Incrementally Verifiable Computation for NP from Standard Assumptions

based on joint work with






Pratish Datta
NTT Research

Abhishek Jain
JHU & NTT Research

Zhengzhong Jin
Northeastern

Surya Mathialagan
MIT → NTT Research

Alexis Korb
UCLA

Amit Sahai
UCLA

Incrementally Verifiable Computation for NP from Standard Assumptions $(i\mathcal{O})$

based on joint work with


Pratish Datta
NTT Research


Abhishek Jain
JHU & NTT Research


Zhengzhong Jin
Northeastern


Surya Mathialagan
MIT → NTT Research


Alexis Korb
UCLA


Amit Sahai
UCLA

Motivation: "Multi-Generation" Computation

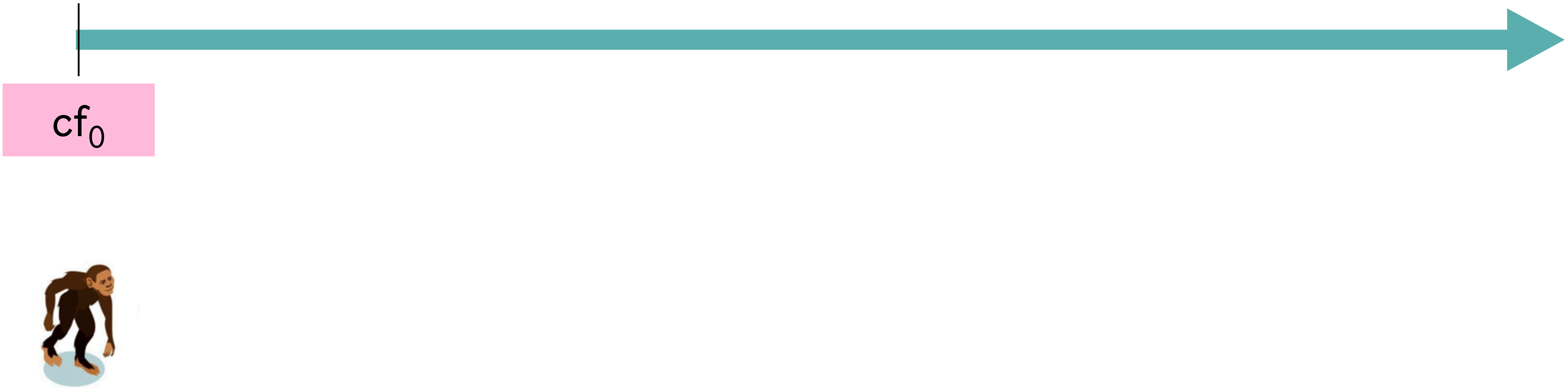
[Valiant '08]

(Nondeterministic) computation $\mathscr{M}$

Motivation: “Multi-Generation” Computation

[Valiant ’08]

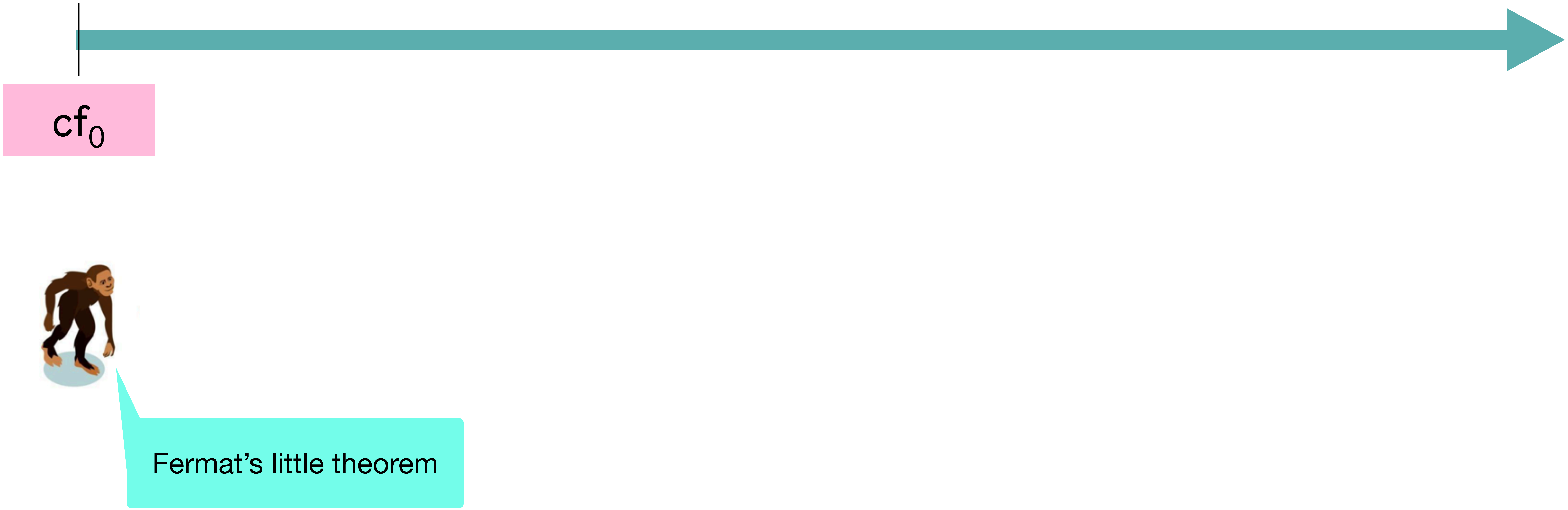
(Nondeterministic) computation $\mathscr{M}$



Motivation: “Multi-Generation” Computation

[Valiant ’08]

(Nondeterministic) computation $\mathscr{M}$



Motivation: “Multi-Generation” Computation

[Valiant ’08]

(Nondeterministic) computation $\mathscr{M}$

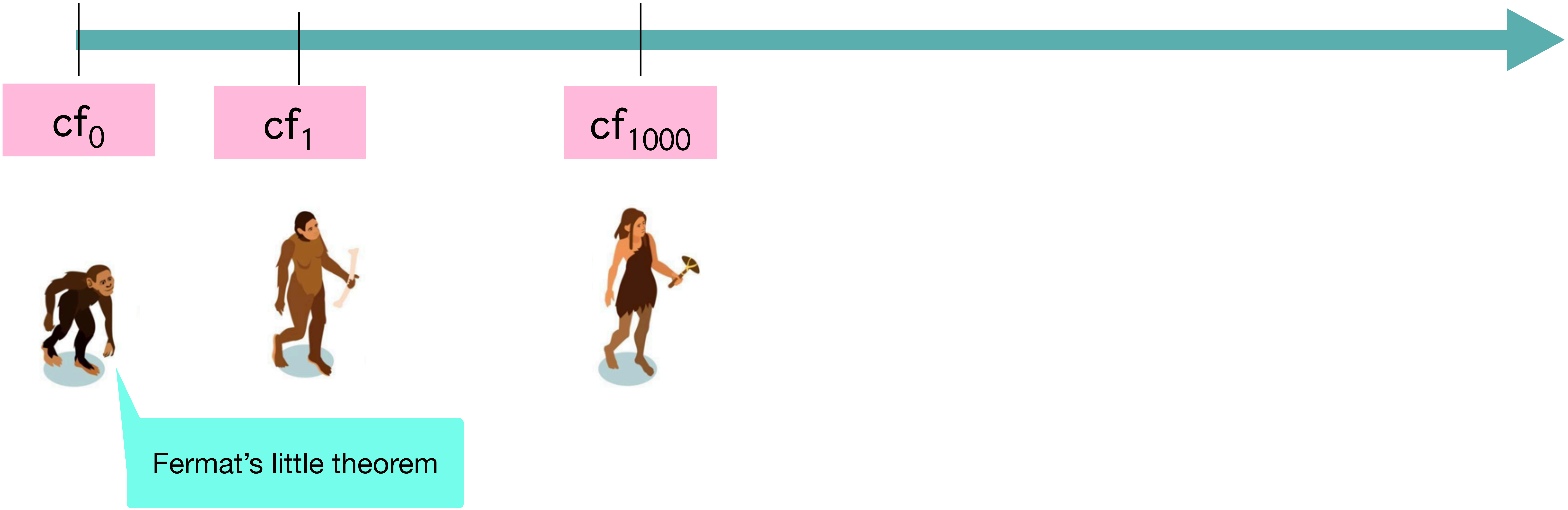
cf_0 cf_1

Fermat’s little theorem

Motivation: “Multi-Generation” Computation

[Valiant ’08]

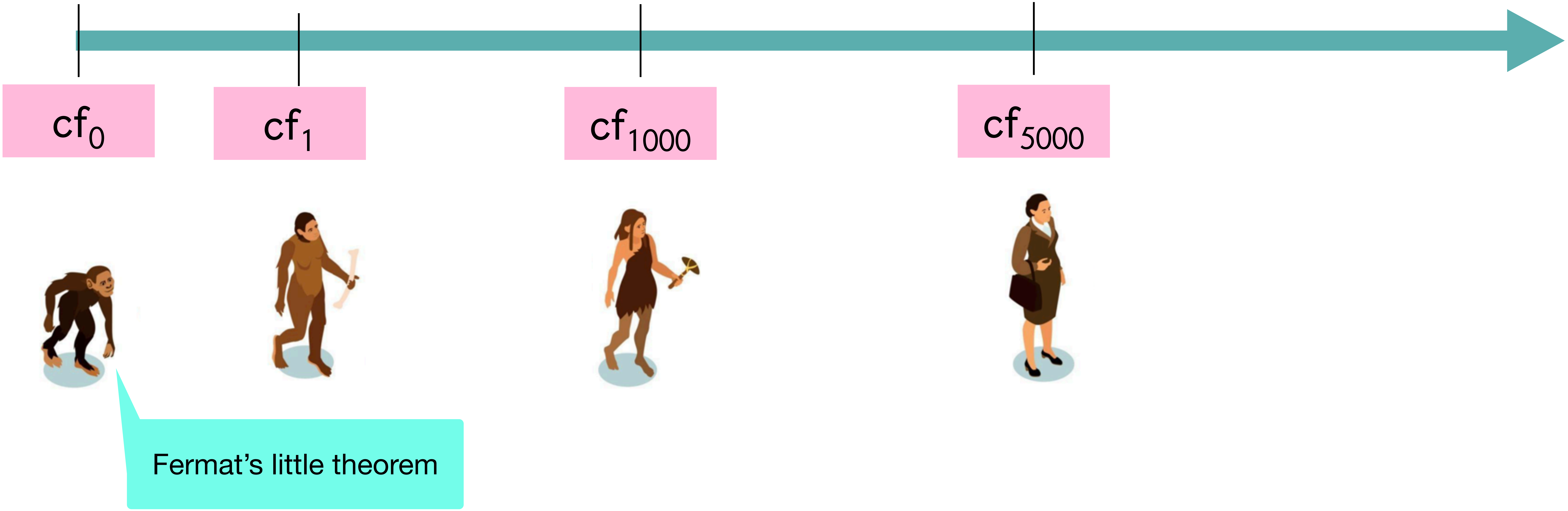
(Nondeterministic) computation $\mathscr{M}$



Motivation: “Multi-Generation” Computation

[Valiant ’08]

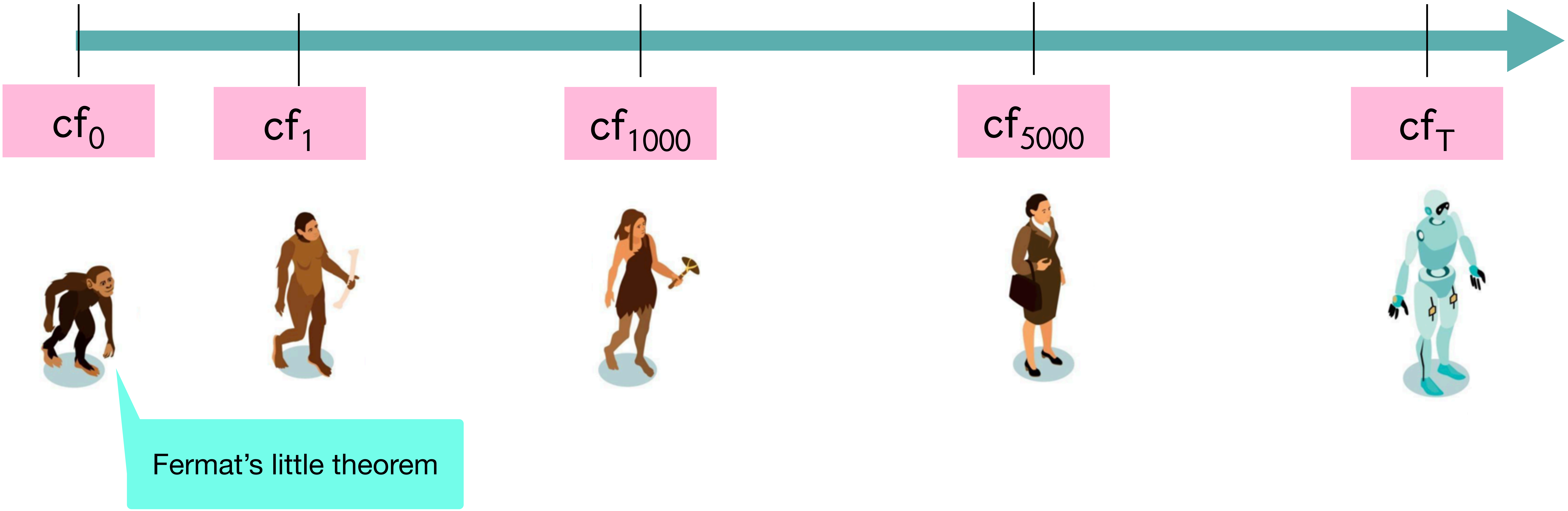
(Nondeterministic) computation $\mathscr{M}$



Motivation: “Multi-Generation” Computation

[Valiant ’08]

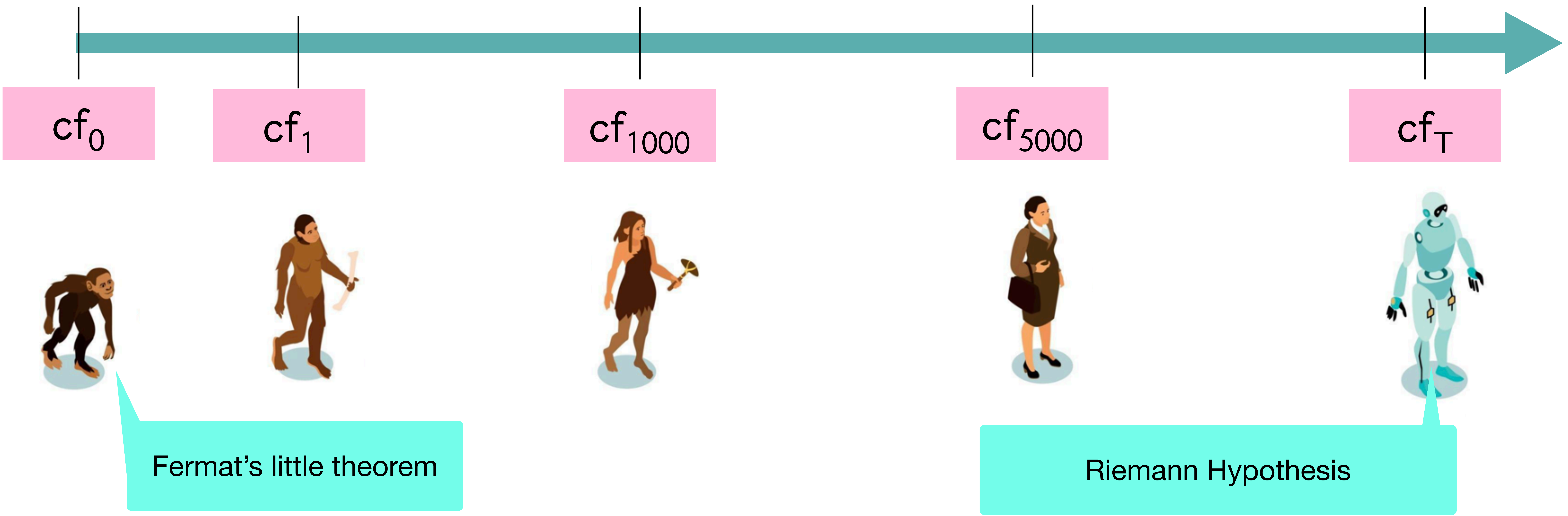
(Nondeterministic) computation $\mathscr{M}$



Motivation: “Multi-Generation” Computation

[Valiant ’08]

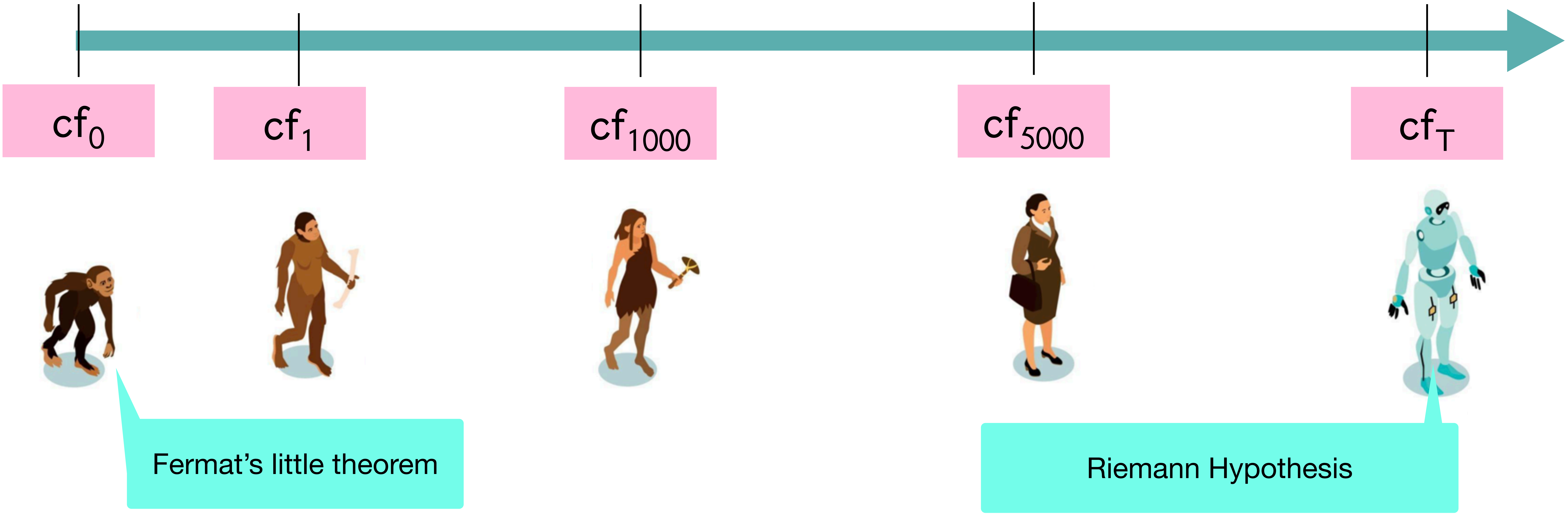
(Nondeterministic) computation $\mathscr{M}$



Motivation: “Multi-Generation” Computation

[Valiant ’08]

(Nondeterministic) computation $\mathscr{M}$



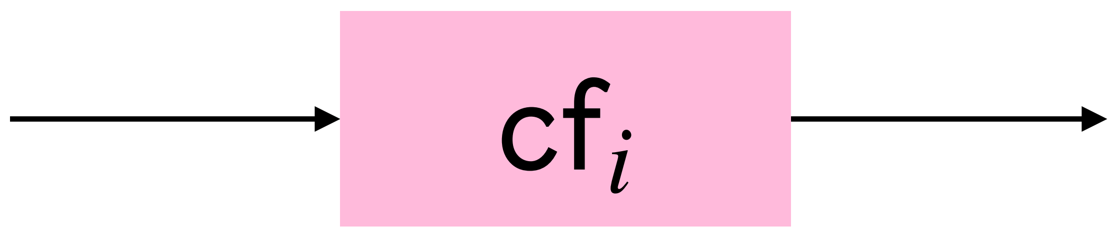
How can we **trust** the validity of the **intermediate** configurations?

Incrementally Verifiable Computation for NP

[Valiant ’08]

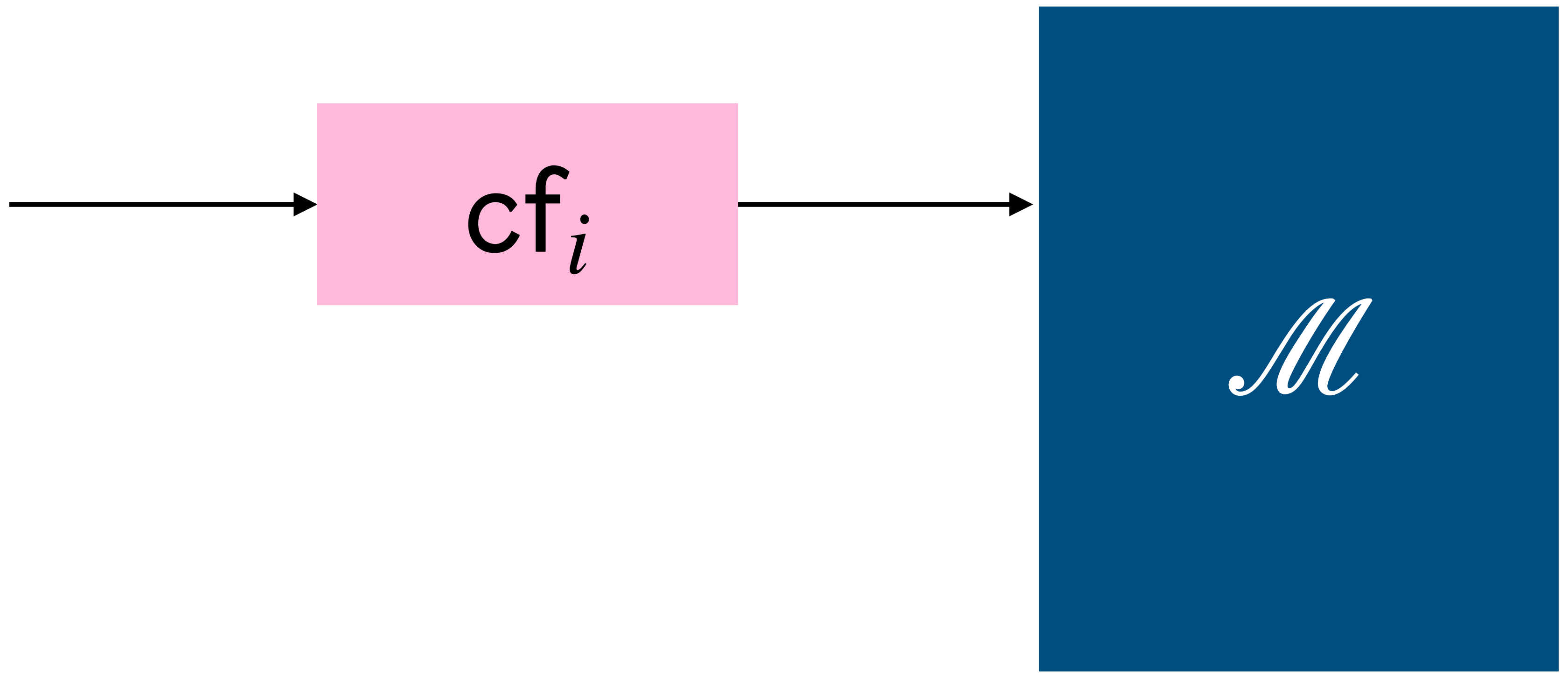
Incrementally Verifiable Computation for NP

[Valiant ’08]



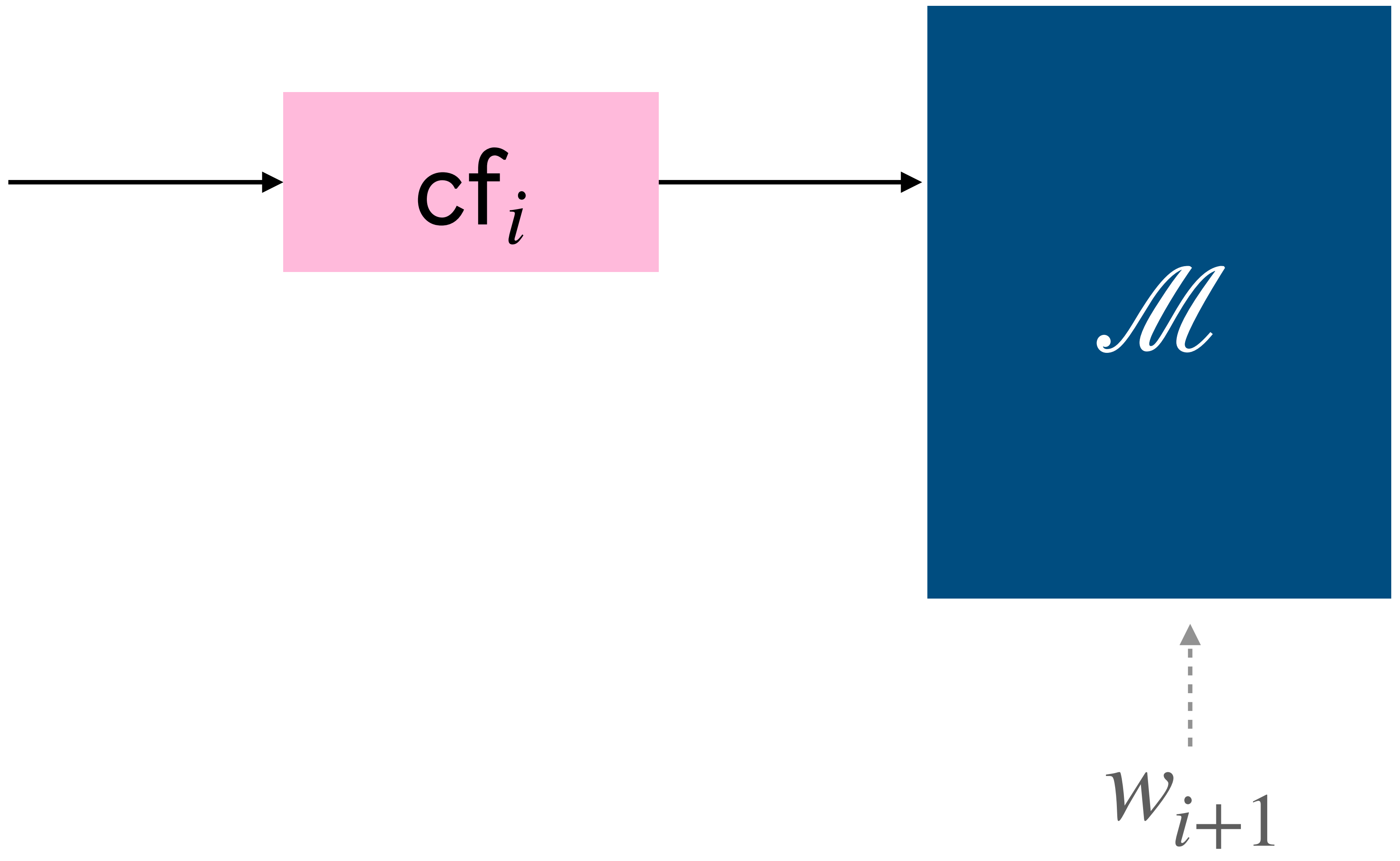
Incrementally Verifiable Computation for NP

[Valiant '08]



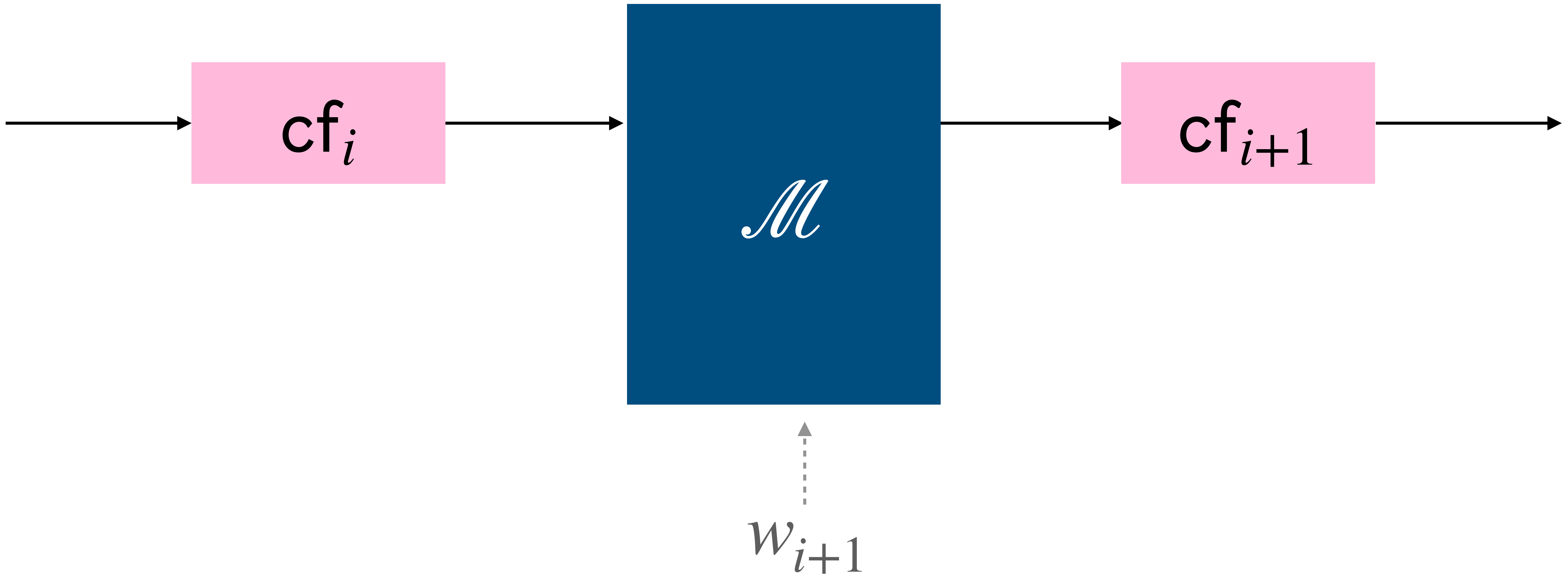
Incrementally Verifiable Computation for NP

[Valiant '08]



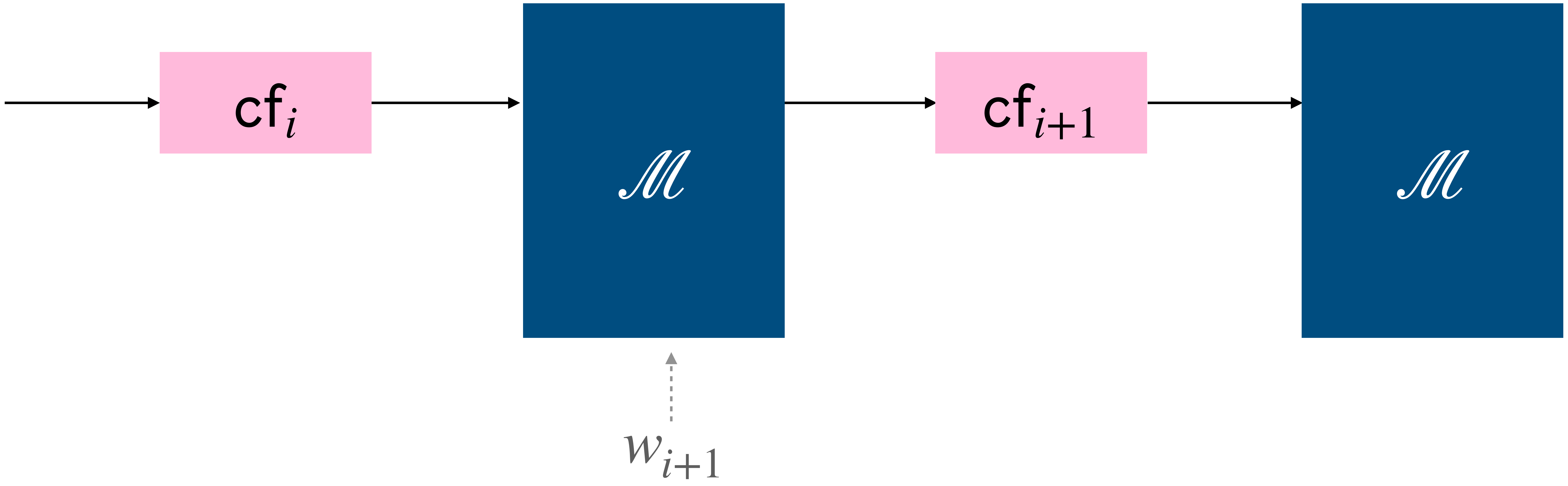
Incrementally Verifiable Computation for NP

[Valiant '08]



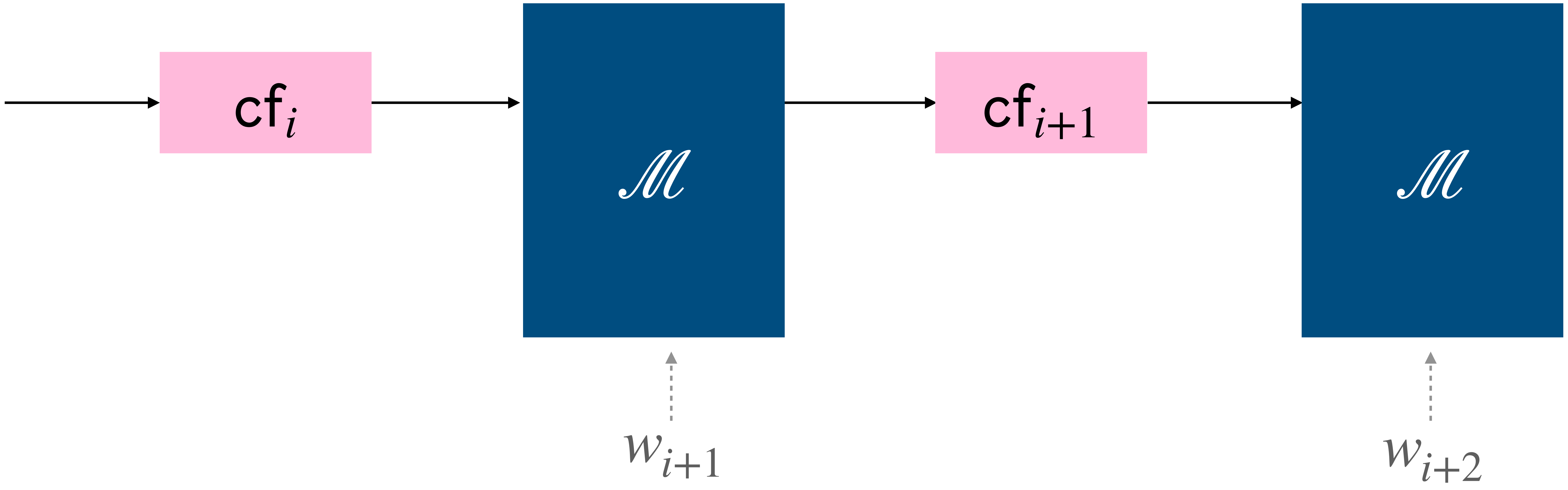
Incrementally Verifiable Computation for NP

[Valiant '08]



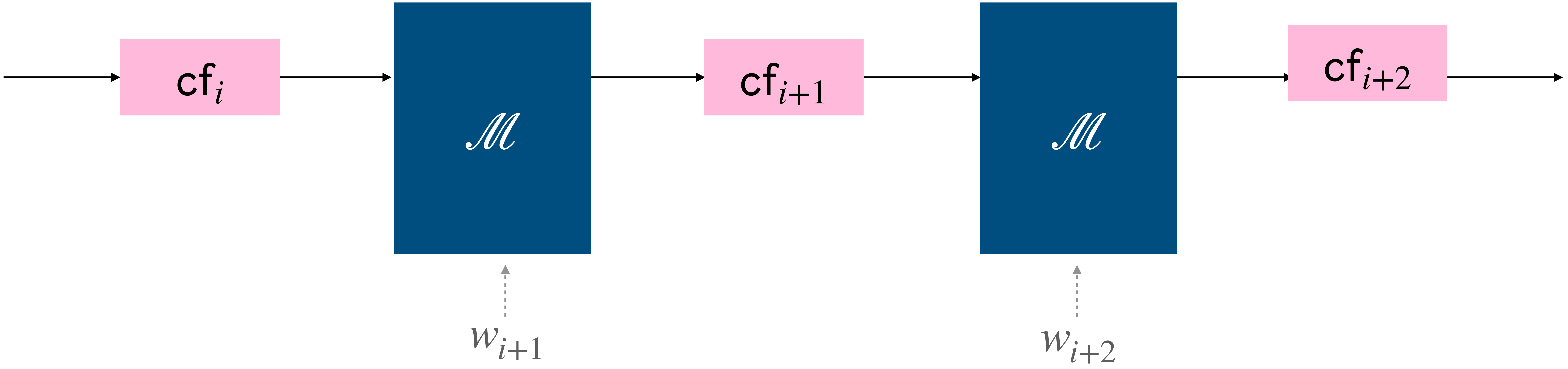
Incrementally Verifiable Computation for NP

[Valiant '08]



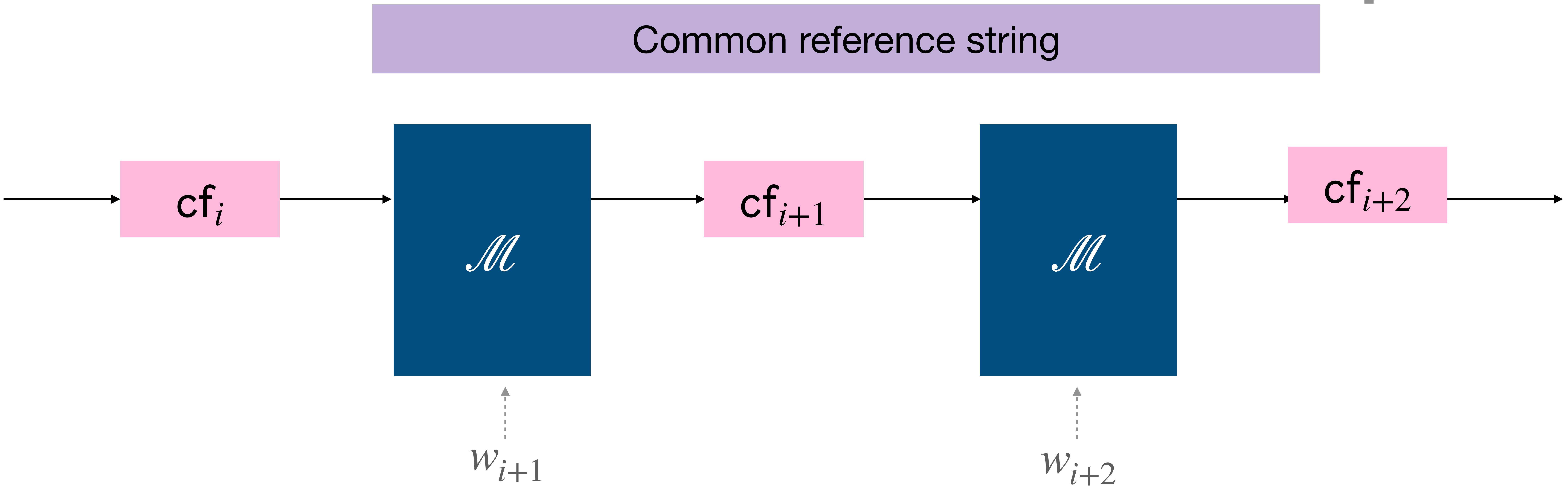
Incrementally Verifiable Computation for NP

[Valiant '08]



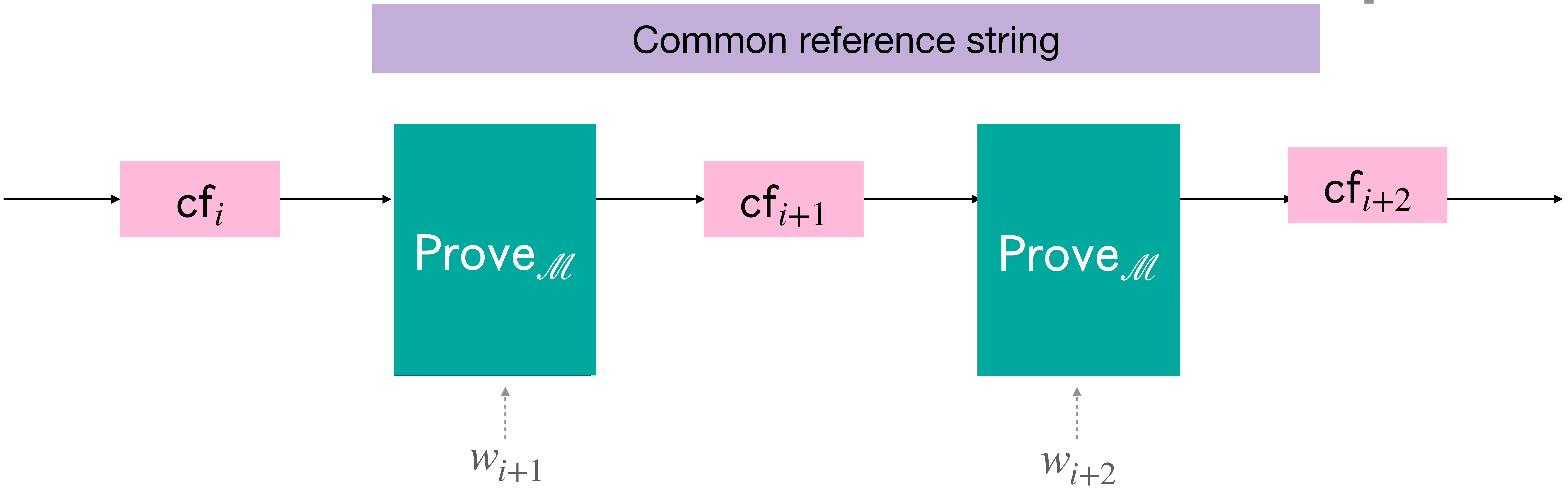
Incrementally Verifiable Computation for NP

[Valiant '08]



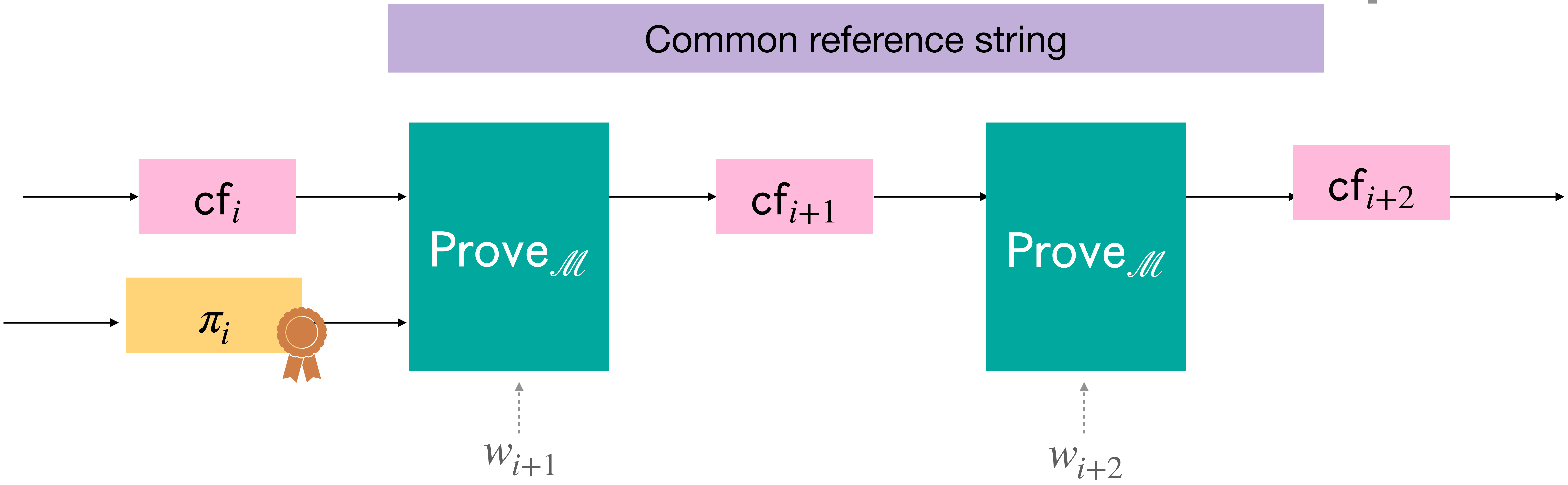
Incrementally Verifiable Computation for NP

[Valiant ’08]



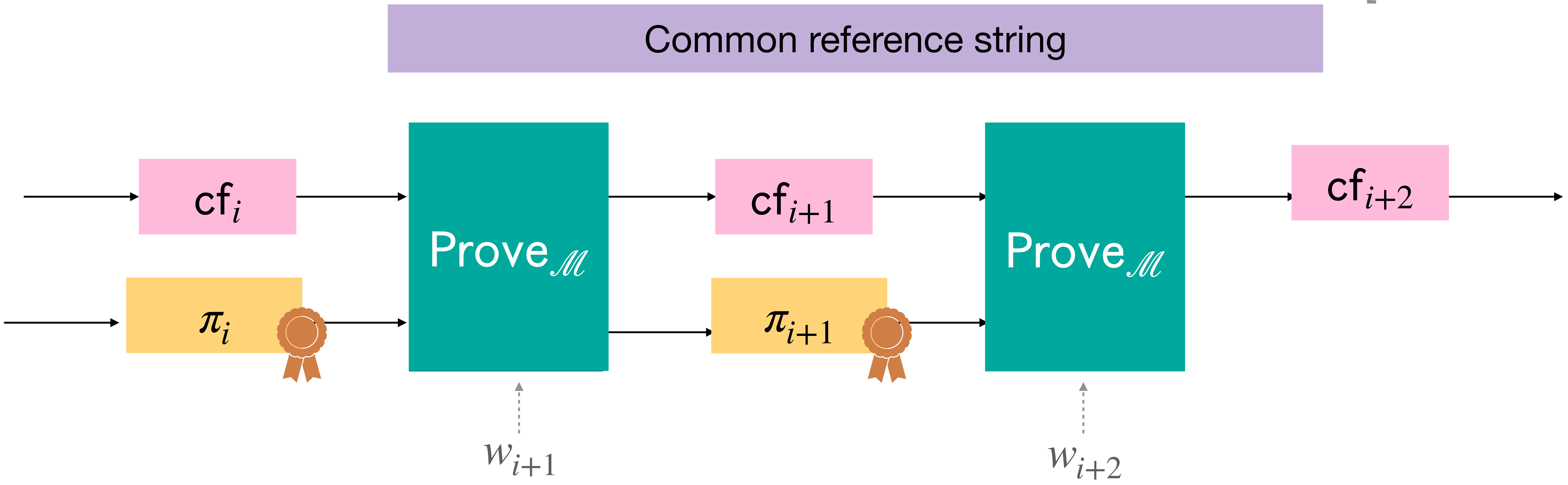
Incrementally Verifiable Computation for NP

[Valiant '08]



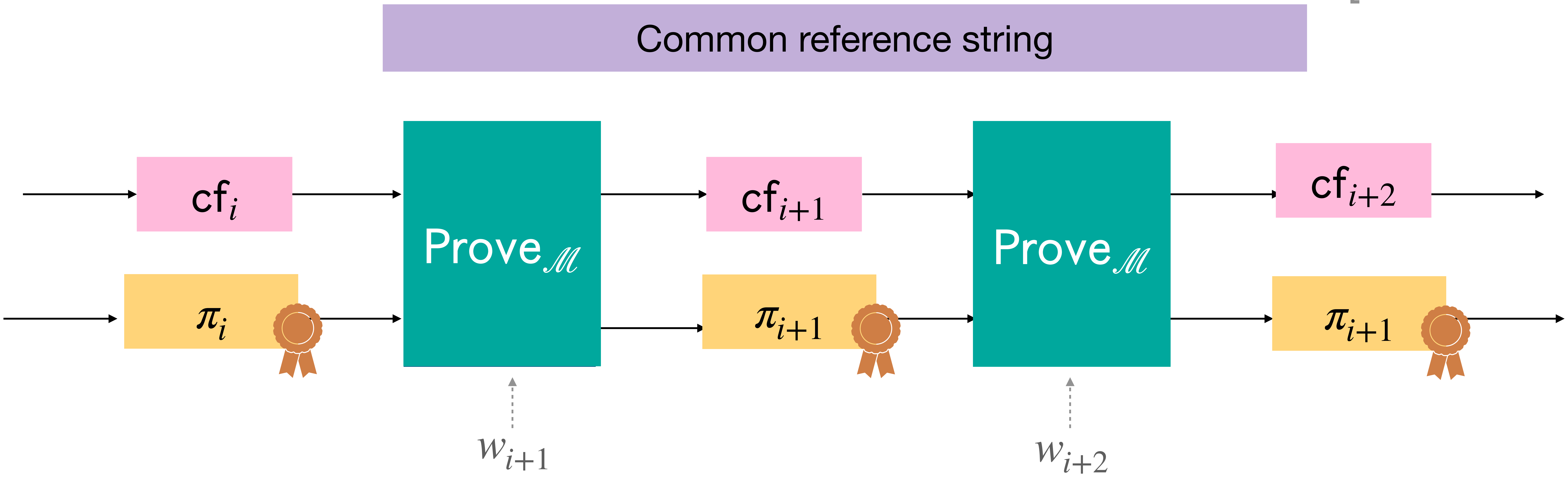
Incrementally Verifiable Computation for NP

[Valiant '08]



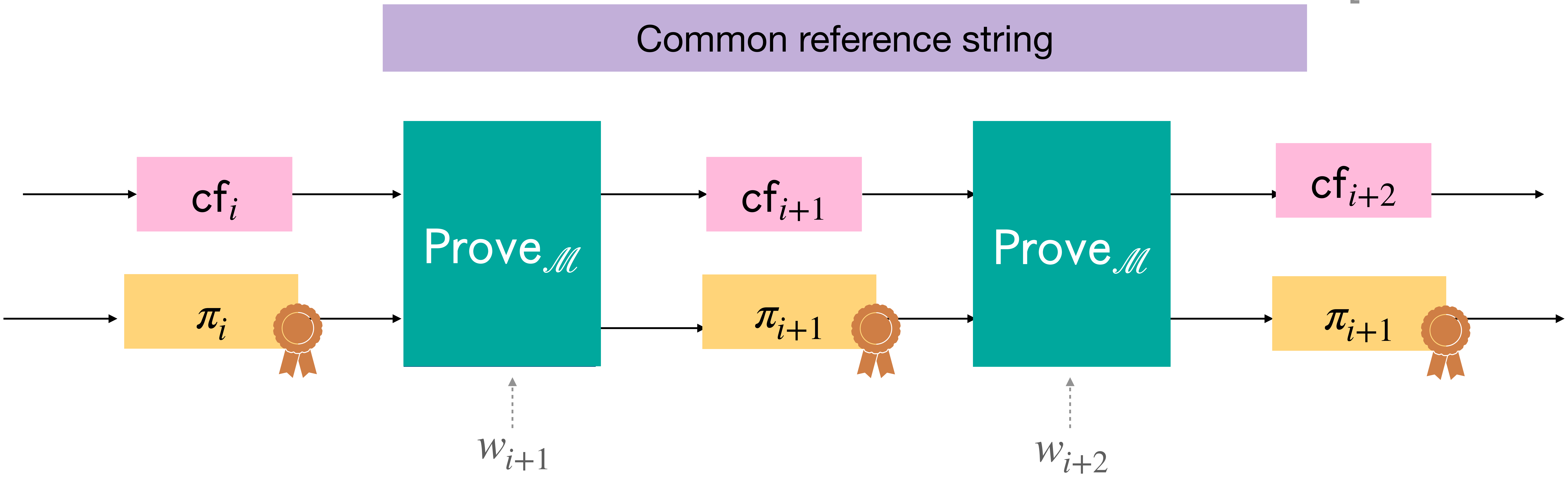
Incrementally Verifiable Computation for NP

[Valiant '08]



Incrementally Verifiable Computation for NP

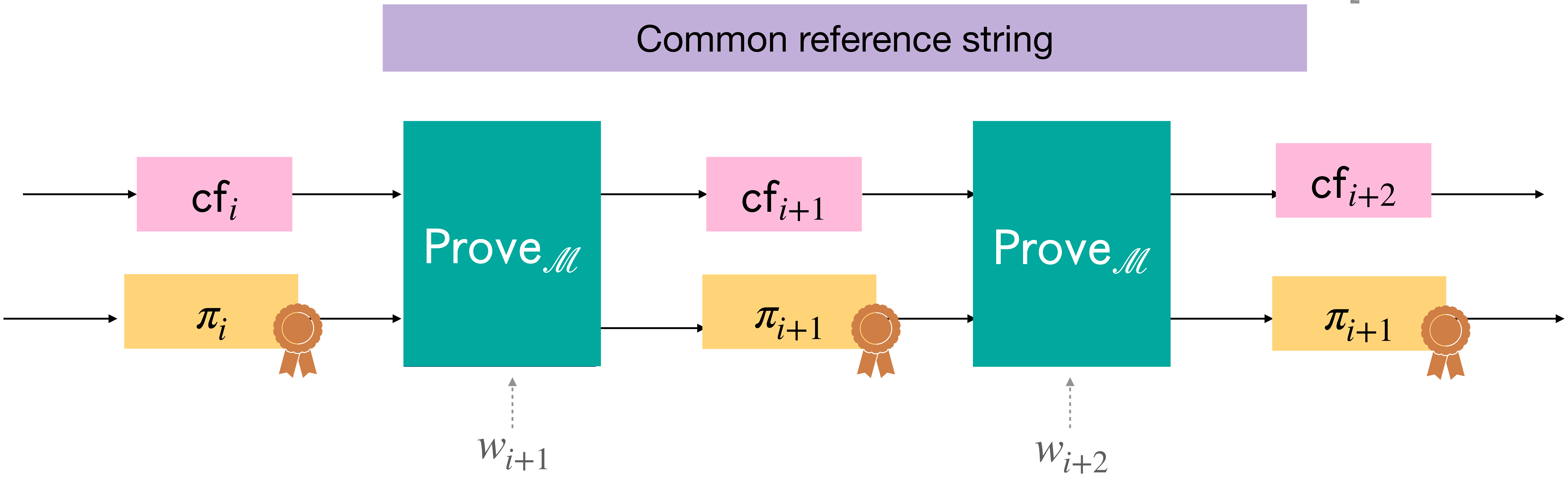
[Valiant '08]



- **Efficiency:** Proof size and verification time are **independent** of the *number of hops*.

