TODA Primer

Text by Adam Gravitis, Norman Goh, and Dann Toliver.

todaqfinance.com

TODA Primer

The Promise of Blockchain

Since its introduction a decade ago, blockchain technology has garnered an unprecedented level of interest from investors, journalists, engineers, entrepreneurs and developers.

Its first major application, the Bitcoin cryptocurrency, enthralled a public that was living in the shadow of the 2008 financial recession. Bitcoin, billed as a "peer-to-peer electronic cash system", promised to be a safe haven for those who have lost trust in the way banks and governments managed the economy. It wasn't controlled by a central authority and therefore free of government supervision and policies like capital controls.

While Bitcoin set the stage for spotlight interest in blockchains, it also set the stage for confusion. **What does blockchain do, exactly? And what problems can it solve?**

These days, you can't have a discussion about blockchain technology without hearing messianic levels of evangelism. Blockchain has been flogged as a panacea for everything from solving poverty to curing cancer. The market cap for all cryptocurrencies went from $17 billion to $565 billion in 2017. Investors looking to use blockchain and find another industry or sector they can disrupt. It has also become politicised, a byword for libertarian dogma bent on destroying all centralised concentrations of power, including governments, financial institutions and fiat currency.

This confusion is aggravated by the kinds of media coverage blockchains have garnered via news, social media and talk shows:

- Volatile cryptocurrencies whose values explode to $7000, then drop to pennies only to rebound again.
- A Dutch firm announcing it wants to put sexual consent on an immutable blockchain - having lovers sign unalterable electronic contracts and sending copies to thousands of strangers for safekeeping.
- Initial Coin Offerings (ICOs) generating billions in raised funds, only to be revealed as cons.

The sensationalized nature of media coverage does a disservice to an ingenious set of cryptographic technologies.

The problem blockchain is really good at solving - proving without a doubt that something happened (and its corollary, proving ownership of digital assets) - was obfuscated behind an international hype apparatus of buzzwords, cryptocurrency price fluctuations, fantastical dreams of disruption and thinly-veiled political dogma.

Arising from the excitement of a newfound panacea were blockchain-based projects that attempted to do too much too soon, creating solutions where problems didn't yet exist, all the while creating unneeded complexity and remaining blind to scalability issues.

Delivering on the Promise: TODA

TODA was designed to act as a foundational protocol, laser-focused on the key strength of decentralized technology: authentic, unforgeable record-keeping. Its overall design and architecture is fundamentally different than other blockchain projects, and so are the guarantees it provides.

Notably, TODA departs from the 'replicated ledger' approach to blockchain, and in doing so, departs from the significant expense and overhead those methods entail. TODA's underlying architecture was designed to solve many of the common deployment challenges today's blockchains are facing: **cost, scalability, confidentiality and governance**.

TODA's design, which relies on a novel, ***per-asset ledger***, is motivated by the desire to reduce work done by others to the absolute minimum. Instead of replicating every transaction into an enormous global ledger, TODA puts almost all the work on the transaction participants, while maintaining strong security guarantees.

In doing so, **TODA is able to dismiss many of the intrinsically limiting aspects of replicated ledgers**, including unmanageable computational effort per transaction, mining for Proof of Work, and other misaligned and wasteful incentives.

In this primer, we will discuss TODA's novel design in detail, with a focus on the key areas where it differs from blockchain architecture, and why it matters to your deployment. This primer is written in plain language, so you can gain a conceptual understanding of the technical considerations regardless of your level of expertise.

After reading, we believe you'll begin to see just how lean, flexible and natural the TODA protocol is, and where it can provide a competitive edge in your industry.

Why we wrote this primer

The goal of this primer is to provide a conceptual understanding of the architecture and technical foundations of both blockchains and the TODA protocol. It is written to be accessible to those with a non-technical background, and to help decision-makers gain an appreciation for the underlying concepts.

A deeper understanding of these concepts will help you:

- Understand the differences between emerging decentralized technologies
- Evaluate applications for specific use-cases
- Assess the value of a technology to a business case
- Develop an accurate expectation of the value and cost after deployment

A strong conceptual understanding tempers any unrealistic expectations caused by the hype and irrational exuberance that pervades this discussion space.

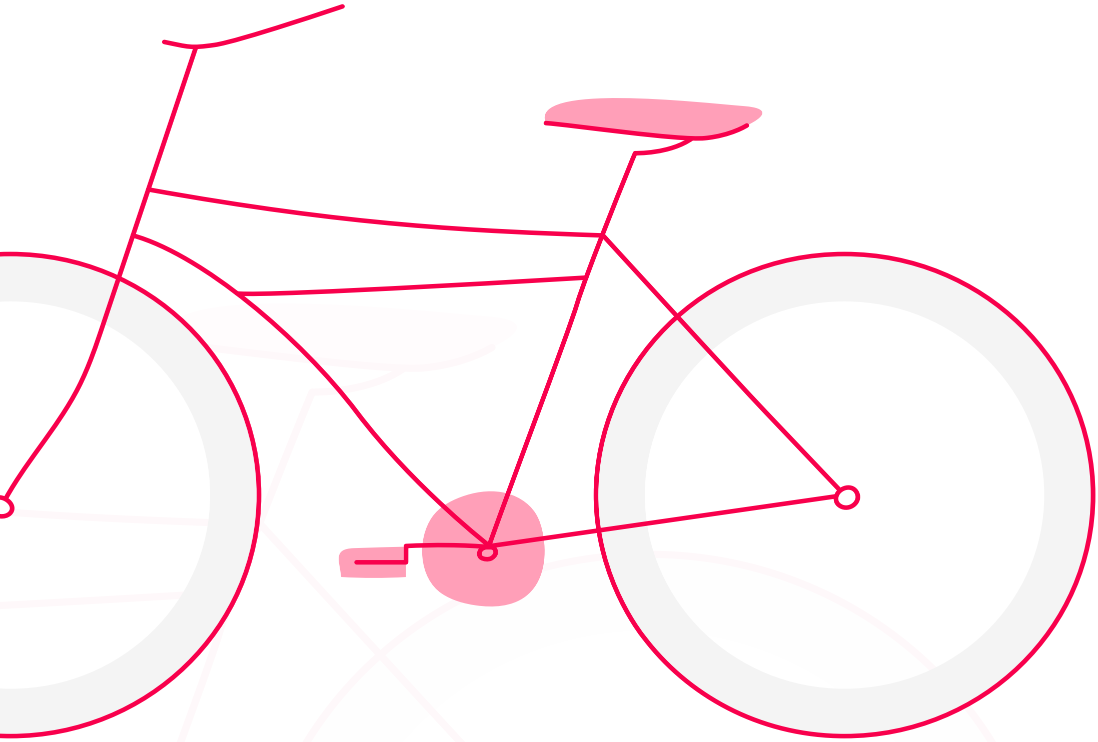
This primer is not a technical document. For more information about the mathematical foundations of TODA, please refer to the technical whitepaper.

This primer is also not a socioeconomic manifesto.
The mission of TODA is to provide a considered, fundamental protocol, engineered for practical, real-world use. Efficiency is our primary goal, not disruption or revolution.

A note regarding our descriptions of blockchain systems:
While we recognize that many different blockchain projects may have unique structures and features, for the sake of simplicity, our explanations of blockchain are deliberately agnostic, describing common structure and behaviours only in general terms.

Blockchain

To better understand the original intent of blockchain technology, we need to better understand the problems it aims to solve.



The problem of ownership

Let's say you had a red bicycle that was stolen outside your office. Later that day, you notice someone riding your bicycle! It must be the thief. You confront the rider, but he claims he had bought it years ago. How do you dispute this?

You might tell them to check the name tag sticker you affixed to the frame. But as you check the frame, you notice that it has been removed.

* Bicycle theft is actually a big problem. Only 5% of stolen bicycles are ever returned to their owner.

You could tell him about specific scratches or dents. You could even show a photograph of those features. Or even a photograph of the serial number engraved to the body. But he counters by accusing you of having observed or taken photographs of his bicycle when it was parked.

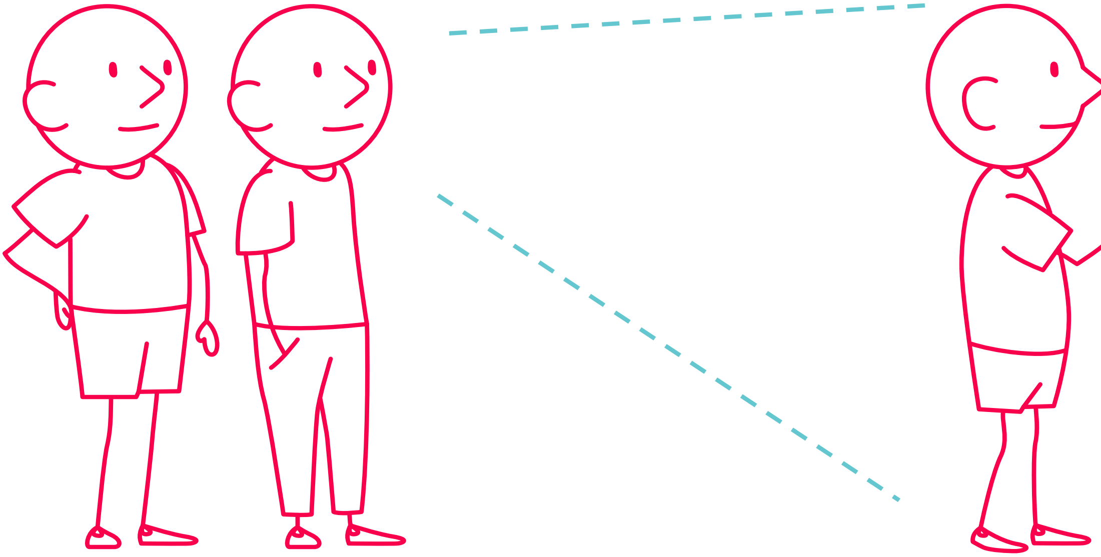
If you were careful, you might be able to produce a receipt from the store that you bought the bike. It might even have your bike's serial number written on it.

The thief counters again, accusing you of forging that receipt. In fact, if the thief is malicious and clever enough, he might have forged a receipt of his own for an earlier date.

How do you definitively prove something is yours? In this case, how do you prove to your peers or an authority that it is yours?

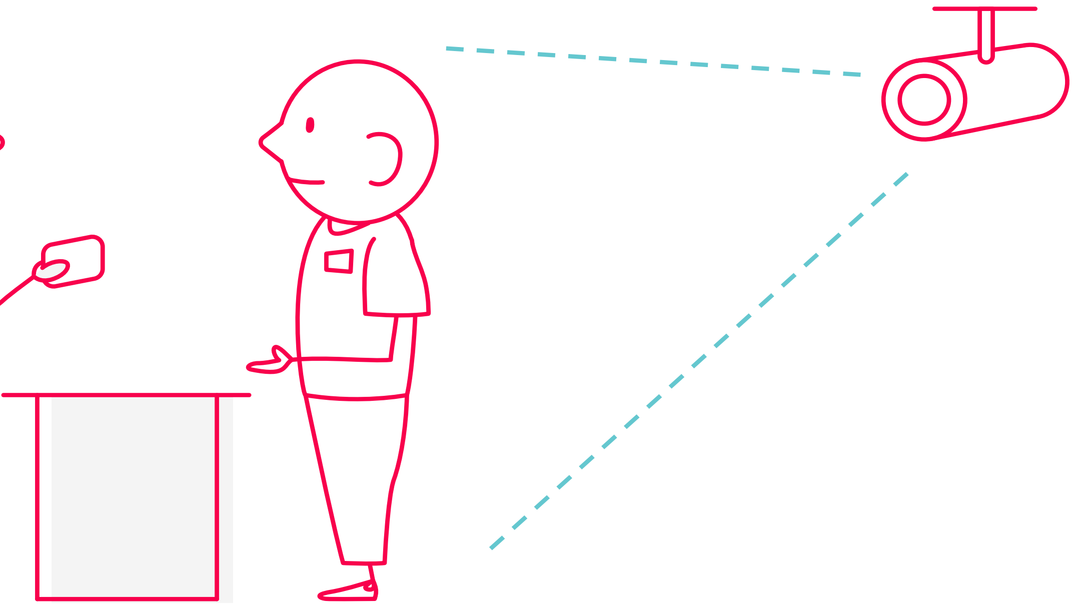
Ownership challenges in our culture involve convincing other people of your right to ownership, and your ability to do that is dependent on the strength of evidence you can provide.

Receipt
BOB'S BIKES
$239.23
Paid by card...
24 AUG 2018



The nature of evidence

So far, you and your bicycle thief have provided equivalent amount of evidence. What else can you provide?



