CIRCULARISE

WHITEPAPER

PATENT PENDING

Jelle Licht, Tim de Jong, Kaj Oudshoorn, Pietro Pasotti

Abstract

In this whitepaper we present Circularise, a system that facilitates the knowledge transfer required for a circular economy to function. Circularise utilises a combination of blockchain, peer-to-peer technology and cryptographic techniques like Zero-Knowledge Proofs (ZKPs) to build a decentralised information storage and communication platform. The goal is to allow information exchange between participants in value chains while allowing them to remain anonymous and fine-tune the amount of information they want to disclose, and who can access it. Despite the anonymity, the platform contains built-in mechanisms for keeping participants accountable, thus disincentivising the spreading of untruthful information, and for allowing parties to share verifiable information without compromising their anonimity, in a way that requires none of the involving parties to trust one another. Furthermore, participants remain responsible for information pertaining their products questions that become important in the future to be answered about products that were produced in the past.

Contents

1 Introduction

At present, most of the world's economy works in a linear fashion. Raw materials are extracted, products are assembled, more products are assembled out of other products, products are consumed, products are thrown away (i.e. incenerated or landfilled). For each product, the set of parties and processes that these parties perform to create it constitute the product's value chain. A circular economy, as described in Andersen (2007), turns value chains into loops by ensuring that as many "consumed" products as possible are reintroduced into the economy instead of being thrown away. Reintroduction happens through one or more of the circular economy loops: repairing, re-using, refurbishing, recycling, etc.

Any stakeholder in a value chain, whether individual or group, is a *party*. Examples of parties include: a consumer, a recycling company, a television screen manufacturer, a PCB manufacturer, a material manufacturer, a retailer.

Goods and/or information move through the value chain bidirectionally. From any position in the value chain, suppliers are parties that supply you with goods. Recipients are parties that receive your goods. Your direct suppliers/recipients are Tier 1, their direct suppliers/recipients are Tier 2, etc.

Supplier Tier 2	**Supplier Tier 1**	**You**	**Recipient Tier 1**	**Recipient Tier 2**

Figure 1: Value chain terminology: suppliers and recipients described in Tiers. Note that real value chains can involve multiple feedback loops and complex networks of manufacturers, retailers and other parties.

Every party involved in the value chain of a product has access to some information about the value chain: they know who their suppliers are and who their recipients are. Also, they know the transformation of the product that occurs between the input and the output. For example, a television screen manufacturer may know who produces the screens and who produces the remotes, but not who produced the batteries that come with the remote (the party that made the remote may know that). Also, they know that the screen has been assembled into the television chassis in some particular way, using certain materials, etc.

For example, one of the main challenge faced by recyclers is to access product information that they need to do their job.[1]

Use Case 1: Recycling Suppose that a recycling company A obtains an electronic product such as a tablet, and wants to recycle it. It is of great importance, to decide how to recycle the tablet, for A to know whether or not it contains specific (hazardous) materials and/or parts. Currently, the only way to find out is inefficient manual inspection. Circularise will enable the following process: supposing the tablet's manufacturer B is known, so A will need to

[1]Refer to our bluepaper for more information on why recycling is so crucial.

contact B and ask whether in the tablet contains e.g. mercury. B, however, does not know: they only take care of the assembly of parts and they purchase from suppliers C, D, E, F all over the world. So B needs to personally contact C, D, E, F and ask them to get this information for them and report back. These parties may in turn refer B to *their* suppliers, and so on.

The main reasons why this is not presently done already are:

1. The manufacturer B is not always known. E.g. unlabelled, unbranded products.
2. Parties have no incentive to proxy an information exchange chain, as it may pose a risk to their competitive advantage (e.g. their recipients and/or suppliers might cut them out).
3. Most notably, current methods are based on outdated technologies (sending emails back and forth), rendering the process inefficient, unscalable and costly.

Use Case 2: Auditing An auditor is a party that is entitled to check that other parties do not misbehave and that they provide correct and consistent answers. Even though the Circularise system does not provide the auditor with special rights, once a company is made to cooperate through real-world channels, Circularise can be used to empower the auditor. Effectively, Circularise hosts logs about the claims that a party has made. Finding out whether someone is lying about the content of these claims, is the first step to indicating whether a party is misbehaving or not. In addition, if a recycler were to encounter suspicious materials that were stated otherwise in the manifest of the product, an immutable, digital trail is left for the auditor to find the source of mischief.

Auditors fulfil an essential role in the system, as they make sure that real-world consequences are connected to the veracity of claims of all parties. The consequences of misbehaviour provides incentive to be a good citizen. Knowing that misbehaviour is punished increases the system's trustworthiness.

1.1 Problem statement: our goal

More generally, *the challenge we face is to facilitate information sharing in a decentralised network, without centralising the information or the network.* And, because of the nature of the information involved and the network, our solution needs to be reliable. That is, all parties involved need to have certain guarantees about the way our solution works: especially the quality of the information they can obtain through it, the robustness of the framework to changes in the shape of the market (companies coming and going) and its information needs (old information becoming irrelevant, new information becoming essential).

The Goal

As we have argued above, for a circular economy to function it is not only resources which need to circulate, but also information. In fact in any economy, information about products can be even more valuable than products themselves.

We are going to enable companies to share information while staying in control of their secrets, and without the need of trusting anybody – not even us – except the framework we provide.

More generally, circularise needs to *connect information users to information providers*, and that involves: 1) someone in need of information has to find their way to someone who has the information, and 2) a channel has to be set up to allow information exchange.

These basic requirements constitute the foundation of our solution, the Circularise platform. The solution is in fact *a smart question-/answering platform based on a graph structure, where each node represents a party in the value chain of a product, from smelters supplying materials to consumers and recyclers. A platform where information seekers can ask questions, and the system will try and find a path from them to some other party in possession of the answer.*

The nature of the problem however imposes certain restrictions on how exactly we need to implement this generic solution strategy. Our core contribution is to identify these further restrictions and work out a technical solution that overcomes them. In the rest of this section, we walk through Fig. 2 to understand these restrictions and the subgoals that they entail for our solution to be satisfying.

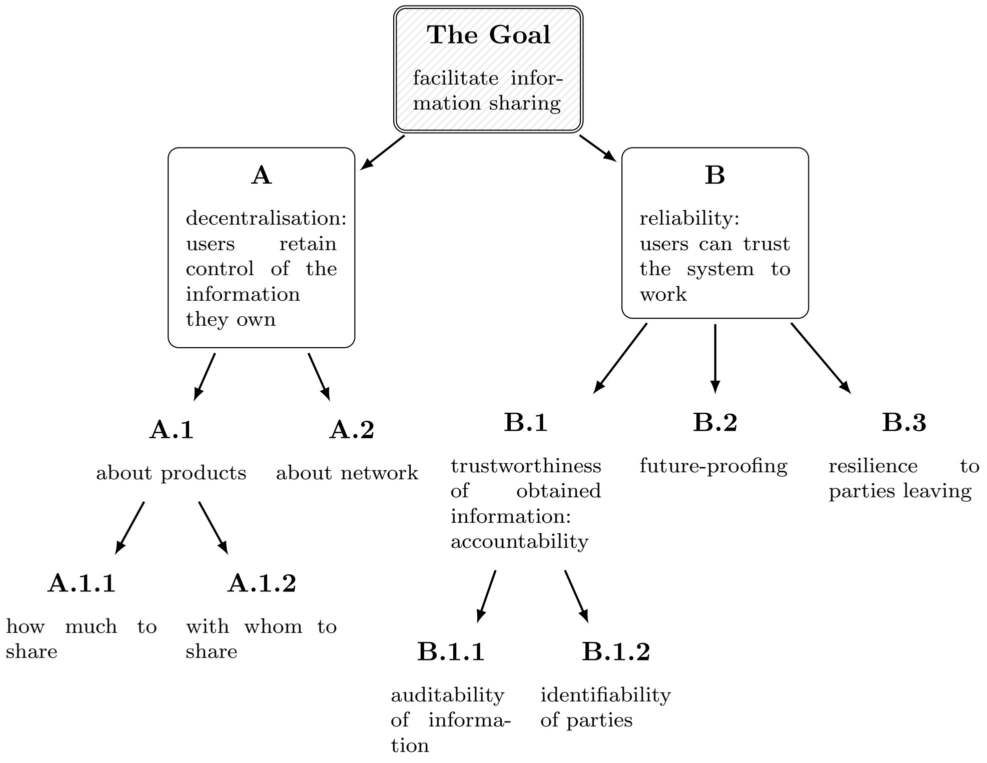

The Goal
facilitate information sharing
A
decentralisation: users retain control of the information they own
B
reliability: users can trust the system to work
A.1
about products
A.2
about network
B.1
trustworthiness of obtained information: accountability
B.2
future-proofing
B.3
resilience to parties leaving
A.1.1
how much to share
A.1.2
with whom to share
B.1.1
auditability of information
B.1.2
identifiability of parties

Figure 2: A breakdown of our core goal into subgoals.

1.1.1 Decentralisation

Firstly, the parties' information in a value chain is **decentralised** and desirably so, which means that the information our final users need is distributed (and often purposefully hidden) therein. **A**

A product generally consists of multiple components. Consider for example the electronics industry: a television contains a circuit board, an LCD panel, a power regulator and many more parts. While the television may be assembled by one company, its parts may be bought from other companies and in turn consists of more parts. Each of the parties in the value chain has certain pieces of knowledge about the finished product, a television, and not any single party has them all. Such information can in fact be a valuable asset – a trade secret **A.1** – and therefore be part of the value of a company. Therefore, if you want to gather a piece of information about a product, the first challenge is to find the party that has it.

Not only information about products is distributed across the value chain, but also the information about who has information about products is. Some companies are very transparent about their suppliers (e.g. fairphone), but there are also companies that want to keep their suppliers a secret in fear of being left out. Some parties are simply intermediaries. Parties like these are susceptible to **A.2** losing their value once a supplier gets in direct contact with a recipient. Knowing a supplier can be worth money by itself and most people are not willing to share this kind of knowledge.

Data therefore has to be handled in a secure, decentralised way. So when someone in a party's network is looking for information, and while they don't have the answer, they know someone else in their network who do, they might not be willing connect the two parties directly in order to exchange information. They may, however, be willing to act as proxies if it is guaranteed that the two communicating parties will not be able to exchange identifying information.

A party should not obtain knowledge about business partners through requesting information. Likewise, a party should not obtain this knowledge either when providing information.

Furthermore, companies want to stay in control of their own data and do not typically want to rely on anyone else to handle it for them.

Therefore, to ensure that information is exchanged, we cannot simply gather all information in one place and manage it centrally or trust someone else to do so. We need to keep the information in the hands of their owners, and create a game which they all have an interest to play, and design it in such a way that the information is allowed to flow where it needs to.

To make this possible, while ensuring that as much information as possible can circulate without stripping any involved party of their trade secrets, we need to enable parties to fine-tune how much information is exchanged and who it is accessible by.

Some information is confidential. For example, the precise amount of mercury that goes into party A's batteries may be a trade secret.

However, party A may be willing to disclose partial information, or may be willing to disclose full information but only to certain other parties. **A.1.1**

Party A may be willing to answer questions such as: "Does the product contain mercury?", or "Does the product contain less than five grams of mercury?" [2] Therefore, we need to enable them to do so.

Also, A may only be willing to share their information with certain types of other parties only. For example, they may be willing to share the make-up of **A.1.2** their product *with recyclers only.* To accomplish this, a party should be able to specify which groups are authorised to access their information. As mentioned before, it is in a party's best interest to keep their business partners secret. **A.2**

1.1.2 Reliability

Secondly, the Circularise system has to be **reliable** to be up to the challenge of facilitating the transition to a circular economy. We build Circularise to address a real-world issue, and to be used by real parties in real situations. It follows that the information exchange the system provides has to be reliable. **B**

Reliability here has two meanings: firstly, the information itself needs to be reliable, as in trustworthy, and while the parties who exchange it may wish to remain anonymous, there needs to be a way to hold them accountable for the information they provide. Secondly, the information exchange protocols we provide need to be resilient to parties leaving the network, and scalable enough to handle complex scenarios of value chains with many mediators.

Firstly, for Circularise to be useful, information obtained through it has to be trustworthy. Companies might be willing to lie to make their product worth **B.1** more if they can get away with it (e.g. about whether it contains materials that are harmful for the environment). It is therefore necessary to incentivise parties to only share correct information, and to hold them accountable when they don't.

Most of the trustworthiness of the data in Circularise is generated by the work of auditors. The better they can do their job, the more trustworthy the **B.1.1** system becomes. It is therefore in the interest of Circularise to help facilitate the auditing process, by holding parties accountable for the information they exchange. However Circularise cannot provide any form of special privilege to the auditors (or any category of users) by design. Therefore, auditing will always require the cooperation of the audited parties, and go through the traditional out-of-Circularise channels. What we can ensure, on the other hand, is that the log of all information transfers are kept forever, so that parties cannot at a later time deny that information was exchanged, and what information was exchanged, when requested to reveal that data during an audit.

Another factor in keeping parties accountable for the role they play in Circularise is to ensure that any on-Circularise action can always be traced back to its real-world actor. This means that parties' digital identities need to be connected **B.1.2** to their real identities, and that connection needs to be public knowledge.

