

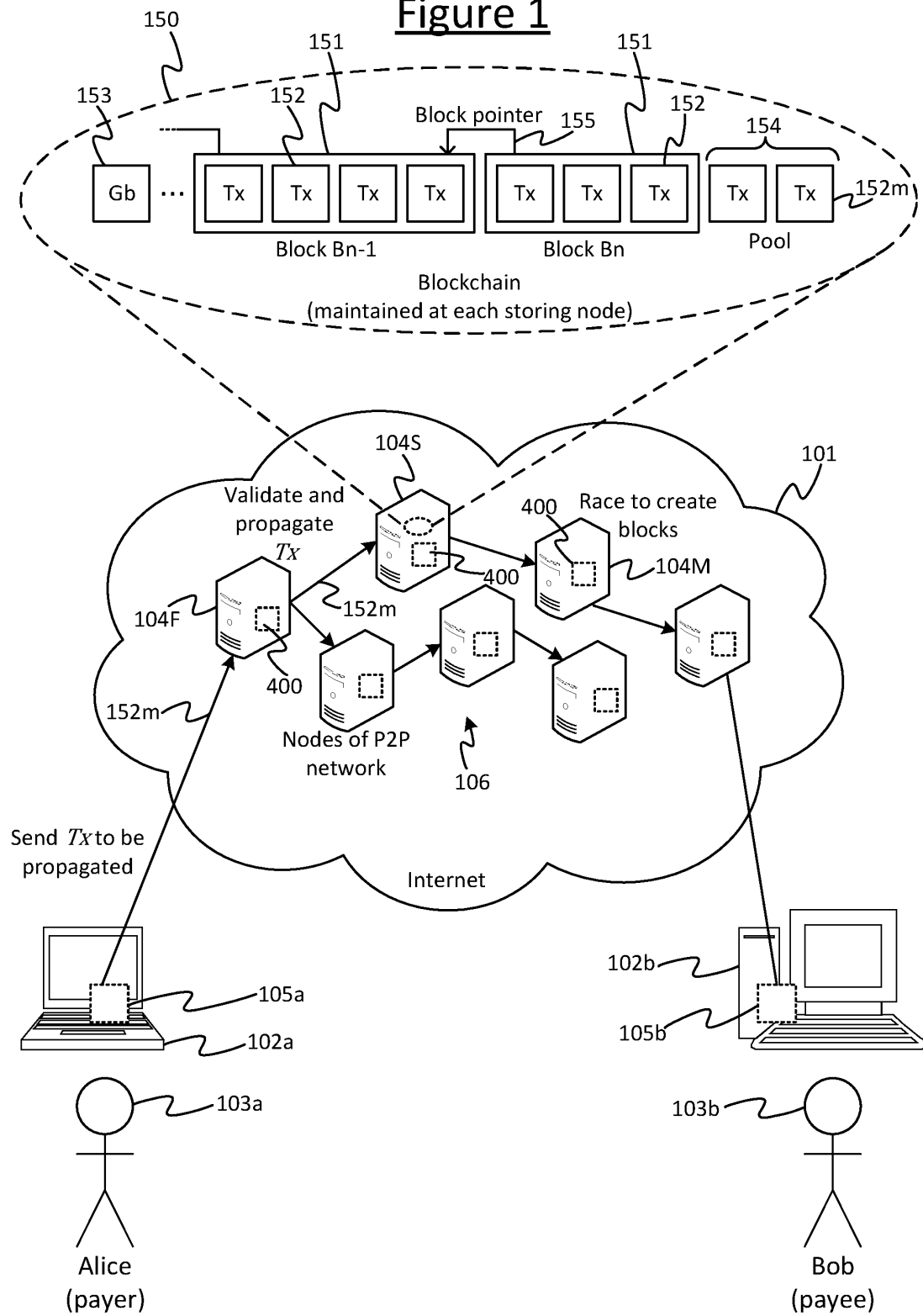
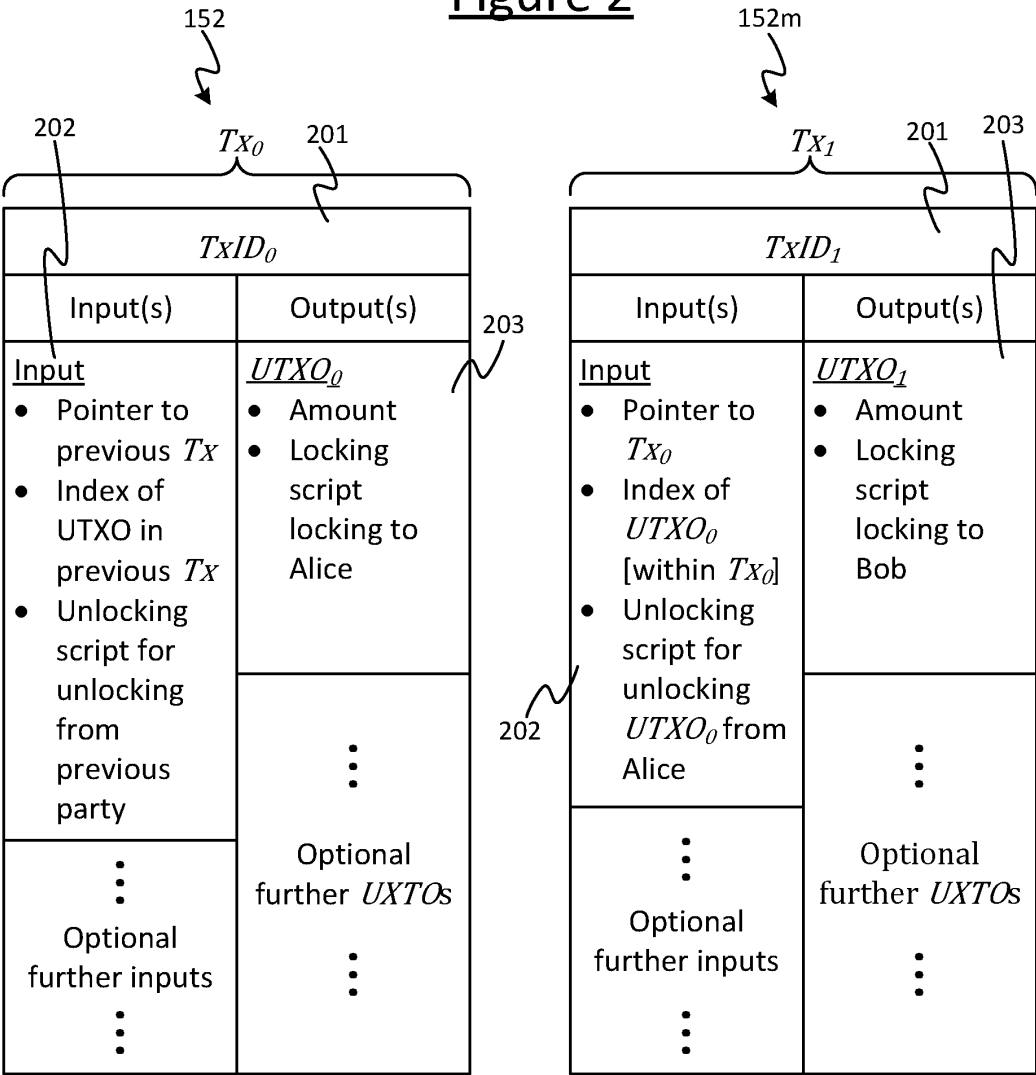
Figure 1

