White Paper

Version 2020-Jan-15

BOTLabs GmbH
Berlin

Executive Summary

KILT is a simple protocol for creating, claiming, issuing, presenting and verifying digital credentials. In contrast to peer-to-peer solutions for this, KILT features self-sovereign data as well as revocable credentials using blockchain technology.

KILT was built to be a business enabler, not only for the software industry, but also for any entity, which has or wishes to establish a business model based on trust.

KILT features a simple mechanism to describe and publish the content of a claim or the corresponding credential.

KILT provides a JavaScript SDK, which makes it very accessible for developers.

We believe KILT will be an essential building block of the Web 3.0. In particular KILT proposes:

- A **universal blockchain protocol** for individuals, organisations, objects, and artificial intelligences to claim arbitrary attributes about themselves and get them attested by trusted entities.
- A **Trust Market** for the Attesters of such claims, which allows trusted entities to attach prices to their valuable attestation work and get paid.
- Mechanisms for putting **claim holders in control** of their data by storing the information on their storage and by giving them the choice which information they want to disclose to whom.

Our main goal with KILT is to generate a level playing field for companies to explore new business models, related to trust relationships and data sovereignty. With our proposed system we would enable businesses and governments to rely on a common standard which is owned by everyone participating and not by a single company.

The white paper is structured as follows:

Chapter 1 describes trust in the internet and the problems and approaches around this topic in general and for KILT. If you are deep into trust, blockchain, and internet technology, you might want to skip the beginning of this chapter and start with the Conclusions and the Solution Statement resulting from them. The description of the whole **concept** behind KILT Protocol is divided into three chapters. Chapter 2 (Top-Down Trust Structures in KILT) introduces the fundamental and characteristic feature set of the protocol. Chapter 4 discusses the need for Claim Standardisation and describes the concept of CTYPEs. Chapter 5 describes how KILT Protocol would implement bottom-up trust with Token-Curated Attesters (TCAs). These three chapters should be read by anyone who wants to use or understand the concepts behind KILT.

We describe the planned Trust Market **economy** in Chapter 3 and we outline an envisioned token economy for the KILT token in Chapter 6. If you are less interested in the economic implications of KILT Protocol you might choose to skip these chapters.

The details of the **technical implementation** and the SDK of KILT Protocol are described in Chapter 7 and our roadmap plans for launching the KILT Network in Chapter 8. Chapter 7 is highly technological and addresses developers who plan to build services and applications on top of KILT Protocol.

Main Changes

Compared to previous version (2019-05-28)

- Executive summary
- 1.3 Where We See the Opportunity Through Blockchain?
- 1.4 Conclusion
- 1.5 Solution Statement
- 3.2 Participant Views and Economic Benefits
- 4.3 Incentivising Standardisation
- 5. Bottom-Up Trust: Token-Curated Attester (TCA)
- 6. KILT Token Economy
- 7. System Architecture
    - Nested CTYPEs
    - Quote
- 8.3 Release Roadmap
- GDPR Considerations

Previous version:
2019-05-28 https://ipfs.io/ipfs/QmbQytoeFev8h67gD7QSVo2EqiUD3XVGTUj7hXVjRkjDjG

Table of Contents

1. Why the Internet Needs a Trust Network

It is widely accepted that the internet needs a distributed online trust framework to solve overarching problems of representing trust among participants all around the world who are generally unknown to each other. We aim to solve these challenges by incorporating a network approach into the core functionality of the current internet.

1.1. Trust on the Internet

Ten years after the World Wide Web went online and the Internet has gone fully commercial and self-sustained, researchers have already judged that "of all the changes that are transforming the Internet, the loss of trust may be the most fundamental." In the light of the significant risks to fall prey to abuses by malicious actors, "the simple model of the early Internet – a group of mutually trusting users attached to a transparent network – is gone forever."[1]

Evolution of Trust through Cooperation

To understand why it is hard to establish trust in the internet's disembodied and spaceless interactions, it is essential to highlight how trust is tied up with cooperation. Somewhere along their evolution, humans have learned to leverage the superpowers of cooperation. We do things together we could not do alone and share the benefits amongst us. We invest by helping others, hoping to be supported later when we may need it. And we heavily and daily rely on information shared with us, where it is impossible to find out everything for ourselves. But while cooperation is of utmost importance for our modern way of life, it is also delicate and vulnerable to exploitation: in fact, many are puzzled that cooperation could occur at all in nature, given that it may likely break down as soon as a few malicious actors are introduced who take advantage of the other's support, hurting them in the process. In such an environment it would be rational for actors to cease cooperation in order to avoid being exploited and cheaters gaining advantage over them.

One of the reasons that we are still able to rely heavily on cooperation is that we have learned to decide when to accept the risk that comes with it: whom to trust and whom to refuse cooperation and thus sort out the bad apples. Trust is hence a filter or protection mechanism and an enabler at the same time, allowing us to take the necessary risks and "bridge the gap to the unknown"[2] – which basically means that trust enables us to explore and grow our abilities.

Additionally, trust may be defined as a state where subjective information can be reliably assumed to be objective. For example, Bob tells you that Alice is married. If you trust Bob,

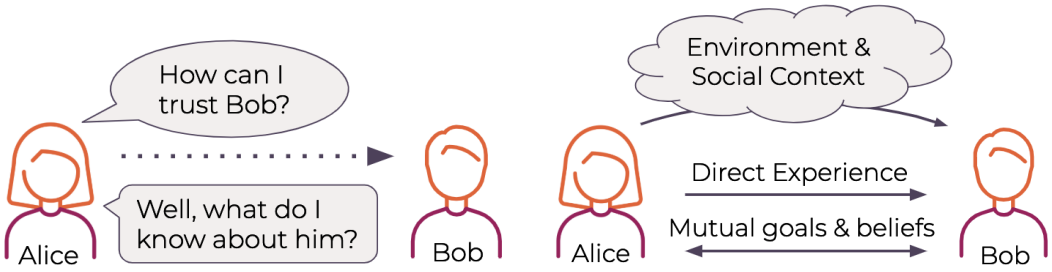
[1] Blumenthal, Marjory S. & Clark, David D. (2001): Rethinking the Design of the Internet: The End-to-End Arguments vs. the Brave New World, ACM Transactions on Internet Technology, 1: 70-109. [pdf]

[2] Rachel Botsman, The currency of the new economy is trust, TED talk, last accessed on 19th March 2019.

you will believe him and will not ask Alice out. If you don't trust Bob, then you might decide that Bob simply wants to eliminate a competitor[3].

Forming Trust Relationships

People form trust relationships with each other based on multiple factors, but it all boils down to the relevant information one has about the other person. Based on what Alice knows about Bob, she constructs her argument that relates to the trust she puts in Bob. This argument may be constructed from multiple types of patterns[4], such as direct experience (e.g. she met him at a bar and has a personal impression of him) or mutual goals (both of them trying to adopt recycling in their common workspace) and beliefs (sharing political views). Additionally, there might be other patterns for Alice to consider which are rooted in environmental or social factors (e.g. reputation, authority, etc.).



Forming new trust relationships is based on the information we have about the other person.

Forming a new trust relationship is a distinct transaction between two (or more) parties and in some cases, these relationships are only used for one transaction (e.g. Alice buys a necklace from a friendly merchant at the local market while she is travelling in Peru). However, in many cases a trust relationship is built up and maintained through an iterative process[5] and is determined by the consecutive reciprocal behaviour of the participants (e.g. relationship between Alice and her new employee at her company). In these cases, as a general principle, one will grant trust to those people, who have already cooperated with her in the past, making it more likely that the relationship will be beneficial for her in the future[6].

Problems of Creating Online Trust

Trust seems hard to come by when interaction partners remain anonymous or pseudonymous and may as well be living on the opposite side of the globe. This challenge is also a direct consequence of the dramatic growth in number and cultural diversity of people and businesses connected to the internet. In face-to-face communication people have an abundance of information at their disposal on which to judge the trustworthiness of others: from a first, superficial impression to many little cues such as facial expressions, posture, or emotions

[3] Bulkin (2018): Curate This: Token Curated Registries That Don't Work, as seen on 10th December 2019.

[4] For a more in-depth analysis and discussion about the patterns for constructing trust relationships see Parsons et al. (2014): Argument schemes for reasoning about trust, Argument & Computation 5: 160-190.

[5] Both one-time and iterative trust relationships can be simply modelled with the Prisoner's Dilemma game theoretic model.

[6] Heintz, C.; Karabegovic, M. & Molnar, A. (2016): The Co-evolution of Honesty and Strategic Vigilance, Frontiers in Psychology 7: 1503.

conveyed in the way they speak. On the internet, little to nothing is revealed about the people we interact with and what is disclosed might just as well be untrue. The problem to establish trust online, then, appears to come down to an accountability problem on one hand and a bootstrapping problem on the other. These issues can be analysed when turning to currently existing implementations that aim to provide solutions for building trust online on a secure communication substrate.

Accountability

The fact that online interactions frequently take place between geographically, culturally and socially remote individuals massively reduces leverage to hold trust partners accountable. Even more so, as individuals are often further shielded by aliases, under which online interactions usually take place. Seeking redress, personally or legally, may be close to impossible under these circumstances, and reputational damage is limited to the internet alias. Stakes are low for malicious actors to create new aliases, and this lowers trust in the system.

Bootstrapping

The accountability problem could be alleviated by disclosing personal information (such as a physical address) to the partner in a newly formed trust relationship. However, this process faces the problem that there is little reason to trust this information if the source is the individual in question. Any third parties vouching for the individual to back up the claims made about itself in turn face the same challenges to establish their own trustworthiness. This continues ad infinitum if no entity can be found that is 'close enough to home' (namely, a trusted root authority) to be held accountable, or in some other way able to credibly assert its trustworthiness.

Current Solution Proposals and their Limitations

The solution to these problems must build upon public key cryptography that a sender can use to securely communicate with a recipient without meeting and sharing a secret with each other. However, as the internet is an open communication channel, attackers may intercept communication by impersonating the designated recipient, convincing the sender to encrypt messages with a public key that in reality is part of the attacker's key pair. Therefore, trusted associations of a public key with the designated recipient of a message is essential for secure communication as a basic building block to build up online trust relationships. Different solutions based on public key cryptography have been put forward to solve these problems of online trust.

Public Key Infrastructure (PKI)

In a hierarchical *Public Key Infrastructure* (PKI) public keys are associated to a recipient by means of signed certificates issued by *Certificate Authorities (CAs)* or their delegates. Trusted CAs are selected by application (browser, email client, etc.) vendors in most cases and represent single points of failure in the system: if a CA can be convinced to sign a certificate associating an attacker's key with a service or website, any communication with that site can be intercepted.[7]

[7] For in depth discussion on the pitfalls of the current PKI system see: https://www.eff.org/deeplinks/2010/03/researchers-reveal-likelihood-governments-fake-ssl,

PGP Web of Trust

The *Pretty Good Privacy* (PGP) *Web of Trust* (WoT) is an orthogonal solution to the above. It employs direct peer-to-peer trust and customisable vetting schemes instead of trusted authorities, leaving it to the user to decide whom to trust and from whom to accept referrals. But for numerous reasons (limitations of the PGP certificate format, a nightmarish usability of the key signing process, barrier is high to introduce a new key association as a trusted identity, dependency on key servers for revocations, etc.) the Web of Trust has not found sufficient adoption to gain traction. A critical point appears to be that just as PKI forced users to adopt a hierarchical trust model, PGP forced users to adopt a peer-to-peer key signing party model, however the code would also be able to support hierarchical trust models.[8] Trusted authorities are part of many use cases, especially in corporate or governmental organisations or when the attribute to be verified is established only through the authority or expertise of a select few. Additionally, the PGP WoT did not account for the fact that different users/entities may be qualified to verify only some traits and not others, meaning they should also only be trusted to sign specific types of information. The PGP WoT does not offer a straightforward solution to making these limitations.

Reputation Platforms

Many web platforms implement reputation systems as a 'wisdom of the crowd' approach to establishing user's trustworthiness to solve the bootstrapping problem. These systems gather a collective opinion in order to build trust between users of an online community. Reputation systems recreate what people do offline to establish who is trustworthy: gossip, exchange experiences, give recommendations, etc.

These, however, only appear to work as part of larger community management schemes curated by the platform provider. The curation may include bans of accounts that spam the reputation system or that have been reported by users, as well as the verification of personal details of users by the platform provider. This comes as no surprise: if affirmations made by an anonymous user cannot be trusted without further support by a trusted third party, why should an anonymous crowd stipulate any more trust? As a pitfall, most reputation systems do not allow users to filter who they deem a credible judge of trustworthiness and in which matters. Namely, these platforms are often so general that they offer little information in what domain the rated user can be trusted.
Some proposals aim to solve these issues through Token-Curated Registries (TCRs). We address these in Chapter 5, where we introduce our Token-Curated Attester concept.

https://www.wired.com/2010/03/packet-forensics/, as seen on 15th March 2019.

[8] Berners-Lee, Why did the PGP Web of Trust fail? https://medium.com/@bblfish/what-are-the-failings-of-pgp-web-of-trust-958e1f62e5b7, as seen on 15th March 2019.

1.2. Arguments for a Network Solution

We believe that an argument for a network solution can be made from contrasting the dynamics and challenges the platform web is currently facing with the multi-stakeholder government of the internet's core protocols. Lessons drawn from these considerations will motivate KILT's solution approach, as we believe that a network solution can remedy some of these difficulties.

End-to-end Principle

When the core protocols of the internet and the World Wide Web, such as TCP/IP, were first conceived and implemented, many of the design decisions made were guided by what we know today as the *end-to-end principle*. Its rationale is to develop minimalistic protocols that focus on efficient transmission of data between two end points on the network, while remaining application-agnostic and indifferent to the specific types of data they exchange. The *protocol layer* focuses on efficient data transmission *in* the network. However, *on* the network, *application-specific functionality* is implemented in the endpoints connected via the protocol layer. Keeping the core protocols lightweight is meant to improve efficiency and speed by avoiding overhead and potentially makes them more reliable and easier to upgrade. It also reduces the number of specifications that all parties need to agree on.

Standardisation for Interoperability

The language which we use is a fundamental aspect of cooperation. It allows for naming certain things and through words and sentences we are able to convey what we mean. The language is thus an objective standard which we can use to compare things and decide if they are truly the same or not. However, the proliferation of languages hindered the efficient interaction of people already during the early cultural evolution (take for example the Biblical story of the Tower of Babel). To alleviate the burden of friction and inefficiency of requiring translations of laws and employing a large number of interpreters, empires introduced standard languages[9] to streamline the everyday life of citizens (e.g. Latin in Roman empire). Current day example is the *de facto* standard language of English in aviation, science, etc.

As a major problem, current solutions lack a standardised language for managing trust relationships on the internet and this is paramount for the interoperability of trust networks.

Internet Governance

It is in fact not a trivial task to reach such an agreement among all peers in the network. Participants must agree on a common language to be able to exchange and propagate information on the network. The internet as a communications network is a typical coordination problem as discussed in economics and social science: incentives for all implicated parties are largely aligned and favour a cooperative solution. In this case, participants need to agree on a common standard in order to be able to harvest the benefits of global interconnection and interoperability. Dissonance can only arise through differing preferences as to what the common solution should be. But no single company or national government can oversee the

[9] Standard Language, https://en.wikipedia.org/wiki/Standard_language, as seen on 17th March 2019.

multitude of interconnected autonomous networks and systems that is the internet. In absence of a central authority able to legitimately issue binding standards and specifications, these can only develop and uphold if they are multilaterally backed by the network's peers.

In addition, these specifications are necessarily public domain, and individual payoffs for the significant efforts of developing or improving them are low by extension, as benefits are harvested by all participants in the network. Nevertheless, the internet does have working governance schemes in place. Currently, several bodies are tasked with the ongoing development of standards and specifications (non-exhaustive list):

- The Internet Protocol suite (TCP/IP, message routing and transport protocols) is maintained and developed by the *Internet Engineering Task Force* (**IETF**), which operates under the umbrella of the *Internet Society* (**ISOC**). Standards are published in the form of Requests For Comments (RFCs) through the ISOC. The ISOC has more than 100,000 organisational and individual members.
- The *Internet Corporation for Assigned Names and Numbers* (**ICANN**) oversees domain name (DNS) and IP address allocation. As a central body with administrative authority, for example controlling registration of Top-Level Domains, it has created controversy[10], amongst others, through its association with the US Ministry of Commerce.
- The *World Wide Web Consortium* (**W3C**) develops the core specifications for the World Wide Web such as HTML and its successors, CSS, XML, etc. Members are universities, businesses, non-profit organisations, governmental institutions, and individuals. Standards and specifications are published under the name of Recommendations.

Some Challenges of the Platform Web

Governance of the internet architecture is thus *distributed* and *consensus-based*. However, this mostly does not apply to functionality on the network, i.e. in the *application layer*. Many Web 2.0 services and applications have the sole function of being a mediator and facilitator, connecting individual users on a platform for the purpose of exchanging information or services. Functionality (e.g. storage of user data, search tools, rating/recommender systems, etc.) built around the specific purpose of a platform (e.g. social networking, ride sharing or private sales) may be regulated in part by the respective laws of countries in which the user and the platform provider reside and may indirectly be governed by the user base, as far as the platform provider's economy depends on it. But ultimately, the system for interaction remains proprietary and users typically have little redress when their specifics or the terms for their use change, while the majority of the economic value is captured by the platform providers.

The end-to-end design principles most certainly have a key role in the internet's success story, allowing it to cater for even unanticipated new applications. But it also has a role to play in why the internet, and particularly the World Wide Web, has not become the utopian, egalitarian

[10] by famously refusing systematically and over years to register the Top-Level Domains .halal and .islam, while at the same time registering the .sucks gTLD, which now targets trademark holders with exploitive schemes.

space many saw coming in its early days, but has instead produced winner-take-all markets and mega-corporations that have grown to the size and power of nation states.

By defining the internet as a transport medium and putting the place and the responsibility for any kind of more specific functionality in the edges, it has favoured individual solutions by individual actors even for very common and recurrent functionalities. For the Web, this initially showed with the first large scale service providers developing around discovery and directory, which gave rise to internet giants such as Yahoo and Google. It became particularly clear, however, when the Web transitioned from a retrieval system for mostly static displays and information to an interactive and individualised experience. The original internet protocols did not implement user identification and could not maintain state. Service providers thus came up with a wide range of solutions and formats to store session and user identification data and user-generated content[11].

The architectural decisions to build the Internet as a rather thin networking and transport layer of universally accepted protocols have certainly contributed to the current shape of the Web, which could be characterised as follows:

(A) *Service/Application providers silo user data*. Since different application providers use a variety of formats to store user data and content on their servers, along with differing technologies for user identification, data is usually very hard to integrate. At the same time, for social networks such as Facebook, Instagram, or Twitter, or sharing economy platforms such as Airbnb and Uber, the user base and its data are the most valuable assets. Thus, in the absence of incentives to share data and with major hurdles to overcome to do so, users are unable to use their profiles and data in any other context. This creates dependencies hindering competitors to enter the market.

(B) *A single party may control vital parts of the network's functionality.* Private corporations owning and developing a particular service and the respective endpoints means they control who can access it and who can build integrations or contribute to development. They are also free to shape the service at their whim, compelling them to follow their user's needs and wants only if that is in their best interest.

(C) *Innovation happens at the edges of the network.* Because value is captured in end point applications, incentives are high to invest in newer and better applications and technologies. Incentives to improve the networking and transport protocols are comparatively small, being further complicated by the need to coordinate the efforts of many parties with possibly conflicting and competing interests. The network thus mainly grows and matures by virtue of new services connecting to it.

[11] The stateless nature of HTTP and IP made the early server architecture of the web simple since the servers did not need to store and maintain information about the state of a client-server interaction.

1.3. Where we see the Opportunity Through Blockchain?

Why We Think Blockchain is the Solution

Blockchains embody a network solution to features which internet protocols previously could not implement. They are constituted by a network of participants implementing a single protocol to agree on a common representation of state. Blockchain enables decentralised data and identity management by employing a ledger that provides immutable single source of truth that all involved parties (Claimer, Attester, Verifier) can trust.

What is the Role of Blockchain-Technology

Blockchains have two properties that remedy the problems outlined above: (A) they are able to self-organise through incentivisation mechanisms, (B) they can, and to some degree must, implement distributed governance structures that give participants in the network parts of the power to decide on the shape and future of the common protocol.

Network Solution through Blockchain

Taken together, a network solution will, by implementing functionality in the network rather than in its edges, distribute both the responsibility for and power over the solution among stakeholders, while also sharing the spoils among them.

Service/Application providers cannot silo user data.

Data is stored or indexed publicly in a blockchain solution. If a given functionality is offered by a blockchain solution, no single party can monopolise the data it generates. It either remains with the data subjects, with proof on the chain, or is available for everyone. Anyone can build and connect new solutions using these data, preventing monopolistic market dynamics.

No single party may control vital parts of the network's functionality.

Permissionless open solutions are always governed by nodes and validators or by token holders and can be complemented with democratic governance structures that facilitate consensus on updates to the underlying protocols. Also, nobody can be shut out for political, competitive, or arbitrary reasons.

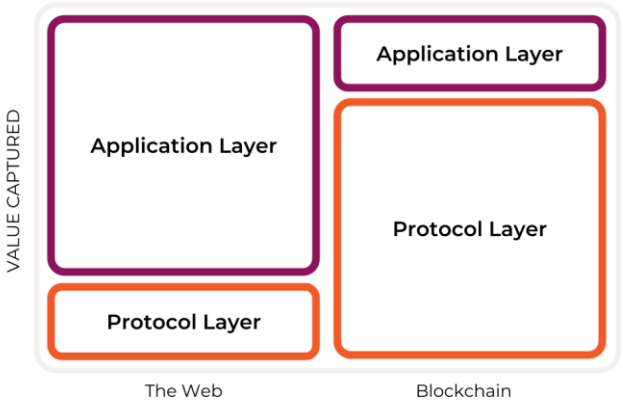
Innovation and value creation move to the protocol layer of the network.

Blockchains are incentive machines; part of the principles which they operate on is to reward participants for sustaining the network and furthering its development. They offer new ways to compensate contributors through block rewards or investment pools and create new incentives for protocol innovators (ICOs) which makes creating free-to-use open-source protocols economically attractive. Consequently, any holder of a token of a specific network invests in this network. If the network grows and the token value rises, the holder benefits. The value creation of blockchain networks are evenly distributed to all token holders instead of being siloed in a single company. This might be the most dramatic effect of blockchain networks: they democratise the profit of successful systems.

1.4. Conclusion

Value can be captured on the protocol layer.

The original internet lacks a series of concepts, which turned out to be necessary for the commercial internet. The three core concepts in this regard were search, identity and payment. Commercial companies stepped in, providing services for these concepts on application level. Unfortunately, the structure of the internet leads to monopolies and
this is why those companies today are the most valuable ones on the planet. While regulatory measures against these monopolies seem to have little effect, we believe that fostering the protocol level can solve the problem. Blockchain technology gives us the power to implement basic services like identity or payment on protocol level. Protocols are common good, as they belong to everyone and anyone is free to use them. The value, which was accumulated by the companies on the application level in the Web, could be accumulated directly in the protocol, with everyone who invests in the network by using its token benefitting from it.



Value capture, adapted from Union Square Ventures (www.usv.com)

Protocols can acquire economic superpowers.

When protocols are accepted as standard means of communication, they can provide investment security for new businesses. It is completely safe to build an email client based on SMTP, because it is absolutely sure that it will be able to communicate with all email servers worldwide. For the protocols of the internet, this security generated new business ideas and the possibility to invest. HTTP for example has created millions of jobs, most of them in businesses the creators of HTTP could not even imagine when they defined the standard.

Data belong to their producers and not to service providers.

Commercial companies collect siloed identities. They often know more about the consumer than the consumer himself. This is not only questionable in times of GDPR, it also leaves a 1984 feeling with the people. We shall store the properties of an entity at the entity itself on the end user's device or (encrypted) on a cloud service of the user's choice. The entity ultimately decides which part of its data, for which purpose, and with whom it wants to share it with.

Existing data silos should be made interoperable.

We understand that the existence of data silos will not end with a new protocol coming up. We want to incentivise owners of silos to make them interoperable through a protocol. This will provide more freedom and more convenience to the end user.

Standardisation could be a key to success.

Interoperability needs consensus on data formats. Credentials often constitute complex data structures. If we want to foster interoperability, we must enable the creation of standards for credentials. This standardisation will also provide investment security to companies which build applications for certain types of credentials.

Providing trust can turn into a relevant business model for many companies.

For any protocol adoption is key to success. Adoption will be high if commercial companies find business opportunities in using the protocol. It is necessary to provide business incentives to players in the trust ecosystem. We shall find ways to enable entities, which own trust or are able to build trust, to monetise this trust using the protocol.

Mapping of existing organisational structures is essential.

We recognise and accept existing trust structures even if trust is often not earned but defined by authority. These structures exist in companies, in governments, and in our daily lives. It would be futile to develop a solution which ignores these structures.

1.5. Solution Statement

In the following chapters we will outline KILT Protocol as we see it from today's perspective. KILT is a blockchain-based fat protocol.

The current state of KILT Protocol (Test Net) already includes the following functionalities in its basic, preliminary form:

- Enabling parties to claim arbitrary properties about themselves (where a party can be a person, an organisation or an object)
- Providing mechanisms to define the contents of such claims in a structured way
- Enabling trusted parties to select attractive claim structures and attesting claims of this type and issuing credentials to the claiming party
- Moving data sovereignty to the claiming party by giving it full control over the credential
- Providing a blockchain where the validity of a credential can be verified by anyone who the credential is presented to
- Offering mechanisms to build complex trust structures for authoritative (top-down) trust
- Decoupling the verification process from the Attester, creating huge scalability and privacy
- Solving the revocation problem of P2P Network approaches through blockchain technology.

In the future we would envision that KILT Protocol also

- Provides a Zero-Knowledge-based solution where the validity of a credential can be verified only by someone who was entitled to do this by the claiming party

- Builds an ecosystem where Attesters can monetise their earned trust but depend on their continuous accountability
- Provides a novel organisational structure to build earned trust by introducing the (bottom-up) Token-Curated Attester concept.

2. Top-Down Trust Structures in KILT

KILT Protocol aims to enable the digital representation of real-life trust relationships across all kinds of entities. In this chapter KILT Protocol is introduced by describing a set of basic roles and functionalities which represent a simple infrastructure that shall be used for building digital representations of conventional top-down trust structures.

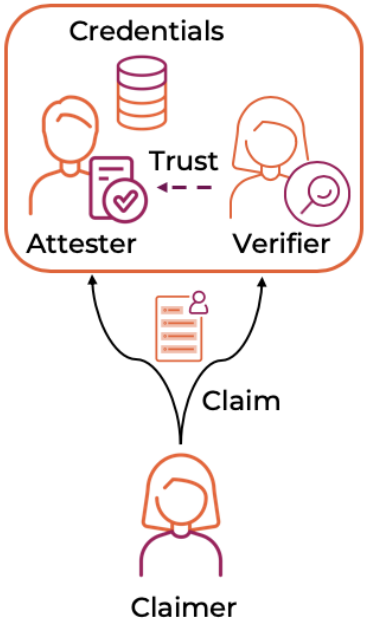
Top-down trust implies that there are entities which are trusted by other entities because of their status, their reputation, or just by definition. Trusted entities in this sense can be for example

- Government agencies, like the chamber of commerce
- Structures defined in companies, like the person responsible for issuing user rights within the company's network
- Commercial entities, like a company entitling a user to access a service
- Persons or other entities having earned a good reputation
- Machines or artificial intelligences designed to make decisions on claims from other entities, like a ticket vending machine in public transport

These trust structures exist in our current world. Therefore, a system dealing with trust needs to recognise them and must propose a way of representing them on a technological level.

2.1. Current Problems with Trust Structures on the Internet

Identity is the core function of trust relationships between physical individuals, devices, or any forms of legal entities like businesses, organisations, and governments. Identity in this context also refers to unique data that defines the properties, characteristics and qualities of persons, groups and things. With the rapid technological evolution and its consequences that we see for example in social networks, the use of artificial intelligence, autonomous vehicles on our doorstep, and the emerging decentralised governance structures, in short the digitalisation of all aspects of our daily lives, require a change of the fundamental trust structure of the internet.



Conventional trust model on the internet

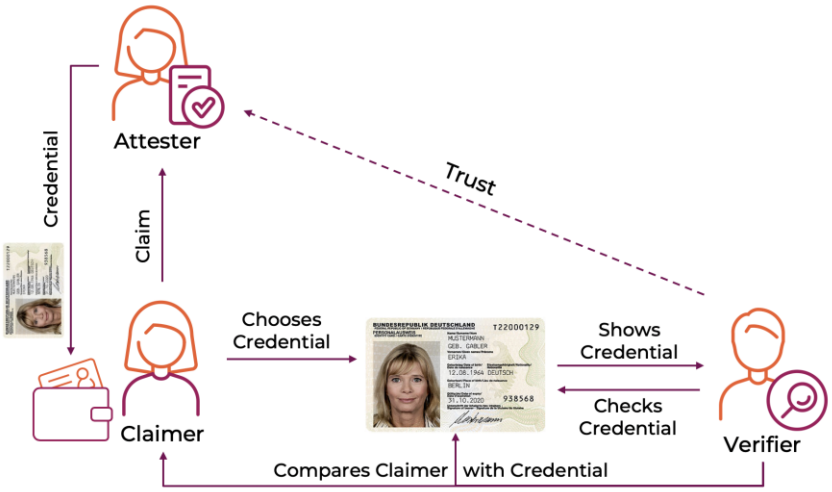
Right now these trust structures are dominated by centralised powerful service providers. For example, when a user wants to access a service, she normally registers by proposing a username and a password. If the username does not yet exist, the password meets certain criteria and some other checks are performed successfully, the user is granted access to the service. The username/password combination is stored in the database of the service. The same combination is also known to the user. Consequently, service and user share a secret. As many users undertake this procedure if the service is attractive and thus successful, the database accumulates a big number of username/password pairs. This is a security risk, as an attacker only needs to break into one system to gain access to a large number of secrets.

If the user wants to access the service for a second time, she can log in with her username/password combination. The service provider checks in its database if this combination is known and valid and decides on whether the user can use the service or not. This is problematic, because the issuer of the right to use the service and the verifier are the same entity. For commercial companies this is an incentive to build closed user groups, which leads to monopolistic structures if the service is very successful. In the internet such structures can be observed in certain sectors. They seem to hinder innovation because new providers, even if their services are better, cannot enter the market due to the monopolistic accumulation of users at the established provider. As a result, established providers dwell on large user bases and monopolies can only be replaced by new monopolies.
In most cases the username/password combination also allows entry to one specific service only, which results in a lot of passwords a user needs to remember. Many users choose similar or identical passwords for many services, which makes attacks on the central databases of a large service even more attractive. The username/password combinations stolen there might also work for other services.

The problem described here is even more astounding considering that it was solved in the real world many centuries ago by issuing credentials.

2.2. Current State in the Real World

Credentials in the real world can be best explained with simple examples: a person might have a passport, a driver's license, a student card, and many more. These Credentials are issued by central services. To obtain a Credential, the person (Claimer) has to turn to the central service (Attester). Unlike in the internet, the Credential then is not stored with the Attester, but with the Claimer, i.e. in the Claimer's wallet. If the Claimer for example wants to visit a bar where the minimum age is 18, she needs to prove to the bouncer (Verifier), that she is over 18. If this is the case, the Verifier will let her in. Unlike in the internet, the Claimer can choose which Credential she will present. The Verifier will probably accept her student card, her passport, and her driver's license. This provides freedom of choice.



