

System and Method For Multi-Party Generation Of Blockchain-Based Smart Contract

Field of Invention

This invention relates generally to the execution of smart contract between multiple (e.g., more than two) parties, and more particularly to implementations in which a verification
5 key for a smart contract is collectively generated by two or more parties of a smart contract and a third party (e.g., a worker node on a blockchain network) is utilised to execute the smart contract in a computationally verifiable manner. The third computing entity may generate a proof of correct execution of the smart contract, which can be used to unlock digital assets encumbered by the first computing entity and the second computing entity.
10 The invention is particularly suited, but not limited to, use in a blockchain network such as a Bitcoin-based blockchain network.

Background of Invention

A blockchain may refer to a peer-to-peer, electronic ledger which is implemented as a computer-based decentralised, distributed system made up of blocks which in turn may be
15 made up of transactions and other information. In some examples, a “blockchain transaction” refers to an input message encoding a structured collection of field values comprising data and a set of conditions, where fulfilment of the set of conditions is prerequisite for the set of fields to be written to a blockchain data structure. For example, with Bitcoin each transaction is a data structure that encodes the transfer of control of a
20 digital asset between participants in the blockchain system, and includes at least one input and at least one output. In some embodiments, a “digital asset” refers to binary data that is associated with a right to use. Examples of digital assets include Bitcoin, ether, and Litecoins. In some implementations, transferring control of a digital asset can be performed by reassociating at least a portion of a digital asset from a first entity to a second
25 entity. Each block of the blockchain may contain a hash of the previous block to that blocks become chained together to create a permanent, unalterable record of all transactions which have been written to the blockchain since its inception. Transactions contain small programs known as scripts embedded into their inputs and outputs, which specify how and by whom the outputs of the transactions can be accessed. On the Bitcoin
30 platform, these scripts are written using a stack-based scripting language.

Although blockchain technology is most widely known for the use of cryptocurrency implementation, digital entrepreneurs have begun exploring the use of both the cryptographic security system Bitcoin is based on and the data that can be stored on the Blockchain to implement new systems. It would be highly advantageous if the blockchain
5 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.

The present disclosure describes technical aspects of one or more blockchain-based
10 computer programs. A blockchain-based computer program may be a machine readable and executable program recorded in a blockchain transaction. The blockchain-based computer program may comprise rules that can process inputs in order to produce results, which can then cause actions to be performed dependent upon those results. One area of current research is the use of blockchain-based computer programs for the implementation
15 of “smart contracts”. Unlike a traditional contract which would be written in natural language, smart contracts may be computer programs designed to automate the execution of the terms of a machine-readable contract or agreement.

Summary of Invention

Thus, it is desirable to provide a protocol for multi-party verification key recording on a
20 blockchain by exchanging quantities that can be used to determine powers of a shared secret between two or more parties. In various embodiments, it may be desirable for two or more parties of a smart contract to exchange quantities that are usable to determine a common reference string that comprises a verification key and an evaluation key. In various embodiments, the techniques described herein allow two or more parties to
25 exchange powers of a shared secret without the use of cryptographic techniques such as encryption, and furthermore does not require the parties to establish a communications channel that requires cryptographically verifiable assurances confidentiality of data exchanged over said communications channel.

Such an improved solution has now been devised.

Thus, in accordance with the present invention there are provided systems and methods as defined in the appended claims.

In accordance with the invention there may be provided a computer-implemented method for a node of a blockchain network, the computer-implemented method comprising, at a
5 first computing entity: determining, based at least in part on a first polynomial and at least two elliptic curve points, a set of elliptic curve points for a second computing entity; making a subset of the set of elliptic curve points available to the second computing entity; receiving a second set of elliptic curve points generated using a second polynomial; determining a power of a secret based at least in part on the first set and the second set;
10 determining, based at least in part on the power of the secret, a common reference string comprising a verification key and an evaluation key, wherein the common reference string is also determinable by the second computing entity as a result of the first computing entity providing the subset to the second computing entity; and generating a smart contract comprising a first transaction input provided by the first computing entity and a second
15 transaction input provided by the second computing entity, wherein correct execution of the smart contract by a third computing entity results in the third computing entity being able to generate a blockchain transaction using an output of the smart contract.

The set of elliptic curve points as described above may comprise corresponding elliptic curve points for powers of the first polynomial (e.g., an elliptic curve point for each
20 polynomial power).

Preferably, the first polynomial may be of at least order 2.

Preferably, the subset described above is the set of elliptic curve points. Furthermore, unless otherwise noted or contradicted by context, the term “subset” of a corresponding set does not necessarily denote a proper subset of the corresponding set, but the subset and the
25 corresponding set may be equal.

The secret may be shared between the first computing entity and the second computing entity without using a cryptographically protected communications channel.

The first computing entity and the second computing entity may collectively determine both the first digital asset and the second digital asset.

Preferably, some or all of the methods described herein may further comprise: determining, based on a third polynomial and the at least two elliptic curve points, a third set of elliptic curve points for the second computing entity; making a second subset of the third set of elliptic curve points available to the second computing entity; receiving a fourth set of
5 elliptic curve points; determining a parameter based at least in part on the third set and the fourth set, the parameter also determinable by the second computing entity as a result of the first computing entity providing the second subset to the second computing entity; and wherein the determining of the common reference string is based further at least in part on the parameter.

10 Preferably, some or all of the methods described herein may further comprise sharing an elliptic curve parameter between the first computing entity and the second computing entity using Shamir's Secret Sharing Scheme.

Preferably, some or all of the methods described herein may further comprise exchanging a scalar parameter between the first computing entity and the second computing entity using
15 a Diffie-Hellman scheme (e.g., using a Diffie-Hellman key exchange algorithm).

The smart contract may comprise a Pay-To-Script-Hash (P2SH) type unlocking script that allows the third computing entity to unlock both the first digital asset and the second digital asset in response to providing a valid proof of correct execution.

The first computing entity may make the subset of the set of elliptic curve points available
20 to the second computing entity via an off-chain communications channel.

The second polynomial may be inaccessible to the first computing entity.

Preferably, the at least two elliptic curve points are two different elliptic curve points.

It is also desirable to provide a system, comprising: a processor; and memory including executable instructions that, as a result of execution by the processor, causes the system to
25 perform any of the methods as claimed.

It is also desirable to provide a non-transitory computer-readable storage medium having stored thereon executable instructions that, as a result of execution by one or more

processors of a computer system, cause the computer system to at least perform any of the methods as claimed.

Brief Description of the Drawings

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

Figure 1 illustrates a computing environment in which multiple parties of a smart contract enter into an agreement with a third party to execute the smart contract;

10 Figure 2 illustrates a computing environment where a first computing entity and a second computing entity exchange quantities that can be used to determine powers of a shared secret between two or more parties;

Figure 3 illustrates a computing environment in which a first computing entity and a second computing entity exchange a set of parameters that renders a zero-knowledge;

15 Figure 4 illustrates a diagram of a protocol based on a two-party common reference string (CRS) and corresponding proof-of-correctness (POC) or proof of correct execution;

Figure 5 illustrates a process for generating a two-party common reference string comprising a verification key and evaluation key, in accordance with an embodiment

Figure 6 illustrates a process for sharing powers of a shared secret between multiple
20 parties, in accordance with at least one embodiment; and

Figure 7 illustrates a simplified block diagram of a computing device that can be used to practice at least one embodiment of the present disclosure.

Detailed Description

Figure 1 illustrates a blockchain environment in which various embodiments can be
25 implemented;

This disclosure describes techniques that may be utilised to implement systems and methods that allow multiple parties to securely share elements of a group, whose exponent or multiplicative coefficient depends on powers of a shared secret. Quantities may be shared such that the multiple parties exchange quantities that are based on the shared secret
5 (e.g., powers of the shared secret) without exposing the shared secret. Accordingly, in various embodiments, a plurality of n participants establishes a representation of powers (e.g., $\langle s^i \rangle_G$ in the multiplicative case) of a shared secret.

In an embodiment, a protocol utilising techniques and methods described herein are used by two parties of a smart contract to share quantities that can be used by the parties to
10 determine powers of a shared secret without sharing the secret itself and without revealing information that would allow another computing entity (e.g., a computing entity that is not a party to the smart contract) to determine the secret. In one embodiment, the protocol comprises a first party of a smart contract computing a first set of parameters that are transmitted to a second party of the smart contract, the second party computing a second
15 set of parameters and submitting those parameters to the first party, wherein, upon the exchange of the parameters as described above, both parties are able to calculate the identical common reference string, which comprises a verification key. The parties may then agree on a transaction in which they make proportionate contributions of digital assets to the smart contract which are locked to an address (e.g., address of a worker node of a
20 blockchain network) and can be unlocked (e.g., spent). In an embodiment, off-chain communications between the parties of the smart contract are limited to the exchange of the parameters used to generate the common reference string, while ensuring security is maintained (e.g., the secret value is not divulged or otherwise determinable, based on the parameters exchanged, by an adversary or other computing entity that is not a party to the
25 smart contract). In an embodiment, the two parties (or, in the more general case, two or more parties) utilise techniques that share powers of a shared secret, such as in the manner described elsewhere in this document, for example, in connection with Figures 1-7.

