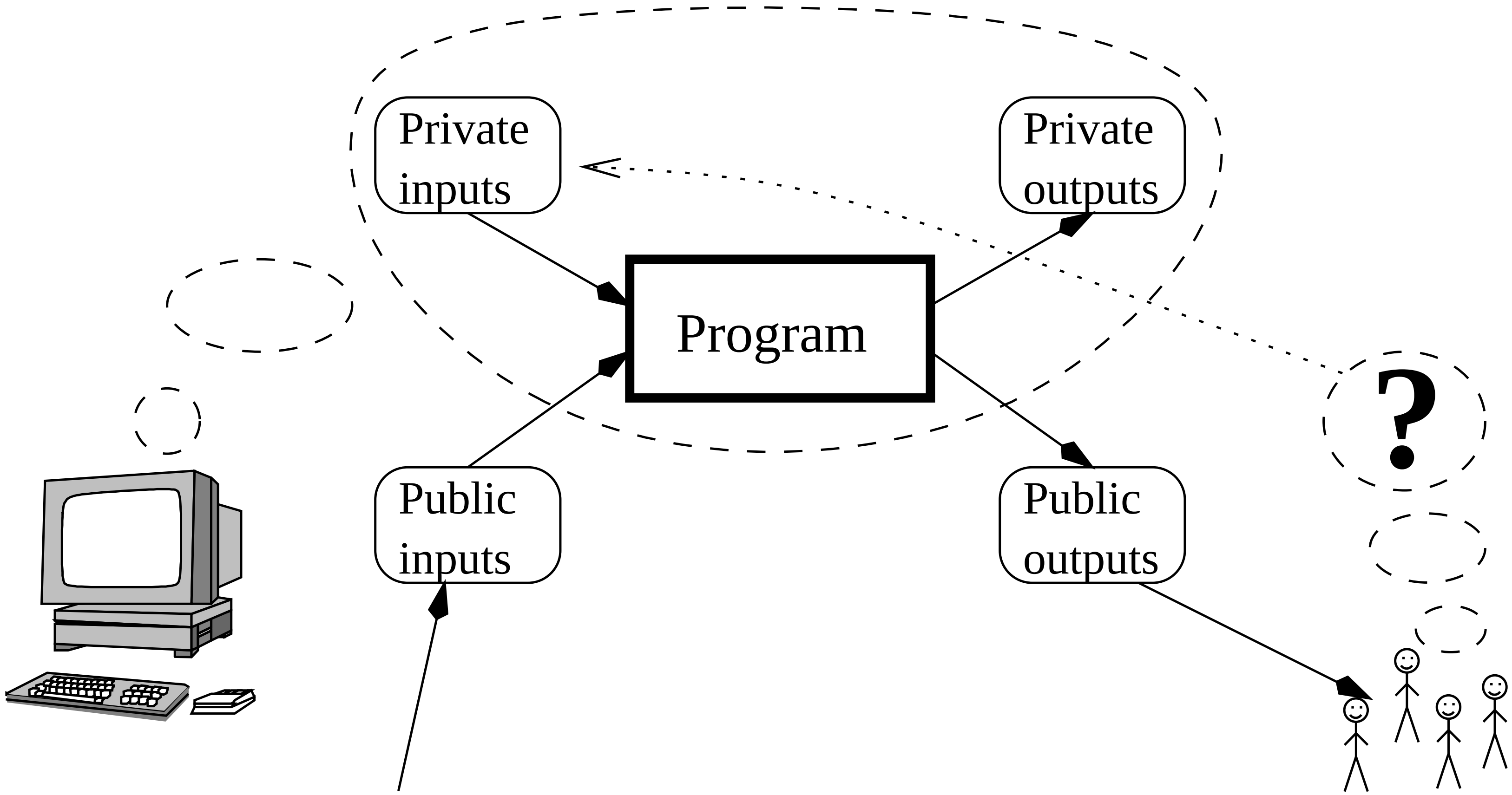
Computationally Secure Information Flow

Peeter Laud
Universität des Saarlandes

Problem statement



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

The program might contain cryptographic operations

$\boxed{\textbf{Structure of the talk}}$

- Syntax
- Security definition
  - A simple measure of security
  - A suitable semantics
  - Final definition
- An analysis detail

