Posted May 10, 2016 by Gideon Greenspan (http://www.multichain.com/blog/author/gdg/) in Private blockchains (http://www.multichain.com/blog/category/private-blockchains/).

Four genuine blockchain use cases

Where shared ledgers add real value in enterprise IT

Almost a year after first releasing MultiChain (http://www.multichain.com/), we've learnt a huge amount about how blockchains, in a private and non-cryptocurrency sense, can and cannot be applied to real-world problems. Allow me to share what we know so far.

To begin with, the first idea that we (and many others) started with, appears to be wrong. This idea, inspired by bitcoin directly, was that private blockchains (or "shared ledgers") could be used to directly settle the majority of payment and exchange transactions in the finance sector, using on-chain tokens to represent cash, stocks, bonds and more.

This is perfectly workable on a technical level, so what's the problem? In a word, **confidentiality**. If multiple institutions are using a shared ledger, then every institution sees every transaction on that ledger, even if they don't immediately know the real-world identities of the parties involved. This turns out to be a huge issue, both in terms of regulation and the commercial realities of inter-bank competition. While various strategies are available or in development for mitigating this problem, none can match the simplicity and efficiency of a centralized database managed by a trusted intermediary, which maintains full control over who can see what. For now at least, it seems that large financial institutions prefer to keep most transactions hidden in these intermediary databases, despite the costs involved.

I base this conclusion not only on our own experience, but also on the direction taken by several prominent startups whose initial goal was to develop shared ledgers for banks. For example, both R3CEV and Digital Asset are now working on "contract description languages", in Corda (https://gendal.me/2016/04/05/introducing-r3-corda-a-distributed-ledger-designed-for-financial-services/) and DAML (https://digitalasset.com/press/introducing-daml.html) respectively (earlier examples include MLFi (https://www.lexifi.com/product/technology/contract-description-language) and Ricardian Contracts (http://iang.org/papers/ricardian_contract.html)). These languages allow the conditions of a complex financial contract to be represented formally and unambiguously in a computer readable format, while avoiding the shortcomings (http://www.multichain.com/blog/2015/11/smart-contracts-good-bad-lazy/) of Ethereum-style general purpose computation. Instead, the blockchain plays only a supporting role, storing or notarizing the contracts in encrypted form, and performing some basic duplicate detection. The actual contract execution does not take place on the blockchain – rather, it is performed only by the contract's counterparties, with the likely addition of auditors and regulators.

In the near term, this is probably the best that can be done, but where does it leave the broader ambitions for permissioned blockchains? Are there other applications for which they can form a more significant part of the puzzle?

This question can be approached both theoretically and empirically. Theoretically, by focusing on the key differences between blockchains and traditional databases, and how these inform the set of possible use cases. And in our case, empirically, by categorizing the real-world solutions being built on MultiChain today. Not surprisingly, whether we focus on theory or practice, the same classes of use case arise:

- Lightweight financial systems.
- Provenance tracking.
- Interorganizational record keeping.
- Multiparty aggregation.

Before explaining these in detail, let's recap the theory. As I've discussed before (http://www.multichain.com/blog/2016/03/blockchains-vs-centralized-databases/), the two most important differences between blockchains and centralized databases can be characterized as follows:

1. **Disintermediation**. Blockchains enable multiple parties who do not fully trust each other to safely and directly share a single database without requiring a trusted intermediary.
2. **Confidentiality**: All participants in a blockchain see all of the transactions taking place. (Even if we use pseudonymous addresses and advanced cryptography to hide some aspects of those transactions, a blockchain will always leak more information than a centralized database.)

In other words, blockchains are ideal for shared databases in which *every user* is able to *read* everything, but *no single user* controls who can *write* what. By contrast, in traditional databases, a single entity exerts control over all read and write operations, while other users are entirely subject to that entity's whims. To sum it up in one sentence:

Blockchains represent a trade-off in which disintermediation is gained at the cost of confidentiality.

In examining the four types of use case below, we'll repeatedly come back to this core trade-off, explaining why, in each case, the benefit of disintermediation outweighs the cost of reduced confidentiality.

Lightweight financial systems

Let's start with the class of blockchain applications that will be most familiar, in which a group of entities wishes to set up a financial system. Within this system, one or more scarce assets are transacted and exchanged between those entities.

