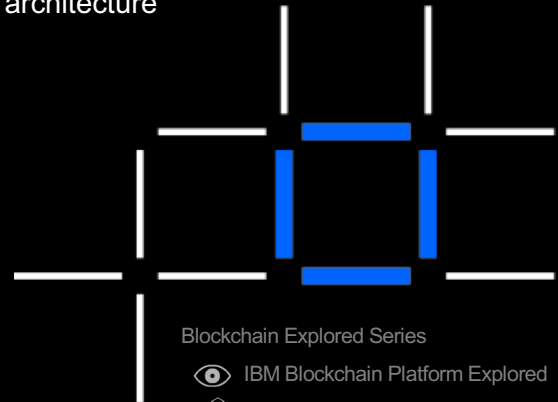







Architectures Explored

What's inside blockchain, and how it fits in a systems architecture



Blockchain Explored Series


-  IBM Blockchain Platform Explored
-  Modeling Applications
-  **Architectures Explored**
-  Fabric Explored
-  What's New in Tech

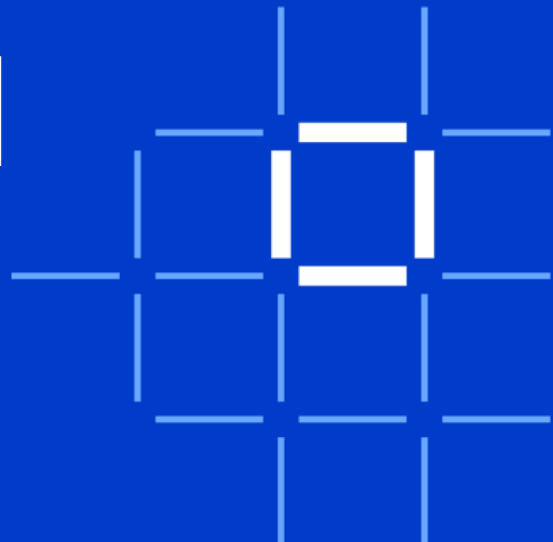
V3.01, 1 March 2019

IBM Blockchain



 **Blockchain Data Structures**
Describing the blockchain using computer science principles

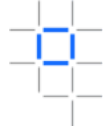
 **Operational Considerations**
Consensus, integration, security, business and non-functional requirements



IBM Blockchain

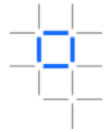


Note

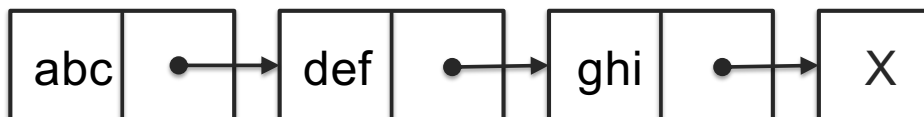


- Blockchain implementations vary
- We'll try and focus on what's common and point out when implementations vary

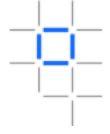
The Linked List



- Linear collection of data elements
- Each element is linked to the next
- Concept dates from 1955



One-Way Hash Functions



- Any function that can be applied to a set of data that is guaranteed to produce the same output for the same input
- One-way means that you can't derive the input from the output
- Often, outputs are *unlikely* to repeat for different inputs
- Forms the basis of much cryptography

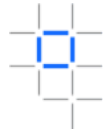
$$h(abc) = 7859$$

$$h(def) = 8693$$

$$h'(7859) = ?$$

$$h(abc) = 7859$$

The Hash Chain



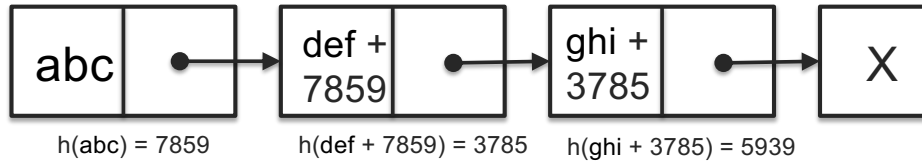
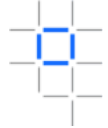
Hash chain: A successive application of a hash function

$$h(h(h(abc))) = 1859$$

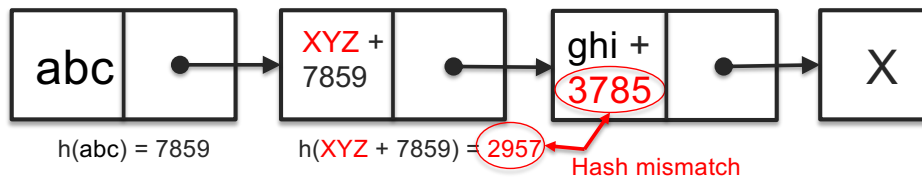
Can combine new data with each successive hash to produce a tamper resistant linked list

$$h(ghi+h(def+h(abc))) = 5783$$

How is This Tamper Resistant?



