Foundations of Probabilistic Proofs

A course by **Alessandro Chiesa**

Lecture 06

Intro to PCPs

Checking a proof without reading it?

GOAL in the next few lectures:

probabilistically check non-interactive proofs

at few locations with small error

CHALLENGE: what if the proof has only a single "mistake"?

(How to catch the mistake w.h.p.?)

EXAMPLE: given a graph $G=(V,E)$ and a coloring $\chi: V \to \{1,2,3\}$,

how to probabilistically check that χ is a 3-coloring of G?

(If G is not 3-colorable, at worst all but one edges are valid.)

The theory of PROBABILISTICALLY CHECKABLE PROOFS (PCPs)

addresses this challenge!

Tools and ideas from interactive proofs, coding theory,

property testing, and more.

NYT, 1992.04.07

Science Times

The New York Times

Biologists Tell a Tale Of Interfering In-Laws

Asteroid Defense: 'Risk Is Real,' Planners Say

New Short Cut Found For Long Math Proofs

New Short Cut Found For Long Math Proofs

A proof can be tested by checking just a part, inventors say.

By GINA KOLATA

IN a discovery that overturns centuries of mathematical tradition, a group of graduate students and young researchers has discovered a way to check even the longest and most complicated proof by scrutinizing it in just a few spots.

The finding, which some mathematicians say seems almost magical, depends upon transforming the set of logical statements that constitute a proof into a special mathematical form in which any error is so amplified as to be easily detectable.

Using this new result, the researchers have already made a landmark discovery in computer science. They showed that it is impossible to compute even approximate solutions for a large group of practical problems that have long foiled researchers. Even that negative finding is very significant, experts say, because in mathematics, a negative result, showing something is impossible, can be just as important and open just as many new areas of research as a positive one.

The discovery was made by Sanjeev Arora and Madhu Sudan, graduate students at the University of California at Berkeley, Dr. Rajeev Motwani, an assistant professor at Stanford University, and Dr. Carsten Lund and Dr. Mario Szegedy, young computer scientists at A.T.&T. Bell Laboratories. Dr. Motwani, who is the senior member of the group, just turned 30 on March 26.

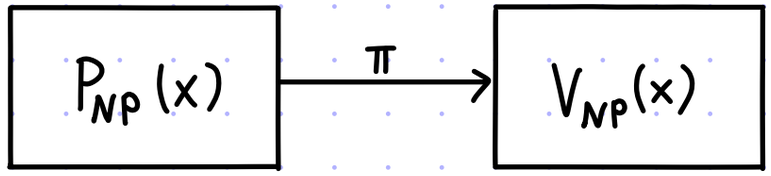
"With the conventional notion of a proof, you had to check it line by line," said Dr. Michael Sipser, a theoretical computer scientist at the Massachusetts Institute of Technology. "An error might be buried on page 475, line 6. A 'less than or equal to' should have been a 'less than.' That would totally trash the whole proof. But you'd have to dig through the whole thing to find it," Dr. Sipser said. Now, he added, "the new idea is that there is a way to transform any proof so that if there is an error, it appears almost everywhere. I'd say, 'You have a proof? Show me a page.' If there is an error, it will be there."

The finding, which is built on two and a one half years work by leading

Continued on Page C10

New Model: Probabilistically Checkable Proofs

NP represents proofs checkable via a deterministic polynomial-time verifier:



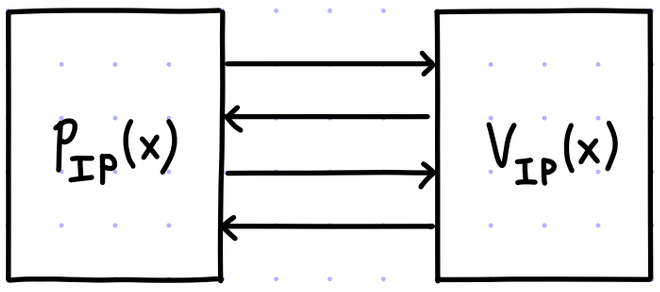
TODAY WE STUDY A NEW MODEL

INTERACTIVE PROOFS:

