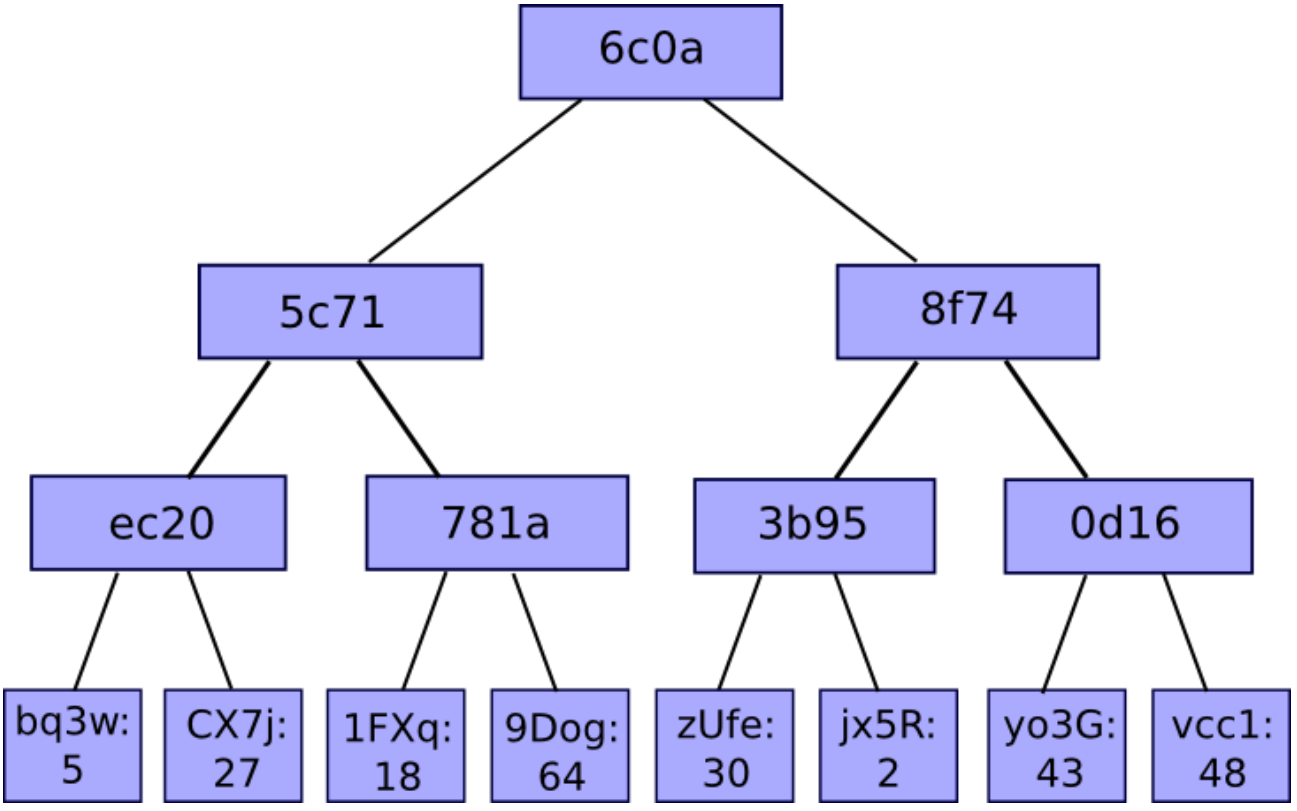
Деревья Меркла. Концепт, области применения, доказательства Меркла.

Определение

- Дерево Меркла (для простоты иногда называемое просто деревом Хэшей) является **бинарным деревом**, каждый **элемент** которого **является хэшем** от **значений дочерних вершин**.
- Сразу же стоит сказать: если мы рассматриваем **классическое дерево** Меркла, то оно действительно **является бинарным** - но есть и деревья интереснее: те же **Merkle Patricia Tries**. О них будет чуть позже.

Чем по сути являются?

- Деревья Меркла на деле являются способом хеширования **большого числа данных**. Мы берем изначальный **набор данных**, **делим** его **на бакеты** (в простейшем случае это пара элементов) и **находим** от каждого **бакета хэш**, а затем повторяем эту операцию **снова и снова**.
- В конце концов мы **дойдем до корня** (***Merkle Root***), вся прелесть которого заключается в том, что он целиком и полностью **характеризует ВЕСЬ** этот огромный **набор данных**. Другими словами, если хоть одно значение в дереве изменится - мы получим новый корень.



