NETHERMIND RESEARCH

Zinc

Succinct Arguments with Small Arithmetization Overheads from IOPs of Proximity to the Integers

Albert Garreta, Hendrik Waldner, Katerina Hristova, Luca Dall'ava

Arguments of knowledge

Arguments of knowledge

Let R be a relation, consisting of pairs $(x; w)$.

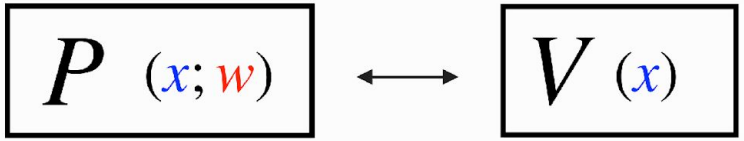
Arguments of knowledge

Let R be a relation, consisting of pairs $(x; w)$. x is a public instance, w is a witness

Arguments of knowledge

Let R be a relation, consisting of pairs $(x; w)$. x is a public instance, w is a witness

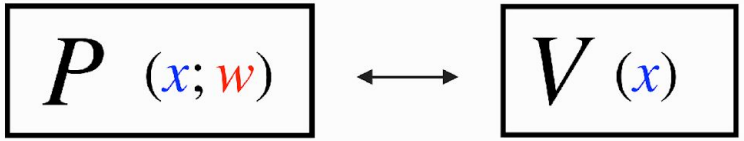
In an argument (of knowledge) Π for R,
given x, P convinces V that they know w such that $(x; w) \in R$



Arguments of knowledge

Let R be a relation, consisting of pairs $(x; w)$. x is a public instance, w is a witness

In an argument (of knowledge) Π for R,
given x, P convinces V that they know w such that $(x; w) \in R$

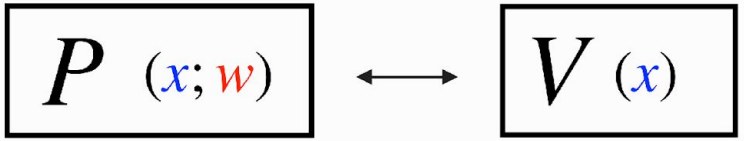


Often, R is an NP-complete relation expressed by means of polynomial equations over a finite field $\mathbb{F}$.

Arguments of knowledge

Let R be a relation, consisting of pairs $(x; w)$. x is a public instance, w is a witness

In an argument (of knowledge) Π for R,
given x, P convinces V that they know w such that $(x; w) \in R$



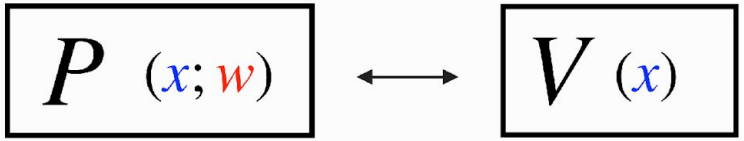
Often, R is an NP-complete relation expressed by means of polynomial equations over a finite field $\mathbb{F}$.

Example: the R1CS relation over $\mathbb{F}$

Arguments of knowledge

Let R be a relation, consisting of pairs $(x; w)$. x is a public instance, w is a witness

In an argument (of knowledge) Π for R,
given x, P convinces V that they know w such that $(x; w) \in R$



Often, R is an NP-complete relation expressed by means of polynomial equations over a finite field $\mathbb{F}$.

Example: the R1CS relation over $\mathbb{F}$

$$R_{\text{R1CS},\mathbb{F}} = \left\{ (x; w) \;\middle|\; \begin{array}{l} x \in \mathbb{F}^n,\ w \in \mathbb{F}^m \\ Az \circ Bz = Cz, \quad z = (w, x, 1) \end{array} \right\}$$

NETHERMIND RESEARCH

Arithmetization in arguments

Arithmetization in arguments

Let Π be an argument for an NP-complete relation R.

Arithmetization in arguments

Let Π be an argument for an NP-complete relation R.

What if we want to prove statements about a different NP relation R'? We can either:

Arithmetization in arguments

Let Π be an argument for an NP-complete relation R.

What if we want to prove statements about a different NP relation R'? We can either:

1. Use a different argument for R'. This is costly in terms of research and engineering

Arithmetization in arguments

Let Π be an argument for an NP-complete relation R.

What if we want to prove statements about a different NP relation R'? We can either:

1. Use a different argument for R'. This is costly in terms of research and engineering
2. "Rewrite" instance-witness pairs from R' as instance-witness pairs from R.

Arithmetization in arguments

Let Π be an argument for an NP-complete relation R.

What if we want to prove statements about a different NP relation R'? We can either:

1. Use a different argument for R'. This is costly in terms of research and engineering
2. "Rewrite" instance-witness pairs from R' as instance-witness pairs from R.
   This can be done because R is NP-complete

Arithmetization in arguments

Let Π be an argument for an NP-complete relation R.

What if we want to prove statements about a different NP relation R'? We can either:

1. Use a different argument for R'. This is costly in terms of research and engineering
2. "Rewrite" instance-witness pairs from R' as instance-witness pairs from R.

   This can be done because R is NP-complete

   We call this process arithmetization (or reduction) of R' into R.

Arithmetization in arguments

Let Π be an argument for an NP-complete relation R.

What if we want to prove statements about a different NP relation R'? We can either:

1. Use a different argument for R'. This is costly in terms of research and engineering
2. "Rewrite" instance-witness pairs from R' as instance-witness pairs from R.

   This can be done because R is NP-complete

   We call this process arithmetization (or reduction) of R' into R.

   Then P can use Π to prove it knows a witness for the transformed instance.

Examples

Examples

Common arithmetizations of a relation R' into $R_{R1CS,\mathbb{F}}$:

Examples

Common arithmetizations of a relation R' into $R_{R1CS,\mathbb{F}}$:

R’	Relevant for

Examples

Common arithmetizations of a relation R' into $R_{R1CS,\mathbb{F}}$:

R’	Relevant for
Operations mod 2^32 or 2^64	CPU operations, lattice cryptography

Examples

Common arithmetizations of a relation R' into $R_{R1CS,\mathbb{F}}$:

R’	Relevant for
Operations mod 2^32 or 2^64	CPU operations, lattice cryptography
Operations mod n=pq for p,q two primes	RSA related cryptography

Examples

Common arithmetizations of a relation R' into $R_{R1CS,\mathbb{F}}$:

R’	Relevant for
Operations mod 2^32 or 2^64	CPU operations, lattice cryptography
Operations mod n=pq for p,q two primes	RSA related cryptography
Operations with rational numbers	ML, finance

Examples

Common arithmetizations of a relation R' into $R_{R1CS,\mathbb{F}}$:

R'	Relevant for
Operations mod 2^32 or 2^64	CPU operations, lattice cryptography
Operations mod n=pq for p,q two primes	RSA related cryptography
Operations with rational numbers	ML, finance
Operations over lattices and Galois rings	Lattice-based cryptography, FHE

Examples

Common arithmetizations of a relation R' into $R_{R1CS,\mathbb{F}}$:

R’	Relevant for
Operations mod 2^32 or 2^64	CPU operations, lattice cryptography
Operations mod n=pq for p,q two primes	RSA related cryptography
Operations with rational numbers	ML, finance
Operations over lattices and Galois rings	Lattice-based cryptography, FHE
Operations over a non-native prime	Recursive proving, IVC, PCD, etc.

Examples

Common arithmetizations of a relation R' into $R_{R1CS,\mathbb{F}}$:

R'	Relevant for
Operations mod 2^32 or 2^64	CPU operations, lattice cryptography
Operations mod n=pq for p,q two primes	RSA related cryptography
Operations with rational numbers	ML, finance
Operations over lattices and Galois rings	Lattice-based cryptography, FHE
Operations over a non-native prime	Recursive proving, IVC, PCD, etc.

Arithmetization can increase sizes of instance-witness by 16x-32x, or more.

Examples

Common arithmetizations of a relation R' into $R_{R1CS,\mathbb{F}}$:

R'	Relevant for
Operations mod 2^32 or 2^64	CPU operations, lattice cryptography
Operations mod n=pq for p,q two primes	RSA related cryptography
Operations with rational numbers	ML, finance
Operations over lattices and Galois rings	Lattice-based cryptography, FHE
Operations over a non-native prime	Recursive proving, IVC, PCD, etc.

Arithmetization can increase sizes of instance-witness by 16x-32x, or more.

Example: ECDSA verification over a non-scalar field has 2^{21} R1CS constraints, vs 2^{16}.

Examples

Common arithmetizations of a relation R' into $R_{R1CS,\mathbb{F}}$:

R'	Relevant for
Operations mod 2^32 or 2^64	CPU operations, lattice cryptography
Operations mod n=pq for p,q two primes	RSA related cryptography
Operations with rational numbers	ML, finance
Operations over lattices and Galois rings	Lattice-based cryptography, FHE
Operations over a non-native prime	Recursive proving, IVC, PCD, etc.

Arithmetization can increase sizes of instance-witness by 16x-32x, or more.

Example: ECDSA verification over a non-scalar field has 2^{21} R1CS constraints, vs 2^{16}.

We call this factor the arithmetization overhead.

Examples

Common arithmetizations of a relation R' into $R_{R1CS,\mathbb{F}}$:

R'	Relevant for
Operations mod 2^32 or 2^64	CPU operations, lattice cryptography
Operations mod n=pq for p,q two primes	RSA related cryptography
Operations with rational numbers	ML, finance
Operations over lattices and Galois rings	Lattice-based cryptography, FHE
Operations over a non-native prime	Recursive proving, IVC, PCD, etc.

