Доказательство с нулевым разглашением

Артём Круглов

Дисклеймер

- Теоретический обзор с целью разобраться в теме
- Сделан за короткое время
- Первый доклад на блокчейн-школе
- Применяются материалов других авторов

- Спасибо Алексею за подводку☺

План

- Введение в Zero-knowledge Proof
- Применение Zero-knowledge Proof
- Под капотом zkSNARK
- Использование zkSNARK в Zcash
- Семейства ZKP-систем

Введение
в Zero-knowledge Proof

Что это и почему это круто

Zero-knowledge proofs

- Доказательство с нулевым разглашением - протокол, который:
  - позволяет доказать, что доказываемое утверждение верно,
  - и Доказывающий знает это доказательство,
  - в то же время не предоставляя никакой информации о самом доказательстве данного утверждения

- Zero-knowledge proofs – вероятностные доказательства

Mix
Bytes
()

Zero-knowledge proofs

- Zero-knowledge proofs:
  - криптографическое доказательство некоторого утверждения, без раскрытия дополнительной информации
  - в протоколе участвуют две стороны: доказывающий (*prover*) и проверяющий (*verifier*), они договариваются о проверке некоторого утверждения(*statement*)
  - *prover* предоставляет некоторые данные + *proof*, доказывающий что для этих данных утверждение: true
  - *verifier*: проверяет proof и убеждается, что *prover* его не обманул

Mix
Bytes
()

Zero-knowledge proofs - свойства

- встретите в любом описании zk-протокола

1) *Completness*. Для любого правдивого statement prover может предоставить корректное доказательство

2) *Soundness*. Prover не может предоставить корректное доказательство для ложного statement.

3) *Zero Knowledge*. В процессе доказательства Verifier не может получить никакой дополнительной информации, кроме истинности statement.

Mix Bytes ()

Zero-knowledge - interactive proofs (IZK)

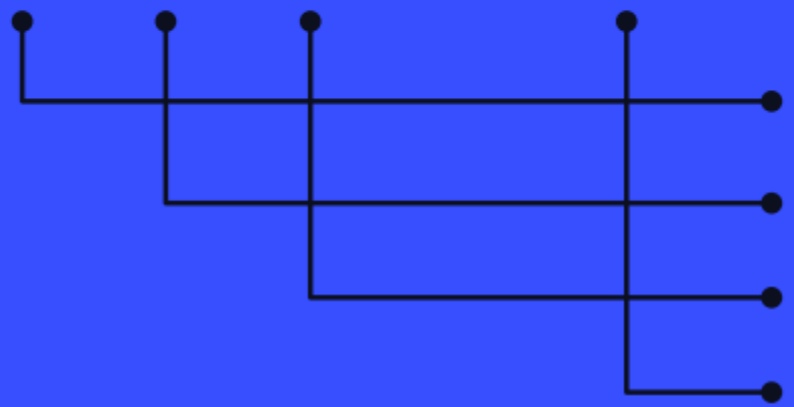
- Interactive: требуют несколько итераций между prover и verifier
  - *prover* отправляет *verifier* некоторое сообщение: *commitment*
  - *verifier* отвечает случайной битовой строкой: *challenge*
  - *prover* возвращает сообщение **z**, вычисление которого связано с *commitment*, секретным *witness* и *challenge*

Mix Bytes ()

Zero-knowledge - non-interactive proofs (NIZK)

- Non-Interactive:
  - второй шаг заменяется на Fiat-Shamir heuristic:
    - *prover* сам генерирует случайное число, и на основе него и других параметров создает *challenge* (просто хеширует параметры)
  - не надо делать обмен данными с *verifier-ом* до предоставления *proof-a*
- NIZK - единственный рабочий вариант для блокчейнов

F(X, Y) = Z



F - Your function

X - Public input

Y - Private input

Z - Public output

Proof | *Here is **X** and **Z**, I know of an **Y** such that **F(X, Y) = Z***

Подходы к созданию Zero-knowledge-систем в децентрализованных сетях

- SNARKs
- STARKs
- Bulletproofs
- Custom solutions (Aztec)

