Tokeda
Viable token-driven metadata within Bitcoin

By Joannes Vermorel, last updated, March 30, 2018

Acknowledgements: Tomas van der Wansem for discussing metadata placement options, Jason Cox for pointing out the direct mode, Shammah Chancellor for need of structure within UTX specialists, Amaury Séchet for the quantum resistance angle as well as plenty of insight on Bitcoin as a whole and finally Forrest Elizabeth and Antonia Schmidt-Lademann for numerous phrasing improvements!

Status of the draft: EARLY DRAFT, INCOMPLETE

In the present document, Bitcoin always refers to Bitcoin Cash

Abstract: Tokeda addresses both the challenge of viably preserving an unbounded amount of metadata without endangering Bitcoin itself and the challenge of introducing tokens within Bitcoin by weaving the two problems through aligned economic incentives. Tokeda is compatible with stateless wallets (which include SPV wallets) and requires no consensus change. As a token scheme, Tokeda relies on a trust-but-verify security model centered around the issuer. The issuer is trusted with the relay of inbound transactions from users. The issuer takes care of routing the metadata to remedy the lack of such capabilities within the Bitcoin script. As a metadata layer, Tokeda leverages the UTX set (unspent transactions) as a purposefully pruneable key-value store, which is a superset of the UTXO set (unspent transaction outputs). Tokeda creates a market signaling mechanism at the issuer level, to foster an ecosystem of nodes which can selectively persist the metadata in the UTX set depending on the originating issuer. The author argues that Tokeda is an economically superior form of tokens compared to the code-is-law approach adopted by some of the competitors of Bitcoin. The author also argues that Tokeda is a provable way to incentivize miners to foster a token-driven economy backed by Bitcoin, instead of expecting the miners to subsidize tokens operated over Bitcoin.

Tokeda

Overview

Bitcoin faces both the problem of its own tokenization capabilities, and the problem of the persistence of the metadata. Historically, the solutions to do either of the two challenges have quite a few undesirable drawbacks.

On the metadata side, transactions have been largely restricted to standard transactions that purposefully contain no metadata at all. Indeed, in the long run, Bitcoin will be operated from the UTXO perspective, as only the UTXO set is required to operate Bitcoin. Some form of UTXO commitment will ensure that Bitcoin can be provably operated through the UTXO set alone. As the ongoing maintainability of the UTXO set is critical to the ongoing survival of the cash use case Bitcoin, it is reasonable not to expose the UTXO set to any metadata at all. The UTXO set is reserved for the one first-class citizen of Bitcoin: cash.

Unfortunately, this means that any metadata must reside somewhere else. In this regard, OP_RETURN presents two challenges. First, the OP_RETURN entry is pruneable, therefore it should and will be pruned. Bitcoin does not offer a built-in economic mechanism to reward a miner that persists the OP_RETURN data. Second, at 220 bytes, the OP_RETURN may not be large enough to contain all the relevant metadata, not for all use cases. Unless a suitable convention is adopted - collating multiple OP_RETURN entries - the metadata has to be hashed and pushed to an undefined layer 2 solution which in practice looks a lot like a blockchain that competes with Bitcoin itself. This situation begs the question of the continued existence of Bitcoin: if another blockchain happens to be more capable than Bitcoin while preserving the qualities of Bitcoin, then Bitcoin should become this very blockchain somehow.

Tokeda introduces the notion of the UTX set, which is a superset data-wise of the UTXO set. The UTX set contains all the UTXO set plus all the Bitcoin transactions - the whole transaction not just its outputs - that still contain one unspent output or more. This concept is covered in greater details in the following.

On the token side, the Bitcoin script does not offer any control mechanism to enforce the propagation of the metadata from one transaction to another. It is possible to consider new script operators that would grant such a degree of control; however, this is a difficult undertaking. By design, control over outputs originating from inputs implies that iterators of some kind must be used. As a result, all naïve implementations result in quadratic computation time as a worst-case situation, which can be exploited for denial of service. Quadratic computation times have been eliminated in the BCH fork, and we have good reasons not to re-introduced them.

Tokeda leverages the token issuers to relay the token metadata of their users. Simply put, when Alice wants to send her token to Bob while the token issuer is Trent, it goes:

- Alice signals a token transfer intent to Trent pointing out Bob as the recipient. This requires one Bitcoin transaction, the signaling transaction.
- Trent settles the token transfer by indicating that Bob is the new token owner. This requires a second Bitcoin transaction, the settlement transaction.

In other words, users trade among themselves by leveraging issuers as clerical metadata forwarders.

A stateless wallet is defined as a wallet can be setup once - offline - and is never require to persist any further data. A stateless wallet is a more stringent form of SPV wallet. The stateless wallet compatibility of Tokeda is ensured because a user only needs to persist two elements to transact with the token: its own secret seed and the issuer identity. All the relevant token metadata are located in the UTX set which can be dynamically queried on the Bitcoin network. This ensures that the users can operate in fire-and-forget mode. Users do not need to interact directly with issuers; the blockchain itself takes care of relaying the intents.

The security model is *trust-but-verify*. Nothing prevents an issuer from becoming corrupt except the issuer's own economic self-interest. Tokeda ensures that issuers are fully observable, and that observers can collect proof of bad behaviors; including not being diligent enough in acting as a relay.

This *trust-but-verify* approach also offers the possibility for certain Tokeda issuers to position themselves as *certification authorities*. Other issuers could seek their services to be publicly certified through their tokens. This approach offers the possibility for arbitrarily complex validations and due diligence to be performed on Tokeda issuers, while making the result of those certification accessible to users operating over stateless wallets.

Tokeda locates its metadata within the UTX set in a non-hashed form to avoid the need of introducing a layer 2 solution. Instead of relying on selfless behaviors of the miners to preserve the token metadata, Tokeda introduces explicit payment fees to be defined by the issuers and paid to the issuers. Among other options, the issuers can leverage those fees to pay mining nodes to preserve not only the UTXO set, but the subset of the UTX set relevant to the issuers as well. Tokeda specifies an extension of the networking language of the Bitcoin node to indicate which issuers' metadata are supported.

Naturally, any Bitcoin node remains free of preserving the whole blockchain. Tokeda is not a scheme to restrict nodes from gaining access to blockchain data, but on the contrary, a scheme to incentivize them to keep and service this data, while this data do not directly serve the first-class citizen use case, that is cash.

Tokeda also offers market signaling mechanisms such as *proof-of-worth* to let the market differentiate the valuable tokens from the lesser ones. Such mechanisms are important in order to facilitate the emergence of specialized actors who ensure that economically sustainable

issuers get their metadata widely broadcasted and readily available. In practice, it is expected that such a service would take a form similar to the CDN services offered by many cloud computing platforms nowadays.

Tokeda does not specify how those actors would be paid to reliably serve the UTX data. However, it's interesting to consider a bootstrapped approach where an UTX specialist would position itself as an issuer of a token dedicated to the distribution of the UTX metadata. Payments would be made to the UTX specialist granting issuers - the users of the UTX specialist - with a token representing their own commitment to serve their specific UTX metadata. The user's tokens would be regularly updated by the UTX specialist as resources get consumed serving their users.

Simplified use cases

When considering anything "blockchain-based", the primary concern is: *why do it on a blockchain at all?* Compared to a centralized SQL database, it is more difficult to build software on the blockchain. For the sake of readability, this section is only presenting brief three scenarios. The economic motivation behind those use cases as well as the most of the technical details are left to a section near the end of this document.

Long lived fungible token. A company wants to issue non-voting shares, which will grant token holders access to future dividends to be paid by the company. Users can trade their shares. As the company is long lived, a mechanism is required from the issuer to rotate its keys if the need arise. Also, as the company is already very large, it seeks a solution that scales properly up to 50 million shareholders.

Non fungible token. A social network company wants to offer a mechanism for its community to trade the valuable usernames which represent entities within the social network. The token mechanism should enforce that no two users can be granted the same username. In order to avoid name-squatting, usernames are only ever rented from the social network, and regular payments should be made, just like internet domain names.

