Bulletproofs++

Next Generation Confidential Transactions via Reciprocal Set Membership Arguments

Liam Eagen [1, 2] Sanket Kanjalkar [2] Jonas Nick [2] Tim Ruffing [2]

[1]Alpen Labs

[2]Blockstream Research

May 30, 2024

Motivation

Blockchains

- Blockchains allow decentralizing payments

Motivation

Blockchains

- Blockchains allow decentralizing payments
- Users broadcast transactions, which are added to a global ledger

Motivation

Blockchains

- Blockchains allow decentralizing payments
- Users broadcast transactions, which are added to a global ledger
- Everyone can see all transactions

Motivation

Blockchains

- Blockchains allow decentralizing payments
- Users broadcast transactions, which are added to a global ledger
- Everyone can see all transactions
- Problem: there is no privacy!

Motivation

Adding Privacy to Blockchains

- How can we recover privacy?

Motivation

Adding Privacy to Blockchains

- How can we recover privacy?
- Instead of broadcasting a transaction, broadcast a proof of knowledge of a transaction

Motivation

Adding Privacy to Blockchains

- How can we recover privacy?
- Instead of broadcasting a transaction, broadcast a proof of knowledge of a transaction
- Replace all "coins" with hiding commitments to their value

Motivation

Adding Privacy to Blockchains

- How can we recover privacy?
- Instead of broadcasting a transaction, broadcast a proof of knowledge of a transaction
- Replace all “coins” with hiding commitments to their value
- Can hide information about transactions by making proofs zero knowledge

Related Work

- Large body of work on adding private payments to blockchains

Related Work

- Large body of work on adding private payments to blockchains
- Roughly breaks into two levels

Related Work

- Large body of work on adding private payments to blockchains
- Roughly breaks into two levels
    1. Confidential transactions (CT) that just hide “internal transaction information

Related Work

- Large body of work on adding private payments to blockchains
- Roughly breaks into two levels
  1. Confidential transactions (CT) that just hide "internal transaction information
  2. Fully private transactions that hide relations betweeen transactions

Related Work

- Large body of work on adding private payments to blockchains
- Roughly breaks into two levels
  1. Confidential transactions (CT) that just hide "internal transaction information
  2. Fully private transactions that hide relations betweeen transactions
- Former includes original CT protocol of Maxwell and Bulletproofs

Related Work

- Large body of work on adding private payments to blockchains
- Roughly breaks into two levels
  1. Confidential transactions (CT) that just hide "internal transaction information
  2. Fully private transactions that hide relations betweeen transactions
- Former includes original CT protocol of Maxwell and Bulletproofs
- Latter includes original ZeroCash protocol, Zcash, Monero, etc.

Related Work

- Large body of work on adding private payments to blockchains
- Roughly breaks into two levels
  1. Confidential transactions (CT) that just hide “internal transaction information
  2. Fully private transactions that hide relations betweeen transactions
- Former includes original CT protocol of Maxwell and Bulletproofs
- Latter includes original ZeroCash protocol, Zcash, Monero, etc.
- Private transaction more powerful, but also more expensive to prove

This Work

- In this work, focus on confidential transactions

This Work

- In this work, focus on confidential transactions
- Want to hide amounts and types of assets

This Work

- In this work, focus on confidential transactions
- Want to hide amounts and types of assets
- Aim to achieve concretely small proof size, efficient verifier, without a trusted setup

This Work

- In this work, focus on confidential transactions
- Want to hide amounts and types of assets
- Aim to achieve concretely small proof size, efficient verifier, without a trusted setup
- Four main contributions

This Work

- In this work, focus on confidential transactions
- Want to hide amounts and types of assets
- Aim to achieve concretely small proof size, efficient verifier, without a trusted setup
- Four main contributions
    1. A new generalization of multiset equality arguments called the "reciprocal argument"

This Work

- In this work, focus on confidential transactions
- Want to hide amounts and types of assets
- Aim to achieve concretely small proof size, efficient verifier, without a trusted setup
- Four main contributions
  1. A new generalization of multiset equality arguments called the “reciprocal argument”
  2. An arithmetization incorporating the reciprocal argument

This Work

- In this work, focus on confidential transactions
- Want to hide amounts and types of assets
- Aim to achieve concretely small proof size, efficient verifier, without a trusted setup
- Four main contributions
  1. A new generalization of multiset equality arguments called the "reciprocal argument"
  2. An arithmetization incorporating the reciprocal argument
  3. A variant of the Bulletproof inner product argument for self-inner products called a "norm argument"