Incrementally Verifiable Computation for NP

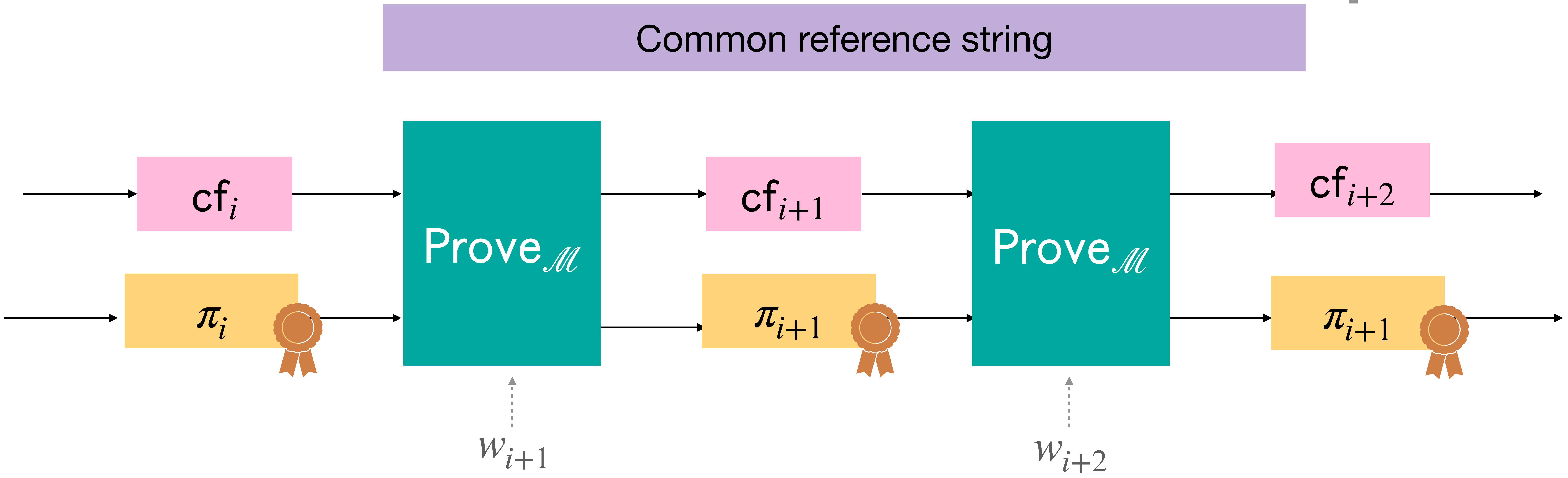
[Valiant '08]



- **Efficiency:** Proof size and verification time are **independent** of the *number of hops*.
- **Soundness:** Hard to come up with proofs for $\mathsf{cf}_0 \not\rightarrow \mathsf{cf}_T$.

Incrementally Verifiable Computation for NP

[Valiant '08]

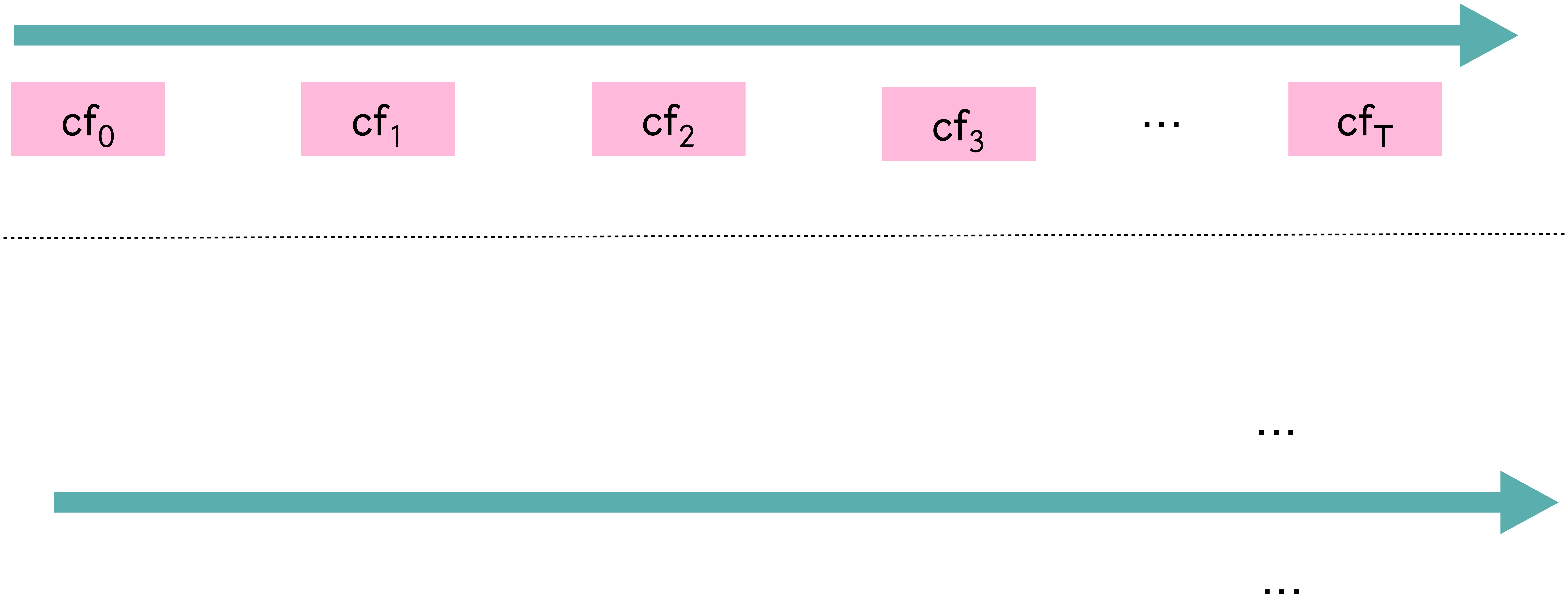


- **Efficiency:** Proof size and verification time are **independent** of the *number of hops*.
- **Soundness:** Hard to come up with proofs for $\mathsf{cf}_0 \not\to \mathsf{cf}_T$.

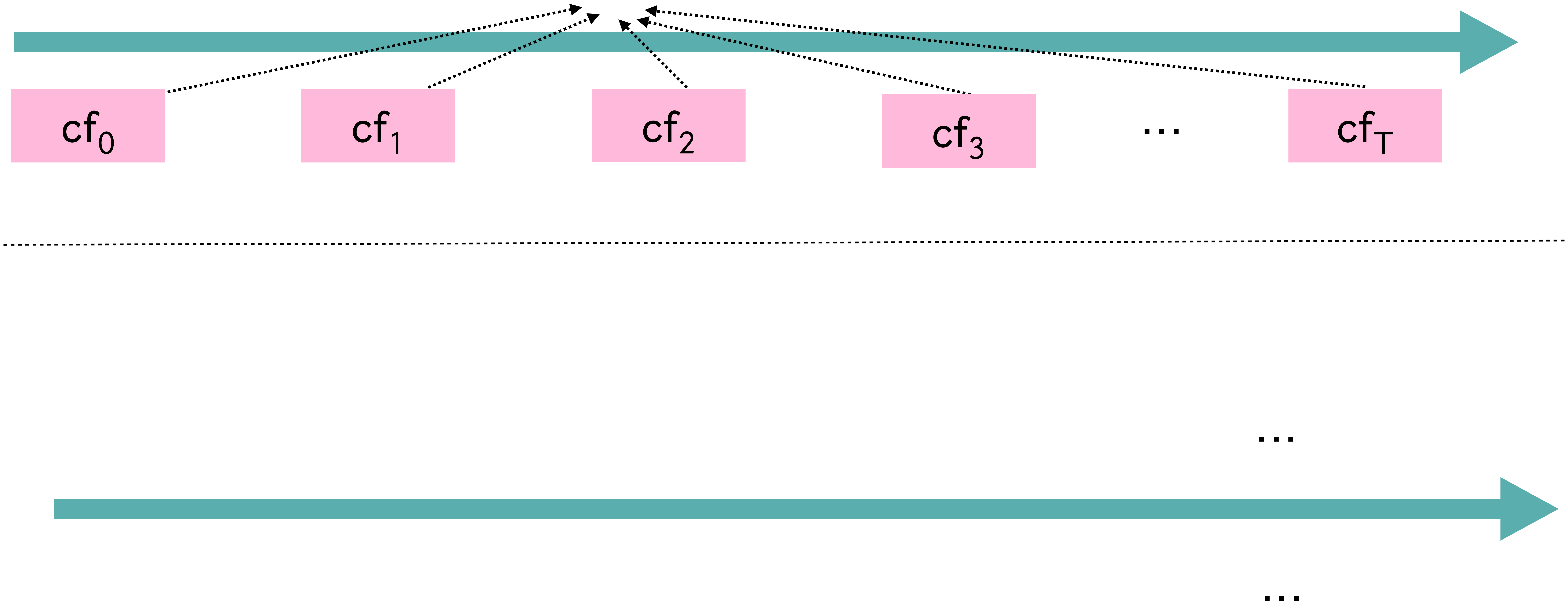
*We will not consider knowledge soundness since we are focusing on **standard assumptions.**

IVC = Multi-Hop SNARG

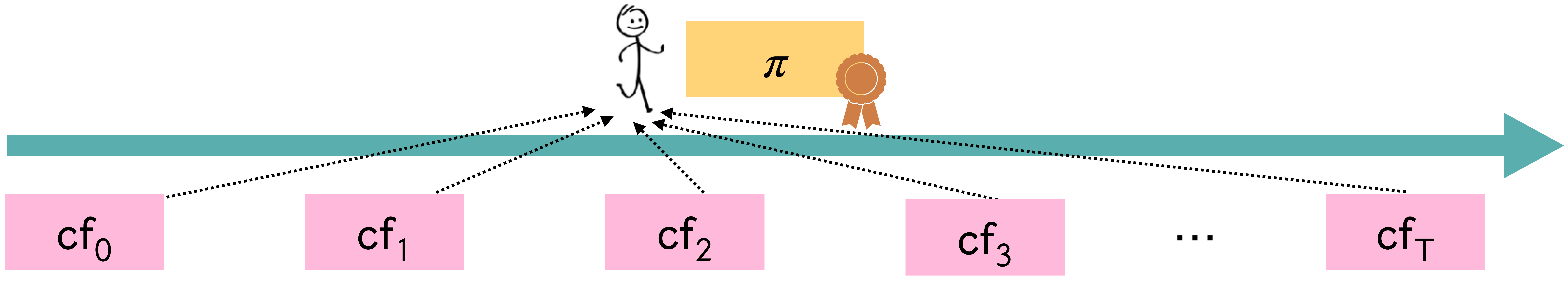
IVC = Multi-Hop SNARG



IVC = Multi-Hop SNARG



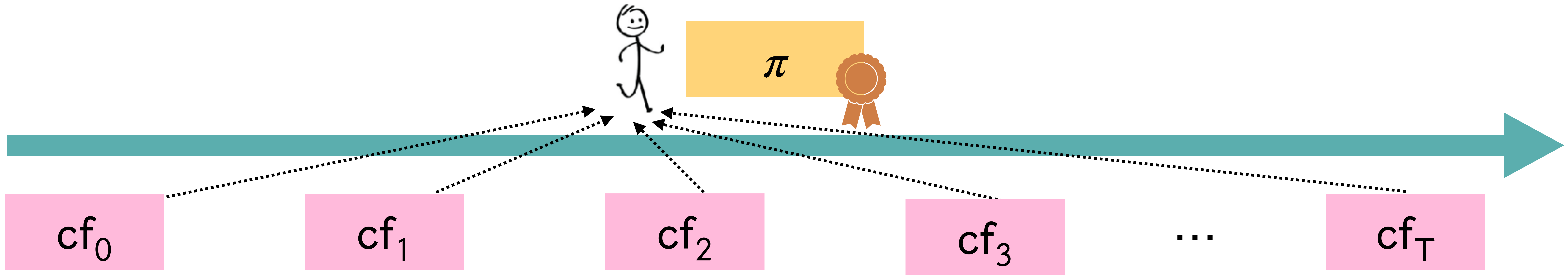
IVC = Multi-Hop SNARG



...

...

IVC = Multi-Hop SNARG

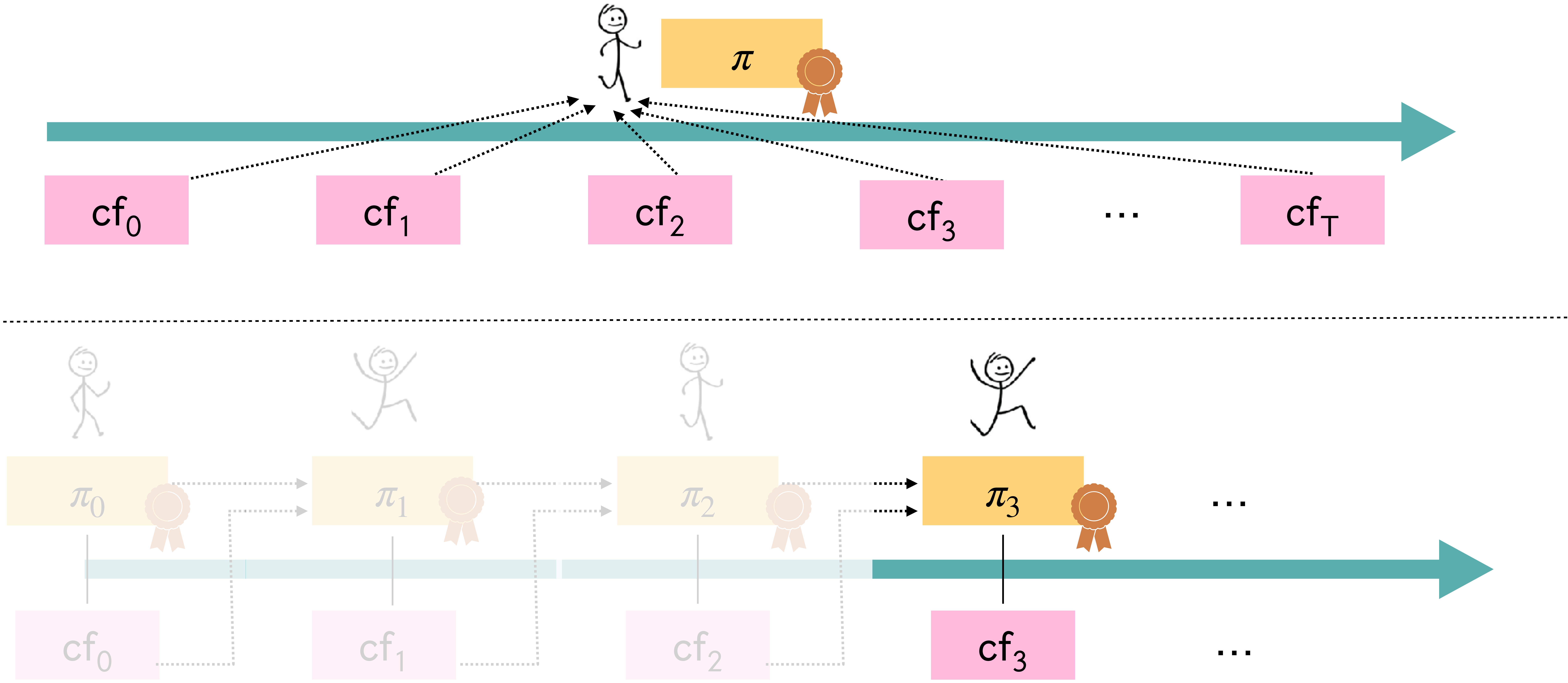


IVC = Multi-Hop SNARG

IVC = Multi-Hop SNARG

IVC = Multi-Hop SNARG

IVC = Multi-Hop SNARG



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

- N parties encrypt **votes** under a rerandomizable scheme.

Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

- N parties encrypt **votes** under a rerandomizable scheme.

ct_1

ct_1

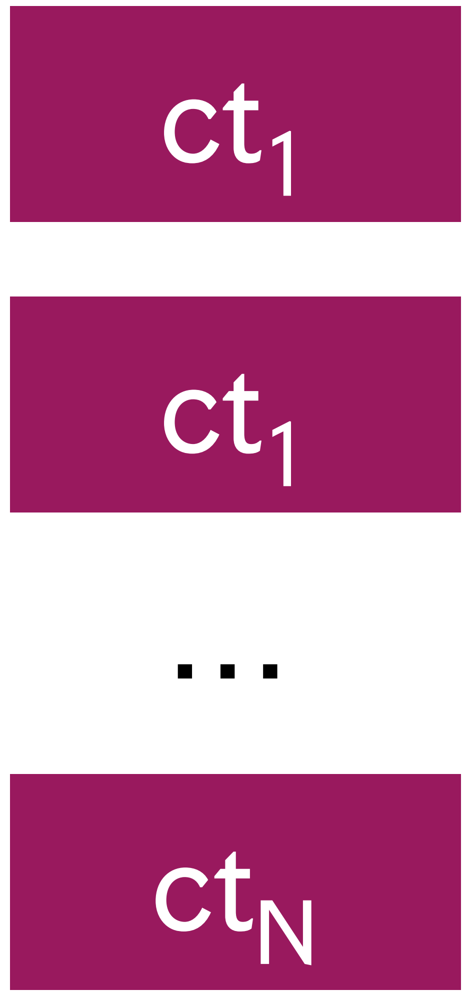
…

ct_N

Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

- N parties encrypt **votes** under a rerandomizable scheme.



Encrypted votes!

Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.

ct_1

ct_1

…

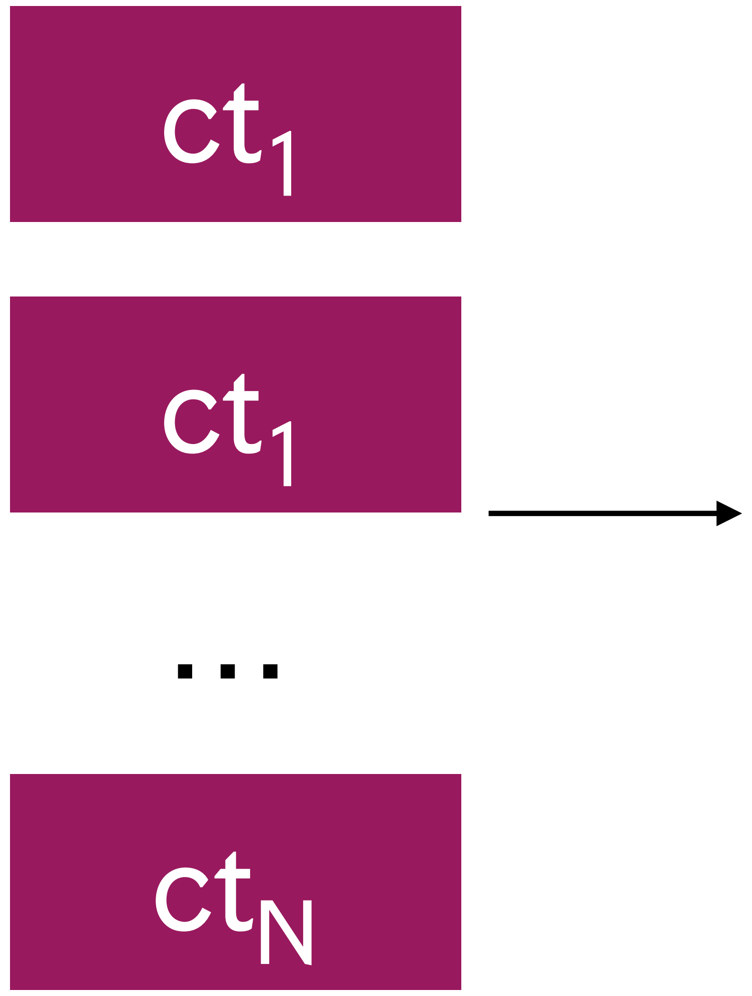
ct_N

Encrypted votes!

Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.

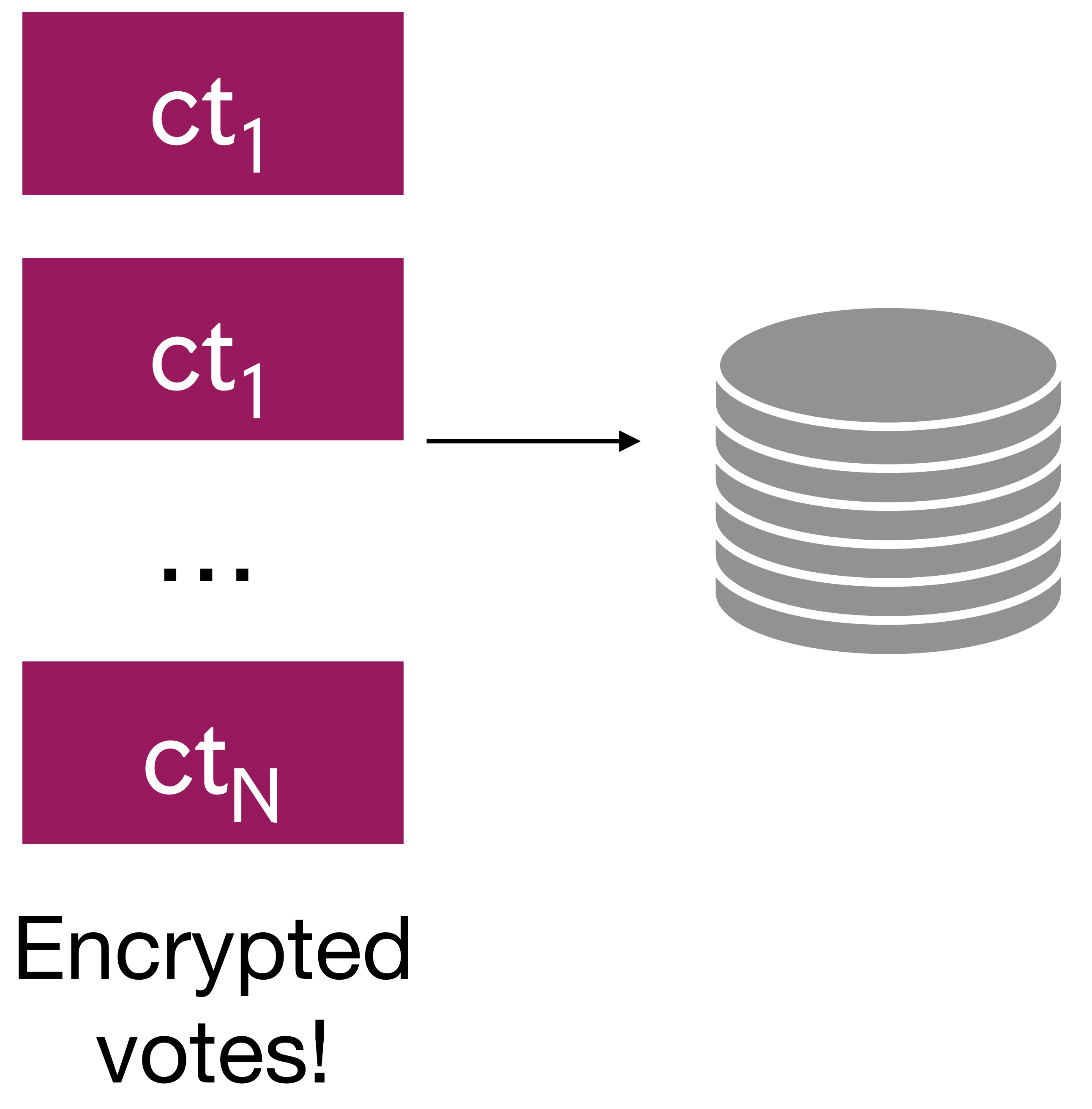


Encrypted votes!

Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

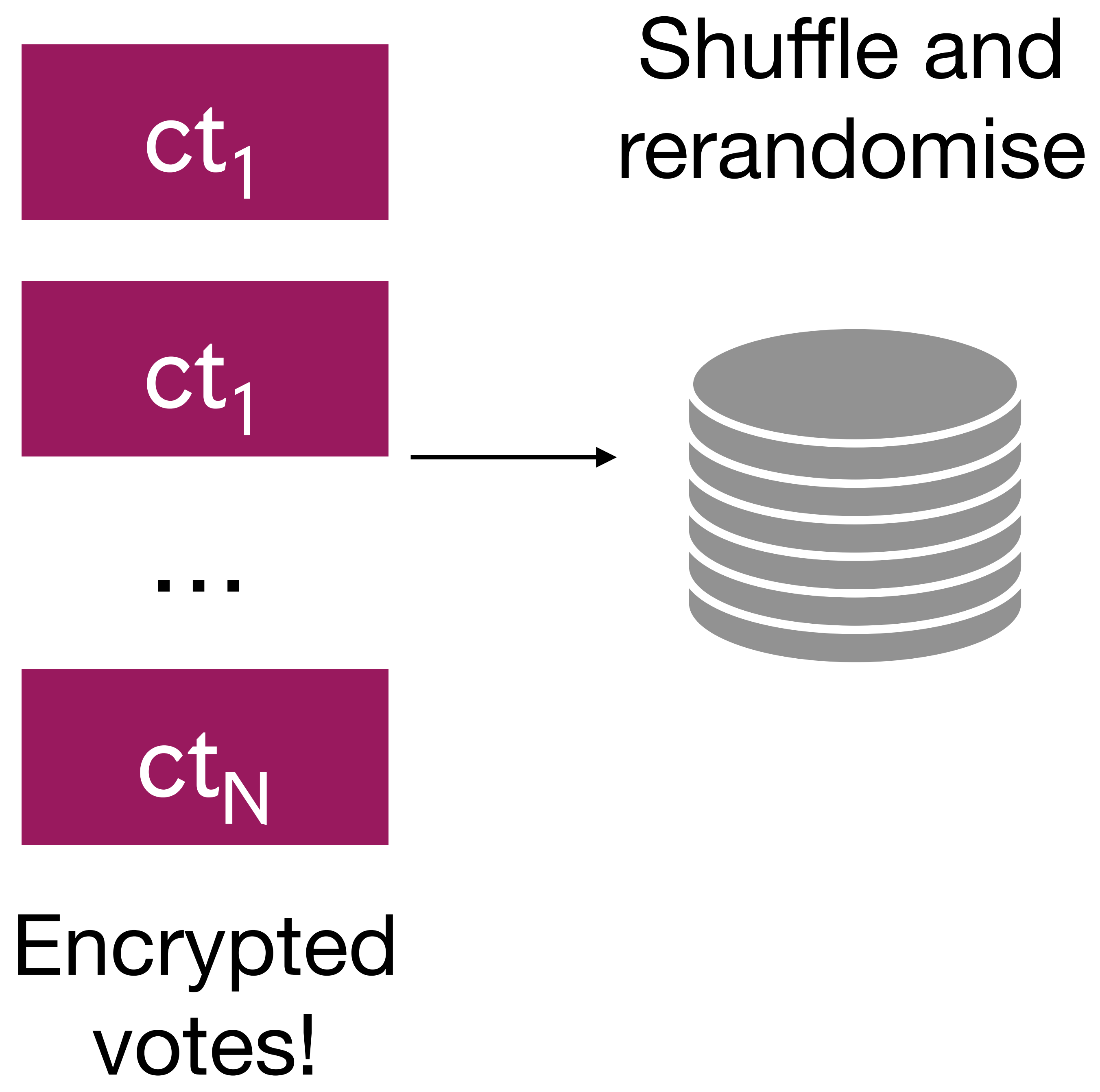
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

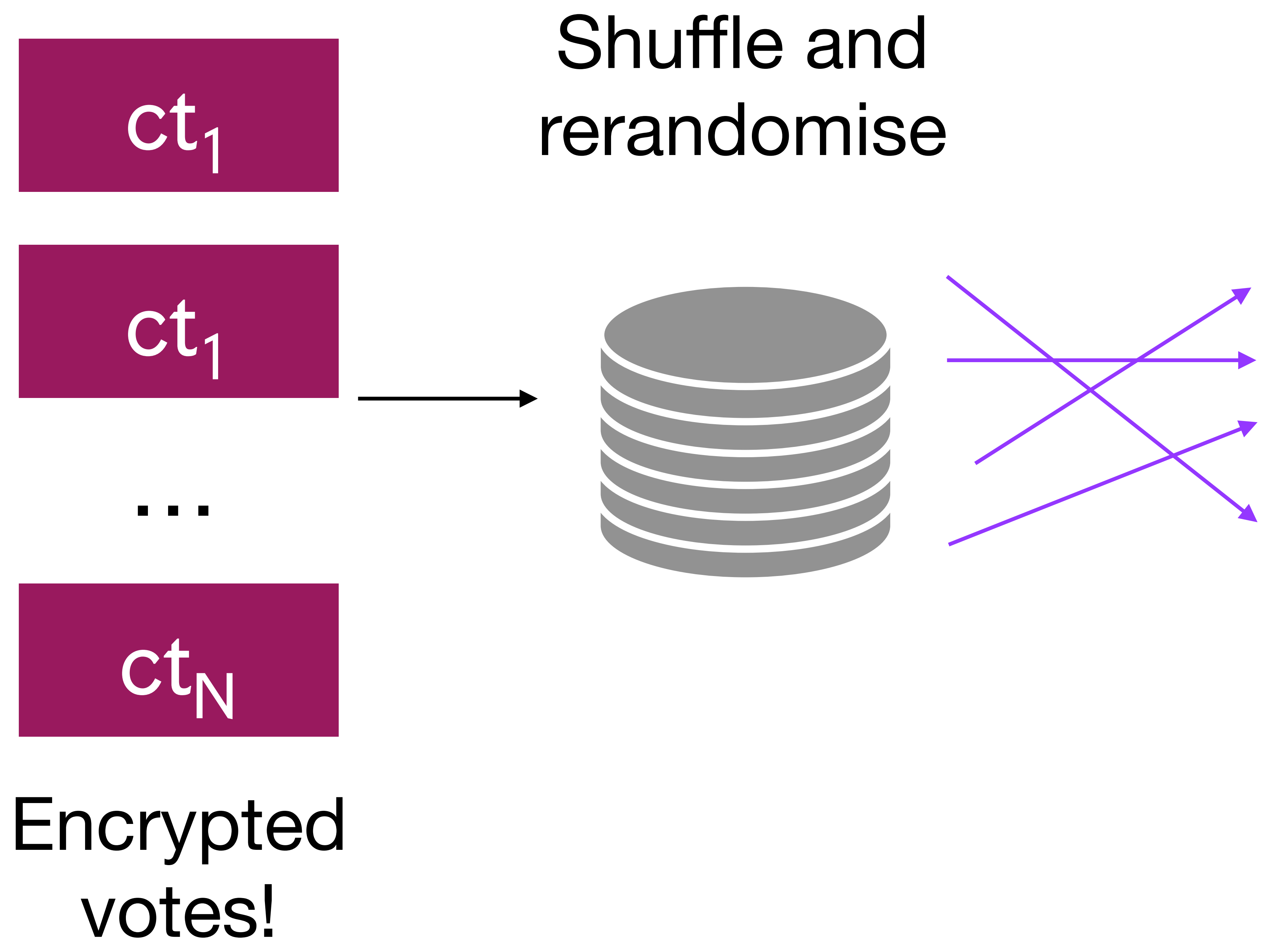
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

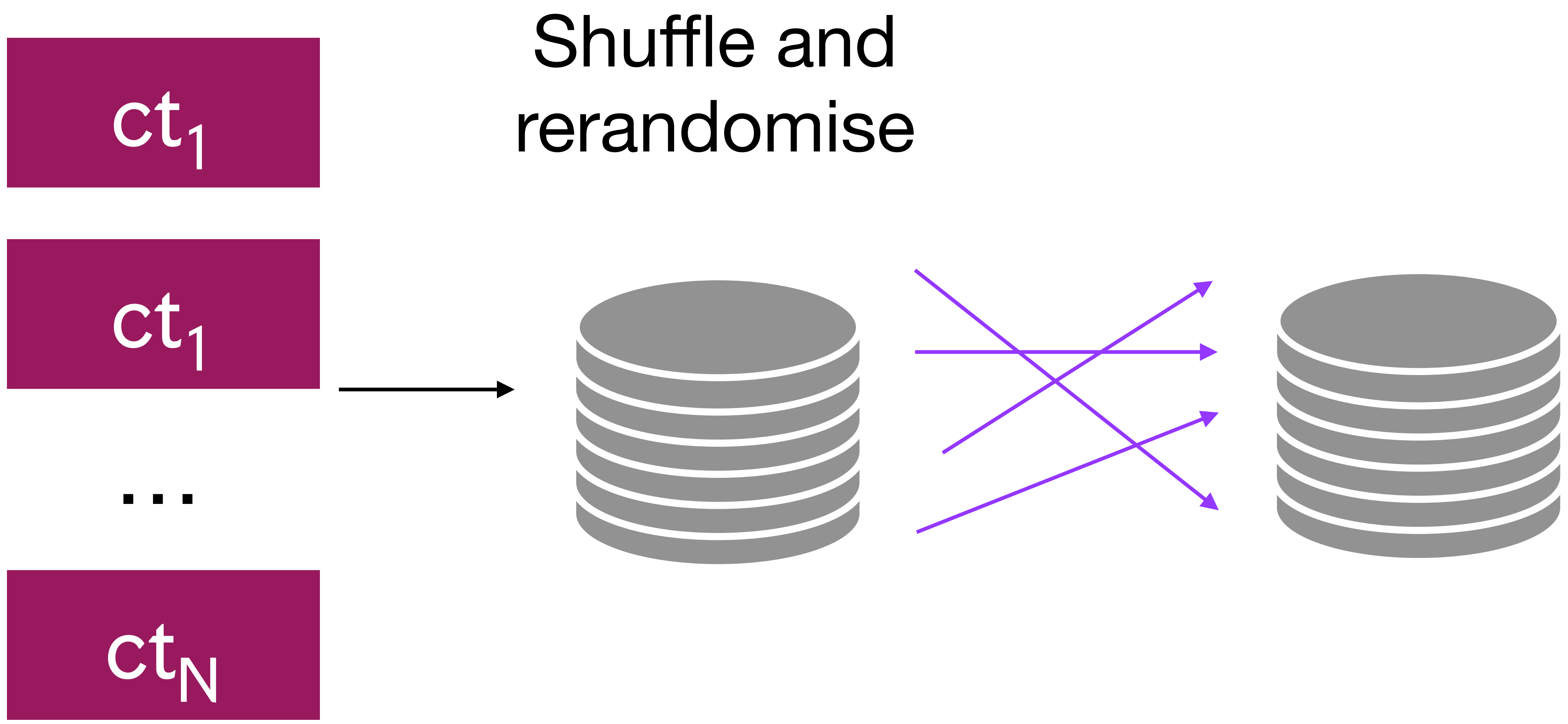
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.

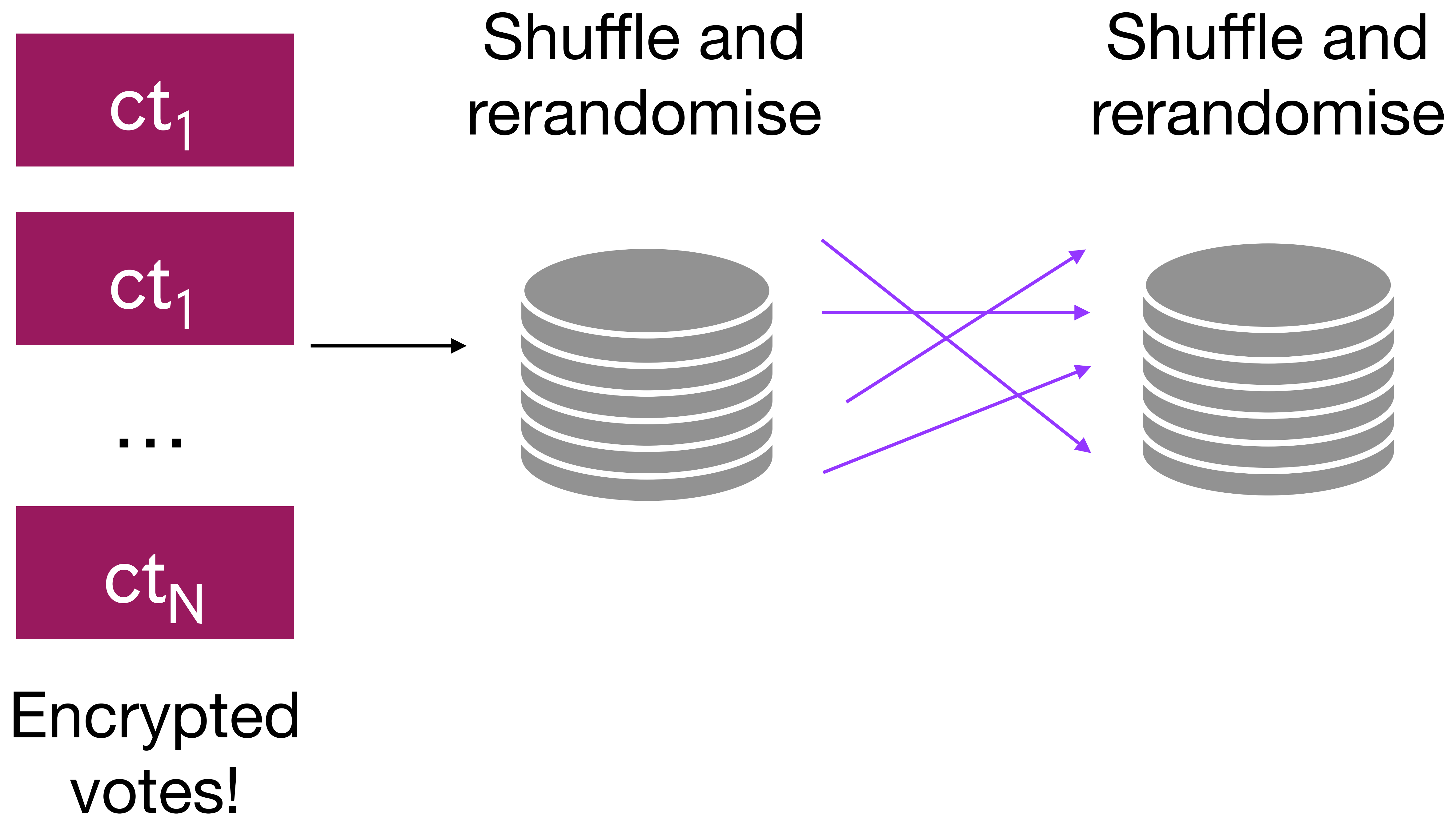


Encrypted votes!

Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

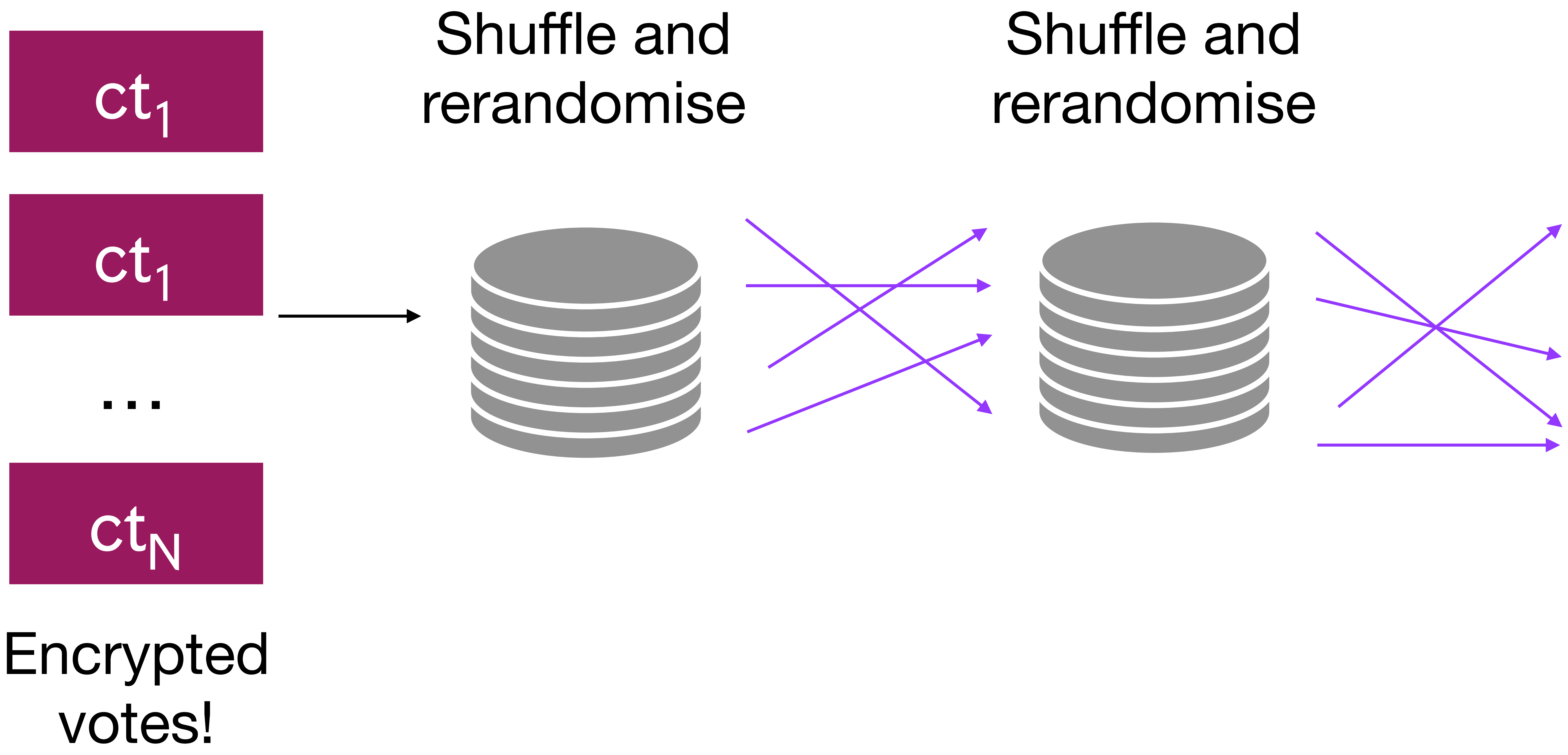
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

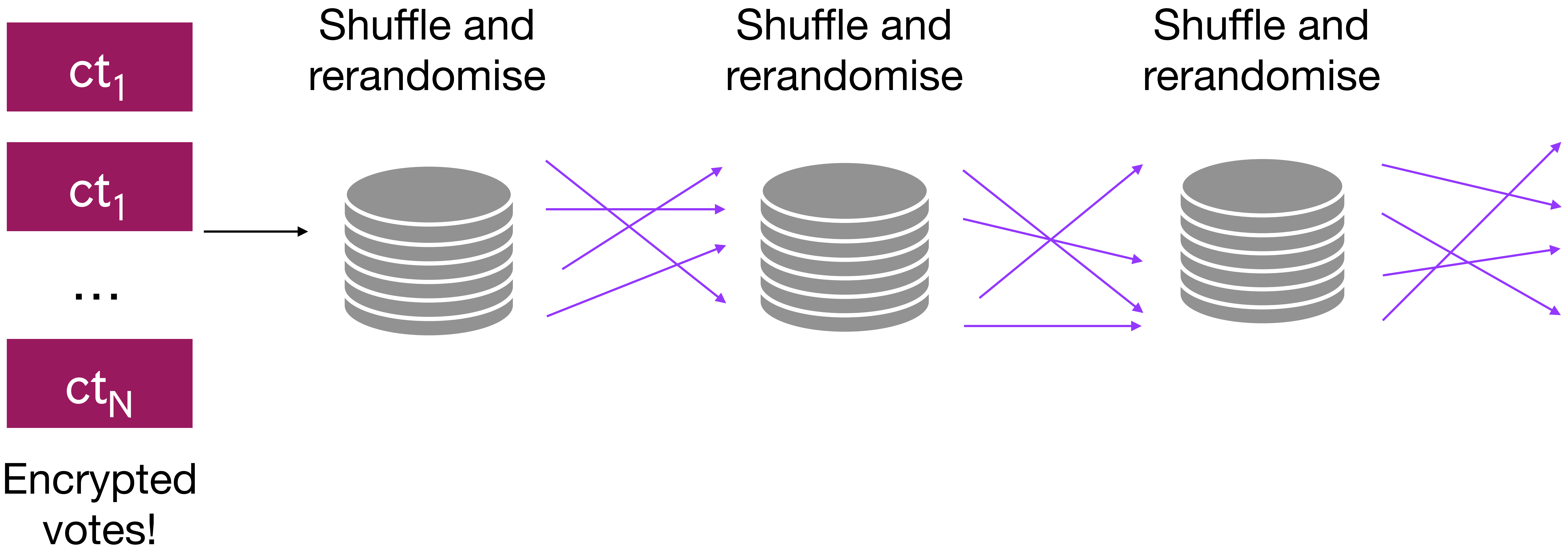
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

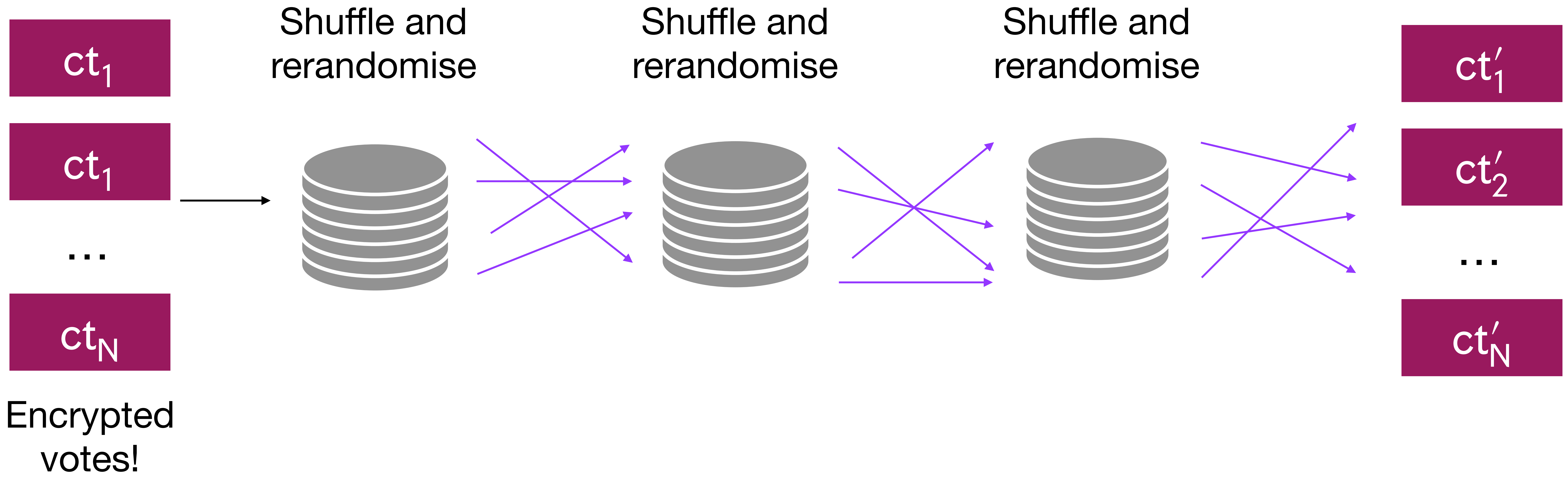
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

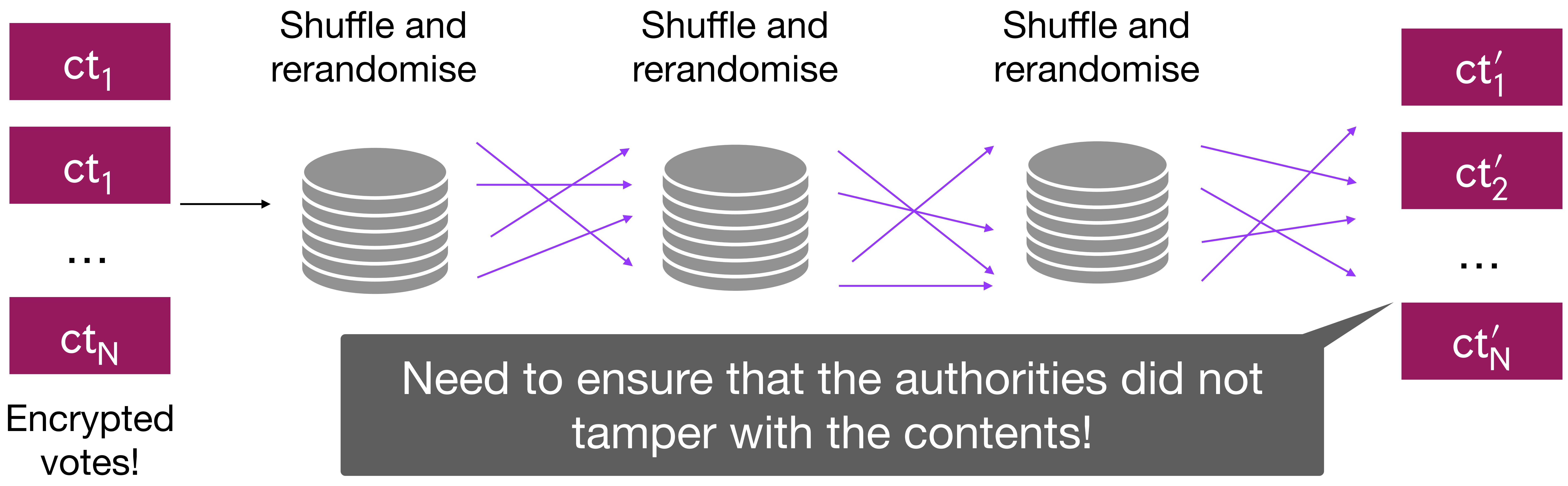
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

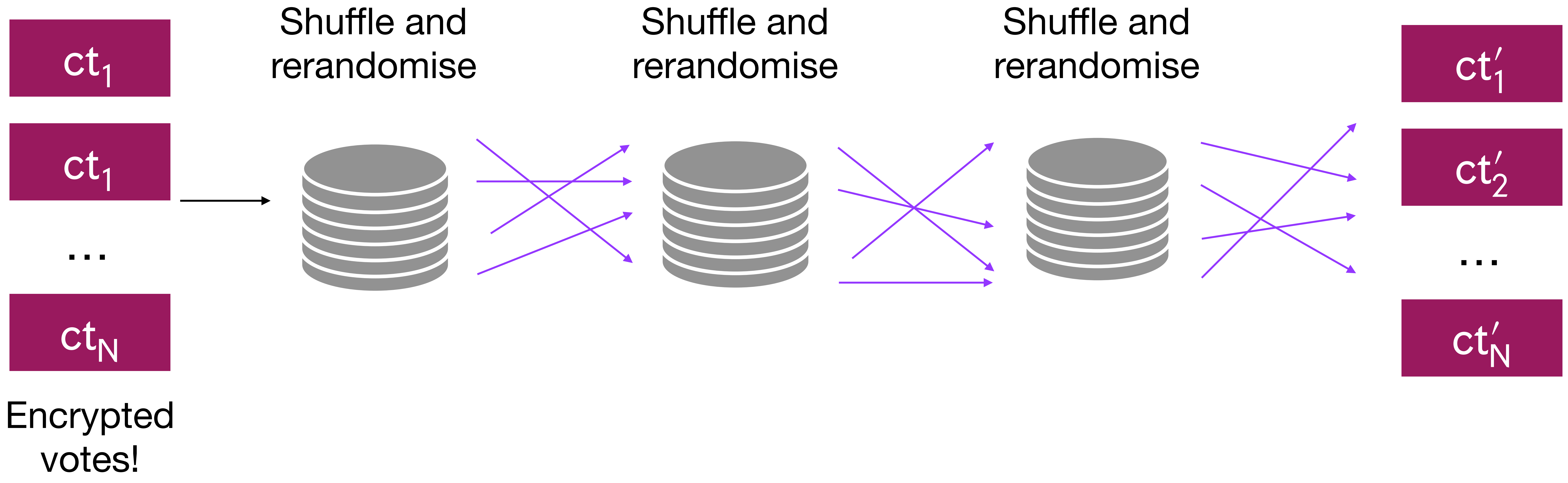
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

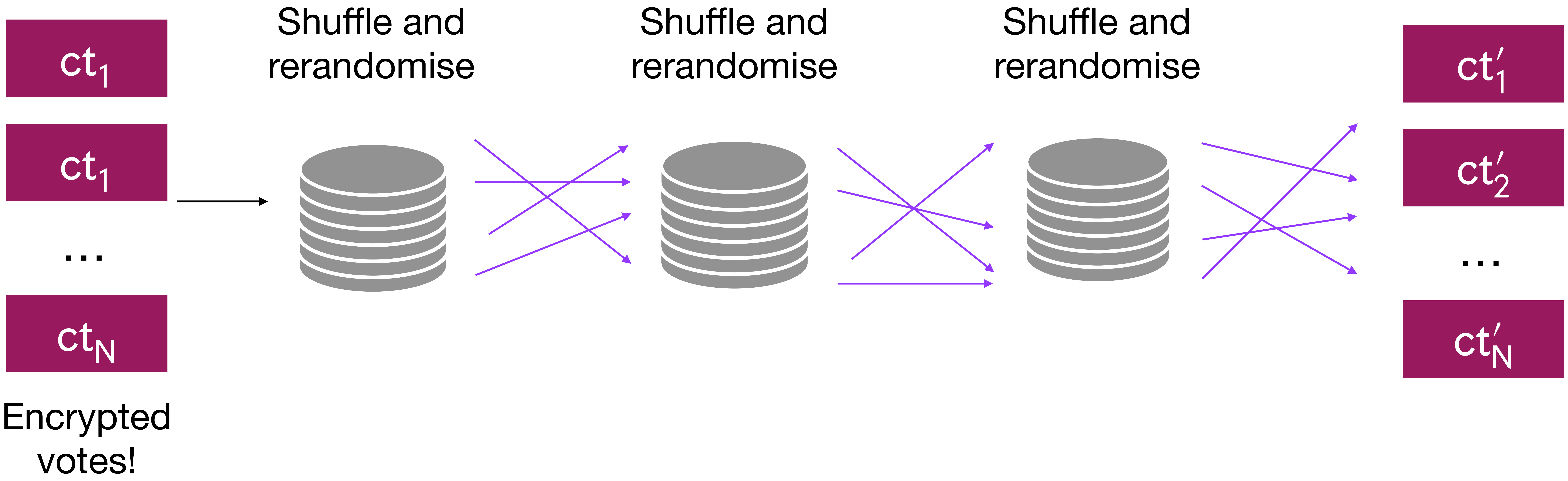
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

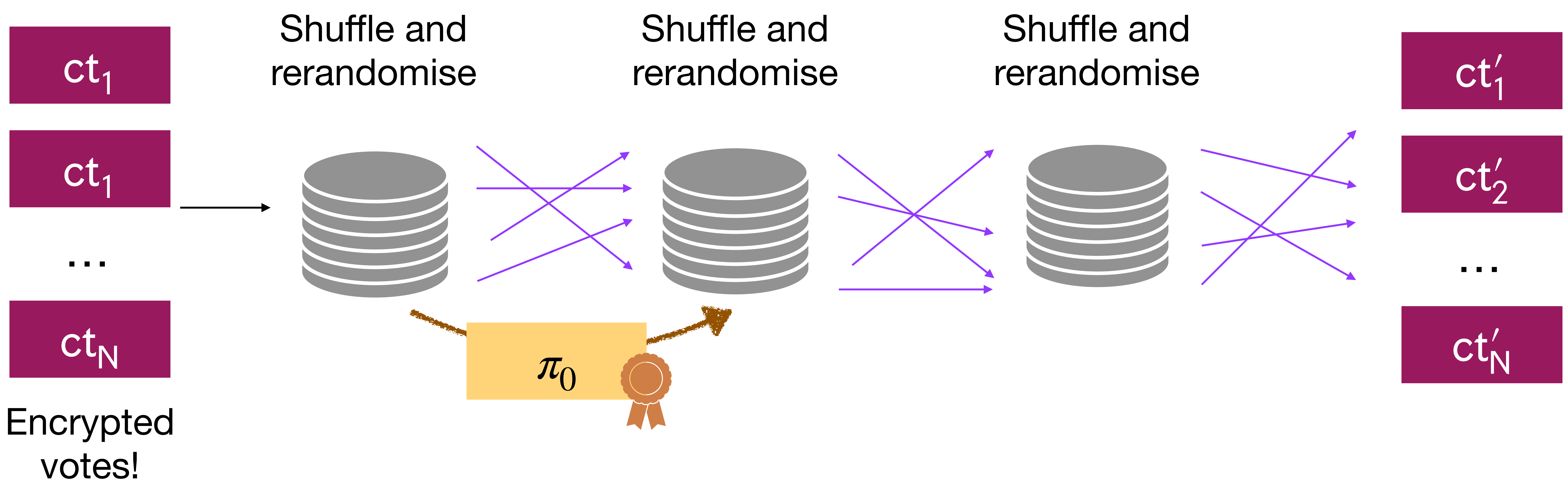
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.
- Use ZK-IVC to **verify that the final list** is honest without reading L NIZKs!



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

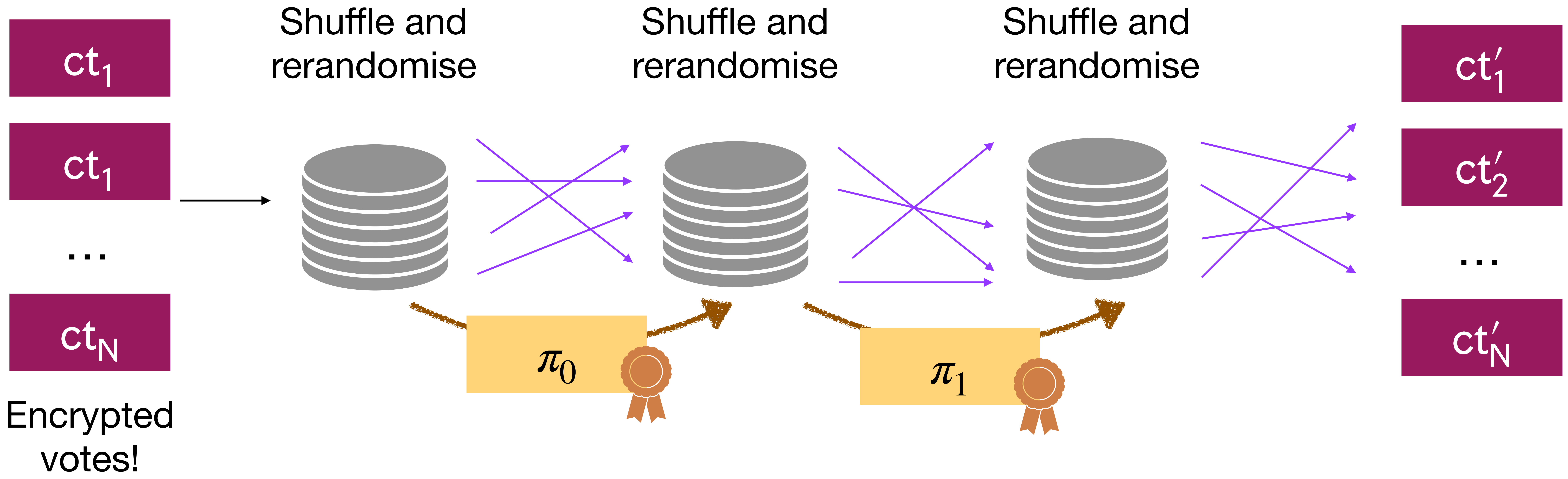
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.
- Use ZK-IVC to **verify that the final list** is honest without reading L NIZKs!



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

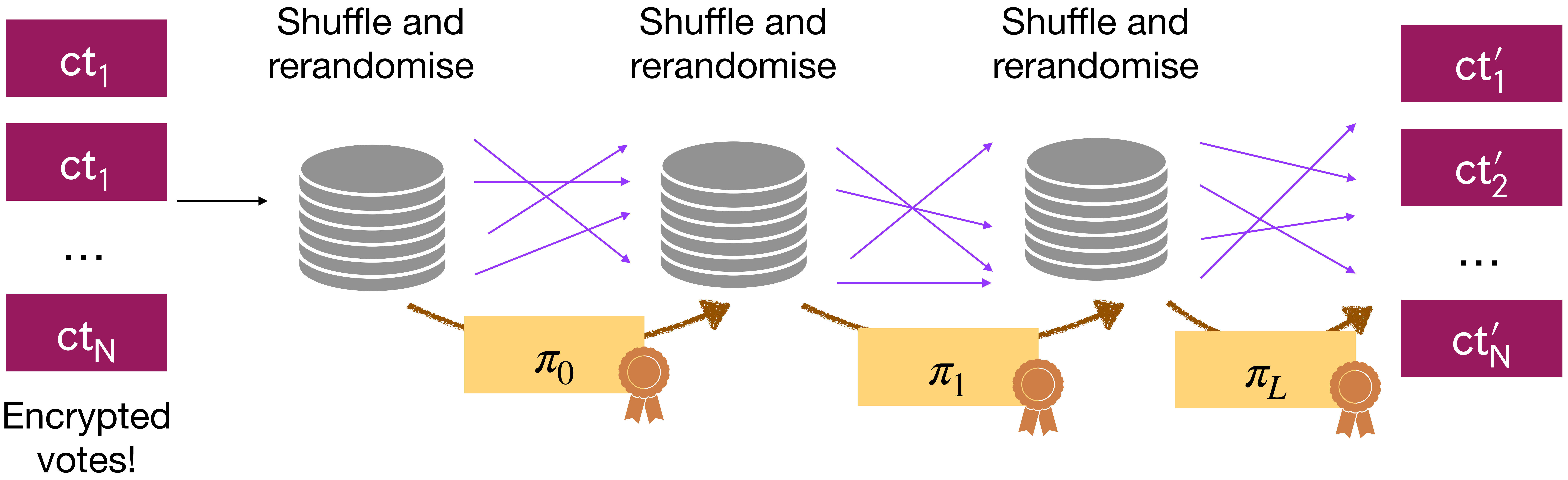
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.
- Use ZK-IVC to **verify that the final list** is honest without reading L NIZKs!



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

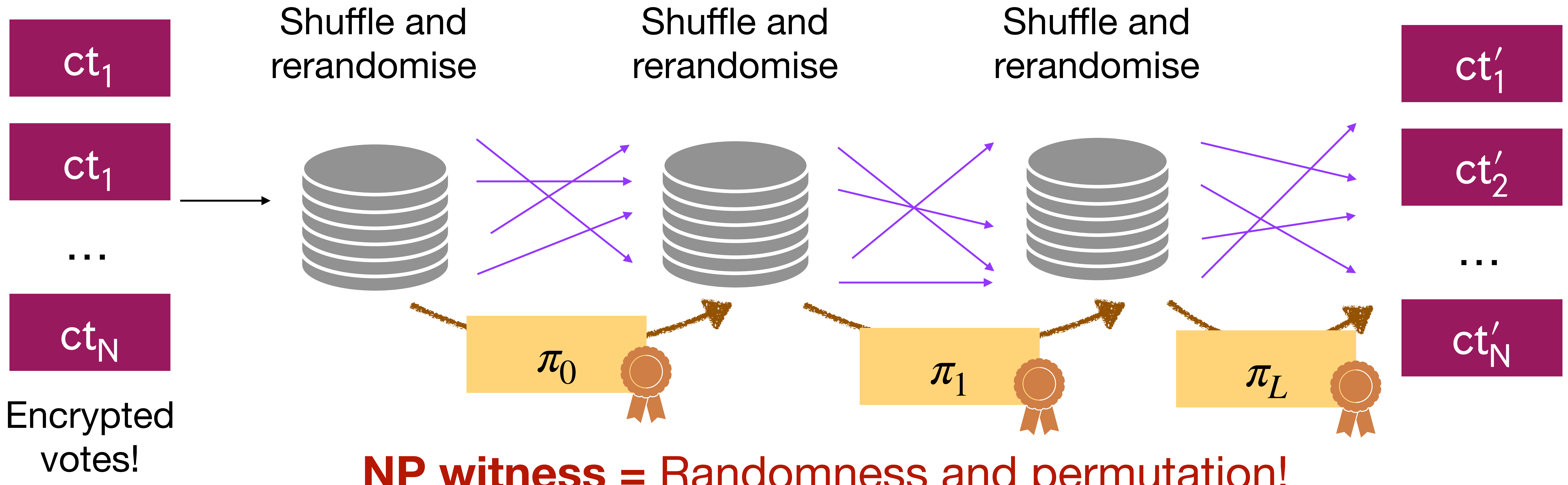
- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.
- Use ZK-IVC to **verify that the final list** is honest without reading L NIZKs!



Application: Verifiable Shuffling

[GL07, CKLM12, CKLM13]

- N parties encrypt **votes** under a rerandomizable scheme.
- L authorities **shuffle and rerandomise** the ciphertext.
- Use ZK-IVC to **verify that the final list** is honest without reading L NIZKs!



NP witness = Randomness and permutation!

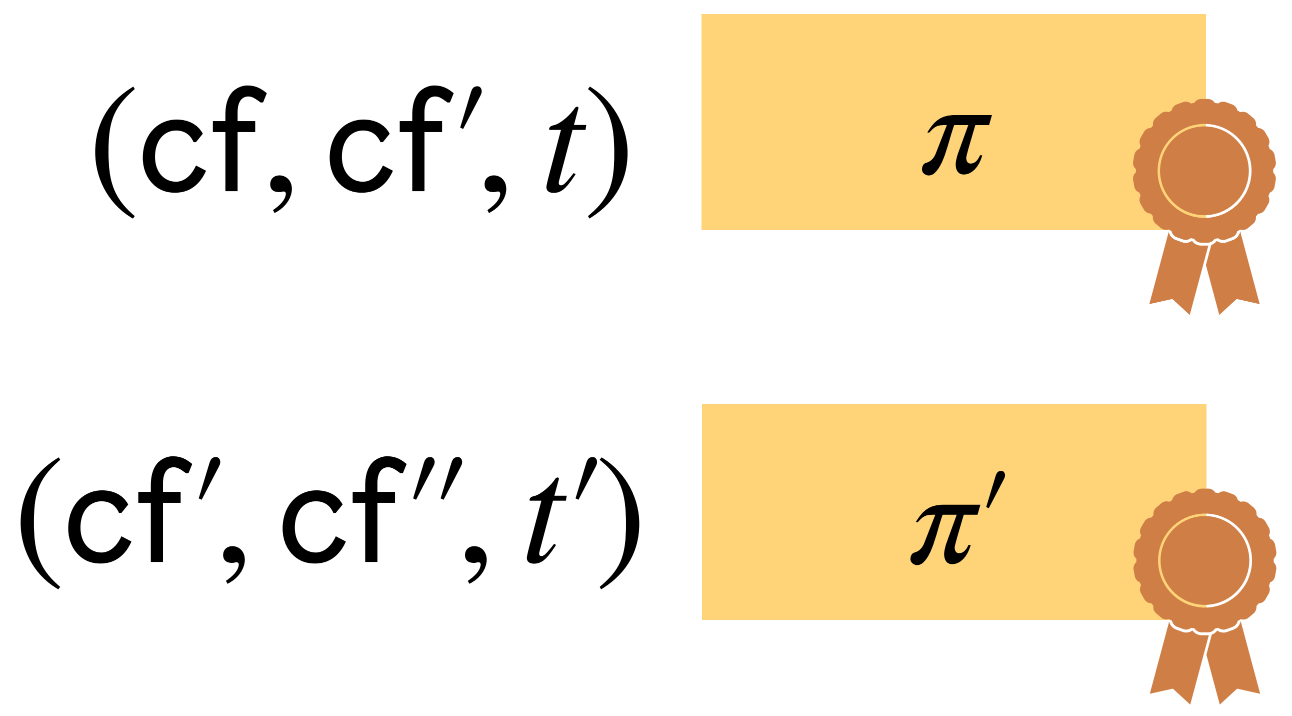
How do we construct IVC?

Valiant's Recipe: Proof Merging

(Prior work)

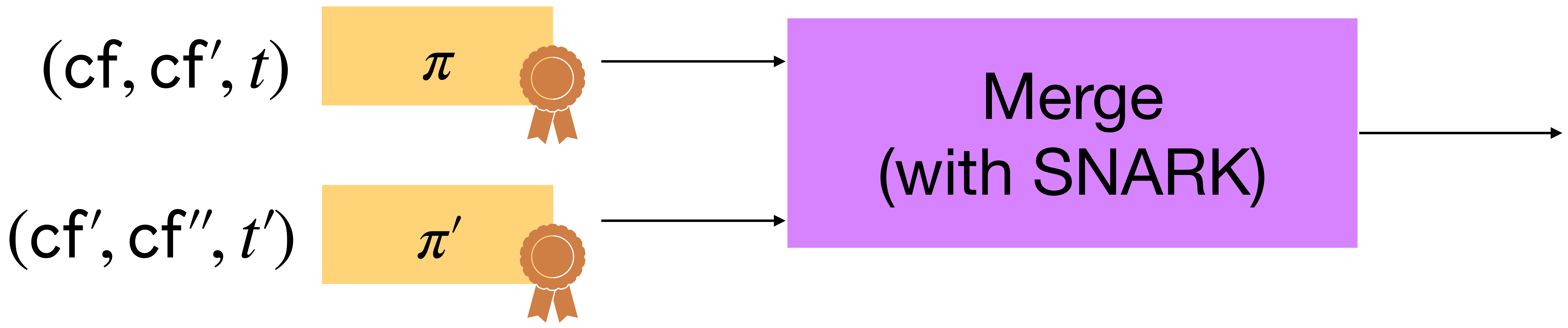
Valiant's Recipe: Proof Merging

(Prior work)



Valiant's Recipe: Proof Merging

(Prior work)



Valiant's Recipe: Proof Merging

(Prior work)

Valiant's Recipe: Proof Merging

(Prior work)

- **Proof of knowledge:** If adversary gives **accepting** $(\mathsf{cf}, \mathsf{cf}'', t + t'), \pi''$, one can **extract** accepting tuples $(\mathsf{cf}, \mathsf{cf}', t), \pi$ and $(\mathsf{cf}', \mathsf{cf}'', t'), \pi'$.

Valiant's Recipe: Proof Merging

(Prior work)

- **Proof of knowledge:** If adversary gives **accepting** $(\mathsf{cf}, \mathsf{cf}'', t + t'), \pi''$, one can **extract** accepting tuples $(\mathsf{cf}, \mathsf{cf}', t), \pi$ and $(\mathsf{cf}', \mathsf{cf}'', t'), \pi'$.

- **Succinctness:** $|\pi''| \approx |\pi|, |\pi'|$

Proof Merging → IVC! "Tree merging"

Level 3

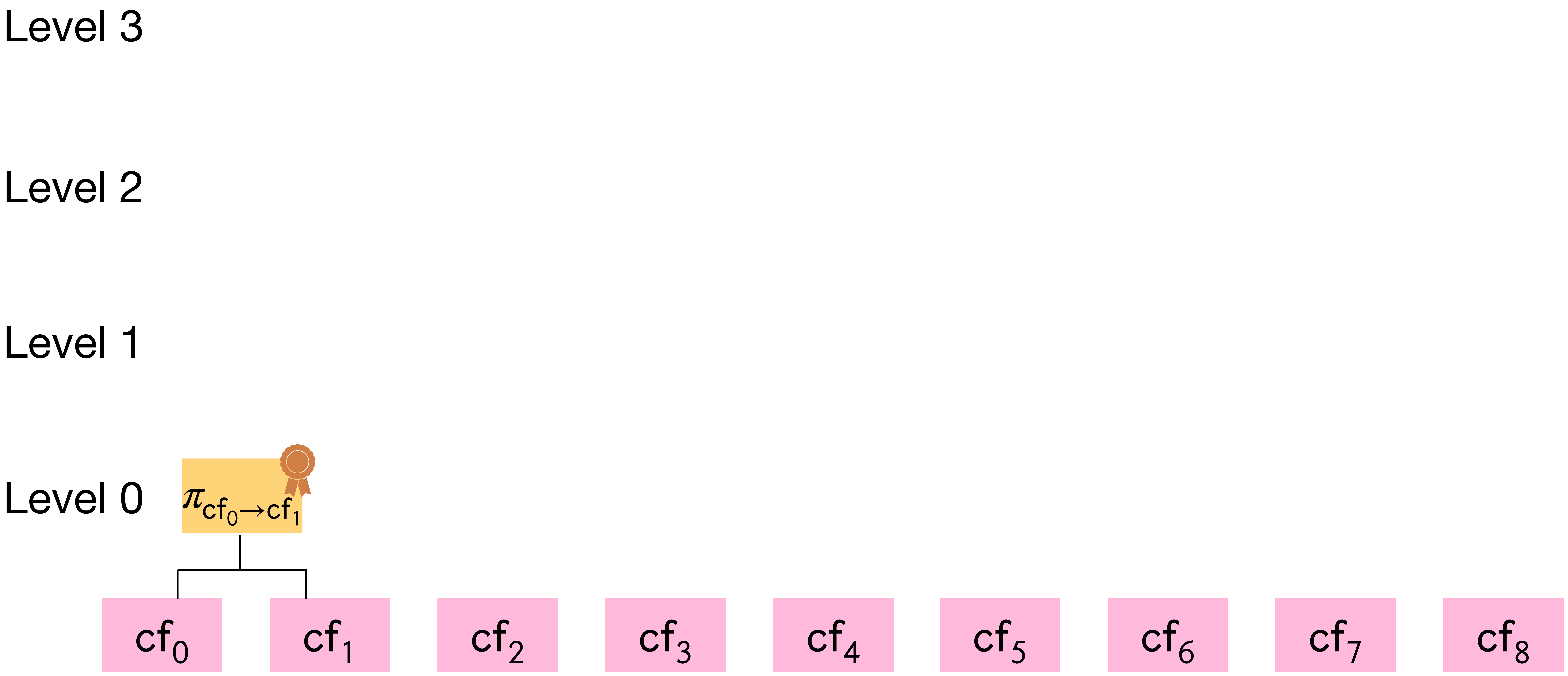
Level 2

Level 1

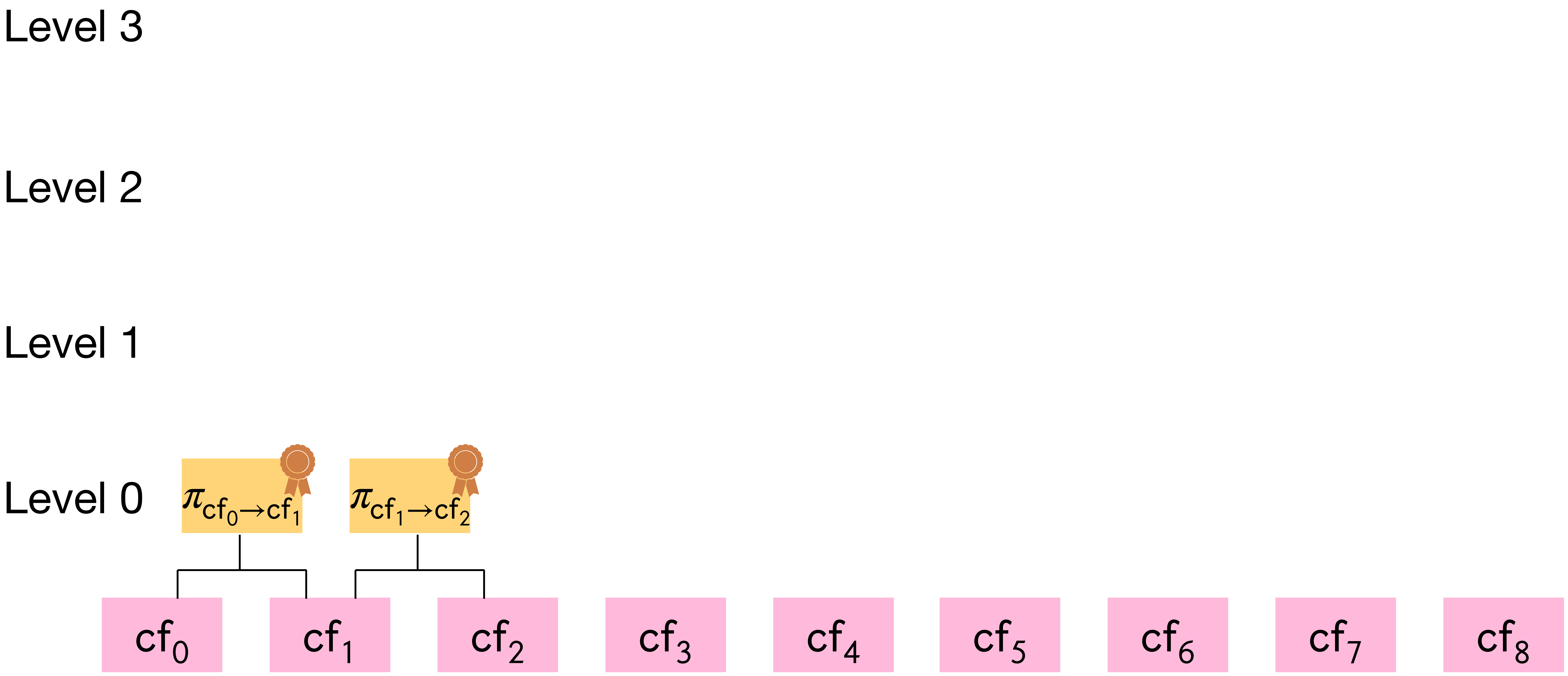
Level 0

cf_0 cf_1 cf_2 cf_3 cf_4 cf_5 cf_6 cf_7 cf_8

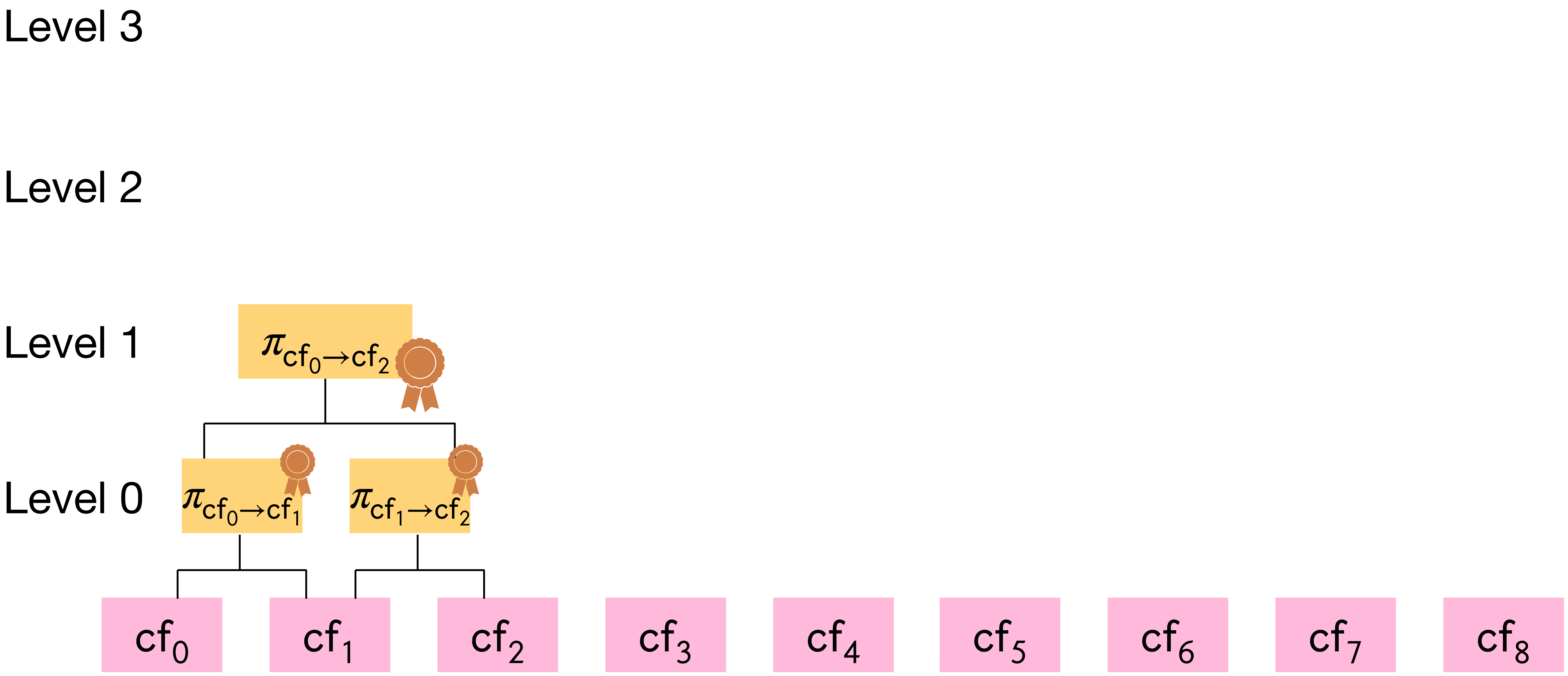
Proof Merging → IVC! “Tree merging”



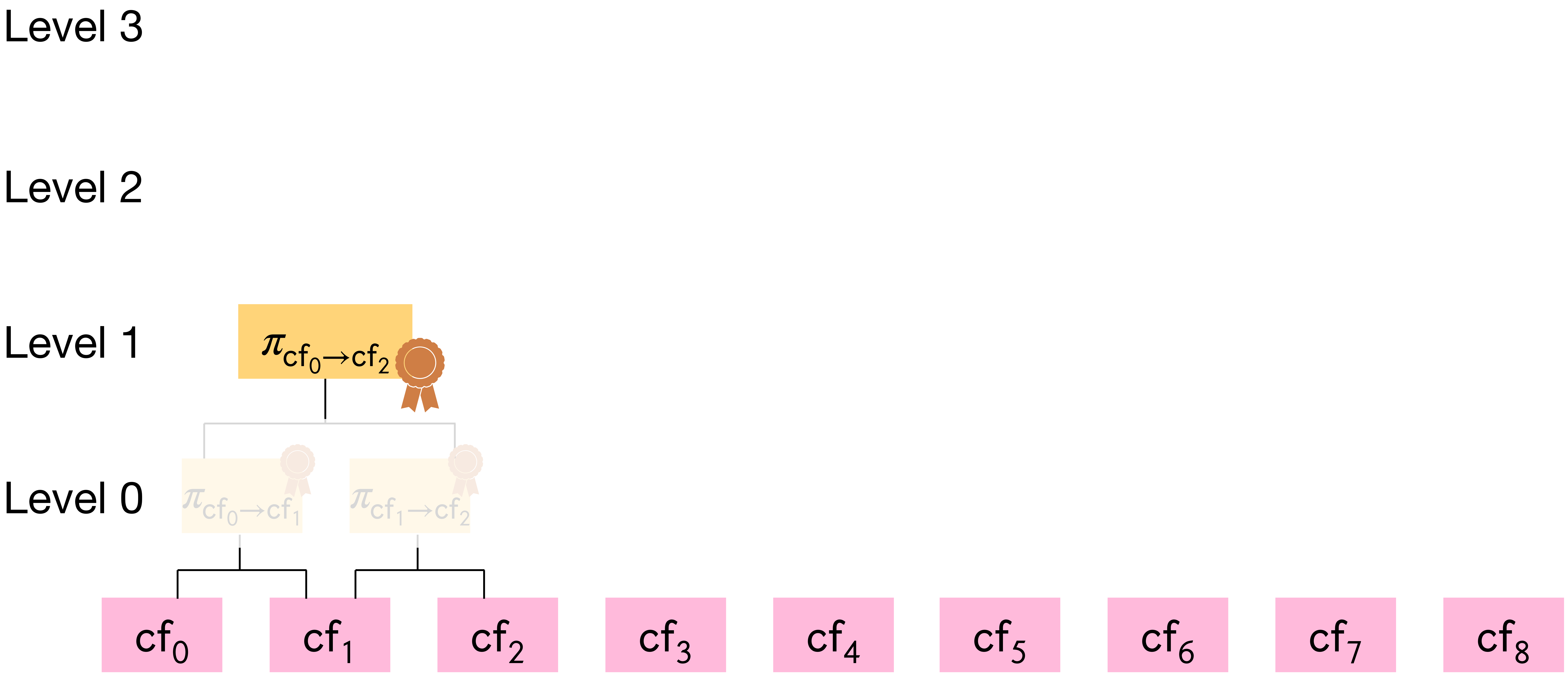
Proof Merging → IVC! "Tree merging"



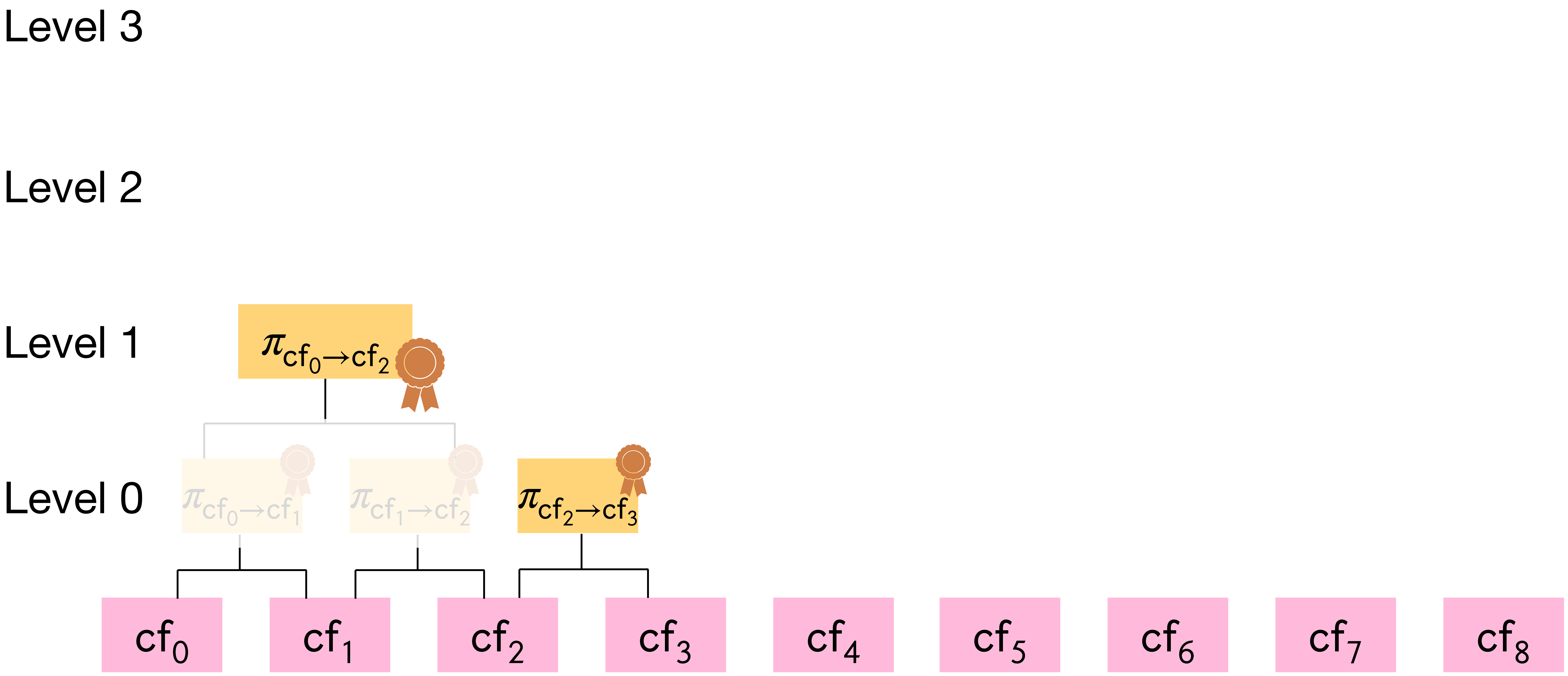
Proof Merging → IVC! "Tree merging"



