Een block bestaat uit 3 delen om te kunnen werken. de body, de header en de output hash.

Het eerst wat je moet weten is de body. De reden hiervoor is dat de **body gebruikt** wordt **in** de **header**. Niet de hele body maar een **deprecated deel** hiervan. Dit deprecated deel **heet** de **merkle root**.

## Wat is de body?

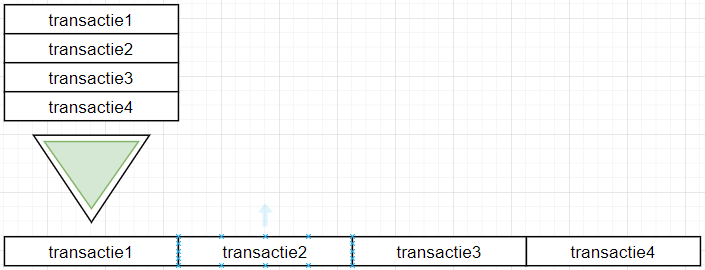
De **body** is de **data** van een **block**. Voorbeeld is van cryptocurrency de transacties. Deze data is een lijst van veranderingen.

### Wat is de Merkle Root?

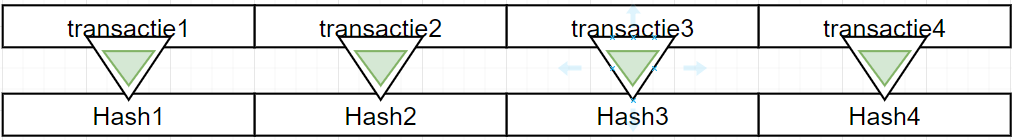
De merkle **root** is een **hash** van **alle hashes** van de **datalijst** in de body?

### Hoe wordt de Merkle Root Gegenereerd?

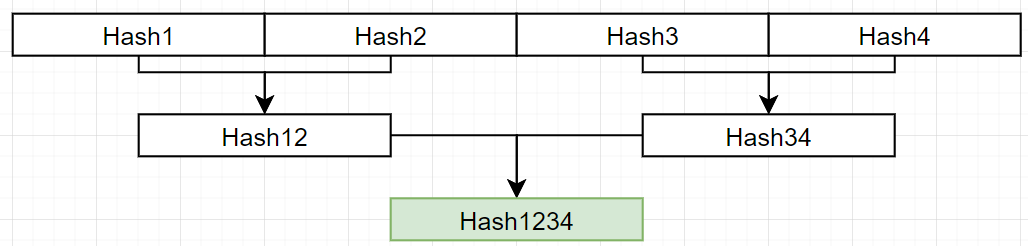
**stap 1**: de lijst van data in de body en zet deze langs elkaar.



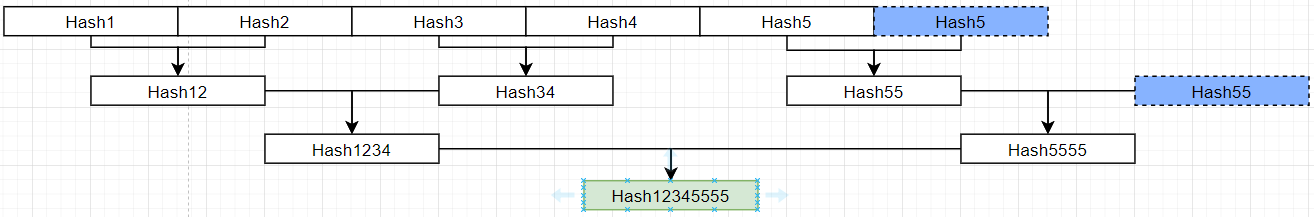
**stap 2**: hash alle transacties.



**stap 3**: combineer hash1 en hash2 naar een node



**Uitzondering stap 3:** van oneven aantal in de lijst:



de Groene hash is de Merkle root.

#### Verdere toelichting stap 3 uizondering Merkle root

Het is niet alleen hash1 plust hash2 of hash1 \* hash2. alhoewel dit wel gebeurt wordt de uitkomst hiervan nog een keer gehasht en dat is de nodehash12.

De uitzondering van oneven aantal is dat de eenzame hash zichzelf kopieert en daarmee een nodehash maakt.

Dus dan krijg je in het voorbeeld hierboven hash5 + hash5 en dit wordt nog eens gehasht. En dit maakt nodehash55

### Waarom wordt de merkle root gemaakt?

De merkle tree techniek van de data omzetten naar een hash wordt gedaan omdat: dan het verifiëren van een block een stuk sneller gaat. Het biedt schaalbaarheid voor grote blockchains.