The process of using of real-world Credentials

The Claimer may also choose not to reveal all the information on the Credential, for example by covering her name with her finger while showing the Credential. The bouncer needs to know her age but not her name.

The Verifier makes sure that the Credential is an original by checking its security features and whether the Claimer matches the Credential by comparing the picture on the Credential with the face of the person in front of him. This provides a high level of security (something I have + something I am). In the course of this, the Verifier also checks if he trusts the issuer (Attester) of the credential. If he trusts, for example, the government of the issuing country or the issuing university, he can also trust the Claimer that she is over 18.

This system is extremely scalable, as the Attester is not involved in the Verification process. The bouncer does not check with the university or the Department of Motor Vehicles if the person standing in front of him is over 18. The credential together with the person is enough. This also preserves a lot of privacy for the Claimer: as the Attester is not involved in the Verification process, the university or the state will never find out that she visited this bar that night.

2.3. Solution Statement: KILT Protocol

KILT Protocol basically applies the already proven real-world processes of issuing credentials and enables transparent and permissionless trust structures for the internet. In order to achieve this, KILT relies on the following simple concepts.

Self-Sovereign Data and Identity

Identity is the sum of characteristics, attributes and traits which describe an entity or an object and as an individuum distinguishes it from all others. Self-Sovereign Identity entails:

- any entity can establish its own identity (without the consent of any other entity).
- the Claimer is in control of her Credentials
  - can choose which Credentials to present to a specific Verifier
  - has the choice to show only the relevant info to a Verifier

These concepts for Self-Sovereign Identity can be used for any other data that is defining an entity or an object or that defines qualities or characteristics of such entity or object.

Self-Sovereign Identity

Self-Sovereign Identity is a model of digital identity where individuals and entities alike are in full control over central aspects of their digital identity, including their underlying encryption keys; creation, registration, and use of their decentralised identifiers or DIDs; and control over how their Credentials and related personal data is shared and used. SSI can be best understood as an infrastructural innovation that solves the interoperability and security problems of isolated and federated identity by facilitating a decentralised architecture for cryptographic roots of trust in a combination with Verifiable Credentials on the basis of encryption technologies, Distributed Ledger Technology, Open Standards and interoperable protocols. The architecture gives individuals and entities the power to directly control and manage their digital identity without the need to rely on external authorities.

KILT is a permissionless system, anyone (or anything) could create an identity or define qualities or characteristics of such entity and therefore become an entity in the KILT network. In KILT the identity (that may be connected to any piece of data) is created as public-private key pair and then linked to a Decentralised Identifier (DID)[12]. This DID is under full control of the entity which created it.

[12] Decentralised Identifiers (DIDs) v1.0, last accessed on 2020.01.15.

Roles

Any entity can take on any of the following three basic roles in the network.

Claimer

The **Claimer** is an entity (an individual, organisation, or object) that claims something (a property) about itself using its decentralised identifier and stores this **Claim** in its client wallet software (see Chapter 7 for technical details). In KILT the Claimer always signs the claim, which proves that the **creator** of the claim is the same entity as the **owner** of the claim (i.e. a Claimer is always the origin of the Claim)[13].

In KILT a Claim is always based on a Claim Type definition (**CTYPE**), which is a well-defined data structure expressing a specific set of properties and rules for a certain type of claims.

Claimers are not inherently trusted by Verifiers. They rely on Attesters and the trust those Attesters provide for their interaction with Verifiers. The Claimer can request an **Attester** to validate and confirm with his digital signature that the Claimer's Claim is true (i.e. to **attest** the claim). The attestation is issued by the Attester, sent to, and stored with the Claimer. We call these attested claims **Credentials.**
The Claimer may then **present** the credential to any **Verifier**. As the Credential is stored in the storage of the Claimer, the Claimer controls

- who to present the Credential to
- which Credential is presented and
- which parts of the content of the Credential she makes visible to the Verifier.

Attester

Attesters are trust providers who receive Claims, validate them, and confirm them by issuing attestations for them. In other words, they create Credentials by cryptographically signing the attestations and sending them back to the Claimers.

Any Attester will only be prepared to check and confirm a limited range of properties. According to these capabilities the Attester chooses one or more Claim Types (CTYPEs) offering to attest them for interested Claimers.

In this process Attesters perform actual work as they must check the validity of Claimer's assertions. Consequently, Attesters in KILT can choose to be compensated. In the current internet this compensation is done through the data the Attesters receive from the Claimers and store in their private data silos. In the KILT network the Credential remains with Claimer. KILT Protocol offers a mechanism for Claimers to pay for the work of Attesters by transferring KILT tokens to an Attester.

SSI Box: The definition is from the Blockchain Bundesverband's position paper about Self-Sovereign Identity.

[13] KILT protocol cannot control the content of the Claims and in theory a Claimer could claim something about someone else. There are cases where this might be needed (e.g. family status with a spouse) and we plan to address these use cases in KILT protocol as well.

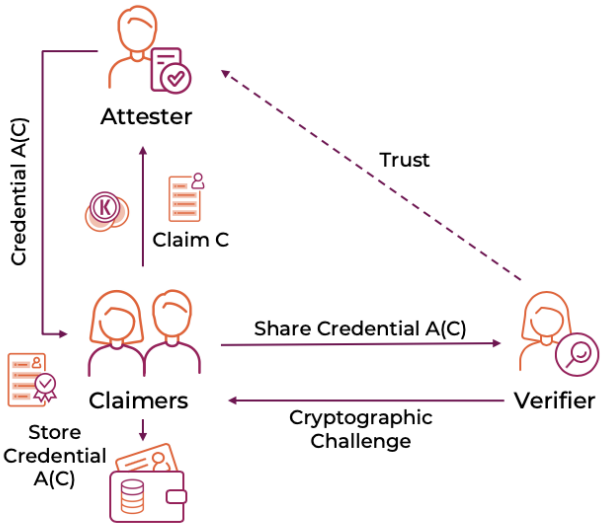
In order to receive or keep up reputation amongst Verifiers and Claimers, Attesters need to be careful to attest only Claims they checked and found to be valid. If they fail to do so, they might lose the trust of their Verifiers, which would make their Credentials worthless and could destroy their business. Attesters compete for the attestation of claims. The Claimers can select the Attester who best fits their needs (price, speed of attestation, supported Credential types, number and importance of Verifiers trusting their Credentials). This creates business opportunities for Attesters (see Chapter 3 for details).

Additionally, Attesters can amplify their trust by cooperating with other attesters or can delegate trust to other entities (see Section Building Top-Down Trust Structures for details).

Verifier

The Verifier creates the demand for attested Credentials as he offers a service to Claimers. As the Claimers are not inherently trusted by the Verifiers, the Verifiers rely on the Attesters and the trust they provide. For example when using a passport to cross a border, the Attester (issuing government) and the Verifier (border control agent of another country) are linked only by trust. In other words, the Verifier is an entity that receives Credentials from Claimers and performs an action desired by the Claimer if the Verifier recognises the Credential as valid. Verifiers decide which Attesters they trust. Claimers will select Attesters which are trusted by Verifiers and are relevant to the Claimer's use case.

Verifiers identify Claimers through a Cryptographic Challenge. This prohibits the unauthorised use of stolen Credentials and man-in-the-middle attacks. KILT uses Merkle Trees to ensure that Credentials can be validated against hashes retrieved from the KILT Blockchain, even if the Claimer only reveals parts of the information inside the Credential. The details of these features are explained in System Architecture.



Core processes in KILT

Roles and Processes

- The **Claimer** is an entity which states to have a certain property (i.e. **Claim**) and can request an Attestation.
- An **Attester** answers a **Request for Attestation** in an affirmative way, which is called **Attesting the Claim** (i.e. issuing a **Credential**).
- The **Verifier** is an entity which will perform, on request, a certain **Action** for a Claimer who shows proof of having certain properties (i.e. **Verifying** the **Credentials**).

Comparison to Current Standard Proposals and Definitions

There are several organisations and communities aiming to define the terminology and the different roles and processes in the Self-Sovereign Identity ecosystem. This chapter describes how the roles and processes on which KILT protocol shall be based differ from the current

W3C Verifiable Credentials (VC) Data Model[14] standard proposal. The W3C Model defines the following roles in comparison to the roles defined in KILT:

KILT	W3C Verifiable Credentials
Claimer	Subject and Holder
Attester	Issuer
Verifier	Verifier

The first and foremost difference is that the VC model differentiates between the Subject and the Holder of a Claim or Credential. **Subject** is the individual, entity, or thing that a given Claim or Credential is about or relates to. Contrary to KILT, the VC model allows a Claim or a Credential to be about multiple Subjects. In the VC Model it is assumed that the process starts with the Attester, proposing properties of Claimers, while KILT assumes that the Claimer starts the process by proposing properties of herself. While this might only be seen as a conceptual difference, we are of the opinion that Claimer entities should be the origin of these trust relationships.

Holder is the individual or entity in control of the digital wallet or agent that stores and controls the use of a given claim or credential about a Subject. Often the Holder and Subject will be the same entity, but there are cases where they may be different (e.g. a parent may be the Holder of a digital passport for their child who is the Subject of that credential). KILT currently does not distinguish between the Subject and the Holder of a claim or credential. Instead, both roles are unified in the role of the Claimer. We believe that the distinction of roles (Subject vs. Holder) could be solved on the application level and, if so, the KILT Protocol could stay lean. Since the standardisation process for this concept is under work, it will be followed closely and might result in a later adaption.

[14] The W3C Verifiable Claims Working Group states that the definition of the roles is still under discussion and therefore not final. https://w3c.github.io/vc-data-model/ as seen on 30th January 2019.

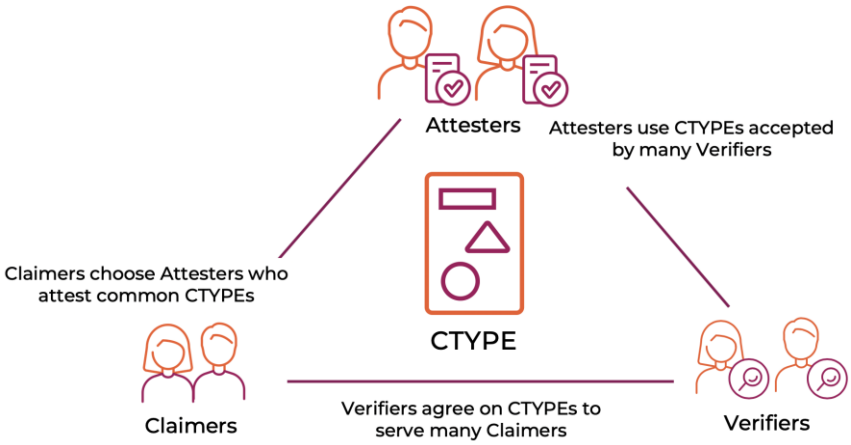
Claim Types (CTYPEs)

Enabling systems to communicate about properties of entities requires not only standardisation on *how* systems communicate, but also on *what* they communicate i.e. the content structure of Claims and Credentials. As it is impossible to standardise the content structure for an infinite number of use cases, KILT utilises the capabilities of blockchain technology to standardise Claim Types. While the standardisation mechanisms are discussed later in Chapter 4 & 5, Claim Types are briefly introduced here.

A Claim Type (CTYPE) in KILT is the JSON description of a data structure. It contains a list of key value pairs, where each value is of a defined type. Chapter 7 will further detail the rules on how CTYPEs are structured and how the KILT SDK offers functions for the creation and storage of CTYPEs.

Example of a CTYPE

Since it is essential that all parties in one use case agree on using the same CTYPE, KILT SDK functions for Claiming, Attesting and Verifying use CTYPEs as an underlying basic building block. Attesters publicly announce which CTYPEs they attest. They may also promote a list of Verifiers who accept their Credentials in order to receive more requests for Attestation. Claimers will use CTYPEs which are widely accepted by Verifiers they frequently use. Verifiers are interested in maximising the number of Claimers who can easily use their service. They will accept the most common CTYPEs for the use case. This leads to an implicit per-use-case standardisation. The creation of CTYPEs is permissionless. Anyone can create CTYPEs and reference them on the KILT Blockchain.



CTYPEs create investment security for all participants

Why KILT Needs a Blockchain

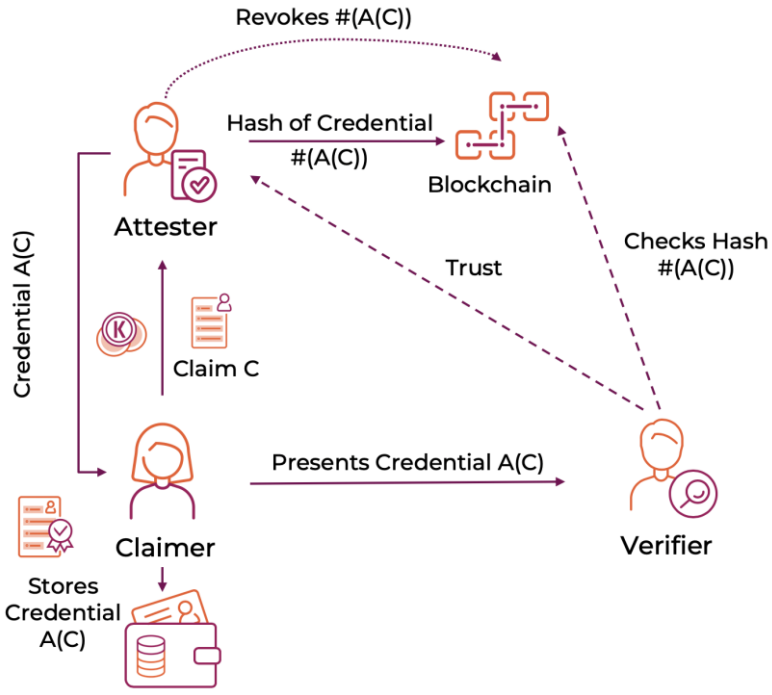
Blockchains provide an immutable yet decentralised log of transactions and we can use this superpower to enable revocable credentials and more complex trust structures in KILT. However, since the blockchain stores information indefinitely in a public manner, a great care has to be taken to conform to the latest privacy laws.

The operation and storage space of blockchains is very costly and they have limited throughput. KILT protocol honours these facts by writing as little information on the blockchain as possible. Since data written on the blockchain is publicly available and cannot be deleted, KILT aims to never write any personal data on the blockchain. Instead, only hash values[15] are stored on-chain (please see the GDPR section at the end of this chapter for an in-depth discussion on this topic.) There are basically three types of information written on the KILT Blockchain: the hash of revocable attestations of Claims (Credentials), the hash of claim type schemas (CTYPEs), and KILT token payment information.

Revocable Attestations

Putting the Claimer in control of her Credentials imposes a new problem: as the Attester is no longer involved in the Verification process, he cannot revoke the validity of Credentials. That would lead to all kinds of unsatisfying situations, where driver's licences could not be revoked and bullies could not be excluded from social networks. In addition, Attesters who once received the power to issue Credentials by an authority would be able to do so forever.

The KILT Blockchain solves this problem by writing a hash of the Attestation onto the blockchain. This way Attestations can be securely stored and time can be stamped on the KILT blockchain, which then fulfils a notary function and enables credential revocations.



Core processes in KILT using the blockchain

In case of an on-chain Attestation four pieces of information are written onto the blockchain:

- Public key of the Attester
- Hash of the Claim that is being attested
- Signature of the Attester
- Placeholder field to mark if the Attestation is revoked later

[15] Hash is a condensed, non-reversible cryptographic transformation of the original data.

Any Verifier who receives the resulting Credential from a Claimer can hash the Credential and check if the identical hash is also present on the blockchain in order to verify that action.
The Attester has the right to revoke the hash on the blockchain, thus marking it as invalid. Verifiers checking for the hash on the blockchain will find out about the revocation (for details please read about Revocation in Chapter 7). This process can also be applied to Legitimations and Delegations of trust, described later in the section Building Top-Down Trust Structures in KILT.

From a data security and privacy perspective, an Attestation contains hashed Claim data. Therefore, it doesn't reveal the Claim content (i.e. personal data is written onto the chain in a hashed format). The content of the Claim is not revealed and the Claim itself is not made available on the blockchain. If an entity knows the content of the Claim though, it can verify its existence on the blockchain. The public key of the Attester is stored on the chain. This is a wanted property which enables Verifiers to determine if the Attester of a Claim is trusted. In case of a simple on-chain attestation, only the Attester can revoke his or her attestation. In the next section, more complex trust structures are introduced where the right to revocation can be delegated to other entities.

CTYPE

CTYPEs are credential definition schemas needed for broad semantic interoperability of credentials. New CTYPEs will be created and added to the chain by entities which cannot find a suitable existing CTYPE for their use case.

Only the **hashes of the CTYPEs** are stored on the KILT Blockchain, while the whole CTYPE schema is published and stored in a registry service[16]. When an entity creates a new CTYPE, it generates a hash of the CTYPE (by hashing the schema structure) and requests to register this CTYPE to the KILT chain identified by this hash. This ensures that entities (Claimers, Attesters and Verifiers) can check if a Claim is conforming to its corresponding CTYPE by acquiring the CTYPE from a registry, hashing it and comparing it to the CTYPE hash on the blockchain. While storing the whole structure of a CTYPE on the blockchain would be a wasteful use of secure block space, this concept already creates investment security and interoperability amongst the participants of KILT protocol.

The KILT Blockchain stores the **public key of the entity that registers a new CTYPE**. It is assumed that the public keys of Attesters will be publicly known. In order to do business by issuing attestations, they will advertise their services which also means they will publish their public keys.

Payment Transactions

The KILT blockchain will contain its own cryptocurrency: the KILT Coin. This Coin shall be used to incentivise the security and continuous operation of the KILT network. Moreover, the KILT Coin shall be used to pay for the registration of CTYPEs, writing Attestations, and issuing revocations on the blockchain. The KILT Coin could also be used for paying the Attester for the Attestation service.[17] While these transactions as such do not incur privacy issues, since

[16] BOTLabs will provide a central CTYPE storage registry in the beginning, but anyone is free to run their own storage service and store any CTYPEs.

[17] Some of these transactions are already available in the Mash-net and are paid with the Mash Coin - therefore we often speak in general about the KILT token which would mean either the Mash Coin or

all these transactions will be publicly available on the blockchain, one could try to correlate a payment from the Claimer to the Attester right around the time the Attester writes the attestation on the blockchain. An attacker could determine that a Claimer received an attestation for a specific CTYPE (since the attestation contains the CTYPE). In certain cases, this might reveal sensitive information about the Claimer, even though the content of the Claim will never be revealed. However, it is under consideration that the KILT network might implement a zero-knowledge proof-based payment transactions system (e.g. like Zerocash[18]) in the future to solve this issue.

Quotes

A Quote enables Attesters to build structured contracts for the service of an Attestation. Further explanation can be found in Chapter 7: Quote.

Building Top-Down Trust Structures in KILT

Many current credential management systems rely on a model where organisations issue Credentials about entities. Hence, one organisation attests to various claims by entities. However, the level of trust is determined by the trustworthiness of the organisation attesting to the claims. This is a rigid and fragmented way of providing trusted attestations to the individual elements of one's complete identity.

As KILT shall be a permissionless network, anyone can become an Attester and provide attestations for any Claim. Removing the barrier to being part of the system and to embody any role is fundamental to the design of the KILT network. This implies that Claimers and Verifiers know which Attester to trust and which not to trust. In other words, the Verifiers decide to accept a Claim based on their trust in the Attester. The Verifiers' trust in an Attester can be achieved in different ways. The Verifier might directly trust the Attester. Alternatively, the Attester inherits trust through legitimation or delegation from another Attester and the Verifiers' trust in the Attester is derived from a specific trust structure, modelling real-world trust relationships.

The next sections describe different ways of how we envision that entities can create trust structures in the KILT network. First the principle of legitimation is explained. After that, two KILT-inherent mechanisms (Hierarchy of Trust and Private Curated Registries) are introduced which we want to manage on the KILT blockchain.

Legitimation

Authority by identity is the process of authorising a specific entity based on their identity. These processes typically ask the question: "Who are you?"

---

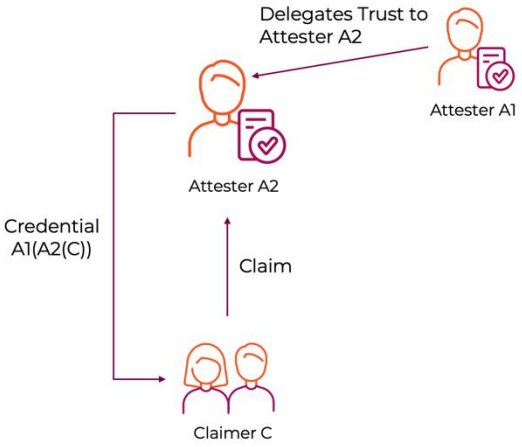
the Wash Coin or the KILT Coin (referring to the KILT token of the KILT main-net that we plan to call "Spirit-net") as they are available at a certain point of time. See Release Roadmap for details.

[18] Zerocash, https://z.cash/, last accessed on 17th February 2019.

Authority by possession is the process of giving access to a resource to any entity that possesses something, like a key. These processes typically ask the question: "Do you have a key that fits this lock?"

The concept of **Legitimation** in KILT is a simplified model of Object Capabilities for authorisation. A legitimation is a Credential by a trusted source that grants specific permissions (“capabilities”) to another entity.

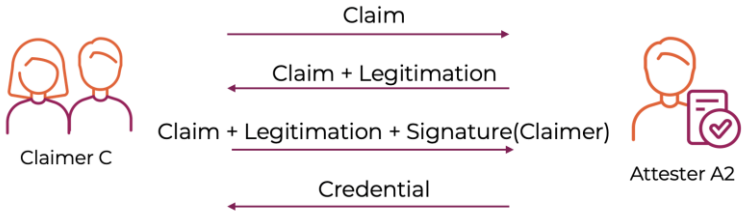
The simplest and most general example: Attester A1 legitimises A2 by attesting its arbitrary claim. Attester A2 then uses this attested claim (legitimation) to serve Claimer C. For example: the Chamber of Commerce (A1) attests that ACME (A2) is a company. When ACME (A2) attest claims related to its own business (e.g. attest employee status for Claimer C), it includes the Legitimation from the Chamber of Commerce (A1) into all of these claims.



Legitimation

In detail, this is what happens in KILT between Claimer and Attester A2:

- The Claimer sends her Claim to Attester A2.
- The Attester sends back the Claim and his legitimations for attesting this CTYPE.
- If the Claimer is satisfied with the legitimations, she signs Claim and legitimation and sends it to the Attester, signalling that she will accept (and pay for) the attestation.
- The Attester sends the Credential, which includes the legitimation and the Attester’s signature.



Request for Attestation with Legitimation

The above example is theoretically possible without using the KILT Blockchain. This case is called off-chain-legitimation. In case of off-chain legitimation(s), the **KILT Validator Node cannot check the legitimation chain, so it must shift this responsibility to the Claimer and the Verifier**.
In case of on-chain legitimations the Validator Node writes all attestations on the chain that are requested to be written on chain (e.g. if the attestation should be revocable). This should be the normal case.

When requesting an attestation, it is the Claimer's responsibility to check if the legitimation of the Attester is sound. It also is the Verifier's responsibility to check if the legitimation chain is complete (i.e. no link was revoked) when verifying a credential backed by a legitimation chain. The Claimer and the Verifier must be able to access all information that is required to make sure that the legitimation chain is valid. This information is included in all attested claims that are based on legitimations and the KILT protocol defines processes for such verifications (see Chapter 7 for details).

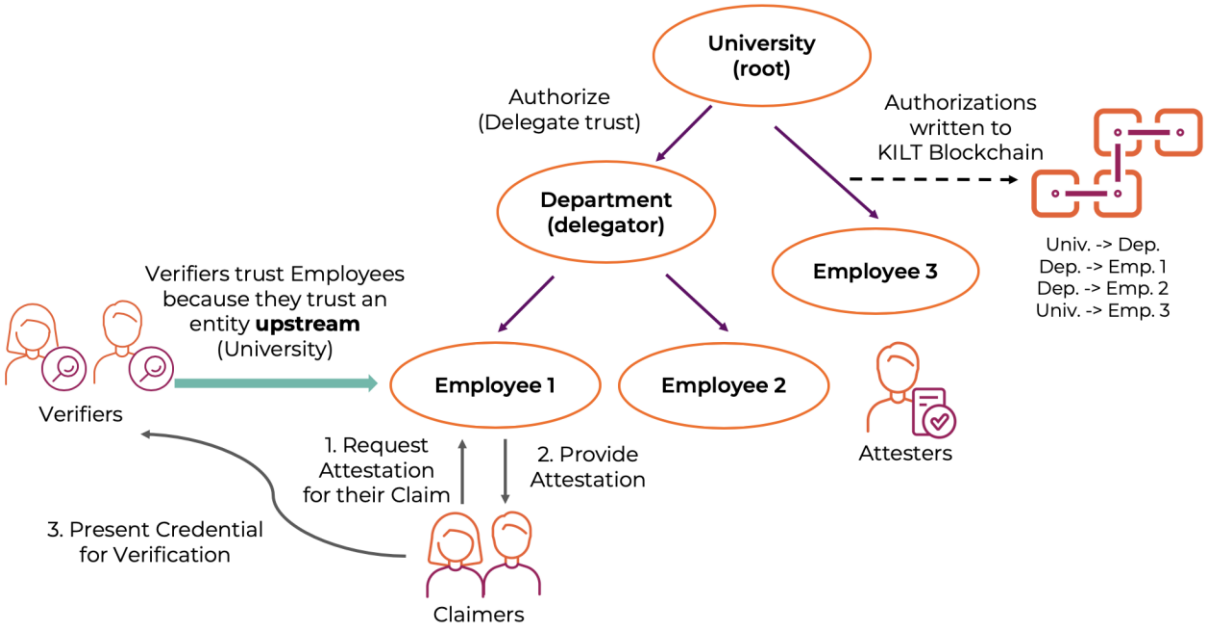
Hierarchy of Trust

A Hierarchy of Trust is a hierarchical top-down trust structure where the KILT Validator Nodes check the complete authorisation or delegation chain for attesting a specific CTYPE. In case of the Hierarchy of Trust, trust is distributed in a "traditional" manner from a trusted root entity to additional entities in the system. These models represent real world organisational structures of states, government agencies or the corporate world.

Adopting this concept for KILT means that a trusted root can delegate trust to other entities and build a Hierarchy of Trust around a specific use case (i.e. a CTYPE). Starting from the root, entities can delegate the right to issue attestations to Claimers for a certain CTYPE and also delegate the right to delegate the right to attest and to delegate further.

These authorised entities will be trusted by the community by virtue of the trust in a parent or root delegator. In other words, Attesters inherit the trust directly or indirectly from a trusted source or root. Anyone can build its own hierarchy which is essentially a directed acyclic graph (DAG) of delegated trust.

These structures are always stored and updated on the KILT blockchain (using delegation transactions described in Chapter 7: System Structure). The resulting delegation chains are always available for the Validator Nodes since they are stored on-chain. This means that the Validator Nodes write an attestation backed by delegation to the chain only if the delegation chain is valid at the time of attestation. This way the Verifier can be sure that the delegation chain was valid if the attestation is valid (in contrast to attestations backed by legitimations, where the Verifier has to check if the legitimation chain is valid).



Trust relationships between Claimers, Attesters and Verifiers in a Hierarchy of Trust

For example, Bob can claim that he has a university degree and an Attester, which in this case would probably be an employee of the university, checks his claim and issues an attestation for it. Now the Verifier might not know and trust the employee of the university. **The employee obtains trust when the leadership of the university delegates confidence to him through the organisational structure of the institution to attest to claims of university degree CTYPE.** When Bob applies for a job, the recruiter can easily verify his claim of having a university degree because the Validator Nodes did check the validity of the delegation chain, when creating the attestation.

As it can happen that an Employee loses the right to attest the specific CTYPE of the Hierarchy (e.g. leaves a company), this also needs to be reflected in the path of authorisation. In case the Department revokes the right of Employee 1 to attest the University Degree CTYPE, it issues a revocation of authorisation of Employee 1.

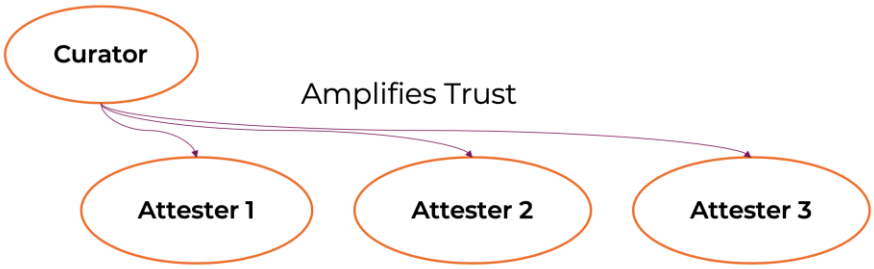
Should Employee 1 later try to issue an attestation for this CTYPE and try to forge the path of authorisation, the Validators will not write the attestation on the blockchain.
Also, the Claimer could present a forged off-chain attestation, but when the Verifier checks the authorisation chain on the KILT blockchain, he will not find it and therefore not trust the Credential.

Past attestations that happened before the revocation of authorisation remain valid as the time of the revocation is later than the time of the attestation. In a Hierarchy of Trust, revocations can be done by the Attester itself, or every entity upstream in the hierarchy than the Attester. The exact structural and implementation details of the Hierarchy of Trust is described later in Chapter 7.

Private Curated Registries (PCR)

Private Curated Registries are unordered permissioned lists that emulate rather traditional trust models where the Curator of the list has already built a trust relationship with Claimers

and Verifiers, e.g. established and well-known companies, government institutions, groups of individuals that already have a reputation in a specific area, etc. PCRs enable those entities to build business on the trust they accumulated, by sharing the trust with other Attesters and thus taking responsibility for their actions.



The Attesters belonging to a PCR gain their trust from the Curator of the list

A PCR serves one specific CTYPE and it is typically a top-down approach for establishing trust. A PCR puts the Curator in complete control of the list in regard to who gets accepted to or removed from the list as an Attester. From a technical perspective, a PCR is simply a flat Hierarchy of Trust where the Curator is the root entity. As an Attester of a PCR inherits trust from the PCR which he leverages to do Attestations, he must pay a fee to the Curator for every attestation he does that is based on the CTYPE of the PCR and that is utilising the PCR in its attestation. If an Attester is part of multiple PCRs, when he creates an Attestation, he clarifies in the attestation which PCR's trust he uses in that specific case.

The governance of PCR will be defined by the Curator. This will typically be a mutual agreement between the Curator and Attesters, where the offer or request can come from both sides (request to become listed, offer for becoming listed and giving revenue share in return). The curator of a PCR will be able to revoke Attestations by Attesters that are not anymore on PCR and acted on behalf of PCR. This might be done by Curators to protect the reputation of their PCR.

PCRs can also be utilised to avoid unwanted flows of money: a department which issues official documents does not want the fees to accumulate at the civil servants working (attesting) there. In this case the department would establish a PCR with a revenue share of 0% for the Attesters and 100% for the Curator. All Civil Servants attesting the CTYPE would then be invited to be Attesters in the PCR.

The foreseeable economic models behind PCRs are discussed in Chapter 3 and details of the PCR implementation are described later in Chapter 7.

3. KILT Trust Market Economy

In this chapter, we define what a Trust Market is. We elaborate on the economic aspects from the previous chapter and describe a possible future economy built on KILT. We discuss potential economic benefits for Attesters, Claimers and Verifiers. Eventually we discuss imaginable business opportunities for aggregators, Private Curated Registries (PCRs) and the Hierarchy of Trust.