[2]Not all "Is there less than X mercury?" questions have to be answered. Otherwise one might ask this question repeatedly with decreasing-increasing amounts and thus pinpoint the exact amount. To figure this out is always up to the answerer.

The second core aspect of reliability is the guarantee that the system addresses real-world challenges. The main challenge that we face in that respect is that it is not always known in advance what information will be needed in the future to close a circular loop. It is sometimes known which information one **B.2** needs to attach to a product, during production, which would make recycling easier. The problem is that the life span of many products can range from several years to decades, even more in a circular economy. It is impossible to have complete knowledge today on what will be important to know in such a distant future. An example is asbestos, a resource which was used for years before it became known that it was a health hazard. The system has therefore to be future-proof and provide techniques for addressing this sort of challenges.

That Circularise needs to be reliable, finally, means that it needs to be resilient to changes in the network. Since a party is in control of both a piece of **B.3** the network information and some information on a product, this information could be lost when they leave Circularise or when the company goes bankrupt, merges, or undergoes other forms of change which are common in the trade. It cannot be expected that a bankrupt party will still be receptive to incoming queries and contribute actively to Circularise. Therefore the availabilty of the information present in the system should not have a critical dependency on any parties, since anyone may stop using Circularise, for whatever reason, at any time.

1.2 Solution outline

Now that the problem is clear, we can outline the solution.

The Goal (Information needs to be exchanged)

At its core, our information exchange platform consists of a question-answering protocol. Products are tagged with CIR**LABEL**s that point to the product's manifest: metadata stored online. As products are assembled by parties involved in the value chain into progressively more final products, at each step the party who manipulates the product will update the manifest to make the supply chain traceable. The question-answering protocol consists of a way to use the label of a product to access the metadata it points to, and then use it to track down parties in the value chain who may know the answer to some question.

A (The information has to stay decentralised:)

To keep the information decentralised, we ensure two things:

A.1 (Parties have to remain in control of their products' information:) First, that parties can choose what to share, and with whom to share, about their product information. Therefore, all parties can determine for themselves how much they can share without losing valuable information. In particular:

A.1.1 (Enable to fine-tuning how much information is shared) We implement for this purpose two question-answering mechanisms: *push and pull questions*. Using push questions, parties can permanently attach the answers to some predetermined questions to the product's manifest. Parties in need for information can simply access it from the product's manifest. Using pull questions, parties in need for information can find their way to who has it, and they in turn can decide on each information request what exactly to disclose. To further limit the amount of information that is disclosed with each exchange, we implement zero-knowledge information transfer protocols, thanks to which it is possible to only transfer exactly the information you intend to transfer with the mathematical guarantee that no other information is leaked in the process. Finally, we allow parties to disclose information in a nonspecific format: ranges instead of precise values. In this way, parties are allowed to fine-tune the amount of accuracy of the information they provide.

A.1.2 (Enable fine-tuning with whom information is shared) In the case of push questions, we allow parties to encrypt the answers in such a way that only parties belonging to some groups of parties can read them. This also needs to be determined in advance. In the case of pull questions, parties have full control over the information and can restrict access as much as they want to.

A.2 (Parties have to remain in control of their network information) Product CIR**LABEL**s need to contain some information about their creator's network, that is, their position in the value chain. In this way they enable parties in need for answers to find parties who have them. Network information includes who supplied the parts of a product and who the finished product was sold to.
To protect the privacy of all parties, the information is obfuscated by hiding each identity behind a secret name known only to the label's owner.

B (The system has to be reliable:) This means that:

B.1 (Information should be trustworthy) When a party obtains an answer to a question, we offer some guarantee that the information is trustworthy. We do this in two ways:

B.1.1 (Information owners should be held accountable for the information they exchange) Any information that is shared in Circularise will be stored on-chain, forever immutable and traceable. Parties will be able to prove that the information that was shared is undeniably linked to the data that was stored. Therefore the shared information can be audited, with the cooperation of involved parties.

B.1.2 **(Information owners need to be identifiable)** Information leaves a paper trail that can be followed. Even though the providers of this information are generally anonymous, with the cooperation of intermediate parties the actual identities can be uncovered.

B.2 **(Circularise has to be future-proof)** The main way by which we address this is the pull question mechanics: pull questions allow parties in need for information to trace back parties able to provide it, even long after the product was created. Also for this purpose, we build on open source technology and existing standards to allow circularise to embrace new future developments.

B.3 **(Network should be robust to parties leaving)** We achieve this by allowing questions to be asked directly to any party in the (anonymised) value chain. This makes the system resilient to gaps in the network caused by intermediate parties that have left. So long as the party that owns the information is responsive, it can be contacted by the party in need of its information.

Any party involved in the value chain of a product can interface with the product's manifest, through its label, if it already has one, or create one for it. In labelling a product and creating a manifest, a party becomes a participant of the Circularise system. Joining Circularise is thus simply a matter of interfacing with an existing label, or purchasing one, putting it on a product and then attach some information to it.

In the rest of this white paper we explain how precisely we build Circularise's model to achieve the goals stated above. In Section 2 the technologies are presented that form the base for Circularise along with a number of technologies that were considered, but eventually discarded. Section 3 provides the core concepts of the system as an abstract overview. This is followed by a number of definitions and terminology in Section 4. In Section 5 a number of protocols are explained in depth, which are used in Section 6 to verify whether the proposed solution meets the goals.

2 Related work

Here we share a high-level overview of some relevant existing technologies that have the potential to help us achieve some of Circularise's goals. We argue whether and how these technologies are useful for us.

2.1 ZkSNARKs

Verification with no knowledge was accomplished by using zkSNARKs, a protocol first introduced by Bitansky et al. (2012). ZkSNARKs help provide convincing evidence that a given statement is true, without giving away any information beyond this statement. It is theoretically possible but computationally infeasible to falsify one such proof.

Circularise has requirements where it is essential that only the truth of a statement is proven, to avoid other information leaking. Namely it is needed to prove that an inquirer is a member of a group without allowing the inquirer to be identified.

Bitansky et al. have shown that zkSNARKs exist for at least all NP-problems. ZkSNARKs are applicable if you can transform the statement you want to prove into the problem of knowing the answer to a specific NP-Complete problem (there currently exist zkSNARKs for the quadratic assignment problem and quadratic span programs). It is therefore not straightforward to use zkSNARKs in our case, for we'd first need to translate it into an equivalent NP-complete problem. So the statements that are to be proven need custom-made transformations to an NP-Complete problem. However ring signatures, described in Section 2.4, are made specifically for this kind of problem and therefore are a better choice.

A.1 is another case where zkSNARKs are possibly useful to make verifiable claims with hidden information. However, once again we have found a more specific solution for this problem (see Section 2.3).

ZkSNARKs are useful in cases where information needs to be hidden. However, while offering a more general approach, they require too much tailoring compared to techniques that are made for a specific application. In conclusion, we do not use zkSNARKs because our problems can be solved by other more specific, off-the-shelf techniques.

2.2 IPFS

The InterPlanetary File System (IPFS) is a peer-to-peer file sharing system, inspired to BitTorrent. The system provides Content Addressable Storage (CAS), meaning files and their contents are identified by their hash, and can be addressed as such. In the context of IPFS, files are referred to as *objects*. An object can contain content and any number of links to other objects, which are also simply the hashes of those objects. Under the assumption that you use a secure enough hashing function, the data structure makes it impossible to change the contents of an object or the objects under it without changing its hash.[3]

In the context of Circularise, using IPFS is desirable if we can use it to store our manifests. A consideration is the difficulty of using encrypted content and links on the platform. IPFS has built-in support for object-level cryptography, as well as having encrypted links to objects available.

While IPFS offers immutability of content[4], availability is not guaranteed. An object is only available for retrieval if at least one connected peer in the

[3]IPFS currently uses SHA-256 hashes, which are, at the time of this writing, understood to be secure.

[4]Under the assumption that SHA-256 is secure

IPFS network has a full copy of it. If no peer has a copy of the file anymore, it is simply lost. For normal information, a request can usually be made to the owner of the information to share it, but this does not work for orphaned information. This lack of availability poses an issue for orphaned information, especially so if there are non-technical reasons that this information needs to stay available (e.g. legal).

IPFS has many desirable properties, but the guarantee of data permanence is a serious need for Circularise. Besides, IPFS is still under development and it may be at too early a stage to reliably host a production service such as ours.

If there really turns out to be a need to use it, e.g. because blockchain storage turns out to be too costly, we should think of a solution that uses IPFS for bulk storage but keeps essential structural data on the blockchain.

2.3 Zero-Knowledge proofs of ranges

Zero-Knowledge proofs of ranges can be used to prove that x lies in a given range without revealing the value of x. These proofs work with commitments. Anyone can commit to a value x_1 by creating a commitment $E(x_1, r_1)$, a value which can be computed by knowing (x_1, r_1) but for which it is computationally infeasible to find any other pair (x_2, r_2) such that $E(x_1, r_1) = E(x_2, r_2)$. Meaning one cannot feasibly switch the value of x_1 during their proof for that commitment.

Commitments can be used to provide a convincing proof that $x \in [a, b]$ without revealing the value of x, this is called a Zero-Knowledge Proof (ZKP). Furthermore, since the proof is attached to the commitment, one cannot lie about the answer without being caught. To implement these Zero-Knowledge proofs of ranges we adapt the methods introduced by (Chan et al., 1998) and refined by (Boudot, 2000). A summary of those methods is included in Appendix A.

For Circularise, Zero-Knowledge proofs of ranges are useful as they allow parties to flexibly give verifiable answers about their sensitive information without revealing the exact content of the information. **A.1** states that parties should be able to verifiably answer questions like "Does the product contain at most 5 grams of cadmium?" without revealing the exact amount of cadmium in the product. This question can be re-written to a range question by changing "at most 5" to $x \in [0, 5]$, where x is the amount of cadmium in the product (in grams). In fact, any question about the amount of x can be written as a range question. A question on whether a product contains x can refer to the range $[0, m]$, where m is the total mass of the product (or perhaps an even larger amount).

In addition, commitments also help fulfil **B.1**, make information trustworthy. Audits are only needed on the commitment of x instead of every answer given about x. This decreases the amount of audits needed to generate the same amount of confidence in the system.

2.4 Ring Signatures

One of the important requirements of Circularise is **A.2**, the value chain has to remain a secret. To guarantee this secret, parties need to remain anonymous when answering questions and when asking them. Anonymous answering is easier to achieve than anonymously asking a question, since it does not really matter who gives the answer as long as it is auditable (auditors are an exception when it comes to learning secrets). In contrast, an inquirer needs to be authorised by the respondent before an answer is given. The challenge here is that an inquirer should provide enough information to be authorised while at the same time remaining anonymous.

Ring signatures were first introduced as a way to leak authoritative secrets in an anonymous way by Rivest et al. (2001). In a more general sense, ring signatures can be used to provide a means of giving proof that a message m was signed by a member of a group without a way to identify which member it was. An alternative for this property exists in the form of group signatures which were proposed by Chaum and Van Heyst (1991), however ring signatures have some advantages over group signatures.

One of these advantages is that there is no notion of prearranged groups and that there is no need to set anything up. Furthermore, it requires no cooperation or permission of other members of the group that are included in the signature. This is useful for Circularise since parties can enter (or leave) the network over time and there is no need for members of a group to permit a new member to their signature group.

Another important advantage of ring signatures is that it does not include a means of revoking anonymity (unlike group signatures). Thus there is no one who needs to be trusted to keep the anonymity of inquirers. Inquirers do not need to be audited so there is no need for revoking anonymity. The ring signature protocol is explained in detail in Appendix B.

2.5 Diffie-Hellman Key Exchange

As there are some answers which should not be heard by intermediate parties, a secure channel needs to be established between the inquirer and respondent. If the answers would consist of a single response, providing a single public key for the inquirer might have sufficed, but for Circularise this is not always the case. An answer in the form of a ZKP usually requires a challenge and response kind of communication.

To set up an encrypted channel between an inquirer and respondent with only secret keys, one can use a Diffie-Hellman key exchange (Diffie and Hellman, 1976). This is an exchange where two users construct a secret key, without ever actually posting it on the potentially insecure channel. This provides forward secrecy in that even when the complete exchange is recorded, the created key can never be obtained by a simple eavesdropper. The full Diffie-Hellman key exchange is provided in Appendix C.

The constructed secret can have multiple purposes, but for Circularise its purpose is to act as the key for symmetric encryption between two parties. Do note that a Diffie-Hellman key exchange is susceptible to man-in-the-middle attacks. Therefore it is important that both Alice and Bob are able to verify each other's identity (which is also where the ring signatures are useful).

2.6 Blockchain

It has been mentioned before that centralisation of data is unwanted by companies. Circularise is therefore to be a peer-to-peer system. Given the sensitive nature of the information, furthermore, the data distributed to the peers has to be suitably protected by encryption. On top of that, parties need to be identifiable for them to be held liable (i.e. from their digital identity it must be possible to backtrack their real-world one).

For these reasons, Circularise uses a blockchain. This section describes the specifications of a blockchain that are required for the implementation of Circularise and provides motivation for our choice.

