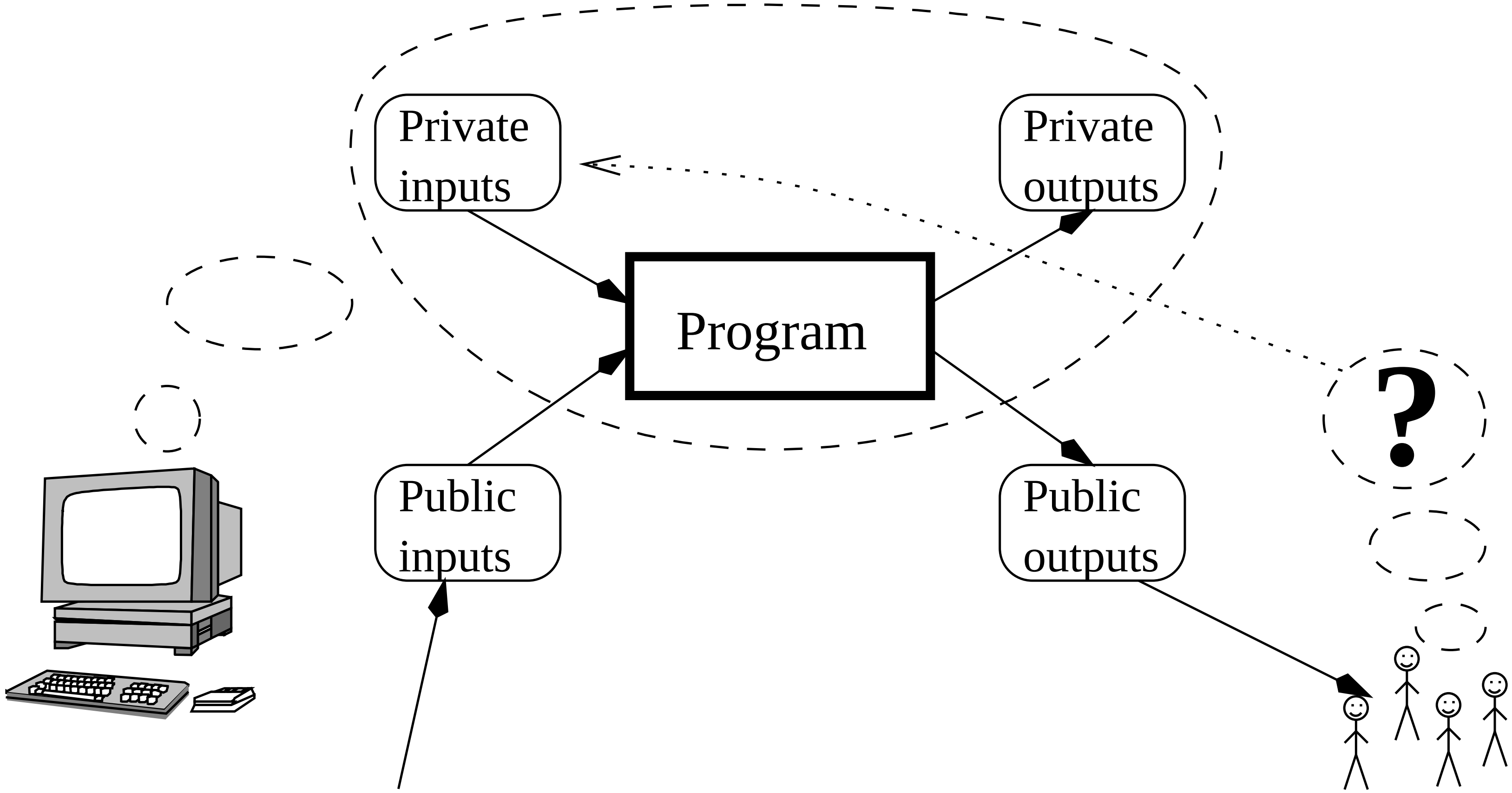
Handling Cryptographic Primitives in an Analysis for Secure Information Flow

