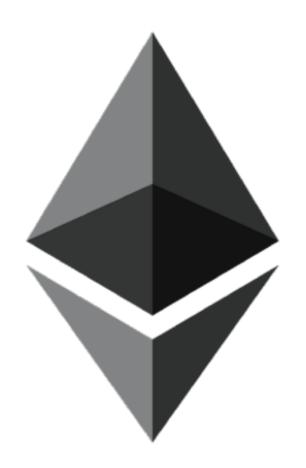
https://gitter.im/web3j/ jaxlondon



Building Ethereum ĐApps in Java with web3j



Conor Svensson conor@blk.io





Content

- Blockchain fundamentals
- Ethereum
- web3j
- Smart contracts
- RxJava in web3j
- Quorum



Schedule

- 09:00 09:45 Blockchain 101
- 09:45 10:30 Getting started with Ethereum
- 10:30 11:00 Break
- 11:00 12:30 Ethereum and web3j
- 12:30 13:30 Lunch
- 13:30 14:30 Smart contracts
- 14:30 15:00 RxJava
- 15:00 15:30 Break
- 15:30 16:30 Quorum
- 16:00 17:00 Back to web3j & wrap up



About me

- Developed trading/risk/regulatory platforms on the JVM
- Co-founded a couple of fintech startups in Australia
- web3j author
- Founded blk.io this year
- Enterprise Ethereum Alliance member
 - EEA London organiser
 - Co-chair of integration and tools working group
 - Co-chair of Quorum working group
- Bearish on ICOs...



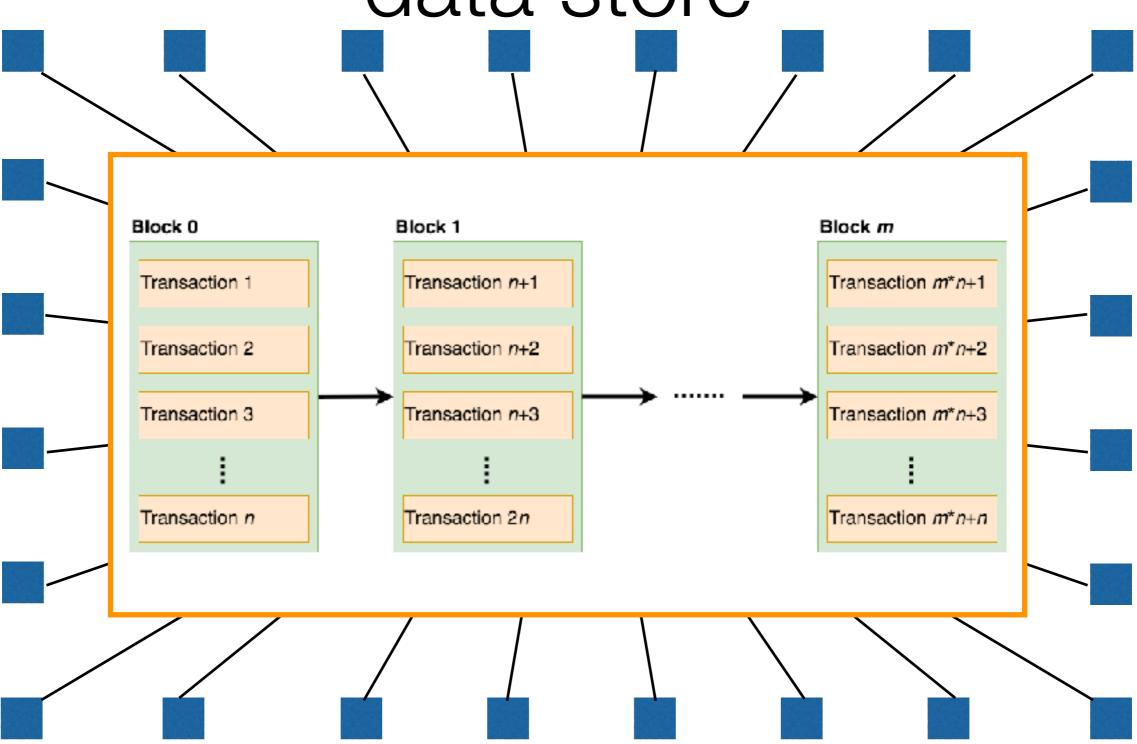
About you...

What do you want to get out of today?



Blockchain 101

Decentralised, immutable data store



Blockchain technologies







2013



2015+

2008

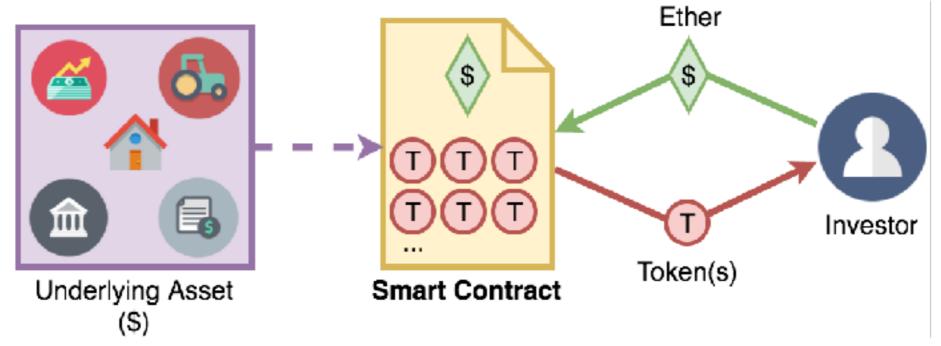


ripple

2014



Distributed Ledger Technology



- Smart Contracts
- Public and private
- Consensus mechanisms



Use cases

- Smart contracts
- Sharing inter-organisational data
- Digital asset registry
- Identity management
- IoT device data



Considerations

- Public or private?
- Wallet security
- Cross chain interoperability
- Blockchain skills shortage



Practicalities

- Rapidly changing
- Limited scalability
- Network stability
- Immature tooling
- New architectural paradigms



Platforms













Digital Asset Holdings





Some prerequisites

Key technologies

- Cryptographic Hashing
- Consensus
 - Proof of Work
 - Proof of Authority
 - Proof of Stake
- Merkle trees
- Public key cryptography (asymmetric cryptography)
- Digital signatures



Cryptographic Hash

- One way function (cannot decipher input)
- Maps arbitrary input to fixed size output (the message digest) Google Security Blog
- Avoid MD5 & SHA1!

The latest news and insights from Google on security and safety on the Internet



Security

'First ever' SHA-1 hash collision calculated. All it took were five clever brains... and 6,610 years of processor time

Tired old algo underpinning online security must die

now

By John Leyden, Thomas Claburn and Chris Williams 23 Feb 2017 at 18:33

SHARE T

Announcing the first SHA1 collision

February 23, 2017



Hashing in Ethereum

- KECCAK-256 used in Ethereum (modified SHA3)
 - 32 byte hash
 - See <u>org.web3j.crypto.HashTest</u>



Proof of Work

- Miners continually trying to verify blocks for the blockchain
 - 5 ether reward for each solution
- Based on Cryptographic hash function

```
hash(<block>) => a7ffc6f8bf1ed76651c14756a061d662f580ff4de43b49fa82d80a4b80f8434a
```

- Miners applying hash function millions (mega) of times/sec = MH/s
- Single GPU generates 5-30 MH/s
- CPU ~ 0.25 MH/s



Ethash Algorithm

- Ethash Proof of Work algorithm (formerly Dagger Hashimoto)
 - SHA3-256 variant hashing function
 - Memory-hard computation
 - Memory-easy validation
 - Can't use ASICs (Application Specific Integrated Circuits)
 - Uses 4GB directed acyclic graph file (DAG) regenerated every 30000 blocks by miner



Proof of Work Difficulty

- Hashing blocks
 - Difficulty dynamically adjusts parameter defined originally in genesis block (one block produced every 12s)
 - Started at 0x40000000 (0.017 TH)
 - Now at 0x3205AF767000 (55 TH)
- Simplified example:

Fetches bytes from DAG + combine with block Returns SHA3-256 hash

```
nonce = random int
while hashimoto(block, nonce) * difficulty > threshold
  increment nonce
return nonce
```

