When Can We Incrementally Prove Computations of Arbitrary Depth?

Matteo Campanelli
Offchain Labs
University of Tartu, Estonia
matteo@offchainlabs.com
www.binarywhales.com

eprint:2025/1413

A joint work with
Dario Fiore and Mahak Pancholi
(IMDEA Software Institute)

This Talk in a Nutshell

This Talk in a Nutshell

A study of the security of
Incrementally Verifiable Computation (IVC) under
the lens of the <u>depth</u> of the proven computation.

This Talk in a Nutshell

A study of the security of
Incrementally Verifiable Computation (IVC) under
the lens of the depth of the proven computation.

Our main motivation:
how can we prove security (or insecurity)
when we move beyond constant depth?

Succinct Cryptographic Proofs (SNARKs)

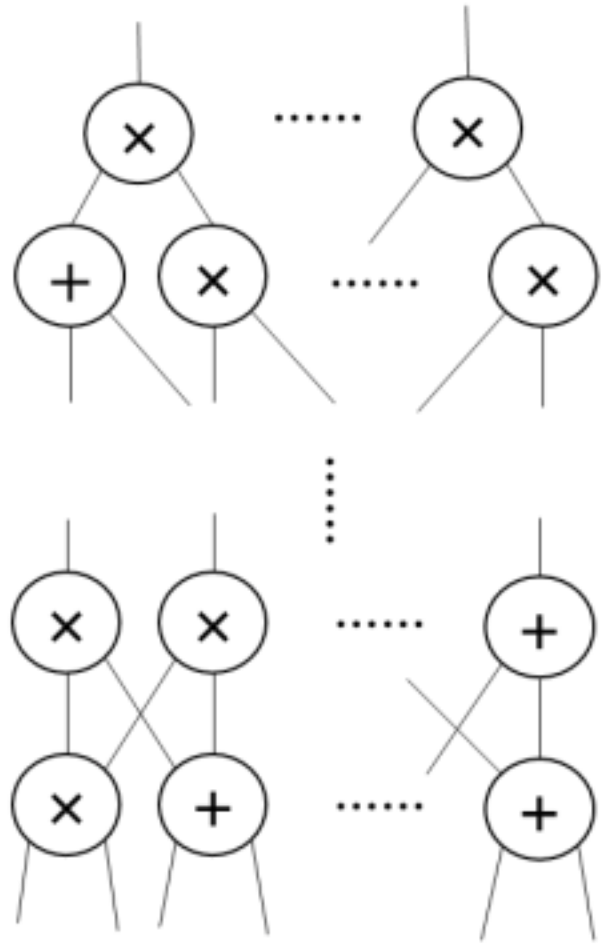
Server (Prover)

Client (Verifier)

Succinct Cryptographic Proofs (SNARKs)

Server (Prover)

Client (Verifier)

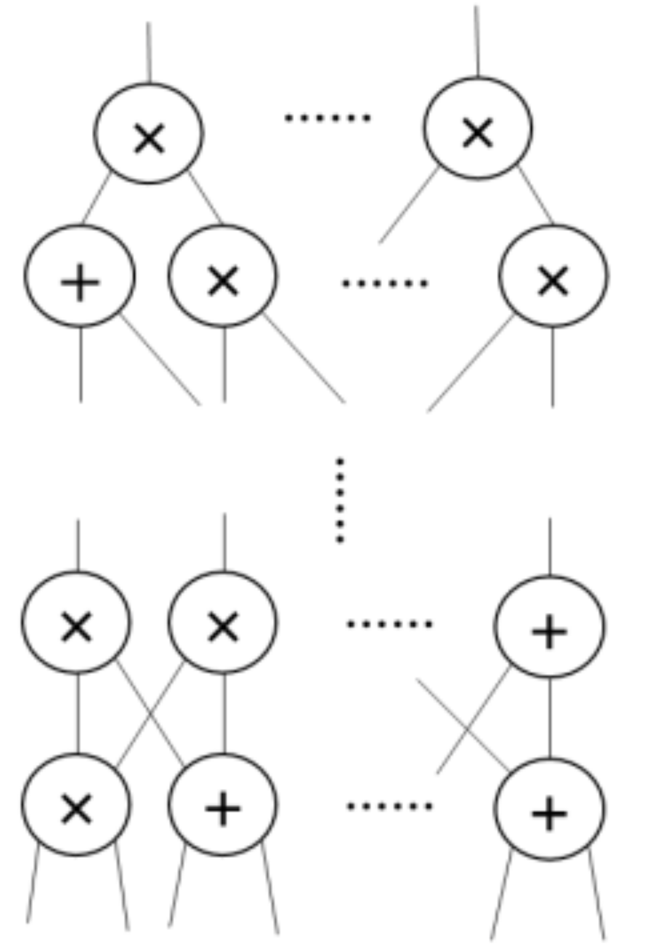


Some program *F*

Succinct Cryptographic Proofs (SNARKs)

Server (Prover)

Client (Verifier)



Some program *F*

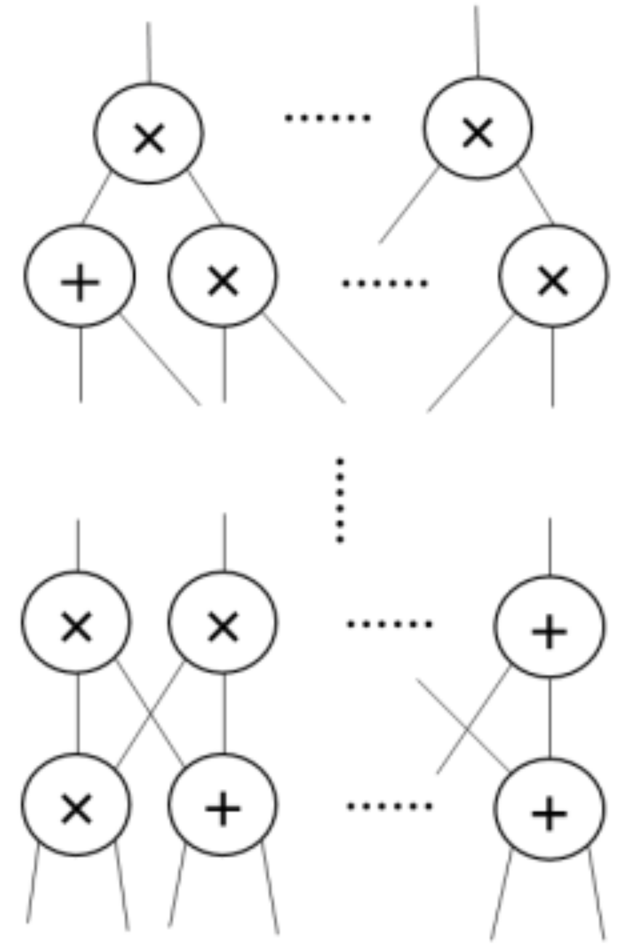
Client would like to know whether $\exists w : F(\mathrm{x}, \mathrm{w}) = 1$

Succinct Cryptographic Proofs (SNARKs)

Server (Prover)

π

Client (Verifier)



Some program *F*

Client would like to know
whether $\exists w : F(\mathrm{x}, \mathrm{w}) = 1$

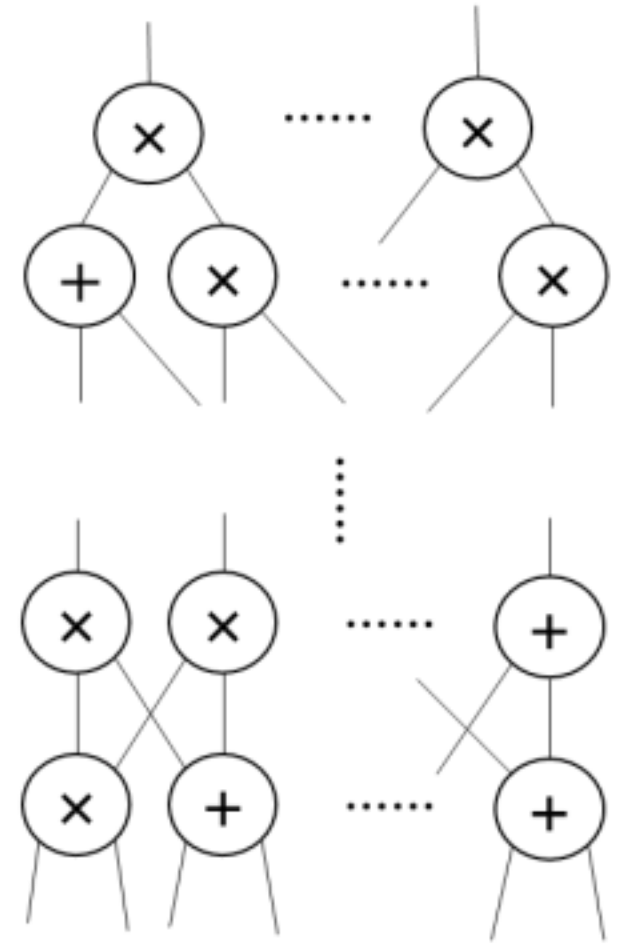
Succinct Cryptographic Proofs (SNARKs)

Server (Prover)

π

Proof that statement is true

Client (Verifier)



Some program *F*

Client would like to know whether $\exists w : F(\mathrm{x}, \mathrm{w}) = 1$

Succinct Cryptographic Proofs (SNARKs)

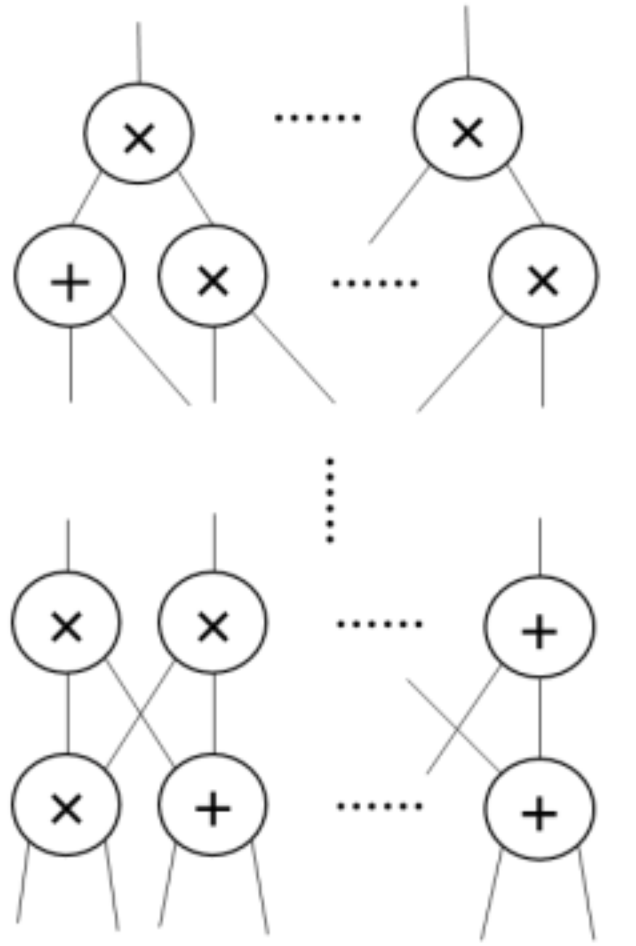
Server (Prover)

π

Proof that statement is true

Verify(x, π)

Client (Verifier)



Some program *F*

Client would like to know whether $\exists w : F(\mathrm{x}, \mathrm{w}) = 1$

Succinct Cryptographic Proofs (SNARKs)

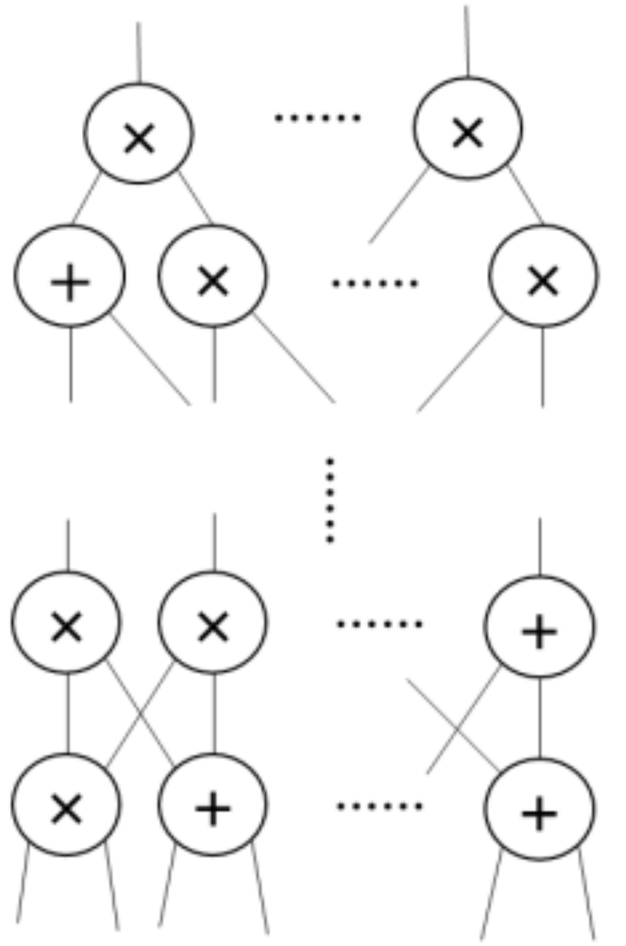
Server (Prover)

π

Proof that statement is true

Verify(x, π)

Client (Verifier)



Some program *F*

Common requirement: **Succinctness**

(π is very small; Verify is very fast)

Client would like to know whether $\exists w : F(\mathrm{x}, \mathrm{w}) = 1$

Limitations of Traditional “Monolithic” Proofs

Limitations of Traditional “Monolithic” Proofs

- Large memory requirements

Limitations of Traditional “Monolithic” Proofs

- Large memory requirements
  - whole “trace” of the computation should in principle be kept all in memory at the same time

Limitations of Traditional “Monolithic” Proofs

- Large memory requirements
    - whole “trace” of the computation should in principle be kept all in memory at the same time
- No pipelining

Limitations of Traditional “Monolithic” Proofs

- Large memory requirements
  - whole “trace” of the computation should in principle be kept all in memory at the same time
- No pipelining
  - Must finish the computation before starting proving

Limitations of Traditional “Monolithic” Proofs

- Large memory requirements
  - whole “trace” of the computation should in principle be kept all in memory at the same time
- No pipelining
  - Must finish the computation before starting proving
  - Cannot take advantage of incremental computations (next slide)

Incremental Computations

$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \cdots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

$$z_i = F(z_{i-1}, w_{i-1})$$

Incremental Computations

$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \cdots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

$$z_i = F(z_{i-1}, w_{i-1})$$

Examples of natural applications:

Incremental Computations

$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \cdots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

$$z_i = F(z_{i-1}, w_{i-1})$$

Examples of natural applications:

- Streaming algorithms

Incremental Computations

$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \cdots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

$$z_i = F(z_{i-1}, w_{i-1})$$

Examples of natural applications:

- Streaming algorithms
- RAM computations

Incremental Computations

$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \cdots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

$$z_i = F(z_{i-1}, w_{i-1})$$

Examples of natural applications:

- Streaming algorithms
- RAM computations
- Verifiable Delay Functions (VDF)

Incremental Computations

$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \cdots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

$$z_i = F(z_{i-1}, w_{i-1})$$

Examples of natural applications:

- Streaming algorithms
- RAM computations
- Verifiable Delay Functions (VDF)
- Round functions in symmetric primitives

Incremental Computations

$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \cdots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

$$z_i = F(z_{i-1}, w_{i-1})$$

Examples of natural applications:

- Streaming algorithms
- RAM computations
- Verifiable Delay Functions (VDF)
- Round functions in symmetric primitives
- Recurrent neural networks

Incremental Computations

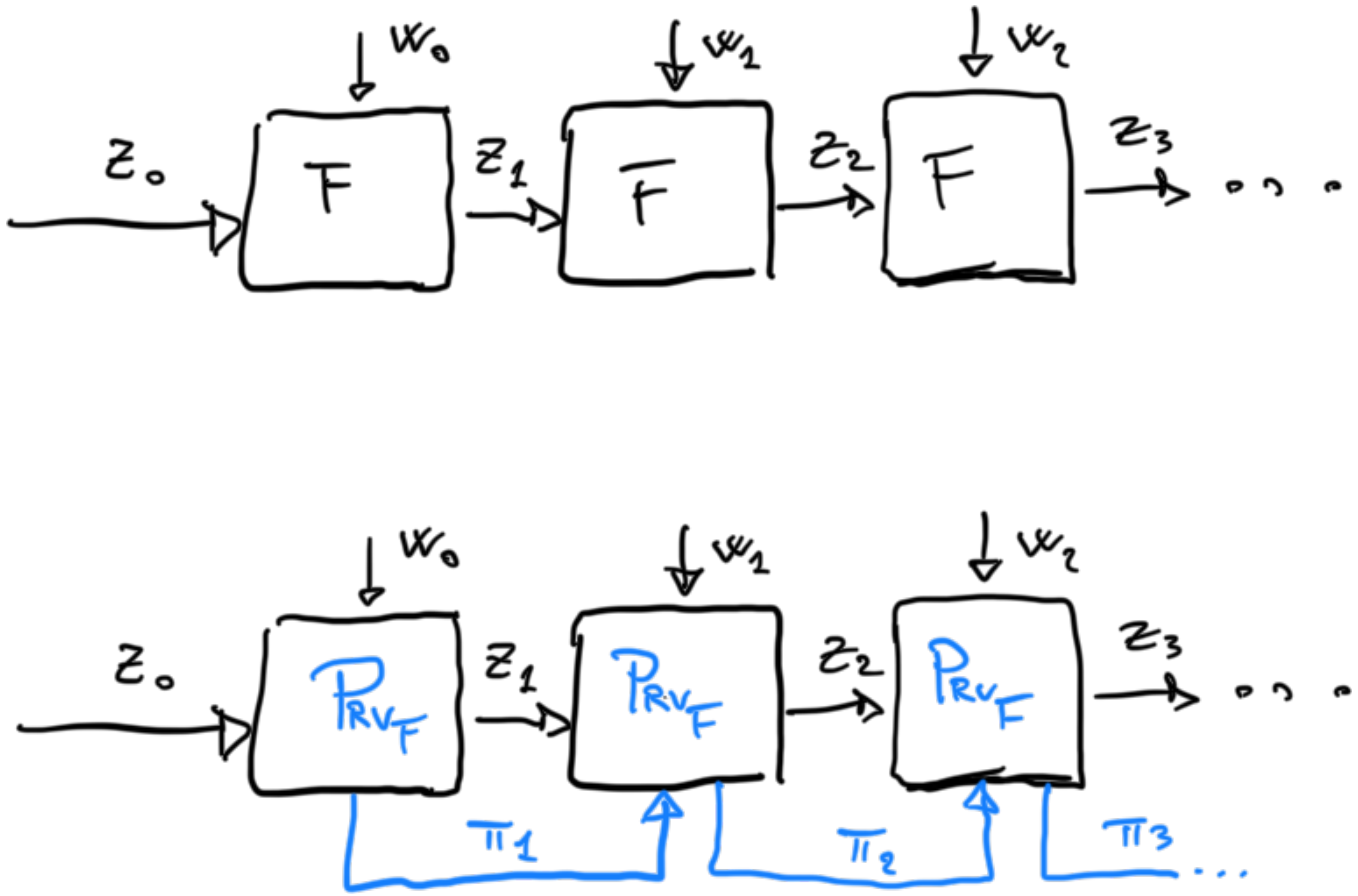
$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \cdots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

$$z_i = F(z_{i-1}, w_{i-1})$$

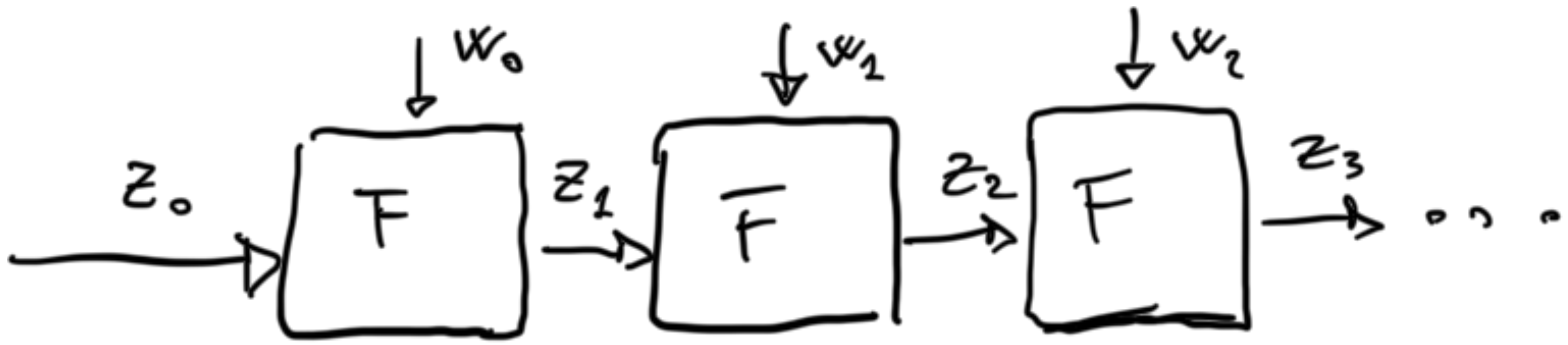
Examples of natural applications:

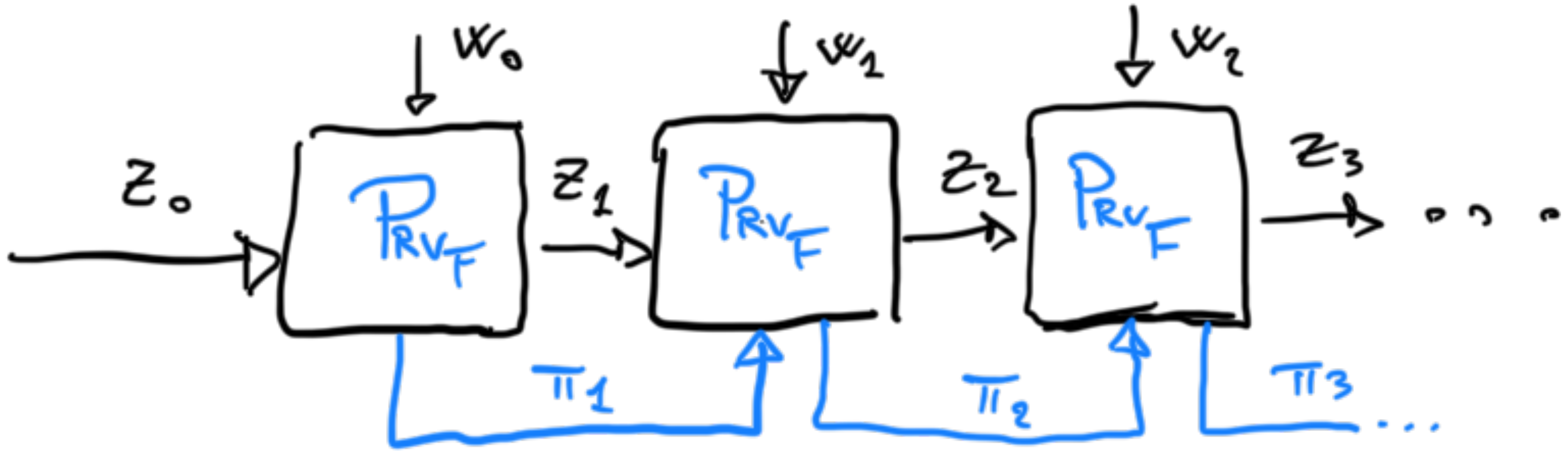
- Streaming algorithms
- RAM computations
- Verifiable Delay Functions (VDF)
- Round functions in symmetric primitives
- Recurrent neural networks
- …

Incrementally Verifiable Computations (IVC)



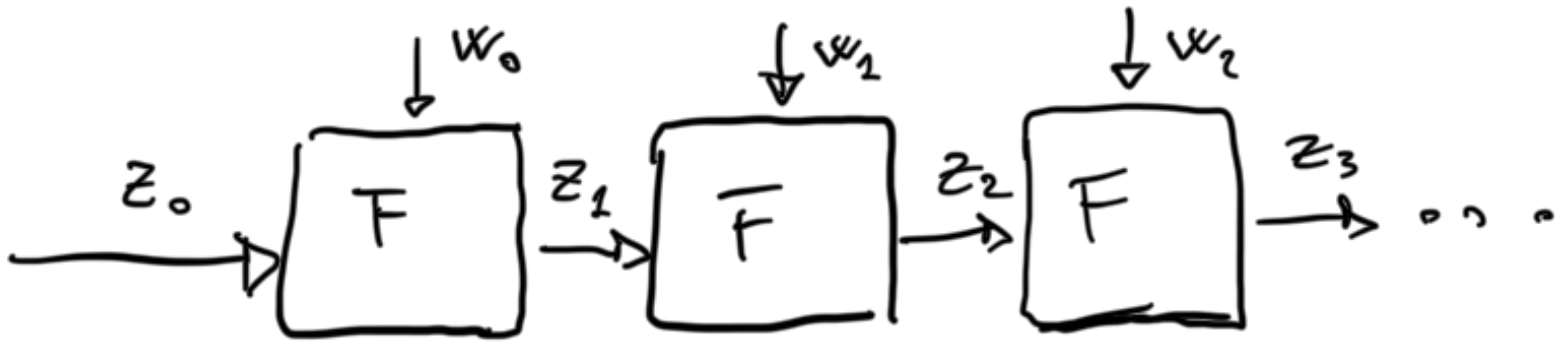
Incrementally Verifiable Computations (IVC)

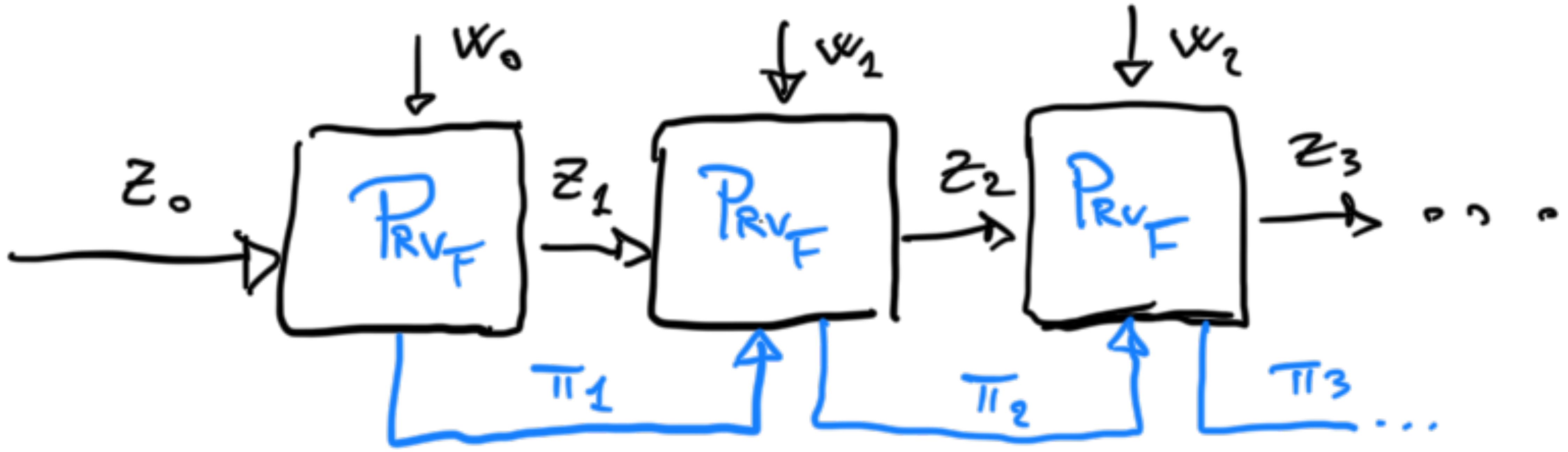




Proof size should be sublinear in the # of steps (the *depth* of the computation)

Incrementally Verifiable Computations (IVC)

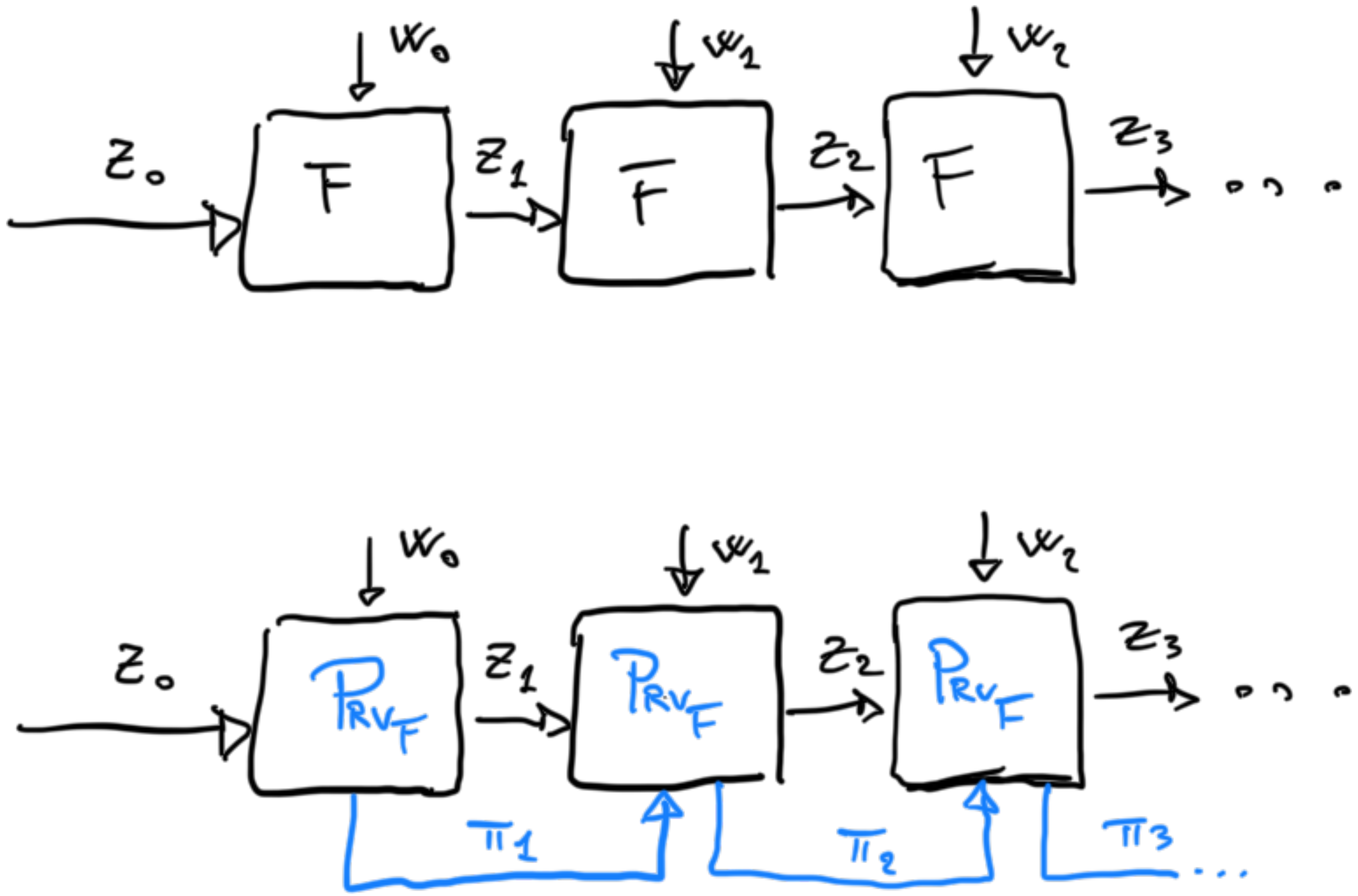




Advantages

Proof size should be sublinear in the # of steps (the *depth* of the computation)

Incrementally Verifiable Computations (IVC)

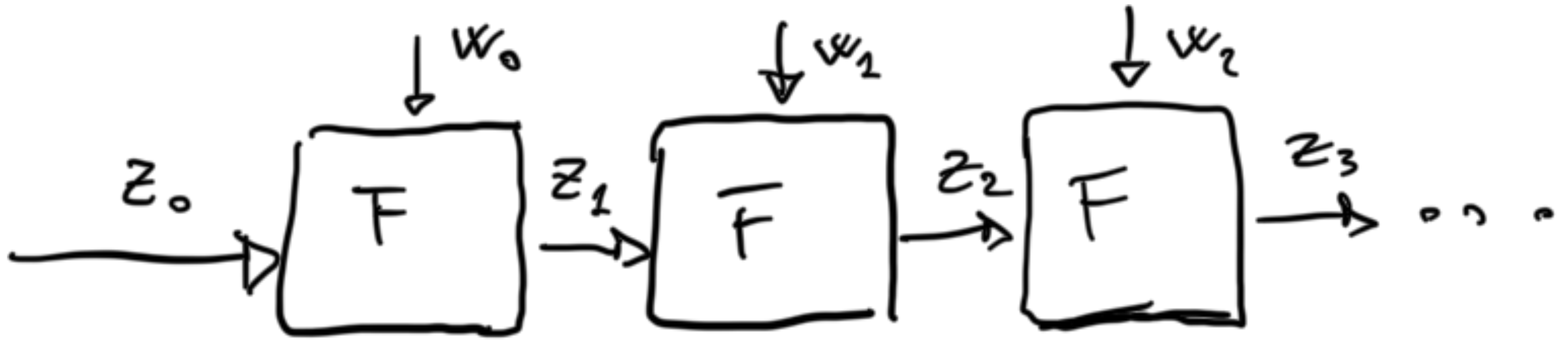


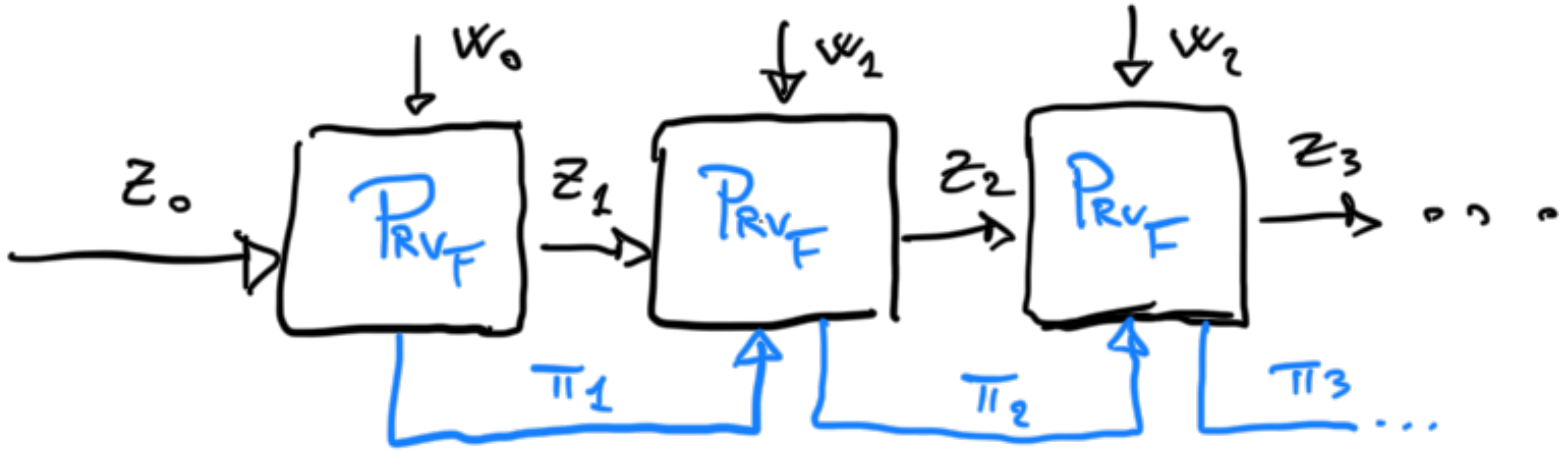
Advantages

- Low memory footprint

Proof size should be sublinear in the # of steps (the *depth* of the computation)

Incrementally Verifiable Computations (IVC)



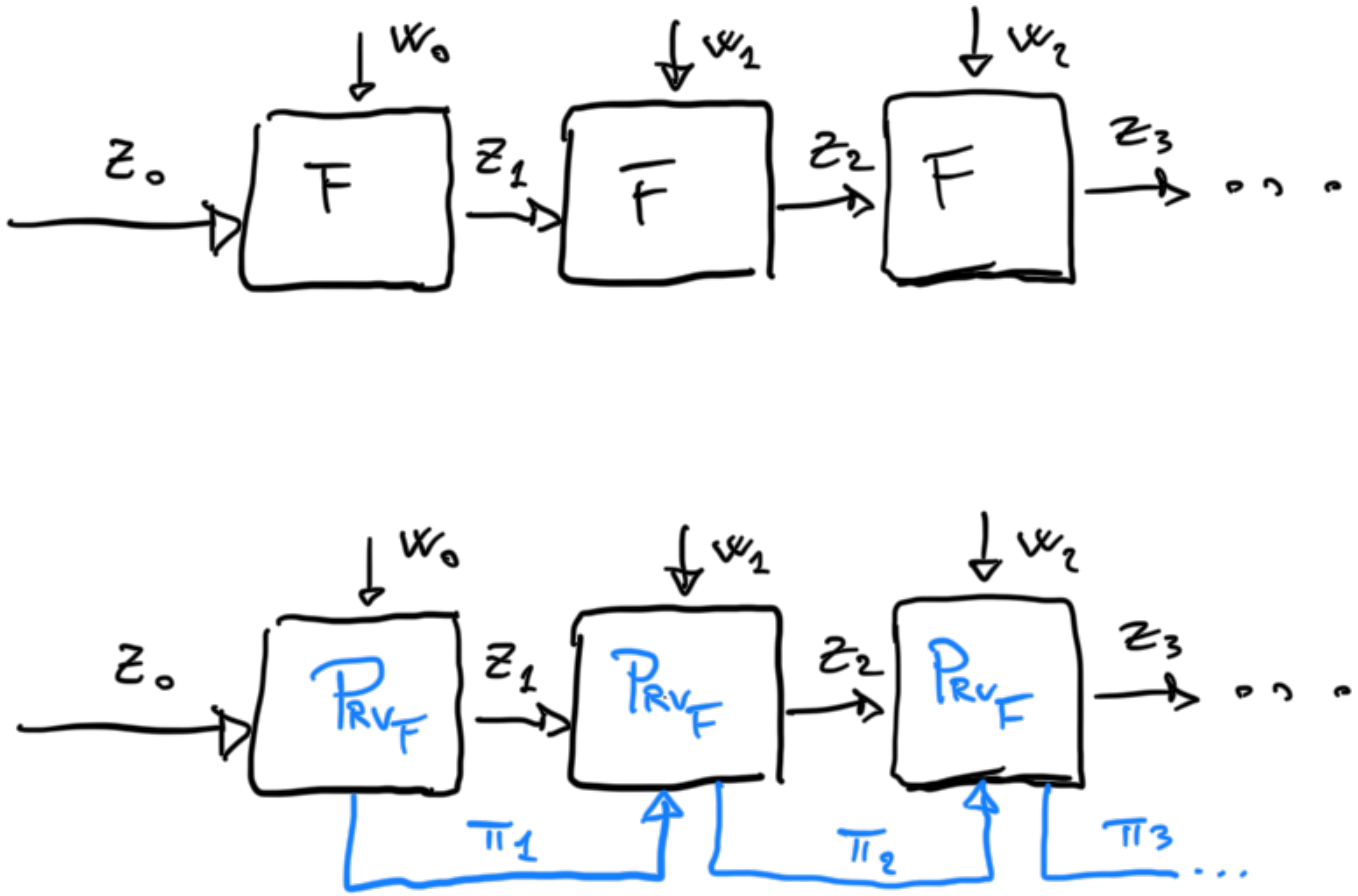


Advantages

- Low memory footprint
- Pipelining opportunities

Proof size should be sublinear in the # of steps (the *depth* of the computation)

Incrementally Verifiable Computations (IVC)

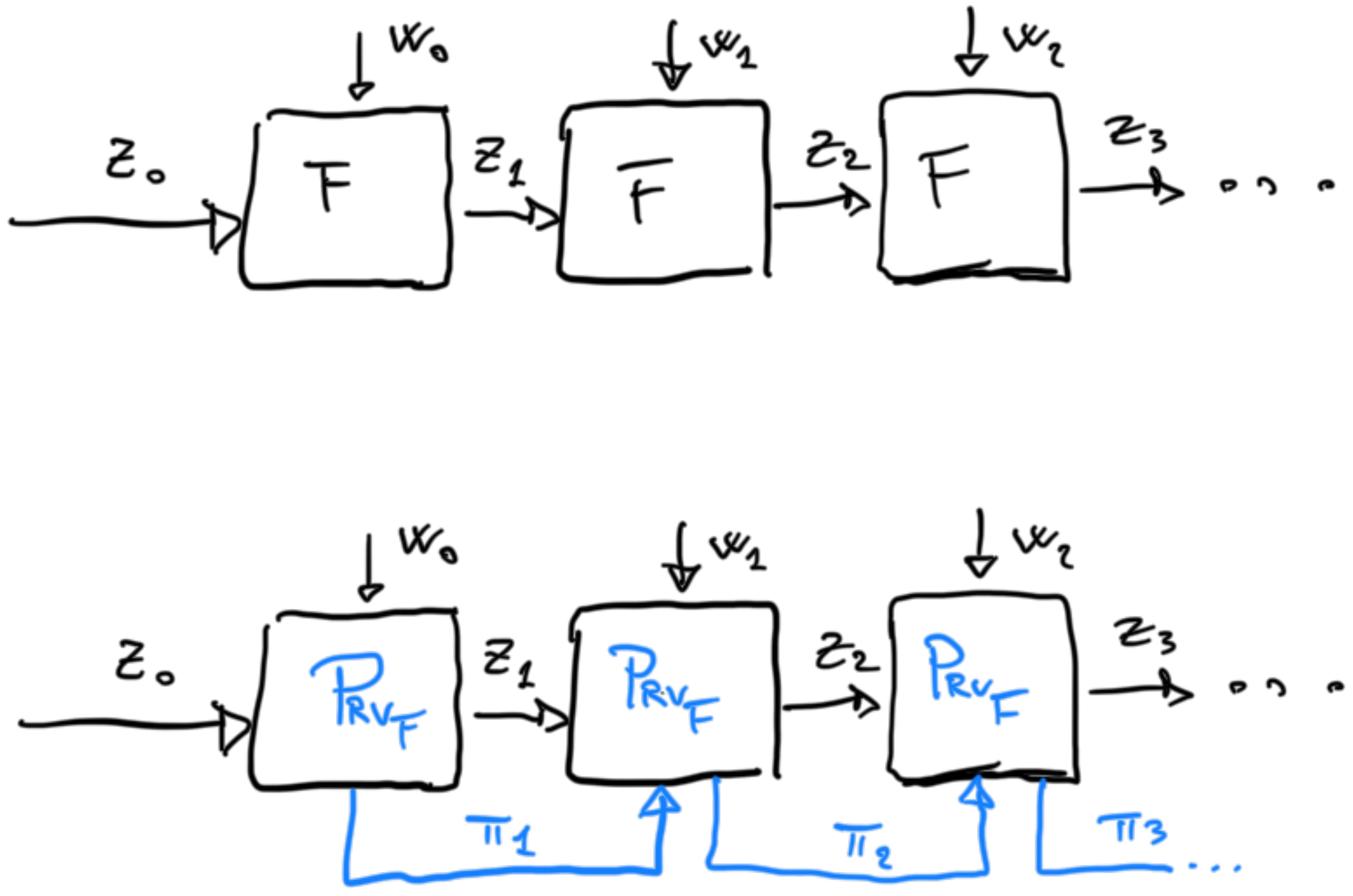


Advantages

- Low memory footprint
- Pipelining opportunities
- Natural model for incremental computations

Proof size should be sublinear in the # of steps (the *depth* of the computation)

Incrementally Verifiable Computations (IVC)



Advantages

- Low memory footprint
- Pipelining opportunities
- Natural model for incremental computations
- Proofs can be distributed (e.g., in settings with zero-knowledge)

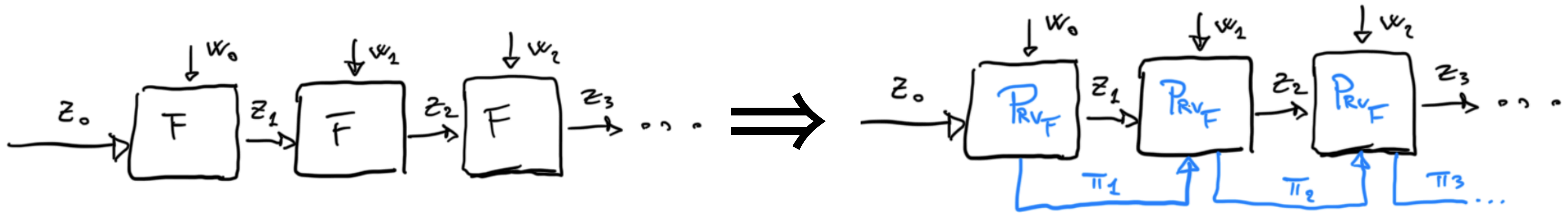
Proof size should be sublinear in the # of steps (the *depth* of the computation)

Constructions of IVC

(Practical or nearly-practical)

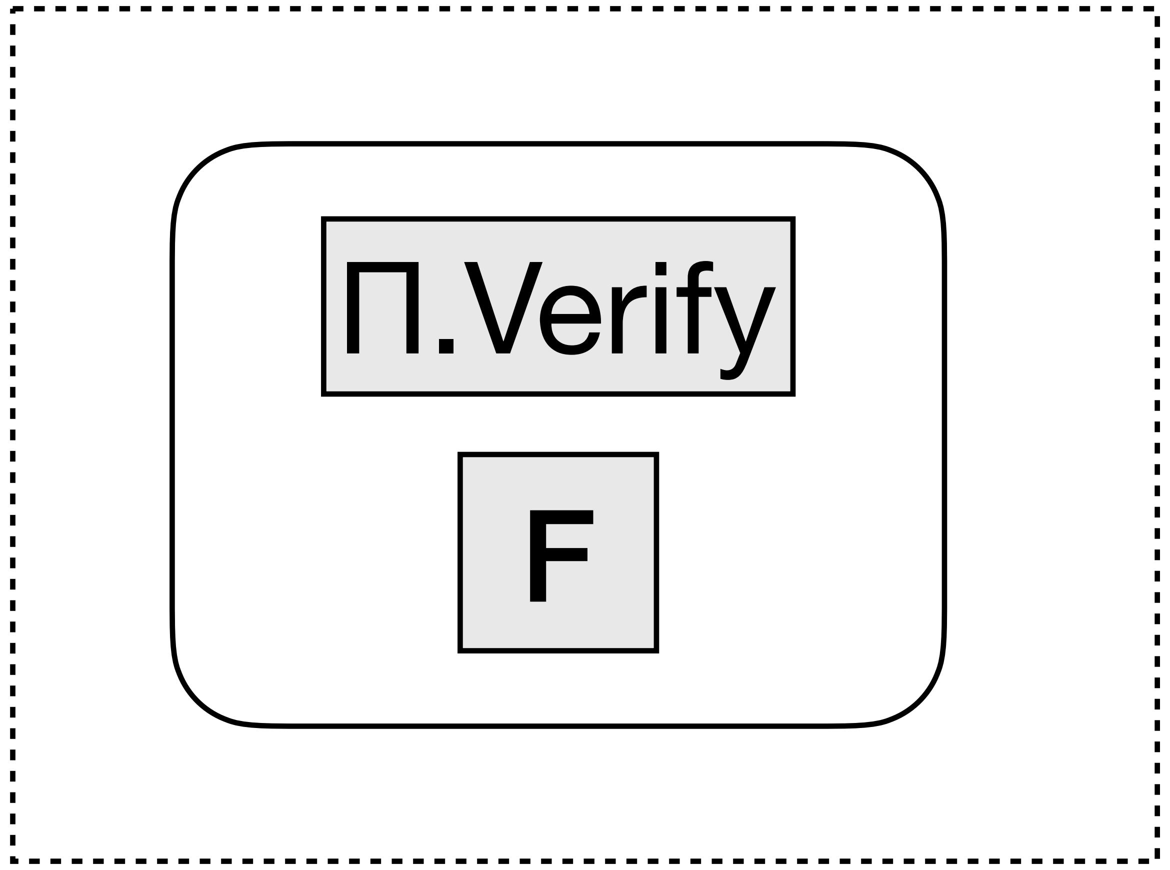
Constructions of IVC

(Practical or nearly-practical)



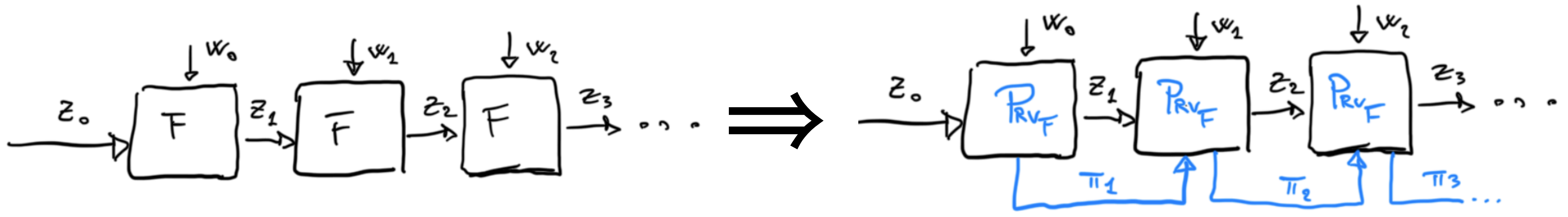
Constructions of IVC

(Practical or nearly-practical)



SNARK Prover’s statement (Π is a SNARK)

Canonical construction
(SNARK recursion)

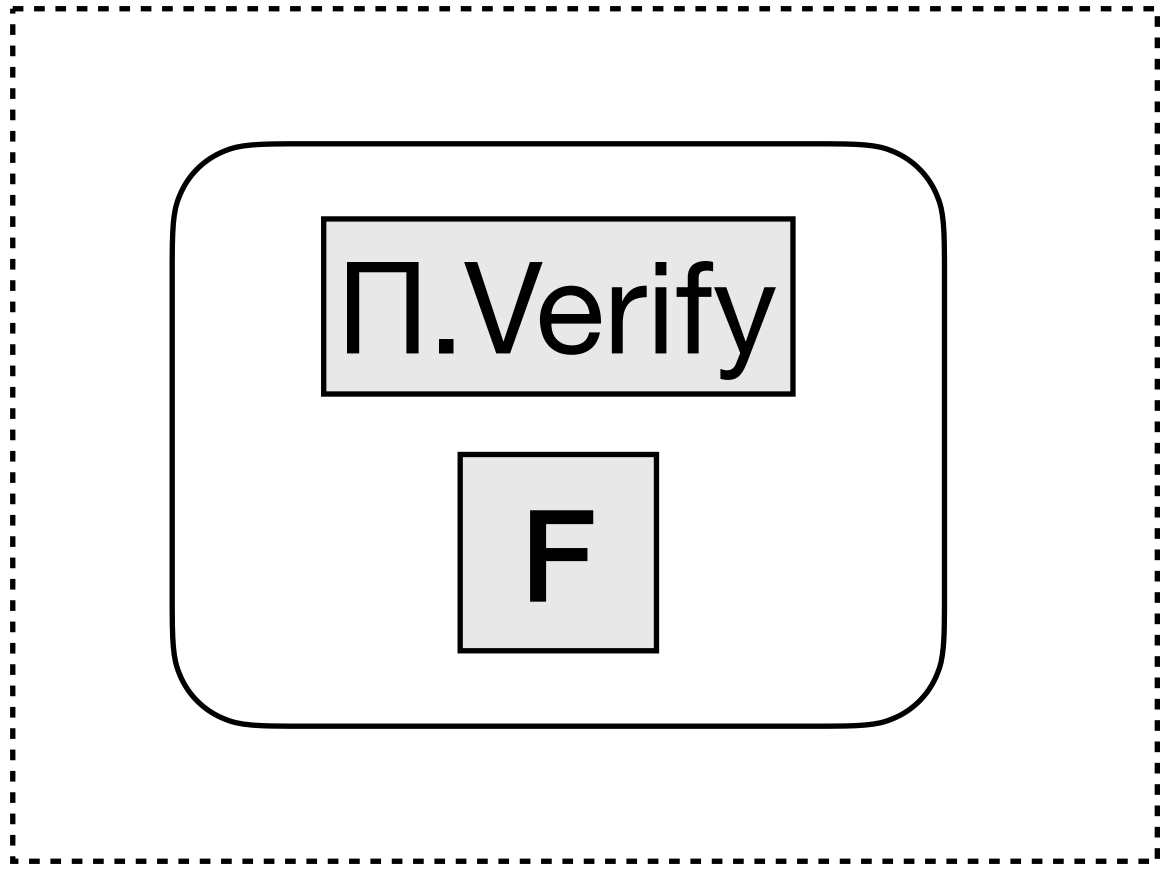


Constructions of IVC

(Practical or nearly-practical)

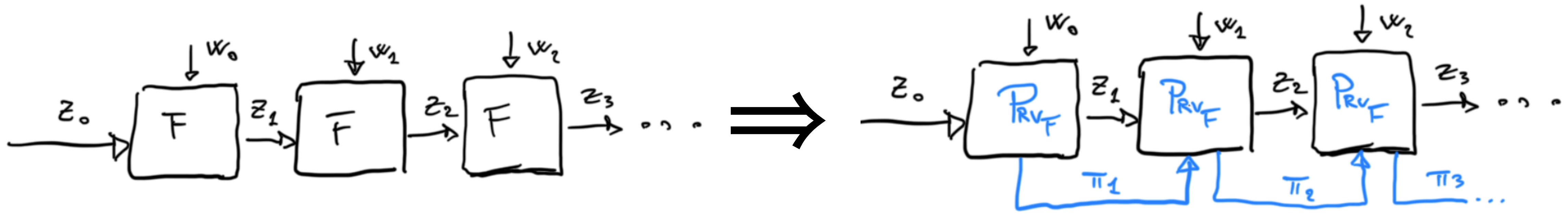
Nova: Recursive Zero-Knowledge Arguments from Folding Schemes

Abhiram Kothapalli[†] Srinath Setty[*] Ioanna Tziala[‡]
[†]Carnegie Mellon University [*]Microsoft Research [‡]New York University



SNARK Prover's statement (Π is a SNARK)

Canonical construction (SNARK recursion)



Constructions of IVC

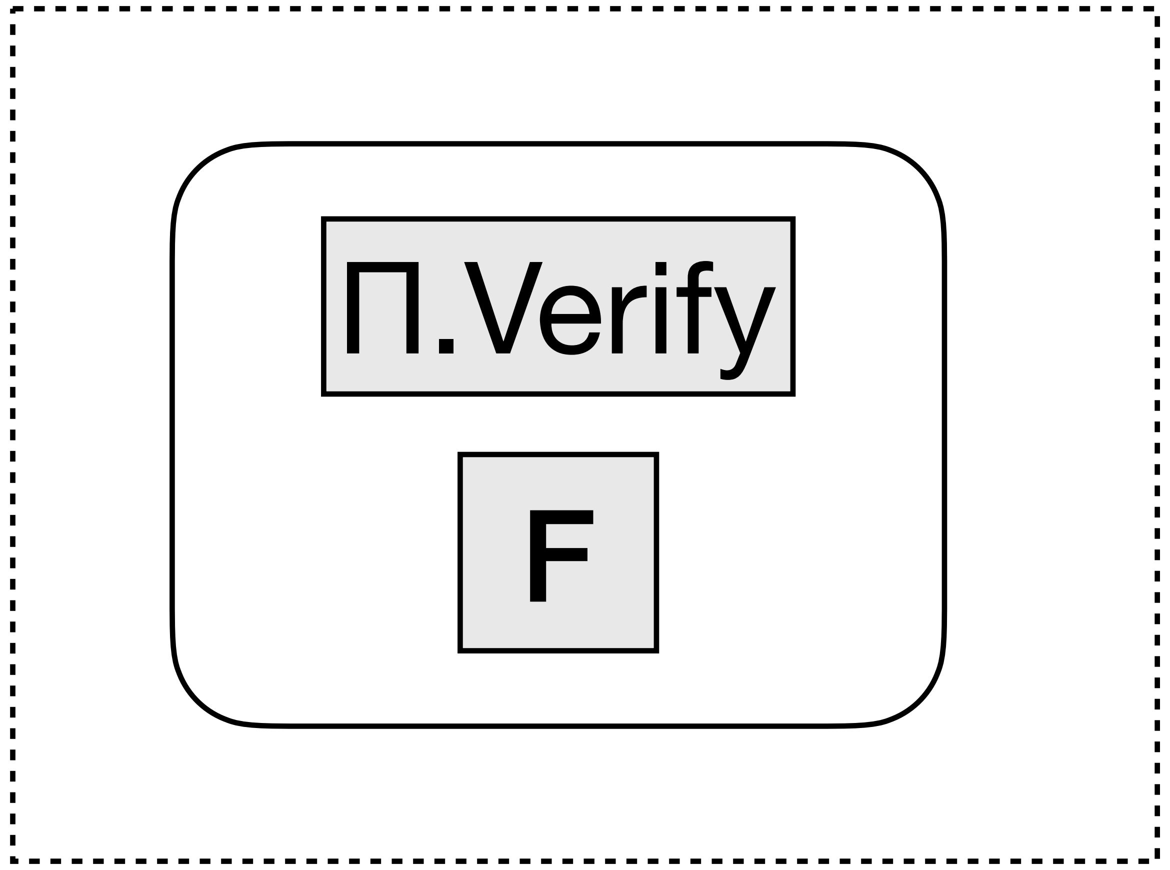
(Practical or nearly-practical)

