Credereum – Postgres с поддержкой блокчейн

Alexander Korotkov

Postgres Professional

2018

Зачем?

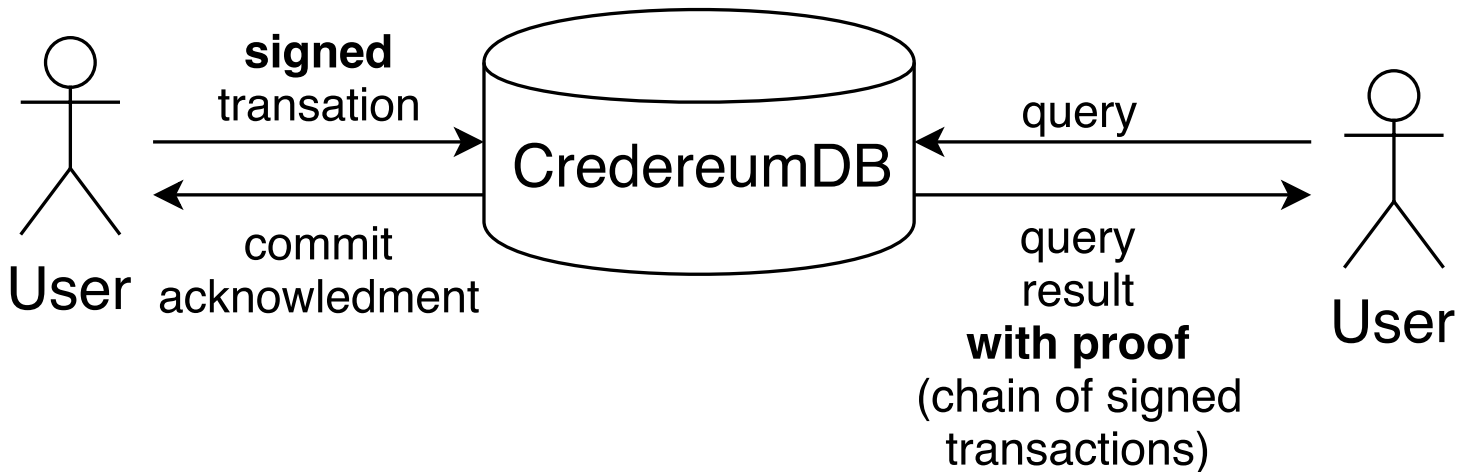
- Вокруг blockchain образовался большой хайп, есть как сторонники, так и противники.
- Можно ли извлечь из данной технологии что-то полезное для СУБД?

Что такое Credereum?

Проект Credereum состоит в привнесении элементов blockchain в реляционную СУБД (конкретно в PostgreSQL), а именно:

- Пользователь **подписывает** каждую транзакцию, которую он выполняет;
- СУБД **доказывает** пользователю корректность результатов запроса;
- При этом система остаётся централизованной, **владелец БД** остаётся;
- **Приватность** данных тоже остаётся, каждый пользователь видит только свою часть данных.

СУБД с цифровой подписью



Проблема всё равно остаётся?

Из исходной статьи про bitcoin [1].

A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution.

ОК

Digital signatures provide part of the solution,

Здесь всё понятно!

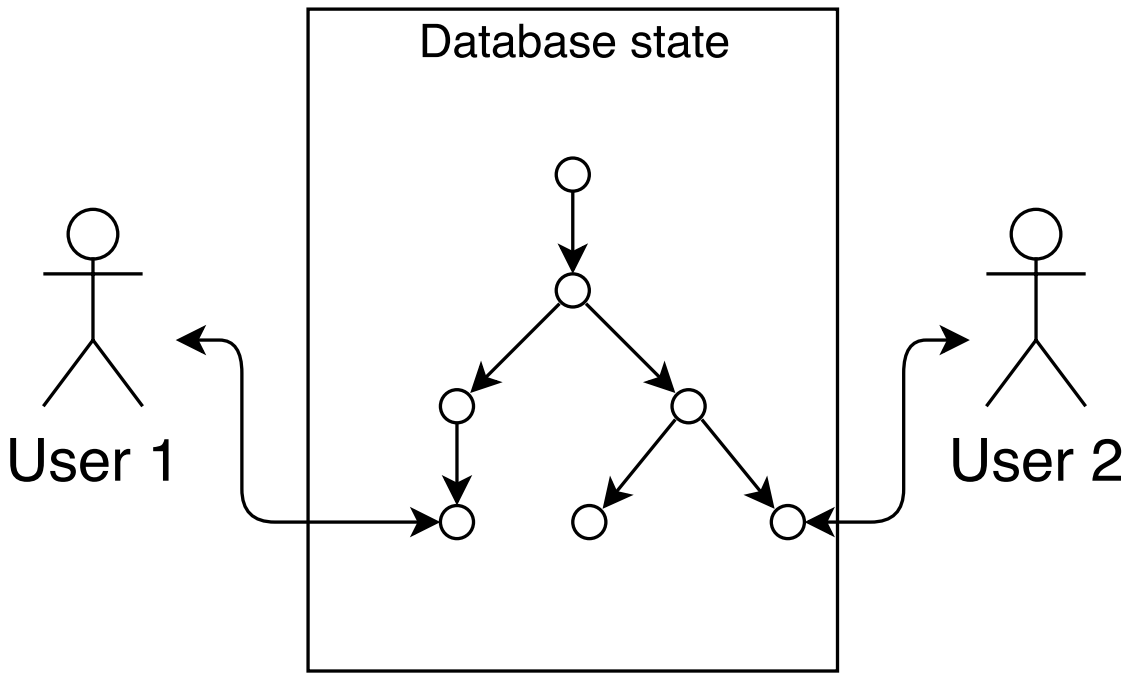
but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network.

Хмм...