References may be made to figure 1, which illustrates an example computing environment
100 in which various embodiments of the present disclosure may be practiced. Systems
30 and methods described herein may relate to a protocol for parties of a smart contract to exchange quantities that the first and second computing entities can use to calculate the

identical common reference string. Figure 1 illustrates a computing environment 100 comprising a first computing entity 102 and a second computing entity 104 that exchange a set of parameters that, allow both the first computing entity and the second computing entity to determine a common reference string 108. The common reference string may be
5 utilised by the parties to generate a smart contract 110 that locks digital assets which either or both of the computing entities contributes as transaction inputs. The common reference string may comprise an evaluation key 112 and a verification key 112. The smart contract 110 may be published to a blockchain, such as the blockchain 118 illustrated in Figure 1. The smart contract 110 may be executed by a third computing entity 106 that is not a party
10 to the smart contract 106. As part of or in association with executing the smart contract, the third computing entity (e.g., worker) may generate a proof of correct execution 116 of the smart contract based at least in part on the evaluation key of the common reference string. The proof of correct execution 116 may be computationally verifiable by any suitable computing system (e.g., parties of the smart contract or a node of the blockchain network
15 that acts as a verifier node). In an embodiment, the verification key 114 is utilised by a fourth computing entity (e.g., verifier computer system) to verify the proof that is published to the blockchain network 118 is correct.

The first computing entity 102 and the second computing entity 104 are computer systems that are parties to a smart contract, according to at least one embodiment. Parties of a smart
20 contract may refer to two or more computing entities that agree (e.g., pursuant to user input provided through associated user input devices) on the terms to the execution of a smart contract. The first computing entity 102 and the second computing entity 104 may both agree upon a smart contract and contribute transaction inputs to the smart contract such that the respective transaction inputs of the smart contract are encumbered by a locking
25 script that can be unlocked (e.g., spent) as a result of a worker node providing a proof of correct execution of the smart contract. Systems and methods described herein relate to enabling a locking script to secure the verification key V_K from alteration and checking validity of a proof π , thereby allowing execution of a zero-knowledge protocol on a blockchain during transaction validation.

30 In various embodiments, the first computing entity 102 and the second computing entity 104 may agree upon a smart contract, in an embodiment, by exchanging a set of messages

that encode parameters for the smart contract, such as dates, times, conditions, and actions (e.g., transfer of control of digital assets) that are used to control the execution of the smart contract. For example, a smart contract (e.g., an executable program) may insure a party against delays of a particular flight, and execution of the program may include determining
5 whether the particular flight was delayed using external data such as flight information of a particular commercial flight on a particular day. If the flight was delayed, a party of the program may receive a transfer of assets (e.g., a smart contract that provides travel insurance against delays).

In an embodiment, the smart contract 110 is encoded in a high-level programming
10 language as source code such as C, C++, or Java. These are merely illustrative examples and the smart contract may be encoded using other suitable programming languages. In an embodiment, software such as a compiler, interpreter, and/or assembler may be utilized to transform the smart contract 110 to an arithmetic circuit which consists of “wires” that carry values from a field F and connect to addition and multiplication gates. It should be
15 noted that the arithmetic circuit may refer to a logical circuit that can be implemented by a physical circuit comprising a series of physical gates (e.g., using transistor–transistor logic (TTL) integrated circuits such as 7400-series gates, flip-flops, buffers, decoders, multiplexers, and the like) connected by physical wires. While the execution of a smart contract 110 is described in the context of Figure 1 and elsewhere, the use of a smart
20 contract is merely one non-limiting example of source code that can be transformed to an arithmetic circuit. In an embodiment, a client (e.g., the first computing entity 102, either alone or in conjunction with the second computing entity 104) determines source code for performing a task defined by a set of operations, wherein execution of the task is delegated to a third computing entity 106 (referred to as a worker or prover). Generally speaking, a
25 verifier may perform tasks associated with determining that the prover executed the task correctly, such as by verifying the validity of a proof of correct execution 116 generated by the prover.

A verifiable computation is a technique that allows the generation of proofs of computation. In an embodiment, such a technique is utilized by a client to outsource, to another
30 computing entity referred to herein as a prover, the evaluation of a function f on an input x . In some cases, the client is computationally limited so that it is infeasible for the client to

perform the evaluation of the function (e.g., the expected runtime of the calculation using computing resources available to the client exceeds a maximum acceptable threshold), although such need not be the case, and the client may, generally, speaking, delegate evaluation of the function f on the input x based on any suitable criterion, such as

- 5 computational runtime, computational cost (e.g., the financial cost of allocating computing resources to perform the evaluation of the function), and more.

A prover, in an embodiment, is any suitable computing entity such as a blockchain node as described in greater detail elsewhere in the present disclosure. In an embodiment, a prover (e.g., a blockchain node) evaluates the function f on input x and generates an output y and
10 a proof π of the correctness of the output y that can be verified by other computing entities such as the client as described above and/or other nodes of the blockchain network. Proofs, which may also be referred to as arguments, can be verified faster than doing the actual computational — accordingly, computational overhead can be reduced (e.g., reducing power overhead and the cost associated with powering and running computing resources)
15 by verifying the correctness of the proof instead of re-computing the function f over input x to determine the correctness of the output generated by the prover described above. In zero-knowledge verifiable computation the prover provides an attestation to the client that the prover knows an input with a particular property.

- 20 An efficient variant of a zero-knowledge proof of knowledge is zk_SNARK (Succinct Non-interactive ARgument of Knowledge). In an embodiment, all pairings-based zk-SNARKs include a process where the prover computes a number of group elements using generic group operations and the verifier checks the proof using a number of pairing product equations. In an embodiment, the linear interactive proof works over a finite field
25 and the prover's and verifier's message include, encode, reference, or otherwise include information usable to determine vectors of field elements.

In an embodiment, the first computing entity and/or the second computing entity agree on terms of execution of a smart contract by exchanging a set of messages that encode proposed parameters for the execution of the smart contract, such as one or more Boolean
30 expressions that encodes a set of conditions that determine whether and/or how to execute the smart contract and a set of operations to perform based on a condition being satisfied. In an embodiment, one computing entity sends a set of parameters to the second computing

- entity as part of a protocol, and the second computing entity determines whether the parameters are acceptable for the smart contract. If the parameters are not accepted, the second computing entity may provide a different set of parameters to the first computing entity as a second proposed set of parameters for execution of the smart contract. The
- 5 second computing entity may also provide a signal that the first set of parameters were not acceptable, and the first computing entity determines a second set of parameters to provide. In either case, once all parties have signalled agreement to the parameters, either computing entity can, in an embodiment, generate a locking transaction wherein one of the outputs is locked by the program (e.g., a smart contract script) and sends it to a
- 10 counterparty of the smart contract. The locking transaction may refer to a transaction that initialises constraints upon which an unlocking transaction can be validated. In some examples, an “unlocking transaction” refers to a blockchain transaction that reassociates (e.g., transferring ownership or control) at least a portion of a digital asset, indicated by an UTXO of a previous transaction, to an entity associated with a blockchain address.
- 15 In an embodiment, the first computing entity generates a locking transaction and adds a transaction input that covers a portion of the worker fee. It should be noted that at this point, the locking transaction is not yet valid because the value of the transaction inputs is not equal to the value of the transaction outputs of the locking transaction. Continuing with the example, when the second computing entity receives the locking transaction, the
- 20 second computing entity verifies the smart contract (e.g., verifies the common reference string and parameters for execution of the smart contract) and adds an input to the locking transaction and unlocks a UTXO to transfer to the issuer agreed upon digital assets and also a fee that is to be paid to the worker for execution of the program (e.g., smart contract) and an output that has a value of the fee to the worker. In cases where both the first
- 25 computing entity and the second computing entity contribute transaction inputs to the smart contract, the smart contract may be jointly owned by both parties, and the transfer (e.g., exchange or sale) of the smart contract may require an attestation from both parties.

The smart contract 110 may be executed by a third computing entity 106 such as a node of a blockchain network. The third computing entity 106 may be referred to as a worker or a

30 prover. In an embodiment, the worker executes the smart contract by at least performing a computational task that involves the computation of a function on an input. In an

embodiment, the worker is any suitable computer system that the owner(s) of the smart contract may delegate a computational task to. An input, in an embodiment, includes information that attests to the worker's identity, such as a digital signature generated using a private key associated with the worker. In an embodiment, the worker is a computer
5 system that the first and second computing entities agree to transfer digital assets to in return for successfully completing a computational task. The owner(s) of the smart contract, in an embodiment provides an input x and the evaluation key E_K 112 to a prover, the prover uses an evaluation module to a compute routine to compute the output y (i.e., $y=f(x)$ wherein the input is x and the function is f) and uses the evaluation key E_K to produce a
10 proof of correct execution 116, which may also be referred to as a proof-of-correctness elsewhere in this disclosure. In an embodiment, the worker is a computer system comprising hardware and/or software that includes instructions that, if executed by one or more processors of the computer system, cause the computer system to evaluate the values of the internal circuit wires of a QAP and produce an output y of the QAP.

