Succinct Arguments: Constructions and Open Problems

Alessandro Chiesa
UC Berkeley

The Role of Cryptography

The **Power** of PCPs

The **Power** of PCPs

Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

The **Power** of PCPs

Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

The **Power** of PCPs

Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language $L \in NTIME(T)$ ∀ instance $x \in \{0,1\}^n$

a prover can (given a witness for x) produce in time $T \log(T)^{O(1)}$

The **Power** of PCPs

Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

a prover can (given a witness for x) produce in time T $\log(T)^{O(1)}$

a proof π for the statement "x∈L"

The **Power** of PCPs

Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language $L \in NTIME(T)$ ∀ instance $x \in \{0,1\}^n$

a prover can (given a witness for x) produce in time $T \log(T)^{O(1)}$

a proof π for the statement "$x \in L$"

that can be *probabilistically* checked in time $\log(T)^{O(1)}$

The **Power** of PCPs

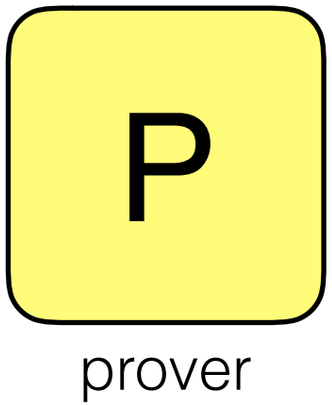
Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

a prover can (given a witness for x) produce in time T $\log(T)^{O(1)}$

a proof π for the statement "x∈L"

that can be *probabilistically* checked in time $\log(T)^{O(1)}$



The **Power** of PCPs

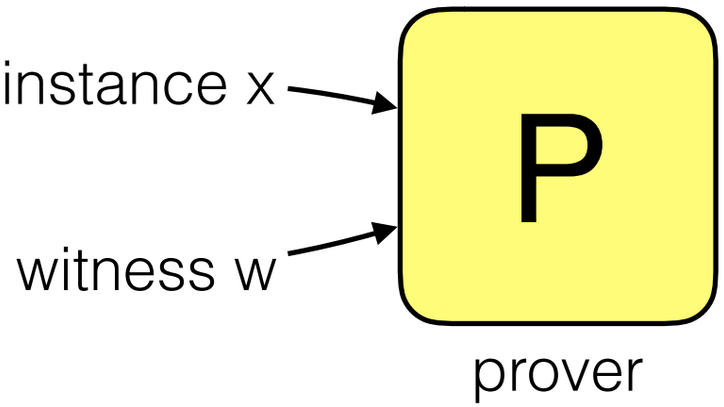
Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

a prover can (given a witness for x) produce in time T $\log(T)^{O(1)}$

a proof π for the statement "x∈L"

that can be *probabilistically* checked in time $\log(T)^{O(1)}$



The **Power** of PCPs

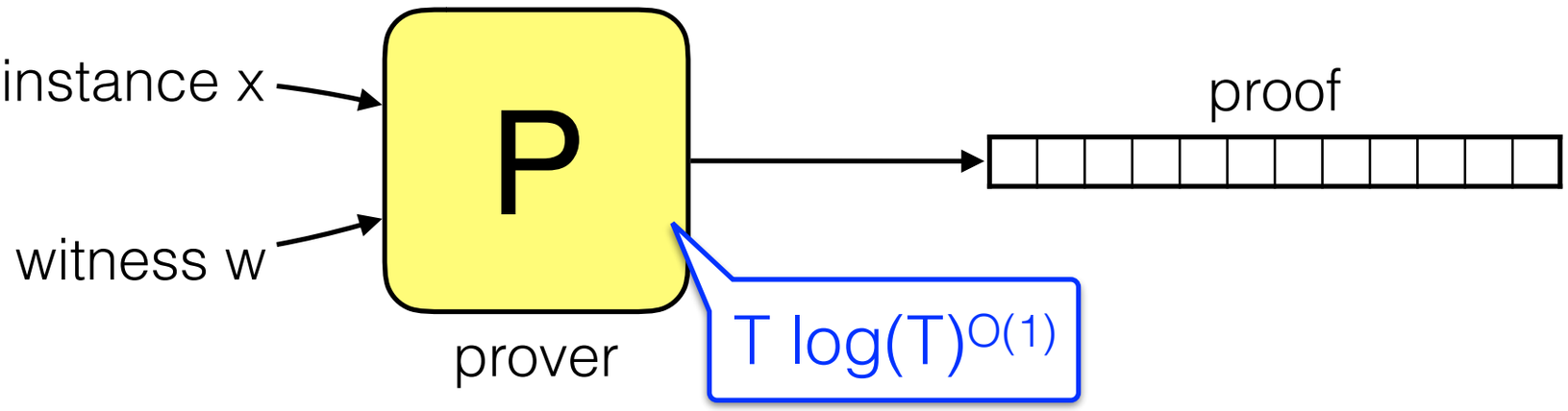
Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

a prover can (given a witness for x) produce in time T $\log(T)^{O(1)}$

a proof π for the statement "x∈L"

that can be *probabilistically* checked in time $\log(T)^{O(1)}$



The **Power** of PCPs

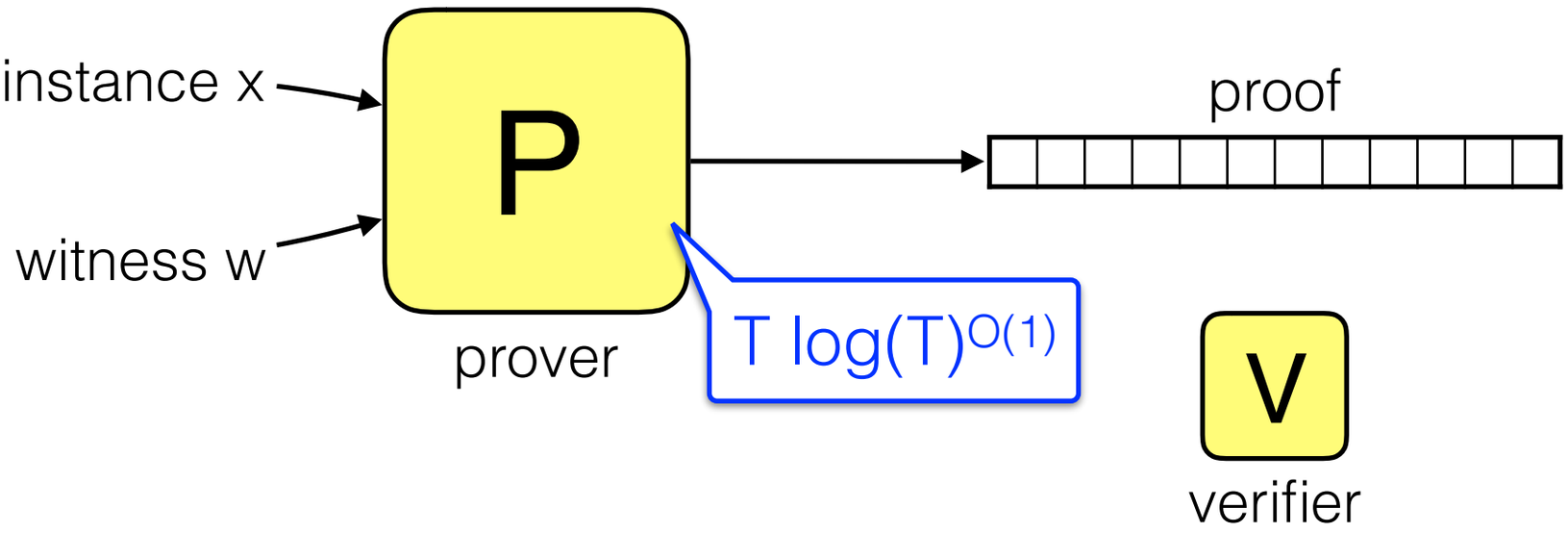
Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

a prover can (given a witness for x) produce in time T $\log(T)^{O(1)}$

a proof π for the statement "x∈L"

that can be *probabilistically* checked in time $\log(T)^{O(1)}$



The **Power** of PCPs

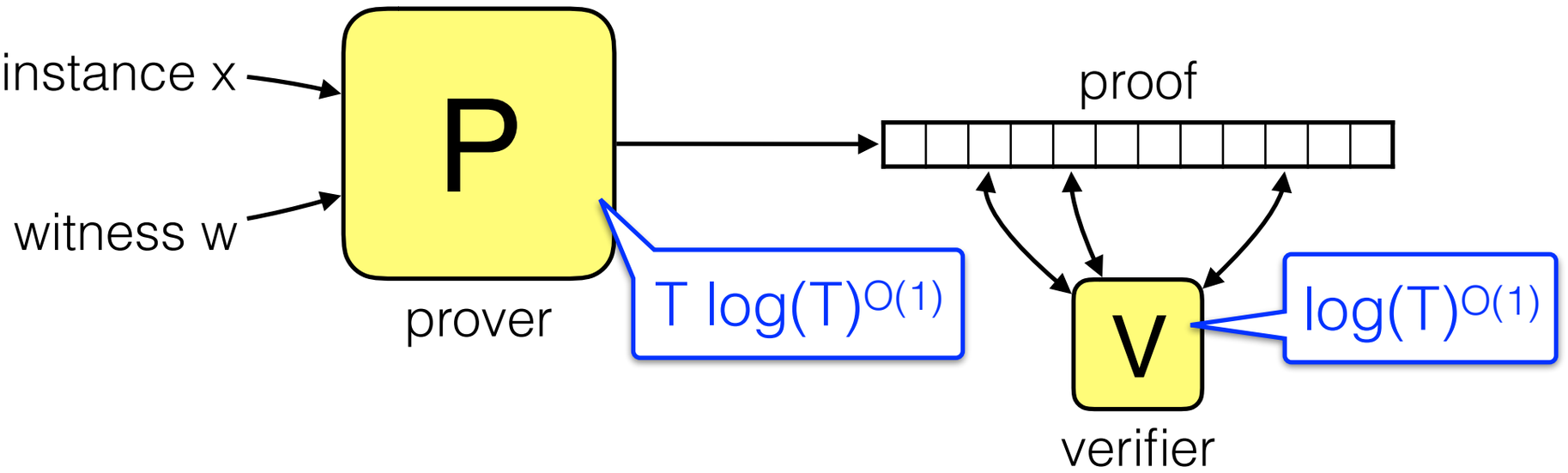
Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

a prover can (given a witness for x) produce in time T $\log(T)^{O(1)}$

a proof π for the statement "x∈L"

that can be *probabilistically* checked in time $\log(T)^{O(1)}$



The **Power** of PCPs

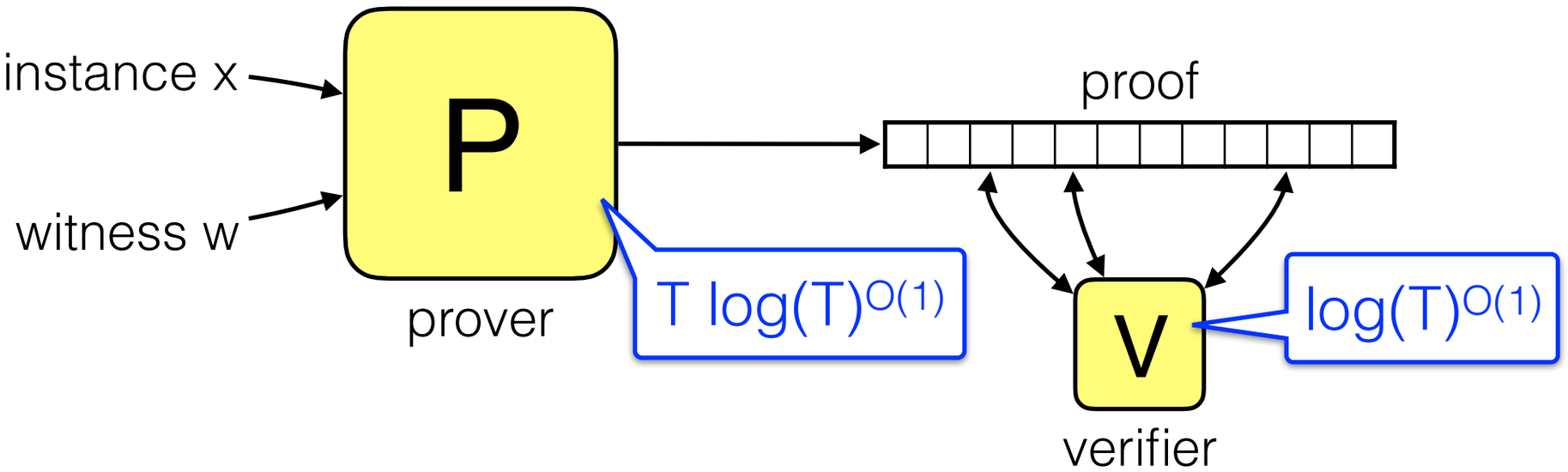
Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

a prover can (given a witness for x) produce in time T $\log(T)^{O(1)}$

a proof π for the statement "x∈L"

that can be *probabilistically* checked in time $\log(T)^{O(1)}$



Fundamental observation in BFLS91

The **Power** of PCPs

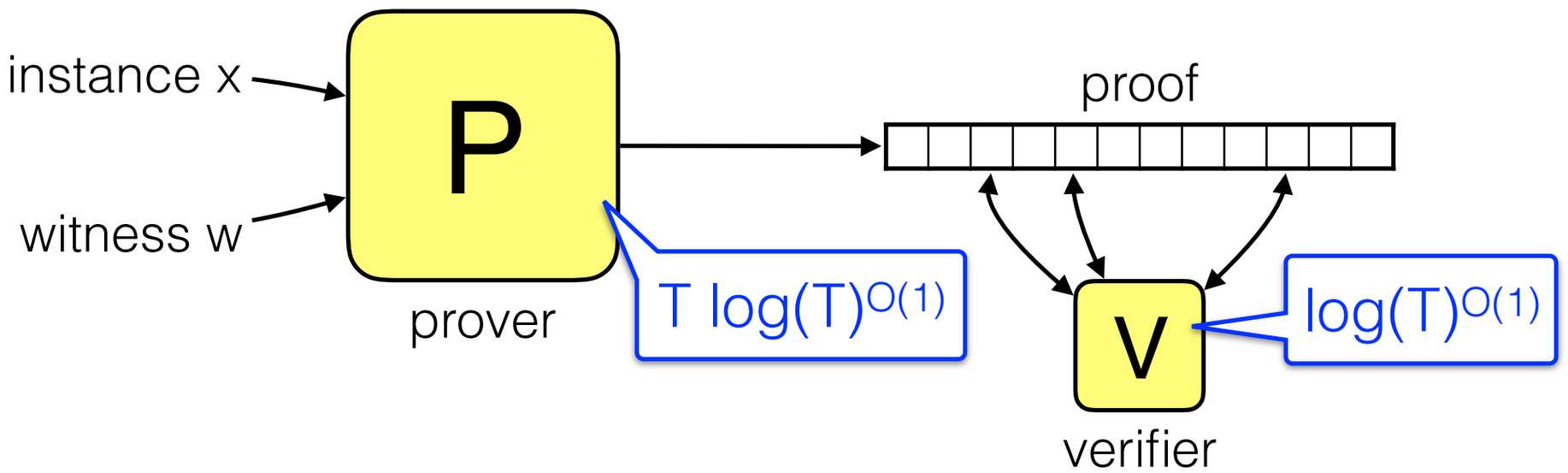
Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

a prover can (given a witness for x) produce in time T $\log(T)^{O(1)}$

a proof π for the statement "x∈L"

that can be *probabilistically* checked in time $\log(T)^{O(1)}$



Fundamental observation in BFLS91

In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware.

The **Limitation** of PCPs

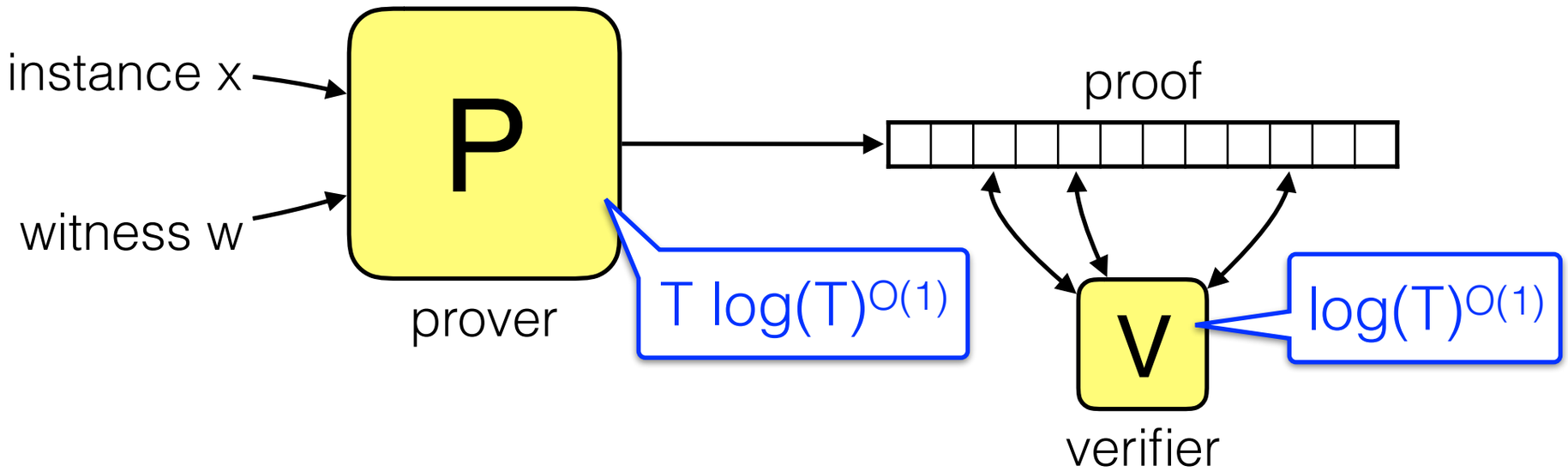
Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

a prover can (given a witness for x) produce in time T $\log(T)^{O(1)}$

a proof π for the statement "x∈L"

that can be *probabilistically* checked in time $\log(T)^{O(1)}$



Fundamental observation in BFLS91

In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware.

The **Limitation** of PCPs

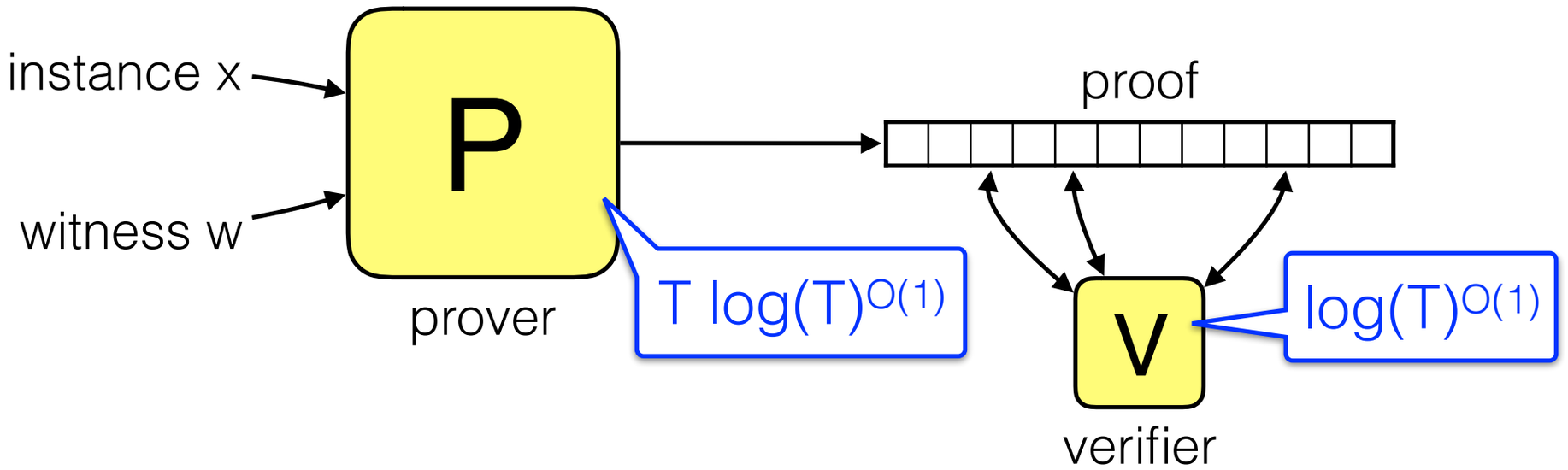
Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

a prover can (given a witness for x) produce in time T $\log(T)^{O(1)}$

a proof π for the statement "x∈L"

that can be *probabilistically* checked in time $\log(T)^{O(1)}$



Fundamental observation in BFLS91

In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware.

The **Limitation** of PCPs

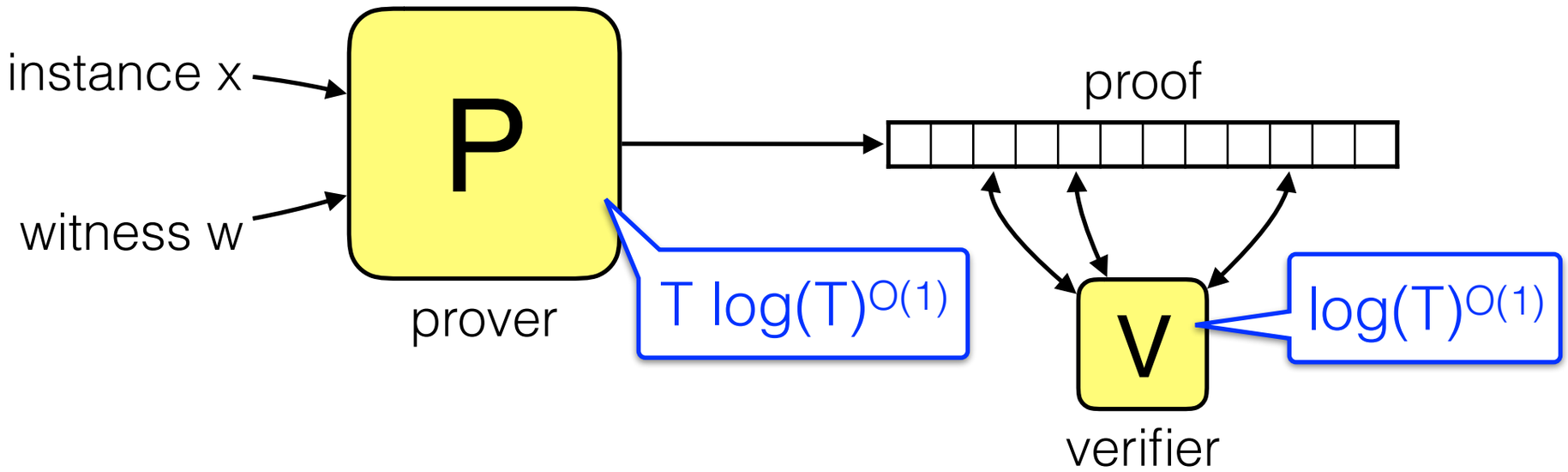
Theorem [BFLS91, FGLSS91, AS98, ALMSS98, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

a prover can (given a witness for x) produce in time T $\log(T)^{O(1)}$

a proof π for the statement "x∈L"

that can be *probabilistically* checked in time $\log(T)^{O(1)}$



Fundamental observation in BFLS91

In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware.

When is such a setup acceptable?

What about other models?

What about other models?

Theorem [BFL90, …]

What about other models?

Theorem [BFL90, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

What about other models?

Theorem [BFL90, …]

∀ language $L \in NTIME(T)$ ∀ instance $x \in \{0,1\}^n$

there is a 2-prover 1-round interactive proof for the statement "$x \in L$"

What about other models?

Theorem [BFL90, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

there is a 2-prover 1-round interactive proof for the statement "x∈L"

where each prover runs in time T $\log(T)^{O(1)}$

What about other models?

Theorem [BFL90, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

there is a 2-prover 1-round interactive proof for the statement "x∈L"

where each prover runs in time T $\log(T)^{O(1)}$

and the verifier runs in time $\log(T)^{O(1)}$

What about other models?

Theorem [BFL90, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

there is a 2-prover 1-round interactive proof for the statement "x∈L"

where each prover runs in time T $\log(T)^{O(1)}$

and the verifier runs in time $\log(T)^{O(1)}$

verifier

What about other models?

Theorem [BFL90, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

there is a 2-prover 1-round interactive proof for the statement "x∈L"

where each prover runs in time T $\log(T)^{O(1)}$

and the verifier runs in time $\log(T)^{O(1)}$

What about other models?

Theorem [BFL90, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

there is a 2-prover 1-round interactive proof for the statement "x∈L"

where each prover runs in time T $\log(T)^{O(1)}$

and the verifier runs in time $\log(T)^{O(1)}$

What about other models?

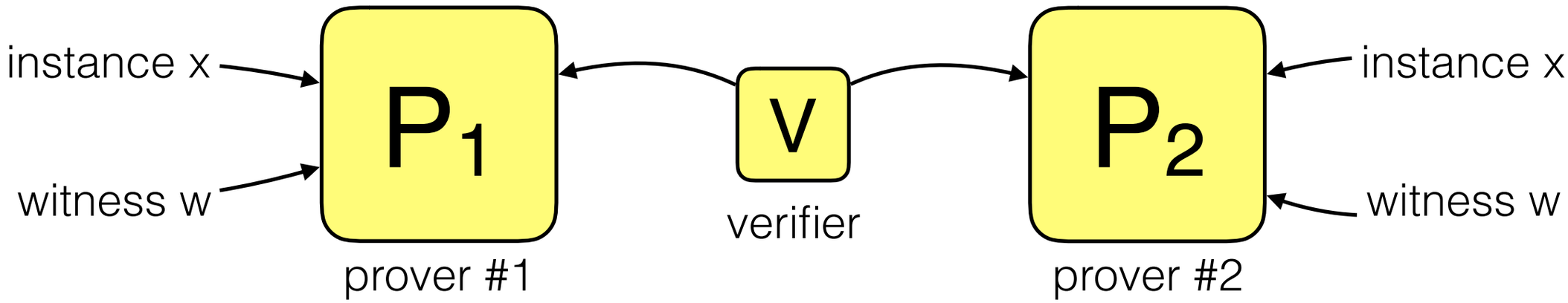
Theorem [BFL90, …]

∀ language L∈NTIME(T) ∀ instance x∈$\{0,1\}^n$

there is a 2-prover 1-round interactive proof for the statement "x∈L"

where each prover runs in time T $\log(T)^{O(1)}$

and the verifier runs in time $\log(T)^{O(1)}$



What about other models?

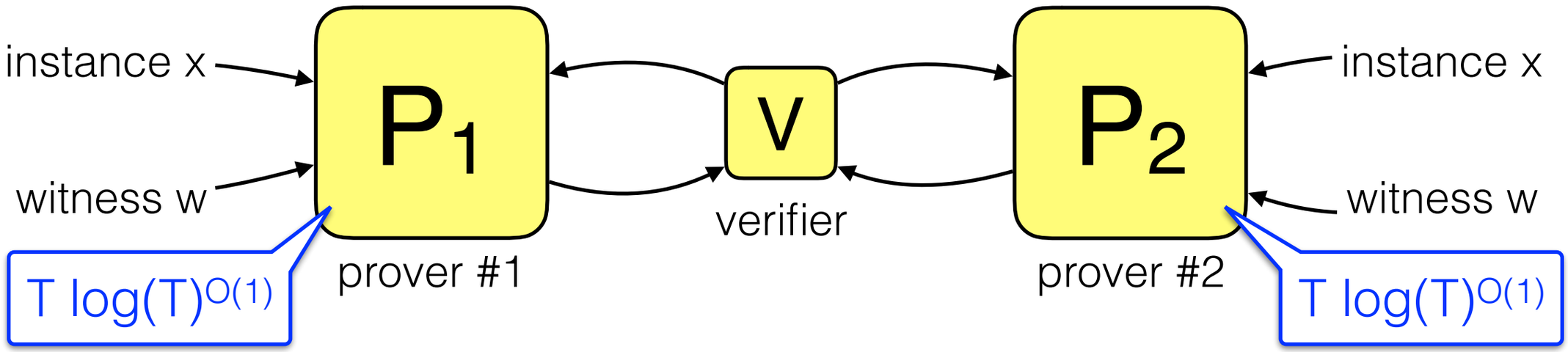
Theorem [BFL90, …]

∀ language L∈NTIME(T) ∀ instance x∈{0,1}^{n}

there is a 2-prover 1-round interactive proof for the statement "x∈L"

where each prover runs in time T log(T)$^{O(1)}$

and the verifier runs in time log(T)$^{O(1)}$



What about other models?

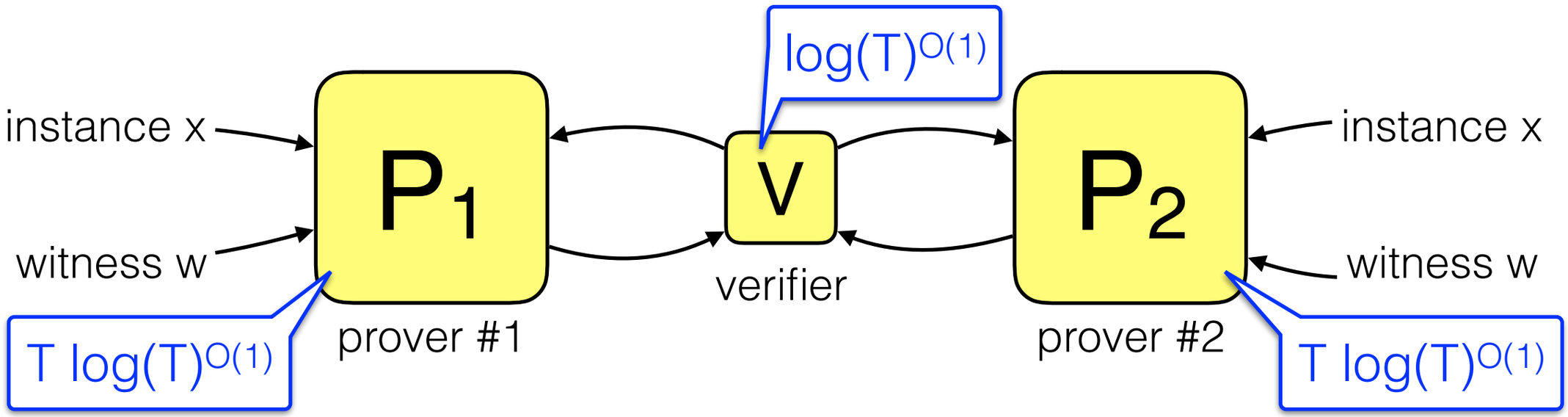
Theorem [BFL90, …]

∀ language L∈NTIME(T) ∀ instance x∈{0,1}^{n}

there is a 2-prover 1-round interactive proof for the statement "x∈L"

where each prover runs in time T log(T)$^{O(1)}$

and the verifier runs in time log(T)$^{O(1)}$



What about other models?

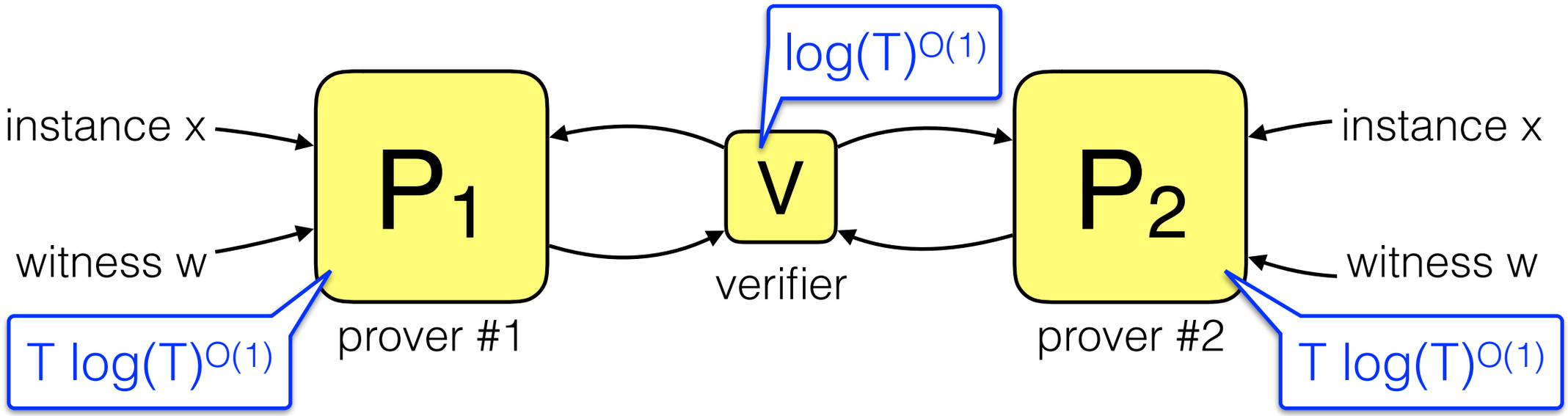
Theorem [BFL90, …]

∀ language L∈NTIME(T) ∀ instance x∈{0,1}^{n}

there is a 2-prover 1-round interactive proof for the statement "x∈L"

where each prover runs in time T log(T)$^{O(1)}$

and the verifier runs in time log(T)$^{O(1)}$



Same observation holds

In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware.

What about other models?

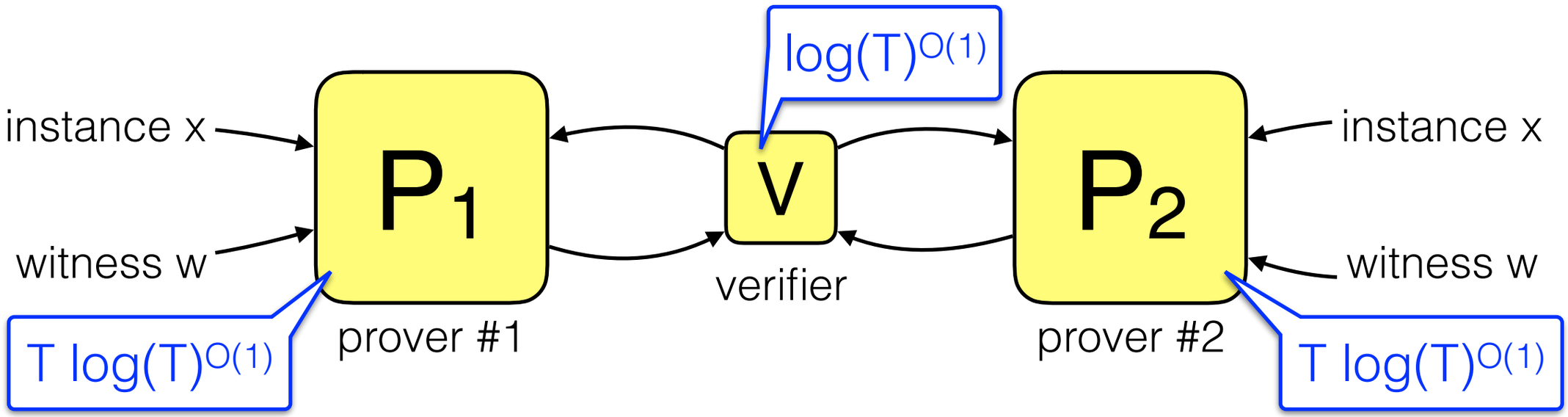
Theorem [BFL90, …]

∀ language L∈NTIME(T) ∀ instance x∈{0,1}^{n}

there is a 2-prover 1-round interactive proof for the statement "x∈L"

where each prover runs in time T log(T)$^{O(1)}$

and the verifier runs in time log(T)$^{O(1)}$



Same observation holds

In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware.

What about other models?