$\boxed{\textbf{Syntax}}$

The WHILE-language. A statement s is one of

•	Assignment	$x := o(x_1, \ldots, x_k)$	$x, x_1, \ldots, x_k \in \mathbf{Var}$, $o \in \mathbf{Op}$ A distinguished operator $\texttt{encrypt} \in \mathbf{Op}$
•	Sequence	$s_1; s_2$	
•	Branch	$\texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2$	$b \in \mathbf{Var}$
•	Loop	$\texttt{while}\ b\ \texttt{do}\ s_1$	$b \in \mathbf{Var}$

Represent programs as flowcharts

- A node is an assignment or a test

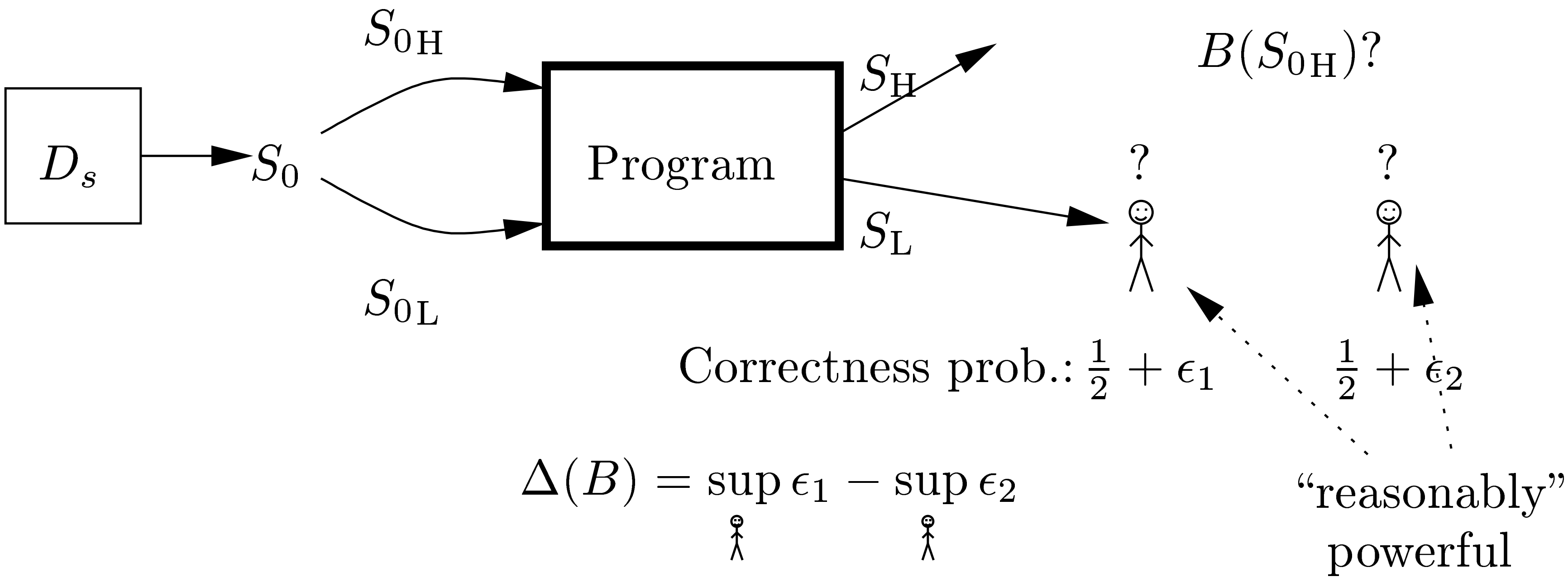
Set $\mathbf{Var}$ is partitioned: $\mathbf{Var} = \mathbf{Var}_{\mathrm{H}} \uplus \mathbf{Var}_{\mathrm{L}}$