Solution



Genesis Block

```
"nonce": "0x0000000000000042",
"timestamp": "0x0",
"extraData": "0x0",
                     Set to a low value in test networks
"gasLimit": "0x8000000",
"difficulty": "0x40000000",
"alloc": {
```



Proof of Authority

- Only authorities allowed to create new blocks
- Suitable for private chains
- Less computationally expensive
- Used by Kovan (Parity) and Rinkeby (Geth) testnets



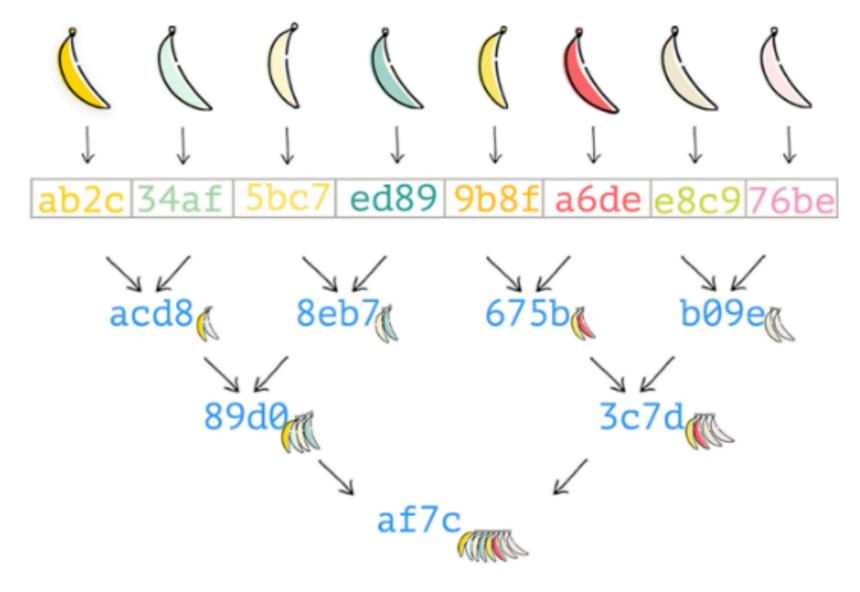
Proof of Stake

- Validators commit money to the network (their stake)
- Loose their stake if they don't abide by the rules
- Ethereum implementation Casper
- Go-live 2018?



Merkle Trees

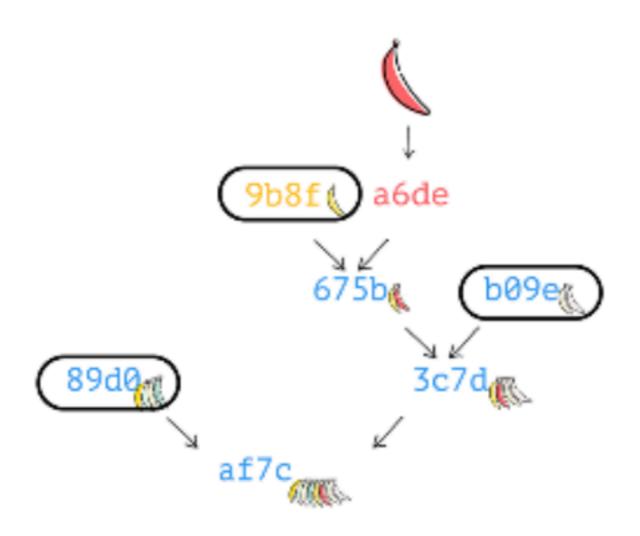
How to encode 8 bananas?





Source: http://bit.ly/2g5lmOM

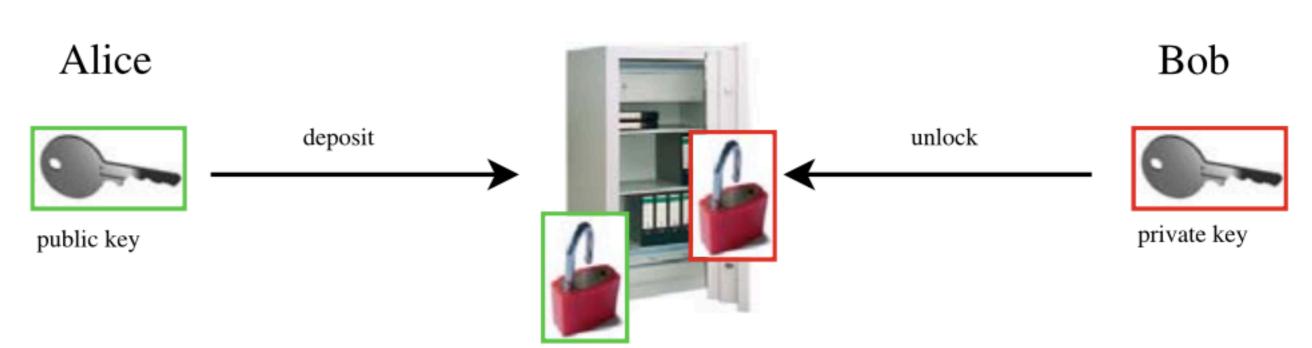
Merkle Tree (Banana) Verification





Public Key Cryptography

- Asymmetric cryptography (2 keys)
 - Bob publishes public key
 - Alice encrypts with public key
- Only Bob can decrypt via private key



Digital Signatures

Basic Digital Signature Protocol Alice

Bob

 $k_{pub,B}$

generate $k_{pr,B}$, $k_{pub,B}$

publish public key

sign message:

$$s = \operatorname{sig}_{k_{pr}}(x)$$

send message + signature

 \leftarrow (x,s)

verify signature:

$$\operatorname{ver}_{k_{pr,B}}(x,s) = \operatorname{true/false}$$





Ethereum

- The world computer
- Turing-complete virtual machine
- Public blockchain (mainnet & testnets)



Ether

- The fuel of the Ethereum blockchain
- Pay miners to process transactions
- Market capitalisation ~\$38bn USD (Bitcoin ~\$71bn)
- Associated with an address + wallet file
 0x19e03255f667bdfd50a32722df860b1eeaf4d635

```
String hash = Hash.sha3(publicKeyNoPrefix);
return hash.substring(hash.length() - ADDRESS_LENGTH_IN_HEX); // right most 160 bits
```

See <u>org.web3j.crypto.Keys</u>



1 Ether = \$300 USD

Ethereum (ETH) Price





\$300.59 \(\nu-0.24\)

Today's Open Today's High Today's Low

\$301.31 \$303.34 \$295.79 Change Market Cap Supply ▼ \$-0.73 \$28.53B 94,906,058



Obtaining Ether

- Buy it
 - Find someone
 - Coinbase
 - BTC Markets
- Mine it
 - mainnet => requires dedicated GPUs
 - testnet => quick using your CPU, via a faucet



Smart Contracts

- Computerised contract
- Code + data that lives on the blockchain at an address
- Transactions call functions => state transition

