# Engenharia reversa de uma transação via geth

TDC POA 2018

ASSEMBLEIA LEGISLATIVA
ESTADO DE GOIÁS
A CASA DO POVO

THE DEVELOPER'S CONFERENCE

# Júlio Campos

- Servidor Público
- Engenheiro de Software
- Agnóstico
- Ainda tentando terminar minha Pokedex

# CONCURSO DA ALEGO

Olá, candidato! Preparamos um site com as informações mais importantes sobre o concurso público Alego 2018. Aqui você encontra documentos como regimento interno, resoluções e não perde nenhum detalhe sobre a prova.

**Fique ligado e um bom concurso!**

## São VAGAS

Candidato,
Este canal foi criado para ajudá-lo a conhecer a Alego e se ambientar à estrutura e ao funcionamento da Casa. Todas as informações referentes ao edital, bem como os trâmites do concurso, são de responsabilidade exclusiva da banca organizadora, o Instituto Americano de Desenvolvimento (Iades).

*** No endereço www.iades.com.br você pode acessar os documentos oficiais do concurso e realizar os processos para isenção, inscrição e interposição de recursos, além de obter gabaritos, divulgação e homologação de resultados.**

## Quadro de Vagas
Assistente Legislativo

Remuneração inicial **R$ 5.789,37**
Taxa de inscrição: **R$ 80,00**

Mais detalhes e inscrição

| Categoria Funcional | Carga Horária | Vagas * | Total | Reserva * | Total |
|---|---|---|---|---|---|

* O valor entre parênteses representa a quantidade de vagas para pessoas com deficiência

Remuneração inicial **R$ 7.931,53**

## Blockchain

- Banco de dados

**+**

- Descentralizado

**=** **Lento**

**+**

- Distribuído

## Ethereum

## Ethereum

Uma plataforma descentralizada que executa contratos inteligentes, que são aplicações que são executadas exatamente como programadas sem possibilidade de downtime, censura, fraude ou interferência de terceiros.

Crunch
SF
SF
SF

# ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER
## BYZANTIUM VERSION 12779ac - 2018-11-27

DR. GAVIN WOOD
FOUNDER, ETHEREUM & PARITY
GAVIN@PARITY.IO

Abstract. The blockchain paradigm when coupled with cryptographically-secured transactions has demonstrated its utility through a number of projects, with Bitcoin being one of the most notable ones. Each such project can be seen as a simple application on a decentralised, but singleton, compute resource. We can call this paradigm a transactional singleton machine with shared-state.

Ethereum implements this paradigm in a generalised manner. Furthermore it provides a plurality of such resources, each with a distinct state and operating code but able to interact through a message-passing framework with others. We discuss its design, implementation issues, the opportunities it provides and the future hurdles we envisage.

## 1. Introduction

With ubiquitous internet connections in most places of the world, global information transmission has become incredibly cheap. Technology-rooted movements like Bitcoin have demonstrated through the power of the default, consensus mechanisms, and voluntary respect of the social contract, that it is possible to use the internet to make a decentralised value-transfer system that can be shared across the world and virtually free to use. This system can be said to be a very specialised version of a cryptographically secure, transaction-based state machine. Follow-up systems such as Namecoin adapted this original "currency application" of the technology into other applications albeit rather simplistic ones.