Arithmetization can increase sizes of instance-witness by 16x-32x, or more.

Example: ECDSA verification over a non-scalar field has 2^{21} R1CS constraints, vs 2^{16}.

We call this factor the arithmetization overhead.

We believe that reducing this overhead is key to improving SNARK efficiency.

Our goals

Our goals

Design a relation R and a SNARK for R such that:

Our goals

Design a relation R and a SNARK for R such that:

Universal arithmetization

Our goals

Design a relation R and a SNARK for R such that:

Universal arithmetization

Similar to SNARKs used in practice

Our goals

Design a relation R and a SNARK for R such that:

Universal arithmetization

We can arithmetize all/most relations of interest R' into R with almost no overhead

Similar to SNARKs used in practice

Our goals

Design a relation R and a SNARK for R such that:

Universal arithmetization

We can arithmetize all/most relations of interest R' into R with almost no overhead

Similar to SNARKs used in practice

Runs over finite fields

Our goals

Design a relation R and a SNARK for R such that:

Universal arithmetization

We can arithmetize all/most relations of interest R' into R with almost no overhead

Similar to SNARKs used in practice

Runs over finite fields

Error correcting codes

Our goals

Design a relation R and a SNARK for R such that:

Universal arithmetization

We can arithmetize all/most relations of interest R' into R with almost no overhead

Similar to SNARKs used in practice

Runs over finite fields

Error correcting codes

Hash functions

Our goals

Design a relation R and a SNARK for R such that:

Universal arithmetization

We can arithmetize all/most relations of interest R' into R with almost no overhead

Similar to SNARKs used in practice

Runs over finite fields

Error correcting codes

Hash functions

STARKs, Plonky, Brakedown, etc.

Our goals

Design a relation R and a SNARK for R such that:

Universal arithmetization

We can arithmetize all/most relations of interest R' into R with almost no overhead

Similar to SNARKs used in practice

Runs over finite fields

Error correcting codes

Hash functions

STARKs, Plonky, Brakedown, etc.

Relations R' of interest = the ones in our example table.

Our goals

Design a relation R and a SNARK for R such that:

Universal arithmetization

We can arithmetize all/most relations of interest R' into R with almost no overhead

Similar to SNARKs used in practice

Runs over finite fields

Error correcting codes

Hash functions

STARKs, Plonky, Brakedown, etc.

Relations R' of interest = the ones in our example table.

As first step, we limit ourselves to R1CS over $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{Z}$, $\mathbb{Q}$ for arbitrary $n \geq 1$.

Our goals

Design a relation R and a SNARK for R such that:

Universal arithmetization

We can arithmetize all/most relations of interest R' into R with almost no overhead

Similar to SNARKs used in practice

Runs over finite fields

Error correcting codes

Hash functions

STARKs, Plonky, Brakedown, etc.

Relations R' of interest = the ones in our example table.

As first step, we limit ourselves to R1CS over $\mathbb{Z}/n\mathbb{Z}, \mathbb{Z}, \mathbb{Q}$ for arbitrary $n \geq 1$.

Our ideas can be extended to other rings (WIP)

NETHERMIND RESEARCH

Zinc

*

NETHERMIND RESEARCH

Zinc

We present a framework and family of succinct arguments that:

*

NETHERMIND RESEARCH

Zinc

We present a framework and family of succinct arguments that:

Spartan + Brakedown

*

NETHERMIND RESEARCH

Zinc

We present a framework and family of succinct arguments that:

Spartan + Brakedown

In its execution, Zinc is simply Spartan + Brakedown (S+B) run over a random finite field.

*

NETHERMIND RESEARCH

Zinc

We present a framework and family of succinct arguments that:

Algebraic relations over $\mathbb{Z}$ and $\mathbb{Q}$

Spartan + Brakedown

In its execution, Zinc is simply Spartan + Brakedown (S+B) run over a random finite field.

*

NETHERMIND RESEARCH

Zinc

We present a framework and family of succinct arguments that:

Algebraic relations over $\mathbb{Z}$ and $\mathbb{Q}$

Zinc can handle polynomial constraints over $\mathbb{Z}$, and even $\mathbb{Q}$.

Spartan + Brakedown

In its execution, Zinc is simply Spartan + Brakedown (S+B) run over a random finite field.

*

Zinc

We present a framework and family of succinct arguments that:

Algebraic relations over $\mathbb{Z}$ and $\mathbb{Q}$

Zinc can handle polynomial constraints over $\mathbb{Z}$, and even $\mathbb{Q}$.

It's easy to arithmetize many relations of interest into such type of relation.

Spartan + Brakedown

In its execution, Zinc is simply Spartan + Brakedown (S+B) run over a random finite field.

*

Zinc

We present a framework and family of succinct arguments that:

Algebraic relations over $\mathbb{Z}$ and $\mathbb{Q}$

Zinc can handle polynomial constraints over $\mathbb{Z}$, and even $\mathbb{Q}$.

It's easy to arithmetize many relations of interest into such type of relation.

Spartan + Brakedown

In its execution, Zinc is simply Spartan + Brakedown (S+B) run over a random finite field.

Performance

*

Zinc

We present a framework and family of succinct arguments that:

Algebraic relations over $\mathbb{Z}$ and $\mathbb{Q}$

Zinc can handle polynomial constraints over $\mathbb{Z}$, and even $\mathbb{Q}$.

It's easy to arithmetize many relations of interest into such type of relation.

Spartan + Brakedown

In its execution, Zinc is simply Spartan + Brakedown (S+B) run over a random finite field.

Performance

(Preliminary): Zinc is only 2-3x slower than standard implementations of S+B.

*

NETHERMIND RESEARCH

Zinc

We present a framework and family of succinct arguments that:

Algebraic relations over $\mathbb{Z}$ and $\mathbb{Q}$

Zinc can handle polynomial constraints over $\mathbb{Z}$, and even $\mathbb{Q}$.

It's easy to arithmetize many relations of interest into such type of relation.

Spartan + Brakedown

In its execution, Zinc is simply Spartan + Brakedown (S+B) run over a random finite field.

Performance

(Preliminary): Zinc is only 2-3x slower than standard implementations of S+B.

Proof time for R1CS with 2^{16} constraints: $\approx$ 250-500ms*

NETHERMIND RESEARCH

Zinc

We present a framework and family of succinct arguments that:

Algebraic relations over $\mathbb{Z}$ and $\mathbb{Q}$

Zinc can handle polynomial constraints over $\mathbb{Z}$, and even $\mathbb{Q}$.

It's easy to arithmetize many relations of interest into such type of relation.

Spartan + Brakedown

In its execution, Zinc is simply Spartan + Brakedown (S+B) run over a random finite field.

Performance

(Preliminary): Zinc is only 2-3x slower than standard implementations of S+B.

Proof time for R1CS with 2^{16} constraints: $\approx$ 250-500ms *

* With R1CS matrices being the identity matrix

NETHERMIND RESEARCH

Zinc

We present a framework and family of succinct arguments that:

Algebraic relations over $\mathbb{Z}$ and $\mathbb{Q}$

Zinc can handle polynomial constraints over $\mathbb{Z}$, and even $\mathbb{Q}$.

It's easy to arithmetize many relations of interest into such type of relation.

IOP of proximity to $\mathbb{Z}$

Spartan + Brakedown

In its execution, Zinc is simply Spartan + Brakedown (S+B) run over a random finite field.

Performance

(Preliminary): Zinc is only 2-3x slower than standard implementations of S+B.

Proof time for R1CS with 2^{16} constraints: $\approx$ 250-500ms *

* With R1CS matrices being the identity matrix

NETHERMIND RESEARCH

Zinc

We present a framework and family of succinct arguments that:

Algebraic relations over $\mathbb{Z}$ and $\mathbb{Q}$

Zinc can handle polynomial constraints over $\mathbb{Z}$, and even $\mathbb{Q}$.

It's easy to arithmetize many relations of interest into such type of relation.

IOP of proximity to $\mathbb{Z}$

New primitive

Spartan + Brakedown

In its execution, Zinc is simply Spartan + Brakedown (S+B) run over a random finite field.

Performance

(Preliminary): Zinc is only 2-3x slower than standard implementations of S+B.

Proof time for R1CS with 2^{16} constraints: $\approx$ 250-500ms *

* With R1CS matrices being the identity matrix

NETHERMIND RESEARCH

Initial timid attempts

Initial timid attempts

Let's start by trying to build a proof system for R1CS over $\mathbb{Z}/n\mathbb{Z}$.

Initial timid attempts

Let's start by trying to build a proof system for R1CS over $\mathbb{Z}/n\mathbb{Z}$.

$$R_{\text{R1CS},\mathbb{Z}/n\mathbb{Z}} = \left\{ (x; w) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}/n\mathbb{Z}^m,\ w \in \mathbb{Z}/n\mathbb{Z}^k \\ Az \circ Bz = Cz \ \text{ over } \mathbb{Z}/n\mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Initial timid attempts

Let's start by trying to build a proof system for R1CS over $\mathbb{Z}/n\mathbb{Z}$.

$$R_{\text{R1CS},\mathbb{Z}/n\mathbb{Z}} = \left\{ (x; w) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}/n\mathbb{Z}^m,\ w \in \mathbb{Z}/n\mathbb{Z}^k \\ Az \circ Bz = Cz \ \text{ over } \mathbb{Z}/n\mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Observe: $x = y$ over $\mathbb{Z}/n\mathbb{Z}$ if and only if $x = y + v \cdot n$ over $\mathbb{Z}$, for some $v \in \mathbb{Z}$.

