So far in this course, we looked at building zero-knowledge proof systems – goal is minimizing the "knowledge complexity"

↳ in this lecture, our goal is to minimize the communication complexity of the proof system

Soundness against unbounded provers

soundness against computationally bounded provers

What is a succinct proof / argument system?

Consider the language of Boolean circuit satisfiability: $\mathcal{L}_C = \{ \underbrace{x \in \{0,1\}^n}_{\text{statement}} \mid \exists \underbrace{w \in \{0,1\}^m}_{\text{witness}} : \underbrace{C}_{\text{Boolean circuit}}(x, w) = 1 \}$

Trivial proof system for $\mathcal{L}_C$:

prover (x, w) $\qquad\qquad$ verifier (x)

$\xrightarrow{\quad w \quad}$

↳ check that $C(x,w) = 1$

Proof size is $|w|$. Can we have a proof system where the total communication is significantly shorter than the witness?

Definition. A non-interactive proof/argument system for $\mathcal{L}_C$ is succinct if

- the length of the proof π satisfies $|\pi| = \text{poly}(\lambda, \log |C|)$
- the running time of the verifier is $\text{poly}(\lambda, |x|, \log |C|)$

both the proof size as well as the verifier complexity are much smaller than the size of the NP witness

Very strong notion: succinct non-interactive proofs (statistically sound proofs) unlikely to exist unless $NP \subseteq DTIME(2^{o(n)})$

statements of length n

↳ Even for argument systems (computationally sound proofs), succinct non-interactive arguments unlikely in the standard model

↳ Constructions known in the random oracle model and in the common reference string (CRS) model

"CS proofs" – computationally sound proofs

[Kilian's protocol + Fiat-Shamir]

constructions from pairings (public verifiability)

Constructions from additively homomorphic encryption [more precisely, linear-only]

(secretly verifiable)

Related primitives: SNARK: Succinct arguments of knowledge (SNARG + proof of knowledge)

zkSNARK: zero-knowledge succinct arguments of knowledge (zero-knowledge + SNARK) ⟵ core primitive behind Zcash

High-level blueprint for constructing SNARGs

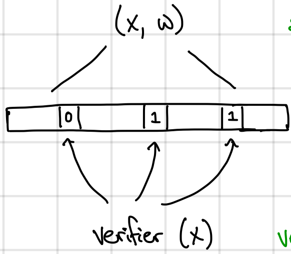
1. Construct information-theoretic proof system with security against an algebraically-bounded prover
2. Apply a cryptographic primitive to bind computationally-bounded provers to respect information-theoretic constraints

Today: Kilian's protocol: PCP + CRHF ⟹ succinct interactive argument

↳ Can be made non-interactive via Fiat-Shamir

Information-theoretic primitive: probabilistically-checkable proofs (PCPs)

Traditional PCPs:



statement-witness pair for an NP language

but computable in poly(|x|) time

probabilistically-checkable proof (long bitstring)

chosen randomly

Verifier reads a constant number of bits of the PCP and is convinced with constant probability! [long line of results — one of the deepest results of complexity theory!]

Model as following tuple of algorithms:

for Boolean circuit satisfiability, $\ell = \text{poly}(|C|)$, where C is the Boolean circuit

- $\text{Prove}(x, w) \to \pi \in \{0,1\}^\ell$ [Encodes statement and witness as PCP]
- $\text{Query}(x) \to (st, i_1, \dots, i_k)$ [Returns the k indices the verifier reads and a verification state] — query indices are non-adaptive
- $\text{Verify}(st, \{\pi_{i_j}\}_{j \in [k]}) \to 0/1$ [Accepts/rejects given bits of the PCP proof]

Completeness: For all x, w where $R(x, w) = 1$:

$$\Pr[\text{Verify}(st, \{\pi_{i_j}\}_{j\in[k]}) = 1 : \pi \leftarrow \text{Prove}(x,w),\ (st, i_1, \dots, i_k) \leftarrow \text{Query}(x)] = 1$$

Soundness: For all $x \notin \mathcal{L}$ and all $\pi^* \in \{0,1\}^\ell$

$$\Pr[\text{Verify}(st, \{\pi^*_{i_j}\}_{j\in[k]}) = 1 : (st, i_1, \dots, i_k) \leftarrow \text{Query}(x)] \le \frac{1}{3}.$$

Approach to construct succinct argument for NP from PCPs:

prover (x,w)		verifier (x)
$\pi \leftarrow \text{Prove}(x,w)$		
	$\xrightarrow{\pi}$	
		- Verifier runs the PCP verification on π (only needs to read a few bits of π)

Protocol is NOT succinct!

