LONG FINANCE

Auditing Mutual Distributed Ledgers (aka Blockchains)

A Foray Into Distributed Governance & Forensics

DasCoin™
The Currency of Trust

November 2017

Auditing Mutual Distributed Ledgers (aka Blockchains)

A Foray Into Distributed Governance & Forensics

Professor Michael Mainelli
Executive Chairman, Z/Yen Group

Matthew Leitch
Z/Yen Group

Foreword

We are thrilled to sponsor this much needed project to promote serious thinking about the creation of an international auditing standard for Mutual Distributed Ledgers. This is a very interesting challenge and one that has so far been almost entirely overlooked.

The audit issues around multi-site, distributed data storage and processing, controlled by sophisticated cryptographic means, are complex. To investigate them, the project team has combined desk research with their own expert knowledge of Mutual Distributed Ledgers and auditing techniques. They also hosted a day-long symposium that gathered together experts to discuss the challenges and possible solutions. It was an absorbing event that we were pleased to attend. The project work was carried out during September and October 2017 with financial support from DasCoin.

We are pleased to share this final project report with you and hope that you find it helpful and inspiring. We intend for it to be an important contribution in an area that needs to be dealt with properly if the technology of Mutual Distributed Ledgers is to achieve its full potential.

Michael Mathias
Founder & CEO, DasCoin

Contents

Foreword 2
Project Overview 4
The Symposium 5
Session 1 Auditing Distributed Ledgers 8
1.1 Applications Using Mutual Distributed Ledgers 8
1.2 Auditing 9
1.3 MDL Risk Assessment 9
1.4 Audit Tools and Techniques 12
1.5 Quantity Of Audit Work 14
Session 2 Auditing Distributed Data 15
2.1 Distributed Copies of the Data Store 15
2.2 Date Cut-Offs 15
2.3 Reconciling Differences 16
2.4 Evaluating Technical Designs 17
2.5 Distributed Controls Operated by People 18
Session 3 Auditing Distributed 'Smarts' 20
3.1 Embedded Code 20
3.2 Audit Aspects Of Smart Contracts 22
3.3 Quality Assurance for Code 22
3.4 Money Supply Algorithms 23
3.5 High-Tech Audit Techniques 24
Session 4 Auditing Consortium MDL Systems 26
4.1 Consortium MDL Systems 26
4.2 Governance Needed 26
Other Potential Audit Issues 29
Suggestions for the Profession 31
Conclusion 35
Participants 36

Project Overview

This project, Auditing Mutual Distributed Ledgers (MDLs, aka Blockchains), was commissioned by DasCoin in the summer of 2017. Z/Yen's Long Finance project team reviewed the limited literature on the topic to produce four discussion papers that described the issues and explored some potential ways of addressing them.

On 4 October 2017, Long Finance held a symposium to explore the discussion papers in the ornate Main Reception Room at Chartered Accountants' Hall in London. The 28 participants had wide ranging backgrounds and interests. They included representatives from regulators, professional associations, external audit firms, technology companies, and academia.

This Report includes the background information chapters shown in Figure 1, along with a brief summary of the symposium discussions. The final chapter makes suggestions for the audit profession to address.

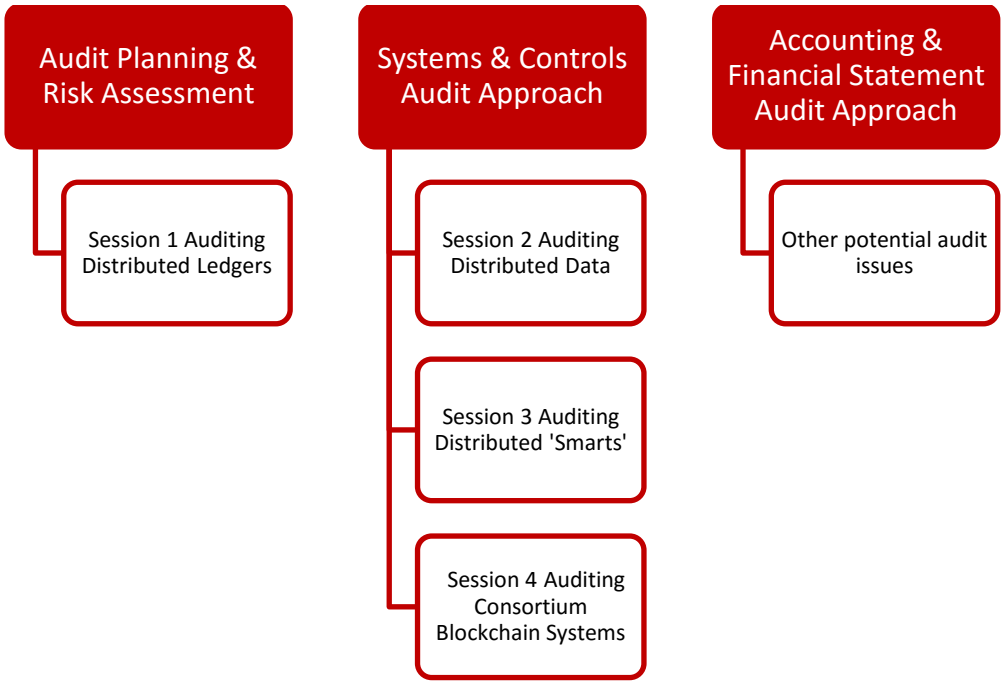

Figure 1 The coverage of the background information chapters is linked to the main parts of an audit.

The Symposium

Background

The 4 October symposium began by focusing on MDLs as "multi-organisational databases with a super audit trail". The relevant technology and approaches date back to the 1970s. Even 'smart contract' concepts date back to LISP code, capable of writing code and then executing it itself. What is new is the increasing interest and confidence in deploying MDLs.

MDLs with blockchains are not cheaper and not faster than a central database. The fundamental change is to the role of central third parties. Some relevant symposium points from the open discussion include:

- Trusted third parties are not eliminated by MDLs (though their functions are changed).
- A *rapid* growth in MDL applications has not happened and is unlikely; steady growth is more likely.
- There will be multiple technologies for building MDLs, not a single dominant one. MDLs will be different for different applications, e.g. a high-speed internet-of-things ledger versus a payment system. Applications may well combine different MDLs, e.g. identity, transaction, documentation, and payment MDLs.
- MDL implementations are slow because technical complexity is higher (say, 10 times more difficult than a traditional database) and because multiple organisations are involved project complexity is higher (say, 10 times higher). The result can be a project that is 100 times more complex than a simple database installation; and many organisations find simple database installations tough.
- MDLs are not necessarily anonymous; that depends on how they are designed.
- Mutual systems are not new. We (auditors) already rely on some systems that are not owned by anyone, such as email, and depend on other mutual systems such as TCP/IP.

Session 1 Auditing Mutual Distributed Ledgers

The first discussion session was a broad introduction to the subject with participants making initial comments rather than getting into details of audit methods or technology. Some thought that audit risk would change but not rise, while others thought the risk level would increase due to novelty or because of the distributed data stores. For the highly regulated financial sector, clear answers to a number of questions about compliance are needed before boards could confidently opt for MDL applications. That would depend very much on what the MDL would be used for and what data would be held on it.

Session 2 Auditing Distributed Data

Ideas for audit techniques that were suggested included:

- A taxonomy of MDL types – a technical taxonomy and an application taxonomy.
- The auditor to be hosting a node of the MDL and using it to get information on the state of the MDL on other nodes (though an information collector that was in contact with all or most nodes might do this better).
- Design the MDL to be auditable, with an auditability standard of some kind, e.g. ensure that timestamps record the date and time of the deal, not just when the transaction was added to a block.

Futures Wheel Group Discussions

Participants broke into four groups. Two were asked to consider the primary, secondary, and tertiary consequences of the statement: "Businesses use MDLs without concern for formal auditability". Their ideas covered both the consequences of poor control internally, such as unreliable management information and resulting flawed decisions, and the consequences of potentially qualified audit opinions, with knock-on consequences for financing.

The other two groups were asked to consider the possible consequences of the statement: "Profession refuses to provide audit opinions on MDLs." Their ideas reflected uncertainty as to how important MDLs would be and how beneficial for organisations and society. If the drive towards MDLs is not too strong, then this refusal might block the progress of MDLs, either causing a missed opportunity or perhaps avoiding wasting time on a technology that later proves to be a dead end. On the other hand, if the drive towards MDLs is stronger (a possibility the groups spent more time considering), the profession's refusal could lead to loss of business for audit firms, the rise of alternative assurance providers, and other similar consequences – unless the audit profession changed its stance in response.

