whitehorses

member of the human network

WHITEBOOK

# Blockchain: smart contracts in Ethereum

## April 2018

**Auteur:**
Laurens van der Starre
INTEGRATIE SPECIALIST

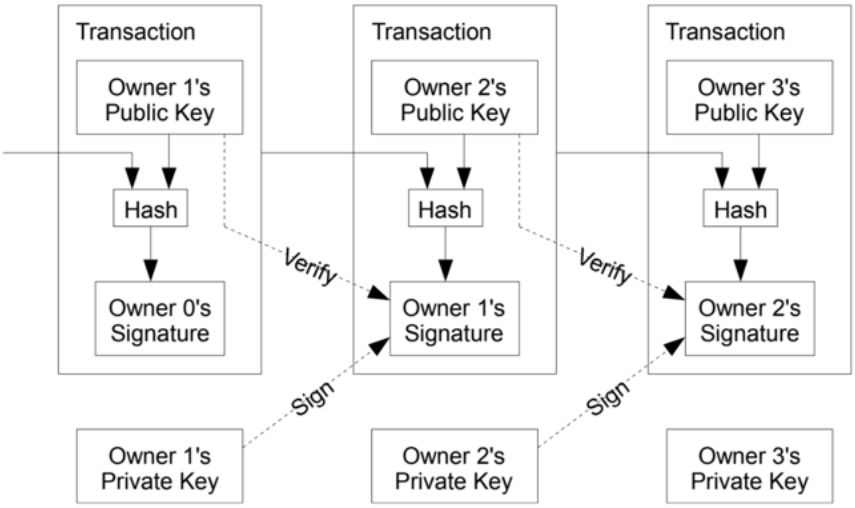
# Introductie

Je kan er niet meer om heen. Overal buzzt het van de termen als cryptovaluta, blockchain en bitcoin. Wat voor de één een grote zeepbel lijkt, schijnt de ander niet te deren en stopt er heel veel geld in. Wat zeker is, is dat de blockchain technologie één van de meest disruptieve technologieën van dit moment is, en bedrijven kunnen niet wachten om er een toepassing voor te vinden in hun bedrijfsvoering.

# Wat is een blockchain?

Een blockchain is onderdeel van een gedistribueerd netwerk waarin virtuele valuta wordt gegenereerd. Deze valuta kan worden overgemaakt tussen zgn. accounts, en deze transacties worden als blokken aan elkaar gelinkt: een ketting van blokken, de blockchain. Omdat er geen centrale autoriteit is die de transacties controleert, moeten alle nodes in het netwerk samen de consensus hebben over de waarheid van de transacties. Kan de transactie wel plaatsvinden, of is de valuta al uitgegeven? Het is daarom belangrijk dat de gehele blockchain bekend is, dat deze kan worden gecontroleerd en niet kan worden aangepast.
Hiervoor heeft een account sleutels: een privé sleutel en een publieke sleutel, een soort van PKI (Public Key Infrastructure) die we bijvoorbeeld kennen uit TLS/SSL. Als er een transactie plaatsvindt, ontstaat er een nieuw block. De privé sleutel van het verzendende account wordt gebruikt om het nieuwe block in de chain te voorzien van een handtekening. De link met het voorgaande block in de chain wordt gelegd door een hash te berekenen op basis van het huidige block en eigenschappen van het vorige block. Er worden speciale eisen gesteld aan deze hash, waardoor het maken van deze hash een significantie rekenkracht vereist. Dit is de zgn. Proof of Work. Als een kwaadwillende node een blockchain wil aanpassen om zo een transactie aan te passen, betekent dit dat alle hashes in de hele keten moeten worden overgedaan. Zolang er meer eerlijke “rekenkracht” dan fraudulente in het netwerk bestaat zal er geen fraude in de blockchain kunnen optreden.

Een blockchain kan worden weergegeven als in Figuur 1 (ontleend uit 10)




*Figuur 1 Een blockchain*

Een blockchain is dus eigenlijk niets anders dan een gedistribueerd grootboek. Dit grootboek leeft in het grote peer-to-peer netwerk en is openbaar voor iedereen. Een blockchain bevat records die aan elkaar gelinkt zijn. Op deze manier ontstaat er een geverifieerde gedistribueerde transactiedatabase.

## Smart contracts

Nu is een gedistribueerd grootboek een knap stukje techniek, maar wat bepaalde blockchains bijzonder maakt zijn de zgn. smart contracts. Een smart contract is code in een account en is middels een API beschikbaar in het blockchain netwerk om te worden uitgevoerd. Een smart contract is eigenlijk een soort van service met een API die aan te roepen is. In dit Whitebook zoomen we in op de smart contracts van het Ethereum netwerk.

