Blockchain for Large Scale Scientific Computing

Lindsey Gray, Lothar Bauerdick, Bo Jayatilaka, Gabe Perdue (FNAL),
Josh Bendavid (CERN)

12 July, 2018

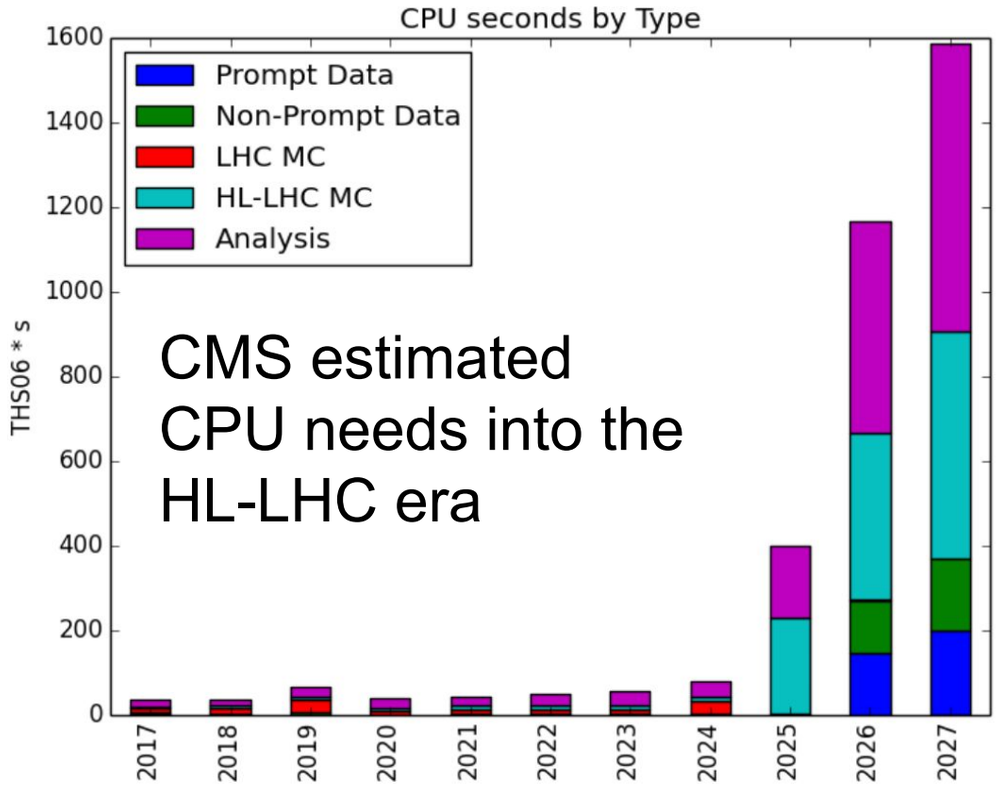
The Data and Computational Challenge of the HL-LHC

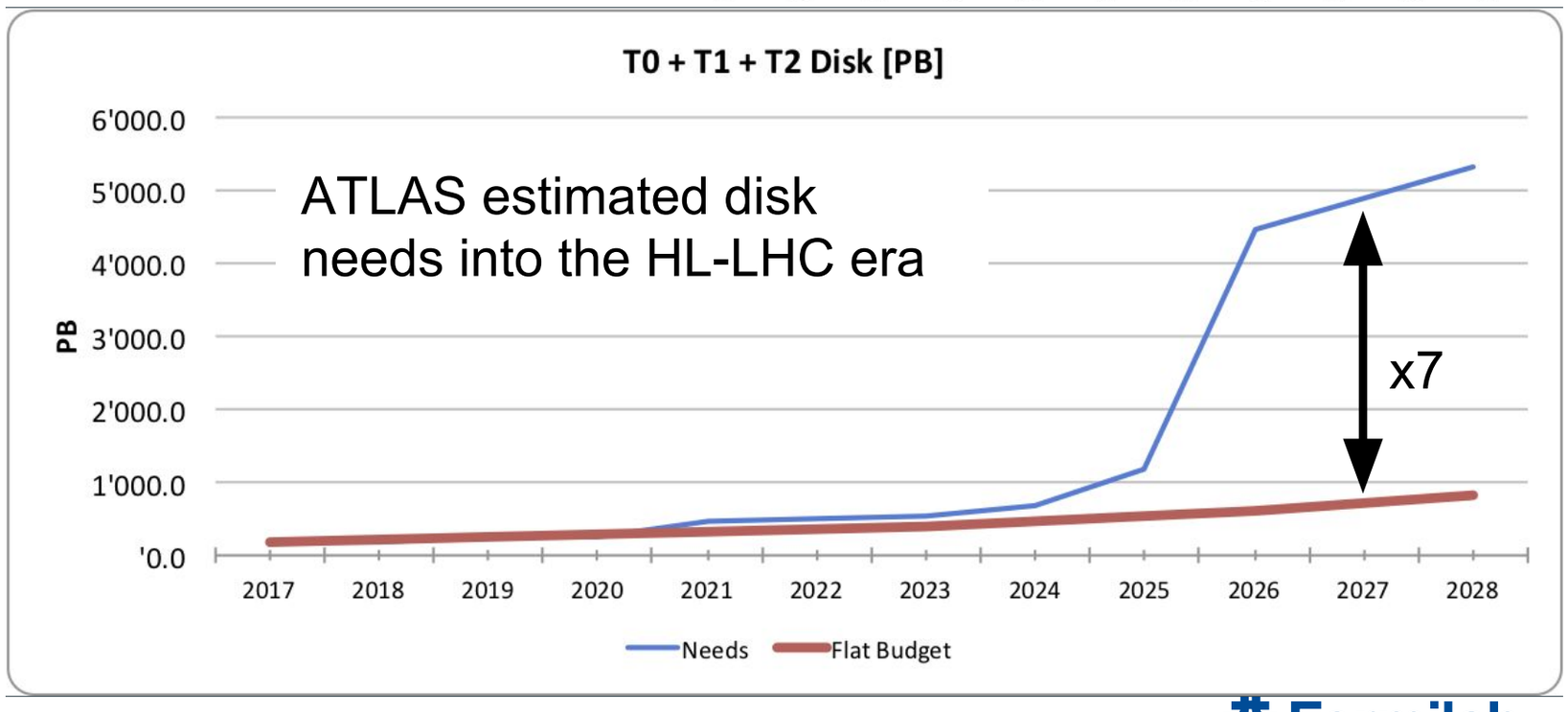
- The simultaneous increase of computational complexity and data rate at the HL-LHC presents an enormous challenge
  - Advancements in traditional storage and computing technologies cannot cope with this increase

- Solving this challenge will require novel solutions in many domains

- The global amount storage and computing resources is enormous

- Can we access it in a trustworthy, incentivized manner?





Fermilab

Current Crowdsourced Scientific Computing Models

Folding@home recent compute statistics

OS Type	Native TFLOPS*	x86 TFLOPS*	Active CPUs	Active Cores	Total CPUs
Windows	857	857	67,467	187,104	5,857,235
Mac OS X	91	91	8,083	85,382	217,033
Linux	87	87	6,383	26,457	882,200
NVIDIA GPU	1	2	4	4	348,371
ATI GPU	10,243	21,613	7,178	7,178	426,335
NVIDAI Fermi GPU	36,065	76,097	21,570	21,587	624,822
Total	47,344	98,747	110,685	327,712	8,355,996

- Developed as a platform for original SETI@Home, to combat security flaws
  - People will try to cheat any game they can win (have to incentivize playing fair)
  - There were completely falsified computations performed on SETI@Home nodes
  - Scientific credibility is our foremost priority, finding ways to maintain it is paramount

- Highly successful projects based on BOINC middleware - but no storage!
  - Triplicate verification - expensive error checking, heavily disincentivized cheating
  - Folding@Home, LHC@Home, Einstein@Home, ATLAS@Home, ~26 petaflops this year

