THE TREND TOWARDS BLOCKCHAIN PRIVACY: ZERO KNOWLEDGE PROOFS

GEORGE SAMMAN

George is a blockchain and cryptocurrency consultant and advisor to global financial institutions and startups. He recently co-authored a seminal report on blockchain architecture with KPMG: http://bit.ly/293Tv9o. George is Entrepreneur in Residence for Blockchain at Tyro Fintech Hub and Startupbootcamp NYC. He also writes a blog on blockchain technology and use cases at sammantics.com. George also co-founded BTC.sx, now magnr, a bitcoin trading platform in 2013. He is also a former Wall Street Senior Portfolio Manager and Market Strategist as well as a technical analyst. George holds the Chartered Market Technician (CMT) designation.

GILBERT +TOBIN

CONTENTS

1 STRIKING A BALANCE: BLOCKCHAIN TRANSPARENCY VERSUS COMMERCIAL CONFIDENTIALITY

One of the key trends in the blockchain world, particularly for financial services and capital markets, is the design of private blockchain solutions to address the need for privacy and confidentiality.

This is driving the development of various cryptographic techniques to encrypt transaction data from everyone except the parties involved. Many blockchain solutions are using advanced cryptographic techniques that provide strong mathematically provable guarantees for the privacy of data and transactions.

However, when you build for privacy and confidentiality, there are trade-offs that come with that. Mainly you lose ***transparency***, which was the major feature of the first blockchain: Bitcoin.

(a) Starting at the beginning: public blockchain was designed as a transparency machine

For public blockchain:

- Computers are distributed and no one entity controls the network.
- Anyone can be a validator and anyone can write to or read from the network.
- Clients and validators can be anonymous, and all the data gets stored locally in every node (replication). This makes all transaction data public.
- The security of Bitcoin is made possible by a verification process in which all participants can individually and autonomously validate transactions.
- While Bitcoin addresses the privacy problem by issuing pseudonymous addresses, it is still possible to find out whose addresses they are, via various techniques.

(b) Moving to private blockchain world

In the private blockchain world, we are seeing the polar opposite - decentralization and transparency are not necessary for many of the capital markets use cases.

In private blockchain world, the focus is on how to preserve privacy and confidentiality, while still achieving speed, scalability, and network stability.

At a minimum, the nodes ***must*** be known in order to satisfy regulatory and compliance requirements – ensuring that legal recourse is still available, even between parties who don't necessarily trust each other.

What is important is privacy and confidentiality, latency (speed) and scalability (able to maintain high performance as more nodes are added are added to the blockchain).

Examples of private blockchain solutions that can achieve privacy and confidentiality include:

- **Encrypted node to node (n2n) transactions:** the only entities to receive data are the two parties involved in the transaction. In many of these systems, there are also opt ins for third party nodes (regulators) to be a part of the transaction.
- **Designated block generator:** Systems can use one designated block "***Generator***" to collect and validate all of the proposed transactions, periodically batching them together into a new-block proposal. Consensus is provided by the Generator, which applies rules (validates) agreed to by the nodes (chain cores) to the block and designated block signors.

In these systems, decentralization is simply not necessary because all of the nodes are known parties.

2 THE IMPORTANCE OF STRONG, DURABLE CRYPTOGRAPHIC IDENTIFICATION

(a) (What is Cryptography and Encryption?

With privacy and confidentiality being pivotal, encryption has become a major focus for all blockchains. Many of these solutions are using advanced cryptographic techniques that provide strong mathematically provable guarantees for the privacy of data and transactions.

In a recent blog post titled "A Gentle Reminder About Encryption" by Kathleen Breitman of R3CEV, she succinctly provides a great working definition:

> Encryption refers to the operation of disguising plaintext, information to be concealed. The set of rules to encrypt the text is called the encryption algorithm. The operation of an algorithm depends on the encryption key, or an input to the algorithm with the message. For a user to obtain a message from the output of an algorithm, there must be a decryption algorithm which, when used with a decryption key, reproduces the plaintext.