This Work

- In this work, focus on confidential transactions
- Want to hide amounts and types of assets
- Aim to achieve concretely small proof size, efficient verifier, without a trusted setup
- Four main contributions
  1. A new generalization of multiset equality arguments called the “reciprocal argument”
  2. An arithmetization incorporating the reciprocal argument
  3. A variant of the Bulletproof inner product argument for self-inner products called a “norm argument”
  4. Protocols for range proofs and CTs

Reciprocal Argument

Recap: Multiset equality arguments

- Recall a multiset equality argument checks (a_i) and (b_i) represent the same multiset

Reciprocal Argument

Recap: Multiset equality arguments

- Recall a multiset equality argument checks (a_i) and (b_i) represent the same multiset
- That is there exists permutation σ such that $a_i = b_{\sigma(i)}$

Reciprocal Argument

Recap: Multiset equality arguments

- Recall a multiset equality argument checks (a_i) and (b_i) represent the same multiset
- That is there exists permutation σ such that $a_i = b_{\sigma(i)}$
- Simple protocol due to Groth and Bayer
  1. Commit to $(a_i), (b_i)$
  2. Choose random challenge β
  3. Check $\prod_i(\beta + a_i) = \prod_i(\beta + b_i)$

Reciprocal Argument

Recap: Multiset equality arguments

- Recall a multiset equality argument checks (a_i) and (b_i) represent the same multiset
- That is there exists permutation σ such that $a_i = b_{\sigma(i)}$
- Simple protocol due to Groth and Bayer
  1. Commit to $(a_i), (b_i)$
  2. Choose random challenge β
  3. Check $\prod_i (\beta + a_i) = \prod_i (\beta + b_i)$
- Completeness follows from commutativity of multiplication

Reciprocal Argument

Recap: Multiset equality arguments

- Recall a multiset equality argument checks (a_i) and (b_i) represent the same multiset
- That is there exists permutation σ such that $a_i = b_{\sigma(i)}$
- Simple protocol due to Groth and Bayer
  1. Commit to $(a_i), (b_i)$
  2. Choose random challenge β
  3. Check $\prod_i(\beta + a_i) = \prod_i(\beta + b_i)$
- Completeness follows from commutativity of multiplication
- Can we use addition instead of multiplication?

Reciprocal Argument

- Instead of products of $\beta + a_i$ use sums of $1/(\beta + a_i)$

Reciprocal Argument

- Instead of products of $\beta + a_i$ use sums of $1/(\beta + a_i)$
- This is the "logarithmic derivative" of the Groth Bayer check

Reciprocal Argument

- Instead of products of $\beta + a_i$ use sums of $1/(\beta + a_i)$
- This is the "logarithmic derivative" of the Groth Bayer check
- Reciprocal argument generalizes multiset argument to include multiplicities

Reciprocal Argument

- Instead of products of $\beta + a_i$ use sums of $1/(\beta + a_i)$
- This is the “logarithmic derivative” of the Groth Bayer check
- Reciprocal argument generalizes multiset argument to include multiplicities
- Given a sequence (a_i, m_i) check all multiplicities for same a_i sum to zero

Reciprocal Argument

- Instead of products of $\beta + a_i$ use sums of $1/(\beta + a_i)$
- This is the "logarithmic derivative" of the Groth Bayer check
- Reciprocal argument generalizes multiset argument to include multiplicities
- Given a sequence (a_i, m_i) check all multiplicities for same a_i sum to zero
  1. Commit to (a_i, m_i)

Reciprocal Argument

- Instead of products of $\beta + a_i$ use sums of $1/(\beta + a_i)$
- This is the "logarithmic derivative" of the Groth Bayer check
- Reciprocal argument generalizes multiset argument to include multiplicities
- Given a sequence (a_i, m_i) check all multiplicities for same a_i sum to zero
  1. Commit to (a_i, m_i)
  2. Random β

Reciprocal Argument

- Instead of products of $\beta + a_i$ use sums of $1/(\beta + a_i)$
- This is the "logarithmic derivative" of the Groth Bayer check
- Reciprocal argument generalizes multiset argument to include multiplicities
- Given a sequence (a_i, m_i) check all multiplicities for same a_i sum to zero
    1. Commit to (a_i, m_i)
    2. Random β
    3. Commit to $r_i = m_i/(\beta + a_i)$

Reciprocal Argument

- Instead of products of $\beta + a_i$ use sums of $1/(\beta + a_i)$
- This is the "logarithmic derivative" of the Groth Bayer check
- Reciprocal argument generalizes multiset argument to include multiplicities
- Given a sequence (a_i, m_i) check all multiplicities for same a_i sum to zero
  1. Commit to (a_i, m_i)
  2. Random β
  3. Commit to $r_i = m_i/(\beta + a_i)$
  4. Check $\sum_i r_i = 0$ and $(\beta + a_i)r_i = m_i$

