Online/Offline OR Composition of ∑-Protocols

Michele Ciampi
DIEM
Università di Salerno
ITALY

Giuseppe Persiano
DISA-MIS
Università di Salerno
ITALY

Alessandra Scafuro
Boston University and
Northeastern University
USA

Luisa Siniscalchi
DIEM
Università di Salerno
ITALY

Ivan Visconti
DIEM
Università di Salerno
ITALY

Proofs of Knowledge (PoKs)

A fundamental crypto tool with many applications

- Identification Schemes
- Simulation-Based Security
- E-Voting Systems
- …

Useful in cryptography when the witness is protected: Witness Indistinguishable (WI), Witness Hiding (WH), Zero Knowledge (ZK)

e.g., prove knowledge of one thing OR another thing OR …

Proofs of Knowledge (PoKs)

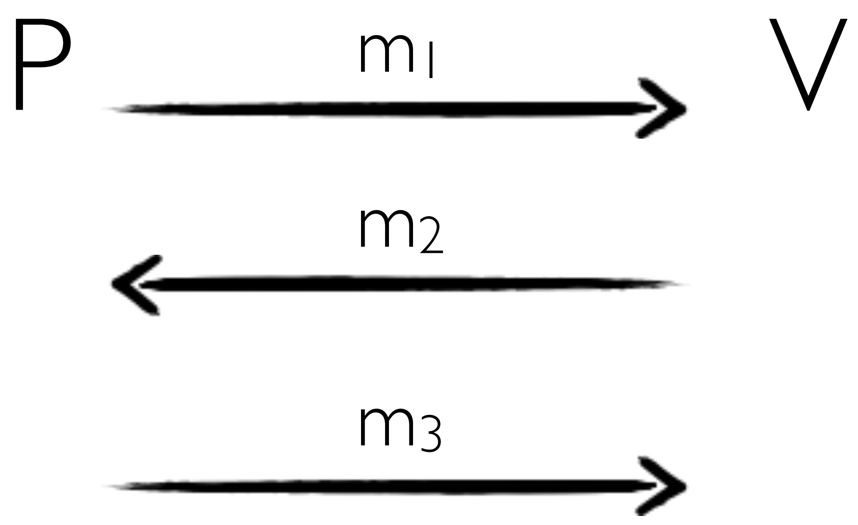
In theory

$x \in L$

$x \xrightarrow{\text{NP-reduction}} G$

"(G, C) in R_{HAM}"

WI Proof of Knowledge of Hamiltonicity
[Blum86, **L**apidot**S**hamir**90**]



In practice

Proofs of Knowledge (PoKs)

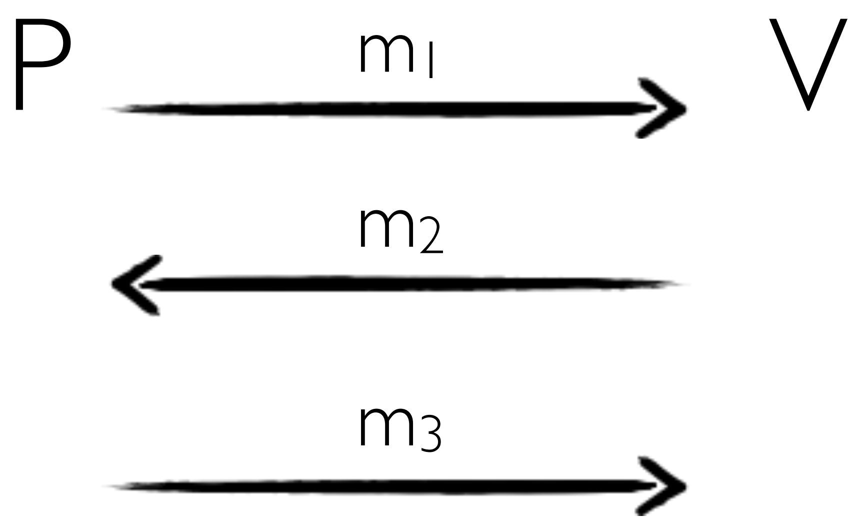
In theory

$x \in L$

$x \xrightarrow{\text{NP-reduction}} G$

"(G, C) in $\mathbf{R}_{HAM}$"

WI Proof of Knowledge of Hamiltonicity
[Blum86, **L**apidot**S**hamir**90**]

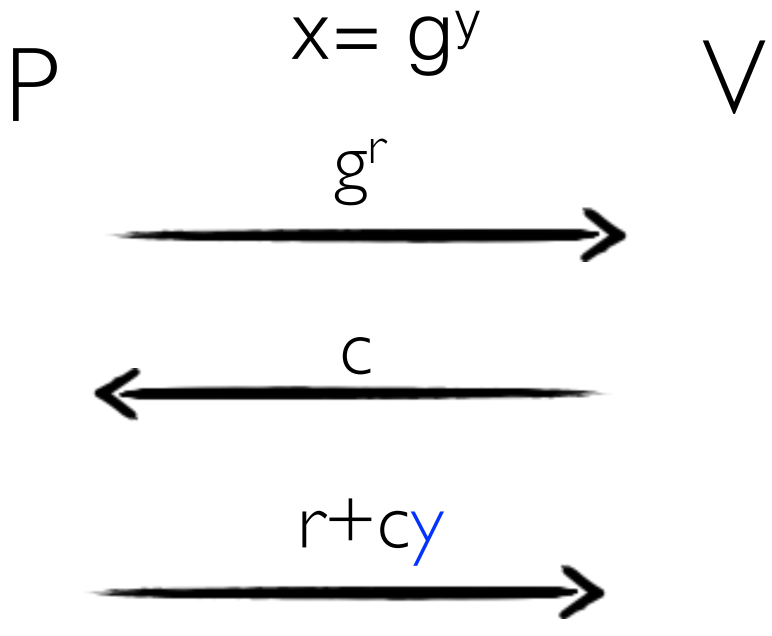


In practice

∑-protocol for $\mathbf{R}$

"(x, y) in $\mathbf{R}_{Dlog}$"

e.g. Discret Log [Schnorr89])



Proofs of K ledge (PoKs)

Observation: [LS90] and [Schnorr89] need the theorem and witness only in the last round

In theor

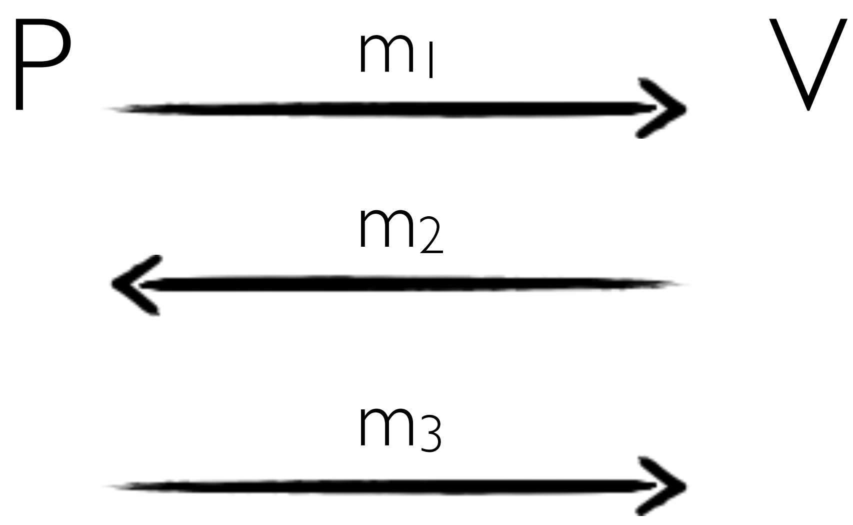
actice

x∈L

$x \xrightarrow{\text{NP-reduction}} G$

“(G, C) in $\mathbf{R}_{\text{HAM}}$”

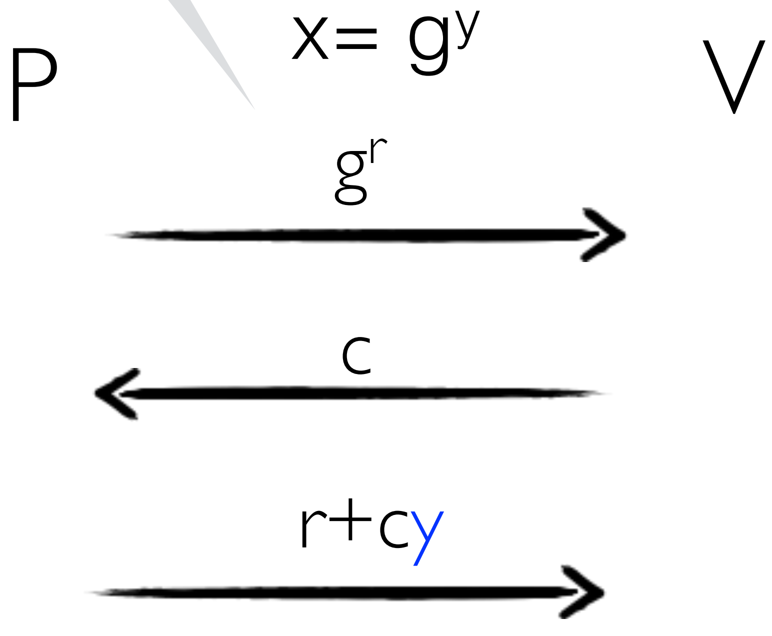
WI Proof of Knowledge of Hamilt onicity
[Blum86, **L**apidot**S**hamir**9**]



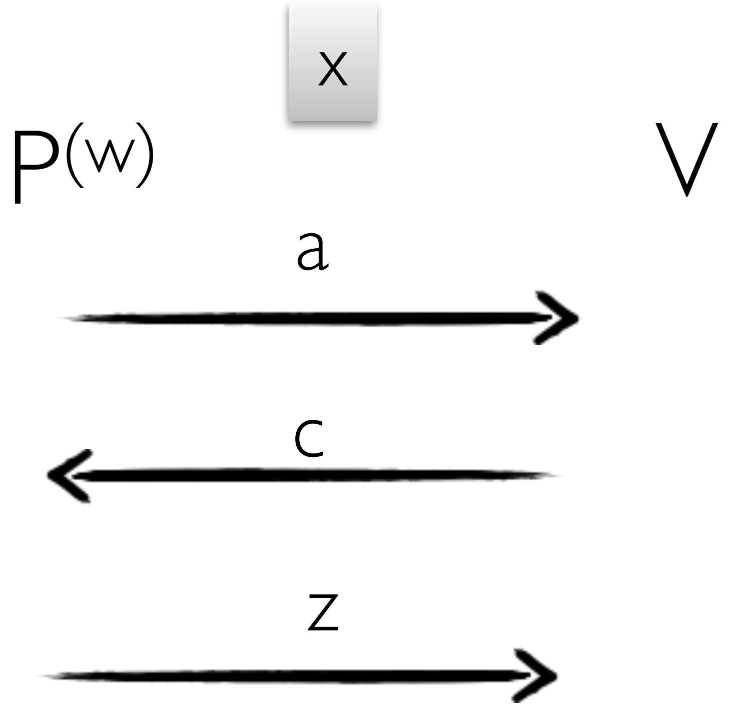
Σ-protocol for **R**

“(x, y) in $\mathbf{R}_{\text{Dlog}}$”

e.g. D scret Log [Schnorr89])



∑-protocol for relation **R**



x

$P^{(w)}$ V

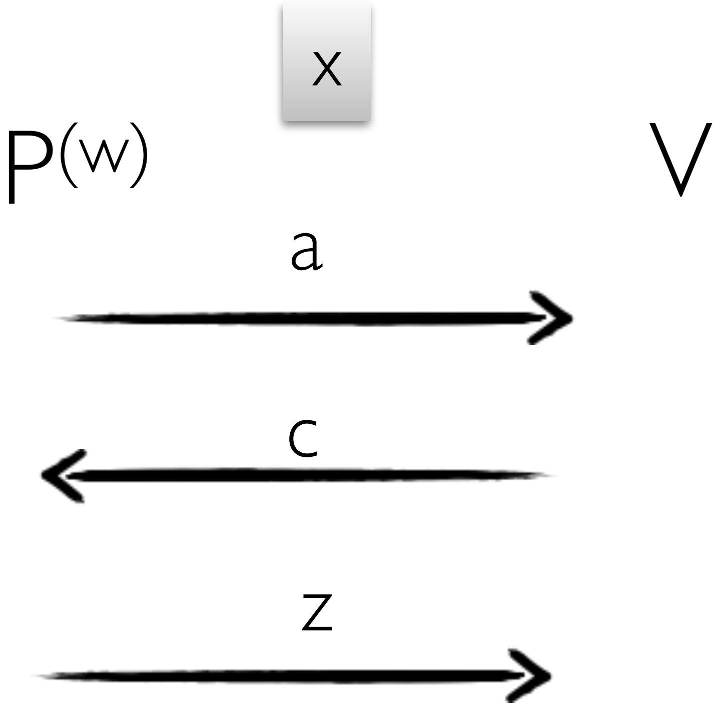
a

c

z

∑-protocol for relation R

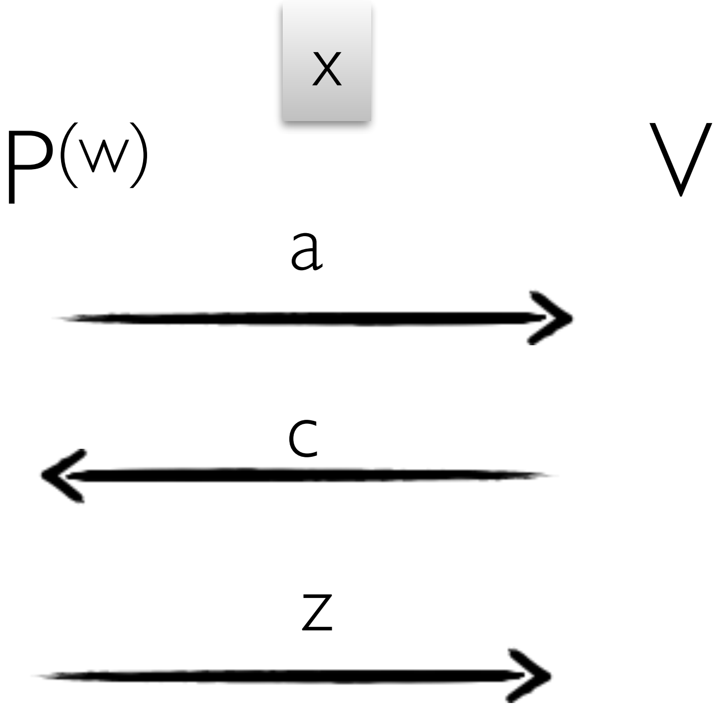
- **Completeness**



Σ-protocol for relation $\mathbf{R}$

- **Completeness**

- **SHVZK** Sim(x,c)⇒



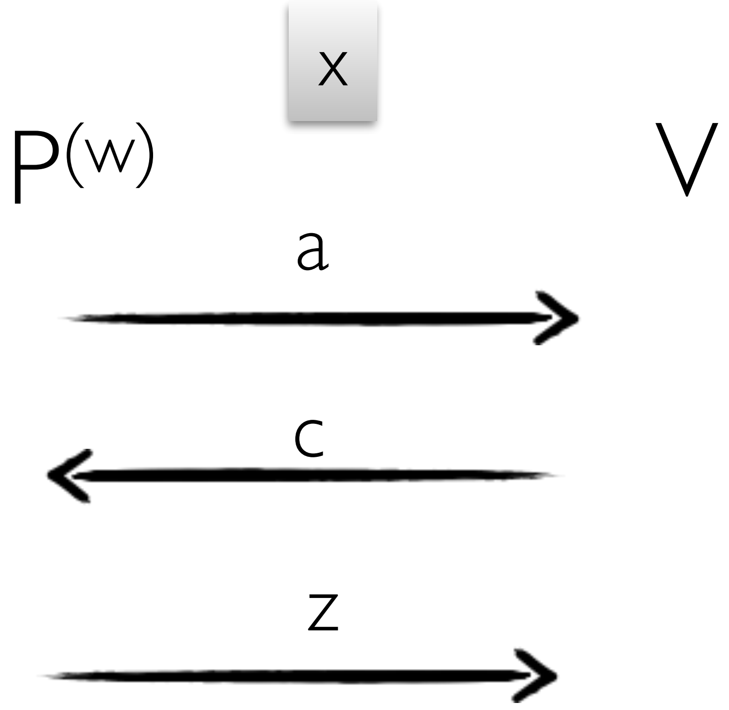
Σ-protocol for relation $\mathbf{R}$

- **Completeness**

a'

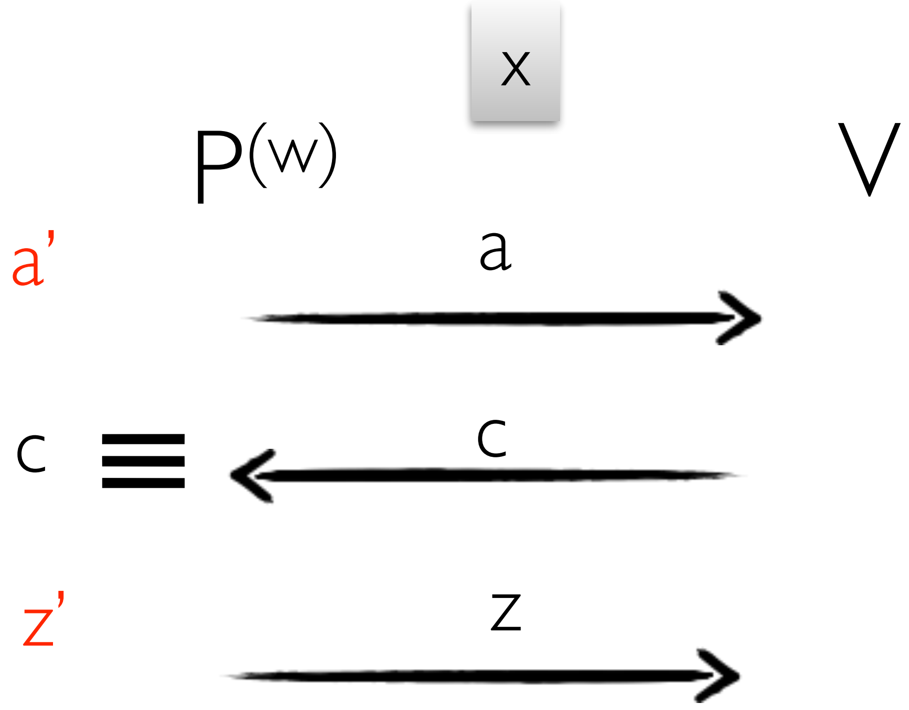
- **SHVZK** Sim(x,c)⇒ c

z'