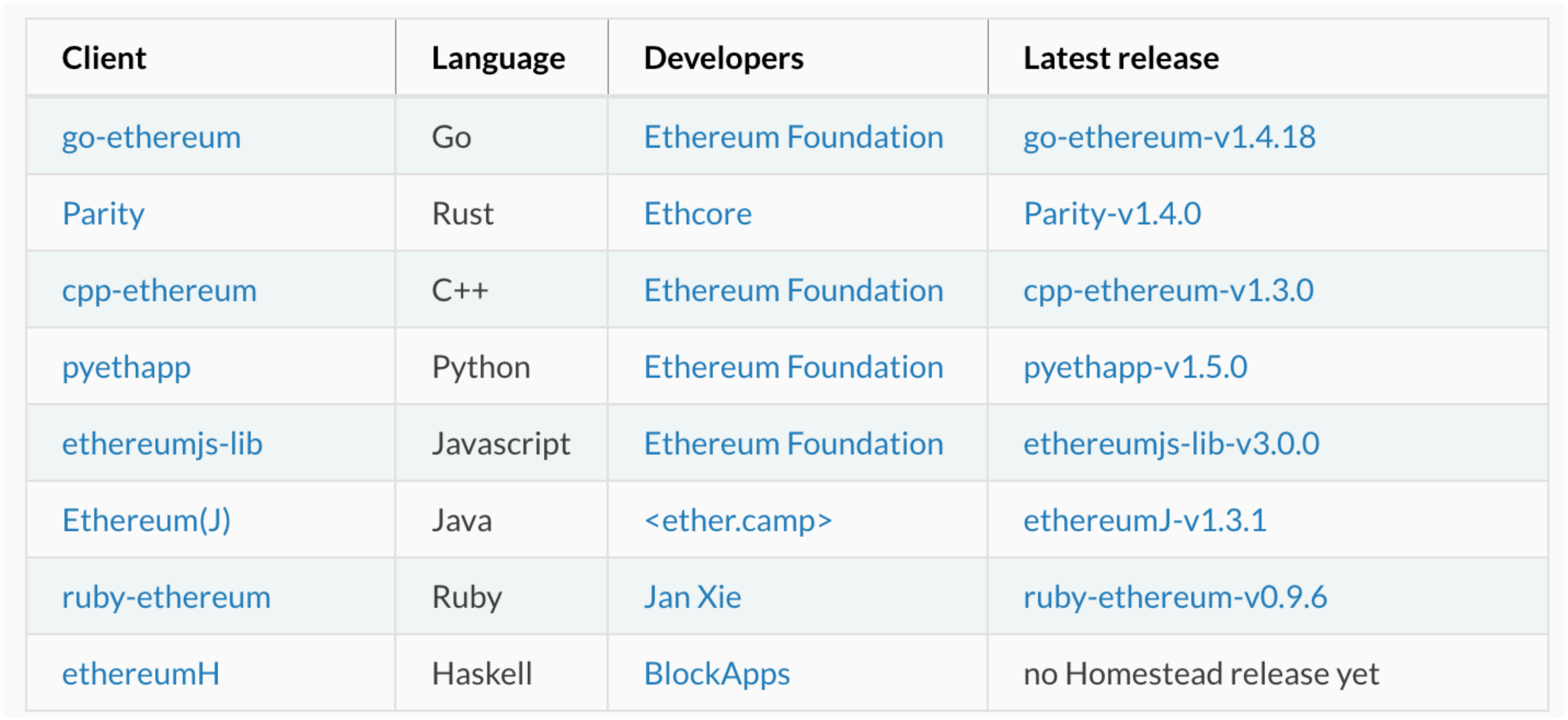
Ethereum is a project which attempts to build the generalised technology; technology on which all transaction-based state machine concepts may be built. Moreover it aims to provide to the end-developer a tightly integrated end-to-end system for building software on a hitherto unexplored compute paradigm in the mainstream: a trustful object messaging compute framework.

is often lacking, and plain old prejudices are difficult to shake.

