Zero-Knowledge Proofs for Secure and Private Machine Learning

Dario Fiore | IMDEA Software Institute

Foundations and Applications of Zero-Knowledge Proofs | Edinburgh, UK | Sep 6, 2024

Agenda

Security of ML inference

How to use ZKPs for secure ML

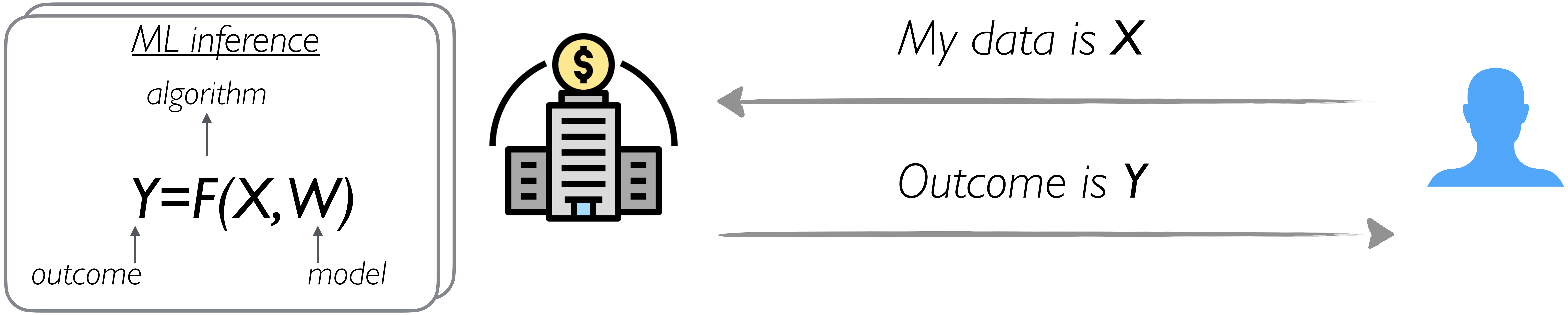
Efficiency challenges of ZKPs for ML

Efficient ZKPs for Neural Networks

Efficient ZKPs for Decision Trees

Conclusions

Motivation: outsourcing machine learning



ML inference

algorithm

$Y=F(X,W)$

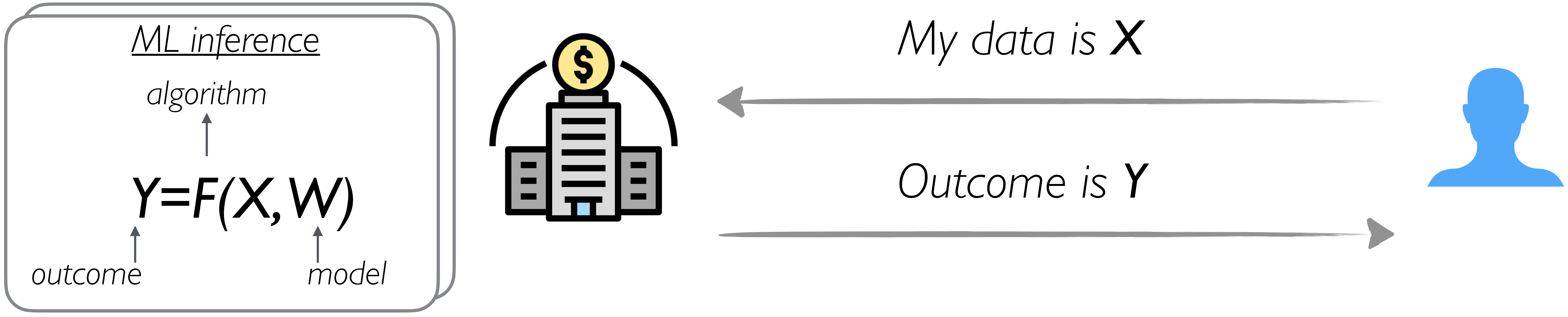
outcome

model

My data is X

Outcome is Y

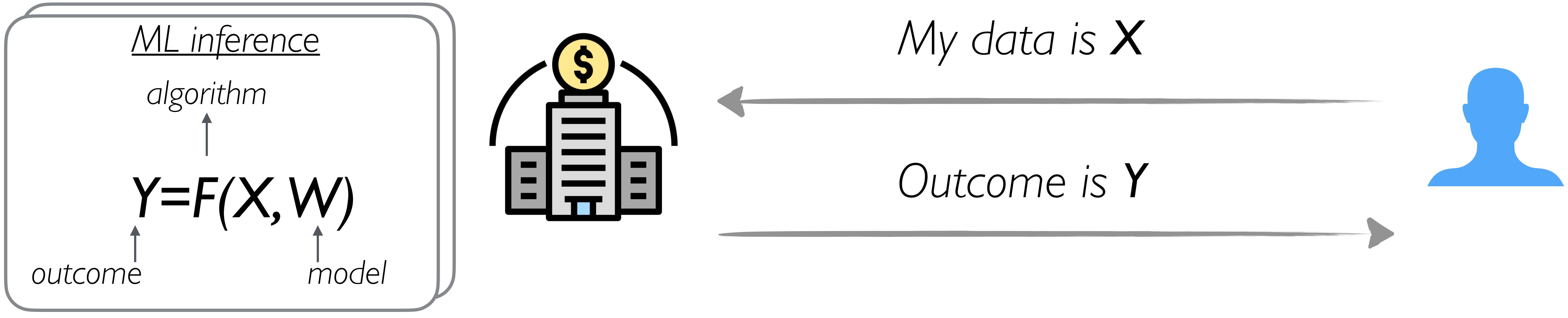
Motivation: outsourcing machine learning



Banking & Finance

Can I have a loan?

Motivation: outsourcing machine learning



<u>*ML inference*</u>

algorithm

Y=F(X,W)

outcome

model

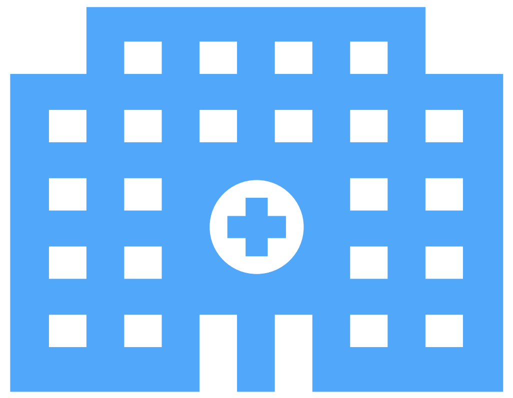
*My data is **X***

*Outcome is **Y***

Banking & Finance

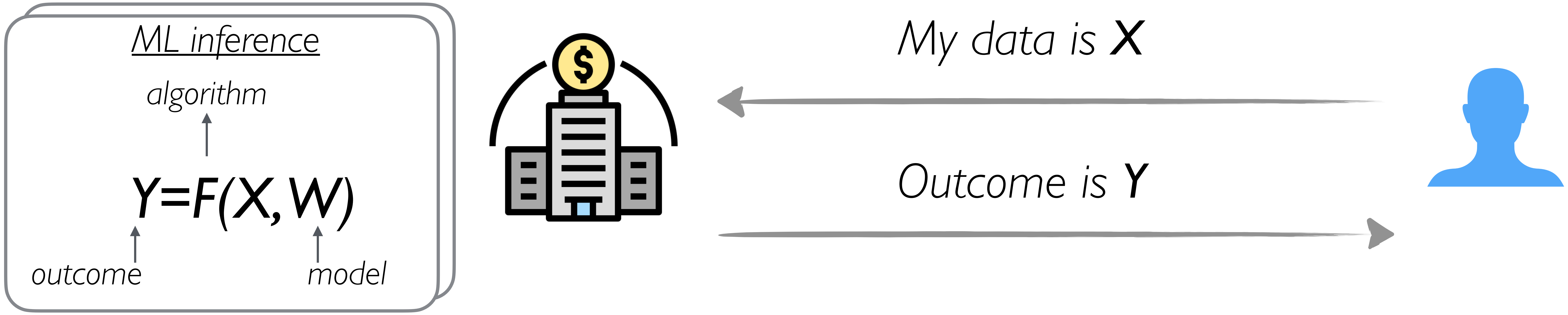
Can I have a loan?

Healthcare



Risk of a disease?

Motivation: outsourcing machine learning



ML inference

algorithm

$Y=F(X,W)$

outcome *model*

My data is X

Outcome is Y

Banking & Finance

Can I have a loan?

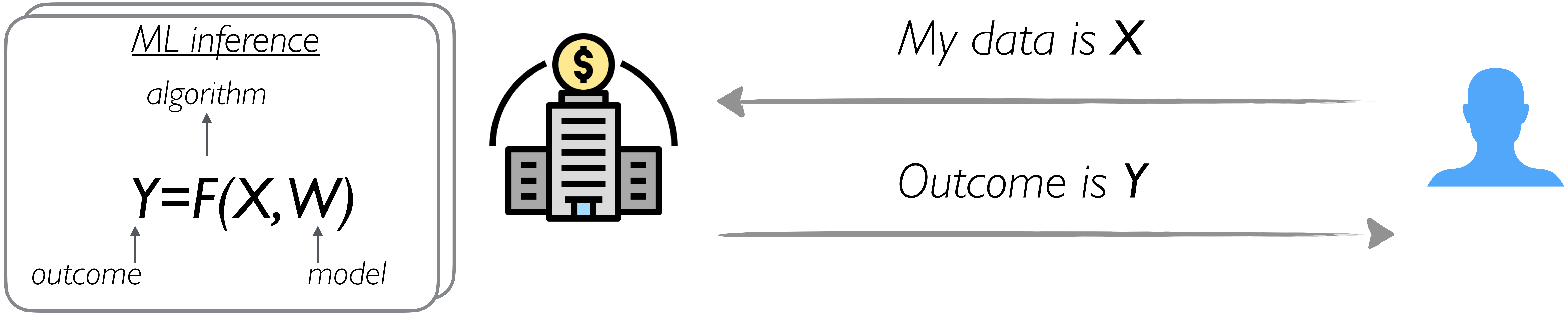
Healthcare

Risk of a disease?

Criminal justice

Released or retained?

Security of outsourced machine learning



ML inference

algorithm

$Y=F(X,W)$

outcome

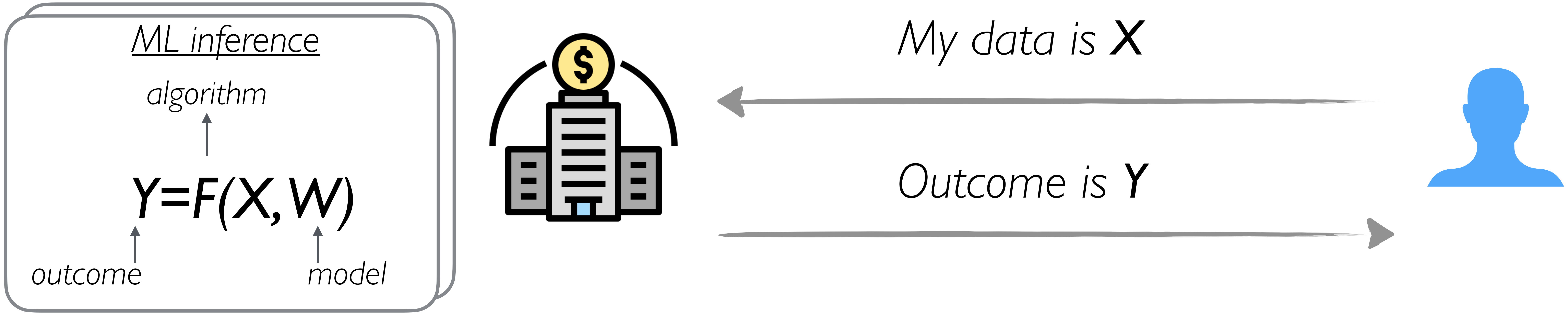
model

My data is X

Outcome is Y

Can I trust this?

Security of outsourced machine learning



ML inference

algorithm

Y=F(X,W)

outcome

model

My data is X

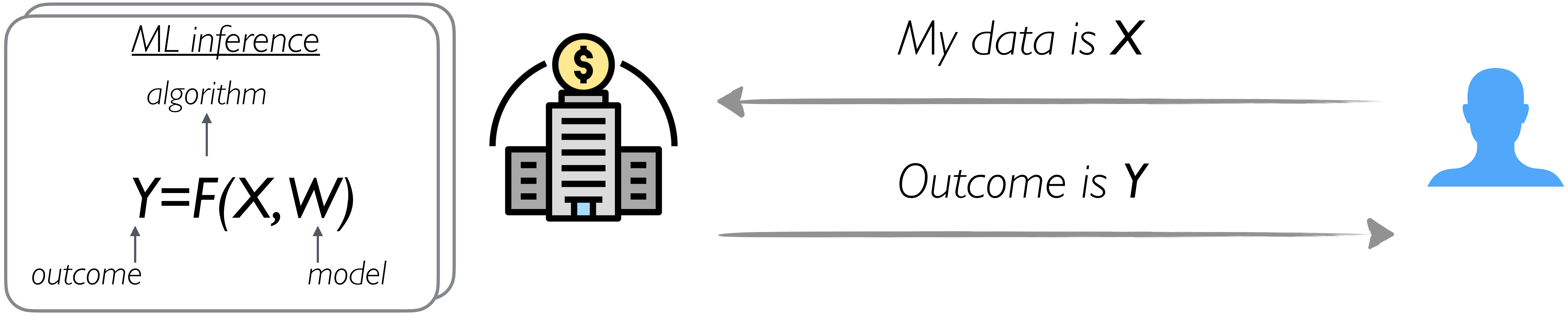
Outcome is Y

Can I trust this?

Goals

Integrity: detect tampered computations

Security of outsourced machine learning



ML inference

algorithm

$Y=F(X,W)$

outcome

model

My data is X

Outcome is Y

Can I trust this?

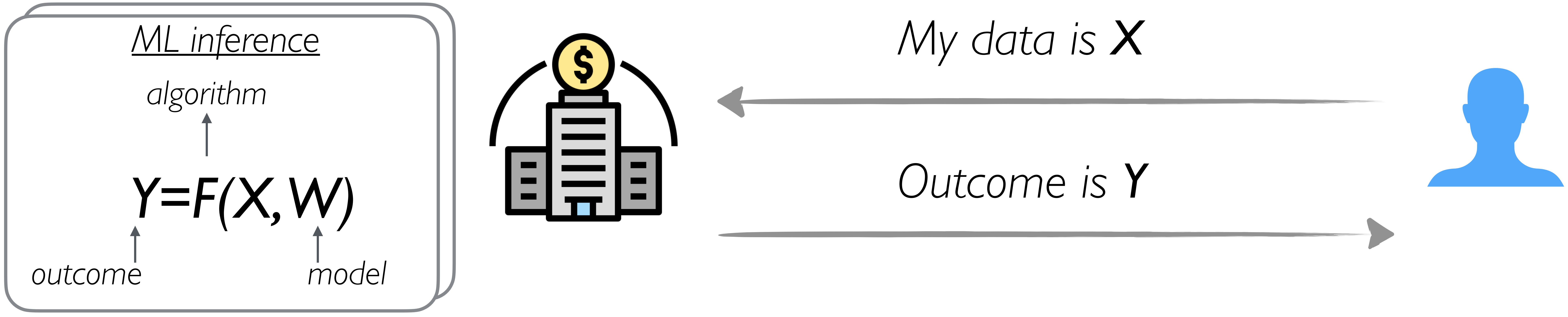
Is the model fair?

Goals

Integrity: detect tampered computations

Fairness: assuming the model is trusted, decision process is the same for all clients

Security of outsourced machine learning



ML inference

algorithm

$Y=F(X,W)$

outcome *model*

My data is X

Outcome is Y

Can I trust this?

Is the model fair?

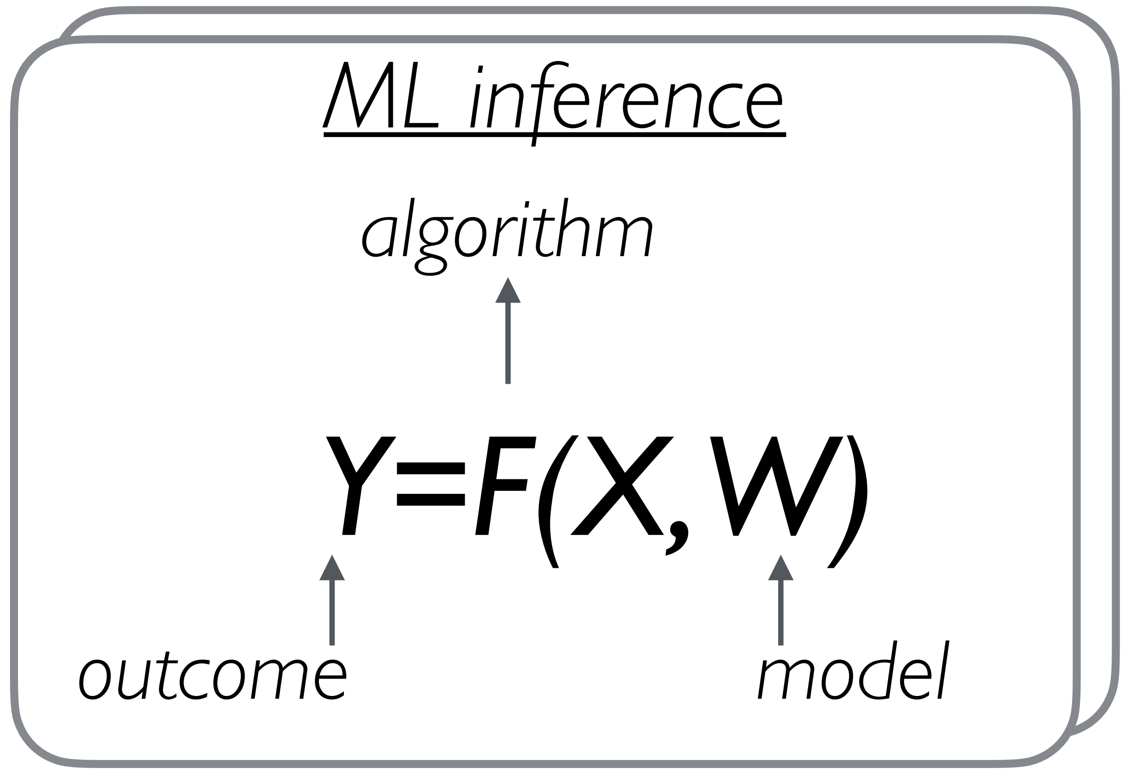
Goals

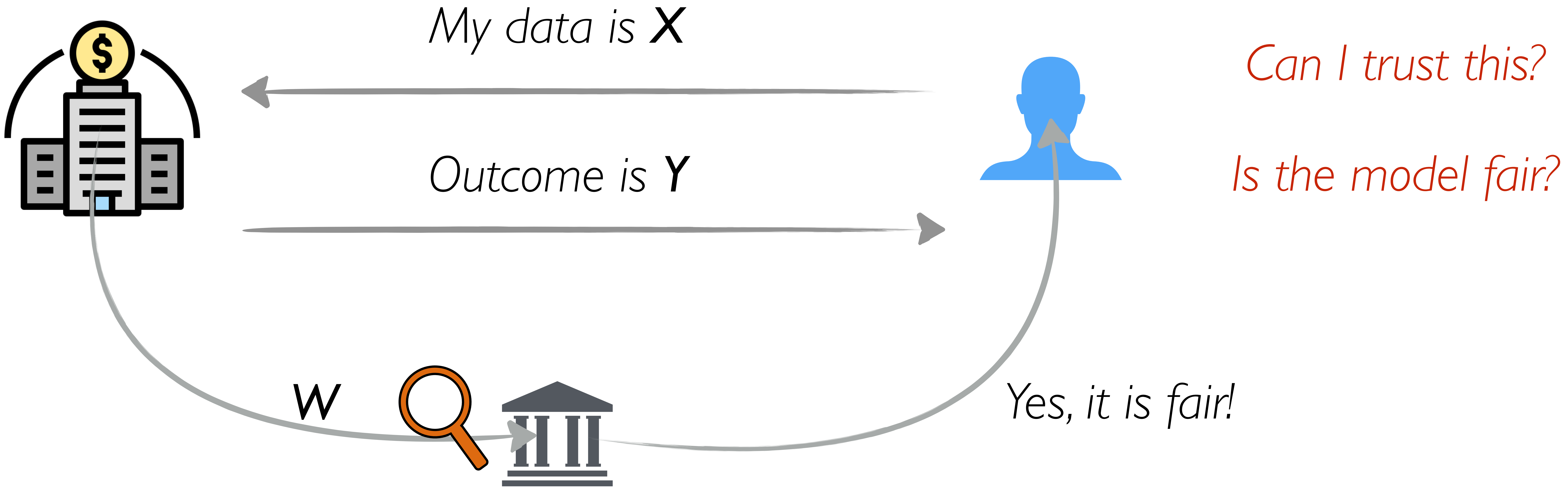
Integrity: detect tampered computations

Fairness: assuming the model is trusted, decision process is the same for all clients

Privacy: clients should not learn anything about the model *W*

Security of outsourced machine learning





Can I trust this?

Is the model fair?

Goals

Integrity: detect tampered computations

Fairness: assuming the model is trusted, decision process is the same for all clients

Privacy: clients should not learn anything about the model W

Secure and Private ML Inference

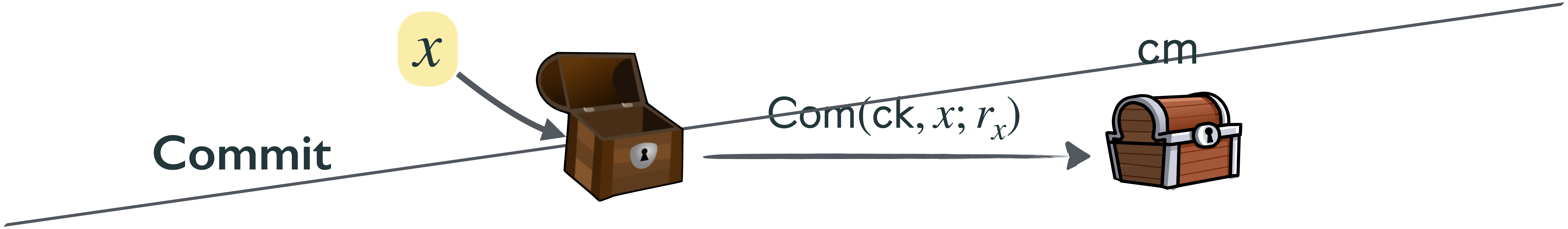
Ingredients

Zero Knowledge Proofs

Commitments

Commitments

[Blum, Even '81]



Commit

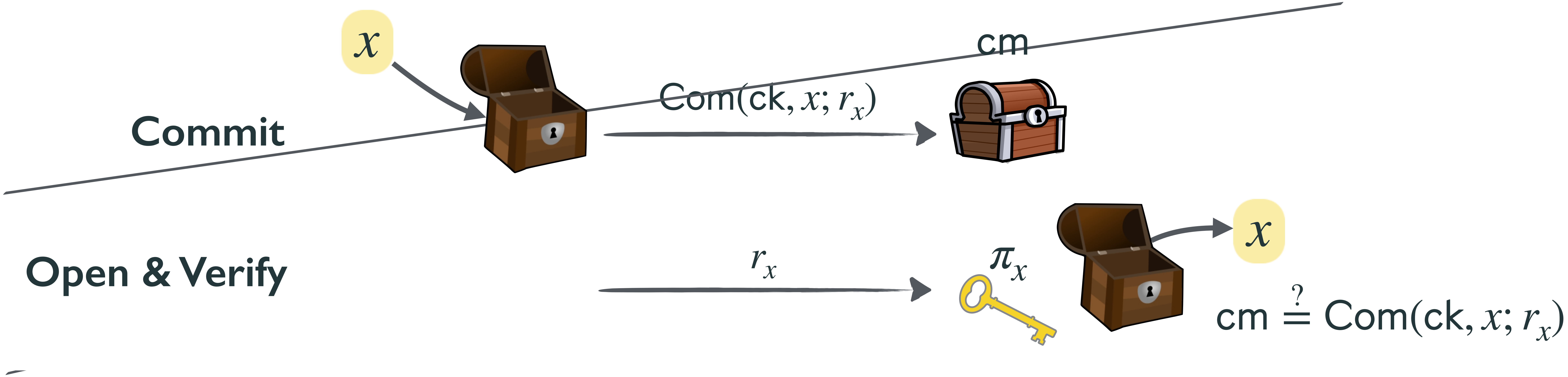
x

$\mathsf{Com}(\mathsf{ck}, x; r_x)$

cm

Commitments

[Blum, Even '81]



Commit

x

$\mathsf{Com}(\mathsf{ck}, x; r_x)$

cm

Open & Verify

r_x

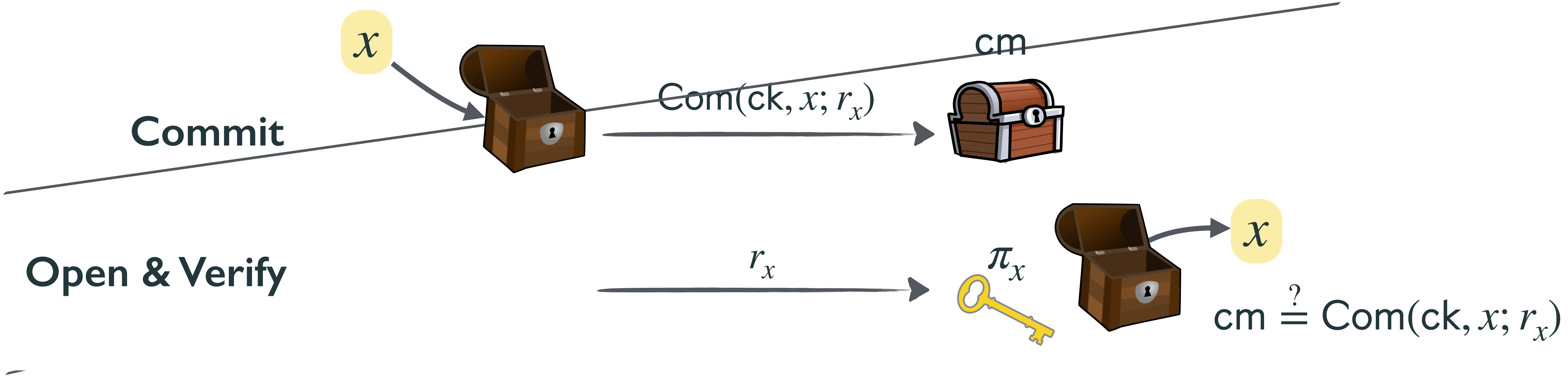
π_x

x

$\mathsf{cm} \overset{?}{=} \mathsf{Com}(\mathsf{ck}, x; r_x)$

Commitments

[Blum, Even '81]



Commit

x

$\mathsf{Com}(\mathsf{ck}, x; r_x)$

cm

Open & Verify

r_x

π_x

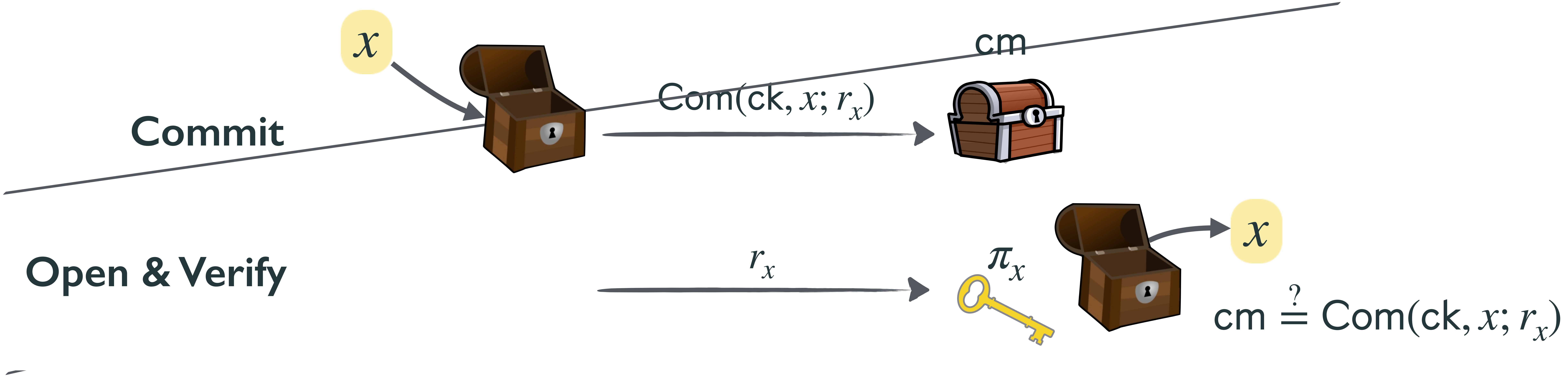
x

$\mathsf{cm} \overset{?}{=} \mathsf{Com}(\mathsf{ck}, x; r_x)$

- **Hiding:** $\mathsf{Com}(x) \approx \mathsf{Com}(x')$

Commitments

[Blum, Even '81]



Commit

x

$\mathrm{Com}(\mathrm{ck}, x; r_x)$

cm

Open & Verify

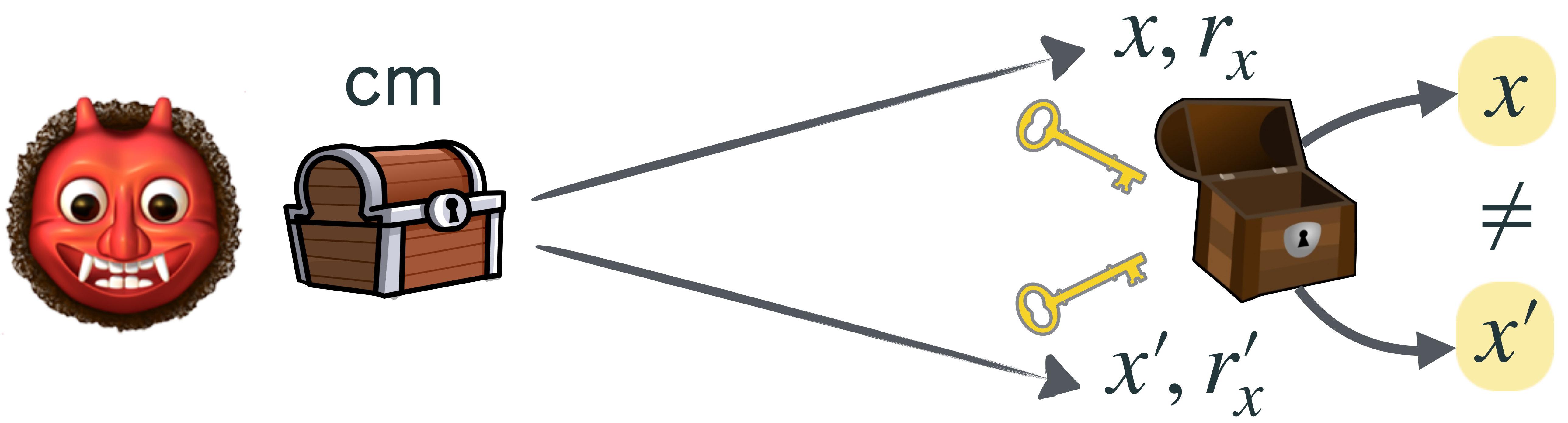
r_x

π_x

x

$\mathrm{cm} \overset{?}{=} \mathrm{Com}(\mathrm{ck}, x; r_x)$

- **Hiding:** $\mathrm{Com}(x) \approx \mathrm{Com}(x')$
- **Binding:**



cm

x, r_x

x', r'_x

x

$\neq$

x'

$\mathrm{cm} = \mathrm{Com}(\mathrm{ck}, x; r_x)$

$\mathrm{cm} = \mathrm{Com}(\mathrm{ck}, x'; r'_x)$

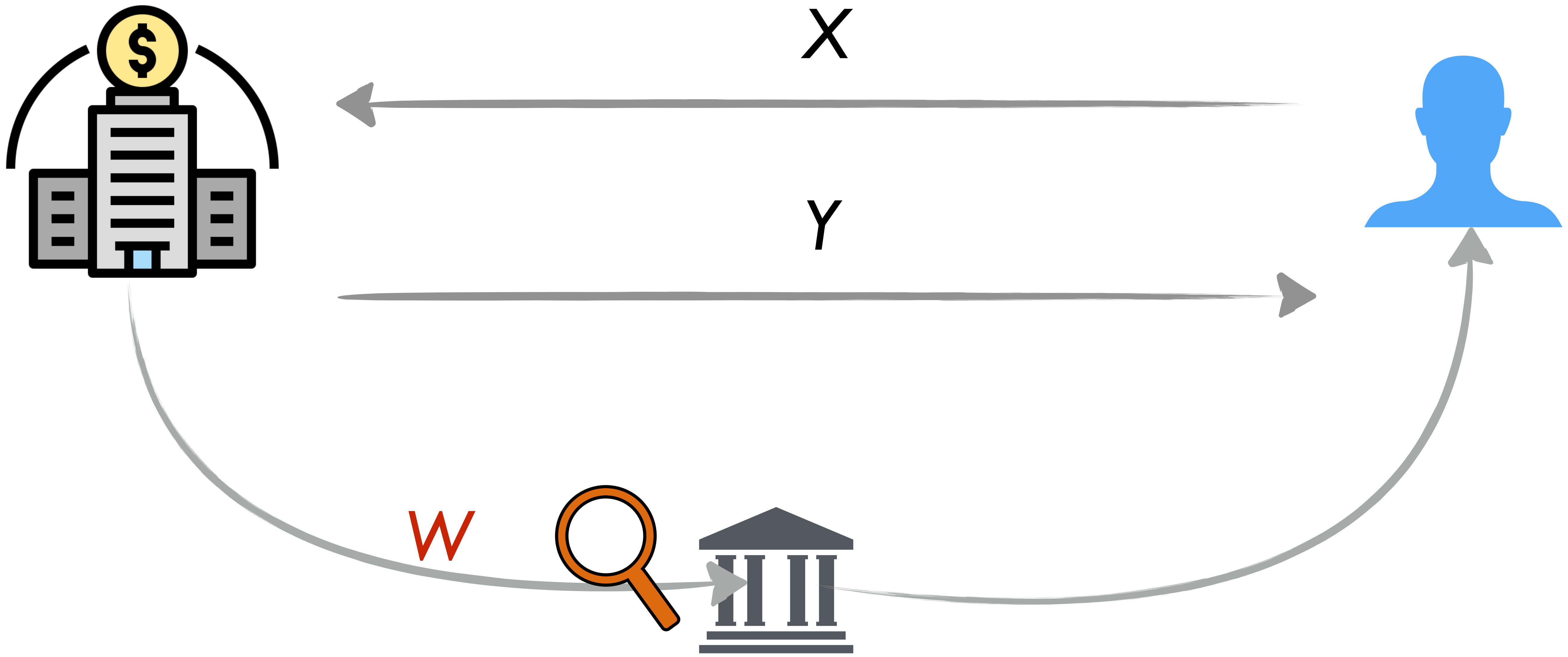
ZKPs for secure and private ML inference

ML inference

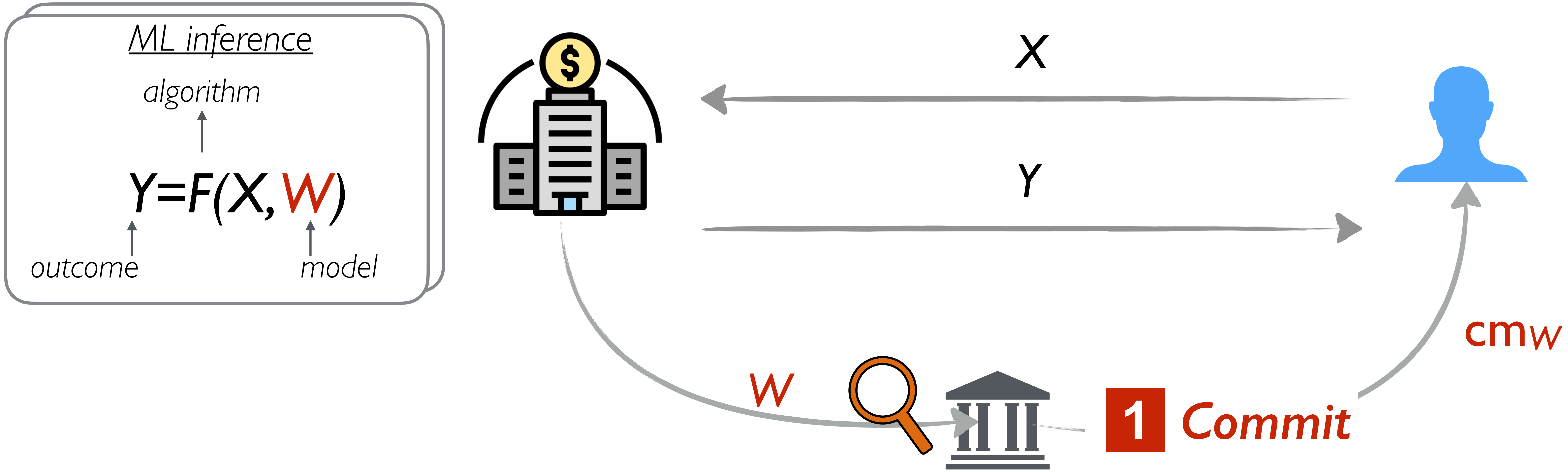
algorithm

$Y=F(X,W)$

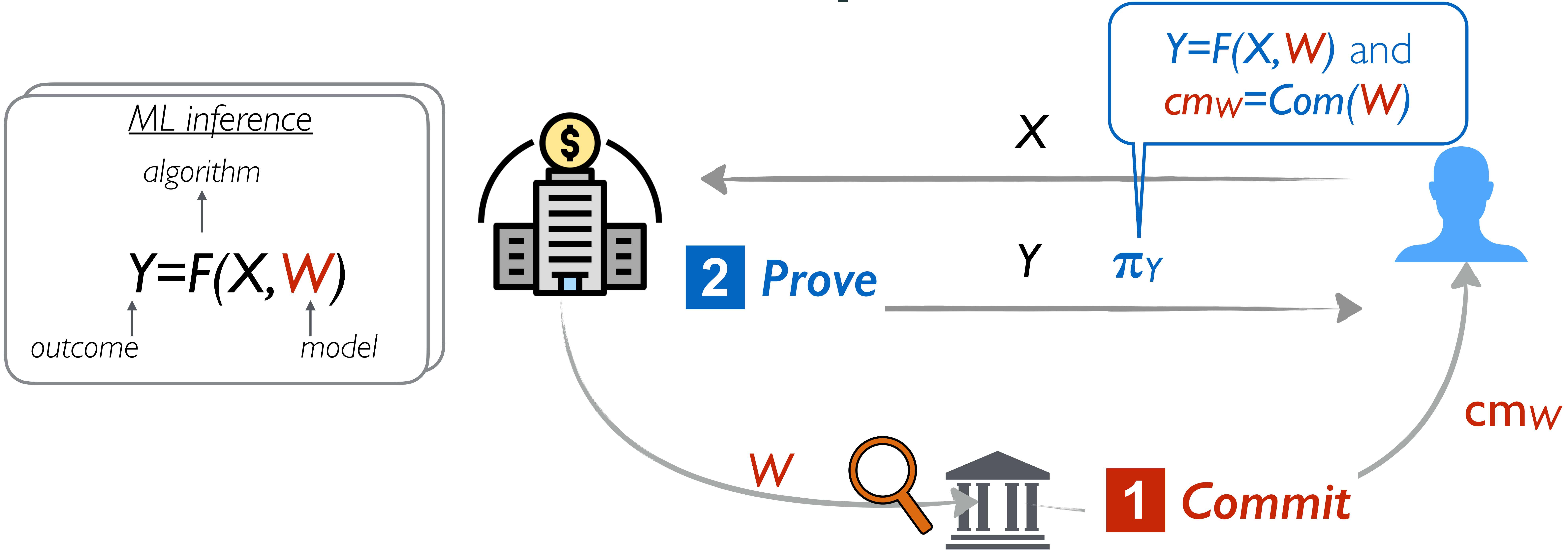
outcome *model*



ZKPs for secure and private ML inference



ZKPs for secure and private ML inference



ML inference

algorithm

$Y=F(X,W)$

outcome

model

$Y=F(X,W)$ and $cm_W=Com(W)$

X

Y

π_Y

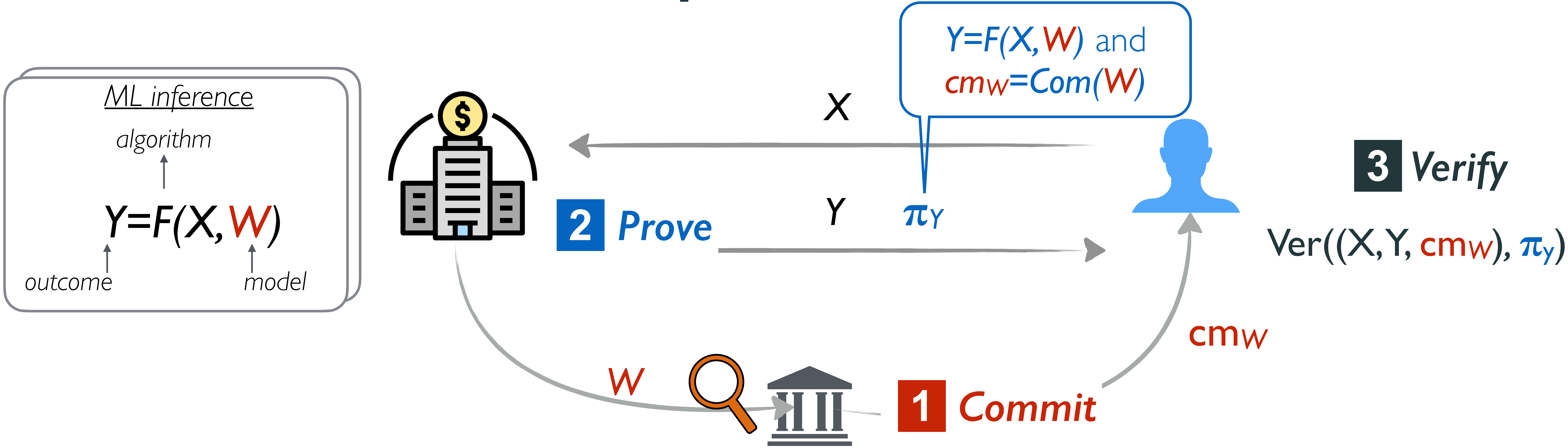
2 Prove

W

1 Commit

cm_W

ZKPs for secure and private ML inference



ML inference

algorithm

$Y=F(X,W)$

outcome

model

X

$Y=F(X,W)$ and $cm_W=Com(W)$

Y π_Y

2 Prove

3 Verify

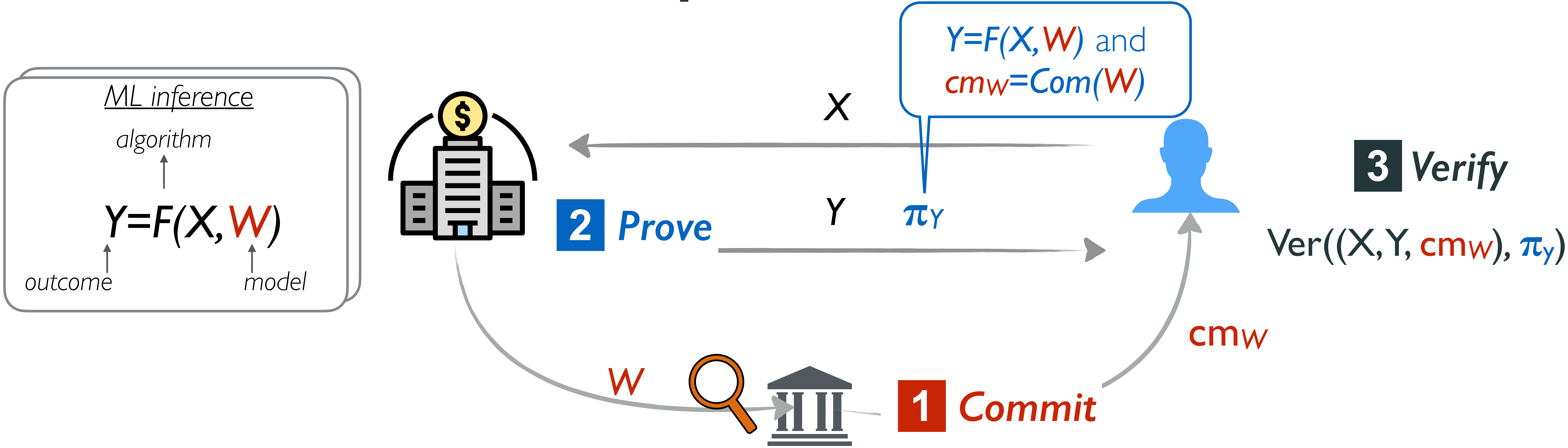
$\mathrm{Ver}((X,Y,cm_W),\pi_y)$

W

1 Commit

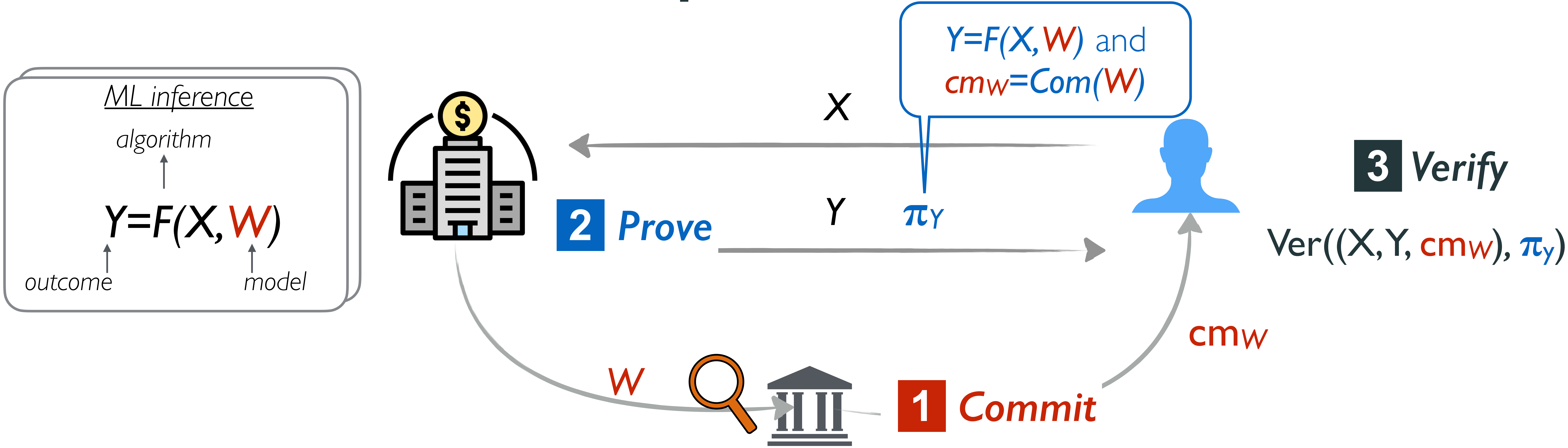
cm_W

ZKPs for secure and private ML inference



Integrity: detect tampered computations ⬅ **ZKP Soundness + Com binding**

ZKPs for secure and private ML inference



ML inference

algorithm

$Y=F(X,W)$

outcome

model

X

$Y=F(X,W)$ and $cm_W=Com(W)$

Y π_Y

2 *Prove*

3 *Verify*

$Ver((X,Y,cm_W),\pi_y)$

cm_W

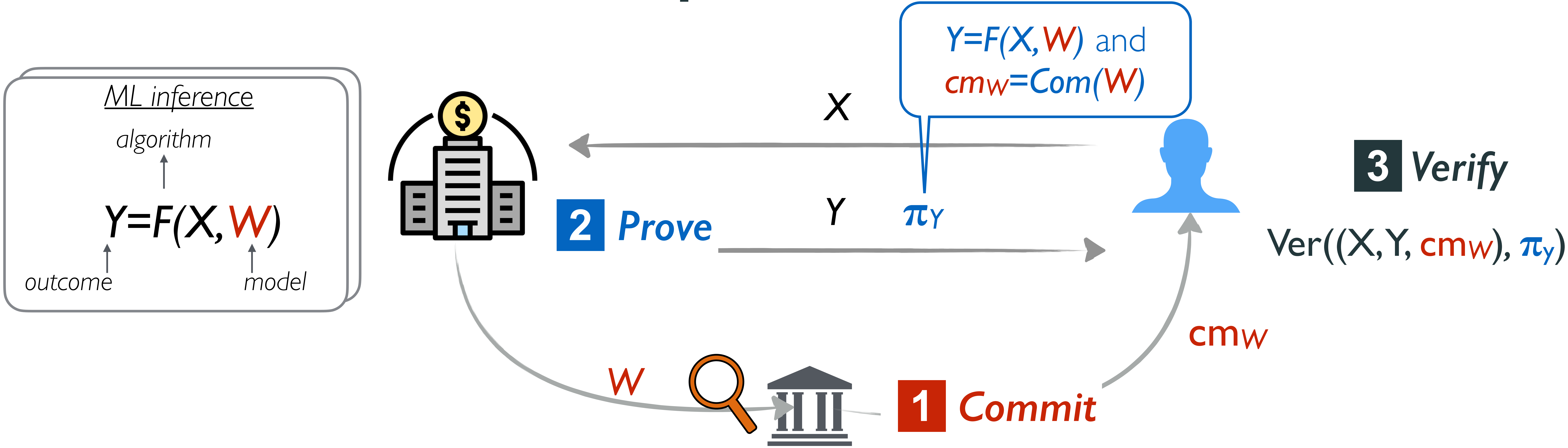
W

1 *Commit*

Integrity: detect tampered computations ⬅ **ZKP Soundness + Com binding**

Privacy: clients should not learn anything about the model W ⬅ **ZKP ZK + Com hiding**