If this encryption uses ciphertext to decrypt this plaintext, you get homomorphic encryption and this (combined with digital signature techniques) is the basis for the cryptographic techniques discussed in this article.

+ Homomorphic encryption allows for computations to be done on encrypted data without first having to decrypt it.
+ In other words, this technique allows the privacy of the data/transaction to be preserved while computations are performed on it, without revealing that data/transaction. Only those with decrypt keys can access what exactly that data/transaction was.

Homomorphic encryption means that decrypt(encrypt(A) + encrypt(B)) = A+B. This is known as homomorphic under addition:

A computation performed on the encrypted data when decrypted is equal to a computation performed on the encrypted data.

3 CRYPTOGRAPHIC TECHNIQUES FOR MAINTAINING PRIVACY AND CONFIDENTIALITY

The key question being asked here is:

How can you convince a system of a change of state without revealing too much information? After all, blockchains want to share a (change of) state; not information.

On a blockchain, a business process may be at state X and then move to state Y. This needs to be recorded and proved, while preserving privacy and not sharing a lot of information. Furthermore, this change of state needs to happen legally, otherwise there is a privacy breach.

As a result, we are now seeing the emergence of various cryptographic techniques, some old and some new, to encrypt transactions and associated data from everyone except the parties involved.

Cryptographic techniques such as zero knowledge proofs (ZKPs) - which use different types of homomorphic encryption – are able to separate:

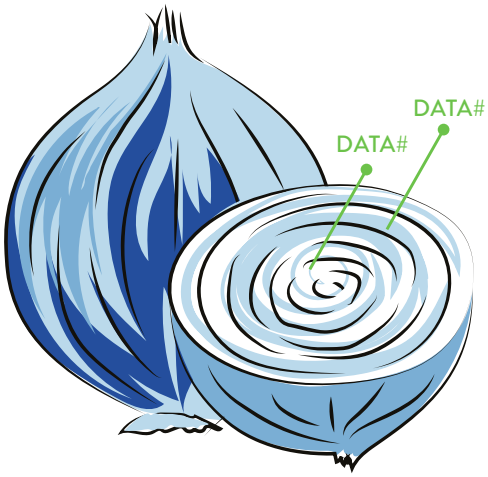
1) reaching a conclusion on a state of affairs; and

2) the information needed to reach that state of affairs,

thereby showing that the relevant state is valid.

Outside of blockchain, there are various examples of homomorphic encryption in practice.

+ CryptDB is an example of system that uses homomorphic encryption and other attribute preserving encryption techniques to query databases securely. It is used in production at Google and Microsoft amongst other places.
+ It does have limitations though: you have to define the kinds of queries you want ahead of time and it is easy to leak data. CryptDB provides confidentiality for data content and for names of columns and tables; however CryptDB does not hide the overall table structure, the number of rows, the types of columns, or the approximate size of data in bytes.
+ One method CryptDB uses to encrypt each data items is by oncoming. This allows each data item to be placed in layers of increasingly stronger encryption.



Onioning: allows each data item to be placed in layers of increasingly stronger encryption

We focus below on some of the key cryptographic techniques for blockchain: Zero Knowledge Proofs, zk-SNARKs, Hawk, confidential signatures, state channels and homomorphic encryption.

(a) Confidential Transactions

Gregory Maxwell designed a cryptographic tool (CT) to improve the privacy and security of Bitcoin-style blockchains. It keeps the amounts transferred visible only to participants in the transaction. CT's make the transaction amounts and balances private on a blockchain through encryption, specifically additively homomorphic encryption. What users can see is the balances of their own accounts and transactions that they are receiving. Zero knowledge proofs are needed to demonstrate to the blockchain that none of the encrypted outputs contain a negative value.

The problem with Confidential Transactions is that they only allow for very limited proofs as mentioned above. zkSNARKs and Zero Knowledge Proofs (ZKPs) which will be described in detail below, allow you to prove virtually any kinds of transaction validation while keeping all inputs private.

