POLYNOMIAL COMMITMENTS

AND WHERE TO FIND THEM

ARANTXA ZAPICO - ETHEREUM FOUNDATION

FOUNDATIONS AND APPLICATIONS OF ZERO-KNOWLEDGE PROOFS WORKSHOP
2-6 SEPTEMBER, ICMS. EDINBURGH.

THIS TALK

WHAT ARE POLYNOMIAL COMMITMENT SCHEMES (R)

THIS TALK

WHAT ARE POLYNOMIAL COMMITMENT SCHEMES (R)

↓

WHERE TO FIND THEM (R)

THIS TALK

WHAT ARE POLYNOMIAL COMMITMENT SCHEMES (R)

↓

WHERE TO FIND THEM (R)

↓

IMPORTANCE OF KZG (C)

THIS TALK

WHAT ARE POLYNOMIAL COMMITMENT SCHEMES (R)

↓

WHERE TO FIND THEM (R)

↓

IMPORTANCE OF KZG (C)

↓

KZG (!)

THIS TALK

WHAT ARE POLYNOMIAL COMMITMENT SCHEMES (R)

↓

WHERE TO FIND THEM (R)

↓

IMPORTANCE OF KZG (C)

↓

KZG (!)

↓

IMPORTANCE OF KZG: KZG TO BUILD SNARKs (C!)

THIS TALK

WHAT ARE POLYNOMIAL COMMITMENT SCHEMES (R)

↓

WHERE TO FIND THEM (R)

↓

IMPORTANCE OF KZG (C)

↓

KZG (!)

↓

IMPORTANCE OF KZG: KZG TO BUILD SNARKs (C!)

↓

EXAMPLE (?)

WHAT ARE POLYNOMIAL COMMITMENTS? [KZG10]

A COMMITMENT SCHEME

For a message space F[X]

A COMMITMENT SCHEME

For a message space F[X]

$\text{pp} \leftarrow \textbf{Setup}(1^\lambda, d)$

A COMMITMENT SCHEME

For a message space F[X]

$\text{pp} \leftarrow \textbf{Setup}(1^\lambda, d)$

$\text{com} \leftarrow \textbf{Commit}(\text{pp}, f(X))$

A COMMITMENT SCHEME

For a message space F[X]

pp $\leftarrow$ **Setup(**$1^\lambda, d$**)**

com $\leftarrow$ **Commit(**pp, $f(X)$**)**

$(\pi, s) \leftarrow$ **Open(**pp, com, $f(X)$, α**), for** $f(\alpha) = s$

A COMMITMENT SCHEME

For a message space F[X]

pp $\leftarrow$ **Setup(**$1^\lambda, d$**)**

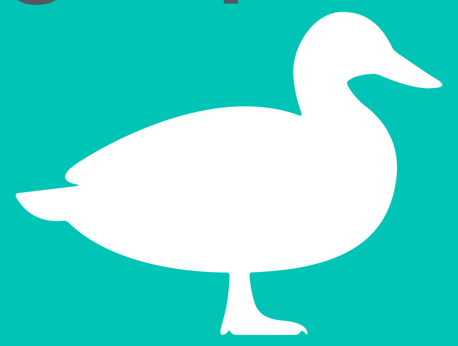
com $\leftarrow$ **Commit(**pp, $f(X)$**)**

$(\pi, s) \leftarrow$ **Open(**pp, com, $f(X), \alpha$**), for** $f(\alpha) = s$

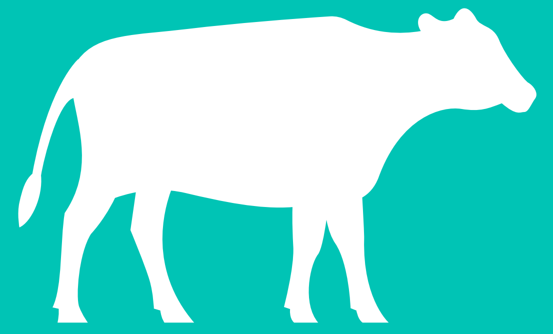
$1/0 \leftarrow$ **Verify(**pp, com, α, s, π**)**

A COMMITMENT SCHEME

For a message space F[X]



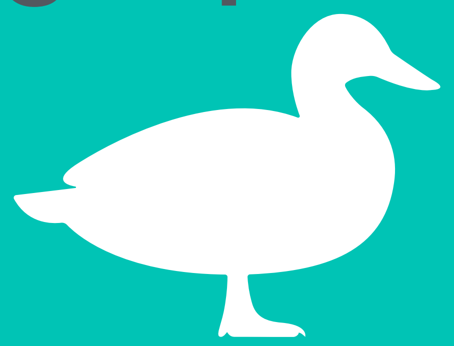
Prover



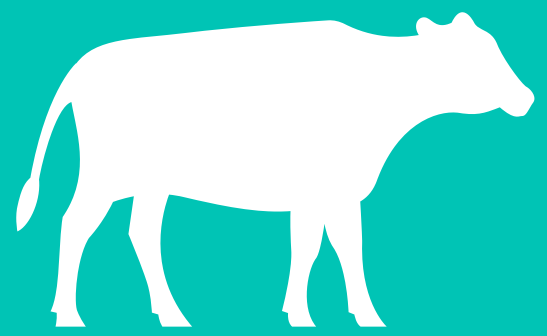
Verifier

A COMMITMENT SCHEME

For a message space F[X]



Prover



Verifier

$\mathsf{pp} \leftarrow \textbf{Setup}(1^{\lambda}, d)$

$\mathsf{com} \leftarrow \textbf{Commit}(\mathsf{pp}, f(X))$

$(\pi, s) \leftarrow \textbf{Open}(\mathsf{pp}, \mathsf{com}, f(X), \alpha)$, **for** $f(\alpha) = s$

$1/0 \leftarrow \textbf{Verify}(\mathsf{pp}, \mathsf{com}, \alpha, s, \pi)$

SECURITY PROPERTIES

Completeness:

SECURITY PROPERTIES

Completeness:

$$Pr\left[\mathsf{Verify}(\mathsf{pp}, \mathsf{com}, \pi, \alpha, s) = 1; \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ \mathsf{com} \leftarrow \mathsf{Commit}(\mathsf{pp}, f(X)) \\ (\pi, s) \leftarrow \mathsf{Open}(\mathsf{pp}, \mathsf{com}, f(X), \alpha) \end{array}\right] = 1$$

SECURITY PROPERTIES

Evaluation Binding

SECURITY PROPERTIES

Evaluation Binding

$$Pr\left[\begin{matrix}\mathsf{Verify}(\mathsf{pp},\mathsf{com},\pi_1,\alpha,s_1)=1\ \wedge\\ \mathsf{Verify}(\mathsf{pp},\mathsf{com},\pi_2,\alpha,s_2)=1\ \wedge\\ s_1\neq s_2\end{matrix}\ ;\ \begin{matrix}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,d)\\ (\mathsf{com},\alpha,(\pi_1,\pi_2,s_1,s_2))\leftarrow\mathscr{A}(\mathsf{pp})\end{matrix}\right]\leq negl(\lambda)$$

SECURITY PROPERTIES

Hiding

SECURITY PROPERTIES

Hiding

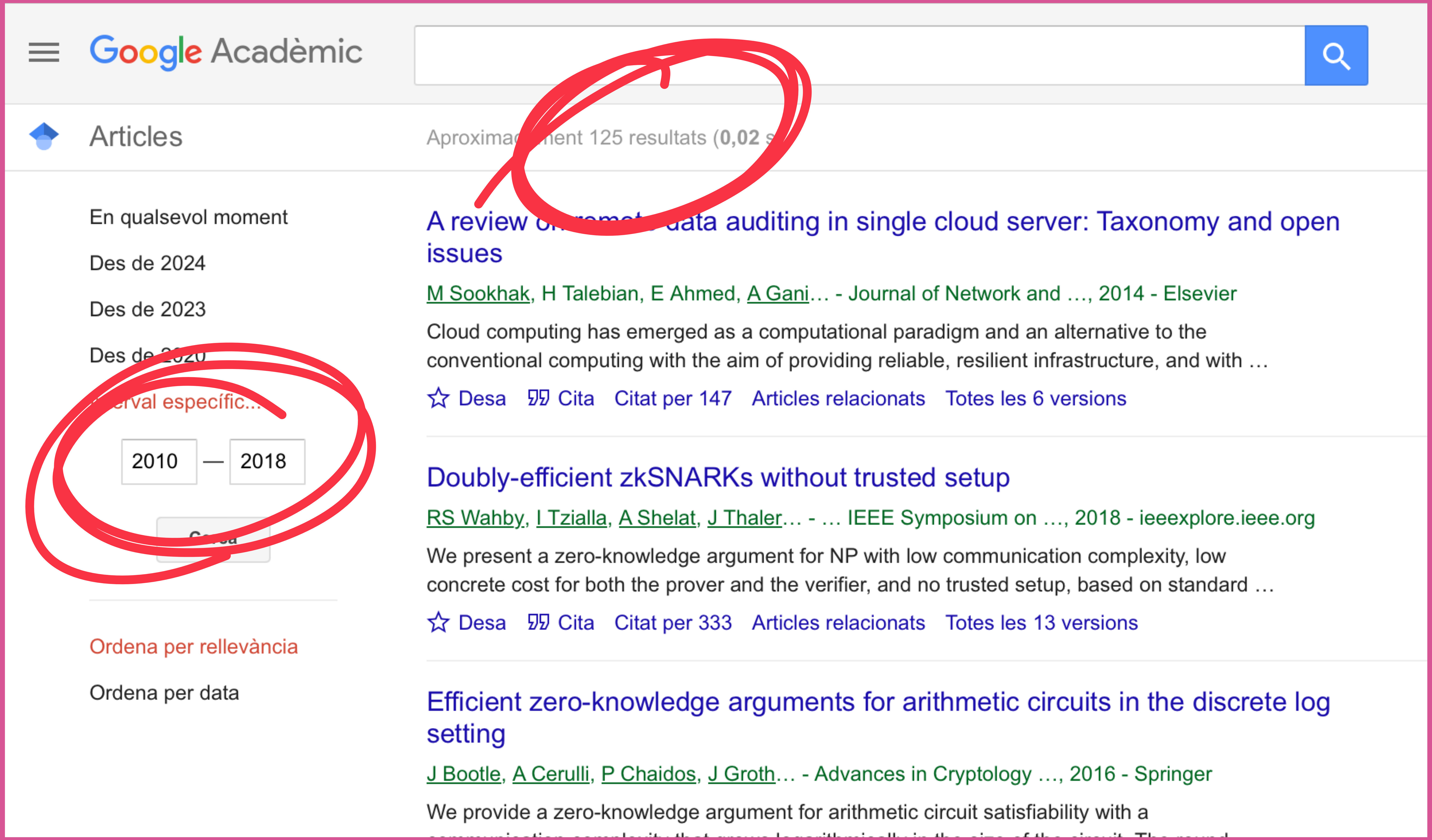
$$Pr\left[\mathscr{A}(\mathsf{pp}, \mathsf{com}, \alpha, (\pi, s)_{\mathsf{sim}}) = 1; \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ \mathsf{com} \leftarrow \mathsf{Commit}(\mathsf{pp}, f(X)) \\ (\pi, s)_{\mathsf{sim}} \leftarrow \mathsf{Sim}(\mathsf{pp}, \mathsf{com}, \alpha) \end{array}\right] \approx$$

$$Pr\left[\mathscr{A}(\mathsf{pp}, \mathsf{com}, \alpha, \pi, s) = 1; \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ \mathsf{com} \leftarrow \mathsf{Commit}(\mathsf{pp}, f(X)) \\ (\pi, s) \leftarrow \mathsf{Open}(\mathsf{pp}, \mathsf{com}, f(X), \alpha) \end{array}\right]$$