15 In embodiments, an output y , values of the internal circuit wires (or a subset thereof), and the evaluation key E_K are used to produce the proof-of-correctness. The proof π can be stored on the blockchain and verified by multiple parties without requiring the worker to separately interact with the multiple parties. In this manner, a fourth computing entity (e.g.,
20 a verifier computer system) can validate the broadcasted transaction using the public verification key V_K 114 and the proof π , thereby validating the smart contract. In some cases, the owner(s) of the smart contract may reclaim digital assets encumbered by the broadcasted transaction if the verification fails. In some cases, the owner(s) of the smart contract can perform the verification of the proof.
25 In an embodiment, the verification key 114 and the corresponding proof 116 are generated according to techniques described above and/or below. Accordingly, a verifier is given verification key V_K and proof π :

$$V_K = \left\{ \begin{array}{c} \mathcal{P} \\ \mathcal{Q} \\ \alpha_v \mathcal{Q} \\ \alpha_w \mathcal{Q} \\ \alpha_w \mathcal{P} \\ \alpha_y \mathcal{Q} \\ \beta \mathcal{P} \\ \beta \mathcal{Q} \\ r_y t(s) \mathcal{P} \\ r_v v_i(s) \mathcal{P} \\ r_w w_i(s) \mathcal{Q} \\ r_y y_i(s) \mathcal{P} \end{array} \right\}_{i=0..N}$$

$$Proof \ \pi = \left\{ \begin{array}{c} \sum_{i=N+1}^m a_i r_v v_i(s) \mathcal{P} \\ \sum_{i=N+1}^m a_i \alpha_v r_v v_i(s) \mathcal{P} \\ \sum_{i=N+1}^m a_i r_w w_i(s) \mathcal{Q} \\ \sum_{i=N+1}^m a_i \alpha_w r_w w_i(s) \mathcal{P} \\ \sum_{i=N+1}^m a_i r_y y_i(s) \mathcal{P} \\ \sum_{i=N+1}^m a_i \alpha_y r_y y_i(s) \mathcal{P} \\ \sum_{i=N+1}^m a_i (r_v \beta v_i(s) + r_w \beta w_i(s) + r_y \beta y_i(s)) \mathcal{P} \\ \sum_{i=0}^d h_i s^i \mathcal{Q} \end{array} \right\}$$

such that the verifier computes a plurality of elliptic curve multiplications (e.g., one for each public input variable) and five pair checks, one of which includes an additional pairing multiplication.

Given verification key V_K , proof π , and $(a_1, a_2, \dots, a_N)$, to verify that $t(x)$ divides $p(x)$ and hence $(x_{N+1}, \dots, x_m) = f(x_0, \dots, x_N)$, the verifier proceeds as follows. First it checks all the three α terms:

$$\begin{aligned} 10 \quad e(\alpha_v r_v V_{\text{mid}}(s) \mathcal{P}, \mathcal{Q}) &= e(r_v V_{\text{mid}}(s) \mathcal{P}, \alpha_v \mathcal{Q}) \\ e(\alpha_w r_w W_{\text{mid}}(s) \mathcal{P}, \mathcal{Q}) &= e(\alpha_w \mathcal{P}, r_w W_{\text{mid}}(s) \mathcal{Q}) \\ e(\alpha_y r_y Y_{\text{mid}}(s) \mathcal{P}, \mathcal{Q}) &= e(r_y Y_{\text{mid}}(s) \mathcal{P}, \alpha_y \mathcal{Q}) \end{aligned}$$

wherein $V_{\text{mid}}(s) = \sum_{i=N+1}^m a_i v_i(s)$, $W_{\text{mid}}(s) = \sum_{i=N+1}^m a_i w_i(s)$, and $Y_{\text{mid}}(s) = \sum_{i=N+1}^m a_i y_i(s)$. Then, the verifier checks the term β :

$$e(r_v V_{\text{mid}}(s)\mathcal{P} + r_y Y_{\text{mid}}(s)\mathcal{P}, \beta Q) \cdot e(\beta \mathcal{P}, r_w W_{\text{mid}}(s)Q) = e(Z_{\text{mid}}(s)\mathcal{P}, Q)$$

and $Z_{\text{mid}}(s) = \sum_{i=N+1}^m a_i (r_v \beta v_i(s) + r_w \beta w_i(s) + r_y \beta y_i(s))$. Finally, the verifier

5 checks the divisibility requirement:

$$e(r_v V(s)\mathcal{P}, r_w W(s)Q) = e(r_y Y(s)\mathcal{P}, Q) \cdot e(r_y t(s)\mathcal{P}, h(s)Q)$$

wherein $r_v V(s)\mathcal{P} = \sum_{i=0}^m r_v a_i v_i(s)\mathcal{P}$, $r_w W(s)Q = \sum_{i=0}^m r_w a_i w_i(s)Q$, $r_y Y(s)\mathcal{P} = \sum_{i=0}^m r_y a_i y_i(s)\mathcal{P}$, and $h(s)Q = \sum_{i=0}^d h_i Q$.

10 Thus, upon considering the notation from the sections described above and the examples described in this disclosure, the verification comprises a set of pair checks of the following elements, in accordance with one embodiment:

$$e(\pi_2, V_K^2) = e(\pi_1, V_K^3)$$

$$e(\pi_4, V_K^2) = e(V_K^5, \pi_3)$$

$$15 \quad e(\pi_6, V_K^2) = e(\pi_5, V_K^6)$$

$$e((\pi_1 + \pi_6), V_K^2) = e(\pi_7, V_K^2)$$

$$\begin{aligned} & e\left((a_0 V_K^{10} + a_1 V_K^{11} + a_2 V_K^{12} + a_3 V_K^{13} + a_4 V_K^{14} + \pi_2 + a_7 V_K^{15}), (a_0 V_K^{16} + a_1 V_K^{17} \right. \\ & \quad \left. + a_2 V_K^{18} + a_3 V_K^{19} + a_4 V_K^{20} + \pi_4 + a_7 V_K^{21})\right) \\ & = e\left((a_0 V_K^{22} + a_1 V_K^{23} + a_2 V_K^{24} + a_3 V_K^{25} + a_4 V_K^{26} + \pi_6 + a_7 V_K^{15}), V_K^2\right) \\ & * e(V_K^9, \pi_8) \end{aligned}$$

20

Figure 2 illustrates a computing environment 200 where a first computing entity 202 and a second computing entity 204 exchange quantities that can be used to determine powers of a shared secret between two or more parties. The first computing entity 202 and the second computing entity 204 may exchange quantities (as illustrated below the horizontal arrow
25 illustrated in Figure 2) that are used to calculate an identical common reference string. In an embodiment, the first computing entity and the second computing entity are nodes of a blockchain network that are in accordance with those described in connection with Figure 1. In accordance with at least one embodiment, let $F: \mathbb{F} \rightarrow \mathbb{F}$ be a function on field, and let $\mathcal{C}$ be the corresponding arithmetic circuit, and $Q = (t(x), \mathcal{V}, \mathcal{W}, \mathcal{Y})$ the

corresponding QAP (quadratic arithmetic program) of size m and degree d . Furthermore, let $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear mapping, and G a generator of $\mathbb{G}$. In an embodiment, the additive representation (as opposed to the exponential representation) is chosen. In an embodiment, an evaluation key and a verification key are chosen by: choosing random

5 $r_v, r_w, s, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma \in \mathbb{F}$, and setting $r_y = r_v \cdot r_w$, $r_v \cdot G = \langle g_v \rangle$ (generally: $\langle a \rangle = a \cdot G$), and correspondingly $\langle g_w \rangle, \langle g_y \rangle$, to construct the keys:

$$EK = (\{ \langle v_k(s) \rangle \}_{k \in I_{mid}}, \{ \langle w_k(s) \rangle \}_{k \in I_{mid}}, \{ \langle y_k(s) \rangle \}_{k \in I_{mid}}, \{ \langle \alpha_v v_k(s) \rangle \}_{k \in I_{mid}}, \{ \langle \alpha_w w_k(s) \rangle \}_{k \in I_{mid}}, \\ \{ \langle \alpha_y y_k(s) \rangle \}_{k \in I_{mid}}, \{ \langle s^i \rangle \}_{i \in [d]}, \{ \langle \beta v_k(s) \rangle \langle \beta w_k(s) \rangle \langle \beta y_k(s) \rangle \}_{k \in I_{mid}})$$

10 and

$$VK = (\langle 1 \rangle, \langle \alpha_v \rangle, \langle \alpha_w \rangle, \langle \alpha_y \rangle, \langle \gamma \rangle, \langle \beta \gamma \rangle, \langle t(s) \rangle, \{ \langle v_k(s) \rangle \langle w_k(s) \rangle \langle y_k(s) \rangle \}_{k \in \{0\} \cup [N]}).$$

where $N = N_{in} + N_{out}$, i.e., the number of in- and outputs. In an embodiment where an asymmetric pairing is considered, the pairing mapping would be defined as: $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, and G_i a generator of $\mathbb{G}_i$, $i = 1, 2$. In that case the CRS would change slightly and the

15 VK would be

$$VK = (\langle 1 \rangle_1, \langle 1 \rangle_2, \langle \alpha_v \rangle_2, \langle \alpha_w \rangle_2, \langle \alpha_w \rangle_1, \langle \alpha_y \rangle_2, \langle \beta \rangle_1, \langle \beta \rangle_2, \langle r_y t(s) \rangle_1, \{ \langle v_k(s) \rangle_1 \langle w_k(s) \rangle_2 \langle y_k(s) \rangle_1 \}_{k \in \{0\} \cup [N]})$$

where $r_y = r_v \cdot r_w$ and $\langle a \rangle_i = a \cdot G_i$.

