Babelchain Machine Communication

PROOF OF UNDERSTANDING

Benedikt Herudek
(benedikt. herudek@gmail.com)

April 2016

Abstract

Bitcoin asks ‘How can participants in a distributed payment network come to an irreversible agreement about the status of financial transactions without resorting to a trusted third party ?’ Bitcoin achieves consensus via using hashing and incentivizing miners with bitcoins to solve these hash puzzles. We ask ‘How can participants in a distributed network come to a binding agreement with which machine language they will communicate?’ We want to use an approach similar to bitcoin. Our proposal will be to replace the hashing algorithm via a machine learning algorithm, predicting message formats allowing Machines to communicate, while keeping a version of Bitcoins Immutability Feature. Translators, the counterparts of Bitcoin Miners, will be incentivized by the network via rewards in crypto-currency.

The proposal tries to help solving the well known problem of Inter-Machine or Application Communication, where entities talk about the same things with different machine languages. The problem is prominent in areas like enterprise IT and particularly relevant for the Internet of Things, with its numerous device communication protocols.

One motivation for the approach is to use the computing power the bitcoin network brings up for hashing and creating immutability for the purpose of machine communication and hence enable transactions via enabling understanding between participants. The result we suggest will be that such a System is possible while maintaining a version of immutability, which promises to be sufficient for many usecases.

An important Implication of such a System could be reducing the tendency towards Information Silos in the Net: everyone would be enabled to communicate with everyone. Those willing to communicate in an open and permission-less network wouldn’t be obligated to use a standard set by a larger player anymore to interact with an eco system.

Table of Contents

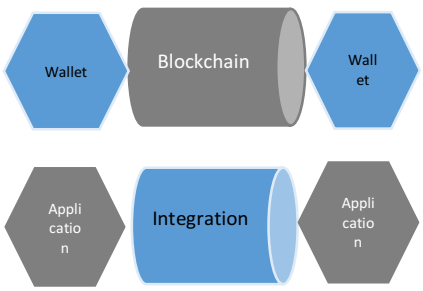
Machines can't (hardly) talk - how we usually try solving this

Consider the case of a CRM and Billing System in a Telecommunications enterprise. Client orders are taken in with the CRM System. After the Order is fulfilled, the Billing System will be responsible to generate a bill. Both Systems (in fact there are many more) know the concept of a Client and of Products, but they have different ways of talking about them and their internal Data Models differ.
With the rise of the Internet of Things, the problem sometimes takes the form: Devices have different message Formats but need to be able to communicate, e.g. the iPhone needs to be able to take over from remote controls and talk to all TV's.

The typical way of addressing the issue is to suggest an Industry Standard or follow a Standard set by a large player. From a Bitcoin & Blockchain one might want to see here a (broken) trusted middleman solution. Instead of trying handling the issue directy, peer2peer one reverts to a 3rd party, so to say a trusted middle man that can oversee both sides and mediates and settles between the two. There are some well known problems with reverting to such standards. First of all, there might be none, then there might be too many, then who ever sets the standard, especially a large vendor, might try to dominate an area.

Bitcoin Integration

Bitcoin is able to integrate endpoints like wallets and miners in a large distributed network seemlessly without any integration effort. If You ever opened a wallet on your smartphone and see a a wallet as some kind of application and if you ever worked on a project connecting for example an Enterprise Application into a network, you might find it suprising that you are able to make Your application talk to a large distributed network within minutes.
Bitcoin achieves this by flipping around the notion of Transactions and Applications. For Bitcoin, the Transactions (Blockchain) are permanent, while the endpoint, the wallets is derived.



Blockchain
Transactions (Blockchain) are permanent & primary
Enpoints (Wallets) are derived & ephemeral

Conventional Systems
Transactions (EBS) are derived & ephemeral
Endpoints (Applications) are permanent & primary

A bitcoin wallet is not an account holding a permanent value, rather it deducts the bitcoins the owner of the private / public masters from the transactions in the blockchains.

In spite of flipping this notion around, Bitcoin does something very conventional. It simply mandates the use of certain data formats, only if you read and write 'blockchainish' can you participate in the network. That is the right approach for a limited (even though very important) set of usecases like financial transactions.

Existing SOA Style Intergation patterns often follow the same pattern in dictating a message exchange format, often relying upon industry or vendor standards.

How Bitcoin Creates Consensus on Transactions

The so – called 'Proof of Work' algorithm is at the heart of Bitcoins Security Model. Security for Bitcoin is foremost the fact that the transaction log can for all practical purposes after a short time not be changed again and that is an indispensable feature for a finance transaction System, where we want to avoid that someone could spent the same bitcoin twice. The 'clou' – if you will – is that bitcoin achieves this without restricting any access to its payment system. Anyone can open a wallet and participate in the network and no matter how malicious a participant wants to be, he will not be able to overcome the System. The typical approach of nearly any System dealing with sensitive transaction would be tp restrict access, introduce Identity Management and monitor, audit and if necessary exclude players from the System. Bitcoin doesn't do any of this and still comes up with a remarkable stable Algorithm.

The 'proof of work' algorithm achieving this relies on hash algorithms. A hash algorithm takes an arbitrary-length data input and produces a fixed-length deterministic result. One metaphor to visualize this is the notion that a hash algorithm takes a unique fingerprint of a dataset, a fingerprint that for any practical reasons only this specific dataset can have and a fingerprint which is so interlinked with the dataset it reflects that any, even tiny change of one bit in the dataset will cause the fingerprint to look entirely different. Bitcoin uses what is called a 'nonce as this but that can arbitrarily changed by miners to try to impact the resulting fingerprint.

The fingerprint allows no reverse-conclusions as to how the input dataset might have looked like. It looks truly random – but its not. For any specific dataset, the fingerprint computed will always be the same. Another important feature is that that the fingerprint can be easily und undoubtedly verified to result from of a specific dataset. All one will have to do is processing the dataset through the hash algorithm in question and check the resulting fingerprint.
For any hash function for any practical purposes we can make the assumption that it is impossible to find two different inputs that produce the same fingerprint. This has an important implication: It is also virtually impossible to select an input in such a way as to produce a desired fingerprint. Hence, the only way to achieve this would be to try random inputs until one generates the desired fingerprint. Bitcoin's 'Proof of Work' draws heavily on this feature by demanding that certain fingerprints must meet certain standards, for example to be smaller that a certain target see. This makes it difficult to 'get the job done' and that is exactly what bitcoin wants to be the case, because this is its way to create an immutable blockchain.