In order for *any* asset to remain scarce, two related problems must be solved. First, we must ensure that the same unit of the asset cannot be sent to more than one place (a "double spend"). Second, it must be impossible for anyone to create new units of the asset on a whim ("forgery"). Any entity which could do either of these things could steal unlimited value from the system.

A common solution to these problems is physical tokens, such as metal coins or securely printed paper. These tokens trivially solve the problem of double spending, because the rules of physics (literally) prevent one token from being in two places at the same time. The problem of forgery is solved by making the token extremely difficult to manufacture. Still, physical tokens suffer from several shortcomings which can render them impractical:

- As pure bearer assets, physical tokens can be stolen with no trace or recourse.
- They are slow and costly to move in large numbers or over long distances.
- It is tricky and expensive to create physical tokens that cannot be forged.

These shortcomings can be avoided by leaving physical tokens behind, and redefining asset ownership in terms of a ledger managed by a trusted intermediary. In the past, these ledgers were based on paper records, and today they tend to run on regular databases. Either way, the intermediary enacts a transfer of ownership by modifying the ledger's content, in response to an authenticated request. Unlike settlement with physical tokens, questionable transactions can quickly and easily be reversed.

So what's the problem with ledgers? In a nutshell, **concentration of control**. By putting so much power in one place, we create a significant security challenge, in both technical and human terms. If someone external can hack into the database, they can change the ledger at will, stealing others' funds or destroying its contents completely. Even worse, someone *on the inside* could corrupt the ledger, and this kind of attack is hard to detect or prove. As a result, wherever we have a centralized ledger, we must invest significant time and money in mechanisms to maintain that ledger's integrity. And in many cases, we require ongoing verification using batch-based reconciliation between the central ledger and those of each of the transacting parties.

Enter the blockchain (or "shared ledger"). This provides the benefits of ledgers without suffering from the problem of concentration. Instead, each entity runs a "node" holding a copy of the ledger and maintains full control over its own assets, which are protected by private keys. Transactions propagate between nodes in a peer-to-peer fashion, with the blockchain ensuring that consensus is maintained. This architecture leaves no central attack point through which a hacker or insider could corrupt the ledger's contents. As a result, a digital financial system can be deployed more quickly and cheaply, with the added benefit of automatic reconciliation in real time.

So what's the downside? As discussed earlier, all participants in a shared ledger see all of the transactions taking place, rendering it unusable in situations where confidentiality is required. Instead, blockchains are suitable for what I call *lightweight* financial systems, namely those in which the economic stakes or number of participants is relatively low. In these cases, confidentiality tends to be less of an issue – even if the participants pay close attention to what each other are doing, they won't learn much of value. And it is precisely *because* the stakes are low that we prefer to avoid the hassle and cost of setting up an intermediary.

Some obvious examples of lightweight financial systems include: crowdfunding, gift cards, loyalty points and local currencies – especially in cases where assets are redeemable in more than one place. But we are also seeing use cases in the mainstream finance sector, such as peer-to-peer trading between asset managers who are not in direct competition. Blockchains are even being tested as *internal* accounting systems, in large organizations where each department or location must maintain control of its funds. In all these cases, the lower cost and friction of blockchains provides an immediate benefit, while the loss of confidentiality is not a concern.

Provenance tracking

Here's a second class of use case that we repeatedly hear from MultiChain's users: tracking the origin and movement of high-value items across a supply chain, such as luxury goods, pharmaceuticals, cosmetics and electronics. And equally, critical items of documentation such as bills of lading or letters of credit. In supply chains stretching across time and distance, all of these items suffer from counterfeiting and theft.

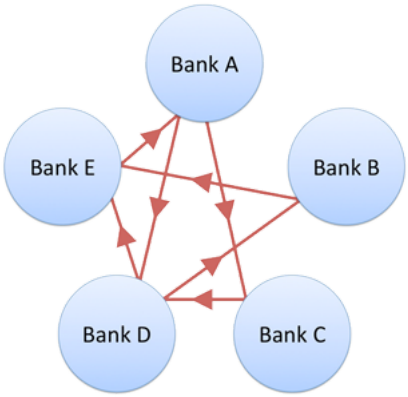
The problem can be addressed using blockchains in the following way: when the high-value item is created, a corresponding digital token is issued by a trusted entity, which acts to authenticate its point of origin. Then, every time the physical item changes hands, the digital token is moved in parallel, so that the real-world chain of custody is precisely mirrored by a chain of transactions on the blockchain.