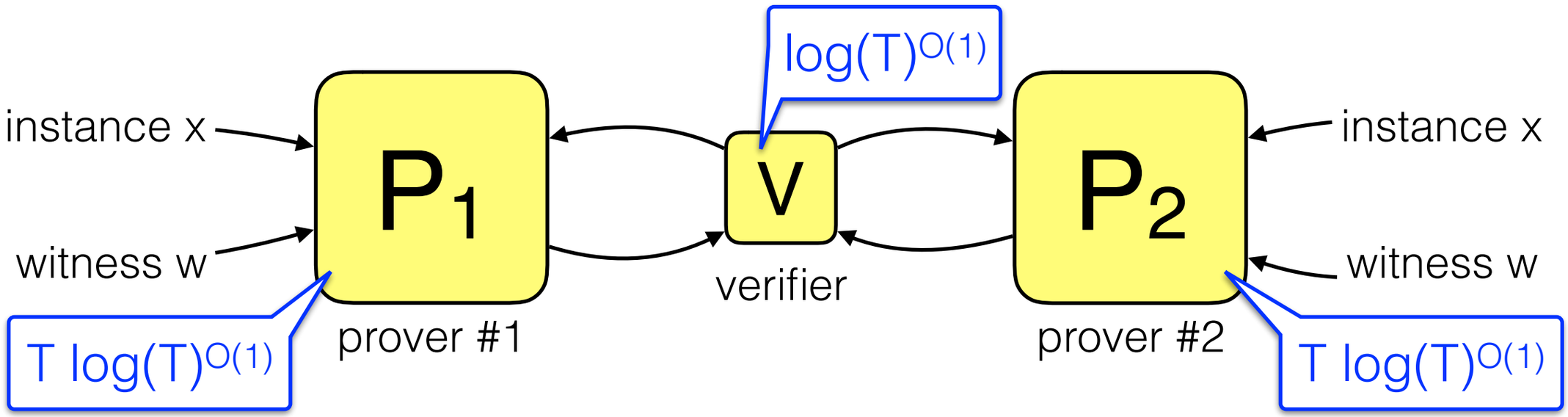
Theorem [BFL90, …]

∀ language L∈NTIME(T) ∀ instance x∈{0,1}^{n}

there is a 2-prover 1-round interactive proof for the statement "x∈L"

where each prover runs in time T log(T)$^{O(1)}$

and the verifier runs in time log(T)$^{O(1)}$



Same observation holds

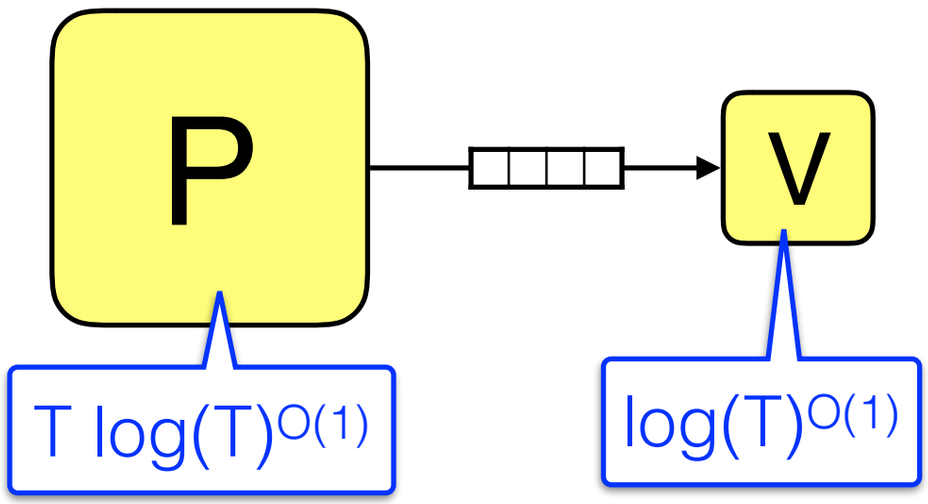
In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware.

When is such a setup acceptable?

What about a “Plain” Model?

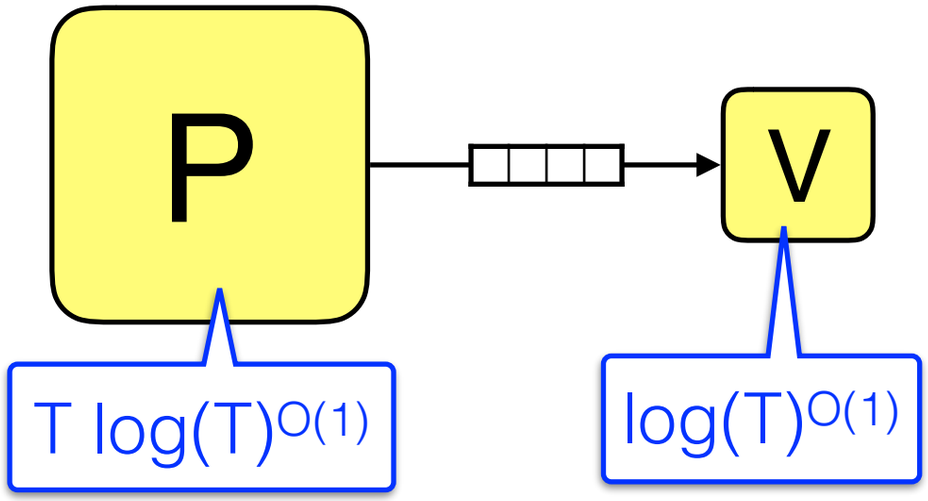
What about a “Plain” Model?

Non-Interactive Proof

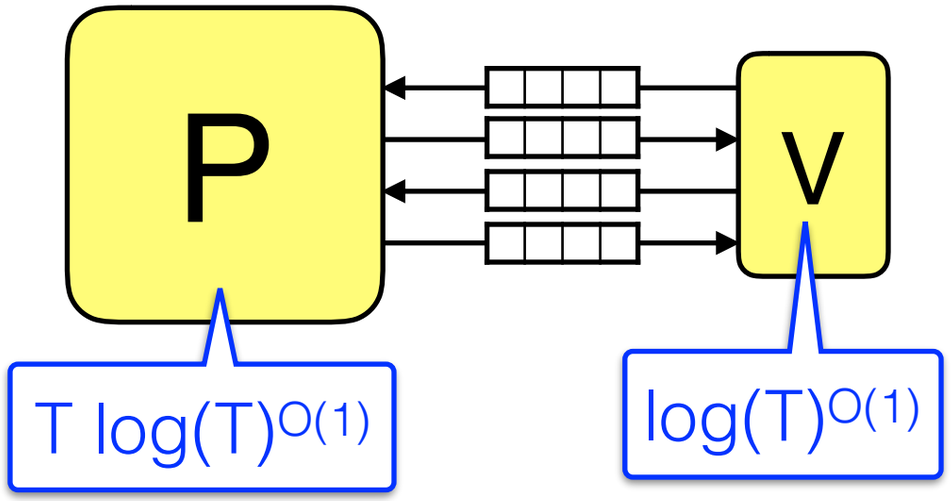


What about a “Plain” Model?

Non-Interactive Proof

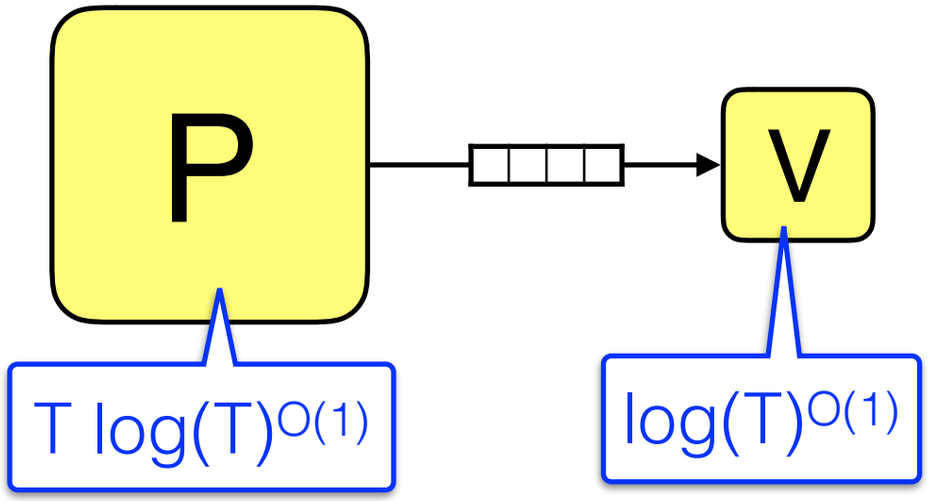


Interactive Proof

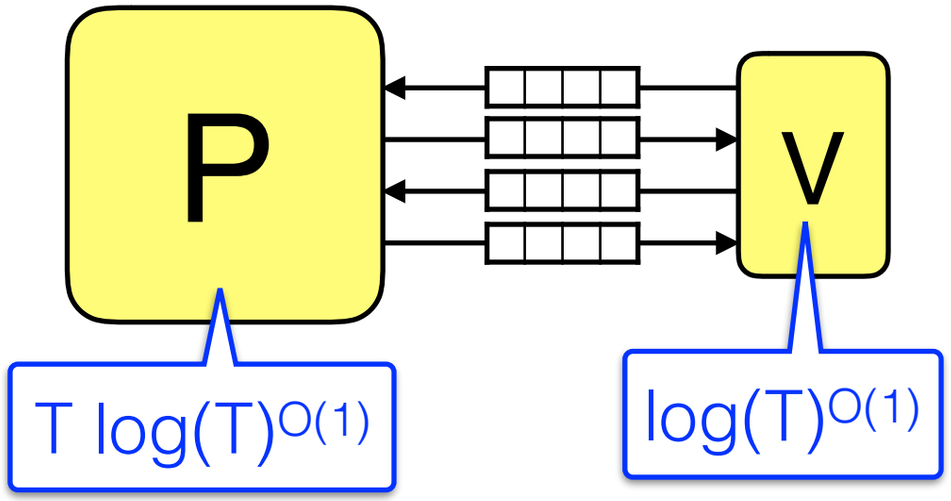


What about a “Plain” Model?

Non-Interactive Proof



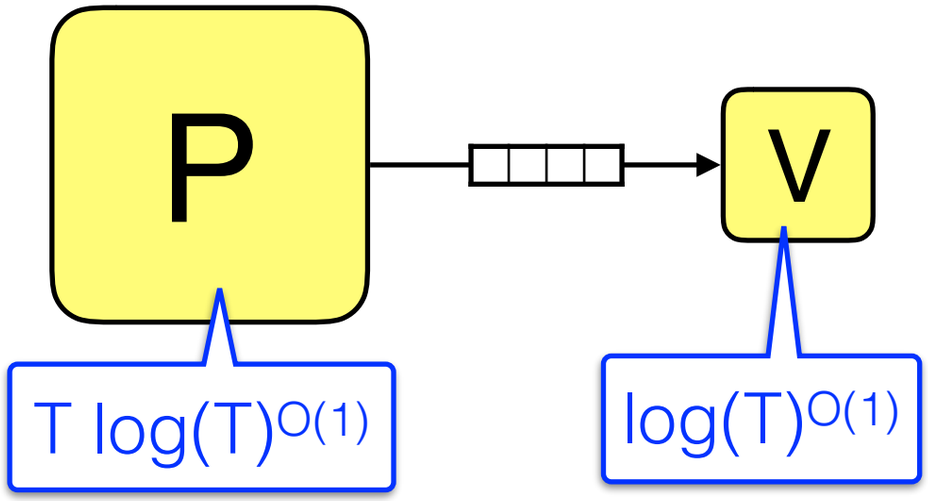
Interactive Proof



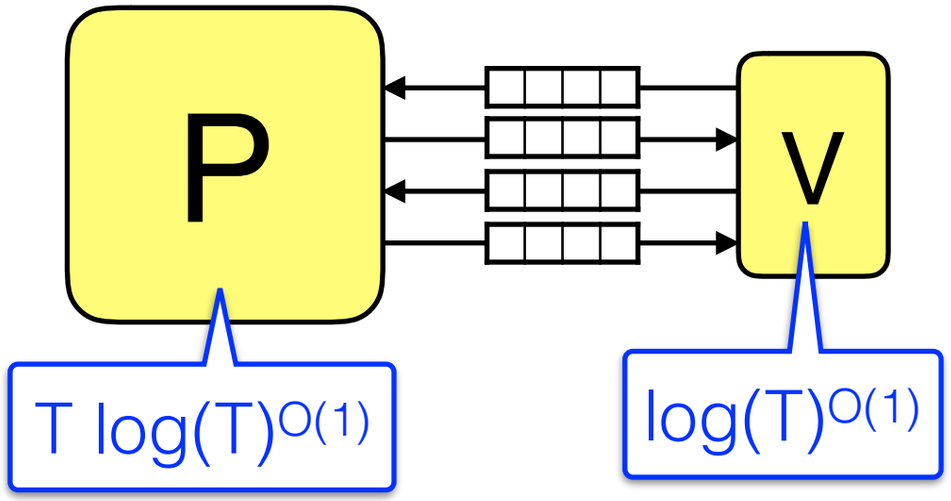
Bad News: cannot expect *succinct* proofs for hard languages (e.g., **NP**)

What about a “Plain” Model?

Non-Interactive Proof



Interactive Proof



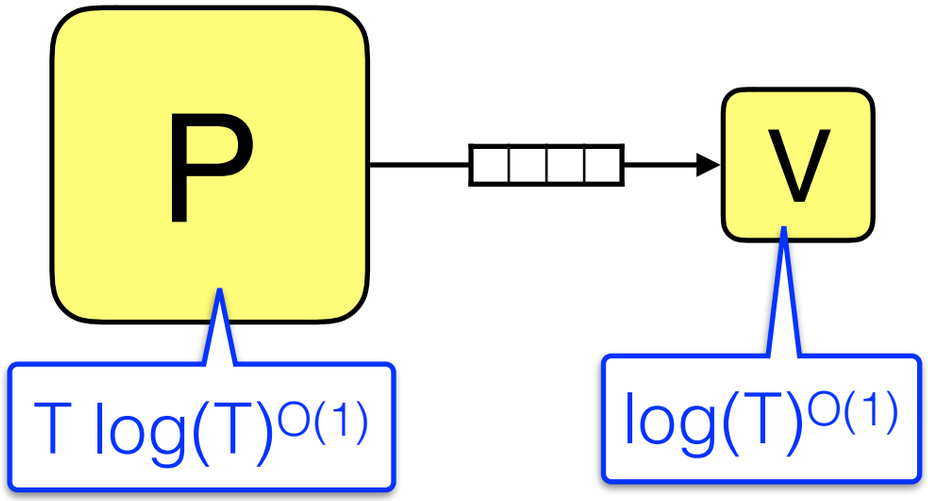
Bad News: cannot expect *succinct* proofs for hard languages (e.g., **NP**)

Theorem [GH98]

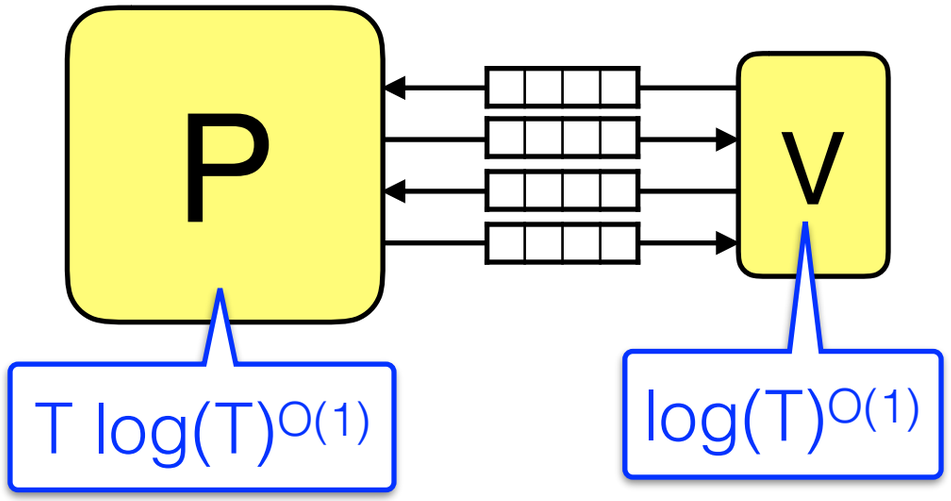
any Interactive Proof with “small” communication is “easy” to decide

What about a “Plain” Model?

Non-Interactive Proof



Interactive Proof



Bad News: cannot expect *succinct* proofs for hard languages (e.g., **NP**)

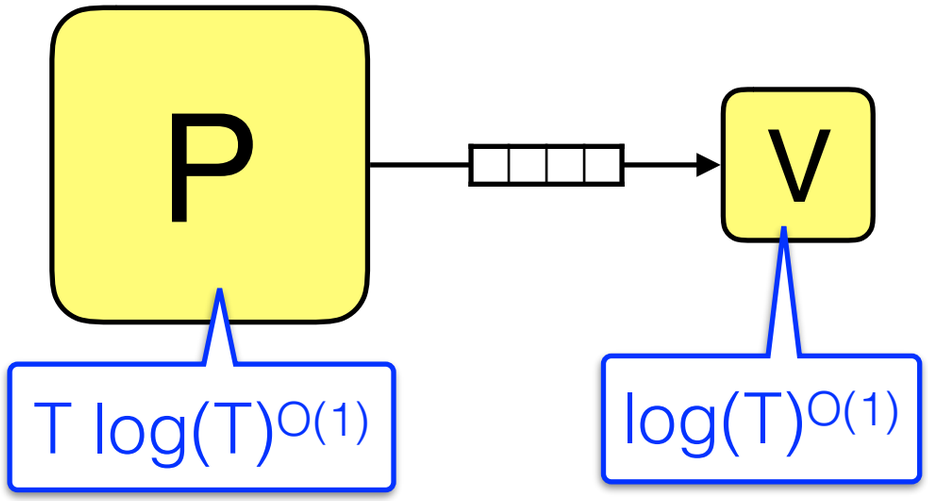
Theorem [GH98]

any Interactive Proof with “small” communication is “easy” to decide

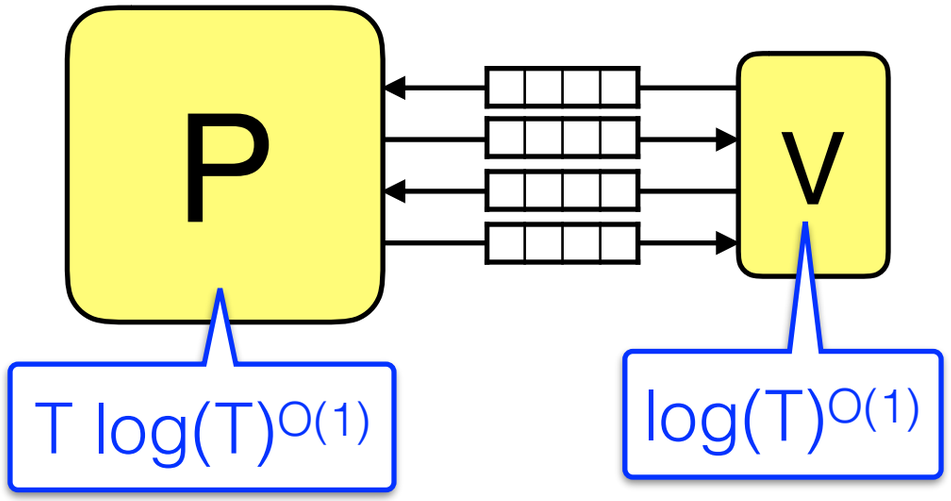
$\leq c$ $\in$ **BPTime**$(2^{O(c)})$

What about a “Plain” Model?

Non-Interactive Proof



Interactive Proof



Bad News: cannot expect *succinct* proofs for hard languages (e.g., **NP**)

Theorem [GH98]

any Interactive Proof with “small” communication is “easy” to decide

$\leq c$ $\in$ **BPTime**$(2^{O(c)})$

(Note: does NOT rule out doubly-efficient interactive proofs!)