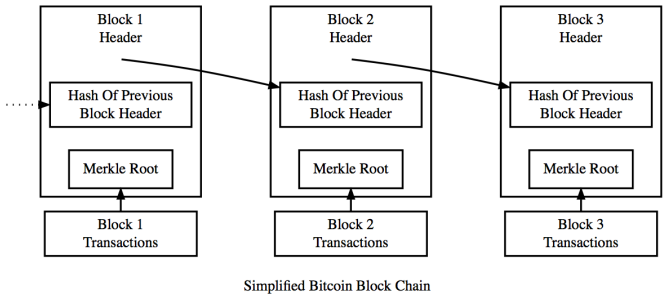
Bitcoin wraps this in a process called 'Mining'. Mining is essentially the way you get someone to execute this task, the way to incent him. Because if someone achieves the goal set by bitcoin, he will have a good chance to receive the mining reward, which corresponds to a number of bitcoin representing a certain value.

Technically, Mining is the process of hashing the block header repeatedly, changing one parameter ('nonce') , until the resulting hash matches a specific target. The hash function's

result cannot be determined in advance, nor can a pattern be created that will produce a specific hash value. Hence, the only way to produce a hash result matching a specific target is to randomly modifying the input until the desired hash result appears by chance.

Bitcoin requires hashing the Blockcheader because the Blockheader describes a set of transactions, that the Miners gathered in the distributed network. Hence, if you secure the Blockheader in the sense that you cannot easily revert it, you achieve your goal of having transactions immutable.

Now, not only does Bitcoin make Mining artificially difficult with introducing a difficulty target, it introduces an additional 'trick' that would make it very difficult to revert transactions after a couples of blocks were added. That, as we remember, was the whole point of the exercise, to make the Blockchain immutable after it was recorded.



Simplified Bitcoin Block Chain

- The "**previous block hash**" field is **inside** the **block header**
- When the parent is modified in any way (eg fraud), the parent's hash changes and hence the child hash changes and so on
- The **cascade effect** would require that much computing power that deeper layers are practically **immutable**

The "previous block hash" field is inside the block header and thereby affects the current block's hash. The child's own identity changes if the parent's identity changes. When the parent is modified in any way, the parent's hash changes.
The parent's changed hash necessitates a change in the "previous block hash" pointer of the child. This in turn causes the child's hash to change, which requires a change in the pointer of the grandchild, which in turn changes the grandchild, and so on. This cascade effect ensures that once a block has many generations following it, it cannot be changed without forcing a recalculation of all subsequent blocks. Because such a recalculation would require enormous computation, the existence of a long chain of blocks makes the blockchain's deep history immutable, which is a key feature of bitcoin's security.
Now, miner's or the companies they run them know all these difficulties when they would try to cheat. Hence, their reasoning will be and actually is: its best to play fair and just try to with this hash computing so we can add a transaction to the Block that will send some bitcoin to us. That is the whole point, the algorithm is set up such that playing by the rules will be more advantageous than trying to cheat the network. The consensus on the state of transactions is achieved implicitly by miners choosing the longest chain, where most computing power is already lined up.

One can use metaphors to approach and understand this mechanism. One common way to think about the blockchain is for example to think of it as layers in a geological formation. The surface layers might be changeable, if you wish, but once you go a few inches deep, geological layers become more and more stable.

There would be more to say eg about so – called 51% attacks where one miner controls the majority of the network and could adjust transactions. We wont dive into this but summarize what matters for our discussion

Important to realize is that the work bitcoin demands is useful only for creating security, there is no other intrinsic value attached to hashing used here other than making the blockchain immutable. The difficulty of the work is artificially created by (1) hashing against a difficulty target and (2) linking fingerprints such that change of one parent will create a cascade effect and demand redoing younger hashes.

Proof of Understanding – how Babelchain creates consensus on Terminology

Before we go into the question how we could create network consensus on how machines would need to communicate, we will introduce the algorithm, where we will need the Translator to do his work, that is translate.

After we did all the plumbing work and Sender and Receiver connected via some transport technology and protocol, we come to the moment where we model Machines Communication essentially along the way humans would interact: A sender will ask a receiver to execute a certain action.

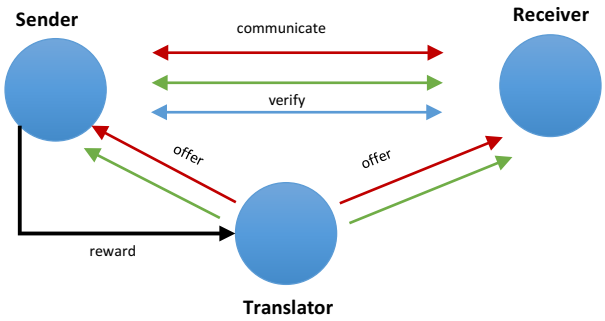
```
<ZAP_TV>
  <to>
   <receiver>TV in my chinese hotel
    room</receiver>
    <path> TV Cloud Server</path>
 </to>
 <from>
   <sender>my smart phone remote app
   </sender>
   <path> smart phone Cloud Server</path>
 </from>

<body>
    <device> TV <device>
    <command> change channel
          </command>
   <channel> 12</value>
 </body>
</ ZAP_TV >
```

The term 'Meaning' in this context be defined as consisting of three parts, where each depends on its predeessor:

- Message Format like file with fields, XML messages, csv files, binary files.
- There are variable slots in which values with Content will go.
- Action resulting from Communciation

Often, this triple is referred to as Syntax, Semantics and Pragmatics of a statement.