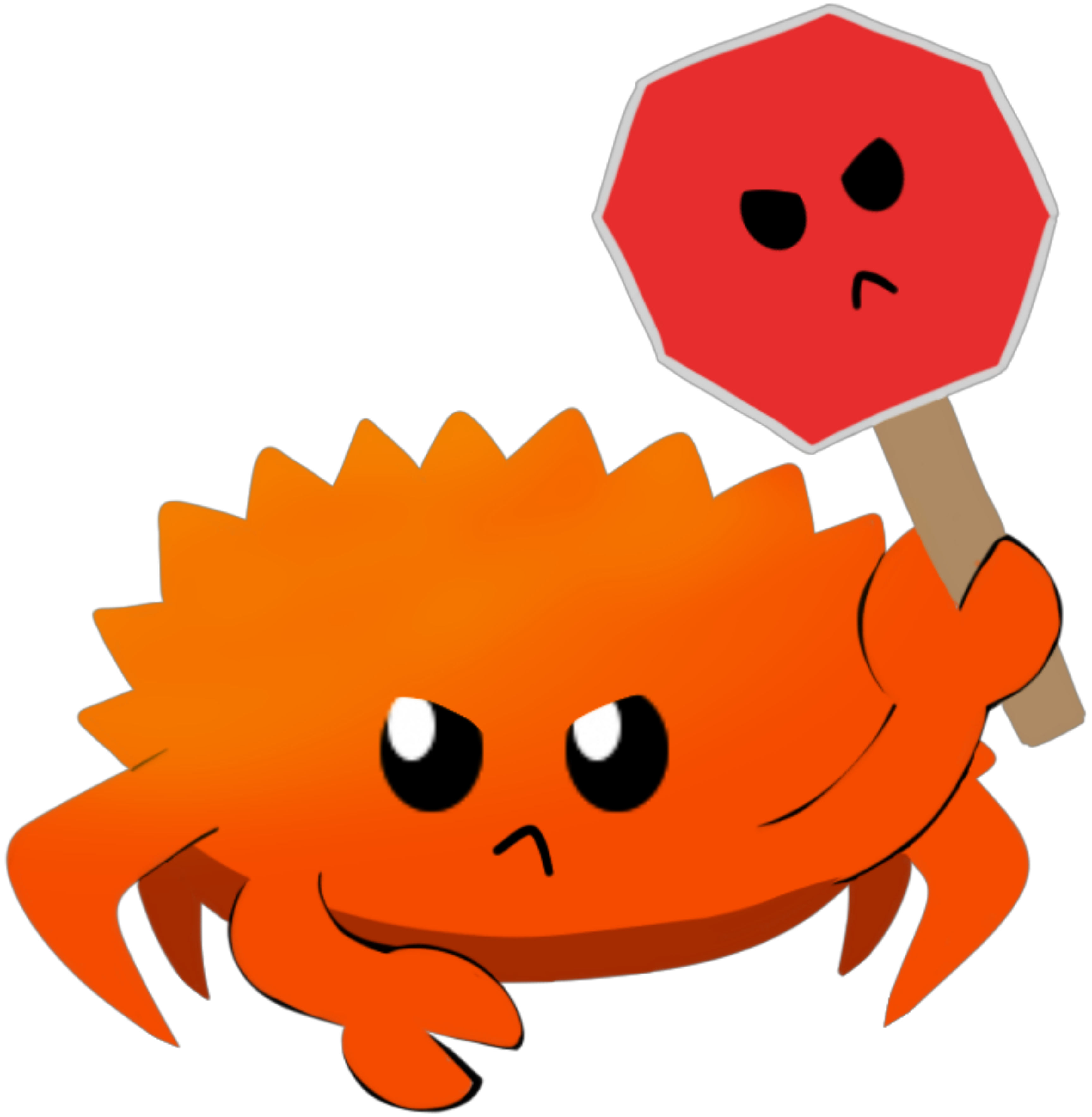
Overall, we wish to provide a system such that users can be guaranteed that no matter with which other individuals, systems or organisations they interact, they can do so with absolute confidence in the possible outcomes and how those outcomes might come about.

**1.2. Previous Work.** Buterin [2013a] first proposed the kernel of this work in late November, 2013. Though now evolved in many ways, the key functionality of a blockchain with a Turing-complete language and an effectively unlimited inter-transaction storage capability remains unchanged.