3.1. Trust Market

Any opportunity to use trust for receiving rewards is considered as a business opportunity in KILT which all together make up the KILT Trust Market that we envision to create. Being in a position where one is trusted by others, one could perform attestations and receive payments for that. Trust could also be delegated in return for a revenue share of these payments in more complex trust structures as with PCRs. Eventually these PCRs shall amplify the trust of Attesters that are listed.

So, there are two ways trust could be marketed: First, trust that Attesters already have could be monetised. Therefore, Attesters would have to combine their level of trust with the work they perform - the attestations - which could then be monetised by charging the Claimers for attesting their claims. Second, trust could be delegated in return for rewards by different mechanisms like the hierarchy of trust and PCRs, which have been introduced in Chapter 2. The potential economic mechanism behind these structures shall be outlined in the following.

For marketing trust, it first must be obtained. There are two ways on how an Attester may obtain trust: from *outside* and/or from *within* the KILT ecosystem.

In the former, Attesters are trusted through their position in the social/organisational structure, e.g. trust in central authorities as it is the case in the traditional, real-life economy. Such trust level can be conveyed either through word-of-mouth or through a seal/certificate that is shown on the Attester's website. This seal or certificate can be thought of as an offline legitimation which is securely referencing the public key of the Attester. This is a very important step where users may relate a cryptographic identity used by KILT to the real-life Attester in question.

In the latter, trust may be delegated from one Attester (or root trusted entity) to another or amplified through network mechanisms. So, the delegation of trust for rewards is both a way to market trust but also to receive trust depending on the point of view.

In both cases users may check whether specific verifiers know and trust an Attester or root trusted entity. Once having a certain trust level, demand from Claimers for attestations of a specific Attester emerges (assuming the respective CTYPE is accepted by verifier). Since the history of (on-chain) attestations done by an Attester could factor into the trust level of the Attester, he can enhance the trust of other participants he enjoys through issuing appropriate attestations within the system. Having issued appropriate attestations that are accepted by various verifiers can lead to good reputation which is an indicator of trust.

From a technical perspective, trust is originated in the application layer and it is the protocol layer that enables to represent and manage the trust relationships. Thus, the trust marketplace

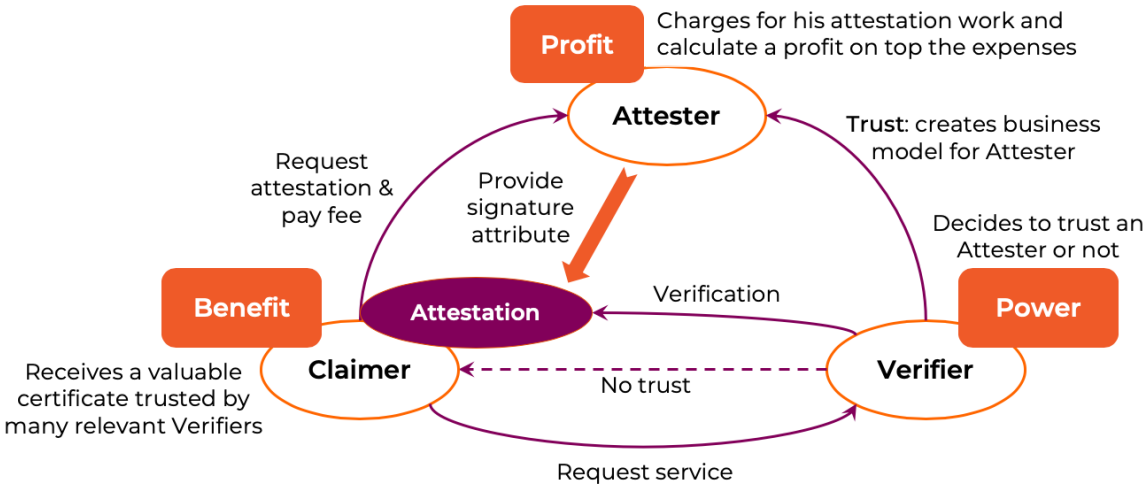
is not directly implemented on the KILT protocol but rather on the application layer where the KILT protocol facilitates the building of the application layer and its Trust Market.

3.2. Participant Views and Economic Benefits

This section considers the economic benefits for Claimers, Attesters and Verifiers in the KILT network.

Value Flow in KILT

The value flow diagram below shows simple exchanges between the Claimer, the Attester and the Verifier.

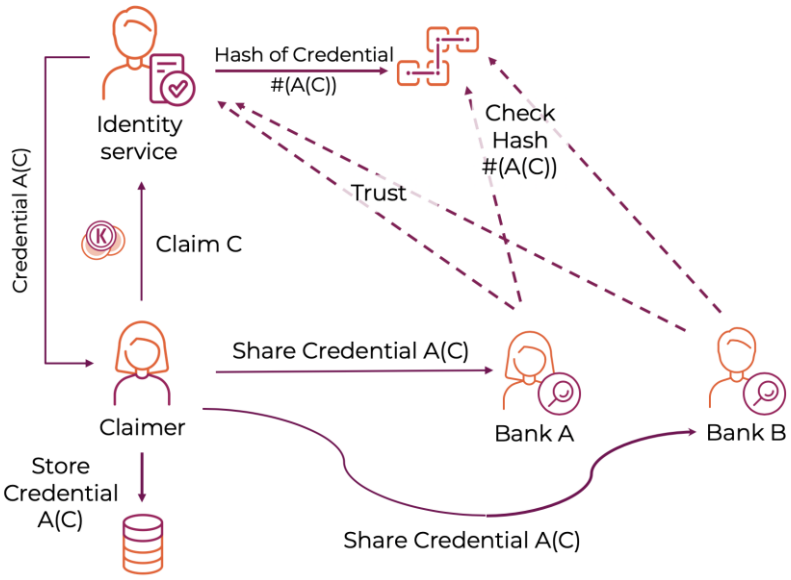


Value and cash flow in the Claiming – Attestation – Verification process

The whole attestation market is based upon the fact that the Verifier does not trust the Claimer. So, the Verifier needs a trustful attestation of the claims made by the Claimer. The Verifier is more likely to trust a third party with a good reputation because there is no direct conflict of interest. So, for the Verifier to trust the Claimer's claims, they must be attested. This is why the Claimer requests claim attests for which he pays a service fee. The attester in turn performs the attestation and signs the claim in case it is trustful. With this attested claim (i.e. Credential), the Claimer may approach the Verifier who trusts the attestation in case he trusts the Attester. This general concept is true for both, off-chain and on-chain attestations. However, on-chain attestations have the advantage of time stamping and revocation, so the Verifier can be sure that the attestation is still valid.

Claimer

The KILT network allows Claimers to create claims on the fly wherever they are, in a very convenient and easy manner, and transform them to Credentials. Apart from the general improvements in convenience and empowerment, there are specific use cases imaginable that would improve the user experience for the Claimer. A Credential hashed on the KILT blockchain could be more easily reused than traditional Credentials as we expect that an immutable proof of a Credential with a timestamp is valued far higher than a traditional offline Credential which can be faked more easily. So, there is a chance that a blockchain-based Credential created for one Verifier will be accepted by other Verifiers for the same purpose. For example, an identity authorisation through a Know Your Customer (KYC) process can be more easily reused at another service if stored on-chain since it might be more trusted by other banks. Currently, attestations must be done for every service provider (opening a bank account, applying for a credit card etc).



Reuse Credential for different Verifiers

Having the hash of a Credential stored on the blockchain, different Verifiers may check the same attestation for doing KYC. For a Verifier to accept a particular Credential, the Attester who issued the verification needs to be trusted. For example, a Claimer may reuse an identity verification he claimed for opening a bank account at Bank A for applying for a credit card from Bank B.

Storing the revocation state of the Credential on the blockchain allows to check the status independently of the Attester which is important in case the Attester closes his business or has downtime issues[19]. Generally, when Claimers have stored their Credential on their wallet, they may present the Credential to Verifiers including a reference to the hash and the revocation status as stored on the blockchain. As a result, Verifiers can be sure that the Credential is still valid if its state is not revoked as noted on the blockchain. So eventually Claimers are more self-sovereign over their Credentials.

[19] "Blockchain Could Make the Insurance Industry Much More Transparent" by Disparte, Harvard Business Review, 2011, last accessed on 2019.02.14

The KILT blockchain solution is bound to high levels of privacy since all Credentials are stored off-chain on the user's device and only the hash of the Credential is stored on the KILT blockchain. The Credential stored off-chain is recommended to be only accessible with a private key owned by the Claimer, so the data belongs to the Claimer which makes him self-sovereign over his data. As a result, the Claimer has the opportunity to sell his data to advertisers. As data is not burned in comparison to consumer products, the Claimer may sell the same data several times to different parties. Data is not oil but can be reused infinitely.

Attester

Attesters generally charge a fee for their service as they have to perform actual work in making sure that the claim is valid, e.g. check if someone really has a valid KYC Credential. However, a Claimer will only request an attestation which will be accepted later by Verifiers, since a Credential will be only accepted if the Attester is trustful in the eyes of the Verifiers. This has tremendous implications on the fees that can be charged which will be explained in this section.

Factors influencing the value of an attestation

- Number of Verifiers trusting the Attester
- Quality of Verifiers trusting the Attester
- Competition among Attesters
- Use case
- Reusability of attestation
- Speed of attestation
- Revocable attestation or not
- Validity time frame

Attesters derive value from being trusted by Verifiers. Both, the quantity of Verifiers that trust the Attester and the quality is important. This means that an Attester who is trusted solely by one Verifier, that has a huge demand for attestations, may make a lot of business and charge high attestation fees. This can be leveraged in a monopoly situation. So, the level of fees that can be charged are not only dependent on the demand side for attestations but also on the supply side. When supply is great and there is great competition among Attesters, then the fees will be lower according to common market mechanisms. When there is low competition for one specific claim type for which attestation is however demanded, then the Attester may charge high fees.

There are categories of Credentials where their values depend on the frequency the Claimer uses it, e.g. can be used for different use cases with different Verifiers as well as the importance of the use case, e.g. the importance of a credential of university degree that is needed for applying for a job. A cheap train ticket can be used only once which is why it is considered a low value Credential. There are high and low value Credentials as well as all the variations in between. Attesters in turn are only trusted for some claim types. Someone who has a high reputation in attesting restaurants is not automatically also trusted for attesting KYC related Claims. So, there is no absolute level of trust for an Attester but instead it depends on the use case. Attesters who do not have the expertise or trust for attesting certain claims may leverage other forms of trust such as their social media history to make business for low value Credentials like attesting social network memberships, community ratings or peer endorsements. A market for attestations of all kinds emerges, even for those of low value that serve as weak reputation signals such as account age or number of contacts (e.g. on LinkedIn). Anything that serves as a measure of trust can now be marketed in form of attestations. Another factor that influences the value of an attestation can be, for example, the speed of the attestation process, where a faster process *ceteris paribus* may charge a higher

fee. The validity time frame of an attestation also influences its value as well as whether the Credential is revocable or not.

The market for attestation may also expand in the following years since firms may be forced to have products attested due to competitive, social or legal reasons. This could make the profession of being an Attester more attractive. Attesters may also save costs and improve efficiency of attestations on the KILT network since an Attester may use already accepted CTYPEs for his own attestations and leverage the groundwork done on KILT. Eventually, an Attester's reputation might be amplified through listings (Private Curated Registry) or through obtaining credentials as an expert from a Token Curated Attester which could often be cheaper than the establishment of reputation through conventional marketing.

Verifier

Checking the validity of an attested claim reduces the Verifier's risks as data attested by a trusted entity is more likely to be reliable. This is especially interesting when Verifiers are searching for trust signals they can use to verify a claim of a customer or partner. KILT allows one to outsource the validation service to Attesters, so Verifiers don't have to make attestations themselves but instead focus on their core business. This increases efficiency and may reduce overall costs.

Revocable attestations improve the reliability of an attestation even more since the Verifier knows whether the attestation is still valid or not. So revocable attestations are more valuable to Verifiers since they reduce uncertainty in regard to potential changes in the validity of the claim. This increase in value for Verifiers automatically impacts the value of a revocable attestation for Claimers. Verifiers may even only accept revocable attestations, so they have more confidence in the validity of the attestation.

Generally, Verifiers are interested in the standardisation of data schemes for the Credentials (i.e. CTYPEs) to improve interoperability which is meant to be facilitated by the KILT Protocol. Improved standardisation leads to lower friction, lower costs and increased efficiency[20]. For verifying on-chain attestations, an application able to access the information on the blockchain is required, and the development of such software will be enabled by the KILT Protocol SDK (see Chapter 7 for details).

Example: Concept for the Food-net

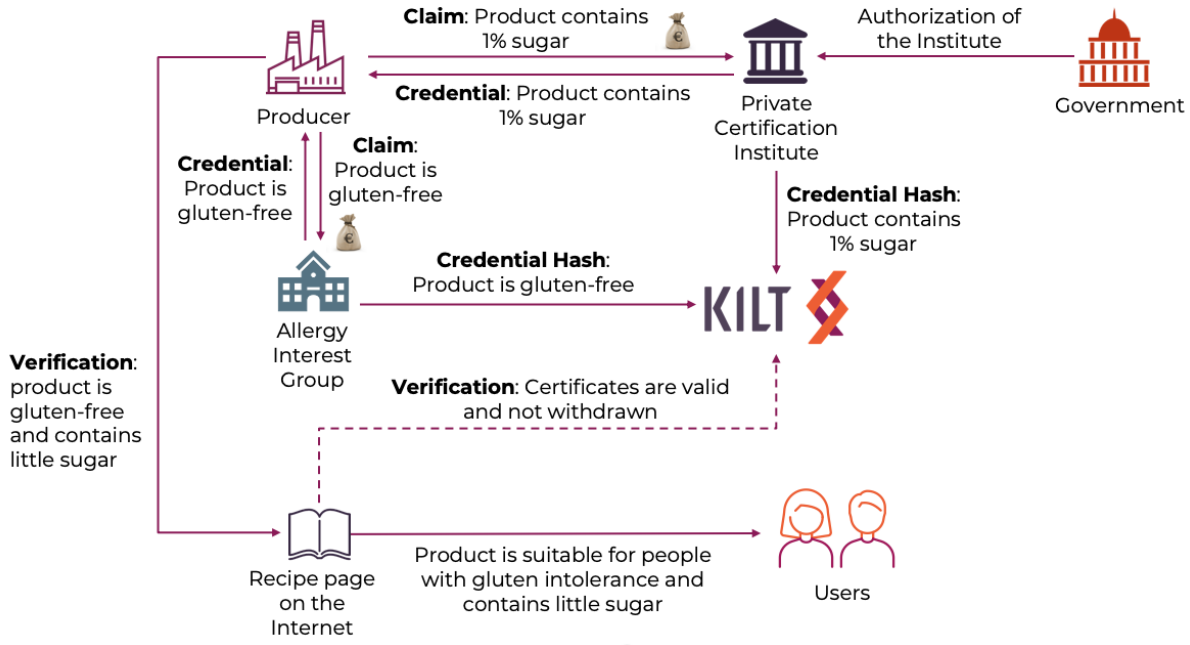
KILT supports the development of new business models by allowing Attesters to monetize the trust they carry, e.g. by issuing certificates that are relevant to their communities.
Taking a closer look at the food industry, there exists a lot of information about food products: ingredients, information about allergens, supply chain, if a product is well tolerated by people with a certain medical condition, etc. However, this information is not accessible to everyone, because it is stored in different databases that are partly permissioned and that are not compatible with each other. The KILT Protocol as a public permissionless blockchain enables food producers to get all the different kind of attributes of their products certified and to make these certificates available through different channels to the consumer.

[20] "Chain Reaction: How Blockchain Technology Might Transform Wholesale Insurance" by Mainelli & Manson, pwc, last accessed on 2019.02.14, https://www.pwc.lu/en/fintech/docs/pwc-how-blockchain-technology-might-transform-insurance.pdf

An estimated 7% of the German population suffers from a food allergy which, in the most severe cases, can result in a deadly allergic shock. This is one of the reasons why it would be of advantage to provide a register of food related information which is accessible for everyone and not siloed under the control of one or several companies.
Such information can be, for example, that the product contains 1% of sugar or is vegetarian. Additionally, potentially undesirable allergens can be declared, or production procedures explained. Producers let these claims be attested by trusted and authorized institutes and trusted organizations, which results in a certificate stored under the control of the producer. As the Attesters have to perform actual work to issue these certificates and are leveraging their trust, they will charge the producer for attesting their claims.
Trusted and authorized institutes might have the delegated power of the administration to issue “official” certificates, or this is their business model and they have a reputation in this field. Trusted organizations are organizations that carry the trust of certain communities (e.g. vegetarians, people with allergies, etc.), which makes the certificates they issue valuable to their respective community.

The certificates can be made available by the producers to various Verifiers. Verifiers can be the consumers that want to know certain information about the product, the supermarkets or aggregators like cooking websites and recipe apps, or organizations that serve a certain community (people with allergies, vegetarians, etc.).

- **Claimer** is any food (for example, identified by an EAN). For example, the manufacturer claims that his product contains 5% sugar, is nut-free and 100% vegan.
- **Attesters** are organisations that are trusted by consumers because of their work or position. These can be government agencies, but also the German Allergy and Asthma Association (DAAB) or the Association of Vegetarians. The attesters check the claims and certify them if necessary.
- **Verifiers** are both the consumers themselves and aggregators, such as recipe apps and cook websites. They provide the consumer with valuable additional information.



In Food-net different Attesters could attest different properties of a food

The advantages of such an open system would be:

For the **Claimer**:

- Certificates are made digitally available in a decentralized way and can be distributed in various ways (e.g. cooking magazines, allergy websites, etc.) which generates awareness.
- Attributes of the product get certified from independent entities that carry trust in a certain community.
- All different kind of attributes can get certified: the ingredients, if a product is vegetarian or if a product is well tolerated by people with a certain medical condition.
- The external verification leads to a high credibility.
- The certificate is under the control of the certificate holder.

For the **Attester**:

- New business models can be developed by monetizing the trust of a certain community.
- Gain of reach by aggregators applications and websites.
- Their apps can also show attestations of other Attesters, in order to increase the benefit for the consumer.

For the **Verifier**:

- Better data because there are a lot of different Attesters.
- They can use the trust they build up for becoming Attester themselves.

3.3. Economic Benefits in Trust Structures

This section examines the potential economic benefits for Aggregators and Attesters within complex trust structures, such as a Hierarchy of Trust as well as PCRs.

Aggregator

Aggregators may build an application that enables access to information attested on the KILT blockchain. An example for this could be a service for the collection and validation of claims about certain conditions or ingredients of food products, such as sugar content or allergens. In this case, the KILT network would allow to digitise offline trust such as the printing of seals on products by writing these Credentials on the KILT Blockchain. Providing trusted source and access to such information gives the company behind the product a competitive advantage, for which the Aggregator could charge these companies. Aggregators may also buy data from users, where the users can select what to share with them, and then sell it in bulk to big companies (e.g. health data for medical research, personal preference data for targeted advertising, etc.).

Hierarchy of Trust

Trust in an Attester may be inherited from another Attester in a hierarchy of trust as it was explained in Chapter 2. In a hierarchy of trust, nodes lower down in the hierarchy receive their trust from the entities higher up which is a top-down trust delegation approach. A simple case is the university degree certificate example in which those attesting degree certificates receive the power to do so from the central authority further up. The revenue streams of the different parties depend on the design which will differ in each use case. In the university degree example, all revenue generated from attestations flow to the root of the tree, whereas the Attesters will be paid by the root for performing their work. There might be solutions in which there is a revenue share, i.e. where Attesters receive a portion of the attestation fees directly. This concept of sharing revenue, however, is most prominent with Private Curated Registries.

Private Curated Registries (PCRs)

In PCRs, a highly trusted Curator amplifies the trust level of the rest of the Attesters that are listed in the registry. Since an Attester is listed in an environment with other high-quality Attesters, they can enhance each other's trust as well. This is valuable for Attesters since they can do more attestation or charge a higher fee which is in turn charged by the Curator of the PCR.

Creating Revenue

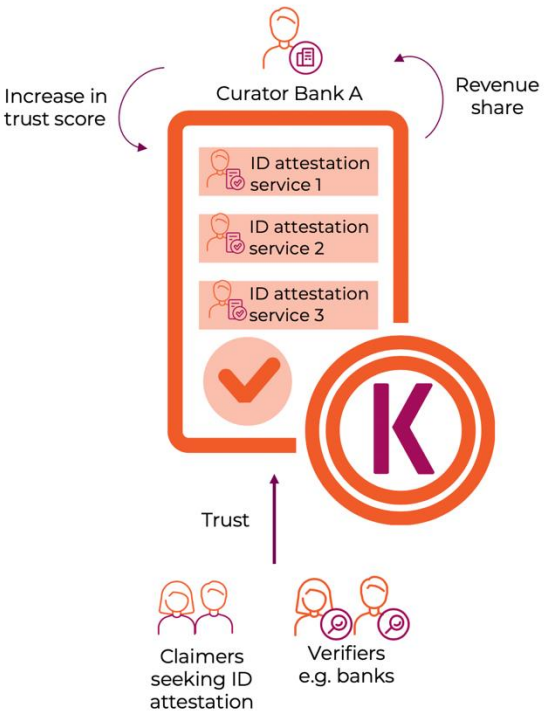
Anyone can start a PCR and become the only Curator of the list. This person defines the policy between the PCR members and the Curator. Generally, the Curator defines a revenue share between the PCR members and the Curator for participating in the trust of the PCR. Then, the Curator invites other participants to become a part of the PCR. Whoever wants to join the PCR needs to agree to the policy (e.g. the revenue share model) of the PCR. Instead of a proportional revenue share, the Curator may also decide to collect a fixed fee per attestation done by an Attester of the PCR. The Curator could also prefer a subscription (e.g. monthly/yearly) fee instead of a revenue share and he may also include a joining fee. In

summary, we can envision many different fee models and all kind of mixtures of these models. The level of fees a Curator may charge for a listing depends on the supply and demand to be listed on a PCR. This is influenced by the reputation of the PCR, the use case of the PCR, the quality of the PCR and other factors as listed above in the “Factors influencing the value of an attestation” box.

Concept of revenue share is highly dependent on the privacy aspects of token transfers in KILT. Assuming the Claimer pays attesters on-chain with KILT tokens, then there is a privacy issue, since the payment information is public and anyone may learn the owner of the Credential related to the attestation (see Chapter 2, “Payment Transactions” in section 2.3.4 for more details). However, information on attestations is necessary for enforcing the revenue share. If not stored on-chain, there must be another reliable information source on the attestations so the Attester may not cheat the Curator in reporting less attestations. Apart from the highly complex zero-knowledge token transfer concept discussed in section 2.3.4, a simple workaround to this problem is the implementation of a monthly fee instead of a revenue share.

Attestation from a PCR

A PCR may have received such a high level of trust, that verifiers trust attestations made by Attesters on the PCR independent of the Attester regarded. To leverage such a situation, we envision to allow the concept that a Claimer may merely submit a request for attestation from a PCR independent of the Attester. Such a high level of trust accelerates the value of being listed in a PCR which in turn influences how much the Curator may charge the members for being listed on a PCR. In any case, the Curator has an interest to maintain a high value listing as a decrease in reputation heavily impacts the percentage of revenue share that can be charged from Claimers. So, the Curator, has an incentive to only include appropriate Attesters. When a PCR has received such a high level of trust that members are trusted independent of their own reputation, the PCR becomes central to the decision of attestation whereas the reputation of the concrete Attester becomes irrelevant.



Example of a PCR

PCR Example 1: Know Your Customer (KYC) Process

In the following, an example for a Trust Market that emerges with a PCR is given: Bank A creates a PCR for ID attestation services. The bank aims to list which verification services the bank trusts for their KYC process. In the start of the creation of the PCR, Bank A has put great effort in checking every provider in great detail. Then, Bank A offered the ID attestation services that were found reliable the option of becoming listed in exchange for a revenue share, i.e. Bank A receives 1% of the revenue for every ID claim the ID attestation service

attested. Some ID attestation services did not agree on this schema whereas others did consent. Eventually the PCR became a list with top ID attestation services however without being collectively exhaustive. Other Verifiers may rely on the list to check which ID attestation services they trust for their KYC process. So essentially, this list amplifies the trust of listed ID attestation providers. An increase in trust *ceteris paribus* attracts new customers or enables the ID provider to charge higher fees for their service (with the risk of losing customers that are not willing to pay the price). In addition, the bank could market this PCR to other banks or organisations in need for ID services, so this PCR could be established as a brand.

PCR Example 2: Market for Used Cars

The following is a real-life example that could be mirrored on the blockchain to enhance the value of attestations. In Africa, used cars from western countries are ordered and sold whose conditions must be attested by trusted car mechanics. A large market for attesting used cars has emerged[21] where the attestation of some service companies is valued higher than others. An attestation by a high value service company increases the demand for the car which levers the price on the secondary market. Such service companies can be mirrored on the blockchain as PCRs with their car mechanic employees acting as Attesters and members of the PCR. In this case, the revenue share is at 100%, which means that all fees from attestations go directly to the curator of the PCR which is the service company. Storing attestations on the blockchain ensures that Attesters and Claimers cannot collude on the time of attestation since the hash is stored immutably on the blockchain with timestamp. The hashing mechanism makes sure that attestations are not faked as only the original document will match the hash on the blockchain. This improves the credibility of the attestation for Verifiers which then enhances the value of the attestation for Claimers.

[21] "Brief on the PCFV Used Vehicle Working Group" by Kamau, Partnership for Clean Fuels and Vehicles (PCFV), last accessed on 2019.02.14
"Importing Used or Salvage Vehicles from the United States into South Africa" by Richards, Auto Auction Mall, last accessed on 2019.02.14

4. Claim Standardisation

The KILT blockchain aims to create a structure and mechanisms where standardisation of claim types (CTYPEs) for various applications emerges. The KILT protocol places claim standardisation at the centre of the protocol by incentivising Attesters to use standardised claims for specific use cases. As a protocol that incentivises the attestation of standardised claims, KILT would enable businesses and governments to rely on common standards, which are owned by everyone participating in the network and not by a single company.

4.1. We Need Standardised Claims for Investment Security

Standardisation is one of the great achievements of modern society, it has many forms and is applied across all industries. Standardisation is the process of developing, promoting and possibly mandating compatible and interoperable technologies within a given industry. It maximises safety, repeatability and quality while fostering interaction beyond walled systems. The results of standardisation are pervasive in all aspects of everyday lives, from computers to driving cars, or switching on a light.

Diversity of Standardisation Processes

Standardisation is a joint effort of multiple types of communities to create a shared base for comparing and thus connecting various things. This process can take on different shapes in case of the different types of communities[22].

de jure standardisation

The first type approaches standardisation from an authoritative perspective, where a social body has been delegated to define a standard. This process is called a *de jure*, top-down, or committee-based standardisation. The standards are a result of a top to bottom process, with organisations or committees such as ISO (International Organization for Standardization), DIN (German Institute for Standardization), and IETF (Internet Engineering Task Force) that promote standardisation and endorse official standards for given applications. It usually entails a centralised decision-making process, and slow to adapt to changes in the real world. Since it is a well-defined process, usually the timeframe is known in advance and the outcome is predictable.

de facto standardisation

Standards can emerge as the result of public acceptance, when a custom or convention achieves a dominant position. This process requires that a community is involved and makes an impact by making use of, and engage with, or relying upon the standards in question. It is an open, market based or bottom-up process, which encompasses decentralised decision making. Most of these processes are undefined and the outcomes can be unpredictable. The results of these processes are called *de facto* standards: they are called standards based on

[22] Natalie Smolenski, Centripetal Standardization: Top-Down and Bottom-Up Vectors of Value Creation, https://medium.com/learning-machine-blog/centripetal-standardization-cc33e23a1acb, last accessed on February 25, 2019.

the fact that they are widely used. Prominent examples include the QWERTZ keyboard, MP3 audio compression format, and the PDF document format.

Competing Forces in the Standardisation Process

In real life standardisation processes, there is no hard separation between the two types of approaches, but they rather embody two ideal poles of the authoritative communities behind the process[22]. We set out to improve standardisation by building on top of the grassroots open source process yet reducing the inherent chaotic decision process and outcome. We envision that claims can express virtually any kind of relationship. Or said otherwise, most relationships can be modelled by one or many claims. This could translate in the following characteristics:

- Unpredictable use cases appearing which cannot be encompassed by centralised decision making
- Use cases evolving quickly and the standardised claims that meet their needs having to espouse their evolution
- Need for rapid consensus on the parameters of a claim
- Openness of the decision process, allowing all stakeholders to participate and voice their opinions

As claims can be about everything, their use cases will live at all levels. For claims about highly regulated qualities and/or qualifications, e.g. in transportation engineering and production closed and top-to-bottom standardisation processes make sense. But for most other types of claims, opening the standardisation process supports the emergence of living claim-standards. Attesters, verifiers, and any entity interested by the standardisation of a certain claim type can participate in the development of the standard. This is essential to a healthy claim system, which needs to encompass all use cases and invite all interested parties to the standardisation process. The KILT network builds up the supply of valuable CTYPEs so the demand may follow. This triggers a positive self-enforcing network effect through which all participants experience the benefits of standardised claims.

Investment Security

Standardisation can foster investment security, which we envision to be the main driver in our case. Verifiers and Attesters have to build software to attest and consume claims. Investing work, time and money into this software is only useful if the Verifier can read and understand the contents of the attested claim. That means, we have to create an ecosystem, where Claimers, Attesters and Verifiers mutually agree on the contents of claim types for certain use cases. Expressing qualities and qualifications about an entity is useful only if the language expressing this information is understood by all the interested parties.

Examples of claims where standardisation could provide huge benefits:

- Identity credentials: KYC claims that are required for the whole banking sector
- Diplomas and certifications: university degrees conforming to the same structure
- Access control for IoT devices: car claims it has access to parking

All these examples imply the existence of a Claimer making a claim, and a Verifier verifying that claim through its attestation (as we introduced in Chapter 2). Implicitly this means that there are existing processes that allow a Claimer to submit their claim for attestation. On the other side, a similar process exists to verify the attestation of a claim. The attestation and verification process, to be efficient, is designed to be automatic and require as little *ad hoc*

interventions as possible. This implies that the substrate should be well known and agreed upon by all partakers in order for the different processes to be built. Put differently, the protocol for managing trust relationships needs a content definition language for interoperability between the actors. Without this underlying standardisation of claims, attesters might find it difficult and possibly unprofitable to build such attestation systems, same can be hypothesised for verifiers. A claim protocol without standardisation of the claims is like having HTTP (transfer rules) without HTML (agreed schema for content).

Current approaches for claim standardisation are to use ontology systems[23] in order to automatize machine readability[24] for any type of use cases. While these approaches help bridging the gap between systems, they cannot fully provide investment security around claim types. In summary, the main problem is not machine readability but finding the right form for a given use case on which all participants of trust relationships can easily agree and build systems upon. In the following we describe CTYPEs as our proposed solution for the problem, which help converging on usable set of claim fields for a given use case. CTYPEs are open source by definition, and no one can collect license fees on them from others.

CTYPEs in the KILT Economy

Attesters, Verifiers, and any entity having an interest in the standardisation of certain claim types can participate in their development. This will translate into a market-driven emergence of claim standards. Once in use, these claim standards will evolve with the needs of the market, bringing them closer to the necessities of their use case. Overall, this will lessen friction and tension on the side of Claimers, Attesters, Verifiers, and industries, that will have the opportunity to participate in the standardisation process.

