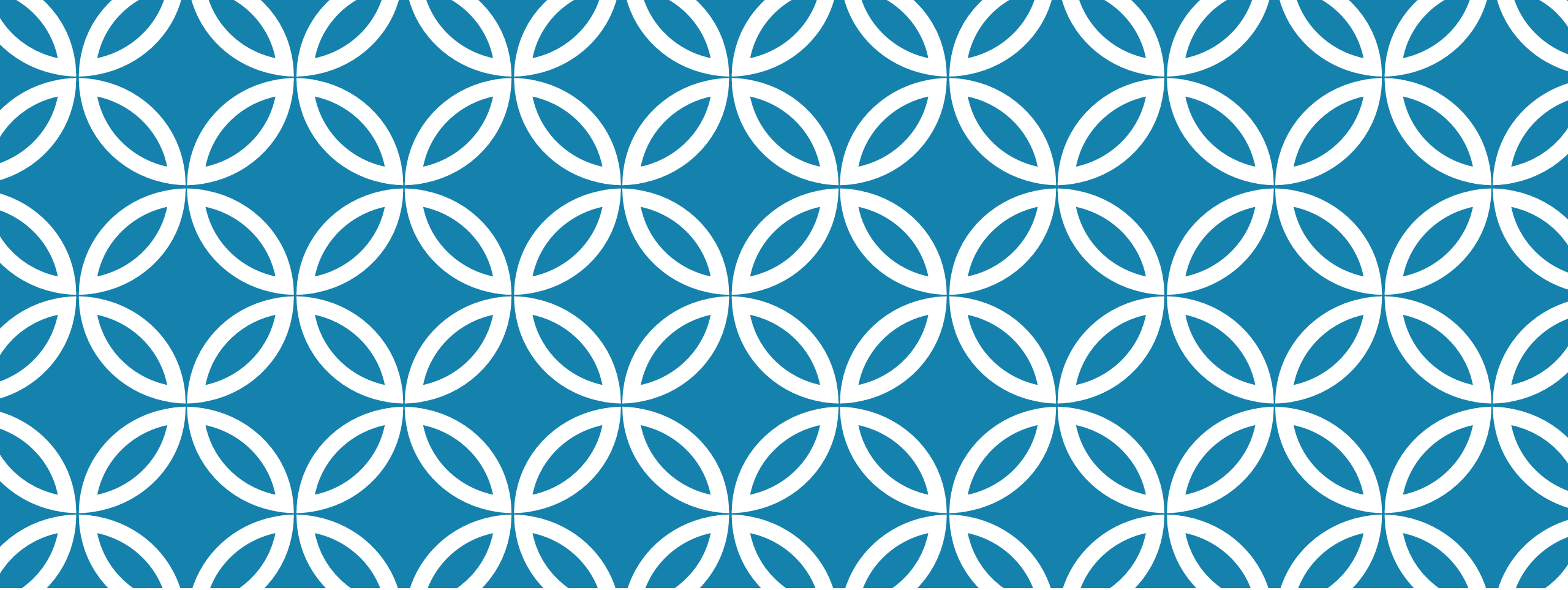


CS5412/LECTURE 14 BLOCKCHAINS FOR IOT (PART 1)

Ken Birman

CS5412 Spring 2020

BLOCKCHAINS FOR IOT

Lucas Mearian. Not afraid of hyperbole!

What is blockchain? The most disruptive tech in decades!

"The distributed ledger technology, better known as blockchain, has the potential to eliminate huge amounts of record-keeping, save money and disrupt IT in ways not seen since the internet arrived."

Lucas Mearian, ComputerWorld Staff Writer

A MORE TECHNICAL ANSWER?

"A blockchain, originally block chain, is a growing list of records, called blocks, which are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data (generally represented as a Merkle Tree root hash).

By design, a blockchain is resistant to modification of the data."

Wikipedia

TERMINOLOGY

In BlockChain settings, a *transaction* is a digital record describing some event.

Some transactions are complete and self-contained.

But BlockChain also supports transaction languages in which one transaction might refer to events defined by a past *or future* transaction.

