

Tokenisation Method and System for Implementing Exchanges on a Blockchain

This invention relates generally to a tokenisation method and corresponding system for the exchange of entities or assets via a computer-implemented network, and control thereof. It relates to security mechanisms for exchanges and transfers on a network. In particular, it relates to the transfer of contracts. It may be suited for use with, but not limited to, a non-debt lending process. It may be used in conjunction with any peer-to-peer distributed network. This may be blockchain-related technology, including (but not limited to) the Bitcoin Blockchain.

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 an electronic ledger which is implemented as a computer-based decentralised, distributed system made up of blocks which in turn are made up of transactions. 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 so 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”. Network nodes (miners) perform work to ensure that each transaction is valid, with invalid transactions rejected from the network. Software clients installed on the nodes perform this validation work on an unspent transaction (UTXO) by executing its locking and unlocking
5 scripts. If execution of the locking and unlocking scripts evaluate to TRUE, the transaction is valid and the transaction is written to the blockchain. Thus, in order for a transaction to be written to the blockchain, it must be i) validated by the first 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
10 public ledger of past transactions.

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
15 Blockchain to implement new systems. It would be highly advantageous if the blockchain could be used for automated tasks and processes which are not limited to the realm of cryptocurrency. Such solutions would be able to harness the benefits of the blockchain (e.g. a permanent, tamper proof records of events, distributed processing etc) while being more versatile in their applications.

20

One area of current 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
25 comprises rules that can process inputs in order to produce results, which can then cause actions to be performed dependent upon those results.

Another area of blockchain-related interest is the use of ‘tokens’ (or ‘coloured coins’) to represent and transfer real-world entities via the blockchain. A potentially sensitive or
30 secret item can be represented by the token which has no discernable meaning or value. The token thus serves as an identifier that allows the real-world item to be referenced from the blockchain.

The present invention incorporates various technical concepts to provide a blockchain-based mechanism and infrastructure which enables lending between different parties. This can be achieved without the presence of debt. Lending is an integral part of the financial services marketplace, allowing borrowers to receive funds from lenders in return for subsequent payment of those advanced funds. Traditional lending via a financial institution such as a bank has, in recent years, been extended through peer-to-peer (P2P) lending where individuals lend pooled funds to a borrower in general for a higher individual return, but with increased risk of loss of the advanced funds.

10

There are a number of P2P pools with their own bespoke trading exchanges requiring individual registration onto those applications in order to participate in the P2P lending process (e.g. Zopa, Funding circle). These loans are underpinned and implemented upon the traditional banking network and technical infrastructure within the territory in which they operate. Therefore, the present systems for P2P lending are restrictive and complex by nature.

15

It would be advantageous to provide an alternative solution. Benefits of this solution could include, for example, elimination of the need for local bespoke exchanges whilst enabling sophisticated lending processes to be carried out. Known benefits of the blockchain (such as its tamper-proof, permanent record of transactions) could be harnessed to advantage. This solution would provide an entirely new architecture and technical platform. Such an improved solution has now been devised.

20

Thus, in accordance with the present invention there is provided a method and system as defined in the appended claims.

25

Therefore, in accordance with the invention there may be provided a method and corresponding system for controlling the performance of a process conducted via (i.e. using) a blockchain. The block chain may or may not be the Bitcoin blockchain. The process may be a lending process. The invention may also incorporate tokenisation and

30

cryptographic techniques which provide enhanced security to the transfer of a digital asset or item.

The invention may provide a blockchain-implemented method. It may provide a mechanism for embedding data. The data may be embedded in a blockchain transaction (Tx). Additionally or alternatively, it may comprise a method for the control, exchange and/or transfer of an asset or item. This may be a digital asset. Additionally or alternatively, it may provide a tokenisation method. The asset and/or contract may be tokenised.

10

It may comprise the steps:

deriving a public-key-private key cryptographic pair for an item or portion of data;
deriving a signature for the data using the public key-private key cryptographic pair;
codifying the data to generate codified metadata for the data.

15

The data may relate to an asset or entity, or may relate to a (smart) contract associated with the asset. The smart contract may be stored in a computer-based resource, which may be separate to the blockchain.

20 The invention may be arranged to control a transfer made via a blockchain in respect of an asset or representation of an asset. The asset or representation may comprise or be associated with one or more private cryptographic keys, each private key being associated with an owner/controller of the asset. The private key(s) may part of a cryptographic key pair. One or more of the private keys may be generated according to a technique
25 substantially as described below in the section entitled “generating a public key for an Asset”. One or more keys may be generated using a common secret. It may be a sub-key which is generated from a master cryptographic key.

The invention may comprise the allocation or generation of an address and/or
30 cryptographic key for an asset. It may also comprise the allocation and/or generation of an address and/or cryptographic key for an owner or controller of the asset, or some other entity associated with the asset. The address/key for the asset and/or controller be created

using the deterministic sub-key process defined below in the section entitled “Generating a public key for an asset”. In one or more embodiments, the address may be a blockchain address. Preferably, it is generated as a sub-address derived from a master address (key). The master key may belong to, or be associated with, the controller/owner of the asset.

- 5 The use of the sub-key generation technique may allow direct interaction between an asset and an owner, in a secure manner which is cryptographically enforced. It may also enable the ability to nest or structure, possibly in a hierarchical manner, multiple (smart) contracts, each being controlled by a respective computing resource, within the same blockchain transaction.

10

The invention may comprise a technique for determining a secret which is common to a first and a second entity. The entity may be referred to as a node, or an asset. The first or second entity may be an asset and the other entity may be a controller or other entity associated with the asset. The invention may comprise a method substantially as described

- 15 below in the section entitled “generating a public key for an Asset). This method may comprise the steps:

- Determining a first entity second private key based on at least a first entity master private key and a generator value;
- Determining a second entity second private key based on at least a second entity master private key and the generator value;
- Determining a common secret (CS) at the first entity based on the first entity second private key and the second entity second public key, and determining the common secret (CS) at the second entity based on the second entity second private key and first entity second public key;
- wherein the first entity second public key and the second entity second public key are respectively based on at least the first/second entity master key and the generator value.

20

25

The method may further comprise the step of transmitting the codified metadata to the blockchain. The codified metadata may be arranged so as to reference or provide access to a contract. This may be a smart contract as known in the art. The metadata may be codified substantially in accordance with a codification scheme as described below in the

30

section entitled “Codification Scheme”. It may comprise three parameters or data items relating to:

- 1) an amount of shares available under the contract
- 2) A quantity of transfer units or value items to be transferred from one party to
5 another; these may be units of a cryptocurrency
- 3) A factor for calculating a value for the quantity of transfer units

The method may further comprise the step of receiving a signature and a script from at least one user to enable access to the embedded data. The script may comprise a public
10 key of a signatory. The term “user” can refer to a human user or a computer-implemented resource or agent.

The metadata may comprise a hash of the data. The metadata may comprise a pointer or other reference to the data. This may allow the data to be referenced and/or accessed.
15 Thus the data may be stored “off-block” ie not in the blockchain.

The hash may be used as a primary key in a lookup table where the data is stored. This may be a DHT.

20 The method may also comprise a blockchain-implemented system. The system may be operative to:

derive a public-key-private-key cryptographic pair for the data;
derive a signature for the data using the public key-private key cryptographic pair;
codify the data to generate codified metadata for the data.

25

The system may be arranged to implement any embodiment of the method above.
The system may be operative to transmit the codified metadata to the blockchain.

The system may be operative to receive a signature and a script from at least one user to
30 enable access to the embedded metadata. The script may comprise a public key of a signatory.

The system may be operative to generate a hash of the data and to store that hash as part of the metadata.

The system may be operative to generate a pointer or other reference to the data. The

5 system may be operative to store the hash as a primary key in a lookup table where the data is stored.

Additionally or alternatively, one or more embodiments of the invention may comprise a method for the generation of one or more cryptographic keys. This may be substantially as

10 described below in the section entitled "Generating a public key for an asset". It may comprise the following steps:

- determining, at a first node (C), a common secret (CS) that is common with the first node (C) and a second node (S), wherein the first node (C) is associated with a first asymmetric cryptography pair having a first node master private key (V_{1C}) and a first node master public key (P_{1C}), and the second node (S) is associated with a second asymmetric cryptography pair having a second node master private key (V_{1S}) and a second node master public key (P_{1S}), wherein the method comprises:

15

- determining a first node second private key (V_{2C}) based on at least the first node master private key (V_{1C}) and a deterministic key (DK);

20

- determining a second node second public key (P_{2S}) based on at least the second node master public key (P_{1S}) and the deterministic key (DK); and

- determining the common secret (CS) based on the first node second private key (V_{2C}) and the second node second public key (P_{2S}),

wherein the second node (S) has the same common secret (S) based on a first node second public key (P_{2C}) and a second node second private key (V_{2S}), wherein:

25

- the first node second public key (P_{2C}) is based on at least the first node master public key (P_{1C}) and the deterministic key (DK); and

- the second node second private key (V_{2S}) is based on at least the second node master private key (V_{1S}) and the deterministic key (DK).

30

The deterministic key (DK) may be based on a message (M). The method may also comprise the steps of:

-8-

- generating a first signed message (SM1) based on the message (M) and the first node second private key (V_{2C}); and
 - sending, over the communications network, the first signed message (SM1) to the second node (S),
- 5 wherein the first signed message (SM1) can be validated with a first node second public key (P_{2C}) to authenticate the first node (C).

Preferably, the method may comprise the steps of:

- receiving, over the communications network, a second signed message (SM2) from
- 10 the second node (S);
- validating the second signed message (SM2) with the second node second public key (P_{2S}); and
 - authenticating the second node (S) based on the result of validating the second signed message (SM2),
- 15 wherein the second signed message (SM2) was generated based on the message (M), or a second message (M2), and the second node second private key (V_{2S}).

