes the number of similar vouchers in the group. Default is 1, meaning that a default group of equal ("fungible") vouchers represents only one "non-fungible" voucher.

$quantity_{remaining}$ denotes the remaining number of available vouchers in the group. Default value is equal to $quantity_{group}$. It is decreased when voucher orders get filled.

Finally, a $Voucher$ is instantiated from the $Offer$ when the offer is accepted by the Consumer, usually against payment, but not mandatory.
$Voucher \equiv \{ID_{voucher}, Cn, ID_{offer}\}$

11.2 Development roadmap

There are many projects exploring multi-layered operation, as shown at a high-level in the table below. Until they mature, a trust-minimized mode is an unavoidable compromise at decentralization.

Design options	Additional consensus	Fit for few state updates from undefined parties	Instant	Finality	Mass Exit, Destruct	Fees	Private	Business-entity coupling
Sidechain, general	*yes*	*yes*	*no*	*depends*	*yes (-)*	*small*	*could be*	*/*
Standalone chain	*yes*	*yes*	*no*	*depends*	*no*	*small*	*could be*	*/*
Hardware-based (TEE)	*yes*	*yes*	*depends*	*depends*	*no*	*small*	*could be*	*Intel (-)*
Plasma	*yes*	*yes*	*depends*	*depends*	*yes (-)*	*small*	*future*	*/*
Cosmos	*yes*	*yes*	*~7s*	*yes*	*no*	*small*	*future*	*Cosmos*
Polkadot/ Dothereum	*no*	*yes*	*~10s*	*~yes*	*no*	*small*	*Parity-specific*	*Parity*
State-channels	*none (+)*	*no (-)*	*yes (+)*	*~yes*	*no*	*min (+)*	*yes*	*/*

Extract of design options
(+) marks a significant advantage, (-) marks a significant disadvantage.

The comparisons in the table are simplified for brevity and are bound to change. New alternatives will emerge and will be constantly evaluated, for example, there are high expectations from new zk-rollup techniques, however we are also not excluding the ultra secure Bitcoin-based sidechain later on. The initial onus is on constructing the core logic for vouchers and not immediately lock-in on a detailed architecture.

In summary: there are significant advantages in using multi-party virtual state-channels[14], but not only are they currently underdeveloped[15], they are also not (yet) suitable for Redeemeum use-case with very few state updates between parties in a life cycle of a voucher. Accounting for that, in this nascent space, greatest flexibility is advantageous, so approaches that support gradual evolution are preferable, e.g. if one wants to leverage Ethereum's superior (token) ecosystem,

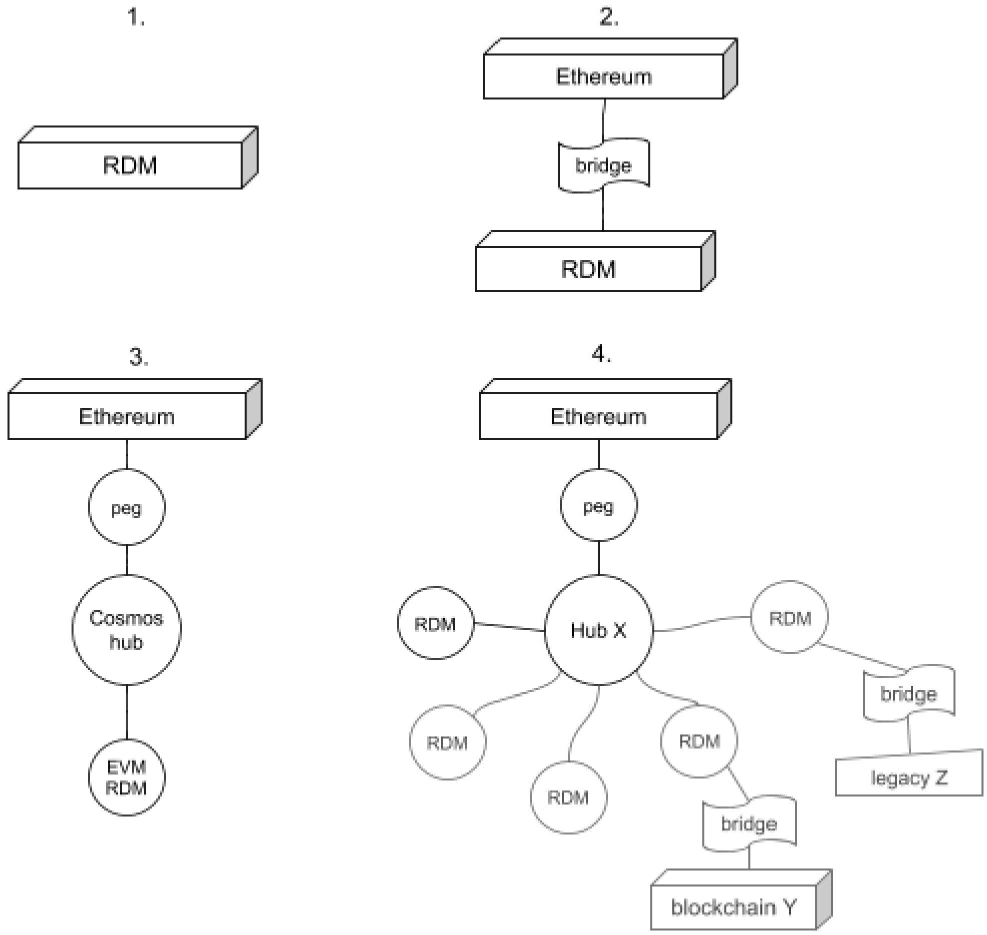
[14] "Systematization of Knowledge: Off The Chain Transactions - L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, A. Gervais" https://eprint.iacr.org/2019/360 . Accessed 20. Apr. 2019.