IPs are proofs checkable via a polynomial-time verifier that has these additional resources:

① randomness

② interaction

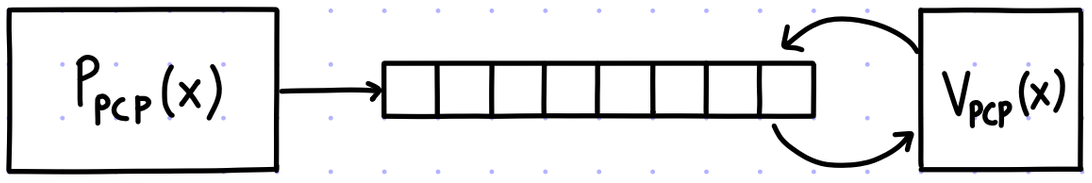


PROBABILISTICALLY-CHECKABLE PROOFS:

PCPs are proofs checkable via a polynomial-time verifier that has these additional resources:

① randomness

② oracle access to proof



Definition of PCP

[The definition for a language L is a special case.]

We say that (P,V) is a **PCP system** for a relation R with completeness error ε_c and soundness error ε_s (with $1-\varepsilon_c > \varepsilon_s$) if the following holds:

① **Completeness:** $\forall\,(x,w)\in R \quad \Pr\left[V^{\pi}(x)=1 \;\middle|\; \pi \leftarrow P(x,w)\right] \geq 1-\varepsilon_c$.

② **Soundness:** $\forall\, x \notin L(R) \quad \forall\, \tilde{P} \quad \Pr\left[V^{\tilde{\pi}}(x)=1 \;\middle|\; \tilde{\pi} \leftarrow \tilde{P}\right] \leq \varepsilon_s$.

Equivalently: $\forall\, x \notin L(R) \;\; \forall\, \tilde{\pi} \;\; \Pr[V^{\tilde{\pi}}(x)=1] \leq \varepsilon_s$

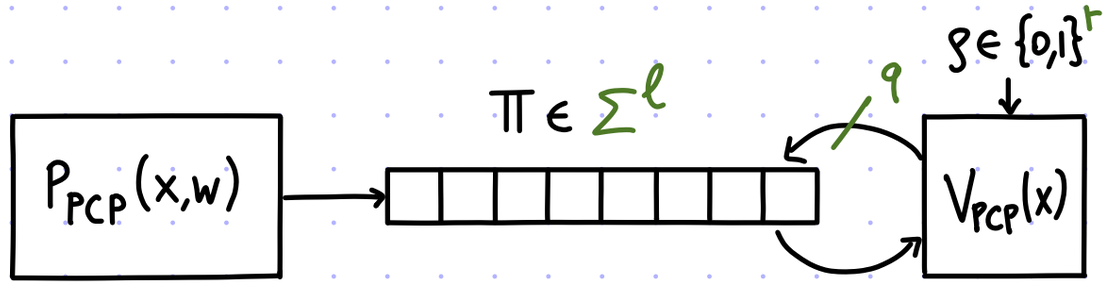
We use the notation $V^{\pi}(x;r)$ to make explicit that r is the randomness of V.

We call π a "PCP string". Intuitively, π is a "robust encoding" of a witness, which admits a probabilistic verification that reads a few symbols of it.

For IPs we cared about: round complexity, communication complexity, ...

For PCPs we have different parameters:

- Σ : proof alphabet
- ℓ : proof length
- q : verifier query complexity
- r : verifier randomness complexity



(Queries to π are typically non-adaptive.)

Some Special Cases

We wish to understand $PCP[\varepsilon_c, \varepsilon_s, \Sigma, \ell, q, r, \dots]$ in different regimes.