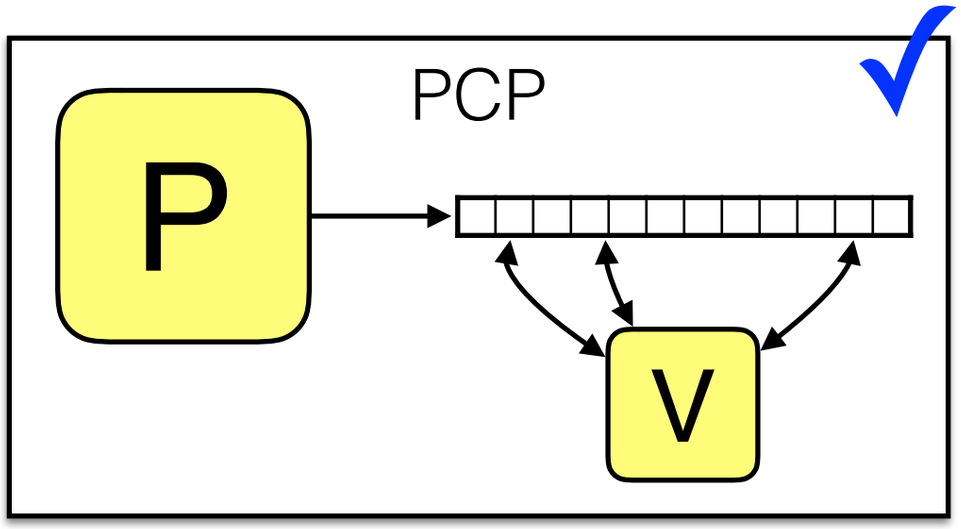
An Awkward Situation

An Awkward Situation

Inconvenient Succinct Proofs

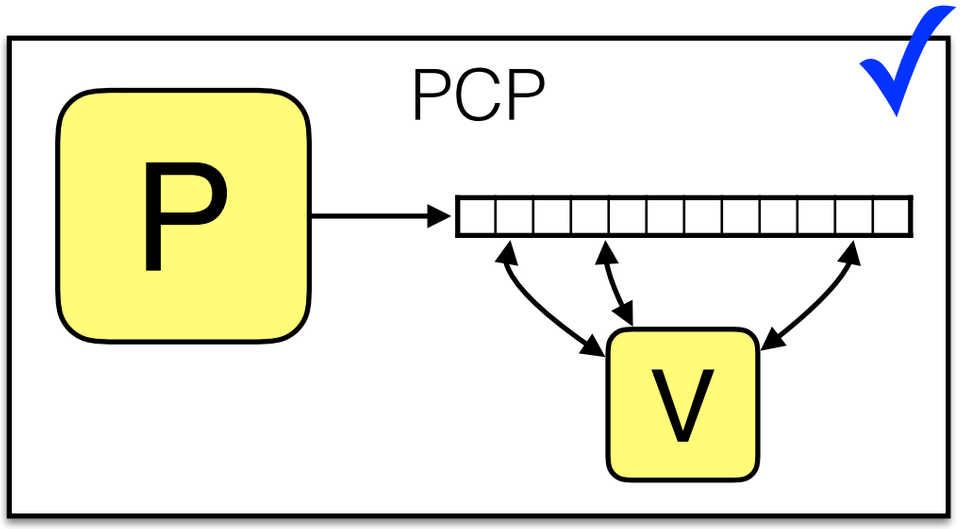
An Awkward Situation

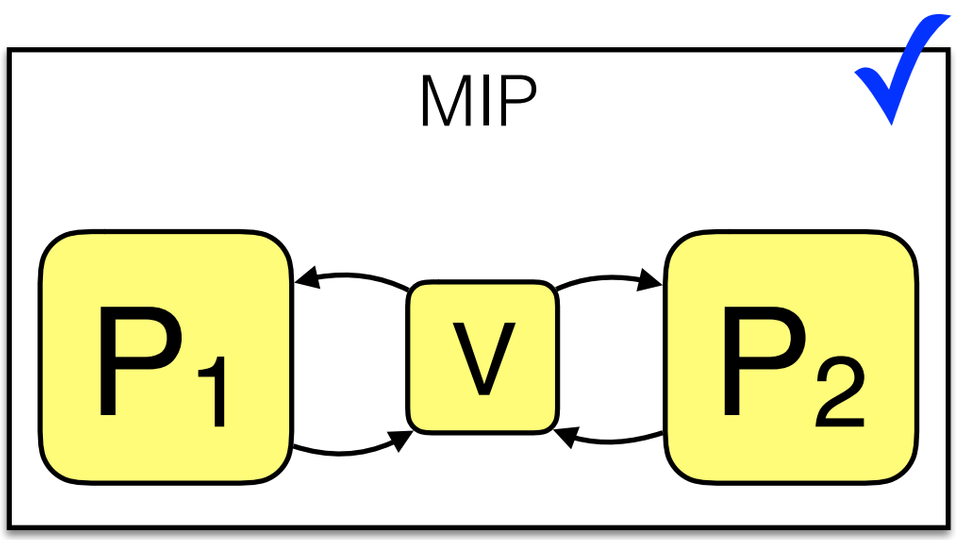
Inconvenient Succinct Proofs



An Awkward Situation

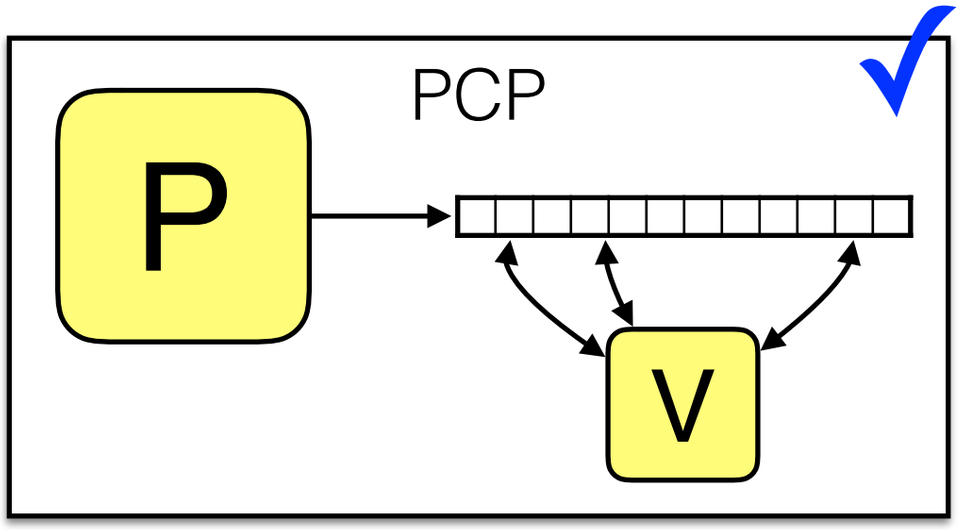
Inconvenient Succinct Proofs

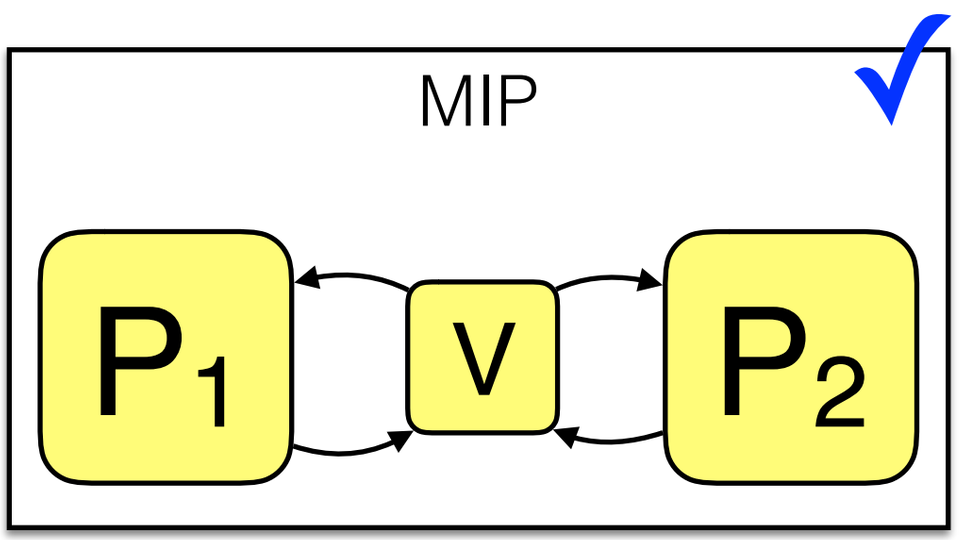


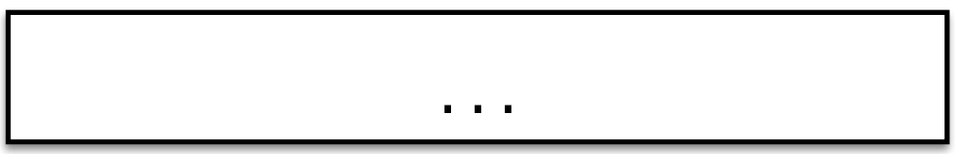


An Awkward Situation

Inconvenient Succinct Proofs

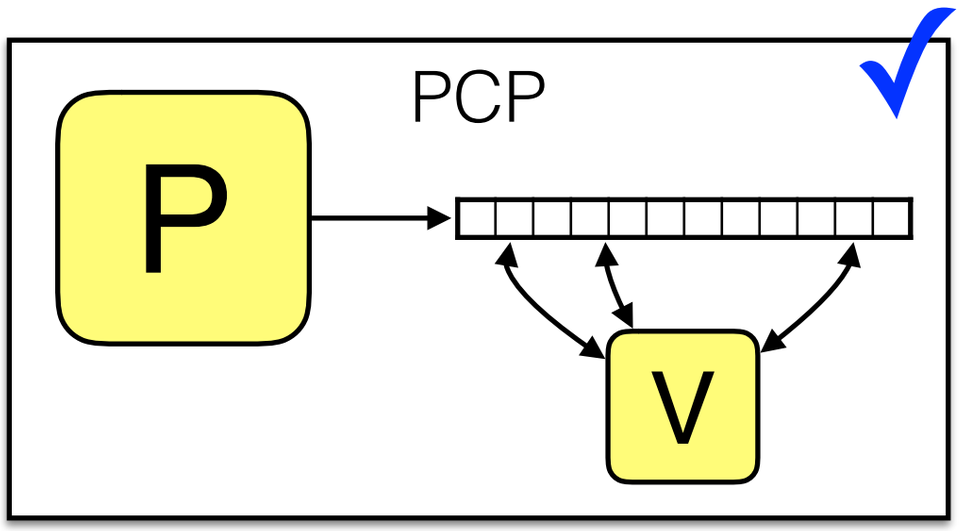


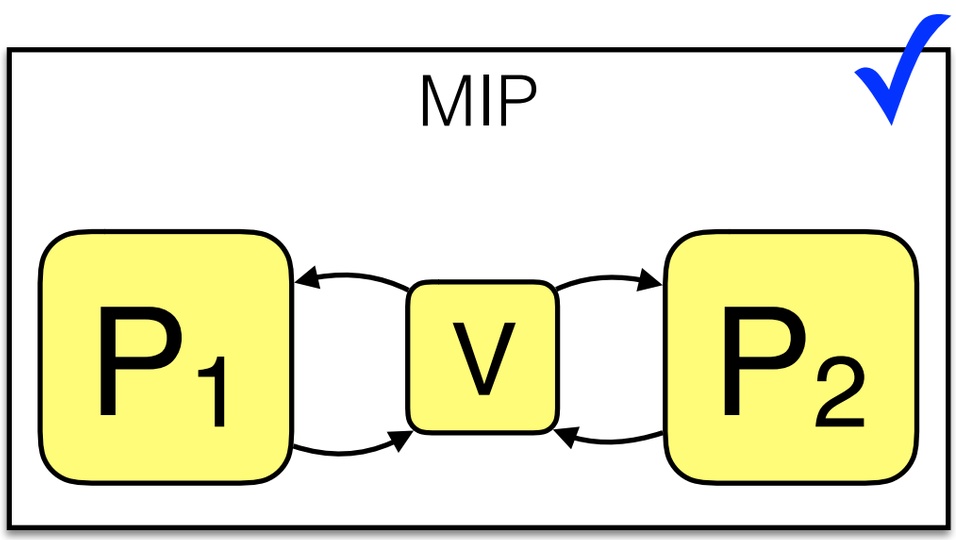


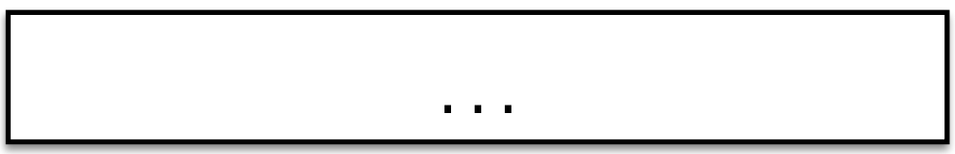


An Awkward Situation

Inconvenient Succinct Proofs



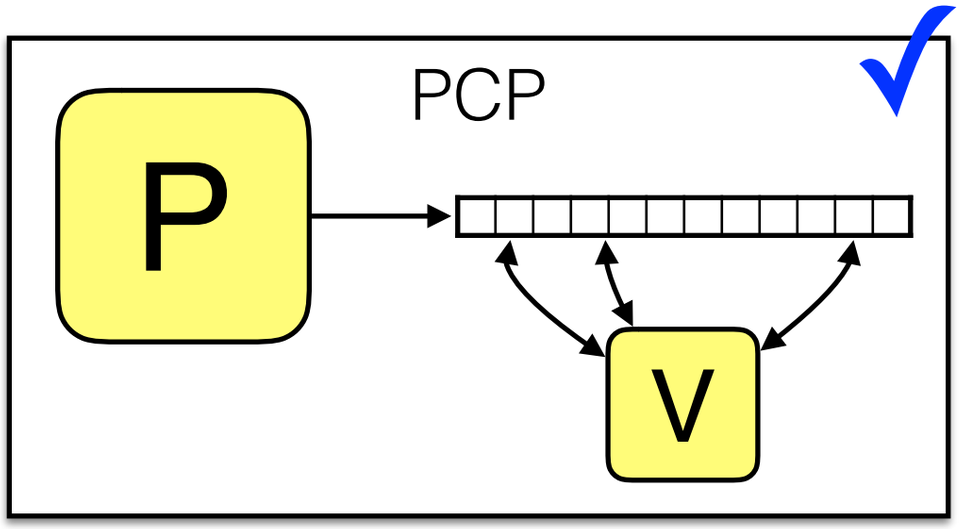


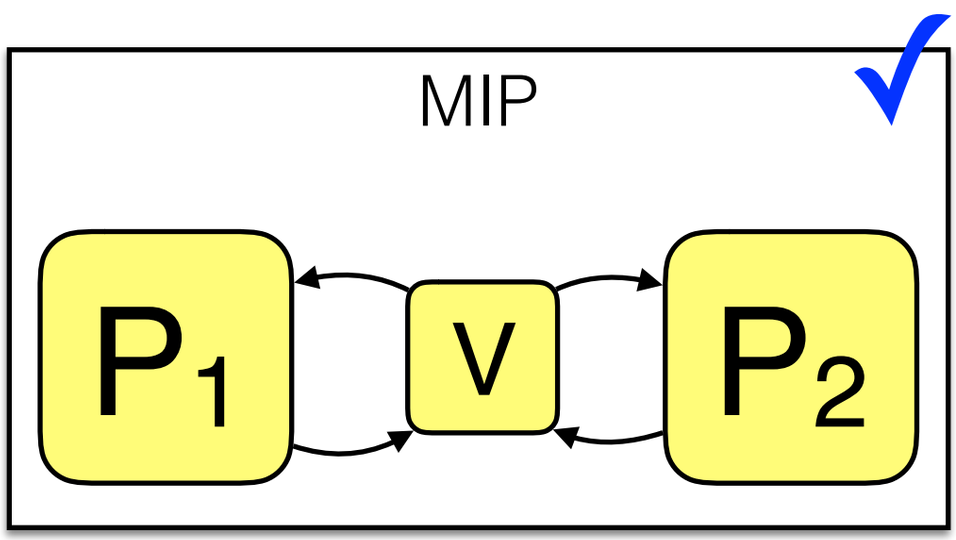


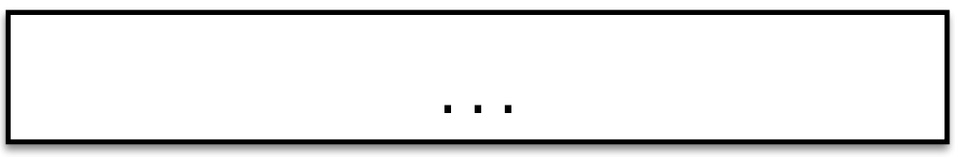
Convenient Succinct Proofs

An Awkward Situation

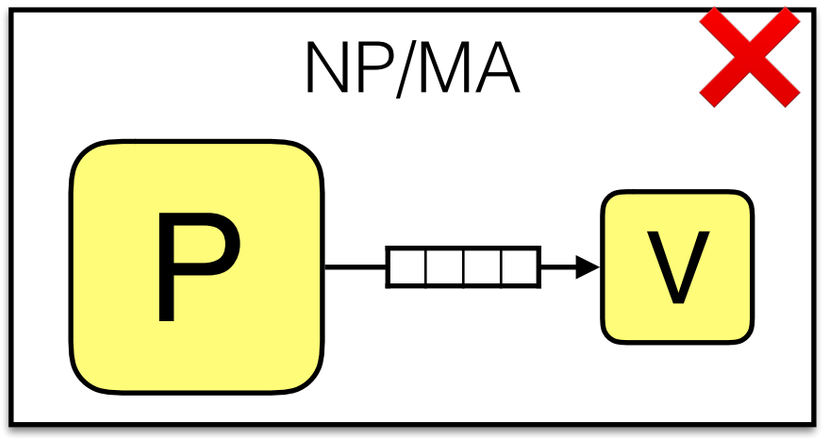
Inconvenient Succinct Proofs





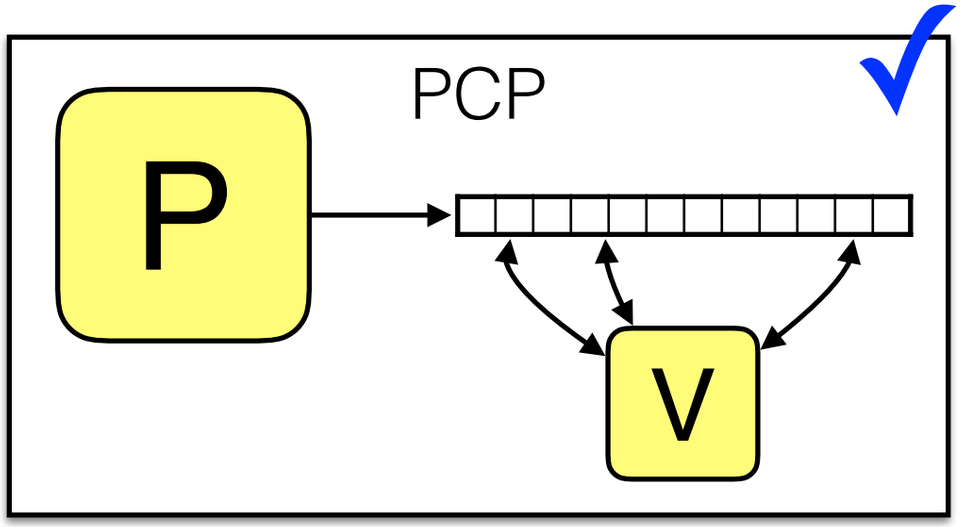


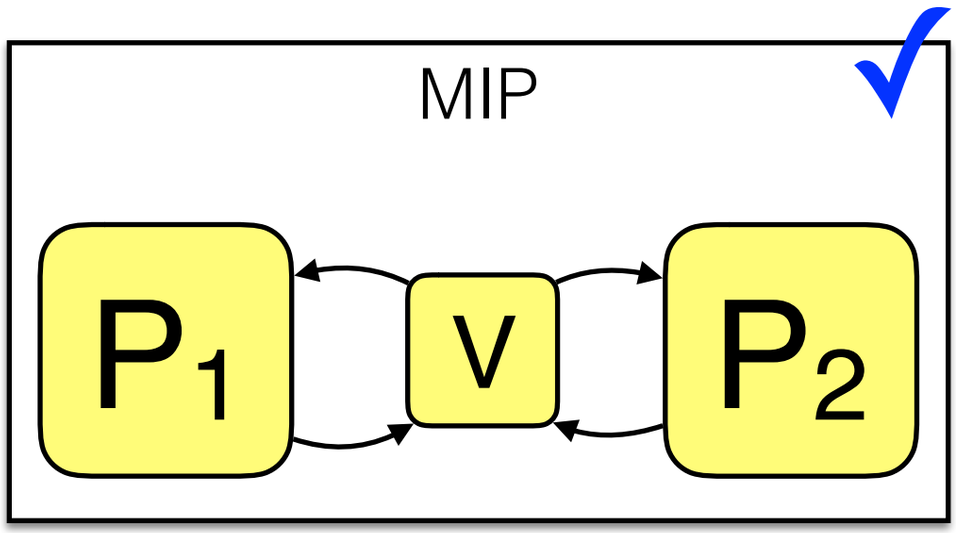
Convenient Succinct Proofs

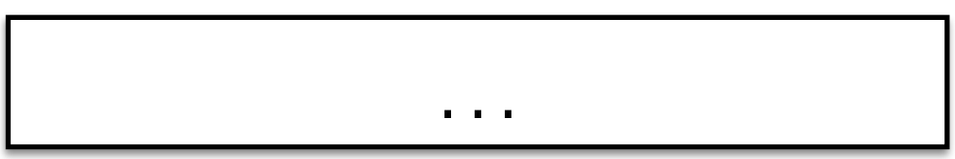


An Awkward Situation

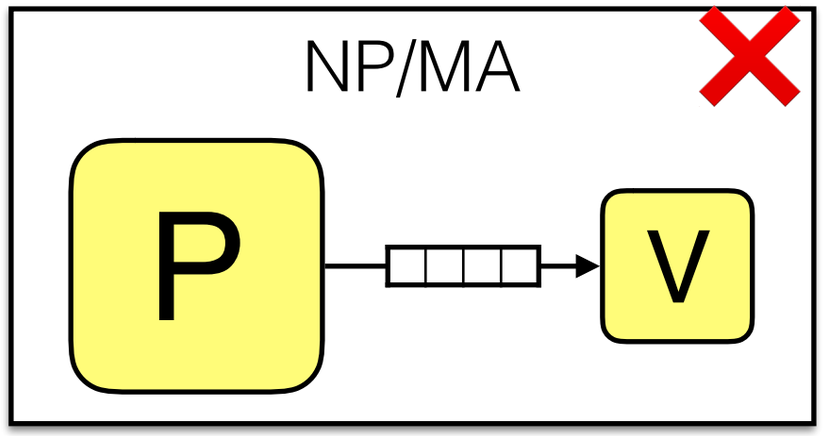
Inconvenient Succinct Proofs

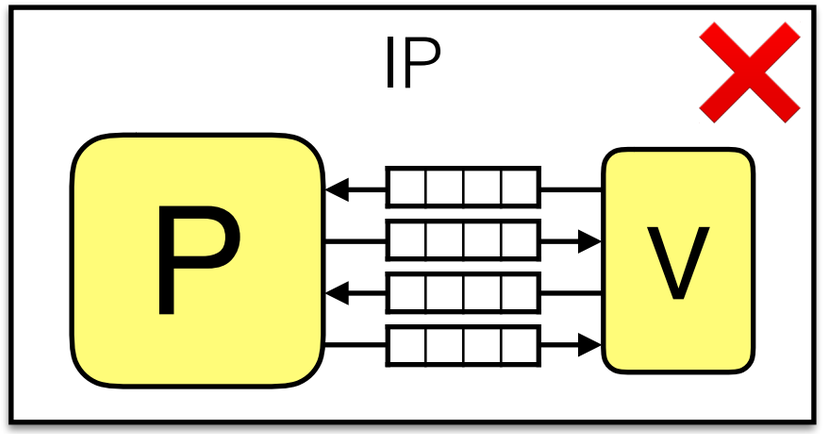






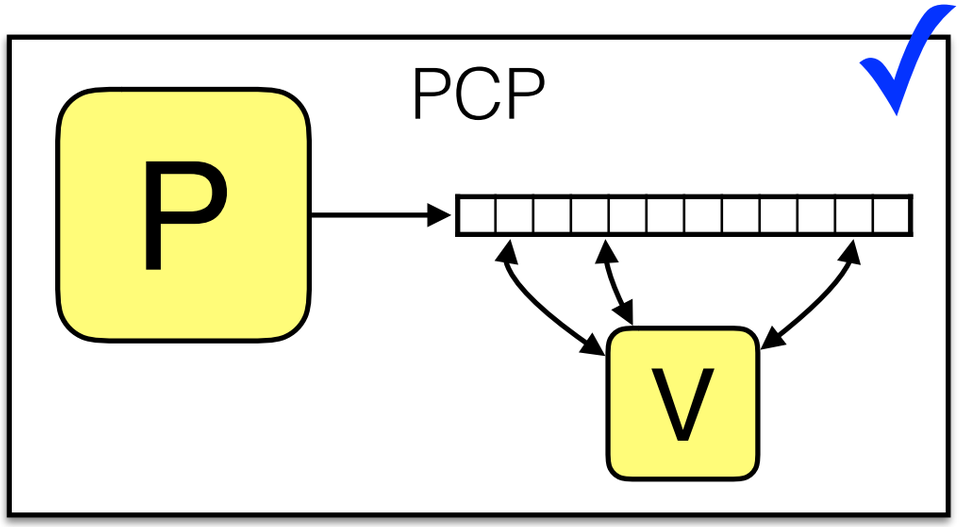
Convenient Succinct Proofs

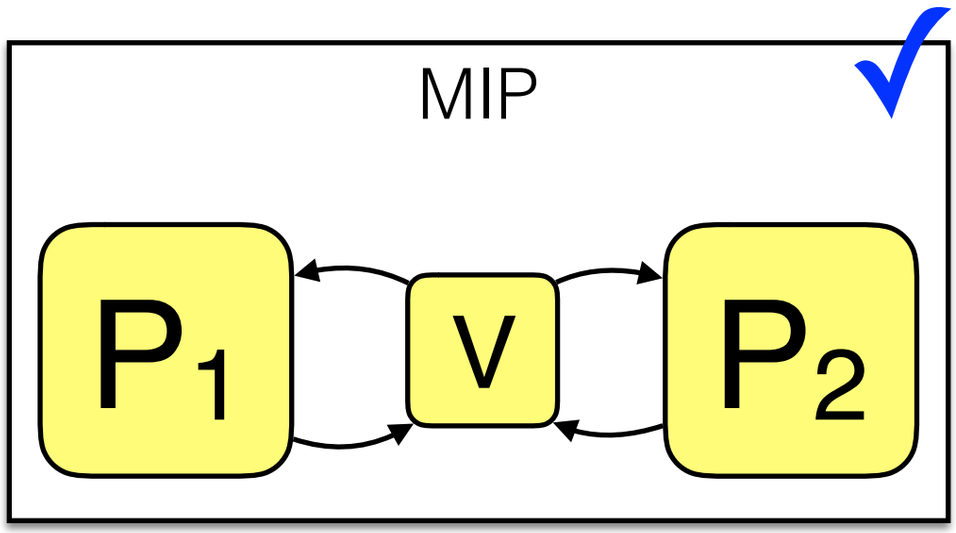




Cryptography As A Bridge

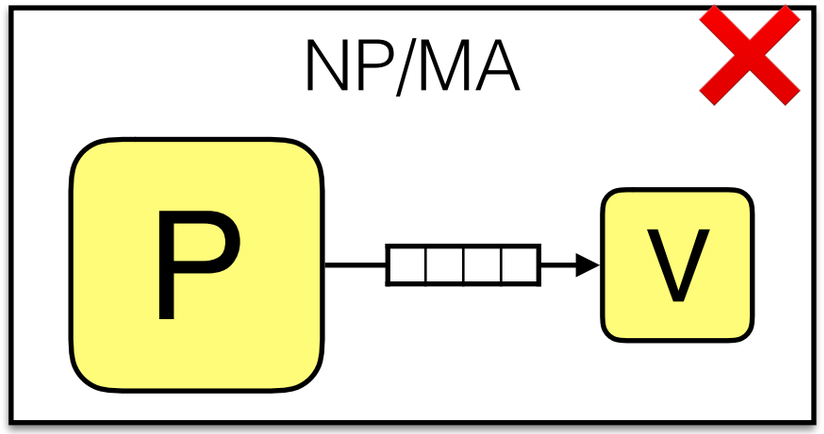
Inconvenient Succinct Proofs

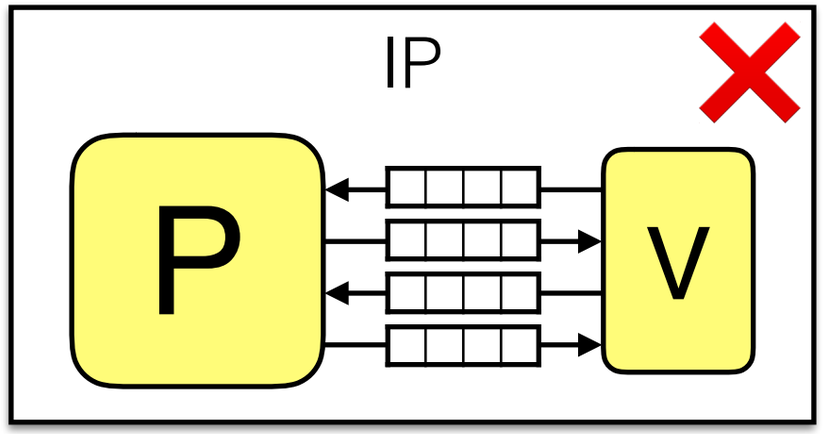




...