ZKPs for secure and private ML inference

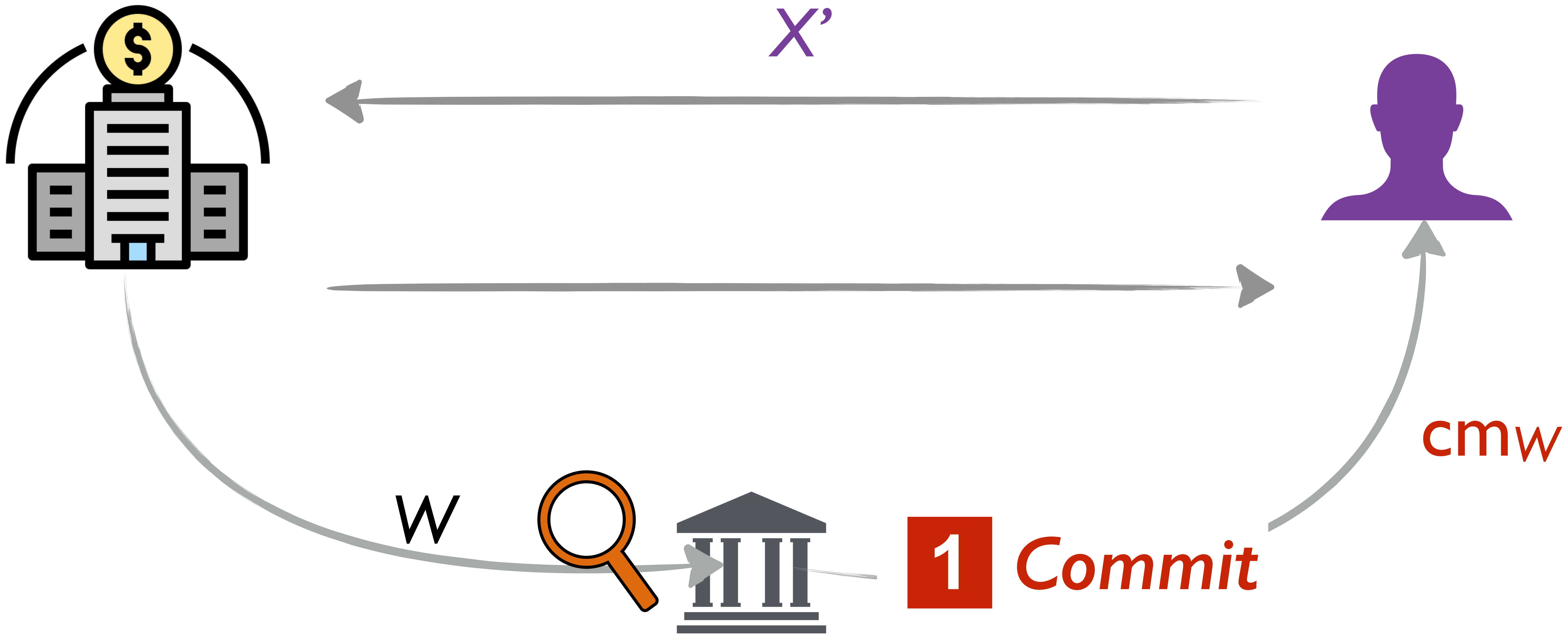


Integrity: detect tampered computations ⬅ **ZKP Soundness + Com binding**

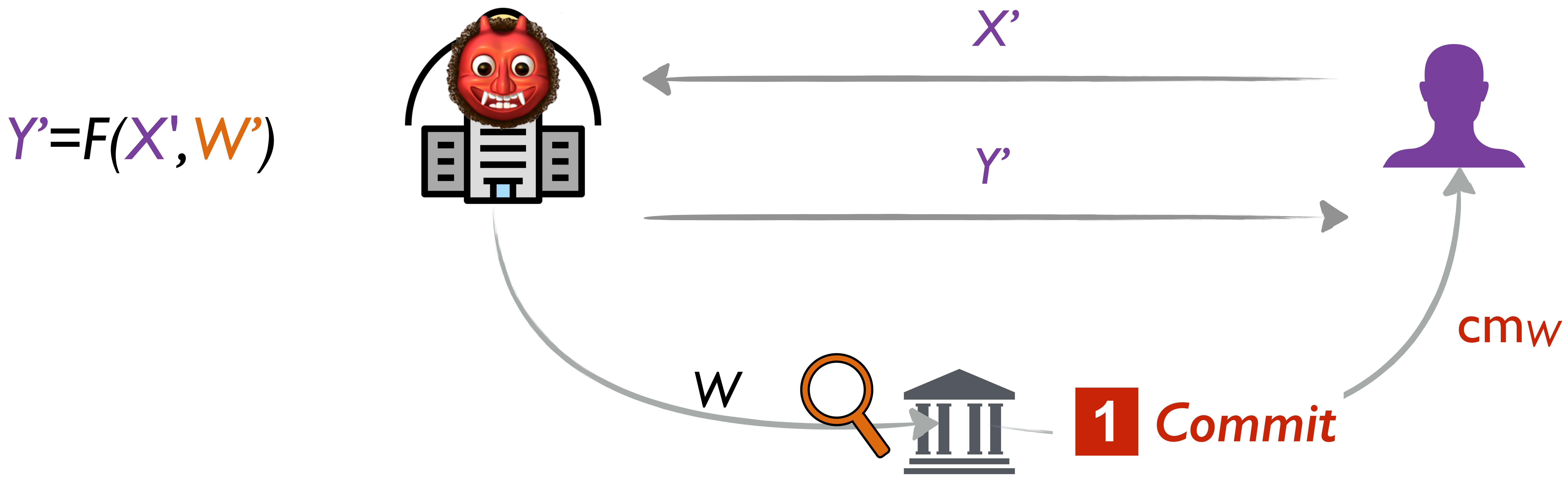
Privacy: clients should not learn anything about the model W ⬅ **ZKP ZK + Com hiding**

Fairness: decision process the same for all clients…

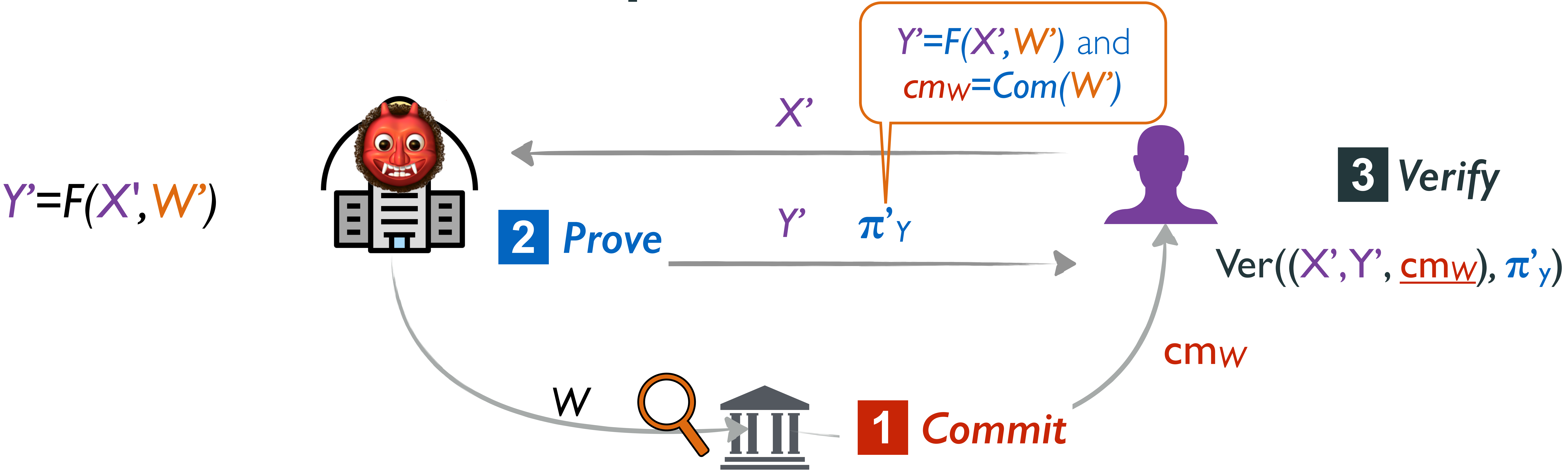
ZKPs for secure and private ML inference



ZKPs for secure and private ML inference



ZKPs for secure and private ML inference



$Y' = F(X', W')$

X'

$Y' = F(X', W')$ and $cm_W = Com(W')$

2 *Prove*

Y' π'_Y

3 *Verify*

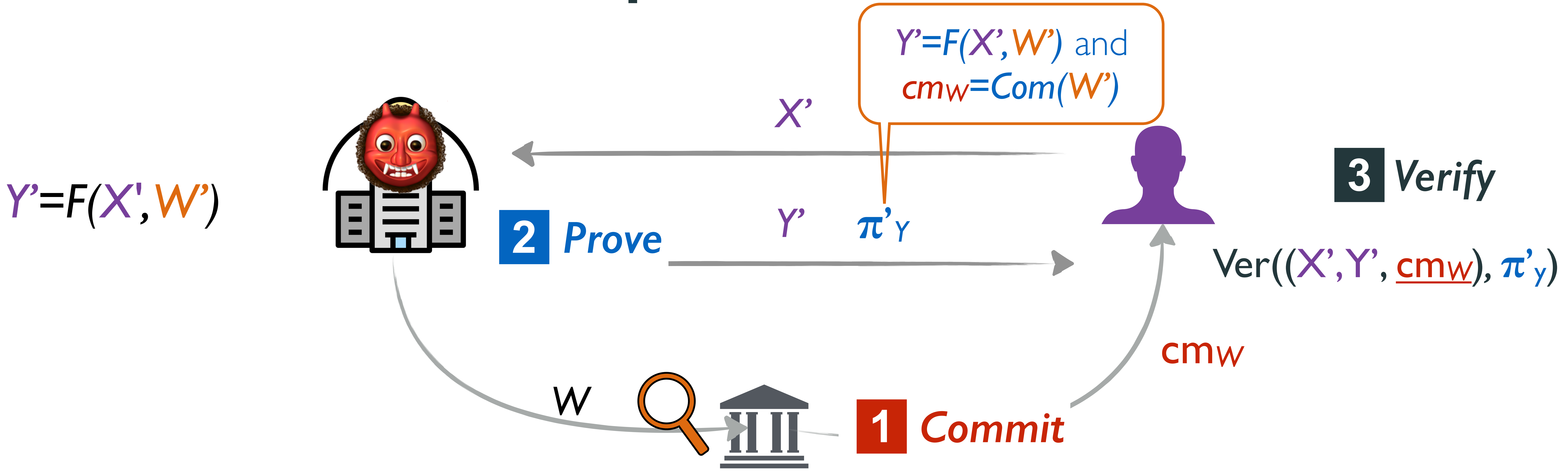
$Ver((X', Y', \underline{cm_W}), \pi'_y)$

W

cm_W

1 *Commit*

ZKPs for secure and private ML inference



$Y'=F(X',W')$

X'

$Y'=F(X',W')$ and $cm_W=Com(W')$

2 *Prove*

Y' π'_Y

3 *Verify*

$Ver((X',Y',cm_W),\pi'_y)$

cm_W

W

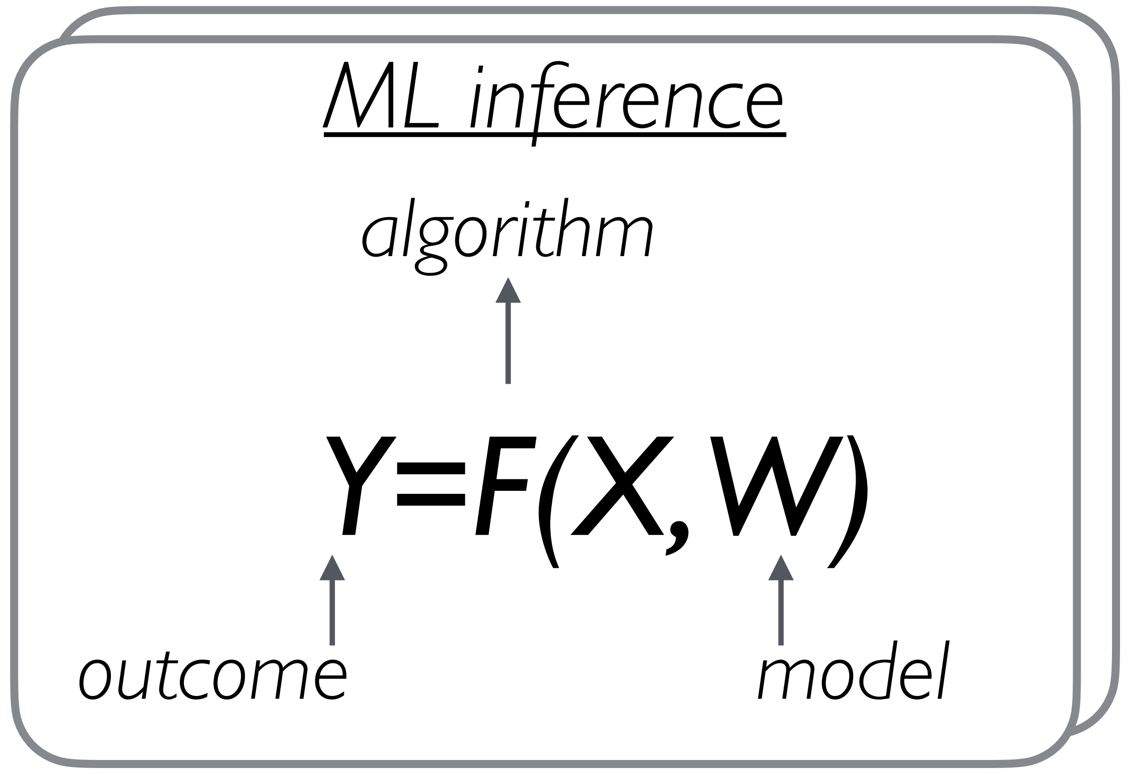
1 *Commit*

Integrity: detect tampered computations ← **ZKP Soundness + Com binding**

Privacy: clients should not learn anything about the model W ← **ZKP ZK + Com hiding**

Fairness: decision process the same for all clients ← **ZKP Soundness + Com binding**

Practical challenges of constructing ZKPs for ML

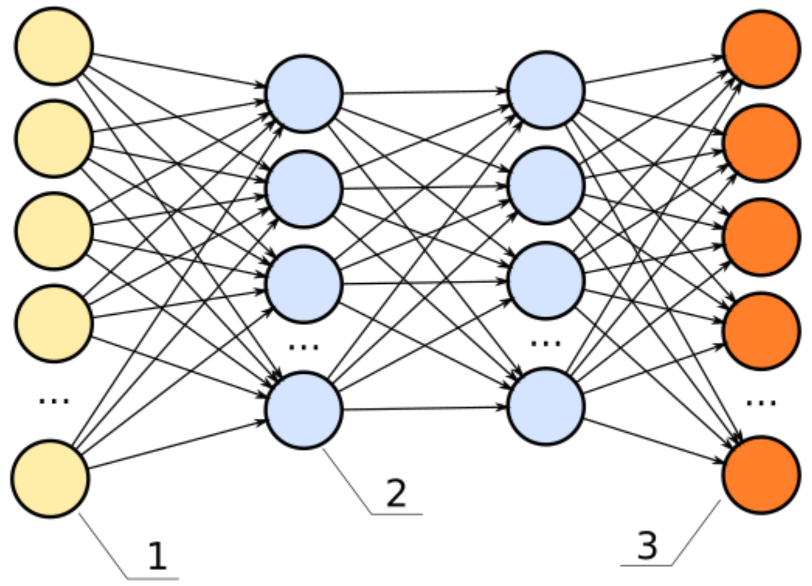


ML inference

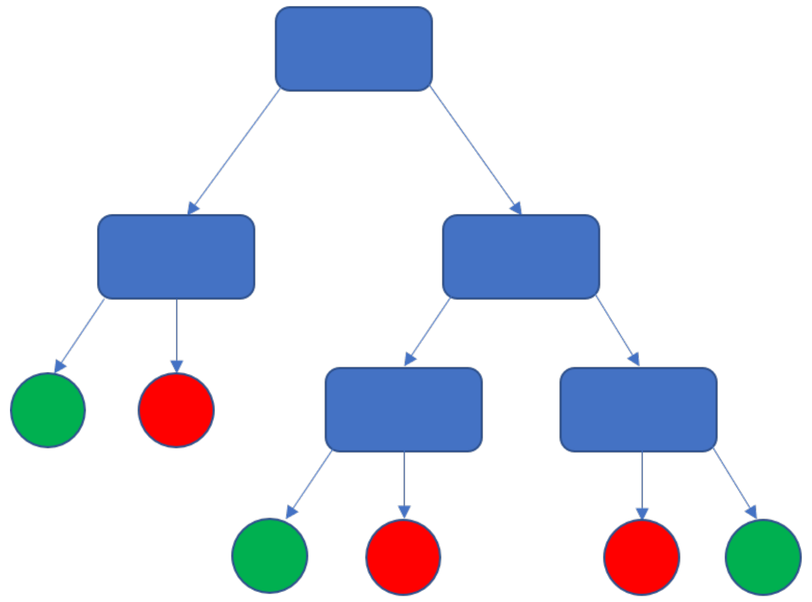
algorithm

$Y=F(X,W)$

outcome *model*



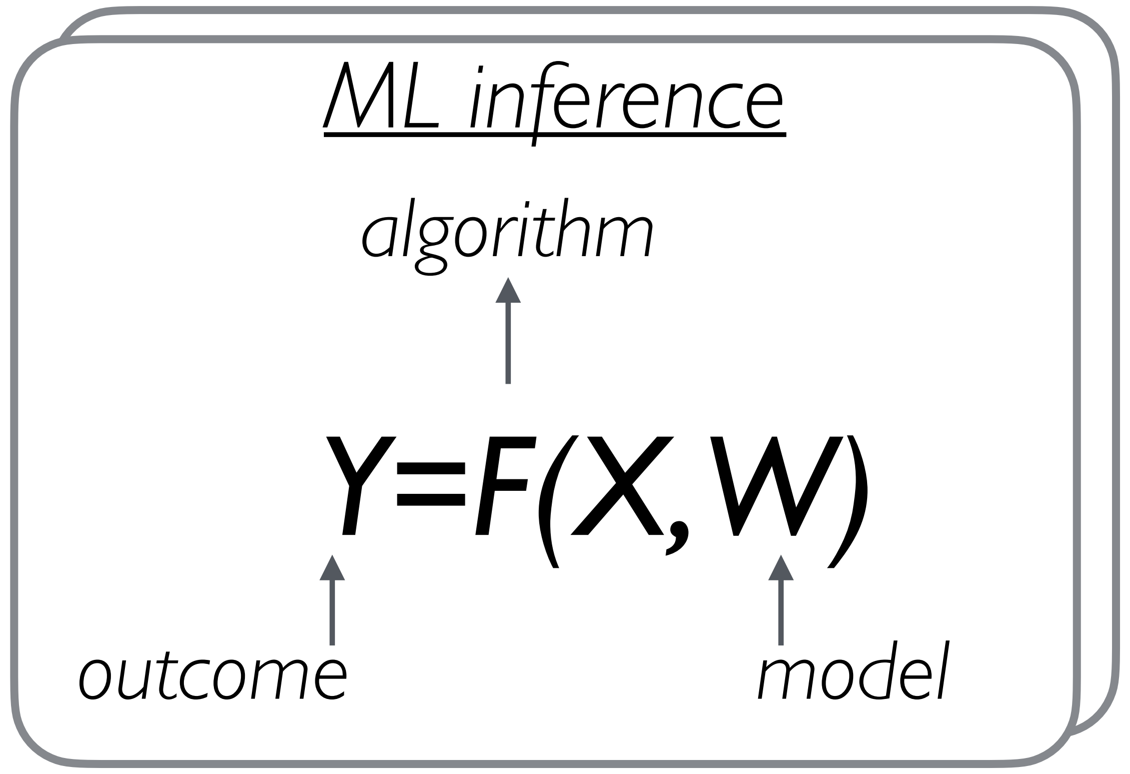
Neural Networks



Decision Trees

Scale with **large models** and **not-that-ZKP-friendly** computations

Practical challenges of constructing ZKPs for ML



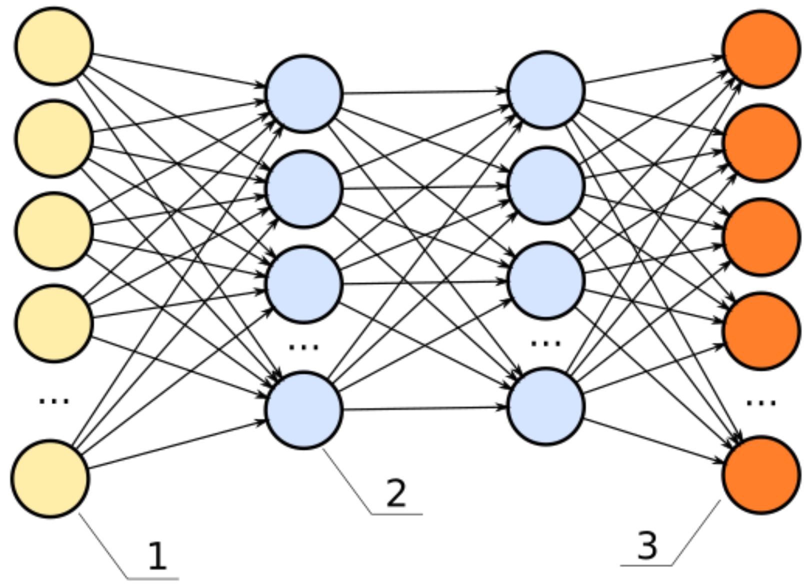
ML inference

algorithm

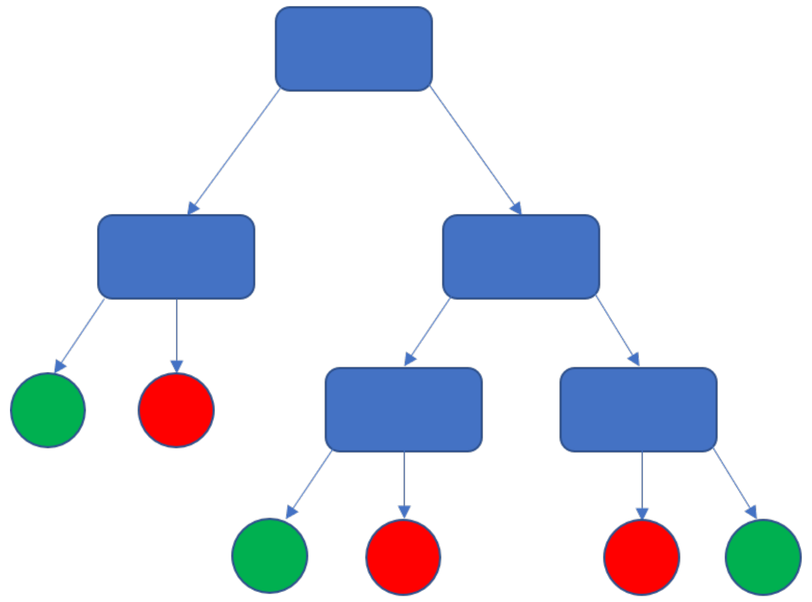
$Y=F(X,W)$

outcome

model



Neural Networks



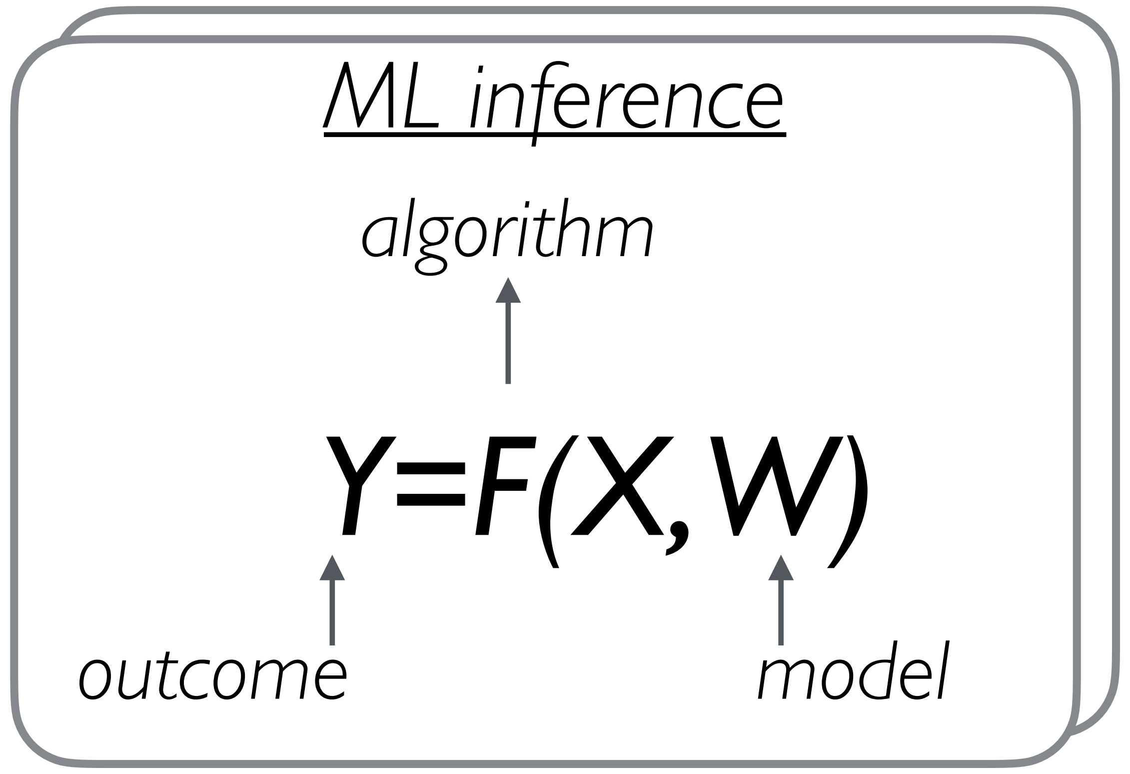
Decision Trees

Scale with **large models** and **not-that-ZKP-friendly** computations

VGG16 (one of the best computer vision NN) has ~500MB parameters

Dense matrix operations, non-linear layers (NN), threshold&comparison gates (DT)

Practical challenges of constructing ZKPs for ML



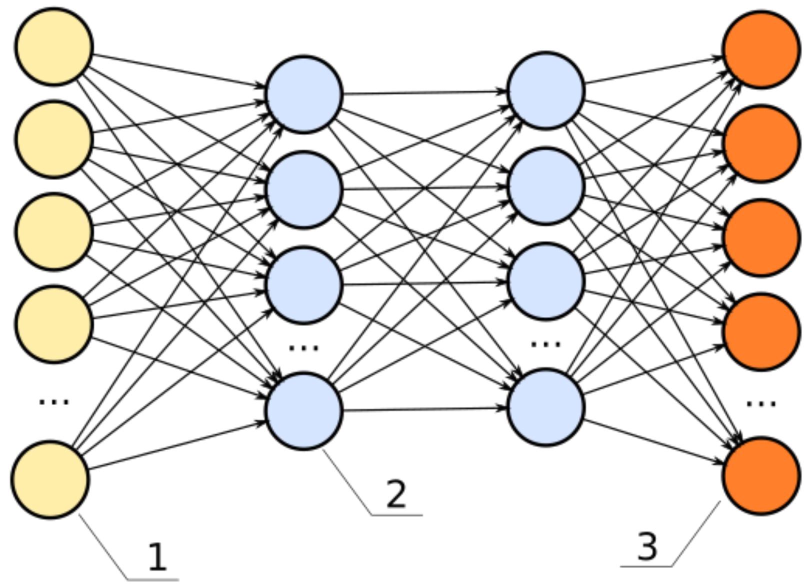
ML inference

algorithm

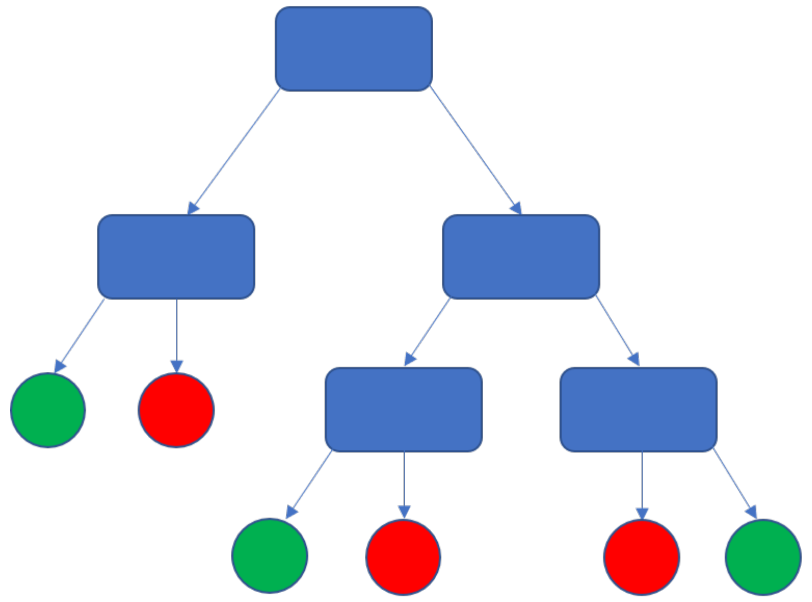
$Y=F(X,W)$

outcome

model



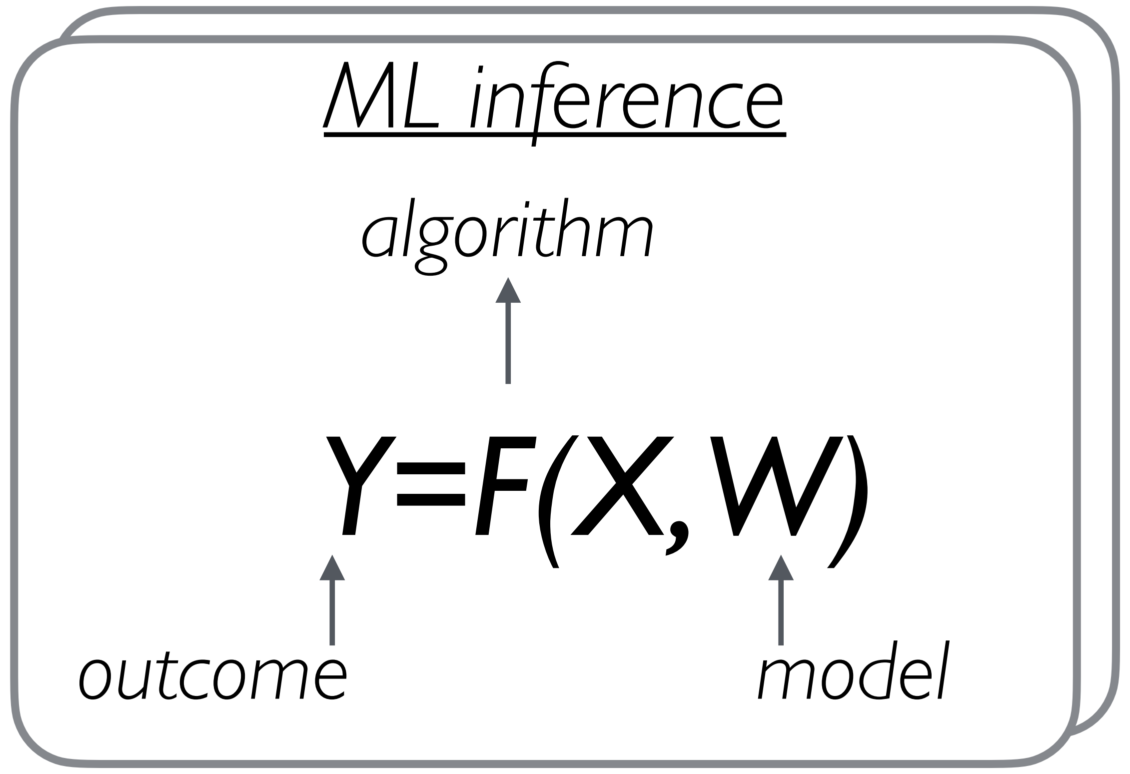
Neural Networks



Decision Trees

- Scale with **large models** and **not-that-ZKP-friendly** computations
  - VGG16 (one of the best computer vision NN) has ~500MB parameters
  - Dense matrix operations, non-linear layers (NN), threshold&comparison gates (DT)
- **Proof verification** — can leverage SNARKs

Practical challenges of constructing ZKPs for ML

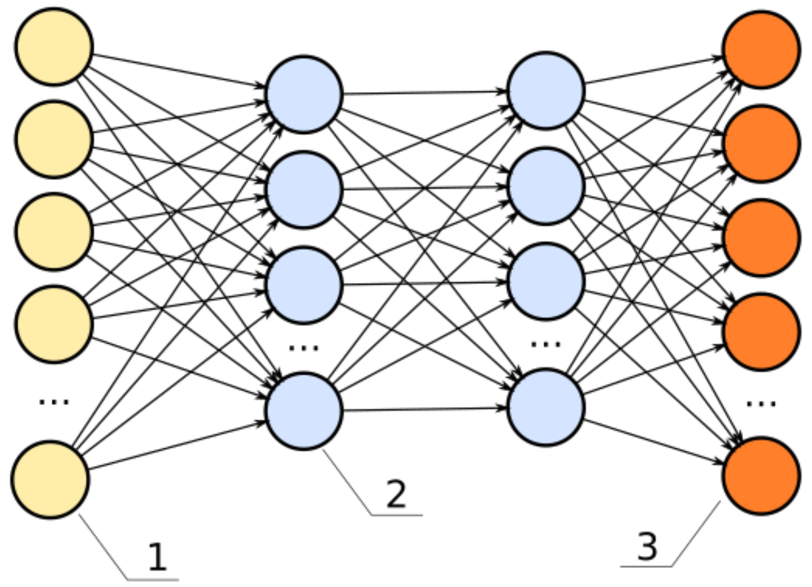


ML inference

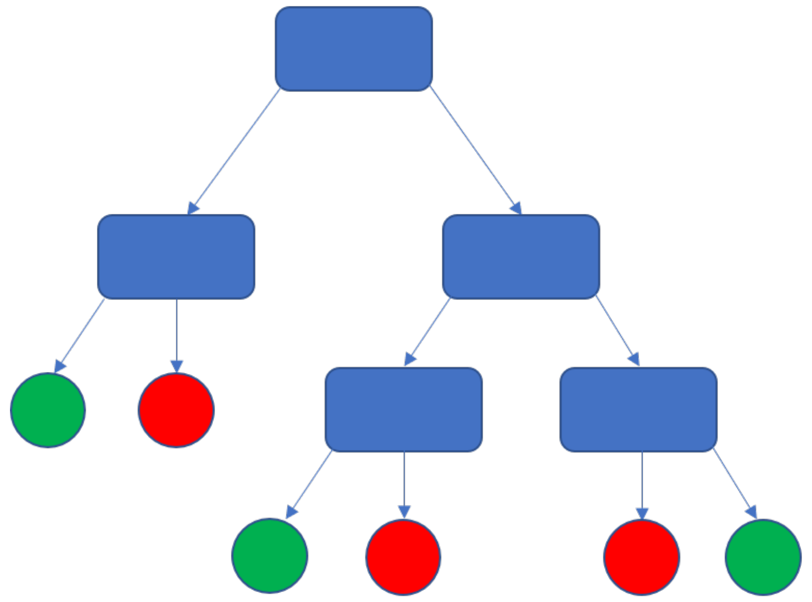
algorithm

$Y=F(X,W)$

outcome *model*



Neural Networks



Decision Trees

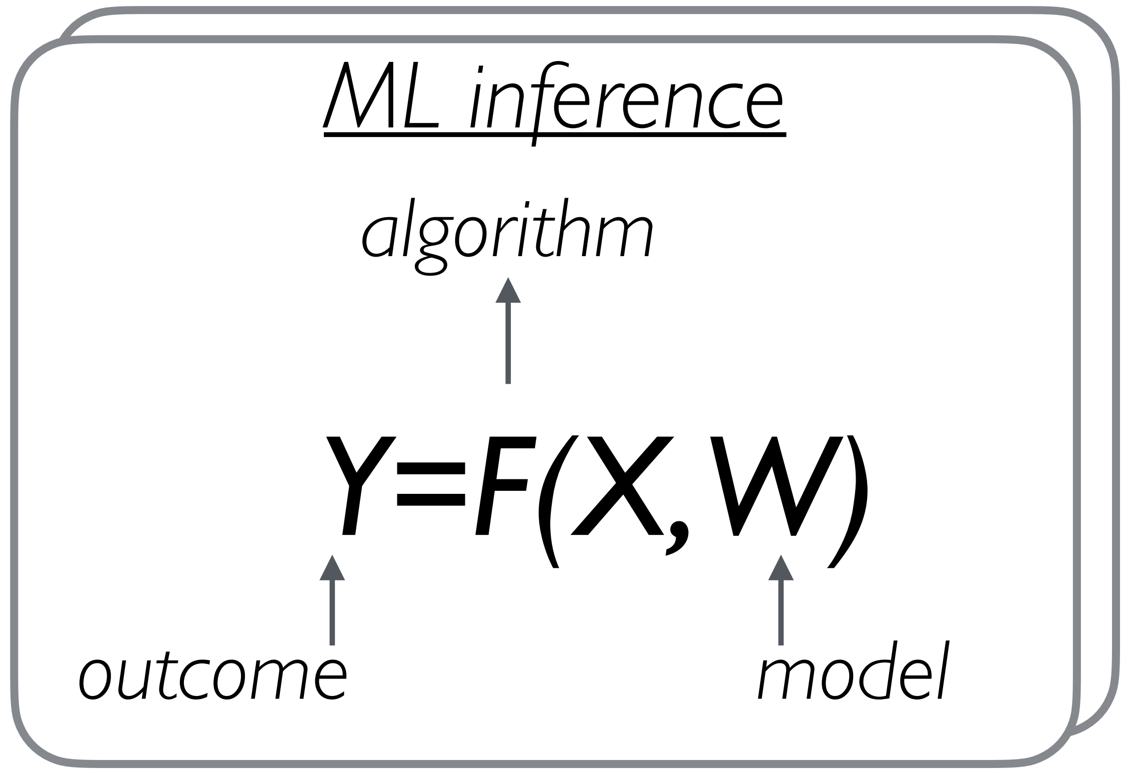
Scale with **large models** and **not-that-ZKP-friendly** computations

- VGG16 (one of the best computer vision NN) has ~500MB parameters
- Dense matrix operations, non-linear layers (NN), threshold&comparison gates (DT)

Proof verification — can leverage SNARKs

Proof generation — the most challenging, we'd like proving time $O(|F|)$ *and concretely close to* $|F|$

Practical challenges of constructing ZKPs for ML



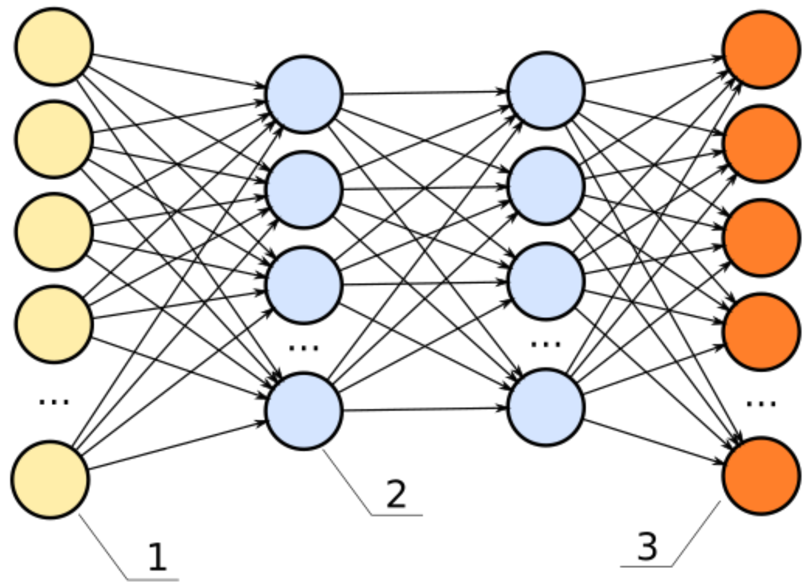
ML inference

algorithm

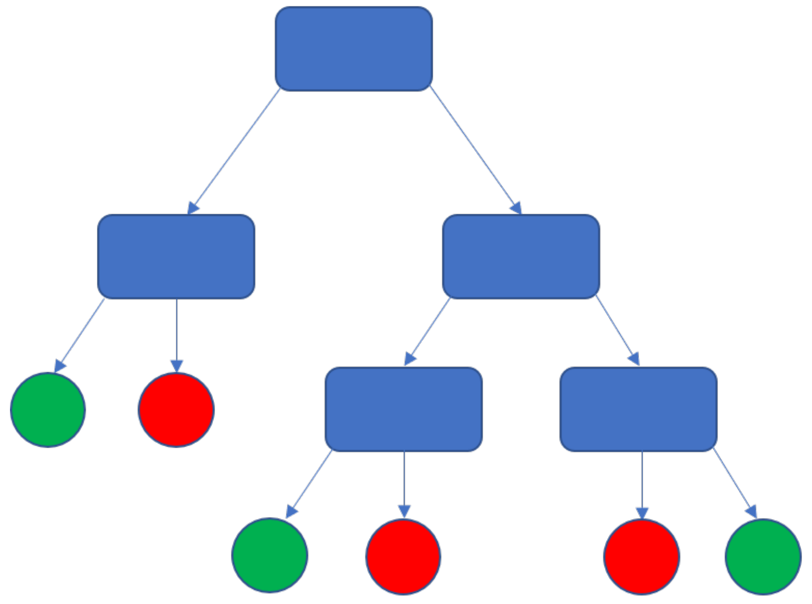
$Y=F(X,W)$

outcome

model



Neural Networks



Decision Trees

Scale with **large models** and **not-that-ZKP-friendly** computations

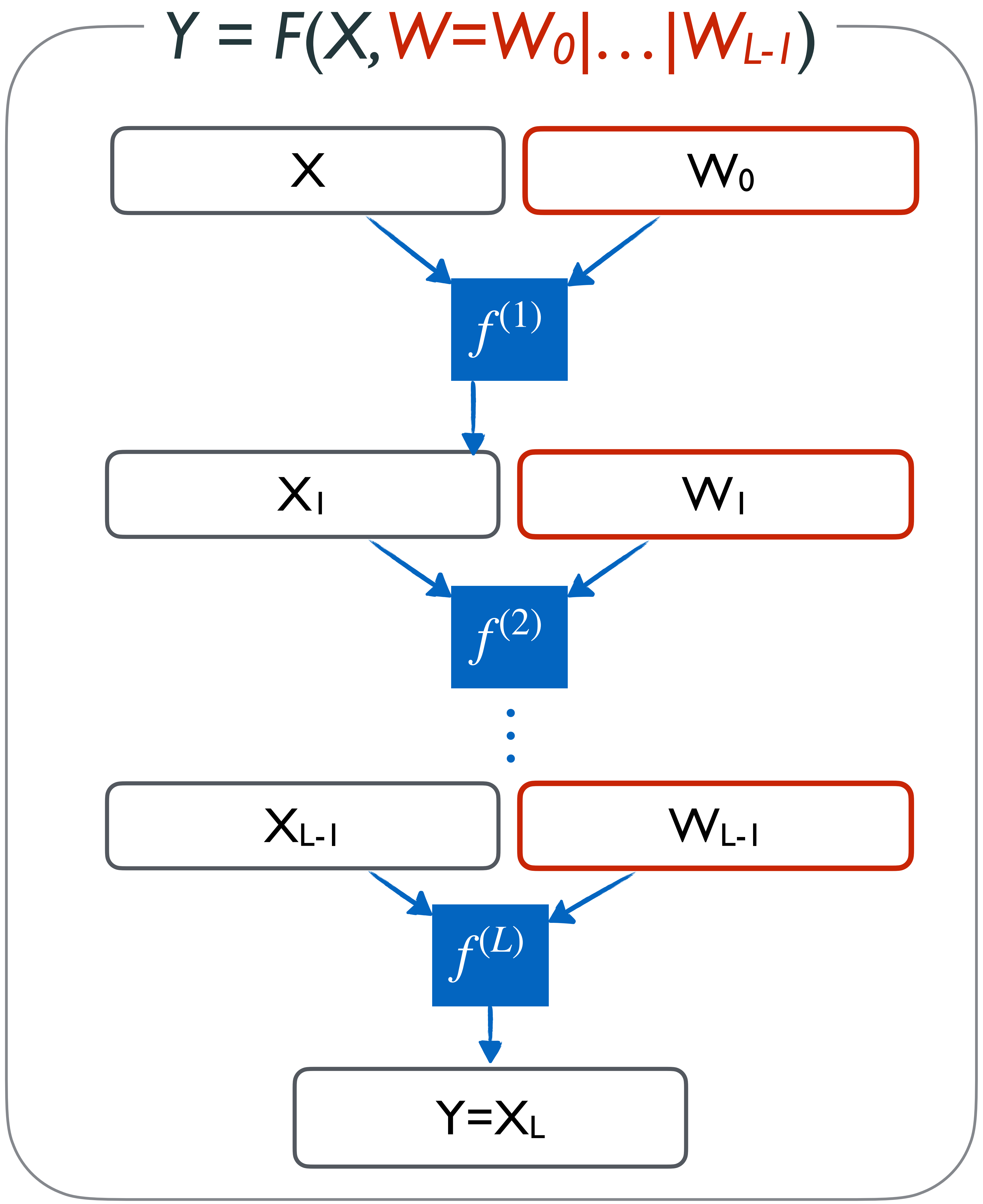
- VGG16 (one of the best computer vision NN) has ~500MB parameters
- Dense matrix operations, non-linear layers (NN), threshold&comparison gates (DT)

Proof verification — can leverage SNARKs

Proof generation — the most challenging, we'd like proving time $O(|F|)$ *and concretely close to* $|F|$

Solution: special-purpose ZKPs!