Evidence of ownership needs to provide a minimum of the following:

- The identity of the owner (you)
- The identity of the object owned (this specific red bicycle)
- A convincing mapping of ownership from the owner to the object.

What if there was an independent witness at the store in which you purchased that bicycle? And this witness was able to recognize you and remember the make, model and serial number of the bicycle?

An independent witness could be the store manager, who had kept his own copy of the receipt. It could be a CCTV camera that captured your purchase on video. The more independent witnesses you can bring to bear, the stronger your claim to ownership.

Multiple independent witnesses strengthen your claim to ownership

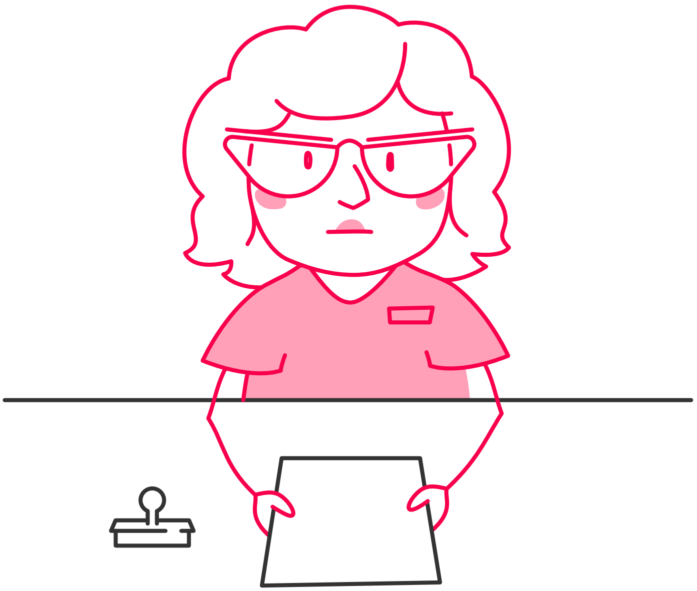
There is a fourth piece of information that is useful for refuting the thief's claim: proof that you bought the bicycle on a certain date. Perhaps you could note the date on the store manager's copy of the receipt. There might be a timestamp on the CCTV camera that witnessed your purchase.

Assuming you can prove that you purchased the bicycle on a certain date. If the thief claims to have gained ownership from you after this date, the burden of proof falls on him to prove his ownership.

What if the thief claims ownership of the bicycle on a date that precedes yours? That shouldn't be a problem if you can simply trace the history of the bicycle. You can ask the store you bought it from for proof of their ownership. The store manager might procure records that the store purchased the red bicycle from a local distributor. That distributor could verify that she purchased it directly from the manufacturer. And finally that manufacturer could prove that it created the specific bicycle.

In light of multiple independent witnesses and a complete verified history of the bicycle, it would be extremely difficult for the thief to dispute your claim to the police.

*** If you were really concerned, you might mail photocopies of your receipt to your friends and tell them not to break the envelope's seal when they receive it. The postmark suggests that the receipt has existed since the postmarked date. The sealed envelope suggests that the receipt has not been tampered with since that date.**



Modelling ownership

From the previous example, you can show people that you own something if you have enough testimony from independent witnesses that know enough of the history of that owned item.

Having enough evidence to prove this in the physical world is difficult enough, but there are sources of evidence that can help prove your ownership. Trusted third-parties may adjudicate your transaction. For example, when you buy a used car, the transaction is recorded in an automobile registry maintained by the government. This central registry, or ***ledger***, tracks ownership of all vehicles in its jurisdiction. A notary public is an official ***trusted individual*** who can serve as an impartial witness to signing important documents.

Proving digital ownership is much more difficult, since data of the transaction and data of the digital asset can be easily altered. Even if you register your ownership to a trusted third-party online, their registry may be damaged, corrupted, or destroyed. Proof of ownership with one source is risky.

However, what if your ownership was documented and broadcast to every computer in a network? That would provide the multiple independent witnesses needed. And proof of ownership would survive even if many of the computers were damaged.

To take it one step further, if all owners of a digital asset, past and present, are on this network, we can document and broadcast a complete transaction history of that digital asset, thus providing a complete proof of current and past ownership. Also, we can update this record of transactions whenever a transfer of ownership occurs.

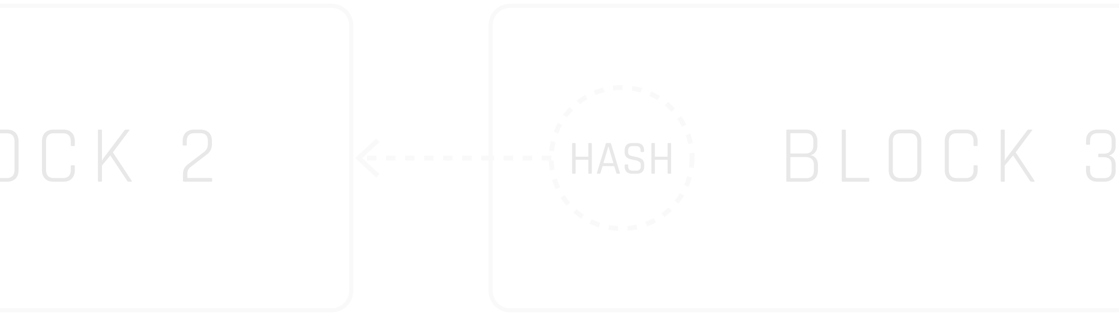
That is the fundamental design of all blockchain systems today.

Blockchains provide a robust record of ownership. They do so by distributing to all nodes on a network an **unalterable, complete record of transactions**. Like an insect trapped in amber, these records can be viewed but not altered. They are locked at the time of creation. When proof of ownership is required, anyone can simply look up the item history on the blockchain and verify the current owner.

When new transactions occur, every node must first agree that the transaction is valid *- is the person making the transaction the legitimate owner? -* before adding them to their individual copy of records.

Wait, what exactly is a blockchain?

We will discuss the blockchain structure in detail later, but at its core, it is a software protocol that empowers participants to verify and validate transactions between two parties in a permanent way without relying on a trusted third party.



Blockchains provide the necessary multiple "eye-witnesses" by distributing transaction information to a network of users. This transaction information is cryptographically secured to prevent anyone from changing any details later on.

Whenever new information needs to be added, it is put into a cryptographically sealed ***block*** and linked to a chain of previous blocks. This confirms that transactions were executed in a specific order, effectively timestamping all transactions.

In order to link the blocks together, every block in the chain contains the ***cryptographic fingerprint*** of the block before; the previous block's fingerprint is encoded into the present block's fingerprint.

For tamper resistance, the fingerprint (a ***cryptographic hash***) is specially calculated to contain an encoded version of all the stored data in the block. If anyone were to modify that stored data, it would alter the hash, which would in turn alter the hash of every subsequent block.

UNIQUE FINGERPRINT
OF THE DATA.

The linked nature of a blockchain makes it impossible for someone to modify data without everybody knowing that you did it.

If you can follow the chain unbroken back to the first block, you can be sure that the data has not been tampered with because of this design. The order of information is also unalterable, meaning you can form a timeline of things that have happened.

Fundamental Blockchain Guarantees

- Agreement on the current block implies agreement on all of history
- Any change in historical data changes the current block - and is therefore detectable
- Every block in history is locked into a specific order

What blockchain isn't

We can now see that blockchain systems revolve around records of transactions and verification of ownership. But as with many new technologies, blockchain is often confused with other concepts. It might be conflated with a narrow application. Or evangelists may overpromise what blockchains can deliver.

In either case, it is important to take a moment to discuss what blockchains are not, to better understand their value proposition.

Blockchain is not cryptocurrency

Blockchain is often confused with its most successful application: cryptocurrency. Why is this?

Blockchains offer an accurate, resilient way of tracking transactions and proving ownership that is hard to manipulate. It is most robust when all tokens of a certain type are completely tracked within the blockchain, so the easiest thing to track are digital tokens.

That's all cryptocurrencies are: digital tokens with alleged value, whose ownerships are verified using blockchain technology.

Many blockchains also incorporate cryptocurrency into their implementations, issuing cryptocurrency tokens as a means to compensate participation on the network - for users to be ***eye-witnesses*** for transactions.

With a lot of attention being directed to cryptocurrencies, **it's easy to forget that blockchains have many other valuable uses**.

Blockchain is not distributed computing

Certain blockchain implementations are coupled with a complete distributed computing infrastructure, and in some cases, with a Turing-complete programming language. These so-called ***"dApps"*** add significant weight and complexity to a transaction system.

"dApps" have been branded as a panacea for any number of distributed computing problems. As we'll discuss, dApps represent one extreme design choice among many for arriving at sophisticated transactions.

The reality is that the blockchain architecture grossly underperforms centralised counterparts in cost and latency. For many applications, the tradeoff is not worth it. For most people, they would prefer centralised for the speed they've come to expect. Corporations or institutions trying to convert users over to dApps will find it a very costly process an experience difficulty in gaining traction.

Blockchain is not a complete service solution

Blockchains are underlying processes that hide beneath the application layer. If you try to use them as a silver bullet, you'll soon discover blockchain's shortcomings.

Blockchain protocols are not intended to be standalone applications. Cash is free-er. Paypal and Mastercard provide value-added services and mediate fraud disputes. Banks verify identity of buyer and seller. Some institutions offer deposit guarantees.

With blockchain, you're a bit on your own.

While blockchain can be a part of an overall system design, it solves very few problems when standing alone.

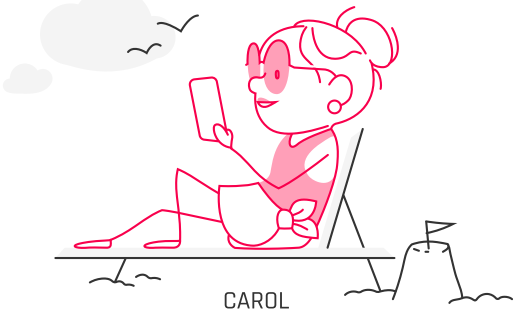
Approaches to Blockchain

Replicated Ledgers

A ledger is a collection of entries, often concerning financial accounts. The terms database and ledger are sometimes used interchangeably. A replicated ledger has many copies spread around, which all must be updated at roughly the same time and in exactly the same way. Having many copies provides some advantages over having a single copy or a few copies, but it carries costs as well.

TIME	TO/FROM	BALANCE
14:00	Alice	-42
14:00	Bob	+42
14:05	Bob	+3000
14:05	Carol	-3000
...	...	...

TIME	TO/FROM	BALANCE
14:00	Alice	-42
14:00	Bob	+42
14:05	Bob	+3000
14:05	Carol	-3000
...	...	...



TIME	TO/FROM	BALANCE
14:00	Alice	-42
14:00	Bob	+42
14:05	Bob	+3000
14:05	Carol	-3000
...	...	...

What replicated ledger blockchains do **well**

AVAILABILITY

Everyone has a complete copy of everything which has occurred. If there is a major network disruption, there is no difficulty finding others who can independently verify your ownership of an asset.

OPENNESS

Anyone can submit a transaction or attempt to build the next block. There is no requirement to register. Transactions and blocks are submitted pseudo-anonymously.

CENSORSHIP-RESISTANCE

Because anyone can submit transactions for inclusion in a block, it is difficult to intentionally prohibit a specific node from successfully including their transactions.

PSEUDO-ANONYMITY

While transactions are available to all to inspect, they don't directly disclose identity. Users on the network are identified instead by their cryptographic addresses, which might be difficult to link to a particular person.

RESILIENCE

Because everyone has a full copy of the same replicated ledger, it protects owners if a few nodes fail.

What replicated ledger blockchains do **poorly**

COST

Every item added to a replicated ledger requires network, computation, and storage, because the item must be sent to everyone, they must all confirm its validity, and then they must all store it forever.

THROUGHPUT

In addition to the necessary costs of replicated ledgers there is also a consensus cost. For proof-of-work based ledgers, that cost is quite high. It also severely limits how many items can be put into the ledger in a single day.

LATENCY

To increase the number of items they can transact in a single day, ledger-based systems have invented experimental extensions, such as **DAG ledgers** or **hashgraph ledgers**. Unfortunately, this increase in throughput comes at the cost of higher latencies.

PRIVACY

The great thing about replicated ledgers is that information put there is there forever, for everyone to see. Including everything you buy using it, and all of your other financial transactions. This is a boon for datamongers who want to sell you ads, but clearly undesirable for most people.

CENTRALIZATION

Three major conglomerates have taken over the majority of Bitcoin mining. Five for Ethereum. Picking a lunch spot for your office probably involves consensus from more individuals than that.

CONSISTENCY

When you have high availability, but encounter a network partition, it's possible that all of your "transactions" are, in fact, undone when the network is re-established. Not knowing that the network may be in this state leads to uncertainty around the completeness of a transaction.