(b) Zero Knowledge Proofs (ZKPs)

Zero Knowledge Proofs (ZKPs) are not new. They were first conceptualized in 1985 in a paper "The Knowledge Complexity of Interactive proof Systems." A ZKP is a cryptographic technique which allows two parties (a prover and a verifier) to prove that a proposition is true, without revealing any information about that thing apart from it being true. In the case of cryptocurrencies and blockchains, this will generally be data about transactional information.

"A zero-knowledge proof must satisfy three properties:

Completeness: if the statement is true, the honest verifier (that is, one following the protocol properly) will be convinced of this fact by an honest prover.

Soundness: if the statement is false, no cheating prover can convince the honest verifier that it is true, except with some small probability.

Zero-knowledge: if the statement is true, no cheating verifier learns anything other than this fact. This is formalized by showing that every cheating verifier has some simulator that, given only the statement to be proved (and no access to the prover), can produce a transcript that "looks like" an interaction between the honest prover and the cheating verifier.

The first two of these are properties of more general interactive proof systems. The third is what makes the proof zero-knowledge."

(c) zk-SNARKs

A zk-SNARK (zero-knowledge Succinct Non-Interactive Arguments of Knowledge) is a Zero Knowledge proof that is a way to prove some computational fact about data without revealing the data.

Zk-SNARKs are the underlying cryptographic tool used in Zcash and Hawk, both of which are building blockchains with ZKPs (as outlined below).

+ In the case of Zcash, these SNARKs are used for verifying transactions.
+ In the case of Hawk, these SNARKs are used for verifying smart contracts. This is done while still protecting users privacy.

A zk-SNARK is a non-interactive zero-knowledge proof of knowledge that is succinct and for which proofs are very short and easy to verify. They can be thought of as little logic circuits that need to generate a proof of statement to verify each and every transaction. They do this by taking a snapshot of each transaction, generate a proof and then need to convince the receiving side that the calculation was done correctly without revealing any data except the proof itself. The basic operation of a SNARK execution is a coded input into this circuit which can be decrypted.

Since zk-SNARKs can be verified quickly, and the proofs are small, they can protect the integrity of the computation without burdening non-participants. It should be noted that this technology is just now starting to mature but still has limitations. They are very CPU intensive to generate proofs and it takes up to 1 minute to generate new proofs, so scaling is still an issue that needs to be resolved.

The very first data points for zk-SNARKs will be Zcash, which is a combo of distributed state and proof that you own the assets.

(d) Zcash

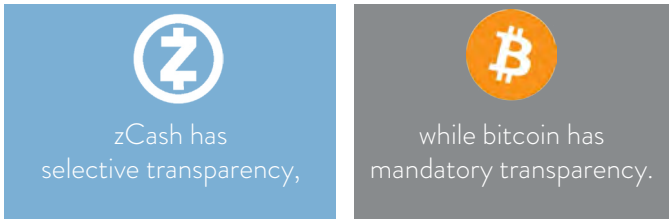
Zcash is an encrypted open, permissionless, replicated ledger - a cryptographic protocol for putting private data on a public blockchain.

Zcash can be thought of as an extension of the bitcoin protocol. Basically Zcash added some fields to the bitcoin transaction format to support encrypted transactions. Zcash uses SNARKs (Zero Knowledge Proofs) to encrypt all of the data and only gives decryption keys to authorized parties to see that data.

+ This could not be done on a public blockchain until now. because if you encrypted everything in the past it would prevent miners from checking to see if transactions are valid.
+ ZKPs have made this possible by allowing the creator of a transaction to make a proof that the transaction is true without revealing the sender's address, the receiver's address and the transaction amount.

Zooko describes this as follows:

+ Bitcoin has 3 columns, which are the three mentioned above (sender address, receiver address, transaction amount).
+ Zcash has 4 columns – and the 4th column proof doesn't know the sender address, the receiver address or the amount transferred. However, it does know that nobody could have created the proof that comes with the encrypted values unless they have a secret key with sufficient value to cover the amount being transacted.
+ This is a proof that the data inside the encryption correctly satisfies the validity constructs – it allows the prevention of double spends and transactions of less than zero.



