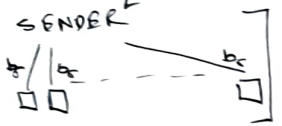
DOLEV STRONG

$ext_i = \phi$ (for every node i)

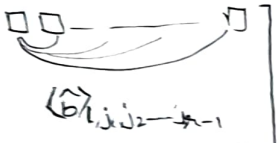
ROUND 0

SENDER



Round 0

$r=1$ with S



$\langle b \rangle_{i,j,j_2 \ldots j_{r-1}}$

$r = f+1$

IF $\langle S \rangle \notin ext_i$ ADD $\langle S \rangle_i$

send $\langle b \rangle_{j,j_1 \ldots j_{r-1},i}$ to everyone.

At end, IF $|ext_i| = 1$ node i o/p bit in $\langle ext_i \rangle$ else node i o/p $\langle 0 \rangle$

After $f+1$ Round

4 ASSUMPTIONS

- PERMISSIONED [#Nodes, IPaddr all known upfront]
- PKI [can setup PK/SK pair]
- SYNCHRONY (IF HONEST NODE SENDS (m) in round(r) IT REACHES DES BY $(r+1)^{th}$ round
- ALL HONEST

NAIVE MAJ VOTING PROTOCOL

- SENDER ← I/P = b

Round 1

ROUND 2 (VOTES SENT TO ALL OTHER NON SENDER)

IF NO BIT/BOTH BIT RECEIVED $> n/2$ VOTES

O/P = 0

ELSE 'b'

ROUND 3

- VIOLATES AGREEMENT in case of BYZ SENDR.

SMR — CONSISTENCY, LIVENESS

BB — AGREEMENT, VALIDITY

BB without PKI (TB^wt Bound) (Byz $n/3$)

Byz $n/3$ is this $< n/3$ STRICTLY. ARE ONLY CORRUPT

STICKY BIT $\in \{0, 1, \perp\}$

SENDER STICKY BIT = $\langle$ I/P BIT $\rangle$

EVERY NONSENDER STICKY BIT = $\langle \perp \rangle$

LEADER CHOSEN BY

$L_r := H(r)$ $\quad H: \{0,1\}^* \to [n]$ RANDOM ORACLE

FOR EACH ROUND $r = \{1, 2 \ldots k\}$ SUFFICIENTLY many rounds.

ROUND 0: (Leader)

IF L_r STICKY BIT $\neq \perp$ STICKY BIT = $\langle b \rangle$

ELSE $\langle b \rangle \in \{0,1\}$ randomly

ROUND 1:

IF a node has STICKY BIT $\neq \perp$ choose STICK BIT

ELSE choose L_r.

VOTE ON ITS STICKY BIT

ROUND 2:

IF $\geq \frac{2n}{3}$ nodes VOTED ON b'

UPdate its sticky bit to b'

ELSE update it to $\perp$.

LUCKY ROUND

- IF L_r honest
- L_r proposes $\langle b \rangle$ in 'r' & no honest sticky bit at beg of 'r+1' is $\langle \bar{b} \rangle$

(cost of this validity)

Lemma 4: IF $r \leq R$ is lucky L_r proposes $b \in \{0,1\}$ in r then every honest node sticky bit at end of protocol is $b \in \{0,1\}$

PROOF: By Lemma 3.

Lemma 5: $\forall r \; 1 \leq r \leq R$ even when no round prior to r are lucky, r^{th} iteration is lucky.

$\frac{1}{2} \cdot \frac{2}{3} \geq \frac{1}{3}$

Round Honest any Nodes lucky

Lemma 3: Node i sees $\geq 2n/3$ votes for b | Node j sees $\geq 2n/3$ votes for $\bar{b}$ | CANNOT HAPPEN

PROOF: $|S_i \cap S_j| \geq n/3$

Lem 6: $\exists$ lucky its with prob $= 1 - (2/3)^k$

THM 4: CONSISTENCY WITH $P \geq 1 - (2/3)^R$, $\forall$ nodes O/P same

THM 5: VALIDITY IF L_r honest → Honest nonsender O/P 1.

	PERMISSIONED	PKI	SYNCHRONY
$f=n-1$ DOLEV STRONG NAIVE (SYNC)	✓	✓	✓
TENDERMINT (PART SYNC)	✓	✓	PARTIAL

CAP ~~FLP~~

P → Part-Tolerance means C, A both hold even in network partition

CAP vs FLP

No dis sys (ASYNC) HAS ALL 3 IF IN NOT PART Then grow up only C, A

Holds for $f=0$

No Det. PRO to BA in if $f=1$ in ASYNC

No Work if $f=0$

Message delay

SYN	ASYC	PART	
✓	X	✓	SHARED GLOBAL CLOCK
✓	X	✓ SYNC start	Δ = known (priors) msg delay bound
DOLEV STRONG (with PKI ok)	WEAK ASSUMPTION	GST	Good
RULES ON Network OUTAGES	FLP, No det prot solves BB ($f \geq 1$)	Unknown	BAD

BA prblm

- Termination: $\forall i$ halts with some O/P w_i
- Agreement: All honest nodes O/P same (no matter their I/P)
- Validity: If $v_i = v^*$ for $\forall$ honest nodes i then $w_i = v^* \forall i$

Goals in Part-SYNC Model

- Safety: Always hold even in ASYN phase
- Eventual liveness: Not long after the GST, safety & liveness both hold.