See presentations of: D. Cameron

Fermilab

Questions and Contents in This Talk

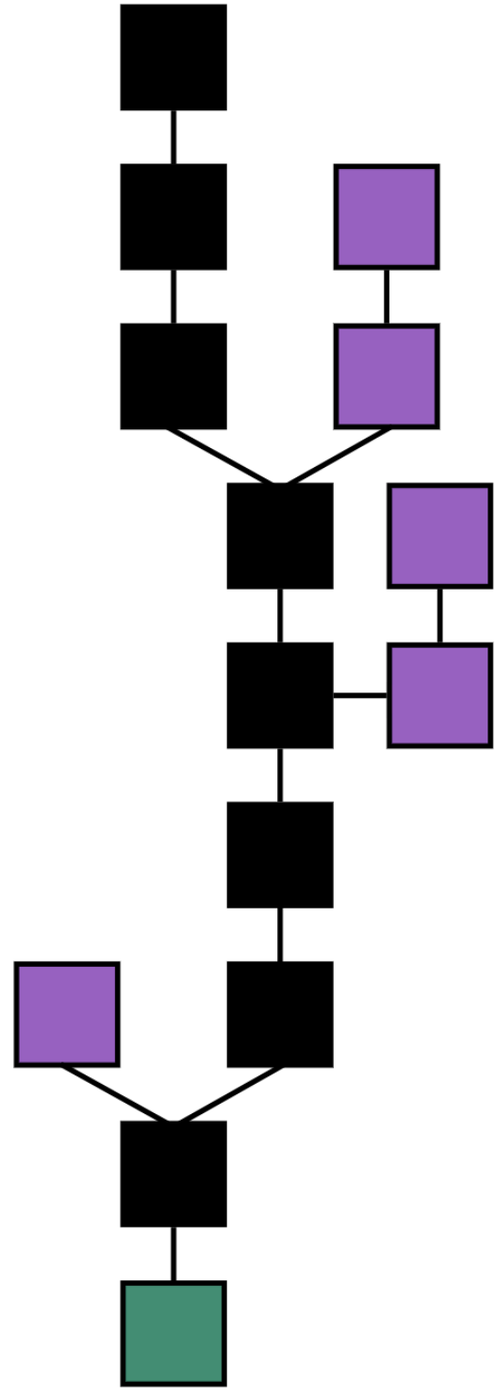
- Description of blockchain technology and its present ecosystems
  - What sort of computing power do decentralized networks presently wield?
  - Does this technology actually fit what we need to do?
  - Can this cover storage as well?
  - Does it scale well?

- Possible applications of blockchain and cryptographic technology to create further incentives for doing scientific work
  - Can we disincentivize cheating more efficiently with better algorithms?
  - What sort of scientific computing tasks can be covered?

- Statistical methods for verifying correct execution of algorithms
  - Worked out example for a Kalman Filter

Blockchain - A Distributed Ledger Technology

- A blockchain is a linked list where each node is connected to its predecessor by a cryptographic hash
  - All pointing back to the “genesis” block (right, in green) which may contain defining information about the rules for the blockchain protocol
  - In this way a blockchain comprises a verifiable public ledger



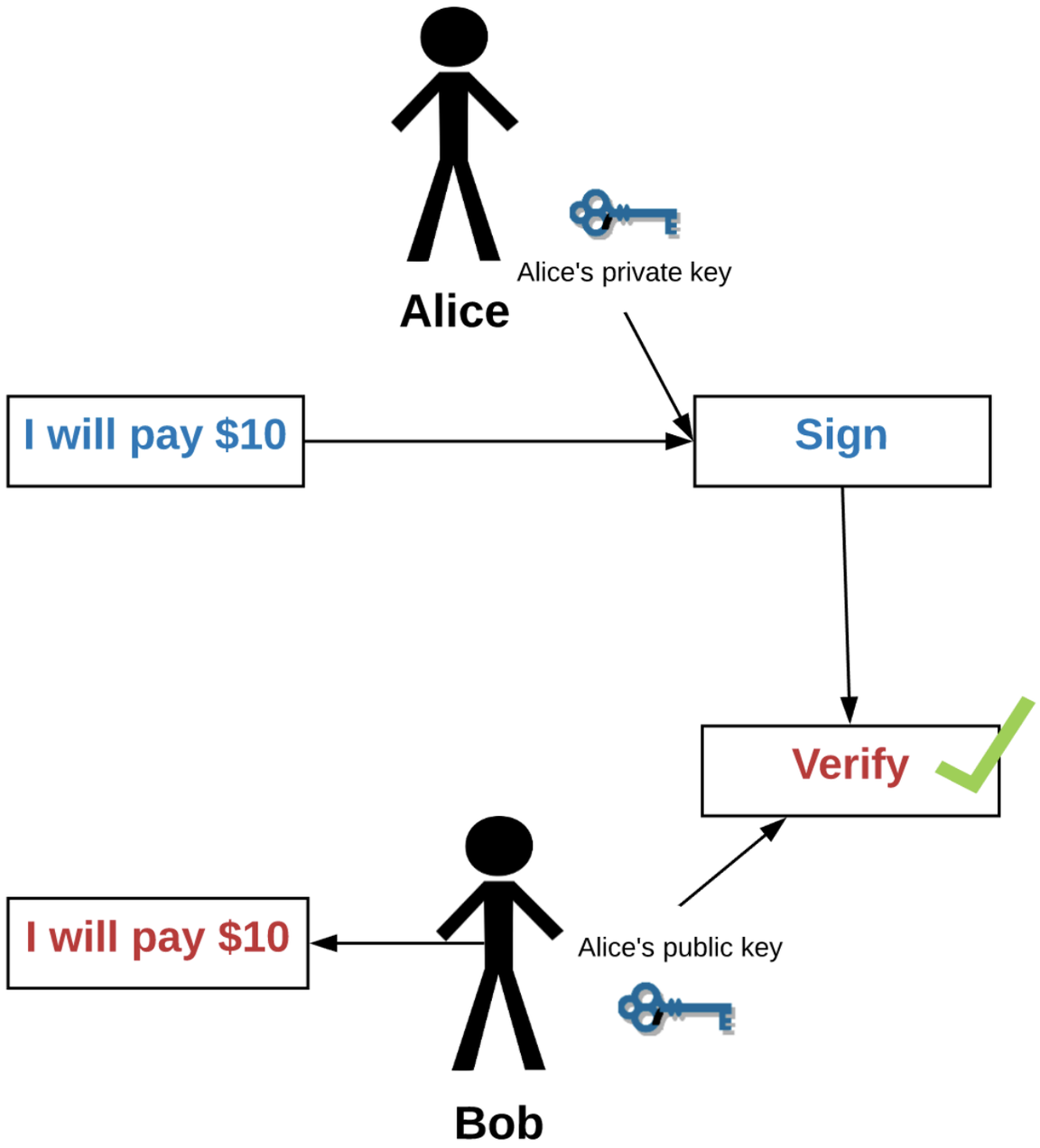
- Each node of the linked may contain additional transaction data that is itself hashed and verifiable

- Typically it’s the longest contiguous chain (right, in black) which is considered valid (purple are orphaned blocks)
  - However it’s up to the developers who define the protocol to determine the rules for consensus and evolution of the chain

- A variety of blockchains exist today, some exploring alternative architectures to test multiple aspects of scalability

What problem does this solve?

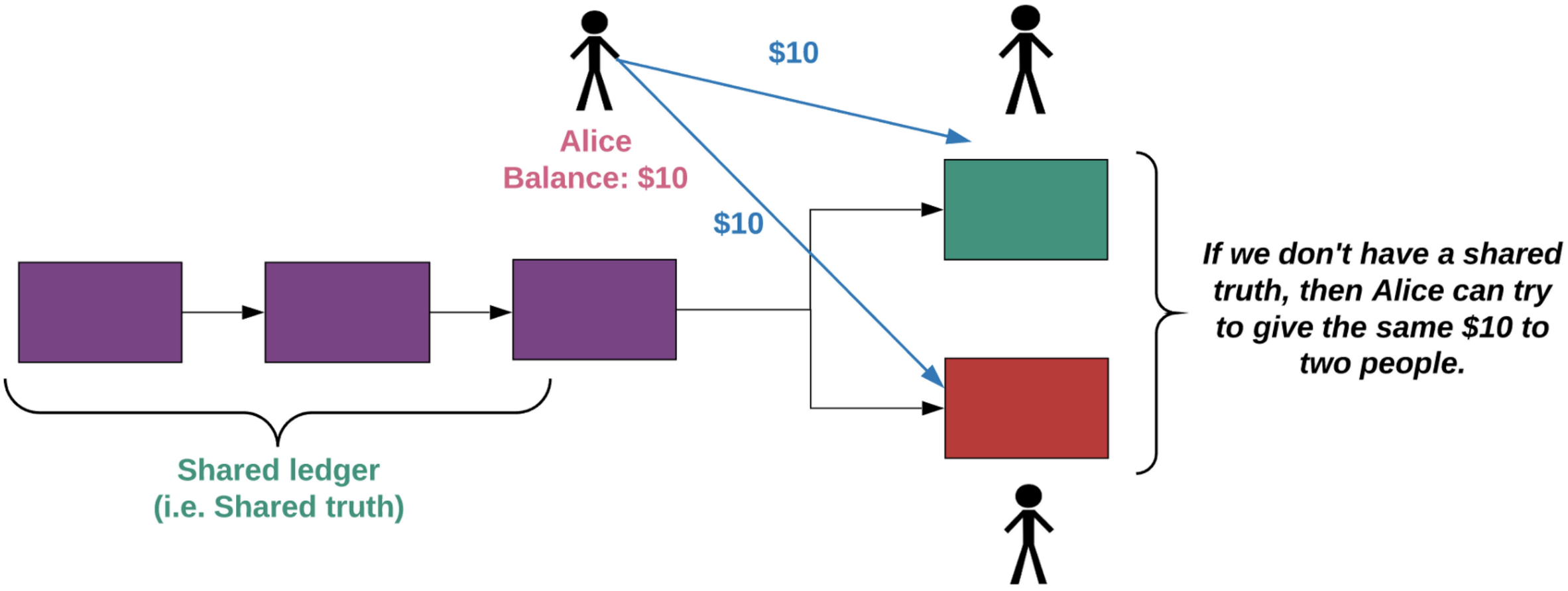
Alice wants to transfer $10 to Bob using only a digital signature.