2.6.1 Immutable storage

Data is always under control of its owner, so Circularise needs some guarantee that the owner can be trusted. Audits give some form of trustworthiness of the data, but auditors need to know what should be audited. It has already been mentioned in Section 2.3 that commitments are a great way to provide an immutable promise of a value, but this commitment still needs to be stored somewhere where the committer cannot change its value unnoticed. This means we want the commitments to be immutable. Parties need to be allowed to change their commitment (in case they made a mistake), but when they do this it should be known it changed, when such a change happened and what the previous commitment was. This kind of immutability is made possible by blockchain technology.

Commitments need to be undeniable, not only in their value but also in their origin. The value of a commitment may be faulty or not, but in order to do a proper audit, the auditor needs to know who created the commitment. The creator can be linked to a commitment by providing a digital signature, but this only means that a commitment can be linked to a public key (and we need to know who the owner of that key is). This provides a second case where immutability is required: the identification of parties. Parties need to be linked to their public keys to keep them accountable for their commitments. In the same way that the value of a commitment should not be changed unnoticeably, the link between public key and party should also share this characteristic.

2.6.2 Read access

In addition to the authentication of parties, it is important that the Circularise network is able to authorise parties for certain tasks (such as managing group membership). Examples of actions that require authorisation include viewing data and adding/updating data. The immutability of the data prevents records from getting lost due to misbehaving parties.

One issue is that providing authorisation for the viewing of data is not doable on a public blockchain. As data is public on the blockchain, it is visible to everyone at all times. This makes it impossible to deny someone access to data that is stored on a blockchain. Therefore, authorisation for the viewing of data should be accomplished through different means, like encrypting the data for the intended recipients.

On the other hand, authorisation for adding and updating data is doable using almost any blockchain technology. In some cases, the public wallet of an updater is sufficient in order to determine their rights. However, in case the group of a public wallet needs to be verified an additional step is required. In such cases, the group of a wallet needs to be obtained first and then used for determining authorisation.

2.6.3 Choice of blockchain

We need a blockchain technology which offers immutable storage, facilitates (classification-based) authorisation and allows the creation of a token. These requirements are not that special, so Circularise is in principle not linked to a specific blockchain technology. However, so far as permissionless blockchains go, the Ethereum blockchain is the ideal candidate.

The Ethereum blockchain provides immutable transactions where its sender is publicly visible. What makes Ethereum a proper fit for our requirements are its smart contracts. Because all operations on smart contracts are done through blockchain transactions, they retain the desired immutability and accountability of a typical blockchain transaction. Smart contracts can also check whether someone is authorised to perform a certain action in a flexible way. If the classifications of all participating parties are stored in a smart contract, other smart contracts can use that contract to look up the classification of anyone who is trying to perform an operation.

Ethereum is a widely adopted blockchain with many users, which is essential for a distributed technology. It also has an active development team and community. In conclusion, because Ethereum offers decent support for our requirements, and because of its popularity and good developer support, this blockchain technology is used to implement Circularise.

2.6.4 Limitations

Ethereum has known limitations. One of these is the transaction capacity (low amount of transactions per second) and another is the transaction speed (it takes a long time to fulfil a transaction). Given that we are dealing with the production of products and that each product will need some information stored on the blockchain, a fully operational Circularise may require millions of transactions on your average Monday. The current state of Ethereum certainly does not support this scale of transactions.

Furthermore, the transaction fee of Ethereum can become quite high (there was a peak at januari 2018 of $ 4.15 per transaction), and this may be a problem for companies in need of high transaction volumes.

Ethereum, however, is working on improving these limitations. So it is a question of whether Circularise will grow faster than Ethereum can fix their problems. Given our requirements, many blockchain services could host the Circularise platform. For this reason we keep our options open and do not commit to a specific blockchain.

2.6.5 Token model

No matter what blockchain service we choose to implement CIR**BASE** on, the service is going to have an upkeep. To make circularise a self-sustaining system, we build into it a payment system based on its own cryptocurrency, CIR**COIN**.

Fiat currencies are going to be exchanged for CIR**COIN**S, and CIR**COIN**S will be spent to fuel blockchain operations. Depending on which blockchain Circularise is implemented, the specifics of the token model may vary[5].

The fundamental idea is that the upkeep will have to be supported by the users. In the case of Ethereum, transaction costs have to be paid by parties every time on-chain operations (not read-only) are executed. For example, when a party creates a CIR**LABEL** or modifies a manifest. So, some of the actions that parties can perform on Circularise are going to have a (small) cost to be paid in CIR**COIN**.

3 Our proposal: CIRBASE

Here we provide a concrete description of a platform on which to build all of Circularise, referred to as CIR**BASE**. In addition, a description for systems that allow all participants to interact with CIR**BASE** is presented as well.

3.1 Circularise

Every party having something to do with a product can create a manifest for it, which become nodes in a network that overlaps the value chain the product is involved in. Every manifest's owner corresponds in turn to some party that had something to do with the product or component and that can, in principle, provide or request information about it. The network acts as a peer-to-peer system where every participant provides information about a product while maintaining ownership of their data. They decide what information to share and with whom they share it.

Each product/component/resource has a CIR**LABEL** attached which links to a manifest that is stored on the blockchain. This manifest contains information that is used to answer questions and also link to other manifests. In order to keep secret which parties are part of the value chain, the identities of the manifest

[5]Token model will be finalised in V3 of the paper

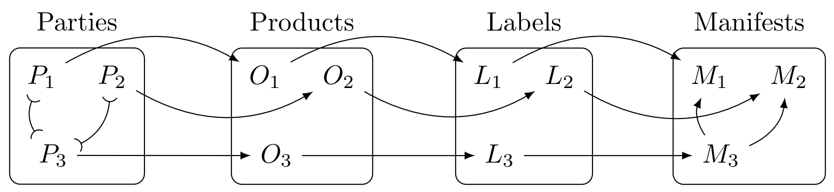


Figure 3: A depiction of the core Cirbase elements. $)\!\!-\!\!($ arrows represent "real-world knowledge". E.g. P_1 and P_3 interact in real life: they know one another. $\rightarrow$ arrows denote various forms of information access: users can access the products they manufacture; the products contain their labels; the labels point to their manifests; manifests point to other manifests.

creators are anonymised. By following the links, called network links, from a single manifest one can obtain more manifests and follow their links in turn, gradually building an anonymous representation of the value chain. However, all one obtains by doing so is the structure of the network and the names of the groups of parties involved, not the identities of the individual nodes or their information.

For example, one may find out that 3 manufacturers, 2 repairers, 1 retailer and 2 brokers were involved in the value chain, and which was dealing with which, but not their real identities.

Within this network, only the direct contacts (see the thin arrows between parties in Fig. 3) will know the actual identities of a manifest's creator: just like tier 1 suppliers and recipients know each other's identities in the real world. Keeping the knowledge as confined as possible ensures that the information about the network remains secret, while still allowing for a participant to be identified by leaving a digital trail that can be followed with the cooperation of the manifest's creators.

Besides product level manifests, CIR**BASE** also allows the creation of manifests for a more generic level of products such as product lines or brands. This avoids the need to modify each individual product's manifest, when the information to be updated is actually about a whole class thereof. These higher level manifests are linked to via so-called product links and will not contain network links themselves.

It needs not be that everyone in the value chain is part of Circularise, though this would obviously mean more and better information. If a party involved in the value chain chooses not to label its product, but its suppliers do, if its recipients receive the labels they can choose to scan them and link them to their own, thereby creating a link in Circularise that jumps one step in the actual value chain. This way, manifests can be linked even though their creators are not directly connected themselves. This is however a risk for parties that do not want their suppliers and recipients to get in touch. These are therefore incentivised to join Circularise as well.

Obtaining information from manifests happens through questions and answers. In Circularise there is an important distinction between two types of questions: those that a participant has answered beforehand (push-questions) and those that can be asked at a later time (pull-questions). Push-questions's intended usage is mainly to address questions that are mandatory by law, but can also consist of custom questions that are defined by other participants. They can be used when it is important that the answers are available at any future time, even if the participant that supplied them has left the network. Pull-questions are appropriate when participants want to obtain extra information or when the answers are more confidential. These kind of questions are somewhat less important in terms of "answer uptime" but more important in terms of maintaining ownership of data. The two kinds of questions will therefore be facilitated in different ways, though both will involve adding information to the manifest.

For push-questions answers are put directly in the manifest and subsequently propagated to recipients as the label moves along the value chain. This allows anyone with access to the manifest and the right authorisation (i.e. belonging to the right groups) to access the answers.

Pull-questions can be asked to anyone in the value chain. The network formed by the manifests can be used to find the right anonymised party to ask the question to. If that party knows the answer and accepts the inquirer's authorisation, they can respond through their own secure channel.

Pull-questions can be turned into push-questions when a participant deems it necessary by updating the manifest. This can be useful when for instance, a question becomes mandatory in the future, when it was not in the past. The other way around, push-question to pull-question, is also possible, though effectively the answer would still be on the blockchain (due to the immutable history).

3.2 CirBase protocols: what can participants do?

A participant is any party who has a Circularise wallet. A participant could be e.g. an individual, a company, a government, a corporation of companies, a group of governments, an NGO, etc.

CirBase is a platform that consists of a number of smart contracts that implement several information exchange protocols that we define. Using these protocols, participants can exchange information in a way that meets the goals stated in Section 1.1.

The CirBase Protocols:

Protocol 1 **(Publish a group.)** Each participant decides who they trust with their information. For example, a manufacturer may entrust only certified recyclers with the answers to certain sensitive questions. To implement this feature, Circularise uses the concept of groups. Any participant can create and manage groups of participants that they give a label, e.g., "trusted recyclers". When a participant shares a piece of information, they can choose which groups are authorised to view it.

Each participant is supposed to state which groups it belongs to, and each group does the same. This acts as a kind of handshake. Only when a participant is in a groups's list *and vice versa*, Circularise will consider the participant to be a member of the group.

The group's creator is responsible for who they choose to include in their group. Whether participants choose to authorise the group to include it among its members will depend on its trust in the group's creator or its current members. This makes it likely, that the most popular creators will be (supra/inter)national institutions which are bound by strict and verifiable rules in the real world. The creator can delegate group management by adding pointers to other groups to their group, which effectively adds the members of those groups to their group.

Note that any participant can own multiple groups, and each participant can belong to multiple groups. For example, participant p could be in p's "recyclers" group, but in p''s "retailers" group. participants who know p in the real world and know that p is in fact a retailer will draw their own conclusions about who to trust.

Interactions regarding the creation, modification and deletion of groups all happen on the blockchain via smart contracts, and as such are transparent to anyone with access to the public ledger of transactions.

Every group has an asymmetric key pair for which answers can be encrypted, so that only members of the group have access to that answer.

Protocol 2 **(Add/Remove participant to group)** Participants can be added to existing groups and removed from them. These operations are fairly straightforward, but they do require extra management of the group key.

Extra management of the group key pair is required because it is unwanted to let participants, that leave a group G, access any new information encrypted for G.

Protocol 3 **(Manage lists of questions)** Circularise supports the formulation of question lists. Any participant can create a question list. For example, a company like KPMG can choose to create their own question list and participants can choose to subscribe to that list, i.e. answer to every question in the a list, to gain some form of recognition or certification. KPMG will then be the quiz master of this question list. To avoid tampering, the lists are stored on the blockchain.

A question list can only be updated by its quiz master. An update of a question list will create a new version of that list. Participants specify which version of the question list they subscribe to. As the list evolves, the subscription does not update automatically. (In future versions, this will be an optional feature.) This allows users to choose whether they still want to support the latest version, as an update might include new questions that they do not want to answer.

Quiz masters are motivated participants who want to actively contribute to the system. They need not be individuals: like any participant, quiz masters can also be a collective decision-making entity, such as an assembly, consortium or board.

Question lists can also contain questions which participant are legally obliged to answer. There is no mechanism in CIR**BASE** that can enforce that they answer these questions, but the identifiers can be made known by a trusted participant such as Circularise or the government. However, motivation could come from formal or informal recognitions (awards, certificates...).

Protocol 4 **(Subscribe to question lists)**

Participants can subscribe to a question list to show their intention of answering those questions. The list of subscriptions, plus information about which groups are allowed to read the answers is included in the manifest. This information is useful for interested participants to determine whether their questions are going to be answered or not.

A notification will be pushed as an event on a smart contract whenever a question list is updated. It is possible to be subscribed to multiple lists (also multiple versions of the same list).

Protocol 5 **(Publish or modify a manifest)** A participant can publish a new manifest or modify an existing one that they own.

They do so by creating an empty data structure and populating it with information such as general product information, push-questions and their answers, links to other manifests, subscribed question lists and some anonymised ownership information.

The manifest is linked to a CIR**LABEL** and published on the blockchain. Although the manifest can be viewed by anyone, the identity its creator is anonymised and sensitive information like answers to push-questions and product links are encrypted.

Almost all information in a manifest can be modified, but only by its creator. Both the old version and the modification will still be available on the blockchain due to the immutable history, which prevents malicious participants from tampering with their manifests.

Protocol 6 **(Ask a Push-Question)** In this case the participant attempts to read the stored answers of a push-question. The answers can be found using the information in the product's manifest linked to its CIR**LABEL**. So whether a participant can access the answer of a push-question depends on the group(s) that they are a member of. The inquirer may also follow the network links to find, for example, whether the question is answered in the components of the product, or even by someone else in the value chain.