Session 3 Auditing Distributed 'Smarts'

Initial contributions stressed the immaturity of 'smart contract' technology, ways of using it, and the governance of crises precipitated by such code being exploited in unexpected ways. This was followed by suggested ways to focus development so that it was beneficial and less risky, namely:

- Focus on operational efficiencies from straight through processing.
- Avoiding or deferring automation of medium and long term contracts.
- Excluding settlement from automated contracts (concerns were expressed about the volatility of Bitcoin, as an example, or the monetary effects on leverage and pooling by putting payments so far ahead into ledgers).
- Restricting the information used by the contract code to just the code already on the MDL and a time source – and avoiding other information piped in from outside, such as the LIBOR rate, Oscar winners, or the weather.

- Not trying to automate all clauses.
- Having a human 'bug out' clause so that people can suspend automated execution if they see a problem, perhaps then permitting these 'bug out' incidents to go to arbitration, mediation, or expert determination, before any legal proceedings.

Session 4 Auditing Consortium MDL Systems

The conversation began with some interesting examples of long-established commercial consortia without MDLs at this time, then moved on to consider the life cycle of consortia, and the potential advantages of applications that helped companies located in different countries (and cultures) interpret their agreements consistently. The problem of data location and data protection law was raised again.

Group Discussions

Participants were divided into four groups and asked to come up with suggestions under the heading "Towards a trained profession and an auditing standard" but were encouraged to think more widely if they wished.

This exercise generated many ideas but no obvious consensus. In the plenary discussion that followed, it emerged that some kind of collaborative, perhaps pan-institute group might be a sensible way forward.

Session 1 Auditing Distributed Ledgers

1.1 Applications Using Mutual Distributed Ledgers

An application using a Mutual Distributed Ledger (MDL) has the following properties:

- **Mutual**: meaning that all users of the application share what *appears to them* to be one data store, even if they are working in different organisations.
- **Distributed**: meaning that the shared data store is physically stored as multiple, nearly identical copies of the same file(s), held on different computers, usually in different organisations, with processes working automatically to keep those copies identical except for the most recent additions.
- **Ledger**: because the data store is a list of the details of transactions, typically in (nearly) chronological order, that can be added to, but not amended or deleted from retrospectively.

The data are stored in the form in a data chain of data blocks cryptographically linked so that they are impossible to alter retrospectively without the cooperation of other parties holding a copy of the ledger.

People may use an MDL application without holding a copy of the MDL, or they may be users that do have a copy of the MDL on a computer they use. For example, many people today have a Bitcoin 'wallet', but relatively few have the Bitcoin blockchain on their computer. By the end of August 2017, that blockchain was more than 130GB in size and growing rapidly.[1]

Distinctions should be made between a particular MDL (e.g. the set of data files used by Bitcoin), the software that operates an MDL (analogous to a DBMS[2]), and applications created using those as components (e.g. a payment method, a timestamping service, a group decision-making tool). An organisation's reliance on an MDL application might range from:

- occasional use of a timestamping service, such as Alderney's MetroGnomo[3] to secure intellectual property rights; to
- a customer incentive scheme that issues cryptocoins instead of points on a loyalty card; to
- participating in a market where all its business is transacted and recorded on an MDL, with money and securities changing hands through an MDL application, often driven by program code that is itself held within an MDL application.

[1] Data from https://blockchain.info/charts/blocks-size

[2] Database Management System

[3] See https://www.metrognomo.com/

1.2 Auditing

External financial auditing and internal auditing have a tendency to focus on finance. However, there are numerous types of audits, e.g. for certification against international or British standards, or audits trying to establish very specific properties of systems related to their security.

The traditional approach to financial auditing is not focused on the ledgers themselves. We audit financial statements (typical in external auditing) or business processes (typical in internal auditing). When auditing financial statements, we work back to accounting cycles (another name for business processes that do book-keeping), again not focusing specifically on the ledger. When auditing through an accounting cycle (or business process), we identify the stages of information processing and look at them in detail. Doing this, we identify the computer applications used and the computer environments where they reside. This is where the ledger itself comes up as an object of auditing, with work perhaps done to look at how people manage the software, as well as the hardware on which the software runs.

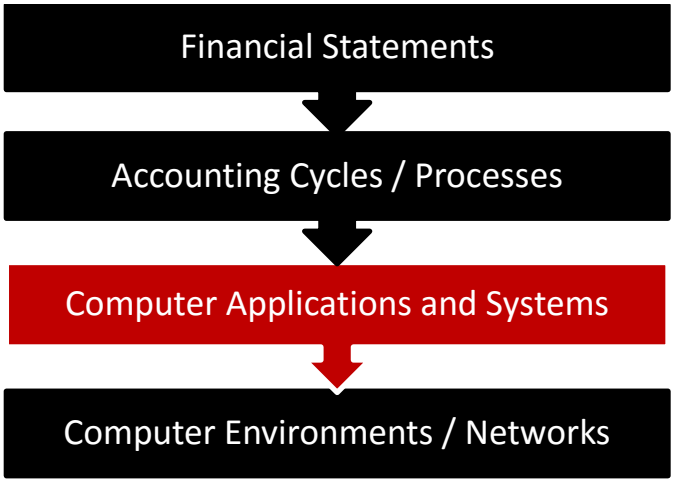

Figure 2 MDLs become an object of audit at the level of computer applications and systems, but only because of their support for accounting cycles and, ultimately, reporting.

At a high level, we aim to establish that the financial statements (or internal management information perhaps) agree to the underlying accounting records, and that these agree to the details of the commercial reality they record. We want to check that the amounts, values, timing, parties, and classification of transactions agree to the commercial reality, not just that they are of the required data type and logically feasible according to the internal logic of the application.

1.3 MDL Risk Assessment

A feature of MDLs that reduces audit risk is that cryptographic techniques are used to make it very difficult (impossible?) to make changes to the recorded details of a transaction after the transaction has been added to the MDL. An 'immutable' audit trail is widely considered

'a good thing.' It reduces the risk of the recorded details of the transactions being changed fraudulently or by some error.

However, this particular risk has not in the past been a major concern for auditors. It is not one of the more likely problems, and audit tests deal with it to an adequate degree. Usually, the substantive tests focus on whether the transaction records existing **now** agree with other evidence of the reality of transactions when they occurred. Control tests focus on access restriction for computer applications and environments. Further layers of control, as well as analytical review audit tests, indirectly provide additional assurance that records have not been changed in between being created and being used to generate financial statements.

Other aspects of MDLs are not so helpful for auditors when compared to conventional computerised ledgers. To understand them, consider how a typical MDL works. The MDL is represented by multiple copies of the entire ledger. New transactions are entered by people or captured by automatic equipment at various points around the network of computers linked to the ledger. Those details of new transactions get moved and copied across the network and, eventually, are herded together by one of the computers hosting a copy of the ledger ('nodes') and compiled into a new block that is then added to the chain on that computer. That new block also needs to be sent off to the other nodes so that they can all keep up to date.

What could go wrong within that computerised process, assuming that a valid transaction happened and that its details were initially entered correctly? There might be delays (potentially days) in between initial recording and compilation into a new block, which could prove to be problematic around a financial cut-off date. Some transactions might never be added. The same transaction might be input twice, or added to the MDL twice. Two transactions might be entered that are incompatible, but this can go unnoticed because the first transaction has yet to be added to the ledger. For example, the second transaction spends money that the first has already spent, but before the first payment has reached the MDL. Maybe not all copies of the ledger get the update. Or two nodes create the next block at the same time and there is conflict as to which has created the true next block. Finally, someone might try to go back and change the details of a transaction that is part of the MDL.

MDLs contain programmed mechanisms to guard against these problems, usually involving cryptography, but not all MDLs work the same way. Some of those control mechanisms are quite strong and some even have mathematical proofs of their degree of effectiveness (though not always that they are completely effective). However, this does not apply to all the problems and may rely on some false assumptions.

Subtle forms of attack exploiting weaknesses may be possible. For example, with Bitcoin, a problem called transaction malleability has been identified.[4] Things could get more complex

[4] See Broby, D, and Paul, G, (2017). The financial auditing of distributed ledgers, blockchain and cryptocurrencies. *Journal of Financial Transformation*, vol 46.

still, if high transaction costs or slow speed drive system designers to create off-chain mechanisms (e.g., the Lightning Network) to record transactions before bundling them up to be submitted to the main MDL. Bitcoin transaction costs have often been several dollars per transaction. Established payment systems have much lower costs, perhaps a few cents, though charges may be much higher.