If Alice is a bad actor then they could try to pay two people the same $10.

Fermilab

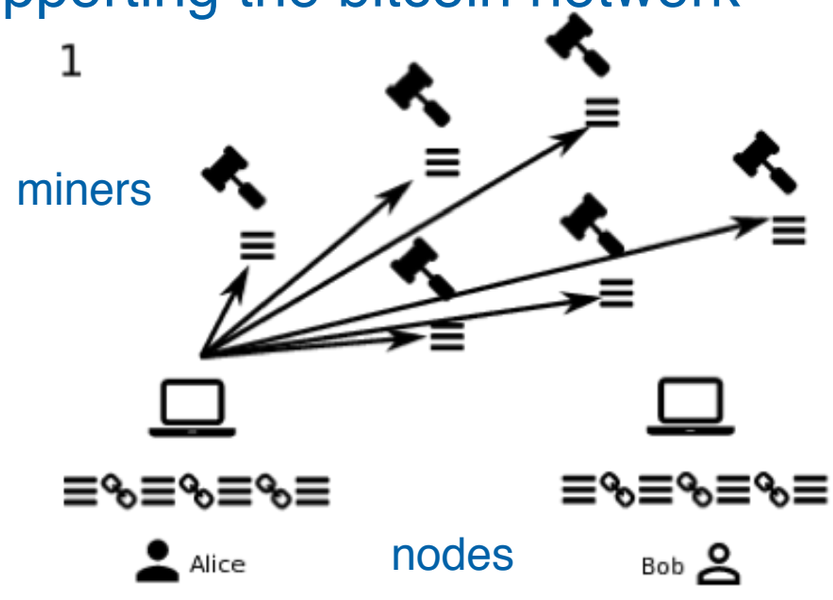
What problem does this solve?

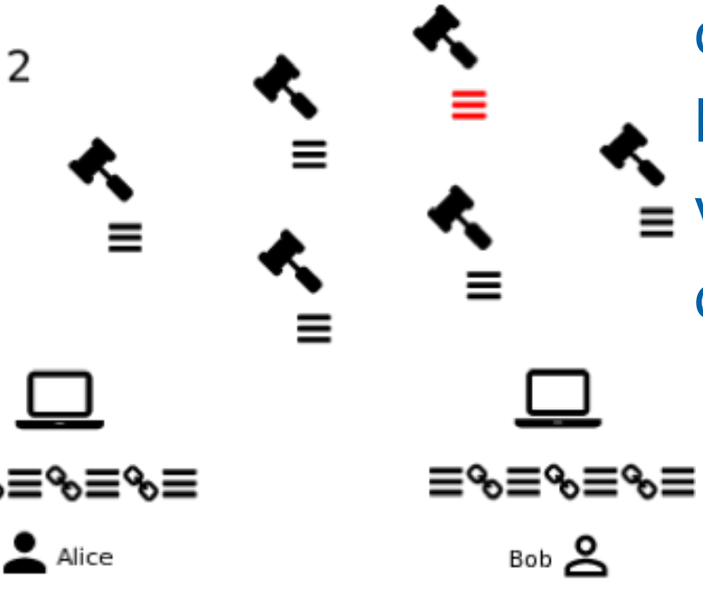


However, with a ‘truth’ that is shared and synced between all users you can keep track of Alice’s balance and ensure that they may not spend twice.

Consensus Example in BitCoin - Leveraging Processing Power

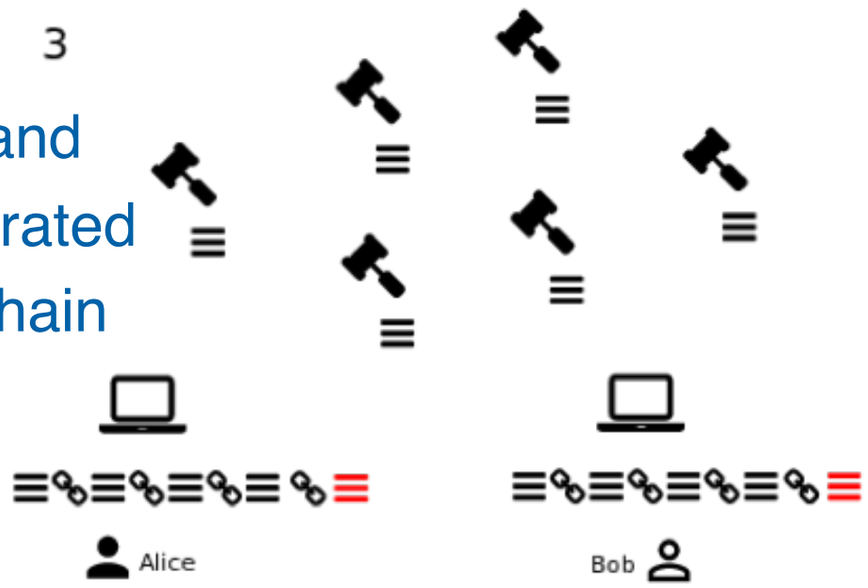
1) Transactions are sent to miners supporting the bitcoin network





2) Miners perform Proof-of-Work (producing a hash below a certain value), the first miner to do so “wins”

3) The winning PoW and transactions are integrated into the distributor’s chain



4) This new block’s PoW is then verified by **all** other node maintainers in the BitCoin network, eventually achieving global consensus on the state of the chain

https://eprint.iacr.org/2015/261.pdf

NB: This is just one possible implementation of a network that achieved consensus

Fermilab

Impact of Good Incentives: The Computing Power of Bitcoin

- Bitcoin started with a very clearly stated purpose to create and sustain a digital currency not controlled by any government except itself
  - Decisions for algorithmic changes can be established in the chain, voting protocols, etc.
  - A high-minded goal is a first incentive towards people to work together

- This is countered by the possibility to make a considerable profit from an attack
  - Most forgery-style attacks are countered by the global consensus mechanism, invalid blocks are expunged from the system
  - Attacks by network takeover are prevented by the considerable computing power needed to create an alternative chain that is growing faster than the current bitcoin blockchain

(miners) (nodes)

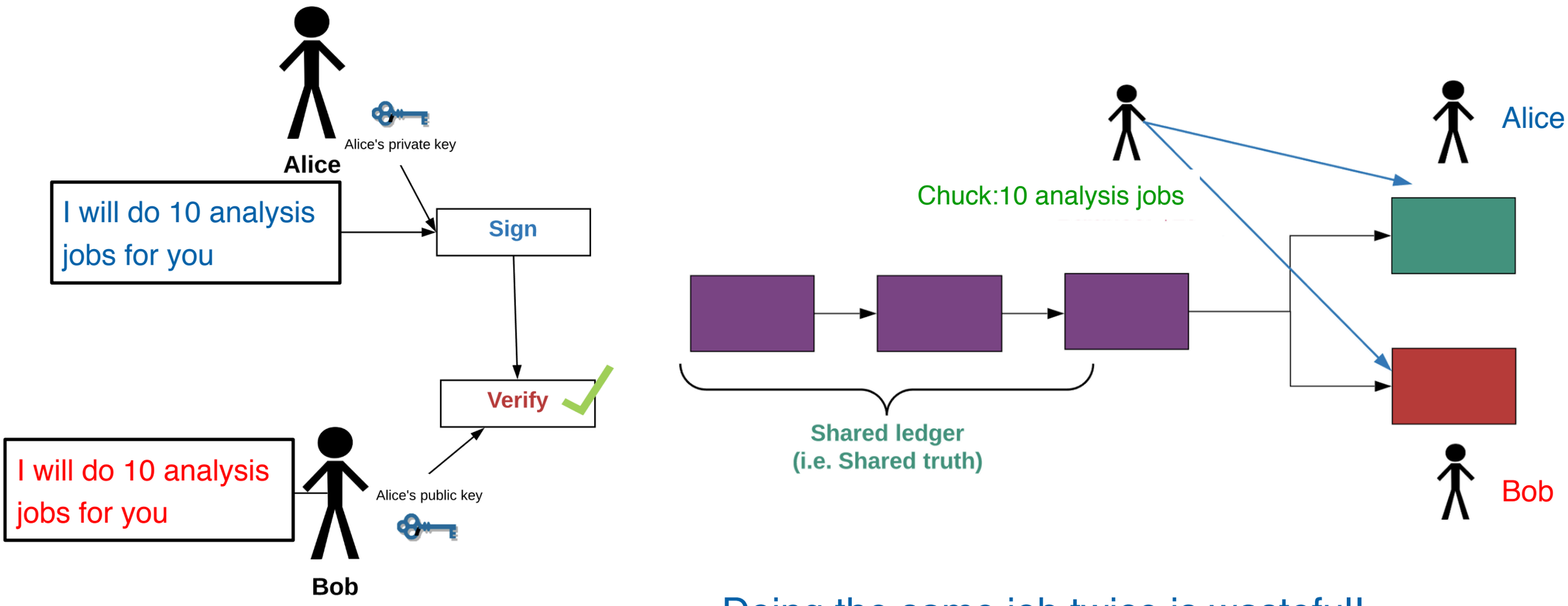
- **Bitcoin network capacities: 80 zettaFLOPs (10^{21}), storage: 1.7 PB**
  - The question arises: Could we use this immense amount of computing power to really do science in a way that produces publication quality, trustable data?
  - The essential incentive is there but demonstrating valid execution is much harder!

