We may then define the sender function $S$ of the transaction as:

$$S(T) \equiv \mathcal{B}_{96..255}\big(\texttt{KEC}\big(\texttt{ECDSARECOVER}(h(T), v, T_{\mathrm{r}}, T_{\mathrm{s}})\big)\big) \tag{321}$$

$$v \equiv \begin{cases} T_{\mathrm{w}} - 27 & \text{if } T_{\mathrm{x}} = 0 \wedge T_{\mathrm{w}} \in \{27, 28\} \\ (T_{\mathrm{w}} - 35) \bmod 2 & \text{if } T_{\mathrm{x}} = 0 \wedge T_{\mathrm{w}} \in \{2\beta + 35, 2\beta + 36\} \\ T_{\mathrm{y}} & \text{if } T_{\mathrm{x}} = 1 \end{cases} \tag{322}$$

The assertion that the sender of a signed transaction equals the address of the signer should be self-evident:

$$\forall T : \forall p_{\mathrm{r}} : S(G(T, p_{\mathrm{r}})) \equiv A(p_{\mathrm{r}}) \tag{323}$$

## Appendix G. Fee Schedule

The fee schedule $G$ is a tuple of scalar values corresponding to the relative costs, in gas, of a number of abstract operations that a transaction may effect.

| Name | Value | Description |
|---|---|---|
| $G_{\mathrm{zero}}$ | 0 | Nothing paid for operations of the set $W_{\mathrm{zero}}$. |
| $G_{\mathrm{jumpdest}}$ | 1 | Amount of gas to pay for a JUMPDEST operation. |
| $G_{\mathrm{base}}$ | 2 | Amount of gas to pay for operations of the set $W_{\mathrm{base}}$. |
| $G_{\mathrm{verylow}}$ | 3 | Amount of gas to pay for operations of the set $W_{\mathrm{verylow}}$. |
| $G_{\mathrm{low}}$ | 5 | Amount of gas to pay for operations of the set $W_{\mathrm{low}}$. |
| $G_{\mathrm{mid}}$ | 8 | Amount of gas to pay for operations of the set $W_{\mathrm{mid}}$. |
| $G_{\mathrm{high}}$ | 10 | Amount of gas to pay for operations of the set $W_{\mathrm{high}}$. |
| $G_{\mathrm{warmaccess}}$ | 100 | Cost of a warm account or storage access. |
| $G_{\mathrm{accesslistaddress}}$ | 2400 | Cost of warming up an account with the access list. |
| $G_{\mathrm{accessliststorage}}$ | 1900 | Cost of warming up a storage with the access list. |
| $G_{\mathrm{coldaccountaccess}}$ | 2600 | Cost of a cold account access. |
| $G_{\mathrm{coldsload}}$ | 2100 | Cost of a cold storage access. |
| $G_{\mathrm{sset}}$ | 20000 | Paid for an SSTORE operation when the storage value is set to non-zero from zero. |
| $G_{\mathrm{sreset}}$ | 2900 | Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero. |
| $R_{\mathrm{sclear}}$ | 15000 | Refund given (added into refund counter) when the storage value is set to zero from non-zero. |
| $R_{\mathrm{selfdestruct}}$ | 24000 | Refund given (added into refund counter) for self-destructing an account. |
| $G_{\mathrm{selfdestruct}}$ | 5000 | Amount of gas to pay for a SELFDESTRUCT operation. |
| $G_{\mathrm{create}}$ | 32000 | Paid for a CREATE operation. |
| $G_{\mathrm{codedeposit}}$ | 200 | Paid per byte for a CREATE operation to succeed in placing code into state. |
| $G_{\mathrm{callvalue}}$ | 9000 | Paid for a non-zero value transfer as part of the CALL operation. |
| $G_{\mathrm{callstipend}}$ | 2300 | A stipend for the called contract subtracted from $G_{\mathrm{callvalue}}$ for a non-zero value transfer. |
| $G_{\mathrm{newaccount}}$ | 25000 | Paid for a CALL or SELFDESTRUCT operation which creates an account. |
| $G_{\mathrm{exp}}$ | 10 | Partial payment for an EXP operation. |
| $G_{\mathrm{expbyte}}$ | 50 | Partial payment when multiplied by the number of bytes in the exponent for the EXP operation. |
| $G_{\mathrm{memory}}$ | 3 | Paid for every additional word when expanding memory. |
| $G_{\mathrm{txcreate}}$ | 32000 | Paid by all contract-creating transactions after the *Homestead* transition. |
| $G_{\mathrm{txdatazero}}$ | 4 | Paid for every zero byte of data or code for a transaction. |
| $G_{\mathrm{txdatanonzero}}$ | 16 | Paid for every non-zero byte of data or code for a transaction. |
| $G_{\mathrm{transaction}}$ | 21000 | Paid for every transaction. |
| $G_{\mathrm{log}}$ | 375 | Partial payment for a LOG operation. |
| $G_{\mathrm{logdata}}$ | 8 | Paid for each byte in a LOG operation's data. |
| $G_{\mathrm{logtopic}}$ | 375 | Paid for each topic of a LOG operation. |
| $G_{\mathrm{keccak256}}$ | 30 | Paid for each KECCAK256 operation. |
| $G_{\mathrm{keccak256word}}$ | 6 | Paid for each word (rounded up) for input data to a KECCAK256 operation. |
| $G_{\mathrm{copy}}$ | 3 | Partial payment for *COPY operations, multiplied by words copied, rounded up. |
| $G_{\mathrm{blockhash}}$ | 20 | Payment for each BLOCKHASH operation. |

## Appendix H. Virtual Machine Specification

When interpreting 256-bit binary values as integers, the representation is big-endian.

When a 256-bit machine datum is converted to and from a 160-bit address or hash, the rightwards (low-order for BE) 20 bytes are used and the leftmost 12 are discarded or filled with zeroes, thus the integer values (when the bytes are interpreted as big-endian) are equivalent.