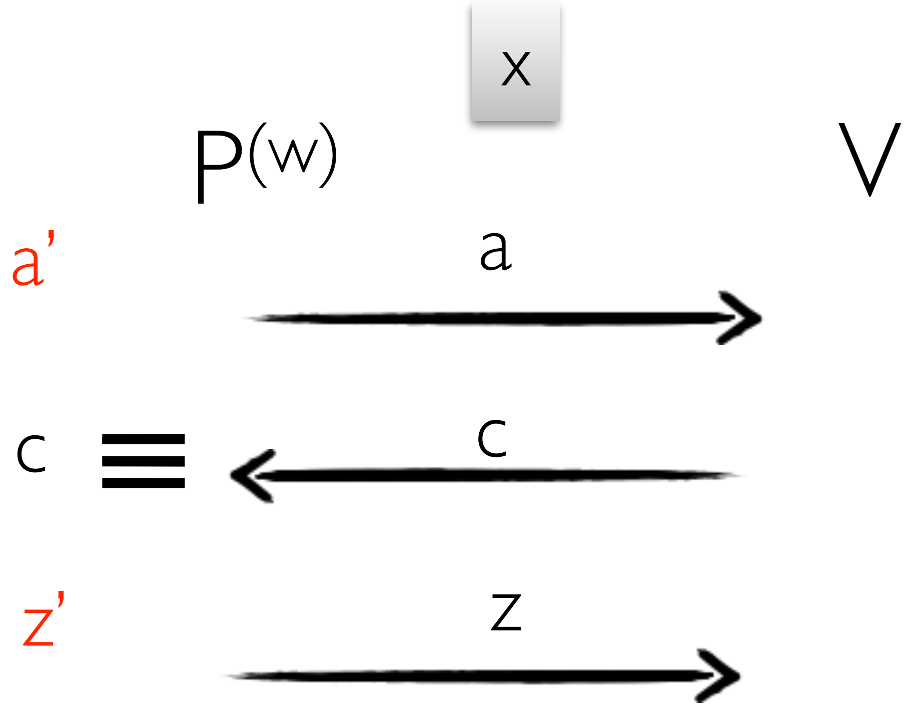
∑-protocol for relation $\mathbf{R}$

- **Completeness**
- **SHVZK** Sim(x,c)⇒



Σ-protocol for relation $\mathbf{R}$

- **Completeness**
- **SHVZK** Sim(x,c)⇒
- **Special Soundness**



x

$P^{(w)}$

V

a'

a

c ≡

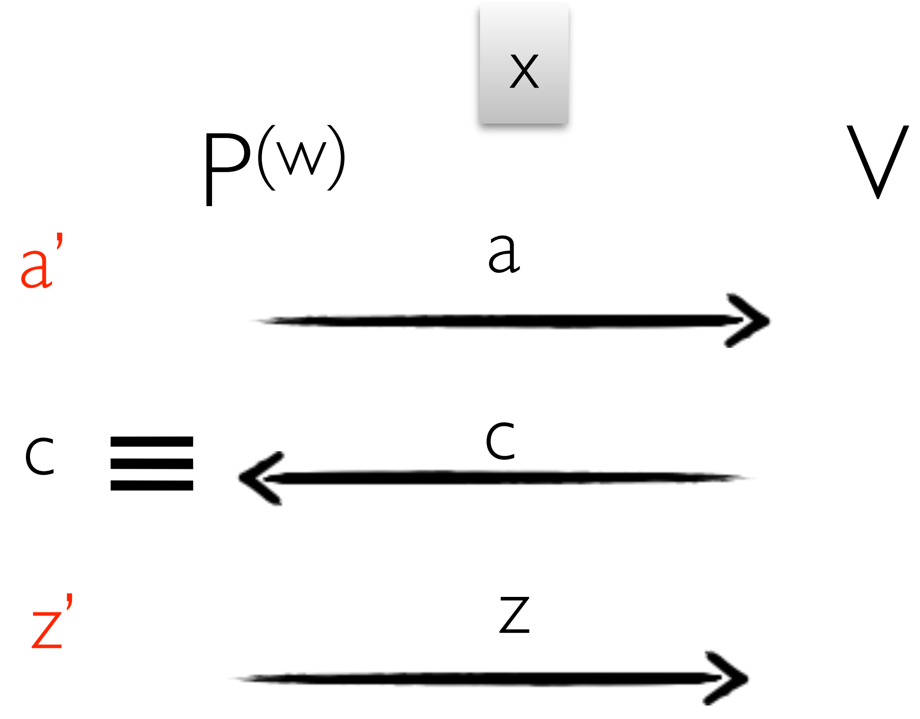
c

z'

z

∑-protocol for relation R

- **Completeness**
- **SHVZK** Sim(x,c)⇒
- **Special Soundness**



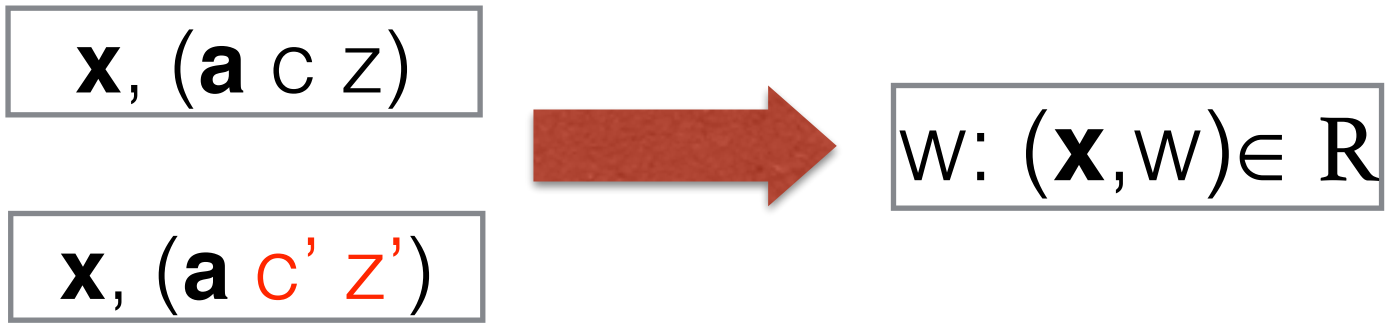
x

$P^{(w)}$ V

a' a

c ≡ c

z' z



x, (**a** c z)

x, (**a** c' z')

w: (**x**,w)∈ R

R_0 OR R_1

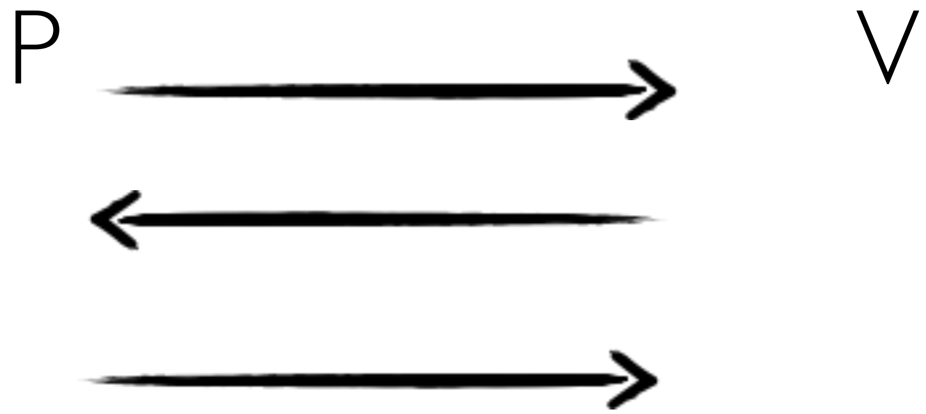
R_0 OR R_1

In theory

$$(x_0 \vee x_1) \xrightarrow{\text{NP-reduction}} G$$

"(G, C) in $\mathrm{R}_{\mathrm{HAM}}$"

WI Proof of Knowledge of Hamiltonicity
[Blum86, LS90]



In practice

Consider the ∑-protocols Σ_0 and Σ_1 for R_0 and R_1 and compile them using

[**C**ramer**D**amgard**S**choenmakers**94**]

In both cases you get 3 rounds, WI and PoK

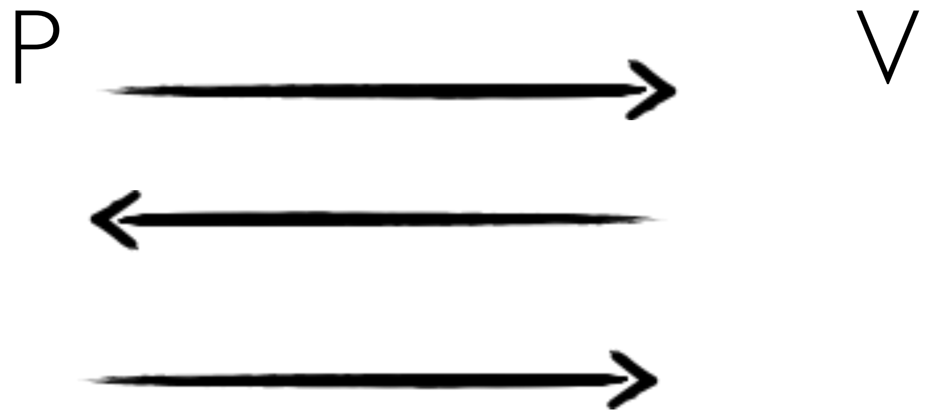
$\mathbf{R}_0$ OR $\mathbf{R}_1$: The Gap

In theory

$$(x_0 \vee x_1) \xrightarrow{\text{NP-reduction}} G$$

"(G, C) in $\mathbf{R}_{\mathrm{HAM}}$"

WI Proof of Knowledge of Hamiltonicity
[Blum86, **LS90**]



In practice

Consider the ∑-protocols Σ_0 and Σ_1 for $\mathbf{R}_0$ and $\mathbf{R}_1$ and compile them using

[**C**ramer**D**amgard**S**choenmakers**94**]

R_0 OR R_1: The Gap

In theory

$$(x_0 \vee x_1) \xrightarrow{\text{NP-reduction}} G$$

"(G, C) in R_{HAM}"

WI Proof of Knowledge of Hamiltonicity
[Blum86, **LS90**]

$$P \longrightarrow V$$

No need to know any theorem already at the 1rd round

In practice

Consider the Σ-protocols Σ_0 and Σ_1 for R_0 and R_1 and compile them using [**C**ramer**D**amgard**S**choenmakers**94**]

x_0 and x_1 are needed already at the 1rd round

R_0 OR R_1: The Gap

In theory

[LS90]

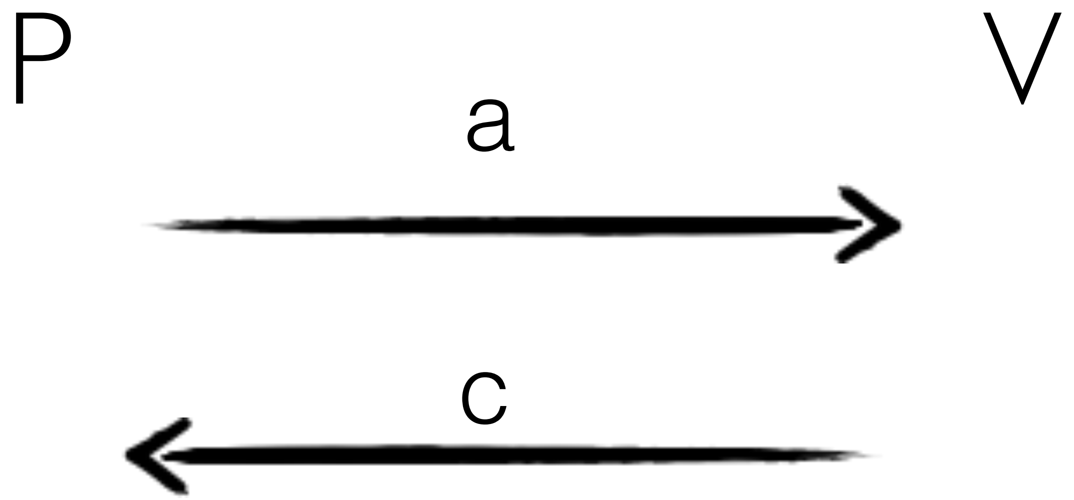
- **Delayed-Input Completeness**

In practice

[CDS94]

- **Completeness**

Delayed-Input Completeness



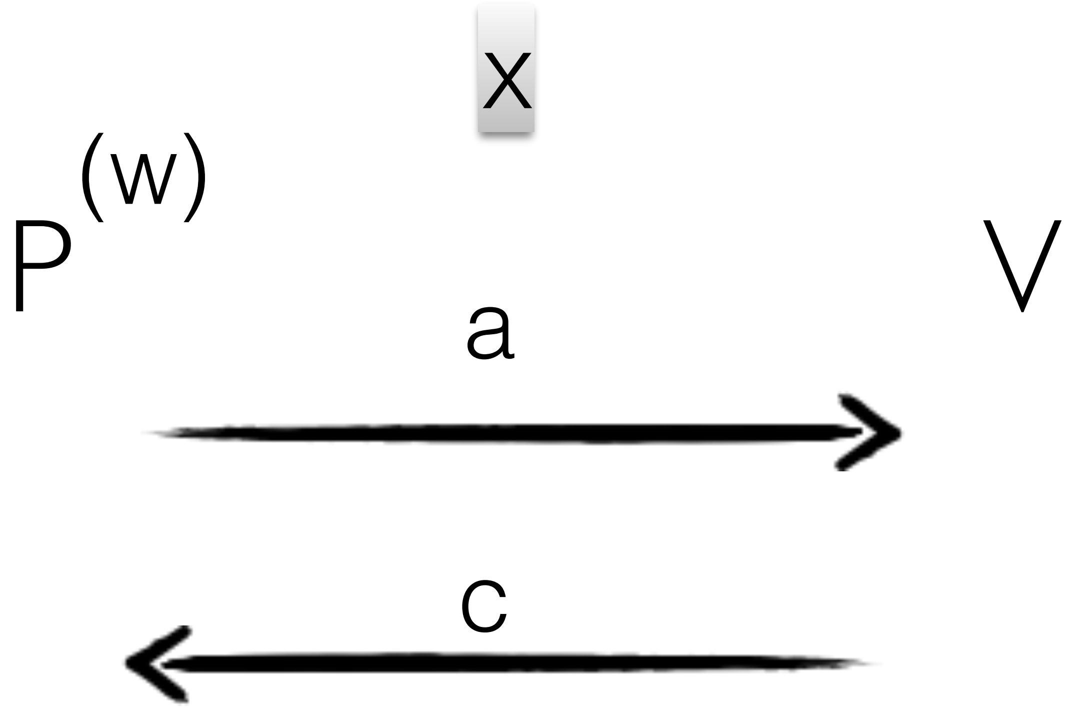
P

V

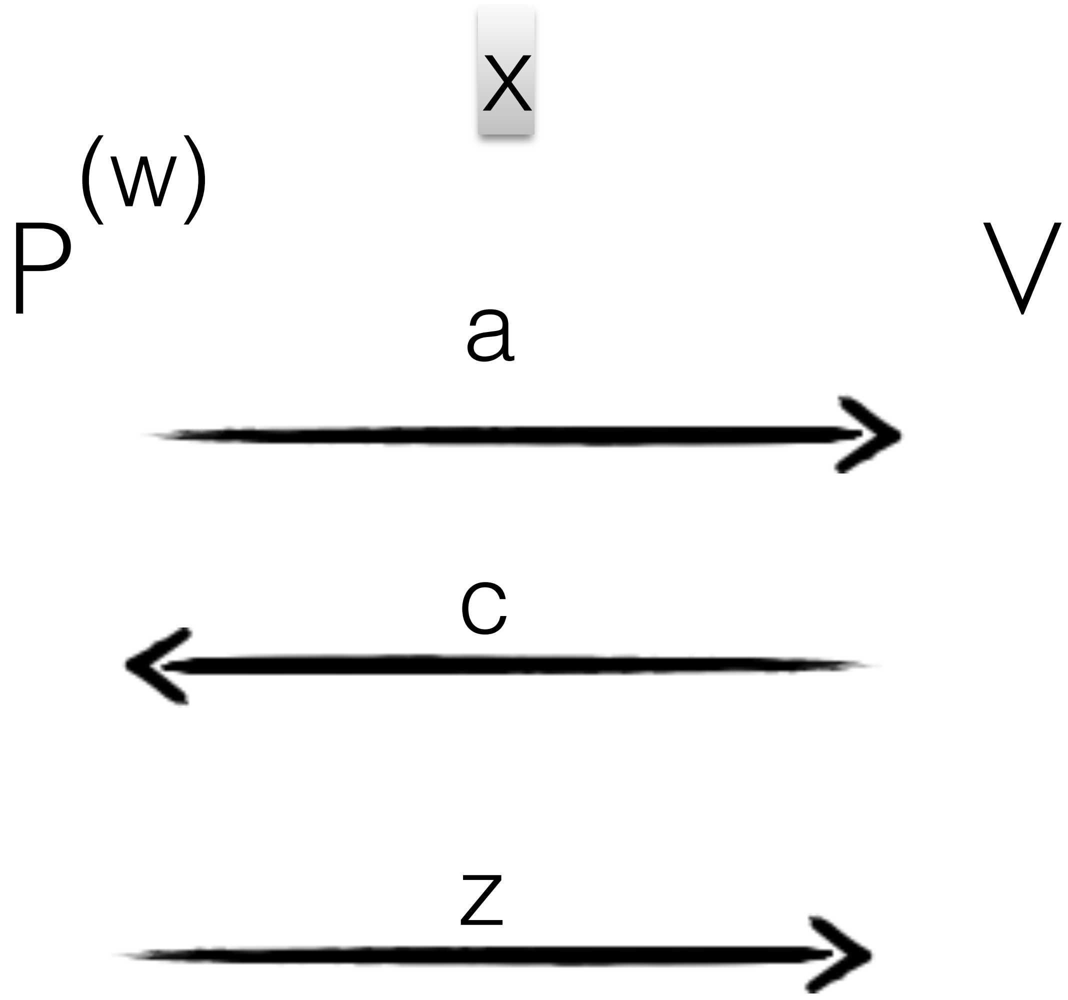
a

c

Delayed-Input Completeness



Delayed-Input Completeness



x

P^(w) V

a

c

z

R_0 OR R_1: The Gap

In theory

[LS90]