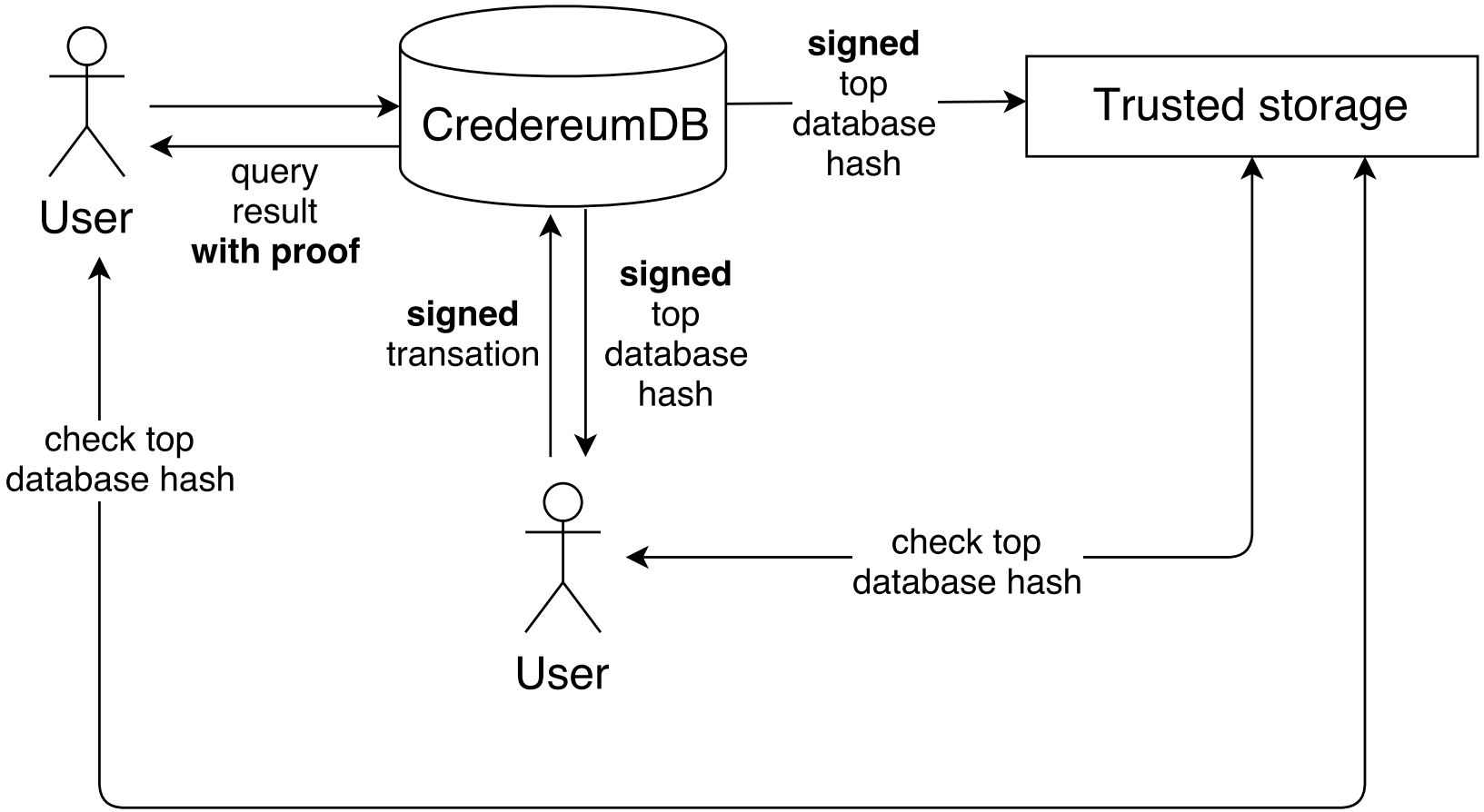
[1] `https://bitcoin.org/bitcoin.pdf`

Проблема “double spending”

Владелец базы может вести несколько версий одной БД, показывая разные версии разным пользователям.