As indicated, circuits are described by means of polynomials v, w , which, in turn, are

20 evaluated in a secret s , which is only known to the party owning/creating the circuits and corresponding QAP (e.g., the owners of the smart contract).

More precisely, as described above, the client generates the elements:

$$r_v, r_w, s, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma \xleftarrow{\mathcal{R}} \mathbb{F}_T^*$$

Whereas the security of the proposed solution relies on the parameter, s , in some

25 embodiments, exposing the remaining $(r_v, r_w, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma)$ may reveal information that

does not render the system zero-knowledge and/or that the client does not want other entities to know about.

In an embodiment, for solutions where a worker is required to provide a proof-of-correctness, there may exist an OP_code (or equivalents) for verifying the proof-of-
5 correctness against a verification key.

It should be noted that throughout this disclosure, unless otherwise stated, the polynomials in this paper are defined over a field, $\mathbb{F}_r$. Thus, let $(\mathbb{F}_r, +, \cdot)$ be a field. Accordingly, a polynomial over $\mathbb{F}_r$ is given by $P: \mathbb{F} \rightarrow \mathbb{F}'$

$$P(x) = \sum_{i=0}^n a_i x^i$$

10 where $a_i \in \mathbb{F}$.

In an embodiment, the common reference string is expressed by means of the polynomials $v(x), w(x)$ evaluated in a secret, s , expressed in the form:

$$v(s) = a_0 + a_1 s + a_2 s^2 + \dots + a_n s^n$$

$$w(s) = b_0 + b_1 s + b_2 s^2 + \dots + b_n s^n$$

15 In an embodiment, techniques described herein are utilised to determine and share elliptic curve points of form: $\langle v(s) \rangle_G = \langle a_0 \rangle_G + \langle a_1 s \rangle_G + \langle a_2 s^2 \rangle_G + \dots$, for a generator G of the associated group (e.g., of elliptic curve points). Thus, in an embodiment, systems and methods described herein are utilised to determine and distribute $s^r G = \langle s^r \rangle_G$, for arbitrary integer powers r .

20 Techniques to share and distribute $\langle s^r \rangle_G$, in accordance with at least one embodiment, are illustrated in Figure 2. As an example, the $n = 2$ case is described in greater detail below in connection with Figure 2, and should be considered a non-limiting example of sharing powers of secrets between parties of a smart contract. Furthermore, it is noted, that in various embodiments described herein, the equivalent of a threshold is given *a priori* and it
25 is assumed that the necessary number of participants first agree.

Figure 2 illustrates techniques to share and distributed powers of a shared secret in the case of two participants, in accordance with at least one embodiment. As illustrated in Figure 2 and in accordance with at least one embodiment, exactly two parties are participants that share powers of a shared secret (i.e., $n = 2$ case). The first and second computing entities
 5 may respectively be referred to as A and B. In an embodiment, A and B may exchange the following information: A sends $\langle p_1(x_i) \rangle_G$ to B, and receives $\langle p_2(x_i) \rangle_G$ in return ($i \in \{1,2\}$). In this way, both can calculate

$$\langle p(x_i) \rangle_G = \langle p_1(x_i) + p_2(x_i) \rangle_G$$

Using Lagrange interpolation (Lagrange Polynomial, n.d.) it is possible to express p in
 10 terms of $p(x_1)$ and $p(x_2)$:

$$p(x) = p(x_1) \frac{x - x_2}{x_1 - x_2} + p(x_2) \frac{x - x_1}{x_2 - x_1} = \frac{p(x_1) - p(x_2)}{x_1 - x_2} x + \frac{-p(x_1)x_2 + p(x_2)x_1}{x_1 - x_2}$$

and, by extension, $\langle p(x) \rangle_G$ by means of $\langle p(x_i) \rangle_G$ (see WP0559). While the participants cannot reconstruct $p(x)$, they can reconstruct $\langle p(x) \rangle_G$ (and in particular $\langle p(0) \rangle_G$), by means of the exchanged points $\langle p_i(x_j) \rangle_G$. This is true about higher powers of $\langle p^n(x) \rangle_G$ as
 15 well. As a result of the multinomial formula (Multinomial Theorem, n.d.):

$$\langle p^n(x_j) \rangle_G = \langle [p_1(x_j) + \dots + p_m(x_j)]^n \rangle_G = \langle \sum_{k_1+k_2+\dots+k_m=n} \binom{n}{k_1, k_2, \dots, k_m} \prod_{t=1}^m p_t^{k_t}(x_j) \rangle_G$$

For $m = 2$ this becomes:

$$\langle p^n(x_j) \rangle_G = \langle [p_1(x_j) + p_2(x_j)]^n \rangle_G = \langle \sum_{k_1+k_2=n} \binom{n}{k_1, k_2} \prod_{t=1}^2 p_t^{k_t}(x_j) \rangle_G$$

which, in turn, yields the following “tower” of terms (here $i = 1,2$):

n	Terms that need to be exchanged (multiplied by G, and $j = 1, 2$)
-----------------------	--

n	Terms that need to be exchanged (multiplied by G , and $j = 1, 2$)
1	$p_1(x_j), p_2(x_j)$
2	$p_1(x_j), p_2(x_j), p_1^2(x_j), p_2^2(x_j)$ (optionally: $p_1(x_j) \cdot p_2(x_j)$)
3	$p_1(x_j), p_2(x_j), p_1^2(x_j), p_2^2(x_j), p_1^3(x_j), p_2^3(x_j)$ (optionally: $p_1(x_j) \cdot p_2(x_j), p_1^2(x_j) \cdot p_2(x_j), p_1(x_j) \cdot p_2^2(x_j)$)
$\vdots$	...

Schematically the exchange could look like the following (for some x_i):

Step	$A \leftrightarrow B$
1	A: for $j = 1, 2$ and $k = 1, 2, \dots$: send $\{\langle p_1^k(x_j) \rangle_G\}$ to B
2	B: for $j = 1, 2$ and $k, k' = 1, 2, \dots$: send $\{\langle p_2^{k'}(x_j) \rangle_G\}$, (and optionally) $\{\langle p_1^k(x_j) \cdot p_2^{k'}(x_j) \rangle_G\}$ to A

After this exchange (following certain, pre-arranged conventions) both parties can calculate $\langle p^n(x_j) \rangle_G$ and, in particular, $\langle p^n(0) \rangle_G = \langle s^n \rangle_G$.

Figure 3 illustrates a computing environment 300 in which a first computing entity 302 and a second computing entity 304 exchange a set of parameters that renders a protocol — such as those described in connection with Figure 1 — zero-knowledge. In accordance with various embodiments, a public verification key may take on the form:

$$VK = (\langle 1 \rangle_1, \langle 1 \rangle_2, \langle \alpha_v \rangle_2, \langle \alpha_w \rangle_2, \langle \alpha_w \rangle_1, \langle \alpha_y \rangle_2, \langle \beta \rangle_1, \langle \beta \rangle_2, \langle r_y t(s) \rangle_1, \{\langle v_k(s) \rangle_1 \langle w_k(s) \rangle_2 \langle y_k(s) \rangle_1\}_{k \in \{0\} \cup [N]})$$

Whereas the security of the proposed solution relies on the parameter, s , in some embodiments, exposing the remaining $(r_v, r_w, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma)$ may reveal information that does not render the system zero-knowledge and/or that the client does not want other entities to know about. Accordingly, in an embodiment, some or all of the remaining
 5 parameters used to generate a verification key 306 are shared using techniques described in connection with Figure 3.

In an embodiment, polynomials are exchanged between the first computing entity 302 and the second computing entity 304 (which may be referred to as, respectively, A and B) according to techniques described elsewhere in this disclosure, such as those discussed in
 10 connection with Figures 1, 2 and 4. Accordingly, in an embodiment, the first computing entity 302 computes and shares a set of elliptic curve points $\{\langle p_1^k(x_l) \rangle_{G_i}\}_{k \in \{1, \dots, n\}, l \in \{1, 2\}, i \in \{1, 2\}}$ to the second computing entity 304 and the second computing entity 304 computes and shares a set of elliptic curve points $\{\langle p_2^k(x_l) \rangle_{G_i}\}_{k \in \{1, \dots, n\}, l \in \{1, 2\}, i \in \{1, 2\}}$ to the first computing entity 302. Additionally, other
 15 parameters used to generate the verification key 306 may not be revealed in a plaintext format across a communications channel and instead transmitted using techniques described hereinbelow.