Peeter Laud

Universität des Saarlandes

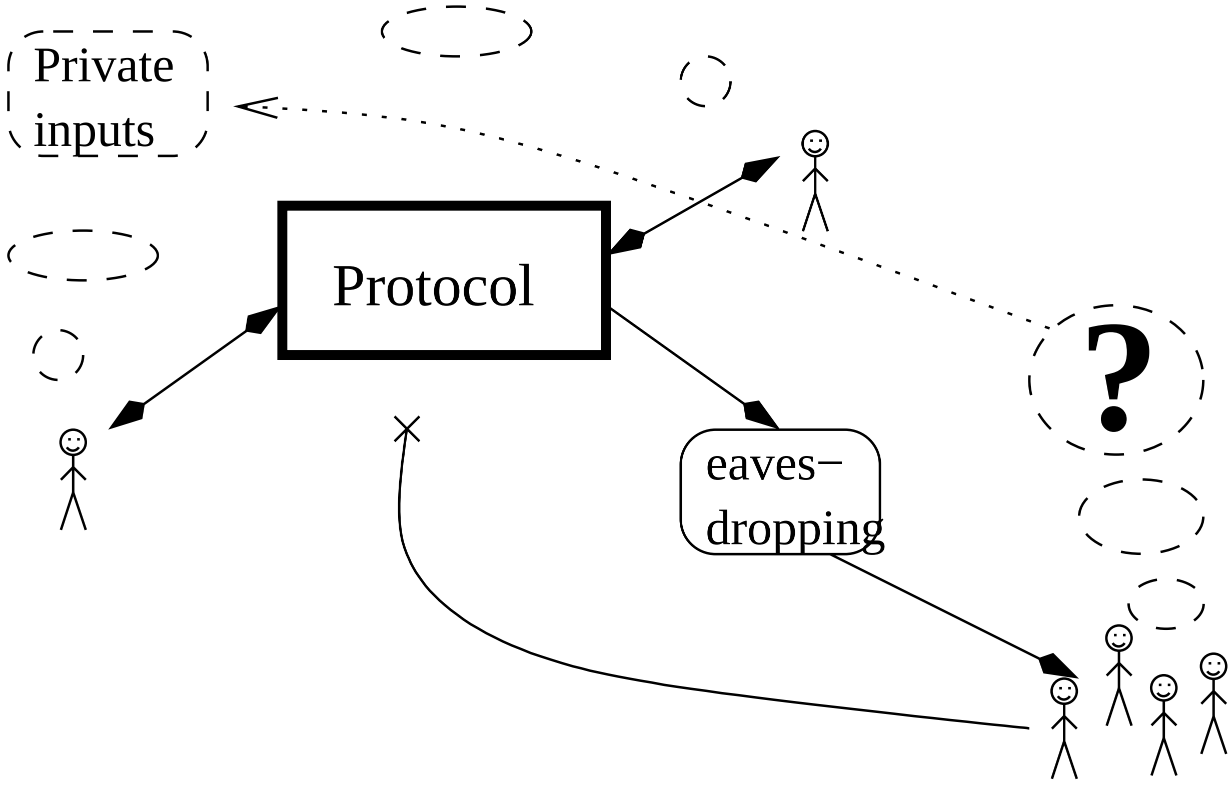
Secure information flow



Distribution of inputs $D_s \in \mathcal{D}(\mathbf{State})$ is public knowledge.

The program might contain cryptographic operations

Relevance to protocol analysis



Security against passive adversary.

Structure of the talk

- Definition of secrecy
  - indistinguishability of distributions
  - independence of variables
- Abstraction [of distributions over program states]
- Abstract semantics, in syntax-directed way