Dwork and Naor [1992] provided the first work into the usage of a cryptographic proof of computational expenditure ("proof-of-work") as a means of transmitting a value signal over the Internet. The value-signal was utilised here as a spam deterrence mechanism rather than any kind of currency, but critically demonstrated the potential for a basic data channel to carry a *strong economic signal*, allowing a receiver to make a physical assertion without



| Client | Language | Developers | Latest release |
|---|---|---|---|
| go-ethereum | Go | Ethereum Foundation | go-ethereum-v1.4.18 |
| Parity | Rust | Ethcore | Parity-v1.4.0 |
| cpp-ethereum | C++ | Ethereum Foundation | cpp-ethereum-v1.3.0 |
| pyethapp | Python | Ethereum Foundation | pyethapp-v1.5.0 |
| ethereumjs-lib | Javascript | Ethereum Foundation | ethereumjs-lib-v3.0.0 |
| Ethereum(J) | Java | <ether.camp> | ethereumJ-v1.3.1 |
| ruby-ethereum | Ruby | Jan Xie | ruby-ethereum-v0.9.6 |
| ethereumH | Haskell | BlockApps | no Homestead release yet |



## geth

Uma das três implementações originais do protocolo Ethereum.

transactions, creating new, valid, blocks is difficult and, 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 $V(H)$:

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

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

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

## 6. Transaction Execution

The execution of a transaction is the most complex part of the Ethereum protocol: it defines the state transition function $\Upsilon$. 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 gas limit is no smaller than the intrinsic gas, $g_0$, used by the transaction; and
(5) the sender account balance contains at least the cost, $v_0$, required in up-front payment.

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

$$(51)\qquad \boldsymbol{\sigma}' = \Upsilon(\boldsymbol{\sigma}, T)$$

Thus $\boldsymbol{\sigma}'$ is the post-transactional state. We also define $\Upsilon^{\mathrm{g}}$ to evaluate to the amount of gas used in the execution of a transaction, $\Upsilon^{\mathrm{l}}$ to evaluate to the transaction's accrued

log items and $\Upsilon^{\mathrm{z}}$ 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 *transaction substate*, and represent it as $A$, which is a tuple:

$$(52)\qquad A \equiv (A_{\mathbf{s}}, A_{\mathbf{l}}, A_{\mathbf{t}}, A_{\mathrm{r}})$$