- Но есть и другие подходы☺

Применения и перспективы

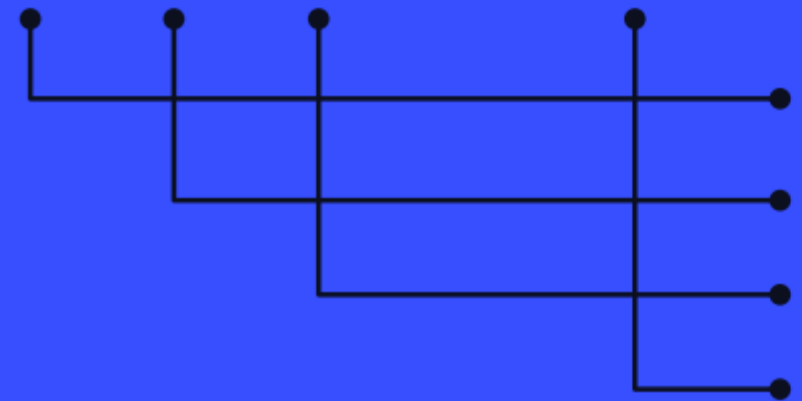
Публичные сети и кейсы

Примеры публичных сетей

- Ethereum DeFi
- Zcash
- Monero
- Coda
- Beam
- GRIN
- …

Range (Age) proof example

$$F(X, Y) = Z$$



- **F** - Your function (check 18+)
- **X** - Public input (id)
- **Y** - Private input (your age)
- **Z** - Public output (Confirmation 18+)

Proof

*Here is **X** and **Z**, I know of an **Y** such that **F(X, Y) = Z***

id надо генерить так, чтобы была возможность с помощью некоторой функции определять по id нужное доказательство - в данном случае range proof.

Mix
Bytes
()

Zero-knowledge - примеры

- Membership/Non-Membership Proofs
  - доказательство нахождения в списке
  - работа с произвольными реестрами в zk-парадигме - мечта безопасников
    - не раскрывается ни идентифицируемый, ни содержимое реестра
  - базируется на Merkle trees и zk proof-ах, доказывающих корректность Merkle proofs
  - примеры?
    - Zero-knowledge ГИБДД: проверка membership в списке имеющих права
    - Zero-knowledge ГИБДД: проверка non-membership в списке лишенных прав :)
    - вообще отлично подходит для разного рода запретов :)

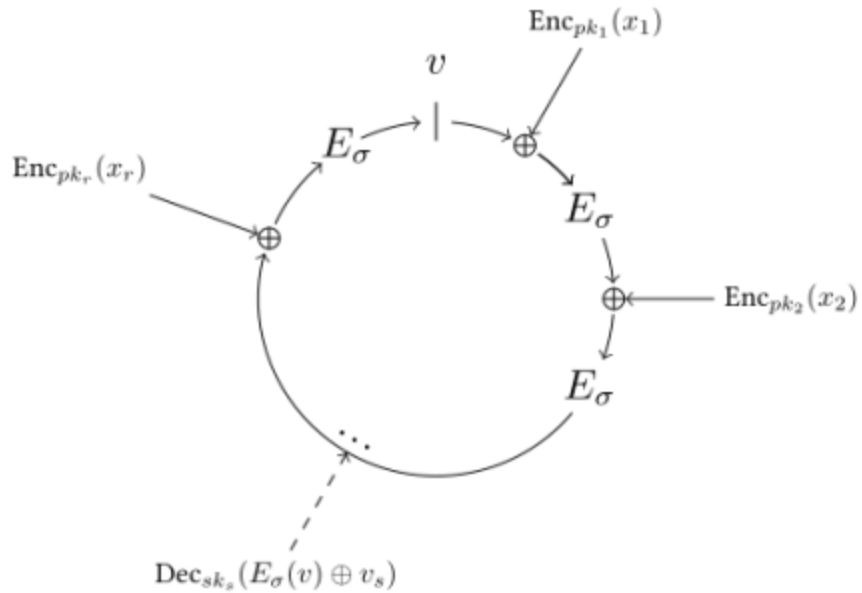
Membership proofs

- Merkle tree
- Cryptographic accumulators

Mix
Bytes
()