  WHILE-language

  Encryption as a primitive operation

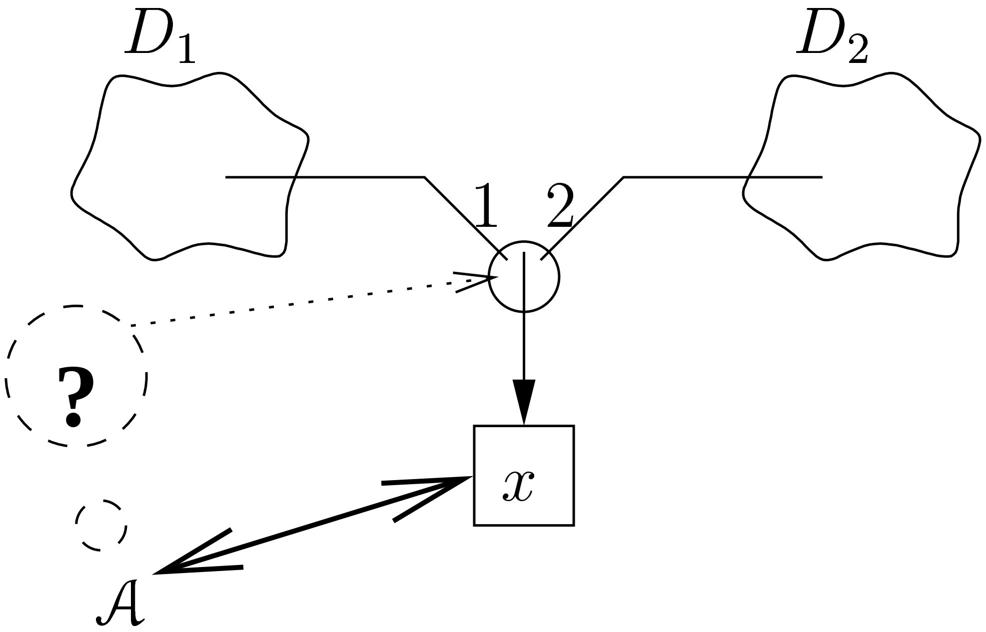
$\boxed{\textbf{Secrecy definition}}$

- Imperative programming language.
- Program state: $\mathbf{State} = \mathbf{Var} \to \mathbf{Val}$.
- Program semantics: $[\![\mathsf{P}]\!] : \mathbf{State} \to \mathcal{D}(\mathbf{State})$.
- Secret inputs: $\mathbf{Var}_{\mathrm{S}} \subseteq \mathbf{Var}$, public outputs $\mathbf{Var}_{\mathrm{P}} \subseteq \mathbf{Var}$.

Assume that P does not change the values of $\mathbf{Var}_{\mathrm{S}}$.

P is *secure* $\Leftrightarrow$ values of $\mathbf{Var}_{\mathrm{S}}$ and $\mathbf{Var}_{\mathrm{P}}$ are *independent* in $[\![\mathsf{P}]\!](D_s)$.