Nova: Recursive Zero-Knowledge Arguments from Folding Schemes

Abhiram Kothapalli[†] Srinath Setty[*] Ioanna Tzialla[‡]
[†]Carnegie Mellon University [*]Microsoft Research [‡]New York University

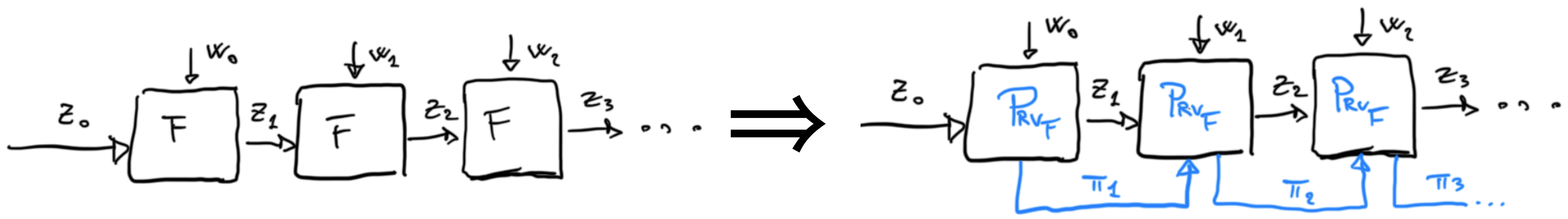
Proof-Carrying Data without Succinct Arguments

Benedikt Bünz, benedikt@cs.stanford.edu, Stanford University
Alessandro Chiesa, alexch@berkeley.edu, UC Berkeley
William Lin, will.lin@berkeley.edu, UC Berkeley
Pratyush Mishra, pratyush@berkeley.edu, UC Berkeley
Nicholas Spooner, nspooner@bu.edu, Boston University



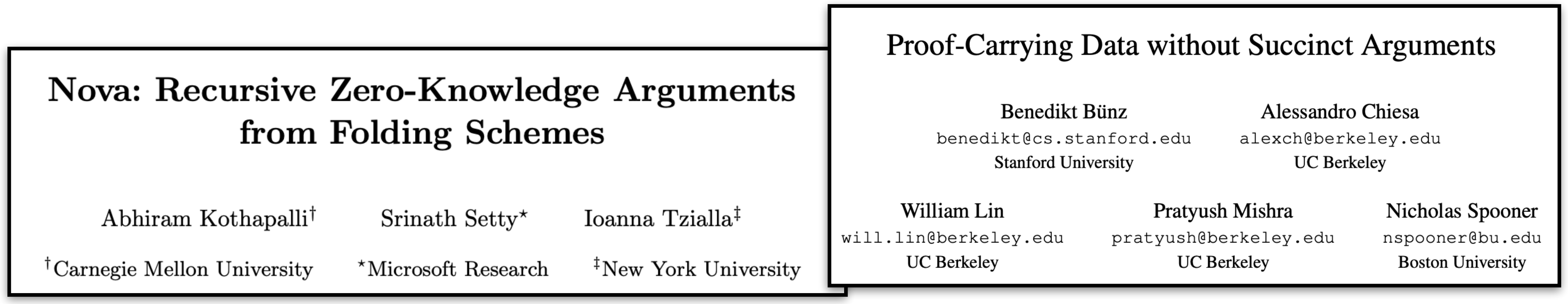
SNARK Prover's statement (Π is a SNARK)

Canonical construction (SNARK recursion)



Constructions of IVC

(Practical or nearly-practical)

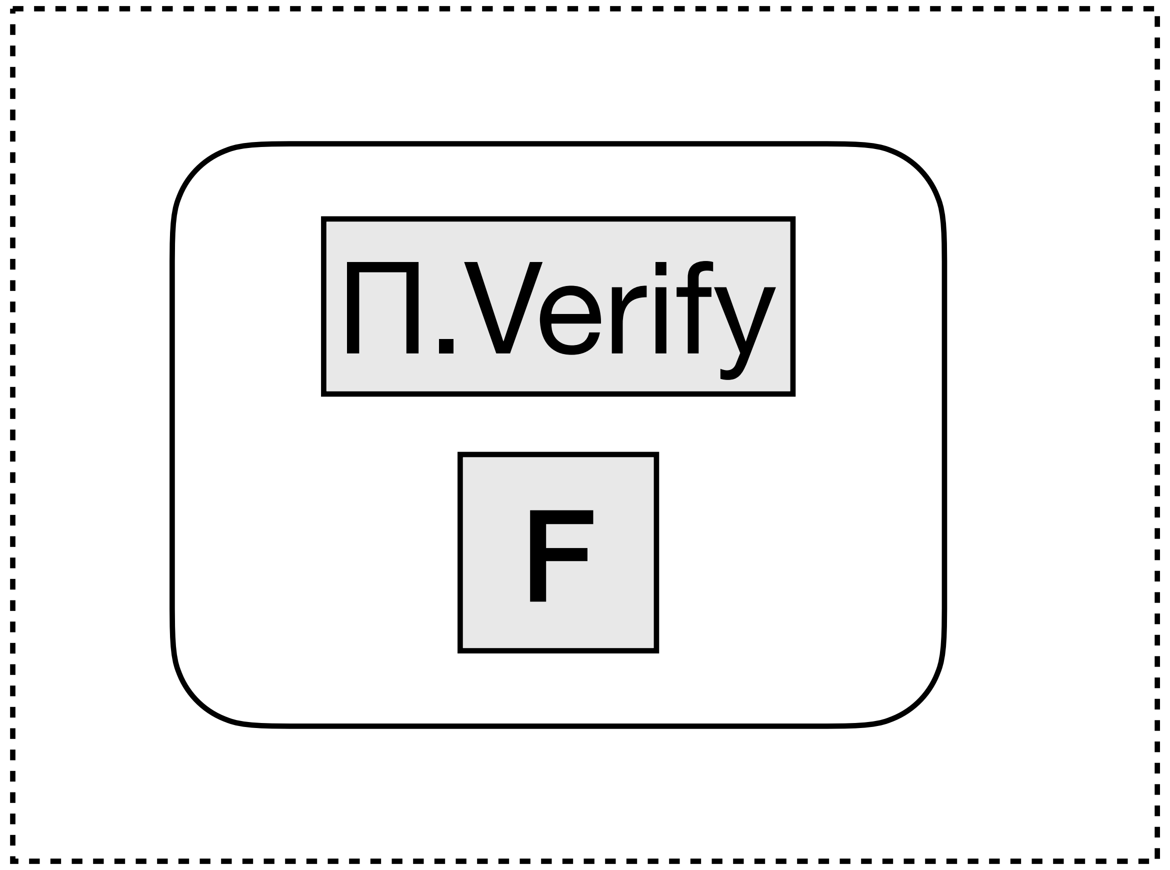


Nova: Recursive Zero-Knowledge Arguments from Folding Schemes

Abhiram Kothapalli[†] Srinath Setty[*] Ioanna Tzialla[‡]
[†]Carnegie Mellon University [*]Microsoft Research [‡]New York University

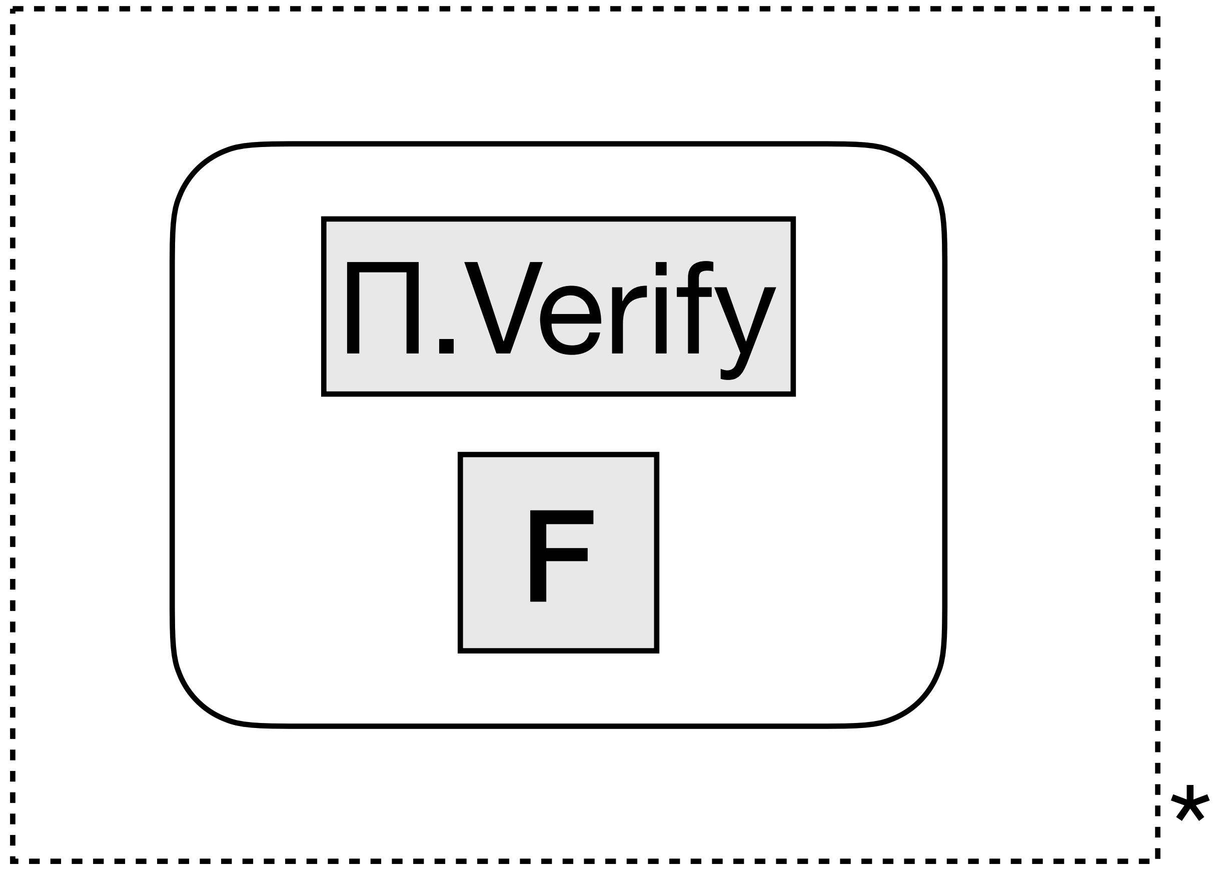
Proof-Carrying Data without Succinct Arguments

Benedikt Bünz benedikt@cs.stanford.edu Stanford University
Alessandro Chiesa alexch@berkeley.edu UC Berkeley
William Lin will.lin@berkeley.edu UC Berkeley
Pratyush Mishra pratyush@berkeley.edu UC Berkeley
Nicholas Spooner nspooner@bu.edu Boston University



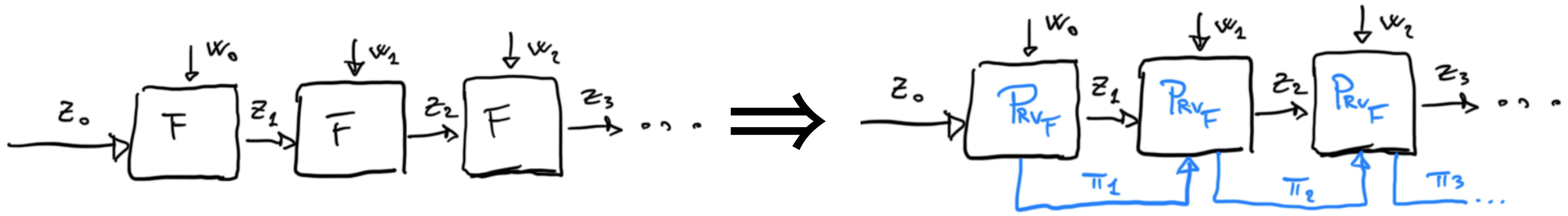
SNARK Prover's statement (Π is a SNARK)

Canonical construction (SNARK recursion)



Folding/acc. Prover's statement (Π is a folding/acc. scheme)

Lightweight version (folding/accumulation recursion)



* very approximate rendition (there are more details)

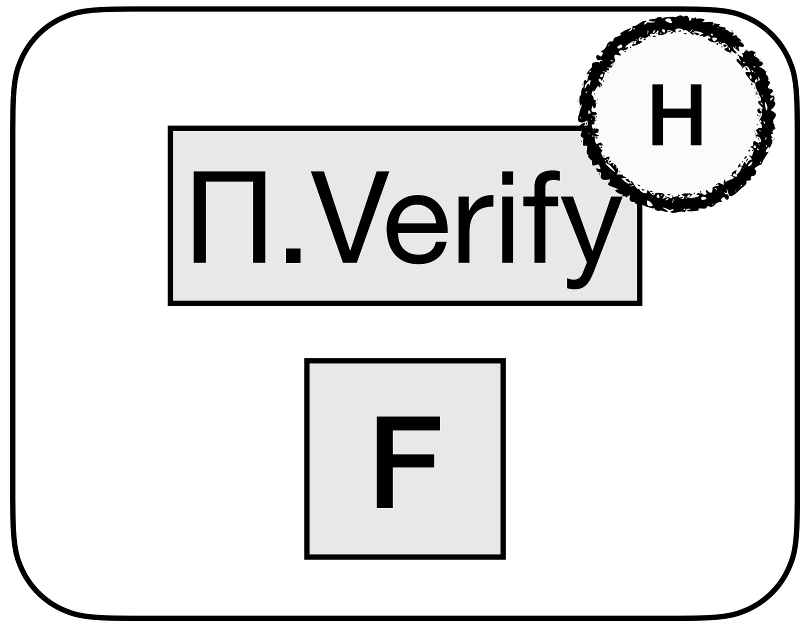
Challenges in Proving the Security of IVC

Challenges in Proving the Security of IVC

- **First challenge:** idealized models and “theoretical hygiene”

Challenges in Proving the Security of IVC

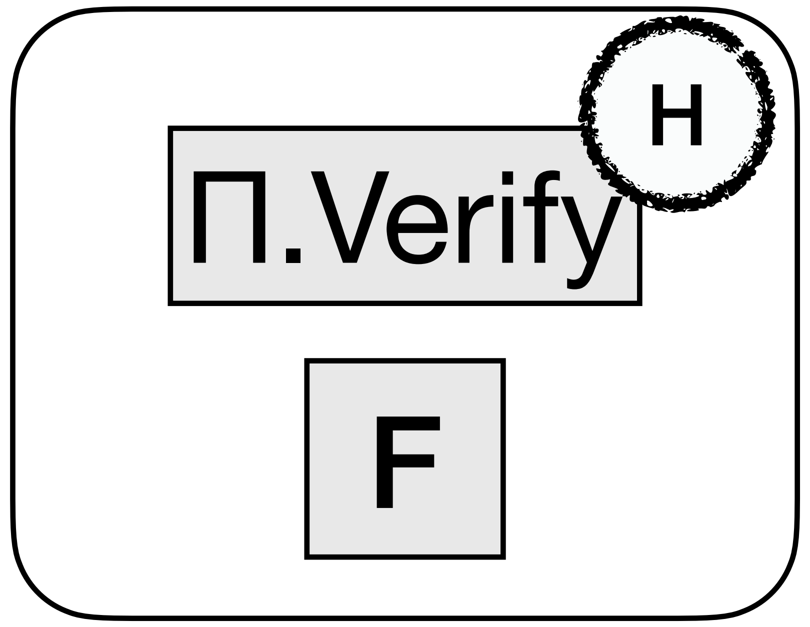
- **First challenge:** idealized models and “theoretical hygiene”



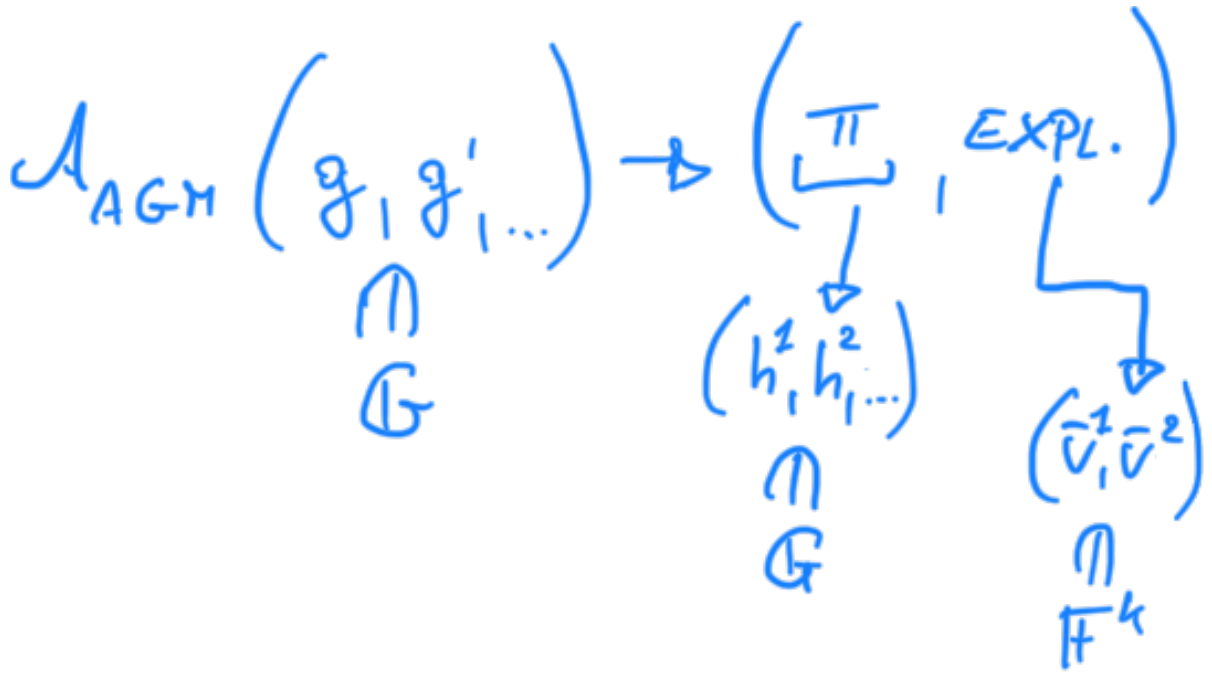
Random Oracle

Challenges in Proving the Security of IVC

- **First challenge:** idealized models and "theoretical hygiene"



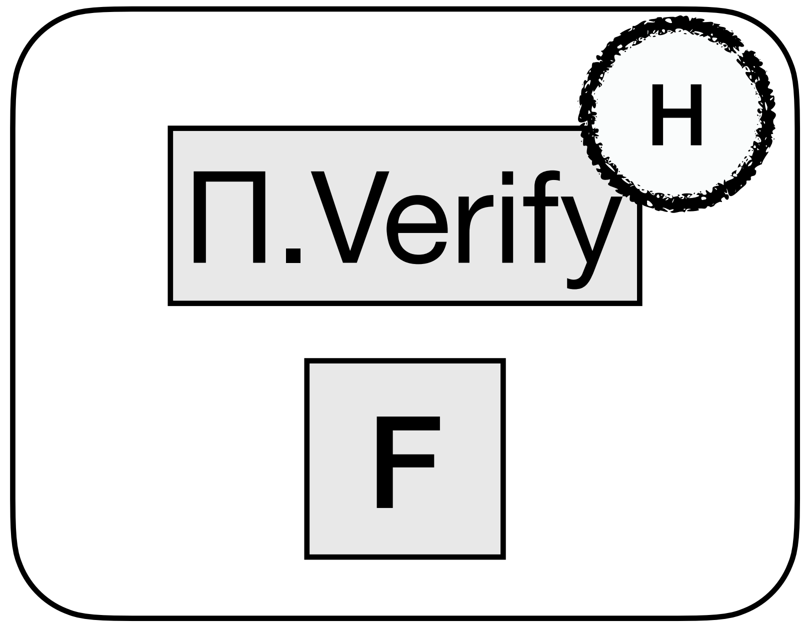
Random Oracle



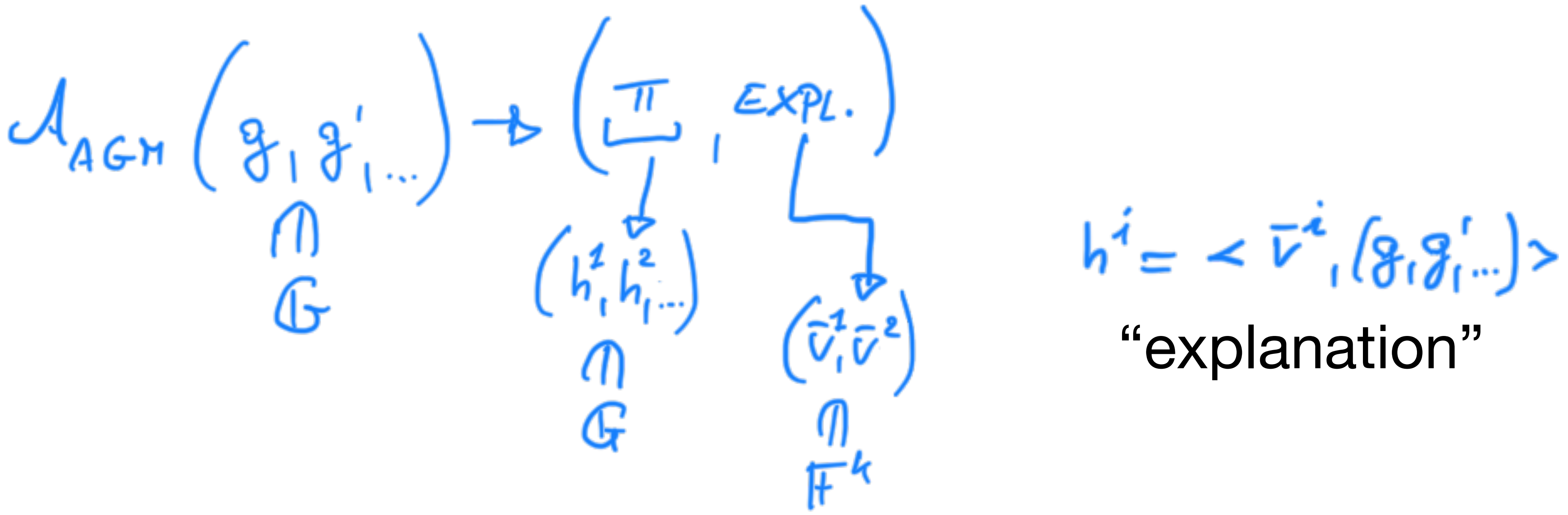
Algebraic Group Model (AGM)

Challenges in Proving the Security of IVC

- **First challenge:** idealized models and “theoretical hygiene”



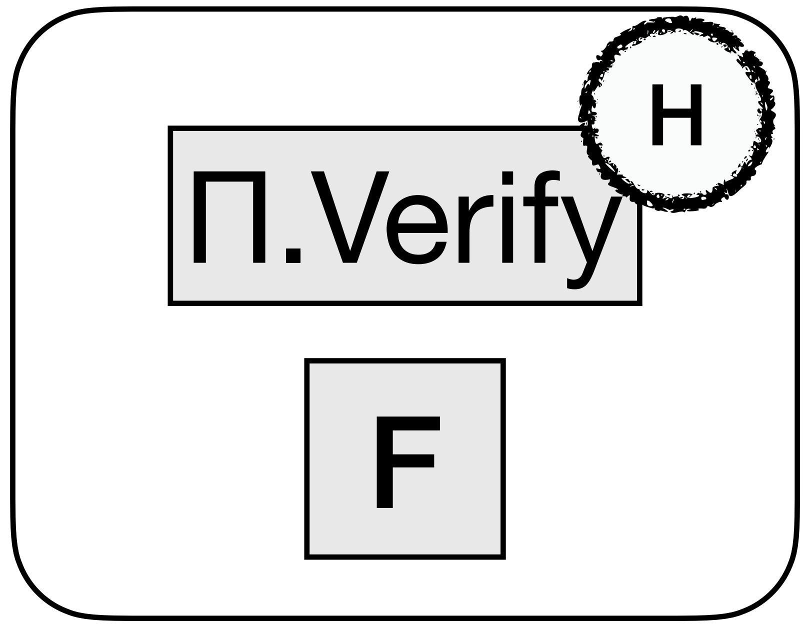
Random Oracle



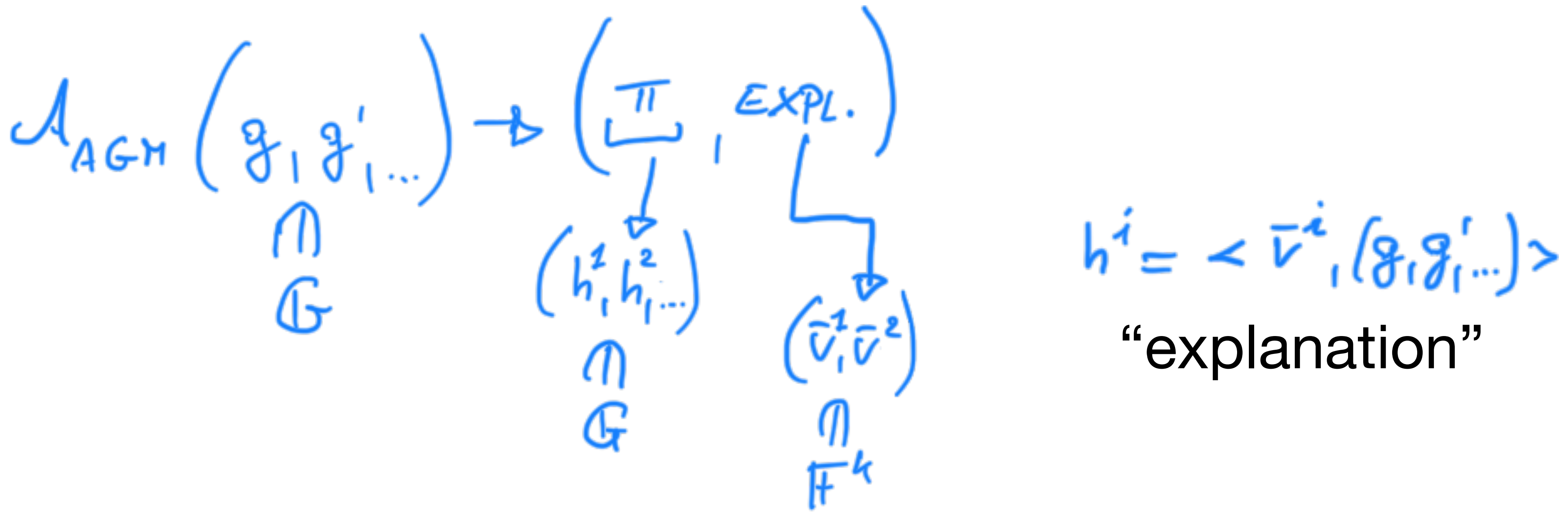
Algebraic Group Model (AGM)

Challenges in Proving the Security of IVC

- **First challenge:** idealized models and “theoretical hygiene”



Random Oracle



Algebraic Group Model (AGM)

- **Second challenge (<u>our focus</u>):** depth of the computation

How Do We Usually Prove Security in IVC?

A glimpse of what can go wrong and what depth has to do with it

$\mathcal{A}_{IVC} \to (z_0, z_d, d, \pi_d)$

$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \cdots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

How Do We Usually Prove Security in IVC?

A glimpse of what can go wrong and what depth has to do with it

IVC extractability

$\forall$ PPT $\mathcal{A}_{IVC}$ $\exists$ $\mathcal{E}_{IVC}$:

$$\left[\begin{array}{l} \mathcal{A}_{IVC} \to (z_0, z_d, d, \pi_d) \\ \mathcal{E}_{IVC} \to (w_0, \dots, w_{d-1}) \\ (z_0, z_d, d, \pi_d) \checkmark_{\text{verify}} \Rightarrow z_0 \overset{F^{(\times d)}}{\underset{w_0, \dots, w_{d-1}}{\rightsquigarrow}} z_d \end{array}\right.$$

$\mathcal{A}_{IVC} \to (z_0, z_d, d, \pi_d)$

$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \dots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

How Do We Usually Prove Security in IVC?

A glimpse of what can go wrong and what depth has to do with it

IVC extractability

$\forall$ PPT $\mathcal{A}_{\mathsf{IVC}}$ $\exists$ $\mathcal{E}_{\mathsf{IVC}}$ (poly time):

$$\left[\begin{array}{l} \mathcal{A}_{\mathsf{IVC}} \to (z_0, z_d, d, \pi_d) \\ \mathcal{E}_{\mathsf{IVC}} \to (w_0, \dots, w_{d-1}) \\ (z_0, z_d, d, \pi_d) \checkmark_{\text{verify}} \Rightarrow z_0 \overset{F\ (\times d)}{\underset{w_0,\dots,w_{d-1}}{\rightsquigarrow}} z_d \end{array}\right.$$

$\mathcal{A}_{\mathsf{IVC}} \to (z_0, z_d, d, \pi_d)$

$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \dots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

How Do We Usually Prove Security in IVC?

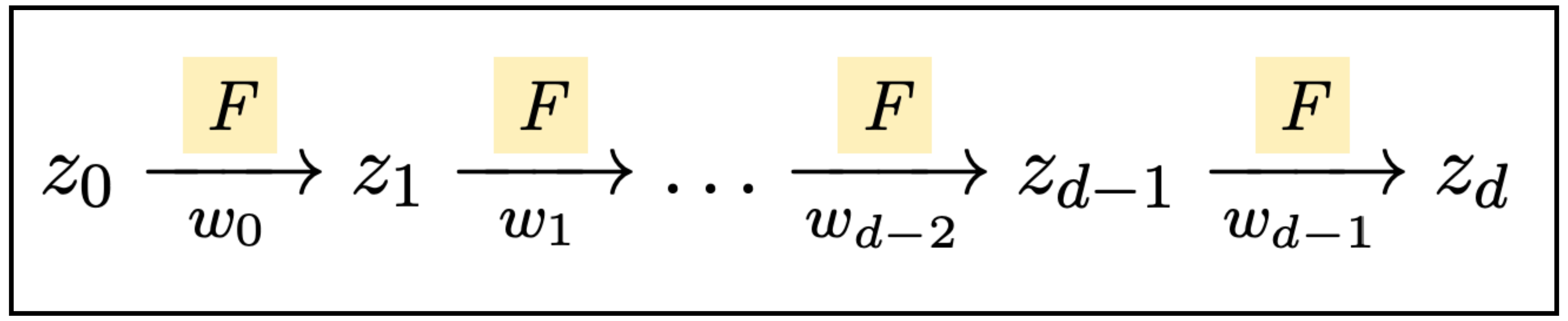
A glimpse of what can go wrong and what depth has to do with it

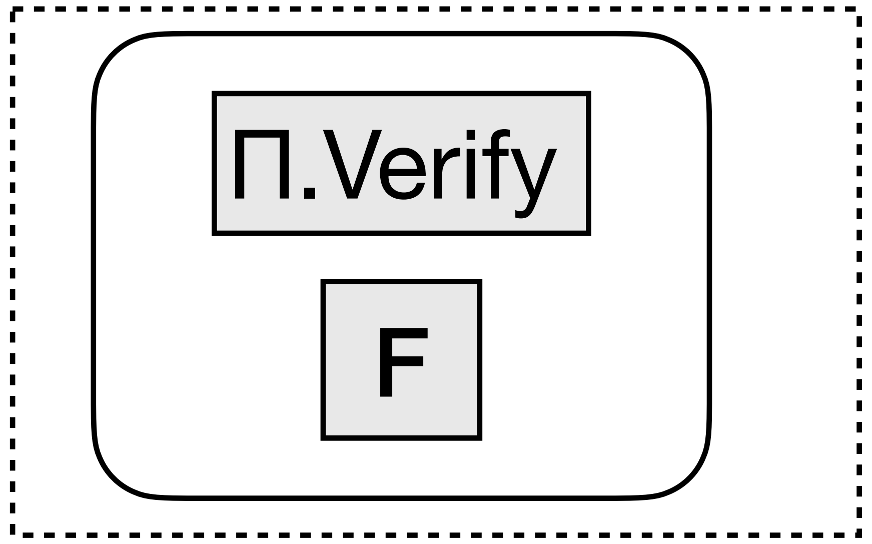
IVC extractability

$\forall$ PPT $\mathcal{A}_{\mathrm{ivc}}$ $\exists$ $\mathcal{E}_{\mathrm{ivc}}$ (poly time):

- $\mathcal{A}_{\mathrm{ivc}} \rightarrow (z_0, z_d, d, \pi_d)$
- $\mathcal{E}_{\mathrm{ivc}} \rightarrow (w_0, \ldots, w_{d-1})$
- (z_0, z_d, d, π_d) ✓ Verify $\Rightarrow z_0 \xrightarrow[w_0, \ldots, w_{d-1}]{F\,(\times d)} z_d$

$\mathcal{A}_{\mathrm{ivc}} \rightarrow (z_0, z_d, d, \pi_d)$





SNARK Prover's statement (Π is a SNARK)

How Do We Usually Prove Security in IVC?

A glimpse of what can go wrong and what depth has to do with it

IVC extractability

(poly time)

$\forall$ PPT $\mathcal{A}_{ivc}$ $\exists\, \mathcal{E}_{ivc}$:

$$\left[\begin{array}{l} \mathcal{A}_{ivc} \to (z_0, z_d, d, \pi_d) \\ \mathcal{E}_{ivc} \to (w_0, \dots, w_{d-1}) \\ (z_0, z_d, d, \pi_d)\ \checkmark_{\text{verify}} \Rightarrow z_0 \xrightarrow[w_0, \dots, w_{d-1}]{F\ (\times d)} z_d \end{array}\right.$$

$$\mathcal{A}_{ivc} \to (z_0, z_d, d, \pi_d)$$

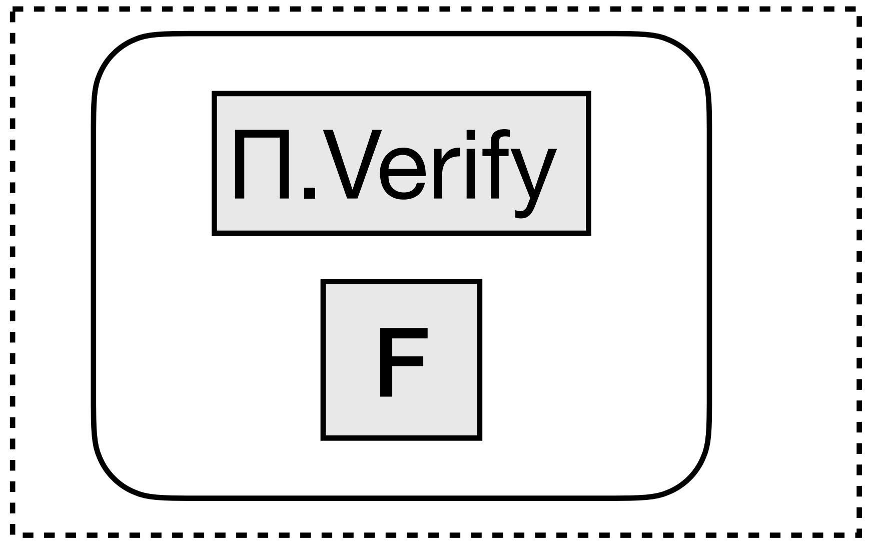
$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \dots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

SNARK extractability

(poly time)

$\forall$ PPT $\mathcal{A}_{\Pi}$ $\exists\, \mathcal{E}_{\mathcal{A}}$:

$$\left[\begin{array}{l} \mathcal{A}_{\Pi} \to (x, \pi) \\ \mathcal{E}_{\mathcal{A}} \to w \\ (x, \pi)\ \checkmark_{\text{verify}} \Rightarrow (x, w) \in R_F \end{array}\right.$$

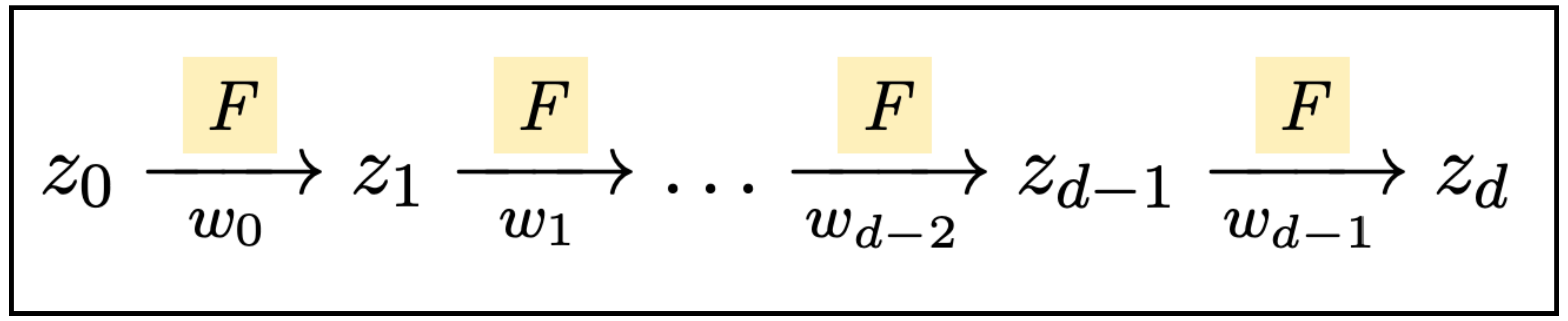


SNARK Prover's statement (Π is a SNARK)

How Do We Usually Prove Security in IVC?

A glimpse of what can go wrong and what depth has to do with it

$\mathcal{A}_{ivc} \to (z_0, z_d, d, \pi_d)$



IVC extractability

$\forall$ PPT $\mathcal{A}_{ivc}$ $\exists\, \mathcal{E}_{ivc}$ (poly time):

$\mathcal{A}_{ivc} \to (z_0, z_d, d, \pi_d)$

$\mathcal{E}_{ivc} \to (w_0, \dots, w_{d-1})$

(z_0, z_d, d, π_d) ✓ verify $\Rightarrow z_0 \xrightarrow[w_0,\dots,w_{d-1}]{F\ (\times d)} z_d$

SNARK extractability

$\forall$ PPT $\mathcal{A}_\Pi$ $\exists\, \mathcal{E}_\mathcal{A}$ (poly time):

$\mathcal{A}_\Pi \to (x, \pi)$

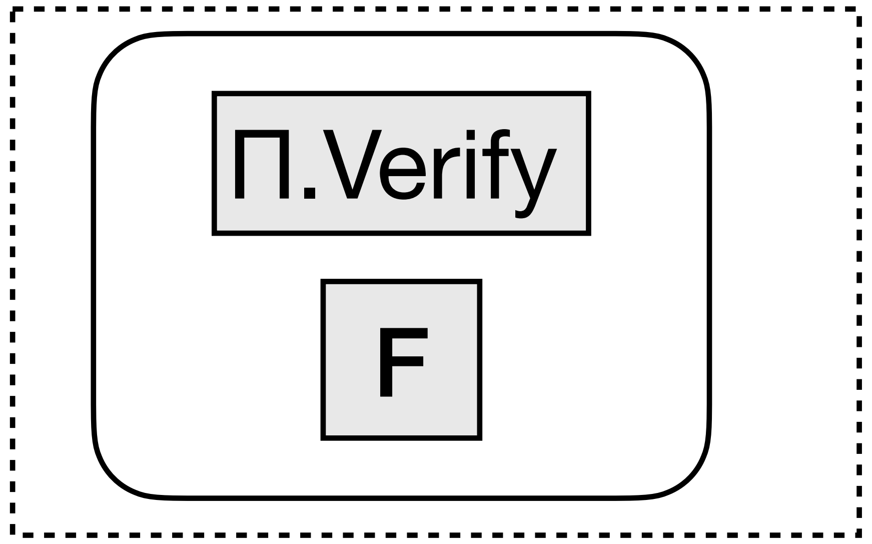
$\mathcal{E}_\mathcal{A} \to w$

(x, π) ✓ verify $\Rightarrow (x, w) \in R_F$

OBS:

$|\mathcal{A}| = T$

$\Rightarrow \mathcal{E}_\mathcal{A} = T^k \quad (k = O(1))$



SNARK Prover's statement (Π is a SNARK)

How Do We Usually Prove Security in IVC?

A glimpse of what can go wrong and what depth has to do with it

IVC extractability

(poly time)

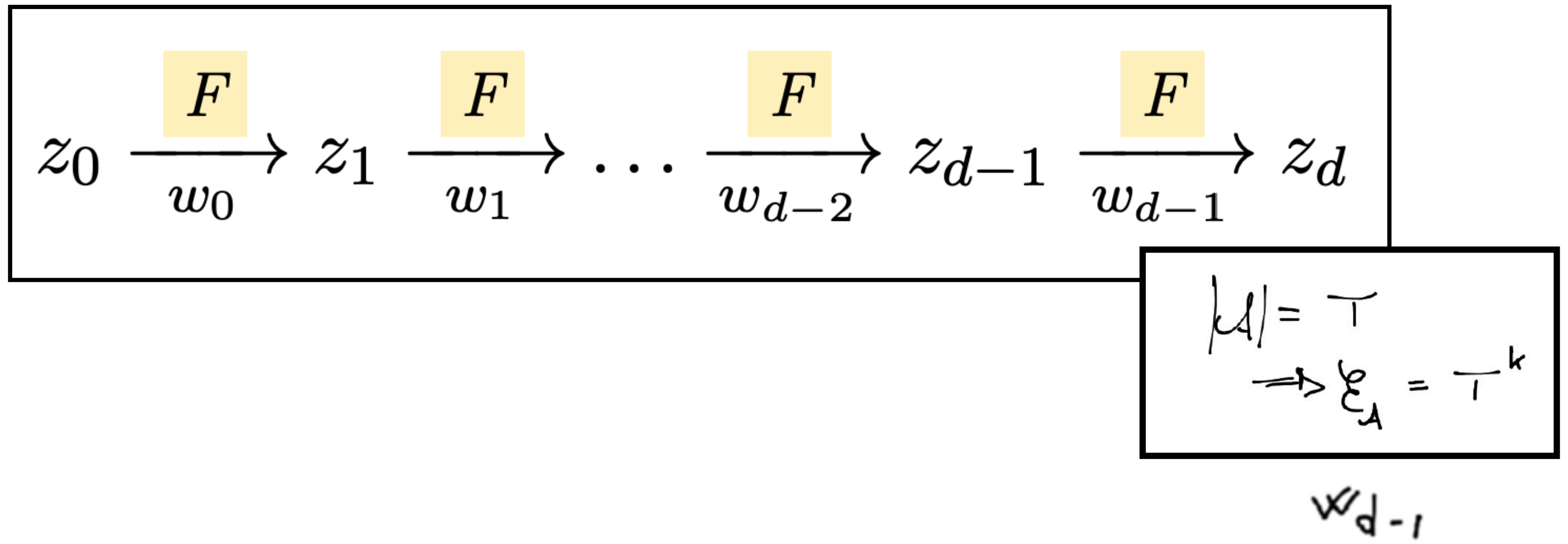
$\forall$ PPT $\mathcal{A}_{IVC}$ $\exists\, \mathcal{E}_{IVC}$:

$\mathcal{A}_{IVC} \rightarrow (z_0, z_d, d, \pi_d)$

$\mathcal{E}_{IVC} \rightarrow (w_0, \ldots, w_{d-1})$

(z_0, z_d, d, π_d) ✓ verify $\Rightarrow z_0 \xrightarrow[w_0, \ldots, w_{d-1}]{F\ (\times d)} z_d$

$\mathcal{A}_{IVC} \rightarrow (z_0, z_d, d, \pi_d)$



SNARK extractability

(poly time)

$\forall$ PPT $\mathcal{A}_{\Pi}$ $\exists\, \mathcal{E}_{\mathcal{A}}$:

$\mathcal{A}_{\Pi} \rightarrow (x, \pi)$

$\mathcal{E}_{\mathcal{A}} \rightarrow w$

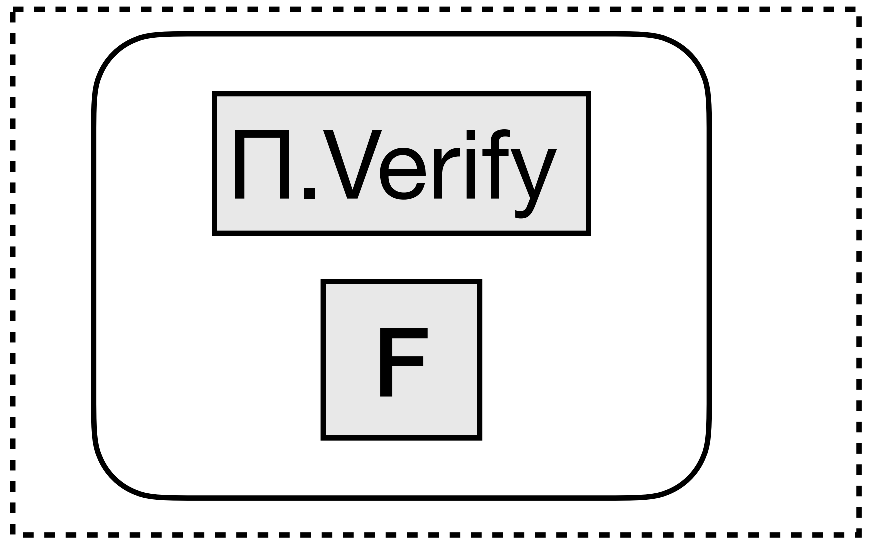
(x, π) ✓ verify $\Rightarrow (x, w) \in R_F$

OBS:

$|\mathcal{A}| = T$

$\Rightarrow \mathcal{E}_{\mathcal{A}} = T^k$

$(k = O(1))$



SNARK Prover's statement (Π is a SNARK)

How Do We Usually Prove Security in IVC?

A glimpse of what can go wrong and what depth has to do with it

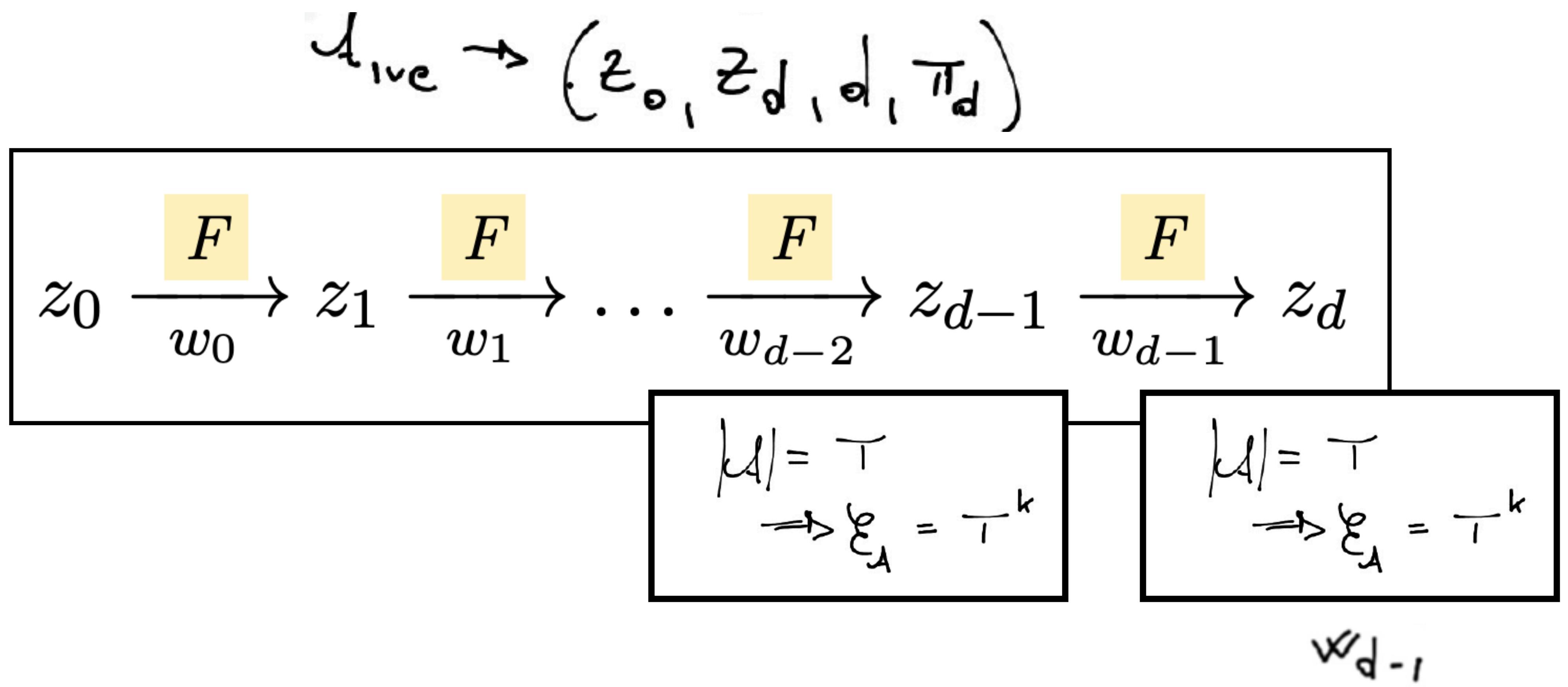
IVC extractability

$\forall$ PPT $\mathcal{A}_{IVC}$ $\exists \mathcal{E}_{IVC}$ (poly time):

$\mathcal{A}_{IVC} \to (z_0, z_d, d, \pi_d)$

$\mathcal{E}_{IVC} \to (w_0, \dots, w_{d-1})$

(z_0, z_d, d, π_d) ✓ verify $\Rightarrow z_0 \xrightarrow[w_0, \dots, w_{d-1}]{F^{(\times d)}} z_d$



SNARK extractability

$\forall$ PPT $\mathcal{A}_{\Pi}$ $\exists \mathcal{E}_{\mathcal{A}}$ (poly time):

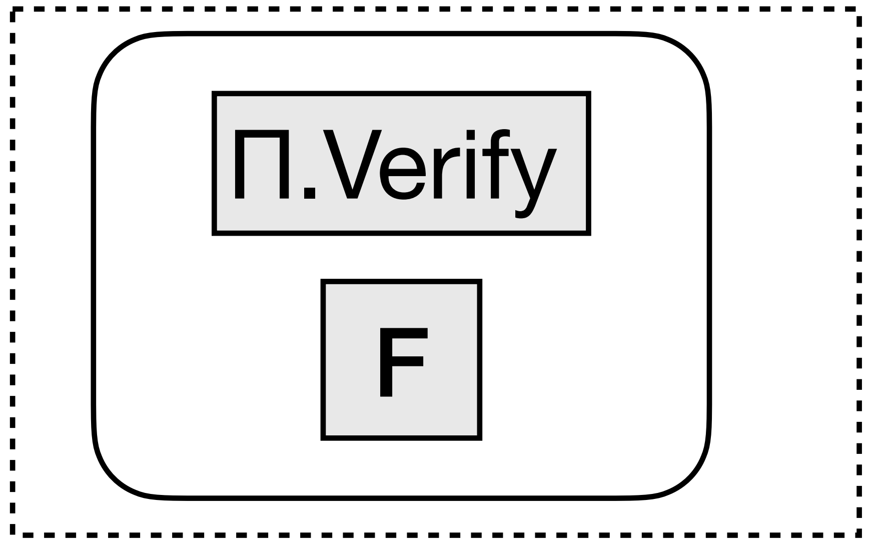
$\mathcal{A}_{\Pi} \to (x, \pi)$

$\mathcal{E}_{\mathcal{A}} \to w$

(x, π) ✓ verify $\Rightarrow (x, w) \in R_F$

OBS:

$|\mathcal{A}| = T \Rightarrow \mathcal{E}_{\mathcal{A}} = T^k$ $(k = O(1))$



SNARK Prover's statement (Π is a SNARK)

How Do We Usually Prove Security in IVC?

A glimpse of what can go wrong and what depth has to do with it

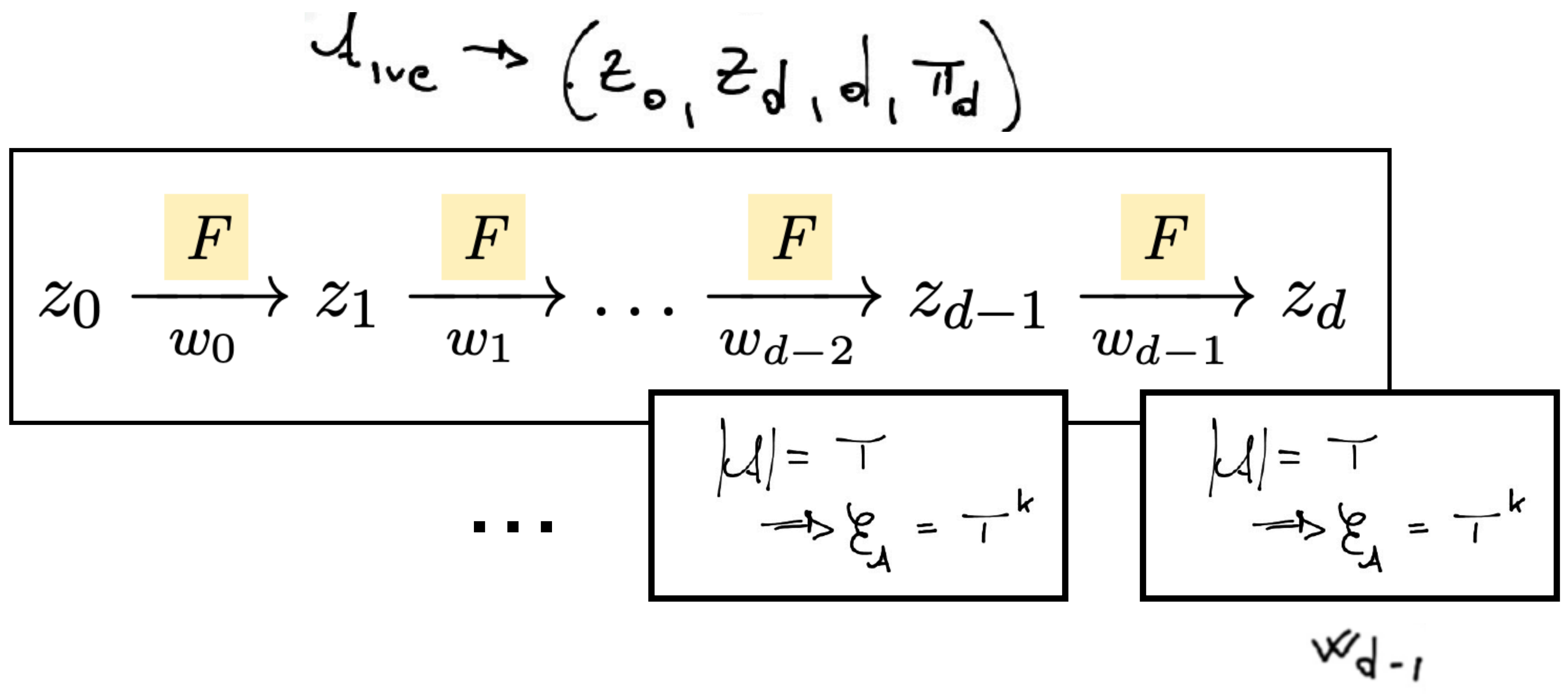
IVC extractability

$\forall$ PPT $\mathcal{A}_{IVC}$ $\exists \mathcal{E}_{IVC}$ (poly time):

$\mathcal{A}_{IVC} \to (z_0, z_d, d, \pi_d)$

$\mathcal{E}_{IVC} \to (w_0, \ldots, w_{d-1})$

(z_0, z_d, d, π_d) ✓ verify $\Rightarrow z_0 \xrightarrow[w_0, \ldots, w_{d-1}]{F \ (\times d)} z_d$



SNARK extractability

$\forall$ PPT $\mathcal{A}_\pi$ $\exists \mathcal{E}_\mathcal{A}$ (poly time):

$\mathcal{A}_\pi \to (x, \pi)$

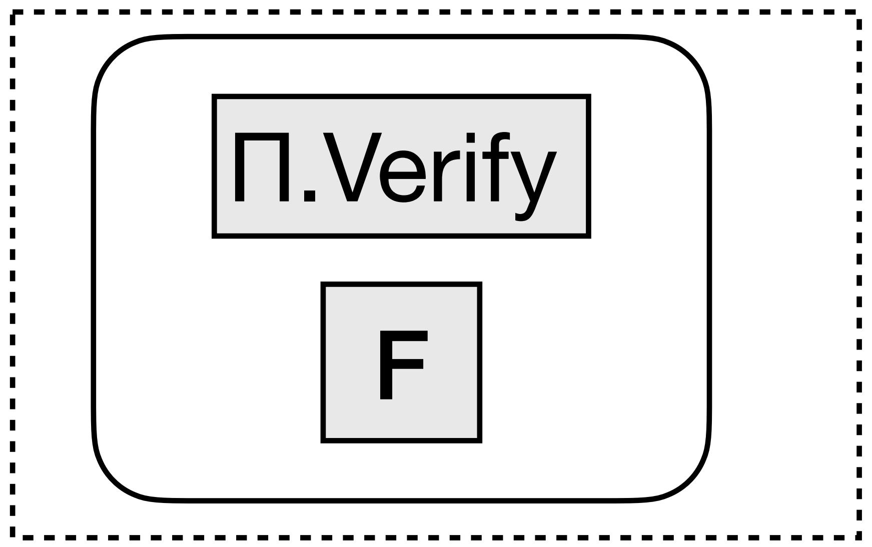
$\mathcal{E}_\mathcal{A} \to w$

(x, π) ✓ verify $\Rightarrow (x, w) \in R_F$

OBS:

$|\mathcal{A}| = T$

$\Rightarrow \mathcal{E}_\mathcal{A} = T^k$ $(k = O(1))$



SNARK Prover's statement (Π is a SNARK)

How Do We Usually Prove Security in IVC?