Figure 2



Transaction
from Alice to Bob

↓

Validated by running: Alice's locking script (from output of Tx_0), together with Alice's unlocking script (as input to Tx_1). This checks that Tx_1 meets the condition(s) defined in Alice's locking script.

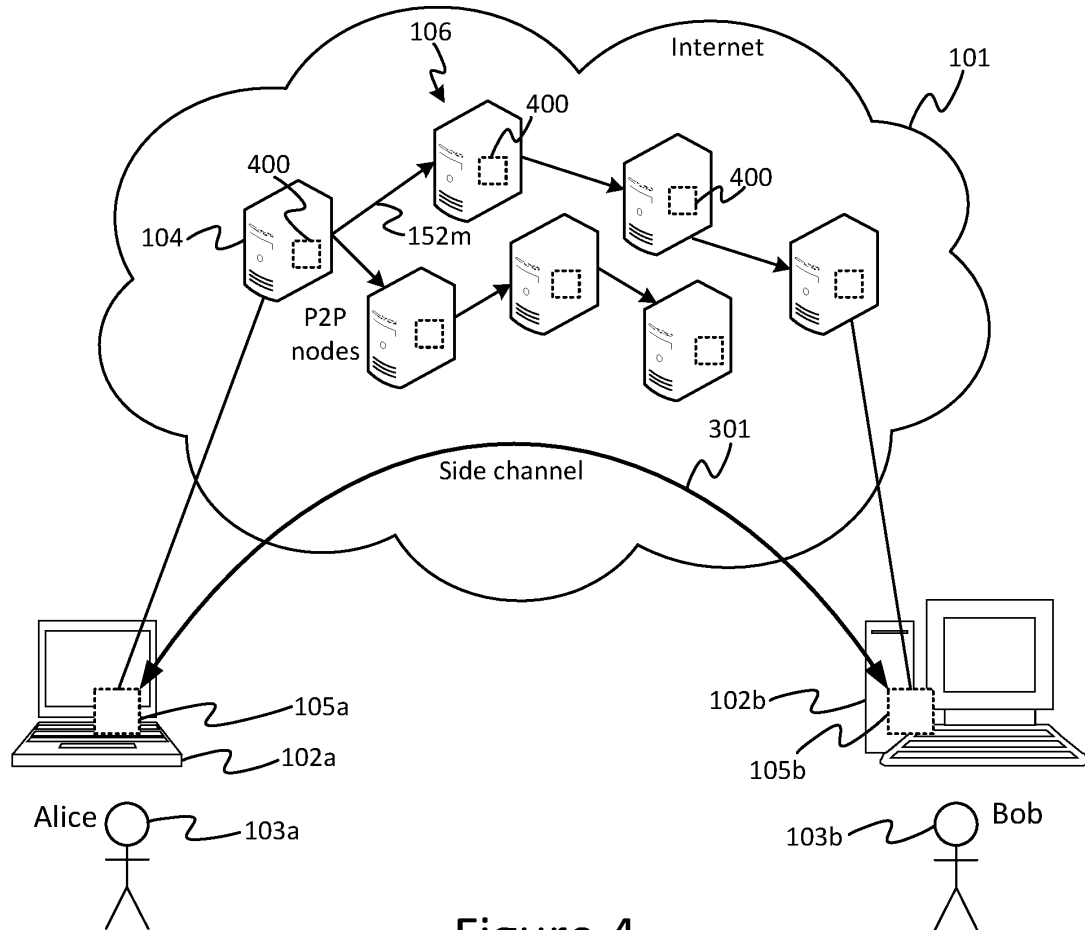
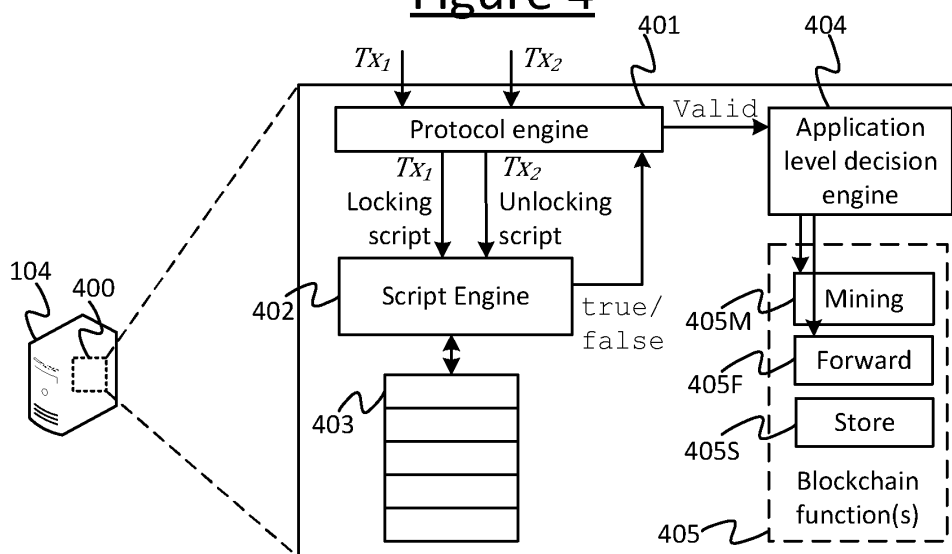
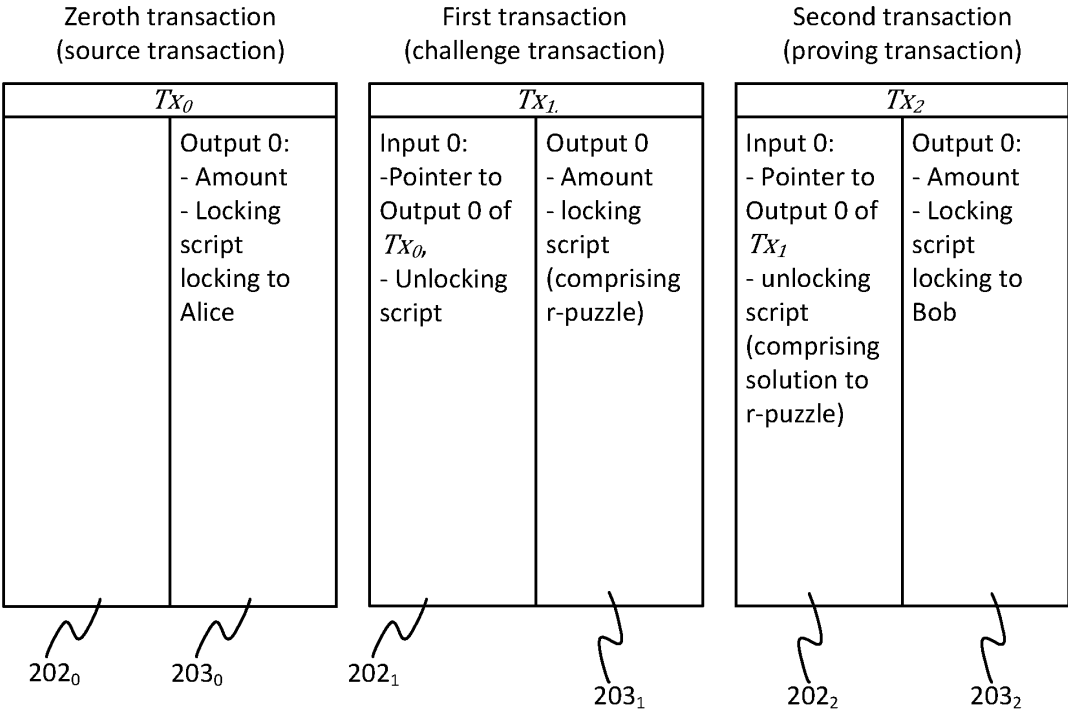
Figure 3**Figure 4**