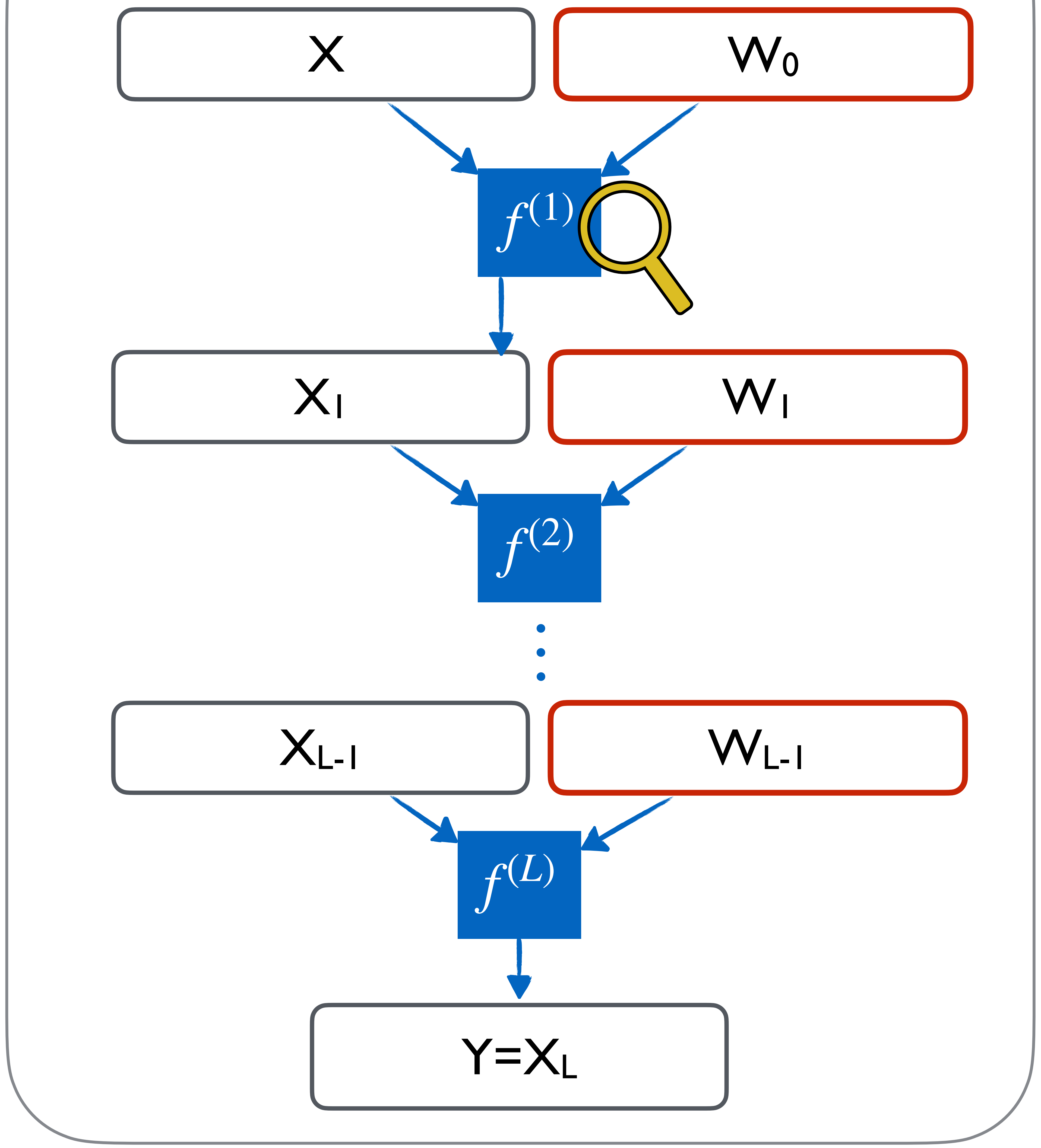
Convolutional Neural Networks



$Y = F(X, W = W_0 | \ldots | W_{L-1})$

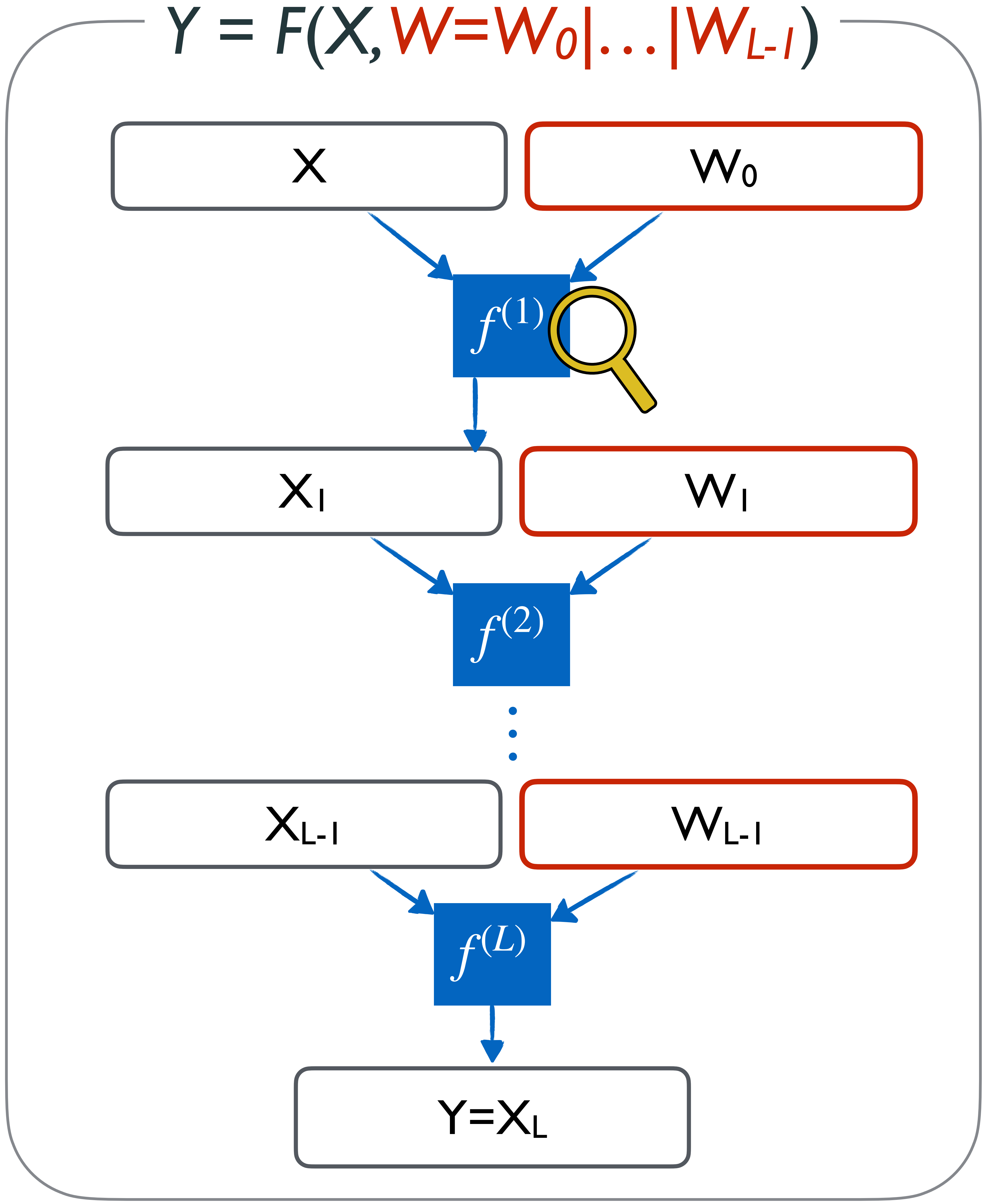
Convolutional Neural Networks

$Y = F(X, W = W_0 | \ldots | W_{L-1})$

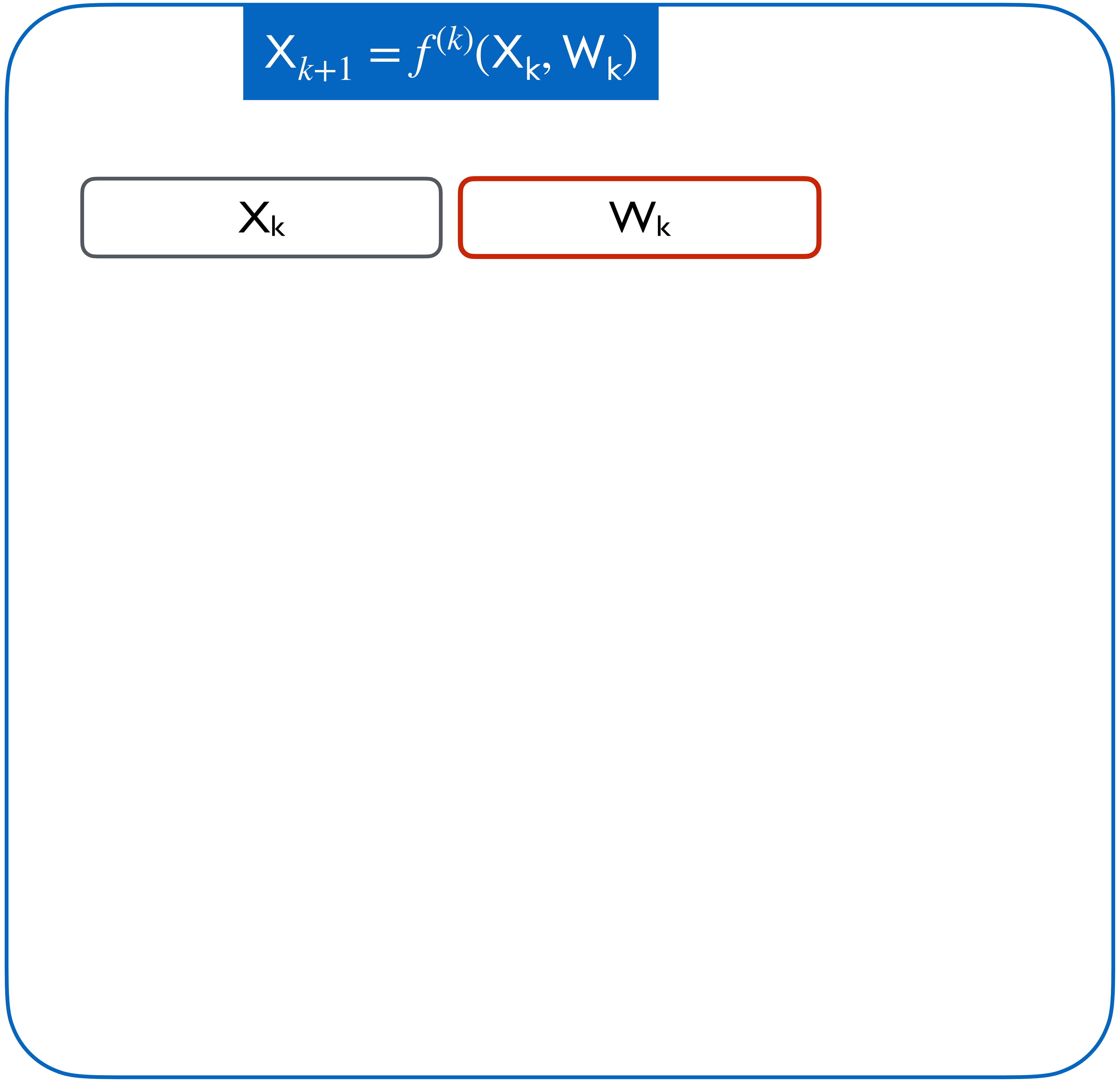


$X_{k+1} = f^{(k)}(X_k, W_k)$

Convolutional Neural Networks

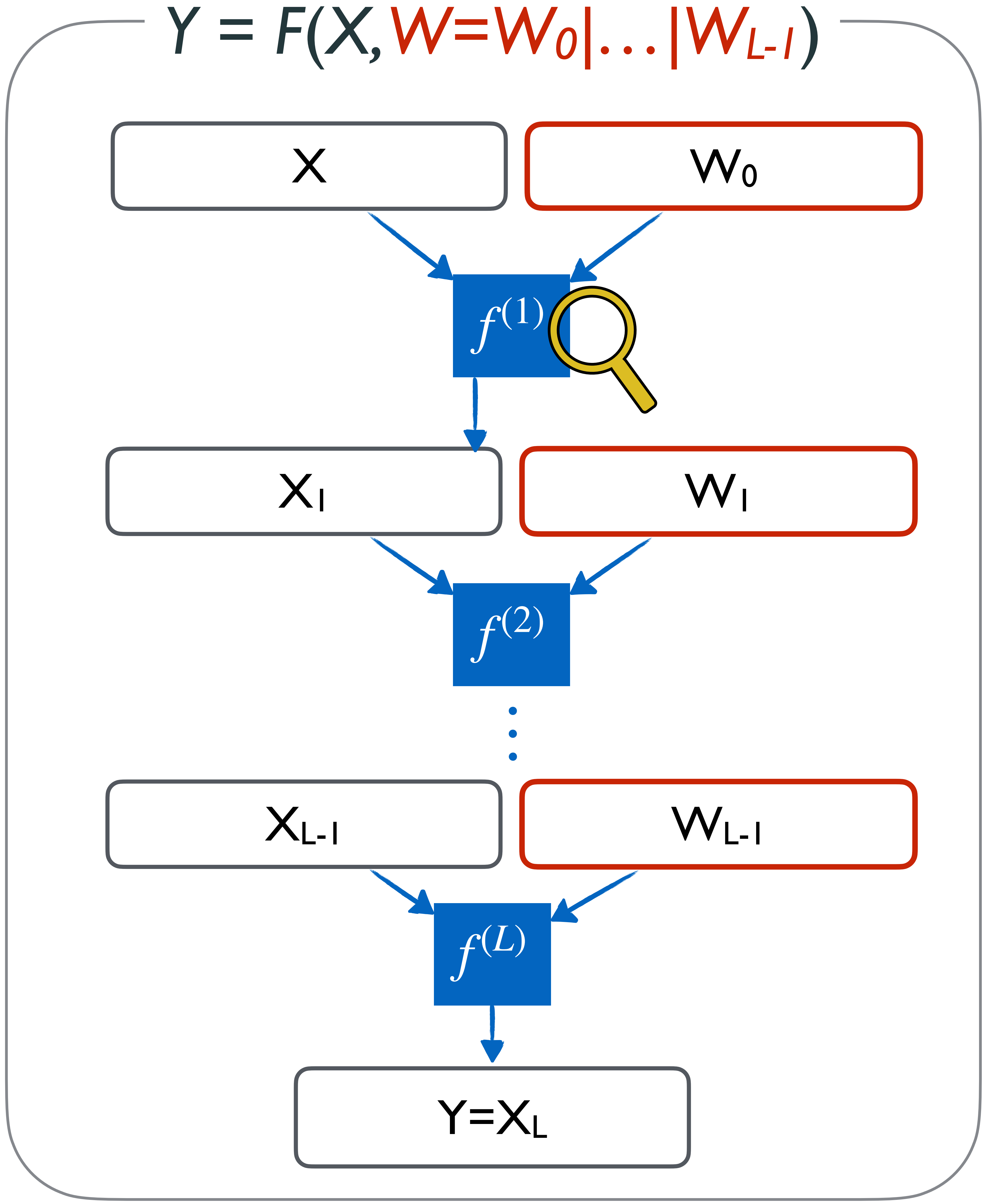


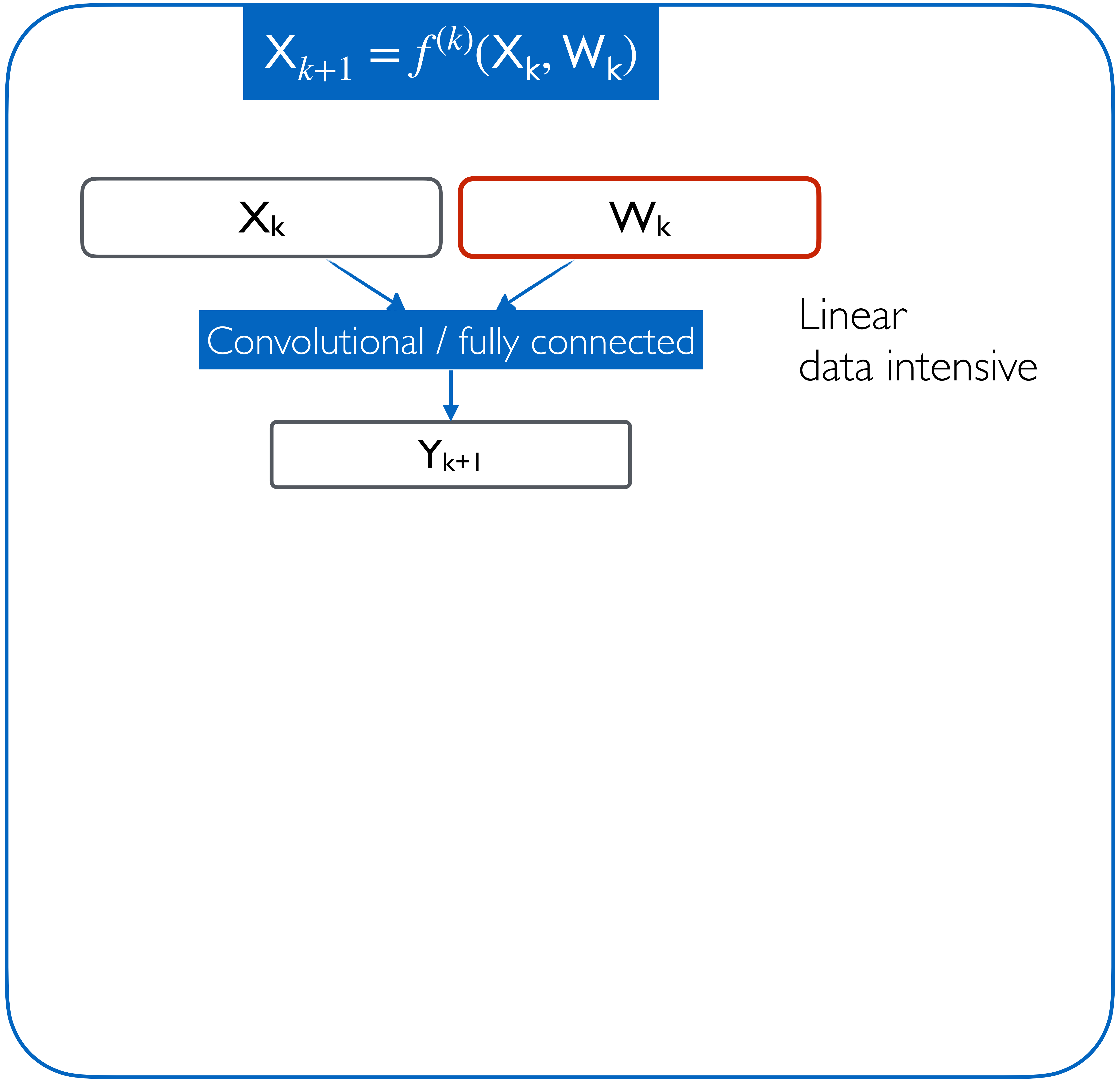
$Y = F(X, W = W_0|\ldots|W_{L-1})$



$X_{k+1} = f^{(k)}(X_k, W_k)$

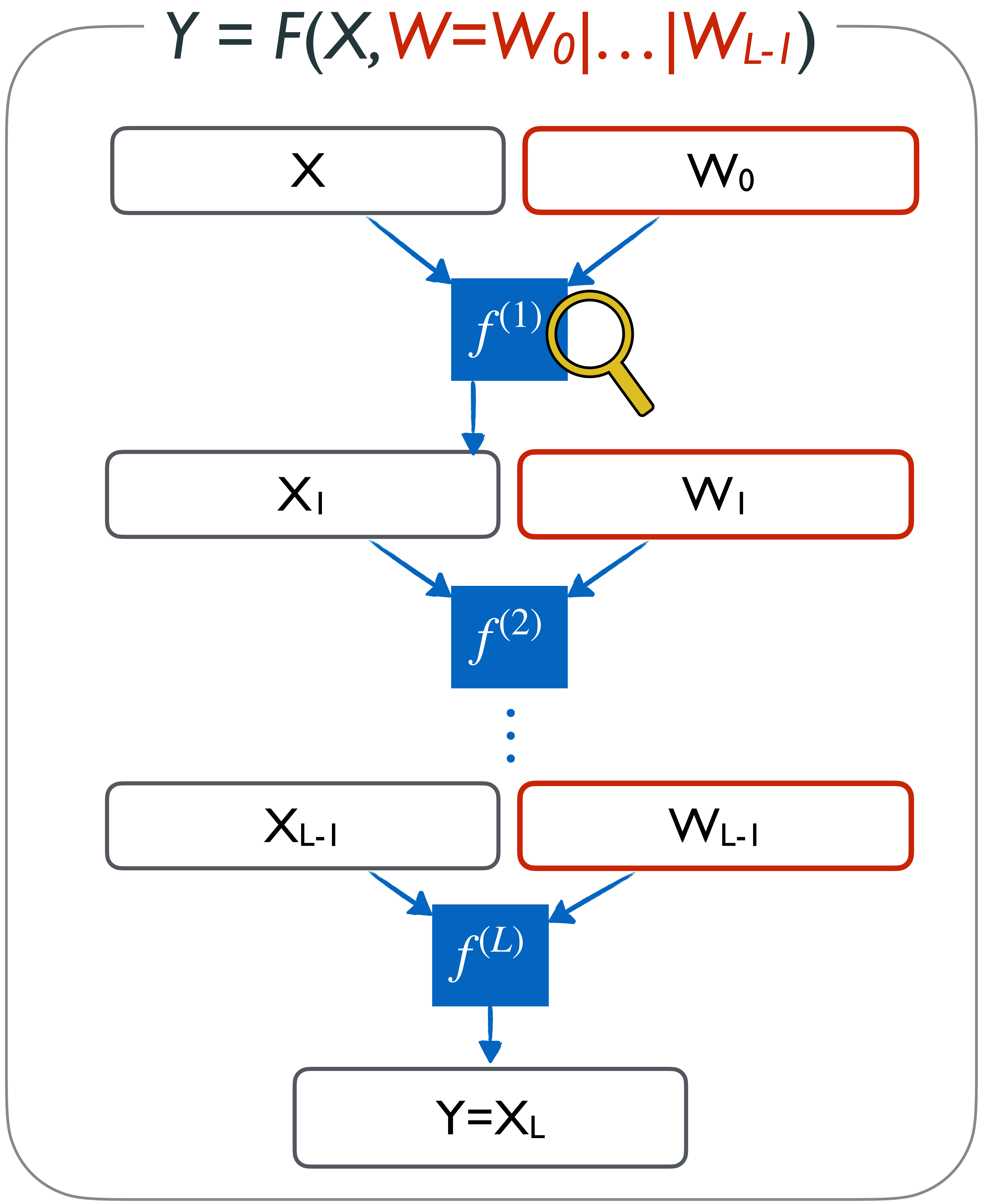
Convolutional Neural Networks





Linear
data intensive

Convolutional Neural Networks



$Y = F(X, W=W_0|\ldots|W_{L-1})$

X

W_0

$f^{(1)}$

X_1

W_1

$f^{(2)}$

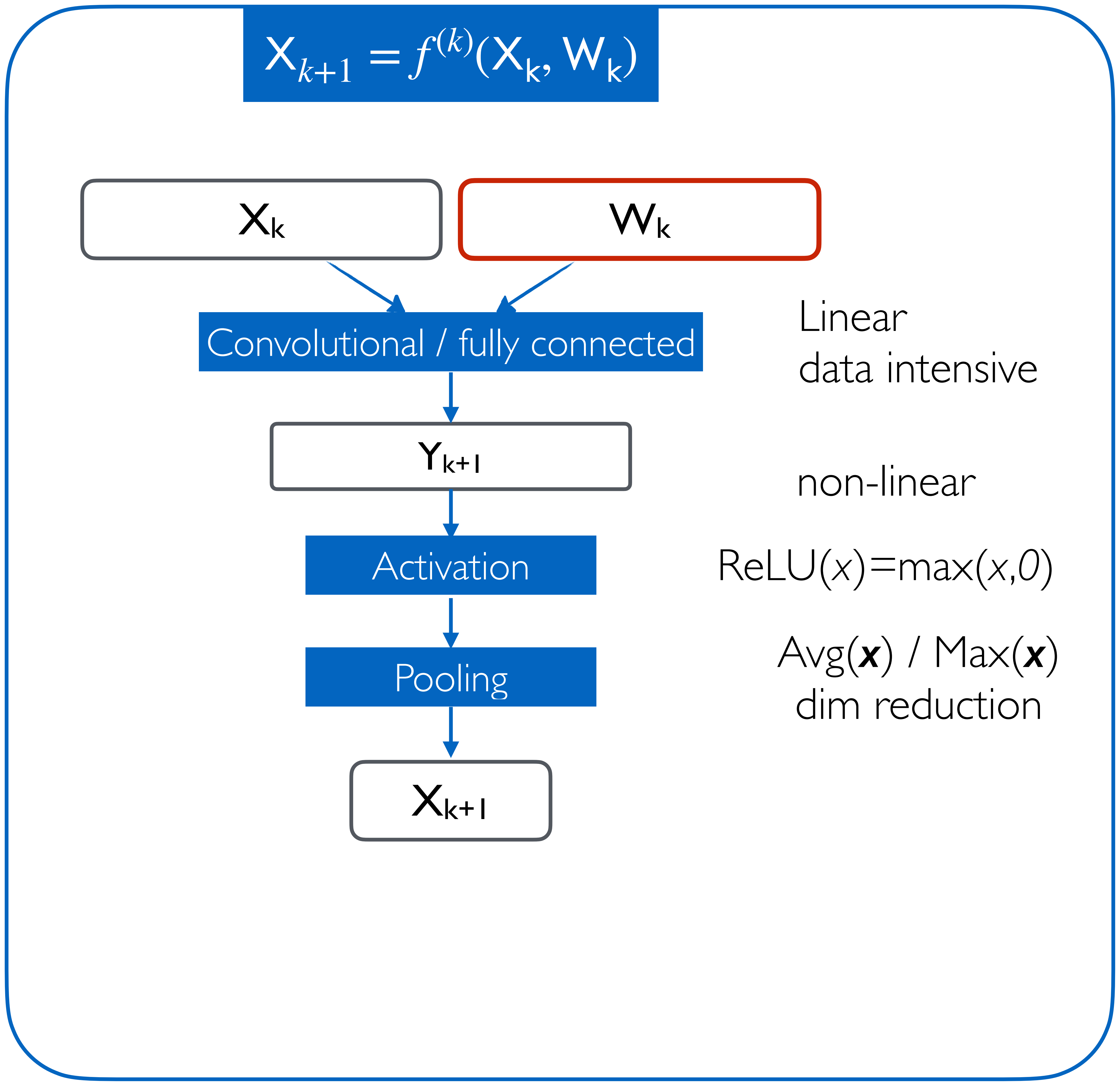
$\vdots$

X_{L-1}

W_{L-1}

$f^{(L)}$

$Y=X_L$



$X_{k+1} = f^{(k)}(X_k, W_k)$

X_k

W_k

Convolutional / fully connected

Linear
data intensive

Y_{k+1}

non-linear

Activation

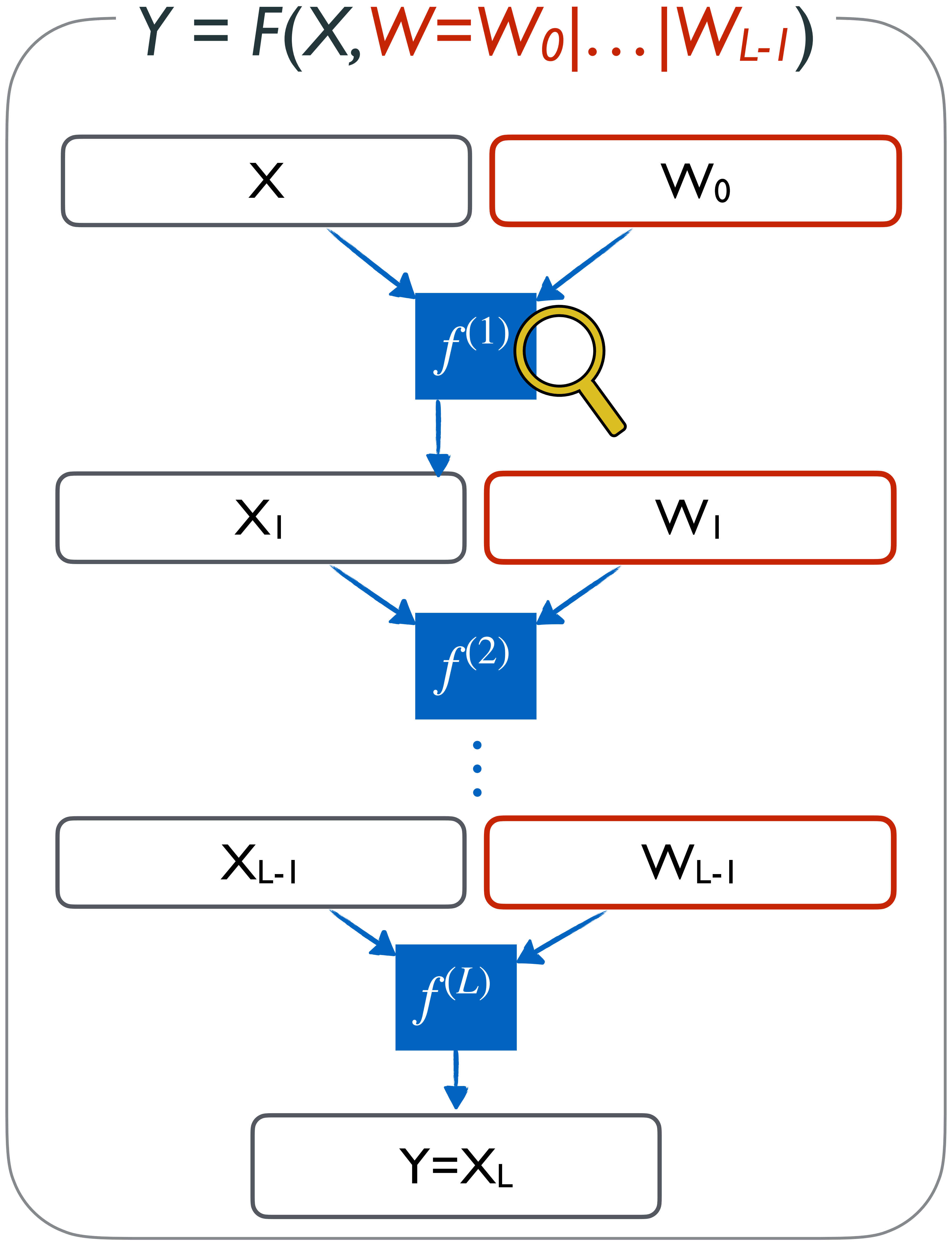
ReLU(x)=max(x,0)

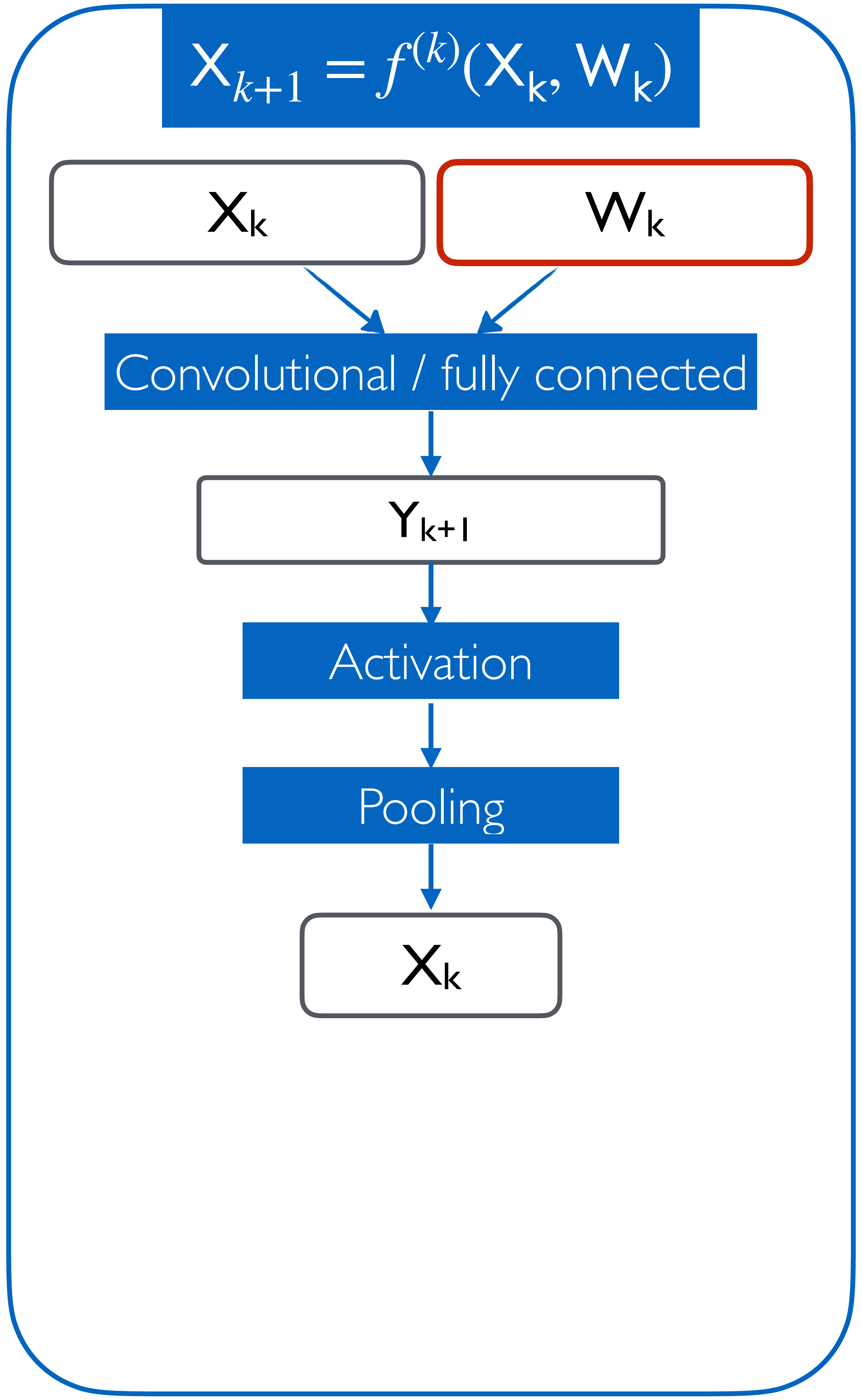
Pooling

Avg($\boldsymbol{x}$) / Max($\boldsymbol{x}$)
dim reduction

X_{k+1}

ZKPs for CNNs



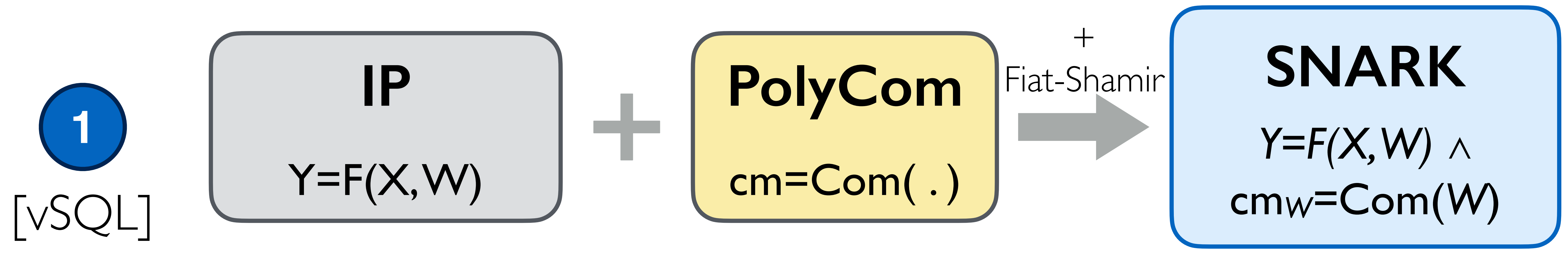


- **Sumcheck-based proofs** [LXZ21, BFGRS23]
- Suitable for layered computation
- Proof generation mostly information-theoretic. Cryptographic work is only $O(|X|+|W|)$

[LXZ21] Liu, Xi, Zhang. *zkCNN: Zero Knowledge Proofs for Convolutional Neural Network Predictions and Accuracy.* CCS 2021

[BFGRS23] Balbás, Fiore, Gonzalez-Vasco, Robissout, Soriente. *Modular Sumcheck Proofs with Applications to Machine Learning and Image Processing.* CCS 2023

Modular approach for CNNs [BFGRS23]



① [vSQL]

IP

Y=F(X,W)

\+

PolyCom

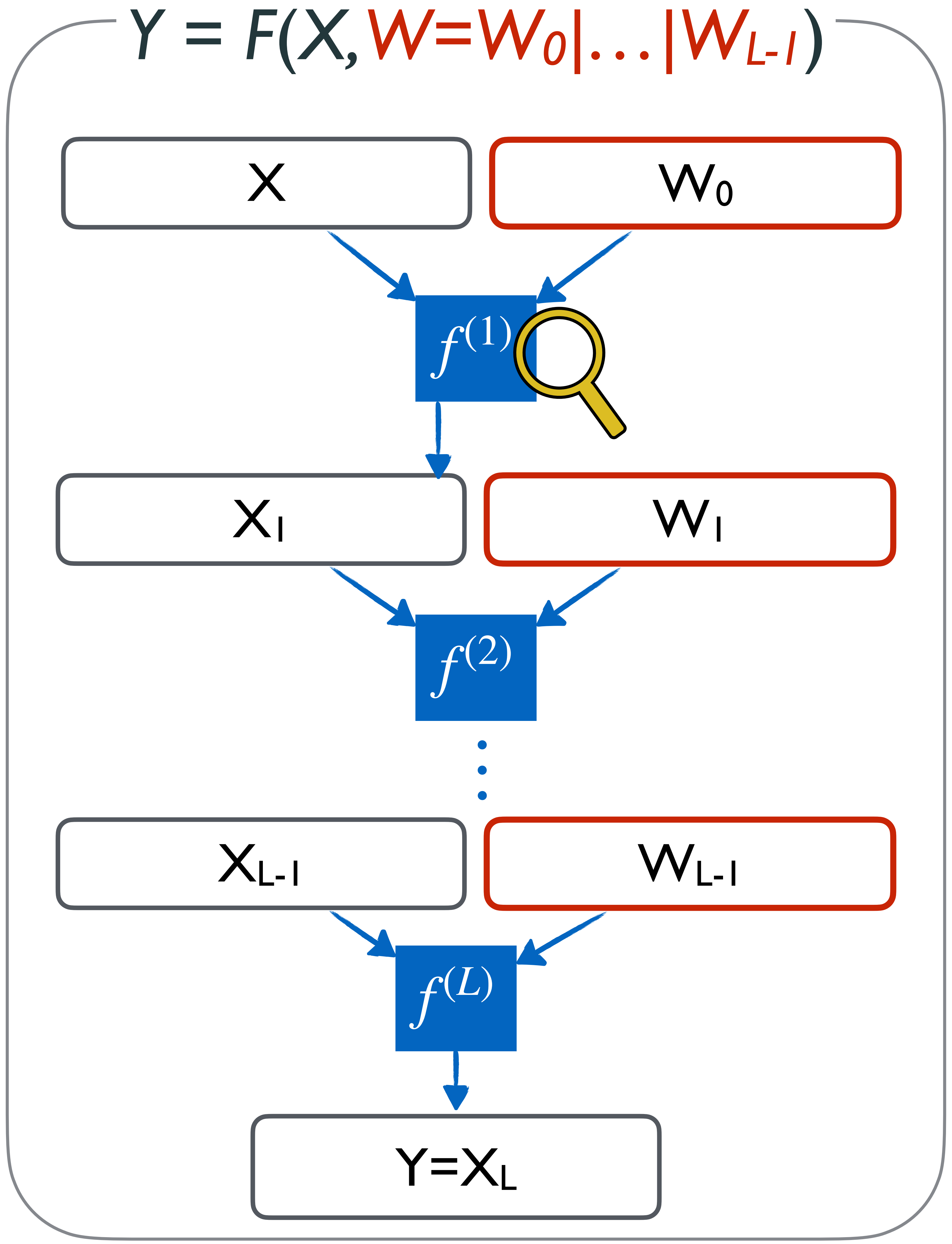
cm=Com(.)

\+ Fiat-Shamir

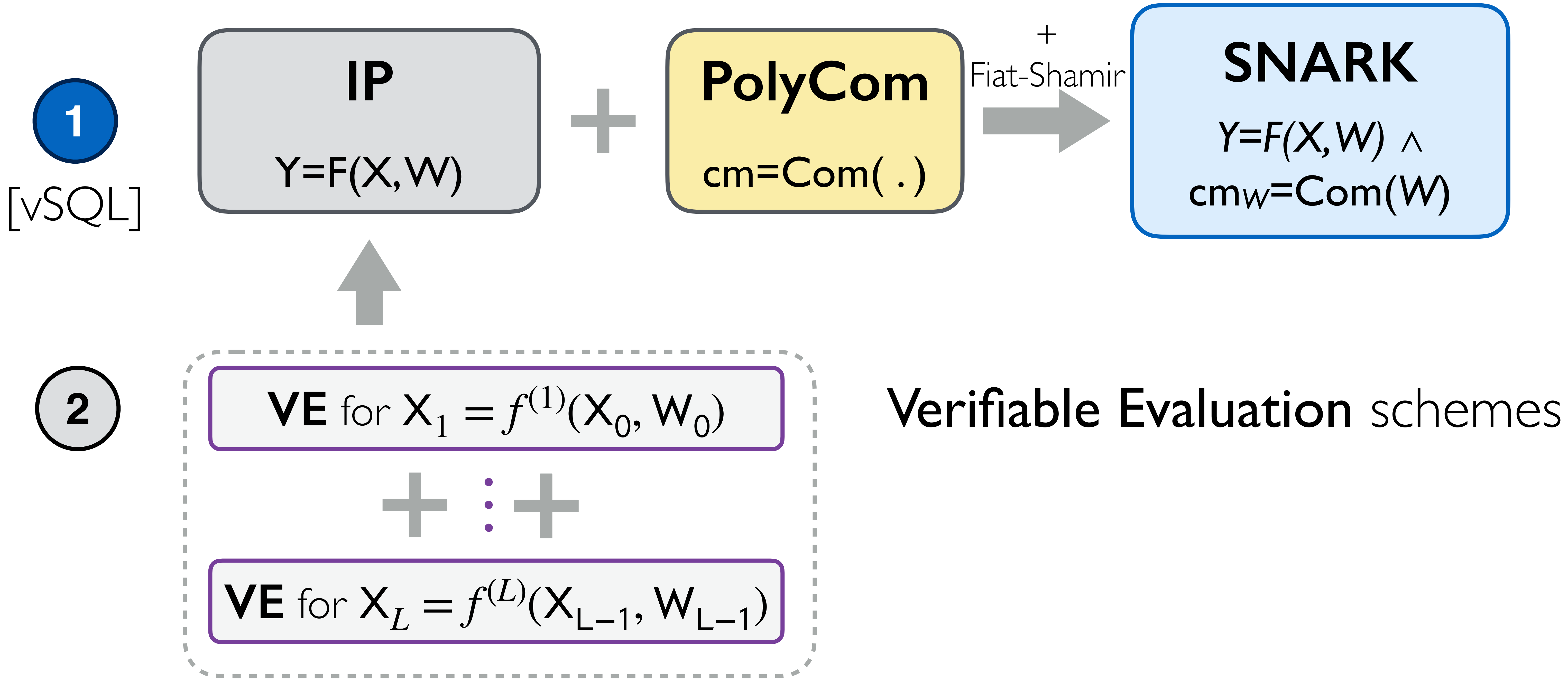
SNARK

$Y=F(X,W) \wedge$

$cm_W=Com(W)$



Modular approach for CNNs [BFGRS23]



1 [vSQL]

IP

$Y=F(X,W)$

\+

PolyCom

cm=Com(.)