CRYPTOGRAPHIC HASH

A cryptographic hash is a bit string computed from some block of data in a manner that yields a constant-length result irrespective of the data size, and yet such that it would be infeasible to find other data that would hash to the same result.

There are a number of hashing schemes. A highly robust one is SHA-256. SHA-512 is even stronger. MD5 and SHA-1 have been compromised and are unsafe.

EVEN FASTER HASH METHODS EXIST

(single core performance, all Golang implementations, see benchmark).

BenchmarkHighwayHash	11,986 MB/s
BenchmarkSHA256_AVX512	3,552 MB/s
BenchmarkBlake2b	972 MB/s
BenchmarkSHA1	950 MB/s (insecure)
BenchmarkMD5	684 MB/s (insecure)
BenchmarkSHA512	562 MB/s
BenchmarkSHA256	383 MB/s

Note: the AVX512 version of SHA256 uses the multi-buffer crypto library technique as developed by Intel, more details can be found in sha256-simd.

https://blog.minio.io/highwayhash-fast-hashing-at-over-10-gb-s-per-core-in-golang-fee938b5218a

HARDWARE CAN GET EVEN FURTHER

FPGA and ASIC solutions can be purchased that will run SHA-256 or SHA-512 at speeds of 25,000 to 30,000 MB/s

In some parts of the world there are entire datacenters equipped with huge numbers of these accelerator solutions. China dominates the business.

Very hard to be a BlockChain miner with a single desktop computer today!

CRYPTOGRAPHIC HASH KEY

All of these functions require a key in addition to the message.

If the key is public, anyone can recompute the same hash from the message.

If the key is private, then only the application holding the key can do so. The hash can then serve as a form of signature, verifiable by other components of the same application, since they would also know the key.

PUBLIC/PRIVATE KEY PAIR (NORMALLY, RSA)

This is a classic cryptographic method.

RSA creates two "keys", both just long numbers together with a modulus *n* that itself is a product of two very long prime numbers. Call them K, $\overline{K}$

One is designated as the public key and shared. You keep the other private.

$$RSA_K(RSA_{\overline{K}}(X)) = RSA_{\overline{K}}(RSA_K(X)) = X$$

WHY RSA WORKS

In 1796, Gauss came up with the theory that ultimately gave us the (very simple) RSA technology. Gauss himself didn't suggest this application.

In RSA encryption and decryption are just mathematical steps that involve a form of "bignum" arithmetic (modular exponentiation), performed block by block.

RSA is secret because there is no known method for factoring a giant composite number that might have 1000's of binary digits. If we could factor the modulus, it would be trivial to recover the secret key from the public one.

Quantum computers *might* offer a path to doing so, but it would require devices with millions of qbits, way beyond anything feasible anytime soon.

RSA STRENGTHS, WEAKNESSES

Very widely supported, basis of most "certificates" used in the Internet.

Many tricks exist, based on commutativity of RSA computation. Basically, for *any* two RSA keys, A and B, $RSA_A(RSA_B(M)) = RSA_B(RSA_A(M))$.

But RSA is fairly slow. The speed is a function of the data size. We don't casually encrypt entire messages with RSA: it would be feasible but slow.

HOW WOULD PROCESS P SIGN MESSAGE M?

1. Compute the SHA-256 hash of M using a public SHA key.
2. Now use P's private key to encrypt the hash: $SHA(M)_{private\text{-}key\text{-}of\text{-}P}$

Process Q can easily verify that M has not been tampered with:

1. Q recomputes the SHA-256 hash for M, using the same public SHA key
2. Now Q uses RSA with P's public key to crypt P's signature.
3. If they match, then Q has confirmed that M hasn't changed.

NOTARIZING BLINDED DATA

There is even a method, by David Chaum, for signing an object that the signatory cannot see. It would be useful for secure voting:

- Prepare your ballot, then blind it and obtain a signature.
- The signature is proof that your vote was valid and only cast once. Submit it for counting now, unblinded, via a secure anonymous "onion route"
- The ballot itself has no identifying information, and neither does the signature. So a third party can see that your vote is valid, and can count it, and yet can't learn how any particular individual voted.
- Chaum also showed how to get a receipt which can be used to be sure your vote was properly tabulated.