Indistinguishability

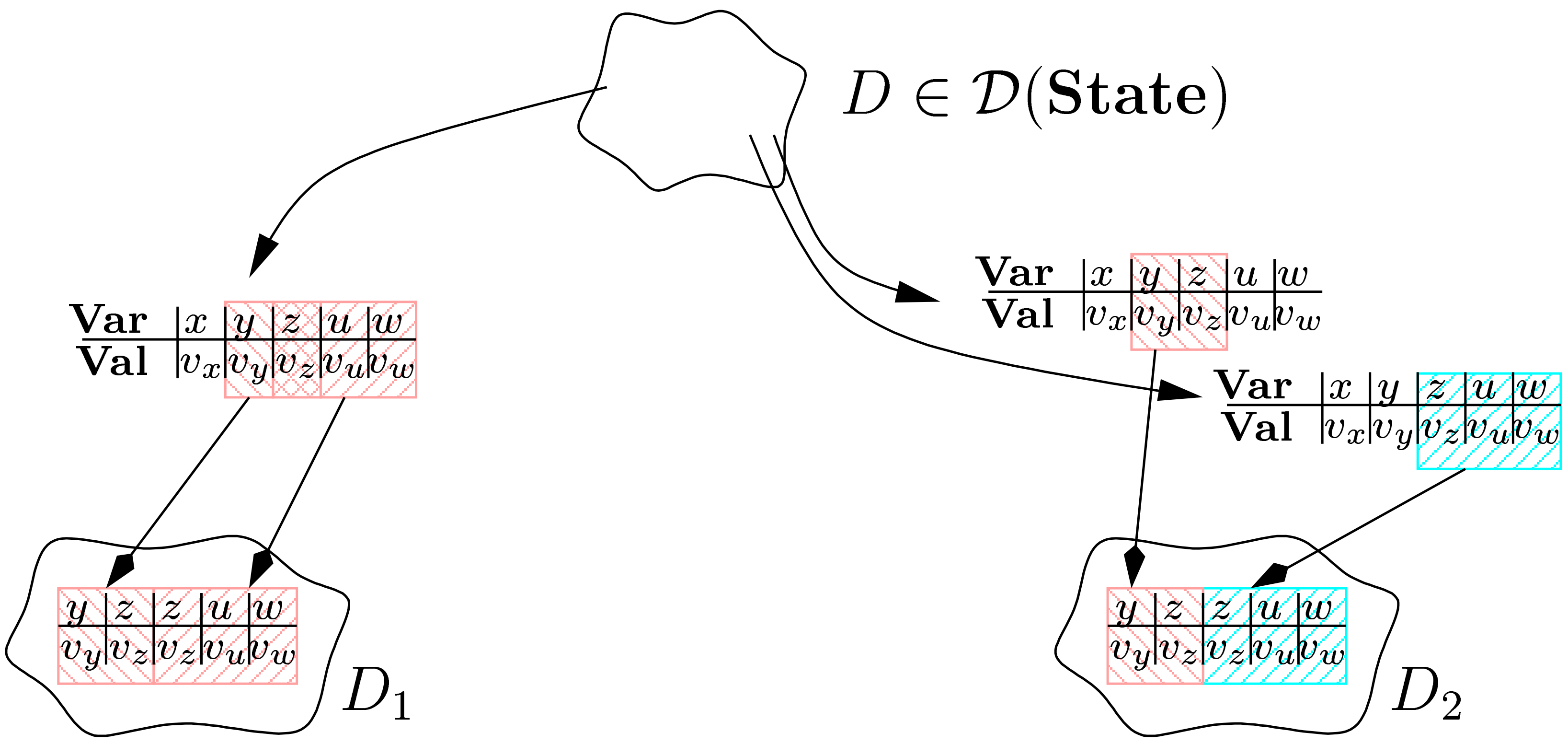


$$D_1, D_2 \in \mathcal{D}(\mathcal{X})$$

D_1 and D_2 are $(\mathbf{B}, \epsilon)$-*indistinguishable*:

$$\forall \mathcal{A} \in \mathbf{B} : \quad \underbrace{\Pr\big[x \leftarrow D_1 \ : \ \mathcal{A}(x) = 1\big] - \Pr\big[x \leftarrow D_2 \ : \ \mathcal{A}(x) = 1\big]}_{\textit{advantage} \text{ of } \mathcal{A} \text{ for } (D_1, D_2)} \leq \epsilon \ .$$

Independence of sets of variables



Values of $\{y, z\}$ and $\{z, u, w\}$ are *independent* in $D \Leftrightarrow$

D_1 and D_2 are indistinguishable.

$\boxed{\textbf{Polynomiality}}$

Fixing $(\mathbf{B}, \epsilon)$.

- *Security parameter* $n \in \mathbb{N}$
- $\mathbf{Val}_n = \{\mathbf{0}, \mathbf{1}\}^{\ell(n)}$, $\ell \in \mathbb{Z}[n]$
- $\mathbf{State}_n = \mathbf{Var} \to \mathbf{Val}_n$
- $[\![\mathsf{P}]\!] \in \prod_{n\in\mathbb{N}}(\mathbf{State}_n \to \mathcal{D}(\mathbf{State}_n))$

- $\mathcal{A} \in \mathbf{B}$ iff $\mathcal{A}(\mathbf{1}^n, \cdot)$ works in time polynomial to n.
- ϵ — any function (of n) smaller than $\dfrac{1}{poly(n)}$.

Abstraction

- $\beta(D) \in \mathcal{P}_{\mathrm{L}}(\mathcal{P}(\mathbf{Var}) \times \mathcal{P}(\mathbf{Var}))$
  $(X, Y) \in \beta(D) \Leftrightarrow$ values of X and Y are independent in D.

- Abstract semantics
  $\mathcal{W}(\mathsf{P}) : \mathcal{P}_{\mathrm{L}}(\mathcal{P}(\mathbf{Var}) \times \mathcal{P}(\mathbf{Var})) \to \mathcal{P}_{\mathrm{L}}(\mathcal{P}(\mathbf{Var}) \times \mathcal{P}(\mathbf{Var}))$

- Correctness criterion:

$$(X, Y) \in \mathcal{W}(\mathsf{P})(\beta(D_s)) \Rightarrow (X, Y) \in \beta(\llbracket \mathsf{P} \rrbracket(D_s))$$

Definition of $\mathcal{W}(\mathsf{P})$ (as well as $\llbracket \mathsf{P} \rrbracket$) is syntax-directed.

$\boxed{\textbf{Syntax}}$

$$
\begin{array}{lcl}
\mathsf{P} & ::= & x := o(x_1, \ldots, x_k) \\
 & & x := \mathcal{E}nc(k, y) \\
 & & x := \mathcal{G}en() \\
 & | & skip \\
 & | & \mathsf{P}_1 ; \mathsf{P}_2 \\
 & | & if\ \ b\ \ then\ \mathsf{P}_1\ \ else\ \mathsf{P}_2 \\
 & | & while\ b\ \ do\ \mathsf{P}_1,
\end{array}
$$

$$\mathcal{W}(skip) = id. \qquad\qquad \mathcal{W}(\mathsf{P}_1 ; \mathsf{P}_2) = \mathcal{W}(\mathsf{P}_2) \circ \mathcal{W}(\mathsf{P}_1).$$

Abstract semantics — assignments (1/2)