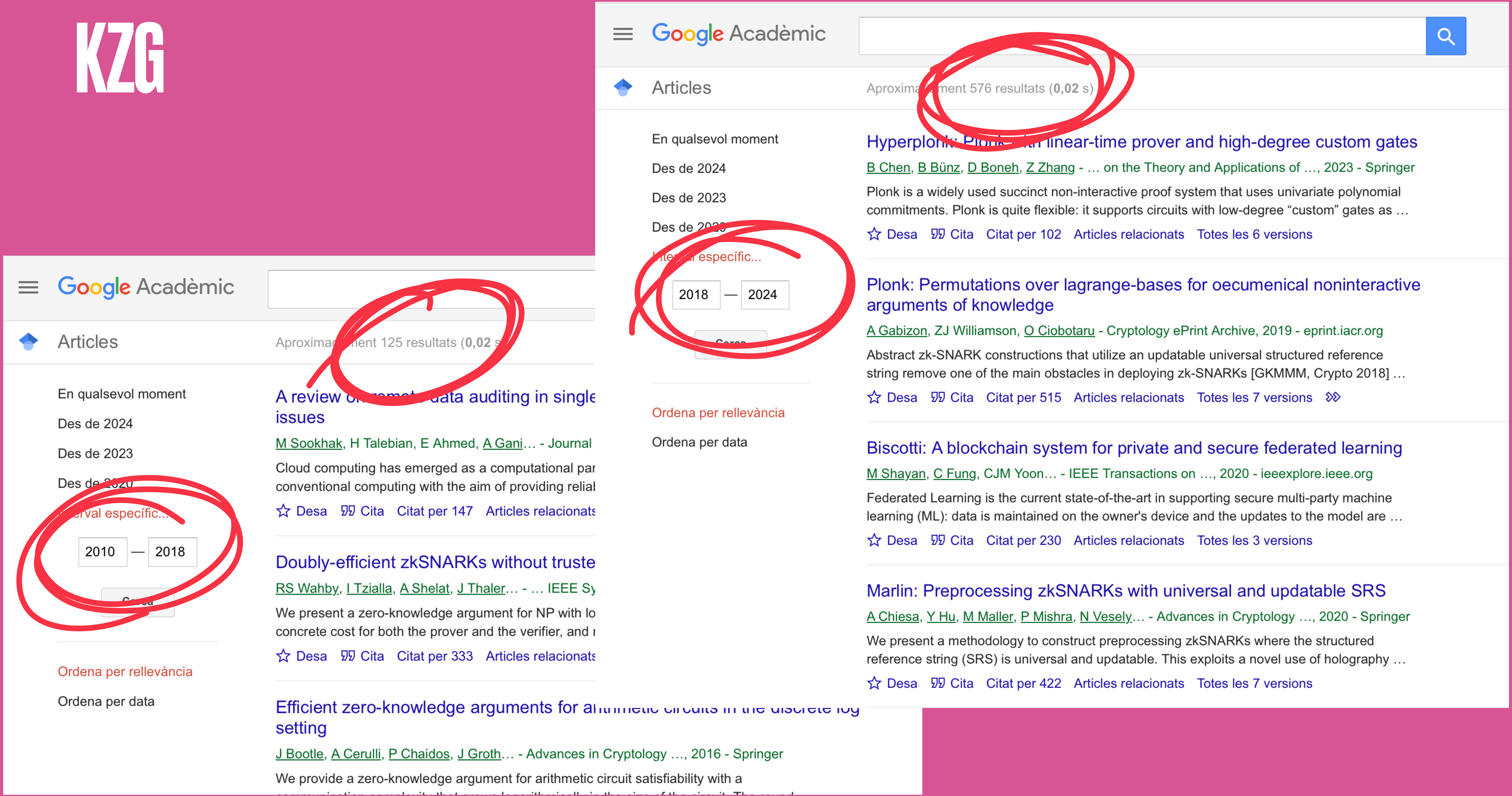
WHERE TO FIND THEM

EVERYWHERE ON EPRINT

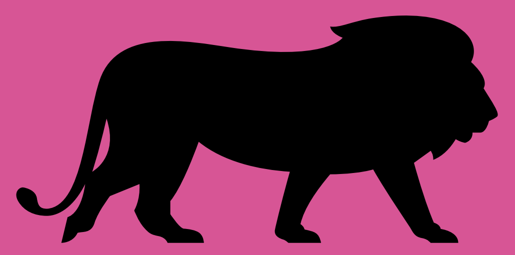
KZG


Google Acadèmic
Articles
Aproximadament 125 resultats (0,02 s)
En qualsevol moment
Des de 2024
Des de 2023
Interval específic...
2010 — 2018
Ordena per rellevància
Ordena per data
data auditing in single cloud server: Taxonomy and open issues
M Sookhak, H Talebian, E Ahmed, A Gani… - Journal of Network and …, 2014 - Elsevier
Cloud computing has emerged as a computational paradigm and an alternative to the conventional computing with the aim of providing reliable, resilient infrastructure, and with …
Desa Cita Citat per 147 Articles relacionats Totes les 6 versions
Doubly-efficient zkSNARKs without trusted setup
RS Wahby, I Tzialla, A Shelat, J Thaler… - … IEEE Symposium on …, 2018 - ieeexplore.ieee.org
We present a zero-knowledge argument for NP with low communication complexity, low concrete cost for both the prover and the verifier, and no trusted setup, based on standard …
Desa Cita Citat per 333 Articles relacionats Totes les 13 versions
Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting
J Bootle, A Cerulli, P Chaidos, J Groth… - Advances in Cryptology …, 2016 - Springer
We provide a zero-knowledge argument for arithmetic circuit satisfiability with a

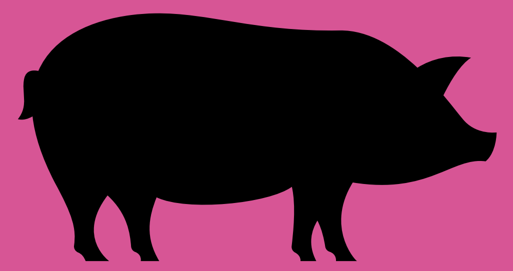
KZG


Google Acadèmic
Articles
Aproximadament 576 resultats (0,02 s)
En qualsevol moment
Des de 2024
Des de 2023
Interval específic...
2018 — 2024
Ordena per rellevància
Ordena per data
Hyperplonk: Plonk with linear-time prover and high-degree custom gates
B Chen, B Bünz, D Boneh, Z Zhang - … on the Theory and Applications of …, 2023 - Springer
Plonk is a widely used succinct non-interactive proof system that uses univariate polynomial commitments. Plonk is quite flexible: it supports circuits with low-degree "custom" gates as …
Desa Cita Citat per 102 Articles relacionats Totes les 6 versions
Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge
A Gabizon, ZJ Williamson, O Ciobotaru - Cryptology ePrint Archive, 2019 - eprint.iacr.org
Abstract zk-SNARK constructions that utilize an updatable universal structured reference string remove one of the main obstacles in deploying zk-SNARKs [GKMMM, Crypto 2018] …
Desa Cita Citat per 515 Articles relacionats Totes les 7 versions
Biscotti: A blockchain system for private and secure federated learning
M Shayan, C Fung, CJM Yoon… - IEEE Transactions on …, 2020 - ieeexplore.ieee.org
Federated Learning is the current state-of-the-art in supporting secure multi-party machine learning (ML): data is maintained on the owner's device and the updates to the model are …
Desa Cita Citat per 230 Articles relacionats Totes les 3 versions
Marlin: Preprocessing zkSNARKs with universal and updatable SRS
A Chiesa, Y Hu, M Maller, P Mishra, N Vesely… - Advances in Cryptology …, 2020 - Springer
We present a methodology to construct preprocessing zkSNARKs where the structured reference string (SRS) is universal and updatable. This exploits a novel use of holography …
Desa Cita Citat per 422 Articles relacionats Totes les 7 versions
Google Acadèmic
Articles
Aproximadament 125 resultats (0,02 s)
En qualsevol moment
Des de 2024
Des de 2023
Interval específic...
2010 — 2018
Ordena per rellevància
Ordena per data
A review on remote data auditing in single cloud server: Taxonomy and open issues
M Sookhak, H Talebian, E Ahmed, A Gani… - Journal
Cloud computing has emerged as a computational par
conventional computing with the aim of providing relial
Desa Cita Citat per 147 Articles relacionats
Doubly-efficient zkSNARKs without truste
RS Wahby, I Tzialla, A Shelat, J Thaler… - … IEEE Sy
We present a zero-knowledge argument for NP with lo
concrete cost for both the prover and the verifier, and
Desa Cita Citat per 333 Articles relacionats
Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting
J Bootle, A Cerulli, P Chaidos, J Groth… - Advances in Cryptology …, 2016 - Springer
We provide a zero-knowledge argument for arithmetic circuit satisfiability with a

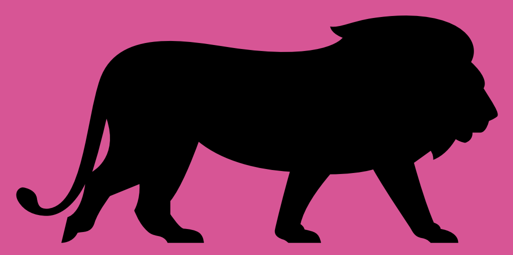
Trusted setup



Transparent Setup

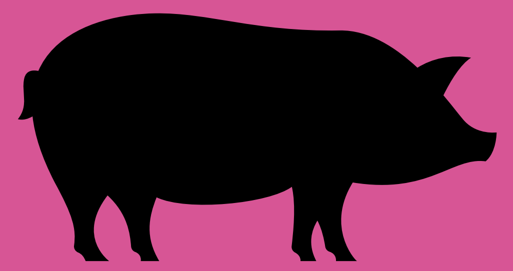


Trusted setup

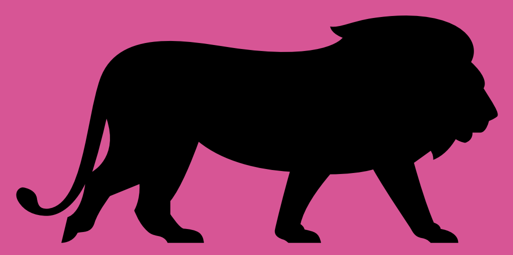


- Fast
- Trust

Transparent Setup

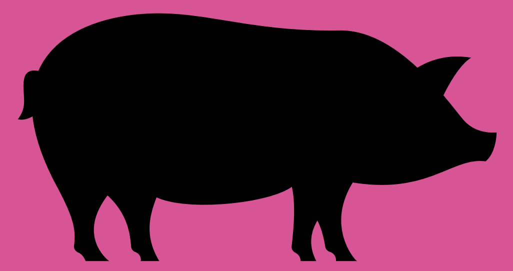


Trusted setup



- Fast
- Trust

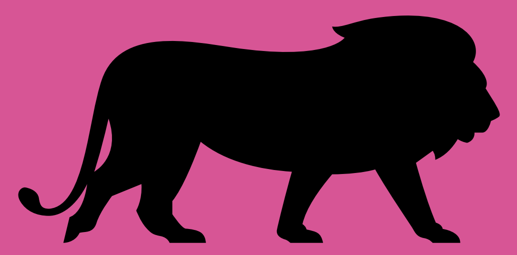
Transparent Setup



- Slow
- Don’t trust

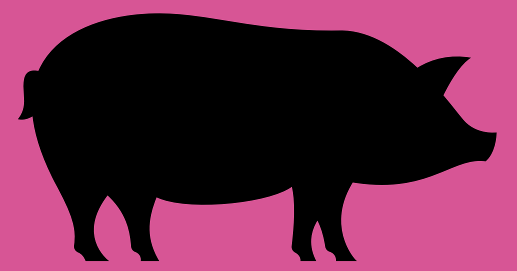
THE UNIVERSAL AND UPDATABLE SRS [GKMMM18]

Trusted setup



- Fast
- Trust

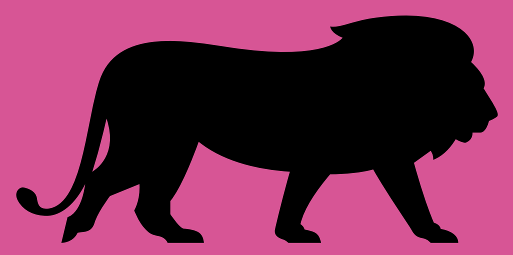
Transparent Setup



- Slow
- Don’t trust

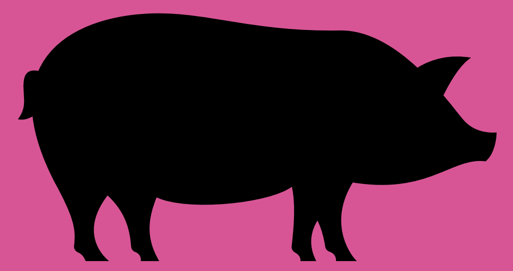
THE UNIVERSAL AND UPDATABLE SRS [GKMMM18]

Trusted setup



- Fast
- Trust

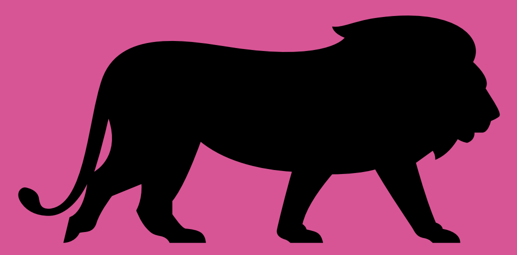
Transparent Setup



- Slow
- Don’t trust

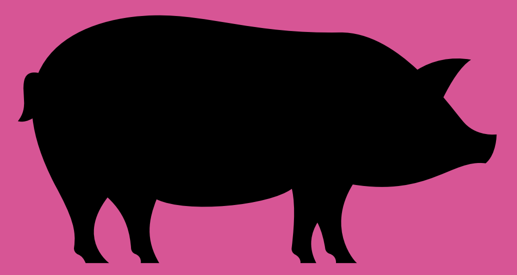
THE UNIVERSAL AND UPDATABLE SRS [GKMMM18]