Zero-knowledge - примеры

- Идентификация и цифровые права
  - про кольцевые подписи(для разнообразия)
  - анонимная, отвергаемая подпись в “кольце”
    - “кто то из списка подписал сообщение”
  - примеры?
    - публичная групповая threshold подпись, например, для паллиативных или судебных решений
    - сокрытие действий для борьбы с поведенческим анализом судей, трейдеров, врачей

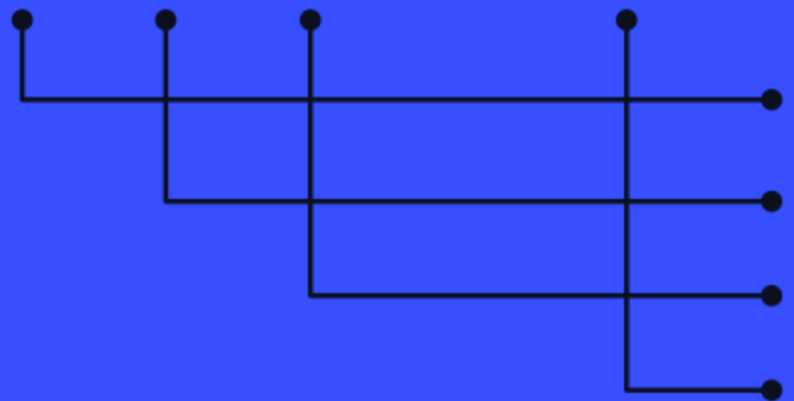


Mix
Bytes
()

Zero-knowledge - примеры

- Доказательство корректности поведения в протоколе
  - отличный пример - от Matter-ов, морской бой в смарт-контрактах на SNARK-ах
    - существует zk-proof корректности игрового поля
    - игрок публикует зашифрованное состояние поля + zk-proof
    - proof доказывает, что игрок выполнил корректный state transition из одного состояния поля в другое
    - смарт-контракт проверяет proof и фиксирует состояние игрового поля
  - вместо морского боя может быть по сути любая операция
    - которую можно затолкать в SNARK

F(X, Y) = Z

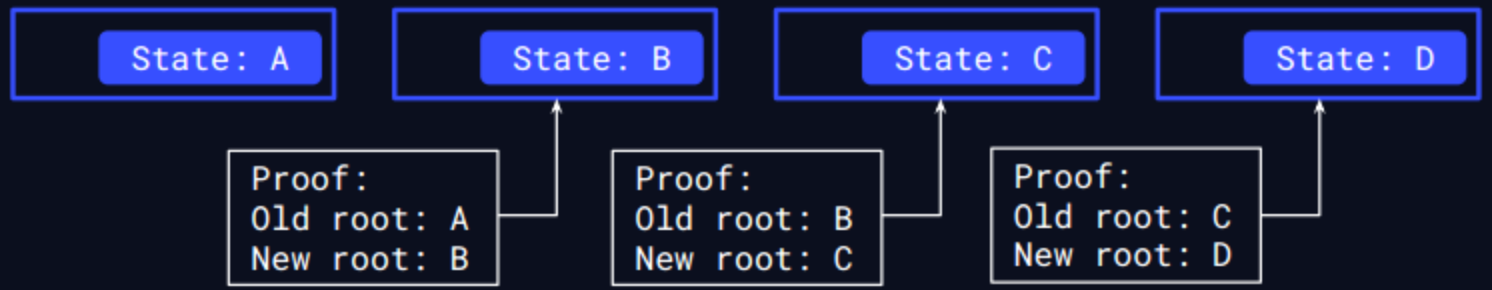


F - Verify transactions & update balances

X - Merkle root

Y - Merkle trees, signed TX's

Z - New merkle root
{new vaults, new state}



Помечтаем?

Туманные вычисления

- в теории это фундамент для всемирного суперкомпьютера
- отдаем недоверенному компьютеру задание зашифрованные входные данные
- он считает результат, подготавливая succinct zk-proof и зашифрованный результат используя homomorphic computations

примеры?