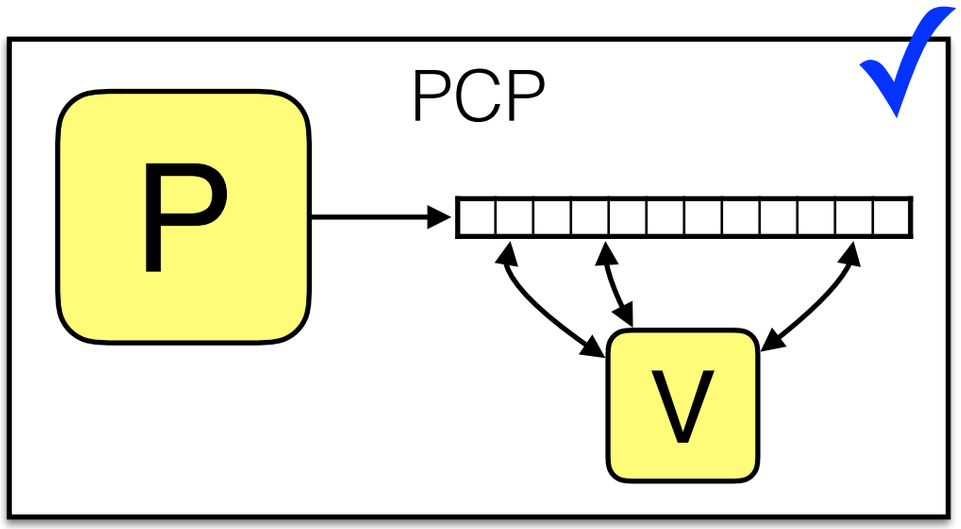
Convenient Succinct Proofs

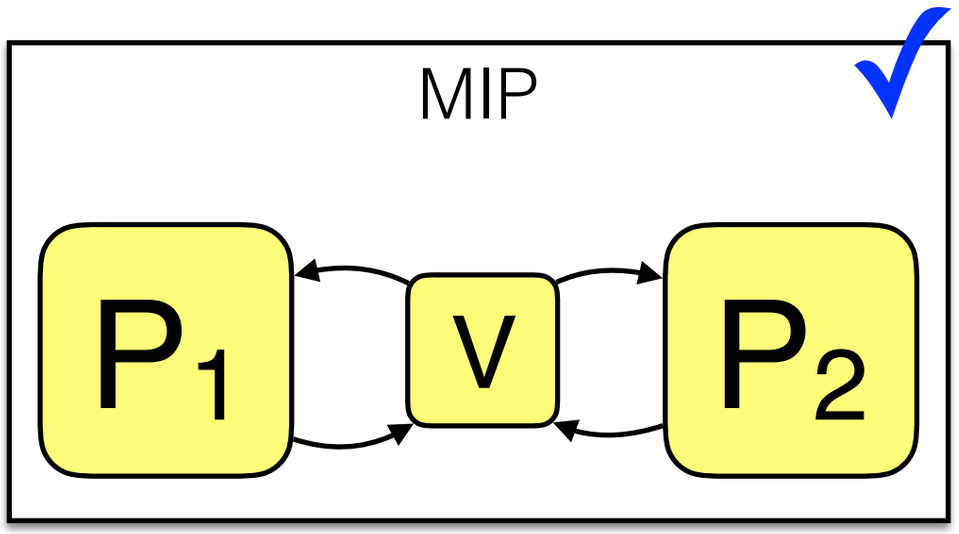


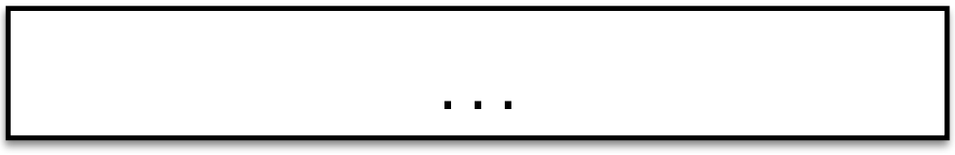


Cryptography As A Bridge

Inconvenient Succinct Proofs

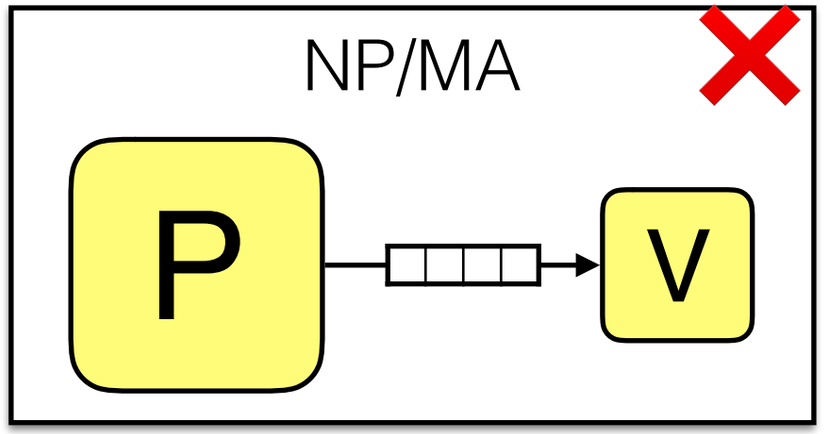


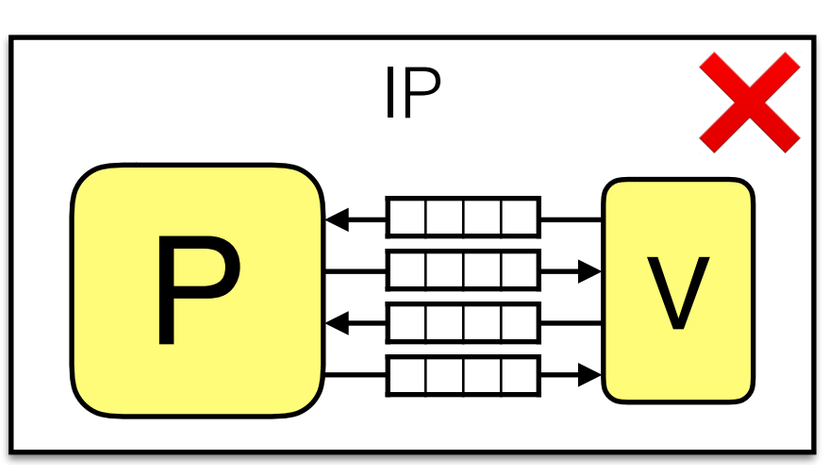




CRYPTO

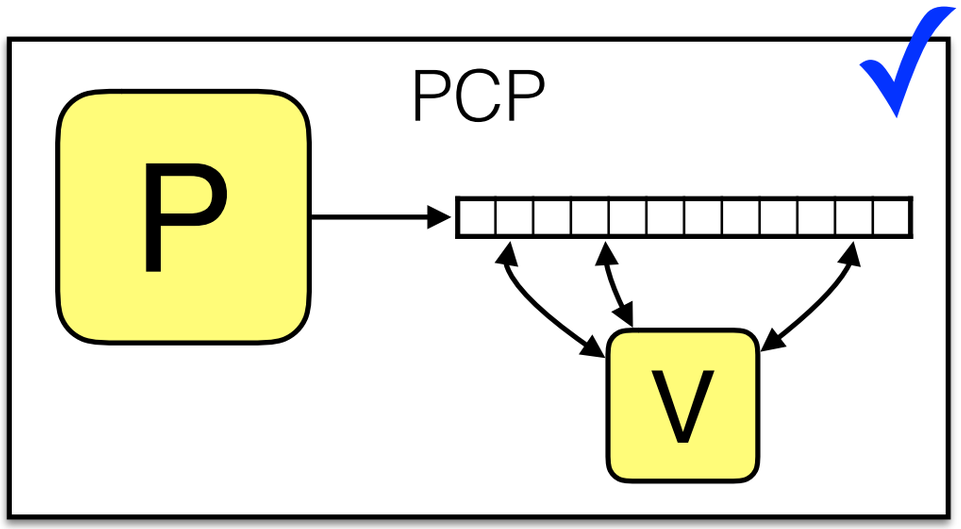
Convenient Succinct Proofs

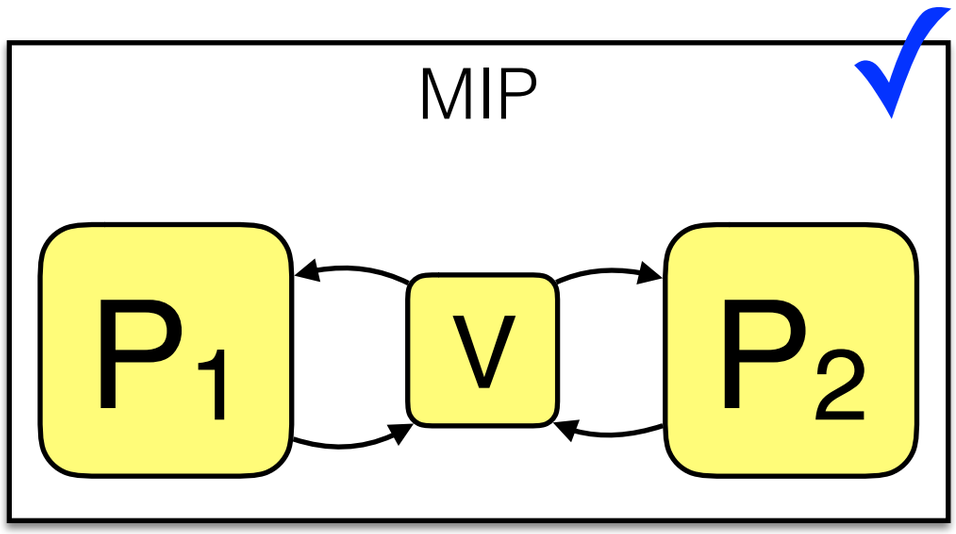


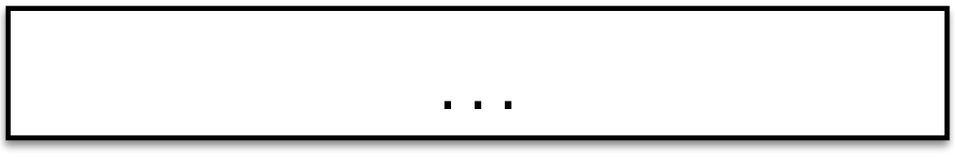


Cryptography As A Bridge

Inconvenient Succinct Proofs



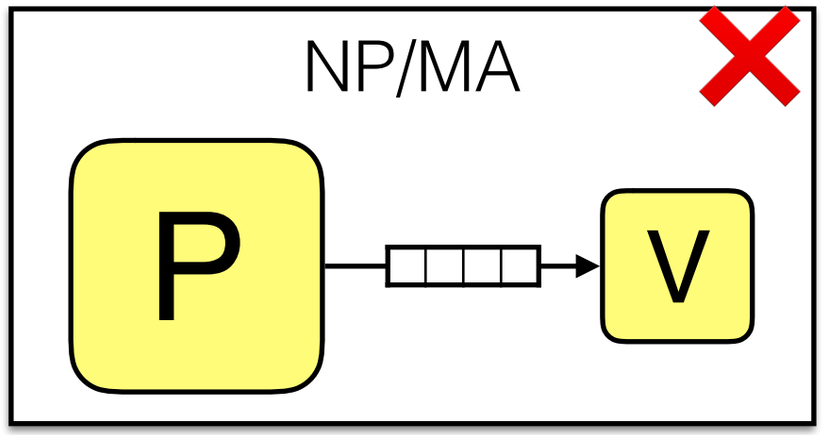


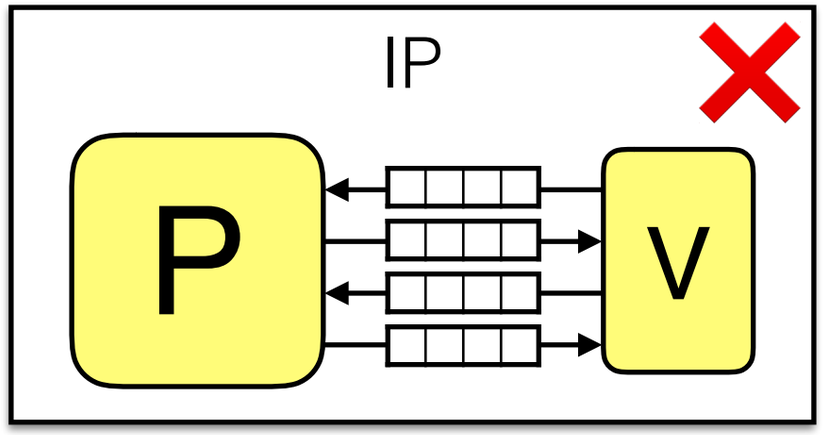


CRYPTO

Arguments

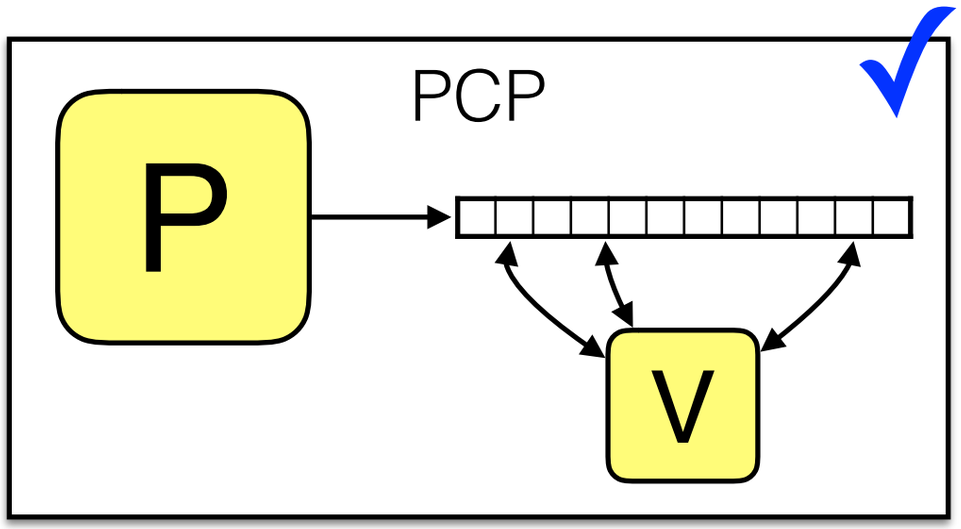
Convenient Succinct ~~Proofs~~

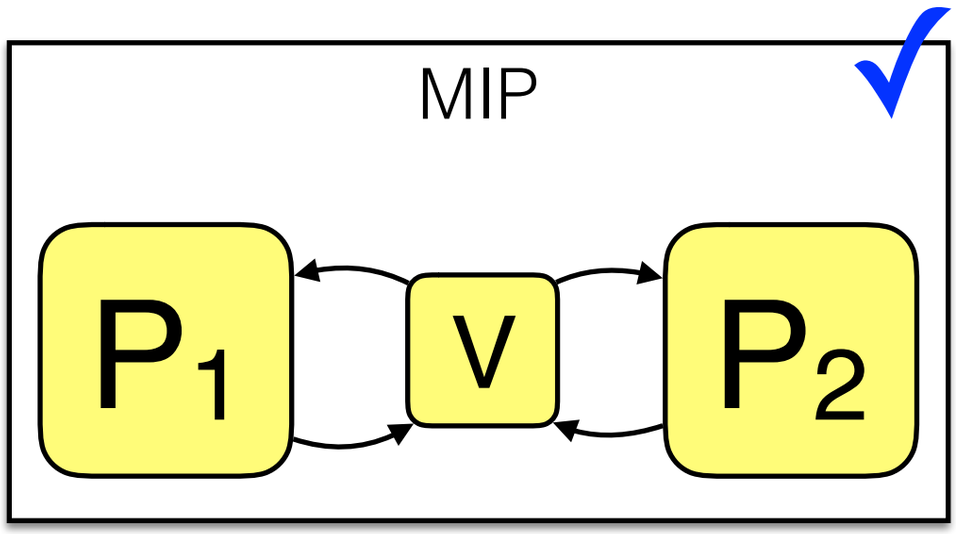


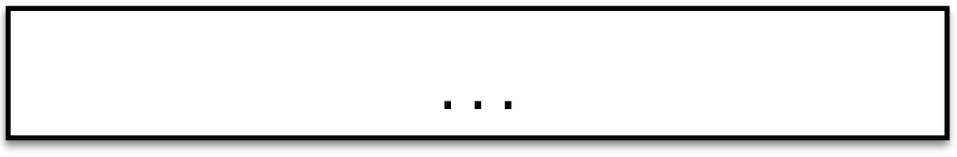


Cryptography As A Bridge

Inconvenient Succinct Proofs



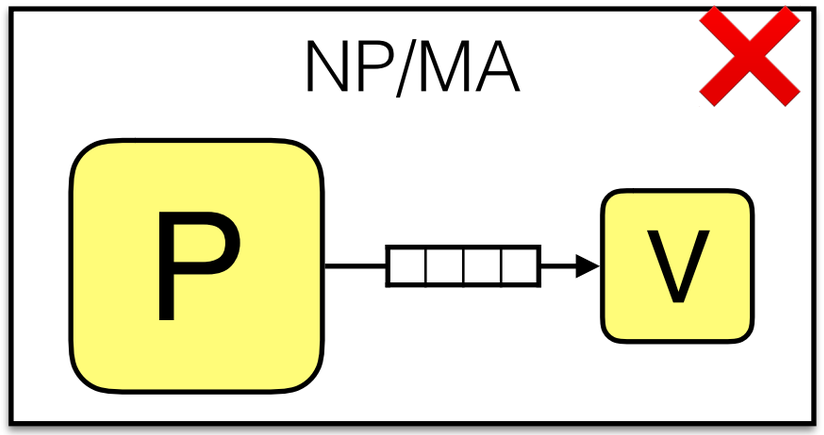


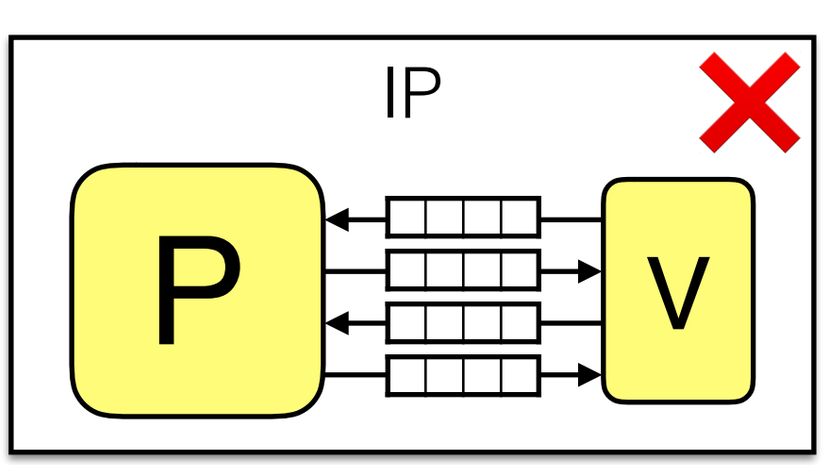


CRYPTO

Arguments

Convenient Succinct ~~Proofs~~



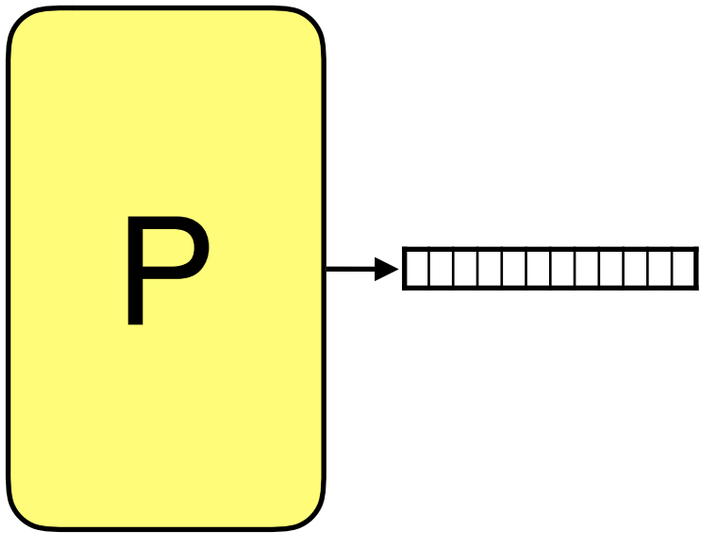


A very productive approach
to construct *succinct arguments*

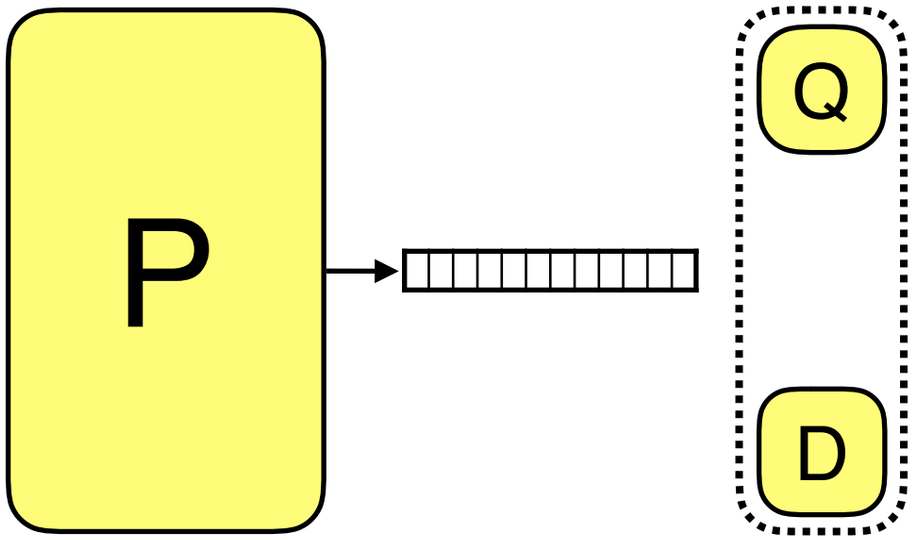
Constructions

From **PCPs**

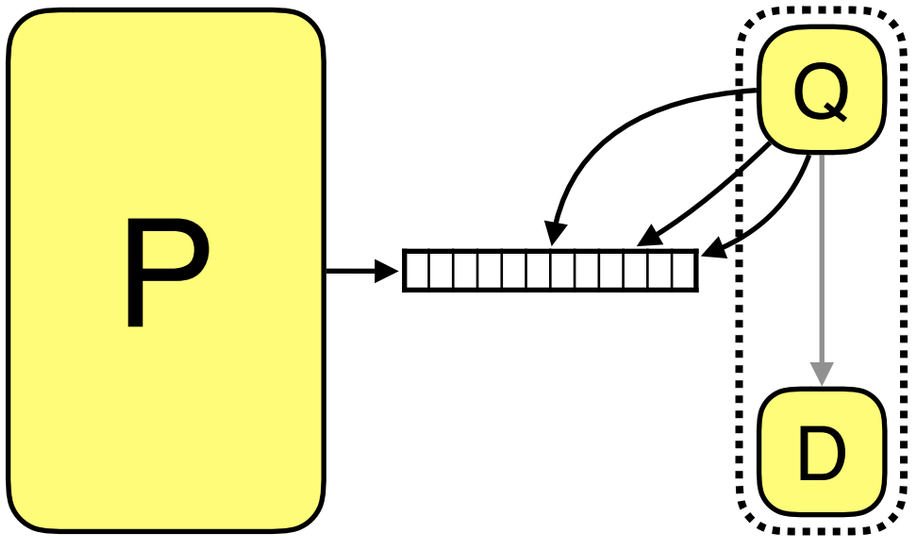
From **PCPs**



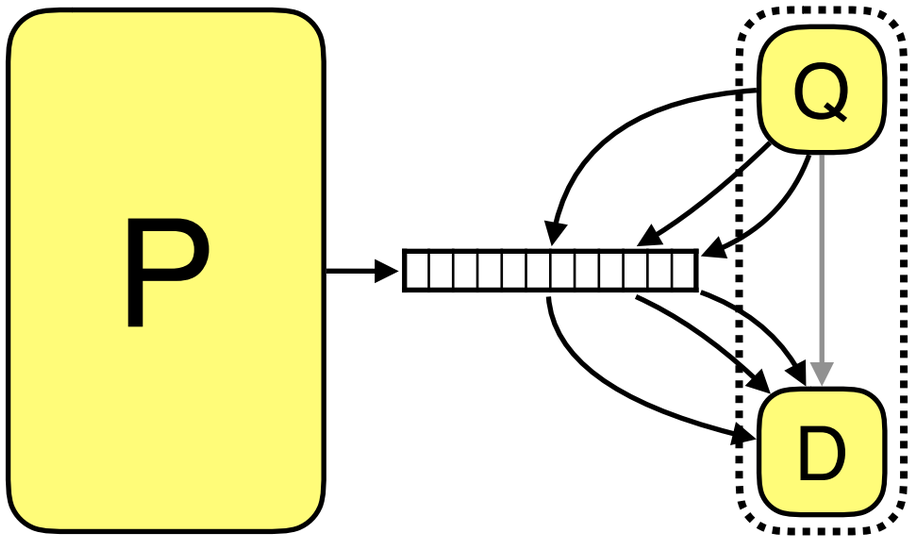
From **PCPs**



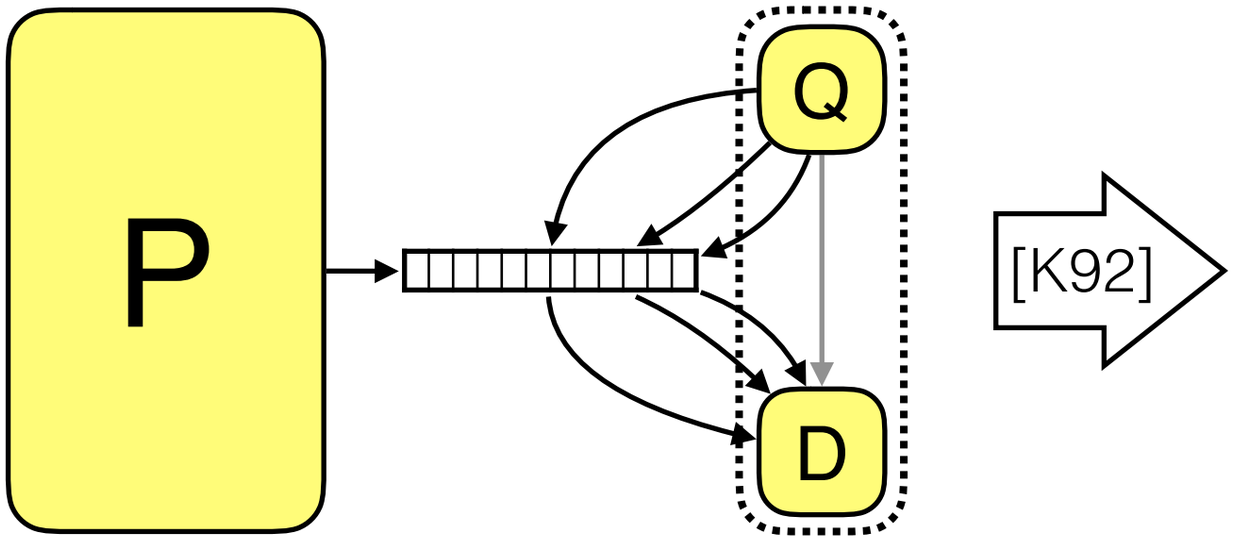
From **PCPs**



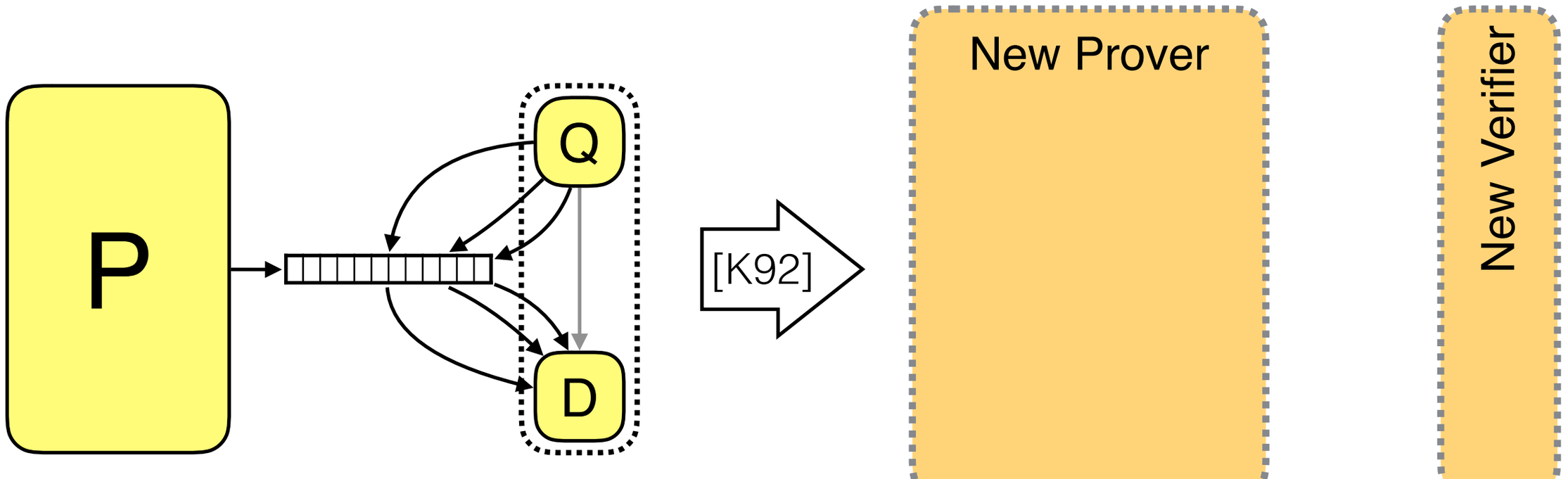
From **PCPs**



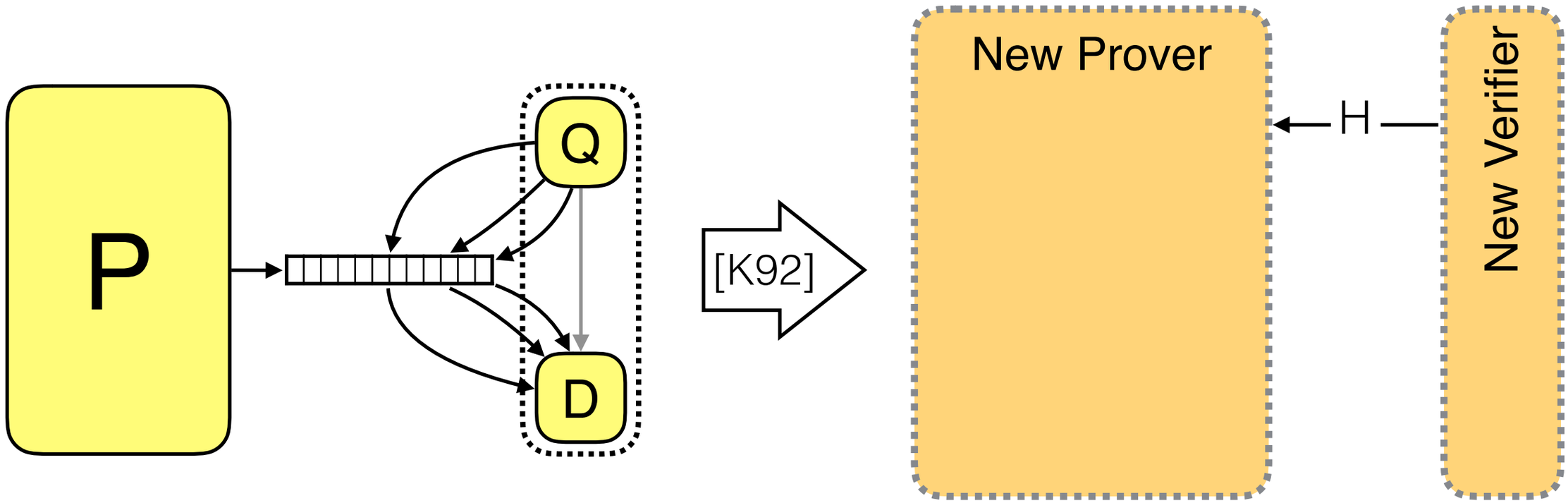
From **PCPs**



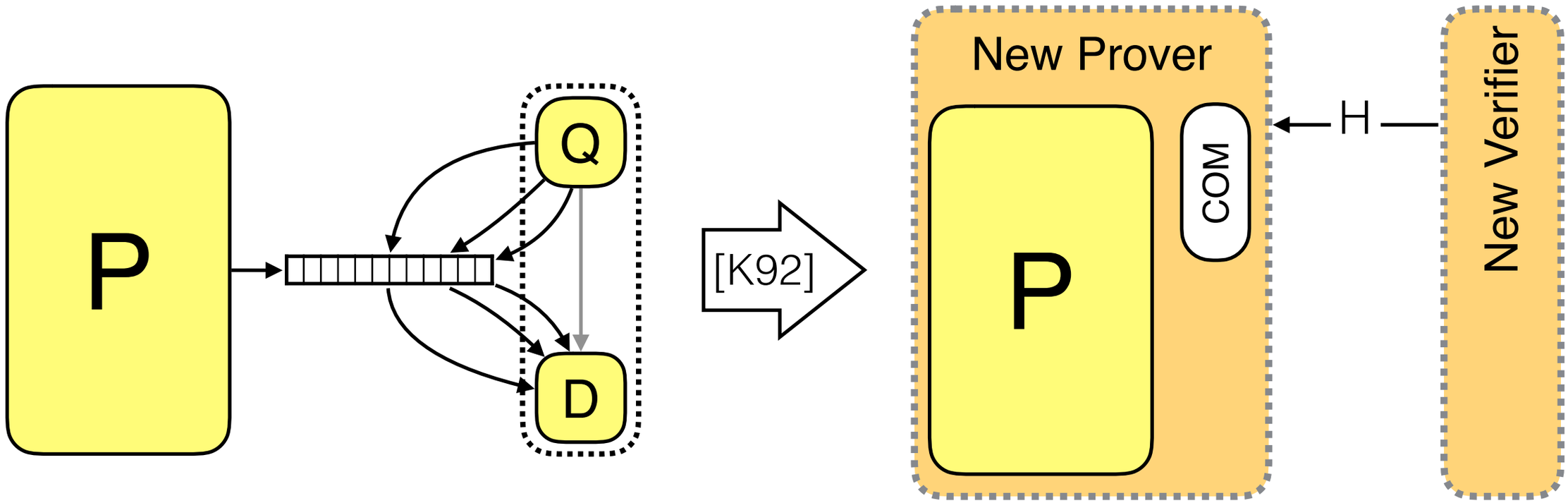
From **PCPs**



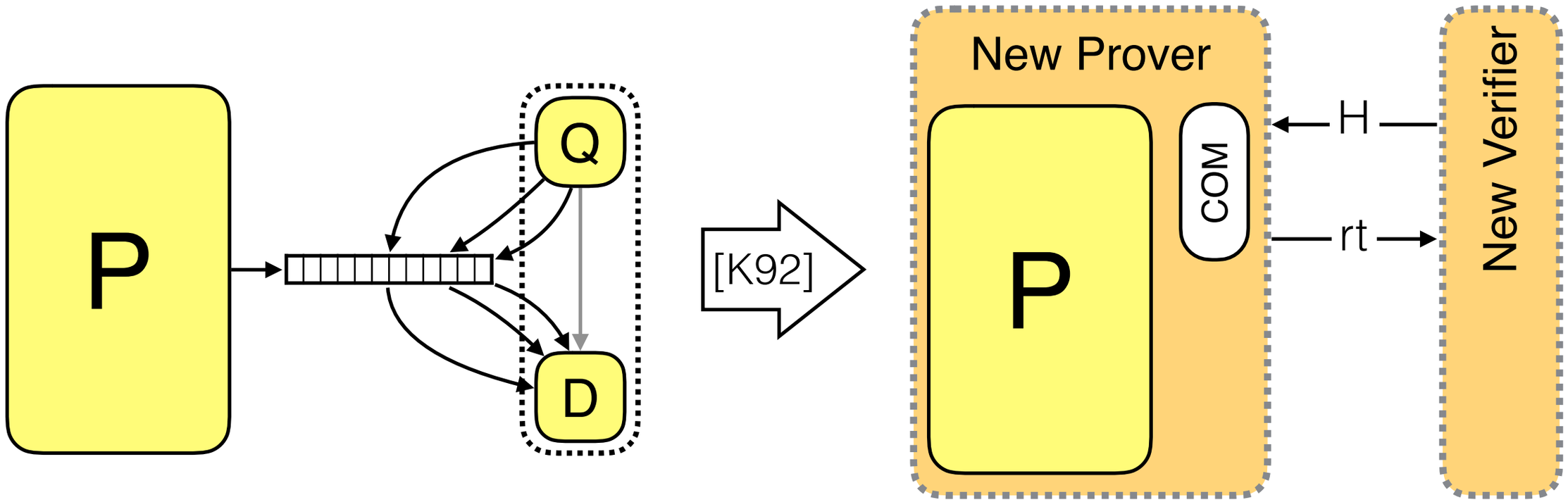
From **PCPs**



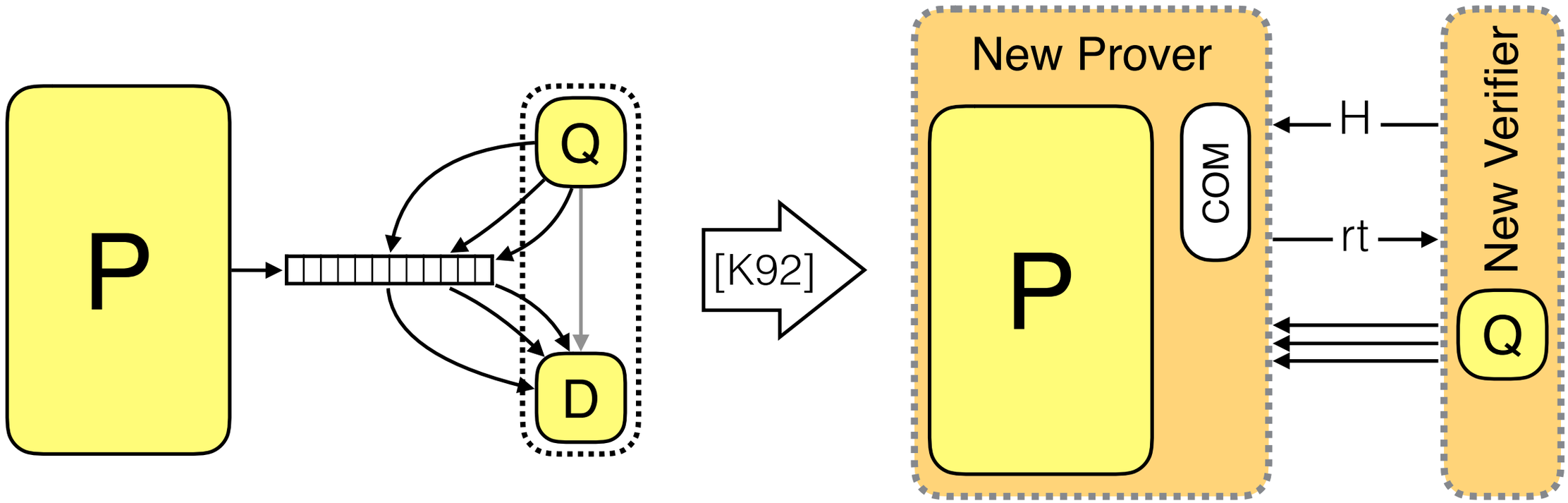
From **PCPs**



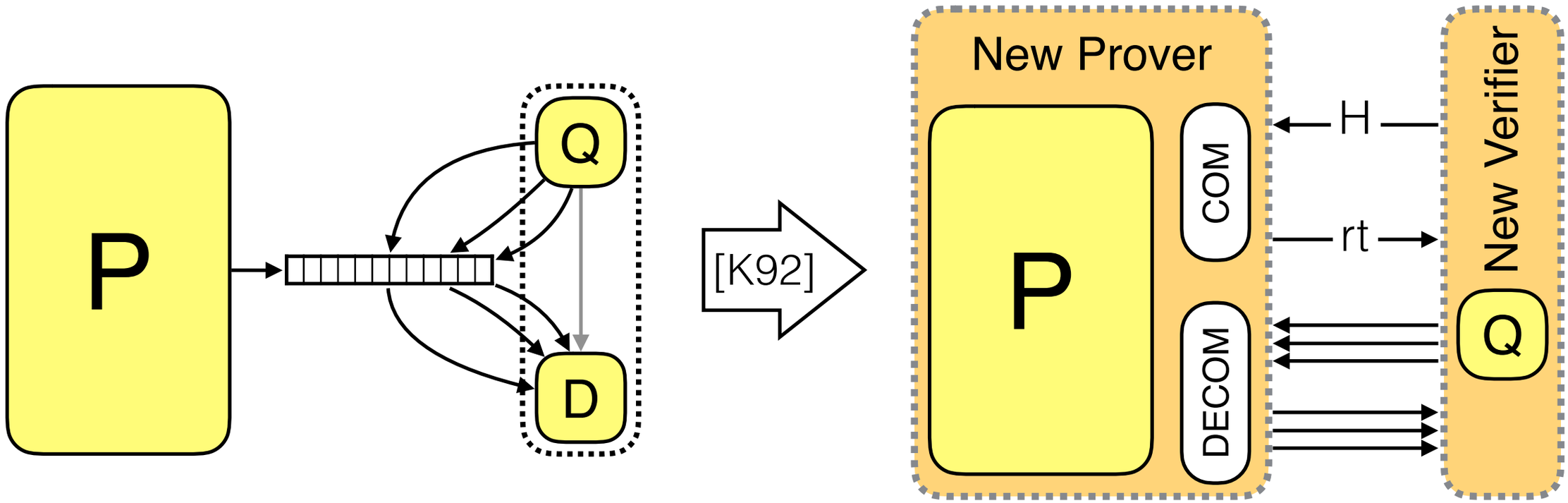
From **PCPs**



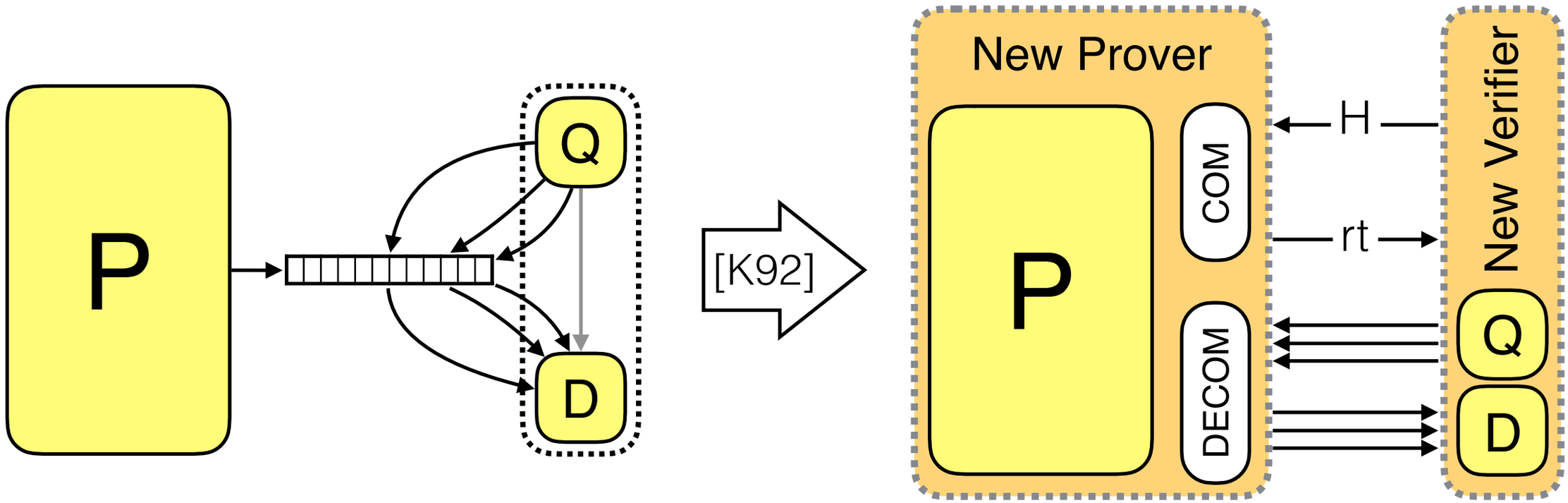
From **PCPs**



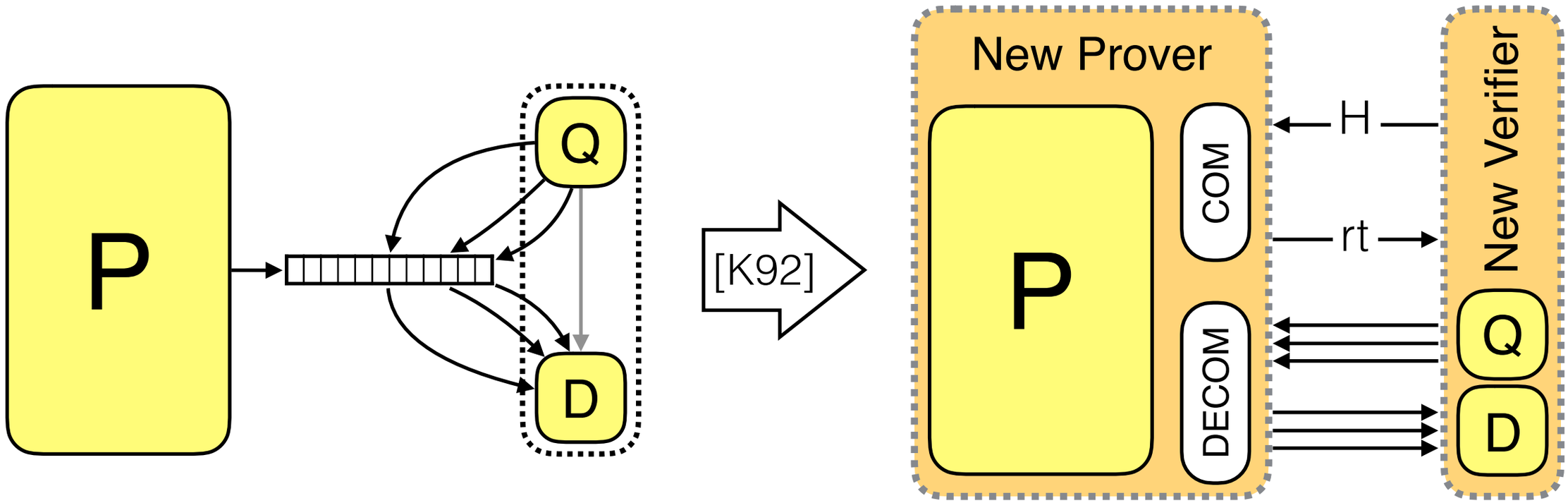
From **PCPs**



From **PCPs**

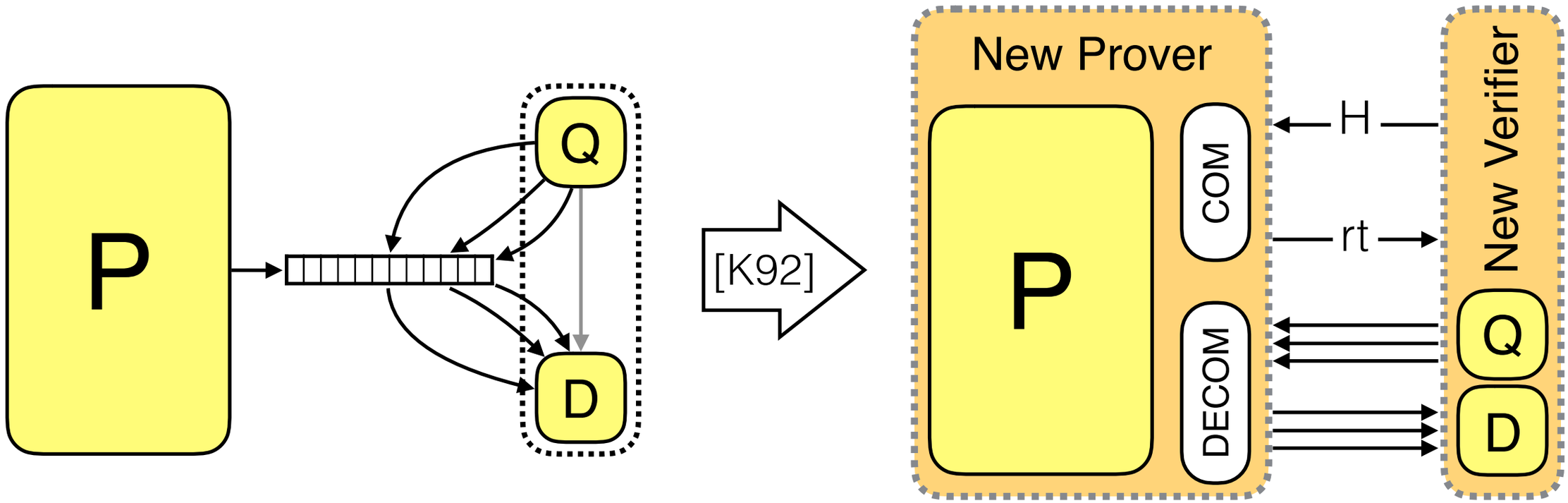


From **PCPs**



work	number of messages	cryptographic tool
[K92][BG08]	4	collision-resistant hashing
[BKP17]	3	multicollision-resistant hashing
[BCCGLRT16] [DFH12]	2	extractable collision-resistant hashing
[M94]	1	random oracle

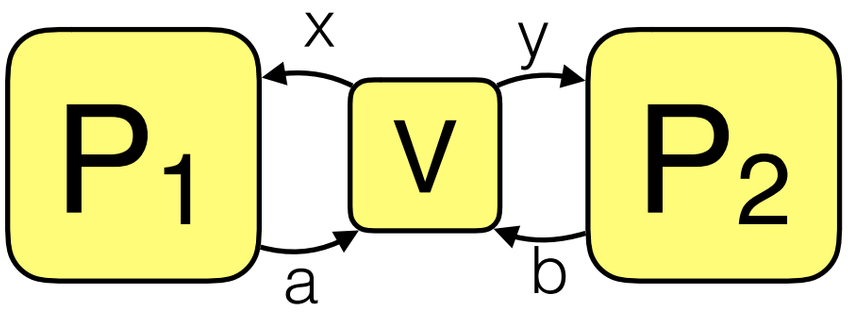
From **PCPs**



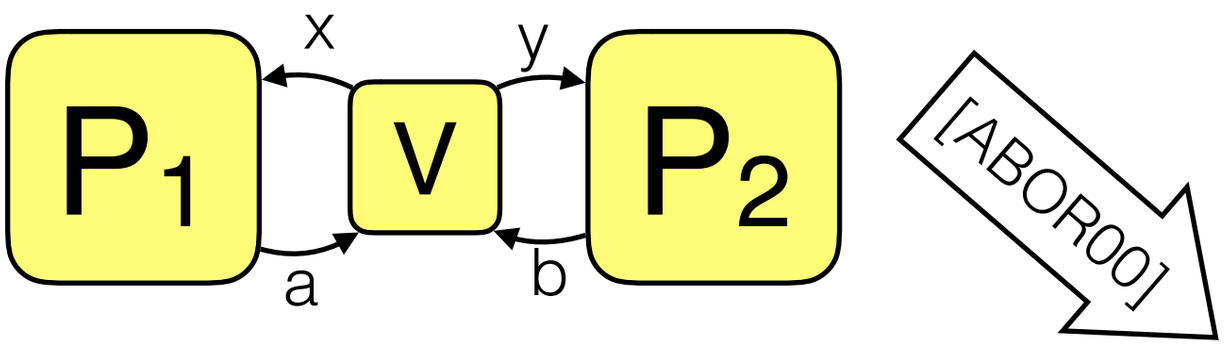
work	number of messages	cryptographic tool
[K92][BG08]	4	collision-resistant hashing
[BKP17]	3	multicollision-resistant hashing
[BCCGLRT16] [DFH12]	2	extractable collision-resistant hashing
[M94]	1	random oracle