https://bitnodes.earn.com/ https://www.blockchain.com/en/charts/blocks-size https://bitcoincharts.com/bitcoin/

How can we apply this to HEP? (simple example)

Account Balance -> Job Queue

Calculating a Hash -> Doing some useful scientific work



Doing the same job twice is wasteful!

Must ensure no one tricked into doing same job twice.

But how do we ensure people do the work that we want them to?

Fermilab

Limitations of 1st Generation Blockchain Technology

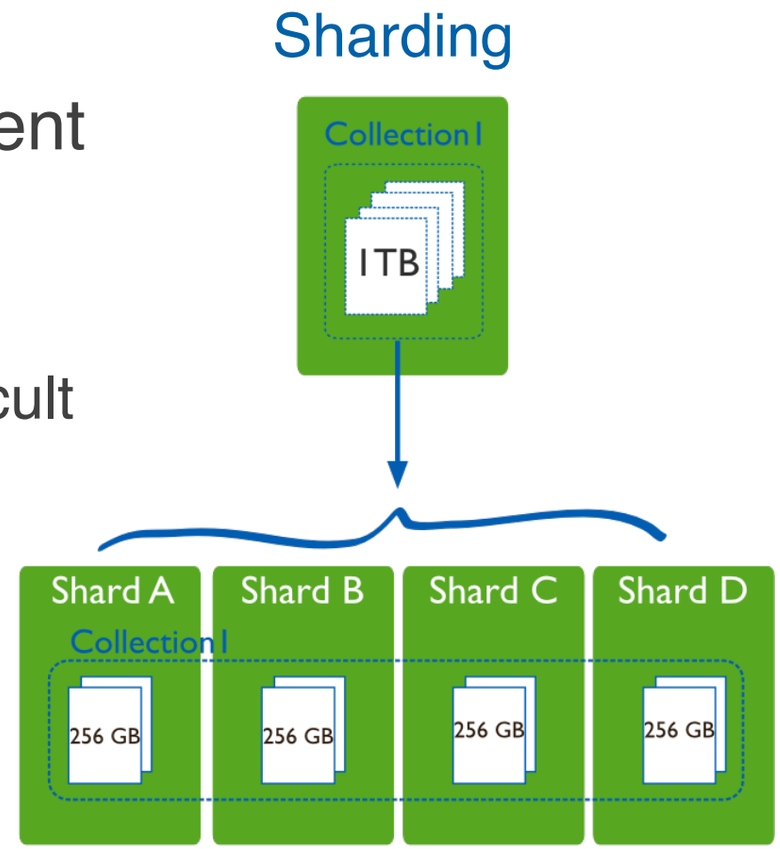
- Despite the immense interest and public adoption of this technology it is a first generation software product and has its flaws

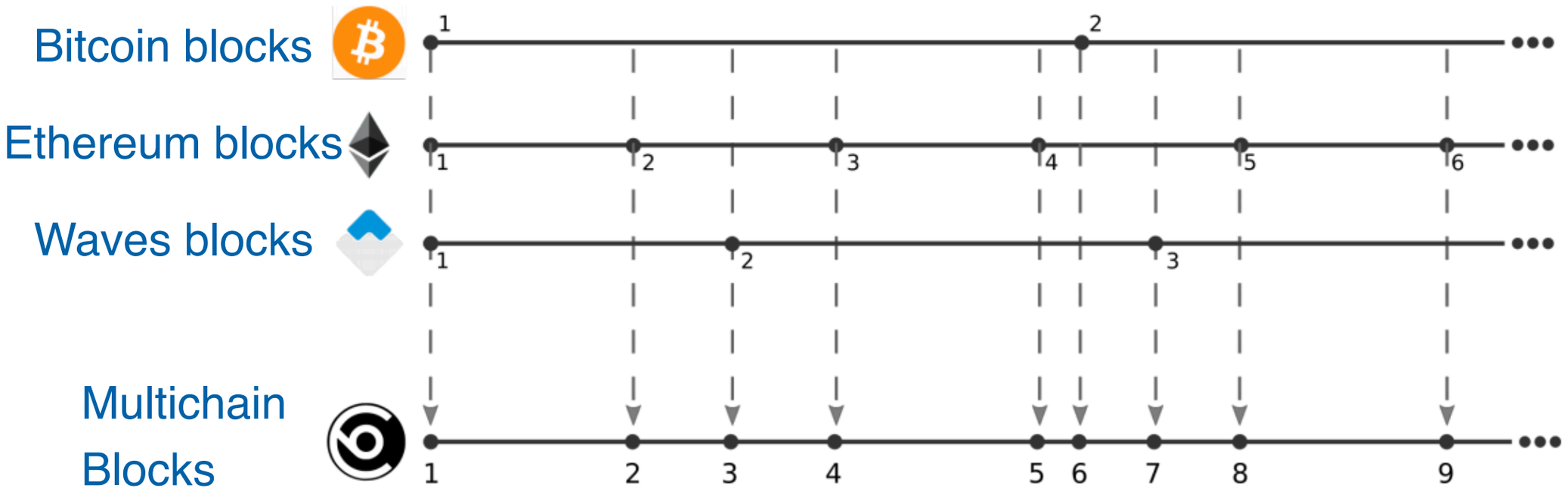
- Bad scaling as network size increases (bitcoin avg. transaction time is 600s)
  - The number of possible transactions exponentially increases with the network size
  - Takes longer and longer to validate each transaction due to backlog
  - Not a desirable quality for developing computing systems, our ‘tempo’ is constant

- Blockchains do not talk to each other without a central authority to dictate those interactions
  - This is very much against the ethos of a decentralized community
  - For a computing infrastructure we’d likely want different blockchains representing the state of different programs, so this is a serious impedance to deploying any sort of decentralized computing infrastructure at present

2nd Generation Blockchains and Interoperability

- Solutions to these issues are currently in development
- Creation of sharded blockchains to maintain faster transactions with growing volume are being studied
  - Maintaining global consensus in this regime becomes difficult and is eventually limited by the same problems
- Blockchains which examine and record the state of other blockchains in the process of mining
  - Accumulating and tracking state can join ledgers together





