EECS 498/CSE 598: Zero-Knowledge Proofs
Winter 2026
Lecture 6

$\sum$ ✅

Paul Grubbs

paulgrub@umich.edu

Beyster 4709

Agenda for this lecture

- Announcements
- What is a “backend”?
- Interactive proofs
- The sumcheck protocol
- Analyzing sumcheck
- (If time) Applications of sumcheck

Agenda for this lecture

- Announcements
- What is a “backend”?
- Interactive proofs
- The sumcheck protocol
- Analyzing sumcheck
- (If time) Applications of sumcheck

Announcements

- Project 1 is online
- Autograder is working
- P2 release delayed → Friday

  due date pushed back
- P1 grace period

Agenda for this lecture

- Announcements
- What is a “backend”?
- Interactive proofs
- The sumcheck protocol
- Analyzing sumcheck
- (If time) Applications of sumcheck

“Backends”

Question: How does ZKP software get used to solve a real problem?

First, use frontend component to:

1. Write human-readable program in ZK programming language.
   - captures requirements of problem in typical programming abstractions
   - Program P takes public input x and private witness w, outputs 0/1.
2. Run a compiler on P to transform it into an algebraic representation C.
   - Also need to generate (or have programmer write) an auxiliary program E that does “witness extension”: take x,w inputs for P, turn them into x’,w’ inputs for C. Need property that for all x,w, $P(x, w)=1 \Leftrightarrow C(x', w')=1$

The frontend’s output is P, C, and E.

“Backends”

Question: How does ZKP software get used to solve a real problem?

First, use frontend component to:

1. Write human-readable program in ZK programming language.
   - captures requirements of problem in typical programming abstractions
   - Program P takes public input x and private witness w, outputs 0/1.
2. Run a compiler on P to transform it into an algebraic representation C.
   - Also need to generate (or have programmer write) an auxiliary program E that does “witness extension”: take x,w inputs for P, turn them into x’,w’ inputs for C. Need property that for all x,w, P(x, w)=1 ⇔ C(x’, w’)=1

<u>The frontend’s output is P, C, and E.</u>

Given concrete x,w so that P(x, w)=1, use <u>backend</u> to generate and verifies proofs about P, x, w.

Implements interfaces:

$$Verify(pp, P, C, E, x, \pi) \rightarrow \{0,1\} \quad Prove(pp, P, C, E, x, w) \rightarrow \pi$$

The program $Prove$ first runs E on x,w to get x’,w’, then runs a ZK prover to generate a proof that C,x’ has a corresponding w’.

The program $Verify$ first uses E to turn x into x’, then verifies proof π using ZK verifier.

How to build a backend

Question: How is the backend protocol built?

1. Assume we already have C, x’, w’ and C is in “convenient” algebraic form like R1CS
2. Express C, x’, w’ as polynomials (via eg univariate or multilinear extensions)
3. Come up with a relationship between, or property of, these polynomials such that:
    - property holds if and only if C(x’, w’) = 1

How to build a backend

Question: How is the backend protocol built?

1. Assume we already have C, x’, w’ and C is in “convenient” algebraic form like R1CS
2. Express C, x’, w’ as polynomials (via eg univariate or multilinear extensions)
3. Come up with a relationship between, or property of, these polynomials such that:
   - property holds if and only if C(x’, w’) = 1
4. Design protocol between prover and verifier in an “ideal” model
   - Interactive oracle proofs (IOPs): like IPs but <u>force</u> honest prover behavior
   - Protocol lets prover convince the verifier these particular polynomials have the property hold
5. Use cryptography to “compile” this IOP to a real protocol:
   1. Compile IOP to interactive argument of knowledge via polynomial commitments
   2. Remove interaction using fiat-shamir transform
   3. Hide witness (ZK) by blinding “leaky” parts of proof with random data

ZKSNARK

How to build a backend

Question: How is the backend protocol built?

1. Assume we already have C, x’, w’ and C is in “convenient” algebraic form like R1CS
2. Express C, x’, w’ as polynomials (via eg univariate or multilinear extensions)
3. Come up with a relationship between, or property of, these polynomials such that:
   - property holds if and only if C(x’, w’) = 1
4. Design protocol between prover and verifier in an “ideal” model
   - Interactive oracle proofs (IOPs): like IPs but <u>force</u> honest prover behavior
   - Protocol lets prover convince the verifier these particular polynomials have the property hold
5. Use cryptography to “compile” this IOP to a real protocol:
   1. Compile IOP to interactive argument of knowledge via polynomial commitments
   2. Remove interaction using fiat-shamir transform
   3. Hide witness (ZK) by blinding “leaky” parts of proof with random data

Right now this is all (intentionally) very abstract. The next two projects and several weeks of class will make this very concrete.

