Computer-Implemented Systems and Methods for Linking a Blockchain to a Digital Twin

Field of Invention

This specification relates generally to computer-implemented methods and systems suitable for implementation in nodes of a blockchain network. The invention is particularly suited, but not limited, to use with the Bitcoin blockchain. The invention also relates to security, integrity and efficiency in computer implemented systems.

Background of Invention

Today industrial assets are designed relying on various models and a high number of data sources. Data scientists work with enormous amounts of data while specialized teams create models separately and conduct analysis for their specific tasks. The most current information and calculations may not be readily available for crucial decisions and this way of working in silos drives cost and inefficiencies, creates uncertainties, and a vast amount of time and resources get wasted. In order to get the most out of data generated by sensors and processes, digital twins are very useful.

A digital twin is a virtual dynamic copy of a real object, process, or service through which it is possible to conduct tests and prevent errors or failures. It is possible to create a digital twin of a product, a mechanical part of an aircraft or car, a production process, and so on. Digital twins can be seen as a revolutionary combination of simulations and real time data and responses.

The act of simulating a process or a system presumes knowledge of how all the variables involved in the simulation behaved in the past and a sufficiently large timeframe during which the system under test has been observed during which all the input/output variables have been recorded. Having knowledge of the past is a prerequisite for building a good simulator. The possibility of adding real time information to this process can dramatically increase the precision and the accuracy of the whole simulation until the point of mirroring the exact behaviour of a system or process. This possibility is revolutionary for the

industrial world (Industry 4.0), and Gartner classified digital twins as one of the most important disruptive technologies in 2017 [http://www.gartner.com/smarterwithgartner/gartners-top-10-technology-trends-2017/].

Figure 1 shows a simple schematic illustration of a digital twin for a physical system. Sensors gather data about a physical system which could be, for example, a part of an aircraft. Historical data is used to construct the digital twin which then evolves to simulate the physical system as further real time data is fed into the system. The digital twin can be used to monitor parameters of the physical system, assess the current status of the physical system, predict the future status of the physical system, and conduct tests in order to, for example, predict failures of the physical system and thus aid in avoiding failures, e.g. by replacing a part or by operating the physical system in a different manner.

Summary of Invention

A number of problems have been identified with current digit twin technology as discussed below.

The reliability and security of a digital twin is dependent upon the security of the data on which the digital twin is reliant. The data should ideally be immutable such that the operation of the digital twin cannot be tampered with during real time operation and correctly reflects the status of the physical system. This can be important, for example, to prevent a third party from interfering with the data such that the digital twin does not properly represent the real time status of the physical system and provides misleading information which could lead to an operation error or failure in the physical system.

Furthermore, the stored data should be immutable such that an accurate and reliable historical record of the performance of the system is retained. This may be important, for example, if the physical system fails and the data needs to be checked to confirm why the physical system failed and if any liability exists on the part of the user or manufacture of the physical system. It may also be important in a process where a certain action may be reliant on the performance of an earlier action. In this case, a third party could potentially

alter the stored data to make it look like a certain action has been performed, when it has not, thereby erroneously triggering a further action.

A further problem is that of accessibility of the data to each of a number of interested parties. The parties may well have conflicting interests and so it would be advantageous to have a neutral record of the data which is secure and immutable yet accessible by all parties.

Yet a further problem is that conventional data storage solutions, such as a black box recorder in an aircraft, may be damaged or lost in the event of an accident.

Yet another problem is that in a scenario where a further action is required once a previous action has been completed, a digital twin can indicate that the further action is required but cannot ensure that it is actually performed. For example, a digital twin may indicate that a certain physical process has been completed, thus requiring a further step such as payment for completion of the physical process. However, the digital twin cannot ensure that such a payment is actually made and thus is reliant on a party to the process to be trustworthy and pay, or otherwise pay in advance which is then depend on the trustworthiness of the provider of the process to properly complete the process.

It is an aim of certain embodiments of the present invention to address these problems by providing solutions as set out herein.

The present inventors have realized that the aforementioned problems can be addressed by using blockchain technology as a storage system for data acquired from physical systems and processes including real time applications using digital twins. The data generated from the digital twin can be, for example, data associated with one or more parameters of a physical system generated by one or more sensors monitoring the one or more parameters of the physical system. A blockchain can be used to generate an immutable transaction history of data produced by a digital twin. In the case of an error, failure, incident, or accident, parties of interest can then access and analyse an immutable set of data. This can be particularly important in safety-critical systems such as aircraft. Furthermore, as a

blockchain provides a distributed storage of data then it is not susceptible to damage or loss of an individual storage unit. Further still, the present inventors have realized that a digital twin can be made a party to a digital smart contract implemented on a blockchain network. This can ensure that steps can be executed by the blockchain network according to data received by the digital twin indicative of the state of a real physical system. That is, the blockchain network can be used to execute a digital smart contract with multiple parties related to a system or process incorporating the digital twin.

Current blockchain technology is capable of fulfilling the aforementioned functionality when relatively small amounts of data are required to be stored at relative infrequent time periods. However, block size limits, and the fact that blocks are only incorporated into the blockchain approximately every 10 minutes, means that standard blockchain technology is not well suited as a storage system for real time applications where the amount of data generated is very large and/or where there is a requirement to store the data at a high frequency / fidelity, e.g. every second or millisecond. Approaches to overcoming these issues in order to utilize the blockchain as a storage system for such real time systems are also described herein. The approaches involve one or more of: suitable protocols implemented within a blockchain network for handling large quantities of data from a digital twin; modified blockchain network architectures for handling the data within a blockchain network; and provision of a suitable interface between a digital twin and the blockchain network for handling large quantities of data from a digital twin.