Ethereum is een populaire cryptovaluta gebaseerd op het blockchain concept. Ethereum kan code executeren in de Ethereum Vituele Machine (EVM). Deze virtual machine is qua concept vergelijkbaar met een Java Virtuele Machine. Programmacode wordt omgezet in bytecode die vervolgens in een sandboxed omgeving uitgevoerd wordt in de EVM.  Omdat elke node in het netwerk de gehele blockchain heeft, wordt een mutatie door of in een smart contract ook op elke node uitgevoerd. Een mutatie is immers een transactie en wordt dus onderdeel van de blockchain.

Ethereum is een zgn. Turing Complete blockchain. Dit betekent grofweg dat elke berekening of gegevensbewerking die geprogrammeerd kan worden, ook in dit systeem geprogrammeerd kan worden. Het is dus mogelijk volwaardige programmatuur te maken in de Ethereum blockchain die draait in een enorm groot gedistribueerd netwerk.

## Wat kost het?

Het runnen van de code van smart contracts, het opslaan van de data die deze contracts persisteren, de rekenkracht die deze contracts vereisen en het berekenen van de hash-codes in de chain kost uiteindelijk allemaal geld voor de eigenaar van de node. Daarom vraagt het Ethereum-netwerk transactiekosten, het zgn. Gas. Alle operaties van een transactie verbruiken een bepaalde hoeveelheid brandstof (Gas). Gas wordt betaald in Ether, de valuta van het Ethereum netwerk. Degene die het block verwerkt van de transactie, de miner, ontvangt de Gas, en dus Ether, in zijn account.

Dit betekent direct dat bij het maken van een smart contract je jezelf constant moet realiseren dat elke operatie, variabele die je zet, berekeningen die je maakt etc, Gas kost. En met de Ethereum prijzen die op het moment van schrijven rond de 800 euro liggen kan dat een duur grapje worden: per transactie op je smart contract kan je zo maar 10 tot 30 cent kwijt zijn.

De Ethereum block chain is groot en krachtig, maar ook openbaar. Grote applicaties bouwen met smart contracts kan een dure aangelegenheid worden. Gelukkig is Ethereum open source en kan je eenvoudig een eigen Ethereum block chain beginnen. Elke KA-computer is een potentiele node en miner, dus kan in theorie de hele kantoortuin een groot block chain netwerk worden. Afnemers van smart contracts in dit netwerk zijn gebonden aan hun IT-budget in Ether en het infrastructuur beheer kan hun budget verdienen door de mining.

# Voorbeeld

Om het fenomeen van een smart contract te laten leven gebruik ik in dit Whitebook het voorbeeld van de zgn. strippenkaart van Whitehorses. Deze strippenkaart is vergelijkbaar met de strippenkaart voor het openbaar vervoer van vroeger. Een reisje in de bus kostte een X-aantal strippen die door de buschauffeur werden afgestempeld. De Whitehorses strippenkaart stelt een klant in staat consulting uren in te kopen. Deze kunnen vervolgens worden verzilverd wanneer een consultant werk verricht voor deze klant. Ik zal laten zien hoe dit als een smart contract kan worden geïmplementeerd in het Ethereum netwerk. Dit contract is geschreven in de programmeertaal Solidity.

De functionaliteit:

1. De klant koopt uren in voor de consultant op een strippenkaart a x 1/6 ETH per uur.
   Operatie: koopStrippen.
   De strippenkaart wordt aangemaakt en toegewezen aan de klant. Deze operatie wordt aangeroepen en de Ether wordt direct overgemaakt naar de strippenkaart.
2. De eigenaar van de Strippenkaart kan de wallet van de consultant en Whitehorses zetten met de operatie zetConsultant. Een wallet binnen een blockchain is de portemonnee van een account in het Ethereum netwerk waarin Ether kan worden opgeslagen.
3. Consultant claimt werk: afstempelenStrippen
   Het geld, 80%, wordt overgeboekt naar de consultant, 20% gaat naar Whitehorses.