\+ Fiat-Shamir

SNARK

$Y=F(X,W) \wedge$

$cm_W=Com(W)$

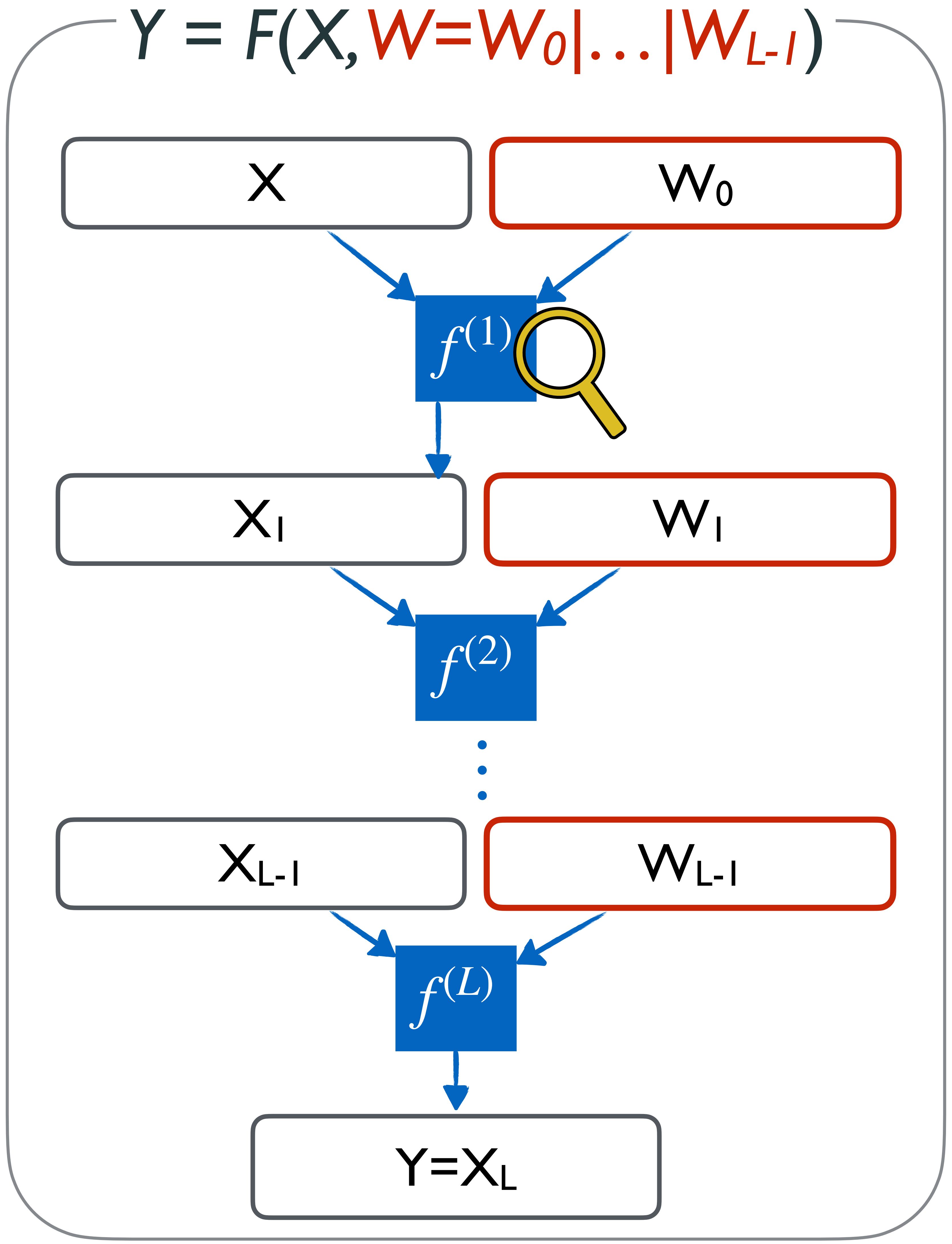
2

VE for $X_1 = f^{(1)}(X_0, W_0)$

\+ ⋮ +

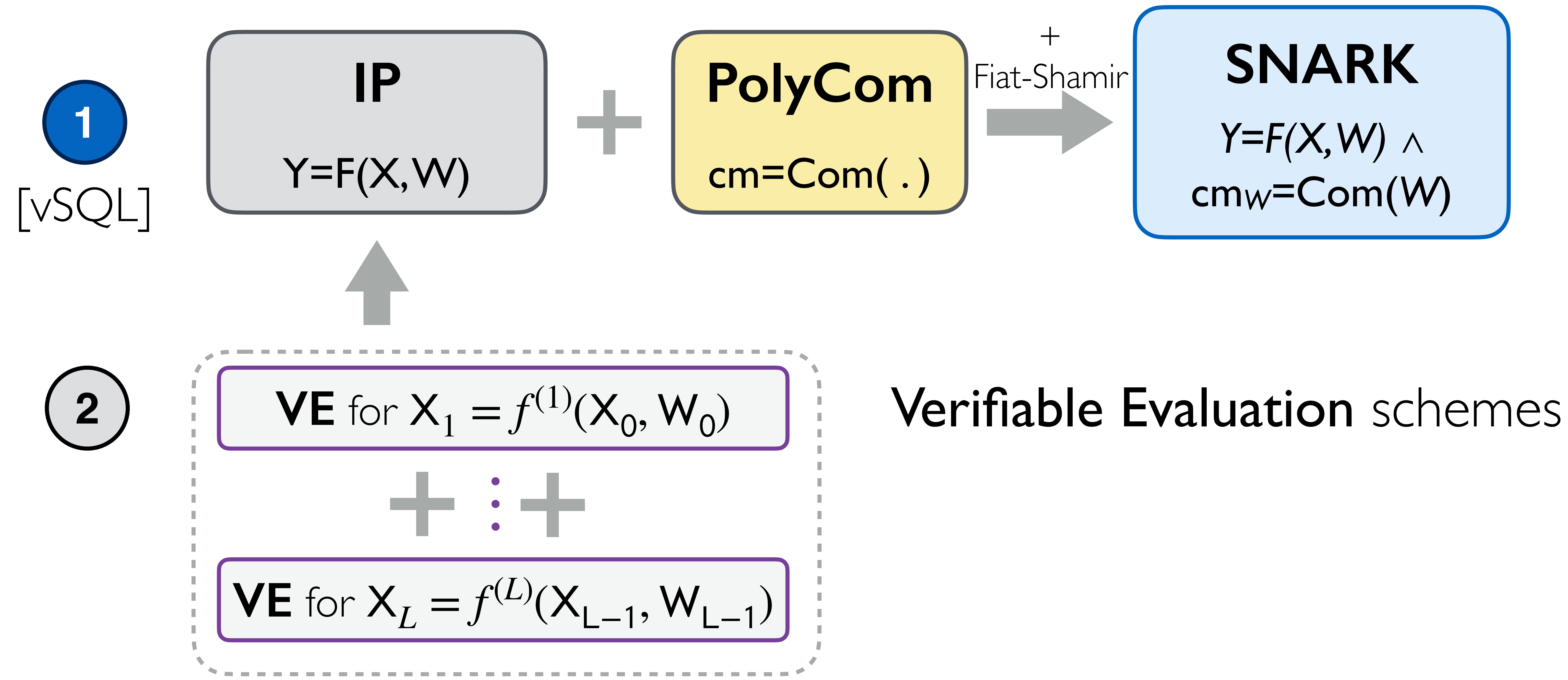
VE for $X_L = f^{(L)}(X_{L-1}, W_{L-1})$

Verifiable Evaluation schemes



$Y = F(X, W=W_0|\ldots|W_{L-1})$

Modular approach for CNNs [BFGRS23]



1

[vSQL]

IP

Y=F(X,W)

+

PolyCom

cm=Com(.)

+ Fiat-Shamir

SNARK

$Y=F(X,W) \wedge$

$cm_W=Com(W)$

2

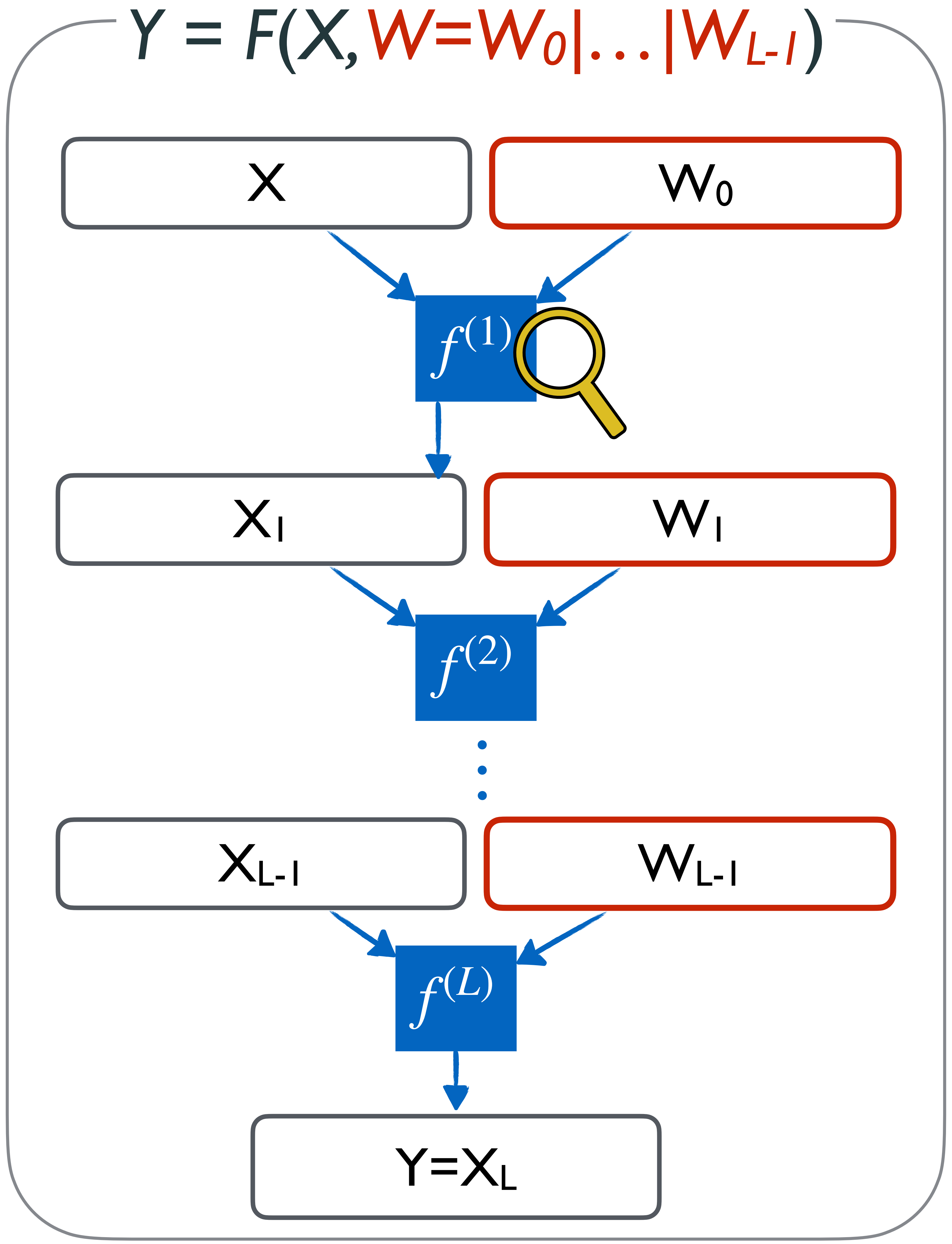
VE for $X_1 = f^{(1)}(X_0, W_0)$

+ ⋮ +

VE for $X_L = f^{(L)}(X_{L-1}, W_{L-1})$

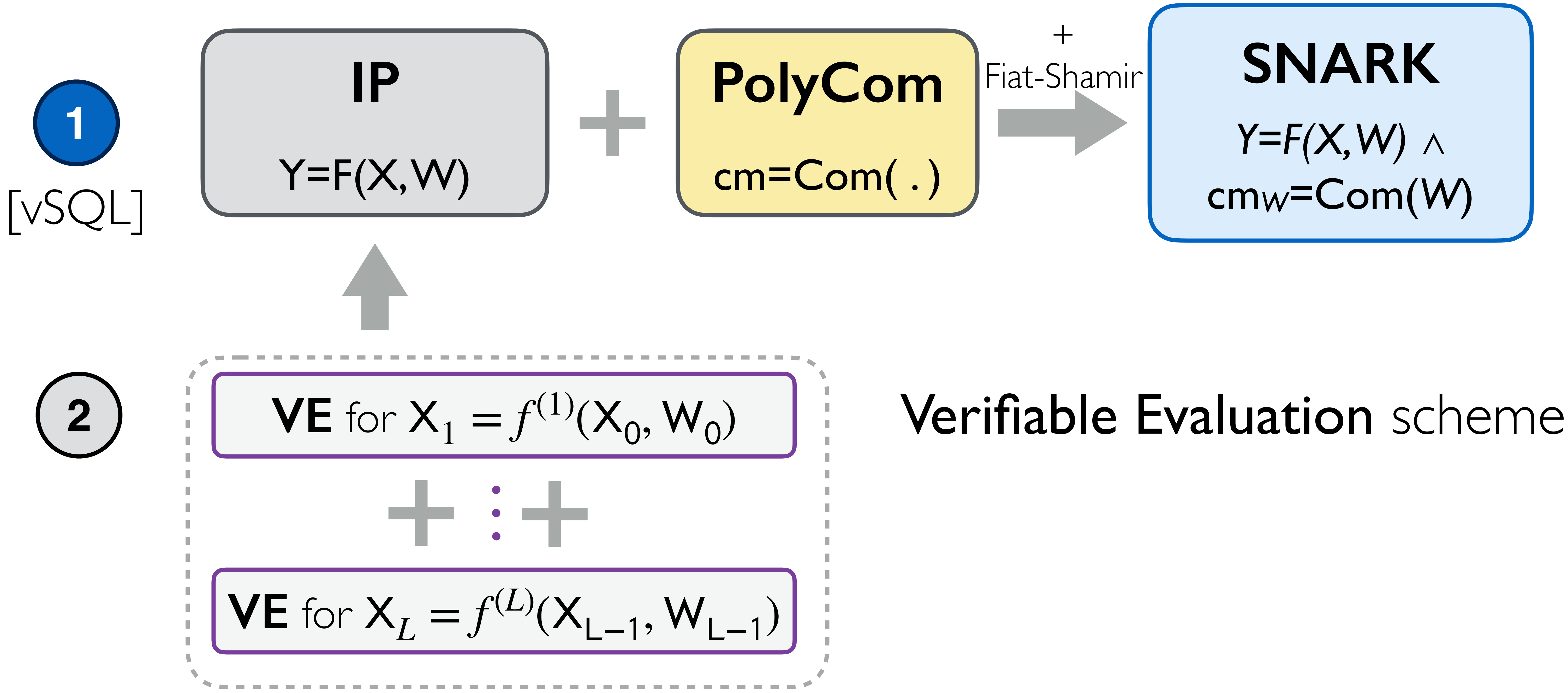
Verifiable Evaluation schemes

3 Sumcheck-based **VEs** for subroutines of CNNs



$Y = F(X, W = W_0 | \ldots | W_{L-1})$

Modular approach for CNNs [BFGRS23]



① [vSQL]

IP

$Y=F(X,W)$

+

PolyCom

cm=Com(.)

+ Fiat-Shamir

SNARK

$Y=F(X,W) \wedge$

$cm_W=Com(W)$

②

VE for $X_1 = f^{(1)}(X_0, W_0)$

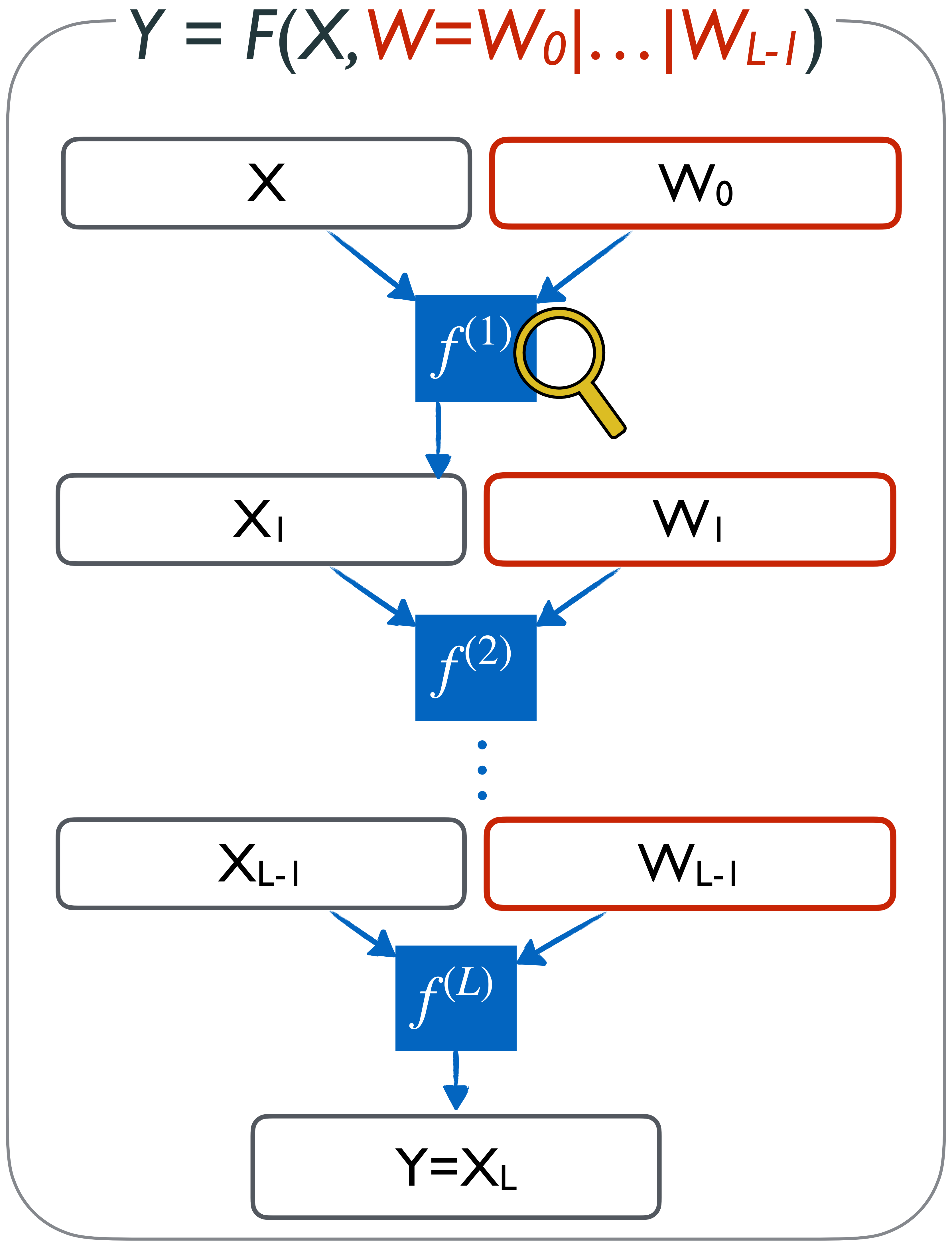
+ ⋮ +

VE for $X_L = f^{(L)}(X_{L-1}, W_{L-1})$

Verifiable Evaluation schemes

③ **Sumcheck-based VEs for subroutines of CNNs**

- Modular design&composition generalization of GKR-style IPs
- Easier to focus on designing efficient specialized VEs



Interactive Proofs and Fingerprints

$\langle \mathscr{P}, \mathscr{V} \rangle(f, x, y) \rightarrow b$: **IP** for language $\mathscr{L}_F = \{(f, x, y) : f(x) = y\}$ *complete* and *sound*

$\mathscr{P}(f, x, y)$

$\mathscr{V}(f, x, y)$

Interactive Proofs and Fingerprints

$\langle \mathcal{P}, \mathcal{V} \rangle(f, x, y) \rightarrow b$: **IP** for language $\mathcal{L}_F = \{(f, x, y) : f(x) = y\}$ *complete* and *sound*

$\mathcal{P}(f, x, y)$ $\mathcal{V}(f, x, y)$

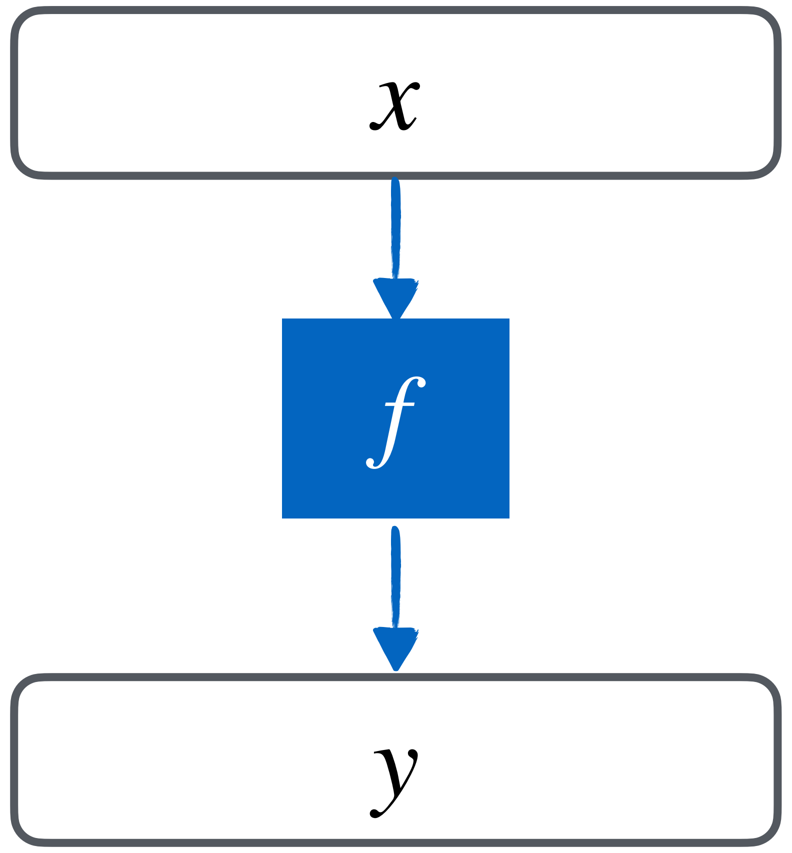
$\vdots$

Fingerprint of x on $r : c_x \leftarrow \mathsf{H}(x, r)$

- Compressing: $|c_x| \ll |x|$
- Statistically binding: $\forall x \neq x^*, \Pr_r[\mathsf{H}(x, r) = \mathsf{H}(x^*, r)] = \mathsf{negl}(\lambda)$

Example: for $\vec{x} \in \mathbb{F}^n, \vec{r} \in \mathbb{F}^{\log n}$, **H** is the MLE evaluation $\mathsf{H}(\vec{x}, \vec{r}) = \tilde{x}(\vec{r})$

Structure of common IPs

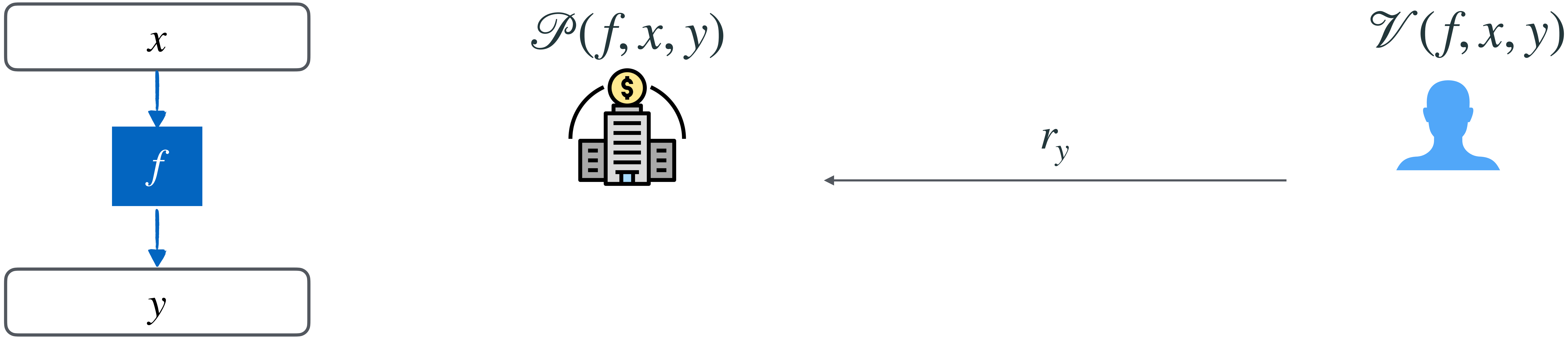


$\mathcal{P}(f, x, y)$

$\mathcal{V}(f, x, y)$

- Public coin verifier

Structure of common IPs



x

f

y

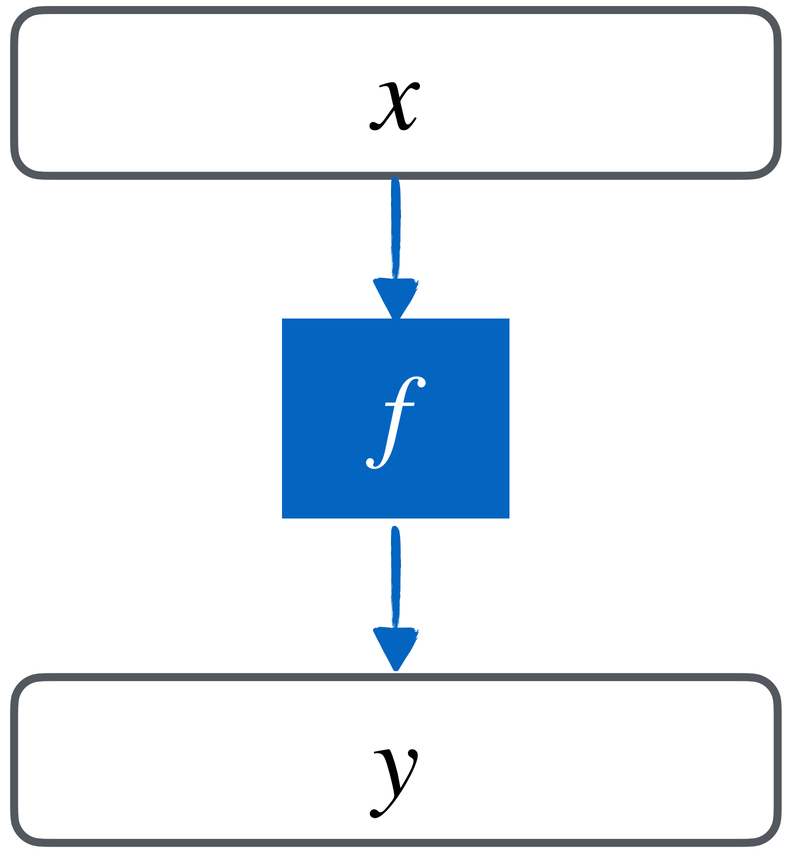
$\mathscr{P}(f, x, y)$

r_y

$\mathscr{V}(f, x, y)$

- Public coin verifier

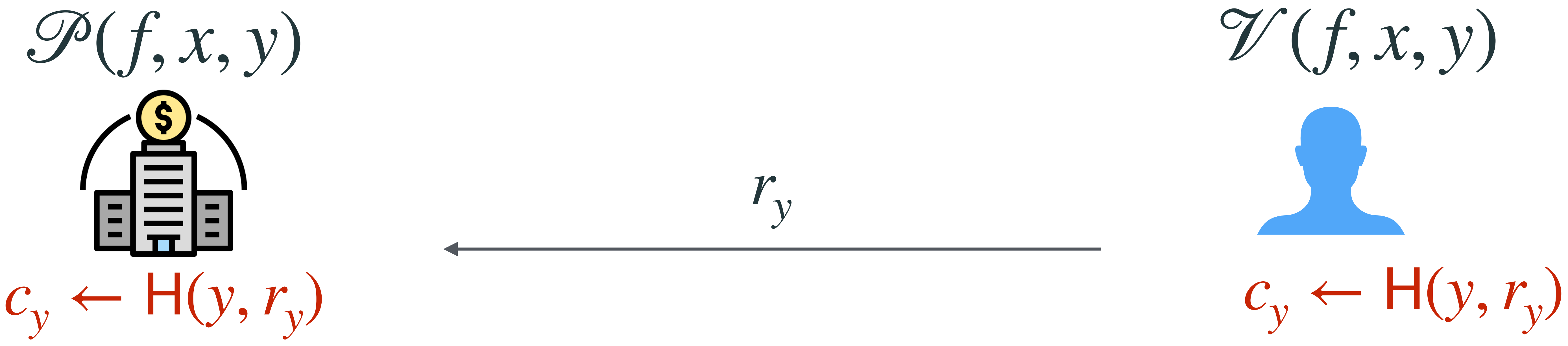
Structure of common IPs



x

f

y



$\mathcal{P}(f, x, y)$

$\mathcal{V}(f, x, y)$

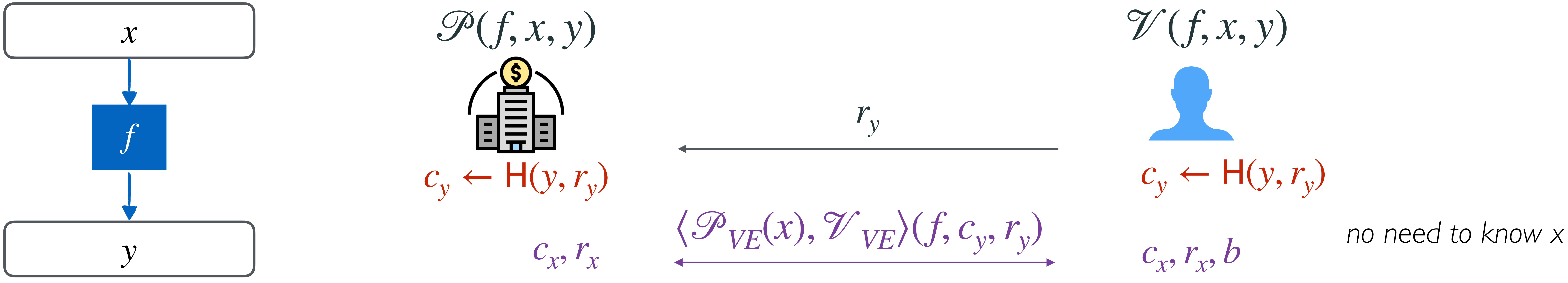
r_y

$c_y \leftarrow \mathsf{H}(y, r_y)$

$c_y \leftarrow \mathsf{H}(y, r_y)$

- Public coin verifier
- Output fingerprint

Structure of common IPs



$\mathcal{P}(f, x, y)$

$\mathcal{V}(f, x, y)$

r_y

$c_y \leftarrow \mathrm{H}(y, r_y)$

$c_y \leftarrow \mathrm{H}(y, r_y)$

$\langle \mathcal{P}_{VE}(x), \mathcal{V}_{VE} \rangle (f, c_y, r_y)$

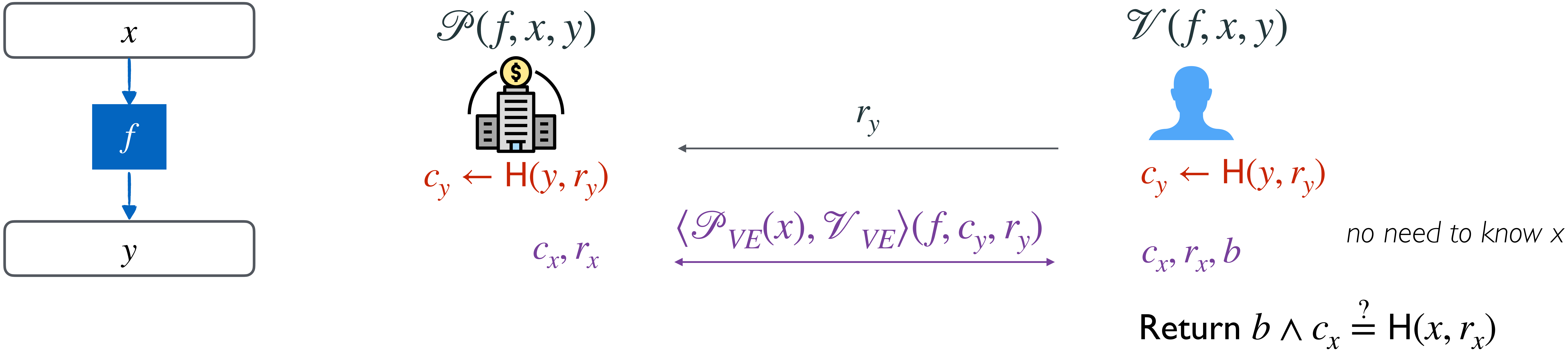
c_x, r_x

c_x, r_x, b

no need to know x

- Public coin verifier
- Output fingerprint
- **Subroutine**: verifiable evaluation (VE) schemes on fingerprinted data ($\mathcal{V}_{VE}$ runs w/o x, y)

Structure of common IPs



x

f

y

$\mathscr{P}(f, x, y)$

$\mathscr{V}(f, x, y)$

r_y

$c_y \leftarrow \mathsf{H}(y, r_y)$

$c_y \leftarrow \mathsf{H}(y, r_y)$

$\langle \mathscr{P}_{VE}(x), \mathscr{V}_{VE} \rangle (f, c_y, r_y)$

c_x, r_x

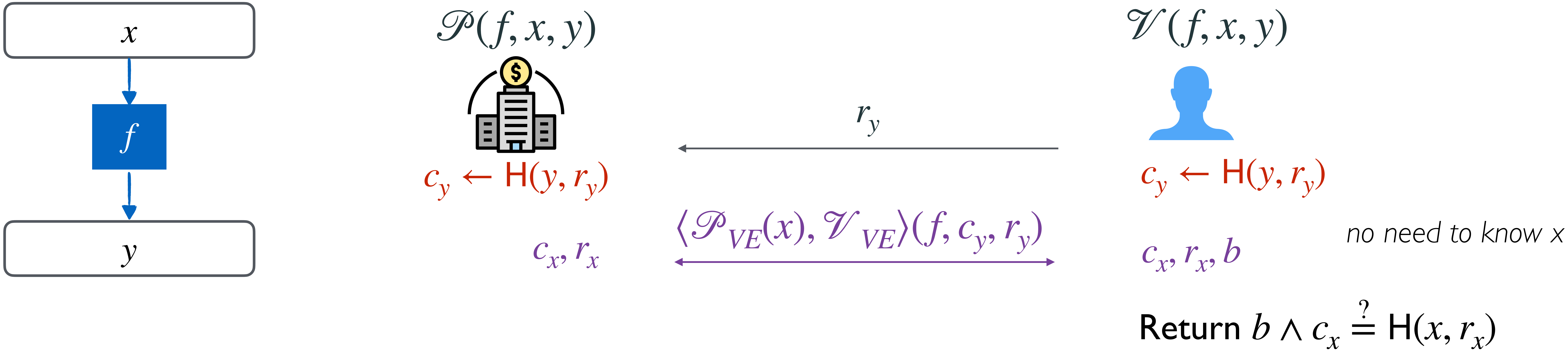
c_x, r_x, b

no need to know x

Return $b \wedge c_x \overset{?}{=} \mathsf{H}(x, r_x)$

- Public coin verifier
- Output fingerprint
- **Subroutine**: verifiable evaluation (VE) schemes on fingerprinted data ($\mathscr{V}_{VE}$ runs w/o x, y)
- Input fingerprint

Structure of common IPs



x

f

y

$\mathcal{P}(f, x, y)$

$c_y \leftarrow \mathrm{H}(y, r_y)$

c_x, r_x

r_y

$\langle \mathcal{P}_{VE}(x), \mathcal{V}_{VE} \rangle (f, c_y, r_y)$

$\mathcal{V}(f, x, y)$

$c_y \leftarrow \mathrm{H}(y, r_y)$

c_x, r_x, b

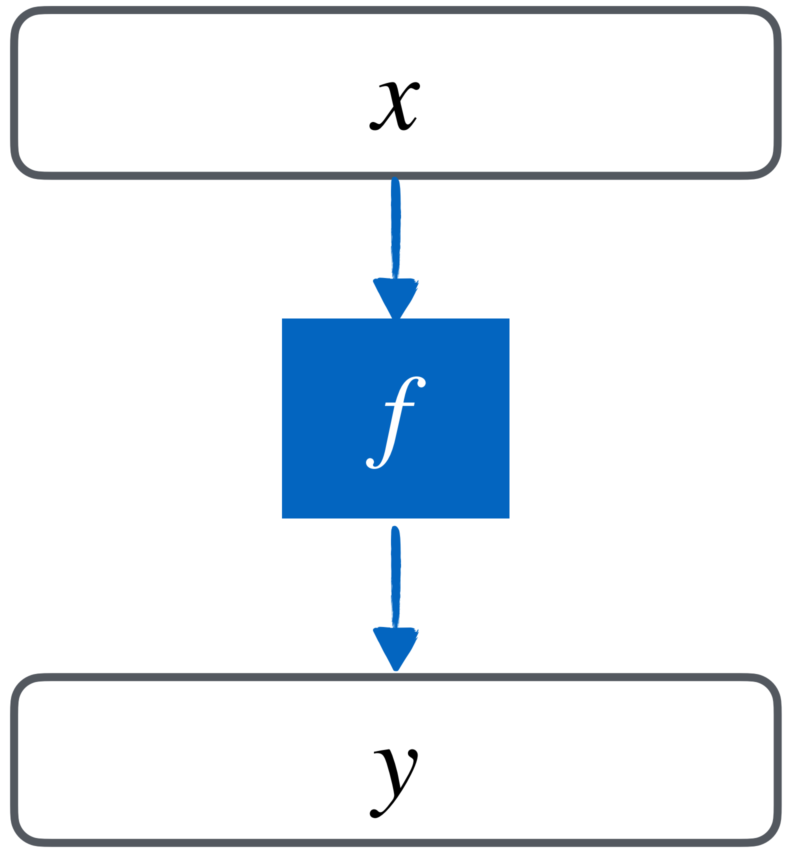
no need to know x

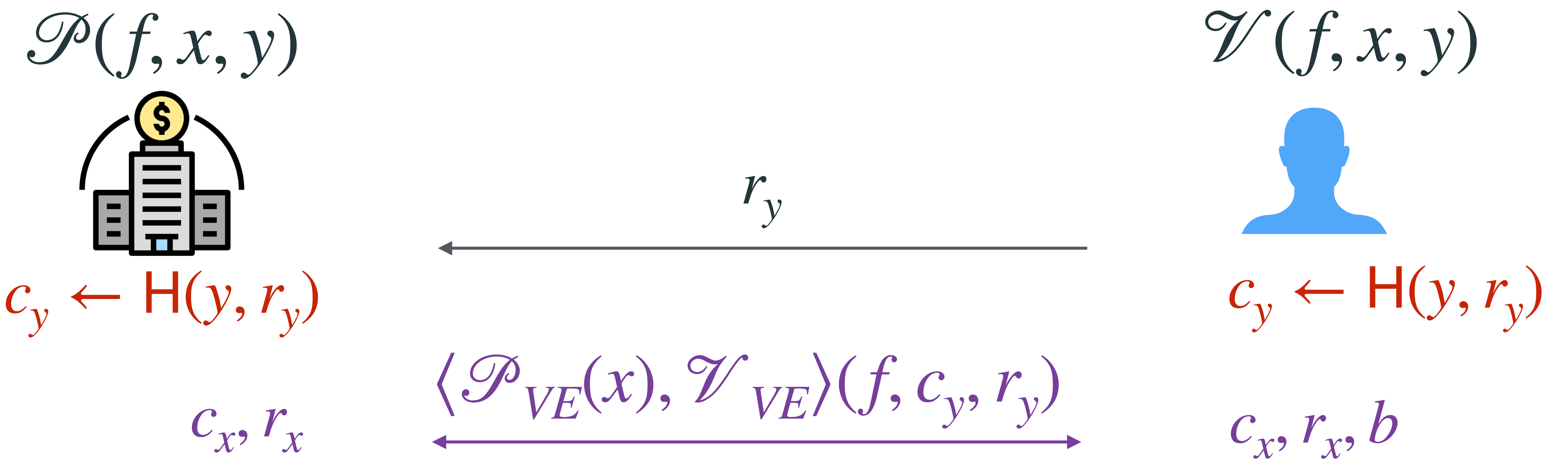
Return $b \wedge c_x \overset{?}{=} \mathrm{H}(x, r_x)$

- Public coin verifier
- Output fingerprint
- **Subroutine**: **verifiable evaluation (VE)** schemes on **fingerprinted** data ($\mathcal{V}_{VE}$ runs w/o x, y)
- Input fingerprint

Examples of VE-based IPs: sumcheck protocol, GKR, ….