Verifier does not need to read all of the PCP.

Can we let prover only send the bits the verifier needs to read?

prover (x,w)		verifier (x)
$\pi \leftarrow \text{Prove}(x,w)$		$(st, i_1, \dots, i_k) \leftarrow \text{Query}(x)$
	$\xleftarrow{i_1, \dots, i_k}$	
	$\xrightarrow{\pi_{i_1}, \dots, \pi_{i_k}}$	
		↳ accept if $\text{Verify}(st, \pi_{i_1}, \dots, \pi_{i_k}) = 1$

Protocol is NOT sound!

Prover can choose the values of $\pi_{i_1}, \dots, \pi_{i_k}$ so that verifier always accepts. Soundness of PCP only applies in setting where PCP is independent of verifier's queries.

Solution: Prover commits to PCP first and then verifier reveals queries.

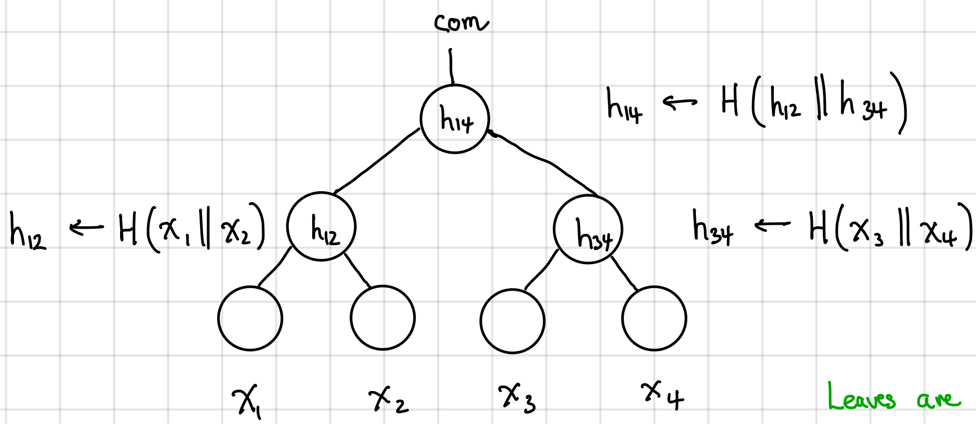
Commitment scheme should be succinct and computationally binding. (No need for hiding).

↳ construction from HW5 does not satisfy this requirement!

Merkle trees: Vector commitment scheme with succinct openings.

Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a collision-resistant hash function.

We build a commitment on N-dimensional vectors $v \in \{0,1\}^N$ where $N = 2^t$ as follows:

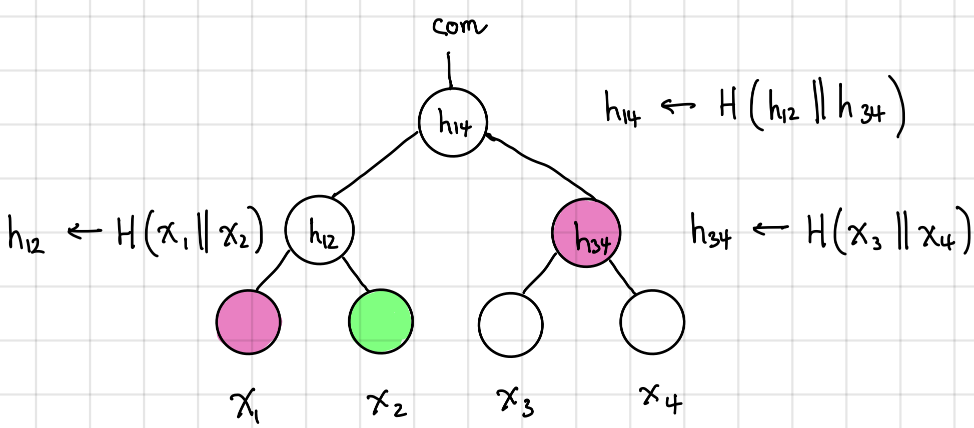


Each node is the hash of its children

Leaves are associated with input bits

Commitment is a single hash value (λ bit string, independent of N)

To open the value at x_i, it suffices to reveal the value of each sibling node from root to leaf i



↑ to open x_2, reveal x_1 and h_{34}

Verifier can compute every node from the leaf to the root and checks that root is consistent with commitment

└ "authentication path"

Theorem. If H is collision-resistant, then the Merkle tree is computationally binding.