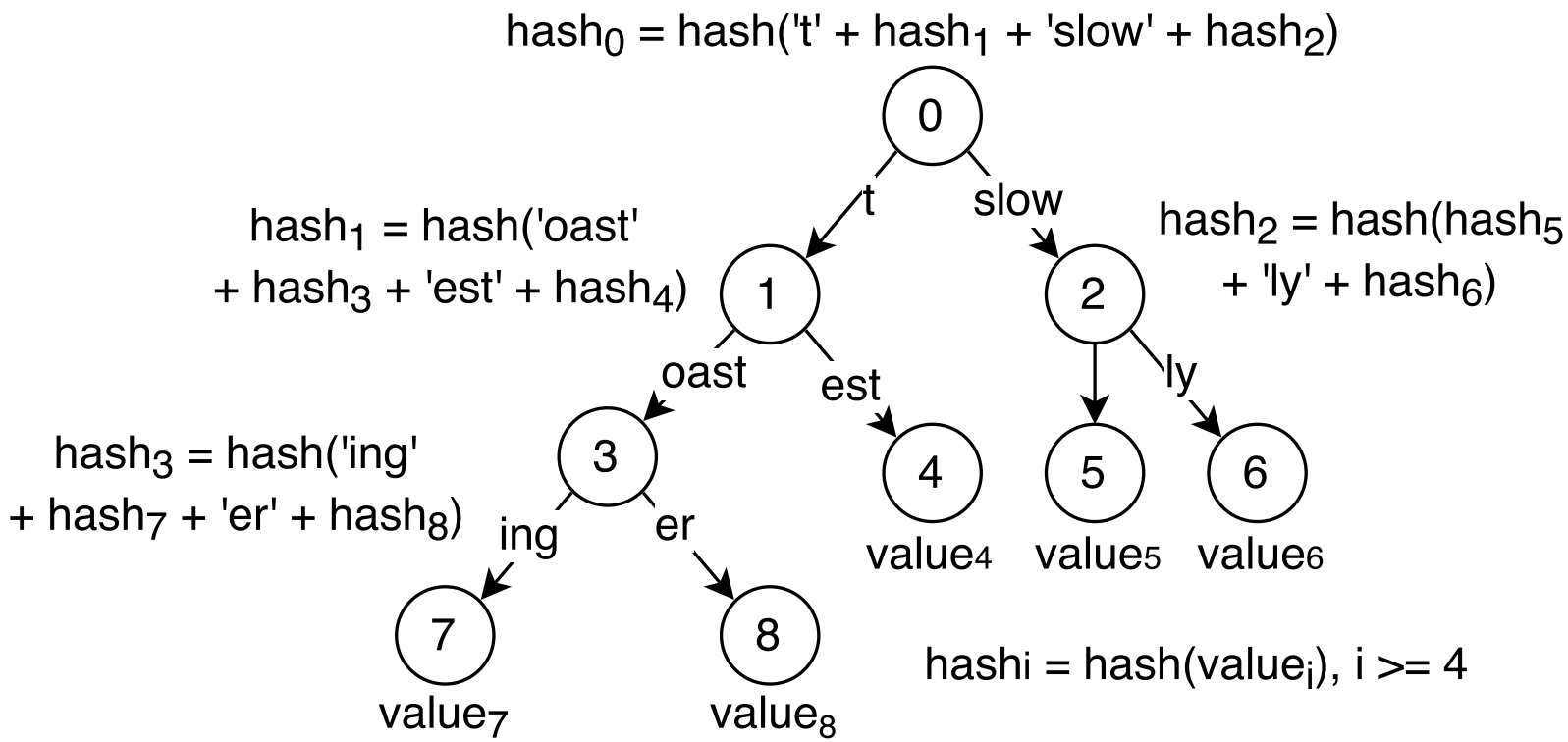


Решением проблемы “double spending” с помощью trusted storage



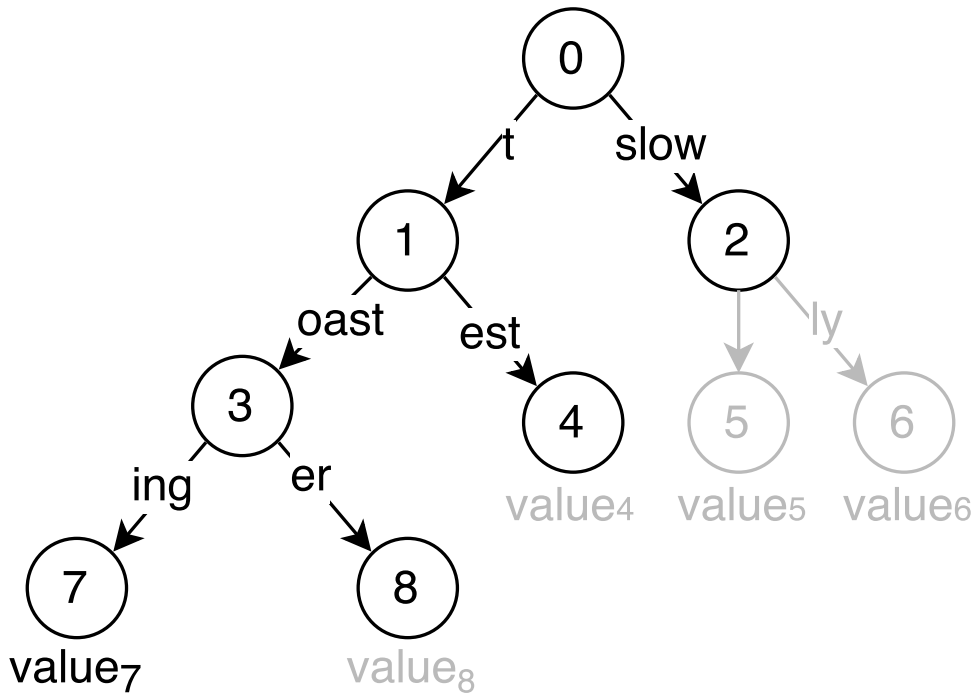
Нужно больше деталей?