In Bitcoin the Miner does the work of securing the Blockchain via hashing blocks against difficulty targets. In what we will call Babelchain, the Translator does the work of proposing message formats to Sender & Receiver until they can agree to one and start communicating



- Translators suggest Message formats (Syntax) and Value Distribution (Semantics) to Sender and Receiver
- Sender rewards the Translator upon success and can try to claim back (party of) the amount from the Receiver in a separate transaction
- Sender and Receiver communicate, after the Translator suggested to them a Translation, they both can agree (digital signature) to

Translators have a key function in enabling the Communication. They do enable Communication via suggesting message formats, which they deduct from previous communication patterns. We will first describe now the Algorithm and then explain more in detail, how Translators will work.

Format Handshake

The starting point to facilitate machine communication is to agree on a message format. This is what often is attempted to standardize via consortiums, industry boards. Large vendors with a quasi monopolistic position often dictate standards. Since such standards enable applications of the eco system to interact they actually fulfill a function smaller actors, which couldn't establish a standard, rely on.
Attempting to agree on a message format without reverting to any kind of pre-defined could take place in an algorithm working along the following sequence:

1. A sender creates a message with content (key value pairs in potentially a hierarchical structure) and sends it to the Receiver

```
…
<device> TV<device>
    <command> change channel
            </command>
    <channel> 12</value>
…
```

2. Unless the Receiver ('believes to') understand the message, the message gets offered to the Babelchain Translators, including announcement of a bounty for the case the message will get translated

3. Translators generate different data formats and offer those to sender and receiver

```
…
<thing> …<thing>
    <action > …
            </action>
    <what_action> …</what_action>
…
```

4. Sender & Receiver pick one message formats they ‘believe’ they would be able to understand.

The Translator that generated the format that is first picked by both sender and receiver has won the first round. It is an open question, when the bounty will be paid out and how high it will be at this stage. Eventually, the goal of a message exchange is to trigger certain actions, just agreeing on a message format as such isn’t useful. One could therefore decide to stall payment until the content and action handshake was successfully. To incent Translators it could also be considered to pay a part at this stage and the rest only after content and action handshake were agreed upon.

The Translator will need to find a way to distinguish within the message offered by Sender between the message formats and the values in the format. One could mandate the Sender to mark the variable fields, in many cases (eg when message formats are known or can be parsed easily along rules like xml or csv) it is easy for the Translator to make that distinction by himself.

If there was no agreement achieved the algorithm will have to start again, otherwise the Algorithm will continue with the Content Handshake.

Content Handshake

A Content Handshake is the Confirmation that both Sender & Receiver ‘understand’ a message in the same way. Therefore, after a Message Format is agreed upon, there needs to be an understanding how the message format would be filled.

It will be up to the Translator to suggest a distribution of values into the agreed message format. The Translator received the initial message of the sender and he ‘knows’ to which message format both Sender and Receiver agreed. It will be up to him to make suggestions, how to distribute the values. In practice, there might be three situations. The agreed message format is one that is:

(1) quite common, a quasi standard like a certain xml format.
(2) a message format that these (class of) senders & receiver have used with minor variations successfully before
(3) an entirely new message format for these (class of) senders & receivers.

In cases (1) and (2) the distribution of message values is rather clear and most likely a successful content handshake will follow. In case of (3) the content handshake might not be successful, but at least there is a fast and clear to decide upon success or failure. We place the X amounts of values from the original message into the Y amount of value slots and run through all possible combinations until either both sender & receiver agree or they both don’t.

In the form of an algorithm this would work along the following lines:

1. Translators will compile all possible combinations of value assignments to the format.

```
…
<thing> TV<thing>
    <action > change channel
          </action>
   <what_action> 12</what_action>
…
```

2. The Receiver will pick the combination, that ‘works for him’

3. The Sender will have to agree to the solution, if he doesn’t another message format has to be found and the format handshake round of the quiz will have to start

4. If Sender and Receiver agree on the correct assignment, then whoever of the Translators offered the correct solution will get marked for the Content part of the reward..

5. If no Action Handshake is mandated, both sign the hashed messaged header and the rewards will get paid out

```
…
<thing> TV<thing>
    <action > change channel
          </action>
   <what_action> 12</what_action>
…
```

If there was no agreement on the content distribution achieved, the algorithm will have to start again. Two typical failure scenarios would be:

- sender and receiver expect different data types in the message of the slots. This would be a case, where apparently the message format handshake didn’t create the consensus between Sender & Receiver that it was meant to create.
- Sender or receiver have verification criteria between certain values, e.g. if the value field on a purchase order exceeded a certain threshold the field manager approval must have been filled. Such checks or also checks on mandatory fields are usually done as part of the application logic of any message exchange system.

These two categories in our context often might have the underlying message format as the common root. For example the manager approval might be existing in a field in the message but the receiver didn’t ‘get’ the message right and hence, what looks like a conventional application logic error might in reality be an issue with an underlying message format.

In case of a successful Content Handshake, we will have to make two interconnected decisions. In case Sender & Receiver are both confident that Format and Content Handshake were successful, the Translator would receiver the full bounty. It would be a question of the Implementation, but we would probably mandate that the Content Handshake can be facilitated only by the Translator, which mediated the Format Handshake in the first place.

This is more efficient because the Translator will 'know', what he 'had in mind' with the suggested message format. It would be more 'just' and hence render a better incentive structure, because the Content Handshake would be easier to generate than the Format handshake and we would therefore give the Translator, which won the Format Handshake, the opportunity to gain more rewards.

Usually, content handshakes will be successful for two above listed classes of handshakes, when the Format handshake generated a quasi-standard format or if senders and receivers communicate in a way in which (these classes of senders and receivers) usually communicate. In case there was an entirely new message format introduced for example because sender and receiver never communicated, at least not concerning a certain topic, the Action Handshake will be useful.

An important question is, how Sender and Receiver will decide, if a message format offered by a Translator would work for them. A simple way would be to trust upon the standard error mechanism any System already has: If the message is processed, format and content handshake were successful. If the message wasn't processed, one of format and content handshake wasn't successful and the Translator would receive back an error message from one of the Systems. This straight forward approach would allow to not change existing Systems, they would communicate with the Babelchain like they do with any other Systems. In many cases though, it could be useful to equip both Sender and Receiver with more sophisticated ways to interact with the Babelchain. Particularly helpful would be, they give a standardized feedback, why a handshake hasn't succeeded. This would allow the Translator to work more efficient and lower costs for the Sender.

