Haaga-Helia
University of Applied Sciences

Building privacy-sensitive applications with Ethereum and smart contracts

Mari Luukkainen

2 September 2018

Abstract

2.9.2018

Author Mari Luukkainen	
Degree programme Bachelor’s degree in business information technology	
Report/thesis title Building privacy-sensitive applications with Ethereum and smart contracts	**Number of pages and appendix pages** 25

Privacy becomes more important in the global economy which is turning to be more online. Online economy is vulnerable to digital era threads but meanwhile new technologies like blockchain and Ethereum based smart contracts can offer solution to make the digital future even more private and secure.

This thesis will go through the history of blockchain applications and privacy concepts, and Ethereum’s potential to build solutions that answer to complex privacy issues of online applications.

To widen aspects to more concrete points of view internationally operating blockchain entrepreneur and a software engineer were interviewed for hand on experiences of the scene.

Keywords
Blockchain, cryptocurrency, smart contract

Table of contents

1. Introduction

Today, the world’s economy, governments, organizations and people are increasingly reliant on emerging and innovative technologies. According to McKinsey Global Institute's 2013 analysis, the world’s global market will experience significant change by 2025 driven by disruptive technologies. Consequently, their cumulative potential impact is valued at 14 to 33 trillion USD (US dollars). This technological shift is widely referred to as the fourth industrial revolution, with some of the notable technologies being: energy storage, biotechnology, quantum computing, materials science, the IoT (Internet of Things), nanotechnology, autonomous vehicles, robotics, 3-D printing, cloud computing, Big Data, and AI (Artificial Intelligence). Incidentally, a relatively new technology, Blockchain, which is not listed in the disruptive technologies, is emerging as glue that binds and converges these disruptive technologies.

Blockchain technology describes a software mechanism that provides for creation of trusted assets and related transactions on a distributed system devoid of a centralized trust authority (Lovells, 2017, 7). Coined by Satoshi Nakamoto in 2008 and developed by a core group of pioneers, the technology was implemented as a vital component in Bitcoin’s cryptocurrency venture. Initially referred to as a “block” and “chain” technology, the concept grew to denote new applications developed within a distributed database platform. Ultimately, is has now been recognized as a foundation technology capable of establishing socioeconomic infrastructure systems across various fields, such as, manufacturing, healthcare, government, financial services sector, ICT sector, among others. For instance, it has been applied in development of money transactions applications; DAO (Decentralized Autonomous Organization); privacy protection and decentralized data (such as use of secure identities and smart contracts); and possible future applications in banking, IoT (Internet of Things), online-voting, human intelligence and storage of mind files, SCM (Supply Chain Management), and stock trading.

This paper seeks to explore the use of Blockchain technology and related technologies (smart contracts) within the realm of privacy protection in the development of privacy-sensitive applications. This will be based on the following areas: a background of blockchain technology and related concepts, with a focus of early application options (such as bitcoin, metacoins, namecoin, and colored coins) and new blockchain trends and smart contracts; an analysis of the implementation of authentication and authorization processes in privacy protection applications; an in-depth discussion of Ethereum, which will include Ethereum accounts, transactions and messages, Ethereum state transition function, code execution, smart contracts, and decentralized autonomous organization;

and an exploration of possible applications of blockchain technology and smart contracts in token systems, trusted payments systems, identity and reputation systems, decentralized file storage, decentralized systems access, secure data provenance management, automotive security and privacy systems, and trade finance systems. Finally, a conclusion will be derived as to the viability of deploying blockchain technology with smart contracts in privacy-sensitive applications.

2. Background

Regardless of blockchain technology's short history, it is evolving rapidly. Sophistication of technology, usability, security and growing range of use cases of blockchain is enabling technology companies to offer more anonymized and data protected services to users in global markets.

2.1. History of Blockchain and related concepts

A blockchain defines a distributed database that captures records popularly referred to as a public ledger that stores, and shares executed digital events / transactions across all participating entities. For every transaction recorded, a verification process (via a majority consensus) among the system participants is performed. In addition, once an entry has been into the public ledger, it cannot be arbitrarily erased (Swan 2015, 5). In essence, a blockchain is a representation of all verifiable transactions recorded within a network of participants. The development and deployment of this technology across a myriad of sectors remains at its nascent stage although it has mainly been successful within the financial world via decentralized digital currency (based on a peer-to-peer network).

The existence of decentralized digital currencies and related concepts such as property registries dates back a couple of decades – the 1980s and 1990's e-cash anonymous protocols, which utilized Chaumian blinding (a cryptographic primitive) and exhibited high levels of privacy (Evgeii 2017, 14). However, their dependence on an intermediary that was centralized contributed to lack of growth and wide acceptance. Consequently, b-money proposed by Wei Dai in 1998, sought to mint money through a combination of decentralized consensus and computational puzzle solutions. As it predecessors, it did not gain traction due to the lack of clarity in decentralized consensus implementation. Hal Finney, then developed a new concept in 2005 that utilized a proofs of work (reusable) system, which combined Hashcash puzzles (by Adam Back) and b-money's concepts to derive a cryptocurrency. However, as with previous attempts, it failed due to a back-end implementation of trusted computing. Ultimately, Satoshi Nakamoto sought to develop a decentralized currency in 2008 by combining a consensus algorithm, a proof of work system for tracking coin ownership, and established primitives (utilized in ownership management via public key cryptography) – what came to be known as Bitcoin.