Het gaat te ver om de gehele code in detail te bespreken. De belangrijkste elementen zijn:

- Je roept de functionaliteit van een contract aan vanaf je account in het Ethereum-netwerk. Dit betekent dat de vereiste Gas-kosten uit dit account worden betaald.
- Sommige functionaliteit in een smart contract is alleen voor de eigenaar van het contract. Dit is af te dwingen door een functie modifier toe te passen op de desbetreffende functions. Deze modifier injecteert code (die het eigenaarschap afdwingt bijvoorbeeld) in de function.
- Payable-functions vereisen dat de aanroeper Ether meestuurt in het request. Deze Ether zal bij het smartcontract worden bijgeschreven.
- Veel operaties zijn fire & forget in Solidity. In de huidige versie is nog niet overal een consistente methode doorgevoerd voor asynchrone callbacks. Soms is er sprake van een zgn. promise-implementatie, vaak (nog) niet. Een manier om terugkoppeling te krijgen is om een Event te gooien. Een event is een soort van log-regel in de blockchain, die vervolgens uitgelezen kan worden. Op deze manier kan je bijvoorbeeld melding geven over het succesvol uitvoeren van een actie in de code.

Laten we kijken naar het aanmaken en draaien van ons smart contract (De volledige code staat aan het einde van dit Whitebook). We kunnen het contract in een lokale Ethereum node draaien en testen in een commandline console met de illustere naam truffle :

Na het compileren en deployen van het contract wordt de constructor van het contract gedraaid. In deze constructor zetten we het eigenaarschap: de verzender van de transactie (de aanmaker van het contract in de block chain).

```
  function Strippenkaart() public {
    // constructor
    owner = msg.sender;
  }
Vervolgens willen we natuurlijk strippen kopen:
  function koopStrippen (string _klantNaam, uint _aantalStrippen) public
payable {
    // kan alleen bij nieuwe strippenkaart, of opwaarderen
    require(msg.sender == klant || klant == 0);
    require(msg.value >= _aantalStrippen * UURLOON);

    // nieuwe Strippenkaart
    if (klant == 0) {
      klant = msg.sender;
    }

    aantalStrippen += _aantalStrippen;
    klantNaam = _klantNaam;
    strippenGekochtEvent(klantNaam, _aantalStrippen);

  }
```

Hier zien we dat deze functie alleen kan worden aangeroepen door de klant. Als deze klant nog niet bestaat (in het geval van een nieuwe strippenkaart) zal de eerste aanroeper van deze functie de klant worden. Ook wordt een bepaalde hoeveelheid Ether verwacht: het aantal strippen maal het uurloon.

Aan het einde van procedure gooien we een event waarin we aangeven dat een klant een aantal strippen heeft gekocht. Dit event gaan we terugzien in truffle.
In truffle instantiëren we allereerst het contract als de referentie “app”:

```
truffle(development)>
Strippenkaart.deployed().then(function(instance) {app = instance; })
```

En vervolgens abonneren we ons op het event.

```
truffle(development)> strippenGekochtEvent =
app.strippenGekochtEvent().watch (function(error, event)
{console.log(event); })
```

Dan gaan we met een ingebouwd account van de testomgeving (accounts[1]) een strip kopen op de strippenkaart. We noemen de klant “Laurens BV”. In truffle geven we nu de afzender plus een bedrag van 1/6 Ether mee aan de aanroep naar de operatie koopStrippen. In Solidity worden bedragen in Wei uitgedrukt. 1 Ether staat gelijk aan 1 * 1018 Wei.

```
truffle(development)>

truffle(development)> app.koopStrippen("Laurens BV", 1, {from : web3.eth.
accounts[1], value: 160000000000000000})
{ tx: '0x6e0512ab90505a4aedfe77a724757c7c870a45d472602c2fbf556642b6b5c1c6',
  receipt:
   { transactionHash:
'0x6e0512ab90505a4aedfe77a724757c7c870a45d472602c2fbf556642b6b5c1c6',
     transactionIndex: 0,
     blockHash:
'0x83bf7cbd0cbf90eb57c04b008340dae7d965c5930526eff9141c29264b94c764',
     blockNumber: 5,
     gasUsed: 87374,
     cumulativeGasUsed: 87374,
     contractAddress: null,
     logs: [ [Object] ],
     status: 1 },
  logs:
   [ { logIndex: 0,
       transactionIndex: 0,
       transactionHash:
'0x6e0512ab90505a4aedfe77a724757c7c870a45d472602c2fbf556642b6b5c1c6',
       blockHash:
'0x83bf7cbd0cbf90eb57c04b008340dae7d965c5930526eff9141c29264b94c764',
       blockNumber: 5,
       address: '0x33d5fd32a27189268891640a1bce91a132ebc947',
       type: 'mined',
       event: 'strippenGekochtEvent',
       args: [Object] } ] }
```