Suppose that there is no proof ($q=0$):

- $PCP[q=0, r=0] = P$ (no proof and no randomness)
- $PCP[q=0, r=O(\log n)] = P$ (can try all random strings in polynomial time)
- $PCP[q=0, r=\text{poly}(n)] = BPP$ (any amound of randomness and no proof)

Suppose that there is no randomness ($r=0$):

- $PCP[q=\text{poly}(n), r=0] = NP$ (queries are fixed and polynomially many)

A trivial PCP:

- $3SAT \in PCP[\varepsilon_c = 0, \varepsilon_s = 1-\frac{1}{m}, \Sigma=\{0,1\}, \ell = n, q=3, r=\log m]$

proof: $V_{PCP}^{a\in\{0,1\}^n}(\varphi) :=$ 1. Sample a random clause $j \in [m]$.
2. Check that a satisfies the j-th clause of φ. ■

We denote by PCP the case with no restrictions (beyond V_{PCP} runs in polynomial time):

$$PCP := PCP[\varepsilon_c = 0, \varepsilon_s = 1/2, |\Sigma| = \exp(n), \ell = \exp(n), q = \text{poly}(n), r = \text{poly}(n)]$$

Upper Bound: NEXP

- Σ : proof alphabet
- ℓ : proof length
- q : verifier query complexity
- r : verifier randomness complexity

theorem: $PCP \subseteq NEXP$

REVIEW: $NTIME[T] = \left\{ L \;\middle|\; \exists \text{ machine } M \text{ s.t. } \begin{cases} x \in L \to \exists w\ M(x,w)=1 \\ x \notin L \to \forall w\ M(x,w)=0 \end{cases} \text{ and } M(x,w) \text{ runs in } T(|x|) \text{ time} \right\}$

nondeterministic polynomial time: $NP = \bigcup_{c \in \mathbb{N}} NTIME[n^c]$

nondeterministic exponential time: $NEXP = \bigcup_{c \in \mathbb{N}} NTIME[2^{n^c}]$

lemma: $PCP[\varepsilon_c, \varepsilon_s, \Sigma, \ell, r] \subseteq NTIME[T = (2^r + \ell \cdot \log|\Sigma|) \cdot poly(n)]$

proof: Let (P,V) be a PCP system for L with the parameters above.

Define the decider as follows:

$D(x, \pi) :=$ (where $x \in \{0,1\}^n$, $\pi \in \Sigma^\ell$)

1. For every $\rho \in \{0,1\}^r$: compute $b_\rho := V^\pi(x;\rho) \in \{0,1\}$.
2. Output 1 if and only if $(\Sigma_\rho b_\rho)/2^r \geq 1-\varepsilon_c$.

If $x \in L$ then $\exists \pi$ s.t. $D(x,\pi)=1$. If $x \notin L$ then $\forall \pi$ $D(x,\pi)=0$. ■

lemma: (i) $\ell \leq 2^r \cdot q$ for non-adaptive verifiers

(ii) $\ell \leq 2^r \cdot |\Sigma|^q \cdot q$ for adaptive verifiers

(in PCP constructions ℓ is typically smaller than these upper bounds)

proof of (i): there are at most 2^r different query sets

proof of (ii): each query answer may lead to a different next query ■

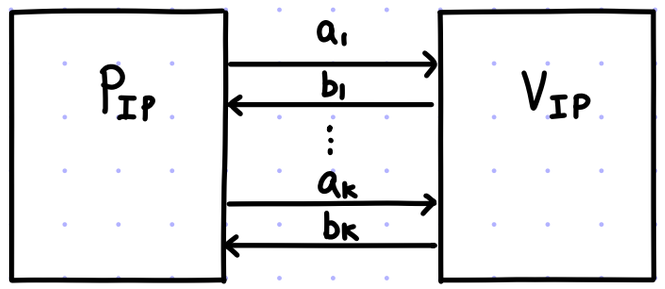
Lower Bound: PSPACE

theorem: PSPACE ⊆ PCP

We proved that IP = PSPACE. So the following lemma suffices.

lemma: $\mathrm{IP}[\varepsilon_c, \varepsilon_s, \underbrace{k}_{\text{rounds}}] \subseteq \mathrm{PCP}[\varepsilon_c, \varepsilon_s, q=k]$

proof: Let (P_{IP}, V_{IP}) be a k-round IP for L.