PARALLELISM?

A further win is to maximize parallelism and reduce record sizes.

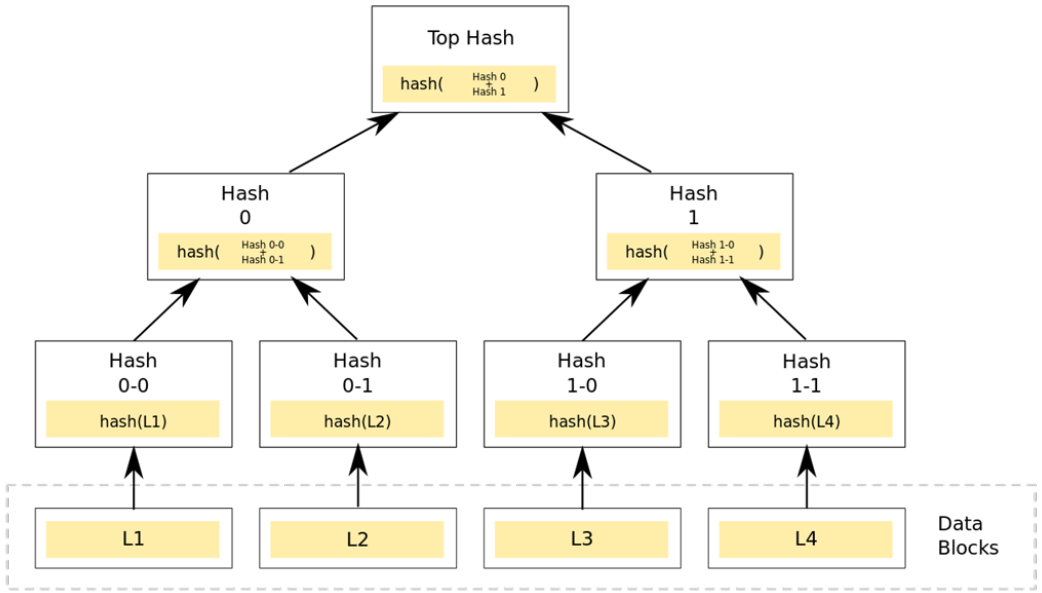
In the case of BlockChain, each block contains a set of transactions represented as binary records. These records might be huge, hence slow to hash (and very slow to encrypt, were you to try that).

By having the creator store the record someplace reasonable and then just storing signatures in the BlockChain, we use it as efficiently as possible.

MERKLE TREE: A TREE OF SIGNED RECORDS.

Rather than making one list of N records and then hashing them, we often create a binary tree of hashes.

Very common in BlockChains, permits us to run SHA-256 or SHA-512 in its fastest "mode" of operation.



Often we replace the entire "log" with a tree and our Block "chain" becomes a sequence of Merkle trees that change (only) by adding nodes (== append)