IMPOSSIBILITY RESULTS

	STMT	REMARKS
PSL 80 FLM85 (SYNC)	BB IMP under ($\geq 1/3$) corruption without PKI	DOLEV STRONG WORKS IN PKI
		PROOF: HEXAGONAL ARGUMENT
CAP Consistency Availability Partition Tolerance	Can't have all 3 together, must pick 2	
FLM	IN SYNC, $f \geq n/3$, no BB protocol satisfies: Ter, Agre, Validity	Proof HEX argument Sticky bit proof to get bound
FLP (ASYNC)	$N \geq 2$ $f \geq 1$, then also NO DETERMINISTIC protocol for BA satisfying TER, AGR VAL	
some THM PART SYNC	$\exists$ deterministic protocol for SMR with CON & liveness (post GST) iff $f < n/3$	

~~PROOF~~

ASYNC IMP FLP

- Time dilated (100 sec time steps) — Nodes stride for long
- No shared clock
- Arbitrary Δ.

Have to show even if all msgs arrive within some time, the above IMP holds.

PROOF (By contradiction)

- Assume it solves BA
- Config C [INFO TO RESTART WHEN IT left]
  - M (msg pool)
  - I/P [init I/P of each node]
  - RCVD msgs (seq of msgs received thus far)
- $C \xrightarrow{(p,m)} C'$ strategies which lead to V_0, V_1
- 3 types of config: 0, 1, ambi

BCOZ TI satisfies agreement (assumption)
- All honest nodes O/P 0 OR all O/P 1

BYZ node can say its O/P $<0,1$ forcing the config to ambi to all 0 or all 1

MODEL ASYNC (event driven (no shared clock))

1) M → pool of outstanding msgs $= \{(r, m)\}_{r=1}^{n}$ →
2) while (TRUE)
   - an arbitrary msg (r, m) is delivered to recipient 'r' →
   - r can add any no of messages to M. →

Since our goal is to get a protocol which has provable guarantee, let's assume they are all for BYZ nodes.

BYZ NODES CONTROL msg delivery →

BYZ AGREEMENT

- Why is it needed when we have BB & SMR →
- BCOZ, BB in ASYNC is trivially UNSOLVABLE
- HENCE BA, FLP IMP result → SMR IMP.

FLP IMP THM [2]

For $n \geq 2$, with → deterministic - msgs NO corrupt
$f \geq 1$, no deterministic protocol to BA satisfies TER, AGR, VAL IN ASYNC mode

O/P = f (I/P, msgs received so far)

Q) SBC of leader is sufficiently random to ensure [illegible] chain quality

Where does the penalty of PoS go redistribution OR lost

Sybil resistance

distinct n nodes.

$\underbrace{H_1 \cdots H_n}$

make n distinct nonces guess/sec

$Pr[\text{node } i \text{ is leader of round } x] = \frac{N_i}{\sum N_i}$

$N_i \rightarrow$ node hashrates / computational power.

+ The price of the bitcoin is dependent on the popularity of the bitcoin let's say if one guy has $>\frac{1}{2}$ all comp power then the other participants would leave the system, popularity decreases, and eventually the price.

+ POS & POW are sybil resistant mechanism where BFT & LCC are consensus mechanism.] Making it sybil would require a cost [POS - stake, POW - Compu power]

+ Nakamoto consensus.

+

POS	$\checkmark_{ETH}$	✓
POW	X	$\checkmark_{BITCOIN}$
	BFT	LCC

POW - BFT DN work well as voting is involved

In permissionless setting we need a sybil resistance mechanism because nodes are not known or just how many are there Why sybil resistance?

SNARK

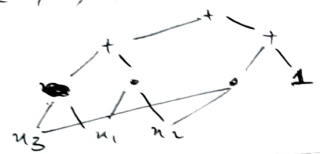
[short Proofs that are fast to verify]

Application (Non blockchain)
- C2PA standard (Camera)
- See the XVIII-th slide

SS
20/11/23

+ ~~Def of NARK~~
ARIthmetic circuit.

$p(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3 + 1$



+ NARK

+ C — public circuit
takes two inputs (x, ω)

$C(x, \omega) \longrightarrow \mathbb{F}$

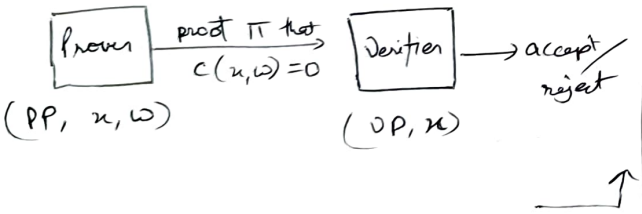
x = a public stmt $\quad x \in \mathbb{F}^n$

ω = witness $\quad \omega \in \mathbb{F}^m$

+ Preprocessing step

$S(C) = (PP, VP)$

PP: prover public parameter; VP: verifier public parameter



A NARK is a tuple of three algorithms (S, P, V) where

1) $S(C)$ returns a pair of public parameters (PP, VP) (Preprocessing)
2) $P(PP, x, \omega)$ produces a proof
3) $V^{\pi}(VP, x, \pi)$ either accepts / rejects the proof π.

All algorithms & adversary have access to Random Oracle.

SNARK
- Length of proof $\pi = |\omega|$ (sublinear)
- Runtime of V should be $O(|x|, \text{sublinear}(|C|))$

Strongly succint NARK ($|C|$ = length of circuit)
- Length of proof $|\pi| = O_\lambda(\log|C|)$
- Runtime of V $O_\lambda(|x| \log|C|)$
  - linear in x
  - it has to read x so linear in x & logarithmic in $|C|$

SECURITY PROP of NARK

1) Completeness
$\forall x, \omega : C(x,\omega)=0 \Rightarrow Pr\left[V(VP, x, P(PP, x, \omega)) = \text{accept}\right] = 1$

2) Adaptivity knowledge sound: V accepts $\Rightarrow$ P knows a 'ω' $| C(x,\omega)=0$

3) Optimal zero knowledge
(C, PP, VP, x, π) reveals nothing new about ω.

In SNARK, π is small.