Domain-driven fungible token. A Bitcoin miner wants to sell mining contracts that have three attributes: a nominal hash rate, a start date (inclusive) and a end date (inclusive). Tokens are tradeable, however the fungibility rules are non-trivial. Indeed, the hashrate attribute of the token is only additive for matching time periods. Also, the miner wants to be provably honest from the ecosystem at large.

The UTX set as a metadata layer

The UTX set is a concept introduced by Tokeda as a superset of the UTXO set. The UTX set includes all the UTXO set, but also all the OP_RETURN metadata that are associated to Bitcoin

transactions that have at least one UTXO entry. However, unlike the UTXO set which is a must-have for all Bitcoin nodes; the support of the UTX set is intentionally selective.

Each Tokeda-capable Bitcoin node, a networking dialect just like Compact Block, can decide to selectively persist selected OP_RETURN metadata at a issuer-level granularity. The issuer is identified through inspection of the input script. Tokeda emphasizes, but this is a convention rather than a requirement, that metadata associated with a transaction becomes pruneable once the transaction does not contain any more any unspent output.

The UTX set, as opposed to UTXO (outputs) appears to be the most suitable location to put the *non-hashed* token metadata, and potentially a substantial amount of it. In this section, we discuss the tradeoff and alternative options.

To store the token metadata, there are two orthogonal options to consider. First, we have to consider whether the actual data is stored in the blockchain or if only a hash of the data is stored. Second, we have to consider where within the blockchain the data is stored.

Tokeda favors the storage of plain, i.e. non-hashed, data. Hashing metadata diminishes the burden for the Bitcoin *blockchain*, however, it immediately creates another problem: the only way to reliably recover the hashed metadata is to have a layer 2 of some kind involved. Indeed, Tokeda relies on observers: issuers only behave correctly because they are under the scrutiny of the market. In practice, in the past the need to reliably recover hashed data has driven Bitcoin contributors into building a layer 2 solution that is itself a blockchain of some kind. This approach is self-defeating: if a blockchain can be made capable of dealing with the metadata that we need for tokens but that we can't have in Bitcoin, then, by definition, this is the blockchain we want for Bitcoin. Having Bitcoin as a crippled blockchain acting only as a notary service of some kind for another blockchain - the "real" one where everything of substance happens – does not make sense. Facing such a situation, it would be logical to fusion the two blockchains into one and call it Bitcoin.

Then, to decide where the data is stored, we have two options:

1. OP_RETURN in the transaction output
2. OP_PUSHDATA in the transaction input, OP_DROP in the transaction output with P2SH.

The option 1 is the most appealing one because OP_RETURN precisely carry the *semantic* of being intended for disposal. However, it's main limitation is the limit at 220 bytes. At the present date of writing, it is still 80 bytes, but various parties have expressed their intent to raise it to 220 bytes.

The option 2 offers the possibility to pack a much larger amount of metadata into the blockchain without resorting to non-standard transaction. However, it must be noted that the metadata is only "revealed" in its non-hashed form once an UTXO is claimed. From the issuer perspective,

the option 2 requires two sequential transactions; which might be acceptable if the metadata payload is substantial anyway (KB or more).

Other options could be considered for metadata such as non-standard transactions with output scripts containing OP_PUSHDATA / OP_DROP. However, such options fundamentally interfere with *money use case of Bitcoin* because of the burden that the token metadata put on the Bitcoin network is not paid for in full. Indeed, if the token metadata can sit in the UTXO (non-prunable) forever, then it can incur arbitrarily large storage costs for miners. For plain money use case, users are paying for the opportunity cost associated with not spending their money, which keeps the UTXO itself from unstainable runaway growth. Indeed, as Bitcoin will most certainly move toward one form or another of UTXO commitments in the future, only the UTXO needs to be preserved to keep Bitcoin being mined. It is reasonable to assume that Bitcoin miners may only preserve the UTXO set, entirely pruning the blockchain, and leaving other actors the burden of preserving more data beyond the UTXO. Similarly, hobbyists only interested by the pure money use case of Bitcoin – the most important one – may also do the same to minimize non-trivial hardware costs.

Input scripts containing OP_PUSHDATA / OP_DROP could be considered as well. However, until the UTXO is redeemed, the metadata is only available in its hashed form. The metadata is secured, but not accessible to observers which is an essential ingredient of the market dynamics of Tokeda. Then, by introducing two more transactions, i.e. having both the user and then the issuer broadcast not one but two transactions, the metadata could be made available in non-hashed form through the input scripts (redeemScript). While feasible, this approach accentuates further the overhead of Tokeda which requires two transactions instead of one. Yet, in the goal is to push large metadata payload to Bitcoin, largely beyond the OP_RETURN limits then, this approach would become the most efficient one.

Tokeda favors the OP_RETURN approach because it is intended to be fully pruneable - but it gives Bitcoin nodes the latitude to decide not to.

The UTX set is a market-driven compromise. Some nodes in the Bitcoin network may signal their capability to serve the UTX dataset at least in parts. The UTX set, like the UTXO set, is expected to be substantially smaller than the blockchain. The UTX set offers a signaling mechanism from the market to signal that some entries are more valuable than others. Indeed, all UTX entries represent “spendable” transactions. Thus, if there is a small non-zero amount of satoshis attached to a UTX entry, with a great statistical confidence, it can be asserted that this entry will be spent in the future; precisely because it’s too costly for the Bitcoin owner to attempt to bloat the UTX that way. Indeed, it does not make economical sense to let money sleep forever. This rationale is identical to the rational guaranteeing that a runaway UTXO growth will not happen as the miners will not bankrupt themselves.

The UTX represents a second-class data citizen – but a citizen still – of the Bitcoin world, while UTXO is the first-class citizen. Carrying the burden of the UTX is left to a class of actors who are

willing to cope with larger infrastructure costs, but without fundamentally changing the infrastructure costs for the actors who only care about the UTXO. If there is no actors who are willing to cope with the UTX infrastructure costs, then the most logical explanation is that those metadata and their underlying tokens are not worth it.

Within the UTX set, the token economic signaling (discussed in the following) plays a key role to offer more granularity in the selective preservation of parts of the UTX set. Indeed, the issuer itself ensures the ongoing preservation of the metadata relevant to its token. For example, the issuer could use the revenues generated by its own token to pay actors to make the relevant UTX subset available to the Bitcoin network at large. If the issuer fails at supporting the simple preservation of its own token metadata, then this metadata is simply not economically sustainable, and the market is doing the right thing: deleting the data, and recycling the data storage capacity for a better use.

Trust-but-verify security model

While it might be tempting to think of tokens as purely mechanical programs that operate only looking inward within the blockchain, there is little support from actual use cases for this perspective. A token is fundamentally attached its issuer, and hence trusting the issuer is a requirement, not an option. However, trust does not have to be blind. Tokeda emphasizes a mechanism where observers can automatically detect entire classes of bad behaviors from the issuers and report them as *corrupt*.

Tokeda is not a trustless scheme. However, for any user holding the token, the ownership of the token itself is a demonstration of the user faith in the good behavior of the issuer. Indeed, the issuer is trusted - at the minimum - with the continued economic value of the real-world asset backing the token. Thus, while the scheme might appear as requiring more trust than desirable to be put on the issuer; in practice, Tokeda does not change the security model associated with a token in the first place.

Furthermore, by putting the metadata in the UTX set, the token itself is at risk of being pruned if it cannot generate enough incentives across the market to even persist its own metadata. Those incentives don’t necessarily originate from the issuer itself though; yet, for most use cases, the issuer is clearly the one having the most to lose if its own token is not persisted by anyone.

Exposing a bogus token transfer is sufficient to prove an issuer corrupt, but the issuer can also be corrupt through other classes of subtle-yet-important inactions. Let’s consider a company who has issued shares represented to pay dividends. Proving that the company is honest in its dividend payments is as important as ensuring that the share transfers are honest. Yet, if the company has 1 million shareholders - as many large companies do - then multiple Bitcoin transactions to pay out the dividends are needed in practice. Indeed, although Bitcoin transactions may grow from their current limit (100 kB), transactions of 10 GB are probably to remain impractical in the foreseeable future because of the DOS vector they represent (see

Annex). Thus, the company would have to fragment its payouts over multiple transactions which may end up being in different blocks. The use cases listed in the previous section provide even more subtle cases, for example the verification of the payouts on Bitcoin mining contracts.

