Online Synthesis of Adaptive Side-Channel Attacks Based On Noisy Observations

Lucas Bang, Nicolás Rosner, Tevfik Bultan

Verification Lab (VLab)
Department of Computer Science
University of California Santa Barbara

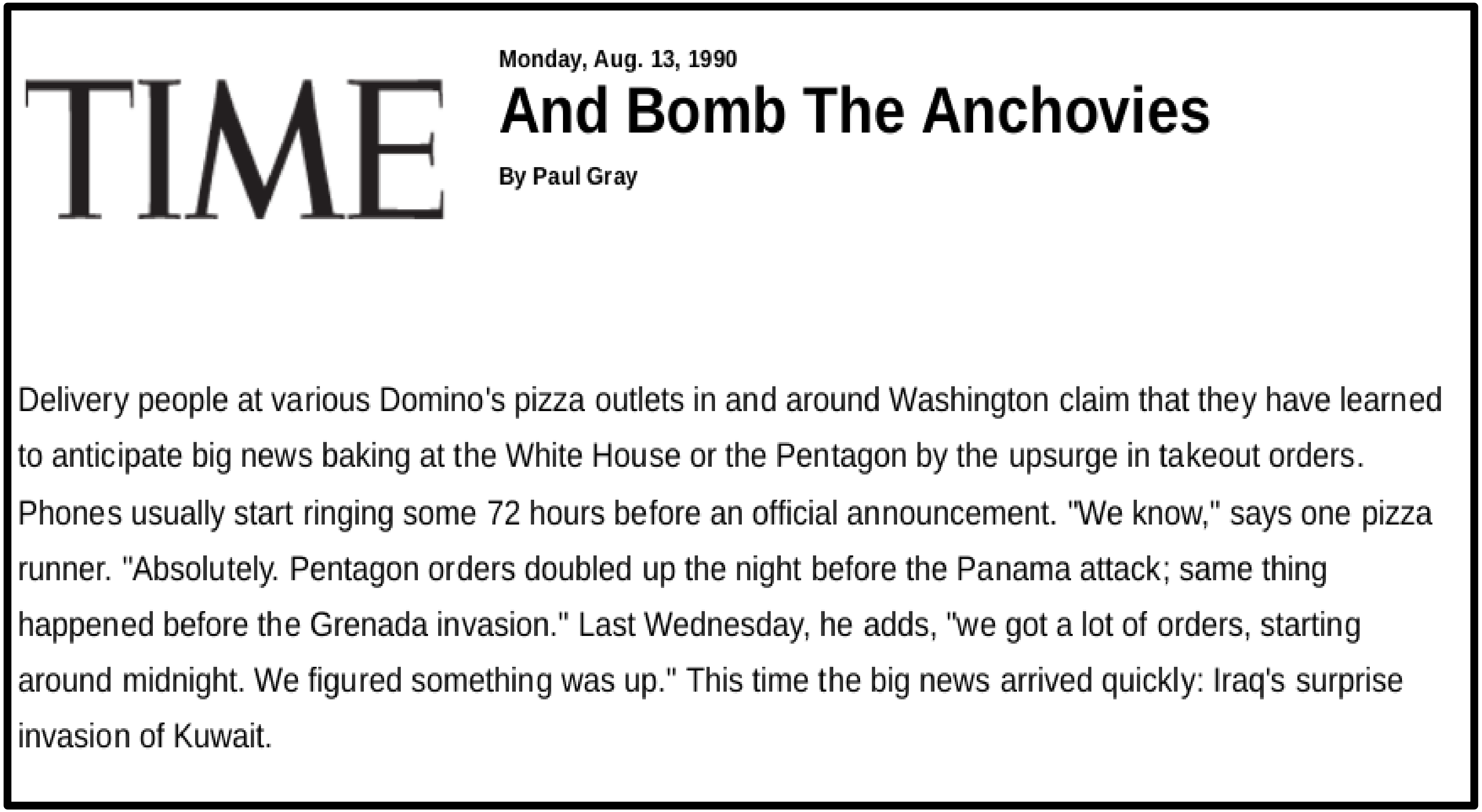
UC Santa Barbara Verification Lab + Collaborations

- Phan, Bang, Pasareanu, Malacaria, Bultan. [CSF 17]
  "Synthesis of Adaptive Side-Channel Attacks."
- Bang, Aydin, Phan, Pasareanu, Bultan. [FSE 2016]
  "String Analysis for Side Channels with Segmented Oracles."
- Bang, Rosner, Bultan. [Euro S&P 2018]
  "Online Synthesis of Adaptive Side-Channel Attacks Based On Noisy Observations."
- Brennan, Tsiskaridze, Rosner, Aydin, Bultan. [FSE 2017]
  "Constraint normalization and parameterized caching for quantitative program analysis."

More Related Work

- Köpf, Basin. [CCS 2007]
  "An information-theoretic model for adaptive side-channel attacks"
- Pasareanu, Phan, Malacaria. [CSF 2016]
  "Multi-run Side-Channel Analysis Using Symbolic Execution and Max-SMT."
- Jia Chen, Yu Feng, Isil Dillig.[CCS 2017]
  "Precise Detection of Side-Channel Vulnerabilities using Quantitative Cartesian Hoare Logic."
- Antonopoulos, Gazzillo, Hicks, Koskinen, Terauchi, Wei. [PLDI 2017]
  "Decomposition instead of self-composition for proving the absence of timing channels."

What is a side channel?