- Any modification to a data element means that the hashes will not match up
 - You would need to recreate the downstream chain

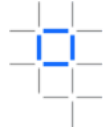


IBM Blockchain

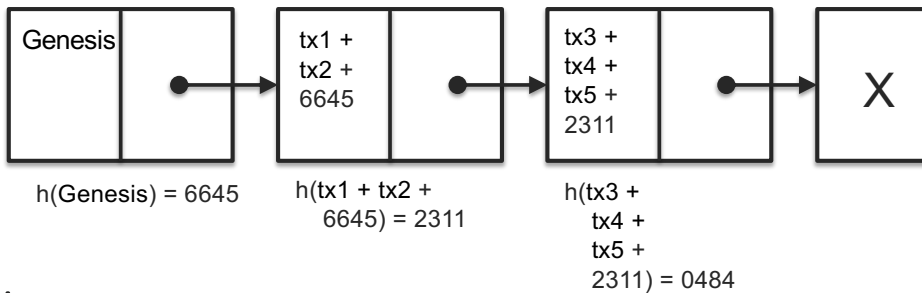
IBM

7

Applied to Blockchain



- A blockchain is a hash chain (*with optimizations* that we'll cover shortly)
- Each element (block) in the linked list is a set of zero or more transactions
 - Transactions are an implementation-dependent data object
- First block known as a genesis block
 - May contain some identifying string or other configuration metadata

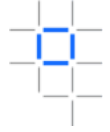


IBM Blockchain

IBM

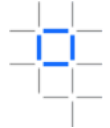
8

Some Problems with This Approach

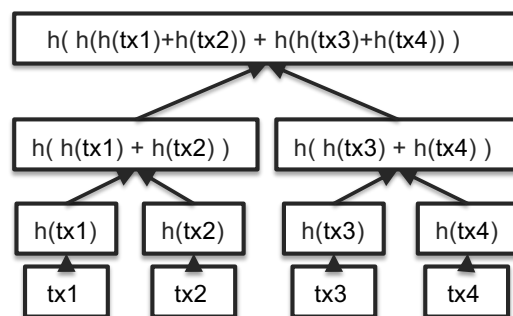


- In the event of tampering, it can be difficult to identify which transaction was modified (particularly when there are many transactions in a block)
 - It is not feasible to have one transaction per block
- It requires all transaction data in order to retain integrity of chain
- Searching transactions is linear (time consuming)

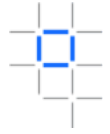
Merkle Tree



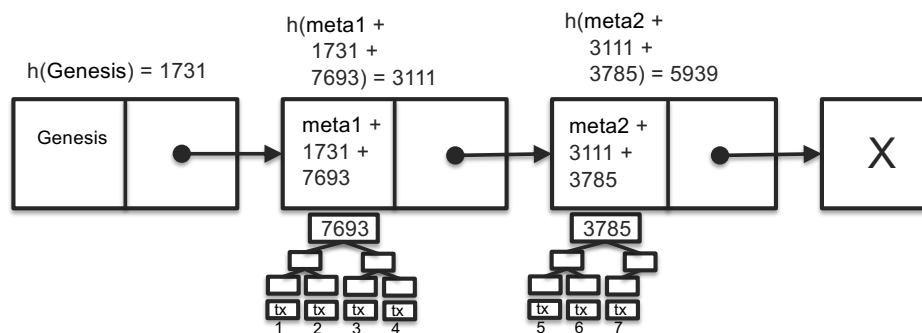
- It is possible to optimise the chain data structure if we arrange it as a tree
 - Makes it easier to identify tampering without sacrificing stability
 - Makes it quicker to traverse
- However, this makes it impossible to add new transactions without re-hashing root nodes



Combining Tree and Chain



- Each element in the chain contains:
 - A pointer (“Merkle root”) to the tree of transactions
 - Other metadata (e.g. timestamp)
 - A hash of the previous block’s data (i.e. Merkle root, metadata and hash)

