Lunar (and) Eclipse:

Commit-and-Prove SNARKs with Universal SRS

Matteo Campanelli @ AU Crypto Summer Day #1 2021

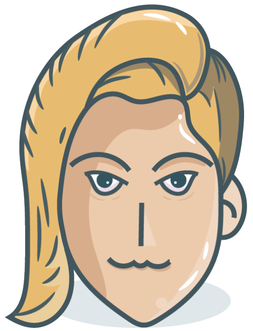
Lunar is a joint work with: Dario Fiore, Antonio Faonio, Anaïs Querol

ECLIPSE is a joint work with: Diego F. Aranha, Emil Madsen Bennedsen, Chaya Ganesh, Claudio Orlandi, Akira Takahashi

ZK

P

V

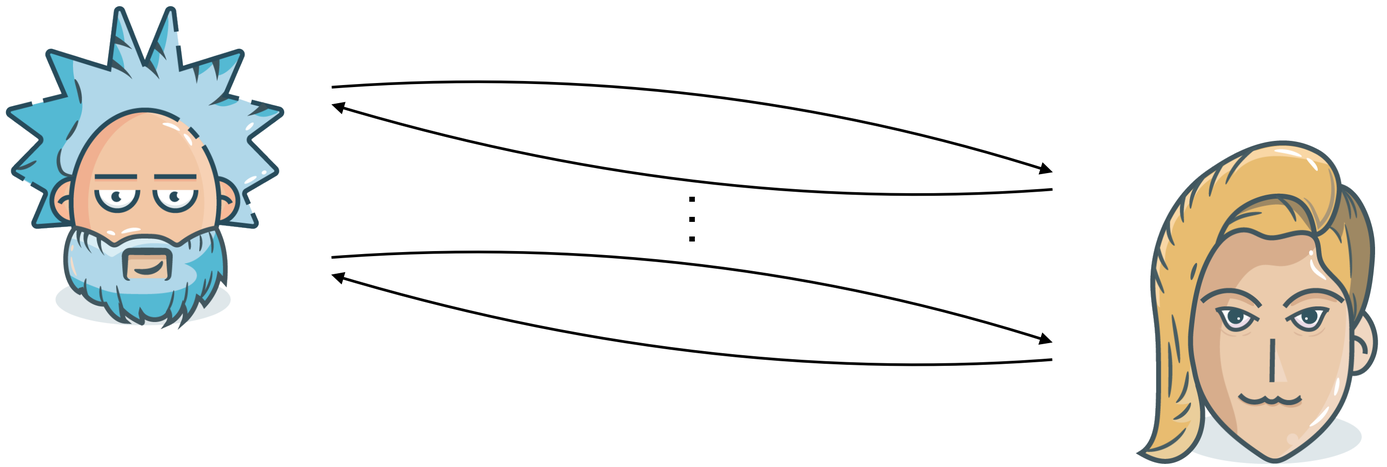


“Trusting properties claimed by someone else on data that you have not seen”

ZK

P

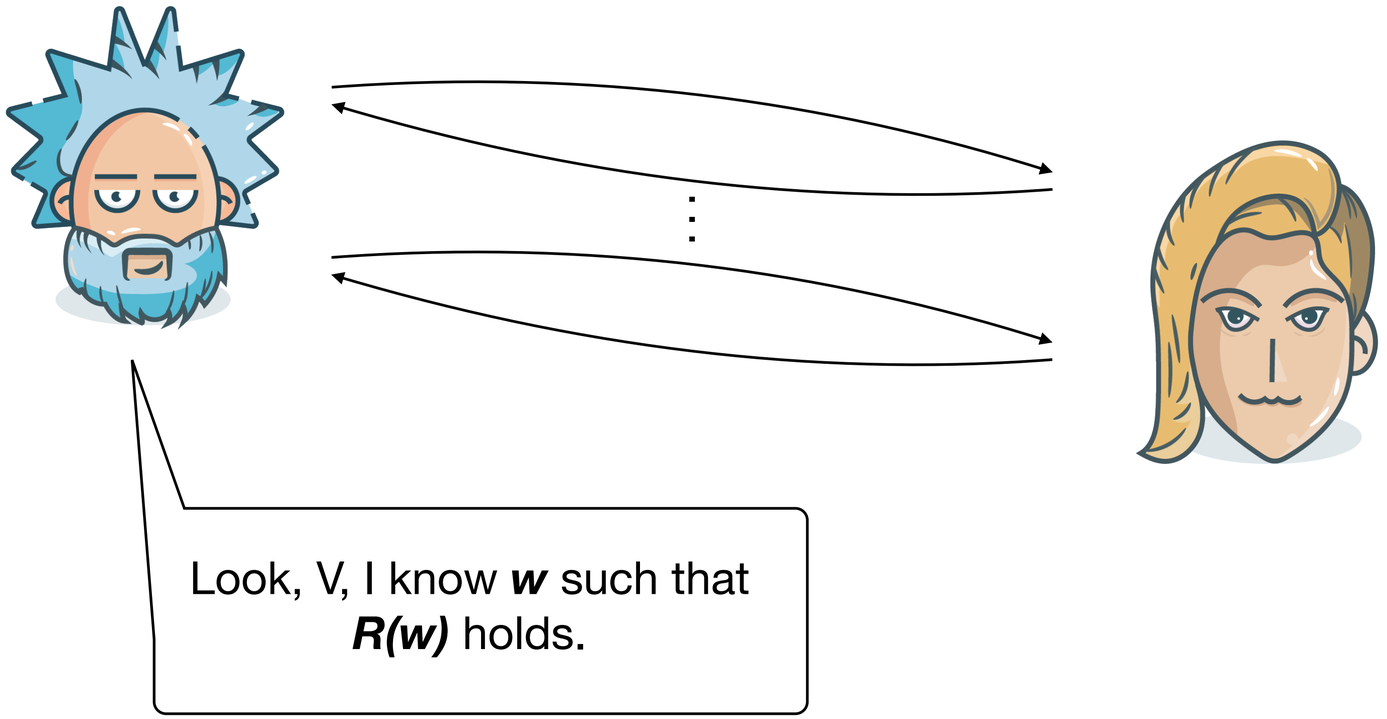
V



“Trusting properties claimed by someone else on data that you have not seen”

ZK

P **V**

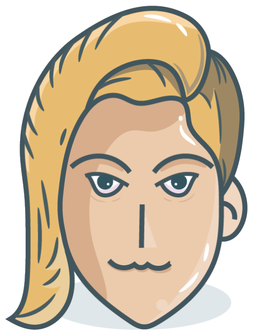


“Trusting properties claimed by someone else on data that you have not seen”

Commit-and-Prove (CP) ZK

P

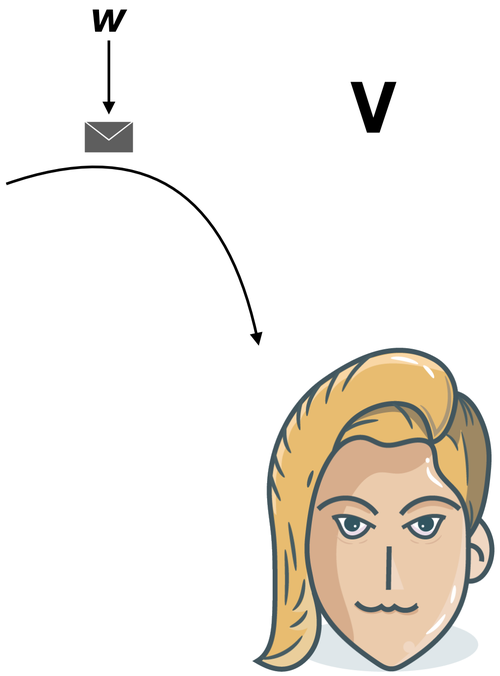
V



“Trusting properties claimed by someone else on data that you have not seen
but that can be pointed to”

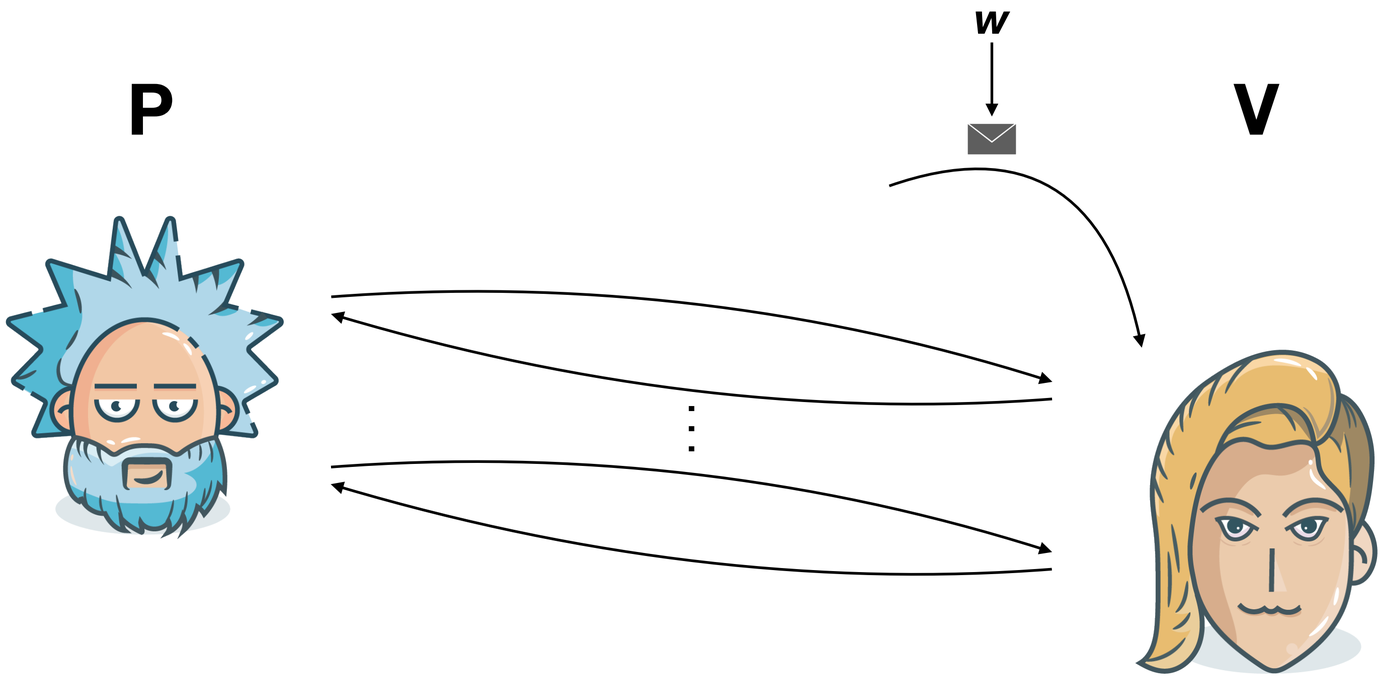
Commit-and-Prove (CP) ZK

P



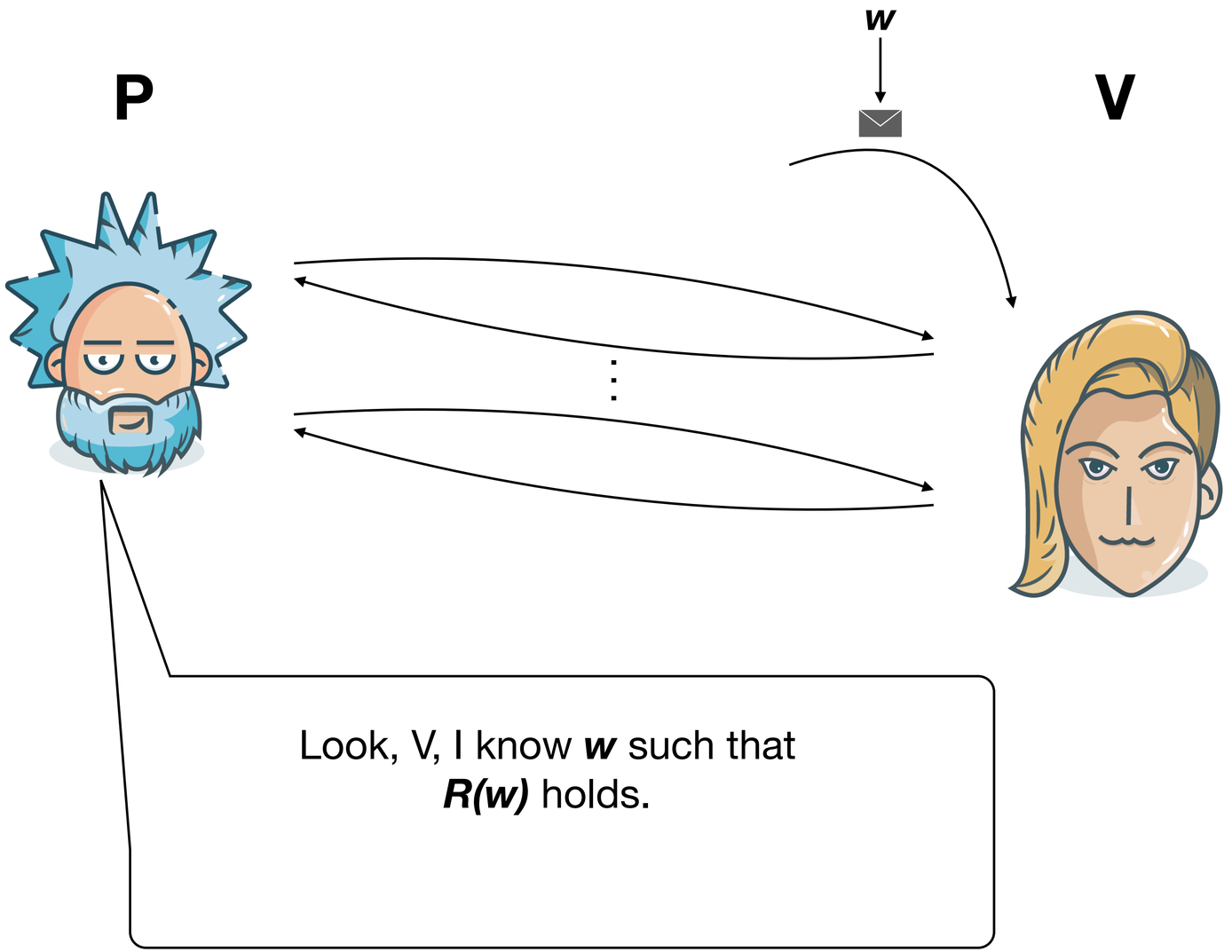
“Trusting properties claimed by someone else on data that you have not seen
but that can be pointed to”

Commit-and-Prove (CP) ZK



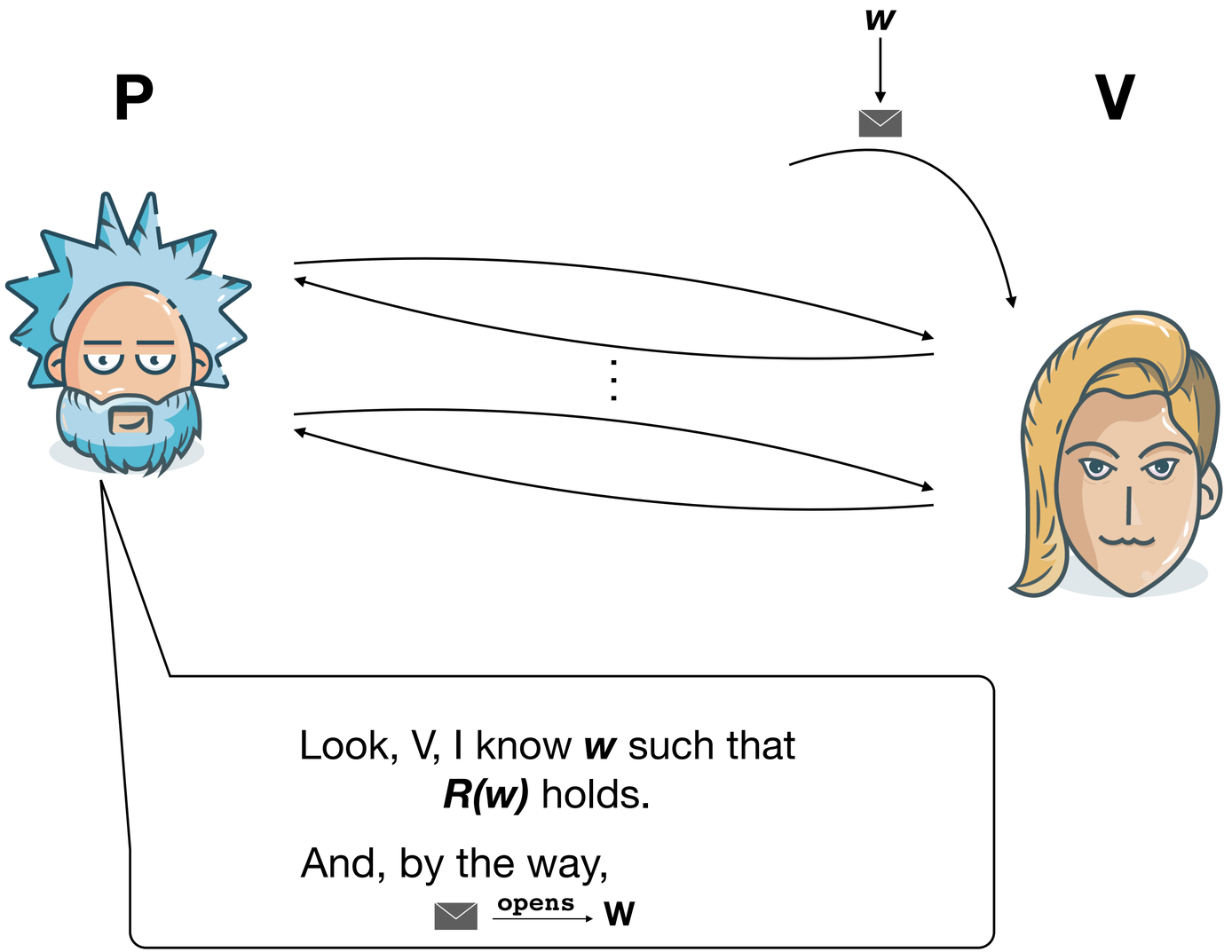
"Trusting properties claimed by someone else on data that you have not seen
but that can be pointed to"

Commit-and-Prove (CP) ZK



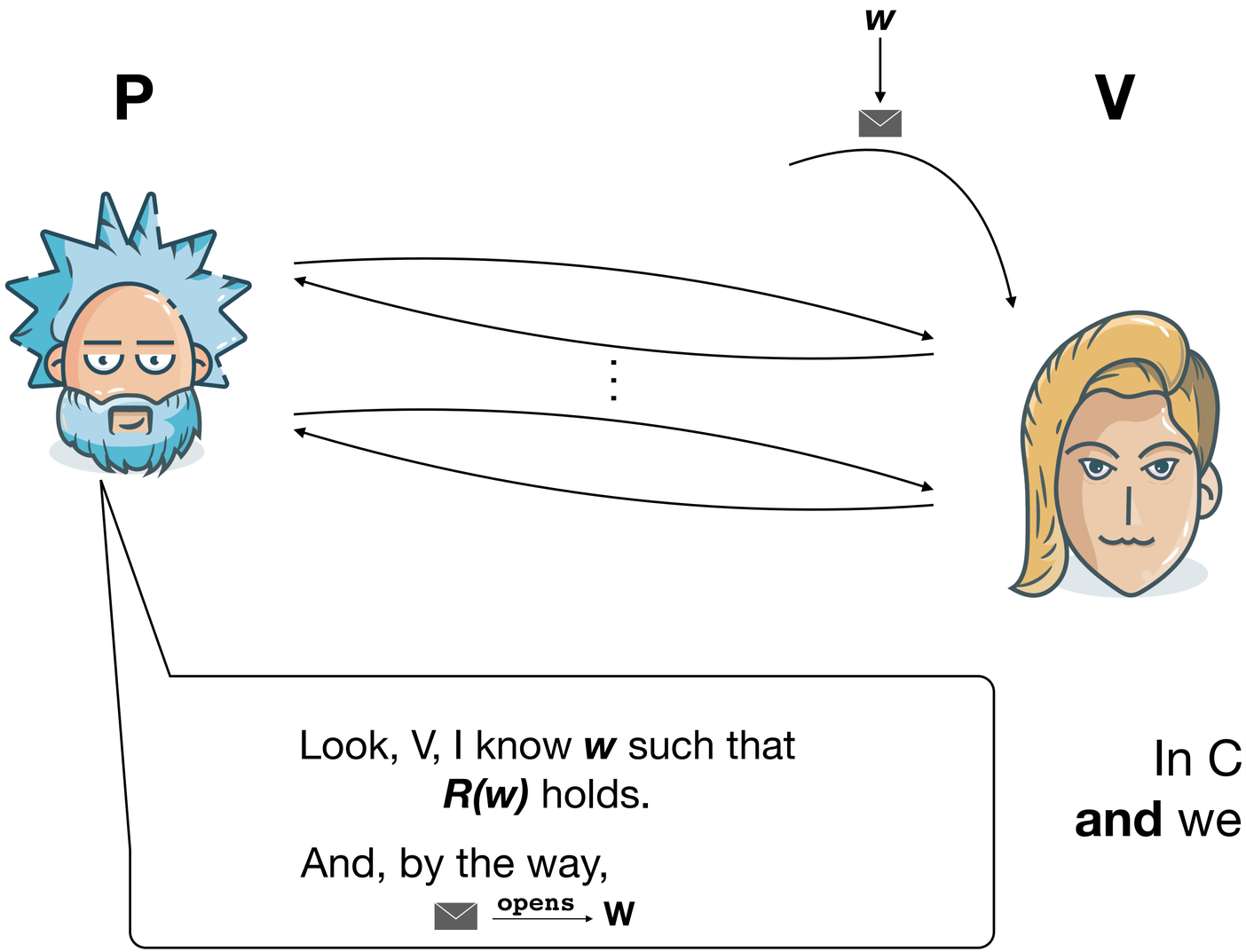
"Trusting properties claimed by someone else on data that you have not seen
but that can be pointed to"

Commit-and-Prove (CP) ZK



"Trusting properties claimed by someone else on data that you have not seen
but that can be pointed to"

Commit-and-Prove (CP) ZK



"Trusting properties claimed by someone else on data that you have not seen
but that can be pointed to"