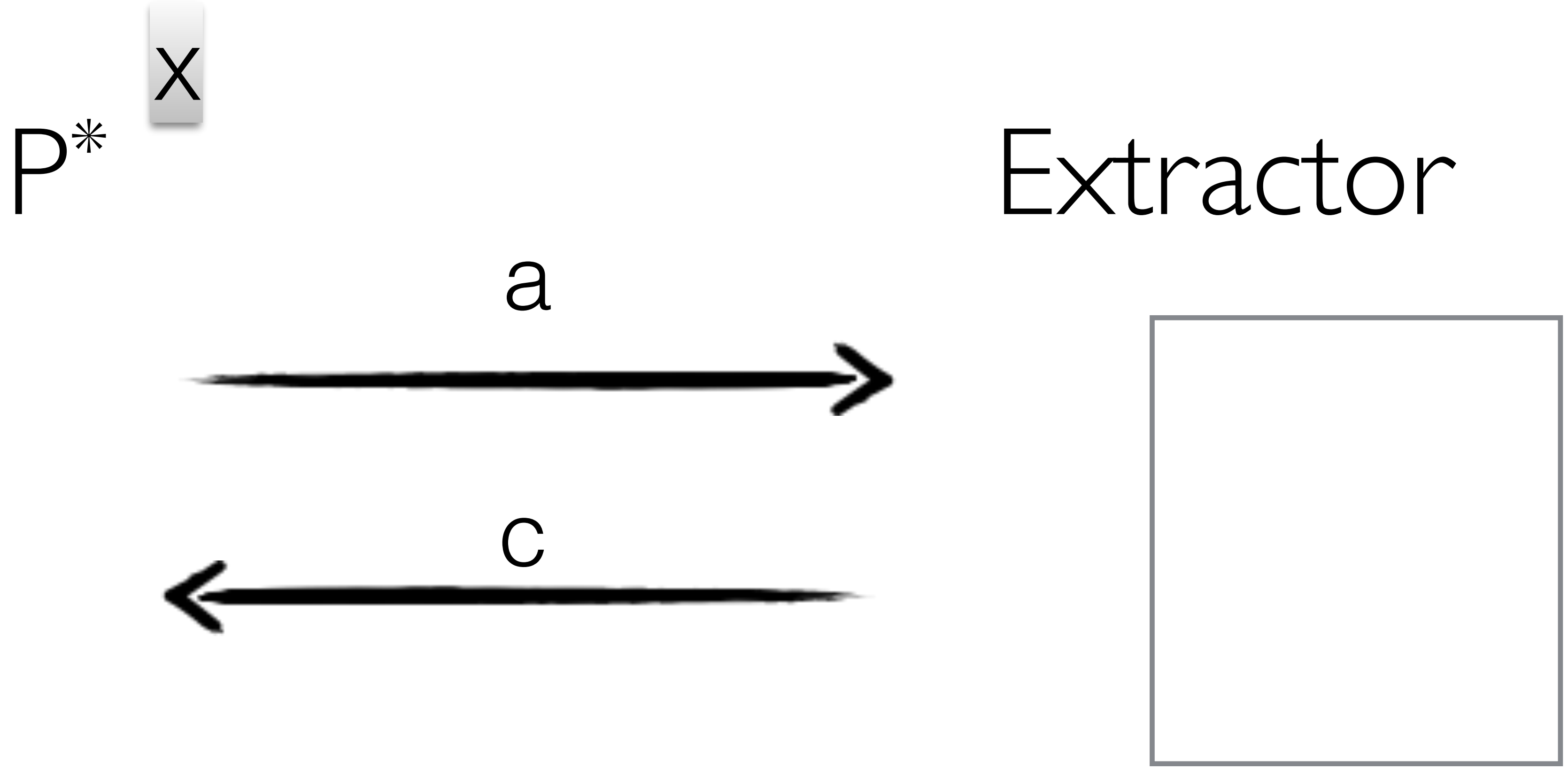
- **Delayed-Input Completeness**
- **Adaptive-Input Proof of Knowledge**

In practice

[CDS94]

- **Completeness**
- **Proof of Knowledge**

Adaptive-Input PoK



x

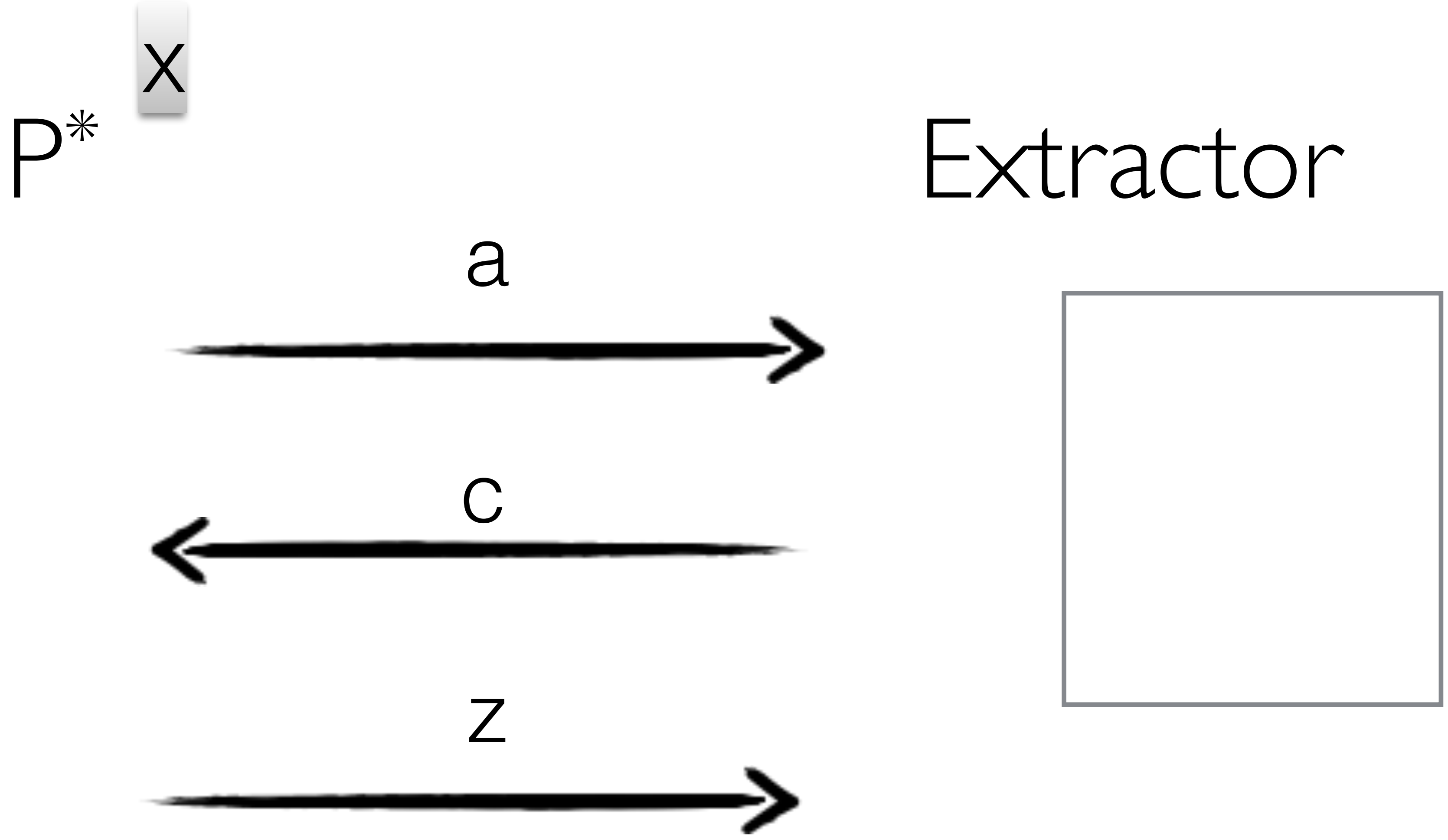
P*

Extractor

a

c

Adaptive-Input PoK



x

P*

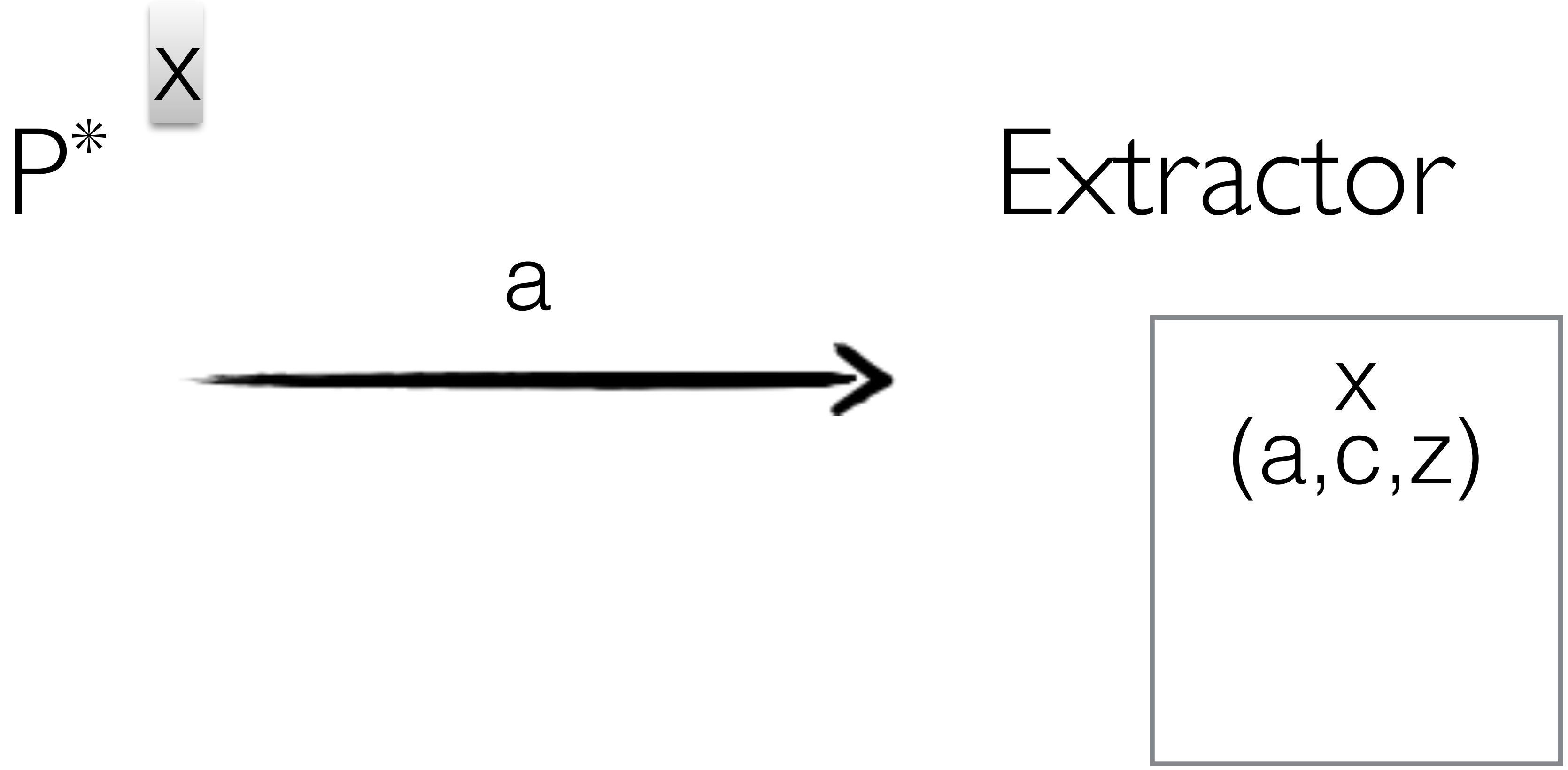
Extractor

a

c

z

Adaptive-Input PoK



P*

x

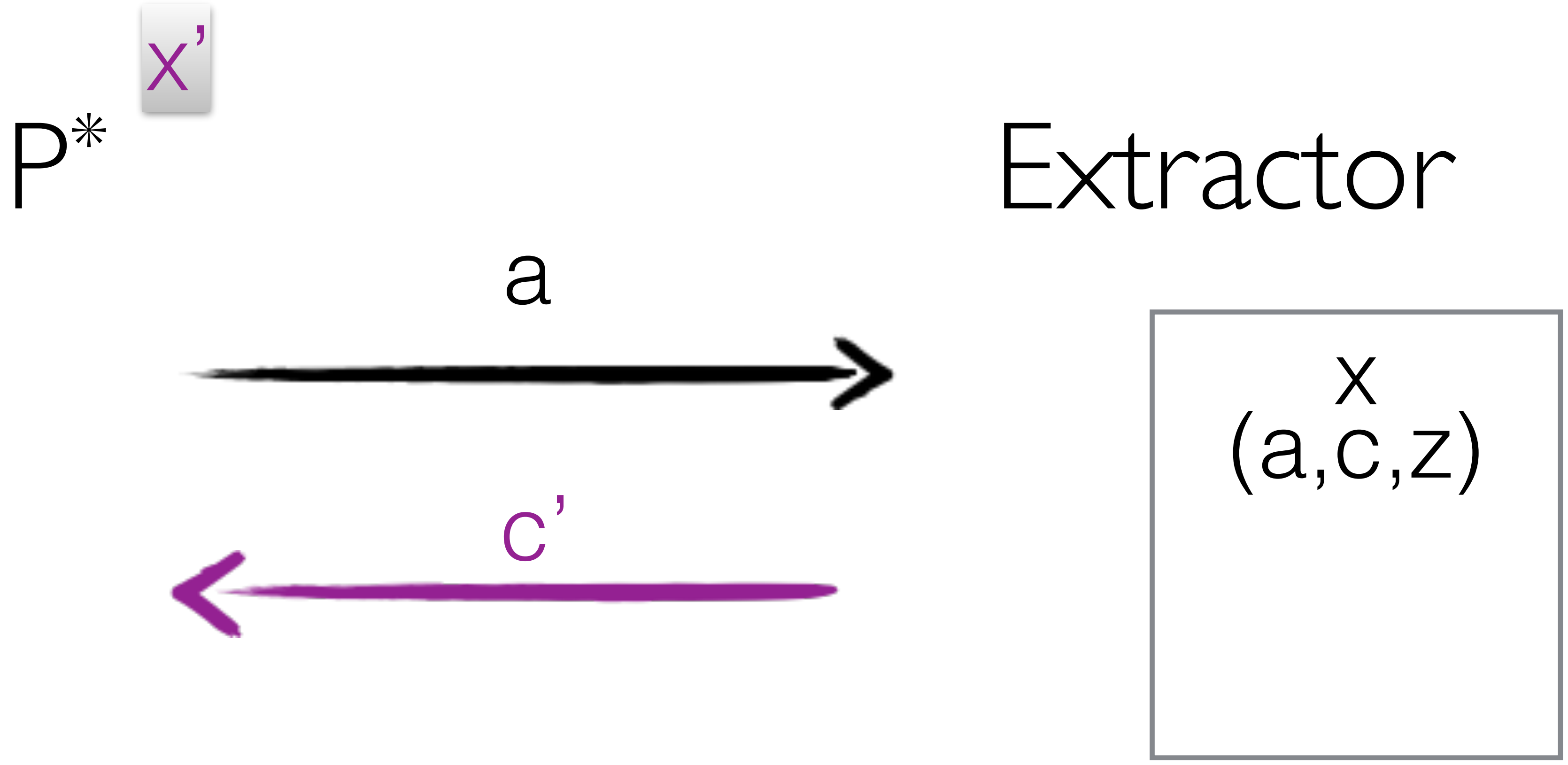
Extractor

a

x

(a,c,z)

Adaptive-Input PoK



x’