Fermilab

A Few More Tools: Smart Contracts

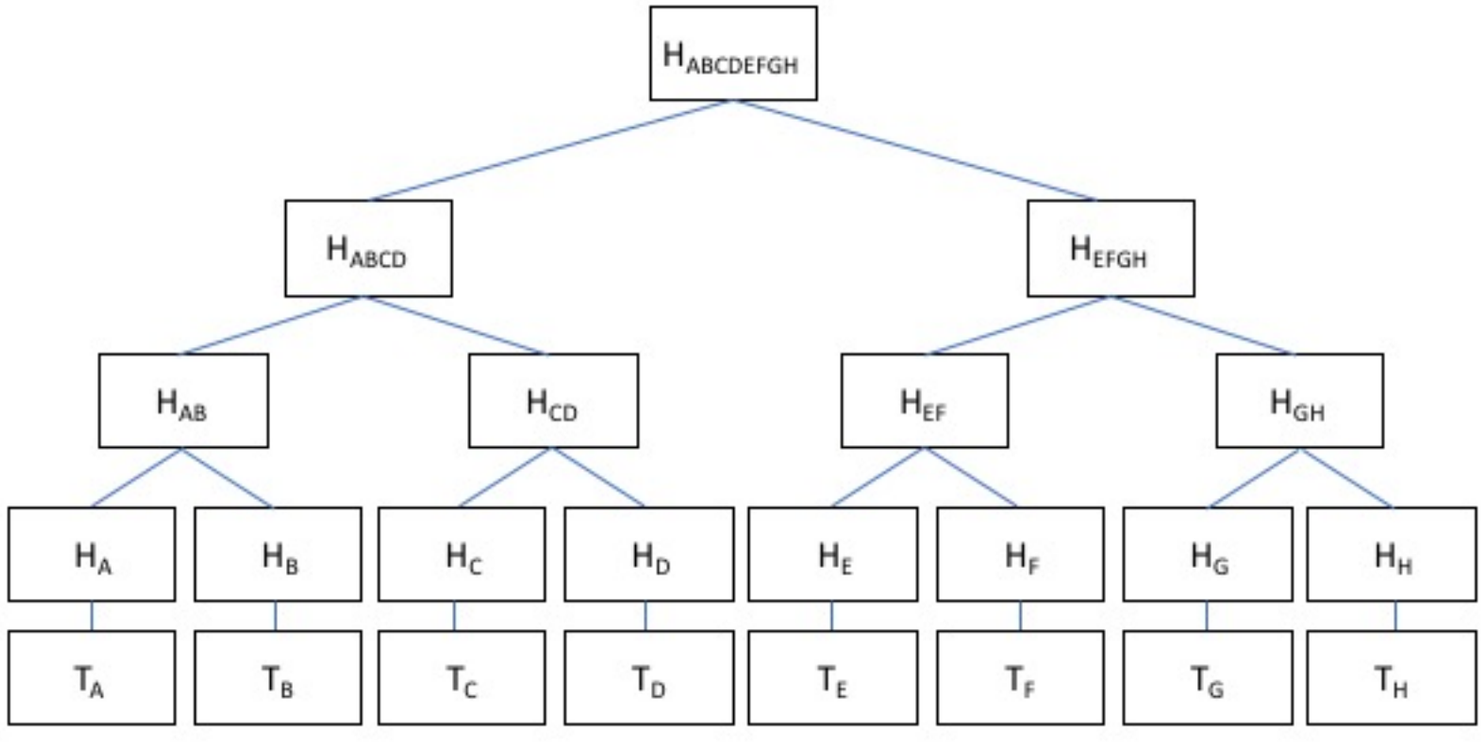
- Newer blockchains, Ethereum for instance, implement virtual machines that can execute byte code
- Smart contracts, implemented in this code allow binding between blockchain addresses and actions that are taken by the code
  - Typically the same code gets executed by all nodes in the network (extension of Nakamoto consensus)
- This can be used to implement a huge range of tasks
  - sub-currencies
  - timed payments
  - running of mathematical proofs
- Limited by blockchain transaction speed