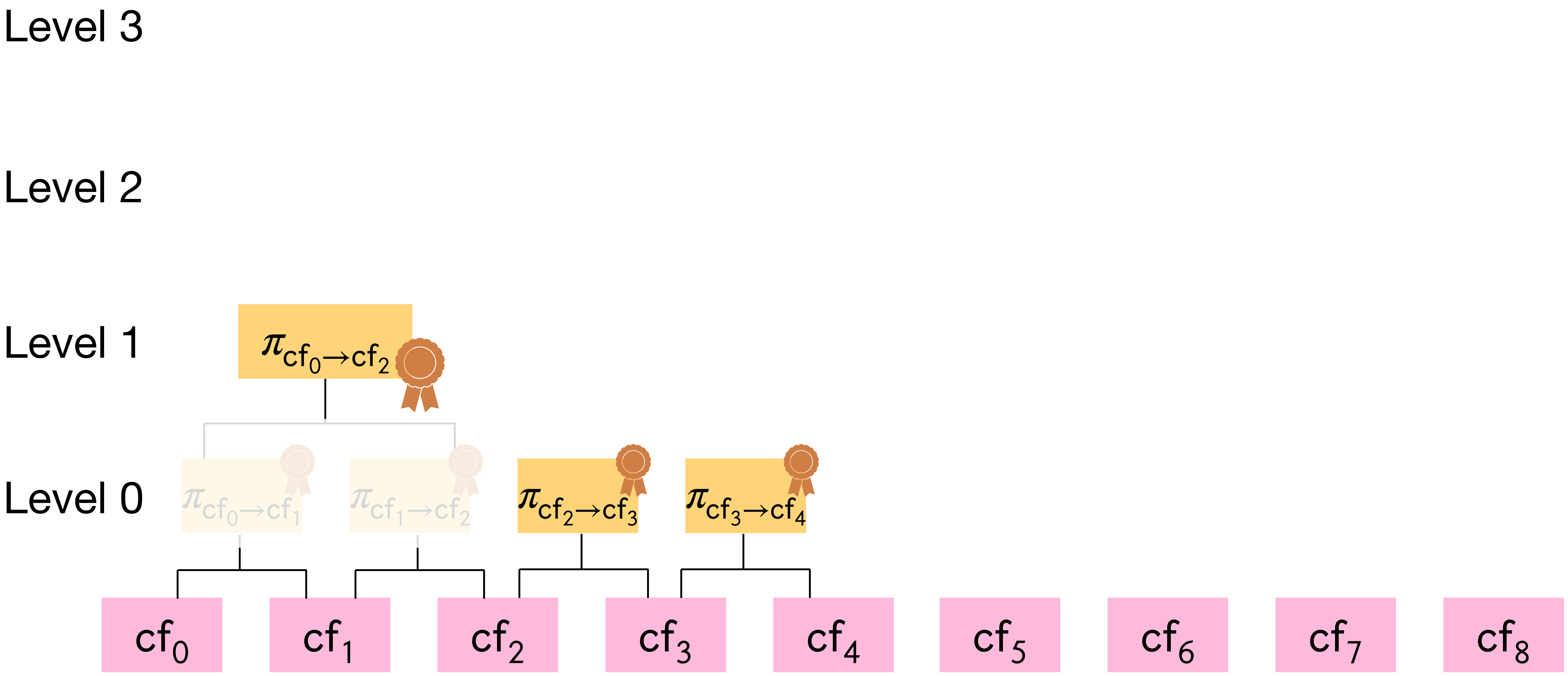
Proof Merging → IVC! "Tree merging"



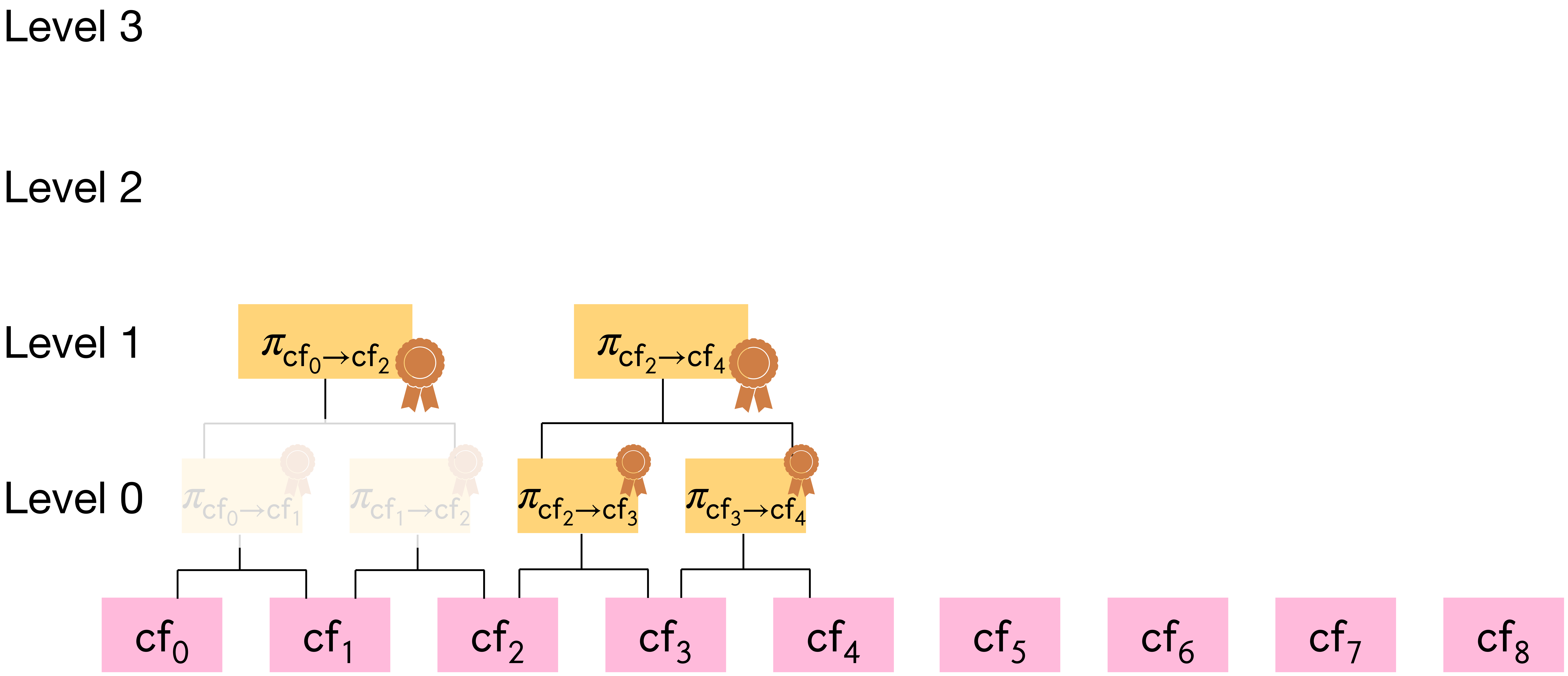
Proof Merging → IVC! “Tree merging”



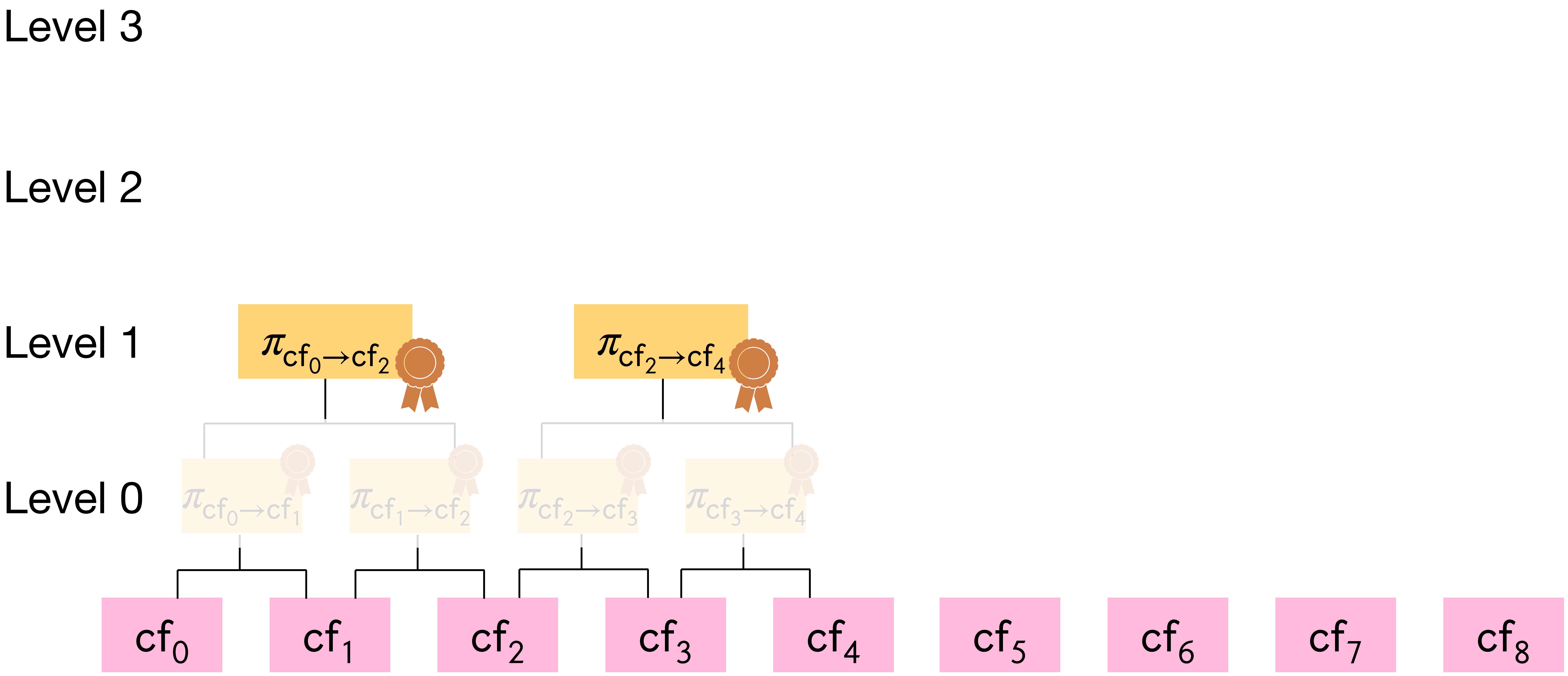
Proof Merging → IVC! "Tree merging"



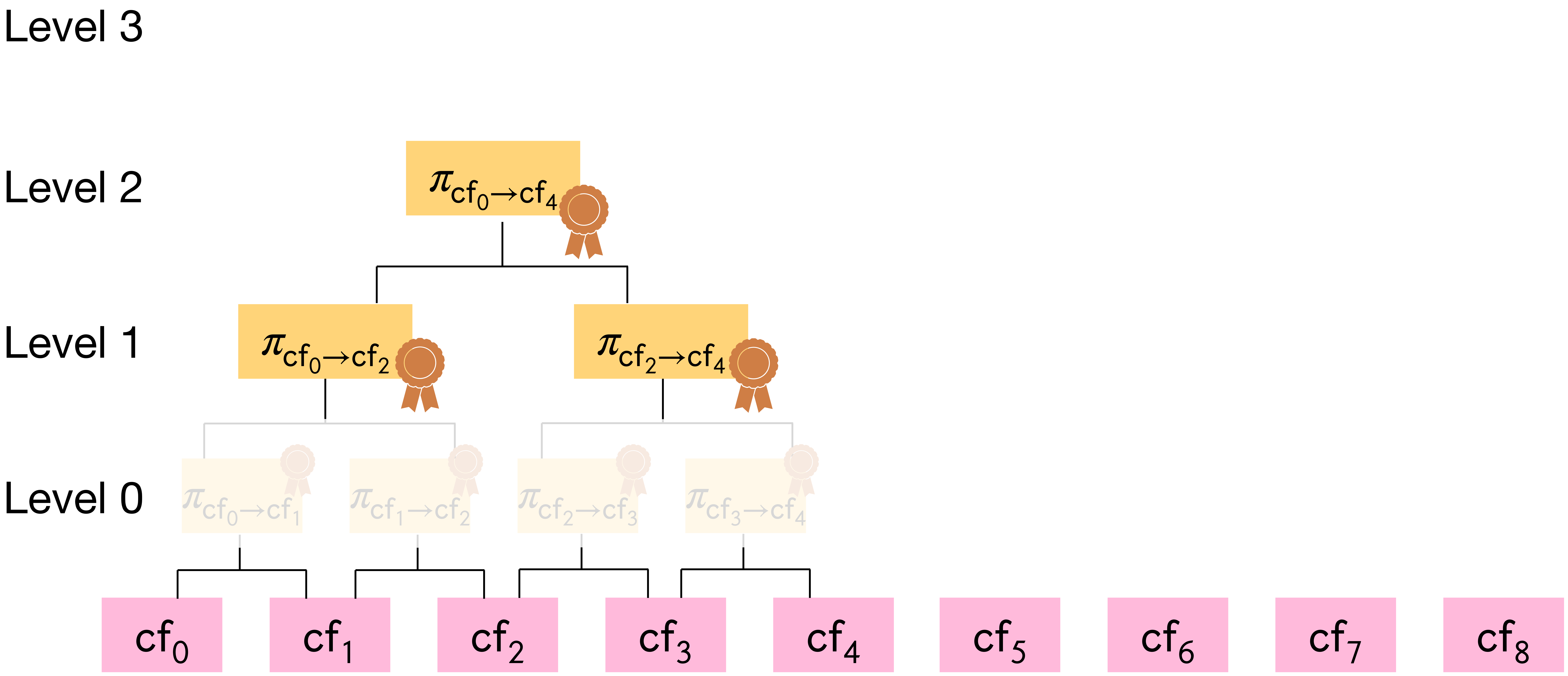
Proof Merging → IVC! "Tree merging"



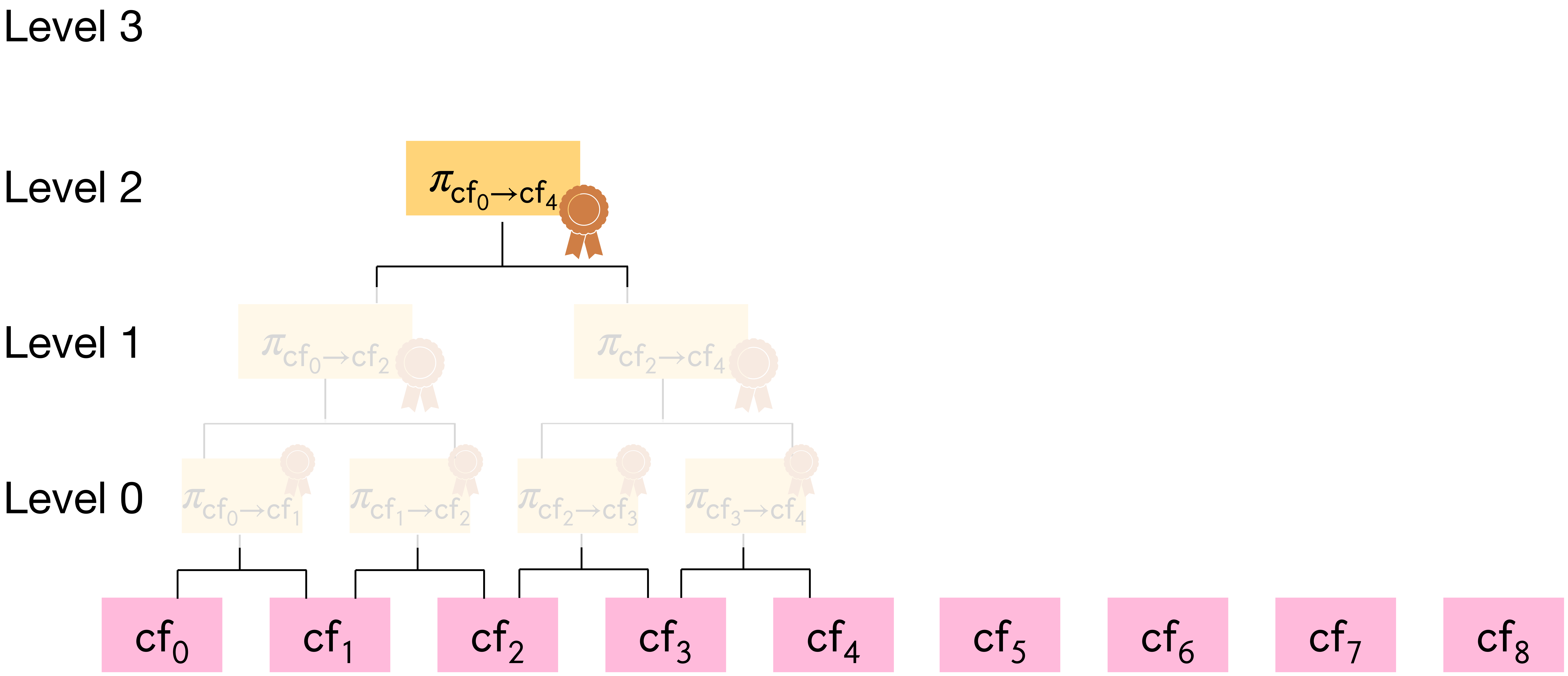
Proof Merging → IVC! “Tree merging”



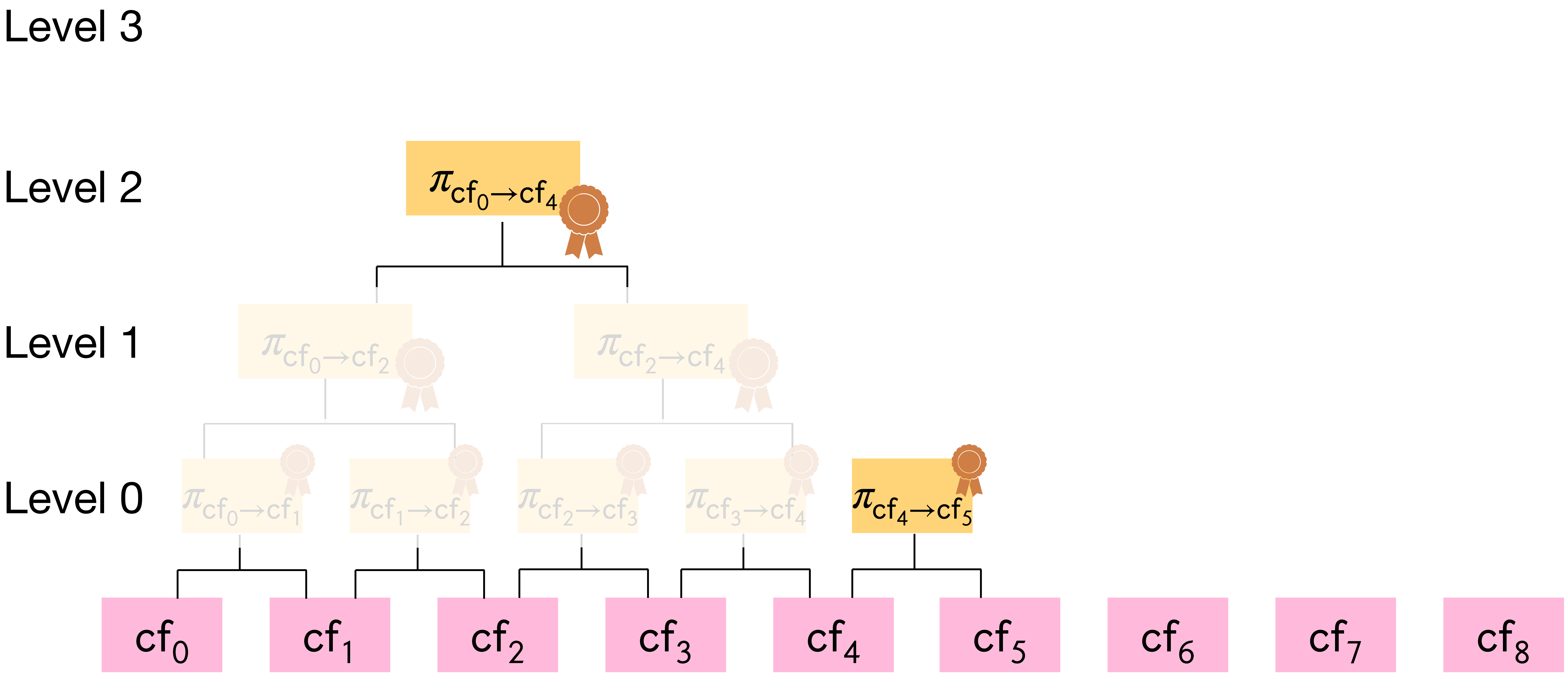
Proof Merging → IVC! "Tree merging"



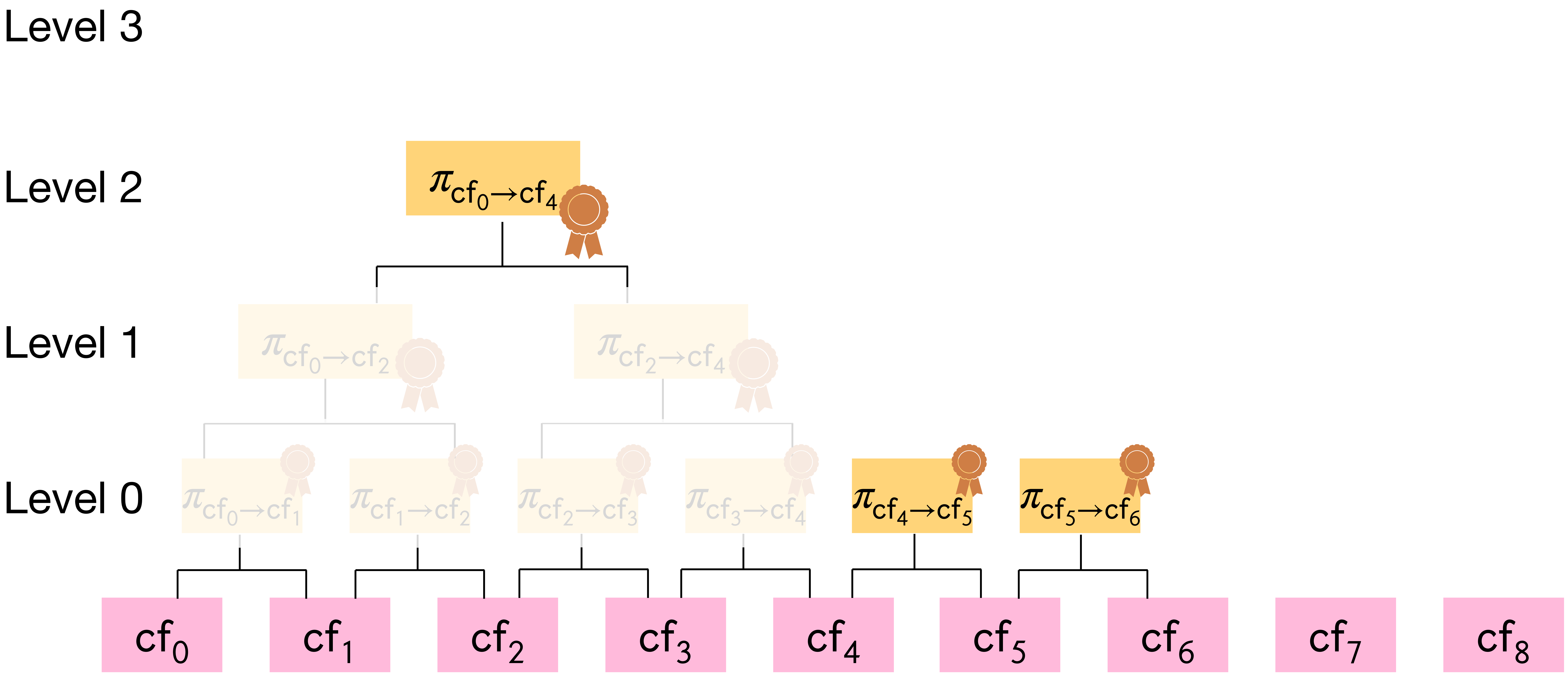
Proof Merging → IVC! "Tree merging"



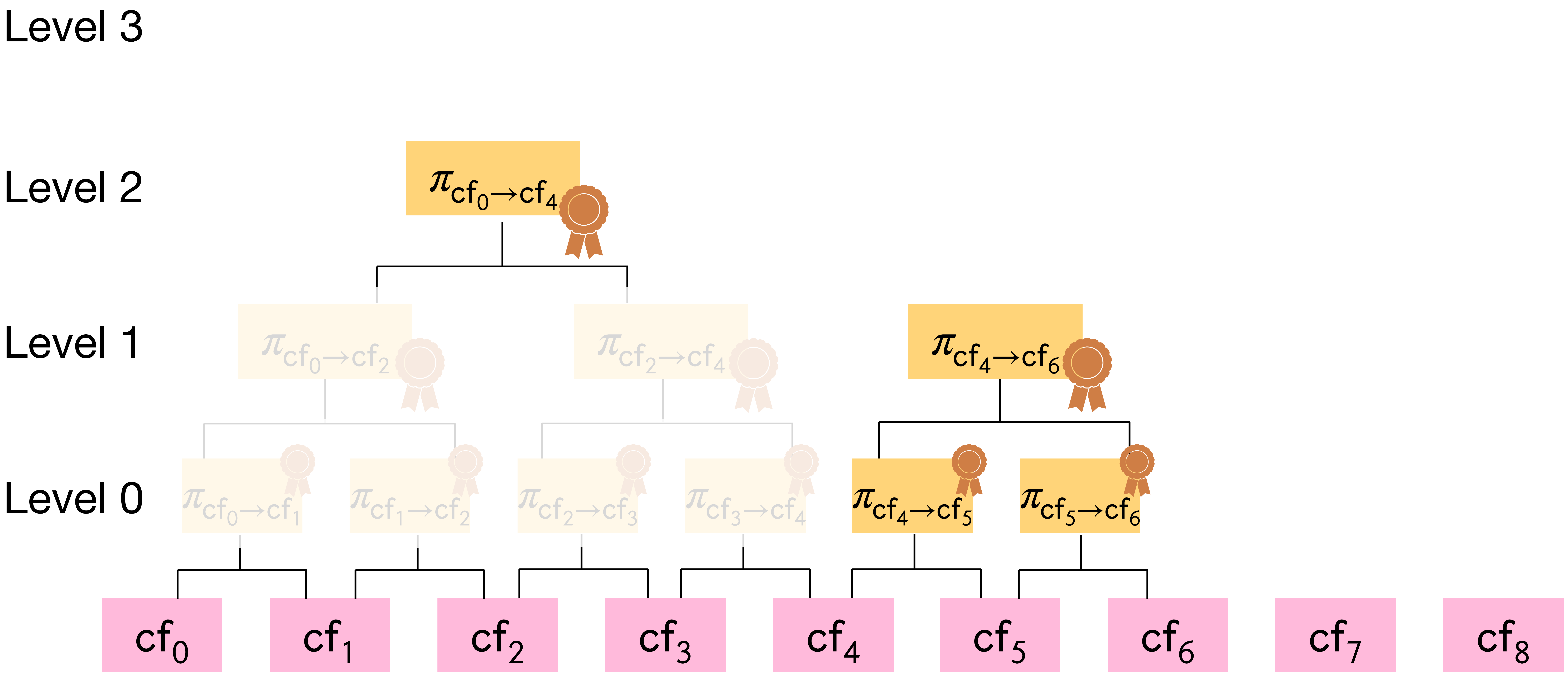
Proof Merging → IVC! "Tree merging"



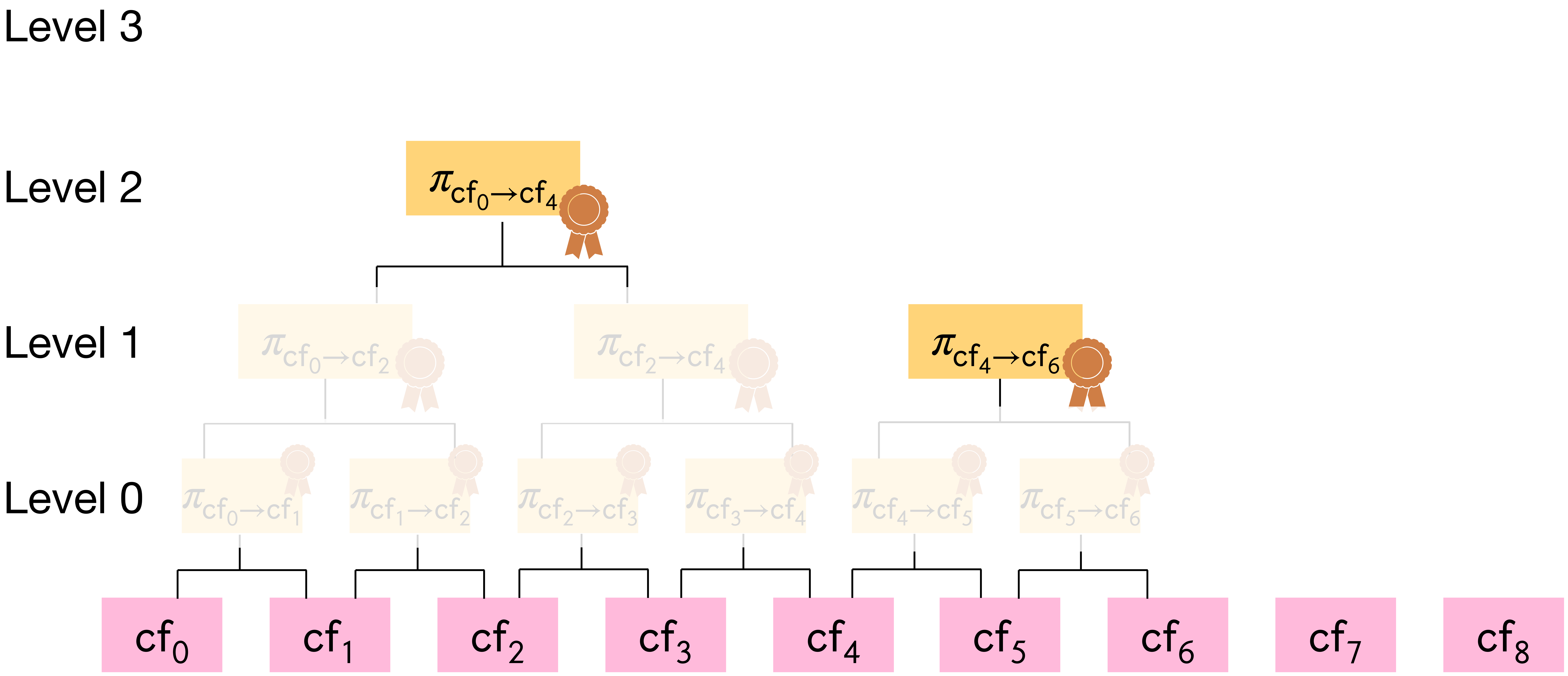
Proof Merging → IVC! “Tree merging”



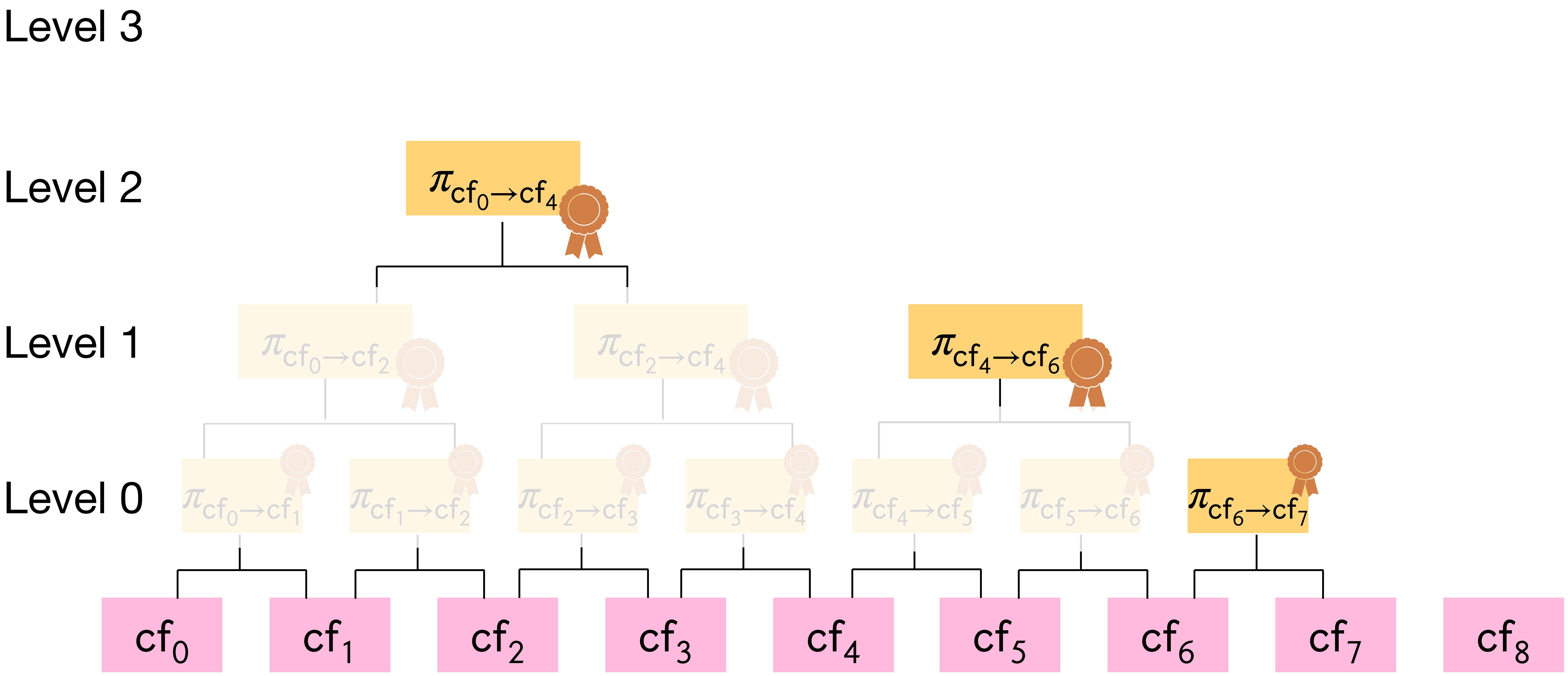
Proof Merging → IVC! "Tree merging"



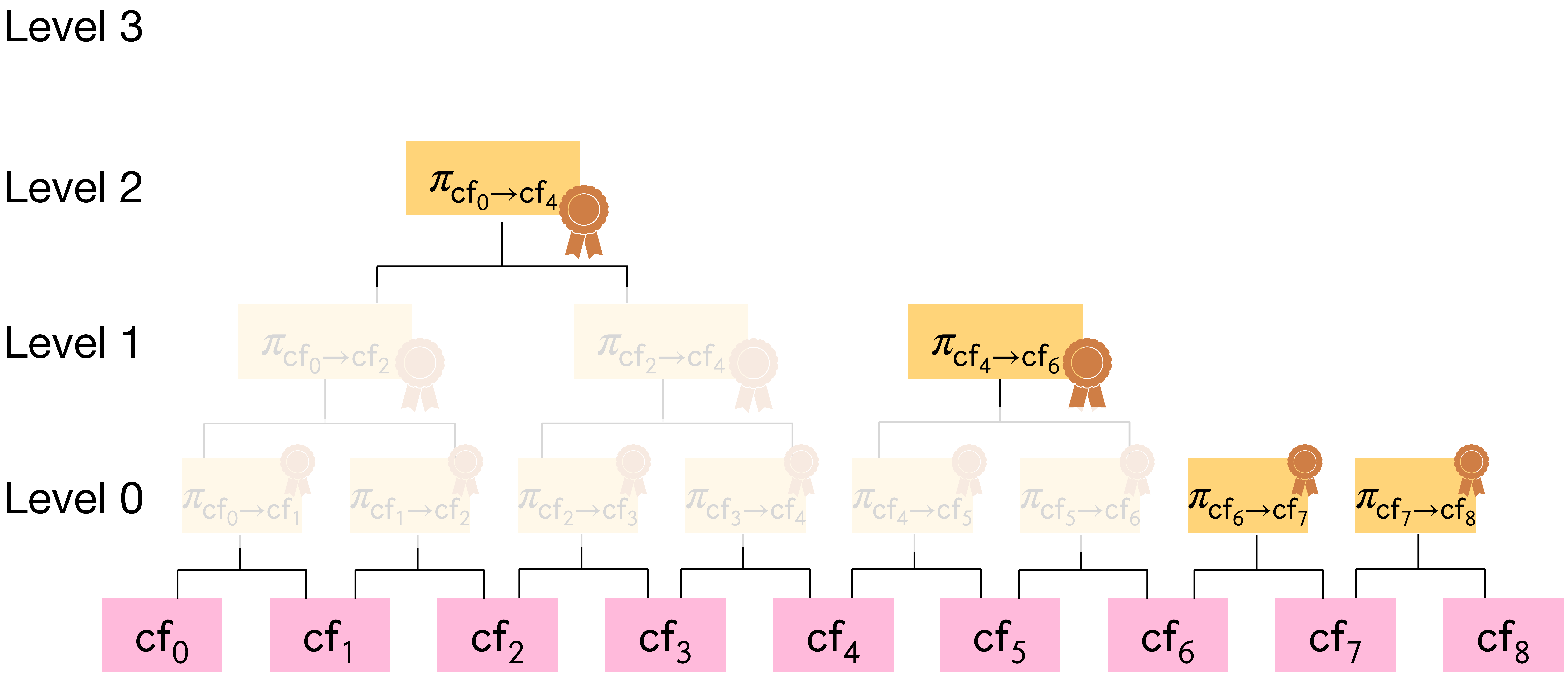
Proof Merging → IVC! “Tree merging”



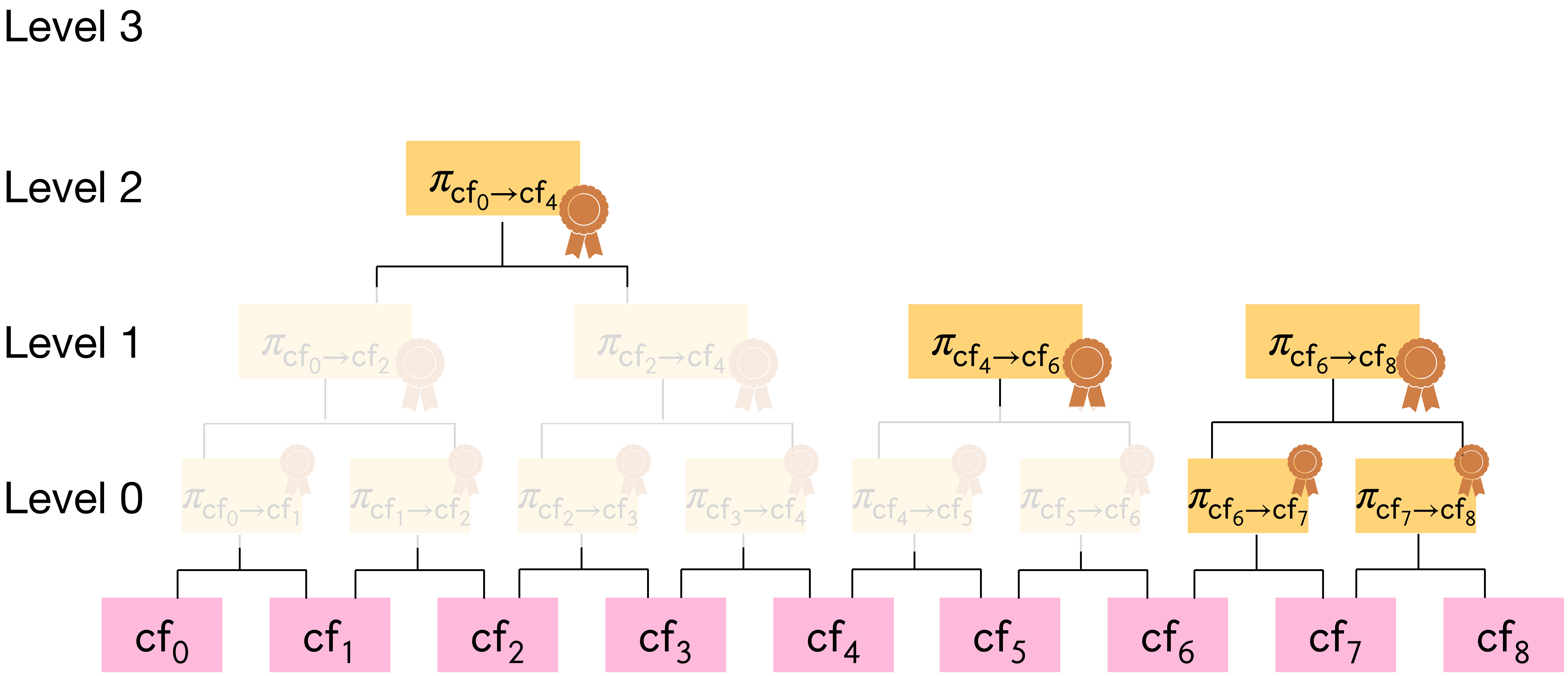
Proof Merging → IVC! "Tree merging"



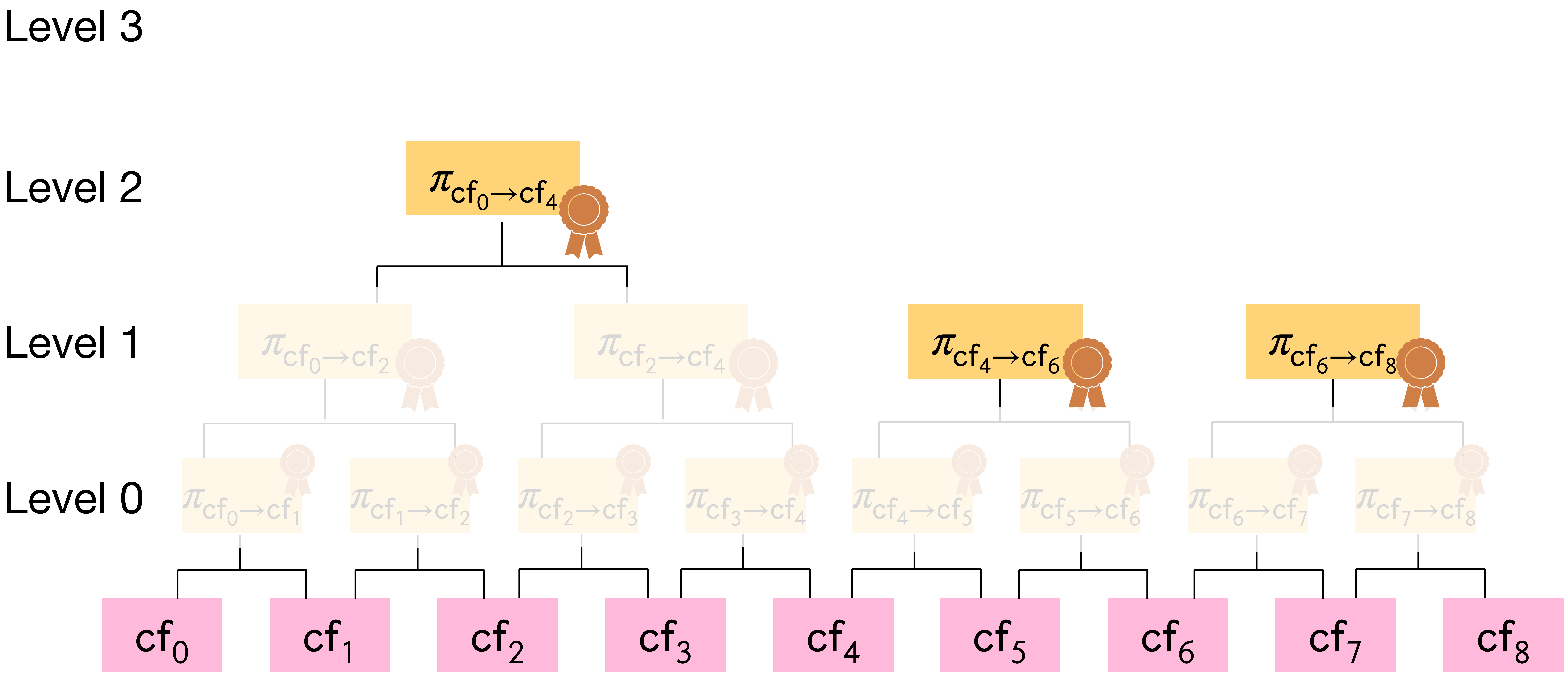
Proof Merging → IVC! “Tree merging”



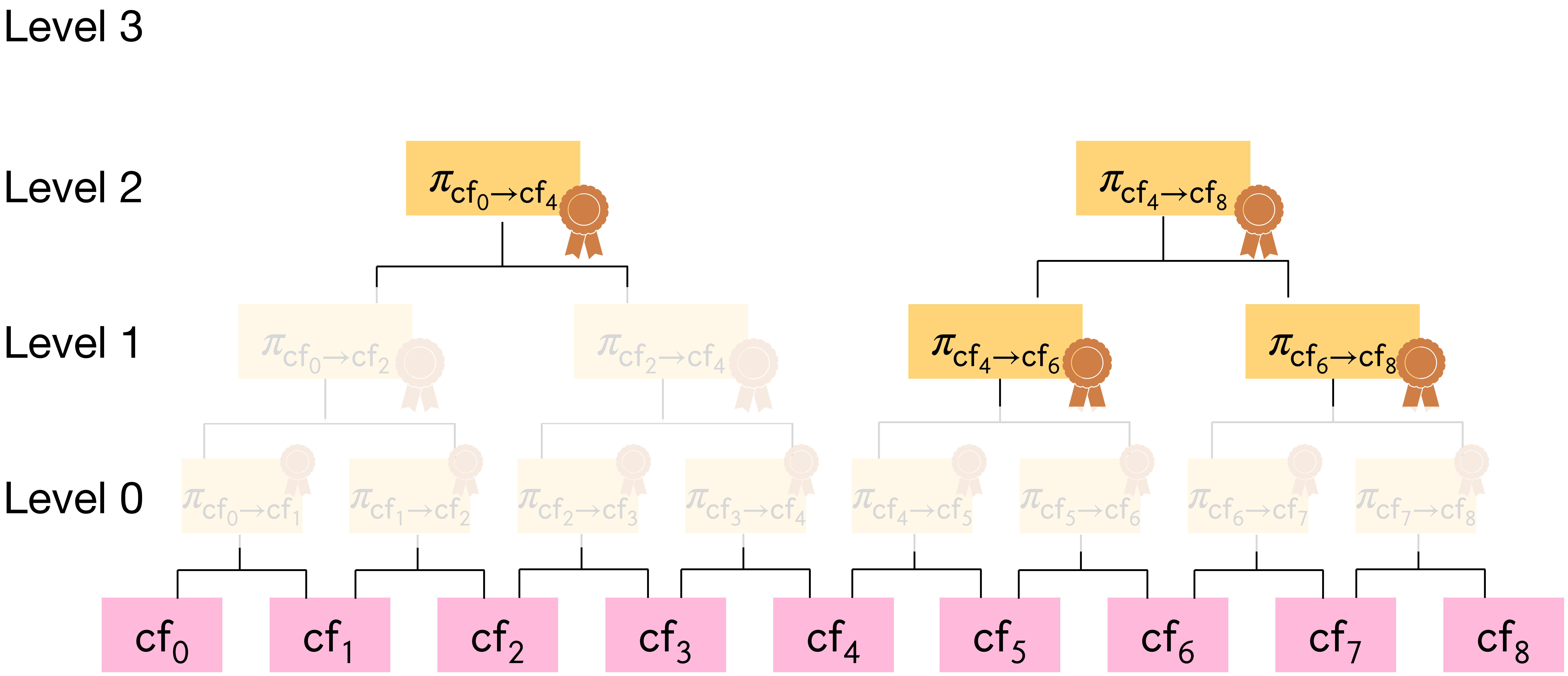
Proof Merging → IVC! “Tree merging”