P*

Extractor

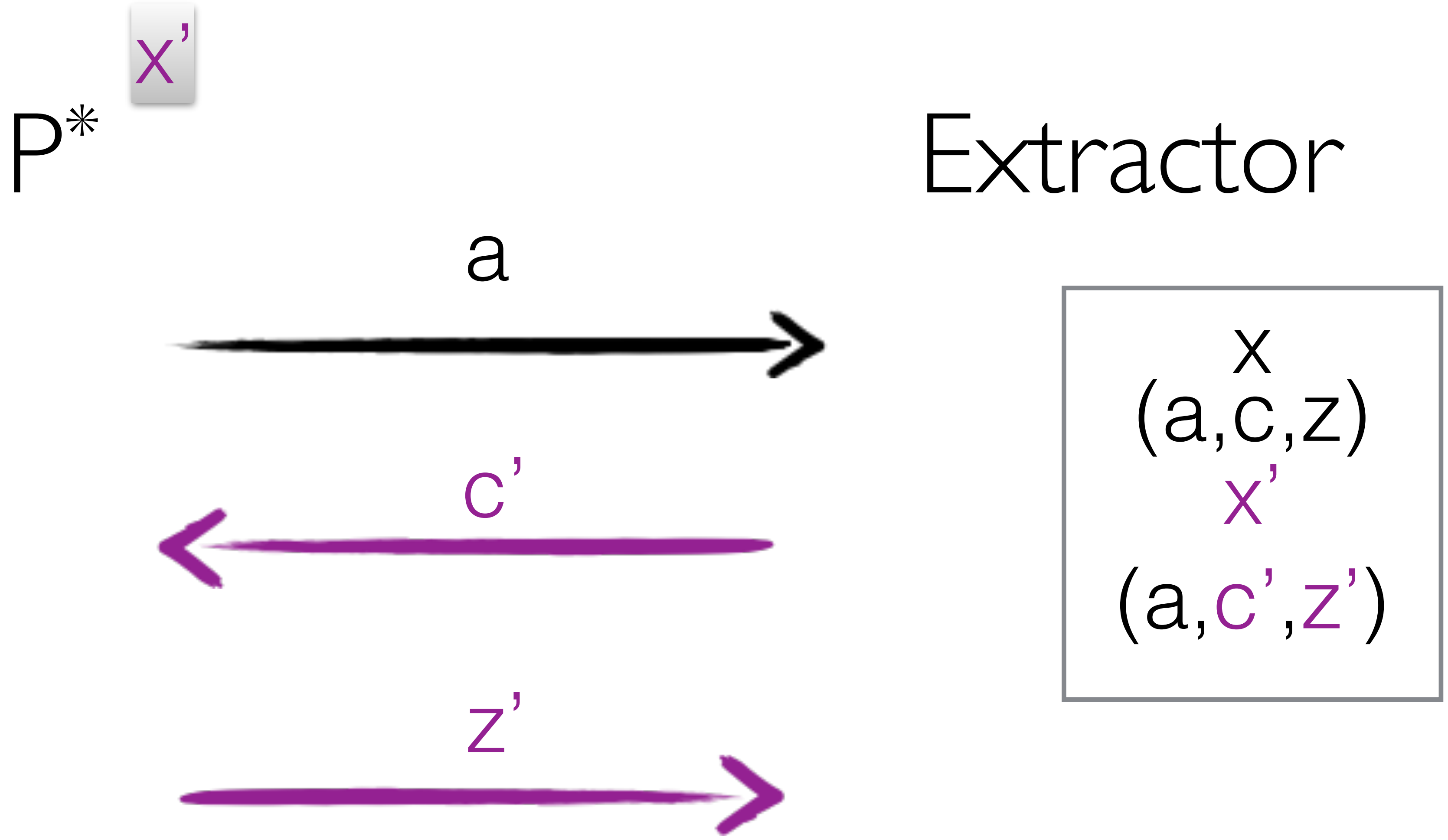
a

c’

x

(a,c,z)

Adaptive-Input PoK



x’

P*

Extractor

a

c’

z’

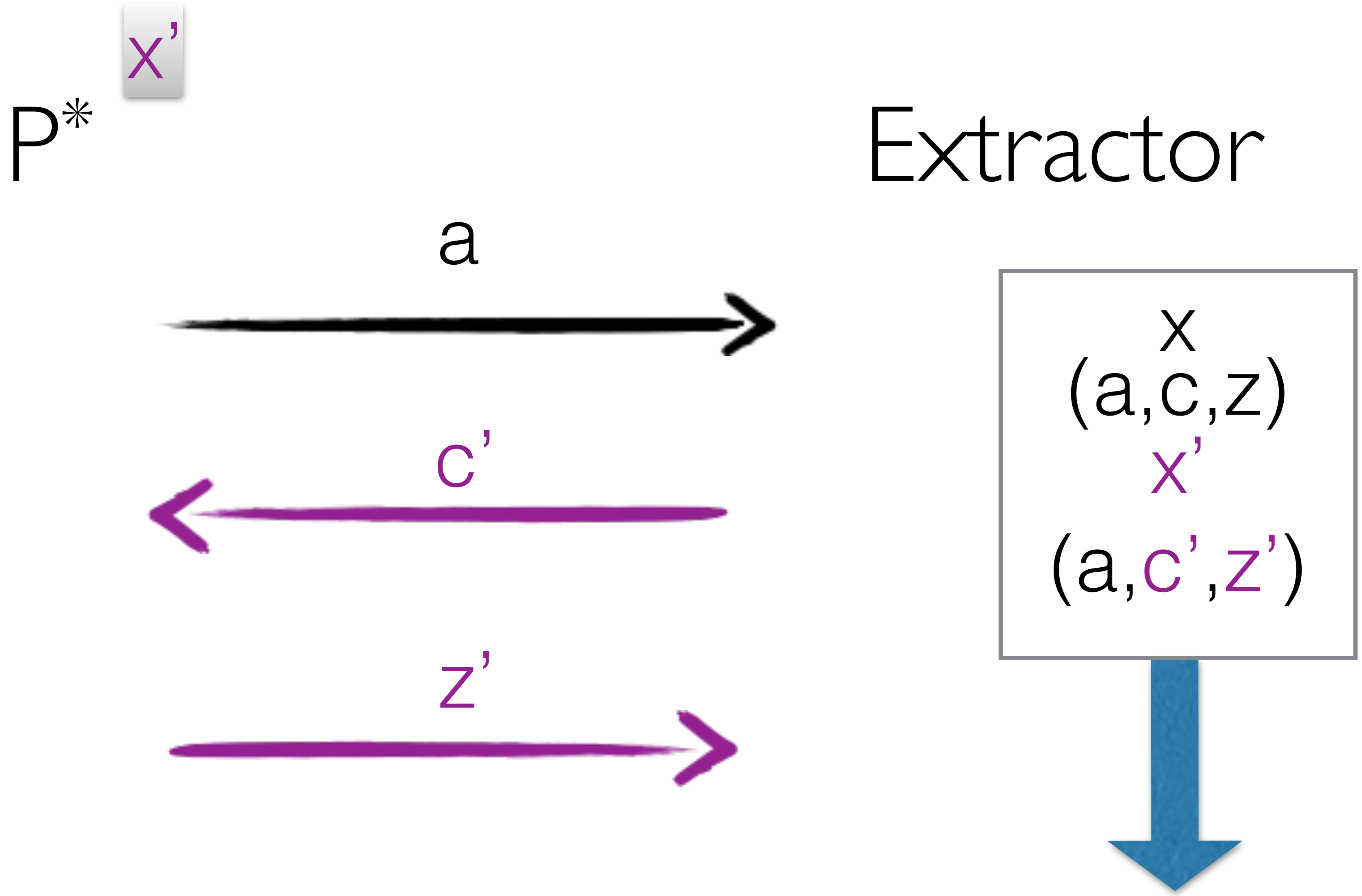
x

(a,c,z)

x’

(a,c’,z’)

Adaptive-Input PoK



x'

P*

Extractor

a

c'

z'

x

(a,c,z)

x'

(a,c',z')

w witness for x

R_0 OR R_1: The Gap

In theory

[LS90]

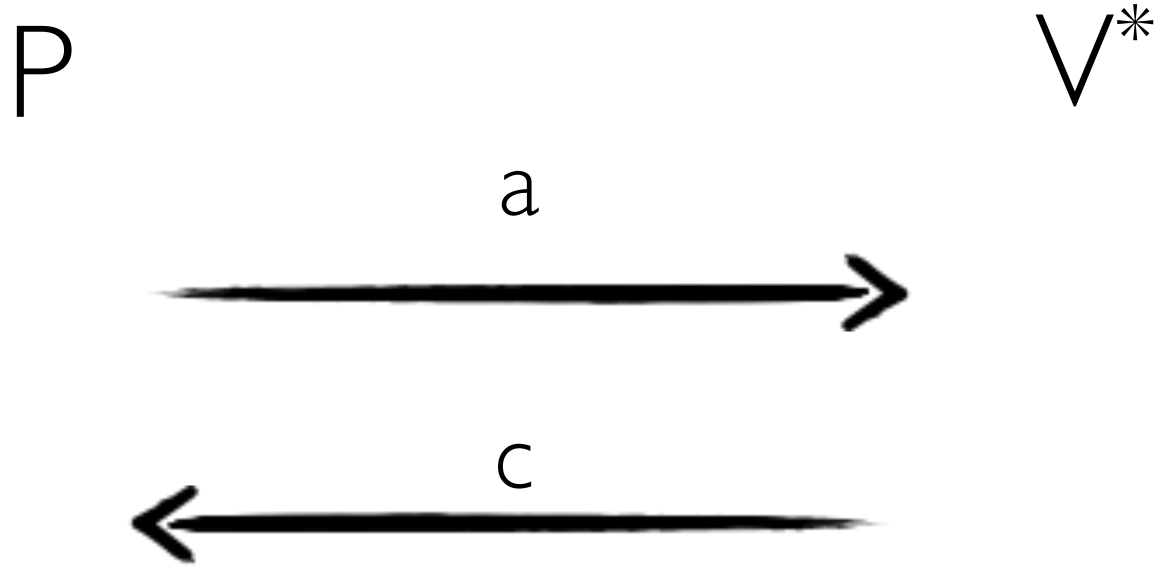
- **Delayed-Input Completeness**
- **Adaptive-Input Proof of Knowledge**
- **Adaptive-Input Witness Indistinguishable**

In practice

[CDS94]

- **Completeness**
- **Proof of Knowledge**
- **Witness Indistinguishable**

Adaptive-Input WI



P

V^*

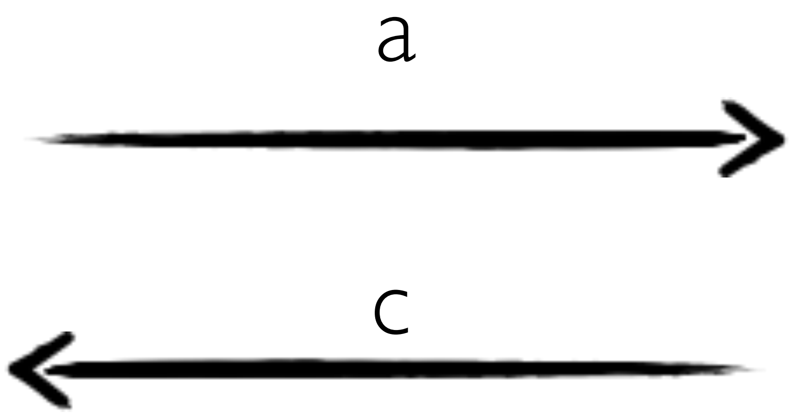
a

c

Adaptive-Input WI

(x, w_1, w_2)

$P^{(w_b)}$ V^*

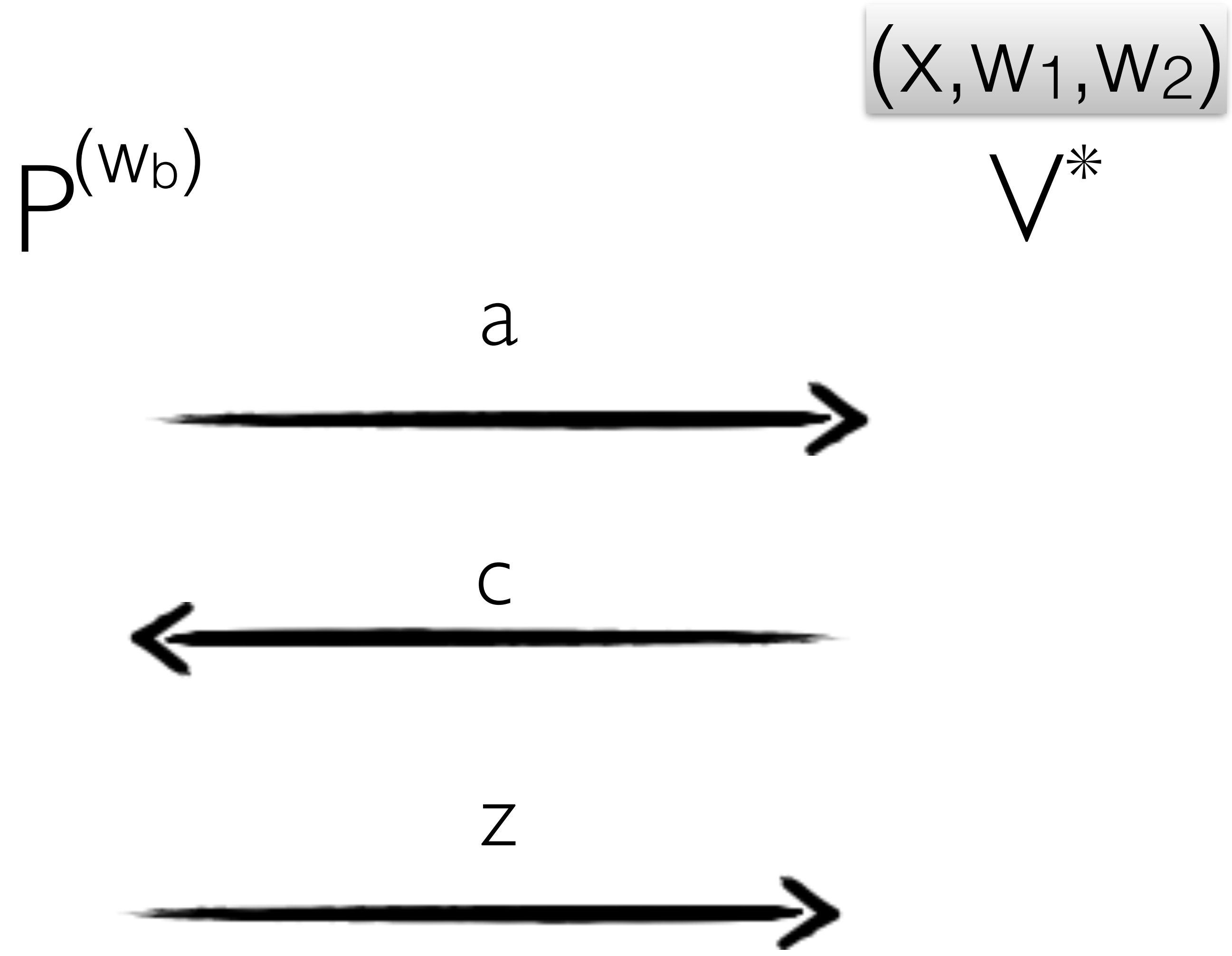


a

c

w_1, w_2 witnesses for x

Adaptive-Input WI



(x,w_1,w_2)

$P^{(w_b)}$ V^*

a

c

z

w_1,w_2 witnesses for x

R_0 OR R_1: The Gap

In theory

[LS90]

- **Delayed-Input Completeness**
- **Adaptive-Input Proof of Knowledge**
- **Adaptive-Input Witness Indistinguishable**
- **Assumption:** OWP

In practice

[CDS94]

- **Completeness**
- **Proof of Knowledge**
- **Witness Indistinguishable**
- **Assumption:** none

R_0 OR R_1: The Gap

In theory

[LS90]

- **Delayed-Input Completeness**
- **Adaptive-Input Proof of Knowledge**
- **Adaptive-Input Witness Indistinguishable**
- **Assumption:** OWP
- **Requires NP-reduction and gives Computational WI**

In practice

[CDS94]

- **Completeness**
- **Proof of Knowledge**
- **Witness Indistinguishable**
- **Assumption:** none
- **No NP-reduction and gives Perfect WI**

R_0 OR R_1: The Gap

In theory

[LS90]

- Delayed-Input Completeness
- Adaptive-Input Proof of Knowledge
- Adaptive-Input Witness Indistinguishable
- Assumption: OWP
- Requires NP-reduction and gives Computational WI
- Applicable to All NP

In practice

[CDS94]

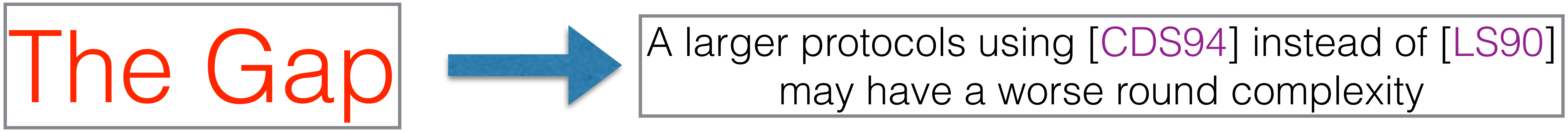
- Completeness
- Proof of Knowledge
- Witness Indistinguishable
- Assumption: none
- No NP-reduction and gives Perfect WI
- Restricted to ∑-protocols

R_0 OR R_1

The Gap →

A larger protocols using [CDS94] instead of [LS90] may have a worse round complexity

R_0 OR R_1



The Gap

A larger protocols using [CDS94] instead of [LS90] may have a worse round complexity

e.g. [Pass – Eurocrypt 03], [KaOs – Crypto 04], [YuZh – Eurocrypt 07][ScVi – Eurocrypt 12]…

R_0 OR R_1

The Gap → A larger protocols using [CDS94] instead of [LS90] may have a worse round complexity

e.g. [Pass – Eurocrypt 03], [KaOs – Crypto 04], [YuZh – Eurocrypt 07][ScVi – Eurocrypt 12]…