A glimpse of what can go wrong and what depth has to do with it

IVC extractability

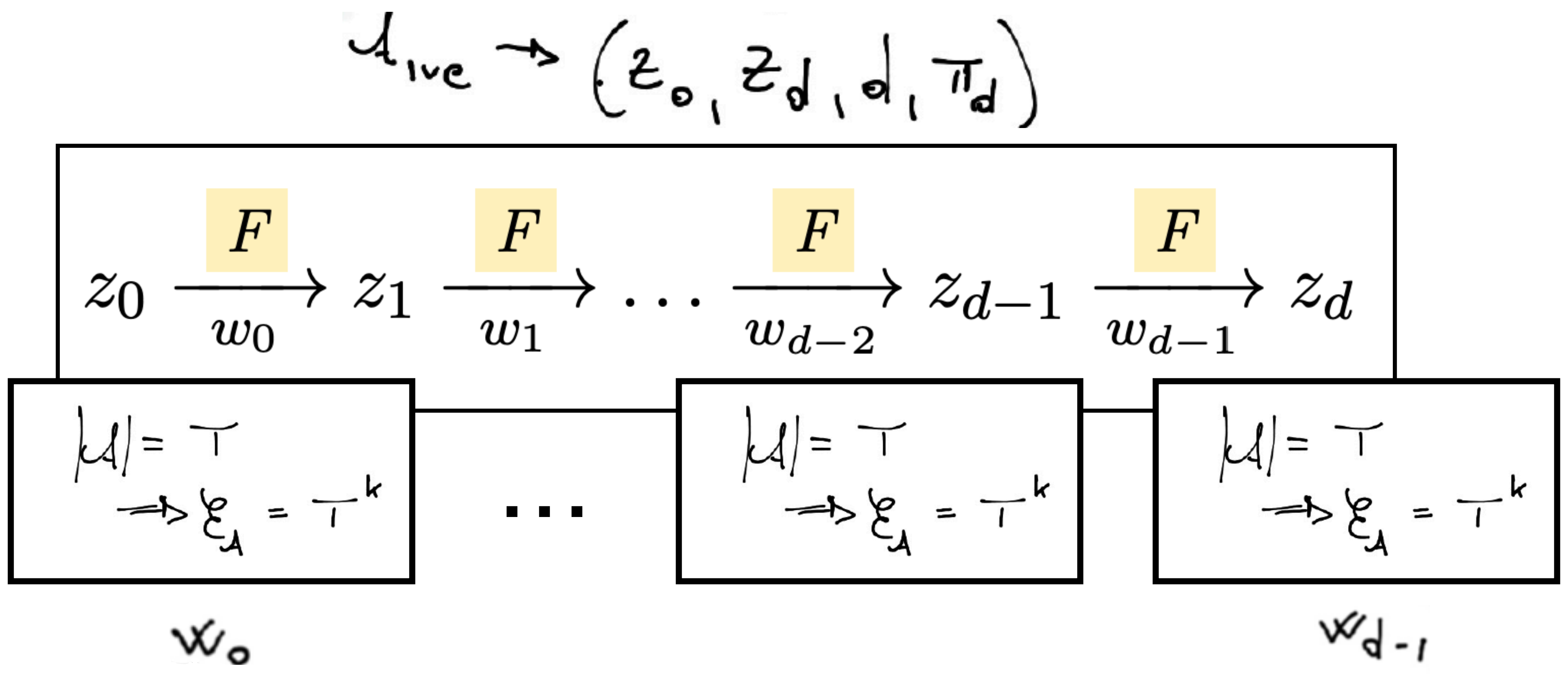
(poly time)

$\forall$ PPT $\mathcal{A}_{ivc}$ $\exists$ $\mathcal{E}_{ivc}$:

$\mathcal{A}_{ivc} \rightarrow (z_0, z_d, d, \pi_d)$

$\mathcal{E}_{ivc} \rightarrow (w_0, \ldots, w_{d-1})$

(z_0, z_d, d, π_d) ✓ verify $\Rightarrow z_0 \overset{F\,(\times d)}{\rightsquigarrow} z_d$ ($w_0, \ldots, w_{d-1}$)



SNARK extractability

(poly time)

$\forall$ PPT $\mathcal{A}_\pi$ $\exists$ $\mathcal{E}_\mathcal{A}$:

$\mathcal{A}_\pi \rightarrow (x, \pi)$

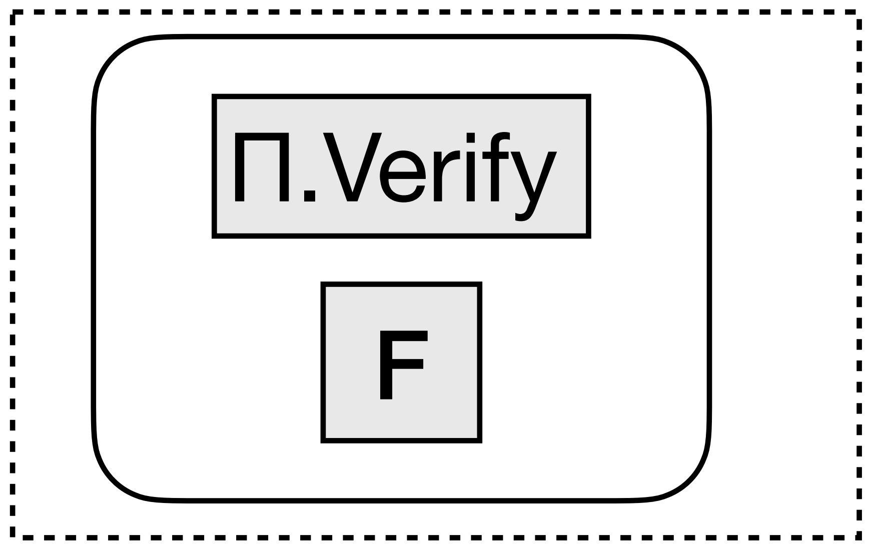
$\mathcal{E}_\mathcal{A} \rightarrow w$

(x, π) ✓ verify $\Rightarrow (x, w) \in R_F$

OBS:

$|\mathcal{A}| = T$

$\rightarrow \mathcal{E}_\mathcal{A} = T^k$ $(k = O(1))$

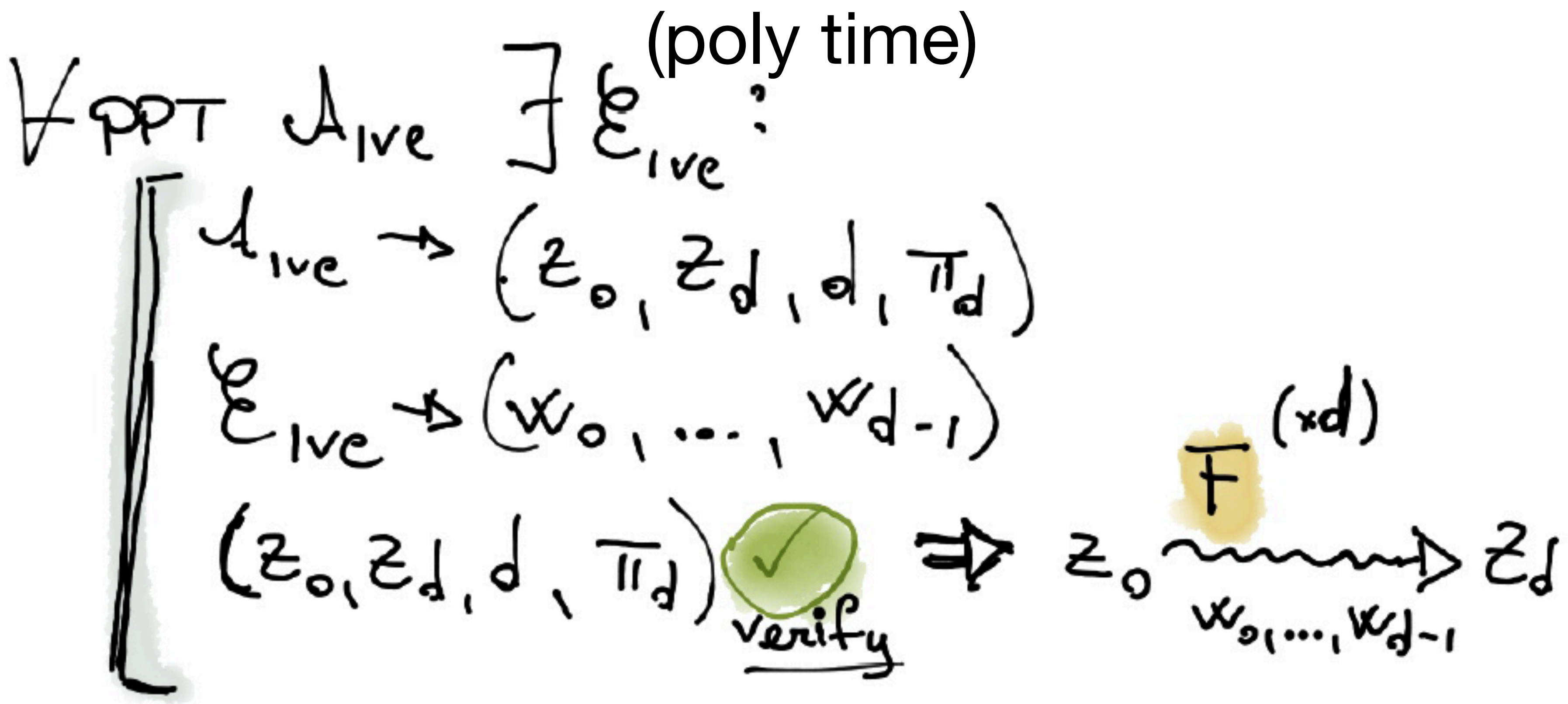


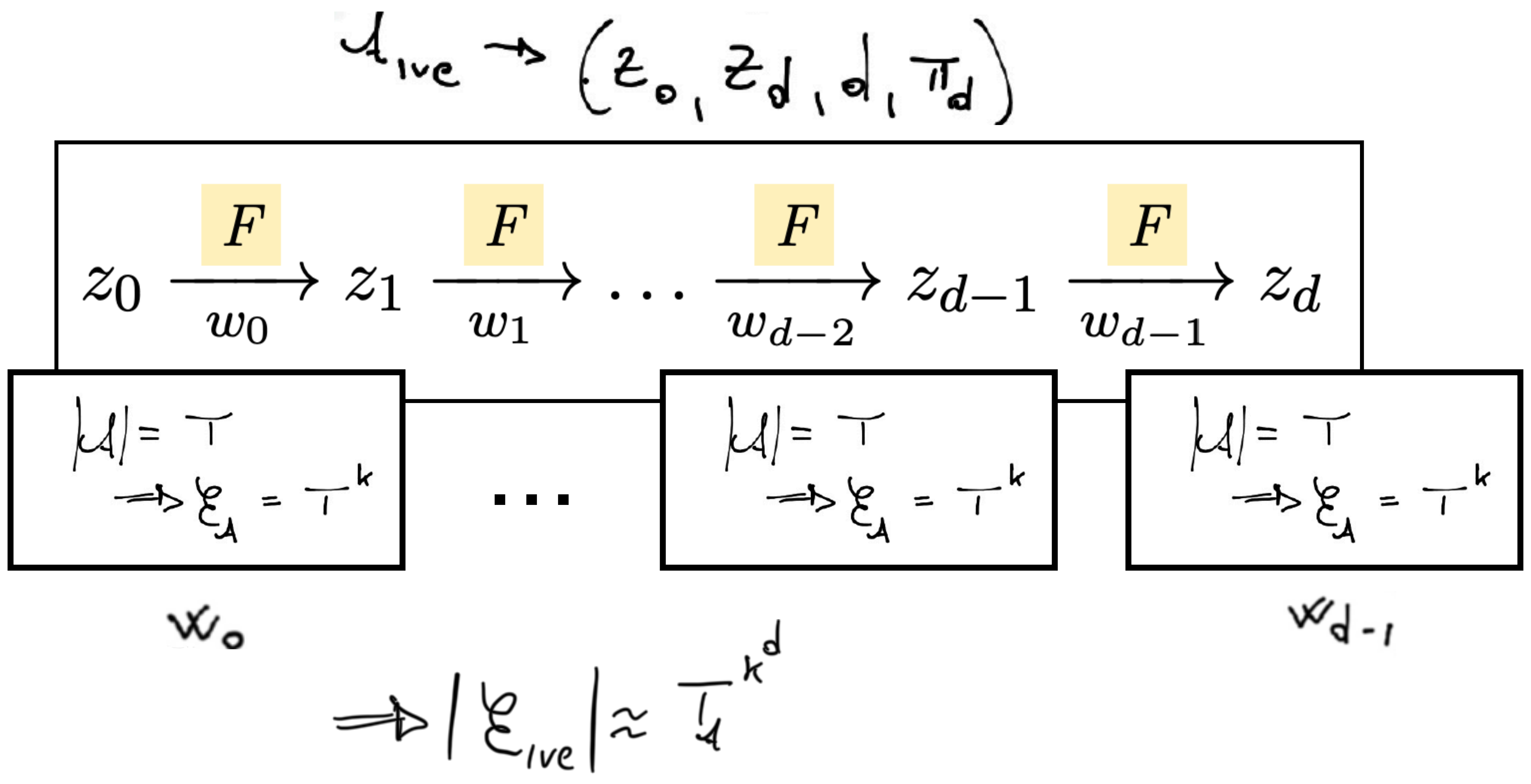
SNARK Prover's statement (Π is a SNARK)

How Do We Usually Prove Security in IVC?

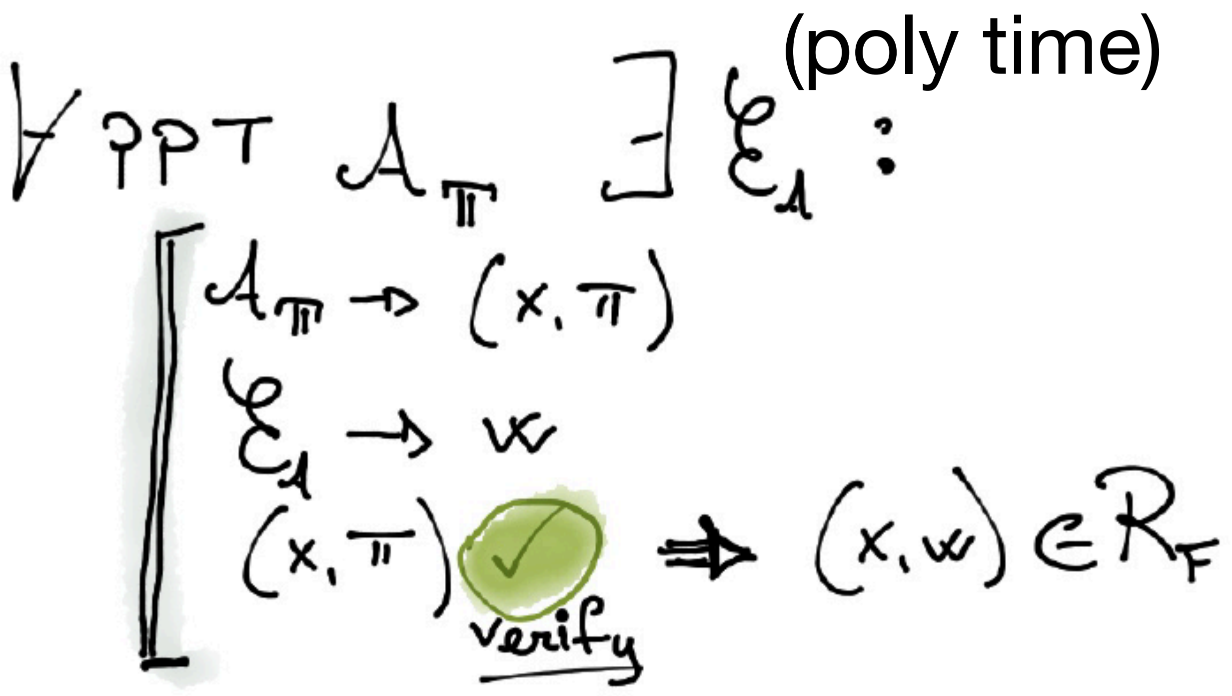
A glimpse of what can go wrong and what depth has to do with it

IVC extractability

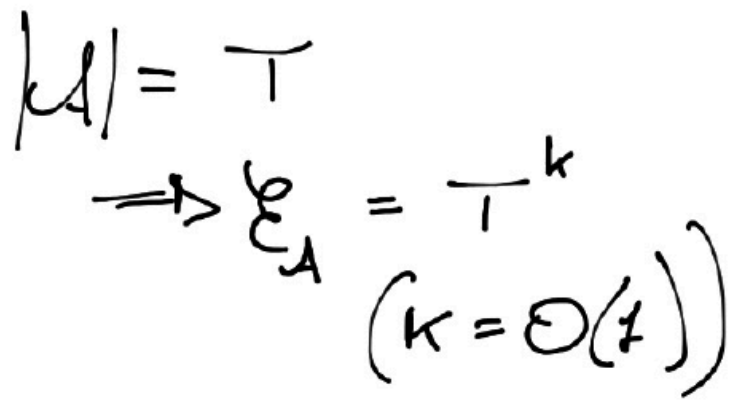


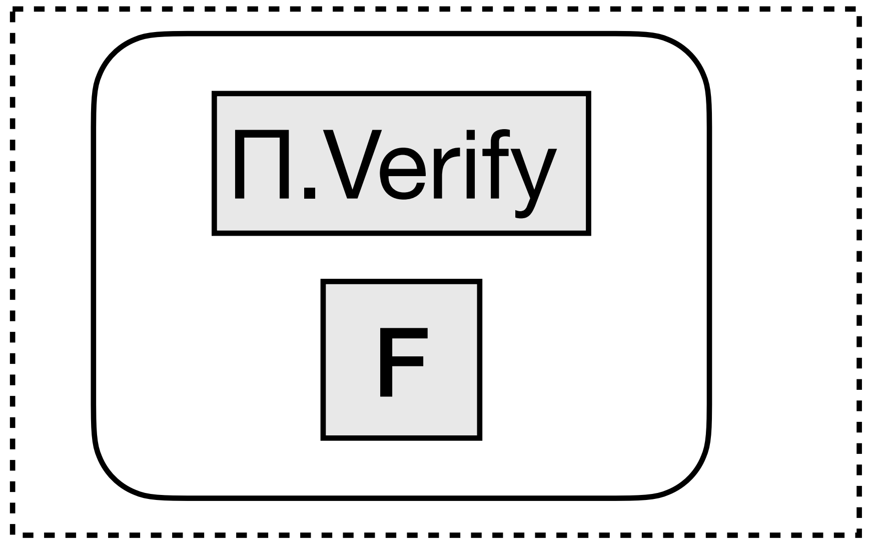


SNARK extractability



OBS:





SNARK Prover's statement (Π is a SNARK)

How Do We Usually Prove Security in IVC?

A glimpse of what can go wrong and what depth has to do with it

IVC extractability

$\forall$ PPT $\mathcal{A}_{ivc}$ $\exists\, \mathcal{E}_{ivc}$ (poly time):

- $\mathcal{A}_{ivc} \to (z_0, z_d, d, \pi_d)$
- $\mathcal{E}_{ivc} \to (w_0, \ldots, w_{d-1})$
- (z_0, z_d, d, π_d) ✓ verify $\Rightarrow z_0 \xrightarrow[w_0,\ldots,w_{d-1}]{F\ (\times d)} z_d$

SNARK extractability

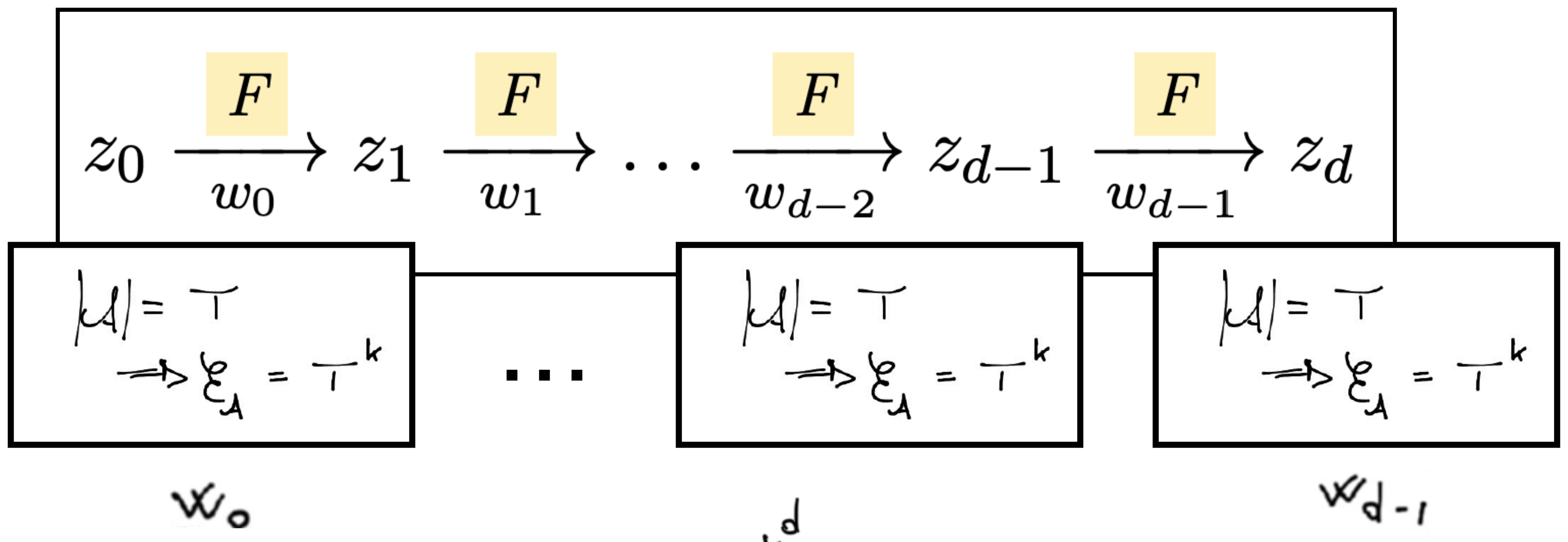
$\forall$ PPT $\mathcal{A}_\pi$ $\exists\, \mathcal{E}_\mathcal{A}$ (poly time):

- $\mathcal{A}_\pi \to (x, \pi)$
- $\mathcal{E}_\mathcal{A} \to w$
- (x, π) ✓ verify $\Rightarrow (x, w) \in R_F$

OBS:

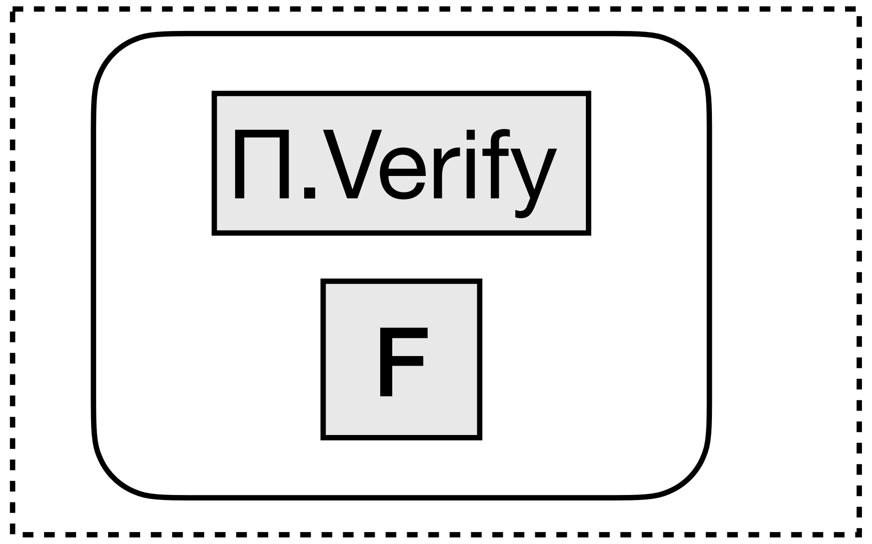
$|\mathcal{A}| = T \Rightarrow \mathcal{E}_\mathcal{A} = T^k$ $(k = O(1))$

$\mathcal{A}_{ivc} \to (z_0, z_d, d, \pi_d)$



$\Rightarrow |\mathcal{E}_{ivc}| \approx T_\mathcal{A}^{k^d}$

$\Rightarrow \left(d \in \omega(1) \Rightarrow |\mathcal{E}_{ivc}| \notin poly\right)$



SNARK Prover's statement (Π is a SNARK)

The “Tree Approach”

A canonical way to go around the problem we just saw (via extractability)

The “Tree Approach”

A canonical way to go around the problem we just saw (via extractability)

Recursive Composition and Bootstrapping
for SNARKs and Proof-Carrying Data

Nir Bitansky*
nirbitan@tau.ac.il
Tel Aviv University

Ran Canetti*
canetti@tau.ac.il
Boston University and
Tel Aviv University

Alessandro Chiesa
alexch@csail.mit.edu
MIT

Eran Tromer†
tromer@tau.ac.il
Tel Aviv University

December 28, 2012

The “Tree Approach”

A canonical way to go around the problem we just saw (via extractability)

Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data

Nir Bitansky*
nirbitan@tau.ac.il
Tel Aviv University

Ran Canetti*
canetti@tau.ac.il
Boston University and Tel Aviv University

Alessandro Chiesa
alexch@csail.mit.edu
MIT

Eran Tromer†
tromer@tau.ac.il
Tel Aviv University

December 28, 2012

Mangrove: A Scalable Framework for Folding-based SNARKs

Wilson Nguyen Trisha Datta Binyi Chen Nirvan Tyagi Dan Boneh

{wdnguyen, tcdatta, binyi, tyagi, dabo}@cs.stanford.edu
Stanford University

The “Tree Approach”

A canonical way to go around the problem we just saw (via extractability)

Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data

Nir Bitansky*
nirbitan@tau.ac.il
Tel Aviv University

Ran Canetti*
canetti@tau.ac.il
Boston University and Tel Aviv University

Alessandro Chiesa
alexch@csail.mit.edu
MIT

Eran Tromer†
tromer@tau.ac.il
Tel Aviv University

December 28, 2012

Mangrove: A Scalable Framework for Folding-based SNARKs

Wilson Nguyen Trisha Datta Binyi Chen Nirvan Tyagi Dan Boneh

{wdnguyen, tcdatta, binyi, tyagi, dabo}@cs.stanford.edu
Stanford University

Idea: proceed in a tree fashion rather than along a path.

The “Tree Approach”

A canonical way to go around the problem we just saw (via extractability)

Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data

Nir Bitansky*
nirbitan@tau.ac.il
Tel Aviv University

Ran Canetti*
canetti@tau.ac.il
Boston University and Tel Aviv University

Alessandro Chiesa
alexch@csail.mit.edu
MIT

Eran Tromer†
tromer@tau.ac.il
Tel Aviv University

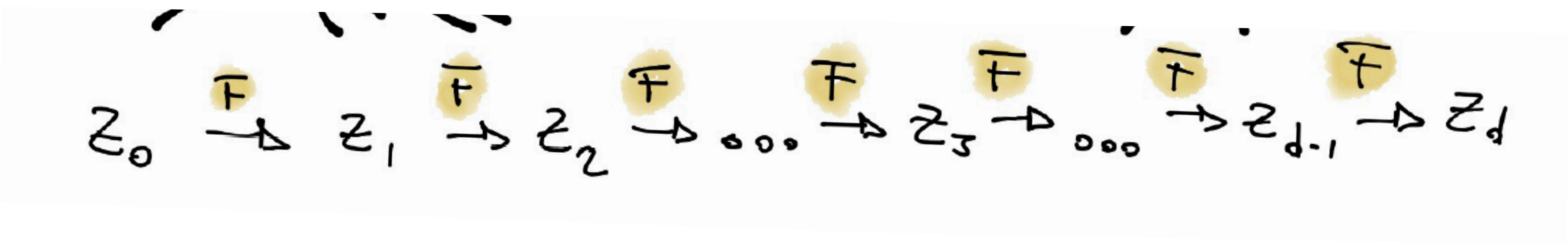
December 28, 2012

Mangrove: A Scalable Framework for Folding-based SNARKs

Wilson Nguyen Trisha Datta Binyi Chen Nirvan Tyagi Dan Boneh

{wdnguyen, tcdatta, binyi, tyagi, dabo}@cs.stanford.edu
Stanford University

Idea: proceed in a tree fashion rather than along a path.



The "Tree Approach"

A canonical way to go around the problem we just saw (via extractability)

Recursive Composition and Bootstrapping
for SNARKs and Proof-Carrying Data

Nir Bitansky*
nirbitan@tau.ac.il
Tel Aviv University

Ran Canetti*
canetti@tau.ac.il
Boston University and
Tel Aviv University

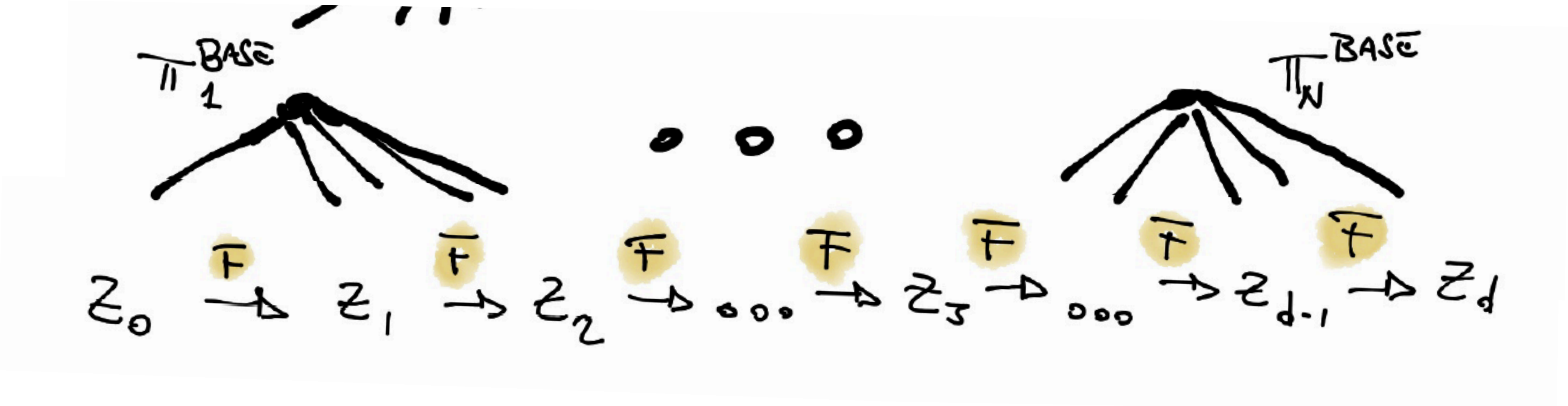
Alessandro Chiesa
alexch@csail.mit.edu
MIT

Eran Tromer†
tromer@tau.ac.il
Tel Aviv University

December 28, 2012

Mangrove: A Scalable Framework for Folding-based SNARKs

Wilson Nguyen Trisha Datta Binyi Chen Nirvan Tyagi Dan Boneh
{wdnguyen, tcdatta, binyi, tyagi, dabo}@cs.stanford.edu
Stanford University



Idea: proceed in a tree fashion rather than along a path.

The "Tree Approach"

A canonical way to go around the problem we just saw (via extractability)

Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data

Nir Bitansky*
nirbitan@tau.ac.il
Tel Aviv University

Ran Canetti*
canetti@tau.ac.il
Boston University and Tel Aviv University

Alessandro Chiesa
alexch@csail.mit.edu
MIT

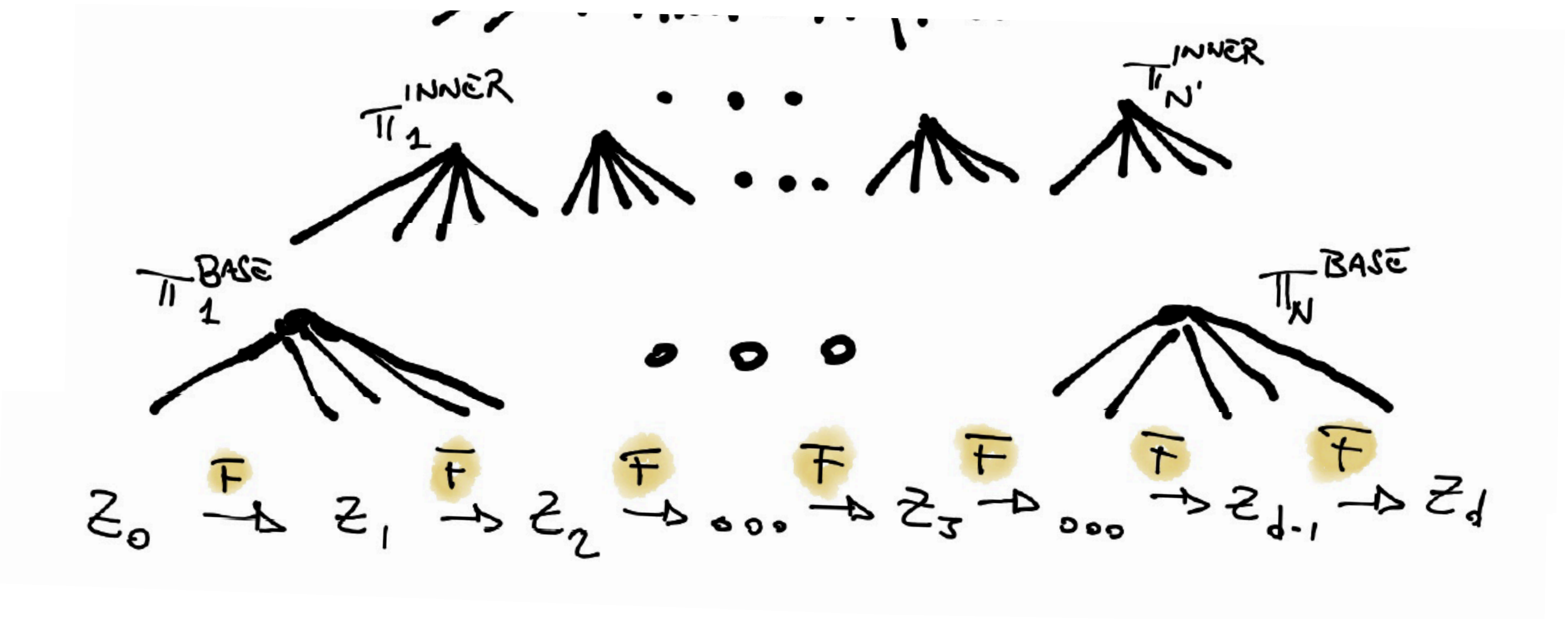
Eran Tromer†
tromer@tau.ac.il
Tel Aviv University

December 28, 2012

Mangrove: A Scalable Framework for Folding-based SNARKs

Wilson Nguyen Trisha Datta Binyi Chen Nirvan Tyagi Dan Boneh

{wdnguyen, tcdatta, binyi, tyagi, dabo}@cs.stanford.edu
Stanford University



Idea: proceed in a tree fashion rather than along a path.

The “Tree Approach”

A canonical way to go around the problem we just saw (via extractability)

Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data

Nir Bitansky*
nirbitan@tau.ac.il
Tel Aviv University

Ran Canetti*
canetti@tau.ac.il
Boston University and Tel Aviv University

Alessandro Chiesa
alexch@csail.mit.edu
MIT

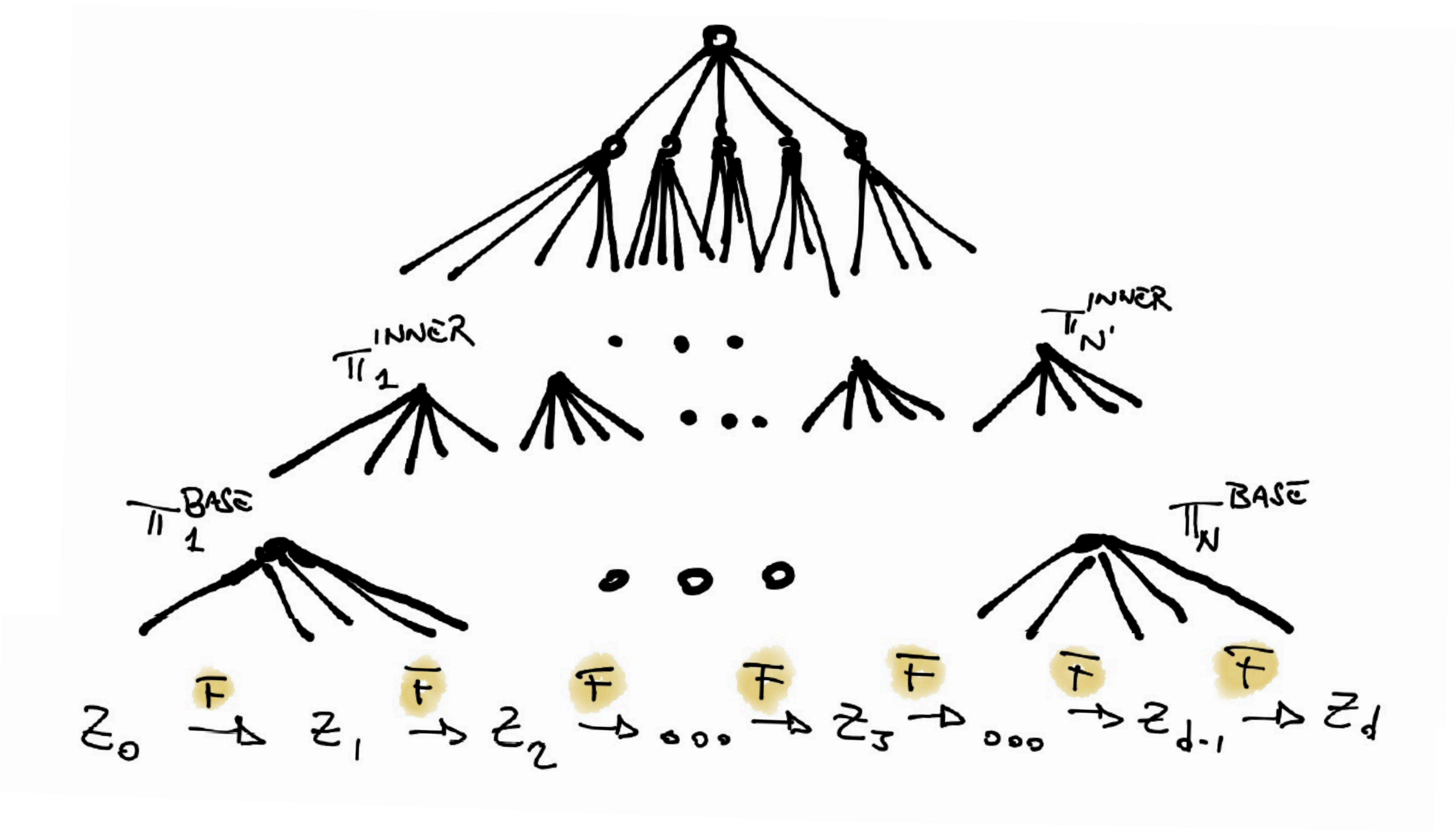
Eran Tromer†
tromer@tau.ac.il
Tel Aviv University

December 28, 2012

Mangrove: A Scalable Framework for Folding-based SNARKs

Wilson Nguyen Trisha Datta Binyi Chen Nirvan Tyagi Dan Boneh

{wdnguyen, tcdatta, binyi, tyagi, dabo}@cs.stanford.edu
Stanford University



Idea: proceed in a tree fashion rather than along a path.

The “Tree Approach”

A canonical way to go around the problem we just saw (via extractability)

Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data

Nir Bitansky*
nirbitan@tau.ac.il
Tel Aviv University

Ran Canetti*
canetti@tau.ac.il
Boston University and Tel Aviv University

Alessandro Chiesa
alexch@csail.mit.edu
MIT

Eran Tromer†
tromer@tau.ac.il
Tel Aviv University

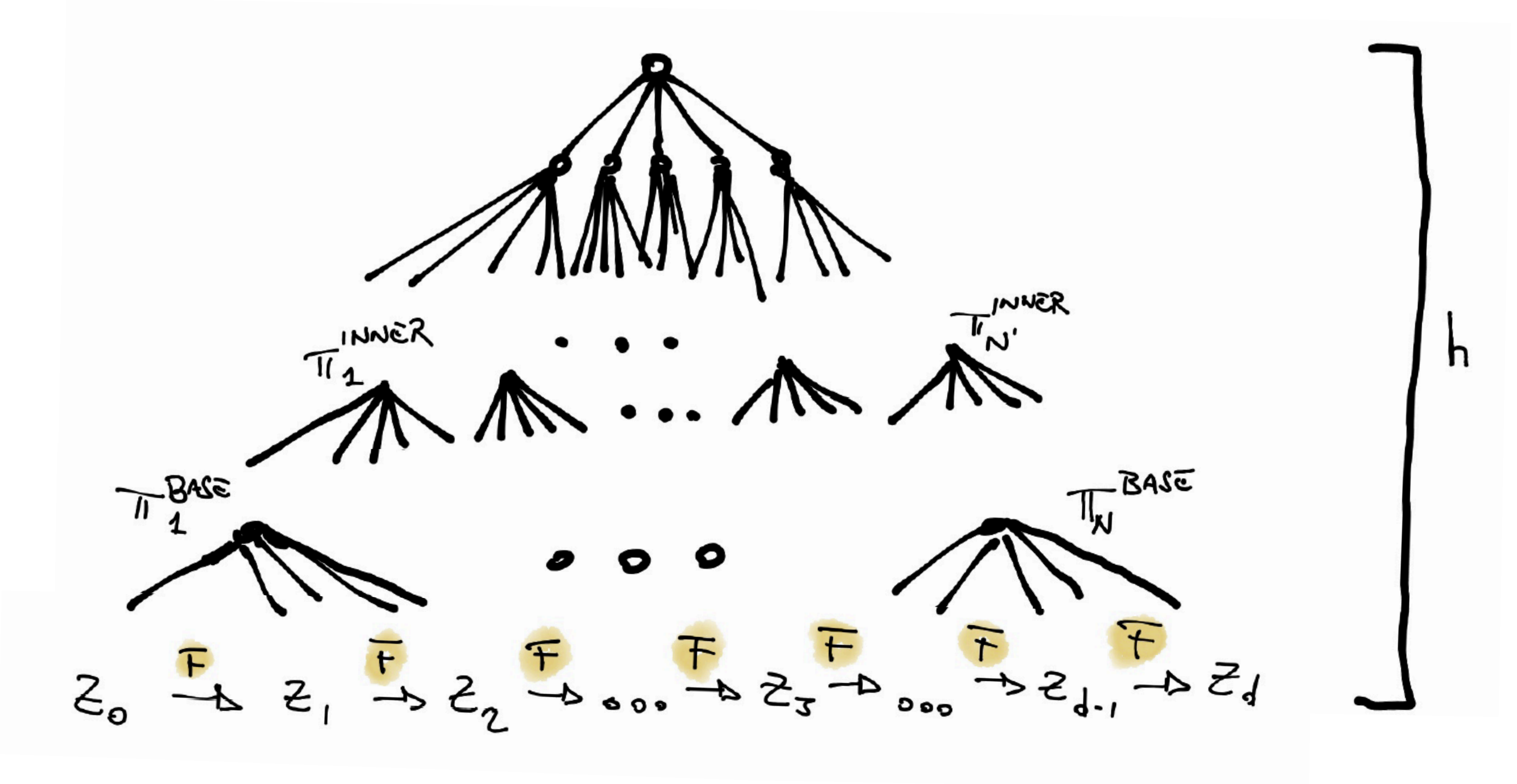
December 28, 2012

Mangrove: A Scalable Framework for Folding-based SNARKs

Wilson Nguyen Trisha Datta Binyi Chen Nirvan Tyagi Dan Boneh

{wdnguyen, tcdatta, binyi, tyagi, dabo}@cs.stanford.edu
Stanford University

Idea: proceed in a tree fashion rather than along a path.



The “Tree Approach”

A canonical way to go around the problem we just saw (via extractability)

Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data

Nir Bitansky*
nirbitan@tau.ac.il
Tel Aviv University

Ran Canetti*
canetti@tau.ac.il
Boston University and Tel Aviv University

Alessandro Chiesa
alexch@csail.mit.edu
MIT

Eran Tromer†
tromer@tau.ac.il
Tel Aviv University

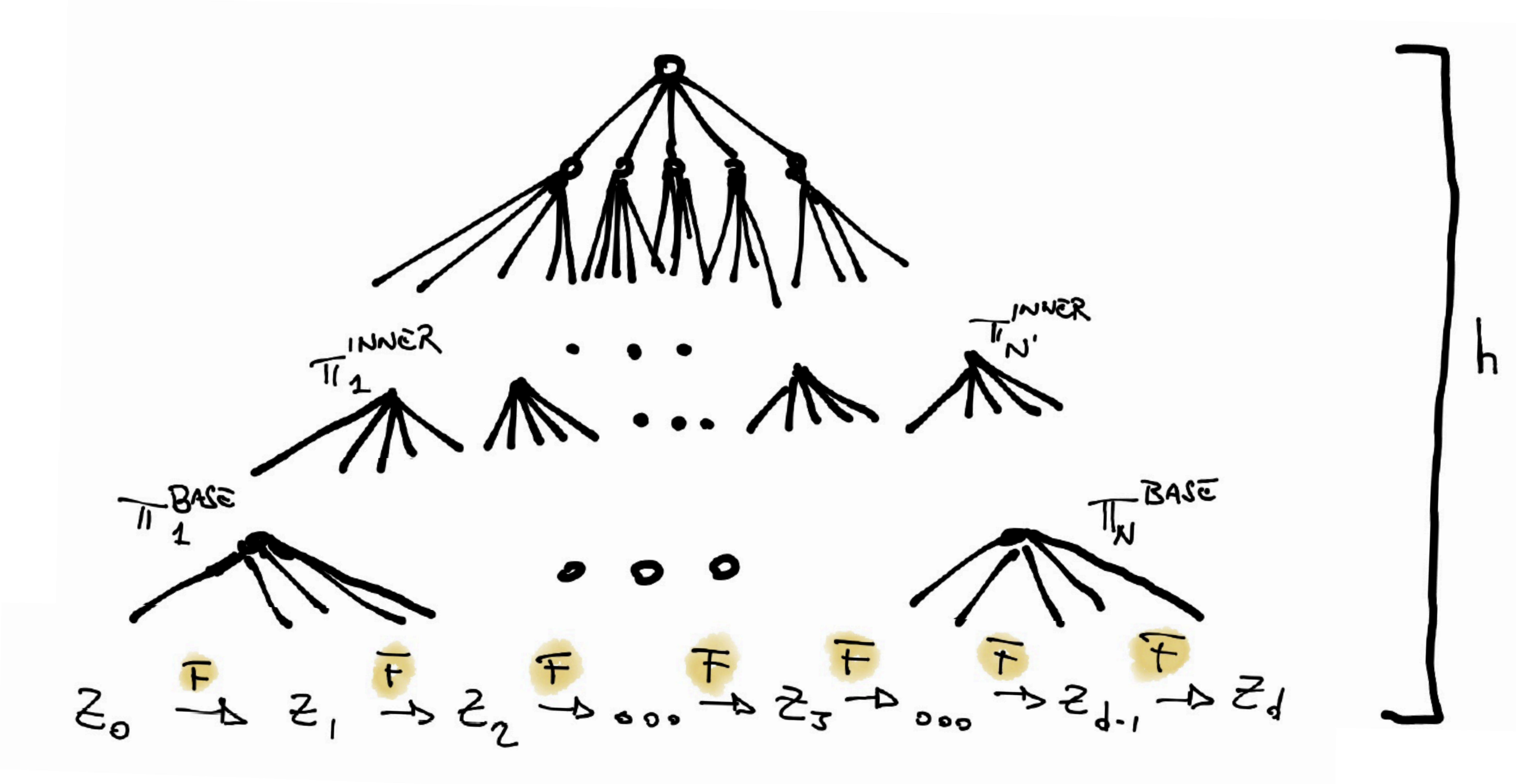
December 28, 2012

Mangrove: A Scalable Framework for Folding-based SNARKs

Wilson Nguyen Trisha Datta Binyi Chen Nirvan Tyagi Dan Boneh

{wdnguyen, tcdatta, binyi, tyagi, dabo}@cs.stanford.edu
Stanford University

Idea: proceed in a tree fashion rather than along a path.



We extract h times. We want h to be O(1)

The “Tree Approach”

A canonical way to go around the problem we just saw (via extractability)

Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data

Nir Bitansky*
nirbitan@tau.ac.il
Tel Aviv University

Ran Canetti*
canetti@tau.ac.il
Boston University and Tel Aviv University

Alessandro Chiesa
alexch@csail.mit.edu
MIT

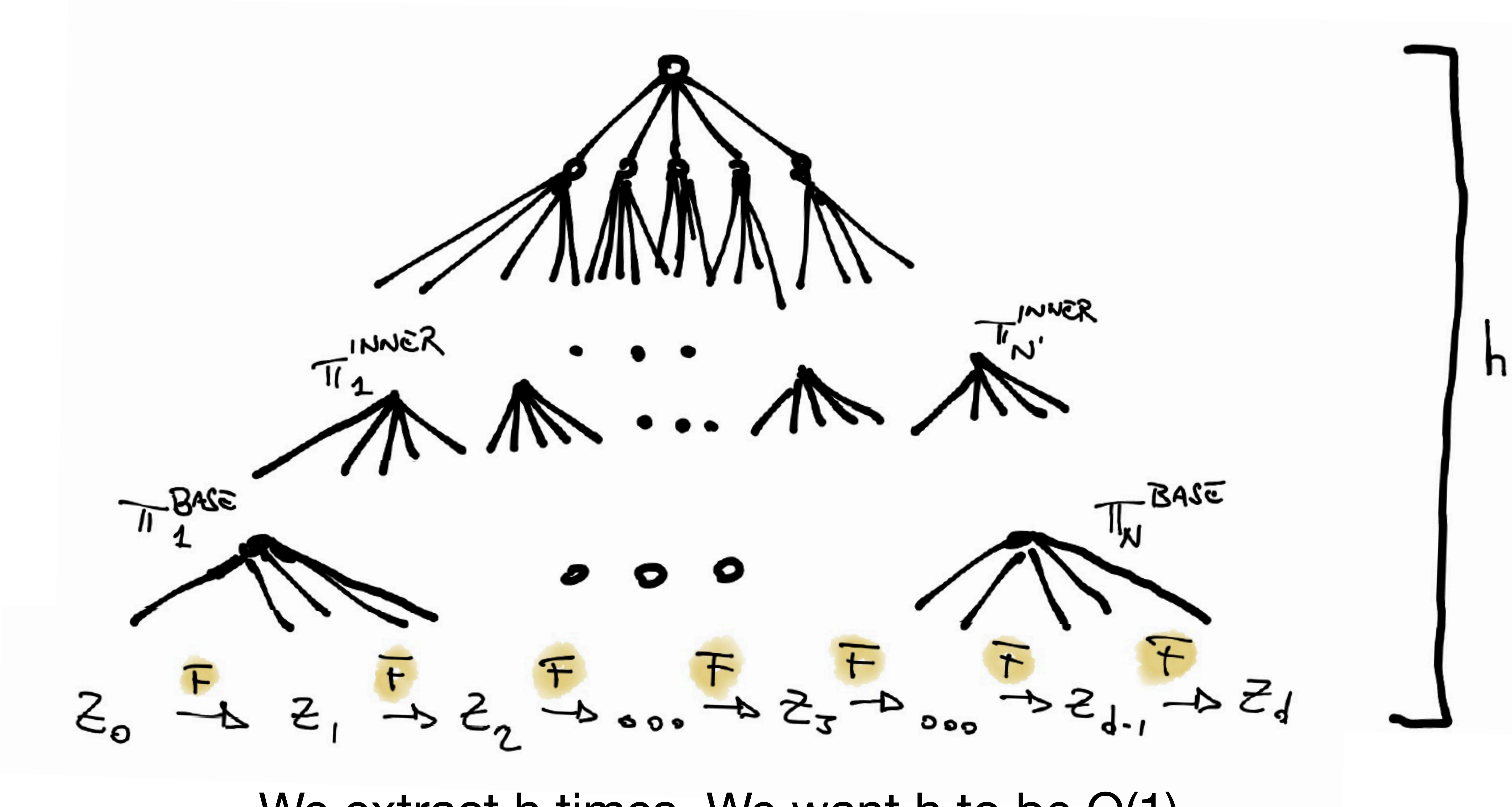
Eran Tromer†
tromer@tau.ac.il
Tel Aviv University

December 28, 2012

Mangrove: A Scalable Framework for Folding-based SNARKs

Wilson Nguyen Trisha Datta Binyi Chen Nirvan Tyagi Dan Boneh

{wdnguyen, tcdatta, binyi, tyagi, dabo}@cs.stanford.edu
Stanford University



Idea: proceed in a tree fashion rather than along a path.

We extract h times. We want h to be O(1)
For that, choose branching factor O(λ).