Merklix = Merkle tree + Radix tree

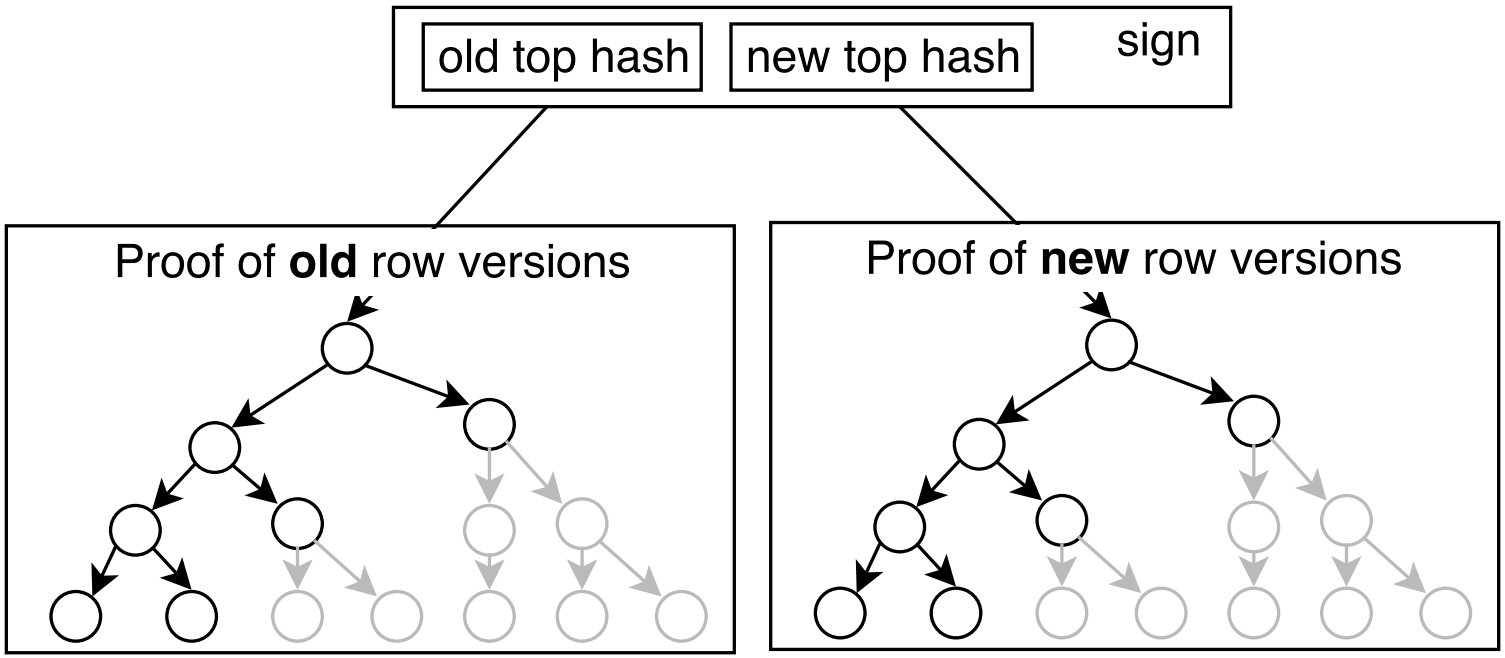


Merkle proof in merklix tree

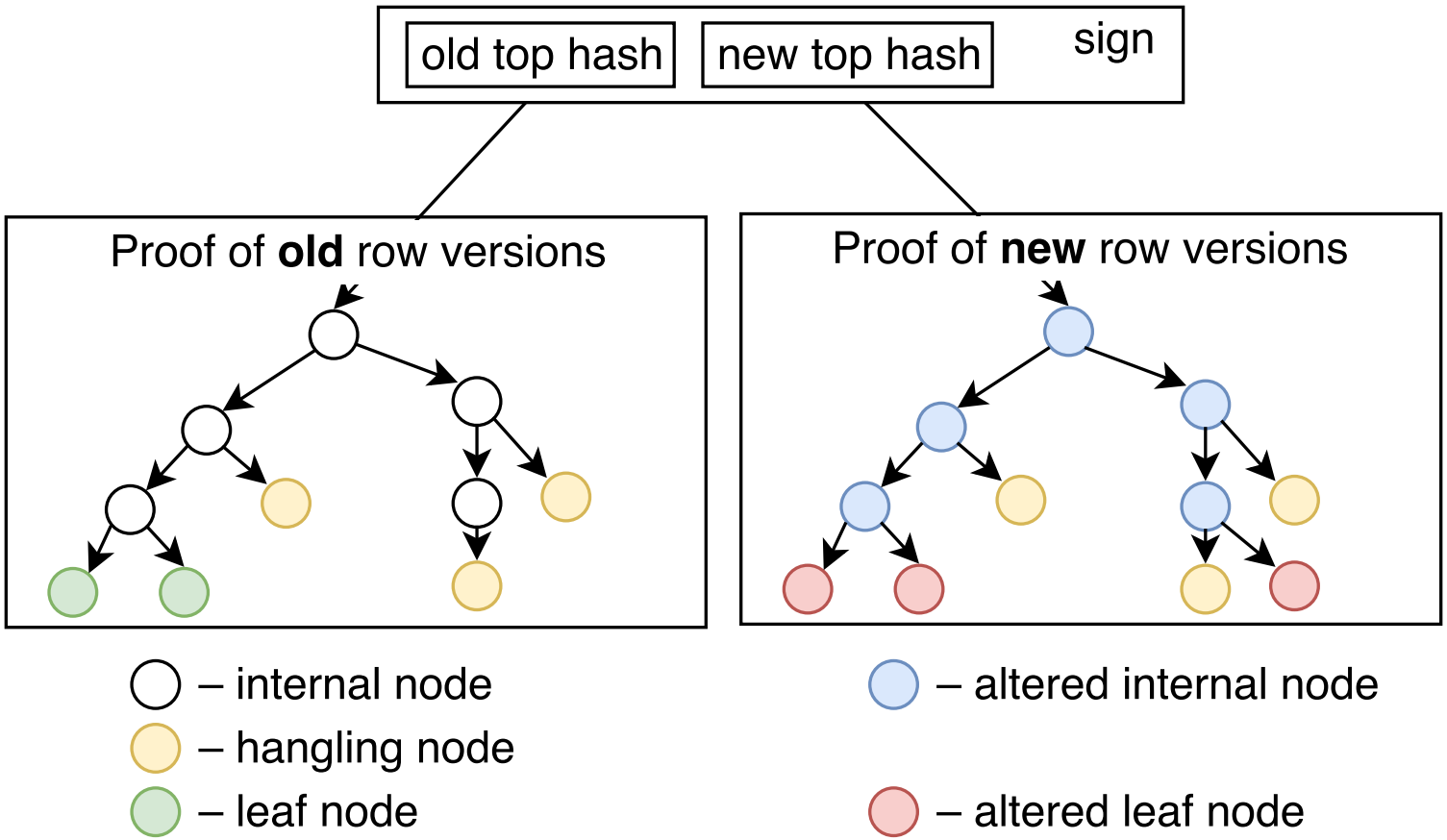
Merkle proof для “toasting”.



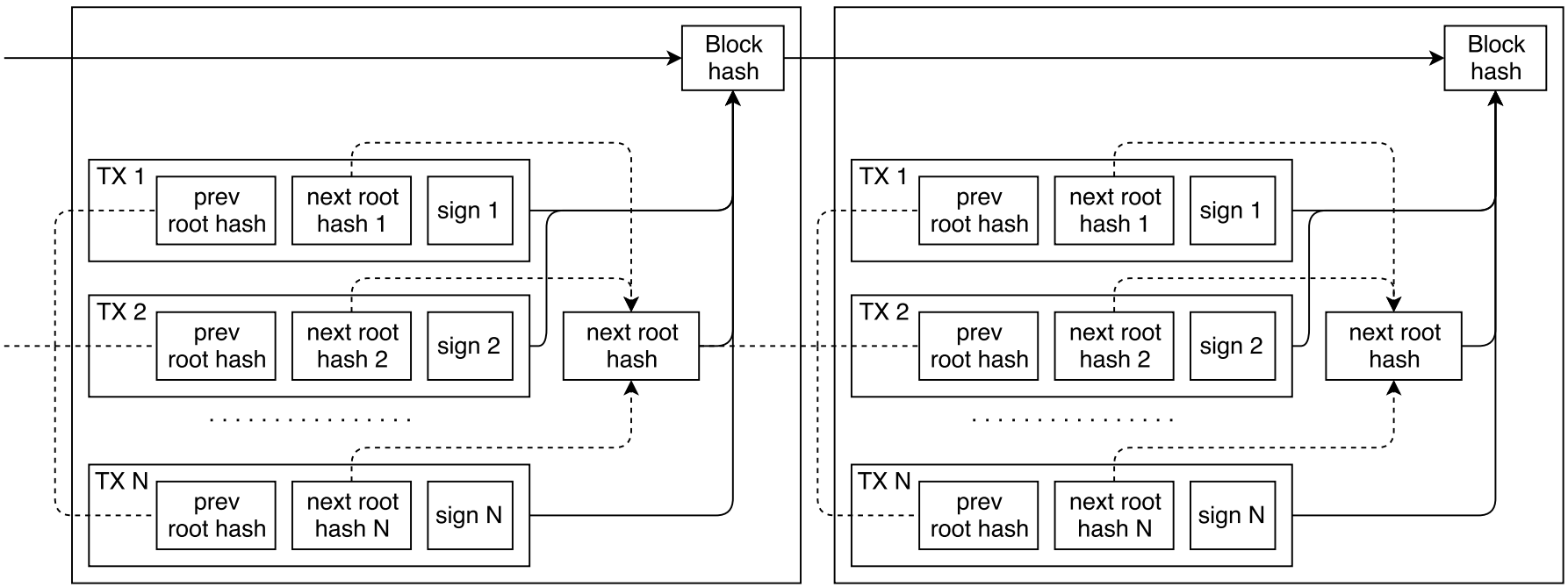
Что подписывает пользователь?