TIME

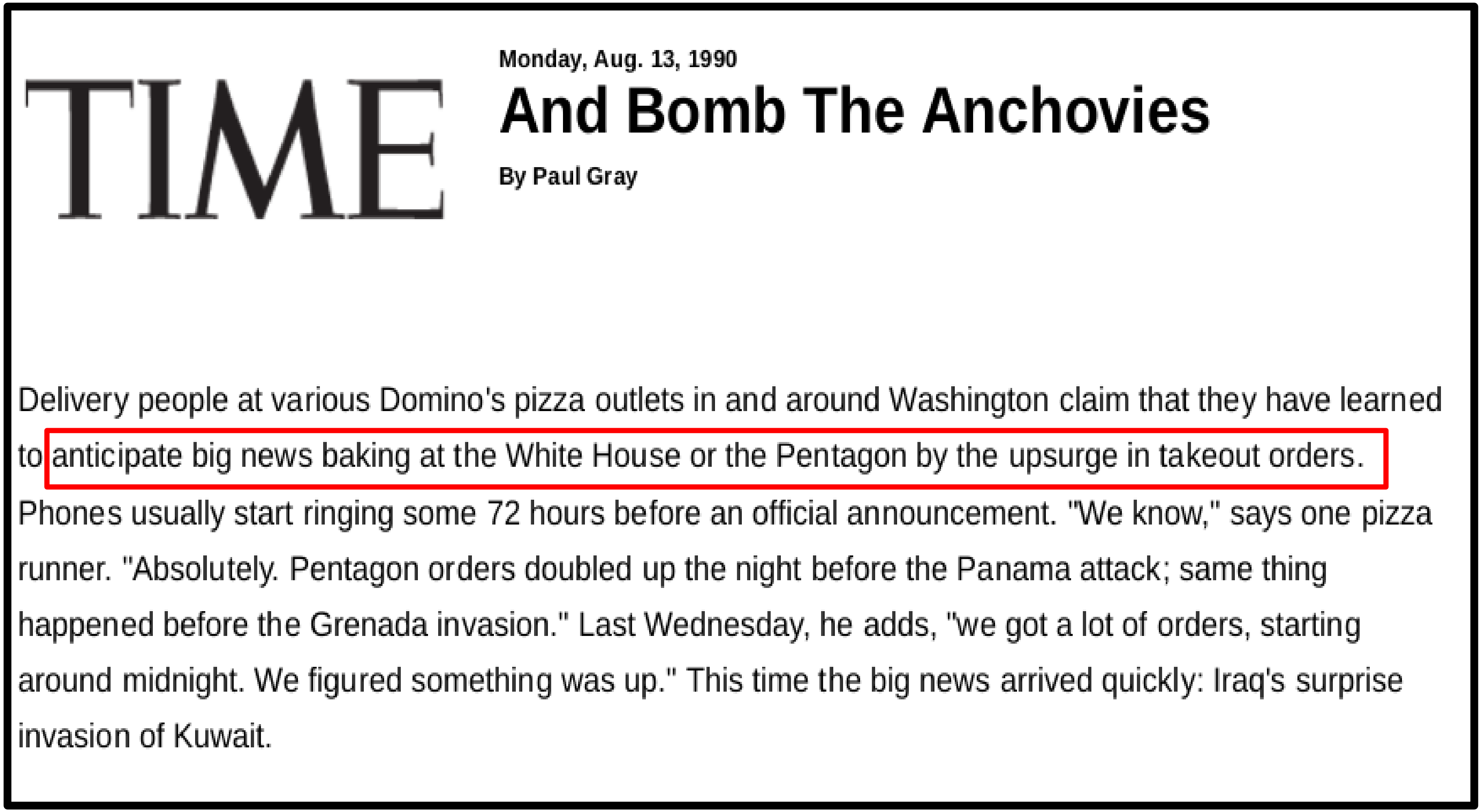
Monday, Aug. 13, 1990

And Bomb The Anchovies

By Paul Gray

Delivery people at various Domino's pizza outlets in and around Washington claim that they have learned to anticipate big news baking at the White House or the Pentagon by the upsurge in takeout orders. Phones usually start ringing some 72 hours before an official announcement. "We know," says one pizza runner. "Absolutely. Pentagon orders doubled up the night before the Panama attack; same thing happened before the Grenada invasion." Last Wednesday, he adds, "we got a lot of orders, starting around midnight. We figured something was up." This time the big news arrived quickly: Iraq's surprise invasion of Kuwait.

What is a side channel?



TIME

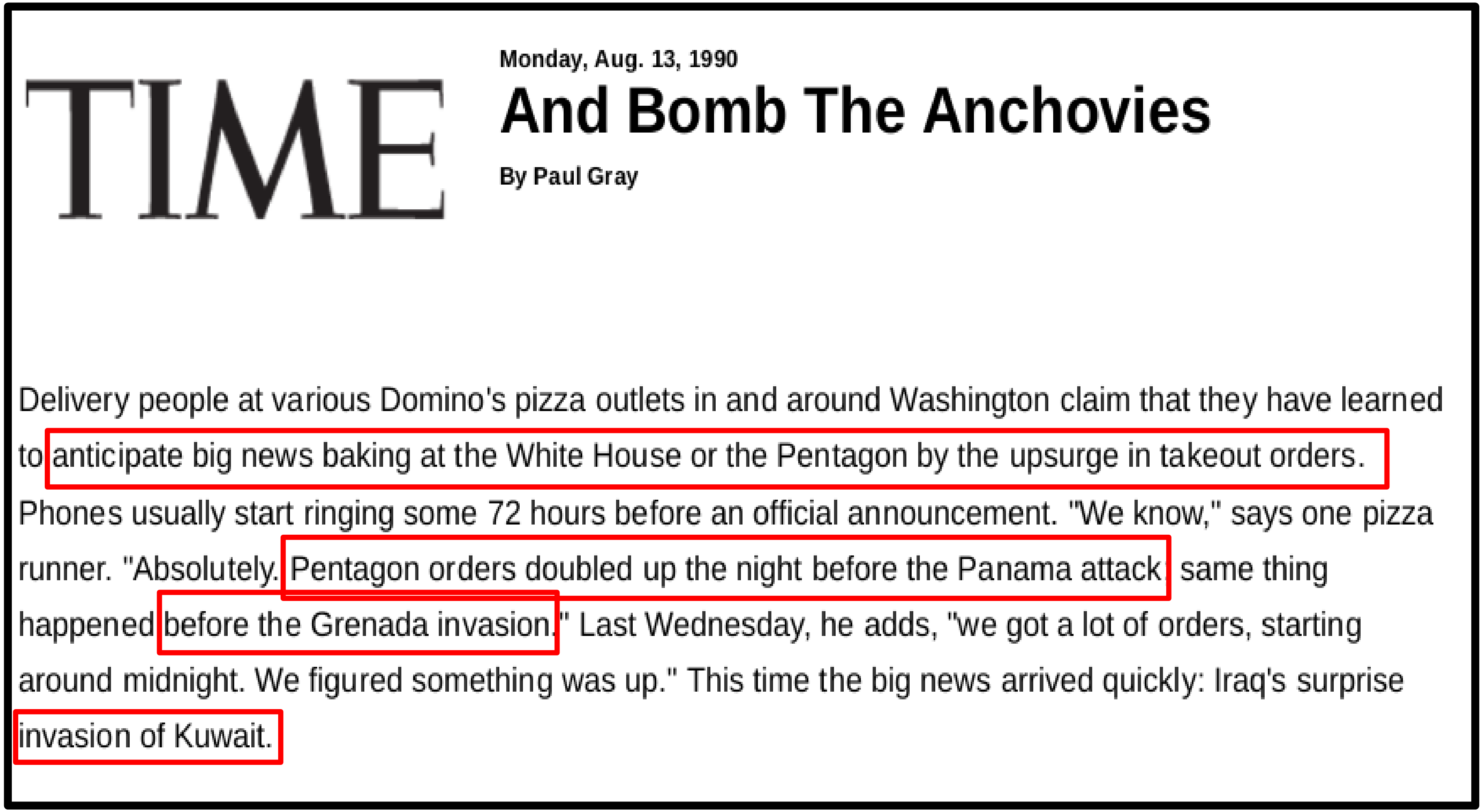
Monday, Aug. 13, 1990

And Bomb The Anchovies

By Paul Gray

Delivery people at various Domino's pizza outlets in and around Washington claim that they have learned to anticipate big news baking at the White House or the Pentagon by the upsurge in takeout orders. Phones usually start ringing some 72 hours before an official announcement. "We know," says one pizza runner. "Absolutely. Pentagon orders doubled up the night before the Panama attack; same thing happened before the Grenada invasion." Last Wednesday, he adds, "we got a lot of orders, starting around midnight. We figured something was up." This time the big news arrived quickly: Iraq's surprise invasion of Kuwait.