Proof. We proceed inductively in the height of the tree. Suppose $t = 1$.

Then, $com = H(x_1 \| x_2)$. This is binding unless there exists a collision on 2-bit inputs.

Suppose Merkle trees with height t are computationally binding.

Suppose there is efficient adversary A that outputs

$$com \in \{0,1\}^\lambda, \quad i \in \mathbb{N}, \quad x_i, x_i' \in \{0,1\}, \quad (v_1, \dots, v_{t+1}), \quad (v_1', \dots, v_{t+1}')$$

where $x_i \neq x_i'$ and $(v_1, \dots, v_{t+1})$ is an opening of com to x_i and $(v_1', \dots, v_{t+1}')$ is an opening to x_i.

Then, either $v_1 = v_1'$ or we have a collision since it must be the case that

$H(u_1 \| v_1) = H(u_1' \| v_1')$ or $H(v_1 \| u_1) = H(v_1' \| u_1')$ for some value u_1, u_1', which is the hash of the sibling subtree

If $v_1 = v_1'$, then we consider the subtree rooted at the node associated with v_1. This is a Merkle tree of depth t, and by our inductive hypothesis, any adversary that breaks the binding property can also break collision-resistance of H.

Kilian's succinct argument for NP:

verifier chooses CRHF to defeat preprocessing attacks (e.g., having a hard-coded collision to H)

prover $\qquad\qquad\qquad\qquad$ verifier

$\xleftarrow{\quad H \quad}$

$\pi \leftarrow \text{Prove}(x, \omega)$

$\xrightarrow{\quad com \leftarrow \text{Commit}(\pi, \omega) \quad}$

$(st, i_1, \dots, i_k) \leftarrow \text{Query}(x)$

$\xleftarrow{\quad i_1, \dots, i_k \quad}$

$\xrightarrow{\quad \pi_{i_1}, \dots, \pi_{i_k}, \text{ openings for indices } i_1, \dots, i_k \quad}$

$\hookrightarrow$ accept if 1) valid opening to commitments

2) $\text{Verify}(st, \pi_{i_1}, \dots, \pi_{i_k}) = 1$

Soundness against bounded provers follows by binding property of commitment scheme and soundness of PCP.

(Intuition: com completely determines the proof string π)

(Formally: we can extract a PCP from the prover by rewinding)

<u>Succinctness</u>: $|\text{com}| = \lambda$

for constant soundness, need to read $O(1)$ bits of the PCP

↳ $O(1)$ openings — each opening has size $\lambda \cdot \log|\pi_{PCP}| = \lambda \cdot \log(\text{poly}(|C|))$
$= O(\lambda \log |C|)$

can amplify soundness to $1 - \text{negl}(\lambda)$ by repeating λ times so overall communication is $O(\lambda^2 \log |C|)$.

Kilian's protocol is public-coin so can apply Fiat-Shamir to obtain a succinct non-interactive argument (SNARG) for NP in the random oracle model.

Proof size is $O(\lambda^2 \log |C|)$.

Some recent progress by Chiesa-Yoger [CY21]

<u>Open problem</u>: can we get $O(\lambda \log |C|)$ size proofs?

Concrete efficiency of PCPs still quite high. Modern constructions consider a generalization called interactive oracle proof (IOP)

↳ Can be compiled to a succinct argument / SNARG in same manner.

Merkle trees are very useful also for building an <u>authenticated data structure</u>.

- <u>Certificate transparency</u>: log of all certificates issued by a CA
    - ↳ given Merkle root, very easy to <u>prove</u> that a certificate is in the log
- <u>Bitcoin</u>: Merkle tree used to prove that a particular transaction has been posted to the blockchain