Sender and Receiver would need a unit, not existing in any current System, to interact with a Translator, in case an Action Handshake was initiated.

Action Handshake

An Action handshake is optional, it can be initiated by sender or receiver and be advised by the translator. Choosing for this handshake will depend on probability to fail and the risk implied upon a misunderstanding.

Imagine, there was an entirely new message format and content distribution introduced for a sender and a receiver that never communicated with each other, at least not via the agreed format and content. On the other hand, the sender demands rather sensitive actions like e.g. opening a houses front door upon typing a command on a smart phone or launching an engine or submitting an expensive purchase order.
One of sender or receiver might not 'feel comfortable' with assuming there is a well – understood communication path opened already. In that case sender or receiver can demand that there will be an additional action handshake.
Also, the translator could advice sender or receiver to initiate this additional check, the translator might while generating the message realize to be making a proposal based on relatively few data. The sender and receiver didn't communicate before or (s)he cannot find similar message exchange formats in the blockchain and therefore sees a significant chance for a misunderstanding, no matter what was agreed before hands.

1.Sender or Receiver can choose to open this round before continuing. Translator can advice to initiate this action before continuing.

2. Receiver upon his or Senders initiative has to offer evidence of the action, if it would be executed.

3.Evidence will be checked by machines or humans from the Sender. Upon success, both sites sign the agreement.

A good way of understanding what should happen during this handshake in case humans would need to verify the action is to describe it as a one time human handshake.
Imagine the IT Administrator of a large cooperation realizes a format and content handshake was successfully concluded, where his employer is the sender and the original message format is that of a sales order for ordering a large machine of another cooperation. The value exceeds a certain threshold and he also realizes, that the supplier is a new supplier with whom they never worked before. He realizes all that, because the Translator has sent him a notification and warning email, indicating there was a successful handshake but receiver and message format are unknown yet to him and he therefore suggests to double check the format and content agreement. He will call up the IT administrator of the Receiving party and discuss with him the message format the sending and receiving machine agreed upon. The IT administrator of the receiving party will have to double - check the agreed message format and verify with the Sender that the order is really according to what the Sending party wants. The IT Administrator would do this by his understanding of their internal application and how it would interpret the message format or by executing internal tests. At the end, they would send a simple, often digitally sugned email, stating that the machine agreed message format is approved by them. They would also update this status in their Systems, notifying sender, receiver and translator that they can continue. As a result, not only will order the go through correctly but the Translator will place an example of a working message format into the Blockchain allowing in the future seamless communication, without the need for any human interaction.

Action Handshakes can as well be digital and automated in as far as we can rely upon machine intelligence to verify evidence of ‘would-be-action’ and the involved risks seem acceptable. Imagine a sender is using a smartphone, wanting to send a signal to a thermostat switching on the air - conditioning. Provided these devices communicate over the internet they could agree on format and content, but since the sender never communicated with the receiver before hands, (s)he demands a proof of action. The receiver could send him over a digital picture of a thermostat with a certain temperature or the air-conditioning switch placed on ‘on’, he could also share with him internal representations of his machines states regulating the temperature. The sender might be satisfied with a jpg representation of a thermostat, if he can easily scan and interpret it, also because the risk in case of a misunderstanding would be relatively low. Ideally, if such Systems are smart enough, they would feedback the success status, in case the human using the smartphone updates the System if in his view the handshakes were successful (that is, temperature going down) or not.

Note that the Translator's role here is giving advice and recording the result of the Action Handshake. As part of an Implementation it would need to be considered if Translators receive a bonus or malus upon success or failure of action handshakes and their role in advising them or not. With the potential complexity and necessary machine intelligence of verifying action handshakes automatically it would also be necessary to give Translator's a role in helping to verify these handshakes. Sender's or Receiver's could ask Translators to verify the action evidence and give them advice and pay out rewards for those.

Note also, that potentially all handshakes in could fail. There might simple be no combination of format, content and action handshake satisfying both sender and receiver. For a large amount of fairly common message patterns, that is not very likely but it is a possibility.
One 'last attempt' a Translator could undertake is establishing a permanent mediator position. (S)he realizes, he cannot come up with an agreed Format and Content Handshake, but does see that Sender and Receiver both agreed to several format and contents, only not one to which the other party also agreed. Since the translator suggested these formats and contents, he would 'know' how to map these two. Therefore, he could just suggest the Sender and Receiver they stay with ever format they like and he maps them for them. He could ask for extra charge for this service.
This is not a desire able approach because Sender and Receiver permanently depend on the mapping capabilities of Translator, even if the mapping would get written into the Blockchain for future reference. The overall goal of this form of consensus algorithm on understanding is actually to avoid middlemen. In certain situations this could be however the best and only option to establish communication. In such situations typically the Translator would advice an Action handshake to ensure, the mapping suggested was correct. For Sender & Receiver it could actually lead to a redefinition of their System internals and communication formats to avoid being depend on such a mediation instance and avoiding extra payments.

How Translators 'do it' – Blockchains as a training Set for Machine Learning Algorithms

Translators for Babelchain play the role that Miners play for Bitcoin. They provide the distributed shared infrastructure and via the work they do in order to be eligible for rewards, they create whatever the network wants them to do: In the case of Bitcoin that is consensus on the status of transactions, in the case of the chain we want to suggest, that would be foremost creating a common language between Machines while keeping a version of the immutability of the Transactions. We will come to the immutability feature later and try to explain here, how message formats that have a good chance to create the described handshakes successfully can be generated.