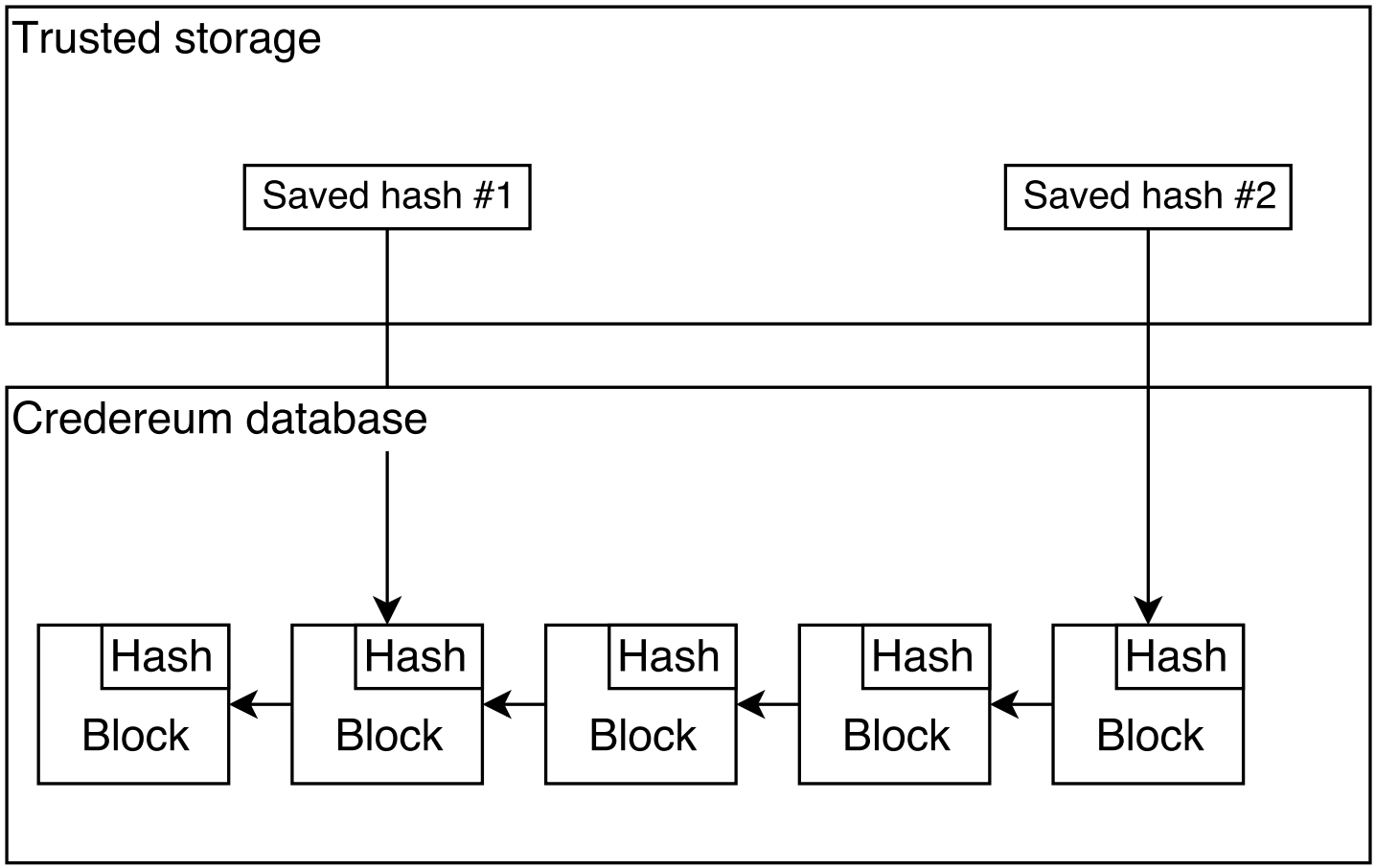
Как пользователь проверяет транзакцию?