Let G_i be a generator of $\mathbb{G}_i$, $i = 1, 2$, and $\langle p_k(s) \rangle_i = r_p \cdot p_k(s) \cdot G_i$ for $p = v, w, y$, according to an embodiment. Further, in an embodiment, $\langle p_k^i(s) \rangle_{G_i}$ are shared, and the
 20 other parameters are propagated in a manner that ensures the confidentiality of the parameters (e.g., encryption or other concealment techniques). It is noted that either of the participants may, conversely, generate the parameters $r_v, r_w, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma \in \mathbb{F}$, and transfer them to the other, either off- or online (e.g., on-chain). Regarding the former — that is, ensuring confidentiality of the other parameters — various techniques may be
 25 utilized. For example, by using Shamir's Secret Sharing Scheme (4S) to share the elements $\langle \alpha_v \rangle_2, \langle \alpha_w \rangle_2, \langle \alpha_w \rangle_1, \langle \alpha_y \rangle_2$ (and possibly $\langle \beta \rangle_1, \langle \beta \rangle_2$, depending on implementation/protocol) and/or using a Diffie-Hellman like secret generation for r_v, r_w, β, γ .

In an embodiment, the elements $\langle \alpha_v \rangle_2, \langle \alpha_w \rangle_2, \langle \alpha_w \rangle_1, \langle \alpha_y \rangle_2, \langle \beta \rangle_1, \langle \beta \rangle_2$, or some combination thereof are of the form $\langle a \rangle_i = a \cdot G_i$. Like in the case of s , each participant, i ,
 30 generates a polynomial, q_i , evaluates it in $x_j, j \in \{1, \dots, m\}$, and shares the corresponding

$q_i(x_j)$ with participant j . Each participant can thus determine $\langle q(x) \rangle_{G_i} = \sum_j \langle q_j(x) \rangle_{G_i}$ and, in particular, $q(0) \cdot G_i = \langle q(0) \rangle_i$ where $q(0)$ can be $\alpha_v, \alpha_w, \alpha_y, \beta$, or some combination thereof, the particular combination of which may be based on a protocol/convention, such as a zk_SNARKs protocol.

- 5 In an embodiment, the α –parameters are shared by means of elliptic curve points whereas other parameters may be scalar values. For such values, in accordance with at least one embodiment, a Diffie-Hellman scheme is used to share scalar parameters between the two computing entities, without sharing the parameters themselves. Accordingly, in an embodiment, let $P = \{t_1, \dots, t_N\}$ be a set of N parameters. In an embodiment, it is assumed
- 10 that A and B have agreed on using a multiplicative group Γ with modulus μ and generator γ , and the participants (A and B) proceed with following the steps (while the exponential representation is used here as an illustrative example, others suitable representations may be utilised): for each $i \in \{1, \dots, N\}$, the first and second computing entities create a (private) random number $v_{A,i}, v_{B,i}$, respectively and both derive a (public) element: $\gamma^{v_{A,i}}$
- 15 and $\gamma^{v_{B,i}}$. The computing entities then exchange $\gamma^{v_{A,i}}$ and $\gamma^{v_{B,i}}$, and calculate $(\gamma^{v_{B,i}})^{v_{A,i}}$ and $(\gamma^{v_{A,i}})^{v_{B,i}}$ ($= (\gamma^{v_{B,i}})^{v_{A,i}} = \gamma^{v_{A,i}v_{B,i}}$), respectively. Accordingly, the parameters are set $t_i = \gamma^{v_{A,i}v_{B,i}}$.

- Accordingly, the techniques above have demonstrated that both the first and second computing entities share the parameters t_i without having exchanged them by, instead,
- 20 exchanging $\gamma^{v_{A,i}}$ and $\gamma^{v_{B,i}}$. It should be noted that in accordance with at least one embodiment, the number of participants is limited to $n=2$ as a result of at least one parameter being exchanged in the preceding manner.

Figure 4 illustrates a diagram 400 of a protocol based on a two-party common reference string (CRS) and corresponding proof-of-correctness (POC) or proof of correct execution.

- 25 The diagram 400 illustrates a first computing entity 402, a second computing entity 404, and a third computing entity 106 wherein the first computing entity 402 and the second computing entity 404 jointly make contributions to a smart contract that can be unlocked by the third computing entity 406 upon execution of the smart contract. In an embodiment, the protocol is implemented at least in part using a blockchain network.

In accordance with this disclosure, and described in greater detail (e.g., in connection with Figure 4), a scheme and protocol for two participants, A and B, may be utilised to generate a shared secret and, thus, a shared common reference string (CRS) which can be used to verify the correct execution of an associated circuit. In an embodiment, the scheme
5 assumes an off-chain exchange of data, first between A and B, and secondly between A+B (or either) and a worker C that performs a computational task on behalf of at least one of A or B. To cause the worker C to perform the computational task (e.g., execution of a smart contract), A and B both sign a transaction (which may or may not contain a P2SH-type redeem script), which requires the worker C to provide a proof-of-correctness and prove
10 possession of correct verification key (VK) in order to unlock the funds.

Techniques for implementing the protocol as presented in this disclosure, in some embodiments, do not require any protocol changes to existing blockchain networks (e.g., may be implemented on a Bitcoin-based blockchain network using existing commands that are already supported). In some embodiments, extensions to the existing set of commands
15 supported by the Bitcoin protocol are also discussed herein —the extensions may include new commands (e.g., new OP codes) that may have various benefits such as improving efficiency of the execution of smart contracts, reducing the size of the smart contract (which may reduce the amount of storage space needed by nodes of the blockchain network to properly operate), and more. In some embodiments, the cost of confirming a
20 smart transaction to the blockchain is based at least in part on the size of the smart contract.

In an embodiment, the exchange and transfer of elliptic curve points and other data pertaining to the common reference string are transferred off-chain. In an embodiment, the verification key is eventually broadcast or otherwise made available on-chain via the exchange of digital assets for work (e.g., execution of a smart contract) carried out by the
25 worker C and two parties wishing to evaluate their smart contract (A and B). As described herein, several schemes are possible. For instance, A and B may or may not both supply the VK or hashes of VK when preparing the locking transaction. In other words, in an embodiment, the majority of the capacity-intensive workload is done off-chain.

In an embodiment, the protocol includes both off-chain and on-chain components, as
30 denoted by the dotted line illustrated in Figure 4. Off-chain components may include communications and exchanges of data and information that can occur without storing data

to a blockchain ledger. For example, an off-chain component of the protocol may include an exchange of IP packets between a source and destination (e.g., the first computing entity 402 is a source that transmits a first set of parameters to a destination, the second computing entity 404). For example, an on-chain component of the protocol may include
 5 broadcasting data to a blockchain ledger that is made available to nodes of the blockchain network.

In an embodiment, the first computing entity 402 computes, based at least in part on a first polynomial, a set of elliptic curve points $\{\langle p_1^k(x_l) \rangle_{G_i}\}_{k \in \{1, \dots, n\}, l \in \{1, 2\}, i \in \{1, 2\}}$. The first
 computing entity 402 may transmit data to the second computing entity 404 comprising at
 10 least some of the set of elliptic curve points. For example, the first computing entity 402 may transmit the entire set of elliptic curve points. As a second example, the first computing entity 402 may transmit a subset of the elliptic curve points $\{\langle p_1^k(x_2) \rangle_G\}_{k \in \{1, \dots, n\}}$.

In some embodiments, additional data is transmitted that is not required to maintain the secrecy of a share secret s , but may be necessary to render the system zero-knowledge. For
 15 example, $\{\gamma^{v_{A,i}}\}_{i \in \{1, \dots, N\}}$, $q_{1,I}$ ($I = v, w, y$), or some combination thereof may also be computed (as described above in connection with Figure 3) to generate additional parameter values that can be used as part of determining a common reference string. In an embodiment, $\{\gamma^{v_{A,i}}\}_{i \in \{1, \dots, N\}}$ is a set of scalar values. In an embodiment, $q_{1,I}$ ($I = v, w, y$) refers to a set of elliptic curve points.

20 In an embodiment, the second computing entity 404 likewise computes, based on a polynomial that may be different from that used by the first computing entity 402, a set of elliptic curve points $\{\langle p_2^k(x_l) \rangle_{G_i}\}_{k \in \{1, \dots, n\}, l \in \{1, 2\}, i \in \{1, 2\}}$. The second computing entity 404 may perform similar operations as described above. For example, the second computing entity 404 may transmit a subset of the generated elliptic curve points $\{\langle p_2^k(x_1) \rangle_G\}_{k \in \{1, \dots, n\}}$
 25 to the first computing entity 402. Additionally, in an embodiment, optional parameters such as $\{\gamma^{v_{B,i}}\}_i$, $q_{2,I}$ ($I = v, w, y$), or some combination thereof may be exchanged, which, while not being necessary to maintain the secrecy of the shared secret s , may be utilised to ensure that the protocol is zero-knowledge.

In an embodiment, the exchanged quantities can be utilised by both the first computing entity 402 and the second computing entity 404 to calculate the identical common reference string. They may or may not provide the third computing entity (e.g., worker) with the common reference string, as the third computing entity 406 has to prove
5 possession of the correct verification key later on. The determination of the same common reference string by both the first and second computing entities may be performed off-chain.