TRANSPARENT
- ~~NO~~ $S(C)$ does not use any secret data
- NOTE, NO destruction

Bullet proof STARK, DARK

TYPES OF PREPROCESSING
- Trusted setup per circuit
- Trusted but universal updatable setup.
- Transparent

TRUSTED SETUP
- $S(C, r) \longrightarrow (PP, VP)$
  $r \in \{0,1\}^{\lambda}$
- r (kept secret from the prover)
- Prover learns $r \longrightarrow$ he can prove false stmts.
- $\forall C$, choose fresh r, making sure r is DESTROYED

TRUSTED BUT UNIVERSAL UP- - - -
- All ptcls in TRUSTED
- $S = (S_{init}, S_{index})$ deterministic
- $S_{init}(\lambda, r) \longrightarrow gp$
- $S_{index}(gp, C) \longrightarrow (PP, VP)$ (Anyone can verify (PP, VP) was correctly generated)

S_{init} machine is destroyed ~~PP~~

Formal def^n of Adaptively Knowledge Sound. Adversary has (A_0, A_1) also.

→ for prover

→ to ~~verifier~~

$gp \leftarrow Sinit(\lambda)$ → $st = \omega'$

$(C, x, st) \leftarrow A_0(gp)$

$(pp, vp) \leftarrow S_{index}(gp, C)$

$\pi \leftarrow A_1(pp, x, st)$

$P_r[V(vp, x, \pi) = accept] \geq 10^{-6}$

$\exists$ there is an efficient extractor E that uses A_0 A_1

$gp \leftarrow Sinit(\lambda)$

$(C, x, st) \leftarrow A_0(gp)$

$\omega \leftarrow E(gp, C, x)$

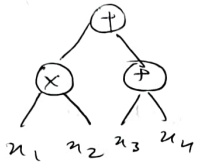
$P_r[C(x, \omega) = 0] \geq 10^{-6} - \epsilon$

Succinct preprocessing argument system

$P(pp, x, w) \to$ short π $\quad |\pi| = O(\log|c|, d)$

$V(pp, x, \pi) \to$ accept/reject $\quad time(V) = O(|x|, \log|c|, d)$

$|c|$ = # gates in c



Q) WHY preprocess C,

Ans). V( ) is logarithmic in |c| so, there's no time to read circuit C. And if V can't read the circuit C, how is it going to verify π.

So preprocessing creates a short summary of circuit (a short string that V can read quickly).

S(C) does that summarizing of C for V to verify π in log|c| time

TRUSTED

TRUSTED BUT UNIVERSAL

TRANSPARENT

Better

Def(2) ZK, SPV is ZK if $\forall x \in L^n$ proof π reveals nothing new about w.

- If V can gen π by itself → it learned nothing new about π
- If $\exists$ efficient SIM s.t. $(s_P, s_V, \pi) \leftarrow SIM(C, x)$ looks like the real $(s_P, s_V, \text{and } \pi)$

- (C, s_P, s_V, x, π) : $S(C) \to (s_P, s_V)$, $\pi \leftarrow P(s_P, x, w)$ is IND + m
  (C, s_P, s_V, x, π) : $(s_P, s_V, \pi) \leftarrow SIM(C, x)$

KNOWLEDGE SOUNDNESS

V accepts → P knows 'w'

Def^n: (S, P, V) is knowledge sound for a circuit C if splitting A into 2

$\forall$ poly time adversary $A = (A_0, A_1)$ such that

a) $S(C) \to (s_P, s_V)$,

b) $(x, state) \leftarrow A_0(s_P)$,

c) $\pi \leftarrow A_1(s_P, x, state)$:

A succeeds if $R_x\left[V(s_V, x, \pi) = acc\right] > 1/10^6$

$s_P = pp$, $s_V = vp$

state in A_0 is given to A_1

masquerades as P.

$\exists$ an efficient extractor E (that uses A_1 as black box) s.t.

$S(C) \to (s_P, s_V)$

$(x, state) \leftarrow A_0(s_P)$,

$w \leftarrow E^{A_1(s_P, x, state)}(s_P, x)$: $P_R\left(C(x, w) = 0\right] > 1$

CRYPTO → FUNC COMM
INFO THEORETIC → IOP
→ SNARK

(1)

FUNC COMM

P — V

$t \in \mathcal{F}$

P → V: $com_f := commit(f, r)$

V → P: $x \in X$

P → V: $y \in Y$, proof π

V: accept / rej

1) $f(x) = y$
2) $t \in \mathcal{T}$
3) $commit(gp, t, r) = com_f$

FUNC comm scheme for $\mathcal{F}$

(2)

- $setup(\lambda) \to gp$
- $commit(gp, f, r) \to com_f$
- $eval(P, V)$: for given com_f, $x \in X$, $y \in Y$

  $P(gp, f, r, x, y) \to$ short proof π

  $V(gp, com_f, x, y, \pi) \to$ acc/rej

SNARK for the relation eval

f, r = witness

com_f, x, y = public statement

(3)

3 examples of $\mathcal{F}$ family

- Poly nomial comm.

  $f(x) \in \mathbb{F}_p^{\leq d}[x]$

  (univariate poly of degree at most d)

  $3x_2^2 + x_3$

- Multilinear comms.

  $f(x) \in \mathbb{F}_p^{\leq 1}[x_1, \dots, x_k]$ (linear in each x_i)

  $f_1(\cdot) = x_1 x_3 + x_1 x_4 x_5 + x_7$ ✓

  $f_2(\cdot) = x_1^2 x_3$ ✗

- Linear commitments (inner product)

  $f_{\vec{v}}(\vec{u}) = \langle \vec{u}, \vec{v} \rangle = \sum_{i=1}^{n} u_i v_i$

(4)

Linear COMM → Multi → Poly (given this, we can build this)

Use examples.