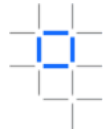


IBM Blockchain

IBM

11

Benefits of This Approach



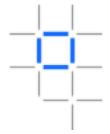
- It allows tampered transactions to be identified easily
- More [efficient search](#) within a block
 - $O(\log N)$ rather than $O(N)$
- Allows transaction detail to be stubbed
 - Bitcoin has a [Simplified Payment Verifier](#) (SPV) concept: a type of user that doesn't have the entire tree available, just the Merkle roots
 - Note it is also possible to checkpoint and archive old blocks, creating a new Genesis block mid-way through the chain

IBM Blockchain

IBM

12

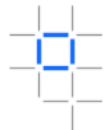
Common Transactions



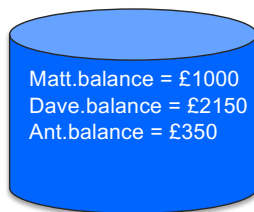
- What's Dave's balance?
- Does Matt have funds to clear a £1000 transaction? (Assuming no overdraft)

#	Transaction	Initiator	Receiver	Amount
1	Create a/c	Cash	Matt	£1000
2	Create a/c	Cash	Dave	£2000
3	Transfer	Matt	Dave	£100
4	Create a/c	Cash	Ant	£500
5	Transfer	Ant	Matt	£50
6	Transfer	Ant	Dave	£200
7	Transfer	Dave	Matt	£100
8	Transfer	Dave	Ant	£50
9	Transfer	Matt	Ant	£50

World State

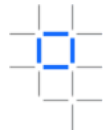


- It is clearly not feasible to reparse the entire transaction log to complete a new transaction
- Blockchains often include an associated database (world state) – e.g. Hyperledger Fabric
- Transactions become a set of creates, reads, updates and deletes of records in this data store

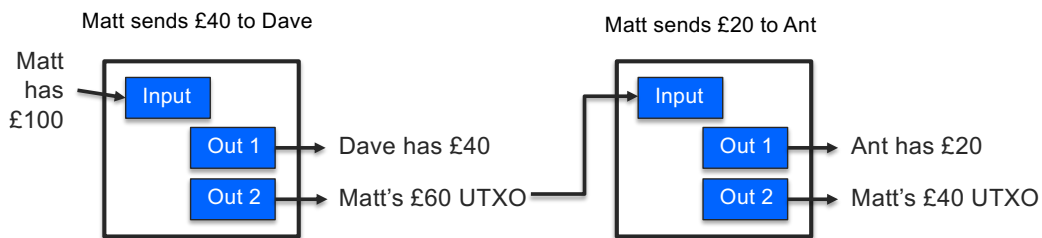


#	Transaction	Initiator	Receiver	Amount
1	Create a/c	Cash	Matt	£1000
2	Create a/c	Cash	Dave	£2000
3	Transfer	Matt	Dave	£100
4	Create a/c	Cash	Ant	£500
5	Transfer	Ant	Matt	£50
6	Transfer	Ant	Dave	£200
7	Transfer	Dave	Matt	£100
8	Transfer	Dave	Ant	£50
9	Transfer	Matt	Ant	£50

Unspent Transaction Outputs



- Some blockchains (e.g. Bitcoin) don't maintain balances
 - Transactions are linked to earlier transactions using an ID (TXID)
 - Outputs always equal inputs
 - Unspent funds are marked as an "Unspent Transaction Output" (UTXO)
 - Only UTXOs can be used as inputs (to prevent double spending)
 - Your "balance" is the aggregation of all of your UTXOs
- In Bitcoin, if your application doesn't specify the UTXO output then the miner gets the excess!

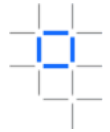


IBM Blockchain

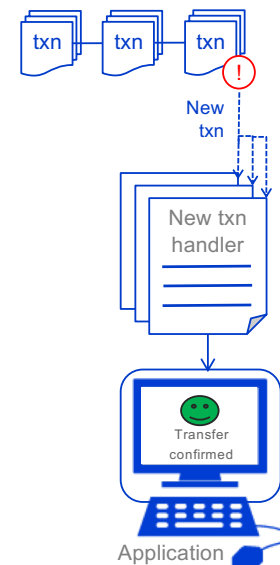
IBM

15

How Events are Used in Blockchain



- In computing, an **event** is an occurrence that can trigger handlers
 - e.g. disk full, fail transfer completed, mouse clicked, message received, temperature too hot...
- Events are important in asynchronous processing systems like blockchain
- The blockchain can emit events that are useful to application programmers
 - e.g. Transaction has been validated or rejected, block has been added...
- Events from external systems might also trigger blockchain activity
 - e.g. exchange rate has gone below a threshold, the temperature has gone up, a time period has elapsed...