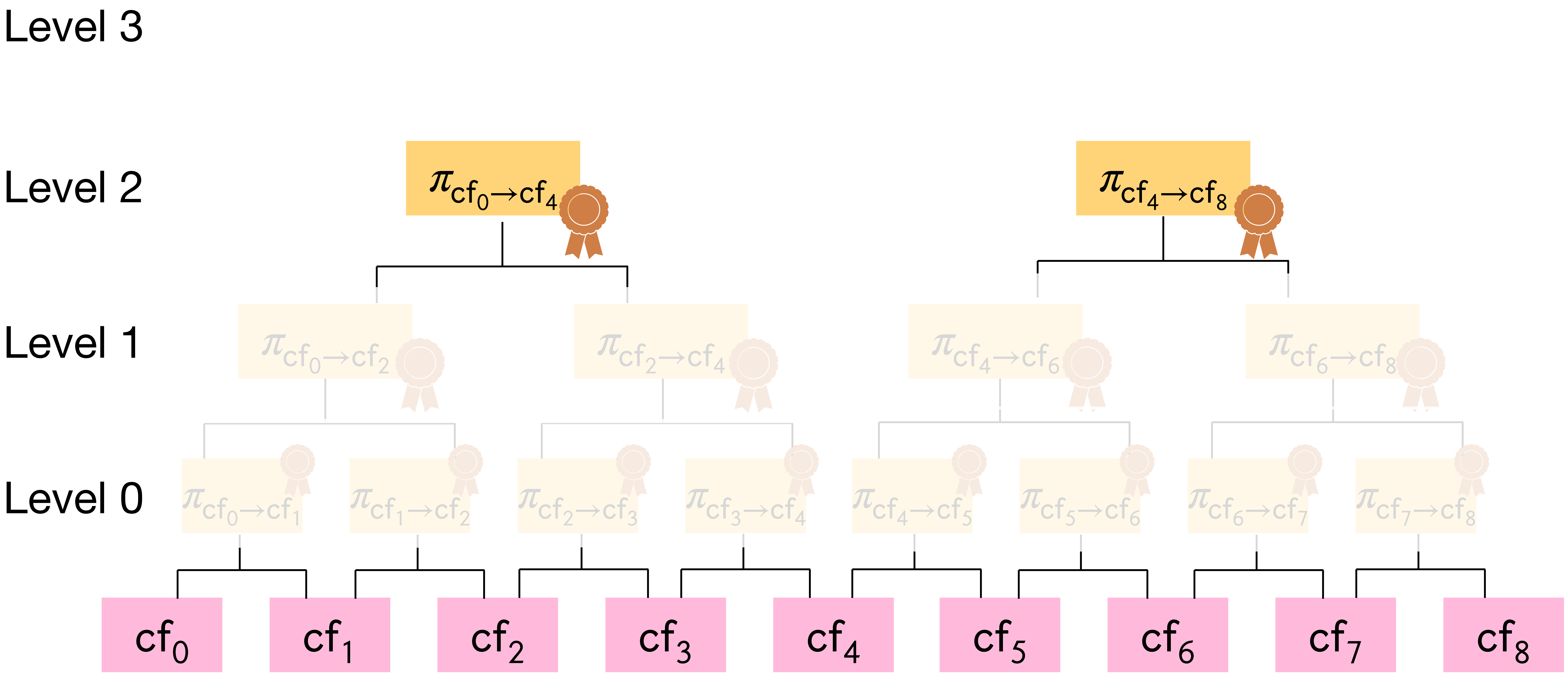
Proof Merging → IVC! “Tree merging”



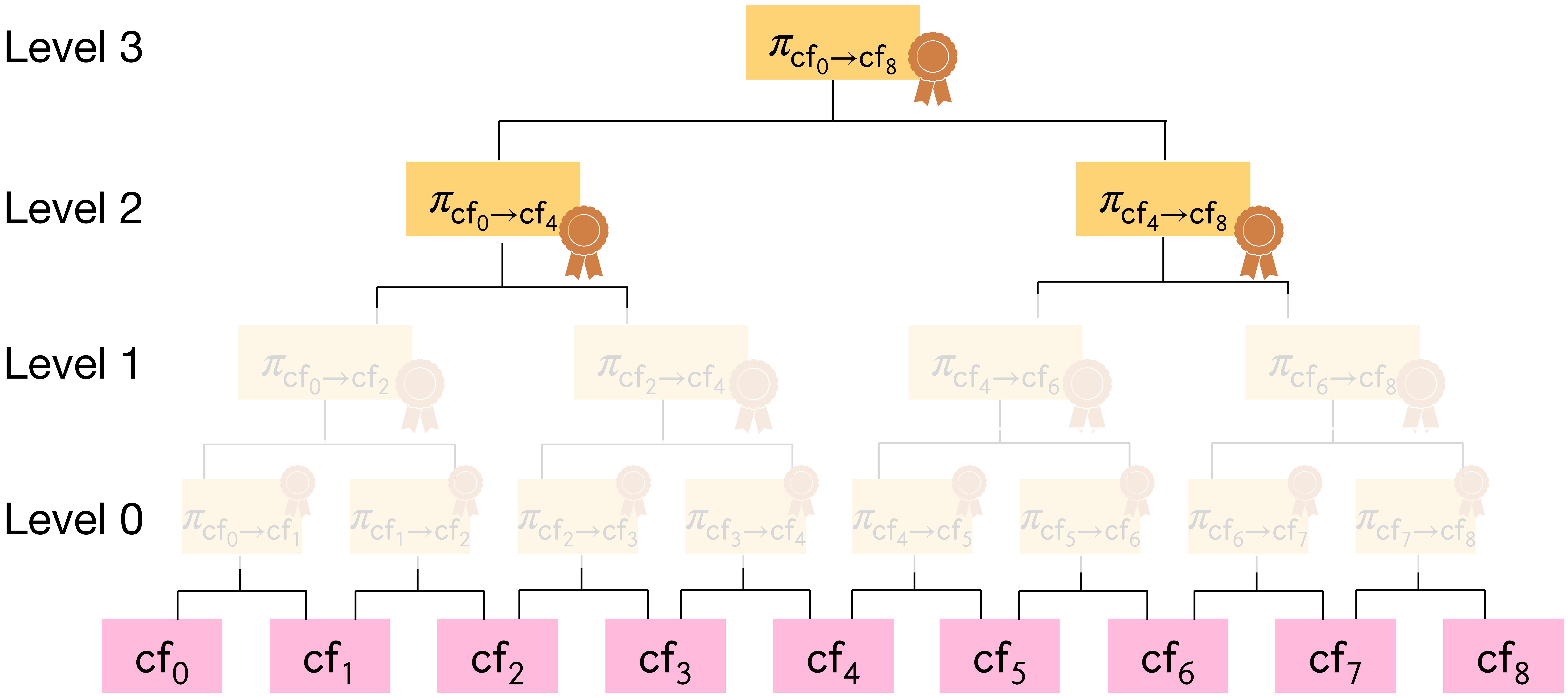
Proof Merging → IVC! "Tree merging"



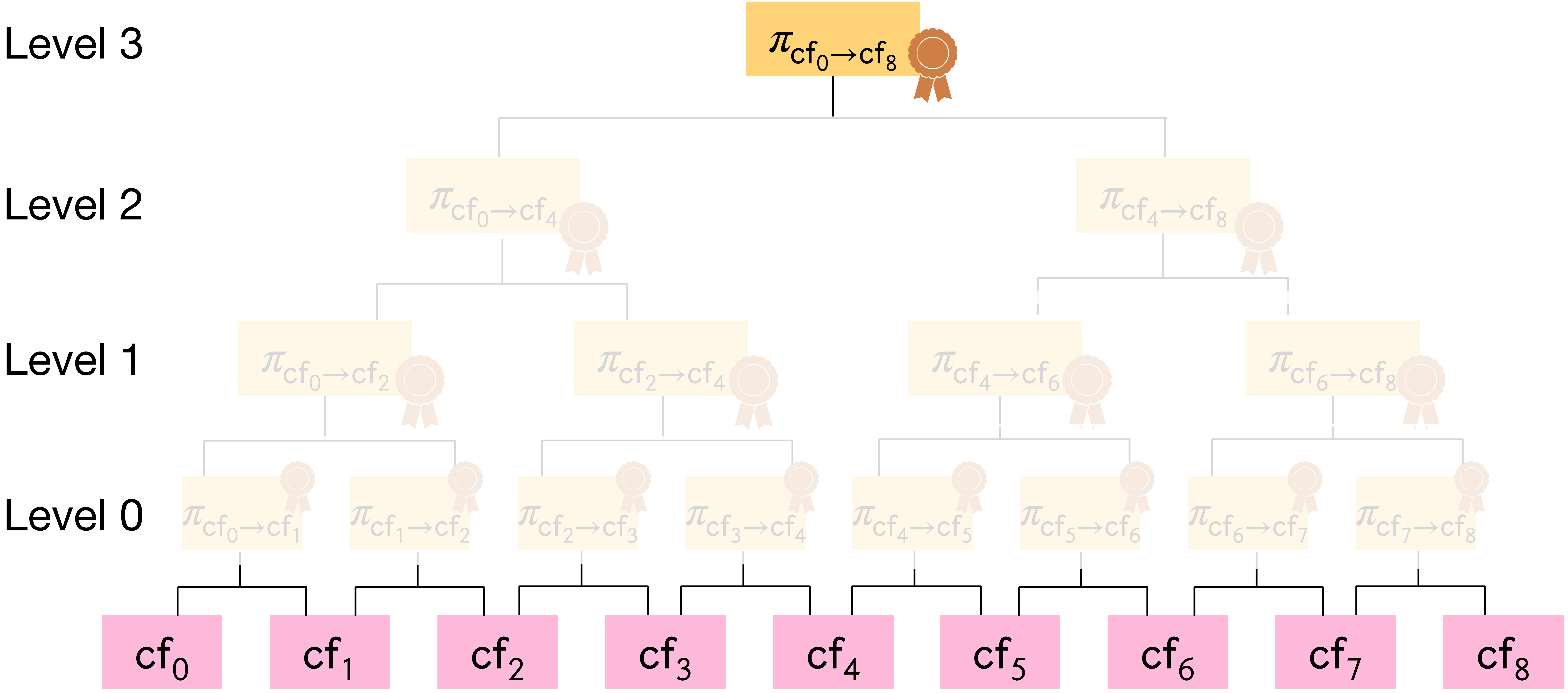
Proof Merging → IVC! “Tree merging”



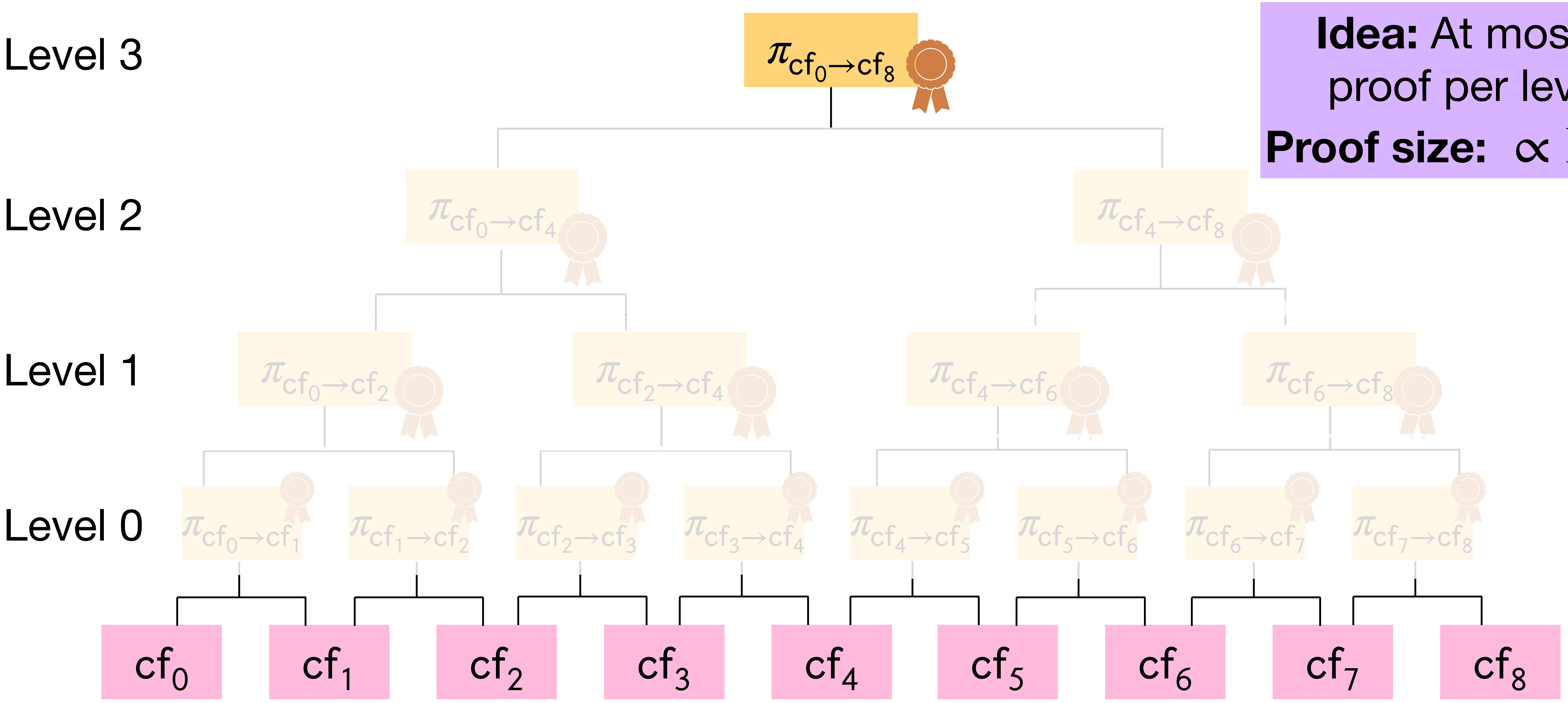
Proof Merging → IVC! "Tree merging"



Proof Merging → IVC! "Tree merging"



Proof Merging → IVC! “Tree merging”



Idea: At most 1 proof per level

Proof size: $\propto \log T$

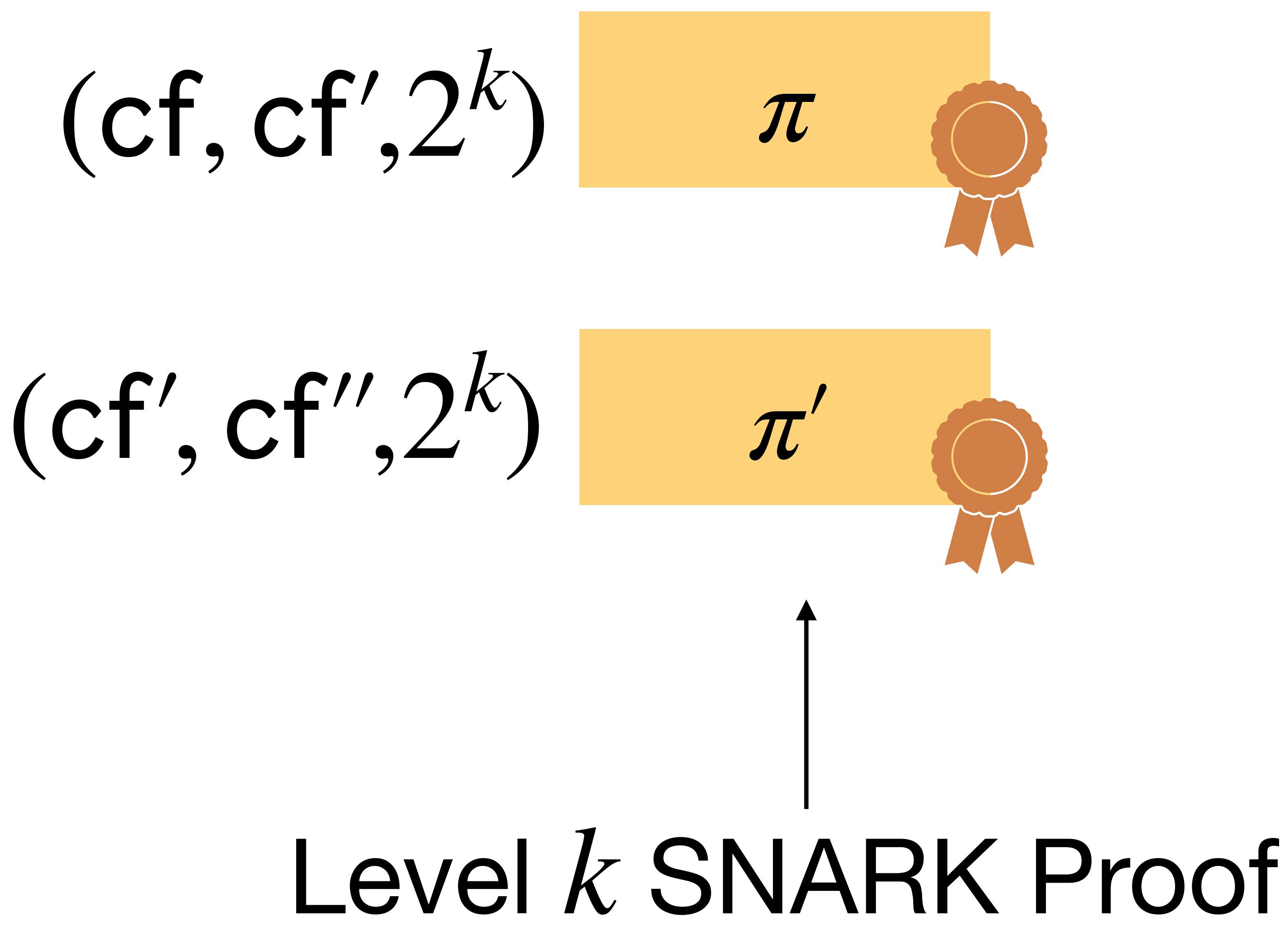
Valiant's Recipe: SNARK the SNARK!

Valiant's Recipe: SNARK the SNARK!

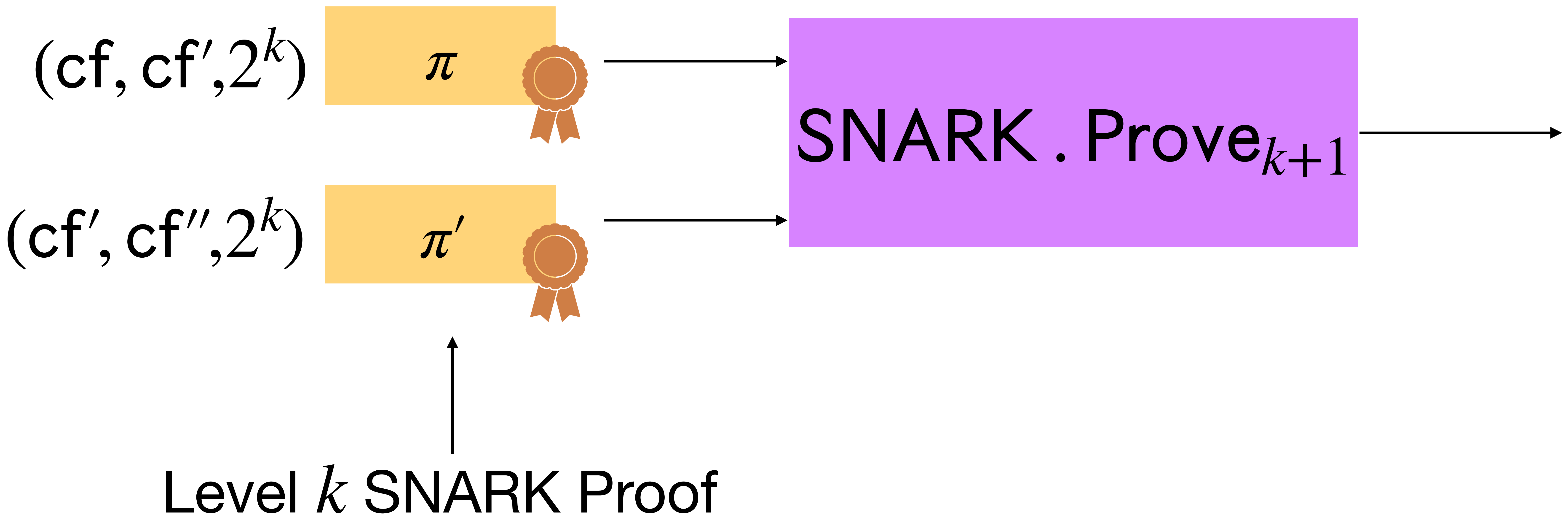
$(\mathsf{cf}, \mathsf{cf}', 2^k)$ π

$(\mathsf{cf}', \mathsf{cf}'', 2^k)$ π'

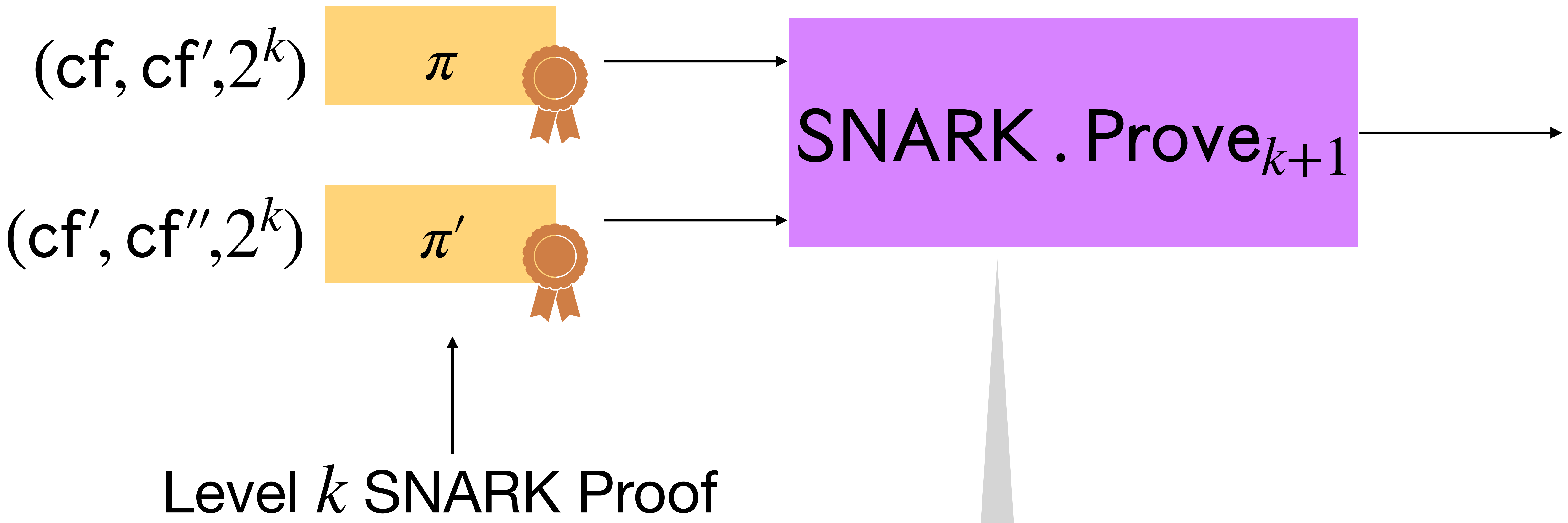
Valiant's Recipe: SNARK the SNARK!



Valiant's Recipe: SNARK the SNARK!

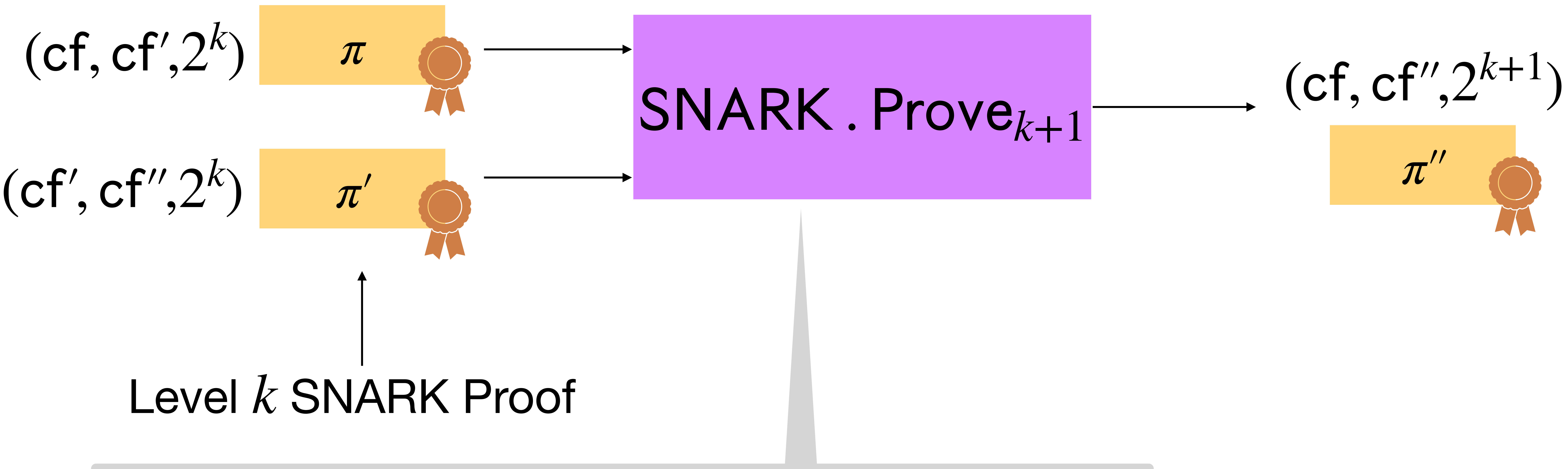


Valiant's Recipe: SNARK the SNARK!



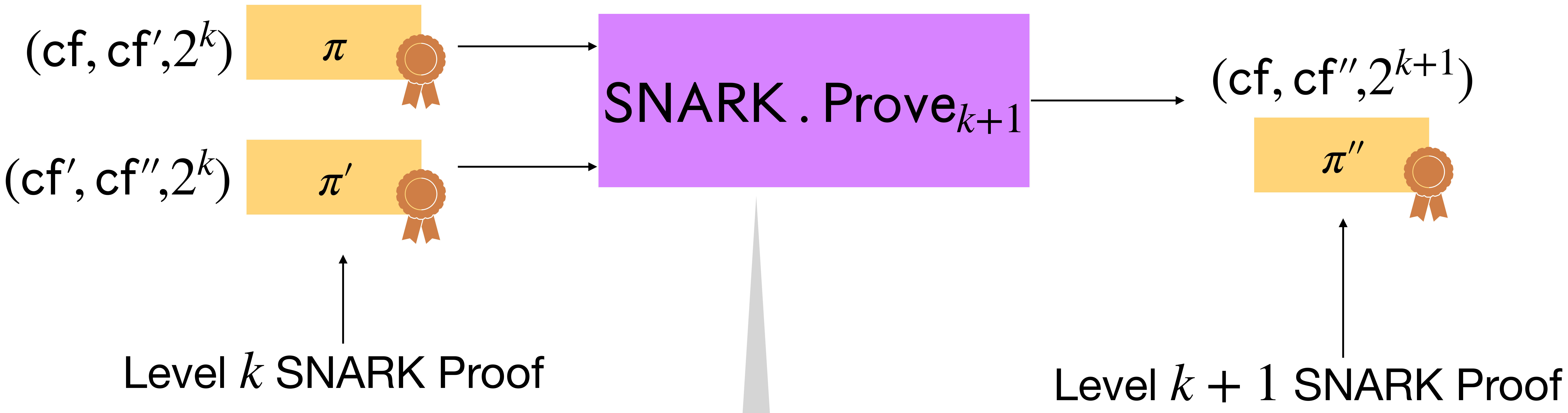
Claim: There exists cf', π, π' such that $\mathsf{Ver}_k(\mathsf{cf}, \mathsf{cf}', \pi) = 1$ and $\mathsf{Ver}_k(\mathsf{cf}', \mathsf{cf}'', \pi') = 1$

Valiant's Recipe: SNARK the SNARK!



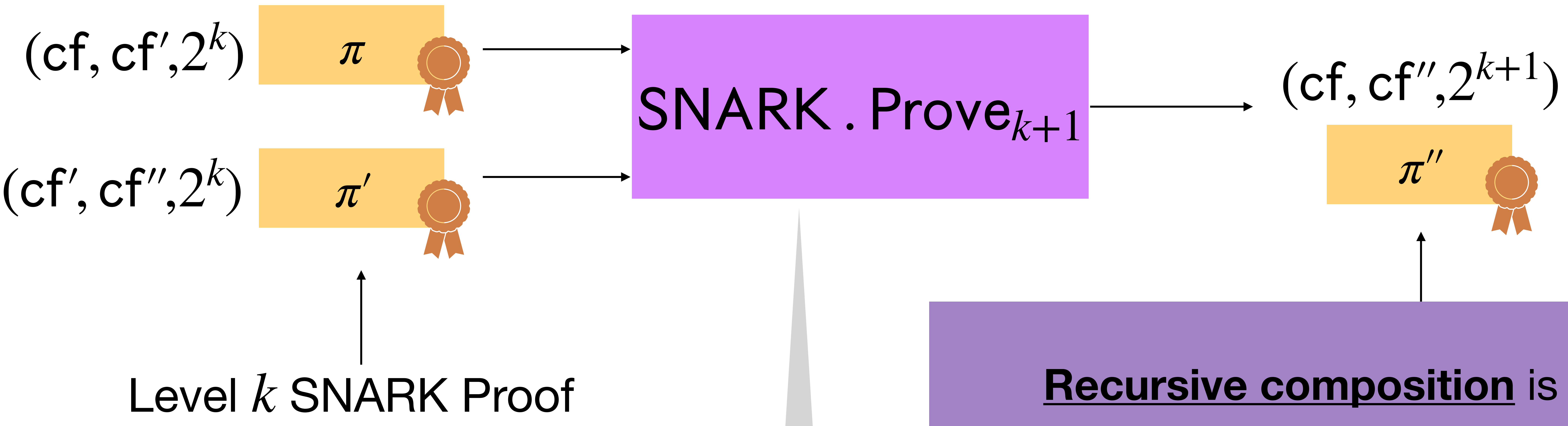
Claim: There exists cf', π, π' such that
$\text{Ver}_k(\text{cf}, \text{cf}', \pi) = 1$ and $\text{Ver}_k(\text{cf}', \text{cf}'', \pi') = 1$

Valiant's Recipe: SNARK the SNARK!



Claim: There exists cf', π, π' such that $\mathsf{Ver}_k(\mathsf{cf}, \mathsf{cf}', \pi) = 1$ and $\mathsf{Ver}_k(\mathsf{cf}', \mathsf{cf}'', \pi') = 1$

Valiant's Recipe: SNARK the SNARK!



Claim: There exists cf', π, π' such

$\mathsf{Ver}_k(\mathsf{cf}, \mathsf{cf}', \pi) = 1$ and $\mathsf{Ver}_k(\mathsf{cf}', \mathsf{cf}'',$

Recursive composition is the backbone of many works! [CT10, BCCT13,BGH19, BCMS20, BDFG21, BCLMS21, KS22, CCS22, etc]

Proof strategy: EXTRACT!

Level 3

$\pi_{cf_0 \to cf_8}$

This is an intuitive sketch missing many details

Proof strategy: EXTRACT!

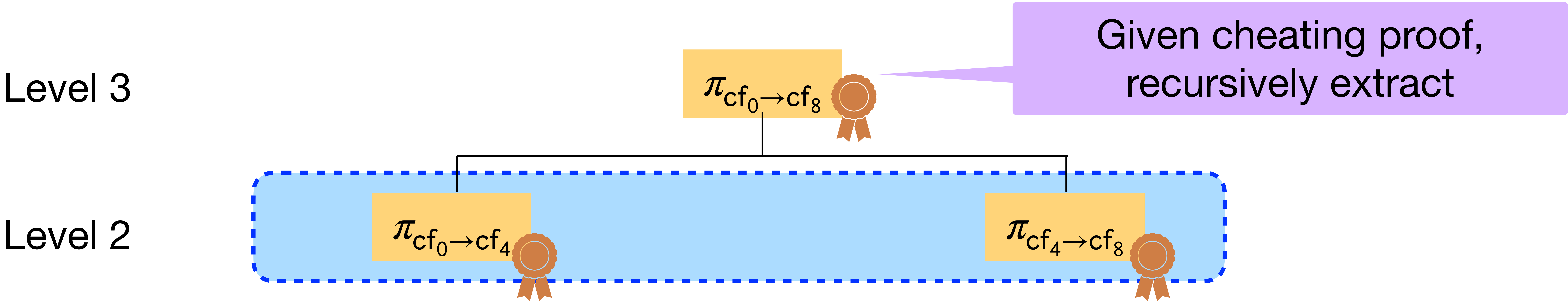
Level 3

$\pi_{\mathsf{cf}_0 \rightarrow \mathsf{cf}_8}$

Given cheating proof, recursively extract

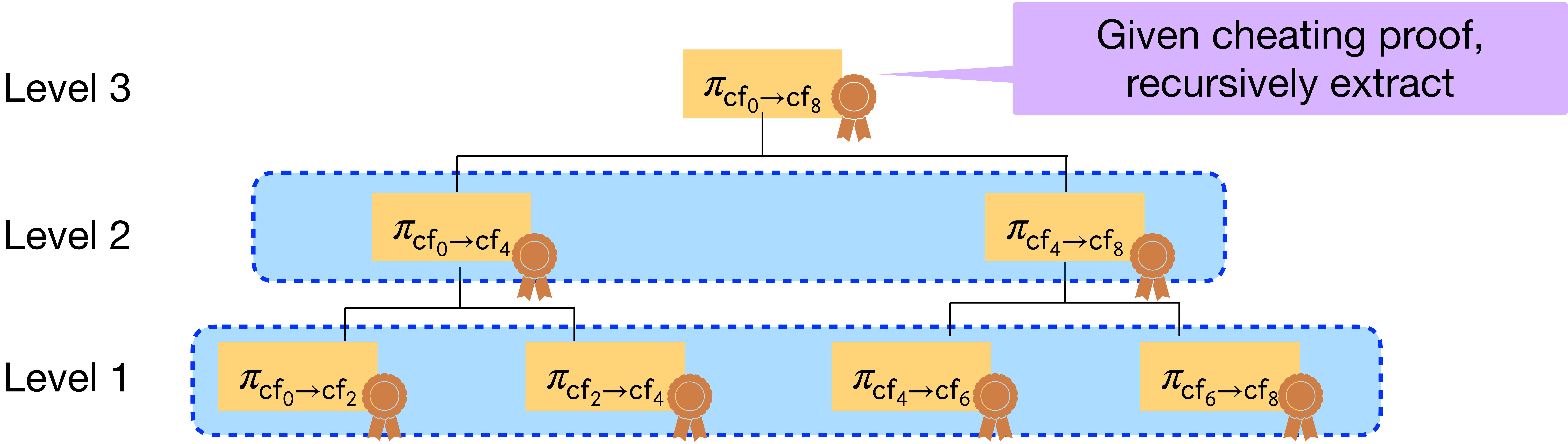
This is an intuitive sketch missing many details

Proof strategy: EXTRACT!



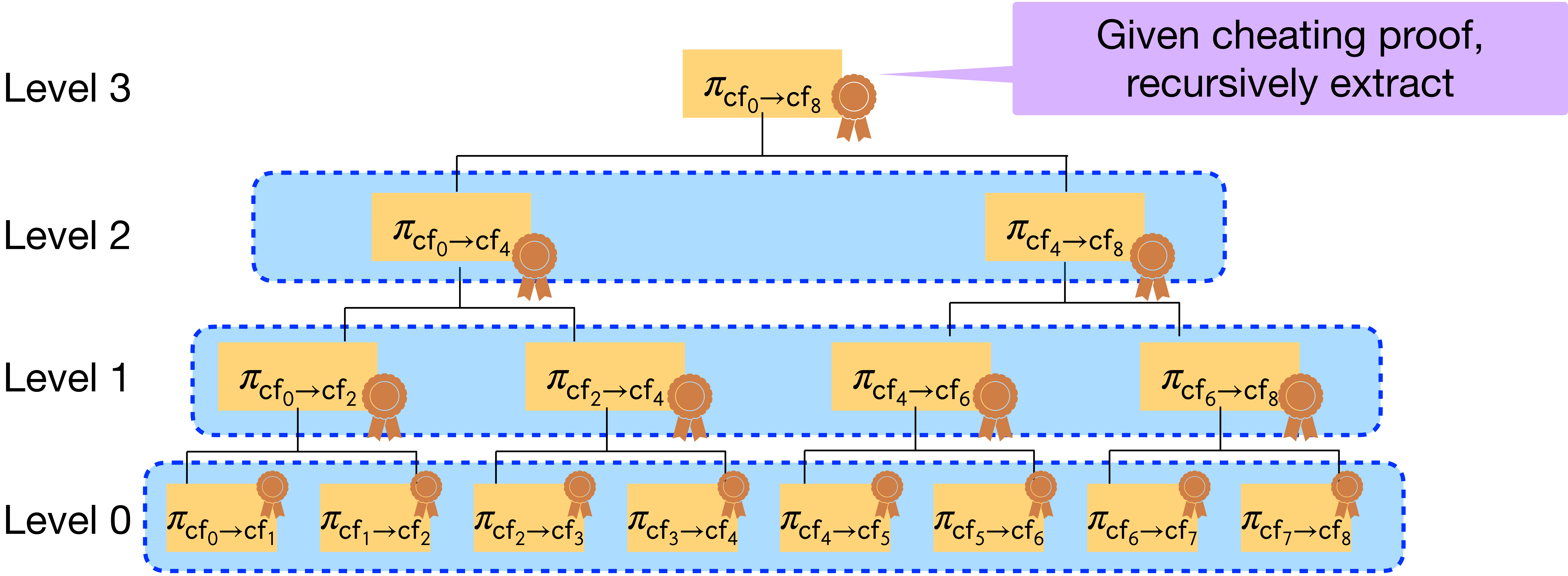
This is an intuitive sketch missing many details

Proof strategy: EXTRACT!



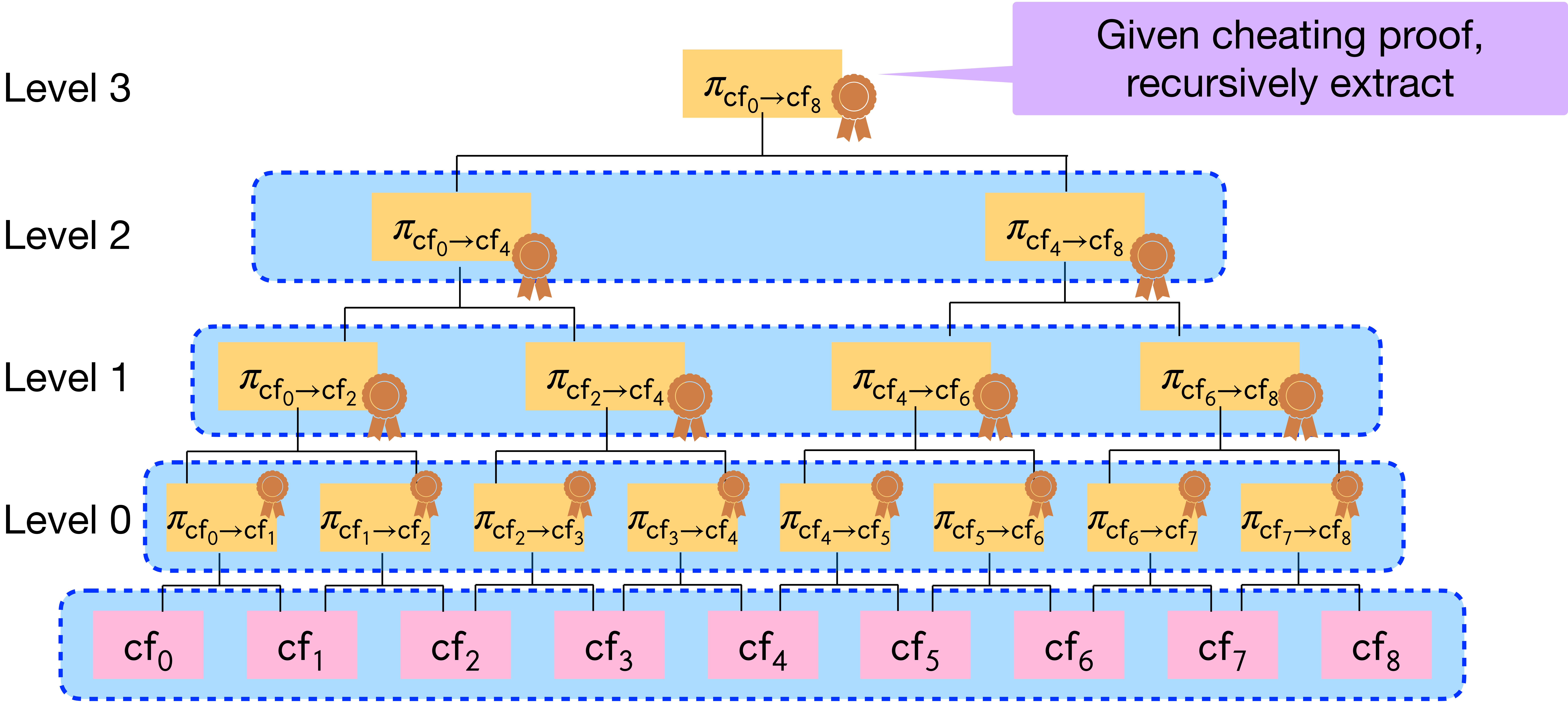
This is an intuitive sketch missing many details

Proof strategy: EXTRACT!



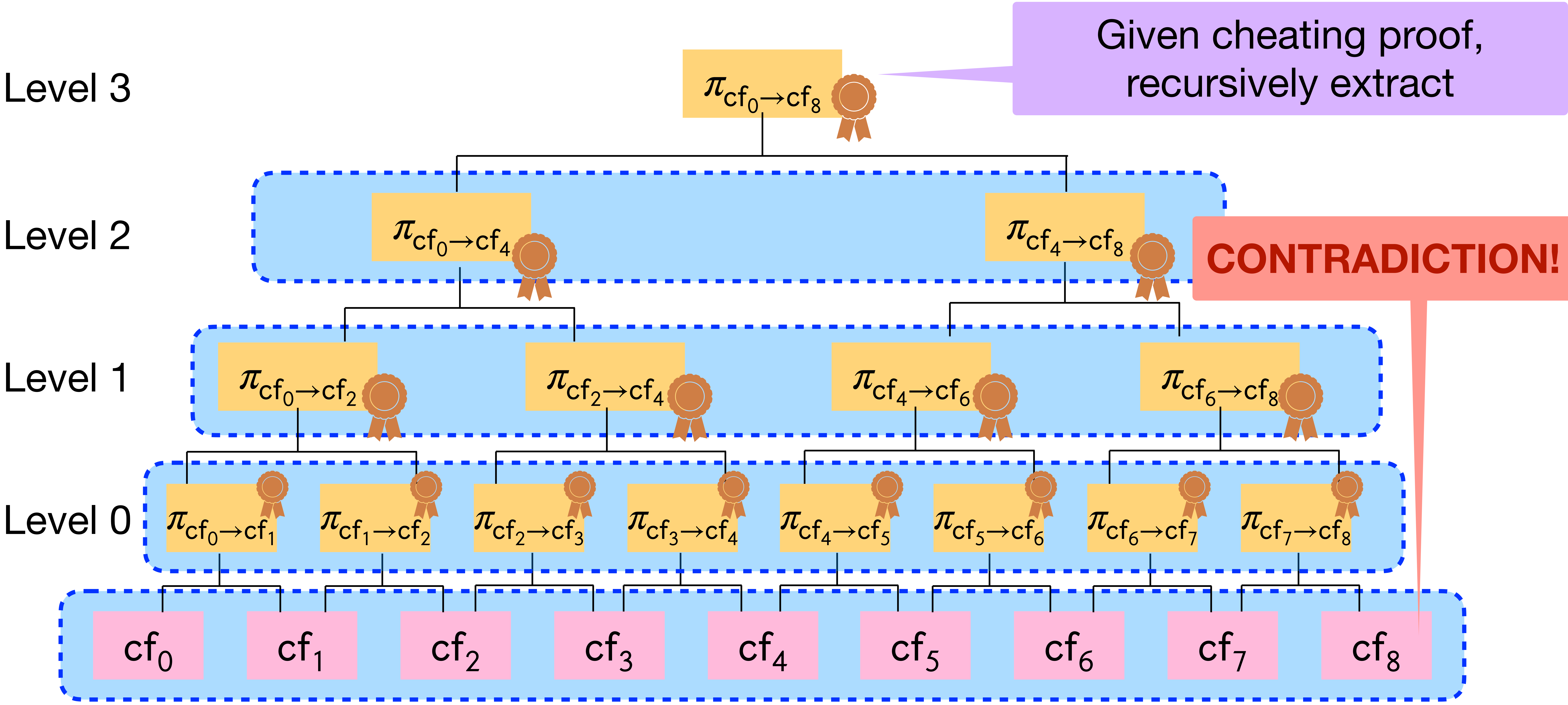
This is an intuitive sketch missing many details

Proof strategy: EXTRACT!



This is an intuitive sketch missing many details

Proof strategy: EXTRACT!



This is an intuitive sketch missing many details

IVC from Standard Assumptions?

*IVC for P exists from standard assumptions, and I will get to that soon :)

IVC from Standard Assumptions?

- **Issue 1:** SNARKs **do not exist** from standard assumptions!! [CGKS23]

*IVC for P exists from standard assumptions, and I will get to that soon :)

IVC from Standard Assumptions?

- **Issue 1:** SNARKs **do not exist** from standard assumptions!! [CGKS23]
    - ***Extraction*** was extremely crucial to make the soundness analysis go through!

*IVC for P exists from standard assumptions, and I will get to that soon :)

IVC from Standard Assumptions?

- **Issue 1:** SNARKs **do not exist** from standard assumptions!! [CGKS23]
  - ***Extraction*** was extremely crucial to make the soundness analysis go through!
- **Actually**, what about **random oracle model**?

*IVC for P exists from standard assumptions, and I will get to that soon :)

IVC in Random Oracle Model?

IVC in Random Oracle Model?

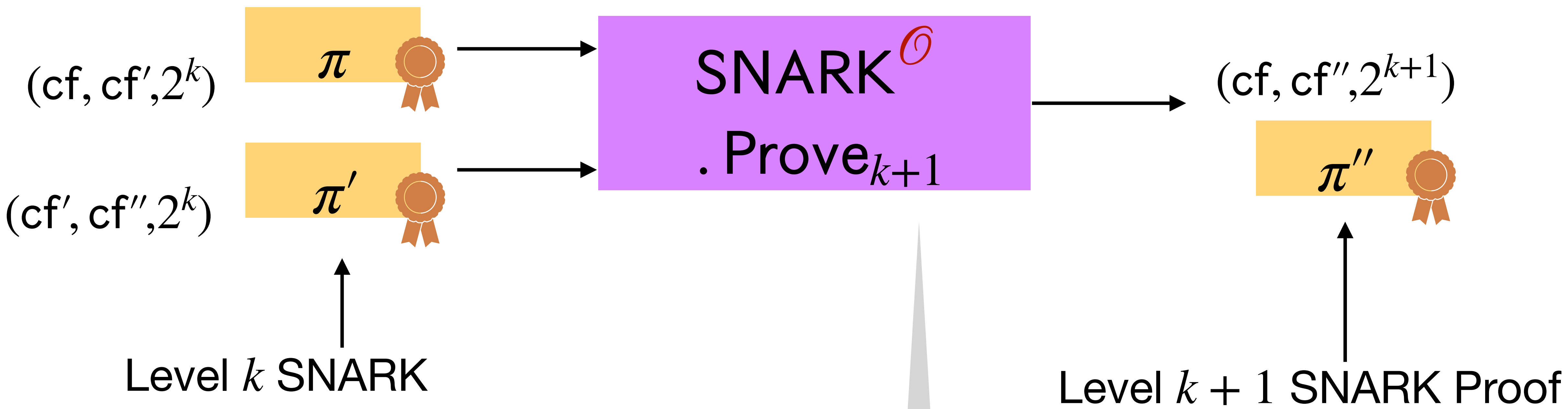
- **SNARKs exist in the ROM!**

IVC in Random Oracle Model?

- **SNARKs exist in the ROM!**
  - Valiant's construction: Random oracle is **an oracle** and a **concrete function,** and then an **oracle again.**

IVC in Random Oracle Model?