Preferably, the method may comprise the steps of:

- generating a message (M); and
- 20 - sending, over a communications network, the message (M) to the second node (S).

Preferably, the method may comprise the steps of:

- receiving the message (M), over the communications network, from the second node (S).

25 Preferably, the method may comprise the steps of:

- receiving the message (M), over the communications network, from another node.

Preferably, the method may comprise the steps of:

- receiving the message (M) from a data store, and/or an input interface associated with the
- 30 first node (C).

Preferably, the first node master public key (P_{1C}) and second node master public key (P_{1S}) may be based on elliptic curve point multiplication of respective first node master private key (V_{1C}) and second node master private key (V_{1S}) and a generator (G).

5 Preferably, the method may comprise the steps of:

- receiving, over the communications network, the second node master public key (P_{1S}); and
- storing, at a data store associated with the first node (C), the second node master public key (P_{1S}).

10

Preferably, the method may comprise the steps of:

- generating, at a first node (C), the first node master private key (V_{1C}) and the first node master public key (P_{1C});
- sending, over the communications network, the first node master public key (P_{1C}) to the

15 second node (S) and/or other node; and

- storing, in a first data store associated with the first node (C), the first node master private key (V_{1C}).

Preferably, the method may comprise the steps of:

- 20 - sending, over the communications network, to the second node, a notice indicative of using a common elliptic curve cryptography (ECC) system with a common generator (G) for the method of determining a common secret (CS), and wherein the step of generating the first node master private key (V_{1C}) and the first node master public key (P_{1C}) comprises:

- 25 - generating the first node master private key (V_{1C}) based on a random integer in an allowable range specified in the common ECC system; and
- determining the first node master public key (P_{1C}) based on elliptic curve point multiplication of the first node master private key (V_{1C}) and the common generator (G) according to the following formula:

30 $P_{1C} = V_{1C} \times G.$

Preferably, the method may comprise the steps of:

-10-

- determining the deterministic key (DK) based on determining a hash of the message (M),
and

wherein the step of determining a first node second private key (V_{2C}) is based on a scalar
addition of the first node master private key (V_{1C}) and the deterministic key (DK)

5 according to the following formula:

$$V_{2C} = V_{1C} + DK, \text{ and}$$

wherein the step of determining a second node second public key (P_{2S}) is based on the
second node master public key (P_{1S}) with elliptic curve point addition to the elliptic curve
point multiplication of the deterministic key (DK) and the common generator (G)

10 according to the following formula:

$$P_{2S} = P_{1S} + DK \times G.$$

Preferably, the deterministic key (DK) may be based on determining a hash of a previous
deterministic key.

15 The first asymmetric cryptography pair and the second asymmetric cryptography pair may
be based on a function of respective previous first asymmetric cryptography pair and
previous second asymmetric cryptography pair.

Any feature described above in relation to one embodiment or aspect may be used in
20 relation to any other aspect or embodiment.

These and other aspects of the present invention will be apparent from and elucidated with
reference to, the embodiment described herein.

25 An embodiment of the present invention will now be described, by way of example only,
and with reference to the accompany drawings, in which:

Figure 1 illustrates a P2P network comprising a plurality of computing devices;

30 Figure 2a illustrates the relative positioning of the parties involved with the system;

Figures 2b to 2f illustrate the generation of a public-key for an asset;

Figure 3 illustrates a set-up transaction for Scenario 1;

Figure 4 illustrates the offer of the investment by Colin as a transaction;

5

Figures 5 to 8 illustrate bids on the investment by Colin, Bob, Mary and Frank;

Figure 9 illustrates a transaction which details the issuance of the raised funds to Colin as well as the issuance of the issued units to the investors.

10

Figure 10 illustrates the generation of returns on the investment by Colin;

Figure 11 illustrates the generation of returns to Colin, Bob, Mary and Frank;

Figure 12 illustrates the metadata generated by the system for the asset;

15

Figure 13 illustrates a transaction characterising Jane's purchase from Bob;

Figure 14 illustrates a transaction which issues the ownership to Jane from Bob;

20 Figure 15 illustrates a transaction which enables the return on Jane's asset to be returned by the generated asset;

Figure 16 illustrates asset metadata in respect of a second investment generated against an asset;

25

Figure 17 illustrates asset metadata in respect of the establishment of an estate comprising houses;

Figure 18 illustrates a transaction which creates the property on the blockchain;

30

Figure 19 illustrates a transaction in which the first house is released to Alice;

Figure 20 illustrates a transaction in which Alice generates rental income from the first house;

Figure 21 illustrates a transaction which enables a return to be paid on an investment;

Figure 22 illustrates a transaction which enables Alice to sell a part of her investment; and

Figure 23 illustrates a transaction which enables rental income to be generated by the investment.

One or more embodiments of the invention may enable the ultimate owner of a property or other asset to borrow funds against that asset. This can be achieved in a manner which does not require the return to the investors to be determined through the payment of interest, making it compliant with non-interest forms of lending.

By tokenising the asset, it is possible to separate the ownership of the asset from the asset's use. Thus, the owner (or "controller") and their asset become two separate entities within the present invention. As a result, the owner can use, control and manage the asset in a more powerful and flexible way e.g they can contract it out to other parties.

In order to achieve this, the invention may comprise the use of a separate address for the asset from the owner. This can be created using the deterministic sub-key process defined below in the section entitled "Generating a public key for an asset". In one or more embodiments, the address is a blockchain address. Preferably, it is generated as a sub-address derived from a master address (key). The master key may belong to, or be associated with, the controller/owner of the asset.

Furthermore, a smart contract is generated which implements a contract structure. The smart contract allows others to bid on an investment in the asset. Additional smart contracts can also be registered against the property/asset and recorded on the Blockchain such as, for example, somebody renting the property themselves, providing a particular service related to the property or the facilitation of a financial investment in the property.

-13-

Investors in the property can then receive returns or pay the costs associated with the upkeep of the property.

The invention enables the ultimate owner of an asset (or the majority of the current owners of the contract if that is how the terms of the contract are structured) to:

- Represent their asset on the Blockchain in a manner that allows their ownership to be hidden if so desired; thus, anonymity is provided for network users
- Sell, transfer or exchange all or part of their share of the asset to another investor;
- Allow the trading of that asset between investors at a price set by the open market without reference to the ultimate owner;
- Allow the asset ownership to generate a return payable to the current investors;
- Allow the asset to recoup its costs from the current investors;
- Buy back the investments (subject to the terms of the original investment contract itself); and
- Raise addition investment against the asset either through increasing the total number of units in existence (akin to a rights issue) or by creating an additional investment against the asset.

The following benefits may be realised by a system in accordance with the invention.

- Investors with limited funds can invest in property in a way that negates the need for a Fixed Unit Trust or Real Estate Investment Trust (REIT). This is because the owner of the property stipulates in the smart contract what bids are acceptable. It is then up to the investor to decide if the investment is for them. Depending on the specifics of the contract, it can allow a smaller investor access to investments not easily available using the traditional investment methods; and
- Investors with an ethical or religious belief that prevents them from investing in a scheme with a usury, can do so because all the returns are published up front in the smart contract. As such it is 100% transparent up front what the financial returns will be.

A system which may be used to implement the embodiment may comprise a P2P network as illustrated in Figure 1. Figure 1 illustrates a plurality of computers which are configured to form a P2P network.

- 5 A blockchain is a P2P network with a distributed architecture which shares work and tasks between interconnected computing devices which are known as “nodes” or “peers”. Such a network is decentralised in that no individual computer is designated as being “in charge”. Recent years have witnessed the use of P2P networks for the implementation of the consensus-based Bitcoin Blockchain and Bitcoin inspired variations.

10

Nomenclature

The following terms have been used herein and may be construed in accordance with the following meanings:

Name	Type
Owner	Defines the actor that owns the Asset which will underpin the investment
Asset	Defines the asset which will create the investment product.
Investor	Defines an actor (there can be more than one instance of this actor within any given scenario) that wishes to purchase units within an investment product.
Secondary Investor	Defines an actor (there can be more than one instance of this actor within any given scenario) that wishes to purchase units for an investment product from an existing investor.
Repository	The Repository is an actor, which may be an individual or corporate entity, which acts as a trusted repository for any documents.
Document	The Document is an abstract entity that contains structured information about the investment or asset process
Rule	This is a specialised type of Document that contains the formal definition of the rule that determines how to calculate one or more return against an investment.

Name	Type
Asset Definition	This is a specialised type of Document that contains the definition of the underlying Asset
Investment Contract	This is a specialised type of Document that contains the definition of the investment contract
Facilitator	The Facilitator is an actor, which may be an individual or corporate entity, which acts as an intermediary between the Investor(s) and Asset(s) in the arrangement, or ongoing management of the Investment.

The relative positioning of these individuals in the use of the described system is illustrated in Figure 2a.

- 5 Although the invention is described in the context of the purchase of property this is not intended to be limiting and it is to be understood that the invention can be used advantageously for a variety of purposes or applications, and provides a variety of technical benefits.

10 **Scenario 1 – Property Purchase**

Figures 3 to 8 illustrate the process by which a buyer named Colin purchases a property costing 1000 BTC.

- 15 Figure 3 illustrates a transaction which enables Colin to set up the purchase of the property. This acts as a seed transaction which enables the invitation to be initiated.

Colin wishes to purchase a property costing 1000 BTC using a unit trust arrangement with other investors. Colin generates transaction metadata using a suitably programmed

- 20 computing (client) device.