The industry demands standardisation to improve collaboration, interoperability and efficiency. The introduction of CTYPEs in the KILT protocol incentivises the industry to agree on a specific standardised form of a claim schema in a given use case. The more a CTYPE is used, the more valuable it is as using a CTYPE creates value through the network effect. The most often used CTYPEs automatically become a new standard according to the principle “standard by adoption”. CTYPEs in the KILT protocol serve as an alternative to ISO and similar standardisation organisations. This accelerates the establishment of standards as compared to the lengthy processes of central authorities. Interoperability and the establishment of meaningful standards is demanded which results in improved key performance indicators (KPIs) such as increased efficiency, cost reduction and improved convenience.

[23] RDF Vocabularies, https://www.w3.org/standards/techs/rdfvocabs#w3c_all, last accessed on 22th February 2019.
Schemas for structured data, https://schema.org/, last accessed on 22th February 2019.
[24] Eduard Hovy, Combining and standardizing large scale, practical ontologies for machine translation and other uses, In The First International Conference on Language Resources and Evaluation (LREC), pages 535-542, 1998

4.2. What is a Claim Type (CTYPE)?

The KILT protocol places claim standardisation at the centre of the protocol by enabling Attesters to use a certain claim schema for creating specific credentials. These standards for claims are defined by CTYPEs in the KILT protocol. In this section we explain how CTYPEs are generated, located and used within the KILT Protocol, elaborate on the structure and capabilities of CTYPEs and then we provide some possible CTYPE examples.

Basic Concept behind CTYPEs

As we already mentioned in Chapter 2, CTYPE (a mash of the words *claim* and *type*) is a well-defined structure expressing the properties and rules for certain types of claims. It defines how the content of a claim should look like. A CTYPE is similar to a struct in most programming languages: it is a generic definition of a data structure. CTYPEs are the means by which propositions for claim-standards are made. As anyone can create a CTYPE, there is a very low barrier to proposing a new standard for a specific use case.

Here we list a few fundamental properties about CTYPEs:

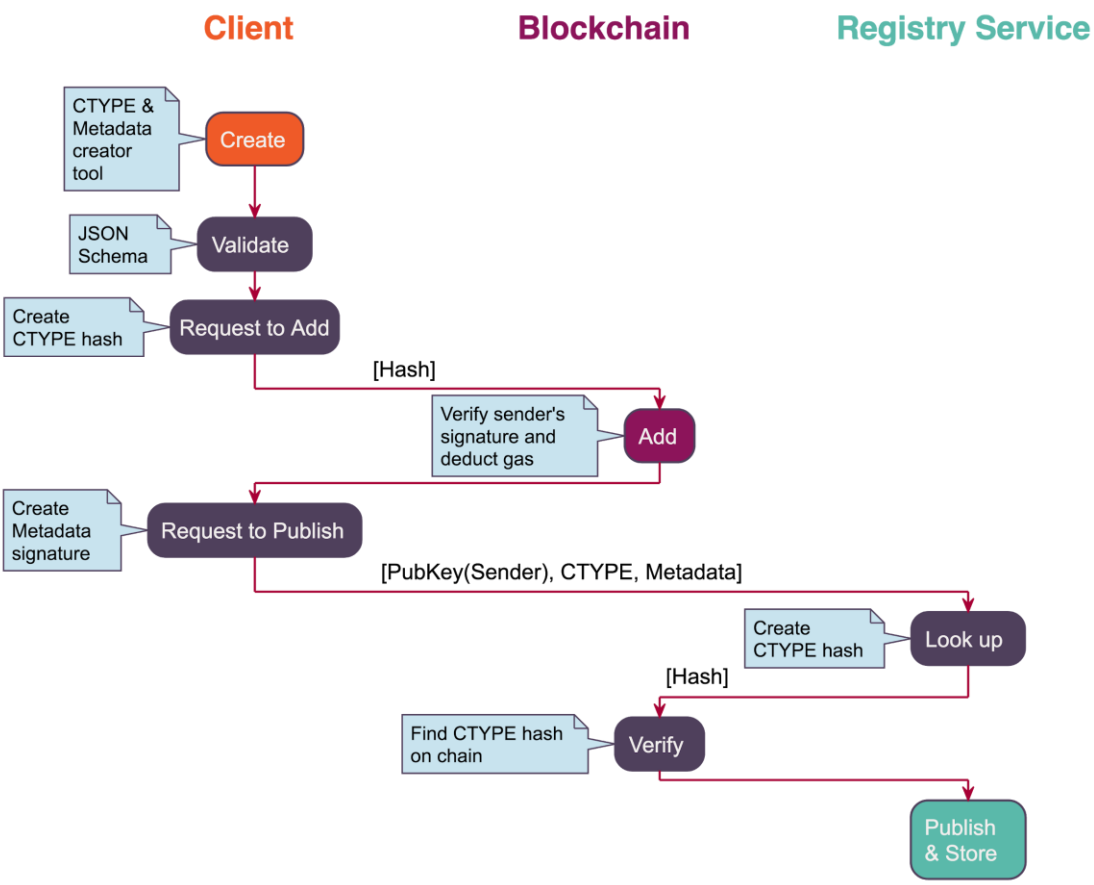
- Claims are always of a given CTYPE (it's their type)
- Anyone can create a CTYPE using a simple CTYPE builder utility or SDK
- A CTYPE defines the content and structure of a claim by containing relevant fields
- Hashes of all CTYPEs are added to (and registered through) the KILT blockchain
- CTYPEs are published and stored by the creator and/or in an open storage registry
- Anyone can use a CTYPE to create a new claim

CTYPE Metadata

CTYPE metadata contains additional information about the fields of a CTYPE. This will enable language translations for CTYPEs, and creators may include other user interface (UI) specific info, such as e.g. which type of control to display for a field (checkbox, toggle or radio button, etc.). In the end there could be even a complete HTML form built up by metadata. Metadata will be created but this part of the CTYPE will not be contained in the hash that identifies the CTYPE on the chain.

Creating and Storing CTYPEs

Creating and managing CTYPEs are cornerstone functions of the KILT protocol and the complete process is depicted on the following diagram. First, a KILT Client (user of a mobile wallet or a web-based interface) creates a CTYPE and its metadata (language translations, UI specific info, etc.) with the provided builder tool. Next, the Client generates the hash of the CTYPE, by hashing the core part (identifier and the structure, without metadata) and requests to add (register) this CTYPE to the KILT chain identified by the hash. The miners check the sender’s signature and deduct the transaction fee (gas or angel’s share) from the sender’s balance and add the hash of the CTYPE to the chain.



Creating a CTYPE: dark purple steps are provided as part of the KILT SDK

Now, the creator may publish her new CTYPE on a CTYPE Registry service. This service can be run locally on the client device or somewhere else on the internet. During this process, the creator first sends the full CTYPE, with its metadata and her public key to the registry. The registry then verifies the CTYPE by locally creating the hash and checking if it is already added to the chain (via a proxy blockchain node). If the CTYPE is present on the chain, then it is stored and published by the Registry service. From this point forward, the new CTYPE is accessible to everyone. Note that the Client, the Registry and the Blockchain communicate through a common KILT SDK (described later in Chapter 7), which can be used by other organisations to connect to KILT protocol.

Nested CTYPEs

Nested CTYPEs will enable one to create nested data structures for credentials. When creating new CTYPEs, any CTYPE can be referenced ("nested") into a more complex CTYPEs without duplicating the original CTYPE. Further explanation can be found in Creating a Nested CTYPE.

Services

There will be services (e.g. discovery interface), which list and promote CTYPEs, so that users and software developers can select the fitting CTYPE for their use case. To support the birth of an open ecosystem, the CTYPE usage will be free (other than the angels' share involved

with the necessary blockchain transactions) and the CTYPE schemas will be publicly available and open-source.

Benefits of Using CTYPEs

Claims can express all kinds of relationships; or the other way around, most relationships can be modelled by one or many claims. Because of their wide field of application, different types of claims will require different types of standardisation approaches: claims about highly regulated qualities and/or qualifications might require closed and top-down standardisation processes; for other claims, opening the standardisation process can support the emergence of living claim-standards. In the following we describe some context and simple use cases where standardised claims are particularly relevant.

Application Development

In closed systems (e.g. KYC process in a bank) the Attester and the Verifier belong to the same organisation. In these cases, standardisation can be made by definition. As soon as the system breaks its boundaries and aims to generate business beyond the traditional model (e.g. reusing the KYC Credential issued by Bank A at Bank B), application on the Attester side, the Claimer side and the Verifier side need to agree on the content of the claim.

When developers can base the development of their applications on underlying standards which are widely accepted and adopted, they gain security for their investments. For example, it was safe to invest money into building applications for email because with SMTP[25] there was an accepted standard for sending emails.

Usability

Claimers can use standardised, tested and already accepted structures to express their claims, lowering the barrier to make a claim. Attesters will also benefit from knowing beforehand what type of claim structure to expect and build attestation pipelines accordingly. For Verifiers, standardised claims will facilitate the verification process and increase security by reducing the amount of discovery needed when verifying a claim. All put together, standardised claims will drive usage.

Trust

Claimers will have more trust towards systems using well accepted CTYPEs and will be more willing to pay for attestations the use these CTYPEs. Verifiers on the other hand will know in advance what structure to expect for a given claim, increasing the trust they have in the claim.

4.3. Incentivising Standardisation

In earlier versions of the whitepaper we discussed incentive mechanisms for CTYPE standardisation. Research has shown that this approach is far too academic to promise wide adoption. We might pick up the standardisation incentives at a later stage.

[25] Simple Mail Transfer Protocol, https://tools.ietf.org/html/rfc821, last accessed on February 26th, 2019.

5. Bottom-Up Trust: Token-Curated Attester (TCA)

In Chapter 2 we explained top-down trust structures, but trust can be obtained in a "bottom-up" manner as well. In this case, Attesters acquire trust which is generated by the participants (users) of a system. This model resembles social networks like e.g. Facebook and online reputation systems like e.g. eBay, Amazon, StackExchange, etc.

However, current internet based bottom-up trust systems are in general hardly reliable. At Yelp, for example, everyone can rate a restaurant, but the preferences of a tester and of a consumer might not match and there is no structured system of criteria for a rating. Moreover, there is no quality assurance in the attestations, namely the reviews are not curated, and Attesters are not incentivised to be truthful or reliable.

KILT Protocol aims to solve the issues of the current internet based bottom-up trust systems by implementing Token-Curated Attester (TCA) structures, improving on the original Token-Curated Registry (TCR) ideas. Various forms of TCRs exist in the minds of a few developers and this topic is still currently under research. Therefore, this part of the White Paper is also subject to change on the basis of experiences of other projects and technical and legal feasibility. Nevertheless, we still wish to outline the basic functionalities as we envision them from today's perspective.

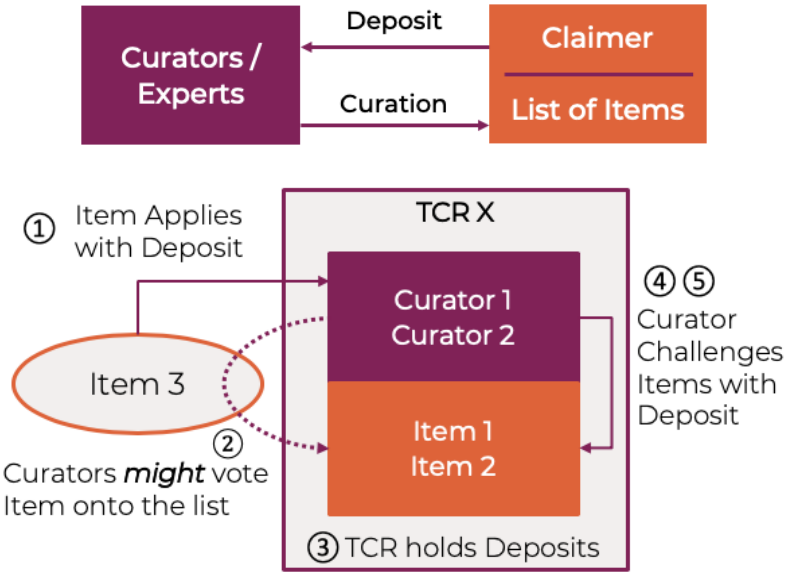
5.1. Comparing TCRs with Real World Organisations

Token-Curated Registries[26] (TCRs) are known as decentrally-curated lists with intrinsic economic incentives that, through the wisdom of the crowd, provide answers to specific questions we cannot directly verify. Aleksandr Bulkin elaborated[27] in-depth on the shortcomings of conventional TCRs and identified three criteria that must be fulfilled in order for a TCR to be successful. Due to human nature they can only answer questions if:

1) There is an objective answer to the particular question.
2) The answer is publicly observable.
3) The answer is very cheap to observe.

The fact that subjective TCRs do not work makes them unusable for most questions they were intended for. A simple example is the "List of great restaurants", which cannot be solved sufficiently through a TCR. In this example Curators gather to vote on restaurants, which apply to be on the list by stating their name and by paying a deposit. The information if a particular restaurant belongs on the list is clearly subjective. And though it is publicly available it is not cheap to observe. Voters actually need to go to the restaurant and try their food at least once. What will happen is that the highest bet has the best chances to win. On top of that, Curators are not necessarily good restaurant testers. Given these conditions the list becomes useless.

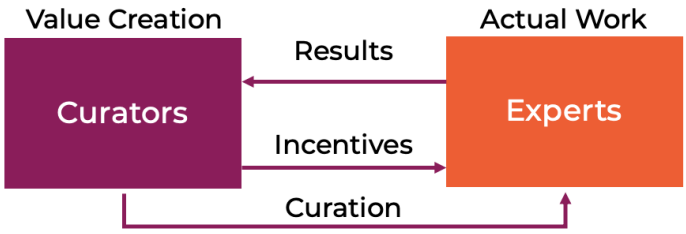
[26] "Token-Curated Registries 1.0" by Mike Goldin, last accessed on 2019.02.18.
[27] "Token-Curated Registries that don't work" by Aleksandr Bulkin,last accessed on 2019.10.25.



1. Item is de-incentivised through Deposit to apply (multiple times) to the list.
   → *keep the workload for the Curators low*
2. Curators will rather take the Deposit when uncertain if Item would increase the value of the list.
   → *keep the list clean*
3. Once on the list, the Item will take care not to be challenged, as it might loose its Deposit, which is locked in the TCR.
   → *keep the list valuable*
4. Curators are de-incentivised through own Deposit to randomly start Challenges.
   → *keep the list calm*
5. If Item's presence on the list damages the value of the list, Curators are incentivised to Challenge, because risk of loosing Deposit is lower than possible damage to value of the list
   → *keep the list clean #2*

Conventional TCR curation process

In the analogue world we observe a different economy-driven mechanism to solve complex subjective problems. Companies employ experts to make decisions. Good decisions have a positive effect on the company and thus enable the company to compensate the experts for their work. The added value of the actual company is the curation of a list of good experts who can maintain or even leverage the positive effect on the company by making the right decisions.

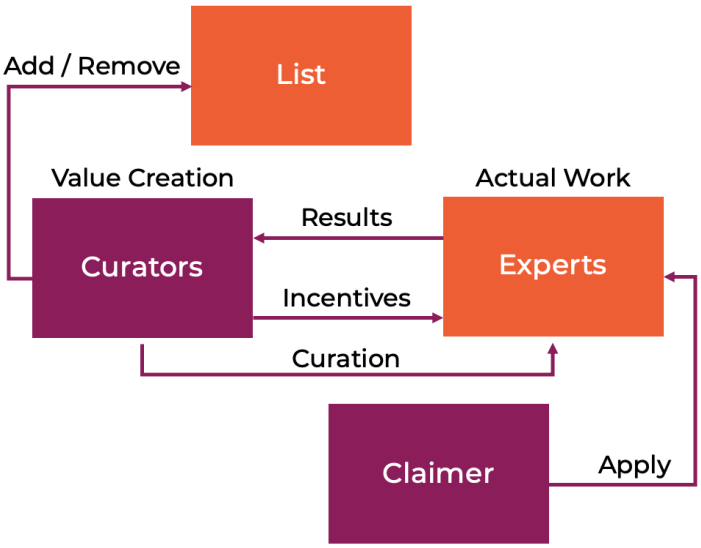


The Analogue World Approach

In this scenario the **Curators** resemble the management of a company or a brand. They increase the value of their organisation by curating a list of experts who work for them. These Experts are attracted by incentives, e.g. money or reputation, provided by the curator organisation. The better a curator organisation is, the greater incentives it can offer to the Experts.

A restaurant tester (**Expert**) will choose to work for a high-profile restaurant guide (i.e. curator organisation), because they will be able to pay him enough to pay the price for his services and the costs connected to them. Furthermore, his personal reputation will increase, working for a high-profile company. The Expert will also be careful in his decisions, because if he does a bad job, he will be expelled from the list by the Curators sooner or later, damaging his reputation and losing his income.

A restaurant (**Claimer**) claiming that it's the best spot for business lunch in Berlin would apply to or be chosen by the organisation of the Curators proactively. The Experts of that organisation would then execute their complex job of testing the restaurant. As they are paid by the Curators and risk losing their credibility and responsibility if they make decisions damaging the Curators' reputation, the likelihood of bribed decisions is low. Put differently, the Experts are disincentivised to accept bribes (possibly from a bad quality restaurant) over doing quality work as that could result in damaging the quality of the curator organisation and hence leading to the exclusion of such an Expert from the organisation. The Experts deliver the results of their work to the Curators, who would publish it under their company/brand name on the list. Good decisions by good Experts should increase the value of the list for the consumers and thus the value of the organisation maintained by the Curators should increase.

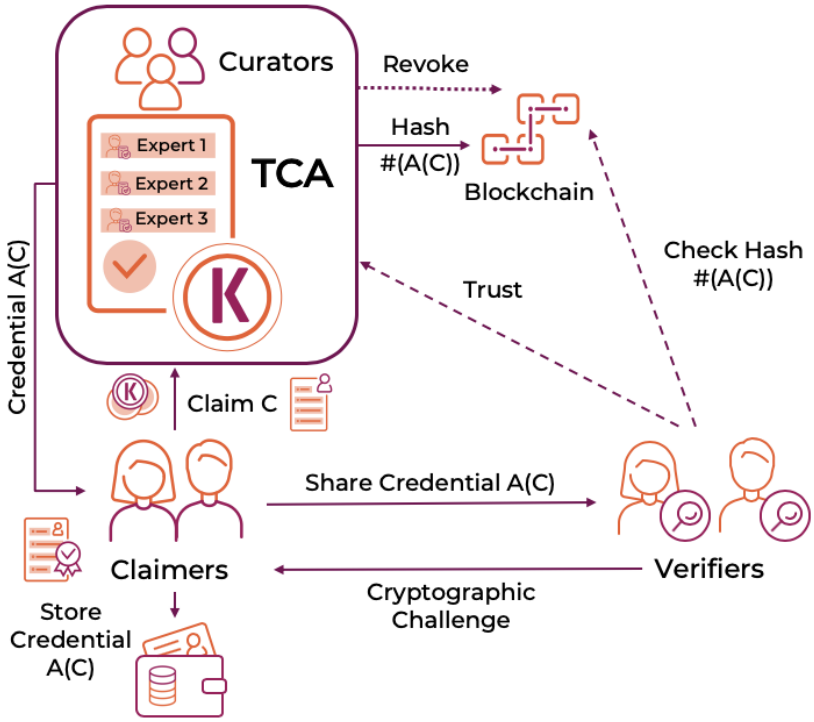


Decoupling the roles and mechanisms in the TCR and adapting them to the Analogue World

So the basic principle of TCRs which allows one to make decisions through a curation process is to be seen very positive and useful, if we decouple the Curation from the actual work of making complex decisions. This means that the decision whether to include a restaurant on the list should be done by experts in the relevant field and not the Curators.

5.2. Introduction to the Token-Curated Attester

We aim to combine the technological benefits of the TCR, which decouples the curation from the decision-making part, and implement it in a way that resembles the real world approach (curated Experts doing the work) into the KILT Protocol to receive a very powerful tool we call Token-Curated Attester (TCA).



TCAs can build up trust with Curators selecting the best experts

TCA Issues Credentials to Claimers

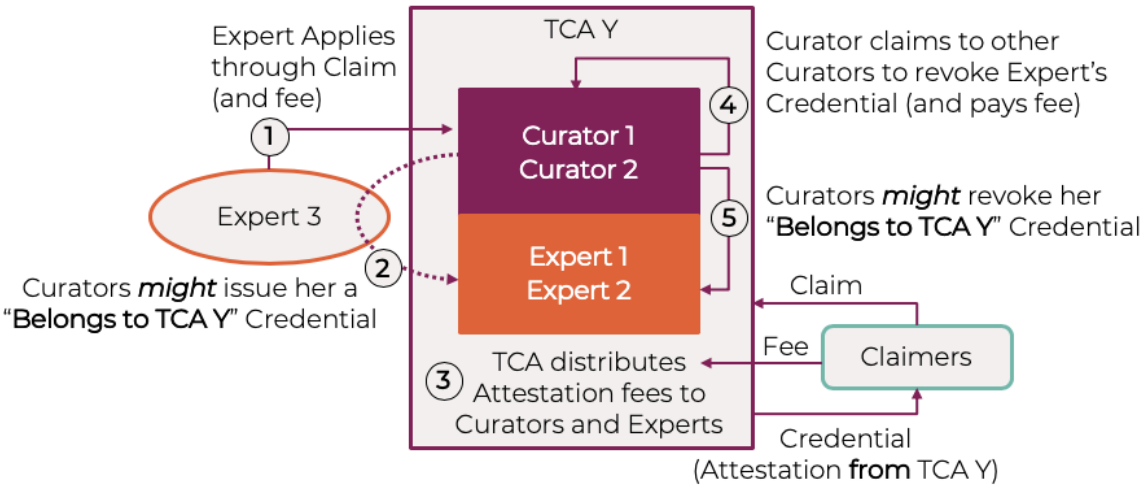
The protocol elements for Claimers, Attesters and Verifiers, moreover the Credential storage and revocation mechanisms of the blockchain would not differ from the claiming-attestation-verification process described in Chapter 2. The main process for a TCA as we imagine it at this point of time, would be the following. Like in the Top-Down Approach, the Claimer would send its attributes and ask the TCA the price for the Attestation. The TCA would automatically reply with a price. If the Claimer then should decide he would want to become certified under these circumstances, he would reply with a (claimer-)signed version of the Claim, accompanied by the necessary funds.

Experts Do the Inspection Work before Issuing TCA Credentials

At this stage one or more Experts (depending on the configuration of the respective TCA) would pick up the task[28] and take all necessary measures to check if the Claimer was legitimated to have her claim certified. If this would be the case, the Credential would be signed and sent to the Claimer. A hash of the Credential would be stored on the KILT Blockchain on

[28]The exact way of how a specific Expert is assigned to a specific claiming process is yet to be defined. It might be that a business logic outside of the KILT Protocol takes care of it or it might be the case that it will be implemented in the KILT blockchain code.

behalf of the TCA. If the Experts should decide that the Claimer would fail to meet the criteria to receive the Credential, the process would end without a Credential issued. The fee paid by the Claimer would remain in the TCA and would not be returned in any case, as it would have been paid for the check of the criteria that has duly been performed irrespective of the outcome.



The inside mechanisms of TCAs would be enabled by KILT Credentials

Curators Select the Best Experts for their TCA

An Expert would apply to the TCA by claiming to be a fitting Expert for the TCA (also using the basic processes described in Chapter 2). Curators would be incentivised to only accept and keep Experts who would really provide value to the TCA. This process may include that the Expert would need to hand in all necessary Legitimations (defined by the parameters of the TCA) and an application fee (also defined in the parameters). If the Curators would decide to accept the Expert for the TCA Y, the Expert would receive a "Belongs to TCA Y" Credential for being part of TCA Y and would be referred certain tasks from TCA Y (make decisions on whether or not a Credential would be issued in the name of this specific TCA). Experts would receive a share of the fees the Claimers pay to the TCA for the Attestation made based on the Expert's work. Experts would be free to leave at any time by simply discarding their "Belongs to TCA Y" Credentials (note that they do not get the Claim fee for the initial application back). Any Curator of TCA Y could challenge an Expert at any point of time. This would mean that the Expert's "Belongs to TCA Y" Credential would be filed for revocation. If the Curators' vote would decide to withdraw this Credential for her to be an Expert for the TCA, the Expert would no longer be working for the TCA.

5.3. Economic Incentives for Curators and Experts in a TCA

TCAs would be economic entities that shall

- have an inherent value, defined by all the KILT tokens locked through them
- give work to Experts, which get paid for their work by the TCA
- provide services (Attestations) to the public
- be able to make a profit. The profit would be the share of the Attestation fees paid by Claimers, which would be distributed amongst the Curators and Experts.

In the following list we summarize the incentive mechanisms that, with some reasonable assumptions on certain details of the process, would make the TCA structure financially sustainable (the numbers of the bullet points refer to the steps on the previous figure).

1. Expert is de-incentivized through fee to apply (multiple times) to the list.
   → *keep the workload for the Curators low*
2. Curators always get the fee. Though they will only accept Experts, which will increase the value of the list.
   → *keep the list clean*
3. Once on the list, the Expert will take care not to be challenged, as might loose his income.
   → *keep the list valuable*
4. Curators are de-incentivized through fee to other Curators to randomly start Challenges.
   → *keep the list calm*
5. If Expert's presence on the list damages the value of the list, Curators are incentivized to Challenge, because paying fee is less than possible damage to value of the list.
   → *keep the list clean*

Incentive mechanisms in TCAs

Experts would be interested in being listed on a TCA to increase their reputation. Being listed on a well-known TCA would improve the Expert's trust so more Verifiers would believe in his attestation which in turn would allow him to charge more for his attestations or would create more traction for his work. The economic mechanism behind this increase in trust would be two-fold. First, trust could come from the token holders as they curate the Experts and secure the value. In the beginning this could be considered rather a top-down approach but as soon as there were enough Curators, the process could be considered bottom-up. Second, the trust level of each Expert would amplify the trust level of the rest of the Experts that would be listed in the registry. Since an Expert would be listed in an environment with other high-quality Experts, they would enhance each other's trust score.

5.4. TCA Subtoken Model

A TCA is planned to be a self-sustainable public entity, hence it shall be enabled and organised with a TCA specific (intrinsic) subtoken.

TCAs, just like TCRs, are planned to satisfy the tenets of Mike's Cryptosystems Manifesto. A token would be a necessary element of such a system if the use of any other payment method in its place would damage the system's normal functioning. TCAs would therefore require intrinsic subtokens because token holders would have to realize both the upside and downside of their good or bad work in order to be motivated to perform their essential curation task. The model of TCAs would satisfy the criteria of token-necessity.

A system is self-sustaining if it would continue to function normally in the indefinite absence of its creators. In the model for TCAs no entity in the TCA would have special privilege. All TCA subtokens of a TCA would be equal and only token weight would determine the weight of one's privilege in a TCA structure. The creator of TCAs could disappear and the closed-loop incentive system of the TCAs would be indifferent. Therefore, TCAs would be created to be truly decentralized systems and would satisfy self-sustainability.

A system is a public utility if it is permissionless, rent-free, and does something useful. TCAs are planned to be permissionless[29], and therefore they would be truly decentralized, and only token weight determines privilege. They are planned to be rent-free, as tokens would never be at stake which are not necessary to incentivize other actors to perform some necessary task, or to disincentivize actors from griefing attacks. TCAs are also intended to produce useful output, namely Certificates, and therefore to satisfy public utility.

Following from the above, a TCA shall be organised with a TCA specific subtoken so that Curators may enforce their voting rights in the curation process and on the governance structure of the TCA. However, KILT tokens shall be used by candidates to become an Expert for a TCA and by Curators to initialize revocation of the Expert listed for the TCA. On the other hand, subtokens shall entail curation rights where the curation right shall be proportional to the relative subtoken weight of entities holding the subtoken. Subtokens could be freely traded outside theTCA but would always have a fixed exchange rate to the KILT tokens within the TCA. If subtokens were sold back to the TCA, they would be burned. The mechanism is explained in more detail in subsection "Bonding curves".

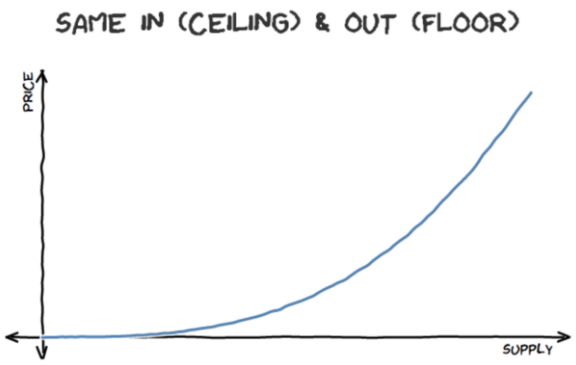
For becoming a subtoken holder, the considered entity had to buy into the TCA with KILT tokens receiving the respective TCA's subtokens in return. Subtokens were minted as needed. The price of the subtoken would increase when subtokens were bought. More precisely, the price would follow a predefined bonding curve[30] which will be explained in great detail in the next section. The price of the subtoken would reflect the demand for becoming a subtoken holder of the TCA. Rational actors would buy exactly the amount of subtokens where the value of the list is higher than the price to buy in. The value would be influenced by the quality of the list among others. As a result, subtoken holders were incentivised to maintain a popular, high-quality pool of Experts. Essentially, subtoken holders would keep candidates desirous of being listed in the registry by maintaining consumer interest in the registry by keeping the quality of listings high.[31] Details on the factors that could influence the value of a TCA are explained in section "Economic incentive for TCA participants".

Bonding Curve

As already mentioned, the price of a subtoken in a TCA would follow a predefined bonding curve. The shape of the bonding curve might be linear or exponential. In any case, if one would buy in early, one would receive more subtokens for the same price. In line with this, if one

[29] Subject to the Creator of the TCA. See Starting a TCA and Becoming a Curator section for details.
[30] "Tokens 2.0: Curved Token Bonding in Curation Markets" by Simon de la Rouviere, Medium, 2017, last accessed on 2019.02.13.
[31] "What is a Token Curated Registry?" by Token Curated Registry, Medium, 2018, last accessed on 2019.02.13.

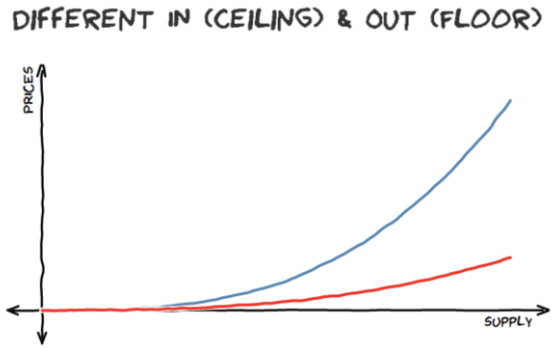
would buy in later, one would receive less subtokens for the same price. With more subtokens in circulation, the cost to buy new subtokens would go up.



Bonding curve with same buy and sell curves[30]

When subtoken holders would buy into the subtoken with KILT token, the KILT tokens were locked in a communal pool. At any point in time, someone could sell back their subtoken into the communal pool and get out a certain amount of KILT tokens that was set by the sell curve and the current supply. If the amount of subtokens in circulation had increased since the respective subtokens were bought, then the token holder would receive a higher amount of KILT tokens when selling the subtokens back to the communal pool than when he originally bought into the TCA. Importantly, this would be only always true for bonding curves where the buy and sell curves were the same as in the picture above. If token holders sold back their tokens at a time where less subtokens were in circulation as to when they bought subtokens, then the token holders would always make a loss. If everyone would leave the TCA, all KILT tokens would be refunded and all subtokens would cease to exist. The TCA would thus grow and shrink naturally as the usefulness of the list changes. If the registry was useful and a group of subtoken holders judiciously curate it beneficially, there would be more buyers who want to be part of the list and more buyers who want to participate in the curation of it (in order to receive rewards for doing so).