IBM Blockchain

IBM

16

Blockchain Data Structures
Describing the blockchain using computer science principles

Operational Considerations
Consensus, integration, security, business and non-functional requirements

IBM **Blockchain** IBM

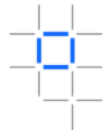
The Power of the Network

- These data structures are just bytes on disk
- Can still be manipulated or destroyed (e.g. by a DB admin)
- Proof (and trust) in the blockchain comes from the power of the network...

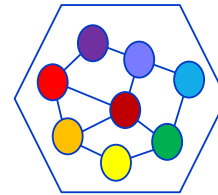
IBM **Blockchain** IBM

18

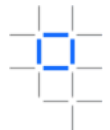
Network Nodes



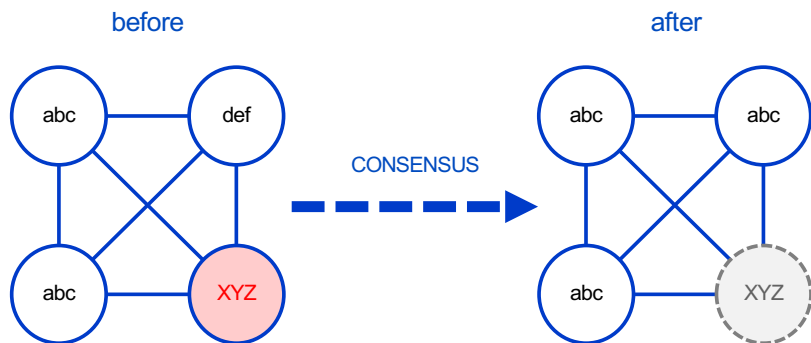
- A blockchain network comprises a set of nodes that share information
 - Usually peer-to-peer
 - Some blockchains are worldwide, others are private to a business network
 - It might make sense to have one node per business network participant, but this is not necessarily so
- Responsibilities include
 - Holding and maintaining the ledger
 - Receiving transactions from applications (and other nodes)
 - Validating transactions
 - Notifying applications about the outcome of submitted transactions
- There is an assumption that **some nodes might be malicious!**
 - Different networks require different tolerances for malicious behaviour



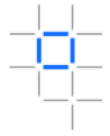
The Art of Maintaining a Consistent Ledger



- Keeping nodes up-to-date
- Fixing any peers in error
- Ignoring all malicious nodes



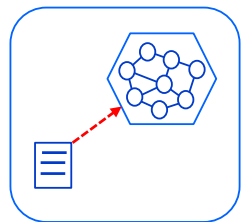
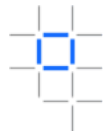
Consensus Algorithms



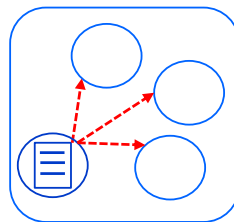
- There are lots of ways of achieving this
 - Proof of Work (e.g. Bitcoin, Ethereum)
 - Proof of Stake (e.g. NXT)
 - Proof of Elapsed Time (e.g. Sawtooth)
 - BFT-based (e.g. Iroha)
 - Apache Kafka/Zookeeper-based (e.g. Fabric)

- Different algorithms have different qualities of service
 - Tolerances for malicious behavior
 - Compute requirements
 - Performance characteristics
 - Need for intrinsic incentives

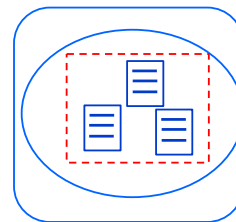
One of Many Transaction Flow Implementations



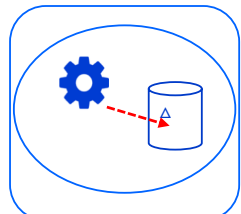
1. The application submits a request to invoke a transaction