The transaction is constructed in such a way that the transaction metadata sets out details of the property and makes no mention of the owner or of the share that he owns.

The transaction metadata sets out that the buyer will put up 51% of the purchase cost, thereby holding a controlling interest, but raise the additional 49% from investors. In return for this investment, each of the investors will get a 'rental' income every six months linked to the current market rate. The transaction metadata is published to a blockchain as a transaction as set out in Figure 4.

10 The system generates a smart contract detailing the:

- Total size of the investment pot (1000 BTC)
- The number of units (100) plus the 'face' value of those units (10 BTC)
- The fact that the investment is uncapped (e.g. there is no termination date defined)
- There is a break clause that can be triggered at any point with the 'face' value of the units returned to the investors (to cover Colin selling the property)
- The return conditions, of a return plus the calculation rules every six months

A total of four investors put up the required 1000 BTC investment. One of those investors is Colin (the de-facto owner who puts up the majority stake to ensure that he continues to control the underlying asset). The other investors are: Bob who puts up 230 BTC of the required 1000 BTC investment; Mary who puts up 90 BTC of the required 1000 BTC investment; and Frank who puts up 170 BTC of the required 1000 BTC investment. The number of units are allocated proportionally to Colin (51 Units), Bob (23 Units), Mary (9 Units) and Frank (17 Units) by the system. The transactions illustrating the bids from Colin, Bob, Mary and Frank are illustrated in Figures 5 to 8.

The asset issues the raised funds to Colin in transaction ID ASSET S1-T3 as illustrated in Figure 9. ASSET S1-T3 receives each of the transactions set out in Figures 5 to 8 as an input and the units are issued to Colin, Bob, Mary and Frank in the output of ASSET S1-T3.

-17-

That is to say, the transaction ASSET-S1-T3 performs two functions within the same transaction. It firstly consolidates and passes the raised funds to Colin and simultaneously issues units in the asset to the investors Colin, Bob, Mary and Frank.

- 5 For Colin to unlock his output from ASSET S1-T3, he must sign the hash of the redeem script:

OP_2 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
PubK-Colin PubK-Asset OP_6 OP_CHECKMULTISIG

10

For Bob to unlock his output from ASSET S1-T3, he must sign the hash of the redeem script:

OP_2 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
15 PubK-Bob PubK-Asset OP_6 OP_CHECKMULTISIG

For Mary to unlock her output from ASSET S1-T3, she must sign the hash of the redeem script:

- 20 OP_2 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
PubK-Mary PubK-Asset OP_6 OP_CHECKMULTISIG

For Frank to unlock his output from ASSET S1-T3, he must sign the hash of the redeem script:

25

OP_2 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
PubK-Frank PubK-Asset OP_6 OP_CHECKMULTISIG

- Periodically, the asset will generate returns on the underlying investment that will be
30 calculated at a per share return and then scaled up to cover the individual ownership. The
current ownership position can be maintained directly by the Asset, or derived by scanning
the asset transactions on the Blockchain from the base issuance transaction generated by

the Asset (ASSET-S1-T3). For this scenario, it is assumed that Colin (the owner of the asset) will provide the funds to return to the investors using transaction ID colin-S1-T3, as illustrated in Figure 10, which generates an output which is used as an input (COLIN S1-T3 with output index IDX-00) to transaction ASSET S1-T3 to generate the returns as is
 5 illustrated in Figure 11.. This doesn't have to be the case - in practice the source of those funds can be anything but importantly it is the Asset's responsibility to calculate and pay the return.

Colin can spend his return by signing the hash of redeem script:

10

OP_1 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
 PubK-Colin OP_5 OP_CHECKMULTISIG

Bob can spend his return by signing the hash of redeem script:

15

OP_1 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
 PubK-Bob OP_5 OP_CHECKMULTISIG

Mary can spend her return by signing the hash of redeem script:

20

OP_1 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
 PubK-Mary OP_5 OP_CHECKMULTISIG

Frank can spend his return by signing the hash of redeem script:

25

OP_1 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
 PubK-Frank OP_5 OP_CHECKMULTISIG

Scenario 2 – Sale of Units

30

In this example, Bob decides to sell 10 of the units he acquired in Scenario 1 and negotiates a price of 12 BTC per unit. The spreadsheet shows this as a bilateral transaction between himself and Jane.

- 5 In this situation, there is a transaction between two investors, which results in the following occurring:

Investor	Investment Returned	Investment Amount	Units Assigned
Bob	120 BTC		-10
Jane		120 BTC	10
	120 BTC	120 BTC	0

- 10 This leaves the state of the holdings as follows (assuming that Jane paid the market rate):

Investor	Original Investment	Current Valuation	Current Units
Colin	510 BTC	612 BTC	51
Bob	230 BTC	156 BTC	13
Mary	90 BTC	108 BTC	9
Frank	170 BTC	204 BTC	17
Jane	120 BTC	120 BTC	10
		1200 BTC	100

The system generates the asset metadata for the asset which enables it to trade the units that Bob would like to sell. The metadata is illustrated in Figure 12.

- 15 A purchase transaction is then generated which enables Jane to purchase the units from Bob that he would like to sell. This transaction (transaction ID JANE-S2-T1) is illustrated in Figure 13.

-20-

In order to spend the output from this transaction, Bob needs to sign a hash of redeem script:

OP_2 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
 5 PubK-Bob PubK-Asset OP_6 OP_CHECKMULTISIG

Note that this differs from the original investment detailed in Scenario 1 in that the payment is to an account controlled by Bob rather than one managed on behalf of the underlying asset itself.

10

A transaction detailing the transfer of the units to Jane from Bob is illustrated in Figure 14 with transaction ID BOB-S2-T1. In order to spend the first output a hash of the following redeem script will need to be signed:

15 OP_2 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
 PubK-Jane PubK-Asset OP_6 OP_CHECKMULTISIG

In order to spend the second output a hash of the following redeem script will need to be signed:

20

OP_2 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
 PubK-Bob PubK-Asset OP_6 OP_CHECKMULTISIG

The asset will, periodically, generate returns on the underlying investment that will be
 25 calculated at a per share return and then scaled up to cover the individual ownership. The current ownership position can be maintained directly by the Asset, or derived by scanning the asset transactions on the Blockchain from the base issuance transaction generated by the Asset (transaction ID ASSET-S1-T3). For this scenario, it is assumed that Colin (the owner of the asset) will provide the funds to return to the investors; this doesn't have to be
 30 the case - in practice the source of those funds can be anything but importantly it is the Asset's responsibility to calculate and pay the return.

-21-

The generation of returns by the asset is illustrated in the transaction set out in Figure 15 with transaction ID ASSET S2-T1. Transaction ID COLIN-S2-T1 is used as an input to transaction ID ASSET S2-T1 which generates 5 outputs in the form of returns for each of the 5 investors.

5

In order to spend the return from the first output, Colin will need to provide a signed hash of the redeem script:

OP_1 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
10 PubK-Colin OP_5 OP_CHECKMULTISIG

In order to spend the return from the second output, Bob will need to provide a signed hash of the redeem script:

15 OP_1 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
PubK-Bob OP_5 OP_CHECKMULTISIG

In order to spend the return from the third output, Mary will need to provide a signed hash of the redeem script:

20

OP_1 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
PubK-Mary OP_5 OP_CHECKMULTISIG

In order to spend the return from the fourth output, Frank will need to provide a signed
25 hash of the redeem script:

OP_1 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
PubK-Frank OP_5 OP_CHECKMULTISIG

30 In order to spend the return from the fifth output, Jane will need to provide a signed hash of the redeem script:

-22-

OP_1 AssetMetaDataA AssetMetadataB InvestmentMetadataA InvestmentMetadataB
 PubK-Jane OP_5 OP_CHECKMULTISIG

That is to say, the system enables the asset to generate returns automatically using a
 5 Blockchain transaction.

Scenario 3 – Home Improvement

The system enables the creation of a second investment against an existing asset. In this
 10 situation, the owners collectively decide to arrange for energy efficiency upgrades to be
 applied to the property. There are two ways of funding this:

- Additional units would be offered, prioritising existing investors first in proportion to their holding (this is how a traditional rights issue would operate).
- A new investment would be created that returned a percentage of the energy
 15 savings to those investors (e.g. 50% of the saving on the property's energy cost will be paid annually)

It is this second option that would require a second investment on the same asset. (You may implement the first one in a similar manner, but in practice re-issuance of the original investment with the additional units is a more efficient method of achieving the same
 20 objective.

In terms of transaction flow, Scenario 3 is a straight repeat of Scenario 1 with different InvestmentMetadata blocks to define the new investment properties.

25 The system generates asset investment metadata. The asset investment metadata is illustrated in Figure 16. Resulting transactions are identical to those illustrated by Figures 3 to 11, but clearly with the investment metadata illustrated by Figure 16.

Scenario 4 - Full Life Cycle

30

The scenario is initiated by Colin who has a desire to raise 490 BTC against a house valued at 1000 BTC. A first investor, Bob, wishes to invest 230 BTC. A second investor, Mary,

wishes to invest 90 BTC. A third investor, Frank, wishes to invest 170 BTC. The asset needs to issue 100 shares with a nominal value of 10 BTC each.

A company named Blockchain Law Associates Ltd have set up a subsidiary called BLA Registrar Ltd (or, BLAR for short). BLAR are offering a new online Blockchain based service for title deed registrations and guarantee that their processes and documents conform to all relevant laws and regulation in their jurisdiction. Furthermore they are licensed by the official State Registrar to store and manage title deeds on their behalf.

