RLN or How to Fight Spam in Anonymous Environments

Rasul Ibragimov, Software & ZK Engineer
Privacy & Scaling Explorations

О чем поговорим

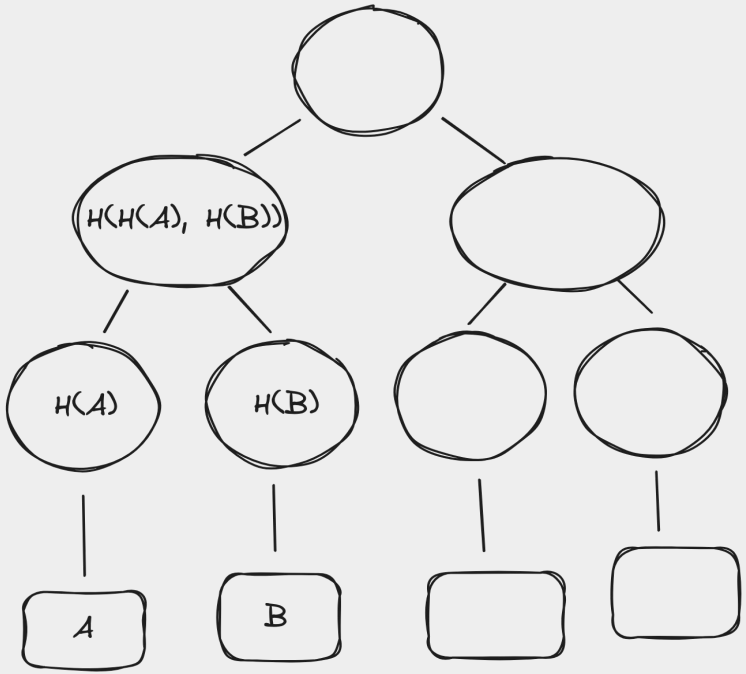
- Анонимные окружения, их отличия и проблемы
- Полиномиальная интерполяция и схема Разделения Секрета Шамира
- Протокол RLN
- Улучшения RLN

Анонимные окружения

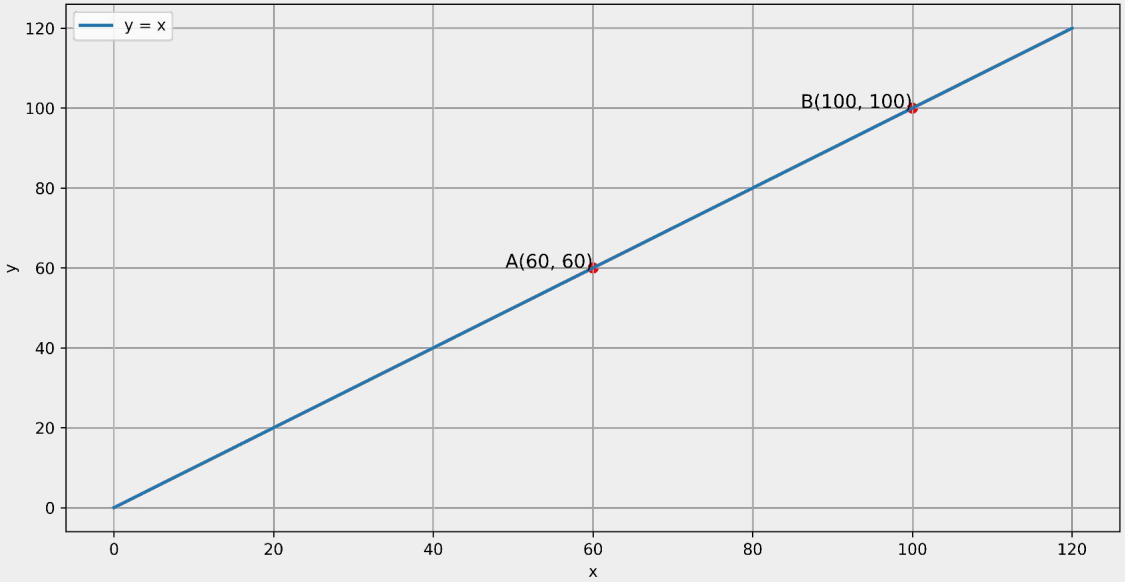
- Каждый участник “по-настоящему” анонимен
  - используются деревья Меркла для доказательства включения в группу
  - Semaphore Protocol
- Одна из проблем - спам
  - невозможно идентифицировать спамера или заблокировать его просто по IP-адресу