Initial timid attempts

Let's start by trying to build a proof system for R1CS over $\mathbb{Z}/n\mathbb{Z}$.

$$R_{\text{R1CS},\mathbb{Z}/n\mathbb{Z}} = \left\{ (x; w) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}/n\mathbb{Z}^m,\ w \in \mathbb{Z}/n\mathbb{Z}^k \\ Az \circ Bz = Cz \ \text{ over } \mathbb{Z}/n\mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Observe: $x = y$ over $\mathbb{Z}/n\mathbb{Z}$ if and only if $x = y + v \cdot n$ over $\mathbb{Z}$, for some $v \in \mathbb{Z}$.

$$Az \circ Bz = Cz \text{ over } \mathbb{Z}/n\mathbb{Z}$$

Initial timid attempts

Let's start by trying to build a proof system for R1CS over $\mathbb{Z}/n\mathbb{Z}$.

$$R_{\text{R1CS},\mathbb{Z}/n\mathbb{Z}} = \left\{ (x; w) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}/n\mathbb{Z}^m,\ w \in \mathbb{Z}/n\mathbb{Z}^k \\ Az \circ Bz = Cz \ \text{ over } \mathbb{Z}/n\mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Observe: $x = y$ over $\mathbb{Z}/n\mathbb{Z}$ if and only if $x = y + v \cdot n$ over $\mathbb{Z}$, for some $v \in \mathbb{Z}$.

$Az \circ Bz = Cz$ over $\mathbb{Z}/n\mathbb{Z}$ $\Longleftrightarrow$

Initial timid attempts

Let's start by trying to build a proof system for R1CS over $\mathbb{Z}/n\mathbb{Z}$.

$$R_{\text{R1CS},\mathbb{Z}/n\mathbb{Z}} = \left\{ (x; w) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}/n\mathbb{Z}^m,\ w \in \mathbb{Z}/n\mathbb{Z}^k \\ Az \circ Bz = Cz \ \text{ over } \mathbb{Z}/n\mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Observe: $x = y$ over $\mathbb{Z}/n\mathbb{Z}$ if and only if $x = y + v \cdot n$ over $\mathbb{Z}$, for some $v \in \mathbb{Z}$.

$Az \circ Bz = Cz$ over $\mathbb{Z}/n\mathbb{Z}$ ⟺ There exists $u \in \mathbb{Z}^{m+k+1}$ such that

Initial timid attempts

Let's start by trying to build a proof system for R1CS over $\mathbb{Z}/n\mathbb{Z}$.

$$R_{\text{R1CS},\mathbb{Z}/n\mathbb{Z}} = \left\{ (x; w) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}/n\mathbb{Z}^m,\ w \in \mathbb{Z}/n\mathbb{Z}^k \\ Az \circ Bz = Cz \ \text{ over } \mathbb{Z}/n\mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Observe: $x = y$ over $\mathbb{Z}/n\mathbb{Z}$ if and only if $x = y + v \cdot n$ over $\mathbb{Z}$, for some $v \in \mathbb{Z}$.

$Az \circ Bz = Cz$ over $\mathbb{Z}/n\mathbb{Z}$ ⟺ There exists $u \in \mathbb{Z}^{m+k+1}$ such that $Az \circ Bz = Cz + u \cdot n$ as integers

Initial timid attempts

Let's start by trying to build a proof system for R1CS over $\mathbb{Z}/n\mathbb{Z}$.

$$R_{\text{R1CS},\mathbb{Z}/n\mathbb{Z}} = \left\{ (x; w) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}/n\mathbb{Z}^m,\ w \in \mathbb{Z}/n\mathbb{Z}^k \\ Az \circ Bz = Cz \ \text{ over } \mathbb{Z}/n\mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Observe: $x = y$ over $\mathbb{Z}/n\mathbb{Z}$ if and only if $x = y + v \cdot n$ over $\mathbb{Z}$, for some $v \in \mathbb{Z}$.

$Az \circ Bz = Cz$ over $\mathbb{Z}/n\mathbb{Z}$

There exists $u \in \mathbb{Z}^{m+k+1}$ such that
$Az \circ Bz = Cz + u \cdot n$ as integers

So, let's try to build a proof system for:

Initial timid attempts

Let's start by trying to build a proof system for R1CS over $\mathbb{Z}/n\mathbb{Z}$.

$$R_{\text{R1CS},\mathbb{Z}/n\mathbb{Z}} = \left\{ (x; w) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}/n\mathbb{Z}^m,\ w \in \mathbb{Z}/n\mathbb{Z}^k \\ Az \circ Bz = Cz \ \text{ over } \mathbb{Z}/n\mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Observe: $x = y$ over $\mathbb{Z}/n\mathbb{Z}$ if and only if $x = y + v \cdot n$ over $\mathbb{Z}$, for some $v \in \mathbb{Z}$.

$Az \circ Bz = Cz$ over $\mathbb{Z}/n\mathbb{Z}$ $\iff$ There exists $u \in \mathbb{Z}^{m+k+1}$ such that $Az \circ Bz = Cz + u \cdot n$ as integers

So, let's try to build a proof system for:

$$R_{\text{R1CS}\ell,\mathbb{Z}} = \left\{ (x; w, u) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}^m,\ w \in \mathbb{Z}^k,\ u \in \mathbb{Z}^{m+k+1} \\ Az \circ Bz = Cz + u \cdot n \ \text{ over } \mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Initial timid attempts

Let's start by trying to build a proof system for R1CS over $\mathbb{Z}/n\mathbb{Z}$.

$$R_{\text{R1CS},\mathbb{Z}/n\mathbb{Z}} = \left\{ (x; w) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}/n\mathbb{Z}^m,\ w \in \mathbb{Z}/n\mathbb{Z}^k \\ Az \circ Bz = Cz \ \text{ over } \mathbb{Z}/n\mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Observe: $x = y$ over $\mathbb{Z}/n\mathbb{Z}$ if and only if $x = y + v \cdot n$ over $\mathbb{Z}$, for some $v \in \mathbb{Z}$.

$Az \circ Bz = Cz$ over $\mathbb{Z}/n\mathbb{Z}$ $\Longleftrightarrow$ There exists $u \in \mathbb{Z}^{m+k+1}$ such that $Az \circ Bz = Cz + u \cdot n$ as integers

So, let's try to build a proof system for:

$$R_{\text{R1CS}\ell,\mathbb{Z}} = \left\{ (x; w, u) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}^m,\ w \in \mathbb{Z}^k,\ u \in \mathbb{Z}^{m+k+1} \\ \cancel{Az \circ Bz = Cz + u \cdot n} \ \text{ over } \mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

$Az \circ Bz = Cz + u \circ d$ ← $d \in \mathbb{Z}^{m+k+1}$ different moduli per constraint

Initial timid attempts

Let's start by trying to build a proof system for R1CS over $\mathbb{Z}/n\mathbb{Z}$.

$$R_{\text{R1CS},\mathbb{Z}/n\mathbb{Z}} = \left\{ (x; w) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}/n\mathbb{Z}^m,\ w \in \mathbb{Z}/n\mathbb{Z}^k \\ Az \circ Bz = Cz \ \text{ over } \mathbb{Z}/n\mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Observe: $x = y$ over $\mathbb{Z}/n\mathbb{Z}$ if and only if $x = y + v \cdot n$ over $\mathbb{Z}$, for some $v \in \mathbb{Z}$.

$Az \circ Bz = Cz$ over $\mathbb{Z}/n\mathbb{Z}$ $\Longleftrightarrow$ There exists $u \in \mathbb{Z}^{m+k+1}$ such that $Az \circ Bz = Cz + u \cdot n$ as integers

So, let's try to build a proof system for:

$$R_{\text{R1CS}\ell,\mathbb{Z},B} = \left\{ (x; w, u) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}_B^m,\ w \in \mathbb{Z}_B^k,\ u \in \mathbb{Z}_B^{m+k+1} \\ \cancel{Az \circ Bz = Cz + u \circ n} \ \text{ over } \mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

$$Az \circ Bz = Cz + u \circ d \quad \leftarrow \quad d \in \mathbb{Z}^{m+k+1} \text{ different moduli per constraint}$$

$\mathbb{Z}_B$ is the set of integers with bit-size less than B.