(5)

PCS (Polynomial Comm scheme)

- $\mathcal{F} \in \mathbb{F}_p^{\leq d}[x]$
- Prover commits to f in $\mathcal{F}$, and later can prove that $v = f(u)$ for public $u, v \in \mathbb{F}_p$
- Examples.
  - + Bulletproof (ECurve)
  - + KZG'10 (bilinear group)

Proof time & Verifier time should be $O_\lambda(\log d)$

as $x \to x^2$

- Trivial example.

$f = a_0 + a_1 x + a_2 x^2 \dots a_d x^d$

$com(f, r) := H((a_0 \dots a_d); r)$ — COM → V

V → P: u

$\pi = ((a_0, a_1 \dots a_d); r)$ — π, v → V

V: if $[(f(u) = v) \wedge (H(\pi) = com_f)]$: accept

Problem about π, Verif time is linear in degree of polynomial

Next page

PROPs of a polynomial

(1)

Non zero $f \in \mathbb{F}_p^{\,d}[x]$

$$r \xleftarrow{R} \mathbb{F}_p : \Pr[f(r)=0] \le \frac{d}{p} \quad -(*)$$

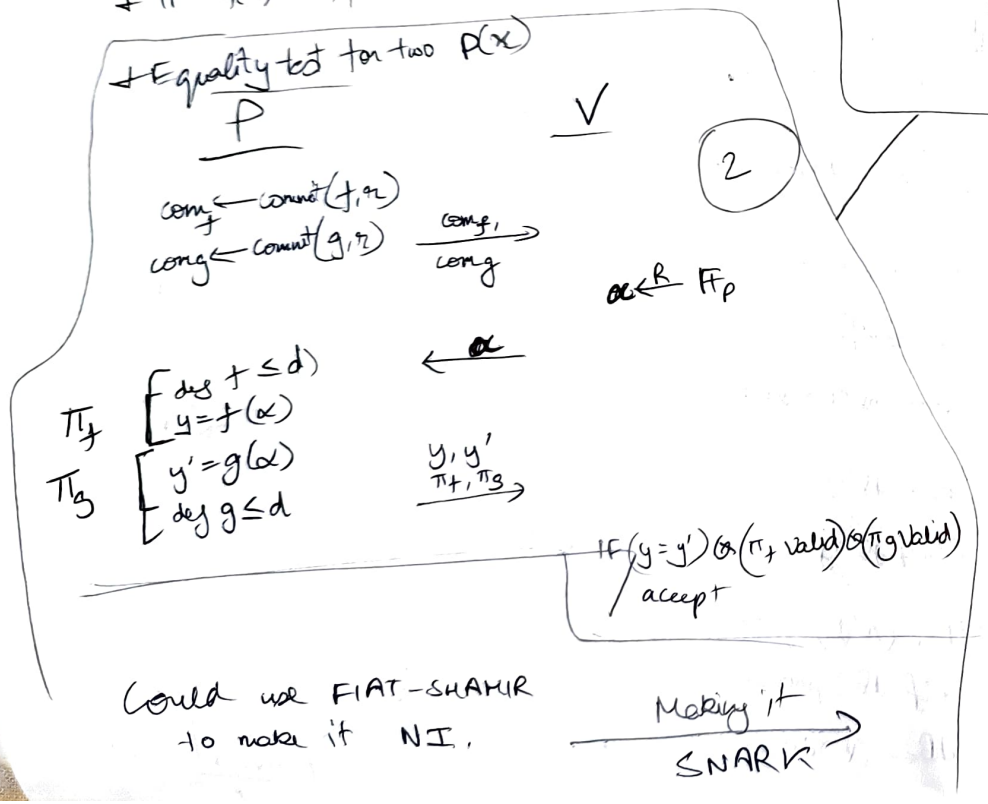
→ that many roots

→ field is small then repeated roots

+ Use above to check if f is a zero $p(x)$
+ sample 'r', check $f(r) \overset{?}{=} 0$
+ IF $f(r)=0$, f is a zero $p(x)$ with high prob

SZDL lemma : $(*)$ also holds for multivariate poly f, where d is the total degree of f.

+ Equality test for two $p(x)$



Could use FIAT-SHAMIR to make it NI.

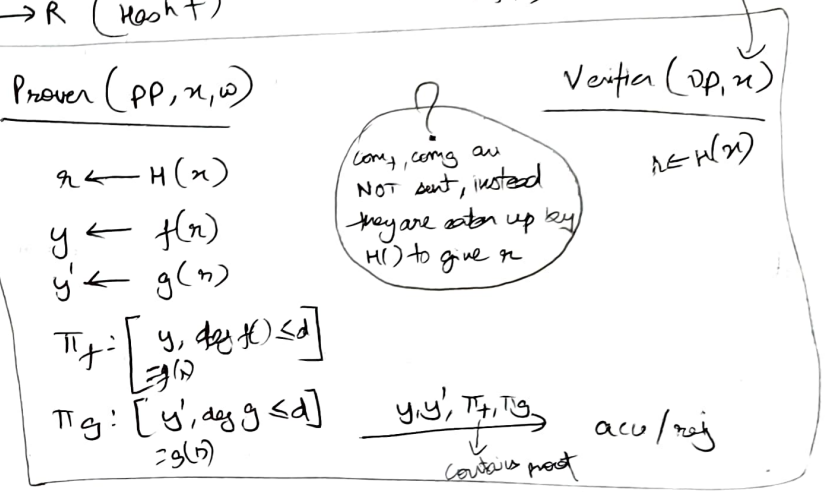
Making it SNARK →

Making Equality test a SNARK

(3)

+ $H : M \to R$ (Hash)

$x = (com_f, com_g)$
$w = (f, g)$



THM: The above protocol is secure if H is modeled as a random oracle & $\frac{d}{p}$ is negligible

Now comes : Polynomial IOP.