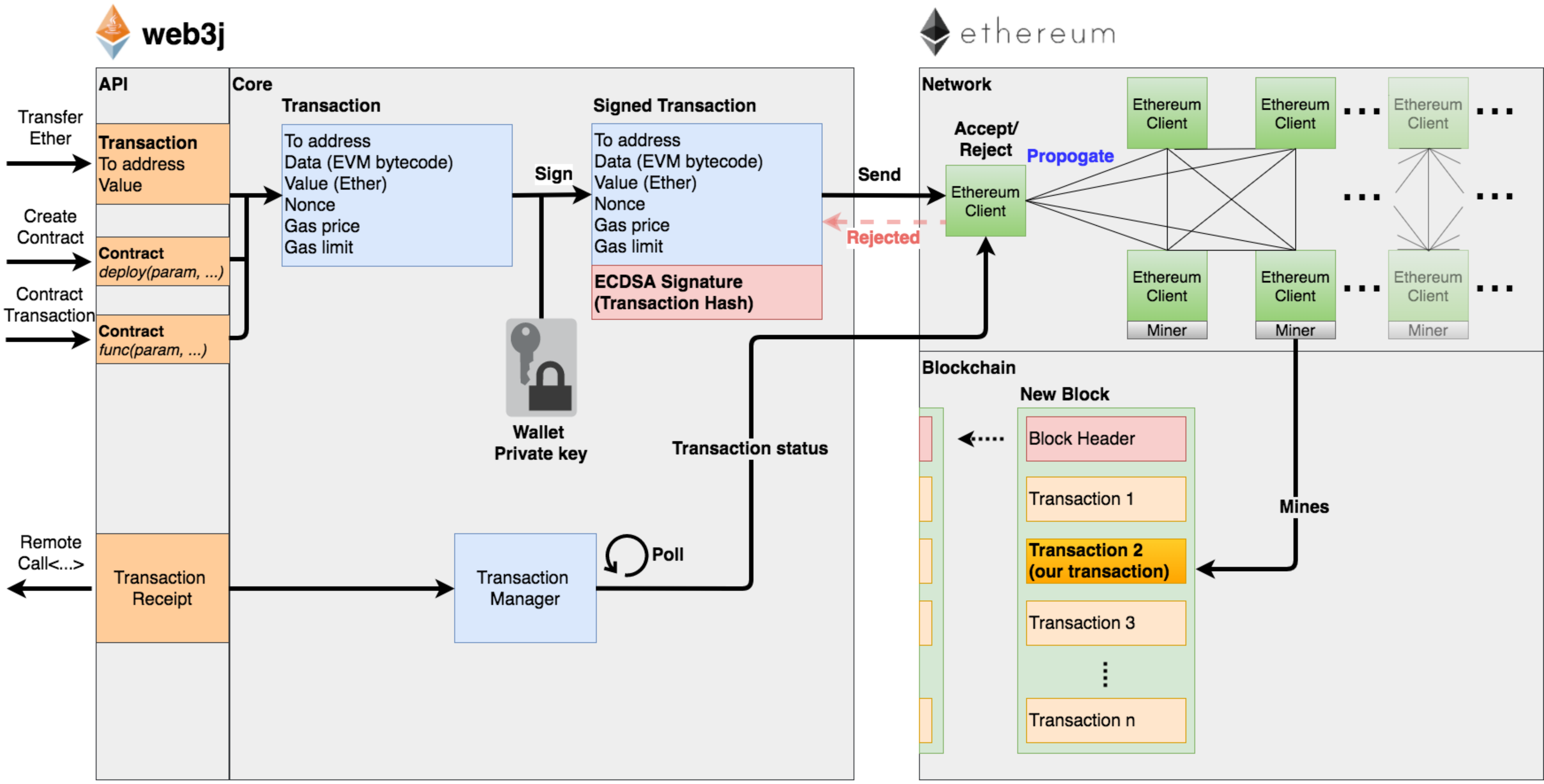
The tuple contents include $A_{\mathbf{s}}$, the self-destruct set: a set of accounts that will be discarded following the transaction's completion. $A_{\mathbf{l}}$ is the log series: this is a series of archived and indexable 'checkpoints' in VM code execu