Recently Delayed-Input completeness is widely used

[GMPP16 – tomorrow], [Kiayias0Z15 – CCS15], [BBKPV16 – eprint]…

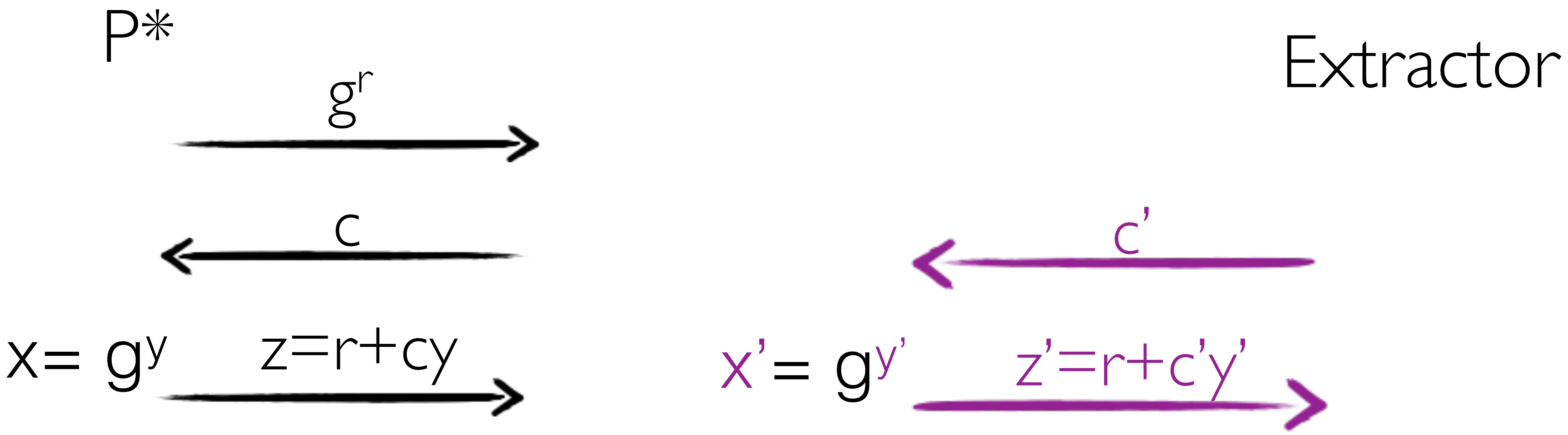
Our Results

1) From PoK to Adaptive-Input PoK

2) Bridging the gap

Our First Result: from PoK to Adaptive-Input PoK

Σ-Protocols (in general) are not Adaptive-Input PoK



P*

Extractor

g^r

c

c'

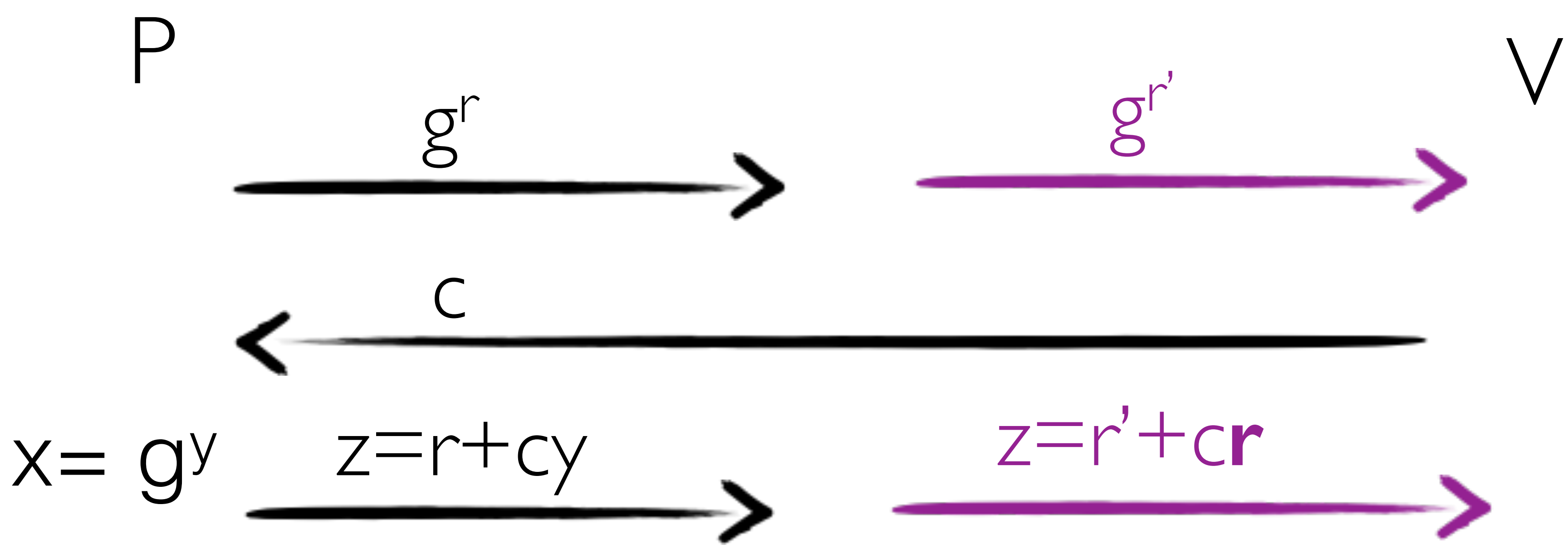
$x= g^y$ $z=r+cy$

$x'= g^{y'}$ $z'=r+c'y'$

Issue observed in [**B**ernhard**P**ereira**W**arinschi**12**] about the weak Fiat-Shamir transform

Our Transform

From PoK to Adaptive-Input PoK



P

V

g^r

$g^{r'}$

c

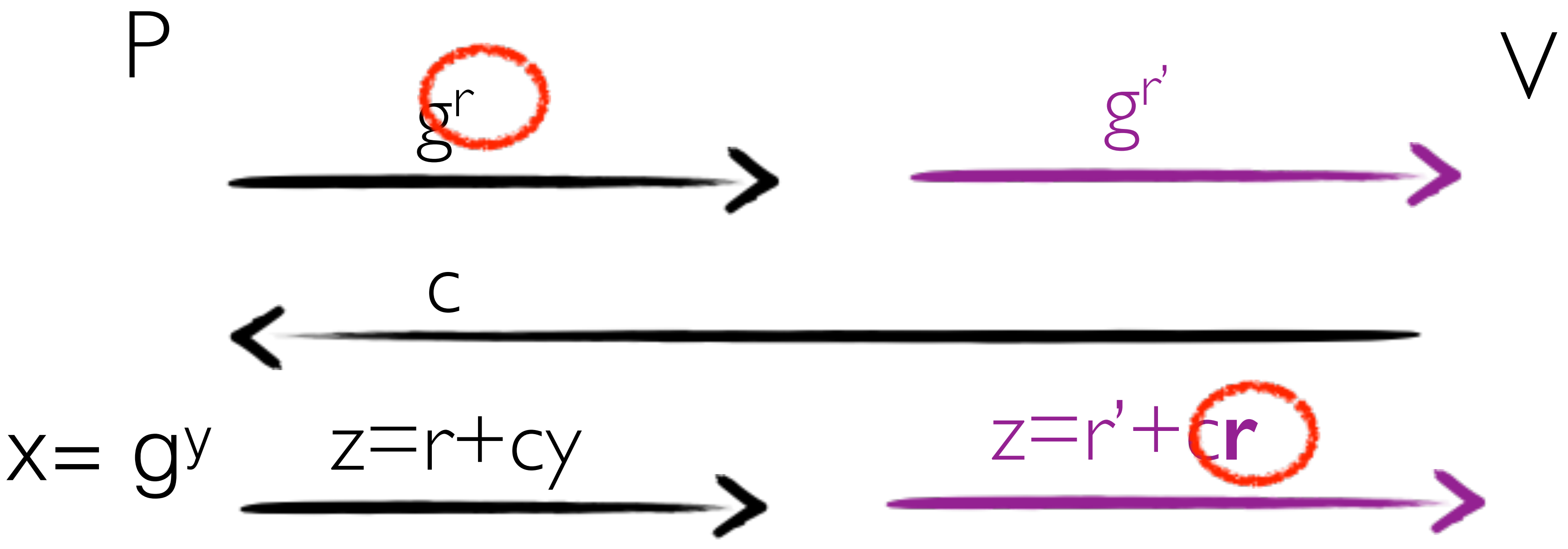
$x = g^y$

$z=r+cy$

$z=r'+c\mathbf{r}$

Our Transform

From PoK to Adaptive-Input PoK



P

V

g^r

$g^{r'}$

c

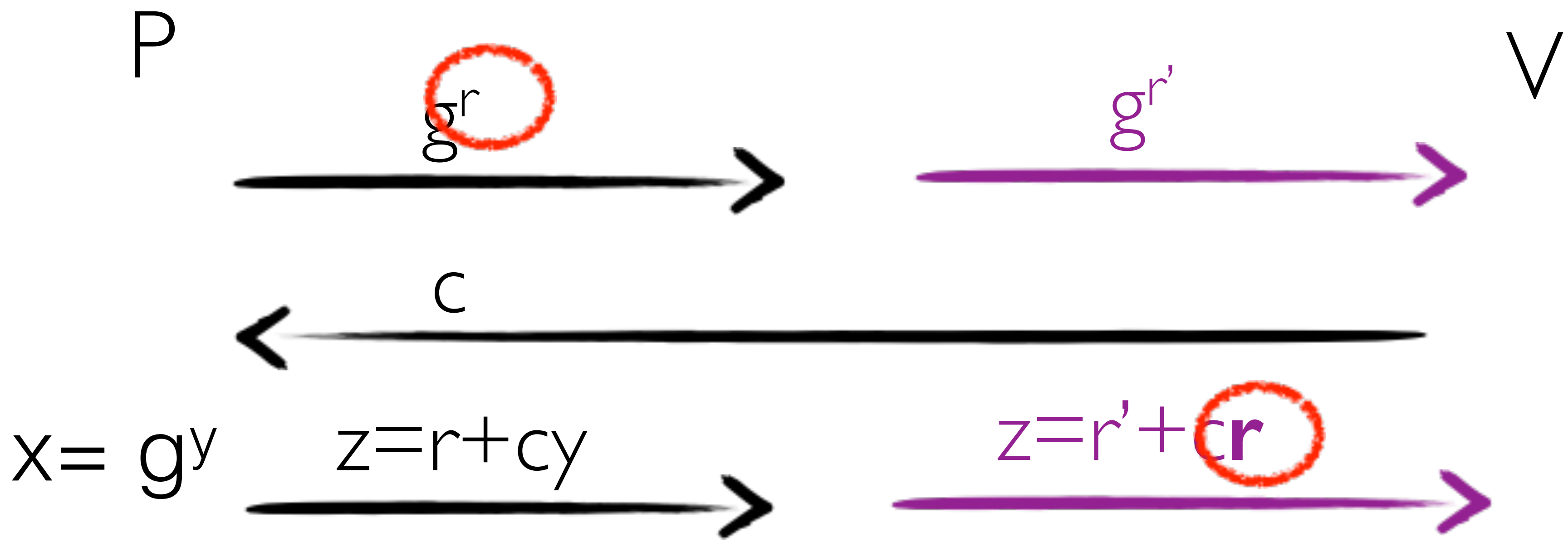
$x= g^y$

$z=r+cy$

$z=r'+c\mathbf{r}$

Our Transform

From PoK to Adaptive-Input PoK



P

V

g^r

$g^{r'}$

c

$x = g^y$

$z=r+cy$

$z=r'+c\mathbf{r}$

Our transform applies to the class described in [**C**ramer**96**, **M**aurer**15**, **C**ramer**D**amgard**98**]

e.g. Schnorr, Guillou–Quisquater, Diffie–Hellman, Multiplication proof for pedersen commitments, …

Our Results

1) From PoK to Adaptive-Input PoK

2) Bridging the gap

$\mathbf{R}_0$ OR $\mathbf{R}_1$: Bridging the Gap

In theory

[LS90]

In practice

[CDS94]

[CPS+ TCC 2016-A]

This work

$\mathbf{R}_0$ OR $\mathbf{R}_1$: Bridging the Gap

In theory

[LS90]

- Delayed-Input Completeness

In practice

[CDS94]

- Completeness

[CPS+ TCC 2016-A]

- Semi-Delayed Input Completeness

This work

- Delayed-Input Completeness: All input ∑-protocols have to be Delayed-Input

$\mathbf{R}_0$ OR $\mathbf{R}_1$: Bridging the Gap

In theory

[LS90]

- Delayed-Input Completeness
- Adaptive-Input PoK

In practice

[CDS94]

- Completeness
- Proof of Knowledge

[CPS+ TCC 2016-A]

- Semi-Delayed Input Completeness
- Proof of Knowledge

This work

- Delayed-Input Completeness: All input ∑-protocols have to be Delayed-Input
- Proof of Knowledge

R_0 OR R_1: Bridging the Gap

In theory

[LS90]

- Delayed-Input Completeness
- Adaptive-Input PoK
- Adaptive-Input WI

In practice

[CDS94]

- Completeness
- Proof of Knowledge
- Witness Indistinguishable

[CPS$^+$ TCC 2016-A]

- Semi-Delayed Input Completeness
- Proof of Knowledge
- Semi-Adaptive Input WI: one of two instances is adaptively chosen by V*

This work

- Delayed-Input Completeness: All input ∑-protocols have to be Delayed-Input
- Proof of Knowledge
- Adaptive-Input WI

R_0 OR R_1: Bridging the Gap

In theory

[LS90]

- Delayed-Input Completeness
- Adaptive-Input PoK
- Adaptive-Input WI
- Assumption: OWP

In practice

[CDS94]

- Completeness
- Proof of Knowledge
- Witness Indistinguishable
- Assumption: none

[CPS+ TCC 2016-A]

- Semi-Delayed Input Completeness
- Proof of Knowledge
- Semi-Adaptive Input WI: one of two instances is adaptively chosen by V*
- Assumption: none

This work

- Delayed-Input Completeness: All input ∑-protocols have to be Delayed-Input
- Proof of Knowledge
- Adaptive-Input WI
- Assumption: DDH

R_0 OR R_1: Bridging the Gap

In theory

[LS90]

- Delayed-Input Completeness
- Adaptive-Input PoK
- Adaptive-Input WI
- Assumption: OWP
- Works with multiple OR compositions

In practice

[CDS94]

- Completeness
- Proof of Knowledge
- Witness Indistinguishable
- Assumption: none
- Works with multiple OR compositions

[CPS$^+$ TCC 2016-A]

- Semi-Delayed Input Completeness
- Proof of Knowledge
- Semi-Adaptive Input WI: one of two instances is adaptively chosen by V*
- Assumption: none
- Works with only one OR composition

This work

- Delayed-Input Completeness: All input ∑-protocols have to be Delayed-Input
- Proof of Knowledge
- Adaptive-Input WI
- Assumption: DDH
- Works with multiple OR compositions

R_0 OR R_1: Bridging the Gap

In theory

[LS90]

- Delayed-Input Completeness
- Adaptive-Input PoK
- Adaptive-Input WI
- Assumption: OWP
- Works with multiple OR compositions
- Requires NP-reduction and gives Computational WI

In practice

[CDS94]

- Completeness
- Proof of Knowledge
- Witness Indistinguishable
- Assumption: none
- Works with multiple OR compositions
- No NP-reduction and gives Perfect WI

[CPS+ TCC 2016-A]

- Semi-Delayed Input Completeness
- Proof of Knowledge
- Semi-Adaptive Input WI: one of two instances is adaptively chosen by V*
- Assumption: none
- Works with only one OR composition
- No NP-reduction and gives Perfect WI

This work

- Delayed-Input Completeness: All input ∑-protocols have to be Delayed-Input
- Proof of Knowledge
- Adaptive-Input WI
- Assumption: DDH
- Works with multiple OR compositions
- No NP-reduction and gives Computational WI

R_0 OR R_1: Bridging the Gap

In theory

[LS90]

- Delayed-Input Completeness
- Adaptive-Input PoK
- Adaptive-Input WI
- Assumption: OWP
- Works with multiple OR compositions
- Requires NP-reduction and gives Computational WI
- Applicable to All NP

In practice

[CDS94]

- Completeness
- Proof of Knowledge
- Witness Indistinguishable
- Assumption: none
- Works with multiple OR compositions
- No NP-reduction and gives Perfect WI
- Restricted to Σ-protocols

[CPS^{+} TCC 2016-A]

- Semi-Delayed Input Completeness
- Proof of Knowledge
- Semi-Adaptive Input WI: one of two instances is adaptively chosen by V*
- Assumption: none
- Works with only one OR composition
- No NP-reduction and gives Perfect WI
- Restricted to (a large class of) Σ-protocols

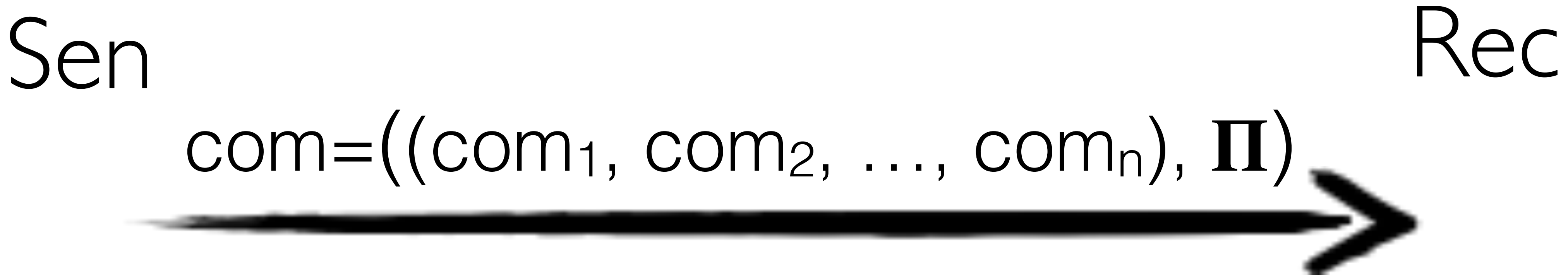
This work

- Delayed-Input Completeness: All input Σ-protocols have to be Delayed-Input
- Proof of Knowledge
- Adaptive-Input WI
- Assumption: DDH
- Works with multiple OR compositions
- No NP-reduction and gives Computational WI
- Restricted to (a large class of) Σ-protocols

Comparison: Summary

	Assumption	Completeness	Adaptive WI	Adaptive PoK	Online Efficiency
[LS90]	OWP	Delayed-Input	k out of n (all adaptive)	k out of n	NP-reduction
[CDS94]	/	/	/	k out of n*	Entire protocol
[CPSSV16]	/	Semi-Delayed Input	1 out of 2 (1 adaptive)	k out of n*	Entire protocol
This work	**DDH**	Delayed-Input	k out of n (all adaptive)	k out of n*	≤ CDS94

Our Construction: Tools



Sen

Rec

com=((com$_1$, com$_2$, …, com$_n$), $\mathbf{\Pi}$)