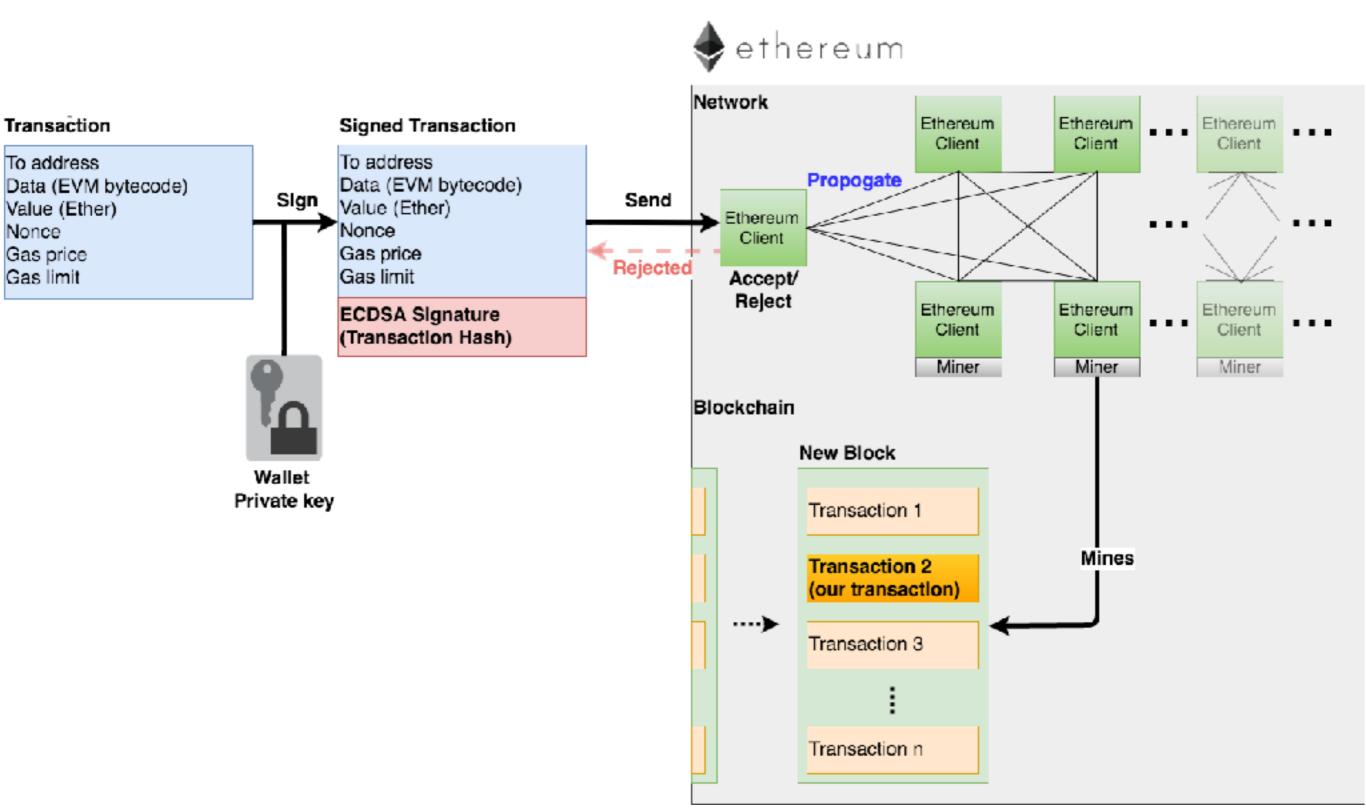


Transactions

- Transfer Ether
- Deploy a smart contract
- Call a smart contract



Transactions



Getting started with Ethereum

Free cloud clients @ https://infura.io/

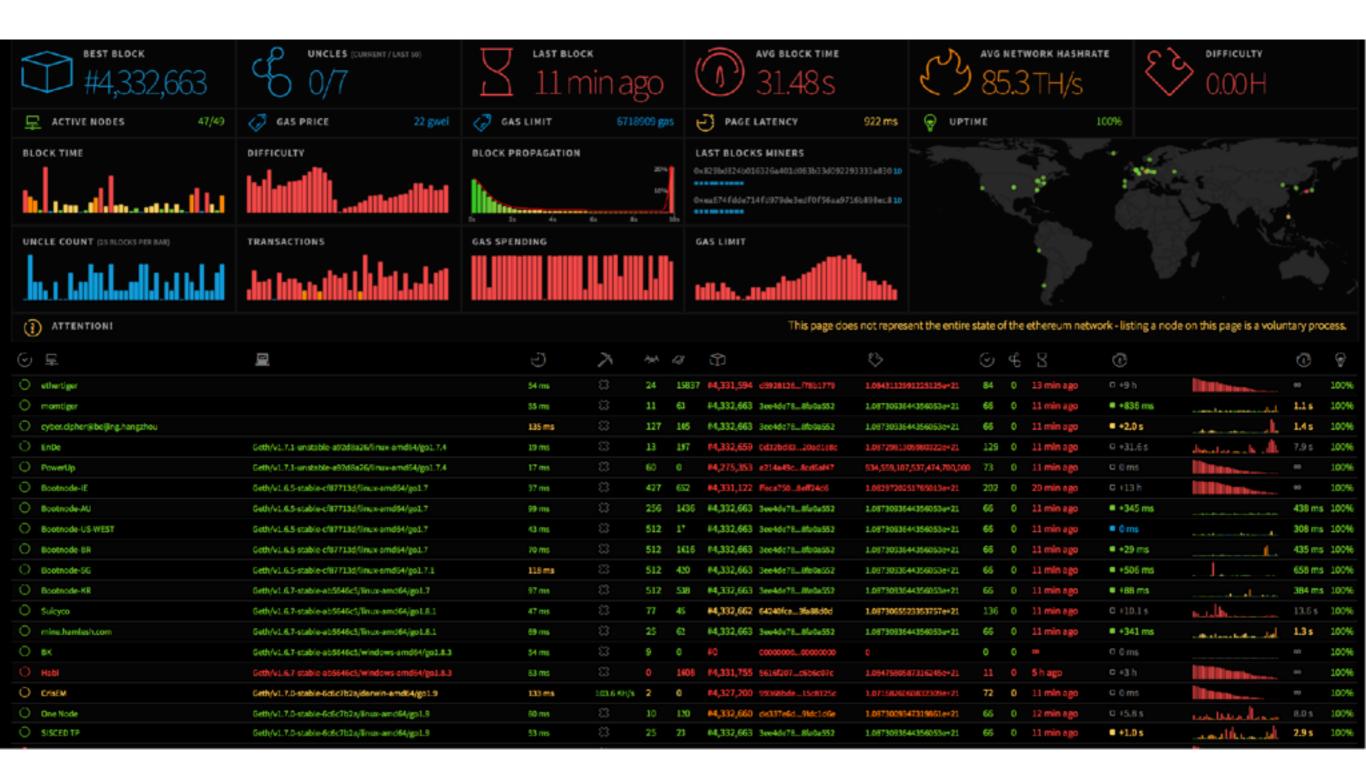
Run a local client (to generate Ether):

```
$ geth --rpcapi personal,db,eth,net,web3
--rpc -rinkeby
```

\$ parity --chain testnet



ethstats.net



etherscan.io



Search by Address / Txhash / Block / Token / Ens

GO

HOME

BLOCKCHAIN ~

ACCOUNT ~

TOKEN Y

CHART

MISC



Sponsored Link: eigen Enjin Coin - Smart Cryptocurrency for Gaming. 12M raised. Join now



LAST BLOCK 4332780 (30.11s Avg)

Hash Rate 94,063.17 GH/s **TRANSACTIONS** 63736285

Network Difficulty 2,858.59 TH

View All





Mined By DwarfPool1

72 txns in 9 secs

Block Reward 5.04893 Ether

Block 4332779 > 42 secs ago

Block 4332780

> 33 secs ago

Mined By bw.com

164 txns in 61 secs

Block Reward 5.19278 Ether

Transactions View All TX# 0XC85A608C7FCE63FE5B54406... > 33 secs ago From 0xea674fdde714fd9... To 0x0348ded7e20190... Amount 0.099042293169857136 Ether TX# 0XB79E995CC4084680DF3B008... > 33 secs ago

From 0xea674fdde714fd9... To 0xf5520b62a20f4f8...