From **MIPs**

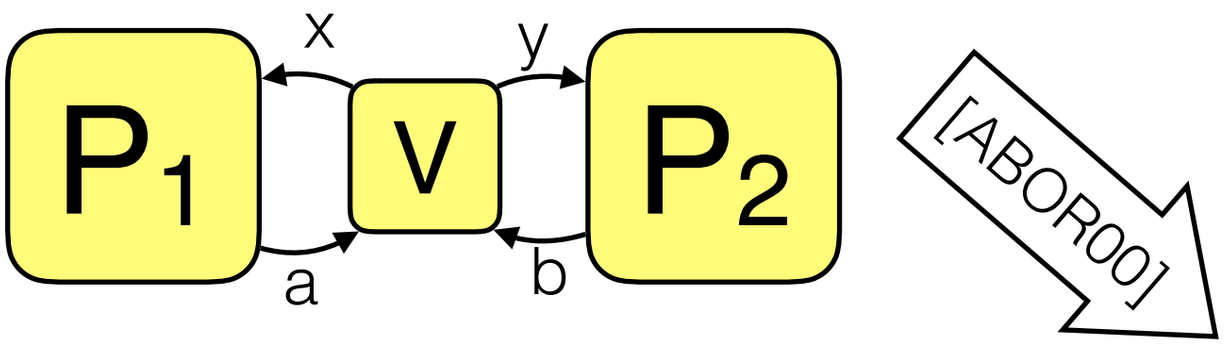
From **MIPs**



From **MIPs**



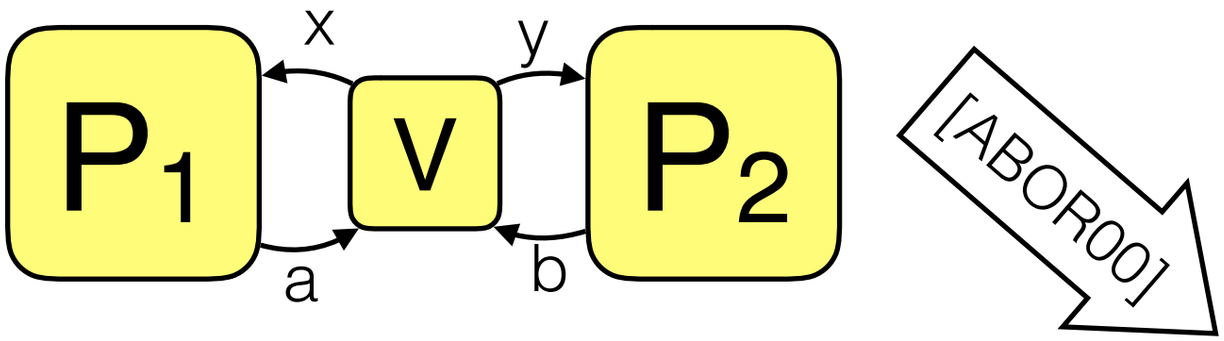
From **MIPs**

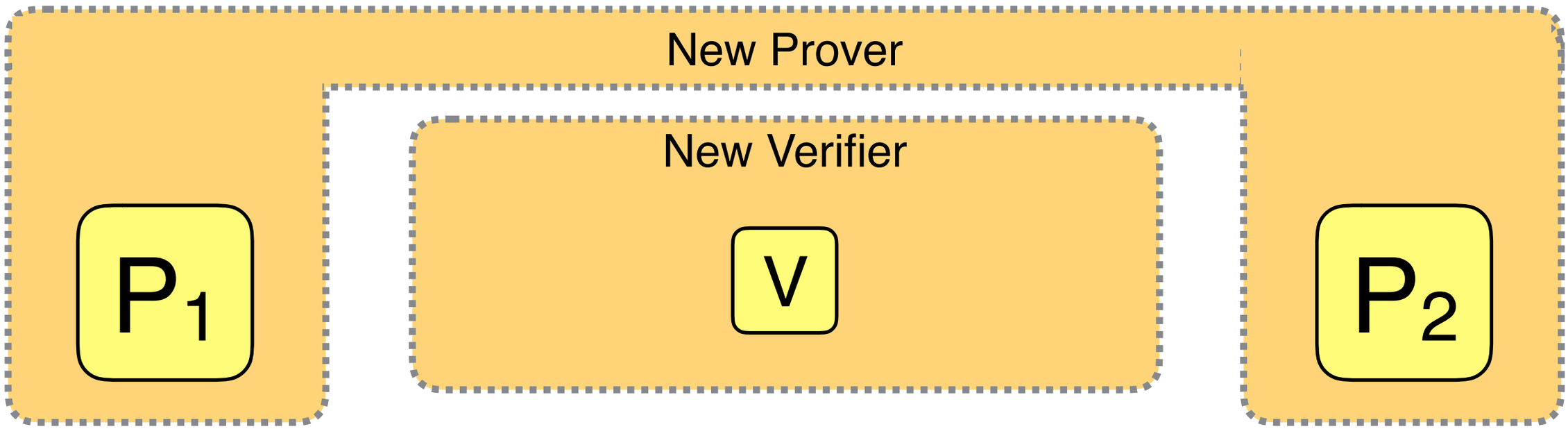


New Prover

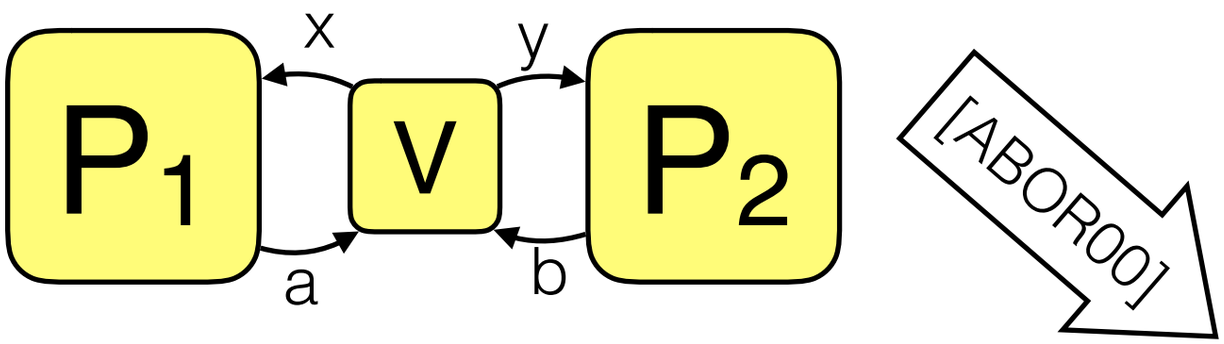
New Verifier

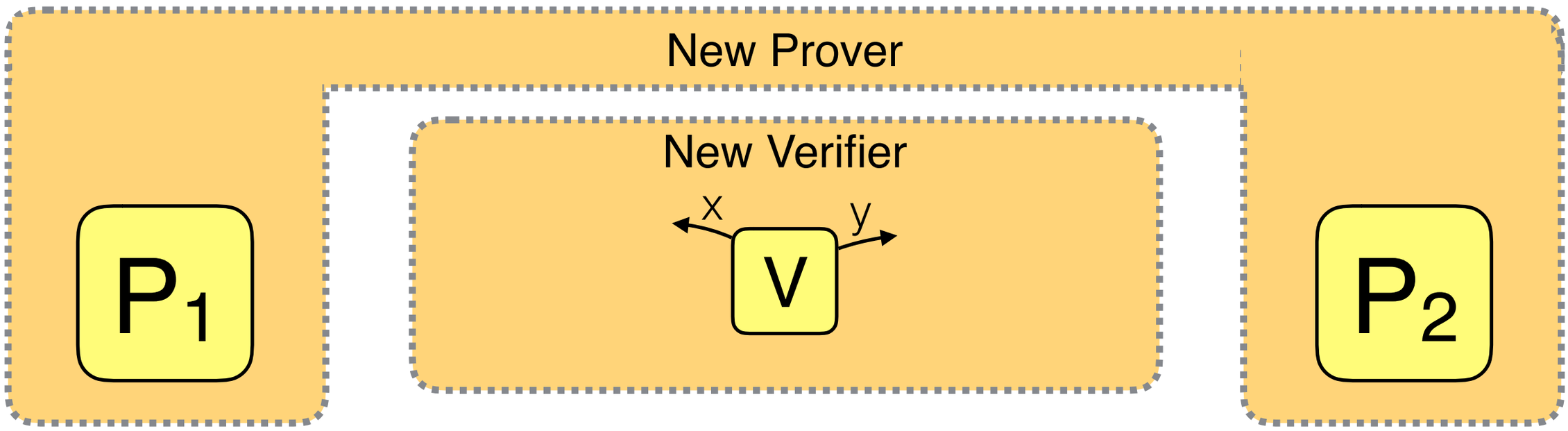
From **MIPs**



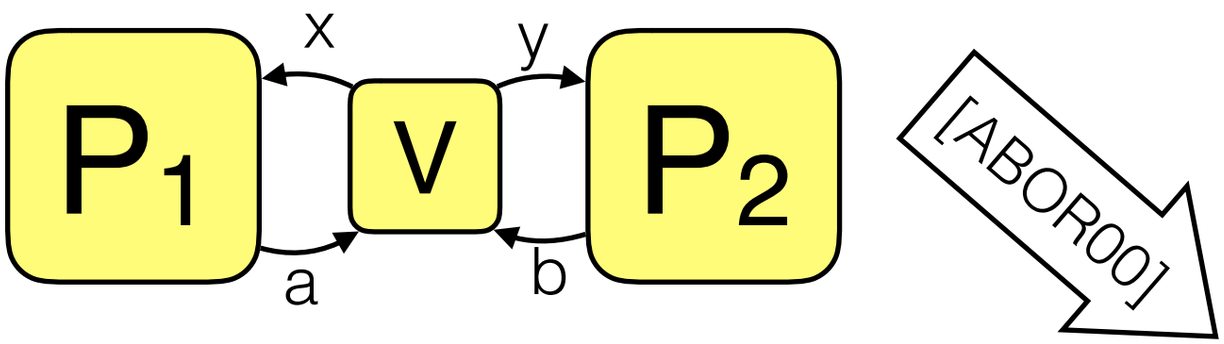


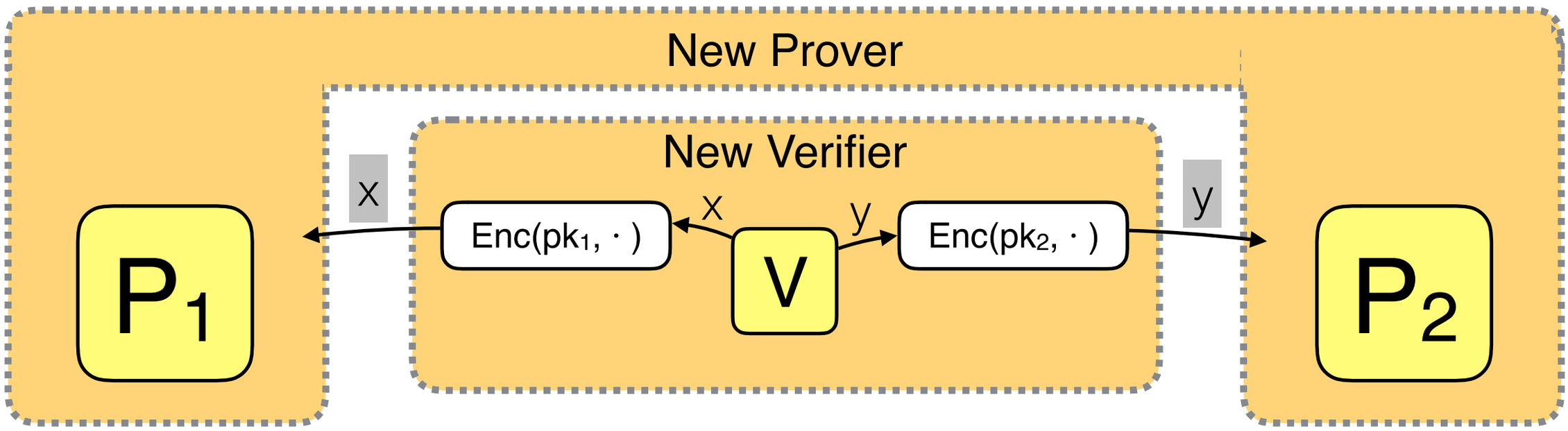
From **MIPs**



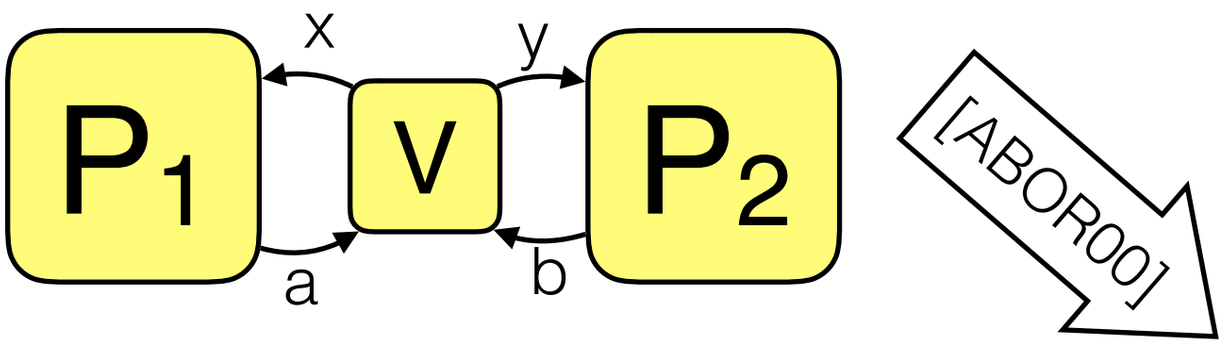


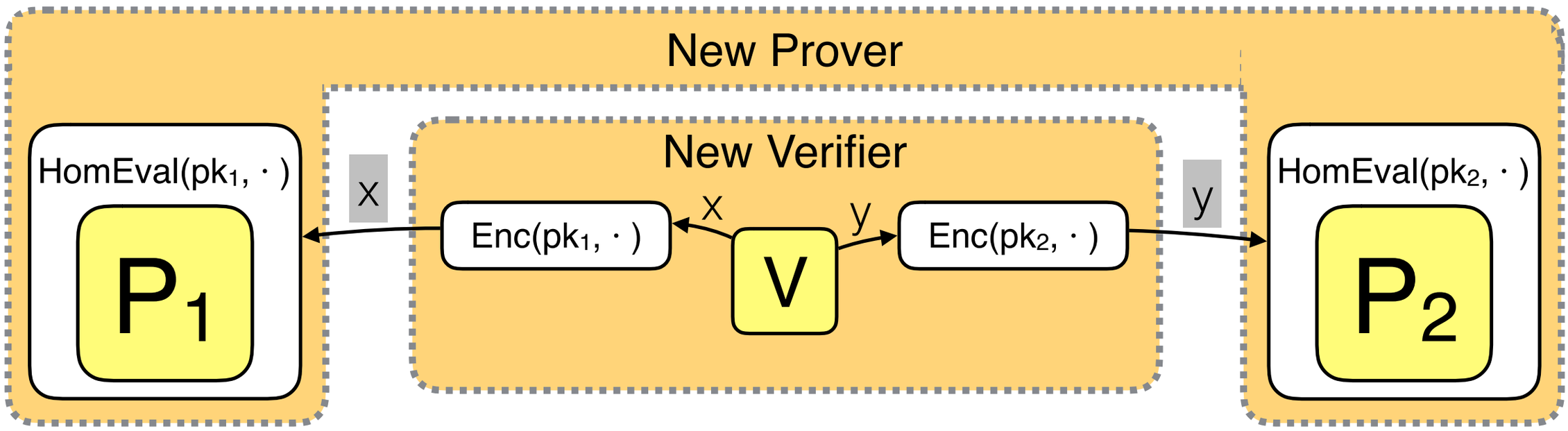
From **MIPs**


P1
V
P2
x
y
a
b
[ABOR00]

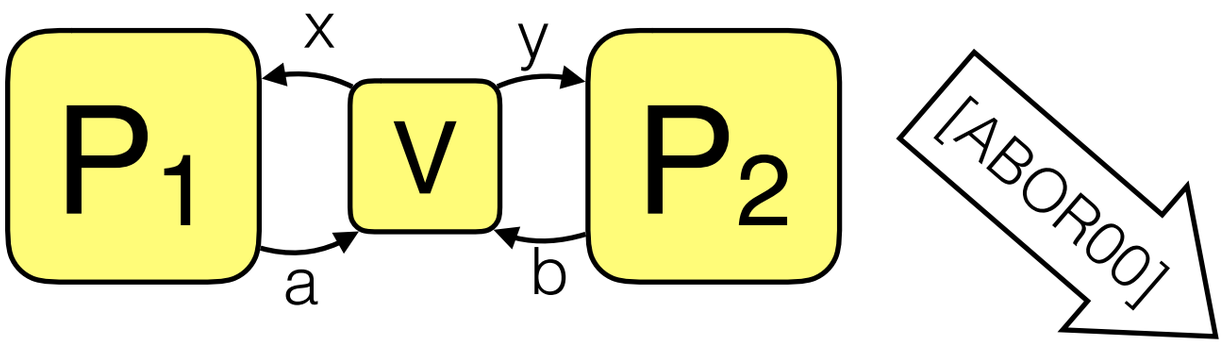

New Prover
New Verifier
x
Enc(pk1, ·)
x
y
Enc(pk2, ·)
y
P1
V
P2

From **MIPs**

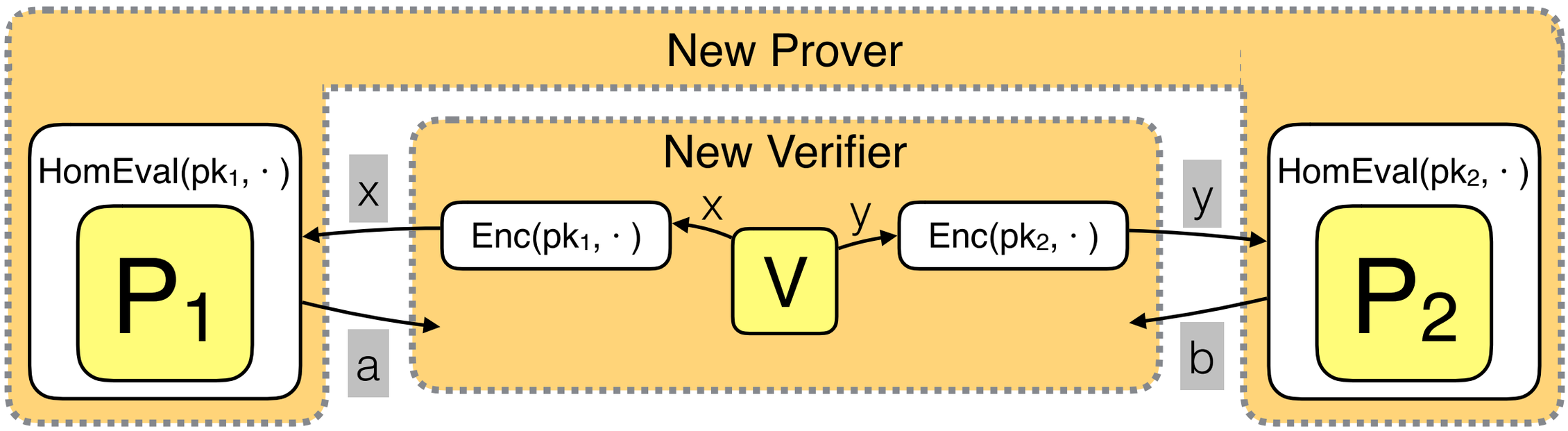

P1
V
P2
x
y
a
b
[ABOR00]


New Prover
HomEval(pk1, ·)
P1
x
New Verifier
Enc(pk1, ·)
x
y
V
Enc(pk2, ·)
y
HomEval(pk2, ·)
P2

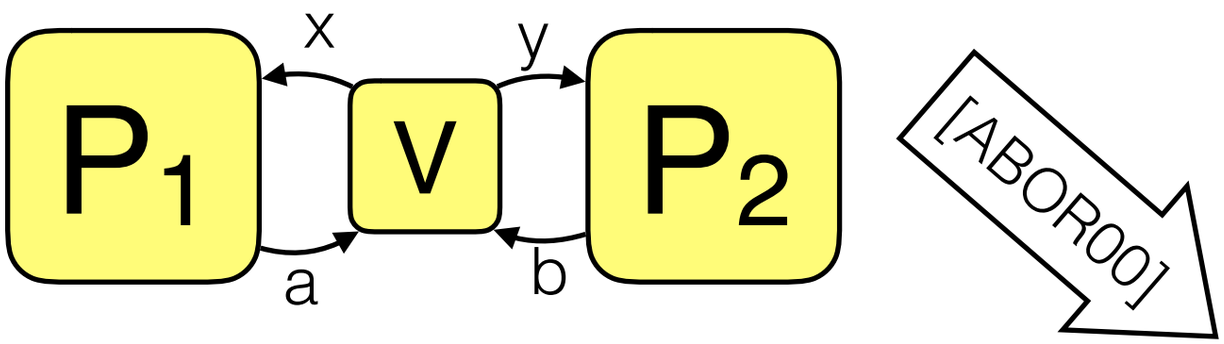
From **MIPs**

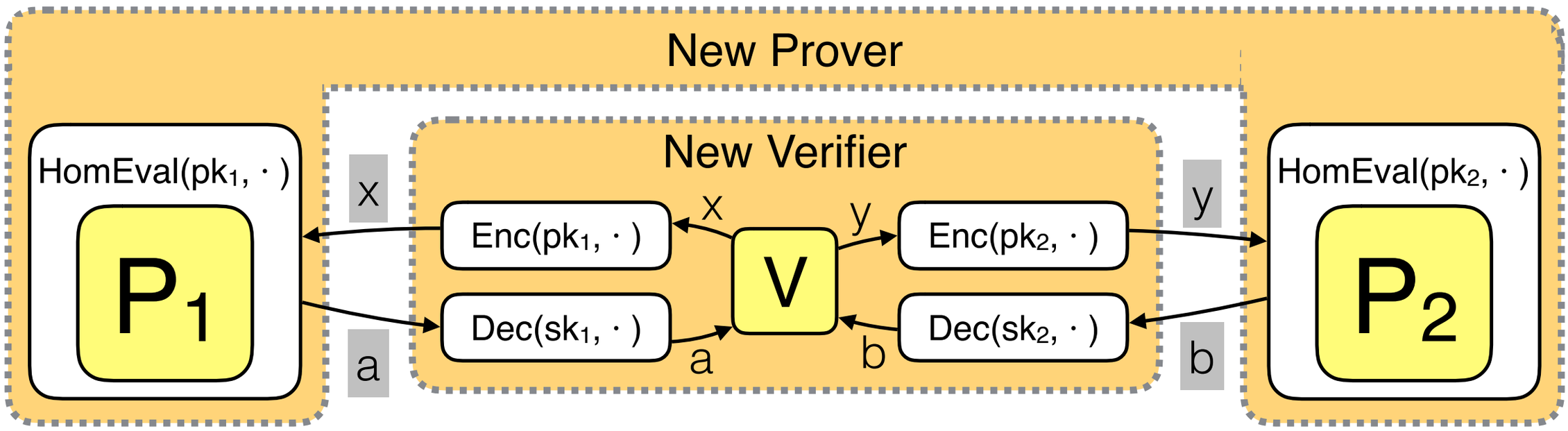

P1
V
P2
x
y
a
b

[ABOR00]

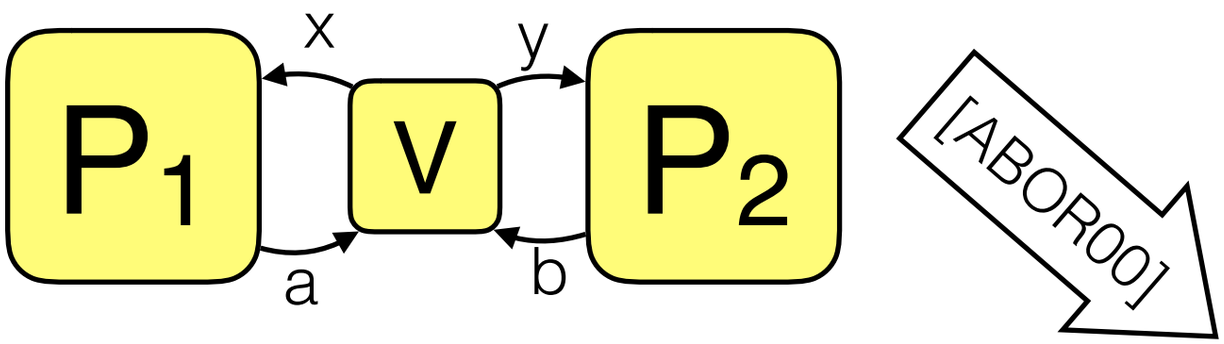

New Prover
New Verifier
HomEval(pk1, ·)
HomEval(pk2, ·)
Enc(pk1, ·)
Enc(pk2, ·)
x
y
V
P1
P2
a
b

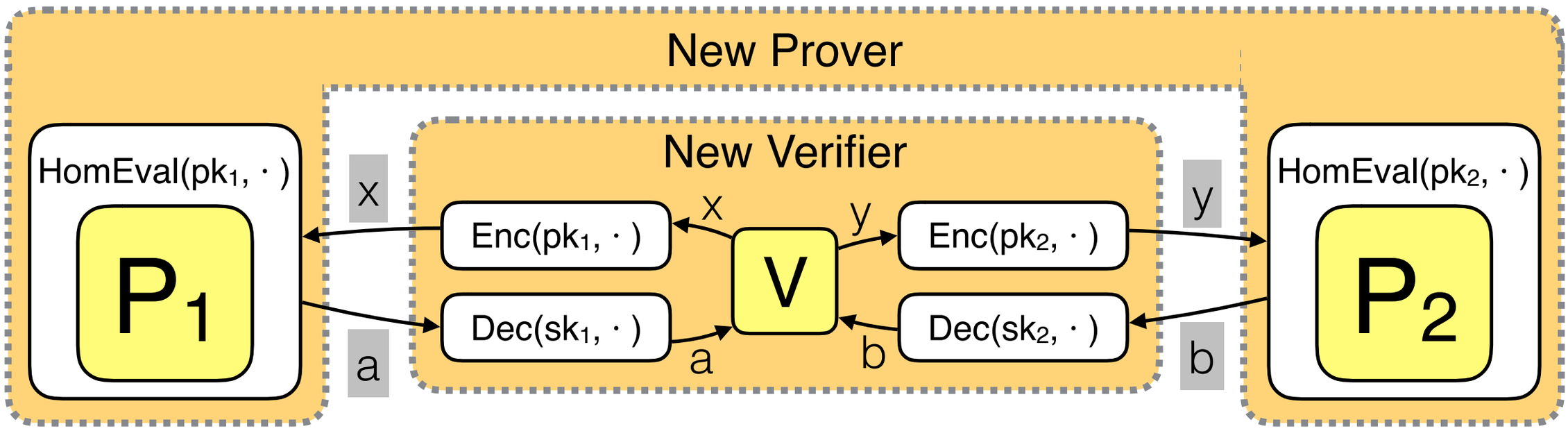
From **MIPs**


P1
V
P2
x
y
a
b
[ABOR00]


New Prover
New Verifier
HomEval(pk1, ·)
HomEval(pk2, ·)
P1
P2
V
Enc(pk1, ·)
Enc(pk2, ·)
Dec(sk1, ·)
Dec(sk2, ·)
x
y
a
b

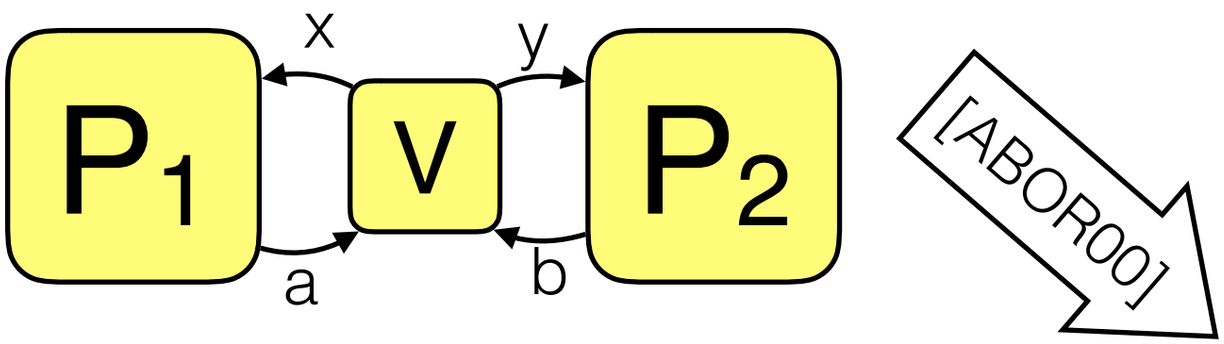
From **MIPs**

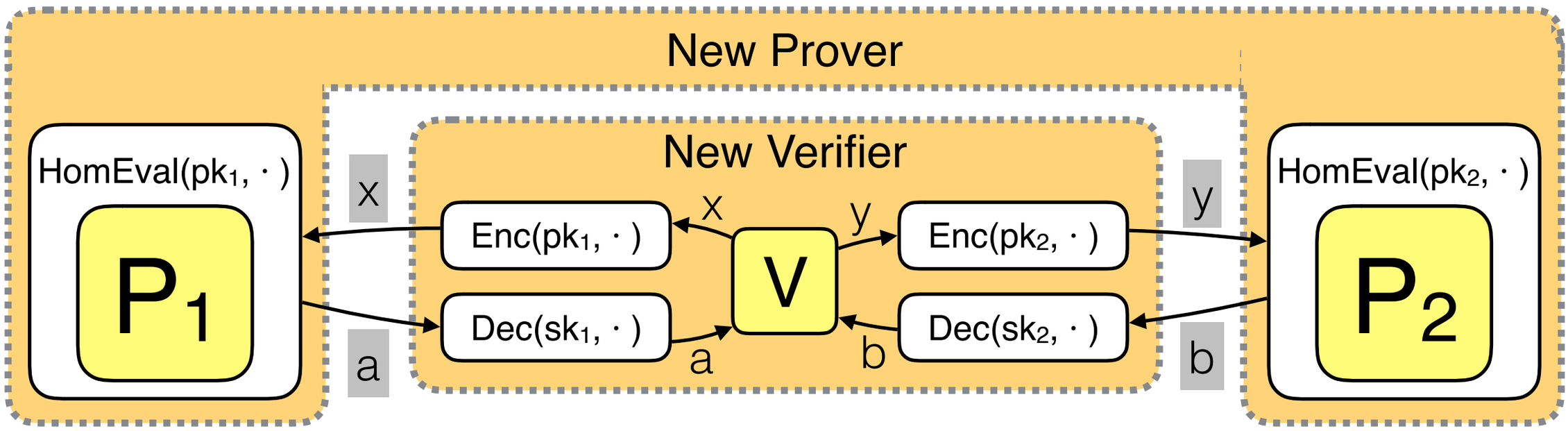




work	type of MIP	number of messages	cryptographic tool
[BC12]	standard (captures **NP**)	4	FHE
[BC12]		2	*extractable* FHE
[DHRW16]		2	*spooky-free* FHE
[KRR13] [KRR14]	no-signaling (captures **P**)	2	FHE

From **MIPs**

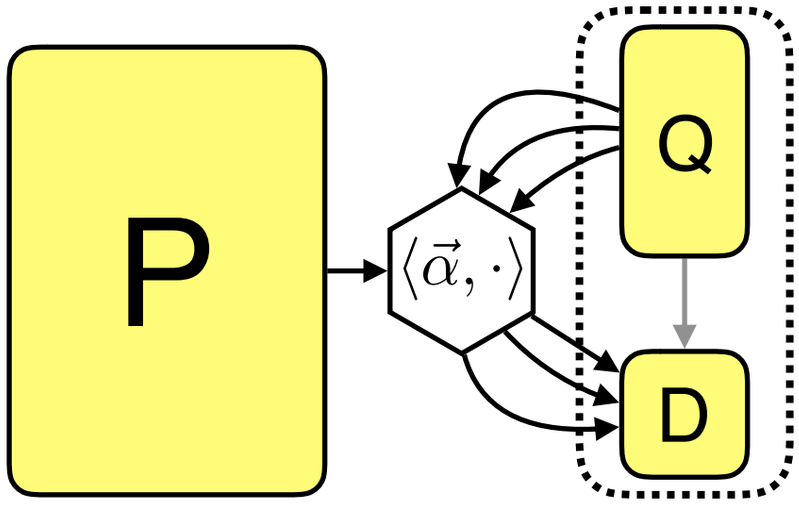




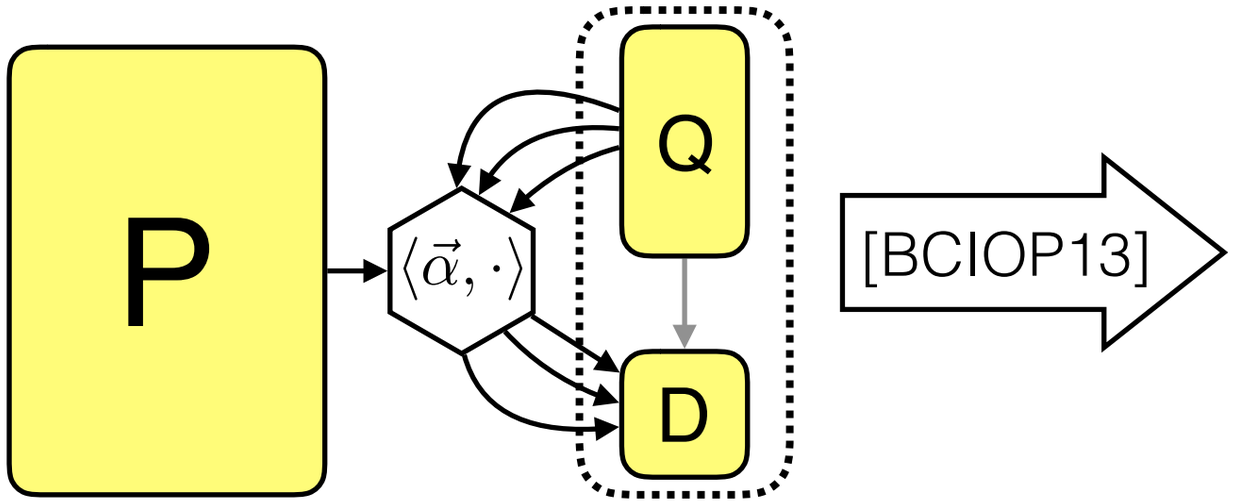
work	type of MIP	number of messages	cryptographic tool
[BC12]	standard (captures **NP**)	4	FHE
➡ [BC12]		2	*extractable* FHE
➡ [DHRW16]		2	*spooky-free* FHE
➡ [KRR13] [KRR14]	no-signaling (captures **P**)	2	FHE