VE Soundness





$\mathcal{P}(f, x, y)$

$\mathcal{V}(f, x, y)$

$\xleftarrow{\quad r_y \quad}$

$c_y \leftarrow \mathsf{H}(y, r_y)$

$c_y \leftarrow \mathsf{H}(y, r_y)$

$\langle \mathcal{P}_{VE}(x), \mathcal{V}_{VE}\rangle(f, c_y, r_y)$

c_x, r_x

c_x, r_x, b

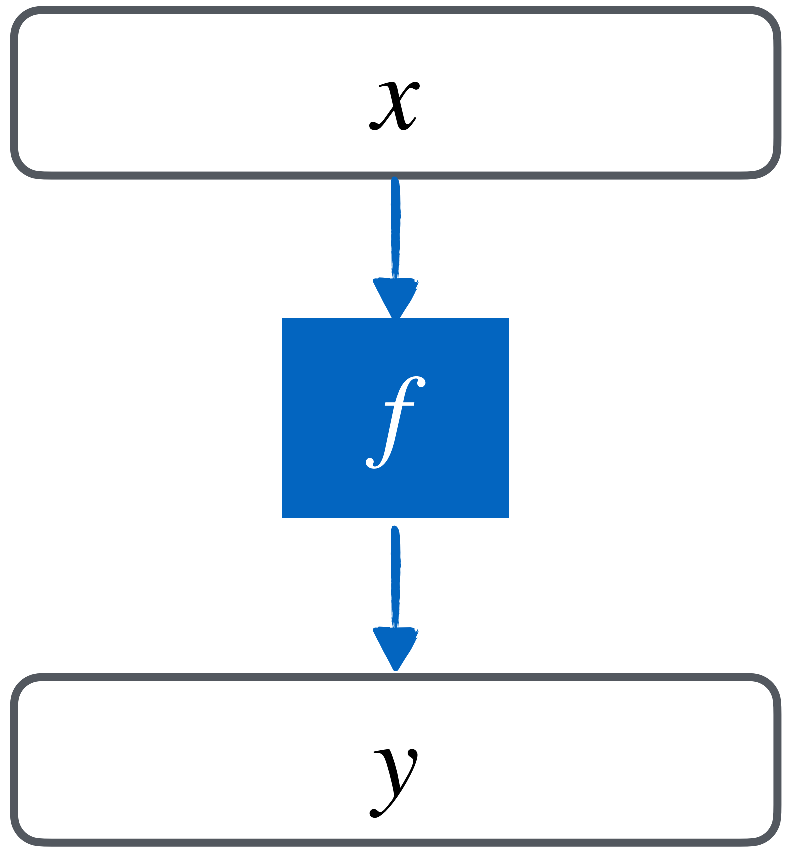
Return $b \wedge c_x \overset{?}{=} \mathsf{H}(x, r_x)$

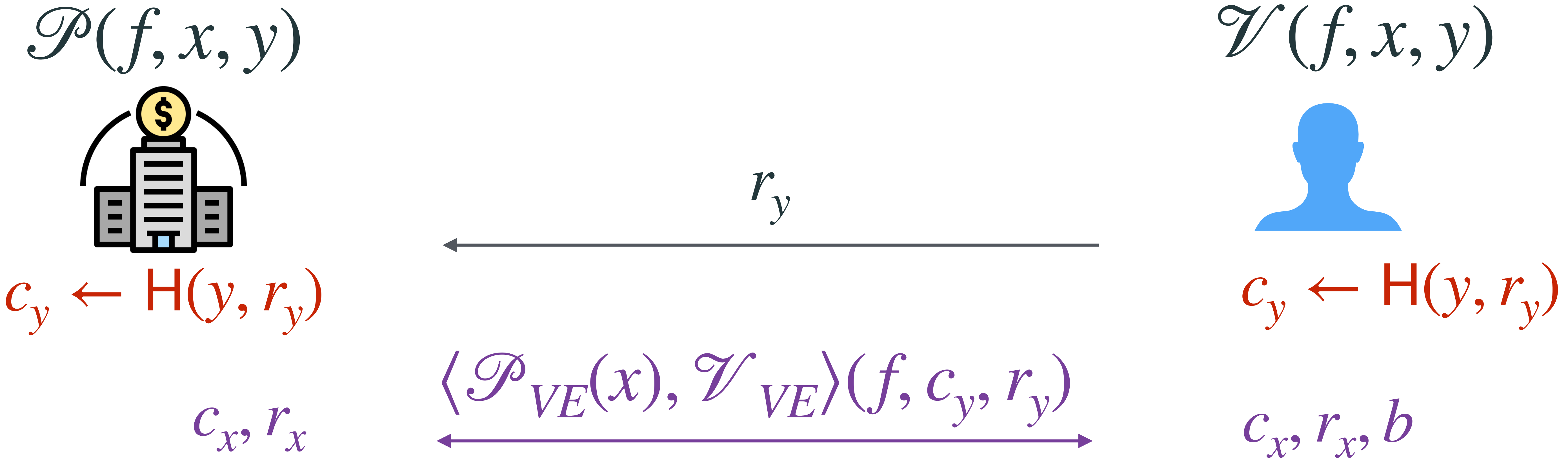
Soundness of VE subroutine: for random r_x, r_y and any unbounded $\mathcal{A}$

Correct input fingerprint
⇒ c_y^* fingerprints correct output

$$\Pr\left[\begin{matrix} c_y^* \neq \mathsf{H}(f(x), r_y) \\ \wedge\, b \end{matrix} \;\middle|\; \begin{matrix} (f, c_y^*, x) \leftarrow \mathcal{A}(r_y) \\ (c_x^*; r_x; b) \leftarrow \left\langle \mathcal{A}(x), \mathcal{V}_{VE}(r_x) \right\rangle (f, c_y^*, r_y) \\ c_x^* = \mathsf{H}(x, r_x) \end{matrix}\right] = \text{negl}$$

VE Soundness





$c_y \leftarrow \mathsf{H}(y, r_y)$

$c_y \leftarrow \mathsf{H}(y, r_y)$

$\langle \mathcal{P}_{VE}(x), \mathcal{V}_{VE} \rangle (f, c_y, r_y)$

c_x, r_x

c_x, r_x, b

Return $b \wedge c_x \stackrel{?}{=} \mathsf{H}(x, r_x)$

Correct input fingerprint

$\Rightarrow c_y^*$ fingerprints correct output

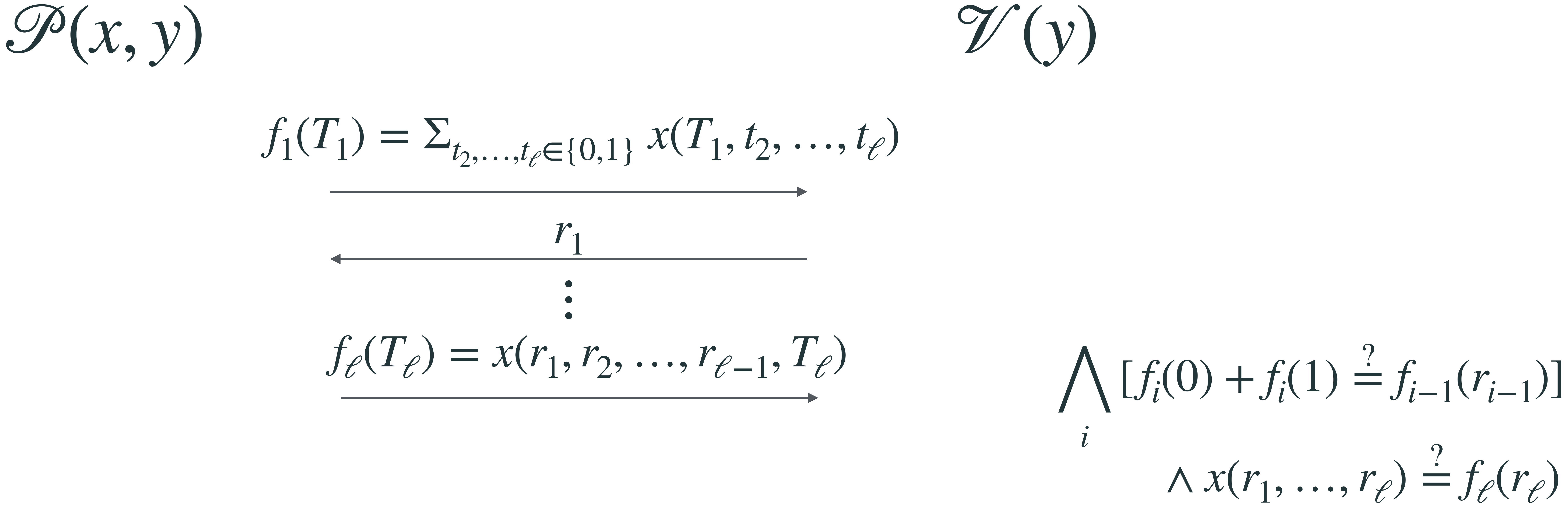
Soundness of VE subroutine: for random r_x, r_y and any unbounded $\mathcal{A}$

$$\Pr\left[\begin{array}{c} c_y^* \neq \mathsf{H}(f(x), r_y) \\ \wedge\, b \end{array} \;\middle|\; \begin{array}{c} (f, c_y^*, x) \leftarrow \mathcal{A}(r_y) \\ (c_x^*; r_x; b) \leftarrow \left\langle \mathcal{A}(x), \mathcal{V}_{VE}(r_x) \right\rangle (f, c_y^*, r_y) \\ c_x^* = \mathsf{H}(x, r_x) \end{array}\right] = \text{negl}$$

Sound VE + Binding Fingerprint ⇒ Sound IP

Sumcheck protocol as VE-based IP [LFKN92]

$x \in \mathbb{F}[T_1, \ldots, T_\ell]$ multilinear polynomial. **Goal:** prove $y = f(x) = \Sigma_{t_1,\ldots,t_\ell \in \{0,1\}} x(t_1, \ldots, t_\ell)$



$\mathscr{P}(x, y)$ $\mathscr{V}(y)$

$$\xrightarrow{f_1(T_1) = \Sigma_{t_2,\ldots,t_\ell \in \{0,1\}} x(T_1, t_2, \ldots, t_\ell)}$$

$$\xleftarrow{r_1}$$

$$\vdots$$

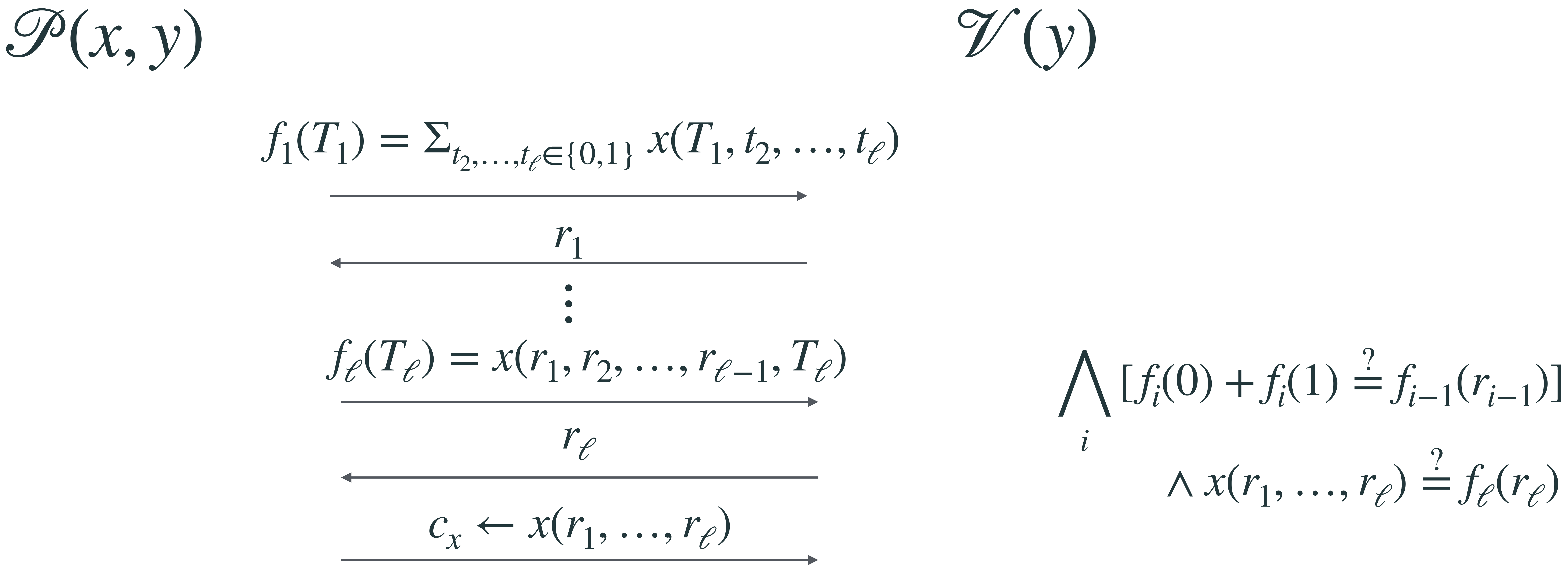
$$\xrightarrow{f_\ell(T_\ell) = x(r_1, r_2, \ldots, r_{\ell-1}, T_\ell)}$$

$$\bigwedge_i [f_i(0) + f_i(1) \overset{?}{=} f_{i-1}(r_{i-1})]$$

$$\wedge\, x(r_1, \ldots, r_\ell) \overset{?}{=} f_\ell(r_\ell)$$

Sumcheck protocol as VE-based IP [LFKN92]

$x \in \mathbb{F}[T_1, \ldots, T_\ell]$ multilinear polynomial. **Goal:** prove $y = f(x) = \Sigma_{t_1,\ldots,t_\ell \in \{0,1\}} x(t_1, \ldots, t_\ell)$



$\mathscr{P}(x, y)$

$\mathscr{V}(y)$

$$f_1(T_1) = \Sigma_{t_2,\ldots,t_\ell \in \{0,1\}} x(T_1, t_2, \ldots, t_\ell) \longrightarrow$$

$$\xleftarrow{\quad r_1 \quad}$$

$$\vdots$$

$$f_\ell(T_\ell) = x(r_1, r_2, \ldots, r_{\ell-1}, T_\ell) \longrightarrow$$

$$\xleftarrow{\quad r_\ell \quad}$$

$$c_x \leftarrow x(r_1, \ldots, r_\ell) \longrightarrow$$

$$\bigwedge_i [f_i(0) + f_i(1) \overset{?}{=} f_{i-1}(r_{i-1})]$$

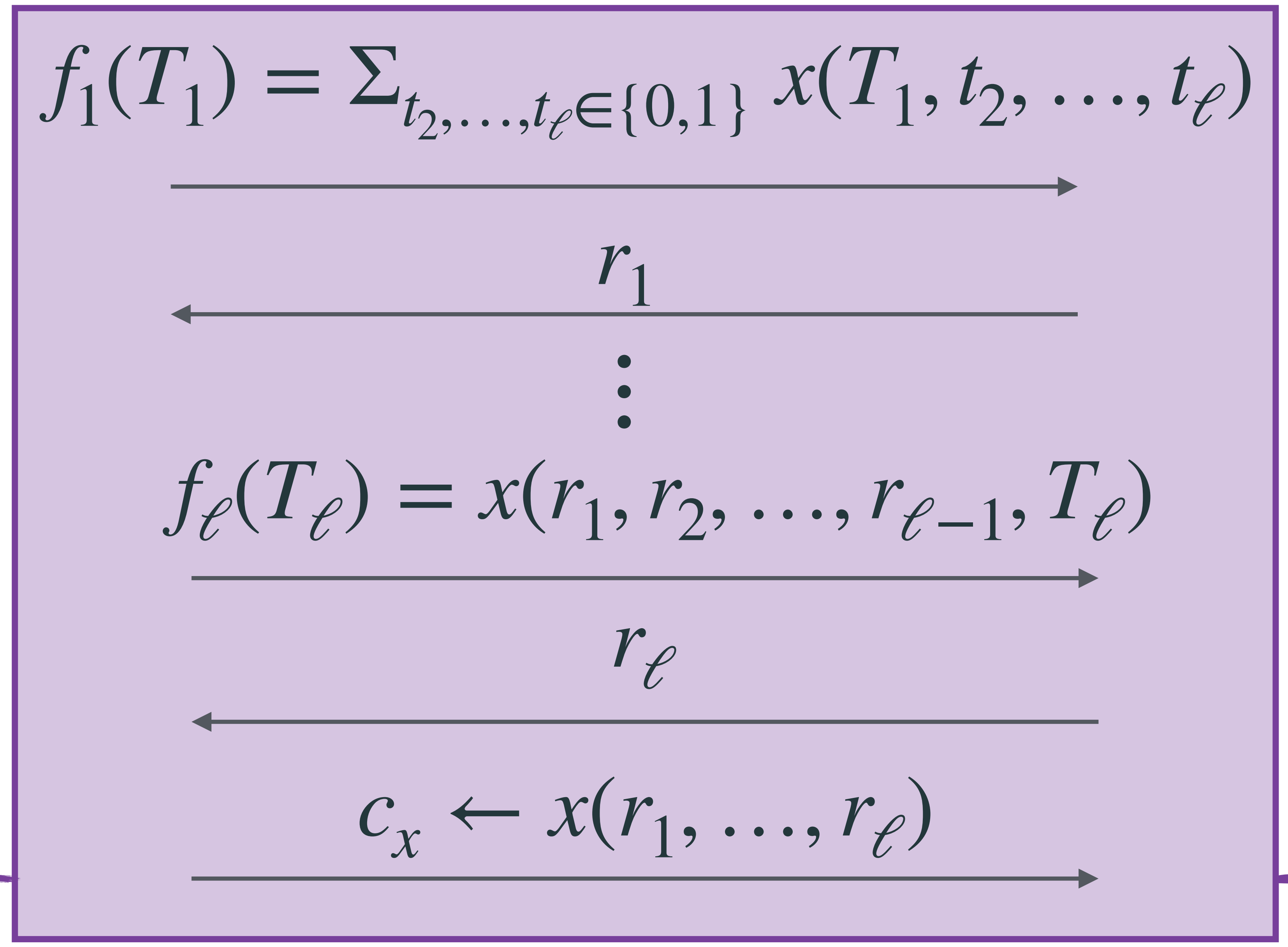
$$\wedge\, x(r_1, \ldots, r_\ell) \overset{?}{=} f_\ell(r_\ell)$$

Sumcheck protocol as VE-based IP [LFKN92]

$x \in \mathbb{F}[T_1, \dots, T_\ell]$ multilinear polynomial. **Goal:** prove $y = f(x) = \Sigma_{t_1,\dots,t_\ell \in \{0,1\}} x(t_1, \dots, t_\ell)$

$\mathscr{P}(x, y)$

$\mathscr{V}(y)$



$f_1(T_1) = \Sigma_{t_2,\dots,t_\ell \in \{0,1\}} x(T_1, t_2, \dots, t_\ell)$

r_1

$\vdots$

$f_\ell(T_\ell) = x(r_1, r_2, \dots, r_{\ell-1}, T_\ell)$

r_ℓ

$c_x \leftarrow x(r_1, \dots, r_\ell)$

$\bigwedge_i [f_i(0) + f_i(1) \overset{?}{=} f_{i-1}(r_{i-1})]$

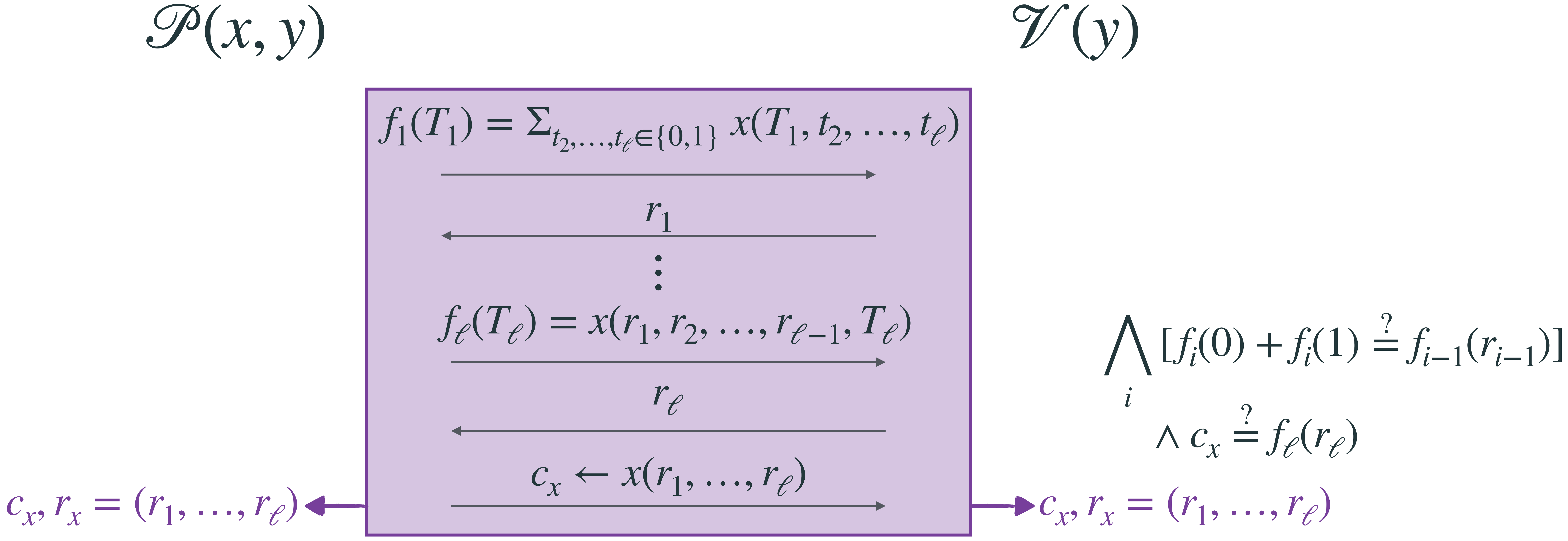
$\wedge\, x(r_1, \dots, r_\ell) \overset{?}{=} f_\ell(r_\ell)$

$c_x, r_x = (r_1, \dots, r_\ell)$

$c_x, r_x = (r_1, \dots, r_\ell)$

Sumcheck protocol as VE-based IP [LFKN92]

$x \in \mathbb{F}[T_1, \ldots, T_\ell]$ multilinear polynomial. **Goal:** prove $y = f(x) = \Sigma_{t_1,\ldots,t_\ell \in \{0,1\}} x(t_1, \ldots, t_\ell)$



$\mathscr{P}(x, y)$

$\mathscr{V}(y)$

$f_1(T_1) = \Sigma_{t_2,\ldots,t_\ell \in \{0,1\}} x(T_1, t_2, \ldots, t_\ell)$

r_1

$\vdots$

$f_\ell(T_\ell) = x(r_1, r_2, \ldots, r_{\ell-1}, T_\ell)$

r_ℓ

$c_x \leftarrow x(r_1, \ldots, r_\ell)$

$$\bigwedge_i [f_i(0) + f_i(1) \overset{?}{=} f_{i-1}(r_{i-1})]$$

$$\wedge\, c_x \overset{?}{=} f_\ell(r_\ell)$$

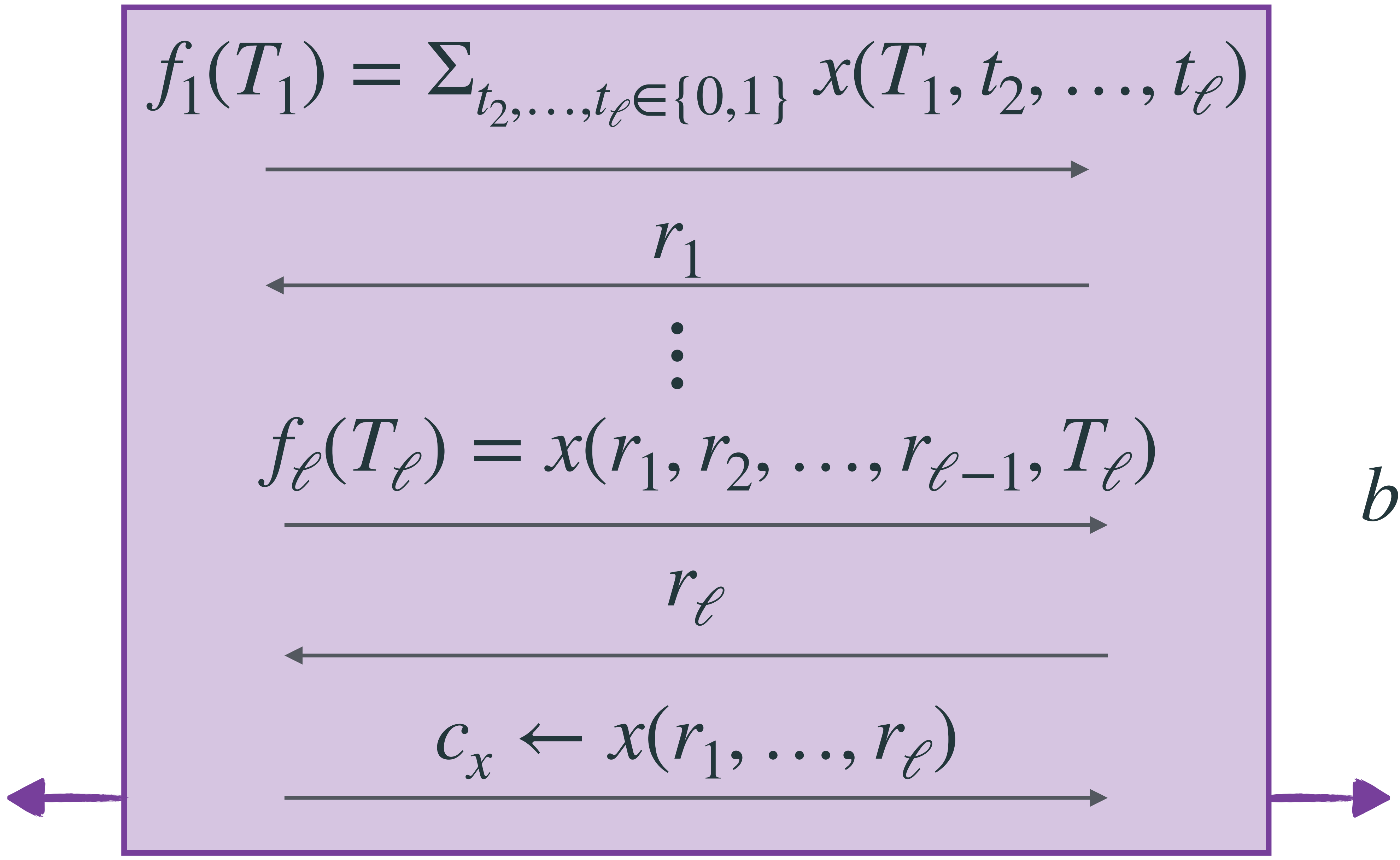
$c_x, r_x = (r_1, \ldots, r_\ell)$

$c_x, r_x = (r_1, \ldots, r_\ell)$

Sumcheck protocol as VE-based IP [LFKN92]

$x \in \mathbb{F}[T_1, \ldots, T_\ell]$ multilinear polynomial. **Goal:** prove $y = f(x) = \Sigma_{t_1,\ldots,t_\ell \in \{0,1\}} x(t_1, \ldots, t_\ell)$

$\mathscr{P}(x, y)$ $\mathscr{V}(y)$



$$f_1(T_1) = \Sigma_{t_2,\ldots,t_\ell \in \{0,1\}} x(T_1, t_2, \ldots, t_\ell) \xrightarrow{\hspace{3cm}}$$

$$\xleftarrow{\quad r_1 \quad}$$

$$\vdots$$

$$f_\ell(T_\ell) = x(r_1, r_2, \ldots, r_{\ell-1}, T_\ell) \xrightarrow{\hspace{3cm}}$$

$$\xleftarrow{\quad r_\ell \quad}$$

$$\xrightarrow{c_x \leftarrow x(r_1, \ldots, r_\ell)}$$

$$b \leftarrow \bigwedge_i [f_i(0) + f_i(1) \overset{?}{=} f_{i-1}(r_{i-1})]$$

$$\wedge\, c_x \overset{?}{=} f_\ell(r_\ell)$$

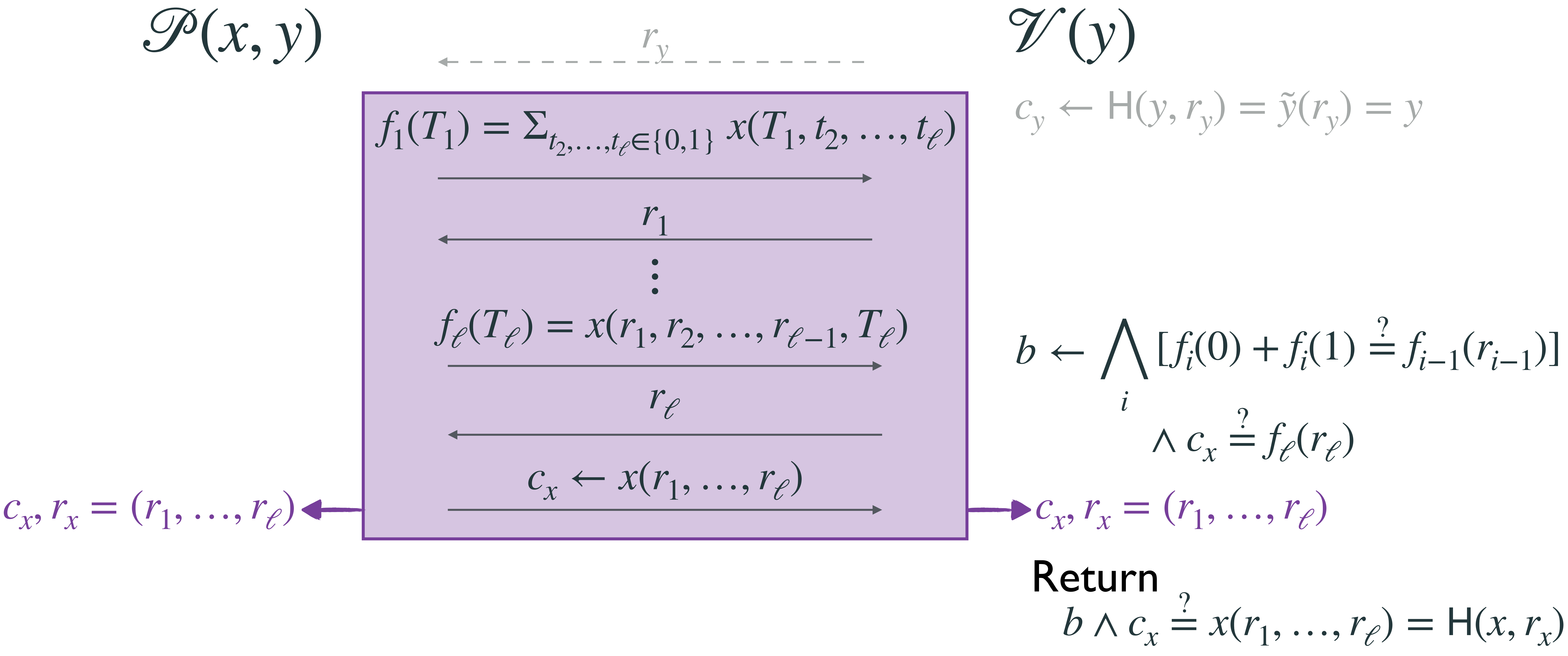
$c_x, r_x = (r_1, \ldots, r_\ell)$ $c_x, r_x = (r_1, \ldots, r_\ell)$

Return

$$b \wedge c_x \overset{?}{=} x(r_1, \ldots, r_\ell) = \mathsf{H}(x, r_x)$$

Sumcheck protocol as VE-based IP [LFKN92]

$x \in \mathbb{F}[T_1, \ldots, T_\ell]$ multilinear polynomial. **Goal:** prove $y = f(x) = \Sigma_{t_1,\ldots,t_\ell \in \{0,1\}} x(t_1, \ldots, t_\ell)$



$\mathscr{P}(x, y)$

$\mathscr{V}(y)$

$$\xleftarrow{\quad r_y \quad}$$

$c_y \leftarrow \mathsf{H}(y, r_y) = \tilde{y}(r_y) = y$

$$\xrightarrow{f_1(T_1) = \Sigma_{t_2,\ldots,t_\ell \in \{0,1\}} x(T_1, t_2, \ldots, t_\ell)}$$

$$\xleftarrow{\quad r_1 \quad}$$

$$\vdots$$

$$\xrightarrow{f_\ell(T_\ell) = x(r_1, r_2, \ldots, r_{\ell-1}, T_\ell)}$$

$$\xleftarrow{\quad r_\ell \quad}$$

$$\xrightarrow{c_x \leftarrow x(r_1, \ldots, r_\ell)}$$

$c_x, r_x = (r_1, \ldots, r_\ell)$

$c_x, r_x = (r_1, \ldots, r_\ell)$

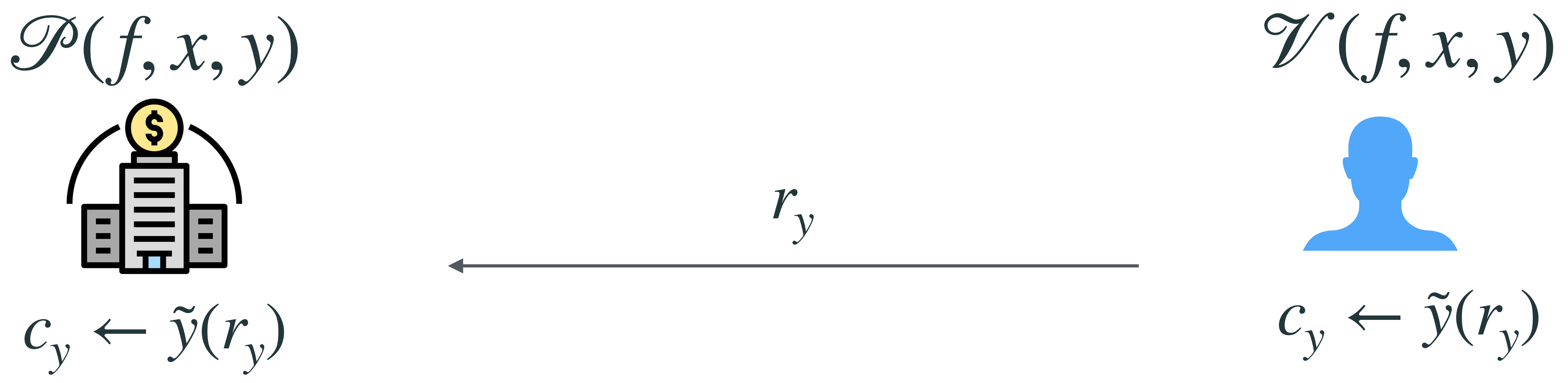
$$b \leftarrow \bigwedge_i [f_i(0) + f_i(1) \overset{?}{=} f_{i-1}(r_{i-1})]$$

$$\wedge\, c_x \overset{?}{=} f_\ell(r_\ell)$$

Return

$$b \wedge c_x \overset{?}{=} x(r_1, \ldots, r_\ell) = \mathsf{H}(x, r_x)$$

GKR as VE-based IP



$\mathscr{P}(f, x, y)$

$\mathscr{V}(f, x, y)$

r_y

$c_y \leftarrow \tilde{y}(r_y)$

$c_y \leftarrow \tilde{y}(r_y)$

GKR as VE-based IP

$\mathcal{P}(f, x, y)$

$\mathcal{V}(f, x, y)$

r_y

$\xleftarrow{\hspace{3cm}}$

$c_y \leftarrow \tilde{y}(r_y)$

$c_y \leftarrow \tilde{y}(r_y)$

Sumcheck claim: $c_y = \tilde{V}_d(r_y) = \sum_{b_1, b_2 \in \{0,1\}^\ell} \tilde{add}_{d-1}(r_y, b_1, b_2)(\tilde{V}_{d-1}(b_1) + \tilde{V}_{d-1}(b_2)) + \tilde{mult}_{d-1}(r_y, b_1, b_2)(\tilde{V}_{d-1}(b_1) \cdot \tilde{V}_{d-1}(b_2))$

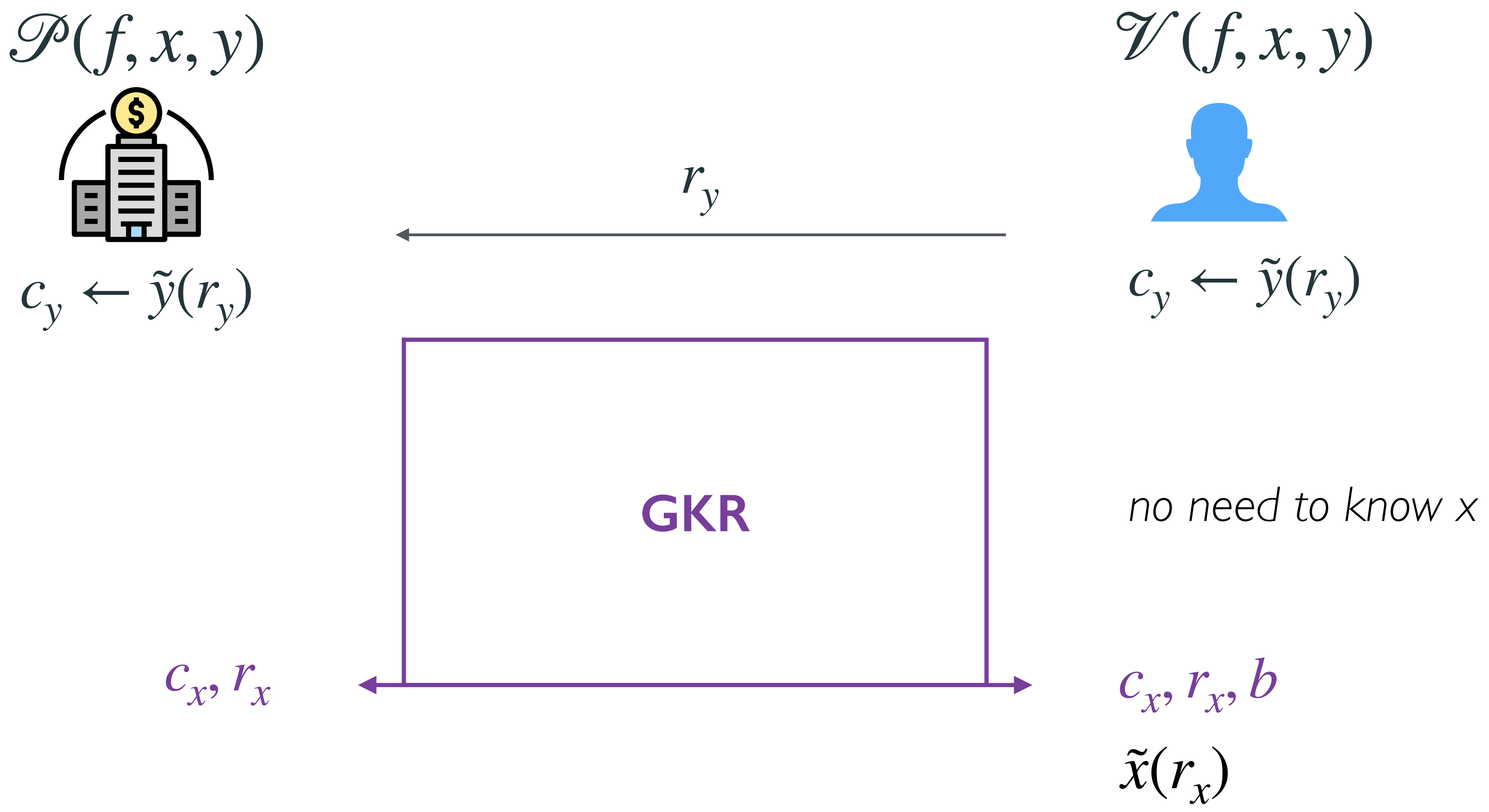
⋮

no need to know x

Sumcheck claim: $c_1 = \tilde{V}_1(r_1) = \sum_{b_1, b_2 \in \{0,1\}^\ell} \tilde{add}_1(r_1, b_1, b_2)(\tilde{x}(b_1) + \tilde{x}(b_2)) + \tilde{mult}_1(r_1, b_1, b_2)(\tilde{x}(b_1) \cdot \tilde{x}(b_2))$

$\tilde{x}(r_x)$

GKR as VE-based IP



$\mathscr{P}(f, x, y)$

$\mathscr{V}(f, x, y)$

r_y

$c_y \leftarrow \tilde{y}(r_y)$

$c_y \leftarrow \tilde{y}(r_y)$

GKR

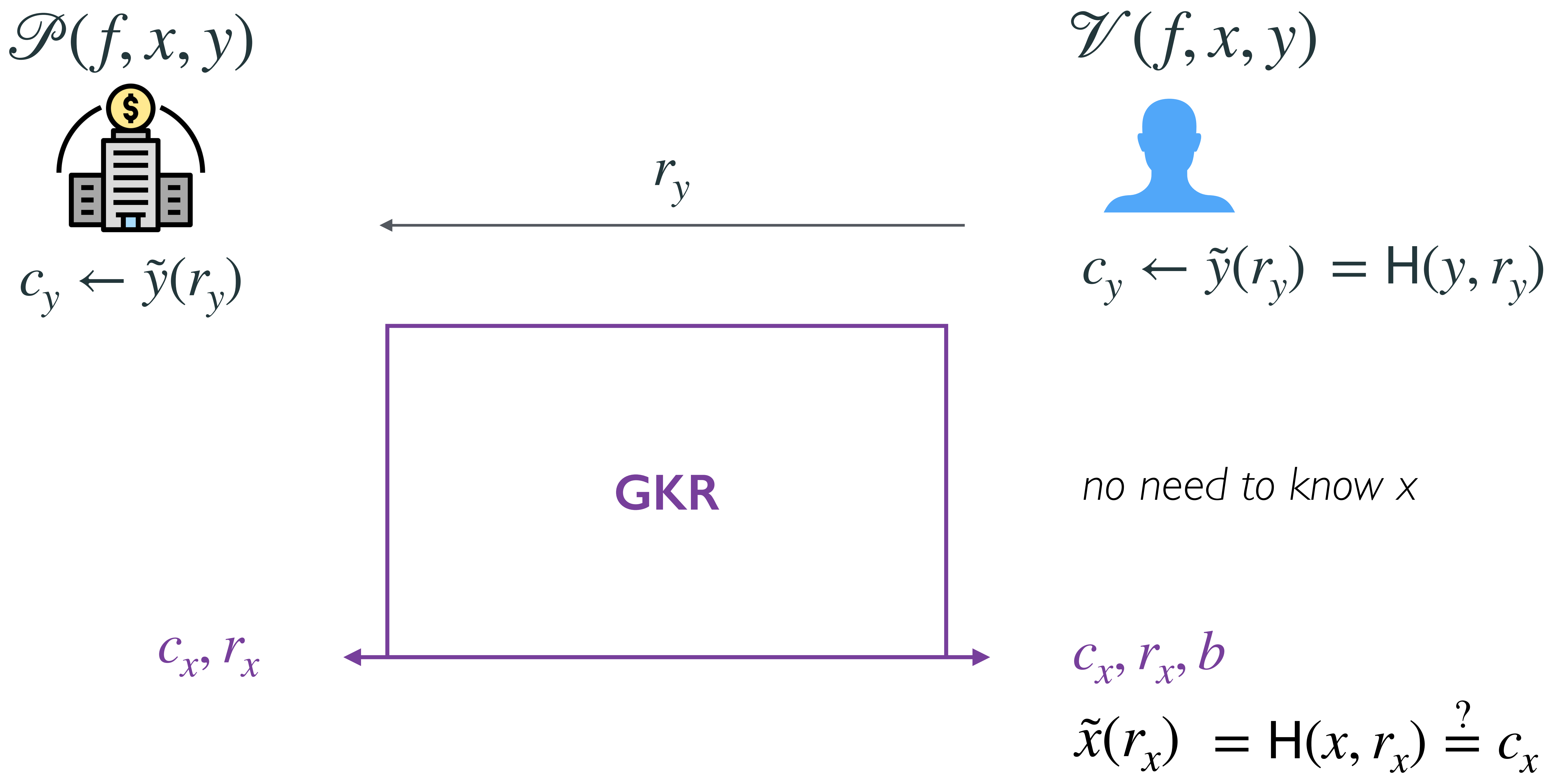
no need to know x

c_x, r_x

c_x, r_x, b

$\tilde{x}(r_x)$

GKR as VE-based IP



$\mathscr{P}(f, x, y)$

$\mathscr{V}(f, x, y)$

r_y

$c_y \leftarrow \tilde{y}(r_y)$

$c_y \leftarrow \tilde{y}(r_y) \; = \mathsf{H}(y, r_y)$

GKR

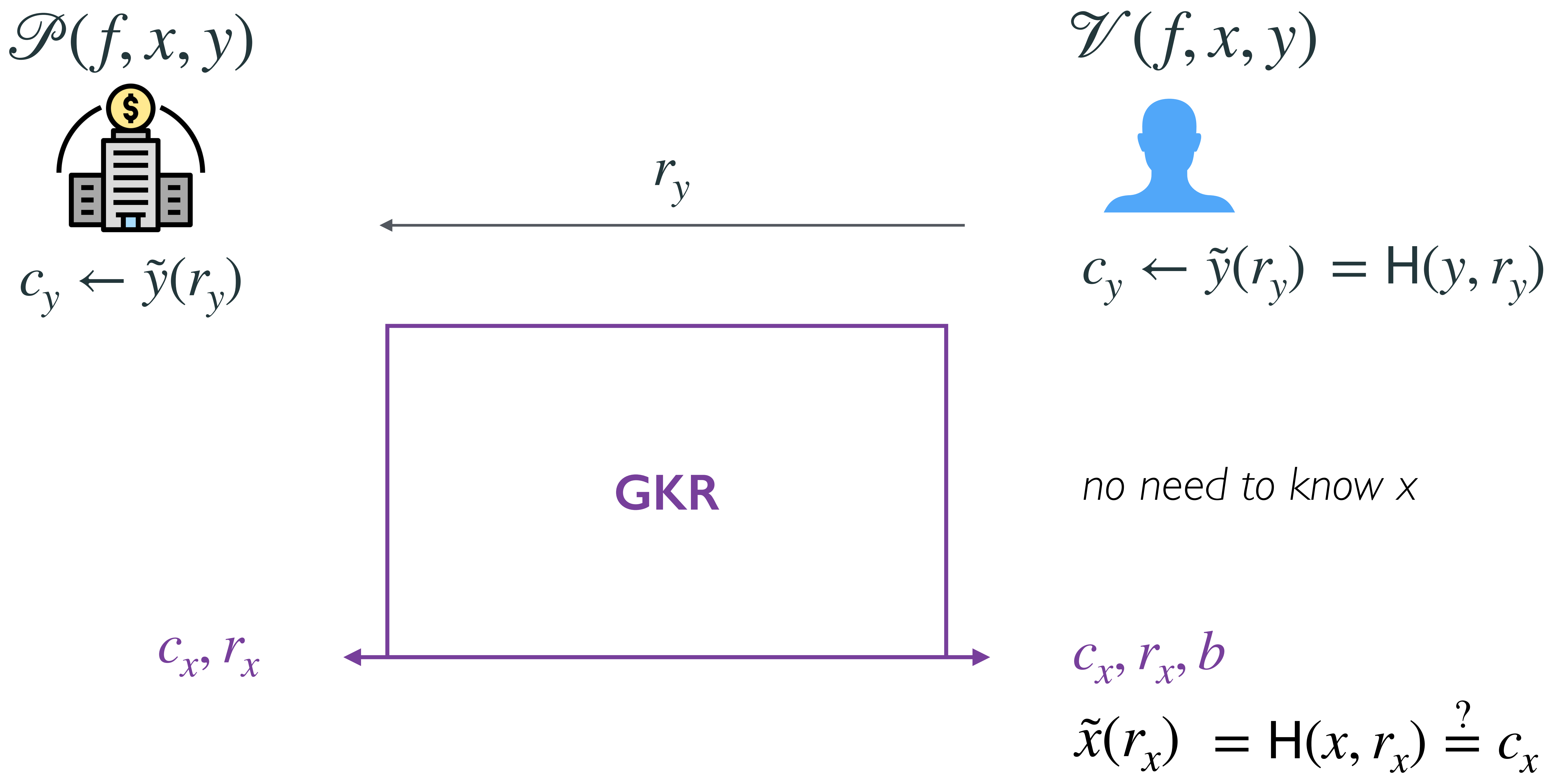
no need to know x

c_x, r_x

c_x, r_x, b

$\tilde{x}(r_x) \; = \mathsf{H}(x, r_x) \stackrel{?}{=} c_x$

GKR as VE-based IP



$\mathscr{P}(f, x, y)$

$\mathscr{V}(f, x, y)$

r_y

$c_y \leftarrow \tilde{y}(r_y)$

$c_y \leftarrow \tilde{y}(r_y) \ = \mathsf{H}(y, r_y)$

GKR

no need to know x

c_x, r_x

c_x, r_x, b

$\tilde{x}(r_x) \ = \mathsf{H}(x, r_x) \overset{?}{=} c_x$

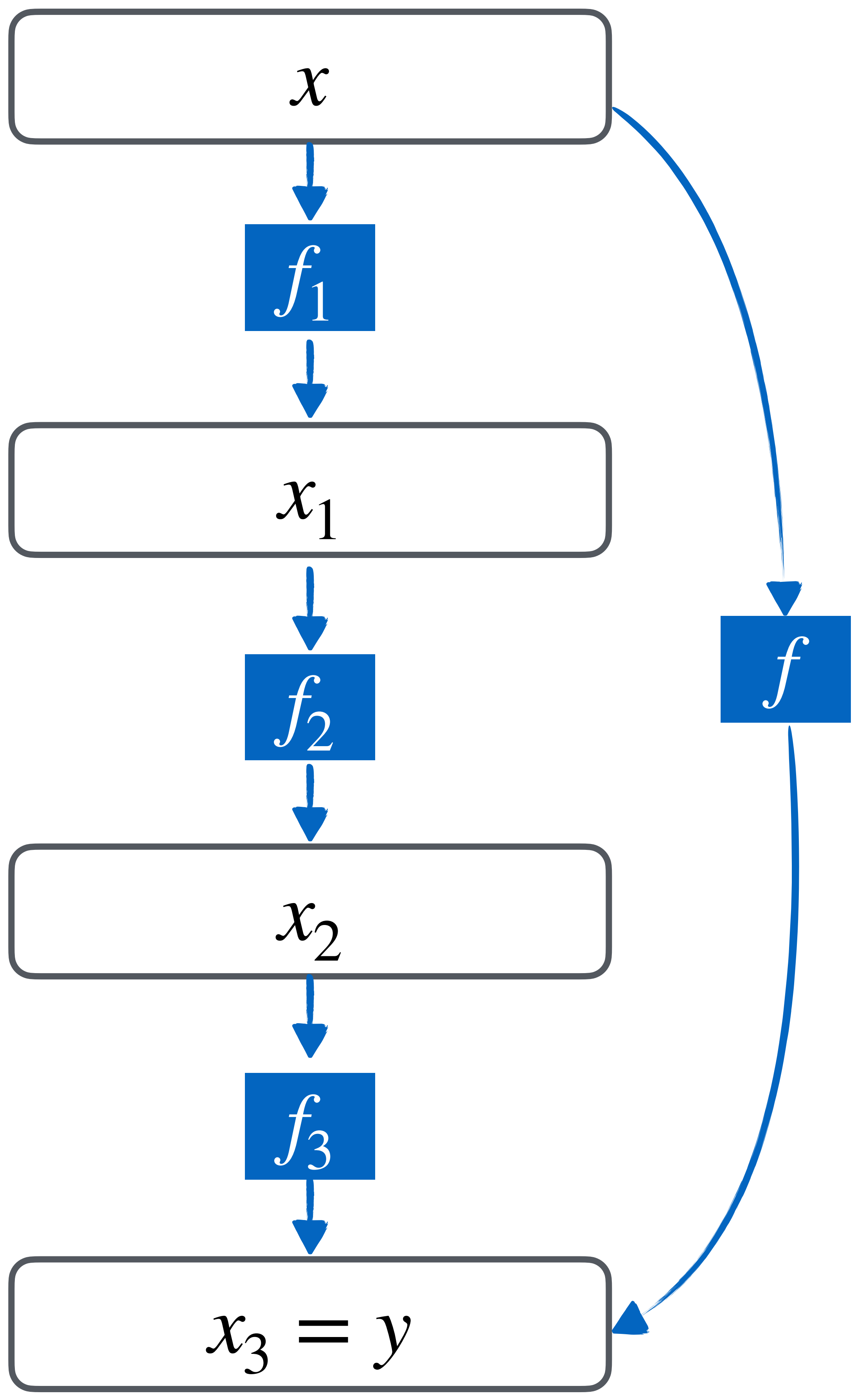
And even the layer subprotocols of GKR can be seen as VEs