A 'common sense' approach to building a Translator would be as follows:

(1) Build a list of
    a. common Internet, Enterprise and other message exchange formats
    b. Categorize them along Vendor, Technology Platform, Industry, Usecase a.o.
(2) Ensure that Sender & Receiver exchange any information that could be relevant for finding a common message:

   a. Which parts of the message are variable (content) and which are fixed (format)
   b. Categories like Industry, used Technology Vendor they fall under as listed in (2)
   c. Relevant contextual Information, e.g. smart phone owner trying to change temperature is located in a hot country with current temperature x amouny Fahrenheit and so on

(3) Ensure the Translator has a message parsing mechanism, a.o. allowing to:
   a. differentiate format from content
   b. categorize format along the categories described in (1b)

(4) Ensure the Translator has message creation mechanism, a.o. allowing to
   a. Accept information shared by sender and receiver (2) as input
   b. pick a message from common message exchange patterns from table (1a) with the help of the categories listed in (1b)
   c. adjust these messages where necessary with module (3)

(5) Ensure the Translator has a content distribution mechanism, allowing to take all values from the message initially shared by sender and placing them in all possible combinations in the value slots provided in the message format suggested by the sender

(6) Ensure the Translator keeps a log of all handshakes and
   a. updates the logs after each (un-) successful handshake with as much information as possible
   b. for any new handshakes reads the log and tries to deduct in a smart way what handshakes could work this time

For many sender / receiver communications, there is arguably a reasonably good chance to find a common way of communicating with such an algorithm. Such a table would work well for the class of messages that are (variations of) common message exchange formats, that large vendors, dominating technology platforms or Industries choose. For entirely new message formats, we would need a sophisticated mechanism to create message formats, based on what is known, in many cases we might need human intervention in the start to direct the machine algorithm towards the right track.

The suggestion for the the System we introduce now is to design the Blockchain as such a table (1) (2) and use Machine Learning Algorithms, more precisely supervised statistical and logical Machine Learning algorithms to implement the relevant algorithms (3), (4), (6). What Machine Learning refers to as ‘features’ would be categories and context information, which we list in (2b,c).

The Blockchain would contain a list of all *successful* or *unsuccessful* (in machine learning terms these would allow forming a 'supervised training set') handshakes together with a large list of **features** that are relevant.

	… <device> **TV**<device> <command> **change channel** </command> <channel> **12**</value> …
Identity Sender	Owner smartphone
Identity Receiver	TV Hotel
Vendor Sender	Apple iPhone
Vendor Receiver	Samsung TV
Technology Sender	proprietary xml
Technology Receiver	Java xml
…	
Location message	Hotel, front of TV
Time of the Day message	noon
…	
Sender Acknowledge Format	Yes
Receiver Acknowledge Format	yes
Sender Acknowledge Content	yes
Receiver Acknowledge Content	yes
Sender Acknowledge Action	yes
Receiver Acknowledge Action	yes
Format Handshake	**<entity>** …**<entity>** **<action>** … **</action>** **<value>** …**</value>**
Content Handshake	**TV** **Change channel** **12**
Action Handshake	Not requested

- Blockchain Transaction Log forming a training Set for a supervised Machine Learning Algorithm
- Creates a (statistical) model of Machine Message Communication Patterns
- Allows applying a predictive message format algorithm
- **Features** are all attributes that impact or allow to read, which message formats can be useful for closing handshakes successfully
- *Supervised Learning Set* as it records successful and unsuccessful handshakes

The learning algorithm would then use this table to predict the message formats for new handshakes. There are numerous machine learning algorithms implemented, that fulfil comparable tasks. One of the nicer examples is swift-spear, which is a statistical machine learning algorithms that creates what looks like a Shakespeare Sonnett based on the statistical analysis of existing Shakespeare Sonnetts.

There would need to be a distinct level of data obfuscation for this blockchain learning set : most real values would get masked such that it is still relevant as a training set, and all data would be secured with private public key cryptography. Permissioning such a system or some record sets could be a legitimate option for some usecases, we will come to that in the following section.

Typically Translators would implement their functionality on any platform of their choice as log as they connect to the Babelchain, there would be much more to investigate here. For this investigation relevant would be that we need to design the Blockchain such that it can serve as a training set and implement relevant predicitive machine learning algorithm to create a Translator.

Implementing an immutable Babelchain

One important question for Implementing such a System would be, which role we would have the 'proof of understanding' algorithm play. One way could be to see this algorithm like regular functionality of an Application. Like some applications allow entering and storing Client Data or navigating through traffic, proof of understanding would allow translating messages. Using Proof of Understanding as a System Feature could in principle be implemented with any platform, choosing a platform that allows binding agreements and crypto payments would be a good choice and therefore a natural choice could be for example choosing smart contract on bitcoin or a platform like ethereum.
A different approach, the one we want to follow here, would be to design a System such that this algorithm will get a role in generating the incentives necessary to render the Infrastructure and secure the Transactions. This could be done as a full or partical replacement of a proof of work algorithm. Next to 'creating' the System and 'securing' the Transactions, an important aspect would be that we might able using the energy such a System needs in a smarter way. With the simplification (probably incorrect), that hashing and machine learning algorithms use a similar amount of energy, we could list the options as follows:

	Description	Energy Waste	Goods & Bads
Smart Contract	announce a competition, a bounty smart contract where anyone solving the translation task will get a cryptocurrency award like ethereum paid out	**High**: as high as the hosting blockchain e.g. ethereum No smart reuse of the work translators do	+ Ideal for prototyping - Low scalability
Prereq for Proof of Work	whoever generated a valid proof of understanding (there can be several) can start the proof of work Hashing puzzle and finding a nonce will include the agreed message, hence proof of work can be started only after proof of understanding Agreed Message will be signed by sender and receiver, so anyone in the network can check	**Smarter, not lower:** Machine work will be used for a useful task Combined work overall would be similar to bitcoin blockchain	+ Re-use proven Bitcoin approach to make the Blockchain immuteable + high scalability - Energy consumption and waste as high as Bitcoin *- To be clarified, how the effort of proof of understanding can be measured and how proof of work difficulty target could be adjusted dynamically*
Replace Proof of Work	In any transaction system between connected parties having a common terminology is a pre req to make useful transactions. With the size and amounts of usecases in IoT the 'one agreed transactio data format' strategy of Bitcoin will not be feasible.	**Smarter and lower**: Machine work will be used for a useful task Effort for Translation will decrease with the learning effect	+ high scalability + use of machine work to cover 2 goals: translation & immutability *To be clarified how Blockchain Immutability will be achieved, even more with learning effect and decreased work* *Potentially smart combination of machine work with consent sender & receiver to message reverts*

Before we discuss the latter two options more in detail, we might want to investigate, if we really need an immutable Blockchain. Bitcoin implements payment transactions and for that usecase, undoubtedly one needs a very strong version of immutability to avoid doublespending. One should note though, that Bitcoin even for that usecase has a rather strong concept of immutability. This is an implication of the feature to not use any access management and know their 'real-life' identity to track and punish or exclude them from the System in case they try to cheat. Anyone can open a wallet, there is no 'bitcoin – corporation'

that like banks do want to know about your identity and might want to exclude you in case you try fraud the System.