The “Tree Approach”

A canonical way to go around the problem we just saw (via extractability)

Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data

Nir Bitansky*
nirbitan@tau.ac.il
Tel Aviv University

Ran Canetti*
canetti@tau.ac.il
Boston University and Tel Aviv University

Alessandro Chiesa
alexch@csail.mit.edu
MIT

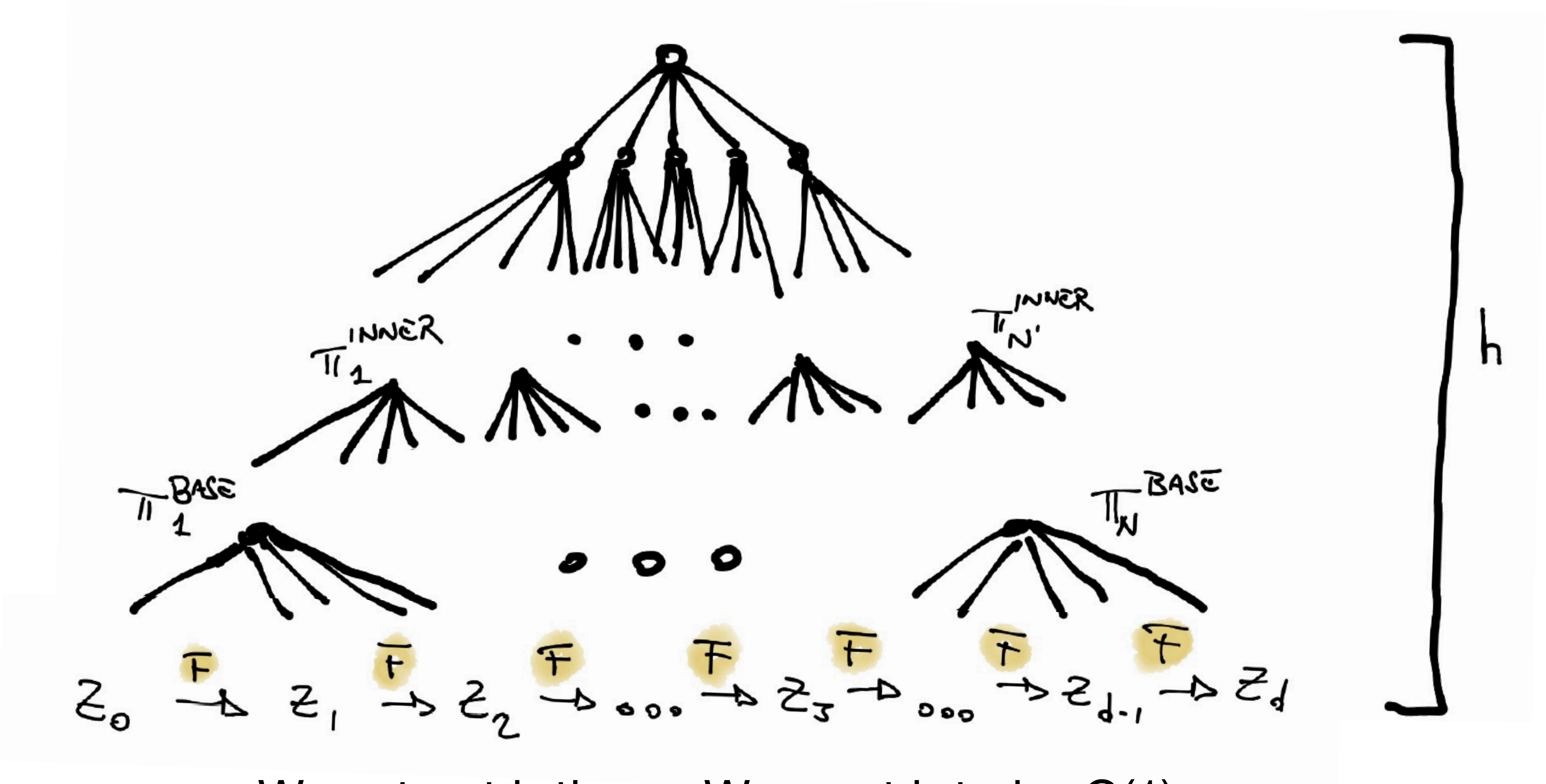
Eran Tromer†
tromer@tau.ac.il
Tel Aviv University

December 28, 2012

Mangrove: A Scalable Framework for Folding-based SNARKs

Wilson Nguyen Trisha Datta Binyi Chen Nirvan Tyagi Dan Boneh
{wdnguyen, tcdatta, binyi, tyagi, dabo}@cs.stanford.edu
Stanford University

This construction works but it is not the plain recursive construction from before anymore.



We extract h times. We want h to be O(1)
For that, choose branching factor O(λ).

Idea: proceed in a tree fashion rather than along a path.

Why Does Security Beyond O(1) Depth Matters?

A digression on motivation

Why Does Security Beyond O(1) Depth Matters?

A digression on motivation

- Cryptographic settings

Why Does Security Beyond O(1) Depth Matters?

A digression on motivation

- Cryptographic settings
  - VDFs

Why Does Security Beyond O(1) Depth Matters?

A digression on motivation

- Cryptographic settings
  - VDFs
    - requires ω(1) iterations

Why Does Security Beyond O(1) Depth Matters?

A digression on motivation

- Cryptographic settings
  - VDFs
    - requires $\omega(1)$ iterations
  - Hashing and other symmetric key primitives

Why Does Security Beyond O(1) Depth Matters?

A digression on motivation

- Cryptographic settings
  - VDFs
    - requires $\omega(1)$ iterations
  - Hashing and other symmetric key primitives
    - round functions require $\approx \lambda$ iterations

Why Does Security Beyond O(1) Depth Matters?

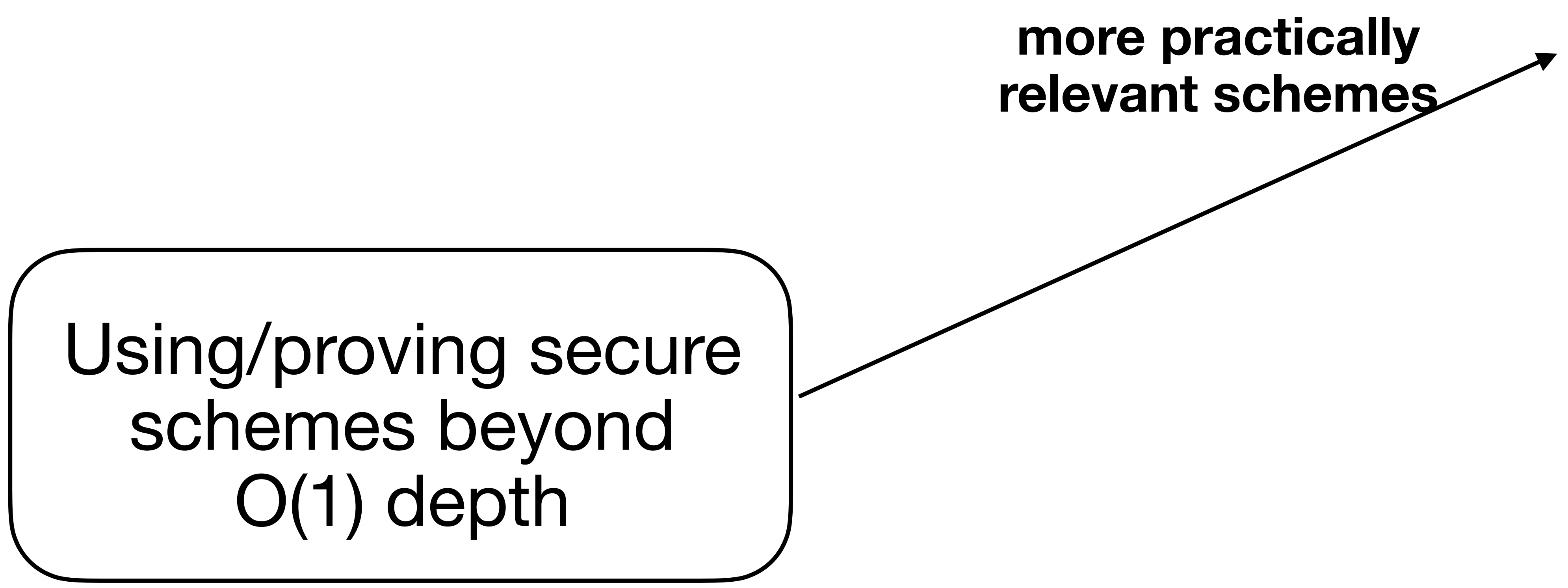
A digression on motivation

- Cryptographic settings
  - VDFs
    - requires $\omega(1)$ iterations
  - Hashing and other symmetric key primitives
    - round functions require $\approx \lambda$ iterations
- General improved understanding of *where* we can use *which* constructions

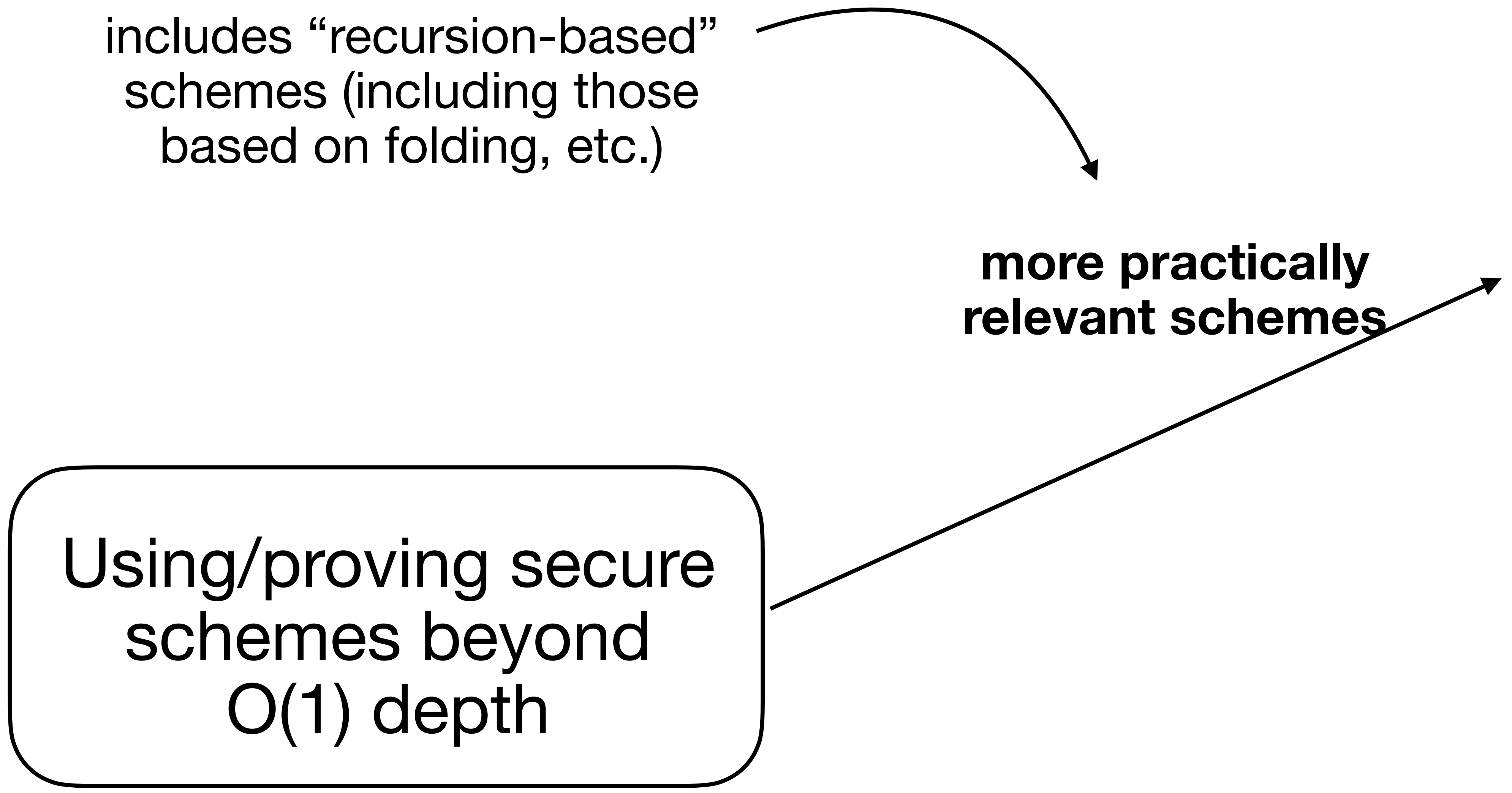
How the Community Has Addressed This—A Landscape

Using/proving secure schemes beyond O(1) depth

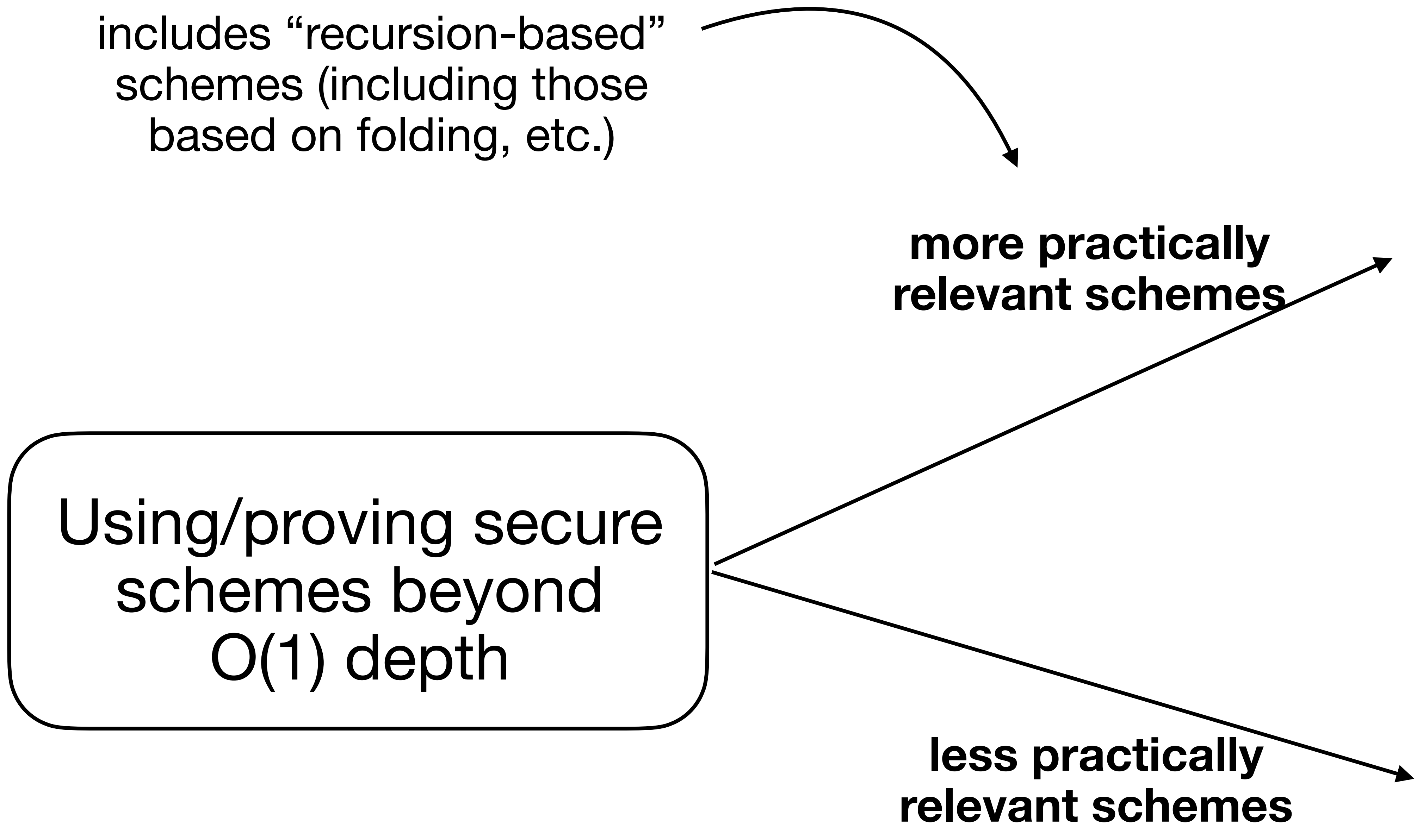
How the Community Has Addressed This—A Landscape



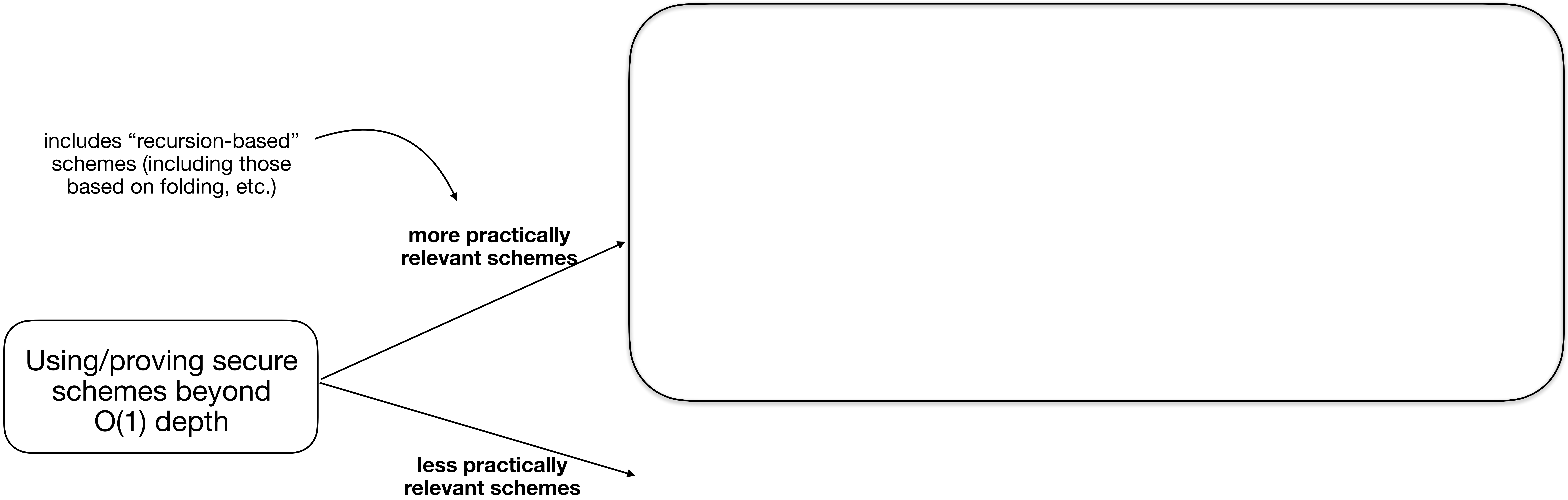
How the Community Has Addressed This—A Landscape



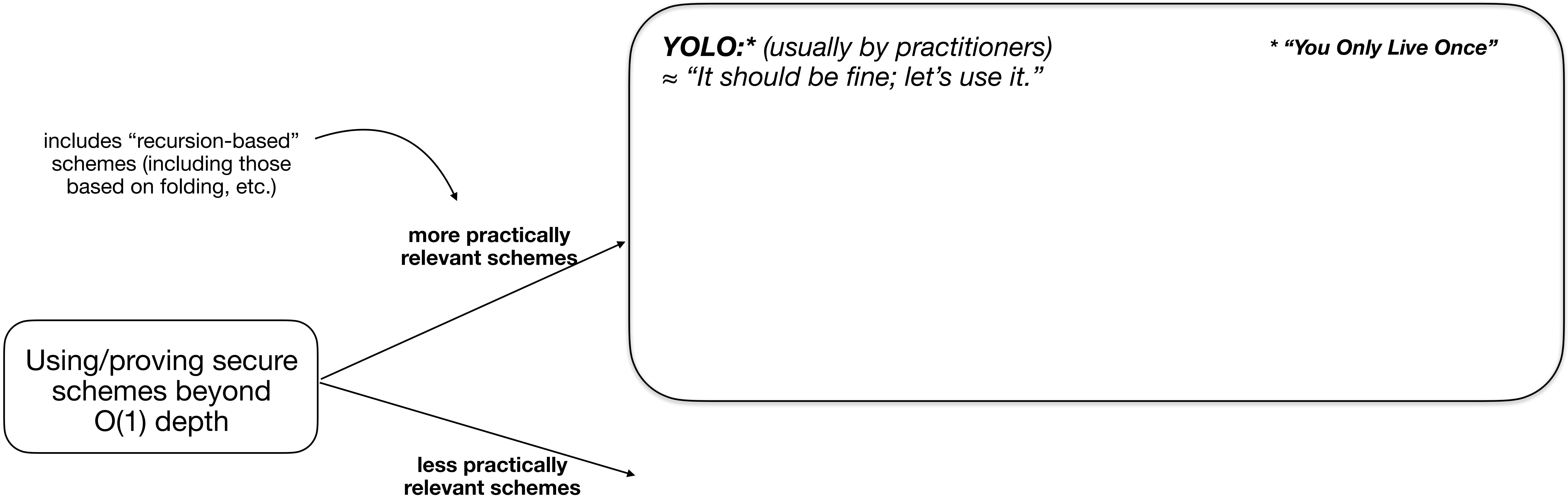
How the Community Has Addressed This—A Landscape



How the Community Has Addressed This—A Landscape



How the Community Has Addressed This—A Landscape



How the Community Has Addressed This—A Landscape

Using/proving secure schemes beyond O(1) depth

includes “recursion-based” schemes (including those based on folding, etc.)

more practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ *“It should be fine; let’s use it.”*

***** *“You Only Live Once”***

Heuristic assumptions on extractor size/time:

less practically relevant schemes

How the Community Has Addressed This—A Landscape

Using/proving secure schemes beyond O(1) depth

includes "recursion-based" schemes (including those based on folding, etc.)

more practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ *"It should be fine; let's use it."*

***** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

less practically relevant schemes

How the Community Has Addressed This—A Landscape

Using/proving secure schemes beyond O(1) depth

includes "recursion-based" schemes (including those based on folding, etc.)

more practically relevant schemes

less practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ "It should be fine; let's use it."

*** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

How the Community Has Addressed This—A Landscape

includes "recursion-based" schemes (including those based on folding, etc.)

Using/proving secure schemes beyond O(1) depth

more practically relevant schemes

less practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ "It should be fine; let's use it."

**** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa
alessandro.chiesa@epfl.ch
EPFL

Ziyi Guan
ziyi.guan@epfl.ch
EPFL

Shahar Samocha
shahars@starkware.co
StarkWare

Eylon Yogev
eylon.yogev@biu.ac.il
Bar-Ilan University

How the Community Has Addressed This—A Landscape

includes "recursion-based" schemes (including those based on folding, etc.)

Using/proving secure schemes beyond O(1) depth

more practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ "It should be fine; let's use it."

**** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa alessandro.chiesa@epfl.ch EPFL
Ziyi Guan ziyi.guan@epfl.ch EPFL
Shahar Samocha
Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

less practically relevant schemes

How the Community Has Addressed This—A Landscape

Using/proving secure schemes beyond O(1) depth

includes "recursion-based" schemes (including those based on folding, etc.)

more practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ "It should be fine; let's use it."

*** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa, alessandro.chiesa@epfl.ch, EPFL
Ziyi Guan, ziyi.guan@epfl.ch, EPFL
Shahar Samocha
Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

"Tree-approach"
[Mangrove (CRYPTO24),…]

less practically relevant schemes

How the Community Has Addressed This—A Landscape

includes "recursion-based" schemes (including those based on folding, etc.)

Using/proving secure schemes beyond O(1) depth

more practically relevant schemes

YOLO: (usually by practitioners)*
≈ "It should be fine; let's use it."

**** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa, alessandro.chiesa@epfl.ch, EPFL
Ziyi Guan, ziyi.guan@epfl.ch, EPFL
Shahar Samocha
Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

"Tree-approach"
[Mangrove (CRYPTO24),…]

less practically relevant schemes

How the Community Has Addressed This—A Landscape

Using/proving secure schemes beyond O(1) depth

includes "recursion-based" schemes (including those based on folding, etc.)

more practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ "It should be fine; let's use it."

***** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa, alessandro.chiesa@epfl.ch, EPFL
Ziyi Guan, ziyi.guan@epfl.ch, EPFL
Shahar Samocha
Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

"Tree-approach"
[Mangrove (CRYPTO24),…]

less practically relevant schemes

Soundness for deterministic F from batch arguments

How the Community Has Addressed This—A Landscape

Using/proving secure schemes beyond O(1) depth

includes "recursion-based" schemes (including those based on folding, etc.)

more practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ "It should be fine; let's use it."

***** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

... in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up ...

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa, alessandro.chiesa@epfl.ch, EPFL
Ziyi Guan, ziyi.guan@epfl.ch, EPFL
Shahar Samocha
Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

"Tree-approach"
[Mangrove (CRYPTO24),…]

less practically relevant schemes

Soundness for deterministic F from batch arguments

Rate-1 Non-Interactive Arguments for Batch-NP and Applications *

Lalita Devadas, MIT
Rishab Goyal, University of Wisconsin-Madison
Yael Kalai, Microsoft Research and MIT
Vinod Vaikuntanathan, MIT

How the Community Has Addressed This—A Landscape

includes "recursion-based" schemes (including those based on folding, etc.)

Using/proving secure schemes beyond O(1) depth

more practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ "It should be fine; let's use it."

*** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa, alessandro.chiesa@epfl.ch, EPFL
Ziyi Guan, ziyi.guan@epfl.ch, EPFL
Shahar Samocha
Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

"Tree-approach"
[Mangrove (CRYPTO24),…]

less practically relevant schemes

Soundness for deterministic F from batch arguments

Rate-1 Non-Interactive Arguments for Batch-NP and Applications *

Lalita Devadas, MIT
Rishab Goyal, University of Wisconsin-Madison
Yael Kalai, Microsoft Research and M
Vinod Vaikuntanathan, MIT

Incrementally Verifiable Computation via Rate-1 Batch Arguments

Omer Paneth*[1] and Rafael Pass†[1,2]

How the Community Has Addressed This—A Landscape

Using/proving secure schemes beyond O(1) depth

includes "recursion-based" schemes (including those based on folding, etc.)

more practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ "It should be fine; let's use it."

***** *"You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa, alessandro.chiesa@epfl.ch, EPFL
Ziyi Guan, ziyi.guan@epfl.ch, EPFL
Shahar Samocha
Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

"Tree-approach"
[Mangrove (CRYPTO24),…]

less practically relevant schemes

Soundness for deterministic F from batch arguments

Rate-1 Non-Interactive Arguments for Batch-NP and Applications *

Lalita Devadas, MIT
Rishab Goyal, University of Wisconsin-Madison
Yael Kalai, Microsoft Research and MI
Vinod Vaikuntanathan, MIT

Incrementally Verifiable Computation via Rate-1 Batch Arguments

Omer Paneth*[1] and Rafael Pass†

Verifiable Streaming Computation and Step-by-Step Zero-Knowledge

Abtin Afshar, Rishab Goyal*

How the Community Has Addressed This—A Landscape

includes "recursion-based" schemes (including those based on folding, etc.)

Using/proving secure schemes beyond O(1) depth

more practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ "It should be fine; let's use it."

*** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa, alessandro.chiesa@epfl.ch, EPFL
Ziyi Guan, ziyi.guan@epfl.ch, EPFL
Shahar Samocha
Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

"Tree-approach"
[Mangrove (CRYPTO24),…]

less practically relevant schemes

Soundness for deterministic F from batch arguments

Rate-1 Non-Interactive Arguments for Batch-NP and Applications *

Lalita Devadas, MIT
Rishab Goyal, University of Wisconsin-Madison
Yael Kalai, Microsoft Research and MI
Vinod Vaikuntanathan, MIT

Incrementally Verifiable Computation via Rate-1 Batch Arguments

Omer Paneth*[1] and Rafael Pass†

Verifiable Streaming Computation and Step-by-Step Zero-Knowledge

Abtin Afshar, Rishab Goyal*

Soundness for non-deterministic computations

Incrementally Verifiable Computation for NP from Standard Assumptions

Pratish Datta*, NTT Research
Abhishek Jain†, NTT Research and JHU
Zhengzhong Jin‡, Northeastern
Alexis Korb§, UCLA
Surya Mathialagan¶, MIT
Amit Sahai‖, UCLA

How the Community Has Addressed This—A Landscape

Using/proving secure schemes beyond O(1) depth

more practically relevant schemes

includes "recursion-based" schemes (including those based on folding, etc.)

YOLO:* *(usually by practitioners)*
≈ "It should be fine; let's use it."

*** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa alessandro.chiesa@epfl.ch EPFL
Ziyi Guan ziyi.guan@epfl.ch EPFL
Shahar Samocha
Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

"Tree-approach"
[Mangrove (CRYPTO24),…]

less practically relevant schemes

Soundness for deterministic F from batch arguments

Rate-1 Non-Interactive Arguments for Batch-NP and Applications *

Lalita Devadas MIT
Rishab Goyal University of Wisconsin-Madison
Yael Kalai Microsoft Research and MI
Vinod Vaikuntanathan MIT

Incrementally Verifiable Computation via Rate-1 Batch Arguments

Omer Paneth*[1] and Rafael Pass†

Verifiable Streaming Computation and Step-by-Step Zero-Knowledge

Abtin Afshar, Rishab Goyal*

Soundness for non-deterministic computations

Incrementally Verifiable Computation for NP from Standard Assumptions

Pratish Datta* NTT Research
Abhishek Jain† NTT Research and JHU
Zhengzhong Jin‡ Northeastern
Alexis Korb§ UCLA
Surya Mathialagan¶ MIT
Amit Sahai‖ UCLA

Limitations

How the Community Has Addressed This—A Landscape

Using/proving secure schemes beyond O(1) depth

includes "recursion-based" schemes (including those based on folding, etc.)

more practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ "It should be fine; let's use it."

*** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa, alessandro.chiesa@epfl.ch, EPFL
Ziyi Guan, ziyi.guan@epfl.ch, EPFL
Shahar Samocha
Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

"Tree-approach"
[Mangrove (CRYPTO24),…]

less practically relevant schemes

Soundness for deterministic F from batch arguments

Rate-1 Non-Interactive Arguments for Batch-NP and Applications *

Lalita Devadas, MIT
Rishab Goyal, University of Wisconsin-Madison
Yael Kalai, Microsoft Research and MIT
Vinod Vaikuntanathan, MIT

Incrementally Verifiable Computation via Rate-1 Batch Arguments

Omer Paneth*[1] and Rafael Pass†

Verifiable Streaming Computation and Step-by-Step Zero-Knowledge

Abtin Afshar, Rishab Goyal*

Soundness for non-deterministic computations

Incrementally Verifiable Computation for NP from Standard Assumptions

Pratish Datta*, NTT Research
Abhishek Jain†, NTT Research and JHU
Zhengzhong Jin‡, Northeastern
Alexis Korb§, UCLA
Surya Mathialagan¶, MIT
Amit Sahai‖, UCLA

Limitations

Often it might not be warranted.

How the Community Has Addressed This—A Landscape

Using/proving secure schemes beyond O(1) depth

includes “recursion-based” schemes (including those based on folding, etc.)

more practically relevant schemes

YOLO:* *(usually by practitioners)*
≈ “It should be fine; let’s use it.”

*** “You Only Live Once”***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa alessandro.chiesa@epfl.ch EPFL
Ziyi Guan ziyi.guan@epfl.ch EPFL
Shahar Samocha
Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

“Tree-approach”
[Mangrove (CRYPTO24),…]

less practically relevant schemes

Soundness for deterministic F from batch arguments

Rate-1 Non-Interactive Arguments for Batch-NP and Applications *

Lalita Devadas MIT
Rishab Goyal University of Wisconsin-Madison
Yael Kalai Microsoft Research and MI
Vinod Vaikuntanathan MIT

Incrementally Verifiable Computation via Rate-1 Batch Arguments

Omer Paneth*[1] and Rafael Pass†

Verifiable Streaming Computation and Step-by-Step Zero-Knowledge

Abtin Afshar, Rishab Goyal*

Soundness for non-deterministic computations

Incrementally Verifiable Computation for NP from Standard Assumptions

Pratish Datta* NTT Research
Abhishek Jain† NTT Research and JHU
Zhengzhong Jin‡ Northeastern
Alexis Korb§ UCLA
Surya Mathialagan¶ MIT
Amit Sahai‖ UCLA

Limitations

Often it might not be warranted.

Modifies schemes (or applicable only at times)

How the Community Has Addressed This—A Landscape

Using/proving secure schemes beyond O(1) depth

includes "recursion-based" schemes (including those based on folding, etc.)

more practically relevant schemes

YOLO: (usually by practitioners)*
≈ "It should be fine; let's use it."

**** "You Only Live Once"***

Heuristic assumptions on extractor size/time:

Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa, alessandro.chiesa@epfl.ch, EPFL
Ziyi Guan, ziyi.guan@epfl.ch, EPFL
Shahar Samocha
Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

"Tree-approach"
[Mangrove (CRYPTO24),…]

less practically relevant schemes

Soundness for deterministic F from batch arguments

Rate-1 Non-Interactive Arguments for Batch-NP and Applications *

Lalita Devadas, MIT
Rishab Goyal, University of Wisconsin-Madison
Yael Kalai, Microsoft Research and MIT
Vinod Vaikuntanathan, MIT

Incrementally Verifiable Computation via Rate-1 Batch Arguments

Omer Paneth*[1] and Rafael Pass[†]

Verifiable Streaming Computation and Step-by-Step Zero-Knowledge

Aditm Afshar, Rishab Goyal*

Soundness for non-deterministic computations

Incrementally Verifiable Computation for NP from Standard Assumptions

Pratish Datta*, NTT Research
Abhishek Jain[†], NTT Research and JHU
Zhengzhong Jin[‡], Northeastern
Alexis Korb[§], UCLA
Surya Mathialagan[¶], MIT
Amit Sahai[‖], UCLA

Limitations

Often it might not be warranted.

Modifies schemes (or applicable only at times)

More complex, more inefficient

How the Community Has Addressed This—A Landscape

YOLO:* (usually by practitioners)

* "You Only Live Once"

Still open:

When are existing constructions secure/insecure beyond O(1) depth?

includes "recursion-based" schemes (including those based on folding, etc.)

Using/proving secure schemes beyond O(1) depth

relevant schemes

Straight-Line Extraction

Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa, alessandro.chiesa@epfl.ch, EPFL — Ziyi Guan, ziyi.guan@epfl.ch, EPFL — Shahar Samocha — Eylon Yogev

On Composing AGM-Secure Functionalities with Cryptographic Proofs
Applications to Unbounded-Depth IVC and More*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

"Tree-approach"
[Mangrove (CRYPTO24),…]

less practically relevant schemes

Soundness for deterministic F from batch arguments

Rate-1 Non-Interactive Arguments for Batch-NP and Applications *

Lalita Devadas, MIT — Rishab Goyal, University of Wisconsin-Madison — Yael Kalai, Microsoft Research and MIT — Vinod Vaikuntanathan, MIT

Incrementally Verifiable Computation via Rate-1 Batch Arguments

Omer Paneth*[1] and Rafael Pass†

Verifiable Streaming Computation and Step-by-Step Zero-Knowledge

Aarushi Afshar, Rishab Goyal*

Soundness for non-deterministic computations

Incrementally Verifiable Computation for NP from Standard Assumptions

Pratish Datta*, NTT Research — Abhishek Jain†, NTT Research and JHU — Zhengzhong Jin‡, Northeastern — Alexis Korb§, UCLA — Surya Mathialagan¶, MIT — Amit Sahai‖, UCLA

Limitations

Often it might not be warranted.

Modifies schemes (or applicable only at times)

More complex, more inefficient

This Work's Question

Still open:

When are existing constructions secure/insecure beyond O(1) depth?

The problem at hand

This Work's Question

Still open:
When are existing constructions secure/insecure beyond O(1) depth?

The problem at hand

When is any construction secure/insecure beyond O(1) depth?

This Work's Question

Still open:

When are <u>existing</u> constructions secure/insecure <u>beyond O(1) depth</u>?

The problem at hand

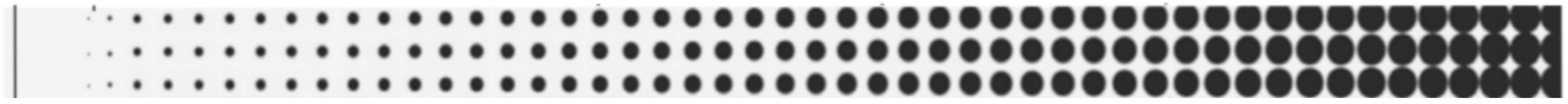
When is <u>any</u> construction secure/insecure <u>beyond O(1) depth</u>?

We approach this question through two main conceptual lenses.

Lens 1: “*Depth*” as a Core Object of Study

Lens 1: “*Depth*” as a Core Object of Study

??



O(1) O($\log^2(\lambda)$) O($\sqrt{\lambda}$) O(λ)

The asymptotic depth “line”.

Lens 1: “*Depth*” as a Core Object of Study

The asymptotic depth “line”.

Lens 1: “*Depth*” as a Core Object of Study

The asymptotic depth “line”.

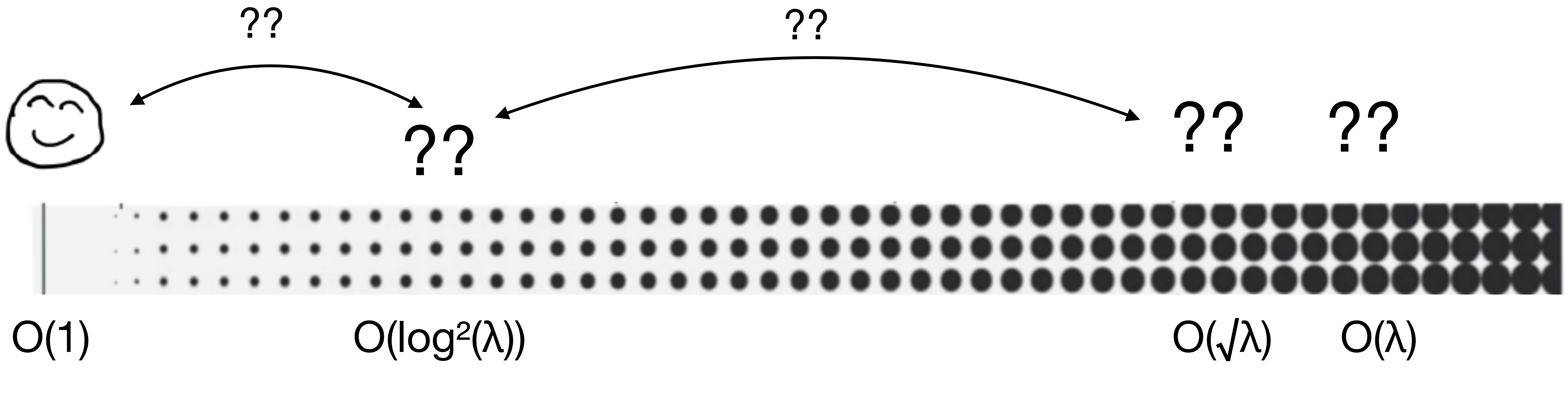
Lens 1: “*Depth*” as a Core Object of Study

The asymptotic depth “line”.

Lens 1: “*Depth*” as a Core Object of Study

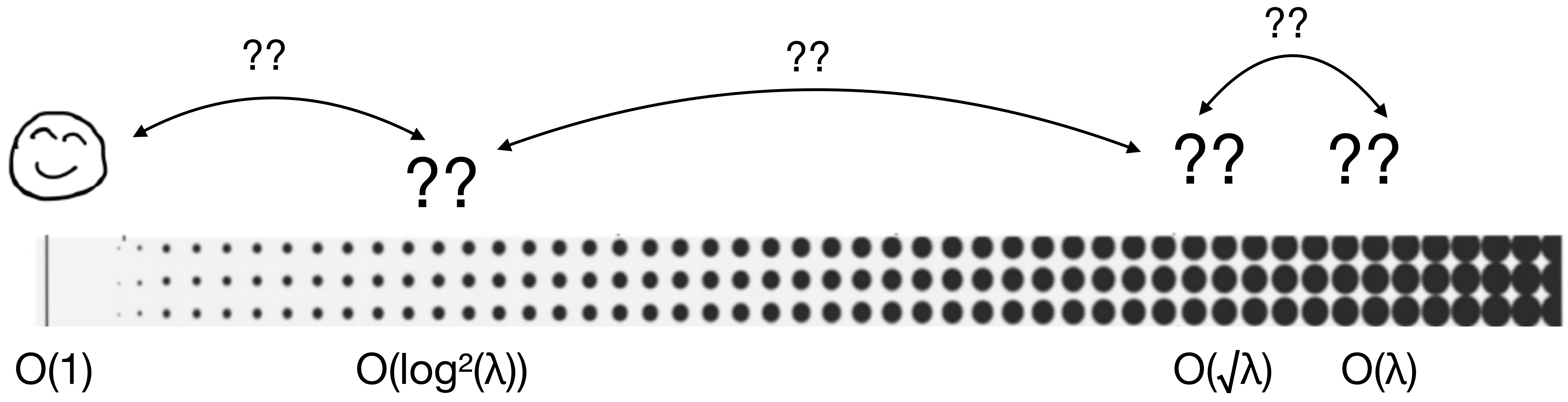
The asymptotic depth “line”.

Lens 1: “*Depth*” as a Core Object of Study



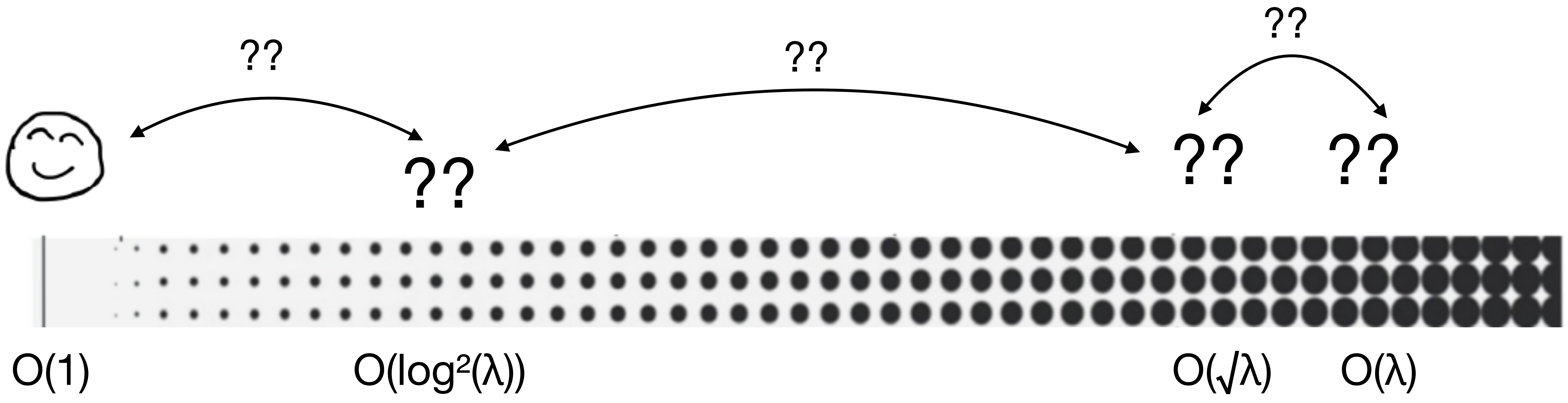
The asymptotic depth “line”.

Lens 1: “*Depth*” as a Core Object of Study



The asymptotic depth “line”.

Lens 1: "*Depth*" as a Core Object of Study



The asymptotic depth "line".

A note on abuse of language:
I will say
"*big/bigger*" to mean "*fast/er growing*";
"*small/smaller*" to mean "*slow/er growing*"

Lens 2: Keeping Extractable Security in the Background

(And having in the foreground the relation [adversarial advantage] ↔ [depth])

Lens 2: Keeping Extractable Security in the Background

(And having in the foreground the relation [adversarial advantage] ↔ [depth])

- The proof strategy that usually fails:

Lens 2: Keeping Extractable Security in the Background

(And having in the foreground the relation [adversarial advantage] $\leftrightarrow$ [depth])

- The proof strategy that usually fails:
  - In order to show extractability, **show** an extractor that succeeds **in polynomial time**.

Lens 2: Keeping Extractable Security in the Background

(And having in the foreground the relation [adversarial advantage] ↔ [depth])

- The proof strategy that usually fails:
  - In order to show extractability, **show** an extractor that succeeds **in polynomial time**.
- *Constructing* such a machine is hard.

Lens 2: Keeping Extractable Security in the Background

(And having in the foreground the relation [adversarial advantage] ↔ [depth])

- The proof strategy that usually fails:
  - In order to show extractability, **show** an extractor that succeeds **in polynomial time**.
- *Constructing* such a machine is hard.
- ⇒ We *mostly* look for techniques that avoid extractability and look at the advantage of the adversary in:

Lens 2: Keeping Extractable Security in the Background

(And having in the foreground the relation [adversarial advantage] ↔ [depth])

- The proof strategy that usually fails:
  - In order to show extractability, **show** an extractor that succeeds **in polynomial time**.
- *Constructing* such a machine is hard.
- ⇒ We *mostly* look for techniques that avoid extractability and look at the advantage of the adversary in:
  - soundness for deterministic computations (Adv. succeeds ⇒ $z_d = \underbrace{F(F(\ldots F(F(z_0))\ldots))}_{d \text{ times}}$)

Lens 2: Keeping Extractable Security in the Background

(And having in the foreground the relation [adversarial advantage] ↔ [depth])

- The proof strategy that usually fails:
  - In order to show extractability, **show** an extractor that succeeds **in polynomial time**.
- *Constructing* such a machine is hard.
- ⇒ We *mostly* look for techniques that avoid extractability and look at the advantage of the adversary in:
  - soundness for deterministic computations (Adv. succeeds ⇒ $z_d = \underbrace{F(F(\ldots F(F(z_0))\ldots))}_{d \text{ times}}$)
  - soundness for non-deterministic computations (Adv. succeeds ⇒ $\exists w_0, \ldots, w_{d-1} : z_d = \underbrace{F(F(\ldots F(F(z_0, w_0))\ldots), w_{d-1})}_{d \text{ times}}$)

Lens 2: Keeping Extractable Security in the Background

(And having in the foreground the relation [adversarial advantage] ↔ [depth])

- The proof strategy that usually fails:
  - In order to show extractability, **show** an extractor that succeeds **in polynomial time**.
- *Constructing* such a machine is hard.
- ⇒ We *mostly* look for techniques that avoid extractability and look at the advantage of the adversary in:
  - soundness for deterministic computations (Adv. succeeds ⇒ $z_d = \underbrace{F(F(\ldots F(F(z_0))\ldots))}_{d \text{ times}}$)
  - soundness for non-deterministic computations (Adv. succeeds ⇒ $\exists w_0, \ldots, w_{d-1} : z_d = \underbrace{F(F(\ldots F(F(z_0, w_0))\ldots), w_{d-1})}_{d \text{ times}}$)
  - another notion of (non-extractable) soundness that we introduce, but that is more expressive than the above.

Lens 2: Keeping Extractable Security in the Background

(And having in the foreground the relation [adversarial advantage] ↔ [depth])

- The proof strategy that usually fails:
  - In order to show extractability, **show** an extractor that succeeds **in polynomial time**.
- *Constructing* such a machine is hard.
- ⇒ We *mostly* look for techniques that avoid extractability and look at the advantage of the adversary in:
  - soundness for deterministic computations (Adv. succeeds ⇒ $z_d = \underbrace{F(F(\ldots F(F(z_0))\ldots))}_{d \text{ times}}$)
  - soundness for non-deterministic computations (Adv. succeeds ⇒ $\exists w_0, \ldots, w_{d-1} : z_d = \underbrace{F(F(\ldots F(F(z_0, w_0))\ldots), w_{d-1})}_{d \text{ times}}$)
  - another notion of (non-extractable) soundness that we introduce, but that is more expressive than the above.
    - ≈ incremental analogue of functional commitments

Our Results

Our Results

Our goals (distilled):

- finding **tools to prove security/insecurity** of any construction (including efficient existing ones).
- **studying IVC depth and its** relation to **security** as a subject in its own right.

Our Results

Our goals (distilled):

- finding **tools to prove security/insecurity** of any construction (including efficient existing ones).
- **studying IVC depth and its** relation to **security** as a subject in its own right.

First result*:

* NB: in the eprint, the results are presented in a different order.

“weak insecurity” at big depths ⇒ security at smaller depths.

Our Results

Our goals (distilled):

- finding **tools to prove security/insecurity** of any construction (including efficient existing ones).
- **studying IVC depth and its** relation to **security** as a subject in its own right.

First result*:

* NB: in the eprint, the results are presented in a different order.

“weak insecurity” at big depths ⇒ security at smaller depths.

Our Results

Our goals (distilled):

- finding **tools to prove security/insecurity** of any construction (including efficient existing ones).
- **studying IVC depth and its** relation to **security** as a subject in its own right.

First result*:

* NB: in the eprint, the results are presented in a different order.

"weak insecurity" at big depths ⇒ security at smaller depths.

Our Results

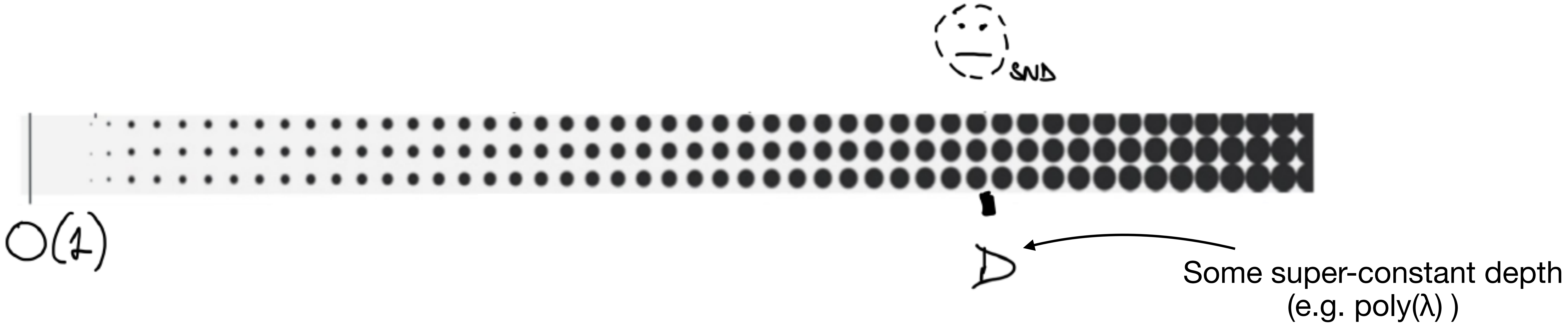
Our goals (distilled):

- finding **tools to prove security/insecurity** of any construction (including efficient existing ones).
- **studying IVC depth and its** relation to **security** as a subject in its own right.

First result*:

* NB: in the eprint, the results are presented in a different order.

“weak insecurity” at big depths ⇒ security at smaller depths.



Our Results

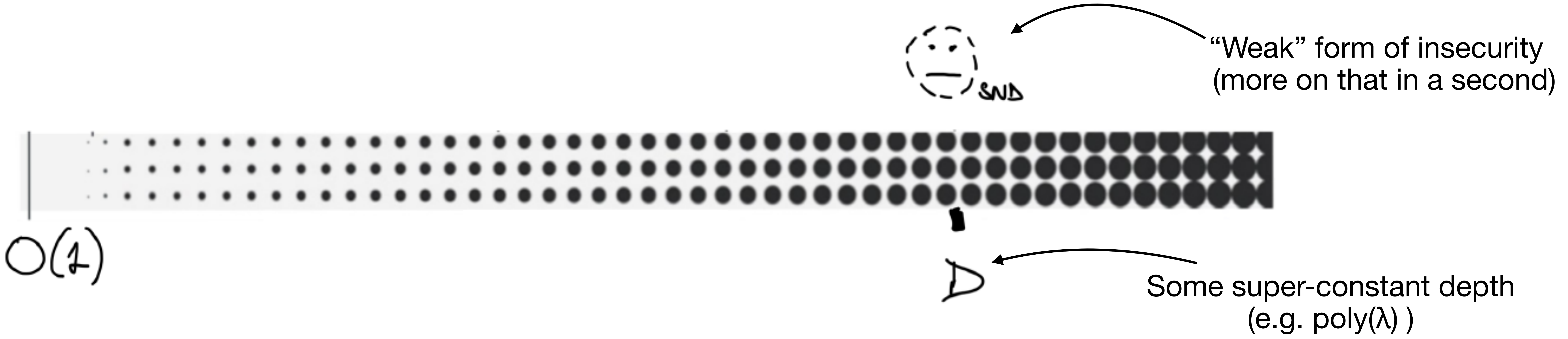
Our goals (distilled):

- finding **tools to prove security/insecurity** of any construction (including efficient existing ones).
- **studying IVC depth and its** relation to **security** as a subject in its own right.

First result*:

* NB: in the eprint, the results are presented in a different order.

“weak insecurity” at big depths ⇒ security at smaller depths.



Our Results

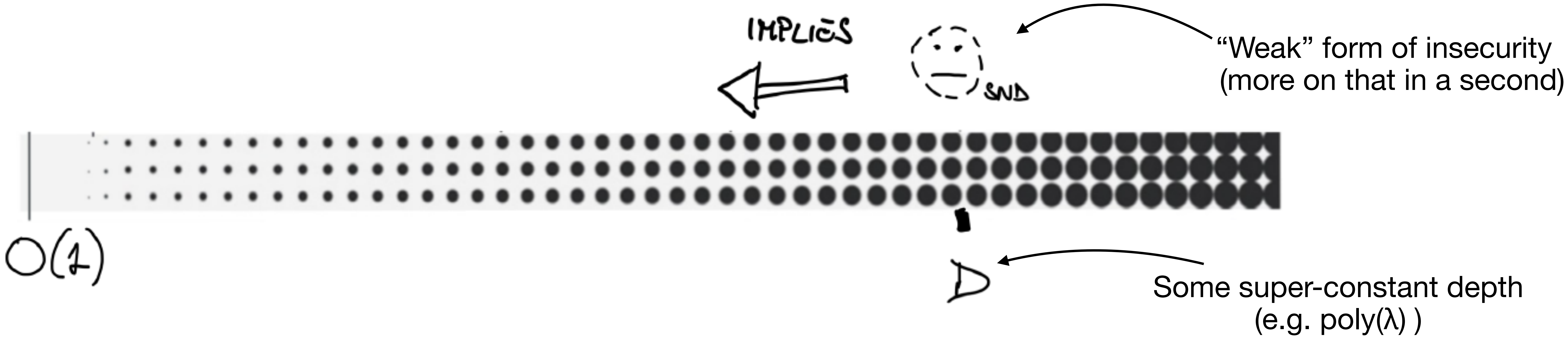
Our goals (distilled):

- finding **tools to prove security/insecurity** of any construction (including efficient existing ones).
- **studying IVC depth and its** relation to **security** as a subject in its own right.

First result*:

* NB: in the eprint, the results are presented in a different order.

“weak insecurity” at big depths ⇒ security at smaller depths.



Our Results

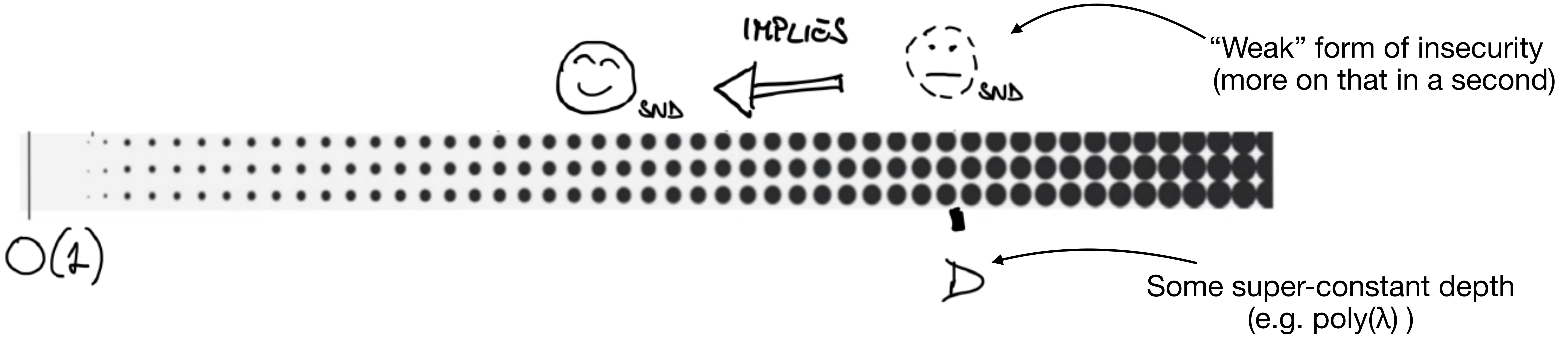
Our goals (distilled):

- finding **tools to prove security/insecurity** of any construction (including efficient existing ones).
- **studying IVC depth and its** relation to **security** as a subject in its own right.

First result*:

* NB: in the eprint, the results are presented in a different order.

“weak insecurity” at big depths ⇒ security at smaller depths.



Our Results

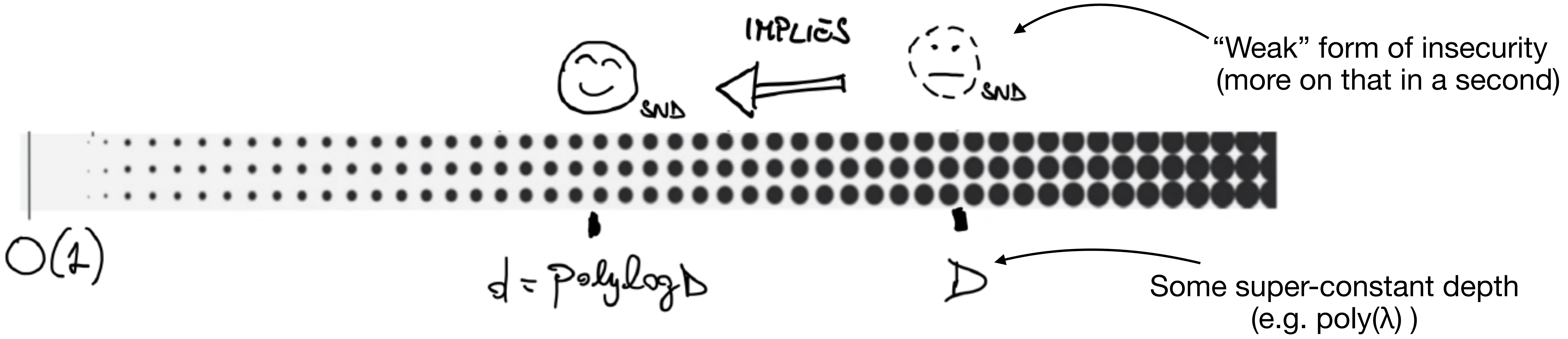
Our goals (distilled):

- finding **tools to prove security/insecurity** of any construction (including efficient existing ones).
- **studying IVC depth and its** relation to **security** as a subject in its own right.