In terms of providing a suitable interface between a digital twin and the blockchain network for handling large quantities of data from a digital twin, a computer-implemented method can be provided which comprises:

receiving a stream of data from a digital twin at a buffer;

transforming the stream of data into a sequence of discrete messages; and

sending the sequence of discrete messages to a receiving node of a blockchain network for subsequent storage of transactions in a blockchain, the transactions containing data associated with the sequence of discrete messages.

The sequence of discrete messages can sent from the buffer to the receiving node through a secure channel by encrypting each message of the sequence of messages. Such a secure channel can be established using a key negotiating protocol between the buffer and the receiving node to generate a shared key for encrypting and decrypting each message. A new value for the shared key can be generated each time the secure channel is established. For example, a new value for the shared key is generated for each discrete message sent to the receiving node.

For each discrete message m_i sent to the receiving node, the buffer can calculate and stores a hash $H(m_i)$. After a time t when a number of discrete massages have been sent, the buffer can send a message m_h which is composed of a sequence of hashes relating to the discrete messages sent up to the time t. The buffer can also send a further message m_H which contains a hash of the message m_h. The sequence of discrete messages can be verified at the receiving node using m_h and m_H, where m_h which is composed of a sequence of hashes relating to the discrete messages sent up to a time t and m_H contains a hash of the message m_h.

These features both provide a secure means of transmitting data from a digital twin to a blockchain network and also ensure that the data is provided in a form which is more readily handled by the blockchain network. An additional level of security can be provided by sending the sequence of discrete messages from the buffer to the receiving node through two secure channels. A comparator is able to verify if the sequence of discrete messages sent over the two secure channels is consistent, and if the sequences of discrete messages received from the two secure channels is different, the receiving node can reject the sequence and send a notification to retransmit the sequence of discrete messages.

Furthermore, the buffer can initialize a counter which keeps track of the number of discrete messages sent and can determine the number and/or rate of discrete messages sent to the receiving node, for example, based on one or more parameters indicating the state of the blockchain network at the time of transmission. Again, this can aid data management to ensure that the blockchain network is not overloaded with data.

A computer-implemented method for the receiving node is also provided comprising: receiving, at a receiving node of a blockchain network, a sequence of discrete messages generated from a stream of data from a digital twin; and verifying the sequence of discrete messages for subsequent storage of transactions in a blockchain, the transactions containing data associated with the sequence of discrete messages. Storage of transactions could be in the receiving node, in another node of the blockchain network, or distributed over a plurality of nodes in the blockchain network. For example, after verification, the sequence of discrete messages can be stored in a distributed memory pool of the blockchain network. Furthermore, the sequence of discrete messages, or hashes of the sequence of discrete messages, can then be stored in the blockchain.

The receiving node can be a node of a public blockchain network. However, in a modified interface configuration, the receiving node can be a node of a private blockchain network and the sequence of discrete messages generated by the buffer is stored in a private blockchain prior to incorporating data associated with the sequence of discrete messages into a public blockchain. In this configuration, for each message m_i generated by the buffer and stored in the private blockchain, a hash H(m) is generated and stored in the public blockchain.

Embodiments of the present invention can be provided in a variety of forms. For example, a computer readable storage medium can be provided which comprises computer-executable instructions which, when executed, configure one or more processors to perform the method as described herein. An electronic device can also be provided which comprises: an interface device; one or more processor(s) coupled to the interface device; and a memory coupled to the one or more processor(s), the memory having stored thereon computer executable instructions which, when executed, configure the one or more processor(s) to perform the method as described herein. The computer implemented methods as described herein can also be implemented in one or more of a digital twin, a buffer, a blockchain network node, or a system comprising said components. The components could be located remotely from each other or one or more of the components may be integrated.

Brief Description of the Drawings

These and other aspects of the present invention will be apparent from and elucidated with reference to, the embodiments described herein. Embodiments of the present invention will now be described, by way of example only, and with reference to the accompany drawings, in which:

Figure 1 shows a simple schematic illustration of a digital twin for a physical system;

Figure 2 shows a simple example of a commercial chain with processes being automated among parties using the blockchain and smart contracts setting conditions for the processes;

Figure 3 shows a process diagram incorporating a digital twin into the system of Figure 2;

Figure 4 shows the use of a blockchain as an independent storage system for a digital twin associated with an aircraft;

Figure 5 illustrates an incremental hashing procedure which can be used to store a given amount of data generated by a digital twin in a given time frame;

Figure 6 illustrates a modified blockchain network architecture capable of storing large quantities of data generated by a digital twin;

Figure 7 illustrates the use of a buffer, a secure channel, and a receiver as an interface configuration between a digital twin and a blockchain network;

Figure 8 illustrates the role of the buffer shown in Figure 7 including transforming a data stream from a digital twin into a sequence of discrete messages;

Figure 9 illustrates the role of the receiver shown in Figure 7 including verifying the sequence of messages received from the buffer over the secure channel;

Figure 10 shows a modified version of an interface configuration between a digital twin and a blockchain network including a second secure channel and a comparator to increase the level of security by verifying that the sequence of discrete messages sent over the two secure channels is consistent;

Figure 11 illustrates the steps of storing the received messages in a Distributed Hash Table and then storing the hashes into the blockchain;

Figure 12 shows a modified version of a portion of an interface configuration between a digital twin and a blockchain network including a private blockchain for storing data from the digital twin prior to incorporating data into the public blockchain; and

Figure 13 is a schematic diagram illustrates a computing environment in which various embodiments can be implemented.