Bitcoin, is a concept that utilizes cryptographic proof as security for the execution of an online transaction (between willing parties), as opposed to a trusted third-party. Every transaction performed is secured using digital signature. This entails the transaction initiator sending a digitally-signed message (using a private key), which the receiver opens with a public key. Successful opening of the message by the receiver using the sent public key proves ownership of cryptocurrency, hence a financial transaction can occur. Upon completion of a transaction, it is broadcast across the Bitcoin network (to all nodes) and simultaneously recorded in the public ledger upon verification. It is important to note prior to the recording of a transaction in the public ledger, it is necessary for verification to take place to assure validity. The node performing the transaction verification needs to ensure: (1) a spender of the cryptocurrency is the rightful owner (via verification of the transaction's digital signature); and (2) spender has adequate cryptocurrency within his/her account (via crosschecking of each spender's transaction against their ledger account record using the sender's public key). To ensure order in transactions recording and legitimate approval of spending, the Bitcoin system utilizes blockchain technology. It entails transaction ordering in groups (referred to as blocks) that are linearly linked (chain), in a chronological order, hence the blockchain. Each blockchain contains a previous block's hash, which it utilizes as a proof of work in acceptance into a larger block and generation of a new block by a node.

The proof of work concept has proved critical in the success of Bitcoin implementation due to: (1) it has created a simplistic and effective consensus algorithm, enabling unison agreement of Bitcoin ledger state updates across all nodes; (2) it has created a free-entry avenue into the consensus process, hence solving decision issues related to consensus influence allocation and concurrently blocking possible Sybil attacks. In addition, a single node's weight (within the consensus voting process) is proportionate (directly) its computing power. As an improvement to the proof of work concept, a complementing approach described as proof of stake has been recently proposed, where the calculation of a node's weight is directly proportional to its currency holdings within the network. While each approach has been described as having its own merits, they both provide the core foundation for cryptocurrency development.

2.2. Early Blockchain Application Options

Although, the applications potential of blockchain technology is immense, early success in its implementation has been mainly in: (1) decentralized data protection and privacy; and (2) digital cryptocurrency. In terms of digital cryptocurrency, notable solutions include: Bitcoin, Metacoins, Namecoin and Colored Coins (Crosby et al. 2015, 12). There have been newer cryptocurrencies such as Certcoin and Netcoin

2.2.1. Bitcoin

Besides the description of this cryptocurrency provided in earlier sections, it contains additional features and qualities that have enhanced its success. Firstly, its ledger system can be defined as a system that implements state transition, based on ownership status definition (across all existing bitcoins) and a transition function that computes a resultant new state by combining two inputs (a transaction and the current state). This quality enables the following elements: (1) it deters senders from spending non-existent coins through transactions blocking; (2) it deters senders from utilizing coins from other people to execute transactions; and (3) it promotes value conservation.

The second quality, mining, facilitates dynamic and valid production of transaction packages (blocks) across the node network (an implementation of the decentralized consensus concept). This entails continuous block production (approximately one in ten minutes), with each block being validated using a hash (of the previous block’s reference), a transactions list (containing prior transactions performed in the previous block), a nonce, and a time stamp. The third quality, Merkle trees, represents a scalability feature that facilitates a multi-level block-storage data structure. The Merkle tree represents a binary tree composed of a node set with related leaf nodes containing data. The cascaded node network is interlinked via a hash system, where a parent node constitutes the combined hash values of its children nodes. This concept allows for piecemeal delivery of data in blocks and based on a upward propagation of node hashes. Any illegal alterations at the bottom nodes of the merkle tree are propagated upwards to the head of the block, hence resulting in protocol registration as a totally new / different block. Consequently, it presents as an invalid proof of work, thus rejected as a legitimate transaction within the blockchain.

2.2.2. Metacoins

Metacoin represents a protocol built on top of Bitcoin’s concept, where it employs Bitcoin transactions as a means of transactions storage but applying a variant state transition function. The protocol allows for creation of discretionary cryptocurrency protocols based on advanced features (not implemented in Bitcoin) and at a low development cost, due to delegation of networking and mining complexities to Bitcoin. Its main application areas have been in name registration, financial contracts and decentralized exchanges.

2.2.3. Namecoin

Developed in 2010, Namecoin defines a name registration database that is decentralized. Through the implementation of a “first-to-file” paradigm, the platform allows account holders to register name identifiers (as opposed to pseudo random hashes) within the decentralized database (Thakur 2017, 32). This facilitates easier interaction, identification and transaction between two account holders. In addition, the first-to-file paradigm prevents account impersonation or name replication.

2.2.4. Colored Coins

Colored coins provide a protocol platform that facilitates the creation of other digital currencies. Using this protocol, one can publicly issue a new currency by assigning it a color (based on Bitcoin's UTXO), hence enabling recursive color definition (of other UTXO) similar to the initial transaction inputs (Swan 2015, 15). Consequently, it facilitates maintenance of specific-color UTXO wallets by users and enables backtracking capabilities across the blockchain to determine received UTXO colors.

2.2.5. Certcoin

This cryptocurrency is founded on a combination of Namecoin and Bitcoin concepts, which allows utilization of name dictionary data structures within a given blockchain to protect current users’ identity. This is based on the implementation of a hash table whose lookup is assured via a distributed public key. During the lookup process, a public key is generated and combined with user identity for network entry. Once entry is achieved the lookup process entails traversal of the entire blockchain while bypassing signature controls as these were already in place during name registration within the blockchain.