Fundamentally, checking that the issuer is honest requires a holistic view on the issuer activity, which is data-wise an unbounded problem. Thus, to some degree, keeping the issuer fully in check is forever beyond the capability of a stateless model; and of any Bitcoin node looking only inward as well.

Yet, users do not need more than stateless wallets to operate; practically speaking, they can "fire-and-forget" their transactions. Users do not need to directly reach out to the issuer to have a special transaction crafted for them. This prevents hostile third parties from censoring the issuer or censoring users from reaching the issuer. Moreover, thanks to the economic incentives put on the metadata availability within the UTX set, stateless wallets don't have to rely on unbounded internal storage. Deterministic public key generation can be used to recover all the tokens possessed by a given user.

The issuer does have to routinely publish transactions to relay. Thus, the issuer cannot stay strictly offline. However, Tokeda does not require the issuer to reliably be online. Also, by using multiple signatures, Tokeda offers the possibility to the issuer to delegate the relaying work to specialized trusted companies who would execute the mundane token relay transactions.

The trust-but-verify security model of Tokeda is similar - in spirit - with the *certificate transparency* scheme that is nowadays governing the issuance of SSL certificates: https://www.certificate-transparency.org/

Certification authorities

Validating whether an issuer is honest requires efforts for an external observer. While there are classes of easily identifiable dishonest behaviors, real-world situations typically requires an holistic view of the world at large to decide whether an issuer can be trusted or not. Tokeda offers a natural solution to this problem: issuers who act as *certification authorities* for the issuer's themselves.

When an economic actor positions itself as a certification authority for Tokeda tokens, this actor emits its own token which explicitly lists all the issuers who are deemed as "certified", that is non-corrupt according to the certification authority.

The business of the certification authority is to charge - as a service - any issuer it certifies. As the certification only reflects the honesty of an issuer at a certain point of time, an issuer who is seeking to maintain an ongoing market trust logically seeks a subscription service where the certification authority updates its own token entry on a daily / weekly / monthly basis.

As the certificate authority itself can be proven dishonest or simply not diligent enough in identifying bad issuers, this issuer has an economic interest to remain honest. Indeed, as certificate authorities compete, it is in the best interest of a competitor to prove the issuer corrupt, as its clients would flock away seeking alternative solutions.

Certificate authorities are particularly appealing from a stateless wallet perspective, because they provide a practical way for the wallet to check whether a given issuer is honest or not in practice. Successful certificate authorities would make themselves part of the default settings of popular wallets, offering a high degree of security to users with little friction involved.

Issuer's fees

The issuer needs a mechanism to get paid for the transactions to be relayed. Indeed, an issuer cannot be held liable for an indefinite amount of token transfers that are beyond its control. Therefore Tokeda proposes that all inbound transactions must carry a payment greater or equal to their own Bitcoin miner fee.

A minima, a token specifies a *multiplier ratio* greater or equal to one. This ratio is used to compute the fees that must be paid to the issuer to cover the costs associated with the relay transaction. The fee paid to the issuer by the user should be at least the miner's fee multiplied by the ratio. On top of this minimal fee, other fees might be requested by the issuer according to various rules of his own design.

There are two purposes for those issuer's fees. The first purpose is to prevent an economic exhaustion of the issuer simply caused by a huge growth in usage of his own token. It would be counterproductive if all widely popular tokens invariably bankrupted their issuers in mining fees; thus users have to pay issuer's fees to cover the settlement transactions.

The second purpose of the fees is to make sure that the token metadata are properly persisted. Indeed, a widely popular token can generate substantial issuer fees as well. If the issuer cannot support paying enough actors to distribute the UTX sets; then the token itself is not economically viable. Users, the ultimate beneficiaries of the token existence, cannot expect to get unbounded data storage for free; they have to pay for that one way or another.

The overall design puts a healthy pressure on the issuer itself to make sure that the amount of metadata generated by his own token remains economically viable.

As the need for transaction relay is an essential ingredient of Tokeda, it is of primary interest to make it as easy as possible for specialized actors to emerge to fill that role within the market. For example, a company emitting shares as token might not be interested in devoting internal resources to relaying its own token and would rather trust an actor that has a long blockchain-provable history of being an honest co-issuer across many tokens.

Code-is-law is economically inefficient

While the Bitcoin script is fairly capable, it still lacks mechanisms to control a flow of metadata from one transaction input to one transaction output. Tokeda solves this problem by displacing the tokenization logic outside the realm of the Bitcoin miners and transferring this responsibility to the issuers themselves - who are already trusted with the continuation of the assets. At its core, Tokeda is nothing more than an opinionated convention on the token metadata. We argue that this issuer-centric approach is not only suitable for Bitcoin, but that, unlike code-is-law, it has a reasonable chance of succeeding in the real world. Thus, this aspect of Tokeda should not be seen as a weakness to be improved upon, but rather a fundamentally desirable property of any tokenization mechanism intended to cope with the irreducible complexity of the world at large.

The code-is-law approach has been widely popularized by Ethereum. Yet, it remains to be proven that economic actors involved with physical contingencies (e.g. supply chains) are willing to enslave themselves to “dapps” (distributed app) of their own making and achieve operational success while doing so. So far, the code-is-law has been plagued by a succession of catastrophes brought by a series of breaches in the contracts; one of them resulting in a notable fork of Ethereum itself. It is possible that the software ecosystem at large will ultimately learn how to craft perfectly secure dapps over time, however, as the last 40 years of software engineering has empirically demonstrated, that even companies who have access to the most incredibly talented workforce - Google, Apple, Amazon, Microsoft, Intel, etc - are still struggling with bugs in their critical systems. Why would code-is-law software be immune to those problems if anything else is attempted but “dumb” contracts which are trivial to implement and verify? We fail to see any reasonable answer to this question.

At its core, code-is-law puts token issuers under tremendous pressure to have, from Day 1, not only a dapp that is provably secure *forever*, but that will also scale *indefinitely* if the usage of the dapp grows. Those two goals fundamentally work in complete opposition.

For example, let’s consider a simple non-fungible token intended to trade usernames on a social network. The simplest, most trivally secure, implementation of the token state consists of maintaining a list of all the usernames that have been already claimed. When a user wants to claim ownership on a username that supposedly remains available, the issuer iterates over the list, and does an equality test between every single existing single usernames to the candidate username. If no match is found, then the issuer adds the new username at the end of the list. This approach is easily provable as correct. This approach is also dramatically wasteful in computing resources. Yet, as long as there are less than 100,000 users, there is no economic incentive for the issuer in figuring out a solution to a scalability problem *he does not have*. If the number of users grow beyond 100,000, then the token logic should most certainly be re-implemented with a hashtable. In practice, this will require many times more lines of code, and create many dangerous opportunities to ship a flawed token logic. However, as the social

network has grown, more economic resources can be put to the task. If the number of users grow beyond 100,000,000, then the performance delivered by a non-concurrent hashtable will start to fall apart. Again, the solution most certainly lies in having a distributed hashtable, which creates orders of magnitude many more opportunities to ship a flawed token logic. Yet, such a complicated solution has now become economically feasible because the issuer is now a large corporation with plenty of resources at its disposal.

The belief of the author is that no amount of standardized libraries will be sufficient to even closely match the diversity of the real-world use cases. Even if the distributed hashtable case pointed out above happens to be fully covered as a standardized use case, any slightly innovative token - almost by definition - will require more than what the standardized code-is-law libraries will provide.

This belief is not inconsistent with the fact that *some* tokens - mostly trivial ones - implemented with a code-is-law approach might succeed economically. After all, Bitcoin itself is a trivially fungible token. Yet it took a decade of effort, still ongoing, to make Bitcoin both *secure* and *scalable*. Our main point here is that achieving success with a code-is-law approach will be the exception rather than the norm.