Detailed Description

In this document we use the term 'blockchain' to include all forms of electronic, computer-based, distributed ledgers. These include, but are not limited to consensus-based blockchain and transaction-chain technologies, permissioned and un-permissioned ledgers, shared ledgers and variations thereof. The most widely known application of blockchain technology is the Bitcoin ledger, although other blockchain implementations have been proposed and developed. While Bitcoin may be referred to herein for the purpose of convenience and illustration, it should be noted that the invention is not limited to use with the Bitcoin blockchain and alternative blockchain implementations and protocols fall within the scope of the present invention.

A blockchain is a consensus-based, electronic ledger which is implemented as a computer-based decentralised, distributed system made up of blocks which in turn are made up of transactions and other information. In the case of Bitcoin, each transaction is a data

structure that encodes the transfer of control of a digital asset between participants in the blockchain system, and includes at least one input and at least one output. Each block contains a hash of the previous block to that blocks become chained together to create a permanent, unalterable record of all transactions which have been written to the blockchain since its inception. Transactions contain small programs known as scripts embedded into their inputs and outputs, which specify how and by whom the outputs of the transactions can be accessed. On the Bitcoin platform, these scripts are written using a stack-based scripting language.

In order for a transaction to be written to the blockchain, it must be "validated". Some network nodes act as miners and perform work to ensure that each transaction is valid, with invalid transactions rejected from the network. For example, software clients installed on the nodes perform this validation work on transactions that reference unspent transaction outputs (UTXO). Validation may be performed by executing its locking and unlocking scripts. If execution of the locking and unlocking scripts evaluate to TRUE and, if certain other conditions are met, the transaction is valid and the transaction may be written to the blockchain. Thus, in order for a transaction to be written to the blockchain, it must be i) validated by a node that receives the transaction – if the transaction is validated, the node relays it to the other nodes in the network; and ii) added to a new block built by a miner; and iii) mined, i.e. added to the public ledger of past transactions. The transaction is considered to be confirmed when a sufficient number of blocks are added to the blockchain to make the transaction practically irreversible. At the time of writing, the Bitcoin blockchain network is based on a blocksize which contains approximately 2000 transactions and a block is mined approximately every 10 minutes.

Although blockchain technology is most widely known for the use of cryptocurrency implementation, digital entrepreneurs have begun exploring the use of both the cryptographic security system Bitcoin is based on and the data that can be stored on the blockchain to implement new systems. It would be highly advantageous if the blockchain could be used for automated tasks and processes which are not purely limited to payments denominated in cryptocurrency. Such solutions would be able to harness the benefits of

the blockchain (e.g. a permanent, tamper proof record of events, distributed processing etc.) while being more versatile in their applications.

One area of research is the use of the blockchain for the implementation of "smart contracts". These are computer programs designed to automate the execution of the terms of a machine-readable contract or agreement. Unlike a traditional contract which would be written in natural language, a smart contract is a machine executable program which comprises rules that can process inputs in order to produce results, which can then cause actions to be performed dependent upon those results.

The present specification describes the use of the blockchain in combination with a digital twin and optionally also in combination with the use of smart contracts. As previously described, a digital twin can simplify supply management processes and can be an important diagnosis tool for safety critical systems (e.g. aircraft and aerospace industry or transportation in general). The potential applications for a combination of digital twin and blockchain technology are numerous. For example, a blockchain network can be used for safely storing information generated by a digital twin or for executing a contract with multiple parties involved related to a system or process incorporating a digital twin. A digital twin can effectively function as an interface between a physical system and the blockchain such that data about a system or process can be acquired by the digital twin and stored on the blockchain and the blockchain can trigger certain actions based on the received data according to one or more smart contracts stored on the blockchain.

The amount of real time data generated by a digital twin can vary depending upon the complexity of the system being mirrored. For example, a digital twin can be constructed that mirrors the temperature of an office. In this case, considering that the temperature of a room does not usually have sudden variations, it can be reasonable to transmit a value of temperature every minute (or every 5 minutes), in order to reduce the amount of data to be transmitted over a network. In the case of a value of temperature every minute, the digital twin will record and transmit (in real time), 60 values of temperature every hour.

Such amount of data is certainly reasonable for storage on a blockchain, and it does not create any major issues. However, the situation becomes more complicated if the system being mirrored is, for example, an engine of an aircraft during a flight, or a cylinder of a train during a train journey. In these examples, even transmitting values every second may not be sufficient, and it could be necessary to guarantee higher fidelity, e.g., a sample every millisecond (very possible in the case of an aircraft). Also, the number of variables recorded can vary and in complex systems there may be a requirement to record a large number of variable at high fidelity.

The need to consider the amount and frequency of data transmitted by a digital twin is an important premise that entails constraints when considering the blockchain as a possible backbone infrastructure for storing information related to a specific object or process.

Smart Contracts, Digital Twins and Blockchain

Digital twins can be seen as external objects interacting with a blockchain, for example as an involved party in a digital contract implemented on a blockchain. That is, digital twins can be considered entities able to record transactions on the blockchain and participate in one or more digital contracts. A protocol can be provided which allows agents:

to compose financial instruments using domain specific languages (DSLs);
to outsource contract executions to untrusted parties; and
to publicly verify the correctness of the contract execution.

Such protocols make use of cryptographic primitives that ensure:

completeness (i.e., if the protocol is followed correctly then the honest verifier will be convinced of the validity of the output).

soundness (i.e., no cheating prover can convince the honest verifier that the authenticity of the output is true); and

zero-knowledge (i.e. if the output is valid, no cheating verifier learns anything other than this fact).

The principal benefits of such protocols are:

man-in-the-middle attacks are prevented since no communications between the participants are requested;

it is hard for malicious nodes to tamper with the data due to the use of blockchain technologies; and

contract validations do not imply code re-execution. Computations are not replicated by every node in the network. Instead, proofs of honest execution are stored in the public blockchain and used for validation purposes only.