If you like, the token is acting as a virtual "certificate of authenticity", which is far harder to steal or forge than a piece of paper. Upon receiving the digital token, the final recipient of the physical item, whether a bank, distributor, retailer or customer, can verify the chain of custody all the way back to the point of origin. Indeed, in the case of documentation such as bills of lading, we can do away with the physical item altogether.

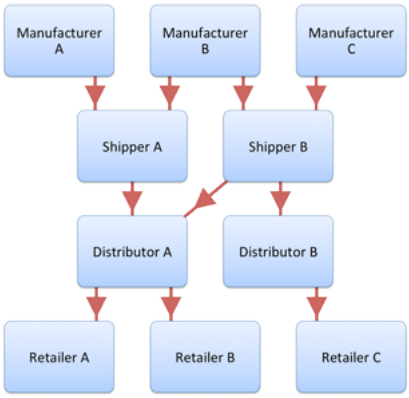
While all of this makes sense, the astute reader will notice that a regular database, managed (say) by an item's manufacturer, can accomplish the same task. This database would store a record of the current owner of each item, accepting signed transactions representing each change of ownership, and respond to incoming requests regarding the current state of play.

So why use a blockchain instead? The answer is that, for this type of application, there's a benefit to distributed trust. No matter where a centralized database is held, there will be people in that place who have the ability (and can be bribed) to corrupt its contents, marking forged or stolen items as legit. By contrast, if provenance is tracked on a blockchain belonging collectively to a supply chain's participants, no individual entity or small group of entities can corrupt the chain of custody, and end users can have more confidence in the answers they receive. As a bonus, different tokens (say for some goods and the corresponding bill of lading) can be safely and directly exchanged, with a two-way swap guaranteed (http://www.multichain.com/blog/2015/09/delivery-versus-payment-blockchain/) at the lowest blockchain level.

What about the problem of confidentiality? The suitability of blockchains for supply chain provenance is a happy result of this application's simple pattern of transactions. In contrast to financial marketplaces, most tokens move in a single direction, from origin to endpoint, without being repeatedly traded back-and-forth between the blockchain's participants. If competitors rarely transact with each other (e.g. toy manufacturer to toy manufacturer, or retailer to retailer), they cannot learn each others' blockchain "addresses" and connect those to real-world identities. Furthermore, the activity can be easily partitioned into multiple ledgers, each representing a different order or type of good.



Asset movements
in financial marketplaces



Asset movements
in supply chains

Interorganizational record keeping

Both of the previous use cases are based on tokenized assets, i.e. on-chain representations of an item of value transferred between participants. However there is a second group of blockchain use cases which is not related to assets. Instead, the chain acts as a mechanism for collectively recording and notarizing *any* type of data, whose meaning can be financial or otherwise.

One such example is an audit trail of critical communications between two or more organizations, say in the healthcare or legal sectors. No individual organization in the group can be trusted with maintaining this archive of records, because falsified or deleted information would significantly damage the others. Nonetheless it is vital that all agree on the archive's contents, in order to prevent disputes.

To solve this problem, we need a shared database into which all of the records are written, with each record accompanied by a timestamp and proof of origin. The standard solution would be to create a trusted intermediary, whose role is to collect and store the records centrally. But blockchains offer a different approach, giving the organizations a way to jointly manage this archive, while preventing individual participants (or small groups thereof) from corrupting it.

One of the most enlightening conversations I've had in the past two years was with Michael Mainelli (https://en.wikipedia.org/wiki/Michael_Mainelli) of Z/Yen (http://www.zyen.com/). For 20 years his company has been building systems in which multiple entities collectively manage a shared digital audit trail, using timestamping, digital signatures and a round robin consensus scheme. As he explained the technical details of these systems, it became clear that they are permissioned blockchains in every respect. In other words, there is nothing new about using a blockchain for interorganizational recordkeeping – it's just that the world has finally become aware of the possibility.