[15] "Multi-party Virtual State Channels - S. Dziembowski, L. Eckey, S. Faust, J. Hesse, K. Hostáková" 18 Apr. 2019 https://link.springer.com/chapter/10.1007/978-3-030-17653-2_21. Accessed 19. May 2019.

then Cosmos and Polkadot offer a comparatively long runway. Thus, Redeemeum will kick-start with application-specific blockchain design.

There is a downside of having a secondary shared ledger: the cost of securing it. Its maintainers are expected to offer *secure and reliable service*, which bears non-negligible costs, covered via a tenable token distribution model.

The development roadmap must thus be expressed in evolutionary stages, honoring the nascent nature of the ecosystem. The technologies on the diagram below are based on current research.



Architecture roadmap of Redeemeum

Initially, a proof-of-concept is created on a private chain, developing the core constructs of the protocol. It is immediately followed by stage 2, where the previously isolated chain is connected to Ethereum network via a trust-minimized cross-chain bridge. This stage takes longer to develop as it showcases the full potential of the protocol. It is a contemporary design, found in many adjacent projects, that could support Redeemeum and grow with it for a duration.

Stage 3 is characterized as a short, intermediate step towards the longer-term design, where smart contracts are migrated into a Cosmos realm. This is an infrastructural change: the core logic stays the same, Solidity code is preserved and moved into the EVM environment of Ethermint as a Redeemeum zone, connected to Cosmos hub. The link to Ethereum is migrated from a trusted bridge to a general-purpose Ethereum peg zone, a.k.a Peggy.

Next stage 4 represents an optimized and scalable Redeemeum. The logic is rewritten in Golang thus making it more powerful, the hub is chosen such that supports the desired requirements, integrations with legacy systems and other blockchains are established, e.g. using Interledger. Last, but not least, Redeemeum can scale horizontally by deploying multiple zones. Industry- or use-case- specific applications of the protocol could be deployed in separate zones, while maintaining seamless communication between the zones and other blockchains by leveraging on Cosmos Inter-Blockchain Communication (IBC).

The majority of the logic is constructed in stage 2, so it is the focus of this paper.

11.3 Smart Contracts

The core logic is driven by smart contracts that enable a high degree of autonomy, needed for an uninterrupted and uniformly accessible public protocol. Measures are taken to evolve contracts in a graceful way and to allow intervening in the automatic operation via a chosen governance mechanism.

Asset Registry serves to provide the transactional base. Vouchers can be issued multiple times for the same asset. Versioning enables updating the asset details as its properties change. Example: rent-a-car package, a bar of chocolate ...

Voucher Kernel controls the use of the voucher, such as the issuance of the token, possible transfers between Consumers and final redemption. Voucher Kernel covers core logic:

- managing records of available vouchers (voucher orders, groups etc.)
- minting of voucher tokens
- map between Term contracts and voucher tokens
- routing payments and fees between actors
- routing metadata
- arbitration in case of disputes
- ratings by Consumers

Terms Contracts are recording the financial terms and conditions of vouchers. Being in a separate contract, they can be reused for multiple vouchers while also making it easier to shield potentially private data. Financial terms for vouchers generally conform to a common format, with flexible price/costs variables, starting from the simplest zero-fee tokens awarded to consumers directly, to more complex value-chain, e.g. reselling, dispute management, collateralization agreements etc.

Collection Contract defines the agreements between Producers and Collectors (authorizations and fees), which are verified during the redemption. It is used for registering redemption events.

Cashier Contract is managing the funds and is minimalistic for security reasons. It can accept funds as inputs from participants as per the Terms Contract, hold them in escrow and release them to corresponding actors after the redemption.

Transfer Proxy is an intermediary contract managing approvals of fund transfers in any currency. It is a separate contract in order to support upgrades of the protocol logic without requiring additional re-negotiating.

Oracle Contract enables interactions between blockchain and the external world. Some examples include: intercepting fiat payments, fetching currency rates, etc.