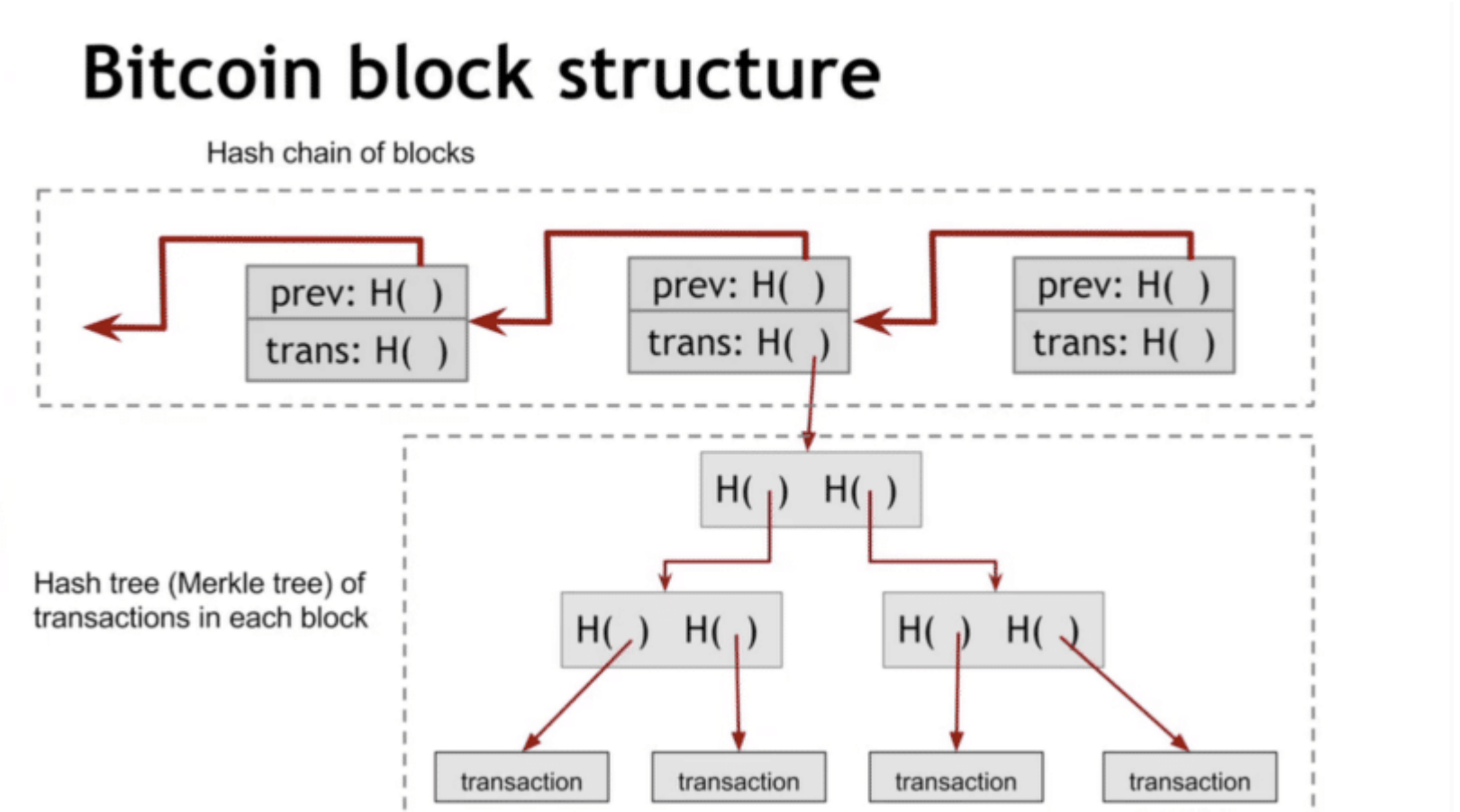
THIS ALREADY GIVES US A BASIC SOLUTION!

Compute a series of records, each containing transactions signed by the initiator. The record needs to include the "name" of the initiator so that anyone needing to do so can look up the matching public key.

Associate each record with a key for lookup, and insert the (key,record) objects into the Merkle tree.

Then create some form of cryptographic proof that the new tree extends the prior tree. Logs are just one possible representation.

OUR BASIC SOLUTION



https://coincentral.com/merkle-tree-hashing-blockchain/

WHY IS THIS SECURE?

If anyone tampers with any record in the chain, we can sense this by recomputing the Merkle tree. The signature won't match.

To verify the entire chain, block by block recompute the Merkle tree, then recompute the sequence of pairwise hash values.

Verification is required when an untrusted BlockChain is initially loaded.

PERMISSIONED/PERMISSIONLESS

BlockChain solutions split into two categories.

A **permissioned** BlockChain is managed by an authorized group of servers, for example inside some datacenter. We generally assume that attackers either compromise the *entire* data center, or can't attack the servers.

A **permissionless** BlockChain is managed by an anonymous group of servers that volunteer to play the role, and might come and go at will.

PERMISSIONLESS ATTACKS ARE AN ISSUE!

Permissionless BlockChain can come under many forms of attack.