Practical example

The advent of the blockchain opened the possibility of new ways for automating processes involving a plurality of parties without the necessity of setting up an expensive and centralized network infrastructure and with the great advantage that all the participants have access to the same data stored in a tamper resistant record. In addition, blockchain technology enables smart contract protocols able to facilitate the negotiation and execution of a contract.

Let us consider a practical example in which three parties are involved, namely: (i) a supermarket; (ii) a delivery company; and (iii) an organic farmer. The farmer sells organic vegetables which must be consumed within 48 hours and, if sold to a supermarket (or a shop in general), the products must reach the final destination in within 8 hours, being kept refrigerated during the delivery. Figure 2 shows the commercial chain with processes being automated among parties using the blockchain and smart contracts setting conditions for the processes.

The three parties involved in the process are independent and all involved in the value chain. This simple example shows how trust between parties is fundamental in this scenario, if no measures were considered in this process. In a simple world, the farmer knows and trusts the delivery man, being sure that the lorry used for the delivery works properly and that the temperature inside the van will be kept under 3 degrees, as requested in the contract stipulated with the supermarket. The farmer also trusts the fact that the products will be delivered in a maximum of 8 hours. At the same time, the supermarket

manager knows both the farmer and the delivery man, and trusts both of them. In the case in which there is no trust, or previous history, the combination of a blockchain and a digital twin can solve the problem.

A digital twin is used in the lorry and mirrors the environmental conditions under which the organic vegetables are transported. Such information can be stored in the blockchain, and both the farmer and the supermarket manager would be able to verify what happened during the delivery. The blockchain can function as the backbone of the entire process. Figure 3 shows a simple example of how the blockchain can be used in the process including: (i) registering data provided by the digital twin during the delivery; and (ii) recording different transactions for assuring that all conditions match the contract. The scenario can be more defined and more complex, considering the case in which all the parties involved in the process can sign all the transactions. In the proposed example, the supermarket manager does not deal with the farmer. The case proposed is very simple and the purpose is to show how the blockchain can be intertwined with digital twin technology.

In the example illustrated in Figures 2 and 3, the digital twin can monitor both the temperature within the lorry and also the time of delivery in order to mimic the delivery process. However, in a more simple example, if it is desired to mimic only the behaviour of temperature, then a sensor of temperature can be defined as a digital twin for such a system.

Blockchain as storage system for Digital Twins

This section describes using the blockchain as a key element of a digital twin network infrastructure. The scenario in Figure 4 shows a practical safety-critical example in which the blockchain is a fundamental component of the infrastructure. In the illustrated arrangement, a blockchain is used as an independent storage system for a digital twin associated with an aircraft.

Aircraft are highly complex systems subject to frequent maintenance activities and strict controls performed by both aviation authorities and airline companies (for civil aviation).

A digital twin mirroring different subsystems of the aircraft, e.g., hydraulic pump, brake system, wings, landing gear, etc., can be extremely helpful in optimizing the maintenance process and in preventing accidents. The blockchain generates an immutable transaction history of data produced by a digital twin.

In case of near misses, or worst case scenario in case of accidents, it will be possible for both the general aviation authority and the airline company to access an immutable set of data stored on the blockchain and analyse the whole history of the flight. An array of digital twins recording the behaviour of an aircraft during a flight would constitute a powerful "flight recorder" stored in a neutral record (the blockchain) accessible by all the parties interested in checking the dataset.

Data Size and Frequency Issues

As previously described, current blockchain technology is capable of fulfilling the aforementioned functionality when relatively small amounts of data are required to be stored at relative infrequent time periods. However, block size limits, and the fact that blocks are only incorporated into the blockchain approximately every 10 minutes, means that standard blockchain technology is not well suited as a storage system for real time applications where the amount of data generated is very large and/or where there is a requirement to store the data at a high frequency / fidelity, e.g. every second or millisecond. This specification set out two approaches for overcoming the constraints imposed by the blockchain (as it is today): (i) incremental hashing of history (which may be implemented on the current bitcoin network architecture); and (ii) the use of a modified bitcoin network architecture which is adapted for handle larger quantities of data at higher rates.

Incremental hashing of history

The idea of incremental hashing of history is strictly linked with the functioning of the blockchain. The information stored in the blockchain is not the data generated by the digital twin, but only the signed hash of a given amount of data D, generated in a given

frame of time T. The time T depends on the system under analysis (e.g. if the digital twin is mirroring the temperature of a room, the frame of time will be larger than, say, if the digital twin is mirroring the function of an aircraft engine). The storage node can be a dedicated one with the warranty that the hash stored and signed in the blockchain can provide a proof of existence of the data generated by the digital twin.

Figure 5 illustrates an incremental hashing procedure which can be used to store a given amount of data generated by a digital twin in a given time frame. The steps are as follows:

1. The digital twin starts recording data at a given frequency *f;*
2. data is recorded in a private node;
3. at time T0, the node generates the first hash (H0) hashing the data recorded by the digital twin: H0=H(D) at time T0 and records the hash both locally and in the blockchain;
4. at time T0+x, every new hash will be linked with the previous one (the second hash will be linked with H0, and so on), generating a chain of hash that will be recorded in the blockchain.

The blockchain will contain a sequence of hash through which it will be possible to rebuild the whole history of data generated by the digital twin, and it will be possible to verify the authenticity of the recorded information by the private node.

Use of a modified bitcoin network architecture

The second solution uses a modified bitcoin network architecture providing specialized nodes and protocols for validation, mining, and storage functions in the bitcoin network. The architecture we propose for the Bitcoin network is illustrated in Figure 6 which shows an operational diagram indicating the steps from the moment a user submits a transaction until it ends up on the blockchain. This architecture allows storage on the blockchain of the full history of data generated by a digital twin, even when the dataset is large and generated at high frequency/fidelity.