Amount 1.000022407287363584 Ether

Testnet Monitoring

Kovan

- http://kovan-stats.parity.io/
- https://kovan.etherscan.io

Rinkeby

- http://rinkeby.io
- https://rinkeby.etherscan.io

Ropsten

• https://ropsten.etherscan.io



Exercise

- Sign up to Infura
- Start running a node locally





WEB3J.IO FEATURES OVERVIEW PROJECTS COMMUNITY





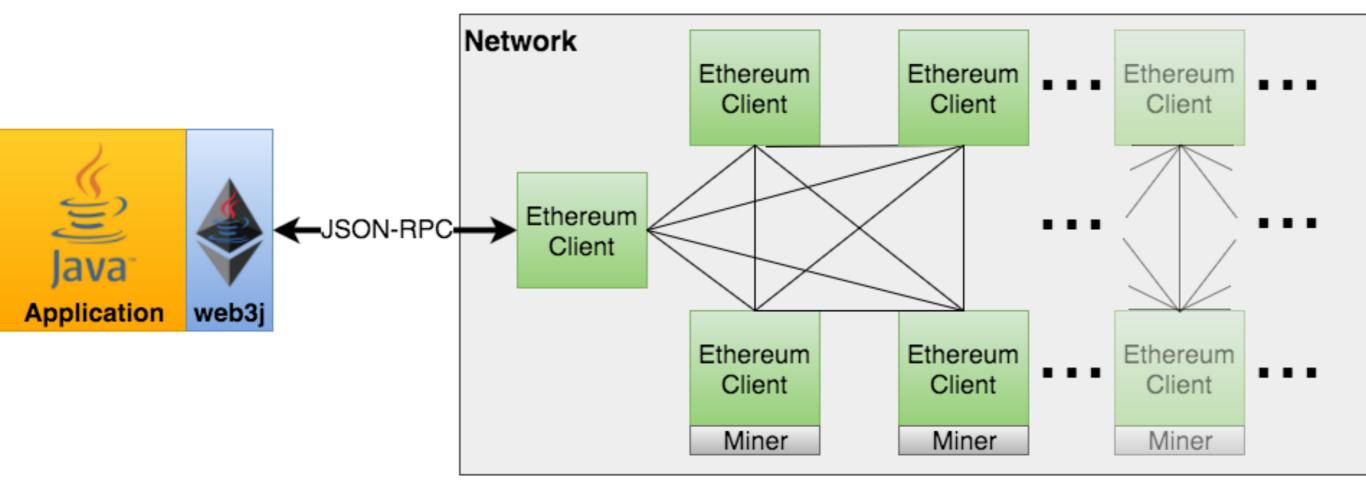






Integration with Ethereum





Integration challenges

- Smart contract application binary interface encoders/ decoders
- 256 bit numeric types
- Multiple transaction types
- Wallet management

• ...



web3j

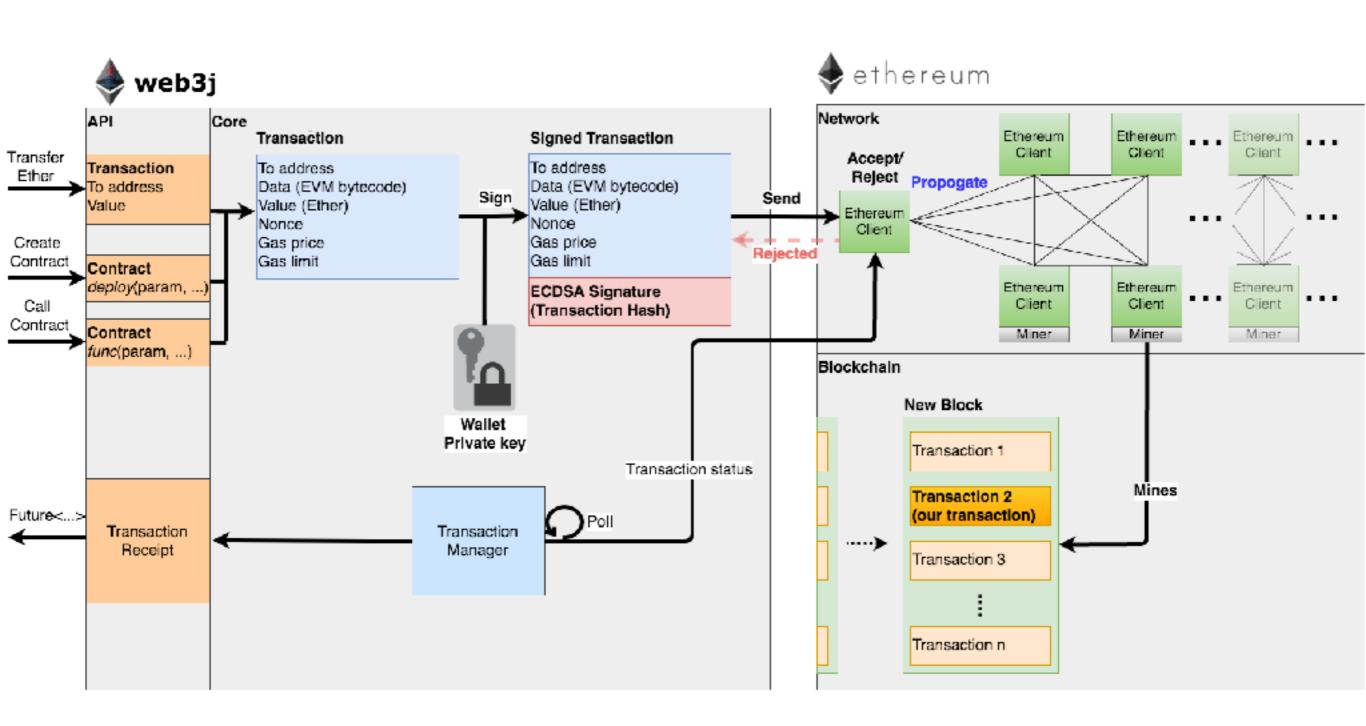
- Complete Ethereum JSON-RPC implementation
- Sync/async & RX Observable API
- Ethereum wallet support
- Smart contract wrappers
- Command line tools
- Android compatible



web3j artefacts

- Maven's Nexus & Bintray's JFrog repositories
 - Java 8: org.web3j:core
 - Android: org.web3j:core-android
- web3j releases page:
 - Command line tools: web3j-<version>.zip
- Homebrew:
 - brew tap web3j/web3j && brew install web3j

web3j transactions



Using web3j

Create client

```
Web3j web3 = Web3j.build(new HttpService());
// defaults to http://localhost:8545/
```

Call method

```
web3.<method name>([param1, ..., paramN).
[send()|sendAsync()|observable()]
```



Display client version

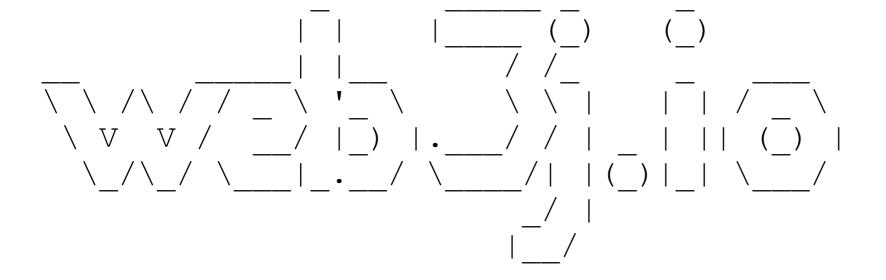
```
Web3j web3 = Web3j.build(new HttpService());
Web3ClientVersion clientVersion =
    web3.web3ClientVersion()
      .send();
System.out.println("Client version: " +
    clientVersion.getWeb3ClientVersion());
Client version: Geth/v1.7.1-stable-05101641/
darwin-amd64/go1.9.1
```

Exercise

- You can refer to https://docs.web3j.io
- Create a simple Java application to display the network version being used by web3j
- Create a new Java project and add the web3j 2.3.1 project dependency or clone https://github.com/blk-io/jaxlondon
- Connect to:
 - Your local node
 - Or, an Infura node
- Bonus see if you can figure out what the network version means and interpret the result