Zcash is mostly the same as bitcoin:

+ The miners and full nodes are transaction validators.
+ Zcash uses Proof of Work (POW) that has miners checking Zero Knowledge Proofs attached to each transaction and getting a reward for validating those transactions.
+ Full nodes are the same, except that if you have the private keys you can detect if some transactions have money that is there for you. SNARKs enable miners to reject a transaction from someone if their private key doesn't have enough money for that transaction.
+ By keeping all data private except for the 4th column, it omits information from leaking onto a private blockchain - which would allow everyone to view the transaction information.
+ zCash has selective transparency, while bitcoin has mandatory transparency. This means that Zcash can reveal specific things to specific people by permissioning. It reveals specific transactions that anyone looking at them can verify in the blockchain.

Some differences that are highlighted in the zCash whitepaper include:

"Value in Zcash is carried by notes, which specify an amount and a paying key. The paying key is part of a payment address, which is a destination to which notes can be sent. As in Bitcoin, this is associated with a private key that can be used to spend notes sent to the address; in Zcash this is called a spending key.

A payment address includes two public keys: a paying key matching that of notes sent to the address, and a transmission key for a key-private asymmetric encryption scheme. "Key-private" means that ciphertexts do not reveal information about which key they were encrypted to, except to a holder of the corresponding private key, which in this context is called the viewing key. This facility is used to communicate encrypted output notes on the block chain to their intended recipient, who can use the viewing key to scan the block chain for notes addressed to them and then decrypt those notes.

The basis of the privacy properties of Zcash is that when a note is spent, the spender only proves that some commitment for it had been revealed, without revealing which one. This implies that a spent note cannot be linked to the transaction in which it was created."

Zcash is what's known as a decentralized anonymous payment schemes (DAP schemes):

- A DAP scheme enables users to directly pay each other privately: the corresponding transaction hides the payment's origin, destination, and transferred amount.
- In Zcash, transactions are less than 1 kB and take under 6 ms to verify — orders of magnitude more efficient than the less-anonymous Zerocoin and competitive with Bitcoin.
- However the privacy achieved is significantly greater than with Bitcoin. De-anonymizing bitcoin has become much easier through services that track and monitor bitcoin movements and the data associated with it. Mixer services allow for coins to be changed as they move through the system via a central party but this still is not sufficient enough.

The zCash whitepaper states:

> "mixes suffer from three limitations:
>
> (i) the delay to reclaim coins must be large to allow enough coins to be mixed in;
>
> (ii) the mix can trace coins; and
>
> (iii) the mix may steal coins.
>
> For users with "something to hide," these risks may be acceptable. But typical legitimate users (1) wish to keep their spending habits private from their peers, (2) are risk-averse and do not wish to expend continual effort in protecting their privacy, and (3) are often not sufficiently aware of their compromised privacy."

The major motivations for ZKPs and the Zcash protocol are 1) privacy and 2) fungibility.

- Fungibility is being able to substitute individual units of something like a commodity or money for an equal amount. This can be a real problem when some units of value are deemed less because they are considered "dirty". Hiding the metadata history doesn't allow for a coin with a bad history to be rejected by a merchant or exchange. In the words of Gregory Maxwell:

> "Insufficient privacy can also result in a loss of fungibility--where some coins are treated as more acceptable than others--which would further undermine Bitcoin's utility as money."

Zcash is expected to launch soon and, with that, the genesis block of the Zcash blockchain. Like the bitcoin blockchain, this will allow anyone in the world to mine for Zcash. It will be an open, permissionless system (fully decentralized). Users will be able to send it to anyone using zero-knowledge privacy.

ZCash's use of cutting edge cryptographic techniques comes with substantial risks. A cryptographic attack that permits the forging of zero knowledge proofs would allow an attacker to invisibly create unlimited currency and debase the value of Zcash. Attacks of this kind have been found and fixed in the recent past. Fortunately, the metadata hiding techniques used in Zcash tread are more production-hardened and can be considered less risky.