Figure 5



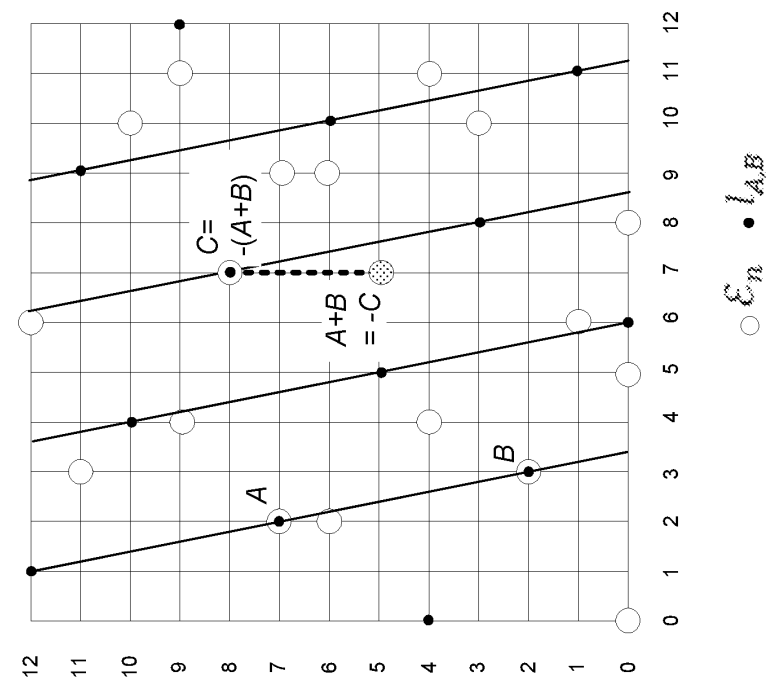


FIG. 6B

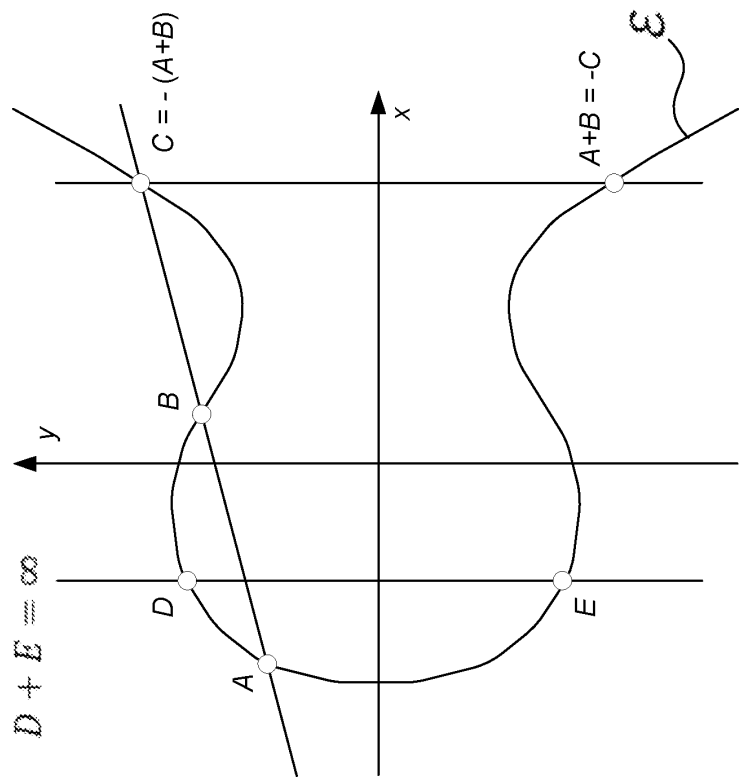


FIG. 6A

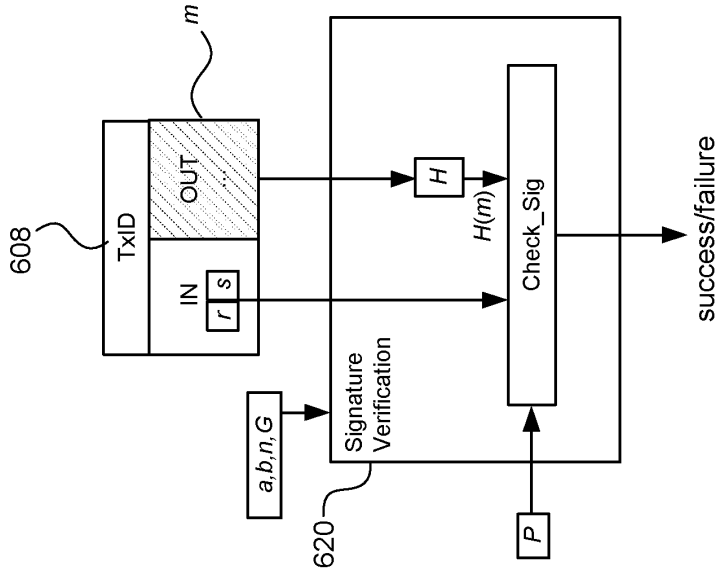


FIG. 6D

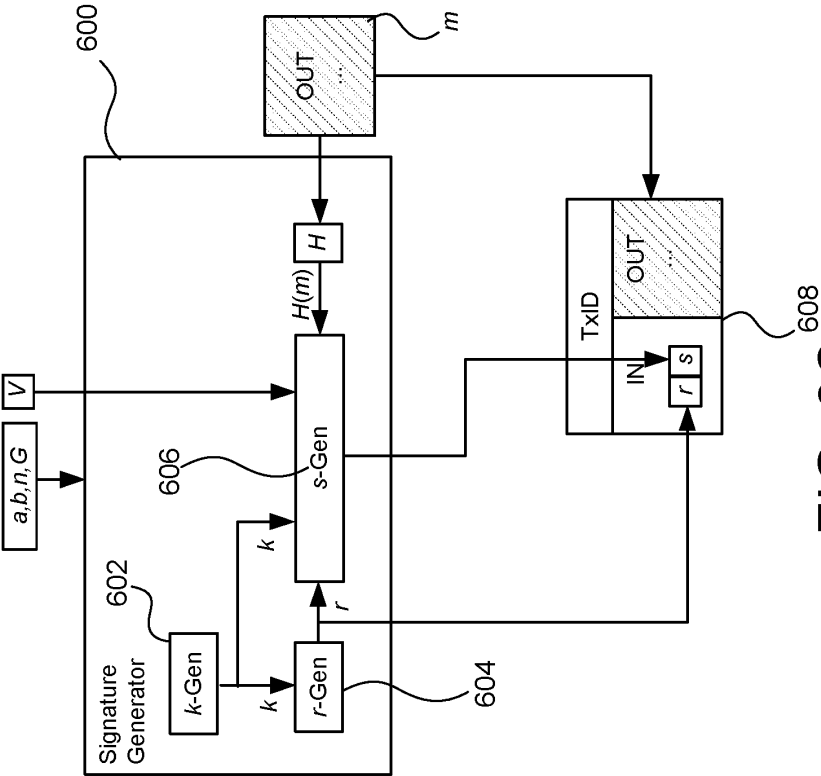


FIG. 6C

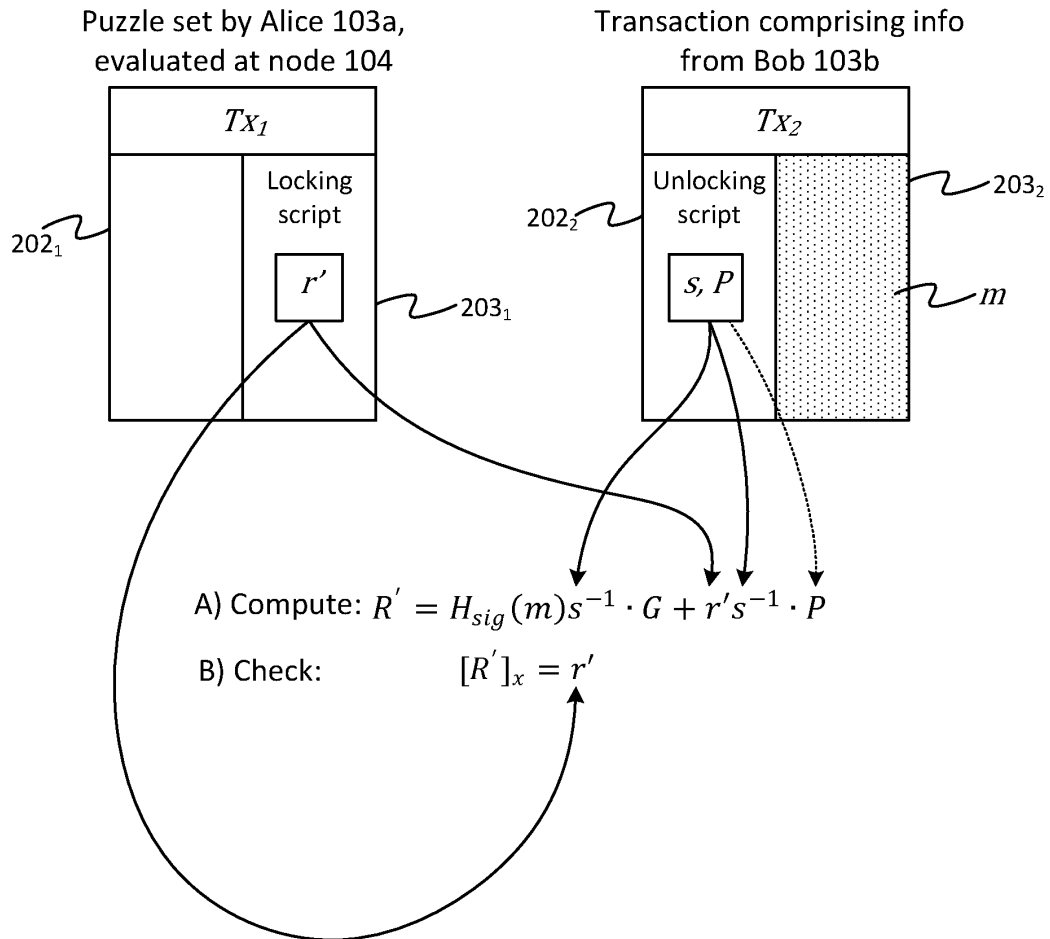
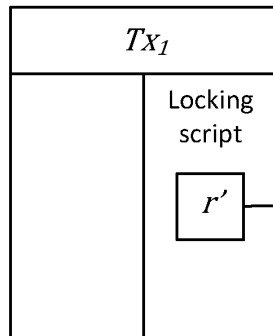
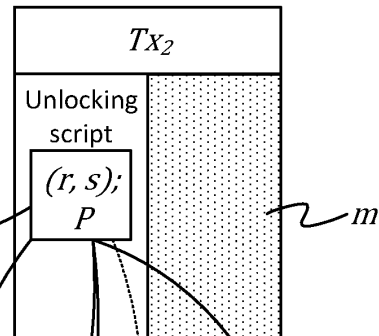