Naïve attempt: A succinct argument for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

Naïve attempt: A succinct argument for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

Let's build a succinct argument Π for the following relation:

Naïve attempt: A succinct argument for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

Let's build a succinct argument Π for the following relation:

$$R_{\mathsf{R1CS}\ell,\mathbb{Z},B} = \left\{ (x; w, u) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}_B^m,\ w \in \mathbb{Z}_B^k,\ u \in \mathbb{Z}_B^{m+k+1} \\ Az \circ Bz = Cz + u \circ d \ \text{ over } \mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Naïve attempt: A succinct argument for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

Let's build a succinct argument Π for the following relation:

$$R_{\mathsf{R1CS}\ell,\mathbb{Z},B} = \left\{ (x; w, u) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}_B^m,\ w \in \mathbb{Z}_B^k,\ u \in \mathbb{Z}_B^{m+k+1} \\ Az \circ Bz = Cz + u \circ d \ \text{ over } \mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Adapt Spartan PIOP (or SuperSpartan) for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

Naïve attempt: A succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Let's build a succinct argument Π for the following relation:

$$R_{\text{R1CS}\ell,\mathbb{Z},B} = \left\{ (x; w, u) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}_B^m,\ w \in \mathbb{Z}_B^k,\ u \in \mathbb{Z}_B^{m+k+1} \\ Az \circ Bz = Cz + u \circ d \ \text{ over } \mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Prove it is sound against P^* that use polys over $\mathbb{Z}_B$

Naïve attempt: A succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Let's build a succinct argument Π for the following relation:

$$R_{\text{R1CS}\ell,\mathbb{Z},B} = \left\{ (x; w, u) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}_B^m,\ w \in \mathbb{Z}_B^k,\ u \in \mathbb{Z}_B^{m+k+1} \\ Az \circ Bz = Cz + u \circ d \ \text{ over } \mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Prove it is sound against P^* that use polys over $\mathbb{Z}_B$

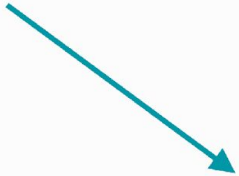
Design a PCS for polynomials over $\mathbb{Z}_B$

Naïve attempt: A succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Let's build a succinct argument Π for the following relation:

$$R_{\text{R1CS}\ell,\mathbb{Z},B} = \left\{ (x; w, u) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}_B^m,\ w \in \mathbb{Z}_B^k,\ u \in \mathbb{Z}_B^{m+k+1} \\ Az \circ Bz = Cz + u \circ d \ \text{ over } \mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$



Prove it is sound against P^* that use polys over $\mathbb{Z}_B$

Design a PCS for polynomials over $\mathbb{Z}_B$

Naïve attempt: A succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

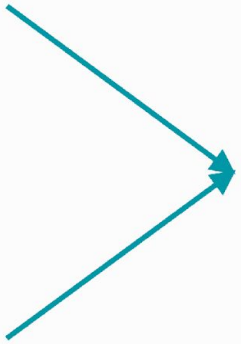
Let's build a succinct argument Π for the following relation:

$$R_{\text{R1CS}\ell,\mathbb{Z},B} = \left\{ (x; w, u) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}_B^m,\ w \in \mathbb{Z}_B^k,\ u \in \mathbb{Z}_B^{m+k+1} \\ Az \circ Bz = Cz + u \circ d \ \text{ over } \mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Prove it is sound against P^* that use polys over $\mathbb{Z}_B$

Design a PCS for polynomials over $\mathbb{Z}_B$



Naïve attempt: A succinct argument for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

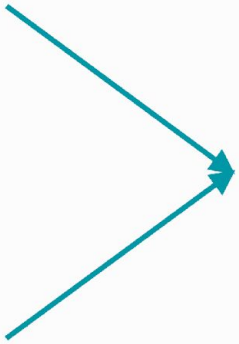
Let's build a succinct argument Π for the following relation:

$$R_{\mathsf{R1CS}\ell,\mathbb{Z},B} = \left\{ (x; w, u) \;\middle|\; \begin{array}{l} x \in \mathbb{Z}_B^m,\ w \in \mathbb{Z}_B^k,\ u \in \mathbb{Z}_B^{m+k+1} \\ Az \circ Bz = Cz + u \circ d \ \text{ over } \mathbb{Z}, \quad z = (w, x, 1) \end{array} \right\}$$

Adapt Spartan PIOP (or SuperSpartan) for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

Prove it is sound against P^* that use polys over $\mathbb{Z}_B$

Design a PCS for polynomials over $\mathbb{Z}_B$



Compile into succinct argument for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

NETHERMIND RESEARCH

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Issues:

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Issues:

(1) requires operating with integers of thousands of bits.

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Issues:

(1) requires operating with integers of thousands of bits.

(2) is a very strong primitive.

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Issues:

(1) requires operating with integers of thousands of bits.

(2) is a very strong primitive.

The only one we are aware of is due to Block et al.

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Issues:

(1) requires operating with integers of thousands of bits.

(2) is a very strong primitive.

The only one we are aware of is due to Block et al.

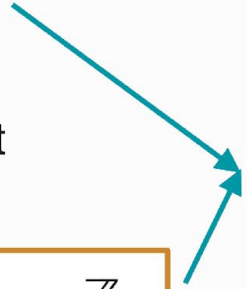
Based on the DARK scheme. Uses hidden order groups. Is slow in practice.

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$



Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Issues:

(1) requires operating with integers of thousands of bits.

(2) is a very strong primitive.

The only one we are aware of is due to Block et al.

Based on the DARK scheme. Uses hidden order groups. Is slow in practice.

Let's modify our naïve attempt so as to address these issues.

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) requires operating with integers of thousands of bits.

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) requires operating with integers of thousands of bits.

Solution: Campanelli and Hall-Andersen [CH2024]: have V sample a random prime q.

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) requires operating with integers of thousands of bits.

Solution: Campanelli and Hall-Andersen [CH2024]: have V sample a random prime q.

Then execute Spartan over $\mathbb{F}_q$, rather than over $\mathbb{Z}_B$.

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) requires operating with integers of thousands of bits.

Solution: Campanelli and Hall-Andersen [CH2024]: have V sample a random prime q.

Then execute Spartan over $\mathbb{F}_q$, rather than over $\mathbb{Z}_B$.

They call the resulting PIOP a *mod-PIOP* (or *mod-AHP*).

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

With soundness holding against P^* that use polys over $\mathbb{Z}_B$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) requires operating with integers of thousands of bits.

Solution: Campanelli and Hall-Andersen [CH2024]: have V sample a random prime q.

Then execute Spartan over $\mathbb{F}_q$, rather than over $\mathbb{Z}_B$.

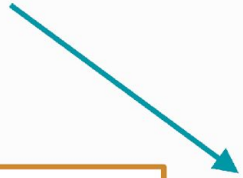
They call the resulting PIOP a *mod-PIOP* (or *mod-AHP*).

[CH2024] compile mod-PIOPs with (2) into a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

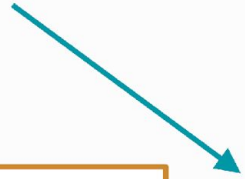


Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$



Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) is a very strong primitive.

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

→ Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) is a very strong primitive.

[CH2024] propose using the PCS over $\mathbb{Z}_B$ of Block et al.

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

→ Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) is a very strong primitive.

[CH2024] propose using the PCS over $\mathbb{Z}_B$ of Block et al.

Difficult to do better because extraction procedures often yield a rational polynomial.

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

→ Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) is a very strong primitive.

[CH2024] propose using the PCS over $\mathbb{Z}_B$ of Block et al.

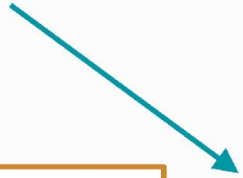
Difficult to do better because extraction procedures often yield a rational polynomial.

(Coming from solving systems of equations over $\mathbb{Z}$)

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

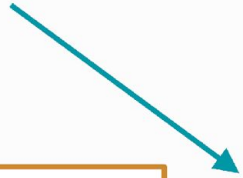


Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$



Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

Designing a succinct argument for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

→ Compile into succinct argument for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

Designing a succinct argument for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

→ Compile into succinct argument for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

Designing a succinct argument for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

→ Compile into succinct argument for $R_{\mathsf{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

PIOP for $R_{\mathsf{R1CS}\ell,\mathbb{Q},B}$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

→ Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Soundness holds against P^* that use polys over $\mathbb{Q}_B$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Design a PCS for polynomials over $\mathbb{Q}_B$

Soundness holds against P^* that use polys over $\mathbb{Q}_B$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

→ Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Design a PCS for polynomials over $\mathbb{Q}_B$

→ Soundness holds against P^* that use polys over $\mathbb{Q}_B$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

→ Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

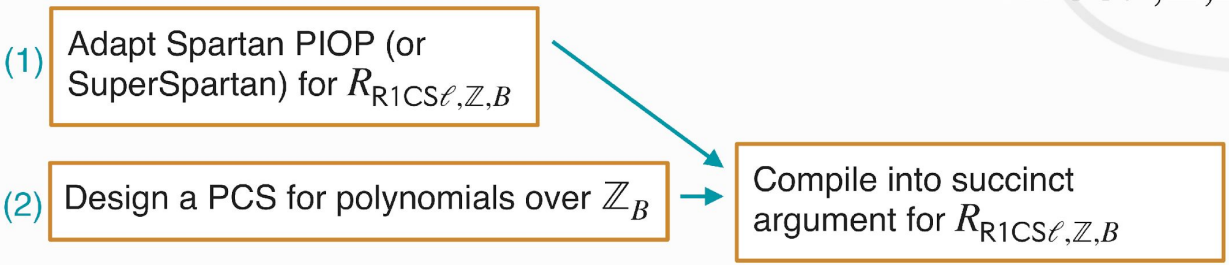
PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Design a PCS for polynomials over $\mathbb{Q}_B$

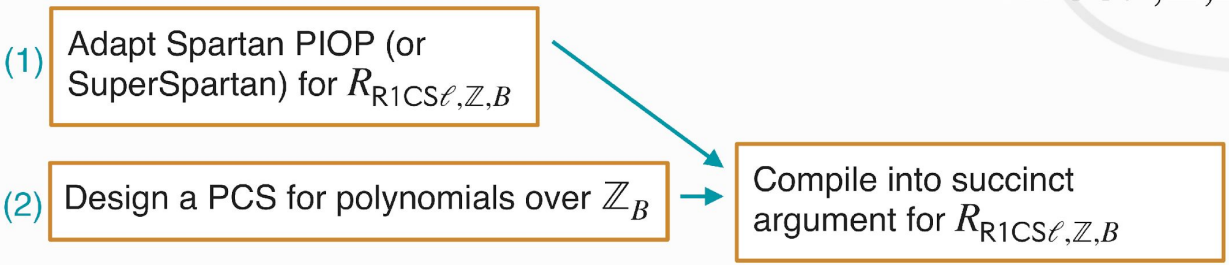
Soundness holds against P^* that use polys over $\mathbb{Q}_B$

→ Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$



Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$



We make the following key design choice:

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Soundness holds against P^* that use polys over $\mathbb{Q}_B$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Design a PCS for polynomials over $\mathbb{Q}_B$

Soundness holds against P^* that use polys over $\mathbb{Q}_B$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Design a PCS for polynomials over $\mathbb{Q}_B$

Soundness holds against P^* that use polys over $\mathbb{Q}_B$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Design a PCS for polynomials over $\mathbb{Q}_B$

Soundness holds against P^* that use polys over $\mathbb{Q}_B$

Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Designing a succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(1) Adapt Spartan PIOP (or SuperSpartan) for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

(2) Design a PCS for polynomials over $\mathbb{Z}_B$

→ Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Z},B}$

We make the following key design choice:

We work over $\mathbb{Q}_B$ instead of $\mathbb{Z}_B$.