- 5
- 10 To implement this, Blockchain Law Associated Ltd creates a new sub-key for BLAR. The sub-key generation may be performed substantially in accordance with the method described below in the section entitled “generating a public key for an asset”.

NewTech Developers Ltd (NTDL) build a new housing estate and contract BLAR to store all the title deeds on BLAR’s registry. For each house in the estate, BLAR:

- 15
- (i) Creates the title deed in the form of a contract stored on a DHT in accordance with the smart contract codification scheme as described above.
 - (ii) Issues a token representing the title deed as a Bitcoin transaction to a pay-to-script-hash (P2SH), where the locking script contains the contract metadata (DHT location plus look-up hash) as well as the public keys of both BLAR and NTDL.
- 20

At this point, each property on the estate has a UTXO representing it on the Blockchain as shown in transaction BLAR-S4-T1.

25

BLAR-S4-T1 is the initial transaction to create the contract for each property on the estate. In this example there are only four properties but it is to be understood that this is illustrative and that the description herein could be applied to an estate with any number of properties.

30

The metadata fields used by the asset are illustrated in Figure 17 where the asset investment metadata for the housing estate is depicted.

The initial transaction to create the contract for each property on the estate is shown in Figure 18. It should be noted that each of the four properties has different metadata although, in this example anyway, the underlying structure of the metadata is the same for
5 each of the four properties.

BLAR-S4-T1 generates 5 outputs, i.e. one for each of the properties and a miner's fee.

In order for the first output to be spent, the system requires a signed hash of the following
10 redeem script:

OP_2 House1MetadataA House1MetadataB PubK-NTDL PubK-BLAR OP_4
OP_CHECKMULTISIG

15 In order for the second output to be spent, the system requires a signed hash of the following redeem script:

OP_2 House2MetadataA House2MetadataB PubK-NTDL PubK-BLAR OP_4
OP_CHECKMULTISIG

20

In order for the third output to be spent, the system requires a signed hash of the following redeem script:

OP_2 House3MetadataA House3MetadataB PubK-NTDL PubK-BLAR OP_4
25 OP_CHECKMULTISIG

In order for the fourth output to be spent, the system requires a signed hash of the following redeem script:

30 OP_2 House4MetadataA House4MetadataB PubK-NTDL PubK-BLAR OP_4
OP_CHECKMULTISIG

-25-

The system is then operative to update the DHT (in a separate field from the contract) with the unhashed version of the locking script of the transaction (i.e. showing the public keys and the metadata blocks). As a registrar, BLAR will also need to maintain KYC information against their customers (currently NTDL) which will be maintained in an internal database.

Alice buys a newly built house from NTDL and as part of that exchange the title deed is re-registered to Alice, by transferring the token to her nominated public key as well as to BLAR's public key in another P2SH transaction. The Contract (and thus look-up hash) remains unchanged, however, BLAR again update the 'ownership' field on the DHT with the unhashed version of the locking script (this time showing Alice's public key instead of NTDL's). BLAR also update any internal KYC database with Alice's details. At this point, the UTXO representing the ownership of the property by NTDL is spent into a new UTXO controlled by Alice. This is shown in transaction NTDL-S4-T1 in Figure 19.

The system is then operative to generate a further transaction which enables the first house (transaction output reference BLAR-S4-T1, that is the first output from the transaction illustrated in Figure 18) to be released into the ownership of Alice. For the purposes of this example, the purchase transaction itself occurred in fiat currency and isn't shown. However, the transaction to issue the property rights is illustrated in Figure 19.

In order for the property rights to be issued to Alice, Alice must provide a signed hash of the following redeem script:

```
OP_2   House1MetadataA   House1MetadataB   PubK-Alice   PubK-BLAR   OP_4
OP_CHECKMULTISIG
```

The system enables Alice to generate an investment product using service provider SP secured against the rental income for the house. The ownership of the house does not transfer to another party. The initial allocation of the house is back to Alice herself.

-26-

The system generates a transaction in respect of the investment product generated by Alice using the system. The transaction is illustrated in Figure 20.

Alice lives in the house but she also gets some rent by letting out the spare rooms. She
5 also wants to raise some capital so she decides to create an investment secured against this rental income with the return to investors being a proportion of the rental received.

Her service provider (SP) issues a limited, divisible contract with 100 shares in the income generated by the house. The contract specifies all the terms and conditions including rental
10 income and upkeep on a pro-rata basis with an escrow service contracted to facilitate the investment.

When the investment is initially created, all 100 shares are issued back to Alice for onward divestment. The locking script associated with this investment includes the metadata
15 associated with the house, the metadata associated with the investment plus the public key of the service provider and the public key for Alice.

At this point, a UTXO representing the ownership of the investment by Alice is created. This is shown in transaction SP-S4-T1 shown in Figure 20.

20

In order to realise the initial asset allocation back to Alice, Alice must provide a signed hash of the following redeem script:

25 OP_2 House1MetadataA House1MetadataB InvestMetadata1 InvestMetadata2 PubK-Alice
PubK-SP OP_6 OP_CHECKMULTISIG

The product generates returns and the system generates a transaction to enable the returns to be paid out to Alice. The transaction is illustrated in Figure 21.

30

-27-

The transaction (ID SP-S4-T2) illustrated in Figure 21 has two outputs. The first output requires a hash of the public key of Alice and the second output requires a hash of the public key of SP.

- 5 The associated automated agent(s) may be set up by the SP in accordance with the contract and the procedures detailed in the section above entitled agent establishment. For a while, as Alice is the only owner of the Investment, she (her nominated public key) is the sole recipient of the accrued rental and also the payer of contractual upkeep payments. Transaction SP-S4-T2 shows the first income distribution for this back to Alice which is
10 assumed to be 2,000 Satoshis / share.

Later Alice offers 49 shares using a P2P Exchange service and manages to sell 29 shares to Bob and 20 shares to Eve in a transaction as set out in Figure 22. As the automated agents are already set up, the change in income/cost payments occurs seamlessly based solely on
15 the transfer of the Investment tokens.

At this point, the issuance UTXO is spent with three outputs: one back to Alice, one to a redeem script with Bob's public key and one to a redeem script with Eve's public key. This is shown in transaction ALICE-S4-T1.

20

There is also the second investment return transaction shown (SP-S4-T3) which shows the new split of the investment return.

The transaction detailed in Figure 22 has four outputs. In order for Alice to spend the first
25 output, the system requires a signed hash of the following redeem script:

```
OP_2 House1MetadataA House1MetadataB InvestMetadata1 InvestMetadata2 PubK-Alice
PubK-SP OP_6 OP_CHECKMULTISIG
```

- 30 In order for Bob to spend the second output, the system requires a signed hash of the following redeem script:

-28-

OP_2 House1MetadataA House1MetadataB InvestMetadata1 InvestMetadata2 PubK-Bob
PubK-SP OP_6 OP_CHECKMULTISIG

5 In order for Eve to spend the third output, the system requires a signed hash of the
following redeem script:

OP_2 House1MetadataA House1MetadataB InvestMetadata1 InvestMetadata2 PubK-Eve
PubK-SP OP_6 OP_CHECKMULTISIG

10 The fourth output spends the change from the transaction back to Alice.

The system is operative to generate a transaction enabling a return to be paid by SP to Alice, Bob and Eve. Such a transaction is illustrated in Figure 23.

In summary, the system enables the Asset and the Investment to remain separate both in terms of documentation and processing which means there are separate tokens representing the asset itself (house) and the investment product. There are also separate DHTs that are separately managed: the registry DHT is managed by BLAR while the Investment DHT
5 (and its attendant Automated agents) are managed by Alice's SP.

The system may be configured to enable a contract to be referenced using metadata in a blockchain transaction, using a codification scheme which enables the contract to be formatted a in a variety of ways. Such a suitable codification scheme is now described
10

Codification Scheme

The metadata which is used to reference the contract can be formatted in a variety of ways. However, a suitable codification scheme is described here.

15 A contract is transferable if the rights it defines are conferred upon the holder or owner of the contract. An example of a non-transferable contract is one in which the participants are named - that is, where the rights are conferred upon a specific named entity rather than the holder of the contract. Only transferable contracts are discussed in this codification scheme.

20 A token represents a specific contract that details or defines rights conferred by a contract. In accordance with the present invention, the token is a representation of the contract in the form of a bitcoin transaction.

25 This codification method uses metadata comprising three parameters or data items. This data may be indicative of:

- i) an amount of shares available under the contract
(this may be referred to herein as 'NumShares');
- ii) a quantity of transfer units to be transferred from a sender to at least one recipient
30 (this may be referred to herein as 'ShareVal'); and

-30-

iii) a factor for calculating a value for the quantity of transfer units (this may be referred to herein as a 'pegging rate').

An advantage of this codification scheme is that it can be used to encapsulate or represent contracts as tokens on a blockchain using only the three parameters described above. In effect, the contract can be specified using a minimum of these three data items. As this codification scheme can be used for any type of transferable contract, common algorithms can be devised and applied. Further detail of these metadata items is provided as follows.

10 **Generating a Public Key for an Asset**

The system may generate a public key for an asset using a method of sub-key generation as now described.

Figure 2b illustrates a system 1 that includes a first node 3 which is in communication with a second node 7 over a communications network 5. The first node 3 has an associated first processing device 23 and the second node 5 has an associated second processing device 27. The first and second nodes 3, 7 may include an electronic device, such as a computer, phone, tablet computer, mobile communication device, computer server etc. In one example, the first node 3 may be a client (user) device and the second node 7 may be a server. The server may be a digital wallet provider's server.