In CP-ZK we prove R
and we open a commitment

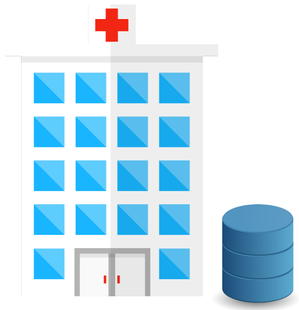
Motivation for CP

Motivation for CP

Compression/
Fingerprinting

Motivation for CP

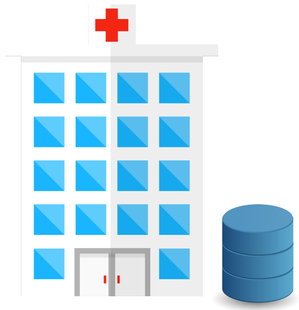
Compression/
Fingerprinting



Sensitive
DB

Motivation for CP

Compression/
Fingerprinting

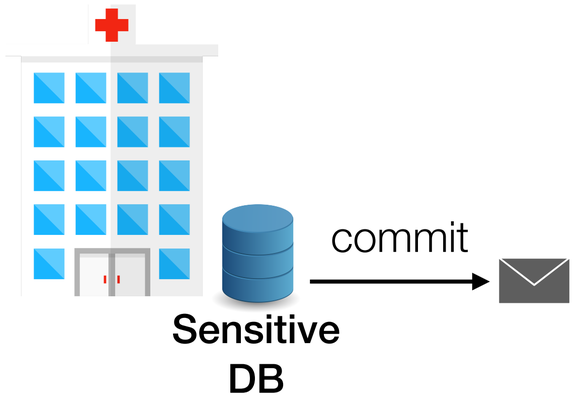


Sensitive
DB

Public ML models

Motivation for CP

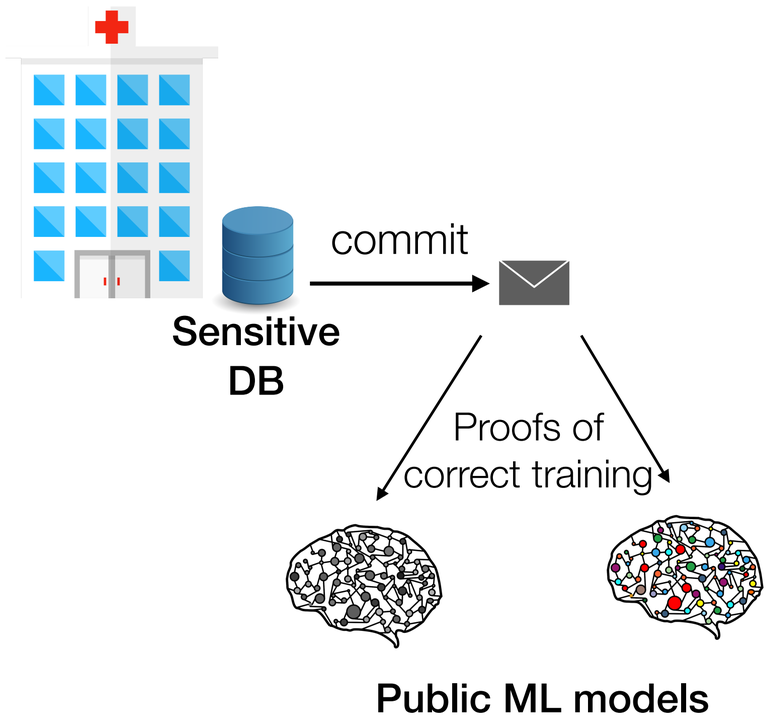
Compression/ Fingerprinting



Public ML models

Motivation for CP

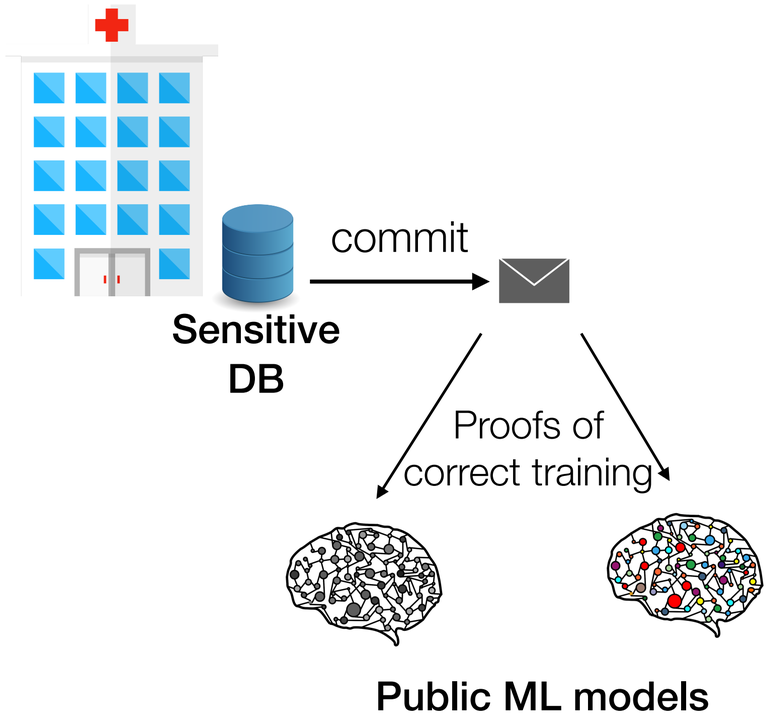
Compression/ Fingerprinting



Motivation for CP

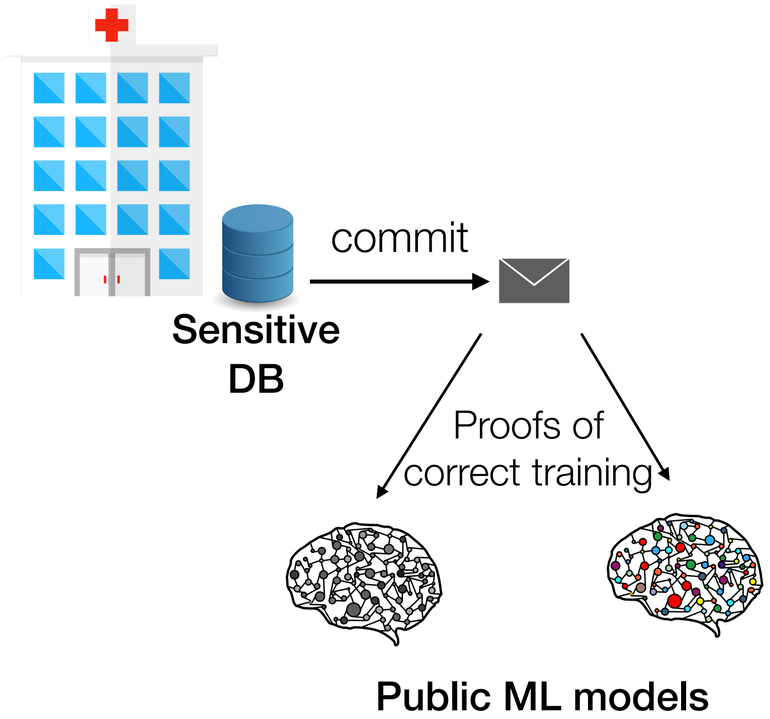
Compression/ Fingerprinting

Commit-ahead-of-time

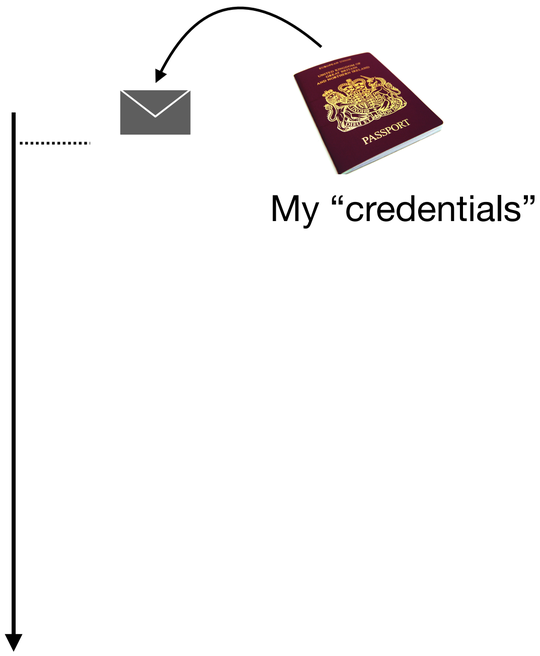


Motivation for CP

Compression/ Fingerprinting

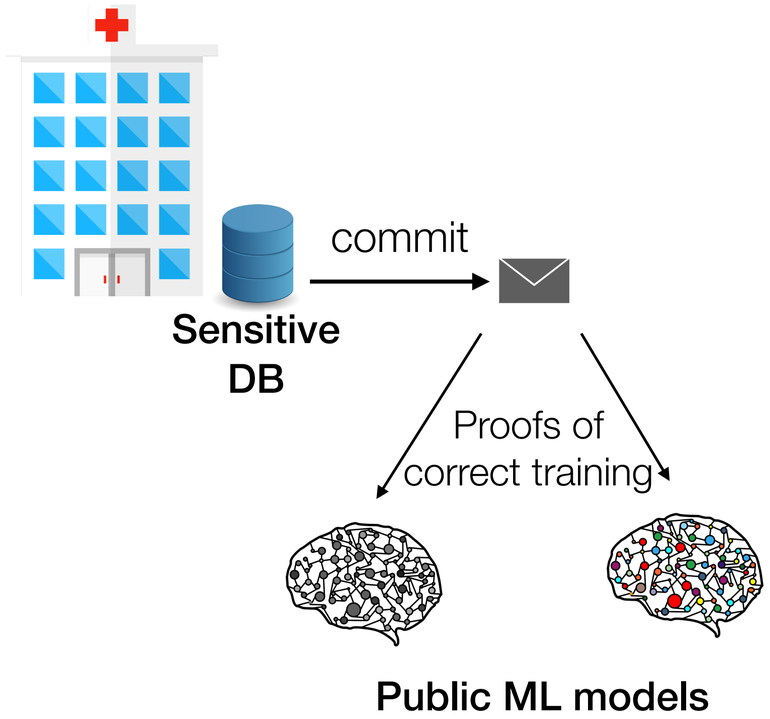


Commit-ahead-of-time

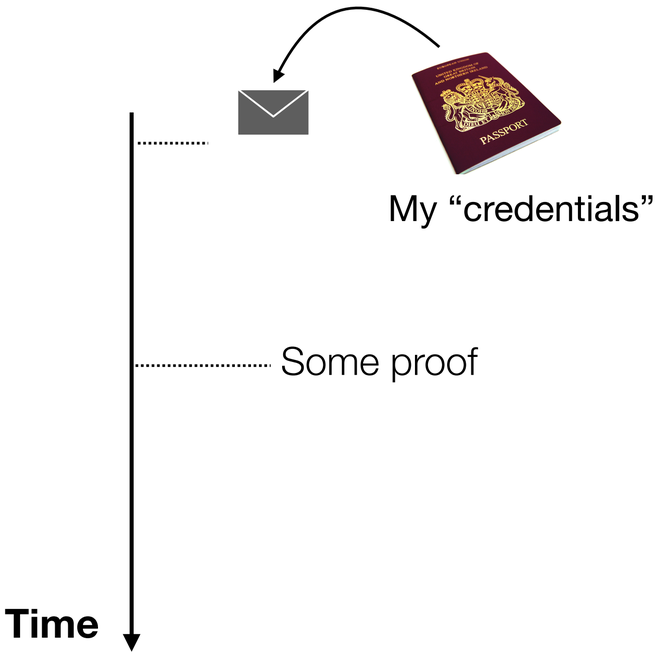


Motivation for CP

Compression/ Fingerprinting

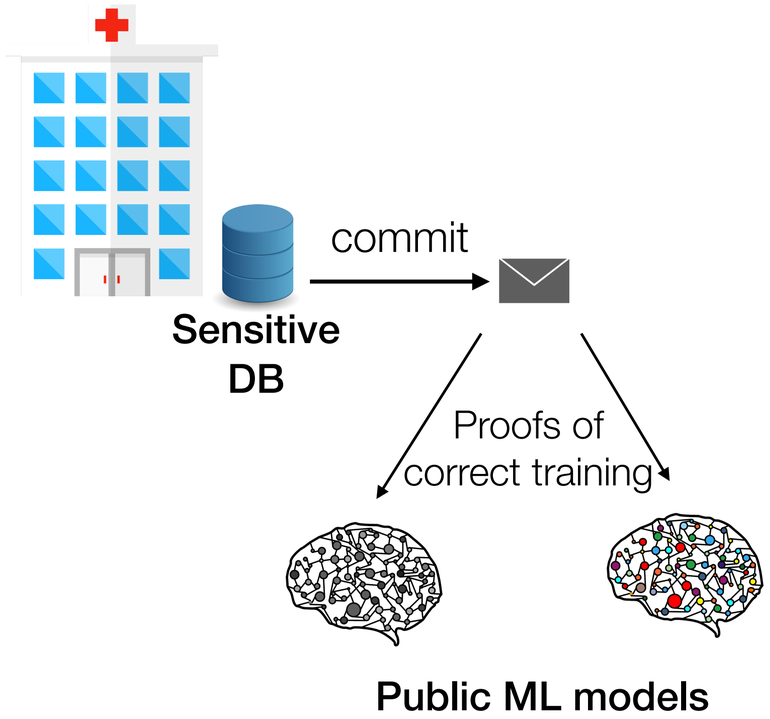


Commit-ahead-of-time

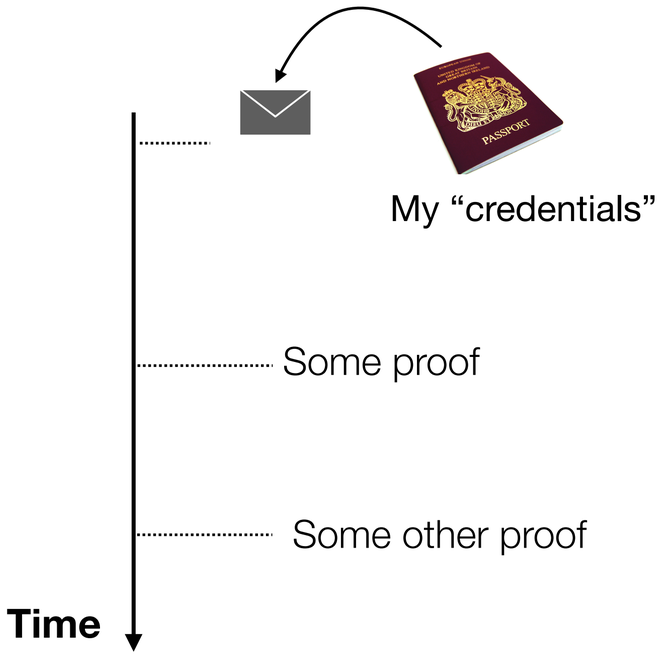


Motivation for CP

Compression/ Fingerprinting

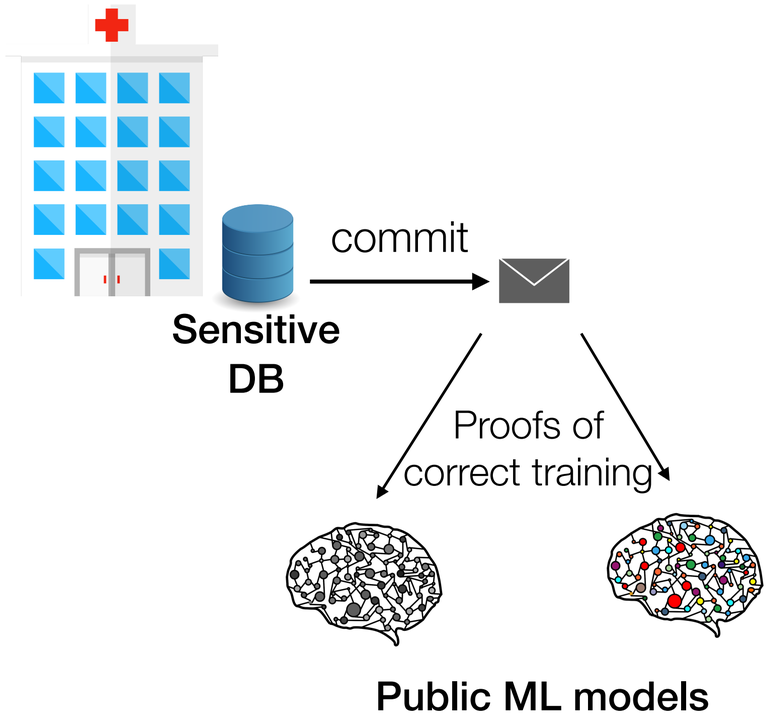


Commit-ahead-of-time

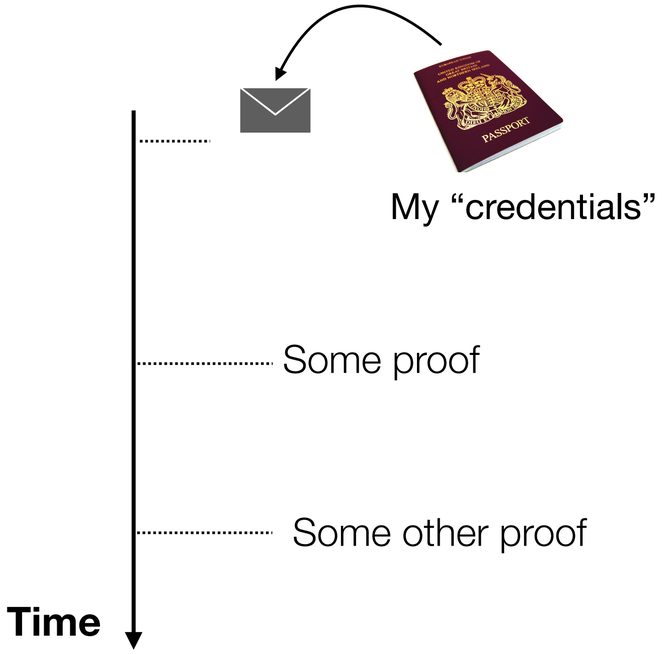


Motivation for CP

Compression/ Fingerprinting



Commit-ahead-of-time

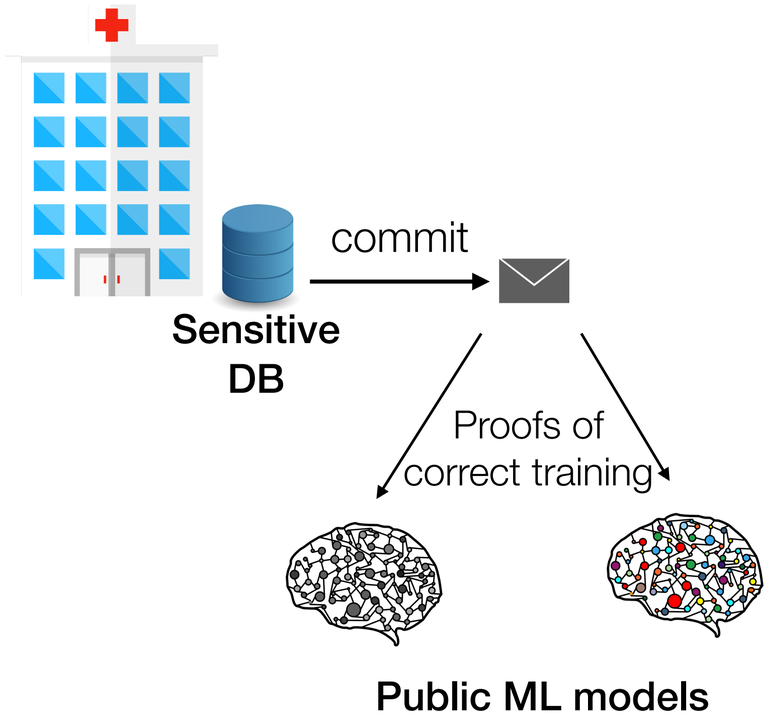


Modular/efficient composition of proofs

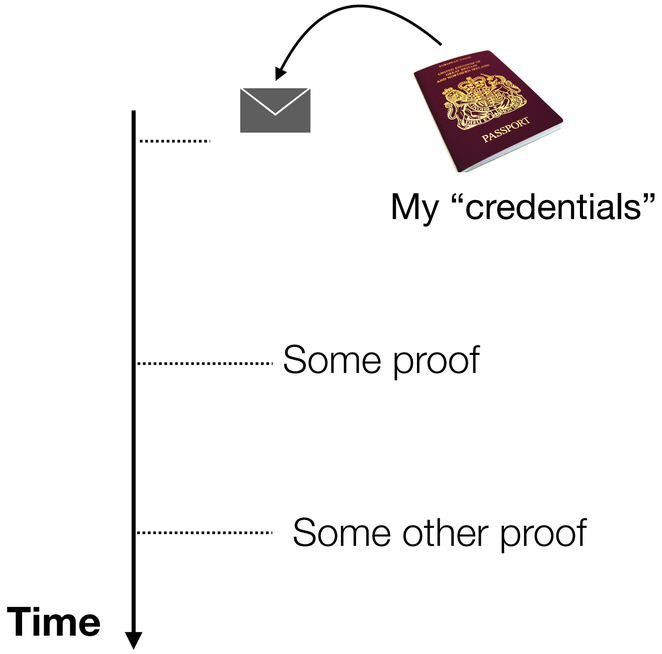
[AGM18,**C**FQ19]