First result*:

* NB: in the eprint, the results are presented in a different order.

"weak insecurity" at big depths ⇒ security at smaller depths.



Our Results

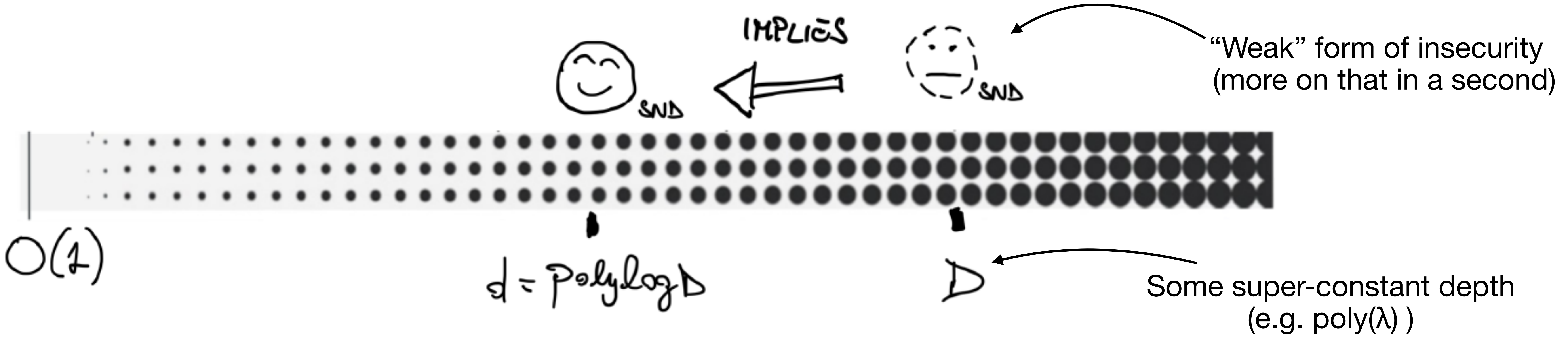
Our goals (distilled):

- finding **tools to prove security/insecurity** of any construction (including efficient existing ones).
- **studying IVC depth and its** relation to **security** as a subject in its own right.

First result*:

* NB: in the eprint, the results are presented in a different order.

“weak insecurity” at big depths ⇒ security at smaller depths.

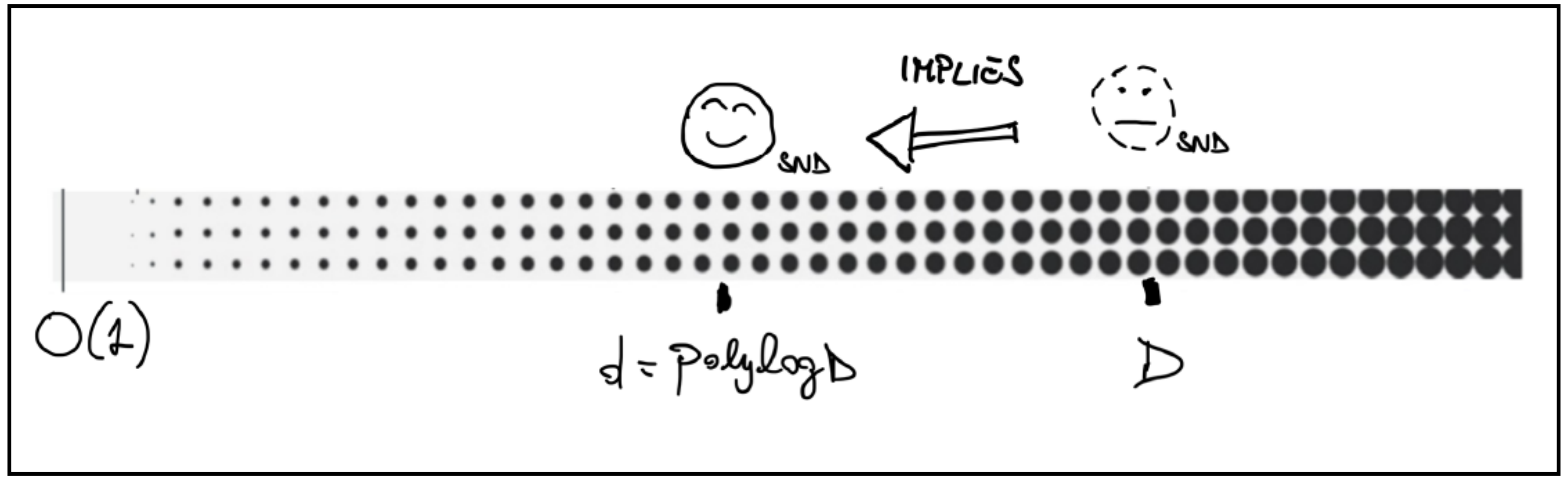


Implication:

to prove security at some ω(1) depth d,

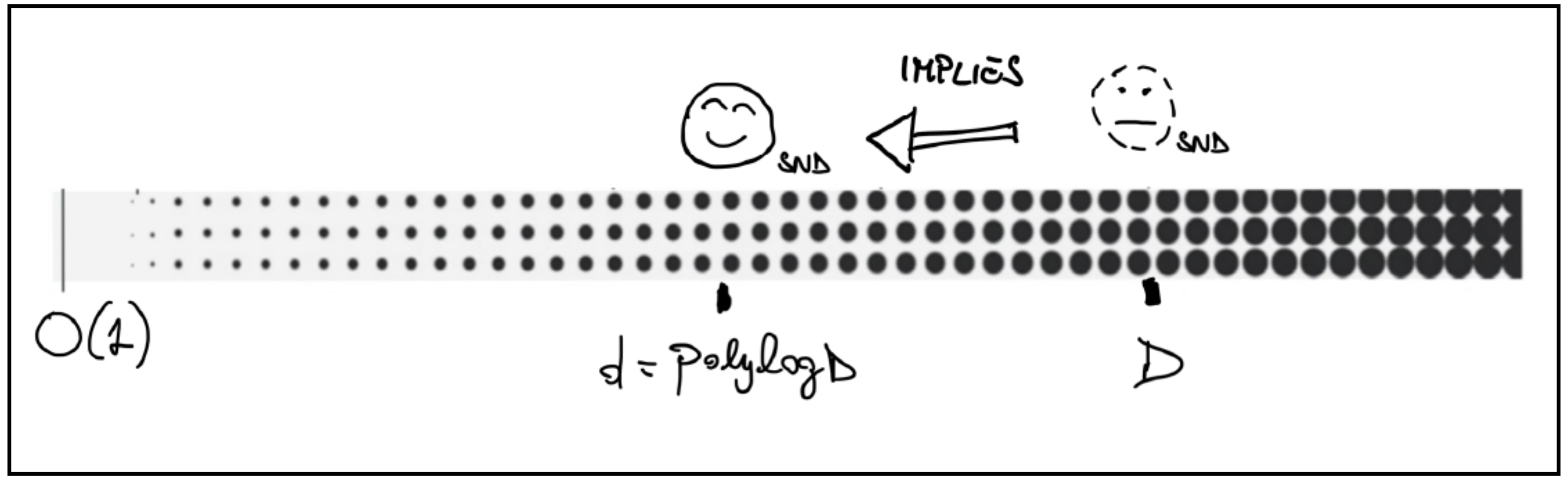
show some ω(1) depth D where this weak property holds.

What Do We Mean by "Weak" Insecurity?



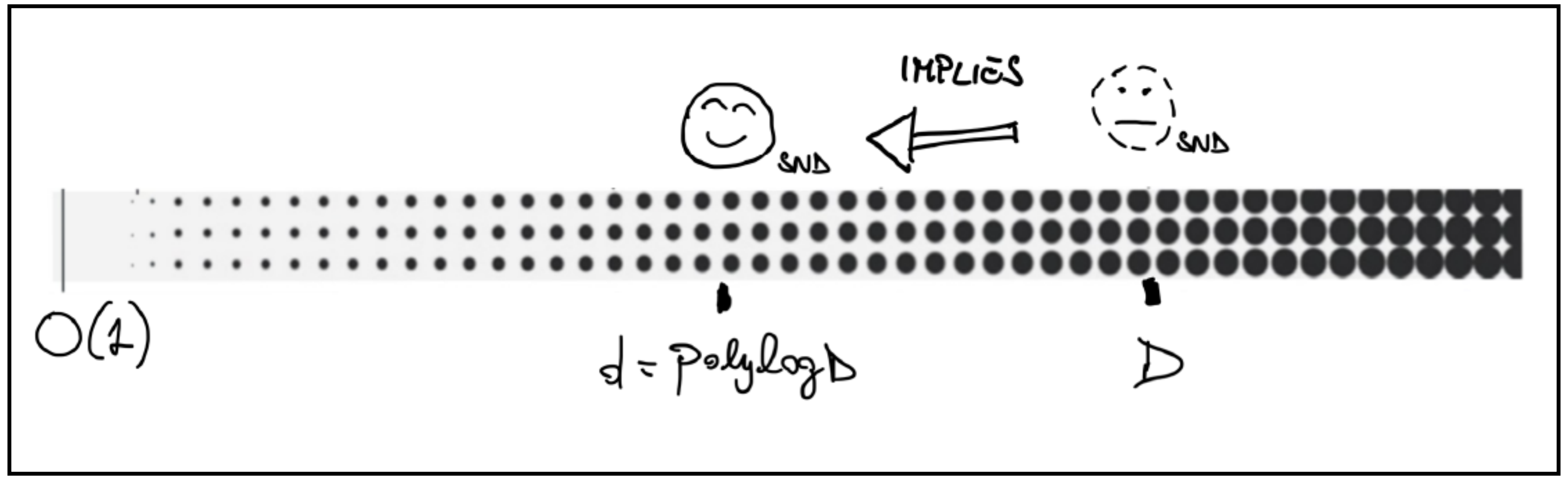
What Do We Mean by “Weak” Insecurity? 

:= ***infinitely-often*** **soundness (io-SND)**



What Do We Mean by “Weak” Insecurity? 

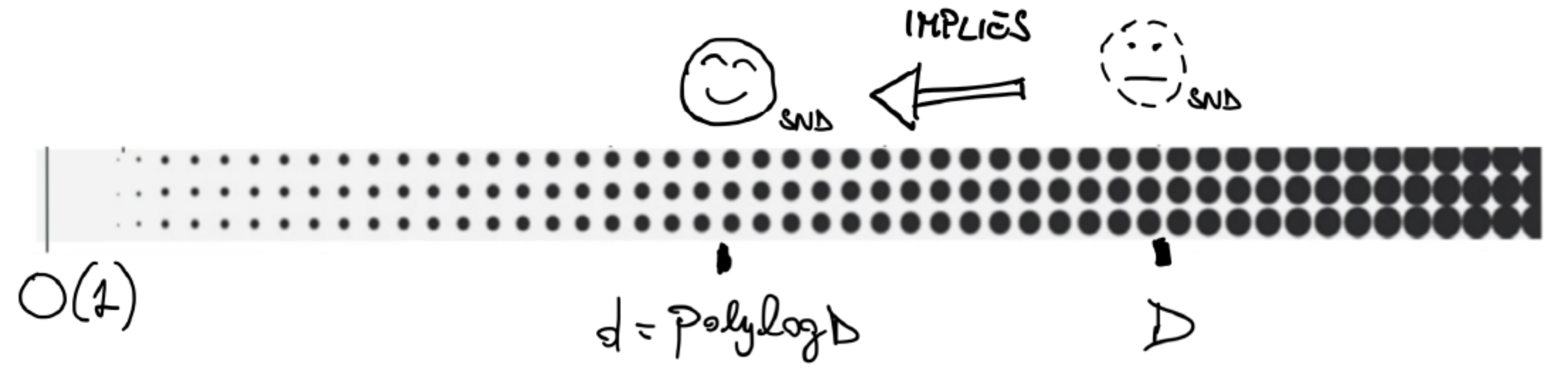
:= *infinitely-often* soundness (io-SND)



Traditional “almost-everywhere” soundness (ae-SND):

$$\forall \text{ PPT } \mathcal{A} \;\; \exists \varepsilon \in negl(\lambda) : \; ADV^{\mathcal{A}}_{SND}(\lambda) \leq \varepsilon(\lambda) \;\; \forall \lambda \in \mathbb{N}$$

What Do We Mean by “Weak” Insecurity? 



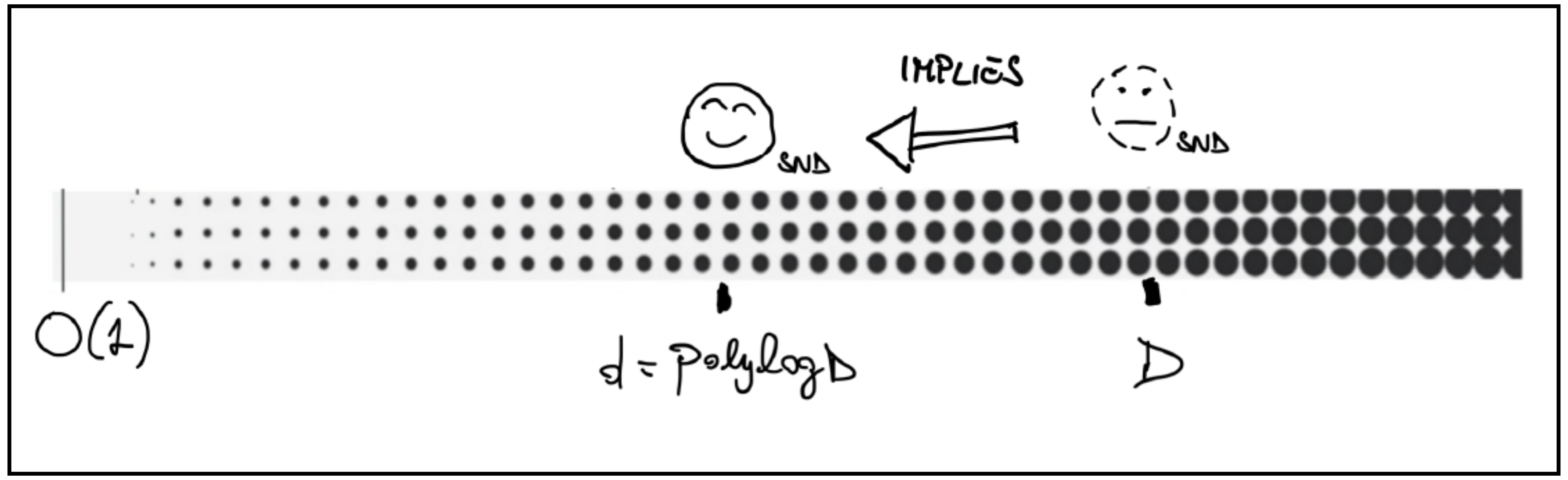
:= ***infinitely-often*** **soundness (io-SND)**

Traditional “almost-everywhere” soundness (ae-SND):

$$\forall \text{ PPT } \mathcal{A} \ \exists \varepsilon \in \text{negl}(\lambda) : \text{ADV}^{\mathcal{A}}_{\text{SND}}(\lambda) \leq \varepsilon(\lambda) \quad \forall \lambda \in \mathbb{N}$$

What Do We Mean by “Weak” Insecurity? 

:= ***infinitely-often*** **soundness (io-SND)**



Traditional “almost-everywhere” soundness (ae-SND):

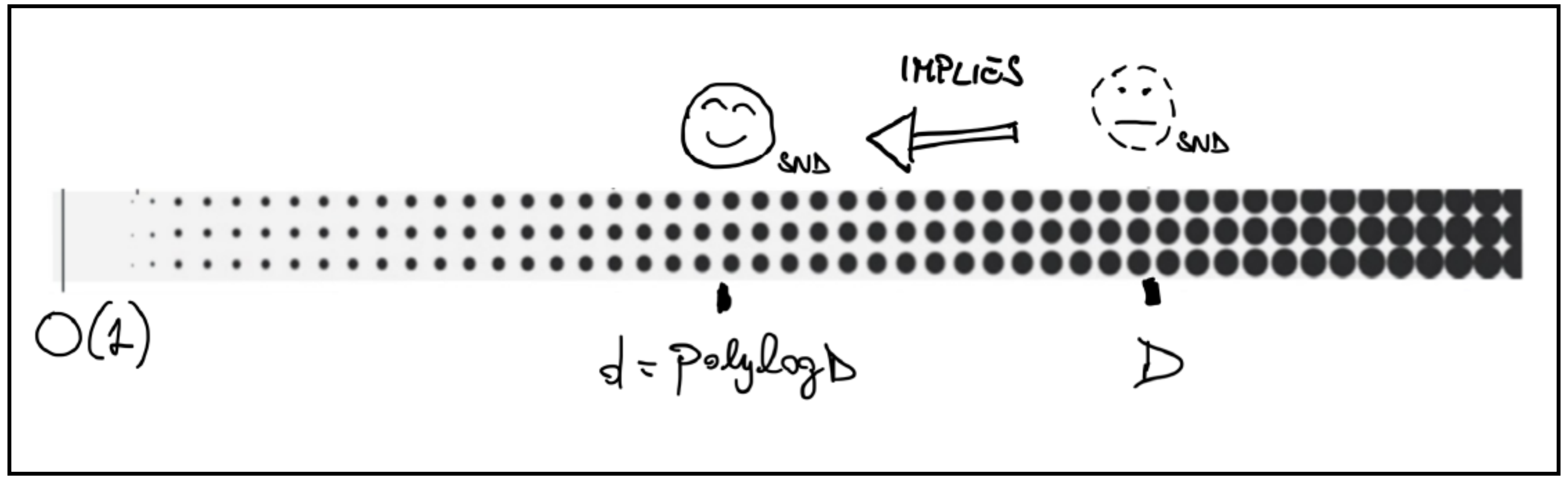
$$\forall \text{PPT } \mathcal{A}\ \ \exists \varepsilon \in negl(\lambda) : ADV^{\mathcal{A}}_{SND}(\lambda) \le \varepsilon(\lambda) \quad \forall \lambda \in \mathbb{N}$$

“Infinitely-often” soundness (io-SND):

$$\exists E \subseteq \mathbb{N}\ \ \forall \text{PPT } \mathcal{A}\ \ \exists \varepsilon \in negl(\lambda) : ADV^{\mathcal{A}}_{SND}(\lambda) \le \varepsilon(\lambda) \quad \forall \lambda \in E$$

What Do We Mean by “Weak” Insecurity? 

:= *infinitely-often* **soundness (io-SND)**

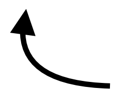


Traditional “almost-everywhere” soundness (ae-SND):

$$\forall \text{PPT } \mathcal{A}\ \exists \varepsilon \in negl(\lambda) : \mathrm{ADV}^{\mathcal{A}}_{\mathrm{SND}}(\lambda) \le \varepsilon(\lambda) \quad \forall \lambda \in \mathbb{N}$$

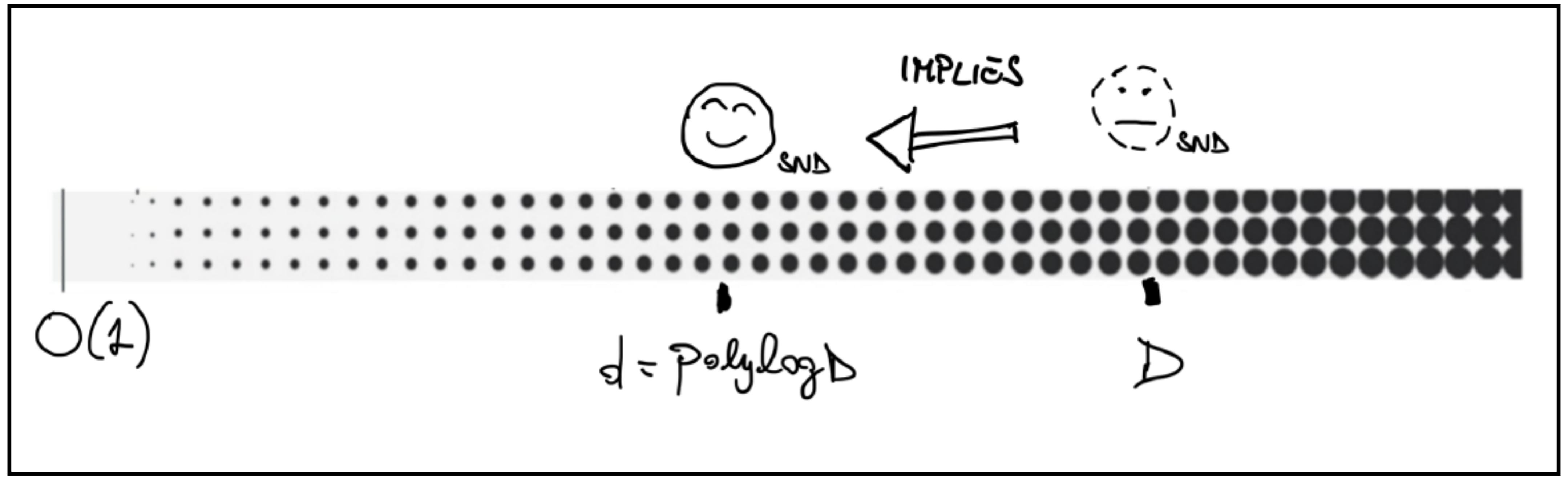
“Infinitely-often” soundness (io-SND):

$$\exists E \subseteq \mathbb{N}\ \forall \text{PPT } \mathcal{A}\ \exists \varepsilon \in negl(\lambda) : \mathrm{ADV}^{\mathcal{A}}_{\mathrm{SND}}(\lambda) \le \varepsilon(\lambda) \quad \forall \lambda \in E$$



infinite set
of parameters

What Do We Mean by “Weak” Insecurity? 



:= *infinitely-often* **soundness (io-SND)**

Traditional “almost-everywhere” soundness (ae-SND):

$$\forall \text{PPT } \mathcal{A} \;\; \exists \varepsilon \in negl(\lambda) : ADV^{\mathcal{A}}_{SND}(\lambda) \le \varepsilon(\lambda) \;\; \forall \lambda \in \mathbb{N}$$

“Infinitely-often” soundness (io-SND):

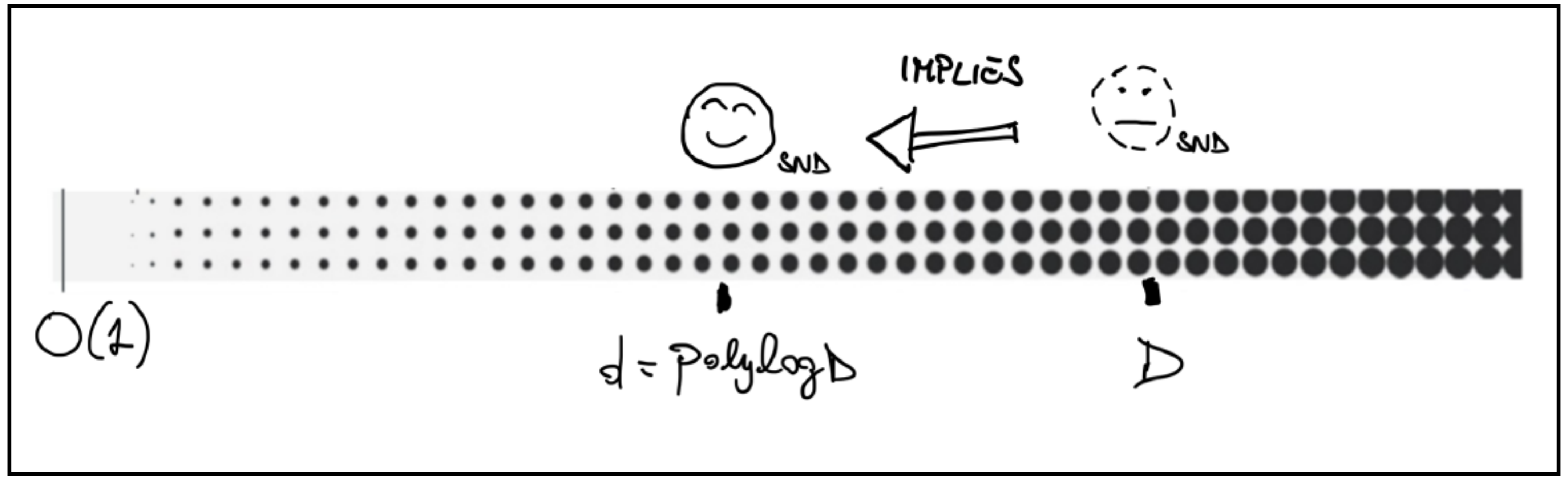
$$\exists E \subseteq \mathbb{N} \;\; \forall \text{PPT } \mathcal{A} \;\; \exists \varepsilon \in negl(\lambda) : ADV^{\mathcal{A}}_{SND}(\lambda) \le \varepsilon(\lambda) \;\; \forall \lambda \in E$$

infinite set of parameters

$$E = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \dots\}$$

What Do We Mean by “Weak” Insecurity?

SND := ***infinitely-often*** **soundness (io-SND)**



Traditional “almost-everywhere” soundness (ae-SND):

$$\forall \text{PPT } \mathcal{A} \;\; \exists \varepsilon \in \text{negl}(\lambda) : \; \text{ADV}^{\mathcal{A}}_{\text{SND}}(\lambda) \le \varepsilon(\lambda) \;\; \forall \lambda \in \mathbb{N}$$

“Infinitely-often” soundness (io-SND):

$$\exists E \subseteq \mathbb{N} \;\; \forall \text{PPT } \mathcal{A} \;\; \exists \varepsilon \in \text{negl}(\lambda) : \; \text{ADV}^{\mathcal{A}}_{\text{SND}}(\lambda) \le \varepsilon(\lambda) \;\; \forall \lambda \in E$$

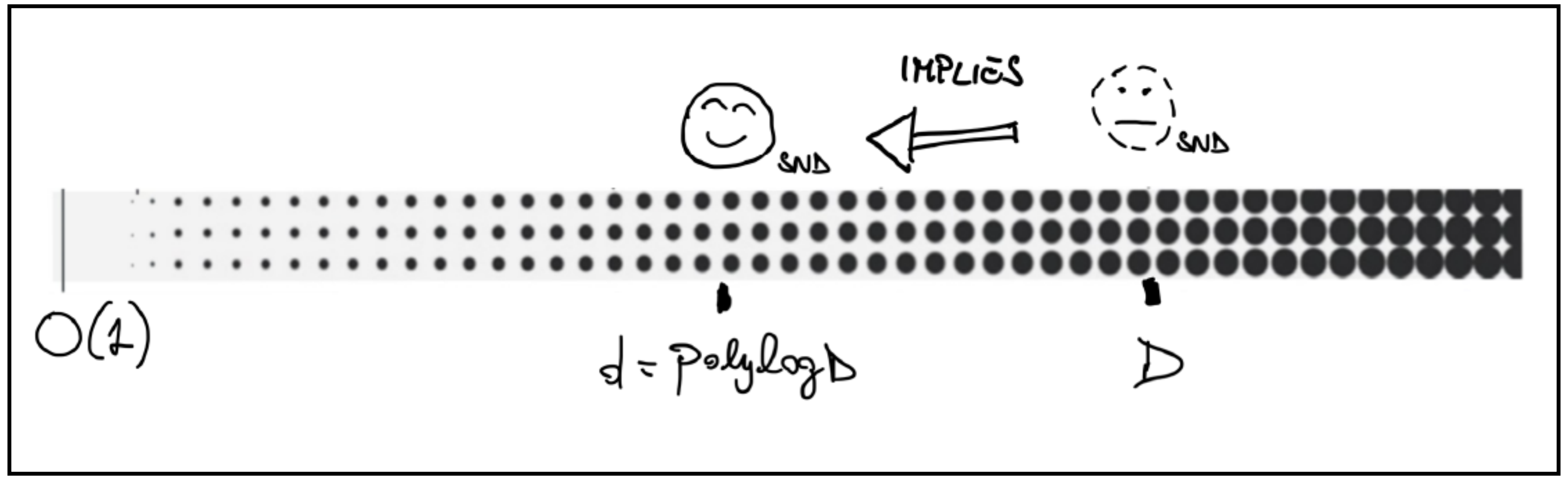
infinite set of parameters

$E = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \ldots\}$

Cryptographers do find i.o. security interesting:

What Do We Mean by "Weak" Insecurity? 

:= ***infinitely-often*** **soundness (io-SND)**



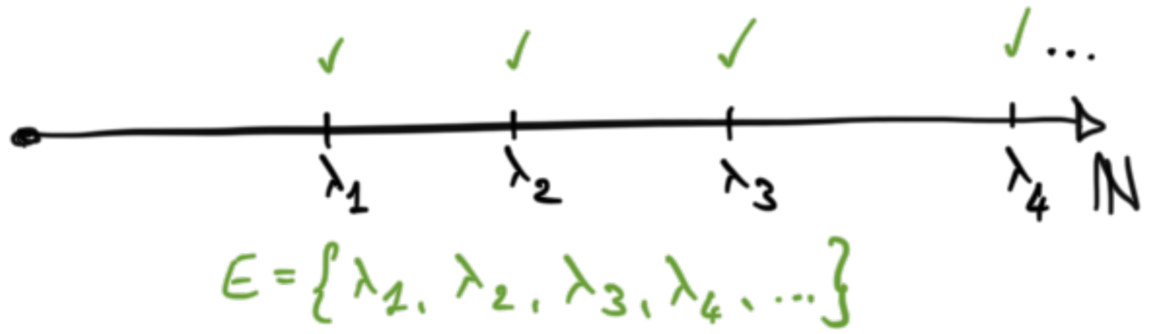
Traditional "almost-everywhere" soundness (ae-SND):

$$\forall \text{PPT } \mathcal{A}\ \exists \varepsilon \in \mathsf{negl}(\lambda) : \mathrm{ADV}^{\mathcal{A}}_{\mathsf{SND}}(\lambda) \le \varepsilon(\lambda)\ \forall \lambda \in \mathbb{N}$$

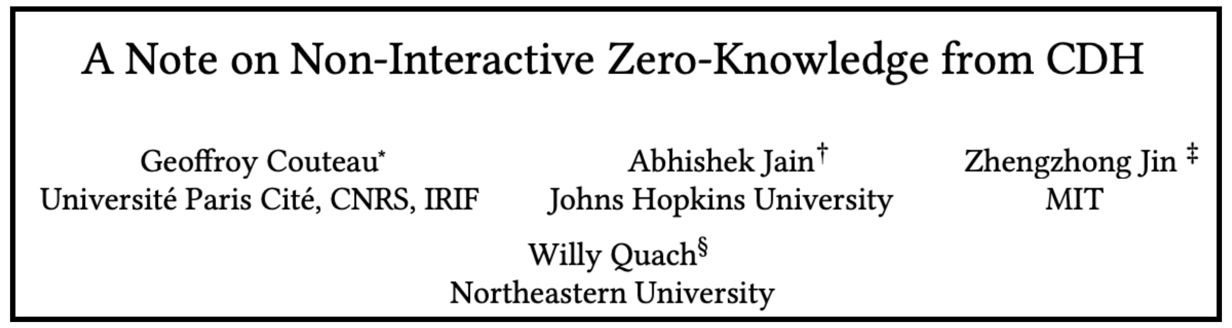
"Infinitely-often" soundness (io-SND):

$$\exists E \subseteq \mathbb{N}\ \forall \text{PPT } \mathcal{A}\ \exists \varepsilon \in \mathsf{negl}(\lambda) : \mathrm{ADV}^{\mathcal{A}}_{\mathsf{SND}}(\lambda) \le \varepsilon(\lambda)\ \forall \lambda \in E$$

infinite set of parameters



Cryptographers do find i.o. security interesting:



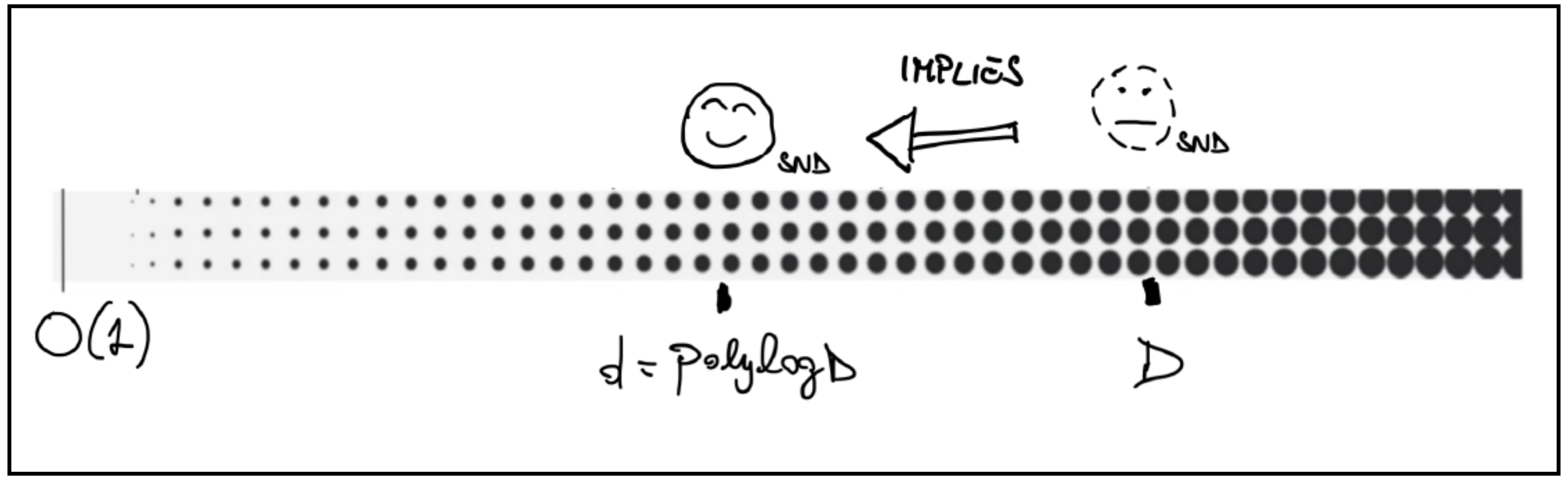
A Note on Non-Interactive Zero-Knowledge from CDH

Geoffroy Couteau[*] Université Paris Cité, CNRS, IRIF
Abhishek Jain[†] Johns Hopkins University
Zhengzhong Jin[‡] MIT
Willy Quach[§] Northeastern University

[CRYPTO '23]: builds i.o.-SND NIZKs from sub-exp CDH.

What Do We Mean by "Weak" Insecurity? 

:= *infinitely-often* **soundness (io-SND)**



Traditional "almost-everywhere" soundness (ae-SND):

$$\forall \text{PPT } \mathcal{A}\ \exists \varepsilon \in \text{negl}(\lambda) : \text{ADV}^{\mathcal{A}}_{\text{SND}}(\lambda) \le \varepsilon(\lambda)\ \forall \lambda \in \mathbb{N}$$

"Infinitely-often" soundness (io-SND):

$$\exists E \subseteq \mathbb{N}\ \forall \text{PPT } \mathcal{A}\ \exists \varepsilon \in \text{negl}(\lambda) : \text{ADV}^{\mathcal{A}}_{\text{SND}}(\lambda) \le \varepsilon(\lambda)\ \forall \lambda \in E$$

infinite set of parameters

$$E = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \ldots\}$$

Cryptographers do find i.o. security interesting:

A Note on Non-Interactive Zero-Knowledge from CDH

Geoffroy Couteau* — Université Paris Cité, CNRS, IRIF
Abhishek Jain† — Johns Hopkins University
Zhengzhong Jin ‡ — MIT
Willy Quach§ — Northeastern University

[CRYPTO '23]: builds i.o.-SND NIZKs from sub-exp CDH.

On the Possibility of Basing Cryptography on EXP ≠ BPP

Yanyi Liu — Cornell University — yl2866@cornell.edu
Rafael Pass* — Cornell Tech — rafael@cs.cornell.edu

One-Way Functions and pKt Complexity

Shuichi Hirahara* Zhenjian Lu† Igor C. Oliveira‡

[CRYPTO '21,TCC '24]:
connect ∃ of i.o.-OWF to certain worst-case assumptions.

Our Results

(continued)

Our Results
(continued)

Motivating question for next result:

Let Π be an IVC (e.g., secure at $O(1)$ depth).

Our Results
(continued)

Motivating question for next result:
Let Π be an IVC (e.g., secure at O(1) depth).

Our Results
(continued)

Motivating question for next result:
Let Π be an IVC (e.g., secure at O(1) depth).

Our Results

(continued)

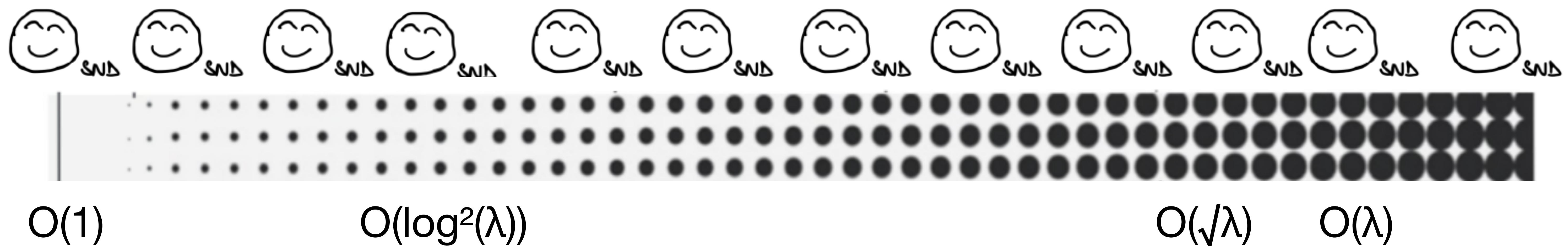
Motivating question for next result:

Let Π be an IVC (e.g., secure at O(1) depth).

Case 1: security everywhere.

Our Results
(continued)

Motivating question for next result:
Let Π be an IVC (e.g., secure at O(1) depth).



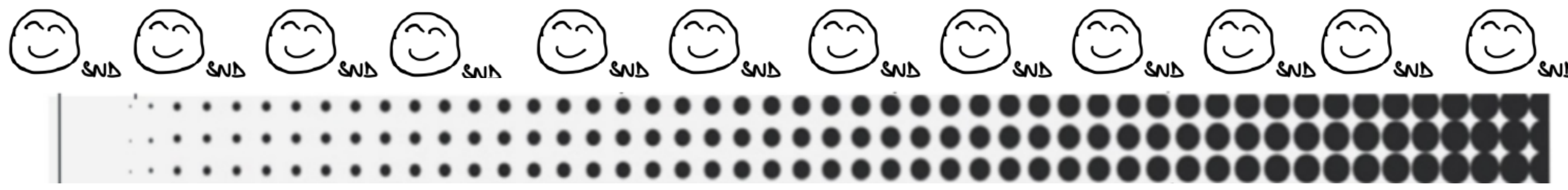
Case 1: security everywhere.

Our Results

(continued)

Motivating question for next result:

Let Π be an IVC (e.g., secure at $O(1)$ depth).



$O(1)$ $O(\log^2(\lambda))$ $O(\sqrt{\lambda})$ $O(\lambda)$

Case 1: security everywhere.

Our Results
(continued)

Motivating question for next result:

Let Π be an IVC.

Our Results

(continued)

Motivating question for next result:

Let Π be an IVC.

Case 2: insecure somewhere.

Our Results
(continued)

Motivating question for next result:

Let Π be an IVC.

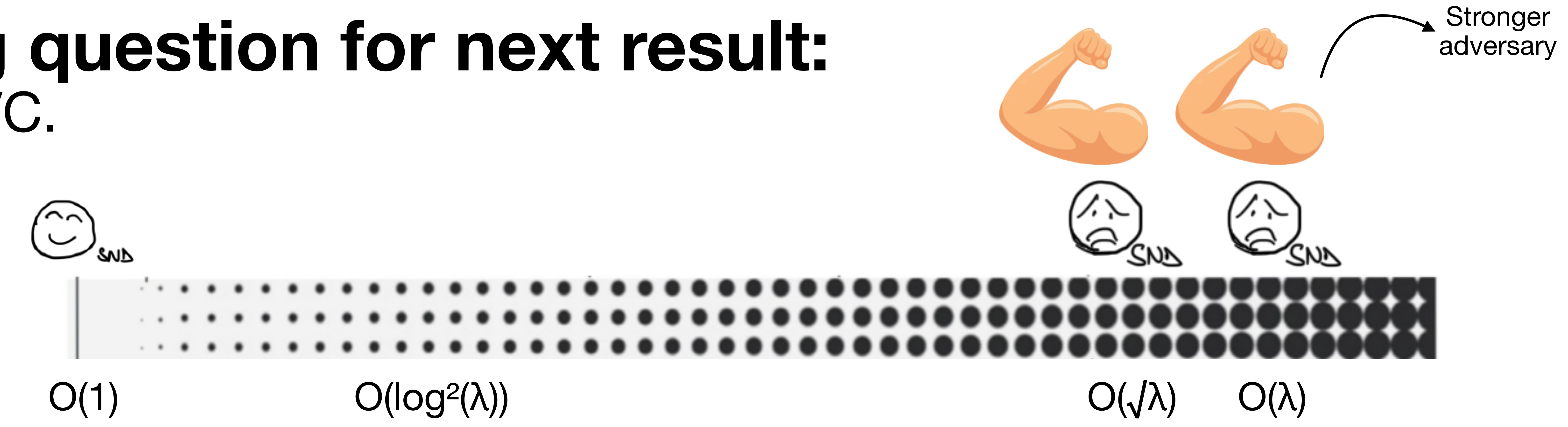
Case 2: insecure somewhere.

Our Results

(continued)

Motivating question for next result:

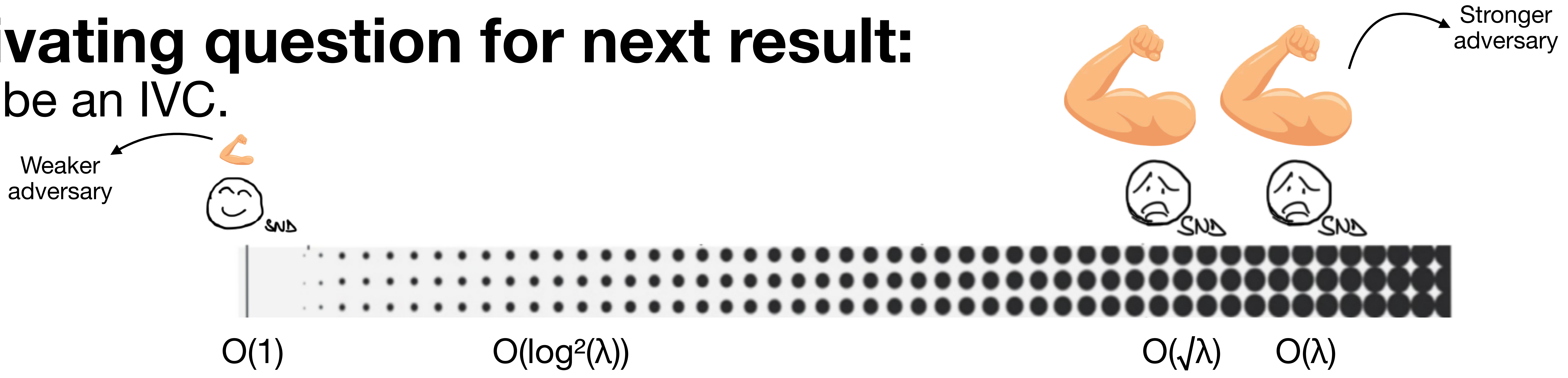
Let Π be an IVC.



Case 2: insecure somewhere.

Our Results
(continued)

Motivating question for next result:
Let Π be an IVC.



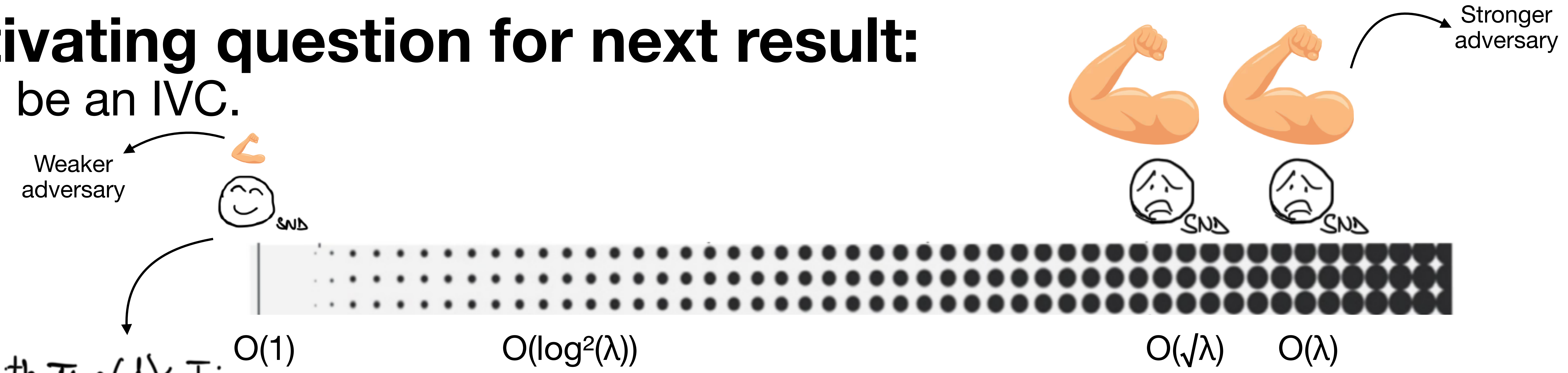
Case 2: insecure somewhere.

Our Results

(continued)

Motivating question for next result:

Let Π be an IVC.



$\forall T = poly(\lambda)$, $\forall \mathcal{A}$ with $Time(\mathcal{A}) \le T$:

$ADV_{\mathcal{A}}(\lambda)@[DEPTH = O(1)]$ is NEGL.

Case 2: insecure somewhere.

Our Results

(continued)

Motivating question for next result:

Let Π be an IVC.

Case 2: insecure somewhere.

Our Results

(continued)

Motivating question for next result:

Let Π be an IVC.

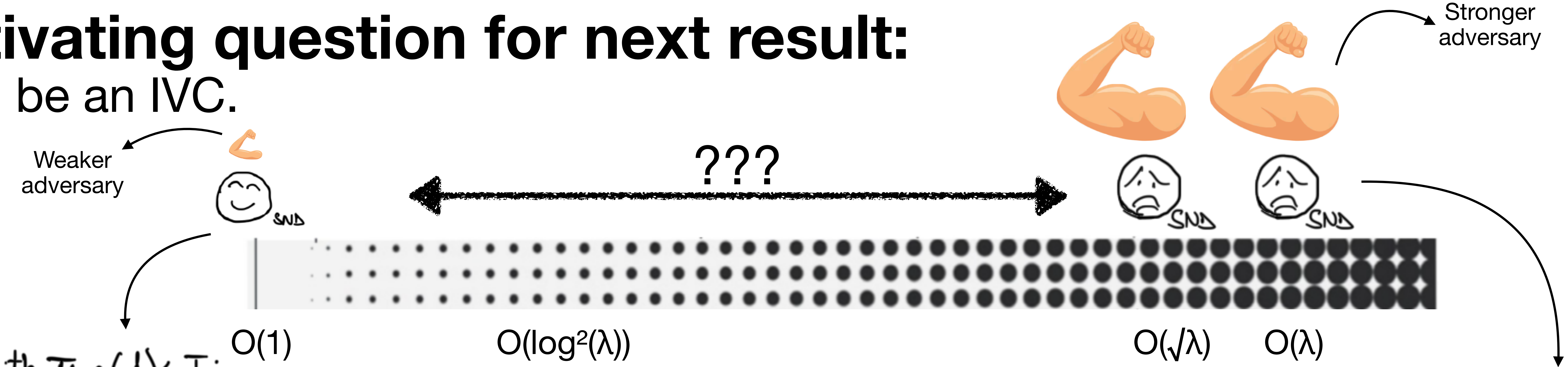
Case 2: insecure somewhere.

Our Results

(continued)

Motivating question for next result:

Let Π be an IVC.



$\forall T = \text{poly}(\lambda)$, $\forall \mathcal{A}$ with $\text{Time}(\mathcal{A}) \leq T$:
$\text{ADV}_{\mathcal{A}}(\lambda)@[\text{DEPTH} = O(1)]$ is NEGL.

Case 2: insecure somewhere.

$\exists T = \text{poly}(\lambda)$, $\mathcal{A}$ with $\text{Time}(\mathcal{A}) \leq T$:
$\text{ADV}_{\mathcal{A}}(\lambda)@[\text{DEPTH} = O(\lambda)]$ NON-NEGL.

$\approx \text{ADV}_{\mathcal{A}}(\lambda) \geq \frac{1}{P(\lambda)}$

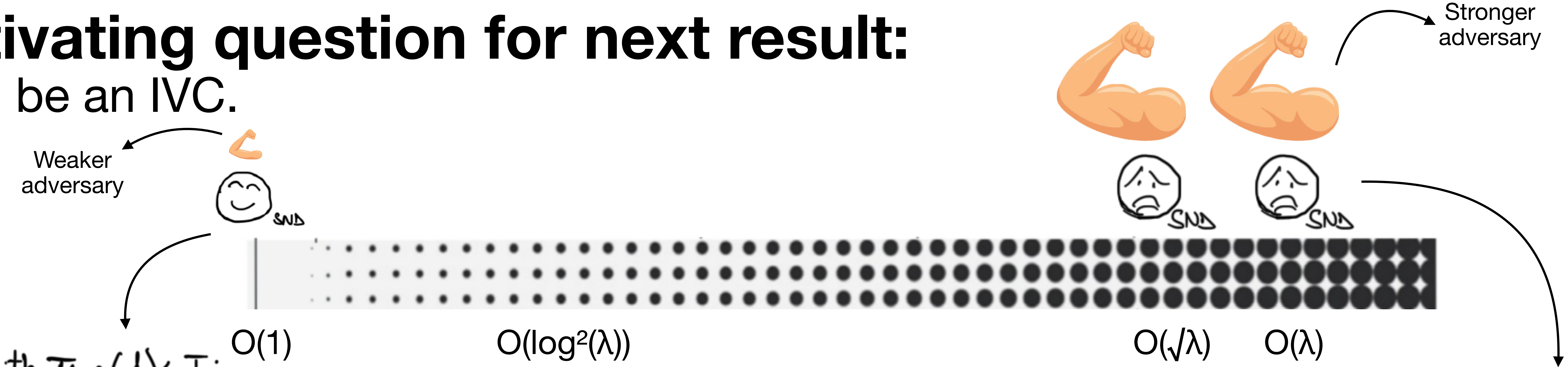
FOR poly $P(\cdot)$, INFINITELY MANY λ-s

Our Results

(continued)

Motivating question for next result:

Let Π be an IVC.



$\forall T = poly(\lambda)$, $\forall \mathcal{A}$ with $Time(\mathcal{A}) \leq T$:
$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(1)]$ is NEGL.

$\exists T = poly(\lambda)$, $\mathcal{A}$ with $Time(\mathcal{A}) \leq T$:
$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(\lambda)]$ NON-NEGL.

$\approx ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{P(\lambda)}$

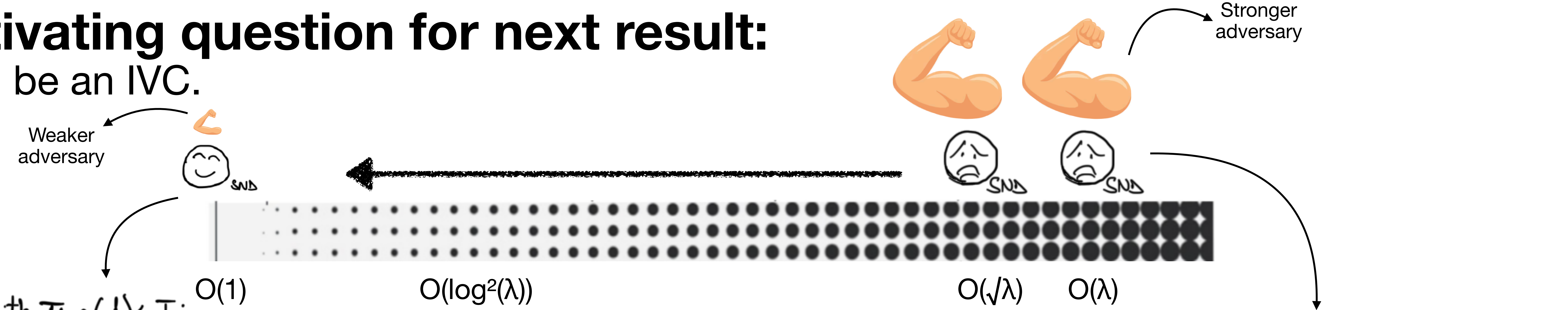
FOR poly $P(\cdot)$, INFINITELY MANY λ-S

Case 2: insecure somewhere.

Our Results
(continued)

Motivating question for next result:

Let Π be an IVC.



$\forall T = poly(\lambda), \forall \mathcal{A}$ with $Time(\mathcal{A}) \leq T$:
$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(1)]$ is NEGL.

$\exists T = poly(\lambda), \mathcal{A}$ with $Time(\mathcal{A}) \leq T$:
$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(\lambda)]$ NON-NEGL.

$\approx ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{P(\lambda)}$

FOR poly $P(\cdot)$, INFINITELY MANY λ-s

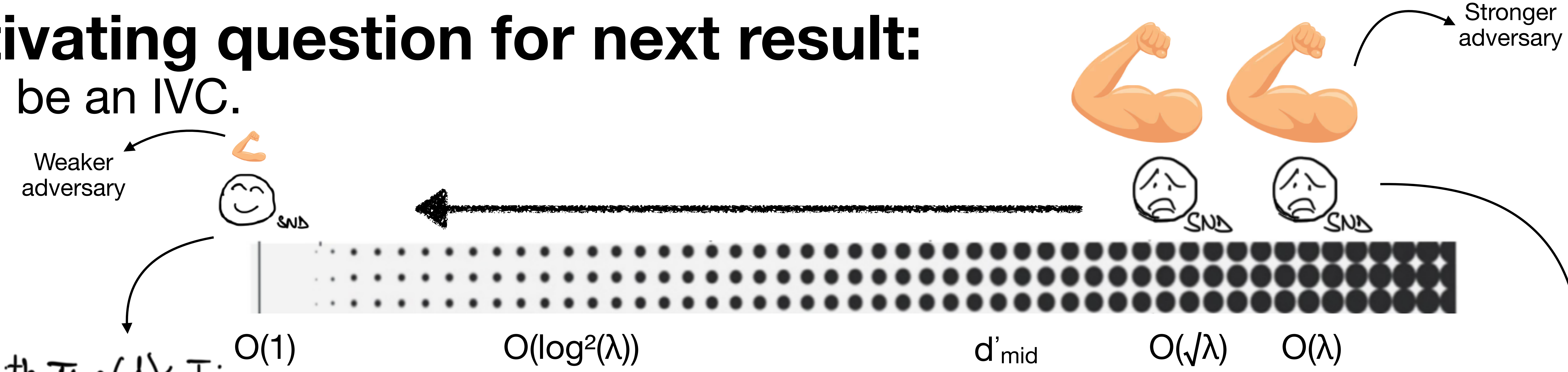
Case 2: insecure somewhere.

Our Results

(continued)

Motivating question for next result:

Let Π be an IVC.



$\forall T = poly(\lambda), \forall \mathcal{A}$ with $Time(\mathcal{A}) \le T$: $Adv_{\mathcal{A}}(\lambda)@[DEPTH = O(1)]$ is NEGL.

$\exists T = poly(\lambda), \mathcal{A}$ with $Time(\mathcal{A}) \le T$: $Adv_{\mathcal{A}}(\lambda)@[DEPTH = O(\lambda)]$ NON-NEGL.