Goal : Boost FUNC COMM scheme to build SNARK for general circuit.

Let $C(x,w)$ be some arithmetic circuit
s.t $x \in \mathbb{F}_p^{\,n}$

Pdy-IOP : a proof system that proves $\exists w : C(x,w)=0$ as follows

setup(C) → S_P, [$S_V = (com_{f_0}, com_{f_1}, \ldots, com_{f_s})$]

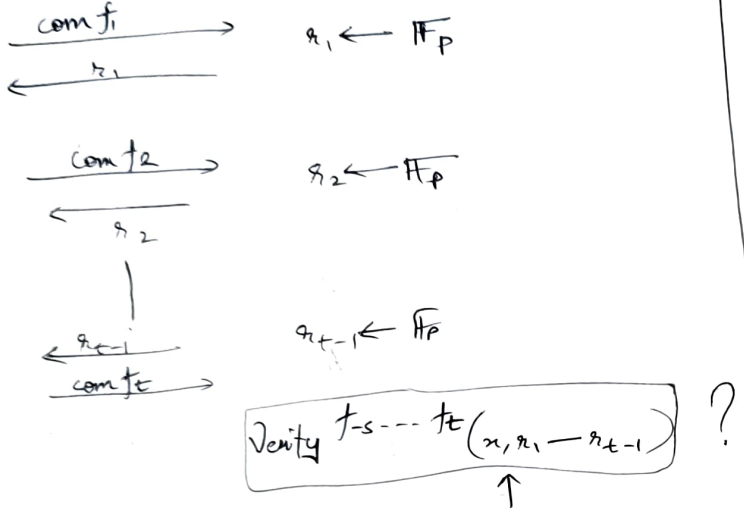
for preprocessing — Commit to use — ve index

later, an interactive process takes place →

F - IOP

Prover (S_P, x, ω) — Verifier (S_V, x)

$f_i \in \mathbb{F}_p^{\leq d}[X]$



Verify $f_1 \dots f_t(x, r_1, \dots r_{t-1})$?

↑ test verify that can evaluate f_i at any point in $\mathbb{F}_p$

O/P s yes/no

r_i's are sent after the prover commits so that prover could not prove a false statt if he had known r_i

PROPS.

- completeness : $\exists \omega : C(x,\omega) = 0$ the $P_r[V \text{ accepts}] = 1$
- Know sound : wt $x \in \mathbb{F}_p^n$, For every p^* that convinces the verifier with prob $\geq 10^{-6}$, there is an efficient extractor E of

$$P_{r_2}\left[E(x, f_1, r_1, \dots f_{t-1}, r_{t-1}, f_t) \to \omega \mid C(x,\omega) = 0 \right] \geq 10^{-6} - \epsilon$$

Resulting SNARK

(t, q) poly IOP $t :=$ # polys committed, $q :=$ # eval queries in verify (usually $t, q \leq 3$)

The SNARK

— Prover — Verifier

t poly comB →

During verify, run PCS eval protocol q times ⇄

But this verification is interactive, but SNARKs should be Non-Interactive, so use FIAT SHAMIR transform

length of SNARK proof : t poly commits + q eval proofs

Verifier time : q × time (eval verify) + time (IOP-verify)

Prover time : t × time (commit) + q × time (prove) + time (IOP-prover)

independent of $|C|$

SZDL lemma

wt $P \neq Q$ be 2 univariate poly's of deg $\leq d$ Th

$P_r[P(r) = Q(r)] \leq d/p$ on random $r \in \mathbb{F}$

Extend to multivariate poly, $P \neq Q$, ℓ variate poly of total degree $\leq d$

$P_r[P(\bar{r}) = Q(\bar{r})] \leq d/p$ on random $\bar{r} \in \mathbb{F}^\ell$

Example of poly-IOP

$C(x, w) = 0 \iff x \subseteq w \subseteq \mathbb{F}_p$

Prover (pp, x, w) — Verifier (vp, x)

$f(z) = \prod_{w \in W} (z - w)$

$g(z) = \prod_{x \in X} (z - x)$

$q(z) = f(z)/g(z)$

$g(z) = \prod_{n \in X} (z - n)$

oracle f →

oracle q →

$r \in_R \mathbb{F}_p$ ←

$y_f = f(r)$

$y_q = q(r)$

$x = \frac{y_f}{y_q} = g(r)$

Accept if

$y_f = x\, y_q$

Knowledge soundness

V accepts $\longrightarrow f = q \cdot g$ whp $\Rightarrow x \subseteq w$

$f = q \cdot g \Rightarrow g(z) \mid f(z) \Rightarrow x \subseteq w$

Extractor (x, f, q, r) : o/p witness w by computing the all roots of $f(z)$

Making it SNARK

- Replace oracle access of f, q by comm$_f$, comm$_q$ (Apply a poly commitment scheme)
- Instead of verifier querying the oracles (not true now) to a random r, prover applies H on $X \longrightarrow r$ $H(x) = r$ to make protocol NI.
- Prover sends the y_i's to π_i's
- V accepts if π_f, π_q are valid to ~~$y_i = x y_j$~~ $y_i = x\, y_j$

Interactive Proofs

- Completeness
- Statistical soundness
- Argument

Soundness vs Knowledge soundness

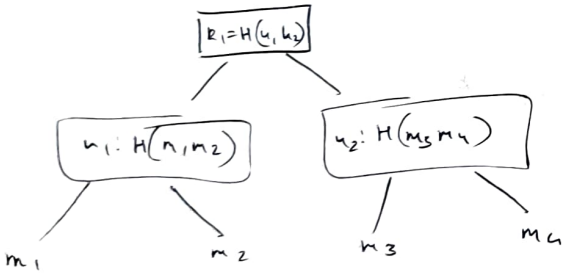
Validity

$\exists w : C(x,w) = 0$

Validity

Prover $| C(x,w) = 0$

Merkle Tree



Problem of designing PCS using merkle Tree

- Prover commits to all evaluations of poly f
- Can't do that

Prover → commits on all eval pts of f → Verifier

Verifier → r → Prover → $f(r)$ → Verifier

- So use low degree of multilinear extension

Def^n : given a $f^n f : \{0,1\}^l \to \mathbb{F}$, a l-variate poly g on $\mathbb{F}$ is said to extend $f(x) = g(x)$ $\forall$ $x \in \{0,1\}^l$