Why Immutability, why not 'just' Identity & Access Management ?

One way to approach the Implementation would be to 'do away' with any type of Consensus and 'Proof of X'. Overall, what matter is that we can translate messages and communicate, so we could just take the machine learning part, provide some training sets in a transaction log and then setup the System in a 'good old fashioned way': One registers, provides credit card and company details, then one logs on and identifies and authenticates oneself. In case you act malicious, be aware whoever hosts the solution will audit and monitor you and eventually track you down. No need for 'Proof of X' and Consensus Algorithms in this setting.
Technically, this is a possibility but the 'real beauty' of the Bitcoin Proof of Work Consesnus algorithm lies within its economical and social implications. This Point of View defended here could be subject to debate and is subjective but for the sake of this investigation it is the motivation to investigate the possibility of implementing immutability and consensus.
Introducing Identity & Access management with all connected auditing, monitoring, revenue split and potentially sanctions is in reality just a small step away from introducing some version of 'online store X' with some vendor or consortium dominating the closed network. Consider what great of a difference it is for a small vendor trying to compete with a 'big fish' between offering him an open and permission-less communication chain, where anyone can participate and can communicate with anyone to a 'Online Store X' solution. In fact, any such centralized solution also invalidates to some extent the usage of sophisticated learning algorithms to establish communication. If someone dominates a network, we could just as well follow this body's friendly suggestions which standards to use. Even if they allow us to use our own terminology, it would be more a question of convenience for us and on their side potentially efficiency gains to use an adaptive learning algorithm.
In fact, 'bringing order' to the multi language 'tower of babel' is one of the main values such quasi - monopolistic structure offer. Delivering this service in a different way, potentially will also considerable weaken the closed and promote an open network with more innovative power.
Imagine, small independent players would bring up social networks, maybe with different attributes like levels of data protection or exposure to online ads. Currently, such networks all face the problem, that not-everyone else is on the same network. It is in fact arguably the most valuable function a monopolist, whose name I won't mention here, has that he allows to communicating to nearly everyone else in the social space – provided he is on this network, including all the rules such a monopolist believes he should set up. However, if there was a glue between smaller, more diverse social networks, allowing all of them to communicate to anyone else wanting to communicate, that would take away the need to join a monopolist to communicate to a large amount of people. A similar argument could be made for online app stores of the large vendors dominating the market and creating closed networks.

Proof Understanding as a Pre-req for Proof of Work

The basic idea is that, any translator can start a 'proof of work' only after he completed the 'proof of understanding' successfully, that is sender and receiver have agreed to the message

format, content and potentially action handshake. The 'Proof of working' would have to hash a.o. the agreed message signed by sender and receiver. As for bitcoin the winner would receive a difficulty target and other translators could verify him to be the winner of the competition by checking both the digital signatures and the fingerprint.
The intention would however be that we do not increase the overall amount of work done, but rather 'cut out' the machine learning part and require less 'proof of work' effort. That would require:

(1) we can measure the effort that went into the proof of understanding, a simple way of doing this could be the time needed until the translator came up with an accepted proposal
(2) we can adjust the difficulty target for 'proof of work' dynamically and reliably, such that it will be reduced by the amount of 'proof of understanding' done. This would be possible eg by statistical insight on how fast the network will be able a 'proof of work' task against a certain difficulty task
(3) We can compare the effort for 'proof of understanding' and 'proof of work'. 'time spent' is a simple common denominator, more sophisticated mathematically proof ways of measuring might be possible

Trying to revert such a transaction would then contain re-asking consent of sender & receiver to a then different handshake and redoing the proof of work.
One could strengthen the 'proof of understanding' irreversibility by demanding that not only sender and receiver have to agree to a new handshake but also other members of the network. The reason for that could be that, sender & receiver decide to roll back a transaction because changing it could be in their advantage but go to the expense of a third party. Imagine company A is about to go bankrupt and decides with company B that they adjust an old order concerning shipment of some bicycles into shipment of several cars. The liquidator of Company A couldn't get hold of the value, while company A and B could have an agreement to split the win from selling these cars. This could be for example very successful Translators, which because of their apparent skill set represent a trusted instance. Another common way of finding such trusted entities is to define Trustability via the amount of currency won and still in possession, since these entities will have an interest to keep the network mutually trusted and alive.
In a strict (maybe even puristic) interpretation of the advantages of a 'proof of work' mechanism one might see in such trusted entities an incarnation of those 'skilled and rich' trying to dominate a network. For many practical purposes and usecases the combination of 'signing & working' might however deliver sufficient degree of immutability towards the System we suggest.