Consider PCP strings in this format:

$$\pi := \left(a_1, (a_2[b_1])_{b_1}, (a_3[b_1,b_2])_{b_1,b_2}, \ldots, (a_k[b_1,\ldots,b_{k-1}])_{b_1,\ldots,b_{k-1}}\right)$$

The PCP verifier (which adaptively makes k queries to π) works as follows:

$V^{\pi}(x)$:= 1. Sample IP randomness ρ.

2. Simulate $V_{IP}(x;\rho)$ where, in round i, the IP prover message is $a_i[\overbrace{b_1,\ldots,b_{i-1}}^{\text{prior } i-1 \text{ messages by } V_{IP}(x;\rho)}]$.

COMPLETENESS: the honest PCP string is $\pi := \left(P_{IP}(x), (P_{IP}(x,b_1))_{b_1}, \ldots, (P_{IP}(x,b_1,\ldots,b_{k-1}))_{b_1,\ldots,b_{k-1}}\right)$.

SOUNDNESS: any PCP string in the above format is an "unrolled" IP prover. ■

If V_{IP} is PUBLIC-COIN then each b_i is (independently) random.

Hence $\rho = (b_1,\ldots,b_k)$ and the k queries $((b_1,\ldots,b_{i-1}))_{i\in[k]}$ to π are non-adaptively determined.

Preview of PCP Capabilities

We can ask several questions about PCPs, inspired from our study of IPs.

- Which languages have PCPs? We learned that $PSPACE \subseteq PCP \subseteq NEXP$.

  A: $PCP = NEXP$

- Do PCPs have benefits for NP languages? E.g. PCP query complexity $\ll$ NP witness size

  A: Yes

- Do PCPs have benefits for P languages? E.g. PCP verifier time $\ll$ P decision time

  A: Yes

- Are there ZK PCPs for NP?

  A: Yes

$\longrightarrow$ Many Good News!

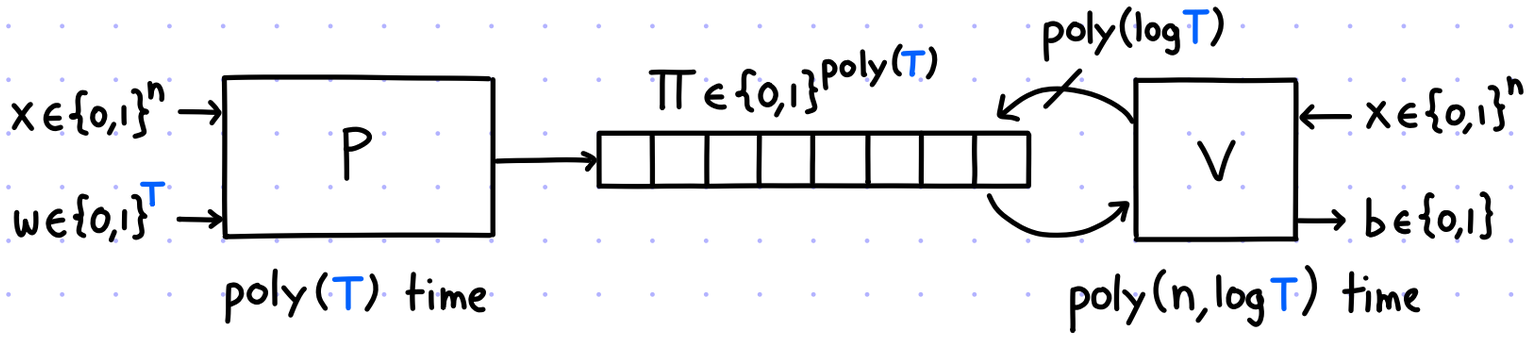
We study most of these questions in the next few lectures.

Delegation of Computation via PCPs

We will prove (in a few lectures) the PCP Delegation Of Computation Theorem:

$$NTIME[T] = \left\{ L \;\middle|\; \exists \text{ machine } M \text{ s.t. } \begin{cases} x \in L \to \exists w\ M(x,w)=1 \\ x \notin L \to \forall w\ M(x,w)=0 \end{cases} \text{ and } M(x,w) \text{ runs in } T(|x|) \text{ time} \right\}$$

theorem: $NTIME[T] \subseteq PCP\left[\begin{array}{ll} \text{proof length } \ell = \text{poly}(T), & \text{query complexity } q = \text{poly}(\log T) \\ \text{prover time } pt = \text{poly}(T), & \text{verifier time } vt = \text{poly}(n, \log T) \end{array}\right]$



Proving the theorem will involve:

- recycling techniques (arithmetization, sumcheck protocol, ...)
- new ideas (low-degree testing, succinct arithmetization, ...)