Example. $f : \{0,1\}^2 \to \mathbb{F}$ ← from $\mathbb{F}_2^2$ to $\mathbb{F}^2$

$f(00) = 1$
$f(01) = 2$
$f(10) = 8$
$f(11) = 10$

$\tilde{f} : \mathbb{F}^2 \to \mathbb{F}$

$\tilde{f}(x_1, x_2) = 1(1-x_1)(1-x_2) + 2(1-x_1)(x_2) + 8x_1(1-x_2) + 10\, x_1 x_2$

$\tilde{f}(0,2) = 1(-1) + 2(2) + \cancel{8} = 3$

$\tilde{f}(2,2) = 1 + (-4) + -16 + 40 \cdot$

21

Example of ~~non~~ multilinear extension

$3(\quad) < -x_1^2$ ✗

Efficient evaluation of multilinear extension

Given as input all 2^ℓ evaluations of a given function $f: \{0,1\}^\ell \to \mathbb{F}$, for any point $r \in \mathbb{F}^\ell$, there is a $O(2^\ell)$ time algo for eval $\tilde{f}(r)$

1) $\forall\, w \in \{0,1\}^\ell$, define Lagrange basis.

$$\delta_w(r) = \prod_{i=1}^{\ell} (r_i w_i + (1-r_i)(1-w_i))$$

$\delta_w(r)$ is a basis polynomial

2) $$\tilde{f}(r) = \sum_{w \in \{0,1\}^\ell} f(w)\,\delta_w(r)$$

PROPS

- Build each δ_w req $O(\ell)$ field ops
- Overall TC $= O(\ell 2^\ell)$
- Uses DP.

SUM check protocol.

I/P: V is given an oracle access to an ℓ-variate polynomial g over the field $\mathbb{F}$

Goal: $$C_1 = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_\ell} g(b_1, b_2, \ldots b_\ell)$$

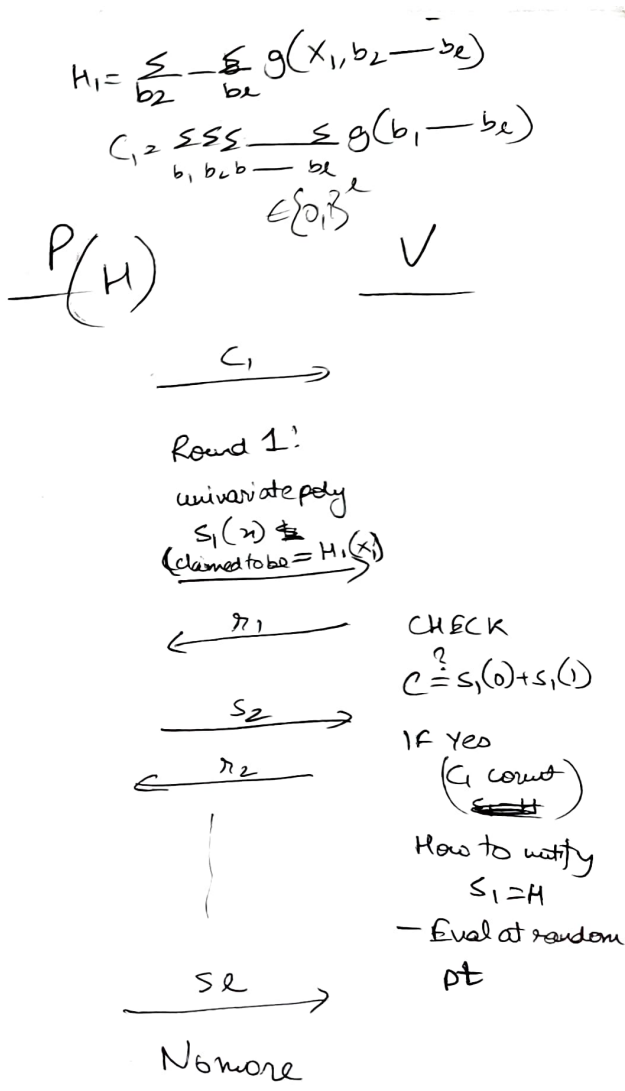
— Computing f is expensive as 2^ℓ possible asgs.

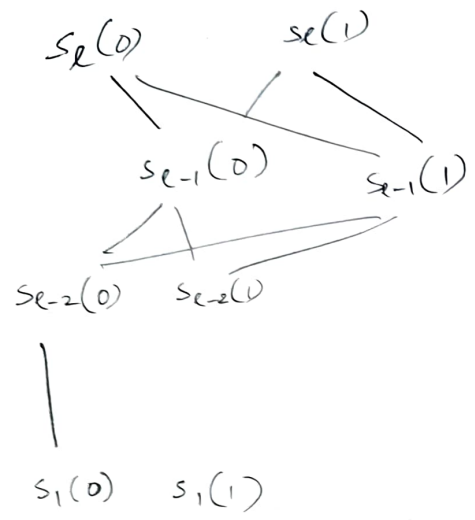
— V delegates comp to UNTrusted prover

— V needs to validate UNTRUSTED proof!