- Здесь сразу же напрашивается хороший вопрос: а **зачем нам так париться** и строить это дерево. Не легче ли просто взять, **поместить весь объем** данных **в один бакет** и **найти** от него **хэш**?

- Действительно, если нам нужно лишь отследить **ФАКТ изменения значений**, то нам вполне **сгодится** и **наивный подход**. Но Дерево Меркла может предоставить нам такой крутой инструмент как ***Merkle Proof***.

Merkle Proof и как пруфануть за ***O(log2(N))*** **по памяти**?

- **Так в чем преимущество** этой штуки? Она позволяет нам **доказать нахождение элемента** в массиве исходных данных НЕ за ***O(N)***, а за ***O(log2(N))*** **-** здесь я говорю о том, **какое именно количество** элементов для пруфа **надо предоставить**.

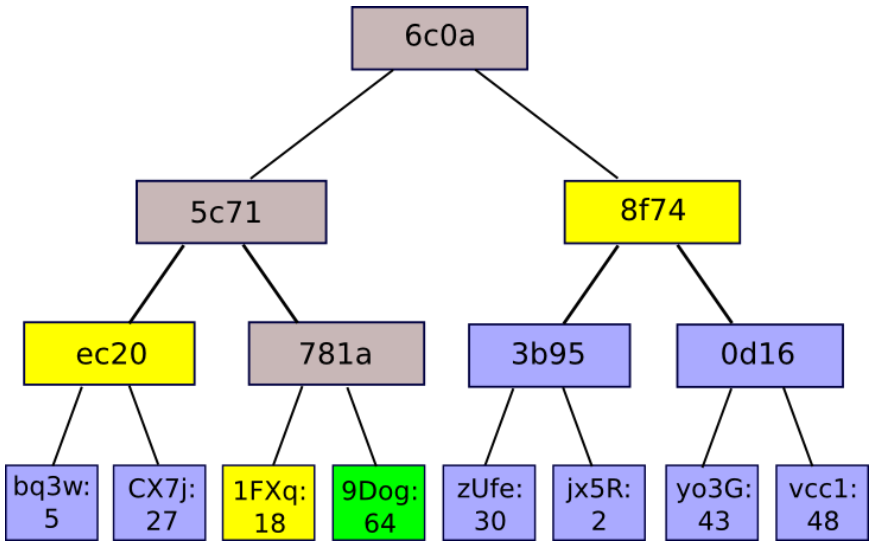
- Очевидно, что чем **больше размер массива** - тем **больше мы выигрываем** от Меркла.

- Сам по себе ***Merkle Proof*** включает в себя **3 важные составляющие**:

    - ***Merkle Root*** - **корень дерева Меркла**, построенного на массиве данных.
      К слову, если у нас есть какой-нибудь сервер с минимальным количеством памяти, то ему **для проверки достаточно** будет хранить у себя **только это значение** и всё. Остальное принесет клиент.

    - ***Value*** **-** то самое **значение**, наличие которого проверяется. На практике мы можем передавать серверу даже **не само** это **значение**, а **сразу же хэш** от него.

    - ***Branch Values*** **-** это **минимальный набор хэшей**, необходимый нам для проверки. То есть в ветку входят лишь необходимые хэши, идущие вверх по пути от нашего значения до корня. Ниже **желтым** обозначены эти **необходимые хэши**:



Где можно использовать?

- В традиционных базах данных, содержимое которых можно сохранить в форме дерева Меркла. **Корень** при этом **публично доступен** и **подписан** (никто его не подменит).