Checking Computations in Polylogarithmic Time


László Babai
Univ. of Chicago [6] and
Eötvös Univ., Budapest


Lance Fortnow
Dept. Comp. Sci.
Univ. of Chicago [6]


Leonid A. Levin
Dept. Comp. Sci.
Boston University [4]


Mario Szegedy
Dept. Comp. Sci.
Univ. of Chicago [6]

The PCP Model Is Unrealistic

Checking Computations in Polylogarithmic Time

László Babai
Univ. of Chicago [6] and
Eötvös Univ., Budapest

Lance Fortnow
Dept. Comp. Sci.
Univ. of Chicago [6]

Leonid A. Levin
Dept. Comp. Sci.
Boston University [4]

Mario Szegedy
Dept. Comp. Sci.
Univ. of Chicago [6]

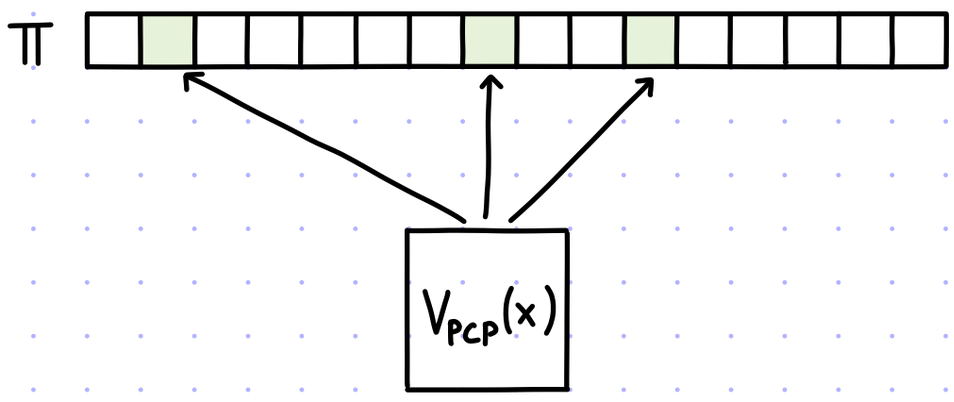
In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware.

CHALLENGE: How can PCPs be useful?

The PCP model gives the verifier an urealistic "super" power:

ORACLE ACCESS to a long proof π.

How can a verifier have such a capability?



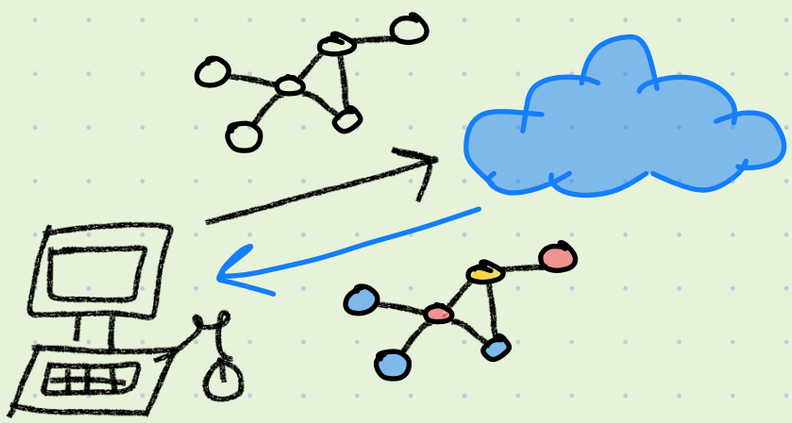
Applications of PCPs

PCPs are useful as an intermediate model.

They are an invaluable TOOL in two major applications.