The MDL system software running on each node is supposed to be the same, or at least compliant with particular standards, but how do we know that it is? A node might be running something slightly or very different. It might be running the software on a computer that is insecure and might allow some kind of crooked, unauthorised replacement of the software. Some types of MDL use voting mechanisms to decide what is true, and these might be subverted, if too many of the nodes are controlled by the same people (either the legitimate controllers of those computers or hackers).

When the auditor consults the MDL to see what it says, the result that comes back is not a direct response from one database as we would normally expect. The MDL is really a set of multiple, nearly identical copies of a data file or set of files. They are *nearly* identical because of the most recent transactions and blocks. The software that delivers a result from 'the MDL' has to decide which slightly different version is to be provided.

However, there is another layer in between the auditor and the underlying MDL records. The MDL file itself is an unindexed list of transactions. In order to answer almost any interesting question (e.g. "What are my transactions for July?"), the node has to create a representation in memory or on disk that allows direct access to the MDL, or a restructured version of it, rather than requiring a serial search through the entire MDL on disk. So, when you consult 'the MDL', what are you really seeing?

What about the quality of MDL systems software? Compared to a conventional DBMS, most MDL implementations are younger software. This is an advantage in some ways but also means there has been less time to discover and fix bugs. Also, although the notion of an MDL seems quite simple, the processes needed to maintain the multiple copies, form new blocks, and allow efficient access to the data are complex new challenges. Bugs are to be expected. Many of the control mechanisms in MDL systems – reassuringly described with words like 'validation' and 'consensus' – are really just addressing problems that do not exist in a conventional application with one conventional database.

Can an MDL be relied on to continue providing reliable financial records in future? It requires that the computer hardware keep up with the increasing demands of the MDL. As the MDL itself gets bigger it requires more space to store it and more power to process it. By the end of August 2017, the Bitcoin blockchain was around 130GB in size, and the Bitcoin Cash blockchain was even larger. The Ethereum blockchain is larger. You may not be able to archive old transactions. The more nodes there are, the greater the network traffic, as the nodes share new transactions and completed blocks.

Another source of risk to continued support is the impact of hard forks. Users of MDLs can reach governance crises (e.g. disputes over the path for future technology) where the only way out is for the MDL to split into two MDLs. Bitcoin split into Bitcoin and Bitcoin Cash in August 2017, after a disagreement on how to deal with a performance bottleneck. Ethereum became Ethereum and Ethereum Classic after a difference of opinion on how to deal with the DAO theft in 2016. Functioning blockchains remained but with uncertainty for users.

Continued support of the MDL requires, of course, continued support of the software that uses it. Who is doing that work and what does the future hold for them? Many MDL suppliers are not large, established system development companies.

1.4 Audit Tools and Techniques

Control and audit of computerised systems increasingly has relied on software tools. Auditors use tools built into application software (e.g. control reports) as well as running their own tools on the underlying data files or by communicating with the DBMS. To a large extent we have learned to trust computerised arithmetic, to expect double entry systems to have a net balance of zero, and to expect control accounts to be in agreement with sub-ledgers. Some MDL implementations have better controls than others. Tools for auditing MDLs may be rudimentary or non-existent.

A fundamental part of substantive audit tests is to compare records of transactions made at different stages of an accounting cycle to see if they agree, or can be, at least, reconciled. For example, a recorded sale might be agreed to the invoice raised, to records of goods delivered, and to records of stock released. Typically, we are more interested in comparisons between independent records and we tend to assume that records within one computerised system will be internally consistent. For example, trade customers are often asked if they agree with the organisation's record of money owed at a year-end date.

Audit experience, especially in large, complex organisations, is that any two databases that are supposed to agree will be at least slightly different, even if they are within the same organisation.

Auditors also take some comfort from the fact that organisations hold records of transactions and systematically compare those with what their customers, suppliers, and others have recorded. For example, a bank reconciliation compares the organisation's records of cash movements with its bank's records. The aim is to find errors by either organisation.

The potential change that MDL applications bring is 'one' shared ledger of transactions between two or more organisations[5], with no need or opportunity to use comparison to detect errors or fraud. The options for control in this design include:

1. Each organisation inefficiently creating another record anyway, so that comparisons can be made.
2. Using multi-signature transactions to build control into the initial agreement and capturing transaction details so that all parties to a transaction confirm that the details have been captured correctly and can be recorded on the immutable ledger.
3. Using a process whereby multiple transactions are entered on the MDL to record what is, commercially, one transaction. An example of this would be recording a contract as an offer transaction (entered by one party) followed by an acceptance transaction (entered by the other party).

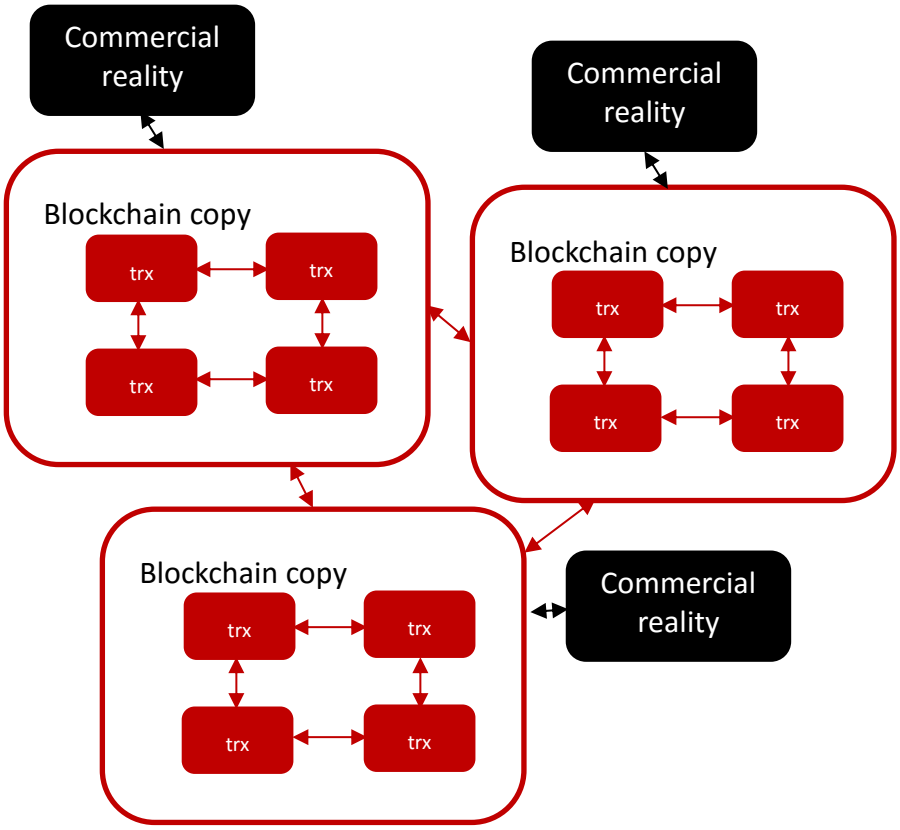

Figure 3 Auditors look for logical consistency between transaction records through a process, between MDL copies, and, crucially, between the data and commercial reality.

A shift towards one shared record of transactions would mean that auditors would need to change their approach, relying less on reconciliations between multiple records, and more on the processes by which transactions are initially recorded, or subsequently confirmed.

[5] This would also be true when an exchange or bureau service hosts the database on behalf of a group of trading companies, so is not unique to MDLs.

1.5 Quantity Of Audit Work

Recently, there have been publications online claiming that MDL technology will all but eliminate the need for auditors. An MDL should provide a more reliable audit trail than we are used to because of the hashing and chaining. However, changes to data after initial recording have rarely been a major audit concern in the past, and the crucial and bigger task of proving that records agree with reality remains. In addition, the distributed nature of MDLs introduces new audit concerns.

The extent of control provided by MDLs has sometimes been exaggerated. Within MDL implementations there is often a consensus mechanism that confirms that a block has been created in a *logically valid* way given the transactions awaiting inclusion on a block and the existing MDL, despite the multiple copies of that MDL. This is fundamentally different to all parties to a transaction reviewing and confirming that the details added to the MDL are *true* (figure 3). That may or may not be happening depending on the implementation and it is not a defining characteristic of MDLs.

Claims of 'validation' have to be considered sceptically, since the word could mean no more than a check that the details of a transaction are of the correct data type. Consensus, the favourite mechanism for establishing truth in the MDL world, is not ideal either. For example, the Augur prediction market uses voting by participants to try to establish the truth on propositions on which people have been betting. This is wrong in principle; the truth is not a matter of opinion. Rather, Augur documents majority opinion.