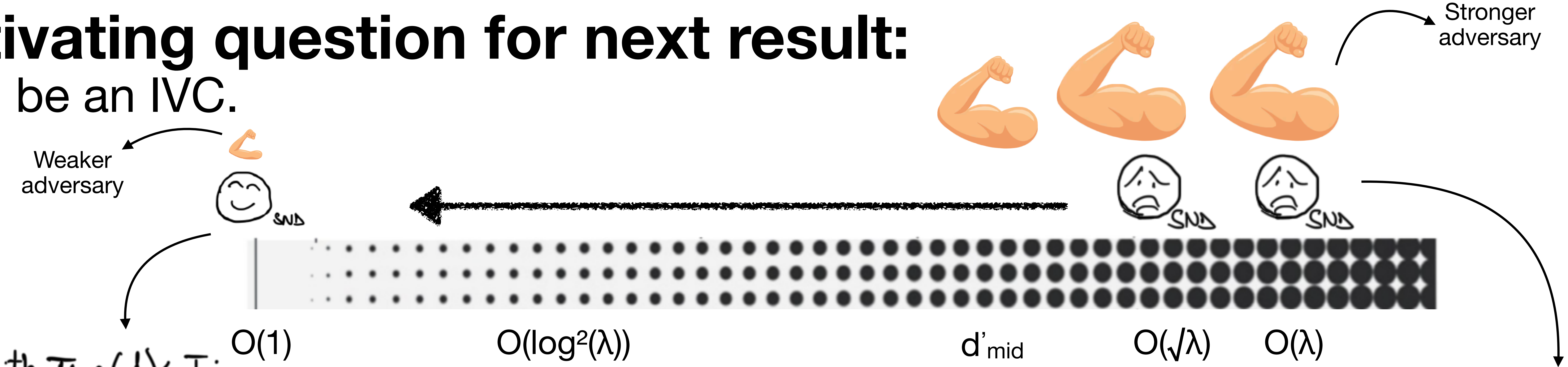
$\approx Adv_{\mathcal{A}}(\lambda) \ge \frac{1}{P(\lambda)}$

FOR poly $P(\cdot)$, INFINITELY MANY λ-s

Case 2: insecure somewhere.

Our Results
(continued)

Motivating question for next result:
Let Π be an IVC.



$\forall T = poly(\lambda), \forall \mathcal{A}$ with $Time(\mathcal{A}) \leq T$:
$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(1)]$ is NEGL.

$\exists T = poly(\lambda), \mathcal{A}$ with $Time(\mathcal{A}) \leq T$:
$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(\lambda)]$ NON-NEGL.

$\approx ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{P(\lambda)}$

FOR poly $P(\cdot)$, INFINITELY MANY λ-s

Case 2: insecure somewhere.

Our Results
(continued)

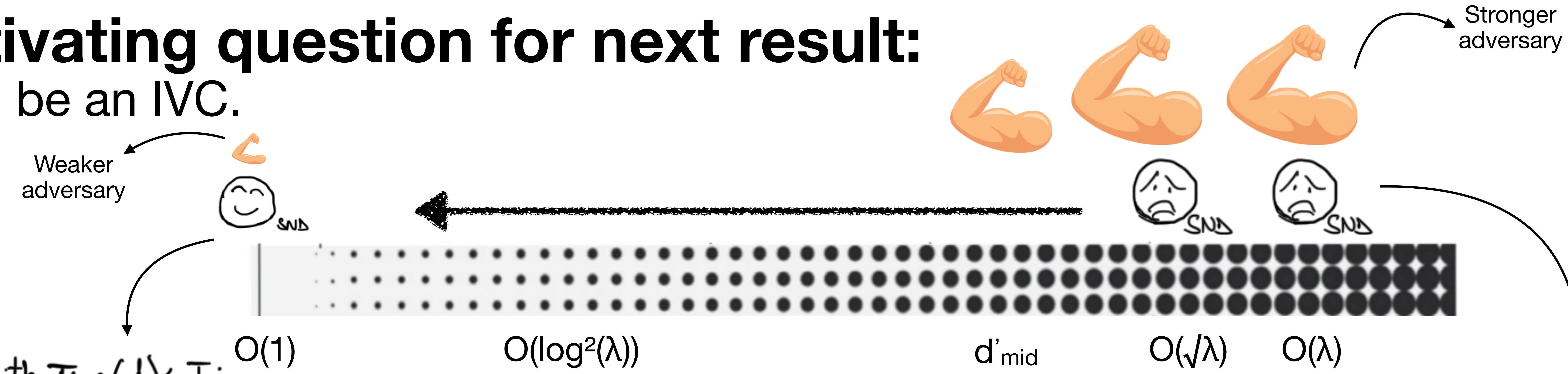
Motivating question for next result:
Let Π be an IVC.

Case 2: insecure somewhere.

Our Results
(continued)

Motivating question for next result:

Let Π be an IVC.



$\forall T = poly(\lambda)$, $\forall \mathcal{A}$ with $Time(\mathcal{A}) \leq T$:
$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(1)]$ is NEGL.

$ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{p'(\lambda)}$ with $p' = \omega(p)$

p' got "worse" from the perspective of adversary

$\exists T = poly(\lambda)$, $\mathcal{A}$ with $Time(\mathcal{A}) \leq T$:
$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(\lambda)]$ NON-NEGL.

$\approx ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{p(\lambda)}$

FOR poly $p(\cdot)$, INFINITELY MANY λ-s

Case 2: insecure somewhere.

Our Results
(continued)

Motivating question for next result:

Let Π be an IVC.

Case 2: insecure somewhere.

Our Results
(continued)

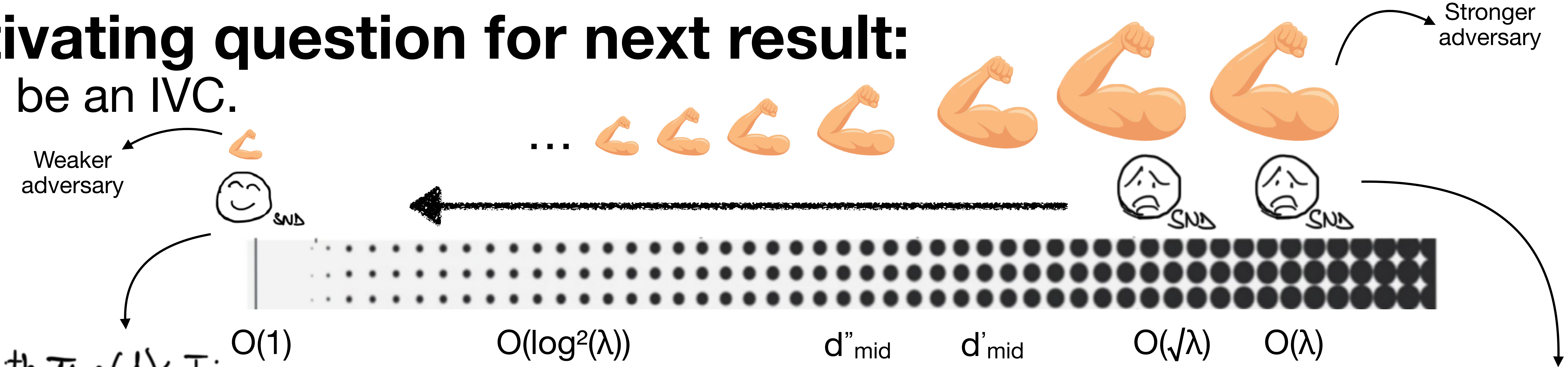
Motivating question for next result:

Let Π be an IVC.

Case 2: insecure somewhere.

Our Results
(continued)

Motivating question for next result:
Let Π be an IVC.



$\forall T = poly(\lambda)$, $\forall \mathcal{A}$ with $Time(\mathcal{A}) \leq T$: $ADV_{\mathcal{A}}(\lambda)@[DEPTH = O(1)]$ is NEGL.

$ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{p'(\lambda)}$ WITH $p' = \omega(p)$

p’ got “worse” from the perspective of adversary

$\exists T = poly(\lambda)$, $\mathcal{A}$ with $Time(\mathcal{A}) \leq T$: $ADV_{\mathcal{A}}(\lambda)@[DEPTH = O(\lambda)]$ NON-NEGL.

$\approx ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{P(\lambda)}$ FOR poly $P(\cdot)$, INFINITELY MANY λ-s

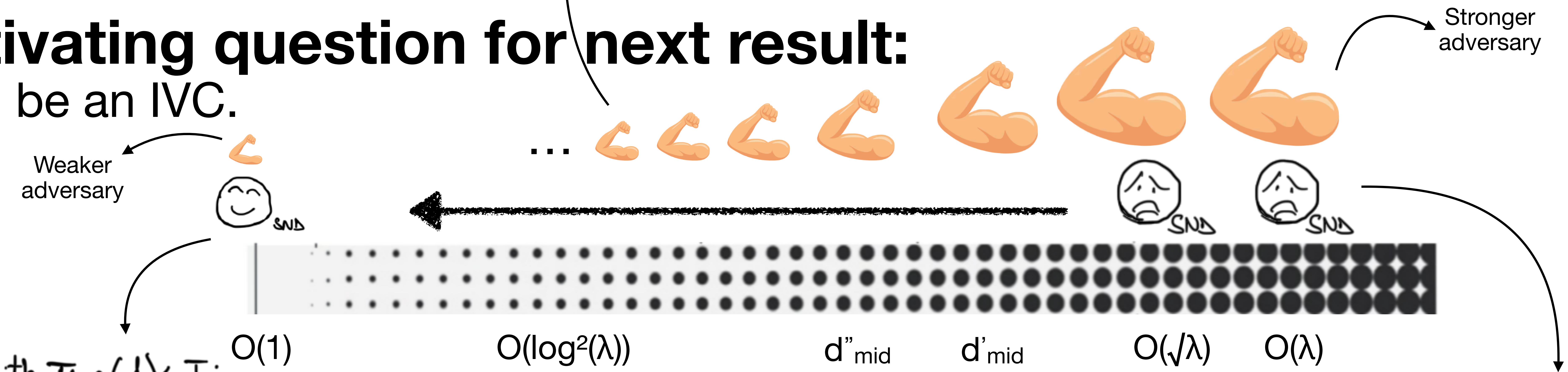
Case 2: insecure somewhere.

Our Results
(continued)

We call this (potential) pattern in IVC
graceful security degradation

Motivating question for next result:

Let Π be an IVC.



$\forall T = poly(\lambda), \forall \mathcal{A}$ with $Time(\mathcal{A}) \le T$:
$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(1)]$ is NEGL.

$ADV_{\mathcal{A}}(\lambda) \ge \frac{1}{p'(\lambda)}$
WITH $p' = \omega(p)$
p' got "worse" from the perspective of adversary

$\exists T = poly(\lambda), \mathcal{A}$ with $Time(\mathcal{A}) \le T$:
$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(\lambda)]$ NON-NEGL.
$\approx ADV_{\mathcal{A}}(\lambda) \ge \frac{1}{P(\lambda)}$
FOR poly $P(\cdot)$, INFINITELY MANY λ-s

Case 2: insecure somewhere.

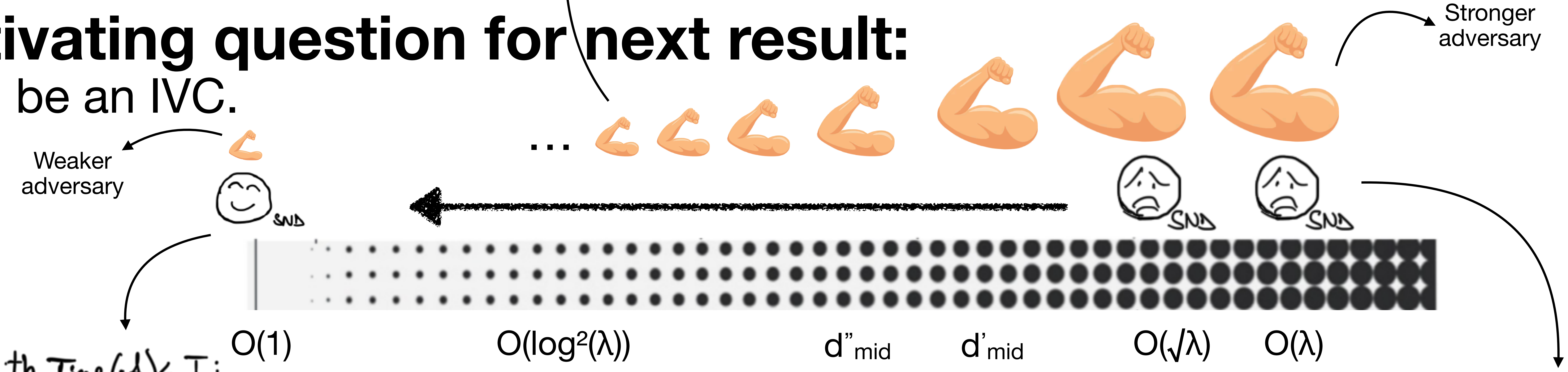
Our Results
(continued)

Q: Can an IVC exhibit it?

We call this (potential) pattern in IVC **graceful security degradation**

Motivating question for next result:

Let Π be an IVC.



$\forall T = poly(\lambda), \forall \mathcal{A}$ with $Time(\mathcal{A}) \le T$: $ADV_{\mathcal{A}}(\lambda)@[DEPTH = O(1)]$ is NEGL.

$ADV_{\mathcal{A}}(\lambda) \ge \frac{1}{p'(\lambda)}$ WITH $p' = \omega(p)$

p' got "worse" from the perspective of adversary

$\exists T = poly(\lambda), \mathcal{A}$ with $Time(\mathcal{A}) \le T$: $ADV_{\mathcal{A}}(\lambda)@[DEPTH = O(\lambda)]$ NON-NEGL.

$\approx ADV_{\mathcal{A}}(\lambda) \ge \frac{1}{p(\lambda)}$

FOR poly $p(\cdot)$, INFINITELY MANY λ-s

Case 2: insecure somewhere.

A practical framing around graceful sec. degradation:

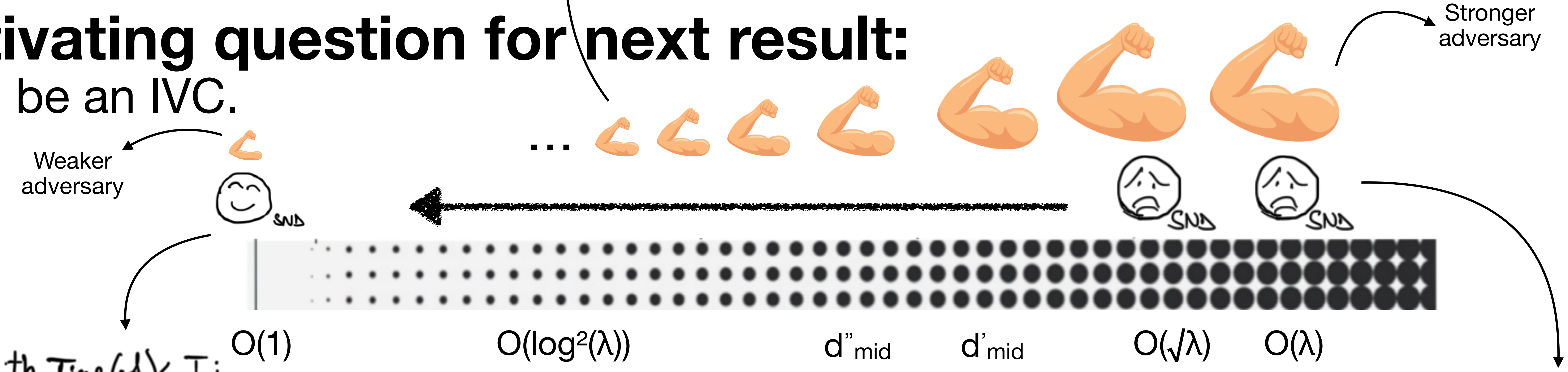
Our Results (continued)

Q: Can an IVC exhibit it?

We call this (potential) pattern in IVC **graceful security degradation**

Motivating question for next result:

Let Π be an IVC.



$\forall T = poly(\lambda), \forall \mathcal{A}$ with $Time(\mathcal{A}) \le T$: $Adv_{\mathcal{A}}(\lambda) @ [DEPTH = O(1)]$ is NEGL.

$Adv_{\mathcal{A}}(\lambda) \ge \frac{1}{p'(\lambda)}$ with $p' = \omega(p)$

p' got "worse" from the perspective of adversary

$\exists T = poly(\lambda), \mathcal{A}$ with $Time(\mathcal{A}) \le T$: $Adv_{\mathcal{A}}(\lambda) @ [DEPTH = O(\lambda)]$ NON-NEGL.

$\approx Adv_{\mathcal{A}}(\lambda) \ge \frac{1}{p(\lambda)}$

FOR poly $p(\cdot)$, INFINITELY MANY λ-s

Case 2: insecure somewhere.

Our Results (continued)

Q: Can an IVC exhibit it?

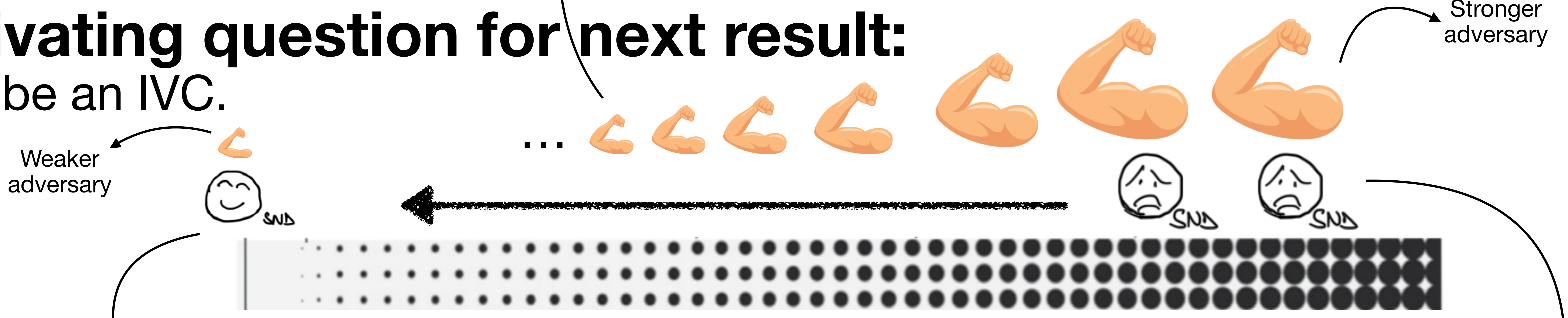
We call this (potential) pattern in IVC **graceful security degradation**

A practical framing around graceful sec. degradation:

💪💪💪💪💪💪 ≈ better and better inverse poly-s

Motivating question for next result:

Let Π be an IVC.



O(1) O(log²(λ)) d”mid d’mid O(√λ) O(λ)

$\forall T = poly(\lambda), \forall \mathcal{A}$ with $Time(\mathcal{A}) \leq T$:
$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(1)]$ is NEGL.

$ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{p'(\lambda)}$ WITH $p' = \omega(p)$

p’ got “worse” from the perspective of adversary

$\exists T = poly(\lambda), \mathcal{A}$ with $Time(\mathcal{A}) \leq T$:
$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(\lambda)]$ NON-NEGL.

$\approx ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{p(\lambda)}$

FOR poly $p(\cdot)$, INFINITELY MANY λ-S

Case 2: insecure somewhere.

Our Results (continued)

Q: Can an IVC exhibit it?

We call this (potential) pattern in IVC **graceful security degradation**

A practical framing around graceful sec. degradation:

$\approx$ better and better inverse poly-s

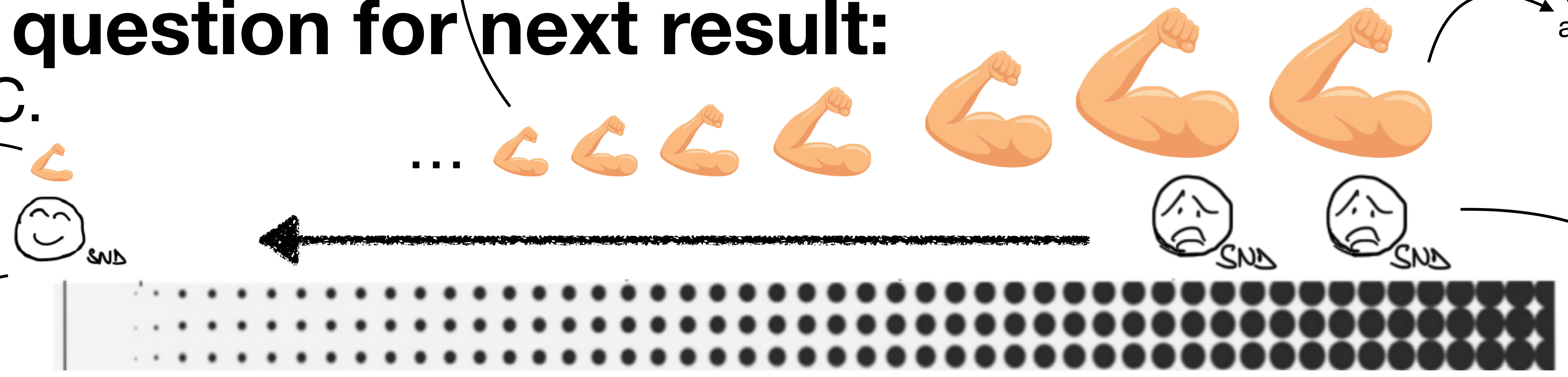
And cryptographers do sometimes work with inverse poly sec.

Motivating question for next result:

Let Π be an IVC.

Weaker adversary

Stronger adversary



O(1) — $O(\log^2(\lambda))$ — d''_{mid} — d'_{mid} — $O(\sqrt{\lambda})$ — $O(\lambda)$

$\forall T = poly(\lambda), \forall \mathcal{A}$ with $Time(\mathcal{A}) \leq T$:

$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(1)]$ is NEGL.

$ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{p'(\lambda)}$

WITH $p' = \omega(p)$

p' got "worse" from the perspective of adversary

$\exists T = poly(\lambda), \mathcal{A}$ with $Time(\mathcal{A}) \leq T$:

$ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(\lambda)]$ NON-NEGL.

$\approx ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{p(\lambda)}$

FOR poly $p(\cdot)$, INFINITELY MANY λ-s

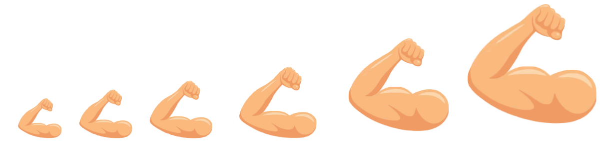
Case 2: insecure somewhere.

Our Results (continued)

Q: Can an IVC exhibit it?

We call this (potential) pattern in IVC **graceful security degradation**

A practical framing around graceful sec. degradation:



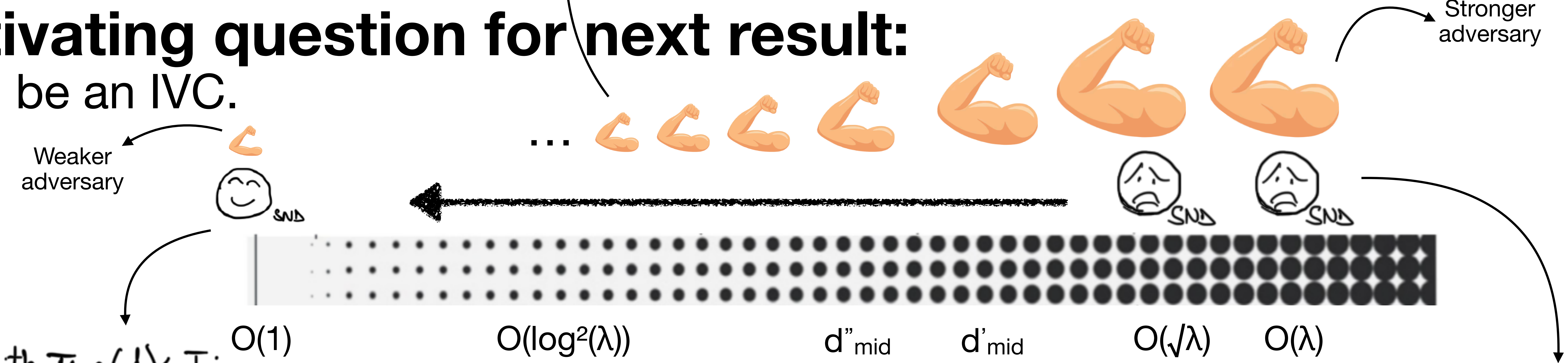
≈ better and better inverse poly-s

And cryptographers do sometimes work with inverse poly sec.

may offer tradeoffs to practitioners.

Motivating question for next result:

Let Π be an IVC.



$\forall T = poly(\lambda), \forall \mathcal{A}$ with $Time(\mathcal{A}) \le T$: $Adv_{\mathcal{A}}(\lambda) @ [DEPTH = O(1)]$ is NEGL.

Case 2: insecure somewhere.

$Adv_{\mathcal{A}}(\lambda) \ge \frac{1}{p'(\lambda)}$ WITH $p' = \omega(p)$

p' got "worse" from the perspective of adversary

$\exists T = poly(\lambda), \mathcal{A}$ with $Time(\mathcal{A}) \le T$: $Adv_{\mathcal{A}}(\lambda) @ [DEPTH = O(\lambda)]$ NON-NEGL.

$\approx Adv_{\mathcal{A}}(\lambda) \ge \frac{1}{p(\lambda)}$

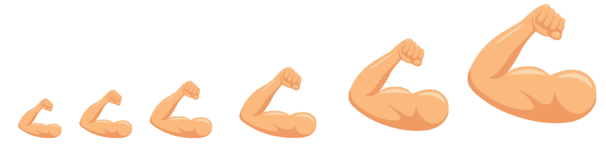
FOR poly $p(\cdot)$, INFINITELY MANY λ-s

Our Results (continued)

Q: Can an IVC exhibit it?

We call this (potential) pattern in IVC **graceful security degradation**

A practical framing around graceful sec. degradation:

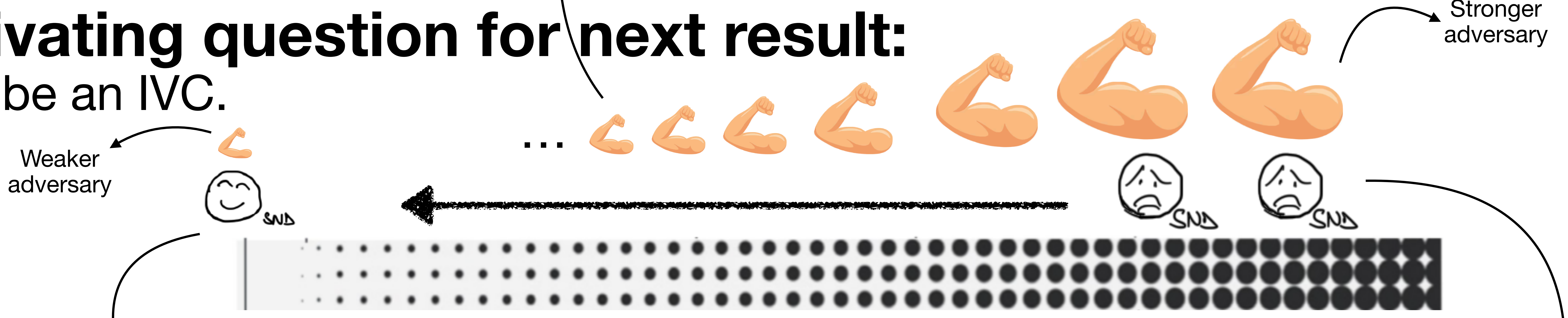
 ≈ better and better inverse poly-s

And cryptographers do sometimes work with inverse poly sec.

 may offer tradeoffs to practitioners.

Motivating question for next result:

Let Π be an IVC.



O(1) O($\log^2(\lambda)$) d''$_{mid}$ d'$_{mid}$ O($\sqrt{\lambda}$) O(λ)

$\forall T = poly(\lambda)$, $\forall \mathcal{A}$ with $Time(\mathcal{A}) \leq T$: $ADV_{\mathcal{A}}(\lambda)@[DEPTH = O(1)]$ is NEGL.

$ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{p'(\lambda)}$ WITH $p' = \omega(p)$

p' got "worse" from the perspective of adversary

$\exists T = poly(\lambda)$, $\mathcal{A}$ with $Time(\mathcal{A}) \leq T$: $ADV_{\mathcal{A}}(\lambda)@[DEPTH = O(\lambda)]$ NON-NEGL.

$\approx ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{p(\lambda)}$ FOR poly $p(\cdot)$, INFINITELY MANY λ-s

Case 2: insecure somewhere.

Result ("no free snack" theorem):

Let Π be an IVC. Then:

Our Results (continued)

Q: Can an IVC exhibit it?

We call this (potential) pattern in IVC **graceful security degradation**

A practical framing around graceful sec. degradation:

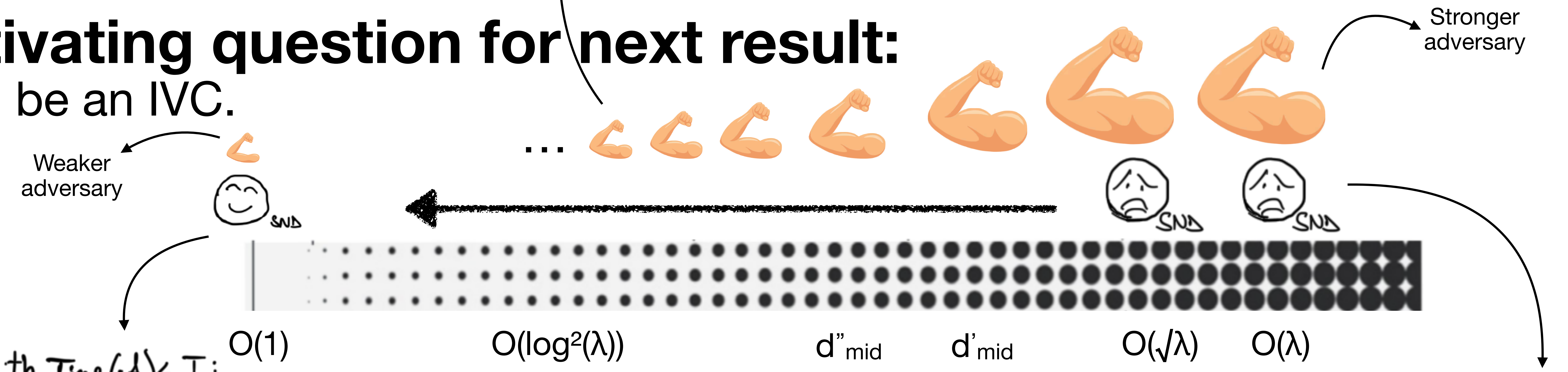
≈ better and better inverse poly-s

And cryptographers do sometimes work with inverse poly sec.

may offer tradeoffs to practitioners.

Motivating question for next result:

Let Π be an IVC.



$\forall T = poly(\lambda)$, $\forall \mathcal{A}$ with $Time(\mathcal{A}) \leq T$: $ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(1)]$ is NEGL.

$ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{p'(\lambda)}$ WITH $p' = \omega(p)$

p' got "worse" from the perspective of adversary

$\exists T = poly(\lambda)$, $\mathcal{A}$ with $Time(\mathcal{A}) \leq T$: $ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(\lambda)]$ NON-NEGL.

$\approx ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{p(\lambda)}$

FOR poly $p(\cdot)$, INFINITELY MANY λ-s

Case 2: insecure somewhere.

Result ("no free snack" theorem):

Let Π be an IVC. Then:

- **<u>either</u> Π is secure at arbitrary polynomial depths,**

Our Results
(continued)

Q: Can an IVC exhibit it?

We call this (potential) pattern in IVC **graceful security degradation**

A practical framing around graceful sec. degradation:

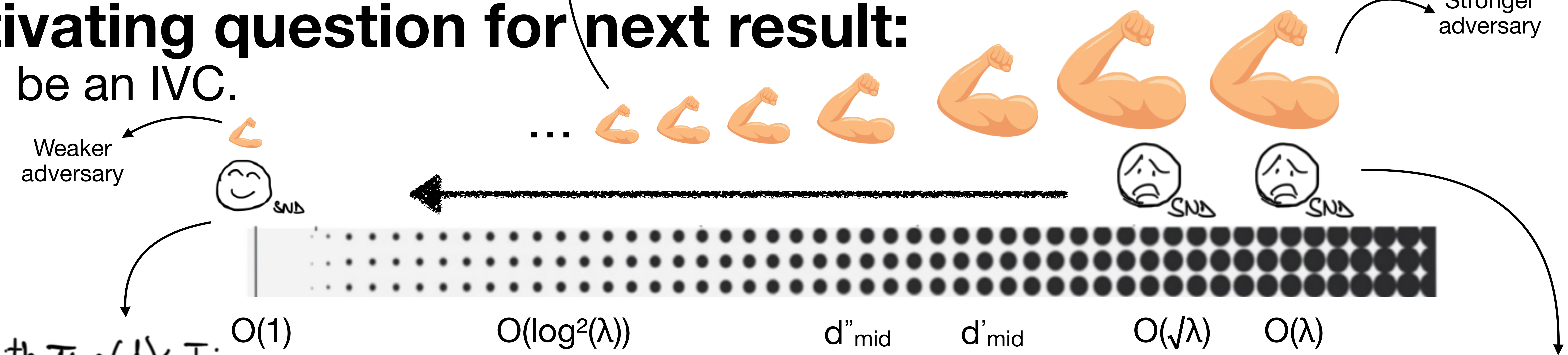
≈ better and better inverse poly-s

And cryptographers do sometimes work with inverse poly sec.

may offer tradeoffs to practitioners.

Motivating question for next result:

Let Π be an IVC.



$\forall T = poly(\lambda), \forall \mathcal{A}$ with $Time(\mathcal{A}) \leq T$: $ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(1)]$ is NEGL.

$ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{p'(\lambda)}$ WITH $p' = \omega(p)$

p' got "worse" from the perspective of adversary

$\exists T = poly(\lambda), \mathcal{A}$ with $Time(\mathcal{A}) \leq T$: $ADV_{\mathcal{A}}(\lambda) @ [DEPTH = O(\lambda)]$ NON-NEGL.

$\approx ADV_{\mathcal{A}}(\lambda) \geq \frac{1}{P(\lambda)}$

FOR poly $P(\cdot)$, INFINITELY MANY λ-s

Case 2: insecure somewhere.

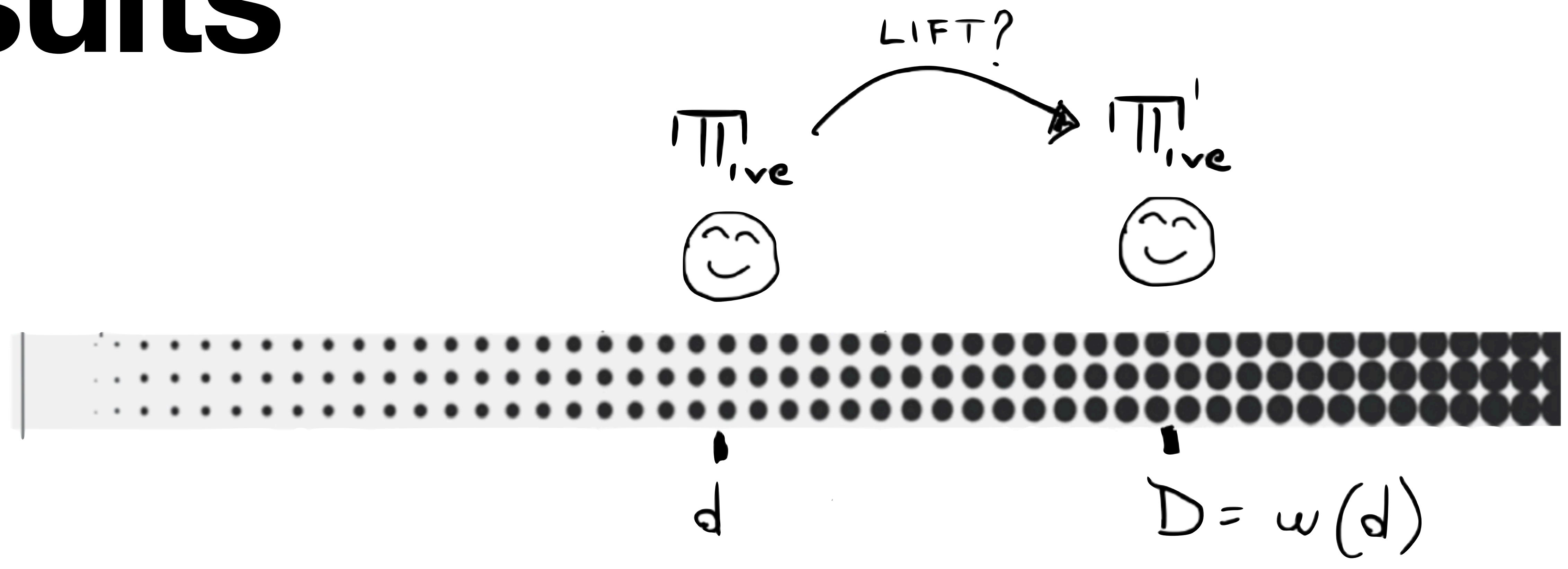
Result ("no free snack" theorem):

Let Π be an IVC. Then:

- **<u>either</u> Π is secure at arbitrary polynomial depths,**
- **<u>or</u> Π <u>cannot</u> exhibit graceful security degradation.**

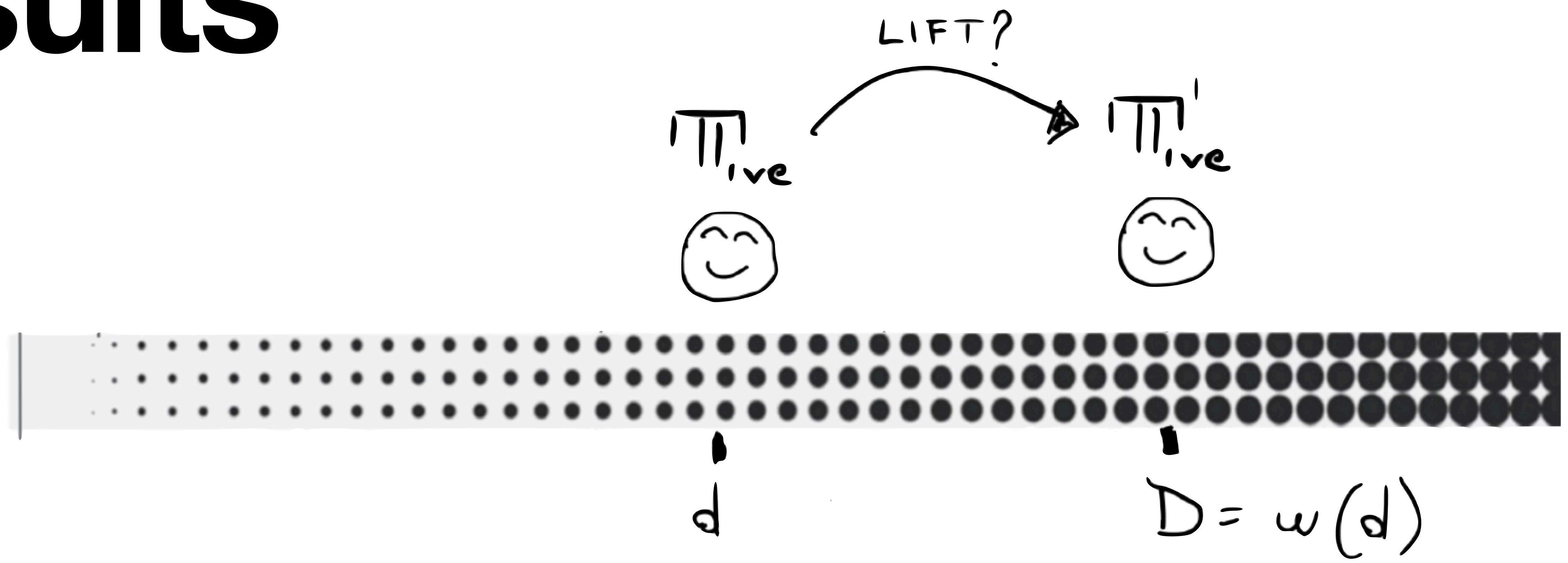
Our Results
(continued)

Our Results
(continued)



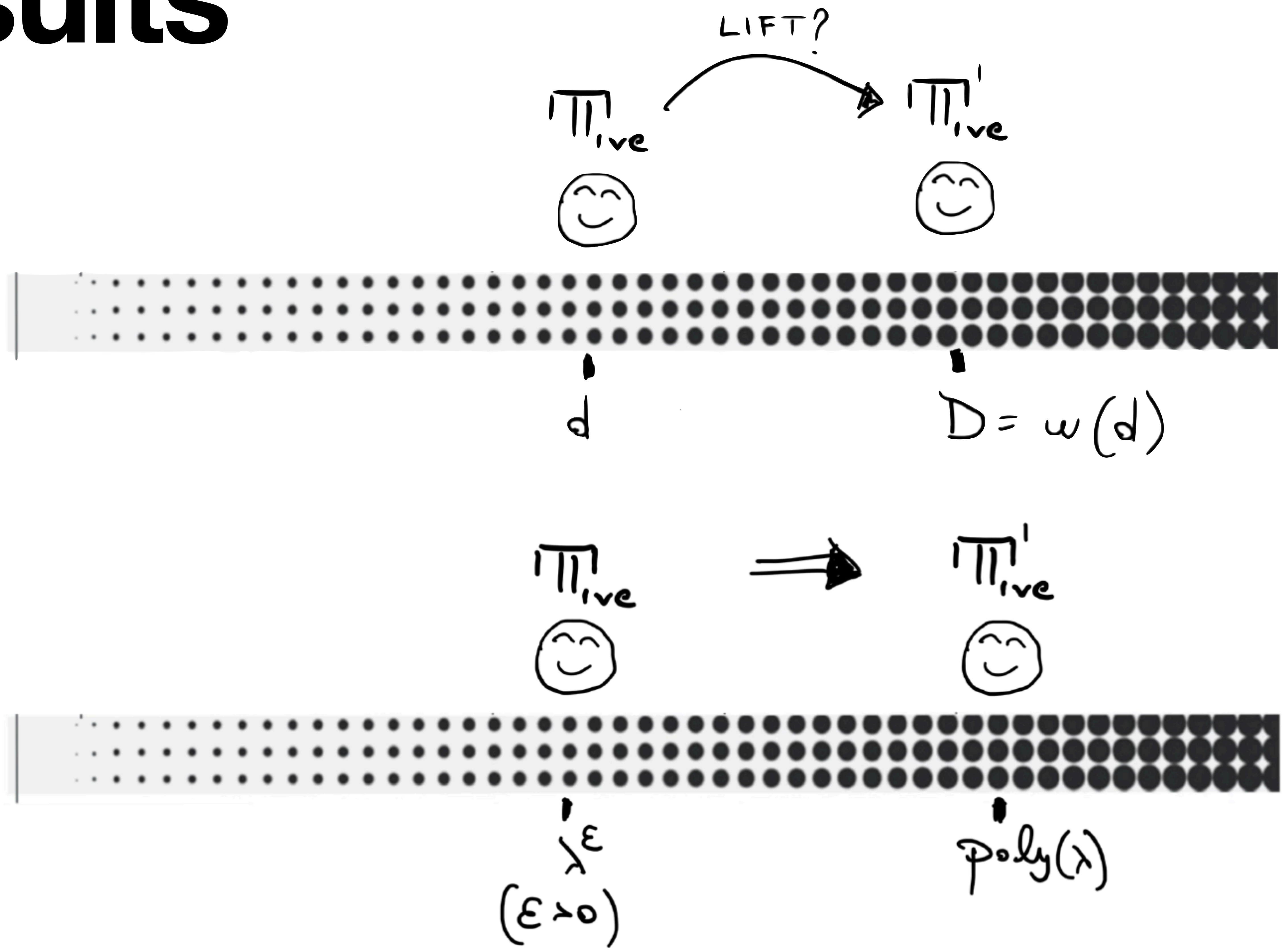
Our Results
(continued)

NB: We are interested in:
- <u>black-box</u> lifting results,
- and that preserve performance.



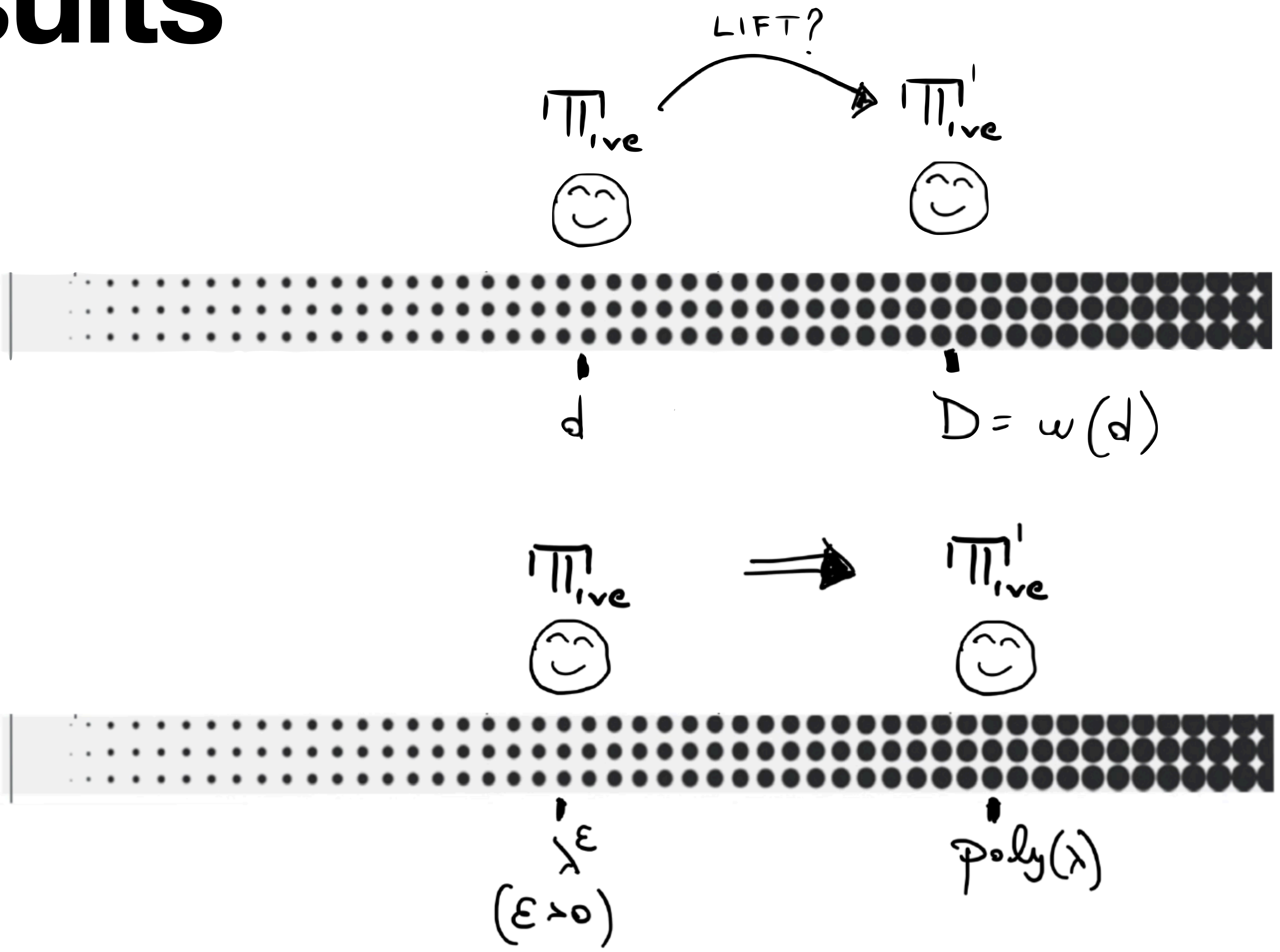
Our Results
(continued)

NB: We are interested in:
- black-box lifting results,
- and that preserve performance.



Our Results
(continued)

NB: We are interested in:
- black-box lifting results,
- and that preserve performance.



Theorem (sublinear depths):

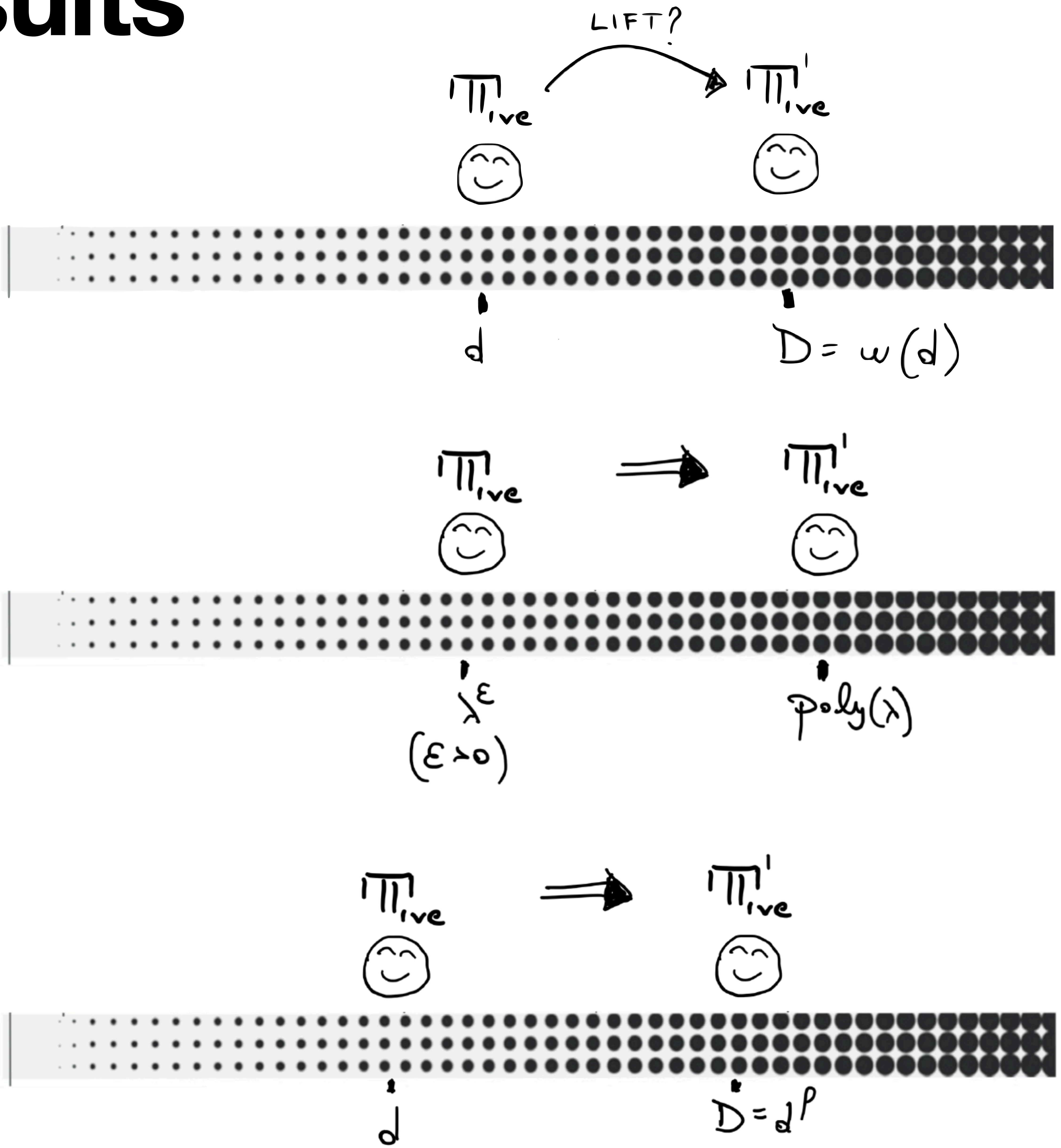
∃ IVC Π SND at depth λ^{ε} (for some ε>0)
⇒ ∃ IVC Π' SND at arbitrary depth.

Overhead for P/V/proof size in Π' is $O_{\lambda}(1)$.

Our Results

(continued)

NB: We are interested in:
- black-box lifting results,
- and that preserve performance.



Theorem (sublinear depths):

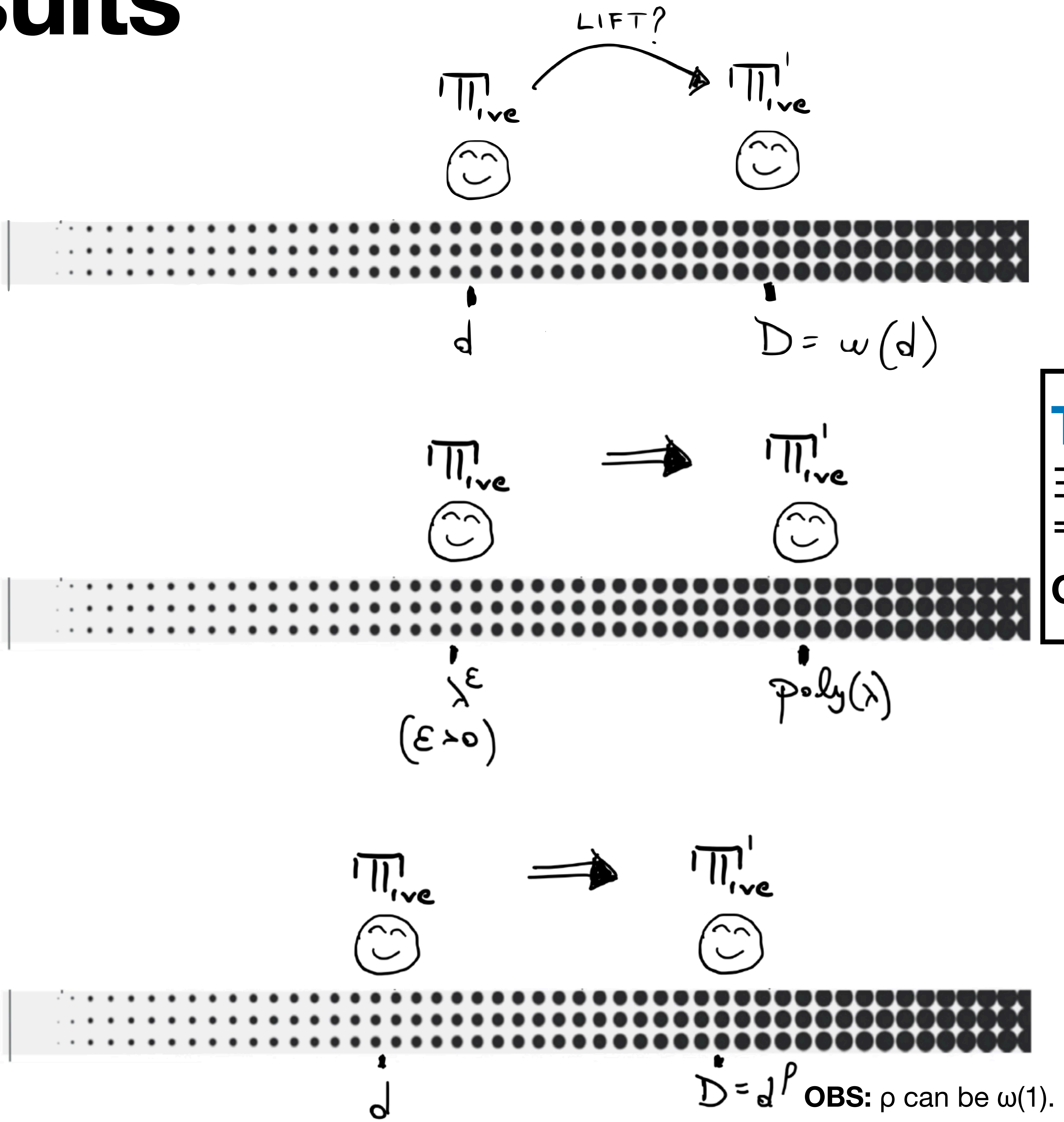
∃ IVC Π SND at depth λ^{ε} (for some ε>0)
⇒ ∃ IVC Π' SND at arbitrary depth.

Overhead for P/V/proof size in Π' is $O_{\lambda}(1)$.

Our Results

(continued)

NB: We are interested in:
- black-box lifting results,
- and that preserve performance.