What is a side channel?



TIME

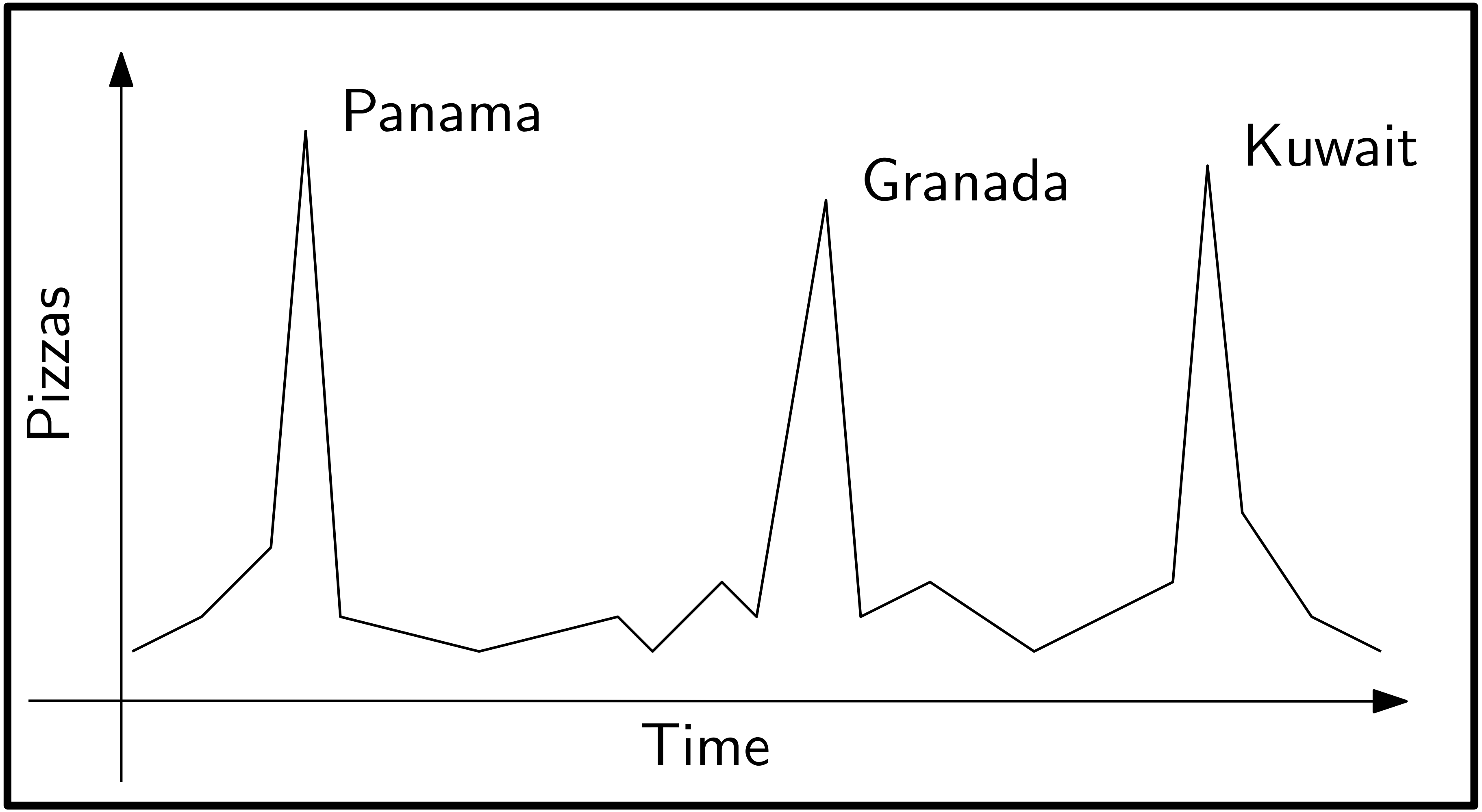
Monday, Aug. 13, 1990

And Bomb The Anchovies

By Paul Gray

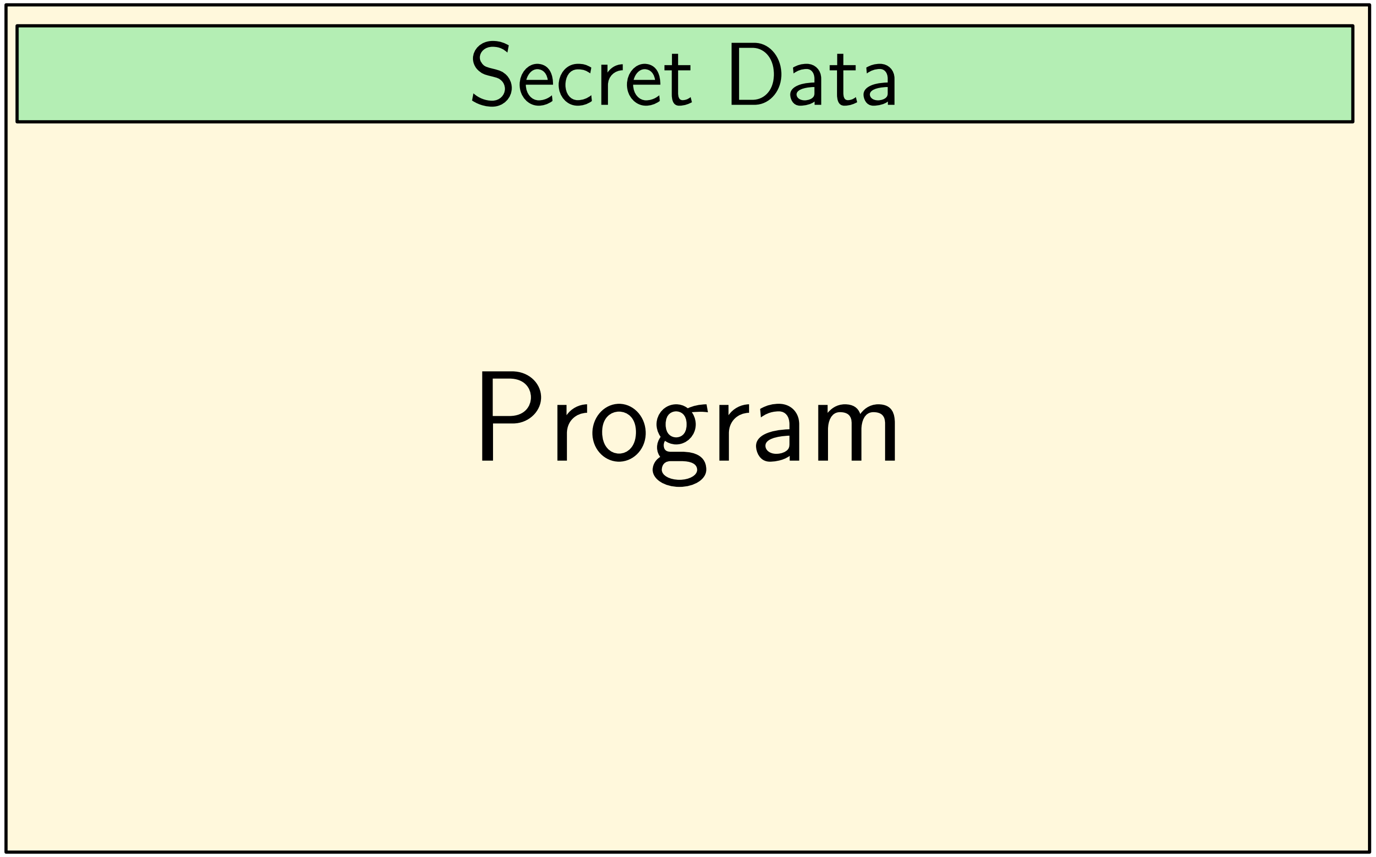
Delivery people at various Domino's pizza outlets in and around Washington claim that they have learned to anticipate big news baking at the White House or the Pentagon by the upsurge in takeout orders. Phones usually start ringing some 72 hours before an official announcement. "We know," says one pizza runner. "Absolutely. Pentagon orders doubled up the night before the Panama attack; same thing happened before the Grenada invasion." Last Wednesday, he adds, "we got a lot of orders, starting around midnight. We figured something was up." This time the big news arrived quickly: Iraq's surprise invasion of Kuwait.