Create a wallet

\$ web3j wallet create

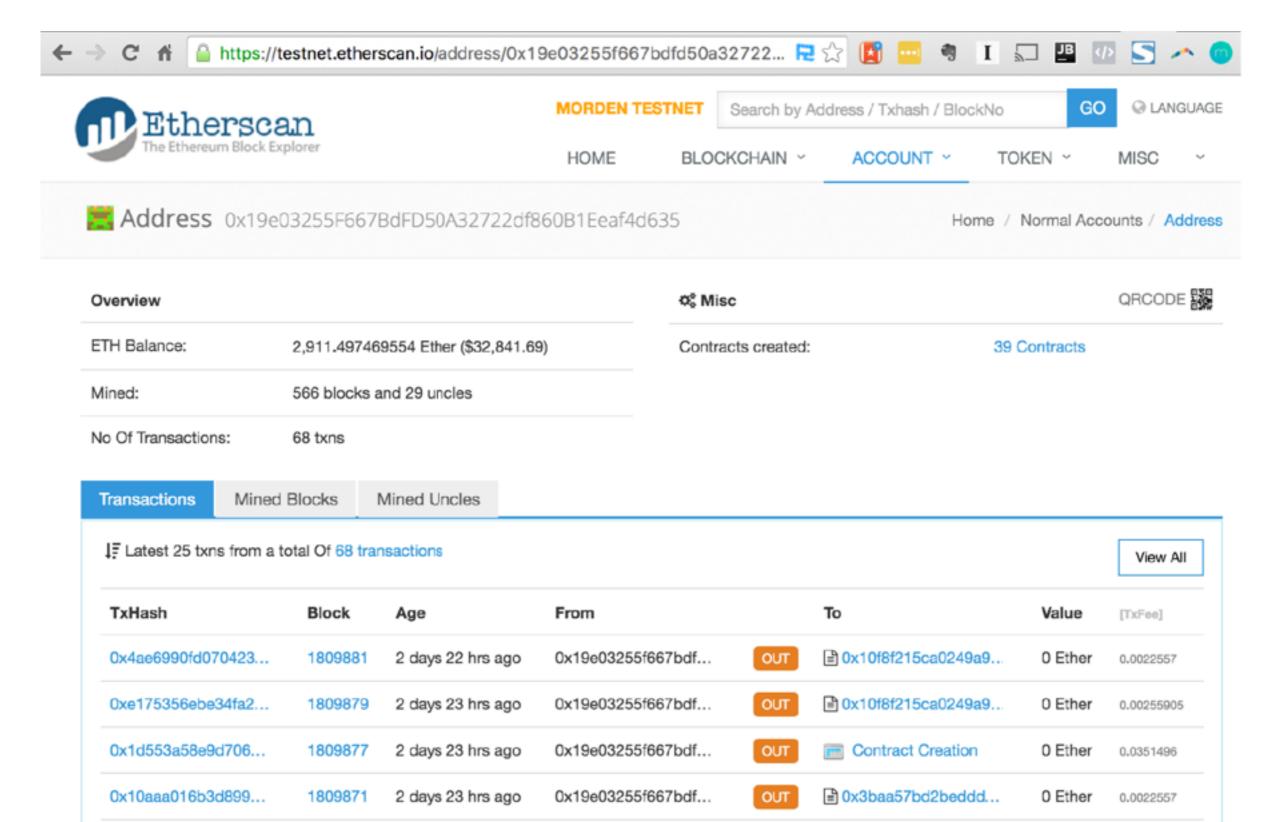


Please enter a wallet file password:
Please re-enter the password:
Please enter a destination directory location [/Users/Conor/Library/Ethereum/testnet/keystore]: ~/testnet-keystore
Wallet file UTC--2016-11-10T22-52-35.722000000Z-a929d0fe936c719c4e4d1194ae64e415c7e9e8fe.json successfully
created in: /Users/Conor/testnet-keystore

Wallet file

```
"address": "a929d0fe936c719c4e4d1194ae64e415c7e9e8fe",
   "id": "c2fbffdd-f588-43a8-9b0c-facb6fd84dfe",
   "version":3,
   "crypto":{
      "cipher": "aes-128-ctr",
"ciphertext": "27be0c93939fc8262977c4454a6b7c261c931dfd8c030b2d3e60ef76f99bfdc6"
      "cipherparams": {
         "iv": "5aa4fdc64eef6bd82621c6036a323c41"
      "kdf": "scrypt",
      "kdfparams":{
         "dklen":32,
         "n":262144,
         "p":1,
         "r":8,
"salt": "6ebc76f30ee21c9a05f907a1ad1df7cca06dd594cf6c537c5e6c79fa88c9b9d1"
      "mac": "178eace46da9acbf259e94141fbcb7d3d43041e2ec546cd4fe24958e55a49446"
```

View transactions



Sending Ether

```
Web3j web3 = Web3j.build(new HttpService());
Credentials credentials = WalletUtils.loadCredentials(
    "password", "/path/to/walletfile");
TransactionReceipt transactionReceipt =
    Transfer.sendFundsAsync(
        web3,
        credentials, "0x<to address>",
        BigDecimal.valueOf(0.2),
        Convert.Unit.ETHER) .get();
System.out.println("Funds transfer completed..." + ...);
```

Funds transfer completed, transaction hash: 0x16e41aa9d97d1c3374a4cb9599febdb24d4d5648b607c99e01a8 e79e3eab2c34, block number: 1840479



MORDEN TESTNET

HOME

BLOC

Transaction 0x16e41aa9d97d1c3374a4cb9599febdb24d4d5648b607c99e01a8e79e3eab2c34

Overview

Transaction Information

TxHash: 0x16e41aa9d97d1c3374a4cb9599febdb24d4d5648b607c99e01a8e79e3eab2c34

Block Height: 1840479 (1318 block confirmations)

TimeStamp: 12 hrs 38 mins ago (Nov-06-2016 09:54:34 PM +UTC)

From: 0x19e03255f667bdfd50a32722df860b1eeaf4d635

To: 0x9c98e381edc5fe1ac514935f3cc3edaa764cf004

Value: 0.2 Ether (\$2.17)

Gas: 2000000

Gas Price: 0.00000005 Ether

Gas Used By Transaction: 21000

Actual Tx Cost/Fee: 0.00105 Ether (\$0.01)

Cumulative Gas Used: 21000

Nonce: 1048657

Input Data:

0x

Block #1840479

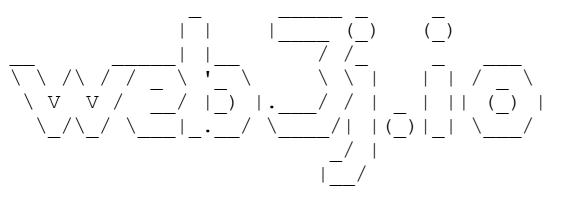
First Prev Page 3 of 3 Next Last

A total of 59 transactions found

TxHash	Block	Age	From		То	Value ‡	[TxFee]
0xb82b28aa84ae9cc	1840479	1 day 1 hr ago	0x4d6bb4ed029b33	•	0x0d31cd433711f3f	2.07602264 Ether	0.00042
0x0e027376c2b9805	1840479	1 day 1 hr ago	0x4d6bb4ed029b33	•	0x0d31cd433711f3f	2.23921522 Ether	0.00042
0x49fa39f065c2fb67	1840479	1 day 1 hr ago	0x4d6bb4ed029b33	•	0x0d31cd433711f3f	7.99908632 Ether	0.00042
0xbf4c7634884a130	1840479	1 day 1 hr ago	0xb45b1f6c9b5baf7	•	■ 0x0731729bb66243	0 Ether	0.00219832
① 0xa08ff139de28a40c	1840479	1 day 1 hr ago	0xf677878cddfeaf74	•	0xbec0ff6a41436e93	0 Ether	0.005
0x3067e3ba360d2f6	1840479	1 day 1 hr ago	0xf677878cddfeaf74	•	0xa9d65d777bfa927	0 Ether	0.00187942
① 0x0661a82a8ff78d0	1840479	1 day 1 hr ago	0xf677878cddfeaf74	•	0x2c1659253481be8	0 Ether	0.005
0x6df8129025bdb1e	1840479	1 day 1 hr ago	0x7804eb181e45082	•	■ 0x2afa3528a226640	0.01 Ether	0.00069822
0x16e41aa9d97d1c3	1840479	1 day 1 hr ago	0x19e03255f667bdf	•	0x9c98e381edc5fe1	0.2 Ether	0.00105

Sending via the command line

\$ web3j wallet send ~/.ethereum/keystore/<walletfile> 0x<destination address>



Please enter your existing wallet file password:

Wallet for address 0x<source address> loaded

Please confirm address of running Ethereum client you wish to send the transfer request to [http://localhost:8545/]: https://mainnet.infura.io/<infura token>

Connected successfully to client: Parity//v1.4.4-beta-a68d52c-20161118/x86_64-linux-gnu/rustc1.13.0

What amound would you like to transfer (please enter a numeric value): 10

Please specify the unit (ether, wei, ...) [ether]: ether

Funds have been successfully transferred from 0x<source address> to 0x<destination address>

Transaction hash: 0x<tx hash> Mined block number: 2673468



Faucets

- Request free Ether for testnets
- Geth (Rinkeby)
 - Crypto Faucet at https://www.rinkeby.io/
 - Provide Gist with wallet address
- Parity (Kovan)
 - https://gitter.im/kovan-testnet/faucet
 - State wallet address