Figure 7

Figure 8

Puzzle set by Alice 103a,
evaluated at node 104



Transaction comprising info
from Bob 103b



I) Check:

II) Compute: $R' = H_{sig}(m)s^{-1} \cdot G + rs^{-1} \cdot P$

III) Check:

$$[R']_x = r$$

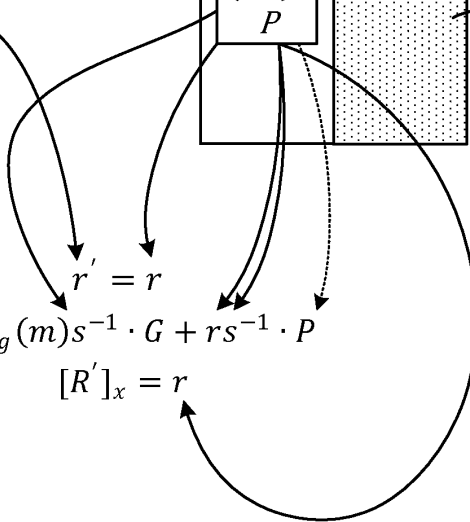
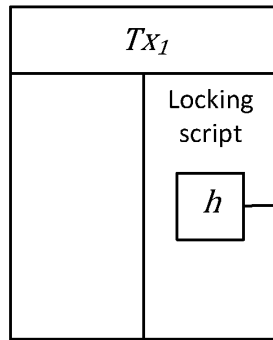