Answers can be either public, or encrypted for specific groups. For example, a customer that bought a product may only be allowed to see general product information, whereas a verified recycler/auditor may also see claims made about the particular contents of the product: the way it was manufactured, the origin of the materials, etc.

Protocol 7 **(Ask or answer a Pull-Question)** In this case the participant becomes an inquirer or respondent. The inquirer uses the information in the manifests, like the network links, to find a manifest whose creator has (part of) the answer. They then use the anonymised identity of the creator to create a secure channel with them and ask the question.

The Circularise system guarantees that neither the inquirer nor the respondent will know exactly which entity they are in contact with (unless they willingly disclose that information).

At the same time, they will have the certainty that the inquirer is of the claimed group and that the respondent is the real owner of a manifest in the relevant value chain. The group of an inquirer can be used by the respondent to determine whether they want to answer the question.

Protocol 8 **(storing and verifying identification)** For **B.1**, the trustworthiness of information, it was stated that participants need to be held accountable for their behaviour. As not all enforcement of rules (e.g. fines) can be handled through Circularise, the real-world identity of participants is an important asset to allow this enforcement. With this protocol, the Circularise identity of a participant is publicly linked to their real-world identity.

Every participant remains, however, the sole owner of its secret (anonymised) identities used for publishing manifests.

4 Terminology and definitions

This section contains a formal definition of the important parts of the Circularise system. Section 4.1 gives a definition for participants and groups. Following that, Section 4.2 describes the format of information relating to push- and pull-questions. Then, in Section 4.3 a full definition of the manifest is given. Finally, registries for managing identity are defined in Section 4.4.

4.1 Participants and Groups

A participant is potentially any subject that has a wallet address in the blockchain platform circularise will be implemented in. So for simplicity's sake, we say that the set of all participants coincides with the set of all wallets.

Definition 1 (Participant) *The set of all wallets is* $\mathbb{P}$. *A participant is then* $p \in \mathbb{P}$

A group is a named list of participants. Crucially, a group is a "for internal use" denomination, and as such we don't envision a unified, fixed list of them.

If a future group creator sees fit to categorise all of the participants she knows as "potatoes" or "squirrels", that's up to them.[6]

Since audits can only occur through external channels, the categorisation of some participant as an auditor (or some other equivalent label) only has the in-system effect of vouching for the trustworthiness of the information that the auditor publishes on Circularise, so long as one trusts the categorisation in the first place. So if a malicious party P starts to label itself or another party Q as an auditor, third parties will only trust that labelling as much as they trust P's judgement, or, if they know Q directly, their own.

Besides a collection of members, groups contain additional information. First of all there is a participant linked to the group that is considered the owner. Additionally a unique, possibly descriptive name is used identify every group.

The final piece of information is a public key, which can be used to encrypt data specifically for the members of a group (the members only having access to the associated private key). New key pairs have to be generated every time the group's composition changes, to prevent leaving/banned members to retain group privileges.

Definition 2 (Group of participant p**)** *A group G, created by a participant p is a (owner, name, members, public key) tuple, where the name is unique, members is a list of participants and pointers to other groups ('embedded groups'), and public key is the public key of the group. The group name string needs to be unique for p.*

A group name n is a string; the set of all names N is a subset of all strings:

$$N \subseteq str$$

We call a $\langle p, n\rangle$ pair a 'pointer' to a group, in the sense that one can iterate through p''s groups until it finds a group named n. A pointer t is defined as:

$$t := \mathbb{P} \times N$$

Let $\mathcal{T}$ be the set of all pointers.

A group is:

$$G := \mathbb{P} \times N \times \mathcal{P}(\mathbb{P} \cup \mathcal{T}) \times \mathbb{N} \times K_{pub}$$

Where K_{pub} is the set of all public keys. Subject to the constraint $\forall G = \langle p, n, P, k_{pub}\rangle, \langle p, n\rangle \notin flatten(P)$. In words: no group contains a pointer to (a group that contains a pointer to(...)) itself.

The set of all groups is $\mathbb{G}$. We use the short-hand notation:

$$\mathbb{G}_p := \{G \in \mathbb{G} \mid G = \langle p, \_, \_, \_\rangle\}$$

The definition of groups is recursive. When a group G_A contains a pointer to another group G_B, the members of G_B are also considered members of G_A.

[6] Presumably, not many will use those lists.

The same holds for when G_B points to a group G_C, then the members of G_C are also considered members of G_A. All of these members (direct and indirect) are given access to the private key that is paired with group G_A's public key. We formally define this relationship with the *members* function:

$$members(\langle p, n, P, \_\rangle) := (P \cap \mathbb{P}) \;\cup\; \{m \mid m \in members(\mathbb{G}_{p'}[n']),\ \langle p', n'\rangle \in P)\}$$

Members of a group can change over time. They can get added, or removed from the group, or any of the groups that it (in)directly points to. The term *generation* is used to refer to the specific composition of a group at a certain point in time. Whenever a member is added or removed, the generation is incremented. In general we always refer to the last generation of a group, though in some cases there is a need to refer to older versions of a group in which case the generation is specifically mentioned. The generation of a group G is denoted as G^i, where $i \in \mathbb{N}$.

4.2 Questions and Answers

The core information that can be exchanged on the circularise system are questions and answers. Questions are predefined and answers relating to their products are linked to the questions by participants.

Questions are freeform strings that have to be manually answered. On the other hand, questions contain metadata concerning their topic (e.g. whether the question is about mercury), this information is used by participants to verify the correctness of the answer (using commitments, as explained later).

Definition 3 *A question Q contains a (human-readable) description and a topic. Similar to group names, the set of all descriptions $\mathbb{D}$ and the set of all topics $\mathbb{T}$ are a subset of all strings:*

$$\mathbb{D} \subseteq str,\ \mathbb{T} \subseteq str$$

A question Q is then a pair:

$$Q = \langle \mathbb{D}, \mathbb{T}\rangle$$

4.2.1 Question lists

Question lists work in much the same way as groups: everyone can publish their own question list and import questions from other lists.

We call $\mathbb{L}_p$ the set of question lists published by participant p. For an $L_p \in \mathbb{L}_p$, $L_p[i]$ is the ith question of that list. $L_p[i]$ either contains a string (description of the question) or a pointer to another list's question $L_{p'}[j]$, where $p' \in \mathbb{P}$ and $L_{p'} \in \mathbb{L}_{p'}$. Note that the pointed-to question can be from a different list of p or from the list of a different participant. Finally, we define the set of all question lists $\mathbb{L}$ as follows:

Definition 4 *The set of all question lists is* $\mathbb{L} = \bigcup_{p \in \mathbb{P}} \mathbb{L}_p$

4.2.2 Question/answer pairs

The combination of the questions and answers are called question/answer pairs (QA pairs). QA pairs are the pieces of data that are used when it concerns push-questions. As conforms to the name of push-questions, their data is stored in manifests and forever pushed along in the value chain.

The answers of the QA pairs can be encrypted or not. If they are encrypted, a symmetric encryption scheme is used. The symmetric key is then encrypted for the authorised groups and passed along.

Without loss of generality we assume that a QA pair is a pointer-string pair (the pointer being a reference to the question in a question list). When asking a push-question, you are in fact just looking up a piece of data that is linked to the question. Thus the answer is already there, all you need is the proper authorisation to see it.

Definition 5 (QA pairs) *Question/answer pair are encoded as:*

$$QA = \begin{cases} \langle \{L_1[i_1], \ldots, L_n[i_n]\}, S \rangle & \textit{if public,} \\ \langle \{L_1[i_1], \ldots, L_n[i_n]\}, E_k(S), \{G_1, \ldots, G_m\}, & \multirow{2}{*}{\textit{otherwise}} \\ \qquad \{E_{G_1}(k), \ldots, E_{G_m}(k)\} \rangle & \end{cases}$$

Where $\{L_1[i_1], \ldots, L_n[i_n]\}$ are all the questions that S is an answer to. The questions may be synonyms, or simply different questions answered by the same answer. $\{G_1, \ldots, G_m\}$ is the set of all groups that are allowed to read S. $E_k(S)$ is S encrypted with a symmetric key k. The authorisation of groups is realised by $\{E_{G_1}(S), \ldots, E_{G_m}(k)\}$ which are copies of k encrypted for each of these groups.

4.2.3 Topic/commitment pairs

Pull-questions are usually questions that have been defined after the product was created (it is infeasible to update all the linked manifests for a question). So even though participants will generally have a locally stored QA pair, this data will not be stored in the manifest. To keep the answers to these types of questions verifiable, additional information is added to the manifest in the form of topic/commitment pairs (TC pairs).

TC pairs contain information that is more generic than those of QA pairs, which have been made for specific questions. Instead of specific questions, they focus on storing verifiable information on topics. Currently the only possible commitments are those of numeric values. The commitment on the quantity of an ingredient can be used to answer (in a verifiable way) the exact amount, a threshold or a range of the quantity of an ingredient. So even though it is unknown which questions will be asked in the future, there exists a subset of answers which can be verified with commitments that have been provided in the past. Answering pull-questions using existing commitments, while not required, will make the information more trustworthy.

Definition 6 (TC pairs) *Topic/commitment pairs are encoded as:*

$$TC = \langle T, C \rangle$$

Where $T \subseteq \mathbb{T}$ is the set of synonym topics that C is a commitment for.

4.3 Manifests

A manifest is a bundle of information linked to a CIR**LABEL**, and published on the blockchain. They are used by both inquirers and respondents to acquire information for asking and answering questions. Additionally, the manifest can also be used for verifying answers.

The anonymous identity of the creator is recorded on the manifest which allows creators to claim ownership of the manifest when needed. Because a new anonymous identity is used for each manifest, it cannot be directly linked to the respondent's public identity without additional information. The real identity of the creator is also stored in the manifest, encrypted with the holder public key. This way, only a holder (typically a tier 1 recipient) can reveal the creator's public identity.

A manifest can also have any number of holders, a holder being any party who has or has had physical access to the CIR**LABEL**, in which a private key is stored. Anyone with access to the label can use this private key and therefore become identified as the holder of the label. Parties are required to store the private holder key when they see it, so that they can continue to act as holder after they no longer have possession of the label. This is especially important for audits as this is the only way to identify manifest creators.

The information a manifest contains can have different security levels. Some information can be public knowledge, while other information might be intended for more specific recipients and therefore be encrypted for specific groups.

Manifests also contain TC pairs for answering pull-questions. These commitments are made for the bill of materials. If the list of materials is confidential in addition to the amounts, the information can be obfuscated by adding to the manifest spurious TC pairs about materials that are not in the product – commitments for the value 0. The commitments allow auditors to verify the veracity of claims made by the participant.

Network links are used to point to the child and parent manifests. The parent manifests are the manifests of the components and raw materials that were involved in the creation of the product. The child manifests are the manifests of any product(s) of which this product is a material or component. In a manufacturing context, there is usually only one child. For example, the network links of a motherboard's manifest may point to the manifests of all the chips present on the board as well as the manifest of the laptop that is manufactured using the motherboard.

For some parties, like production companies, many of the created products will have very similar information in their manifests. For example, all products from a product line might have the same material composition and user manual. To avoid unnecessarily duplicating information in this manner, we allow a manifest to include references to virtual manifests that contain information for the whole product line, while that manifest in turn could point to a manifest containing the warranties that are shared by all electronic products of that company. These links are called product links.

In general, product links are encrypted to preserve the anonymity of the manifest creator. If a product line manifest were publicly linked to all products of that line, one could expose the owner of all of those manifests by identifying the owner of one them. Furthermore the higher level manifests are more likely to contain company-identifying information, such as a user manual or a product warranty. This information could be used to link companies to (product) manifests and expose the classified structure of the value chain.

Summarising, the entries of a manifest can be:

1. push QA pairs
2. a bill of materials in combination with commitments (is added as TC pairs)
3. a pointer to a collection of question list
4. pointers to product line manifest, model manifest, etc: product links, encrypted for the manifest holder
5. pointers to the manifests of parent and child products: network links
6. pointers to the manifests of the parts of the product - and the other products that the manifest's product is a part of[7]
7. the manifest holder
8. the manifest creator (anonymous address)
9. the manifest creator's real (corporate) identity, encrypted with the holder key
10. any other metadata, e.g. a serial number (which should not give away the identification of the creator)

Definition 7 *(Manifest M_c of a* CIR**LABEL** *c) A manifest of a* CIR**LABEL** *c is a list of key-value pairs, i.e. a dictionary.*

$$M_c \in \mathcal{P}(Stmt_{M_c})$$

[7]the parent-child product relationship means that a product was "made of" another product, such as a batch of butter is made from a batch of milk. The part-whole relationship means that a product maintains its integrity and could in principle be recovered, or exchanged for an equivalent part, such as when a hard drive is part of a laptop. This also means that the parent-child relationship is permanent, the part-whole relationship can be updated over time, e.g. when the hard drive is replaced.