The first node 3 is associated with a first asymmetric cryptography pair having a first node master private key (V_{1C}) and a first node master public key (P_{1C}). The second node (7) is associated with a second asymmetric cryptography pair having a second node master private key (V_{1S}) and a second node master public key (P_{1S}). In other words, the first and second nodes are each in possession of respective public-private key pairs.

The first and second asymmetric cryptography pairs for the respective first and second nodes 3, 7 may be generated during a registration process, such as registration for a wallet. The public key for each node may be shared publicly, such as over communications network 5.

-31-

To determine a common secret (CS) at both the first node 3 and second node 7, the nodes 3, 7 perform steps of respective methods 300, 400 without communicating private keys over the communications network 5.

- 5 The method 300 performed by the first node 3 includes determining 330 a first node second private key (V_{2C}) based on at least the first node master private key (V_{1C}) and a Generator Value (GV). The Generator Value may be based on a message (M) that is a shared between the first and second nodes, which may include sharing the message over the communications network 5 as described in further detail below. The method 300 also
- 10 includes determining 370 a second node second public key (P_{2S}) based on at least the second node master public key (P_{1S}) and the Generator Value (GV). The method 300 includes determining 380 the common secret (CS) based on the first node second private key (V_{2C}) and the second node second public key (P_{2S}).
- 15 Importantly, the same common secret (CS) can also be determined at the second node 7 by method 400. The method 400 includes determining 430 a first node second public key (P_{2C}) based on the first node master public key (P_{1C}) and the Generator Value (GV). The method 400 further include determining 470 a second node second private key (V_{2S}) based on the second node master private key (V_{1S}) and the Generator Value (GV). The method
- 20 400 includes determining 480 the common secret (CS) based on the second node second private key (V_{2S}) and the first node second public key (P_{2C}).

The communications network 5 may include a local area network, a wide area network, cellular networks, radio communication network, the internet, etc. These networks, where

25 data may be transmitted via communications medium such as electrical wire, fibre optic, or wirelessly may be susceptible to eavesdropping, such as by an eavesdropper 11. The method 300, 400 may allow the first node 3 and second node 7 to both independently determine a common secret without transmitting the common secret over the communications network 5.

30

Thus one advantage is that the common secret (CS) may be determined securely and independently by each node without having to transmit a private key over a potentially

unsecure communications network 5. In turn, the common secret may be used as a secret key (or as the basis of a secret key).

The methods 300, 400 may include additional steps. See Figure 2f. The method 300 may include, at the first node 3, generating a signed message (SM1) based on the message (M) and the first node second private key (V_{2C}). The method 300 further includes sending 360 the first signed message (SM1), over the communications network, to the second node 7. In turn, the second node 7 may perform the steps of receiving 440 the first signed message (SM1). The method 400 also includes the step of validating 450 the first signed message (SM2) with the first node second public key (P_{2C}) and authenticating 460 the first node 3 based on the result of validating the first signed message (SM1). Advantageously, this allows the second node 7 to authenticate that the purported first node (where the first signed message was generated) is the first node 3. This is based on the assumption that only the first node 3 has access to the first node master private key (V_{1C}) and therefore only the first node 3 can determine the first node second private key (V_{2C}) for generating the first signed message (SM1). It is to be appreciated that similarly, a second signed message (SM2) can be generated at the second node 7 and sent to the first node 3 such that the first node 3 can authenticate the second node 7, such as in a peer-to-peer scenario.

Sharing the message (M) between the first and second nodes may be achieved in a variety of ways. In one example, the message may be generated at the first node 3 which is then sent, over the communications network 5, to the second node 7. Alternatively, the message may be generated at the second node 7 and then sent, over the communications network 5, to the second node 7. In some examples, the message (M) may be public and therefore may be transmitted over an unsecure network 5. One or more messages (M) may be stored in a data store 13, 17, 19. The skilled person will realise that sharing of the message can be achieved in a variety of ways.

Advantageously, a record to allow recreation of the common secret (CS) may be kept without the record by itself having to be stored privately or transmitted securely.

Method of registration 100, 200

An example of a method of registration 100, 200 will be described with reference to Figure 2d, where method 100 is performed by the first node 3 and method 200 is performed by the second node 7. This includes establishing the first and second asymmetric cryptography pairs for the respective first and second nodes 3, 7.

5

The asymmetric cryptography pairs include associated private and public keys, such as those used in public-key encryption. In this example, the asymmetric cryptography pairs are generated using Elliptic Curve Cryptography (ECC) and properties of elliptic curve operations.

10

In the method 100, 200, this includes the first and second nodes agreeing 110, 210 on a common ECC system and using a base point (G). (Note: the base point could be referred to as a Common Generator, but the term 'base point' is used to avoid confusion with the Generator Value GV). In one example, the common ECC system may be based on

15 secp256K1 which is an ECC system used by Bitcoin. The base point (G) may be selected, randomly generated, or assigned.

Turning now to the first node 3, the method 100 includes settling 110 on the common ECC system and base point (G). This may include receiving the common ECC system and base point from the second node 7, or a third node 9. Alternatively, a user interface 15 may be associated with the first node 3, whereby a user may selectively provide the common ECC system and/or base point (G). In yet another alternative one or both of the common ECC system and/or base point (G) may be randomly selected by the first node 3. The first node 3 may send, over the communications network 5, a notice indicative of using the common ECC system with a base point (G) to the second node 7. In turn, the second node 7 may settle 210 by sending a notice indicative of an acknowledgment to using the common ECC system and base point (G).

The method 100 also includes the first node 3 generating 120 a first asymmetric cryptography pair that includes the first node master private key (V_{1C}) and the first node master public key (P_{1C}). This includes generating the first master private key (V_{1C}) based, at least in part, on a random integer in an allowable range specified in the common ECC

30

-34-

system. This also includes determining the first node master public key (P_{1C}) based on elliptic curve point multiplication of the first node master private key (V_{1C}) and the base point (G) according to the formula:

$$P_{1C} = V_{1C} \times G \quad (\text{Equation 1})$$

5

Thus the first asymmetric cryptography pair includes:

V_{1C} : The first node master private key that is kept secret by the first node.

P_{1C} : The first node master public key that is made publicly known.

10 The first node 3 may store the first node master private key (V_{1C}) and the first node master public key (P_{1C}) in a first data store 13 associated with the first node 3. For security, the first node master private key (V_{1C}) may be stored in a secure portion of the first data store 13 to ensure the key remains private.

15 The method 100 further includes sending 130 the first node master public key (P_{1C}), over the communications network 5, to the second node 7, as shown in figure 2d. The second node 7 may, on receiving 220 the first node master public key (P_{1C}), store 230 the first node master public key (P_{1C}) in a second data store 17 associated with the second node 7.

20 Similar to the first node 3, the method 200 of the second 7 includes generating 240 a second asymmetric cryptography pair that includes the second node master private key (V_{1S}) and the second node master public key (P_{1S}). The second node master private key (V_{1S}) is also a random integer within the allowable range. In turn, the second node master public key (P_{1S}) is determined by the following formula:

$$25 \quad P_{1S} = V_{1S} \times G \quad (\text{Equation 2})$$

Thus the second asymmetric cryptography pair includes:

V_{1S} : The second node master private key that is kept secret by the second node.

P_{1S} : The second node master public key that is made publicly known.

30

The second node 7 may store the second asymmetric cryptography pair in the second data store 17. The method 200 further includes sending 250 the second node master public key

(P_{1S}) to the first node 3. In turn, the first node 3 may receive 140 and stores 150 the second node master public key (P_{1S}).

It is to be appreciated that in some alternatives, the respective public master keys may be received and stored at a third data store 19 associated with the third node 9 (such as a trusted third party). This may include a third party that acts as a public directory, such as a certification authority. Thus in some examples, the first node master public key (P_{1C}) may requested and received by the second node 7 only when determining the common secret (CS) is required (and vice versa).

The registration steps may only need to occur once as an initial setup.

Session initiation and determining the common secret by the first node 3

An example of determining a common secret (CS) will now be described with reference to Figure 2e. The common secret (CS) may be used for a particular session, time, transaction, or other purpose between the first node 3 and the second node 7 and it may not be desirable, or secure, to use the same common secret (CS). Thus the common secret (CS) may be changed between different sessions, time, transactions, etc.

The following is provided for illustration of the secure transmission technique which has been described above.

Generating a message (M) 310

In this example, the method 300 performed by the first node 3 includes generating 310 a message (M). The message (M) may be random, pseudo random, or user defined. In one example, the message (M) is based on Unix time and a nonce (and arbitrary value). For example, the message (M) may be provided as:

$$\text{Message (M)} = \text{UnixTime} + \text{nonce} \quad (\text{Equation 3})$$

In some examples, the message (M) is arbitrary. However it is to be appreciated that the message (M) may have selective values (such as Unix Time, etc) that may be useful in some applications.

The method 300 includes sending 315 the message (M), over the communications network 3, to the second node 7. The message (M) may be sent over an unsecure network as the message (M) does not include information on the private keys.

5

Determining a Generator Value (GV) 320

The method 300 further includes the step of determining 320 a Generator Value (GV) based on the message (M). In this example, this includes determining a cryptographic hash of the message. An example of a cryptographic hash algorithm includes SHA-256 to
10 create a 256-bit Generator Value (GV). That is:

$$GV = \text{SHA-256}(M) \quad (\text{Equation 4})$$

It is to be appreciated that other hash algorithms may be used. This may include other hash algorithms in the Secure Hash Algorithm (SHA) family. Some particular examples include
15 instances in the SHA-3 subset, including SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256. Other hash algorithms may include those in the RACE Integrity Primitives Evaluation Message Digest (RIPEMD) family. A particular example may include RIPEMD-160. Other hash functions may include families based on Zémor-Tillich hash function and knapsack-based hash functions.

