EIP 792

The standard for Arbitra(tion/ble) smart contracts.

A Primer on Blockchains

They:

A Primer on Blockchains

They:

- Are distributed data stores that can be used to store any sort of data (e.g. Bitcoin stores balances for a bunch of addresses.).

A Primer on Blockchains

They:

- Are distributed data stores that can be used to store any sort of data (e.g. Bitcoin stores balances for a bunch of addresses.).

- Can be edited as long as certain rules are followed (e.g. No double spending, no spending of others' funds, etc.).

A Primer on Blockchains

They:

- Are distributed data stores that can be used to store any sort of data (e.g. Bitcoin stores balances for a bunch of addresses.).

- Can be edited as long as certain rules are followed (e.g. No double spending, no spending of others' funds, etc.).

- Guarantee provable data integrity, commonly through different consensus protocols based on `cryptography + economics = cryptoeconomics`.

Smart? Contracts

They:

Smart? Contracts

They:

- Are turing complete, stateful programs that get sent in a special deployment TX and become immutable.
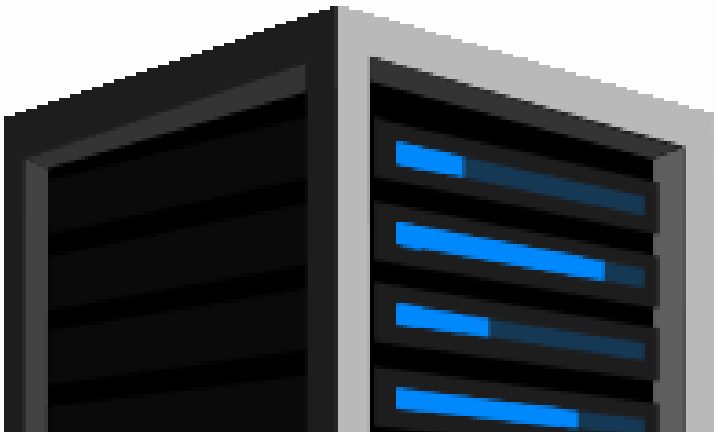
Smart? Contracts

They:

- Are turing complete, stateful programs that get sent in a special deployment TX and become immutable.

- Have their own address to which you can send TXs with some input data and value and have some logic executed on-chain.

Smart? Contracts

They:

- Are turing complete, stateful programs that get sent in a special deployment TX and become immutable.

- Have their own address to which you can send TXs with some input data and value and have some logic executed on-chain.

- Guarantee provable data and state transition integrity through the chain they live on, in our case, Ethereum.

A Note on Smart Contract Design

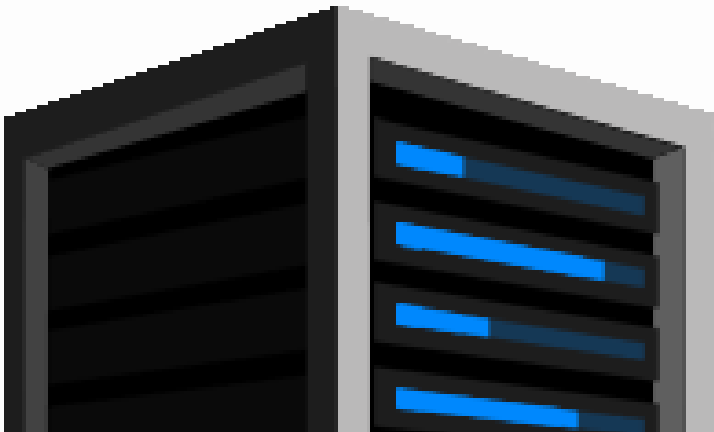# A Note on Smart Contract Design

Traditional Back Ends:

- Do most of the heavy lifting for the front end, because computation is cheap and faster than on the client.
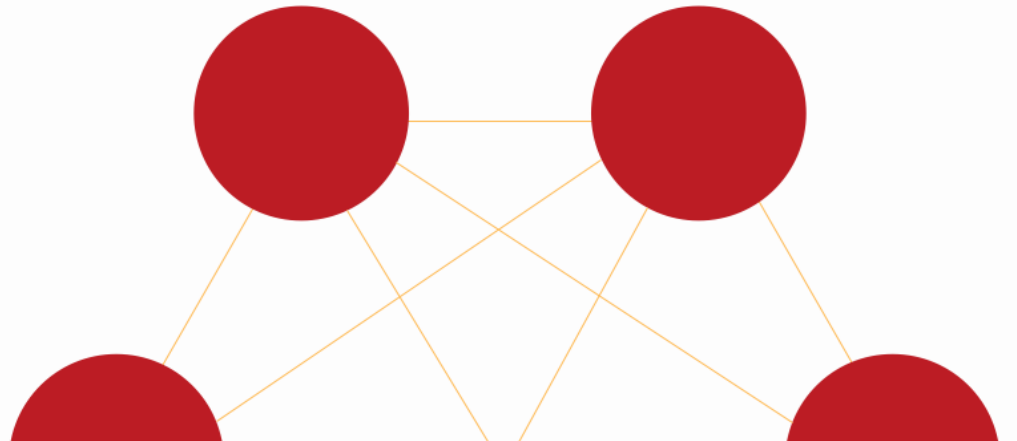
A Note on Smart Contract Design

Traditional Back Ends:

- Do most of the heavy lifting for the front end, because computation is cheap and faster than on the client.

Smart Contracts Back Ends:

- Leave most of the heavy lifting for the front end, because computation is expensive and slower than on the client.

EIP 792 Arbitrator: Interface

```
contract Arbitrator {
    function createDispute(
        uint _choices, bytes _extraData
    ) public requireArbitrationFee(_extraData) payable returns(uint disputeID) {};

    function arbitrationCost(bytes _extraData) public view returns(uint fee);

    function appeal(
        uint _disputeID, bytes _extraData
    ) public requireAppealFee(_disputeID,_extraData) payable {
        emit AppealDecision(_disputeID, Arbitrable(msg.sender));
    }

    function appealCost(uint _disputeID, bytes _extraData) public view returns(uint fee);

    function appealPeriod(uint _disputeID) public view returns(uint start, uint end) {}

    function disputeStatus(uint _disputeID) public view returns(DisputeStatus status);

    function currentRuling(uint _disputeID) public view returns(uint ruling);
}
```

EIP 792 Arbitrator: Types and Events

```
contract Arbitrator {
    enum DisputeStatus { Waiting, Appealable, Solved }

    event DisputeCreation(uint indexed _disputeID, Arbitrable indexed _arbitrable);

    event AppealPossible(uint indexed _disputeID, Arbitrable indexed _arbitrable);

    event AppealDecision(uint indexed _disputeID, Arbitrable indexed _arbitrable);
}
```

EIP 792 Arbitrable: Interface and Events

```
contract Arbitrable {
    event Ruling(Arbitrator indexed _arbitrator, uint indexed _disputeID, uint _ruling);

    function rule(uint _disputeID, uint _ruling) public onlyArbitrator;
}
```

Evidence Standard:

Link

Composed Arbitrable Contracts:

Link