Valid key-value pairs are statements of the following types:

$$\begin{array}{lll} Stmt_{M_c} ::= & \text{'push qa'} \mapsto \mathcal{P}(PushQA) & \mid \\ & \text{'pull tc'} \mapsto \mathcal{P}(PullTC) & \mid \\ & \text{'question list'} \mapsto \mathbb{L} & \mid \\ & \text{'network links'} \mapsto \mathcal{P}(M_c) & \mid \\ & \text{'product links'} \mapsto \mathcal{P}(E_h(M_c)) & \mid \\ & \text{'holder'} \mapsto \mathbb{P} & \mid \\ & \text{'creator'} \mapsto \mathbb{P} & \mid \\ & \text{'reveal id'} \mapsto E_h(p_{public}) & \mid \\ & \text{'metadata'} \mapsto string & \end{array}$$

where E_h is a function that encrypts its argument for the holder.

4.4 Circularise Identities

To make the system trustworthy, participants's identities on the blockchain network need to be connected to their real-world identities. On the other hand, to make the communication protocol private, each participant also needs to have secret identities which can be used to perform actions anonymously on the blockchain.

4.4.1 Public identities

For the former purpose we introduce registries, a bijection between real-world identities and Circularise participants.

Definition 8 (Public Registry) *An identity i is a string, starting with an alphanumeric character; the set of all identities I is a strict subset of all strings:*

$$I \subset str$$

A registry is then encoded as:

$$R = I_c \longleftrightarrow \mathbb{P}$$

where $I_c = \{i_0, i_1, \ldots, i_n\}$ are all identities that have been claimed. We introduce the notations $R_{I_c \mapsto \mathbb{P}}$ and $R_{\mathbb{P} \mapsto I_c}$ for accessing the bidirectional mapping.

Registries are publicly, immutably stored on the blockchain for all to see. The validity of such identities is peer reviewed. The parties that do business with a participant are in the perfect position to check whether the identity that is linked to the manifest of a product corresponds to that of the one they are buying the product of.

4.4.2 Secret identities

For the latter purpose, each participant has to choose a secret identity every time they create a manifest.[8] Secret identities are used to tag the manifests created by the participant. Each participant will then be in possession of their own Private Registry, which is nothing but a list of all of the secret identities hey have created in the past to sign the manifests they have published.

Definition 9 (Private Registries) *A private registry of a participant p is encoded as:*

$$R_p = I_{c,p}$$

where $I_{c,p}$ is the set of all identities that have been claimed by p.[9]

5 Cirbase Protocols implementation

We have explained all concepts and the relationships among them. Next we make this more precise and explain how the implementation works.

5.1 Implementation of Protocol 1: Publishing a group.

Publishing (and creating) a group consists out of a number of steps. Creating the group itself with an initial set of members is the first. Then, an asymmetric key pair is generated of which the public key is published along with the other meta-data of the group (owner, members and name). The private key, however, is not immediately distributed among the prospective participants. Before they obtain the private key, they need to confirm that they want to be part of the group.

Algorithm 1: publishing a group

```
input: group creator $p$, name $n$, participants $P$, groups $G$
    generate key pair $(p_0, s_0)$ // public and secret key
    let $G^1 := (p, n, P \cup G, p_0)$

    distributeKey $(G^1, members(G^1), s_0)$

    publish group $G^1$
end

fun distributeKey
input group $G$, participants $P$, secret key $s_0$
    foreach $p \in P$
        let $E_p$ := encryption function using $p$'s public key
        send $E_p(\langle G, s_0 \rangle)$ to $p$
```

[8]Or every product line; or every so many manifests they create. The more often they change the secret identity, the more protected their identity is.

[9]Assuming the secret identities are created with a strong random generator, the probability of a duplicate occurrence can be made small enough.

```
    end
end
```

When someone leaves Network information can be shared between peers, though initially encrypted. The key to unlocking this information is held by the provider of the network information (though they do not actually need it as they have access to the information unencrypted).

5.2 Implementation of Protocol 2: Add/Remove a participant to/from a group

The main challenge of a dynamic group is the management of the group's key pairs. New members should be able to decrypt old information and leaving members should not be able to decrypt new information.

When a participant leaves a group, a new public-private key pair is generated by the introducer. The public key is published and linked to the group, the private key is distributed among the remaining members. Because the leaving party does not obtain the new key pair, they cannot decrypt information which was made after their departure.

Due to the generation of new key pairs, a situation arises where answers that were created at different dates require different keys to access even when they were encrypted for the same group. If a participant, that joins a group, is only given the current key, they will not be able to read the answers of older questions. So instead of just giving the current key, the list of older private keys is sent to joining members along with the current one.

Leavers will still retain access to the answers to older questions (which were shared when they were still authorised to read them). We don't consider this to be a solvable problem, because when a party had access to the information in the past, they could have made a copy anyways. The only guarantee is that they will not be able to access information that is added after their access has been revoked.

Algorithm 2 describes how to add or remove an entity to/from a group. This entity can be either a participant or a (pointer to a) group.

Algorithm 2: add/remove a participant to/from a group

```
input   group $G^r$,
        participant/group $e$
        $a \in \{\text{remove}, \text{add}\}$
    let $\langle p, n, P, k_p^r \rangle$ := $G^r$
    if $a$ = remove
        generate key pair ($k_p^{r+1}, k_s^{r+1}$) // public and secret key
        let $G^{r+1} := \langle p, n, P \setminus \{e\}, k_p^{r+1} \rangle$
        distributeKey ($G^{r+1}, P \setminus \{e\}, k_s^{r+1}$) // from Algorithm 1
    elif $a$ = add
        let $k_p^{r+1} := k_p^r$ , $k_s^{r+1} := k_s^r$
        let $G^{r+1} := \langle p, n, P \cup \{e\}, k_p^{r+1} \rangle$
        for $i \in [1, r+1]$
            distributeKey ($G^i, \{e\}, k_s^i$) // from Algorithm 1
        end
    end

    publish $G^{r+1}$
end
```

5.3 Implementation of Protocol 3: Manage lists of questions

Every participant can publish a question list on the appropriate smart contract. The publisher of a question list can manage it by adding or removing questions, or deleting the list.

To publish a question list, one needs a name for it (and the initial list of questions). All newly created question lists have version number 1. The version number is going to increase as the list is modified.

Algorithm 3: Creating a question list

```
input name $n$, question list $L$,
    publish $(n, L, 1)$
end
```

Updating a question list L involves creating a duplicate of L, appending to it some new questions or removing from it some old ones and publishing the result under the same name, with the version number increased by one. To avoid data duplication, the references of re-used questions are used instead of their actual content.

Algorithm 4: Update a question list

```
input name $n$, removed questions $Q_{rem}$, added questions $Q_{add}$
    let $\langle n, L_{old}, v \rangle$ := latest question list
    let $L_{ref}$ := references to $L_{old} \setminus Q_{rem}$
    let $L_{new}$ := $L_{ref} \cup Q_{add}$

    publish $\langle n, L_{new}, v+1 \rangle$
end
```

5.4 Implementation of Protocol 4: Subscribing to and reviewing a question list

Automating your decision of whether you want to subscribe to a question list or not could be done in one of two ways. The first option is to create a parsing tool for questions and let an AI decide whether the question list is acceptable. The second option is to unconditionally trust a given quiz master and accept all question lists that they make. Option one involves some fairly complex technology and option two requires one to trust the quiz master, which may not always be the case. Deciding which question lists to subscribe to will therefore likely be a manual task for the participant, unless they want to go through the trouble of automating the task, or they choose to trust a third party.

When a participant decides they want to subscribe to a question list, they have to use the protocol given below. As the input to the algorithm, we provide a manifest and a list of tuples consisting of:

- a question q
- a question type $k \in \{\text{‘push qa’}, \text{‘pull qa’}\}$
- either an answer or commitment a
- authorisation set $g \in \mathcal{P}(\mathbb{G}) \cup \{\text{‘public’}\}$

Algorithm 5: Protocol subscribing to a question list

```
input: manifest $M$,
        set of tuples $T := \{t | t = \langle q, k, a, g \rangle\}$
    let $M'$ := $M$
    foreach $\langle q, k, a, g \rangle \in T$
        if $g$ is ‘public’
            $M'[k]$ := $M'[k] \cup \langle \langle q \rangle, a \rangle$
        else
            let $e$ := encrypt $a$ for each of the groups in $g$
            $M'[\text{‘push qa’}]$ := $M'[\text{‘push qa’}] \cup \langle \langle q \rangle, g, e \rangle$
        end
    end
    publish $M'$ as an update to $M$
end
```

5.5 Implementation of Protocol 5: creating a manifest

When a party creates a product, they need to create and publish a new manifest for it. How to create a manifest is described in Algorithm 6. Given that p is the party creating the manifest, we choose p' to be the anonymous one-time identity of p, h the public holder key, M_l to be the set of manifests related to product links, and M_i is the set of manifests of the parent and child products (network links). Finalle, M_j is the set of manifests that are part of the manifest being created. So for example if the manifest is being created for e.g. a computer, M_j could contain screen, hard drive, keyboard, etc.

Algorithm 6: Protocol create manifest

```
input:  CIRLABEL $c$,
        public holder key $h$,
        anonymous party $p'$,
        question list $L$,
        set of QA pairs $PushQA$,
        set of TC pairs $PullTC$,
        set of manifests $M_l = \{M_{l_1}, M_{l_2}, \ldots, M_{l_n}\}$,
        set of manifests $M_i = \{M_{i_1}, M_{i_2}, \ldots, M_{i_m}\}$,
        set of manifests $M_j = \{M_{i_1}, M_{i_2}, \ldots, M_{i_o}\}$, %[pwp1] added input parts
        string $meta$,

    let $M$ := an empty dictionary
    set $M$['question list'] := $L$
    set $M$['metadata'] := $meta$

    set $M$['network links'] := $\{pt \mid pt$ is a pointer to $m,\ m \in M_l\}$

    let $E_h$ := encryption function for $h$
    set $M$['product links'] :=
        $\{e \mid e = E_h(pt),\ pt$ is a pointer to $m \in M_i\}$
    set $M$['parts'] :=
        $\{e \mid e = E_h(pt),\ pt$ is a pointer to $m \in M_j\}$ %[pwp1] added parts declaration

    set $M$['push qa'] := $PushQA$
    set $M$['pull tc'] := $PullTC$
    set $M$['creator'] := $p'$
    set $M$['holder'] := $h$

    publish $M$ as the manifest of $c$
end
```

In Algorithm 7 (updating a manifest) a new manifest is created as well, with the exception that a signature $s_{p'}$ over all other inputs is added to the list of inputs. This signature should demonstrate knowledge of the secret key counterpart to p', the creator of manifest M. If the signature is valid, an updated version of M is published.

Algorithm 7: Protocol update manifest

```
input:
        manifest $M$
        question list $L$,
        set of QA pairs $PushQA$,
        set of TC pairs $PullTC$,
        set of manifests $M_l = \{M_{l_1}, M_{l_2}, \ldots, M_{l_n}\}$,
        set of manifests $M_i = \{M_{i_1}, M_{i_2}, \ldots, M_{i_n}\}$, %[pwp1] this is now input parts, not p
        string $meta$,
        signature $s_{p'}$

    let $p'$ := $M$['creator']
    let $h$ := $M_c$['holder']

    if $s_{p'}$ is not a valid signature w.r.t $p'$
```

```
            return
        else
            let $M'$ := $M$
            set $M'$[‘question list’] := $L$
            set $M'$[‘metadata’] := $meta$

            set $M'$[‘network links’] := $\{pt \mid pt = pointer\ to\ m,\ m \in M_l\}$

            let $E_h$ := encryption function for $h$
            set $M'$[‘parts’] := %[pwp1] changed from plinks to parts
                $\{e \mid e = E_h(pt),\ pt = pointer\ to\ m,\ m \in M_i\}$

            set $M'$[‘push qa’] := $PushQA$
            set $M'$[‘push qc’] := $PullTC$

            publish $M'$ as an update to $M$
        end
end
```

As you can see, we only allow updating the ‘parts’ field of a manifest, and not the ‘network links’ field. The reason is, the fact that a product was made using some other product cannot be changed, while the fact that a product currently contains a part that can be swapped for another, can.

Linking to a manifest can be done by anyone, but network links have to be confirmed by parent manifests for them to be considered valid links in the network. In Algorithm 8 the creator of the parent manifest M_p chooses to confirm a link made by the child manifest M_c.

Algorithm 8: Protocol confirm link manifest

```
input:
            manifest $M_c$ ,
            manifest $M_p \in M_c$[‘network links’] ,

            let $M_p'$ := $M_p$

            set $M_p'$[‘network links’] := $M_p$[‘network links’] $\cup \{M_c\}$

            publish $M_p'$ as an update to $M_p$
        end
end
```

5.6 Implementation of Protocol 6: Asking a push-question

When a participant wants to get the answer to a push-question about a product, they scan the CIR**LABEL** to find its manifest. If the manifest does not contain a QA pair that answers the question directly, they can recursively follow the network links to find manifests that do, and combine the answers. For example, if the manifest of a T.V. does not specify whether it contains mercury, they can still obtain a full answer if each of its components do specify it, or a partial one if only some do. For each QA pair, the participant has to check if they are authorised to read it. The exact protocol is shown in Algorithm 9.