Trusted setup

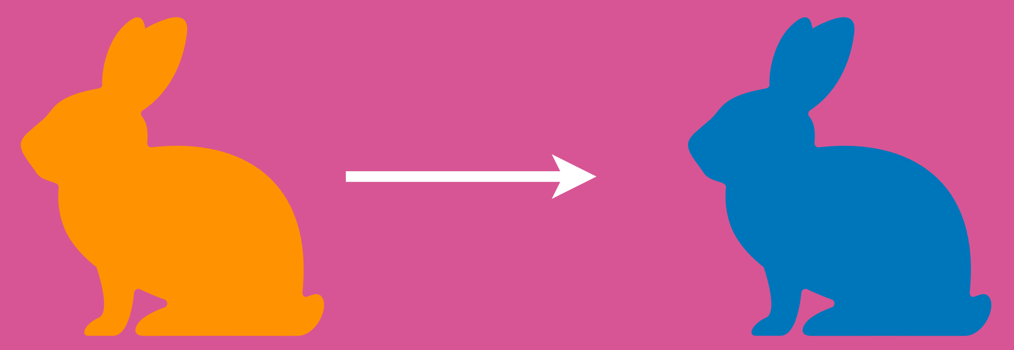


- Fast
- Trust

Transparent Setup

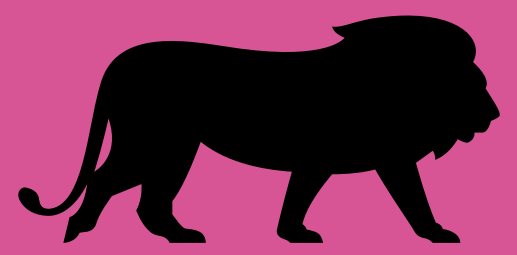


- Slow
- Don’t trust



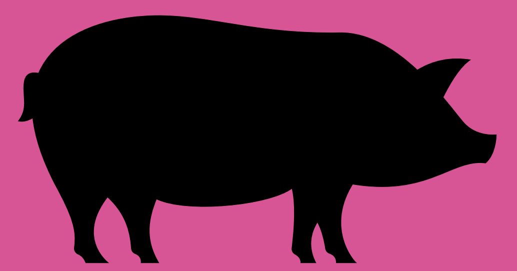
THE UNIVERSAL AND UPDATABLE SRS [GKMMM18]

Trusted setup

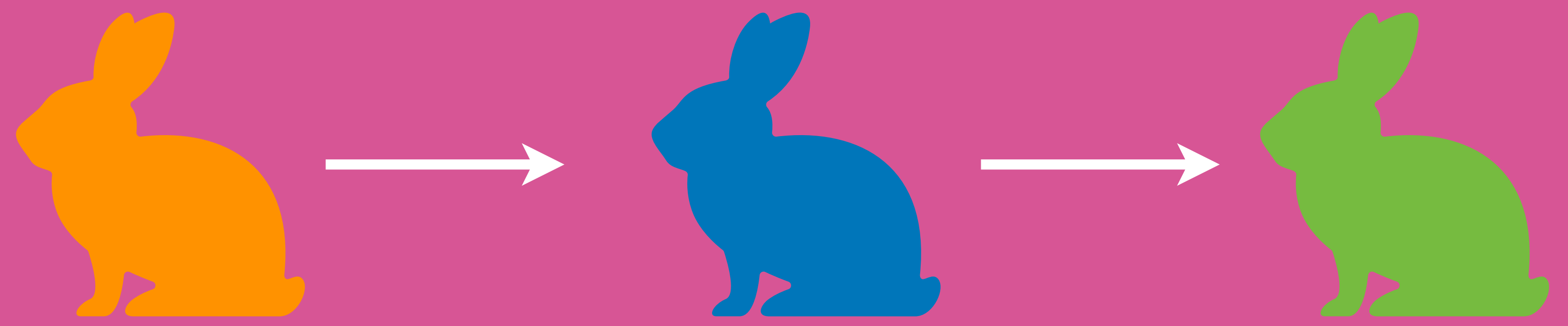


* Fast
* Trust

Transparent Setup

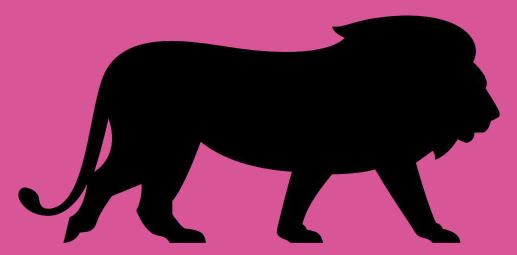


* Slow
* Don’t trust



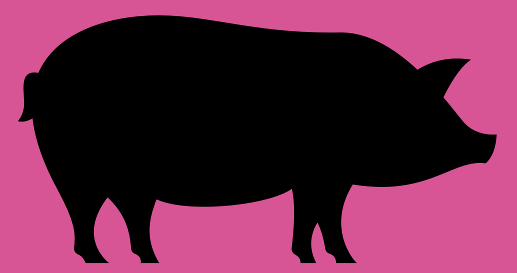
THE UNIVERSAL AND UPDATABLE SRS [GKMMM18]

Trusted setup

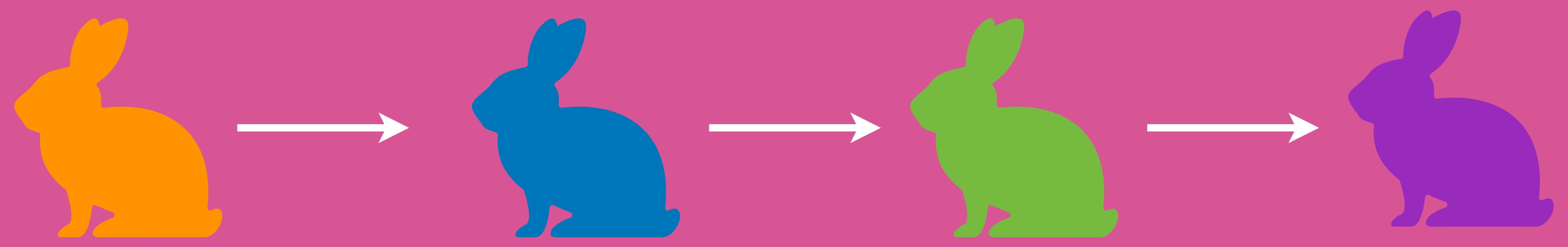


* Fast
* Trust

Transparent Setup

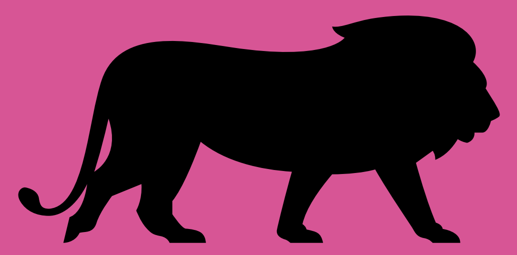


* Slow
* Don't trust



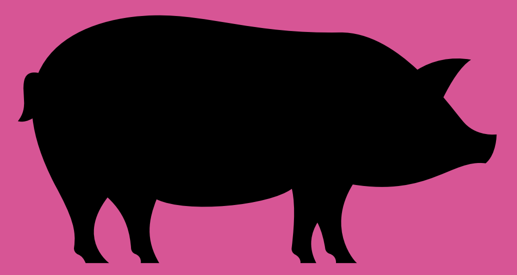
THE UNIVERSAL AND UPDATABLE SRS [GKMMM18]

Trusted setup

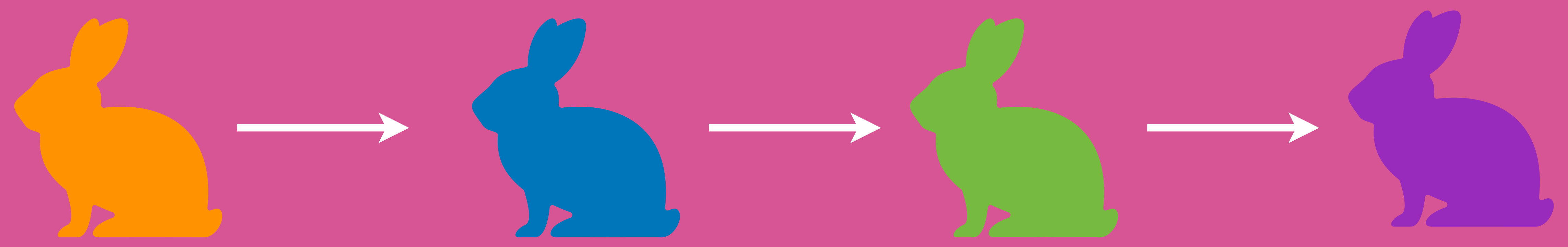


- Fast
- Trust

Transparent Setup



- Slow
- Don’t trust



Can be re-used!!!

SONIC [MBKM19]

SONIC [MBKM19]

Common input: $\mathsf{info} = bp,\ \mathsf{srs},\ s(X,Y),\ k(Y),\ e(g,h^{\alpha})$
Prover's input: $\mathbf{a}, \mathbf{b}, \mathbf{c}$

$\mathsf{zkP}_1(\mathsf{info}, \mathbf{a}, \mathbf{b}, \mathbf{c}) \mapsto R$:

$c_{n+1}, c_{n+2}, c_{n+3}, c_{n+4} \xleftarrow{\$} \mathbb{F}_p$
$r(X,Y) \leftarrow r(X,Y) + \sum_{i=1}^{4} c_{n+i} X^{-2n-i} Y^{-2n-i}$
$R \leftarrow \mathsf{Commit}(bp, \mathsf{srs}, n, r(X,1))$
send R

$\mathsf{zkV}_1(\mathsf{info}, R) \mapsto y$:

send $y \xleftarrow{\$} \mathbb{F}_p$

$\mathsf{zkP}_2(y) \mapsto T$:

$T \leftarrow \mathsf{Commit}(bp, \mathsf{srs}, d, t(X,y))$
send T

$\mathsf{zkV}_2(T) \mapsto z$:

send $z \xleftarrow{\$} \mathbb{F}_p$

$\mathsf{zkP}_3(z) \mapsto (a, W_a, b, W_b, W_t, s, \mathsf{sc})$:

$(a = r(z,1), W_a) \leftarrow \mathsf{Open}(R, z, r(X,1))$
$(b = r(z,y), W_b) \leftarrow \mathsf{Open}(R, yz, r(X,1))$
$(t = t(z,y), W_t) \leftarrow \mathsf{Open}(T, z, t(X,y)))$
$(s = s(z,y), \mathsf{sc}) \leftarrow \mathsf{scP}(\mathsf{info}, s(X,Y), (z,y))$
send $(a, W_a, b, W_b, W_t, s, \mathsf{sc})$

$\mathsf{zkV}_3(a, W_a, b, W_b, W_t, s, \mathsf{sc}) \mapsto 0/1$:

$t \leftarrow a(b+s) - k(y)$
check $\mathsf{scV}(\mathsf{info}, s(X,Y), (z,y), (s, \mathsf{sc}))$
check $\mathsf{pcV}(bp, \mathsf{srs}, n, R, z, (a, W_a))$
check $\mathsf{pcV}(bp, \mathsf{srs}, n, R, yz, (b, W_b))$
check $\mathsf{pcV}(bp, \mathsf{srs}, d, T, z, (t, W_t))$
return 1 if all checks pass, else return 0

Figure 2: The interactive Sonic protocol to check that the prover knows a valid assignment of the wires in the circuit. The stated algorithms describe the individual steps of each of the parties (e.g., zkV_i describes the i-th step of the verifier given the output of zkP_{i-1}), and both parties are assumed to keep state for the duration of the interaction.

SONIC [MBKM19]

WE CAN BUILD SNARKS FROM KZG

SONIC [MBKM19]

BECAUSE KZG IS EXTRACTABLE

SONIC [MBKM19]

BECAUSE KZG IS EXTRACTABLE

$$Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{srs}, \mathsf{com}, \alpha, \pi, s) = 1 \wedge \\ f(\alpha) \neq s\end{array} ; \begin{array}{c}\mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ (\mathsf{com}) \leftarrow \mathscr{A}(\mathsf{srs}, d) \\ f(X) \leftarrow \mathscr{E}(\mathsf{srs}, \mathsf{com}) \\ (\alpha, \pi, s) \leftarrow \mathscr{A}(\mathsf{srs}, d, \mathsf{com})\end{array}\right] \leq negl(\lambda)$$

ONE SNARK TO RULE THEM ALL: KZG

$\mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, d)$:

srs $\leftarrow$ Setup($1^\lambda, d$): Generate group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$.

srs $\leftarrow$ **Setup**$(1^\lambda, d)$**:** Generate group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$.
Sample $\tau \leftarrow \mathbb{F}$.

srs $\leftarrow$ Setup($1^\lambda, d$): Generate group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathscr{G}(1^\lambda)$.
Sample $\tau \leftarrow \mathbb{F}$. Output:

$$\text{srs} := \{[1]_{1,2}, [\tau]_{1,2}, [\tau^2]_1, [\tau^3]_1, \ldots, [\tau^d]_1\}$$

srs ← Setup($1^\lambda, d$): Generate group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$.
Sample $\tau \leftarrow \mathbb{F}$. Output:

$$\text{srs} := \{[1]_{1,2}, [\tau]_{1,2}, [\tau^2]_1, [\tau^3]_1, \ldots, [\tau^d]_1\}$$

com ← Commit(srs, $f(X)$):

srs $\leftarrow$ Setup($1^\lambda, d$): Generate group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$.
Sample $\tau \leftarrow \mathbb{F}$. Output:

$$\text{srs} := \{[1]_{1,2}, [\tau]_{1,2}, [\tau^2]_1, [\tau^3]_1, \ldots, [\tau^d]_1\}$$

com $\leftarrow$ Commit(srs, $f(X)$): Output com $:= [f(\tau)]_1$

srs $\leftarrow$ Setup($1^\lambda, d$): Generate group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$.
Sample $\tau \leftarrow \mathbb{F}$. Output:

$$\text{srs} := \{[1]_{1,2}, [\tau]_{1,2}, [\tau^2]_1, [\tau^3]_1, \ldots, [\tau^d]_1\}$$

com $\leftarrow$ Commit(srs, $f(X)$): Output com $:= [f(\tau)]_1$

$(\pi, s) \leftarrow$ Open(srs, com, $f(X)$, α):

srs $\leftarrow$ **Setup**($1^\lambda, d$): Generate group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$.
Sample $\tau \leftarrow \mathbb{F}$. Output:

$$\text{srs} := \{[1]_{1,2}, [\tau]_{1,2}, [\tau^2]_1, [\tau^3]_1, \ldots, [\tau^d]_1\}$$

com $\leftarrow$ **Commit**(srs, $f(X)$): Output com $:= [f(\tau)]_1$

$(\pi, s) \leftarrow$ **Open**(srs, com, $f(X)$, α): Calculate $s := f(\alpha)$

srs $\leftarrow$ Setup($1^\lambda, d$): Generate group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$.
Sample $\tau \leftarrow \mathbb{F}$. Output:

$$\text{srs} := \{[1]_{1,2}, [\tau]_{1,2}, [\tau^2]_1, [\tau^3]_1, \ldots, [\tau^d]_1\}$$

com $\leftarrow$ Commit(srs, $f(X)$): Output com $:= [f(\tau)]_1$

$(\pi, s) \leftarrow$ Open(srs, com, $f(X), \alpha$): Calculate $s := f(\alpha)$

Calculate $Q(X) = \dfrac{f(X) - s}{X - \alpha}$.

srs $\leftarrow$ Setup$(1^\lambda, d)$: Generate group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathscr{G}(1^\lambda)$.
Sample $\tau \leftarrow \mathbb{F}$. Output:

$$\text{srs} := \{[1]_{1,2}, [\tau]_{1,2}, [\tau^2]_1, [\tau^3]_1, \ldots, [\tau^d]_1\}$$

com $\leftarrow$ Commit(srs, $f(X)$): Output com $:= [f(\tau)]_1$

$(\pi, s) \leftarrow$ Open(srs, com, $f(X), \alpha$): Calculate $s := f(\alpha)$

Calculate $Q(X) = \dfrac{f(X) - s}{X - \alpha}$.