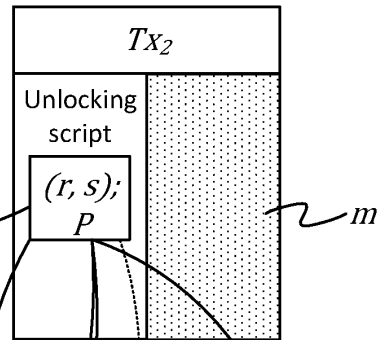
Figure 9

Puzzle set by Alice 103a,
evaluated at node 104



where $h = H_{puz}(r')$

Transaction comprising info
from Bob 103b



I) Check:

II) Compute: $R' = H_{sig}(m)s^{-1} \cdot G + rs^{-1} \cdot P$

III) Check:

$$[R']_x = r$$

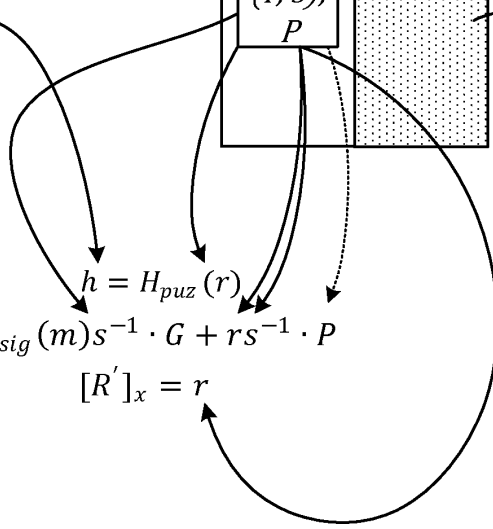
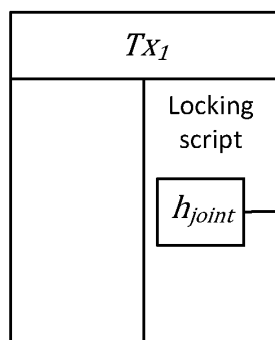


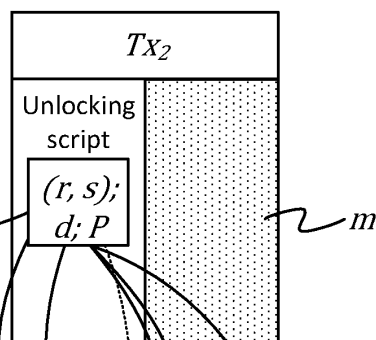
Figure 10

Puzzle set by Alice 103a,
evaluated at node 104



where $h_{joint} = H_{puz}(r' || d')$

Transaction comprising info
from Bob 103b



I) Check:

II) Compute: $R' = H_{sig}(m)s^{-1} \cdot G + rs^{-1} \cdot P$

III) Check:

