over time, requires approximately the total compute power of the trustworthy portion of the mining peers.

Thus we are able to define the block header validity function $V(H)$:

$$
\begin{aligned}
(56)\quad V(H) \quad \equiv \quad & n \leqslant \frac{2^{256}}{H_{\mathrm{d}}} \wedge m = H_{\mathrm{m}} \quad \wedge \\
& H_{\mathrm{d}} = D(H) \quad \wedge \\
& H_{\mathrm{g}} \leq H_{\mathrm{l}} \quad \wedge \\
& H_{\mathrm{l}} < P(H)_{\mathrm{H_l}} + \left\lfloor \frac{P(H)_{\mathrm{H_l}}}{1024} \right\rfloor \quad \wedge \\
& H_{\mathrm{l}} > P(H)_{\mathrm{H_l}} - \left\lfloor \frac{P(H)_{\mathrm{H_l}}}{1024} \right\rfloor \quad \wedge \\
& H_{\mathrm{l}} \geqslant 5000 \quad \wedge \\
& H_{\mathrm{s}} > P(H)_{\mathrm{H_s}} \quad \wedge \\
& H_{\mathrm{i}} = P(H)_{\mathrm{H_i}} + 1 \quad \wedge \\
& \|H_{\mathrm{x}}\| \leq 32
\end{aligned}
$$

where $(n, m) = \mathtt{PoW}(H_{\not\mathbf{n}}, H_{\mathrm{n}}, \mathbf{d})$

Noting additionally that **extraData** must be at most 32 bytes.

5. Gas and Payment

In order to avoid issues of network abuse and to sidestep the inevitable questions stemming from Turing completeness, all programmable computation in Ethereum is subject to fees. The fee schedule is specified in units of *gas* (see Appendix G for the fees associated with various computation). Thus any given fragment of programmable computation (this includes creating contracts, making message calls, utilising and accessing account storage and executing operations on the virtual machine) has a universally agreed cost in terms of gas.

Every transaction has a specific amount of gas associated with it: **gasLimit**. This is the amount of gas which is implicitly purchased from the sender's account balance. The purchase happens at the according **gasPrice**, also specified in the transaction. The transaction is considered invalid if the account balance cannot support such a purchase. It is named **gasLimit** since any unused gas at the end of the transaction is refunded (at the same rate of purchase) to the sender's account. Gas does not exist outside of the execution of a transaction. Thus for accounts with trusted code associated, a relatively high gas limit may be set and left alone.

In general, Ether used to purchase gas that is not refunded is delivered to the *beneficiary* address, the address of an account typically under the control of the miner. Transactors are free to specify any **gasPrice** that they wish, however miners are free to ignore transactions as they choose. A higher gas price on a transaction will therefore cost the sender more in terms of Ether and deliver a greater value to the miner and thus will more likely be selected for inclusion by more miners. Miners, in general, will choose to advertise the minimum gas price for which they will execute transactions and transactors will be free to canvas these prices in determining what gas price to offer. Since there will be a (weighted) distribution of minimum acceptable gas prices, transactors will necessarily have a trade-off to make between lowering the gas price and maximising the chance that their transaction will be mined in a timely manner.

6. Transaction Execution

The execution of a transaction is the most complex part of the Ethereum protocol: it defines the state transition function Υ. It is assumed that any transactions executed first pass the initial tests of intrinsic validity. These include:

(1) The transaction is well-formed RLP, with no additional trailing bytes;
(2) the transaction signature is valid;
(3) the transaction nonce is valid (equivalent to the sender account's current nonce);
(4) the sender account has no contract code deployed (see EIP-3607 by Feist et al. [2021]);
(5) the gas limit is no smaller than the intrinsic gas, g_0, used by the transaction; and
(6) the sender account balance contains at least the cost, v_0, required in up-front payment.

Formally, we consider the function Υ, with T being a transaction and $\boldsymbol{\sigma}$ the state:

$$
(57)\qquad \boldsymbol{\sigma}' = \Upsilon(\boldsymbol{\sigma}, T)
$$

Thus $\boldsymbol{\sigma}'$ is the post-transactional state. We also define Υ^{g} to evaluate to the amount of gas used in the execution of a transaction, $\Upsilon^{\mathbf{l}}$ to evaluate to the transaction's accrued log items and Υ^{z} to evaluate to the status code resulting from the transaction. These will be formally defined later.

6.1. Substate.

Throughout transaction execution, we accrue certain information that is acted upon immediately following the transaction. We call this the *accrued transaction substate*, or *accrued substate* for short, and represent it as A, which is a tuple:

$$
(58)\qquad A \equiv (A_{\mathbf{s}}, A_{\mathbf{l}}, A_{\mathbf{t}}, A_{\mathrm{r}}, A_{\mathbf{a}}, A_{\mathbf{K}})
$$

The tuple contents include $A_{\mathbf{s}}$, the self-destruct set: a set of accounts that will be discarded following the transaction's completion. $A_{\mathbf{l}}$ is the log series: this is a series of archived and indexable 'checkpoints' in VM code execution that allow for contract-calls to be easily tracked by onlookers external to the Ethereum world (such as decentralised application front-ends). $A_{\mathbf{t}}$ is the set of touched accounts, of which the empty ones are deleted at the end of a transaction. A_{r} is the refund balance, increased through using the SSTORE instruction in order to reset contract storage to zero from some non-zero value. Though not immediately refunded, it is allowed to partially offset the total execution costs. Finally, EIP-2929 by Buterin and Swende [2020a] introduced $A_{\mathbf{a}}$, the set of accessed account addresses, and $A_{\mathbf{K}}$, the set of accessed storage keys (more accurately, each element of $A_{\mathbf{K}}$ is a tuple of a 20-byte account address and a 32-byte storage slot).

We define the empty accrued substate A^0 to have no self-destructs, no logs, no touched accounts, zero refund balance, all precompiled contracts in the accessed addresses, and no accessed storage:

$$
(59)\qquad A^0 \equiv (\varnothing, (), \varnothing, 0, \pi, \varnothing)
$$

where π is the set of all precompiled addresses.