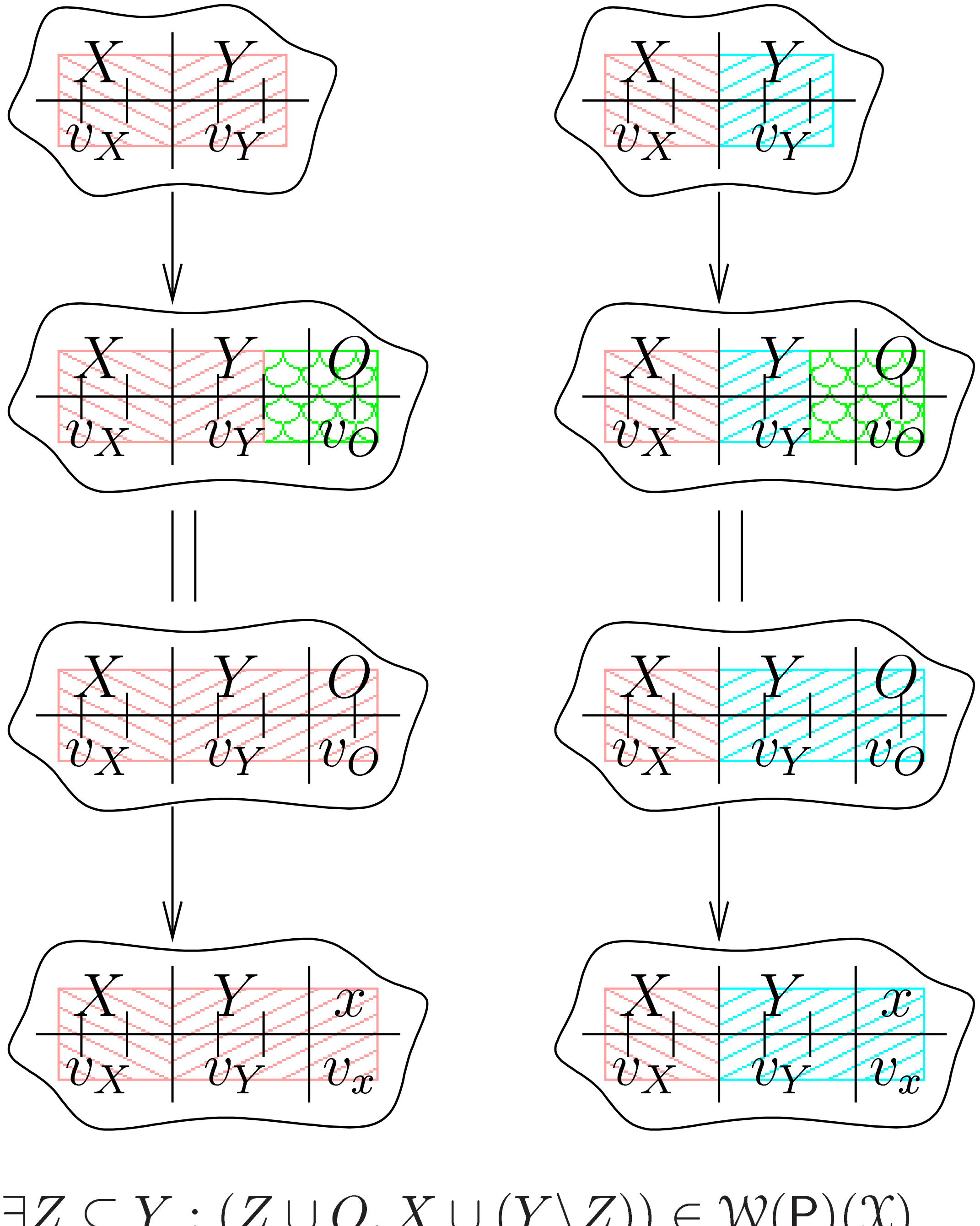
Program P: $x := o(x_1, \ldots, x_k)$

- Let $\mathcal{X} \in \mathcal{P}_{\mathrm{L}}(\mathcal{P}(\mathbf{Var}) \times \mathcal{P}(\mathbf{Var}))$, $(X, Y) \in \mathcal{X}$, $x \notin X \cup Y$.
- Then $(X, Y) \in \mathcal{W}(\mathsf{P})(\mathcal{X})$.

- When may we have $(X, Y \cup \{x\}) \in \mathcal{W}(\mathsf{P})(\mathcal{X})$?
- Necessary condition: $(X, Y) \in \mathcal{W}(\mathsf{P})(\mathcal{X})$.
- When does from "X and Y independent" follow "X and $Y \cup \{x\}$ independent"?
- When does from "X and $Y \cup \{x\}$ dependent" follow "X and Y dependent"?
- "Being dependent" = "there is a certain distinguishing algorithm".