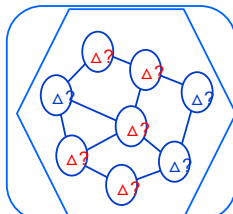
2. The transaction is shared around the network



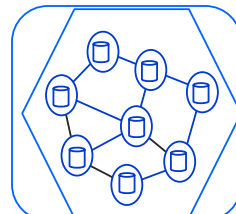
3. A designated peer creates a block containing the transaction



4. The block's transactions are executed and output stored in a delta

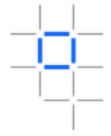


5. The network attempts to agree the correct result



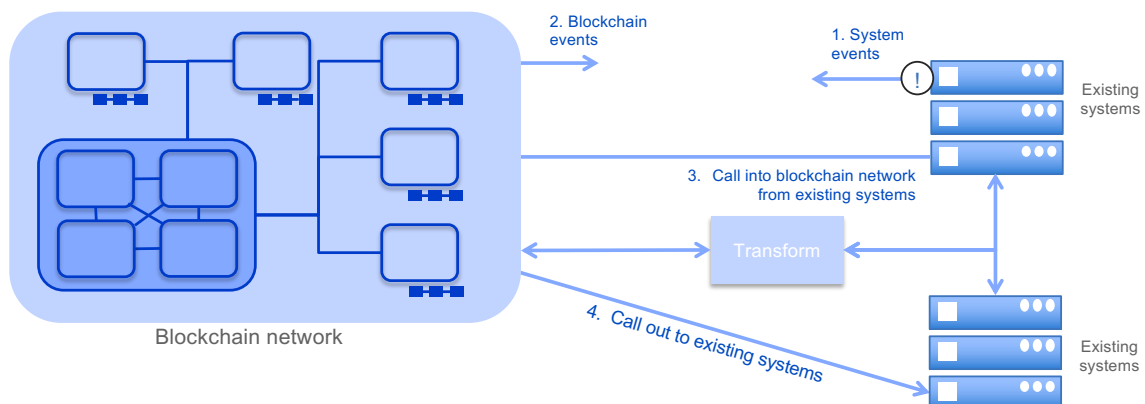
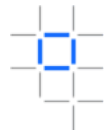
6. If there is agreement, the correct output is applied to the world state

A Note on Cryptographic Mining



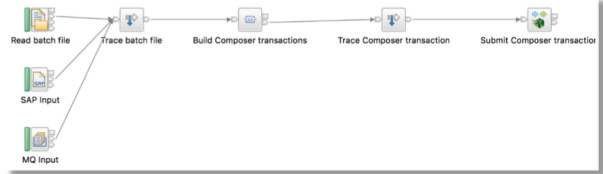
- Cryptographic Mining is a by-product of Proof of Work (PoW)
 - It makes no sense with other consensus mechanisms
- In PoW, nodes show they are legitimate by proving to other nodes that they have burned electricity
 - They do this by revealing the answer to difficult cryptographic puzzles
 - This causes other nodes to add the solver's version of events (block) to their chain
- The first solver (i.e. the producer of the block) gets rewarded
 - A bounty of 12.5 bitcoins (this halves every 210,000 blocks)
 - Any transaction fees present
- The Bitcoin community refers to this "mining", as running a node can occasionally result in Bitcoin rewards

Integrating with Existing Systems – Possibilities



Integrating with Existing Systems – Using Middleware

- Blockchain is a network system of record
- Two-way exchange
 - Events from blockchain network create actions in existing systems
 - Cumulative actions in existing systems result in Blockchain interaction
- Transformation between blockchain and existing systems' formats
 - GBO, ASBO is most likely approach
 - Standard approach will be for gateway products to bridge these formats
 - Gateway connects to peer in blockchain network and existing systems
- Smart contracts can call out to existing systems
 - Query is most likely interaction for smart decisions
 - e.g. all payments made before asset transfer?
 - **Warning: Take care over predictability: transaction must provide same outputs each time it executes...**



IBM **Blockchain**

IBM

25

Non-determinism in blockchain

- Blockchain is a distributed processing system
 - Smart contracts are run multiple times and in multiple places
 - As we will see, smart contracts need to run deterministically in order for consensus to work
 - Particularly when updating the world state
- It's particularly difficult to achieve determinism with off-chain processing
 - Implement oracle services that are guaranteed to be consistent for a given transaction, or
 - Detect duplicates for a transaction in the blockchain, middleware or external system

random()

getExchangeRate()

getDateTime()

getTemperature()

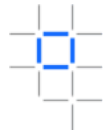
incrementValue
inExternalSystem(...)