SUCCINCT ARGUMENTS

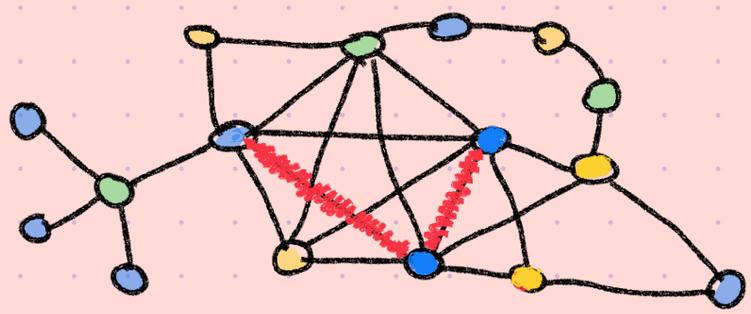
Cryptographic proofs with strong efficiency features (e.g. for delegation of computation).



TODAY

HARDNESS OF APPROXIMATION

Which problems remain hard even if we only seek an approximate solution?



LATER LECTURE

A Cryptography Jaunt: Succinct Arguments Based On PCPs

Succinct Interactive Arguments

An interactive argument (IA) is an interactive proof (IP) where soundness is relaxed to:

COMPUTATIONAL SOUNDNESS: $\forall x \notin L(R) \quad \forall$ efficient $\tilde{P} \quad \Pr_{r_V}\left[\langle \tilde{P}(1^\lambda), V(1^\lambda, x; r_V)\rangle = 1\right] \leq \varepsilon_s(\lambda, x)$.

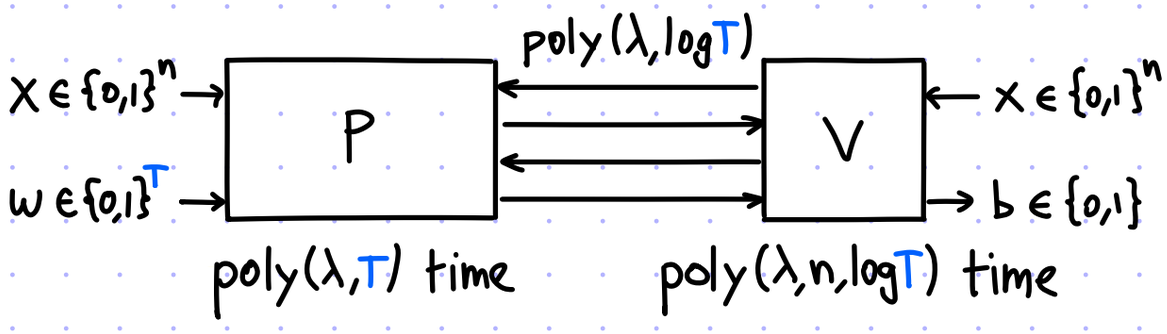
Theorem: Suppose that $L \in PCP\left[\begin{array}{ll} \text{proof alphabet } \Sigma & \text{prover time } pt \\ \text{proof length } \ell & \\ \text{query complexity } q & \text{verifier time } vt \end{array}\right]$.

Then we can use crypto to construct a public-coin interactive argument for L with:

round complexity $k=2$ prover time $O_\lambda(pt)$

communication complexity $O_\lambda(q \cdot \log|\Sigma| \cdot \log \ell)$ verifier time $O_\lambda(vt)$

If we apply this transformation to the PCP on the prior slides, then we get a succinct interactive argument:



NOTE: this does NOT contradict the limitations of IPs with small communication.

Two Attempts At A Succinct Argument

Proof Attempt #1:

$P_{IA}(x,w)$

- Produce PCP string: $\pi := P_{PCP}(x,w)$.
- Deduce PCP query set Q for $V_{PCP}^{\pi}(x;\rho)$.
- Set $a := \pi[Q] \in \Sigma^Q$.

ρ ←

→ a

$V_{IA}(x)$

Sample PCP randomness $\rho \in \{0,1\}^r$.

$V_{PCP}^{[Q,a]}(x;\rho) \stackrel{?}{=} 1$

Problem: a malicious prover can choose answers $a \in \Sigma^Q$ based on the randomness ρ.