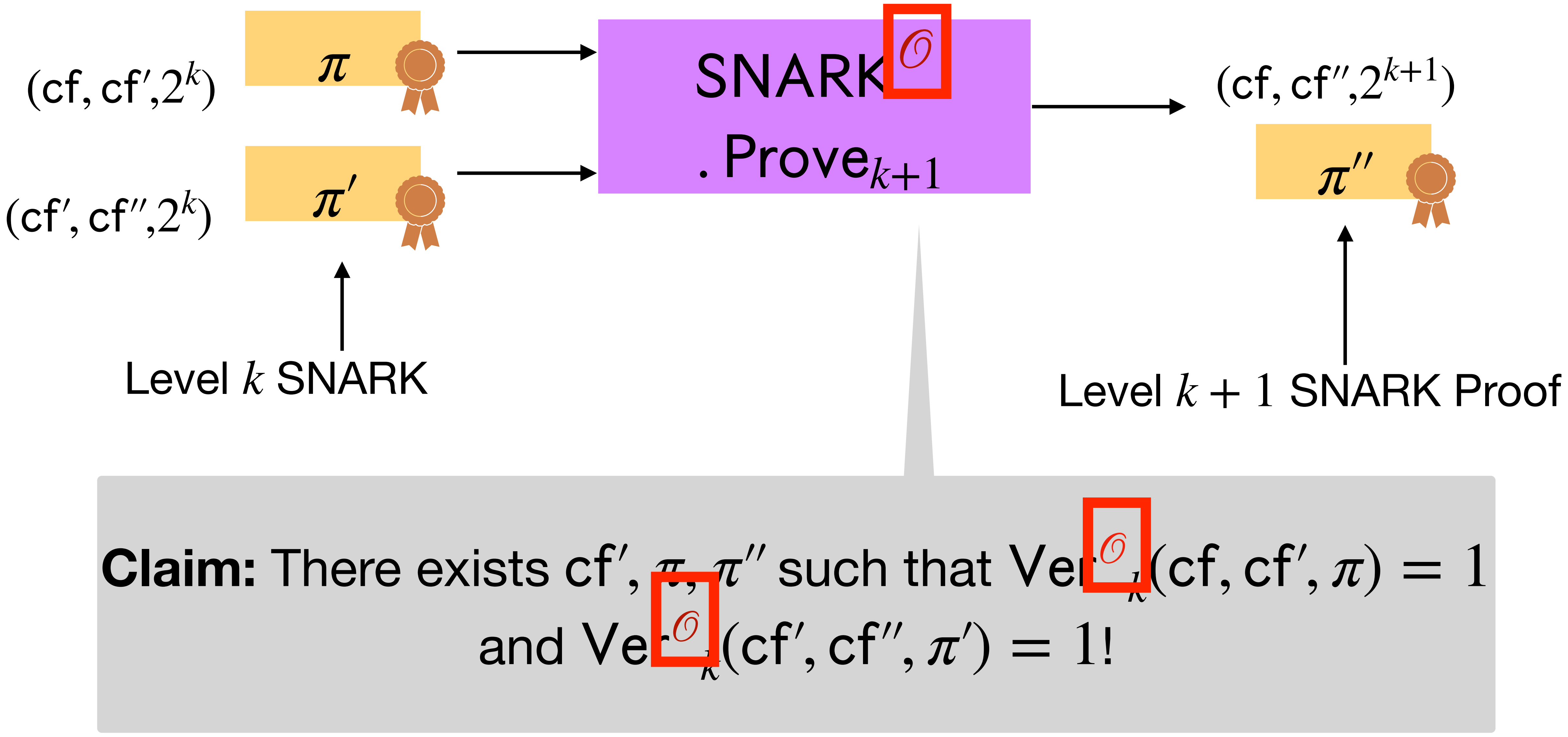
- **SNARKs exist in the ROM!**
    - Valiant's construction: Random oracle is **an oracle** and a **concrete function,** and then an **oracle again.**



Claim: There exists cf', π, π'' such that $\mathsf{Ver}^{\mathcal{O}}{}_k(\mathsf{cf}, \mathsf{cf}', \pi) = 1$ and $\mathsf{Ver}^{\mathcal{O}}{}_k(\mathsf{cf}', \mathsf{cf}'', \pi') = 1$!

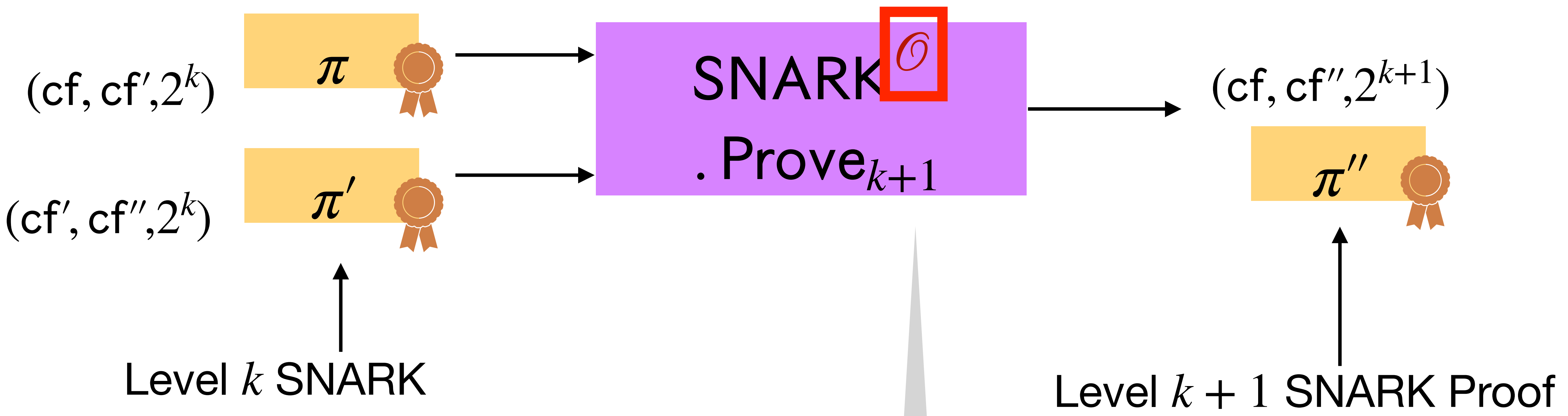
IVC in Random Oracle Model?

- **SNARKs exist in the ROM!**
  - Valiant's construction: Random oracle is **an oracle** and a **concrete function,** and then an **oracle again.**



IVC in Random Oracle Model?

- **SNARKs exist in the ROM!**
  - Valiant's construction: Random oracle is **an oracle** and a **concrete function,** and then an **oracle again.**



Claim: There exists cf′, π, π'' such that $\mathsf{Ver}^{\mathcal{O}}_k(\mathsf{cf}, \mathsf{cf}', \pi) = 1$ and $\mathsf{Ver}^{\mathcal{O}}_k(\mathsf{cf}', \mathsf{cf}'', \pi') = 1$!

IVC in Random Oracle Model?

IVC in Random Oracle Model?

- **SNARKs exist in the ROM!**
  - Valiant's construction: Random oracle is **an oracle** and a **concrete function,** and then an **oracle again.**

IVC in Random Oracle Model?

- **SNARKs exist in the ROM!**
  - Valiant's construction: Random oracle is **an oracle** and a **concrete function,** and then an **oracle again.**
    - Security gap!

IVC in Random Oracle Model?

- **SNARKs exist in the ROM!**
  - Valiant's construction: Random oracle is **an oracle** and a **concrete function,** and then an **oracle again.**
    - Security gap!
  - SNARKs in the ROM cannot be **recursively composed** [BCG24]!

IVC in Random Oracle Model?

- **SNARKs exist in the ROM!**
  - Valiant's construction: Random oracle is **an oracle** and a **concrete function,** and then an **oracle again.**
    - Security gap!
  - SNARKs in the ROM cannot be **recursively composed** [BCG24]!
  - [HAN23] showed some barriers in the ROM

IVC in Random Oracle Model?

- **SNARKs exist in the ROM!**
  - Valiant's construction: Random oracle is **an oracle** and a **concrete function,** and then an **oracle again.**
    - Security gap!
  - SNARKs in the ROM cannot be **recursively composed** [BCG24]!
  - [HAN23] showed some barriers in the ROM
  - Shows that SNARGs and IVC are fundamentally **different** problems

This Work:
How do we construct **IVC for NP**
from **standard assumptions**?

This work

This work

- Two constructions of IVC from **standard assumptions**!

This work

- Two constructions of IVC from **standard assumptions**!
  - **Extraction-based:** IVC for NP via rate-1 BARGs + SNARGs for NP.

This work

- Two constructions of IVC from **standard assumptions**!
  - **Extraction-based:** IVC for NP via rate-1 BARGs + SNARGs for NP.
    - **Proof size:** $\text{poly}(\lambda, |\text{cf}|, \log T)$.

This work

- Two constructions of IVC from **standard assumptions**!
  - **Extraction-based:** IVC for NP via rate-1 BARGs + SNARGs for NP.
    - **Proof size:** poly$(\lambda, |\mathsf{cf}|, \log T)$.
    - Relies heavily on tools from prior works [PP22, DGKV22] on IVC for P

This work

- Two constructions of IVC from **standard assumptions**!
  - **Extraction-based:** IVC for NP via rate-1 BARGs + SNARGs for NP.
    - **Proof size:** poly$(\lambda, |\text{cf}|, \log T)$.
    - Relies heavily on tools from prior works [PP22, DGKV22] on IVC for P
  - **Without extraction:** Even more succinct IVC for **"Trapdoor-NP"** via a "Pure IO" approach.

This work

- Two constructions of IVC from **standard assumptions**!
  - **Extraction-based:** IVC for NP via rate-1 BARGs + SNARGs for NP.
    - **Proof size:** poly$(\lambda, |\mathsf{cf}|, \log T)$.
    - Relies heavily on tools from prior works [PP22, DGKV22] on IVC for P
  - **Without extraction:** Even more succinct IVC for **“Trapdoor-NP”** via a “Pure IO” approach.
    - **Proof size:** poly$(\lambda, \log T)$!

This work

- Two constructions of IVC from **standard assumptions**!
  - **Extraction-based:** IVC for NP via rate-1 BARGs + SNARGs for NP.
    - **Proof size:** poly$(\lambda, |\mathsf{cf}|, \log T)$.
    - Relies heavily on tools from prior works [PP22, DGKV22] on IVC for P
  - **Without extraction:** Even more succinct IVC for **“Trapdoor-NP”** via a “Pure IO” approach.
    - **Proof size:** poly$(\lambda, \log T)$!
- We show how to achieve ZK in both settings.

IVC for NP!

IVC for NP!

Theorem [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: $\text{poly}(\lambda, |\text{cf}|, \log T)$.

IVC for NP!

Theorem [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: $\text{poly}(\lambda, |\text{cf}|, \log T)$.

Independent of time-steps, but grows with intermediate configurations cf_i

IVC for NP!

Theorem [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: poly$(\lambda, |\mathsf{cf}|, \log T)$.

IVC for NP!

Theorem [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: $\mathsf{poly}(\lambda, |\mathsf{cf}|, \log T)$.

- Careful complexity leveraging of [Paneth-Pass ’22], [Devadas-Goyal-Kalai-Vaikuntanathan ’22].

IVC for NP!

Theorem [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: poly$(\lambda, |\mathsf{cf}|, \log T)$.

- Careful complexity leveraging of [Paneth-Pass '22], [Devadas-Goyal-Kalai-Vaikuntanathan '22].

Theorem [PP22/DGKV22]. Assuming rate-1 somewhere extractable BARGs, there exists IVC for *deterministic* computations.

IVC for NP!

Theorem [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: $\mathsf{poly}(\lambda, |\mathsf{cf}|, \log T)$.

- Careful complexity leveraging of [Paneth-Pass '22], [Devadas-Goyal-Kalai-Vaikuntanathan '22].

IVC for NP!

Theorem [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: $\mathrm{poly}(\lambda, |\mathsf{cf}|, \log T)$.

- Careful complexity leveraging of [Paneth-Pass '22], [Devadas-Goyal-Kalai-Vaikuntanathan '22].
- Addresses the common misconception that IVC for NP is **impossible** due to Gentry-Wichs.

Valiant's Recipe: Proof Merging

Valiant's Recipe: Proof Merging

- **Proof of knowledge:** If adversary gives **accepting** $(\mathsf{cf}, \mathsf{cf}'', t + t'), \pi''$, one can **extract** accepting tuples $(\mathsf{cf}, \mathsf{cf}', t), \pi$ and $(\mathsf{cf}', \mathsf{cf}'', t'), \pi'$.

- **Succinctness:** $|\pi''| \approx |\pi|, |\pi'|$

Valiant's Recipe: Proof Merging

Somewhere **extraction:** You can extract either π or π', but not **both.**

- **Succinctness:** $|\pi''| \approx |\pi|, |\pi'|$

Valiant's Recipe: Proof Merging

__Somewhere__ **extraction:** You can extract either π or π', but not **both.**

- **Succinctness:** $|\pi''| \approx |\pi|, |\pi'|$

Valiant's Recipe: Proof Merging

Idea: Can be achieved via
***b**atch **arg**uments (BARGs)*! [DGKV/PP22]
Known from *standard assumptions*

Somewhere **extraction:** You can extract either π or π', but not **both.**

- **Succinctness:** $|\pi''| \approx |\pi|, |\pi'|$

Valiant's Recipe: Proof Merging

Idea: Can be achieved via
***b**atch **arg**uments (BARGs)*! [DGKV/PP22]
Known from *standard assumptions*

***Somewhere* extraction:** You can extract either π or π', but not **both.**

- **Succinctness:** $|\pi''| \approx |\pi|, |\pi'|$

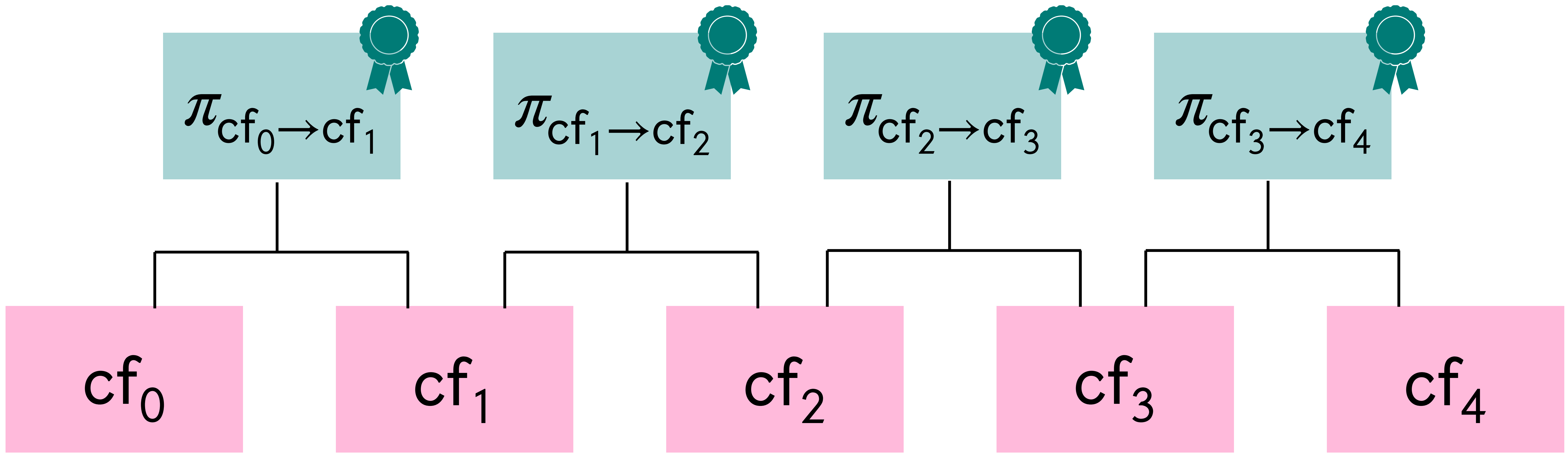
Achieved if BARG is rate-1

Construction

cf_0 cf_1 cf_2 cf_3 cf_4

Construction

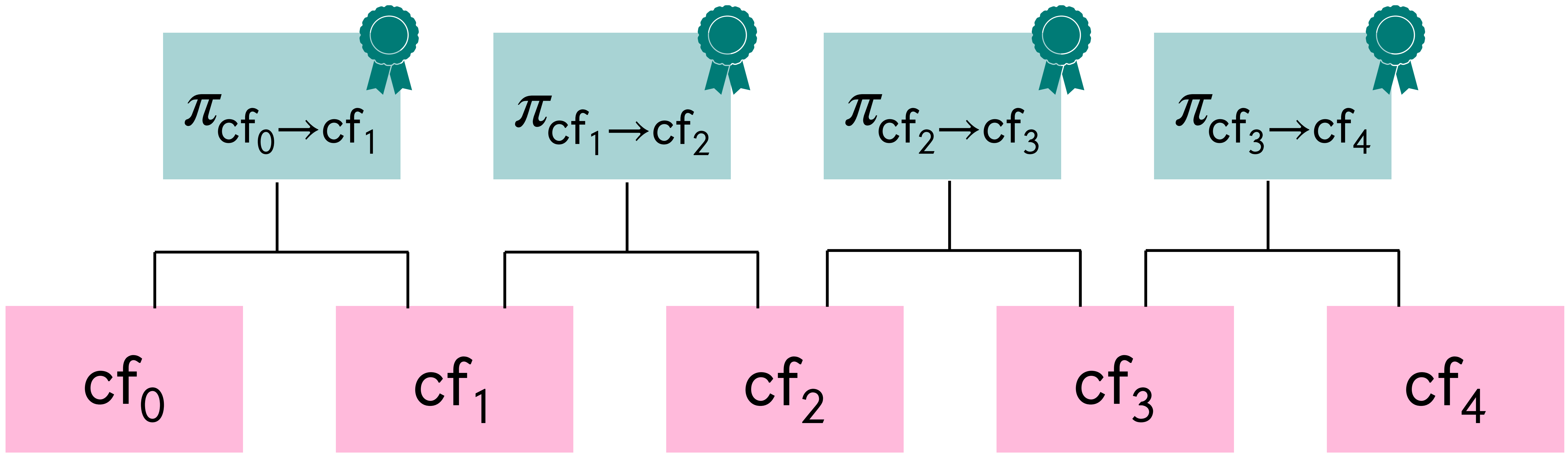
Base case:
SNARG for NP!



Construction

Base case:
SNARG for NP!

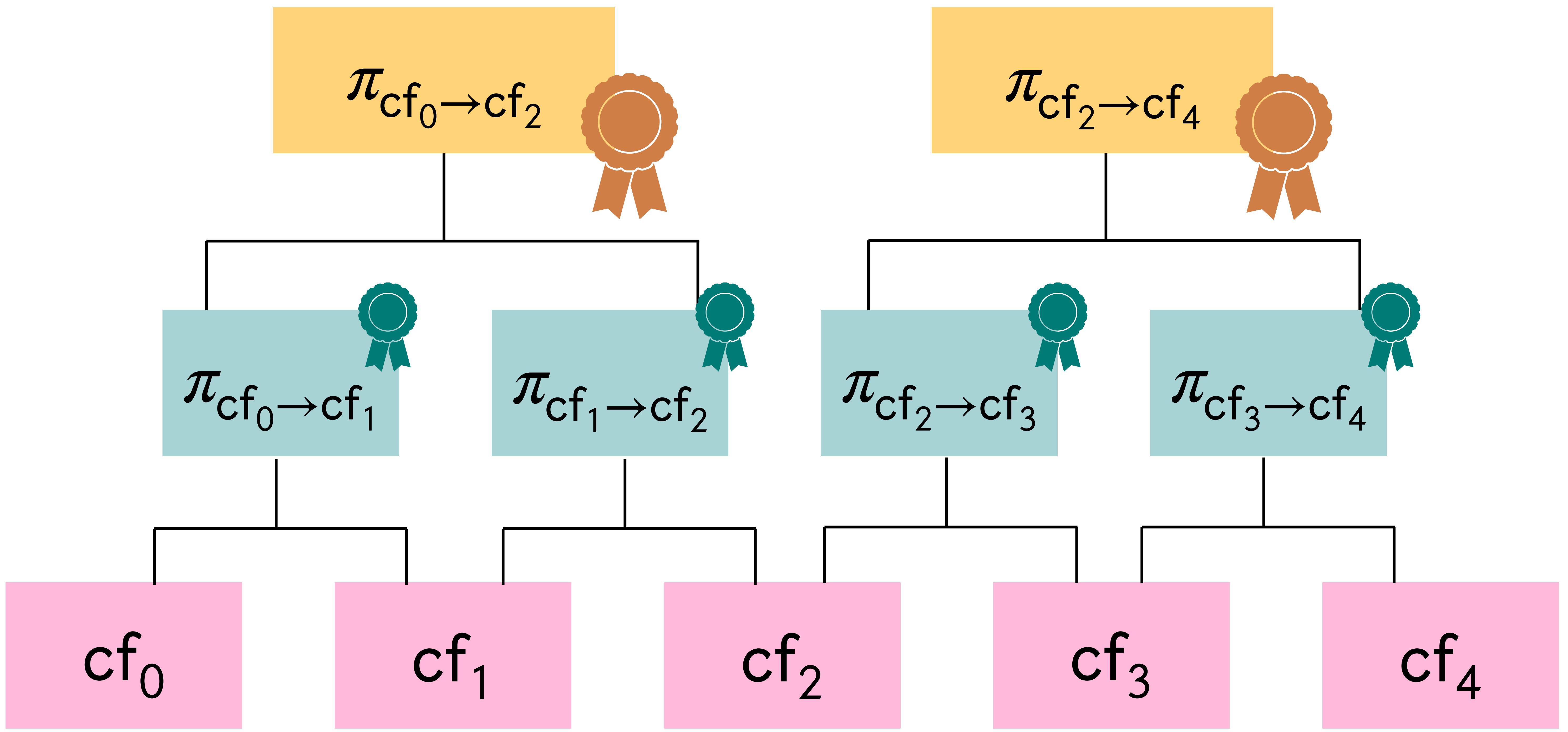
Removes dependence on w



Construction

Base case:
SNARG for NP!

Removes dependence on w

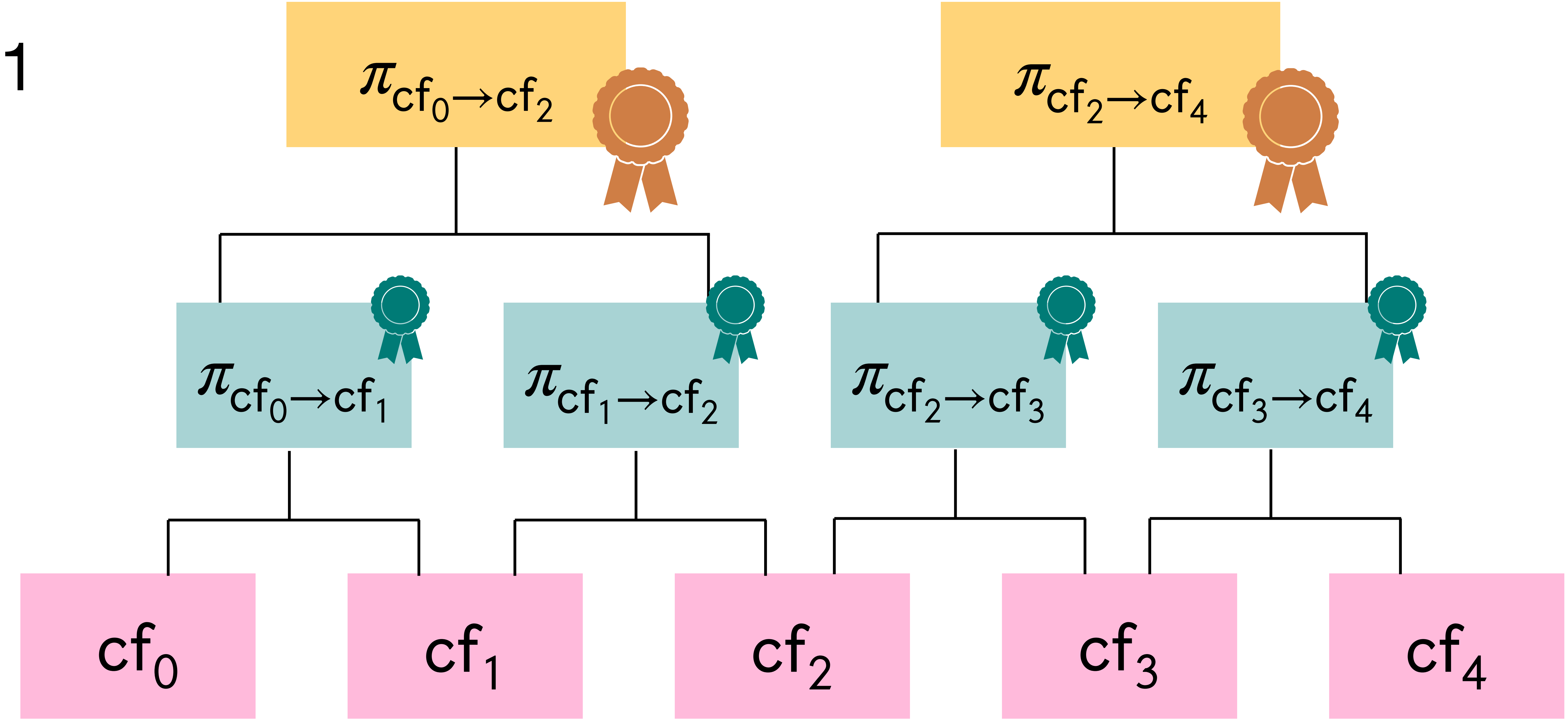


Construction

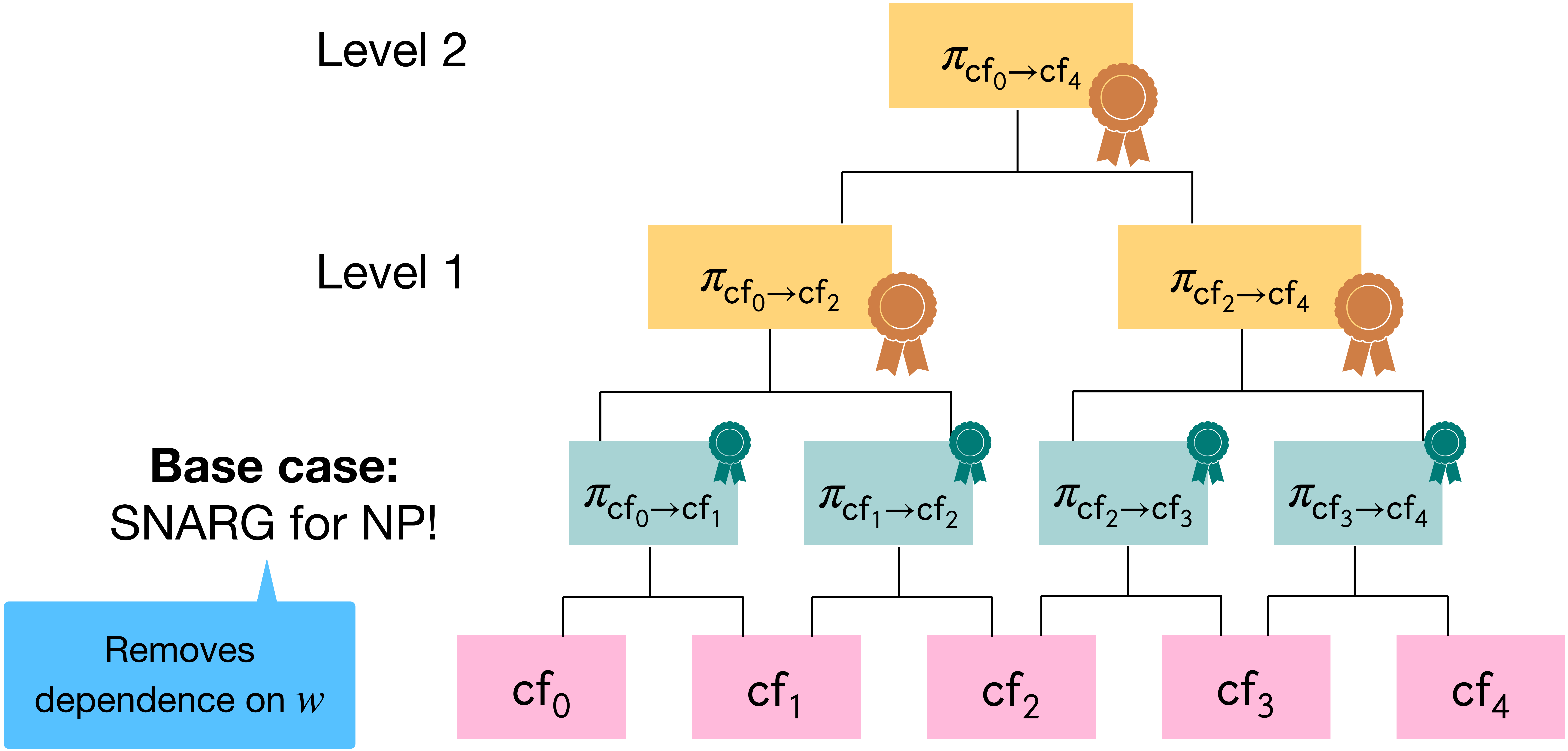
Level 1

Base case:
SNARG for NP!

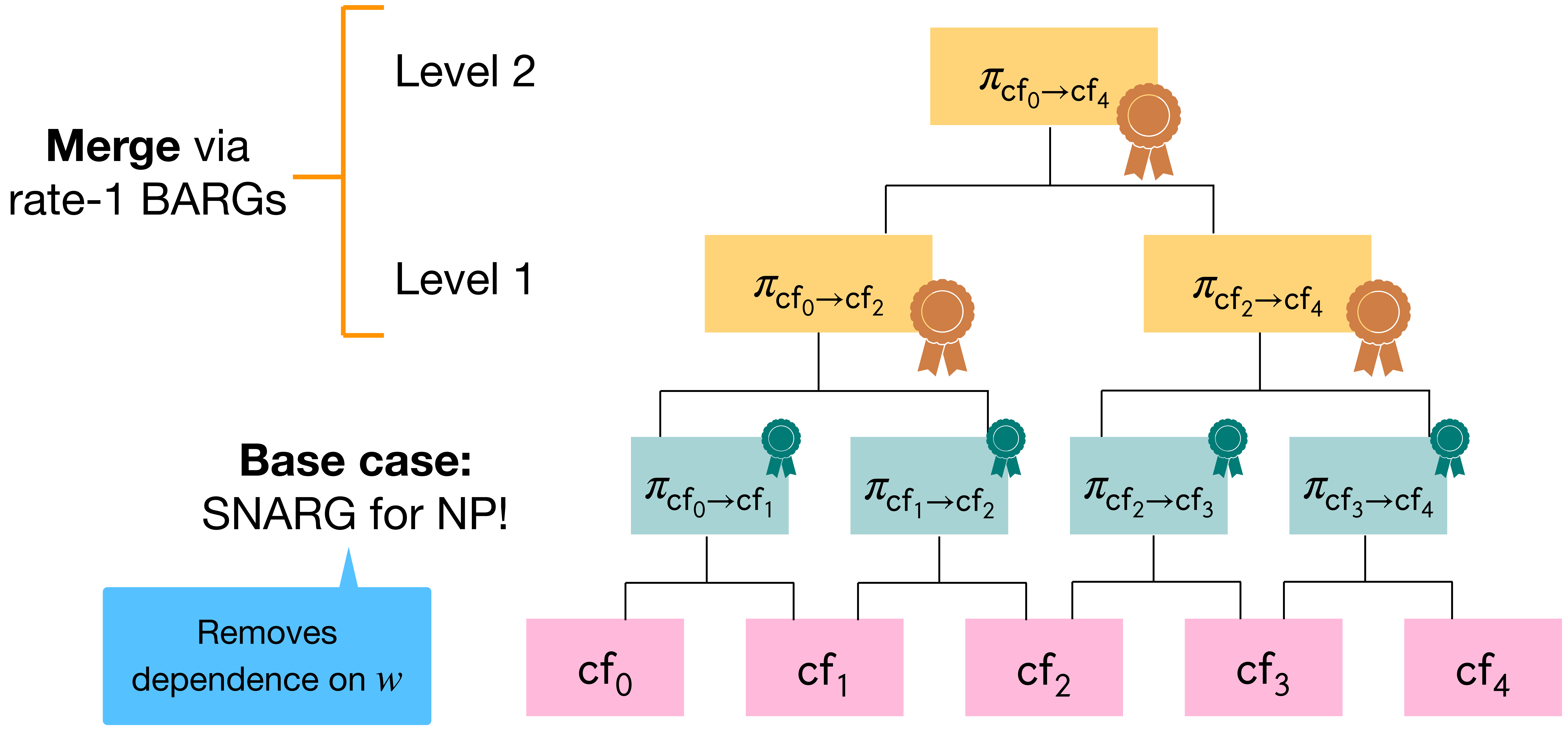
Removes dependence on w



Construction

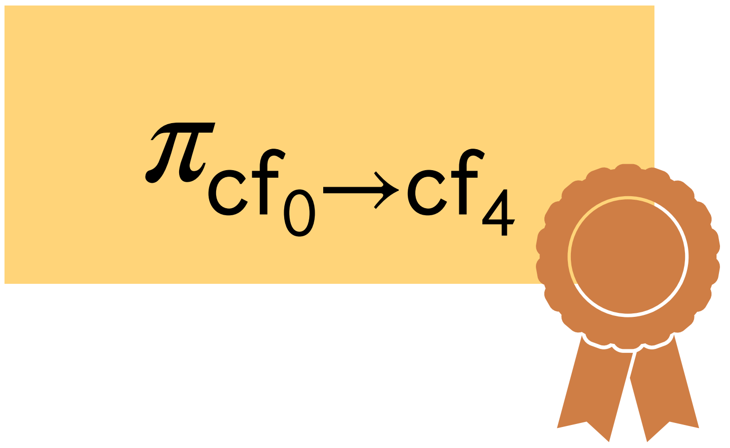


Construction



Security sketch with “Somewhere Soundness”

Level 2

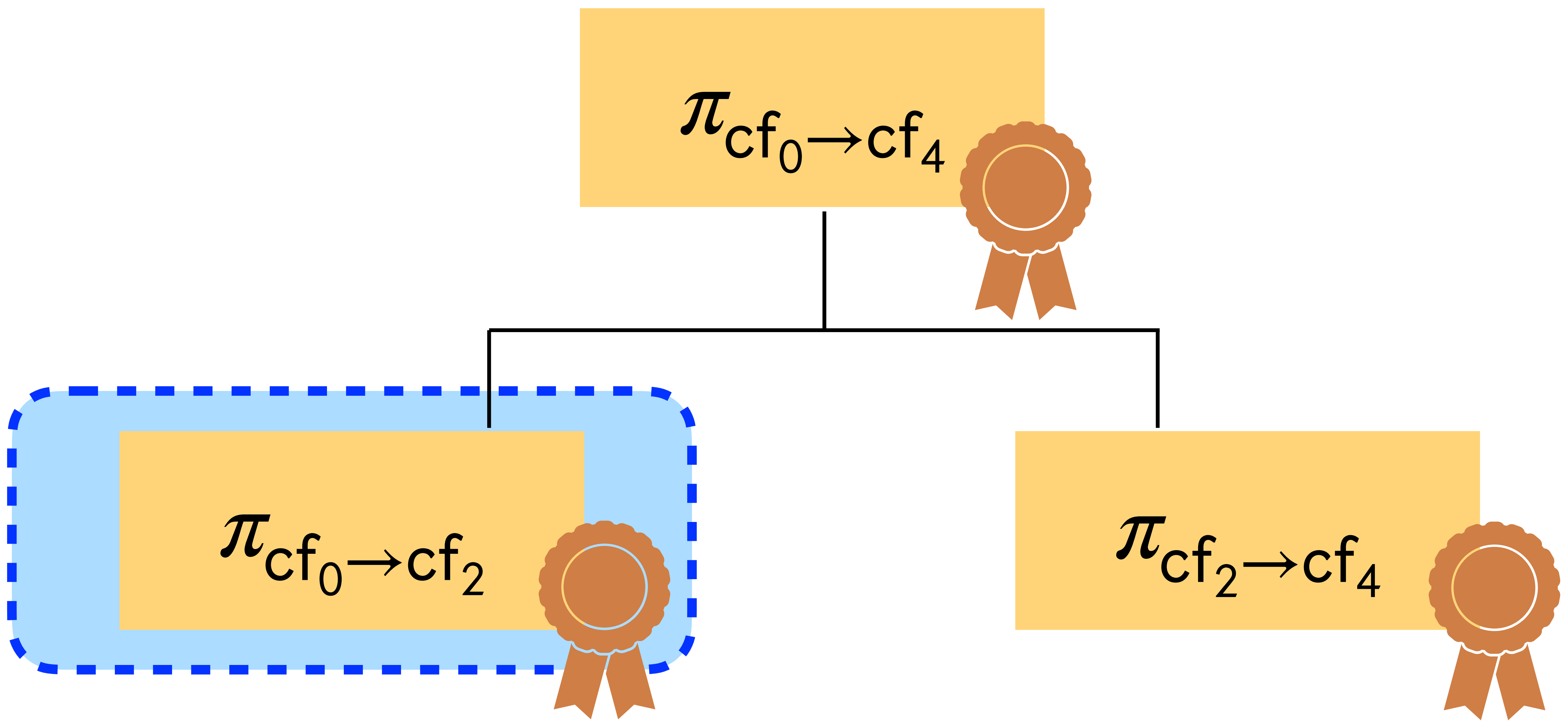


Level 1

Security sketch with “Somewhere Soundness”

Level 2

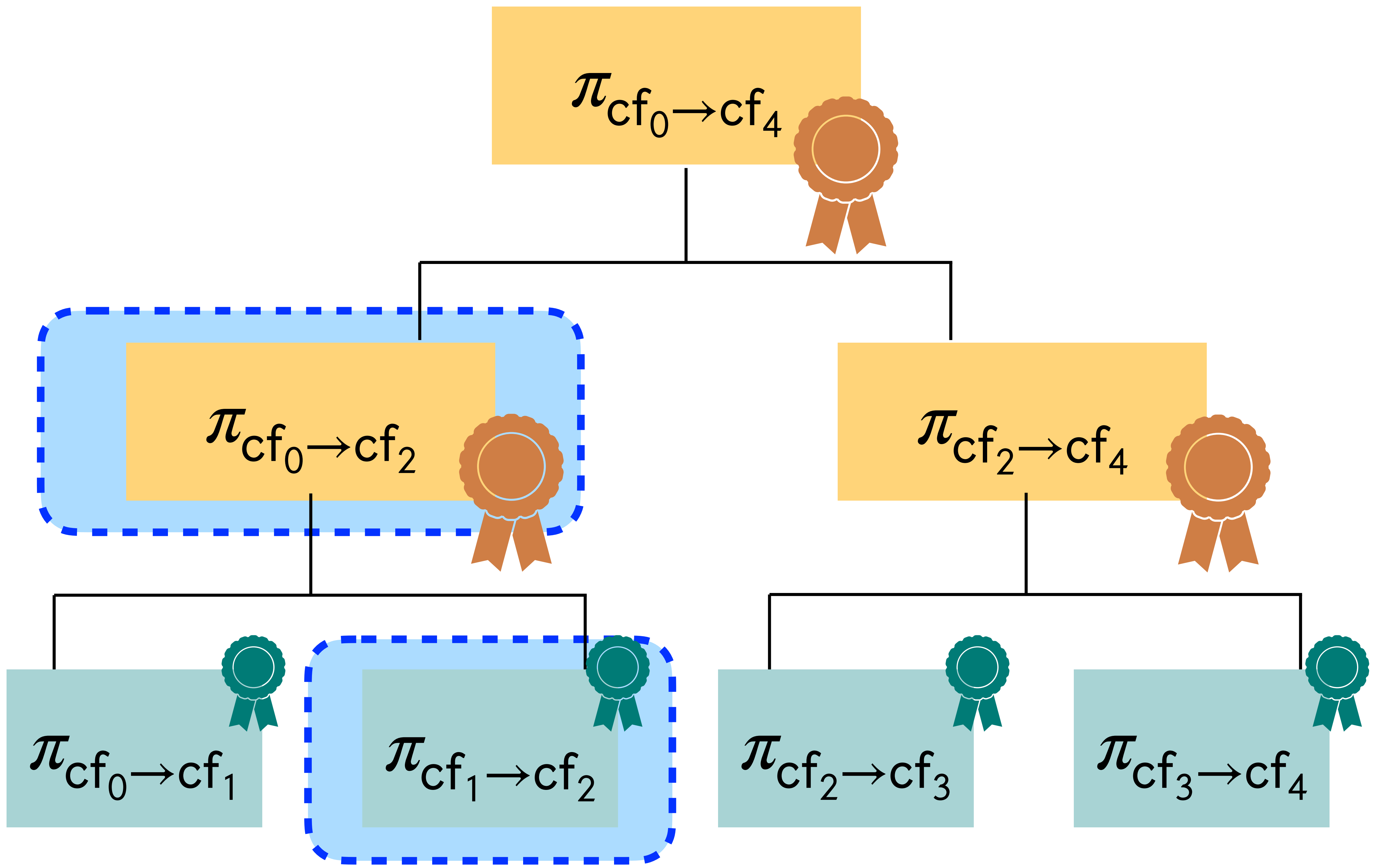
Level 1



Security sketch with "Somewhere Soundness"

Level 2

Level 1

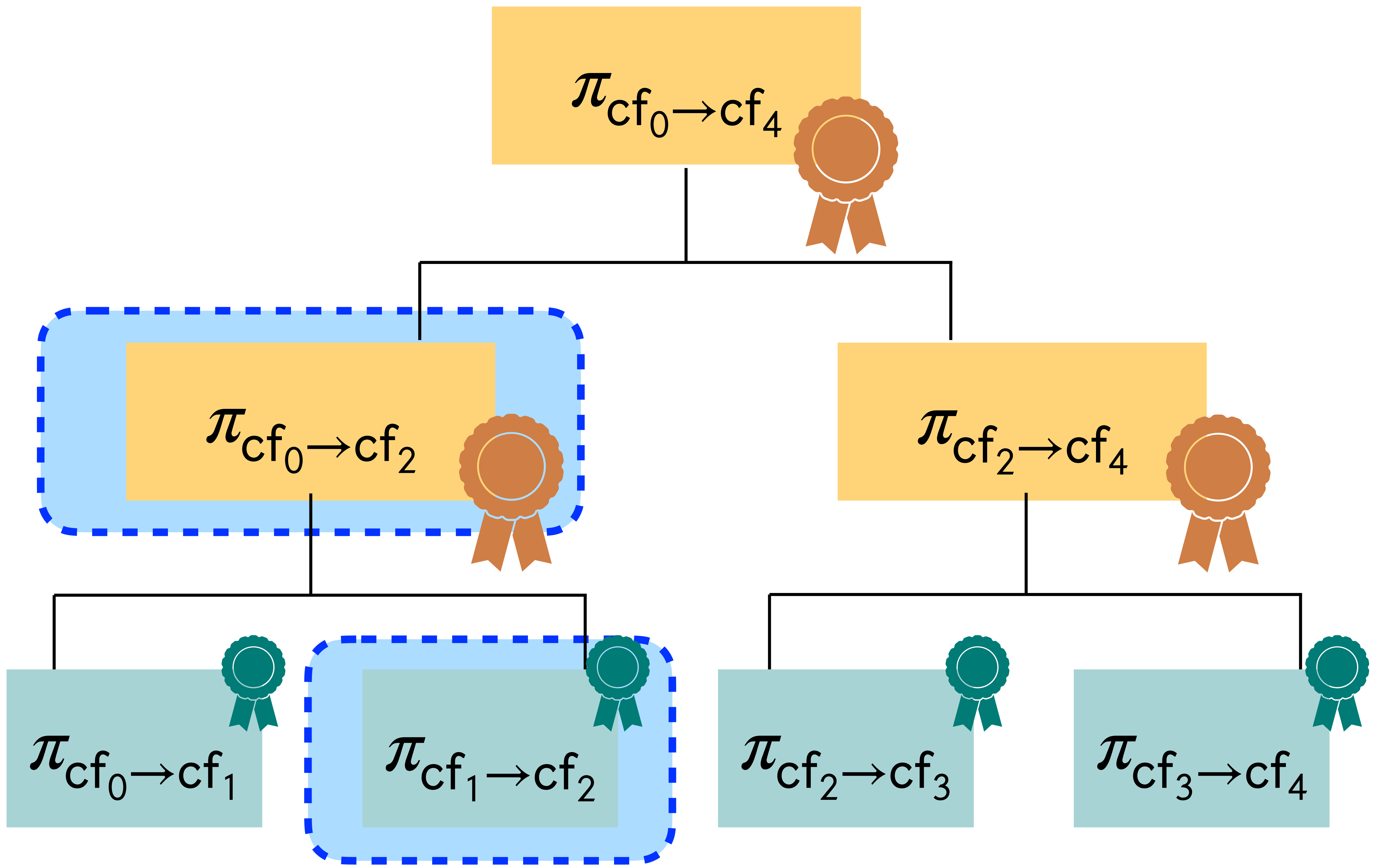


Security sketch with "Somewhere Soundness"

Level 2

Level 1

Base case:
SNARG for NP!