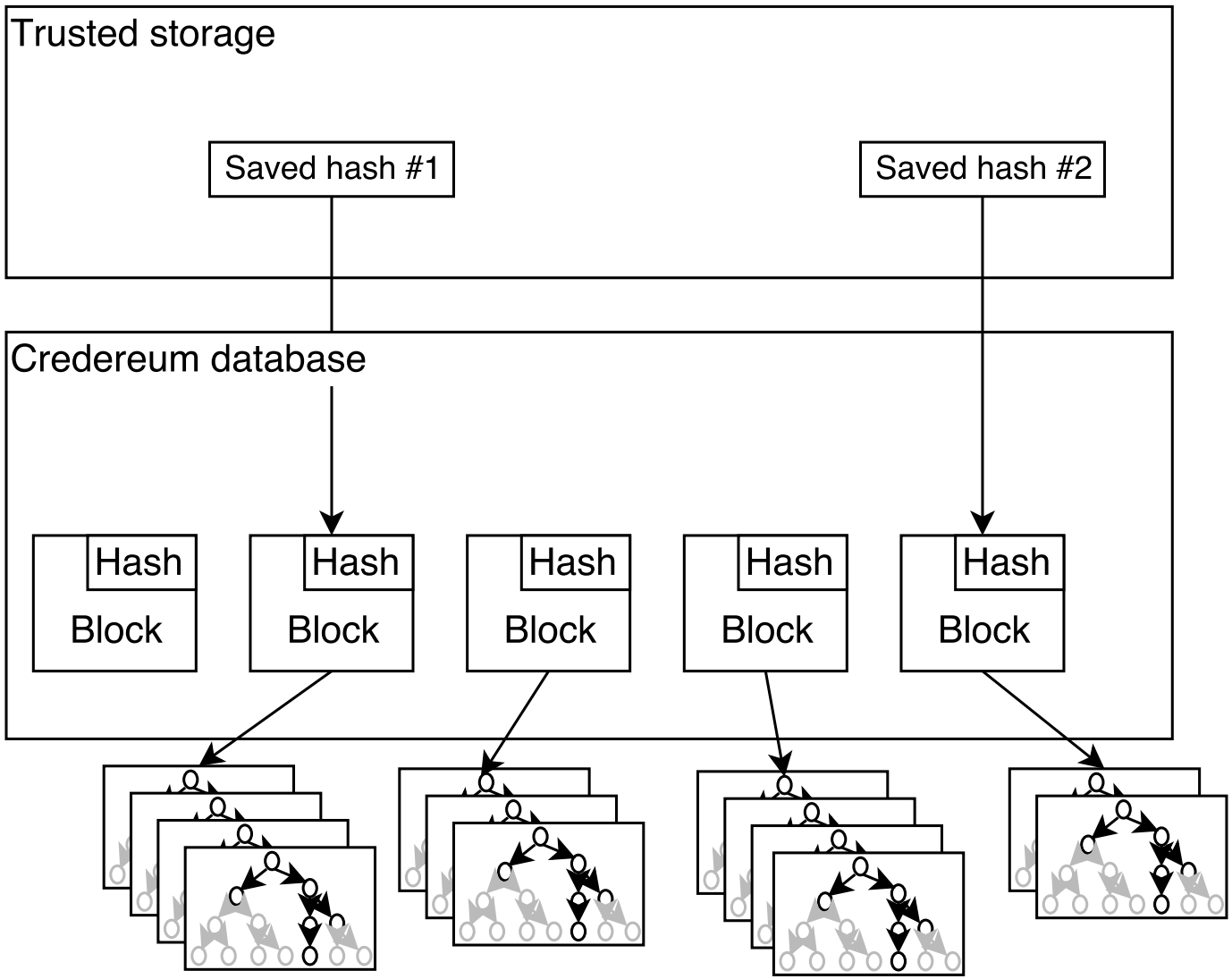
Структура blockchain



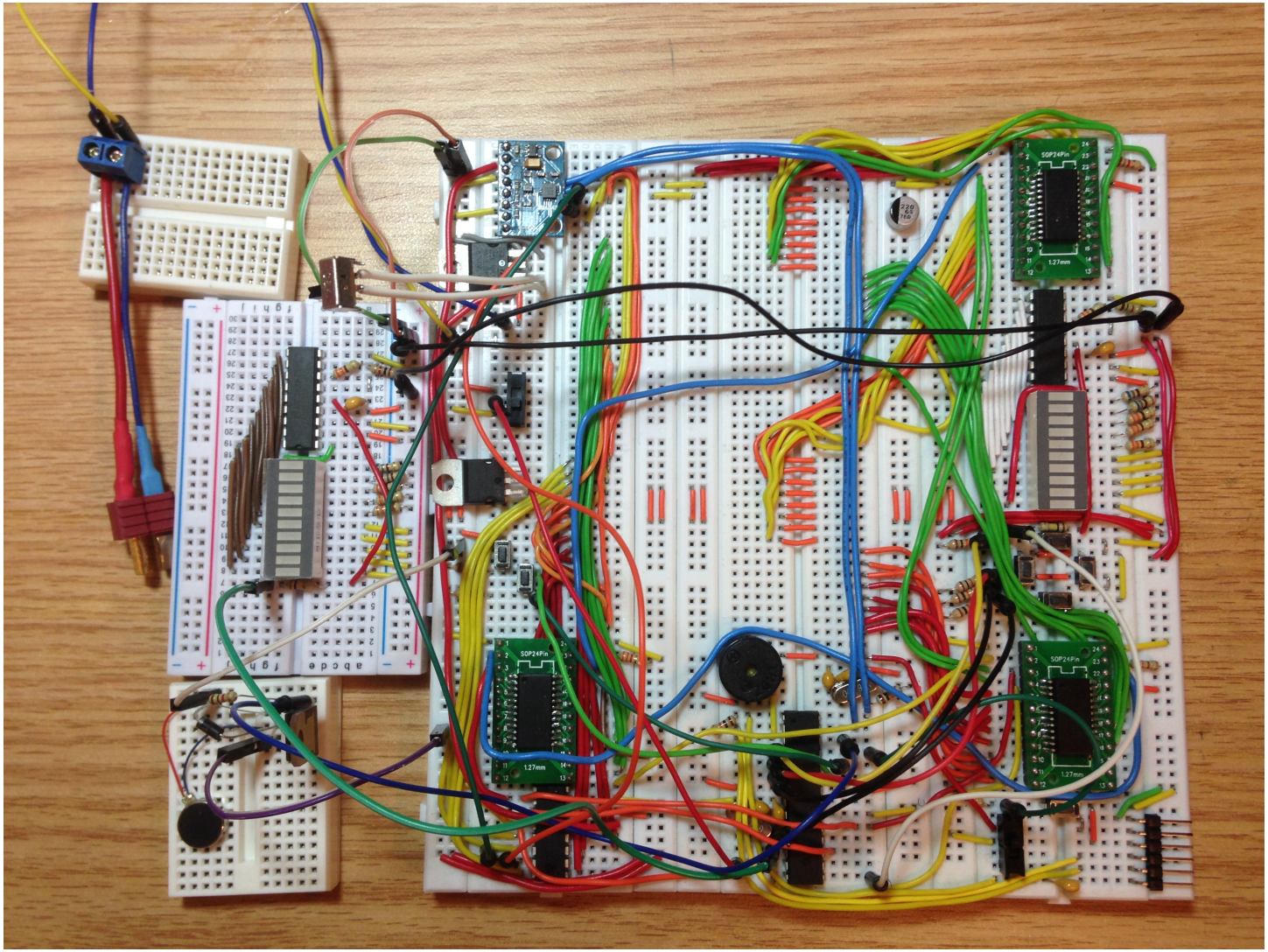
Хэши фиксируются в trusted storage



Доказательство истории

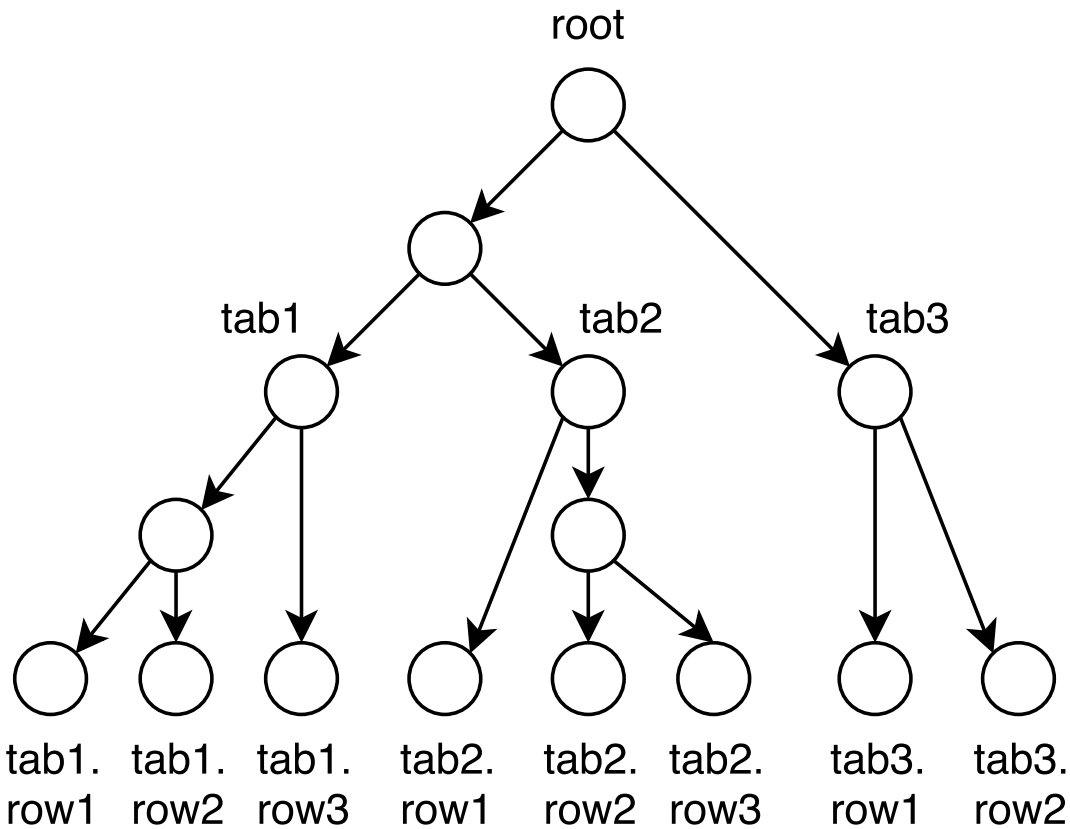


Прототип



Прототип: представление ключа

key = relation_name + row_id(64-bit)



Прототип: хранение дерева

```
CREATE TABLE credereum_merklix
(
    key             varbit NOT NULL,
    generation      bigint NOT NULL,
    transaction_id  bigint,
    children        varbit[],
    leaf            bool NOT NULL,
    hash            bytea,
    value           json,
    PRIMARY KEY (key, generation, transaction_id),
    CHECK (leaf OR value IS NULL) -- Non-leafs doesn't have values
);
```

Прототип: установка (скоро будет доступен)

```
$ git clone git@github.com:postgrespro/pg_credereum.git
$ cd pg_credreum
$ make USE_PGXS=1
$ sudo make USE_PGXS=1 install
$ make USE_PGXS=1 installcheck
$ psql dbname
```

```
CREATE EXTENSION pg_credereum;
CREATE TRIGGER t_after
AFTER INSERT OR UPDATE OR DELETE ON t
FOR EACH ROW EXECUTE PROCEDURE credereum_acc_trigger();
```

Прототип: использование

```
BEGIN;
INSERT INTO t (value) VALUES ('abcdef');
SELECT * FROM credereum_get_changeset();
-- Проверяем полученные merkle proof и подписываем их