Motivation for CP

Compression/ Fingerprinting

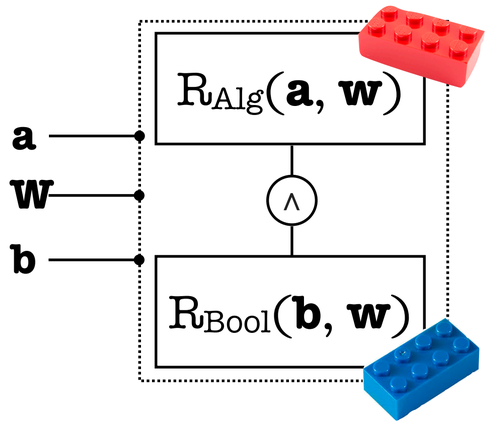


Commit-ahead-of-time



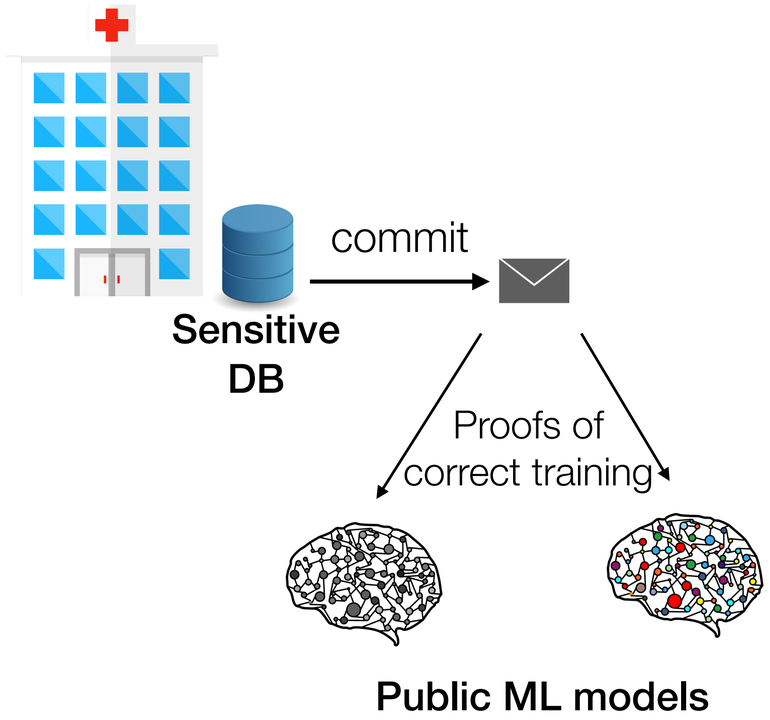
Modular/efficient composition of proofs

[AGM18,**C**FQ19]

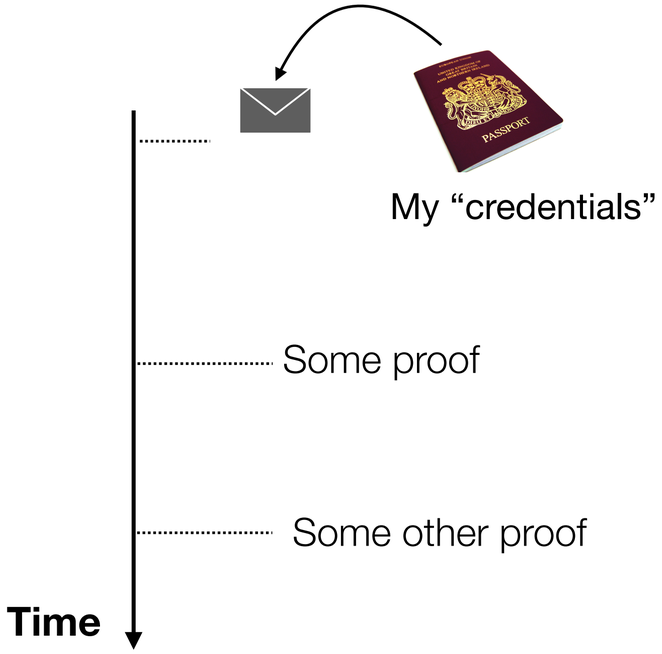


Motivation for CP

Compression/ Fingerprinting

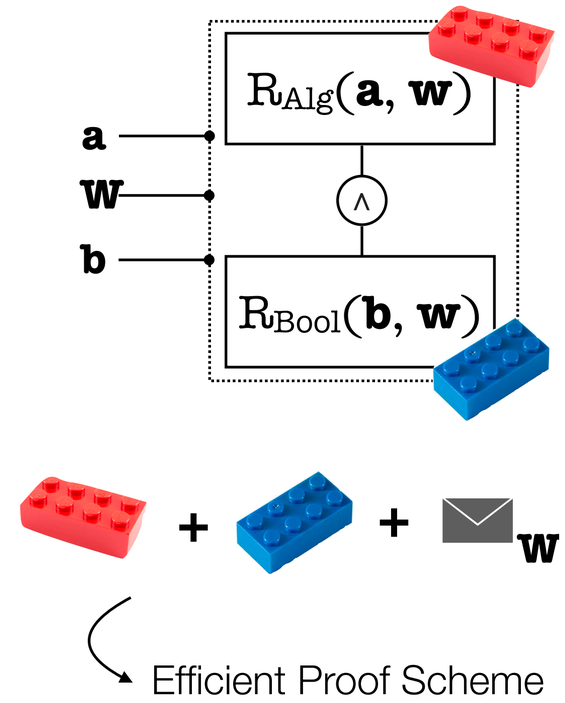


Commit-ahead-of-time



Modular/efficient composition of proofs

[AGM18,**C**FQ19]



Some Applications

Some Applications

- **Anonymous Credentials**

Some Applications

- **Anonymous Credentials**
- **Blockchains:**

Some Applications

- **Anonymous Credentials**
- **Blockchains:**
  - with privacy properties

Some Applications

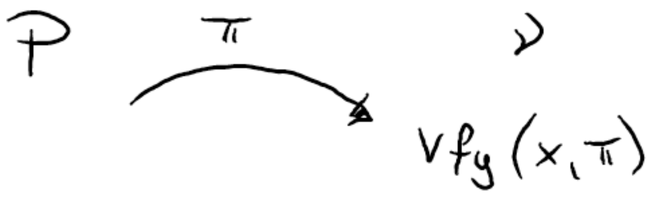
- **Anonymous Credentials**
- **Blockchains:**
  - with privacy properties
  - proofs on data posted on blockchains

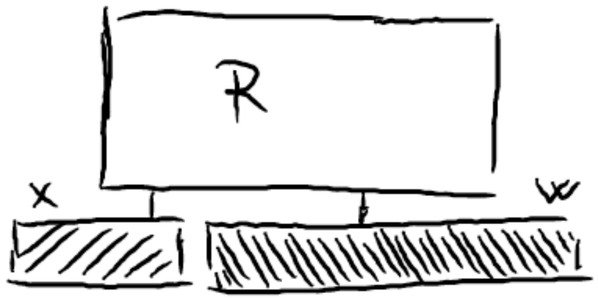
Some Applications

- **Anonymous Credentials**
- **Blockchains:**
  - with privacy properties
  - proofs on data posted on blockchains
- Anywhere data need to be referenced to (privately or succinctly)

Our Focus: (CP-)SNARKs

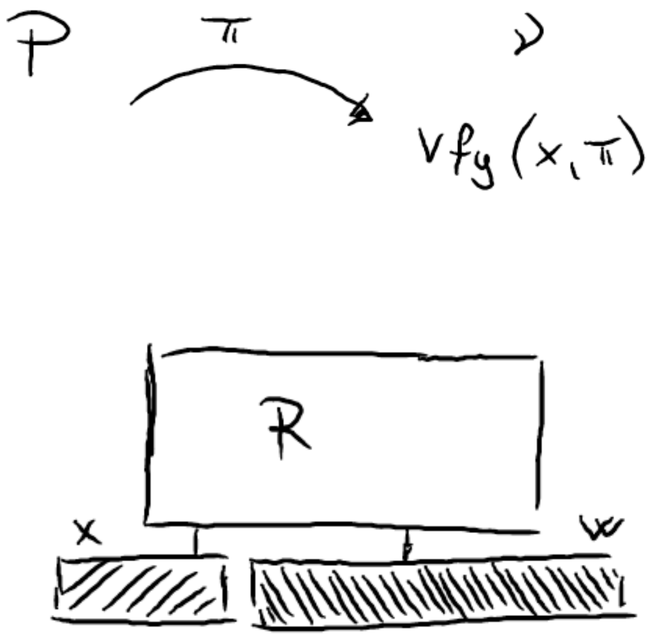
Succinct and Non-Interactive ZK

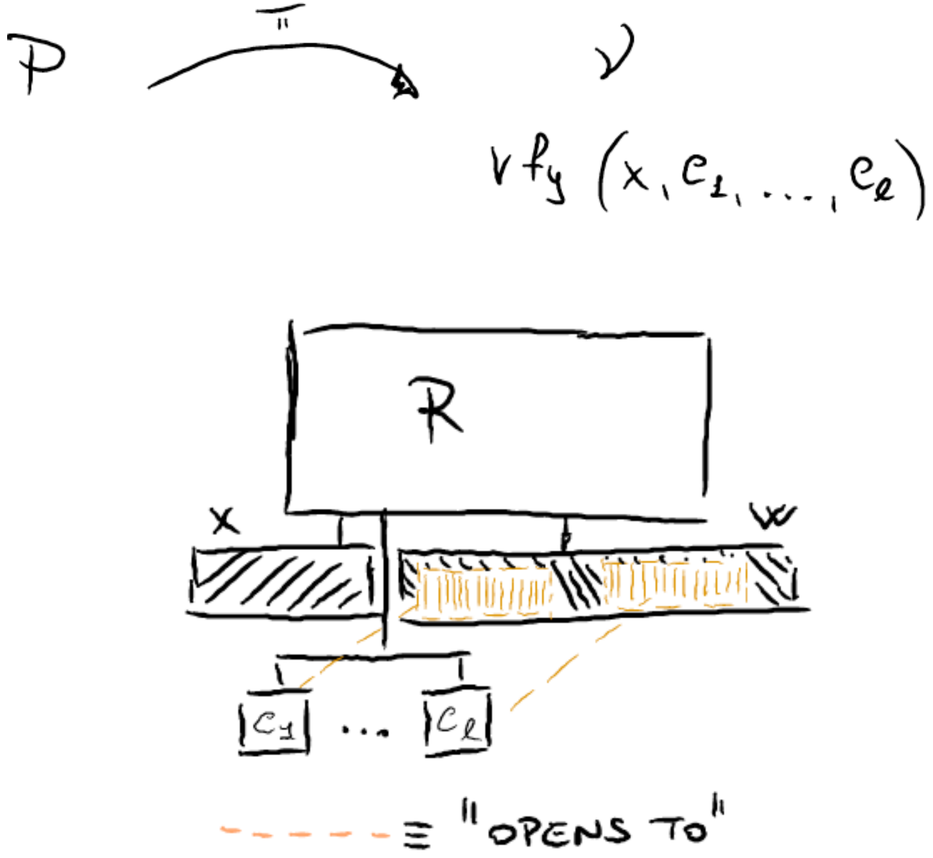




Our Focus: (CP-)SNARKs

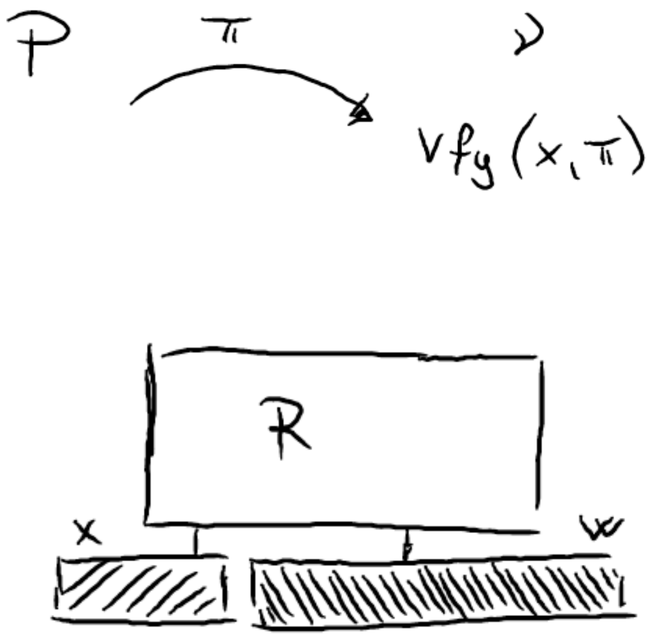
Succinct and Non-Interactive ZK

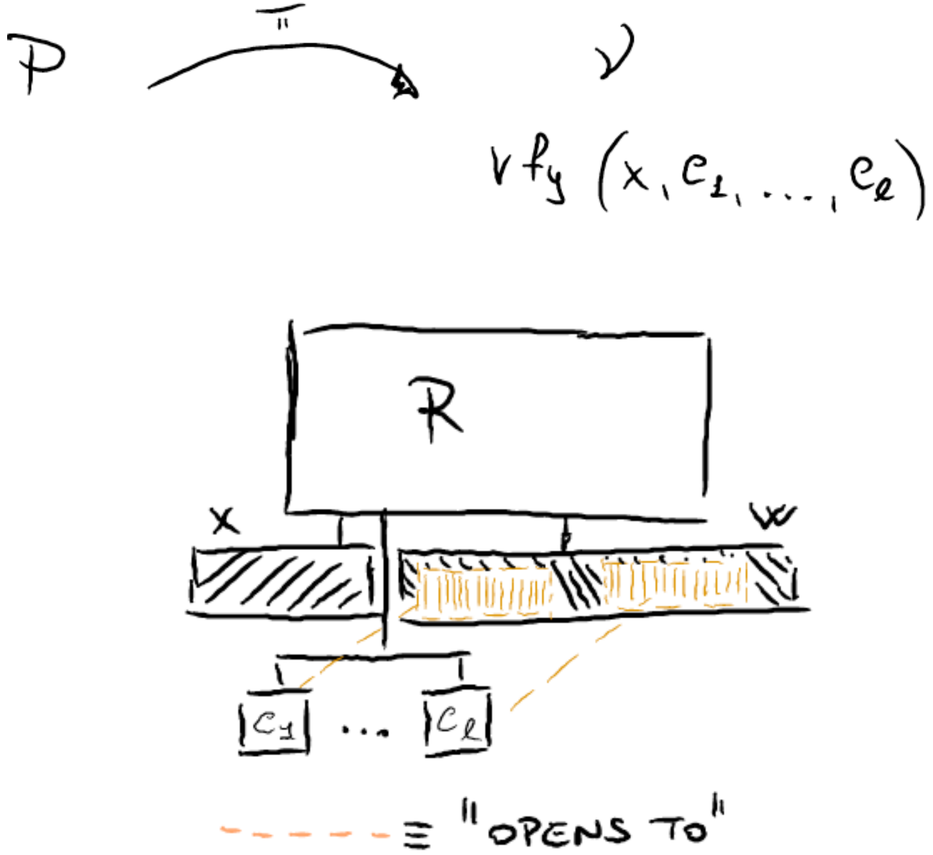




Our Focus: (CP-)SNARKs

Succinct and Non-Interactive ZK

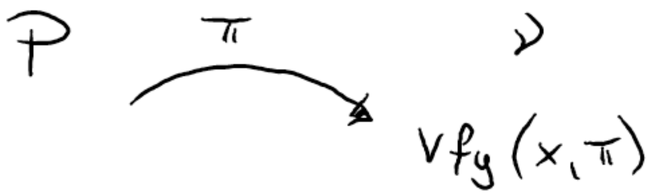


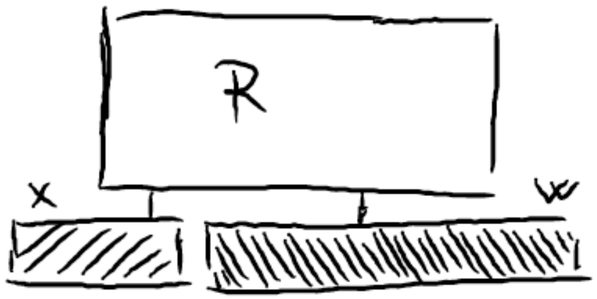


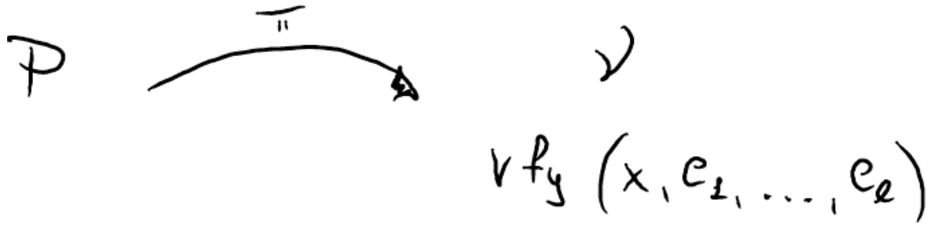
Why caring about the right setting if it's a special case of the left one?

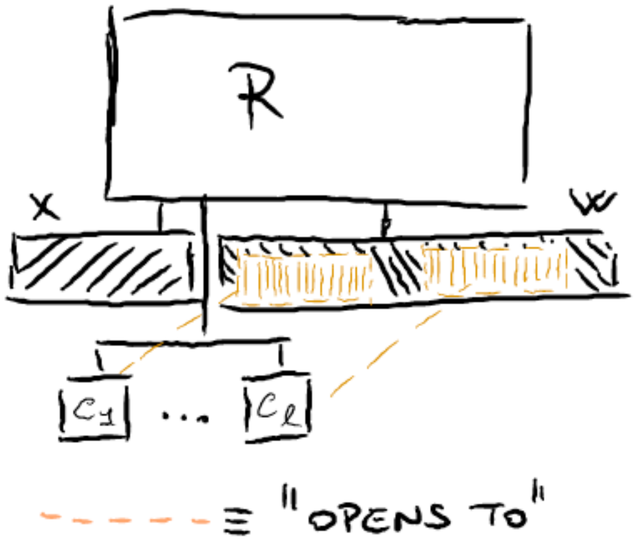
Our Focus: (CP-)SNARKs

Succinct and Non-Interactive ZK









Why caring about the right setting if it's a special case of the left one? **Efficiency**

Trust Models in (CP)-SNARKs

Trust Models in (CP)-SNARKs

- **Transparent :-)))** (Bulletproofs,Hyrax,DARK…)

Trust Models in (CP)-SNARKs

- **Transparent :-)))** (Bulletproofs,Hyrax,DARK…)
  - no trusted setup

Trust Models in (CP)-SNARKs

- **Transparent :-)))** (Bulletproofs,Hyrax,DARK…)
  - no trusted setup
- **SRS (Structured Reference String) :-|** (Pinocchio,Groth16…)

Trust Models in (CP)-SNARKs

- **Transparent :-)))** (Bulletproofs,Hyrax,DARK…)
  - no trusted setup
- **SRS (Structured Reference String) :-|** (Pinocchio,Groth16…)
  - Keygen(R) -> srs_R

Trust Models in (CP)-SNARKs

- **Transparent :-)))** (Bulletproofs,Hyrax,DARK…)
  - no trusted setup
- **SRS (Structured Reference String) :-|** (Pinocchio,Groth16…)
  - Keygen(R) -> srs_R
- **Universal SRS (USRS) :-)** (GKMM18,LegoSNARK,Sonic,Marlin,PLONK,…)

Trust Models in (CP)-SNARKs

- **Transparent :-)))** (Bulletproofs,Hyrax,DARK…)
  - no trusted setup
- **SRS (Structured Reference String) :-|** (Pinocchio,Groth16…)
  - Keygen(R) -> srs_R
- **Universal SRS (USRS) :-)** (GKMM18,LegoSNARK,Sonic,Marlin,PLONK,…)
  - Keygen(maxSize) -> srs_gen

Trust Models in (CP)-SNARKs

- **Transparent :-)))** (Bulletproofs,Hyrax,DARK…)
  - no trusted setup
- **SRS (Structured Reference String) :-|** (Pinocchio,Groth16…)
  - Keygen(R) -> srs_R
- **Universal SRS (USRS) :-)** (GKMM18,LegoSNARK,Sonic,Marlin,PLONK,…)
  - Keygen(maxSize) -> srs_gen
  - Specialize(srs_gen, R) -> srs_R

Trust Models in (CP)-SNARKs

- **Transparent :-)))** (Bulletproofs,Hyrax,DARK…)
  - no trusted setup
- **SRS (Structured Reference String) :-|** (Pinocchio,Groth16…)
  - Keygen(R) -> srs_R
- **Universal SRS (USRS) :-)** (GKMM18,LegoSNARK,Sonic,Marlin,PLONK,…)
  - Keygen(maxSize) -> srs_gen
  - Specialize(srs_gen, R) -> srs_R
  - Often also **updatable** (anyone can rerandomize srs_gen)

Lunar&Eclipse results from 10^9 feet:

new ways to construct CP-SNARKs with a Universal SRS generically

Lunar&Eclipse results from 10^9 feet:

new ways to construct CP-SNARKs with a Universal SRS generically

But before giving more details, we will need more background…

Warm up—what parents never say:

Warm up—what parents never say:

Where do (CP-)SNARKs come from?

Warm up—what parents never say:

Where do (CP-)SNARKs come from?

?

?

Warm up—what parents never say:

Where do (CP-)SNARKs come from?

?

?

Warning!

Ensure kids under 12 are under adult supervision before showing next slide.

The truth about where SNARKs come from

Compilers from information-theoretic objects!

The truth about where SNARKs come from

Compilers from information-theoretic objects!

The truth about where SNARKs come from

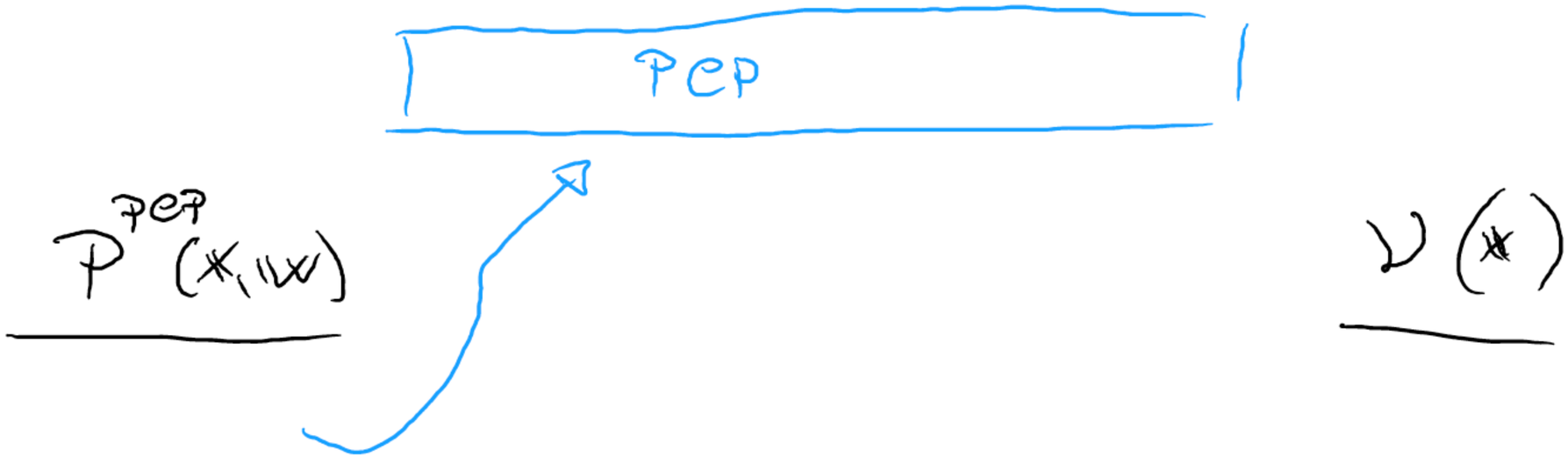
Compilers from information-theoretic objects!