What is a side channel?



Side channel: learn secrets through indirect observation.

secret correlates with observation ⇒ reveal secrets



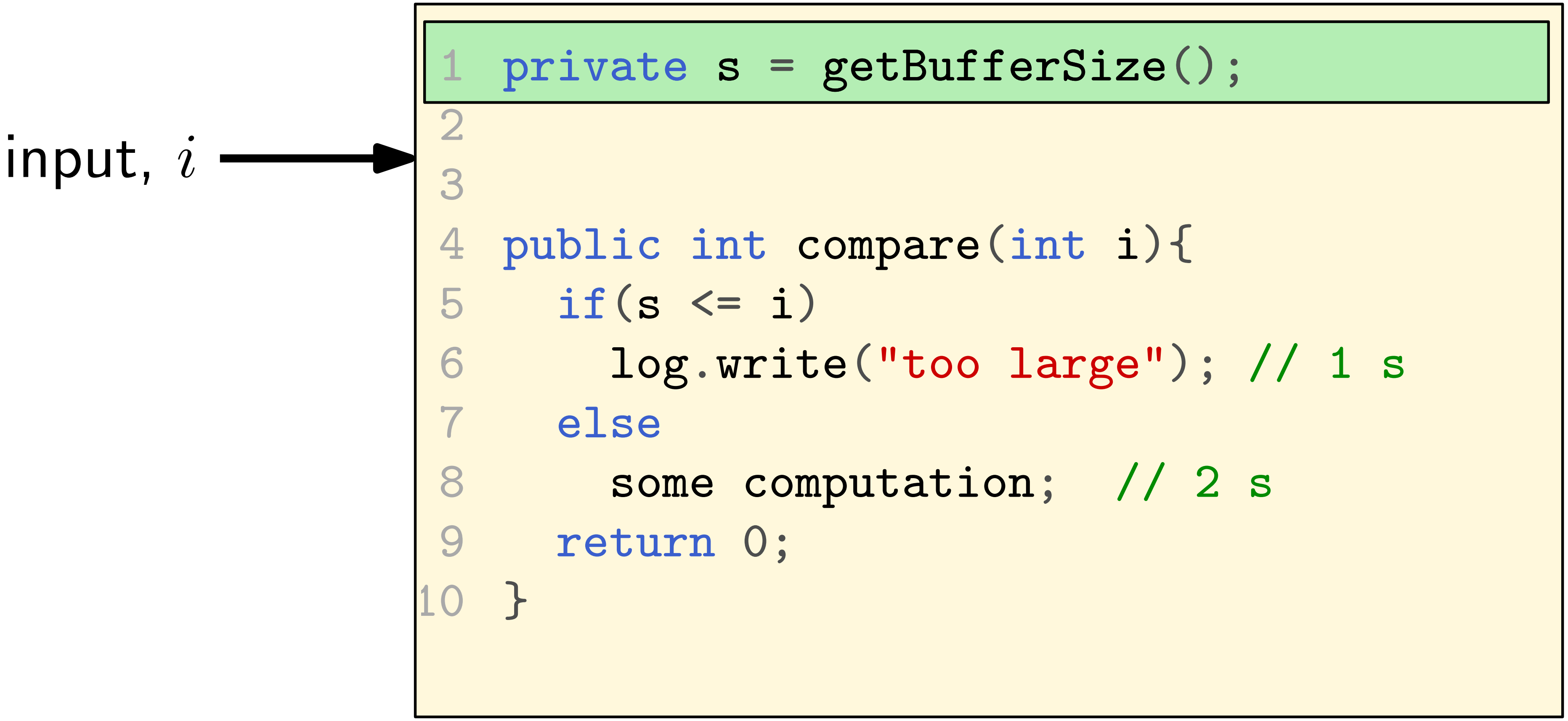
Secret Data

Program