Perhaps the biggest challenge for MDL extreme proponents is to prove that all copies are identical. Clearly the code is designed to provide integral copies, but proving that is difficult to impossible.

Session 2 Auditing Distributed Data

2.1 Distributed Copies of the Data Store

An MDL is represented as multiple near-duplicates of a chain of blocks of data residing on separate computers that are usually owned by different people or organisations. The communication and processing work needed to keep those multiple copies in agreement is crucial and, once security and performance issues are addressed, the software needed is complex.

Despite the design of the software, at any time there will usually be:

- transactions that have been captured on a system but not yet added to any block; and
- blocks that have not yet been added to every copy of the MDL.

While the delays in a conventional computer system might be seconds or mere milliseconds, the delays in an MDL application might be minutes, hours, or even days. The median time needed to confirm a Bitcoin transaction on just one new block in August 2017, ranged from just over 6 minutes to 29 minutes. Bitcoin Cash, during its first month of operation (August 2017), created blocks at a rate of between one per 205 minutes and one per 1.6 minutes. For a trader waiting the typical time of 6 confirmations that is, again, a wait of at least several minutes. Ethereum was faster that month, confirming a transaction (with one new block) after between 20 seconds and 25 seconds.

In some MDL designs, a soft fork can occur if two nodes complete a proof-of-work task at almost the same time. This leads to two different versions of the MDL developing, until the nodes become aware of the situation and the longest MDL is selected for continued support. This delays the point where the MDL becomes reliable. In theory, this is a problem that grows along with the number of nodes.

2.2 Date Cut-Offs

Fortunately, most audits are carried out sufficiently long after a period end, for these delays not to be a problem. However, this also requires that the accounting organisation waits sufficiently long before deciding that the MDL is up to date to the end of the accounting period.

Another problem that might occur in some MDL applications is confusion between the time at which a transaction occurred in reality, the time at which it was recorded on a computer system, and the time at which it was added to a block on the MDL. The relevant time is the time at which the transaction truly took place, but confusion becomes a greater potential problem if there are longer delays between the times recorded.

According to Broby and Paul,[6] the Bitcoin blockchain does not contain individual transaction timestamps. The only times available are the creation times of the blocks.

The appropriate timing of transactions around a period end is more difficult if transactions are recorded in multiple time zones, or if different clocks are used to provide the time of the transaction recorded. The extra potential complexity with an MDL application is the possibility that an organisation's record of a transaction on the MDL might be put there by a computer with a clock controlled by another organisation (e.g. the customer).

In some financial services transactions, exact timing is even more critical, since it establishes the sequence of deals made. A collaborative experiment using timestamping to the nearest microsecond and even nanosecond, using atomic clocks, has been conducted with very rapid, high volume writing of test transactions to an MDL.[7]

2.3 Reconciling Differences

In theory, each copy of the MDL should be identical except for the most recent block, or a few recent blocks. As with date cut-offs, if we wait for a few days, that should usually be long enough for the blocks relating to the audit period to be in agreement across all copies of the MDL. In that case, if we want to know "What was the ABC account total at the year end?", we can look at any copy of the MDL and get the same answer.

In practice, how do we know the distributed MDL mechanism has indeed brought all those earlier blocks into exact agreement? Conventional strategies for checking this would include:

- a substantive approach where we somehow access multiple MDL copies (on multiple nodes) and test sample transactions or totals to see if they agree; and
- a compliance approach where we examine the design and test the operation and results of programmed controls built into the MDL software.

Either way, this is specialist work and likely to be expensive and time consuming. Auditors will not want to do it every time they audit a process through an MDL application. Alternatives include some kind of certification of MDL applications, or simply doing it very occasionally and relying on the assurance from other tests (e.g. analytical review).

[6] Broby, D, and Paul, G, (2017). The financial auditing of distributed ledgers, blockchain and cryptocurrencies. *Journal of Financial Transformation*, vol 46.

[7] A 2017 trial involved the National Physical Laboratory, The Toronto Stock Exchange, Z/Yen, Interxion, the Strathclyde Business School, and Hyperneph - http://www.npl.co.uk/commercial-services/products-and-services/npltime/researchers-bring-atomic-clock-timestamp-precision-to-stock-market-trading-over-distributed-ledgers

An established solution to a similar problem is the use of ISAE 3402 attestations of service organisations, or audits under similar standards.[8] A company providing a service used by more than one client buys an external audit (under the ISAE 3402 standard) and then the auditors of all its clients can use the resulting report as evidence in their audits.

A less conventional approach might be for the auditor to host a node and, probably with some special audit tool software, use that to monitor for differences between MDL copies between itself and its peers. This kind of comparison is a typical feature of MDL software, but reporting and analyzing differences found might not be. More comprehensive statistics on differences and delays might be provided by services such as the Ethereum Node Explorer[9] and Ethereum Network Status[10] websites.

2.4 Evaluating Technical Designs

MDL applications and software components they rely on include a number of controls designed to keep the MDL secure, consistent across all nodes, and up to date. A conventional compliance audit approach is to identify those controls, evaluate their design individually and collectively, and test their operation and results. The challenges for auditors include:

- learning about and understanding the strengths and weaknesses of control mechanisms that are complex, often abstract, and often explained in confusing mathematical ways with unfamiliar language and symbols;
- finding out what is built into MDL software in use; and
- testing that those features have operated.

To illustrate the first challenge, if you learned that a system used homomorphic encryption, would you be encouraged or worried? Homomorphic encryption is any form of encryption where you can perform computations on the cyphertext first, then decrypt, getting you to the result you would have got if you had done the same computations on the unencrypted data. It means you can do processing on financial numbers without knowing what those numbers are (making it more private), but, also, that you can do fraud on numbers without having to crack the code (e.g. adding 10%). Consequently, homomorphic encryption is good for privacy but not necessarily good for fraud prevention. Some encryption schemes are partly homomorphic. It's complicated!

Zero knowledge proofs provide an even more intimidating example of the learning challenge for auditors. A zero knowledge proof is one where A convinces B that A has some secret knowledge without revealing that secret knowledge and without providing evidence to a third party, C, that would allow C to prove to a fourth party, D, that A has the secret

[8] ISAE 3402 *Assurance Reports on Controls at a Service Organisation*, is the relevant International Standard for Assurance Engagements. A well-known equivalent from the USA is SSAE 18, which replaced SSAE 16.

[9] Ethernodes.org

[10] Ethstats.net

knowledge. This usually involves B making choices that B knows are genuinely random but which an observer could not be sure are random.

Often it will be easier to rely on substantive testing, either automated on large samples or complete data sets, or manual on smaller samples. This might be guided by, and combined with, analysis of performance statistics from across the network. However, in time, as the leading MDLs become established and develop a reputation for reliability and performance, and after specialists have reviewed their design several times and shared the results, it may be that auditors come to rely on those programmed controls, just as we tend to assume correct arithmetic by computers today.

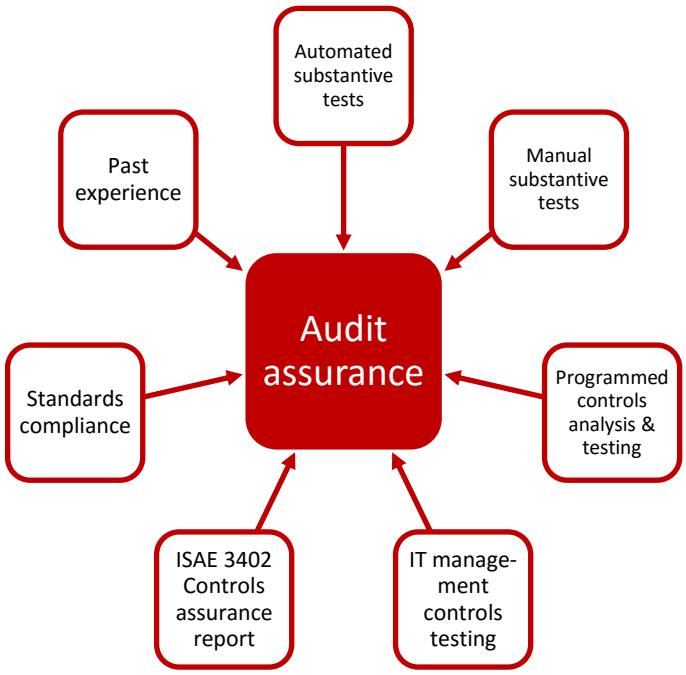

Figure 4 The auditor will seek the most efficient blend of evidence from different sources.

