Non-Native Arithmetic via CRT Codes

Devcon 7

Liam Eagen
Alpen Labs

November 14, 2024

Recap SNARKs

- Succinct Non-interactive ARgument of Knowledge

Recap SNARKs

- Succinct Non-interactive ARgument of Knowledge
- Allow a Prover to convince a Verifier they *know* some witness

Recap SNARKs

- Succinct Non-interactive ARgument of Knowledge
- Allow a Prover to convince a Verifier they *know* some witness
- Witness satisfies some relation $C(x, w) = 1$

Recap SNARKs

- Succinct Non-interactive ARgument of Knowledge
- Allow a Prover to convince a Verifier they *know* some witness
- Witness satisfies some relation $C(x, w) = 1$
- For example, x chain state, w transaction, $C(x, w) = 1$ iff valid

Recap SNARKs

- Succinct Non-interactive ARgument of Knowledge
- Allow a Prover to convince a Verifier they *know* some witness
- Witness satisfies some relation $C(x, w) = 1$
- For example, x chain state, w transaction, $C(x, w) = 1$ iff valid
- Relation defined via a circuit over some *finite field* $\mathbb{F}_p$

Recap SNARKs

- Succinct Non-interactive ARgument of Knowledge
- Allow a Prover to convince a Verifier they *know* some witness
- Witness satisfies some relation $C(x, w) = 1$
- For example, x chain state, w transaction, $C(x, w) = 1$ iff valid
- Relation defined via a circuit over some *finite field* $\mathbb{F}_p$
- Field is sometimes a “free” parameter (e.g. FRI), sometimes fixed (e.g. KZG)

Non-native Arithmetic

- What if the relation is over the wrong field?

Non-native Arithmetic

- What if the relation is over the wrong field?
- For example, w is a secp256k1 signature, but $\mathbb{F}_p$ is BN254?

Non-native Arithmetic

- What if the relation is over the wrong field?
- For example, w is a secp256k1 signature, but $\mathbb{F}_p$ is BN254?
- Secp signatures defined over $\mathbb{F}_r \neq \mathbb{F}_p$

Non-native Arithmetic

- What if the relation is over the wrong field?
- For example, w is a secp256k1 signature, but $\mathbb{F}_p$ is BN254?
- Secp signatures defined over $\mathbb{F}_r \neq \mathbb{F}_p$
- Need to *simulate* $\mathbb{F}_r$ arithmetic in $\mathbb{F}_p$

Non-native Arithmetic

- What if the relation is over the wrong field?
- For example, w is a secp256k1 signature, but $\mathbb{F}_p$ is BN254?
- Secp signatures defined over $\mathbb{F}_r \neq \mathbb{F}_p$
- Need to *simulate* $\mathbb{F}_r$ arithmetic in $\mathbb{F}_p$

  Very simple example: let $p = 5$ and $r = 7$

Non-native Arithmetic

- What if the relation is over the wrong field?
- For example, w is a secp256k1 signature, but $\mathbb{F}_p$ is BN254?
- Secp signatures defined over $\mathbb{F}_r \neq \mathbb{F}_p$
- Need to *simulate* $\mathbb{F}_r$ arithmetic in $\mathbb{F}_p$
- Very simple example: let $p = 5$ and $r = 7$
  - Want to check $4 \times 5 = 20 \equiv 6 \mod 7$ in $\mathbb{F}_r$

Non-native Arithmetic

- What if the relation is over the wrong field?
- For example, w is a secp256k1 signature, but $\mathbb{F}_p$ is BN254?
- Secp signatures defined over $\mathbb{F}_r \neq \mathbb{F}_p$
- Need to *simulate* $\mathbb{F}_r$ arithmetic in $\mathbb{F}_p$
- Very simple example: let $p = 5$ and $r = 7$
  - Want to check $4 \times 5 = 20 \equiv 6 \mod 7$ in $\mathbb{F}_r$
  - Naive approach fails $4 \times 5 = 20 \equiv 0 \not\equiv 6 \mod 5$

Non-native Arithmetic

Is it possible?

- What can we do?

Non-native Arithmetic

Is it possible?

- What can we do?
- Arithmetic circuits are NP-complete, so it is possible

Non-native Arithmetic

Is it possible?

- What can we do?
- Arithmetic circuits are NP-complete, so it is possible
- For example, encode as binary digits

Non-native Arithmetic

Is it possible?

- What can we do?
- Arithmetic circuits are NP-complete, so it is possible
- For example, encode as binary digits
    - In any field we can check $x = 0, 1$ via $x^2 - x = 0$

Non-native Arithmetic

Is it possible?

- What can we do?
- Arithmetic circuits are NP-complete, so it is possible
- For example, encode as binary digits
  - In any field we can check $x = 0, 1$ via $x^2 - x = 0$
  - Can check bitwise operations

Non-native Arithmetic

Is it possible?