20

Determining a first node second private key 330

The method 300 then includes the step 330 of determining 330 the first node second private key (V_{2C}) based on the second node master private key (V_{1C}) and the Generator Value (GV). This can be based on a scalar addition of the first node master private key
25 (V_{1C}) and the Generator Value (GV) according to the following formula:

$$V_{2C} = V_{1C} + GV \quad (\text{Equation 5})$$

Thus the first node second private key (V_{2C}) is not a random value but is instead deterministically derived from the first node master private key. The corresponding public
30 key in the cryptographic pair, namely the first node second public key (P_{2C}), has the following relationship:

$$P_{2C} = V_{2C} \times G \quad (\text{Equation 6})$$

Substitution of V_{2C} from Equation 5 into Equation 6 provides:

$$P_{2C} = (V_{1C} + GV) \times G \quad (\text{Equation 7})$$

where the '+' operator refers to elliptic curve point addition. Noting that elliptic curve

5 cryptography algebra is distributive, Equation 7 may be expressed as:

$$P_{2C} = V_{1C} \times G + GV \times G \quad (\text{Equation 8})$$

Finally, Equation 1 may be substituted into Equation 7 to provide:

$$P_{2C} = P_{1C} + GV \times G \quad (\text{Equation 9.1})$$

$$10 \quad P_{2C} = P_{1C} + \text{SHA-256}(M) \times G \quad (\text{Equation 9.2})$$

Thus the corresponding first node second public key (P_{2C}) can be derivable given knowledge of the first node master public key (P_{1C}) and the message (M). The second node 7 may have such knowledge to independently determine the first node second public
15 key (P_{2C}) as will be discussed in further detail below with respect to the method 400.

Generate a first signed message (SM1) based on the message and the first node second private key 350

The method 300 further includes generating 350 a first signed message (SM1) based on the
20 message (M) and the determined first node second private key (V_{2C}). Generating a signed message includes applying a digital signature algorithm to digitally sign the message (M). In one example, this includes applying the first node second private key (V_{2C}) to the message in an Elliptic Curve Digital Signature Algorithm (ECDSA) to obtain the first signed message (SM1). Examples of ECDSA include those based on ECC systems with
25 secp256k1, secp256r1, secp384r1, secp521r1.

The first signed message (SM1) can be verified with the corresponding first node second public key (P_{2C}) at the second node 7. This verification of the first signed message (SM1) may be used by the second node 7 to authenticate the first node 3, which will be discussed
30 in the method 400 below.

Determine a second node second public key 370'

-38-

The first node 3 may then determine 370 a second node second public key (P_{2S}). As discussed above, the second node second public key (P_{2S}) may be based at least on the second node master public key (P_{1S}) and the Generator Value (GV). In this example, since the public key is determined 370' as the private key with elliptic curve point multiplication with the base point (G), the second node second public key (P_{2S}) can be expressed, in a fashion similar to Equation 6, as:

$$P_{2S} = V_{2S} \times G \quad (\text{Equation 10.1})$$

$$P_{2S} = P_{1S} + GV \times G \quad (\text{Equation 10.2})$$

10 The mathematical proof for Equation 10.2 is the same as described above for deriving Equation 9.1 for the first node second public key (P_{2C}). It is to be appreciated that the first node 3 can determine 370 the second node second public key independently of the second node 7.

15 Determine the common secret 380 at the first node 3

The first node 3 may then determine 380 the common secret (CS) based on the determined first node second private key (V_{2C}) and the determined second node second public key (P_{2S}). The common secret (CS) may be determined by the first node 3 by the following formula:

20
$$S = V_{2C} \times P_{2S} \quad (\text{Equation 11})$$

Method 400 performed at the second node 7

The corresponding method 400 performed at the second node 7 will now be described. It is to be appreciated that some of these steps are similar to those discussed above that were performed by the first node 3.

The method 400 includes receiving 410 the message (M), over the communications network 5, from the first node 3. This may include the message (M) sent by the first node 3 at step 315. The second node 7 then determines 420 a Generator Value (GV) based on the message (M). The step of determining 420 the Generator Value (GV) by the second node 7 is similar to the step 320 performed by the first node described above. In this

example, the second node 7 performs this determining step 420 independent of the first node 3.

The next step includes determining 430 a first node second public key (P_{2C}) based on the first node master public key (P_{1C}) and the Generator Value (GV). In this example, since the public key is determined 430' as the private key with elliptic curve point multiplication with the base point (G), the first node second public key (P_{2C}) can be expressed, in a fashion similar to Equation 9, as:

$$P_{2C} = V_{2C} \times G \quad (\text{Equation 12.1})$$

$$P_{2C} = P_{1C} + GV \times G \quad (\text{Equation 12.2})$$

The mathematical proof for Equations 12.1 and 12.2 is the same as those discussed above for Equations 10.1 and 10.2.

15 The second node 7 authenticating the first node 3

The method 400 may include steps performed by the second node 7 to authenticate that the alleged first node 3, is the first node 3. As discussed previously, this includes receiving 440 the first signed message (SM1) from the first node 3. The second node 7 may then validate 450 the signature on the first signed message (SM1) with the first node second public key (P_{2C}) that was determined at step 430.

Verifying the digital signature may be done in accordance with an Elliptic Curve Digital Signature Algorithm (ECDSA) as discussed above. Importantly, the first signed message (SM1) that was signed with the first node second private key (V_{2C}) should only be correctly verified with the corresponding first node second public key (P_{2C}), since V_{2C} and P_{2C} form a cryptographic pair. Since these keys are deterministic on the first node master private key (V_{1C}) and the first node master public key (P_{1C}) that were generated at registration of the first node 3, verifying first signed message (SM1) can be used as a basis of authenticating that an alleged first node sending the first signed message (SM1) is the same first node 3 during registration. Thus the second node 7 may further perform the step of authenticating (460) the first node 3 based on the result of validating (450) the first signed message.

The second node 7 determining the common secret

The method 400 may further include the second node 7 determining 470 a second node second private key (V_{2S}) based on the second node master private key (V_{1S}) and the

- 5 Generator Value (GV). Similar to step 330 performed by the first node 3, the second node second private key (V_{2S}) can be based on a scalar addition of the second node master private key (V_{1S}) and the Generator Value (GV) according to the following formulas:

$$V_{2S} = V_{1S} + GV \quad (\text{Equation 13.1})$$

$$V_{2S} = V_{1S} + \text{SHA-256}(M) \quad (\text{Equation 13.2})$$

10

The second node 7 may then, independent of the first node 3, determine 480 the common secret (CS) based on the second node second private key (V_{2S}) and the first node second public key (P_{2C}) based on the following formula:

$$S = V_{2S} \times P_{2C} \quad (\text{Equation 14})$$

15

Proof of the common secret (CS) determined by the first node 3 and second node 7

The common secret (CS) determined by the first node 3 is the same as the common secret (CS) determined at the second node 7. Mathematical proof that Equation 11 and Equation 14 provide the same common secret (CS) will now be described.

20

Turning to the common secret (CS) determined by the first node 3, Equation 10.1 can be substituted into Equation 11 as follows:

$$S = V_{2C} \times P_{2S} \quad (\text{Equation 11})$$

$$S = V_{2C} \times (V_{2S} \times G)$$

25

$$S = (V_{2C} \times V_{2S}) \times G \quad (\text{Equation 15})$$

Turning to the common secret (CS) determined by the second node 7, Equation 12.1 can be substituted into Equation 14 as follows:

$$S = V_{2S} \times P_{2C} \quad (\text{Equation 14})$$

30

$$S = V_{2S} \times (V_{2C} \times G)$$

$$S = (V_{2S} \times V_{2C}) \times G \quad (\text{Equation 16})$$

Since ECC algebra is commutative, Equation 15 and Equation 16 are equivalent, since:

$$S = (V_{2C} \times V_{2S}) \times G = (V_{2S} \times V_{2C}) \times G \quad (\text{Equation 17})$$

The common secret (CS) and secret key

- 5 The common secret (CS) may now be used as a secret key, or as the basis of a secret key in a symmetric-key algorithm for secure communication between the first node 3 and second node 7.

The common secret (CS) may be in the form of an elliptic curve point (x_s, y_s). This may
 10 be converted into a standard key format using standard publicly known operations agreed by the nodes 3, 7. For example, the x_s value may be a 256-bit integer that could be used as a key for AES₂₅₆ encryption. It could also be converted into a 160-bit integer using RIPEMD160 for any applications requiring this length key.

- 15 The common secret (CS) may be determined as required. Importantly, the first node 3 does not need to store the common secret (CS) as this can be re-determined based on the message (M). In some examples, the message(s) (M) used may be stored in data store 13, 17, 19 (or other data store) without the same level of security as required for the master private keys. In some examples, the message (M) may be publicly available.
- 20 However depending on some application, the common secret (CS) could be stored in the first data store (X) associated with the first node provided the common secret (CS) is kept as secure as the first node master private key (V_{1C}).

Advantageously, this technique can be used to determine multiple common secrets that
 25 may correspond to multiple secure secret keys based on a single master key cryptography pair.

Hierarchy of Generator Values (keys)

- For example, a series of successive Generator Values (GVs) may be determined, where
 30 each successive GV may be determined based on the preceding Generator Value (GV). For example, instead of repeating steps 310 to 370 and 410 to 470 to generate successive single-purpose keys, by prior agreement between the nodes, the previously used Generator

Value (GV) can be rehashed repeatedly by both parties to establish a hierarchy of Generator Values. In effect, the Generator Value, based on the hash of a message (M), can be a next generation message (M') for the next generation of Generator Value (GV'). Doing this allows successive generations of shared secrets to be calculated without the need for further protocol-establishment transmissions, in particular transmission of multiple messages for each generation of common secrets. The next generation common secret (CS') can be computed as follows.