- block-producer, один блок, 10 тыс транзакций, 10 тыс проверок подписей заменяются на один proof
- map-reduce на кластере из недоверенных компьютеров

нейросеть с zk-нейронами и связями, которая работает и обучается, но её конфигурацию нельзя прочитать

публичный оплачиваемый сервис AI процессинга рентгеновских снимков без раскрытия самих снимков

Privacy

1

Secure Payments

Mixers

2

Settlement layer for DEXes

Prevent front-running attack

3

Private smart contracts

New opportunities for DeFi

Layer two scaling

Под капотом zkSNARK

zk-SNARK

- Zero-Knowledge Succinct Non-Interactive Argument of Knowledge
- Одно из наиболее изученных и реализованных семейств протоколов ZKP

Определение zkSNARK

Generator (C circuit, λ is ☣):
(pk, vk) = G(λ, C)
Prover (x pub inp, w sec inp):
π = P(pk, x, w)
Verifier:
V(vk, x, π) == (∃ w s.t. C(x,w))

```
function C(x, w) {
  return ( sha256(w) == x );
}
```

- фишка в универсальности
- в теории любая программа может быть завернута в арифметический circuit

trusted setup

- процедуры генерации доверенных параметров системы,
- после её завершения проверить эти правила не представляется возможным. Вы можете поверить в корректность генерации, но, если вы в ней не участвовали, стопроцентных гарантий у вас нет.

```
ubuntu@ip-172-31-8-131: ~/test2/sapling-mpc
ubuntu@ip-172-31-8-131:~/test2/sapling-mpc$ sudo /home/ubuntu/.cargo/bin/cargo r
un --release --bin compute
   Compiling libc v0.2.40
   Compiling nodrop v0.1.12
   Compiling futures v0.1.21
   Compiling byteorder v1.2.3
   Compiling crossbeam v0.3.2
   Compiling bit-vec v0.4.4
   Compiling constant_time_eq v0.1.3
   Compiling arrayvec v0.4.7
   Compiling rand v0.4.2
   Compiling num_cpus v1.8.0
   Compiling blake2-rfc v0.2.18
   Compiling futures-cpupool v0.1.8
   Compiling pairing v0.14.2
   Compiling bellman v0.1.0
   Compiling phase2 v0.2.2
   Compiling sapling-mpc v0.2.0 (file:///home/ubuntu/test2/sapling-mpc)
    Finished release [optimized] target(s) in 16.52s
     Running `target/release/compute`
Done!

Your contribution has been written to `./new_params`

The contribution you made is bound to the following hash:
        21ecc296 058cc017 a9b9b1b0 550d0810
        ef5aeeb1 7a21093c c8eff67e eeecc20f
        25e64637 5d69eec3 9f539c31 2a345795
        22c91d12 a8445ea0 be1af52d 2e4f9800


ubuntu@ip-172-31-8-131:~/test2/sapling-mpc$
```

- За Вилкокс (Za Wilcox), брат CEO Zcash Зуко Вилкокса (Zooko Wilcox), уничтожает компьютер, который использовался для генерации параметров алгоритма

- Все проходило на борту воздушного судна на высоте 3000 футов в небе, где-то между штатами Иллинойс и Висконсин.

- Трубка Гейгера, подсоединенная к числовому генератору, конвертировала радиоактивные импульсы от графита из ядра Чернобыльской АЭС в цифры.

Для любой NP-полной задачи существует доказательство с нулевым разглашением.