2.2.6. Netcoin

Netcoin represents a cryptocurrency developed in 2015 by Tewari and Nuallain that is similar to Bitcoin albeit with less complexities. Its implementation involves application of a blockchain in transaction history storage within each coin, hence creating a unique identity within the system. In addition, it allows a user to spend a coin more than once (a feature that is not present in Bitcoin). The cryptocurrency limits computing power usage by utilizing network capabilities in coin legitimacy verification (as opposed to employing proof-of-work approach). Once a coin’s blockchain length nears a particular threshold, a re-issuance of the coin (to current coin owner) is done to avoid large storage and increased bandwidth problems due to a large blockchain.

2.3. New Trends in Blockchain and Smart Contracts

Besides its success in cryptocurrencies, blockchain technology is also gaining traction in other areas such as decentralized data protection and privacy (Lovells, 2017, 14; Satyavolu and Sangamerkar 2016, 3; Crawford et al. 2016, 7; Cong and He 2018, 8). This has seen several innovations on this platform. For instance, Lazarovich (2015), in platform called invisible ink, develops a decentralized, Bitcoin-like blockchain that protects and stores sensitive data. The decentralized nature of the platform allows usage by real-time applications across different service providers. A case in point is the protection of a user's personal and private data being held by trusted third parties. Another application developed by Van Den Hoff et al. (Kosba et al. 2015) known as Hawk, employs a smart contract system to carry out transactions within publicly visible, decentralized digital currency systems. The system takes the burden of encryption of private information from the user by implementing a cryptographic protocol during system compilation.

Blockchain interaction is similar to Bitcoin with the exception of nonavailability of cryptographic private information. The system is suited for digital transactions. Zynskind et al. (2015), suggest another system that utilizes decentralized privacy system, based on Bitcoin's implementation of blockchain technology to protect personal data. This is achieved via blockchain implementation as database storage of personal data. Users input their data within the blockchain and third parties (such as authorities or organizations) that require the information can access it without storage responsibilities, via a control moderator and a myriad of storage solutions (Ghaffari 2016, 16). In essence, users are able to track whom, how and what their personal is being accessed and used for, hence taking total control over their private information.

3. Privacy Protection Concepts

The privacy protection aspect of any system is concerned with the safeguarding of provisioned resources, which may include: its applications, computing processes, network storage, network access, and data access and storage (Thakur 2017, 4-5). Indeed, access control, to these resources is critical component for successful system deployment alongside resource usage measurements, policy enforcement and system audit. Ideally, the general framework for privacy protection entails employment of Authentication and Authorization mechanisms for efficient and effective management of security, resources, network, users and data (Katz et al. 2002, 5). Authentication entails user identity verification processes, while authorization determines user rights to access a requested service or resource. In most scenarios, this is implemented based on a client-server architecture, where user interactions occur in the client application and authentication and authorization services implemented in the server. This is based on a two-step process. Step 1 involves user credential verification, carried out by the authentication service, and if successful, it is forwarded to the next step. Alternatively, an error return is displayed on the user's client application, if unsuccessful. Step 2 entails forwarding of a successful request to the authorization service for determination of user authority. If successful, a user's request is directed to a resource service, which returns the user's requested resource; else an error return is displayed. A detailed discussion of these services is presented in the subsections below.

3.1. Authentication

As earlier indicated, authentication is a real-time procedure of identifying system users prior to granting system access and resources. It enables registered system users to utilize system services based on their registration credentials. These credentials can be categorized into: ownership (who you are), such bio-metrics traits like iris pattern, and fingerprints; knowledge (what you know), such as username and related password; and possession (what you have), such as e-tokens, smart cards, key cards, or identity cards. Some of the widely used authentication techniques include: username and password combinations, bio-metrics, multi-factor, and public key infrastructure.

3.1.1. Username and password

This technique represents the most widely used means. Standard procedure entails user registration, where user data, such as phone number, email, username / official names, password etc are recorded. Once registration is deemed successful, a user gains access to system resources based on the username and accompanying password utilized in the registration. While this technique is easily accomplished, its security varies and is dependent on password characteristics and length. Despite implementation of strong and complex passwords, they can be obtained through brute force, coercion, or by guessing. In addition, use of complex passwords present a challenge in recalling, hence the likelihood of using similar password for multiple user accounts or a password manager that is susceptible to attacks.

3.1.2. Biometrics

Authentication via biometrics represents an advanced technique that harnesses physiological characteristics or a measurable behavioral attribute for user authenticity. Behavioral attributes include: gait analysis, signature recognition, keystroke dynamics, and voice recognition. Physiological characteristics could be: hand, finger, and face recognition, retina and iris pattern recognition, and fingerprints. These traits or markers are unique to each person; hence, an owner reliably proves authenticity. Furthermore, it is more secure (in comparison with other methods) as it is difficult to steal or crack. However, it is an expensive option to implement based on hardware and software specification, and its degree of accuracy is still an ongoing drawback.

3.1.3. Multi-factor

Multi-Factor authentication identifies an advanced authentication technique, which combines two credentials, such as knowledge-possession or knowledge and ownership to facilitate user authenticity. For instance, a website login may involve a combination of a password and pass-code / pass-phrase obtained from a linked hardware, for example mobile device. It represents a user-friendly technique but demands high capital outlay to deploy.

3.1.4. Public Key Infrastructure (PKI)

The Public Key Infrastructure (PKI) technique is founded on a user-generated private-public key (cryptographically), and used for authentication purposes – private key is stored by user and public key distributed to the system. It encompasses the use of the private key for proving user identity and the PKI for use with security protocols such as SET (Secure Electronics Transaction) and SSL / TLS (Secure Socket Layer) for data authentication and confidentiality, non-repudiation, and data integrity. Its mechanism, (of generating cryptographic public-private key) ensures a better security, in addition to being difficult to crack, especially in comparison to the username-password technique. However, it requires some imperative user knowledge on basic key-pair generation and distribution, which may be a cumbersome exercise to the user. Likewise, during the deployment process the likelihood of targeted attacks by hackers to gain the private key, or even crack, are high.