Measuring security

Semantics: $[\![\mathsf{P}]\!] : \mathbf{State} \to \mathbf{State}_\perp$

$\mathbf{State} = \mathbf{Var} \to \mathbf{Val}$

$B : \mathbf{State}_{\mathrm{H}} \to \{0, 1\}$

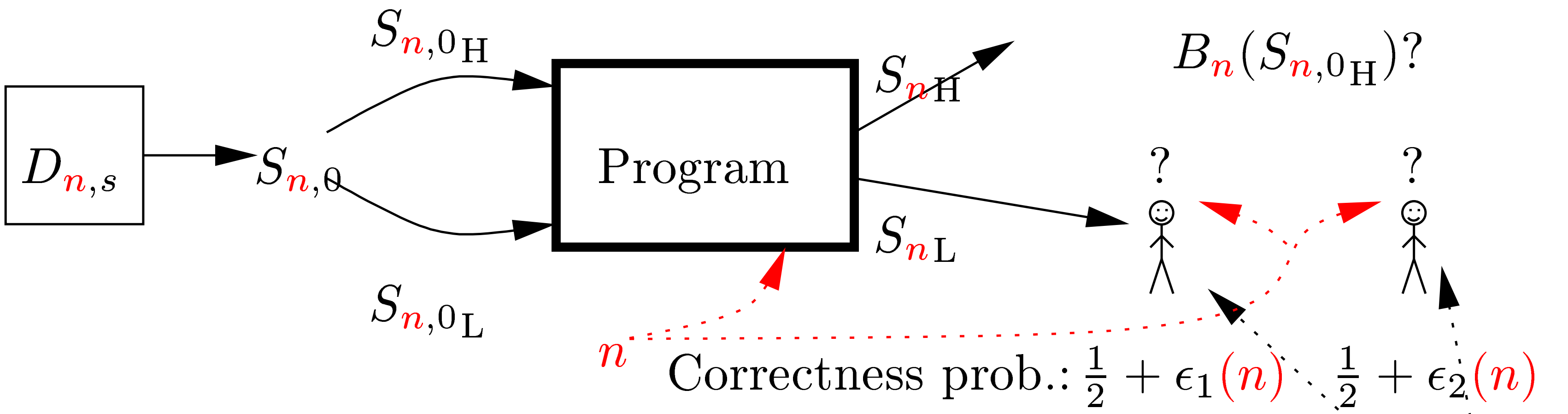


(In)security measure: $\sup_B \Delta(B)$.

$\boxed{\textbf{Asymptotics: easier than numbers}}$

Semantics: $[\![\mathsf{P}]\!] : \prod_{n\in\mathbb{N}}(\mathbf{State}_n \to \quad \mathbf{State}_{n\perp}\ )$
$\mathbf{State}_n = \mathbf{Var} \to \mathbf{Val}_n$

$B_n : \mathbf{State}_{n\mathrm{H}} \to \{0,1\}$, $\exists$ pol. time algo. $\mathcal{B}$, such that $\mathcal{B}(n,\cdot) = B_n$



$$\Delta(B)(n) = \big(\sup \epsilon_1\big)(n) - \big(\sup \epsilon_2\big)(n)$$

running time
poly(n)

Def: Program is secure $\Leftrightarrow$ $\big(\sup_B \Delta(B)\big)(n)$ is negligible in n

$\mathcal{D}(\qquad \qquad)$

$random$

Program with superpolynomial running time

`x2 := encrypt(k, y2)`	
`x1 := encrypt(k, y1)`	
`for i := 0 to max_val() do`	
`  y1' := decrypt(i, x1)`	Crack the key k
`  if y1 = y1' then`	by brute force,
`    break`	using known plaintext-ciphertext
`  fi`	pair (y1, x1)
`od`	here $\mathtt{i} = \mathtt{k}$
`y2' := decrypt(i, x2)`	here $\mathtt{y2}' = \mathtt{y2}$

Such a program should look like a non-terminating one.

The semantics should account for running time.

```
z = y1; x1 := encrypt(k, z)
```

Paths through the flowchart

$\mathbf{Path} \subset \mathit{Node}^*$ — the set of all paths from the beginning of the flowchart to the end.