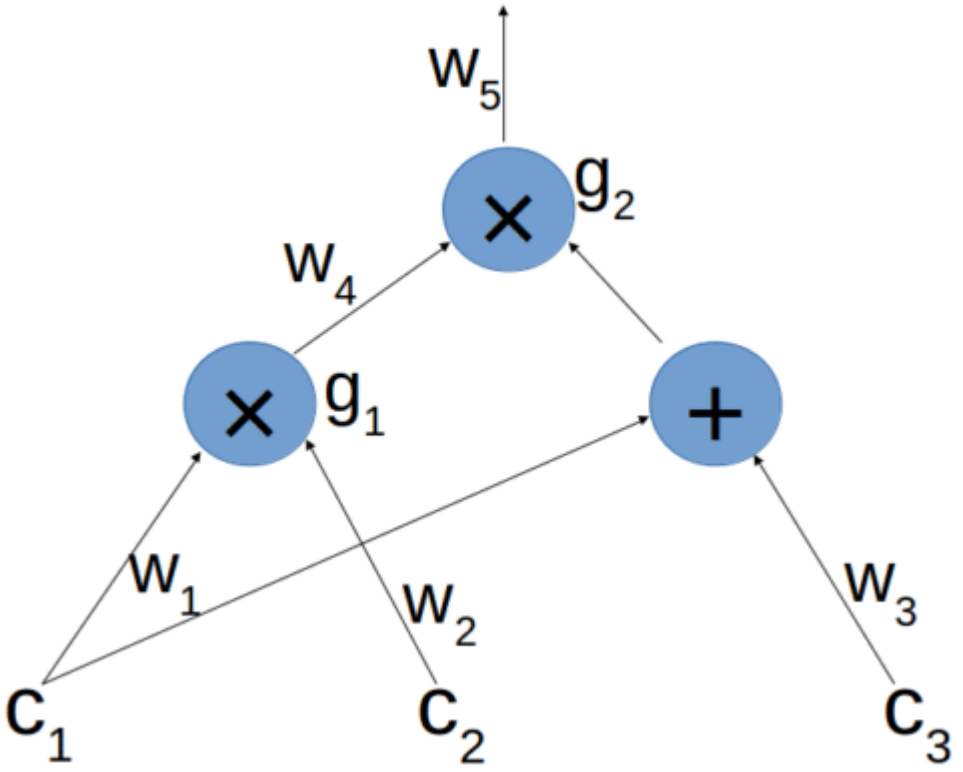
Если при этом использовать односторонние функции, то можно создать корректные криптографические протоколы.

в теории любая функция, возвращающая true/false может быть “завернута” в систему zk proof-ов

- в реальности функции ограничены (иначе получаются огромные proving key и verifying key)

How zk-SNARKs are constructed in Zcash

- Computation
- → Arithmetic Circuit - арифметизация
- → R1CS (Rank 1 Constraint System) – формирование ограничений для проверки корректности
- → QAP (Quadratic Arithmetic Program) – упаковка в единый бандл
- → zk-SNARK



Гомоморфное скрытие

ГС $E(x)$ числа x — это функция, удовлетворяющей следующим свойствам:

- Для большинства x, при известном значении $E(x)$ нахождение x является трудной задачей.
- Различные значения аргументов приводят к разным значениям функции. Поэтому, если $x \neq y$, то $E(x) \neq E(y)$
- Если кому-то известно $E(x)$ и $E(y)$, то он может генерировать ГС от арифметических операций для *x* и *y*. Например, он может вычислять $E(x + y)$, зная $E(x)$ и $E(y)$.

Гомоморфное скрытие (пример)

Алиса хочет доказать Бобу, что знает числа x, y такие, что $x + y = 7$ (Конечно, это не особо интересно, знать такие *x* и *y*, но это хороший пример для объяснения концепции).

1. Алиса отправляет $E(x)$ и $E(y)$ Бобу.
2. Боб вычисляет $E(x + y)$ из этих значений (он может это сделать, т.к. E является ГС).
3. Боб также вычисляет $E(7)$ и теперь проверяет, является ли $E(x + y) = E(7)$. Он принимает доказательство Алисы только в случае выполнения равенства.

Поскольку разные значения аргументов E соответствуют разным скрытиям (значения функции E. Прим. переводчика), Боб действительно принимает доказательство только в том случае, если Алиса отправила скрытия для x, y, такие что $x + y = 7$. С другой стороны, Боб не получает значения x, y, так как у него доступ только к их скрытиям.

Гомоморфное шифрование

- Форма шифрования, позволяющая производить определённые математические действия с зашифрованным текстом и получать зашифрованный результат, который соответствует результату операций, выполненных с открытым текстом.

Пример:

- один человек может сложить два зашифрованных числа, не зная расшифрованных чисел, а затем другой человек может расшифровать зашифрованную сумму — получить расшифрованную сумму, не имея расшифрованных чисел

Применение в Zcash

Zcash: Trusted setup (2 times)

- Sprout (oct 2016, 6 principals)
- Sapling
  - Part 1. Powers of Tau (87 participants)
  - Part 2. Sapling MPC (90+ participants from anyone)

- Multi-Party Computation (MPC)

  Each participant separately generates one *shard* of the public key, which requires them to temporarily use a corresponding private key shard. They all combine their public key shards to generate the final public parameters, and then each deletes their private key shard.