```
private s = getBufferSize();
```

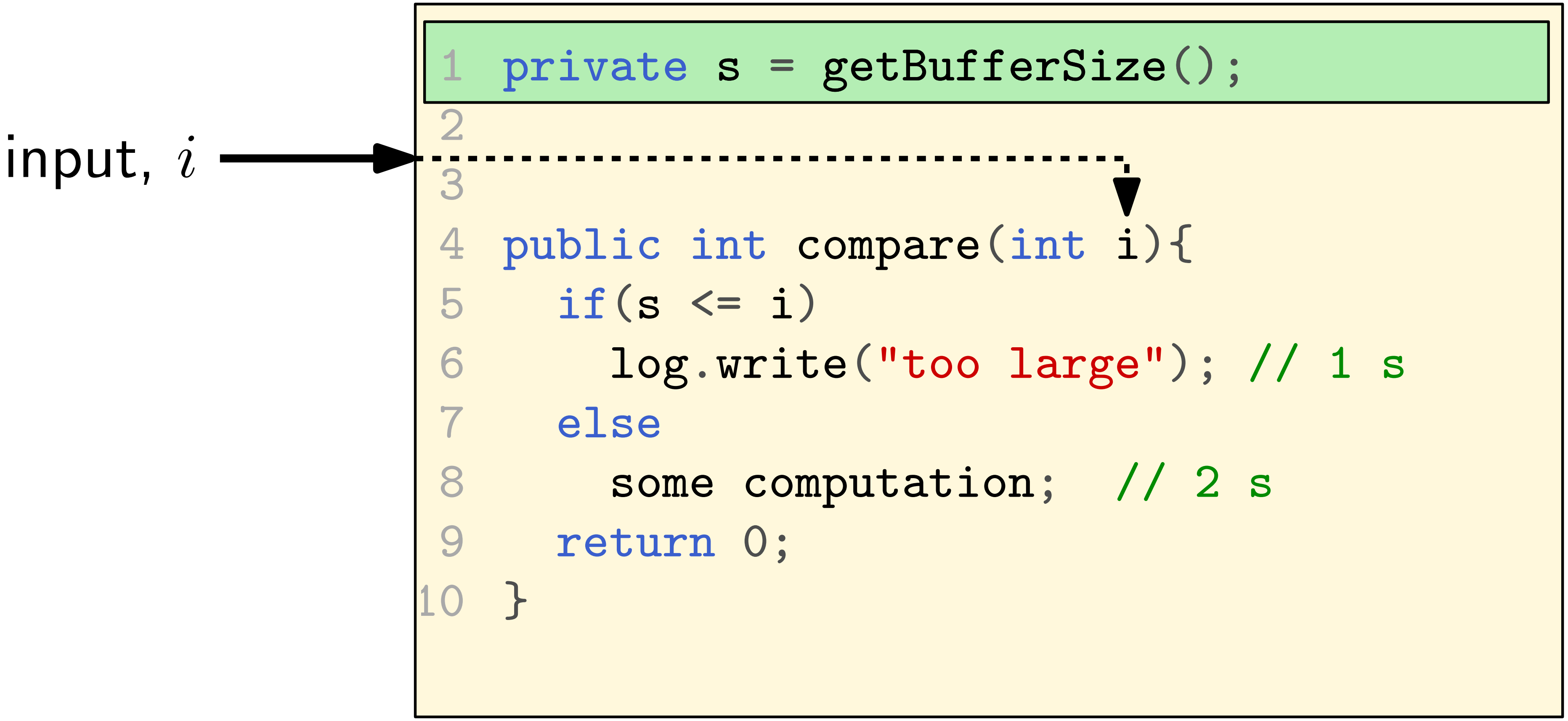
Program

```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```



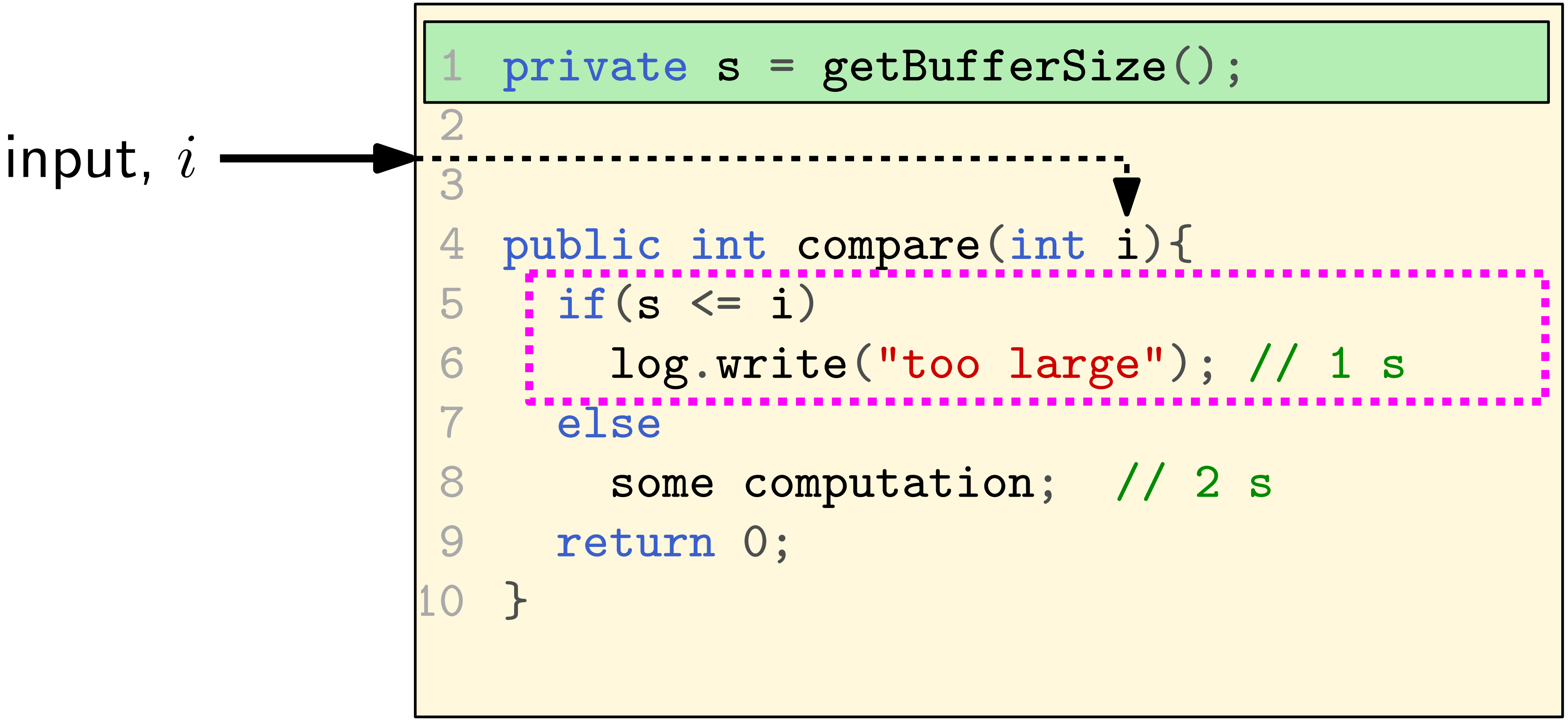
input, i