Tokeda does not pressure token issuers into upfront investments that any business sense would dictate to postpone. Tokeda also puts minimal pressure on Bitcoin itself. While there are two bitcoin transactions involved for a token transfer - which is likely to be the dominant token operation for most tokens - there is only one UTXO entry per token entry, which is as compact as it can get, as an UTXO entry is basically a naked authorization scheme. Miners only have to cope with the raw bandwidth requirements of Bitcoin, and bandwidth is arguably the most scalable aspect of the Bitcoin network.

UTX signaling for Bitcoin nodes

At the present time, Bitcoin nodes are being selfless in the capabilities to serve the entire blockchain to anyone. However, in the future, the blockchain will be pruned by most actors. Indeed, no matter how much the hardware cost decreases, the block size consumption will conversely increase to take advantage of the processing capacity. There is no compelling economical reason to have the miners prevent themselves from radically growing the blocks, while onboarding many users, if its profitable for them to do so.

However, in this future world of UTXO-only Bitcoin, the entire UTX set is also pruned by default, except by the actors who have an explicit financial incentive in preserving the metadata associated to issuers who reward them financially. Thus, Tokeda proposes an extension of the Bitcoin networking language where Bitcoin nodes can signal for Tokeda. This approach is similar in spirit to CompactBlock or XThin that are specialized protocols that are not supported by all nodes.

Each issuer is identified by its hash (a multi-signature script hash in practice). The explicit support for each individual issuer can be probed for each Bitcoin node. However, Tokeda also features an optimization where every node can communicate a Cuckoo filter to express the list of tokens it supports.

The purpose of the Cuckoo filter is to offer a performant option for specialized actors to identify, within the small world of Bitcoin nodes, which is the one most likely to positively acknowledge a UTX request associated to a given issuer.

Proof-of-worth signaling

Tokens, just like money, are a fundamentally social construct. If they are useful, and successful in their usage, then specialist actors may decide to archive the token metadata as it may prove to be of value later. Also, Tokeda is intended to foster the emergence of token marketplaces. It's through those marketplaces that users will gain the most valuable information about tokens. Indeed, as illustrated by the use cases, the honesty of an issuer cannot be accurately reflected with a binary status corrupt / non-corrupt. In the real-world, large organizations make mistakes all the time; and similarly any non-trivial smart contract implementation should be expected to face a bug once in a while.

The idea behind proof-of-worth signaling is to provide a standardized measurement to qualify tokens. The Bitcoin ecosystem should be able to build token marketplaces which provide meaningful information to the users. While the Bitcoin network cannot be effectively "spammed" - because of transaction fees, tokens can still be produced *en masse* potentially overwhelming marketplaces with inconsequential tokens; or even fraudulent tokens that trick users into believing they are buying another token.

Therefore Tokeda proposes to standardize a measurement that represents the total mining fees provably consumed by the token so far. The core challenge of producing this measurement is that it should be resilient to a *bad miner* attack where a single miner is trying to put on display a large mining fee for its own token, while the miner is effectively paying himself. Unless the measurement is correctly done, such a scheme could be used to fool marketplaces. The solution fundamentally lies in measuring the miners fees obtained from a *sequence* of consecutive blocks, while omitting either the first or last block – whichever is the largest.

TODO: considerable more details are needed

Economic viability of the tokenization burden

One critical aspect to consider for a token mechanism to be introduced within Bitcoin is the economic viability of the scheme for the Bitcoin network as a whole. In particular, it is not in the interest of the miners to subsidize third-party economic agents. This would lead to a *tragedy of*

the commons situations where the computing resources of the miners are abused by agents who are not paying the full price for the costs they incur to the Bitcoin network.

Each Bitcoin transaction comes with a miner's fee to cover the infrastructure costs. However, this is a one-time fee. While the bandwidth and the validation of the transaction are paid only once (by every miner though), the data storage will keep incurring linear costs over time, for a duration that is equal to the lifetime of entry within the dataset persisted by the Bitcoin node.

Thus, miners cannot commit themselves in preserving *any* data for an indefinite amount of time. Money itself, hence the UTXO set, is not even an exception to this principle. However, miners are economically saved from runaway UTXO growth, because Bitcoin users themselves have an economic interest in *not* letting their money sleeping forever. Indeed, the essence of capitalism is to generate compounding interests from capital. Sleeping money, while iti can be a store of value, cannot be a sustainable way of generating wealth. Sleeping money is economically outcompeted by circulating money accruing capital through productive use of the money.

The Tokeda scheme relies on a one-time miner fee (or rather two-time considering the transaction relay) to make the miner profitably process the token metadata. However, this process is one time cost as well - all Tokeda data being fully prunable - *unless* the miner decides to persists some UTX entries. The willingness of the miner to engage in this behavior is driven by the capability of the issuer to cost those extra costs some selected miners.

Tokeda does not rely on the miners subsidizing token-driven activities over the Bitcoin network, it rewards them financially creating an incentive to ensure the economic growth of an additional revenue stream.

Stateless wallets

Tokeda is designed to let users transact with stateless wallets. A stateless wallet is defined as an application that lets a user transact tokens without the need to ever be updated after its initial setup, which requires generating a secret seed and recording the issuer identity (*). SPV wallets are a less stringent form of wallet for Bitcoin; thus any stateless wallet is also a SPV wallet.

(*) If the issuer rotates all of its keys, there is nothing left of the original identity of the issuer, and the wallet would have to be updated. This behavior cannot be avoided if we want to preserve the capacity for the issuer to rotate its keys. However, it's still reasonable to refer to such a wallet as "stateless" from a practical perspective.

Stateless wallets are compatible with Tokeda because all the metadata is readily available in the UTX set which can be dynamically queried by the wallet to the Bitcoin network. A subtle but important point is that Tokeda does require a state to be maintained but this state is entirely delegated to the Bitcoin network.

Also, when operating from a stateless wallet, users are trusting the issuer by default. Yet, as pointed out above, if the users don't trust the issuer, they should not even engage in transacting with its tokens anyway.

An issuer also largely operates through a stateless wallet but only as far its own token scheme allows. For example, in the case of a perfectly fungible token representing company shares, relying transfer of shares can be done with a stateless wallet. However, if the issuer manages tokens that represent the usernames of a social network, then, for every request to create a new username, the issuer has to check whether this username has already been issued or not, which obviously requires a stateful implementation.

Direct metadata publishing

TODO: *this approach needs to clarify how we identify the issuer. Adding the issuer to the metadata payload might be too verbose for a good use of the 220 bytes of data*

In order to improve latency and to reduce the overhead on the Bitcoin network, a user can optionally use the direct mode of Tokeda that only requires a single transaction published by the user himself. As the direct mode can be adversely used by the issuer to censor users, it should probably never be the only option made available to users. However, by making the direct mode accessible to users with a public endpoint, an issuer can deliver a low latency option of high interest for certain use cases (e.g. supply chain traceability).

Then, as reaching out to the issuer, when applicable, is a simple way to reduce the overhead on the Bitcoin network, this mode is part of the Tokeda specification if only to avoid having inconsistent implementations to emerge as an afterthought.

The direct mode operates as follows:

1. The user submits its own signed token request, which includes a timestamp, to the issuer.
2. The issuer validates the token request and the timestamps, countersigns the message, and sends it back to the user.
3. The user publishes the final issuer response in a Bitcoin transaction.

Unlike the indirect mode, the user Bitcoin transaction is directed at the final recipient of the token, as the UTXO entry will be used to claim further action on the token.

The direct mode introduces one complication: the issuer can be defrauded by a user who does not proceed with the latest publication step. Similarly, the issuer can be defrauded by a user who manages to blur the situation through a successful double-spent on 0-conf.

Yet, in those cases, the issuer can still publish the proof that it has received the original user request. The countersigned timestamp ensures there is no ambiguity on the time when the user request was made. However, this puts the burden of a later transaction on the issuer himself. This could represent an attack vector against the issuer by bad users.