2.5 Distributed Controls Operated by People

The established pattern for auditing in large, well-controlled organisations today is to do some work on their 'general IT controls' to check that their computer systems are competently looked after and acceptably secure. This requires interviewing people involved in computer operations and related work, doing tests to check that security has been implemented as designed, and so on.

In the case of an MDL, the computers used will usually be owned or controlled by a number of organisations other than the organisation undergoing the audit. And, unlike a situation where computer operations or one of the processes have been outsourced or placed in a shared service centre, there are many of these, and access to the auditor is even more restricted than usual – probably non-existent altogether.

What proportion of the computing power has to be well controlled for audit reliance? How do we know which of those computers are well controlled? Is it the case, as we would expect, that it is only necessary for a sufficient majority of computers to be well controlled, or is there some devious hack that means the overall security is only as good as the least secure computer?

Some of the most important computer control will be provided by software developers working on the MDL application or underlying software components. Their organisations might not be hosting nodes, or they may be working on open source projects, with no legal organisation responsible at all.

Session 3 Auditing Distributed 'Smarts'

3.1 Embedded Code

The 'transactions' recorded on an MDL can be surprisingly complex, along with the details recorded about them. We might expect to see that a contract for a sale is recorded on a ledger in terms of its quantity, value, date, parties, and classification. However, in some MDLs, more of the terms of that contract are recorded, and in such a way that the software that operates the MDL will execute some or all of the contract automatically when its conditions are met.

For example, a bond paying 5% interest annually for 5 years might be recorded as a smart contract in such a way that the initial loan is paid on the agreed date (in crypto-coins perhaps), interest is paid annually at the agreed rate, and the loan is paid back on the due date, all automatically.

If the computer controls a machine that controls *us,* then the sense of a contract being automatically enforced is greater. For example, code might lock a door or deliver goods from a vending machine.[11] Although the code that does this is often called a 'smart contract'[12] it is almost never a complete contract and usually not even part of a contract.[13] It is just code that executes part of what was agreed. As such it is nothing new. The interesting part is that code held in the MDL itself is another way to create applications.

The conventional approach is to have application software that talks to a database management system or an MDL system. MDLs do this, i.e. act like a database. However, they provide an alternative. That alternative is to construct applications using one or more pieces of embedded code. Several pieces of code can work together. Code can write more code that can be executed in the future. These architectures are being used, but the scale of complexity is daunting to an auditor. At least, the MDL provides the ability to have cryptographically secure timestamps of what happened.

Legally, a contract is an agreement between parties. It does not even need to be captured in writing, except for some kinds of contract. Even legally drafted contracts often contain clauses that are too vague or have meaning that is too open-ended to be captured in code on a computer (at least not in practice). They may rely on courts to decide the exact meaning of particular terms, if necessary, or may give a party the right, in certain circumstances, to choose between alternatives based on whatever they think is relevant at the time.

[11] These examples appeared in the seminal article on smart contracts: Szabo, N. (1997). Formalizing and Securing Relationships on Public Networks. *First Monday*, 2(9). doi:http://dx.doi.org/10.5210/fm.v2i9.548.

[12] The meaning of the phrase 'smart contract' is not yet settled, with alternatives proposed and some interesting refinements of terminology too.

[13] *Smart Contracts and Distributed Ledger – A Legal Perspective*, by ISDA and Linklaters, August 2017.

For 'smart contract' code to be more meaningfully a contract, the agreement would have to be that the *code* is what is agreed (perhaps with other, non-operational clauses recorded elsewhere). Alternatively, the written contract might have certain clauses written in a formal language similar to legalese but more constrained, so that software can read the clauses and automatically turn them into code. Or key terms might be tagged using a mark-up language, as in a Ricardian Contract.

Automated enforcement of contracts using an immutable MDL to store the details offers the prospect of cutting out some forms of delay, error, and breaches of contract.[14] However, it comes with some downsides. To record those smart contracts you need to write code in a language understood by the MDL system. The language will let you write a very wide range of contracts and contains constructs to capture almost any logic you might imagine, however complex. However, the code can only react to conditions it knows about. So, for example, if your contract paid money if the weather was bad then the computer would have to have access to information about the weather or someone would have to tell it the weather had been bad enough to qualify under the contract.

Smart contracts are, typically, computer programs held on the MDL and executed by the MDL software (or not if there is a fault). The more conventional approach is for that program logic to be in application programs. All the usual potential problems with unstable code and code that does not reflect the user's intention apply. All the usual controls about specification, testing, and other quality assurance activities are applicable, at least in theory, and especially if the contract specification is in any way complex or novel. That code is not like natural legal language and is not friendly to lawyers. It is a computer language devised by computer experts.

Furthermore, the code is not like most computer languages. The archetypal introductory computer program is the Hello World program that just displays "Hello World" on the screen. In a high level language, this would be something like:

```
Print "Hello World"
```

In Ethereum's Solidity language, used for smart contracts, an example[15] uses almost 20 lines of code to do the same thing and is not easily understood. Weaknesses are also immutable, so even if you realise you have made a mistake it is too late to do much about it and you just have to wait and hope. Ethereum's DAO crowdsourcing project got off to a disastrous start when just such a weakness was spotted and used to siphon off millions of dollars.

Smart contracts are sometimes described as 'self-executing' but code does not execute unless there is a computer to run it on and another program to load the code and run it. In

[14] For example, in the London insurance market. See From "Slips To Smart Contracts: Intelligent Technology In The London Wholesale Insurance Market", a Long Finance report by Z/Yen Group, 2017.

[15] https://www.ethereum.org/greeter

some cases, because the cost of computation across many nodes is high, it also needs to receive a transaction that triggers execution. If these are missing, then the smart contract will not execute. If the information the smart contract needs to know about is not available to it, then it will not execute correctly. There is no guarantee that a smart contract will execute promptly.

3.2 Audit Aspects Of Smart Contracts

A typical *internal* audit concern is that a company might lose money from mistakes with smart contracts. The *external* auditor is only concerned about losses if they lead to the accounts being wrong. This can happen if the problem emerges after the accounts have been finalised, but shows the real situation was not as reported, or if the loss is so serious that the company is no longer a going concern and its accounts need to be restated using different accounting assumptions. Problems might arise from:

- an attack that exploits some quirk of the contract that was thought to be unimportant or was not noticed at all, but can be made important by a clever exploitation; or
- smart contract code that does not perfectly capture the true contract agreed between the parties.

The auditor might review or test smart contracts directly or rely on the organisation doing so. Either way, the task is a challenging one. The most worrying potential problems arise from systematic problems across large numbers of contracts with collectively large values, and from individually large contracts. Consequently, the auditor's efforts should be directed at contract types that are material due to the value of all contracts for which they have been used. On Ethereum, it is possible to set up templates so that parameter values from standard contract forms are slotted into standard smart contract code. A problem with this could lead to systematic errors.

Overall, standard templates established by standards organisations, with standard code that implements them on particular MDL systems, could be the most practical and safest way to use embedded code.[16] The templates would have parameters that can vary (e.g. quantities, dates, parties, locations, calculation formulae) within an otherwise fixed contract.

3.3 Quality Assurance for Code

Quality assuring computer code is a task that has been studied extensively and there have been many scientific studies providing useful information on what works. In this paper, we only consider very briefly three strong contributors: (1) testing, (2) inspection, and (3) proof.

[16] See the discussion in Clack, C.D., Bakshi, V.A., and Braine, L. (2016). Smart contract templates: foundations, design landscape and research directions. *arXiv preprint arXiv:1608.00771*.

Testing would require a test system that allowed code to be written and then confronted with different sequences of events, to see that the code produces the intended response. A lot is known about the difficulties of exercising code with all potential future situations. An unavoidable challenge is that the testers need to have an exact and correct understanding of what the true contract is intended to be. If, like the coders, they have misunderstood the intention, then the code may still be wrong despite testing.

Inspection involves people other than the author of the code reading it very, very slowly and thinking hard about everything that might be wrong. They may do so with explicit quality requirements in mind. There may be a process of meetings and feedback to discuss defects. Empirically, this kind of inspection is highly cost effective with software, but people tend to feel they should be working faster and tend to lose their way over time, returning to hasty bad habits, instead of keeping to the level of rigour that is required.

To some extent, testing and inspection can be automated. Proving code, 'formal verification', is an idea that goes back to, at least, the 1950s. In the 1980s, there was excitement about Z and similar formal mathematical techniques (e.g. VDM, CSP) capable of specifying precisely and completely the behaviour of software systems. The ultimate aim was to start with a high level specification in Z, and then refine it, by steps if necessary, down to executable code. The idea was that this could be supported by software (e.g. a proof editor with a proof assistant), making it easier and more reliable.