```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

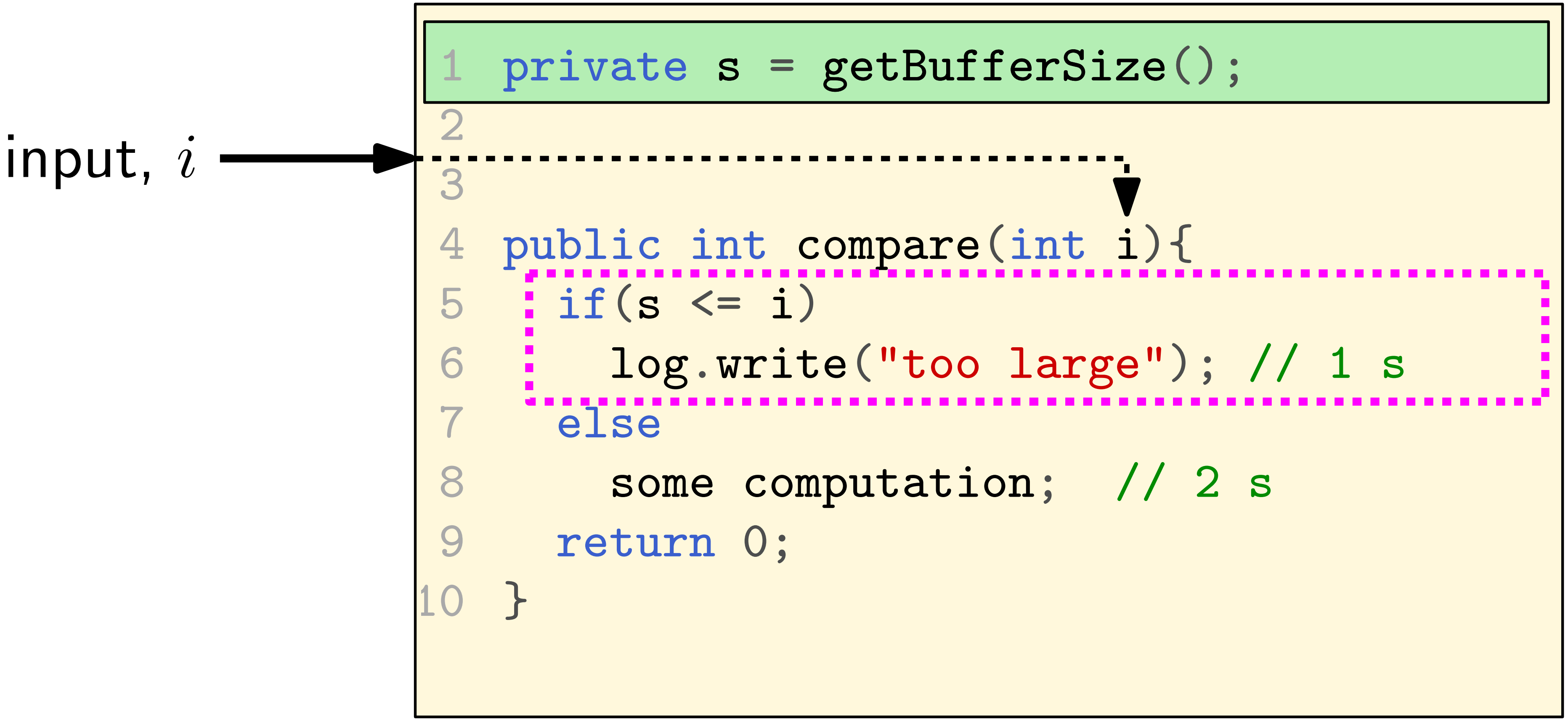


input, i

```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

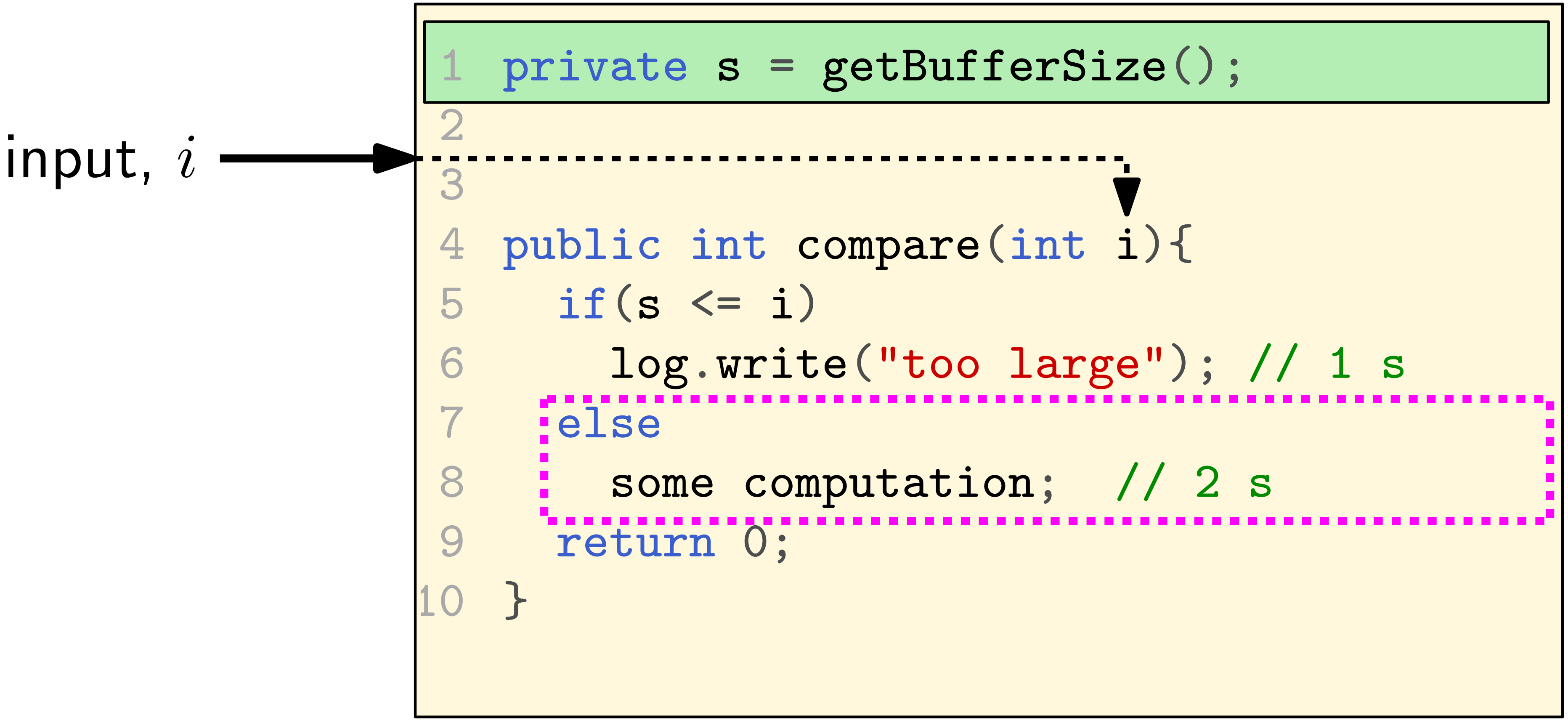


```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation;  // 2 s
 return 0;
}
```