A system is provided where special validation nodes maintain a shared memory pool of transactions between themselves by means of a distributed hash table (DHT). These specialized validation nodes may also be referred to as merchant nodes (referred to in Figure 6 as an "M-node"). M-nodes are designed to focus on fast propagation of transactions. They may or may not store a full blockchain and are not required to perform mining functions. The operational focus of the M-nodes is the quick validation and propagation of unconfirmed transactions, particularly to other M-nodes, from which the unconfirmed transactions are quickly pushed out to other nodes in the blockchain network. To facilitate this functionality, M-nodes are permitted a greater number of incoming and, particularly, outgoing connections that might otherwise be permitted for nodes under the governing protocol.

The specialized validation nodes receive transactions, validate them, and allocate them in the distributed memory pool (also referred to as the mempool). The validation nodes then offer their service, which is to provide a list of valid transactions hashes, to the miners. The miners assemble pre-blocks (block skeletons), based on those hashes and attempt to solve the hash puzzles. When a solution to the puzzle has been found, the winning miner sends a block skeleton back to the validation nodes. The validation nodes then validate the block and ensure it is stored. Initially, it will be possible and feasible for the validation nodes to store the blocks themselves. When the block size eventually exceeds a certain threshold in size the validation nodes will either: a) extend their own storage capability; or b) outsource storing to specialized storage nodes.

In the aforementioned network system, memory pools of transactions need to synchronize between validating nodes. This involves exchanging Invertible Bloom filter Lookup Tables (IBLTs - Michael T. Goodrich, 2011). The validation nodes keep an up-to-date mempool by means of IBLTs exchanged with miners and other validation nodes. Where the validation and storage functionality is combined this is referred to as a new full node as in Figure 6.

The miners send a block skeleton (tuple) consisting of

1. A nonce, *n*

2. IBLT
3. Coinbase transaction

Based on this, the new full node orders the transactions accordingly and assembles the newly mined block. New full nodes then proceed with storing the block on their own storage, as well as propagating the skeleton to other new full nodes.

This solution overcomes the limitations of the incremental hashing of history. The main problem with the incremental hashing solution is due to the fact that the datasets generated by a digital twin would be recorded in one private node (or multiple private nodes) and the blockchain would only contain signature data associated with the history of the full dataset. This condition would be sufficient for numerous applications, e.g., selling products and supply chain management, while other applications, e.g., safety critical like flight history, would require the full set of data to be recorded in a distributed blockchain ledger.

Individual nodes in the bitcoin network can be seen as clusters of nodes providing a distributed memory pool (DMP). The proposed DMP relies on a Distributed Hash Table (DHT) structure deployed in a network composed of individual trust relations between honest nodes. A node's set of connections is built on the collection of both routing and application-level information. No central authority is involved in the release or storage of trust certifications: each node maintains records of its own trusted peers.

Malicious entities need to join the network to perform some forms of attack. For instance, Sybil attacks are focused on the creation of a high number of false identities in order to compromise the system. A Sybil node connected to the network may interrupt or delay legitimate routing queries, and disseminate erroneous routing information. However, the proposed DHT routing protocol has sublinear time and space complexity and is based on the following assumptions:

- Nodes are not able to discriminate honest and malicious nodes.
- The majority of honest nodes has more connections to other honest nodes.
- Each node is responsible for storing information about a partition of key space.

The DHT protocol provides two main functions.

- *UPDATE()* is used to build routing tables and insert keys at each DHT node.
- *GET*(x, k) is used by DHT node x to find the target key-value record represented by key k.

Each DHT node x is usually identified by a public key P_x and the current IP address $addr_x$. This information is securely linked with a record $sign_x(P_x, addr_x)$, where $sign_x()$ represents the signature with the corresponding private key. The node ID is then stored in the DHT using the signed record. When a node changes location or receives a new IP address, a new record $[P_x, addr_x]$ must be stored into the DHT. A malicious node could insert wrong key-value pairs. *GET* method is responsible to verify the signature in the returned key-value records.

The data routing network can be represented by an undirected graph. A *malicious edge* connects a malicious node to an honest node, while an *honest edge* connects two honest nodes. While creating an arbitrary number of Sybil identities may be computationally affordable for a malicious entity, creating a malicious edge requires convincing an honest node to establish a trusted link to one of the Sybil-controlled identities. If there is no sparse cut dividing the honest region in two, a short random walk starting at an honest node is likely to end at an honest node. As such, this solution provides a secure methodology for providing a bitcoin network architecture which is capable of storing large quantities of data generated by a digital twin at high fidelity in order to support the coupling of the blockchain network to a digital twin system.

Interface Configuration

If the digital twin is sending critical information, e.g., is mirroring a mechanical part of a device involved in a safety critical operation, we want to be sure that the information is sent over the network through a secure channel.

In this section a more specific and detailed protocol is described for using the blockchain as a certified system for storing information generated by digital twins. More particularly, this section describes configurations and protocols for providing a suitable interface between a digital twin and the blockchain network.

Figure 7 illustrates the use of a buffer, a secure channel, and a receiver as an interface configuration between a digital twin and a blockchain network. In order to implement a secure channel (B), we need to consider a shared key K between the sender, buffer (A) in Figure 7, and the receiver (C). A prerequisite is that the key K is known only to the sender and the receiver.