In principle, this could have worked, and some leading experts were able to show examples of small programs developed or checked using refinement and proof techniques. In practice, the movement enjoyed a few years of excitement, but then settled down to being a niche community. Even though some quite good software tools were developed, and even though Z is a powerful and elegant specification method and an excellent way to write mathematics, almost nobody had the brain power to do it well.

Using proof techniques with smart contracts would involve specifying the true contract in a formal language and then proving that the code is a refinement of the formal specification. That means you would have to capture the contract correctly in the specification and then write a correct proof (though that might be made easier by a software 'proof assistant' tool).

Making this more challenging still is the fact that examples of smart contracts in use, already, include complex systems devised by designing multiple types of smart contract to work together. This frequently involves not only understanding how the software will work together, but also how people will interact with the software and drive the overall system's behaviour.

3.4 Money Supply Algorithms

One interesting example showing the potential complexity and importance of designing 'smart contracts' is the problem of money supply for cryptocurrencies. The cryptocurrency might be a public one, open to all, or restricted to just members of a private group. In either situation, but especially where the pool of participants increases over time, the total value of cryptocoins held by participants at any time, and the total value of transactions in a period

of time, rises as more people get involved and/or increase their cryptocurrency usage. Unless the supply of the cryptocurrency also rises, in just the right way, the value of each cryptocoin will rise, which is a serious problem. It means that ordinary goods and services cannot be given fixed prices and participants cannot learn to think in terms of units of the cryptocurrency. The cryptocurrency is not suitable for ordinary use as a payment mechanism, and instead exists as a speculative arena, something like online poker.

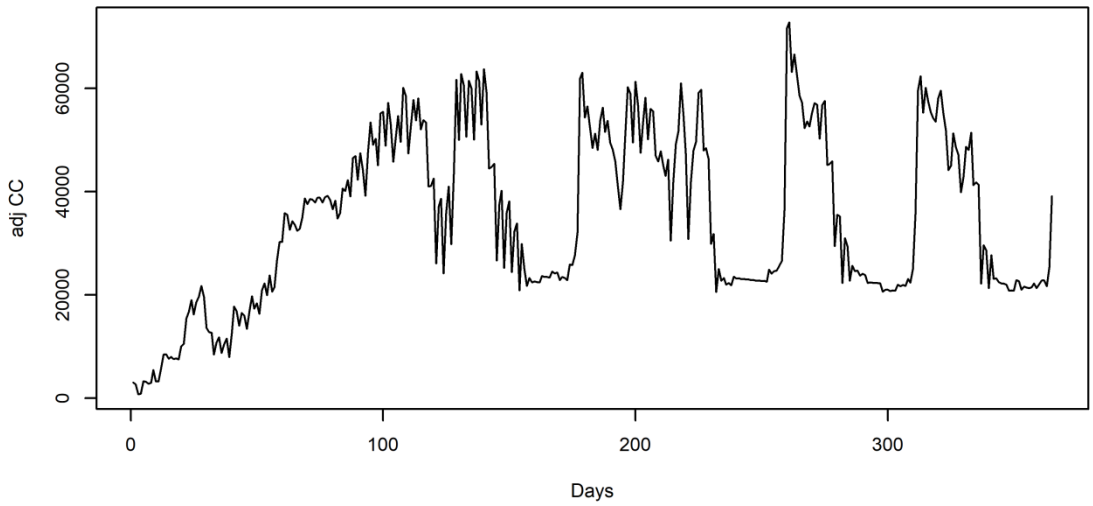

Figure 5: A trace from an agent-based simulation of a generic cryptocurrency, showing distribution-adjusted total holdings of cryptocurrency by merchants and customers. A currency exchange holds the rest – trying to cope with the volatile behaviour.

Unfortunately, managing the money supply is not as simple as increasing the supply at a constant rate, or holding the velocity of the currency constant. Differences in distribution, usage, and behaviour in response to economic changes, also have roles and complex, unpredictable, dynamic behaviour results. A simple rule for money supply that is linked to transaction counting, or time, is unlikely to succeed in providing a stable, practical currency.

Simulations by Z/Yen (see figure 5) have shown that the economic evolution of a cryptocurrency is hard to predict and to control, and that further work with simulation will be needed to develop effective money supply and exchange-rate revision rules. These will almost certainly be complex, adaptive rules rather than simple ratios or fixed plans.

3.5 High-Tech Audit Techniques

Computer assisted audit techniques have been used for decades and have gradually improved. They allow auditors to select samples, search for exceptions and anomalies, and perform detailed agreement between different records of the same transactions. They can be used within an audit project or adopted by an organisation as part of its routine control system. Statistical (e.g. support vector machine) or machine learning techniques can provide Dynamic Anomaly and Pattern Recognition models to identify what is 'normal' and what is anomalous in a stream of transactions.

In principle, the same should be possible with an MDL system, but there are some new problems to be solved and, as discussed earlier, some new concerns to address. The auditor might get data from the organisation's node, from another node that provides an MDL

explorer service, from a node hosted by the auditor, or by visiting a website that provides performance information reported from across all the nodes (e.g. the node[17] and block[18] pages provided for Ethereum). The auditor might gather information by using simple user interfaces, by writing programs that use the Application Programming Interfaces often provided for MDL systems, or by using a third party tool that does so.

[17] https://www.ethernodes.org/network/1

[18] https://etherscan.io/

Session 4 Auditing Consortium MDL Systems

4.1 Consortium MDL Systems

The computer resources needed for a distributed ledger using a conventional MDL are a function of the number of nodes. That's if we ignore for a moment the computer resources needed to keep a backup and the additional overhead from racing to complete 'proof-of-work' computational tasks with no purpose other than deciding who gets to create the next block. So, if you have 10 nodes then that will need at least 10 times the IT resources needed by one conventional database. Increasing node numbers raises resources exponentially in backup, network traffic, and 'proof-of-work' tasks. If the MDL's principal purpose is timestamping, then it is interesting to question how many copies are needed to provide valid proof.

IT resource efficiency is not a reason for using an MDL. Instead, the justifications focus on new multi-organisational efficiency. For example, MDLs may provide mechanisms whereby many parties who do not trust each other can still trade together with some confidence in record keeping and sticking to agreed actions. MDLs might eliminate costly processes of comparison and reconciliation.

Finally, it may be possible to reduce the problem of paying high fees to a monopolist who runs a computing service that is shared by everyone in a market. Suppose you plan to set up an exchange that will allow 10 companies to trade a commodity with each other. Having all the IT done once, in one data centre, by one central exchange looks attractive. However, a few years later, once everyone is entirely dependent for their continued commercial survival on the operation of that exchange, there is no easy way to block gradual increases in exchange fees, as the exchange flexes its negotiating muscles[19].

A consortium of companies wishing to trade together might adopt an MDL application, with a relatively small number of nodes, as their solution. Although less efficient in purely IT terms, it might be more efficient because of reduced discrepancies and less risk of price gouging by a monopolistic exchange operator.

4.2 Governance Needed

The elimination of central third party power is unlikely to be complete though. For example, central third parties are probably efficient ways to:

- fund and control software development, standardised 'smart contracts', data standards, security standards, standard operating procedures and templates;

[19] Not unless the participants jointly own the organisation that runs the exchange.

- specify technical standards (probably de facto or voluntary technical standards) to apply to the design of software and smart contract code,[20] as well as parameters that control the behaviour of an MDL system (e.g. fees, difficulty factors, block size);
- handle governance crises (e.g. fraud, technical dead-ends) requiring human intervention and cooperation.

Similarly, there might be advantages in appointing an auditor to provide common audit aspects of an MDL system and its governance, and give assurance that can be relied on by the auditors of individual consortium members. This might be done under the ISAE 3402 standard, or something similar.

Cooperative decision-making is something that MDL communities are very interested in. Some of the more routine shared decisions might be governed by routine voting. Less routine matters might involve more complex procedures or 'constitutions' covering the way proposed solutions to problems are developed, communicated, assessed, discussed, and finally voted on or otherwise selected. It is important to choose an overall style of governance suited to the situation.[21]

Simulations of the MDL system and its users could be used to explore alternative options when debating new solutions to technical problems, or be an intrinsic part of some of the system's control rules, with short range predictions being used to guidance control decisions.

Since the Bitcoin is a permissionless (i.e. public) system, this means an auditor (in fact, anybody at all) can see every transaction and every account. To stop the world snooping on Bitcoin users, the many accounts used are identified only by long strings of seemingly random characters. Each user has many of these accounts and, to make it harder for others to see and understand their activity, users are often encouraged to set up new accounts for every transaction.