Output $\pi := [Q(\tau)]_1$

srs $\leftarrow$ Setup($1^\lambda, d$): Generate group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$.
Sample $\tau \leftarrow \mathbb{F}$. Output:

$$\text{srs} := \{[1]_{1,2}, [\tau]_{1,2}, [\tau^2]_1, [\tau^3]_1, \ldots, [\tau^d]_1\}$$

com $\leftarrow$ Commit(srs, $f(X)$): Output com $:= [f(\tau)]_1$

$(\pi, s) \leftarrow$ Open(srs, com, $f(X), \alpha$): Calculate $s := f(\alpha)$

Calculate $Q(X) = \dfrac{f(X) - s}{X - \alpha}$.

Output $\pi := [Q(\tau)]_1$

Verify(srs, com, α, π, s):

srs $\leftarrow$ Setup($1^\lambda, d$): Generate group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$.
Sample $\tau \leftarrow \mathbb{F}$. Output:

$$\text{srs} := \{[1]_{1,2}, [\tau]_{1,2}, [\tau^2]_1, [\tau^3]_1, \ldots, [\tau^d]_1\}$$

com $\leftarrow$ Commit(srs, $f(X)$): Output com $:= [f(\tau)]_1$

$(\pi, s) \leftarrow$ Open(srs, com, $f(X), \alpha$): Calculate $s := f(\alpha)$

Calculate $Q(X) = \dfrac{f(X) - s}{X - \alpha}$.

Output $\pi := [Q(\tau)]_1$

Verify(srs, com, α, π, s): $e(\text{com} - s, [1]_2) = e([Q(\tau)]_1, [\tau - \alpha]_2)$

COMPLETENESS

COMPLETENESS

$$Pr\left[\mathsf{Verify}(\mathsf{srs},\mathsf{com},\pi,\alpha,s)=1;\quad \begin{array}{r}\mathsf{srs}\leftarrow\mathsf{Setup}(1^\lambda,d)\\ \mathsf{com}\leftarrow\mathsf{Commit}(\mathsf{srs},f(X))\\ (\pi,s)\leftarrow\mathsf{Open}(\mathsf{srs},\mathsf{com},f(X),\alpha)\end{array}\right]=1$$

COMPLETENESS

$$Pr\left[\text{Verify}(\text{srs}, \text{com}, \pi, \alpha, s) = 1; \begin{array}{r} \text{srs} \leftarrow \text{Setup}(1^\lambda, d) \\ \text{com} \leftarrow \text{Commit}(\text{srs}, f(X)) \\ (\pi, s) \leftarrow \text{Open}(\text{srs}, \text{com}, f(X), \alpha) \end{array}\right] = 1$$

If the Prover behaves honestly, then $s = f(\alpha)$.

COMPLETENESS

$$Pr\left[\textsf{Verify}(\textsf{srs}, \textsf{com}, \pi, \alpha, s) = 1; \quad \begin{array}{r} \textsf{srs} \leftarrow \textsf{Setup}(1^\lambda, d) \\ \textsf{com} \leftarrow \textsf{Commit}(\textsf{srs}, f(X)) \\ (\pi, s) \leftarrow \textsf{Open}(\textsf{srs}, \textsf{com}, f(X), \alpha) \end{array}\right] = 1$$

If the Prover behaves honestly, then $s = f(\alpha)$.
then, $f(\alpha) - s = 0$, or what is the same,
$f(X) - s$ has a root in $X = \alpha$

COMPLETENESS

$$Pr\left[\text{Verify}(\text{srs}, \text{com}, \pi, \alpha, s) = 1; \quad \begin{array}{r} \text{srs} \leftarrow \text{Setup}(1^\lambda, d) \\ \text{com} \leftarrow \text{Commit}(\text{srs}, f(X)) \\ (\pi, s) \leftarrow \text{Open}(\text{srs}, \text{com}, f(X), \alpha) \end{array}\right] = 1$$

If the Prover behaves honestly, then $s = f(\alpha)$.
Then, $f(\alpha) - s = 0$, or what is the same,
$f(X) - s$ has a root in $X = \alpha$.
Then, it must be the case that $(X - \alpha) \mid f(X) - s$.

COMPLETENESS

$$Pr\left[\text{Verify}(\text{srs}, \text{com}, \pi, \alpha, s) = 1; \quad \begin{array}{r} \text{srs} \leftarrow \text{Setup}(1^\lambda, d) \\ \text{com} \leftarrow \text{Commit}(\text{srs}, f(X)) \\ (\pi, s) \leftarrow \text{Open}(\text{srs}, \text{com}, f(X), \alpha) \end{array}\right] = 1$$

If the Prover behaves honestly, then $s = f(\alpha)$. Then, $f(\alpha) - s = 0$, or what is the same, $f(x) - s$ has a root in $x = \alpha$. Then, it must be the case that $(x - \alpha) \mid f(x) - s$. Thus, there exists $Q(x)$ s.t.

$$f(x) - s = (x - \alpha)\, Q(x).$$

COMPLETENESS

$$Pr\left[\text{Verify}(\text{srs}, \text{com}, \pi, \alpha, s) = 1; \quad \begin{array}{r} \text{srs} \leftarrow \text{Setup}(1^\lambda, d) \\ \text{com} \leftarrow \text{Commit}(\text{srs}, f(X)) \\ (\pi, s) \leftarrow \text{Open}(\text{srs}, \text{com}, f(X), \alpha) \end{array}\right] = 1$$

If the Prover behaves honestly, then $s = f(\alpha)$.
Then, $f(\alpha) - s = 0$, or what is the same,
$f(x) - s$ has a root in $x = \alpha$.
Then, it must be the case that $(x - \alpha) \mid f(x) - s$.
Thus, there exists $Q(x)$ s.t.
$f(x) - s = (x - \alpha)\, Q(x)$. By definition of bl-map,
$\text{com} = [f(\tau)]_1$, $\pi = [Q(\tau)]_1$

COMPLETENESS

$$Pr\left[\text{Verify}(\text{srs}, \text{com}, \pi, \alpha, s) = 1; \begin{array}{r} \text{srs} \leftarrow \text{Setup}(1^\lambda, d) \\ \text{com} \leftarrow \text{Commit}(\text{srs}, f(X)) \\ (\pi, s) \leftarrow \text{Open}(\text{srs}, \text{com}, f(X), \alpha) \end{array}\right] = 1$$

If the Prover behaves honestly, then $s = f(\alpha)$. then, $f(\alpha) - s = 0$, or what is the same, $f(x) - s$ has a root in $X = \alpha$. Then, it must be the case that $(x-\alpha) \mid f(x) - s$. Thus, there exists $Q(x)$ s.t. $f(x) - s = (x-\alpha)\, Q(x)$. By definition of bl-map, $com = [f(\tau)]_1$, $\pi = [Q(\tau)]_1 \Rightarrow e(com - s, [1]_2) = e(\pi, [\tau - \alpha]_2)$

EVALUATION BINDING

EVALUATION BINDING

$$Pr\left[\begin{matrix}\mathsf{Verify}(\mathsf{srs},\mathsf{com},\pi_1,\alpha,s_1)=1\ \wedge\\ \mathsf{Verify}(\mathsf{srs},\mathsf{com},\pi_2,\alpha,s_2)=1\ \wedge\\ s_1\neq s_2\end{matrix};\begin{matrix}\mathsf{srs}\leftarrow\mathsf{Setup}(1^\lambda,d)\\ (\mathsf{com},\alpha,(\pi_1,\pi_2,s_1,s_2))\leftarrow\mathscr{A}(\mathsf{srs})\end{matrix}\right]=1$$

EVALUATION BINDING

$$Pr\left[\begin{matrix}\mathsf{Verify}(\mathsf{srs}, \mathsf{com}, \pi_1, \alpha, s_1) = 1 \ \wedge \\ \mathsf{Verify}(\mathsf{srs}, \mathsf{com}, \pi_2, \alpha, s_2) = 1 \ \wedge \\ s_1 \neq s_2\end{matrix} ; \begin{matrix}\mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ (\mathsf{com}, \alpha, (\pi_1, \pi_2, s_1, s_2)) \leftarrow \mathscr{A}(\mathsf{srs})\end{matrix}\right] = 1$$

Suppose there exist com, α, π_1, π_2, s_1, s_2, such that:

EVALUATION BINDING

$$Pr\left[\begin{matrix}\text{Verify}(\text{srs}, \text{com}, \pi_1, \alpha, s_1) = 1 \wedge \\ \text{Verify}(\text{srs}, \text{com}, \pi_2, \alpha, s_2) = 1 \wedge \\ s_1 \neq s_2\end{matrix} ; \begin{matrix}\text{srs} \leftarrow \text{Setup}(1^\lambda, d) \\ (\text{com}, \alpha, (\pi_1, \pi_2, s_1, s_2)) \leftarrow \mathscr{A}(\text{srs})\end{matrix}\right] = 1$$

Suppose there exist com, α, π_1, π_2, s_1, s_2, such that:

$e(\text{com} - s_1, [1]_2) = e([z-\alpha]_1, \pi_1)$,

$e(\text{com} - s_2, [1]_2) = e([z-\alpha]_1, \pi_2)$ & $s_1 \neq s_2$.

EVALUATION BINDING

$$Pr\left[\begin{matrix}\text{Verify}(\text{srs},\text{com},\pi_1,\alpha,s_1)=1\ \wedge\\ \text{Verify}(\text{srs},\text{com},\pi_2,\alpha,s_2)=1\ \wedge\\ s_1\neq s_2\end{matrix};\ \begin{matrix}\text{srs}\leftarrow\text{Setup}(1^\lambda,d)\\ (\text{com},\alpha,(\pi_1,\pi_2,s_1,s_2))\leftarrow\mathscr{A}(\text{srs})\end{matrix}\right]=1$$

Suppose there exist com, α, π_1, π_2, s_1, s_2, such that:

$e(\text{com}-s_1,[1]_2)=e([\tau-\alpha]_1,\pi_1)$,

$e(\text{com}-s_2,[1]_2)=e([\tau-\alpha]_1,\pi_2)$ & $s_1\neq s_2$.

By substracting:

$e(s_2-s_1,[1]_2)=e([\tau-\alpha]_1,\pi_2-\pi_1)$.

EVALUATION BINDING

$$Pr\left[\begin{array}{c|r} \text{Verify}(\text{srs}, \text{com}, \pi_1, \alpha, s_1) = 1 \wedge & \text{srs} \leftarrow \text{Setup}(1^\lambda, d) \\ \text{Verify}(\text{srs}, \text{com}, \pi_2, \alpha, s_2) = 1 \wedge ; & (\text{com}, \alpha, (\pi_1, \pi_2, s_1, s_2)) \leftarrow \mathscr{A}(\text{srs}) \\ s_1 \neq s_2 & \end{array}\right] = 1$$

Suppose there exist com, α, π_1, π_2, s_1, s_2, such that:

$e(\text{com} - s_1, [1]_2) = e([\tau - \alpha]_1, \pi_1)$,

$e(\text{com} - s_2, [1]_2) = e([\tau - \alpha]_1, \pi_2)$ & $s_1 \neq s_2$.

By substracting:

$e(s_2 - s_1, [1]_2) = e([\tau - \alpha]_1, \pi_2 - \pi_1)$.