A simple example of a derived currency:

```
pragma solidity ^0.4.21;

contract Coin {
    // The keyword "public" makes those variables
    // readable from outside.
    address public minter;
    mapping (address => uint) public balances;

    // Events allow light clients to react on
    // changes efficiently.
    event Sent(address from, address to, uint amount);

    // This is the constructor whose code is
    // run only when the contract is created.
    function Coin() public {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        if (msg.sender != minter) return;
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        if (balances[msg.sender] < amount) return;
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Merkle Trees for Fast Verification of Hierarchal Relationships

- A binary tree where:
  - leaf nodes contain some data and its hash
  - non-leaf nodes hash together the hashes of daughter nodes
- Can be used to create fast, verifiable indexing of sequential events
  - Storage medium for all transaction data in most cryptocurrencies
  - Used in cvmfs to index and then quickly validate large file stores that are distributed
- In general, can create a verification structure for any sequenced data
  - Interesting use cases arise when applied to verifying the execution of algorithms



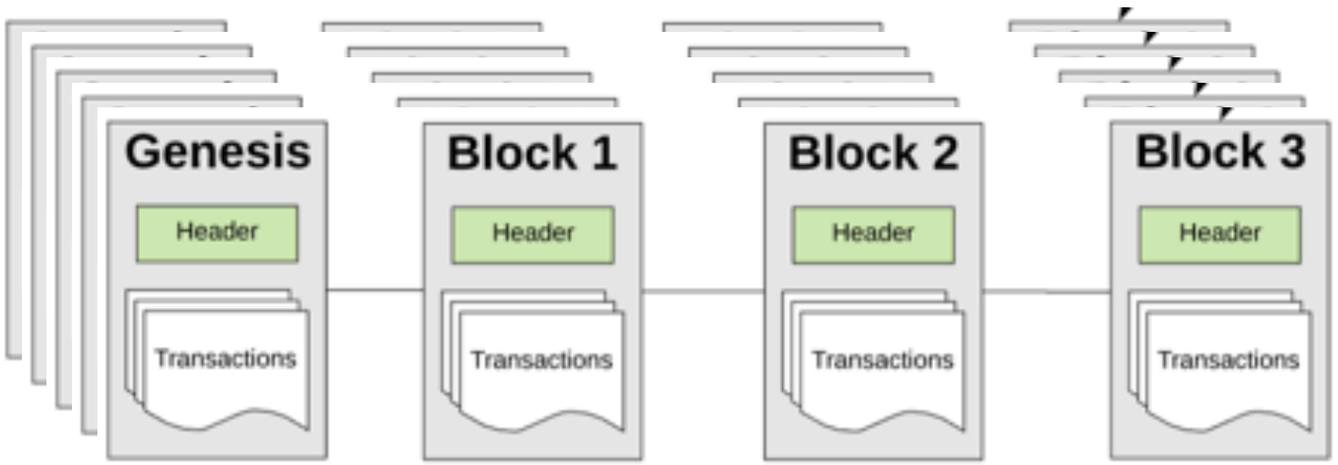
Fermilab

The Standard Model - of Cryptography

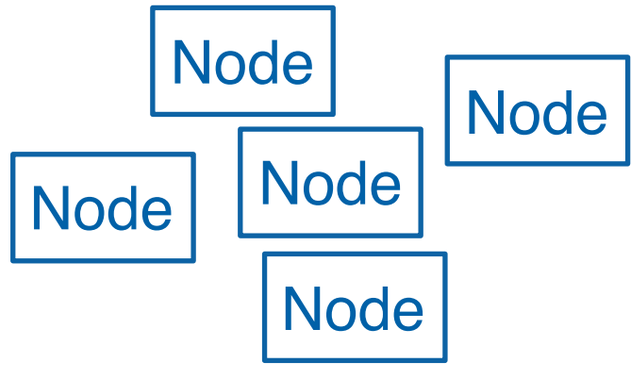
- This is a framework for proving the security of a cryptographic scheme where the adversary (person trying to read your data) is only limited by time and computing power
  - Proof of security here requires rigorous proof of the difficulty of a given cryptographic algorithm
  - Only complexity theory matters, leading to asymmetric encryption algorithms based on prime factorization, etc.

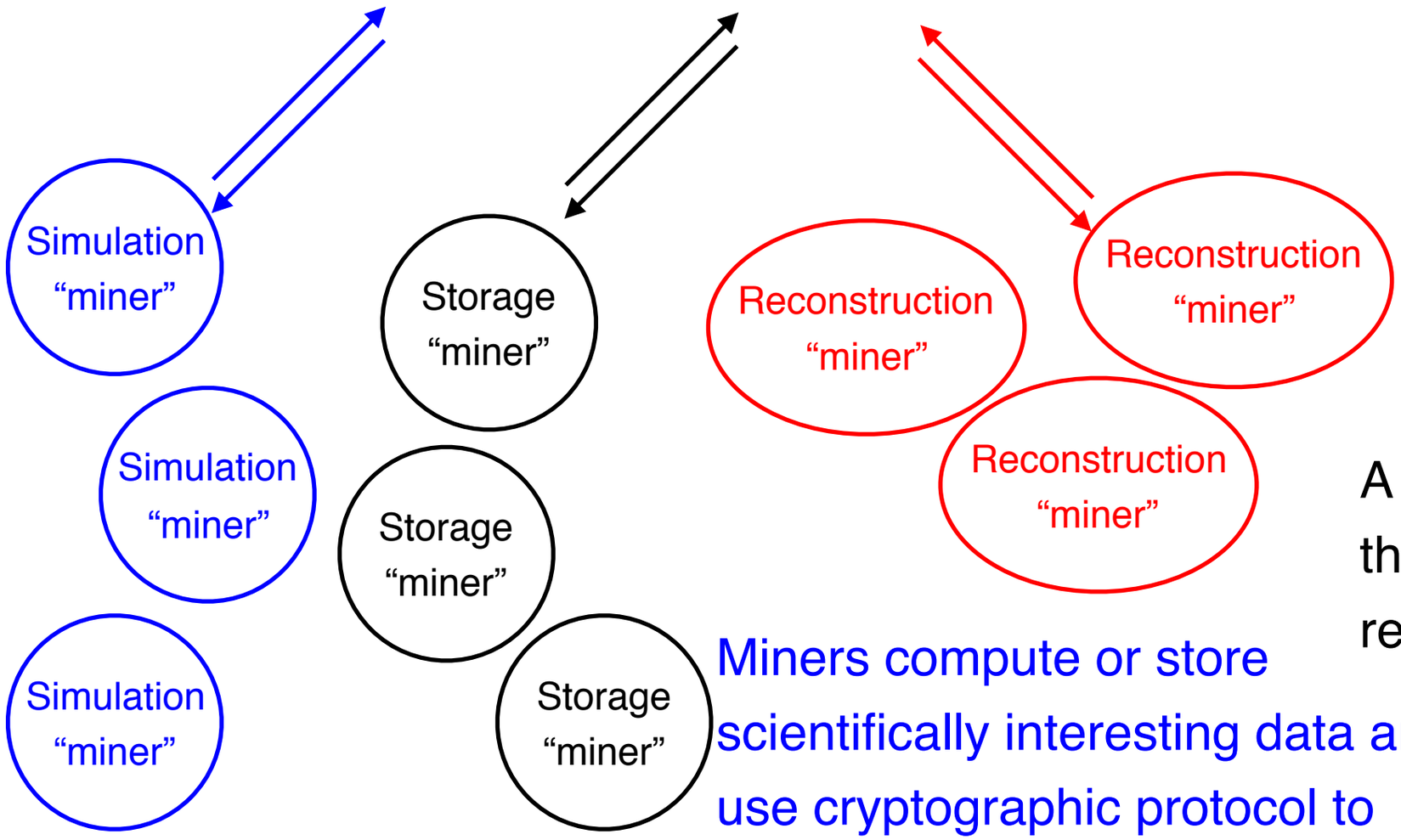
- There are other specializations of this model where certain assumptions are replaced, we'll focus on one later:
  - "Random Oracle Model" where you replace every cryptographic hash function with a very randomizing, but deterministic, function and then demonstrate cryptographic security using these hashes instead of an algorithm that's proven hard by complexity
  - The function used is the "oracle", since it gives seemingly random but deterministic output given some input

Putting it all together for HEP Computing (simple example!)



Nodes define needed resources and available jobs





The blockchains maintain consensus over job states and quality of results with smart contracts and transaction logs

A further blockchain maintains the consensus between all of respective sub-chains.

Miners compute or store scientifically interesting data and use cryptographic protocol to prove fidelity of results for reward!

All powered by useful computations rather than hashing!

Fermilab

1st Attempts at Blockchain-Incentivized Distributed Computing

- Gridcoin
  - BOINC client augmented with blockchain transactions
  - Voting to include projects in list of possible tasks
  - Rewards distributed based on successful execution of a valid task
  - Uses validation mechanisms of BOINC (inefficient!)
  - Actually exists, but has poor uptake, not a generational leap in computing model or blockchain itself
- AElf
  - Aim to implement decentralized cloud computing
  - End goal is to have a sharded set of blockchains representing workflows managed by smart contracts
  - Public code base is mostly empty :-(
- Golem Network
  - Functioning distributed graphics rendering engine in current implementation
  - Roadmap towards scientific computing but examination of codebase indicates a long way to go

https://gridcoin.us/

https://aelf.io/

https://golem.network/

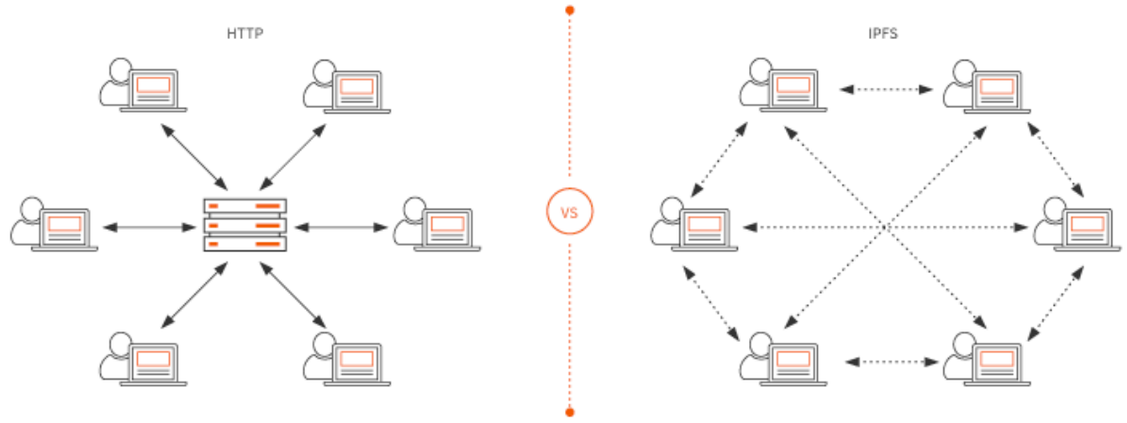
NB: List not exhaustive

Fermilab

Possible Applications: Data Storage & Databases

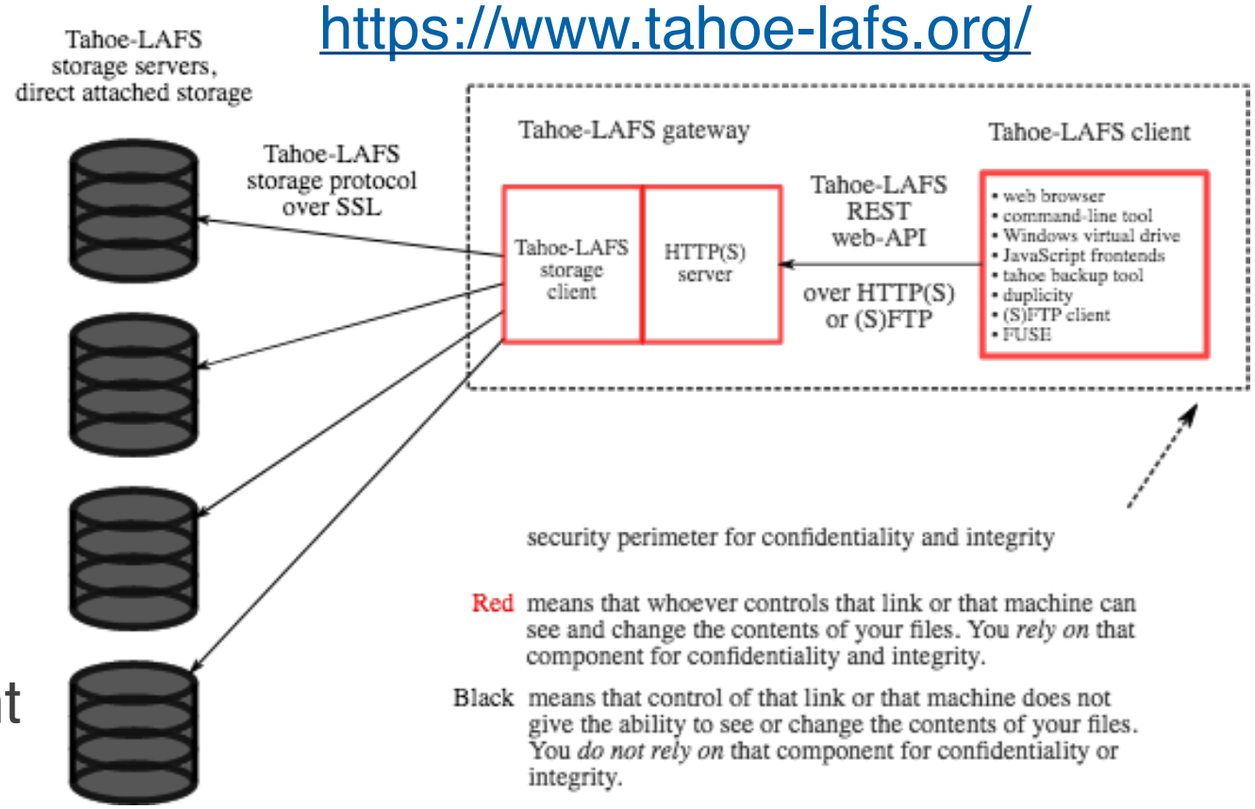
- One missing feature of the previous projects is they are all computing projects without storage