(Separately, we cannot hope to succeed without cryptography.)

Proof Attempt #2:

$P_{IA}(x,w)$

- Produce PCP string: $\pi := P_{PCP}(x,w)$.
- Commit to PCP string: $cm := \text{Hash}(\pi)$.
- Deduce PCP query set Q for $V_{PCP}^{\pi}(x;\rho)$.
- Set $a := \pi[Q] \in \Sigma^Q$.

cm →

ρ ←

a, π →

$V_{IA}(x)$

Sample PCP randomness $\rho \in \{0,1\}^r$.

$V_{PCP}^{[Q,a]}(x;\rho) \stackrel{?}{=} 1$ $\wedge\ \text{Hash}(\pi) \stackrel{?}{=} cm$ $\wedge\ \pi[Q] \stackrel{?}{=} a$

Problem: the honest prover sends the entire PCP string.

Merkle Commitment Schemes

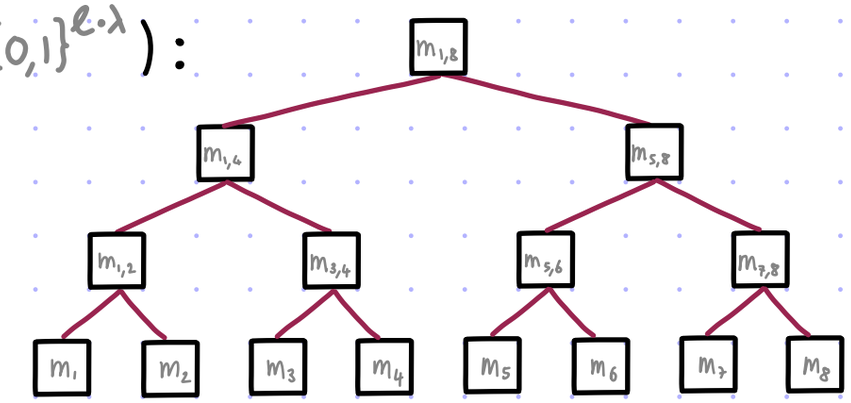
We need a short commitment with local openings.

An example is a MERKLE COMMITMENT.

Let $h: \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ be a hash function.

- MT[h].Commit($m \in \{0,1\}^{\ell \cdot \lambda}$):

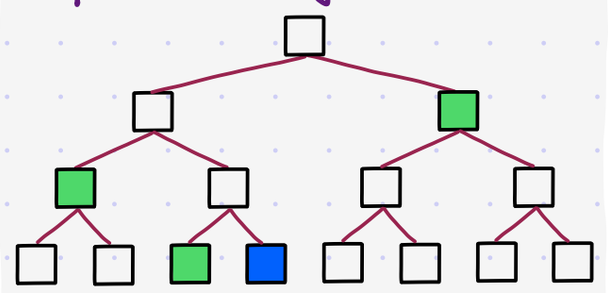
  Pairwise hash the message until you obtain one block.



Output:

- Merkle root: $rt := m_{1,8} \in \{0,1\}^{\lambda}$.
- auxiliary info: $aux := (m, (m_{i,j})_{i,j})$.

Copath(4) is green below:



- MT[h].Open($aux, Q \subseteq [\ell]$):

  For every $i \in Q$, set $pf_i := (m_v)_{v \in copath(i)}$. Output $pf := (pf_i)_{i \in Q}$.

- MT[h].Check($rt, Q \subseteq [\ell], a \in (\{0,1\}^{\lambda})^{Q}, pf = (pf_i)_{i \in Q}$):

  For every $i \in Q$, check that pf_i authenticates $a[i]$ for position i relative to rt.

lemma: $\mathcal{H} = \{H_\lambda = \{h_\lambda : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}\}\}_{\lambda \in \mathbb{N}}$ is collision resistant $\to$ MT[$\mathcal{H}$] is position-binding.