Firstly, both the first node 3 and the second node 7 independently determine the next generation of the Generator Value (GV'). This is similar to steps 320 and 420 but adapted with the following formulas:

$$M' = \text{SHA-256}(M) \quad (\text{Equation 18})$$

$$GV' = \text{SHA-256}(M') \quad (\text{Equation 19.1})$$

$$GV' = \text{SHA-256}(\text{SHA-256}(M)) \quad (\text{Equation 19.2})$$

The first node 3 may then determine the next generation of the second node second public key (P_{2S'}) and the first node second private key (V_{2C'}) similar to steps 370 and 330 described above, but adapted with the following formulas:

$$P_{2S'} = P_{1S} + GV' \times G \quad (\text{Equation 20.1})$$

$$V_{2C'} = V_{1C} + GV' \quad (\text{Equation 20.2})$$

The second node 7 may then determine the next generation of the first node second public key (P_{2C'}) and the second node second private key (V_{2S'}) similar to steps 430 and 470 described above, but adapted with the following formulas:

$$P_{2C'} = P_{1C} + GV' \times G \quad (\text{Equation 21.1})$$

$$V_{2S'} = V_{1S} + GV' \quad (\text{Equation 21.2})$$

The first node 3 and the second node 7 may then each determine the next generation common secret (CS'). In particular, the first node 3 determines the next generation common secret (CS') with the formula:

$$CS' = V_{2C'} \times P_{2S'} \quad (\text{Equation 22})$$

The second node 7 determines the next generation common secret (CS') with the formula:

$$CS' = V_{2S}' \times P_{2C}' \quad (\text{Equation 23})$$

Further generations (CS'', CS''', etc.) can be calculated in the same way to create a chain hierarchy. This technique requires that both the first node 3 and the second node 7 keep track of the original Message (M) or the originally calculated Generator Value (GV), and to which node it relates. As this is publicly known information there are no security issues regarding the retention of this information. Accordingly, this information might be kept on 'hash tables' (linking hash values to public keys) and distributed freely across the network 5 (for example using Torrent). Furthermore, if any individual common secret (CS) in the hierarchy is ever compromised, this does not affect the security of any other common secrets in the hierarchy provided the private keys V_{1C} , V_{1S} remain secure.

Tree structure of keys

As well as a chain (linear) hierarchy as described above, a hierarchy in the form of a tree structure can be created. With a tree structure, a variety of keys for different purposes such as authentication keys, encryption keys, signing keys, payment keys, etc. may be determined whereby these keys are all linked to a single securely maintained master key. This is best illustrated in Fig. 12 that shows a tree structure 901 with a variety of different keys. Each of these can be used to create a shared secret with another party. Tree branching can be accomplished in several ways, three of which are described below.

(i) Master key spawning

In the chain hierarchy, each new 'link' (Public/Private key pair) is created by adding a multiply rehashed Message to the original master key. For example, (showing only the private key of the first node 3 for clarity):

$$V_{2C} = V_{1C} + \text{SHA-256}(M) \quad (\text{Equation 24})$$

$$V_{2C}' = V_{1C} + \text{SHA-256}(\text{SHA-256}(M)) \quad (\text{Equation 25})$$

$$V_{2C}'' = V_{1C} + \text{SHA-256}(\text{SHA-256}(\text{SHA-256}(M))) \quad (\text{Equation 26})$$

... and so on.

To create a branch, any key can be used as a sub-master key. For example V_{2C}' can be used as a sub-master key (V_{3C}) by adding the hash to it as is done for the regular master key:

$$V_{3C} = V_{2C}' + \text{SHA-256}(M) \quad (\text{Equation 27})$$

- 5 The sub-master key (V_{3C}) may itself have a next generation key (V_{3C}'), for example:

$$V_{3C}' = V_{2C}' + \text{SHA-256}(\text{SHA-256}(M)) \quad (\text{Equation 28})$$

This provides a tree structure 903 using the master key spawning method as shown in Fig. 13.

10

(ii) Logical Association

In this method all the nodes in the tree (public/private key pairs) are generated as a chain (or in any other way) and the logical relationships between the nodes in the tree is maintained by a table in which each node in the tree is simply associated with its parent node in the tree using a pointer. Thus the pointer may be used to determine the relevant public/private key pairs for determining the common secret key (CS) for the session.

15

(iii) Message Multiplicity

New private/public key pairs can be generated by introducing a new message at any point in the chain or tree. The message itself may be arbitrary or may carry some meaning or function (e.g. it might be related to a 'real' bank account number, etc). It may be desirable that such new messages for forming the new private/public key pairs are securely retained.

20

Example of an Implementation of an Agent

- 25 The present invention can utilise a computing resource or agent (which may sometimes be referred to as an 'oracle') to perform automated aspects of the contract process. An example of a suitable agent is provided below, although other implementations may be used.

- 30 The agent may operate in conjunction with the blockchain, using it as the non-erasable tape in the implementation of a Turing machine. This agent runs in parallel with the blockchain network, overseeing and handling the execution of a (looping) process. The looping

process is designed to perform a given task such as, for example, the automation of a process or control of a device or system. This parallel resource monitors the state of the blockchain and can cause transactions to be written to the blockchain. In one sense, it utilises the Blockchain as a non-erasable tape of the Turing Machine, with the following definitions and features:

1. the Blockchain acts as the tape of the Turing Machine. Each transaction in the Blockchain represents a cell on the tape. This cell can contain symbols from a finite alphabet.
2. The tape head can read information from the blocks that have already been written onto the Blockchain.
3. The tape head can write new blocks, containing many transactions, to the end of the Blockchain. However, they cannot write onto blocks that already exist. As such, the Blockchain tape is non-erasable.
4. Metadata for each transaction can be stored as part of a multi-signature pay-to-script-hash (P2SH) transaction.

An important function of the agent is to act as an automated entity that monitors the current state of the Blockchain. It can also receive a signal or input from any off-block source. Depending on the Blockchain state and/or a received input, the agent may perform certain actions. The agent decides which action(s) are to be performed. These may or may not involve actions in the 'real world' (i.e. off block) and/or actions on the Blockchain (such as creating and broadcasting new transactions). The action that the agent takes may be triggered by the Blockchain state. The agent may also decide on the next set of transactions to be broadcast to the Bitcoin network, and subsequently written to the Blockchain.

The agent's action(s) run in parallel and simultaneously to the Blockchain (eg Bitcoin) network. In a sense, this extends the function of blockchain (eg Bitcoin) script. This continuous monitoring implements the 'loop' control-flow constructs making the combined agent and Blockchain system Turing Complete.

The Turing Machine includes two stacks:

-46-

- Data stack: This is represented by the Blockchain as described above.
- Control stack: This is represented by the agent function. This stores information relating to the repeat control-flow function.

The separation of the control stack from the data stack provides the advantage of

- 5 preventing infinite loops from occurring within the Bitcoin core, mitigating denial-of-service attacks.

The agent manages and runs subroutines that are able to loop via any type of loop construct (e.g. FOR-NEXT; REPEAT UNTIL; etc). An illustrative embodiment described herein

- 10 includes a process using one example of the ‘repeat’ construct. The user may specify the index (i) and the limit (J). These represent the current iteration number (typically counted starting from 0) and the total number of iterations of the repeat loop respectively.

For each iteration:

- 15 1. The Index increments by 1. For the exit condition, the iterations will stop when the index reaches the limit
2. A code block containing an “if *condition* then *action*” (ICTA) statement is executed; the action may be any action on or off the blockchain;
3. A cryptographic hash of this subroutine is computed. This can be stored in the
- 20 Blockchain as part of a transaction. Since the hash is unique to each code, it will enable verification of which code has been used

The body of the loop includes a code block. Each code block contains a “If *condition* then *action*” (ICTA) statement. This monitors the current state of the Blockchain for

- 25 transactions matching the:

- Start or triggering condition (e.g when a particular date is reached).
- Repeat condition (i.e. a metadata or hash associated with the previous iteration).
- Stop condition (i.e. last iteration of the loop).

- 30 The ICTA statement enables the agent to decide on the next transaction to make, based on the current state of the blockchain. Making the next transaction involves broadcasting the transaction onto the Bitcoin network, and writing the new transaction onto the Blockchain.

This acts as a record that this iteration has been executed. Once the transaction has been written onto the Blockchain, the Manager will subsequently find that the previous iteration has been executed and written onto the Blockchain, and will execute the next iteration. The latter continues until the repeat loop exits when the index (i) reaches the limit (J) specified in the code block.

Each transaction is saved in the blockchain in a way that can be reused. In a Bitcoin implementation, each signature in a transaction is appended with a SIGHASH flag. This flag can take on different values, each indicating whether other parts of the transaction can be amended without involvement of the owner of this signature. A reusable transaction has the SIGHASH flag 'SigHash_AnyoneCanPay' in one of the transaction inputs. This permits anyone to contribute to the inputs of the transaction. This parameter enables the agent's ICTA function to be executed and repeated multiple times and with different inputs. Use of the function can be restricted to authorised parties – for example, via copyright of the reusable transaction.

The 'If *condition*' section of the ICTA code block can monitor any type of condition. This is similar to other programming languages (e.g. C, C++, Java) and not limited to information stored on the Blockchain. For example, it could monitor the date and time (i.e. when a certain date and time are reached) or monitor the weather (i.e. when the temperature is below 10 °C and it is raining), monitor the conditions of a contract or a trust (i.e. when company A buys company B).