(e) Hawk

Andrew Miller in his whitepaper: "***Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts***" has developed a programmable smart contract system which works in much the same way as zCash for smart contracts"

- Hawk does not store financial transactions on the blockchain – the code of the contract, as well as data sent to the contract and money sent and received by the contract are all kept confidential.
- It is only the proof that can be seen – while all other useful information is hidden.
- Like zCash, transparency is selective in Hawk and wouldn't need to be used by all smart contracts - but rather, would be based on use cases and the preferences of the parties involved.
- It also aims to tackle the issues of privacy and fungibility in much the same way as the zCash protocol.

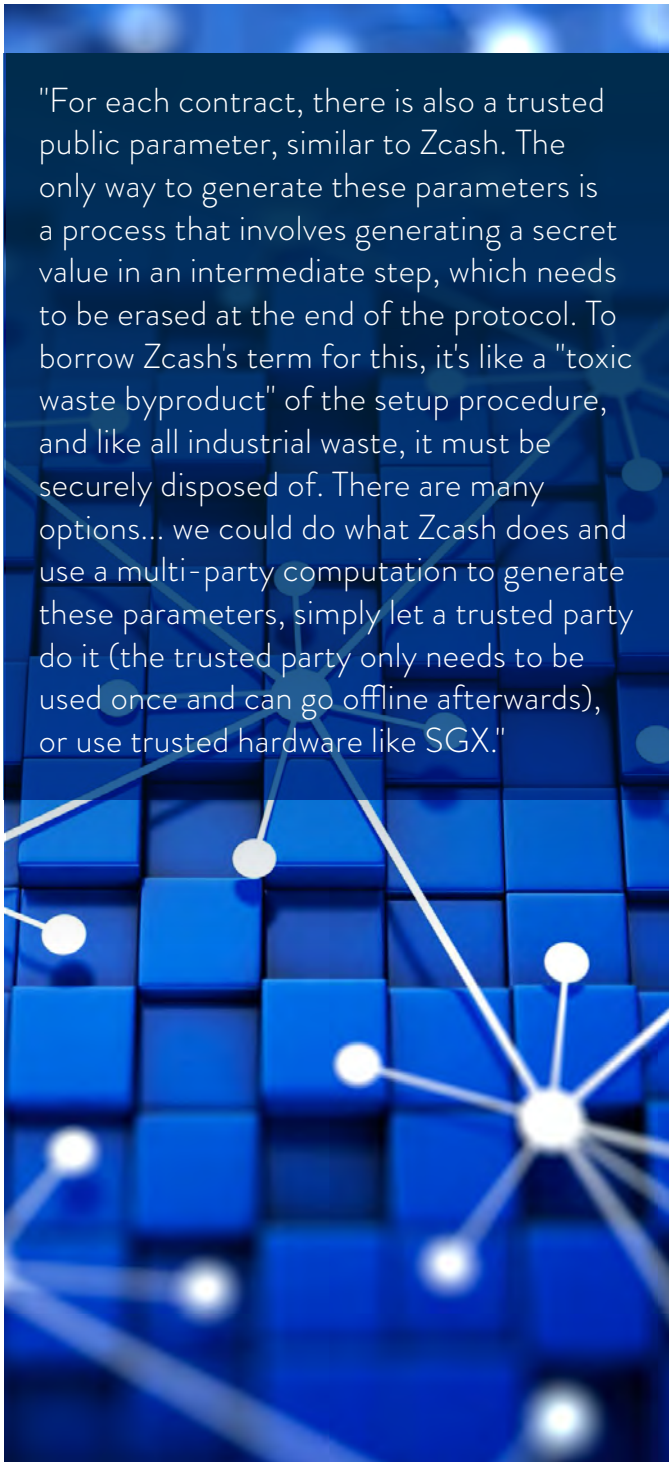
The Hawk whitepaper does a great job of describing the motivation for the contractual security it seeks to provide for financial transactions:

> "While on-chain privacy protects contractual parties' privacy against the public (i.e., parties not involved in the financial contract), contractual security protects parties in the same contractual agreement from each other. Hawk assumes that contractual parties act selfishly to maximize their own financial interest. In particular, they can arbitrarily deviate from the prescribed protocol or even abort prematurely. Therefore, contractual security is a multi-faceted notion that encompasses not only cryptographic notions of confidentiality and authenticity, but also financial fairness in the presence of cheating and aborting behaviour."