### Conclusie Merkle tree en body

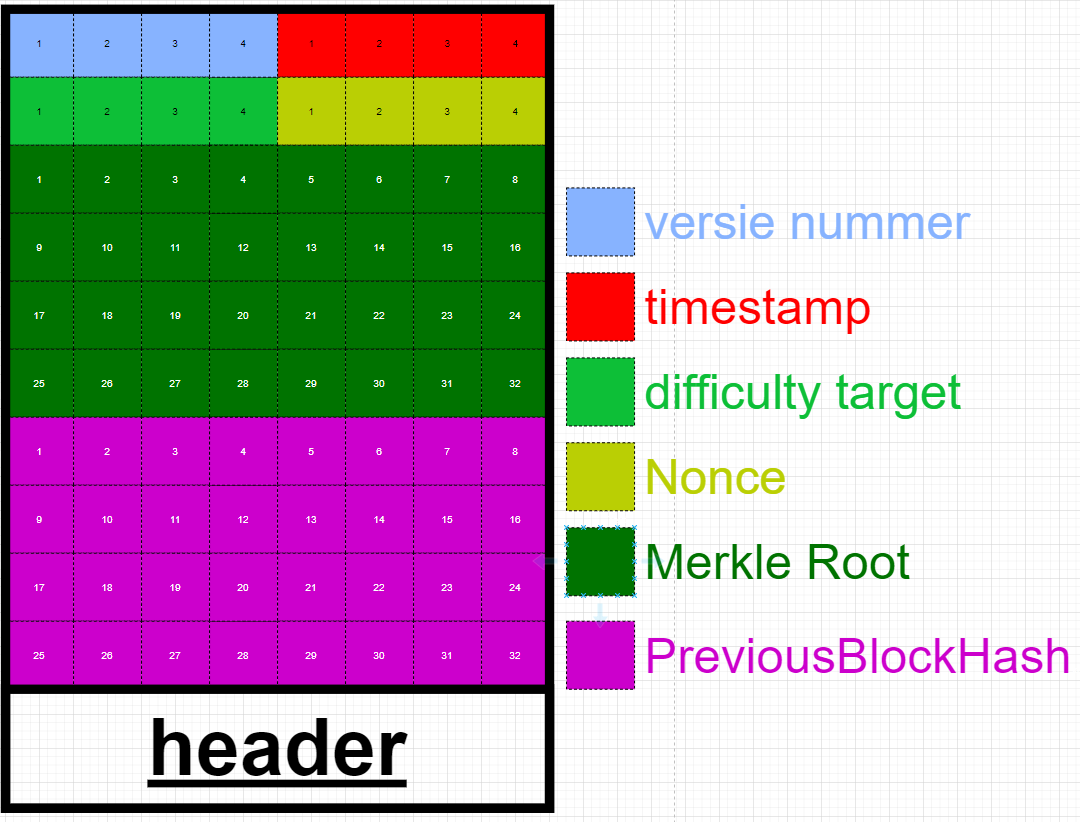
We weten nu dat de body een lijst van data is, deze data wordt gebruikt om de merkle root te maken. Weten hoe de merkle tree werkt. De merkle root wordt gemaakt omdat het verifiëren van de data sneller gaat en makkelijker te schalen is.

## Wat is de header?

De header wordt gebruikt voor het genereren van de output hash. De meest gebruikte variant is die van bitcoin en die bestaat uit:

|  |  |
| --- | --- |
| naam | grootte |
| bitcoin versie nummer | 4 bytes |
| previous block hash | 32 bytes |
| merkle root | 32 bytes |
| timestamp | 4 bytes |
| difficulty target | 4 bytes |
| nonce | 4 bytes |
| **TOTAL** | **80 BYTES** |
| ***Target Hash*** | **32 bytes** |

visuele representatie:



### Wat gebeurt er met de header?

De header wordt elke keer opnieuw gehasht met de nonce totdat het de target hash behaald. Dan wordt de hash die de target hash heeft gehaald een output hash wat gebruikt wordt bij de volgende block zijn header.

#### Waarom heet target hash target hash?

Target hash heeft zijn naam gekregen omdat deze hash moet voldoen aan de difficulty die bepaald wordt. Anders verwoord de target hash is een hash die moet voldoen aan de target die is gesteld door de difficulty.

#### Wat is de nonce?

Number only used once, uniek nummer wat opnieuw gegenereerd blijft worden totdat hiermee de header gegenereerd wordt die de target hash haalt.

#### Wanneer zie je een transactie als voltooid in je “wallet”.

Wat we moeten realiseren is dat elke miner apart blocken zit te maken ( zie document: hoe je als miner geld verdient.)

Dus als 2 miners tegelijkertijd een block maken dan krijg je 2 forks. **[plaatje1]**

Andere miners kijken of dit klopt en zo ja dan bouwen ze hier op verder. **[plaatje2]**

Dit gebeurt vaak dus we weten niet welke chain we verder moeten bouwen en welke transacties nou door zijn gegaan.

De oplossing tegen de eindeloze forks zijn:

Miners kiezen ervoor om te verder bouwen op de langste keten omdat ze meer zekerheid krijgen dat ze hier waarde uit halen. Want alleen de langste keten wordt gezien als geldig. **[plaatje 3]**

Wanneer je zeker weet dat de transactie op de langste keten staat dan heb je een geslaagde transactie.

|  |
| --- |
| plaatje 1 |
| plaatje2 |
| plaatje 3 |