To eliminate this attack vector, the issuer can request a small deposit in Bitcoin to be made ahead of time by a user who seeks to use the direct mode. This option makes sense when one user is expected to perform multiple direct token requests, as the deposit would remain available as long as the user properly publishes its transactions.

Token capabilities

When creating new tokens, Tokeda makes it possible to allow or disallow the following operations:

1. Issuer can update issuer
2. Issuer can grant new units
3. Issuer can revoke existing units
4. Issuer can call back tokens
5. Issuer can freeze tokens
6. Issuer can unfreeze tokens
7. Issuer can update tokens
8. User can revoke existing units
9. User can transfer existing units

A token holder is an agent who has control over an UTXO entry that is decorated with a Tokeda metadata. The metadata is written in the UTX set. This input script identifies the issuer, while the metadata details the state of the token. When the user Alice wants to send the token originally issued by Trent to another user Bob, the scheme goes in two steps. *First*, the user Alice produces a Bitcoin transaction that spends its TXO entry toward an address in control of Trent, the issuer. This first transaction contains metadata that expresses the intent of Alice, which is to forward the ownership of the token to Bob. *Second*, Trent, being a honest issuer, executes the original intent of Alice by producing a second Bitcoin transaction that forwards the ownership of the token to Bob.

Neither Alice nor Bob, the two users, need anything more than a stateless wallet to transact their tokens. Trent, the issuer, needs to be online regularly, but reliably, to act as a relay and get the token effectively transferred. At no point does either Alice or Bob need to establish any direct link with Trent. In practice, using a multiple signature scheme, Trent can even be multiple entities who vote through a k-of-m keys signing mechanism.

Technical specification

The intent of this section to lay the groundwork for an implementation of Tokeda.

Envelop of a Tokeda transaction

A Tokeda transaction is a standard Bitcoin transaction. It's a convention-based scheme that tries to leverage the existing building blocks of Bitcoin to minimize the overhead for the Bitcoin network introduced by the token scheme itself. The *envelope* of a Tokeda refers to the part of the data that refers to the ownership of the token, not the state of the token *per se*.

An issuer's transaction is as follow:

- The input script identifies the issuer through its public addresses
    - Tokeda issuers are expected to have vanity addresses prefixed with 'tokeda'.
- A null-data output contains the actual token metadata
    - 1 byte: the Tokeda version number.
    - 1 varint: the issuer counter
    - 1 signed byte: the chunk identifier, the negative sign flagging the last chunk.
    - all bytes that remain: the inner payload
- The output of index 0 identified as the recipient of the token.
- An optional output of index 1 identified as the change address of the issuer itself.

Generating a Bitcoin address prefixed with 6 characters take about 1 minute on a consumer grade computer at present date. As issuers are expected to be long lived, this burden is a negligible friction. Yet, for the ecosystem at large, it provides a simple way to heuristically isolate all the Tokeda metadata within the blockchain.

The metadata is carried by the OP_RETURN payload. The first byte identifies the Tokeda version: an integer at zero, which can be increased up to 254 in this format if the need arises - the 255 value being reserved for the case where even more versions would be needed.

The second entry is a number acting a counter, namely the *token counter*. Indeed, race conditions can happen, and there should be no ambiguity on the ordering of the token operations as expressed by the token issuer. Tokeda expects strictly tokens to be updated through strict +1 increments, just like invoice numbers. This practice does not prevent large degree of parallel processing in practice, but it simplifies entire class of verifications of the issuer honesty. The counter is expected to be initialized at zero with the *intent* declaration of the issuer itself.

The third entry is the chunk identifier, which is basically a *hack* to work-around the 220 byte limitation of OP_RETURN. If the need arises, the issuer can spread its metadata across multiple transactions, up to 127 according to the specification given here. If we assume that token counter can be represented with 4 bytes, then the maximal payload for a Tokeda token is 9,398 bytes for OP_RETURN at 80 bytes and 27,178 bytes for OP_RETURN at 220 bytes. However, smaller OP_RETURN values come with a substantially higher overhead for the Bitcoin network.

A user's transaction is as follow:

- The input script identifies the issuer through the UTXO that is redeemed
- A null-data output contains the actual token metadata
    - 1 byte: the Tokeda version number.
    - 1 signed byte: the chunk identifier, the negative sign flagging the last chunk.
    - all bytes that remain: the inner payload
- The output of index 0 identified as the issuer.
- An optional output of index 1 identified as the change address of the user itself.

Unlike the issuer's transaction, the user does not get to provide a version number. It's up to the issuer to remove all timing ambiguities on token requests.

Inner payload of a Tokeda token

The inner payload represents the state of Tokeda token. For fungible tokens, the state represents total value of the token expressed in token units, however varied situation exists and Tokeda attempts at reflecting the diversity of those situations. The specification of the inner payload of the Tokeda tokens clarifies the expected serialization format.

The initial transaction published by the issuer is expected to be a public statement about the token itself. The issuer states what the token contains. The statement includes a simple list of fields. Each field has a name, a type and a semantic. The name is intended for human convenience. Every byte count on the blockchain, but

In the following, the data type *string* is always expected to be an UTF-8 sequence prefixed by its length (in bytes) with a varint. A zero-length string is valid.

The format is as follow:

- 1 byte (ISSUER_UPDATE: 0): the command flag intended to express the intent associated with the definition of the token itself.
- 1 string: a compact human-readable name for the token itself.
- 1 string: the URI (Uniform Resource Identifier) to find more about the token itself.
- 1 varint: the issuer counter value that defines when state transition rules starts to apply. This value is intended to give the issuer a way to initialize the state of its own token, while the initialization requires rights that the issuer does not want to grant for himself afterward.
- 2 bytes: the authorization flags that defines which state transitions are advertised by the issuer himself as legitimate, with 1 bit per flag (0 disallow, 1 allow).
    - Bit 0: Issuer can update issuer
    - Bit 1: Issuer can grant new units
    - Bit 2: Issuer can revoke existing units (for any field market as *additive*)
    - Bit 3: Issuer can call back tokens

    - Bit 4: Issuer can freeze the token state (rejecting user requests becomes valid)
    - Bit 5: Issuer can unfreeze the token state (rejecting user requests becomes invalid again)
    - Bit 6: Issuer can update tokens
    - Bit 7: User can revoke existing units or serial
    - Bit 8: User can transfer existing units or serial
- 1 byte: the number of fields within the token.
- For each token field:
    - 1 string: a compact human-readable name for the field.
    - 1 byte: the field type
        - Value 0: varint,
        - Value 1: fixed sequence of 32 bytes,
        - Value 2: fixed sequence of 64 bytes,
        - Value 3: varying-length sequence (zero terminated)
    - 1 byte: the field semantic
        - Value 0: fungible, applies only to the type *varint*, and assumes a preservation of mass for the field for every token transfer.
        - Value 1: unique, applies to any type, and assumes a preservation of the *unicity* of the value for the simply binary equality.
        - Value 2: ad-hoc fungible (partial semantic), applies to any type. Indicates that a fungibility rule exists, but that it is not the canonical one.
        - Value 3: non-contractual (no semantic), applies to any type. Indicates that the contractual rules do not apply to this field. The purpose of the field is something else entirely.

TODO: content missing

UTX commitments at the issuer-level

Tokeda adopts a ECMH-based (1) approach, similar to the one used for UTXO commitments to ensure that every observer can prune the token history, just like miners can prune the blockchain. This commitment reflects the state of all the activate metadata payload operated by the issuer. This allows compact proof of bad issuer to be produced. The UTX commitment also allow observers to safely discard all the issuer's history so far, if the issuer has not been badly behaving so far.

Each transaction pushed by the issuer comes with a counter that expresses an unambiguous ordering for all token operations from the the issuer perspective.The issuer is expected to regularly publish an updated UTX commitment, associated with the counter of its token.

(1) Elliptic Curve Multiset Hash, 2016, https://arxiv.org/pdf/1601.06502.pdf

Identifying Tokeda transactions

A metadata payload for Tokeda cannot be provably distinguished from a random payload pushed within a transaction for a completely distinct purpose. However, in practice, the serialization format of Tokeda should be designed in a way that does not accidentally collide with other metadata-driven schemes in the future - at the very least if those schemes are also trying not to collide with Tokeda. The serialization format should also be versionable.