$[\![\mathsf{P}]\!]_{\mathrm{Path}} : \mathbf{Path} \to \prod_{n\in\mathbb{N}}(\mathbf{State}_n \to \mathcal{D}(\mathbf{State}_{n\perp}))$

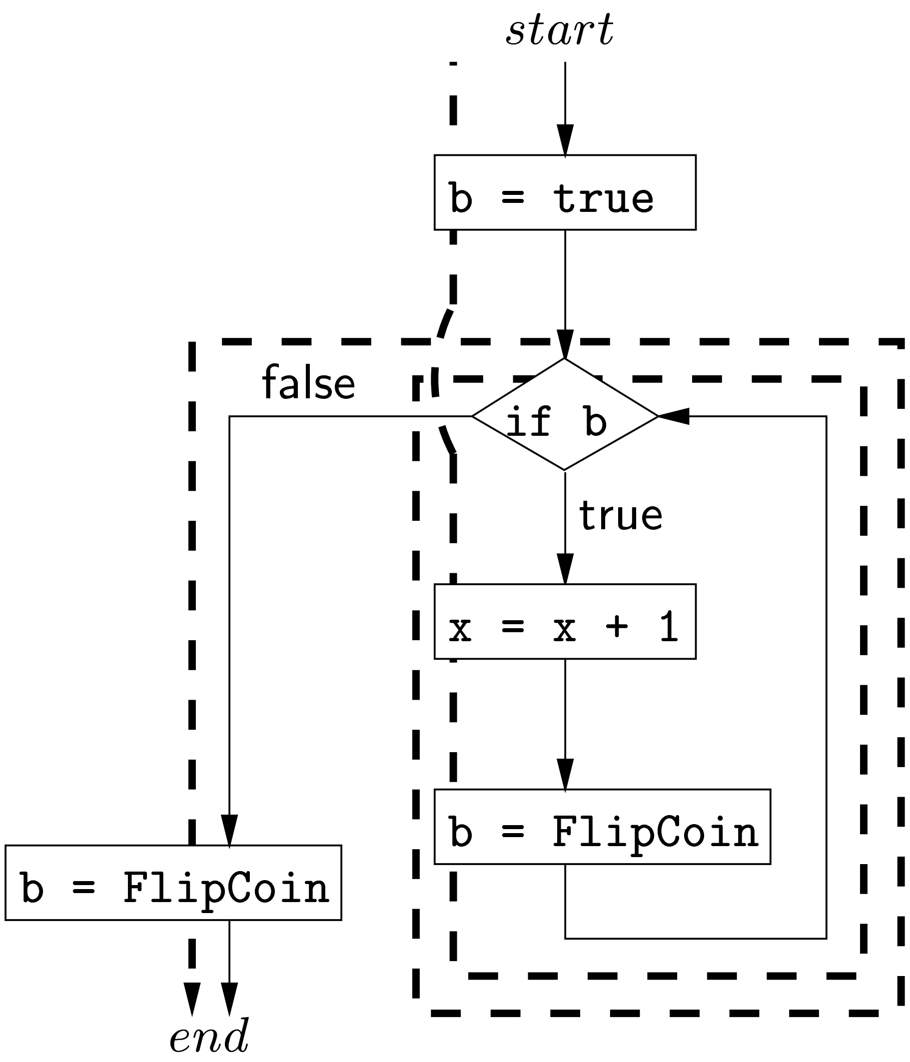
$\mathcal{S} : \mathbb{N} \to \mathcal{P}(\mathbf{Path})$ is *polynomially long*, $\iff$

$$\exists q \in \mathbb{Z}[n] \text{ such that } \forall n \in \mathbb{N},\ \forall P \in \mathcal{S}(n) : |P| \leq q(n).$$

$\mathcal{PLP}$ — the set of all pol. long computations.

$[\![\mathsf{P}]\!] : \mathcal{PLP} \to \prod_{n\in\mathbb{N}}(\mathbf{State}_n \to \mathcal{D}(\mathbf{State}_{n\perp}))$

$$[\![\mathsf{P}]\!](\mathcal{S})(S_{n,0}) = \left\{ \begin{array}{l} S_n \mapsto \sum\limits_{P\in\mathcal{S}(n)} [\![\mathsf{P}]\!]_{\mathrm{Path}}(P)(S_{n,0})(S_n) \\ \perp \ \mapsto \text{what remains over}\ldots \end{array} \right\}$$



Denote the marked path by P.