Continuing with the protocol, in accordance with one embodiment, the first and second computing entities agree on a transaction in which they make proportionate contributions
10 for the execution of the smart contract. In an embodiment, the first and second computing entities agree upon a proportion of contributions and each provide transaction inputs that are encumbered to the smart contract and can be unlocked by the third computing entity upon execution of the smart contract. This may or may not be a pay-to-script-hash (P2SH)
15 P2SH type script may or may not contain the elements of the verification key or hash-values of verification key, i.e., $h_i = \text{HASH}(VK_i)$. In an embodiment, the key has been divided into chunks. The smart contract may be broadcast to a blockchain as the first transaction 410 illustrated in Figure 4 having a first transaction input contributed by the first computing entity 402 and a second transaction input contributed by the second
20 computing entity 404 that serves as a worker fee paid in a proportion agreed upon by the first and second computing entities.

In an embodiment, the third computing entity 406 — also referred to as a worker — unlocks the funds in a second transaction 410 according to the protocols in U.K. Patent Application No. 1719998.5 and/or U.K. Patent Application No. 1720768.9 — and the third
25 computing entity 406 unlocks the funds for the work (correct execution of the circuit) and in doing so, proves that it has possession of (a) the correct verification key and (b) a valid proof-of-correctness. The verification may be performed by another computer system (e.g., a node of the blockchain that is a verifier) or by either/both of the computing entities that are a party to the smart contract.

30 FIG. 5 shows an illustrative example of a process 500 for generating a two-party common reference string comprising a verification key and evaluation key, in accordance with an

- embodiment. Some or all of the process 500 (or any other processes described herein, or variations and/or combinations thereof) may be performed under the control of one or more computer systems configured with computer-executable instructions and may be implemented as code (e.g., computer-executable instructions, one or more computer
- 5 programs, or one or more applications) executing collectively on one or more processors, by hardware, software, or combinations thereof. The code may be stored on a computer-readable storage medium, for example, in the form of a computer program comprising a plurality of computer-readable instructions executable by one or more processors. The computer-readable storage medium may be a non-transitory computer-readable medium. In
- 10 some embodiments, at least some of the computer-readable instructions usable to perform the process 500 are not stored solely using transitory signals (e.g., a propagating transient electric or electromagnetic transmission). A non-transitory computer-readable medium may include non-transitory data storage circuitry (e.g., buffers, caches, and queues) within transceivers of transitory signals.
- 15 In an embodiment, the system performing the process 500 is a computing entity that is a party to a smart contract that performs a process to at least establish information that can be used by the system and another party of the smart contract to calculate an identical common reference string. A common reference string described in connection with the process 500 may be in accordance with those discussed in connection with, for example,
- 20 Figures 1-4. In an embodiment, the common reference string is expressed by means of the polynomials $v(x), w(x)$ evaluated in a secret s of the form:

$$v(s) = a_0 + a_1s + a_2s^2 + \dots + a_ns^n$$

$$w(s) = b_0 + b_1s + b_2s^2 + \dots + b_ns^n$$

- In an embodiment, the first computing entity determines 502 a first polynomial to generate
- 25 a first set of elliptic curve values. In an embodiment, the system generates elliptic curve points of form: $\langle v(s) \rangle_G = \langle a_0 \rangle_G + \langle a_1s \rangle_G + \langle a_2s^2 \rangle_G + \dots$, for some generator G of the associated group (e.g., of elliptic curve points). Unless otherwise noted, the polynomials in this process 500 are defined over a field, $\mathbb{F}_r$. Thus, let $(\mathbb{F}_r, +, \cdot)$ be a field. Then, a polynomial over $\mathbb{F}_r$ is given by $P: \mathbb{F} \rightarrow \mathbb{F}'$ and $P(x) = \sum_{i=0}^n a_i x^i$ where $a_i \in \mathbb{F}$. The set of
- 30 elliptic curve points may be represented as $\{\langle p_1^k(x_l) \rangle_{G_l}\}_{k \in \{1, \dots, n\}, l \in \{1, 2\}, i=1 \cdot}$

The first computing entity may make 504 the set of elliptic curve points available to the second computing entity. In an embodiment, the system need not make the entire set of elliptic curve points available to the second computing entity — rather in an embodiment, the system transmits a subset of the elliptic curve points $\{\langle p_1^k(x_2) \rangle_G\}_{k \in \{1, \dots, n\}}$.

- 5 For example, in accordance with at least one embodiment, where $n = 2$, the first computing entity computes $\{p_1(x_1), p_1^2(x_1), p_1(x_2), p_1^2(x_2)\}$ and shares, with the second computing entity, $\{p_1(x_2), p_1^2(x_2)\}$. The remaining quantities $\{p_1(x_1), p_1^2(x_1)\}$ may be utilised by the first computing entity to calculate the common reference string.

- The second computing entity, which is also a party to the smart contract, may separately
 10 generate a set of elliptic curve points for the same input points (e.g., generate elliptic curve points $\{\langle p_2^k(x_1) \rangle_G\}_{k \in \{1, \dots, n\}}$) and provide either all or some of the generated points to the first computing entity. The first computing entity may receive 506, from the second computing entity, a second set of elliptic curve points that correspond to some or all of the elliptic curve points generated by the second computing entity, wherein the subset may be
 15 determined based on techniques described elsewhere in this disclosure (e.g., in connection with Figures 2 and 6).

- In an embodiment, the system determines 508 an identical common reference string based on at least a portion of the first and second sets of elliptic curve points. For example, after the exchange of elliptic curve points, a Lagrange interpolation may be utilised to express p
 20 in terms of $p(x_1)$ and $p(x_2)$. In an embodiment, both parties of the smart contract can reconstruct powers $\langle p^n(x) \rangle_G$ (and in particular, $\langle p^n(0) \rangle_G = \langle s^n \rangle_G$), by means of the exchanged points $\langle p_i(x_j) \rangle_G$. For powers $\langle p^n(x) \rangle_G$, the multinomial formula as described above may be utilised:

$$\langle p^n(x_j) \rangle_G = \langle [p_1(x_j) + \dots + p_m(x_j)]^n \rangle_G =$$

$$25 \quad \langle \sum_{k_1+k_2+\dots+k_m=n} \binom{n}{k_1, k_2, \dots, k_m} \prod_{t=1}^m p_t^{k_t}(x_j) \rangle_G.$$

For example, for $m = 2$ this becomes:

$$\langle p^n(x_j) \rangle_G = \langle [p_1(x_j) + p_2(x_j)]^n \rangle_G = \langle \sum_{k_1+k_2=n} \binom{n}{k_1, k_2} \prod_{t=1}^2 p_t^{k_t}(x_j) \rangle_G.$$

In an embodiment, additional parameters (e.g., scalar values and/or elliptic curve points) are exchanged between the first and second computing entities, such as in the manner described in connection with Figure 3, and the parameters, in conjunction with a power of the shared secret $\langle s^n \rangle_G$ are utilised to compute the verification key and/or evaluation key.

- 5 In an embodiment, the parameters are exchanged without reliance on encryption and/or communications channels that provide cryptographically verifiable assurances of confidentiality.

In an embodiment, the first and second computing entities agree on a transaction and each makes a contribution to a respective transaction input of the smart contract which can be unlocked by a third computing entity (e.g., worker) that correctly executes the smart contract. In an embodiment, either of the computing entities provides a proportionate worker fee. There may or may not be a P2SH type agreement in which both make contributions to the same address (e.g., address for worker). In an embodiment, the P2SH script includes elements of the verification key or hash values of the verification key. A worker (e.g., third computing entity) may unlock (e.g., unlock) the contributions by providing a computationally verifiable attestation that the worker has the correct verification and provides a valid proof of correctness, for example, by using techniques described in connection with U.K. Patent Application No. 1719998.5 and/or U.K. Patent Application No. 1720768.9.

- 20 FIG. 6 shows an illustrative example of a process 600 for sharing powers of a shared secret between n parties (e.g., $n > 2$), in accordance with at least one embodiment. Some or all of the process 600 (or any other processes described herein, or variations and/or combinations thereof) may be performed under the control of one or more computer systems configured with computer-executable instructions and may be implemented as code (e.g., computer-executable instructions, one or more computer programs, or one or more applications) executing collectively on one or more processors, by hardware, software, or combinations thereof. The code may be stored on a computer-readable storage medium, for example, in the form of a computer program comprising a plurality of computer-readable instructions executable by one or more processors. The computer-readable storage medium may be a non-transitory computer-readable medium. In some embodiments, at least some of the computer-readable instructions usable to perform the process 600 are not stored solely
- 25
- 30

using transitory signals (e.g., a propagating transient electric or electromagnetic transmission). A non-transitory computer-readable medium may include non-transitory data storage circuitry (e.g., buffers, caches, and queues) within transceivers of transitory signals. In an embodiment, the equivalent of a threshold is given *a priori* and it is assumed
 5 that the necessary number of participants first agree. This differs from various existing techniques, such as those described in Shamir's Secret Sharing Scheme (4S), wherein the sharing of a secret only works under the limitation that a given threshold is reached.