Abstract semantics — assignments (2/2)

$O = \{x_1, \dots, x_k\} \backslash Y$

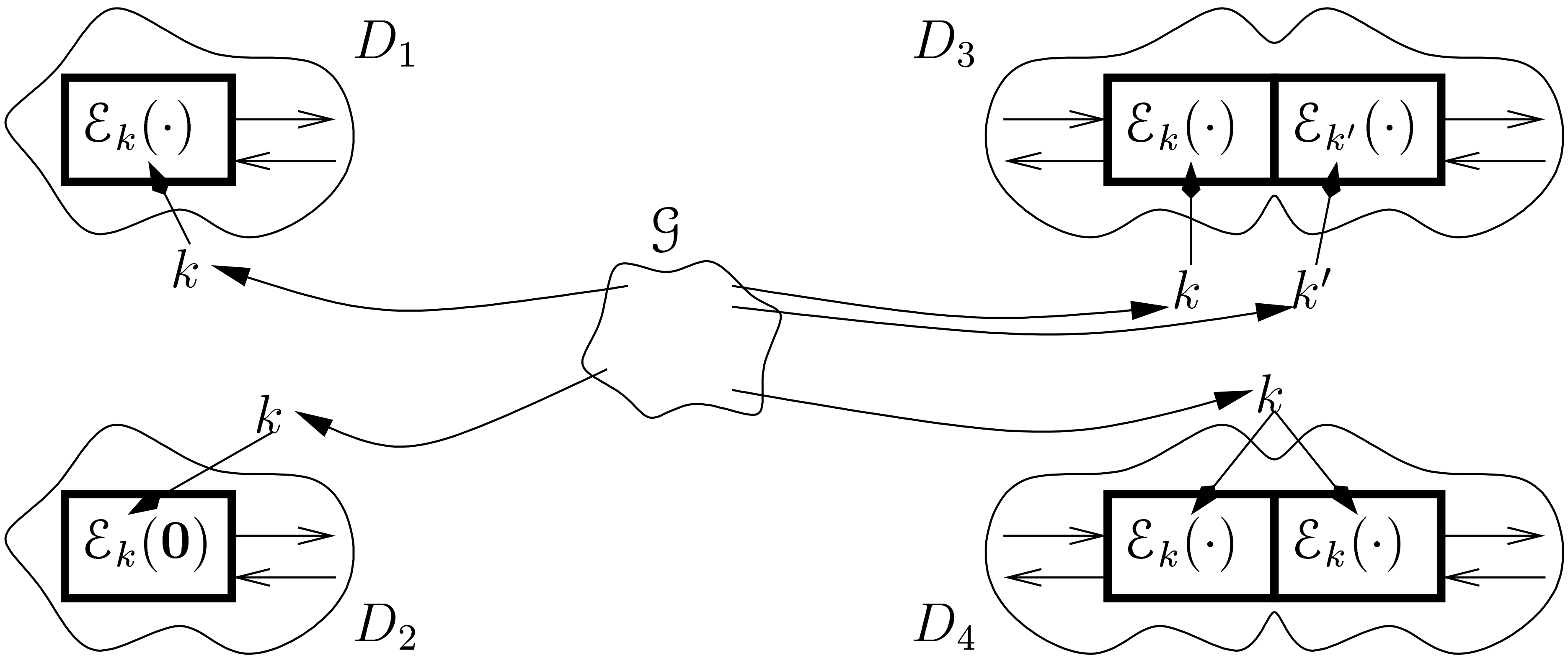


$\exists Z \subseteq Y : (Z \cup O, X \cup (Y \backslash Z)) \in \mathcal{W}(\mathsf{P})(\mathcal{X})$

Security of encryption primitive

Encryption system $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is

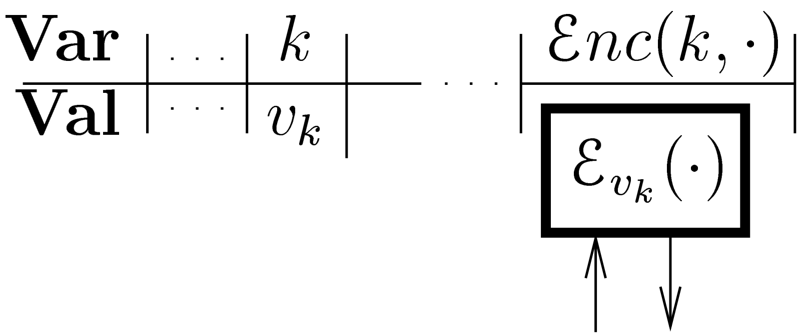
- *repetition-concealing*, if D_1 and D_2 are indistinguishable;
- *which-key concealing*, if D_3 and D_4 are indistinguishable.