$[\![\mathsf{P}]\!]_{\texttt{Path}}(P)$ maps each

$$\left\{\begin{array}{l} x \mapsto x_{\text{init}} \\ b \mapsto b_{\text{init}} \end{array}\right\}$$

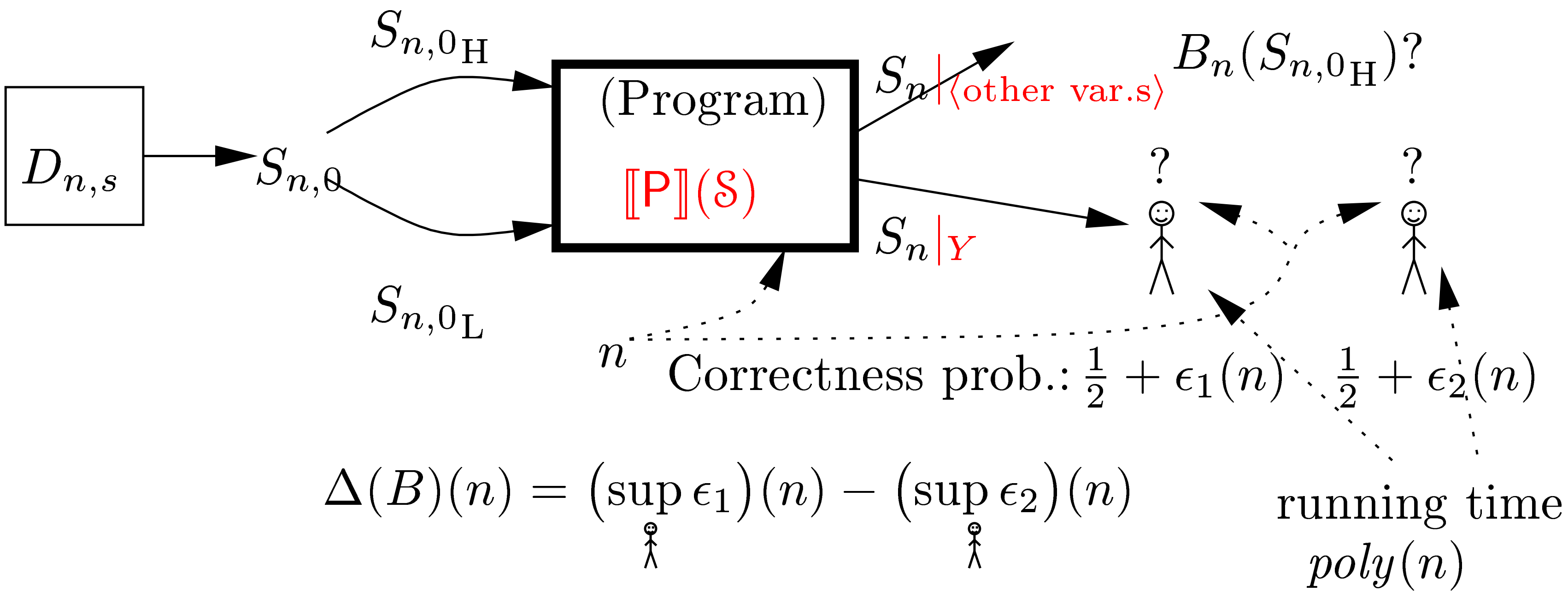
to

$$\frac{1}{8}\left\{\begin{array}{l} x \mapsto x_{\text{init}} + 2 \\ b \mapsto \mathsf{false} \end{array}\right\} + \frac{1}{8}\left\{\begin{array}{l} x \mapsto x_{\text{init}} + 2 \\ b \mapsto \mathsf{true} \end{array}\right\} + \frac{3}{4}\perp.$$

Recoverability (1/2)

Final values of $Y \subseteq \mathbf{Var}$ *help finding* $B_n(S_{n,0_{\mathrm{H}}}) \iff$