3.2. Authorization

Authorization describes the determination process of a user's authority status to enable requested resource access or execute certain commands on the system. Its implementation is tightly coupled with the user authentication procedure as its succeeds an authentication order. This technique is critical in determination of user access rights as systems may consist of two or more categories of users. There are a number of authorization systems and frameworks, with most widely adopted being XACML and Oauth 2, respectively (). An in-depth analysis of the two is presented in the next section.

3.3. XACML

XACML, eXtensible Access Control Markup Language, represents a mainstream authorization system standard. It outlines a declarative fine-grained access control policy based on attribute language. It consists of the following components: PAP (Policy Administration Point), PIP (Policy Information Point), PEP (Policy Enforcement Point), and PDP (Policy Decision Point). PAP develops and sustains policies based on a common central repository, while PDP stores and analyses user requests' policy information. PEP authorizes decisions derived from stored policies within the universal repository, and PIP supplies other attribute values (such as resources or actions). The components facilitate a request-response based protocol interaction with each other and are deployed at the server level. This interaction is guided by 3 elements: the PolicySet, which consists group of policies; the Policy, which is a combination of rules by PAP; and the Rule, which represents a policy unit. PAP is also responsible for specific-role assignment during registration and the resource-request process. Consequently, the PDP checks for role-assigned during a user request against the repository. Upon successful retrieval, a user is supplied with an attribute-added response and thus able to access appropriate resources, else resource-access is denied.

3.4. OAuth 2.0

OAuth 2.0 identifies an authorization framework, which enables limited resource access by a third-party application in the interest of the resource holder and with valid consent (from owner). This access is mainly requested by third-party or client applications such as a mobile app or web service. It is used in roles' definition including: the authorization server, which grants the client access tokens; the resource server, which hosts the resource (protected); the resource owner, who represents a person / entity responsible for granting protected resource access; and the client, which is a resource-accessing application in the interest of the user. The access token issued by the authorization server serves as the user’s identity for a validated period (such as 12 or 24 hours). A resource owner seeking protected resource access actuates the authorization flow process. This then initiates client application of authorization (from resource owner) and on succeeding, returns the authorization grant. An access token is then returned by the authorization server, and consequently sent for validation on the resource server. A successful validation returns the requested protected resource, without client credential-approval processes by a resource owner.

4. Ethereum

Ethereum is a popular blockchain application that was developed to compliment bitcoin weaknesses, particularly its non-Turing complete characteristic and limitation of the Bitcoin script to small instructions. Indeed, the use of Bitcoin script in developing applications entailed developers forking Bitcoin’s core code-base in order to include bespoke logic in line with their use cases. The forking process is cumbersome to maintain and very time consuming; hence such challenges were addressed via Ethereum. Ethereum essentially facilitates the building of applications by programmers atop the blockchain – Ethereum blockchain. Coined by Vitalik Buterin in 2013 (Thakur 2017, 33), it proposed the writing of smart contracts (scripts) using a Turing-complete coding language and an EVM (Ethereum Virtual Machine) in the execution of transactions and smart contracts.

Essentially, Ethereum provides a substitute protocol in developing decentralized applications based on a number of concessions: its ability to handle massive decentralized applications; high suitability for rapid applications deployment; maximum security for small-scale and limited-use applications; and efficient cross-platform applications interaction. To achieve these trade-offs, Ethereum creates a foundation abstract layer, the blockchain, using an inherent programming language (that is Turing-complete), hence enabling universal use of decentralized applications and writing of smart contracts. Furthermore, it assures development of customized state transition functions, discretionary ownership rules, and transaction formats. For instance, the writing of reputation or currency systems can be done using few lines of code (less than 20) and in some cases even two code lines (a basic implementation of Namecoin). Ultimately, Ethereum capabilities surpass Bitcoin’s scripting based on the following qualities: additional strength from Turing-completeness, blockchain-state and awareness, and value-awareness.

The working of Ethereum entails several concepts, which are: Ethereum accounts; transactions and messages; the Ethereum state transition function; and code execution. Likewise, it is enforced alongside concepts such as decentralized autonomous organization and smart contracts.

4.1. Ethereum Accounts

The state in Ethereum comprises objects referred to as accounts. Every account holds a 20-byte addressing and state transitions capacity allowing for direct transmission of information and value among accounts. Separately, an Ethereum account incorporates four fields: its storage space, which is null by default; a nonce, representing a counter that ensures one-time processing of transactions; a contract code (if defined); and the real-time ether balance. Ether defines Ethereum’s prime internal crypto-fuel utilized in transactions’ fees payment.

Ethereum runs two account types: contract and externally owned. The basis for administration in a contract account is its contract code. The code activates on each receipt of a message, allowing reading and writing to its internal storage, and consequently, contracts creation or message sending. An externally owned account, on the other hand, operates without a code, enabling message sending via creation and signing-off of transactions. It is important to note that the application of contracts in Ethereum denotes autonomous agents residing within its execution environment where explicit code execution occurs upon receipt of a transaction or message prompt and have forthright control of their intrinsic value / key store (for persistent variables tracking) and ether balance.