Replacing Proof of Work with Proof of Understanding

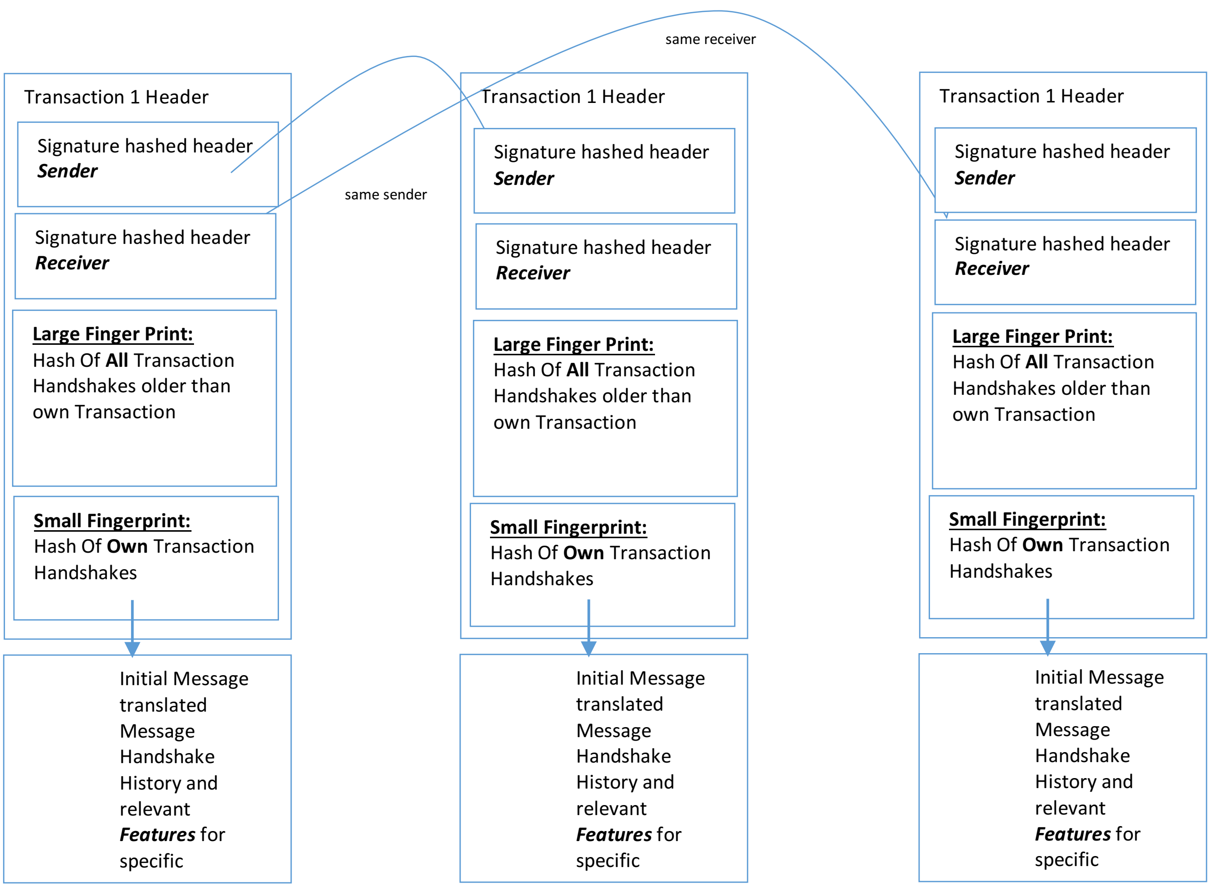
Another way of approaching replacing 'Proof of Work' with 'proof of understanding' would be starting with the cascade effect bitcoin causes when trying to revert transactions.
Bitcoin suggests gathering Transactions ins Blocks and then linking these blocks in a way they any change of a parent block will change its successor block. This in a sense is an artificial way of grouping transactions, many transactions are 'inherently' connected (person A pays bitcoin

X to person B, person B pays bitcoin X to person C) but bitcoin doesn't use that link to link (blocks of transactions) in order to cause a cascade effect.
An important observation here is that in the System we suggest the glue that naturally links handshakes and transactions is the blockchain training set. Handshakes are deducted from the training set, hence changing the training set and its containing transactions can provide a means to impact transactions such, that subsequent transactions would need to change if a handshake in a blockchain was adjusted.
We could therefore use hash fingerprints to link transactions to the underlying trainingg set (the set of all successful and unsuccessful handshakes up to the moment the transaction handshakes are happening) into the Transaction Header:

small fingerprint:
- hash fingerprints of their own Handshake Negotiations

large fingerprint:
- Handshake Negotiation history existing at the time of their handshake
- Fingerprint will be relative to a point in time or length (amount handshakes) of the Blockchain

While this is a strong and explicit coupling note that Transactions are also loosely coupled by the features listed in the Blockchain. One can easily group all messages of a certain sender or receiver and one can also group handshakes along other features like e.g. the vendor, where the signals were sent and alike.



The opportunity to group transaction handshakes against features will render a structure similar to a social network with persons having online conversation and topics (#hashtags), which can be leveraged by the learning algorithm to make proper predictions on message formats.
The strong coupling via introducing fingerprints has the desired cascaded effect in case one wants to revert a transactions.

Weak Immutability:

- Change of Handshake will change the small of the transaction
- The work, which needs to be redone:
  - Re-do translation work for the adjusted transaction. In case only a value was changed this work might be neglectable
  - Re-ask consent from senders and receivers as a format or content handshakes was changed
  - Header needs to get hashed, optionally against a difficulty target

Strong Immutability:

- Change of Handshake will change the Large Fingerprint of all younger Transactions
- The work, which needs to be redone:
  - Re-do translation work for the adjusted transaction. In case only a value was changed this work might be neglectable
  - Re-ask consent from all senders and receivers who handshake after the changed transaction since their fingerprint is invalidated
  - Headers of all sender and receivers who handshaked after the changed transactions need to get hashed, optionally against a difficulty target

Usage of a hash against a difficulty target can be necessary to rule out the possibility that (all affected) sender & (all affected) receivers conspire to revert transactions on expense of a 3rd party. To implement this properly we would need to define what a previous transaction would be so we can incorporate its fingerprint in the current transaction header. That could be the transaction a translator received just around the time before (similar to bitcoin just gathering transactions) or a transaction that is 'inherently connected to the current transaction, e.g. the last transactions of the sender receiver or a certain topic (set of features in the training set) prominent in the network.
The larger the network, the less likely it is that all affected network participants will be 'willing to cheat'. Usage of a hash against a difficulty target can therefore be necessary in the network startup phase when the network is small and the cascade effect for asking re-signing can be neglected.

Note, that we use hash algorithms but that the 'thing' that causes too much is not re-hashing but the consent that the machine learning algorithm generated, because we have to ask now all senders and receivers which handshaked after the reverted transaction to sign their handshakes again to have valid transactions.

Imagine a case where Sender and receiver try to cheat the Translator, they reject a message format and content handshake but in reality shortly afterwards they do use this in their

communication. It is not entirely impossible that sender & receiver would do this outside the network and no one could detect this. For many cases, communicating outside the network might cause more effort than paying the bounty. In any case, in an Implementation it could be wise to allow to check upon the network, if the Translation he offered was used even though rejected before. In that case we should give the Translator the right to retrieve a bounty from the sender just as if he had agreed in the first place. Knowing this mechanism exists should in most cases lead to a situation that it is never applied.