Algorithm 9: Protocol for asking a push-question

```
input: CIRLABEL $c$, question $q$, participant $p$
    let $M$ := the manifest that $c$ links to
    let $\mathbb{M}$ := findContainingManifests$(q, M, \emptyset)$

    if $\mathbb{M} = \emptyset$
        return // The question is not supported
    end

    let $A$ := $\emptyset$
    foreach $M' \in M_s$
        find $\langle L, G, E\rangle \in M'$[‘push qa’] , where $q \in L$

        if $G = \emptyset$
            set $A$ := $A \cup E$ // Answer is public
        else
            find an $i$ such that $p \in G_i \in G$
            if $i$ was found // Check if p is authorised
                let $D_i :=$ decryption function of $G_i$
                let $k := D_i(E_i)$ // Decryption of $E_{G_i}(k)$
                let $D_k :=$ decryption function of $k$
                set $A := A \cup D_k(E_k(S))$
            end
        end
    end

    return $q$ dependent merge of $A$
end

fun findContainingManifests
input question $q$, manifest $M$, accumulator $r$
    let $A_{push} := \langle L, G, E\rangle \in M$[‘push qa’], where $q \in L$
    let $A_{pull} := \langle T, C\rangle \in M$[‘pull qa’], where $q$ is covered by $T$

    if $A_{push}$ or $A_{pull}$ exist
        set $r$ := $r \cup M$
    else
        foreach $M_l \in M$[‘network links’]
            let $M'_l$ := findContainingManifests$(M_l, q, r)$

            if $M'_l$ exists
                set $r$ := $r \cup M'_l$
            end
        end
    end
    return r
end
```

5.7 Implementation of Protocol 7: Asking a pull-question

Pull-questions require a more involved Q&A protocol than push-questions. As the answers are not stored in the manifest like in push-questions, participants need to first find the creator of the manifests, then ask them the question. By scanning the CIR**LABEL** of a product, one can find the manifest. The manifests that contain relevant information can be found by recursively tracking the network links through the network and making use of the topics of TC pairs to filter for relevance. For each of the relevant manifests, the inquirer does not (and shall not) know the real identity of the party that created it, but they can read their anonymous identity, and the groups they belong to, off the respective manifests.

Before posing a question, the creator of the manifest that contains the appropriate topic is asked which groups are authorised to receive an answer to the question. The response determines for which group the inquirer should create a ring signature (if they are part of one). If the respondent does not answer this question, they can respond that no one is authorised for this question. Respondents are advised to keep track of asked questions they do not answer so that they can decide whether they want to support the questions in the future.

With a proper signature, an inquirer can then pose the question and provide a proof of authorisation for the answer. After posing a question to all the creators of the relevant manifests, the inquirer can wait for the answers.

Algorithm 10: Asking a pull-question

```
input CIRLABEL $c$, question $q$, inquirer $i$
    let $M$ := the manifest that $c$ links to
    let $\mathbb{M}$ :=
        findContainingManifests($M, q, \emptyset$) // from Algorithm 9

    foreach $M' \in \mathbb{M}$
        send $\langle M', q \rangle$ to $M'$['creator']
    end
end

fun onAuth
input manifest $M$, question $q$, groups $G$
    let $i$ := identity of inquirer
    let $k_s$ := private key of $i$

    find $j$ such that $i \in G_j \in G$
    if no $j$ found
        return // Inquirer not authorised
    end
    let $K$ := $\{k_p \mid k_p = \text{public key of } p, p \in G_j \setminus \{i\}\}$
    let $S$ := ring signature on $Q$, using $K$ and $k_s$
    send $\langle M, q, S\rangle$ to $M$['creator']
end

fun onAnswer
input question $q$, answers $X$
    return $q$ dependent merge of $X$
end
```

When a respondent is asked who is authorised to access a pull-question answer pertaining to some manifest, they use their local system to retrieve the list of the groups which have access. Such list is sent to the inquirer so that they can provide a ring signature to prove that they belong to any one of them. If they can't, they won't get the answer.

Next, the respondent can verify the provided signature. If this succeeds, the answer is forwarded to the inquirer. This answer can either be a direct answer to the question, or an answer in the form of a proof of ranges (see Appendix B).

Algorithm 11: Answering a pull-question

```
fun onAuth
input manifest $M$, question $q$, inquirer $i$
    let $G$ := authorise($M$,$q$) // groups authorised for $\langle M, q\rangle$

    send $\langle M, Q, G\rangle$ to $i$
end

fun onQuestion
input manifest $M$, question $q$, signature $S$, inquirer $i$
    let $G$ := authorise($M$,$q$)

    if $S$ authorised for one of $G$
        let $x$ := answer to $q$
        send $\langle q, \{x\}\rangle$ to $i$
    end
end
```

The inquirer receives either a single answer, or a collection of more answers that will have to be merged, depending on which parties in the value chain have the topic in their lists. How the answers are merged depends on the question. For a question on the total amount of a certain ingredient included in a product, the answer is the sum of all answers, while a question on whether there is more or less of an ingredient than a certain threshold requires a different calculation which might only have a case-by-case solution.

5.7.1 Security

When the answers are to remain a secret for anyone but the inquirer and respondent, Diffie-Hellman key exchange comes into play.

Following the protocol for executing a Diffie-Hellman key exchange (see Appendix C), we consider Alice the inquirer and Bob a respondent. Alice needs to send three values (along with the question): the two chosen prime numbers g and N, and the generated number $A = g^a \mod N$ for Alice's secret a. Bob uses these values, and his own secret b, to calculate the shared secret $s = A^b \mod N$. The secret is then used to encrypt the answer x with the symmetric encryption function E_s before responding. Bob's response is the encrypted answer $E_s(x)$, and the additional number $B = g^b \mod N$ which Alice requires to calculate s herself. Using Bob's response, Alice calculates the shared secret $s = B^a \mod N$ and decrypts the answer $x = E_s(E_s(x))$.

Know that a Diffie-Hellman key exchange is susceptible to man-in-the-middle attacks. It is possible for an intermediate Charlie to create his own parameters upon receive those of Alice. By sending Alice and Bob both proper responses, Charlie can create two simultaneous tunnels and become able to read messages of both Alice and Bob. Thus Charlie can pass on the signed question of Alice to Bob and obtain an answer that is meant for Alice.

Man-in-the-middle attacks are countered by letting Alice sign the complete message instead of only the asked question using ring signatures. This gives Bob the guarantee that both the question *and* the Diffie-Hellman parameters come from an authorised participant. Bob is not required to sign his parameters as Alice does not need the same guarantee of authorisation. If someone alters the response for Alice, the only result will be that she will not be able to read the answer. This would be annoying, but not harmful. Bob's answer will not have been read by an unauthorised party.

Similarly to providing a tamper-proof tunnel between the respondent and the inquirer, if an answer is based on a commitment they can be given by providing extra information in the answer. The proof of ranges of Boudot (2000) does not require a challenge of the inquirer. Thus any response given by the respondent can include the complete proof for a commitment without requiring additional communication.

5.8 Implementation of Protocol 8: Storing and verifying identification

Each participant p can claim a name conforming to Ethereum Improvement Proposal (EIP) 137. Names are represented on-chain as a hash, which allows us to have identifiers that are theoretically unbounded in length. There is only one public registry.

Algorithm 12: Claiming an identity

```
input identity $i \in I$, participant $p \in \mathbb{P}$
    let $R :=$ fetched registry
    if $R_{\mathbb{P} \mapsto I_c}[p]$ is defined
        return // mapping already exists
    elif $R_{I_c \mapsto \mathbb{P}}[i]$ is defined
        return // mapping already exists
    else
        $R' := R \cup \{(i,p),(p,i)\}$
        publish $R'$ as a new version of $R$
    end
end
```

Interacting with identities is done by accessing the registry R, as demonstrated in the following protocol. Verifying the published information has to be done manually in the general case by each participant using real world channels. As interactions in the network usually model interactions in the real world, this leaves ample opportunity to verify the claimed identities prior to any interaction through Circularise.

Algorithm 13: Retrieving an identity

```
fun getIdentity
input participant $p \in \mathbb{P}$, registry $R$
    let $i$ := $R_{\mathbb{P} \mapsto I_c}[p]$
    if $i$ is defined
        return $i$
    else
        return // No existing mapping
    end
end

fun getAddress
input identity $i \in I$, registry $R$
    let $p$ := $R_{I_c \mapsto \mathbb{P}}[i]$
    if $p$ is defined
        return $p$
    else
        return // No existing mapping
    end
end
```

Advanced features such as name caching and changed ownership of names are currently not supported, they will be in the coming versions.

6 Goal Validation

In the introduction we have defined a set of goals we wish to achieve via Circularise. In the previous sections, we have defined a number of protocols that regulate the actions that the users can take on the Circularise platform. In this section we argue that these protocols are sufficient to achieve those goals: that by using the Circularise platform with these functionalities, the resulting system achieves the goals we had set earlier.

6.1 How we meet The Goal: ensuring that information can be shared

We enable information sharing in two ways: pull-questions and push-questions.

Firstly, information providers can permanently attach information to a CIR**LABEL**'s manifest in the form of push-questions, using the protocol for creating a manifest in Section 5.5. That information can be accessed by anyone with the proper authorisation, using the push-question protocol from Section 5.6. Because the push-questions and their answers are stored in the manifest on the blockchain, they are always available, even if the party that made them no longer exists. This makes push-questions a good way to share information that should always be available, like information that you are legally obliged to provide with the product, or a user manual.

Secondly, the pull-question protocol from Section 5.7 can be used to ask free-form questions that are anonymously forwarded to the participants who know the answer. The pull-question system addresses **B.2**, the importance of some information being unknown, by allowing the inquirer to ask any kind of question, not just predetermined ones. It also addresses **A**, because questions are dynamically forwarded to the right participants after which all relevant answers are sent back to the inquirer. If the information is spread among multiple parties, each of them can answer the inquirer. Pull-questions are meant to share that information that you did not know was important when creating your manifest, or to control the sharing of sensitive information that parties want to keep in their possession at all times.

The combination of push- and pull-questions satisfies **The Goal**, to enable sharing information. Now we have to verify that the way in which **The Goal** is satisfied respects the rest of the subgoals.

6.2 How we meet A.1: participants stay in control of their product information

A.1 refers to the ability of participants to decide what happens to their information once they put it in the circularise system.

Participants regulate their product information in two ways, they decide with whom information is shared and how much. Participants can control what information is shared by fine-tuning when to answer a question. Similarly, they control who obtains the information by making their answers readable only by those in possession of the right key (using groups).

6.2.1 How we meet A.1.1: Fine-tune which information is shared

Information is shared only as answers to questions. In this way, we ensure that the control of whether and when to answer remains with the information's owners. Subscribing to questions lists, participants can control what information they share. Note that they can always create their own question list if they so desire.

Recall that there are two kinds of questions: pull-questions and push-questions. Answers to push-questions are stored on the blockchain, therefore the choice of answering these questions can not be reconsidered. For pull-questions the participant can decide on a case-by-case basis whether they want to answer the question or not as these requires active responses. A given answer for a pull-question is possessed by the inquirer, but unlike a QA pair it is not available on a public network.

Although there are a number of TC pairs stored in a manifest for pull-questions, these pairs contain no publicly readable information. They can only be used to verify whether a given answer to a pull-question is linked to them, which makes them auditable.

6.2.2 How we meet A.1.2: Fine-tune with whom information is shared

Once again we consider the two types of questions. For each type of question, a different means is used to allow parties to decide who has access to what information.

The answers to push-questions are stored in product manifests. Their authorisation is handled by encrypting the answers using a symmetric key and encrypting the key using the public keys of groups that are allowed to read it. The latest public key of a group is always published as described in Algorithm 1. Thus encryption can always be done with the group's current members in mind. Following Algorithm 2, future members will also gain access to this information as the group's private key is shared with them when they join.

One thing to keep in mind is that it cannot be assumed that the current members of a group will delete their private key once they are no longer part of it. This means they will always be able to read the answers that were encrypted when they were still members. So, if one has access to some information by virtue of being members of some group, they will always have access to that information (but not to updates of that information that happen after they leave the group).

For pull-questions the authorisation can be tuned more finely. The ring signature that is provided by the inquirer (see Algorithm 10) proves that they are one of a group of participants. If the respondent decides that any current member of this group is allowed to know the answer, they can return an answer. If not, they can simply refuse to.

6.3 How we meet A.2: parties stay in control of their network information

In the Circularise system, information is shared in two ways: through the information in the manifests (accessible by the push-question protocol) and through the pull-question protocol. It is important that in either case, no leakage about the real structure of the value chain can occur.

The first step Circularise takes is the anonymisation of the creators of manifests by using “throw-away” wallets. Only the label holders are able to reveal the public identity of the manifest creators, but being a tier 1 recipient of the manifest’s creators, they are in possession of that information anyway – they don’t obtain it through Circularise.

The only information one can deduce from a manifest is the number of Circularise participants that are involved in the value chain of that specific product. At the same time, participants in search for information are able to find who can answer the questions without giving away their identity. Additionally, participants that are willing to provide an answer, can do so anonymous as well.

Creators of manifests and respondents are also kept anonymous during pull-questions. Inquirers themselves remain anonymous as well since they use ring signatures which only disclose that one of a group of parties is asking a question (but not which).