For an auditor, it is as if the organisation has thousands of bank accounts, none of them named, and all mixed in with the bank accounts of thousands of other organisations. Money can be traced from account to account, but without knowing who owns the accounts, the exercise is largely futile. Various splits and aggregations of money help to confuse matters further, and some services for Bitcoin users provide a privacy mechanism by deliberately making tracing money next to impossible.

Bitcoin users are individuals or, more precisely, they are anybody who knows the private key of an account. If your chief accountant has a memory stick with the private keys of your company's Bitcoin addresses, and she is the only one with that information, who has the

[20] See "The Missing Links In the Chains?: Mutual Distributed Ledger (aka blockchain) Standards". A Long Finance report prepared by Z/Yen Group.

[21] See "Responsibility without power?: The Governance of Mutual Distributed Ledgers (aka blockchains)", a Long Finance report by Cardano Foundation, 2017.

money? If she accidentally dropped the memory stick and someone else picked it up and took it home, who has the money now? Bitcoin does not check your face, your handwriting, your finger print. You don't have to be at a particular terminal, or within the confines of a secure building.

Within a private MDL, it should be possible to include human-readable information like names, addresses, asset descriptions, and delivery instructions. This could include cross referencing to other systems, such as the identities of securities held by a custodian, registered mortgages, or vintage wines in a commercial cellar.

'Debtors' has traditionally been an important category of assets for audit purposes. In principle, it seems that this is an area where a central exchange system or MDL alternative with a consortium might be easier to audit. Balances within one computer application are usually more likely to be in agreement than balances between separate computer applications. It may also be easier and quicker to conduct substantive tests of details on samples that are within one system than between two. Furthermore, if debts are to be repaid by a process automated with the use of a smart contract then it might seem that repayment is certain, or at least less likely to be missed by accident or disputed (did you mean delayed?) for cashflow reasons.

When a smart contract triggers payment of a debt it cannot or should not succeed if there are no funds to pay with. Also, it is possible that debts might be paid in a situation where they should not be, because the company is insolvent.

Other Potential Audit Issues

The audit challenges discussed in previous chapters are those arising directly from the use of MDL technology for processing and storing data. However, some *uses* of MDLs have associated audit issues related to tax, legal compliance, accounting methods, and disclosures. This chapter mentions some of the more obvious issues often associated with the use of MDLs.

In some cases, it could be difficult for an auditor to apply existing financial reporting standards without guidance or new standards clarifying the particular issues that arise, thanks to this new technology and new ways of using it.

Issuing Tokens to Raise Money

If a company raises money by issuing and selling some kind of digital token, such as a cryptocoin, or a token that entitles the owner to use the company's products or services at some time in the future (when available), the incoming money needs to be classified for accounting purposes. Typically, it is not an equity investment, but neither is it appropriate to recognise it all as revenue for the current year. However, there might be some revenue recognition, and a need to classify the outstanding liability correctly.

At each year end, there is a need to recognise the right amount of revenue and value the outstanding liability correctly. It may not be appropriate to value it at the money received originally. Revaluations would create an impact for profit and, in extreme cases, this could be large. The relevant International Financial Reporting Standard is IFRS 15 *Revenue from Contracts with Customers*.

The way tax authorities approach this may differ between countries, and also may be different from the proper accounting treatment. Or it may simply be unclear, leading to uncertainty and potential problems. If a company accounts for its token issuing activities in a way that is not consistent with appropriate accounting standards, but then needs an audit (perhaps to satisfy an investor), corrections will be needed and these could have a drastic effect on the apparent financial position and performance of the company.

Currency Valuation

If a company holds part of its wealth in the form of cryptocurrency, or transacts in cryptocurrency, then this raises some additional accounting challenges. The relevant international accounting standard is IAS 21 *The Effects of Changes in Foreign Exchange Rates*.

In principle, this might be no different from accounting in any other currency that is different from the currency in which the company reports its results. However, the high volatility and sometimes low liquidity of many cryptocurrencies are potentially challenging. With stable currencies, it is possible to take advantage of convenient short-cuts like using average

exchange rates. That is less likely to be available with highly volatile cryptocurrency exchange rates and it may be necessary to record the rate at the instant of each transaction.

Conceivably there might also be problems in valuing cryptocurrency balances at the period end because of doubts about liquidity. The authors of IAS 21 felt the need to have special rules for hyperinflationary currencies, and might have similar concerns about 'hypervolatile' cryptocurrencies, in the future.

Exchange rate movements create profits and losses, and these, too, require an accounting treatment and a tax treatment that might be different, at least in some countries. Future liabilities of a company expressed in cryptocoins might soar in fiat currency value. Rewards to staff that seemed modest when initially paid in cryptocurrency might appear over-generous when disclosed in the accounts in their fiat currency equivalent at the period end.

Data Protection

Data protection laws create two problems in particular for MDLs: location and deletion. First, what is the correct location for a database that is, really, a network of nearly-synchronised replicas of a database where copies exist in several different countries? Some of those countries may have laws that forbid personal data being held abroad. How can you be in control of data if your computer is automatically sharing it around with perhaps thousands of other computers in different entities and even different countries? Does it help that the data are encrypted?

Second, how do you delete data from an MDL that is specifically designed to prevent changes to records? One potential technical solution to this is to encrypt the data and delete the private key to that data, so that it cannot be accessed by anyone. This is rather like shredding paper documents but then keeping the shreds instead of burning or pulping them.

While these issues may not be a problem for financial auditors, they could be for organisations using MDLs and for their internal auditors or information security auditors.

Decentralised Autonomous Organisations (DAOs)

A loose collection of people coming together through a computer system to trade and make collective decisions in a sort-of democratic way, free from the complexities and restrictions of conventional companies sounds like a great idea to some. But to law makers and tax collectors it sounds like a dodge and they will be keen to get to the substance of these agreements and start interpreting them in more conventional terms, establishing tax charges, responsibilities, and liabilities.

In any particular jurisdiction, it may be unclear whether a person wishing to sue a DAO is blocked by having nobody to go after, or is able to go after anyone who is involved with the DAO in any way. While these seem to be among the largest legal uncertainties, the audit issues arising are not clear. Perhaps there might be difficulties if a company is in dispute with a DAO and a legal case has begun.

Suggestions for the Profession

MDLs could become very important in the future, even if not exactly in their current form. Their arrival could improve internal control, but could also undermine it, depending on the details of individual designs and implementations and on the way systems are managed and governed. The distributed databases bring new data integrity and timing concerns. The smart contracts and immutability drive a shift towards reliance on different controls. At the same time, some activities commonly performed with this technology involve some highly uncertain accounting, legal, and tax issues. All this could lead to more expensive audits but, for external firms, decreased recovery rates and more pressure on internal auditors.

The suggestions below are divided into two parts.

Adapt Internally

Adapt internally by learning about the technology, inventing and refining audit techniques, selecting and educating staff, training them, and equipping them with software.

While there was strong interest at the symposium in putting accounting standards in place to cover the difficult emerging issues, MDLs were considered too specific to be the subject of a dedicated auditing standard. However, there was strong interest in guidance for auditors. Smaller firms in particular would be likely to appreciate guidance from professional associations, and perhaps, a pan-institute effort would be a good idea.
Guidance might include content such as:

- An introduction to the basic technology involved, with an accessible glossary, that counters many of the common misconceptions.
- A taxonomy of types of MDL based on their *technological* choices (e.g. permissioned versus not permissioned), perhaps with some guidance as to how risky each is and why.
- A taxonomy of *applications* for MDL technology, again grouped in such a way that it is easier to see which are higher risk and which are lower risk, for both the organisation and for the auditor.
- Audit techniques for MDLs, covering such things as:
    - Initial risk assessment.
    - Adapting compliance testing to a situation where there may be no backups and where multiple copies of the ledger reside on computers in organisations to which the auditor has little or no access.
    - How to identify technology features, typically involving cryptography, that provide control that is relevant to the auditor.
    - How to use data about the performance of the MDL system from a node or by using an information server that gathers statistics from all or many of the nodes on the network.

- How to perform substantive tests with an MDL, including automated substantive tests.
- Options for cooperation between auditors.