According to Andrew Miller, Hawk is based on several cryptographic primitives.

+ It uses the same zero knowledge proof library as zCash, which is called libsnark.
+ Hawk also uses custom implementations of a lattice-based hash function, and public key encryption. Hawk uses a jSnark tool which is open sourced.

In Hawk, each party generates their own secret keys. Miller stated that



Miller has said there are some differences between Ethereum contracts and Hawk contracts:

+ Unlike Ethereum, the input language for private contracts in Hawk is C code.
+ A private Hawk contract is not a long running stateful process like an Ethereum contract, but rather a 1-shot contract that proceeds in phases, where it first receives the inputs from each party, and then computes the outputs for each party.
+ After the outputs are computed, the contract is finished and no longer holds any balance.

So, it is a slightly different computing model. Hawk supports both private contracts (as described above) as well as public contracts. which are exactly like those in Ethereum. (No privacy guarantees are provided for the public contracts, though).

As in Zcash, there are some challenges to blockchain scaling and optimizing cryptographic schemes so that they are efficient when using ZKPs:

+ Hawk tries to do as much computation "off chain" as possible.
+ This is done because in public blockchains, "on chain" computing is replicated to every node and slows things down dramatically. Producing the proof can take up to several minutes (which is long) and can be costly.
+ By comparison, nodes checking the proof only take milliseconds to do that.
+ The Hawk whitepaper outlines that in Hawk, computation takes about a minute of CPU time for each participant in a Hawk contract. On chain computation takes about 9 to 20 milliseconds.

Hawk has not announced a release date yet as they are still working on optimizing their snark compiling tools to enhance performance.

(f) State Channels

State channels aim to address the scalability issues, privacy issues and confirmation delays associated with public blockchains, while still allowing actors who don't necessarily trust each other to transact.

+ State channels allow for payment channels that are "off chain" and allow for updates to any type of applications that have a change of state.
+ Like the Lightning Network, two or more users can exchange payments that would normally require a blockchain transaction, without needing to publish them on the blockchain or wait for confirmations except when setting up or closing out the channel.

Vitalik Buterin explains this in his paper for R3CEV "Ethereum Platform Review"

> *"State channels are a strategy that aims to solve the scalability challenge by keeping the underlying blockchain protocol the same, instead changing how the protocol is used: rather than using the blockchain as the primary processing layer for every kind of transaction, the blockchain is instead used purely as a settlement layer, processing only the final transaction of a series of interactions, and executing complex computations only in the event of a dispute.*
>
> *State channels are not a perfect solution; particularly, it is less clear how they extend to massively-multi-user applications, and they offer no scalability improvements over the original blockchain in terms of its ability to store a large state size - they only increase de-facto transaction throughput. However, they have a number of benefits, perhaps the most important of which is that on top of being a scalability solution they are also a privacy solution, as the blockchain does not see any of the intermediate payments or contracts except for the final settlement and any disputes, and a latency solution, as state channel updates between two parties are instant - much faster than any direct on-blockchain solution, private or public, possibly could be, and potentially even faster than centralized approaches as channel updates from A to B can be secure without going through a centralized server."*

4 DO YOU NEED A BLOCKCHAIN AT ALL? AND WHY IS CONSENSUS NEEDED?

For many people, all of these cryptographic methods (which mask all of the transactional data) will come as a surprise.

+ The blockchain is supposed to be a transparency machine in which anyone can join the network and, as a result, view all information on that network.
+ Even in private blockchains, there is a more open view into the data than the protocols that have been mentioned in this article.