Now, we can compute: $\left[\frac{1}{\tau - \alpha}\right]_T = \frac{1}{s_2 - s_1} e([1]_1, \pi_2 - \pi_1)$

Q-BSDH ASSUMPTION

$$\mathsf{Adv}^{\mathsf{q-BDSH}}_{\mathscr{A},\mathscr{G}}(1^\lambda) = Pr\left[c \neq \tau \wedge W = \left[\frac{1}{\tau - c}\right]_T ; \begin{array}{r} \mathscr{G} \leftarrow \mathsf{Gen}(1^\lambda), \tau \leftarrow \mathbb{F} \\ (c, W) \leftarrow \mathscr{A}\Big(\mathscr{G}, d, \{[\tau^i]_{1,2}\}_{i=0}^{d}\Big) \end{array}\right] \leq negl(\lambda)$$

HIDING?

com $\leftarrow$ **Commit(srs, $f(X)$):** Output com $\leftarrow [f(\tau)]_1$

$(\pi, s) \leftarrow$ **Open(srs, com, $f(X)$, α):** Calculate $s \leftarrow f(\alpha)$

Calculate $Q(X) = \dfrac{f(X) - s}{X - \alpha}$, set $\pi \leftarrow [Q(\tau)]_1$

HIDING?

HIDING

HIDING

com $\leftarrow$ Commit(srs, $f(X)$): Output com $\leftarrow [f(\tau)+\alpha\hat{f}(\tau)]_1, \hat{f}(X) \leftarrow \mathbb{F}[X]$

HIDING

com $\leftarrow$ Commit(srs, $f(X)$): Output com $\leftarrow [f(\tau)+\alpha\hat{f}(\tau)]_1, \hat{f}(X) \leftarrow \mathbb{F}[X]$

$(\pi, s, \hat{s}) \leftarrow$ Open(srs, com, $f(X), \alpha$): Calculate $s \leftarrow f(\alpha), \hat{s} \leftarrow \hat{f}(x)$

Calculate $Q(X), \hat{Q}(X)$, set $\pi \leftarrow [Q(\tau)+\alpha\hat{Q}(\tau)]_1$

HIDING

com $\leftarrow$ **Commit(srs,** $f(X)$**):** Output com $\leftarrow [f(\tau)+\alpha\hat{f}(\tau)]_1, \hat{f}(X) \leftarrow \mathbb{F}[X]$

$(\pi, s, \hat{s}) \leftarrow$ **Open(srs, com,** $f(X), \alpha$**):** Calculate $s \leftarrow f(\alpha), \hat{s} \leftarrow \hat{f}(x)$

Calculate $Q(X), \hat{Q}(X)$, set $\pi \leftarrow [Q(\tau)+\alpha\hat{Q}(\tau)]_1$

Verify(srs, com, α, π, s**):** $e(\text{com}-s-\alpha\hat{s}, [1]_2) = e([Q(\tau)]_1, [\tau - \alpha]_2)$

EXTRACTABILITY

$$Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{srs}, \mathsf{com}, \alpha, \pi, s) = 1 \wedge \\ f(\alpha) \neq s\end{array} ; \begin{array}{c}\mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ (\mathsf{com}) \leftarrow \mathscr{A}(\mathsf{srs}, d) \\ f(X) \leftarrow \mathscr{E}(\mathsf{srs}, \mathsf{com}) \\ (\alpha, \pi, s) \leftarrow \mathscr{A}(\mathsf{srs}, d, \mathsf{com})\end{array}\right] \leq negl(\lambda)$$

ALGEBRAIC GROUP MODEL [FKL18]

ALGEBRAIC GROUP MODEL [FKL18]

We assume that the adversary $\mathscr{A}$ is an algebraic algorithm.

ALGEBRAIC GROUP MODEL [FKL18]

We assume that the adversary $\mathscr{A}$ is an algebraic algorithm.

Derives new group elements only by applying the group operations to received group elements.

ALGEBRAIC GROUP MODEL [FKL18]

We assume that the adversary $\mathscr{A}$ is an algebraic algorithm.

Derives new group elements only by applying the group operations to received group elements.

If $[y] \leftarrow \mathscr{A}\left([x_1], \ldots, [x_d]\right)$, then $\mathscr{A}$ must also provide $\vec{z}$ such that $[y] = \sum_{i=1}^{d} z_i[x_i]$

KZG CAN BE USED TO BUILD SNARKS [MBKM19]

Common input: info = bp, srs, $s(X,Y)$, $k(Y)$, $e(g,h^{\alpha})$
Prover's input: $\mathbf{a}, \mathbf{b}, \mathbf{c}$

zkP$_1$(info, $\mathbf{a}, \mathbf{b}, \mathbf{c}$) $\mapsto R$:

$c_{n+1}, c_{n+2}, c_{n+3}, c_{n+4} \xleftarrow{\$} \mathbb{F}_p$
$r(X,Y) \leftarrow r(X,Y) + \sum_{i=1}^{4} c_{n+i} X^{-2n-i} Y^{-2n-i}$
$R \leftarrow$ Commit(bp, srs, n, $r(X,1)$)
send R

zkV$_1$(info, R) $\mapsto y$:

send $y \xleftarrow{\$} \mathbb{F}_p$

zkP$_2$(y) $\mapsto T$:

$T \leftarrow$ Commit(bp, srs, d, $t(X,y)$)
send T

zkV$_2$(T) $\mapsto z$:

send $z \xleftarrow{\$} \mathbb{F}_p$

zkP$_3$(z) $\mapsto (a, W_a, b, W_b, W_t, s, \text{sc})$:

$(a = r(z,1), W_a) \leftarrow$ Open($R, z, r(X,1)$)
$(b = r(z,y), W_b) \leftarrow$ Open($R, yz, r(X,1)$)
$(t = t(z,y), W_t) \leftarrow$ Open($T, z, t(X,y)$)
$(s = s(z,y), \text{sc}) \leftarrow$ scP(info, $s(X,Y)$, (z,y))
send $(a, W_a, b, W_b, W_t, s, \text{sc})$

zkV$_3$($a, W_a, b, W_b, W_t, s, \text{sc}$) $\mapsto 0/1$:

$t \leftarrow a(b+s) - k(y)$
check scV(info, $s(X,Y)$, (z,y), (s, sc))
check pcV(bp, srs, n, R, z, (a, W_a))
check pcV(bp, srs, n, R, yz, (b, W_b))
check pcV(bp, srs, d, T, z, (t, W_t))
return 1 if all checks pass, else return 0

Figure 2: The interactive Sonic protocol to check that the prover knows a valid assignment of the wires in the circuit. The stated algorithms describe the individual steps of each of the parties (e.g., zkV$_i$ describes the i-th step of the verifier given the output of zkP$_{i-1}$), and both parties are assumed to keep state for the duration of the interaction.

WE CAN BUILD ARGUMENTS OF KNOWLEDGE FROM POLYNOMIALS AND THEN COMPILE WITH KZG

WE CAN BUILD ARGUMENTS OF KNOWLEDGE
FROM POLYNOMIALS
AND THEN COMPILE
WITH KZG

THIS IS TRUE FOR ANY POLY COMMITMENT

PIOP + POLY COM = SNARK

PlonK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge

Ariel Gabizon* Aztec
Zachary J. Williamson Aztec
Oana Ciobotaru

February 23, 2024

zk-SNARK constructions t
string remove one of the mai

Spartan: Efficient and general-purpose zkSNARKs without trusted setup

Srinath Setty
Microsoft Research

Abstract

This paper introduces Spartan, a new family of zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) for the rank-1 constraint satisfiability (R1CS), an NP-complete language that generalizes arithmetic circuit satisfiability. A distinctive feature of Spartan is that it offers the first zkSNARKs without trusted

MARLIN: Preprocessing zkSNARKs with Universal and Updatable SRS

Alessandro Chiesa, alexch@berkeley.edu, UC Berkeley
Yuncong Hu, yuncong_hu@berkeley.edu, UC Berkeley
Mary Maller, mary.maller.15@ucl.ac.uk, UCL
Pratyush Mishra, pratyush@berkeley.edu, UC Berkeley
Psi Vesely, psi@ucsd.edu, UCL
Nicholas Ward, npward@berkeley.edu, UC Berkeley

May 27, 2020

Abstract

We present a methodology to construct preprocessing zkSNARKs where the structured reference string (SRS) is universal and updatable. This exploits a novel use of *holography* [Babai et al., STOC 1991], where fast verification is achieved provided the statement being checked is given in encoded form.

We use our methodology to obtain a preprocessing zkSNARK where the SRS has linear size and

Transparent SNARKs from DARK Compilers

Benedikt Bünz[1] benedikt@cs.stanford.edu
Ben Fisch[1] benafisch@gmail.com
Alan Szepieniec[2] alan@nervos.org

[1]Stanford University
[2]Nervos Foundation

Abstract

We construct a new polynomial commitment scheme for univariate and multivariate polynomials over finite fields, with logarithmic size evaluation proofs and verification

LUNAR[CFQR21]

LUNAR[CFQR21]

Theorem 15. *Let* $\mathsf{PHP} = (\mathsf{r}, \mathsf{n}, \mathsf{m}, \mathsf{d}, \mathsf{n_e}, \mathcal{RE}, \mathcal{P}, \mathcal{V})$ *be a non-adaptive public-coin PHP over* $\mathcal{F}$ *and* $\mathcal{R}$, *let* CS^* *be a compiling commitment scheme as in Definition 22 equipped with CP-SNARKs* $\mathsf{CP_{opn}}$ *for* $\mathcal{R}_{\mathsf{opn}}$, $\mathsf{CP_{php}}$ *for a relation* $\mathcal{R}_{\mathsf{php}}$, *and* $\mathsf{CP_{link}}$ *for* $\mathsf{R_{link}}$. *Then we have:*

- *If* PHP *has straight-line extractability, then the scheme* UIA *defined above is a universal commit and prove interactive argument in the SRS model for* $\mathcal{R}'$ *such that:*

$$(\mathsf{R}, \mathsf{x}, (\mathsf{u}_j)_{j\in[\ell]}, \omega) \in \mathcal{R}' \iff (\mathsf{R}, \mathsf{x}, \mathsf{Decode}((\mathsf{u}_j)_{j\in[\ell]}), \omega) \in \mathcal{R}.$$

- *If, for a checker* C, PHP *(resp.* $\mathsf{CP_{php}}$*) is* $(\mathsf{b}+\mathbf{1}, \mathsf{C})$*-bounded honest-verifier zero knowledge (resp. trapdoor-commit* (b, C)*-leaky zero-knowledge), and both* $\mathsf{CP_{opn}}$ *and* $\mathsf{CP_{link}}$ *are trapdoor-commitment zero-knowledge, then* UIA *is trapdoor-commitment honest-verifier zero-knowledge.*

LUNAR[CFQR21]

Theorem 15. *Let* PHP $= (\mathsf{r}, \mathsf{n}, \mathsf{m}, \mathsf{d}, \mathsf{n_e}, \mathcal{RE}, \mathcal{P}, \mathcal{V})$ *be a non-adaptive public-coin PHP over* $\mathcal{F}$ *and* $\mathcal{R}$, *let* CS^* *be a compiling commitment scheme as in Definition 22 equipped with CP-SNARKs* $\mathsf{CP_{opn}}$ *for* $\mathcal{R}_{\mathsf{opn}}$, $\mathsf{CP_{php}}$ *for a relation* $\mathcal{R}_{\mathsf{php}}$, *and* $\mathsf{CP_{link}}$ *for* $\mathsf{R_{link}}$. *Then we have:*

– *If* PHP *has straight-line extractability, then the scheme* UIA *defined above is a universal commit and prove interactive argument in the SRS model for* $\mathcal{R}'$ *such that:*

$$(\mathsf{R}, \mathsf{x}, (\mathsf{u}_j)_{j \in [\ell]}, \omega) \in \mathcal{R}' \iff (\mathsf{R}, \mathsf{x}, \mathsf{Decode}((\mathsf{u}_j)_{j \in [\ell]}), \omega) \in \mathcal{R}.$$

– *If, for a checker* C, PHP *(resp.* $\mathsf{CP_{php}}$*) is* $(\mathsf{b}+1, \mathsf{C})$*-bounded honest-verifier zero knowledge (resp. trapdoor-commit* (b, C)*-leaky zero-knowledge), and both* $\mathsf{CP_{opn}}$ *and* $\mathsf{CP_{link}}$ *are trapdoor-commitment zero-knowledge, then* UIA *is trapdoor-commitment honest-verifier zero-knowledge.*

LUNAR[CFQR21]

Theorem 15. *Let* $\mathsf{PHP} = (\mathsf{r}, \mathsf{n}, \mathsf{m}, \mathsf{d}, \mathsf{n_e}, \mathcal{RE}, \mathcal{P}, \mathcal{V})$ *be a non-adaptive public-coin PHP over* $\mathcal{F}$ *and* $\mathcal{R}$, *let* CS^* *be a compiling commitment scheme as in Definition 22 equipped with CP-SNARKs* $\mathsf{CP_{opn}}$ *for* $\mathcal{R}_{\mathsf{opn}}$, $\mathsf{CP_{php}}$ *for a relation* $\mathcal{R}_{\mathsf{php}}$, *and* $\mathsf{CP_{link}}$ *for* $\mathsf{R_{link}}$. *Then we have:*

– *If* PHP *has straight-line extractability, then the scheme* UIA *defined above is a universal commit and prove interactive argument in the SRS model for* $\mathcal{R}'$ *such that:*

$$(\mathsf{R}, \mathsf{x}, (\mathsf{u}_j)_{j\in[\ell]}, \omega) \in \mathcal{R}' \iff (\mathsf{R}, \mathsf{x}, \mathsf{Decode}((\mathsf{u}_j)_{j\in[\ell]}), \omega) \in \mathcal{R}.$$