Interessant om te zien is dat we in de output het block zien dat gecreëerd is. Ook is de Gas-prijs te zien (in giga-Wei ofwel GWei). Het event wordt vervolgens ook op het scherm getoond:

```
{ logIndex: 0,
  transactionIndex: 0,
  transactionHash:
'0x6e0512ab90505a4aedfe77a724757c7c870a45d472602c2fbf556642b6b5c1c6',
  blockHash:
'0x83bf7cbd0cbf90eb57c04b008340dae7d965c5930526eff9141c29264b94c764',
  blockNumber: 5,
  address: '0x33d5fd32a27189268891640a1bce91a132ebc947',
  type: 'mined',
  event: 'strippenGekochtEvent',
  args:
   { _klantNaam: 'Laurens BV',
     _aantalStrippen: BigNumber { s: 1, e: 0, c: [Array] } } }
```

Het ingebouwde account accounts[1] heeft standaard 0,25 Ether. Na deze transactie van 1/6 Ether + Gas, blijft er dus 0,083 Ether over:

```
truffle(development)>
web3.fromWei(web3.eth.getBalance(web3.eth.accounts[1]))
BigNumber { s: 1, e: 1, c: [ 99, 83126260000000 ] }
```

Daarnaast heeft het smartcontract de 1/6 Ether gekregen:

```
truffle(development)>
web3.fromWei(web3.eth.getBalance(Strippenkaart.address))
BigNumber { s: 1, e: -1, c: [ 16000000000000 ] }
```

We moeten niet vergeten het account van de consultant en Whitehorses te zetten, in dit geval respectievelijk accounts[5] en accounts[6]:

**Code:**

```
  // zet de consultant en whitehorses voor uitbetaling
    function zetConsultant (address _consultant, address _whitehorses)
public onlyOwner {
    consultant = _consultant;
    whitehorses = _whitehorses;
  }
```

**Executie:**

```
truffle(development)> app.zetConsultant(web3.eth.accounts[5], web3.eth.
accounts[6]);
{ tx: '0x15c0d02911c895449af610d352ebd5fe0757b2731895f3571cf40a62caf16ab0',
  receipt:
   { transactionHash:
'0x15c0d02911c895449af610d352ebd5fe0757b2731895f3571cf40a62caf16ab0',
     transactionIndex: 0,
     blockHash:
'0x1356b78a7ea02acbbbd64aadf5717b1e90892fcd8554867bda4b4ba48a71ca15',
     blockNumber: 6,
     gasUsed: 65100,
     cumulativeGasUsed: 65100,
     contractAddress: null,
     logs: [],
     status: 1 },
  logs: [] }
```

Als we vervolgens een uurtje van de strippenkaart gaan afstempelen, dan zullen de eerder genoemde accounts naar 20/80 rato worden betaald (minus Gas), en zal het worden afgeboekt van het smart contract:

```
  function afstempelenStrippen (uint _aantalStrippen) public onlyOwner {

    whitehorses.transfer(_aantalStrippen*UURLOON/5);
    // Beetje voor mij
    consultant.transfer(4*(_aantalStrippen*UURLOON)/5);

    strippenAfgestempeldEvent(_aantalStrippen);
  }
```

## Executie:

```
truffle(development)> app.afstempelenStrippen(1);
{ tx: '0xf79634daae95d5439dc0fd095af15a708dbe5122a57d96a9efff00ac97ac65ee',
  receipt:
   { transactionHash:
'0xf79634daae95d5439dc0fd095af15a708dbe5122a57d96a9efff00ac97ac65ee',
     transactionIndex: 0,
     blockHash:
'0x62f69839c4f4451fddf63e18f1be6c178ad935fe99722482645c3bbe012694a',
     blockNumber: 7,
     gasUsed: 38546,
     cumulativeGasUsed: 38546,
     contractAddress: null,
     logs: [ [Object] ],
     status: 1 },
  logs:
   [ { logIndex: 0,
       transactionIndex: 0,
       transactionHash:
'0xf79634daae95d5439dc0fd095af15a708dbe5122a57d96a9efff00ac97ac65ee',
       blockHash:
'0x62f69839c4f4451fddf63e18f1be6c178ad935fe99722482645c3bbe012694a',
       blockNumber: 7,
       address: '0x33d5fd32a27189268891640a1bce91a132ebc947',
       type: 'mined',
       event: 'strippenAfgestempeldEvent',
       args: [Object] } ] }


truffle(development)> web3.fromWei(web3.eth.getBalance(web3.eth.
accounts[5]))
BigNumber { s: 1, e: 2, c: [ 100, 12800000000000 ] }
truffle(development)> web3.fromWei(web3.eth.getBalance(web3.eth.
accounts[6]))
BigNumber { s: 1, e: 2, c: [ 100, 3200000000000 ] }
```

We hebben in vogelvlucht een eenvoudige smart contract gezien. We hebben interactie gehad met het smart contract via de truffle console. Dit is puur ter illustratie en voor tests. Natuurlijk zijn er diverse manieren om een smart contract te gebruiken, waaronder JavaScript, Java etc. Dat is echter buiten de scope gehouden van dit Whitebook.