Example: PCPs

Probabilistically Checkable Proofs

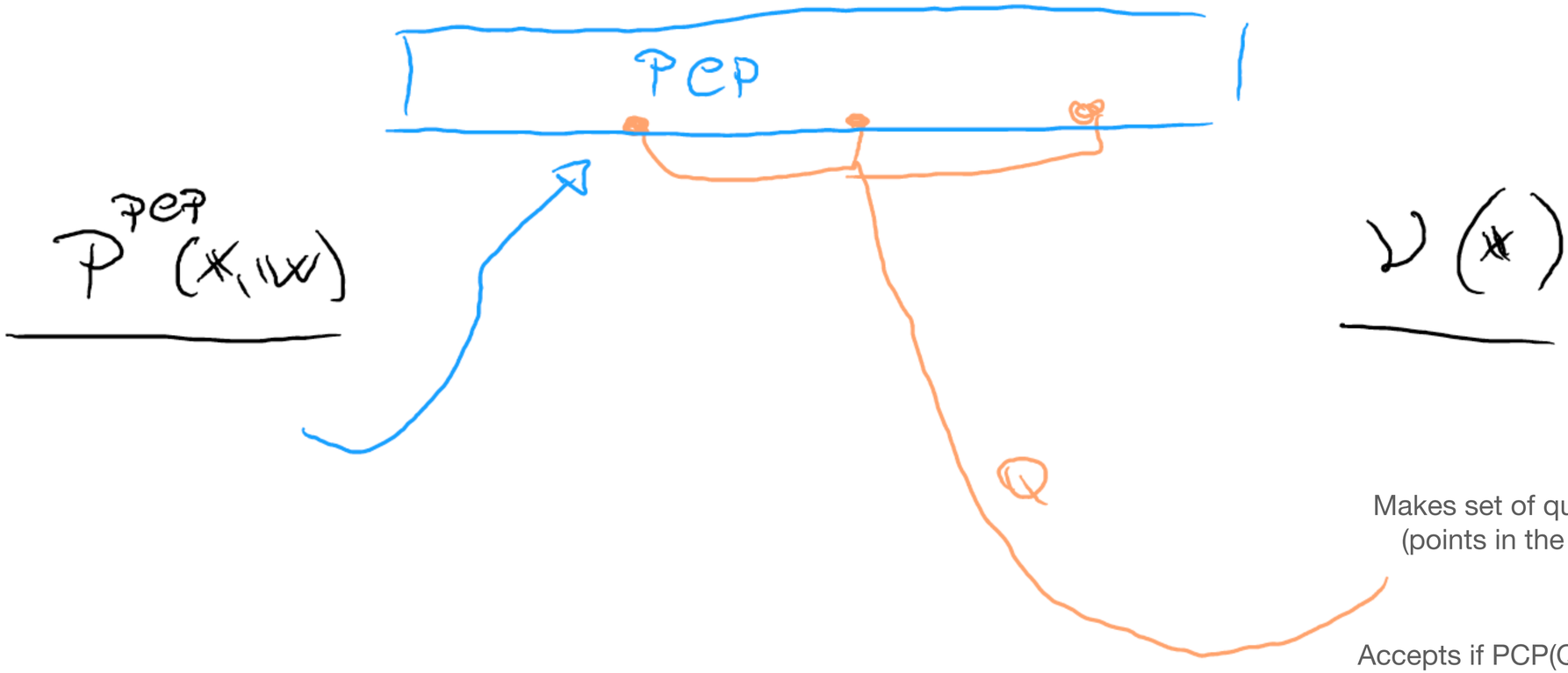
Example: PCPs

Probabilistically Checkable Proofs



Example: PCPs

Probabilistically Checkable Proofs

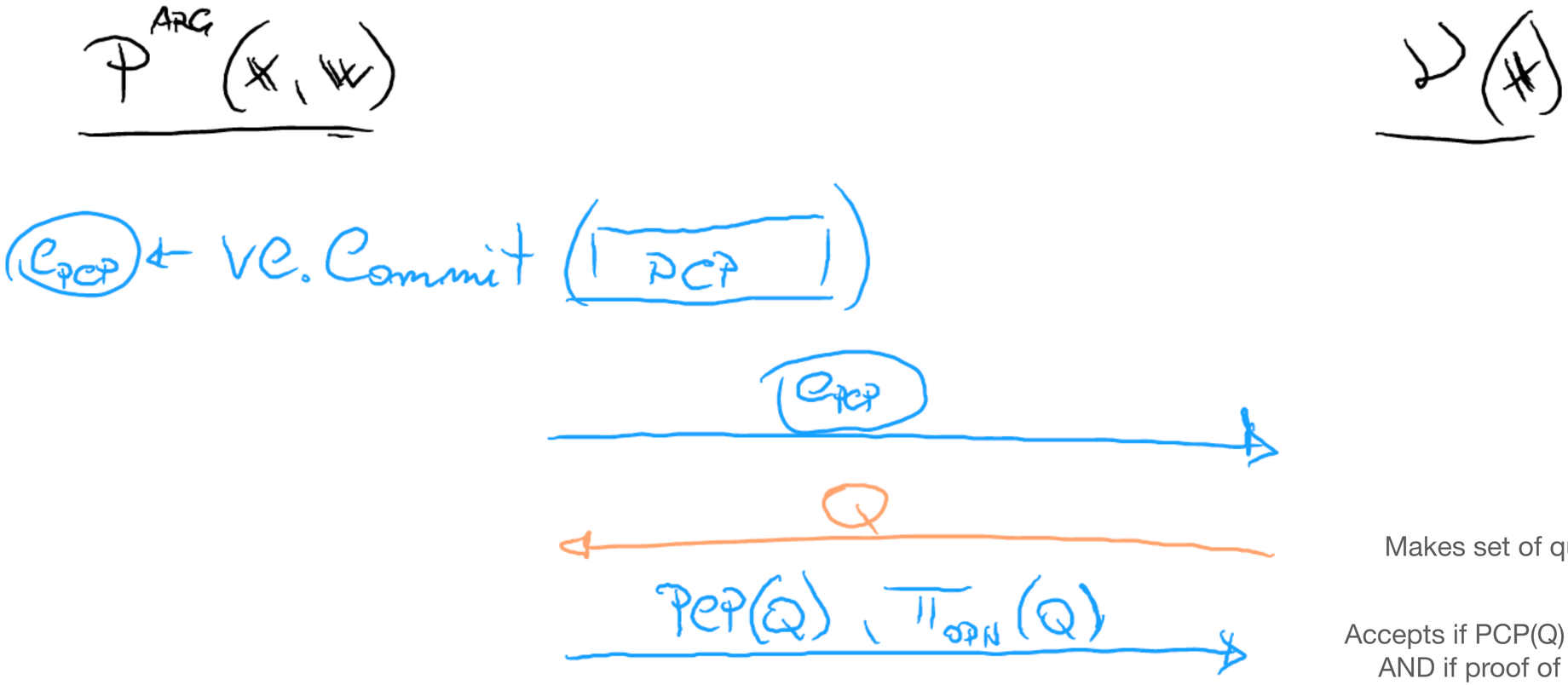


PCP to (Succinct) Interactive Arguments

PCP to (Succinct) Interactive Arguments

- **Vector Commitments:**
  - VC.Commit(v1,…,vn) -> com (short)
  - VC.Open(com, J, (v1, …, vn)) -> prf_opn (J subset of {1,…,n})
  - VC.Verify(com, J, (v_J)) -> 0/1

PCP to (Succinct) Interactive Arguments



Makes set of queries Q

Accepts if PCP(Q) satisfies some test
AND if proof of opening is valid.

~~PCP~~ PHP: an idealization for USRS SNARKs

Polynomially Holographic Proofs

~~PCP~~ PHP: an idealization for USRS SNARKs

Polynomially Holographic Proofs

- **PHPs**: close to AHPs [Marlin,Fractal]/ILDPs [PLONK]/PIOPs [DARK]…

~~PCP~~ PHP: an idealization for USRS SNARKs

Polynomially Holographic Proofs

- **PHPs**: close to AHPs [Marlin,Fractal]/ILDPs [PLONK]/PIOPs [DARK]…
- **Think PCPs but**:

~~PCP~~ PHP: an idealization for USRS SNARKs

Polynomially Holographic Proofs

- **PHPs**: close to AHPs [Marlin,Fractal]/ILDPs [PLONK]/PIOPs [DARK]…
- **Think PCPs but**:
  - **Interactive** (the verifier can sends challenges)

~~PCP~~ PHP: an idealization for USRS SNARKs

Polynomially Holographic Proofs

- **PHPs**: close to AHPs [Marlin,Fractal]/ILDPs [PLONK]/PIOPs [DARK]…
- **Think PCPs but**:
  - **Interactive** (the verifier can sends challenges)
  - “**oracles**”: not strings but polynomials

~~PCP~~ PHP: an idealization for USRS SNARKs

Polynomially Holographic Proofs

- **PHPs**: close to AHPs [Marlin,Fractal]/ILDPs [PLONK]/PIOPs [DARK]…
- **Think PCPs but**:
  - **Interactive** (the verifier can sends challenges)
  - “**oracles**”: not strings but polynomials
  - **Queries**: algebraic properties of these polynomials

~~PCP~~ PHP: an idealization for USRS SNARKs

Polynomially Holographic Proofs

- **PHPs**: close to AHPs [Marlin,Fractal]/ILDPs [PLONK]/PIOPs [DARK]…
- **Think PCPs but**:
  - **Interactive** (the verifier can sends challenges)
  - "**oracles**": not strings but polynomials
  - **Queries**: algebraic properties of these polynomials
- For compilation: ~~Vector~~ Polynomial commitment

What you get from these compilers

Practical* SNARKs with Universal SRS

zkSNARK		size				time	
		[redacted]	$\lvert vk_R \rvert$	$\lvert \pi \rvert$	[redacted]	Prove	Verify
Sonic [46]	$\mathbb{G}_1$		—	20		$273n$	7 pairings
	$\mathbb{G}_2$		3	—		—	
	$\mathbb{F}$		—	16		$O(m \log m)$	$O(\ell + \log m)$
MARLIN [20]	$\mathbb{G}_1$		12	13		$14n+8m$	2 pairings
	$\mathbb{G}_2$		2	—		—	
	$\mathbb{F}$		—	8		$O(m \log m)$	$O(\ell + \log m)$
PLONK (small proof) [28]	$\mathbb{G}_1$		8	7		$11n+11a$	2 pairings
	$\mathbb{G}_2$		1	—		—	
	$\mathbb{F}$		—	7		$O((n+a)\log(n+a))$	$O(\ell + \log(n+a))$
PLONK (fast prover) [28]	$\mathbb{G}_1$		8	9		$9n+9a$	2 pairings
	$\mathbb{G}_2$		1	—		—	
	$\mathbb{F}$		—	7		$O((n+a)\log(n+a))$	$O(\ell + \log(n+a))$

Roughly:
- n: # MUL gates
- a: # ADD gates
- m: # wires

*practical + focus is on O(1) proof size

Compilers for CP-SNARKs

Compilers for CP-SNARKs

- **So far:** very high level picture of compilers to efficient SNARKs

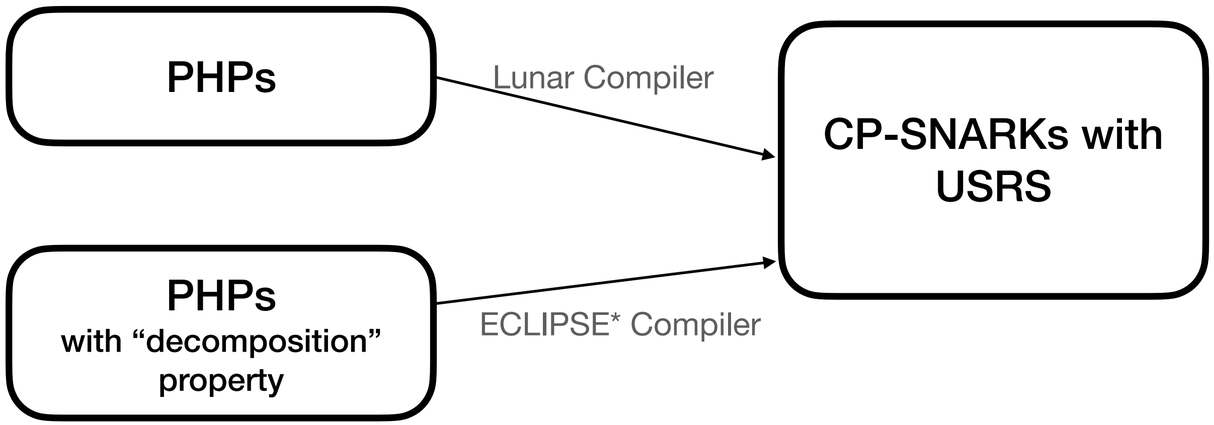
Compilers for CP-SNARKs

- **So far:** very high level picture of compilers to efficient SNARKs
- But what about efficient **CP**-SNARKs? *Do they have such general compilers?*

Compilers for CP-SNARKs

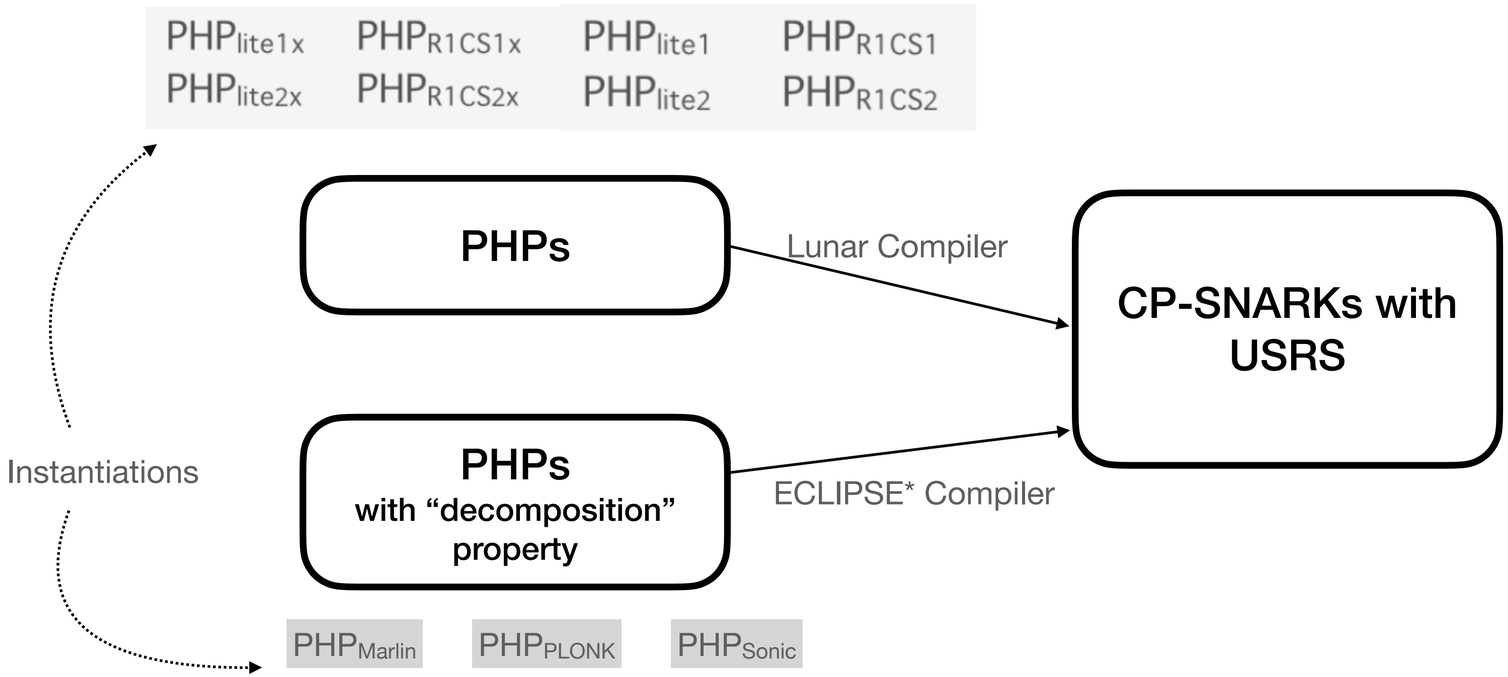
- **So far:** very high level picture of compilers to efficient SNARKs
- But what about efficient **CP**-SNARKs? *Do they have such general compilers?*
- Lunar and ECLIPSE introduce them.

Lunar & ECLIPSE: compilers to USRS CP-SNARKs



*__ECLIPSE__: __E__nhanced __C__ompi__LI__ng method for __P__edersen-committed zk__S__NARK __E__ngines

Lunar & ECLIPSE: compilers to USRS CP-SNARKs

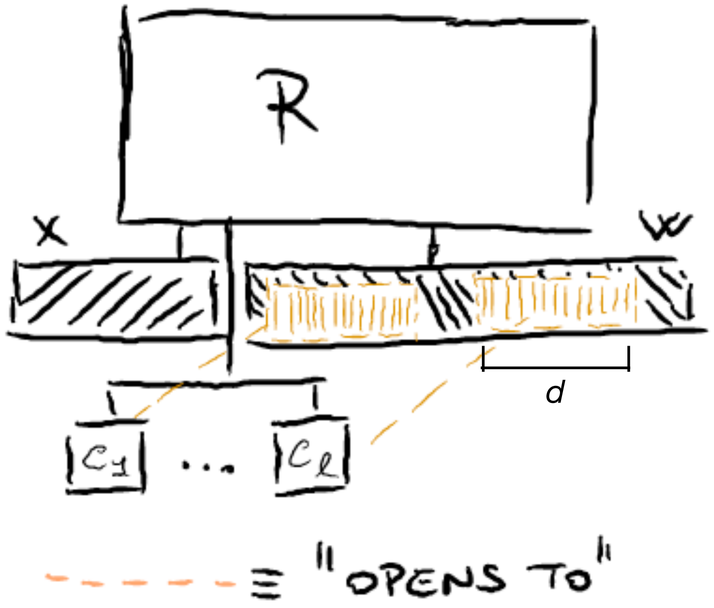


*__ECLIPSE__: __E__nhanced __C__ompi__LI__ng method for __P__edersen-committed zk__S__NARK __E__ngines

Resulting USRS CP-SNARKs: Efficiency

	$\lvert\pi\rvert$	Prove (time)	Verify (time)
ECLIPSE [ABC+21]	$O(\log(\ell \cdot d))$	$O(n + \ell \cdot d)$	$O(\ell \cdot d)$
Lunar [CFF+20]	$O(\ell)$	$O(n + \ell \cdot d)$	$O(\ell)$
LegoUAC [CFQ19]	$O\left(\ell \log^2(n)\right)$	$O(n) + \ell \cdot O(d)$	$O\left(\ell \log^2(n)\right)$

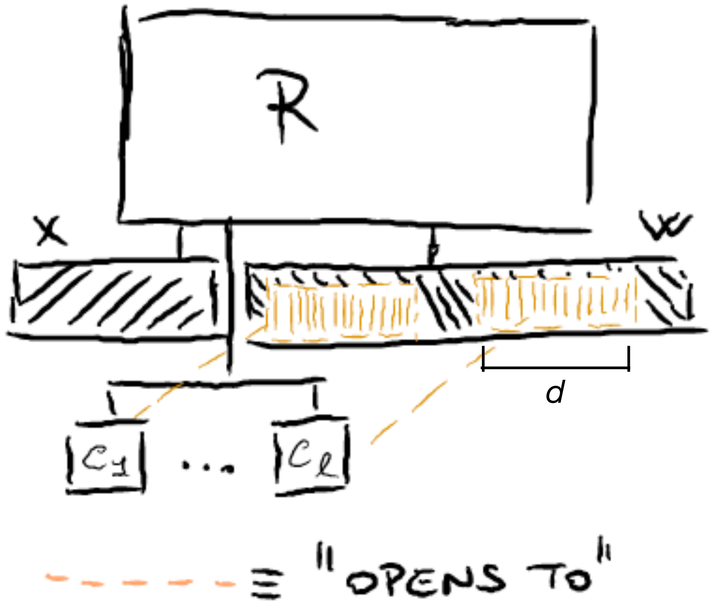
Time is in group operations. Above, n is roughly # of multiplication gates



Resulting USRS CP-SNARKs: Efficiency

	$\lvert\pi\rvert$	Prove (time)	Verify (time)
ECLIPSE [ABC+21]	$O(\log(\ell \cdot d))$	$O(n + \ell \cdot d)$	$O(\ell \cdot d)$
Lunar [CFF+20]	$O(\ell)$	$O(n + \ell \cdot d)$	$O(\ell)$
LegoUAC [CFQ19]	$O(\ell \log^2(n))$	$O(n) + \ell \cdot O(d)$	$O(\ell \log^2(n))$

Time is in group operations. Above, n is roughly # of multiplication gates



In practice the two family of systems show a tradeoff in verification time/proof size.