$\boxed{\textbf{Extension for } \mathcal{P}_{\mathrm{L}}(\mathcal{P}(\mathbf{Var}) \times \mathcal{P}(\mathbf{Var}))}$

After $x := \mathcal{E}nc(k, y)$, $\{x\}$ and $\{k\}$ may be dependent.

Add fictious elements to **Var**:



$\{x\}$ and $\{\mathcal{E}nc(k, \cdot)\}$ are (in general) independent.

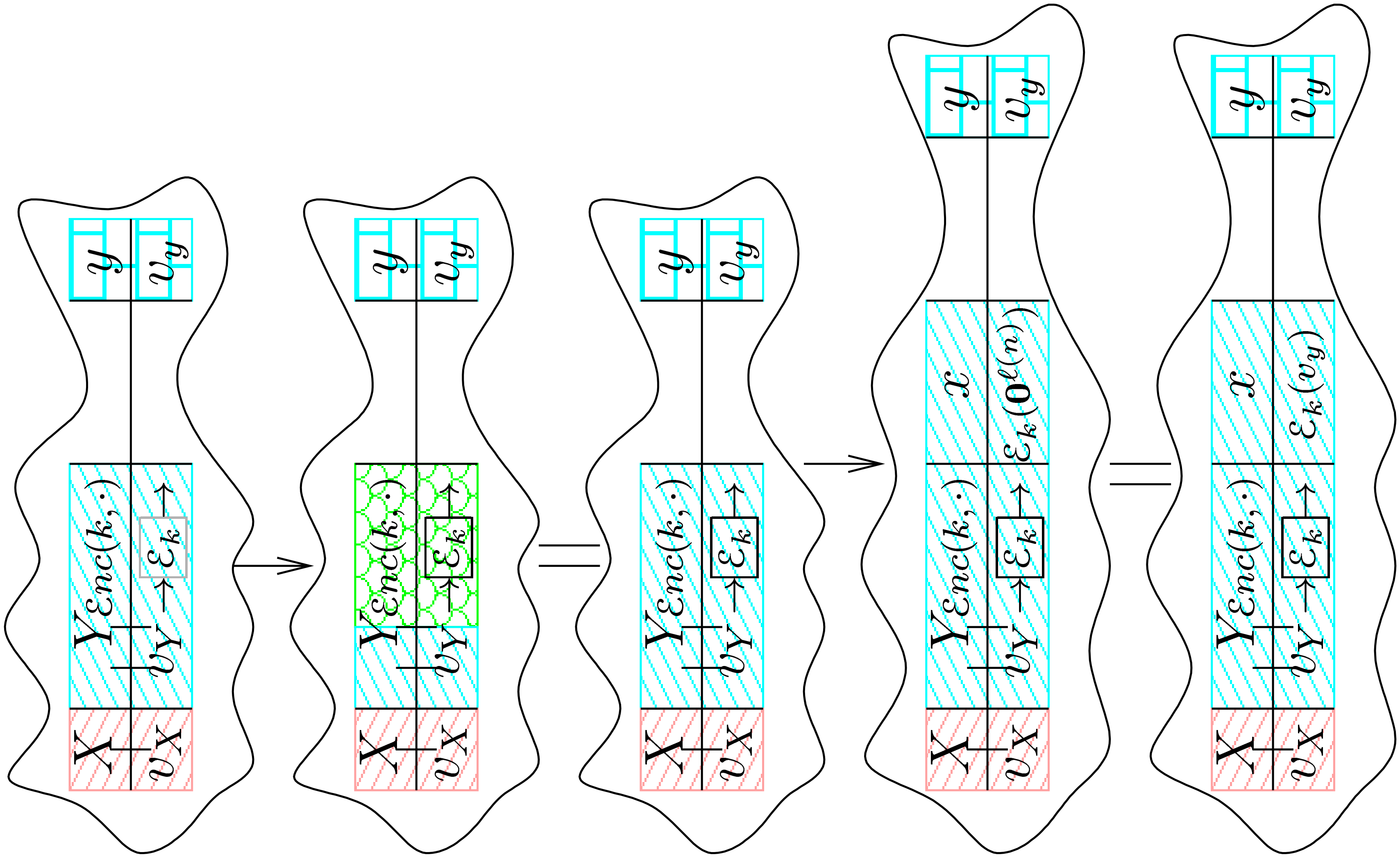
Abstract semantics — encryptions

Program P: $x := \mathcal{E}nc(k, y)$

$$\frac{(X,Y) \in \mathcal{W}(\mathsf{P})(\mathcal{X}) \qquad (\{\mathcal{E}nc(k,\cdot)\}, X \cup (Y \setminus \{\mathcal{E}nc(k,\cdot)\}) \cup \{y\}) \in \mathcal{W}(\mathsf{P})(\mathcal{X})}{(X, Y \cup \{x\}) \in \mathcal{W}(\mathsf{P})(\mathcal{X})}$$