```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

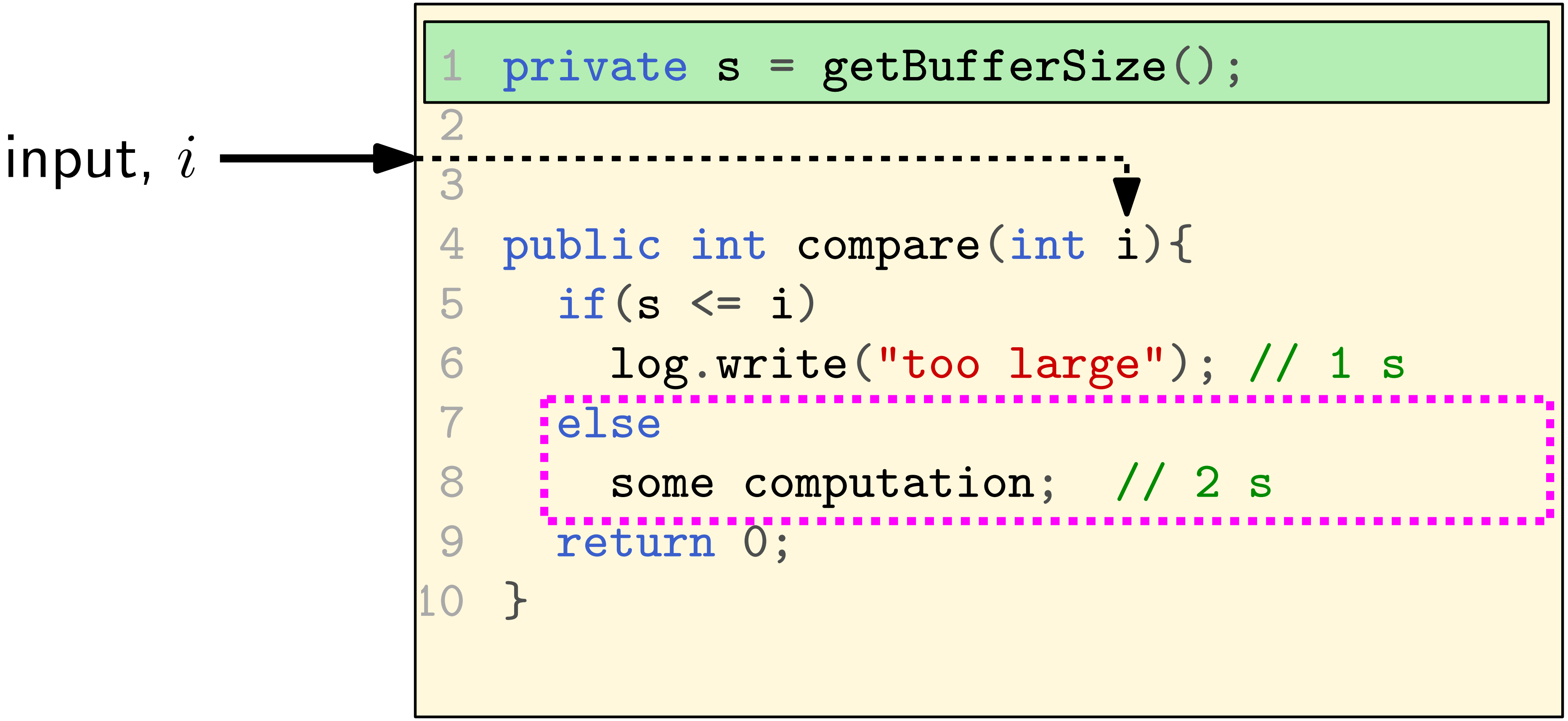
$$s \le i \Rightarrow o = 1$$



input, i

```java
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

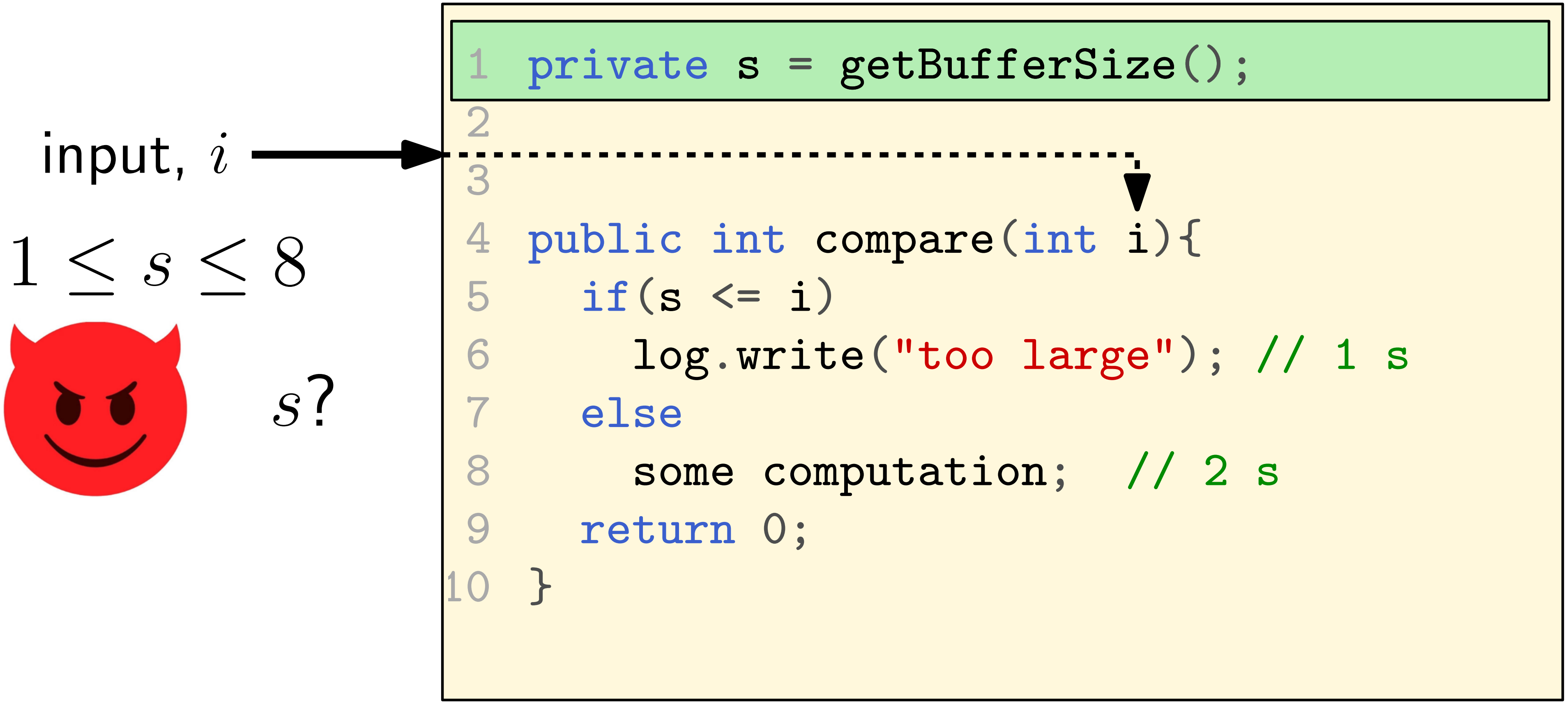
$$s \leq i \Rightarrow o = 1$$