$$[R']_x = r$$

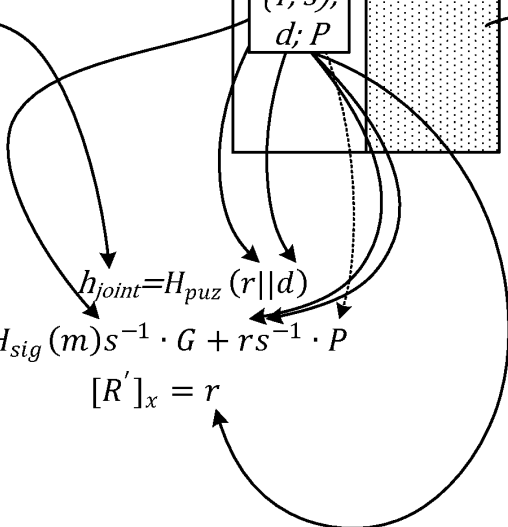


Figure 11

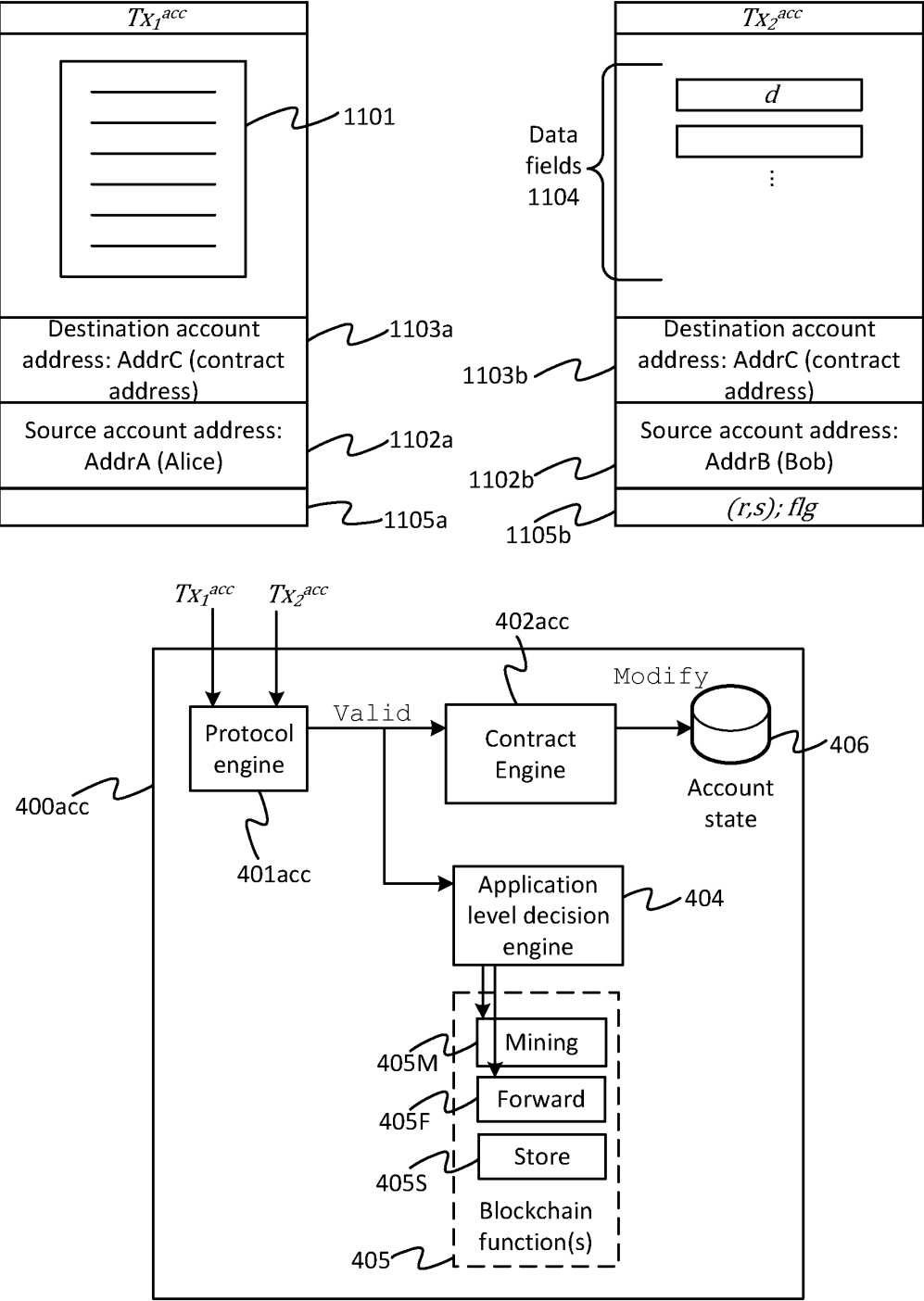
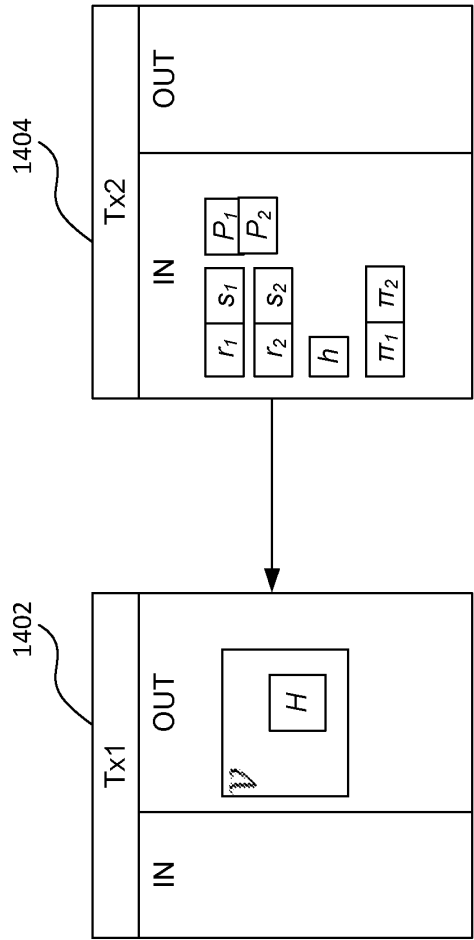


Figure 12



Figure 13

Stack	Script	Description
empty	$\langle P_B \rangle \langle sig_r \rangle$ OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	scriptSig and scriptPubKey are combined
$\langle P_B \rangle \langle sig_r \rangle$	OP_DUP OP_3 OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	Unlocking script constants pushed to stack
$\langle P_B \rangle \langle sig_r \rangle \langle sig_r \rangle$ OP_3	OP_SPLIT OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS (sig) duplicated and constant (3) pushed to stack
$\langle P_B \rangle \langle sig_r \rangle \langle sig_r \rangle [3:] \langle sig_r [3:] \rangle$	OP_NIP OP_1 OP_SPLIT OP_SWAP OP_SPLIT OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS (sig) split at the third byte, right before len(r)