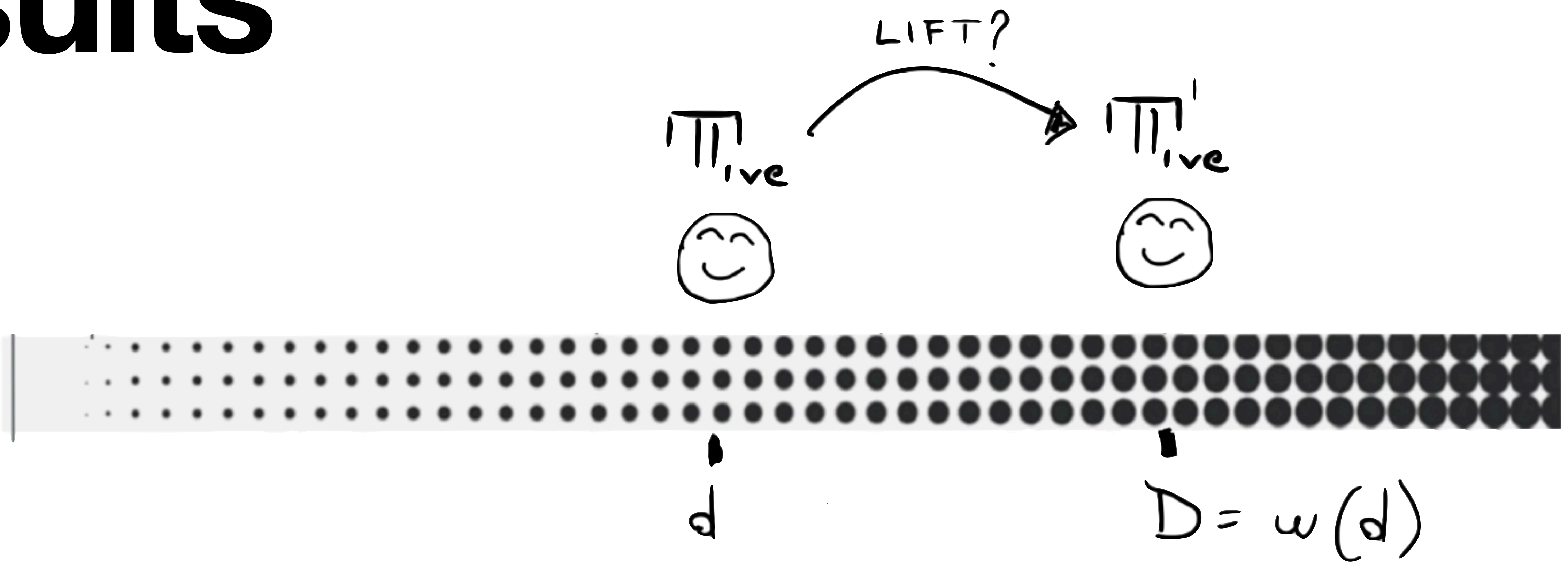
Theorem (sublinear depths):

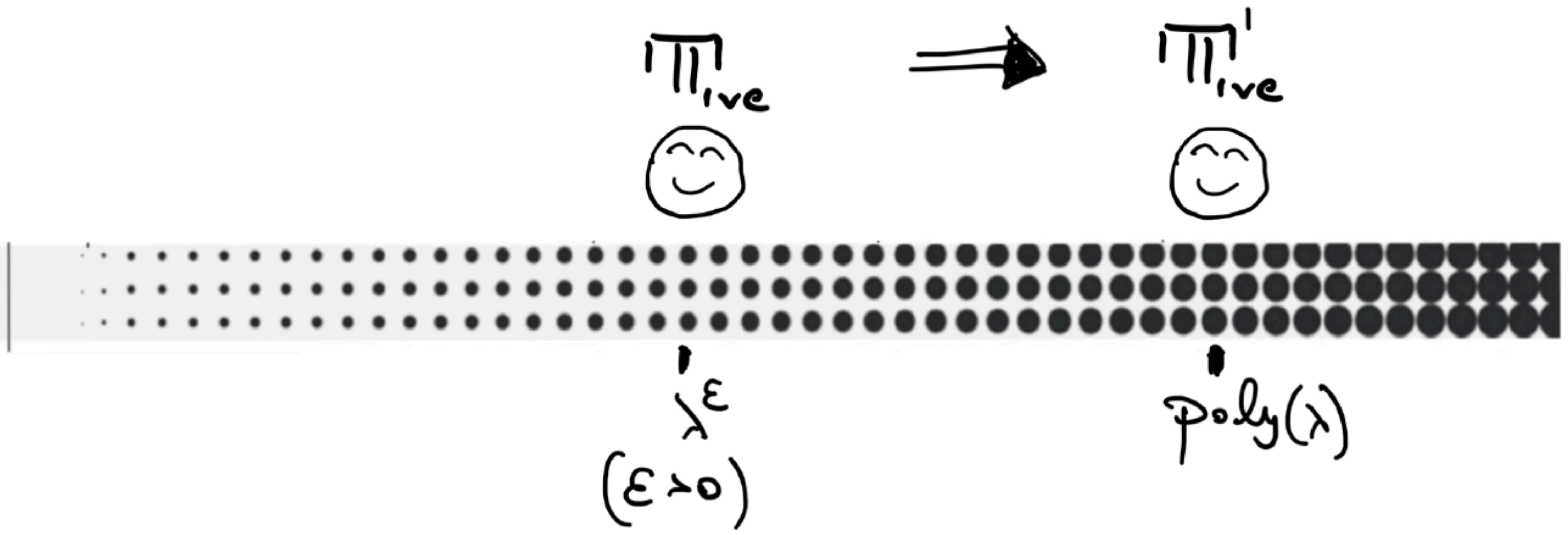
∃ IVC Π SND at depth λ^ε (for some ε>0)
⇒ ∃ IVC Π' SND at arbitrary depth.

Overhead for P/V/proof size in Π' is $O_\lambda(1)$.

Our Results
(continued)

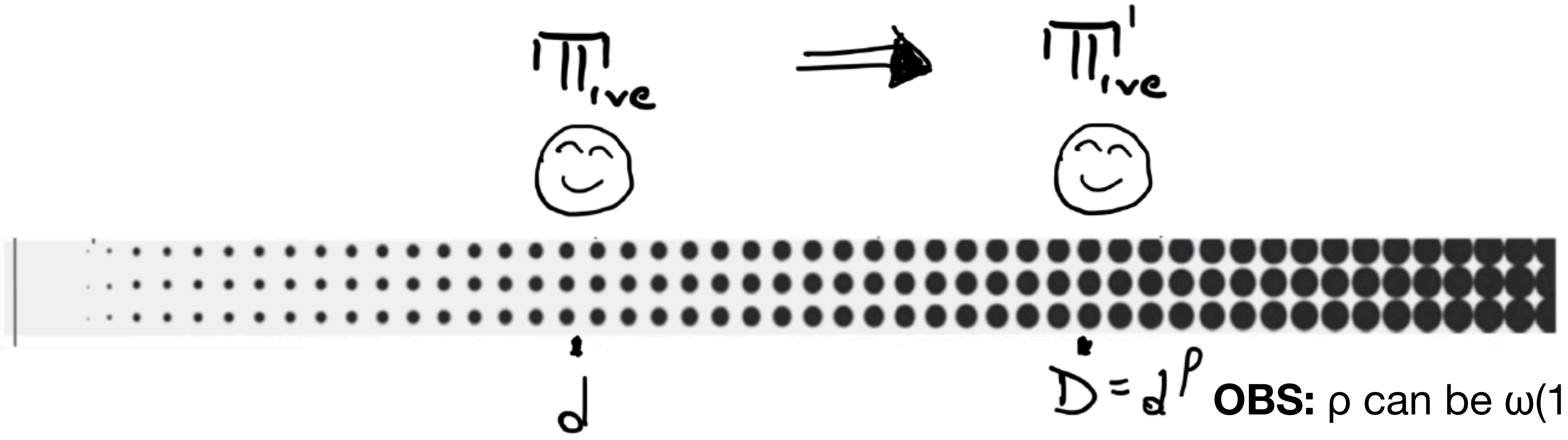
NB: We are interested in:
- black-box lifting results,
- and that preserve performance.





Theorem (sublinear depths):
∃ IVC Π SND at depth λ^{ε} (for some ε>0)
⇒ ∃ IVC Π' SND at arbitrary depth.

Overhead for P/V/proof size in Π' is $O_\lambda(1)$.



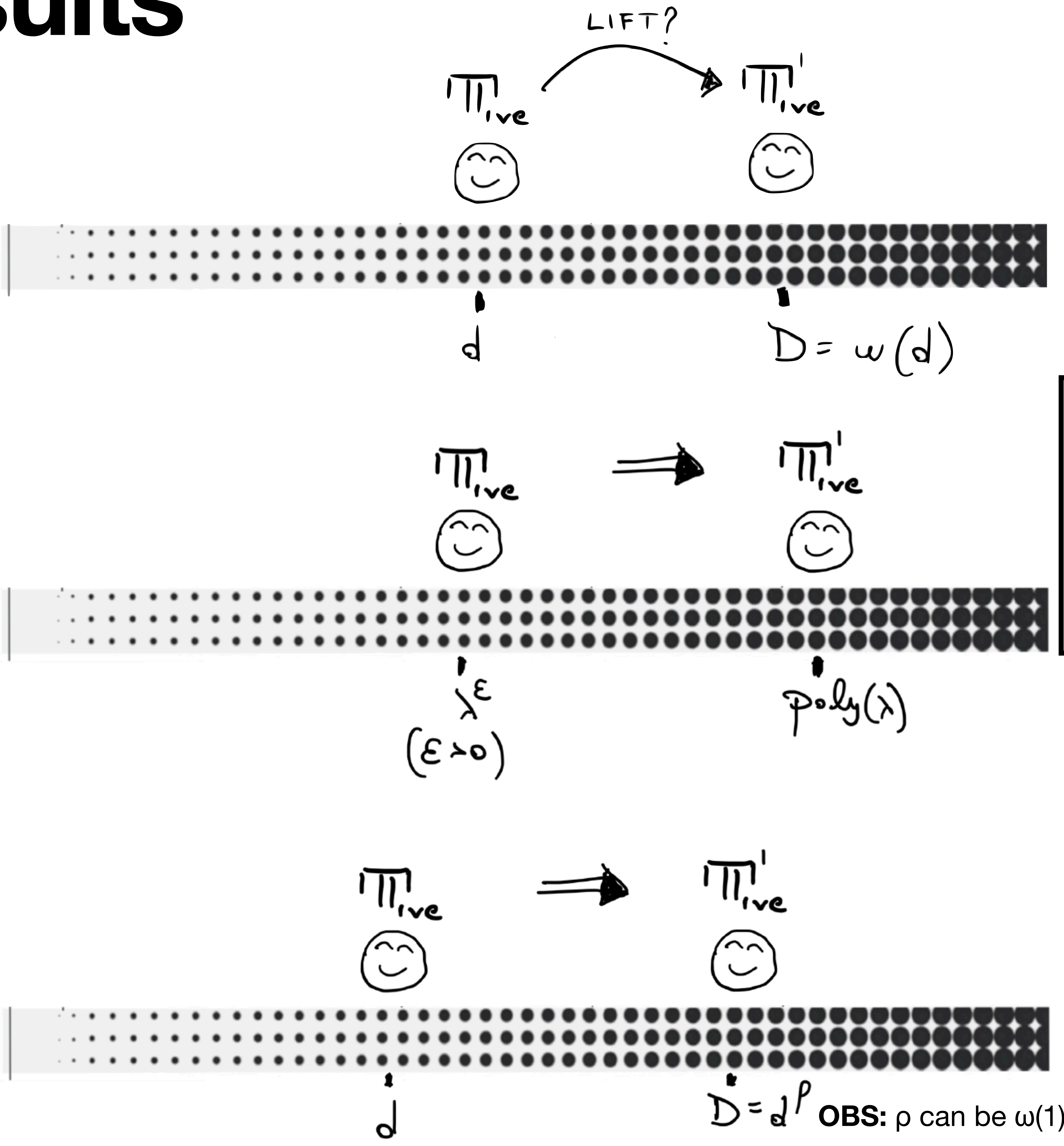
OBS: ρ can be ω(1).

Theorem (general lifting):
∃ IVC Π SND at depth d
⇒ ∃ IVC Π' SND at depth $D = d^{\rho}$.

Overhead* for P/V/proof size in Π' is $O_\lambda(\rho)$

Our Results
(continued)

NB: We are interested in:
- black-box lifting results,
- and that preserve performance.



OBS: ρ can be ω(1).

Theorem (sublinear depths):

∃ IVC Π SND at depth λ^{ε} (for some ε>0)
⇒ ∃ IVC Π' SND at arbitrary depth.

Overhead for P/V/proof size in Π' is $O_\lambda(1)$.

Theorem (general lifting):

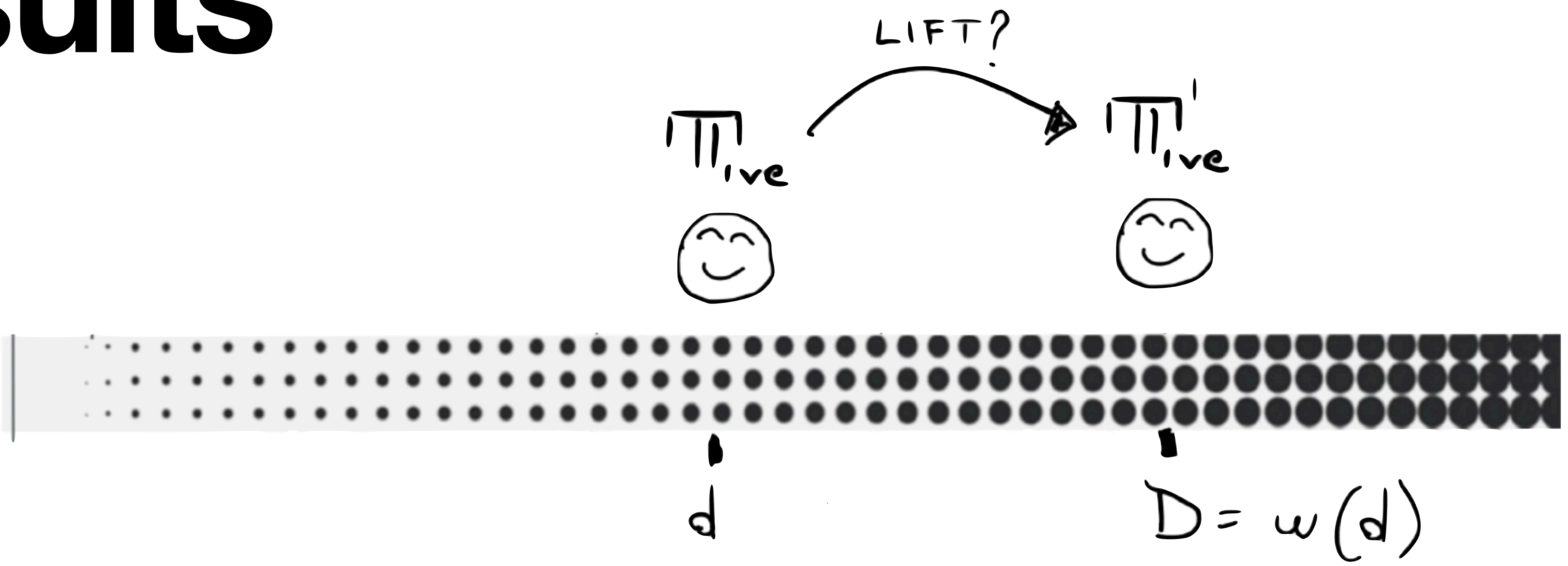
∃ IVC Π SND at depth d
⇒ ∃ IVC Π' SND at depth D = d^ρ.

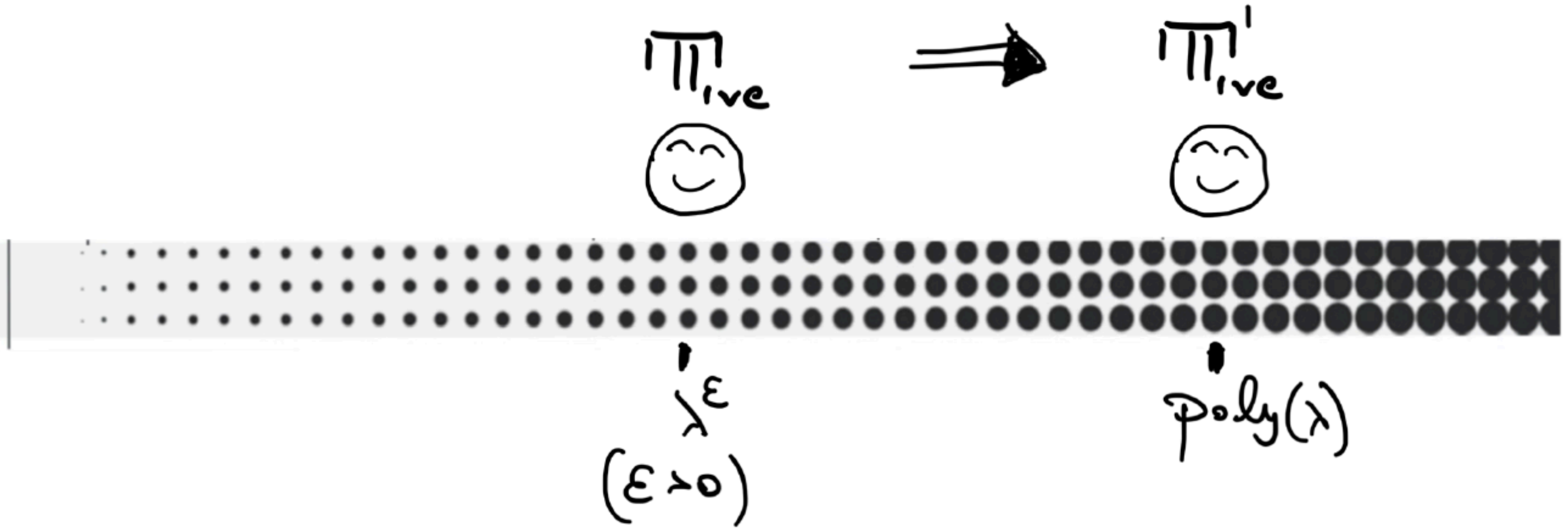
Overhead* for P/V/proof size in Π' is $O_\lambda(\rho)$

*: amortized prover time; applies for linear $T_P(\Pi)$
(if not linear additional polylog overhead).

Our Results
(continued)

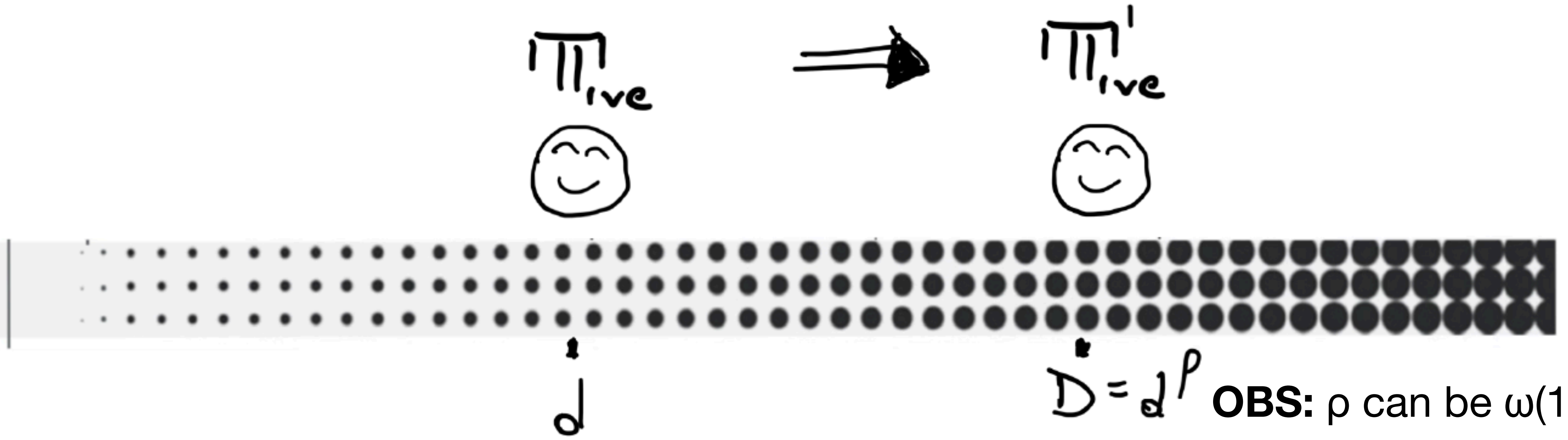
NB: We are interested in:
- black-box lifting results,
- and that preserve performance.





Theorem (sublinear depths):
∃ IVC Π SND at depth λ^{ε} (for some $\varepsilon>0$)
⇒ ∃ IVC Π' SND at arbitrary depth.
Overhead for P/V/proof size in Π' is $O_\lambda(1)$.

Corollary:
∃ IVC SND at O(1) ⇒
∃ IVC Π' SND at depth D = poly.



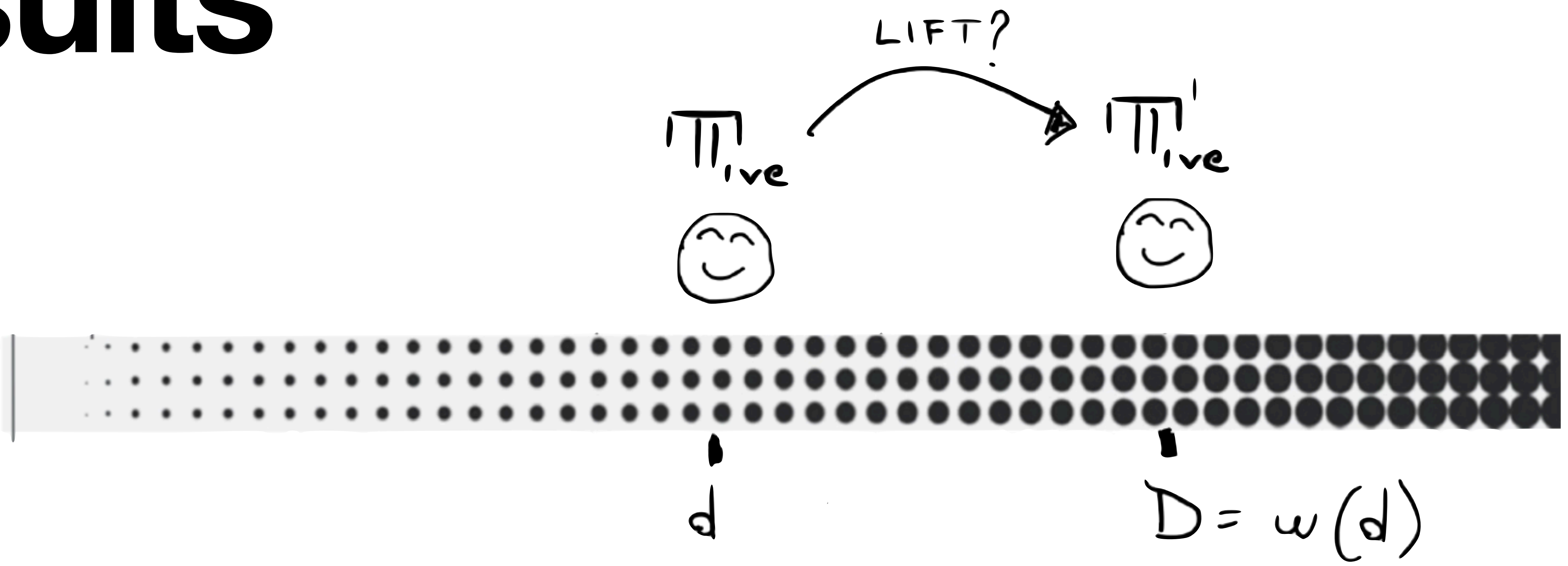
OBS: ρ can be ω(1).

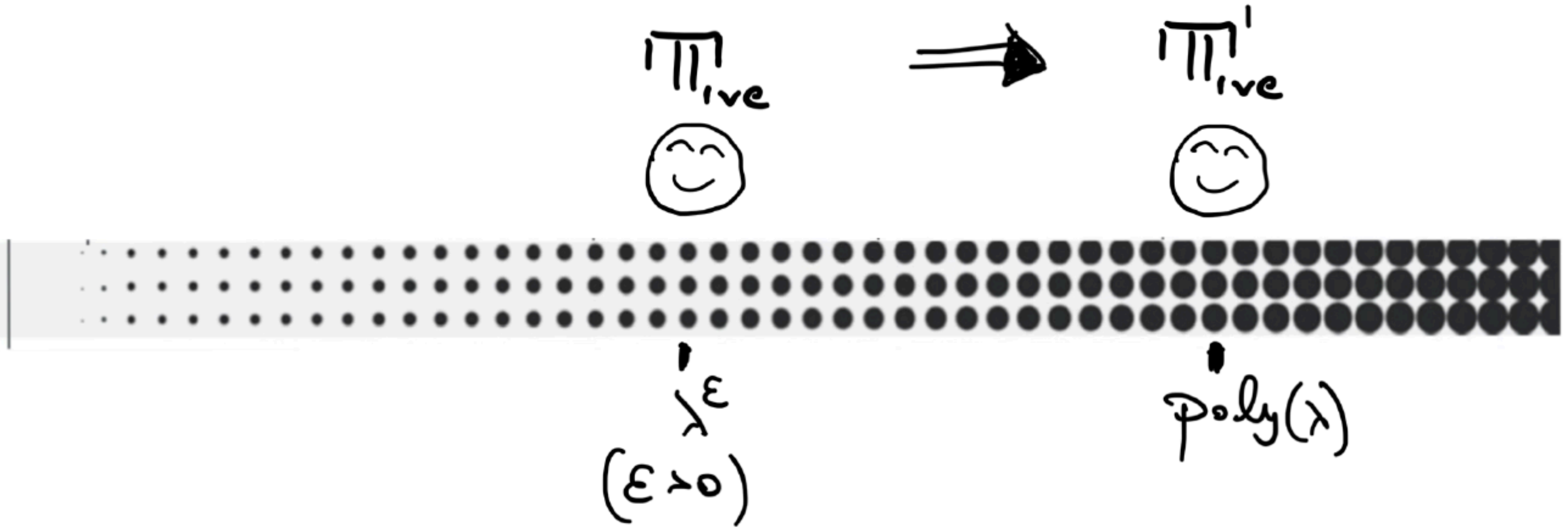
Theorem (general lifting):
∃ IVC Π SND at depth d
⇒ ∃ IVC Π' SND at depth $D = d^{\rho}$.
Overhead* for P/V/proof size in Π' is $O_\lambda(\rho)$

*: amortized prover time; applies for linear $T_P(\Pi)$
(if not linear additional polylog overhead).

Our Results
(continued)

NB: We are interested in:
- black-box lifting results,
- and that preserve performance.





Theorem (sublinear depths):
∃ IVC Π SND at depth λ^ε (for some $\varepsilon>0$)
⇒ ∃ IVC Π' SND at arbitrary depth.

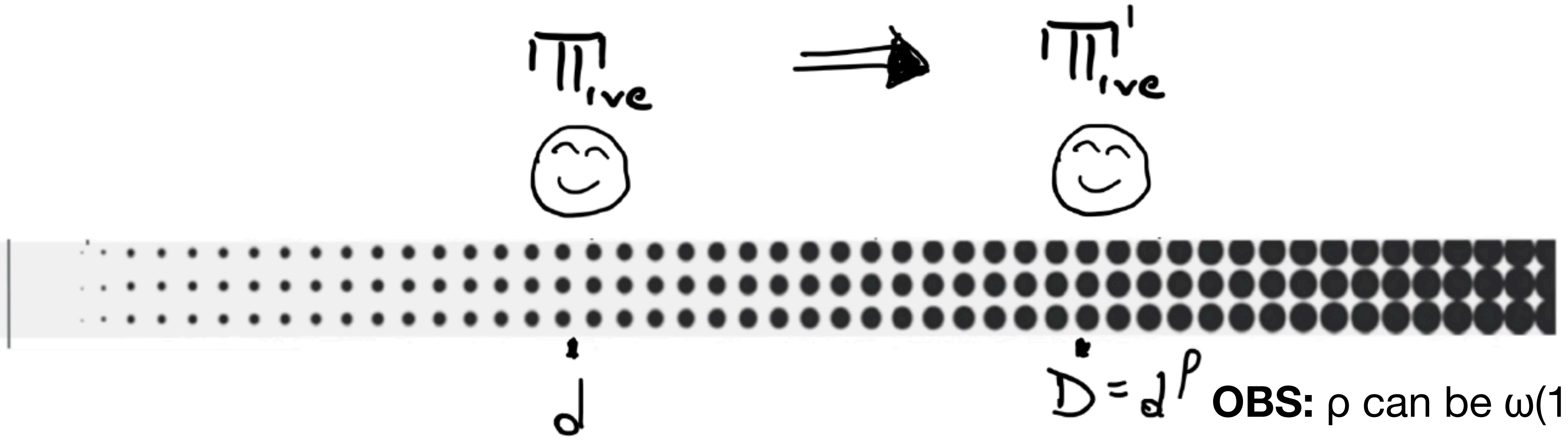
Overhead for P/V/proof size in Π' is $O_\lambda(1)$.

Corollary:

∃ IVC SND at O(1) ⇒
∃ IVC Π' SND at depth D = poly.

↑

Special case: d = O(1); ρ = O(logλ)



OBS: ρ can be ω(1).

Theorem (general lifting):
∃ IVC Π SND at depth d
⇒ ∃ IVC Π' SND at depth D = d^ρ.

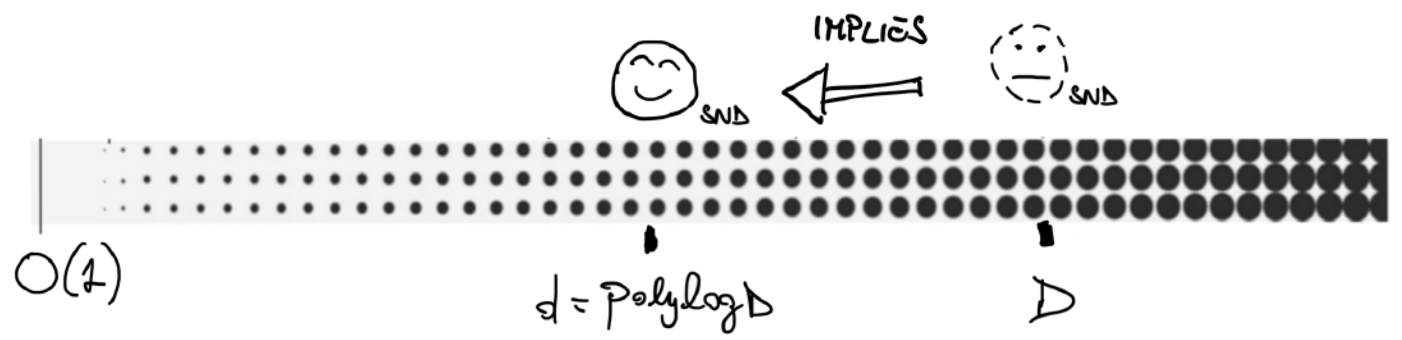
Overhead* for P/V/proof size in Π' is $O_\lambda(\rho)$

*: amortized prover time; applies for linear $T_P(\Pi)$
(if not linear additional polylog overhead).

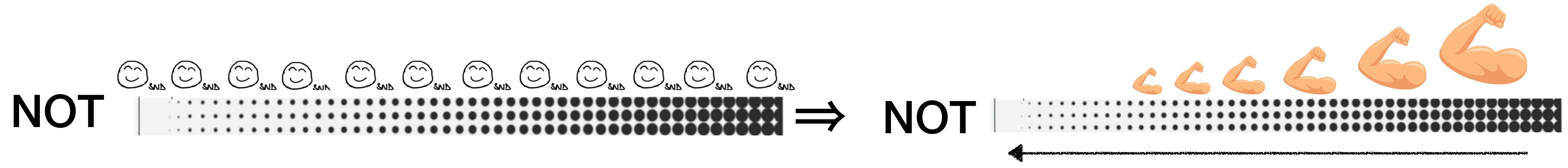
When Do Our Results Apply?

For what security notions do they hold?

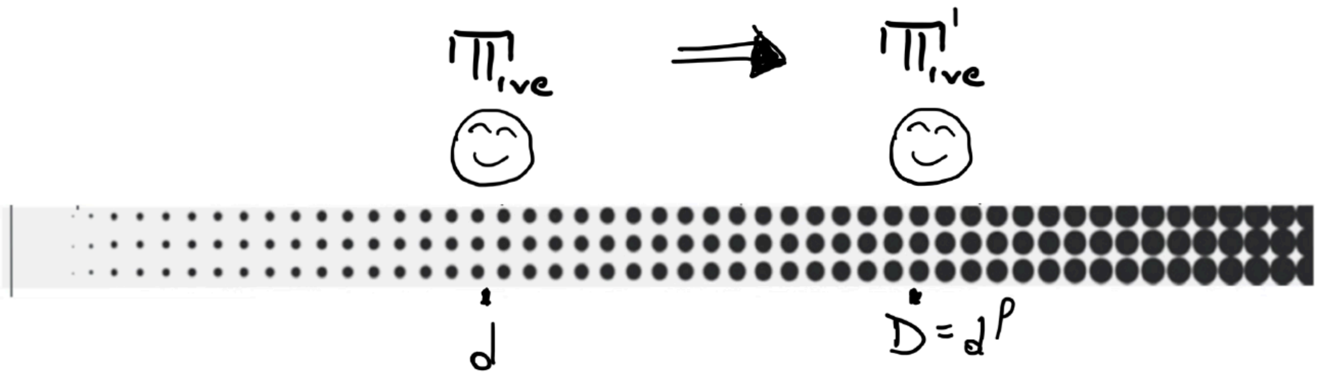
<u>io-sec at D ⇒ sec at d = o(D)</u>



<u>Insecure IVCs cannot exhibit graceful sec. degradation</u>



<u>Black-box lifting with low overhead</u>

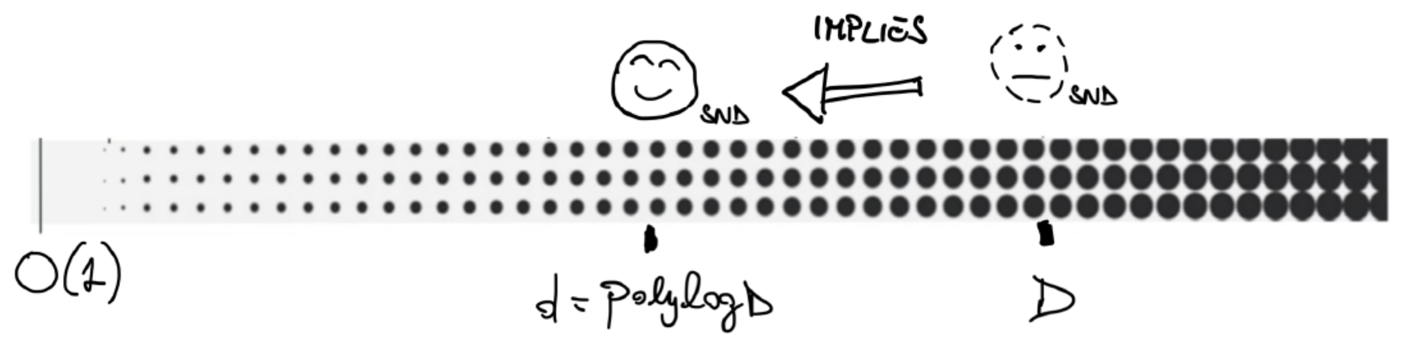


When Do Our Results Apply?

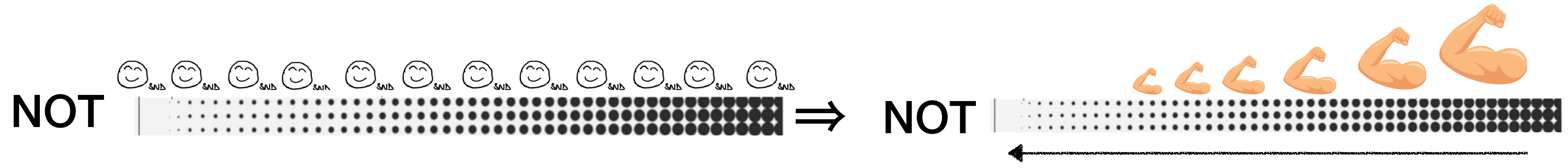
For what security notions do they hold?

- *Soundness for deterministic and non-deterministic computations:*

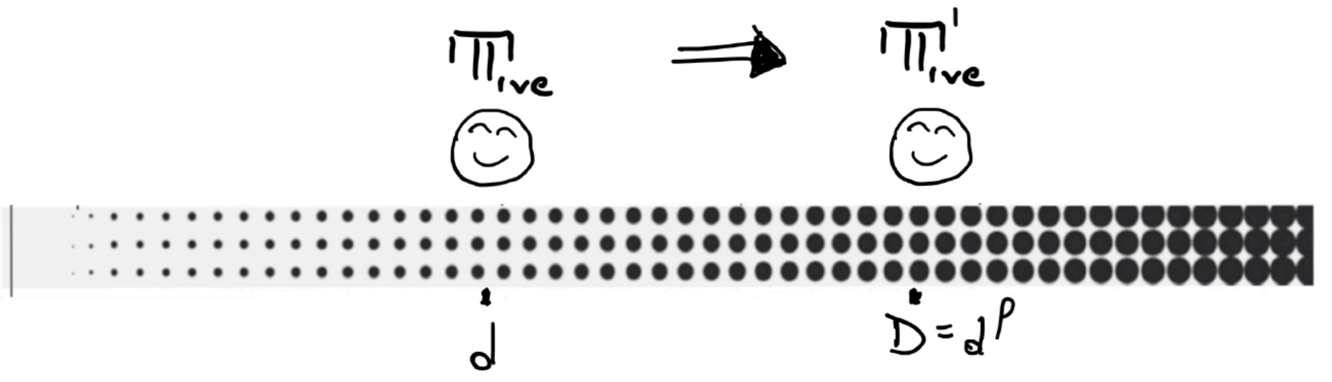
io-sec at D ⇒ sec at d = o(D)



Insecure IVCs cannot exhibit graceful sec. degradation



Black-box lifting with low overhead

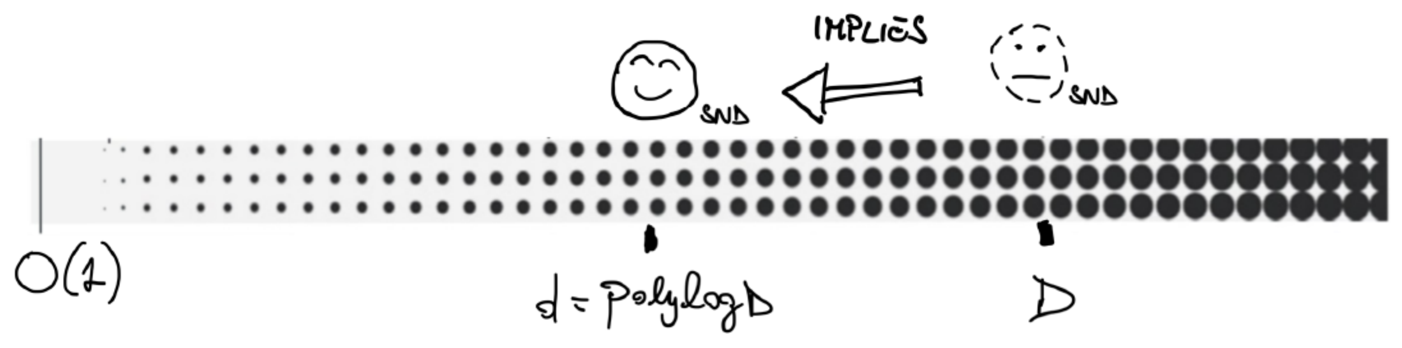


When Do Our Results Apply?

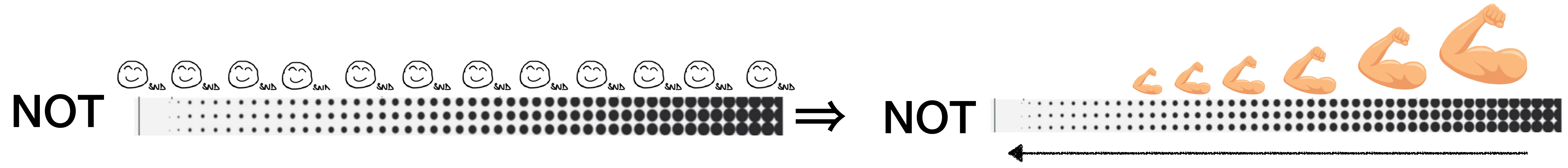
For what security notions do they hold?

- *Soundness for deterministic and non-deterministic computations:*
    - All results apply

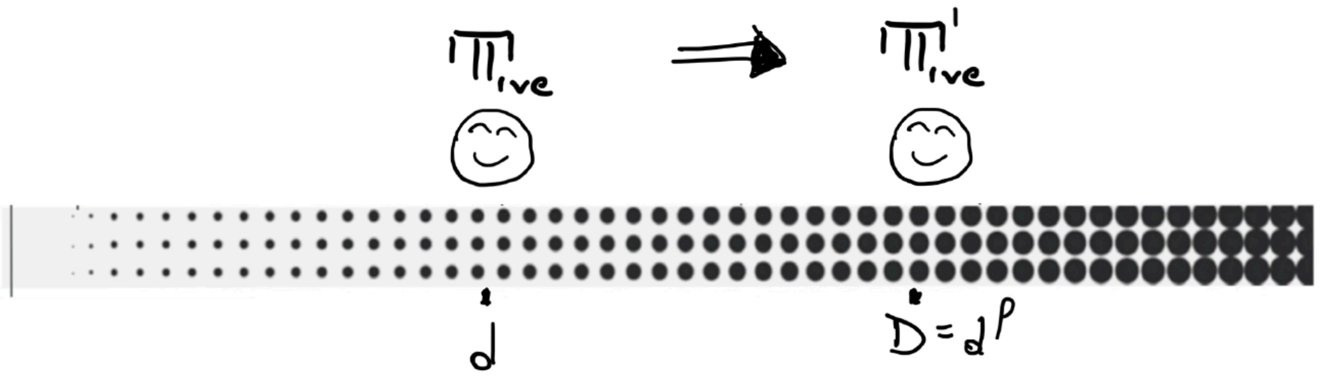
<u>io-sec at D ⇒ sec at d = o(D)</u>



<u>Insecure IVCs cannot exhibit graceful sec. degradation</u>



<u>Black-box lifting with low overhead</u>

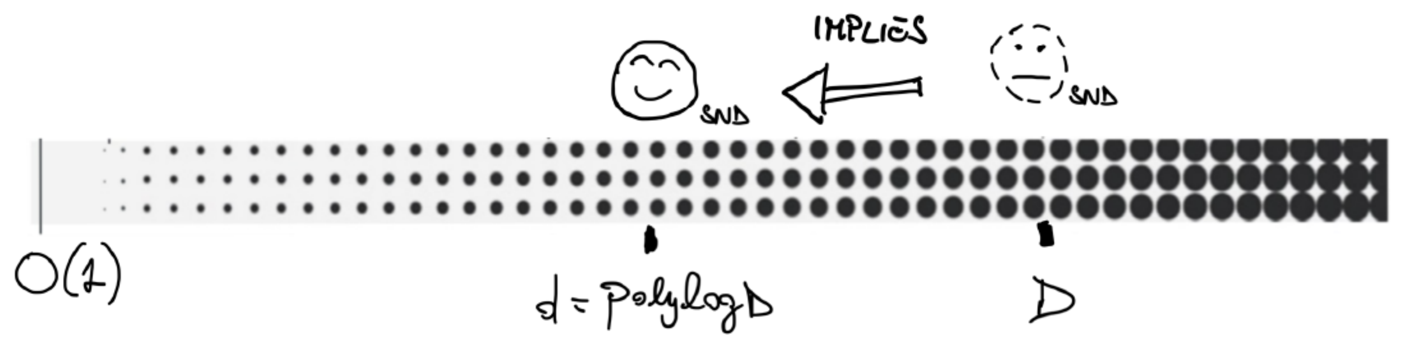


When Do Our Results Apply?

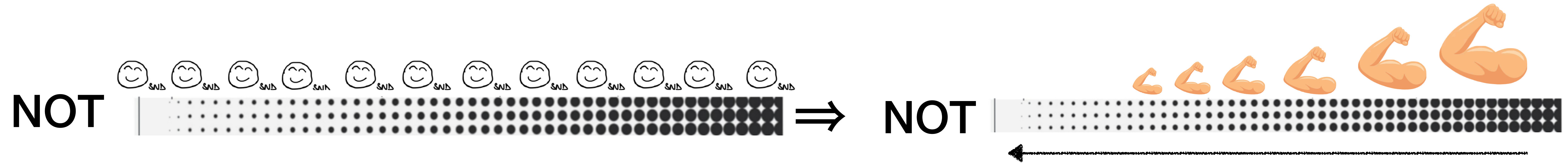
For what security notions do they hold?

- *Soundness for deterministic and non-deterministic computations:*
  - All results apply
- *Evaluation Binding for Incremental Functional Commitments (next slides)*

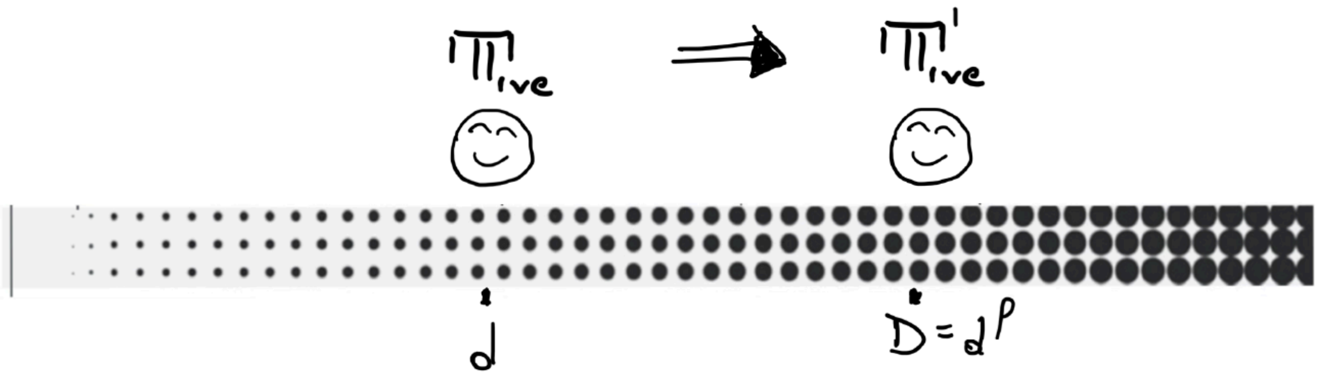
<u>io-sec at D ⇒ sec at d = o(D)</u>



<u>Insecure IVCs cannot exhibit graceful sec. degradation</u>



<u>Black-box lifting with low overhead</u>

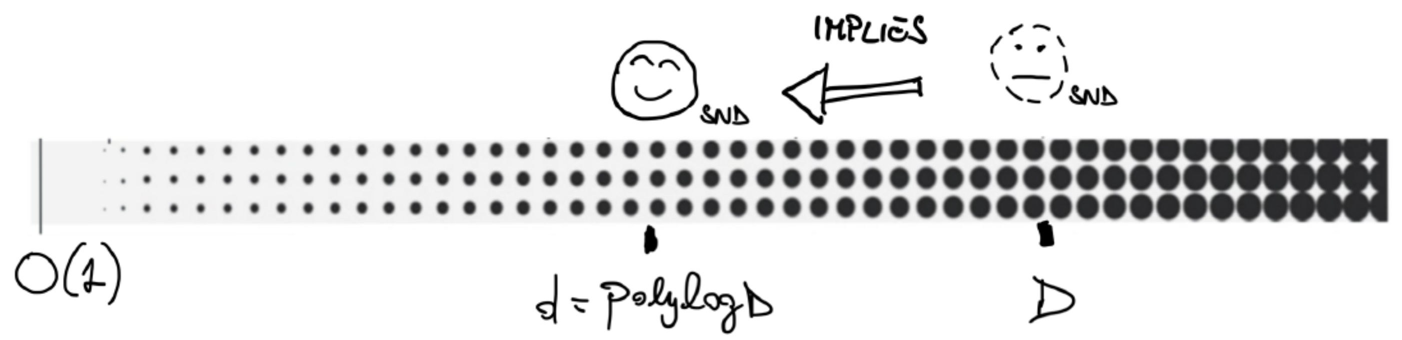


When Do Our Results Apply?

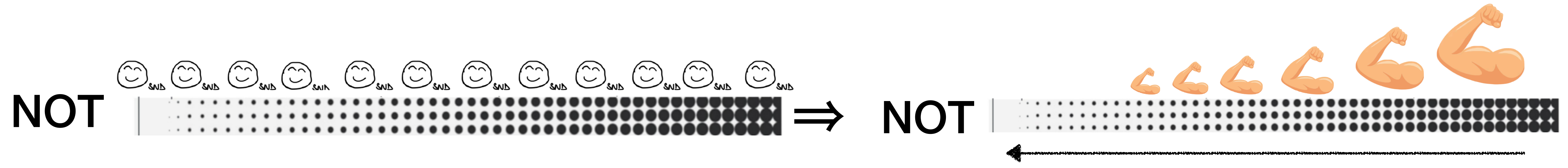
For what security notions do they hold?

- *Soundness for deterministic and non-deterministic computations:*
  - All results apply
- *Evaluation Binding for Incremental Functional Commitments (next slides)*
  - All results apply

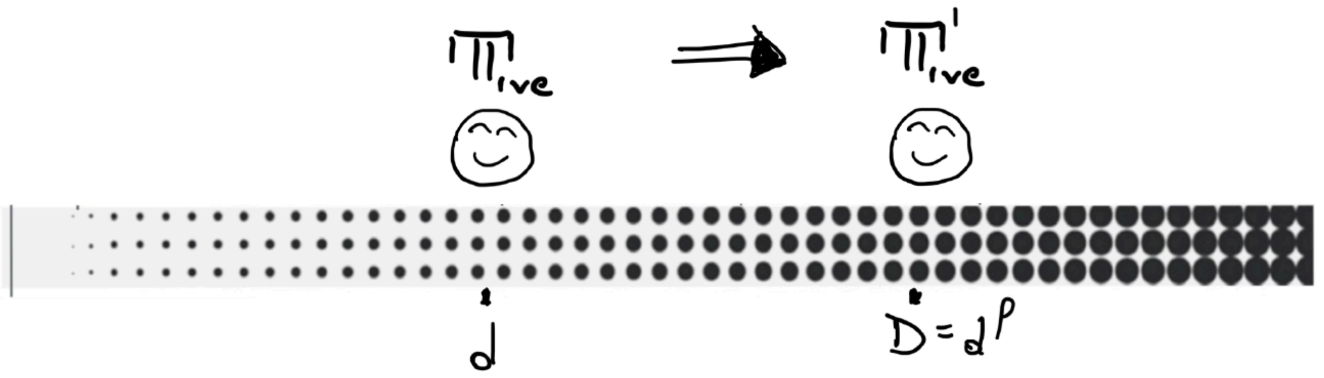
<u>io-sec at D ⇒ sec at d = o(D)</u>



<u>Insecure IVCs cannot exhibit graceful sec. degradation</u>



<u>Black-box lifting with low overhead</u>

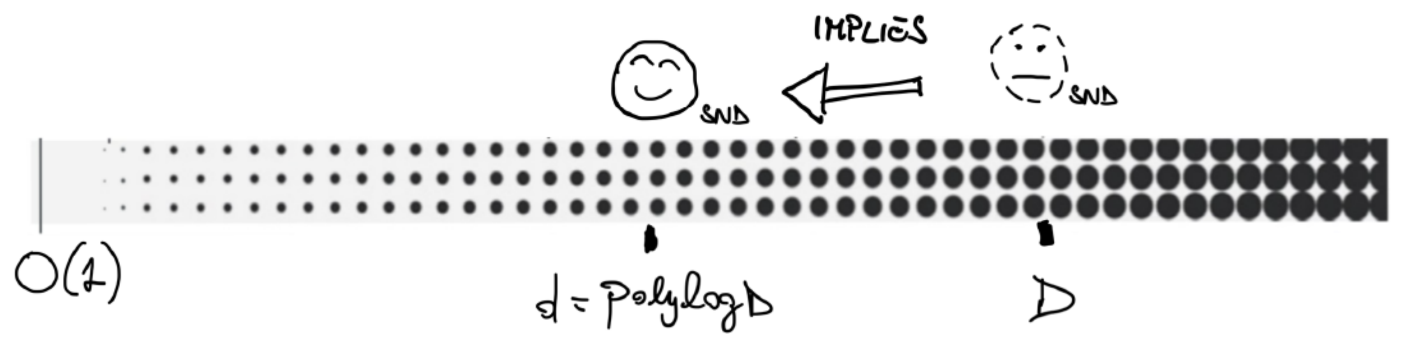


When Do Our Results Apply?

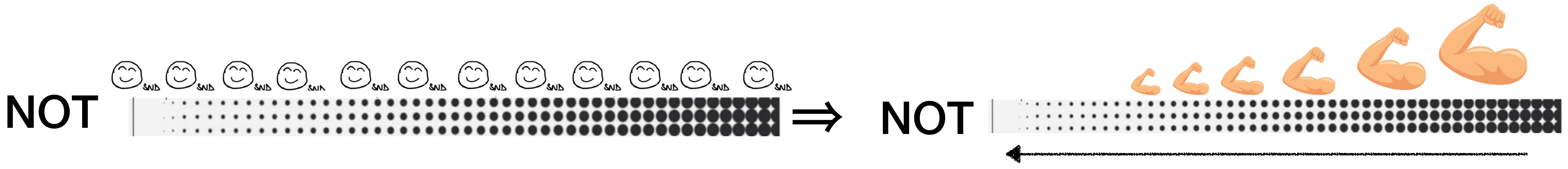
For what security notions do they hold?

- *Soundness for deterministic and non-deterministic computations:*
  - All results apply
- *Evaluation Binding for Incremental Functional Commitments (next slides)*
  - All results apply
- *Extractability*

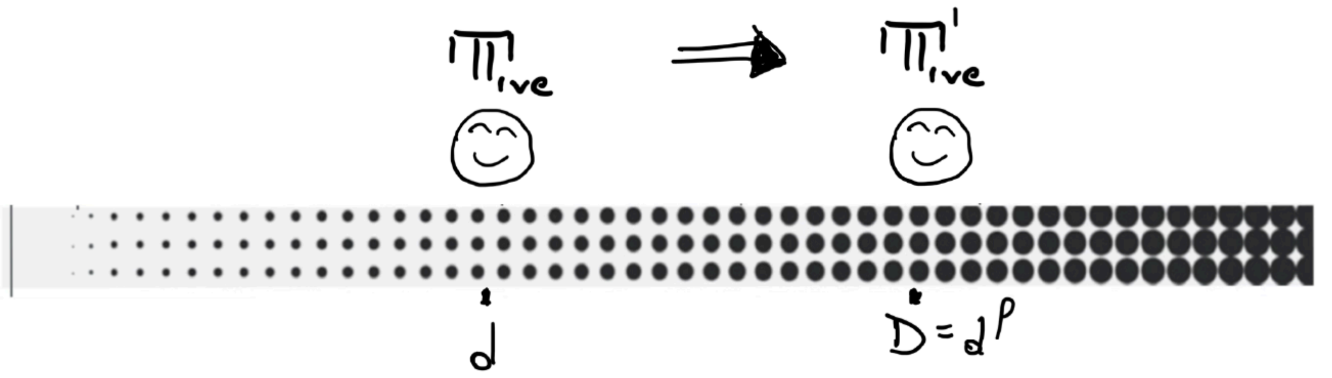
io-sec at D ⇒ sec at d = o(D)



Insecure IVCs cannot exhibit graceful sec. degradation



Black-box lifting with low overhead

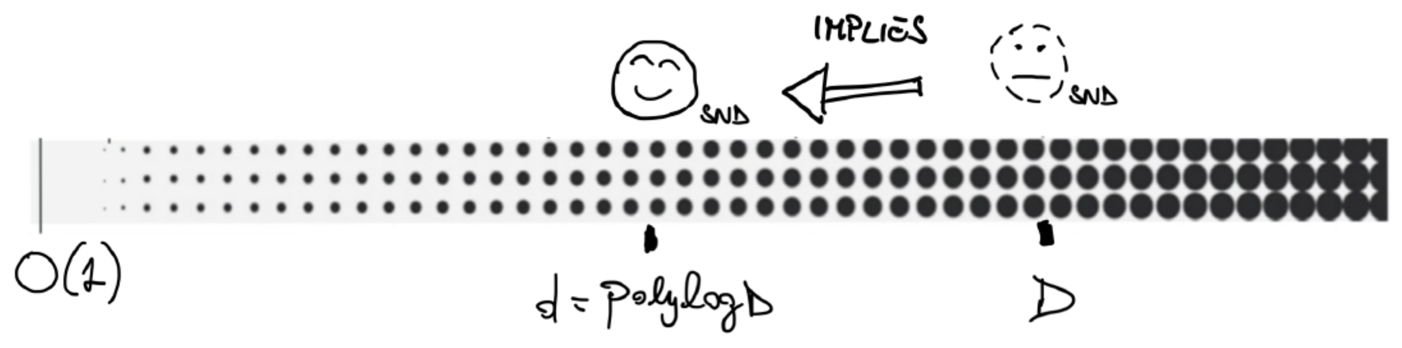


When Do Our Results Apply?

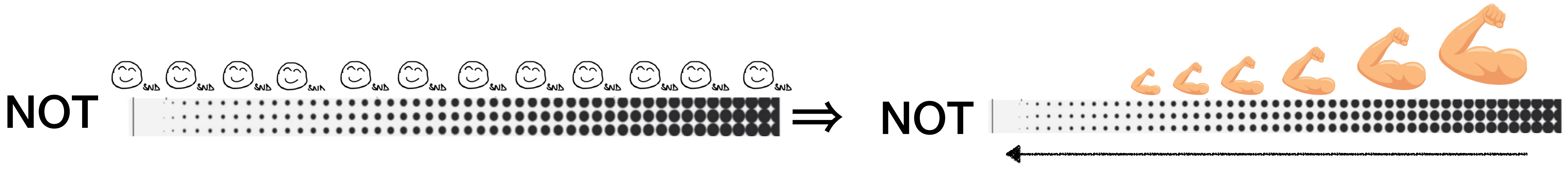
For what security notions do they hold?