Размеры ключей Zcash

- sapling-output.params 3 509 Kb
- sapling-spend.params 46 Mb
- sprout-groth16.params 708 MB (содержит proving key and verifying key)
- sprout-proving.key 888 Mb
- sprout-verifying.key 2 Kb

- Размер proving key непосредственно зависит от количества уравнений в системе R1CS

Виды утверждений в zkSNARK Zcash (4.15)

- 4.15.1 JoinSplit Statement (Sprout)
- 4.15.2 Spend Statement (Sapling)
- 4.15.3 Output Statement (Sapling)

How zk-SNARKs are applied to create a shielded transaction

The sender of a shielded transaction constructs a proof to show that, with high probability:

- the input values sum to the output values for each shielded transfer.
- the sender proves that they have the private spending keys of the input notes, giving them the authority to spend.
- The private spending keys of the input notes are cryptographically linked to a signature over the whole transaction, in such a way that the transaction cannot be modified by a party who did not know these private keys.

How zk-SNARKs are applied to create a shielded transaction

In Zcash, the shielded equivalent of a UTXO is called a “commitment”, and spending a commitment involves revealing a “nullifier”

For each new note created by a shielded payment, a commitment is

- *Commitment = HASH(recipient address, amount, rho, r)*

When a shielded transaction is spent, the sender uses their spending key to publish a nullifier

- *Nullifier = HASH(spending key, rho)*

This hash must not already be in the set of nullifiers tracking spent transactions kept by every node in the blockchain.

The zero-knowledge proof for a shielded transaction verifies that, in addition to the conditions listed above, the following assertions are also true:

- For each input note, a revealed commitment exists.
- The nullifiers and note commitments are computed correctly.
- It is infeasible for the nullifier of an output note to collide with the nullifier of any other note.

Семейства систем доказательств с нулевым разглашением


zkSNARKs
zkSTARKs
Bulletproofs
Based on range proof &
pedersen commitments
(Monero)
Aztec
Custom privacy protocol
With custom elliptic curve
Legend
ZKP = Zero-Knowledge Proof
zkSNARK = ZK Succinct Non-Interactive ARgument of Knowledge
zkSTARK = ZK Scalable Transparent ARgument of Knowledge
AZTEC = Anonymous Z(K) Transactions with Efficient Communication

VS

SNARKs

- **Required trusted setup:** Groth16 SONIC
- **Based on elliptic curves:** BN256 for Ethereum, bls12-381 for Zcash

STARKs

- **Based on hashes in merkle trees**
- **Not proven by time**
- **Post quantum resistant**

SNARKs vs STARKs

	SNARKs	STARKS
Algorithmic complexity: power	O(N * log(N))	O(N * poly-log(N))
Algorithmic complexity: verifier	~ O(1)😄	O(poly-log(N))😒
Communication complexity (proof size)	~ O(1)😄	O(poly-log(N))😒
Size estimate for 1 TX	Tx: 200 bytes, Key: 50 MB 😄	45 kb😒
Size estimate for 10.000 TX	Tx: 200 bytes, Key: 500 MB 😄	135 kb😒
Ethereum/EVM verification gas cost	~ 600k (Groth16)😄	~ 2.5M (estimate, no impl.)😒
Trusted setup required?	YES 😒	NO 😄
Post-quantum secure	NO 😒	YES 😄
Crypto assumptions	Strong 😒	Collision resistant hashes 😄
Time to generate a proof	...	...

https://github.com/matter-labs/awesome-zero-knowledge-proofs

What are SONICs?

SONIC is a proof system, that:

- Universal
- Updatable

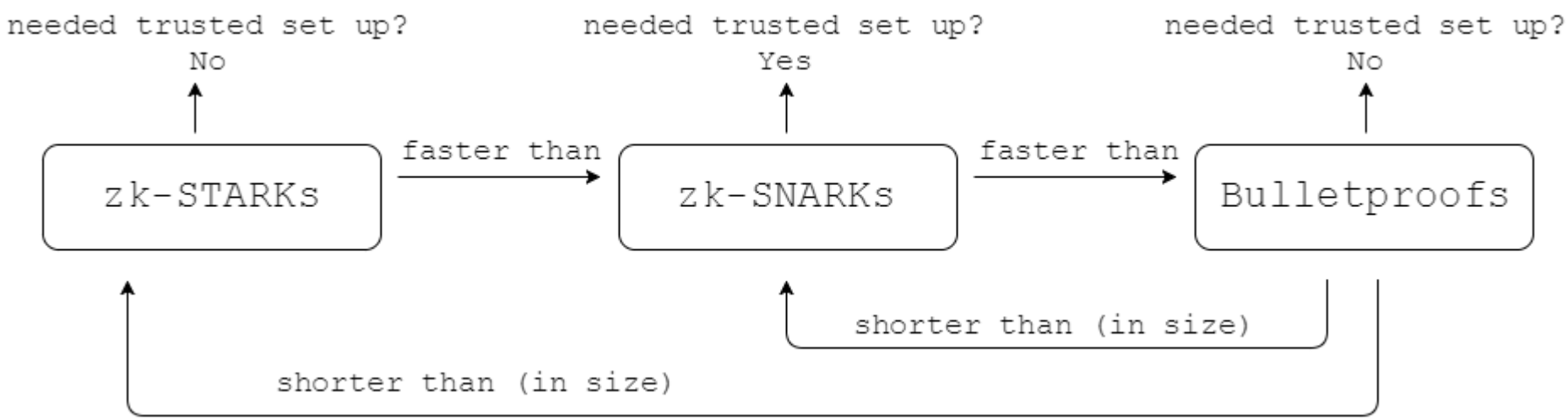
https://eprint.iacr.org/2019/099

Bulletproofs

Bulletproofs [BBB+18] is a recent technique for verifiable computation that is particularly efficient for range proofs (they take only 600 bytes). Bulletproofs has been recently implemented for a few privacy-oriented cryptocurrencies, including Monero [mon18], to reduce the range proof sizes.

Bulletproofs has the following features:

- It does not require a trusted setup as compared to ZK-SNARKs [BCG+13];
- It does not use pairings and works with any elliptic curve with a reasonably large subgroup size; the fastest elliptic curves such as Ristretto [ris18] are supported.
- It uses its own format for computation, which is easily convertible to R1CS [PHGR13] and back using linear algebra.
- The verifier cost scales linearly with the computation size.


needed trusted set up?
No
needed trusted set up?
Yes
needed trusted set up?
No
zk-STARKs
faster than
zk-SNARKs
faster than
Bulletproofs
shorter than (in size)
shorter than (in size)

Aztec Protocol

ZK даёт:

- enable private transactions on the public chain Ethereum.
- This reduces the gas costs of AZTEC transactions on the Ethereum network.

Внутри:

- range proof для защиты от double spending,
- Homomorphic encryption (allows arithmetic checks on encrypted numbers as if they weren’t encrypted)
- trusted setup

ZKP – это всегда хорошо?

Ссылки

- https://github.com/matter-labs/awesome-zero-knowledge-proofs
- https://en.wikipedia.org/wiki/Non-interactive_zero-knowledge_proof
- https://z.cash/technology/zksnarks/
- https://habr.com/ru/users/agentrx/posts/
- Cryptographic accumulators https://medium.com/@aurelcode/cryptographic-accumulators-da3aa4561d77

Спасибо

История и сложности NIZKP

- Теорема, гласящая, что для любой NP-полной задачи существует доказательство с нулевым разглашением, при этом, если использовать односторонние функции, можно создать корректные криптографические протоколы, была доказана Одедом Голдрейхом, Сильвио Микали и Ави Вигдерсоном

- Blum, Feldman, and Micali [1] showed that a common reference string shared between the prover and the verifier is enough to achieve computational zero-knowledge without requiring interaction.
- Goldreich and Oren[2] gave impossibility results for one shot zero-knowledge protocols in the standard model.
- In 2003, Shafi Goldwasser and Yael Tauman Kalai published an instance of an identification scheme for which any hash function will yield an insecure digital signature scheme.[3] These results are not contradictory, as the impossibility result of Goldreich and Oren does not hold in the common reference string model or the random oracle model.
- Non-interactive zero-knowledge proofs however show a separation between the cryptographic tasks that can be achieved in the standard model and those that can be achieved in 'more powerful' extended models.