The 'Then *action*' section of the ICTA code block can execute a number of actions. The invention is not limited with regard to the number or type of actions that can be taken. The action is not limited to a transaction on the Blockchain, although a transaction containing metadata related to the action may be written on the Blockchain.

The metadata can be of any form. However, in one embodiment, the metadata may store a hyperlink to a file containing more data or instructions relating to the action. The metadata may store both a hyperlink to a hash table containing more data or instructions

relating to the action along with a hash of the action that acts as the loop-up key for the hash table.

The agent's control stack can be implemented in a number of ways that are specific to the needs of each user. For example, the repeat loop of the control stack can be based on any Turing Complete language. One possible choice of language is the Forth style stack-based language. An advantage of using this language is that it keeps the control stack consistent in programming style with the Bitcoin scripts which are already known and in wide usage.

10 Using the Bitcoin Script's Alternate Stack as a Data Storage Space

The Bitcoin script contains commands, also called op codes, which enable users to move data onto an alternative stack, known as the 'alt stack'.

The op codes are:

- 15 • OP_TOALTSTACK - which moves data from the top of the main stack onto the top of the alt stack.
 - OP_FROMALTSTACK - which moves data from the top of the alt stack to the top of the main stack.
- 20 This enables data from intermediate steps of calculations to be stored in the alt stack, similar to the 'memory' function which allows data to be stored on the calculator. In one embodiment, the alt stack is used for configuring bitcoin scripts to solve small computation tasks and returning the results in the computation.

25 Using a Code Register to Manage the Agent

The agent also manages a registry of all the codes that it owns and runs. This registry is structured like a lookup table or dictionary that maps a specific key to a specific value. The key and value pair is represented by the hash of the code block (H_1) and the IPv6 address of where the code is stored respectively. To retrieve the code block using the key H_1 , the lookup table is used to retrieve the associated value (this is the location where the code is stored) and retrieves the source code accordingly. The implementation of the code registry can vary.

Transaction Metadata of the Agent's Code, and Re-Spawning of the Loop

Information required to respawn the agent's loop at a particular iteration is stored as metadata in the transaction recorded on the Blockchain.

5

In this way, a transaction on the blockchain stores or provides access to information about a given iteration of the loop which is being executed on the agent. This information can include the values of any variables associated with the loop, such as index i , and any other necessary information such as values for parameters used in the code block or location-related data specifying where further required information can be accessed.

10

The metadata itself is stored as part of a multi-signature pay-to-script-hash script (P2SH) in the transaction. The metadata recorded with the transaction also gives the ability to record an audit trail of how the code has been executed in the past.

15

There are several ways in which the agent could respawn the repeat loop code block at each iteration. The code block might be hard-coded into the agent itself, or could be stored in a private or publicly available file, or stored as an entry on a private or public hash table file, or a combination of the above. The code block could be static with hard-coded variables or could be static but contain parameter(s) that can be populated. The parameters could be single values of any data format, or could be small chunks of code, or be combinations of the above. The parameters could be populated by retrieving them directly from metadata in a transaction (e.g. bitcoin transaction) or from an external source such as an internal database or a private/public file or hash table or any combination of the above. Pointers to the external source of parameter values might be stored in metadata in a transaction.

20

25

The following steps provide one example of how the agent can respawn a repeat loop code block at the i th iteration. In this example, the code registry is a hash table whereby the hash values act as look-up keys for the table and are stored in metadata on transactions.

30

-50-

1. The agent monitors the Blockchain for transactions that contain hashes of the code block that matches entries in the code registry.
2. The agent finds a transaction that contains the corresponding hash (H_1).
3. The agent reads the 'Metadata-CodeHash', gets the CodeHash field to get H_1 and
5 uses it to retrieve the code (C_1). If $\text{RIPEMD-160}(\text{SHA256}(C_1))$ equals H_1 , the code has not been changed and it is safe to proceed to the next step.
4. The agent reads the 'Metadata-CodeHash' which stores the index I , and respawns the code at the i^{th} iteration. In other words, the loop is 'reloaded' at the appropriate iteration
- 10 5. The signature of the User is included in the P2SH command to verify the origin of the metadata.
6. The agent reads the 'Metadata-OutputHash' and 'Metadata-OutputPointer' to retrieve the output of the previous steps, if these data are required for this iteration of the loop.

15

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
20 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
25 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.

30

CLAIMS:

1. A blockchain-implemented control method, the method comprising:
deriving a public-key-private key cryptographic pair for an item or portion of data;
5 deriving a signature for the data using the public key-private key cryptographic pair;
codifying the data to generate codified metadata for the data.
2. A method according to claim 1 and comprising the step of deriving one or both of the
keys in the cryptographic key from an existing cryptographic key pair, by:
10 determining a first entity second private key based on at least a first entity master
private key and a generator value;
determining a second entity second private key based on at least a second entity master
private key and the generator value;
determining a common secret (CS) at the first entity based on the first entity second
15 private key and the second entity second public key, and determining the common
secret (CS) at the second entity based on the second entity second private key and
first entity second public key; and
wherein:
the first entity second public key and the second entity second public key are
20 respectively based on at least the first/second entity master key and the generator
value.
3. A method according to Claim 1 or 2, the method further comprising:
transmitting the codified metadata to the blockchain.
25
4. A method according to any preceding claim, the method further comprising:
receiving a signature and a script from at least one user to enable access to the
embedded data.
- 30 5. A method according to Claim 4 wherein the script comprises a public key of a
signatory.

6. A method according to any preceding claim wherein the metadata comprises a hash of the data.
7. A method according to any preceding claim wherein the metadata comprises a pointer to the data.
8. A method according to Claim 4 wherein the hash is used as a primary key in a lookup table where the data is stored.
9. A method according to any preceding claim and comprising the step of generating or deriving one or both keys in a cryptographic key pair by determining, at a first node (C), a common secret (CS) that is common with the first node (C) and a second node (S), wherein the first node (C) is associated with a first asymmetric cryptography pair having a first node master private key (V_{1C}) and a first node master public key (P_{1C}), and the second node (S) is associated with a second asymmetric cryptography pair having a second node master private key (V_{1S}) and a second node master public key (P_{1S});
wherein the step comprises:
 - determining a first node second private key (V_{2C}) based on at least the first node master private key (V_{1C}) and a deterministic key (DK);
 - determining a second node second public key (P_{2S}) based on at least the second node master public key (P_{1S}) and the deterministic key (DK); and
 - determining the common secret (CS) based on the first node second private key (V_{2C}) and the second node second public key (P_{2S}),wherein the second node (S) has the same common secret (S) based on a first node second public key (P_{2C}) and a second node second private key (V_{2S}), wherein:
 - the first node second public key (P_{2C}) is based on at least the first node master public key (P_{1C}) and the deterministic key (DK); and
 - the second node second private key (V_{2S}) is based on at least the second node master private key (V_{1S}) and the deterministic key (DK).

-53-

10. A method according to claim 9 wherein the deterministic key (DK) is based on a message (M)

11. A blockchain-implemented control system operative to:

- 5 derive a public-key-private-key cryptographic key pair for an item or portion of data;
 derive a signature for the data using the public key-private key cryptographic pair;
 codify the data to generate codified metadata for the data.

12. A system according to Claim 10, the system further operative to:

- 10 transmit the codified metadata to the blockchain.

13. A system according to Claim 11 or Claim 12, the system further operative to:
 receive a signature and a script from at least one user to enable access to the
 embedded metadata.

15

14. A system according to Claim 13, wherein the script comprises a public key of a signatory.

15. A system according to any of Claims 11 to 14, the system further operative to
20 generate a hash of the data and to store that hash as part of the metadata.

16. A system according to any of Claims 11 to 15, the system further operative to
 generate a pointer to the data.

25 17. A system according to Claim 15 wherein the system is further operative to store the
 hash as a primary key in a lookup table where the data is stored.

18. A system according to any of claims 11 to 17, the system is arranged to determine, at a
 first node (C), a common secret (CS) that is common with the first node (C) and a
30 second node (S), wherein the first node (C) is associated with a first asymmetric
 cryptography pair having a first node master private key (V_{1C}) and a first node master
 public key (P_{1C}), and the second node (S) is associated with a second asymmetric

-54-

cryptography pair having a second node master private key (V_{1S}) and a second node master public key (P_{1S}), by:

- determining a first node second private key (V_{2C}) based on at least the first node master private key (V_{1C}) and a deterministic key (DK);

5 - determining a second node second public key (P_{2S}) based on at least the second node master public key (P_{1S}) and the deterministic key (DK); and

- determining the common secret (CS) based on the first node second private key (V_{2C}) and the second node second public key (P_{2S}),

wherein the second node (S) has the same common secret (S) based on a first node second public key (P_{2C}) and a second node second private key (V_{2S}), wherein:

10 - the first node second public key (P_{2C}) is based on at least the first node master public key (P_{1C}) and the deterministic key (DK); and

- the second node second private key (V_{2S}) is based on at least the second node master private key (V_{1S}) and the deterministic key (DK).

15

19. A system according to claim 18 wherein the deterministic key (DK) is based on a message (M)

20. A system according to any of claims 11 to 19, wherein the system is arranged to derive one or both of the keys in the cryptographic key pair from an existing cryptographic key pair, by:

determining a first entity second private key based on at least a first entity master private key and a generator value;

25 determining a second entity second private key based on at least a second entity master private key and the generator value;

determining a common secret (CS) at the first entity based on the first entity second private key and the second entity second public key, and determining the common secret (CS) at the second entity based on the second entity second private key and first entity second public key; and

30 wherein:

-55-

the first entity second public key and the second entity second public key are respectively based on at least the first/second entity master key and the generator value.