Secret sharing, in accordance with various embodiments, is valid for an arbitrary number of parties. In an embodiment, large parts of the formalism described in this disclosure can
 10 be applied to multi-party ($n > 2$) scenarios. In some embodiments, multi-party systems ($n > 2$) are not required to conceal certain other parameters (e.g., some or all of the following non-elliptic curve (e.g., scalar) parameters: $r_v, r_w, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma$). If, however, those parameters are to remain private according to the protocol, different approaches — such as those described in connection with Figure 3 — may be utilised to conceal
 15 parameters such as r_v, r_w .

In an embodiment, all participants agree 602 upon and/or have access to the function $f: \mathbb{F} \rightarrow \mathbb{F}'$ which maps finite field elements to other finite field elements. The functions, in an embodiment, serve as coefficients/exponents of generators of groups. For illustrative purposes, embodiments described hereinafter focus on polynomials since the functions we
 20 are interested in can be expressed in terms of polynomials by means of generalisations of Taylor series. In particular, in various embodiments, the parties involved evaluate the functions in the same value and without passing the value around amongst themselves. For example, in the case of a polynomial of order 1, this amounts to utilising Shamir's Secret Sharing Scheme. For higher orders ($n > 1$), the parties are dealing with expressions of the
 25 form:

$$f(s) = a_0 + a_1s + a_2s^2 + \dots + a_ns^n$$

Accordingly, techniques described herein can be utilised to ensure that all participants have the same $\langle f(s) \rangle_G$. For example, in a protocol described in accordance with at least one embodiment, the same EQ_FSG can be shared among two or more participants (i.e., $n >$
 30 1) by publically distributing points in the form $\langle s^r \rangle_G$, for arbitrary integer powers r , since

$\langle f(s) \rangle_G = \langle a_0 \rangle_G + \langle a_1 s \rangle_G + \langle a_2 s^2 \rangle_G + \dots$, where G is a generator of the group in question (e.g., of elliptic curve points).

In an embodiment, each participant is able to generate 604 polynomials, which are evaluated in a set of points $(x_1, x_2, \dots)$ with $x_i \neq 0 \forall i$, and the points may be known to all parties. In an embodiment, the sum of the polynomials of each participant makes up the (master) polynomial, of which the intersection with the y -axis is the secret, i.e.,

$$p(x) = \sum_{j=1}^m p_j(x)$$

with $p(0) = s$, and where m is the number of participants. The intersection with the y -axis as described above may be referred to as an intersection point.

10 In order to establish s , each participant shares 606 a corresponding polynomial evaluated in the different points $(x_1, x_2, \dots)$. More specifically, participant i creates/calculates $p_i(x_j)$ for $i, j \in \{1, \dots, m\}$, and sends off $\langle p_i(x_j) \rangle_G$ to j . Once these quantities are shared, each participant is able to calculate or otherwise determine 608 a power of the shared secret s^r .

Slightly more scrutiny is needed when considering s^r , since it amounts to sharing representations of

$$s^r = (p(0))^r = \left(\sum_{j=1}^m p_j(0) \right)^r$$

It may not possible to calculate powers of $\langle p_i(x_j) \rangle_G$ because powers of the generator are generally not defined. However, since all participants can infer $\langle s \rangle_G$ by means of the exchanged $\langle p_i(x_j) \rangle_G$, it is possible to start by examining powers of the Lagrange interpolation polynomial $L(x)$ (as the master polynomial generally can be constructed by means of Lagrange interpolation (Lagrange Polynomial, n.d.)). The Lagrange interpolation can be written as:

$$L(x) = \sum_{j=1}^m p(x_j) l_j(x)$$

where

$$l_j(x) = \prod_{\substack{1 \leq k \leq m \\ k \neq j}} \frac{x - x_k}{x_j - x_k}$$

and thus:

$$L^r(x) = \left(\sum_{j=1}^k p(x_j) l_j(x) \right)^r = \sum_{k_1 + k_2 + \dots + k_m = r} \binom{r}{k_1, k_2, \dots, k_m} \prod_{t=1}^m (p(x_t) l_t(x))^{k_t}$$

5 where

$$\binom{r}{k_1, k_2, \dots, k_m} = \frac{r!}{k_1! k_2! \dots k_m!}$$

is the multinomial coefficient (Multinomial Theorem, n.d.). Accordingly, the $l_j(x)$ polynomials can be calculated by each participant independently. Next, each participant can calculate:

10

$$\langle L^r(x) \rangle_G = \left\langle \left(\sum_{j=1}^m p(x_j) l_j(x) \right)^r \right\rangle_G = \left\langle \sum_{k_1 + k_2 + \dots + k_m = r} \binom{r}{k_1, k_2, \dots, k_m} \prod_{t=1}^m (p(x_t) l_t(x))^{k_t} \right\rangle_G$$

such that

$$\langle p^n(x_j) \rangle_G = \langle [p_1(x_j) + \dots + p_k(x_j)]^n \rangle_G$$

which, in turn, can be rewritten as (using the multinomial expansion):

$$15 \quad \langle p^n(x_j) \rangle_G = \langle [p_1(x_j) + \dots + p_k(x_j)]^n \rangle_G = \left\langle \sum_{k_1 + k_2 + \dots + k_m = n} \binom{n}{k_1, k_2, \dots, k_m} \prod_{t=1}^m p_t^{k_t}(x_j) \right\rangle_G$$

This means that participant i (who is the owner/creator of polynomial p_i) can provide participant j with powers of $\langle p_i(x_j) \rangle_G$, i.e., with the set:

$$\{\langle p_i^k(x_j) \rangle_G\}_{j \in \{1, \dots, m\}}$$

This allows participant j to calculate $p_j^l(x_j)p_i^k(x_j) \cdot G$ (and similarly $p_j^l(x_i)p_i^k(x_i) \cdot G$) etc.

In an embodiment, a participant calculates expressions on the form:

$$\begin{aligned} \langle L^r(x) \rangle_G &= \left\langle \left(\sum_{j=1}^m p(x_j) l_j(x) \right)^r \right\rangle_G \\ &= \left\langle \sum_{k_1+k_2+\dots+k_m=r} \binom{r}{k_1, k_2, \dots, k_m} \prod_{t=1}^m (l_t(x))^{k_t} \left(\sum_{k'_1+k'_2+\dots+k'_m=k_t} \binom{k_t}{k'_1, k'_2, \dots, k'_m} \prod_{t'=1}^m p_{t'}^{k'_{t'}}(x_j) \right) \right\rangle \end{aligned}$$

- 5 Consider an example, in accordance with at least one embodiment, in which participants use an elliptic curve and where G is a generator in the corresponding (multiplicative) representation. In the case of two participants, A and B: A sends $\langle p_1(x_2) \rangle_G$ to B, and receives $\langle p_2(x_1) \rangle_G$ in return. Accordingly, participant A can calculate

$$\langle p(x_1) \rangle_G = \langle p_1(x_1) + p_2(x_1) \rangle_G$$

- 10 and similarly, participant B can calculate:

$$\langle p(x_2) \rangle_G = \langle p_1(x_2) + p_2(x_2) \rangle_G$$

Using Lagrange interpolation it is possible to express p in terms of $p(x_1)$ and $p(x_2)$:

$$p(x) = p(x_1) \frac{x - x_2}{x_1 - x_2} + p(x_2) \frac{x - x_1}{x_2 - x_1} = \frac{p(x_1) - p(x_2)}{x_1 - x_2} x + \frac{-p(x_1)x_2 + p(x_2)x_1}{x_1 - x_2}$$

- Each participant can reconstruct $\langle p(x) \rangle_G$ (and in particular $\langle p(0) \rangle_G$), by means of the
15 exchanged points $\langle p_i(x_j) \rangle_G$. For higher powers of $\langle p^n(x) \rangle_G$, the multinomial formula from the previous section may be utilised:

$$\langle p^n(x_j) \rangle_G = \langle [p_1(x_j) + \dots + p_m(x_j)]^n \rangle_G = \left\langle \sum_{k_1+k_2+\dots+k_m=n} \binom{n}{k_1, k_2, \dots, k_m} \prod_{t=1}^m p_t^{k_t}(x_j) \right\rangle_G$$

For $m = 2$ this becomes:

$$\langle p^n(x_j) \rangle_G = \langle [p_1(x_j) + p_2(x_j)]^n \rangle_G = \left\langle \sum_{k_1+k_2=n} \binom{n}{k_1, k_2} \prod_{t=1}^2 p_t^{k_t}(x_j) \right\rangle_G$$

After this exchange (following certain, pre-arranged conventions) both parties can calculate $\langle p^n(x_j) \rangle_G$ and, in particular, $\langle p^n(0) \rangle_G = \langle s^n \rangle_G$.

A protocol according to the process 600 is described hereinbelow. Since each participant needs to be able to obtain expressions of the form $\langle (k_1, k_2, \dots, k_m)^n \prod_{t=1}^m p_t^{k_t}(x_j) \rangle_G$ and

5 ordering may necessary, when exchanging the points. Here we illustrate one such solution.