4.2. Transactions and Messages

A transaction in Ethereum relates to a signed data package responsible for storage of messages to be relayed by an externally owned account. A typical transaction consists of: the message’s recipient; sender’s signature; the ether amount (to be transferred to the recipient from the sender); a data field (optional); a STARTGAS value, which indicates the computational steps maximum limit for transaction execution; and a GASPRICE value, specifying the requisite fee (per computational step) the sender is required to pay. The recipient, signature and ether, represent definitive cryptocurrency fields. While the data field has no designated function, a contract reliably exploits it via the virtual machine’s an opcode to access data. For instance, a contract-implementation of a blockchain-based domain registration service would require data interpretation of various fields, which would be obtained from the message data and conveniently deposit them in storage.

The two fields, STARTGAS and GASPRICE are critical in the execution of the anti-denial of service standard. Typically, during the coding process inherent events such as hostile or accidental infinite loops, and diverse computational wastage could occur, hence the need for explicit declaration of the maximum number of computational steps allowable for code execution by a transaction. In this case, the foundation computational unit is gas. Ordinarily, it costs 1 gas per computational step, although there is likelihood of higher costs in some operations due to increased amount of storage data (as a state component) or computational complexities. Additionally, a fee (of 5 gas) is charged per byte composition of transaction data. The fee system is intended to make possible attacks expensive, as the attacker has to pay for comparable resource consumption, including storage, computations, and bandwidth. Accordingly, a transaction necessitating increased consumption of network resources has a proportionate gas fee attached to the increment. In Ethereum, messages represent virtual, non-serialized objects whose existence is confined to the execution environment. It provides an avenue for communication relay between contracts. A typical message consists of: a sender, who implicitly relays the message; a message recipient; a total amount of ether transferable with message relay; a data field (optional); and a STARTGAS value. Fundamentally, messages are similar to transactions, but for the exception of its origin – it is created by a contract and no external actors are involved. The process of message creation is initiated during a contract's code execution phase via a CALL opcode, which triggers message execution. Similar to the transaction, once a message is executed, a recipient account is able to run its code. Hence, messages have the ability of having contract-to-contract relationships, as exhibited by external actors. In messages, the gas allowance represents the sum of gas consumed by a transaction and related sub-executions, which is assigned by a contract or transaction.

4.3. Ethereum State Transition Function

The state transition function in Ethereum represented by APPLY (S, TX) -> S` fulfills a number of conditions. First, it checks on the transaction formed (if it has all the correct number of values), validity of the signature, a match of the nonce with that of the sender's account, and an error return if any is false. Second, it performs transaction fee calculation (based on STARTGAS * GASPRICE formulation) and derives an address from the sender's signature. It then subtracts the appropriate fee (from the account balance of the sender) and increments the nonce (senders). In case of insufficient balance, an error is returned. Third, it initializes STARTGAS using the GAS = STARTGAS command and takes of gas quantities, on a gas / byte formula depending on the transaction's byte figures. Fourth, it transfers the value of the transaction to the receiving account, from the sender's account. In cases where a receiving account is nonexistent yet, it develops one. Additionally, it runs the contract's code if receipt account represents a contract, until either of two possible scenarios is fulfilled: the contract execution exhausts the gas, or it completes. Fifth, it reverts any executed state changes to the original state, with the exception of fees payment if: code execution exhausted the gas or sender had inadequate money. The fees are normally into a miner's account. The final condition is a reimbursement of fees to the sender (for remaining gas) and forwarding of paid fees to the miner for gas consumed.

In terms of reverting, messages operate in a similar manner to transactions. It executes based on the following conditions: when a message execution exhausts all the gas, its execution and initiated sub-executions (with the exception of the parent execution), revert. This implies that a contract Y with 3 gas is capable of calling another contract Z to execute and can only lose a maximum of 3 gas, in case it reverts. Finally, the CREATE opcode has similar execution structure to the CALL opcode, only that its execution output influences a newly created contract's code.

4.4. Code Execution

Code writing in Ethereum contracts in achieved by EVM (Ethereum Virtual Machine) code, a low-level language that uses stack-based byte-code. The code contains a set of bytes with every byte representing an operation. Ordinarily, the code execution defines an infinite loop comprising operation repetitions based on a program counter that initiates at zero and is incremented by one until code completion is attained, or a RETURN or STOP command is issued, or an error is returned. Three space types facilitate these operations, where data is stored. They are: a value / key store (i.e. a long-term storage point of the contract) that is persistent post operation computations, a dissimilar feature to memory and stack's reset abilities; memory, which defines a boundlessly expandable byte array; and stack, which represents a container for pushing and popping values based on a last-in-first-out criterion (Wood 2018, 2). Besides this, the code has unfettered access to the data, sender and value of an incoming message, block header data, and the ability to return output data as a byte array.

The EVM code's explicit execution is elementary. During the functioning of the EVM, a comprehensive tuple definition of its computational state is represented as "(block_state, transaction, message, code, memory, stack, pc, gas)". The block_state, defines a global state incorporating all accounts and related storage and balances. During the initiation stage of each execution round, the state of current instruction is identified by obtaining the nth byte code value (defined by pc) or executing (0 if pc >= len(code)). Notably, every instruction has its specific rationale related to its influence of the tuple. For instance, the execution of ADD, results in popping of 2 items from the stack and then pushes their resulting sum, increments pc (by 1) and decreases gas (by 1) (Wood 2018, 2-4). Concisely, primitive execution of Ethereum is achievable with relatively few lines of code (approximately a few hundred) that can optimize the EVM executions using just-in-time compilation.