$\langle P_B \rangle \langle sig_r \rangle \langle sig_r [3:] \rangle$ OP_1	OP_SPLIT OP_SWAP OP_SPLIT OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	Second to ToS item removed and constant (1) pushed to stack
$\langle P_B \rangle \langle sig_r \rangle \langle sig_r [4:] \rangle \langle sig_r [3:4] \rangle$	OP_SPLIT OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	ToS split 1 byte in (at byte 4 of sig) and the top two items on the stack are swapped
$\langle P_B \rangle \langle sig_r \rangle \langle sig_r [4:36/37] \rangle \langle sig_r [36/37:] \rangle$	OP_DROP (r) OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	The second to ToS (sig starting at index 4) is split at len(r)= $\langle sig_r [3:4] \rangle$ bytes in which will be either 36 or 37 since r will either be 32 or 33 bytes
$\langle P_B \rangle \langle sig_r \rangle \langle sig_r [4:36/37] \rangle$ (r)	OP_EQUALVERIFY OP_SWAP OP_CHECKSIG	Drop ToS (rest of sig) and push (r) to stack
$\langle sig_r \rangle \langle P_B \rangle$	OP_CHECKSIG	Check if $\langle sig_r [4:36/37] \rangle = (r)$
True	empty	Signature is checked for top two stack items