IBM **Blockchain**

IBM

26

Security: Public vs. private blockchains



Public blockchains



- For example, Bitcoin
- Transactions are viewable by anyone
- Participant identity is more difficult to control

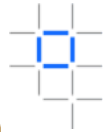


Private blockchains

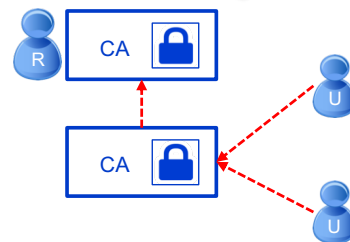
- For example, Hyperledger Fabric
- Network members are known but transactions are secret

- Some use-cases require anonymity, others require privacy
 - Some may require a mixture of the two, depending on the characteristics of each participant
- Most business use-cases require **private, permissioned blockchains**
 - Network members know who they're dealing with (required for KYC, AML etc.)
 - Transactions are (usually) confidential between the participants concerned
 - Membership is controlled

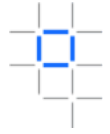
Security: Real-world vs. digital identity



- Consider **real-world identity** documents...
 - The issuers of the identity documents are trusted third parties (e.g. passport office)
 - There is usually a chain of trust (e.g. to get a bank card you need a drivers license or passport)
 - Identity documents are often stored in wallets
- In the **digital world**, identities consist of public/private key pairs known as certificates
 - Identity documents are issued by trusted third parties known as Certificate Authorities (CAs)
- Private blockchain networks also require CAs
 - So network members know who they're dealing with
 - May sit with a regulatory body or a trusted subset of participants

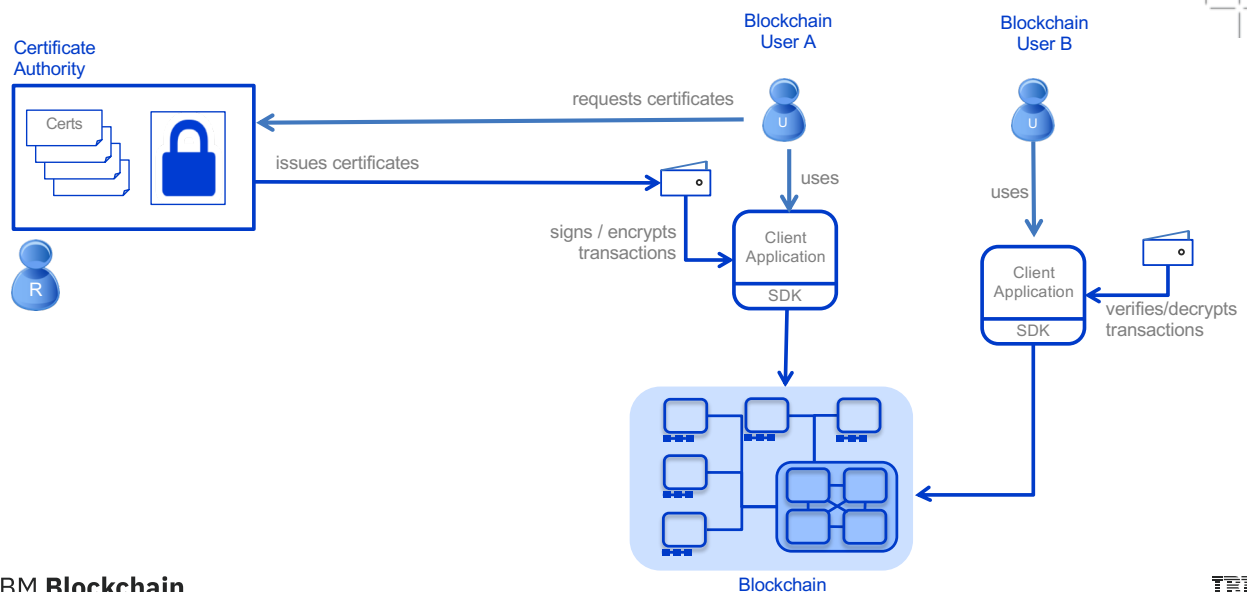
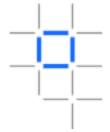