Exercise

- Install web3j command line tools
- Generate a wallet file
- Request some ether from a Rinkeby or Kovan faucet



Smart Contracts

Ethereum Smart Contracts

- Usually written in Solidity
- Statically typed high level language
- Compiled to Ethereum Virtual Machine (EVM) byte code
- Create Java wrappers with web3j



Smart Contract Compilation

Compile

```
$ solc Greeter.sol --bin --abi --
optimize -o build/
```

- Generates
 - Application Binary Interface (ABI) file
 - EVM bytecode (binary) file



Greeter.sol

```
contract mortal {
    address owner;
    function mortal() { owner = msg.sender; }
    function kill() { if (msg.sender == owner) suicide(owner); }
contract greeter is mortal {
    string greeting;
    // constructor
    function greeter(string greeting) public {
        greeting = greeting;
    // getter
    function greet() constant returns (string) {
        return greeting;
```



Greeter.abi

```
"constant": true,
"inputs": [
"name": "greet",
"outputs": [
    "name": "",
    "type": "string"
"payable": false,
"type": "function"
"inputs": [
    "name": " greeting",
    "type": "string"
"type": "constructor"
```

Greeter.bin

6060604052341561000c57fe5b6040516102f03803806102f0833981016040528051015b5b6000 8054600160a060020a03191633600160a060020a03161790555b8051610053906001906020840 19061005b565b505b506100fb565b828054600181600116156101000203166002900490600052 602060002090601f016020900481019282601f1061009c57805160ff19168380011785556100c95 65b828001600101855582156100c9579182015b828111156100c9578251825591602001919060 0101906100ae565b5b5b506100d69291506100da565b5090565b6100f891905b808211156100d6 57600081556001016100e0565b5090565b90565b6101e68061010a6000396000f300606060405 263fffffff60e060020a60003504166341c0e1b5811461002c578063cfae32171461003e575bfe5b3 41561003457fe5b61003c6100ce565b005b341561004657fe5b61004e610110565b6040805160 20808252835181830152835191928392908301918501908083838215610094575b80518252602 083111561009457601f199092019160209182019101610074565b505050905090810190601f168 0156100c05780820380516001836020036101000a031916815260200191505b50925050506040 5180910390f35b6000543373fffffffffffffffffffffffffffffffff9081169116141561010d5760005473fffffffff 100026000190190941693909304601f8101849004840282018401909252818152929183018282 801561019d5780601f106101725761010080835404028352916020019161019d565b820191906 000526020600020905b81548152906001019060200180831161018057829003601f168201915b 505050505090505b90565b604080516020810190915260008152905600a165627a7a723058201 41d86fec5655546a8ea51f05c2df449092e6e94a88e09d4214fdf5836d7b56e0029

Smart Contract Wrappers

Generate wrappers

```
$ web3j solidity generate build/
greeter.bin build/greeter.abi -p
org.web3j.example.generated -o src/main/
java/
```



Greeter.java

```
public final class Greeter extends Contract {
    private static final String BINARY = "6060604052604....";
   public Future<Utf8String> greet() {
        Function function = new Function<Utf8String>("greet",
                Arrays.<Type>asList(),
                Arrays.<TypeReference<Utf8String>>asList(new
TypeReference<Utf8String>() {}));
        return executeCallSingleValueReturnAsync(function);
   public static Future<Greeter> deploy(Web3j web3j, Credentials
credentials, BigInteger gasPrice, BigInteger gasLimit, BigInteger
initialValue, Utf8String greeting) {
        String encodedConstructor =
FunctionEncoder.encodeConstructor(Arrays.<Type>asList( greeting));
        return deployAsync (Greeter.class, web3j, credentials,
gasPrice, gasLimit, BINARY, encodedConstructor, initialValue);
```

Hello Blockchain World!

```
Web3j web3 = Web3j.build(new HttpService());
Credentials credentials =
    WalletUtils.loadCredentials(
        "my password",
        "/path/to/walletfile");
Greeter contract = Greeter.deploy(
    web3, credentials, BigInteger.ZERO,
    new Utf8String("Hello blockchain world!"))
    .get();
Utf8String greeting = contract.greet().get();
System.out.println(greeting.getTypeAsString());
```

Hello blockchain world!

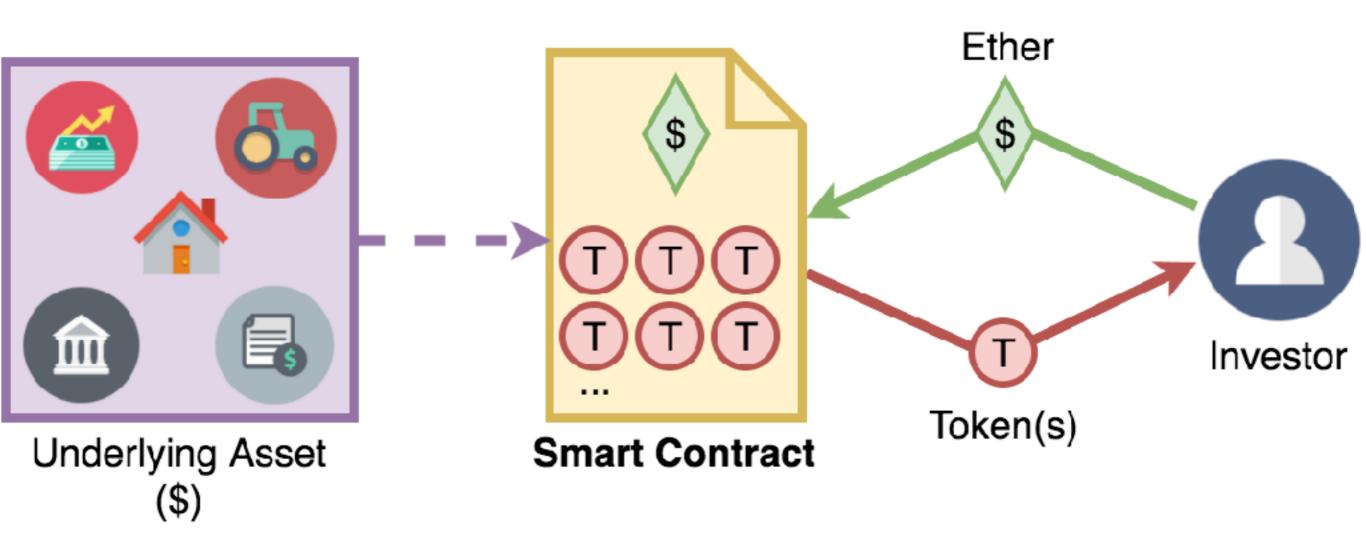


Exercise

- Install Solidity
- Add the Greeter Solidity source code to your project
 - https://github.com/web3j/web3j/blob/master/codegen/ src/test/resources/solidity/greeter/Greeter.sol
- Modify the Greeter to add a setter method
- Deploy & run the Greeter contract!