- (K,N) Trapdoor Commitment

Our Construction: Tools

At least k are perfectly binding

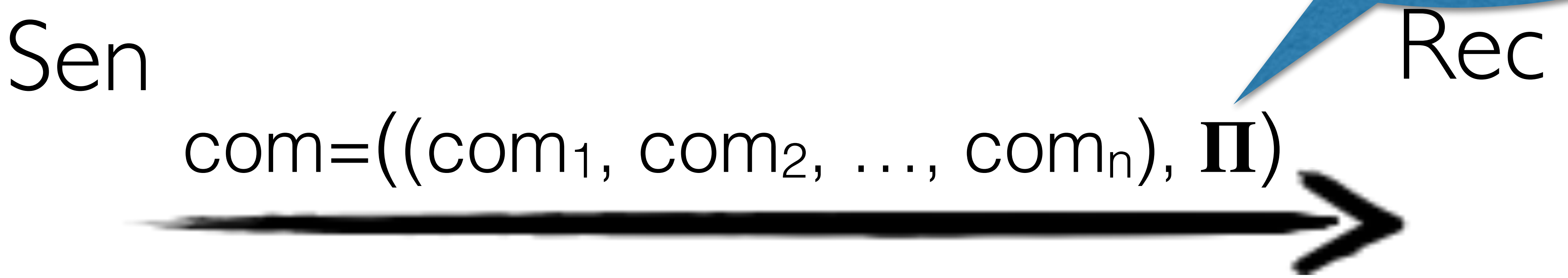
Sen

Rec

$\text{com}=((\text{com}_1, \text{com}_2, \ldots, \text{com}_n), \mathbf{\Pi})$

- (K,N) Trapdoor Commitment

Our Construction: Tools

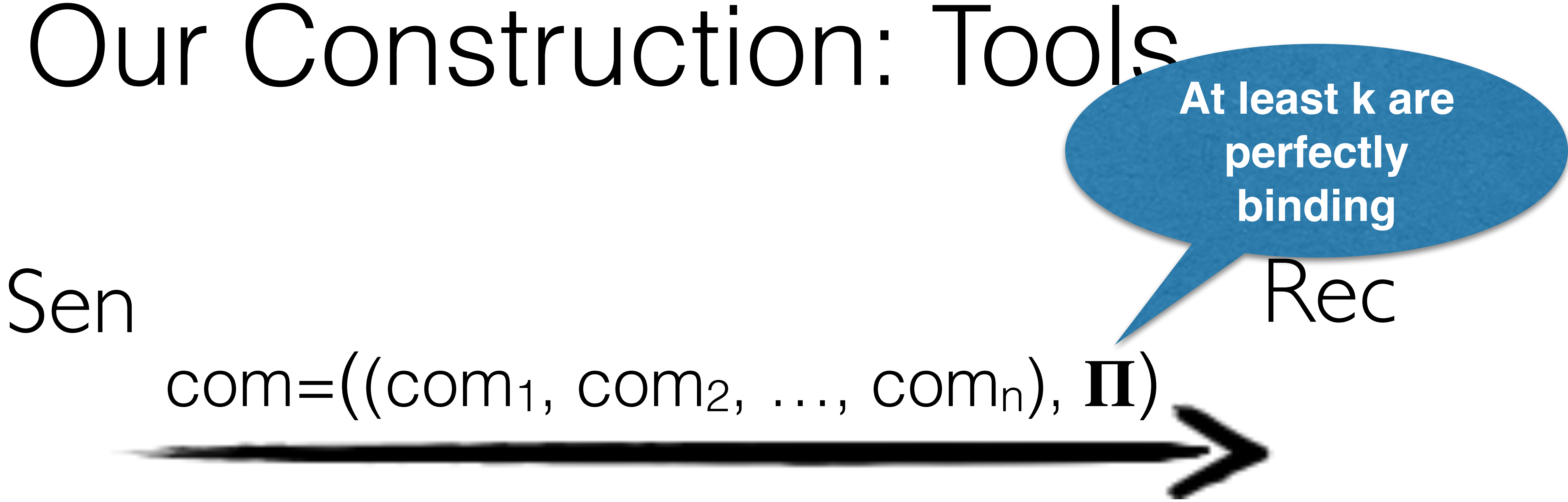


At least k are perfectly binding

- (K,N) Trapdoor Commitment
  - $\mathrm{Com}_{KN}(m_1, m_2, \ldots, m_n)$ → com
  - $\mathrm{Open}_{KN}(\mathrm{com}, m_1^*, m_2^*, \ldots, m_{n-k}^*)$ → dec

n-k commitments can be equivocated

Our Construction: Tools



At least k are perfectly binding

Sen

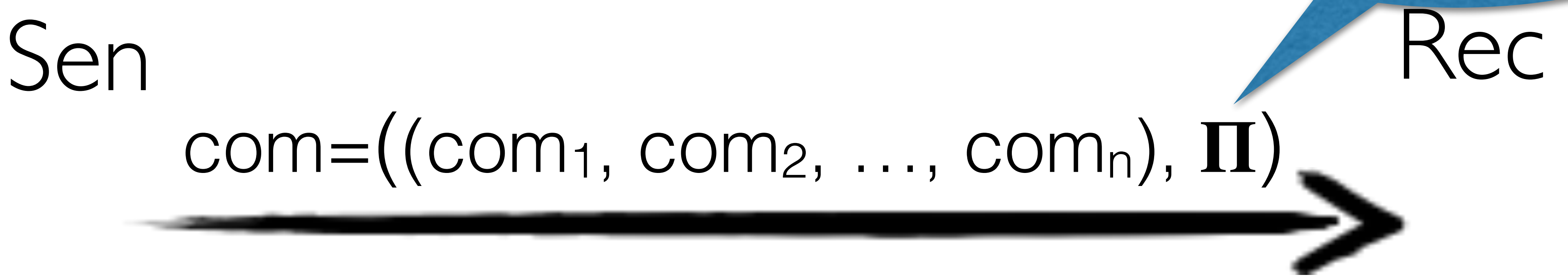
Rec

com=((com$_1$, com$_2$, …, com$_n$), $\mathbf{\Pi}$)

- (K,N) Trapdoor Commitment
  - Com$_{KN}$(m_1, m_2, …, m_n) → com
  - Open$_{KN}$(com, m_1^*, m_2^*, …, $m_{n\text{-}k}^*$) → dec
- $\sum$: Delayed-Input Σ-protocol for the relation $\mathbf{R}$

n-k commitments can be equivocated

Our Construction: Tools



Sen

Rec

com=((com$_1$, com$_2$, …, com$_n$), $\mathbf{\Pi}$)

At least k are perfectly binding

- (K,N) Trapdoor Commitment
  - Com$_{KN}$(m$_1$, m$_2$, …, m$_n$) → com
  - Open$_{KN}$(com, m$_1^*$, m$_2^*$, …, m$_{n-k}^*$) → dec

n-k commitments can be equivocated

- ∑: Delayed-Input ∑-protocol for the relation R
- Sim$_∑$: SHVZK simulator for ∑