New program:

PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Design a PCS for polynomials over $\mathbb{Q}_B$

Soundness holds against P^* that use polys over $\mathbb{Q}_B$

→ Compile into succinct argument for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Add a lookup argument $\mathbb{Q}_B$ to ensure witnesses are in $\mathbb{Z}_B$

The mod-PIOP technique over the rationals

The mod-PIOP technique over the rationals

Step 1 of our program: PIOP for $R_{\mathsf{R1CS}_{\ell,\mathbb{Q},B}}$

The mod-PIOP technique over the rationals

Step 1 of our program: PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

We want to use the mod-PIOP idea from Campanelli and Hall-Andersen

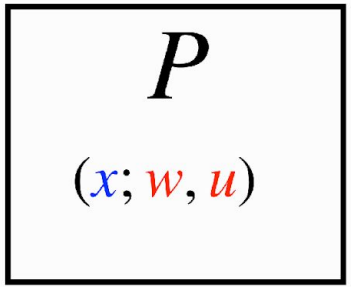
(Run PIOP modulo a random prime)

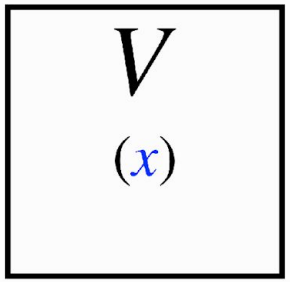
The mod-PIOP technique over the rationals

Step 1 of our program: PIOP for $R_{\text{R1CS}_{\ell,\mathbb{Q},B}}$

We want to use the mod-PIOP idea from Campanelli and Hall-Andersen

(Run PIOP modulo a random prime)



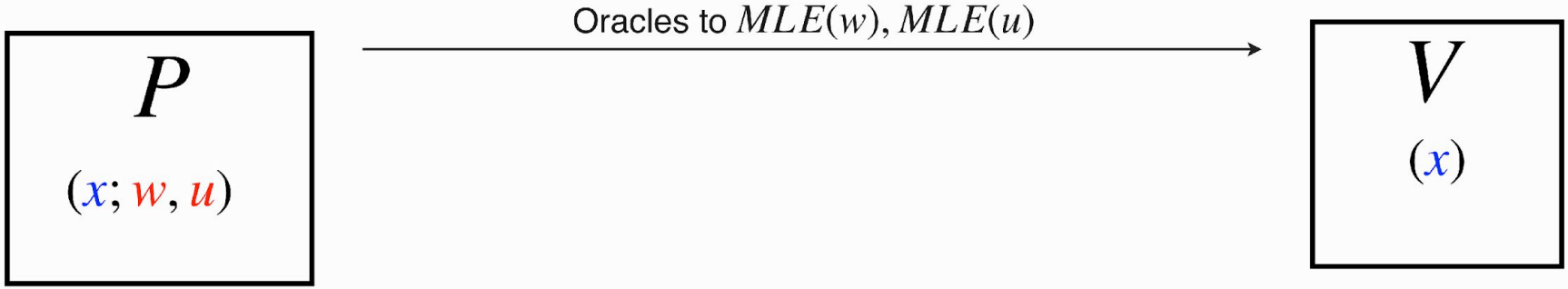


The mod-PIOP technique over the rationals

Step 1 of our program: PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

We want to use the mod-PIOP idea from Campanelli and Hall-Andersen

(Run PIOP modulo a random prime)

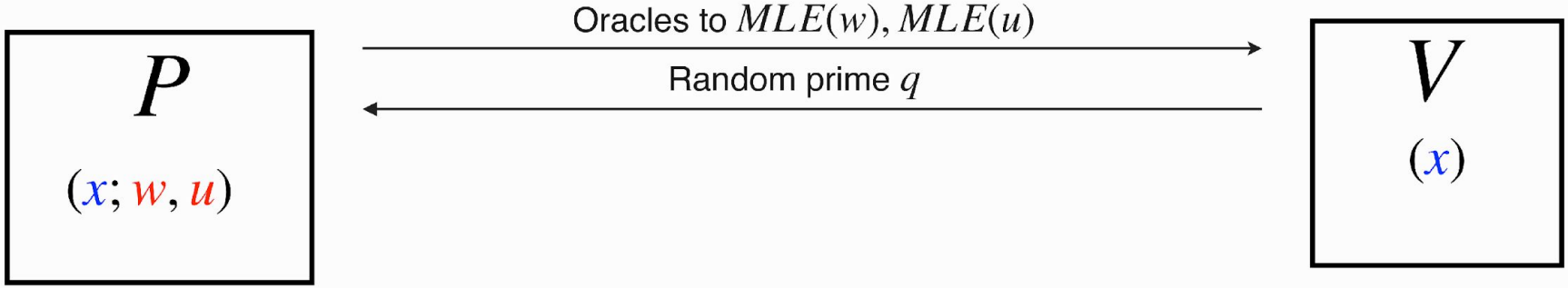


The mod-PIOP technique over the rationals

Step 1 of our program: PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

We want to use the mod-PIOP idea from Campanelli and Hall-Andersen

(Run PIOP modulo a random prime)

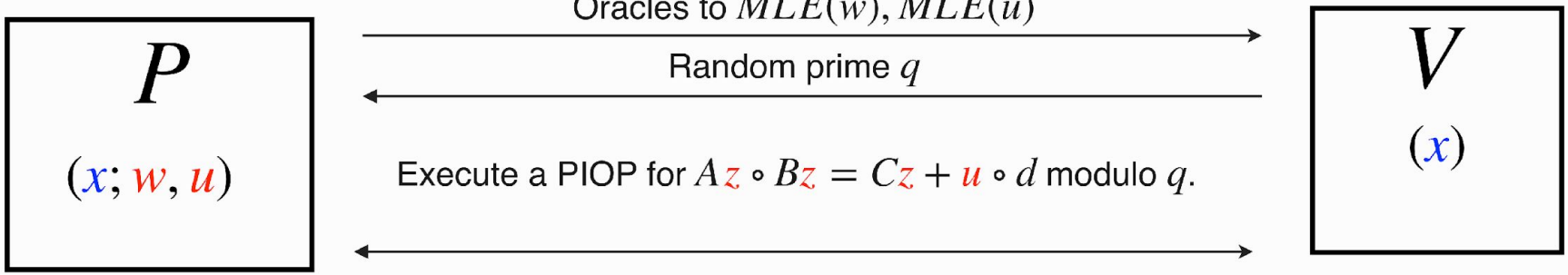


The mod-PIOP technique over the rationals

Step 1 of our program: PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

We want to use the mod-PIOP idea from Campanelli and Hall-Andersen

(Run PIOP modulo a random prime)

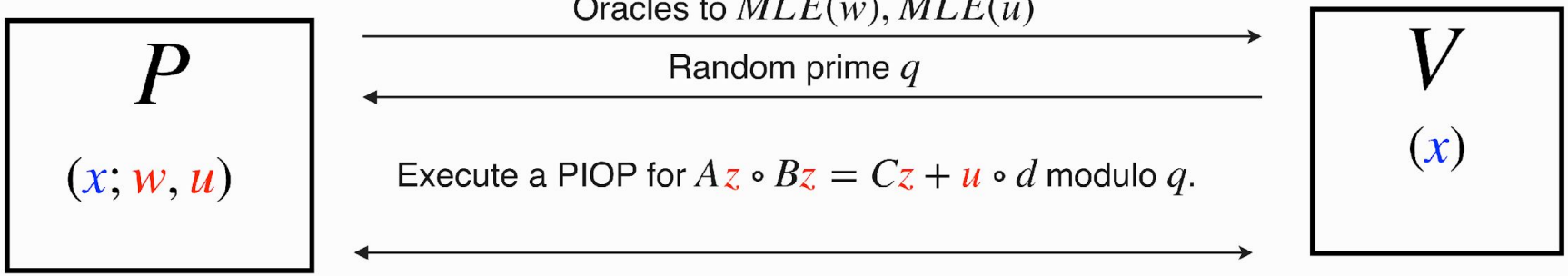


The mod-PIOP technique over the rationals

Step 1 of our program: PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

We want to use the mod-PIOP idea from Campanelli and Hall-Andersen

(Run PIOP modulo a random prime)