By default, the price for buying into the TCA would be the same as the selling price, but potentially it could also be different. This would create an incentive to keep the subtokens for a certain period of time to make a profit when selling subtokens or exiting the TCA. In such a case, if token holders immediately sold back their newly bought subtokens, then they would make a loss since the selling price is lower than the buy price as precisely defined by the bonding curves. Often, the curves could be designed in a way that over time, the gap between the ceiling (buying in, blue line) and floor (selling out, red line) would get larger, so buyers wouldn't be immediately in a profit as soon as new participants enter. The gap between in and out would serve as a funding opportunity for the project.



Bonding curve with different buy and sell curves[30]

In summary, KILT would propose an exponential bonding curve as default to leverage the incentives of buying in early. Moreover, KILT would offer different bonding curves and the creator would choose one when creating the TCA. The curve could not be changed at any later point in time. In addition, we can imagine that the KILT Protocol will enable the creator of the TCA to set the buy and sell curve to be different, so investors would have to wait a certain period of time to make their investment profitable. This could make the TCA more stable.

Subtoken Price Determination

The KILT Blockchain would provide functions for exchanging KILT tokens in any of the TCA's subtokens and back. All these transactions would take place on the KILT Blockchain. The price of a TCA subtoken within the TCA would be determined according to the respective up-to-date supply along a fixed bonding curve. Whenever KILT tokens would be exchanged into TCA subtokens in the TCA, this would create new TCA subtokens, while locking the KILT tokens paid for them. This would be the only way to create new TCA subtokens and it would determine the supply of the TCA subtokens. The respective up-to-date Token Supply would be known to the KILT Blockchain and could be retrieved by anyone at any time. Exchanging TCA subtokens back into KILT tokens would always be possible and would be performed by a call into the KILT Blockchain. Exchanging TCA subtokens back into KILT would unlock the locked KILT tokens and destroy the amount of TCA subtokens changed back. Hence, the Supply of the TCA subtoken would decrease. When the last Token of a TCA would be exchanged back into KILT Coins, all KILT tokens which were locked would be unlocked again and the TCA would cease to exist.
It is important to note that TCA subtokens could also be traded freely on secondary markets, e.g. exchanges or directly between different entities. All these transactions would also take place on the KILT Blockchain.

Curators shall be interested in an increase in subtoken price of the TCA which shall be determined by the subtoken value. The subtoken value would depend on the quality of the Experts, the cash inflow through the Attestations, as well as the future subtoken holders' wish to influence the attested Experts and the general outlook of the TCA. The quality of the TCA would depend on the quality of the Experts doing the work behind the Attestations for it. So, the subtoken holders would have to make sure that high quality Experts would have an interest to work for the TCA and the curation process properly selects only high-quality Experts. In order to create a long-term incentive for the Curators and to avoid that they cash out of the TCA (when it seems opportune because they could make a profit by selling the TCA

subtokens), they should receive a revenue stream. This cash inflow would be the share of the fees paid by the Claimers for the Attestations issued by the TCA based on the work of the Experts in the TCA. Moreover, the Curators would share the fees paid by the Curators during a challenge process to exclude an Expert from the TCA.

The subtoken price would be defined by the bonding curve. Namely, it would increase according to that curve's increase when Curators would buy into the TCA and decrease when Curators are getting their shares out of the TCA. The decision on whether to buy into a TCA would (amongst other reasons outside the logic of the TCA) be defined by the subtoken value: whenever the subtoken value would be below the subtoken price, it would be reasonable to invest. The same would hold true for the other way around. The aspect that subtoken holders would be willing to influence the attested Experts for the TCA and thus to buy into the TCA would be included in the definition of the value of the subtoken. Investors would take this into account when calculating the subtoken value. So, speculators would buy subtokens as long as they expect that there will be more demand for the subtoken in the future.

Starting a TCA

Anyone could start a TCA and become the first Curator of the list. The creator of the TCA would need to exchange a certain amount of KILT token into the TCA's own subtokens, generating the first supply. At first, the creator would have 100% of the supply of the subtokens. Since these subtokens would resemble voting rights among Curators, this first Curator would get 100% of the voting rights within the TCA.

The Creator would create the TCA by specifying his desired initial parameters and then registering it on the the KILT Blockchain with these parameters. Since he would set all initial parameters, he thus would decide on the governance structure of the TCA.

The initial parameters for a TCA might include:

- The Credential Type(s) (CTYPEs) this TCA issues
- Selecting a bonding curve
- A minimum number of TCA subtokens a Curator must hold
- A maximum number of TCA subtokens issued to Curators other than himself
- A maximum supply of the TCA subtoken
- Legitimations a Curator must bring to be accepted (e.g. valid tax number)
- Legitimations the TCA itself must acquire before the blockchain runs the code (e.g. tax number)
- Legitimations an Expert must bring
- Voting frequencies and rules
- Share distributions between Experts and Curators (from the fees paid by Claimers)

The Creator would then call a constructor method on the KILT Blockchain with these parameters. Depending on the initial parameters and possibly required Legitimations the Blockchain would run the TCA, when everything is decided and ready. After the TCA would run on the KILT Blockchain, other Curators would be able to invest in the TCA by purchasing subtokens and Experts could apply for being on the Expert's list. As soon an Expert would be listed, the TCA could start running and be fully operational, which means it could accept claims, conduct checks and issue Credentials.

Becoming a Curator by Buying into the TCA

The Creator would be the first Curator of a TCA. Other Curators may join, by exchanging KILT tokens into the TCA's subtokens. As explained above, the exchange rate would be determined through a bonding curve, which would be inherent to the TCA. Curators would have voting rights on the Experts working for it. The percentage of the TCA's subtokens they could hold (relative to the total supply) would determine their voting power. This might be restricted by governance. The Curator would also receive a share of the TCA's profit, according to the percentage of TCA's subtokens held by him. Depending on the governance, the Curator might exchange his TCA's subtokens back to KILT tokens at any time or after a defined period. The exchange rate would be determined by the TCA's Bonding Curve at the time when the transaction takes place. Becoming a Curator could be restricted by the governance of TCA, such that certain legitimations might be provided during a KYC process, the number of invitations to Curators, their maximum share might be limited or other restrictions that seem to be useful for the TCA according for example to the legal and other framework it would be created and living in.

After the TCA would have started, anyone might buy subtokens and become a Curator. For doing so, one would have to exchange KILT tokens into the TCR's subtokens. Through this exchange, new subtokens were generated and hence the subtoken supply would rise. Exchanging subtokens back into KILT tokens would burn subtokens and so the supply would shrink. The actual exchange rate of the subtoken to KILT would depend on the current supply of the subtoken and would be calculated according to the bonding curve assigned to the TCA. As the price of the subtoken would rise according to a monotonically increasing bonding curve, late Curators would pay more for the same weight in voting.

Governance Mechanisms

There would be the option to include a governance mechanism so parameters could be adapted through Curator vote when the respective TCA would already be live. At this stage in the design phase, it is unclear whether or how this will be implemented. The conceptual choice to design one fixed governance scheme as a standard or one with more flexibility in its mechanisms is still to be made before implementing the options for the TCAs.

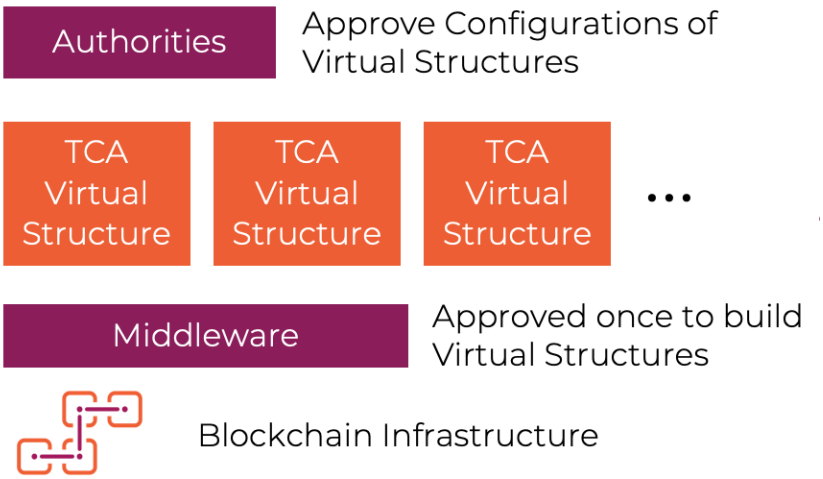
5.5. Regulation-Friendly TCA Ecosystem built on the KILT Protocol

In general, the KILT protocol shall define the framework for TCAs where the exact properties of the TCA shall be left to be defined on the application layer according to the respective use case. So eventually, the TCA creator would decide on the parameters of the TCA and would check how the technical implementation and the legal system of his jurisdiction fit together best. However, the KILT Protocol shall include a proposed general standard to streamline and potentially automate this process.

We believe that numerous business models could be represented with TCAs and that thousands of TCAs could eventually run on the KILT Blockchain in parallel in the future. TCAs

could enable virtual structures that act like companies, running on a blockchain, which could be completely transparent for investors and service providers while being regulated and taxed by governments like traditional companies. Thus we aim to build a TCA Factory on top of KILT, which allows anyone to set up a TCA with a simple request into the KILT Blockchain. Unlike Smart Contracts, TCAs would not be programmable but configurable. A certain set of parameters would be available to TCA-Creators for configuring the TCA according to the particular task.

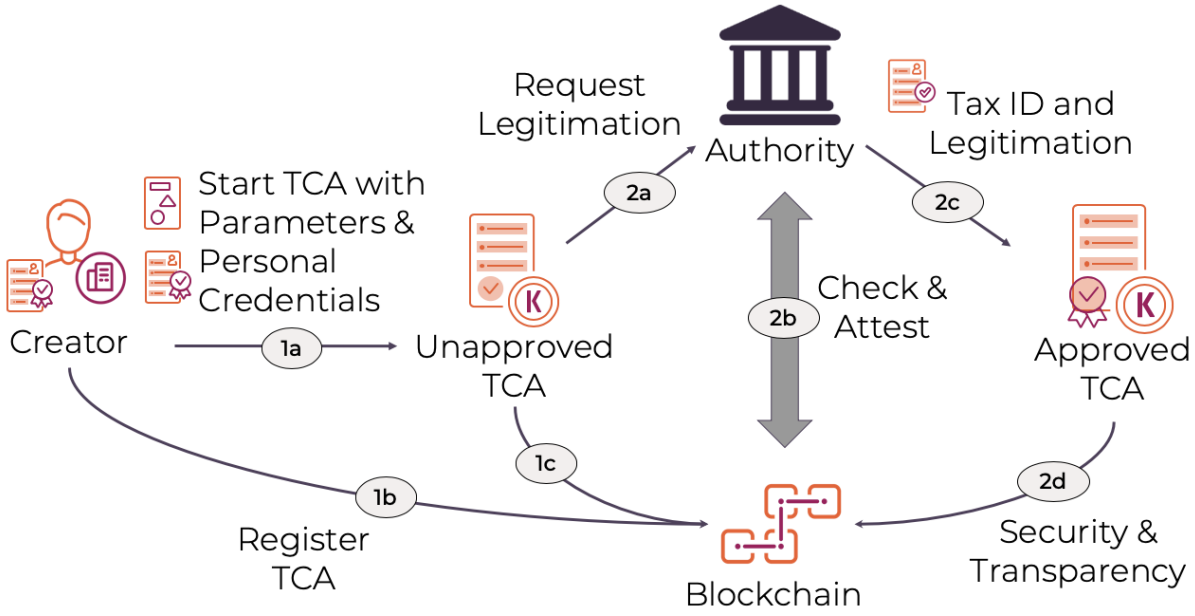
There is a trade-off on the flexibility in designing the TCA individually: the more flexible the TCA would be programmed, the more complex the protocol and the more attack vectors would emerge. On the other side, flexibility would allow a better adaptation to user needs which could improve user satisfaction.



TCA Factory Middleware built on top of the KILT Protocol

A TCA could be configured through numerous parameters. TCA Creators could create TCAs fitting their particular needs through these parameters. Regulators could use the TCA concept to allow commercial organisations, running completely on a blockchain within their jurisdiction and under their rule. They could, for example, define that a TCA and all its Curators and Experts should have an official tax number to be accepted in their jurisdiction. This could be enforced through the definition of necessary Legitimations, which would also be a property of the KILT Protocol. If such a system was ever accepted by a jurisdiction, this would give investors in TCAs the possibility to invest in a legally approved entity while enabling founders to rapidly set up a commercial structure without a street address and a bank account, which functions in a well-defined and transparent way.

Such features could also be mirroring the legal and tax framework in which such a TCA exists, so that authorities could *ex ante* approve TCAs of a certain configuration and provide interfaces to them for tax payment or other legal necessities. These predefined and approved TCA templates could then be used by Creators to configure their own TCAs according to the approval of the local authorities and they would benefit not only from the legal security but also from the infrastructure provided by the tax and other authorities.



Registering TCA with Authorities

6. KILT Token Economy

As explained in Chapter 3, the KILT network shall enable the establishment of an economy where trust can be delegated in return for rewards. This trust market economy shall emerge naturally. In this chapter we discuss the economic aspects of the network from the perspective of the KILT token. It is important to note that the design mechanisms discussed here express our current view on this topic, however, this mechanism is subject to change based on ongoing research until the time the network goes live and beyond.

6.1. KILT Token

The KILT token is required to perform certain actions within the KILT Network. The issuance of the initial tranches of KILT token is done by Botlabs GmbH and then the protocol/network would allow the creation of (block-)rewards and would distribute it following predefined mechanisms. There is a chance for the KILT token[32] to be listed on exchanges after the launch of the KILT main-net which would facilitate its tradebility against other cryptocurrencies and fiat currencies.

Overview of the KILT Token Functions

The KILT token could be used for providing safety of the KILT Blockchain and having access to write to the secure block space for trusted data. Utility of the KILT coin can be broken down to the following possible use cases:

- Paying angel's share (gas) when writing an attestation on the KILT Blockchain
- Paying an Attestation fee[33]
- Register a new CTYPE on the blockchain
- Creating a Hierarchy of Trust (including the Private Curated Registry) and managing the delegations in the structure
- Creating and setting up a Token Curated Attester (TCA, explained in Chapter 5)
- Expert applying to a TCA
- Paying for an Attestation issued by a TCA
- Direct transactions in KILT tokens between users of the network
- Distributing the block reward to Validators
- Contributing to network security (staking/nominating Validators in the proof-of-stake consensus mechanism)

KILT Token Emission

During the whole life of the KILT network, one billion KILT tokens are planned to be created which implies a finite token supply. 40% percent of the token supply would be pre-mined and distributed before the launch of the network. The remaining 60% of the available token supply

[32] As we already discussed in Payment Transactions, the KILT Coin will be the currency of the KILT network after the official launch of the main-net and this currency shall potentially be listed on exchanges.

[33] The Claimer may also pay the attestation fee in fiat or other cryptocurrencies depending on the payment framework set by the Attester.

would be minted over time and would be rewarded to agents that conduct valuable work for the network. The tokens for the block reward would be minted as needed, i.e. created out of thin air according to predetermined rules. The block reward function would be defined in such a way that more tokens are released in the first years of the network life than later to incentivise adoption. The block distribution is currently designed in a way that after 50 years, all one billion KILT tokens would be released and there would be no more block reward after that. By that point in time, we envision KILT to be a well established true ledger for trust relationships, which might be linked to multiple external systems.

Reward Pool

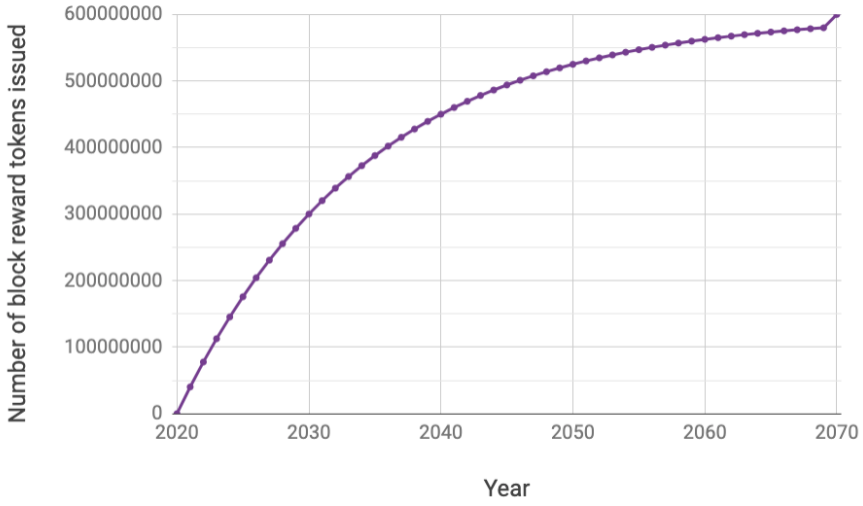
600 million KILT tokens (60% of the total supply) would be emitted by the KILT network over time to Validators (and maybe to Attesters). According to the current design, the emission of the rewards would follow a bounded exponential curve according to the following formula:

$$M = 1 - 0{,}5^{t/H}$$

where:

M = Cumulative ratio of tokens released via block rewards
t = Time (years)
H = Half-life of the block reward in years

After the first *H* years, 50% of the total amount of tokens for block reward would be emitted. Currently we consider that a half-life of 10 years is most reasonable. This would give a balance between sustained network growth during the initial critical period and still incentivising contribution to the network in later years. According to the proposed function and parameters, the KILT Protocol would emit 50% of the block rewards in the first 10 years. Then, every 10 years, the remaining tokens to be minted would be halved.



The distribution of tokens in the 50th year would follow a different distribution curve as the remaining tokens would be released linearly, such that all 1 billion tokens shall eventually be distributed by the end of the 50th year of KILT's life cycle. The total number of tokens that would already have been emissioned in the first 49 years can be easily calculated as $(1 - 0{,}5^{49/10}) * 600.000.000 = 579.904.248$. In turn the remaining number of tokens to be released in year 50 would be $600.000.000 - 579.904.248 = 20.095.752$ that would be issued linearly distributed throughout the year.

It has been argued that in general fiat currencies dissolve after about 50 years (due to the decline of power of the state backing the currency).[34] Inspired by these insights, we are proposing the block reward to diminish over 50 years since we consider it realistic to plan the token economy for such a timeframe. In the current design, most tokens are emissioned in the early years of the protocol to incentivise bootstrapping the system. Then, the focus is set on keeping the blockchain running which is primarily rewarded with transaction fees that are distributed to Validators. In addition, the high reward in the beginning incentivises Validators to secure their stake early and start validating new blocks.

Controlling Inflation in Proof of Stake Systems

Latest research on the token economy of Proof of Stake (PoS) blockchain systems shows that controlling the inflation of tokens could be required to defend against cannibalization attacks.[35] It is argued that decentralised finance solutions such as on-chain lending may disrupt proof of stake consensus depleting the total stake with economic incentives from outside the system to make a 51% attack feasible. In general this piece of research concludes that, if a PoS block reward is decreasing over time, then its long-run equilibrium will be for almost all assets to be lent, not staked.

6.2. Designing Demand and Incentives for the KILT Token

The KILT block reward mechanisms are currently designed in a way that it should keep the KILT Protocol running and aligned with the right incentives to use the network.

Block Rewards for Security and Consensus

This section describes the proposed distribution of the 60% of KILT tokens as block reward on a high-level. These token rewards would be meant to incentivise providing security and consensus about the state of the blockchain through mining that would be performed by Validators.

We are currently proposing one minute as the block time. This means that every minute a block would be written, and rewards would be distributed. The eventual block time would depend highly on technical feasibility which could be only shorter than one minute in case the underlying blockchain would be operating fast enough. Here, the KILT network would depend on the technical evolution of the Substrate blockchain framework on which KILT would be built upon. The rewards for security and consensus would be directly distributed to Validators. Only the Validator who would mine the respective block would receive the block reward which would depend on his stake and on probability.

[34] Chris Mack: Is This Time Different for the Dollar?, https://www.financialsense.com/contributors/chris-mack/is-this-time-different-for-the-dollar, last accessed on 25th March, 2019.

[35] Haseeb Qureshi: How DeFi cannibalizes PoS security, https://medium.com/dragonfly-research/how-defi-cannibalizes-pos-security-84b146f00697, last accessed on 18th December, 2019.

Token Lock-up due to Staking in Proof-of-Stake

From today's perspective, the KILT Blockchain is set out as a proof-of-stake based blockchain network. This means that the validators would need to stake KILT tokens to be able to take part in the validation process. This requirement for staking would increase the scarcity of KILT tokens.

Utility of KILT tokens

Validators would receive block rewards for providing security and ensuring the correctness of state changes on the network through consensus. Essentially, Validators would keep the network alive. They would validate two aspects: formal correctness of attestations/revocations and sound authorisation of the one performing the attestation/revocation. Considering the validation of an attestation, the Validators would first check whether the structure of the CTYPE fits predetermined criteria. Then, they would check whether the Attester would have the permissions to write an attestation/revocation, i.e. the Validator would check the rights of the Attester and whether he would have delegated rights through a hierarchy of trust or through a PCR. Essentially, Validators would provide the supply of secure and correct block space demanded by those that write the hash of credentials on the blockchain.

When writing the hash of credentials on the blockchain, Attesters would have to pay the angel's share (gas or transaction fee) in KILT tokens. In the current design, this cost would then be distributed only to Validators. It would be added to the block reward designated to the Validators and then distributed in the same way as the block reward. In this design, Attesters would not receive any of the angel's share except when they become Validators themselves. The angel's share distribution would be particularly important considering the time when there would be no more KILT tokens emissioned (in 50 years).

Users of the KILT Protocol would be able to use KILT tokens to join TCAs as curators. This would mean that they need to buy-in to the TCA with KILT Coins which then would be exchanged into TCA subtokens.

6.3. Open Topics

In a later version of the white paper, we might go into possible attack vectors in the KILT token economy. Also, the mechanism on token appreciation especially in regard to TCAs might be fleshed out in more detail. In addition to the current incentive mechanism, we envision another one that might include the opportunity to invest in CTYPEs one considers as promising. Moreover, we can also imagine an incentivization mechanism for making the best CTYPEs available to the public. Also, the angel's share (gas) price for writing attestation on-chain shall be defined as well as the pricing mechanisms. Finally, if the KILT Blockchain ever becomes a parachain in the Polkadot network, that might change the security model for our token economy for instance since the Validation function may be delegated to Relay-Chain Validators to share the state and security between parachains.

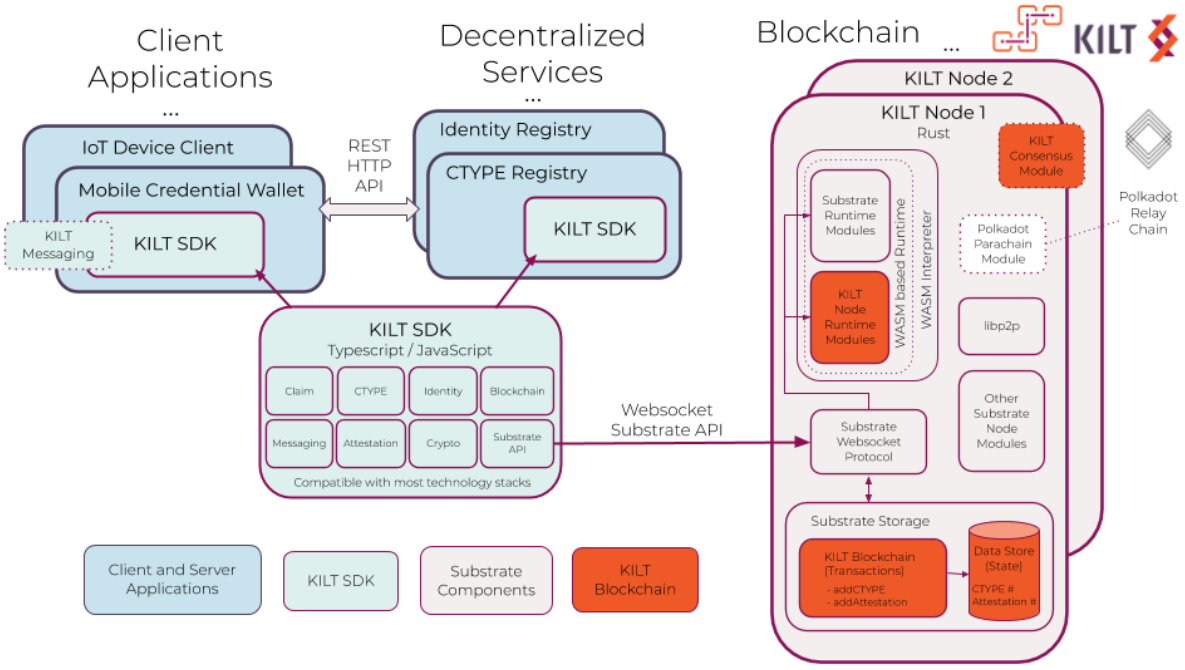
7. System Architecture

In this chapter we show the planned relationship between the various components of the core system and their dependencies to third party developments. This section addresses any developer who wants to integrate the KILT Protocol and blockchain developers, who wish to contribute to the KILT Community by building general purpose services or by improving the KILT Protocol itself.

7.1. KILT Overview

The KILT ecosystem will be composed of different interacting parts (shown below) which will work together to provide the functionalities we described in Chapters 2-4. KILT will be basically composed of client applications (e.g. Credential Wallet App), centralised as well as decentralised services (e.g. CTYPE registry), and the KILT Blockchain.

The applications and services interact with the blockchain by either implementing the KILT protocol or integrating the KILT SDK, which already implements the KILT protocol and is provided by us. Our KILT SDK implements basic functionalities like storing an Attestation or adding a CTYPE on the blockchain, it handles Identities, CTYPEs, Claims, Attestations and Verifications, and Complex Trust Structures. It also includes helper functions to work with cryptographic libraries and KILT specific messages. Here we describe the protocol in detail (always noting important interdependence with the SDK and the services) and in the next chapter we describe how the services fit into the KILT ecosystem.



KILT System Overview

7.2. KILT Protocol

The KILT Protocol describes three main layers:

- The **data formats** and how to handle different types of data, such as Identities, CTYPEs, Claims, Attestations and Verifications, is the core of the KILT Protocol.
- Rules, guidelines and workflows, as well as a **messaging protocol** supporting these processes and exchange data formats.
- The KILT **Blockchain nodes** use Parity Substrate as the underlying blockchain technology stack (see details later in KILT Blockchain section). To support KILT, the nodes need to implement different modules in their runtime.

Identity Management

An Identity contains following properties:

- **signing keypair**: currently uses the ed25519 public-key signature system, but it will be replaced by sr25519 in the next version of Substrate
- **address**: the public address is generated by encoding the signing public key, using the ss58 algorithm.
- **encryption keypair**: uses x25519-xsalsa20-poly1305 for encrypting messages between participants of the system

A client software can easily create a keypair using libraries implementing the ed25519 signature suite, however, this process is greatly simplified by employing the Identity module of the KILT SDK. Currently, all public parts of KILT identities can be registered into the Central Contact Registry in order to participate in sample workflows, but this is subject to the demo application only. In the future we envision all kind of different scenarios for contact management (see next chapter).

KILT Decentralised Identifier (DID)

Although it is not mandatory for using the KILT protocol, users can optionally create a DID and anchor it to the KILT blockchain. KILT currently supports creating DIDs simply by using the **address** of the identity as the method specific identifier, for example:

```
did:kilt:5GZ1ri8q2h7hXJHe9CnJVMAvGdRi3rbrrxfYBNHLwzH55daF
```

KILT DIDs are resolved to a DID entry stored on the KILT Blockchain. This entry includes a signing key, an encryption key, and optionally a link to the corresponding DID Document. The corresponding DID Document referenced by the link in the DID entry can be stored anywhere (user cloud agent, central service, etc.). For demo purposes it may also be stored in the Central Contact Registry.

The DID Document is conform to the current DID specification and it currently contains:

- signing key
- encryption key
- service endpoint pointing to our messaging service.

We are implementing a fully featured DID stack into KILT. This involves publishing our DID specification as a `kilt` DID method specification and implementing the KILT DID resolution as a driver for the Universal Resolver. Moreover, we are also planning to support peer DIDs.

Two ways of using DIDs with KILT

Using DIDs in KILT can be done in two different ways:

- **Static**: For a given DID identifier, a DID Document is created and stored at a given location such as a regular server. A DID entry that also specifies the link to the DID Document is stored on-chain. In this case, it is recommended to protect the DID Document from unwanted tampering. One way to do so is to store a signature of the hashed DID Document together with the DID Document on the above-mentioned storage location. This signature can be verified later on.
- **Dynamic**: The DID entry is stored on-chain, without specifying a link to the DID Document. The DID Document is regenerated on-the-fly using the SDK's DID methods such as createDefaultDidDocument.

Node Implementation of DID Registration

KILT Blockchain nodes need to implement a DID module, with an `add` function.
This function takes the following parameters:

- `owner`: public ss58 address of the caller of the method
- `signKey`: the ed25519 public signing key of the owner
- `boxKey`: the x25519-xsalsa20-poly1305 public encryption key of the owner
- `docRef`: Optional u8 byte vector representing the reference (URL) to the DID document

The blockchain node verifies the transaction signature corresponding to the `owner` and inserts it to the blockchain storage by using a map (done by the substrate framework):

```
owner => (signKey, boxKey, docRef)
```

As DID supports CRUD (Create, Read, Update, Delete) operations, a `get_dids` method reads a DID for an account address, the `add` function may also be used to update a DID and a `remove` function that takes the `owner` as a single parameter removes the DID from the map, so any later read operation call does not return the data of a removed DID.

Creating, Registering and Publishing a CTYPE

As we already introduced and discussed in Chapter 2 & 4, a Claim type (CTYPE) defines a schema, with which Claims can be created. CTYPEs can be created by a client software, then registered on the KILT Blockchain and finally published in a CTYPE registry service.

Building a CTYPE

CTYPEs can be built locally by client software most easily by implementing the CTYPE creator modules of the KILT SDK. The CTYPE Claim schema is built using JSON Schema, with following meta-schema. Every CTYPE needs to validate against this meta-schema:

```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
```

```
    "$id": "http://kilt-protocol.org/draft-01/ctype#",
    "type": "object",
    "properties": {
        "$id": {
            "type": "string"
        },
        "$schema": {
            "type": "string",
            "format": "uri"
        },
        "type": {
            "type": "string",
            "enum": [
                "object"
            ]
        },
        "properties": {
            "type": "object",
            "patternProperties": {
                "^.*$": {
                    "type": "object",
                    "properties": {
                        "type": {
                            "type": "string",
                            "enum": [
                                "string",
                                "integer",
                                "number",
                                "boolean",
                                "array"
                            ]
                        }
                    }
                }
            }
        }
    }
}
```

Register a CTYPE

After creating a CTYPE locally, the creator needs to register the CTYPE to the KILT Blockchain so that all users can check if a Claim is really conforming to a CTYPE. The locally created CTYPE is first normalised by removing all whitespace outside of strings and is hashed by using the blake2b hashing algorithm (encoded as hex string, starting with `0x`). This process is shown in the next example:

Example of Locally Created CTYPE (plain JSON):

```
{
    "$id": "DriversLicense",
```

```
    "$schema": "http://kilt-protocol.org/draft-01/ctype#",
    "properties": {
        "name": {
            "type": "string"
        },
        "age": {
            "type": "number"
        }
    },
    "type": "object"
}
```

Normalised:

```
{"$id":"DriversLicense","$schema":"http://kilt-protocol.org/draft-
01/ctype#","properties":{"name":{"type":"string"},"age":{"type":"number"}},
"type":"object"}
```

Hashed (hex-encoded):

```
0x9ec5ed4f752ad52ce5cf4bbe82f23389eb6dece3b95c13b3cdacf93a0ef92cff
```

This hash is signed by the creator of the CTYPE and registered (added) to the blockchain.