- *Soundness for deterministic and non-deterministic computations:*
  - All results apply
- *Evaluation Binding for Incremental Functional Commitments (next slides)*
  - All results apply
- *Extractability*
  - [io-sec at D ⇒ sec at d = o(D)] **+ [**black-box lifting], both apply

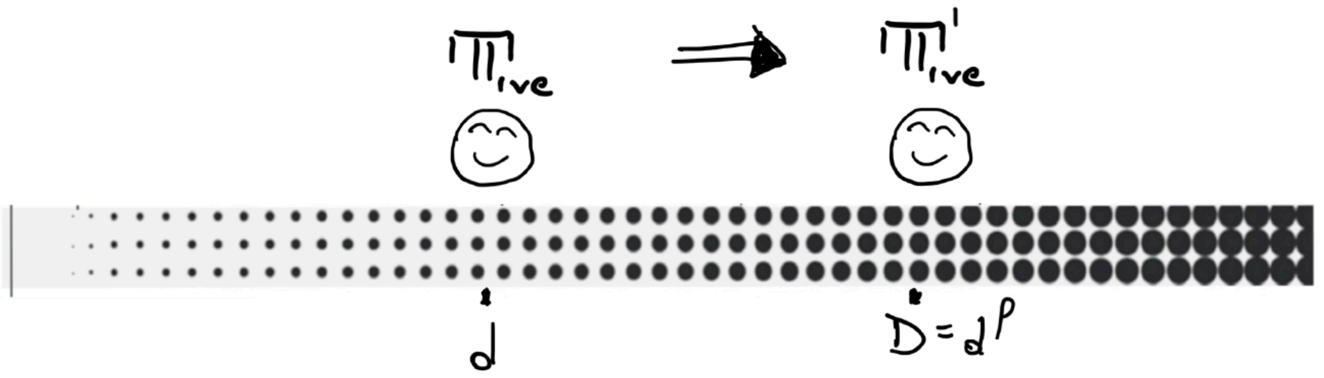
io-sec at D ⇒ sec at d = o(D)



Insecure IVCs cannot exhibit graceful sec. degradation



Black-box lifting with low overhead

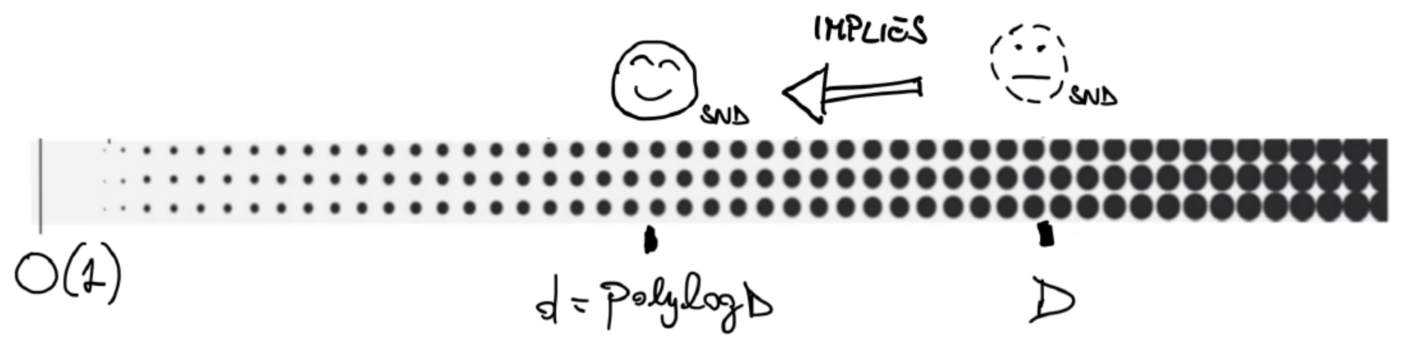


When Do Our Results Apply?

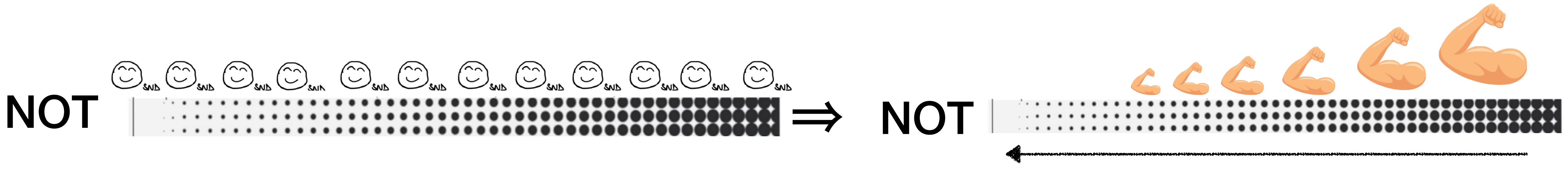
For what security notions do they hold?

- *Soundness for deterministic and non-deterministic computations:*
  - All results apply
- *Evaluation Binding for Incremental Functional Commitments (next slides)*
  - All results apply
- *Extractability*
  - [io-sec at D ⇒ sec at d = o(D)] **+ [**black-box lifting], both apply
  - the result on graceful sec. degradation does not apply mostly because definitions do not translate immediately.

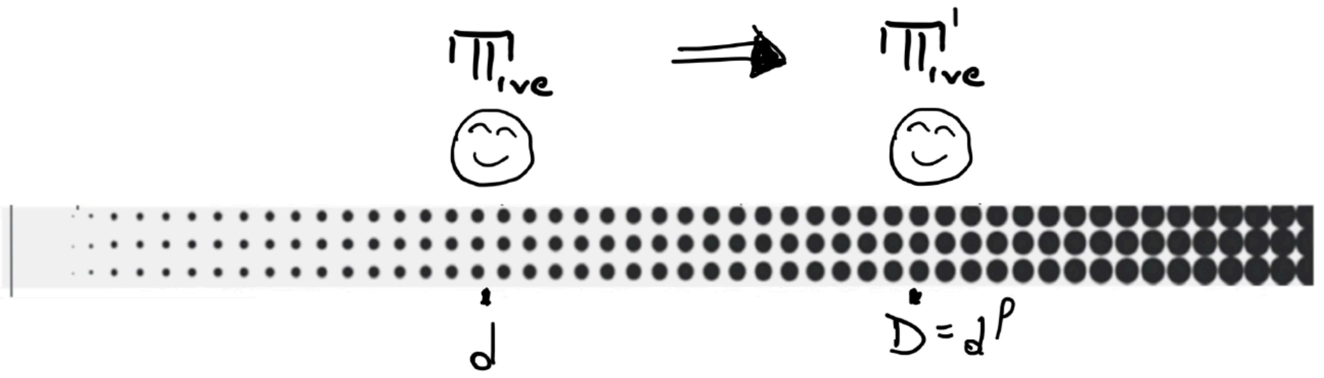
io-sec at D ⇒ sec at d = o(D)



Insecure IVCs cannot exhibit graceful sec. degradation



Black-box lifting with low overhead



New Notion: Incremental Functional Commitments

What are Functional Commitments? (FC)

Server (Prover)

Client (Verifier)

Credit to Dario Fiore for the graphics in the center.

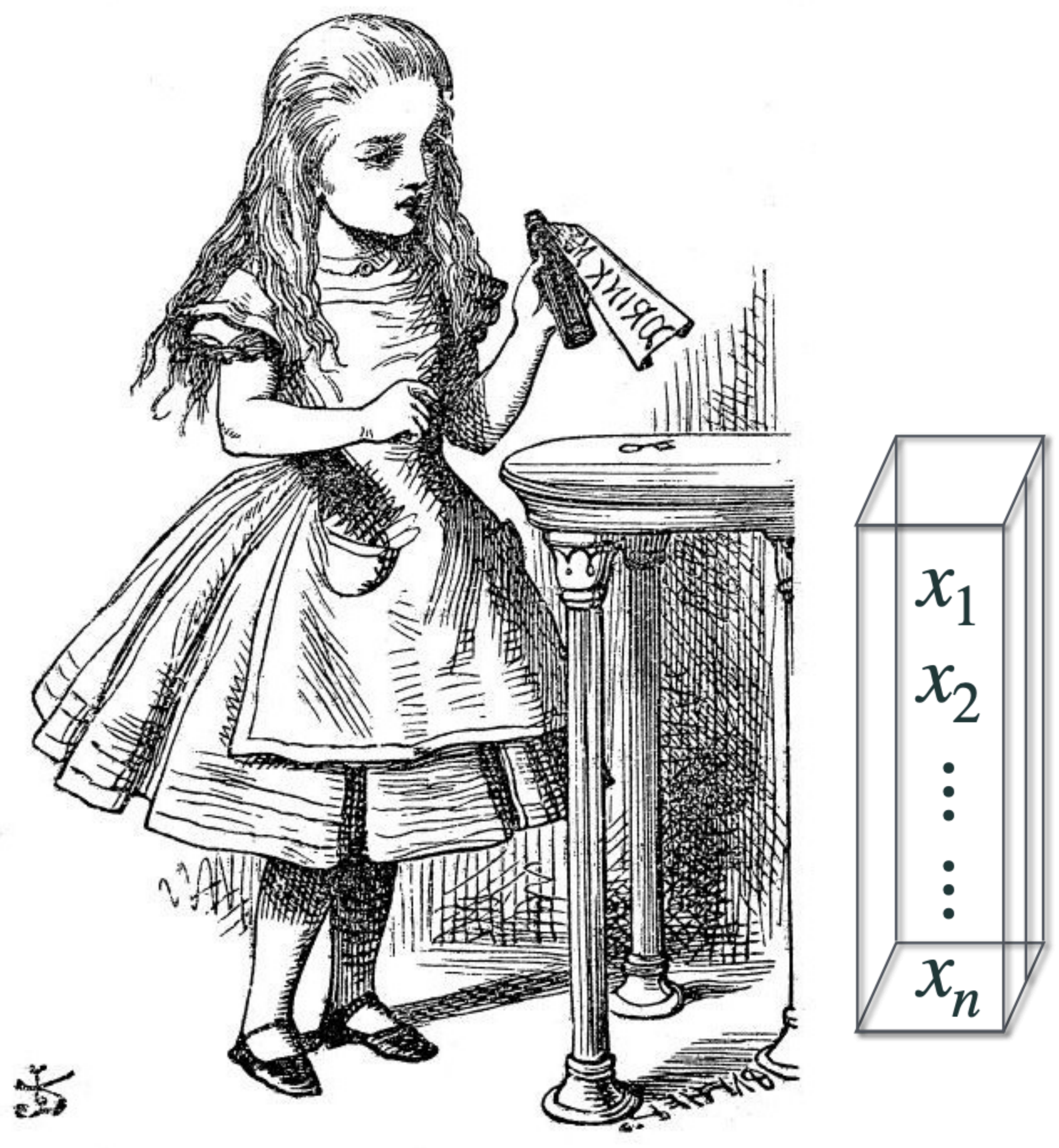
What are Functional Commitments? (FC)

Server (Prover)

Client (Verifier)

Credit to Dario Fiore for the graphics in the center.

What are Functional Commitments? (FC)

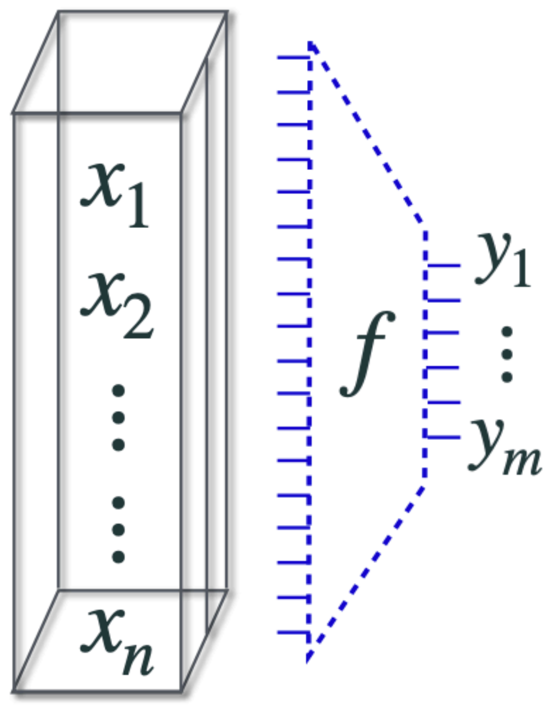


Server (Prover)

Client (Verifier)

Credit to Dario Fiore for the graphics in the center.

What are Functional Commitments? (FC)

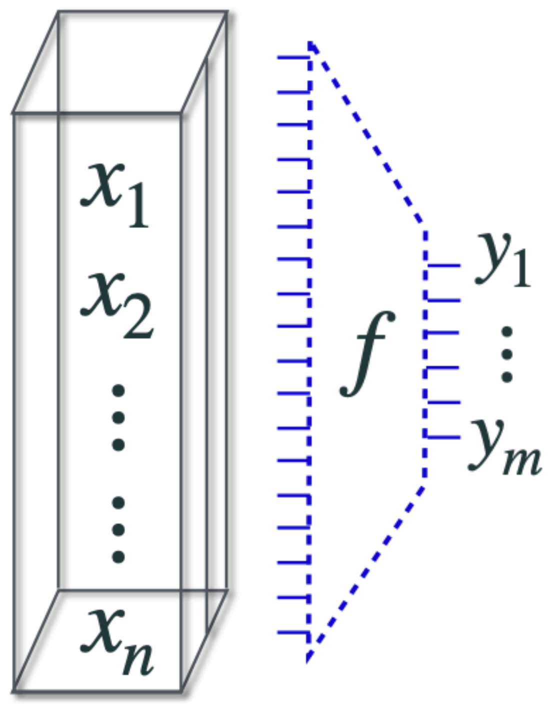


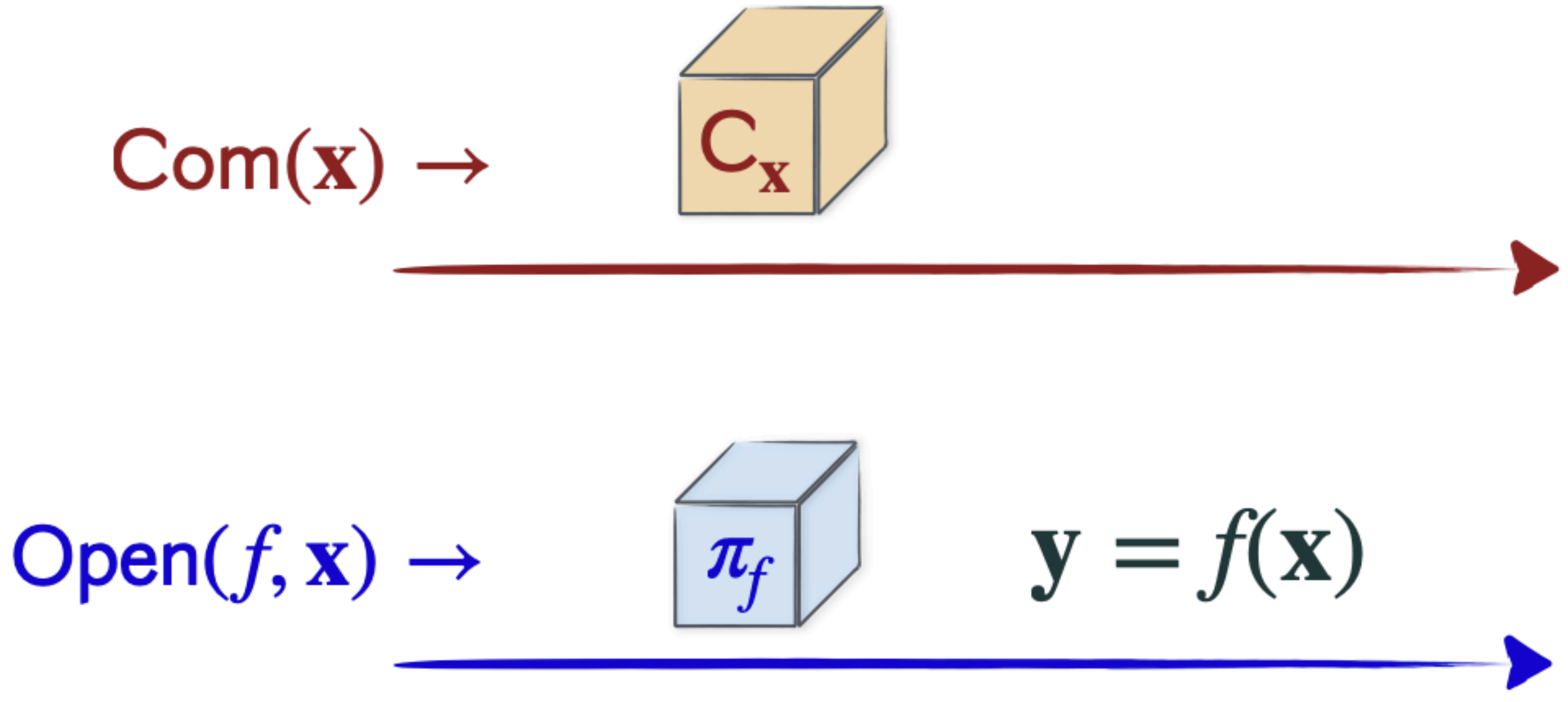
Server (Prover)

Client (Verifier)

Credit to Dario Fiore for the graphics in the center.

What are Functional Commitments? (FC)



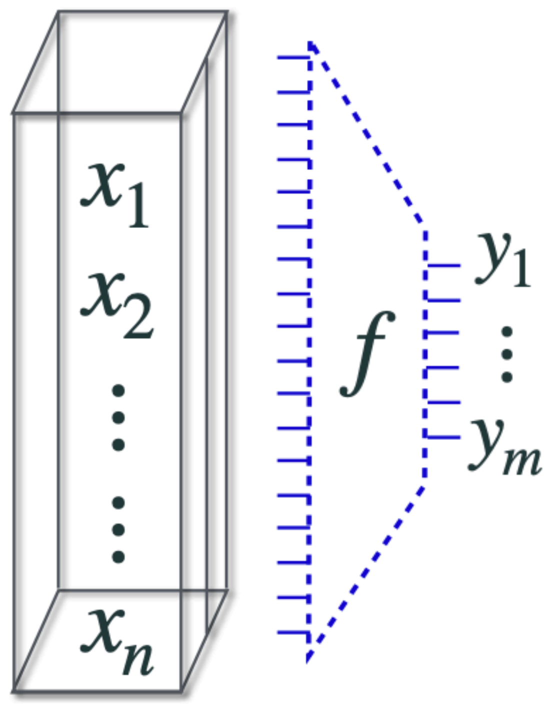


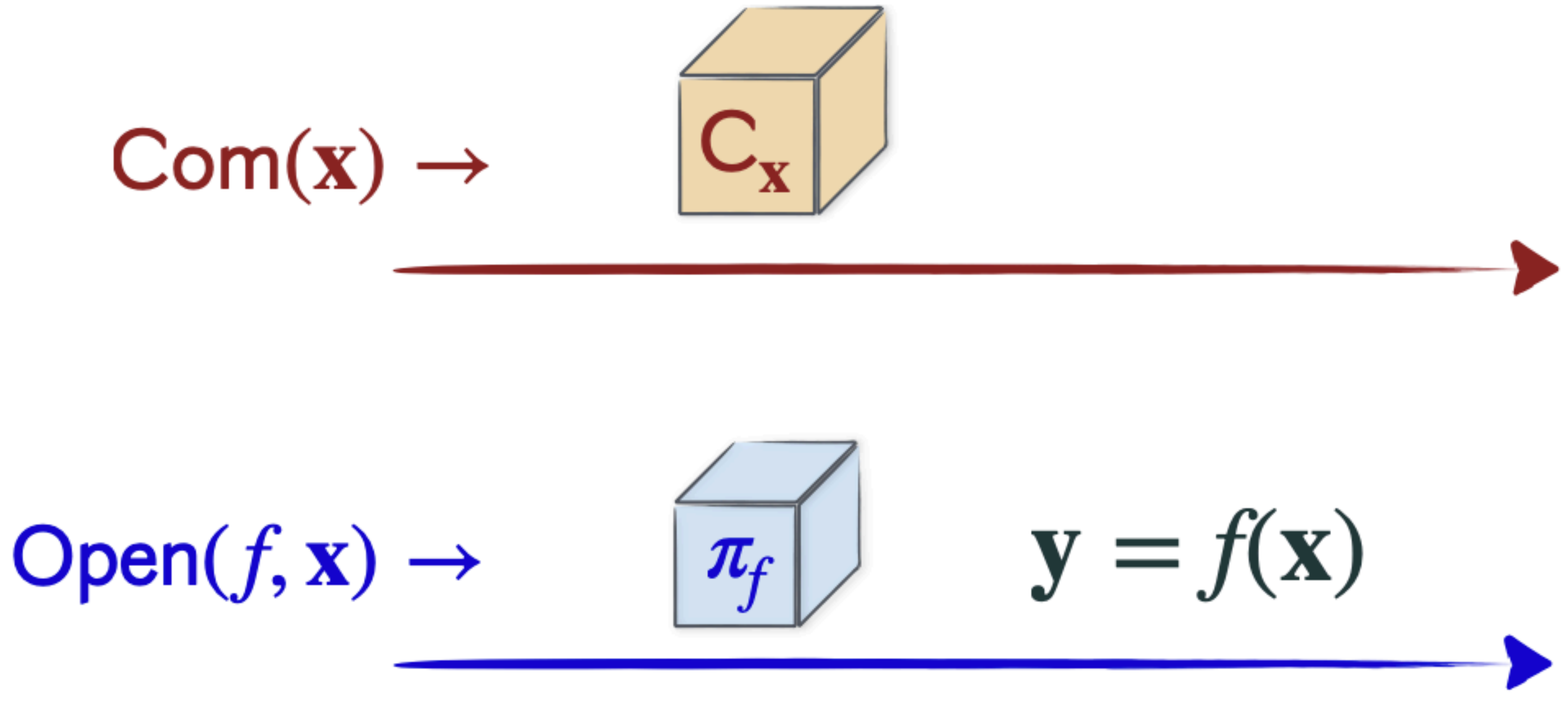
Server (Prover)

Client (Verifier)

Credit to Dario Fiore for the graphics in the center.

What are Functional Commitments? (FC)





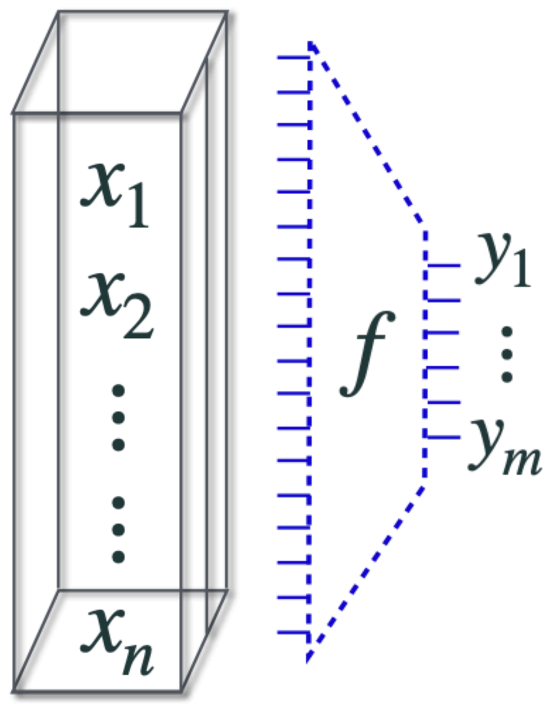
Server (Prover)

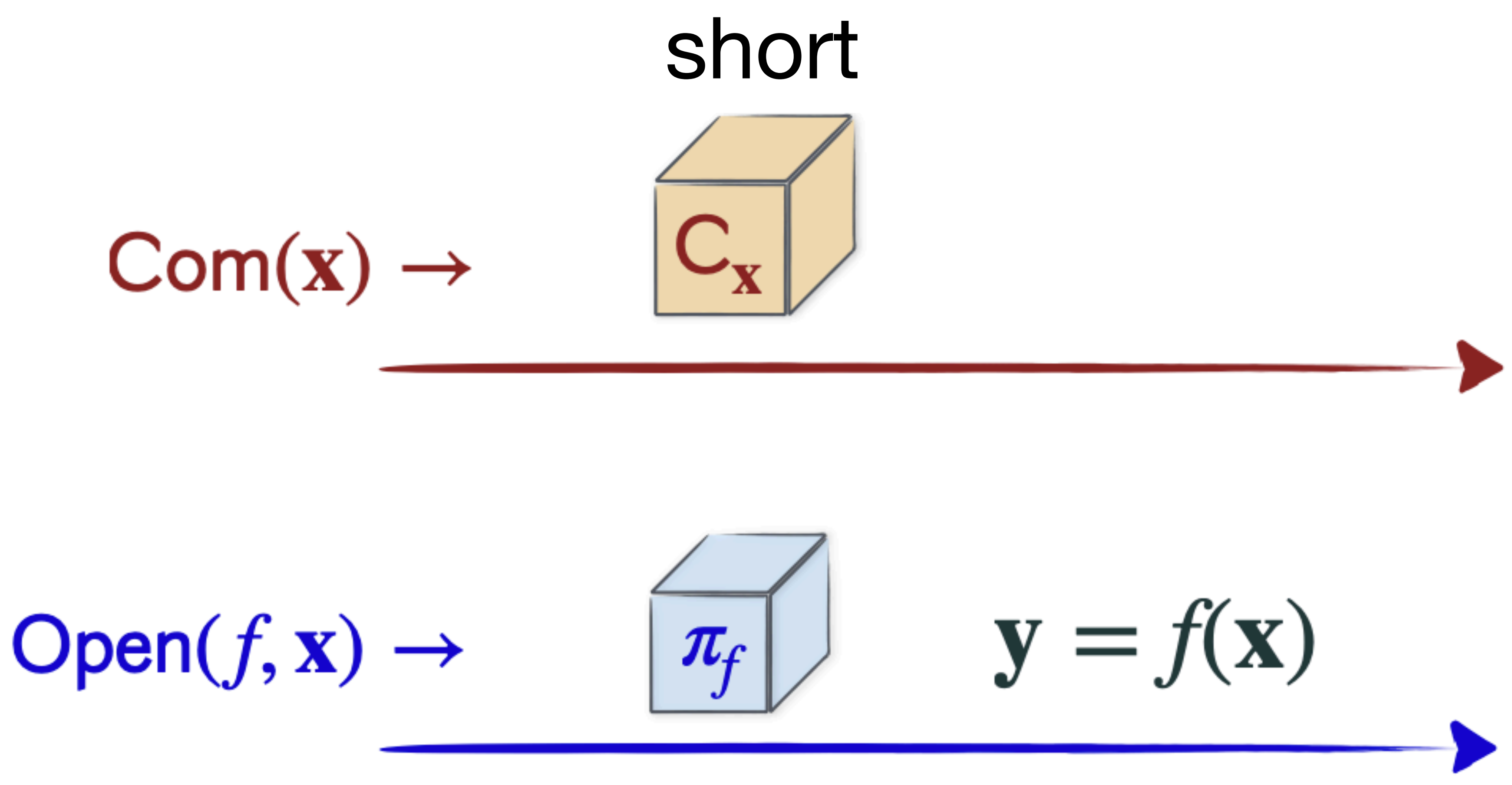
Client (Verifier)

$\mathsf{Ver}(\mathsf{C}_{\mathbf{x}}, f, \mathbf{y}, \pi_f) \stackrel{?}{=} 1$

Credit to Dario Fiore for the graphics in the center.

What are Functional Commitments? (FC)





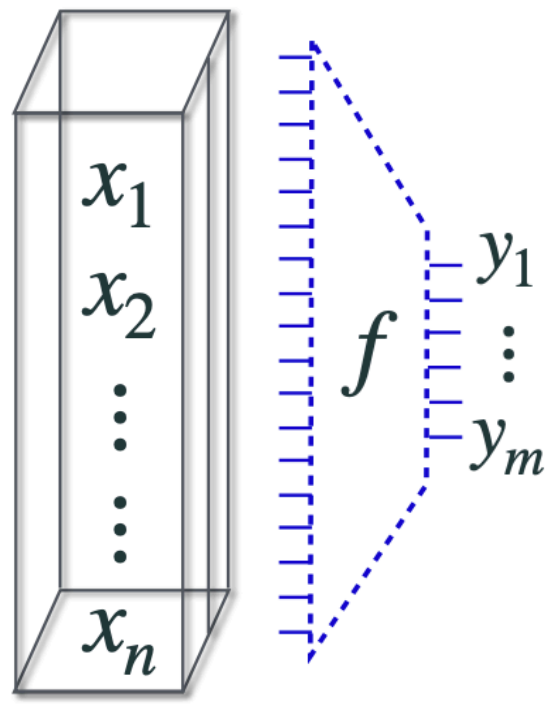
Server (Prover)

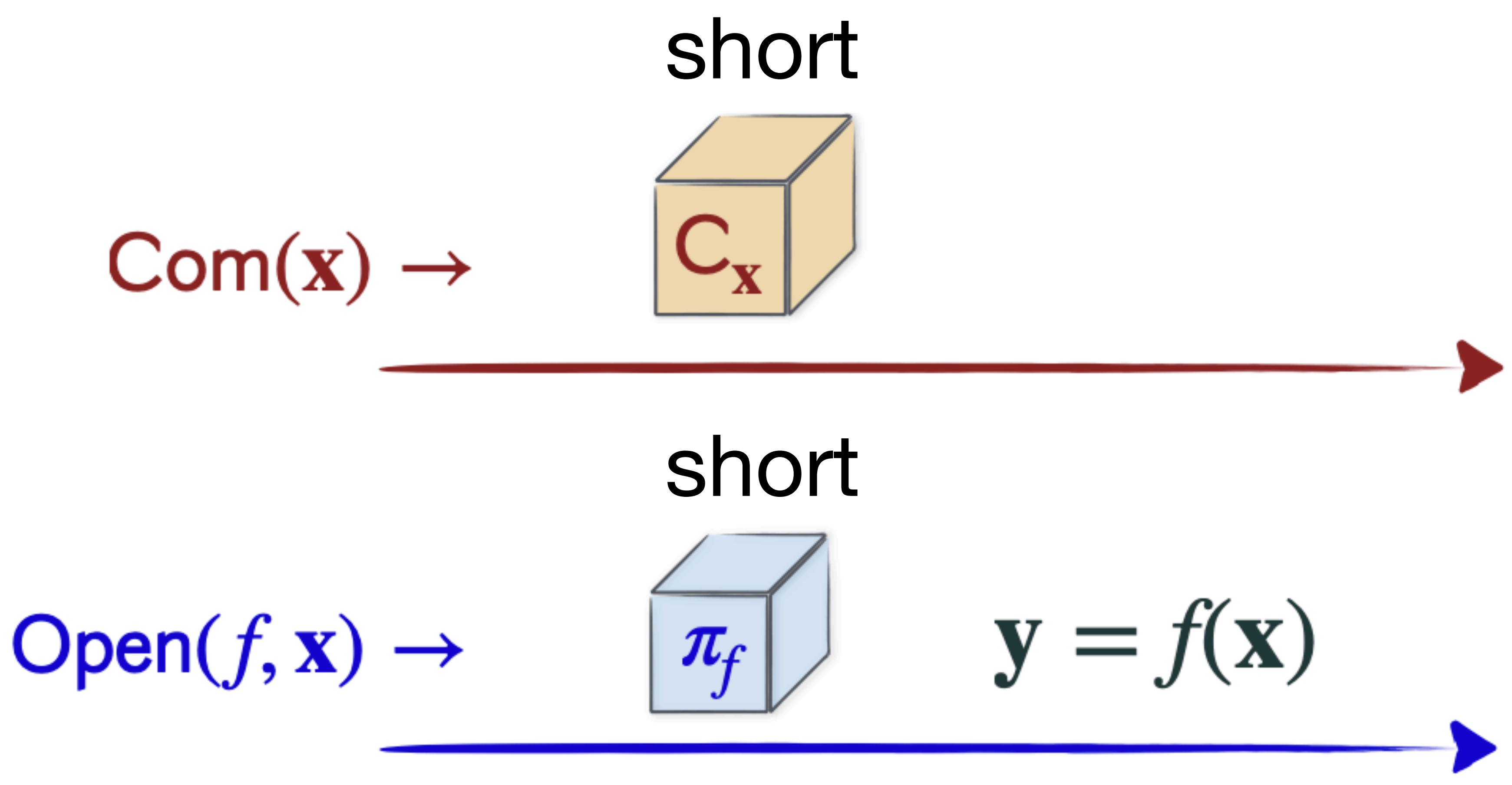
Client (Verifier)

$\mathsf{Ver}(\mathsf{C}_{\mathbf{x}}, f, \mathbf{y}, \pi_f) \stackrel{?}{=} 1$

Credit to Dario Fiore for the graphics in the center.

What are Functional Commitments? (FC)





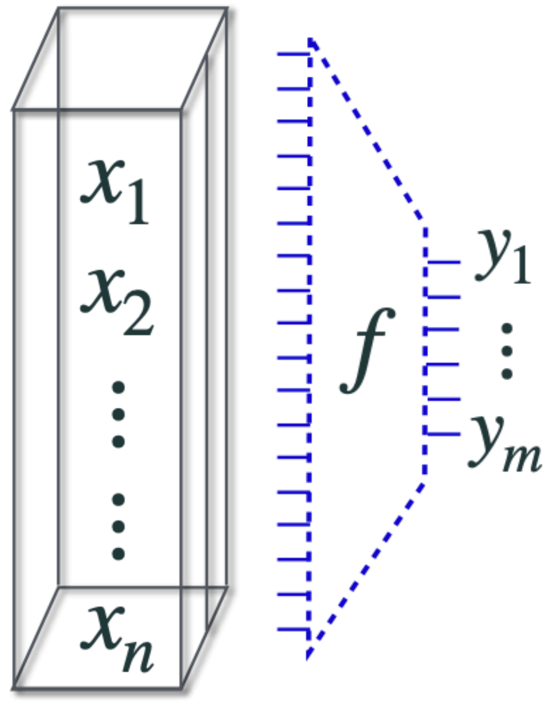
Server (Prover)

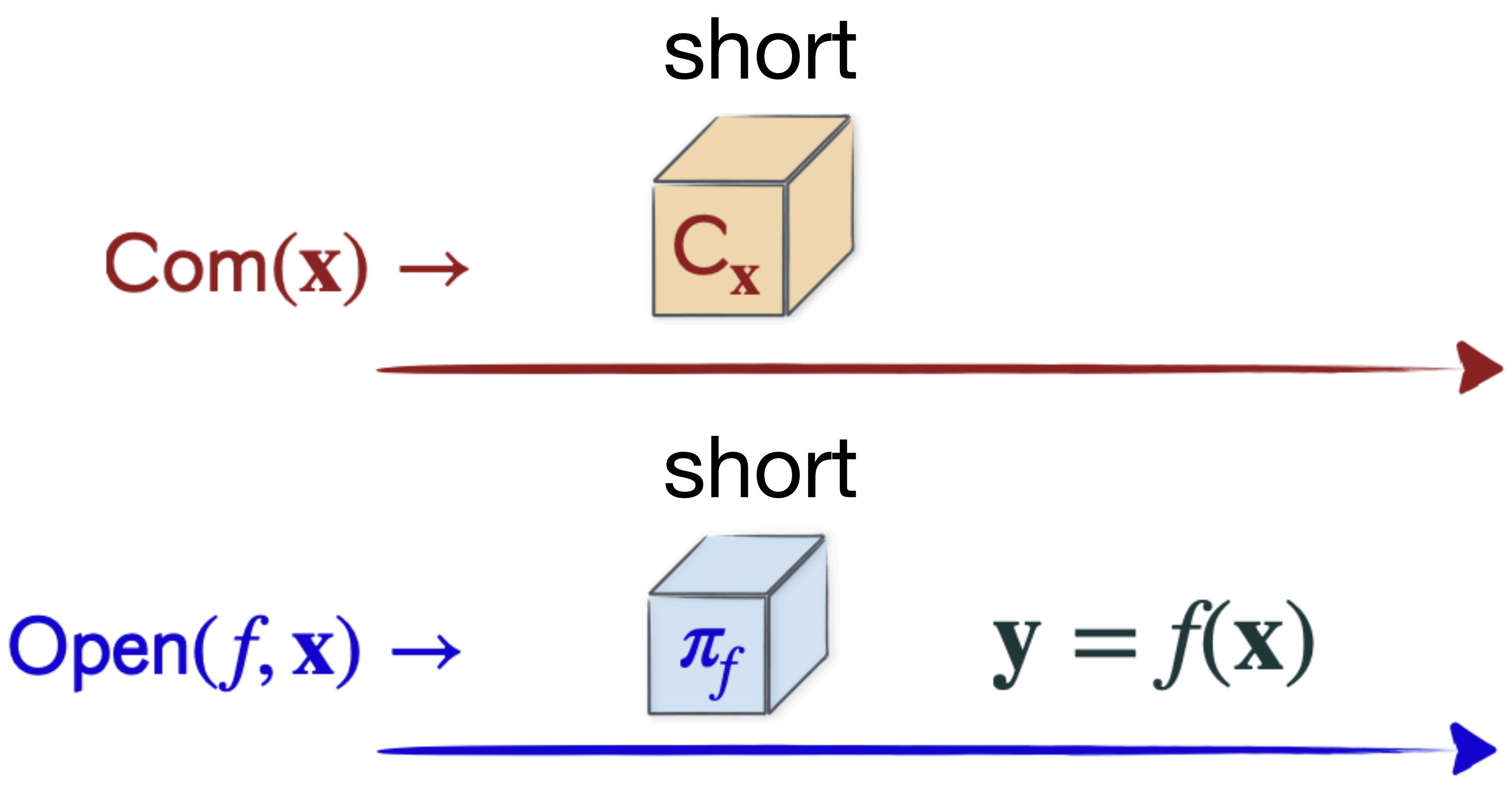
Client (Verifier)

Ver($C_{\mathbf{x}}$, f, $\mathbf{y}$, π_f) $\stackrel{?}{=} 1$

Credit to Dario Fiore for the graphics in the center.

What are Functional Commitments? (FC)





Server (Prover)

Client (Verifier)

$\mathsf{Ver}(C_{\mathbf{x}}, f, \mathbf{y}, \pi_f) \stackrel{?}{=} 1$

Functional commitments generalize polynomial and vector commitments.

Credit to Dario Fiore for the graphics in the center.

Security of Functional Commitments

Evaluation Binding

Malicious Prover

Client (Verifier)

Credit to Dario Fiore for the graphics in the center.

Security of Functional Commitments

Evaluation Binding

Malicious Prover

Client (Verifier)

No adversary can succeed in providing inconsistent valid-looking outputs.

Credit to Dario Fiore for the graphics in the center.

Security of Functional Commitments

Evaluation Binding

$\mathsf{C}_{\mathbf{x}} \quad f, \quad \pi_f, \mathbf{y}, \quad \pi'_f, \mathbf{y}'$ $\longrightarrow$

$\mathbf{y} \neq \mathbf{y}'$

Malicious Prover

Client (Verifier)

No adversary can succeed in providing inconsistent valid-looking outputs.

Credit to Dario Fiore for the graphics in the center.

Security of Functional Commitments

Evaluation Binding

Malicious Prover

$\mathsf{C}_{\mathbf{x}}$ $\quad f, \quad \pi_f, \mathbf{y}, \quad \pi'_f, \mathbf{y}'$

$\mathbf{y} \neq \mathbf{y}'$

$\mathsf{Ver}(\mathsf{C}_{\mathbf{x}}, f, \mathbf{y}, \pi_f) = 1$

$\mathsf{Ver}(\mathsf{C}_{\mathbf{x}}, f, \mathbf{y}', \pi'_f) = 1$

Client (Verifier)

No adversary can succeed in providing inconsistent valid-looking outputs.

Credit to Dario Fiore for the graphics in the center.

Security of Functional Commitments

Evaluation Binding

$\mathsf{C}_{\mathbf{x}} \quad f, \quad \pi_f, \mathbf{y}, \quad \pi'_f, \mathbf{y}'$ →

$\mathbf{y} \neq \mathbf{y}'$

$\mathsf{Ver}(\mathsf{C}_{\mathbf{x}}, f, \mathbf{y}, \pi_f) = 1$

$\mathsf{Ver}(\mathsf{C}_{\mathbf{x}}, f, \mathbf{y}', \pi'_f) = 1$

Malicious Prover

Client (Verifier)

No adversary can succeed in providing inconsistent valid-looking outputs.

NB: intuitively stronger than deterministic and non-deterministic soundness (but weaker than extractability).

Credit to Dario Fiore for the graphics in the center.

Incrementality in Functional Commitments

Motivation

Incrementality in Functional Commitments

Motivation

Our goal:

committing and (especially) proving should be doable in an incremental manner.

Incrementality in Functional Commitments

Motivation

Our goal:

committing and (especially) proving should be doable in an incremental manner.

Example applications:

"streaming" polynomial commitments and neural network evaluation.

Incrementality in Functional Commitments

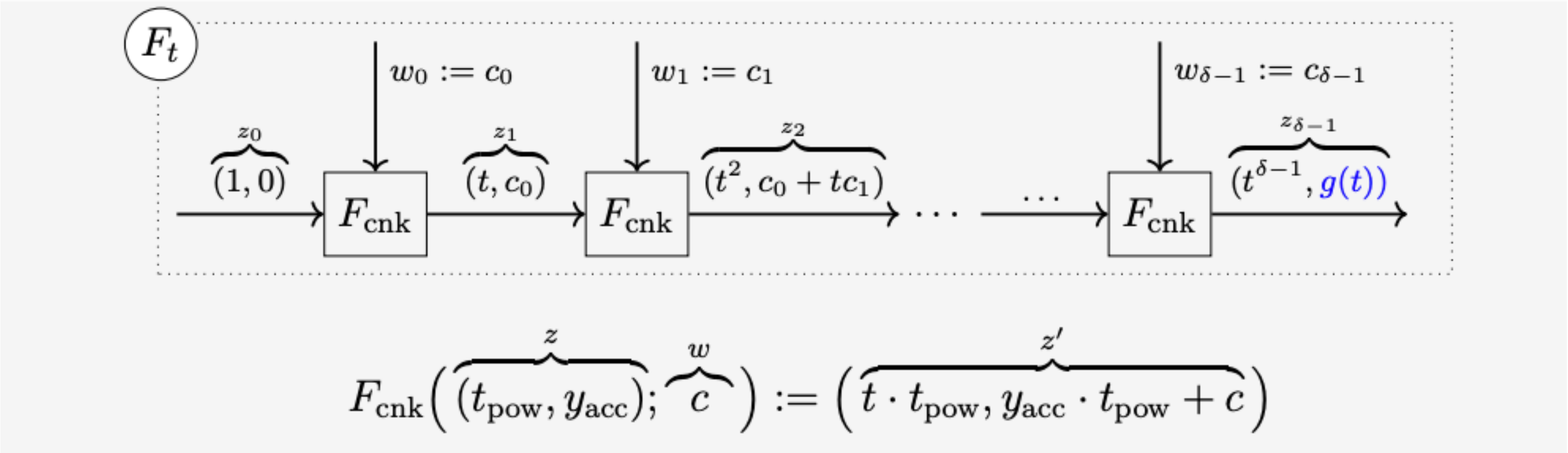
Motivation

Our goal:

committing and (especially) proving should be doable in an incremental manner.

Example applications:

“streaming” polynomial commitments and neural network evaluation.



“Incremental” polynomial

Incrementality in Functional Commitments

Motivation

Our goal:

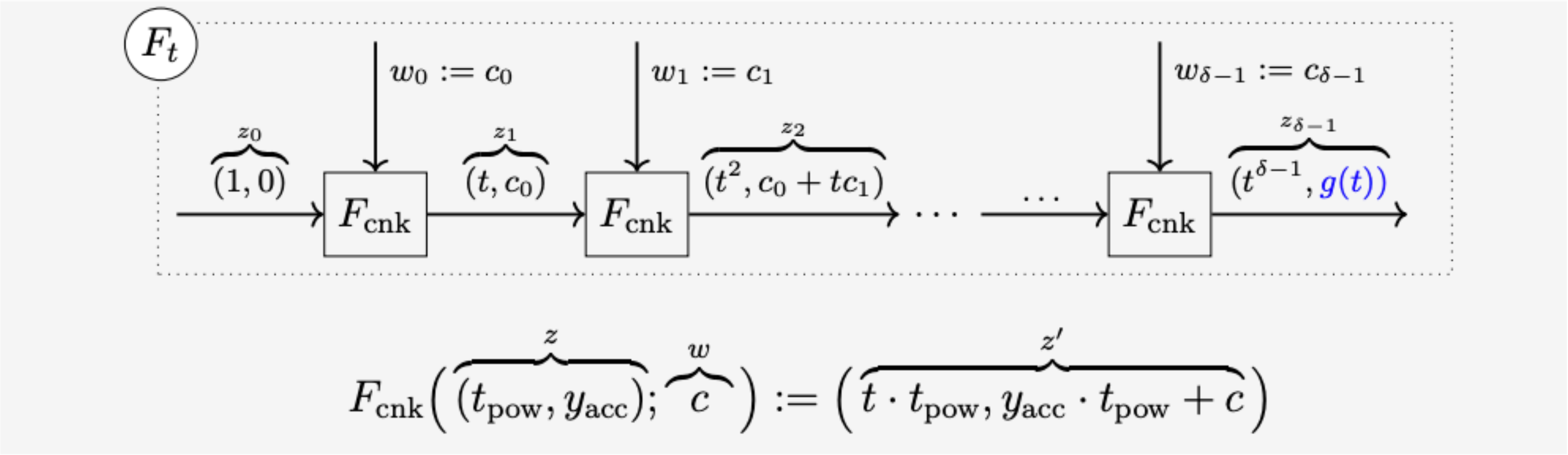
committing and (especially) proving should be doable in an incremental manner.

Example applications:

“streaming” polynomial commitments and neural network evaluation.

Our contributions for IFC:

modeling, canonical construction,
security proofs, connections to other results.



“Incremental” polynomial

Wrapping Up

Summary and, where could one go from here?

Wrapping Up

Summary and, where could one go from here?

- **This work:** A (mostly combinatorial) investigation of the “structure” of security properties in IVCs

Wrapping Up

Summary and, where could one go from here?

- **This work:** A (mostly combinatorial) investigation of the “structure” of security properties in IVCs
- **Moderately intriguing findings (at least to me):**

Wrapping Up

Summary and, where could one go from here?

- **This work:** A (mostly combinatorial) investigation of the “structure” of security properties in IVCs
- **Moderately intriguing findings (at least to me):**
    - IVC *might* be the only primitive where infinitely-often and almost-everywhere security are so interconnected.

Wrapping Up

Summary and, where could one go from here?

- **This work:** A (mostly combinatorial) investigation of the "structure" of security properties in IVCs
- **Moderately intriguing findings (at least to me):**
    - IVC *might* be the only primitive where infinitely-often and almost-everywhere security are so interconnected.
        - Are there others?

Wrapping Up

Summary and, where could one go from here?

- **This work:** A (mostly combinatorial) investigation of the “structure” of security properties in IVCs
- **Moderately intriguing findings (at least to me):**
  - IVC *might* be the only primitive where infinitely-often and almost-everywhere security are so interconnected.
    - Are there others?
    - This gives a sufficient condition for security. Concrete ways of using it?

Wrapping Up

Summary and, where could one go from here?

- **This work:** A (mostly combinatorial) investigation of the “structure” of security properties in IVCs
- **Moderately intriguing findings (at least to me):**
  - IVC *might* be the only primitive where infinitely-often and almost-everywhere security are so interconnected.
    - Are there others?
    - This gives a sufficient condition for security. Concrete ways of using it?
  - Lifting is always possible and does not require extractability (with efficient *amortized* prover)

Wrapping Up

Summary and, where could one go from here?

- **This work:** A (mostly combinatorial) investigation of the "structure" of security properties in IVCs
- **Moderately intriguing findings (at least to me):**
    - IVC *might* be the only primitive where infinitely-often and almost-everywhere security are so interconnected.
        - Are there others?
        - This gives a sufficient condition for security. Concrete ways of using it?
    - Lifting is always possible and does not require extractability (with efficient *amortized* prover)
    - The "insecurity" in insecure IVCs "does not quite improve as you move towards the safe zone" (*no graceful security degradation*)

Wrapping Up

Summary and, where could one go from here?

- **This work:** A (mostly combinatorial) investigation of the “structure” of security properties in IVCs
- **Moderately intriguing findings (at least to me):**
    - IVC *might* be the only primitive where infinitely-often and almost-everywhere security are so interconnected.
        - Are there others?
        - This gives a sufficient condition for security. Concrete ways of using it?
    - Lifting is always possible and does not require extractability (with efficient *amortized* prover)
    - The “insecurity” in insecure IVCs “does not quite improve as you move towards the safe zone” (*no graceful security degradation*)
- **Other open questions:**

Wrapping Up

Summary and, where could one go from here?

- **This work:** A (mostly combinatorial) investigation of the “structure” of security properties in IVCs
- **Moderately intriguing findings (at least to me):**
  - IVC *might* be the only primitive where infinitely-often and almost-everywhere security are so interconnected.
    - Are there others?
    - This gives a sufficient condition for security. Concrete ways of using it?
  - Lifting is always possible and does not require extractability (with efficient *amortized* prover)
  - The “insecurity” in insecure IVCs “does not quite improve as you move towards the safe zone” (*no graceful security degradation*)
- **Other open questions:**
  - How to build Incremental Functional Commitments from falsifiable assumptions?

Wrapping Up

Summary and, where could one go from here?

- **This work:** A (mostly combinatorial) investigation of the “structure” of security properties in IVCs
- **Moderately intriguing findings (at least to me):**
  - IVC *might* be the only primitive where infinitely-often and almost-everywhere security are so interconnected.
    - Are there others?
    - This gives a sufficient condition for security. Concrete ways of using it?
  - Lifting is always possible and does not require extractability (with efficient *amortized* prover)
  - The “insecurity” in insecure IVCs “does not quite improve as you move towards the safe zone” (*no graceful security degradation*)
- **Other open questions:**
  - How to build Incremental Functional Commitments from falsifiable assumptions?
  - Can we prove the security/insecurity of concrete existing systems (with these or other techniques)?

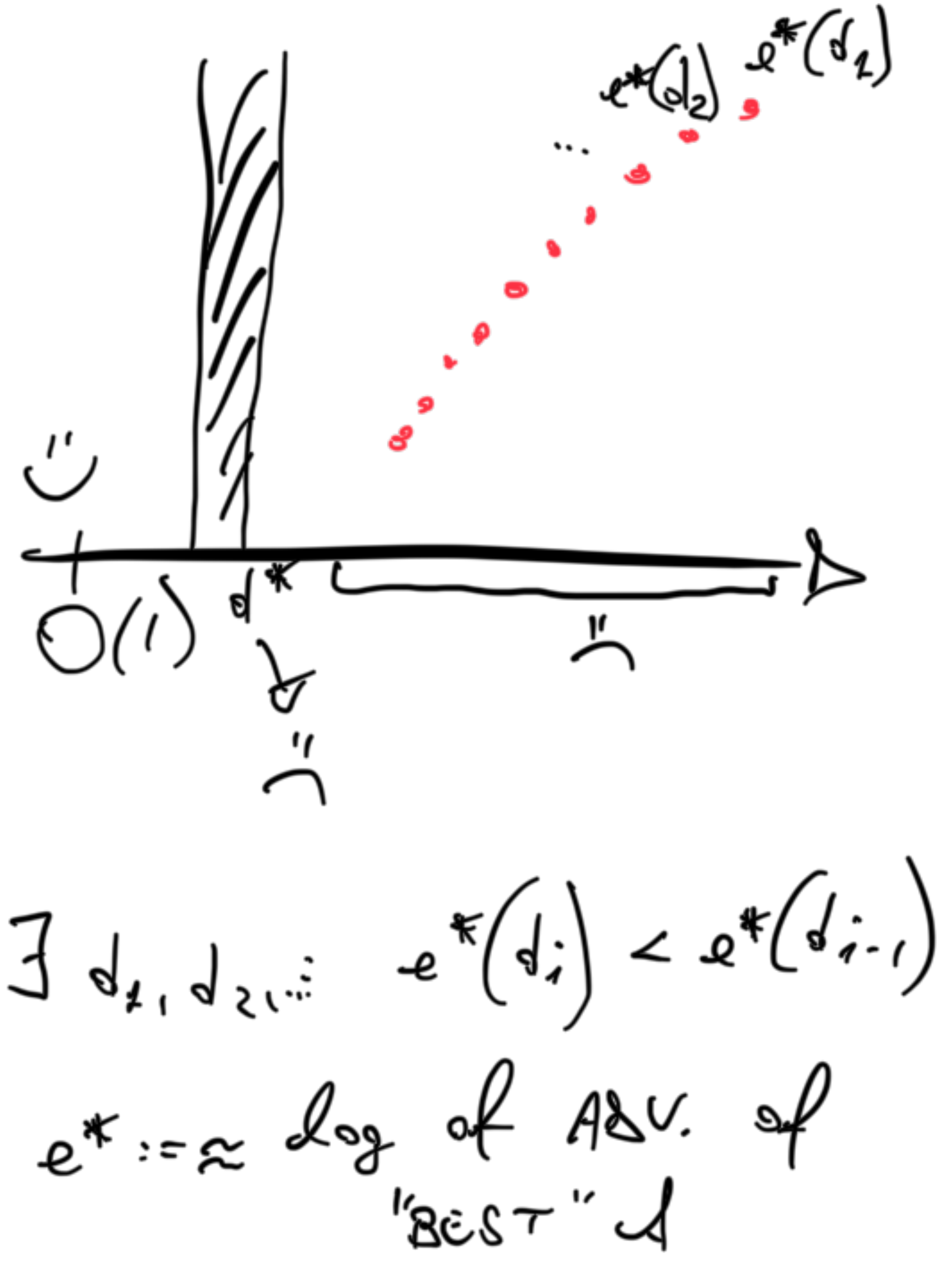
Thank you! Questions?

Wrapping Up

Summary and, where could one go from here?

- **This work:** A (mostly combinatorial) investigation of the "structure" of security properties in IVCs
- **Moderately intriguing findings (at least to me):**
  - IVC *might* be the only primitive where infinitely-often and almost-everywhere security are so interconnected.
    - Are there others?
    - This gives a sufficient condition for security. Concrete ways of using it?
  - Lifting is always possible and does not require extractability (with efficient *amortized* prover)
  - The "insecurity" in insecure IVCs "does not quite improve as you move towards the safe zone" (*no graceful security degradation*)
- **Other open questions:**
  - How to build Incremental Functional Commitments from falsifiable assumptions?
  - Can we prove the security/insecurity of concrete existing systems (with these or other techniques)?

Extra slide on graceful sec. degradation



Theorem 6. *Given any sequence of superconstant—i.e., $\omega(1)$—depth bounds $R_0, R_1, R_2, \ldots$, there always exists a superconstant depth bound L such that for all i, $R_i \succeq L$.*

Extra slide on io-soundness

Theorem (Informal statement of Corollary 1). *Let Π be an* IVC *scheme and* $\mathsf{D} = \omega(1)$ *be a depth bound. Let* $E \subseteq \mathbb{N}$ *be an infinite and "exponentially sparse" set of security parameters where* Π *achieves negligible soundness at depth bound* D*. Then there exists a depth bound* $d = \omega(1)$ *where* Π *achieves (standard) negligible soundness.*

Theorem (Informal statement of Corollary 2). *Let* $\Pi, \mathsf{D} = \omega(1)$, *and* E *as in the previous theorem. Then:*

- E *exponentially sparse* $\implies d = O(\log \mathsf{D})$.
- E *sub-exponentially sparse* $\implies d = O(\mathrm{polylog}\mathsf{D})$.

Theorem 3. *Let* $E = \{\lambda_1 < \lambda_2 < \cdots\} \subseteq \mathbb{N}$ *be a constructible* (2^{κ^T})*-sparse set for some* T *with* $0 < T \leq 1$*. Let* Π *be an IVC that is i.o-sound with respect to* E *for depth bound* $\mathsf{D}(\cdot)$*. Let* $d'(\cdot)$ *be a depth bound. If for all* $i \in \mathbb{N}$,

$$d'(\lambda_{i+1} - 1) \leq \mathsf{D}(\lambda_i), \qquad (\blacktriangle)$$

then Π *is (almost-everywhere) sound for depth bound* d' *if appropriately parameterized (Definition 3). The resulting proving time, verification time and proof size are like those originally in* Π *(up to constant factors).*