- Compromised server might try to hand out "fake" versions of the chain.
- It might try to generate huge rates of transactions on its own, and not include your transactions. This is a form of DDoS attack.
- It could try to corrupt individual transactions or records.

PROOF OF WORK

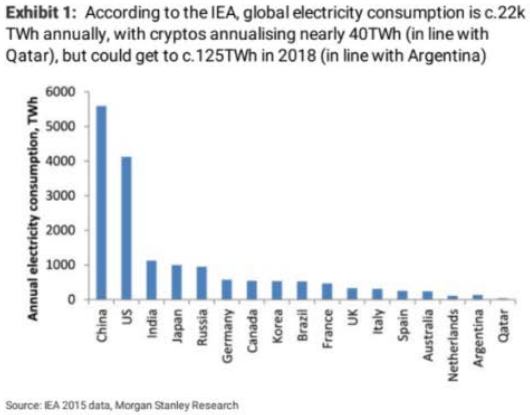
A Proof of Work mechanism adds one more field to the blocks: a "nonce".

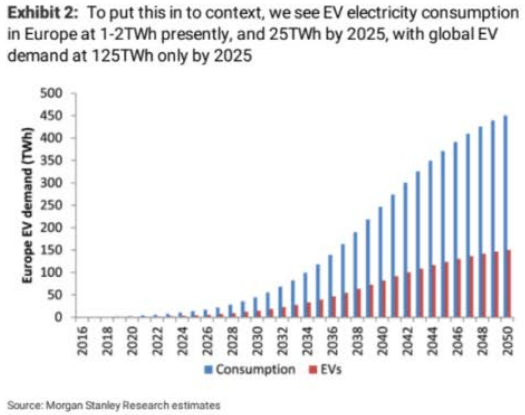
The nonce is just a bit string of some size.

The rule is that to append B_{k+1} to the chain, in addition to hashing it with the hash of the prior block, P must also find a nonce such that when the nonce is included and a new hash is computed, the hash value ends with some desired number of 0 bits.

PROOF OF WORK

Finding such a nonce is hard work!





So while P could be keen to append its block, it may need to search for this nonce for many seconds or minutes.

The difficulty of finding a nonce value that will work prevents DDoS attacks.

HOW MOST BLOCKCHAINS WORK TODAY

You can download the software as a VM or even compile it yourself.

You launch the code on your own servers– this makes you a "miner" as soon as the system has initialized itself.

The system downloads the entire current BlockChain, from other machines already running the BlockChain software (there is a web site listing some you can contact for copies).

ASIDE: WHY THE WHOLE TREE?

There is a lot of research (meaning, a huge amount) on ways to download and verify just a portion of the blockchain.

In some settings this is going to be absolutely obligatory, but right now it isn't how the big public blockchains work.

The most promising approaches provide "countersignatures" for the portion you want, provided by authorities you happen to trust.

NEXT, YOU VERIFY THE BLOCKCHAIN

You'll need to recompute all the Merkle trees and the chain of hashes.

In fact this may not take enormously long… today. Few BlockChains have huge amounts of content.

But someday, we might have BlockChains with hundreds of billions of records and total sizes in the petabytes. Then download speed and verification time and storage will become an issue!

MEANWHILE, NEW BLOCKS ARRIVE

Each block extends some specific sequence of prior blocks.

If you turn out to have downloaded the wrong sequence, you may have to truncate your chain and download the longer sequence.

This is a “rollback”. During startup, substantial rollbacks can occur. Later they shouldn’t (assumes a fully connected network of mostly “correct” miners).

AT THIS POINT YOU CAN CREATE TRANSACTIONS

So, you open for business.

Someone shows up to buy a glass of your fresh lemonade.

You'll generate the transaction (think "credit card payment slip") and submit it to the system. It enters a pool of pending transactions.

WHEN WILL YOUR TRANSACTION GO THROUGH?

Within an hour or so, you should see that your transaction got included into some block, and also that everyone seems to have adopted that block.