– *If, for a checker* C, PHP *(resp.* $\mathsf{CP_{php}}$*) is* $(\mathsf{b}+1, \mathsf{C})$*-bounded honest-verifier zero knowledge (resp. trapdoor-commit* (b, C)*-leaky zero-knowledge), and both* $\mathsf{CP_{opn}}$ *and* $\mathsf{CP_{link}}$ *are trapdoor-commitment zero-knowledge, then* UIA *is trapdoor-commitment honest-verifier zero-knowledge.*

LUNAR[CFQR21]

Theorem 15. *Let* $\mathsf{PHP} = (\mathsf{r}, \mathsf{n}, \mathsf{m}, \mathsf{d}, \mathsf{n_e}, \mathcal{RE}, \mathcal{P}, \mathcal{V})$ *be a non-adaptive public-coin PHP over* $\mathcal{F}$ *and* $\mathcal{R}$*, let* CS^* *be a compiling commitment scheme as in Definition 22 equipped with CP-SNARKs* $\mathsf{CP_{opn}}$ *for* $\mathcal{R}_{\mathsf{opn}}$*,* $\mathsf{CP_{php}}$ *for a relation* $\mathcal{R}_{\mathsf{php}}$*, and* $\mathsf{CP_{link}}$ *for* $\mathsf{R_{link}}$*. Then we have:*

– *If* PHP *has straight-line extractability, then the scheme* UIA *defined above is a universal commit and prove interactive argument in the SRS model for* $\mathcal{R}'$ *such that:*

$$(\mathsf{R}, \mathsf{x}, (\mathsf{u}_j)_{j\in[\ell]}, \omega) \in \mathcal{R}' \iff (\mathsf{R}, \mathsf{x}, \mathsf{Decode}((\mathsf{u}_j)_{j\in[\ell]}), \omega) \in \mathcal{R}.$$

– *If, for a checker* C*,* PHP *(resp.* $\mathsf{CP_{php}}$*) is* $(\mathsf{b}+\mathbf{1}, \mathsf{C})$*-bounded honest-verifier zero knowledge (resp. trapdoor-commit* (b, C)*-leaky zero-knowledge), and both* $\mathsf{CP_{opn}}$ *and* $\mathsf{CP_{link}}$ *are trapdoor-commitment zero-knowledge, then* UIA *is trapdoor-commitment honest-verifier zero-knowledge.*

LETS BUILD A SNARK

AN EXAMPLE - THE PROBLEM

AN EXAMPLE - THE PROBLEM

PROVER AND VERIFIER HAVE

$$v(X), w(X), \mathrm{deg}(v, w) < d$$

AN EXAMPLE - THE PROBLEM

PROVER AND VERIFIER HAVE

$$v(X), w(X), \deg(v, w) < d$$

PROVER WANTS TO CONVINCE THE VERIFIER THAT WHEN GIVEN

AN EXAMPLE - THE PROBLEM

PROVER AND VERIFIER HAVE

$$v(X), w(X), \deg(v, w) < d$$

PROVER WANTS TO CONVINCE THE VERIFIER THAT WHEN GIVEN

$$\{L_i(X)\}_{i=1}^{d}, H = \{h_i\}_{i=1}^{d}, z_H(X)$$

AN EXAMPLE - THE PROBLEM

PROVER AND VERIFIER HAVE

$$v(X), w(X), \text{deg}(v, w) < d$$

PROVER WANTS TO CONVINCE THE VERIFIER THAT WHEN GIVEN

$$\{L_i(X)\}_{i=1}^{d}, H = \{h_i\}_{i=1}^{d}, z_H(X)$$

$$v(X) = \sum_{i=1}^{d} v_i L_i(X) \qquad w(X) = \sum_{i=1}^{d} v_{\sigma(i)} L_i(X)$$

AN EXAMPLE - THE PROBLEM

PROVER AND VERIFIER HAVE

$$v(X), w(X), \deg(v, w) < d$$

PROVER WANTS TO CONVINCE THE VERIFIER THAT WHEN GIVEN

$$\{L_i(X)\}_{i=1}^{d}, H = \{h_i\}_{i=1}^{d}, z_H(X)$$

$$v(X) = \sum_{i=1}^{d} v_i L_i(X) \qquad w(X) = \sum_{i=1}^{d} v_{\sigma(i)} L_i(X)$$

IT IS THE CASE THAT

AN EXAMPLE - THE PROBLEM

PROVER AND VERIFIER HAVE

$$v(X), w(X), \text{deg}(v, w) < d$$

PROVER WANTS TO CONVINCE THE VERIFIER THAT WHEN GIVEN

$$\{L_i(X)\}_{i=1}^{d}, H = \{h_i\}_{i=1}^{d}, z_H(X)$$

$$v(X) = \sum_{i=1}^{d} v_i L_i(X) \qquad w(X) = \sum_{i=1}^{d} v_{\sigma(i)} L_i(X)$$

IT IS THE CASE THAT

$$(v_1, v_2, v_3, \ldots, v_d) \qquad (v_{\sigma(1)}, v_{\sigma(2)}, v_{\sigma(3)}, \ldots, v_{\sigma(d)})$$

AN EXAMPLE - THE PROBLEM

PROVER AND VERIFIER HAVE

$$v(X), w(X), \deg(v, w) < d$$

PROVER WANTS TO CONVINCE THE VERIFIER THAT WHEN GIVEN

$$\{L_i(X)\}_{i=1}^{d}, H = \{h_i\}_{i=1}^{d}, z_H(X)$$

$$v(X) = \sum_{i=1}^{d} v_i L_i(X) \qquad w(X) = \sum_{i=1}^{d} v_{\sigma(i)} L_i(X)$$

IT IS THE CASE THAT

$$(v_1, v_2, v_3, \ldots, v_d) \qquad (v_{\sigma(1)}, v_{\sigma(2)}, v_{\sigma(3)}, \ldots, v_{\sigma(d)})$$

FOR SOME FUNCTION $\sigma : [d] \to [d]$ (DISCLAIMER: σ is NOT a permutation)

AN EXAMPLE - INTUITION

$$v(X) = \sum_{i=1}^{d} v_i L_i(X) \qquad w(X) = \sum_{i=1}^{d} v_{\sigma(i)} L_i(X)$$

$$(v_1, v_2, v_3, \ldots, v_d) \qquad (v_{\sigma(1)}, v_{\sigma(2)}, v_{\sigma(3)}, \ldots, v_{\sigma(d)})$$

AN EXAMPLE - INTUITION

$$v(X) = \sum_{i=1}^{d} v_i L_i(X) \qquad w(X) = \sum_{i=1}^{d} v_{\sigma(i)} L_i(X)$$

$$(v_1, v_2, v_3, \ldots, v_d) \qquad (v_{\sigma(1)}, v_{\sigma(2)}, v_{\sigma(3)}, \ldots, v_{\sigma(d)})$$

BOTH VECTORS SHARE SOME ENCODING

AN EXAMPLE - INTUITION

$$v(X) = \sum_{i=1}^{d} v_i L_i(X)$$

$$w(X) = \sum_{i=1}^{d} v_{\sigma(i)} L_i(X)$$

$$(v_1, v_2, v_3, \ldots, v_d)$$

$$(v_{\sigma(1)}, v_{\sigma(2)}, v_{\sigma(3)}, \ldots, v_{\sigma(d)})$$

BOTH VECTORS SHARE SOME ENCODING

$$T(X) = \prod_{i=1}^{d} (X - v_i) = \prod_{i=1}^{d} (X - v_{\sigma(i)})$$

AN EXAMPLE

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

AN EXAMPLE

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

$$T(v(X)) = \prod_{i=1}^{d} (v(X) - v_i)$$

AN EXAMPLE

$$T(X) = \prod_{i=1}^{d} (X - v_i) \qquad T(v(X)) = \prod_{i=1}^{d} (v(X) - v_i)$$

IT IS TRUE THAT

AN EXAMPLE

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

$$T(v(X)) = \prod_{i=1}^{d} (v(X) - v_i)$$

IT IS TRUE THAT $$T(v(h_i)) = \prod_{i=1}^{d} (v(h_i) - v_i) = 0, \forall h_i \in H$$

AN EXAMPLE

$$T(X) = \prod_{i=1}^{d} (X - v_i) \qquad T(v(X)) = \prod_{i=1}^{d} (v(X) - v_i)$$

IT IS TRUE THAT $T(v(h_i)) = \prod_{i=1}^{d} (v(h_i) - v_i) = 0, \forall h_i \in H$

THEN, $(X - h_i) \mid T(v(X)), \forall h_i \in H$, i.e., $\exists Q(X)$ s.t.

AN EXAMPLE

$$T(X) = \prod_{i=1}^{d} (X - v_i) \qquad T(v(X)) = \prod_{i=1}^{d} (v(X) - v_i)$$

IT IS TRUE THAT $T(v(h_i)) = \prod_{i=1}^{d} (v(h_i) - v_i) = 0, \forall h_i \in H$

THEN, $(X - h_i) \mid T(v(X)), \forall h_i \in H$, i.e., $\exists Q(X)$ s.t. $T(v(X)) = z_H(X)Q(X)$

AN EXAMPLE

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

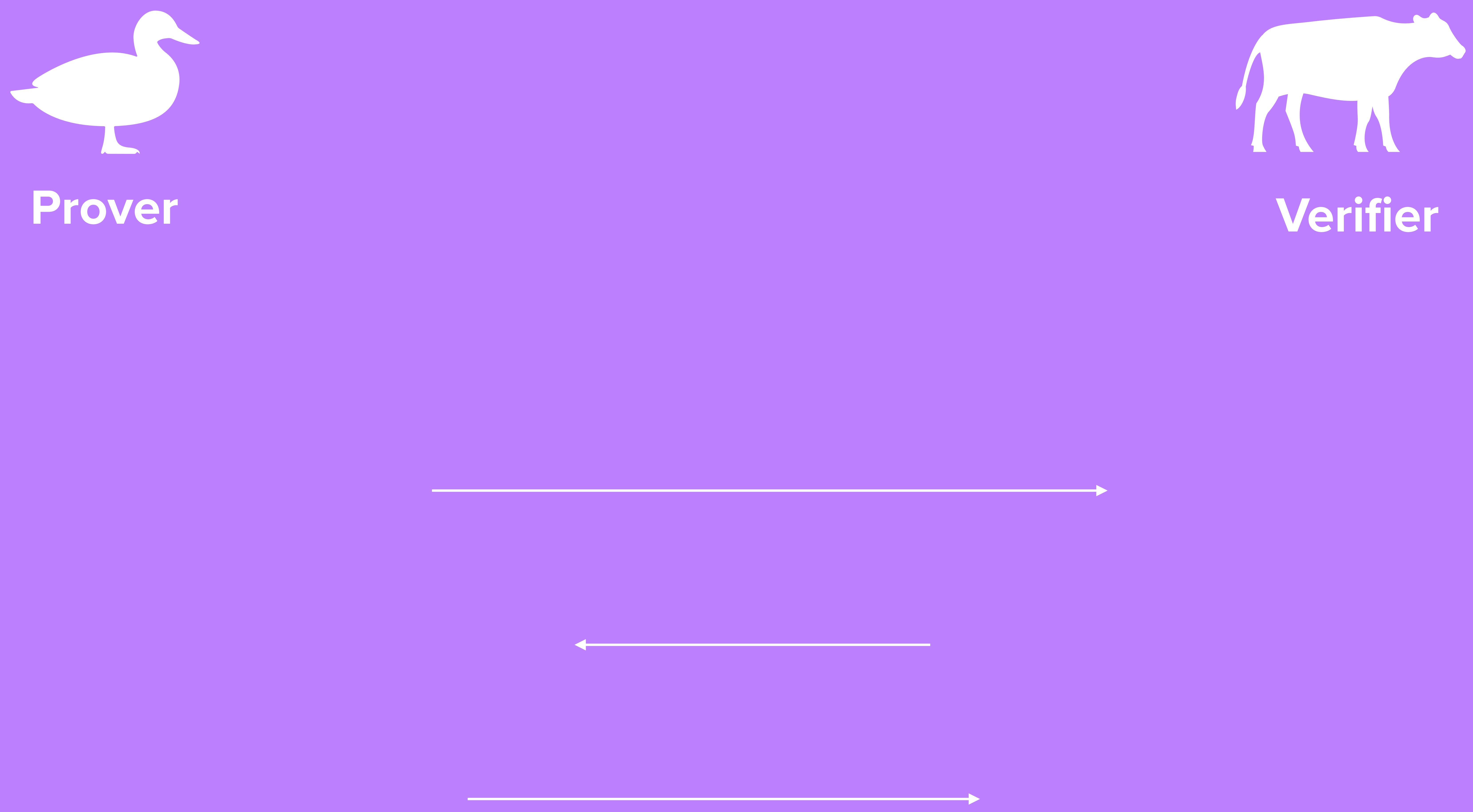
$$T(v(X)) = \prod_{i=1}^{d} (v(X) - v_i)$$

IT IS TRUE THAT $T(v(h_i)) = \prod_{i=1}^{d} (v(h_i) - v_i) = 0, \forall h_i \in H$

THEN, $(X - h_i) \mid T(v(X)), \forall h_i \in H$, i.e., $\exists Q(X)$ s.t. $T(v(X)) = z_H(X)Q(X)$