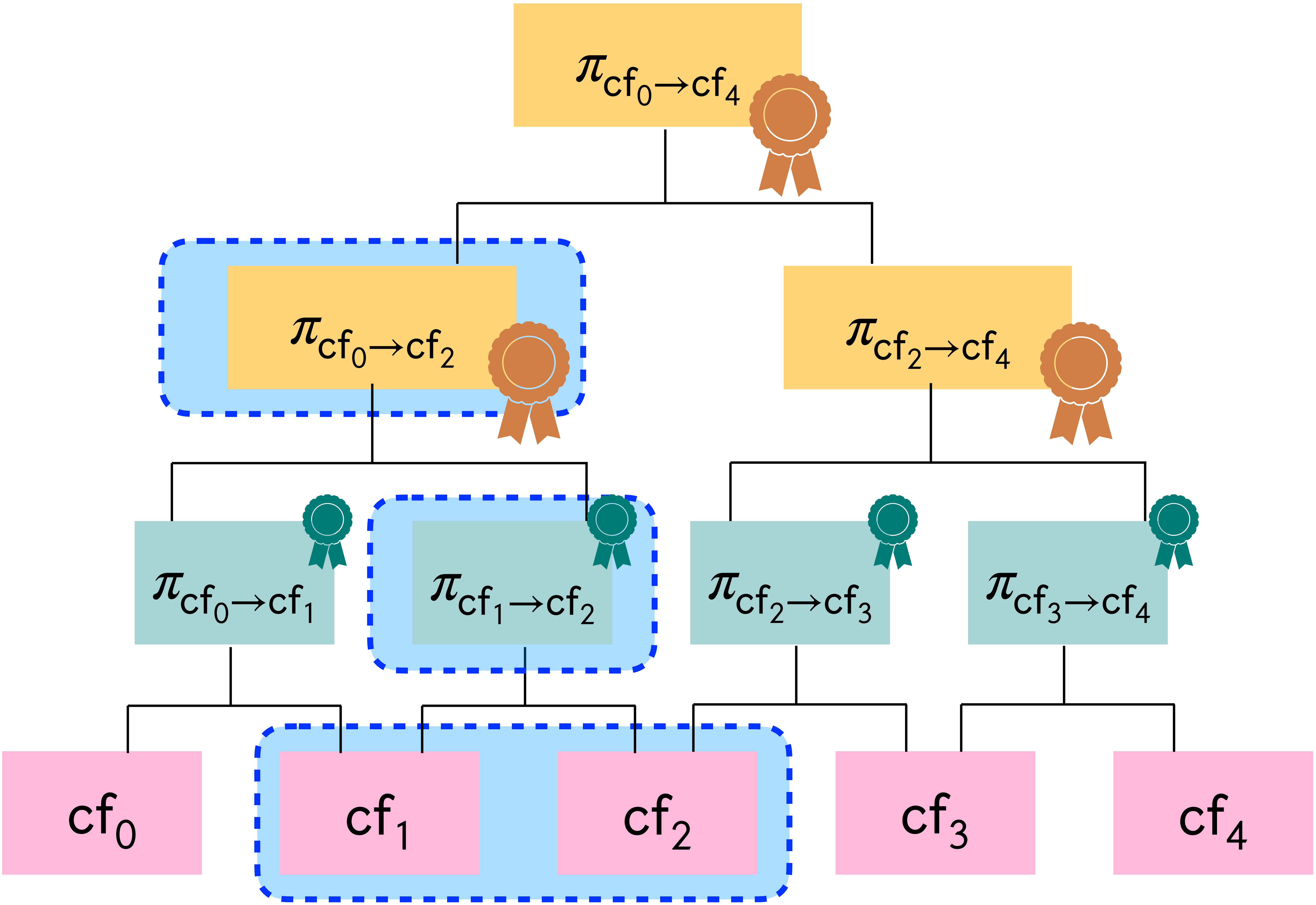


Security sketch with “Somewhere Soundness”

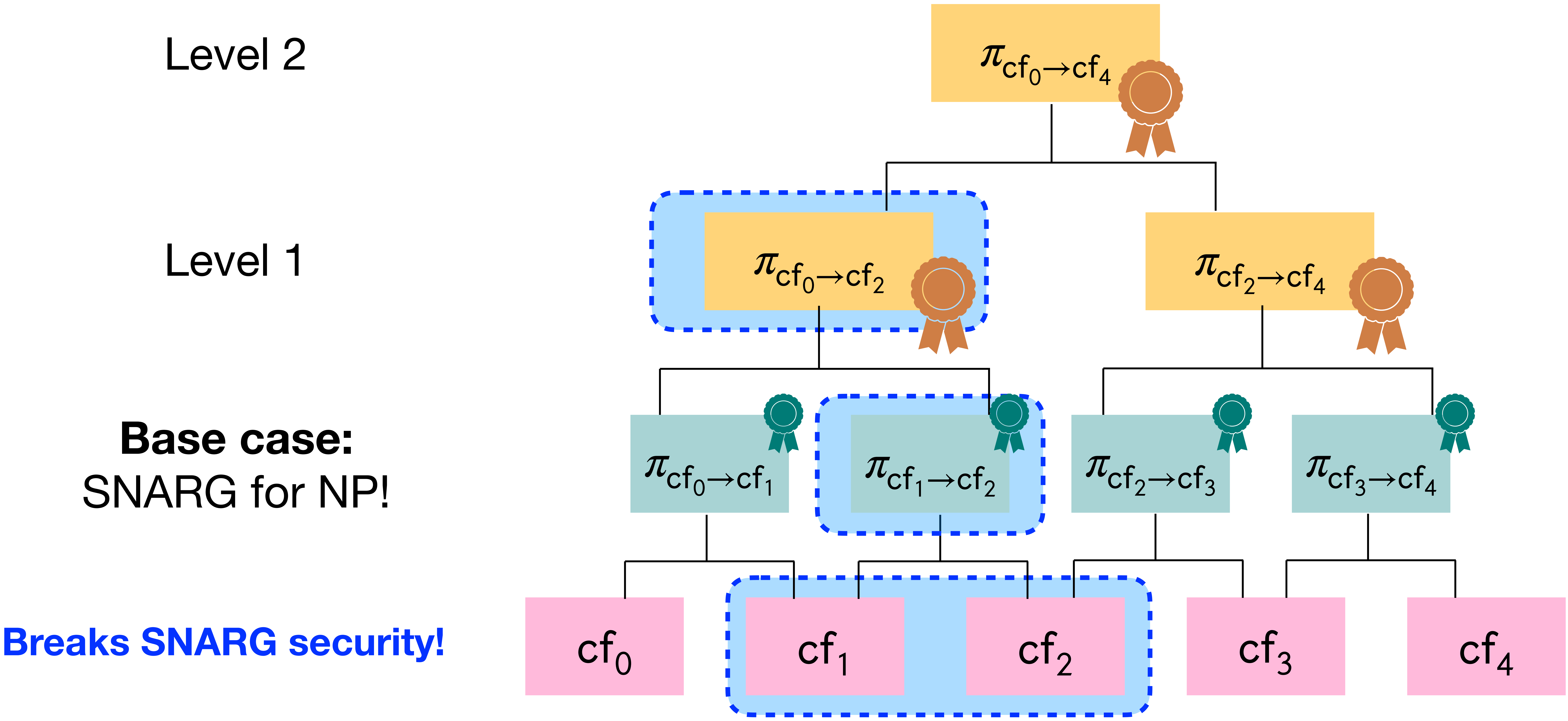
Level 2

Level 1

Base case:
SNARG for NP!



Security sketch with “Somewhere Soundness”



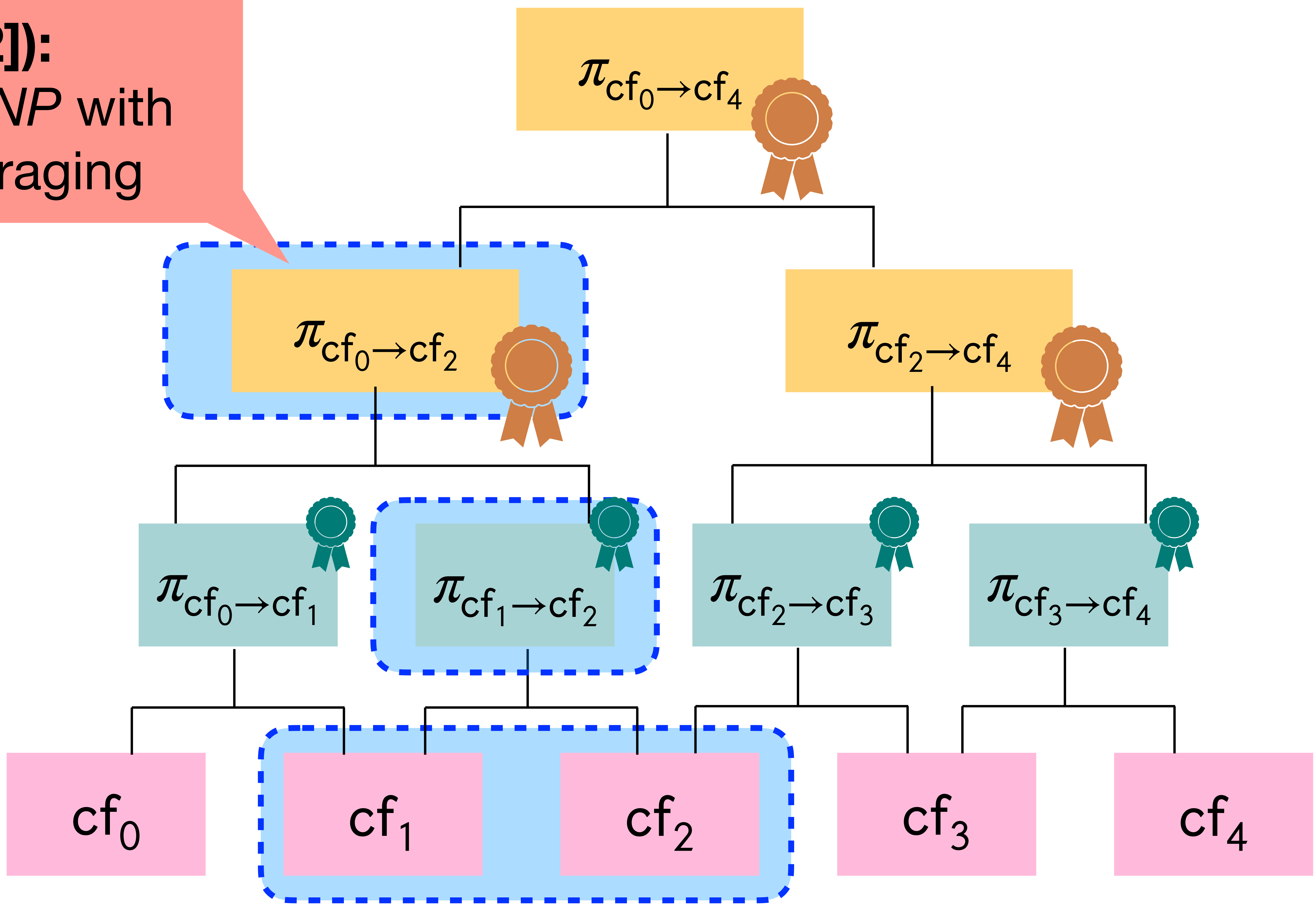
Security sketch with “Somewhere Soundness”

Our main technical contribution (over [DGKV/PP22]):
We show this works for *NP* with careful complexity leveraging

Level 1

Base case:
SNARG for NP!

Breaks SNARG security!



IVC for NP!

Theorem [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: poly$(\lambda, |\mathsf{cf}|, \log T)$.

IVC for NP!

Theorem [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: poly$(\lambda, |\text{cf}|, \log T)$.

Corollary [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

IVC for NP!

Theorem [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: poly$(\lambda, |\mathsf{cf}|, \log T)$.

Corollary [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

IVC for NP!

Theorem [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: poly$(\lambda, |\text{cf}|, \log T)$.

Corollary [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: poly$(\lambda, |\text{cf}|, |w_i|, \log T)$.

IVC for NP!

Theorem [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: poly$(\lambda, |\mathsf{cf}|, \log T)$.

Corollary [DJJMKS25]. There exists IVC for NP assuming subexponential hardness of:

Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: poly$(\lambda, |\mathsf{cf}|, |w_i|, \log T)$.

Independent of time-steps, but grows with intermediate configurations cf_i **and witness**

IVC for NP! ✓

Theorem [DJJMKS25]. There exists IVC for NP assuming <u>subexponential</u> hardness of:

- Non-adaptive SNARG for NP (known from iO + OWF).
- Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: $\text{poly}(\lambda, |\text{cf}|, \log T)$.

Corollary [DJJMKS25]. There exists IVC for NP assuming <u>subexponential</u> hardness of:

Rate-1 Somewhere Extractable BARGs (known from LWE/DLIN/etc).

Proof size: $\text{poly}(\lambda, |\text{cf}|, |w_i|, \log T)$.

Independent of time-steps, but grows with intermediate configurations cf_i **and witness**

More Succinct IVC?

More Succinct IVC?

- We know **adaptively sound SNARGs** with $\text{poly}(\lambda)$ proof size [WW24/25, WZ24]

More Succinct IVC?

- We know **adaptively sound SNARGs** with $\text{poly}(\lambda)$ proof size [WW24/25, WZ24]

- If we follow the **“proof merging”** template, it seems like we are stuck with this **“configuration”** size dependence.

More Succinct IVC?

- We know **adaptively sound SNARGs** with $\text{poly}(\lambda)$ proof size [WW24/25, WZ24]
- If we follow the **"proof merging"** template, it seems like we are stuck with this **"configuration"** size dependence.
    - Need to extract the intermediate configuration!

More Succinct IVC?

- We know **adaptively sound SNARGs** with $\text{poly}(\lambda)$ proof size [WW24/25, WZ24]
- If we follow the **“proof merging”** template, it seems like we are stuck with this **“configuration”** size dependence.
  - Need to extract the intermediate configuration!
- **Can we construct IVC without any extraction?**

More Succinct IVC?

- We know **adaptively sound SNARGs** with $\mathrm{poly}(\lambda)$ proof size [WW24/25, WZ24]
- If we follow the **"proof merging"** template, it seems like we are stuck with this **"configuration"** size dependence.
  - Need to extract the intermediate configuration!
- **Can we construct IVC without any extraction?**
  - SNARGs without extraction? Smells like iO :)

Our Result II: Even More Succinctness

Our Result II: Even More Succinctness

Theorem [DJJMKS25]. There exists IVC for ***trapdoor computations*** $\mathscr{M}$ assuming subexponential hardness of iO and injective PRGs.

Our Result II: Even More Succinctness

Theorem [DJJMKS25]. There exists IVC for ***trapdoor computations*** $\mathscr{M}$ assuming subexponential hardness of iO and injective PRGs.

Definition: $\exists$Td such that $x_i \rightarrow^{\mathscr{M}} x_j$ iff $\mathsf{Td}(x_i, x_j, j - i) = \mathsf{Accept}$.

Our Result II: Even More Succinctness

Theorem [DJJMKS25]. There exists IVC for ***trapdoor computations*** $\mathscr{M}$ assuming subexponential hardness of iO and injective PRGs.

- Proof size: poly$(\lambda, \log T)$!

Definition: $\exists$Td such that $x_i \rightarrow^{\mathscr{M}} x_j$ iff Td$(x_i, x_j, j - i) =$ Accept.

Our Result II: Even More Succinctness

Theorem [DJJMKS25]. There exists IVC for ***trapdoor computations*** $\mathscr{M}$ assuming subexponential hardness of iO and injective PRGs.

- Proof size: poly$(\lambda, \log T)$!

Definition: $\exists$Td such that $x_i \to^{\mathscr{M}} x_j$ iff Td$(x_i, x_j, j - i) =$ Accept.

- Trapdoor only appears in the proof

Our Result II: Even More Succinctness

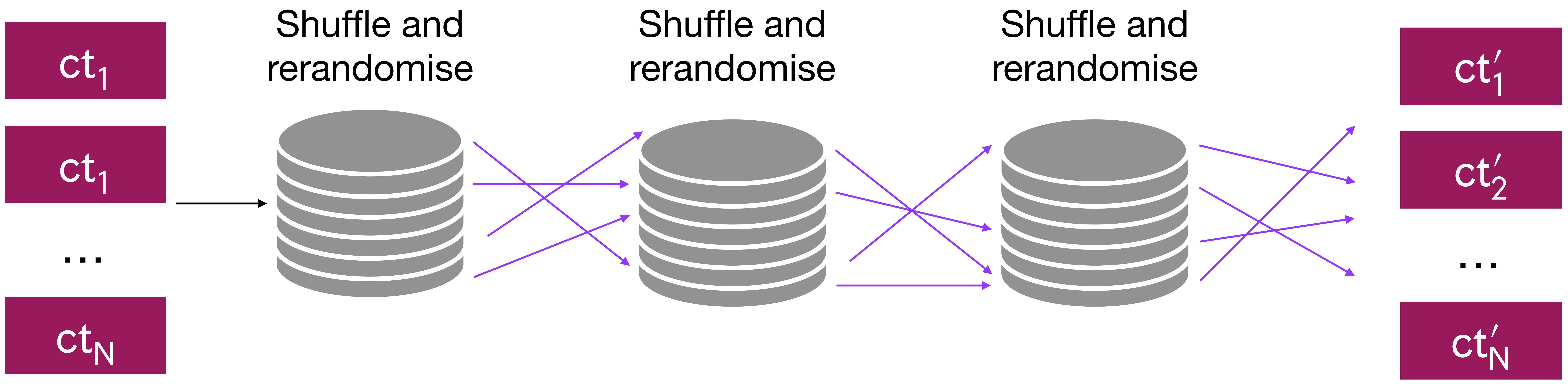
Theorem [DJJMKS25]. There exists IVC for ***trapdoor computations*** $\mathscr{M}$ assuming subexponential hardness of iO and injective PRGs.

- Proof size: poly$(\lambda, \log T)$!

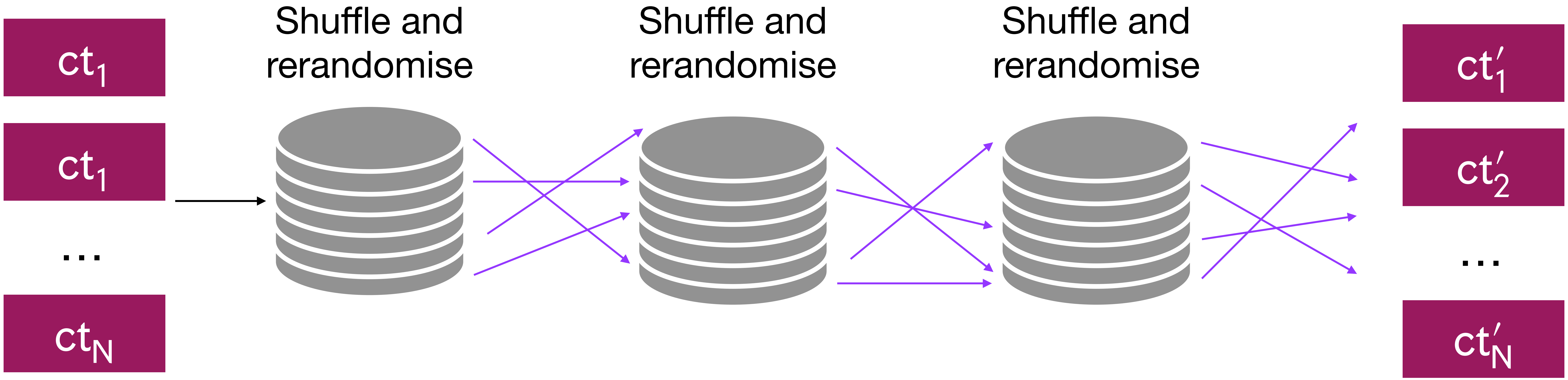
Definition: $\exists$Td such that $x_i \to^{\mathscr{M}} x_j$ iff Td$(x_i, x_j, j - i) =$ Accept.

- Trapdoor only appears in the proof
    - Can generate the CRS and guarantee soundness *only knowing* that a trapdoor exists

Verifiable Shuffling

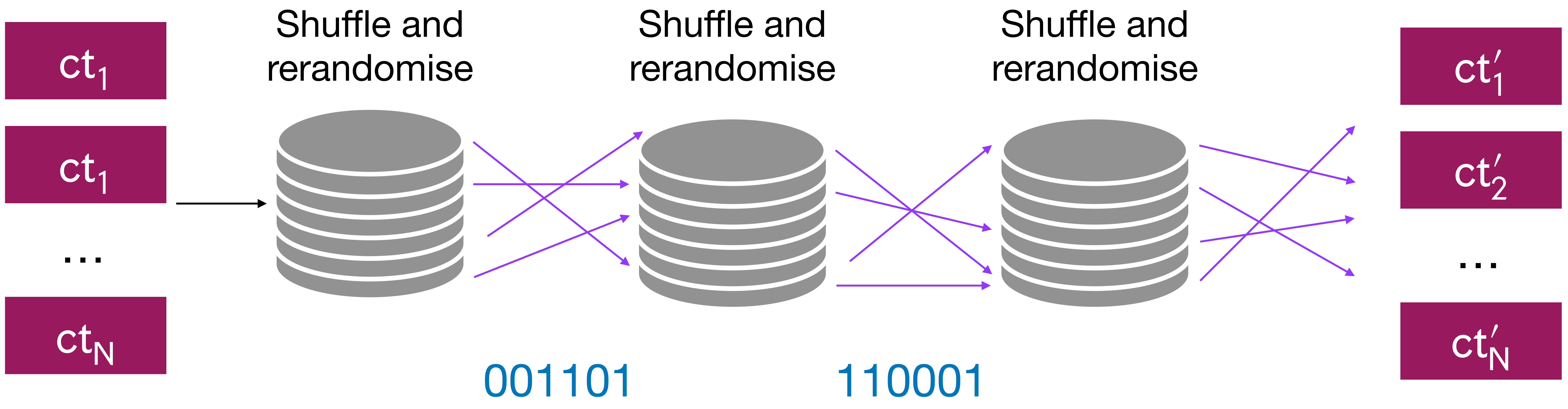


Verifiable Shuffling



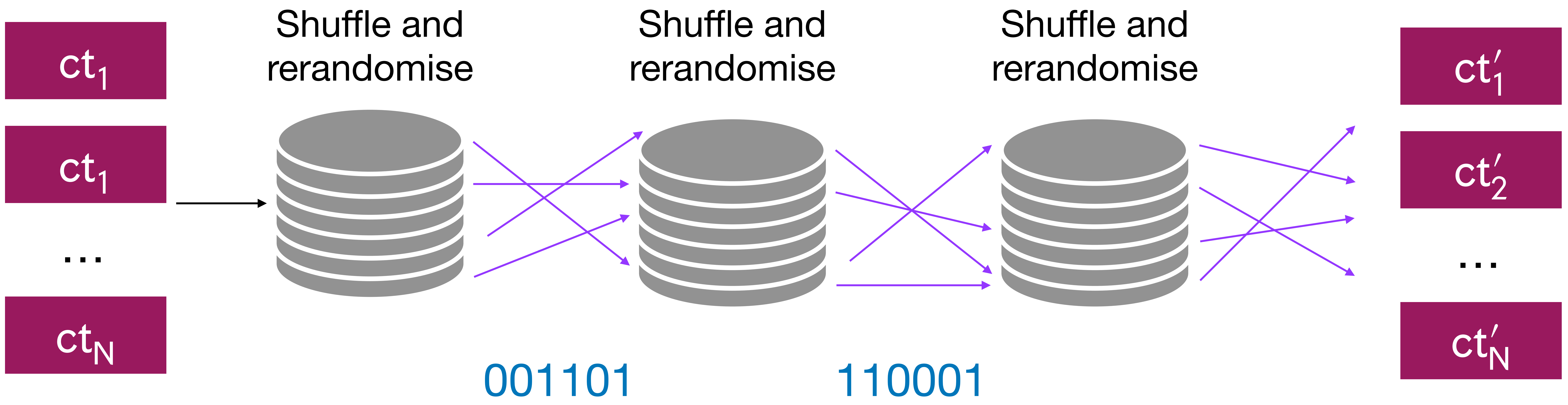
- Trapdoor = Secret key of rerandomisable scheme!

Verifiable Shuffling



- Trapdoor = Secret key of rerandomisable scheme!

Verifiable Shuffling



- Trapdoor = Secret key of rerandomisable scheme!
- Our work gives a **multi-hop** verifiable shuffling scheme with **short proofs** from **standard assumptions** in the plain model.

Our Result II: More Succinct IVC

Theorem [DJJMKS25]. There exists IVC for ***trapdoor computations*** $\mathscr{M}$ assuming subexponential hardness of iO and injective PRGs.

- Proof size: poly$(\lambda, \log T)$!

Definition: $\exists$Td such that $x_i \to^{\mathscr{M}} x_j$ iff $\mathsf{Td}(x_i, x_j, j-i) =$ Accept.

Our Result II: More Succinct IVC

Theorem [DJJMKS25]. There exists IVC for ***trapdoor computations*** $\mathscr{M}$ assuming subexponential hardness of iO and injective PRGs.

- Proof size: poly$(\lambda, \log T)$!

Definition: $\exists$Td such that $x_i \to^{\mathscr{M}} x_j$ iff Td$(x_i, x_j, j - i) =$ Accept.

- Trapdoor only appears in the proof

Our Result II: More Succinct IVC

Theorem [DJJMKS25]. There exists IVC for ***trapdoor computations*** $\mathcal{M}$ assuming subexponential hardness of iO and injective PRGs.

- Proof size: $\text{poly}(\lambda, \log T)$!

Definition: $\exists$Td such that $x_i \rightarrow^{\mathcal{M}} x_j$ iff $\text{Td}(x_i, x_j, j - i) = \text{Accept}$.

- Trapdoor only appears in the proof
- Highly inspired by "pure iO" adaptive SNARG constructions of [WW24/25, WZ24, DWW24] and "chaining approach" of [GSWW22, DWW24].

Our Result II: More Succinct IVC

Theorem [DJJMKS25]. There exists IVC for ***trapdoor computations*** $\mathcal{M}$ assuming subexponential hardness of iO and injective PRGs.

- Proof size: poly$(\lambda, \log T)$!

Definition: $\exists$Td such that $x_i \rightarrow^{\mathcal{M}} x_j$ iff Td$(x_i, x_j, j - i) =$ Accept.

- Trapdoor only appears in the proof
- Highly inspired by "pure iO" adaptive SNARG constructions of [WW24/25, WZ24, DWW24] and "chaining approach" of [GSWW22, DWW24].
  - **Idea:** Do proof *without* extraction!