Our Construction: Main Idea

e.g. k=1, n=2

$\mathbf{R}$ OR $\mathbf{R}$

P

V

$P_{\sum} \rightarrow a_1, a_2$

Our Construction: Main Idea

e.g. k=1, n=2

$\mathbf{R}$ OR $\mathbf{R}$

P V

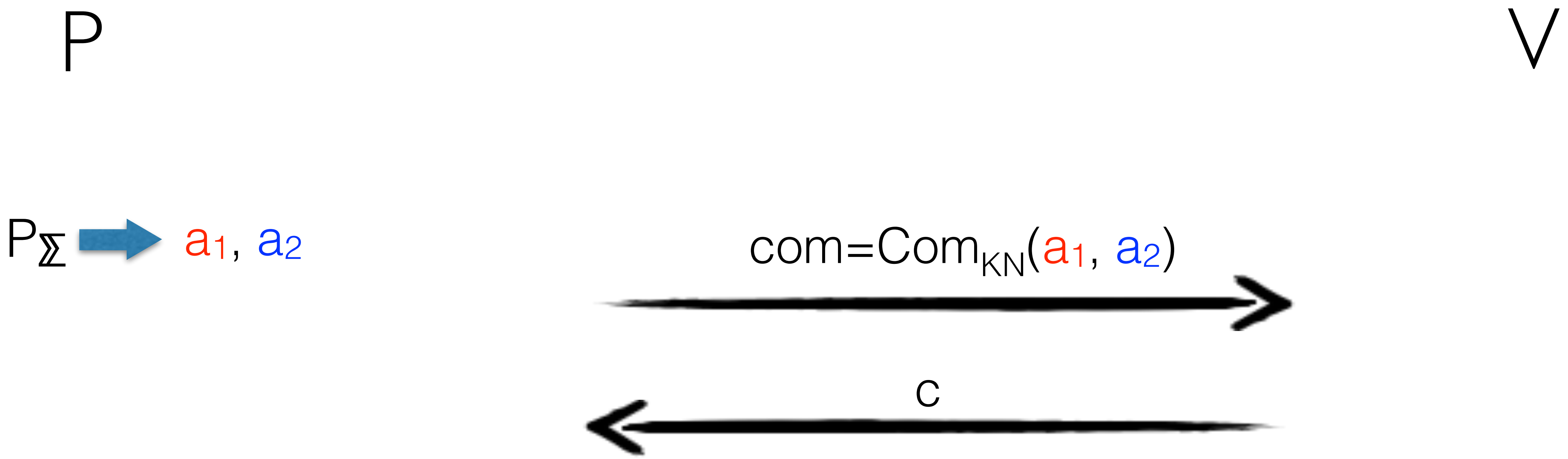
$P_{\Sigma} \rightarrow a_1, a_2$

$$\xrightarrow{\text{com}=\text{Com}_{KN}(a_1,\, a_2)}$$

Our Construction: Main Idea

e.g. k=1, n=2

$\mathbf{R}$ OR $\mathbf{R}$



P

V

P_Σ → a_1, a_2

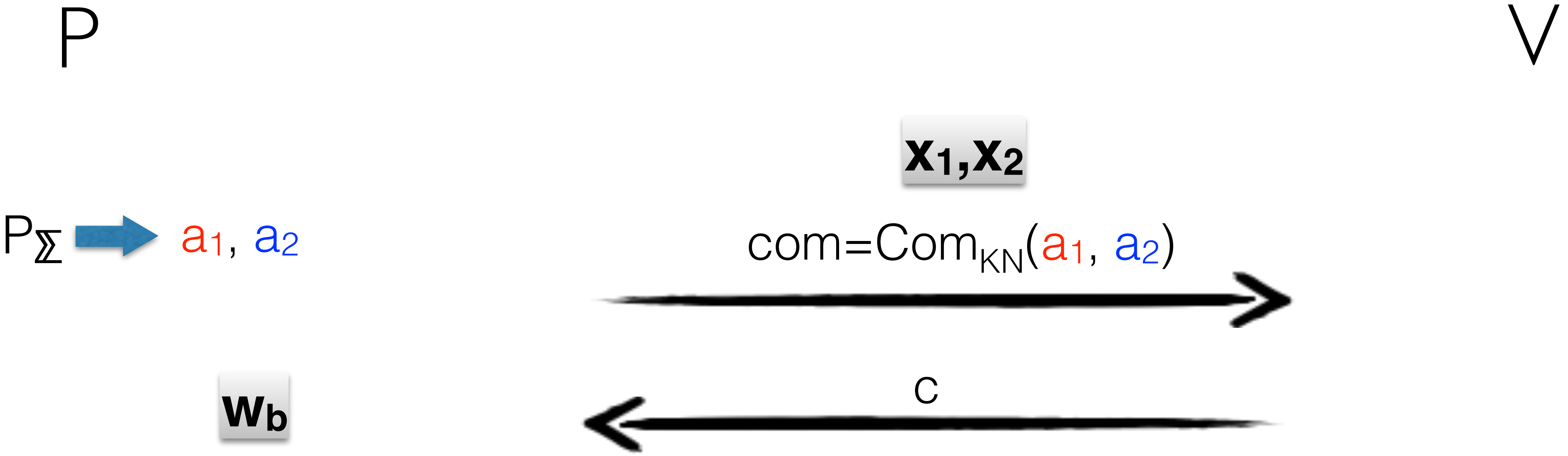
com=$\mathrm{Com}_{KN}(a_1, a_2)$

c

Our Construction: Main Idea

e.g. k=1, n=2

R OR R



P

V

x_1, x_2

P_Σ → a_1, a_2

com=$\mathrm{Com}_{KN}(a_1, a_2)$

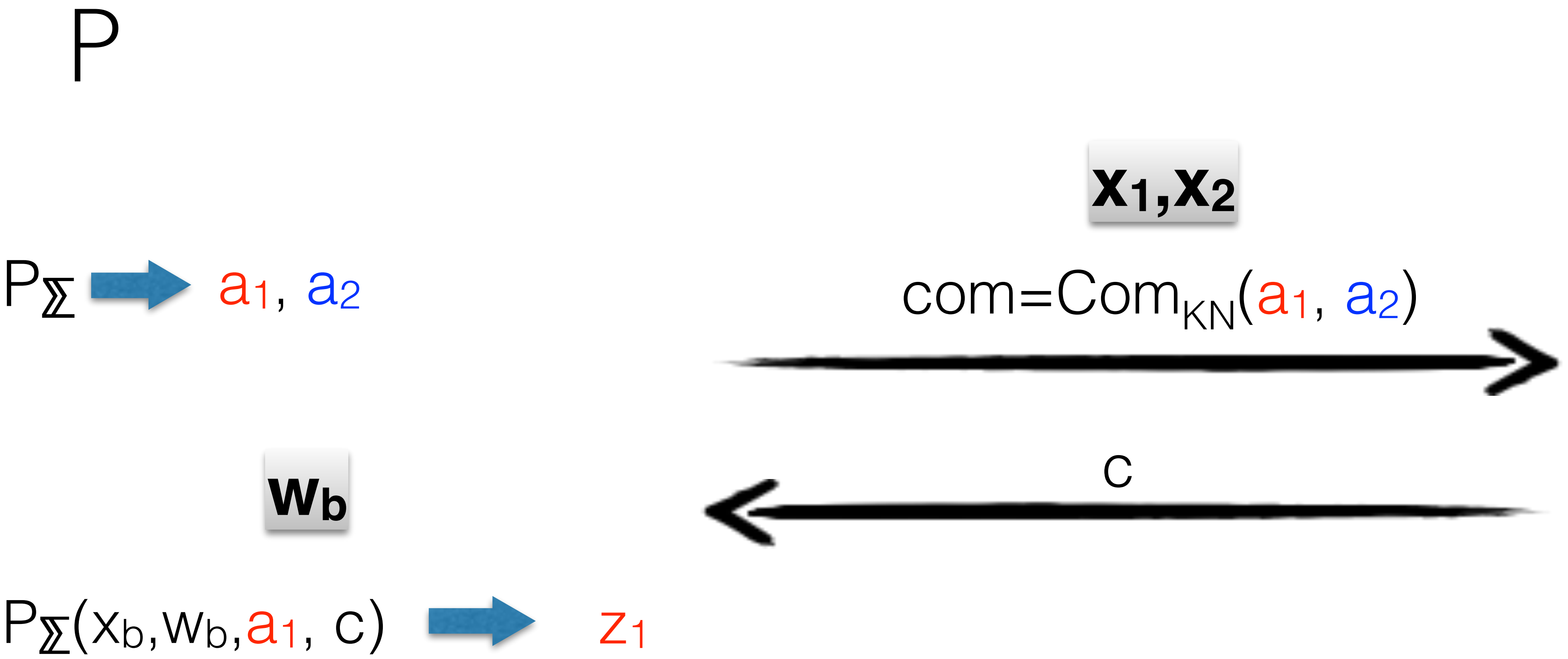
c

w_b

Our Construction: Main Idea

e.g. k=1, n=2

$\mathbf{R}$ OR $\mathbf{R}$



P

V

$\mathbf{x_1,x_2}$

$P_\Sigma \rightarrow a_1, a_2$

$com = Com_{KN}(a_1, a_2)$

c

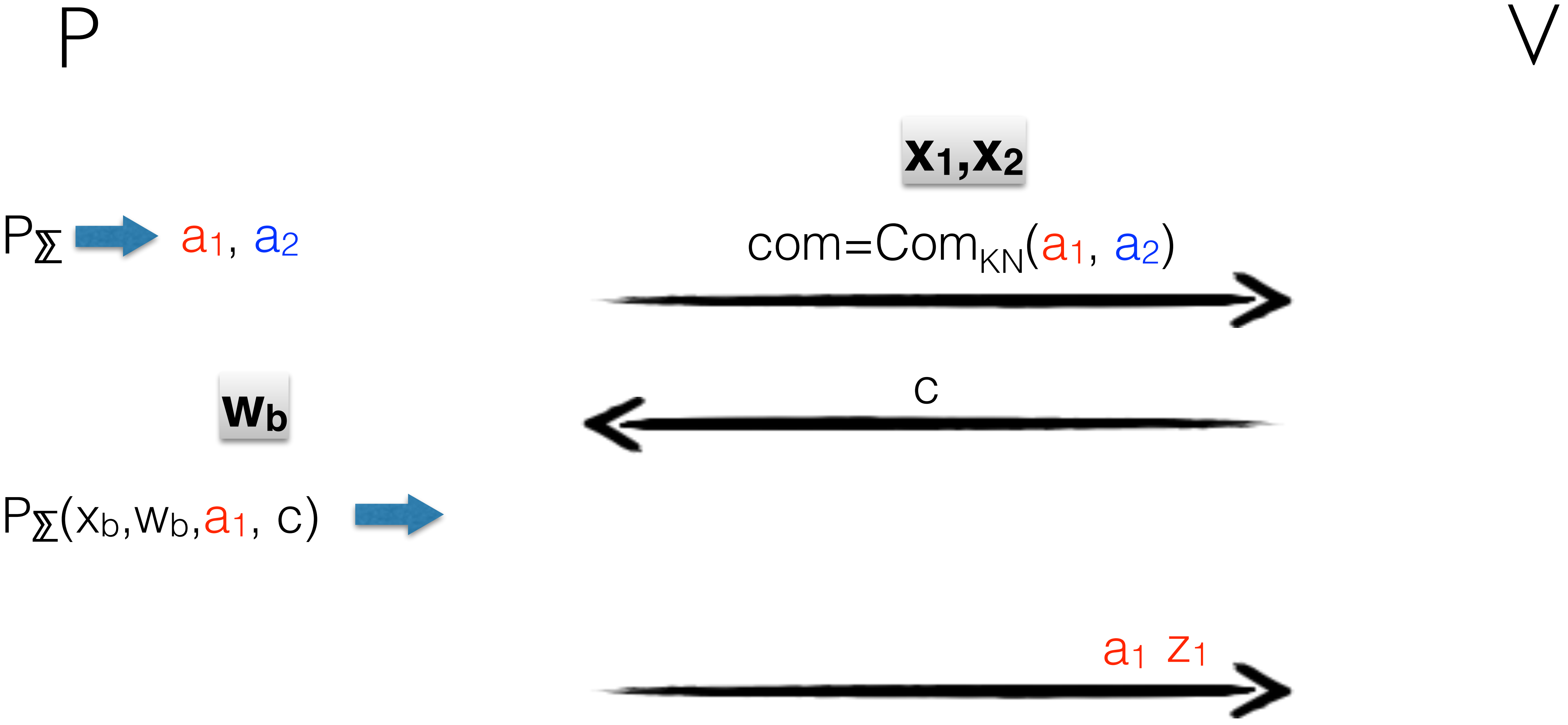
$\mathbf{w_b}$

$P_\Sigma(x_b, w_b, a_1, c) \rightarrow z_1$

Our Construction: Main Idea

e.g. k=1, n=2

R OR R



P

V

x_1,x_2

P_Σ → a_1, a_2

com=$\text{Com}_{KN}(a_1, a_2)$

c

w_b

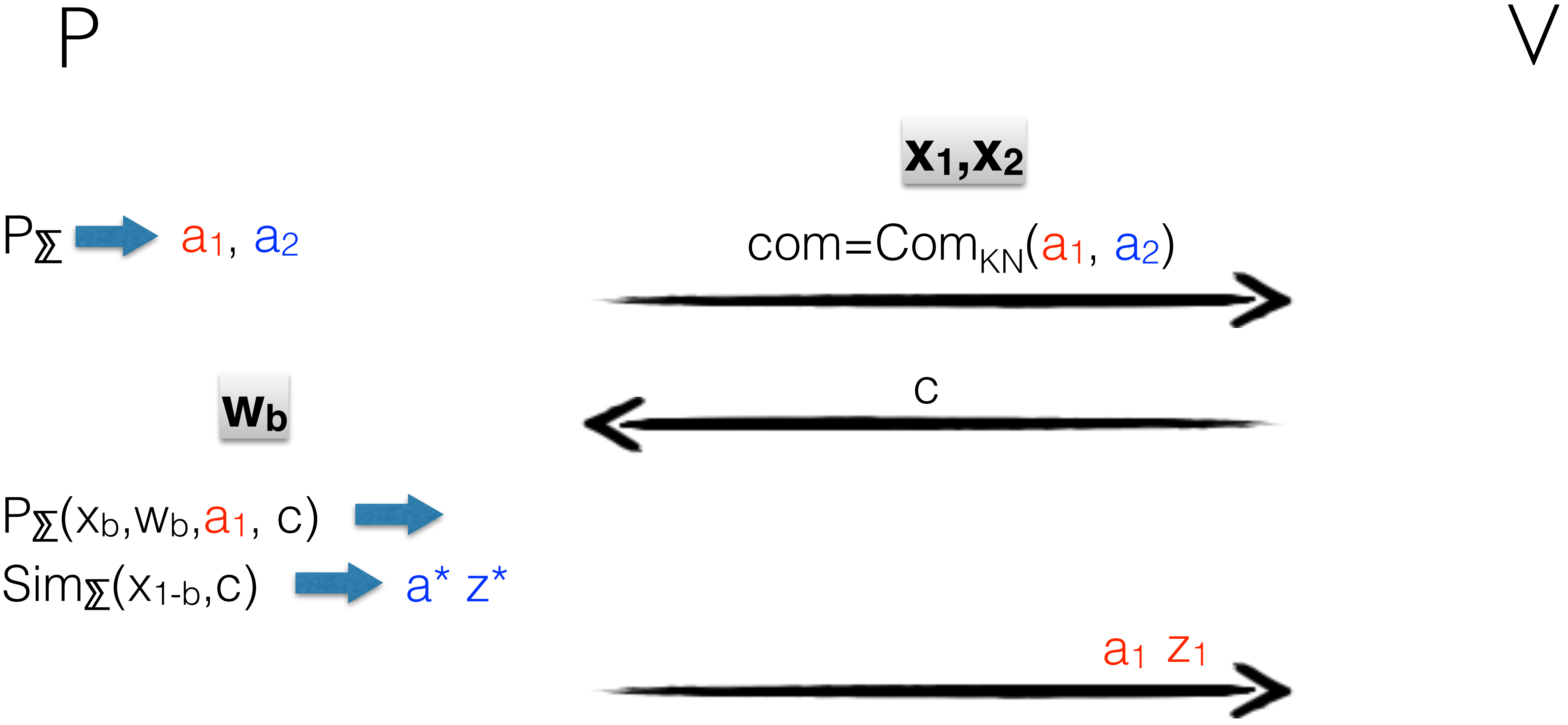
$P_\Sigma(x_b,w_b,a_1, c)$ →

a_1 z_1

Our Construction: Main Idea

e.g. k=1, n=2

$\mathbf{R}$ OR $\mathbf{R}$



P

V

x_1,x_2

P_Σ → a_1, a_2

com=$\mathrm{Com}_{KN}(a_1, a_2)$

c

w_b

$P_\Sigma(x_b,w_b,a_1, c)$ →

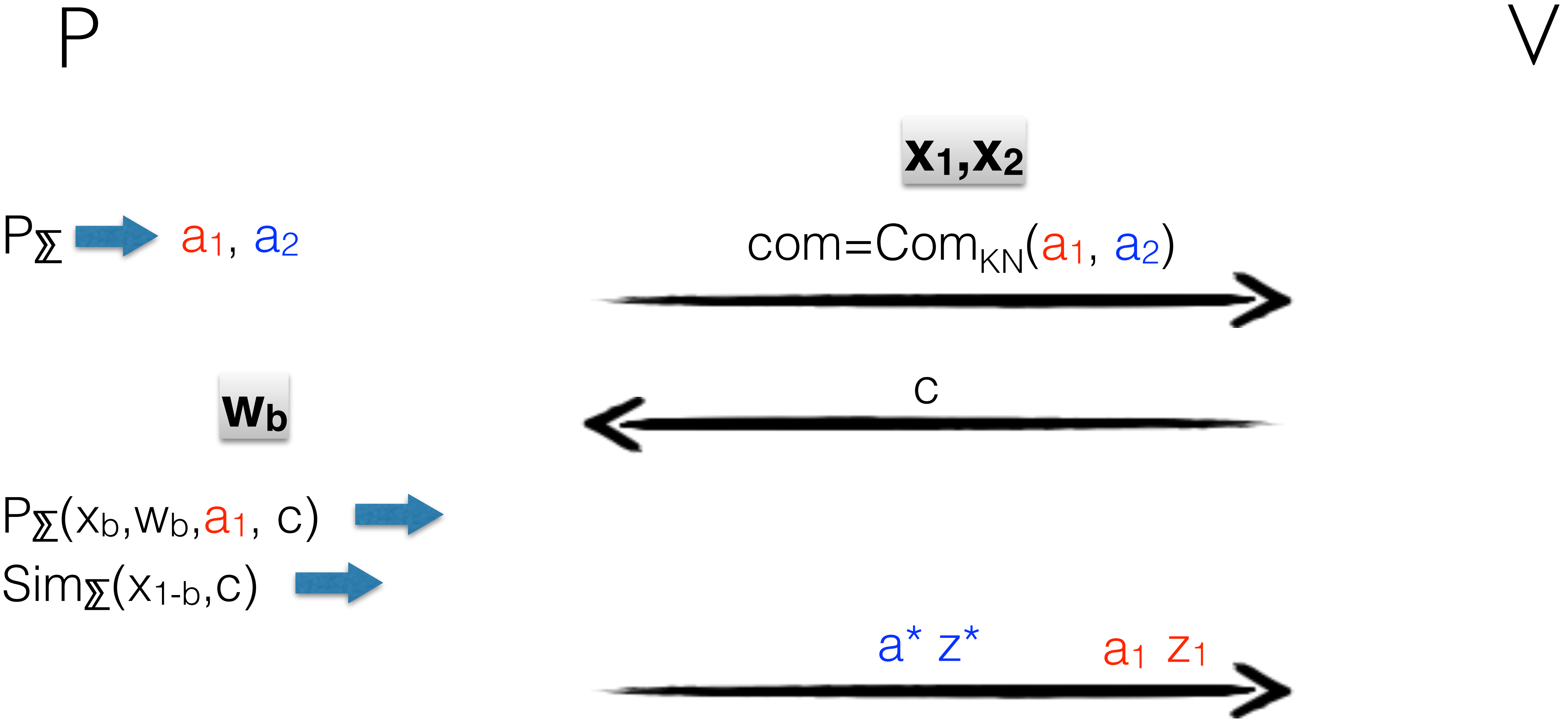
$\mathrm{Sim}_\Sigma(x_{1-b},c)$ → a^* z^*

a_1 z_1

Our Construction: Main Idea

e.g. k=1, n=2

R OR R



P

V

x_1,x_2

P_Σ → a_1, a_2

com=$\mathrm{Com}_{KN}(a_1, a_2)$

c

w_b

$P_\Sigma(x_b,w_b,a_1, c)$ →

$\mathrm{Sim}_\Sigma(x_{1-b},c)$ →

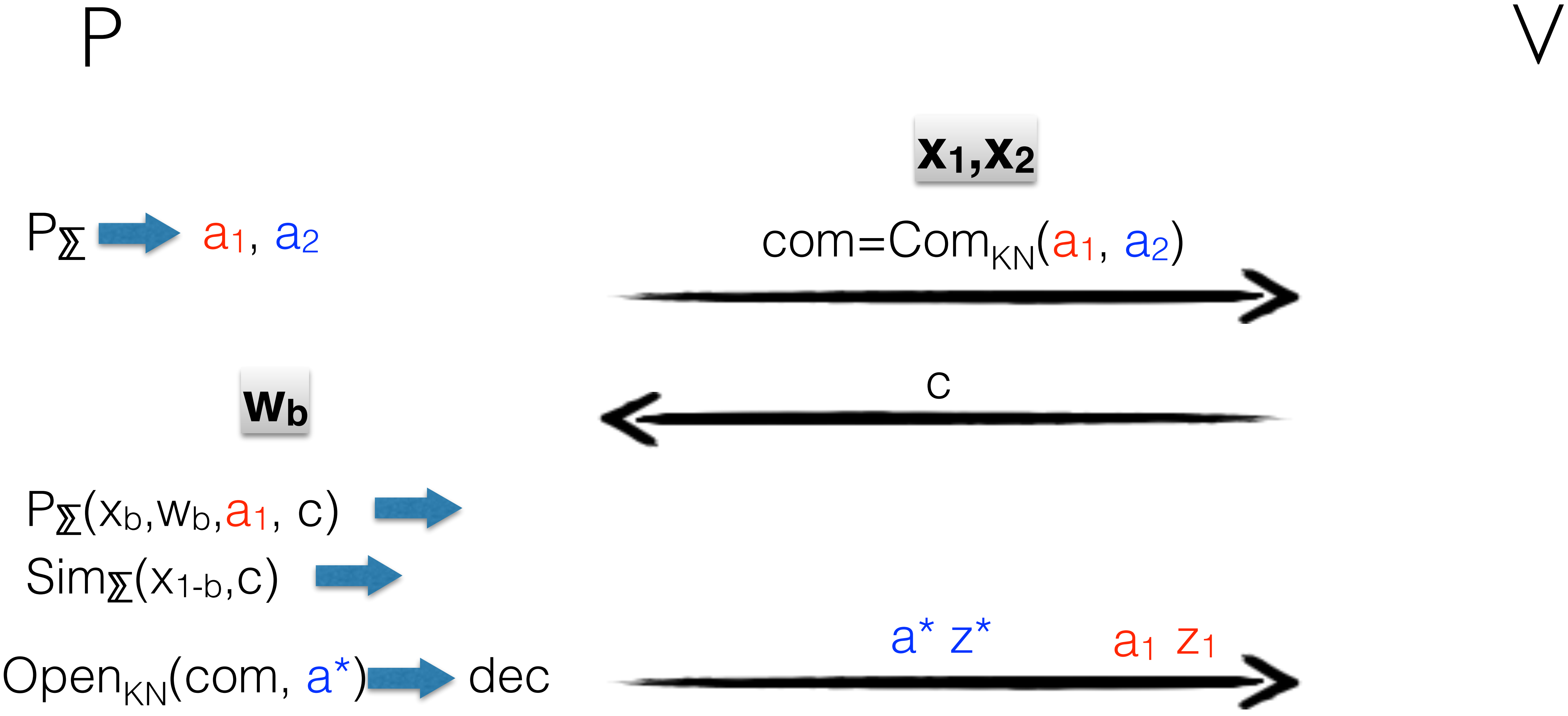
a* z*

a_1 z_1

Our Construction: Main Idea

e.g. k=1, n=2

R OR R



P

V

$\mathbf{x_1,x_2}$

P_Σ → a_1, a_2

com=Com$_{KN}$(a_1, a_2)