The communicated identities are however not the only way network information can leak. The information about the product that is stored in the manifest is also a liability. Manifests creators could accidentally add identifying information to their manifest. For example, a company wishing to remain anonymous may mistakenly add to a manifest one of its product’s names in plaintext, and a google search may reveal that the product name is linked to the company. However, we do enable them to hide this information in higher level manifests through product links which are encrypted for the holders only, but holders already know the identity of the creator.

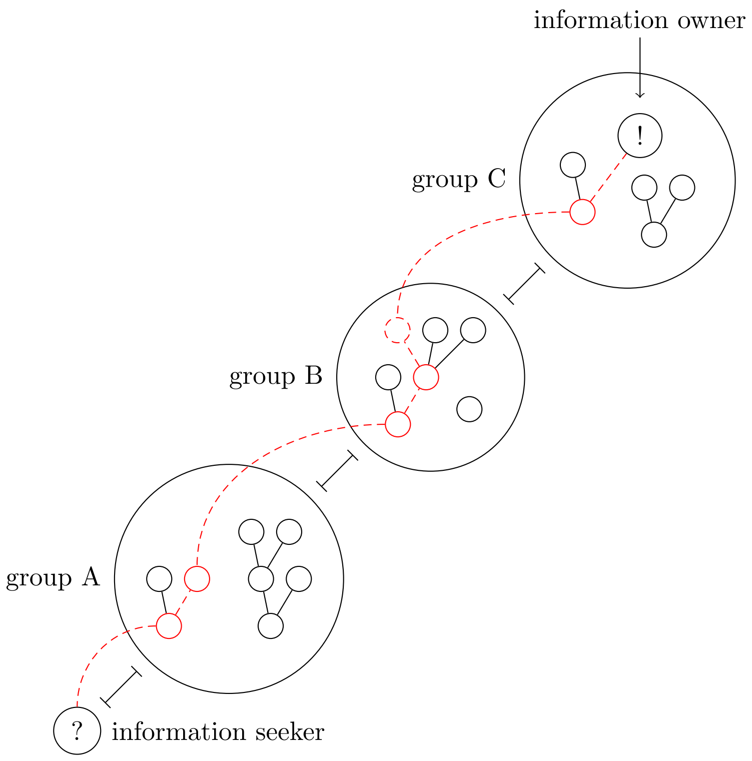
There is not much Circularise can do about bad citizens. However, due to the nature of the question and answer system, exchanging contact information is not made easy. Questions can only be asked if they are linked to a question list and even if there exists a question list that contains the question “what is your email?”, a party would have to publicly subscribe to this list. Public subscriptions to “bad” question lists can be used to identify misbheaviour. Answers on the other hand can be anything. The answer to the amount of mercury in a T.V. component could very well be an encrypted email address in an attempt to establish communication with another party to cut out the middleman. Only the targeted groups would be able to notice this. Circularise cannot prevent parties from adding contact information in a manifest. However, since a party does not know who the inquirer is, they also do not know if the inquirer *wants* to know their contact information. Their faulty answer could just as well be used to invoke an audit.

6.4 How we meet B.1: ensuring that information is trustworthy

The trustworthiness of the information in Circularise is an emergent property of the behaviour of its participants. To incentivise good behaviour, Circularise itself should be trustworthy as well.

Stimulating good behaviour In current, real-world value chains, parties are incentivised to behave by means of controls and punishments: audits. Circularise facilitates auditing by allowing the information that is stored to be permanent and verifiable: given some extra information, auditors are able to read the immutable information and verify its correctness. Information that has already been verified by auditors can be considered trustworthy, but it is is undoable to verify every bit of information in Circularise. Information will therefore be audited by samples.

Participants of Circularise are given incentive to post correct information, as their answers can be traced back without error to their real identities. Suppose that someone obtains a manifest and tries to retrieve information from it (push or pull-questions). If the information it retrieves is incorrect, the party in question may choose to begin following the network links in the manifest backwards, and with the cooperation of the previous owners of the linked manifests, it will eventually reach the owner of the "bad" manifest: Section 6.4.



Auditors, in particular, have the authority to obtain this cooperation and will thus be able to use this procedure to find the source of bad information.[10].

So, if wrong information is found, the associated party can be held accountable by following the trail (cfr. Section 6.4.1 for details).

However, the question remains whether all answers are auditable. For push-questions, the answer is one-to-one on the blockchain, so whatever the answer was that was verified by an auditor will also be the answer that a inquirer will read. On the other hand, pull-questions are answered by the respondent, not by some information in a manifest. Though, if answers are based on the TC pairs, it will be known that the given answer is provably linked to some piece of information on the blockchain (which can in turn be verified by auditors). As we expect most pull-questions to be answered based on TC pairs, most answers will be auditable.

Trustworthiness of the system By making the Circularise system open-source, we guarantee that third parties will be able to independently inspect the code and convince themselves of its trustworthiness. Furthermore, because the system is decentralised, the data does not go to (or through) Circularise either. The important actions of the system like managing groups, question lists and identification of participants are not dependent on Circularise. Moreover, any of these actions that are taken by Circularise are visible on the public blockchain. This combination of open-source, decentralisation and transparency ensures that only a minimal amount of trust has to be placed in Circularise in order to trust the system.

6.4.1 How we address B.1.1: Make information auditable

While auditing needs to take place through normal channels, Circularise offers a way to permanently, immutably log information that can be used to trace misbehaviour. The information published through Circularise can be audited just like any other property of the party who owns/published it.

The goal of the auditor is to verify that participants are not misbehaving. For an audit, the cooperation of the auditee is always required. Cooperation for audits is not enforced by the system itself.

If misbehaviour *is* detected, the auditors can take appropriate action[11].

On demand auditing An auditor who is auditing a participant could, in addition to its normal checks, review the auditee's manifests. First they need to find out which manifests were created by the auditee. As the real identity of a manifest owner is a secret that is known by only the creators and its holders, the auditor will not be able to find the manifests without their help. Asking the auditee for its manifests is one way to achieve this, though this does allow a dishonest auditee to hide manifests (by not telling the auditor about them).

[10]Circularise does not have the power to enforce punishments for misbehaviour, but we assume that auditors do.

[11]At this point, it is unsure whether information shared on Circularise is legally binding. The bottom line is that it is possible to detect misbehaviour.

With the manifest in hand, the content can be verified. For a manifest, there are three kinds of things that need to be audited: QA pairs, TC pairs and the edit history of the manifest.

For public accessible QA pairs nothing special has to be done, the values can be read by the auditor at their leisure. Encrypted values of QA pairs can be shown by either the creator of the manifest or the groups for which they are encrypted. As the creator of the QA pairs made the symmetric key, they are able to show the values of the QA pairs on the basis of the data stored on the blockchain. This proves that the shown value corresponds to the value on the manifest. It does not prove that what is encrypted for the groups is the actual symmetric key, however these parties are able to alert auditors if they obtain a faulty symmetric key.

For the TC pairs in the manifest, the auditee can provide the hidden value and associated random value. As it is infeasible to find a pair that results in the same commitment, this pair can be considered the original one. The auditor can then verify the correctness of the hidden value using real-world channels.

In this way, the auditor can verify that the information participants disclose about their products matches the truth. In the future, we could label verified data as such, to mark that it is more trustworthy than unaudited data, and even score participants according to how often an audit has confirmed information that they have provided.

Secondly, the history of the manifest can be checked using the transaction history of the blockchain. This can be used to verify whether the auditee shows any suspicious behaviour, such as whether the manifest was edited just previous to the audit to hide a lie.

Thirdly, manifests contain information that can certify the provenance of products and their production history, some of which may be subject to regulation. The data contained in manifests can thus be used by auditors to determine whether the associated product was manufactured and obtained through legitimate channels and procedures.

On discovery of a wrong answer Suppose that a recycler receives information that turns out to be wrong, for example, prior to attempting to recycle a TV screen it asks whether there is mercury in it, and receives a “no” as an answer, while in fact there is. Thanks to the traceability of our questioning protocols, it would be possible for an auditor to track down the origin of the false information. In the case of push questions, the secret identity of the origin is contained in the manifest that the QA pair comes from (cf. Protocol 6). For pull questions, the secret identity of the origin is known the very moment a communication channel is opened. In either case the wronged parties can call for an auditor to find the source of the problem.

Linking a participant to an answer is easier than finding out all the manifests of a participant. The participant that calls for an audit at least knows the identity of their tier 1 suppliers. They have seen the holder private key for their manifests, which they can use to decrypt the manifest’s public identity for the auditor. This step can be repeated for each party in the value chain until the respondent is found.

6.4.2 How we meet B.1.2

The Circularise network has two kinds of identities: public, long-lived identities that are directly linked to a real-world entity, and anonymous identities which are used to publish the actual manifests. The link between these two categories of identities can be revealed, with each party in the chain being able to reveal the real identities of their direct suppliers in the network, using the holder key.

The trail can only have dead end if a participant either didn't add the proper network links to their manifest, or they lost the holder private key which is needed to decrypt the identity of their supplier. In either of these cases, that participant is easy to identify by following the trail to them, and they will be held responsible. The trail may still be picked up by going through real-world channels, e.g., by forcing them to reveal who their suppliers are.

6.5 How we meet B.2: Future-proofing of data

One of the goals of Circularise is to facilitate answering questions that are initially unknown. With TC pairs, participants can make commitments to information that they want to keep secret, even if there is no accompanying question. The commitments are however auditable from the moment they are made and, once new questions are added, provide a verifiable way of answering them[12].

Pull-question allow parties to pose and answer questions that did not exist when the product was fabricated. A party needs to update their subscribed question lists to allow for the new question and make sure that they answer the question when it is asked.

6.6 How we address B.3: Robustness to parties leaving

Each participant in Circularise is a potential source of information for the network. Similarly participants who choose to leave can mean that some information is no longer available for the network. The goal, then, is to minimise the amount of information that is lost when this happens.

As QA pairs are stored immutably on the blockchain, they are well preserved. A party would not be able to remove the information even if they wanted to. So push-questions that a party answered while being part of Circularise will be available even after the party leaves.

One thing that cannot be retained is the leaver's ability to answer pull-questions. This is a trade-off between **A** and **B.3**. It is impossible to fully satisfy one goal without violating the other. Providing complete resilience to leaving parties would involve giving someone (or something) else a backup of the information such that they can act as a stand-in for the leaving party. In

[12]Due to the nature of commitments, questions on them can only be about quantities/amounts.

doing so, the party loses control over their information as they would need to trust the third party to not abuse their information. The bottom line is, when a party leaves, they take their information with them. So if they are the only one that are able to answer a specific pull-question, the answer is lost.

On the other hand, it is a good feature of Circularise that a leaving party does not break the network structure used for asking pull-questions. Since the network links are publicly shared, one can always target the (anonymous) creator of a manifest for asking a question anywhere in the value chain. Thus as long as the answer to a pull-question is known by parties that are still active in the network, the question is answerable.[13]

So, even supposing that all of your suppliers and all of your recipients leave Circularise, all parties in search of information you have will always be able to find you, so long as you don't leave as well.

Even though most pull-questions can still be answered after parties have left Circularise, the identifiability of respondents deteriorates. Only their direct children in the network will have had access to the holder key of their manifest. If a party is left without direct children in the network, even though they can still answer pull-questions, no one will be able to identify them. Because the real identity of the party is lost, they can no longer be held liable for their answers. This makes the answers given by them less trustworthy, unless they choose to make themselves available for audits.

As an opt-in feature, companies could store their data on something like IPFS to ensure data perseverance.

[13]This is only partially correct: if the TV manufacturer is gone, and so is the screen manufacturer, the info owned by its peers may only be partial.

References

Ethereum avg. transaction fee historical chart. `https://bitinfocharts.com/comparison/ethereum-transactionfees.html`, 2018. [Online; accessed 31-05-2018].

Mikael Skou Andersen. An introductory note on the environmental economics of the circular economy. *Sustainability Science*, 2(1):133–140, 2007.

Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 326–349. ACM, 2012.

Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 431–444. Springer, 2000.

Agnes Chan, Yair Frankel, and Yiannis Tsiounis. Easy come—easy go divisible cash. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 561–575. Springer, 1998.

David Chaum and Eugène Van Heyst. Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 257–265. Springer, 1991.

Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE, 1979.

Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–565. Springer, 2001.

Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11): 612–613, 1979.