SELECT credereum_sign_transaction(...pubkey..., ...signature...);
COMMIT;
```

Прототип: пример на python

```
cursor = dbconn.cursor()
cursor.execute("INSERT INTO t (value) VALUES (%s)", (value,))
changeset = get_changeset(dbconn, True)
# Проверяем changeset
assert (not changeset['updates']), 'We didn\'t perform updates'
assert (not changeset['deletes']), 'We didn\'t perform deletes'
assert (len(changeset['inserts']) == 1), 'Should be exactly one insert
assert (changeset['inserts'][0]['value'].keys() == [u'id', u'value']),
assert (changeset['inserts'][0]['value']['value'] == value), 'Values m
# Подписываем транзакцию
sign = crypto.sign(pkey, changeset['hash'], "sha256")
cursor.execute("SELECT credereum_sign_transaction(%s, %s);",
               (pubkey, psycopg2.Binary(sign)))
cursor.close()
dbconn.commit()
```

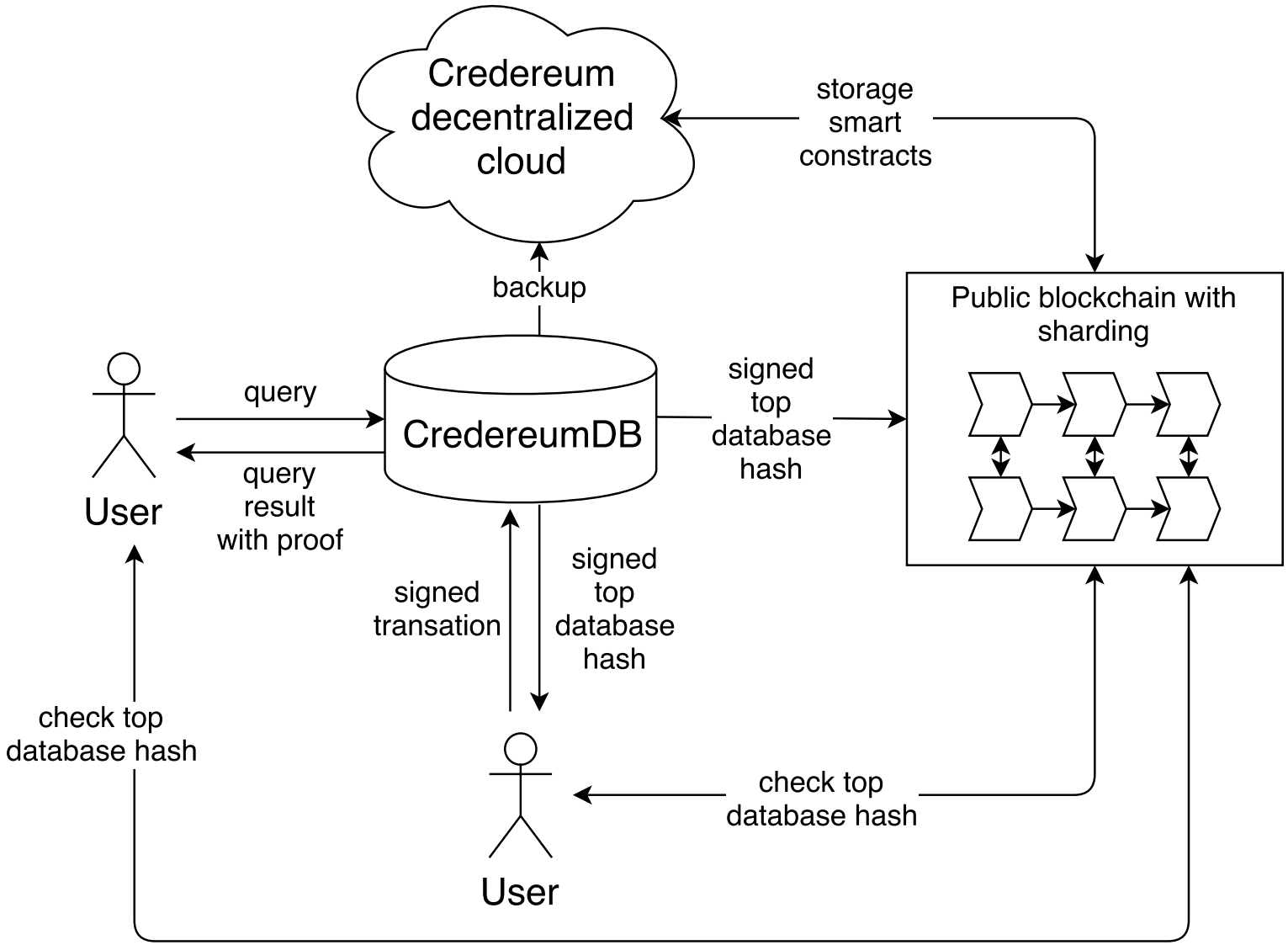
Ограничения предложенного подхода

- Побочные ветки merkle proof’ов открывают coverty channel для данных, которые данный пользователь не должен видеть.
- Цифровая подпись должна осуществляться на стороне конечного пользователя. Таким образом, конечный пользователь непосредственно имеет дело со структурой БД.
- Сложность доказательства истории прямо пропорциональна общему числу транзакций, прошедших за искомый период времени.