$\mathbf{w_b}$

c

$P_\Sigma(x_b, w_b, a_1, c)$ →

$Sim_\Sigma(x_{1-b}, c)$ →

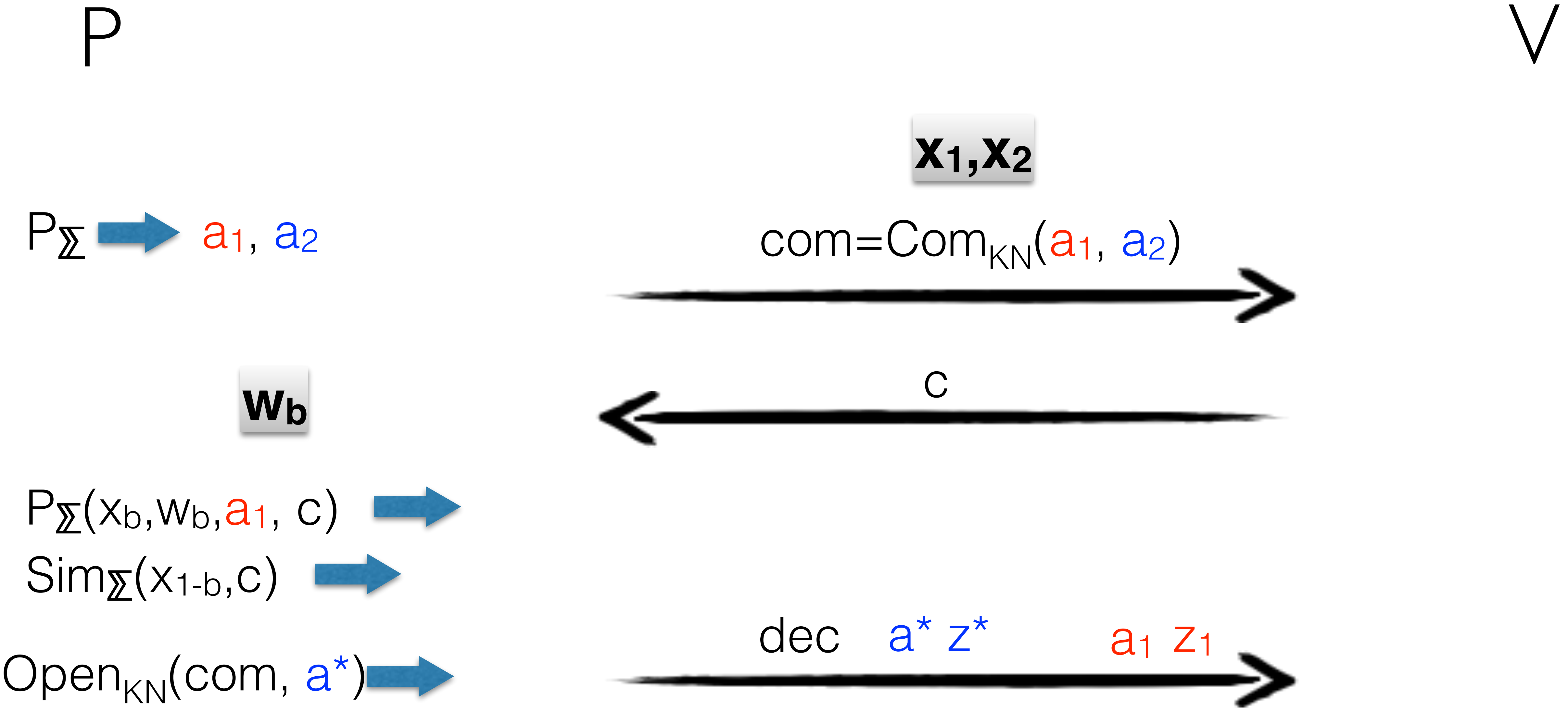
Open$_{KN}$(com, a^*) → dec

a^* z^* a_1 z_1

Our Construction: Main Idea

e.g. k=1, n=2

R OR R



P

V

x_1,x_2

P_Σ → a_1, a_2

com=$\mathrm{Com}_{KN}(a_1, a_2)$

c

w_b

$P_\Sigma(x_b,w_b,a_1, c)$ →

$\mathrm{Sim}_\Sigma(x_{1-b},c)$ →

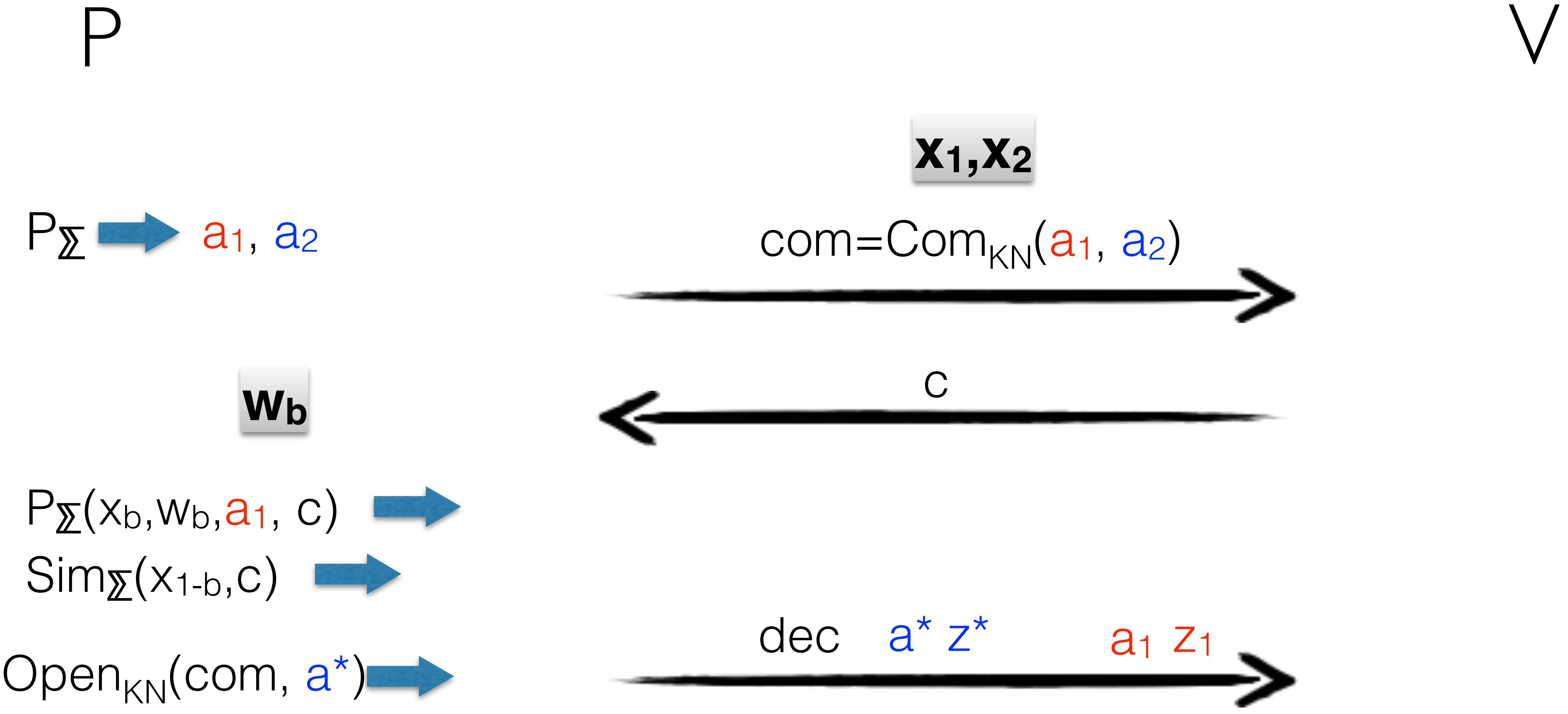
$\mathrm{Open}_{KN}(\mathrm{com}, a^*)$ →

dec a^* z^* a_1 z_1

Our Construction: Main Idea

e.g. k=1, n=2

R OR R



P

V

x_1, x_2

P_Σ → a_1, a_2

com=$\text{Com}_{KN}(a_1, a_2)$

c

w_b

$P_\Sigma(x_b, w_b, a_1, c)$ →

$\text{Sim}_\Sigma(x_{1-b}, c)$ →

$\text{Open}_{KN}(\text{com}, a^*)$ →

dec a^* z^* a_1 z_1

-Is dec a valid opening for a^*
and a_1 w.r.t com?

-Is (a^*, c ,z^*) accepting for ∑?

-Is (a_1, c ,z_1) accepting for ∑?

How to Construct an Efficient (K,N) Trapdoor Commitment

Ingredient 1: DDH

$$(g^{a},g^{b},g^{ab}) \approx (g^{a},g^{b},g^{c})$$

How to Construct an Efficient (K,N) Trapdoor Commitment

Ingredient 1: DDH

$$(g^{a},g^{b},g^{ab}) \approx (g^{a},g^{b},g^{c})$$

↑

DH tuple

How to Construct an Efficient (K,N) Trapdoor Commitment

Ingredient 1: DDH

$$(g^{a},g^{b},g^{ab}) \approx (g^{a},g^{b},g^{c})$$

↑ ↑

DH tuple non-DH tuple

How to Construct an Efficient (K,N) Trapdoor Commitment

Ingredient 2: Instance dependent trapdoor commitment (**IDTC**) from DDH

Given **T**=(g^{a},g^{b},g^{c})

Com(**T**,m) ⇒ dec, com

If T is **NDH** →

Binding Perfect
Hiding Computational

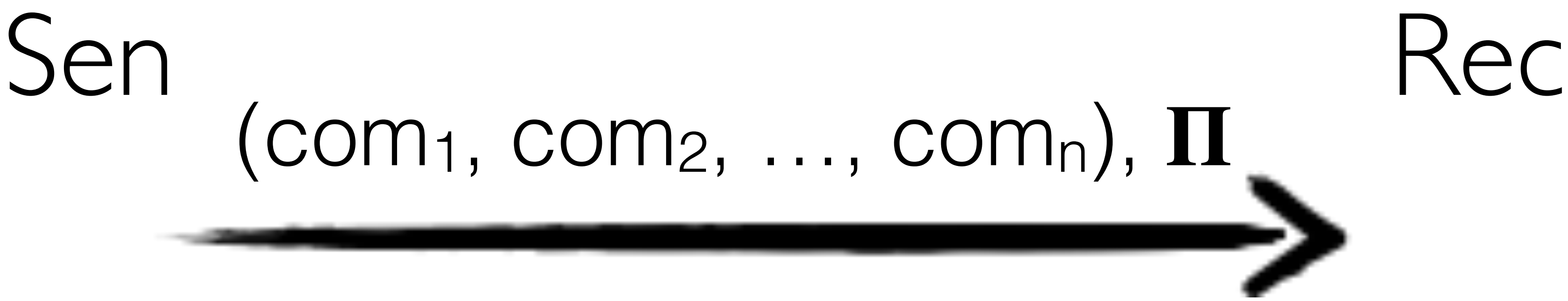
If T is **DH** →

Binding Computational

Equivocal

Constructions of IDTC follow directly from known constructions of Trapdoor Commitments from ∑-Protocols [Dam10, HL10, DN02]

How to Construct an Efficient (K,N) Trapdoor Commitment



Sen

Rec

$(\mathrm{com}_1, \mathrm{com}_2, \ldots, \mathrm{com}_n), \boldsymbol{\Pi}$

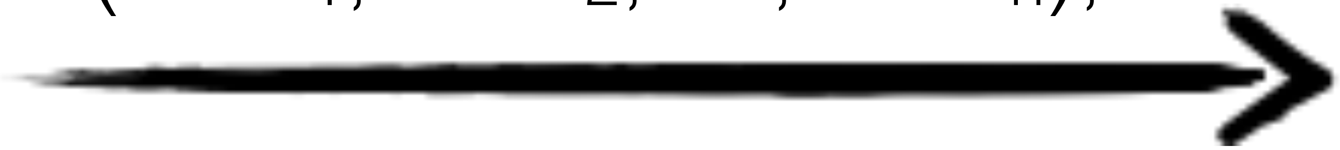
How to Construct an Efficient (K,N) Trapdoor Commitment

At least k are perfectly binding

Sen

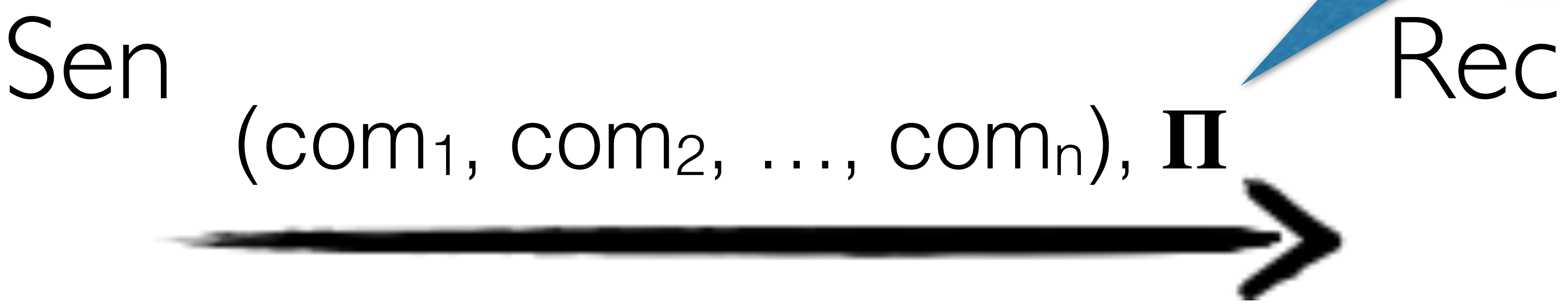
Rec

$(\mathrm{com}_1, \mathrm{com}_2, \ldots, \mathrm{com}_n), \mathbf{\Pi}$



How to Construct an Efficient (K,N) Trapdoor Commitmen

At least k are perfectly binding



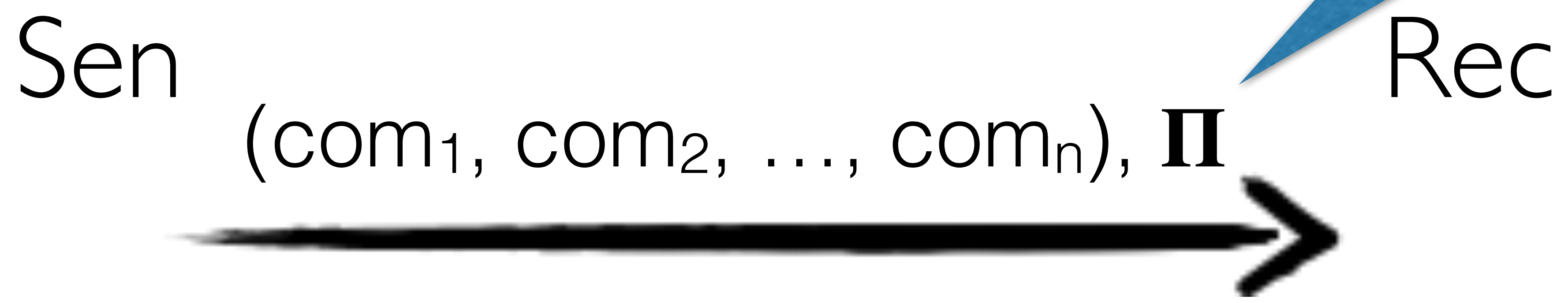
Sen

Rec

$(\mathrm{com}_1, \mathrm{com}_2, \ldots, \mathrm{com}_n), \mathbf{\Pi}$

1) Select $T_1, T_2, \ldots, T_n$

How to Construct an Efficient (K,N) Trapdoor Commitment



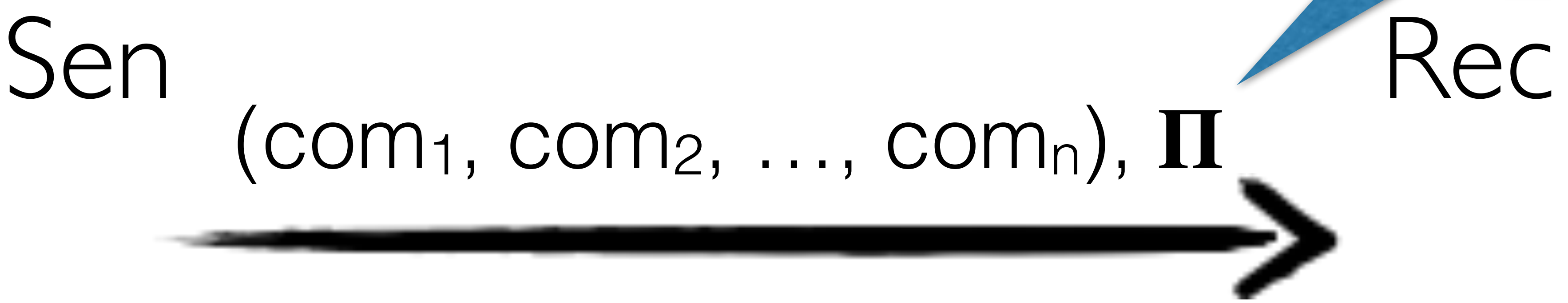
At least k are perfectly binding

1) Select T_1, T_2, …, T_n

2) Run Com($\mathbf{T_i}$,m_i) ⇒ (dec$_i$,com$_i$) **for** i=1,…,n and send (com$_1$, com$_2$, …, com$_n$)

How to Construct an Efficient (K,N) Trapdoor Commitment

At least k are perfectly binding



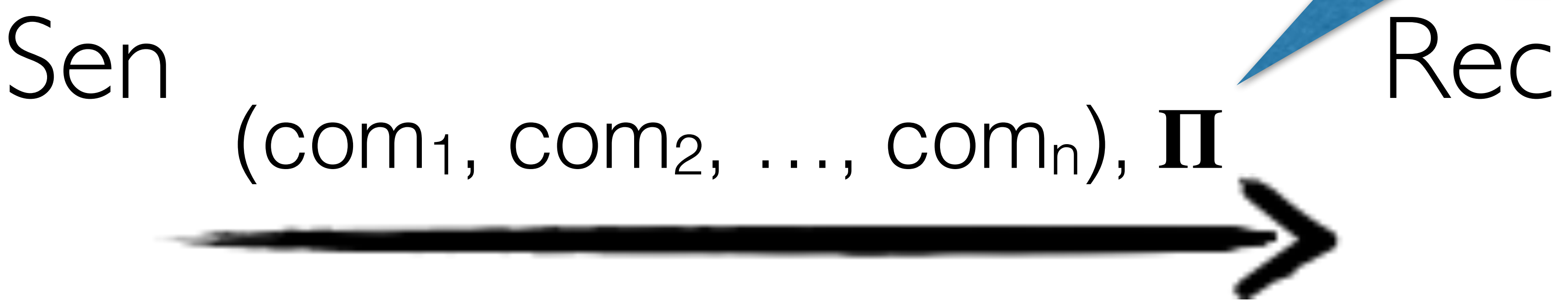
1) Select $T_1, T_2, \ldots, T_n$

2) Run $Com(\mathbf{T_i}, m_i) \Rightarrow (dec_i, com_i)$ **for** $i=1,\ldots,n$ and send $(com_1, com_2, \ldots, com_n)$

3) Prove with $\mathbf{\Pi}$ that at least **k** of the n tuples $T_1, T_2, \ldots, T_n$ are **non-DH** (prove with [CDS94])

How to Construct an Efficient (K,N) Trapdoor Commitment

At least k are perfectly binding



1) Select $T_1, T_2, \ldots, T_n$

2) Run $Com(\mathbf{T_i}, m_i) \Rightarrow (dec_i, com_i)$ **for** $i=1,\ldots,n$ and send $(com_1, com_2, \ldots, com_n)$

3) Prove with Π that at least **k** of the n tuples $T_1, T_2, \ldots, T_n$ are **non-DH** (prove with [CDS94])

Prove with **Π** that at least **k** of the n tuples $T_1, T_2, \ldots, T_n$ are **non-DH**

e.g. k=1, n=2

$\mathbf{T_1}=(g^{a_1},g^{b_1},g^{c_1})$

$\mathbf{T_2}=(g^{a_2},g^{b_2},g^{c_2})$

Prove with **Π** that at least **k** of the n tuples $T_1, T_2, \ldots, T_n$ are **non-DH**

e.g. k=1, n=2

$$\mathbf{T_1}=(g^{a_1},g^{b_1},g^{c_1}) \qquad \mathbf{T_1}'=(g^{a_1},g^{b_1},g^{a_1 \cdot b_1})$$

→

$$\mathbf{T_2}=(g^{a_2},g^{b_2},g^{c_2}) \qquad \mathbf{T_2}'=(g^{a_2},g^{b_2},g^{c_3})$$

Prove with Π that at least **k** of the n tuples $T_1, T_2, \ldots, T_n$ are **non-DH**

e.g. k=1, n=2

$\mathbf{T_1}=(g^{a_1},g^{b_1},g^{c_1})$

$\mathbf{T_2}=(g^{a_2},g^{b_2},g^{c_2})$

→

$\mathbf{T_1}'=(g^{a_1},g^{b_1},g^{a_1 \cdot b_1})$

$\mathbf{T_2}'=(g^{a_2},g^{b_2},g^{c_3})$

P ↔ V

Π^{CDS94}: "**T_1**' is DH **OR** **T_2**' is DH"

Prove with Π that at least **k** of the n tuples $T_1, T_2, \ldots, T_n$ are **non-DH**

e.g. k=1, n=2

$$\mathbf{T_1}=(g^{a_1},g^{b_1},g^{c_1}) \qquad\qquad \mathbf{T_1}'=(g^{a_1},g^{b_1},g^{a_1 \bullet b_1})$$

$$\Rightarrow$$

$$\mathbf{T_2}=(g^{a_2},g^{b_2},g^{c_2}) \qquad\qquad \mathbf{T_2}'=(g^{a_2},g^{b_2},g^{c_3})$$

P $\qquad$ Π^{CDS94}: "$\mathbf{T_1}$' is DH **OR** $\mathbf{T_2}$' is DH" $\qquad$ V

$$\longleftrightarrow$$

V accepts ⇔ One out of

T_1, **T_2** is a non-DH tuple

More Results of Our Work

- Our previous construction works for any (**k**,**n**)
- In the paper you can also find a construction that works for different NP-relations (e.g. R_{Dlog} or R_{DH}) (This construction is non-trivial ad uses as a sub-protocol the construction showed before)
- We give also a compiler that transform a ∑-Protocol (belonging to the class described in [Cra96, Mau15, CD98]) in an **Adaptive-Input PoK**
- **Open problem**
  - Is it possible to extend adaptive PoK to a larger class of ∑-Protocols?

thanks