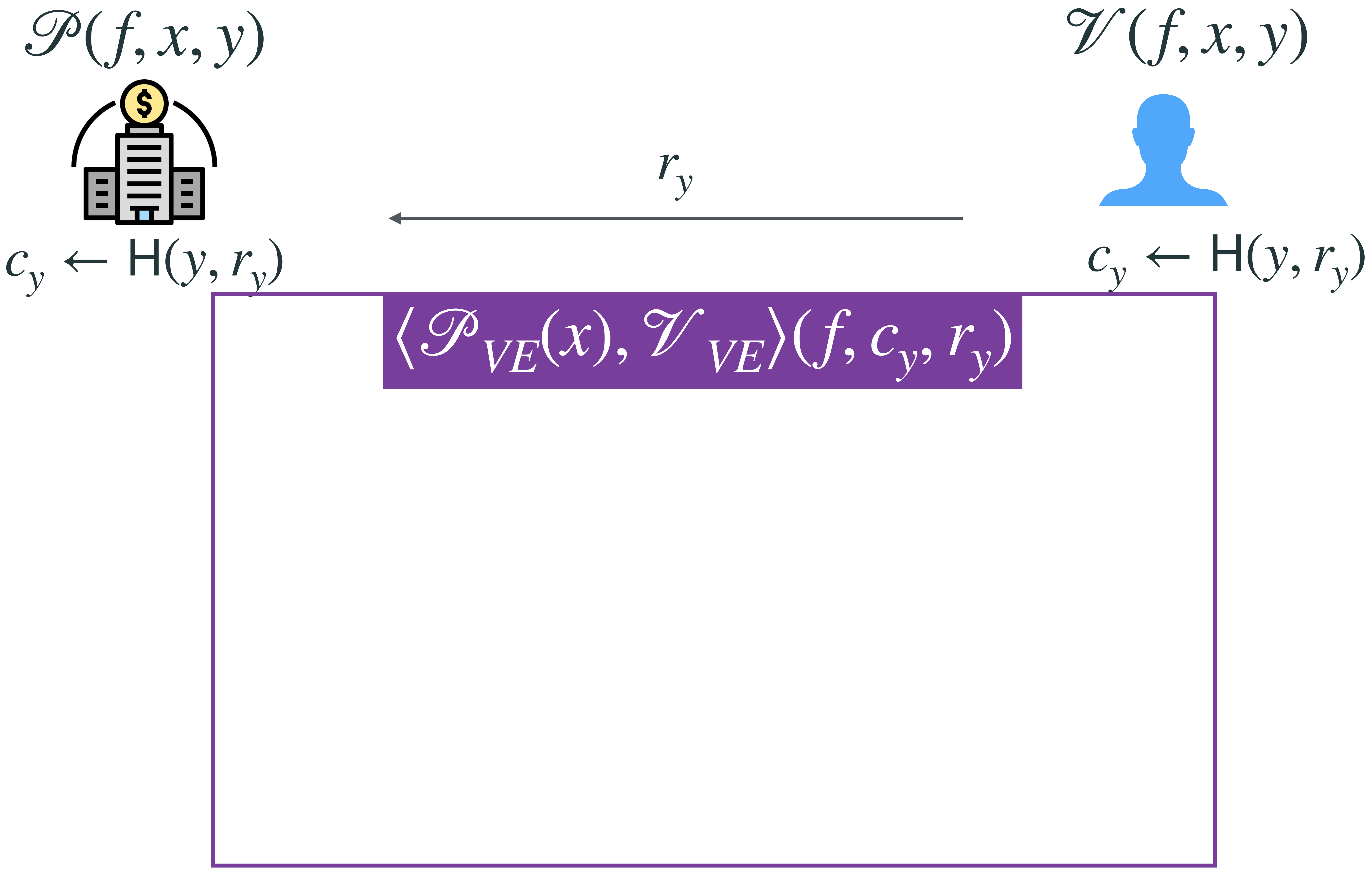
Sequential composition of VEs

VE₁ for $z = f_1(x)$ + **VE₂** for $y = f_2(z, w)$ ➡ **VE** for $y = f(x, w) = f_2(f_1(x), w)$

Sequential composition of VEs

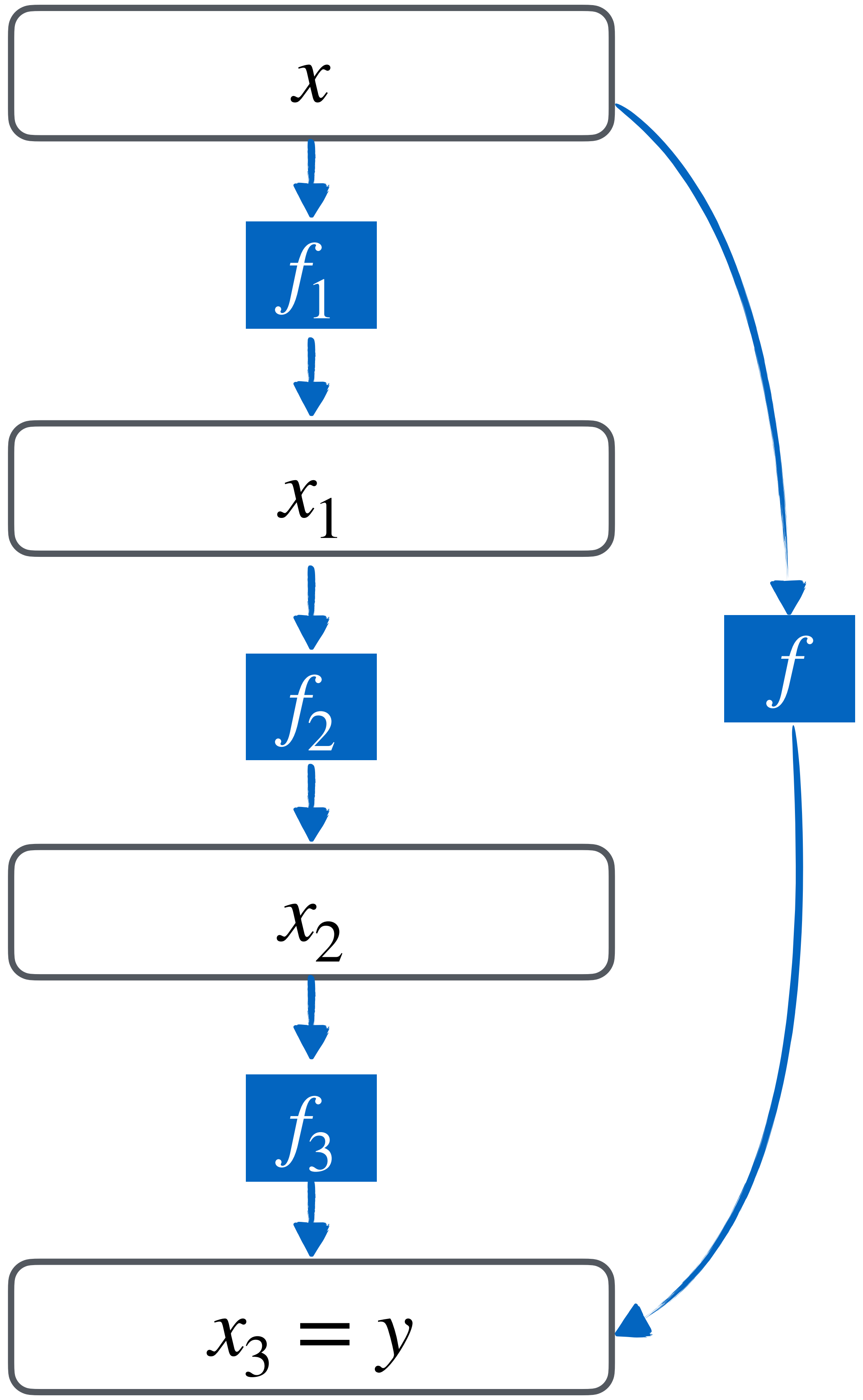
VE₁ for $z = f_1(x)$ + **VE₂** for $y = f_2(z, w)$ → **VE** for $y = f(x, w) = f_2(f_1(x), w)$

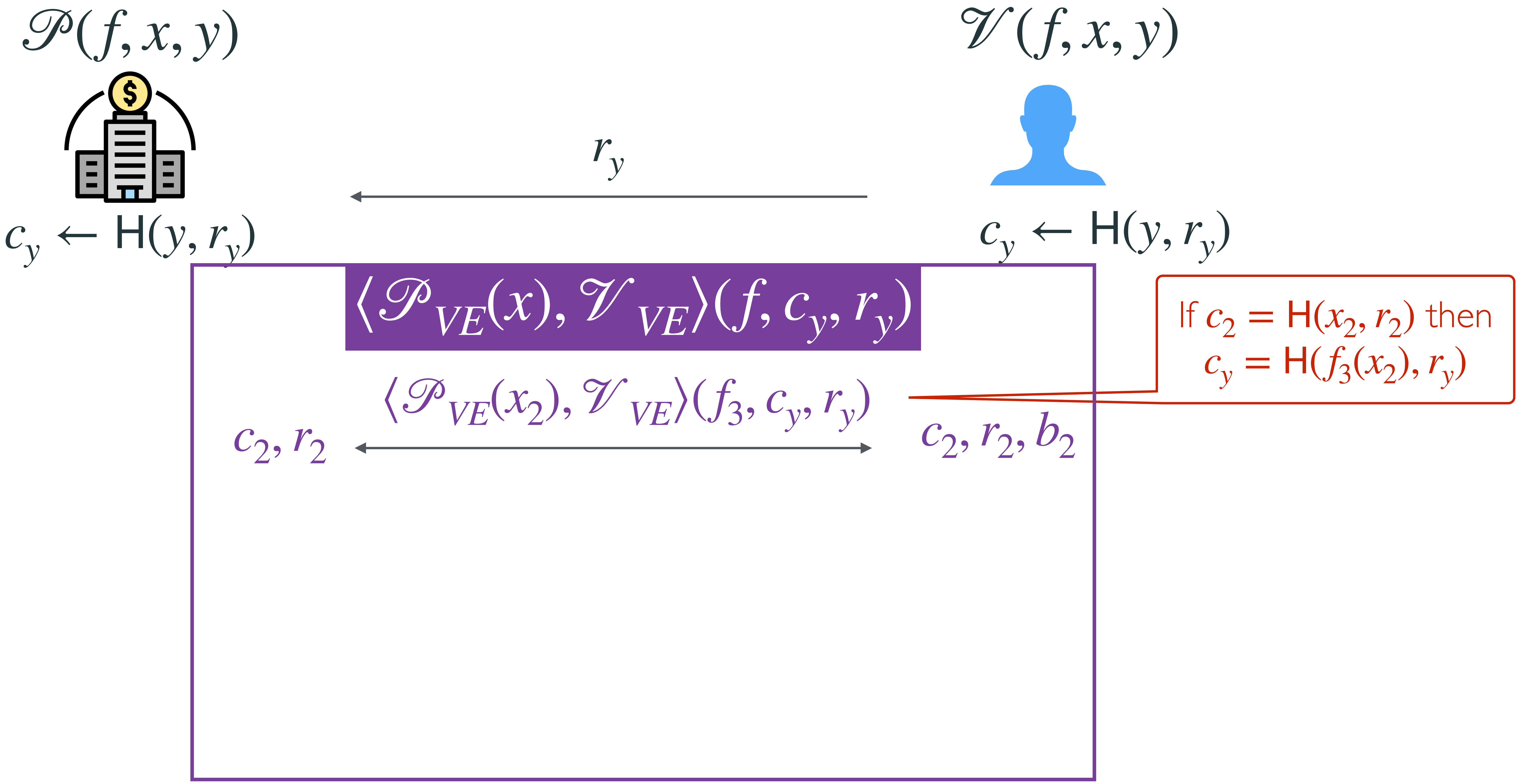




Sequential composition of VEs

VE₁ for $z = f_1(x)$ + **VE₂** for $y = f_2(z, w)$ ➡ **VE** for $y = f(x, w) = f_2(f_1(x), w)$





$\mathcal{P}(f, x, y)$

$\mathcal{V}(f, x, y)$

r_y

$c_y \leftarrow \mathsf{H}(y, r_y)$

$c_y \leftarrow \mathsf{H}(y, r_y)$

$\langle \mathcal{P}_{VE}(x), \mathcal{V}_{VE} \rangle (f, c_y, r_y)$

$\langle \mathcal{P}_{VE}(x_2), \mathcal{V}_{VE} \rangle (f_3, c_y, r_y)$

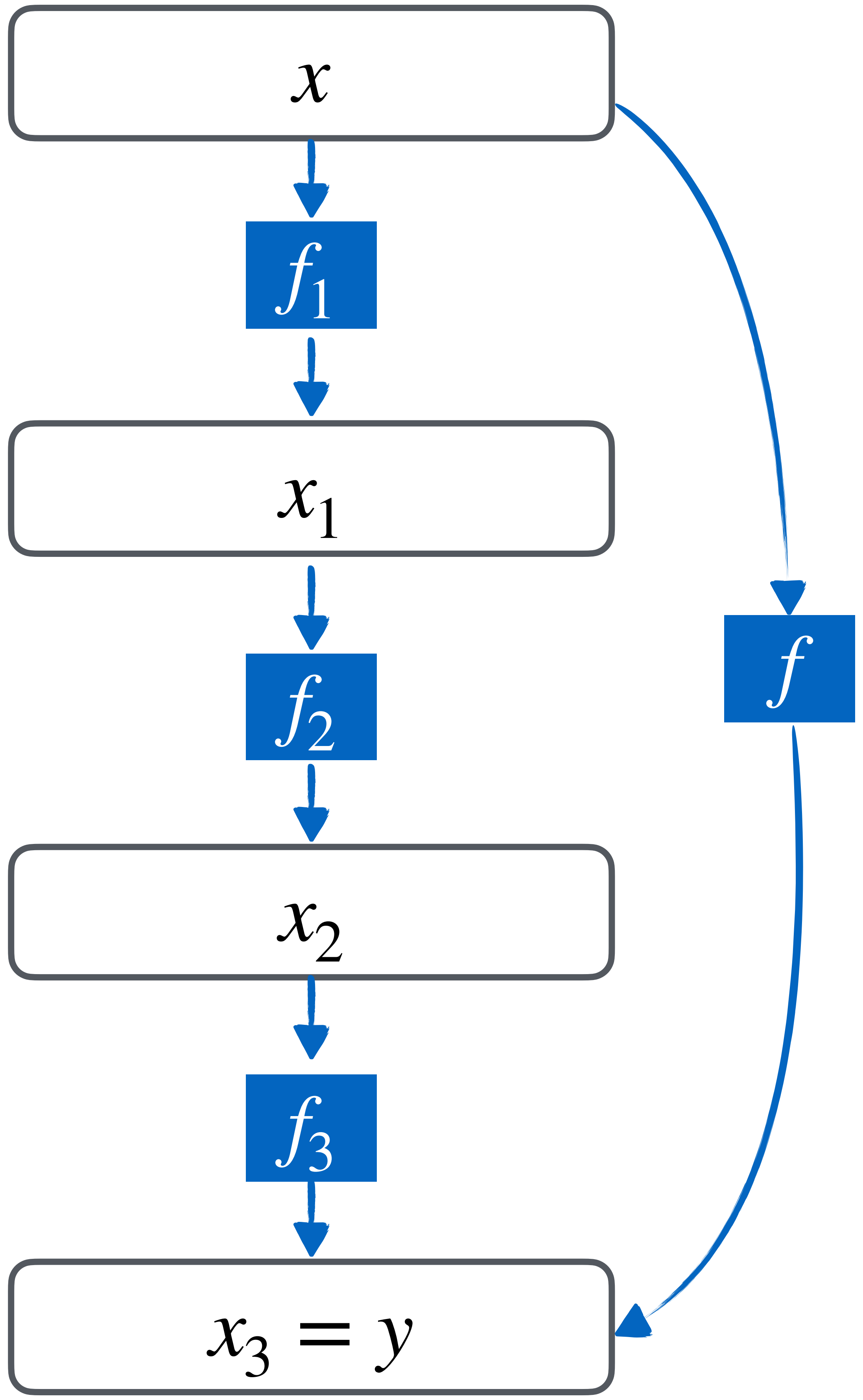
c_2, r_2

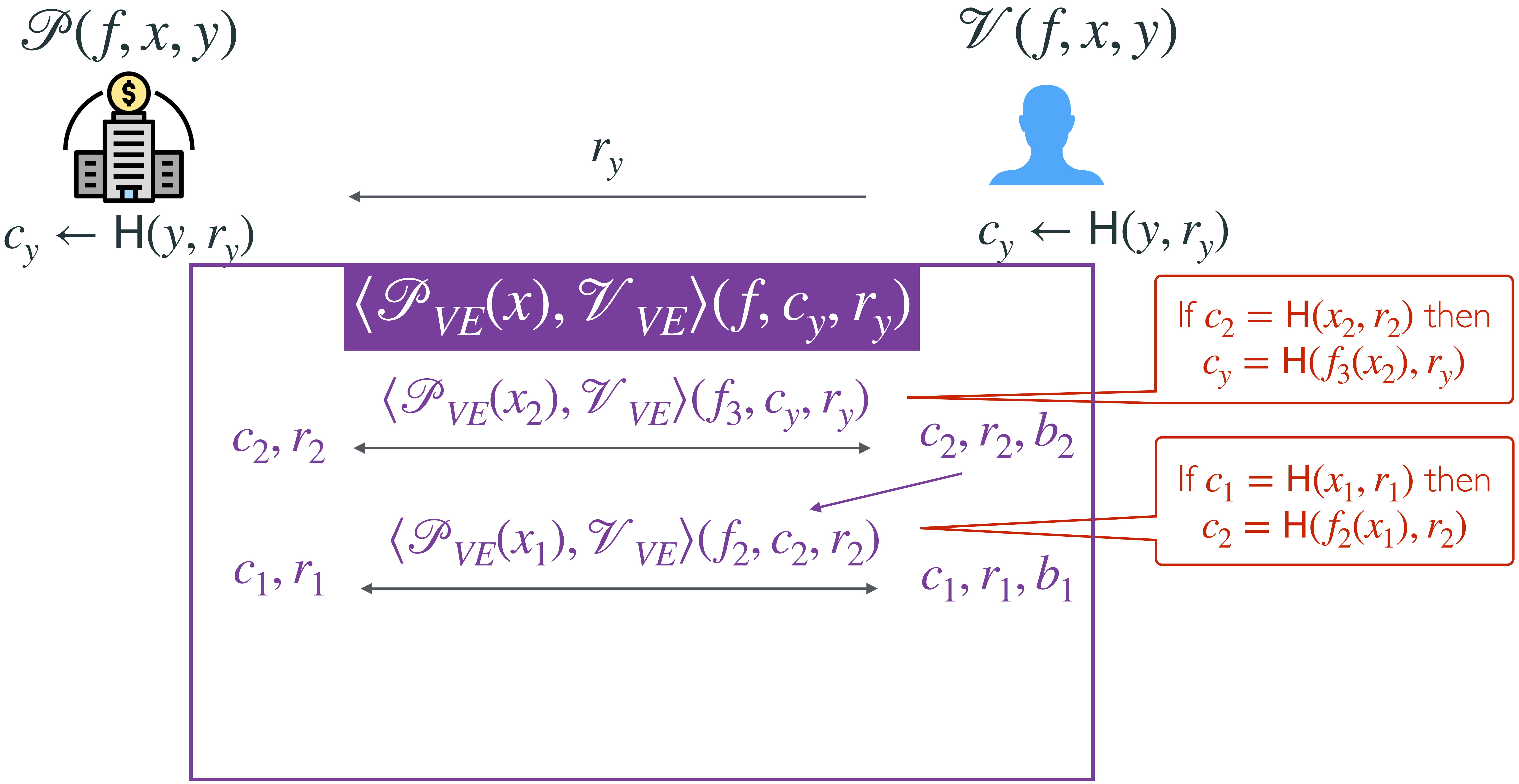
c_2, r_2, b_2

If $c_2 = \mathsf{H}(x_2, r_2)$ then $c_y = \mathsf{H}(f_3(x_2), r_y)$

Sequential composition of VEs

VE₁ for $z = f_1(x)$ + **VE₂** for $y = f_2(z, w)$ → **VE** for $y = f(x, w) = f_2(f_1(x), w)$





$\mathcal{P}(f, x, y)$

$\mathcal{V}(f, x, y)$

r_y

$c_y \leftarrow \mathsf{H}(y, r_y)$

$c_y \leftarrow \mathsf{H}(y, r_y)$

$\langle \mathcal{P}_{VE}(x), \mathcal{V}_{VE} \rangle (f, c_y, r_y)$

$\langle \mathcal{P}_{VE}(x_2), \mathcal{V}_{VE} \rangle (f_3, c_y, r_y)$

c_2, r_2

c_2, r_2, b_2

$\langle \mathcal{P}_{VE}(x_1), \mathcal{V}_{VE} \rangle (f_2, c_2, r_2)$

c_1, r_1

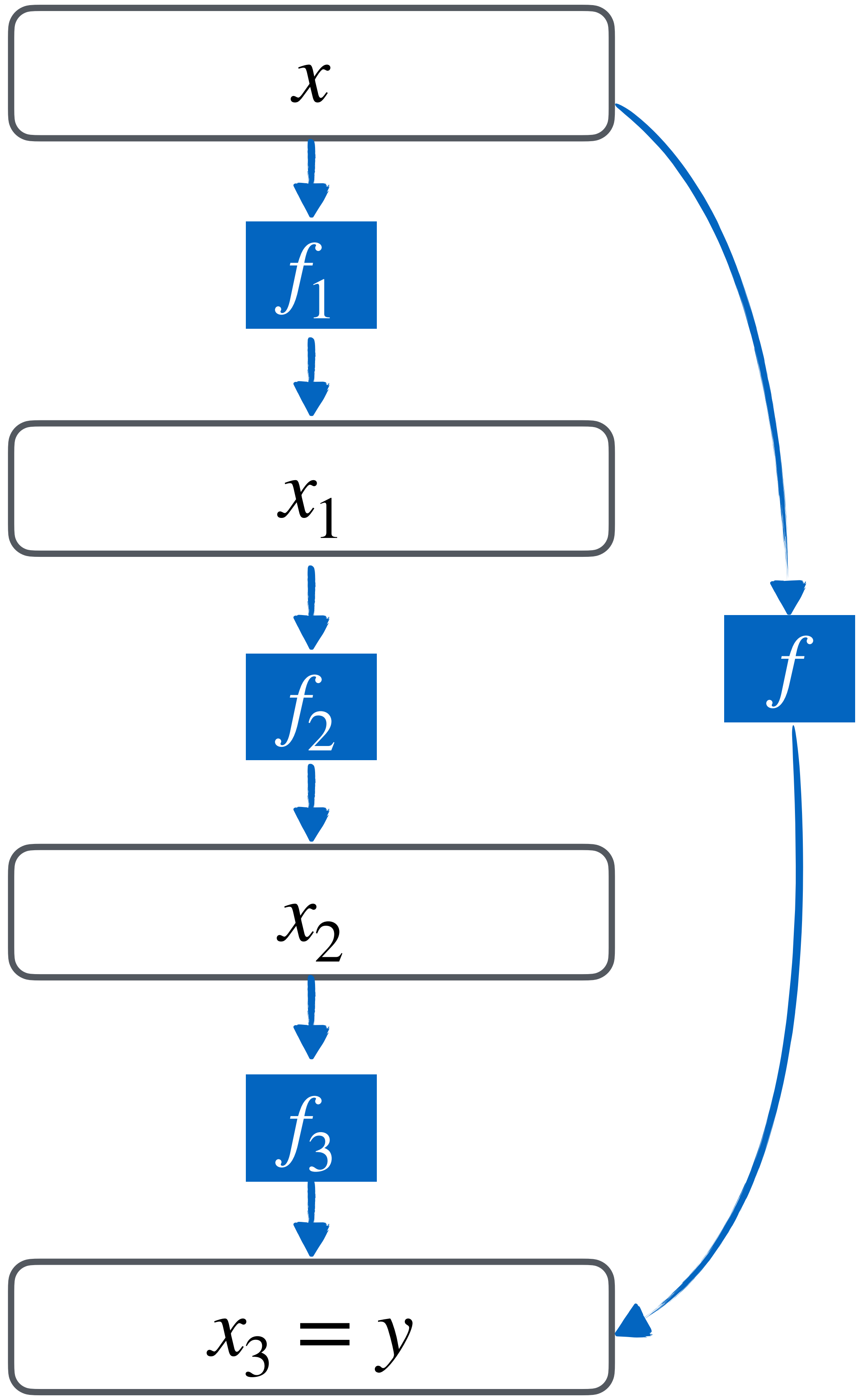
c_1, r_1, b_1

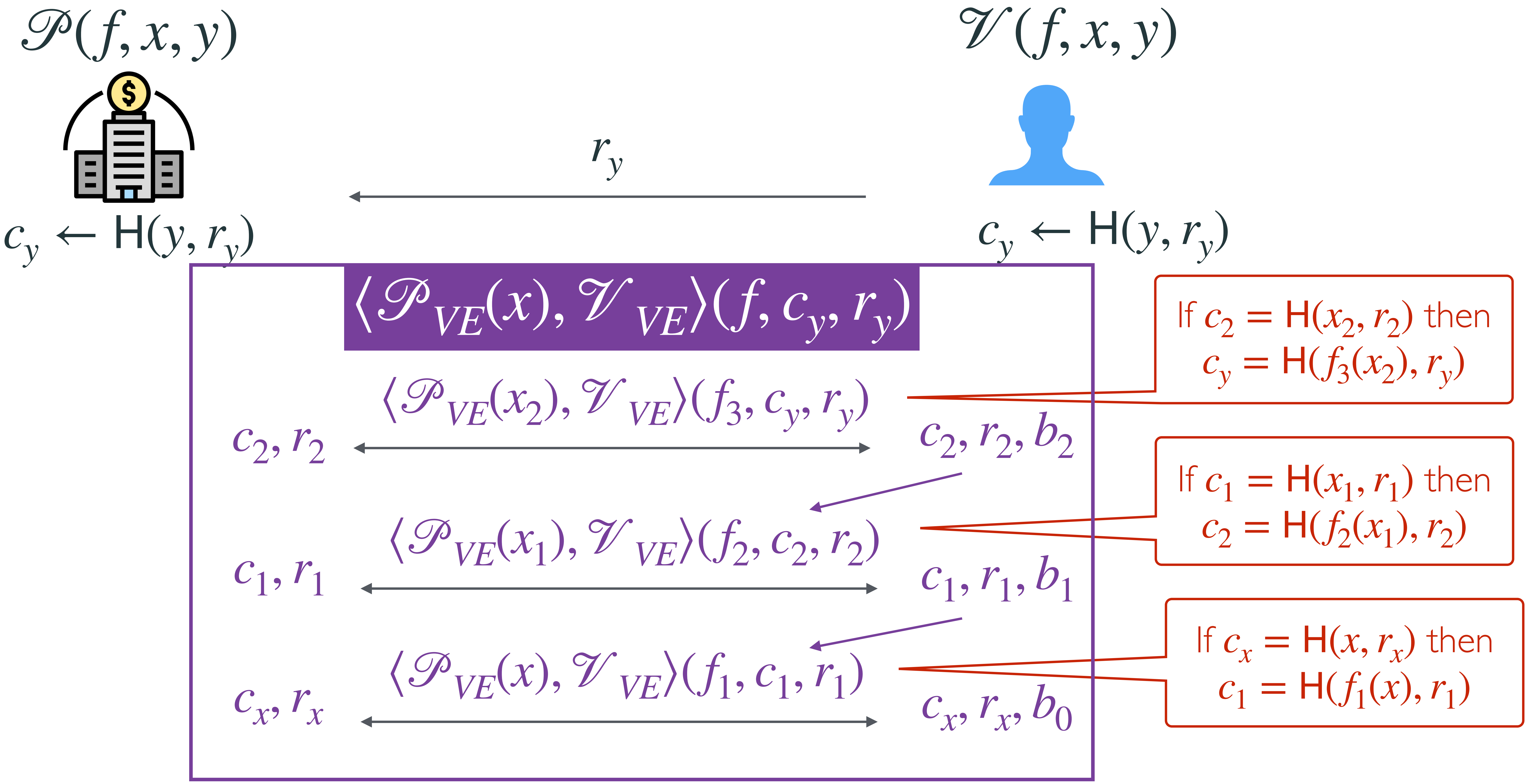
If $c_2 = \mathsf{H}(x_2, r_2)$ then $c_y = \mathsf{H}(f_3(x_2), r_y)$

If $c_1 = \mathsf{H}(x_1, r_1)$ then $c_2 = \mathsf{H}(f_2(x_1), r_2)$

Sequential composition of VEs

VE₁ for $z = f_1(x)$ + **VE₂** for $y = f_2(z, w)$ → **VE** for $y = f(x, w) = f_2(f_1(x), w)$





$\mathcal{P}(f, x, y)$

$\mathcal{V}(f, x, y)$

r_y

$c_y \leftarrow \mathsf{H}(y, r_y)$

$c_y \leftarrow \mathsf{H}(y, r_y)$

$\langle \mathcal{P}_{VE}(x), \mathcal{V}_{VE}\rangle(f, c_y, r_y)$

c_2, r_2 — $\langle \mathcal{P}_{VE}(x_2), \mathcal{V}_{VE}\rangle(f_3, c_y, r_y)$ — c_2, r_2, b_2

c_1, r_1 — $\langle \mathcal{P}_{VE}(x_1), \mathcal{V}_{VE}\rangle(f_2, c_2, r_2)$ — c_1, r_1, b_1

c_x, r_x — $\langle \mathcal{P}_{VE}(x), \mathcal{V}_{VE}\rangle(f_1, c_1, r_1)$ — c_x, r_x, b_0

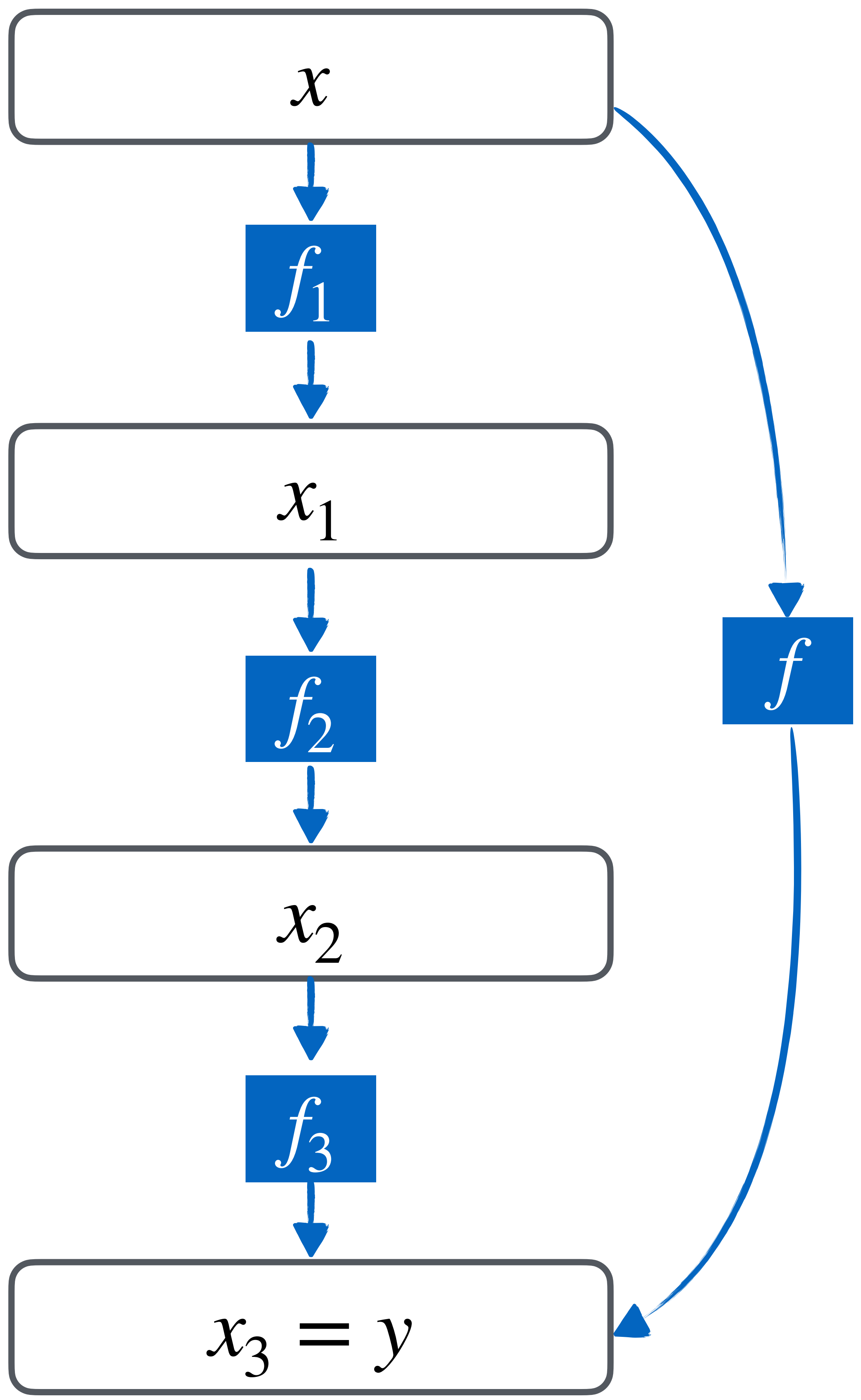
If $c_2 = \mathsf{H}(x_2, r_2)$ then $c_y = \mathsf{H}(f_3(x_2), r_y)$

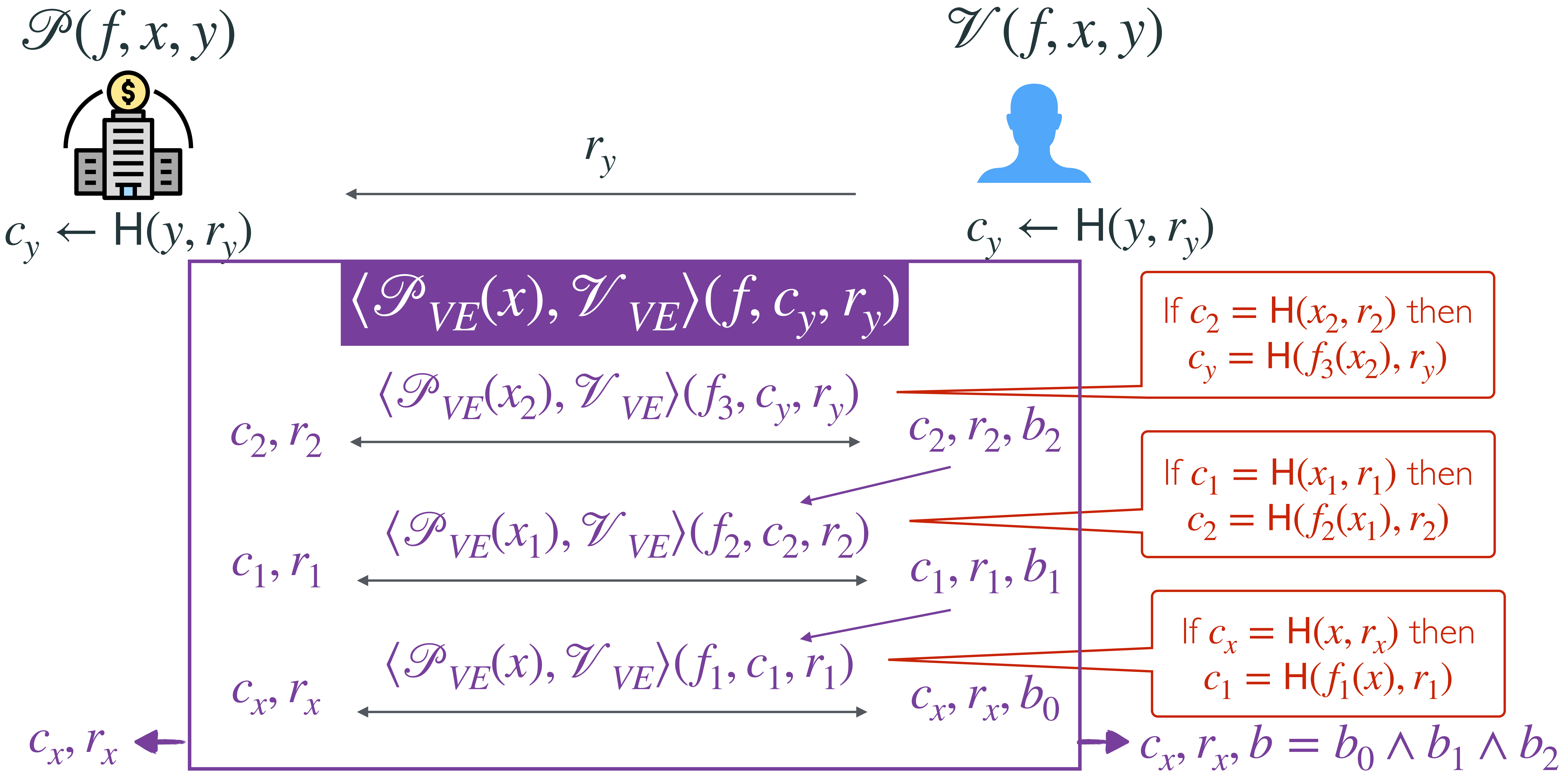
If $c_1 = \mathsf{H}(x_1, r_1)$ then $c_2 = \mathsf{H}(f_2(x_1), r_2)$

If $c_x = \mathsf{H}(x, r_x)$ then $c_1 = \mathsf{H}(f_1(x), r_1)$

Sequential composition of VEs

VE₁ for $z = f_1(x)$ + **VE₂** for $y = f_2(z, w)$ → **VE** for $y = f(x, w) = f_2(f_1(x), w)$





$\mathcal{P}(f, x, y)$

$\mathcal{V}(f, x, y)$

r_y

$c_y \leftarrow \mathsf{H}(y, r_y)$

$c_y \leftarrow \mathsf{H}(y, r_y)$

$\langle \mathcal{P}_{VE}(x), \mathcal{V}_{VE} \rangle (f, c_y, r_y)$

$\langle \mathcal{P}_{VE}(x_2), \mathcal{V}_{VE} \rangle (f_3, c_y, r_y)$

c_2, r_2 ⟷ c_2, r_2, b_2

If $c_2 = \mathsf{H}(x_2, r_2)$ then $c_y = \mathsf{H}(f_3(x_2), r_y)$

$\langle \mathcal{P}_{VE}(x_1), \mathcal{V}_{VE} \rangle (f_2, c_2, r_2)$

c_1, r_1 ⟷ c_1, r_1, b_1

If $c_1 = \mathsf{H}(x_1, r_1)$ then $c_2 = \mathsf{H}(f_2(x_1), r_2)$

$\langle \mathcal{P}_{VE}(x), \mathcal{V}_{VE} \rangle (f_1, c_1, r_1)$

c_x, r_x ⟷ c_x, r_x, b_0

If $c_x = \mathsf{H}(x, r_x)$ then $c_1 = \mathsf{H}(f_1(x), r_1)$

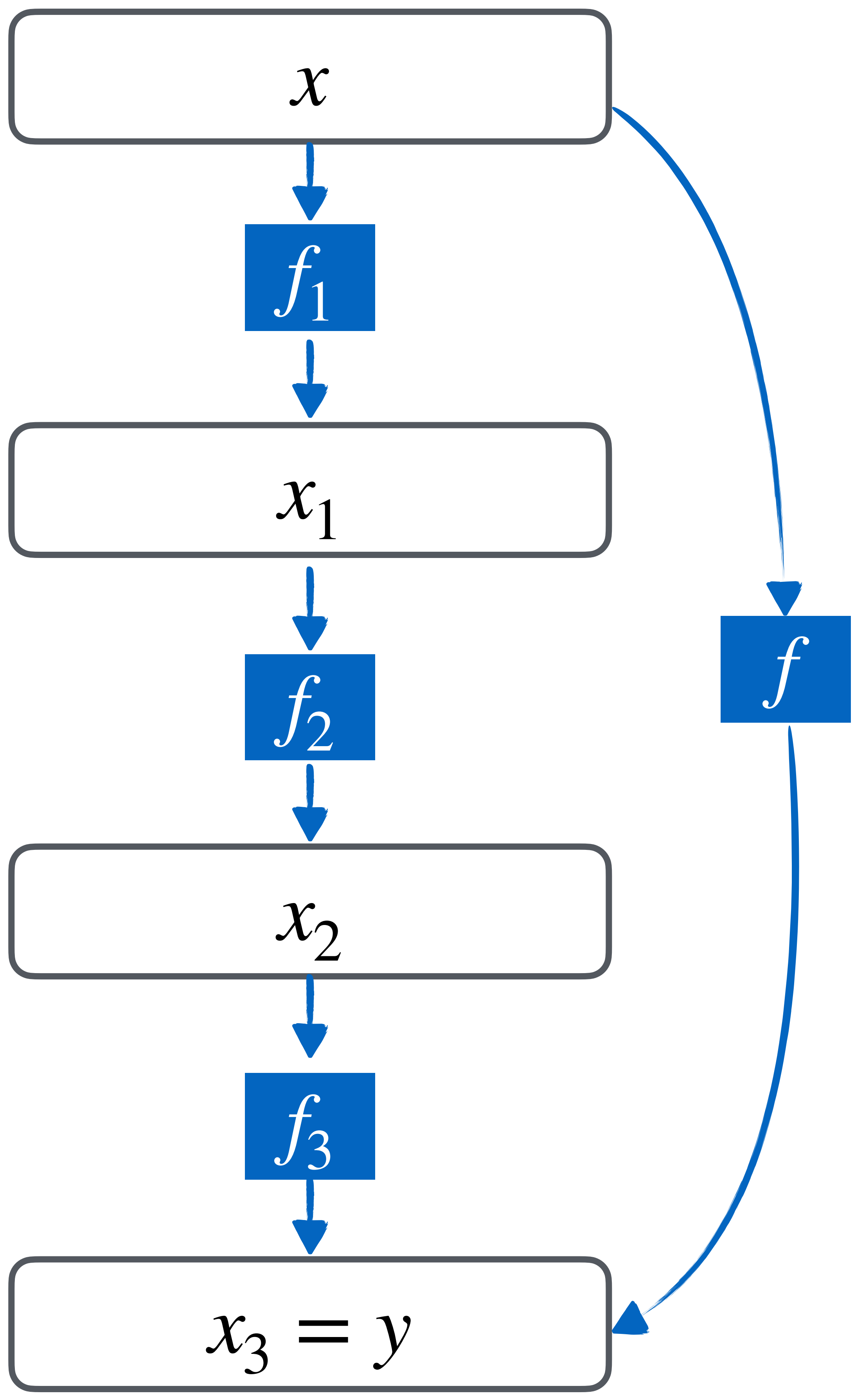
c_x, r_x

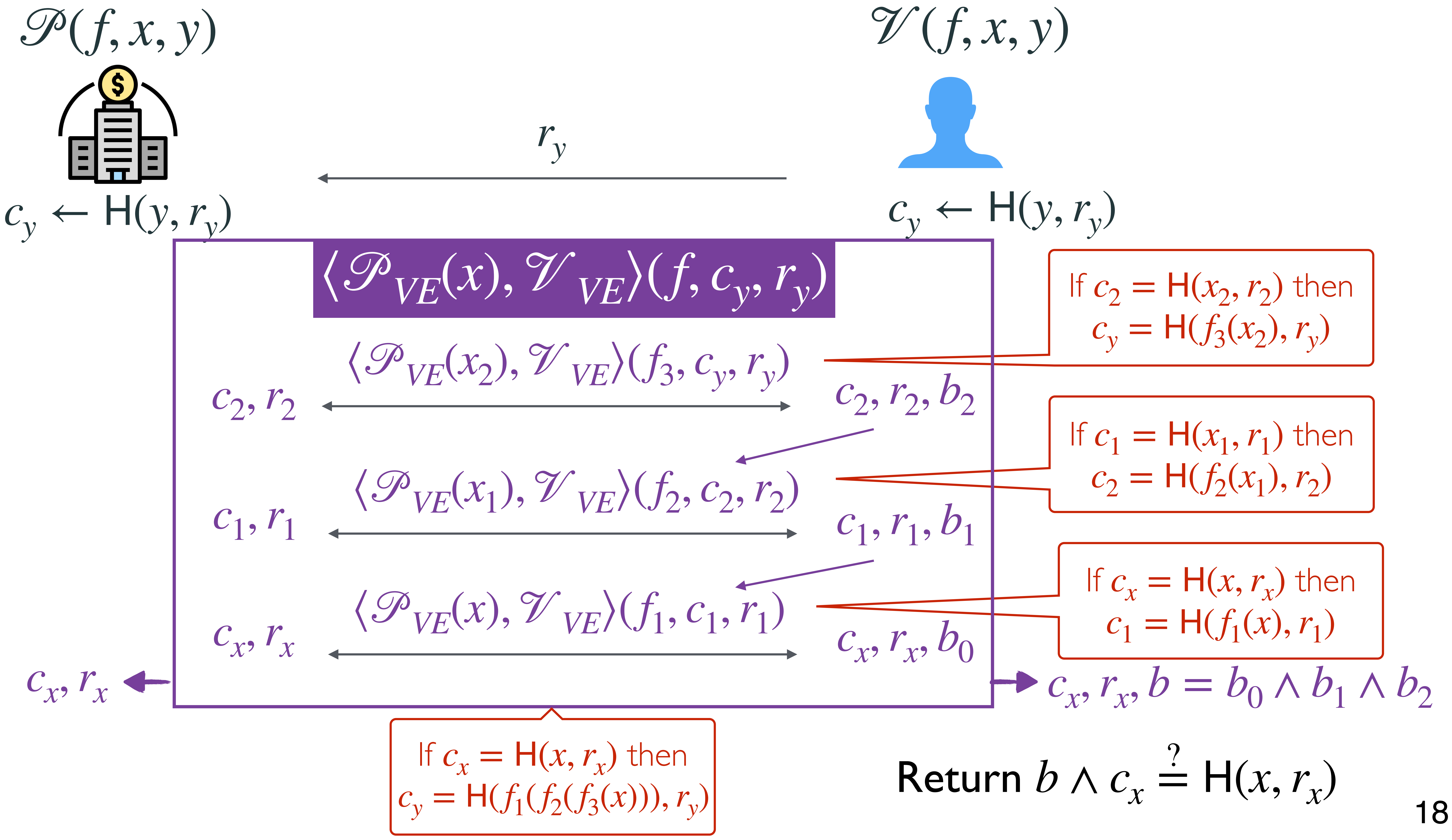
$c_x, r_x, b = b_0 \land b_1 \land b_2$

Return $b \land c_x \overset{?}{=} \mathsf{H}(x, r_x)$

Sequential composition of VEs

VE₁ for $z = f_1(x)$ + **VE₂** for $y = f_2(z, w)$ ➡ **VE** for $y = f(x, w) = f_2(f_1(x), w)$