How much do ledgers really cost to use?

That depends on how many machines are on the network. The amount of replication your data has, and the amount of cost for putting something on the ledger, are controlled by economic factors that have nothing to do with your transaction. There is no way for you to control it, or to accept a lower replication factor in exchange for paying less.

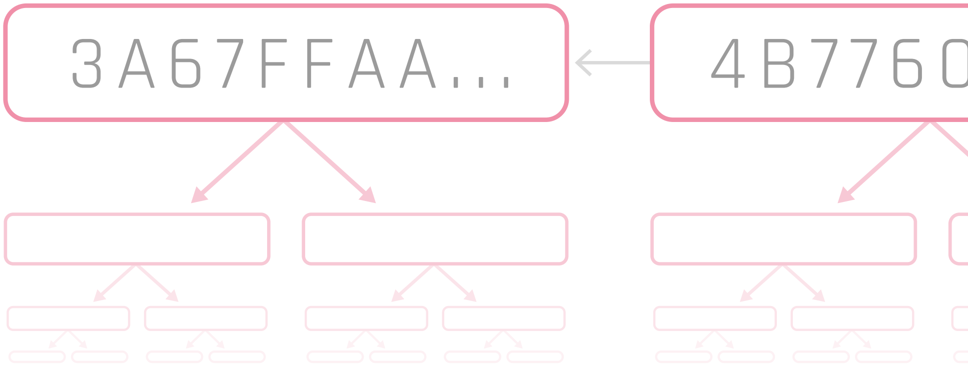
Why TODA is ledgerless

The easiest way to tackle the ownership problem is by giving everyone a copy of the ledger. This makes it trivial to verify anyone's transactions, but comes at the expense of both efficiency and privacy.

Recall that each block has a unique fingerprint, made by combining the block's information with the fingerprint of the previous block.

Every detail of the block contributes to its fingerprint, as well as all subsequent fingerprints. The most recent fingerprint contains within it every action that has ever occured. It is the holographic projection of the totality of spacetime. It anchors reality.

So instead of referencing shared information across a replicated ledger, a transaction can carry its own proof of validity, simply by showing that it is part of that fingerprint. This proof is small enough to pack into the transaction itself: the data carries its own proof.



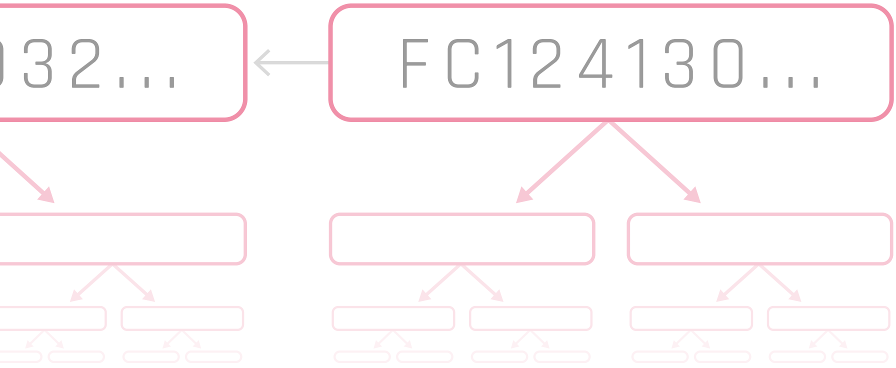
Each block's fingerprint is a 32-byte representation of reality

A fingerprint of reality.

We don't need pristine, flawless ledgers to prove ownership. You don't need to be an accountant to know when something is wrong.

It is a common trope in science fiction. Someone is thrown back in time and inadvertently changes something small. She then goes back to her present and finds everything quite different. The timeline has forked because of the little change she made. She has to then go back in time to fix her mistake.

Our timeline is a series of blocks. Each block's fingerprint is a 32-byte representation of reality. Any little change we make to the transaction history ripples up the blockchain to the present day.



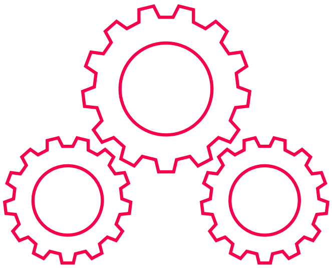
So we just need to prove that an item’s individual history contributed to the block. We need that item’s history and just enough knowledge of the environment surrounding that item to prove it.

Instead of having every node on the network be a neurotic obsessive accountant who verifies every transaction, you just need enough people who have experienced reality. **In TODA, it’s immediately obvious if a transaction fails to fit into your current picture of the world.**

We don’t need to know detailed histories of every object in the world. Just enough to know what you changed. This philosophy is more elegant and strips out a lot of the complex solutions found in other blockchain projects.

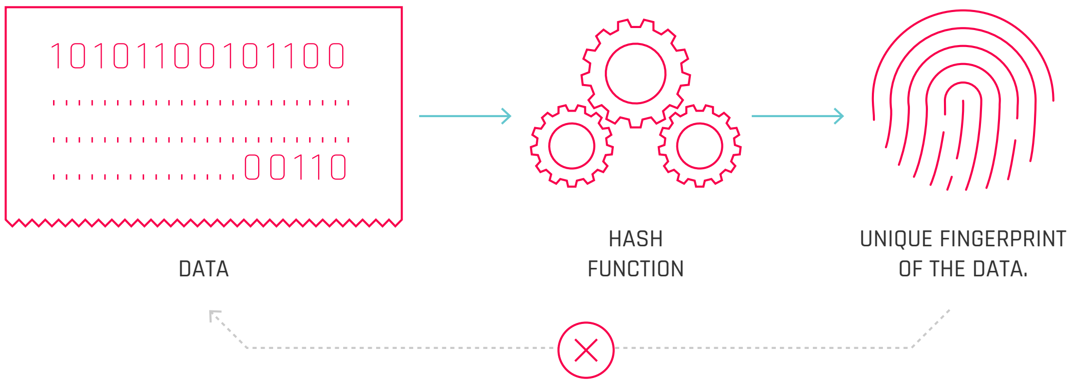
Replicated Ledgers

How do replicated ledger blockchains work, exactly?



Hash functions rehashed

A ***hash function*** is a mathematical operation which creates a representation of some data. It serves the role of a ***fingerprint***. A very simple hash function might count how many bytes are in the supplied data, and return that number. However, blockchains use a specific kind of hash function: a ***cryptographic hash function***.



CRYPTOGRAPHIC HASHES **ARE ONE WAY ONLY**.
YOU CANNOT FIND DATA THAT MATCHES THE FINGERPRINTS.

Goals of **Cryptographic** Hash Functions

- **Diffusion**: Changing any part of the data, even slightly, should change the result of the hash
- **Confusion**: Given the output of a hash function, it should be extremely difficult to find data which will result in a matching hash

One way to think of hash functions is that a hash should be **easy to verify**, and very **difficult to forge**.

* Claude Shannon, known as the "father of Information Theory", identified these properties in a classified 1945 report

Public and Private Keys

Public-private keys are a technology used in many different cryptosystems. Like hash functions, they are useful in applications far outside of blockchain systems.

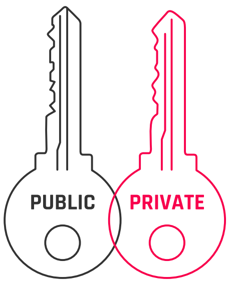
A ***key pair*** is commonly used to refer to two related keys; one ***public*** and one ***private***.

Public-private keys are used for two different cryptographic roles. They can be used to ***encrypt*** data, so that it can only be ***decrypted*** with the right key, and they can also be used to sign data. Blockchains most commonly make use of public key ***signatures***, so let's investigate that capability.

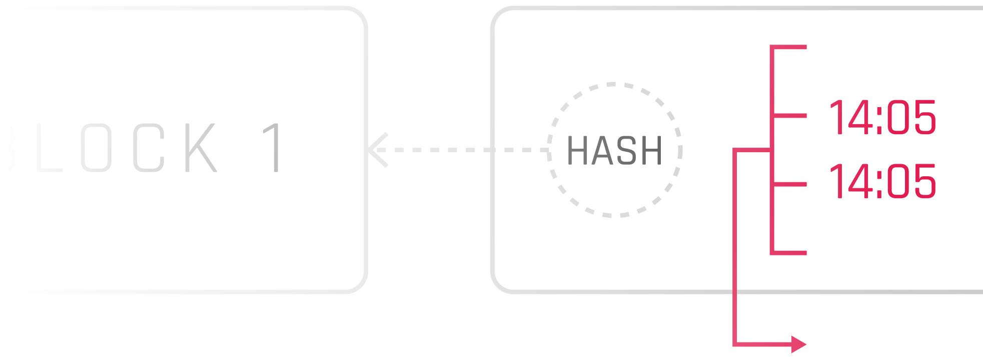
A ***private key*** is the component of a key pair which the owner keeps secret, to themselves, and under their control. When used to sign data, the private key acts to create the unforgeable stamp, proving that whoever holds the private key really intended to sign the data.

But since we never want to reveal our private key to anyone (since this would let them sign anything on our behalf), we can reveal our ***public key***. By comparing the signature on some data with a given public key, cryptography allows us to prove whether the corresponding private key was really used to sign the data.

Public key signatures are an important building block in blockchain systems, because they enable ***authenticity***. This mechanism of signing allows anyone to verify whether a given user legitimately signed a message or not.



A PUBLIC-PRIVATE **KEY PAIR**

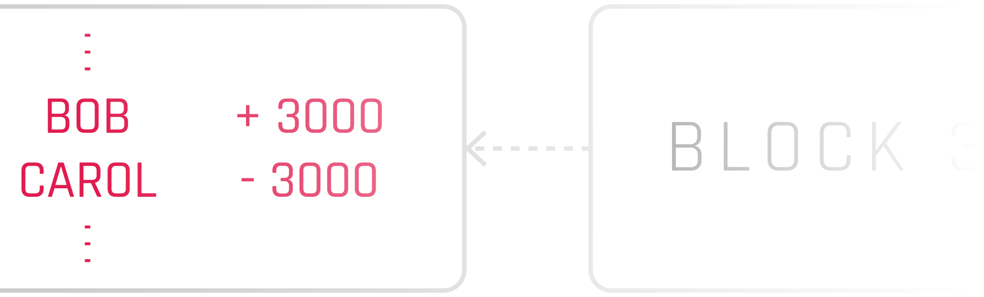


Blockchain data structure

The fundamental concept of a ***blockchain*** is actually very simple. In a blockchain, data arrives in chunks (***blocks***), one after another. The 'chaining' occurs because each block is linked to the previous block by incorporating the **hash of the previous block** into the current block. This means that each block of data references the hash of the previous block of data.

Why is this useful? In most blockchain systems, the ***blocks*** contain a list of transactions which occured in that block.

Suppose someone shows you a block, full of transactions they say occurred. If you want to verify whether those transactions really did happen, it's quite straightforward: they need to show you where in the chain that block fits in. You'll need all the blocks in-between the current block, and whenever in the past the supposed block occurred. If they're telling the truth, you should be able to hash everything together to arrive at the current block.



Proof of Work and Hash Puzzles

Private keys prevent Alice from sending funds from Bob's account, but they don't prevent her from sending both Bob and Charlie the same funds from her own account. This is called **the double spend problem**: Alice could send two different transactions, one to Bob and one to Charlie, spending the same funds twice.

To prevent this, transactions are not added directly to blocks in the blockchain. Instead, they are sent into a waiting pool to cool their heels. Later, someone might pick a few transactions from the pool, wedge them into a block, and try to fit it onto the chain.

It turns out that getting to pick which transactions go into a block gives you quite a bit of power, especially if you get to pick many of them in a row. So we need a way to choose who gets to pick the next block. The original Bitcoin whitepaper suggests the each person should get a vote, but since it's hard to tell who is a person and who is a robot, we can instead proxy the vote to your computer and have "**one vote per CPU**".