If P runs long enough	$\exists \mathcal{S}_0 \in \mathcal{PLP}$
or even longer	$\forall \mathcal{S} \geq \mathcal{S}_0 \ (\mathcal{S} \in \mathcal{PLP})$



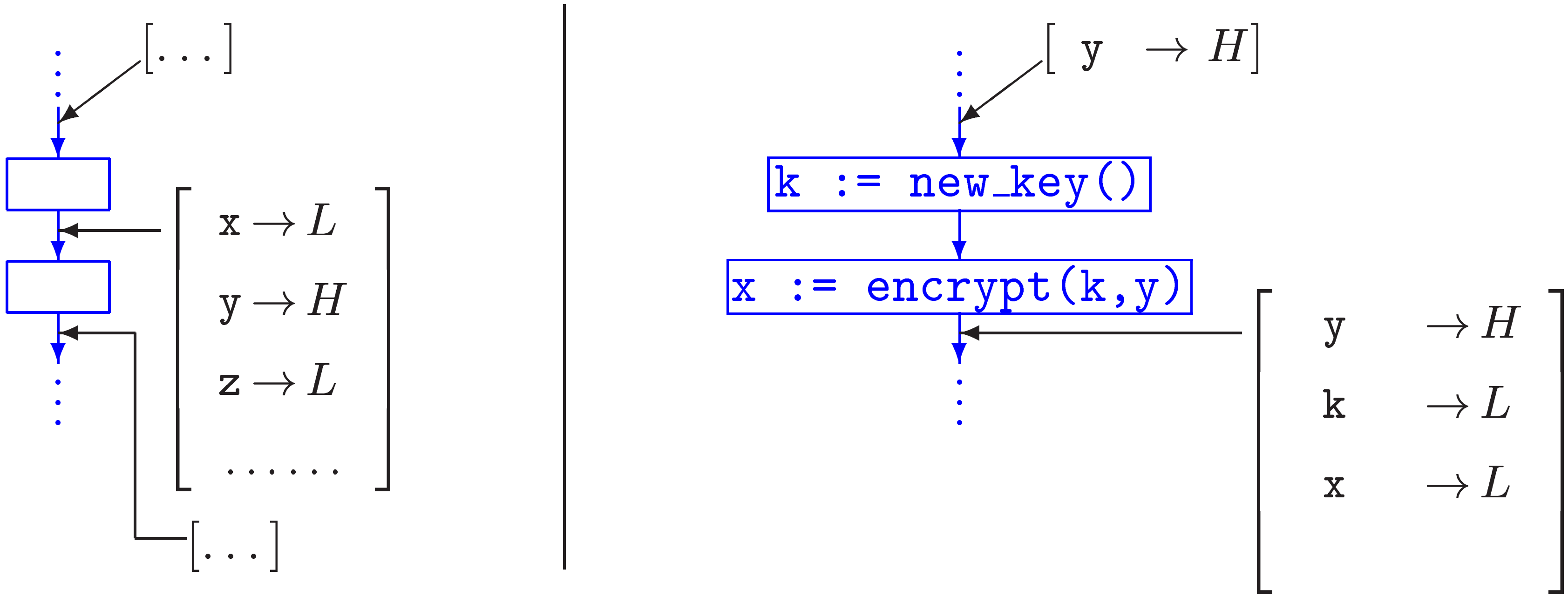
where $\Delta(B)(n)$ is not negligible in n.

$\boxed{\textbf{Recoverability (2/2)}}$

$S_{n,0_{\mathrm{H}}}$ is *recoverable* from final values of $Y \iff$
$\exists B$: final values of Y help finding $B_n(S_{n,0_{\mathrm{H}}})$.

P is *secure* $\iff$
$S_{n,0_{\mathrm{H}}}$ is not recoverable from final values of $\mathbf{Var}_{\mathrm{L}}$.

Enhancements to analysis



"x → H": x may depend on secret inputs at this program point
(in a "recognizable" way)

$\{\ \}$

$\{\ \}$

$\{\ \}$

$\{\ \}$

$\{\mathtt{x}, \mathtt{k}\} \rightarrow H$

$\boxed{\textbf{Conclusions}}$

- Security definition
  - Suitably rich semantics
- Analysis
  - Has been proven correct