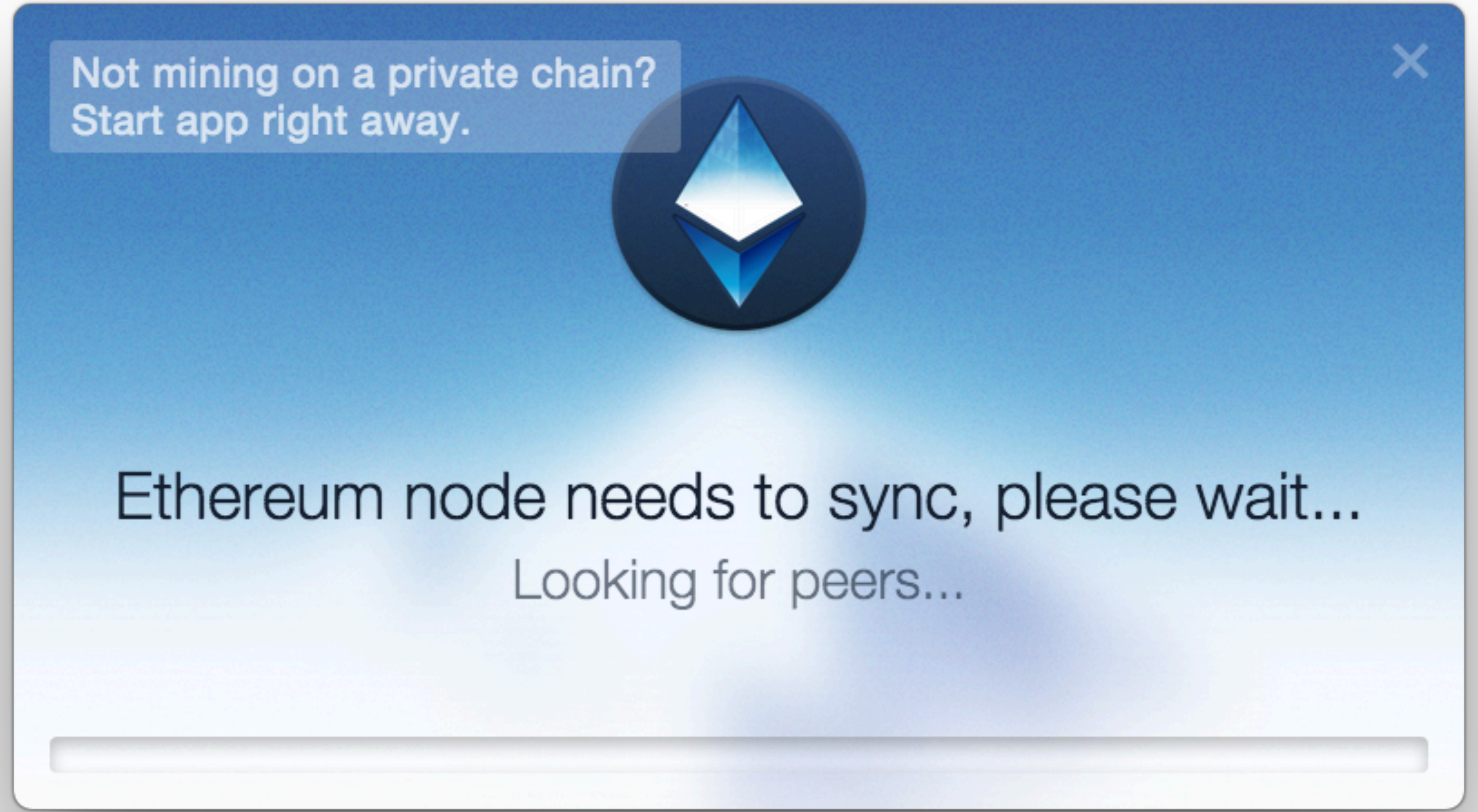
or a message call, results in an eventual state (which may legally be equivalent to the current state), the change to which is deterministic and never invalid: there can be no invalid transactions from this point.

We define the checkpoint state $\boldsymbol{\sigma}_0$:

$$
\begin{aligned}
(59)\qquad \boldsymbol{\sigma}_0 &\equiv \boldsymbol{\sigma} \quad \text{except:} \\
(60)\qquad \boldsymbol{\sigma}_0[S(T)]_{\mathrm{b}} &\equiv \boldsymbol{\sigma}[S(T)]_{\mathrm{b}} - T_{\mathrm{g}}T_{\mathrm{p}} \\
(61)\qquad \boldsymbol{\sigma}_0[S(T)]_{\mathrm{n}} &\equiv \boldsymbol{\sigma}[S(T)]_{\mathrm{n}} + 1
\end{aligned}
$$

Evaluating $\boldsymbol{\sigma}_P$ from $\boldsymbol{\sigma}_0$ depends on the transaction type; either contract creation or message call; we define the tuple of post-execution provisional state $\boldsymbol{\sigma}_P$, remaining


web3j
ethereum
API
Core
Transfer Ether
Transaction
To address
Value
Create Contract
Contract
deploy(param, ...)
Contract Transaction
Contract
func(param, ...)
Transaction
To address
Data (EVM bytecode)
Value (Ether)
Nonce
Gas price
Gas limit
Sign
Signed Transaction
To address
Data (EVM bytecode)
Value (Ether)
Nonce
Gas price
Gas limit
ECDSA Signature (Transaction Hash)
Wallet Private key
Send
Rejected
Network
Accept/ Reject
Propogate
Ethereum Client
Miner
Transaction status
Remote Call<...>
Transaction Receipt
Transaction Manager
Poll
Blockchain
New Block
Block Header
Transaction 1
Transaction 2 (our transaction)
Transaction 3
Transaction n
Mines



Not mining on a private chain?
Start app right away.