Smarter Contracts





Smarter Contracts

- Asset tokenisation
- Hold Ether
- EIP-20 smart contract token standard
- See web3j examples



Events

- Also known as logs
- Write data to the Ethereum blockchain as part of a transaction
- Defined in Solidity as:

```
event Name (paramType param1, ..., )
```

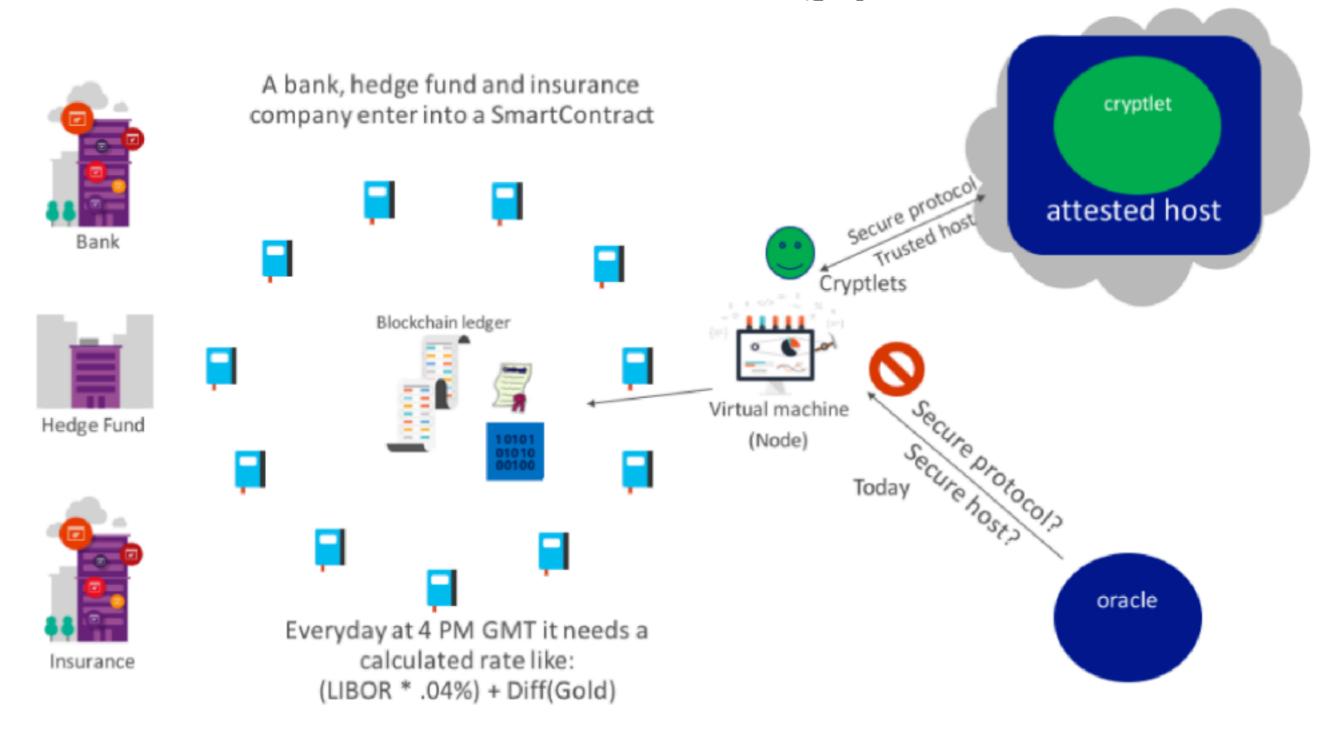
- Up to 3 indexed events (for searching)
- 32 byte size limit per parameter
- Arrays (including string & bytes) only hash is available



Initial Coin Offerings

		Token Information	Price	%Change	MarketCap
1.	89	OmiseGO OmiseGO (OMG) is a public Ethereum-based financial technology for use in mainstream digital wallets	\$8.9581 0.00187579 Btc 0.029858 Eth	▲ 13.08%	\$880,687,959
2.		Qtum Build Decentralized Applications that Simply Work Executable on mobile devices, compatible with major existing blockchain ecosystem	\$12.0947 0.00253258 Btc 0.040313 Eth	▲0.19%	\$713,587,300
3.		MKR - Maker Maker is a Decentralized Autonomous Organization that creates and insures the dai stablecoin on the Ethereum blockchain	\$246.2621 0.0516209397 Bto 0.820819 Eth		\$246,262,103
4.	•	EOS Infrastructure for Decentralized Applications	\$0.5641 0.00011811 Bto 0.001880 Eth	▼-0.56%	\$224,859,249
5.	∞	TenXPay TenX connects your blockchain assets for everyday use. TenX's debit card and banking licence will allow us to be a hub for the blockchain ecosystem to connect for real-world use cases.	\$2.1179 0.00044347 Btc 0.007059 Eth	- 7.71%	\$221,658,002
6.	\bigotimes	REP - Augur Augur combines the magic of prediction markets with the power of a decentralized network to create a stunningly accurate forecasting tool	\$18.6347 0.00390203 Bts 0.062112 Eth	- 2.00%	\$204,981,700
7.	80	GOLEM (GNT) Golem is going to create the first decentralized global market for computing power	\$0.2394 0.00005013 Btc 0.000798 Eth	- 20.30%	\$199,435,358

Oracles & Cryplets



ERC20 Token Standard

- The standard that is driving ICOs (Initial Coin Offerings)
- ERC20 is the Ethereum standard for working with tokens (coins) in smart contracts



ERC20 Functions

Define token

- Name, Symbol
- Total supply

Manage

- transfer by owner or on behalf of
- approve delegated transfer on behalf of

Observe

- get transfer allowance
- get balance



ERC20 interface

```
contract ERC20 is ERC20Basic {
 function allowance(
    address owner, address spender) public constant returns
(uint256);
  function transferFrom(
    address from, address to, uint256 value) public returns (bool);
  function approve(
    address spender, uint256 value) public returns (bool);
  event Approval(
    address indexed owner, address indexed spender, uint256 value);
```



Open Zepplin

- Smart contract frameworks
- Industry best practices
 - Security patterns
 - Modular
- Auditors of ICO contracts (~\$1.5bn of crypto)



Contract libraries

- Available from https://github.com/OpenZeppelin/zeppelin/zeppelin-solidity/tree/master/contracts
- Maths libraries
- Crowdsales
- Tokens
- Payments



ERC20 in web3j

web3j provides ERC20 integration test

https://github.com/web3j/web3j/blob/master/integration-tests/src/test/java/org/web3j/protocol/scenarios/ HumanStandardTokenGeneratedIT.java

Based on ConsenSys ERC20 implementation

https://github.com/ConsenSys/Tokens



Exercise

- Create an ERC20 smart contract wrapper
- Reference implementations to use:
 - ConsenSys
 - Open Zepplin



RxJava in web3j

web3j + RxJava

- Reactive-functional API
- Observables for all Ethereum client methods

```
Web3j web3 = Web3j.build(new HttpService()); //
defaults to http://localhost:8545/
web3j.web3ClientVersion().observable().subscribe(
x -> {
    System.out.println(x.getWeb3ClientVersion());
});
```

Processing all new blocks

```
Web3j web3 = Web3j.build(new HttpService());
Subscription subscription =
    web3j.blockObservable(false)
        .subscribe(block -> {
            System.out.println(
                "Sweet, block number " +
                block.getBlock().getNumber() +
                " has just been created");
        }, Throwable::printStackTrace);
TimeUnit.MINUTES.sleep(2);
subscription.unsubscribe();
```

Replay transactions



Replay all + future



Replay Performance

941667 blocks on Ropsten (14th June 2017):

- Blocks excluding transactions in 7m22s.
- Blocks including transactions in 41m16s

(2013 Macbook Pro)