4.5. Blockchain and Mining

The Ethereum blockchain draws most of its implementation from Bitcoin’s blockchain albeit with distinct differences in architecture. As opposed to Bitcoin, the block in Ethereum maintains a record of an up-to-date states and a transaction list. Besides this, it also stores the block difficulty and number. A primary algorithm of the Ethereum block validation mechanism consists of the following procedures: first, a validity and reference check of the preceding block is done. Second, a counter-check on the preceding block’s time stamp and current block is done to ensure the latter is greater, but below 15 minutes in the forthcoming time. Third, a validity check of Ethereum-specific concepts, such as gas limit, difficulty, uncle root, block number, and transaction root is undertaken. Fourth, a validity of the block’s proof of work is carried out. Fifth, an initialization of previous block end state, S[0], takes place. Sixth, several components are initialized: the transaction list to TX; transactions numbers to n; a statement “For all I in 0..n-1, set S[i+1] = APPLY(S[i], TX[i])” (Wood 2018), else an error is returned (if consumed gas in block surpasses GASLIMIT or an application error is encountered). Seventh, an inclusion of a block reward to the miner is executed using the statement “Let S_FINAL be S[n]”. Finally, an equivalence check is done between two states (the S_FINAL representing the state of the Merkle tree root and the block header’s final state root) (Wood 2018, 2-6). If true, block validity is assured, else it is not.

To enhance efficiency in operations between adjacent blocks, one-time data storage and referencing is implemented via pointers (subtree hashes), using a specialized tree called the Patricia tree. It also performs Merkle tree concept modifications, enabling efficient insertion and deletion of nodes. This is also improving on state information storage and space, where only the last block’s state information is stored as opposed to the absolute blockchain history.

4.6. Smart Contracts

Smart contracts define autonomous, self-executing computer applications, whose execution is dependent on a programmer’s defined conditions. The programs are able to harness blockchain capabilities to facilitate, execute, and enforce two-party agreements. This feature enhances independent (no third party involved) and anonymous business transactions between parties at a cheaper cost (Boucher et al. 2017, 14). For instance, regular expense payments such as learning fees or rent can be remitted without a bank’s involvement. Applications of smart contracts spans several fields such as bond and stock trading markets, property leasing and trading, autonomous digital voting systems, digital notary contract systems, which are autonomous, among others.

A smart contract is a critical component of Ethereum. Existing in the Ethereum's execution environment, execution of a contract is prompted by a transaction or message and runs until either of three conditions – a RETURN, a STOP, or an error – are fulfilled. Smart contracts hold their own ether balances (Bergquist 2017, 17). During execution, a contract requires to access various space types for data storage: a storage space, identified by a key / value pair that is long term; a memory space, which identifies an inexhaustible and scalable byte array; and a stack space, which represents a last-in-first-out package-holder for popping and pushing of values.

Normally, writing of a smart contract is accomplished via serpent or solidity languages. Once written, its deployment within the Ethereum is realized via the Ethereum wallet or Mist. The two applications grant users and developers rights to access Ethereum applications and create accounts (MIK 2017, 16). Moreover, they support Ethereum application testing and deployment to the network based on a test-net, private-net or main-net arrangement.

5. Rationale for use of Ethereum Technology

As identified and discussed in earlier sections, authentication and authorization systems are inherently exposed to varied limitations and vulnerabilities that could result in data breaches and hijack, identity theft, and ultimately financial loss. This is a concern to the user, especially in privacy-sensitive applications where digital privacy and identity is paramount. Furthermore, with increased user applications and services, which all require registration, users are forced into multiple registrations exposing them to increased user data vulnerabilities. Hence, there is a need for employing alternative criteria to mitigate the challenges.

From the analysis of blockchain presented, the technology combines the strengths of digital signatures, P2P, and consensus protocol. The P2P network identifies blockchain's decentralized and distributed network that eliminates any possibilities of failure. An implementation of the consensus protocol guarantees one-time transactions (cannot be reverted) by recording it in a distributed, public ledger. Additionally, nodes within the network assure transaction verification and validity, thus presents a reliable and transparent system.

Digital signatures are implemented via public-private key cryptography for user identification. This identity is further protected using digital hash algorithm that is near impossible to crack. A user's identity and signed data can also be validated or verified by other users in the network. Privacy is also assured in transactions via restricted private key ownership. Concisely, blockchain technology validates security of user identity based on anonymity and complete data ownership within the network.

Several aspects give credence to adoption of blockchain in privacy protection applications: first, possibility of data hacking is minimized drastically as there is no centralized storage; second, a user owns and stores their data, which is safeguarded using advanced, up-to-date hash algorithms; third, a user can deploy same identity across multiple services; and lastly data ownership is confined to the user (and not the service providers). These qualities minimize cases of data breach, as in most scenarios, the breaches are facilitated via carelessness or consent.

The case of blockchain use is solidified by the evolution of Ethereum blockchain and its combination with smart contracts. The technology has eased the design, development and deployment of blockchain applications, particularly for privacy-sensitive applications. Based on a high-level language development (solidity and serpent) it is easy to learn, develop and deploy to a main, private or test network. The interpretation via the EVM allows for transparency and confines applications development to logic design assuring seamless deployment. Finally, a large and active Ethereum community ensures constant innovation, technology enhancements and support for any issues that may arise.