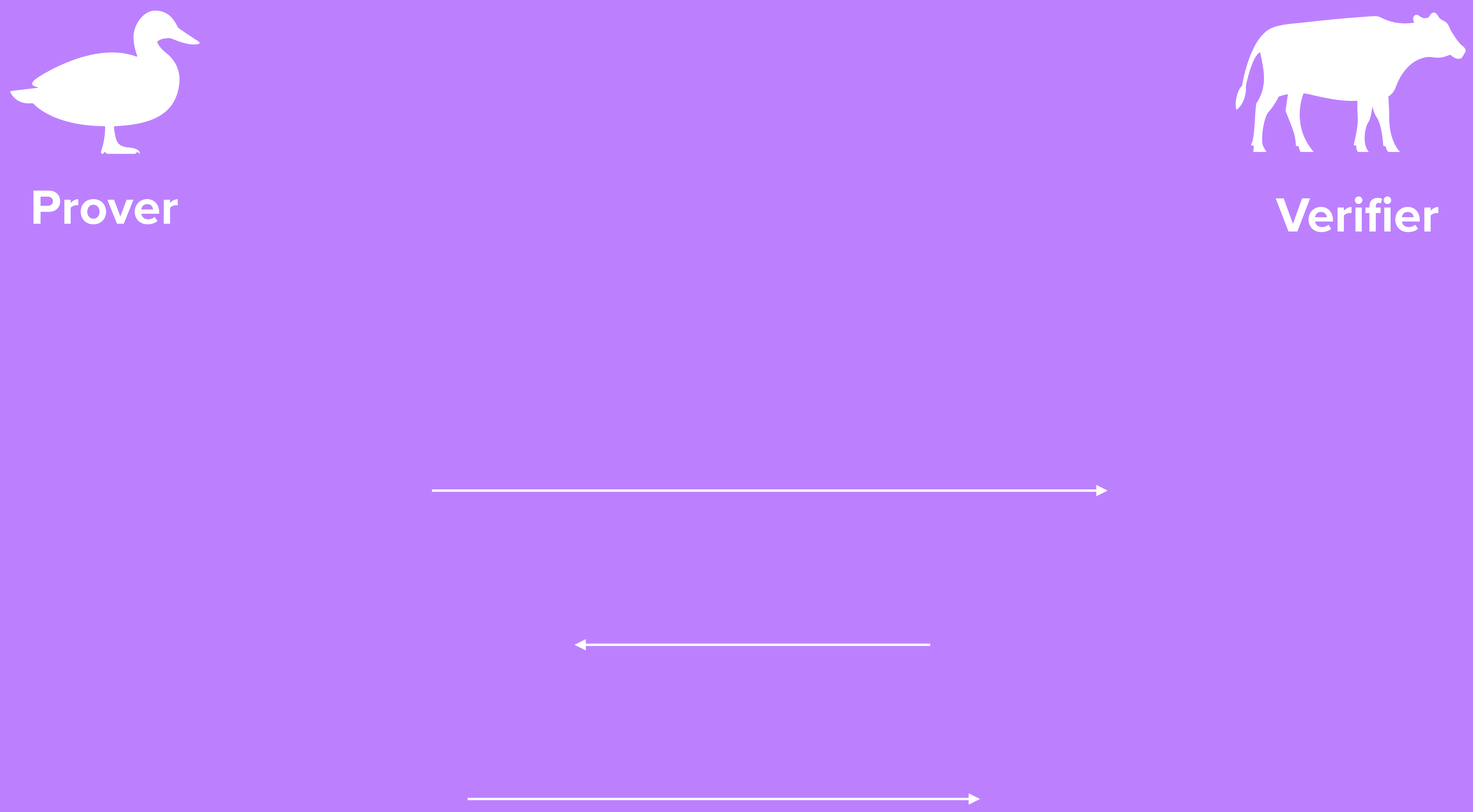
IT IS ALSO TRUE FOR $w(X)$!!

AN EXAMPLE



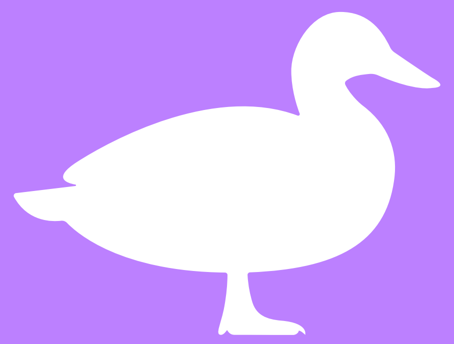
AN EXAMPLE

$$v(X), w(X), \deg(v, w) < d$$

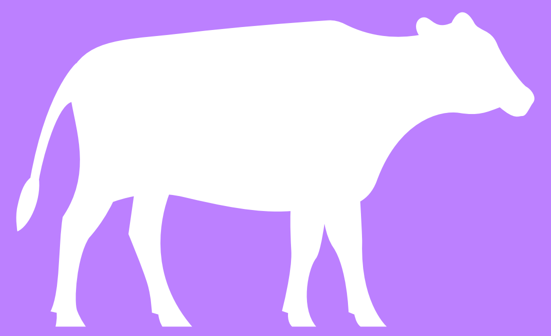


AN EXAMPLE

$$v(X), w(X), \deg(v, w) < d$$



Prover

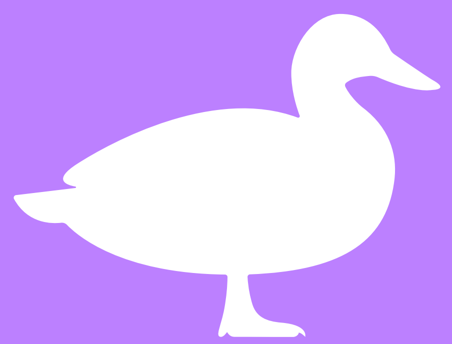


Verifier

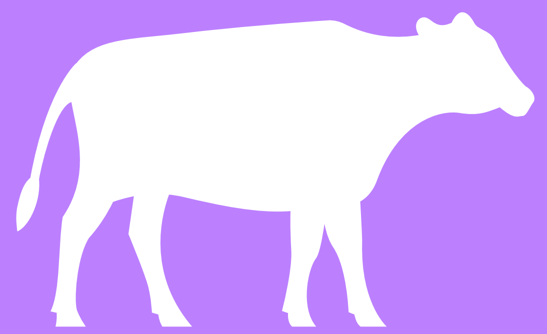
$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

AN EXAMPLE

$$v(X), w(X), \deg(v, w) < d$$



Prover



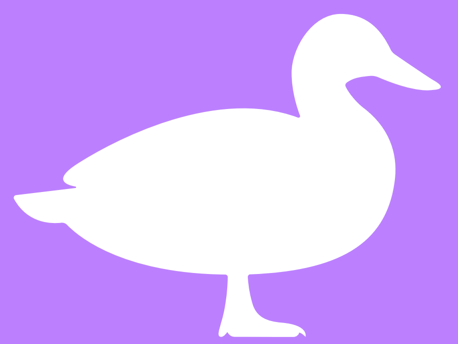
Verifier

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

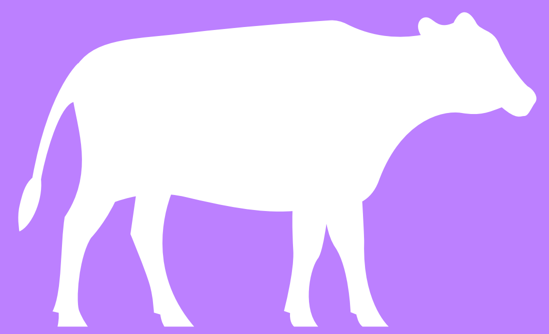
$$T(v(X)) = z_H(X) Q_1(X)$$

AN EXAMPLE

$$v(X), w(X), \deg(v, w) < d$$



Prover



Verifier

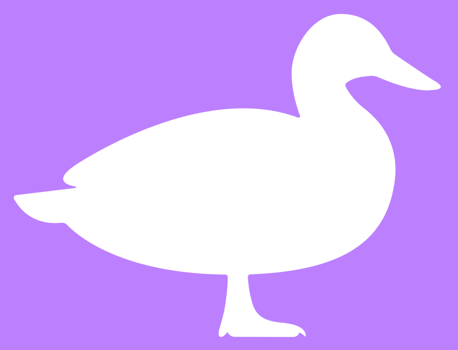
$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

$$T(v(X)) = z_H(X)Q_1(X)$$

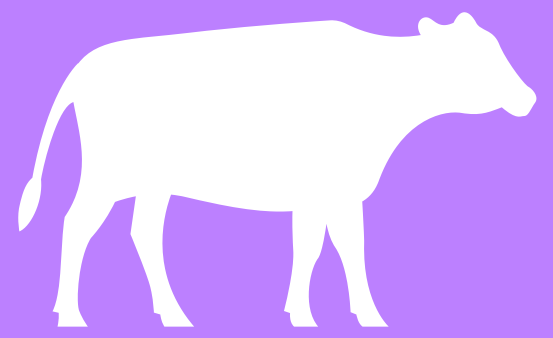
$$T(w(X)) = z_H(X)Q_2(X)$$

AN EXAMPLE

$$v(X), w(X), \mathrm{deg}(v, w) < d$$



Prover



Verifier

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

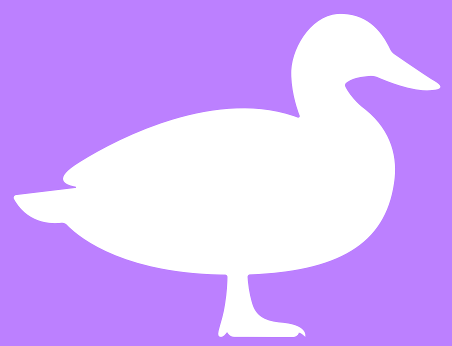
$$T(v(X)) = z_H(X)Q_1(X)$$

$$T(w(X)) = z_H(X)Q_2(X)$$

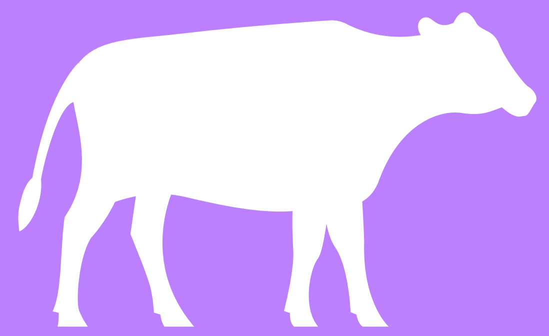
$$T(X), Q_1(X), Q_2(X)$$

AN EXAMPLE

$$v(X), w(X), \text{deg}(v, w) < d$$



Prover



Verifier

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

$$T(v(X)) = z_H(X)Q_1(X)$$

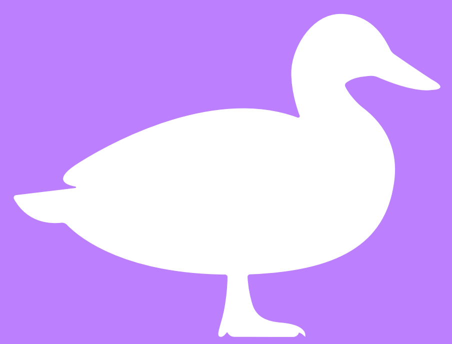
$$T(w(X)) = z_H(X)Q_2(X)$$

$$T(X), Q_1(X), Q_2(X)$$

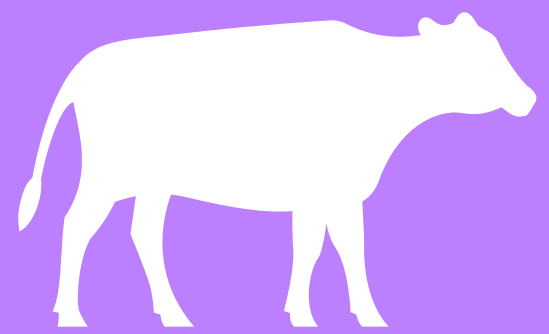
$$\alpha$$

AN EXAMPLE

$$v(X), w(X), \deg(v, w) < d$$



Prover



Verifier

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

$$T(v(X)) = z_H(X)Q_1(X)$$

$$T(w(X)) = z_H(X)Q_2(X)$$

$$T(X), Q_1(X), Q_2(X)$$

→

$$\alpha$$

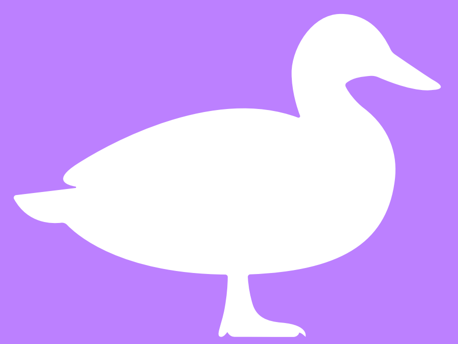
←

$$v(\alpha), w(\alpha), Q_1(\alpha), Q_2(\alpha)$$

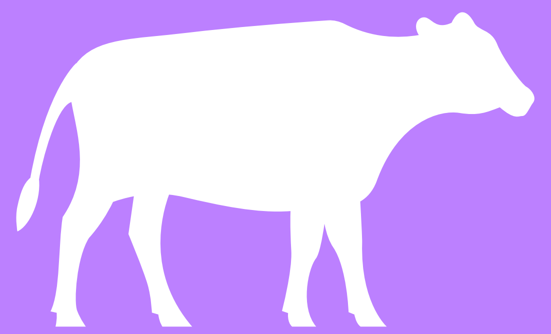
→

AN EXAMPLE

$$v(X), w(X), \text{deg}(v, w) < d$$



Prover



Verifier

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

$$T(v(X)) = z_H(X)Q_1(X)$$

$$T(w(X)) = z_H(X)Q_2(X)$$

$$T(X), Q_1(X), Q_2(X)$$

→

$$\alpha$$

←

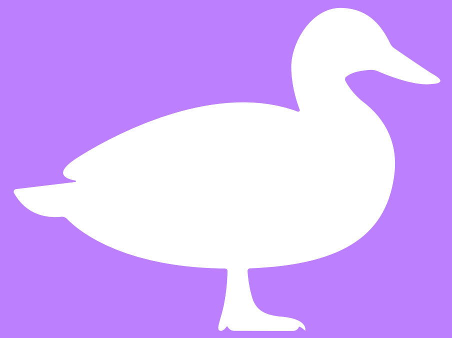
$$v(\alpha), w(\alpha), Q_1(\alpha), Q_2(\alpha)$$