In terms of the actual data stored on the blockchain, there are three popular options:

- **Unencrypted data**. This can be read by every participant in the blockchain, providing full collective transparency and immediate resolution in the case of a dispute.
- **Encrypted data**. This can only be read by participants with the appropriate decryption key. In the event of a dispute, anyone can reveal this key to a trusted authority such as a court, and use the blockchain to prove that the original data was added by a certain party at a certain point in time.
- **Hashed data**. A "hash (https://en.wikipedia.org/wiki/Hash_function)" acts as a compact digital fingerprint, representing a commitment to a particular piece of data while keeping that data hidden. Given some data, any party can easily confirm if it matches a given hash, but inferring data *from* its hash is computationally impossible. Only the hash is placed on the blockchain, with the original data stored off-chain by interested parties, who can reveal it in case of a dispute.

As mentioned earlier, R3CEV's Corda product has adopted this third approach, storing hashes (http://www.coindesk.com/barclays-smart-contracts-templates-demo-r3-corda/) on a blockchain to notarize contracts between counterparties, without revealing their contents. This method can be used both for computer-readable contract descriptions, as well as PDF files containing paper documentation.

Naturally, confidentiality is not an issue for interorganizational record keeping, because the entire purpose is to create a shared archive that all the participants can see (even if some data is encrypted or hashed). Indeed in some cases a blockchain can help manage access to confidential off-chain data, by providing an immutable record of digitally signed access requests. Either way, the straightforward benefit of disintermediation is that no additional entity must be created and trusted to maintain this record.

Multiparty aggregation

Technically speaking, this final class of use case is similar to the previous one, in that multiple parties are writing data to a collectively managed record. However in this case the motivation is different – to overcome the infrastructural difficulty of combining information from a large number of separate sources.

Imagine two banks with internal databases of customer identity verifications. At some point they notice that they share a lot of customers, so they enter a reciprocal sharing arrangement in which they exchange verification data to avoid duplicated work. Technically, the agreement is implemented using standard master–slave data replication (https://en.wikipedia.org/wiki/Replication_(computing)#Database_replication), in which each bank maintains a live read-only copy of the other's database, and runs queries in parallel against its own database and the replica. So far, so good.

Now imagine these two banks invite three others to participate in this circle of sharing. Each of the 5 banks runs its own master database, along with 4 read-only replicas of the others. With 5 masters and 20 replicas, we have 25 database instances in total. While doable, this consumes noticeable time and resources in each bank's IT department.

Fast forward to the point where 20 banks are sharing information in this way, and we're looking at 400 database instances in total. For 100 banks, we reach 10,000 instances. In general, if every party is sharing information with every other, the total number of database instances grows with the square of the number of participants. At some point in this process, the system is bound to break down.

So what's the solution? One obvious option is for all of the banks to submit their data to a trusted intermediary, whose job is to aggregate that data in a single master database. Each bank could then query this database remotely, or run a local read-only replica within its own four walls. While there's nothing wrong with this approach, blockchains offer a cheaper alternative, in which the shared database is run directly by the banks which use it. Blockchains also bring the added benefit of redundancy

(https://en.wikipedia.org/wiki/Redundancy_(engineering)) and failover (https://en.wikipedia.org/wiki/Failover) for the system as a whole.

It's important to clarify that a blockchain is not acting just as a distributed database like Cassandra (http://cassandra.apache.org/) or RethinkDB (https://www.rethinkdb.com/). Unlike these systems, each blockchain node enforces a set of rules which prevent one participant from modifying or deleting the data added by another. Indeed, there still appears to be some confusion about this – one recently released blockchain platform can be broken by a single misbehaving node. In any event, a good platform will also make it easy to manage networks with thousands of nodes, joining and leaving at will, if granted the appropriate permissions.

Although I'm a little skeptical of the oft-cited connection between blockchains and the Internet of Things (https://en.wikipedia.org/wiki/Internet_of_Things), I think this might be where a strong such synergy lies. Of course, each "thing" would be too small to store a full copy of the blockchain locally. Rather, it would transmit data-bearing transactions to a distributed network of blockchain nodes, who would collate it all together for further retrieval and analysis.

Conclusion: Blockchains in Finance

I started this piece by questioning the initial use case envisioned for blockchains in the finance sector, namely the bulk settlement of payment and exchange transactions. While I believe this conclusion is becoming common wisdom (with one notable exception (https://chain.com/)), it does not mean that blockchains have no other applications in this industry. In fact, for each of the four classes of use case outlined above, we see clear applications for banks and other financial institutions. Respectively, these are: small trading circles, provenance for trade finance, bilateral contract notarization and the aggregation of AML/KYC data.