This brings us to an important difference between the 'work done' for bitcoin and the work done for babelchain. Bitcoin can use an objective 'razorsharp' criteria, which is easy to check by anyone. It is an attribute of Hashes that are easy and unambiguously to verify, they are designed this way.
If we come to complex and potentially ambiguous tasks like for example translating languages defining success can get more challenging than deciding on a hash fingerprint. One could revert to a 'trusted middleman' for example for translations of natural languages we could consult a certified & sworn translator in a court setting. However, we started out avoiding 3rd parties in our negotiation process.
For many such types of works success is best defined as a consent between Client and Worker, if the work had been done correctly, documented by a (digital signature).

Comparing Proofs

The main point to realize here is that Hashing and Predictive Machine Learning algorithms are different in nature and purpose. Hashing is among other things used to generate digital fingerprints to check validity of data after transferring data, machine learning algorithms have a wide area of usage from predicting house prices to understanding diseases.
Some of the differences we discussed in the 'Proof of Work' and 'Proof of Understanding we discussed in here can be summarized as follows.

	Bitcoin Proof of Work	Babelchain Proof of Understanding
Purpose	Create an unmuteable Blockchain for Financial Transactions	Create common message formats to allows machines to communicate
Quiz	Find a value for the nonce that results in a block header hash that is less than the difficulty target	Find a format (content, Action) which sender and receiver approve
Deterministic	Yes, for any input (arbitraty length) eg SHA 256 will produce always the same fixed length output	No, sender and receiver could agree on different formats for the same messages asked on different times as long as they both believe the format will work for them
Predictable timeframe	Yes, statistically	*For many messages probably yesy statistically, but there will be non- translatable messages*
Can it fail ?	Statistically not	Yes, sender and receiver could never reach an agreed message format (content / action)
Money supply	decreasing	constant, never ending
Hard to find solutions	Yes, depending on difficulty target	Case by Case. Will generally get easier with the learning effect of the network. There can always be very hard or untranslatable messages.
Easy to verify solution	Yes, feature of hash function	Partially, sender & receiver have to agree and sign the message, which can be verified. However deciding if a message is useful is subject to 'subjective' decisions and not a mathematic and objective algorithm.
Quiz Difficulty adjusting over time	no	Yes, network learning effect implies that translation will get easier. There can always be very hard or untranslatable messages.
Difficulty of winning reward	Increasing (money supply decreasing and quiz difficuty unchanged)	*Decreasing (learning effect and constant money supply). This raises the question, how to reward Transators for (too) easy translations over time*
Payer	Mining Reward: Bitcoin network Transaction Fee: the Sender of Bitcoin	Translation Reward: Sender (try claiming part from Receiver) Transaction Fee: Sender (try claiming part from Receiver)
Cheating possible	no	Yes, senders and receiver might pass on more features outside the network to preferred translators
Immutability	Yes	'weak immutability' per default as sender and receiver need to agree to changes 'strong immutability' depending on either combination with existing proof of work / proof of stake or smart usage of training sets to cause a cascading effect

The underlying mathematics of both algorithms differ significantly. Hashing is discussed in lecture on Cryptography while Machine Learning is analyzed in Linear Algebra. A more profound comparison could start from looking at the foundation mathematics of these two algorithms.

Conclusion

In this paper, we tried describing a way of building a Blockchain with a specific target, enabling machine communication. Solving this problem is not only useful for different areas of Information technology, it also could help leading the way to dissolving quasi - monopolistic information silos via connecting smaller players into a large enough communicating network. Combining this Machine Intelligence with Blockchain Technology and a 'Proof of X' mechanism delivering a version of immutability is a necessary step to allow an open and un-permissioned eco system, even though there are legitimate usages to restrict access to sensitive parts of such networks.

We did not touch the topics of security and scalability. Mostly because, we have nothing new to add to the discussion. Obviously, security in a distributed System is a field of debate and undoubtedly the described System would need to use mechanisms like data masking beyond private public key cryptography to mask transactions in a learning set accessible to all Translators. Moreover, we make the (admittedly bold) assumption that these questions can be worked out just like they were for other Systems, provided a Systems proves to be overall valuable enough to invest the necessary work.

Also, we didn't discuss money supply and the effect of a learning proof of understanding algorithm. The described network would learn and get better, for many translations the result will be known and the 'job to do' rather trivial or sender & receiver don't need help at all. There might still be the need for translators to help verifying rather complex action handshakes once in a while, especially if new entities enter the network, but overall also this job and income resource might diminish.
This is a similar situation as for bitcoin, where the mining rewards are (per design) getting lesser and lesser. Bitcoin offers the miners in exchange fees for enabling transactions. Translators as well could get paid for a number of services, as having applications communicate consists of more than translating messages into each other. There is the technical connection, the storing of logs and transaction data, there could be reporting & analytics over the chain and there could be monitoring of transactions. Translators could be transformed to becoming general 'network stewards' to enable and facilitate communication, guaranteeing them a stable long term income and as a consequence keeping the network supported.

The approach that we followed is based on the believe that Bitcoin's mechanism to secure Transactions without any central 'trusted Middle man' can lead the way to build other than financial networks. It's uniqueness lies in dispatching work to specialized 'workers' (Miners or Translators) that get paid for their services. They will create an Infrastructure independent of any dominating entity and support creating consensus on and immutability of Transactions. The specific problem of Machine Communication we suggest to solve is of relevance for any Transaction System like a Blockchain, as any such System always has to solve the question of am agreed Communication Method.

Literature (draft)

Bitcoin: A Peer-to-Peer Electronic Cash System Satoshi Nakamoto

A Next-Generation Smart Contract and Decentralized Application Platform Vitalik Buterin

Mastering Bitcoin Andreas Antonopoulos

Presentation, Linux IoT Summit 2016, Benedikt Herudek

Article Service Techmagazine on Integration and Blockchain, part 1, part 2, Benedikt Herudek