Подходы к решению проблемы сложности доказательства истории

- Владелец базы публикует bloom filter изменившихся строк для блоков (hint). Пользователь запрашивает доказательство для всех блоков, где исходная строка входит в bloom filter и для случайных других (выборочная проверка).
- Пользователь поддерживает частичную реплику данных и инкрементально её обновляет.

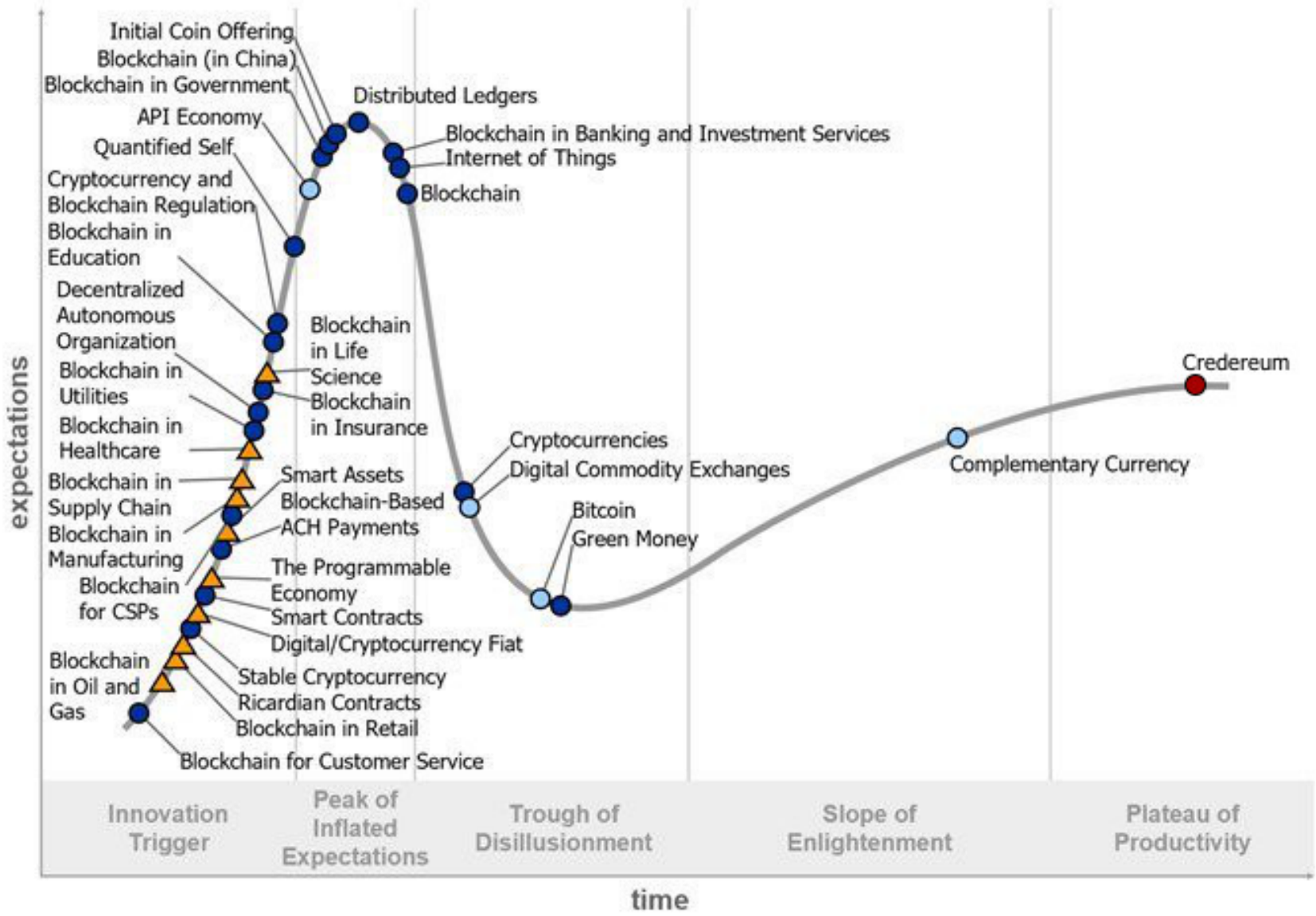
Возможное будущее Credereum



Заключение

- Credereum позволяет владельцу БД доказывать корректность возвращаемых данных, а пользователю БД проверять корректность возвращаемых данных.
- Для того, чтобы убедиться в том, что БД существует в единственном варианте, нужен trusted storage.
- Производительность операций записи будет в разы ниже, чем в обычной БД (overhead поддержания merklix), но на порядки больше, чем в публичных блокчейнах.
- Производительность построения доказательст невелика, но есть подходы к её улучшению.
- Прототип будет опубликован на github.com в ближайшие 2-3 недели (также будет огранизован митап).

Вопросы?


Initial Coin Offering
Blockchain (in China)
Blockchain in Government
API Economy
Quantified Self
Cryptocurrency and Blockchain Regulation
Blockchain in Education
Decentralized Autonomous Organization
Blockchain in Utilities
Blockchain in Healthcare
Blockchain in Supply Chain
Blockchain in Manufacturing
Blockchain for CSPs
Blockchain in Oil and Gas
Blockchain for Customer Service
Blockchain in Retail
Ricardian Contracts
Stable Cryptocurrency
Digital/Cryptocurrency Fiat
Smart Contracts
The Programmable Economy
ACH Payments
Blockchain-Based
Smart Assets
Blockchain in Insurance
Blockchain in Life Science
Distributed Ledgers
Blockchain in Banking and Investment Services
Internet of Things
Blockchain
Cryptocurrencies
Digital Commodity Exchanges
Bitcoin
Green Money
Complementary Currency
Credereum
expectations
Innovation Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
time