Node Implementation of CTYPE Registration

KILT Blockchain nodes need to implement a CTYPE module, with an `add` function.
This function takes following parameters:

- `creator`: public ss58 address of the caller of the method
- `CTYPEHash`: CTYPE hash as a blake2b string

The blockchain node verifies the transaction signature corresponding to the `creator` and inserts it to the blockchain storage by using a map (done by the substrate framework):

```
CTYPEHash => creator
```

Creating a Nested CTYPE

When creating a CTYPE, one can layer (“nest”) CTYPEs within one another, thereby creating more complex structures through the use of the keyword $ref (which is acting as a reference pointer from JSON Schema). The `$ref` is a URI, and KILT expects the format for the URI-reference `KILT:ctype:0x9ec5ed4f752ad52...`, mentioned in the “Register a CTYPE” section. This hash is a makeup of the schema produced and will be unique to the specific CTYPE. The CTYPEs nested in the schemas are known as subschemas which link via the $id keyword.

```
"schema": {
    "$id": "KILT:ctype::0x9ec5ed4f752ad52....", <- Unique Reference
                                                   Identifier (URI)
    "$schema": "http://kilt-protocol.org/draft-01/ctype#",
    "type": "object",
    "properties": {
      "full_name": { "type": "string" },
      "passport_identifer": { "type": "string" },
      "street_address": { "type": "string" },
      "city": { "type": "string" },
      "state": { "type": "string" }
    }
```

```
 },
 "owner": "....",
 "hash": "0x9ec5ed4f752ad52..." <- The URI could be a combination of
                                  the CType or Claim Hash
}
```

The metadata part of a CTYPE is planned to support readability in different languages. As specific CTYPEs could have multiple languages, that would create unique reference identifiers duplicating structures. For this reason the metadata was intentionally left out from the nested CTYPE structure.

Reference Pointer ($ref)

The **$ref**, in essence, has all the properties and types of the specific CTYPEs. These are not combined to make a new Object.

- CTYPE schemas can be recursive, to implement different CTYPEs, and these can perform respectively recursive. E.g. Nested CTYPEs can have further subschemas.
- CTYPE schema URIs are for identifying schemas and validators should not expect to be able to download the schemas from the URIs for the subschemas.
- CTYPEs **$id** needs to be unique identifiers for more than one schema.

Note that, to validate a CTYPE, which has subschemas in it, the user needs to have copies of all the involved CTYPEs (e.g. via a CTYPE registry, etc.).

Validating

KILT uses a library called Ajv to test and validate schemas. An additional step that is mentioned is checking against our meta-schema, which is CTypeModel, as seen below. That allows us to refine the data types.

```
const newCType = "KILT:ctype:CTYPEHash" <--- CType Object
const cTypePassport = "KILT:ctype:PASSPORTHash" <--- CType Object
const cTypeKYC = "KILT:ctype:KYCHash" <--- CType Object
// We can pass an array of Objects to the CTYPE creator
const validate = ajv.addSchema([cTypePassport, cTypeKYC])
                    .compile(newCType) <--- Compiles for validation
const data = {
   cType: "kilt:ctype:PASSPORT Hash",
   contents: {
         "full_name": "Archer Macdonald",
         "passport_identifer": "34jd83jd",
         ...
}
validate(data) // Returns true
```

addSchema must add all relevant CTYPE schemas to the **ajv** instance during its **compile** phase, then **validate** checks the **data** and returns true if the data is valid. This method does not compile schemas into a new object, but stores them within the cache of the instance and it validates against the data along with the meta-schema. It also prevents unnecessary compilation of schemas that contain other schemas.

oneOf Combination

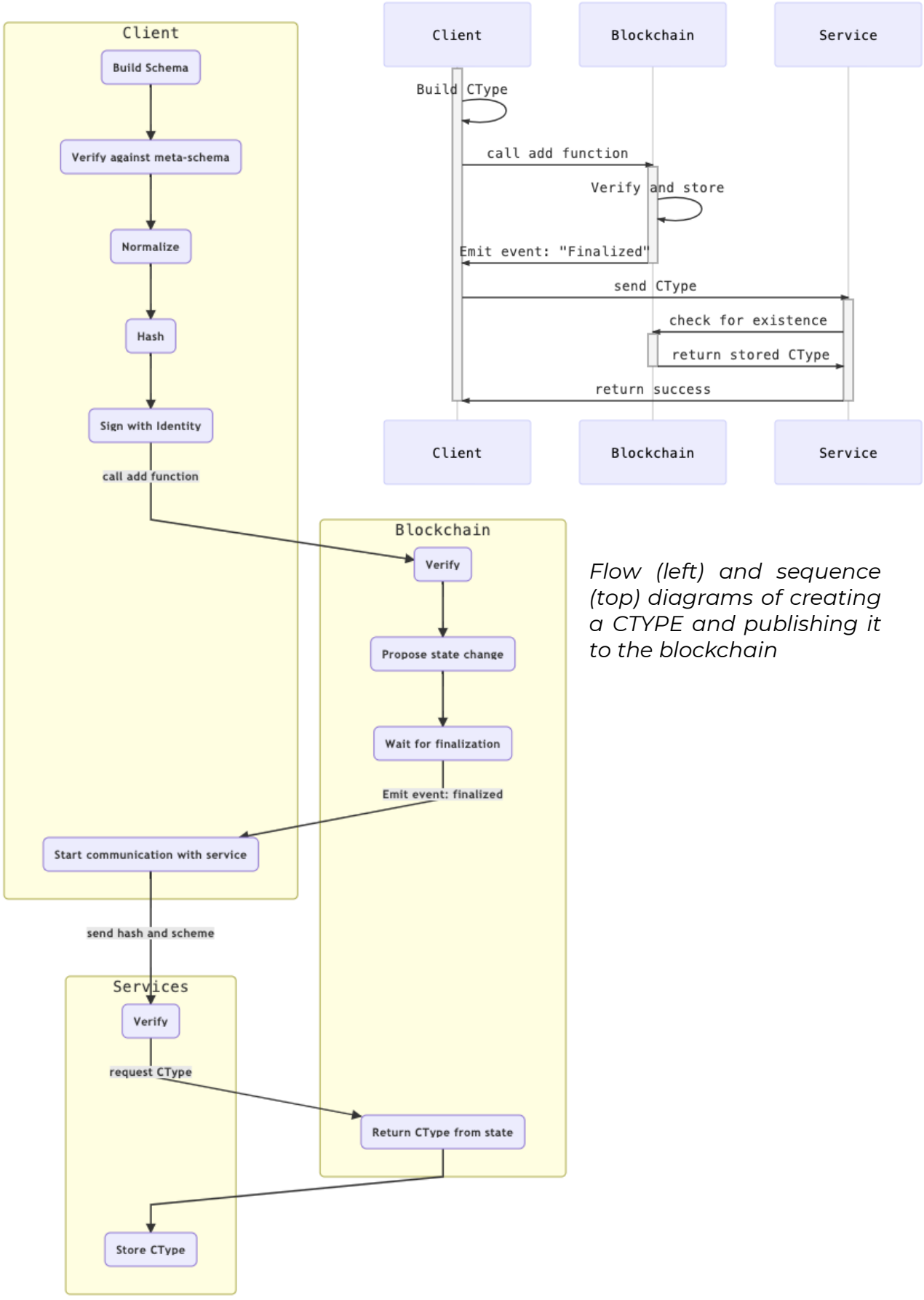
A feature within JSON Schema provides another keyword oneOf, which KILT expects to use. It combines a multitude of schemas, though the given data must be valid against exactly one

of the given subschemas, allowing a value to be validated against multiple criteria at the same time. Giving more choices to the CTYPE creation process and enabling more complex data structures. A simple example is shown as follows:

```
"schema": {
  "$id": "KILT:ctype:CTYPE Hash V2",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "Identifier": {
      "oneOf": [
        {
          "type": "object",
          "$ref": "KILT:ctype:ID Hash"
        },
        {
          "type": "object",
          "$ref": "KILT:ctype:PASSPORT Hash"
        },
        {
          "type": "object",
          "properties": {
            "full_name": { "type": "string" },
            "passport_identifer": { "type": "string" },
            "street_address": { "type": "string" },
            "city": { "type": "string" },
            "Country": { "type": "string" }
          }
        }
      ],
    },
    "Drivers_license": {
          "type": "object",
          "$ref": "KILT:ctype:Drivers_license Hash"
    }
  },
}
```

Publishing a CTYPE on the Registry Service

If the CTYPE hash is present on the chain, then it can be stored and published by a Registry service. Although we currently use a simple central CTYPE Registry, we imagine various methods for storing and publishing CTYPEs (e.g. using IPFS). While the registry service is not a core part of the protocol, the KILT SDK provides methods to publish and store CTYPEs.



Flow (left) and sequence (top) diagrams of creating a CTYPE and publishing it to the blockchain

Claim Structure

Claims are created locally on the Claimers' client device by choosing a CTYPE. As we have shown previously, the CTYPE scheme (**cType**) defines the structure of the data the Claimer has to fill in (**contents**) when creating the claim. When a Claimer creates a new claim for a CTYPE, then the Claim will be associated with the Claimer's identity (**owner**).

General Claim Structure

```
Claim {
      cType: <Hash of the CTYPE>,
      contents: <Instance of the CTYPE scheme>
      owner: <Address of the owner identity>
}
```

Example Claim of the *DriversLicense* CTYPE

```
{
      "cType": "0x39ffc33202410721743e19082986e650b4e847b85bea7eab77...",
      "contents": {
            "name": "Andreas",
            "age": 38
      },
      "owner": "5CJUL8B3xSaPmAM5vzj427HtrJRBiySezN1aKsJ5Q7aSVde3"
}
```

The contents of the claim are verified against the CTYPE scheme when the claim is created. A JSON Schema based validator is used to do this validation.

Claims are not stored on the blockchain. Claimers decide on their own where they want to store their claims. Most likely this will be a local store on the Claimers device (e.g. local storage in a browser).

Request for Attestation

To get an Attestation from an Attester, the Claimer first has to build a "Request for Attestation" object.

```
RequestForAttestation {
  claim: <Claim>
  claimerSignature: string - The signature of the claimer
  claimHashTree: object - All claim properties hashed with nonces
  ctypeHash: object - The ctype hash, hashed again with a nonce
  legitimations: AttestedClaim[] - Array of AttestedClaim objects of the
                 Attester which the Claimer requests from her to include
                 into the attestation as her legitimations
  delegationId: object (optional) - The id of the DelegationNode of the
                Attester, which should be used in the attestation
  claimHash: object - Root hash of the whole claim, built from the hashes
                      of the claim tree, the ctype hash, the legitimations
                      hashes and the delegationId
  quote: object - quote object (see later)
```

```
  quoteHash: object - Root hash of the whole qute, built from the hashes
                      of the quote
}
```

This object contains the **claim** (and its hashes) and **legitimations**/**delegationId** of the Attester and is signed by the Claimer, to make it tamper proof (**claimerSignature**). By including legitimations/delegation into the object, the Claimer can force the Attester to attest an object under his conditions, or put differently, he can control, under which conditions the Attester has to attest. More details how the Complex Trust Structures are implemented will be described in a later section. The object also includes the claim **hash**, which is generated from a **claimHashTree** (also included in the object). The **RequestForAttestation** object, together with **claimHashTree**, also supports hiding of claim data during a Credential Presentation, which is described in a later section.

The Hash-Tree

The Hash-Tree includes a nonce and a hash for every property of the Claim.

```
ClaimHashTree {
  property1: {
      nonce: string - unique identifier (Version 4 UUID)
      hash: string - hex representation of the resulting hash of the
            property and the nonce combined
  }
  property2: {...}
  ...
  propertyX: {...}
}
```

The value of every property of the contents of a claim is hashed, by generating a nonce (e.g. by generating a UUID) and applying our hash algorithm (blake2b) on a string concatenation of the nonce and the JSON-converted and normalised value. The result is saved in hex form.

For example, we have the property:

```
age: 30
```

We generate a nonce:

```
"f942d230-b1a4-428f-9566-358137a0836c"
```

We convert the value of the property to JSON and normalise it:

```
"30"
```

We concatenate the nonce and the value:

```
"f942d230-b1a4-428f-9566-358137a0836c30"
```

and apply our hashing algorithm:

```
"0x4b0fabbf5ac728e1841ef1884e47cac83718b259025f371932141bfba4004928"
```

The resulting **claimHashTree** might look like this:

```
{
  "age": {
    "nonce": "f942d230-b1a4-428f-9566-358137a0836c",
    "hash": "0x4b0fabbf5ac728e1841ef1884e47cac
             83718b259025f371932141bfba4004928"
  },
```

```
  "name": {
    "nonce": "de58f813-0d28-48a0-bffb-6665db6ba727",
    "hash": "0x26bc08454332540fa94fc0b88745be9
             fef1f500395fbb1b88afcc231ced29e88"
  }
}
```

The **ctypeHash** also gets re-hashed by applying the same method. Let's say we have following **ctypeHash**:

"0x5a9d939af9fb5423e3e283f16996438da635de8dc152b13d3a67f01e3d6b0fc0"

We generate a nonce:

"522ac374-4661-4561-9c16-4ff960369198"

The **ctypeHash** is already in JSON compatible and normalised form, so we concatenate it with the **nonce**:

"522ac374-4661-4561-9c16-4ff9603691980x5a9d939af9fb5423e3e283f16996438da635de8dc152b13d3a67f01e3d6b0fc0"

and apply our hashing algorithm:

"0x286f752a68ff3e6652e8936a97c904a1e8f262e6ddfe080ed25fbba9c43a1a2b"

Result looks as follows:

```
{
  "nonce": "522ac374-4661-4561-9c16-4ff960369198",
  "hash": "0x286f752a68ff3e6652e8936a97c904a
           1e8f262e6ddfe080ed25fbba9c43a1a2b"
}
```

The Claim Hash

The **claimHash** is what makes an attested claim unique and immutable, as it is generated from the hashed claim properties, legitimations, delegations and the **ctypeHash**. To generate the **claimHash**, all these values are collected, converted to byte arrays, concatenated and finally hashed (with blake2b). The resulting **claimHash** is then stored with the request for attestation and used as the identifier for the attestation itself.

Changing (or tampering with) an attested claim will always result in a different claim hash, so in such cases a verification will always fail.

Quote

The Quote allows Attesters to express their requirements and prerequisites to become part of the service contract between the Claimer and the Attester based on which the Attestation is made and paid for. The Attester provides their public address to carry out the Attestation of the Claim (which has the CTYPE **cTypeHash**) specified within the Quote structure. A Quote has versioning for different specs and offering a version for Attesters and Claimers to reference such as changes within the Quote from anything to **cost** and **termsAndConditions** these can directly affect the version of the Quote.

```
IQuote {
 attesterAddress: PublicIdentity['address']
```

```
 cTypeHash: IClaim['cTypeHash']
 cost: ICostBreakdown
 currency: string
 quoteTimeframe: string
 termsAndConditions: string
 version: string
}
```

The cost is broken-down to tax, net and gross.

```
ICostBreakdown {
  tax: number
  net: number
  gross: number
}
```

Terms

Terms are built as an object by an Attester after requesting a quote.

```
ITerms {
  claim: string
  legitimations: object[]
  delegationId?: DelegationNode['id']
  quote?: IQuote
  prerequisiteClaims?: Array<IClaim['cTypeHash']>
}
```

The Quote is used within the terms to show the Claimer the terms. Additionally, it enables us to remove all the prerequisites and to allow them to validate the data without exposing the data of the claim. Once the Claimer has agreed to the Terms, they will sign and send it to the Attester.

Attestation and Revocation

Creating a Credential from a Claim through Attestation

To get a claim attested, the Claimer has to send her claim for attestation to the Attester with a **REQUEST_ATTESTATION_FOR_CLAIM** message (detailed later). The Claimer initiates a connection with an Attester, hashes and signs her claim and sends it to the Attester. This signature proves that the Claimer himself created these contents and prevents creation of claims about an arbitrary identity. The Attester verifies the provided information against the CTYPE scheme, checks the signature and builds an Attestation object. To build the Attestation, the Attester signs the claim hash with his identity, such that everybody can check the ownership of the Attestation. The Attestation is then stored on the blockchain.

Scheme of the Attestation

```
Attestation {
  claimHash: string
  signature: string
  revoked: boolean
}
```

Once the Attestation is built, the Attester submits the Attestation wrapped in an **`AttestedClaim`** object, which we also call the Credential. This attested claim or Credential, which the Claimer can now store locally and give to Verifiers, also contains the original request for attestation.

Scheme of the Credential

```
AttestedClaim {
  attestation: Attestation
  requestForAttestation: RequestForAttestation
}
```

Writing an Attestation to the Blockchain

In KILT protocol, the Attestation object can be written to the blockchain. While the Credentials could be used and verified without writing the corresponding attestations to the chain (for the conceptual details on this issue see Chapter 2), in our current testnet, all Attestations are written to the blockchain by default. This is done to provide an immutable decentralised source of the status (valid or revoked) for the Credential.

Storing an attestation must be performed with the attester's identity (**`owner`**). The Attestation entity is stored in a map on the blockchain, with the **`claimHash`** as the key and a tuple of CTYPE hash, owner address and revoked status as the value (see Node Implementation of Attestation and Revocation section for more details). Once stored on-chain, everybody who has access to the KILT protocol can verify a given attestation, while also being able to trust the time of the attestation (based on the creation of the block it is included in).

Revocation

To revoke an Attestation, the Attestation has to be stored on chain. Once the revocation is invoked, it will set the **`revoked`** flag to **`true`** and update the **`Attestation`** entity on chain. A revoked attestation entity can still be queried from the chain, but a verification check will fail.

Attestation as a service

We imagine that Attesters will be service providers in the KILT ecosystem. This would be greatly supported if the Attesters could make clear, which CTYPEs they attest. Although this functionality is not currently part of the protocol, as a first approach, we propose that this function could be part of the Contact service (explained in the next chapter). In the future, this might be integrated in the protocol itself.

Additionally, an Attester should be able to state that he runs an attestation service and to describe the communication channels through which he runs his attestation service (server IP address, port number, etc.). According to our current plans, we will add this information to our contact exchange service (see next Chapter for a description of services in the KILT ecosystem). However, this information could be also included in the Attesters DID Document. This proposed solution could also be used for Verifiers who run a verification service. The exact details of such an implementation will be designed and described at a later stage of the development of the KILT network.

Node Implementation of Attestation and Revocation

The KILT Blockchain node runtime defines an Attestation module exposing functions to

- add an attestation (`add`)
- revoke an attestation (`revoke`)
- lookup an attestation (`lookup`)
- lookup attestations for a delegation (used later in Complex Trust Structures)

on chain.

Add

The `add` function takes following parameters:

- `attester`: The caller of the method, i.e. public address (ss58) of the Attester
- `claimHash`: The Claim hash as blake2b string used as the key of the entry
- `CTYPEHash`: The blake2b hash of CTYPE used when creating the Claim
- `delegationId`: Optional reference to a delegation which this attestation is based on (see Complex Trust Structures for detail)

The node verifies the transaction signature and insert it to the state, if the provided attester didn't already attest the provided `claimHash`. The attestation is stored by using a map:

```
claimHash => (CTYPEHash, attester, delegationId (optional), Revoked)
```

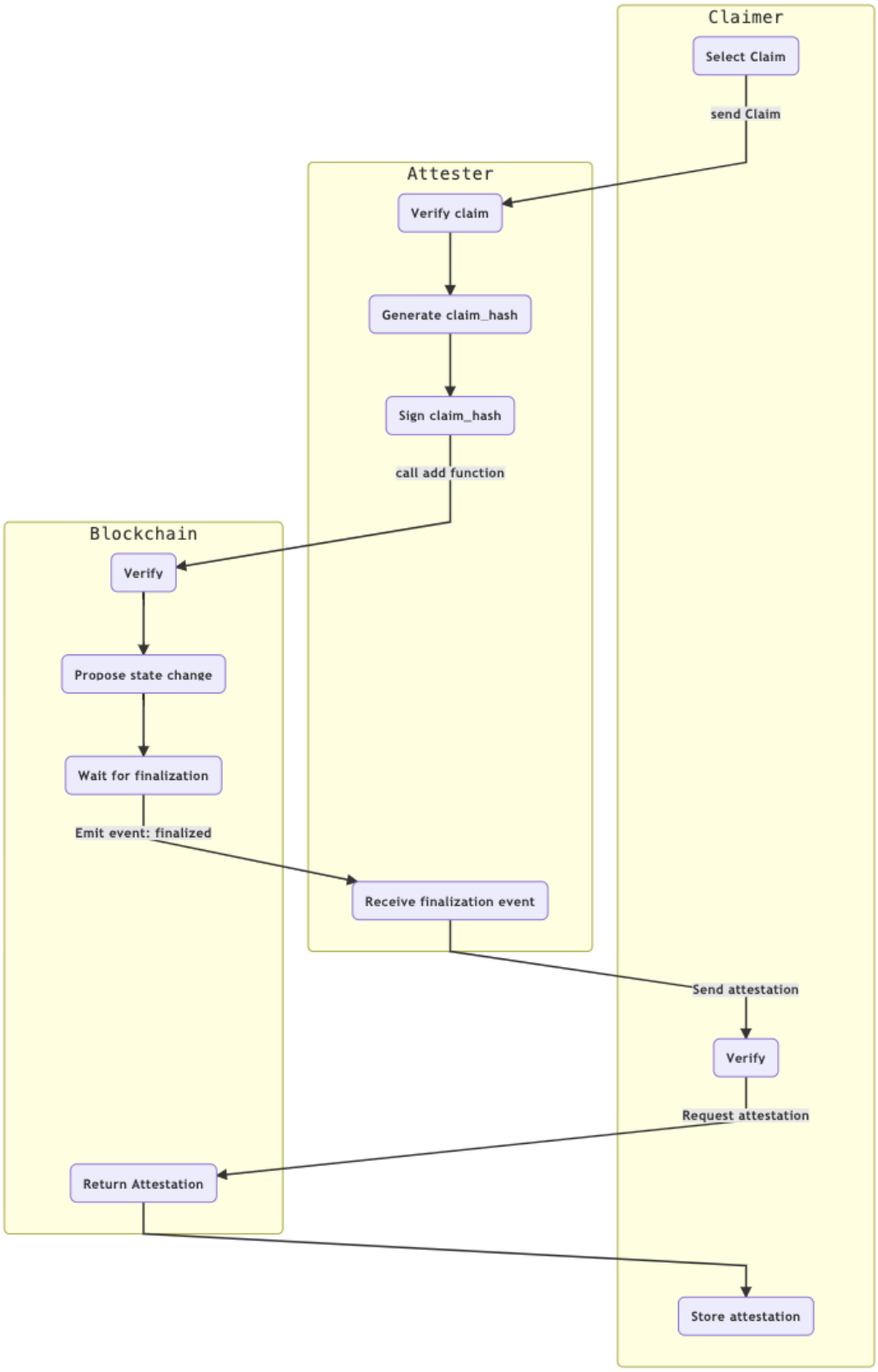
Revoke

The `revoke` function takes the `claimHash` (which is the key to lookup an attestation) as argument. After looking up the attestation and checking invoker permissions, the revoked flag is set to true and the updated attestation is stored on chain.

Lookup

The attestation `lookup` is performed with the `claimHash`, serving as the key to the attestation store. The function `get_attestation(claimHash)` is exposed to the outside clients and services on the blockchain for this purpose.

Similarly, as with the simple `lookup`, to query all attestations created by a certain delegate, the runtime defines the function `get_delegated_attestations(DelegationNodeId)` that is exposed to the outside.



Attesting a Claim and sending back the resulting Credential to the Claimer

Verifying a Credential

To verify a Claim, the Verifier requests Attested Claims for a certain CTYPE from the Claimer by sending an **REQUEST_CLAIM_FOR_CTYPE** message to the Claimer. The Claimer selects the locally stored attested claims and sends them in a **SUBMIT_CLAIM_FOR_CTYPE** message back to the Verifier.

Once the attested claim(s) have been sent, on the Verifier side each attested claim is verified. Assuming the Verifier trusts at least one of the Attesters, the process entails:

- re-hashing and comparing submitted data
- checking hashes of a presentation
- checking if the attestation matches to the request
- verifying that the attestation is stored on chain and non-revoked.

Credential Presentation

For each selected attested claim, the Claimer can select parts of the Claim data she would like to present to (or hide from) the Verifier. This is the process where the Claimer creates a *presentation* of her attested claim.

For every part of an attested claim (e.g. claim fields/data, CTYPE hash, legitimations, delegation) a nonce is created, and the contents of the specific part together with its nonce is hashed using the blake2b hashing algorithm. The nonce is created to prevent predictable hashes for a given data (like true/false fields). All hashes are combined together and are hashed to build the **claimHash**. For a partial Credential presentation, the nonce and data of one or more parts are simply deleted from the attested claim, but the hash of the respective fields are left intact. When building the proof over a partial Credential, all remaining parts and their nonces are checked against their hashes and all hashes (containing also the hashes of deleted parts) are checked if they build up the **claimHash**. The claim hash can then be checked to be signed correctly by the claim owner and attester. This proves that the revealed parts of the partial claim were originally claimed by the claim owner and attested by the attester.

Complex Trust Structures

Attesting a Claim with Legitimations

The Claimer can request the Attester to include legitimations in the Attestation. To do so, the Claimer starts the attestation workflow by requesting the Attesters legitimations for a given CTYPE (**REQUEST_TERMS**). Once the Attester gives her legitimations to the Claimer (**SUBMIT_TERMS**), she can include them in the request for attestation.

```
RequestForAttestation {
  ...
  legitimations: AttestedClaim[] - Array of AttestedClaim objects requested
                 by the Claimer of the Attester to include as legitimations
  ...
}
```

Indeed, a legitimation is a Credential and thus represented by the `AttestedClaim` type. This means, we can handle legitimations as any attested claim, verify it, create presentations of it and so on. Since an `AttestedClaim` also contains the original request for attestation, which in turn contains legitimations, we can have an endless nested structure of legitimations. As legitimations are also just credentials presented to the Claimer (and later the Verifier of the claim) they could also have hidden fields, i.e. contain only parts of the claim data. As any attested claim, legitimations may also contain further legitimations.
As legitimations also may contain data a Claimer does not want to present to every Verifier, it is optional to include the data of the `AttestedClaim` itself, unless the claim's hash remains in the legitimation object to build up the root hash for verification.

When a Claimer includes legitimations in a `RequestForAttestation`, those legitimations are also included in the generated `claimHash`. Legitimations are not checked by the Validator Nodes at all at the time of attestation. The chain offers read functions for all data that is stored (e.g. get a delegation by id). There is no way to add read functions that also implement a checking algorithm, there are only transactions and storage read functions.

Hierarchy of Trust

As described in Chapter 2 the concept of delegations make it possible to implement a Hierarchy of Trust for making attestations. On the lowest level, a delegation structure is always a tree with a root delegation that is referencing a certain CTYPE.

Since we're dealing with a tree structure, the nodes are implemented in a composite hierarchy. So, there is a `DelegationBaseNode` type from which both the `DelegationRootNode` and default `DelegationNode` inherit.

```
DelegationBaseNode {
  id: string
  account: string
  revoked: boolean
  getRoot(): DelegationRootNode
  getParent(): DelegationBaseNode
  getChildren(): DelegationNode[]
  ...
  verify(): boolean
}
```

The root delegation node has the following structure:

```
DelegationRootNode extends DelegationBaseNode {
  cTypeHash: string
}
```

The default delegation node has the following structure:

```
DelegationNode extends DelegationBaseNode {
  rootId: string
  parentId: string (optional)
  permissions: Permission[]
  generateHash(): string
}
```

On-chain there is a Delegations module exposing store, revoke and lookup functions. To store the root delegation, a client-generated Version 4 UUID (`id`) and the CTYPE hash are passed as arguments. The `id` serves as the lookup to the root node. Each node is also associated with an `account` (owner). For root delegations only the owner is allowed to create first level delegations.

Delegation Node Permissions

To create a delegate node within a tree, a member needs to be the owner of a delegation with the permission to delegate. Each delegation can be configured to have a certain permission (*attest* and/or *delegate*). A delegation is always associated with a root and may be a child of a delegation that has the permission to *delegate* or may not be a child of any delegation, which means that it's the top level under the root.

Inviting to Delegate

Inviting someone as child delegate the inviter sends a `REQUEST_ACCEPT_DELEGATION` message to the invitee. For that the inviter creates a volatile delegation object, selects the permissions (attest/delegate) and the parent delegation, hashes and signs it, and sends it to the invitee. The invitee then can verify and accept the delegation invitation, sign the hash and send back the delegation object to the inviter with `SUBMIT_ACCEPT_DELEGATION`. After that the inviter verifies the signature and stores the delegation with the delegate's signature on chain. Finally the inviter sends back a message to inform the invitee (`INFORM_ACCEPT_DELEGATION`) that the delegation has been created and she can import it to her client application to attest and/or delegate.

Attest as Delegate

A Claimer can request her claim to be attested with a certain delegation. To do so, she sends a `REQUEST_LEGITIMATION` to the Attester. The Attester proves he is allowed to attest the requested CTYPE by sending back his delegation. The Claimer then verifies the sent delegation, includes it in the `REQUEST_ATTESTATION_FOR_CLAIM`, hashes everything and sends it to the Attester.

Now, after the claim has been checked by the Attester, he creates the attestation with the delegation he included. To ensure immutability, we use the `claimHash` (which also includes the `delegationId`) as the lookup key to the attestation.

The Verifier can check, if an attested claim has been attested by a delegate he trusts. The tree nature of KILT delegations makes it possible to resolve a delegation path up to a trusted delegate.

Note that on a higher level an Attester can have multiple delegations (from different higher-level Attesters) within the same Hierarchy of Trust, thus forming a directed acyclic graph (DAG). However, this will be translated to the lower implementation level by creating multiple `DelegationNode` entries for the Attester with different `parentId` fields representing multiple delegation trees from the Attesters perspective.

Revoking Delegations

In KILT any delegation can be revoked. After revoking a delegation, the delegate is no longer able to attest or delegate using that delegation. Delegates with a delegate permission can revoke their own or direct child delegations. Revoking a delegation also revokes all child delegations. Attestations that were created with a delegation that has been revoked later are still valid.

It is also possible to revoke all attestations that were created with a certain delegation.

Node Implementation of the Hierarchy of Trust

The KILT Blockchain node runtime defines a Delegation module exposing functions to

- create a root (**`create_root`**)
- add a delegation (**`add_delegation`**)
- revoke a delegation (**`revoke_delegation`**)
- revoke a whole hierarchy (**`revoke_root`**)
- lookup a root (**`lookup_root`**)
- lookup a delegation (**`lookup_delegation`**)
- lookup children of a delegation (**`lookup_children`**)

on chain.