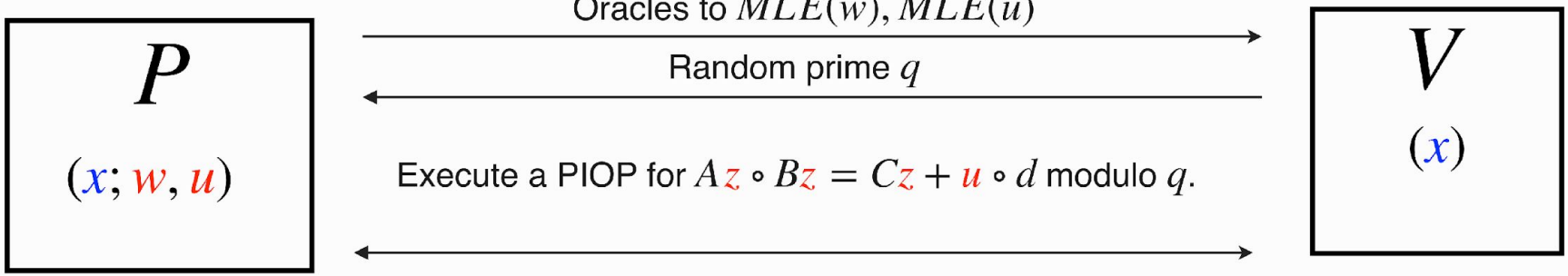
Issue: reduction modulo q is not well-defined in $\mathbb{Q}$

The mod-PIOP technique over the rationals

Step 1 of our program: PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

We want to use the mod-PIOP idea from Campanelli and Hall-Andersen

(Run PIOP modulo a random prime)



Issue: reduction modulo q is not well-defined in $\mathbb{Q}$

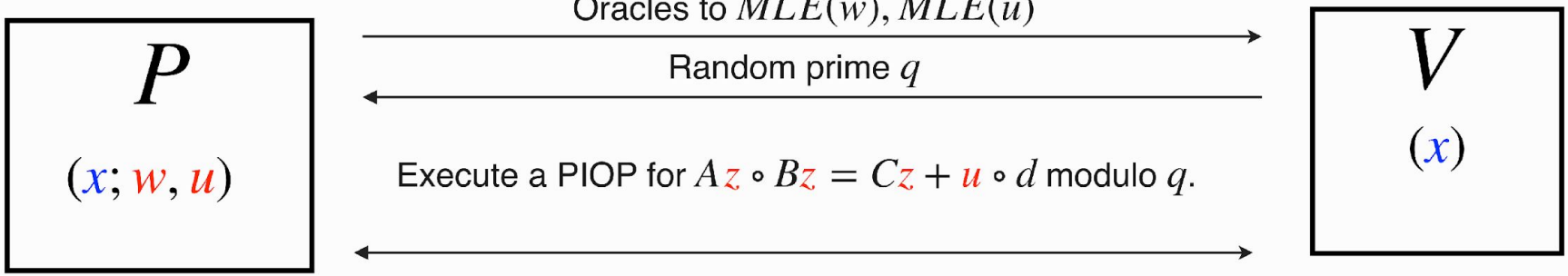
But it is for the local subring $\mathbb{Z}_{(q)} = \{a/b \in \mathbb{Q} \mid q \text{ does not divide } b\}$

The mod-PIOP technique over the rationals

Step 1 of our program: PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

We want to use the mod-PIOP idea from Campanelli and Hall-Andersen

(Run PIOP modulo a random prime)



Issue: reduction modulo q is not well-defined in $\mathbb{Q}$

But it is for the local subring $\mathbb{Z}_{(q)} = \{a/b \in \mathbb{Q} \mid q \text{ does not divide } b\}$

There is a ring homomorphism: $\phi_q : \mathbb{Z}_{(q)} \to \mathbb{F}_q, \quad a/b \mapsto a \cdot b^{-1} \bmod q$

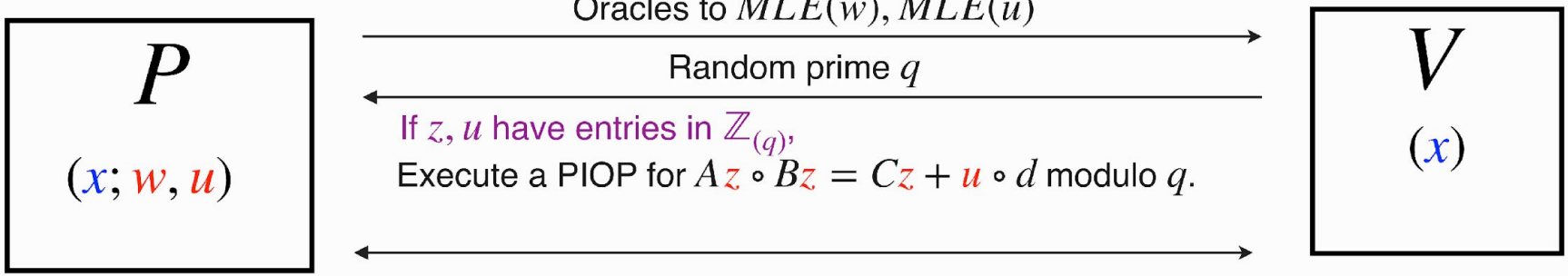
where b^{-1} denotes an inverse of b mod q.

The mod-PIOP technique over the rationals

Step 1 of our program: PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

We want to use the mod-PIOP idea from Campanelli and Hall-Andersen

(Run PIOP modulo a random prime)



Issue: reduction modulo q is not well-defined in $\mathbb{Q}$

But it is for the local subring $\mathbb{Z}_{(q)} = \{a/b \in \mathbb{Q} \mid q \text{ does not divide } b\}$

There is a ring homomorphism: $\phi_q : \mathbb{Z}_{(q)} \to \mathbb{F}_q, \quad a/b \mapsto a \cdot b^{-1} \bmod q$

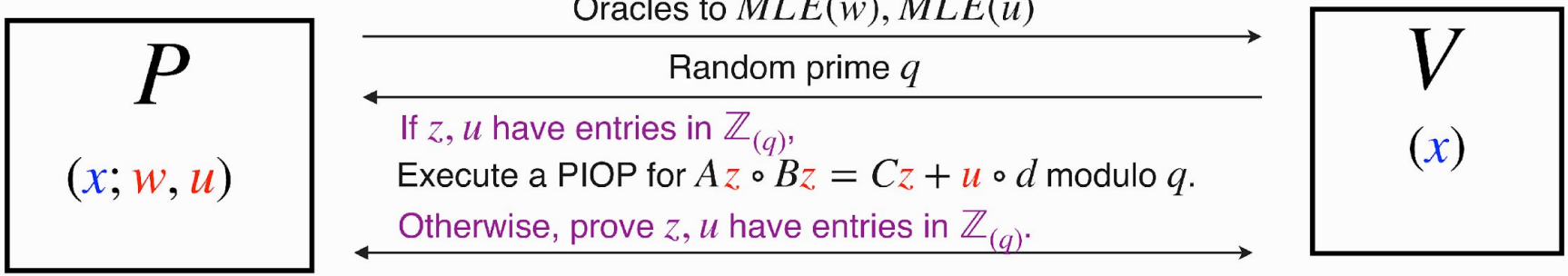
where b^{-1} denotes an inverse of b mod q.

The mod-PIOP technique over the rationals

Step 1 of our program: PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

We want to use the mod-PIOP idea from Campanelli and Hall-Andersen

(Run PIOP modulo a random prime)



Issue: reduction modulo q is not well-defined in $\mathbb{Q}$

But it is for the local subring $\mathbb{Z}_{(q)} = \{a/b \in \mathbb{Q} \mid q \text{ does not divide } b\}$

There is a ring homomorphism: $\phi_q : \mathbb{Z}_{(q)} \to \mathbb{F}_q, \quad a/b \mapsto a \cdot b^{-1} \bmod q$

where b^{-1} denotes an inverse of b mod q.

Where are we?

Where are we?

PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Where are we?

PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Soundness holds against P^* that use polys over $\mathbb{Q}_B$

NETHERMIND RESEARCH

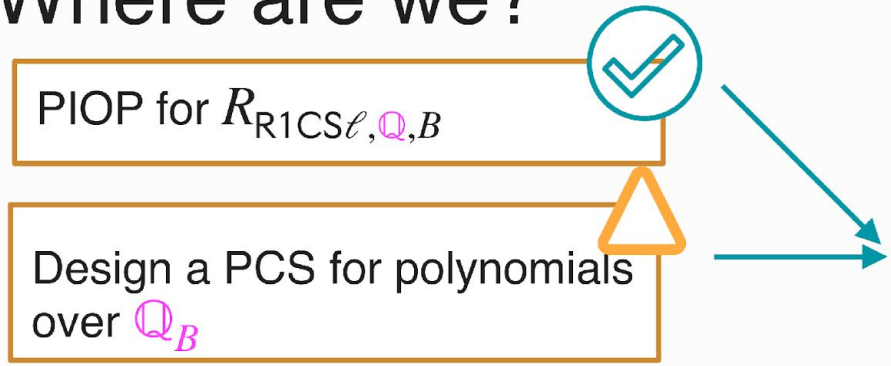
Where are we?