Agenda for this lecture

- Announcements
- What is a “backend”?
- Interactive proofs
- The sumcheck protocol
- Analyzing sumcheck
- (If time) Applications of sumcheck

Interactive proofs

Protocol b/w two parties $\mathbb{P}$ $\mathbb{V}$

$\mathbb{P}$ is "for" a language L

$\mathbb{P}(x)$ $\qquad$ $\mathbb{V}(x)$ $\qquad$ $x \in L$

$\downarrow$ $\qquad\qquad$ $\searrow$

no output $\qquad\qquad$ $\{0/1\}$

Round: pair of $\rightleftarrows$ messages

$\mathbb{P}$ sends last message

Public-coin: $\mathbb{V}$ sends random messages

$\mathrm{Out}_{\mathbb{V}}[\mathbb{P}(x) \leftrightarrow \mathbb{V}(x)]$ $\qquad$ $\mathrm{view}_{\mathbb{V}}[\mathbb{P}(x) \leftrightarrow \mathbb{V}(x)]$

Properties of interactive proofs

$\mathbb{IP}$ for L
$= \langle \mathbb{P}, \mathbb{V} \rangle$

Completeness $\forall (x, w) \in R_L$

$$\Pr[\mathrm{out}_{\mathbb{V}}[\mathbb{P}(x,w) \leftrightarrow \mathbb{V}(x)] = 1] = 1$$

ε-soundness $\forall x \notin L \ \forall \mathbb{P}^*(x)$

$$\Pr[\mathrm{out}_{\mathbb{V}}[\mathbb{P}^*(x) \leftrightarrow \mathbb{V}(x)] = 1] \leq \varepsilon$$

Agenda for this lecture

- Announcements
- What is a “backend”?
- Interactive proofs
- The sumcheck protocol
- Analyzing sumcheck
- (If time) Applications of sumcheck

The sumcheck protocol

g is ℓ-variate over $\mathbb{F}$

d is maximal degree

Claim : $\sum_{i \in \{0,1\}^\ell} g(i) = H$

Sumcheck.Prover(g, H)		Sumcheck.Verifier(g, H)
Compute $g_1(X_1)$ as $\sum_{x_2,\ldots,x_v \in \{0,1\}} g(X_1, x_2, \ldots, x_v)$		
	$\xrightarrow{g_1}$	
		Check $H = g_1(0) + g_1(1)$
		$r_1 \leftarrow_{\$} \mathbb{F}$
	$\xleftarrow{r_1}$	

The sumcheck protocol

Sumcheck.Prover(g, H)		Sumcheck.Verifier(g, H)
Compute $g_1(X_1)$ as $\sum_{x_2,\ldots,x_v \in \{0,1\}} g(X_1, x_2, \ldots, x_v)$		
	$\xrightarrow{g_1}$	
		Check $H = g_1(0) + g_1(1)$
		$r_1 \leftarrow\!\$\; \mathbb{F}$
	$\xleftarrow{r_1}$	

. . . . In round j: . . . .

Compute $g_j(X_j)$ as $\sum_{x_{j+1},\ldots,x_v \in \{0,1\}} g(r_1, \ldots, r_{j-1}, X_j, x_{j+1}, \ldots, x_v)$		
	$\xrightarrow{g_j}$	
		Check $g_{j-1}(r_{j-1}) = g_j(0) + g_j(1)$
		$r_j \leftarrow\!\$\; \mathbb{F}$
	$\xleftarrow{r_j}$	

The sumcheck protocol

Sumcheck.Prover(g, H)		Sumcheck.Verifier(g, H)
Compute $g_1(X_1)$ as $\sum_{x_2,\ldots,x_v \in \{0,1\}} g(X_1, x_2, \ldots, x_v)$		
	$\xrightarrow{g_1}$	
		Check $H = g_1(0) + g_1(1)$
		$r_1 \leftarrow\$ \mathbb{F}$
	$\xleftarrow{r_1}$	

In round j:

Compute $g_j(X_j)$ as $\sum_{x_{j+1},\ldots,x_v \in \{0,1\}} g(r_1, \ldots, r_{j-1}, X_j, x_{j+1}, \ldots, x_v)$		
	$\xrightarrow{g_j}$	
		Check $g_{j-1}(r_{j-1}) = g_j(0) + g_j(1)$
		$r_j \leftarrow\$ \mathbb{F}$
	$\xleftarrow{r_j}$	

Final round:

$g_v \leftarrow g(r_1, \ldots, r_{v-1}, X_v)$		
	$\xrightarrow{g_v}$	
		Check $g_{v-1}(r_{v-1}) = g_v(0) + g_v(1)$
		$r_v \leftarrow\$ \mathbb{F}$
		Check $g_v(r_v) = g(r_1, \ldots, r_v)$
		If all checks pass: Return true
		Else return false

Agenda for this lecture

- Announcements
- What is a “backend”?
- Interactive proofs
- The sumcheck protocol
- Analyzing sumcheck
- (If time) Applications of sumcheck

Analyzing sumcheck

Theorem 1. *Let g be an ℓ-variate polynomial of maximal degree d over the field $\mathbb{F}$. Then for any malicious prover $\mathbb{P}^*$ and value $H \neq \sum_{i \in \{0,1\}^\ell} g(i)$,*

$$\Pr[\mathsf{out}_{\mathbb{V}}\left[\mathbb{P}^*(g, H) \leftrightarrow \mathbb{V}(g, H)\right] = 1] \leq \frac{\ell d}{|\mathbb{F}|}$$

Agenda for this lecture

- Announcements
- What is a “backend”?
- Interactive proofs
- The sumcheck protocol
- Analyzing sumcheck
- (If time) Applications of sumcheck

Applications of sumcheck