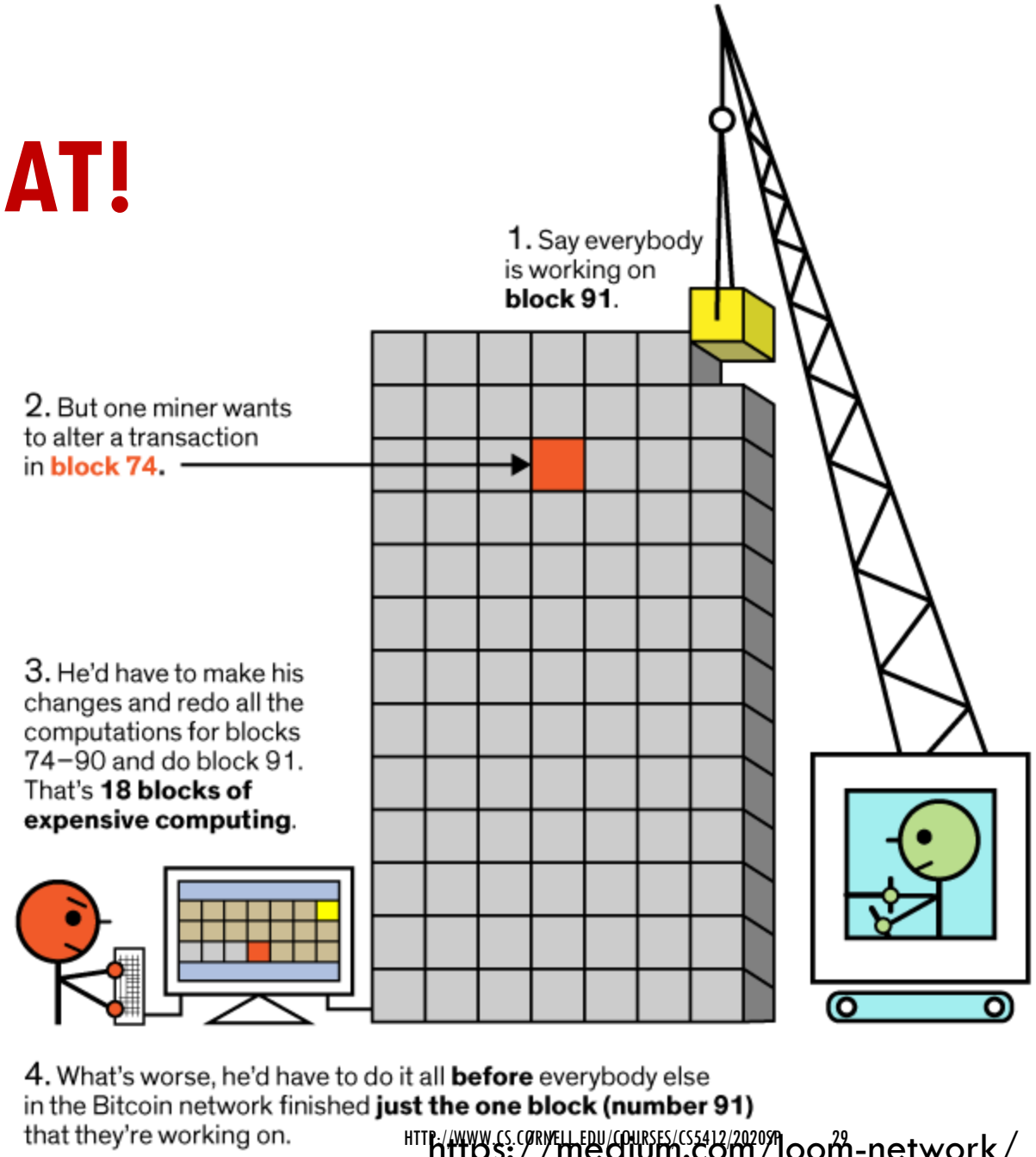
The chain has moved six or more blocks into the future.

So now you can hand that glass of frosty bliss to your happy customer!

BUT NOBODY CAN CHEAT!

To modify a past record you need to also modify every signature subsequent to that record.

The step where you have to find these nonce values will be very slow and you'll lose the race.



https://medium.com/loom-network/

WHAT IF THE ATTACKER IS A COUNTRY?