Construction

Construction

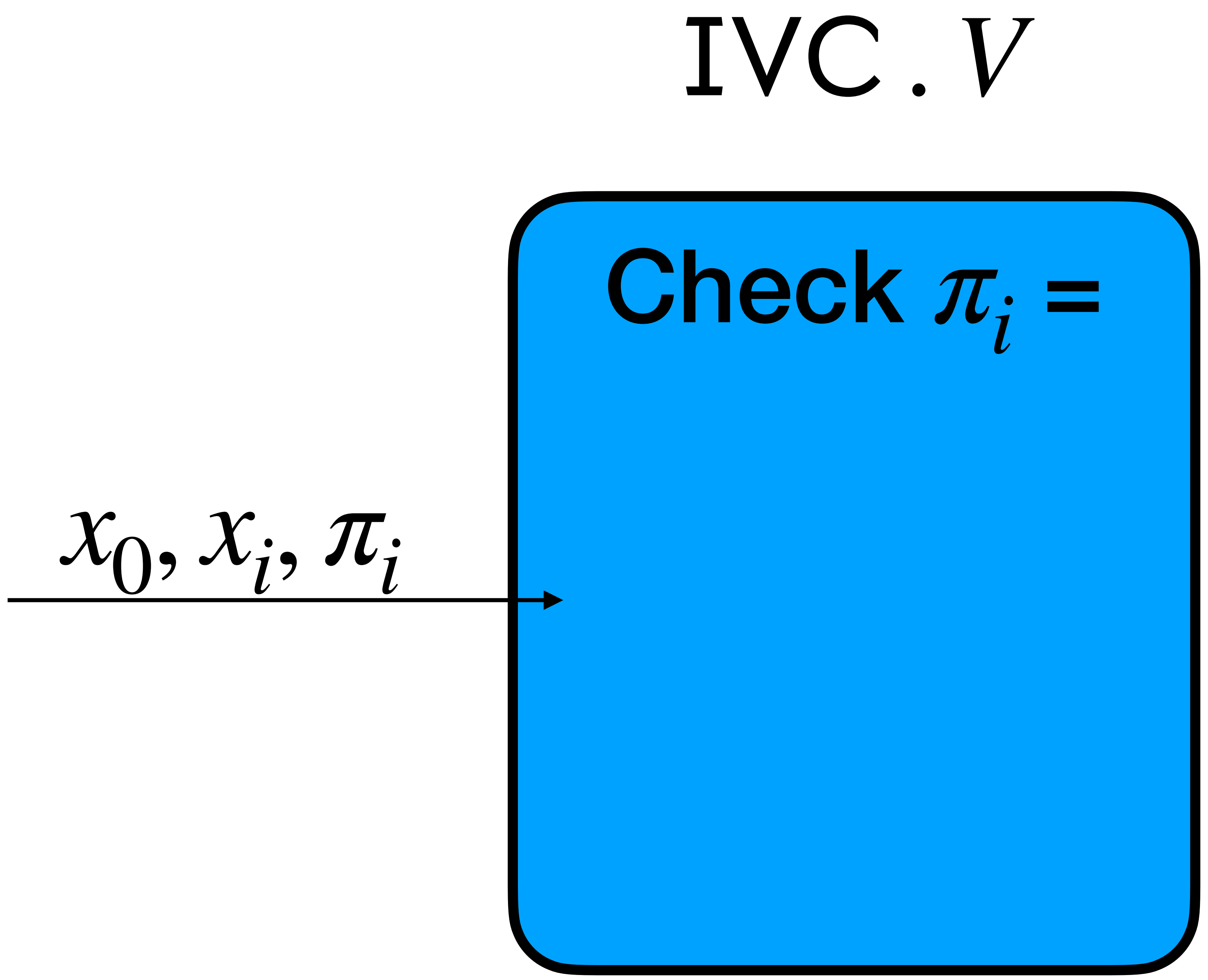
$\texttt{IVC}.V$

Construction

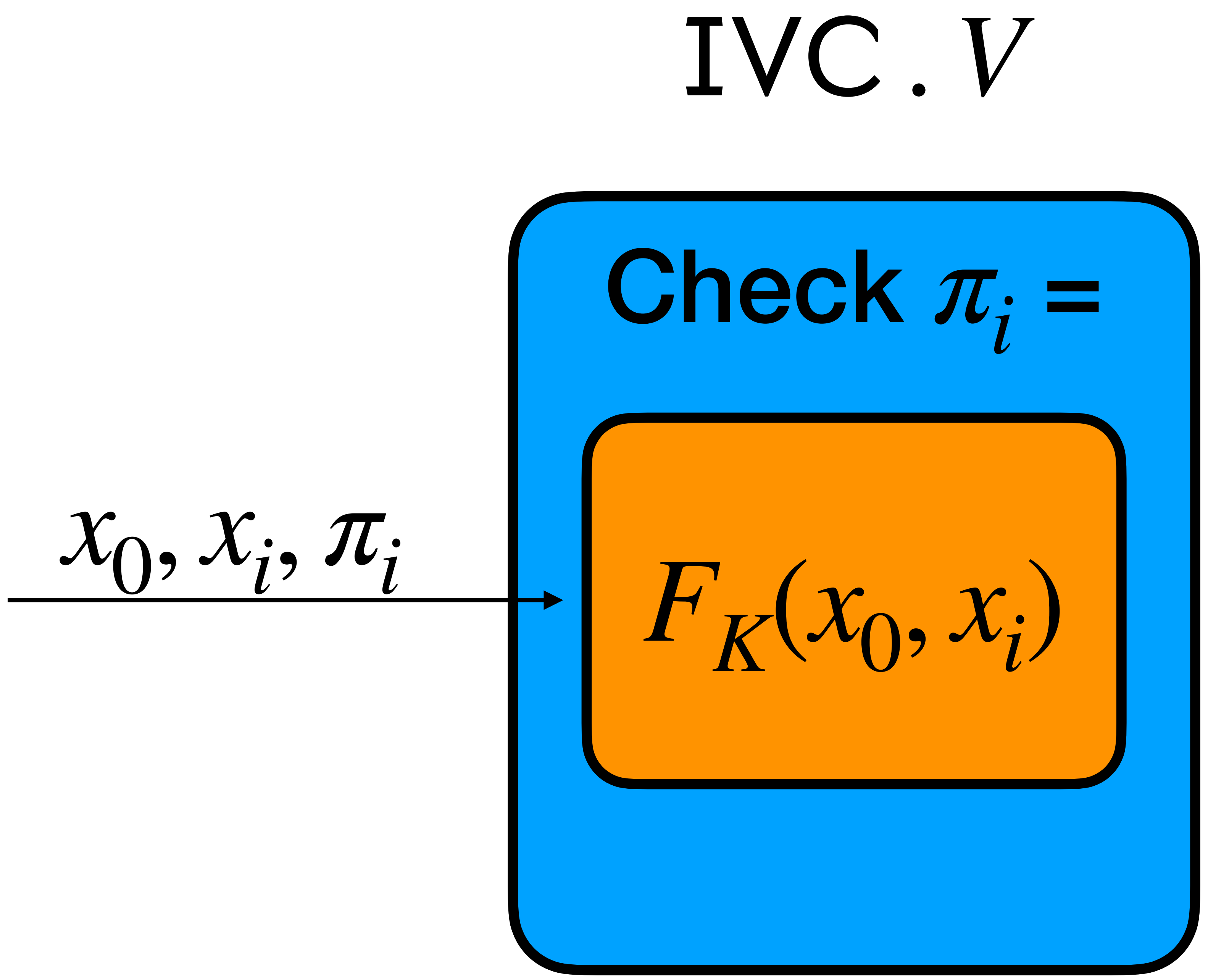
$\texttt{IVC}.V$

x_0, x_i, π_i →

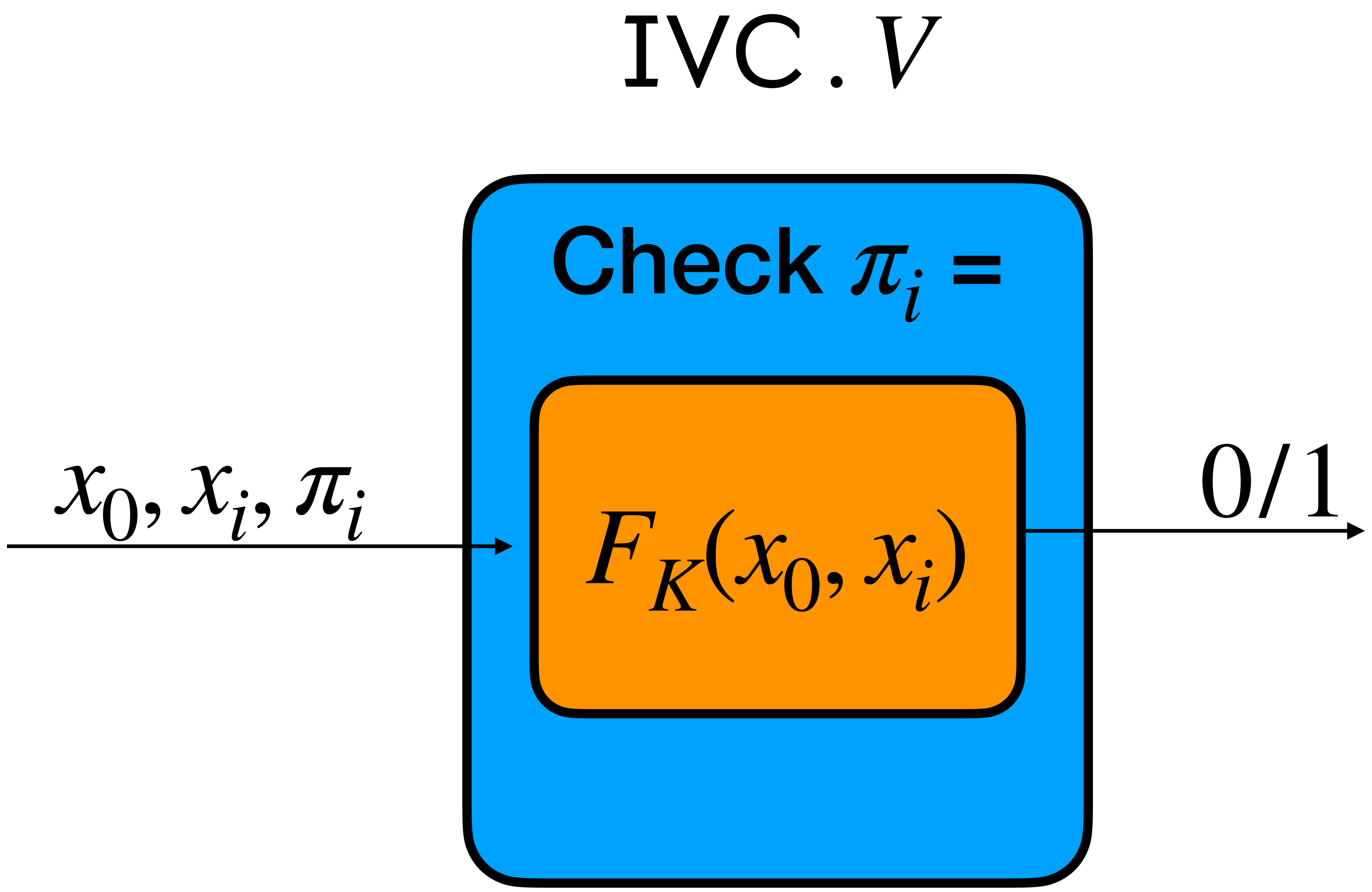
Construction



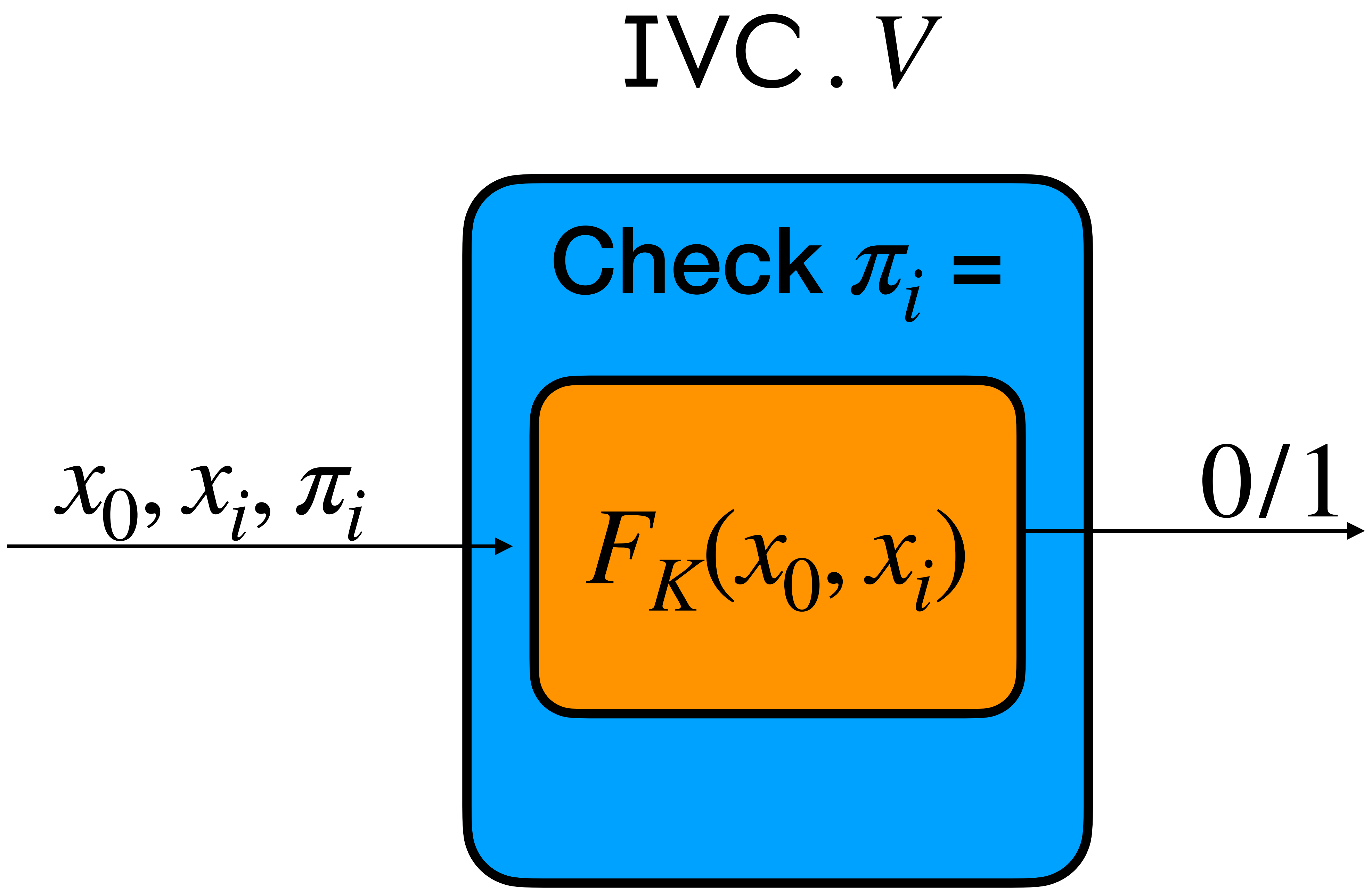
Construction



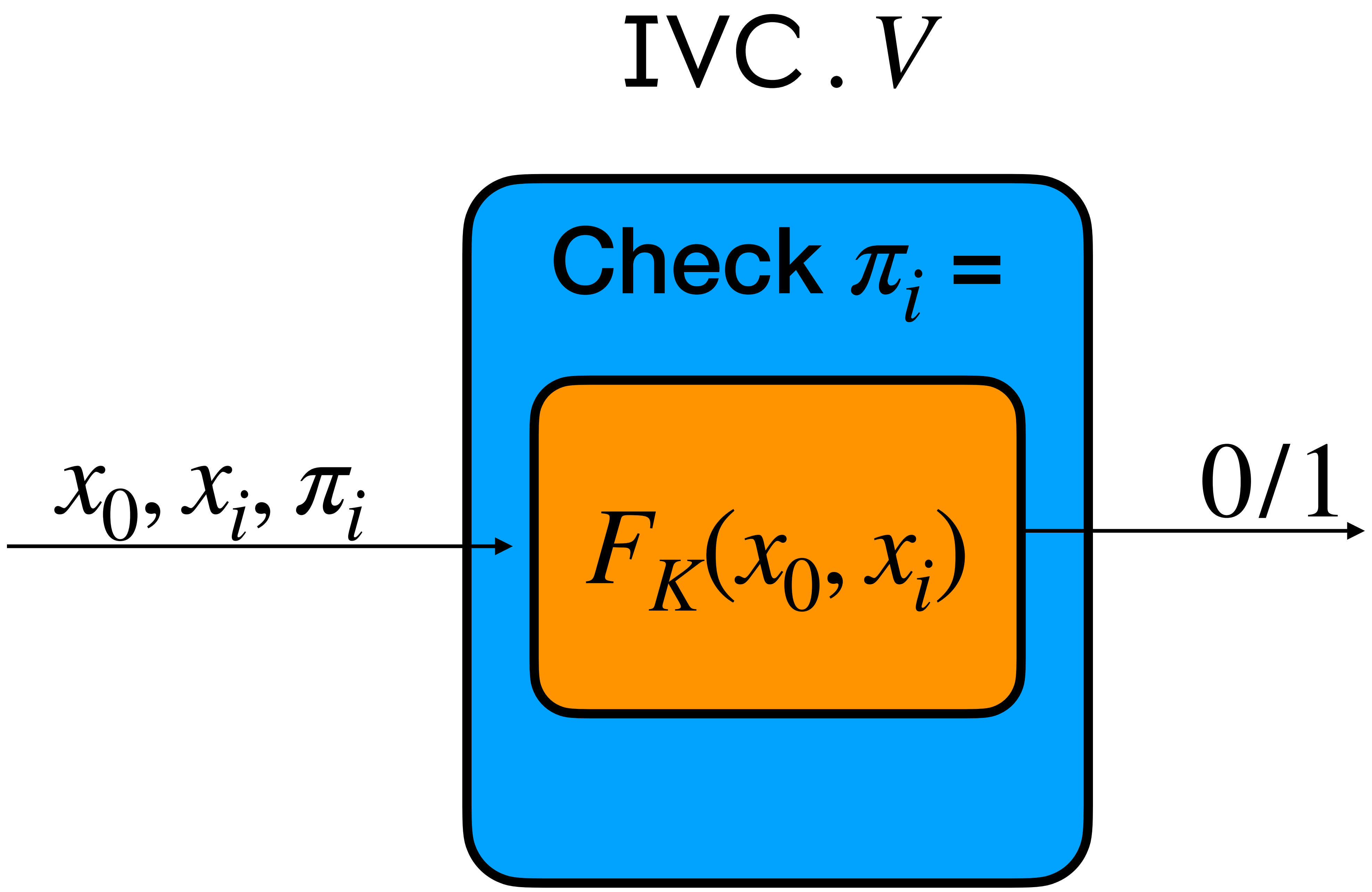
Construction



Construction

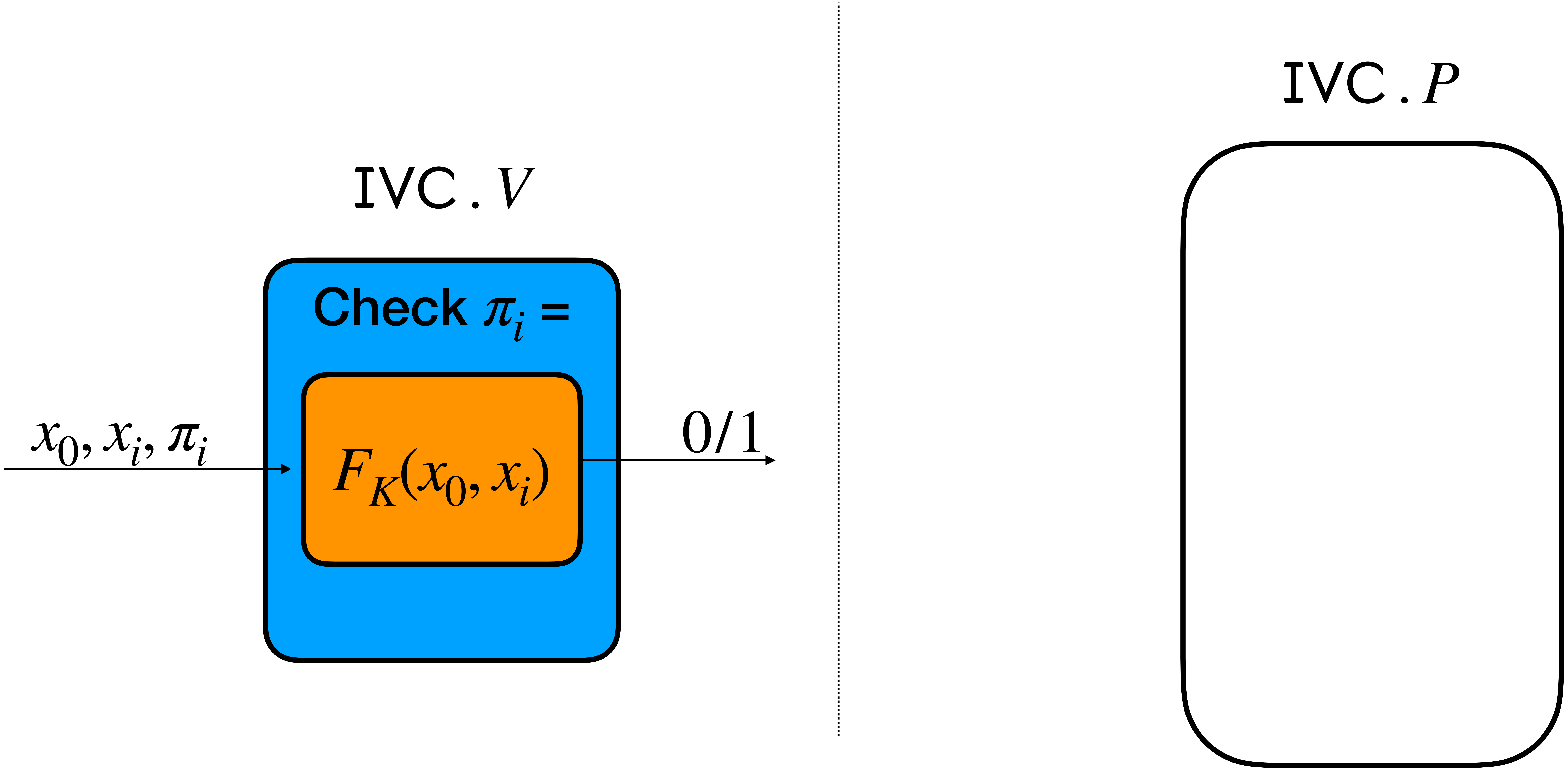


Construction

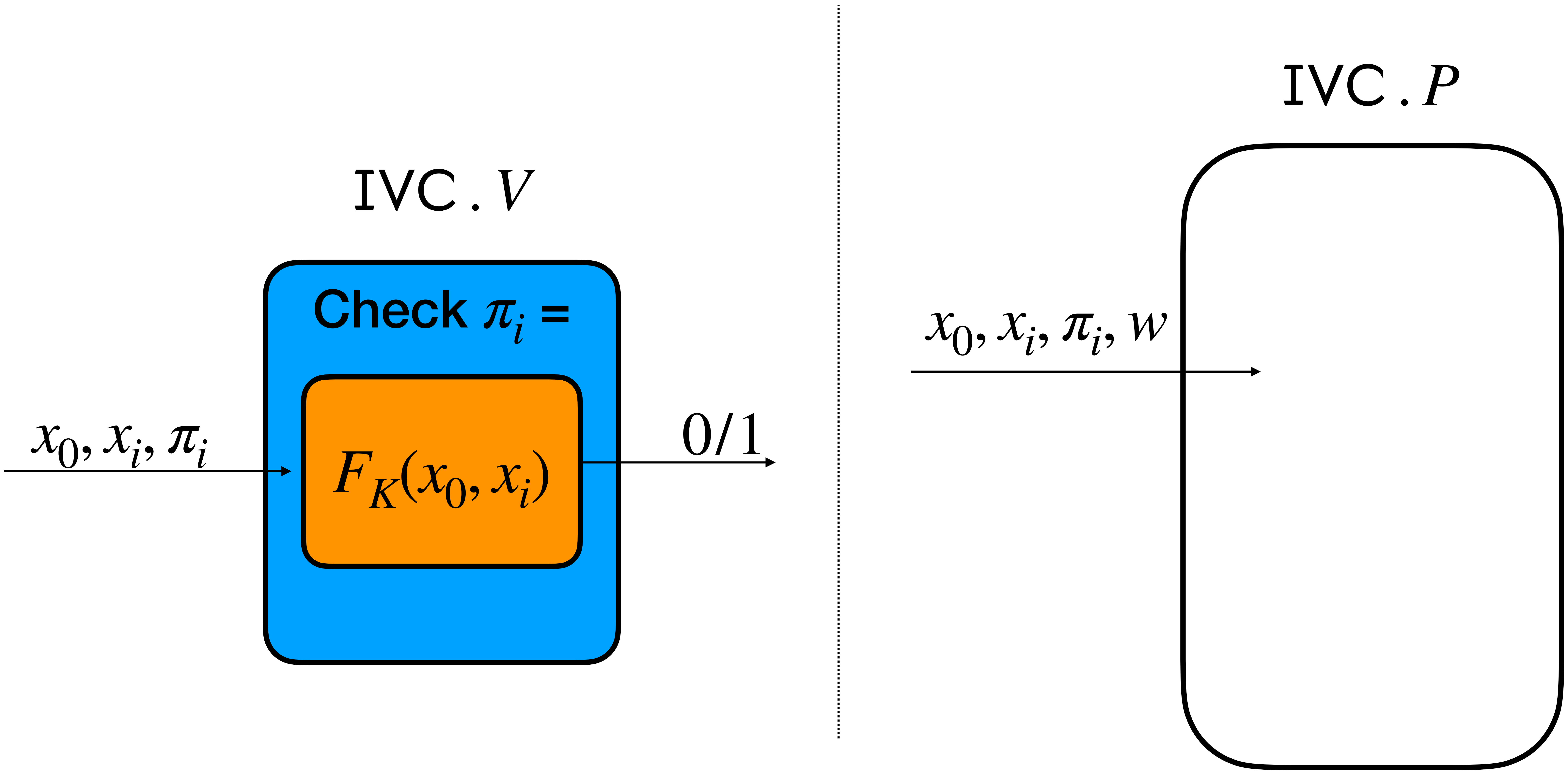


IVC.P

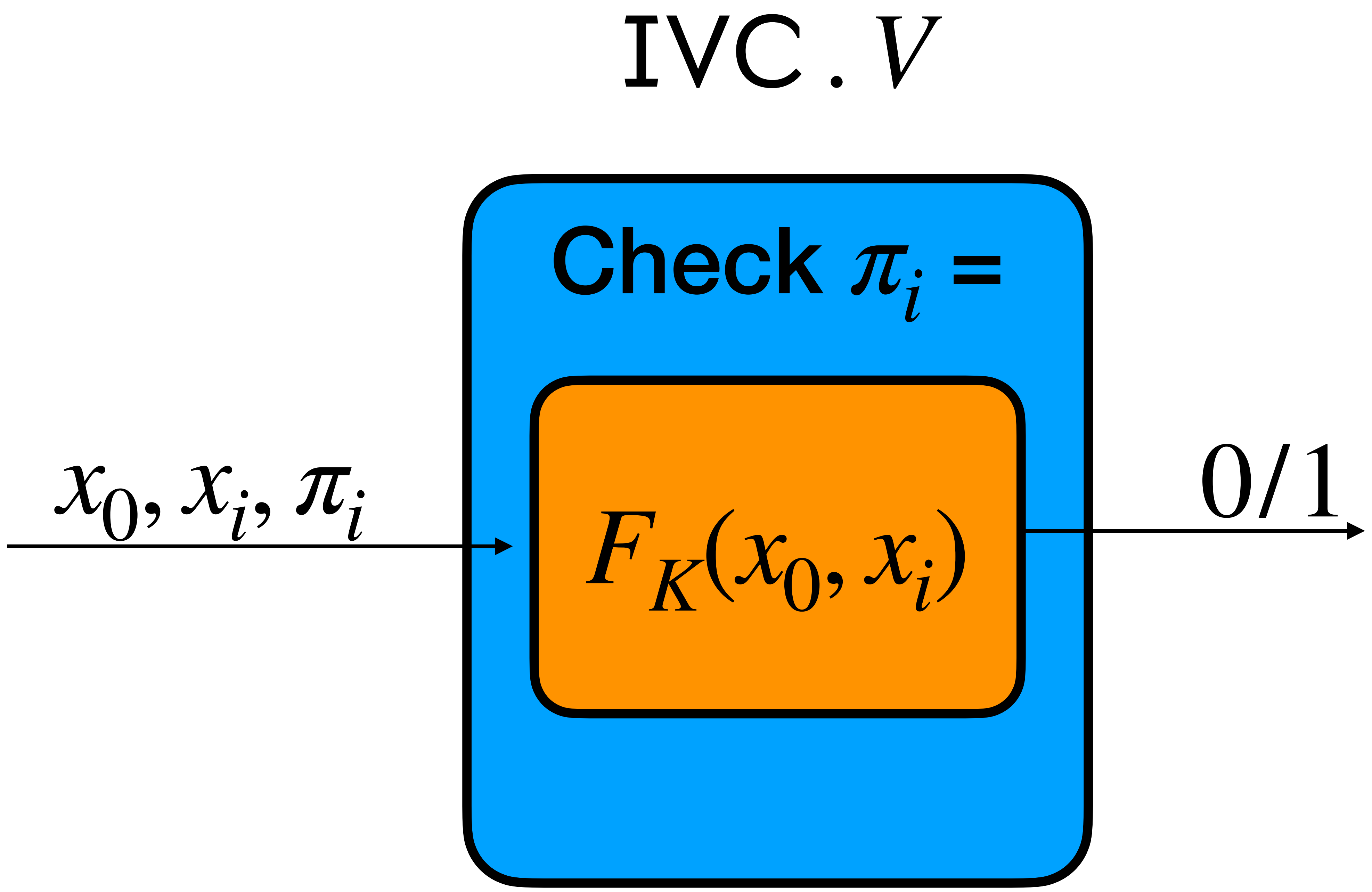
Construction

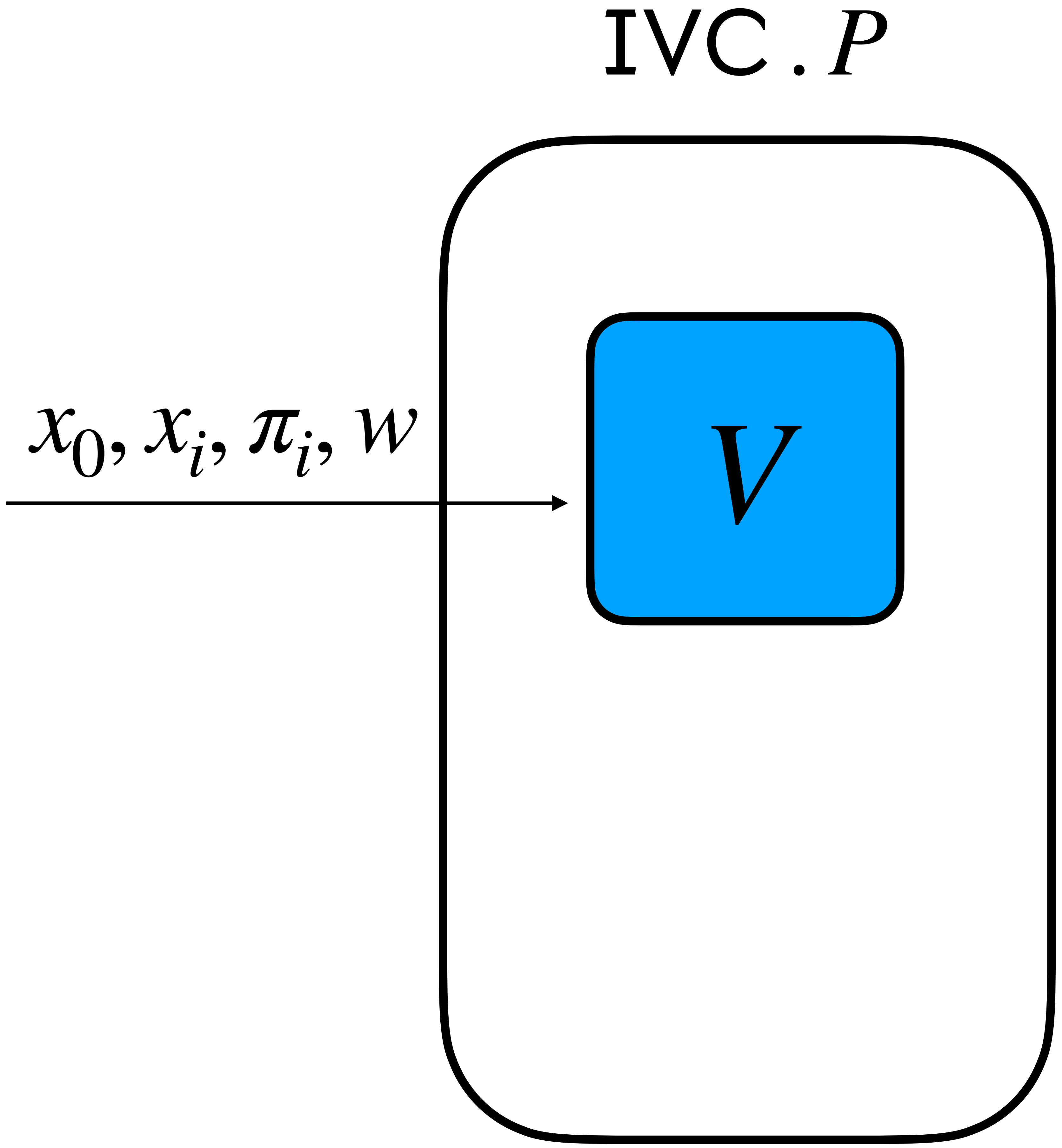


Construction

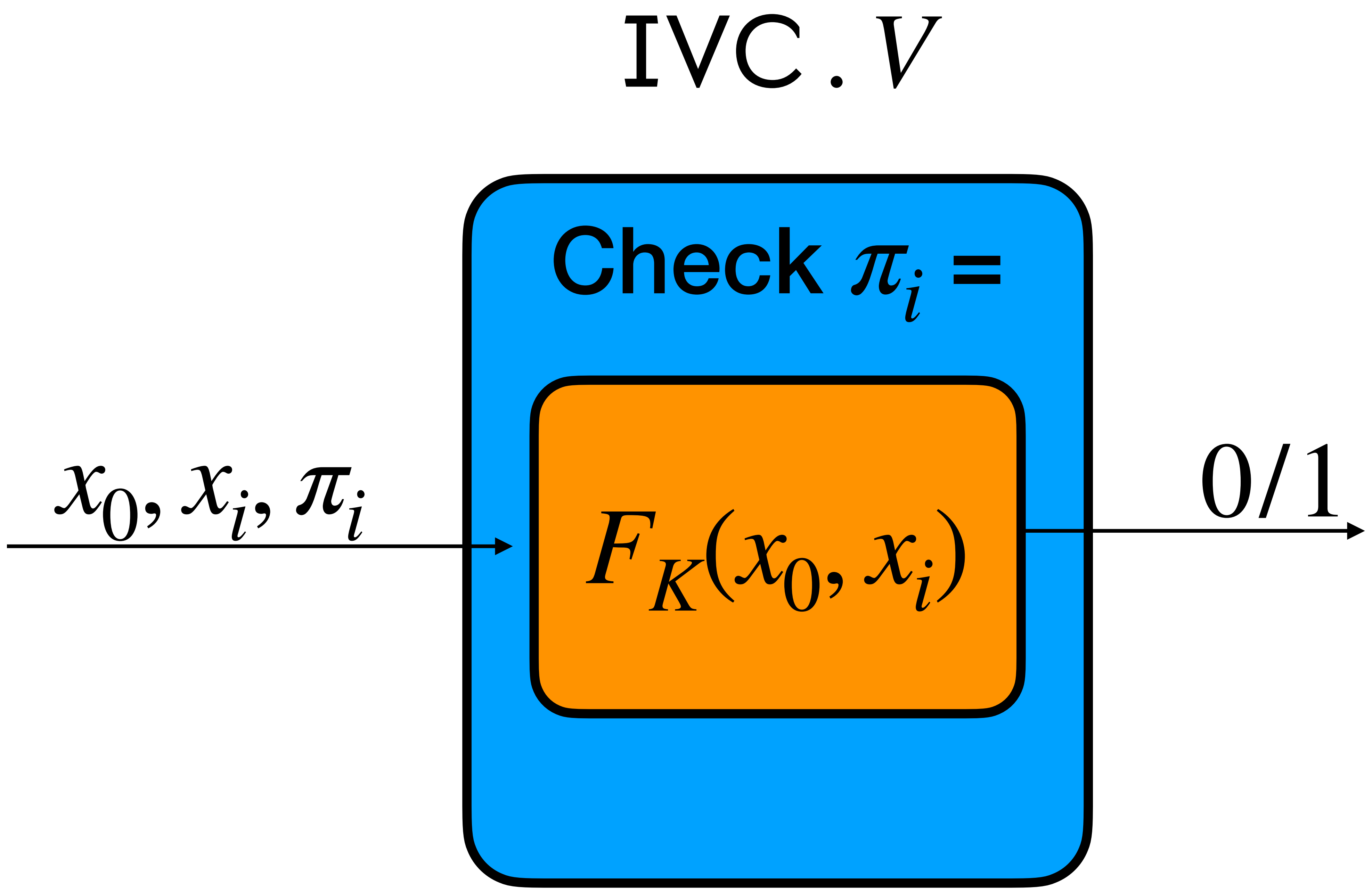


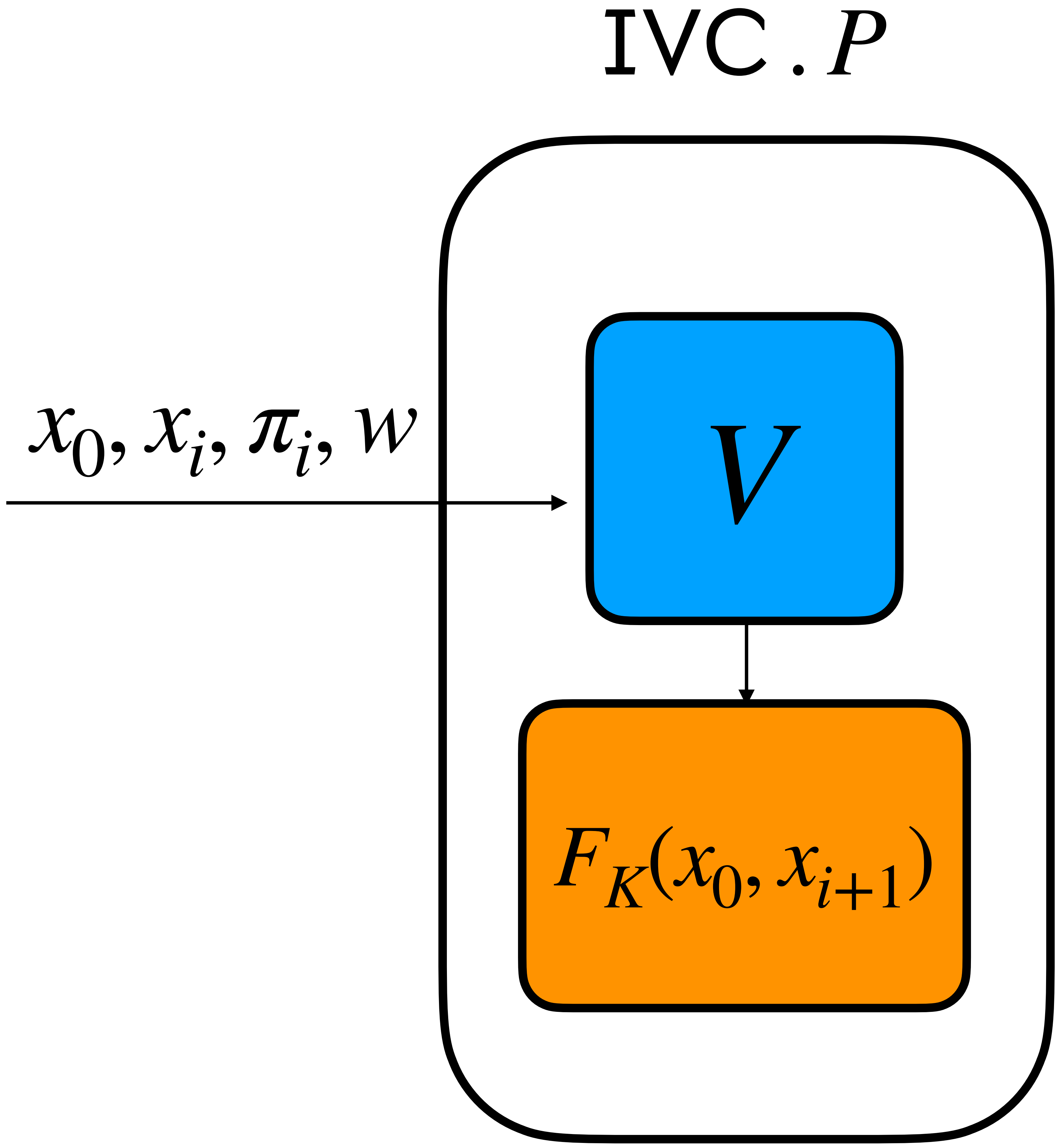
Construction



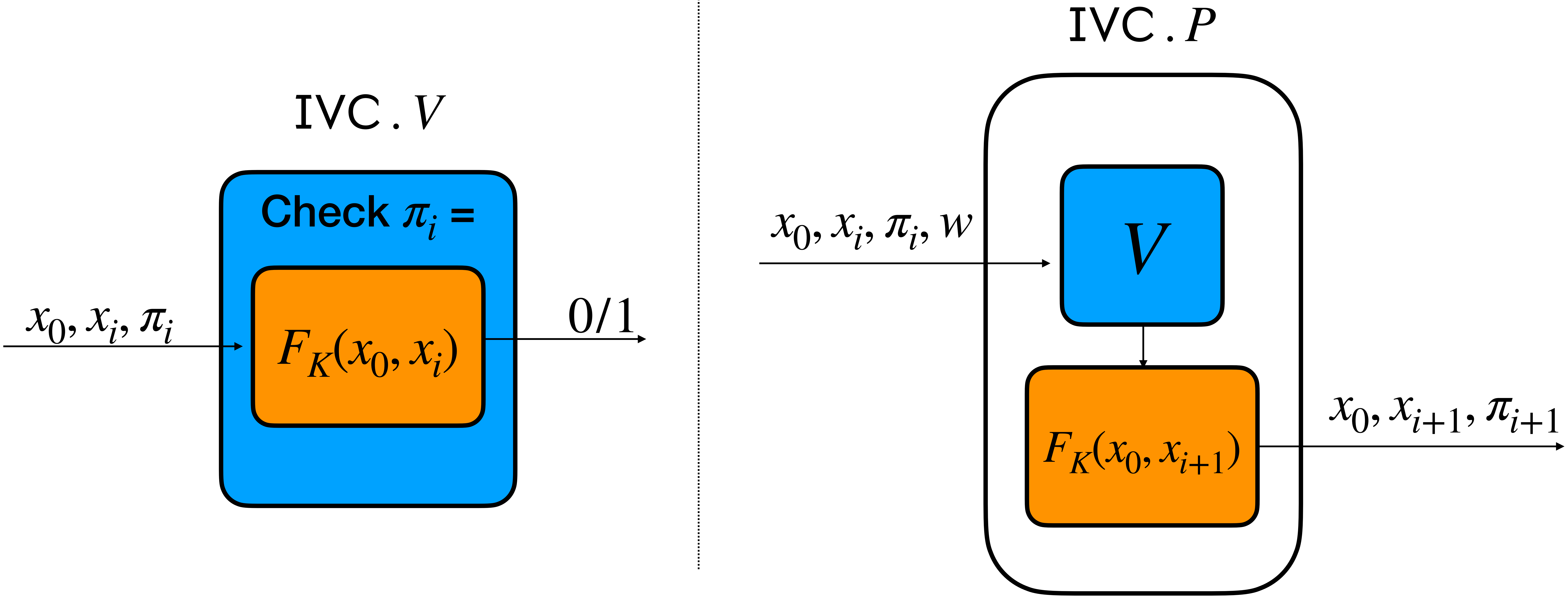


Construction

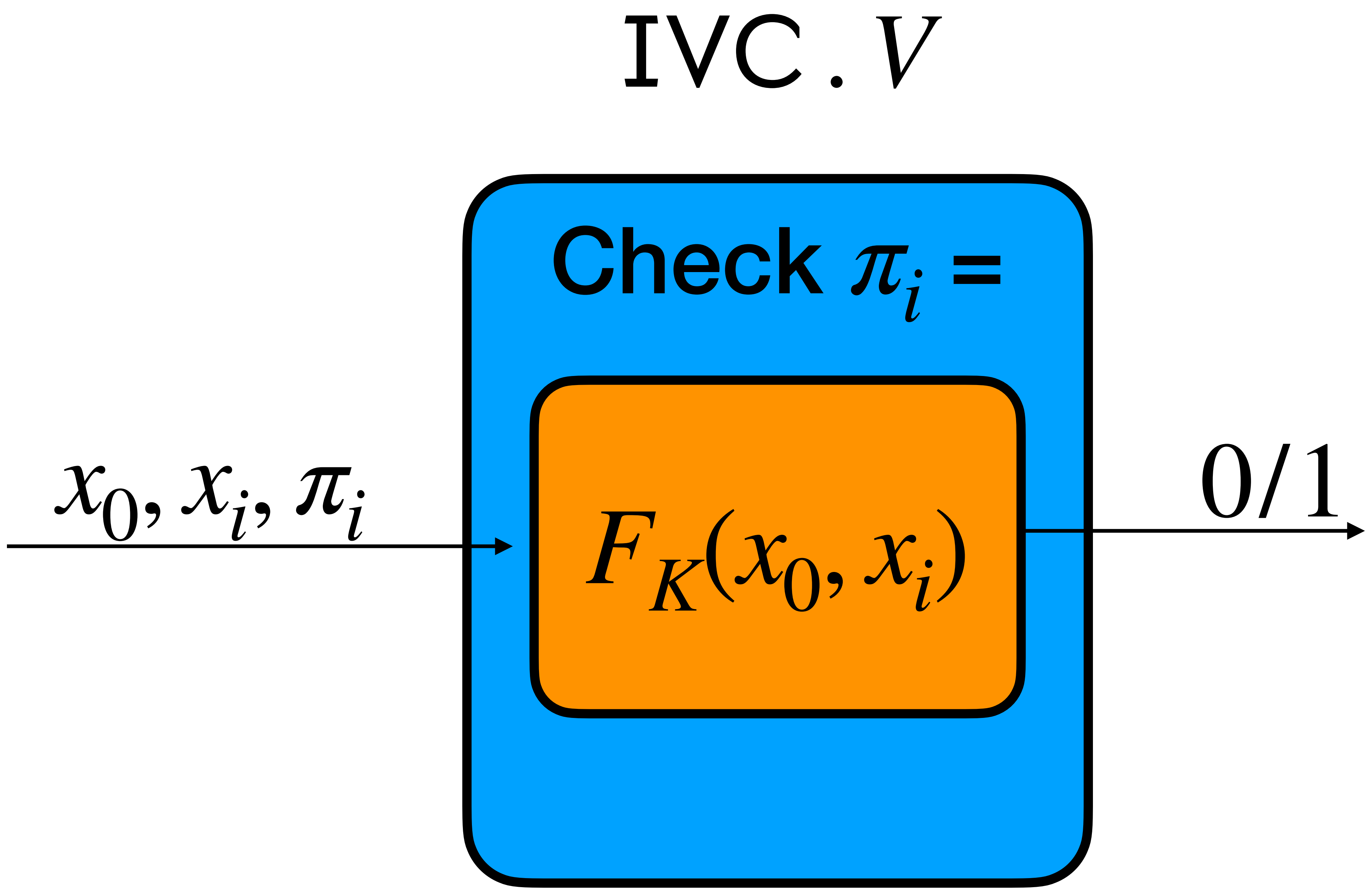


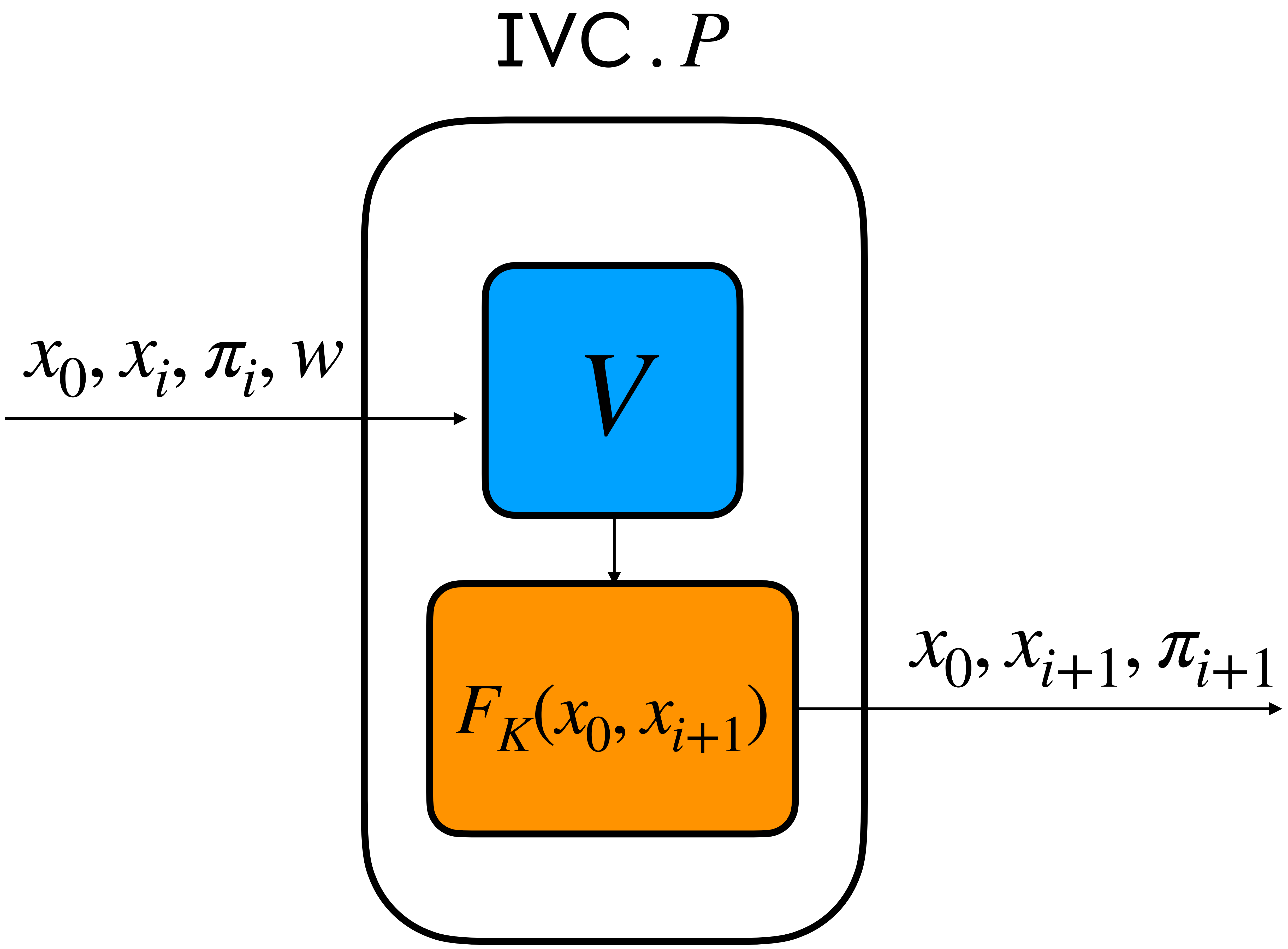


Construction



Construction





Obfuscate these programs and publish as CRS!!

High level idea!

High level idea!

H_{real}

High level idea!

The real scheme

High level idea!

H_{real}

iO Gymnastics! (See paper)

The real scheme

High level idea!

H_{real} → H_{ideal}

iO Gymnastics! (See paper)

The real scheme

High level idea!

iO Gymnastics! (See paper)

H$_{real}$ — The real scheme

H$_{ideal}$ — **No proofs** for **false** statements (uses Td)

High level idea!

iO Gymnastics! (See paper)

H_{real} — The real scheme

H_{ideal} — **No proofs** for **false** statements (uses Td)

- **Idea:** Cheating prover → Distinguisher for these hybrids

High level idea!

H_{real}

iO Gymnastics! (See paper)

H_{ideal}

The real scheme

No proofs for **false** statements (uses Td)

- **Idea:** Cheating prover → Distinguisher for these hybrids
    - Can calculate cheating prover's success probability in H_{real} using Td.

High level idea!

H_{real} *iO Gymnastics! (See paper)* H_{ideal}

The real scheme

No proofs for **false** statements (uses Td)

- **Idea:** Cheating prover → Distinguisher for these hybrids
  - Can calculate cheating prover's success probability in H_{real} using Td.
  - This gives us soundness :)

High level idea!

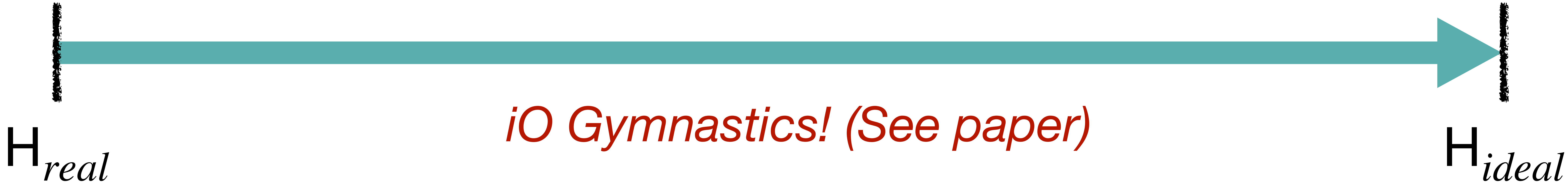
iO Gymnastics! (See paper)

H_{real} — The real scheme

H_{ideal} — **No proofs** for **false** statements (uses Td)

- **Idea:** Cheating prover → Distinguisher for these hybrids
  - Can calculate cheating prover's success probability in H_{real} using Td.
  - This gives us soundness :)
- Marries ideas from [GSWW22], [DWW24], [WW25]!

High level idea!



The real scheme

No proofs for **false** statements (uses Td)

- **Idea:** Cheating prover → Distinguisher for these hybrids
  - Can calculate cheating prover's success probability in H_{real} using Td.
  - This gives us soundness :)
- Marries ideas from [GSWW22], [DWW24], [WW25]!
- This approach **does not** use extraction and achieves poly(λ)-**sized proof**!

Summary

Summary

- To extract or not to extract?

Summary

- To extract or not to extract?
- We show two IVC constructions in the *nondeterministic* setting.

Summary

- To extract or not to extract?
- We show two IVC constructions in the *nondeterministic* setting.
  - We give an **extraction-based** approach to achieve IVC for NP.

Summary

- To extract or not to extract?
- We show two IVC constructions in the *nondeterministic* setting.
  - We give an **extraction-based** approach to achieve IVC for NP.
    - Proof size: $\mathsf{poly}(\lambda, |\mathsf{cf}|, \log T)$.

Summary

- To extract or not to extract?
- We show two IVC constructions in the *nondeterministic* setting.
  - We give an **extraction-based** approach to achieve IVC for NP.
    - Proof size: $\mathsf{poly}(\lambda, |\mathsf{cf}|, \log T)$.
  - We give an **iO-based approach** to IVC for “Trapdoor-NP”.

Summary

- To extract or not to extract?
- We show two IVC constructions in the *nondeterministic* setting.
  - We give an **extraction-based** approach to achieve IVC for NP.
    - Proof size: $\mathsf{poly}(\lambda, |\mathsf{cf}|, \log T)$.
  - We give an **iO-based approach** to IVC for “Trapdoor-NP”.
    - Demonstrates a new approach to IVC without extraction!

Summary

- To extract or not to extract?
- We show two IVC constructions in the *nondeterministic* setting.
  - We give an **extraction-based** approach to achieve IVC for NP.
    - Proof size: $\text{poly}(\lambda, |\text{cf}|, \log T)$.
  - We give an **iO-based approach** to IVC for “Trapdoor-NP”.
    - Demonstrates a new approach to IVC without extraction!
    - Proof size: $\text{poly}(\lambda)$.

Summary

- To extract or not to extract?
- We show two IVC constructions in the *nondeterministic* setting.
  - We give an **extraction-based** approach to achieve IVC for NP.
    - Proof size: poly$(\lambda, |\mathsf{cf}|, \log T)$.
  - We give an **iO-based approach** to IVC for "Trapdoor-NP".
    - Demonstrates a new approach to IVC without extraction!
    - Proof size: poly(λ).
    - **Open problem:** Can we extend this to all of NP?

Thank you for your attention!

Bonus Slides

Recall: Rate-1 Batch Arguments (BARG)

Common reference string

$x_1, \ldots, x_k \in L$

Recall: Rate-1 Batch Arguments (BARG)

Common reference string

$x_1, \ldots, x_k \in L$

w_1

w_2

$\ldots$

w_k

Recall: Rate-1 Batch Arguments (BARG)

Common reference string

$x_1, \ldots, x_k \in L$

π

w_1

w_2

$\ldots$

w_k

Recall: Rate-1 Batch Arguments (BARG)

Common reference string

$x_1, \ldots, x_k \in L$

π

Rate 1: $|\pi| \approx |w_i| + \text{poly}(\lambda)$

w_1

w_2

…

w_k

Recall: Rate-1 Batch Arguments (BARG)

Common reference string

$x_1, \ldots, x_k \in L$

π

Rate 1: $|\pi| \approx |w_i| + \text{poly}(\lambda)$

w_1

w_2

$\ldots$

w_k

Usually only require
$|\pi| \ll k \cdot |w_i|$.

Recall: Rate-1 Batch Arguments (BARG)

Common reference string crs(i^*)

$x_1, \ldots, x_k \in L$

π

Rate 1: $|\pi| \approx |w_i| + \text{poly}(\lambda)$

w_1

w_2

w_k

Recall: Rate-1 Batch Arguments (BARG)

Common reference string crs(i^*)

$x_1, \ldots, x_k \in L$

π

w_1

w_2

w_k

Rate 1: $|\pi| \approx |w_i| + \text{poly}(\lambda)$

- **Somewhere soundness:** Can generate crs(i^*) in **trapdoor mode** that let's you extract a witness for x_{i^*}.

Recall: Rate-1 Batch Arguments (BARG)

Common reference string crs(i^*)

$x_1, \ldots, x_k \in L$

π

w_1

w_2

w_k

Rate 1: $|\pi| \approx |w_i| + \text{poly}(\lambda)$

- **Somewhere soundness:** Can generate crs(i^*) in **trapdoor mode** that let's you extract a witness for x_{i^*}.
- **CRS indistinguishability:** crs $\approx$ crs(i^*).

Recall: Rate-1 Batch Arguments (BARG)

Common reference string crs(i^*)

$x_1, \ldots, x_k \in L$

π

w_1

w_2

w_k

Rate 1: $|\pi| \approx |w_i| + \text{poly}(\lambda)$

- **Somewhere soundness:** Can generate crs(i^*) in **trapdoor mode** that let's you extract a witness for x_{i^*}.
- **CRS indistinguishability:** crs $\approx$ crs(i^*).

For construction, see [PP22, DGKV22, BDSZ24]

Proof Strategy: BARG the BARG

Proof Strategy: BARG the BARG

Construct a BARG proof π_{BARG} corresponding to:

BARG Statement: $(\text{cf}, \text{cf}'), (\text{cf}', \text{cf}'')$.

BARG Witness: Level k proofs π, π''.

Level $k+1$ proof: $\pi'' = (\text{cf}', \pi_{\text{BARG}})$.

Proof Strategy: BARG the BARG

Construct a BARG proof π_{BARG} corresponding to:

BARG Statement: $(\text{cf}, \text{cf}'), (\text{cf}', \text{cf}'')$.

BARG Witness: Level k proofs π, π''.

Level $k+1$ proof: $\pi'' = (\text{cf}', \pi_{\text{BARG}})$.

- Suppose $\mathscr{A}$ creates cheating proofs $(\text{cf}', \pi_{\text{BARG}})$ with **prob.** $\geq \epsilon$.

Proof Strategy: BARG the BARG

Construct a BARG proof π_{BARG} corresponding to:

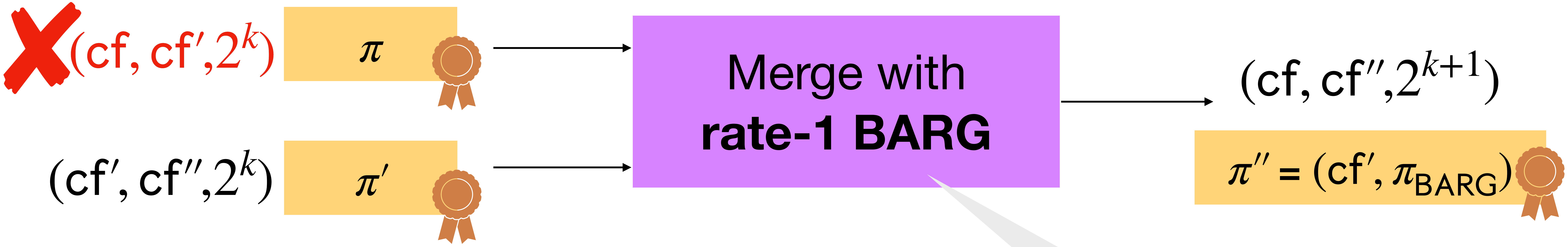
BARG Statement: $(\mathsf{cf}, \mathsf{cf}'), (\mathsf{cf}', \mathsf{cf}'')$.

BARG Witness: Level k proofs π, π''.

Level $k+1$ proof: $\pi'' = (\mathsf{cf}', \pi_{\mathsf{BARG}})$.

- Suppose $\mathscr{A}$ creates cheating proofs $(\mathsf{cf}', \pi_{\mathsf{BARG}})$ with **prob.** $\geq \epsilon$.
    - By **pigeonhole,** $\mathsf{cf} \nrightarrow \mathsf{cf}'$ with prob $\geq \epsilon/2$ or, $\mathsf{cf}' \nrightarrow \mathsf{cf}''$. WLOG first hop.

Proof Strategy: BARG the BARG



Construct a BARG proof π_{BARG} corresponding to:

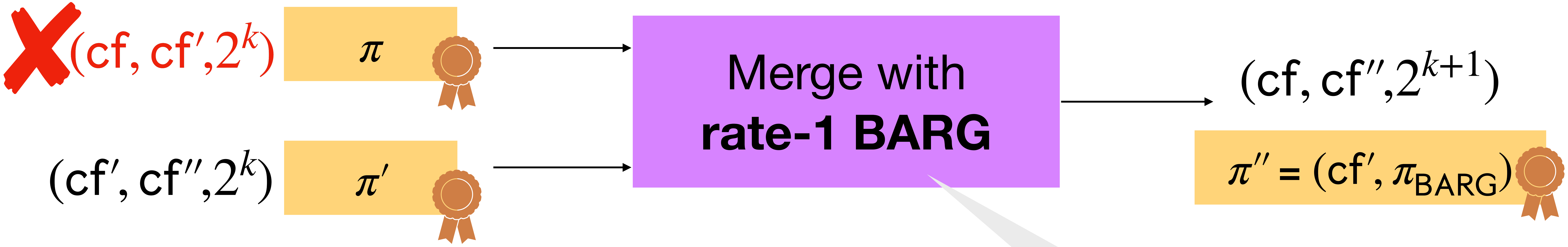
BARG Statement: $(\mathsf{cf}, \mathsf{cf}'), (\mathsf{cf}', \mathsf{cf}'')$.

BARG Witness: Level k proofs π, π''.

Level $k+1$ proof: $\pi'' = (\mathsf{cf}', \pi_{\mathsf{BARG}})$.

- Suppose $\mathscr{A}$ creates cheating proofs $(\mathsf{cf}', \pi_{\mathsf{BARG}})$ with **prob.** $\geq \epsilon$.
  - By **pigeonhole,** $\mathsf{cf} \not\rightarrow \mathsf{cf}'$ with prob $\geq \epsilon/2$ or, $\mathsf{cf}' \not\rightarrow \mathsf{cf}''$. WLOG first hop.

Proof Strategy: BARG the BARG



Construct a BARG proof π_{BARG} corresponding to:

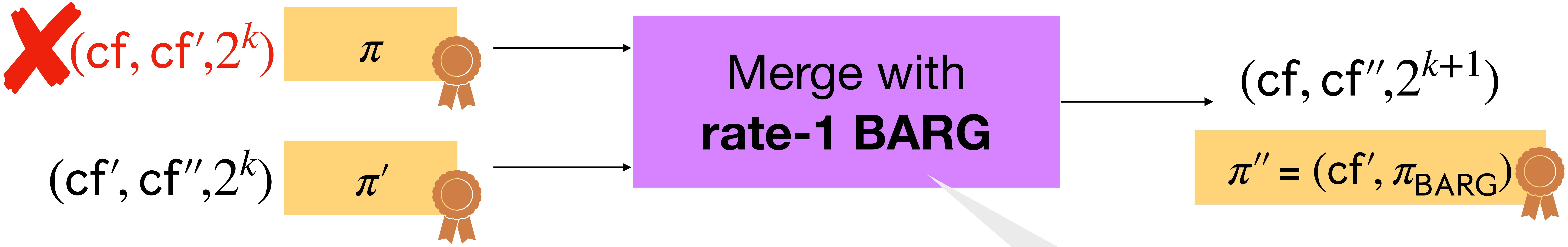
BARG Statement: $(\mathsf{cf}, \mathsf{cf}'), (\mathsf{cf}', \mathsf{cf}'')$.

BARG Witness: Level k proofs π, π''.

Level $k+1$ proof: $\pi'' = (\mathsf{cf}', \pi_{\mathsf{BARG}})$.

- Suppose $\mathscr{A}$ creates cheating proofs $(\mathsf{cf}', \pi_{\mathsf{BARG}})$ with **prob.** $\geq \epsilon$.
  - By **pigeonhole,** $\mathsf{cf} \not\leftrightarrow \mathsf{cf}'$ with prob $\geq \epsilon/2$ or, $\mathsf{cf}' \not\leftrightarrow \mathsf{cf}''$. WLOG first hop.
  - Switch BARG **CRS binding** accordingly!

Proof Strategy: BARG the BARG



Construct a BARG proof π_{BARG} corresponding to:

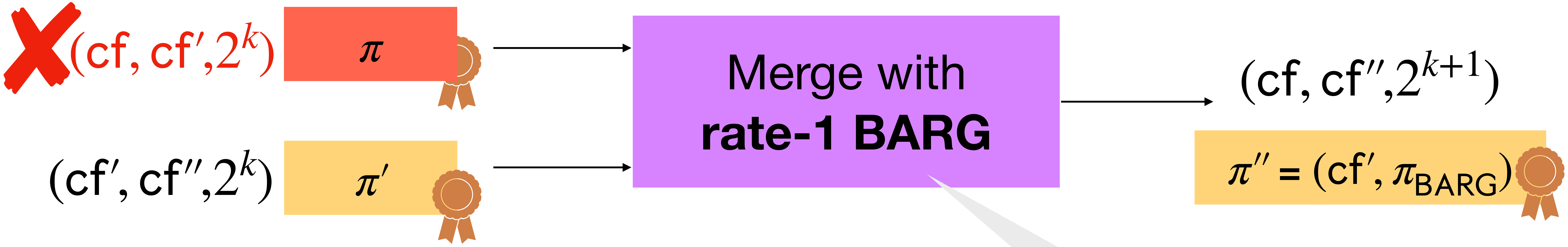
BARG Statement: $(\mathsf{cf}, \mathsf{cf}'), (\mathsf{cf}', \mathsf{cf}'')$.

BARG Witness: Level k proofs π, π''.

Level $k+1$ proof: $\pi'' = (\mathsf{cf}', \pi_{\mathsf{BARG}})$.

- Suppose $\mathscr{A}$ creates cheating proofs $(\mathsf{cf}', \pi_{\mathsf{BARG}})$ with **prob.** $\geq \epsilon$.
  - By **pigeonhole,** $\mathsf{cf} \nrightarrow \mathsf{cf}'$ with prob $\geq \epsilon/2$ or, $\mathsf{cf}' \nrightarrow \mathsf{cf}''$. WLOG first hop.
  - Switch BARG **CRS binding** accordingly!
    - By **index-hiding property** of the BARG, should still cheat on this hop!!

Proof Strategy: BARG the BARG



Construct a BARG proof π_{BARG} corresponding to:

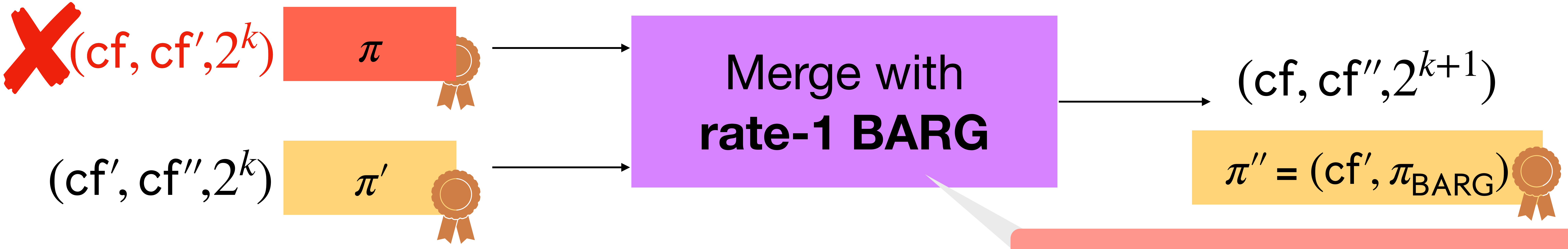
BARG Statement: (cf, cf′), (cf′, cf″).

BARG Witness: Level k proofs π, π''.

Level $k+1$ proof: $\pi'' = (\text{cf}', \pi_{\text{BARG}})$.

- Suppose $\mathscr{A}$ creates cheating proofs (cf′, π_{BARG}) with **prob.** $\geq \epsilon$.
  - By **pigeonhole,** cf $\nrightarrow$ cf′ with prob $\geq \epsilon/2$ or, cf′ $\nrightarrow$ cf″. WLOG first hop.
  - Switch BARG **CRS binding** accordingly!
    - By **index-hiding property** of the BARG, should still cheat on this hop!!
    - **Extract** π and recurse!!

Proof Strategy: BARG the BARG



Needs careful complexity leveraging for NP (this work, based on BKK+17)! Ask me later :)

- Suppose $\mathscr{A}$ creates cheating proofs $(\text{cf}', \pi_{\text{BARG}})$ with **prob.** $\geq \epsilon$.
  - By **pigeonhole,** $\text{cf} \nrightarrow \text{cf}'$ with prob $\geq \epsilon/2$ or $\text{cf}' \nrightarrow \text{cf}''$. WLOG first hop.
  - Switch BARG **CRS binding** accordingly!
    - By **index-hiding property** of the BARG, should still cheat on this hop!!
    - **Extract** π and recurse!!