6. Applications

Presently, Ethereum supports three application categories atop its platform. These include: financial applications, where it facilitates contract entry and management via components such as savings wallet, sub-currencies, wills, hedging contracts, and financial derivatives; semi-financial applications, which combines monetary and non-monetary elements such as implementation of solution bounties that self-enforce; and finally, non-financial applications, including decentralized governance, digital voting systems, among others (Luongo and Pon 2017, 8-9). This paper proposes a more in-depth application in privacy protection via incorporation in privacy-sensitive applications. Some likely areas of implementation include: token systems; trusted payments systems; identity and reputation systems; decentralized file storage; decentralized systems access; secure data provenance management; automotive security and privacy systems; and in trade finance systems.

6.1. Token Systems

Ethereum-based token systems have applications across a number of areas including secure unforgeable coupons, incentivization point systems, and representation of smart property. It is implemented as a sub-currency with the ability to directly the transaction fees based on the defined currency. As a privacy-sensitive implementation, it would entail maintenance of an ether balance that would consequently be employed in activating user accounts and reused in funding contracts. The unforgeability quality would ensure transactions validity and safety.

6.2. Trusted Payments Systems

Traditionally, global, cross-institutional financial transactions require a third-party interaction between two parties (such as SWIFT in the banking sector) to guarantee accuracy and veracity of transactional information and prevent fraudulent transactions. This pushes costs such as fees payable, time taken for verification processes and resolution of arising issues, among others upwards. Additionally, in digital payments could result in double spending or even identity theft. This is mainly due to holding of user data by third-party players. Through Ethereum and smart contracts, decentralized consensus can be executed in facilitating cross-border payments in real-time and global financial transactions while safeguarding user data and information. In addition, decentralized consensus guarantees contingency transfers, and transacting parties’ functionality and authenticity.

6.3. Identity and Reputation Systems

The concept of developing a name registration system, such as attempted by Namecoin provides the platform for development of a reliable identity system, where users can register their names and accompanying data in a publicly database (Perlman 2017, 30; Smart Contracts Alliance 2016, 16). In Ethereum, this would entail having the database within its network for name addition, but no removal or modification capabilities. A user would have name registration rights based on a value, upon which it is recorded permanently. In more advanced systems, a name registration contract would encompass a function element enabling queries from contracts and a transfer of modification of ownership data by the registry entry owner.

6.4. Decentralized File Storage

The file storage market is facing varied inefficiencies, including safety and privacy of user data. These challenges can be mitigated through implementation of Ethereum contracts to create an ecosystem of decentralized file storage. In this implementation a decentralized contract operates through a split of the file into blocks, encryption and creation of a Merkle tree of desired data that is governed by a set of rules created by the owner. File retrieval by the owner is facilitated using a micro-payment channel protocol. A data owner could also track nodes holding a piece of the file by monitoring the contracts. If a contract is still making payments, this indicates the file is still being stored on a node in the network

6.5. Decentralized Systems Access

A decentralized access service to the system can be implemented via generation and provisioning of a private key. Using smart contracts, a user's identity can be certified without system awareness of the blockchain state. This can be implemented using message signing, hence able to perform multi-signature transactions based on a set of articulated rules. Furthermore, it would allow for implementation of the second-factor feature in the authentication process. In this scenario, a contract has the ability to respond to a challenge-response protocol using a set of defied rules.

6.6. Secure Data Provenance Management

Inflexible scientific data provenance management necessitates trustworthy collection, management and verification of data (Ramachandran and Kantarcioglu 2017, 1). Through Ethereum and smart contracts, this is achievable by incorporating a third component – an open provenance model – to create an immutable data trail. This would ensure secure capturing and validation of provenance data and safeguard it from modification, either unintentionally or maliciously.

6.7. Automotive Security and Privacy Systems

Vehicular ecosystem security and user privacy can be protected through deployment of an Ethereum-based public network that would be managed by nodes (Dori et al. 2017, 3). In such a system, a myriad of nodes, such as vehicle assembly lines, cloud storage, smart buildings, OEMs (car manufacturer), smart vehicles, software providers, and user devices (mobile phones, tablets, laptops etc.) would be interlinked to form the overlay. Basic functioning would involve maintenance of an immutable local ledger that was private based on a centralized node such as the smart building. It would allow a vehicle to connect to the overlay via two modes: the smart building or directly. The smart building would house a secure storage space for local data. An overlay block manager would be responsible for transactions verification and storage.

7. Research for entrepreneurs

Forums, communities and entrepreneurs that work with blockchain and cryptocurrencies are almost by default all exchanging knowledge online internationally. Locality and country of origin show less influence in the market, where the companies can be easily established to a country that offers best legalization and taxation policies for blockchain and cryptocurrency businesses. For the research to Finnish blockchain entrepreneur and a software entrepreneur were interviewed. Both of the Finns work with international companies, not limiting their market to their country of origin.

7.1. Research method

The interviews were conducted in a discursive method, with only few guiding predetermined questions and discussion was allowed to flow freely to the areas of individuals expertise and personal views on blockchain technology.

Blockchain entrepreneurs' interview was face-to-face discussion with deeper background dive to persons history with blockchain projects and contributions. Software entrepreneurs' interview was video interview held with Hangouts and was more lead by the main questions.

Three guiding topics in the interviews were privacy, and blockchain technology's ability to provide it, smart contracts and security and thirdly view on the rising number of cryptocurrencies and ICOs.

7.2. Results

At best blockchain will enable new generations international, more anonymized and independent, monetary interaction. New monetary interaction can pass past generations and corporations' mutual sanction- and trade embargo structures.