Without loss of generality, it can be assumed, in accordance with at least one embodiment, that participant 1 is the one submitting the first elliptic curve points. The protocol, in an embodiment, follows the steps:

1. Participant 1 distributes $\langle p_1^{k_1}(x_j) \rangle_G$ to all $i \neq 1$ participants, where $k_1 =$
10 $1, \dots, n$ and $j = 1, \dots, m$, and where m here is the number of participants
2. Participant 2 distributes $\langle p_1^{k_1}(x_j) p_2^{k_2}(x_j) \rangle_G$ for $k_1 = 0, \dots, n-1$, $k_2 =$
 $1, \dots, n$ and $j = 1, \dots, m$ to all $i \neq 2$ participants
3. Participant 3 distributes $\langle p_1^{k_1}(x_j) p_2^{k_2}(x_j) p_3^{k_3}(x_j) \rangle_G$ for $k_1 = 0, \dots, n-1$,
 $k_2 = 0, \dots, n-1$, $k_3 = 1, \dots, n$ and $j = 1, \dots, m$ to all $i \neq 3$ participants
- 15 4. ... (and so on, for each participant)

The l th participant in the sequence distributes $\langle p_1^{k_1}(x_j) \dots p_l^{k_l}(x_j) \rangle_G$ for $k_i = 0, \dots, n-1$ for $i \in \{1, \dots, l-1\}$, $k_l = 1, \dots, n$ and $j = 1, \dots, m$ to all $i \neq l$. When the last participant, m , distributes his/her points, all the participants have the necessary components to calculate $\langle s^n \rangle_G = \langle p^n(0) \rangle_G$.

- 20 In an embodiment, the process 600 comprises a plurality of m participants (e.g., more than 2 participants) that exchange sets of point of the form $\{\langle p_1^{k_1}(x_j) \dots p_m^{k_m}(x_j) \rangle_G\}$ wherein the points are used by each participant to calculate $\langle p^k(x_j) \rangle_G$. The participants may then use the $\langle p^k(x_j) \rangle_G$, together with Lagrange interpolation, to obtain $\langle p^k(x) \rangle_G$ and, in particular, $\langle s^k \rangle_G = \langle p^k(0) \rangle_G$. In an embodiment, the points are elliptic curve points wherein $x_j \neq$
25 $0 \forall j$.

The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense. However, it will be evident that various modifications and changes may be made thereunto without departing from the scope of the invention as set forth in the claims. Likewise, other variations are within the scope of the present disclosure. Thus,
5 while the disclosed techniques are susceptible to various modifications and alternative constructions, certain illustrated embodiments thereof are shown in the drawings and have been described above in detail. It should be understood, however, that there is no intention to limit the invention to the specific form or forms disclosed but, on the contrary, the intention is to cover all modifications, alternative constructions and equivalents falling
10 within the scope of the invention, as defined in the appended claims.

The use of the term “set” (e.g., “a set of items”) or “subset”, unless otherwise noted or contradicted by context, is to be construed as a nonempty collection comprising one or more members. Further, unless otherwise noted or contradicted by context, the term “subset” of a corresponding set does not necessarily denote a proper subset of the
15 corresponding set, but the subset and the corresponding set may be equal.

Conjunctive language, such as phrases of the form “at least one of A, B, and C”, or “at least one of A, B and C”, unless specifically stated otherwise or otherwise clearly contradicted by context, is otherwise understood with the context as used in general to present that an item, term, etc., could be either A or B or C, or any nonempty subset of the
20 set of A and B and C. For instance, in the illustrative example of a set having three members, the conjunctive phrases “at least one of A, B, and C” and “at least one of A, B and C” refer to any of the following sets: {A}, {B}, {C}, {A, B}, {A, C}, {B, C}, {A, B, C}. Thus, such conjunctive language is not generally intended to imply that certain embodiments require at least one of A, at least one of B and at least one of C each to be
25 present. Further, unless stated otherwise or otherwise clear from context, the phrase “based on” means “based at least in part on” and not “based solely on.”

Operations of processes described can be performed in any suitable order unless otherwise indicated or otherwise clearly contradicted by context. Processes described (or variations and/or combinations thereof) can be performed under the control of one or more computer
30 systems configured with executable instructions and can be implemented as code (e.g., executable instructions, one or more computer programs or one or more applications)

executing collectively on one or more processors, by hardware or combinations thereof. In some embodiments, the code can be stored on a computer-readable storage medium, for example, in the form of a computer program comprising a plurality of instructions executable by one or more processors. In some embodiments, the computer-readable
5 storage medium is non-transitory.

The use of any and all examples, or exemplary language (e.g., “such as”) provided, is intended merely to better illuminate embodiments of the invention and does not pose a limitation on the scope of the invention unless otherwise claimed. No language in the specification should be construed as indicating any non-claimed element as essential to the
10 practice of the invention.

Embodiments of this disclosure are described, including the best mode known to the inventors for carrying out the invention. Variations of those embodiments will become apparent to those of ordinary skill in the art upon reading the foregoing description. The inventors expect skilled artisans to employ such variations as appropriate and the inventors
15 intend for embodiments of the present disclosure to be practiced otherwise than as specifically described. Accordingly, the scope of the present disclosure includes all modifications and equivalents of the subject matter recited in the claims appended hereto as permitted by applicable law. Moreover, any combination of the above-described elements in all possible variations thereof is encompassed by the scope of the present
20 disclosure unless otherwise indicated or otherwise clearly contradicted by context.

All references, including publications, patent applications, and patents, cited are hereby incorporated by reference to the same extent as if each reference were individually and specifically indicated to be incorporated by reference and were set forth in its entirety.

It should be noted that the above-mentioned embodiments illustrate rather than limit the
25 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. The mere fact that certain measures are recited in mutually different dependent claims does not indicate that a combination of these measures cannot be used to advantage.

Claims

1. A computer-implemented method comprising, at a first computing entity (102, 302, 402):
 - determining (502), based at least in part on a first polynomial and at least two elliptic curve points, a set of elliptic curve points for a second computing entity (104, 304, 404);
 - making (504) a subset of the set of elliptic curve points available to the second computing entity (104, 304, 404);
 - receiving (506) a second set of elliptic curve points generated using a second polynomial;
 - determining a power of a secret based at least in part on the first set and the second set;
 - determining (508), based at least in part on the power of the secret, a common reference string (108) comprising a verification key (114, 306) and an evaluation key (112), wherein the common reference string (108) is also determinable by the second computing entity (104, 304, 404) as a result of the first computing entity (102, 302, 402) providing the subset to the second computing entity (104, 304, 404); and
 - generating a smart contract (110) comprising a first transaction input provided by the first computing entity (102, 302, 402) and a second transaction input provided by the second computing entity (104, 304, 404), wherein correct execution of the smart contract (110) by a third computing entity (106, 406) results in the third computing entity (106, 406) being able to generate a blockchain transaction using an output of the smart contract (110).
2. A method according to any preceding claim, wherein the set of elliptic curve points comprises corresponding elliptic curve points for powers of the first polynomial.
3. A method according to any preceding claim, wherein the first polynomial is of at least order 2.
4. A method according to any preceding claim, wherein the subset is the set of elliptic curve points.

5. A method according to any preceding claim, wherein the secret is shared between the first computing entity (102, 302, 402) and the second computing entity (104, 304, 404) without using a cryptographically protected communications channel.
6. A method according to any preceding claim, wherein the first computing entity (102, 302, 402) and the second computing entity (104, 304, 404) collectively determine the first digital asset and the second digital asset.
7. A method according to any preceding claim, further comprising:
 - determining, based on a third polynomial and the at least two elliptic curve points, a third set of elliptic curve points for the second computing entity (104, 304, 404);
 - making a second subset of the third set of elliptic curve points available to the second computing entity (104, 304, 404);
 - receiving a fourth set of elliptic curve points;
 - determining a parameter based at least in part on the third set and the fourth set, the parameter also determinable by the second computing entity (104, 304, 404) as a result of the first computing entity (102, 302, 402) providing the second subset to the second computing entity (104, 304, 404); and
 - wherein the determining of the common reference string is based further at least in part on the parameter.
8. A method according to any preceding claim, further comprising sharing an elliptic curve parameter between the first computing entity (102, 302, 402) and the second computing entity (104, 304, 404) using Shamir's Secret Sharing Scheme.
9. A method according to any preceding claim, further comprising exchanging a scalar parameter between the first computing entity (102, 302, 402) and the second computing entity (104, 304, 404) using a Diffie-Hellman scheme.
10. A method according to any preceding claim, wherein the smart contract (110) comprises a P2SH type unlocking script that allows the third computing entity (106, 406) to unlock the first digital asset and the second digital asset in response to providing a proof of correct execution (116).

11. A method according to any preceding claim, wherein the first computing entity (102, 302, 402) makes the subset available to the second computing entity (104, 304, 404) via an off-chain communications channel.
12. A method according to any preceding claim, wherein the second polynomial is inaccessible to the first computing entity (102, 302, 402).
13. A method according to any preceding claim, wherein the at least two elliptic curve points are two different elliptic curve points.
14. A system, comprising:
 - a processor; and
 - memory including executable instructions that, as a result of execution by the processor, causes the system to perform the computer-implemented method according to any preceding claim.
15. A non-transitory computer-readable storage medium having stored thereon executable instructions that, as a result of being executed by a processor of a computer system, cause the computer system to at least perform the computer-implemented method according to any of claims 1 to 13.