  - However in terms of cryptographic approach solutions already exist here
  - IPFS: caching / fast access cryptographically tracked file system
  - LAFS: striped encrypted file system with erasure coding for recovery
  - Possibly looking similar to serverless cvmfs
- There have been attempts to track states of these filesystems or gather contributed space with a blockchain
  - Many demonstrations of this heavily rely on AWS nodes, nothing ‘in the wild’
  - However, a fast decentralized and content-addressable filesystem would have excellent performance and few bottlenecks

IPFS Architecture https://ipfs.io/



Tahoe-LAFS architecture

https://www.tahoe-lafs.org/

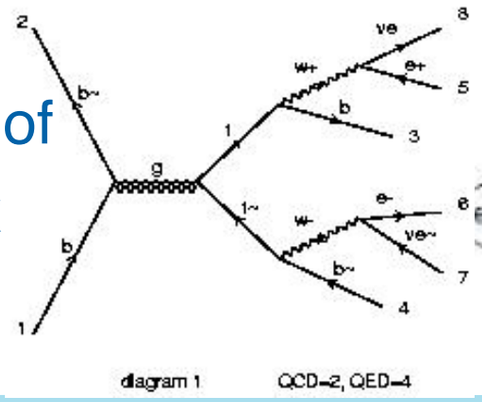


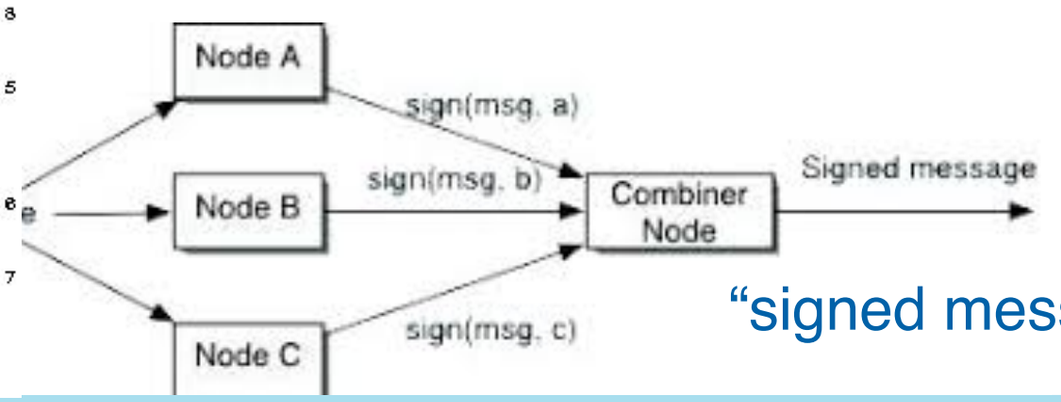
Fermilab

Random Oracles & Monte Carlo Generators

- Reminder: Random Oracle, highly randomizing deterministic function
- In particle physics we have quite similar functions: Matrix Elements
  - Start from some random byte string and produce a highly random but deterministic set of 4-vectors
- An interesting property of some schemes involving random oracles is that if everyone is sharing the same random oracle they can contribute to collectively signed cryptographic keys
  - Private keys are generated and distributed to everyone belonging in the given group using said oracle
  - A valid key can be generated so long as there are fewer than some set amount of bad actors contributing to the collective key
- This means there is a very clear and straightforward way to cryptographically verify that a set of random machines are using a distributed MC generator!

"message" of ttbar matrix element



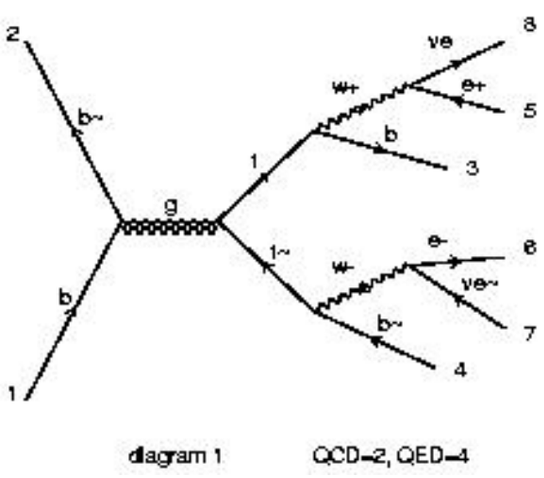


"signed message" -> "signed dataset"

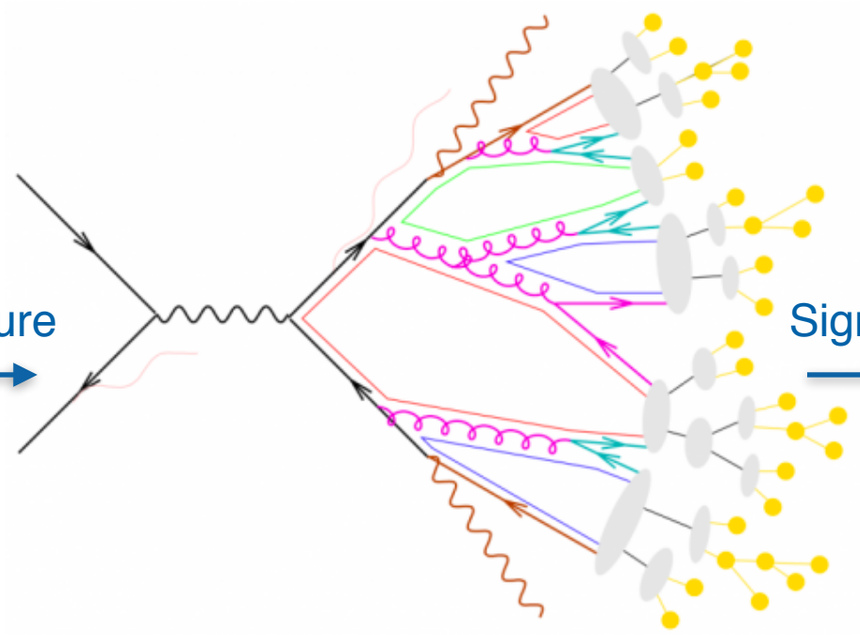
Fermilab

Distributed Computing - Simulation

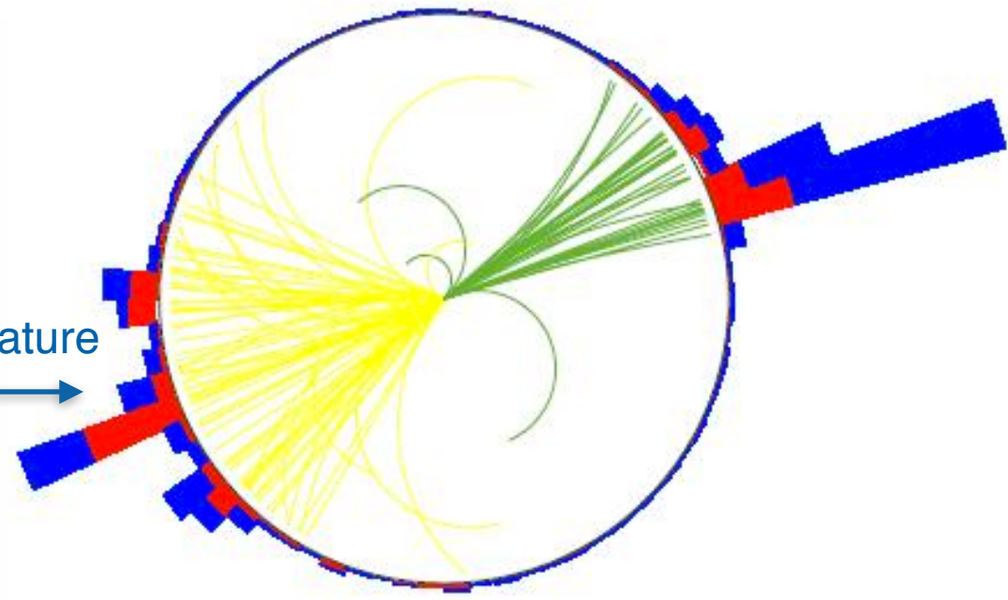
- Starting from (likely) being able to collectively sign low-level MC events it is then possible to have signed event samples comprised of signed events
  - This is extreme for provenance tracking but if you are not allowed to trust your processing nodes this may be needed
- Similar to generating MC events from a matrix element
  - The steps of showering, hadronization, and propagation may be broken down in to similar collective random processes and used to sign some collective key
  - Possible to prove a monte carlo event step by step until digitization takes place
  - Computational overhead from this methodology would need to be carefully monitored, but it is still better than running an event in triplicate



Signature



Signature

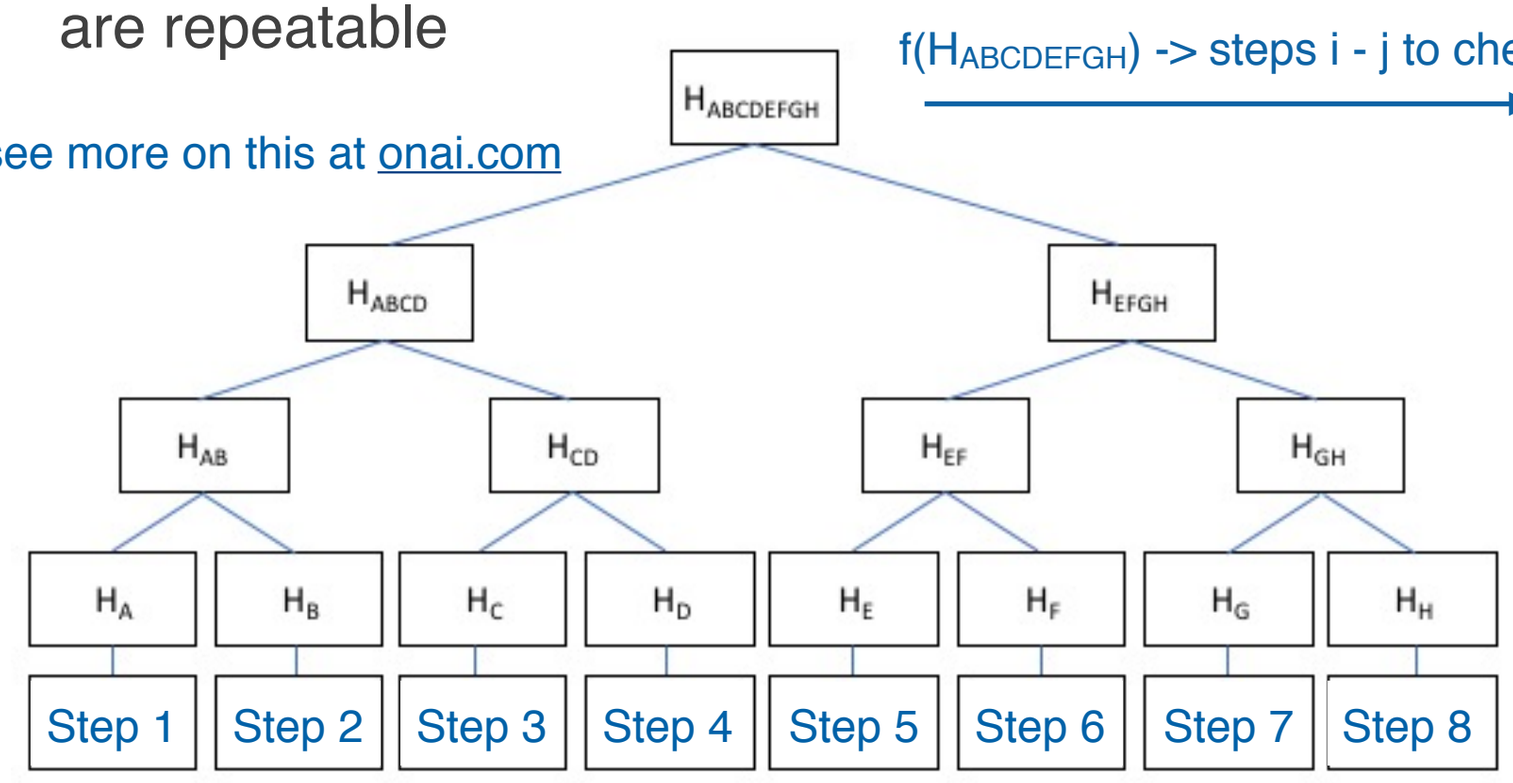


Fermilab

Embedding Algorithms in Merkle Trees

- Instead of some transaction payload use the leafs of the Merkle tree to piecewise track the steps of some iterative algorithm
  - For each state save the inputs and the outputs
  - Assuming there is some random data input into the algorithm (a measurement or some dice roll in a simulation)
- Use the Merkle root hash to determine which steps of the algorithm you will check
  - A validation mechanism of some sort can then determine if the selected algorithm steps are repeatable

see more on this at onai.com



$f(H_{ABCDEFGH})$ -> steps i - j to check

Run steps i to j

Compare to original records:

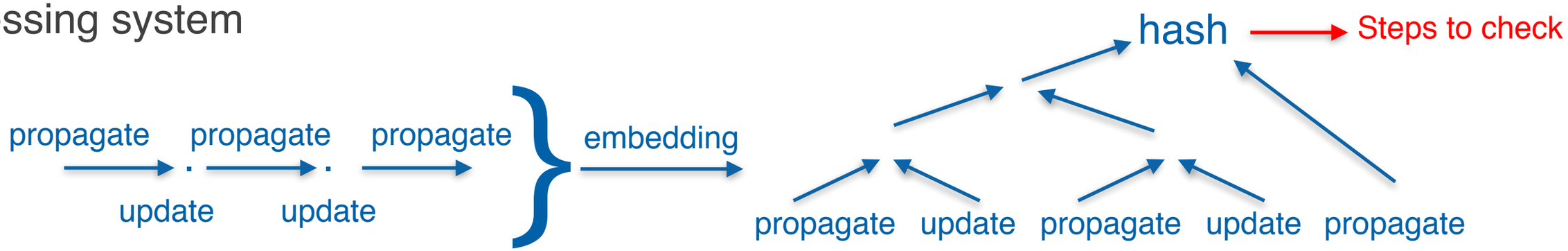
Match: **Reward worker, new tasks**

Mismatch: **Resubmit and penalize worker**

Fermilab

Distributed Computing Example: Kalman Filter - Algorithm

- A very clear application of this Merkle tree embedding is in reconstruction
  - Even statistical algorithms like simulated annealing for vertexing can be embedded this way
- Started by implementing a Kalman filter in this formalism using hits and reconstruction from CMS simulation
  - Presently to the point of writing down all the Merkle embeddings
  - Pixel and strip clusters are very good sources of random numbers, ensuring that the points that are checked in the algorithm are well randomized, reducing the possibility for cheating