Event callbacks

Process events in smart contracts



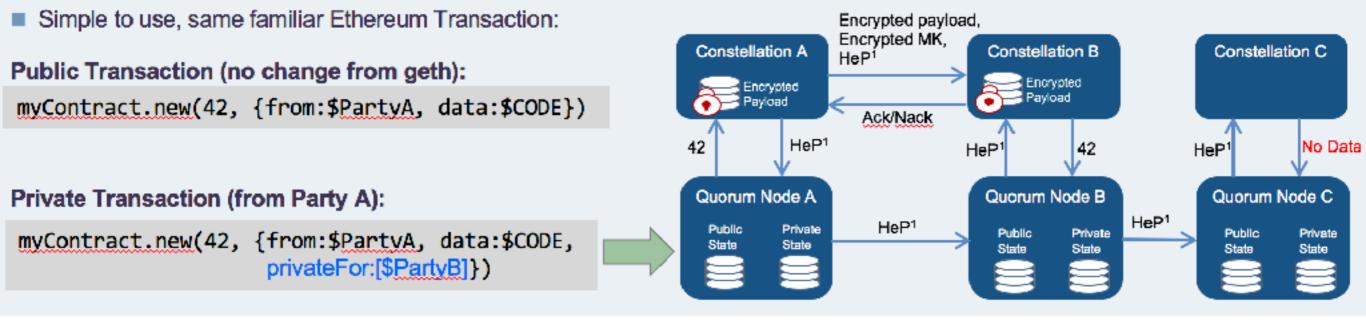


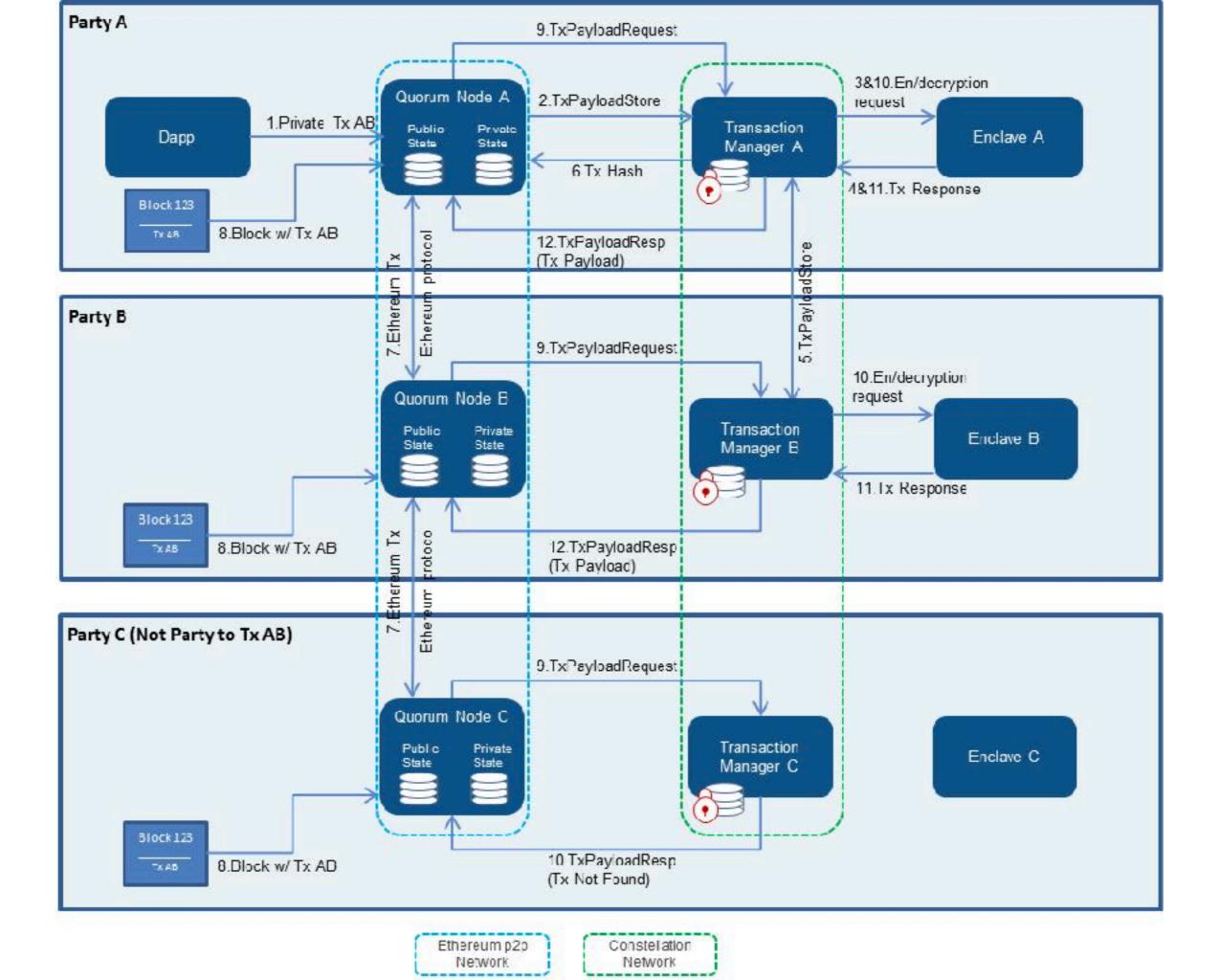
Quorum

- JP Morgan's fork of Ethereum
- Made public November 2016
- Private blockchain technology
- Adds transaction privacy
- RAFT consensus (thousands tx/sec)
- Reference client for Enterprise Ethereum Alliance



Integration with Quorum





web3j-quorum: Java integration library for Quorum

web3j-quorum is an extension to web3j providing support for JP Morgan's Quorum API.

web3j is a lightweight, reactive, type safe Java library for integrating with clients (nodes) on distributed ledger or blockchain networks.

For further information on web3j, please refer to the main project page and the documentation at Read the Docs.

Features

- Support for Quorum's private transactions
- QuorumChain API implementation
- Works out the box with web3j's smart contract wrappers

Getting started

Add the relevant dependency to your project:

Maven

Java 8:

```
<dependency>
  <groupId>org.web3j</groupId>
  <artifactId>quorum</artifactId>
  <version>0.6.0</version>
  </dependency>
```

Hello Quorum World!

```
String fromAddress = "0x<from-address>";
List<String> privateFor = Arrays.asList("<enclave-key>", ..);
Quorum quorum = Quorum.build(
        new HttpService("http://localhost:22001"));
ClientTransactionManager transactionManager =
        new ClientTransactionManager(
                quorum, fromAddress, privateFor);
Greeter contract = Greeter.deploy(
        quorum, transactionManager,
        BigInteger.ZERO, BigInteger.ZERO, BigInteger.ZERO,
        new Utf8String("Hello Quorum world!")).get();
Utf8String greeting = contract.greet().get();
System.out.println(greeting.getTypeAsString());
```



Exercises

- Run up the Quorum 7 node VM example
- Setup

```
git clone https://github.com/jpmorganchase/quorum-examples.git
cd quorum-examples
vagrant up
vagrant ssh
```

• Run

```
cd examples/7nodes/
./raft-init.sh
./raft-start.sh
```



Exercises ctd.

- Adapt Greeter example to use transaction privacy with Quorum
- Use web3j-quorum 0.6.0
- Use privateFor value of key 7
 - keys/tm7.pub:

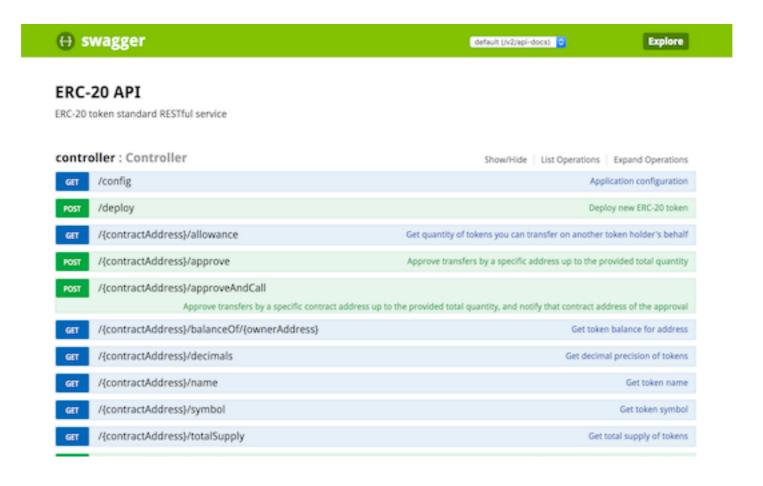
```
<dependency>
  <groupId>org.web3j</groupId>
   <artifactId>quorum</artifactId>
   <version>0.6.0</version>
</dependency>
```

ROAZBWtSacxXQrOe3FGAqJDyJjFePR5ce4TSIzmJ0Bc=



Building RESTful services

- Simple Spring Boot application running on Quorum
- Provides RESTful API for managing ERC tokens





Exercise

- Clone https://github.com/blk-io/erc20-rest-service.git
- Run a few service instances to demonstrate transaction privacy
- Docker images are available



web3j



Hacking on web3j

- git clone https://github.com/web3j/web3j.git
- Run integration tests
 - HumanStandardTokenIT
- Newbie issues labelled with help wanted
 - Increment field ids on JSON-RPC requests
 - Cache network id in RawTransactionManager
- Contribute to documentation



Closing thoughts

What did you get out of today?



Where to go from here

- Reddit
 - https://www.reddit.com/r/ethereum/
- Ethereum blog
 - https://blog.ethereum.org/
- Ethereum Improvement Proposals
 - https://github.com/ethereum/EIPs
- Enterprise Ethereum Alliance
 - https://entethalliance.org/