By the transformation, blockchain networks allow financial transactions between accounts that do not need to be linked to existing individual or company ID:s. This increases the possibility for anonymity and unbanked access to the international financial system. Anonymous transactions provide privacy for the people who are not willing to give up their behavior data to corporate data funnels. (Blockchain entrepreneur, 22 August 2018)

Blockchain also allows the tokenization of services, shares, voting rights and more, making international crowd sales and crowdfunding more accessible to the like-minded international communities. This enables new start-ups and technologies to be able to finance their operations by technological advancement benefit the society. (Blockchain entrepreneur, 22 August 2018)

On the bright side each individual has now the opportunity to have an account for transactions without banking services. On the darker side the services in blockchain ecosystems are still yet in very early stage limiting the available safety features available within the services. Also cyber-attacks have caught notable media account, as hackers seek to breach ICO companies to gain access to customer registries, ‘know your customers’-documents, crypto wallets and social media accounts. As the industry is still young and full of enthusiastic specialists of their own profession, some ICOs fail to emphasize efficient marketing, resource management and cyber security protocols. (Software entrepreneur, 22 August 2018).

8. Conclusion

Although the application possibilities of blockchain and blockchain-based technologies such as Ethereum are immense, actual working applications implementation is still in its infancy. In particular, the utilization of the Ethereum platform combined with smart contracts for development and deployment of privacy-sensitive applications is quite feasible. Based on the characteristics of: minimized data hacking possibilities, user ownership and storage of data securely via cryptographic hash algorithms, deployment of one-identity multi-service capabilities, and confinement of data ownership to the user, there is no doubt that deployment of the technology in privacy-sensitive applications can be achieved.

9. References

Blockchain entrepreneur. 22 August 2018. CFO, ICO Analyst. Interview. Helsinki.

Bergquist, Jonathan. 2017. "Blockchain Technology and Smart Contracts: Privacy-preserving Tools." Masters thesis, Uppsala University.

Boucher, Philip, Nascimento Susana, and Kritikos Mihalis. 2017. How Blockchain Technology Could Change Our Lives. STOA, Brussels: European Union.

Cong, William L., and He Zhiguo. 2018. Blockchain Disruption and Smart Contracts. Chicago: University of Chicago.

Crawford, Shaun, Meadows Ian, and Plesse David. 2016. Blockchain Technology as a Platform for Digitization. UK: Ernst & Young.

Crosby, Michael, Nachiappan, Pattanayak Pradhan, Verma, Sanjeev Verma, and Kalyanaraman Vignesh. 2015. Blockchain Technology: Beyond Bitcoin. Berkeley, CA: Sutardja Center for Entrepreneurship and Technology.

Dori, Ali, Steger Marco, Kanhere Salil, and Jurdak Raja. 2017. "Blockchain: A Distributed Solution to Automotive Security and Privacy. URL: https://pdfs.semanticscholar.org/065d/12235316a63bae48a12bc38cee4e9843fb0a.pdf. Accessed 20 March 2018

Evgneii, Khudnev. 2017. "Blockchain: Foundational Technology to Change the World." Bachelor's thesis, Lapland University of Applied Science.

Ghaffari, Zahra. 2016. "On the Application Areas of Blockchain." Master thesis, Malmo University.

Katz, Jeff, Lisimaque Gilles, Kulhanek Colleen, McGovern Mark, McKeon J, Medich C., Pauze G., Pilozzi J., and Squibbs M. 2002. Smart Cards and Biometrics in Privacy-Sensitive Secure Personal Identification Systems. Princeton, NJ: Smart Card Alliance.

Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C. 2015. "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts."

Lazorich, A., 2015. "Invisible Ink: Blockchain for Data Privacy."
Luongo, Matt, and Pon, Corbin. 2017. "The Keep Network: A Privacy Layer for Public Blockchains.". https://backend.keep.network/whitepaper. Accessed 20 March 2018

Maxwell, Winston, and Salmon John. 2017. A Guide to Blockchain and Data Protection. New York: Hogan Lovells. URL: https://www.hlengage.com/_uploads/downloads/5425GuidetoblockchainV9FORWEB.pdf Accessed 26 August 2018.

MIK, Eliza. 2017. "Smart Contracts: Terminology, Technical Limitations and Real World Complexity." Law, Innovation and Technology 9, no. 2 (October): 269-300.

Perlman, Leon. 2017. Distributed Ledger Technologies and Financial Inclusion. International Telecommunications Union.

Ramachandran, Aravind, and Dr. Kantarcioglu Murat. 2017. "Using Blockchain and Smart Contracts for Secure Data Provenance Management." https://arxiv.org/ftp/arxiv/papers/1705/1705.04958.pdf. Accessed 20, February 2018

Satyavolu, Prasad, and Sangamnerkar Abhijeet. 2016. Blockchain's Smart Contracts: Driving the Next Wave of Innovation across Manufacturing Value Chains. Teaneck, NJ: Cognizant.

Smart Contracts Alliance. 2016. Smart Contracts: 12 Use Cases for Business & Beyond. Washington, DC: Chamber of Digital Commerce.

Software entrepreneur. 24 August 2018. Director, R&D Engineer, Interview. Helsinki.

Swan, Melanie. 2015. Blockchain. Sebastopol, CA: O'Reilly.

Thakur, Mukesh. 2017. "Authentication, Authorization and Accounting with Ethereum Blockchain." Master thesis, University of Helsinki.

Dr. Wood, Gavin. 2018. Ethereum: A Secure Decentralised Generalised Transaction Ledger. https://ethereum.github.io/yellowpaper/paper.pdf. Accessed 2 February 2018.

Zyskind, G., Nathan, O., and Pentland, A. 2015. "Decentralizing Privacy: Using Blockchain to Protect Personal Data." IEEE CS Security and Privacy Workshop.