Remainder of this talk

Remainder of this talk

- Mostly: a high-level view of these compilers

Remainder of this talk

- Mostly: a high-level view of these compilers
- **Roadmap:**

Remainder of this talk

- Mostly: a high-level view of these compilers
- **Roadmap:**
  - Compilers from PHP to SNARKs

Remainder of this talk

- Mostly: a high-level view of these compilers
- **Roadmap:**
  - Compilers from PHP to SNARKs
  - The tweak to allow compilation CP-SNARK (Lunar compiler)

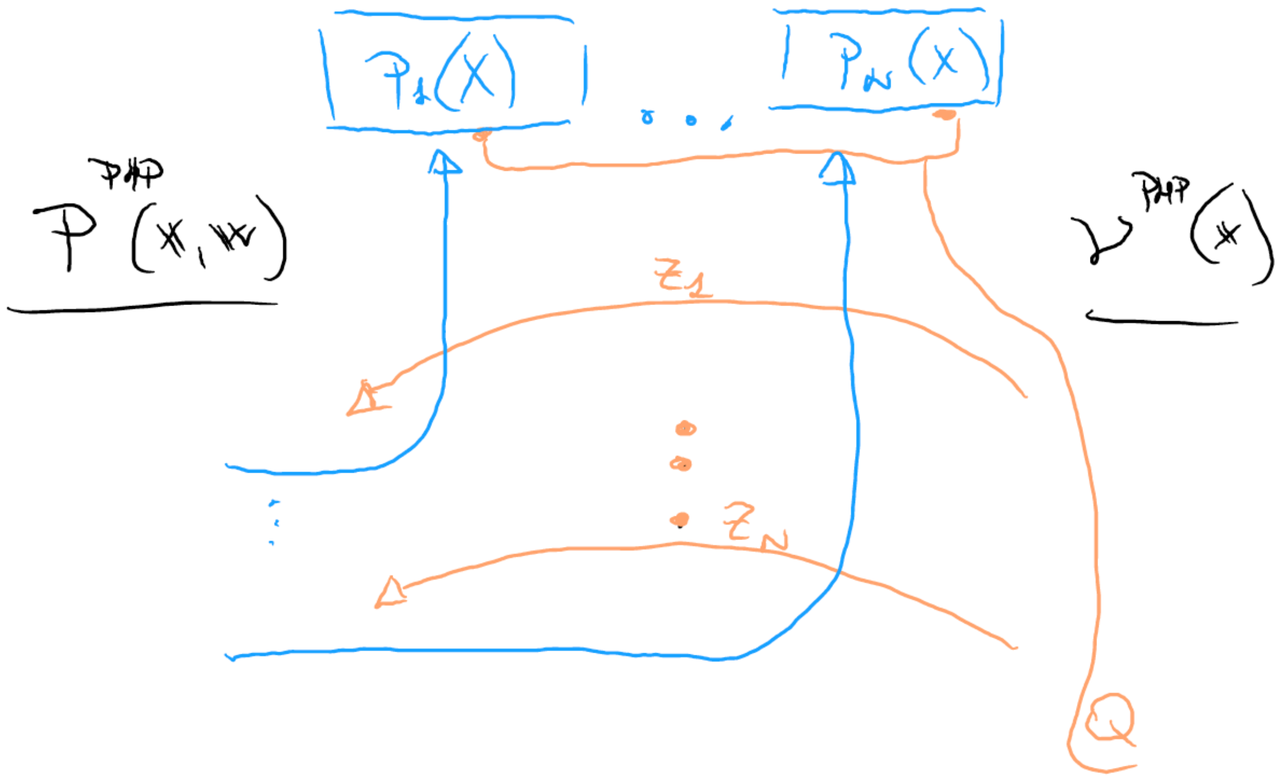
Remainder of this talk

- Mostly: a high-level view of these compilers
- **Roadmap:**
  - Compilers from PHP to SNARKs
  - The tweak to allow compilation CP-SNARK (Lunar compiler)
  - The “decomposition” property in ECLIPSE

Remainder of this talk

- Mostly: a high-level view of these compilers
- **Roadmap:**
  - Compilers from PHP to SNARKs
  - The tweak to allow compilation CP-SNARK (Lunar compiler)
  - The “decomposition” property in ECLIPSE
  - A couple comments on techniques

PHPs



Queries Q:
General properties of polynomials
(many checkable by evaluation in random point)

Examples:

$$\deg(p_2(X)) < D_{\text{Bound}}$$

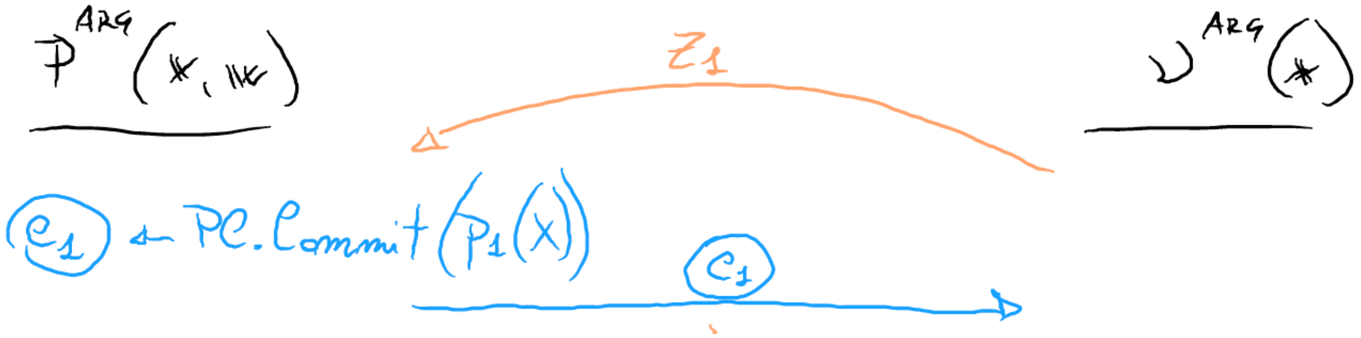
$$p_1(\beta) + \alpha p_3(\beta) p_4(\beta) = 0$$

(for $\alpha, \beta \leftarrow_{\$} \mathbb{F}$)

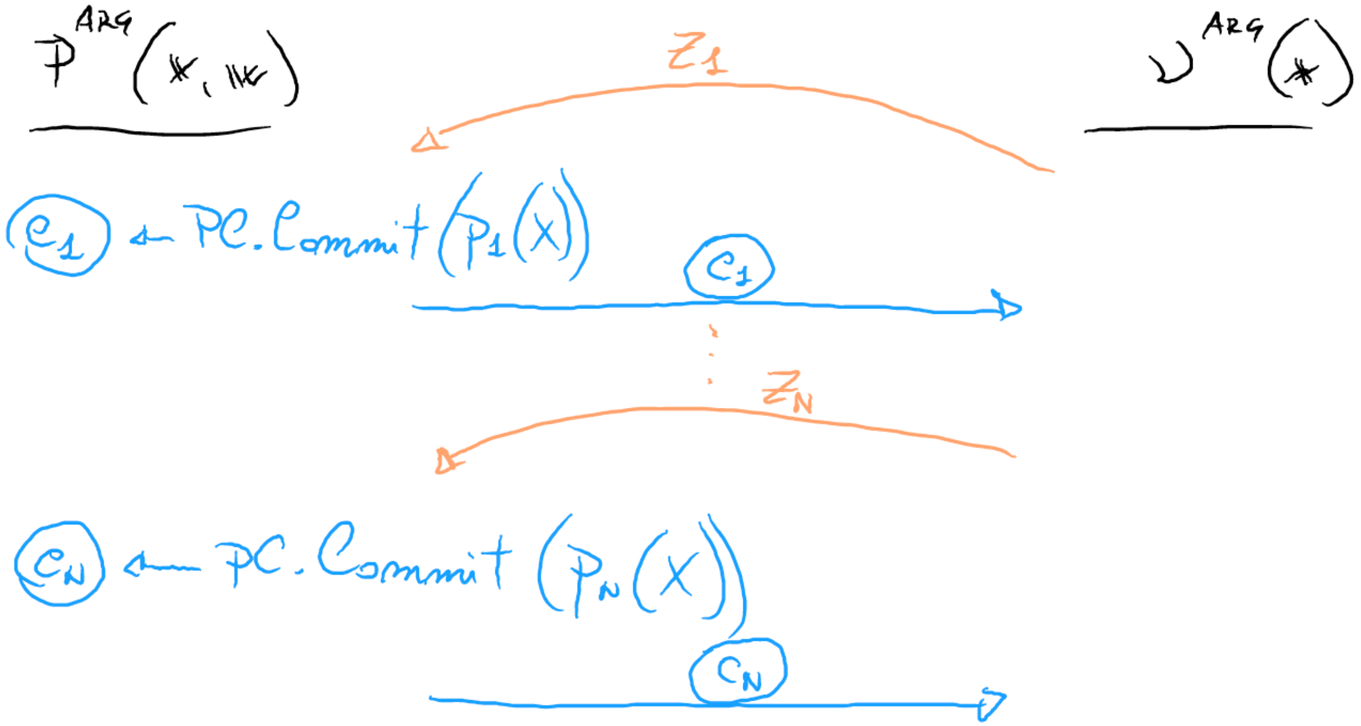
Compiling to USRS SNARKs: Ingredients

- Compiler in Marlin/DARK/Lunar/PLONK
- Main tool Polynomial Commitments PC:
  - A *compressing* commitment to polynomials
  - Allows proving efficiently (and succinctly) in ZK:
    - p(x) = y (evaluation)
    - deg(p) <= Dbound
    - Others…

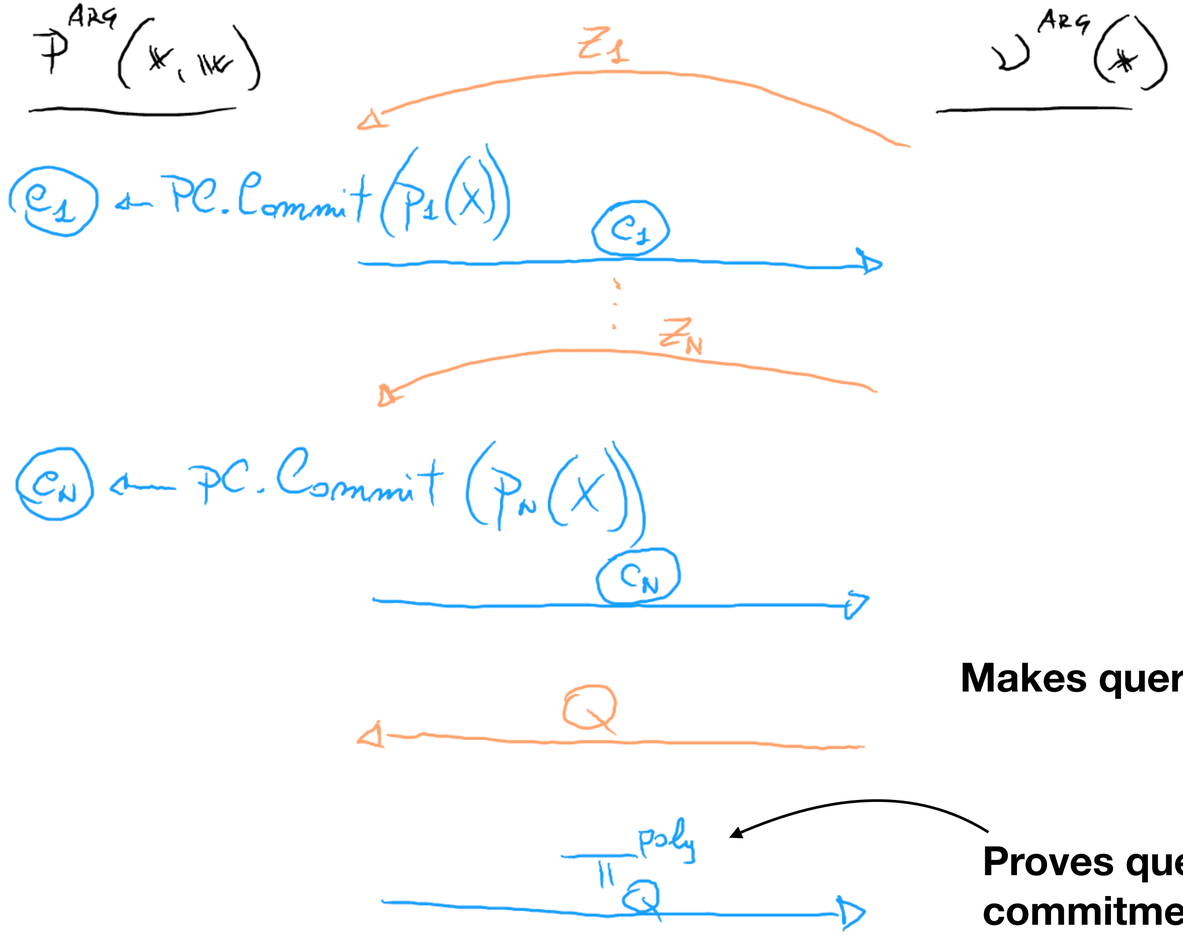
Compiling to USRS SNARKs



Compiling to USRS SNARKs



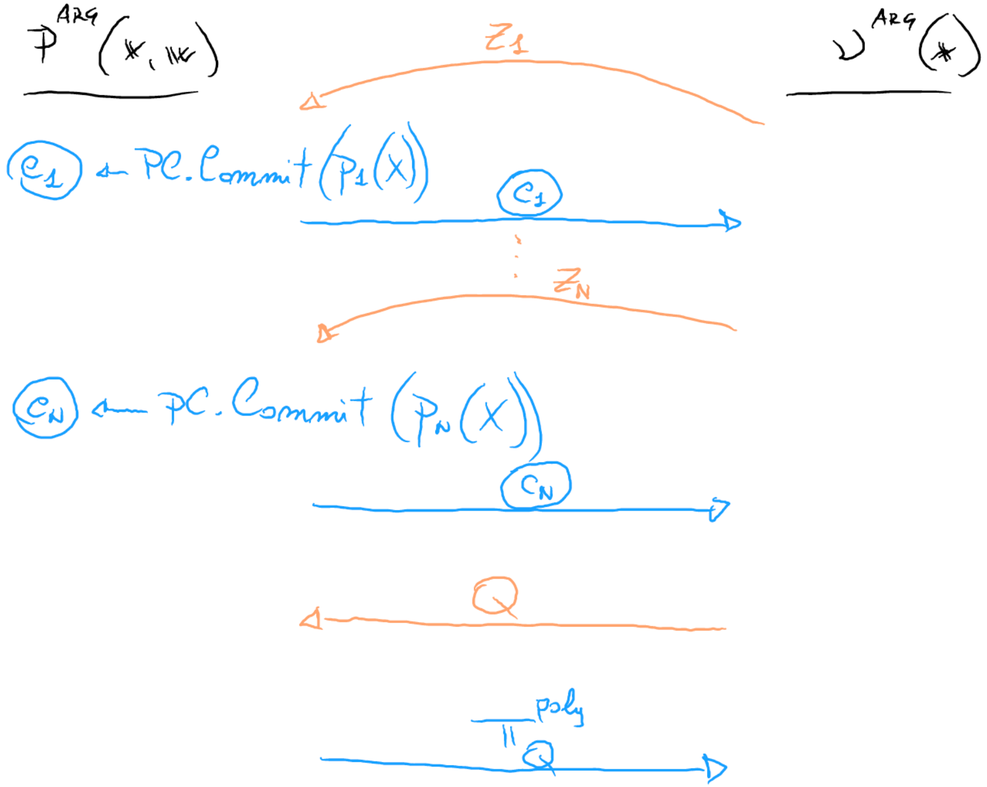
Compiling to USRS SNARKs



Makes queries Q.

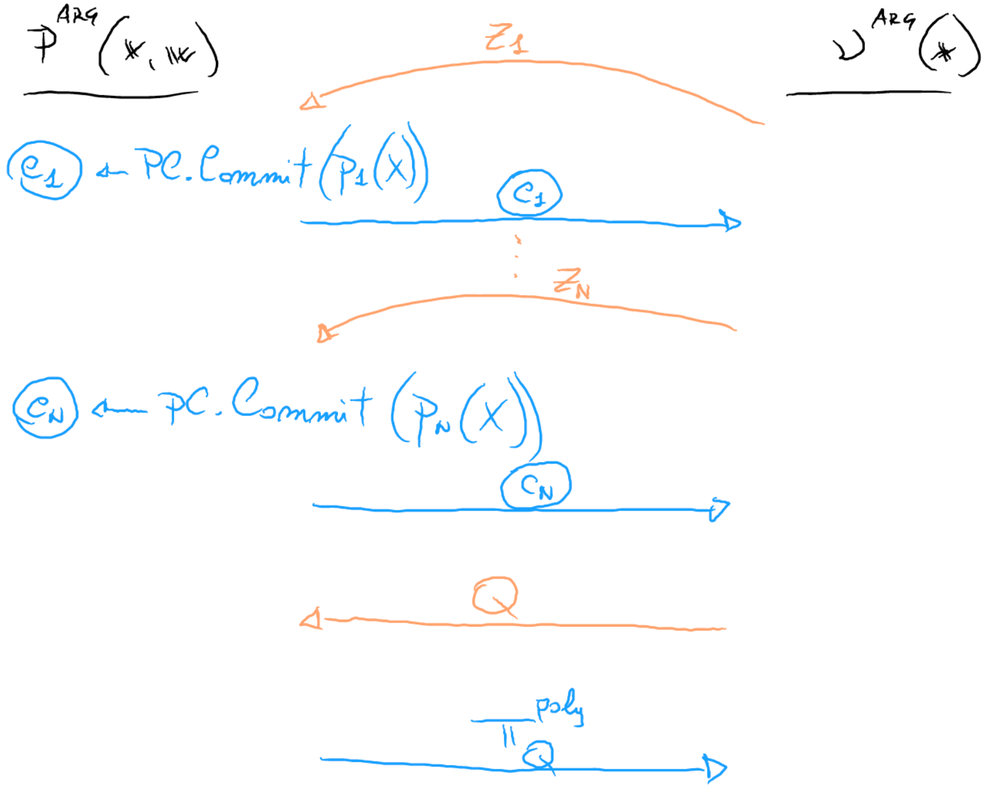
Proves queries Q are satisfied by poly commitments c1,…,cN

The Resulting USRS SNARKs



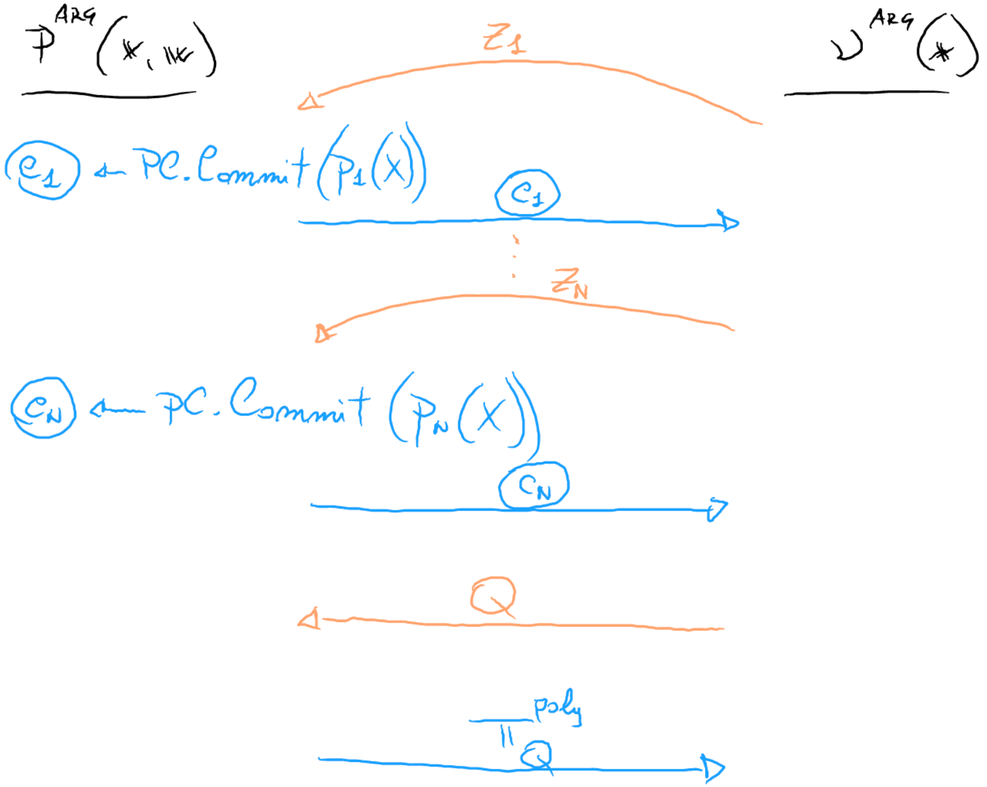
The Resulting USRS SNARKs

- Use Fiat-Shamir for non-interaction



The Resulting USRS SNARKs

- Use Fiat-Shamir for non-interaction
- Why is the SRS Universal?