It turns out that getting to pick which transactions go into a block gives you quite a bit of power

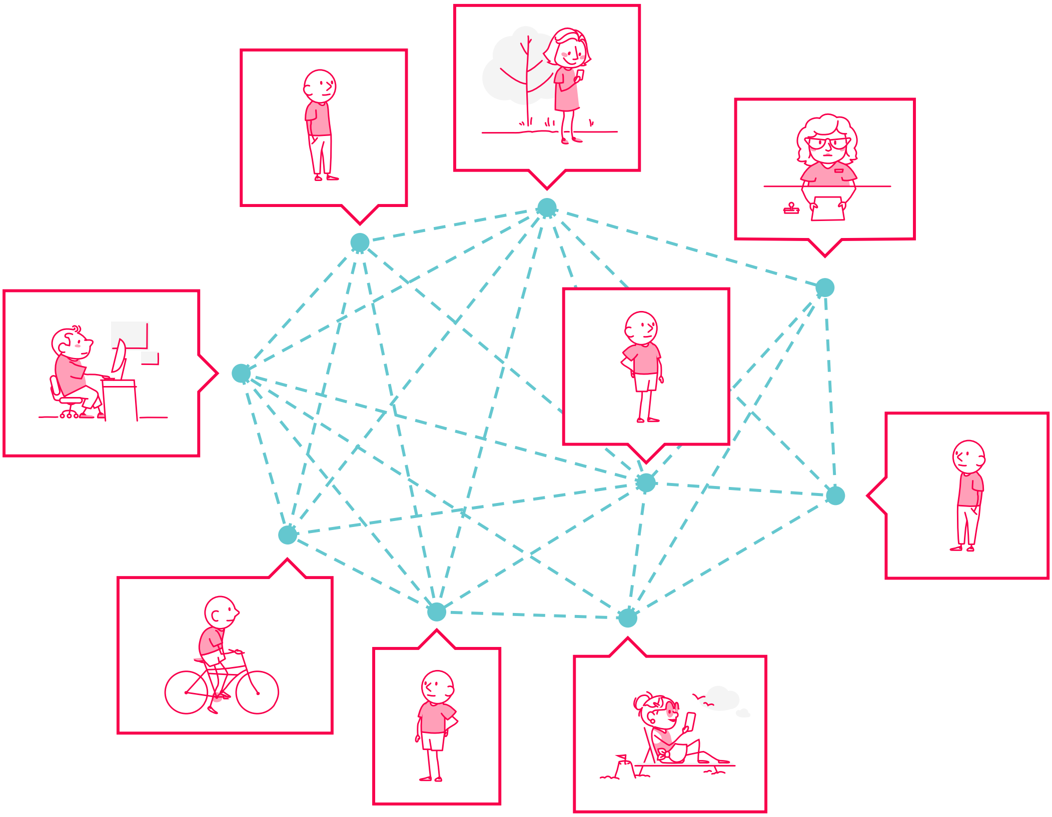
We could do this by having each computer try to find a needle-in-a-haystack, and the first one to find the needle gets to choose the block. In this case the needle is a random number added to the data of the current block that causes its hash to be below a certain value, and the rest of the haystack is all the random numbers that cause its hash to be above that value.

In 2008, when the Bitcoin whitepaper was first published, using CPUs as a proxy for humanity was a clever trick to sidestep the complex problem of digital identity. With hindsight, though, the weakness in this design is clear. It is this weakness that was exploited by the miners to steal all the votes of Bitcoin users and concentrate them in the hands of a few elites. It is this weakness which is now causing vast amounts of power to be wasted on fruitless speculation.

Incentives

Because finding a needle in a haystack is hard work, the computers doing that work need to be compensated. As long as Bitcoin's value is continually increasing, there will always be machines willing to trudge off to the Bitcoin mine to try to dredge up some new ones. The value of Bitcoin needs to constantly increase to make this worthwhile, because other miners are there as well, and in general everyone pays as much in electricity as they expect the Bitcoins to be worth today. The hope is that they'll be worth more tomorrow, allowing miners to pay off the money they borrowed to buy their expensive mining machines.

*** The rationale for taking on this large financial risk and burning up enormous amounts of electricity is called "game theory", which says no matter how shaky your theory, there will always be someone game to try it. (It also explains the existence of pyramid schemes and unlicensed betting parlours.)**



Network communication

Each computer in the peer-to-peer system shares information like humans.

In typical replicated ledger blockchain systems, every peer receives all transaction data, all hash reference values, and all block headers. Essentially, every node needs to be supplied with enough information to be able to re-create the entire chain. This is how ***blockchain explorers*** are able to let you browse through the history of major blockchains.

* Did you know that you can actually browse many replicated ledger systems on the web? Various **blockchain explorers** let you actually search through the entire history of popular chains. These demonstrate the limited degree of privacy such systems really have.

How do peers receive this information about the blockchain? A popular solution is through a ***broadcast protocol***, where peers broadcast all of the transactions they know about to the peers they are connected to. As we'll see, this solution provides resilience and fault-tolerance, but can lead to a significant cost in both network traffic and transaction latency.

Common problems with replicated ledger architecture

Privacy

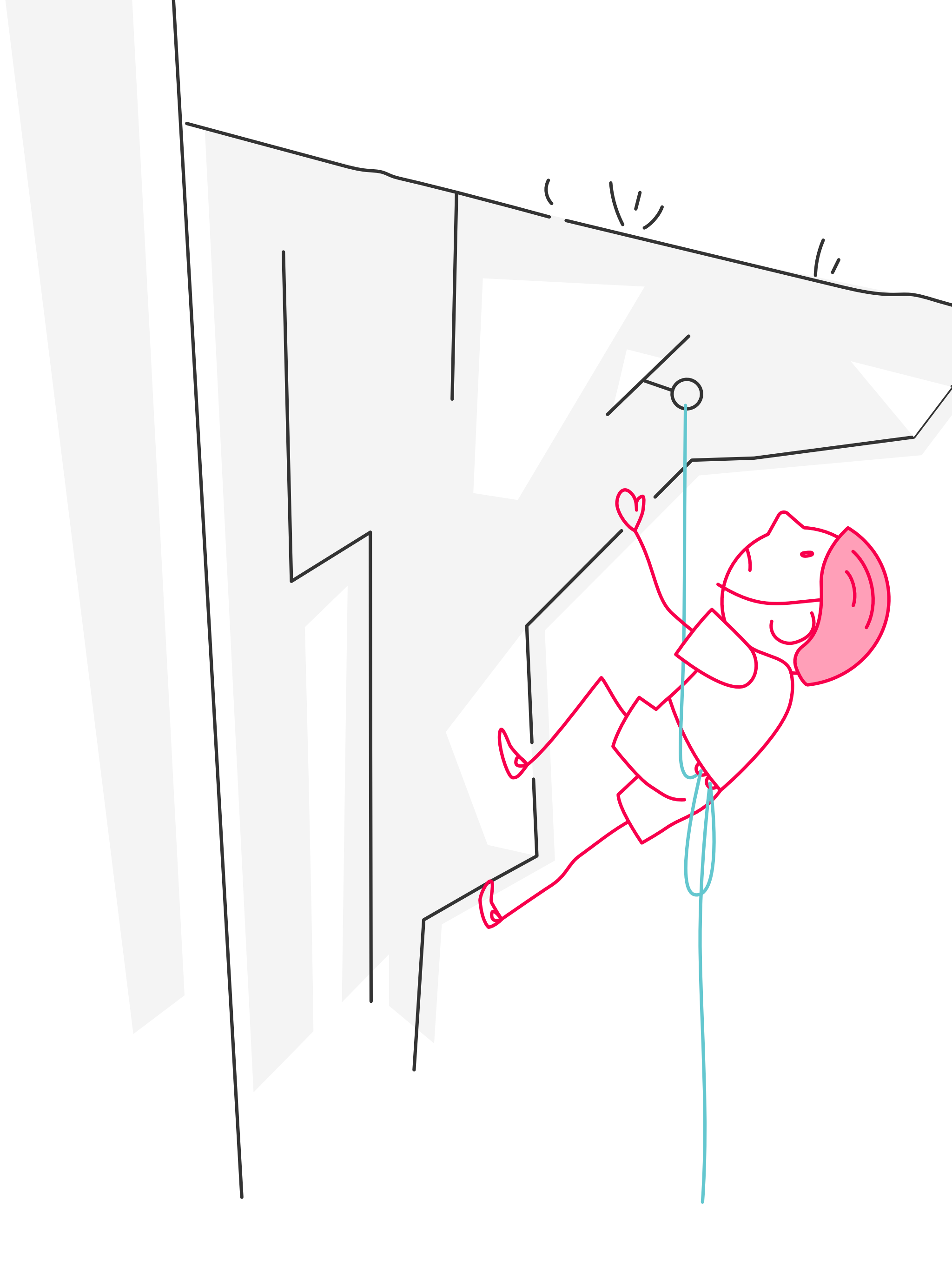
In replicated ledger systems, everyone needs to know about every block - always. This does not allow for any transactions to occur privately; they are all broadcast for the world to see. While you might not attach your name to a transaction, **the overall flow of value can usually be traced**.

Intrinsic centrality

The Proof of Work mechanism in replicated ledgers has a tendency to centralize most of the responsibility (and power) of a blockchain to those few who control substantial computational power. The existence of these participants (***miners***) actually serve to re-centralize replicated ledger networks, and reintroduce many of the drawbacks of centralization.

Risks of Centralized Systems

- **Political risks**: Localized infrastructure in unstable countries; decision making returns to the hands of a few
- **System instability**: Centralized systems have single points of failure
- **Impaired scalability**: The growth of the system is limited to the capacity provided by central nodes



Overhead and Scalability

Typical blockchain implementations suffer from difficulty scaling: they get worse as more people start to use the system. Why do systems get worse as they become more popular?

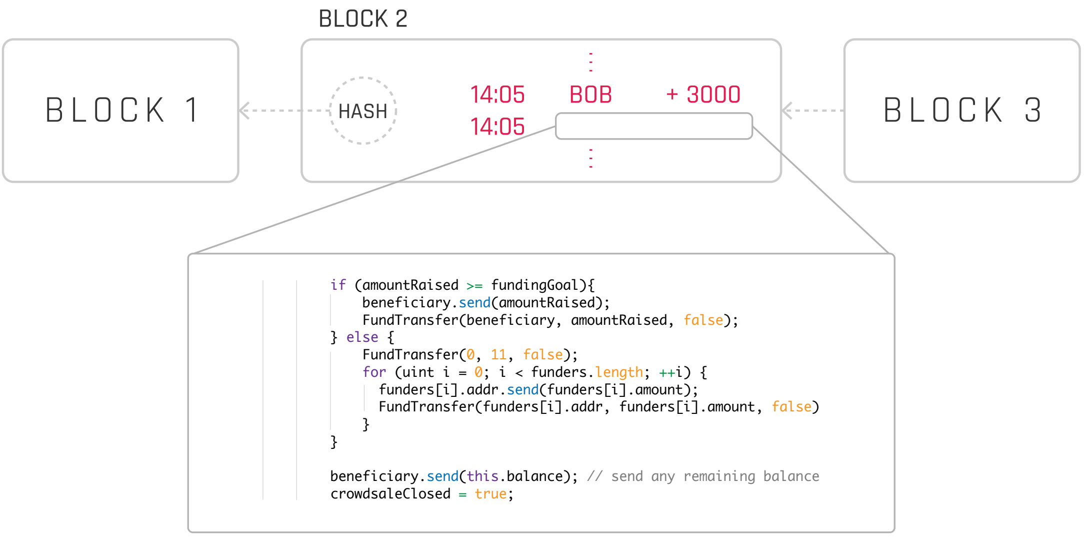
Most replicated ledger blockchains suffer from usage in two ways: they require **more computational power** (and therefore electricity), and they get **slower**. These properties are undesirable in an ideal transaction system.

How does this occur? There are several components of replicated ledgers which conspire to produce these less-than-ideal traits:

- **Coordination overhead** - Coordinating work in any distributed system requires additional costs and computing power

- **Communications overhead** - Sending, receiving, and processing messages requires effort and computing power. The extent of the communications overhead is dependent on what type of distributed network is used. Most replicated ledger systems use a gossip protocol for communication. These protocols, while resilient, are known for their considerable communications overhead.

- **All for one and one for all** - Replicated ledgers require that every fully participating node must process every single transaction which occurs. This means that as more transactions occur, and more nodes participate, overall transaction times suffer severely. This type of protocol offers security, neutrality and censorship resistance at the cost of scalability.



```
if (amountRaised >= fundingGoal){
    beneficiary.send(amountRaised);
    FundTransfer(beneficiary, amountRaised, false);
} else {
    FundTransfer(0, 11, false);
    for (uint i = 0; i < funders.length; ++i) {
      funders[i].addr.send(funders[i].amount);
      FundTransfer(funders[i].addr, funders[i].amount, false)
    }
}

beneficiary.send(this.balance); // send any remaining balance
crowdsaleClosed = true;
```

The problem with decentralized applications

Decentralized applications ("dApps") are a remarkable technical feat. This technology enables an entire computer program to be run on the blockchain itself. That is, "dApps" as defined by the marketing department of at least one major blockchain initiative, are not simply applications which take advantage of blockchain mechanisms, or use a blockchain API. Instead, **"dApps" are basically extraordinarily large and complex transactions which must be run on-chain**.

Why would someone go to the difficulty of designing such a system? Proponents of "dApps" suggest that coding entire programs as sophisticated transactions allows for an entirely new level of ***smart contract*** - one that can have no limit on complexity or computational requirements. Suggested applications include distributed voting systems, and financial funds without anyone at the helm.