Create root

The **`create_root`** function takes the following parameters:

- **`owner`**: The caller of the method, i.e. public address (ss58) of the owner of the trust hierarchy
- **`rootId`**: A V4 UUID identifying the trust hierarchy
- **`CTYPEHash`**: The blake2b hash of the CTYPE the trust hierarchy is associated with

The node verifies the transaction signature and insert it to the state. The root is stored by using a map:

```
rootId => (CTYPEHash, owner, revoked)
```

Add delegation

The **`add_delegation`** function takes the following parameters:

- **`owner`**: The caller of the method, i.e. public address (ss58) of the delegator
- **`delegationId`**: A V4 UUID identifying this delegation
- **`rootId`**: A V4 UUID identifying the associated trust hierarchy
- **`parentId`**: Optional, a V4 UUID identifying the parent delegation this delegation is based on
- **`CTYPEHash`**: The blake2b hash of CTYPE used when creating the Claim
- **`delegate`**: The public address (ss58) of the delegate (ID receiving the delegation)
- **`permissions:`** The permission bit set (having 0001 for attesting permission and 0010 for delegation permission)

- **delegateSignature**: ed25519 based signature by the delegate based on the **delegationId**, **rootId**, **parentId** and **permissions**

The node verifies the transaction signature and the delegate signature as well as all other data to be valid and the delegator to be permitted and then inserts it to the state. The delegation is stored by using a map:

delegationId => (rootId, parentId, delegate, permissions, revoked)

Additionally, if the delegation has a parent delegation, the information about the children of its parent is updated in the following map that relates parents to their children:

delegationId => Vector(delegationId)

Revoke

The **revoke** function takes the **claimHash** (which is the key to lookup an attestation) as argument. After looking up the attestation and checking invoker permissions, the revoked flag is set to true and the updated attestation is stored on chain.

Lookup

The attestation **lookup** is performed with the **claimHash**, serving as the key to the attestation store. The function **get_attestation(claimHash)** is exposed to the outside clients and services on the blockchain for this purpose.

Similarly, as with the simple **lookup**, to query all attestations created by a certain delegate, the runtime defines the function **get_delegated_attestations(DelegationNodeId)** that is exposed to the outside.

Communication and Messaging

All messages are encrypted with the encryption keys of the involved identities. An **EncryptedMessage** is composed of the encrypted **MessageBody** and surrounding data.

```
MessageBody {
      type: string - a message type, which describes what the
                     content will look like
      content: object - an object containing data according to the
                        message type
}

EncryptedMessage {
  message: string - encrypted  MessageBody
  nonce: string - nonce used to encrypt MessageBody
  createdAt: number - unix timestamp in milliseconds
  hash: string -  blake2b hash of message + nonce + createdAt
  signature: string - hash signed with senders ed25519 signing key
  receiverAddress: string - ss58 address of receiver
  senderAddress: string - ss58 address of sender
}
```

Timestamp Proofs

Every time someone sends data about an identity, he or she has to sign the message together with the challenge number (a nonce and the timestamp) to prove access to the corresponding private key. The reason for adding a timestamp is to prevent the replay attack of messages containing the same data. If one gets access to a message that he could send later on to fraud the original sender, the receiver could check and prove with the timestamp that the message has been created earlier or especially before someone requested something that this message is the answer to. As the malicious actor is not possessing the private signature key and the timestamp is part of the signature, he is not able to change the timestamp and still having a valid signature on the message.

Message Types

We have defined specific message formats on the protocol level for core mechanisms in KILT (e.g. claiming-attestation process, legitimation, Credential presentation etc.). In the following we show simple examples of these messages.

REQUEST_ATTESTATION_FOR_CLAIM

With this message the Claimer can request an attestation from an Attester.

```
{
  "type": "request-attestation-for-claim",
  "content": {
    "claim": {
      "cType": "0xd3b3...4cbf",
      "contents": {
        "name": "Claimer",
        "age": "29"
      },
      "owner": "5GZ1ri8q2h7hXJHe9CnJVMAvGdRi3rbrrxfYBNHLwzH55daF"
    },
    "ctypeHash": {
      "nonce": "b7983808-af5d-48cb-b19b-be91796ba810",
      "hash": "0x1d11...3643"
    },
    "legitimations": <Array of AttestedClaim objects>,
    "claimHashTree": {
      "name": {
        "nonce": "f6ef7ce4-f2e7-4db5-9642-457aff510327",
        "hash": "0x42bf...69a3"
      },
      "age": {
        "nonce": "46deed19-d1f5-4267-adb1-20f051498040",
        "hash": "0x04f2...1b28"
      }
    },
    "hash": "0x69ab...09c2",
    "claimerSignature": "0x18c6...1605"
  }
}
```

SUBMIT_ATTESTATION_FOR_CLAIM

Through this message format the Attester can reply to the Claimer, sending back the issued Credential, i.e. the attested claim.

```
{
  "type": "submit-attestation-for-claim",
  "content": {
    "request": {
      "claim": {
        "cType": "0xd3b3...4cbf",
        "contents": {
          "name": "Claimer",
          "age": "29"
        },
        "owner": "5GZ1ri8q2h7hXJHe9CnJVMAvGdRi3rbrrxfYBNHLwzH55daF"
      },
      "ctypeHash": {
        "nonce": "b7983808-af5d-48cb-b19b-be91796ba810",
        "hash": "0x1d11...3643"
      },
      "legitimations": <Array of AttestedClaim objects>,
      "claimHashTree": {
        "name": {
          "nonce": "f6ef7ce4-f2e7-4db5-9642-457aff510327",
          "hash": "0x42bf...669a3"
        },
        "age": {
          "nonce": "46deed19-d1f5-4267-adb1-20f051498040",
          "hash": "0x04f2...1b28"
        }
      },
      "hash": "0x69ab...09c2",
      "claimerSignature": "0x18c6...1605"
    },
    "attestation": {
      "cTypeHash": "0xd3b3...4cbf"
      "owner": "5H7xkU126e8MnjYymUa9rHKiMKjEWndJMRTiHju3saacHM8X",
      "claimHash": "0x69ab...09c2",
      "delegationId": "0x31366336633336332d373962322d"
      "revoked": false
    }
  }
}
```

REQUEST_CLAIM_FOR_CTYPE

This is the message where the Verifier can specify the CTYPE of a Credential it accepts.

```
{
  "content": "0xd3b3...4cbf", // hash of the specific CTYPE
  "type": "request-claims-for-ctype"                              }
```

SUBMIT_CLAIM_FOR_CTYPE

This is the message format for presenting (sending) a Credential to a Verifier.

```
{
  "content": [
    {
      "request": {
        "claim": {
          "cType": "0xd3b3...4cbf",
          "contents": {
            "name": "Claimer"
          },
          "owner": "5GZ1ri8q2h7hXJHe9CnJVMAvGdRi3rbrrxfYBNHLwzH55daF"
        },
        "ctypeHash": {
          "nonce": "b7983808-af5d-48cb-b19b-be91796ba810",
          "hash": "0x1d11...3643"
        },
        "legitimations": <Array of AttestedClaim objects>,
        "claimHashTree": {
          "name": {
            "nonce": "f6ef7ce4-f2e7-4db5-9642-457aff510327",
            "hash": "0x42bf...69a3"
          },
          "age": {
            "hash": "0x04f2...1b28"
          }
        },
        "hash": "0x69ab...09c2",
        "claimerSignature": "0x18c6...1605"
      },
      "attestation": {
        "cTypeHash": "0xd3b3...4cbf"
        "owner": "5H7xkU126e8MnjYymUa9rHKiMKjEWndJMRTiHju3saacHM8X",
        "claimHash": "0x69ab...09c2",
        "delegationId": "0x31366336633336332d373962322d"
        "revoked": false
      }
    }
  ],
  "type": "submit-claims-for-ctype"
}
```

REQUEST_TERMS

Message format used for a Claimer to request legitimations from an Attester.

```
{
 "messageId":"cca66900-a51d-4697-9334-c1be80 ...",
 "receivedAt":1554468621207,
 "body":{
   "content":{
     "cType":"0x4a3981552a77782a1cb4ee924b26 ...",
     "contents":{
       "name":"Alice",
       "Age":30
     },
     "owner":"5CjCBuxqDRknHPRiSVXvNoDsz867N4 ..."
   },
   "type":"request-legitimations"
 },
 "createdAt":1554468621156,
 "receiverAddress":"5CjCBuxqDRknHPRiSVXvNoDsz867N4 ...",
 "senderAddress":"5CjCBuxqDRknHPRiSVXvNoDsz867N4 ...",
}
```

SUBMIT_TERMS

Message format to send legitimations and delegations to a Claimer.

```
{
 "messageId": "bc074220-afb7-4bdc-8654-3748d4 ...",
 "receivedAt": 1554469210277,
 "body": {
   "content": {
     "claim": {
       "cType": "0x4a3981552a77782a1cb4ee924b26 ...",
       "contents": {
         "name": "Alice",
         "age": 30
       },
       "owner": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ..."
     },
     "legitimations": [
       {
         "request": {
           "claim": {
             "cType": "0x4a3981552a77782a1cb4ee924b26 ...",
             "contents": {
               "myProp": 42
             },
             "owner": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ..."
           },
           "ctypeHash": {
             "nonce": "8f16eb14-bf52-4a5c-ba1c-ce9f47 ...",
             "hash": "0xe1673ab99383ab8f124d1bd3526e ..."
           },
           "legitimations": [],
           "delegationId": "0x31366336633336332d373962322d ...",
```

```
            "claimHashTree": {
              "name": {
                "hash": "0x8af2647c8fce665931608edbbdfd ..."
              },
              "age": {
                "nonce": "eb43980c-52fa-4f1f-9c72-71414c ...",
                "hash": "0xa69b7851fd5ac9729c93e986ca0b ..."
              }
            },
            "hash": "0x744a722328071bea6a8f4b45c8ce ...",
            "claimerSignature": "0x1a6afb333ebeef8499e1b5f02ee3 ..."
          },
          "attestation": {
            "owner": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ...",
            "claimHash": "0x744a722328071bea6a8f4b45c8ce ...",
            "cTypeHash": "0x4a3981552a77782a1cb4ee924b26 ...",
            "delegationId": "0x31366336633336332d373962322d ...",
            "revoked": false
          }
        }
      ],
      "delegationId": "0x31366336633336332d373962322d ..."
    },
    "type": "submit-legitimations"
  },
  "createdAt": 1554469210194,
  "receiverAddress": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ...",
  "senderAddress": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ..."
}
```

REQUEST_ACCEPT_DELEGATION

Message to invite someone as delegate in a delegation tree.


```
{
 "messageId": "3260c795-b07f-468c-ba3e-b11f46 ...",
 "receivedAt": 1554470253615,
 "body": {
   "content": {
     "delegationData": {
       "account": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ...",
       "id": "0x36373566643930302d633530332d ...",
       "parentId": "0x31366336633336332d373962322d ...",
       "permissions": [
         1,
         2
       ]
     },
     "metaData": {
       "alias": "my delegation"
     },
     "signatures": {
       "inviter": "0xb517d62504b1937c1cb37a224e95 ..."
     }
   },
   "type": "request-accept-delegation"
 },
 "createdAt": 1554470253571,
 "receiverAddress": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ...",
 "senderAddress": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ..."
}
```

SUBMIT_ACCEPT_DELEGATION

Message sent to accept the invitation to be a delegate in a delegation tree.


```
{
 "messageId": "7317afa0-8abf-49b7-83d0-448b71 ...",
 "receivedAt": 1554472048612,
 "body": {
   "content": {
     "delegationData": {
       "account": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ...",
       "id": "0x36373566643930302d633530332d ...",
       "parentId": "0x31366336633336332d373962322d ...",
       "permissions": [
         1,
         2
       ]
     },
     "signatures": {
       "inviter": "0xb517d62504b1937c1cb37a224e95 ...",
       "invitee": "0xfc429958cfb6aea183949f4f5ccf ..."
     }
   },
   "type": "submit-accept-delegation"
 },
 "createdAt": 1554472048557,
 "receiverAddress": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ...",
 "senderAddress": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ..."
}
```


INFORM_ACCEPT_DELEGATION

Message to inform the invitee (now delegate) that a delegation has been created for her.


```
{
 "messageId": "e6c13b38-6230-4137-8281-6ac3cc ...",
 "receivedAt": 1554472161811,
 "body": {
   "content": "0x36373566643930302d633530332d ...",
   "type": "inform-create-delegation"
 },
 "createdAt": 1554472161579,
 "receiverAddress": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ...",
 "senderAddress": "5CjCBuxqDRknHPRiSVXvNoDsz867N4 ..."
}
```

KILT Blockchain

The KILT Blockchain is the heart and soul behind KILT protocol. It provides the immutable transaction ledger for the various processes in the network.

Building on the Parity Substrate Blockchain Framework

During our first whiteboard phase, we were thinking about developing the KILT protocol on Ethereum smart-contracts, but we realised that we would have less freedom of setting transaction costs, while incurring a high level of overhead. Instead, we started our development on Parity Substrate, a general blockchain framework, and built up the KILT Blockchain from scratch based on its module library.

Building our blockchain on Parity Substrate has multiple advantages. Substrate has a very good fundamental architecture and codebase created by blockchain experts. Substrate framework is developed in Rust, a memory efficient and fast compiled system programming language, which provides a secure environment with virtually no runtime errors. Moreover, the node runtime is also compiled to WebAssembly, so older version native nodes can always run the latest version node runtime in a WebAssembly virtual machine to bypass the problem of a blockchain fork. Importantly, there is a vibrant developer community and rich documentation.

Our implementation is based on the substrate-node-template library (skeleton template for quickly building a substrate based blockchain), which is linked to the main Substrate codebase.

Remote Procedure Calls

The Ethereum ecosystem highly leverages JSON-RPC where one can efficiently call methods and parameters directly on the blockchain node. Based on good experiences, developers decided to use it in Substrate as well. The Polkadot API helps with communicating with the JSON-RPC endpoint, and the clients and services never have to talk directly with the endpoint.

Blocktime

The blocktime is currently set to 5 seconds, but this setting is subject to change based on further research. We will consider what is affected by this parameter, and in the long term it will be fine-tuned to a setting that provides the best performance and user experience for the participants of the KILT network.

Extrinsics and Block Storage

In Substrate, the blockchain transactions are abstracted away and are generalised as *extrinsics* in the system. They are called extrinsics since they can represent any piece of information that is regarded as input from “the outside world” (i.e. from users of the network) to the blockchain logic. The blockchain transactions in KILT are implemented through these general extrinsics, that are signed by the originator of the transaction. We use this framework to write the KILT Protocol specific data entries on the Substrate based KILT Blockchain: DID, CTYPE hash, Attestation and Delegation. The processing of each of these entry types is handled by our custom Substrate runtime node modules.

Under the current consensus algorithm, authority validator nodes (whose addresses are listed in the genesis block) can create new blocks. These nodes validate incoming transactions, put them into the pool, and include them in a new block. While creating the block, the node executes the transactions and stores the resulting state changes in its local storage. Note that the size of the entry depends on the number of arguments the transaction, (i.e. the respective extrinsic method) has. The size of the block is hence dynamic and will depend on the number and type of transactions included in the new block. The valid new blocks are propagated through the network and other nodes execute these blocks to update their local state (storage).

Consensus Algorithm

Since we are the only authority provider in the testnet phase, we use the simple Aura consensus mechanism. At a later stage, we most likely will change to GRANDPA, which supposedly will be superior to Aura in many aspects. The consensus mechanism is also subject to the future possibility to integrate the KILT network into the Polkadot ecosystem.

Polkadot Integration

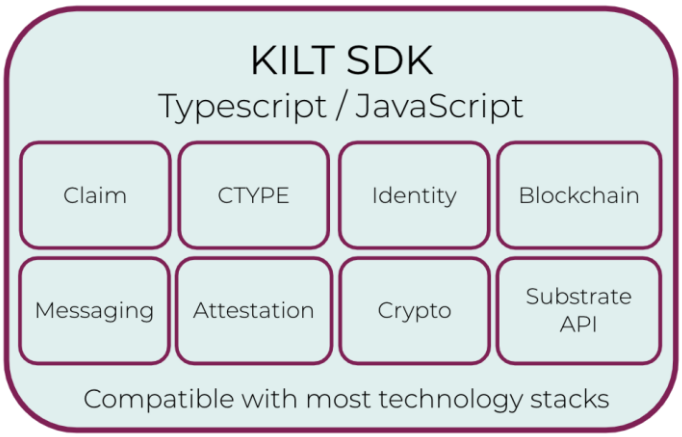
As a further great advantage, by basing ourselves on Substrate, from a technical perspective, we can easily connect to the Polkadot ecosystem. This could provide security for the KILT network by leveraging the global consensus in the Polkadot network. We are planning to integrate KILT into the Polkadot network. It is fairly straightforward to achieve that by simply including specific Substrate modules into the KILT Node implementation. The exact details of this integration is subject to future agreements between Polkadot and KILT and the technological development of Polkadot, Substrate and KILT.

KILT Token

Token transfer is implemented as a balance-based mechanism in Substrate. In our current testnet, every KILT identity can simply request Mash Coins at the KILT Faucet. Importantly, these test tokens will not be usable on our mainnet. After the launch of the mainnet (Spirit-Net) and the public KILT Coin sale, tokens will be available on cryptocurrency exchanges. In describing the functionalities for the Mash Coins and the future KILT Coins together as we have designed them so far, we use the term KILT token.

7.3. KILT SDK

We provide a Software Development Kit (SDK) that implements all KILT protocol functionalities. The SDK contains a complete specification as well as a coherent software library aligned to the current version of the KILT Protocol. This SDK enables application developers to build powerful services and applications on top of the KILT Protocol without deeper knowledge of blockchain technology. It may also be used as a blueprint for developers to create their own SDK implementation for the KILT protocol for a different technology stack than that is currently used in our SDK.



The KILT SDK is written in Typescript, so, on the one hand, it defines strong types for every protocol data scheme and provides classes and functions for all protocol conventions and flows, and, on the other hand, it is compatible with as many platforms (browser, mobile, backend, etc.) as possible, also regarding the coverage of modern software stacks.

From a functional perspective, the SDK as of now is covering all data schemes, conventions and methods described in Chapter 2 to 4 (Top-down Trust Structures, KILT Economy, Claim Standardisation) but does not implement concepts and processes discussed in Chapter 5 (Token Economy) & 6 (Bottom-up Trust). From a non-functional perspective, the SDK focuses on being accessible to a broad developer community (full documentation with examples) and being very stable (thus aiming for a full test coverage). Currently, there is no explicit focus on computation or memory efficiency, batch-processing or other performance related criteria. However, according to our preliminary assessment, the SDK will serve sufficiently for most common applications and may serve as a base or blueprint for other applications using or implementing the KILT protocol.

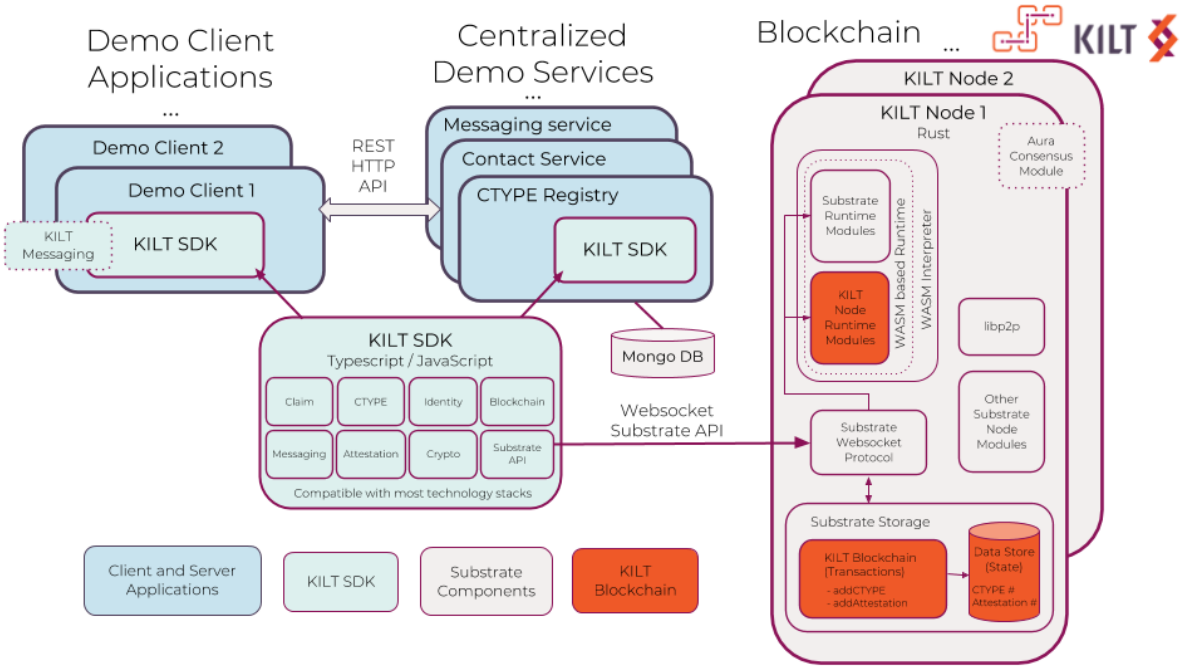
We also provide a reference implementation for the KILT Protocol (KILT testnet, described in the next chapter) that serves as a preliminary environment for developers and stakeholders interested in the KILT ecosystem. To demonstrate a large set of KILT Protocol use cases there is also a web-based demo client that uses nearly all SDK functionality. Additionally, the client makes use of a server-based backend component that provides a CTYPE and contact registry as well as a messaging service (see next chapter for details).

8. KILT Network Launch Roadmap

This chapter gives an overview of the proposed roadmap for the gradual launch of the KILT network. We describe how we imagine the step-by-step build out of the KILT ecosystem with a functioning blockchain, some handy adjacent services and a sample client application, which together can be used to play around or as a toolkit for developing new applications. This section also points out the restrictions of the various stages of the KILT network (testnet, persistent testnet, mainnet).

8.1. Testnet Overview

Currently, KILT is implemented in a testnet stage, which consist of different building blocks and all parts of the system is deployed in Amazon Web Services (AWS).



KILT Blockchain

KILT Network is built upon a lightweight special blockchain based on the Substrate framework. It contains all essential functions described in Chapters 2-4. Details of the current implementation is described in the previous chapter.

Demo Client

We implemented a simple client application that employs the modules of the KILT Protocol SDK. This client includes a cryptocurrency and credential wallet. Users can create KILT identities, claims, request and provide attestations, revoke and verify credentials. Also, they can build up and manage complex trust structures described in Chapter 2.

Centralised Demo Services

To bootstrap the KILT ecosystem, we develop and provide some simple services to enable the seamless onboarding and use of the KILT protocol for early adopters. The repository of these services can be found at: https://github.com/KILTprotocol/prototype-services

The current KILT Demo Services fulfil two purposes, to make the demo client work and to show how service ecosystem around the KILT Protocol would be built up. There are sample implementations for three service types as described below. All service types are implemented in Typescript using node.js as a service framework and MongoDB to store data. To simplify the demo ecosystem, all services are currently implemented as a centralised solution.

CTYPE registry service

To use the KILT protocol, all participants need to share CTYPE definitions as this is the meta model for all claims. A hash of the CTYPE is written to the KILT Blockchain but the data itself needs to be passed from the CTYPE creator to all other participants in some way.

The CTYPE service takes a CTYPE definition, checks its state on the chain and writes it to the database. In the future, there could be decentralised, centralised or even offline ways to distribute the CTYPE.

Messaging service

The messaging service offers a way to send a message from a sender to a receiver in a secure way. Currently, the KILT Protocol works over this messaging service and users can communicate over this central service through which they can send each other KILT related messages. The demo client uses the Crypto and Messaging modules of the SDK to encrypt a message before sending it to the service, so only the receiver is able to decrypt it. The messaging services adds a message identifier and a “received-at” timestamp to the message. The receiver is able to fetch and delete the message from this service.

Contact service

Besides CTYPEs and messages the KILT protocol also requires users to get to know each other. To really simplify demoing use cases of the KILT protocol with the client identities may register themselves after being created to the contact service with an alias. Of course, in real applications there will be many ways to get in contact and exchange information like public keys. The centralised demo contact service is more like a public telephone book, where everybody is registered, and users can find Attesters and Verifiers and learn their public keys or DIDs.

Building new services for KILT

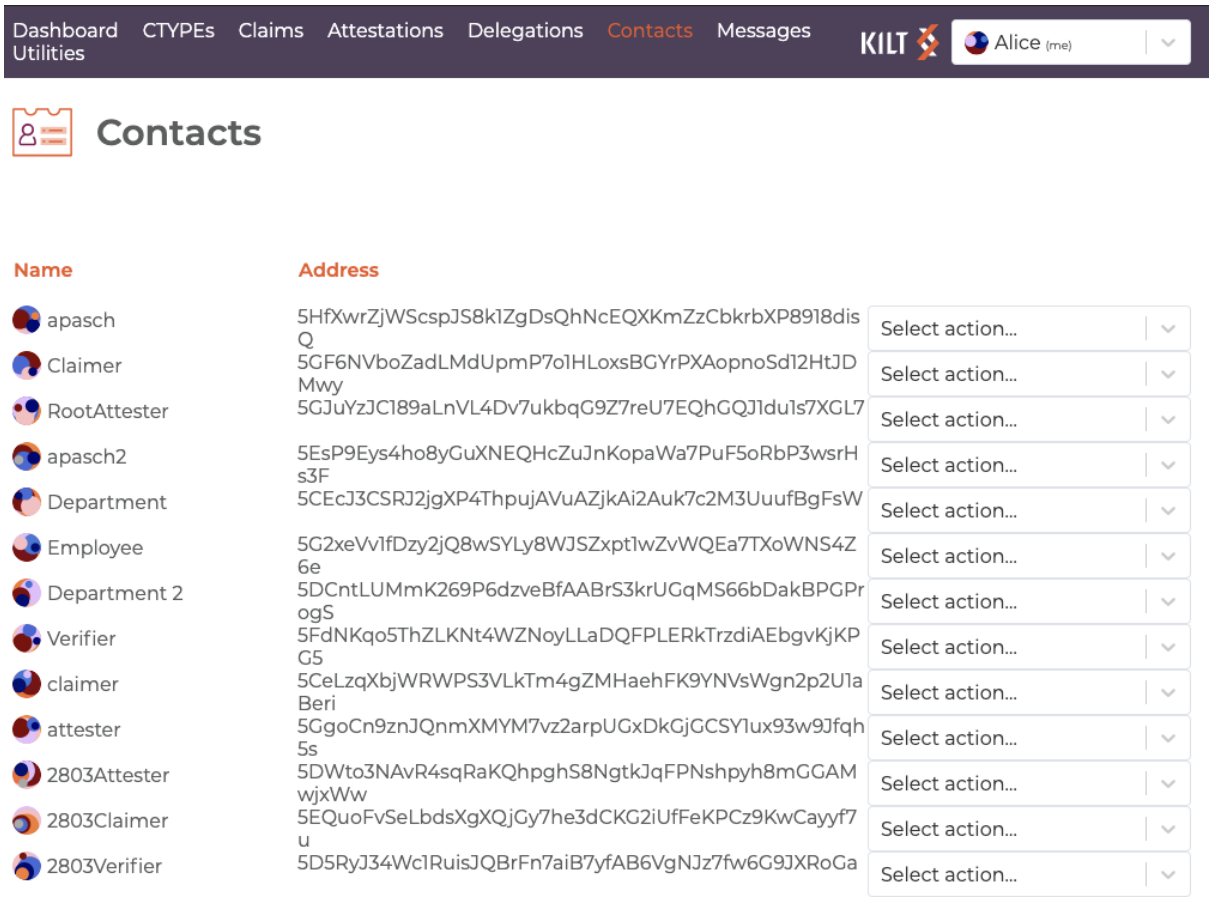
Functionally these services are not designed to be a blueprint for any other KILT service implementation as they for sure lack basic functional and non-functional requirements (like security) and add a very centralised component, whereas a decentralised solution would serve the purpose far better. The service implementations provided are meant for demonstration purposes only, and therefore not to be used in any production system without carefully adapting them to the applications requirements on security, scalability etc.

The service components use the KILT SDK on their server side and show how to check integrity, ownership and validity of data passed around. This may be used as an inspiration for other client-server applications where the system is designed to ensure integrity on both sides.

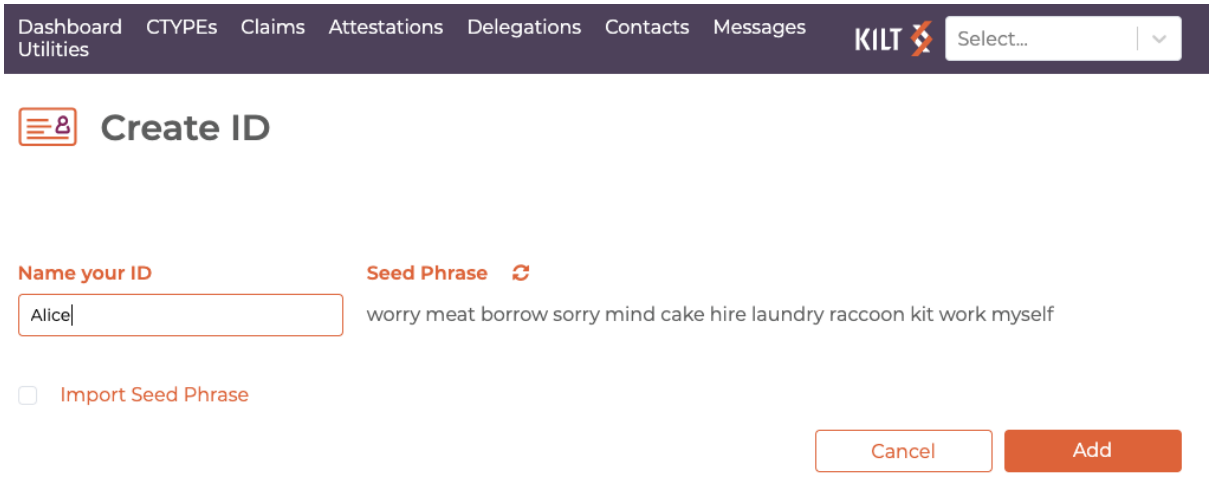
Since our implemented demo services are only simple proposals for additional functionality for the KILT ecosystem, we encourage early adopters to create their own services around KILT. Botlabs plans to run a cloud service for credential storage, and we encourage third parties to support KILT for example in more complex Identity Hub software.