From **Linear PCPs**

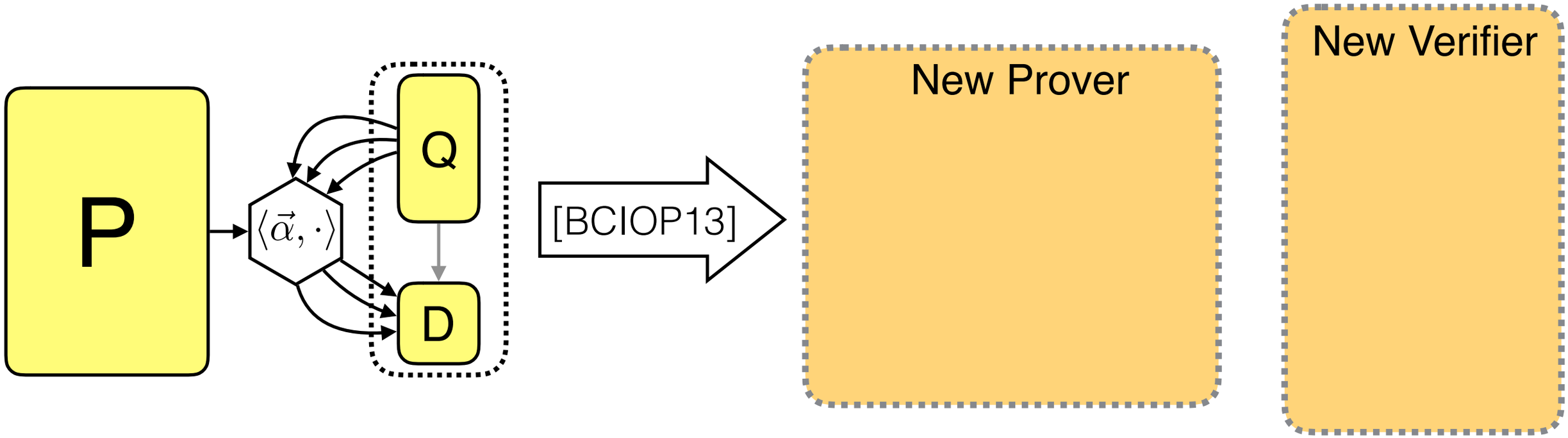
From **Linear PCPs**



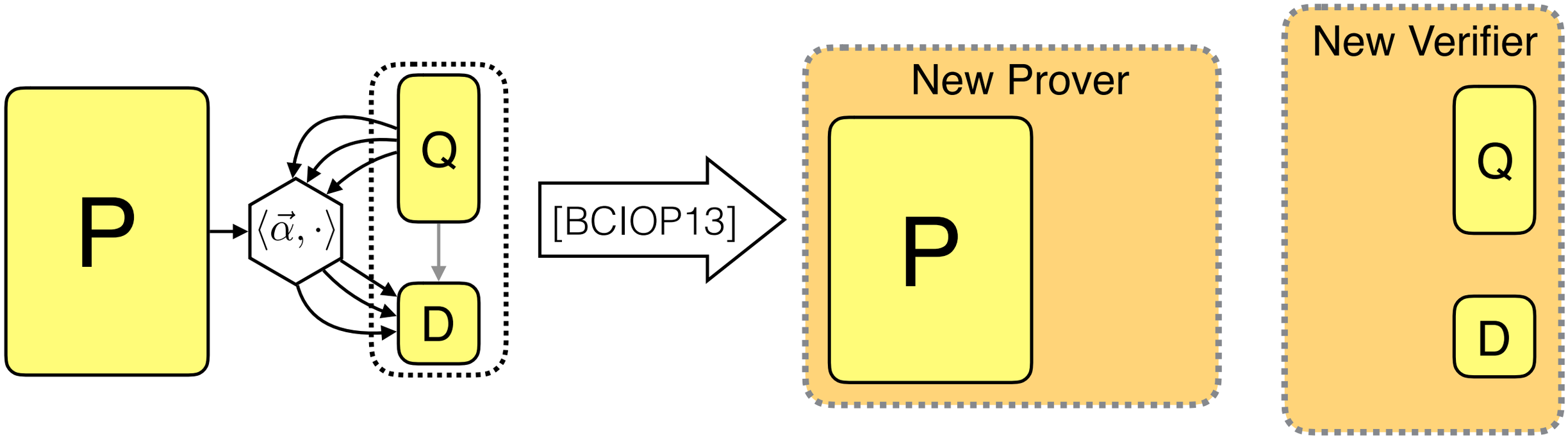
From **Linear PCPs**



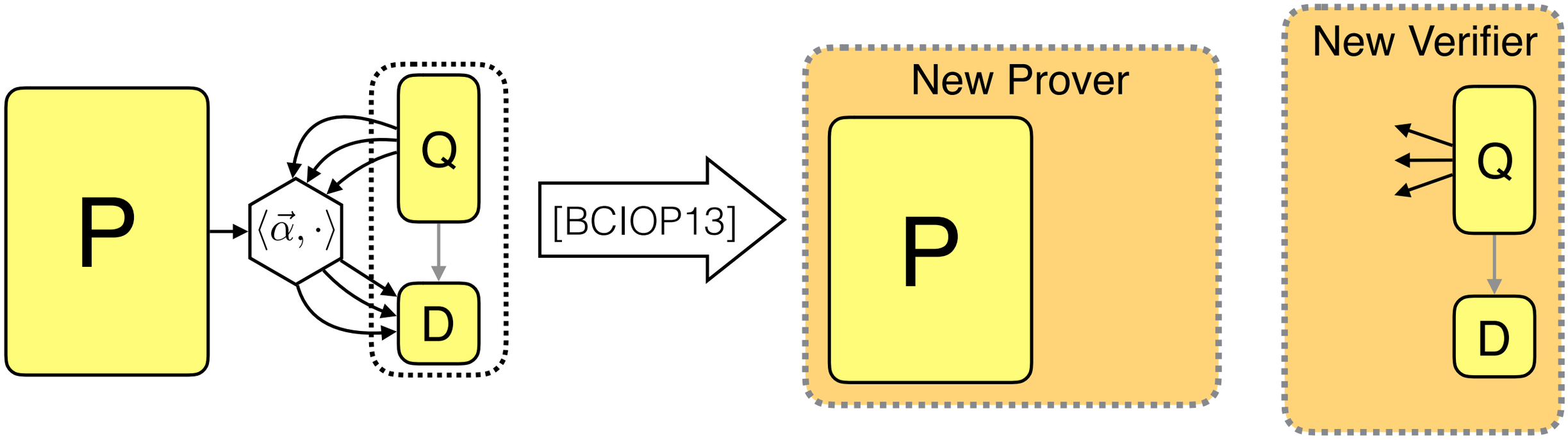
From **Linear PCPs**



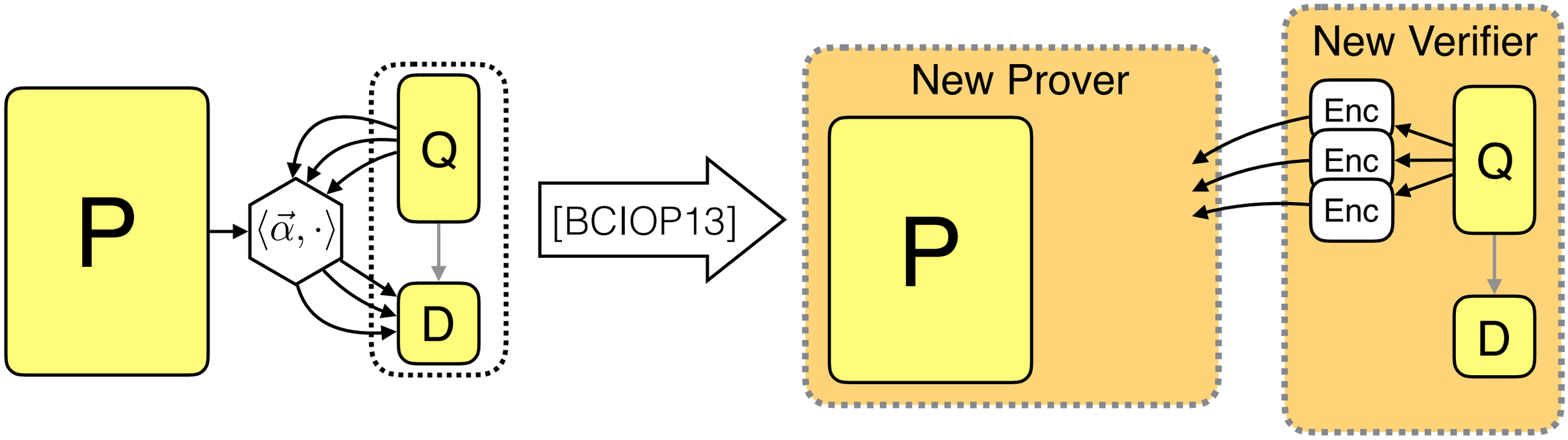
From **Linear PCPs**



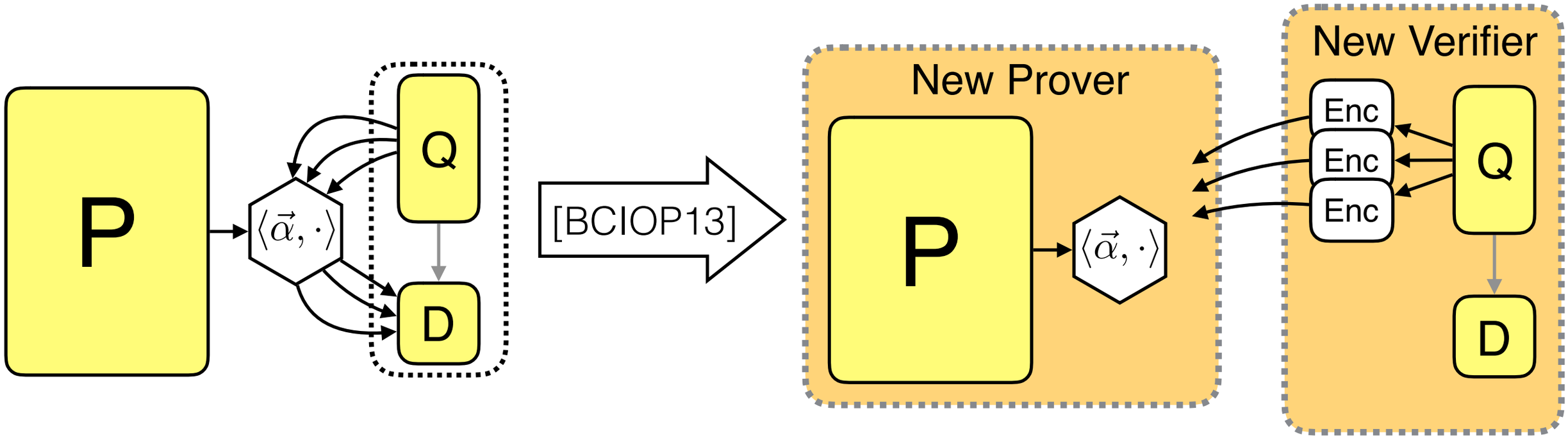
From **Linear PCPs**



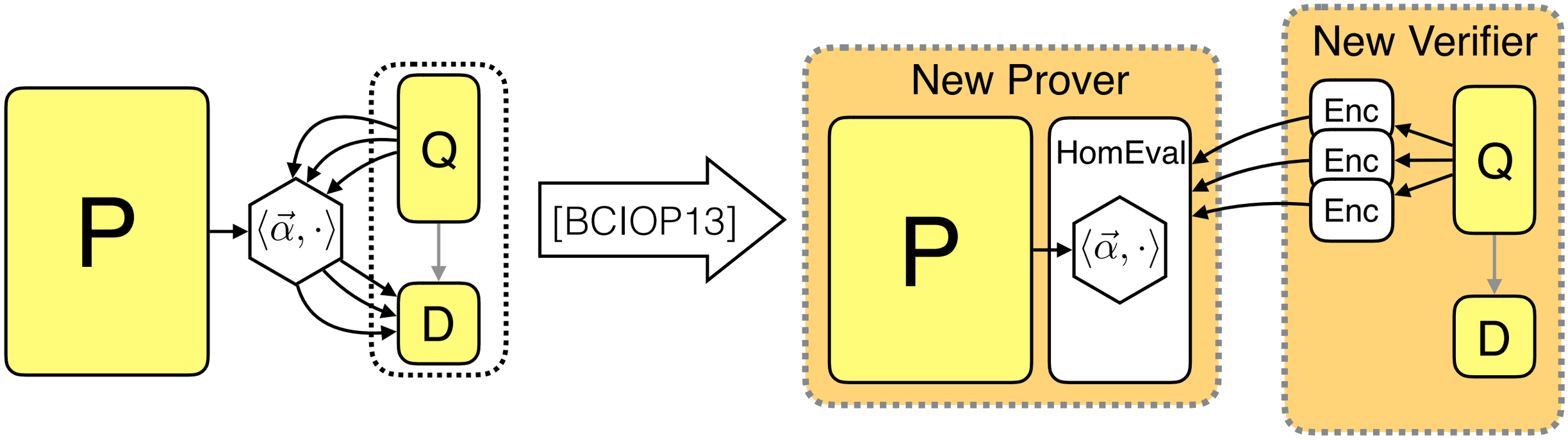
From **Linear PCPs**



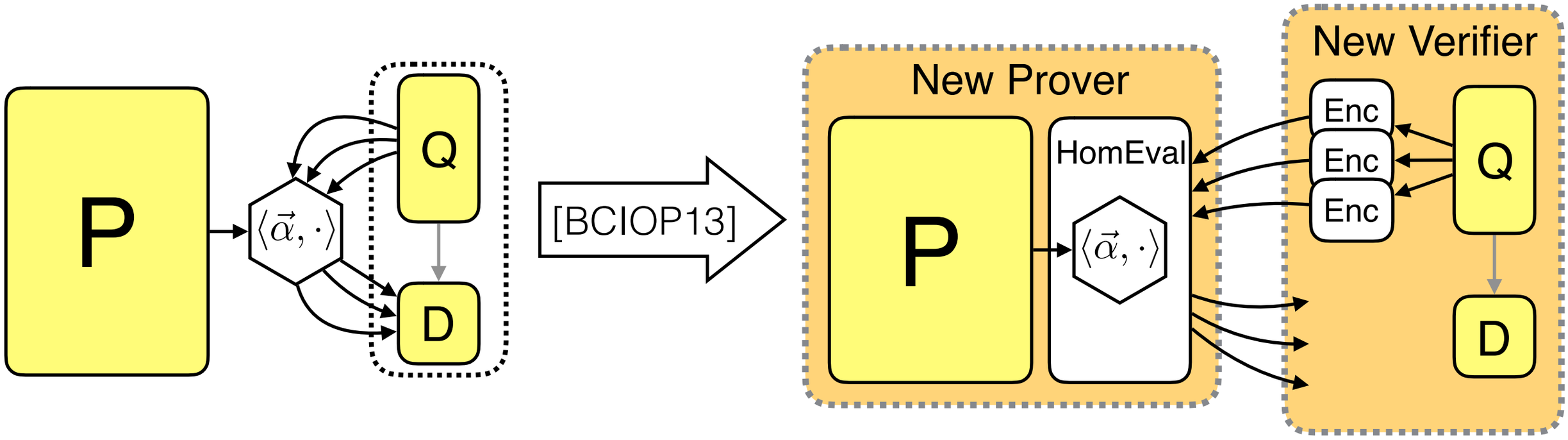
From **Linear PCPs**



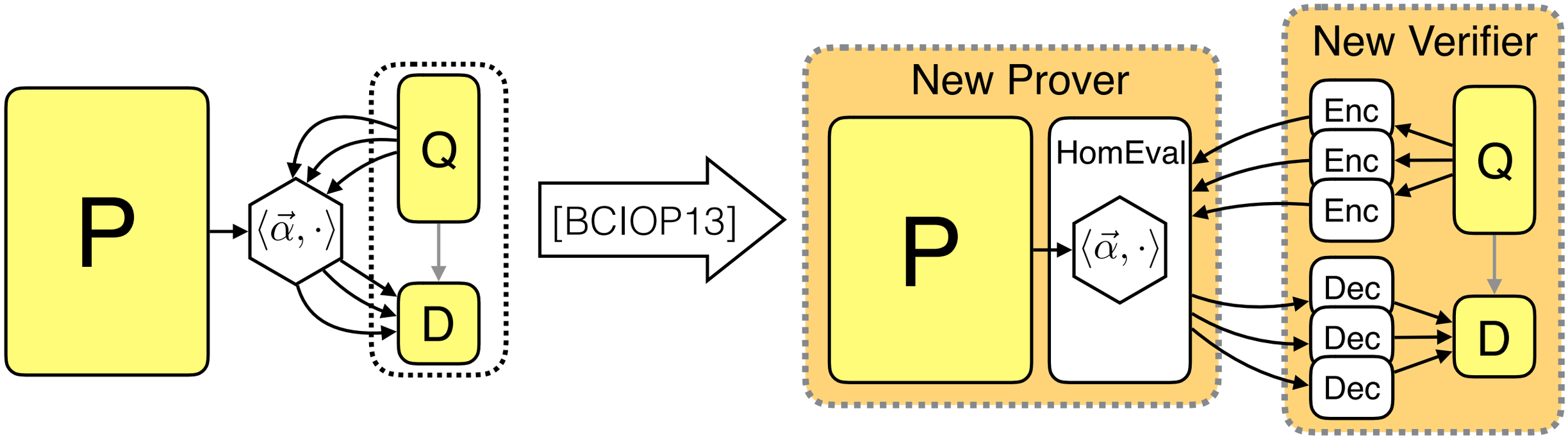
From **Linear PCPs**



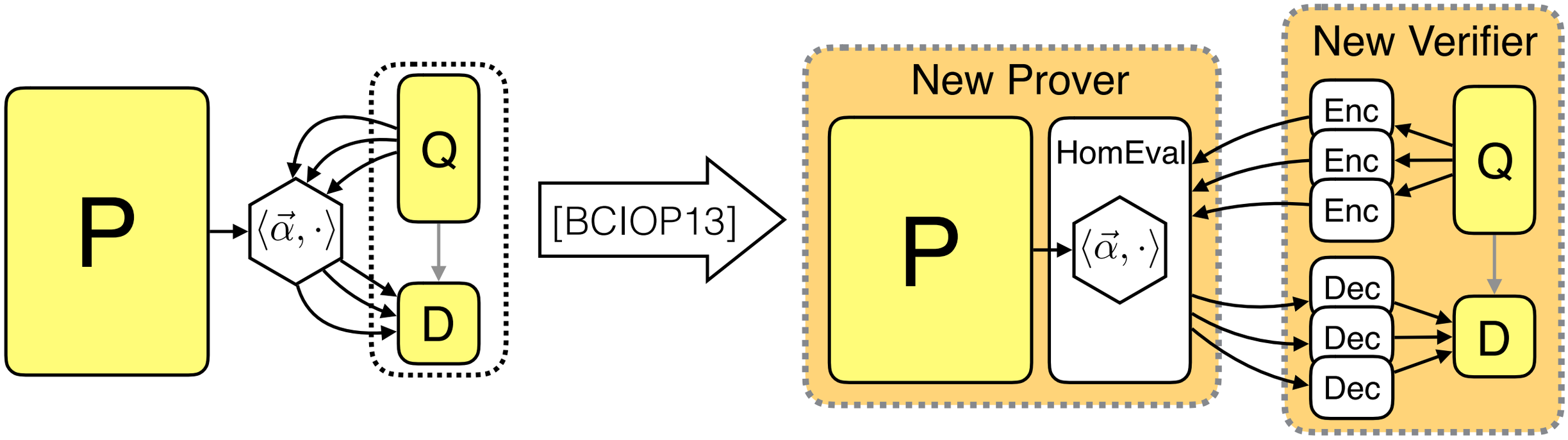
From **Linear PCPs**



From **Linear PCPs**

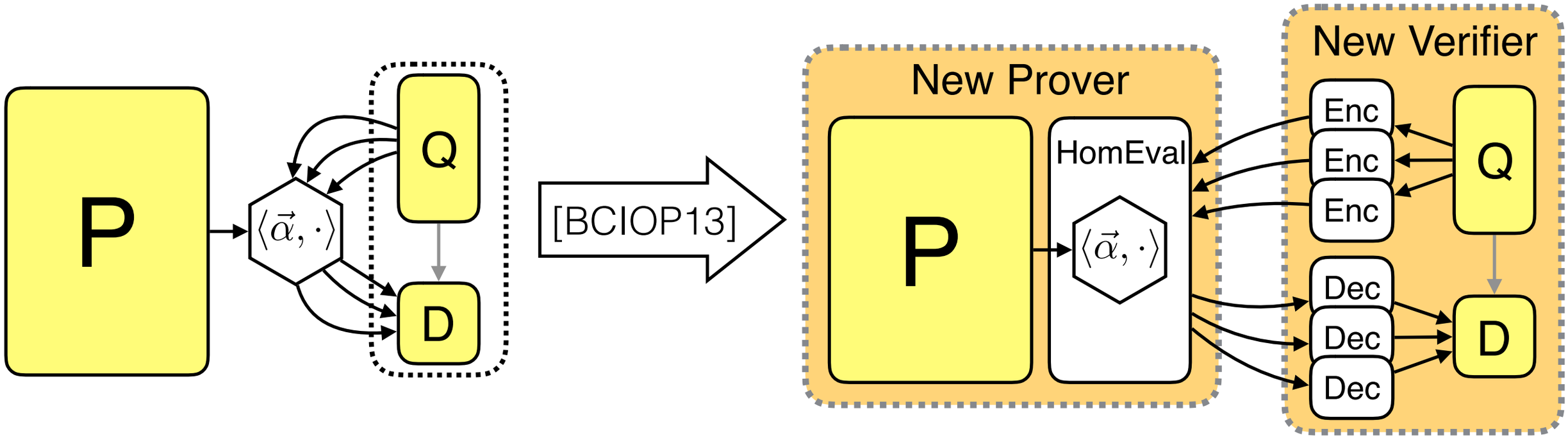


From **Linear PCPs**



work	number of messages	cryptographic tool
[IKO07]	4	linearly-homomorphic encryption
[BCIOP13]	2	linear-only encryption
[BISW17]	2	linear-only vector encryption
[BCIOP13]	1	linear-only encoding

From **Linear PCPs**

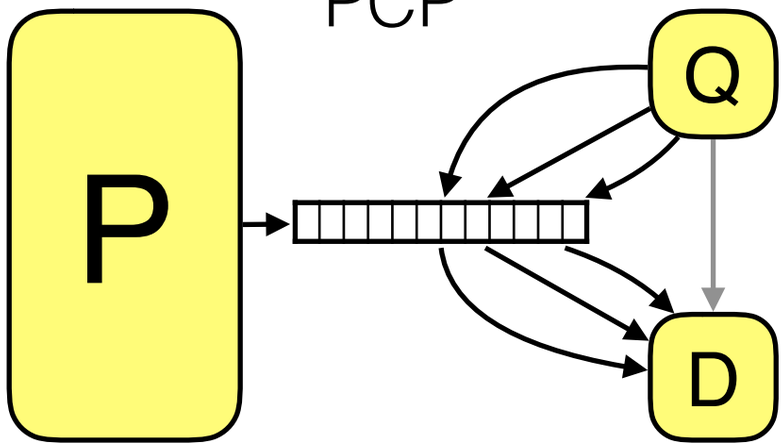


work	number of messages	cryptographic tool
[IKO07]	4	linearly-homomorphic encryption
[BCIOP13]	2	linear-only encryption
[BISW17]	2	linear-only vector encryption
[BCIOP13]	1	linear-only encoding

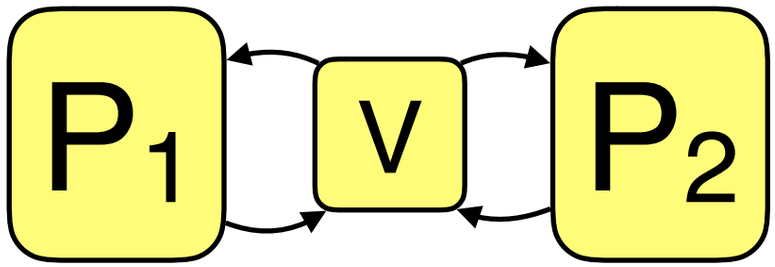
Tradeoffs

Tradeoffs

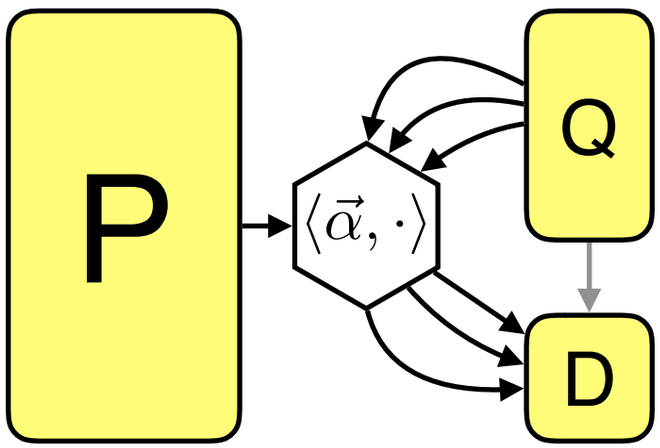
PCP



MIP



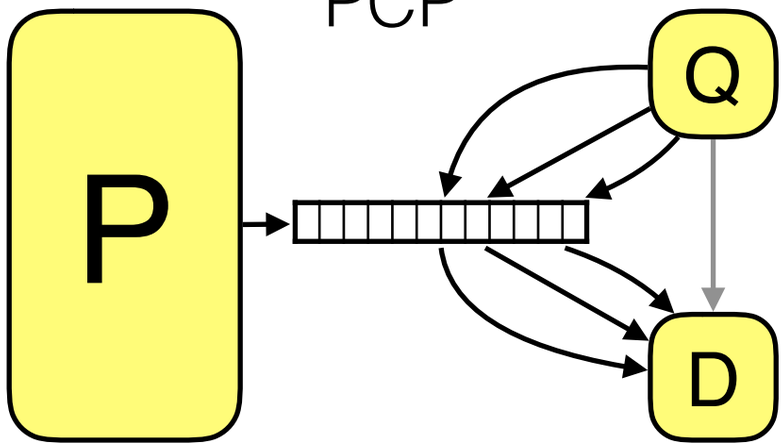
Linear PCP



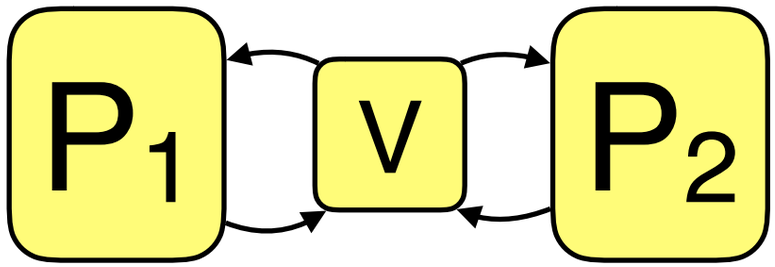
...

Tradeoffs

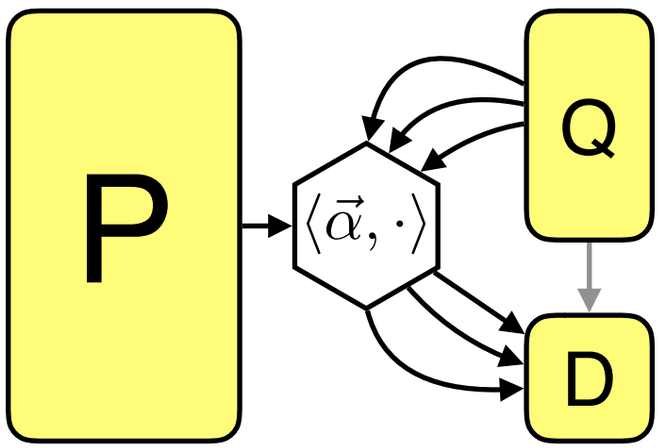
PCP



MIP



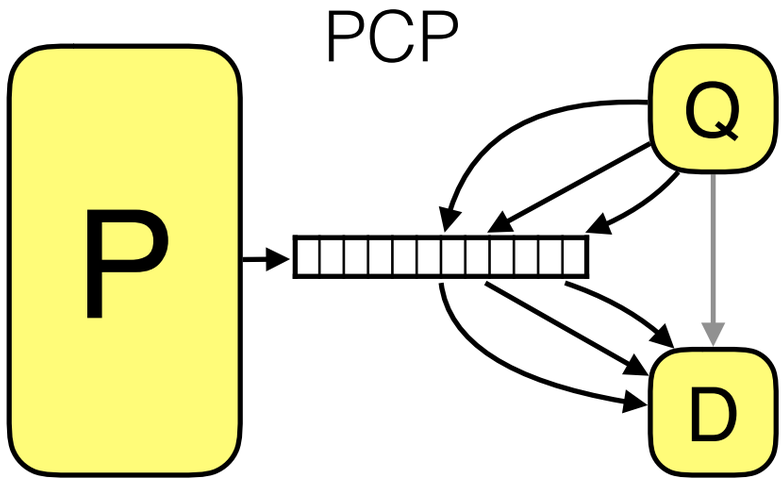
Linear PCP

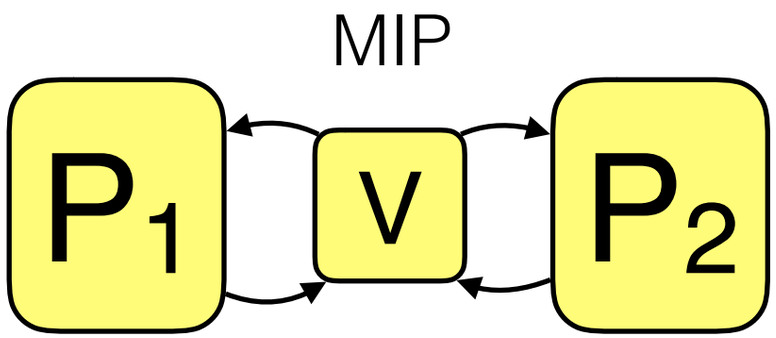


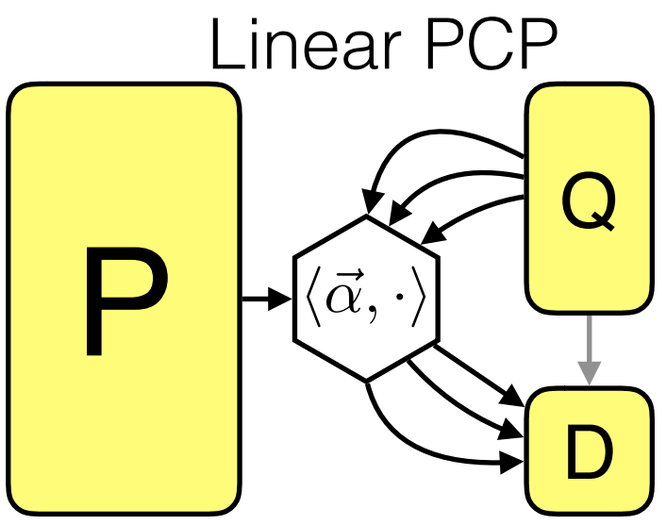
succinct arguments

...

Tradeoffs







...

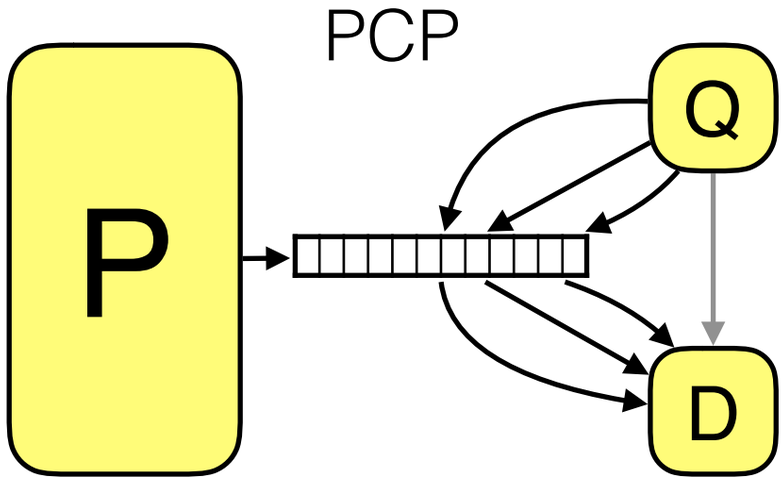
public verifiability

symmetric-key crypto

bottleneck is PCP

succinct arguments

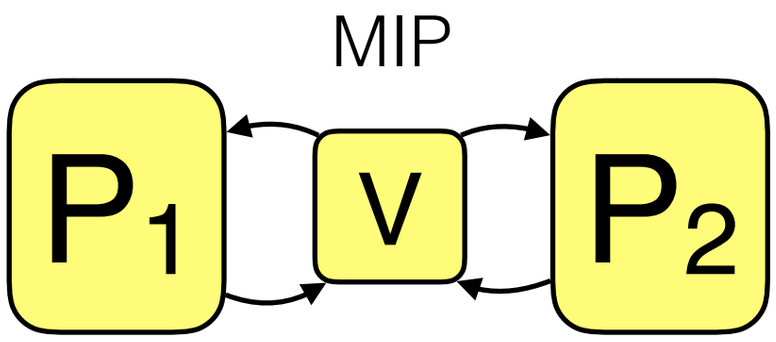
Tradeoffs



public verifiability

symmetric-key crypto

bottleneck is PCP

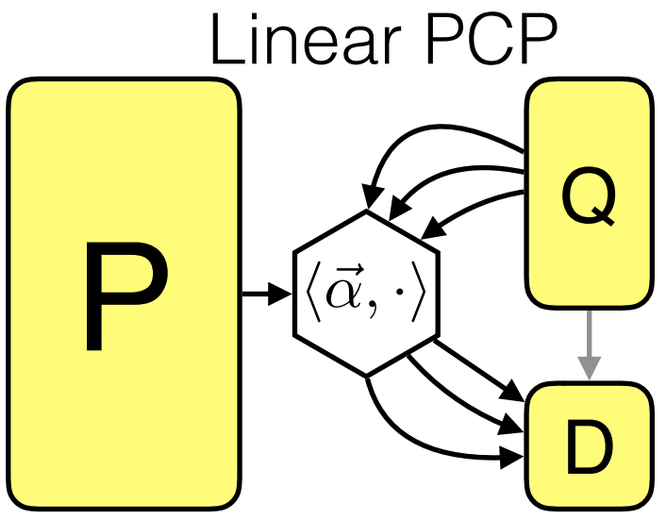


succinct arguments

private verifiability

public-key crypto

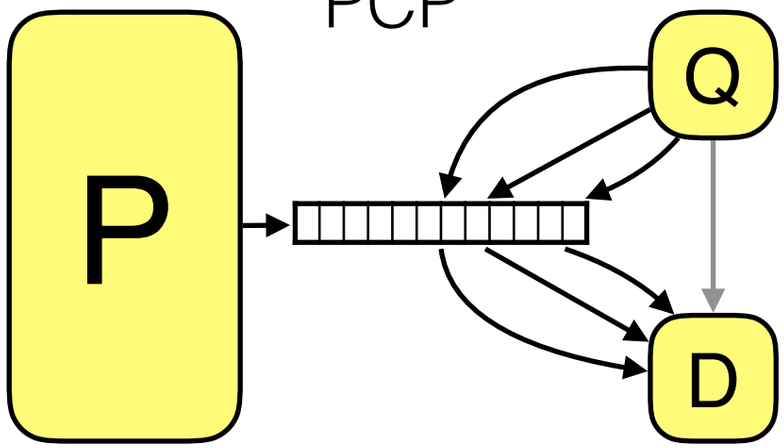
bottleneck is MIP



...

Tradeoffs

PCP

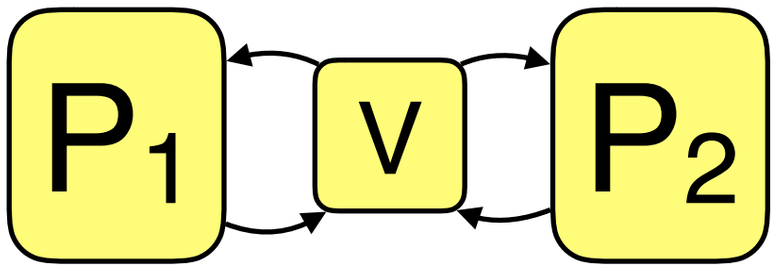


public verifiability

symmetric-key crypto

bottleneck is PCP

MIP



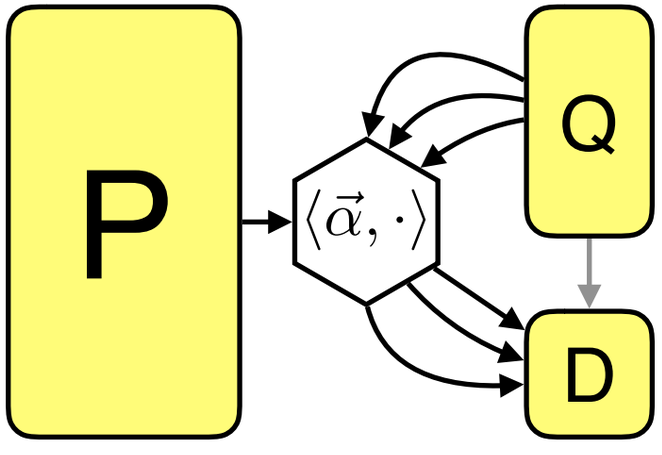
succinct arguments

private verifiability

public-key crypto

bottleneck is MIP

Linear PCP



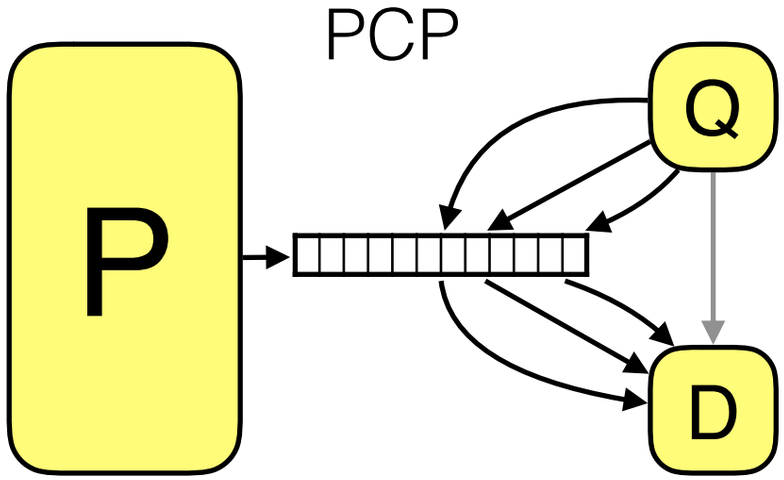
public/private verifiability

public-key crypto

quite efficient!!

...

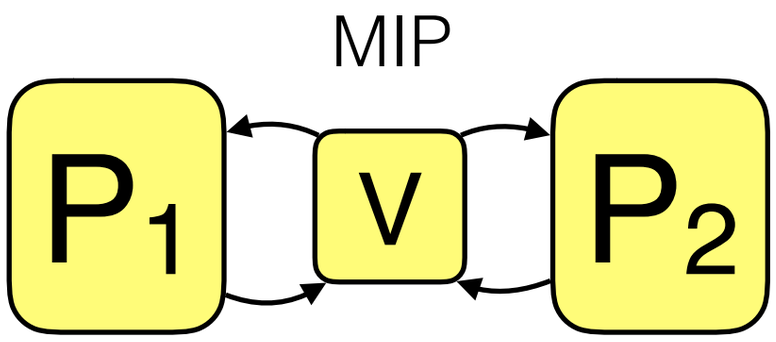
Tradeoffs



public verifiability

symmetric-key crypto

bottleneck is PCP

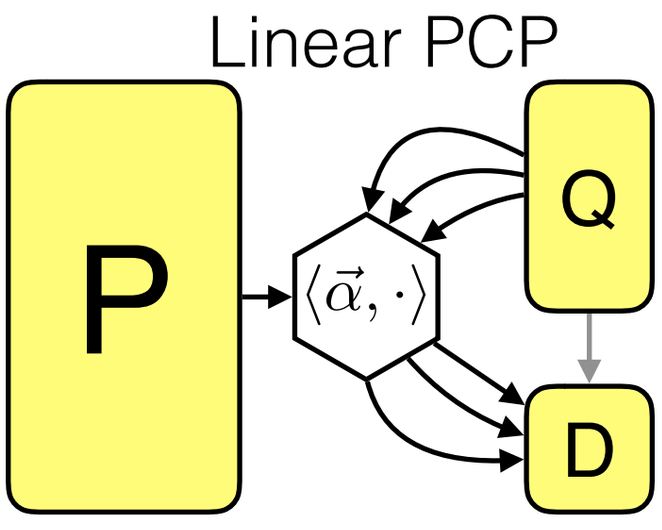


succinct arguments

private verifiability

public-key crypto

bottleneck is MIP



public/private verifiability

public-key crypto

quite efficient!!

|proof| ~ 128 bytes

|verifier| ~ 3ms

libsnark.org

...

Frontiers

Interactive Oracle Proofs

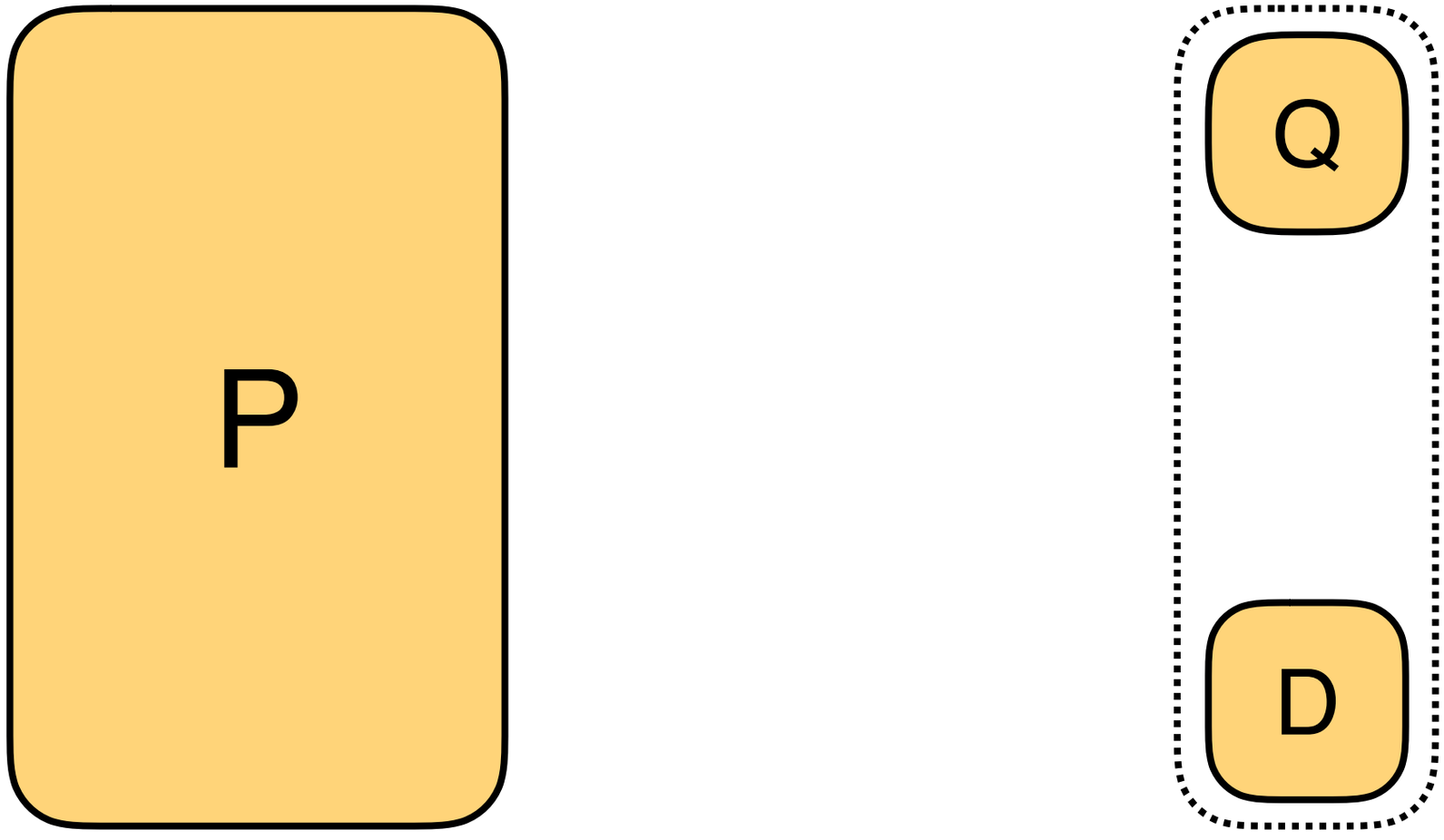
[BCS16][RRR16]

Probabilistically Checkable Interactive Proofs

Interactive Oracle Proofs

[BCS16][RRR16]

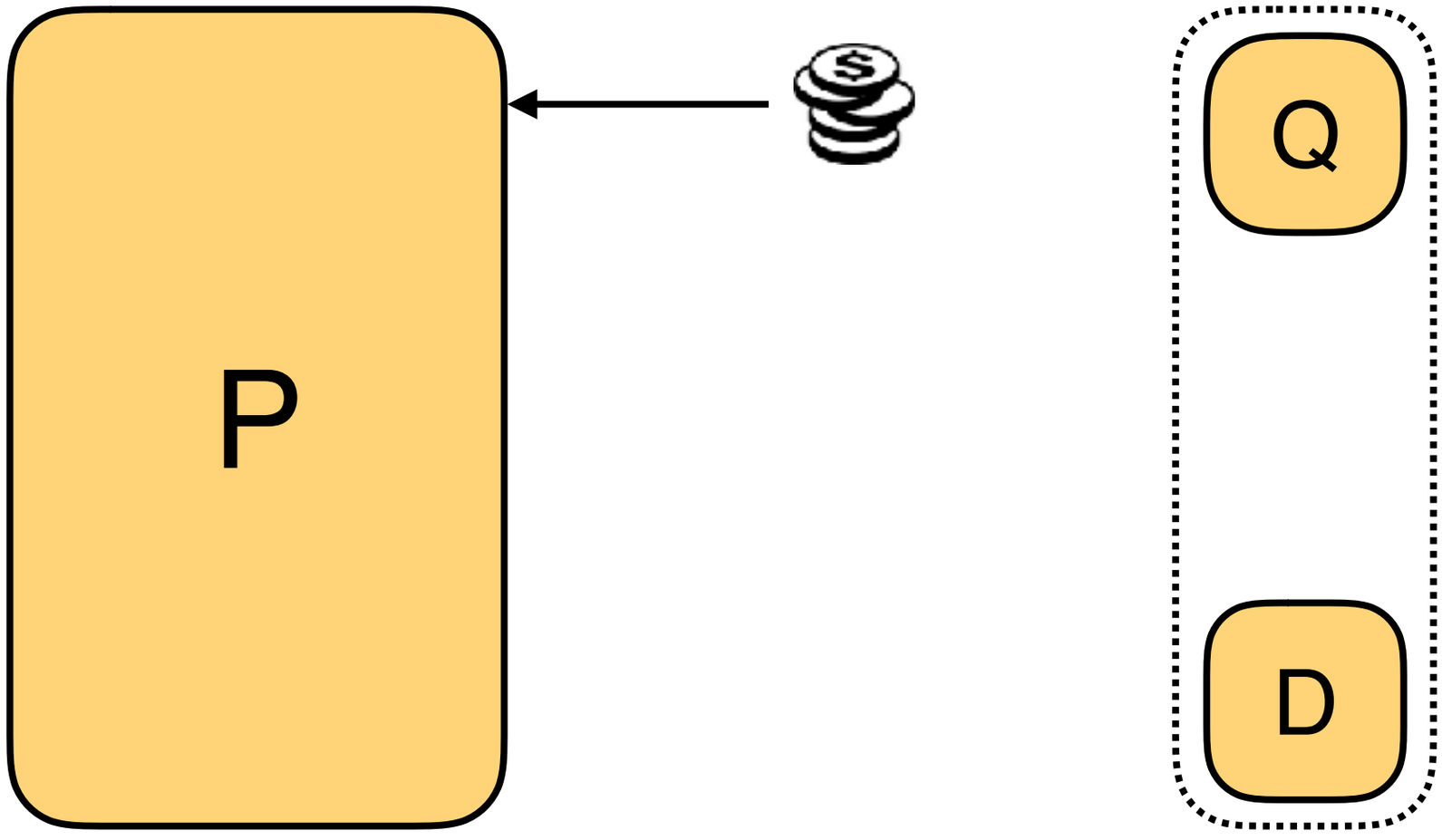
Probabilistically Checkable Interactive Proofs