PIOP for $R_{\text{R1CS}\ell,\mathbb{Q},B}$

Design a PCS for polynomials over $\mathbb{Q}_B$

Soundness holds against P^* that use polys over $\mathbb{Q}_B$

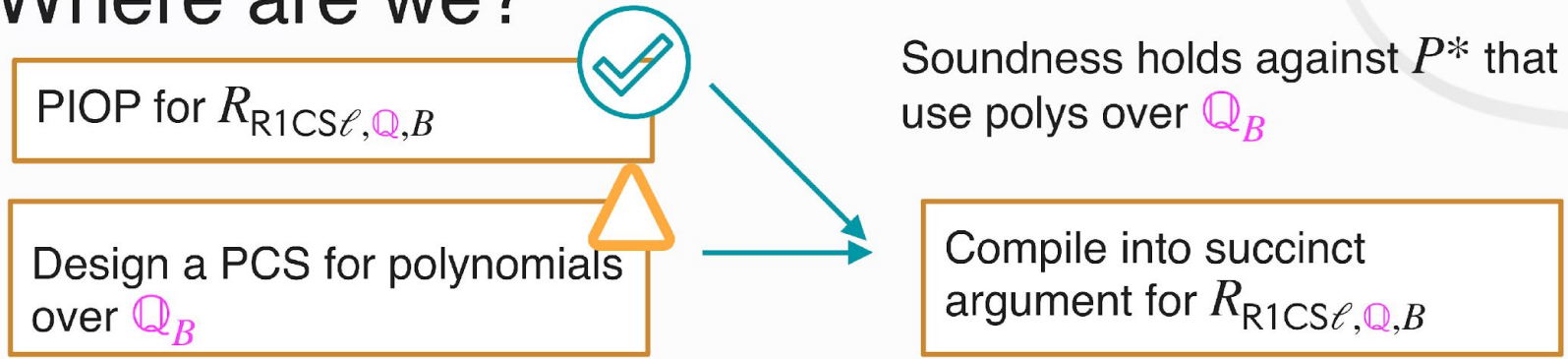
Where are we?



Soundness holds against P^* that use polys over $\mathbb{Q}_B$

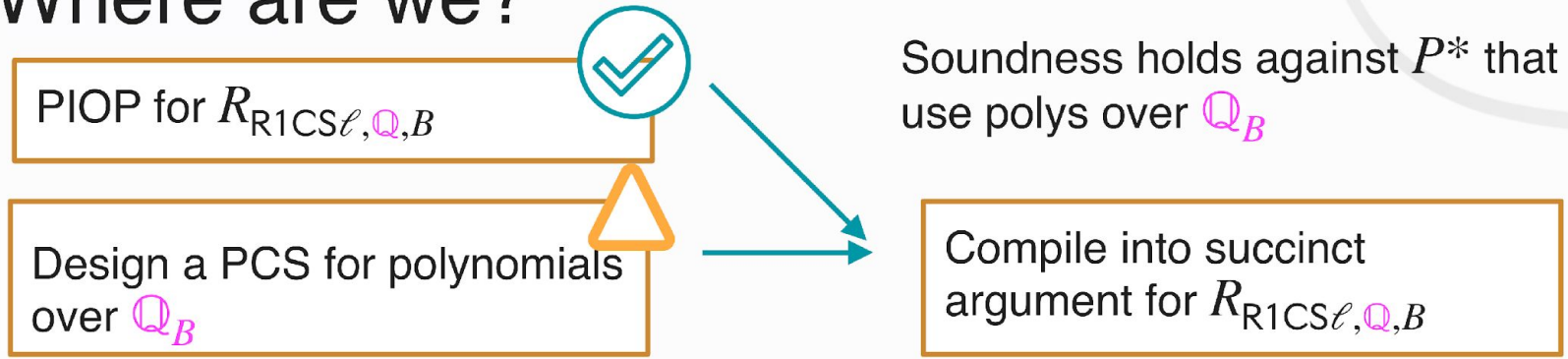
NETHERMIND RESEARCH

Where are we?



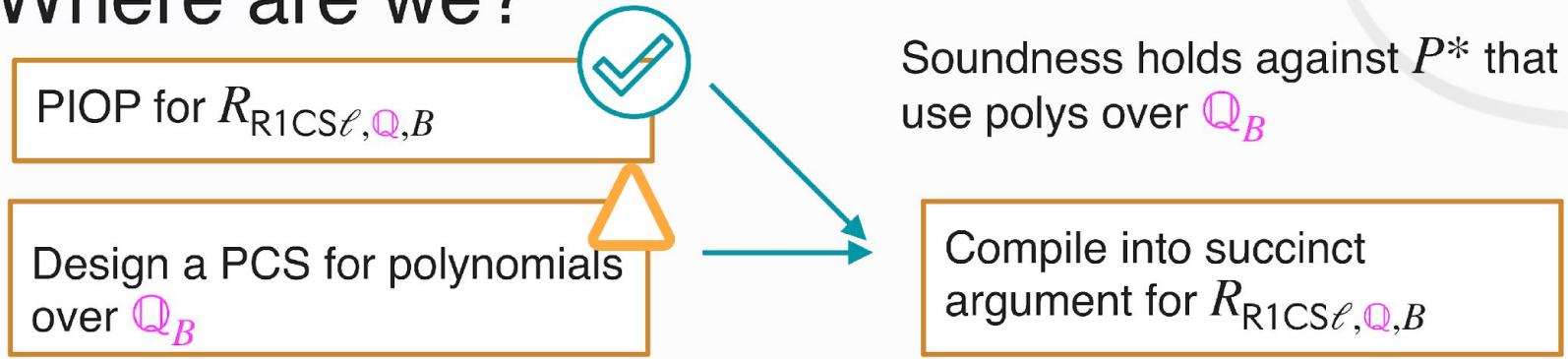
NETHERMIND RESEARCH

Where are we?



Next, we design Zip, a PCS for polynomials over $\mathbb{Q}_B$.

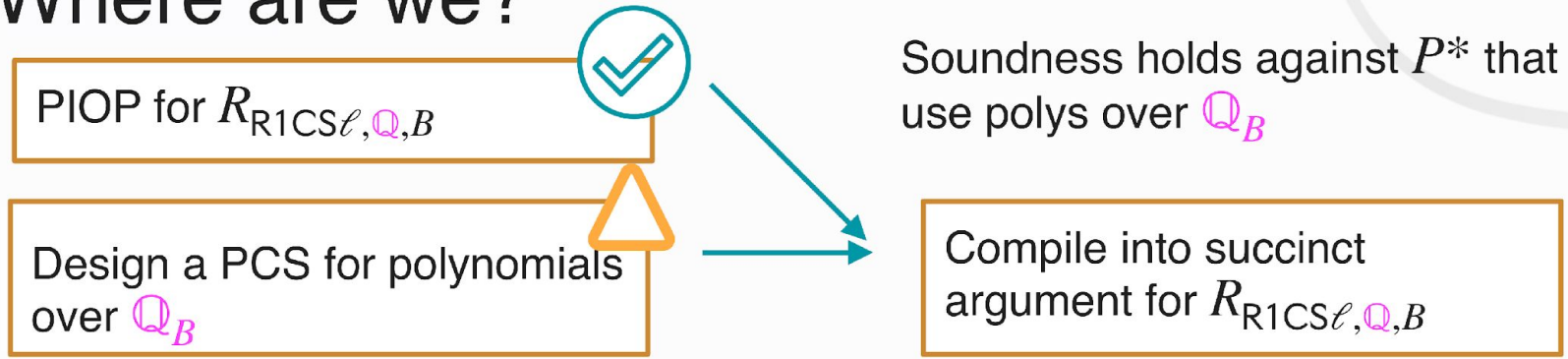
Where are we?



Next, we design Zip, a PCS for polynomials over $\mathbb{Q}_B$.

Zip is based on Brakedown and EA codes (to have encoding matrix with small entries).

Where are we?

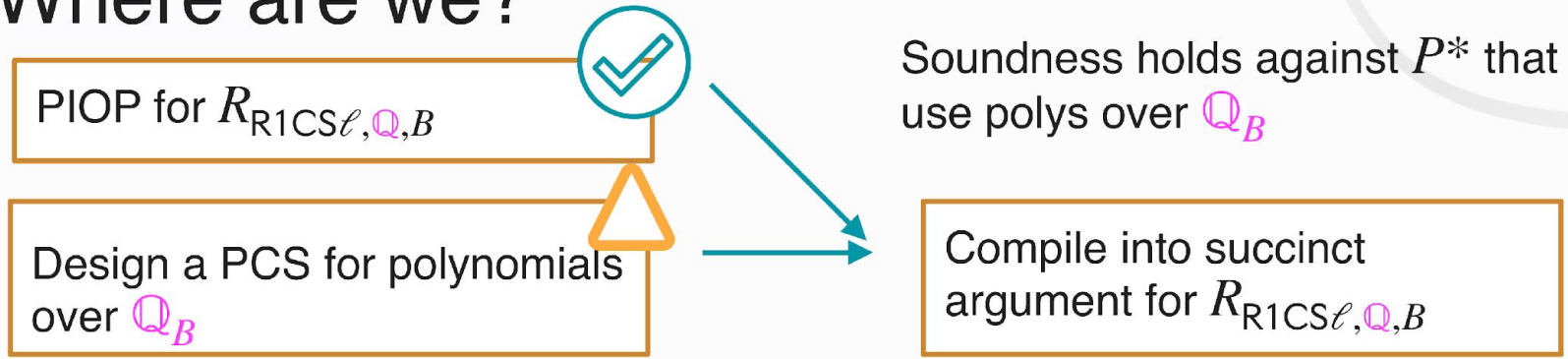


Next, we design Zip, a PCS for polynomials over $\mathbb{Q}_B$.

Zip is based on Brakedown and EA codes (to have encoding matrix with small entries).

Uses error correcting codes and hash functions

Where are we?