- What can we do?
- Arithmetic circuits are NP-complete, so it is possible
- For example, encode as binary digits
  - In any field we can check $x = 0, 1$ via $x^2 - x = 0$
  - Can check bitwise operations
  - Can we do this more efficiently?

Reduction to Integer Relations

- If we can check (bounded) *integer* relations, can check *modular* relations

Reduction to Integer Relations

- If we can check (bounded) *integer* relations, can check *modular* relations
- $ab \equiv c \mod r \iff \exists q \in \mathbb{Z} : ab - c = rq$

Reduction to Integer Relations

- If we can check (bounded) *integer* relations, can check *modular* relations
- $ab \equiv c \mod r \iff \exists q \in \mathbb{Z} : ab - c = rq$
- Thus, sufficient to check integer relations

Reduction to Integer Relations

- If we can check (bounded) *integer* relations, can check *modular* relations
- $ab \equiv c \mod r \iff \exists\, q \in \mathbb{Z} : ab - c = rq$
- Thus, sufficient to check integer relations
- In some cases, this is sufficient to simulate $\mathbb{F}'$ directly

Reduction to Integer Relations

- If we can check (bounded) *integer* relations, can check *modular* relations
- $ab \equiv c \mod r \iff \exists\, q \in \mathbb{Z} : ab - c = rq$
- Thus, sufficient to check integer relations
- In some cases, this is sufficient to simulate $\mathbb{F}'$ directly
- "Small" elements of $\mathbb{F}$ behave like $\mathbb{Z}$

Reduction to Integer Relations

- If we can check (bounded) *integer* relations, can check *modular* relations
- $ab \equiv c \mod r \iff \exists q \in \mathbb{Z} : ab - c = rq$
- Thus, sufficient to check integer relations
- In some cases, this is sufficient to simulate $\mathbb{F}'$ directly
- "Small" elements of $\mathbb{F}$ behave like $\mathbb{Z}$

If $|ab - c - rq| < p$ and $ab - c - rq = 0 \mod p$ then $ab = c \mod r$

Reduction to Integer Relations

- If we can check (bounded) *integer* relations, can check *modular* relations
- $ab \equiv c \mod r \iff \exists q \in \mathbb{Z} : ab - c = rq$
- Thus, sufficient to check integer relations
- In some cases, this is sufficient to simulate $\mathbb{F}'$ directly
- "Small" elements of $\mathbb{F}$ behave like $\mathbb{Z}$
- If $|ab - c - rq| < p$ and $ab - c - rq = 0 \mod p$ then $ab = c \mod r$
- If $r < \sqrt{p/2}$, sufficient for $|a|, |b|, |c|, |q| < \sqrt{p/2}$

Residue Number Systems

- What if $r > \sqrt{p}$? Or even $r > p$?

Residue Number Systems

- What if $r > \sqrt{p}$? Or even $r > p$?
- Encodings of "mod r" values into $\mathbb{F}_p$ no longer behave like $\mathbb{Z}$

Residue Number Systems

- What if $r > \sqrt{p}$? Or even $r > p$?
- Encodings of "mod r" values into $\mathbb{F}_p$ no longer behave like $\mathbb{Z}$
- Might not even be possible at all (when $r > p$)

Residue Number Systems

- What if $r > \sqrt{p}$? Or even $r > p$?
- Encodings of "mod r" values into $\mathbb{F}_p$ no longer behave like $\mathbb{Z}$
- Might not even be possible at all (when $r > p$)

Solution: use the Chinese Remainder Theorem (CRT)

Residue Number Systems

- What if $r > \sqrt{p}$? Or even $r > p$?
- Encodings of "mod r" values into $\mathbb{F}_p$ no longer behave like $\mathbb{Z}$
- Might not even be possible at all (when $r > p$)
- Solution: use the Chinese Remainder Theorem (CRT)

  Says $\mathbb{Z}/(ab)\mathbb{Z} \simeq \mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$ if $\gcd(a, b) = 1$

Residue Number Systems

- What if $r > \sqrt{p}$? Or even $r > p$?
- Encodings of "mod r" values into $\mathbb{F}_p$ no longer behave like $\mathbb{Z}$
- Might not even be possible at all (when $r > p$)
- Solution: use the Chinese Remainder Theorem (CRT)
- Says $\mathbb{Z}/(ab)\mathbb{Z} \cong \mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$ if $\gcd(a, b) = 1$
- Can represent a number via a set of residues $x = x_i \bmod p_i$