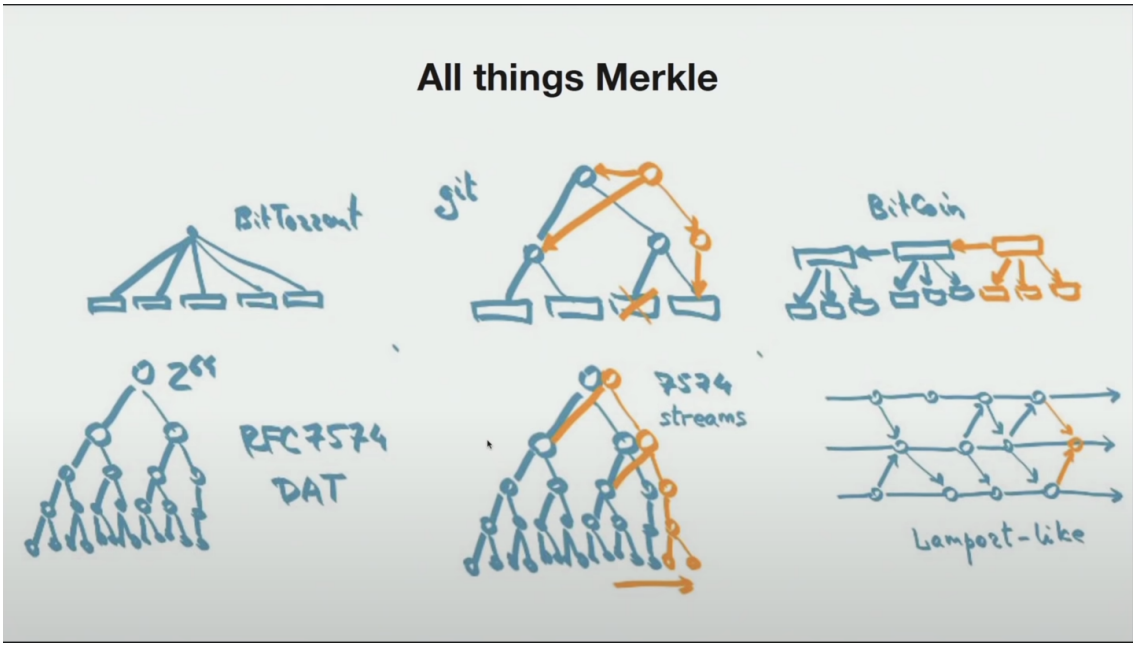
  Если юзер хочет **вытащить элемен**т с каким-то **индексом** из этой БД, а затем проверить, что его не надули - он может попросить дополнительно ещё и ***Merkle Proof***. А затем **проверить**, что он получил **именно тот элемент**, к которому обращался.

- Как уже говорил, можно использовать это **для доказательства факта** стороне, у которой **ограниченное число ресурсов**. Если рассматривать это применительно к крипте, то классическим примером является ***SPV*** **(*Simple Payment Verification*)** в BTC.

  Там мы строим дерево Меркла **на основе транзакций**, то есть они и **являются исходным массивом**. А в хедеры блоков добавляем получившийся ***Merkle Root***.

  SPV позволяет **проверить платежи нодам**, которые **не скачивают блоки полностью**, а **загружают лишь их хедеры**. То есть если у нас есть транзакция и ***Merkle Proof*** - мы можем проверить, действительно ли она есть в одном из блоков. Но мы ничего **не можем узнать** о **состоянии (**о том же количестве монеток, например).

- Помимо прочего, деревья Меркла **также используются** в ***IPFS -*** а ещё внутри ***Git u BitTorrent.*** Есть также и другие применения, но ограничимся этими.



- Ещё одно важнейшее применение деревья Меркла находят в Эфире. К слову, там их целых 3 штуки: для **транзакций**, для **квитанции** и для **состояния**. И там они имеют более сложную структуру и называются **Merkle Patricia Tries**.

Merkle Patricia Tries and **Ethereum**

- Для начала обсудим, **зачем вообще** в Эфире **понадобилось** какое-то **более сложное** дерево Меркла.

  В BTC нам вполне достаточно классического дерева Меркла, поскольку **список транзакций** в блоке никогда уже **не будет изменяться**. Однако в Эфире у нас есть **общее состояние**, которое **постоянно меняется** (создаются новые аккаунты, отрабатывают контракты, меняются балансы).

