CLAIMS

1. A computer-implemented method of enforcing conditions on a second blockchain transaction using a first blockchain transaction, wherein a first one of the conditions is that, when a first unlocking script of the second transaction is executed alongside a first locking script of the first transaction, a representation of the second transaction is output to memory, wherein the representation is based on a plurality of fields of the second transaction and a first output of the first transaction, and wherein the method comprises:

generating the first transaction, wherein the first transaction comprises a first output, wherein the first output comprises the first locking script, and wherein the first locking script comprises:

a message sub-script configured to, when executed, output to memory a candidate message representing the second transaction, wherein the candidate message is based on a plurality of candidate fields of the first and second transactions, wherein one or more of the candidate fields are included in the first unlocking script of the second transaction, and wherein the message sub-script is configured to generate one or more respective parts of the candidate message based on a respective set of the candidate fields, and to re-use at least one of the respective sets of candidate fields as a different respective part of the candidate message;

a signature sub-script configured to, when executed, generate a signature, wherein the signature is a function of at least the candidate message, a private key and an ephemeral private key;

a public key corresponding to the private key; and

a verification sub-script configured to, when executed, i) construct a target message representing the second transaction, wherein the target message is based on a plurality of fields of the second transaction and the first output of the first transaction, and ii) use the public key to verify that the signature is valid for the target message, wherein verifying that the signature is valid for the target message, verifies that the target message matches the candidate message, thereby enforcing the condition that the candidate message output to memory is the representation of the second transaction.

2. The method of claim 1,

wherein one of said respective parts of the candidate message comprises a hash of one or more outputs of the second transaction,

wherein a first respective set of the candidate fields comprises a) a respective length of the first locking script of the first transaction, and b) the first locking script of the first transaction, and

wherein the message sub-script is configured to, when executed, generate the hash of the one or more outputs based on candidate fields a) and b), thereby enforcing a condition that a first output of the second transaction comprises the first locking script of the first transaction.

3. The method of claim 2,

wherein the first respective set of the candidate fields comprises c) a respective value locked by the first locking script of the first transaction, and

wherein the message sub-script is configured to, when executed, generate the hash of the one or more outputs based on candidate fields a), b) and c), thereby enforcing a condition that a first output of the second transaction comprises the first output of the first transaction.

4. The method of any preceding claim, wherein the message sub-script is configured, when executed, to duplicate the at least one or the respective sets of candidate fields as part of the candidate message to be re-used as the different respective part of the candidate message.

5. The method of any preceding claim, wherein the first locking script comprises a distinguished encoding rules, DER, sub-script configured to, when executed, convert the signature to a DER formatted signature, and wherein using the public key to verify that the signature is valid for the target message comprises using the public key to verify that the DER formatted signature is valid for the message.

6. The method of any preceding claim, wherein the signature sub-script comprises a signature flag specifying which of the plurality of fields of the second transaction are to form

the basis of the target message, and wherein the verification sub-script is configured to construct the target message based on the signature flag.

7. The method of claim 6, wherein the signature flag specifies that each input and each output of the second transaction are to form the basis of the target message.

8. The method of claim 6, wherein the signature flag specifies that a) a first input comprising the first unlocking script of the second transaction, and b) a first output of the second that is paired with the first input, are to form the basis of the target message.

9. The method of any preceding claim, wherein one or more of the candidate fields are included in the first locking script of the first transaction

10. The method of claim 9, wherein one or more of the following candidate fields are included in the first locking script of the first transaction:

- version number of the second transaction,
- length of the first locking script of the first transaction,
- value of the first locking script of the first transaction,
- sequence number of the first input of the second transaction,
- locktime of the second transaction,
- signature flag of the first unlocking script of the second transaction.

11. The method of any preceding claim, wherein the candidate message representing the second transaction comprises one or more respective data items that are based on a respective set of one or more respective candidate fields of the second transaction.

12. The method of claim 11, wherein the one or more respective data items comprises one or more of:

- hash of input sequence numbers of the second transaction,
- hash of combined outputs of the second transaction.

13. The method of any preceding claim, wherein the ephemeral private key is fixed by the first locking script as being equal to one and/or wherein the ephemeral private key is fixed as being equal to the private key.

14. The method of claim 13, wherein the private key is fixed by the first locking script as being equal to one.

15. The method of claim 13 or claim 14, wherein the signature is of the form $s = k^{-1}(z + ra)\ mod\ n$, where k is the ephemeral private key, a is the private key, z is a hash of the candidate message, n is the integer order of the elliptic curve generator point G, and r is the x-coordinate of an ephemeral public key modulo n.

16. The method of claim 13 or any claim dependent thereon, wherein the signature is of the form $s = z + G_x\ mod\ n$, where G_x is the x-coordinate of the elliptic curve generator point G, wherein G is both the ephemeral public key and the public key.

17. The method of claim 16, wherein the signature sub-script comprises respective values of G_x and n, and wherein the signature is generated based on the respective values of G_x and n.

18. The method of claim 16 and claim 17, wherein the stack-based memory comprises a main stack and an alternative stack, wherein the first locking script is configured to use the respective values of G_x and n more than once, and wherein the first locking script is configured to, when executed, output the respective values of G_x and n to the alternative stack and, for each time the respective values of G_x and n are required other than an initial time, obtain the respective values of G_x and n from the alternative stack.

19. Computer equipment comprising:

memory comprising one or more memory units; and

processing apparatus comprising one or more processing units, wherein the memory stores code arranged to run on the processing apparatus, the code being configured so as when on the processing apparatus to perform the method of any of claims 1 to 18.

20. A computer program embodied on computer-readable storage and configured so as, when run on one or more processors, to perform the method of any of claims 1 to 18.