Residue Number Systems

- What if $r > \sqrt{p}$? Or even $r > p$?
- Encodings of "mod r" values into $\mathbb{F}_p$ no longer behave like $\mathbb{Z}$
- Might not even be possible at all (when $r > p$)
- Solution: use the Chinese Remainder Theorem (CRT)
- Says $\mathbb{Z}/(ab)\mathbb{Z} \cong \mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$ if $\gcd(a, b) = 1$
- Can represent a number via a set of residues $x = x_i \bmod p_i$
- Use CRT to uniquely recover x

Residue Number Systems

- What if $r > \sqrt{p}$? Or even $r > p$?
- Encodings of "mod r" values into $\mathbb{F}_p$ no longer behave like $\mathbb{Z}$
- Might not even be possible at all (when $r > p$)
- Solution: use the Chinese Remainder Theorem (CRT)
- Says $\mathbb{Z}/(ab)\mathbb{Z} \cong \mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$ if $\gcd(a, b) = 1$
- Can represent a number via a set of residues $x = x_i \bmod p_i$
- Use CRT to uniquely recover x
- Now, just need x small compared to $M = \prod_i p_i$

Reed-Solomon Codes

- Recall Reed-Solomon (RS) codes

Reed-Solomon Codes

- Recall Reed-Solomon (RS) codes
- Treat k vector as a degree $k - 1$ polynomial $f(X)$

Reed-Solomon Codes

- Recall Reed-Solomon (RS) codes
- Treat k vector as a degree $k-1$ polynomial $f(X)$
- Encode by evaluating at $n > k$ distinct points

Reed-Solomon Codes

- Recall Reed-Solomon (RS) codes
- Treat k vector as a degree $k - 1$ polynomial $f(X)$
- Encode by evaluating at $n > k$ distinct points
- Given these n points, can recover $f(X)$

Reed-Solomon Codes

- Recall Reed-Solomon (RS) codes
- Treat k vector as a degree $k - 1$ polynomial $f(X)$
- Encode by evaluating at $n > k$ distinct points Given
- these n points, can recover $f(X)$
- Can even recover $f(X)$ if some of the points are wrong

CRT Codes

- Evaluating a polynomial $f(r_i)$ is equivalent to reducing $f(X) = f(r_i) \bmod X - r_i$

CRT Codes

- Evaluating a polynomial $f(r_i)$ is equivalent to reducing $f(X) = f(r_i)$ mod $X - r_i$
- This lets us generalize RS codes to other ideal domains

CRT Codes

- Evaluating a polynomial $f(r_i)$ is equivalent to reducing $f(X) = f(r_i) \bmod X - r_i$
- This lets us generalize RS codes to other ideal domains
- For example, Algebraic Geometry codes

CRT Codes

- Evaluating a polynomial $f(r_i)$ is equivalent to reducing $f(X) = f(r_i)$ mod $X - r_i$
- This lets us generalize RS codes to other ideal domains
- For example, Algebraic Geometry codes
- Also CRT codes

CRT Codes

- Evaluating a polynomial $f(r_i)$ is equivalent to reducing $f(X) = f(r_i) \mod X - r_i$
- This lets us generalize RS codes to other ideal domains
- For example, Algebraic Geometry codes
- Also CRT codes

Treat our message as a bounded integer $x < M$

CRT Codes

- Evaluating a polynomial $f(r_i)$ is equivalent to reducing $f(X) = f(r_i) \bmod X - r_i$
- This lets us generalize RS codes to other ideal domains
- For example, Algebraic Geometry codes
- Also CRT codes
- Treat our message as a bounded integer $x < M$
- Encode by reducing modulo many small p_i

Protocol Sketch

- Suppose we want to verify some non-native arithmetic
- That is $f(x_1, \dots, x_k) = 0$ over $\mathbb{Z}$ where $\max_i f_i = d$ and $|x_j| < B$
  1. Fix some $p_1, \dots, p_\ell$ where $M = \prod_a p_a > B^d$
  2. Commit to $x_j = y_{j,a} mod p_a$
  3. Commit to $q_{i,a}$ such that $f_i(y_{1,a}, \dots, y_{k,a}) = p_a q_{i,a}$
  4. Prove each $(y_{i,a})_{a=1}^{\ell}$ corresponds to $|x_i| < B$
  5. Choose a random subset of primes and test
- Assuming the primes similar size, easy to compute success probability of dishonest prover
- Spiritually similar to STARKs like FRI, Ligero, etc.