Compact proof of bad issuers

TODO: observers should be able to publicly demonstrate - on the blockchain - that some issuers are corrupt. Ideally, the proof should be as compact as possible; even if obtaining the proof requires to process most of the blockchain transactions in practice.

Quantum resistance

Direct implementations of the scheme of Tokeda would expose the public keys, and thus make them susceptible to quantum attacks (if such a thing ever becomes possible). Yet, this attack vector is mitigated by the possibility of rotating the keys.

Then, as the issuer is trusted, it can maintain a clean state of the tokens off-chain, and use this state to recover from a successful attack. This would assume that Bitcoin itself is capable to recover from such an attack; but also that the market agrees with the recovery process proposed by the issuer.

References

An analysis of Bitcoin OP RETURN metadata, Massimo Bartoletti and Livio Pompianu, March 2017, https://arxiv.org/pdf/1702.01024.pdf

Extended use cases

This section attempts to demonstrate the economic viability of tokens operated over the Bitcoin, by providing a detailed list of archetypal use cases. The following use cases have these shared properties which are primarily desirable for any trading scheme:

- Tokens represent real-world assets, and one of the functions of the financial markets is to let users put a price on those assets and to have users transact *in Bitcoins* for those assets. This can be achieved with atomic swaps on-chain. In particular, the mechanism

should not require the *users* to trust each other; they should only trust Bitcoin, and the issuer (but issuer trust should be minimized whenever possible).

- Users benefit from the pre-existing blockchain infrastructure to propagate and persist their transactions at scale. Indeed, a company that starts to issue shares now might become a giant in 20 years. With a blockchain-based scheme, the company does not have to worry about upgrading its infrastructure or its processes every time it grows.
- Users have standardized, flexible, programmatic ways to exchange fiat currency over the blockchain. Indeed, one of the most frustrating aspects of software is the general lack of standardization; especially among enterprise software. While the blockchain itself might not be a game-changer for a solution that could have potentially lived in a centralized SQL solution, the quality and the depth of the tooling can make a radical difference in practice.
- Built-in inclusion for third-party players. So far, this specification details only two classes of economic agents: the issuers and their users. However, the transparency of the blockchain can and should be leveraged to bring entire classes of third-party players into the ecosystem. For example, marketplaces should be able to survey the tokens, and provide various services layered on top of Bitcoin.
- The transparency brought by the blockchain metadata ensures that bad behaviors from the issuers are publicly visible. Ideally, the technical requirements for identifying bad issuers should be made as low as possible; but there is fundamentally a tradeoff, as the amount of data to be verified for a given issuer can be arbitrarily large and cannot be fully internally conveyed through the blockchain.
- When applicable, tokens should provide issuers with known market mechanics that will create incentives for them to play a long game, and stay honest over time. While this might seem inconsequential at the beginning, in the long run, rational and competitive issuers can be trusted to remain honest. If not, then Bitcoin mining is broken anyway.

Fiat tether

A bank wants to offer the capability to users to transact USD amounts over the Bitcoin network. To do so, the bank freezes 1 billion USD and external audit companies monitor the ongoing availability of the funds. Then, the bank issues a Tokeda token denominated in USD. Token holders can redeem their tokens for a 0.1% fee (in USD) with the bank itself; which makes the token a profitable venture for the bank. As the bank is both regulated and audited, the bank must be capable of updating its tokens directly. However, as the bank's actions are fully visible to the network, the bank has a vested interest in maintaining the quality of its token, as other banks compete with other tokens that are essentially providing similar qualities.

What is the added value of the blockchain?

- The transparency brought by the blockchain metadata makes is much easier for auditors, who would be granted direct access to a bank's internal ledgers to audit the fiat deposits.

What are the gotchas involved?

- The token fundamentally relies on the bank remaining honest; much more than any mathematically provable property of the tokens themselves. Thus, if there are ways to create long term incentives for the bank to remain honest, then those mechanisms should be considered to be of primary importance. They will be the primary drivers to make the token work in the long run.

Ideas for the token

- When creating a token, the bank could structure a fee for its token, expressed in Bitcoin. This fee structure makes the token a profitable operation for the bank in the long run, which ensures the continued honesty of the bank. For example, the token should be able to define that every token transfer relayed should come with a fee (paid to the issuer) of at least 1000 satoshis (about $0.001 at the time of writing).

Non-voting shares

An automotive company wants to sell its own non-voting shares so that the shares can be traded as conveniently and as broadly as possible. The company issues a token denominated in share units. When dividends are to be paid annually, the company sends bitcoins directly to all of its token holders, the amounts being proportional to the number of shares involved.

What is the added-value of the blockchain?

- By issuing its own tokens shared through a public ledger, the company protects itself against a class of bad behaviors such as naked short sells, which have a negative effect on the share price. Shorting the stock is still possible, but if the counterparty is the stock itself, then users have to prove that they have ownership of their tokens before getting a short option from another agent.
- When the company wants to distribute dividends, the dividends can be sent directly to every token holder based on the amount of tokens that they possess. This makes the overall payment process fluid and painless.

What are the gotchas involved?

- Many large companies are over a century old. Companies' life cycles can last significantly longer than human lifetimes. It is reasonable to expect that companies' keys may be compromised over time considering their potential life-cycles. For this reason, key rotation is an obvious requirement for issuing shares that are expected to stay alive as long as the company does, which can be centuries.

- As the company must pay fees to transfer its Bitcoin dividends through transactions, the fees involved in reaching every single token holder must be accounted for. It seems fair to assume that from an initial total dividend (in Bitcoin), the company will define a flat transfer fee to be subtracted from the dividend payment for every single UTXO entry associated with a token. This has several consequences. First, it provides an incentive for token holders to consolidate their UTXO entries, which is good for the Bitcoin network overall. Second, it means that some tokens may hold too little to even receive a payment (because it would be an unacceptable loss for the issuer). Yet, in such situations, the issuer should not be considered as corrupt because it did not honor its dividend payments. The least unambiguous way of dealing with this situation is probably to have the token definition that defines the minimal amount in satoshis that should be expected to be subtracted from the dividends at each payment. By having the issuer announcing this value in advance, it would let the market consolidate its UTXO as users see fit.
- For any company who has a large set of token holders – possibly millions of UTXO entries – it should not be expected that payouts will be processed in a single monolithic transaction. Indeed, generating an arbitrarily large transaction might not be practical or even feasible depending on the Bitcoin consensus rule. Thus, token observers should expect more complex, but still valid, behaviors from the issuer paying its dividends.

Ideas for the token

- Just as dividends are paid quarterly for some stocks, the issuer could define the cadence of dividend payouts. Blockheights would serve as reference points for dividend payouts. The token holder can expect dividends every so many blocks.

Fixed hashrate mining contracts

A Bitcoin mining company wants to sell mining contracts for a fixed hashrate within a specific period. The company issues a token. Inbound transactions, carrying the bitcoin payments to the company, are turned into tokens. The users can then further trade the tokens among themselves. The company regularly pays dividends to all of its token holders as the result of its mining capacity.

What is the added value of the blockchain?

- Mining is obviously a very Bitcoin-centric economic activity, therefore it makes sense for the miners to offset their own investments in mining hardware to the market itself, and furthermore to do so publicly on the blockchain itself.
- A miner is more attractive to investors when the market can observe that it is provably fair, honoring its contracts for a given amount of bitcoins buying a given amount of hashrate over a specific period of time.
- The miner can claim blocks that it finds by attaching its token identity to the coinbase signature. Since an investor owns a percentage of the miner's total hashrate, the investor knows how many blocks the miner should find each day. The investor compares

the miner's total hashrate against the hashrate of the entire network. If a discrepancy emerges over time between the number of blocks that an investor would expect the miner to find and the number of blocks that the miner claims to find, then the investor would know that the miner is corrupt. Observers can know for certain if the distribution of mining payments is fair.

What are the gotchas involved?