8.2. Screenshots of the Implemented Testnet Ecosystem

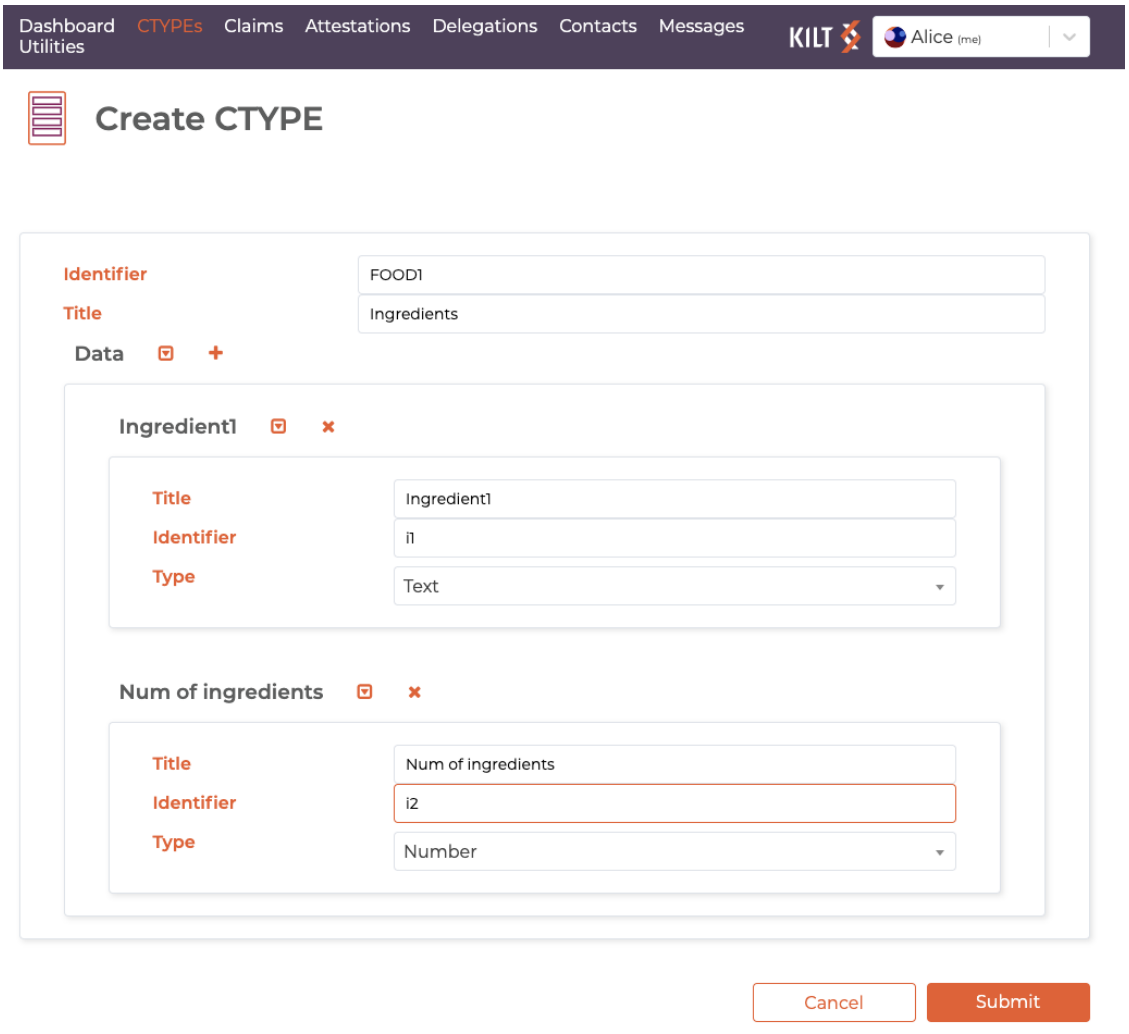
Our testnet ecosystem, shown in the following, currently features all functionality in a Demo Client, using our simple demo services augmented with some third-party blockchain visualisation tools.



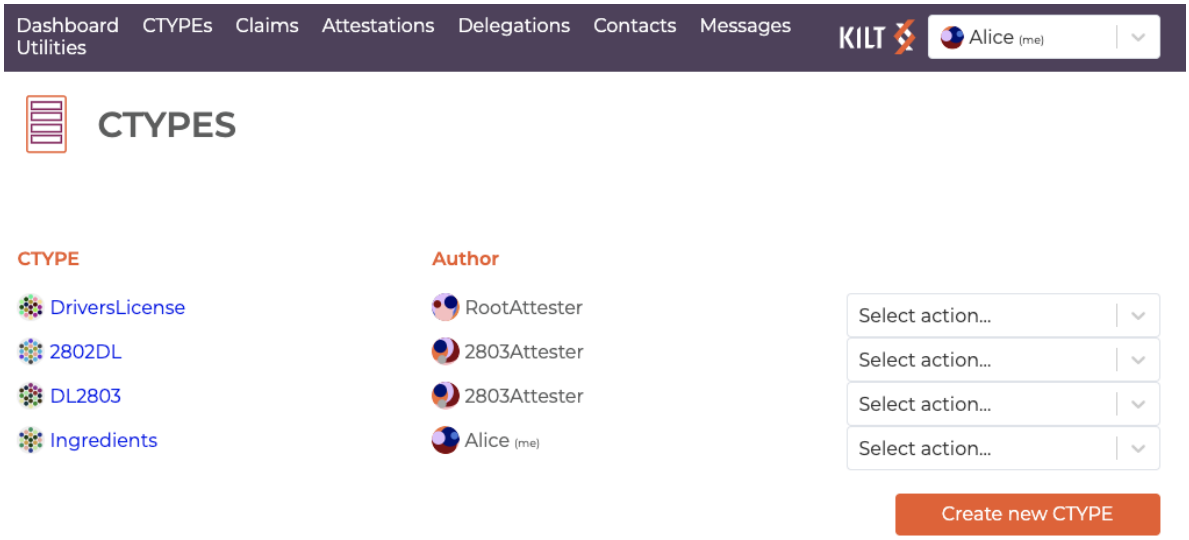
Central Contact Service



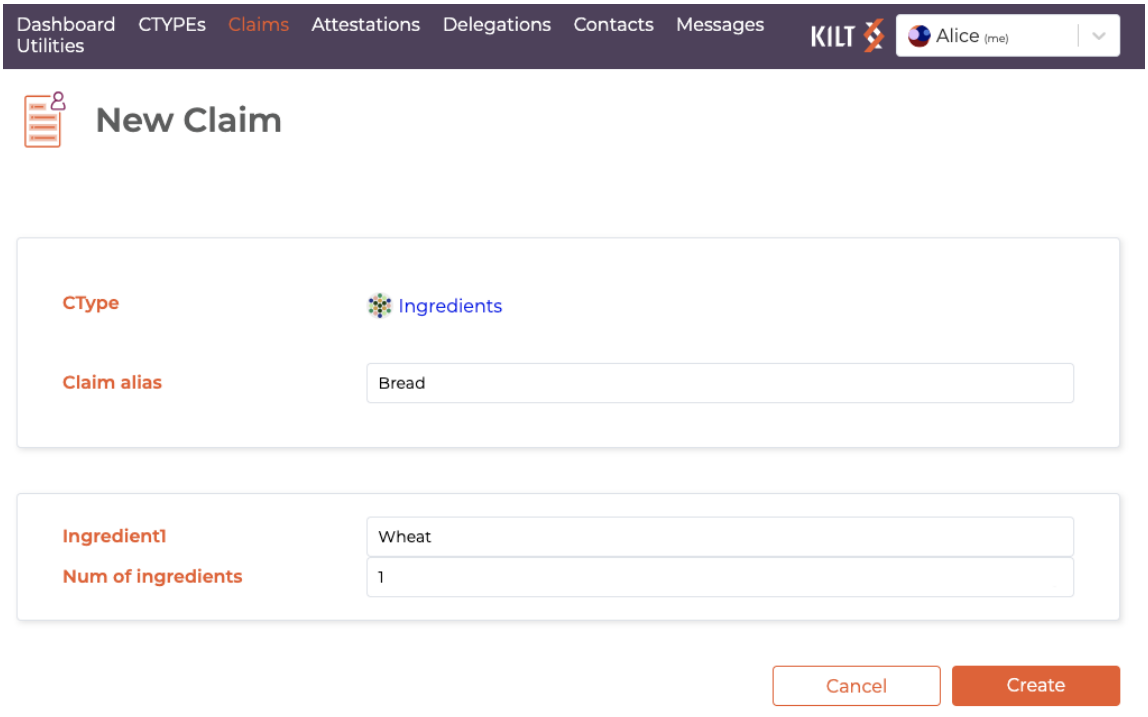
Creating a new KILT ID



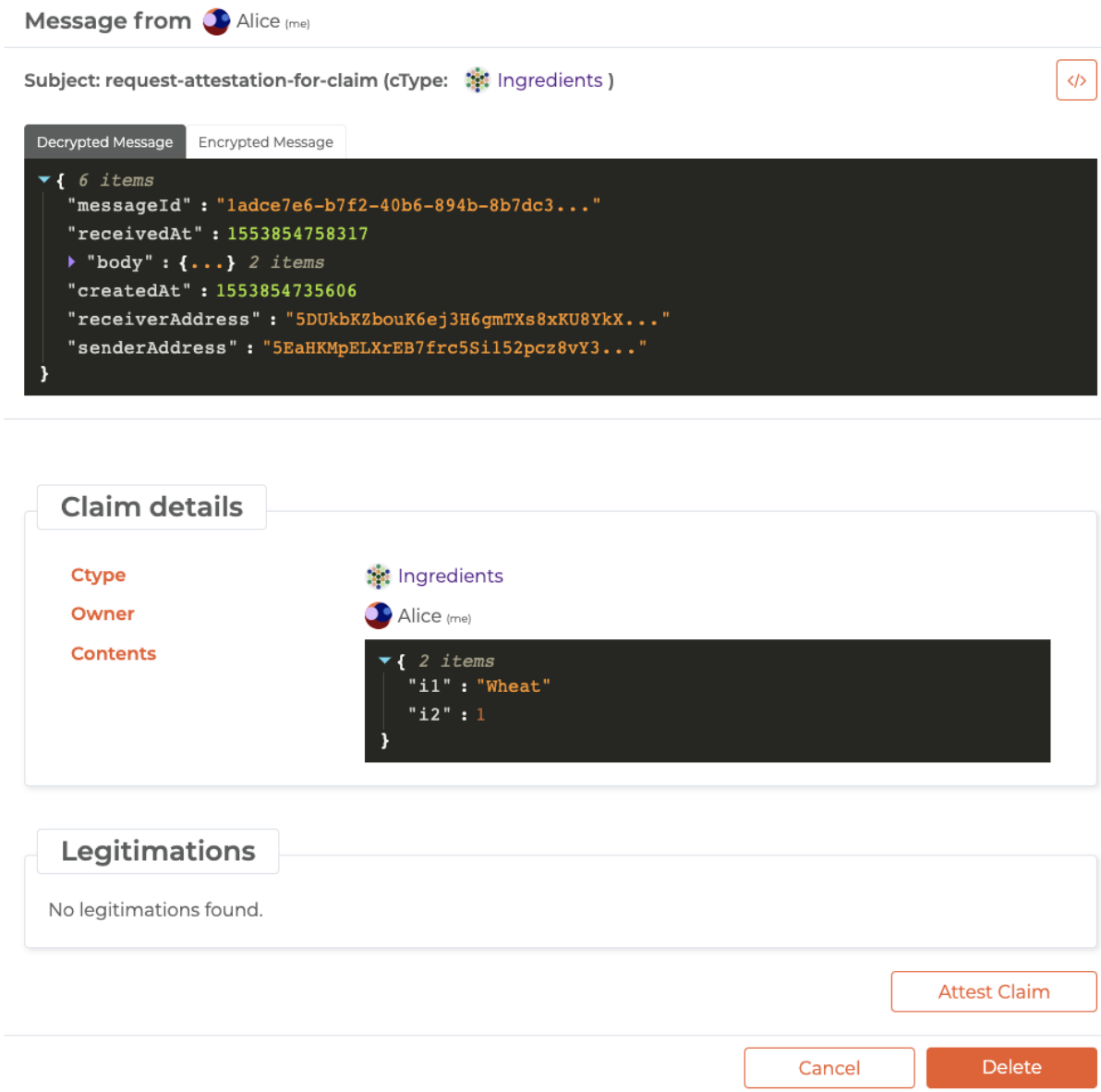
Creating a new CTYPE



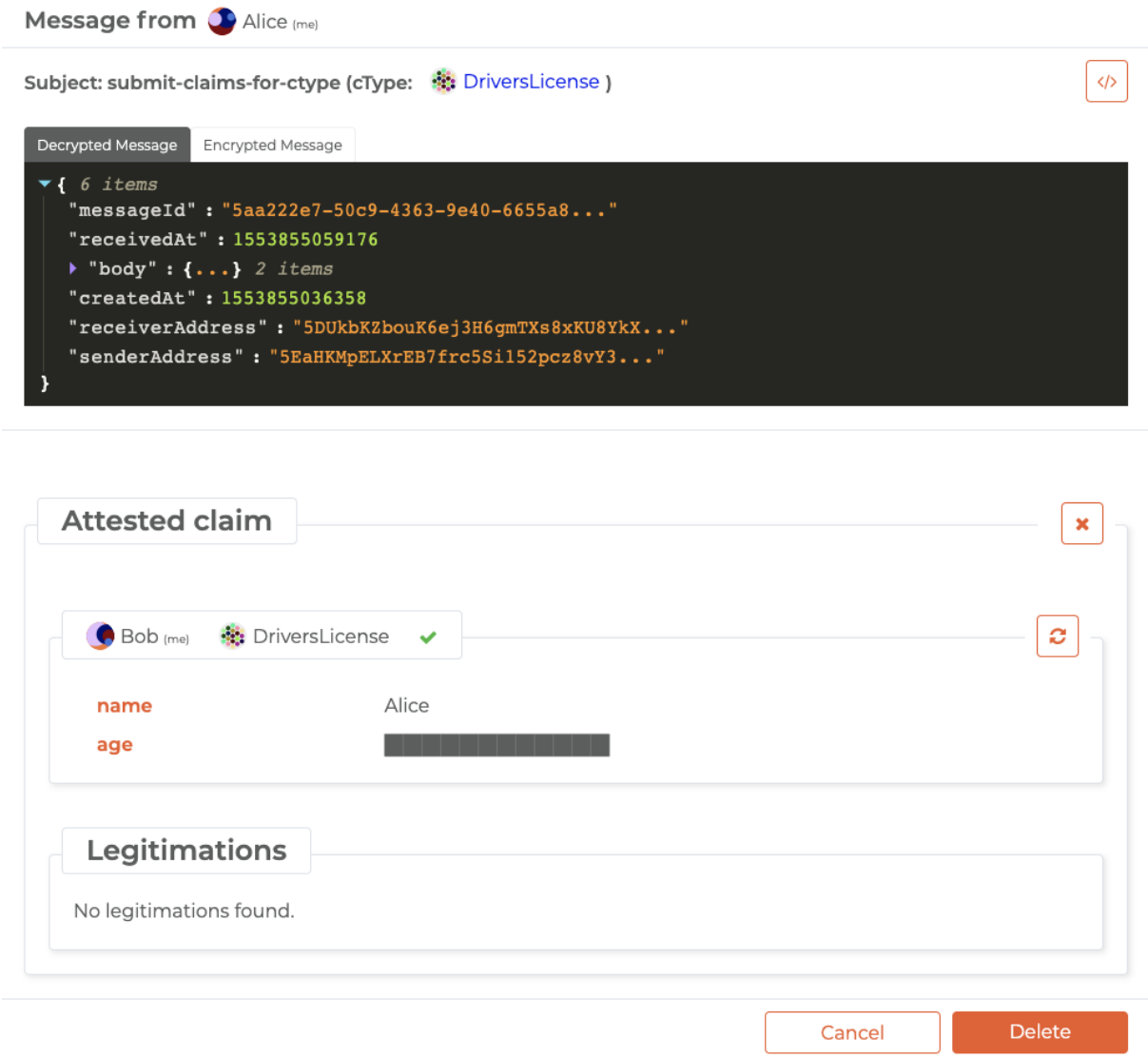
Central CTYPE Service



Creating a new Claim

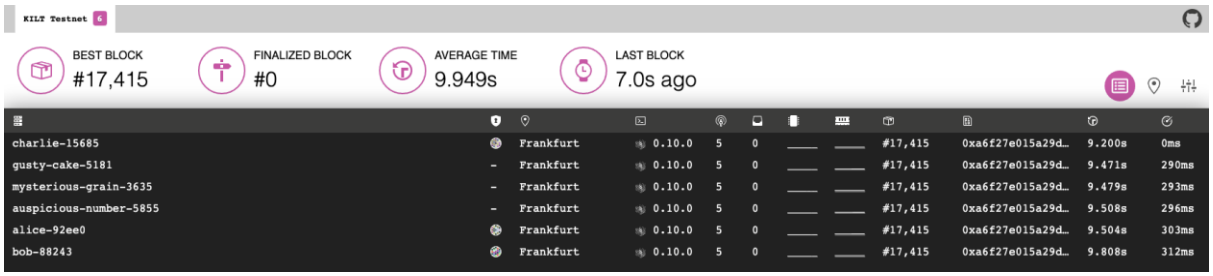


Request for Attestation

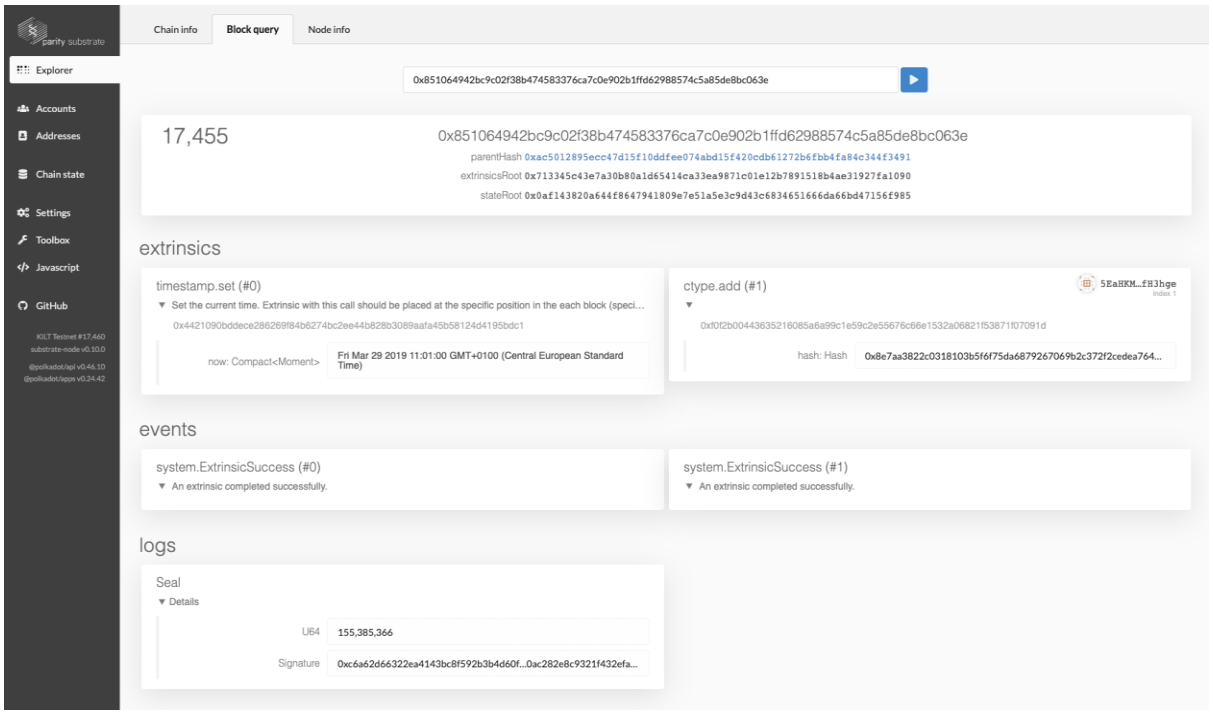


Credential presentation while hiding some fields of the Claim

Telemetry Service



Blockchain Explorer

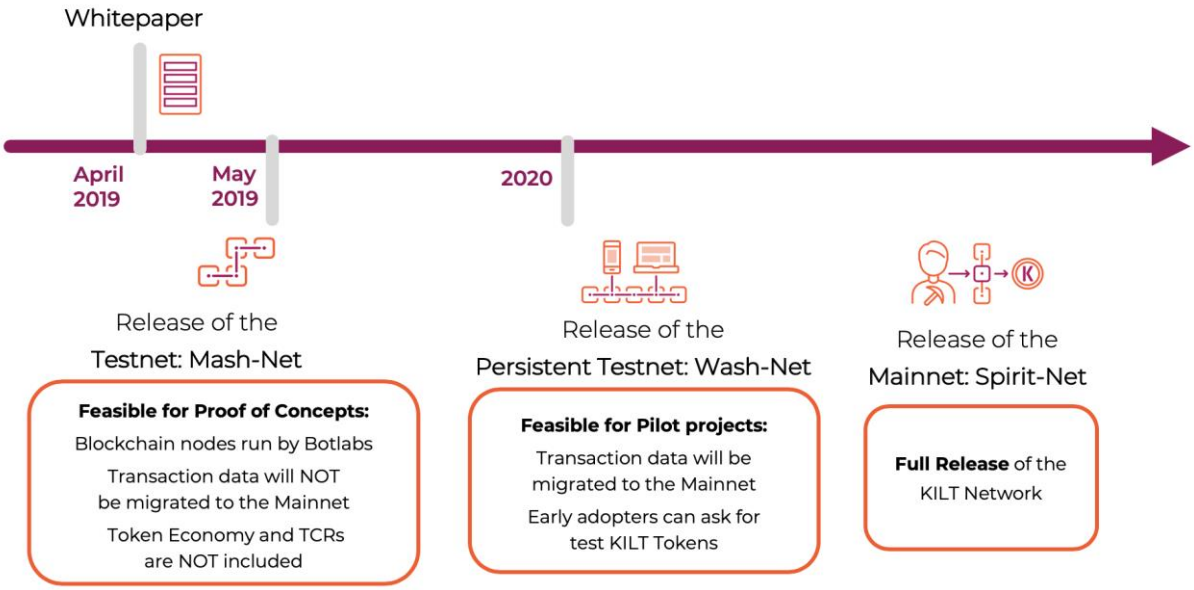


8.3. Release Roadmap

Testnet: Mash-Net

Launch of KILT will be in three steps which we plan as follows. First, Botlabs launched a testnet called Mash-Net in May 2019 which can be used for creating proof of concepts (PoC) by anyone (companies or early implementers and contributors) who is interested in building services and applications on the KILT network. These could be any software projects based on the KILT SDK which are more or less stable throughout the evolution of KILT. Note that, it is always possible to run the KILT testnet locally which gives implementers stability and security in their PoCs and dev environments.
Restrictions of the testnet:

- Play money (a KILT token with similar functionalities as the KILT Coin shall have)
- Transactions written to the testnet chain could be reset anytime.
- No migration of transactions to persistent testnet
- No TCAs
- No Token economy / block rewards
- No connection to Polkadot
- Authority nodes are run Botlabs or by invitation
- Implementers and contributors can run their own node but cannot propose new blocks



Current Release Roadmap for the KILT Network

Persistent Testnet: Wash-Net

The persistent testnet will be called Wash-Net, and it shall be an environment where the transactions shall be safe and preserved for the mainnet. This means you could start productive use of your software with the restrictions that you do not have a running token economy (maybe in an experimental phase) and it will still use play money. So, the persistent testnet is supposed to be transaction safe but not balance safe at that point of time. Launch of the persistent testnet will probably be around 2020 Q2.

Mainnet: Spirit-Net

This shall contain all features described and proposed here in the white paper for the KILT network, with potential changes coming up during the process and additional details of the whole concept we have not figured out yet.

GDPR Considerations

EU Privacy Rules

The General Data Protection Regulation[36] (GDPR) aims to protect the personal data of those living within the borders of the European Union by regulating how such data can be processed by an individual, a company, or an organisation[37]. It is important to understand that the regulation not only applies to companies, individuals, or organisations that are located in the EU but to everyone that offers their products and services to residents of the EU. Although it does not consider the concepts of the blockchain, GDPR and the blockchain share one common goal: the protection of data[38]. In the following paragraph, the most important GDPR rules are introduced. Assessments are based on the Blockchain Bundesverband Position Paper "Blockchain, data protection, and the GDPR"[39].

Definition of Personal Data

The rules defined in the GDPR apply to personal data only. Personal data means any information that relates to an identified or identifiable natural person, also called 'data subject'. An identifiable natural person is one that can be identified directly or indirectly by an identifier, i.e. a name, an identification number, an online identifier, etc. In order to determine if a natural person is identifiable, reasonable means likely to be used for the identification have to be taken into account.

The linkability of information to an individual which enables his identification is crucial to the concept of personal data. Truly anonymous data does not represent personal data. However, pseudonymised data which could be attributed to a natural person by the use of additional information should be considered personal data. This could entail:

- **Public keys**, as soon as they can be associated with a natural person, could considered personal data. If the key does not belong to a natural person, or is not created on behalf of a natural person, or the key cannot be linked to a data subject by reasonable means, then it would not be considered personal data.
- **Hashed data** has been determined as pseudonymous data. However, if the data linking the hashed data to a data subject is kept off-chain and is later erased, the hashed data should once again be considered anonymous.

[36] Regulation (Eu) 2016/679 Of The European Parliament And Of The Council (General Data Protection Regulation), as seen 06th February, 2019, https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:02016R0679-20160504&from=EN,
[37] What does the General Data Protection Regulation (GDPR) govern?, https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-does-general-data-protection-regulation-gdpr-govern_en, as seen 1st February, 2019
[38] How Can Blockchain Thrive In The Face Of European GDPR Blockade?, https://www.forbes.com/sites/darrynpollock/2018/10/03/how-can-blockchain-thrive-in-the-face-of-european-gdpr-blockade/, as seen 1st February 2019
[39] The summary is based on the Blockchain Bundesverband Position Paper "Blockchain, data protection, and the GDPR", as seen 1st February 2019, https://www.bundesblock.de/wp-content/uploads/2019/01/GDPR_Position_Paper_v1.0.pdf,

- **Encrypted data** is considered to be pseudonymous data, as there might be a possibility in the future that encrypted personal data could be linked back to a natural person (e.g. the encryption of the data gets cracked).

General Data Subject Rights

There are three rights laid down in GDPR which we consider here:

- **Right to Erasure (Art. 17)**:
  This basically means that the data subject has the right to have the data controller erase his or her personal data without undue delay. In general, data written on the blockchain is permanent which conflicts with this general principle.
- **Right to Restriction of Processing (Art.18):**
  In general, when the data subject is considering the accuracy or the legitimate grounds for processing his or her personal data in question, then the data controller should automatically restrict the processing of the data (remove from public website, move to a secondary database, etc.).
- **Right to Data Portability (Art. 20):**
  The data subject has the right to receive the personal data concerning him or her which he or she has provided to a controller in a structured, commonly used, and machine-readable format.

These assessments based on the Blockchain Bundesverband Position Paper "Blockchain, data protection, and the GDPR"[40] is not yet a conclusive legal assessment and discussions between the blockchain sphere and the data protection experts are ongoing with the hope that guidelines in line with the GDPR data protection system will be found for Blockchain solutions.

For example there is an ongoing discussion that hashes of personal data could be seen as anonymous data if they were not only to include the personal data but also other randomized data so that even if you knew the personal data you could not know that a certain hash also contains the same personal data. In this case, a hash could also be stored in the blockchain without the GDPR being applicable for it.

All these discussions are just about to begin and there are no binding assessments or decided cases about all these issues available at this point of ime.

How the KILT Protocol protects data

KILT is built on the principles of "Privacy by Design". In particular, KILT does not store any personal data on-chain. In order to achieve this

- The Claimer (who will in many cases be the end user) is in full control of her data
- This implies the Right to Erasure

[40] The summary is based on the Blockchain Bundesverband Position Paper "Blockchain, data protection, and the GDPR", as seen 1st February 2019, https://www.bundesblock.de/wp-content/uploads/2019/01/GDPR_Position_Paper_v1.0.pdf,

- When the Claimer shares Claim or Credential Data with Attesters or Verifiers, these entities have the responsibility to comply with GDPR rules. The protocol cannot enforce this or control the right behaviour.
- The Blockchain only stores hashed data which points to datasets (Credentials) that are under the control of the Claimer. The Claimer is free to erase the datasets so that the hashed data becomes meaningless.

In general, data written to a public blockchain is available to the general public and should be deemed to comply with data portability requirements, but it cannot comply with the right to erasure or right to restrict processing.

The Claimer shall have full control over her credentials containing her personal data as they shall be stored in her data and identity hub under her control and shall not be made available on the blockchain. The identity hub is still to be developed and would be a complex client wallet software system concept which would be controlled by the Claimer that manages her decentralised identifiers (DIDs), handles key creation and recovery methods, and stores credentials. This system shall be comprised of local as well as cloud storage elements which would be synchronised and manage the encrypted and redundant storage of the Claimer's personal data. Identity hubs would be provided by commercial companies and are not part of the KILT Protocol.

KILT does not write Claims containing personal data on the blockchain neither in plain text nor in encrypted form but only in hashed form (combined with random numbers for data protection reasons) inside the on-chain attestations. The Claimer can always decide to delete the claim from her identity hub as well as abandon and never use her private/public key pair again. We describe a simple reference implementation of a client wallet for the KILT Protocol in Chapter 7.

When the Claimer shares her personal data with a third party (Attester or Verifier), the third party might store this data. This however concerns the data management policy of the third party and not KILT Protocol. In this case the third party has to make sure to comply with GDPR wherever applicable. This means that the Claimer is by EU law entitled to forcibly ask everyone who has this data to erase it.
As discussed earlier, it might be possible to relate from a payment made in KILT tokens to the event of an Attestation. Even though the content neither of the Claim nor of the Credential will be disclosed, the event might already be considered personal data. It is considered to introduce Zero Knowledge Payment transactions into KILT in order to prevent this issue.

KILT and the GDPR

Even if the KILT Protocol is designed in a way that should ensure data protection on a very high level, as mentioned above the GDPR and blockchain do not match well and therefore the KILT Protocol does have the same problems with the GDPR as any other blockchain project.

The main source of mismatch is that the GDPR was created with a vision of a central data processor who actively collects data, stores data and has the sole power over the data collected. This system also works for any type of shared data processing where one or more entities hold the power over their stock of data.

Even if there exist translations of this system into the blockchain world, the approach is substantially different from the actual technology based on permissionless and decentralised data processing where none of the involved entities hold the power over the data they process - and this is in essence what makes these decentralised ledger systems safe against attacks who aim to change data for their own benefit.

As Distributed Ledger Technology and the GDPR are both relatively new, there are still many uncertainties about how to match them and neither data protection advisers nor the authorities have found a match that could be used as a standard to ensure to be compliant with GDPR. Therefore our approach is to be as compliant with GDPR's ideas as possible and to protect the personal data of potential users as best as we can, while keeping ourselves up to date with developments and discussions in that field.

Legal Note

The purpose of this white paper is to present the KILT Protocol, the plans for further developments and the technical infrastructure around it to an interested public from today's perspective. The information set forth should not be considered exhaustive and does not imply any elements of a contractual relationship. Its sole purpose is to provide potentially relevant and reasonable information to any developers who think about integrating the KILT Protocol, to any blockchain developers who wish to contribute to the KILT Community and to potential implementation partners who want to get an insight into the current state of the KILT project.

Nothing in this white paper shall be deemed to constitute a prospectus of any sort or a solicitation for investment, nor does it, in any way, pertain to an offering or a solicitation of an offer to buy any securities in any jurisdiction. The document is not composed in accordance with, and is not subject to, laws or regulations of any jurisdiction which are designed to protect investors.

This white paper is a living document subject to constant change. Certain statements, estimates, and financial information contained within this white paper constitute forward-looking, or pro-forma statements, and information. Such statements or information involve known and unknown risks and uncertainties which may cause actual events or results to differ materially from the estimates or the results implied or expressed in such forward-looking statements, even if such statements are not specially marked as unknown or uncertain by an explicit remark or by the grammar or tense used.

Nothing published by the BOTLabs GmbH should be interpreted as investment advice. BOTLabs GmbH is in no way providing trading or investment advice. Please consult with your appropriate licensed professional before making any financial transactions, including any investments related to ideas or opinions expressed, past, present, or future by the aforementioned entity and any future entities that may operate under the parent entities. BOTLabs GmbH does not intend to express financial, legal, tax, or any other advice and any conclusions drawn from statements made by, or on, BOTLabs GmbH shall not be deemed to constitute advice in any jurisdiction. Information is provided for educational and entertainment purposes only.

Imprint

BOTLabs GmbH
Keithstr. 2-4
10787 Berlin
Germany
info@kilt.io info@botlabs.org
https://kilt.io/ https://botlabs.org

Managing Director: Ingo Rübe
AG Charlottenburg HRB 193450 B
VAT-Id No. (USt-IdNr.): DE316284270

Requirements according to § 5 TMG (Germany)

Final Comments

In this document we outlined our current approach to provide credentials for the Web3 by providing a blockchain based fat protocol named KILT. We welcome any comments and feedback about the KILT project at info@kilt.io or any other relevant channel.