> Is consensus even needed,
> if everything is private but for the proof???

+ If the proof is only between the two parties involved in the transaction, why is consensus needed? And why use a public blockchain?
+ It may seem counterintuitive, but the answer is ***yes***: a public blockchain ***is*** needed, and so is consensus, and this is due to the privacy of the proofs. Essentially, complete transparency is needed to maintain the privacy of the proofs.

Zero Knowledge Proofs and blockchains complement each other. You can't just use one to replace the other.

+ A blockchain is used to enable the entire network to agree on some state which may or may not be encrypted.
+ Zero Knowledge Proofs allow you to be confident about some properties of that state.

In this scenario, you still need a canonical source of truth: a view key that reveals all incoming transactions, but not outgoing ones. And for this to happen, you need a fully decentralized ledger with consensus - where everyone agrees with the data written there.

+ For example, zcash has data which contains information which is useless and unreadable to most actors. It's a database of commitments and opaque pieces of data. It's just a way to synchronize data between actors. (Zooko Wilcox has publicly stated that if Chainalysis graphed this out it would just be a series of timestamps of when a transaction occurred.) In cases where the number of transactions are low, then timing attacks could reveal the originator of transactions, imagine this to be equivalent of just one node connected to a Tor network.

The real emphasis is on the wallet side for actors, because this allows them to spend money and move assets around.

+ In bitcoin you can take a private key and move bitcoin.
+ Now it's more. It's a private key and a set of secrets you keep to prove previous proof and generate a new proof that you use to convince others. For this, a fully decentralized ledger is needed with consensus, where everyone agrees with the data written there.

A blockchain is necessary because you need consensus layer from everyone:

+ It is necessary to have an agreement of proofs in the ledger to move assets around later on.
+ If that proof isn't available in every node, then you can't convince anyone of the proof when you need to move assets later on.

These proofs need to be stored in an open way, so that the proofs can be seen as being verified and accepted by receiving parties.

There are two different layers here:

> (i) needs to be agreement on what proofs everyone accepts

> (ii) needs to be agreement on what you can prove; what happens on proof of zero knowledge; and what happens once you know the information.

How do you generate proof and pass that information to the next person?

+ The key is to get authority of the transaction by adding a proof or metadata to the transaction with some type of conditional script ("if then" statements for transaction acceptance). This code contains transaction validity rules. A person sees proof from outside but they don't know if the rule itself has been triggered or not.
+ Now that you have privacy from Zero Knowledge Proofs, then in order to comply with the transaction, you need to prove that the transaction abides by the rules. So you can take 2 proofs and create new proofs that the person receiving them can point at and verify that the proof is accepted by the entire network. Once the proofs have a meaning to you based on the rules, you can agree they were proved in the past and can be used in the future to transact and transfer money.

5 ZERO KNOWLEDGE PROOFS - THE CHALLENGES FOR ADOPTION

Zero Knowledge Proofs are only just commencing real world tests, and they still suffer from big scalability issues. The work of developing a proof is enormous and has massive computation costs:

+ Taking the example of Zcash, it takes between 45 seconds and 1 minute on a really strong computer to create a proof and transfer funds.
+ Presently, people are working on making SNARKs and Zero Knowledge Proofs more efficient by allowing for more proofs per second, or for more elaborate proofs in the same amount of time.

Deep architectual changes need to be made in blockchain technologies to leverage the benefits of Zero Knowledge Proof architecture. It requires an understanding as to the constraints of what we can prove, and at what scale.

Zero Knowledge Proofs are moving out of the realm of theory and becoming production strength. Now is the time to see how successful they are in delivering the demands for confidentiality and privacy in the private blockchain world.

Very Special Thanks to Zaki Manian (@zmanian), Andrew Miller (@socrates1024) Jonathan Rouach (@jonrouach), Anish Mohammed (@anishmohammed)

Hawk section provided by Andrew Miller from a series of questions I asked.

GILBERT +TOBIN