A country could build whole datacenters, equipped with hardware to compute SHA-256 at ultra-high speeds.

In this case P (using the datacenter) could generate a lot of blocks quickly, for which they would be paid. Or could have an entire second BlockChain starting from months ago, and *longer than the official main one*.

To prevent most such attacks, BlockChain solutions make the proof-of-work task harder as a function of the rate at which blocks are being found.

WHAT IF THE ATTACKER IS A COUNTRY?

In effect, if P controls enough computing power, he can "gain control" of the BlockChain. The proof-of-work can become so hard that only P has the compute power to solve the puzzle!

P could then refuse to post some transactions, or cause trouble in other ways.

But this form of attack has not (yet) been seen.

WHAT ABOUT RACES?

Permissionless BlockChains are at risk of a "race" situation in which one group of miners is working to append record R, and some other group, record S. A tie can easily occur.

BlockChain systems "adopt the longest chain" (may the best miners win). This can cause a rollback if a few blocks were appended by group A, but then group B suddenly publishes a longer extension.

In practice, rollbacks longer than 6 blocks are never observed.

IN CONTRAST, PERMISSIONED BLOCKCHAIN DOESN'T NEED PROOF OF WORK

A permissioned system is operated by known, trusted, authorized servers.

They won't attack the chain by trying to overload it with transactions in an unfair way, and they would charge for any transactions they append on behalf of external clients.

So we can avoid this costly step with datacenter BlockChain solutions.

WHAT IF YOU DON'T REALLY TRUST THE PERMISSIONED PROVIDER?

We can mix methods: a global "proof" with a local "data store"

Our permissioned provider can commit to some form of cryptographic root of each new version of the log (or tree), and to a proof that the new version extends the old version.

The "commit" is broadly shared and pins the provide down. Then for an append or a query, the provider can be asked to also provide a proof that they did the append, or that the query response is correct & complete

WHAT'S IN A TRANSACTION?

Some BlockChain systems are very rigid. For example, a BitCoin BlockChain record can only support a few operations on BitCoins.

These represent transactions: Ken sells Ittay a packet of gum for 10฿

In a permissionless scheme, Ken would probably wait for a while before handing the gum to Ittay. With permissions, rollback risk can be reduced or even completely eliminated.

FANCIER TRANSACTIONS

There are several standards for encoding fancier “digital contracts” into records suitable for BlockChain.

One, called HyperLedger, uses HTML as its underlying “language”.

A second, Ethereum, has a sophisticated language of its own, and can even encode computational tasks into the transaction record.

COULD WE USE BLOCKCHAIN FOR IOT?

This is a topic generating huge interest!

For example, in a smart farm, a BlockChain could be used as a tamperproof audit trail, proving that animals had proper vaccinations and vet checkups, tracing food they ate and other life events, and later tracing the entire food supply chain from farm to table.

Cornell's Vegvisir BlockChain focuses on this case. Intermittent connectivity is a strength of Vegvisir: it can handle periods of disconnection.

BUT THERE ARE ALSO MANY ISSUES

From the chain, how can an auditor be sure that the transactions reflect the actual farm with its actual animals and sensors, and not a "simulation" of a farm with fake information?

What should be the requirements for this form of monitoring and auditing, and how costly will it be to perform?

What if we don't trust the software? What about privacy?

MORE ISSUES

Farming is big business, and operates with loans, futures contracts, conditional agreements that can play out in many ways, etc.

Would a single chain somehow need to encode all the farm-related digital contract events in the whole world? Even with one chain, how will its resources be managed? What would we do if a portion is irretrievably lost?

If not, if we have multiple chains, how would they be integrated?

MORE ISSUES

With permissionless BlockChain, is it really "safe" to trust that after six blocks have been appended, the chain won't roll back and invalidate my transaction? ("When should Ken give the lemonade to Sally?")

If a smart contract references future events, what would be the "semantics" of that contract, in a PL sense? Does the meaning depend on waiting for the future to occur? Can chains of dependencies arise, or contracts that are undecidable, or infeasibly complex to "evaluate"?