- When more and more mining computers join the network, the hashrate goes into very high values (petahashes / second). Currently, in order to represent a very large number, you would need to use a floating point number type. Performing calculations with floating point numbers results in imprecise approximations. Thus, it would be desirable if the hashrate is represented through high precision (eg. 64-bit) integers. Tracking the exact number of tokens without the possibility of unexpected variations is crucial to fairness.
- On the surface, the hashrate seems to be a fungible token, however, it's *hashing contracts* that are being considered here. As the hardware does not sustain itself, mining contracts only make economical sense if they are time delimited. When considering tokens representing time delimited hashrate contracts, it is still possible merge and split tokens, but the rules to do so are more complicated as fundamentally two tokens can only be trivially merged in the additive way if their start/end dates are identical.

Ideas for the token

- The token carries three numbers: the hashrate, the starting block height (inclusive), and the ending block height. Dividends are paid on a schedule defined in the token.
- It is probably overkill to have such complicated token fungibility rules being part of the core Tokeda specification. Instead, we make it simple by having a token system that addresses a broad range of use cases and that does not require protocol changes. Tokeda is laying out conventions that issuers can leverage, and the market decides whether deploying custom logic to validate the integrity of issuers is worth the effort if an issuer puts forward non-standard fungibility rules.
- It seems overkill to even try to have such byzantine token merge/split rules put into the specification of Tokeda. Instead, we should have a broad "non-standard" token valuation class that issuers can leverage. Then, it's up to the market of observers to decide whether deploying the custom logic to validate the token values is worth the effort (which depends on the overall market attractiveness of the token itself).

Name registry

A social network company wants to deliver a lifestyle app on mobile. However, to avoid having all the good usernames to be immediately squatted and sitting unused, the company wants its users to be able to trade among themselves the ownership of the usernames of its own

platform. In order to do this, the company issue a *serial* token, where each token is associated to a unique username (rather than a numerical amount like most tokens).

What is the added value of the blockchain?

- The resources available to the company launching the social network can be low at the beginning, and the economic value of its tokens can be near-zero as well. If open source building blocks are available, then the company can delegate both the user authentication problem to the Bitcoin *stack* (e.g. re-using multisig), but also delegate the trading of its own usernames to the Bitcoin network with near zero investment on the company side.
- The company can establish minimal prices, expressed in Bitcoin, to claim short usernames. Even modest prices can deter bad behaviors from users who would otherwise squat the “best” usernames without doing anything productive with it.
- Pushing further into the idea of mitigating cyber-squatting, the company might require yearly payments to be made. The amount might be symbolic, but because payments must be made, it ensures that dead users (who might actually be dead) ultimately return their usernames to general use after a time (possibly a decade).
- The pseudo-anonymous nature of Bitcoin can be a natural fit for the social network if the company is inclined into providing a high degree of free speech - at least as high as the regulation permits in the jurisdiction where the company is established.
- By having users identified as Bitcoin token holders, the users benefit from “natural” payment identities, letting other users tip them if they behave in ways perceived as positive by the social network.

What are the gotchas involved?

- This is a *serial* token rather than a numerical one. Unlike simple numerical token, serial tokens are not fungible. The important property to preserve for the token is the unicity of attribution of each *serial* entry.

Ideas for the token

- Standardize the intent to have the serials as UTF-8 encoded characters intended for humans. TODO: *Should even the Tokeda specification cover whether the token serials are expected to be interpreted as binary (more compact, intended for machines) or as UTF-8 (intended for humans)*.

Purchasing rights on future production

A manufacturer wants to become more demand-driven to make a better use of its production capacity and increase its overall productivity. The manufacturer wants to offer its clients a kickstarter-like service where clients can secure ahead of time production capacity for the product they seek to acquire. In order to convince its clients to even participate into such a scheme that fundamentally offsets the financial risk from the manufacturer to its clients, the

manufacturer wants to put forward a very high degree of transparency where the market can witness that the manufacturer honors its promises, and thus that the manufacturer is worthy of the trust of its clients.

What is the added value of a blockchain?

- There is a social element to selling production capacity. The more the market is pressuring capacity-wise the manufacturer, the more valuable becomes a security on future production. For the manufacturer, being provably transparent can be effective in building the trust it takes to get to offset the financial risks to its clients.
- Unlike Kickstarter, tradeable tokens allow the emergence of market players who specializes in putting the correct price on future production capacity. Making such prices easily accessible to the manufacturer is important to help the manufacturer itself decide when to invest to increase its production capacity.
- Supply chains are complex and heterogeneous. Thus, having not one but a whole chain of manufacturers, wholesalers and retailers relying a shared blockchain has strong positive network effect. As the manufacturing ecosystem grows more integrated into the blockchain, the industry as a whole can become more market-driven.

What are the gotchas involved?

- Tokens on future production capacities are *partly* fungible. Indeed, we have to take into account the physical constraints of the supply chain themselves. For example products may only be shipped by cartons of 20 units, pallets of 60 units or containers of 900 units; and pricing payments or refunds may be issued if the tokens are split or merged as the packaging costs may vary depending of the number of units involved.

Ideas for the token

- The manufacturer should update the token state to advertise the fact that the units have been produced and delivered according to the plan. However, the manufacturer also wants the token users themselves, its clients, to publicly confirm that indeed, they have received the goods. In order to do that, the manufacturer can put secret tokens on its units that can be redeemed upon reception for bitcoins. Thus clients have an incentive to publicly claim the goods delivered upon reception.
- The users who have secured future production capacity might themselves also be issuers down the supply chain. Thus, by advertising their own capacity to secure their supplies, using a shared identity as users and issuers, those dependent manufacturers can generate more market trust for their own production, hence generating a virtuous market signal which help them, in turn, to finance their own production capacity.

Aerospace parts

An OEM (original equipment manufacturer) wants to manufacture and distribute parts to be mounted on commercial aircrafts. Due to the strict aerospace regulations, each airline ultimately

mounting the parts on their aircraft must verify that the part is really originating from the OEM. Yet, the airlines themselves should also be able to trade parts. The OEM issues a token that tracks each part through its serial number.

What is the added value of the blockchain?

- Security in aerospace is paramount; and the actors have to operate complex supply chains which involve producing / moving / repairing many parts all the time. Bitcoin can provide a superior form of auditability of the traceability records themselves associated to every part.
- IT systems in aerospace are fragmented, and despite having decades of experiences at internal collaborations, most actors are still relying on emails and phones to “trade” their parts. This approach is particularly inconvenient to cope with AOG (aircraft on ground) incidents which require immediate supply chain action in order to avoid delaying passengers. Bitcoin could provide a faster way for actors to transact and act on the resolution of their AOG incidents.
- Many actors have still a double process where there are both a digital trail for parts but also a paper trail. Many actors do not trust IT systems enough not to lose traceability information. For any given equipment, losing its certificate basically means that the equipment cannot be serviced any more which literally eliminates its market value (a part without its certifications is nothing but a random scrap of metal). The immutability of the blockchain can be highly appealing to actors to make sure their certificate does not get lost.

What are the gotchas involved?

- Aerospace being a highly regulated environment, the issuer needs to be able to vet users, with the possibility of denying them further interactions with other users (e.g. a company that is not compliant with aerospace standards and that loses its certification after an inspection from a governing authority).
- Users should be identified as such; i.e. it should be provable that it is the same entity that possesses a list of tokens. Indeed, in aerospace, the reputation of the actors play an important part. Some companies maintain their equipment better than others, and this strongly affects the market dynamics and the willingness to trade in the first place.
- In a first approximation, a part can be defined by its part number (specific of a given OEM), by its remaining flight hours, flight cycles and its serviceable / unserviceable status. The token would have to carry those attributes to be of any real use.
- The *standard exchange* is a popular aerospace transaction mechanism where two actors exchange parts that are compatible, one of them serviceable, the other one unserviceable. The recipient of the serviceable part is normally buying a service from the other actor and paying for it. Yet, as the unserviceable part might have more remaining flight hours and flight cycle, the payment can actually go in the other direction, and it’s the recipient of the serviceable part who ends up being paid.