The role of the buffer (A) is to work as an interface able to connect the digital twin to the secure channel, transforming the stream of data generated S, into a sequence of discrete messages m_1, m_2,..., m_n. Using this approach, the problem of dealing with a very large amount of data generated by the digital twin and sent over the network is at least partially solved. The buffer manages the collection and storage of all the data generated by the digital twin, and also manages sending the data over the secure channel. Figure 8 illustrates the role of the buffer shown in Figure 7 including transforming a data stream S from a digital twin into a sequence of discrete messages m_1, m_2, ..., m_n.

After the discretization of the stream of data S, and the generation of the n messages to be sent over the network, a new value for the key K will be generated for each message m_i. A key negotiation protocol is used between the buffer and the receiver to arrange a unique key (session key), each time that the secure channel is established, e.g., every time that a message m_i is sent. The buffer can be configured to decide how many discrete messages will be sent (for example according to the condition of the network at the time of the transmission) and will initialize a counter which will keep trace of the messages sent. For each message m_i sent over the channel, the buffer will calculate and store the hash $H(m_i)$. After a certain number of messages are sent, at a time t the buffer can send a special message m_h composed of a sequence of hashes related to the messages sent up to that specific time t. This procedure helps the receiver in checking the integrity of the discrete messages received. The message m_h has the following form:

$$m_h = H(m_1), H(m_2), \ldots, H(m_n)$$

Finally, a third message m_H is sent for concluding the procedure, a third message m_H containing the hash of the message containing the sequence of hashes, $H(m_h)$:

$$m_H=H(m_h)$$

The buffer thus sends a sequence of messages which are processed by the secure channel and then sent to the receiver.

The receiver then receives and processes the sequence of messages. Figure 9 illustrates the role of the receiver shown in Figure 7 including verifying the sequence of messages received from the buffer over the secure channel. The sequence received can be verified by the receiver using m_h and m_H.

In order to increase the level of security, an additional layer of redundancy can be introduced in the form of a second secure channel and a comparator. Figure 10 shows such a modified version of an interface configuration between a digital twin and a blockchain network including a second secure channel (B) and a comparator (C) to increase the level of security by verifying that the sequence of discrete messages sent over the two secure channels is consistent. The comparator is able to verify if the sequence of discrete messages sent over the two secure channels is consistent. If not, namely the sequence received from channel A is different from the one received from Channel B, the receiver will not accept the sequence and notify the buffer asking to retransmit the sequence.

During initialization of the secure channel, it is necessary to set up both the keys and the message number. As explained in Ferguson, Niels, et al., Cryptography engineering: design principles and practical applications, John Wiley & Sons, 2011, four keys are derived: (I) an encryption key and (II) an authentication key to send a message from the buffer to the receiver, (III) an encryption key and (IV) an authentication key to send a message from the receiver to the buffer. We have a function f with two inputs, key and

role, f(k,r), and the output will be the state for the channel S. The setup of the secure channel may be encoded as follows:

1. $K_{1E}=SHA_{256}(K, Encrypt[Buffer\ to\ Receiver])$

2. $K_{2E}=SHA_{256}(K, Encrypt[Receiver\ to\ Buffer])$

3. $K_{1A}=SHA_{256}(K, Authenticate[Buffer\ to\ Receiver])$

4. $K_{2A}=SHA_{256}(K, Authenticate[Receiver\ to\ Buffer])$

   //set the counters to zero

5. $M_{sent}=0;$

6. $M_{received}=0;$

   //define the State S

7. $S=(K_{1E},K_{2E},K_{1A},K_{2A},M_{sent},M_{received})$

For sending a message (after the discretization process operated by the buffer in order to discretize the stream of data generated by a digital twin) it is necessary to consider the session state as defined above and produce the encrypted and authenticated message.

We have a function f_1 with three inputs: the state S; the message to be sent m; and additional data to be authenticated d:

$$f_1(S,m,d)$$

The output o' is the data to be transmitted to the receiver. This may be encoded as follows:

1. countMessagesSent<n //n is the total number of messages
2. countMessagesSent= countMessagesSent+1
3. c=countMessagesSent

   //compute authentication

4. $auth=HMAC\text{-}SHA_{256}(K_{1A},c+Length(d)+d+m)$

//+ indicates concatenation, Length() returns the length in bytes

5. o=m+auth
6. K=AES_{K1E}(0+c+0)+AES_{K1E}(1+c+0)+… //AES–Advanced Encryption Standard
7. o'=c+o

For receiving a message (after the discretization process operated by the Buffer in order to discretize the stream generated by the digital twin) it is necessary to deal with the encrypted and authenticated message generated during the sending message phase and the additional data d to be authenticated. For receiving a message, we have a function f_2 with three inputs: the session state S; the output o' obtained from f_1 during the sending phase; and the additional data d to be authenticated:

$$f_2(S,o',d)$$

The output obtained from f_2 is the message sent by the buffer. The receiving protocol may be encoded as follows:

1. //the output o' received is split into c and o
2. countMessagesSent= countMessagesSent+1
3. c=countMessagesSent
4. K=AES_{K2E}(0+c+0)+AES_{K2E}(1+c+0)+… //AES – Advanced Encryption Standard
5. (re)compute the authentication
6. Verify the authentication
7. If (auth)&& countMessagesRec<n then countMessagesRec++
   Else delete(k,m)

When all the messages have been received by the receiver, it is necessary to store them in the distributedhash table (DHT) and then store the hashes (and the history of hashes as previously described) into the blockchain. Figure 11 illustrates the steps of storing the received messages in a Distributed Hash Table and then storing the hashes into the blockchain. The steps are the following:

1. Store(m_i,DHT)
2. Store(H(m_i), DHT)
3. Store(H(m_i), Blockchain)
4. Store(m_{i+1},DHT)
5. Store(H(H(m_i)+H(m_{i+1})), Blockchain)

A different approach for enabling the use of the blockchain as backbone for digital twins, is based on the use of a private blockchain which can be then mirrored in the public one. Figure 12 shows a modified version of a portion of an interface configuration between a digital twin and a blockchain network including a private blockchain for storing data from the digital twin prior to incorporating data into the public blockchain.