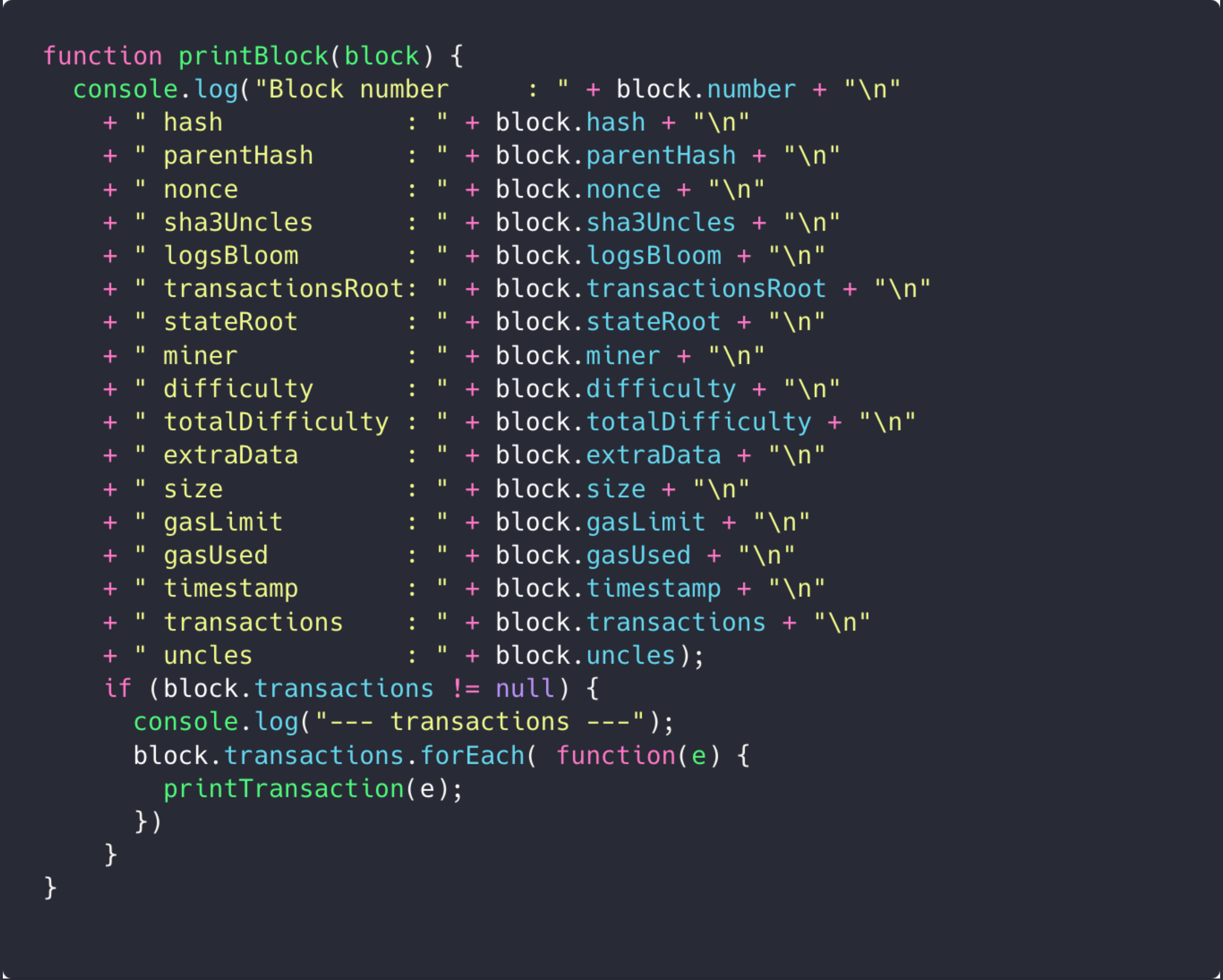
Ethereum node needs to sync, please wait...