If $c_2 = \mathsf{H}(x_2, r_2)$ then
$c_y = \mathsf{H}(f_3(x_2), r_y)$

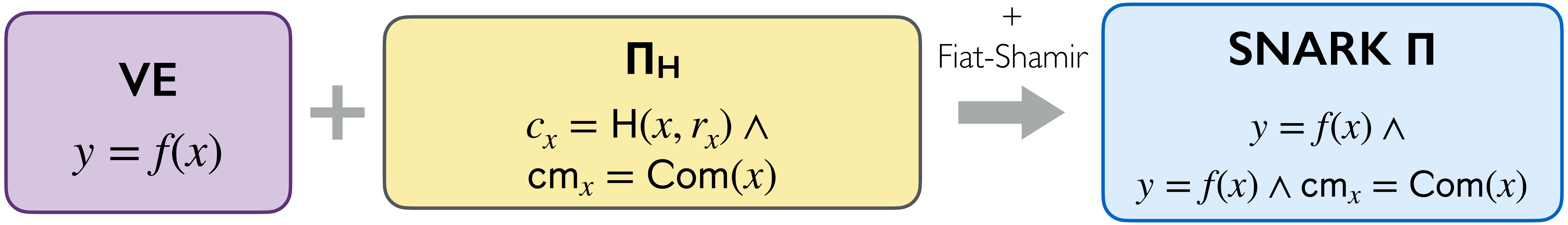
If $c_1 = \mathsf{H}(x_1, r_1)$ then
$c_2 = \mathsf{H}(f_2(x_1), r_2)$

If $c_x = \mathsf{H}(x, r_x)$ then
$c_1 = \mathsf{H}(f_1(x), r_1)$

If $c_x = \mathsf{H}(x, r_x)$ then
$c_y = \mathsf{H}(f_1(f_2(f_3(x))), r_y)$

Return $b \wedge c_x \stackrel{?}{=} \mathsf{H}(x, r_x)$

From VE-based IP to AoK [vSQL]



VE

$y = f(x)$

+

Π_H

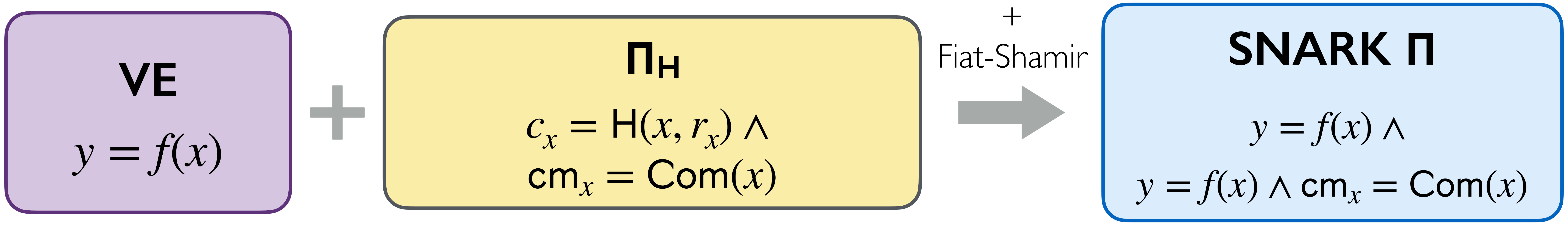
$c_x = \mathsf{H}(x, r_x) \wedge$
$\mathsf{cm}_x = \mathsf{Com}(x)$

+
Fiat-Shamir

SNARK Π

$y = f(x) \wedge$
$y = f(x) \wedge \mathsf{cm}_x = \mathsf{Com}(x)$

From VE-based IP to AoK [vSQL]



VE

$y = f(x)$

+

Π_H

$c_x = \mathsf{H}(x, r_x) \wedge$

$\mathsf{cm}_x = \mathsf{Com}(x)$

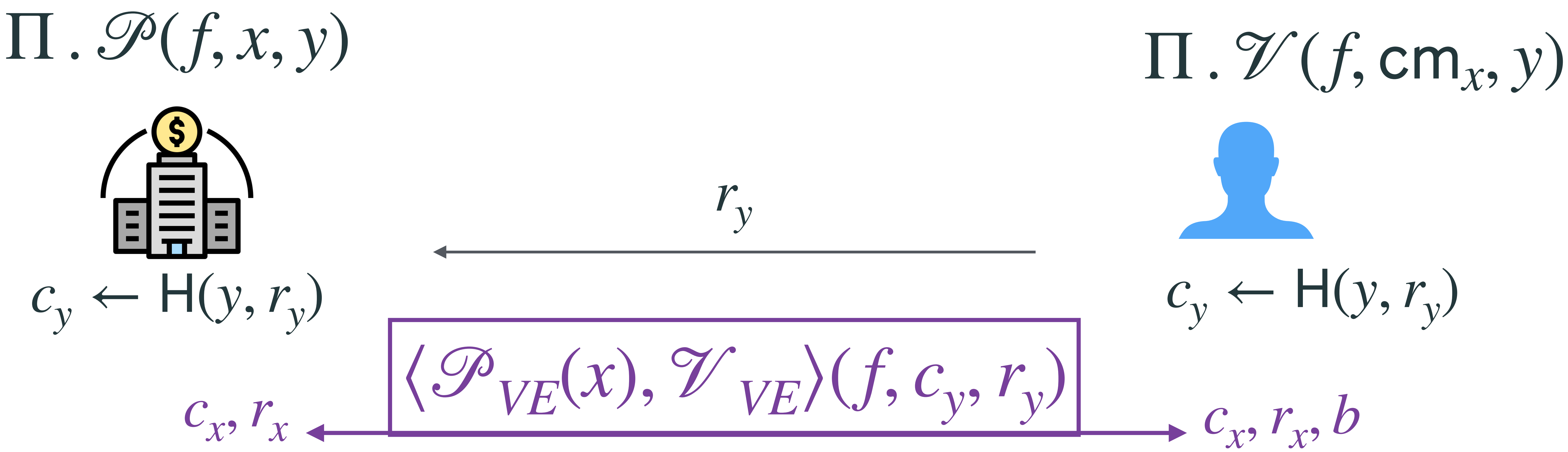
+ Fiat-Shamir

SNARK Π

$y = f(x) \wedge$

$y = f(x) \wedge \mathsf{cm}_x = \mathsf{Com}(x)$

VE-based IP



$\Pi.\mathscr{P}(f, x, y)$

$\Pi.\mathscr{V}(f, \mathsf{cm}_x, y)$

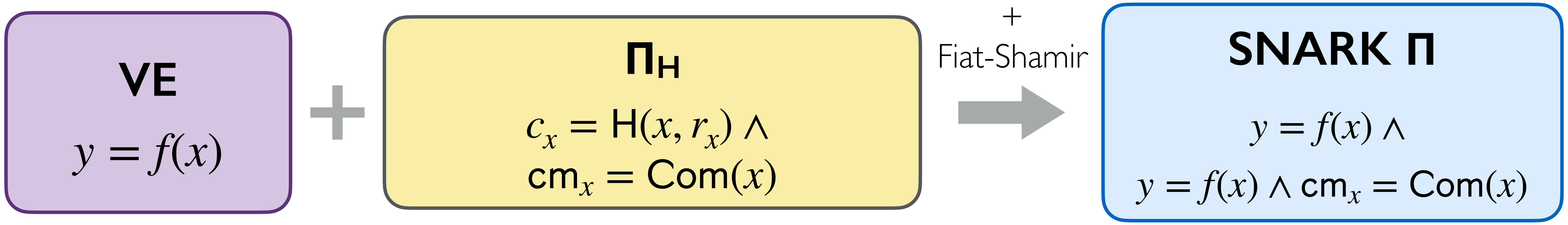
$$\xleftarrow{\quad r_y \quad}$$

$c_y \leftarrow \mathsf{H}(y, r_y)$

$c_y \leftarrow \mathsf{H}(y, r_y)$

$$c_x, r_x \longleftarrow \boxed{\langle \mathscr{P}_{VE}(x), \mathscr{V}_{VE} \rangle (f, c_y, r_y)} \longrightarrow c_x, r_x, b$$

From VE-based IP to AoK [vSQL]



VE

$y = f(x)$

+

Π_H

$c_x = \mathsf{H}(x, r_x) \wedge$

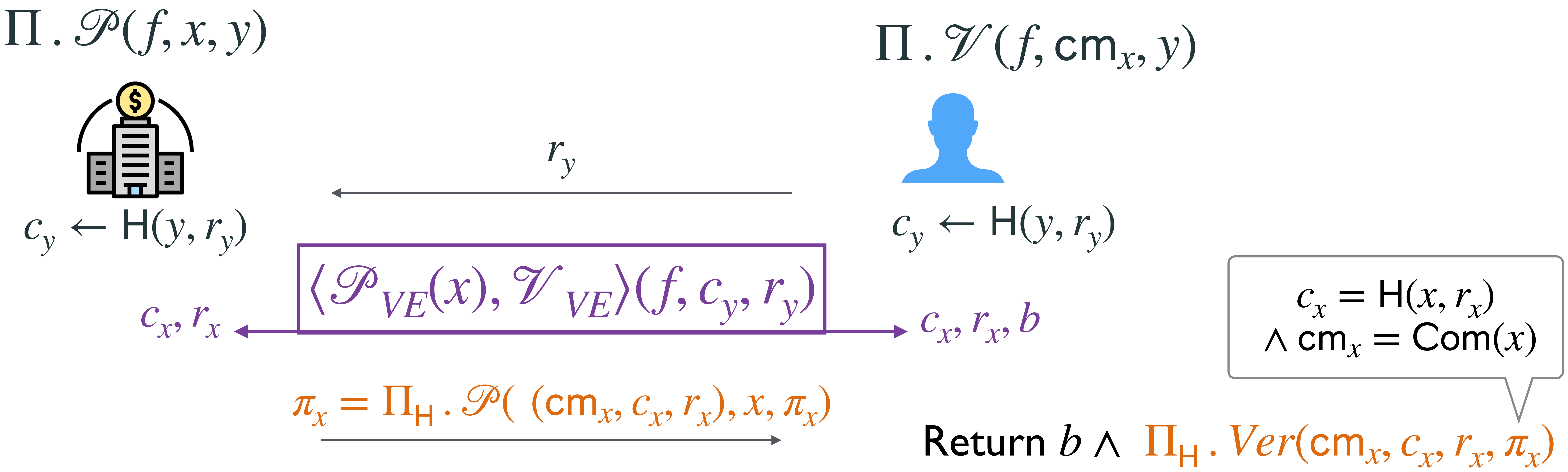
$\mathsf{cm}_x = \mathsf{Com}(x)$

\+ Fiat-Shamir

SNARK Π

$y = f(x) \wedge$

$y = f(x) \wedge \mathsf{cm}_x = \mathsf{Com}(x)$



$\Pi . \mathscr{P}(f, x, y)$

$\Pi . \mathscr{V}(f, \mathsf{cm}_x, y)$

VE-based IP

r_y

$c_y \leftarrow \mathsf{H}(y, r_y)$

$c_y \leftarrow \mathsf{H}(y, r_y)$

Prove that input fingerprint is correct w.r.t. cm_x

$\langle \mathscr{P}_{VE}(x), \mathscr{V}_{VE} \rangle (f, c_y, r_y)$

c_x, r_x

c_x, r_x, b

$c_x = \mathsf{H}(x, r_x)$
$\wedge\, \mathsf{cm}_x = \mathsf{Com}(x)$

$\pi_x = \Pi_\mathsf{H} . \mathscr{P}(\ (\mathsf{cm}_x, c_x, r_x), x, \pi_x)$

Return $b \wedge \ \Pi_\mathsf{H} . Ver(\mathsf{cm}_x, c_x, r_x, \pi_x)$

From VE-based IP to AoK [vSQL]

VE

$y = f(x)$

\+

Π

Π_{H}

$c_x = \mathsf{H}(x, r_x) \wedge$

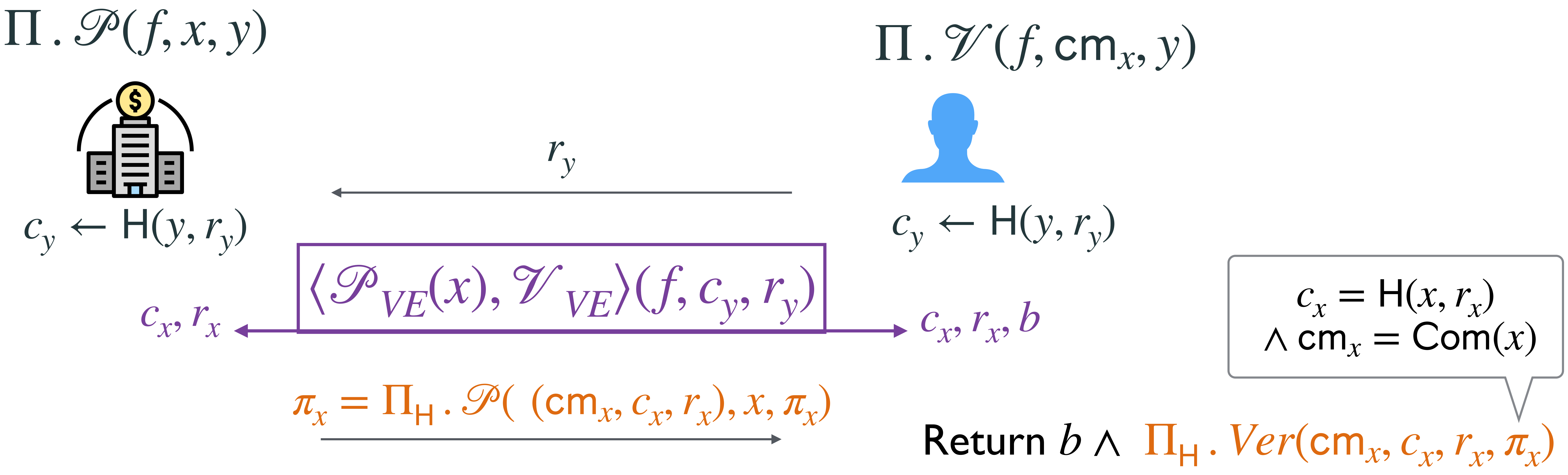
$\mathsf{cm}_x = \mathsf{Com}(x)$

\+ Fiat-Shamir →

SNARK Π

$y = f(x) \wedge$

$y = f(x) \wedge \mathsf{cm}_x = \mathsf{Com}(x)$



$\Pi\,.\,\mathscr{P}(f, x, y)$

$\Pi\,.\,\mathscr{V}(f, \mathsf{cm}_x, y)$

VE-based IP

r_y

$c_y \leftarrow \mathsf{H}(y, r_y)$

$c_y \leftarrow \mathsf{H}(y, r_y)$

$\langle \mathscr{P}_{VE}(x), \mathscr{V}_{VE} \rangle (f, c_y, r_y)$

c_x, r_x

c_x, r_x, b

Prove that input fingerprint is correct w.r.t. cm_x

$\pi_x = \Pi_{\mathsf{H}}\,.\,\mathscr{P}(\,(\mathsf{cm}_x, c_x, r_x), x, \pi_x)$

$c_x = \mathsf{H}(x, r_x)$

$\wedge\ \mathsf{cm}_x = \mathsf{Com}(x)$

Return $b \wedge \Pi_{\mathsf{H}}\,.\,Ver(\mathsf{cm}_x, c_x, r_x, \pi_x)$

When $\mathsf{H}(x, r_x) = \tilde{x}(r_x)$, Π_{H} can be instantiated with a multilinear polynomial commitment

From VE-based IP to AoK [vSQL]

VE

$y = f(x)$

+

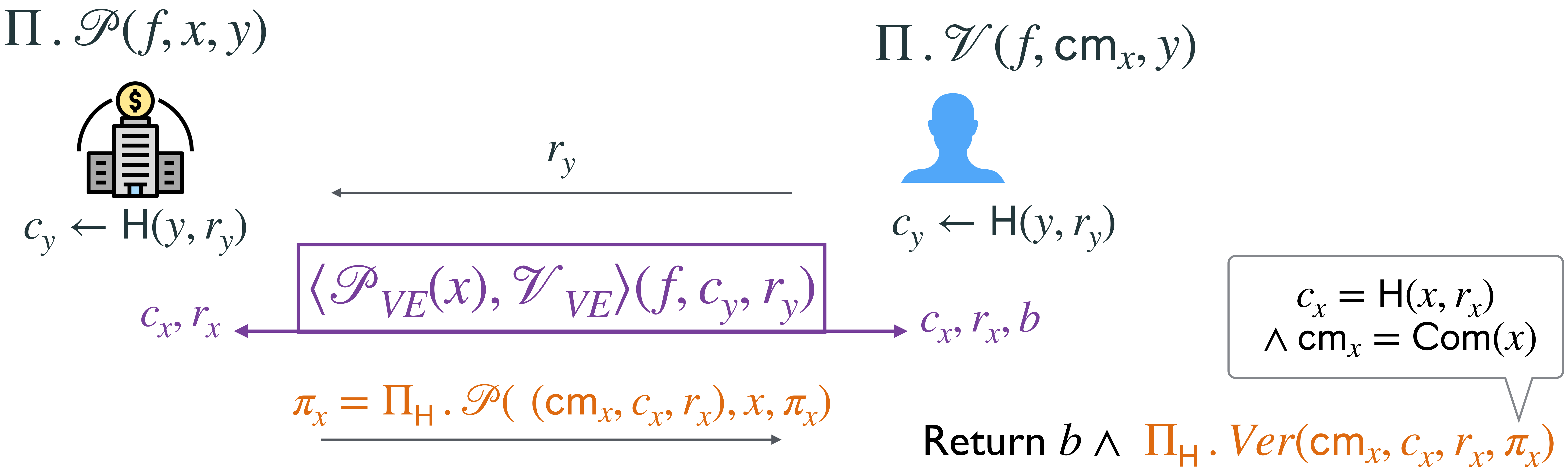
Π_H

$c_x = \mathsf{H}(x, r_x) \wedge$
$\mathsf{cm}_x = \mathsf{Com}(x)$

+ Fiat-Shamir →

SNARK Π

$y = f(x) \wedge$
$y = f(x) \wedge \mathsf{cm}_x = \mathsf{Com}(x)$



$\Pi . \mathscr{P}(f, x, y)$

$\Pi . \mathscr{V}(f, \mathsf{cm}_x, y)$

VE-based IP

r_y

$c_y \leftarrow \mathsf{H}(y, r_y)$

$c_y \leftarrow \mathsf{H}(y, r_y)$

Prove that input fingerprint is correct w.r.t. cm_x

$\langle \mathscr{P}_{VE}(x), \mathscr{V}_{VE} \rangle (f, c_y, r_y)$

c_x, r_x

c_x, r_x, b

$\pi_x = \Pi_\mathsf{H} . \mathscr{P}(\ (\mathsf{cm}_x, c_x, r_x), x, \pi_x)$

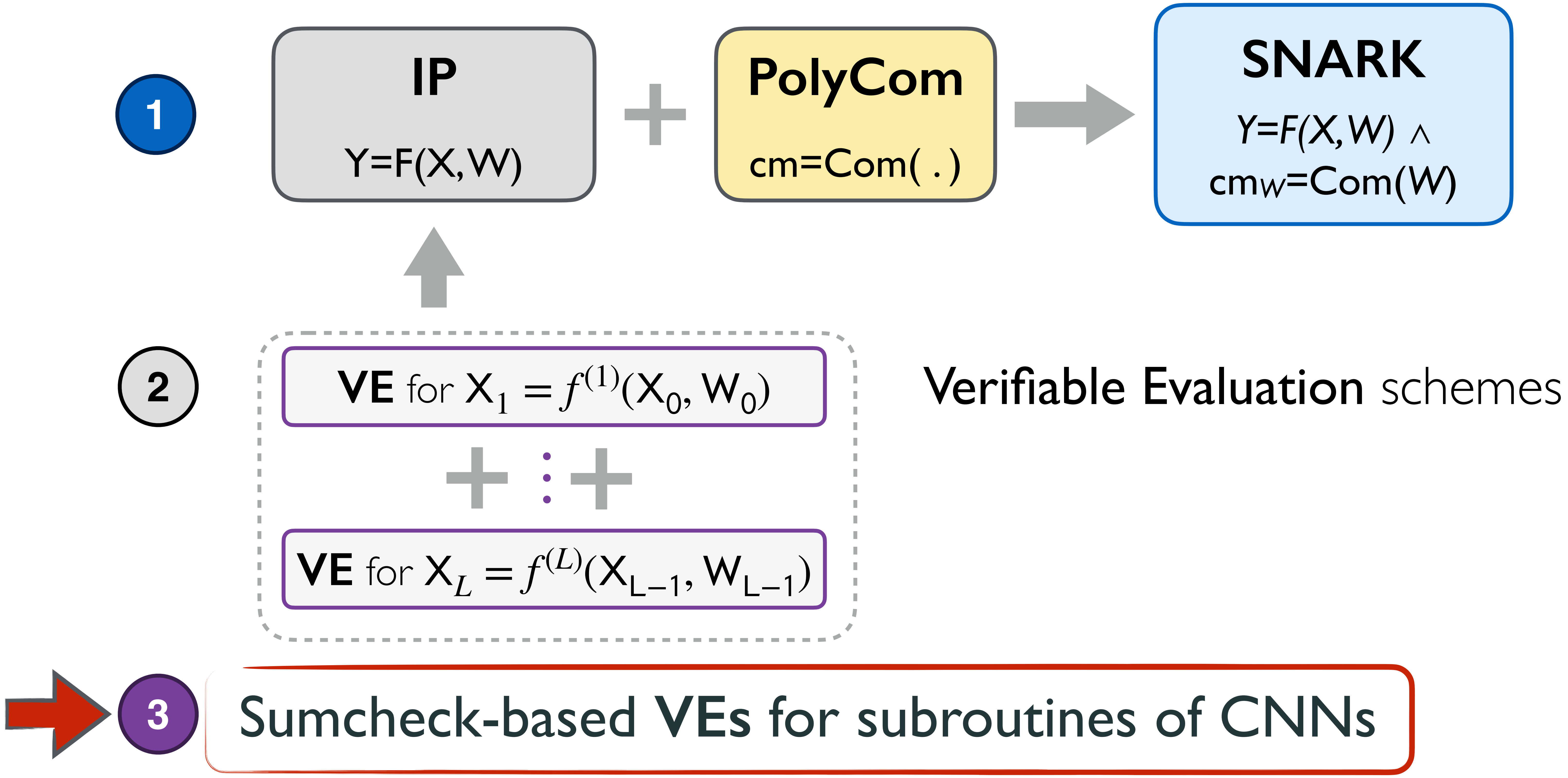
$c_x = \mathsf{H}(x, r_x)$
$\wedge \ \mathsf{cm}_x = \mathsf{Com}(x)$

Return $b \wedge \ \Pi_\mathsf{H} . Ver(\mathsf{cm}_x, c_x, r_x, \pi_x)$

When $\mathsf{H}(x, r_x) = \tilde{x}(r_x)$, **Π**$_\mathsf{H}$ can be instantiated with a multilinear polynomial commitment

To get ZK: hiding of Com **+** ZK of **Π**$_\mathsf{H}$ **+** "ZK of the IP" [Libra] (or "committed IP" [zk-vSQL, Hyrax])

Modular approach for CNNs



1

IP

$Y=F(X,W)$

\+

PolyCom

cm=Com(.)

SNARK

$Y=F(X,W) \wedge$ $cm_W=Com(W)$

2

VE for $X_1 = f^{(1)}(X_0, W_0)$

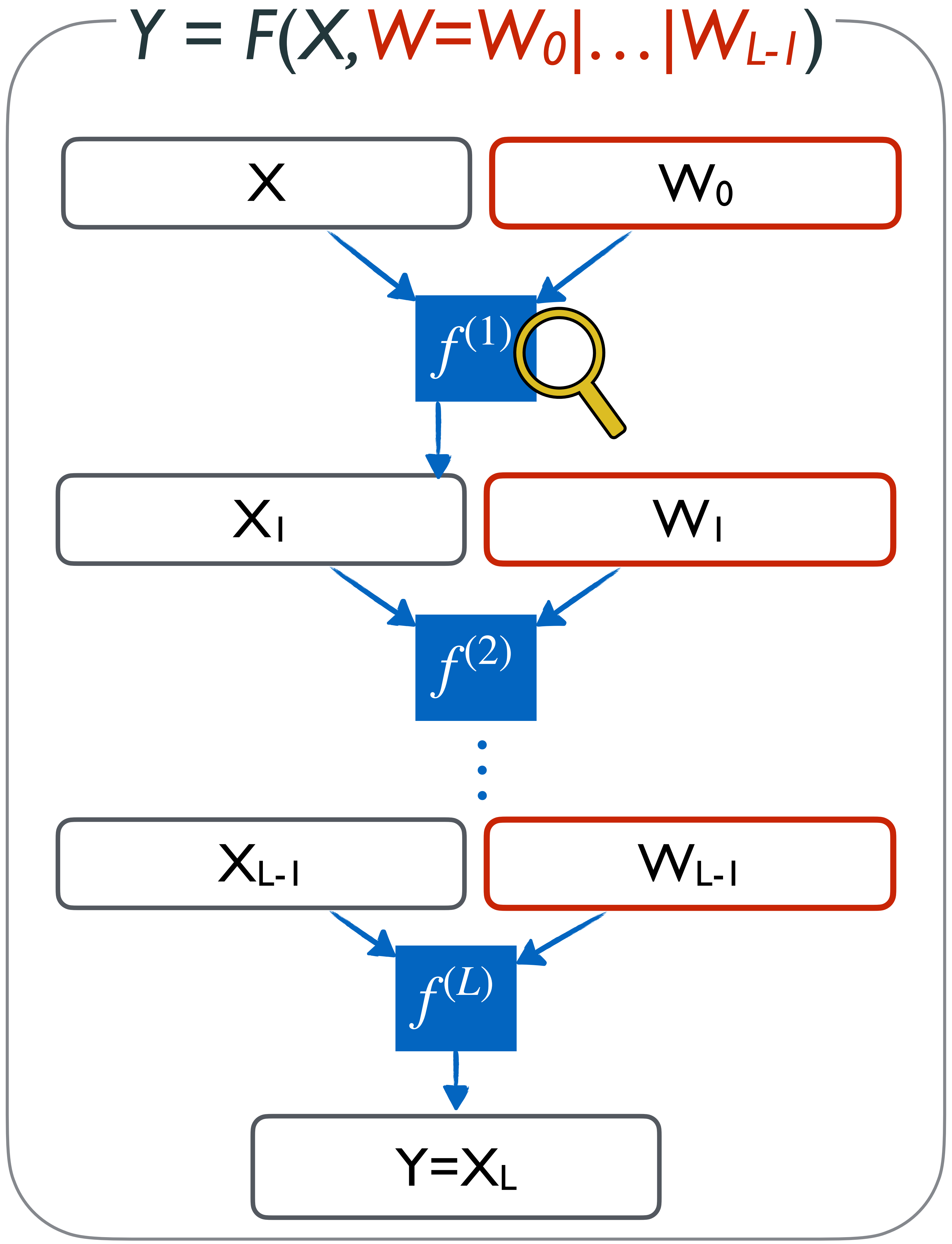
\+ ⋮ +

VE for $X_L = f^{(L)}(X_{L-1}, W_{L-1})$

Verifiable Evaluation schemes

3

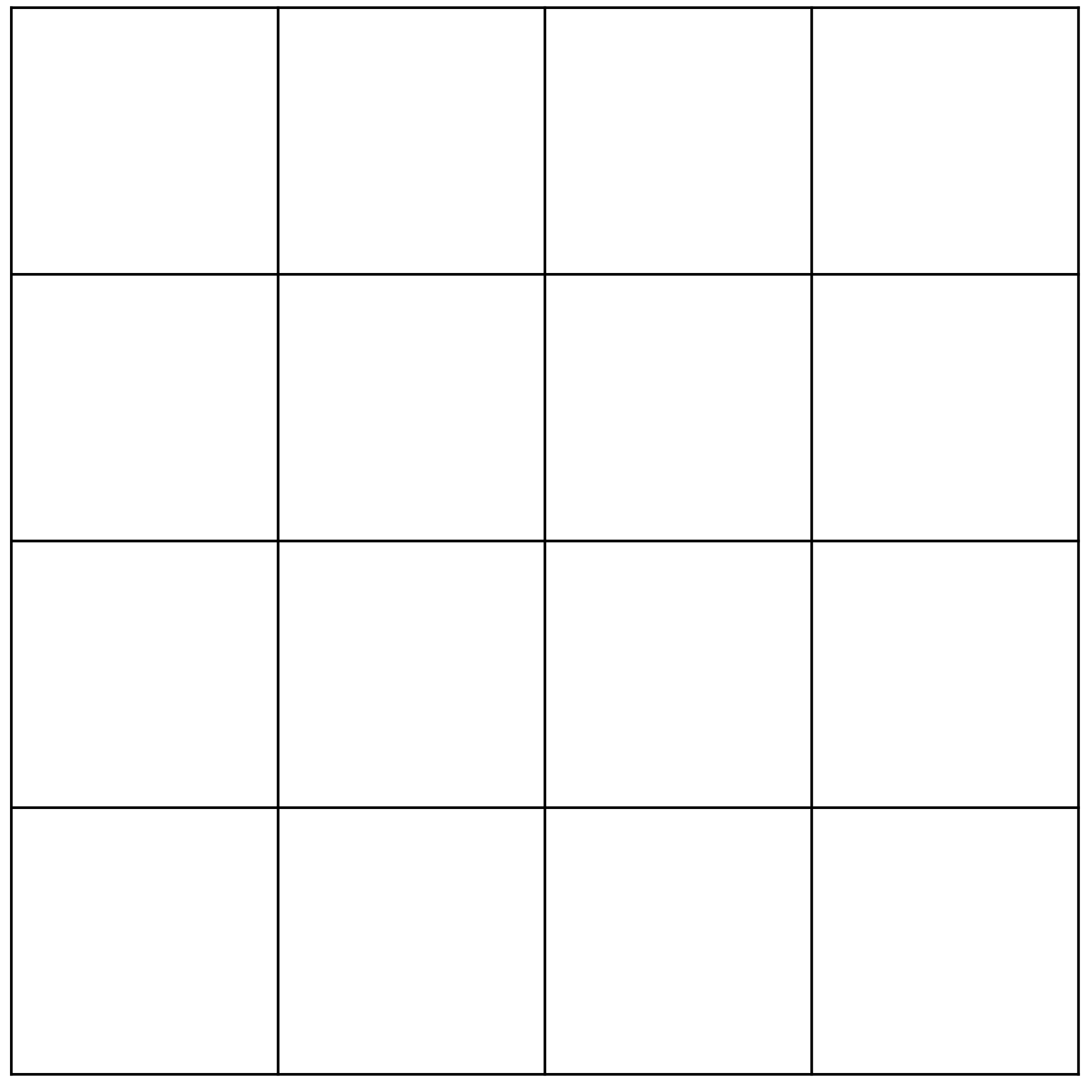
Sumcheck-based **VEs** for subroutines of CNNs



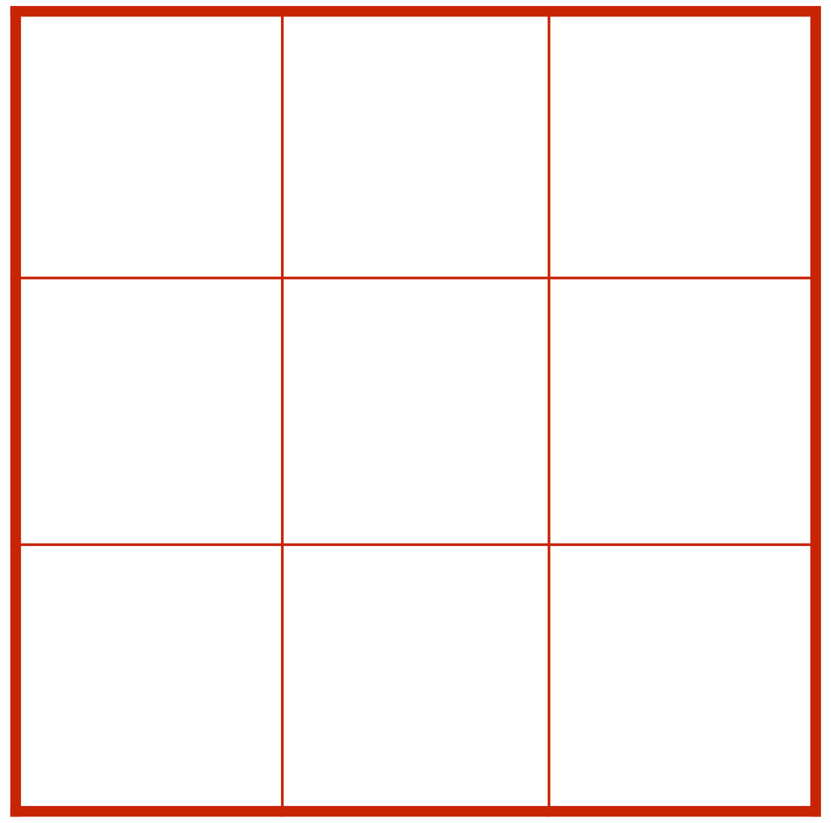
$Y = F(X, W=W_0|\ldots|W_{L-1})$

Convolution

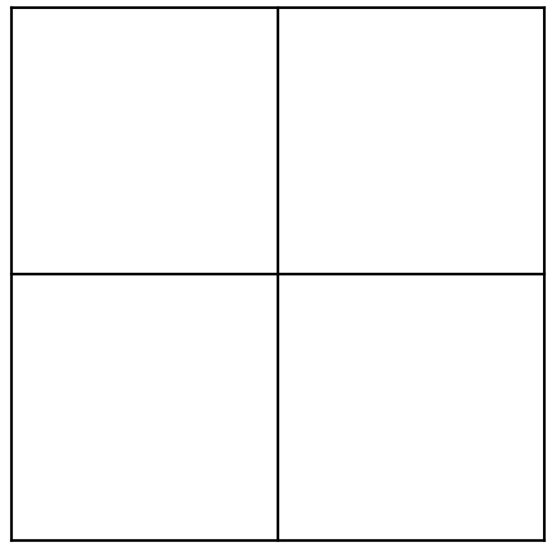
Input X
$(n \times n)$



Kernel W
$(m \times m)$



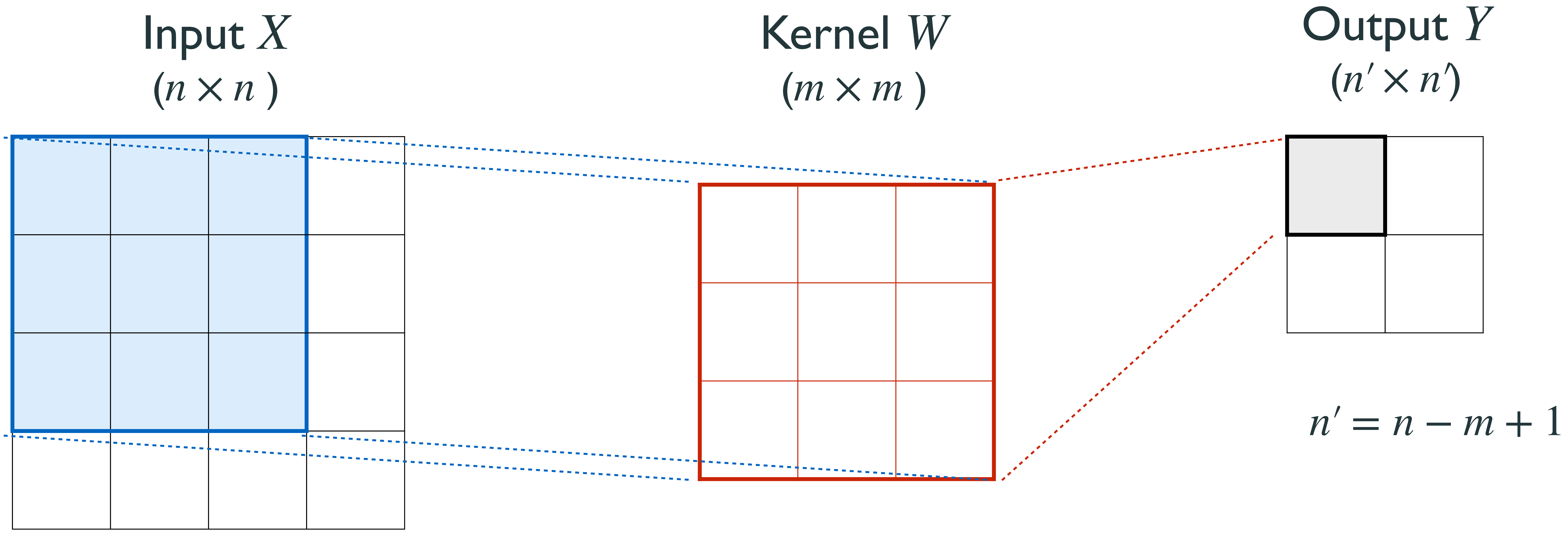
Output Y
$(n' \times n')$



$$n' = n - m + 1$$

$$\text{Output } Y[u, v] = \sum_{i,j=0}^{m-1} X[u+i, v+j] \cdot W[i, j]$$

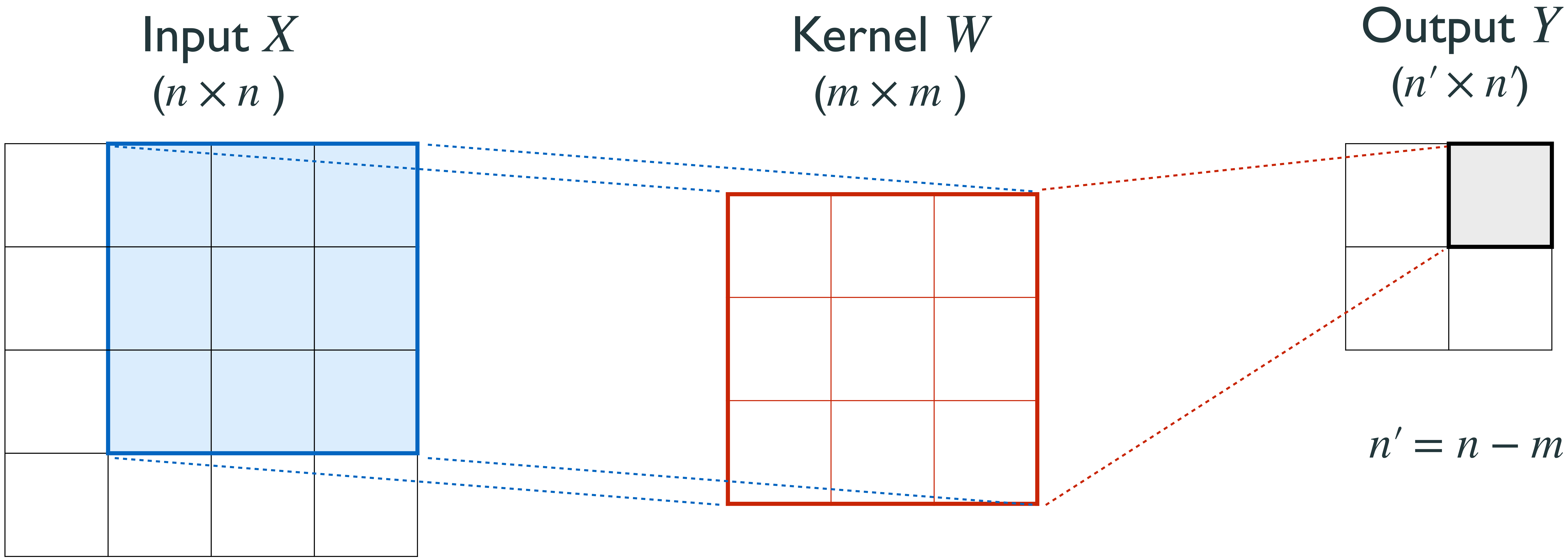
Convolution



$n' = n - m + 1$

$$\text{Output } Y[u, v] = \sum_{i,j=0}^{m-1} X[u+i, v+j] \cdot W[i, j]$$

Convolution



Input X
$(n \times n)$

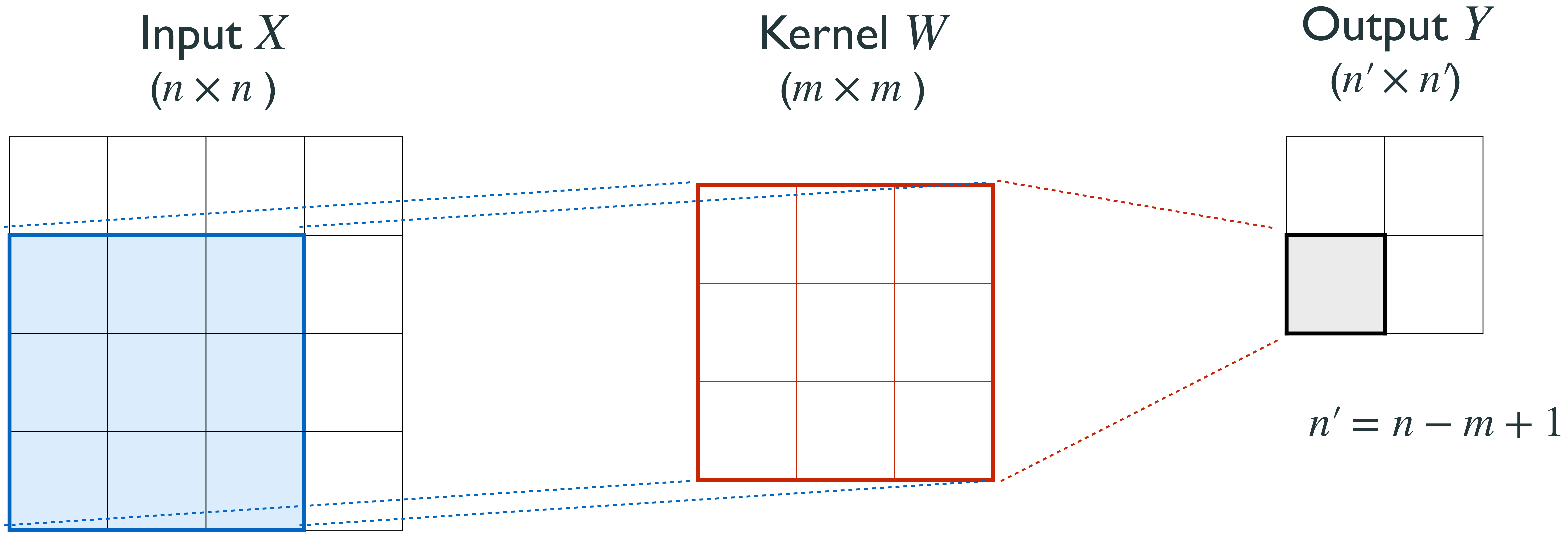
Kernel W
$(m \times m)$

Output Y
$(n' \times n')$

$$n' = n - m + 1$$

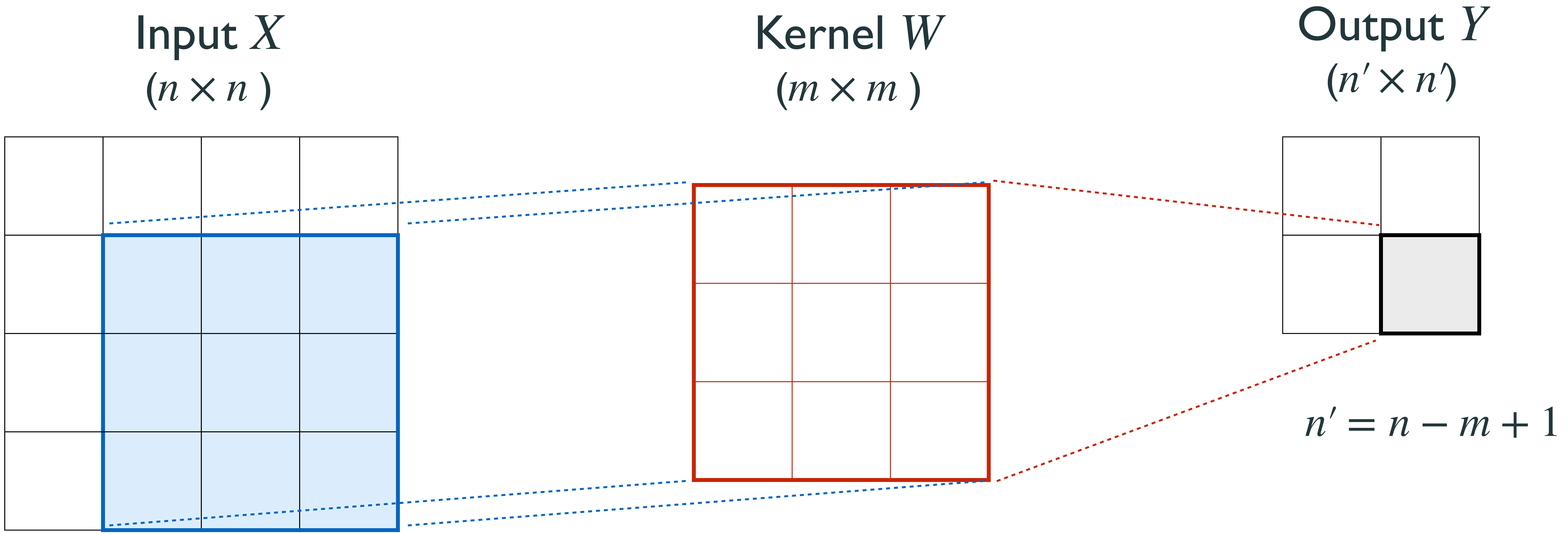
$$\text{Output } Y[u, v] = \sum_{i,j=0}^{m-1} X[u + i, v + j] \cdot W[i, j]$$