The role of the buffer is the one previously described, and even if the solution with a private blockchain can require less time for recording the data transmitted by the digital twin, a discretization of the stream helps in simplifying the second step of the process, mirroring or certifying the messages on the public blockchain. For each message m_i, generated by the buffer and stored in the private chain, a hash H(m) will be generated and stored in the public blockchain.

Computing environment

Figure 13 is a schematic diagram illustrates a computing environment in which various embodiments can be implemented. There is provided an illustrative, simplified block diagram of a computing device 2600 that may be used to practice at least one embodiment of the present disclosure. In various embodiments, the computing device 2600 may be used to implement any of the systems illustrated and described above. For example, the computing device 2600 may be configured for use as a data server, a web server, a portable computing device, a personal computer, or any electronic computing device. As shown in Figure 13, the computing device 2600 may include one or more processors with one or more levels of cache memory and a memory controller (collectively labelled 2602) that can be configured to communicate with a storage subsystem 2606 that includes main memory 2608 and persistent storage 2610. The main memory 2608 can include dynamic random-

access memory (DRAM) 2618 and read-only memory (ROM) 2620 as shown. The storage subsystem 2606 and the cache memory 2602 and may be used for storage of information, such as details associated with transactions and blocks as described in the present disclosure. The processor(s) 2602 may be utilized to provide the steps or functionality of any embodiment as described in the present disclosure.

The processor(s) 2602 can also communicate with one or more user interface input devices 2612, one or more user interface output devices 2614, and a network interface subsystem 2616.

A bus subsystem 2604 may provide a mechanism for enabling the various components and subsystems of computing device 2600 to communicate with each other as intended. Although the bus subsystem 2604 is shown schematically as a single bus, alternative embodiments of the bus subsystem may utilize multiple busses.

The network interface subsystem 2616 may provide an interface to other computing devices and networks. The network interface subsystem 2616 may serve as an interface for receiving data from, and transmitting data to, other systems from the computing device 2600. For example, the network interface subsystem 2616 may enable a data technician to connect the device to a network such that the data technician may be able to transmit data to the device and receive data from the device while in a remote location, such as a data centre.

The user interface input devices 2612 may include one or more user input devices such as a keyboard; pointing devices such as an integrated mouse, trackball, touchpad, or graphics tablet; a scanner; a barcode scanner; a touch screen incorporated into the display; audio input devices such as voice recognition systems, microphones; and other types of input devices. In general, use of the term "input device" is intended to include all possible types of devices and mechanisms for inputting information to the computing device 2600.

The one or more user interface output devices 2614 may include a display subsystem, a printer, or non-visual displays such as audio output devices, etc. The display subsystem

may be a cathode ray tube (CRT), a flat-panel device such as a liquid crystal display (LCD), light emitting diode (LED) display, or a projection or other display device. In general, use of the term "output device" is intended to include all possible types of devices and mechanisms for outputting information from the computing device 2600. The one or more user interface output devices 2614 may be used, for example, to present user interfaces to facilitate user interaction with applications performing processes described and variations therein, when such interaction may be appropriate.

The storage subsystem 2606 may provide a computer-readable storage medium for storing the basic programming and data constructs that may provide the functionality of at least one embodiment of the present disclosure. The applications (programs, code modules, instructions), when executed by one or more processors, may provide the functionality of one or more embodiments of the present disclosure, and may be stored in the storage subsystem 2606. These application modules or instructions may be executed by the one or more processors 2602. The storage subsystem 2606 may additionally provide a repository for storing data used in accordance with the present disclosure. For example, the main memory 2608 and cache memory 2602 can provide volatile storage for program and data. The persistent storage 2610 can provide persistent (non-volatile) storage for program and data and may include flash memory, one or more solid state drives, one or more magnetic hard disk drives, one or more floppy disk drives with associated removable media, one or more optical drives (e.g. CD-ROM or DVD or Blue-Ray) drive with associated removable media, and other like storage media. Such program and data can include programs for carrying out the steps of one or more embodiments as described in the present disclosure as well as data associated with transactions and blocks as described in the present disclosure.

The computing device 2600 may be of various types, including a portable computer device, tablet computer, a workstation, or any other device described below. Additionally, the computing device 2600 may include another device that may be connected to the computing device 2600 through one or more ports (e.g., USB, a headphone jack, Lightning connector, etc.). The device that may be connected to the computing device 2600 may include a plurality of ports configured to accept fibre-optic connectors. Accordingly, this device may be configured to convert optical signals to electrical signals that may be

transmitted through the port connecting the device to the computing device 2600 for processing. Due to the ever-changing nature of computers and networks, the description of the computing device 2600 depicted in FIG. 13 is intended only as a specific example for purposes of illustrating the preferred embodiment of the device. Many other configurations having more or fewer components than the system depicted in FIG. 13 are possible.

Summary

This specification describes how to use the blockchain as an independent storage system for digital twins. This specification also describes how digital smart contracts can be used to manage the interaction between the blockchain, a digital twin, and an external physical system. In the case where large amounts of data is required to be stored, two solutions are described, the first based on incremental hashing of history, in which only specific hashes are recorded in the blockchain, while in the second the use of distributed memory pool and new full nodes is provided for enabling the blockchain as a complete backbone of a digital twin infrastructure. Also described are configurations for providing suitable interfaces between a digital twin and a blockchain network which can handle large quantities of data and process the data securely. Two main solutions are described, the first technique explains how to build an infrastructure able to work as an interface for digital twins. The structure of the solution is based on incremental hashing of history, in which only specific hashes are recorded in the blockchain. The second solution uses a private blockchain, e.g., based on BigChainDB (McConaghy, Trent, et al., “BigchainDB: a scalable blockchain database.", BigChainDB, 2016) and the mirroring of each block in a public blockchain.