The real problem with "dApps" is their complexity. **Complexity in distributed systems is expensive.** And, in the case of replicated ledger blockchains, this complexity is realized in both the extreme expense of supporting such close coordination, as well as in the computational power needed to actually run the program logic.

* Computer scientists call a programming system **Turing complete** when it can support basically any program, without limits. Turing completeness is wonderful if you're creating a programming language – you usually want to let people express any kind of logic imaginable. However, you don't necessarily want your asset ownership system to be Turing complete. Remember, Turing complete logic could take a very long time to run - there are no limits - which could lead to a very expensive procedure to determine the owner of an asset.

Building decentralized application without “dApps”

“dApps” represent an extreme approach to the use of decentralized technology. There are actually a range of options better able to accomodate common industry requirements, such as:

- **Participating as a node**: an application running on a mobile device or desktop computer can participate as a node. Then it is capable of initiating transactions, and taking action in response to incoming transactions.

- **Subscribing to an API**: an application running on a mobile device or desktop computer can query a decentralized system through a managed service. This lowers the cost of application development even further by exposing data through a familiar interface.

In these scenarios, the application itself is not “on-chain”, but it can take full advantage of the data integrity and other protections offered by decentralized systems.

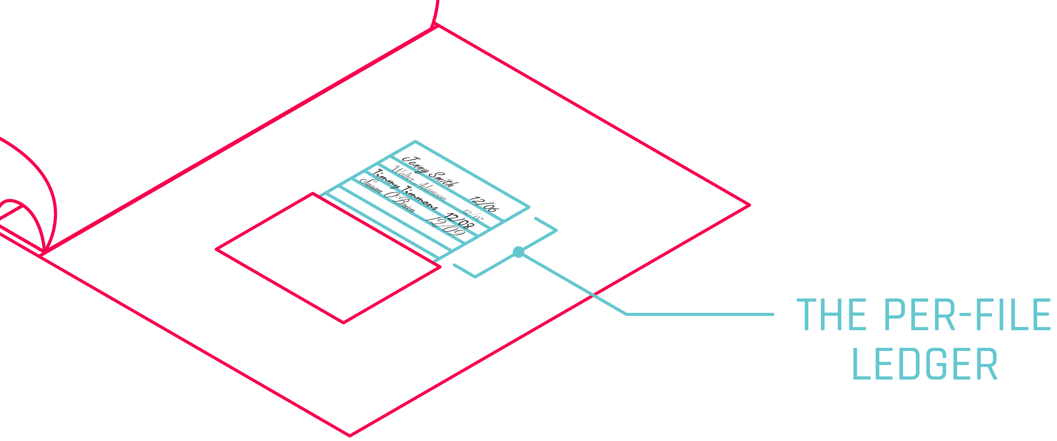
* We recommend TODAQ’s Toda-as-a-Service API!

The TODA Protocol

What's a TODA File?

A ***TODA File*** is a lot like a library book. It has some information printed in it that can't be changed. We call this the ***payload*** of the file.

Until recently, library books had a feature in the back cover called a 'loan card'. The loan card had a new entry added each time the book was borrowed, providing an up-to-date record of the book's borrowing history. As we'll see, TODA files are a lot like these library books, and TODA's unique ***per-file ledger*** works just like the card in the back of the book.



So what can you put inside a TODA file? Well, the payload - just like a book - can be anything you like. A TODA file can contain any digital information whatsoever: a treatise on liberty, a collection of sunset photographs, a movie ticket, or even full-motion video.

Just like a book, **you don't need to ask permission to publish a TODA file**. There's no central registry – you don't even need to tell anyone you've created the file. This also means there's no limit to how many files you create. The cost of creating millions of files is on you alone, just like if you chose to print every article on Wikipedia and store them in your basement.

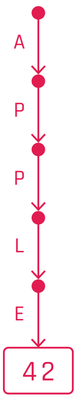
What makes a TODA file special? **The *per-file* ledger transforms TODA files from mere data to real digital assets.** This binding between a file's information its ledger imbues the file with qualities more akin to physical things than to information-based digital things.

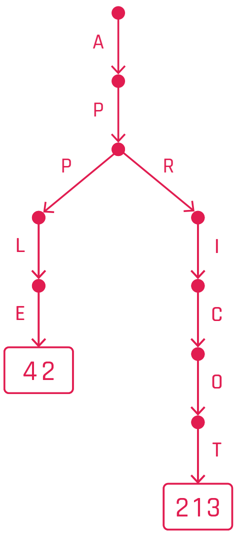
Physical things have inherent uniqueness. The words in our library book can be copied, but two books can't share the same loan card. TODA files have the same quality, which extends beyond perventing double spending to true distributed uniqueness.

TODA seeks to emulate many of the other desireable properties of physical things – like the ability to give a book to someone without having to ask a third party for permission, or having to pay someone to do it for you.

How does our loan card, the per-file ledger for each file, actually serve to create a blockchain? **How can we be certain that the transactions listed in the ledger really occurred?** And how can we know the list is complete and not missing any entries?

In order to establish these fundamental properties, we need to describe a few of the cryptographic data





Merkle Tries

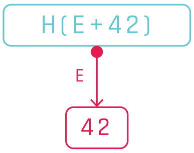
A ***Merkle trie*** is a cryptographic data structure. That means it's a way of storing data which has important cryptographic properties.

A trie is a data structure where every ***value*** you store gets assigned a ***path***. For instance, in the following diagrams, we first stored the value **42** at path **"apple"**, and then we stored **213** at path **"apricot"**.

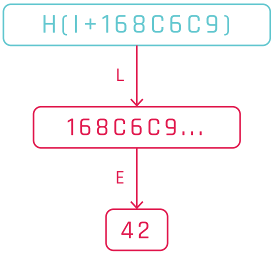
* **Merkle trees are named after their inventor, Ralph Merkle. He developed the concept in 1979 after several other major contributions to cryptography.**

Tries are an efficient way of storing data if you need to very quickly see what's at a certain path. They also have a very helpful property that TODA makes use of: **you can only store one value for a given path**. For instance, there is nowhere in the trie I could assign **"apple"** a different value such as **32**. Additionally, it's very easy to see whether a path has any value at all. For example, if I wanted to determine the value of **"blackberry"**, I could check the trie to see it has no specified value at all.

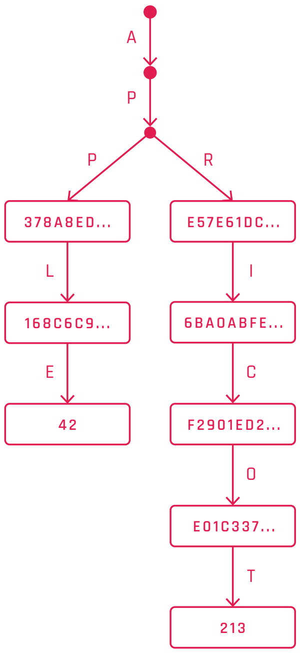
How does a ***trie*** become a ***Merkle trie***? This is where the data structure is endowed with some additional cryptographic information. At each level, we assign a value to each node, which is a hash of the values of its child nodes, combined with their path prefixes.



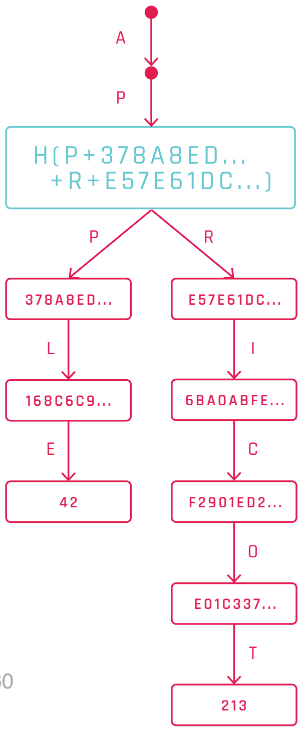
For instance, the prefix ***'e'*** combines with **42**, and is hashed to form the value of its parent.

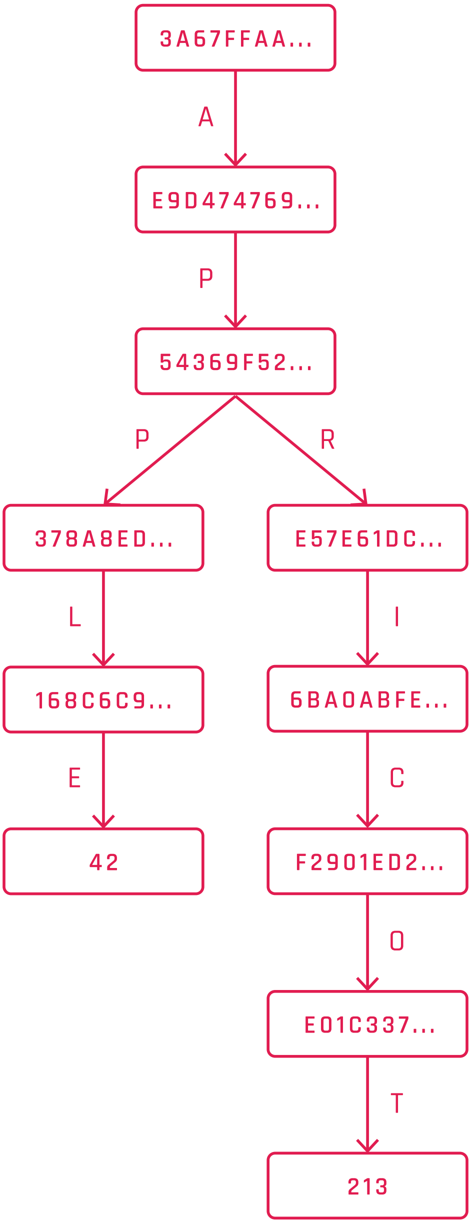


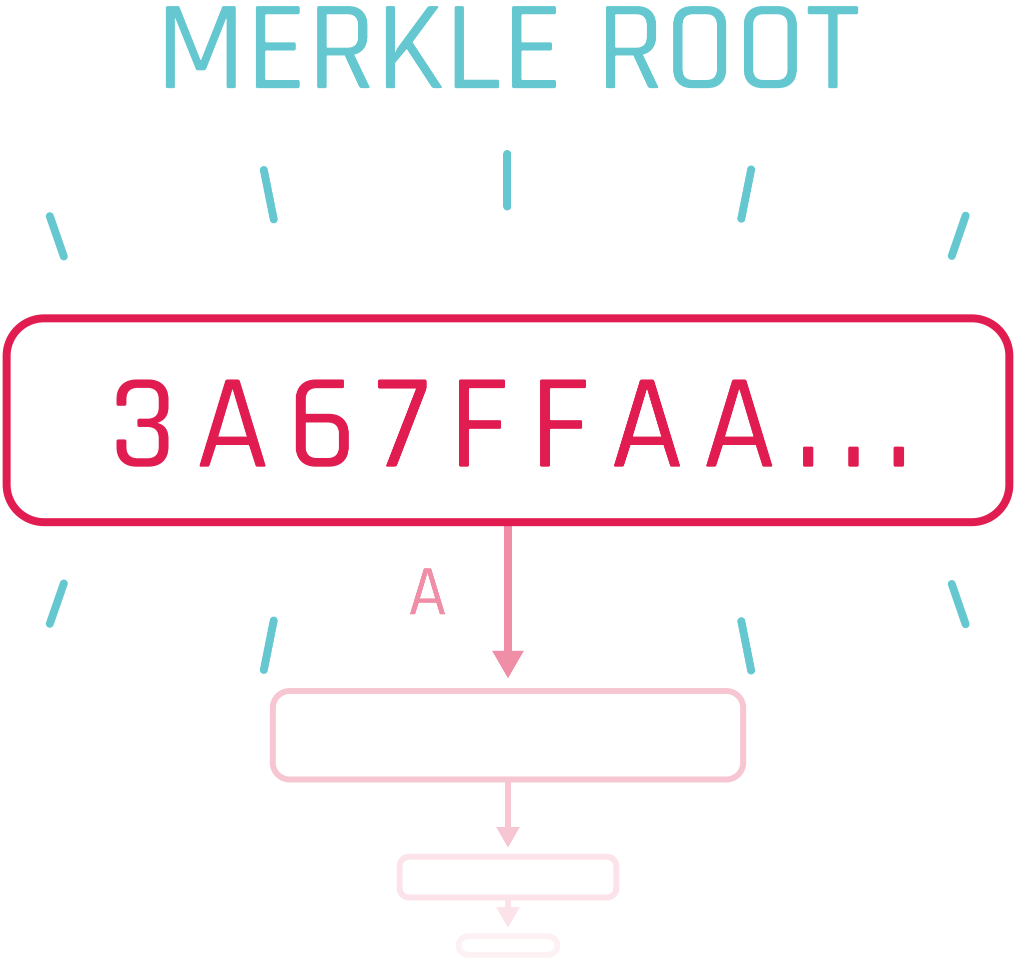
This continues, recursively, up the trie.