11.4 Operational Considerations

The nature of any decentralized project bears risks that must be considered in advance. Just as the use of blockchain technology provides a more resilient security through Object-Capability model[16], its immutable nature limits the options of responding to incidents. Identifying and addressing risks is a crucial part of blockchain-based system design, here only indicated for brevity.

Securing the nodes

As the network will be public, there needs to be appropriate security model in place for the node operators, such as sentry nodes that are protecting full nodes and unikernel-based design, that is both lightweight to maintain and has minimal security footprint. Unikernels, such as MirageOS and UniK provide an immutable infrastructure for node operators, which is often under-appreciated in the blockchain space.

Contingency procedures

Critical smart contracts can be paused and later resumed to respond to emergency situations. Details depend on the governance model in place. Redeemeum will facilitate a predefined way for emergency communications between affected participants, minimizing the uncertainty of such events.

Code audits

Redeemeum is encouraging a wider community to participate in its development and maintenance. An attack where malicious code is submitted to the project's source-code repository needs careful consideration[17]. All contracts must undergo

[16] That is in contrast to the traditional Access Control Lists, which are problematic for being concentrated targets for hackers and thus exploited

[17] "Enterprise Ethereum Alliance Client Specification v3" https://entethalliance.github.io/client-spec/spec.html . Accessed 15 Feb. 2019.

security audits by reputable specialists, no exceptions. Contracts that are involved in the management of funds are maximally restricted in their scope.

Copycats

The defense against parasitic contracts rests in the minimal operating fees of the protocol and its valuable state[18] (funds in escrow, network effects, off-chain integrations and partnerships).

11.5 Details of Funds Distribution

We define the following variables:

- I_j is the input funds for participant j
- O_j is the output funds for participant j
- D_j is the decimal fraction of net voucher price that transfers to output funds for participant j, where $D_j = O_j / V_{NP}$
- F_j is the flat fee that transfers to output funds for participant j
- r is the redemption coefficient r, which represents the degree to which the promise has been delivered
- G_F is the total gas fees*
- V_P is the voucher price

Several methods are provided, such as:

- sending the funds into escrow, registering the input accordingly;
- optional routing of Consumer's actions to the blockchain-connected proxy that pays for technological fees, e.g. gas paid by the Producer on behalf of the Consumer
- calculating the outputs per participant when releasing the funds.

The following table describes a simple example where:

Inputs:

- Input code
  - Consumer Inputs voucher price, so $I_{Cn} = V_P = 100$
  - For all other participants Input is zero, $I_j = 0$

Redemption

- Redemption is completed but a challenge is upheld, with r = 0.5

Outputs

- Output code:
  - Any surplus funds are returned to Consumer, so
    $O_{Cn}= V_P - O_{Pr} - O_{Ag} - O_{Re}$
  - All other participants take a fraction of funds: $O_j = r * D_j * V_P$

[18] "On Value Capture at Layers 1 and 2 - Multicoin Capital." 14 Mar. 2019. https://multicoin.capital/2019/03/14/on-value-capture-at-layers-1-and-2/ . Accessed 26 Apr. 2019.

- Output values:
    - Producer receives 92% of funds
    - Aggregator receives 5% funds
    - Relayer receives 3% funds
    - As r = 0.5 above funds are proportionately reduced
    - All other participants receive 0% funds

So:

$\mathbf{D_j}$	**Value**
D_{Pr}	0.92
D_{Ag}	0.05
D_{Re}	0.03
D_{Cn}	0
D_{Cl}	0
D_{3P}	0

**Note*: technological fees, such as gas on Ethereum main network, are excluded from the example for simplicity. In real scenario, they could be covered by the keepers, in order to make the user experience smoother.

Participant	**Input Code**	**Input values**	**Output Code**	**Output values**
Producer (Pr)	$I_{Pr} = 0$	0	$O_{Pr} = r * D_{Pr} * V_P$ $O_{Pr} = 1 * 0.92 * 100$	46
Aggregator (Ag)	$I_{Ag} = 0$	0	$O_{Ag} = r * D_{Ag} * V_P$ $O_{Ag} = 1 * 0.05 * 100$	2.5
Relayer (Re)	$I_{Re} = 0$	0	$O_{Re} = r * D_{Re} * V_P$ $O_{Re} = 1 * 0.03 * 100$	1.5
Consumer (Cn)	$I_{Cn} = V_P$ $I_{Cn} = 100$	100	$O_{Cn} = V_P - O_{Pr} - O_{Ag} - O_{Re}$ $O_{Cn} = 0$	50
Collector (Cl)	$I_{Cl} = 0$	0	$O_{Cl} = 0$	-
Third Party (3P)	$I_{3P} = 0$	0	$O_{3P} = 0$	-
Gas fees (G_F)*				
	Total Input ($\mathbf{I_{Tot}}$)	**100**	**Total Output ($\mathbf{O_{Tot}}$)**	**100**