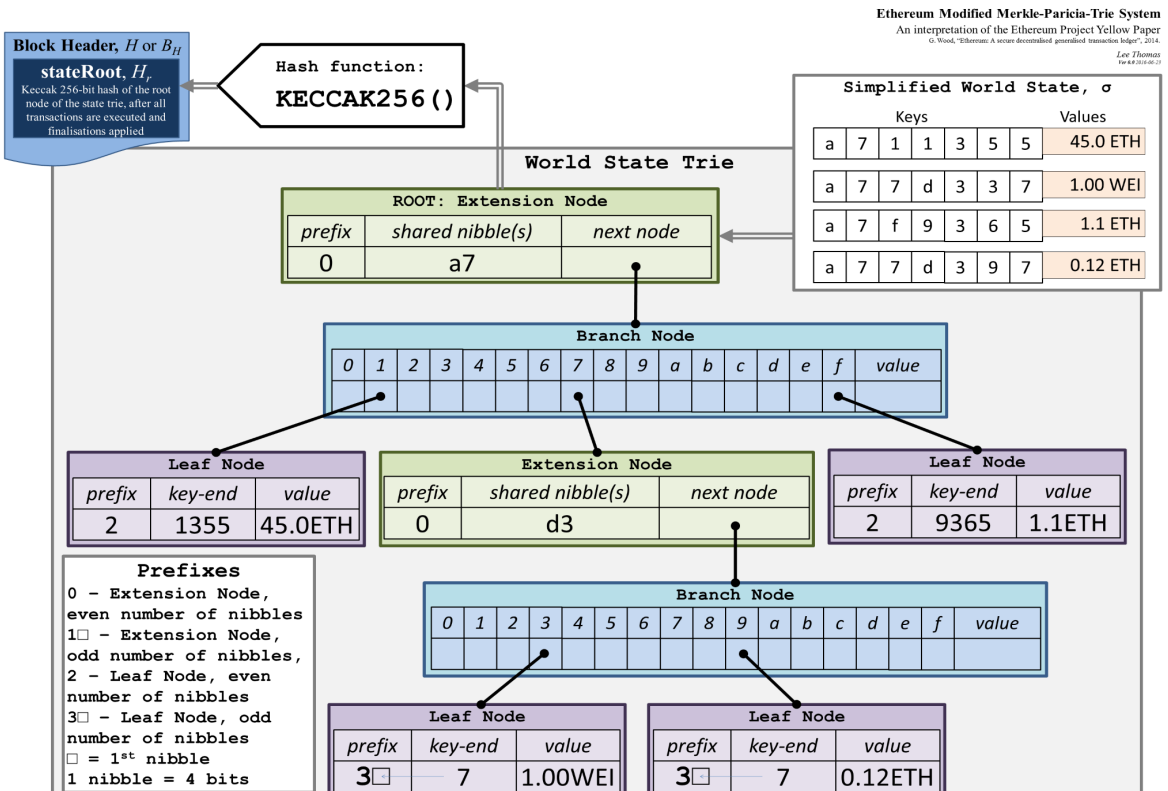
  Значит нам нужна структура данных, поддерживающая **быструю вставку, поиск и удаление**.

- **Merkle Patricia** (**P**ractical **A**lgorithm **T**o **R**etrieve **I**nformation **C**oded **I**n **A**lphanumeric) **Trie** как раз и является такой структурой данных. Идейно она опирается на **префиксное дерево** (***radix tree = trie***), но с **некоторым числом улучшений**.

- Если **описывать эту СД** совсем поверхностно, то она, во-первых, предлагает нам использовать сразу **несколько видов вершин**:

  - **Пустые** (Empty / Blank nodes) - в них **буквально ничего** не хранится;
  - **Листовые** (Leaf nodes) - это просто вершина, хранящая в себе одну **KV-пару и всё**;
  - **Вершины-ветки** (Branch nodes) - вот это уже **17-элементный список**.
  - Вершины-расширения (Extention nodes) - тоже хранят в себе **одну KV-пару**, но Value там является **хэшем** для какой-то **другой вершины**. У меня возникает аналогия с **Symbol Link** в *nix

- Кроме того, **путь до каждой вершины - это хэш**.



- Всё это позволяет даже с легких Эфировских клиентов получать **данные о балансах**, **проверять факт существования** аккаунта и так далее.

Список источников:

1) Ethereum Developers Guide on Github :
    a) Random Crypto-Currency Concept #1 - Merkle Trees.
    b) Merkling in Ethereum
    c) Ever Wonder How Merkle Trees Work?

2) BTC Whitepaper (8. Simplified Payment Verification)

3) MIPT Technopark Blockchain Course: Lecture about symm. crypto.

4) Merkle Trees and Patricia Tries - Blockchain for Developers [Lab 7]
5) Merkle Trees & Patricia Tries for Blockchain - Explained

6) Виктор Грищенко — Децентрализованный веб
7) Understanding the ethereum trie