ASIDE (TIME PERMITTING)

Will Quantum Computing really break cryptography?

Which is closer to the truth:

- A quantum computer can make non-deterministic guesses, check to see if any are right (like guessing the factors of an RSA key), and then output the correct one.
- A quantum computer can compute a near-infinite number of discrete fourier transforms "concurrently", but you can only read out one data-point of the result at a time.

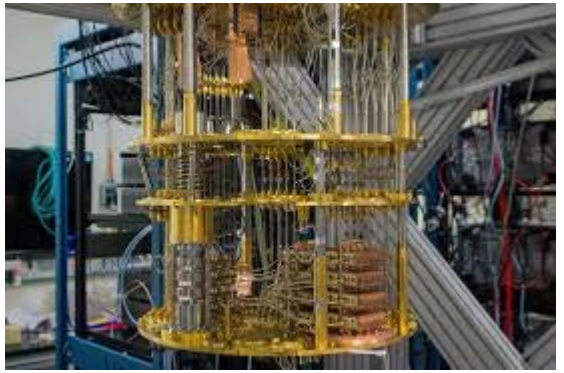
PUBLIC MISUNDERSTANDING

Popularity of the "many worlds" interpretation of physics has clouded the public conception of what a quantum computer can do!

In fact many worlds could be a valid model, for the most elementary level of Planck-scale physics (the layer where people talk about mbranes and string theory, and loop-quantum gravity).

But our macroscopic ("causally emergent") world is very remote from that most basic layer of physical reality.

SHOR'S ALGORITHM



To factor RSA, Shor's algorithm requires a special circuit specific to the size of the keys.

Then we input "all possible" n-bit integers, where n is the key length, like 1024. This involves a "coherent entanglement" of n qbits. But due to errors, qbits rapidly decohere. Error correction will require vastly more qbits, and nobody is sure how many. Perhaps millions or billions.]

The entangled data is then transformed by the circuit, which computes a DFFT

READING THE OUTPUT

You read the output of a quantum computer by setting up the experiment again and again and then repeatedly extracting a single sample.

Over time, the values you read build up to a kind of probability density image, like a photo created pixel by pixel.

In the case of Shor's algorithm this photo shows peaks that hint at the values of the factors. Now you can search for the factors close to those peaks. Quality of the search will depend on the sharpness of the peaks.

A LOT OF ASSUMPTIONS!

Nobody knows how quantum error correction "scales". Today it works for 3 to 5 q-bit entanglements, at best.

Nobody knows how complex a computation we can perform without destroying coherence. In fact these quantum DFFT operations must be reversible in order to remain coherent, and hence perfectly precise.

Nobody knows how quickly we can set up such a run and sample it.

Nobody knows how sharp the peaks will need to be as a function of key length.

AND WORST OF ALL…

Unfortunately, neither Euler nor Ramanujan really looked closely at this question!

Nobody knows if factoring large numbers is even a "hard" problem!

True, we lack a fast solution today. But the complexity of factoring is unknown.

But perhaps some numerical savant will find a solution… with classical computers! The same goes for finding a nonce with the desired hashing properties to mine blocks…

THE ENTIRE EDIFICE COULD COLLAPSE!

If you bet heavily on BlockChain, you are betting that people will figure out a way to ensure that it won't yield to some kind of attack.

But in fact this is just a bet, today.

PROVABLY SECURE SYSTEMS

There is a theory of **semantic cryptography** safe against quantum attacks. It was developed by Goldwasser and Micali, who won the Turing Award for the insight.

They proved that secure encryption schemes must be probabilistic, rather than deterministic, with many possible encrypted texts corresponding to each message.

The **Goldwasser–Micali** (GM) lattice cryptosystem demonstrates the idea.

COULD A BLOCKCHAIN USE LATTICE CRYPTOGRAPHIC TECHNIQUES?

At present, lattice cryptography is too computationally slow for practical use, and also causes too much "inflation" in the size of data.

Each bit in the data becomes a point in a very high dimensional space, leading to a billions-to-one increase in message sizes.

But continued research may yield much more compact solutions with the same properties. A new research initiative just started on this topic.