$$\forall \text{ efficient } A \quad \Pr_{h \leftarrow H_\lambda}\left[\begin{matrix} x \neq y \\ h(x)=h(y) \end{matrix} \middle| (x,y) \leftarrow A(h)\right] = \text{negl}(\lambda)$$

$$\forall \text{ efficient } A \quad \Pr_{h \leftarrow H_\lambda}\left[\begin{matrix} \exists i \in Q_1 \cap Q_2 : a_1[i] \neq a_2[i] \\ \text{MT[h].Check}(rt, Q_1, a_1, pf_1) = 1 \\ \text{MT[h].Check}(rt, Q_2, a_2, pf_2) = 1 \end{matrix} \middle| (rt, \begin{matrix} Q_1, a_1, pf_1 \\ Q_2, a_2, pf_2 \end{matrix}) \leftarrow A(h)\right] = \text{negl}(\lambda)$$

Kilian's Protocol

First commit to the PCP string and then locally open it.

$P_{IA}(1^\lambda, x, w)$

- Produce PCP string: $\pi := P_{PCP}(x,w) \in \Sigma^\ell$.
- Commit to it: $(rt, aux) := MT[h].Commit(\pi)$.
  [For simplicity here we assume $\log|\Sigma| \leq \lambda$.]
- Deduce PCP query set $Q \subseteq [\ell]$ for $V_{PCP}^{\pi}(x;\rho)$.
- Set answers $a := \pi[Q] \in \Sigma^Q$.
- Authenticate answers: $pf := MT[h].Open(aux, Q)$.

$h \leftarrow$

$\xrightarrow{rt}$

$\leftarrow \rho$

$\xrightarrow{Q, a, pf}$

$V_{IA}(1^\lambda, x)$

Sample CRH: $h \leftarrow H_\lambda$.

Sample PCP randomness $\rho \in \{0,1\}^r$.

$V_{PCP}^{[Q,a]}(x;\rho) \stackrel{?}{=} 1$

$MT[h].Check(rt, Q, a, pf) \stackrel{?}{=} 1$

- round complexity: 2
- communication complexity: $poly(\lambda) + \lambda + r + q \cdot (\log\ell + \log|\Sigma| + \lambda \cdot \log\ell) \approx poly(\lambda) + r + q \cdot (\log|\Sigma| + \lambda \cdot \log\ell)$.
- prover time: $time(P_{PCP}) + time(V_{PCP}) + O_\lambda(\ell \cdot \log|\Sigma|) \approx pt + O_\lambda(\ell \cdot \log|\Sigma|)$.
- verifier time: $poly(\lambda) + r + time(V_{PCP}) + O_\lambda(\log\ell \cdot \log|\Sigma|) \approx vt + O_\lambda(\log\ell \cdot \log|\Sigma|)$.

SECURITY (beyond the scope of this class): a REWINDING ARGUMENT, assuming that h is collision resistant
(more generally: a position-binding vector commitment)

Bibliography

PCPs

- [BFLS 1991]: Checking computations in polylogarithmic time, by László Babai, Lance Fortnow, Leonid Levin, Mario Szegedy.
- [FGLSS 1996]: Interactive proofs and the hardness of approximating cliques, by Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, Mario Szegedy.
- [AS 1998]: Probabilistic checking of proofs: a new characterization of NP, by Sanjeev Arora, Shmuel Safra.
- [ALMSS 1998]: Proof verification and the hardness of approximation problems, by Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, Mario Szegedy.
- How computer scientists learned to reinvent the proof. Quanta 2022.
- (▸Scientific revolutions, ToC and PCP), by Avi Wigderson.
- New short cut found for long math proofs, New York Times 1992.

Succinct arguments from PCPs

- [Kilian 1992]: A note on efficient zero-knowledge proofs and arguments, by Joe Kilian.
- [Micali 2000]: Computationally sound proofs, by Silvio Micali.
- [Merkle 1989]: A certified digital signature, by Ralph Merkle.
- [CDGSY 2024]: Untangling the security of Kilian's protocol: upper and lower bounds, by Alessandro Chiesa, Marcel Dall'Agnol, Ziyi Guan, Nicholas Spooner, Eylon Yogev.
- [CY 2024]: Building cryptographic proofs from hash functions, by Alessandro Chiesa, Eylon Yogev. (▸Video)