Glossary

CIRBASE The platform implementation on which all of Circularise is build on.. 2, 17–19, 21

CIRCOIN The Circularise token. 17

CIRLABEL The Circularise label. 9, 10, 17, 21, 26, 27, 32, 34–36, 40

auditor A party that checks whether participants provide the correct and consistent answers.. 5, 14, 23

bluepaper Our business paper.. 4

Circularise The network, protocol, company and product. 1, 5, 8, 10–22, 28, 41–44, 46, 47, 49

Diffie-Hellman key exchange Protocol for exchanging a secret through a public channel Diffie and Hellman (1976). 14, 15, 37, 38, 54

group A listing or categorisation of a bunch of parties. Examples include recycler, retailer, etc. Any party can unilaterally decide to create a group of e.g. recyclers. Party P's Recycler Group is Party A, Party B, Party C.... 2, 8, 14–16, 19–26, 29, 30, 41, 42, 45

inquirer The party initiating a Q&A session. Not expected. Quite possibly Spanish. 12, 14, 19, 21, 22, 26, 36–38, 40–42, 44

introducer A party that is authorised to classify other parties (and thus introduce them to the system). 30

manifest Metadata for a specific product stage. 9, 17–19, 21, 22, 25–27, 29, 32–37, 40–42, 44–47, 49, 50

network link A manifest-link that links a product-manifest to one of its input-manifests. 18, 21, 22, 26–28, 32–36, 43, 46, 47

orphaned information Information whose owner has left the network. 13

participant The main user of the Circularise network. These are the guys that ask questions and/or provide answers. 1, 3, 11, 17–26, 28–32, 34–36, 38, 40–46, 49

party The user of the Circularise network. 3–5, 7–11, 13, 15–19, 26, 28, 30, 32, 36, 40–47, 49

product link A manifest-link that links a product-manifest to a higher level entity such as a brand, category, product or Stock Keeping Unit (SKU). 18, 21, 27, 28, 32, 33, 42

pull-question Questions which require navigating through the tree to obtain an answer. 2, 19, 25, 26, 36, 37, 41, 42, 44, 46, 47, 53

push-question Questions for which the answer is provided upfront along with the manifest. 2, 18, 19, 21, 25, 34–36, 41, 44, 46

quiz master The owner of a question category.. 20, 21, 31, 32

respondent The party providing (one of) the answer(s) to a Q&A session. 14, 22, 26, 36–38, 41, 42, 44, 46, 47

Acronyms

CAS Content Addressable Storage. 12

EIP Ethereum Improvement Proposal. 38

IPFS InterPlanetary File System. 12, 13

QA pair question/answer pair. 2, 25, 27, 34, 35, 41, 45, 46

SKU Stock Keeping Unit. 50

TC pair topic/commitment pair. 2, 25–27, 36, 41, 44–46

ZKP Zero-Knowledge Proof. 1, 13, 14

zkSNARK Zero-Knowledge Succinct Non-interactive Argument of Knowledge. 2, 11, 12

A Zero-Knowledge proofs of ranges

This appendix summarizes the methods that we adapt from (Chan et al., 1998) and (Boudot, 2000) to prove that a number belongs to a given interval without sharing any extra information about the number. We treat the protocols in order of increasing precision of the desired range that an unshared number x can be proven to lie in.

To ease introduction of the protocols that follow, some definitions are in order. We define $E(x_1, r_1)$ to be a commitment to the value x_1, which can be computed by knowing (x_1, r_1) but for which it is computationally infeasible to find any other pair (x_2, r_2) such that $E(x_1, r_1) = E(x_2, r_2)$. Unless otherwise indicated, N is a sufficiently large composite number with a factorisation that is unknown by both Alice and Bob. Let g be an element with high cardinality from $\mathbb{Z}_n^*$ and h generated by g such that for $h = g^a \mod N$ and $g = h^b \mod N$ a and b are unknown for the challenger. Also take $E = E(x, r) = g^x h^r \mod N$ to mean a commitment of $x \in [a, b]$, with r being a random integer from $[-2^s N + 1, 2^s N - 1]$. H is some hash-function with an output length of $2t$ bits. Security parameters t, l, s, known by both Bob and Alice, are chosen in such a way to satisfy the security requirements for any specific exchange. In his work, Boudot defines a number of protocols for a proof of knowledge $PK(\textit{variables} : \textit{predicate})$ which states a zero knowledge proof that given a number of *variables*, the *predicate* holds. $PK_{[square]}$ allows us to show that a commitment hides a squared number, and is taken from (Boudot, 2000).

Protocol: $PK_{[CFG]}(x, r : E = E(x, r) \wedge x \in [-2^{t+l}b, 2^{t+l}b])$. For this protocol, $a = 0$.

1. Alice chooses $\omega \in_R [0, 2^{t+l}b - 1]$ and $\eta \in_R [-2^{t+l+s}N + 1, 2^{t+l+s}N - 1]$.
2. Alice computes $W = g^\omega h^\eta \mod N$
3. Alice computes $C = H(W)$ and $c = C \mod 2^t$
4. Alice computes $D_1 = \omega + xc$ and $D_2 = \eta + rc$.
5. If $D_1 \in [cb, 2^{t+l}b - 1]$, Alice sends (C, D_1, D_2) to Bob. Otherwise start over with new choices of ω and η.
6. Bob verifies that $D_1 \in [cb, 2^{t+l}b - 1]$ and $C = H(g^{D_1} h^{D_2} E^{-c})$ and can subsequently conclude that $x \in [-2^{t+l}b, 2^{t+l}b]$.

Protocol: $PK_{[WithTol.]}(x, r : E = E(x, r) \wedge x \in [a - \theta, b + \theta]$ For this protocol, $\theta = 2^{t+l+1}\sqrt{b - a}$.

1. $\tilde{E} = E/g^a \mod N$, $\overline{E} = g^b/E \mod N$
2. Then: $\tilde{x} = x - a$ and $\overline{x} = b - x$.
3. $\tilde{E} = g^{x-a} h^r \mod N$
4. Compute $\tilde{x}_1 = \lfloor \sqrt{x - a} \rfloor$,$\tilde{x}_2 = \tilde{x} - \tilde{x}_1^2$

5. $\overline{x}_1 = \lfloor\sqrt{b-a}\rfloor, \overline{x}_2 = \overline{x} - \overline{x}_1^2$

6. Then it holds that: $\tilde{x} = \tilde{x}_1^2 + \tilde{x}_2$, where $0 \leq \tilde{x}_2 \leq 2\sqrt{b-a}$, and: $\overline{x} = \overline{x}_1^2 + \overline{x}_2$, where $0 \leq \overline{x}_2 \leq 2\sqrt{b-a}$

7. Randomly select $\tilde{r}_1, \tilde{r}_2 \in [-2^s n + 1, 2^s n - 1]$ with $\tilde{r}_1 \tilde{r}_2 = r$.

8. $\overline{r}_1, \overline{r}_2 \in [-2^s n + 1, 2^s n - 1]$ with $\overline{r}_1, \overline{r}_2 = r$.

9. $\tilde{E}_1 = E(\tilde{x}_1^2, \tilde{r}_1), \tilde{E}_2 = E(\tilde{x}_2, \tilde{r}_2)$

10. $\overline{E}_1 = E(\overline{x}_1^2, \overline{r}_1), \overline{E}_2 = E(\overline{x}_2, \overline{r}_2)$

11. Send $\tilde{E}_1$ and $\overline{E}_1$ to Bob. He computes $\tilde{E}_2 = \tilde{E}/\tilde{E}_1$ and $\overline{E}_2 = \overline{E}/\overline{E}_1$

12. Now $PK_{[square]}(\tilde{x}_1, \tilde{r}_1 : \tilde{E}_1 = E(\tilde{x}_1^2, \tilde{r}_1))$ and $PK_{[square]}(\overline{x}_1, \overline{r}_1 : \overline{E}_1 = E(\overline{x}_1^2, \overline{r}_1))$ to prove that both $\tilde{E}_1$ and $\overline{E}_1$ hide a square.

13. With $\theta = 2^{t+l+1}\sqrt{b-a}$, execute: $PK_{[CFT]}(\tilde{x}_2, \tilde{r}_2 : \tilde{E}_2 = E(\tilde{x}_2, \tilde{r}_2) \wedge \tilde{x}_2 \in [-\theta, \theta])$ and $PK_{[CFT]}(\overline{x}_2, \overline{r}_2 : E(\overline{x}_2, \overline{r}_2) \wedge \overline{x}_2 \in [-\theta, \theta])$.

14. Bob now knows $\tilde{E}_1, \overline{E}_1$ hide positive numbers because they hide squares. $\tilde{E}_2, \overline{E}_2$ hide numbers which are greater than $-\theta$. Alice must know the values hidden by $\tilde{E}$ and $\overline{E}$, as she could not have known $\tilde{x}$ and $\overline{x}$ otherwise. $\tilde{E}$ is the sum of the numbers hidden in $\tilde{E}_1$ and $\tilde{E}_2$, and likewise for $\overline{E}, \overline{E}_1, \overline{E}_2$, convincing Bob that both $\tilde{E}$ and $\overline{E}$ hide numbers greater than $-\theta$. Bob is convinced that $\tilde{E}$ hides $\tilde{x}$, and $\overline{E}$ hides $\overline{x}$. So $\tilde{x} = x - a \geq -\theta$ and $\overline{x} = b - x \geq -\theta$. Then $x \in [a - \theta, b + \theta]$.

Protocol: $PK(x, r : E = E(x, r) \wedge x \in [a, b]$ For this protocol, $T = 2(t + l + 1) + |b - a|$, known by both Bob and Alice. For convenience, $\theta' = 2^{t+l+T/2+1}\sqrt{b-a}$.

1. Alice computes $x' = 2^T x$, $r' = 2^T r$.

2. Both Alice and Bob compute $E' = E^{2^T}$.

3. Execute $PK_{[WithTol.]}(x', r' : E' = E(x', r') \wedge x' \in [2^T a - \theta', 2^T b + \theta']$

4. Seeing $\theta' < 2^T$ leads Bob to be convinced that if $x' \in [2^T a - \theta', 2^T b + \theta']$, then $x' \in (2^T a - 2^T, 2^T b + 2^T)$

5. Given that Alice does not know the factorisation of n, $x' = 2^T x$, so $x \in (a - 1, b + 1)$. As $x \in \mathbb{Z}$, this implies that $x \in [a, b]$.

B Ring Signatures

This appendix describes the ring signature protocol. In his paper Shamir provides examples for RSA (Rivest et al., 1978) and the Rabin crypto system (Rabin, 1979), but basically any asymmetric encryption algorithm fits this purpose. In this paper we describe the ring signature scheme using Schnorr's elliptic curve signatures (Schnorr, 1991).

For the case of elliptic curve signatures, recall that a curve is defined by a sextuple of parameters $\langle p, a, b, G, n, h\rangle$, where G is the base point of the curve. Given a private key S, one can compute the public key P using $P = [S]G$, where $[S]$ is the S-th multiple of the point G given the elliptic curve. The keypoint here is that it is easy to compute P given S and G, but infeasible to compute S (hence being a trapdoor function).

Given a message m, a curve $\langle p, a, b, G, n, h\rangle$ and a set of public keys $\{P_0, \dots, P_{x-1}\}$, someone with access to one of the public keys' private key can create a signature on m proves that it was created by someone with access to one of the private keys (though it will not be known by which). Starting at their own key (assume P_0), a random point on the curve is chosen as a starting point given by $Q_1 = [k]G$. Q_1 is used in combination with m to form the input of a hashing function H[14] for which the output r_1 is used to calculate the next point on the curve. This next point is given by $Q_2 = [s_1]P_1 + [r_1]G$ for which s_1 is randomly chosen. The computation is repeated for each public key. The last point Q_{x-1} is then the starting point for the value r_0. As the computation should form a ring, instead of choosing s_0 random, a value should be found such that $[k]G = [s_0]G + [r_0]P_0$. It holds that $P_0 = [S]G$, it follows that $[k]G = [s_0]G + [r_0 * S]G$ and thus $s_0 = k - r_0 S$.

By now, there is a value s and r for each public key, which can be used in a circular computation where regardless of where you start, you end up at the same r as you started with. All that is left is shifting the indices, such that it becomes obfuscated which public key was the starting point (effectively obfuscating who the signer is).

Algorithm 14: Asking a pull-question

```
input:  message $m$,
        curve $\langle p, a, b, G, n, h\rangle$,
        public keys $\{P_0, \dots, P_{x-1}\}$,
        private key $S$

    take random $k \in \{1, \dots, n-1\}$
    for $i = 0$ to $x - 1$
        if i == 0
            let $Q = [k]G$
        else
            let $Q = [s_i]G + [r_i]P_i$
        end
```

[14]Usually SHA-256 is used, but any cryptographically secure hashing algorithm suffices.

```
        let $r_{i+1} = H(x_Q||y_Q||m)$
        take random $s_{i+1} \in \{1, \ldots, n-1\}$
    end
    let $r_0 = r_m$
    remove $r_x$, $s_x$
    let $s_0 = k - rS \mod n$
    shift indices of $s$ and $r$ randomly
    return $\langle r_0, s_0, \ldots, s_{x-1} \rangle$
end
```

A verifier obtains $m, \{P_0, \ldots, P_{m-1}\}$ and the created signature $\langle r_0, s_0, \ldots, s_{x-1} \rangle$. The signature can then be verified by computing the ring with $Q_{i+1} = [s_i]G + [r_i]P_i$ and $r_i = H(x_{Q_i}||y_{Q_i}||m)$. A signature where $r_0 = r_m$ is a valid signature.

C Diffie-Hellman Key Exchange

This appendix describes the Diffie-Hellman key exchange for secret key exchange.

The protocol:

1. Alice sends Bob two prime numbers g and N.
2. Alice creates a secret number a and computes $A = g^a \mod N$
3. Bob creates a secret number b and computes $B = g^b \mod N$
4. Bob and Alice exchange A and B
5. Bob can now compute secret s with $s = A^b \mod N$
6. Alice can do the same with $s = B^a \mod N$

The secret s is now shared by Alice and Bob with nobody else being able to construct it under the strong RSA assumption.