Reciprocal Argument

Applications

- We use the reciprocal argument in two ways

Reciprocal Argument

Applications

- We use the reciprocal argument in two ways
- First to build a lookup argument

Reciprocal Argument

Applications

- We use the reciprocal argument in two ways
- First to build a lookup argument
- Use this to build more efficient range proofs

Reciprocal Argument

Applications

- We use the reciprocal argument in two ways
- First to build a lookup argument
- Use this to build more efficient range proofs
- Second to build multi-asset confidential transactions

Reciprocal Argument
Applications

- We use the reciprocal argument in two ways
- First to build a lookup argument
- Use this to build more efficient range proofs
- Second to build multi-asset confidential transactions
- This keeps both amounts and kinds of tokens private

Lookup Argument

- A lookup relation requires every x_i belong to a table t_j

Lookup Argument

- A lookup relation requires every x_i belong to a table t_j
- That is, $\forall i : \exists j : x_i = t_j$

Lookup Argument

- A lookup relation requires every x_i belong to a table t_j
- That is, $\forall i : \exists j : x_i = t_j$
- Define m_j to be the number of times t_j occurs in x_i

Lookup Argument

- A lookup relation requires every x_i belong to a table t_j
- That is, $\forall i : \exists j : x_i = t_j$
- Define m_j to be the number of times t_j occurs in x_i
- Apply reciprocal argument to sequence $((-1, x_i)) \cup ((m_j, t_j))$

Lookup Argument

- A lookup relation requires every x_i belong to a table t_j
- That is, $\forall i : \exists j : x_i = t_j$
- Define m_j to be the number of times t_j occurs in x_i
- Apply reciprocal argument to sequence $((-1, x_i)) \cup ((m_j, t_j))$
- Must have number of items smaller than field characteristic

Lookup Argument

- A lookup relation requires every x_i belong to a table t_j
- That is, $\forall i : \exists j : x_i = t_j$
- Define m_j to be the number of times t_j occurs in x_i
- Apply reciprocal argument to sequence $((-1, x_i)) \cup ((m_j, t_j))$
- Must have number of items smaller than field characteristic
- Use this to build range proof with larger bases

Lookup Argument

- A lookup relation requires every x_i belong to a table t_j
- That is, $\forall i : \exists j : x_i = t_j$
- Define m_j to be the number of times t_j occurs in x_i
- Apply reciprocal argument to sequence $((-1, x_i)) \cup ((m_j, t_j))$
- Must have number of items smaller than field characteristic
- Use this to build range proof with larger bases
- $x \in [0, b^n) \iff \exists d_i \in [0, b), x = \sum_i d_i b^i$

Multi-Asset Confidential Transactions

- List of inputs I and outputs O

Multi-Asset Confidential Transactions

- List of inputs I and outputs O
- Each is a pair of an amount a and a type t

Multi-Asset Confidential Transactions

- List of inputs I and outputs O
- Each is a pair of an amount a and a type t
- Want that the amount of each type in I equals that in O

Multi-Asset Confidential Transactions

- List of inputs I and outputs O
- Each is a pair of an amount a and a type t
- Want that the amount of each type in I equals that in O
- Apply reciprocal argument to sequence $((a_i, t_i) \in I) \cup ((-a_i, t_i) \in O)$

Multi-Asset Confidential Transactions

- List of inputs I and outputs O
- Each is a pair of an amount a and a type t
- Want that the amount of each type in I equals that in O
- Apply reciprocal argument to sequence $((a_i, t_i) \in I) \cup ((-a_i, t_i) \in O)$
- Must also verify amounts are small compared to characteristic

Multi-Asset Confidential Transactions

- List of inputs I and outputs O
- Each is a pair of an amount a and a type t
- Want that the amount of each type in I equals that in O
- Apply reciprocal argument to sequence $((a_i, t_i) \in I) \cup ((-a_i, t_i) \in O)$
- Must also verify amounts are small compared to characteristic
- More fundamental advantage of reciprocal argument

What I Have Not Discussed

- Norm argument

What I Have Not Discussed

- Norm argument
- Arithmetic circuits

What I Have Not Discussed

- Norm argument
- Arithmetic circuits
- Incorporating reciprocal argument into arithmetic circuits

What I Have Not Discussed

- Norm argument
- Arithmetic circuits
- Incorporating reciprocal argument into arithmetic circuits
- How to build MACT protocol

Questions?

ia.cr/2022/510