- It is possible to remove this checking on trusted nodes, recovering any computing efficiency loss on known nodes, and increasing proportional to the "mistrust" of any unknown processing node
  - Tunable computational efficiency is a very good incentive for contributing positively to a processing system



Verifiable Computation - Freivalds' Algorithm

- It is possible to further validate the Kalman filter by examining matrices
- A statistical algorithm for proving the correctness of a matrix product
  - Proves that A*B = C with a false positive rate of (1/2)^k , with k trials.
  - Implementable in O(k*N^2) time, which for sufficiently small k is faster than the matrix multiplication itself
- Keep choosing random vectors r in k trials
  - If the result of the difference between the matrix-vector products is 0 then it is likely the product and the resulting matrix are equal, iterate until satisfied

Example of efficient algorithm implementation for N = 2:

$$
\begin{aligned}
A \times (B\vec{r}) - C\vec{r} &= \begin{bmatrix} 2 & 3 \\ 3 & 4 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) - \begin{bmatrix} 6 & 5 \\ 8 & 7 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} 2 & 3 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} - \begin{bmatrix} 11 \\ 15 \end{bmatrix} \\
&= \begin{bmatrix} 11 \\ 15 \end{bmatrix} - \begin{bmatrix} 11 \\ 15 \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ 0 \end{bmatrix}.
\end{aligned}
$$

but…

so these matrices are not equal!

$$
A \times (B\vec{r}) - C\vec{r} = \begin{bmatrix} 2 & 3 \\ 3 & 4 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) - \begin{bmatrix} 6 & 5 \\ 8 & 7 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.
$$

Distributed Computing Example: Kalman Filter - Matrices

- Kalman updates are typically very unique matrices
  - Based on residuals of predicted state with measured state
  - Should be very easy to detect if someone, for some reason, wants to upset track fits by altering the algorithm in some nefarious way
  - Any tampering would certainly show up in this step!

- If, for instance, an error is detected in a computing node which is running kaman filtering and fits the level of error detection can be tuned and the depth at which errors are searched for can be increased
  - This way we develop a further incentive to not disrupt the physics algorithms being executed on a computer by having varying levels of penalization that make a particular computing node less efficient if its results are known to be faulty

- Similarly to the Merkle embedding, if a processing node is known to be part of a trusted sub-network (like a previously existing tier 2) all validation mechanics can be lifted and work can proceed at a fully optimized pace

Concluding Remarks & Next Steps

- Blockchain is a powerful tool for organizing and incentivizing distributed processing systems that has applications far beyond cryptocurrencies
  - Distributed consensus allows the building of arbitrary trustless systems
  - Major limitation right now is the cryptographic technology to implement those systems

- The cryptographic tools necessary to deploy publication-quality computations on contributed hardware are starting to become available
  - Verifiable computation, and statistical algorithms for validation

- Framework for verifying algorithms is possible, next demonstrate its efficacy
  - Statistical error detection using Merkel trees, Freivald's algorithm for known offenders
  - Clear next step is to demonstrate embedded of error catching actually in a blockchain

- The coupling of these two aspects allows for the creation of global processing networks that could greatly benefit scientific computing
  - The full use of this technology will only come from further community engagement
  - It is an untapped direction in sustaining our ever-growing computing needs

Fermilab

Acknowledgements

Block Collider

blockcollider.org

Zilliqa

zilliqa.com

ONAI®

onai.com

Patrick McConlogue

Patrick Schilz

Stefano Schiavi