Het is leuk om te zien dat zelfs bij deze eenvoudige operaties de blockchain al 7 blocks bevat (te zien in de uitvoer van truffle na een transactie). Deze chain is de audit trail van het smart contract en is gevalideerd door het netwerk.

## Conclusie

Een block chain is een knap stuk techniek waar in een gedistribueerde omgeving transacties met cryptovaluta worden gedaan. Alle transacties zijn openbaar en geverifieerd: alle nodes in het netwerk zijn het eens over de waarheid van de block chain en fraude is praktisch onmogelijk.

Maar een block chain is meer dan alleen een netwerk voor virtuele munten. Vele cryptovaluta hebben toegevoegde functionaliteit als smart contracts. Zo ook Ethereum. Naast transacties biedt het Ethereum netwerk de mogelijkheid om je eigen software te schrijven in de vorm van smart contracts. Deze smart contracts worden geschreven in een Turing Complete taal, wat betekent dat het een complete programmeertaal is waar je dus alles in zou kunnen programmeren. De transacties die deze smart contracts genereren zijn dan volledig in te zien en gevalideerd en dus goed te auditen.

De blockchain technologie staat nog in de kinderschoenen en het zal de komende tijd alleen maar gaan groeien. Het wordt een spannende tijd in de cryptovaluta wereld.

# Code

```
pragma solidity ^0.4.4;

contract Strippenkaart {

  string klantNaam;
  address klant;
  uint aantalStrippen = 0;

  address owner;

  // uurloon: 1/6 ETH
  uint256 constant UURLOON = 160000000000000000;
  address consultant = 0; // de consultant
  address whitehorses = 0; // De baas

  event strippenGekochtEvent (string _klantNaam, uint _aantalStrippen);
  event strippenAfgestempeldEvent (uint _aantalSrippen);

  // Modifier om bepaalde functies alleen toegankelijk
  // te maken voor de eigenaar van het contract.
  modifier onlyOwner () {
      require (msg.sender == owner);
      _;
  }

  function Strippenkaart() public {
    // constructor
    owner = msg.sender;
  }

  // zet de consultant en whitehorses voor uitbetaling
  function zetConsultant (address _consultant, address _whitehorses) public
onlyOwner {
    consultant = _consultant;
    whitehorses = _whitehorses;
  }

  // koop een nieuwe strippenkaart of waardeen het op
  function koopStrippen (string _klantNaam, uint _aantalStrippen) public
payable {
```

```
    // kan alleen bij nieuwe strippenkaart, of opwaarderen
    require(msg.sender == klant || klant == 0);
    require(msg.value >= _aantalStrippen * UURLOON);

    // nieuwe Strippenkaart
    if (klant == 0) {
      klant = msg.sender;
    }

    aantalStrippen += _aantalStrippen;
    klantNaam = _klantNaam;
    strippenGekochtEvent(klantNaam, _aantalStrippen);

  }

  // afstempelenStrippen
  function afstempelenStrippen (uint _aantalStrippen) public onlyOwner {

    whitehorses.transfer(_aantalStrippen*UURLOON/5);
    // Beetje voor mij
    consultant.transfer(4*(_aantalStrippen*UURLOON)/5);

    strippenAfgestempeldEvent(_aantalStrippen);
  }

  function overgeblevenStrippen() public constant returns (uint strippen) {
    // Alleen voor de klant of de eigenaar
    require(msg.sender == owner || msg.sender == klant);

    return aantalStrippen;
  }

}
```

# Links:

https://media.consensys.net/time-sure-does-fly-ed4518792679 part I
https://medium.com/@ConsenSys/ethereum-bitcoin-plus-everything-a506dc780106 part II
https://blog.zeppelin.solutions/the-hitchhikers-guide-to-smart-contracts-in-ethereum-848f08001f05
https://github.com/ethereum/wiki/wiki/White-Paper#ethereum
https://solidity.readthedocs.io/en/latest/index.html
https://ethgasstation.info/index.php
https://solidity.readthedocs.io/en/latest/structure-of-a-contract.html
https://en.wikipedia.org/wiki/Blockchain
https://nl.wikipedia.org/wiki/Turingvolledigheid
Bitcoin: A Peer-to-Peer Electronic Cash System, Satoshi Nakamoto