Protocol Sketch

- Suppose we want to verify some non-native arithmetic
- That is $f(x_1, \ldots, x_k) = 0$ over $\mathbb{Z}$ where $\max_i f_i = d$ and $|x_j| < B$
  1. Fix some $p_1, \ldots, p_\ell$ where $M = \prod_a p_a > B^{d}$
  2. Commit to $x_j = y_{j,a} \bmod p_a$
  3. Commit to $q_{i,a}$ such that $f_i(y_{1,a}, \ldots, y_{k,a}) = p_a q_{i,a}$
  4. Prove each $(y_{i,a})_{a=1}^{\ell}$ corresponds to $|x_i| < B$

Protocol Sketch

- Suppose we want to verify some non-native arithmetic
- That is $f_i(x_1, \ldots, x_k) = 0$ over $\mathbb{Z}$ where $\max_i f_i = d$ and $|x_j| < B$
  1. Fix some $p_1, \ldots, p_\ell$ where $M = \prod_a p_a > B^d$
  2. Commit to $x_j = y_{j,a} \mod p_a$
  3. Commit to $q_{i,a}$ such that $f_i(y_{1,a}, \ldots, y_{k,a}) = p_a q_{i,a}$
  4. Prove each $(y_{j,a})_{a=1}^{\ell}$ corresponds to $|x_j| < B$
  5. Choose a random subset of primes and test

Protocol Sketch

- Suppose we want to verify some non-native arithmetic
- That is $f(x_1, \ldots, x_k) = 0$ over $\mathbb{Z}$ where $\max_i f_i = d$ and $|x_j| < B$
  1. Fix some $p_1, \ldots, p_\ell$ where $M = \prod_a p_a > B^{d}$
  2. Commit to $x_j = y_{j,a} \mod p_a$
  3. Commit to $q_{i,a}$ such that $f_i(y_{1,a}, \ldots, y_{k,a}) = p_a q_{i,a}$
  4. Prove each $(y_{j,a})_{a=1}^{\ell}$ corresponds to $|x_j| < B$
  5. Choose a random subset of primes and test
- Assuming the primes similar size, easy to compute success probability of dishonest prover

Protocol Sketch

- Suppose we want to verify some non-native arithmetic
- That is $f(x_1, \ldots, x_k) = 0$ over $\mathbb{Z}$ where $\max_i f_i = d$ and $|x_j| < B$
  1. Fix some $p_1, \ldots, p_\ell$ where $M = \prod_a p_a > B^d$
  2. Commit to $x_j = y_{j,a} \mod p_a$
  3. Commit to $q_{i,a}$ such that $f_i(y_{1,a}, \ldots, y_{k,a}) = p_a q_{i,a}$
  4. Prove each $(y_{j,a})_{a=1}^{\ell}$ corresponds to $|x_j| < B$
  5. Choose a random subset of primes and test
- Assuming the primes similar size, easy to compute success probability of dishonest prover
- Spiritually similar to STARKs like FRI, Ligero, etc.

Ongoing Work: Proximity Gaps

- Step 4 of the protocol tests each $(y_{i,a})$ is a valid codeword

Ongoing Work: Proximity Gaps

- Step 4 of the protocol tests each $(y_{i,a})$ is a valid codeword
- STARKs used to do this, now use a batched proximity test

Ongoing Work: Proximity Gaps

- Step 4 of the protocol tests each $(y_{i,a})$ is a valid codeword
- STARKs used to do this, now use a batched proximity test
- Remarkably, if a *random linear combination* (RLC) of codewords is *close* to a code word, all the combined codewords are *also close*

Ongoing Work: Proximity Gaps

- Step 4 of the protocol tests each $(y_{i,a})$ is a valid codeword
- STARKs used to do this, now use a batched proximity test
- Remarkably, if a *random linear combination* (RLC) of codewords is *close* to a code word, all the combined codewords are *also close*
- Idea: can we take a RLC of the x_j?

Ongoing Work: Proximity Gaps

- Step 4 of the protocol tests each $(y_{i,a})$ is a valid codeword
- STARKs used to do this, now use a batched proximity test
- Remarkably, if a *random linear combination* (RLC) of codewords is *close* to a code word, all the combined codewords are *also close*
- Idea: can we take a RLC of the x_j?
- Yes! but more complex

What I Haven't Discussed

- Subtleties of CRT code distance since p_i different sizes

What I Haven't Discussed

- Subtleties of CRT code distance since p_i different sizes
- Decoding to rationals of bounded height rather than integers

What I Haven't Discussed

- Subtleties of CRT code distance since p_i different sizes
- Decoding to rationals of bounded height rather than integers
- Using number fields instead of rationals

What I Haven't Discussed

- Subtleties of CRT code distance since p_i different sizes
- Decoding to rationals of bounded height rather than integers
- Using number fields instead of rationals
- Technicalities related to this like size, etc.

What I Haven't Discussed

- Subtleties of CRT code distance since p_i different sizes
- Decoding to rationals of bounded height rather than integers
- Using number fields instead of rationals
- Technicalities related to this like size, etc.
- Applications of techniques to STARKs over small fields (i.e. without extensions)

Thanks!