- What conversations to have with management and when. For example:
  - Conversations about computer applications under consideration where MDL technology is to be used, and where the potential problems involved perhaps need to be considered more deeply, with a focus on implications for internal control and audit. Perhaps the use of MDL technology has not even been recognised by senior management.
  - Where management is asking the auditor for guidance that will help them decide whether to implement an application or system involving MDL technology.
  - Conversations where the auditor raises the subject of MDL technology to raise awareness of the implications for internal control and audit.
- Issues arising from MDL technology that are not technological but may still be very important, such as compliance with data privacy laws, tax implications, the status of 'smart' contracts, and issues about legal jurisdiction.

Idea	Explanation	Who and when
1. Participate and learn	Continue to participate in MDL related events, stay close to clients, be involved in projects.	Larger firms, short term and ongoing.
2. Issue guidance for auditors	Not a standard, but guidance that helps auditors understand the technology, relevant law, and related audit issues – and has practical suggestions for audit techniques.	Professional bodies, short/medium term.
3. A pan-institute group to work on the guidance	Collaborating to bring together expertise and share the costs of investigating the issues and developing worthwhile guidance.	Professional bodies, short/medium term.
4. Revise audit risk assessment processes	Revise the way audit risk is assessed during audit planning to increase awareness of MDL technology and related risk, and help auditors anticipate the implications for their audit approach.	All firms, professional bodies, short term.
5. Devise efficient audit tests	Thoroughly investigate the practical challenges of auditing where an MDL is involved, and devise test methods that are effective and feasible.	All firms, professional bodies, short term.
6. Host a node	To monitor MDLs that clients rely on, host a node of some kind and collect information. This is part of a number of strategies to audit the reliability of data on the MDL.	Larger firms, short/medium term.

Idea	Explanation	Who and when
7. Develop tools to help audit MDLs	Software tools within MDL systems, or developed separately, that make MDLs easier to control and to audit. For example, these might allow access to check data on nodes across the network, to see performance statistics for nodes, and to view trends.	MDL developers, computer audit tool developers, short/medium term.
8. Revise exam content	Revise the details of professional examinations to include some basic material on cryptography and MDLs.	Professional bodies, short/medium term.
9. Train audit staff	Provide training in the new techniques for risk assessment, testing, and audit tool use to a selected subset of auditors.	All firms, medium term.

Table 1: Suggestions for adapting internally.

The factors to consider when assessing audit risk where an MDL is involved might include:

- Use
  - Sensitivity: What the MDL implementation is being used for; is it sensitive or not?
  - Required reliance: Is there an overall analytical method (e.g. high level reconciliation) that means we do not have to trust the system?
  - Comparison opportunities: Can we compare/reconcile the MDL to another database in some way, especially if it is already a more trusted one or could be audited quite easily.
- Design
  - Complexity: More complex is probably worse, e.g. smart contracts, off-chain transactions.
  - Process design quality: Has the need to ensure that only valid transactions get added to the MDL been understood and are the processes well designed? (Either pre-entry agreement or post-entry confirmation.)
  - Control and audit tools: What do the MDL systems and MDL application as a whole provide to help with control and audit, e.g. exception reports, activity reports, delay reports, reconciliations, and audit sampling tools.
- History
  - Maturity: How long have the MDL systems components been in use, by how many people, and what has their experience been like? Have problems been reported and if so, have they been sorted out? Similar questions for an MDL application.
- People/community
  - Community of users: Quantity and quality. We want them to stick around and not reduce to one dominant group – unless we are auditing the dominant group.
  - Central support: e.g. of code development and governance.

Influence Externally

Influence externally to encourage clients, developers, standard makers, regulators, and law makers to adopt solutions that are safe and efficient.

Idea	Explanation	Who and when
10. Develop auditability requirements	A non-enforceable but informative list of requirements that make an MDL more auditable and stop MDLs from being impossible to audit effectively.	Larger firms and/or professional bodies, short term.
11. Promote audit requirements	Engage with clients, developers, and regulators to promote the idea of auditable MDLs and suggest requirements.	Larger firms and/or professional bodies, short term.
12. Accounting standards for the emerging issues	Either a new standard or amendments to existing standards to address the challenges of token issue and highly volatile, internationally exchanged cryptocurrencies.	Accounting standards organisations, start now for medium term completion.
13. Legal and tax risk analysis	A project to analyse more deeply the potential legal and tax issues of smart contracts, cryptocurrencies, token issues, and Distributed Autonomous Organisations.	Larger firms, professional bodies, short/medium term.
14. Engage with law makers	To reduce the chances of auditors being faced with dangerous uncertainties, engage with law makers over tax and legal issues arising.	Larger firms, professional bodies, short/medium term.

Table 2: Suggestions for influencing externally.

While it is not the auditor's responsibility to ensure that organisations control MDL applications appropriately, the audit profession is influential and its expertise in control and audit could help to guide clients and others towards sensible choices. In particular, it could be very helpful to provide some simple criteria for MDLs as a guide to what will make an MDL application controllable and auditable, in practice.

Conclusion

While it is too early to consider broad standards for MDL applications and systems, it might be helpful to know which features make audit and control feasible. These could be seen as desirable design features for organisations and their auditors to look for. Some initial suggestions are:

1. The MDL contains real-world references e.g. names and geographical addresses.
2. It provides a user interface and API for audit software, allowing:
    a. Selection of transactions from any node across the network.
    b. Comparison of transaction details between nodes across the network.
    c. Reading current status of any node across the network, including MDL state and node software details.
3. The MDL system provides an ongoing monitoring service across the whole network (e.g. logging delays, soft forks, buffer sizes, errors, invalid items, time to propagate fully, confirmation times, size of MDL, sizes of blocks, size of mempool or equivalent), logs those monitoring records, and makes them available to auditors via an API and a user interface.
4. Ownership of the records lies with the appropriate legal entity (e.g. a company not its employees) and this is enforced by the system so that:
    a. individuals cannot leave their employer and continue to use the MDL;
    b. a program can identify all accounts/transactions relating to a given legal entity, if it has appropriate permission.
5. No transaction can be added to the MDL without authorisation (signature) from all relevant parties. This might be one party (e.g. paying money), two parties (e.g. buyer and seller), or more. However, this might be achieved by representing one commercial transaction as two or more transactions on the MDL.
6. The MDL system adheres to a suitable technical security standard, helping to reduce vulnerability to hacking.
7. A governance mechanism is in place that allows cooperation with auditors of any organisation using the MDL application.

Future consortium applications could satisfy such requirements, putting management and auditors in a much safer position and reducing the risk of qualified or adverse audit opinions.

Participants

Sponsors

The project was sponsored by DasCoin.

Project team

The report was prepared by Z/Yen Group. The principal authors were Professor Michael Mainelli and Matthew Leitch.

Symposium participants

Anne Adrian, ICAS
Nick Andrews, MPAC
Rachel Connolly, World News Media
Brandon Davies, Global Association of Risk Professionals
Andrew Gambier, ACCA
Marek Grabowski, Financial Reporting Council
Osman Gucluturk, Centre for Technology, Ethics, Law & Society
David Henderson, Sweetbridge Foundation
Mark Holland
Henry Irving, ICAEW
Matthew Leitch, Z/Yen
David Lyford-Smith, ICAEW
Michael Mainelli, Z/Yen
Lynsey Marshall
Michael Mathias, DasCoin
Bob McDowall, Cardano Foundation
Paul Moxey, London South Bank University
Myfanwy Neville, BKL
Rodger Oates, BFSI
Jatin Patel, KPMG
Joe Pindar, Gemalto
Matthew Rocksborough-Smith, DasCoin
Thomas Toomse-Smith, Financial Reporting Council
Eric Tracey, Goodenough College
Steve Webb, PwC
John Willmott, Kingston Smith

Sponsored by

DasCoin™
The Currency of Trust

Our vision is to lay down a global bedrock of trust that unlocks prosperity for everyone. By eliminating the problems of traditional money and adapting trust to the digital age, we are creating a better system for holding and exchanging value.

We are focused on delivering solutions that increase empowerment, enhance control and expand freedom. We believe the pursuit of these qualities will lead to a better, fairer, more prosperous future for all involved. We are constantly innovating as we create, maintain and evolve the world's most secure digital currency.

www.dascoin.com

LONG FINANCE

"When would we know our financial system is working?" is the question underlying Long Finance's goal to improve society's understanding and use of finance over the long term. Long Finance aims to:

- expand frontiers - developing methodologies to solve financial system problems;
- change systems - provide evidence-based examples of how financing methods work and don't work;
- deliver services - including conferences and training using collaborative tools;
- build communities - through meetings, networking and events.

www.longfinance.net

Yen

Z/Yen Group Limited
41 Lothbury, London EC2R 7HG, United Kingdom
+44 (0) 20 7562-9562 (telephone)
hub@zyen.com (email)

www.zyen.com