At every point in the trie where a node has multiple children, the parent value is the ***combined hash*** of all their children, with their prefix values.




3A67FFAA...
A
E9D474769...
P
54369F52...
P
R
378A8ED...
E57E61DC...
L
I
168C6C9...
6BA0ABFE...
E
C
42
F2901ED2...
O
E01C337...
T
213

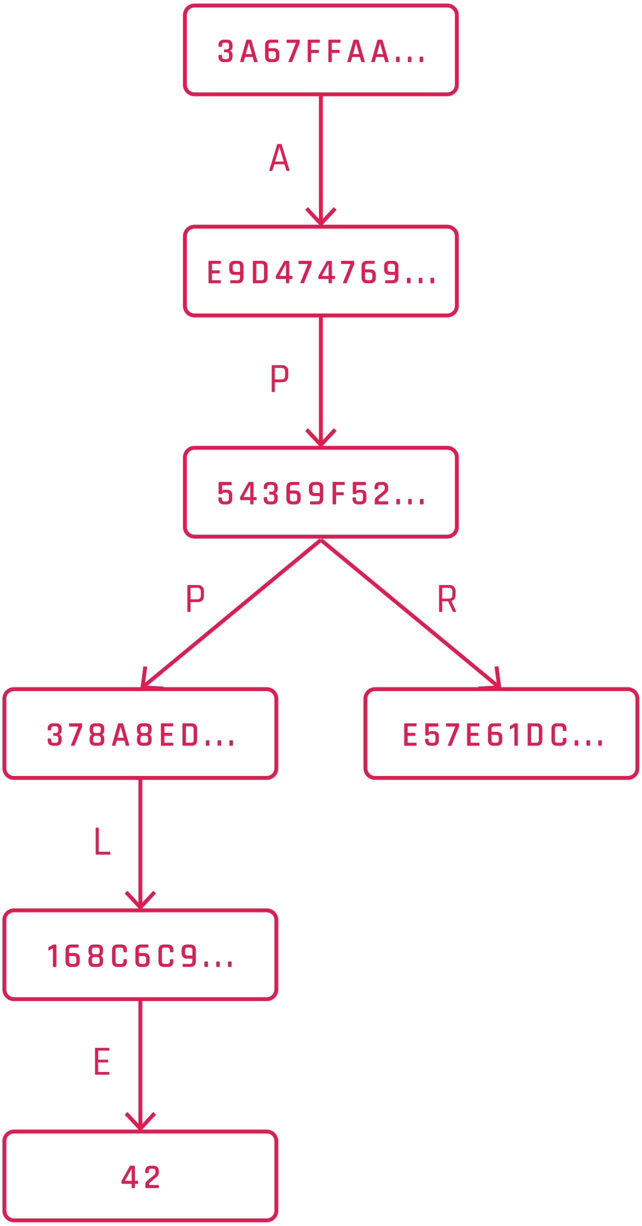

MERKLE ROOT
3A67FFAA...
A

At the top of the trie (the ***root***), we have one resulting hash value. This gives a Merkle trie some very important cryptographic properties.

Properties of Merkle Roots

- **Integrity**: If the value of anything in the trie changes, the value of the root hash also changes. If we can agree on what the root hash value is, we can be confident nothing else has changed.

- **Proof of Contribution**: Merkle roots can be used to prove whether a value was a part of the trie. For instance, if I know the root value for a Merkle trie, you could prove to me that **"apple"** had a value of **42** in that trie. In order to do this, you could provide the other parts of the trie, and show that when everything was properly calculated, it matched the same root value.


3A67FFAA...
A
E9D474769...
P
54369F52...
P
R
378A8ED...
E57E61DC...
L
168C6C9...
E
42

Trimmed tries

In TODA, we frequently encounter Merkle tries with billions of values. We would like to be able to prove that a certain path and value is in a Merkle trie, but it would be very inefficient to provide the several terabytes of information which comprise the rest of the trie.

Luckily, Merkle tries have an important feature which allow us to prove that a value is in the trie without having to send much data. We can take the same Merkle trie as before, and send over just the nodes along that path and their immediate neighbours. **This tiny piece of the trie is all that is needed to prove the value of that path.**

Because of the way a Merkle trie is defined, we are able to provide this trimmed Merkle proof instead of the entire trie. As we'll see, this ability is a critical building block when building an efficient, scalable digital asset management system.

The TODA Proof Concept

How does TODA use Merkle tries in a decentralized way? And how does this relate to our per-file ledger as a library book loan card analogy?

TODA uses Merkle tries in order to prove that each entry in the per-file ledgers were actually included in a large, distributed Merkle trie, whose ***global Merkle root*** is calculated and shared between network participants.

In the following sections, we'll see how someone is able to prove that they actually added a row to the ledger of a file in a ***cycle***. The proof that the entire ledger of a file is legitimate is a collection of the proofs that each entry is legitimate, along with additional data proving that the ledger isn't missing any rows.

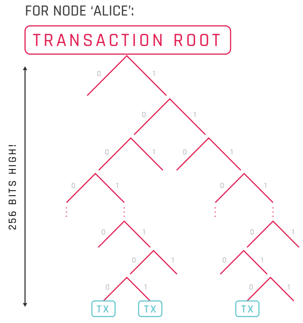
TODA proofs effectively rely on two different tries. The first, the ***transaction trie***, is built by any participant who wishes to transact files. It represents the **total change** they wish to make during a given cycle.

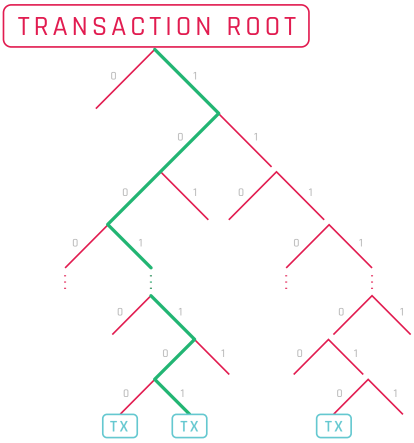
The second is the ***cycle trie***. This is built collectively by the entire network, using the calculated Merkle root from each transaction trie. We will describe each of these in more detail in turn.

* **Cycle** refers to the time between global Merkle roots. During each cycle, network participants have the opportunity to initiate transactions and submit them for inclusion in the next global Merkle root.

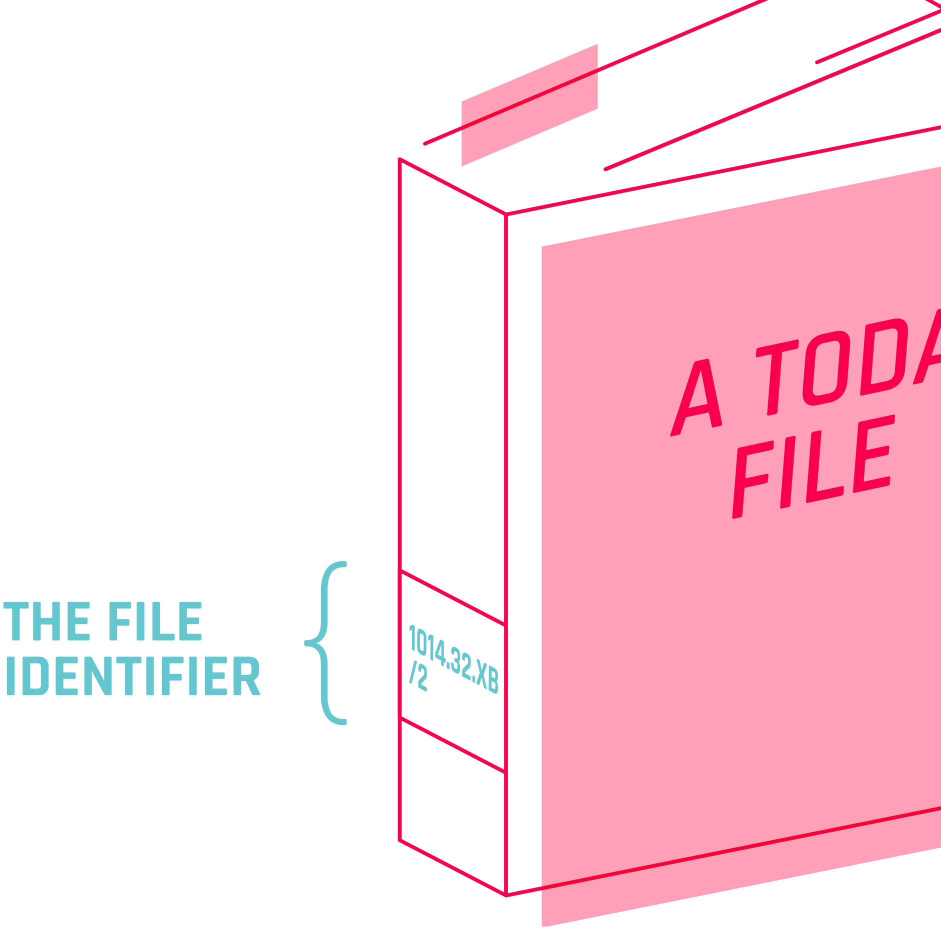
The Transaction Trie

The ***transaction trie*** describes the entire transaction a TODA participant intends to make during a cycle. In this trie, the ***file identifier*** of a file is used as the path, and the value at that path describes the transaction which that file is participating in. We call this value the ***transaction detail***.



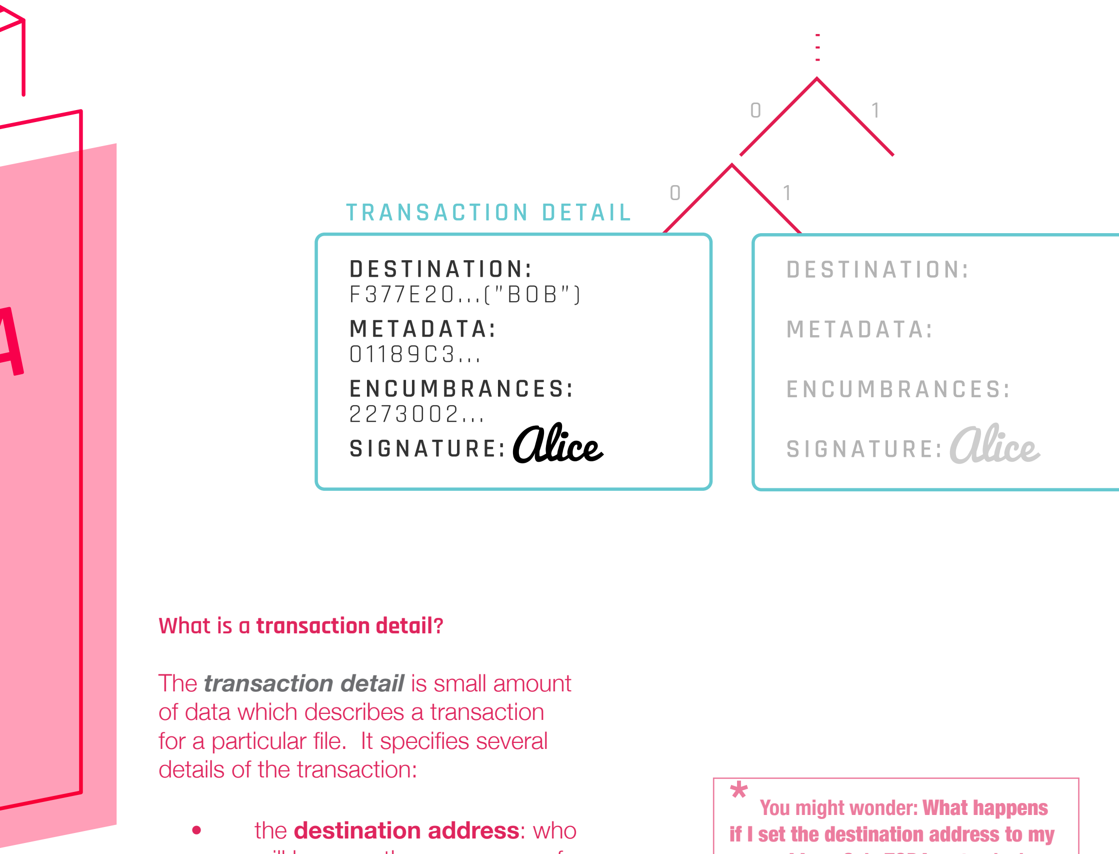


THE **PATH** IN THE TRANSACTION TRIE REPRESENTS THE **FILE IDENTIFIER** FOR THE TRANSACTION



What is a file identifier?