Interactive Oracle Proofs

[BCS16][RRR16]

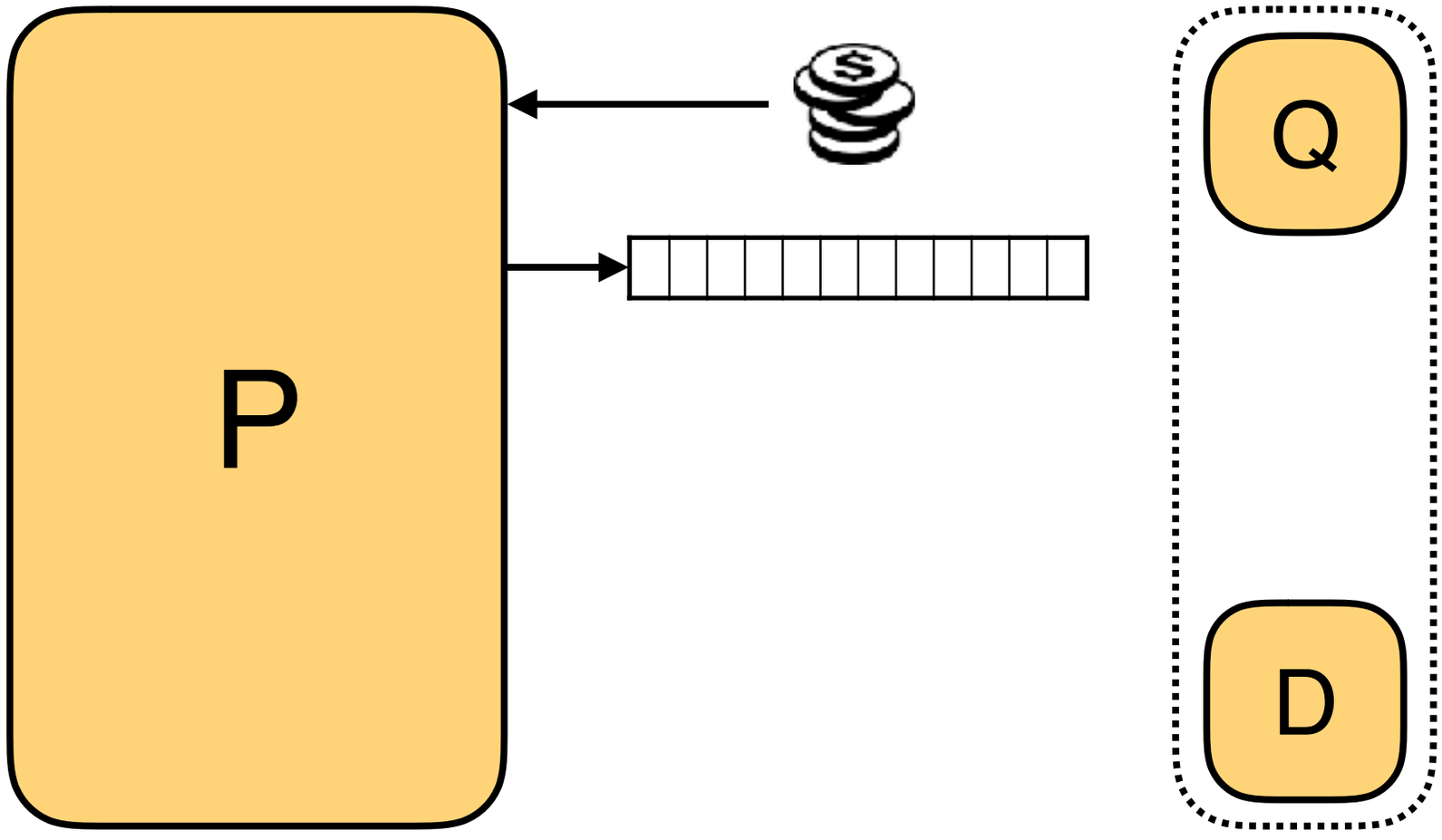
Probabilistically Checkable Interactive Proofs



Interactive Oracle Proofs

[BCS16][RRR16]

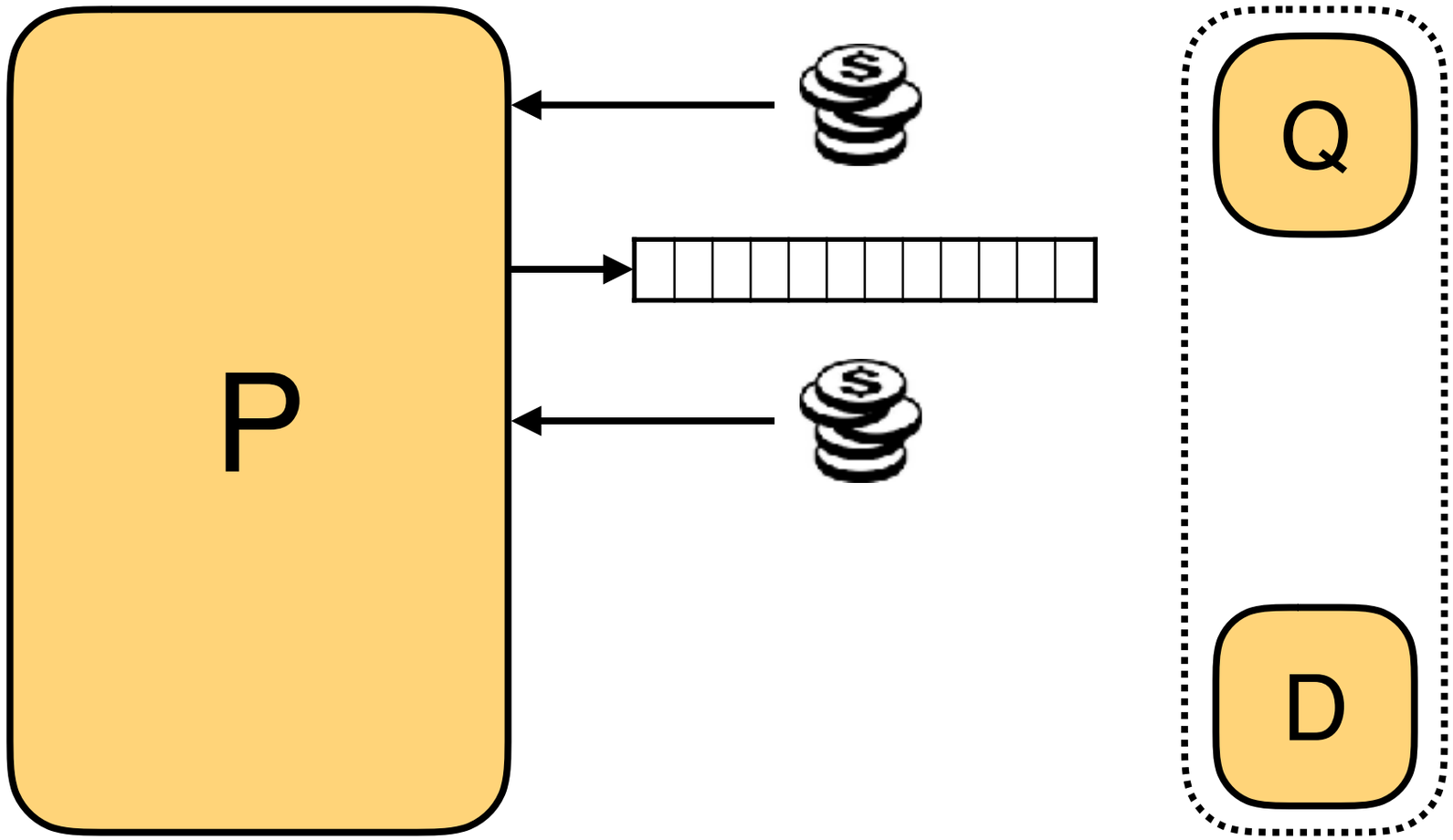
Probabilistically Checkable Interactive Proofs



Interactive Oracle Proofs

[BCS16][RRR16]

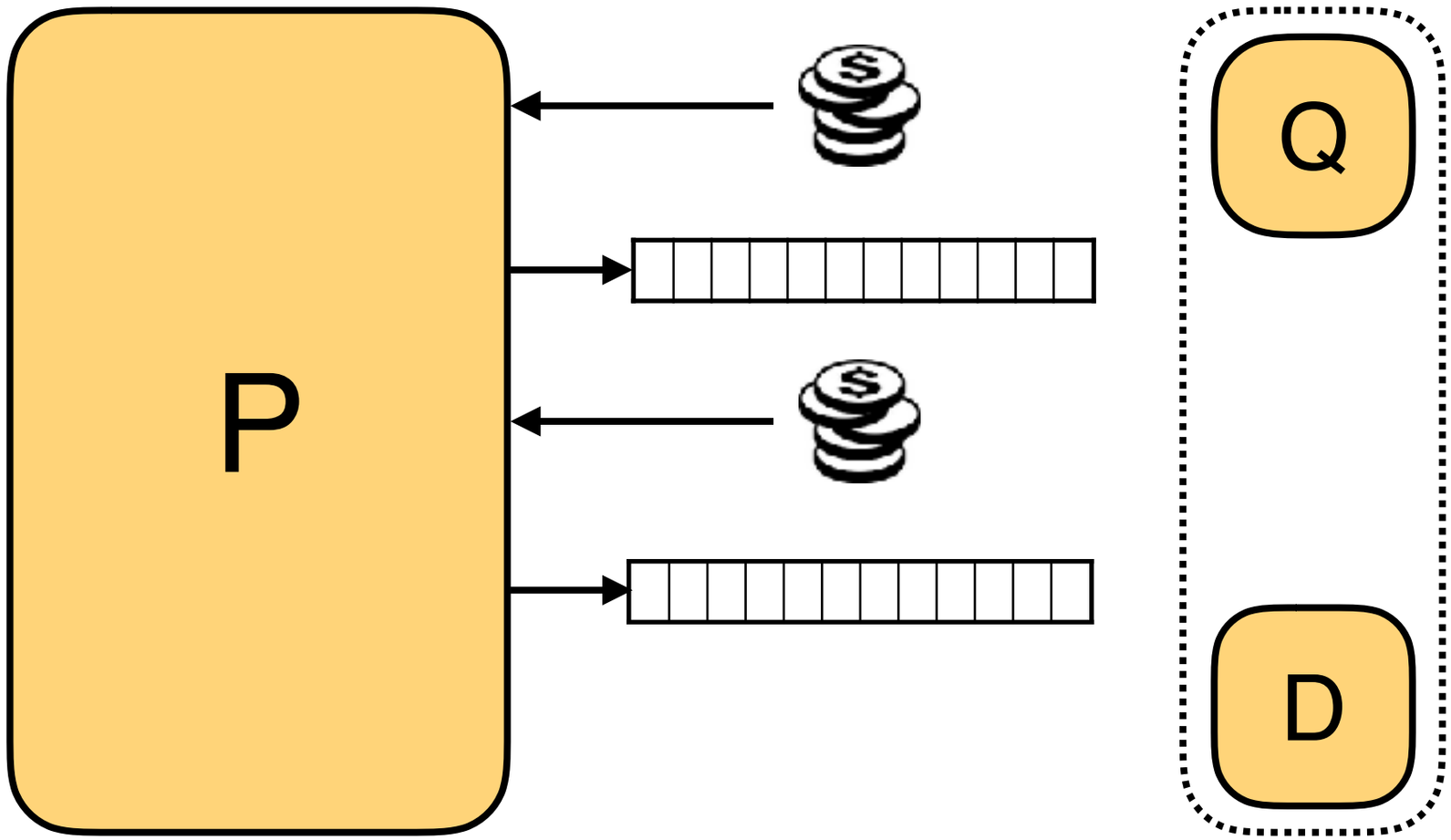
Probabilistically Checkable Interactive Proofs



Interactive Oracle Proofs

[BCS16][RRR16]

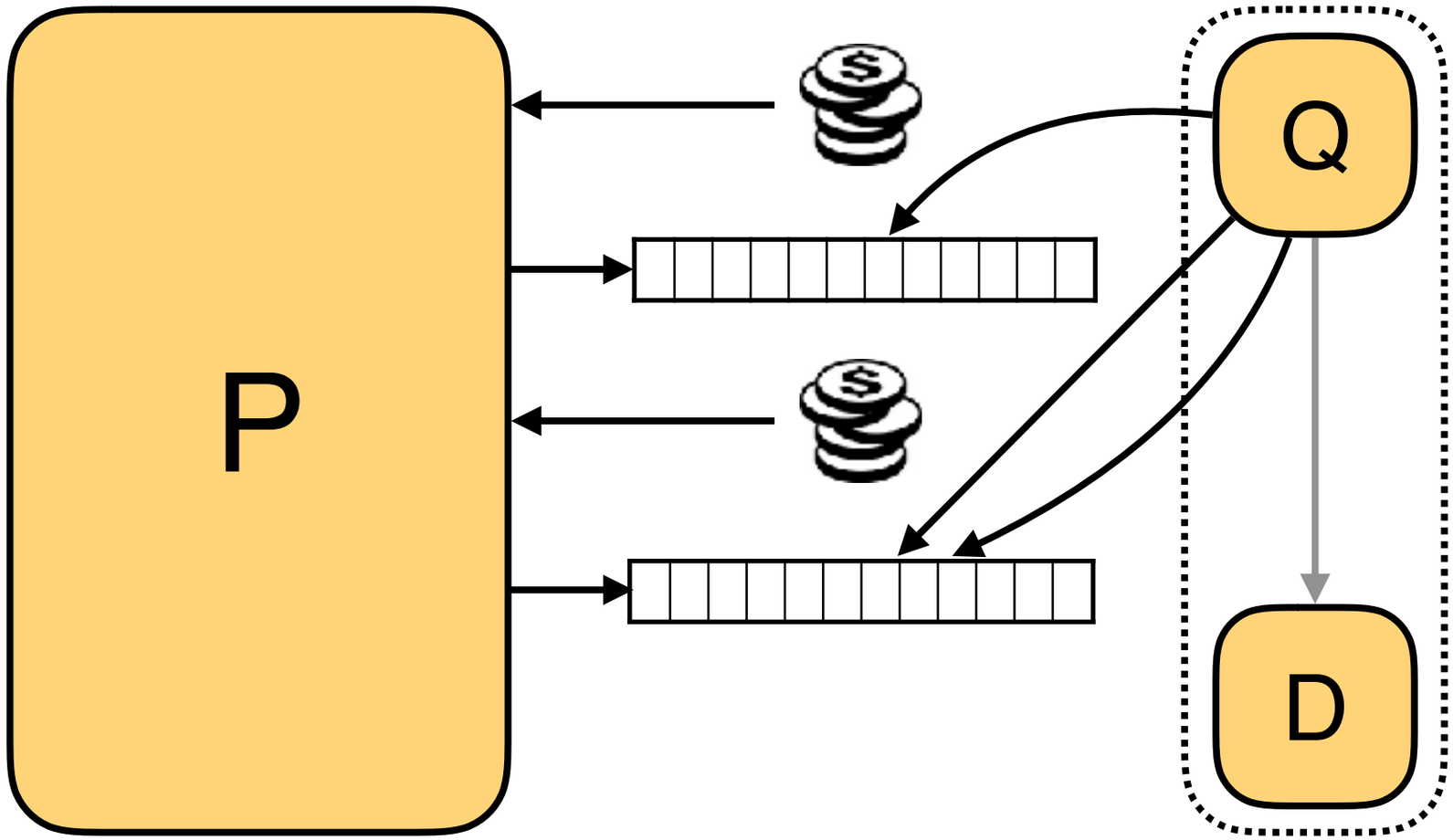
Probabilistically Checkable Interactive Proofs



Interactive Oracle Proofs

[BCS16][RRR16]

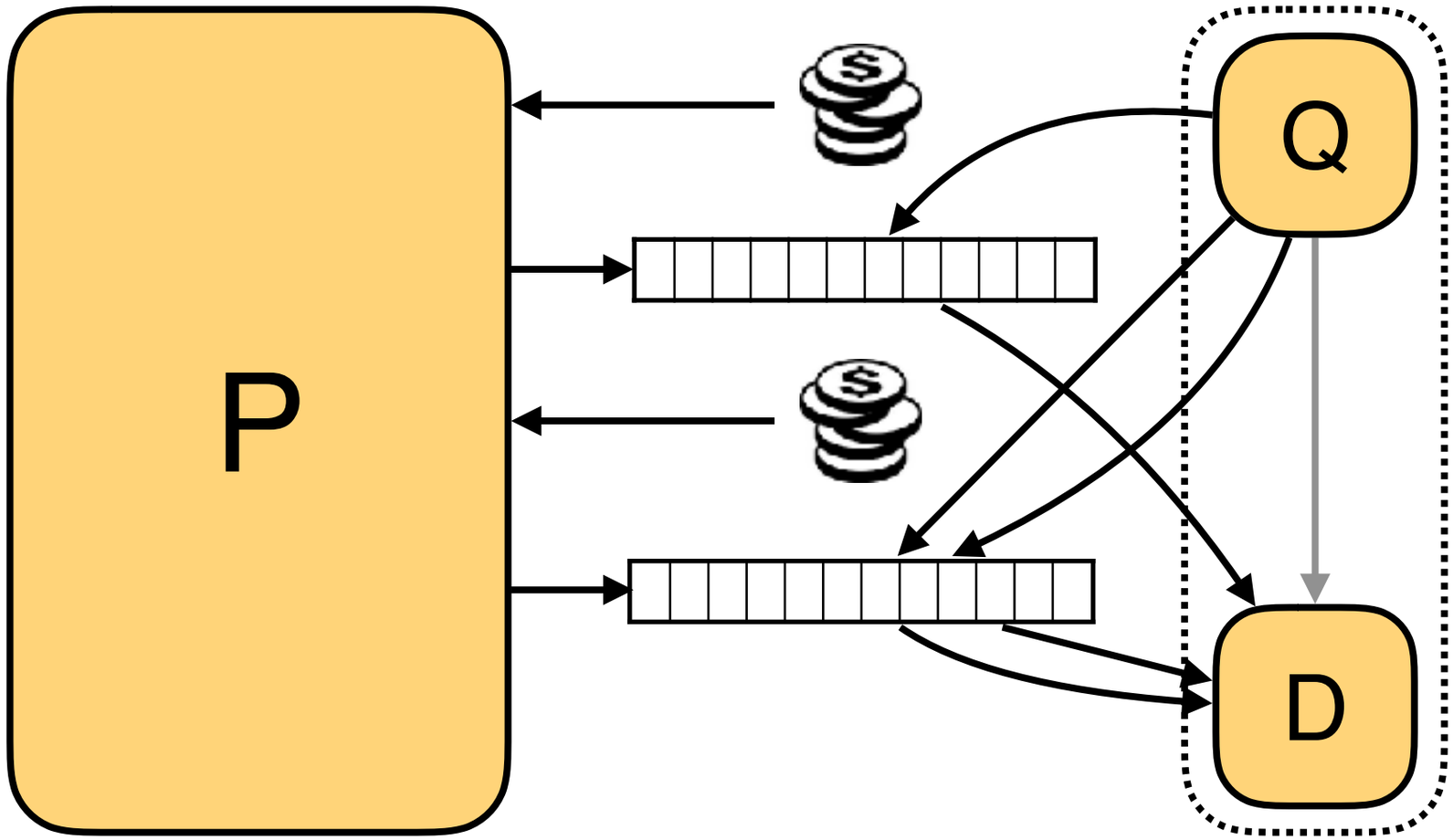
Probabilistically Checkable Interactive Proofs



Interactive Oracle Proofs

[BCS16][RRR16]

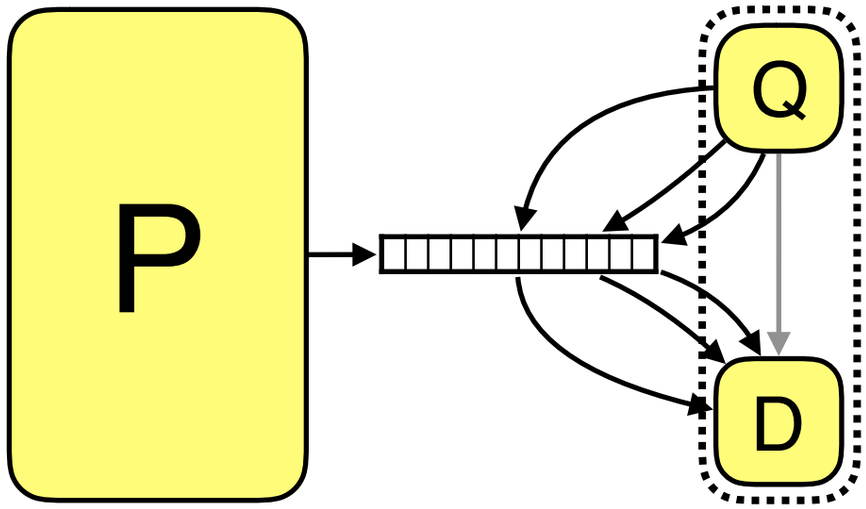
Probabilistically Checkable Interactive Proofs



The verifier can simultaneously leverage randomness, interaction, and probabilistic checking.

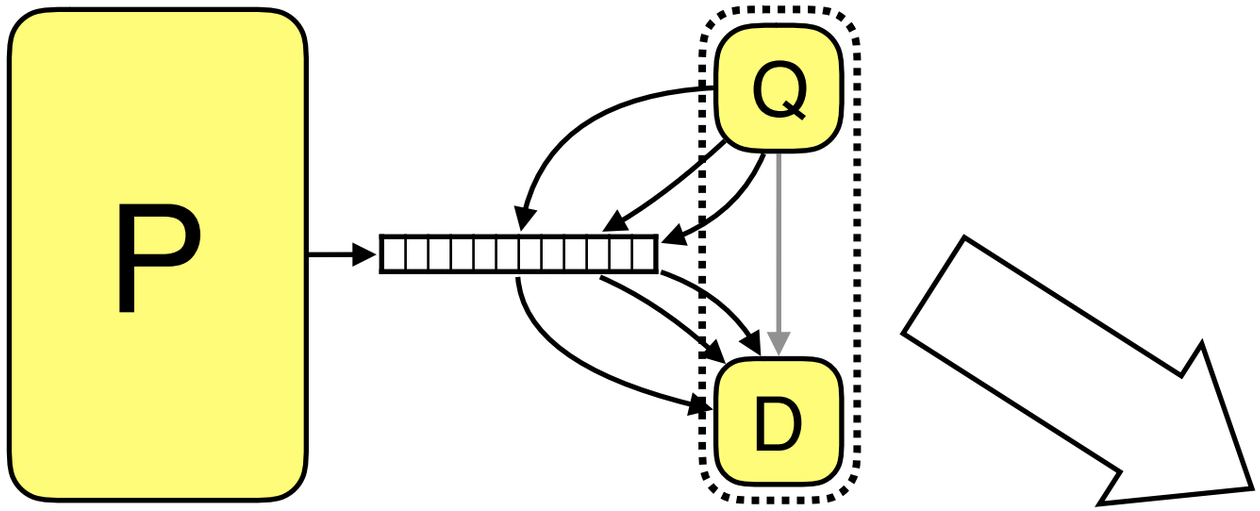
Succinct Arguments From IOPs

Probabilistically Checkable Proof



Succinct Arguments From IOPs

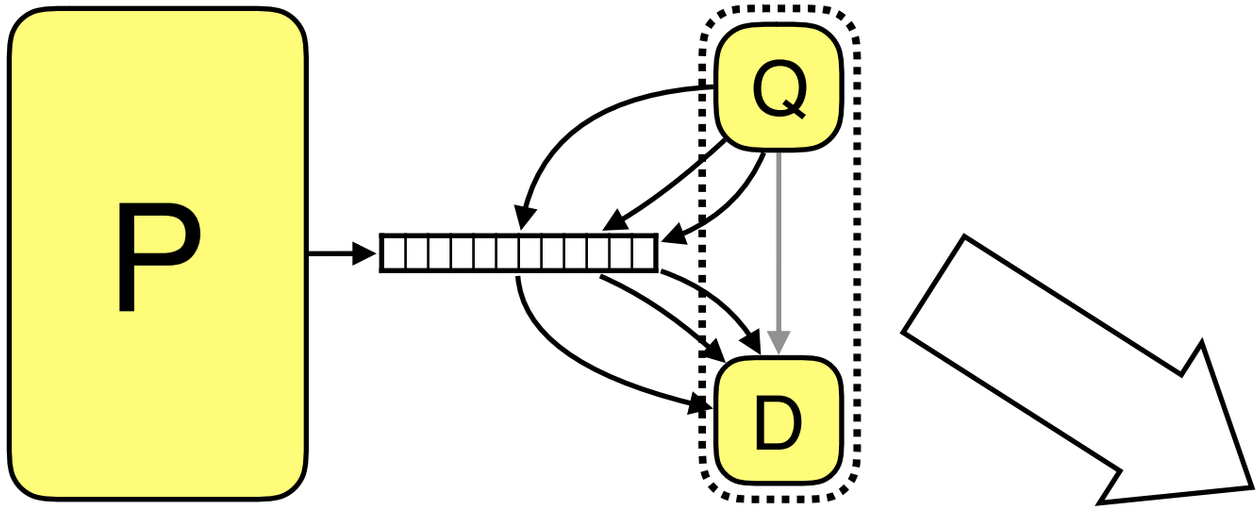
Probabilistically Checkable Proof



Succinct Argument

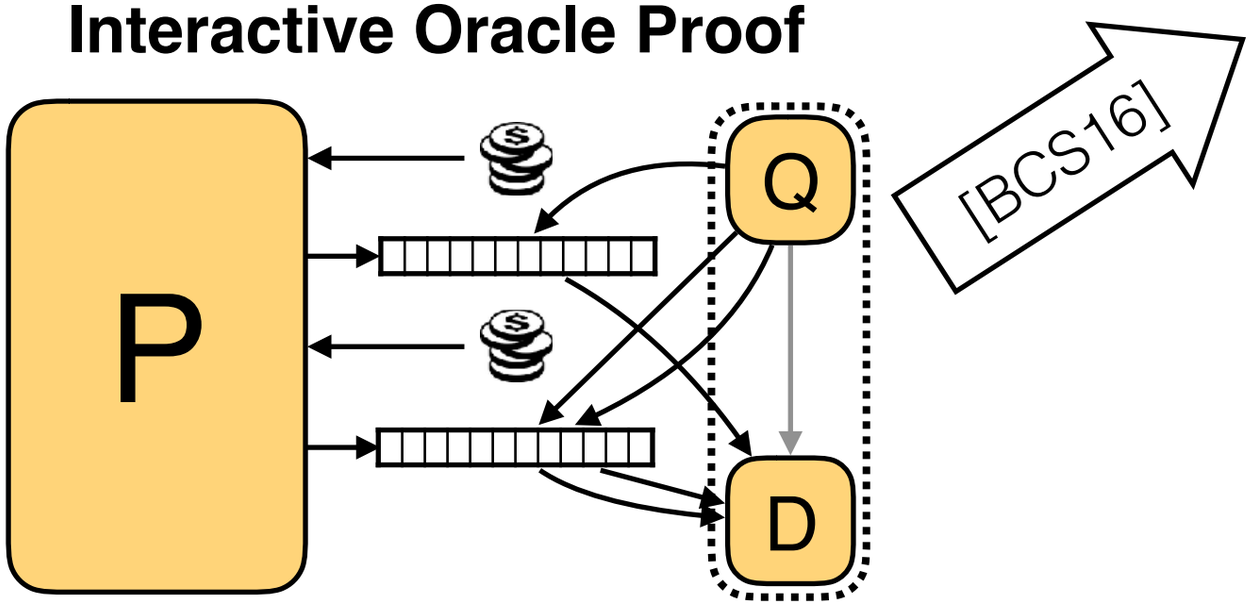
Succinct Arguments From IOPs

Probabilistically Checkable Proof



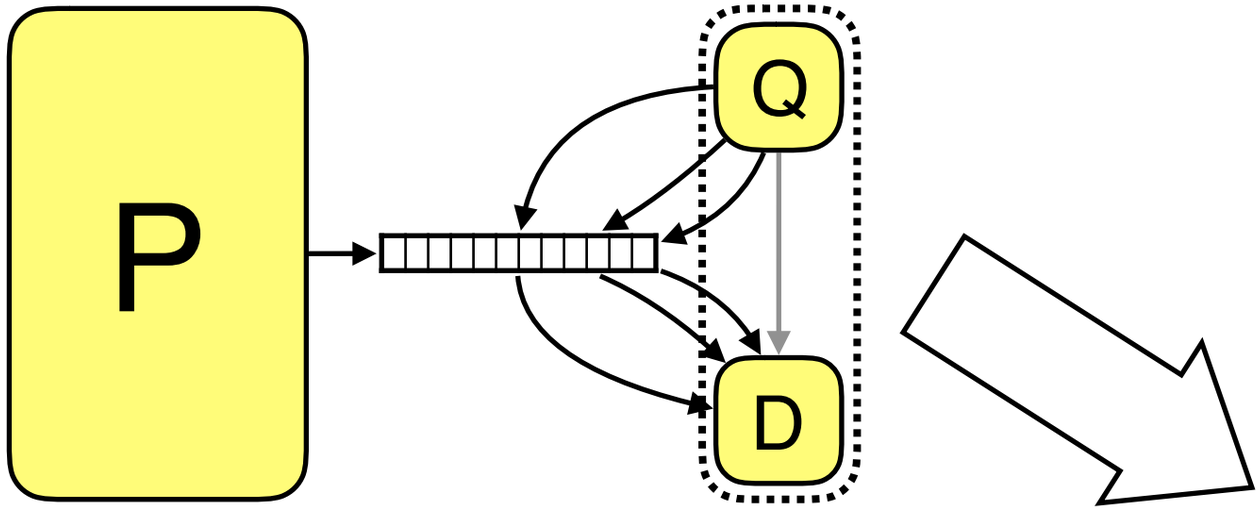
Succinct Argument

Interactive Oracle Proof



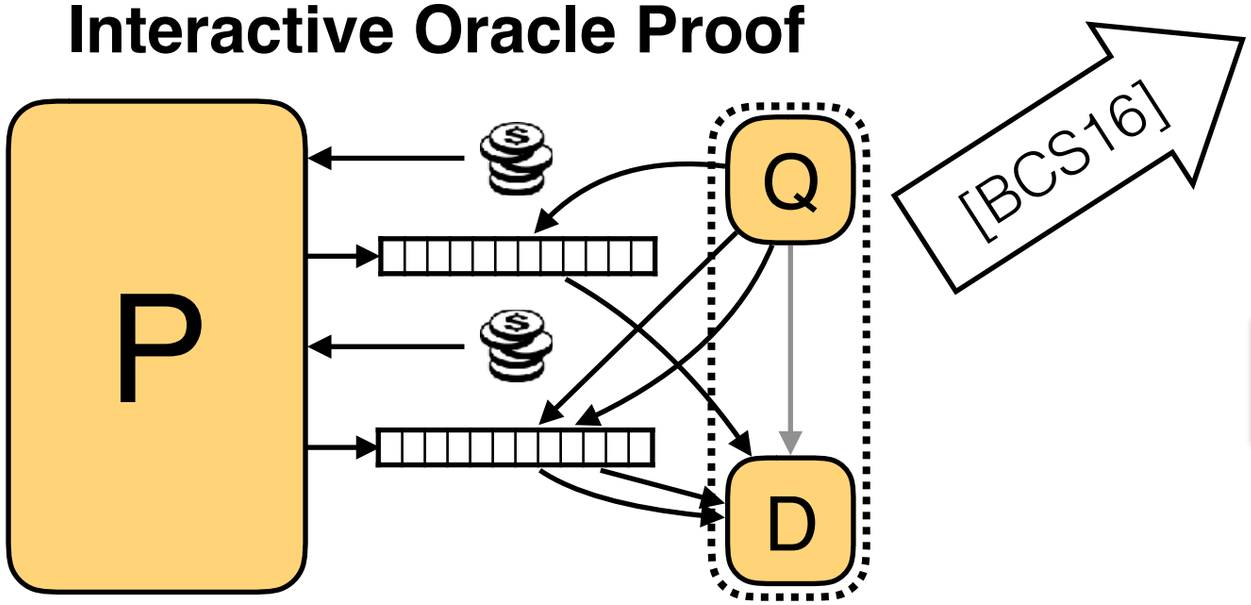
Succinct Arguments From IOPs

Probabilistically Checkable Proof



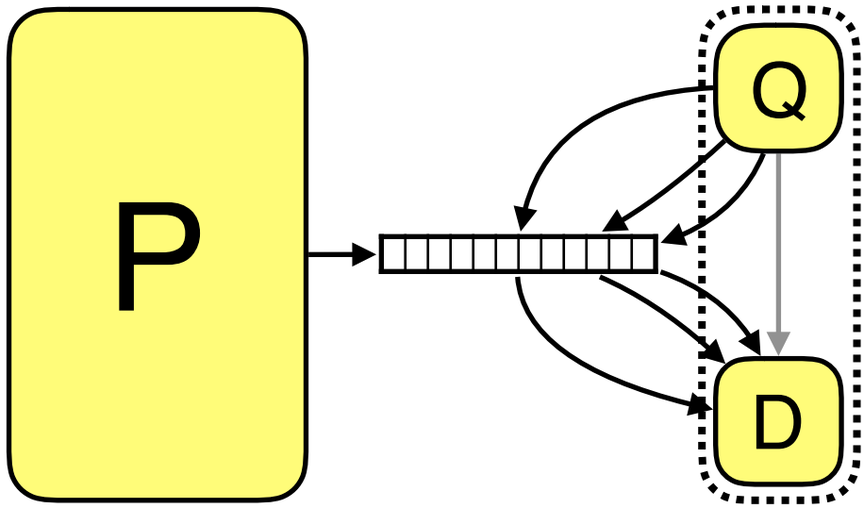
Succinct Argument

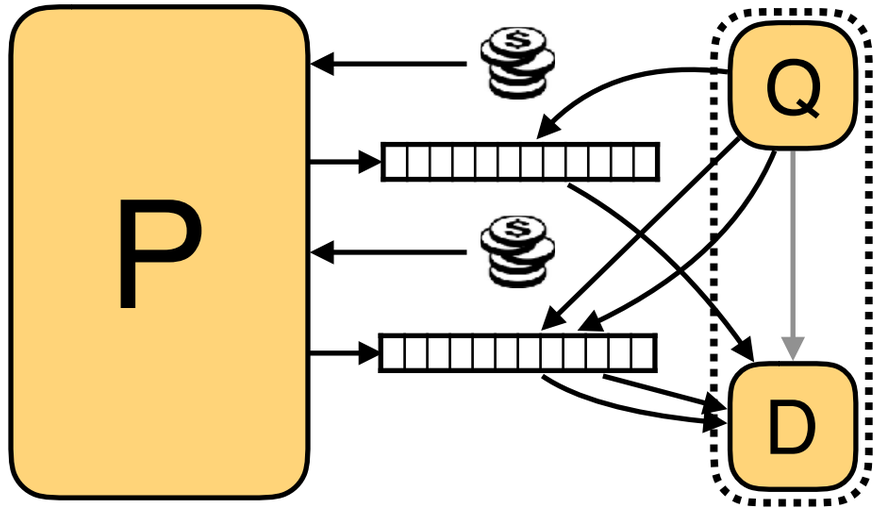
Interactive Oracle Proof



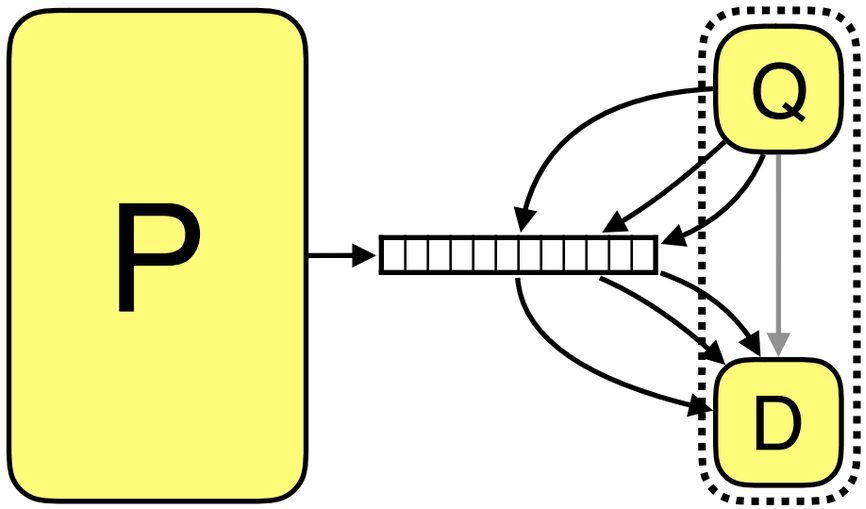
Q: any efficiency gains?