→

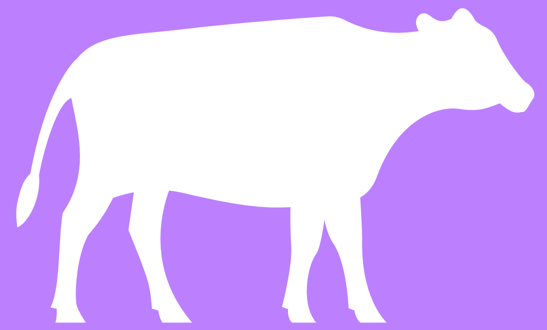
$$T(v(\alpha)) = z_H(\alpha)Q_1(\alpha) \ ???$$

AN EXAMPLE

$$v(X), w(X), \text{deg}(v, w) < d$$



Prover



Verifier

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

$$T(v(X)) = z_H(X)Q_1(X)$$

$$T(w(X)) = z_H(X)Q_2(X)$$

$$T(X), Q_1(X), Q_2(X)$$

→

$$\alpha$$

←

$$v(\alpha), w(\alpha), Q_1(\alpha), Q_2(\alpha)$$

→

$$T(v(\alpha)) = z_H(\alpha)Q_1(\alpha) \ ???$$

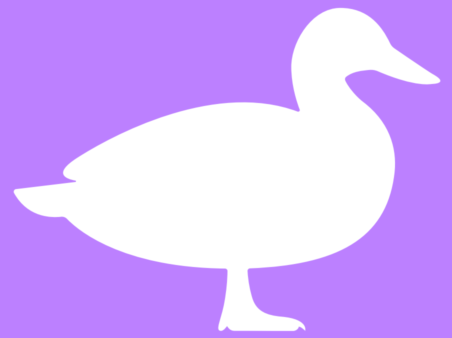
$$T(w(\alpha)) = z_H(\alpha)Q_2(\alpha) \ ???$$

TO DO: PROVE IT IS SOUND

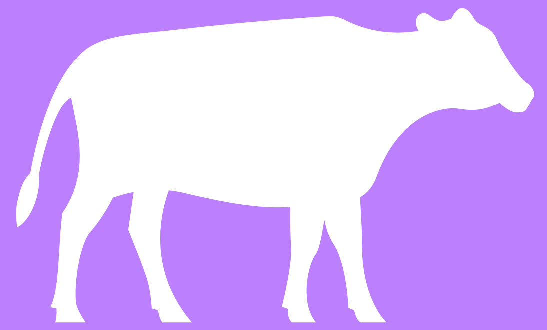
(SCHWARTZ-ZIPPEL)

AN EXAMPLE

$$v(X), w(X), \deg(v, w) < d$$



Prover



Verifier

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

$$T(v(X)) = z_H(X) Q_1(X)$$

$$T(w(X)) = z_H(X) Q_2(X)$$

$$T(X), Q_1(X), Q_2(X)$$

→

$$\alpha$$

←

$$v(\alpha), w(\alpha), Q_1(\alpha), Q_2(\alpha)$$

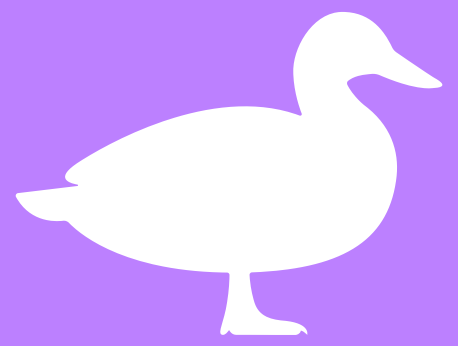
→

$$T(v(\alpha)) = z_H(\alpha) Q_1(\alpha) \text{ ???}$$

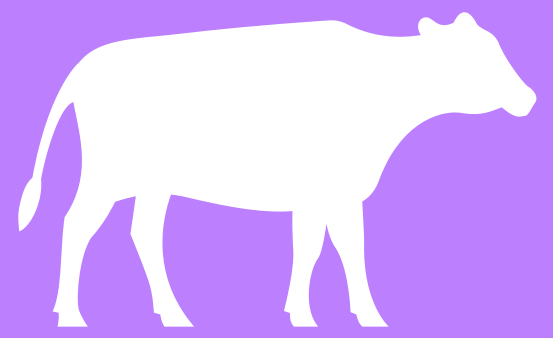
$$T(w(\alpha)) = z_H(\alpha) Q_2(\alpha) \text{ ???}$$

AN EXAMPLE

$$v \leftarrow \textbf{Commit}(v(X))$$
$$w \leftarrow \textbf{Commit}(w(X))$$



Prover



Verifier

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$
$$T(v(X)) = z_H(X) Q_1(X)$$
$$T(w(X)) = z_H(X) Q_2(X)$$

$T(X), Q_1(X), Q_2(X)$ →

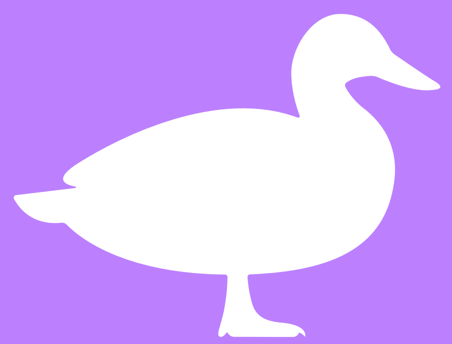
α ←

$v(\alpha), w(\alpha), Q_1(\alpha), Q_2(\alpha)$ →

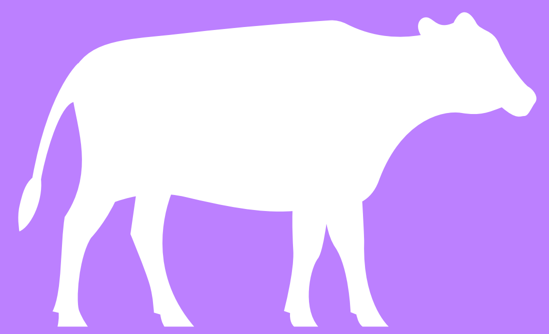
$$T(v(\alpha)) = z_H(\alpha) Q_1(\alpha) \ ???$$
$$T(w(\alpha)) = z_H(\alpha) Q_2(\alpha) \ ???$$

AN EXAMPLE

$$v \leftarrow \textbf{Commit}(v(X))$$
$$w \leftarrow \textbf{Commit}(w(X))$$



Prover



Verifier

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$

$$T(v(X)) = z_H(X) Q_1(X)$$

$$T(w(X)) = z_H(X) Q_2(X)$$

$$(T, Q_1, Q_2) \leftarrow \textbf{Commit}(T(X), Q_1(X), Q_2(X))$$

$\longrightarrow$

$$\alpha$$

$\longleftarrow$

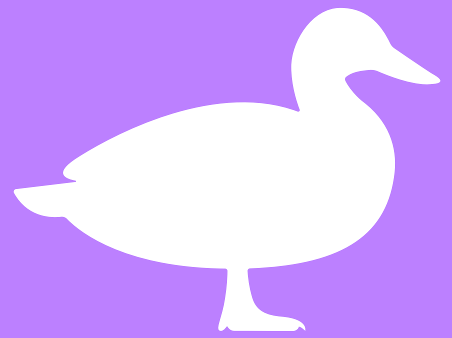
$$v(\alpha), w(\alpha), Q_1(\alpha), Q_2(\alpha)$$

$\longrightarrow$

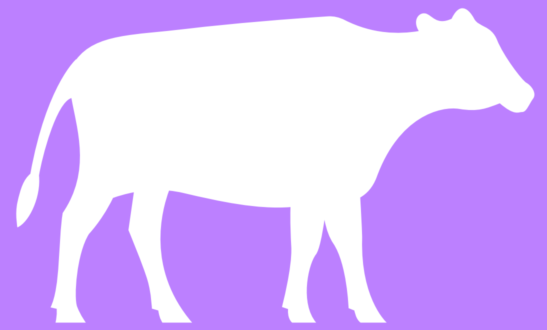
$$T(v(\alpha)) = z_H(\alpha) Q_1(\alpha) \text{ ???}$$
$$T(w(\alpha)) = z_H(\alpha) Q_2(\alpha) \text{ ???}$$

AN EXAMPLE

$$v \leftarrow \textbf{Commit}(v(X))$$
$$w \leftarrow \textbf{Commit}(w(X))$$



Prover



Verifier

$$T(X) = \prod_{i=1}^{d} (X - v_i)$$
$$T(v(X)) = z_H(X)Q_1(X)$$
$$T(w(X)) = z_H(X)Q_2(X)$$

$$(T, Q_1, Q_2) \leftarrow \textbf{Commit}(T(X), Q_1(X), Q_2(X))$$

$$\alpha$$

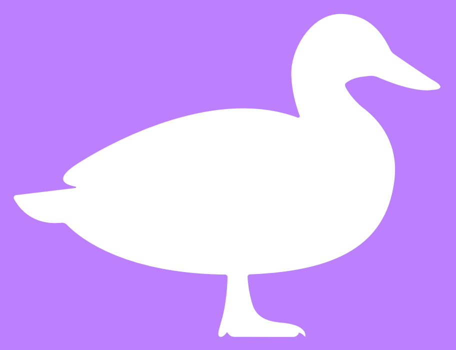
$$(\pi_1, v(\alpha), w(\alpha)) \leftarrow \textbf{Open}(v(X), w(X), \alpha)$$

$$(\pi_2, 0) \leftarrow \textbf{Open}(T(v(\alpha)) - z(X)Q_1(X), \alpha)$$
$$(\pi_3, 0) \leftarrow \textbf{Open}(T(w(\alpha)) - z(X)Q_2(X), \alpha)$$

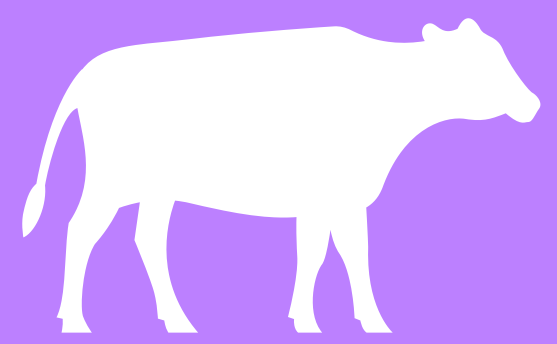
$$T(v(\alpha)) = z_H(\alpha)Q_1(\alpha) \ ???$$
$$T(w(\alpha)) = z_H(\alpha)Q_2(\alpha) \ ???$$

AN EXAMPLE

$$v \leftarrow \textbf{Commit}(v(X))$$
$$w \leftarrow \textbf{Commit}(w(X))$$



Prover



Verifier

$$T(X) = \prod_{i=1}^{d}(X - v_i)$$

$$T(v(X)) = z_H(X)Q_1(X)$$

$$T(w(X)) = z_H(X)Q_2(X)$$

$$(T, Q_1, Q_2) \leftarrow \textbf{Commit}(T(X), Q_1(X), Q_2(X))$$

$$\alpha$$

$$(\pi_1, v(\alpha), w(\alpha)) \leftarrow \textbf{Open}(v(X), w(X), \alpha)$$

$$(\pi_2, 0) \leftarrow \textbf{Open}(T(v(\alpha)) - z(X)Q_1(X), \alpha)$$

$$(\pi_3, 0) \leftarrow \textbf{Open}(T(w(\alpha)) - z(X)Q_2(X), \alpha)$$

$$\textbf{Verify}(v, w, \alpha, \pi_1, v(\alpha), w(\alpha))$$

$$\textbf{Verify}(T(v(\alpha) - z_H(\alpha)Q_1), \alpha, \pi_2, 0)$$

$$\textbf{Verify}(T(v(\alpha) - z_H(\alpha)Q_2), \alpha, \pi_3, 0)$$

KZG DOES THE REST

(WE CAN USE KZG AS A VECTOR COMMITMENT)

AN EXAMPLE - INTUITION

$$v(X) = \sum_{i=1}^{d} v_i L_i(X) \qquad w(X) = \sum_{i=1}^{d} v_{\sigma(i)} L_i(X)$$

$$(v_1, v_2, v_3, \ldots, v_d) \qquad (v_{\sigma(1)}, v_{\sigma(2)}, v_{\sigma(3)}, \ldots, v_{\sigma(d)})$$

RIGHT HERE

BOTH P IE ENCODING

$$T(X) = \prod_{i=1} (X - v_i) = \prod_{i=1} (X - v_{\sigma(i)})$$

PROS AND CONS

Constant size commitment

Constant time prover

Universal SRS

Homomorphic

Allows pre-computation

dlog(d) prover work in the field

Needs a setup

THANK YOU!