It should be noted that the above-mentioned embodiments illustrate rather than limit the invention, and that those skilled in the art will be capable of designing many alternative embodiments without departing from the scope of the invention as defined by the appended claims. In the claims, any reference signs placed in parentheses shall not be construed as limiting the claims. The word "comprising" and "comprises", and the like, does not exclude the presence of elements or steps other than those listed in any claim or the specification as a whole. In the present specification, “comprises” means “includes or

consists of" and "comprising" means "including or consisting of". The singular reference of an element does not exclude the plural reference of such elements and vice-versa. The invention may be implemented by means of hardware comprising several distinct elements, and by means of a suitably programmed computer. In a device claim enumerating several means, several of these means may be embodied by one and the same item of hardware. The mere fact that certain measures are recited in mutually different dependent claims does not indicate that a combination of these measures cannot be used to advantage.

Claims

1. A computer-implemented method for a blockchain network, the computer-implemented method comprising:

receiving a stream of data from a digital twin at a buffer;

transforming the stream of data into a sequence of discrete messages; and

sending the sequence of discrete messages to a receiving node of a blockchain network for subsequent storage of transactions in a blockchain, the transactions containing data associated with the sequence of discrete messages.

2. The computer-implemented method according to claim 1,

wherein the sequence of discrete messages is sent from the buffer to the receiving node through a secure channel by encrypting each message of the sequence of messages.

3. The computer-implemented method according to claim 2,

wherein the secure channel is established using a key negotiating protocol between the buffer and the receiving node to generate a shared key for encrypting and decrypting each message.

4. The computer-implemented method according to claim 3,

wherein a new value for the shared key is generated each time the secure channel is established.

5. The computer-implemented method according to claim 3 or 4,

wherein a new value for the shared key is generated for each discrete message sent to the receiving node.

6. The computer-implemented method according to any preceding claim,

wherein the buffer determines the number and/or rate of discrete messages sent to the receiving node.

7. The computer-implemented method according to claim 6,

wherein the buffer determines the number and/or rate of discrete messages sent to the receiving node based on one or more parameters indicating the state of the blockchain network at the time of transmission.

8. The computer-implemented method according to any preceding claim,

wherein the buffer initializes a counter which keeps track of the number of discrete messages sent.

9. The computer-implemented method according to any preceding claim,

wherein for each discrete message m_i sent to the receiving node, the buffer calculates and stores a hash $H(m_i)$.

10. The computer-implemented method according to claim 9,

wherein after a time t when a number of discrete massages have been sent, the buffer sends a message m_h which is composed of a sequence of hashes relating to the discrete messages sent up to the time t.

11. The computer-implemented method according to claim 10,

wherein the buffer sends a further message m_H which contains a hash of the message m_h.

12. The computer-implemented method according to any preceding claim,

wherein the sequence of discrete messages is sent from the buffer to the receiving node through two secure channels.

13. A computer-implemented method for a blockchain network, the computer-implemented method comprising:

receiving, at a receiving node of a blockchain network, a sequence of discrete messages generated from a stream of data from a digital twin in accordance with the computer-implemented method of any preceding claim; and

verifying the sequence of discrete messages for subsequent storage of transactions in a blockchain, the transactions containing data associated with the sequence of discrete messages.

14. The computer-implemented method according to claims 13,

wherein the sequence of discrete messages is verified at the receiving node using m_h and m_H, where m_h which is composed of a sequence of hashes relating to the discrete messages sent up to a time t and m_H contains a hash of the message m_h.

15. The computer-implemented method according to claims 13 or 14,

wherein the sequence of discrete messages is received through two secure channels whereby a comparator is able to verify if the sequence of discrete messages sent over the two secure channels is consistent, and wherein if the sequences of discrete messages received from the two secure channels is different, the receiving node rejects the sequence and sends a notification to retransmit the sequence of discrete messages.

16. The computer-implemented method according to any one of claims 13 to 15,

wherein, after verification, the sequence of discrete messages is stored in a distributed memory pool of the blockchain network.

17. The computer-implemented method according to any one of claims 13 to 16,

wherein hashes of the sequence of discrete messages are stored in the blockchain.

18. The computer-implemented method according to any preceding claim,

wherein the receiving node is a node of a public blockchain network.

19. The computer-implemented method according to any one of claims 1 to 17,

wherein the receiving node is a node of a private blockchain network and the sequence of discrete messages generated by the buffer is stored in a private blockchain prior to incorporating data associated with the sequence of discrete messages into a public blockchain.

20. The computer-implemented method according to claim 19,

wherein for each message m_i generated by the buffer and stored in the private blockchain, a hash H(m) is generated and stored in the public blockchain.

21. A computer readable storage medium comprising computer-executable instructions which, when executed, configure one or more processors to perform the method of any one of claims 1 to 20.

22. An electronic device comprising:

an interface device;

one or more processor(s) coupled to the interface device;

a memory coupled to the one or more processor(s), the memory having stored thereon computer executable instructions which, when executed, configure the one or more processor(s) to perform the method of any one of claims 1 to 20.

23. A buffer, the buffer configured to perform the method of any one of claims 1 to 12.

24. A node of a blockchain network, the node configured to perform the method of any one of claims 13 to 20.

25. A system comprising a digital twin, the buffer of claim 23 and the blockchain network node of claim 24.