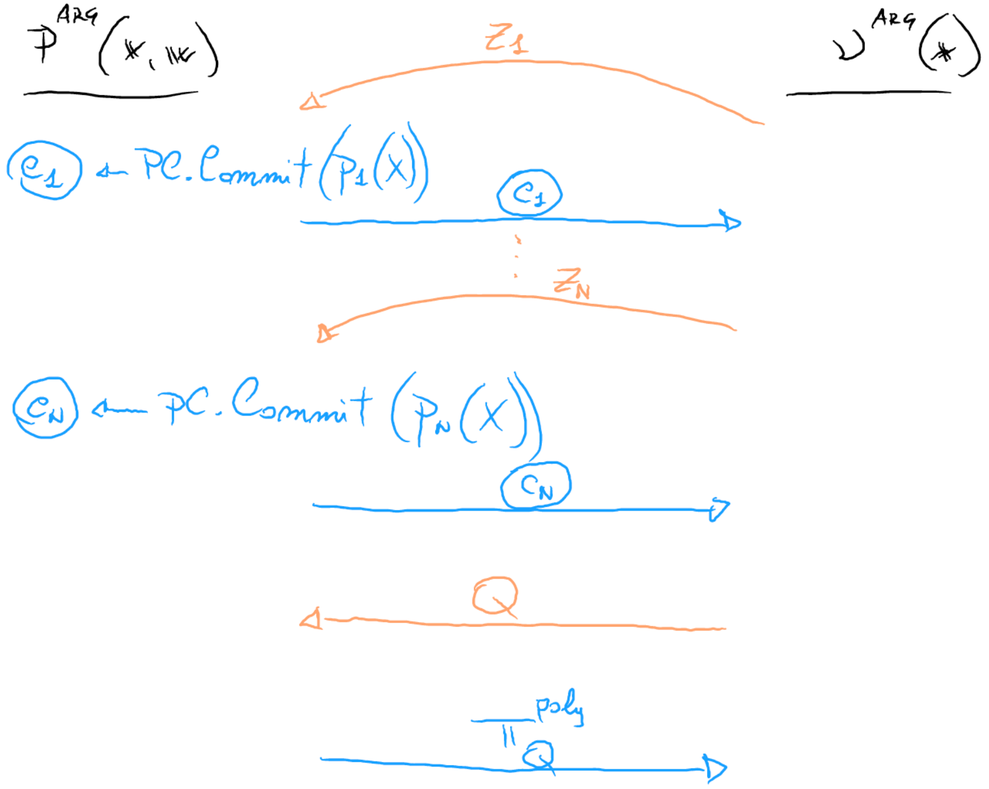


The Resulting USRS SNARKs

- Use Fiat-Shamir for non-interaction
- Why is the SRS Universal?
  - Because we can define

    SNARK.Setup(maxSize) ->

    srs_gen := PC.Setup(maxPolyDeg)



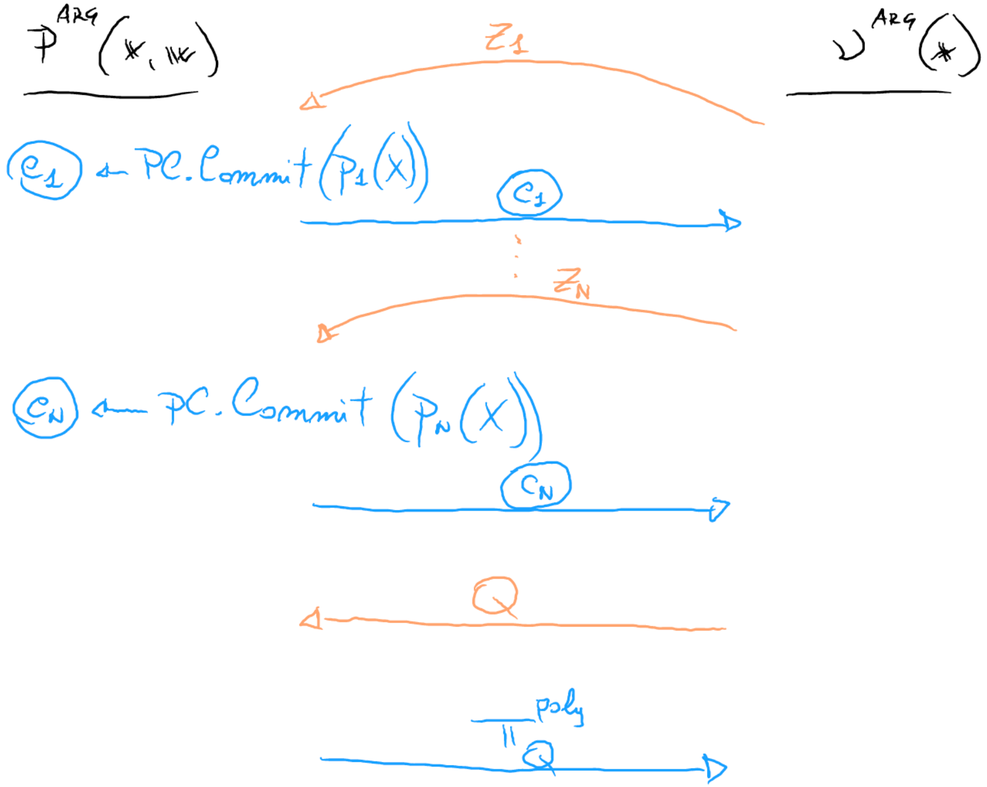
The Resulting USRS SNARKs

- Use Fiat-Shamir for non-interaction
- Why is the SRS Universal?
  - Because we can define

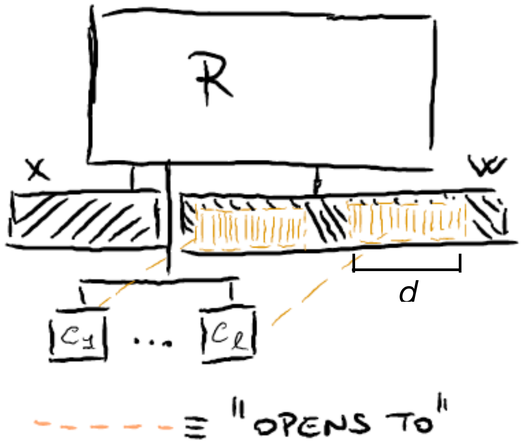
    SNARK.Setup(maxSize) ->

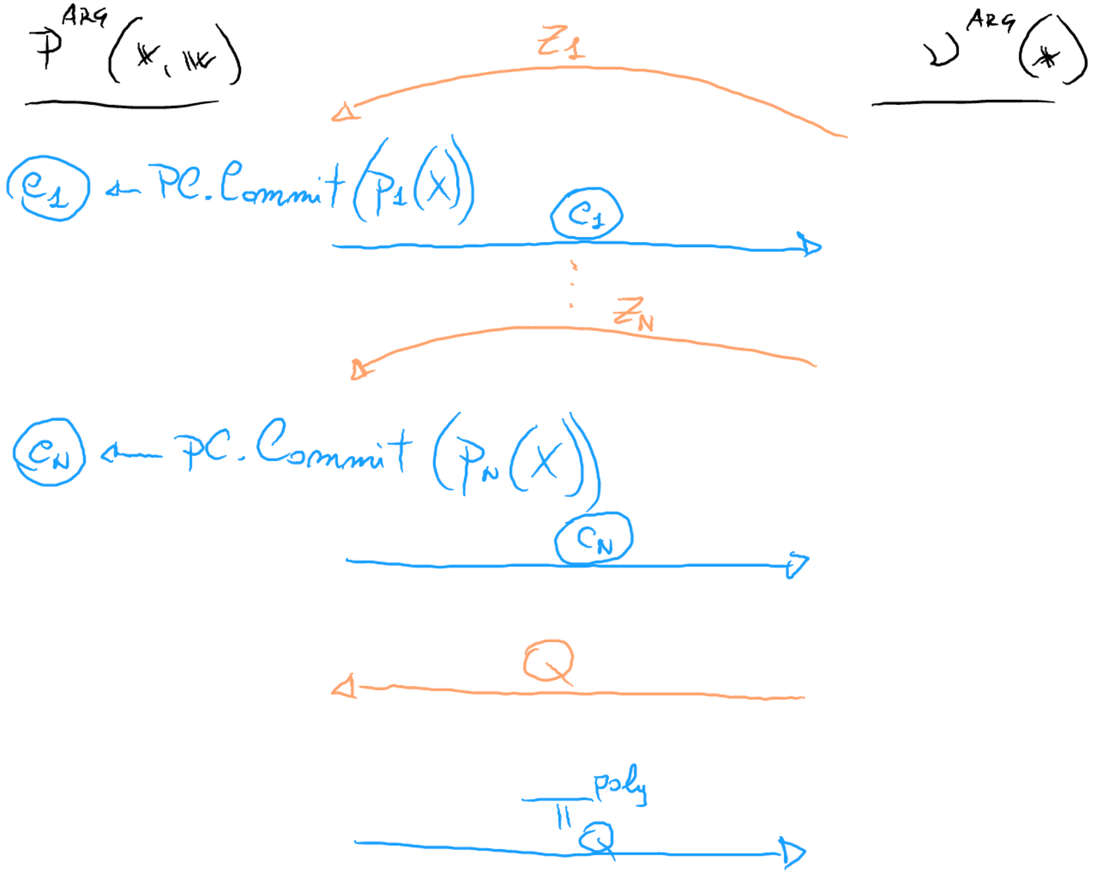
    srs_gen := PC.Setup(maxPolyDeg)

  - Where maxPolyDeg depends on maxSize

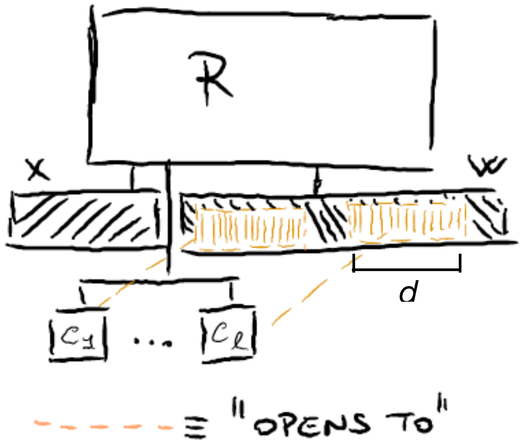


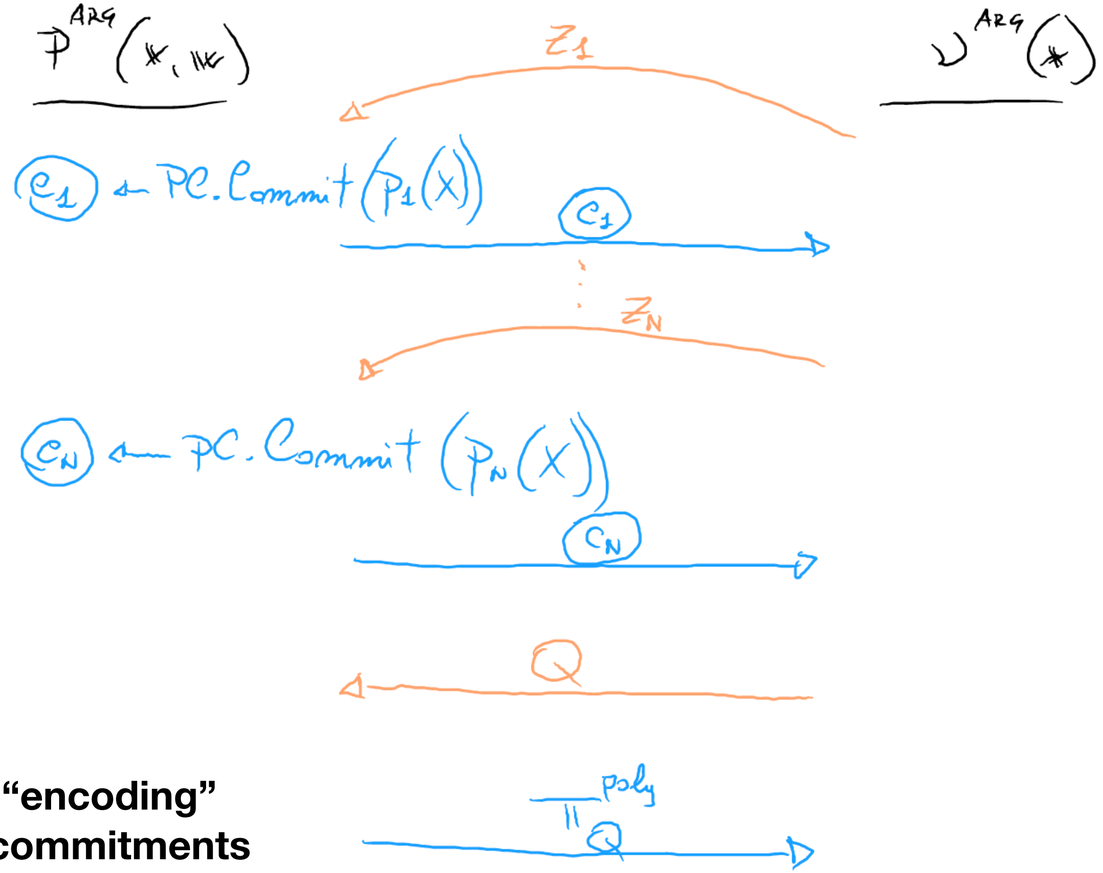
But that was SNARKs. How ‘bout CP-SNARKs?





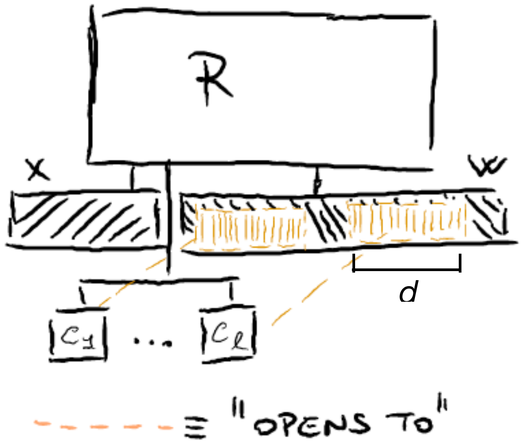
But that was SNARKs. How ‘bout CP-SNARKs?

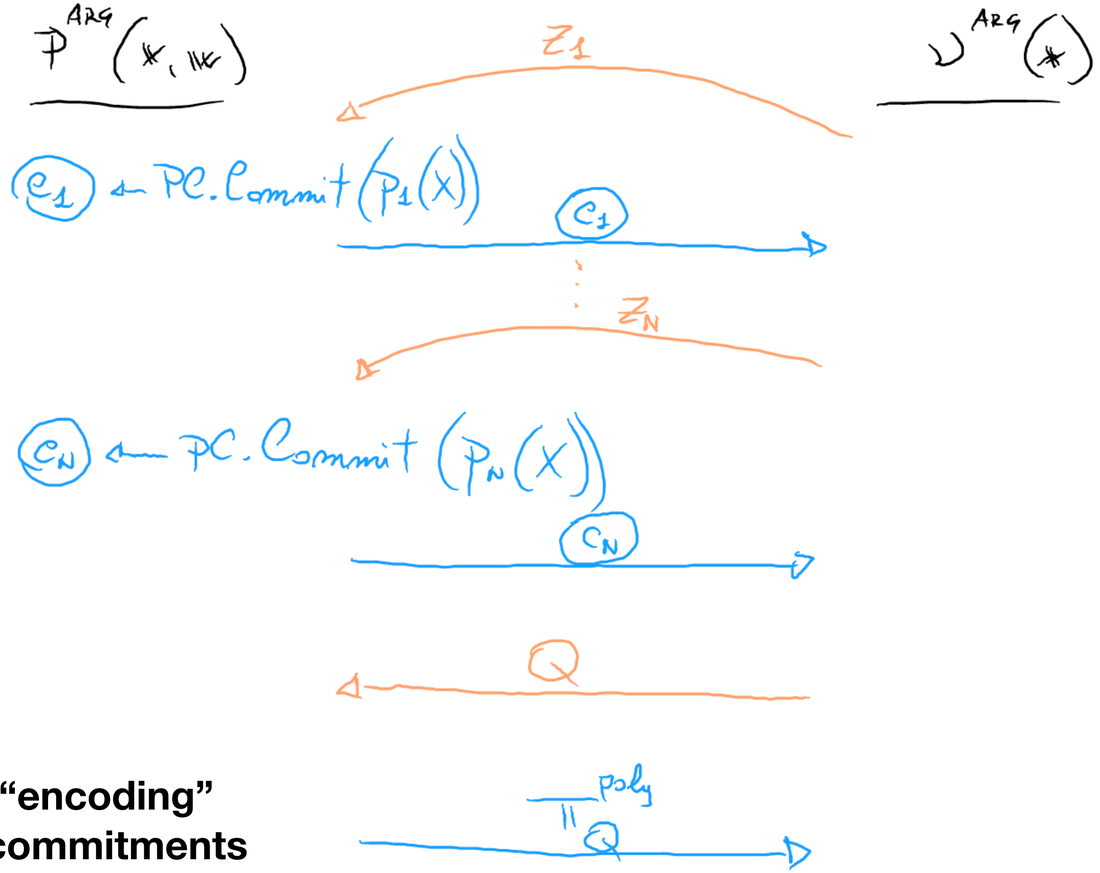




Wait!
These are commitments to polynomials “encoding” the witness. Can’t we just reuse them as commitments for commit and prove?

But that was SNARKs. How ‘bout CP-SNARKs?

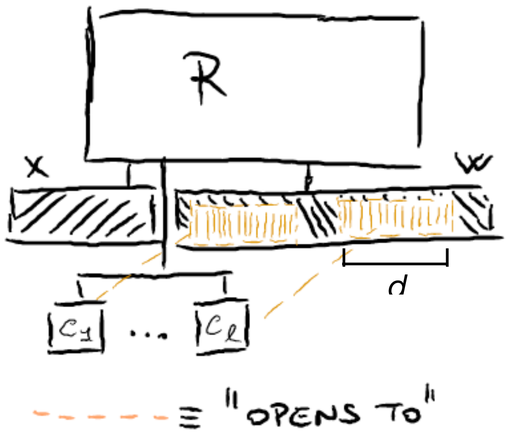


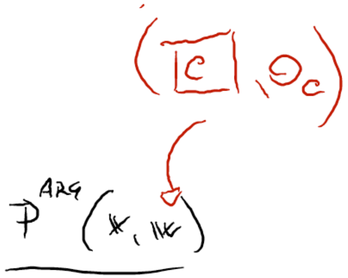


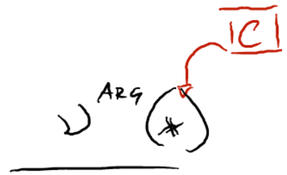
Wait!
These are commitments to polynomials “encoding” the witness. Can’t we just reuse them as commitments for commit and prove?

No. Reusing them would break ZK.