Looking for peers...

## JSRE

## Otto

Interpretador de JavaScript para Go



```javascript
function printBlock(block) {
  console.log("Block number     : " + block.number + "\n"
    + " hash            : " + block.hash + "\n"
    + " parentHash      : " + block.parentHash + "\n"
    + " nonce           : " + block.nonce + "\n"
    + " sha3Uncles      : " + block.sha3Uncles + "\n"
    + " logsBloom       : " + block.logsBloom + "\n"
    + " transactionsRoot: " + block.transactionsRoot + "\n"
    + " stateRoot       : " + block.stateRoot + "\n"
    + " miner           : " + block.miner + "\n"
    + " difficulty      : " + block.difficulty + "\n"
    + " totalDifficulty : " + block.totalDifficulty + "\n"
    + " extraData       : " + block.extraData + "\n"
    + " size            : " + block.size + "\n"
    + " gasLimit        : " + block.gasLimit + "\n"
    + " gasUsed         : " + block.gasUsed + "\n"
    + " timestamp       : " + block.timestamp + "\n"
    + " transactions    : " + block.transactions + "\n"
    + " uncles          : " + block.uncles);
    if (block.transactions != null) {
      console.log("--- transactions ---");
      block.transactions.forEach( function(e) {
        printTransaction(e);
      })
    }
}
```

```
geth --jspath "/tmp" --exec ‘loadScript("printBlock")' attach
http://123.123.123.123:8545
```

Como salvar isso?

```go
type JSRE struct {
    assetPath     string
    output        io.Writer
    evalQueue     chan *evalReq
    stopEventLoop chan bool
    closed        chan struct{}
}

func New(assetPath string, output io.Writer) *JSRE {
    re := &JSRE{
        assetPath:     assetPath,
        output:        output,
        closed:        make(chan struct{}),
        evalQueue:     make(chan *evalReq),
        stopEventLoop: make(chan bool),
    }
    go re.runEventLoop()
    re.Set("loadScript", re.loadScript)
    re.Set("inspect", re.prettyPrintJS)
    return re
}
```

```go
func (self *JSRE) writeFile(call otto.FunctionCall) otto.Value {
    file, err := call.Argument(0).ToString()


    if err != nil {
        return otto.FalseValue()
    }

    data, err := call.Argument(1).ToString()


    if err != nil {
        return otto.FalseValue()
    }

    err = ioutil.WriteFile(file,[]byte(data), 0644)

    if err != nil {
        return otto.FalseValue()
    }

    return otto.TrueValue()
}
```

```go
func New(assetPath string, output io.Writer) *JSRE {
	re := &JSRE{
		assetPath:     assetPath,
		output:        output,
		closed:        make(chan struct{}),
		evalQueue:     make(chan *evalReq),
		stopEventLoop: make(chan bool),
	}
	go re.runEventLoop()
	re.Set("loadScript", re.loadScript)
	re.Set("writeFile", re.writeFile)
	re.Set("inspect", re.prettyPrintJS)
	return re
}
```

```
function checkAllBalances() {
 var i =0;
 eth.accounts.forEach( function(e){
    writeFile(dir+"e.json",stringifyJSON("eth.accounts["+i+"]: "
                +  e + " \tbalance: "
                +    web3.fromWei(eth.getBalance(e), "ether")
                + " ether");
  i++;
 })
};
```

Obrigado

# Contatos

{ ; )

**Telegram**

**@GuildaTech**

**Twitter**

**@JCSerraCampos**

**Github**

**jcserracampos**

**LinkedIn**

**/in/jcserracampos**

STRIKE
52
UNIVERSIDAD
GRANADA

# Referências

- https://github.com/ethereum/go-ethereum
- https://geth.ethereum.org/
- http://www.ethdocs.org/en/latest/ethereum-clients/choosing-a-client.html
- https://ethereum.github.io/yellowpaper/paper.pdf
- https://github.com/robertkrimen/otto