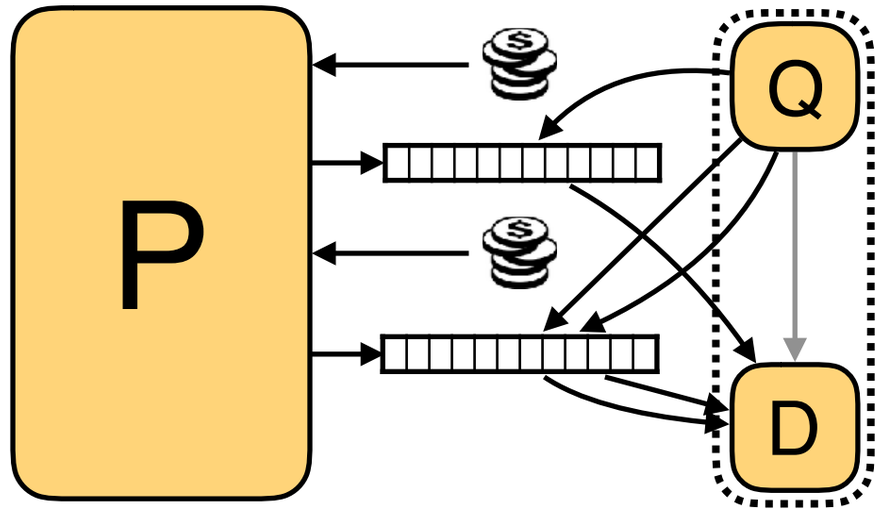
PCPs vs IOPs





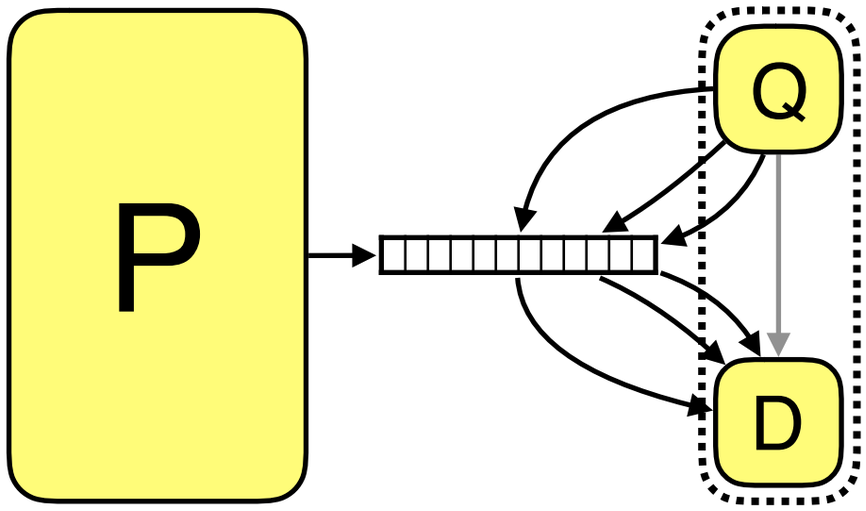
PCPs vs IOPs

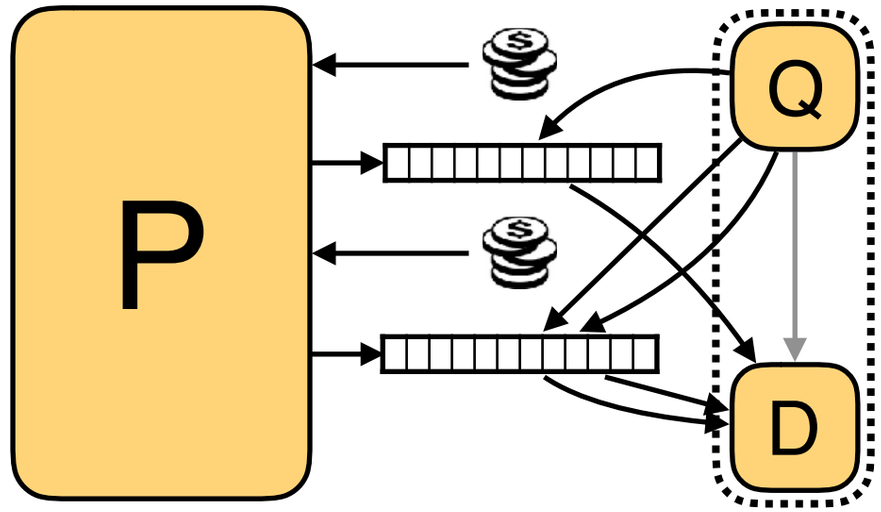




Best proof length vs query complexity tradeoff?

PCPs vs IOPs



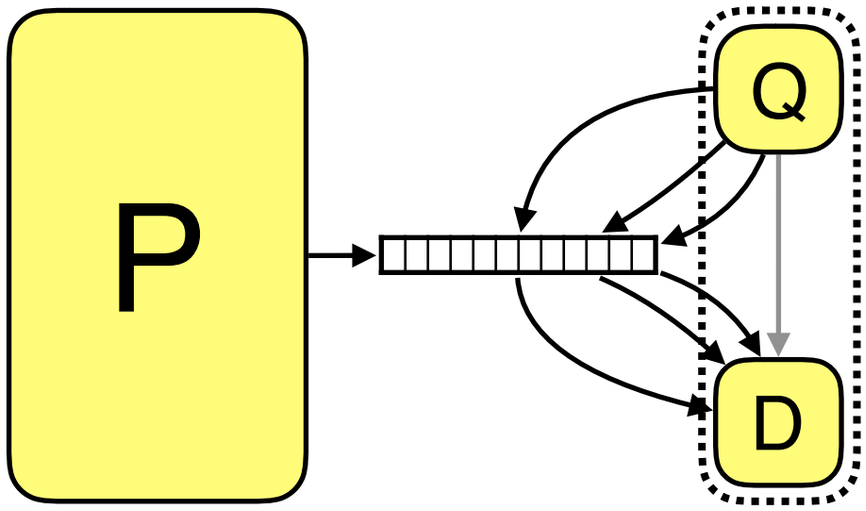


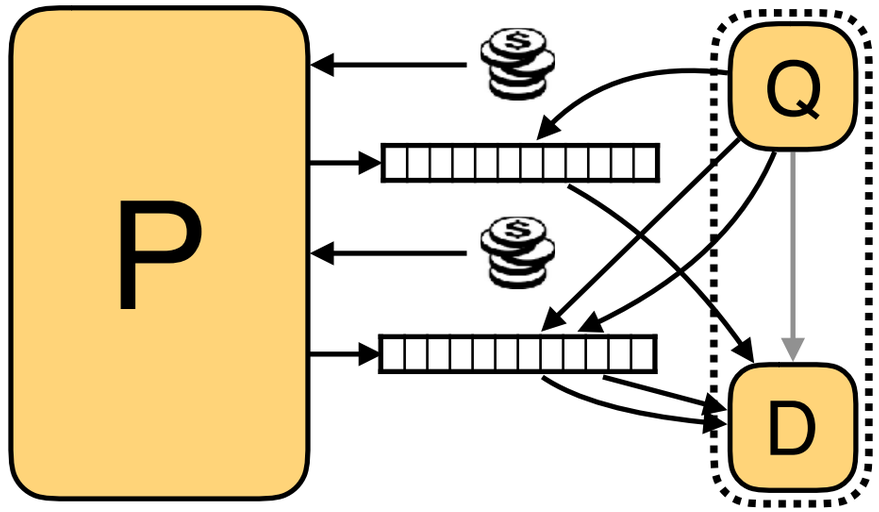
Best proof length vs query complexity tradeoff?

Thm [BS08][Din07][Mie09]

Every language in NTIME(T) has PCPs
with proof length $T \log(T)^{O(1)}$
and query complexity $O(1)$.

PCPs vs IOPs





Best proof length vs query complexity tradeoff?

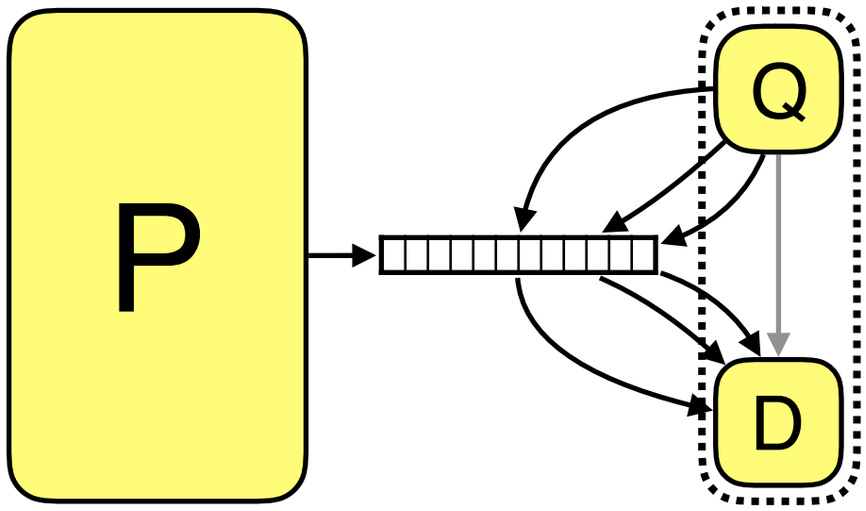
Thm [BS08][Din07][Mie09]

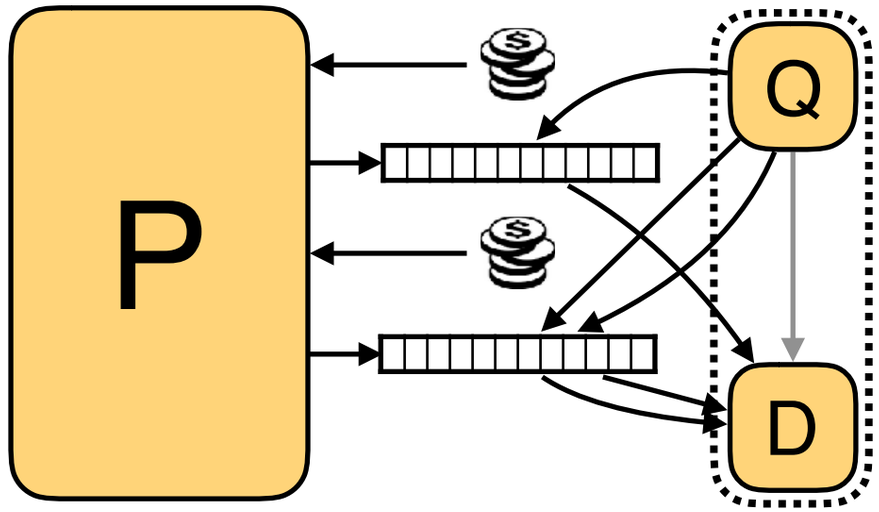
Every language in NTIME(T) has PCPs
with proof length $T \log(T)^{O(1)}$
and query complexity $O(1)$.

Thm [BKKMS13]

$\forall\ \varepsilon>0$ circuit SAT has (non-uniform) PCPs
with proof length $2^{O(1/\varepsilon)}$
and query complexity n^{ε}.

PCPs vs IOPs





Best proof length vs query complexity tradeoff?

Thm [BS08][Din07][Mie09]

Every language in NTIME(T) has PCPs
with proof length $T \log(T)^{O(1)}$
and query complexity $O(1)$.

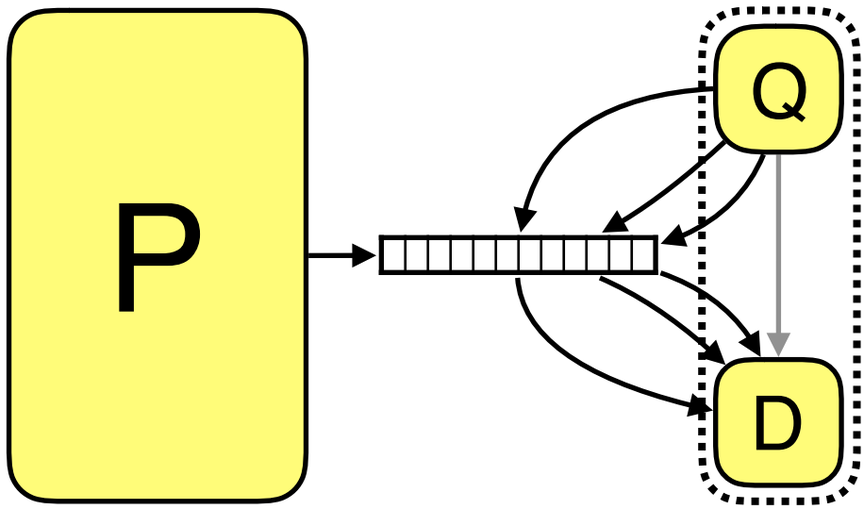
Thm [BKKMS13]

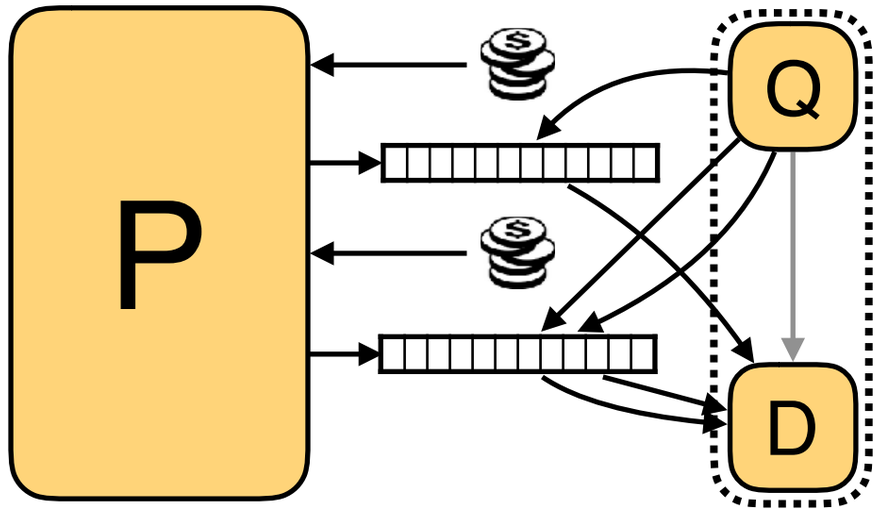
$\forall\ \varepsilon>0$ circuit SAT has (non-uniform) PCPs
with proof length $2^{O(1/\varepsilon)}$
and query complexity n^{ε}.

Thm [BCGRS16]

circuit SAT has 3-round IOPs
with proof length $O(n)$
and query complexity $O(1)$.

PCPs vs IOPs





Best proof length vs query complexity tradeoff?

Thm [BS08][Din07][Mie09]

Every language in NTIME(T) has PCPs
with proof length $T \log(T)^{O(1)}$
and query complexity $O(1)$.

Thm [BKKMS13]

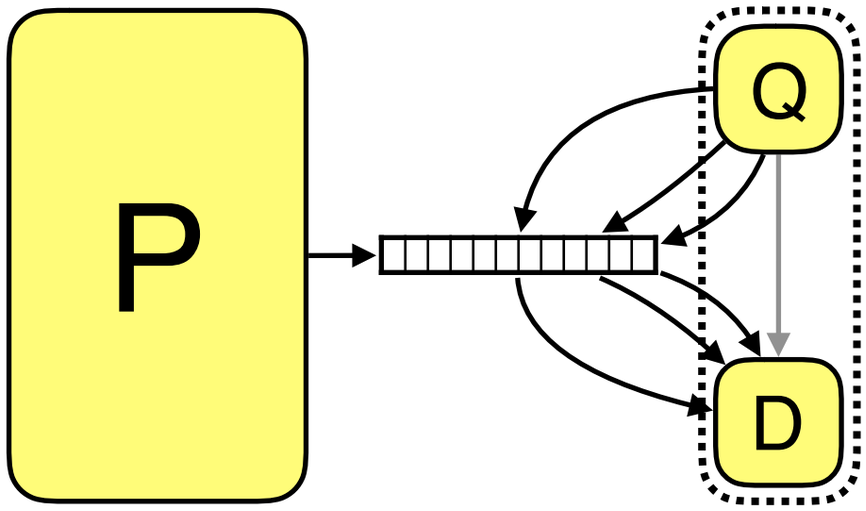
$\forall\ \varepsilon>0$ circuit SAT has (non-uniform) PCPs
with proof length $2^{O(1/\varepsilon)}$
and query complexity n^{ε}.

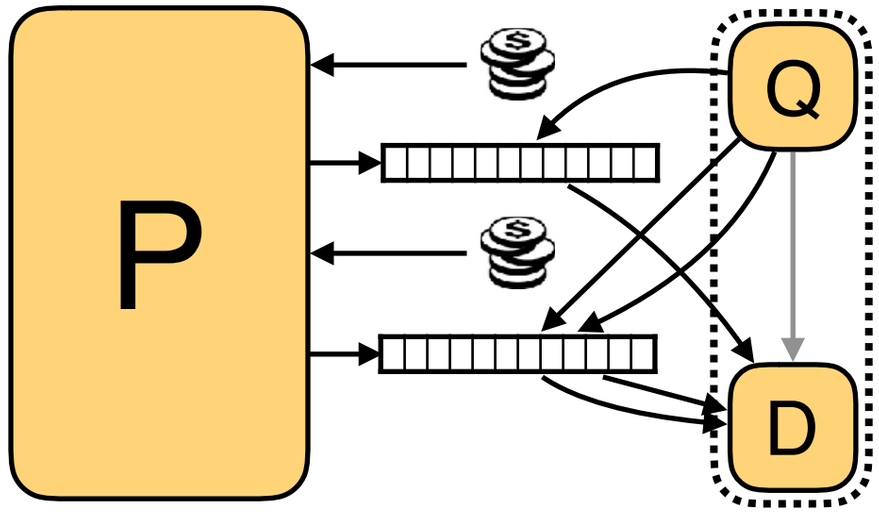
Thm [BCGRS16]

circuit SAT has 3-round IOPs
with proof length $O(n)$
and query complexity $O(1)$.

Q1: optimal round complexity?

PCPs vs IOPs





Best proof length vs query complexity tradeoff?

Thm [BS08][Din07][Mie09]

Every language in NTIME(T) has PCPs
with proof length $T \log(T)^{O(1)}$
and query complexity $O(1)$.

Thm [BKKMS13]

$\forall\ \varepsilon>0$ circuit SAT has (non-uniform) PCPs
with proof length $2^{O(1/\varepsilon)}$
and query complexity n^{ε}.

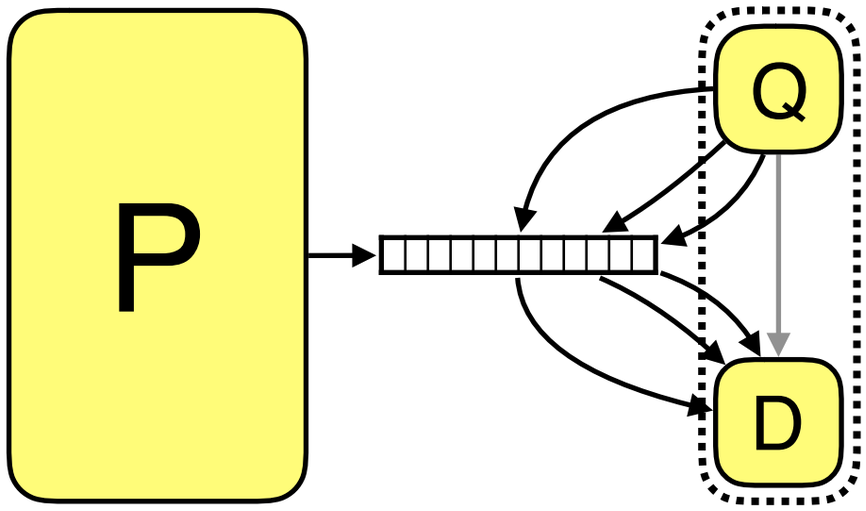
Thm [BCGRS16]

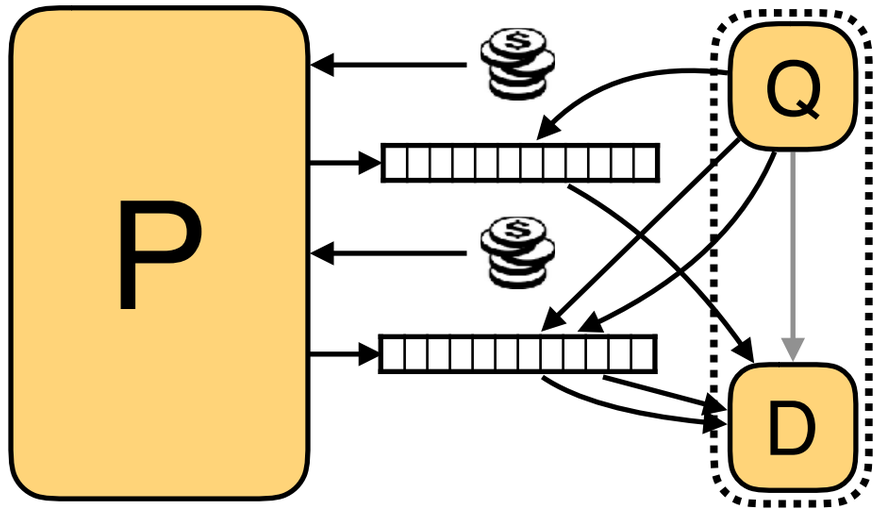
circuit SAT has 3-round IOPs
with proof length $O(n)$
and query complexity $O(1)$.

Q1: optimal round complexity?

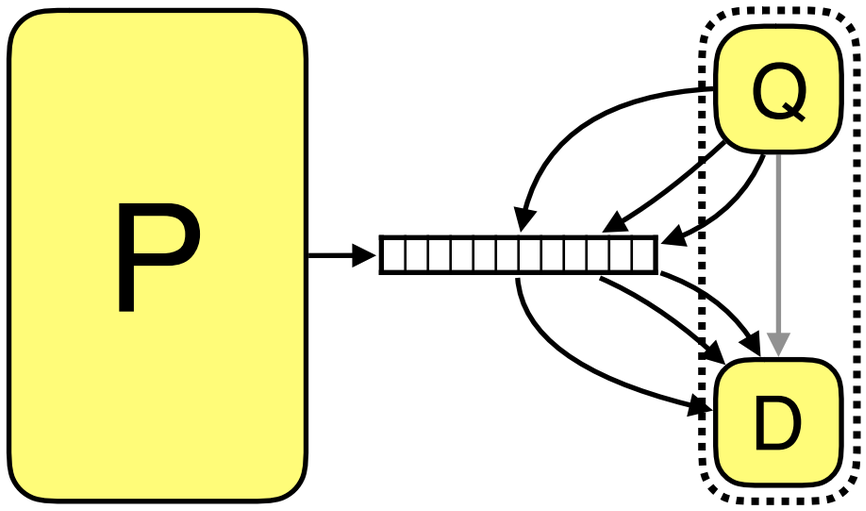
Q2: analogs for NEXP?
(succinct circuit SAT)

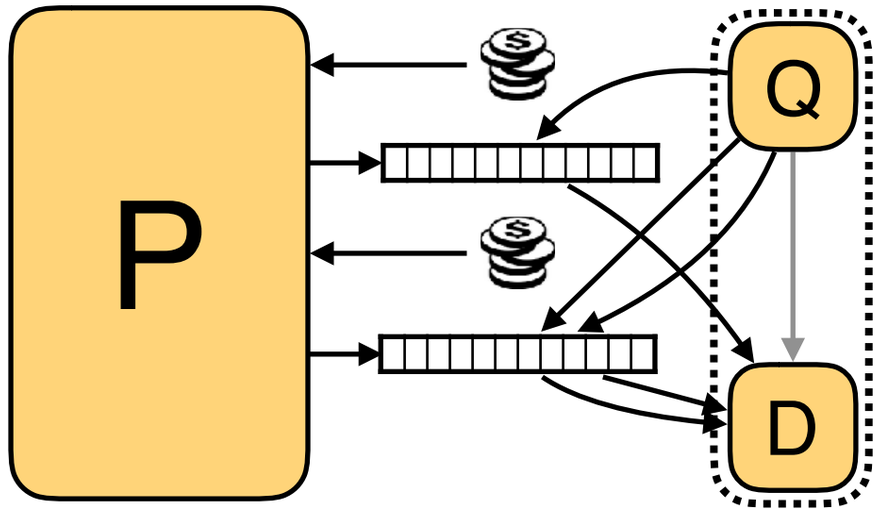
PCPs vs IOPs





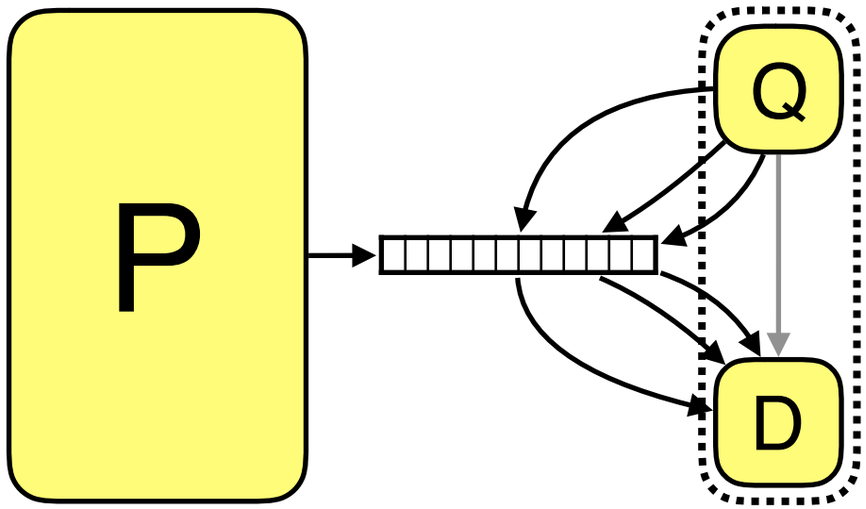
PCPs vs IOPs

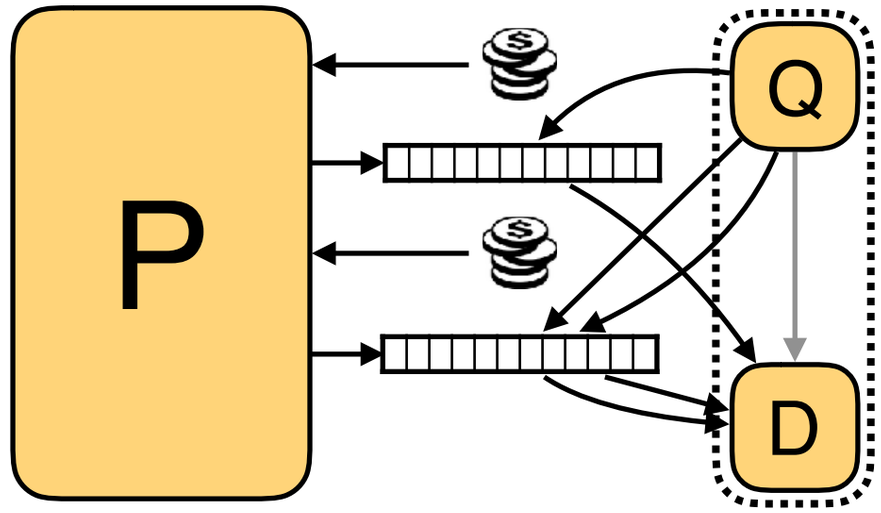




Applications to hardness of approximation?

PCPs vs IOPs





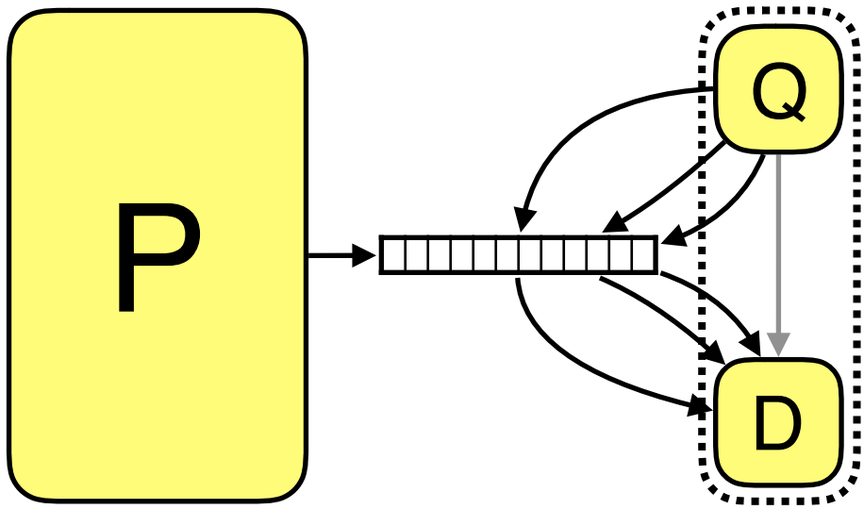
Applications to hardness of approximation?

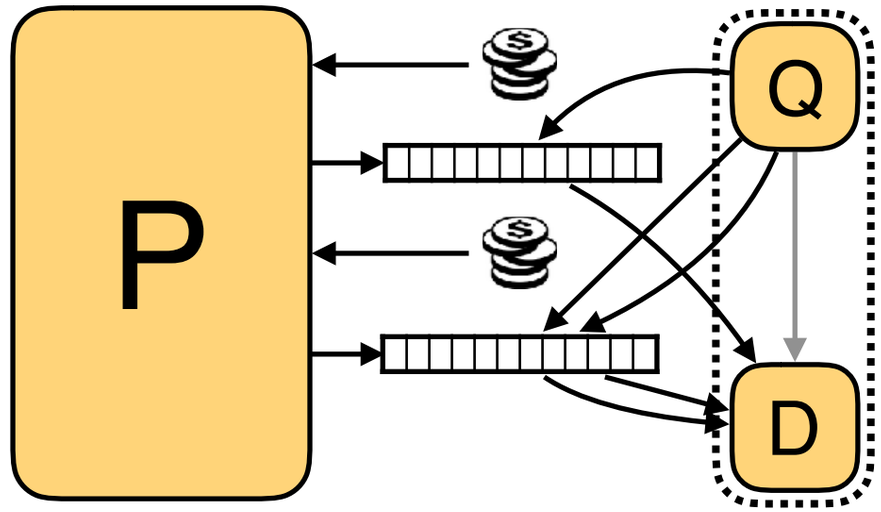
$PCP_{c,s}[r,q]_{\Sigma}$

$\updownarrow$

(c,s)-promise problem
about a q-ary CSP over Σ

PCPs vs IOPs





Applications to hardness of approximation?

$PCP_{c,s}[r,q]_{\Sigma}$

↕

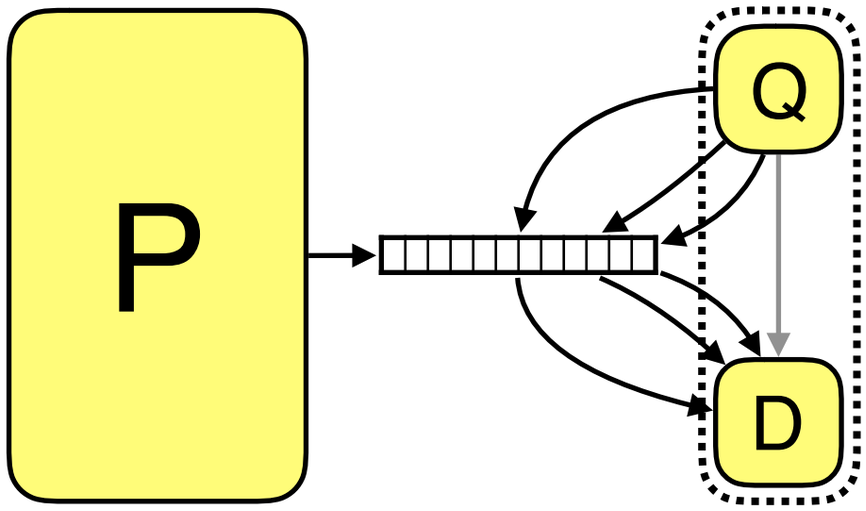
(c,s)-promise problem
about a q-ary CSP over Σ

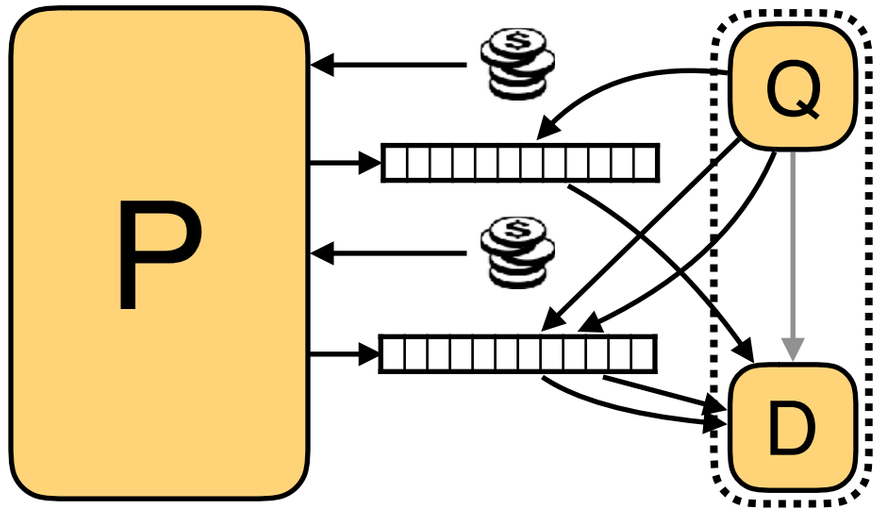
PCPs for NP-hard languages

↕

hardness of approximation for NP

PCPs vs IOPs





Applications to hardness of approximation?

$PCP_{c,s}[r,q]_{\Sigma}$

↕

(c,s)-promise problem
about a q-ary CSP over Σ

PCPs for NP-hard languages

↕

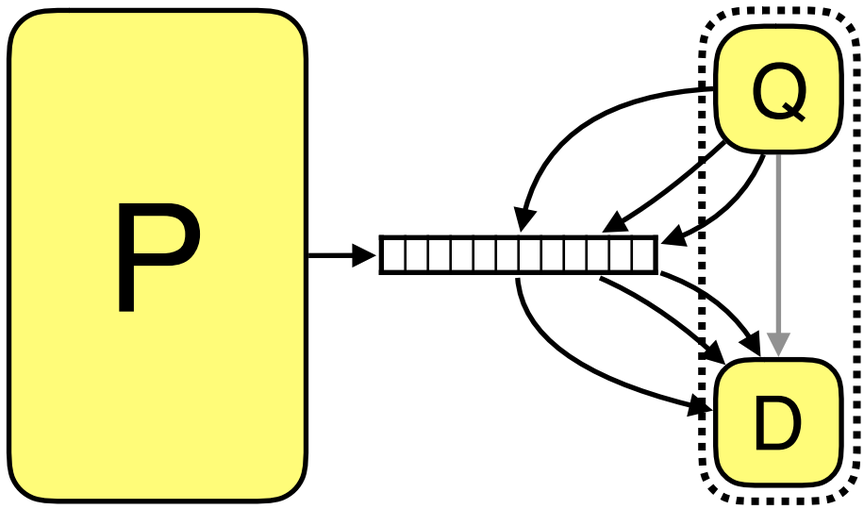
hardness of approximation for NP

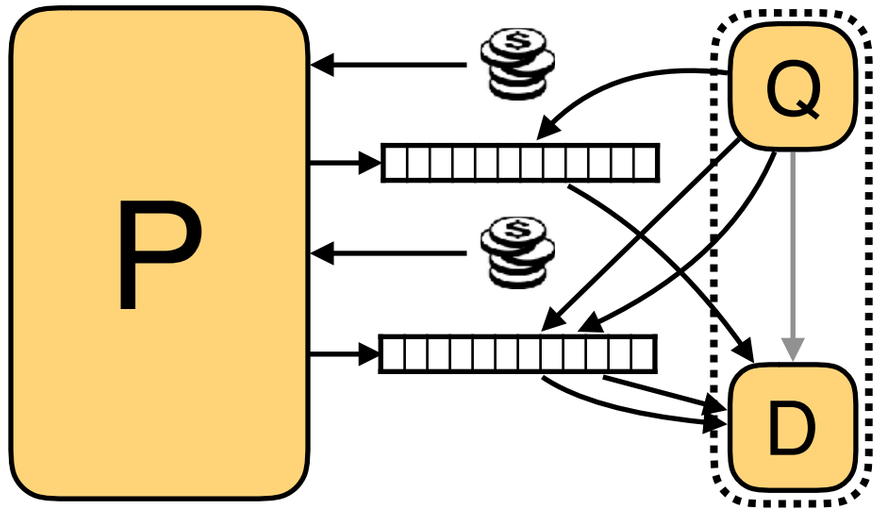
$IOP_{c,s}[k,r,q]_{\Sigma}$

↕

???

PCPs vs IOPs





Applications to hardness of approximation?

$PCP_{c,s}[r,q]_{\Sigma}$

↕

(c,s)-promise problem
about a q-ary CSP over Σ

PCPs for NP-hard languages

↕

hardness of approximation for NP

$IOP_{c,s}[k,r,q]_{\Sigma}$

↕

???

Q: does progress on the theory of IOPs imply any interesting hardness of approximation results?

Thanks!