Ideas for the token

- The rules that govern the tradability of assets are too obscure and too byzantine to be part of the Tokeda specification and probably even part of the specification published by the issuer itself. In real life, it would even be plausible that some rare edge cases would be manually accepted or declined by the issuer; hence a token where the issuer grants itself some operational latitude, precisely to deliver a better service, more suitable to the aerospace environment.
- The ability to have plain text information in the tokens would probably be desirable. Indeed, highly valuable components (say 100k USD) tend to be complex and sometime come with quirks of their own. The possibility to annotate the tokens in plain text (and paying for it) would be desirable.

TODO: Brendan Lee has a B2C aerospace use case at
https://docs.google.com/document/d/1JmkaZ6nmjgzy5vi37h9ad7NbwGyU3fNZV0hJxctlbR8/edit

Pharmaceutical drugs

A pharmaceutical company wants to eliminate counterfeits of its life-saving drugs that are introduced by bad actors in the distribution chain. The company puts a token on every box with can be redeemed by the patient who finally intends to consume the drug. The token can be checked by the patient to eliminate the need to trust all intermediaries, trusting only the original pharmaceutical company.

What is the added value of the blockchain?

- In Africa, while estimates vary, about 40% of the drugs distributed are counterfeit, even for life-critical medications to counter lethal pathologies like AIDS. While certain countries have succeeded in establishing very secure supply chains, some other countries have not. Bitcoin does not rely on traditional authorities to be deployed. A token-based solution can be made very resilient against bad actors who typically act as supply intermediaries.
- As the targeted population can be poor, the solution should not depend on any equipment beyond a smartphone; and should be made convenient enough to be accessible to people who are largely disabled and/or uneducated, possibly relying on voice feedback in rather “niche” languages. A Bitcoin-based solution is very appealing because of its open-innovation potential where the issuer can delegate to the market the implementation of UX suitable to the need of the users, instead of rolling out its own solution.

Ideas for the token

- The pharma company puts a secret token within each box it produces. When the box is opened, breaking a seal, the token is revealed, and the patient can scan the token to claim back a small quantity of Bitcoin. When claiming the token, the mobile app checks that the token properly originates from the pharma company and that this token hasn’t been claimed yet. If the token has already been claimed or does not match the expected

originating company, the mobile app warns the patient that the box is a counterfeit. The small amount of Bitcoin to be claimed is there as an incentive to make sure that all tokens are claimed by users (or at least a substantial portion of them), making it very risky for the distributor to cheat and introduce counterfeit.

- More complex setups can be considered to have the intermediaries performing similar validation at every step of the distribution chains; for example, assuming that there won't be more than 4 hops from the pharma company to the patient, which might actually be a desirable property from the perspective of the pharma company who may wish to put limits on the number of intermediaries. A scheme based on a reverse sequence of Sha256 gradually disclosed could play this role.

Secure software package manager

Modern software is increasingly dependent on package managers (NPM, Nugget, Cargo). However, package managers are security critical because they are relied on to distribute software, which can be turned into malicious software if the package manager is compromised.

What is the added value of the blockchain?

- The most difficult challenge faced by a package manager is coping with security problems on the user's end (the package authors): users getting hacked despite 2FA, CDN servers getting hacked, etc. The Bitcoin ecosystem which features hardware wallets delivers a degree of authorization security which is very hard to replicate in practice anywhere else. The package manager turns to the blockchain to benefit from all the ambient security-in-depth processes.
- The second most difficult challenge faced by a package manager is copying with security problems on its own machines. Again, Bitcoin miners already face similar problems (or will in the future), but only as data corruption problems – malevolent or not - are fatal block-validation-wise, it is safe to assume that miners do their best to remain very secure.
- In the case where a software author gets hacked - which would be akin to getting his Bitcoin stolen in the present case – the issuer should be able to mark a package already published as malicious to prevent the attacker from doing further damage.

Ideas for the token

- BitTorrent works well for download-many situations. Thus, in case of software package distribution, it would be logical to only keep the hashes of the packages within the UTX and to offload entirely the actual distribution of the data to BitTorrent.

TODO: *an alternate use case would be the distribution of software licenses where the proprietary software looks for a token metadata in order to see if the local user claims check out. Then the local copy of the software embeds the identity of the licensee within each file produced by the software.*

Anonymous secure text messaging network

An organization wants to create a social network that features an anonymous secure text message network that features end-to-end encryption for its users. The organization wants a design that is fundamentally peer-to-peer but building an efficient low-latency peer-to-peer network is non-trivial, as there are many challenges involved.

What is the added value of the blockchain?

- Most of the P2P challenges are already solved by Bitcoin such as finding an entry point to the network and establishing incentives for fast transaction relay across nodes. The organization does not want to spend resources on re-inventing solutions.
- The Tokeda metadata layer offers the possibility have stateless Telegram-like clients which rely on no local storage at all. Instead, a user only needs a device that carries a single secret and all its exchanges can be inferred, based on a deterministic wallet scheme.
- Each software author should be able to claim a username that is guaranteed to remain unique.

What are the gotchas involved?

- The issuer needs a token when the issuer's fee is essentially driven by the amount of data pushed by users. The miner's fee only reflects the UTXO cost, not the UTX costs. If UTX costs are varying over time, the issuer needs to be able to update its pricing policy (byte-wise) to relay messages.
- Text is lightweight and feels like a plausible use-case as it does not require larger transactions than what the present consensus allows. For images or even videos, and any larger file, the solution might lie in pushing the seed of the media content to a BitTorrent-like protocol.
- Considering that storing (lots of) data is the intent behind this use case, the issuer needs fine-grained capabilities to elect which UTX entries are worth preserving. Unlike many other tokens, some entries are expected to be much larger than others.

Ideas for the token

- SPV wallet implementations would probably require some logic to give granular control to the users over how much history they want to keep. The software would then reorganize their entries accordingly to the depth of history that is intended to be kept.
- To address upfront the pricing challenge associated with a use case resolving around data storage, the issuer could directly issue a pricing a per byte price for a duration of N years, and let users signal their requirements through the issuer's fee.

Annexes

In this section, we cover miscellaneous points that are somewhat related to the present specification.

OpCodes stay forever

Proposals, such as OP_GROUP, have been made to add basic metadata forwarding capabilities to Bitcoin, however, as we will prove by reviewing a variety of use cases, OP_GROUP is not a viable solution for many of the real-world use cases that we have been able to identify. As OP_GROUP requires a consensus change, it raises the question: why introducing something to be supported forever by BCH while it does not address most of the important token use cases?

OpCodes, once introduced, must be supported forever. Both x86 and ARM are ample empirical proof of that. Removing OP_GROUP will effectively destroy peoples' wealth if people are relying on it. Bitcoin miners have a lot to lose if they ever end-up generating such a situation. No miner would want to be the first miner to destroy the wealth of a Bitcoin holder; even it's not Bitcoin *per se* that is being destroyed. Thus, once OP_GROUP is introduced, the only logical option for any honest miner is to support this feature forever. While OP_GROUP can be introduced, it certainly cannot be removed afterward, not because it's technically not possible, but because the honest miners cannot afford to risk losing user's trust by intentionally destroying their wealth.

Transaction size will remain capped

We argue that transactions will remain capped to relatively moderate sizes in the future. Indeed, while the processing cost of a given transaction is essentially linear in the transaction size, the transaction is not fundamentally "chunkable" in smaller pieces of data which can be treated in full isolation from one another. Thus, to avoid creating performance choking points in the block validation process, every processing piece within a block needs to have clear upper bounds both in CPU time and in memory footprint. Specialized hardware dedicated to transaction validation may be available to the miners, but in all likelihood they won't be in the ecosystem at large. A large company may want to pay dividends to its 50 million shareholders, but doing so by emitting a 8GB "atomic" transaction would most likely choke entire parts of the IT landscape that surrounds Bitcoin and which may be querying transactions. Bitcoin should not put a *scalability challenge* on every piece of software that touches its blockchain. It will remain costly to scale software components in practice - no matter how good the hardware gets, because clock frequency won't improve much.