Hash Collision Bounties

FIG. 14

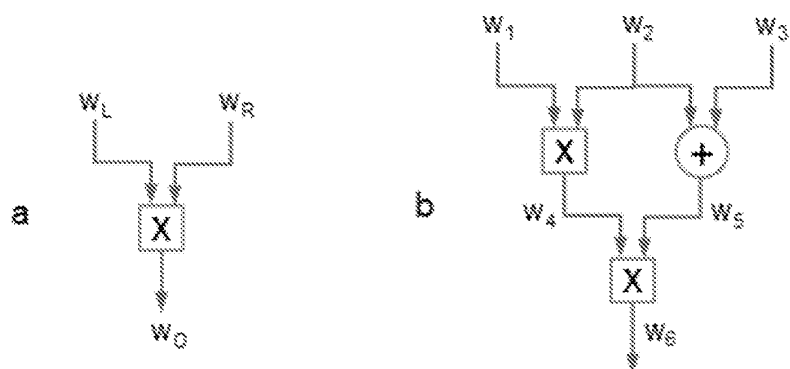


FIG. 15

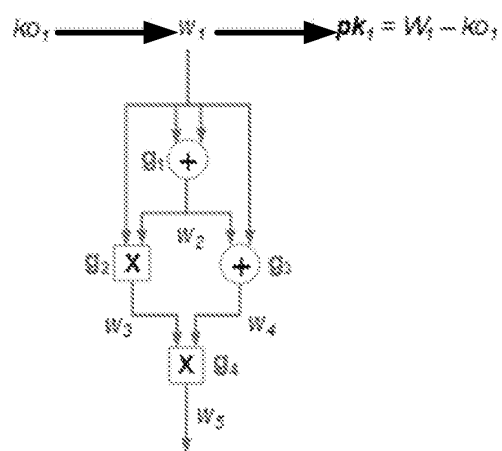


FIG. 16

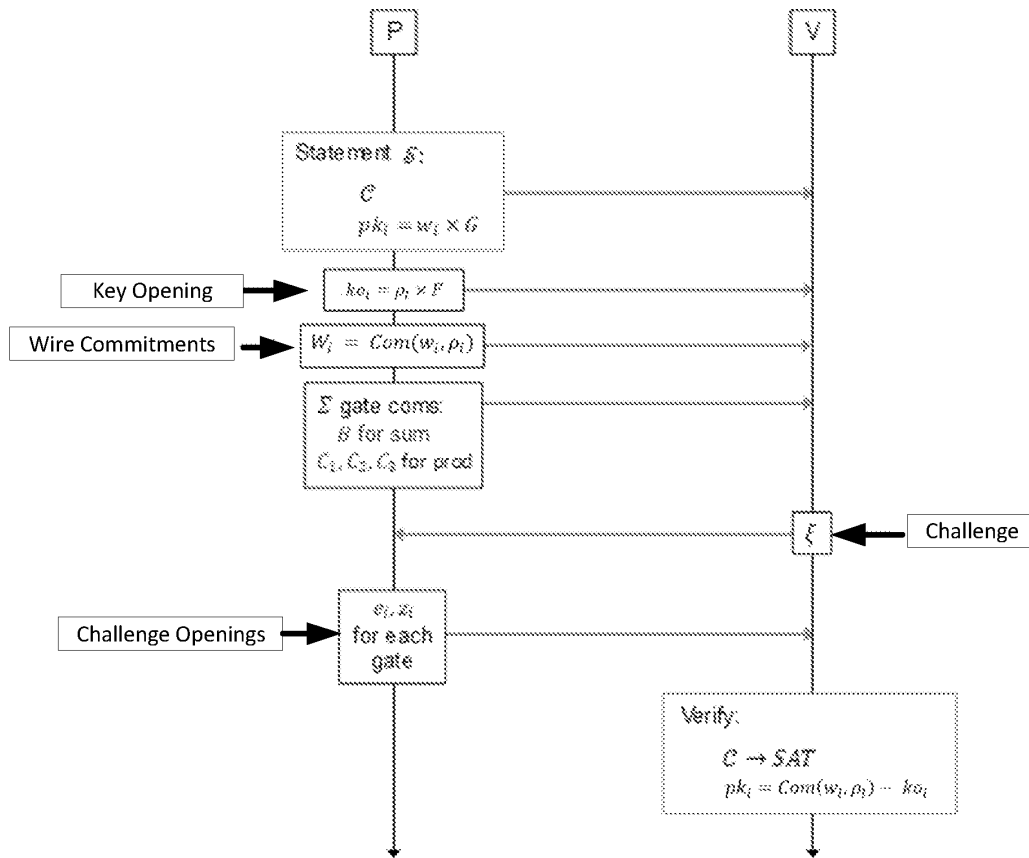


FIG. 17