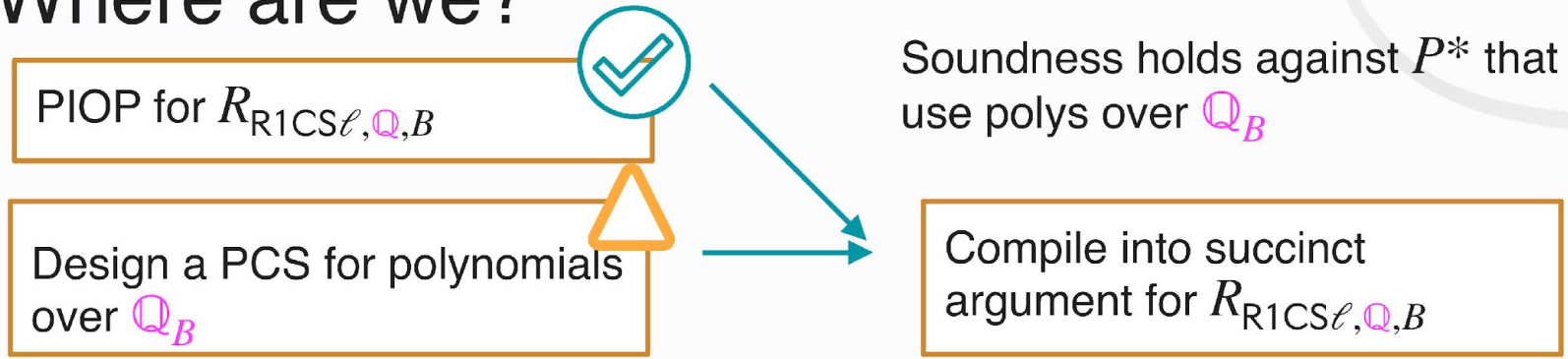
Next, we design Zip, a PCS for polynomials over $\mathbb{Q}_B$.

Zip is based on Brakedown and EA codes (to have encoding matrix with small entries).

Uses error correcting codes and hash functions

Most involved part of our work

Where are we?



Next, we design Zip, a PCS for polynomials over $\mathbb{Q}_B$.

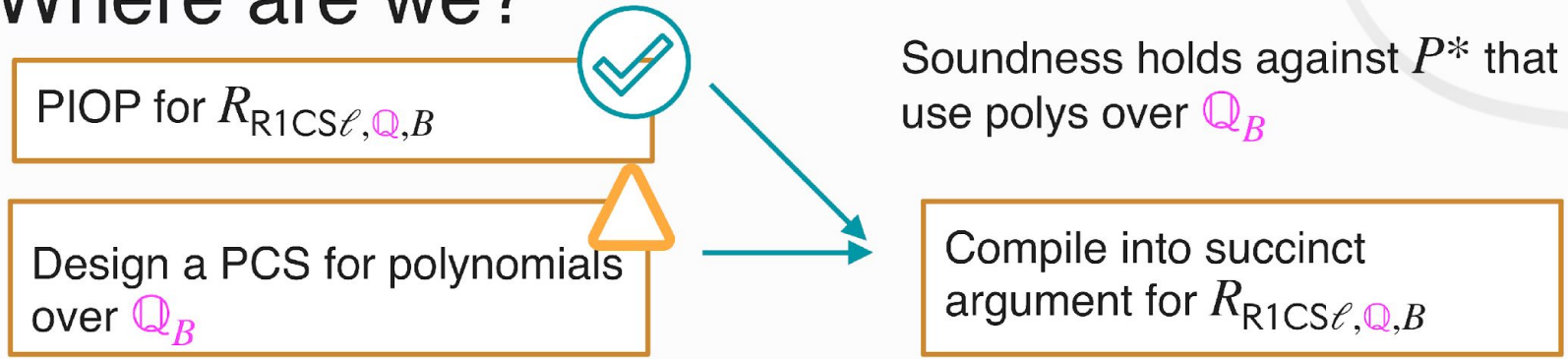
Zip is based on Brakedown and EA codes (to have encoding matrix with small entries).

Uses error correcting codes and hash functions

Most involved part of our work

Zip features both

Where are we?



Next, we design Zip, a PCS for polynomials over $\mathbb{Q}_B$.

Zip is based on Brakedown and EA codes (to have encoding matrix with small entries).

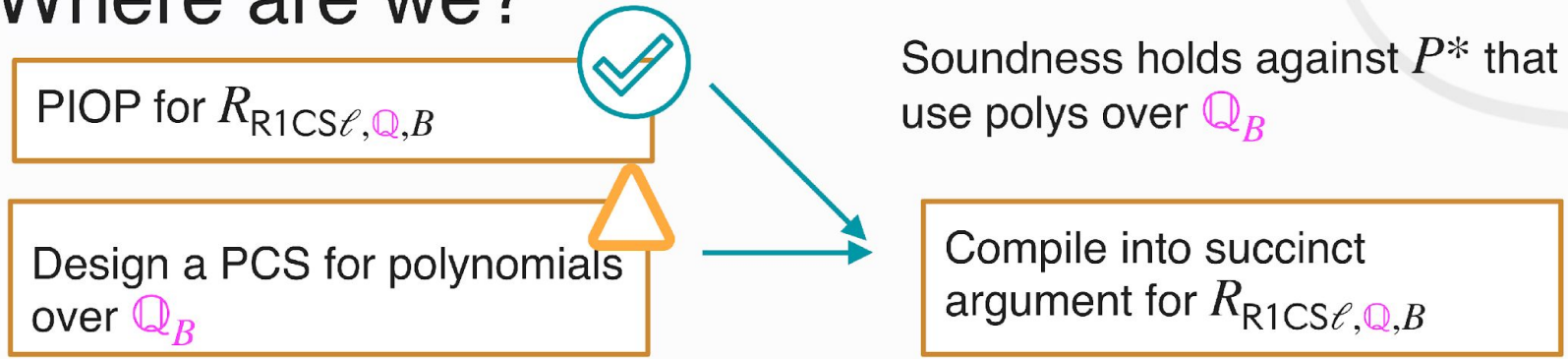
Uses error correcting codes and hash functions

Most involved part of our work

Zip features both

- IOP of proximity to a linear code

Where are we?



Next, we design Zip, a PCS for polynomials over $\mathbb{Q}_B$.

Zip is based on Brakedown and EA codes (to have encoding matrix with small entries).

Uses error correcting codes and hash functions

Most involved part of our work

Zip features both

- IOP of proximity to a linear code ← New!
- IOP of proximity to the integers

Zip

Zip

Soundness

Zip

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Zip

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Zip guarantees P committed to multilinear $f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$.

Zip

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Completeness

Zip guarantees P committed to multilinear $f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$.

Zip

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Completeness

To be used only for

$$f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu],\ B' < B$$

Zip guarantees P committed to multilinear $f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$.

Zip

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Completeness

To be used only for

$$f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu],\ B' < B$$

Zip guarantees P committed to multilinear $f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$.

But we expect the honest P to commit to multilinear $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$,

Zip

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Completeness

To be used only for

$$f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu],\ B' < B$$

Zip guarantees P committed to multilinear $f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$.

But we expect the honest P to commit to multilinear $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$,

Here B' is certain bound determined by B, μ, and other parameters.

Zip

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Completeness

To be used only for

$$f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu],\ B' < B$$

Zip guarantees P committed to multilinear $f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$.

But we expect the honest P to commit to multilinear $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$,

Here B' is certain bound determined by B, μ, and other parameters.

If P doesn't use $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$, completeness may fail.

Zip

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Completeness

To be used only for

$$f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu],\ B' < B$$

Zip guarantees P committed to multilinear $f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$.

But we expect the honest P to commit to multilinear $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$,

Here B' is certain bound determined by B, μ, and other parameters.

If P doesn't use $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$, completeness may fail.

Analogous to IOP of proximity (IOPP) to a code:

Zip

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Completeness

To be used only for

$$f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu],\ B' < B$$

Zip guarantees P committed to multilinear $f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$.

But we expect the honest P to commit to multilinear $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$,

Here B' is certain bound determined by B, μ, and other parameters.

If P doesn't use $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$, completeness may fail.

Analogous to IOP of proximity (IOPP) to a code:

An IOPP guarantees P committed to words close to codewords,

Zip

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Completeness

To be used only for

$$f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu],\ B' < B$$

Zip guarantees P committed to multilinear $f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$.

But we expect the honest P to commit to multilinear $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$,

Here B' is certain bound determined by B, μ, and other parameters.

If P doesn't use $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$, completeness may fail.

Analogous to IOP of proximity (IOPP) to a code:

An IOPP guarantees P committed to words close to codewords,

But completeness is only guaranteed if P actually used codewords

Zip

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Completeness

To be used only for

$$f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu],\ B' < B$$

Zip guarantees P committed to multilinear $f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$.

But we expect the honest P to commit to multilinear $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$,

Here B' is certain bound determined by B, μ, and other parameters.

If P doesn't use $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$, completeness may fail.

Analogous to IOP of proximity (IOPP) to a code:

An IOPP guarantees P committed to words close to codewords,

But completeness is only guaranteed if P actually used codewords

In our use cases, honest P always uses integral polys.

Zip

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Completeness

To be used only for

$$f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu],\ B' < B$$

Zip guarantees P committed to multilinear $f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$.

But we expect the honest P to commit to multilinear $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$,

Here B' is certain bound determined by B, μ, and other parameters.

If P doesn't use $f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu]$, completeness may fail.

Analogous to IOP of proximity (IOPP) to a code:

An IOPP guarantees P committed to words close to codewords,

But completeness is only guaranteed if P actually used codewords

In our use cases, honest P always uses integral polys.

(We can extend Zip to enable completeness for $f \in \mathbb{Q}_{B'}[X_1, \ldots, X_\mu]$.)

Zip as an IOP of proximity to $\mathbb{Z}$

Soundness

Guaranteed: P committed to

$$f \in \mathbb{Q}_B[X_1, \ldots, X_\mu]$$

Completeness

To be used only for

$$f \in \mathbb{Z}_{B'}[X_1, \ldots, X_\mu],\ B' < B$$

Thanks