Program P: $k := \mathcal{G}en()$

$$\frac{(X,Y) \in \mathcal{W}(\mathsf{P})(\mathcal{X})}{(X \cup \{\mathcal{E}nc(k,\cdot)\}, Y \cup \{\mathcal{E}nc(k,\cdot)\}) \in \mathcal{W}(\mathsf{P})(\mathcal{X})}$$



$\mathbf{X}$
υ_{X}
$\mathbf{Y}$
υ_{Y}
$\mathcal{E}nc(k,\cdot)$
$\mathcal{E}_k$
x
$\mathcal{E}_k(\mathbf{0}^{\ell(n)})$
$\mathcal{E}_k(\upsilon_y)$
y
υ_y

Abstract semantics — if-then-else

Program *if* b *then* $\mathsf{P}_{\mathsf{true}}$ *else* $\mathsf{P}_{\mathsf{false}}$ equals

$$N := b$$

$$\mathsf{P}'_{\mathsf{true}}$$

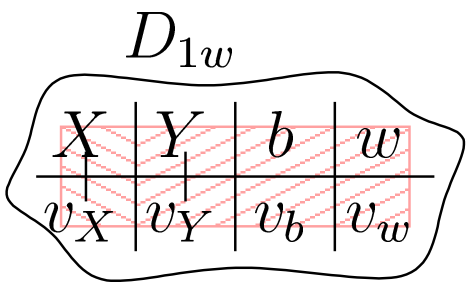
$$\mathsf{P}'_{\mathsf{false}}$$

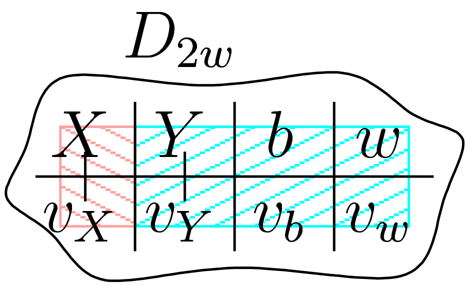
$$\langle x := \mathit{if}\ N\ \mathit{then}\ x_{\mathsf{true}}\ \mathit{else}\ x_{\mathsf{false}} \rangle_{x \in \mathbf{Var}_{\mathrm{asgn}}}$$

- $\mathbf{Var}_{\mathrm{asgn}} \subseteq \mathbf{Var}$ — the set of variables that are assigned to in $\mathsf{P}_{\mathsf{true}}$ or $\mathsf{P}_{\mathsf{false}}$
- $\mathsf{P}'_{\mathsf{true}}$ and $\mathsf{P}'_{\mathsf{false}}$ — $\mathsf{P}_{\mathsf{true}}$ and $\mathsf{P}_{\mathsf{false}}$ with renamed variables

Consider the program P: $x := \mathit{if}\ b\ \mathit{then}\ y\ \mathit{else}\ z$

$$\frac{(X, Y \cup \{b, y\}) \in \mathcal{W}(\mathsf{P})(\mathcal{X}) \qquad (X, Y \cup \{b, z\}) \in \mathcal{W}(\mathsf{P})(\mathcal{X})}{(X, Y \cup \{b, x\}) \in \mathcal{W}(\mathsf{P})(\mathcal{X})}$$





where $w \in \{x, y, z\}$.

Suppose $\mathcal{A}$ distinguishes D_{1x} and D_{2x}.

$\mathcal{B}_y$ tries to distinguish D_{1y} and D_{2y}:

- if $v_b = \mathsf{true}$, invoke $\mathcal{A}$;
- if $v_b = \mathsf{false}$, return "don't know".

$\mathcal{B}_z$ similarly tries to distinguish D_{1z} and D_{2z}.

$$\langle \text{advantage of } \mathcal{A} \rangle = \langle \text{advantage of } \mathcal{B}_y \rangle + \langle \text{advantage of } \mathcal{B}_z \rangle$$

$\boxed{\textbf{Abstract semantics — while-loops}}$

$\mathcal{W}(\textit{while } b \textit{ do } \mathsf{P}) = (\mathcal{W}(\textit{if } b \textit{ then } \mathsf{P} \textit{ else skip}))^*$ (Kleene star)

- multiplication — functional composition
- addition — pointwise intersection