$$H_1 = \sum_{b_2} \cdots \sum_{b_\ell} g(X_1, b_2, \ldots b_\ell)$$

$$C_1 = \sum_{b_1, b_2, \ldots b_\ell \in \{0,1\}^\ell} g(b_1, \ldots b_\ell)$$





Verifier

$C = s_1(0) + s_1(1)$

Round 1 (gets s_1)

Choose r_1, eval $s_1(r_1)$

$s_1(r_1) = s_2(0) + s_2(1)$

$\downarrow$

$b_1 = g_{r_1}$ fixed here

$b_1 \in \{0,1\}$

Round 2 (gets s_2)

(~~V has s_1 already, he can make $s_1 \stackrel{?}{=} s_2$ after combining both~~)

(because r_1 is sent to P)

EXAMPLE

$g(x_1, x_2, x_3) = 2x_1^3 + x_1x_3 + x_2x_3$

$H = \sum_{(b_1,b_2,b_3) \in \{0,1\}^3} g(\,) = 12$

Round 1.

$s_1(x_1) = g_{x_1}(x_1, 00) + g_{x_1}(x_1, 01) + g_{x_1}(x_1, 10) + g_{x_1}(x_1, 11)$

$= 2x_1^3 + 2x_1^3 + x_1$

$= 2x_1^3 + 2x_1^3 + x_1 + 1$

$= \boxed{8x_1^3 + 2x_1 + 1}$

$s_1(0) = 1 \qquad s_1(1) = 11$

$H = s_1(0) + s_1(1)$

r_1 sent to prover.

$r_1 = 2$ (let's say)

$s_2(x_2) = g_2(2, x_2, 0) + g_2(2, x_2, 1)$

$= 16 + 16 + x_2 + 2$

$= 34 + x_2$

$s_2(0) = 34 \qquad s_2(1) = 35$

$s_1(2) = 69 = s_2(0) + s_2(1)$

$A : \{0,1\}^{\log n} \times \{0,1\}^{\log n} \to \mathbb{F}$

If $A_{ij} = b$, then $A(\langle i \rangle, \langle j \rangle) = b$

$\downarrow$

binary rep of i

Application of sum check:

COUNTING Δ's

I/P: $A \in \{0,1\}^{n \times n}$ adj Mx

O/P: $\sum_{i,j,k} A_{ij} A_{jk} A_{ik}$

Fastest known algo runs in $n^{2.37}$

+ V wants to run in $O(n)$
+ V delegate comp to prover
+ P convinces V in linear time about the correctness.

when g is itself a product of small number of multilinear polynomials, then the prover in the Σ-check protocol applied to g can be implemented extremely EFFICIENTLY

Lagrange basis poly

BEST EXAMPLE

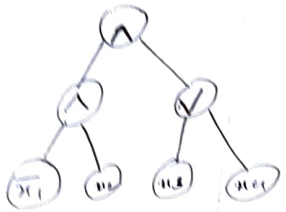
CIRCUIT SAT (IP for SAT problem)

Arithmetization:

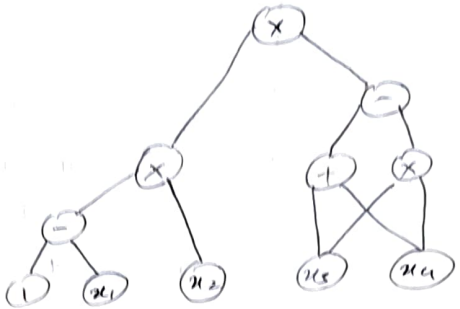
$AND(y,z) \to y \cdot z$

$OR(y,z) \to y + z - y \cdot z$

$\bar{y} \to 1 - y$



Arithmetize →



SAT prob: Find # satisfying assignments.

$$= \sum_{x \in \{0,1\}^n} \prod_i \phi(x_1, x_2, x_3, x_4) \quad \text{(illegible)}$$

Find $\sum_{x \in \{0,1\}^n} \phi(x)$ (illegible)

Count Δ's in a graph. (IP)

$G = (V, E)$ $\quad n = |V|$

I/P: $A \in \{0,1\}^{n \times n}$ $\quad A_{ij} = 1$ iff $(i,j) \in E$

O/P: ~~(i,j,k)~~ unordered triples.

$(i,j,k) \in V \times V \times V$ | i, j, k are all connected each other.

meaning: $(i,j), (j,k), (i,k) \in E$

Q: Where to get g (a low degree polynomial) over all ~~x~~ inputs $\{0,1\}^{\log}$

Ans: Use multilinear extension

Idea: A as a function not a matrix

$f_A : \{0,1\}^{\log n} \times \{0,1\}^{\log n} \to \{0,1\}$

$f_A(x,y) = A_{ij}$ (in multilinear)

$x = \text{Ob } i \quad 1 \le i, j \le n$

$y = \text{Ob } j$

$A_{ij} \in \{0,1\}$

Then # Δ's

$$= \frac{1}{6} \sum_{x,y,z \in \{0,1\}^{\log}} f_A(x,y) \cdot f_A(y,z) \cdot f_A(x,z)$$

3!

$f_A(i,j) = 1$ iff $(i,j) \in E$

A with entries from $\mathbb{F}$

$$\begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \\ 3 & 5 & 7 & 9 \\ 4 & 6 & 8 & 10 \end{bmatrix} \longrightarrow$$

$f_A(0,0,0) = 1$

$f_A(0,0,1) = 3$

$f_A(0,1,0) = 5$

$\vdots$

$f_A(1,1,1) = 10$

n^2 entries in A

n^2 vectors in $\{0,1\}^{\log n} \times \{0,1\}^{\log n}$

$\{0,1\}^{2 \log n} = \{0,1\}^{n^2}$

Entries of A are interpreted as the list of all n^2 evaluations of f_A

Choose $\mathbb{F}$ of size $p \geq 6n^3$

so $6\Delta \in \{0, 1, \ldots, p-1\}$.

as max # Δ's in any graph on n vertices is

$$\binom{n}{3} \leq n^3$$

$$g(x,y,z) = \tilde{f}_A(x,y) \cdot \tilde{f}_A(y,z) \cdot \tilde{f}_A(z,x)$$

$$6\Delta = \sum_{x,y,z \in \{0,1\}^{\log n}} g(x,y,z)$$

so applying sum-check yields an IP computing 6Δ.