Дерево Меркла

- Родительская вершина содержит вычисление хэш-функции от дочерних вершин
- Самый нижний уровень – листья



Полиномиальная интерполяция



Разделение Секрета Шамира

- Позволяет разделить секрет на N частей и восстановить этот секрет при предъявлении любых K частей, где K <= N
- От того какое мы выбираем значение K зависит степень полинома. Например, при K = 2 (восстановить секрет можно при предъявлении двух точек) будет использован полином 1 степени

Разделение Секрета Шамира

- Например, наше секретное значение S = 30. Разделить секрет можно на любое количество частей, скажем на 3 (N = 3). Пусть, полином можно восстановить при предъявлении двух точек (K = 2).
- Секретный полином: *f(x) = a1 * x + a0*, где *a0* = S, а *a1* - это рандомное число (например *a1* = 5), т.е. f(x) = 5 * x + 30

Разделение Секрета Шамира

- N = 3, соответственно нужно сделать 3 вычисления полинома в рандомных точках:

  f(5) = 5 * 5 + 30 = 55 – (часть (5, 55))

  f(8) = 70

  f(16) = 110

Разделение Секрета Шамира

- Чтобы аналитически восстановить полином по двум точкам выпишем:

$$f(x) = y_1 * \frac{x - x_2}{x_1 - x_2} + y_2 * \frac{x - x_1}{x_2 - x_1}$$

где (x1, y1) и (x2, y2) - это предъявленные точки

- Если подставить две любые части, в результате получим секретный полином: f(x) = 5 * x + 30; вычислив его в 0, получим секрет: f(0) = 30

Rate-Limiting Nullifier (RLN)

- zk-протокол, использующий Semaphore и Shamir’s Secret Sharing
- Позволяет ограничить количество взаимодействий с системой в анонимном окружении (например, 1 раз за эпоху, где эпоха - это какой-нибудь промежуток времени)
- Переиспользование системы позволяет восстановить секретный ключ пользователя и удалить его из дерева Меркла

Rate-Limiting Nullifier (RLN)

- Принимает как приватный параметр - секрет (identity)
- Принимает как публичный параметр - эпоху (порядковый номер, epoch)
- Секретный полином: f(x) = a0 + a1 * x, где *a0* – это секрет, а *a1* – это hash(a0, epoch) - таким образом с одним и тем же секретом a0 каждую эпоху меняется полином, что не позволяет получить две точки/вычисления с одного и того же полинома

Rate-Limiting Nullifier (RLN)

- Ограничения zk-circuit (constraints):
  - Хэш от данного секрета является частью общего дерева Меркла
  - Часть секрета, которую предъявляет пользователь - действительно является валидной частью секрета
  - *nullifier* = hash(a1) (нуллифаер - это псевдоним пользователя системы)
- На самом деле для подсчета коэффициента *a1* также включается соль (externalNullifier), чтобы избежать коллизий между разными приложениями, использующими RLN
  - externalNullifier = hash(rlnAppId, epoch)

Улучшения **RLN**

1. В схеме используется полином 1 степени (лимит - 1 сообщение). Что если мы хотим устанавливать более высокие лимиты в эпоху?
2. Как устанавливать разные лимиты для разных пользователей, например, в зависимости от их стейка или репутации?

Улучшения **RLN**

Вместо увеличения степени полинома - можно иметь много полиномов первой степени. Этого можно добиться добавив зависимость коэффициента *a1* от дополнительного параметра-счетчика *messageId*. Также, добавить дополнительный range констрейнт, проверяющий, что *0 <= messageId < messageLimit*. Если использовать дважды тот же messageId, то раскроются две части секрета и можно будет его восстановить.

Улучшения **RLN**

При регистрации, в дерево Меркла будет добавляться rateCommitment = hash(identityCommitment, userMessageLimit), где identityCommitment = hash(secret).

По итогу изменяем констрейнты:

1. Знание и принадлежность rateCommitment дереву Меркла
2. 0 <= messageId <= userMessageLimit

Документация RLN