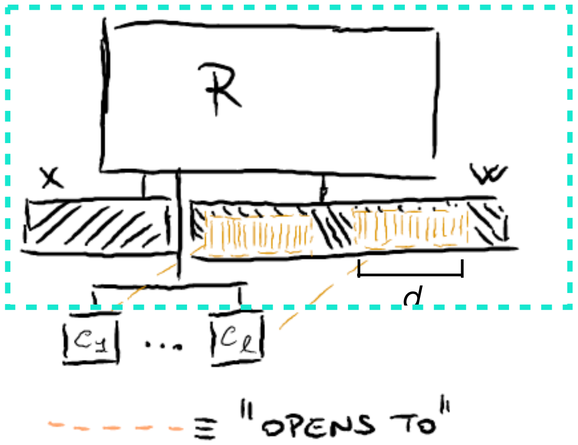
Compiling to USRS CP-SNARKs

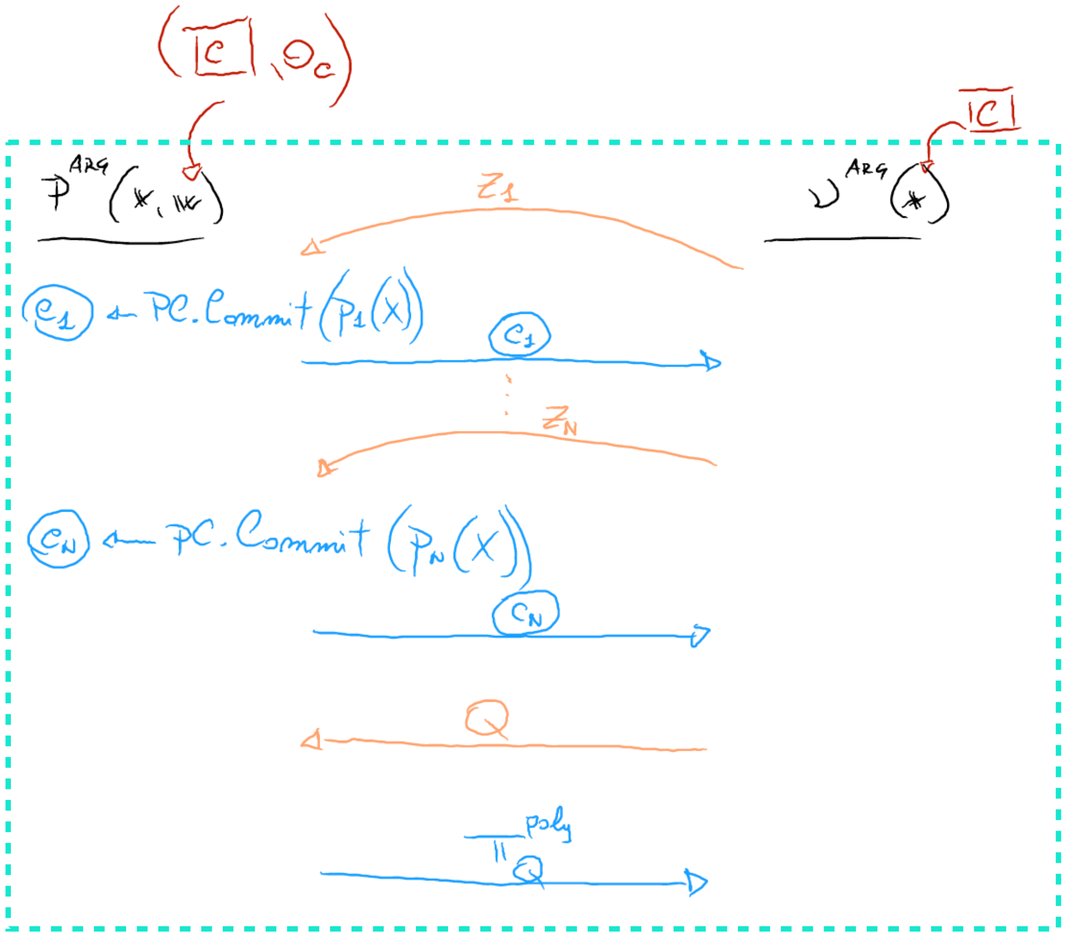




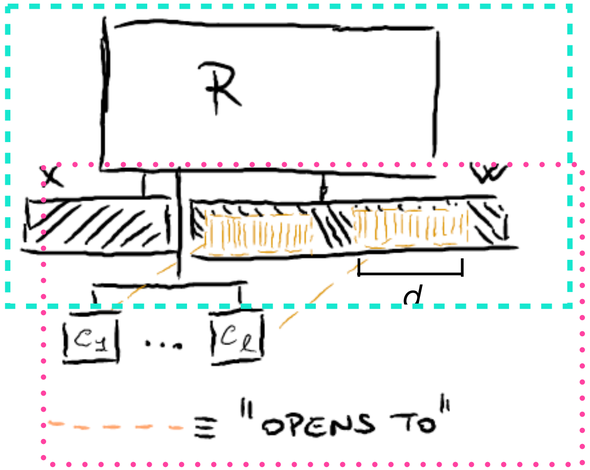


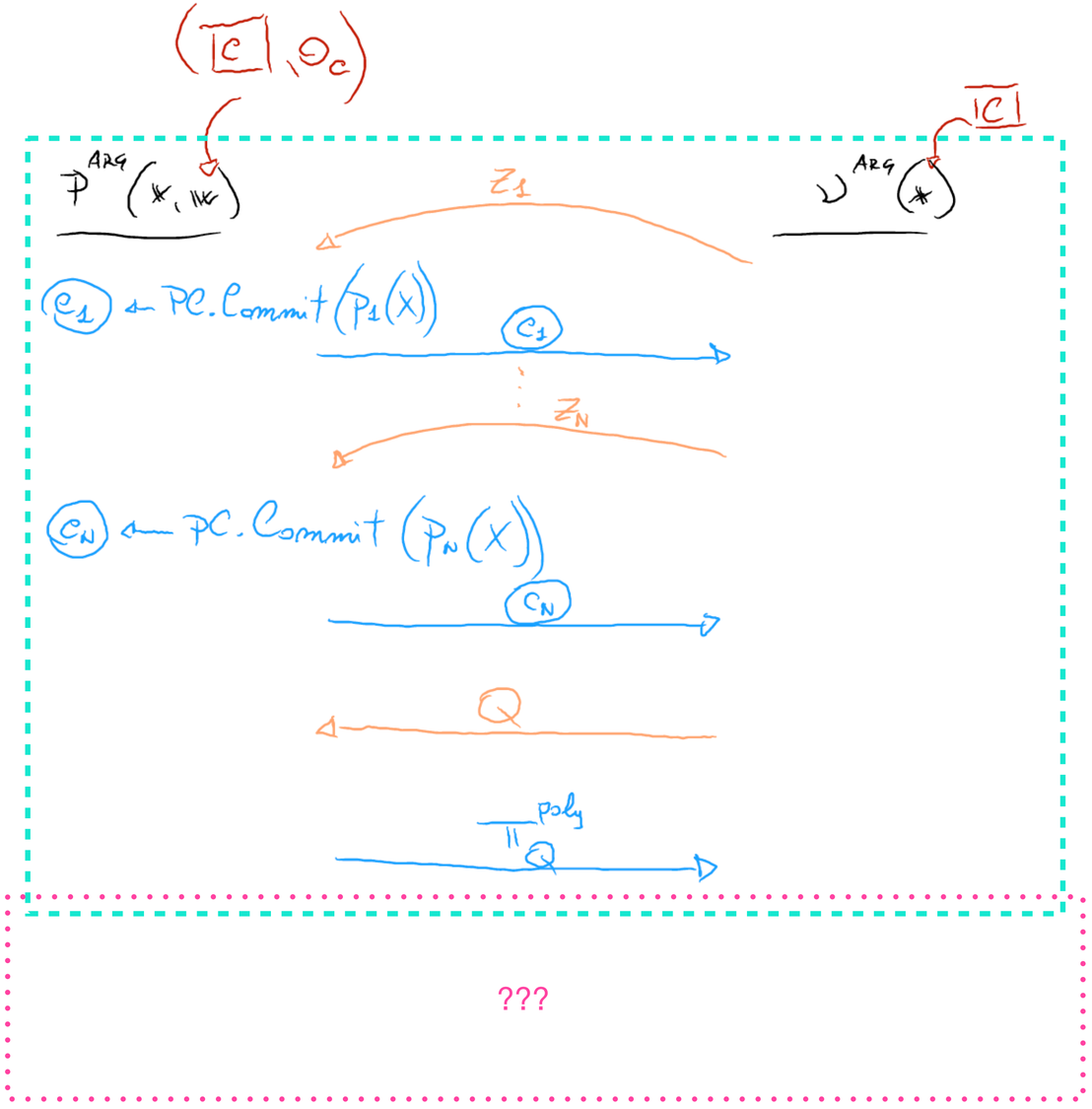
Compiling to USRS CP-SNARKs



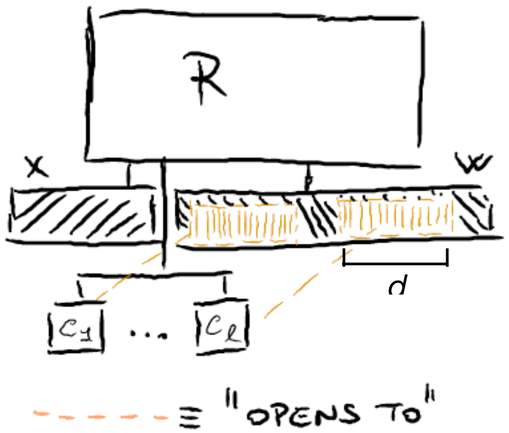


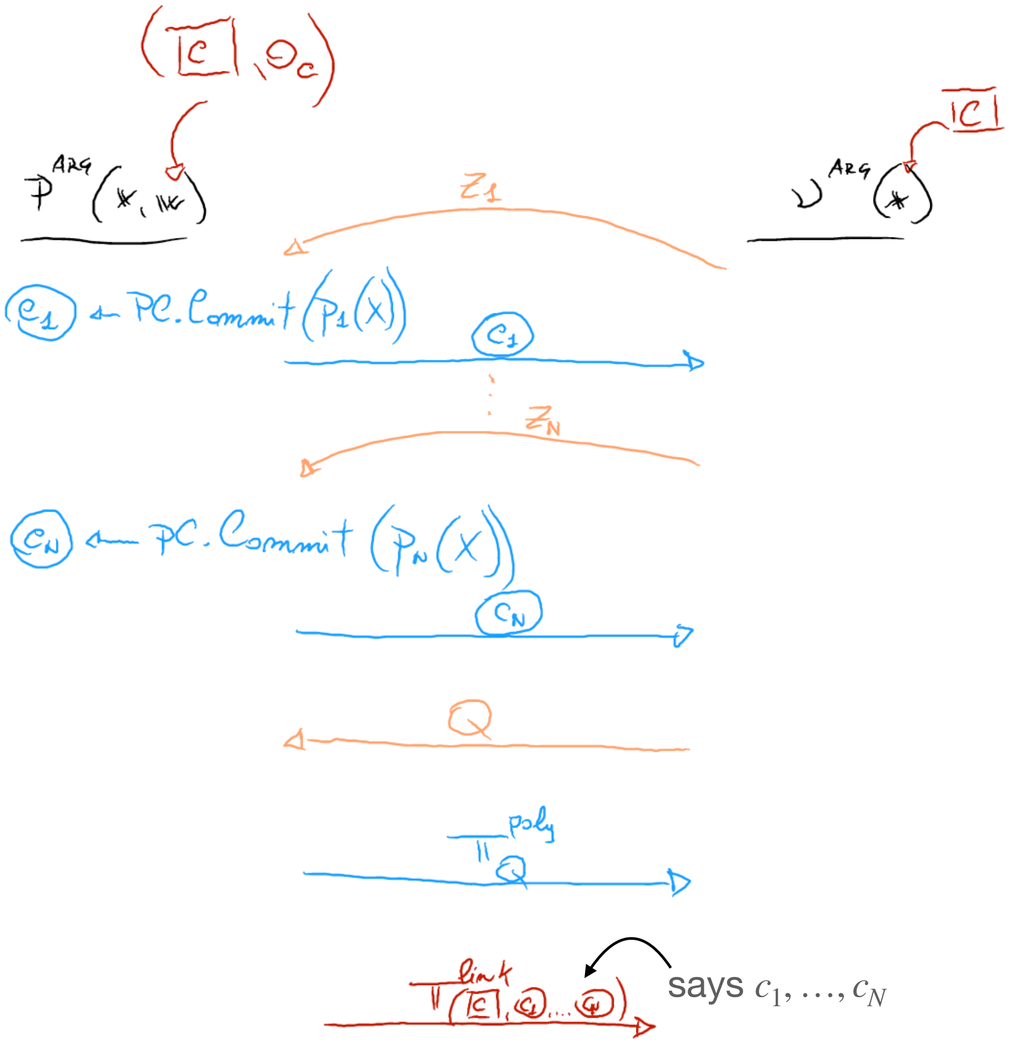
Compiling to USRS CP-SNARKs



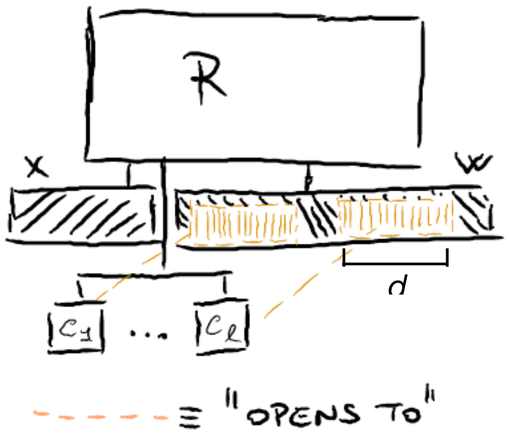


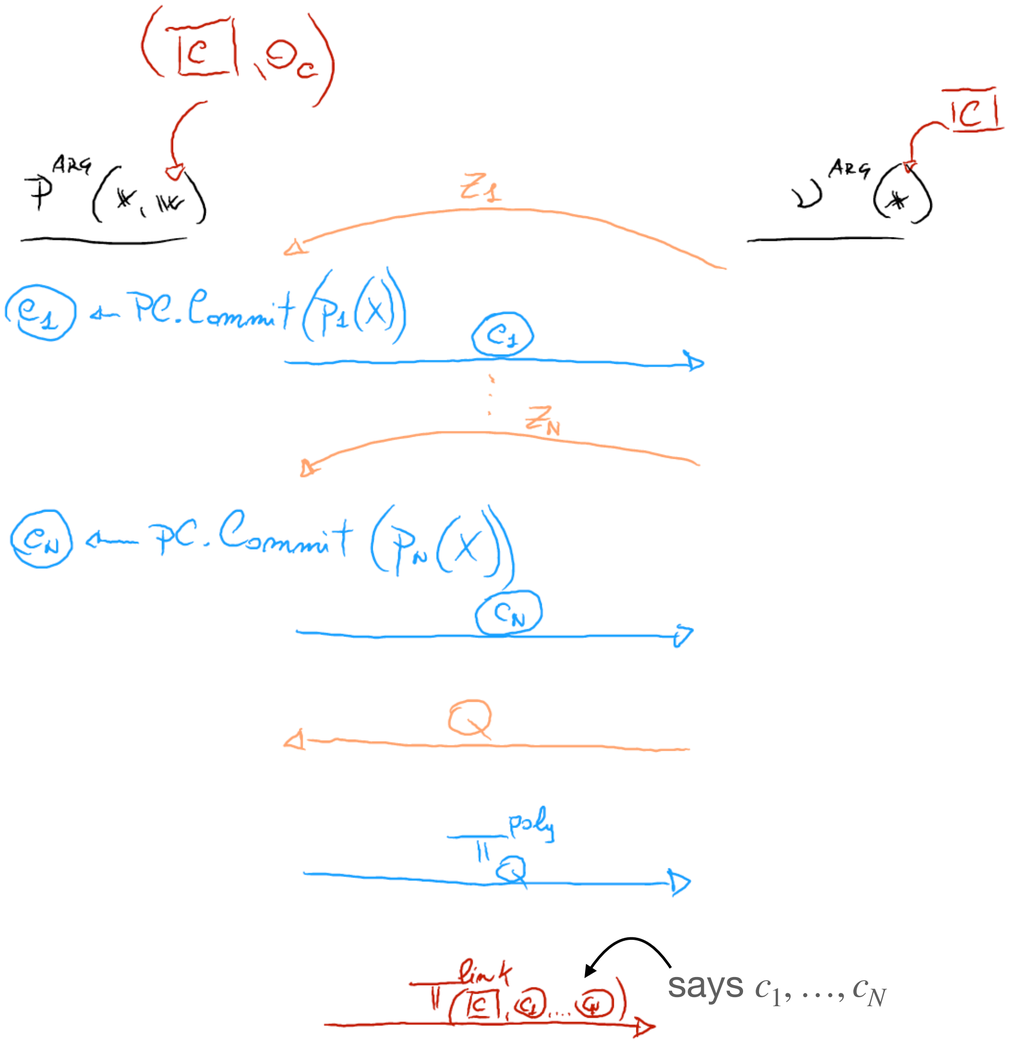
Compiling to USRS CP-SNARKs (Lunar compiler)





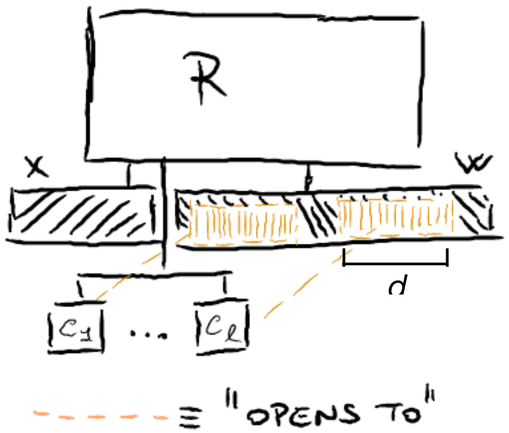
Compiling to USRS CP-SNARKs (Lunar compiler)

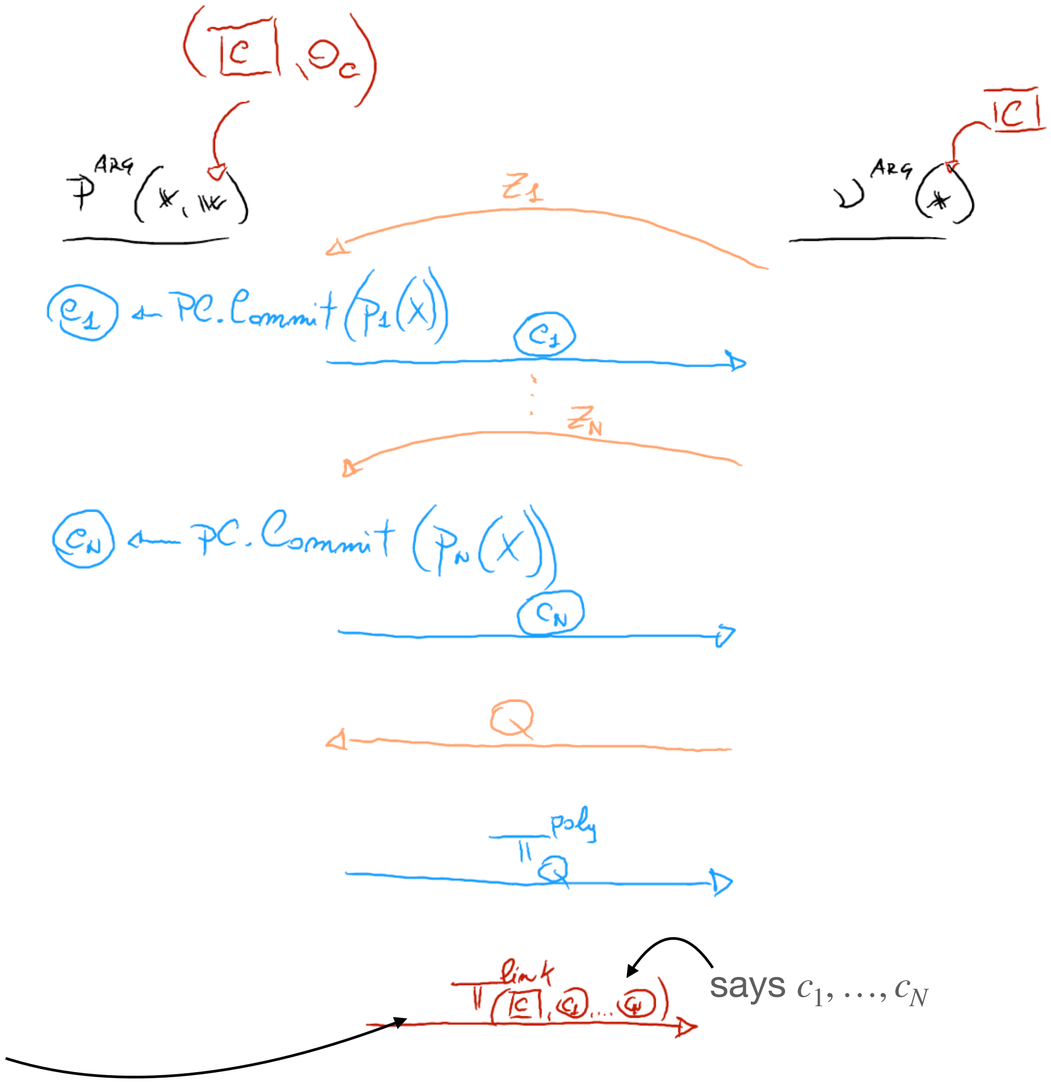




Assume "special extractability":
there exists WitExtr s.t.
w = WitExtr(p1(X)… pN(X))

Compiling to USRS CP-SNARKs (Lunar compiler)

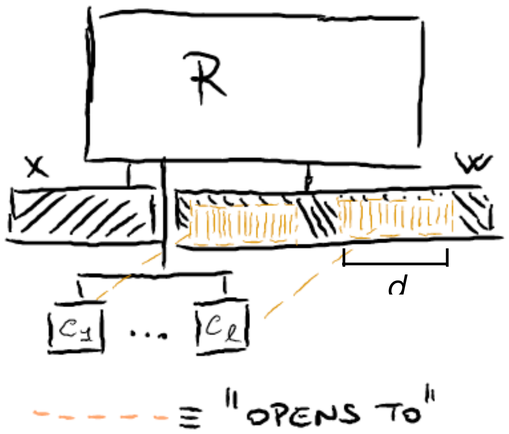




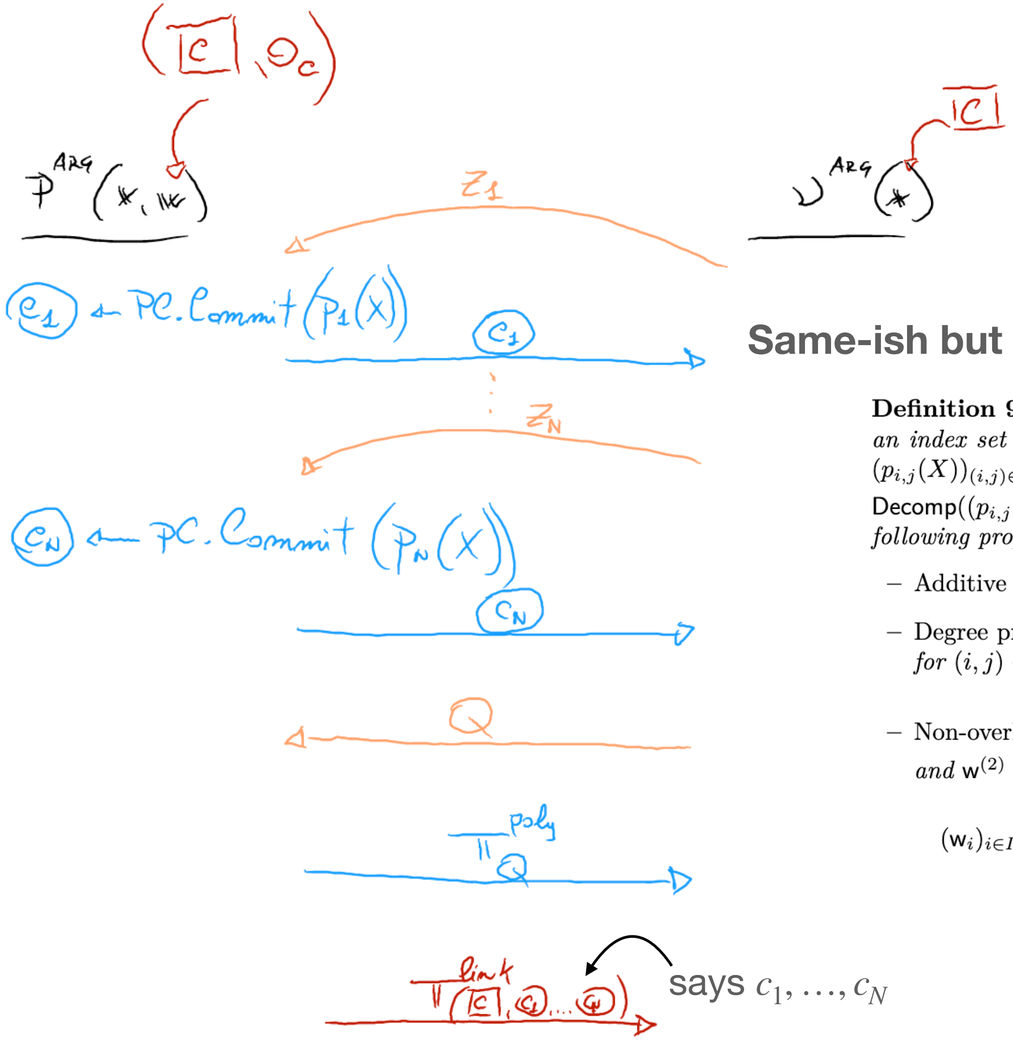
Assume "special extractability":
there exists WitExtr s.t.
w = WitExtr(p1(X)… pN(X))

Specifically proves:
opn([c]) = someSubset(w)
&& opn(c_i) = p_i &&
w = WitExtr(p1(X)… pN(X))

Compiling to USRS CP-SNARKs (ECLIPSE compiler)



Assume "special extractability":
there exists WitExtr s.t.
w = WitExtr(p1(X)… pN(X))



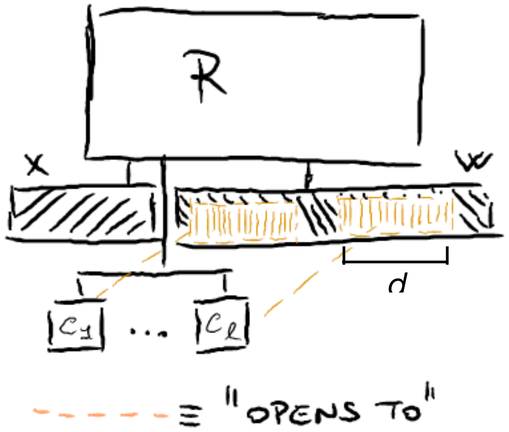
Same-ish but assumes special property on WitExt:

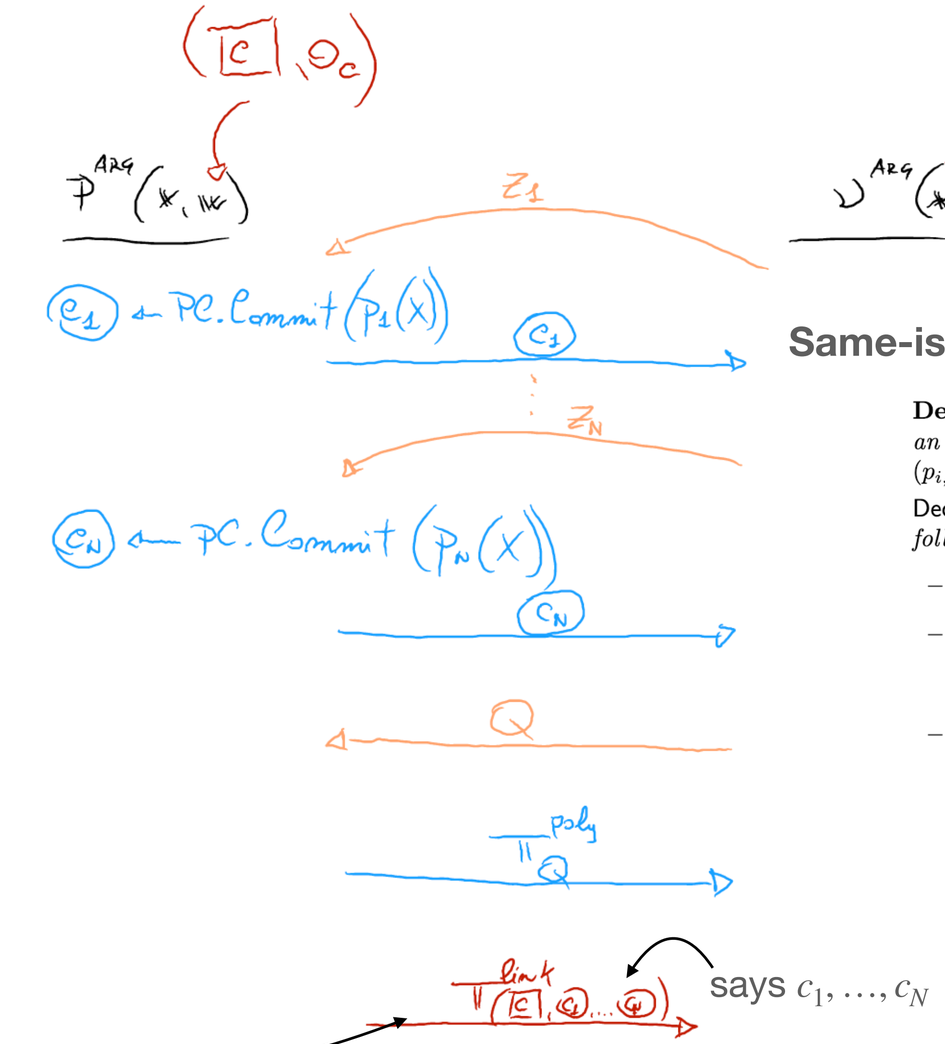
Definition 9 (Decomposable witness-carrying polynomials). *Let W be an index set of witness-carrying polynomials of* AHP*. We say that polynomials $(p_{i,j}(X))_{(i,j)\in W}$ of* AHP *are* decomposable *if there exists an efficient function* $\mathsf{Decomp}((p_{i,j}(X))_{(i,j)\in W}, I) \to (p^{(1)}_{i,j}(X), p^{(2)}_{i,j}(X))_{(i,j)\in W}$ *such that it satisfies the following properties for any $I \subset [n]$.*

- Additive decomposition: $p_{i,j}(X) = p^{(1)}_{i,j}(X) + p^{(2)}_{i,j}(X)$ *for* $(i,j) \in W$.
- Degree preserving: $\deg(p^{(1)}_{i,j}(X))$ *and* $\deg(p^{(2)}_{i,j}(X))$ *are at most* $\deg(p_{i,j}(X))$ *for* $(i,j) \in W$.
- Non-overlapping: *Let* $\mathsf{w} = \mathsf{WitExt}((p_{i,j}(X))_{(i,j)\in W})$, $\mathsf{w}^{(1)} = \mathsf{WitExt}((p^{(1)}_{i,j}(X))_{(i,j)\in}$ *and* $\mathsf{w}^{(2)} = \mathsf{WitExt}((p^{(2)}_{i,j}(X))_{(i,j)\in W})$. *Then*

$$(\mathsf{w}_i)_{i\in I} = (\mathsf{w}^{(1)}_i)_{i\in I} \quad (\mathsf{w}_i)_{i\notin I} = (\mathsf{w}^{(2)}_i)_{i\notin I} \quad (\mathsf{w}^{(1)}_i)_{i\notin I} = 0 \quad (\mathsf{w}^{(2)}_i)_{i\in I} = 0$$

Compiling to USRS CP-SNARKs (ECLIPSE compiler)





Assume "special extractability":
there exists WitExtr s.t.
w = WitExtr(p1(X)… pN(X))

Specifically proves:
opn([c]) = someSubset(w)
&& opn(c_i) = p_i &&
w = WitExtr(p1(X)… pN(X))

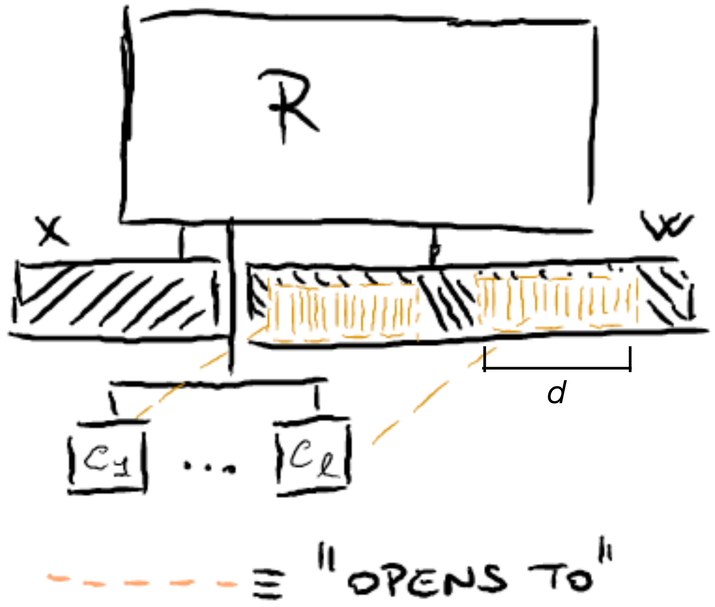
Same-ish but assumes special property on WitExt:

Definition 9 (Decomposable witness-carrying polynomials). *Let W be an index set of witness-carrying polynomials of* AHP*. We say that polynomials $(p_{i,j}(X))_{(i,j)\in W}$ of* AHP *are* decomposable *if there exists an efficient function* $\mathsf{Decomp}((p_{i,j}(X))_{(i,j)\in W}, I) \to (p^{(1)}_{i,j}(X), p^{(2)}_{i,j}(X))_{(i,j)\in W}$ *such that it satisfies the following properties for any $I \subset [n]$.*

- Additive decomposition: $p_{i,j}(X) = p^{(1)}_{i,j}(X) + p^{(2)}_{i,j}(X)$ *for* $(i,j) \in W$.
- Degree preserving: $\deg(p^{(1)}_{i,j}(X))$ *and* $\deg(p^{(2)}_{i,j}(X))$ *are at most* $\deg(p_{i,j}(X))$ *for* $(i,j) \in W$.
- Non-overlapping: *Let* $\mathsf{w} = \mathsf{WitExt}((p_{i,j}(X))_{(i,j)\in W})$, $\mathsf{w}^{(1)} = \mathsf{WitExt}((p^{(1)}_{i,j}(X))_{(i,j)\in}$ *and* $\mathsf{w}^{(2)} = \mathsf{WitExt}((p^{(2)}_{i,j}(X))_{(i,j)\in W})$. *Then*

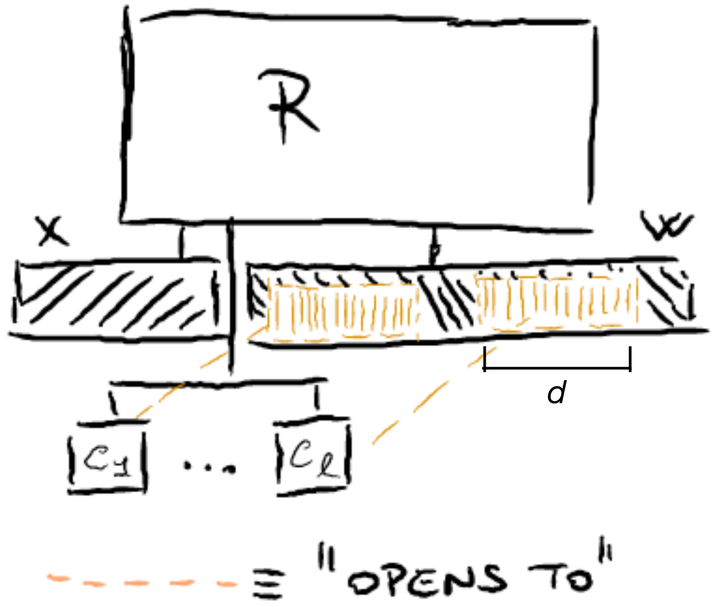
$$(\mathsf{w}_i)_{i\in I} = (\mathsf{w}^{(1)}_i)_{i\in I} \quad (\mathsf{w}_i)_{i\notin I} = (\mathsf{w}^{(2)}_i)_{i\notin I} \quad (\mathsf{w}^{(1)}_i)_{i\notin I} = 0 \quad (\mathsf{w}^{(2)}_i)_{i\in I} = 0$$

The “Linking” component in Lunar



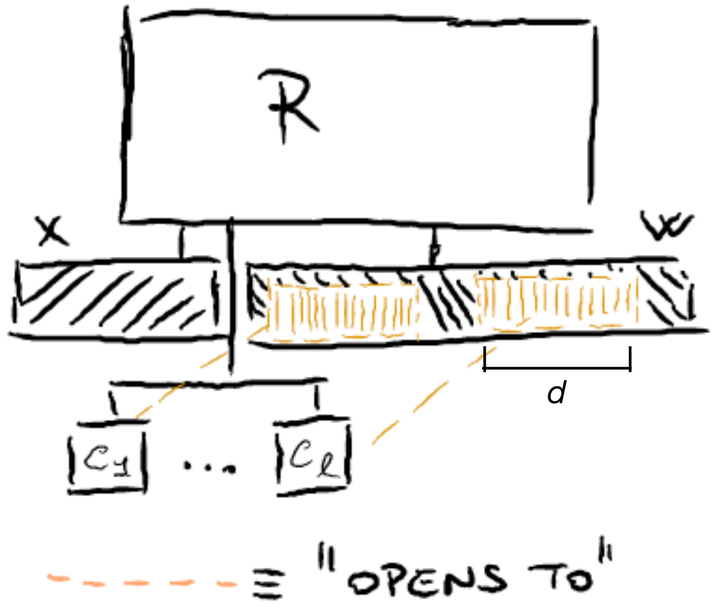
The “Linking” component in Lunar

- **Challenge:** the opening of *input* commitments are “shifted” in some way inside w (encode through the transcript commitments)



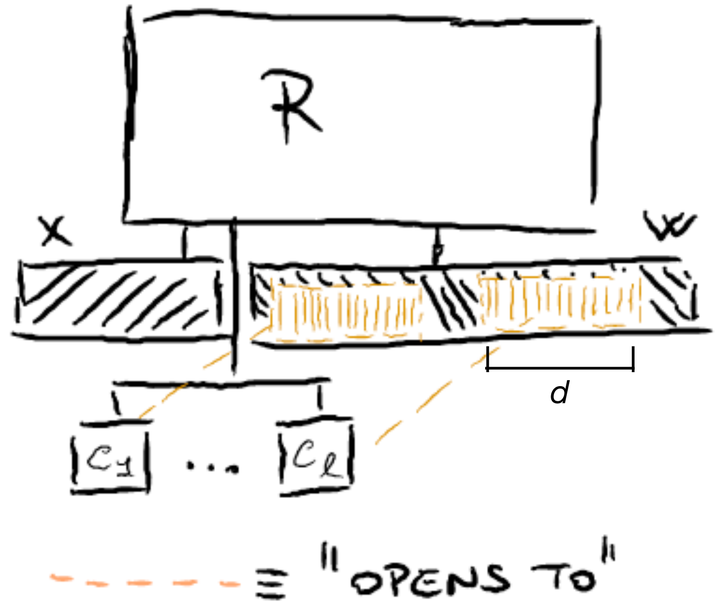
The “Linking” component in Lunar

- **Challenge:** the opening of *input* commitments are “shifted” in some way inside w (encode through the transcript commitments)
- **Solution:** Prove each c_1…c_ell can be expressed through appropriate decomposition



The “Linking” component in Lunar

- **Challenge:** the opening of *input* commitments are “shifted” in some way inside w (encode through the transcript commitments)
- **Solution:** Prove each c_1…c_ell can be expressed through appropriate decomposition
- The result is a pairing-based “linking” proof of roughly O(ell) size.

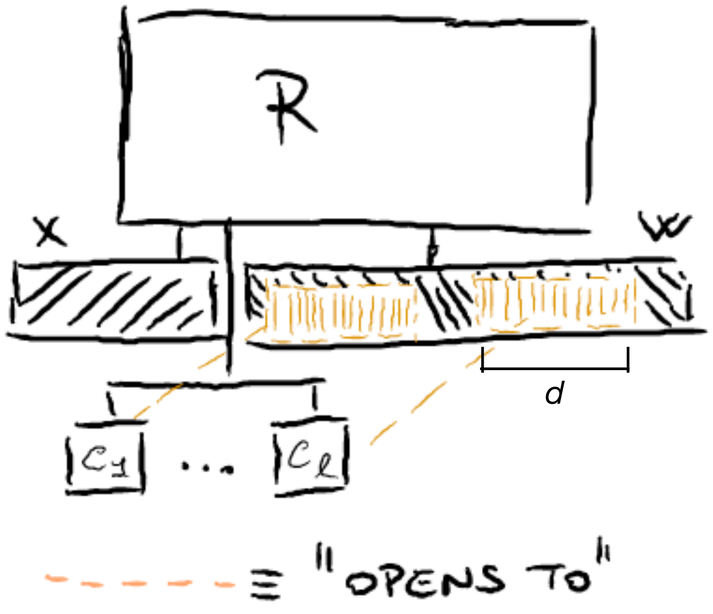


The “Linking” component in ECLIPSE

- Several similarities with the approach in Lunar
- **Differences:**
  - Proves through an (amortized) Sigma-protocol a “squashing” of the input commitments

$$C = \boldsymbol{g}^{\mathbf{w}}\mathbf{h}^{\boldsymbol{\alpha}}, \hat{C}_i = \mathbf{G}^{\mathbf{w}_i}\mathbf{H}^{\boldsymbol{\beta}_i},\ \mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_\ell]$$

    - This requires O(ell d) communication, but it’s then compressed through Compressed-Sigma tricks [AC20] to O(log(ell d))
  - The resulting “squashed” C above is then used to show consistency in a similar fashion as in Lunar



Additional Results in Lunar

Additional Results in Lunar

- **A framework for polynomially holographic proofs (PHPs)**

Additional Results in Lunar

- **A framework for polynomially holographic proofs (PHPs)**
  - Includes previous AHP/ILDPs/PIOPs as special cases

Additional Results in Lunar

- **A framework for polynomially holographic proofs (PHPs)**
  - Includes previous AHP/ILDPs/PIOPs as special cases
- **A new constraint system R1CSLite (“simpler” version of R1CS)**

Additional Results in Lunar

- **A framework for polynomially holographic proofs (PHPs)**
  - Includes previous AHP/ILDPs/PIOPs as special cases
- **A new constraint system R1CSLite (“simpler” version of R1CS)**

$$\begin{cases} \boldsymbol{a} + \boldsymbol{L} \cdot \boldsymbol{c} = 0 \\ \boldsymbol{b} + \boldsymbol{R} \cdot \boldsymbol{c} = 0 \\ \boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c} \end{cases}$$

R1CSLite

Additional Results in Lunar

- **A framework for polynomially holographic proofs (PHPs)**
  - Includes previous AHP/ILDPs/PIOPs as special cases
- **A new constraint system R1CSLite (“simpler” version of R1CS)**
  - Allows to obtain more efficient SNARKs

$$\begin{cases} \boldsymbol{a} + \boldsymbol{L} \cdot \boldsymbol{c} = 0 \\ \boldsymbol{b} + \boldsymbol{R} \cdot \boldsymbol{c} = 0 \\ \boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c} \end{cases}$$

R1CSLite

Additional Results in Lunar

- **A framework for polynomially holographic proofs (PHPs)**
  - Includes previous AHP/ILDPs/PIOPs as special cases
- **A new constraint system R1CSLite (“simpler” version of R1CS)**
  - Allows to obtain more efficient SNARKs
- **Concrete constructions of SNARKs**

$$\begin{cases} \boldsymbol{a} + \boldsymbol{L} \cdot \boldsymbol{c} = 0 \\ \boldsymbol{b} + \boldsymbol{R} \cdot \boldsymbol{c} = 0 \\ \boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c} \end{cases}$$

R1CSLite

Additional Results in Lunar

- **A framework for polynomially holographic proofs (PHPs)**
  - Includes previous AHP/ILDPs/PIOPs as special cases
- **A new constraint system R1CSLite (“simpler” version of R1CS)**
  - Allows to obtain more efficient SNARKs
- **Concrete constructions of SNARKs**
  - Improving on the state of the art in several metrics

$$\begin{cases} \boldsymbol{a} + \boldsymbol{L} \cdot \boldsymbol{c} = 0 \\ \boldsymbol{b} + \boldsymbol{R} \cdot \boldsymbol{c} = 0 \\ \boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c} \end{cases}$$

R1CSLite

Scary tables from Lunar (efficiency of constructions)

zkSNARK		size				time			
		$\|\mathsf{srs}\|$	$\|\mathsf{ek}_\mathsf{R}\|$	$\|\mathsf{vk}_\mathsf{R}\|$	$\|\pi\|$	KeyGen	Derive	Prove	Verify
Sonic [46]	$\mathbb{G}_1$	$4N$	$36n$	—	20	$4N$	$36n$	$273n$	7 pairings
	$\mathbb{G}_2$	$4N$	—	3	—	$4N$	—	—	
	$\mathbb{F}$	—	—	—	16	—	$O(m \log m)$	$O(m \log m)$	$O(\ell+\log m)$
Marlin [20]	$\mathbb{G}_1$	$3M$	$3m$	12	13	$3M$	$12m$	$14n+8m$	2 pairings
	$\mathbb{G}_2$	2	—	2	—	—	—	—	
	$\mathbb{F}$	—	—	—	8	—	$O(m \log m)$	$O(m \log m)$	$O(\ell+\log m)$
PLONK (small proof) [28]	$\mathbb{G}_1$	$3N^*$	$3n+3a$	8	7	$3N^*$	$8n+8a$	$11n+11a$	2 pairings
	$\mathbb{G}_2$	1	—	1	—	1	—	—	
	$\mathbb{F}$	—	—	—	7	—	$O((n+a)\log(n+a))$	$O((n+a)\log(n+a))$	$O(\ell+\log(n+a))$
PLONK (fast prover) [28]	$\mathbb{G}_1$	N^*	$n+a$	8	9	N^*	$8n+8a$	$9n+9a$	2 pairings
	$\mathbb{G}_2$	1	—	1	—	1	—	—	
	$\mathbb{F}$	—	—	—	7	—	$O((n+a)\log(n+a))$	$O((n+a)\log(n+a))$	$O(\ell+\log(n+a))$
LunarLite (this work)	$\mathbb{G}_1$	M	m	—	10	M	—	$8n+3m$	7 pairings
	$\mathbb{G}_2$	M	—	27	—	M	$24m$	—	
	$\mathbb{F}$	—	—	—	2	—	$O(m \log m)$	$O(m \log m)$	$O(\ell+\log m)$
Lunar1cs (fast & short)	$\mathbb{G}_1$	M	m	—	11	M	—	$9n+3m$	7 pairings
	$\mathbb{G}_2$	M	—	60	—	M	$57m$	—	
	$\mathbb{F}$	—	—	—	2	—	$O(m \log m)$	$O(m \log m)$	$O(\ell+\log m)$
Lunar1cs (short vk)	$\mathbb{G}_1$	$3M$	$3m$	12	12	$3M$	$12m$	$9n+8m$	2 pairings
	$\mathbb{G}_2$	1	—	1	—	1	—	—	
	$\mathbb{F}$	—	—	—	5	—	$O(m \log m)$	$O(m \log m)$	$O(\ell+\log m)$

Some of the resulting SNARK construction and comparison

PHP		degree	oracles $\mathcal{RE}$	oracles $\mathcal{P}$	msgs	proof length	$\mathcal{V}$ checks: deg	$\mathcal{V}$ checks: $\deg_{X,\{X_i\}}(G_1)$	$\mathcal{V}$ checks: $\deg_{X,\{X_i\}}(G_2)$
$\mathsf{PHP}_{\mathsf{lite1}}$	4.3	$2m$	8	7	1	$\|\pi\|+2m$	2	2	2
$\mathsf{PHP}_{\mathsf{lite1x}}$	Rk.2	$2m$	5	7	1	$\|\pi\|+2m$	2	2	3
$\mathsf{PHP}_{\mathsf{lite2}}$	4.3	m	24	7	1	$\|\pi\|$	2	2	2
$\mathsf{PHP}_{\mathsf{lite2x}}$	Rk.3	m	16	7	1	$\|\pi\|$	2	2	3
$\mathsf{PHP}_{\mathsf{r1cs1}}$	4.4	$3m$	9	8	1	$\|\pi'\|+4m$	2	2	2
$\mathsf{PHP}_{\mathsf{r1cs1x}}$	Rk.5	$3m$	6	8	1	$\|\pi'\|+4m$	2	2	3
$\mathsf{PHP}_{\mathsf{r1cs2}}$	4.4	m	57	8	1	$\|\pi'\|$	2	2	2
$\mathsf{PHP}_{\mathsf{r1cs2x}}$	Rk.6	m	42	8	1	$\|\pi'\|$	2	2	3
$\mathsf{PHP}_{\mathsf{r1cs3}}$	4.4	$3m$	12	8	1	$\|\pi'\|$	2	2	5

All PHP Constructions in Lunar

Open Questions

- **Better asymptotics:**
  - O(\ell) is inherent in verification time, but can we achieve constant proof size?
  - Maybe with one-level of (specialised) recursion?
- **Different techniques for linking and/or finding other applications for them.**

	$\|\pi\|$	Prove (time)	Verify (time)
ECLIPSE [ABC+21]	$O(\log(\ell \cdot d))$	$O(n + \ell \cdot d)$	$O(\ell \cdot d)$
Lunar [CFF+20]	$O(\ell)$	$O(n + \ell \cdot d)$	$O(\ell)$
Future?	$O(1)$		$O(\ell)$

https://ia.cr/2020/1069

On eprint soon!

Thanks!