Remember how the ***payload*** of a file never changed? Just like a library book, **the core contents of a file can never change after it has been published**. We use the payload, coupled with other identifying information such as who created it, and hash them together to create a **unique identifier** for the file. This identifier never changes, even if the file is transacted, or if it collects new metadata.



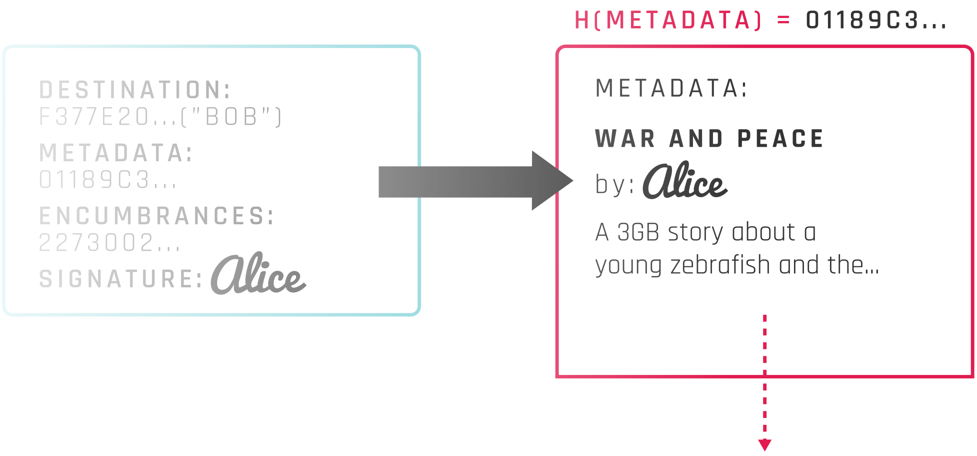
What is a **transaction detail**?

The ***transaction detail*** is small amount of data which describes a transaction for a particular file. It specifies several details of the transaction:

- the **destination address**: who will become the new owner of the file
- a **metadata hash**, which allows users to attach arbitrary metadata to the transaction
- an **encumbrance**, which is a special restriction on how the transaction will occur
- a **signature** on this data, to certify that the current owner really wishes the transaction to occur

* You might wonder: **What happens if I set the destination address to my own address?** In TODA, not only does this work, but it has the beneficial effect of letting you attach new metadata to a file without having to send it to someone else.

* As we'll see, **metadata** has a number of creative uses in the design of applications using TODA technology.



Why not the full metadata?

The ***transaction detail*** becomes an intrinsic part of the proof of a file. TODA strives to keep proofs as small as possible, and if it were possible to include lengthy metadata in each transaction, this **would cause proofs to become unwieldy**. Instead, only the hash of the metadata is required, which guarantees it can't be changed. This also has the benefit of empowering a user to choose not to share the metadata of a transaction with someone else.

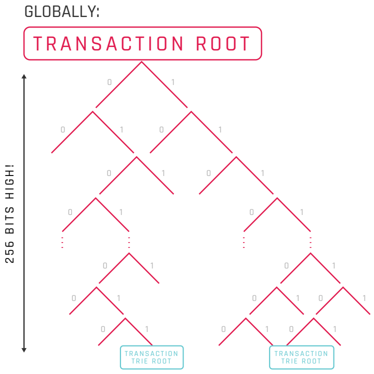
* **You may have noticed that this pattern of requiring only a hash and providing an option to reveal the full data is a theme throughout the design of TODA.**

‘Uploading’ transactions

Suppose you’re interested in transferring the ownership of 100,000 files to a colleague over TODA. These files could represent music licenses, or any type of asset at all. It would be unfortunate if this transaction required you to not only reveal the extensive details about the transfer to the network, but inherently burden the entire network with 100,000 simultaneous transactions.

Fortunately, network participants do not share the details of transactions with the network. Remember our discussion of Merkle tries? Well, the ***transaction trie*** is a merkle trie as well, and **the Merkle root of the transaction trie is a cryptographically secure representation of the entire transaction**. That single 32-byte hash contains a unique fingerprint of 100,000 transaction details. So, naturally, the only data shared with the network is a secure 32-byte representation of your entire transaction for a cycle.

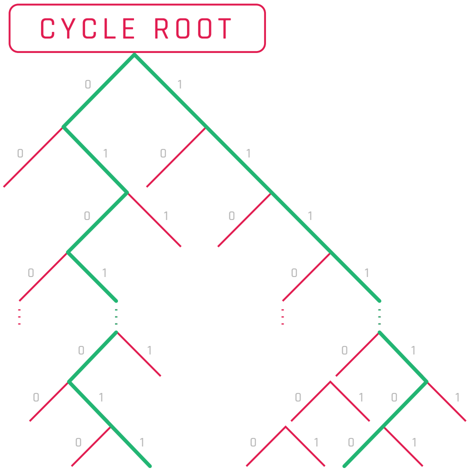
The only data shared with the network is a secure 32-byte representation of your entire transaction for a cycle



The Cycle Trie

The essence of TODA is the construction of a large, distributed Merkle trie by all online nodes. Since this trie is constructed in a distributed way, each online node is only concerned with a small slice of the overall trie. And, there is built-in redundancy to ensure that not only does the trie get built, but that it's built in a provably correct way.

So, what is in this distributed ***cycle trie***? The paths in this trie are a node's ***TODA address***, and the values are the corresponding ***transaction trie merkle roots***.



THE **PATH** IN
THE CYCLE TRIE
REPRESENTS THE
ADDRESS OF THE
TRANSACTING NODE.

What have we achieved with this scheme of ***cycle trie*** and ***transaction trie***? How does this form a secure system of asset ownership?

We need a way to prove that every transaction in the ***per-file ledger*** is legitimate, and we need a way to prove that the ledger is complete and not missing any transactions. Let's consider whether we can achieve this using the mechanisms we've just described.

Let's examine a particular transaction in a ledger of a file. That transaction has accompanying data which is its ***proof***. The role of the ***proof*** is to show that the transaction actually took place at the time it claims to have occurred. To do this, the proof needs to show that the purported transaction was a part of the ***cycle trie*** for that cycle.

Luckily, you know the Merkle root for the cycle trie of that cycle - our distributed trie building system ensured you received it. So the role of the proof is to establish this link between the transaction and the ***global Merkle root*** for that cycle.

*** If a node isn't transacting during a cycle, that simply means they don't have a Merkle root of their own to incorporate into the cycle trie. That's not a problem! In fact, at global scale, most nodes won't be transacting most of the time - and that simply makes the job of cycle trie creation even easier.**

*** If the owner of an asset transacted other assets during a cycle, they would have an entry in the cycle trie, but their corresponding transaction trie would be able to prove that the file in question was not transacted.**

It turns out to be quite easy to create small, efficient proofs which achieve this goal. The proof simply needs to prove that the sender of the transaction had a particular value in the cycle trie, and it needs to prove that the sender's transaction trie included the transaction in question.

Remember our discussion of ***trimmed tries***? Proofs are where they shine. Instead of using someone's entire transaction trie (or worse, everyone's cycle trie), the proof contains critical slices of these two tries. As an independent verifier, you could examine these trimmed Merkle tries, and verify that, when all of the hashing is calculated properly, you're able to match up the transaction all the way back to the global Merkle root.

At that point, it's tough to argue: **there's no way that transaction didn't happen**.

So how do you prove that the ledger is complete? That's an important question, with a fortunately simple answer. You need to show that for each of the cycles where the file was not transacted, that there was no corresponding entry in either the cycle trie or the transaction trie.

Double spending attacks

Double-spending attacks are when an owner of an asset is able to transfer it to two different users (often simultaneously). This attack is able to work in a cryptosystem if one recipient is unaware that the file is already in the process of being transferred to another.

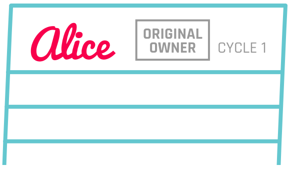
You'll notice that TODA is designed specifically to avoid the problem of double-spending. Remember the transaction trie a node needs to build in order to send a file? Well, **the information about a transaction can only occur in exactly one place in the trie data structure**: where the path is the file identifier for the given file.

What if a nefarious user creates more than one transaction trie, where each one represents sending the same file to different recipients? Here, too, the problem is solved with TODA's choice of data structures. The cycle trie only supports a single value per TODA address per cycle. So, **a node could create several different transaction tries for a cycle, but only one will ever be incorporated into the cycle**.

Nice try, double-spender.

Representing files

Let's investigate the lifecycle of a file. What does it look like when you create one, and how does it accumulate information as it gets transacted over time?



When you create a file

A newly created file starts with whatever payload you specify, and an empty ledger. The file becomes valid after an initial transaction transferring it to its creator. This serves to 'lock-in' the point of file creation, and record the event in its ledger.

Note that, because of how TODA is designed, whether you create one file, or a million files, it's the same tiny impact on the network as a whole.

Why would I create a file?

- Many TODA participants may never create files. Because TODA is designed to transact meaningful digital assets, it's often established organizations which give meaning to a file. A country might issue a file which represents currency, or a music producer might issue a file containing a license to a song.
- However, **there is nothing special about creating files** - there is no restriction on who can create one. Whether someone assigns any value to a file you create is more of a social matter than a technical one.

When you transfer a file

When a file is transferred, it will have a new entry in its ***ledger***, containing the ***transaction detail*** for that transfer. Further, once the transaction has been confirmed by the network, the file also needs to keep the proof that the transaction really took place.

When you do nothing to a file you own

Cycles continue to occur whether you transact or not. In order to be able to continually prove that your file's ledgers are complete and not missing any information, you need to collect a bit of data every cycle that contains proof that your address did not contribute to the cycle. Luckily, these trimmed proofs are especially short.

Gardening

All TODA network participants contribute to the distributed construction of the cycle trie. When nodes act this way, we say they're ***gardening***, because they're helping to grow a tree.

*** TODA uses a deterministic, but unpredictable mechanism to identify which nodes are responsible for building which parts of the trie. As they assemble their components and calculate the Merkle hash, they sign the result of their work using their private key. This ensures they receive credit for having contributed to the network, and provides an ability to hold bad actors accountable for submitting incorrect results. The mechanism is applied recursively, up the trie, until a consensus on a new root value is reached.**

You might be wondering how the actual process of contributing your transaction trie hash into the cycle trie occurs. We'll describe it step-by step here:

- A node signs the Merkle root of their transaction trie using their private key and shares it with their ***neighbourhood*** - the bucket of nodes that share the most similar address characteristics.

- Everyone in the neighbourhood shares their ***signed transaction trie hash*** and their ***address***. Each node takes this list of 16 addresses and 16 hashes and begins building a trie.

- Once everyone confirms they have built the same trie, all nodes in the neighbourhood sign the result with their private key. They then create a hash of the root of this limited trie.

- This process continues recursively, with larger neighbourhoods receiving the resulting partial tries of the sub-neighbourhoods below them. Ultimately, **this results in agreement on a single global Merkle root**.

* **Neighbourhoods** are a concept in TODA which describes how the cycle trie comes to be constructed. Neighbourhoods always have 16 members, and are deterministic, but unpredictable ahead of time.

What if I miss a cycle?

You might be wondering how you can prove your ledgers are complete if you happened to miss a cycle. Maybe you had a network interruption, or your device went offline. Fortunately, TODA has both protocol and service-based solutions to this situation.

At a protocol level, the nodes assembling the cycle trie are responsible for maintaining proofs of their neighbouring nodes in arrears. Fortunately many neighbouring nodes who didn't contribute to a cycle actually share the same proof, so this is a trivial set of data to maintain.

Further, several value-added services may maintain these so-called 'null proofs' on behalf of participants. Those using a fully-hosted services such as the **TODAQ TaaS API** never need to worry about missing a cycle.

Attacks against distributed trie building

- **Fraudulent contribution**: attempting to set the path of another address to a value of your choosing will not succeed as it has not been properly signed
- **Non-participation**: you do not participate in the cycle and are ineligible for rewards. The trie will be built without your assistance.
- **Network takeover**: does not result in an advantage in carrying out any attack other than a denial of service (for which other protections exist)

Hey is this owner valid? Proving ownership on TODA

Every TODA file can prove its validity using its ledger and stored proofs. You'll remember that the proofs are just trimmed cycle tries and transaction tries for every cycle the file has been in existence.