Security: Encryption and Signing

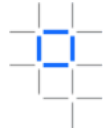


- Cryptography basics
 - Every member of the network has (at least) one public key and one private key
 - Assume that every member of the network knows all public keys and only their own private keys
 - Encryption is the process applying a transformation function to data such that it can only be decrypted by the other key in the public/private key pair
 - Users can sign data with a private key; others can verify that it was signed by that user
- For example
 - Alice can sign a transaction with her private key such that anyone can verify it came from her
 - Anyone can encrypt a transaction with Bob's public key; only Bob's private key can decrypt it
- In private, permissioned blockchains
 - Transactions and smart contracts can be signed to verify where they originated
 - Transactions and their payloads can be encrypted such that only authorized participants can decrypt

Certificate Authorities and Blockchain

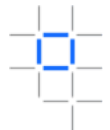


Business Considerations



- As a B2B system, blockchain adds a number of aspects that are not typical in other projects:
 - Who pays for the development and operation of the network?
 - Where are the blockchain peers hosted?
 - When and how do new participants join the network?
 - What are the rules of confidentiality in the network?
 - Who is liable for bugs in (for example) shared smart contracts?
 - For private networks, what are the trusted forms of identity?
- Remember that each business network participant may have different requirements (e.g. trust)
 - Evaluate the incentives of potential participants to work out a viable business model
 - Mutual benefit → shared cost (e.g. sharing reference information)
 - Asymmetric benefit → money as leveler (e.g. pay for access to KYC)

Trade-offs Between Non-Functional Requirements



Performance

- The amount of data being shared
- Number and location of peers
- Latency and throughput
- Batching characteristics

Consider the trade-offs between performance, security and resiliency!

Security

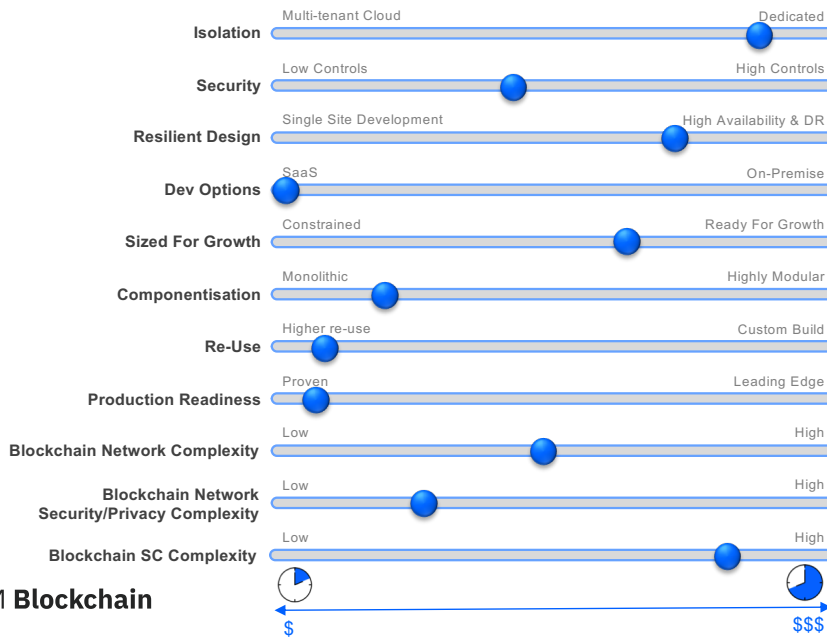
- Type of data being shared, and with whom
- How is identity achieved
- Confidentiality of transaction queries
- Who verifies (endorses) transactions

Resiliency

- Resource failure
- Malicious activity
- Non-determinism



Non-Functional Requirements



Adjust the sliders with the client early in the project so all parties are aligned on the expectations of robustness, isolation, security controls etc. as all these factors have material impact on the cost and complexity of the solution.

Summary

- Blockchain builds on **basic computer science** concepts:
 - Linked Lists
 - Hash Functions
 - Peer-to-peer networks
- Identify key **operational considerations**
 - Consensus is the art of maintaining a consistent ledger
 - It is possible to integrate with existing systems, but take care over determinism
 - Security requirements solved through techniques such as encryption and signing
 - Also consider business and non-functional requirements

Thank you

IBM Blockchain

www.ibm.com/blockchain

developer.ibm.com/blockchain

www.hyperledger.org

© Copyright IBM Corporation 2017. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represents only goals and objectives. IBM, the IBM logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.