The key to understand is that, architecturally, our four classes of use case are not *specific* to finance, and are equally relevant to other sectors such as insurance, healthcare, distribution, manufacturing and IT. Indeed, private blockchains should be considered for any situation in which two or more organizations need a shared view of reality, and that view does not originate from a single source. In these cases, blockchains offer an alternative to the need for a trusted intermediary, leading to significant savings in hassle and cost.

Please post any comments on LinkedIn (https://www.linkedin.com/pulse/four-genuine-blockchain-use-cases-gideon-greenspan).

Recent Posts

- Explaining zero knowledge blockchains (http://www.multichain.com/blog/2016/11/explaining-zero-knowledge-blockchains/)
- First MultiChain Partners Announced (http://www.multichain.com/blog/2016/10/first-multichain-partners-announced/)
- Introducing MultiChain Streams (http://www.multichain.com/blog/2016/09/introducing-multichain-streams/)
- Announcing the new MultiChain wallet (http://www.multichain.com/blog/2016/07/announcing-the-new-multichain-wallet/)
- Smart contracts and the DAO implosion (http://www.multichain.com/blog/2016/06/smart-contracts-the-dao-implosion/)
- Four genuine blockchain use cases (http://www.multichain.com/blog/2016/05/four-genuine-blockchain-use-cases/)
- Beware the impossible smart contract (http://www.multichain.com/blog/2016/04/beware-impossible-smart-contract/)
- Blockchains vs centralized databases (http://www.multichain.com/blog/2016/03/blockchains-vs-centralized-databases/)
- Recent Features and 2016 Roadmap (http://www.multichain.com/blog/2016/03/recent-features-2016-roadmap/)
- Moving on from big blockchains (http://www.multichain.com/blog/2016/01/moving-on-from-big-blockchains/)
- Avoiding the pointless blockchain project (http://www.multichain.com/blog/2015/11/avoiding-pointless-blockchain-project/)
- Smart contracts make slow blockchains (http://www.multichain.com/blog/2015/11/smart-contracts-slow-blockchains/)
- Smart contracts: The good, the bad and the lazy (http://www.multichain.com/blog/2015/11/smart-contracts-good-bad-lazy/)
- Private blockchains are more than "just" shared databases (http://www.multichain.com/blog/2015/10/private-blockchains-shared-databases/)
- Delivery versus payment on a blockchain (http://www.multichain.com/blog/2015/09/delivery-versus-payment-blockchain/)
- Ending the bitcoin vs blockchain debate (http://www.multichain.com/blog/2015/07/bitcoin-vs-blockchain-debate/)

Recent Comments

- Patrick Lismore on Moving on from big blockchains (http://www.multichain.com/blog/2016/01/moving-on-from-big-blockchains/#comment-79)
- Vitalik Buterin (http://ethereum.org) on Smart contracts: The good, the bad and the lazy (http://www.multichain.com/blog/2015/11/smart-contracts-good-bad-lazy/#comment-60)
- Nexus on Private blockchains are more than "just" shared databases (http://www.multichain.com/blog/2015/10/private-blockchains-shared-databases/#comment-49)
- Gideon Greenspan on Private blockchains are more than "just" shared databases (http://www.multichain.com/blog/2015/10/private-blockchains-shared-databases/#comment-33)
- romanix (http://www.orderbook.info) on Private blockchains are more than "just" shared databases (http://www.multichain.com/blog/2015/10/private-blockchains-shared-databases/#comment-32)

Archives

- November 2016 (http://www.multichain.com/blog/2016/11/)
- October 2016 (http://www.multichain.com/blog/2016/10/)
- September 2016 (http://www.multichain.com/blog/2016/09/)
- July 2016 (http://www.multichain.com/blog/2016/07/)

- June 2016 (http://www.multichain.com/blog/2016/06/)
- May 2016 (http://www.multichain.com/blog/2016/05/)
- April 2016 (http://www.multichain.com/blog/2016/04/)
- March 2016 (http://www.multichain.com/blog/2016/03/)
- January 2016 (http://www.multichain.com/blog/2016/01/)
- November 2015 (http://www.multichain.com/blog/2015/11/)
- October 2015 (http://www.multichain.com/blog/2015/10/)
- September 2015 (http://www.multichain.com/blog/2015/09/)
- July 2015 (http://www.multichain.com/blog/2015/07/)