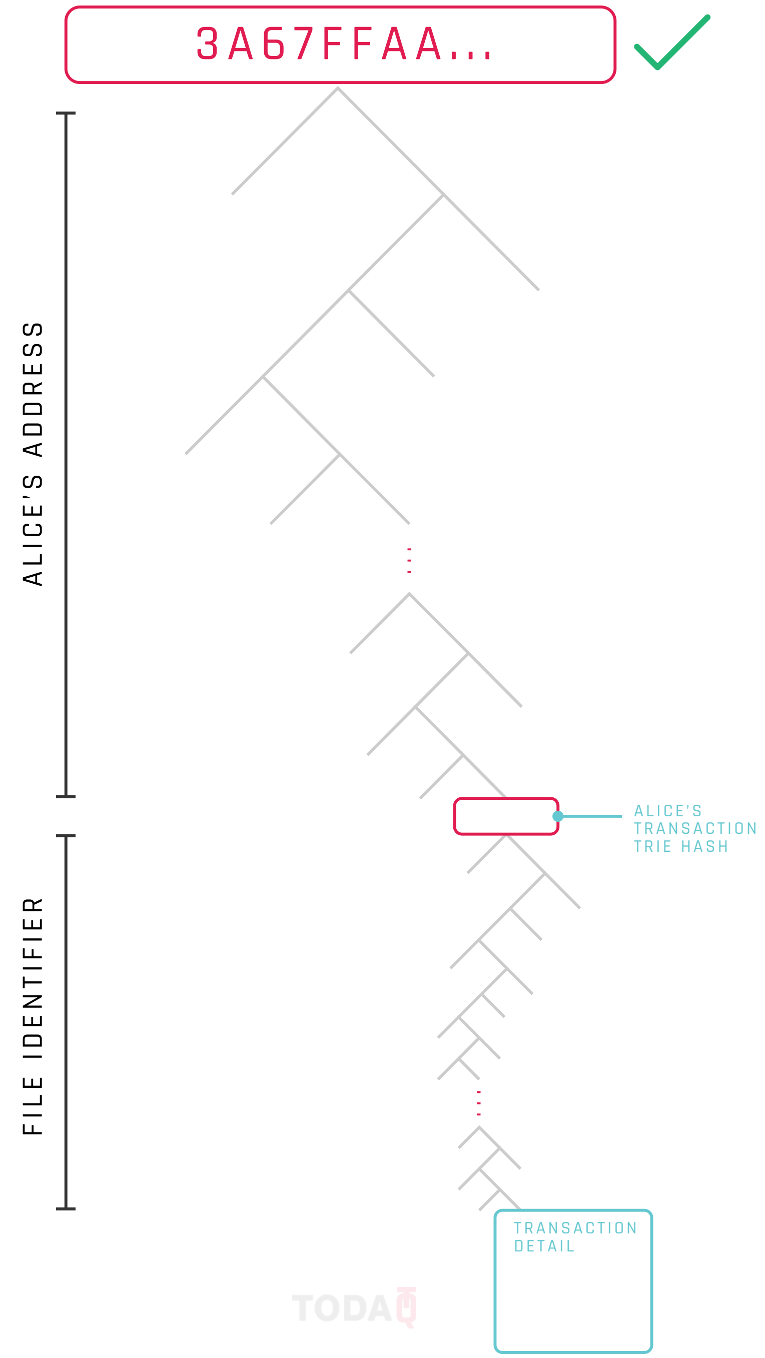
Let's look at how this process works step-by-step. We start with the first row in the file ledger. Let's call this ***Cycle 1***.

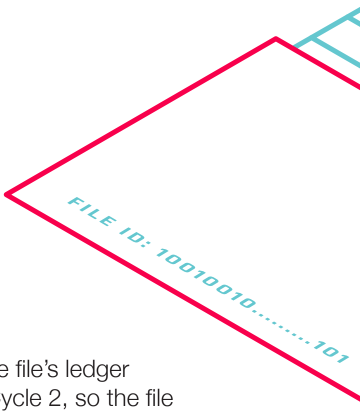
- In the definition of the file, we look up the file's ***initial address***. This is the address of the node that created the file.
- We compute the hash of the trimmed cycle trie for that row, and check that it corresponds with the ***global Merkle root*** for Cycle 1. If it matches, we know the initial owner of this file really did transact during this cycle.
- Now we compute the hash of the ***trimmed transaction trie*** for that row, and check that it corresponds to the value stored in the cycle trie. If it matches, we know that this file was really transacted during the cycle.
- Finally, we check the ***transaction detail*** for this ledger entry. We check that its hash matches what was stored in the transaction trie, and we check that the destination address matches the file's initial address.

* Remember, the proof that a transaction occurred is a combination of a **trimmed cycle trie**, and **transaction trie**.

If all of these checks have been satisfied, it's not possible that the file in question wasn't created and transacted to its initial owner during the cycle.

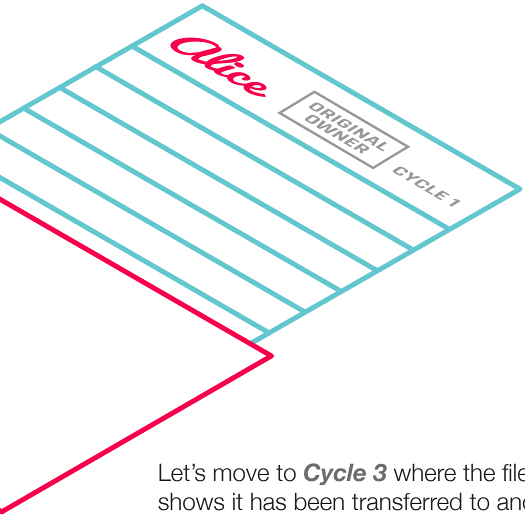
CYCLE 1
GLOBAL MERKLE ROOT:





Let's move to ***Cycle 2***. The file's ledger doesn't have an entry for Cycle 2, so the file shouldn't have been transacted. Let's see how this is proven.

- We look up the file's trimmed cycle trie for Cycle 2, and verify that it hashes to the global Merkle root for Cycle 2.
- We examine the value for the address of its current owner (its initial owner). If there's no value for its current owner, we're guaranteed that owner did not complete any transactions during this cycle.
- If there is a value for the current owner, we examine the trimmed transaction trie and verify that the hashes match. We then look up the file identifier in the transaction trie and verify that it has no value. If so, we're guaranteed this file was not transacted. If we did happen to find a value for this file identifier, then something has gone awry, and we're missing a row in this file's ledger.

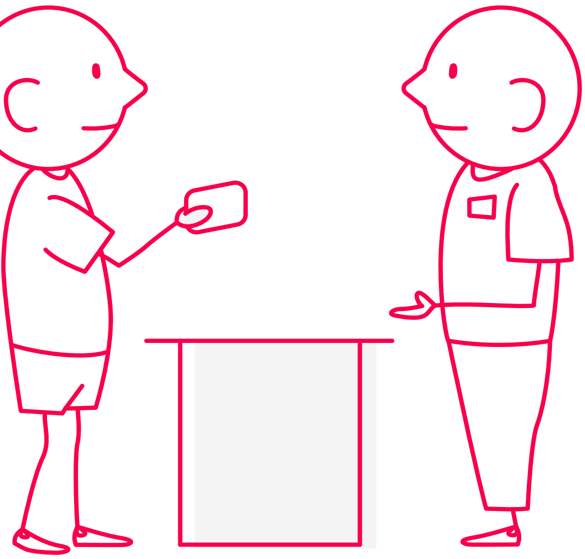


Let's move to ***Cycle 3*** where the file ledger shows it has been transferred to another owner.

- We look up the file's trimmed cycle trie for Cycle 3, and verify that it hashes to the global Merkle root for Cycle 3.
- We examine the value for the address of its current owner (its initial owner). If there's no value for its current owner, the file hasn't been transacted this cycle and the ledger is not correct. In this case, we discover it has been transacted.
- We calculate the value of the trimmed transaction trie for this cycle and verify it matches the value in the cycle trie. We then look up the file identifier in the transaction trie and verify that it does have a value.
- We verify that the hashed transaction detail for this row matches the value in the transaction trie, and we verify that the destination address matches the new file owner.

Designed for Business

TODA was designed with the needs of real-world applications in mind. It represents an engineering solution to a complex theoretical problem.



Why is TODA especially well-suited for real business needs?

Anything you like can be a digital asset

There are no restrictions on what can become a digital asset, and there are no restrictions on creating them. TODA can be used to represent a hotel room reservation, a virtual asset in a game, or a consumer loyalty token.

A TODA file can also be as large as you need. Newly-recorded 3-hour 4k video could be published as a studio asset, without the need to actually distribute the contents of the movie. It's enough that TODA is able to lock the payload of the file to the hash of the video.

The Sword of Power
Why would transforming game inventories into transferable digital assets be important? Well, gamers spend time and effort to earn and purchase components of their inventory within different gaming universes. Unfortunately, games don't last forever, and once the game server shuts down, not only can those assets never be used again, but there's no way for someone to show that they ever earned an asset.

If game developers offer true digital assets, not only do these concerns disappear, but it opens a new ecosystem for development, where games developed by different firms can take advantage of assets someone has earned in another game. This breaks the wall of proprietary participation, to deliver a richer, more resilient development ecosystem and gaming experience.

*** In technical terms, we say that the file payload is shared out-of-band. This frees up the network to focus exclusively on the delicate problem of managing ownership, and leaves the owner of a file to decide whether – and how – to transfer the full content of the asset to someone else.**

Low hardware requirements. Use your phone.

Participating on the TODA network requires minimal work. Some hashing, and the occasional cryptographic signature. There's no unnecessary work, and as we've seen, the burden of large transactions is carried only by the sender and receiver. That means, for most applications, a mid-range consumer phone is more than capable of acting as a first-class participant on TODA.

No Proof of Work (or Stake!)

There is no cyclic "challenge" for TODA participants to engage in. This tragic waste of energy was used to regulate block frequency of early blockchain systems, and continued to serve a role in electing leaders and arriving at a consensus.

As we've seen, TODA has no such requirements, and the bulk of energy use is incurred only by those who are actually transacting. This has the benefit of aligning incentives as well!

Removing the need to mine for currency

TODA is an equal-access protocol. There is no built-in special type of file you need to purchase to make the system work. Because incentives are aligned so that only those transacting do most of the work, there's no need to create a special financial instrument to make the system run. This also has the benefit of reducing incentives for bad actors, and eliminates mining farms and other wastes of effort.

On TODA, you'll never run out of gas.

Low network costs. Transmitting reality 32 bytes at a time.

The fundamental piece of shared information on TODA is the global Merkle root. Instead of replicating a multi-gigabyte ledger to every participating node, the only true universally shared data is a 32 byte hash.

Hash comparisons

You may have noticed that the majority of TODA relates to hash functions - those fingerprints which can summarize data in a short number of bytes. TODA relies on hashes because they're so efficient - much more so than more sophisticated cryptographic operations. Many computers - including mobile devices - also support hardware acceleration of hashing, making them even faster.

No more TMI. Keeping private transactions confidential.

Unlike many systems where the details of every transaction can be snooped using a blockchain explorer, TODA only shares the hash of your transaction with the network. This means that the details of your transaction remain confidential unless you choose to share them with someone. If you do, you're able to prove that the details your providing really do represent what occurred.

Many blockchains distinguish between a ***public chain*** and a ***private chain***. In TODA, there's no need for such a distinction. The chain is available to everyone, but all the details are **private by default**.

Audit trail friendly. Network private.

If you remember our library book loan card metaphor, every transaction ends up in the per-file ledger attached to a file. That means, if you're the current holder of the book, you can look at the loan card and see its whole history. Similarly in TODA, if you're the current owner of a file, you have all the information to trace its whole history.

In contrast, in replicated ledgers, everyone can see everything. **In TODA, you can only see the history of files you currently hold.** If you're no longer the owner of a file, you won't receive updates about where it goes in the future - just like a library book.

Transaction fee free.

The cost of participating in TODA is the minimum amount of computation needed to keep building cycle tries. There are no transaction fees - there's not even an official "currency" you could pay them with.

There are some incentives to continuing to contribute to the network, but those are optionally issued by private parties - and helps them as much as it helps those taking part.

Constraints on file types provide efficient smart contract functionality

TODA doesn’t do “dApps”. That is, it doesn’t support running arbitary code inside a large complex transaction. Instead, TODA is able to support smart contract functionality through the use of a specific set of encumbrances on transactions. Unlike “dApps” which are ***Turing complete***, encumbrances are a more limited set of constraints that can be applied to a transaction, and are guaranteed to be efficient to calculate.

TODA supports a number of encumbrances out-of-the-box, including the ability to constrain a transfer to be a ***loan*** (return to the current owner after a certain amount of time), a ***lock*** (not be transferable for a certain amount of time), or an ***atomic swap*** (require a specific asset be exchanged for this one).

Ready to go today.

TODA has been in research since 2015 and under active development and engineering since 2016. **The easiest way to get started is using a hosted service, such as TODAQ’s Toda-as-a-Service API.**