EXAMPLE.

Q) 2 vertex graph 

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\tilde{f}_A(a,b) = 1a(1-b) + 1b(1-a) + 0(1-a)(1-b) + 0(ab)$$

$$= a(1-b) + b(1-a)$$

$$g(x,y,z) = \underbrace{\left[x(1-y) + y(1-x)\right]}_{f_A(x,y)} \times \underbrace{\left[y(1-z) + z(1-y)\right]}_{f_A(y,z)} \times \underbrace{\left[x(1-z) + z(1-x)\right]}_{f_A(z,x)}$$

KZG polynomial commitment scheme

Group $G := \{0, G, 2G, 3G, \ldots (p-1)G\}$ of order p

Setup $(\lambda) \to gp$

- Sample $\alpha \in_R \mathbb{F}_p$
- $gp = (H_0 = G, H_1 = \alpha \cdot G, \ldots H_d = \alpha^d G) \in G^{d+1}$
- delete α (Trusted setup) ~~Adversary~~ ~~Prover~~ with α, someone can prove false claims

COMMIT $(gp, f) \to com_f := f(\alpha) \cdot G \in G$

- $f(x) = f_0 + f_1 x + \ldots f_d (x^d) \Rightarrow com_f = f_0 \cdot H_0 + \ldots f_d \cdot H_d$
  $= f_0 G + f_1 G\alpha + \ldots f_d G \alpha^d$
  $= G_1 [f_0 + f_1 \alpha + \ldots f_d \alpha^d]$
  $= G_1 f(\alpha)$
- α is unknown to prover but $H_0, \ldots H_d$ are known

so $com_f = f_0 H_0 + f_1 H_1 \ldots f_d H_d$

α unknown, then how to CALC?

small enough

BINDING commitment but no hiding as ~~NO~~ 'x' involved in $COM(f)$.

eval: $Prover(gp, f, u, v)$ — Goal: prove $f(u) = v$

$Verifier(gp, com_f, u, v)$

$f(u) = v \Leftrightarrow u$ is root of $\hat{f} = f - v \Leftrightarrow (x-u) \mid \hat{f}$

$\Leftrightarrow q \in \mathbb{F}_p[x] \mid q(x) \cdot (x-u) = f(x) - v$

Prover

compute $q(x)$ & com_q $\xrightarrow{\pi := com_q \in G_1}$ (size independent of d)

Constant work

Verifier

acc if $(\alpha - u) \cdot com_q = com_f - v \cdot G_1$

$(q_0 G + q_1 G\alpha \ldots)$

(not known but uses pairing on G_1, H_1 from gp)

$H_1 = \alpha G$

(G_1, H_1) —

PROS

- k-variate polys
- (eval proof size is k group elements)
- Batch proof

$com_1, com_2 \ldots com_n$ — can be combined

CONS

- Trusted setup
- gp is $O(d) = (G_1, \alpha G_1, \ldots \alpha^d G_1)$

So we go to DORY.

DORY PCS

+ Transparent setup (No 'x' secret)
+ comm is a single group element (independent of degree d)
+ eval proof size for $f \in \mathbb{F}_p^{\leq d}[x] = O(\log d)$ group elements
+ eval verify time is $O(\log d)$
  Prover time $O(d)$

IOP

IP where, in each round the V is not forced to read the P's entire message but rather is given query access to it, meaning it can choose to look at any desired symbol of ~~message~~ message at the cost of a single query.

This enables IOP verifier to run in time sub-linear in the total proof length (ie $\sum |m_i|$ $|m_i|$ length of message m_i msg> by prover during IOP

Proof gadgets for univariate polynomials.

- let $\Omega \subseteq \mathbb{F}_p$ $|\Omega| = k$
- let $f \in \mathbb{F}_p^{\leq d}[x]$ ($d \geq k$)
- V has comm
- Construct poly IOP to the following tasks:
  1) ZeroTest: Prove that $f(a) = 0 \quad \forall a \in \Omega$
  2) Sum check: " " $\sum_{a \in \Omega} f(a) = 0$
  3) Prod check: " " $\prod_{a \in \Omega} f(a) = 1$

[Stand as ~~equality test~~ ~~SNARK~~ poly IOP]

Vanishing poly. on Ω

$$Z_\Omega(x) := \prod_{a \in \Omega} (x - a), \quad \deg(Z_\Omega) = k$$

let $\omega \in \mathbb{F}_p$ be k^{th} primitive root of unity

- If $\Omega = \{1, \omega, \omega^2, \dots, \omega^{k-1}\} \subseteq \mathbb{F}_p$, then $Z_\Omega(x) = x^k - 1$
- For $r \in \mathbb{F}_p$, eval $Z_\Omega(r)$ requires at most $2 \log k$ field operations.

ZERO test

let $\Omega = \{1, \omega, \dots, \omega^{k-1}\} \subseteq \mathbb{F}_p$

Prover P | V.

$f(a) = 0 \ \forall a$

comm → Goal: Need to convince V that $f(a) = 0 \ \forall a$

$Z_\Omega(x)$ (can built using Ω)

$Z_\Omega(x) = x^k - 1$

$q(x) = \frac{f(x)}{Z_\Omega(x)}$

comm(q)

f, ORACLE q access

$r \xleftarrow{R} \mathbb{F}_p$

Acc if $f(r) = q(r) \cdot Z_\Omega(r)$