Convolution



$$\text{Output } Y[u,v] = \sum_{i,j=0}^{m-1} X[u+i, v+j] \cdot W[i,j]$$

Convolution



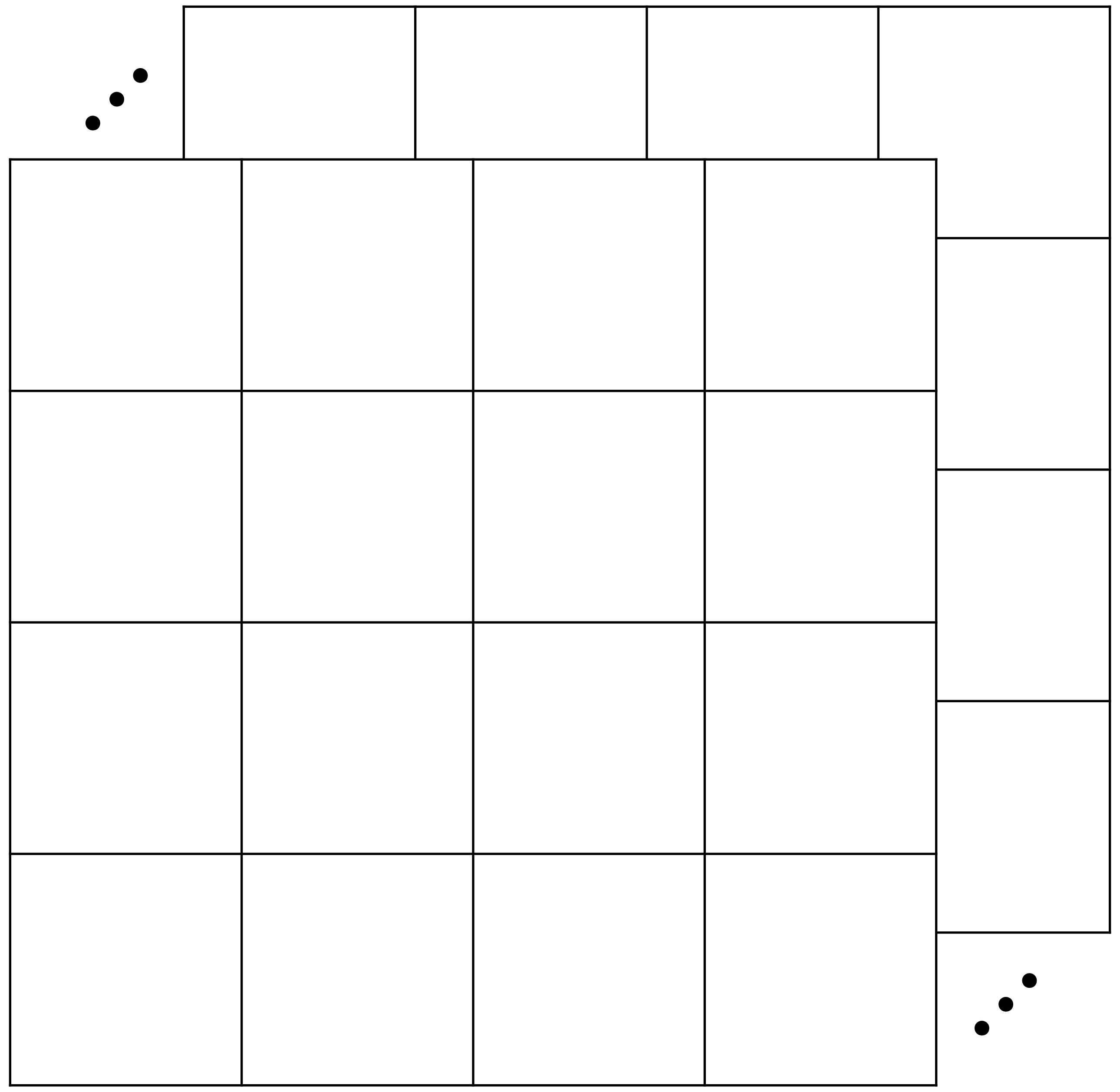
$n' = n - m + 1$

$$\text{Output } Y[u, v] = \sum_{i,j=0}^{m-1} X[u+i, v+j] \cdot W[i,j]$$

Multichannel Convolution of CNNs

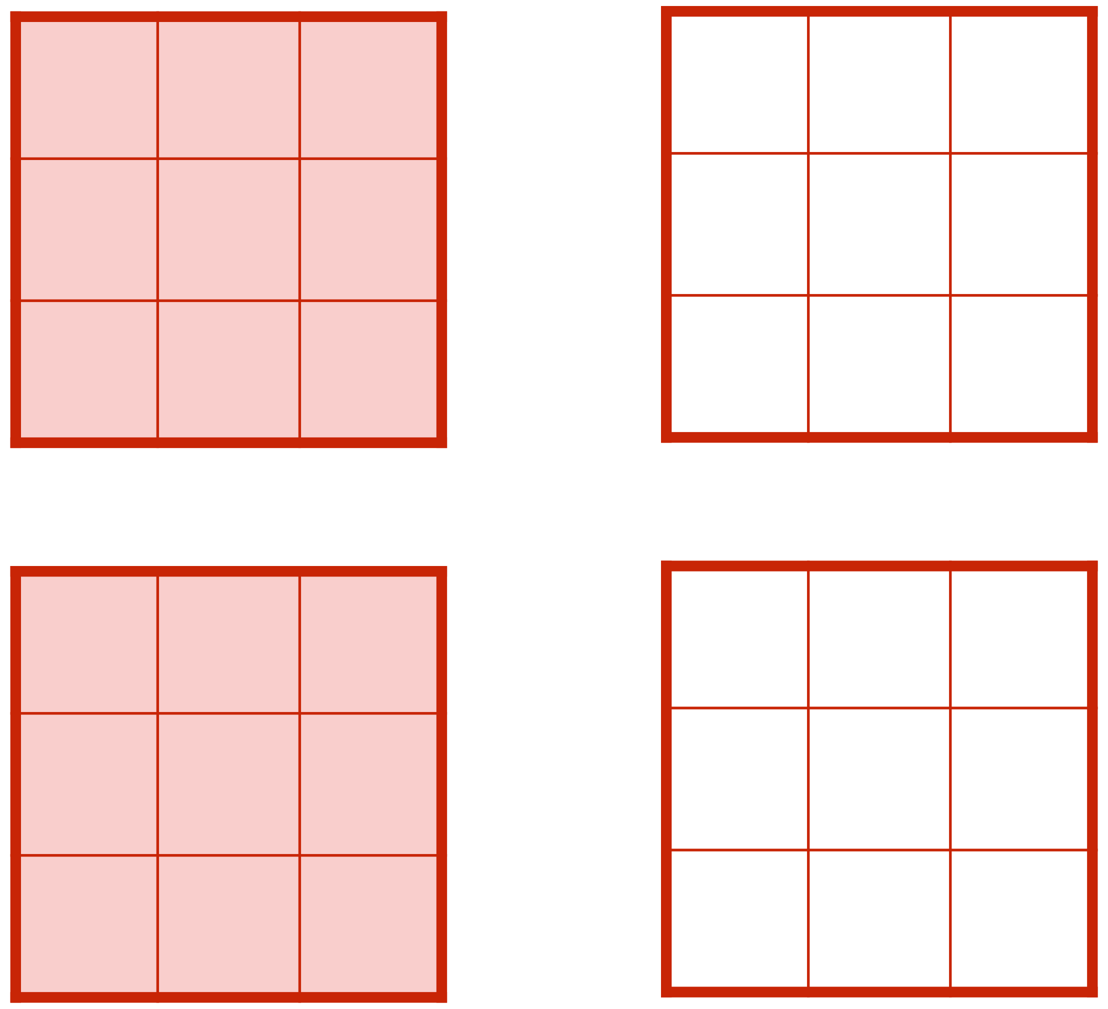
Input $\{X_\sigma^{(k)}\}_{\sigma=0}^{c^{(k)}-1}$

$(n^{(k)} \times n^{(k)})$



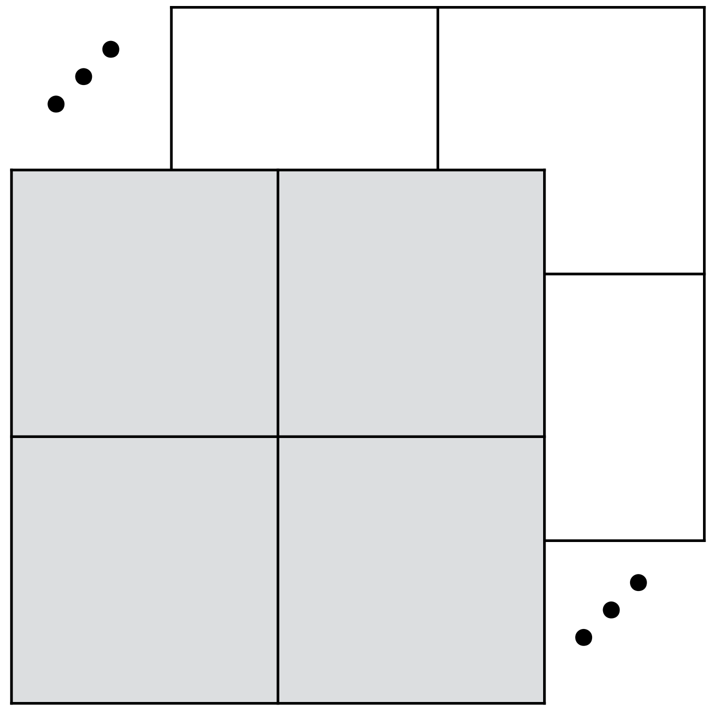
Kernel $\{W_{\sigma,\tau}^{(k)}\}_{\sigma,\tau=0}^{c^{(k)}-1,c^{(k+1)}-1}$

$(m^{(k)} \times m^{(k)})$



Output $\{X_\tau^{(k+1)}\}_{\tau=0}^{c^{(k+1)}-1}$

$(n^{(k+1)} \times n^{(k+1)})$



Output $X_\tau^{(k+1)}[u,v] = \sum_{\sigma=0}^{c^{(k)}-1} \sum_{i,j=0}^{m^{(k)}-1} X_\sigma^{(k)}[u+i, v+j] \cdot W_{\sigma,\tau}^{(k)}[i,j]$

Proving convolution

$Y_\tau[u,v] = \sum_{\sigma=0}^{c-1} \sum_{i,j=0}^{m-1} X_\sigma[u+i, v+j] \cdot W_{\sigma,\tau}[i,j]$ expensive as a circuit, $O(cd\,|Y_\tau| \cdot |W_{\sigma,\tau}|)$

Proving convolution

$$Y_\tau[u,v] = \sum_{\sigma=0}^{c-1} \sum_{i,j=0}^{m-1} X_\sigma[u+i, v+j] \cdot W_{\sigma,\tau}[i,j]$$ expensive as a circuit, $O(cd\,|\,Y_\tau\,|\cdot|\,W_{\sigma,\tau}\,|)$

Special-purpose techniques: convolution → structured matrix multiplication

Proving convolution

$$Y_\tau[u,v] = \sum_{\sigma=0}^{c-1} \sum_{i,j=0}^{m-1} X_\sigma[u+i, v+j] \cdot W_{\sigma,\tau}[i,j]$$ expensive as a circuit, $O(cd\,|\,Y_\tau\,| \cdot |\,W_{\sigma,\tau}\,|)$

Special-purpose techniques: convolution → structured matrix multiplication

zkCNN [LXZ21]

(2dim)Convolution → 1dim-convolution → poly mult → FFT → matrix multiplication

Proving convolution

$Y_\tau[u,v] = \sum_{\sigma=0}^{c-1} \sum_{i,j=0}^{m-1} X_\sigma[u+i, v+j] \cdot W_{\sigma,\tau}[i,j]$ expensive as a circuit, $O(cd\,|Y_\tau| \cdot |W_{\sigma,\tau}|)$

Special-purpose techniques: convolution → structured matrix multiplication

zkCNN [LXZ21]

(2dim)Convolution → 1dim-convolution → poly mult → FFT → matrix multiplication

[BFGRS23]

Convolution → matrix multiplication with reshaped X, W

Reshaping convolution

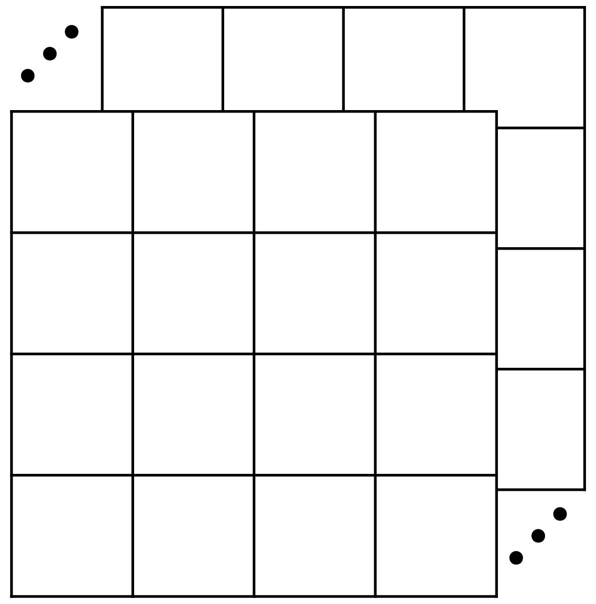
Let $X = \begin{bmatrix} x_0 & x_1 & x_2 \\ x_3 & x_4 & x_5 \\ x_6 & x_7 & x_8 \end{bmatrix}$ and $W = \begin{bmatrix} w_0 & w_1 \\ w_2 & w_3 \end{bmatrix}$. Compact convolution as

$$\text{vec}(Y) = \begin{pmatrix} w_0x_0 + w_1x_1 + w_2x_3 + w_3x_4 \\ w_0x_1 + w_1x_2 + w_2x_4 + w_3x_5 \\ w_0x_3 + w_1x_4 + w_2x_6 + w_3x_7 \\ w_0x_4 + w_1x_5 + w_2x_7 + w_3x_8 \end{pmatrix} = \begin{bmatrix} x_0 & x_1 & x_3 & x_4 \\ x_1 & x_2 & x_4 & x_5 \\ x_3 & x_4 & x_6 & x_7 \\ x_4 & x_5 & x_7 & x_8 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \hat{X} \cdot \hat{W}$$

Reshaping multi-channel convolution

Inputs X_σ

$(n \times n\ )$

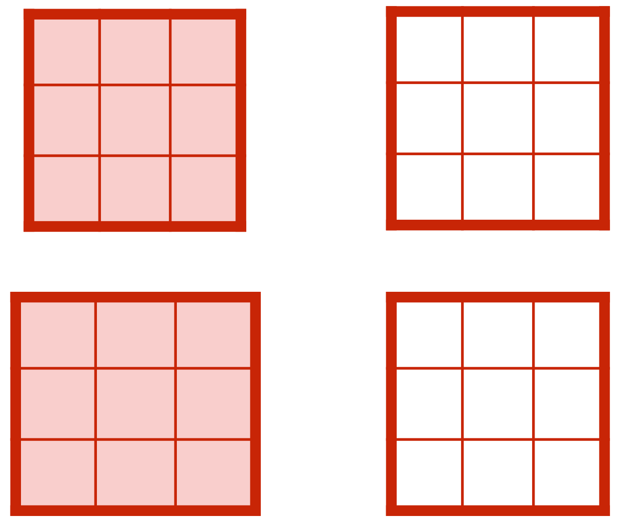


reshaped $\hat{X}_\sigma$

$((n')^2 \times m^2\ )$

Kernel $W_{\sigma,\tau}$

$(m \times m\ )$

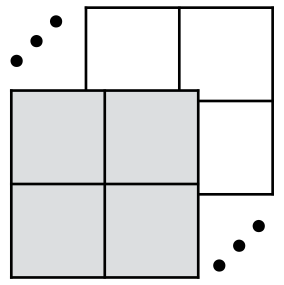


reshaped $\hat{W}_{\sigma,\tau}$

$(m^2\ )$

Outputs Y_τ

$(n' \times n')$



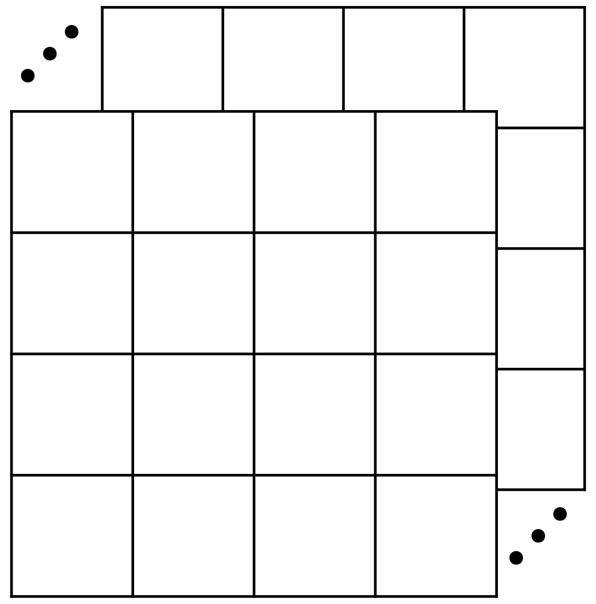
$\text{vec}(Y_\tau) = \hat{X}_\sigma \cdot \hat{W}_{\sigma,\tau}$

$((n')^2\ )$

Reshaping multi-channel convolution

Inputs X_σ

$(n \times n)$

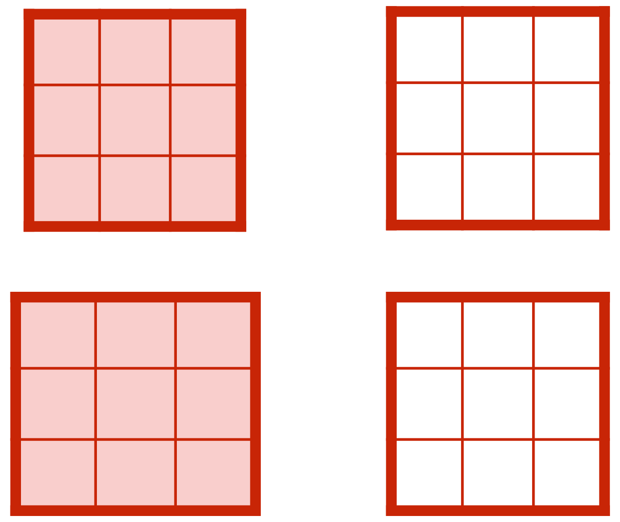


reshaped $\hat{X}_\sigma$

$((n')^2 \times m^2)$

Kernel $W_{\sigma,\tau}$

$(m \times m)$

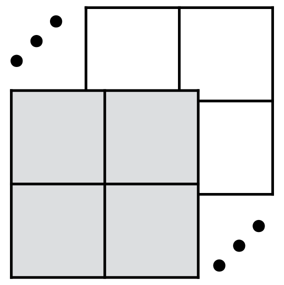


reshaped $\hat{W}_{\sigma,\tau}$

(m^2)

Outputs Y_τ

$(n' \times n')$



$\mathrm{vec}(Y_\tau) = \hat{X}_\sigma \cdot \hat{W}_{\sigma,\tau}$

$((n')^2)$

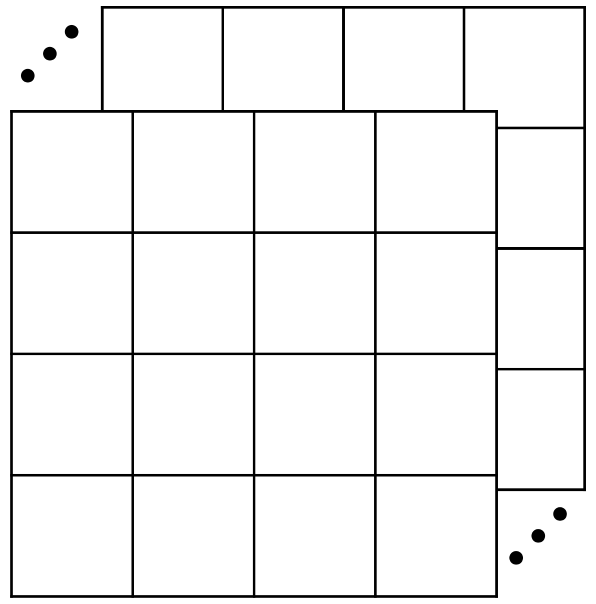
For multiple channels

$$Y = [Y_1 \mid \cdots \mid Y_d] = \sum_\sigma \hat{X}_\sigma \cdot [\hat{W}_{\sigma,1} \mid \cdots \mid \hat{W}_{\sigma,d}] = \sum_\sigma \hat{X}_\sigma \cdot \hat{W}_\sigma$$

sum of matrix multiplications

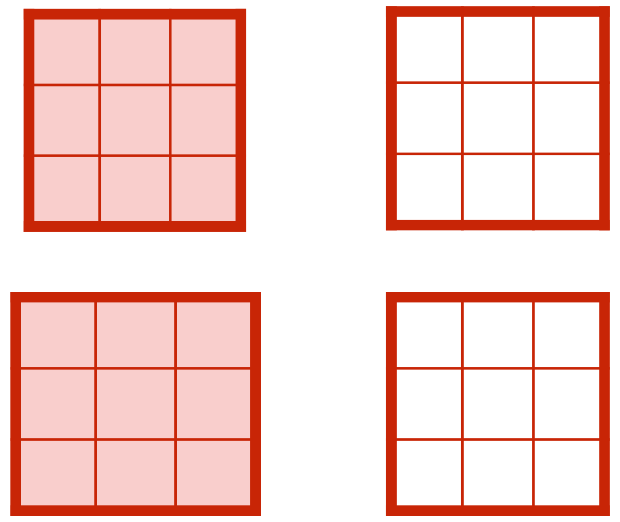
Reshaping multi-channel convolution

Inputs X_σ
$(n \times n)$



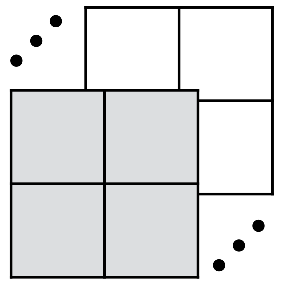
reshaped $\hat{X}_\sigma$
$((n')^2 \times m^2)$

Kernel $W_{\sigma,\tau}$
$(m \times m)$



reshaped $\hat{W}_{\sigma,\tau}$
(m^2)

Outputs Y_τ
$(n' \times n')$



$\mathrm{vec}(Y_\tau) = \hat{X}_\sigma \cdot \hat{W}_{\sigma,\tau}$
$((n')^2)$

For multiple channels

$$Y = [Y_1 \,|\, \cdots \,|\, Y_d] = \sum_\sigma \hat{X}_\sigma \cdot [\hat{W}_{\sigma,1} \,|\, \cdots \,|\, \hat{W}_{\sigma,d}] = \sum_\sigma \hat{X}_\sigma \cdot \hat{W}_\sigma$$

sum of matrix multiplications

VE for convolution ← VE for (sum of) matrix multiplications

VE for matrix multiplication [Thaler13]

Prove $C = A \cdot B$ for $A, B, C \in \mathbb{F}^{n \times n}$. Using MLE: $\forall \vec{i}, \vec{k} \in \{0,1\}^{\log n} : \tilde{C}(\vec{i}, \vec{k}) = \sum_{\vec{j} \in \{0,1\}^{\log n}} \tilde{A}(\vec{i}, \vec{j}) \cdot \tilde{B}(\vec{j}, \vec{k})$

VE efficiency: communication&verification O(log n), prover O(n^2), faster than computing $\boldsymbol{A} \cdot \boldsymbol{B}$ in O(n^3)

VE for matrix multiplication [Thaler13]

Prove $C = A \cdot B$ for $A, B, C \in \mathbb{F}^{n \times n}$. Using MLE: $\forall \vec{i}, \vec{k} \in \{0,1\}^{\log n} : \tilde{C}(\vec{i}, \vec{k}) = \sum_{\vec{j} \in \{0,1\}^{\log n}} \tilde{A}(\vec{i}, \vec{j}) \cdot \tilde{B}(\vec{j}, \vec{k})$

$\mathscr{P}^{mm}(A, B, C)$

$\mathscr{V}^{mm}(A, B, C)$

VE efficiency: communication&verification O(log n), prover O(n²), faster than computing $A \cdot B$ in O(n³)

VE for matrix multiplication [Thaler13]

Prove $C = A \cdot B$ for $A, B, C \in \mathbb{F}^{n \times n}$. Using MLE: $\forall \vec{i}, \vec{k} \in \{0,1\}^{\log n} : \tilde{C}(\vec{i}, \vec{k}) = \sum_{\vec{j} \in \{0,1\}^{\log n}} \tilde{A}(\vec{i}, \vec{j}) \cdot \tilde{B}(\vec{j}, \vec{k})$

$\mathscr{P}^{mm}(A, B, C)$

$c_C = \mathsf{H}(C, \vec{r}_C) = \tilde{C}(\vec{r}_1, \vec{r}_2)$

$\xleftarrow{\vec{r}_C = (\vec{r}_1, \vec{r}_2)}$

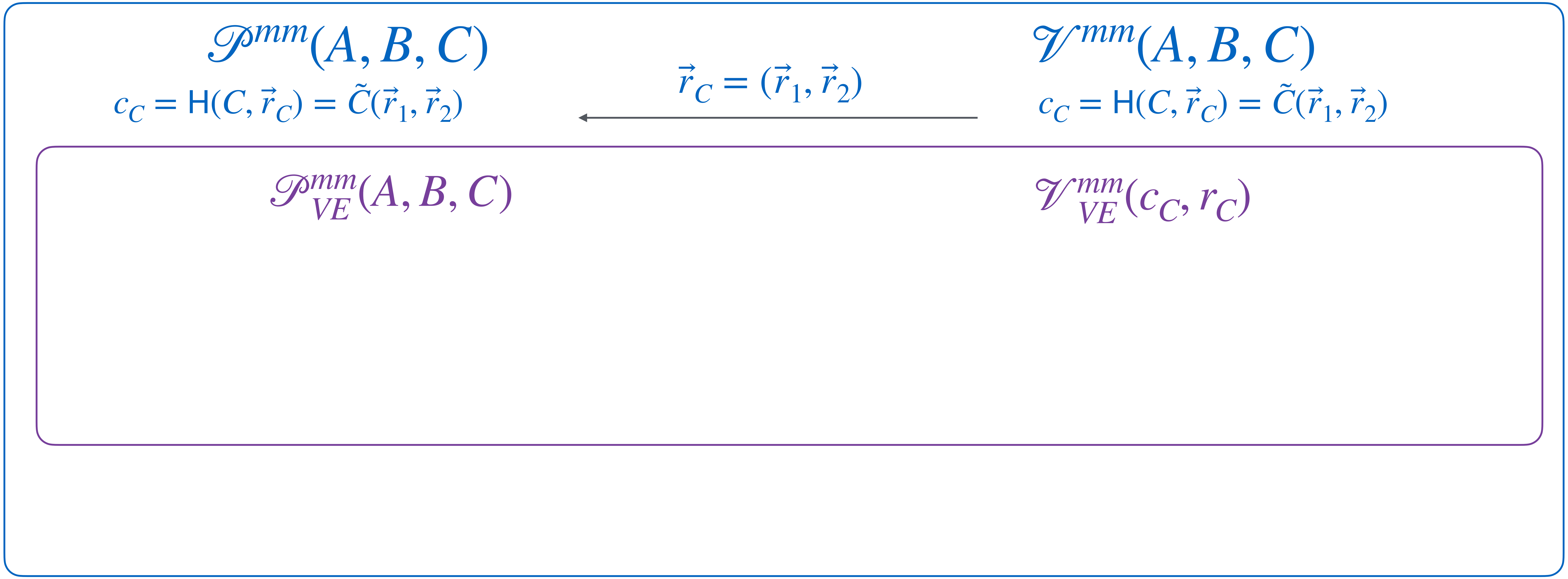
$\mathscr{V}^{mm}(A, B, C)$

$c_C = \mathsf{H}(C, \vec{r}_C) = \tilde{C}(\vec{r}_1, \vec{r}_2)$

VE efficiency: communication&verification O(log n), prover O(n^2), faster than computing $A \cdot B$ in O(n^3)

VE for matrix multiplication [Thaler13]

Prove $C = A \cdot B$ for $A, B, C \in \mathbb{F}^{n \times n}$. Using MLE: $\forall \vec{i}, \vec{k} \in \{0,1\}^{\log n} : \tilde{C}(\vec{i}, \vec{k}) = \sum_{\vec{j} \in \{0,1\}^{\log n}} \tilde{A}(\vec{i}, \vec{j}) \cdot \tilde{B}(\vec{j}, \vec{k})$



$\mathscr{P}^{mm}(A, B, C)$

$c_C = \mathsf{H}(C, \vec{r}_C) = \tilde{C}(\vec{r}_1, \vec{r}_2)$

$\vec{r}_C = (\vec{r}_1, \vec{r}_2)$

$\mathscr{V}^{mm}(A, B, C)$

$c_C = \mathsf{H}(C, \vec{r}_C) = \tilde{C}(\vec{r}_1, \vec{r}_2)$

$\mathscr{P}^{mm}_{VE}(A, B, C)$

$\mathscr{V}^{mm}_{VE}(c_C, r_C)$

VE efficiency: communication&verification O(log n), prover O(n^2), faster than computing $A \cdot B$ in O(n^3)

VE for matrix multiplication [Thaler13]

Prove $C = A \cdot B$ for $A, B, C \in \mathbb{F}^{n\times n}$. Using MLE: $\forall \vec{i}, \vec{k} \in \{0,1\}^{\log n} : \tilde{C}(\vec{i}, \vec{k}) = \sum_{\vec{j}\in\{0,1\}^{\log n}} \tilde{A}(\vec{i}, \vec{j}) \cdot \tilde{B}(\vec{j}, \vec{k})$

$\mathcal{P}^{mm}(A, B, C)$

$c_C = \mathsf{H}(C, \vec{r}_C) = \tilde{C}(\vec{r}_1, \vec{r}_2)$

$\mathcal{V}^{mm}(A, B, C)$

$c_C = \mathsf{H}(C, \vec{r}_C) = \tilde{C}(\vec{r}_1, \vec{r}_2)$

$$\xleftarrow{\vec{r}_C = (\vec{r}_1, \vec{r}_2)}$$

$\mathcal{P}^{mm}_{VE}(A, B, C)$

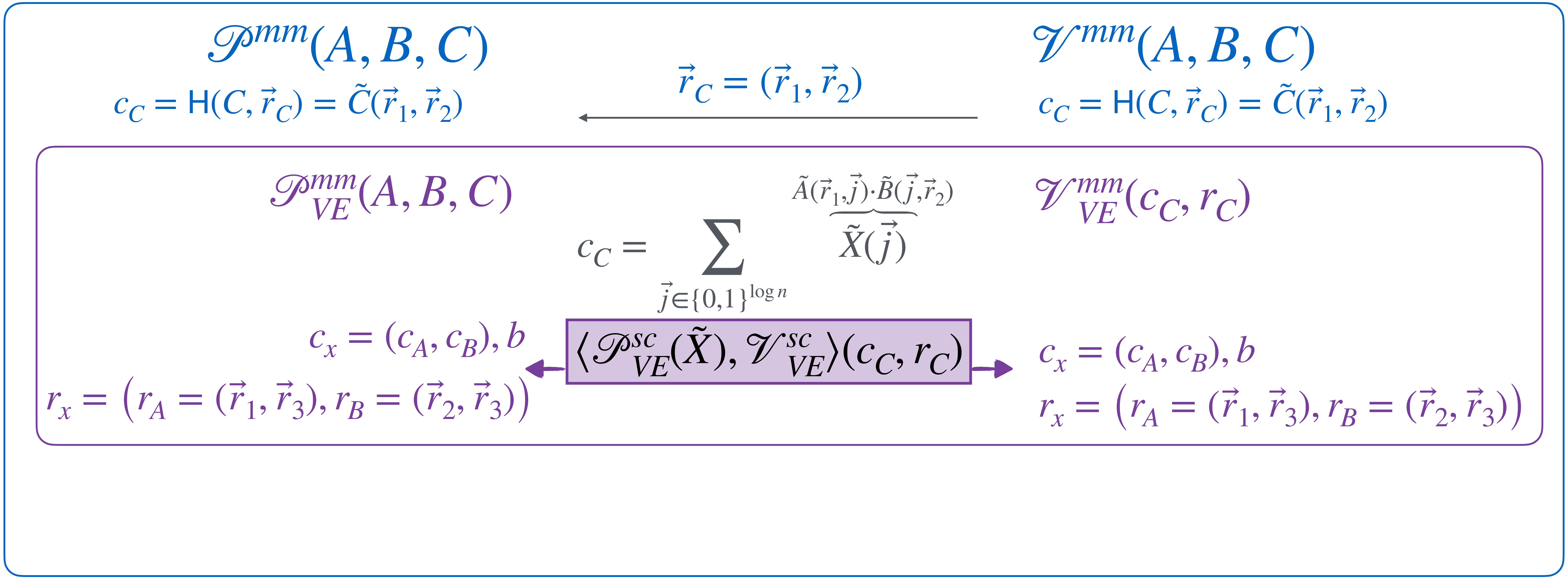
$\mathcal{V}^{mm}_{VE}(c_C, r_C)$

$$c_C = \sum_{\vec{j}\in\{0,1\}^{\log n}} \overbrace{\tilde{X}(\vec{j})}^{\tilde{A}(\vec{r}_1,\vec{j})\cdot\tilde{B}(\vec{j},\vec{r}_2)}$$

VE efficiency: communication&verification O(log n), prover O(n^2), faster than computing $A \cdot B$ in O(n^3)

VE for matrix multiplication [Thaler13]

Prove $C = A \cdot B$ for $A, B, C \in \mathbb{F}^{n \times n}$. Using MLE: $\forall \vec{i}, \vec{k} \in \{0,1\}^{\log n} : \tilde{C}(\vec{i}, \vec{k}) = \sum_{\vec{j} \in \{0,1\}^{\log n}} \tilde{A}(\vec{i}, \vec{j}) \cdot \tilde{B}(\vec{j}, \vec{k})$



$\mathscr{P}^{mm}(A, B, C)$

$\mathscr{V}^{mm}(A, B, C)$

$c_C = \mathsf{H}(C, \vec{r}_C) = \tilde{C}(\vec{r}_1, \vec{r}_2)$

$\xleftarrow{\quad \vec{r}_C = (\vec{r}_1, \vec{r}_2) \quad}$

$c_C = \mathsf{H}(C, \vec{r}_C) = \tilde{C}(\vec{r}_1, \vec{r}_2)$

$\mathscr{P}^{mm}_{VE}(A, B, C)$

$\mathscr{V}^{mm}_{VE}(c_C, r_C)$

$$c_C = \sum_{\vec{j} \in \{0,1\}^{\log n}} \underbrace{\tilde{A}(\vec{r}_1, \vec{j}) \cdot \tilde{B}(\vec{j}, \vec{r}_2)}_{\tilde{X}(\vec{j})}$$

$$\longleftarrow \boxed{\langle \mathscr{P}^{sc}_{VE}(\tilde{X}), \mathscr{V}^{sc}_{VE} \rangle (c_C, r_C)} \longrightarrow$$

$c_x = (c_A, c_B), b$

$r_x = \left(r_A = (\vec{r}_1, \vec{r}_3), r_B = (\vec{r}_2, \vec{r}_3) \right)$

$c_x = (c_A, c_B), b$

$r_x = \left(r_A = (\vec{r}_1, \vec{r}_3), r_B = (\vec{r}_2, \vec{r}_3) \right)$

VE efficiency: communication&verification O(log n), prover $O(n^2)$, faster than computing $A \cdot B$ in $O(n^3)$

VE for matrix multiplication [Thaler13]

Prove $C = A \cdot B$ for $A, B, C \in \mathbb{F}^{n \times n}$. Using MLE: $\forall \vec{i}, \vec{k} \in \{0,1\}^{\log n} : \tilde{C}(\vec{i}, \vec{k}) = \sum_{\vec{j} \in \{0,1\}^{\log n}} \tilde{A}(\vec{i}, \vec{j}) \cdot \tilde{B}(\vec{j}, \vec{k})$

$\mathscr{P}^{mm}(A, B, C)$

$c_C = \mathsf{H}(C, \vec{r}_C) = \tilde{C}(\vec{r}_1, \vec{r}_2)$

$\xleftarrow{\vec{r}_C = (\vec{r}_1, \vec{r}_2)}$

$\mathscr{V}^{mm}(A, B, C)$

$c_C = \mathsf{H}(C, \vec{r}_C) = \tilde{C}(\vec{r}_1, \vec{r}_2)$

$\mathscr{P}^{mm}_{VE}(A, B, C)$

$\mathscr{V}^{mm}_{VE}(c_C, r_C)$

$$c_C = \sum_{\vec{j} \in \{0,1\}^{\log n}} \overbrace{\tilde{X}(\vec{j})}^{\tilde{A}(\vec{r}_1, \vec{j}) \cdot \tilde{B}(\vec{j}, \vec{r}_2)}$$

$$\xleftarrow{} \boxed{\langle \mathscr{P}^{sc}_{VE}(\tilde{X}), \mathscr{V}^{sc}_{VE} \rangle (c_C, r_C)} \xrightarrow{}$$

$c_x = (c_A, c_B), b$

$r_x = \big(r_A = (\vec{r}_1, \vec{r}_3), r_B = (\vec{r}_2, \vec{r}_3)\big)$

$c_x = (c_A, c_B), b$

$r_x = \big(r_A = (\vec{r}_1, \vec{r}_3), r_B = (\vec{r}_2, \vec{r}_3)\big)$

$c_A \overset{?}{=} \mathsf{H}(A, r_A) = \tilde{A}(\vec{r}_1, \vec{r}_3)$

$c_B \overset{?}{=} \mathsf{H}(B, r_B) = \tilde{B}(\vec{r}_2, \vec{r}_3)$

VE efficiency: communication&verification O(log n), prover O(n^2), faster than computing $A \cdot B$ in O(n^3)

Proofs for convolution

zkCNN [LXZ21]

(2dim)Convolution → 1dim-convolution → poly mult → FFT → matrix multiplication

[BFGRS23]

Convolution → matrix multiplication with reshaped X, W

Performance for c input channels, *d* output channels

	[BFGRS23]	**zkCNN [LXZ21]**
Prover	$O(c\ \lvert W\rvert\ (\lvert Y\rvert + d))$	$O(c\ d\ \lvert X\rvert)$
Verifier	$O(\log(c\ \lvert Y\rvert))$	$O(\log^2(c\ d\ \lvert X\rvert))$
Proof size	$O(\log(c\ \lvert Y\rvert))$	$O(log^2(c\ d\ \lvert X\rvert))$

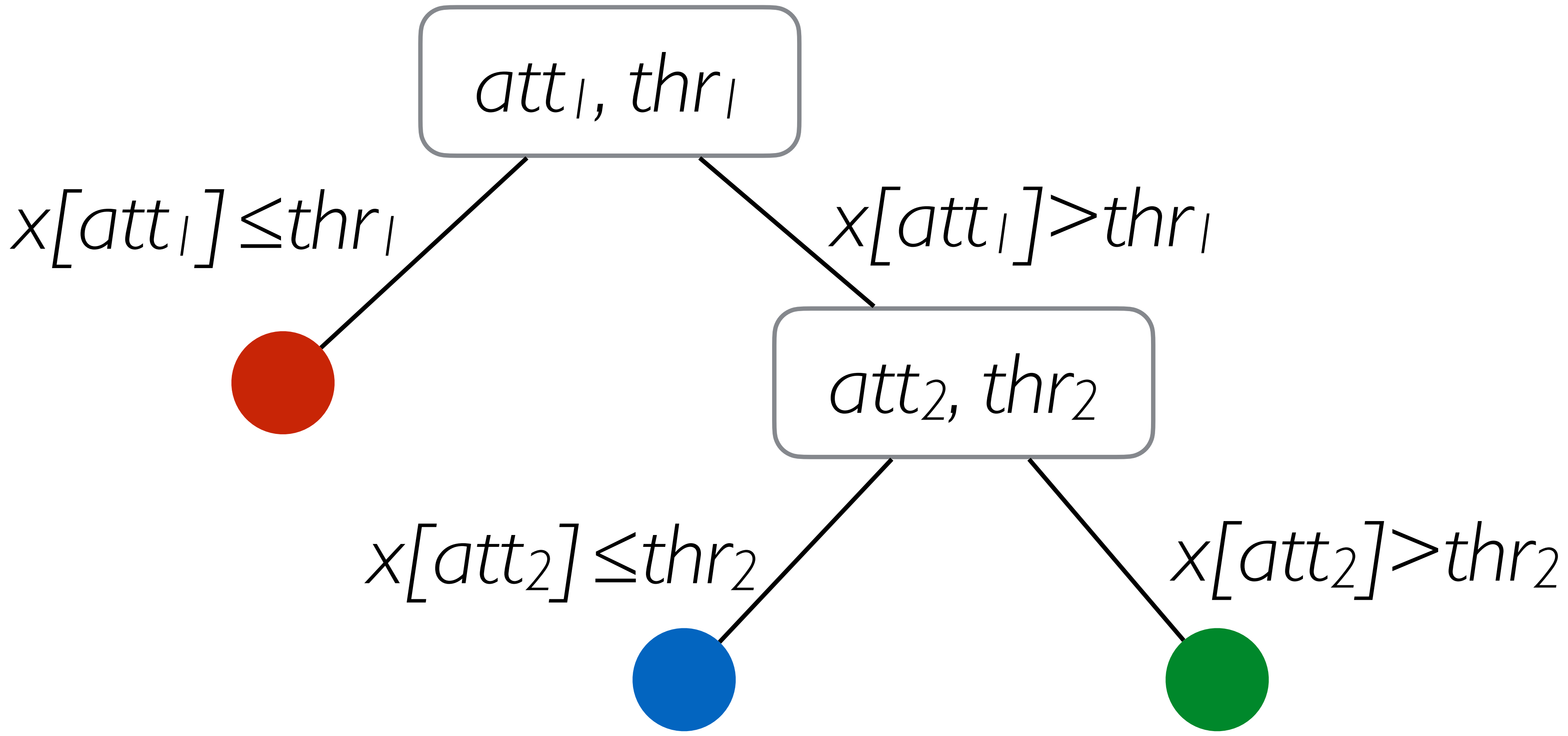
- Very efficient for small kernels $m^2 \leq d$ (VGG16 m=3, d→512)
- Confirmed experimentally (proving VGG11 ~5s)

ZKPs for Decision Trees

Tree $\mathscr{T} : \mathbb{F}^d \to [M]$

Height H, #nodes N

Classify $x = (x[att_1], \ldots, x[att_d])$



[zkDT] J. Zhang, Z. Fang, Y. Zhang, D. Song. *Zero Knowledge Proofs for Decision Tree Predictions and Accuracy*. CCS 2020.

Merkle hash of *T*. Proving inference = proving traversal from leaf (class) to root

[CFFLL24] M. Campanelli, A. Faonio, D. Fiore, T. Li, H. Lipmaa. *Lookup Arguments: Improvements, Extensions and Applications to Zero-Knowledge Decision Trees*. PKC 2024

Matrix-encoding of *T*. Proving inference = proving matrix lookup (reduced to vector lookup)

ZKPs for secure ML

This talk:

How to use ZKPs and Commitments to prove ML inference

Efficient solutions for CNNs and Decision Trees

ZKPs for secure ML

This talk:

How to use ZKPs and Commitments to prove ML inference

Efficient solutions for CNNs and Decision Trees

Other related problems

ZKPs for secure ML

This talk:

How to use ZKPs and Commitments to prove ML inference

Efficient solutions for CNNs and Decision Trees

Other related problems

Proofs of accuracy: prove that model W achieves a claimed accuracy level

Prove $Y_i=F(X_i, W)$ over dataset of labeled samples $\{X_i\}$ and compares results to labels

ZKPs for secure ML

This talk: How to use ZKPs and Commitments to prove ML inference

Efficient solutions for CNNs and Decision Trees

Other related problems

Proofs of accuracy: prove that model W achieves a claimed accuracy level

Prove $Y_i=F(X_i, W)$ over dataset of labeled samples $\{X_i\}$ and compares results to labels

Proofs of training: prove W=NN-Train(Data)

Challenge: several iterations of inference-like computations

[GGJMMP23] S. Garg, A. Goel, S. Jha, S. Mahloujifar, M. Mahmoody, G. Policharla, M. Wang. *Experimenting with Zero-Knowledge Proofs of Training*. CCS 2023

[APKP24] K. Abbaszadeh, C. Pappas, J. Katz, D. Papadopoulos. *Zero-Knowledge Proofs of Training for Deep Neural Networks*. CCS 2024

Thanks!

Questions ?