```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

$$s \leq i \Rightarrow o = 1$$

$$s > i \Rightarrow o = 2$$



input, i

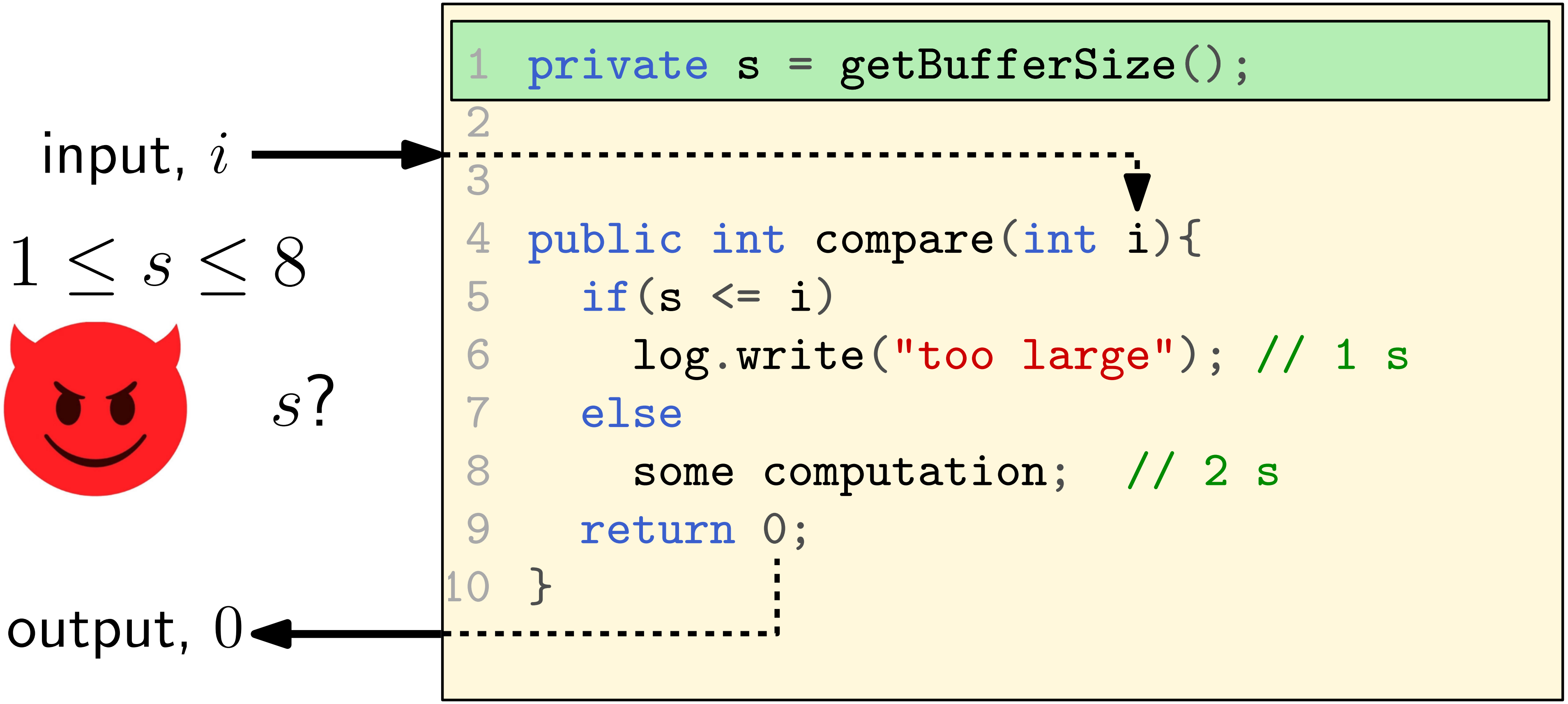
$1 \leq s \leq 8$

$s?$

```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

$$s \leq i \Rightarrow o = 1$$

$$s > i \Rightarrow o = 2$$



input, i

$1 \leq s \leq 8$

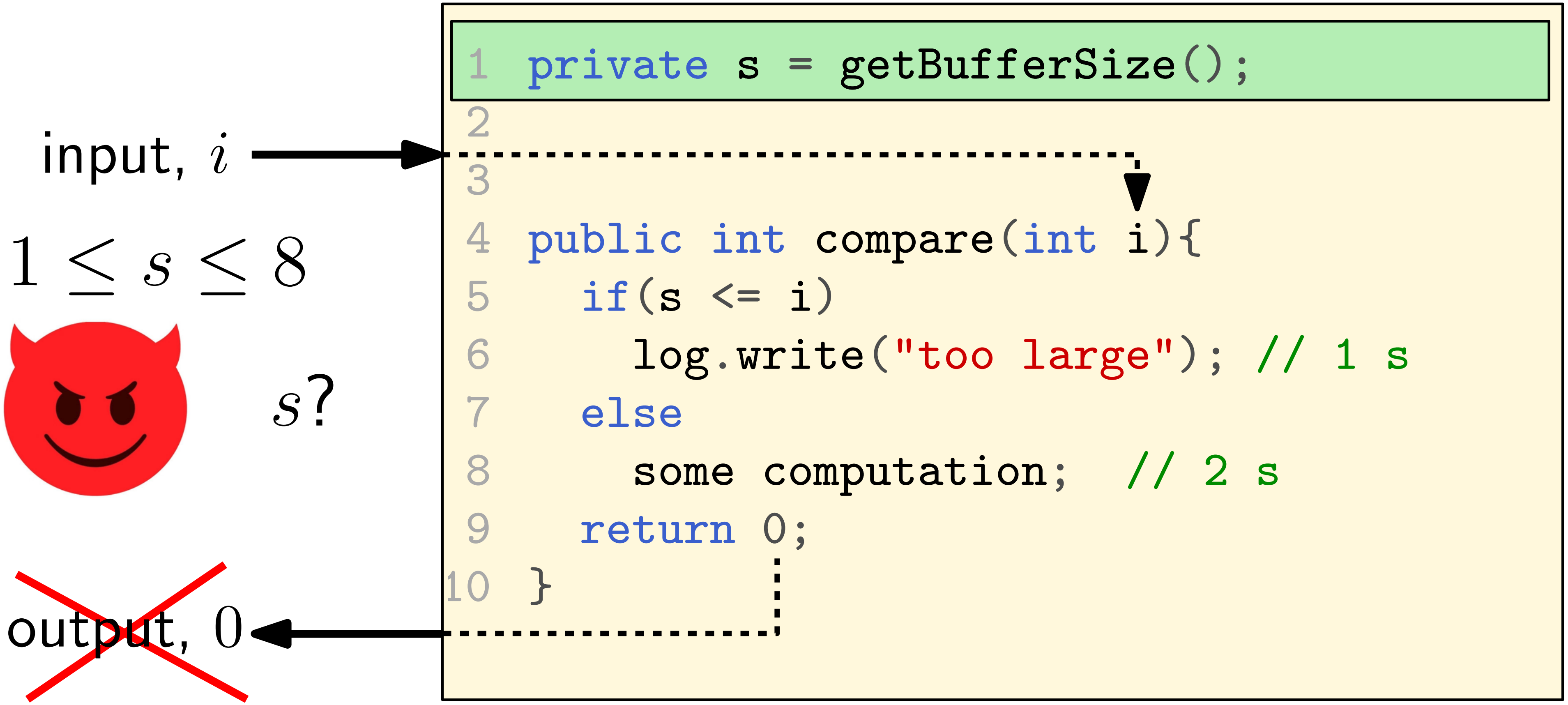
s?

```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

output, 0

$$s \leq i \Rightarrow o = 1$$

$$s > i \Rightarrow o = 2$$



input, i

$1 \le s \le 8$

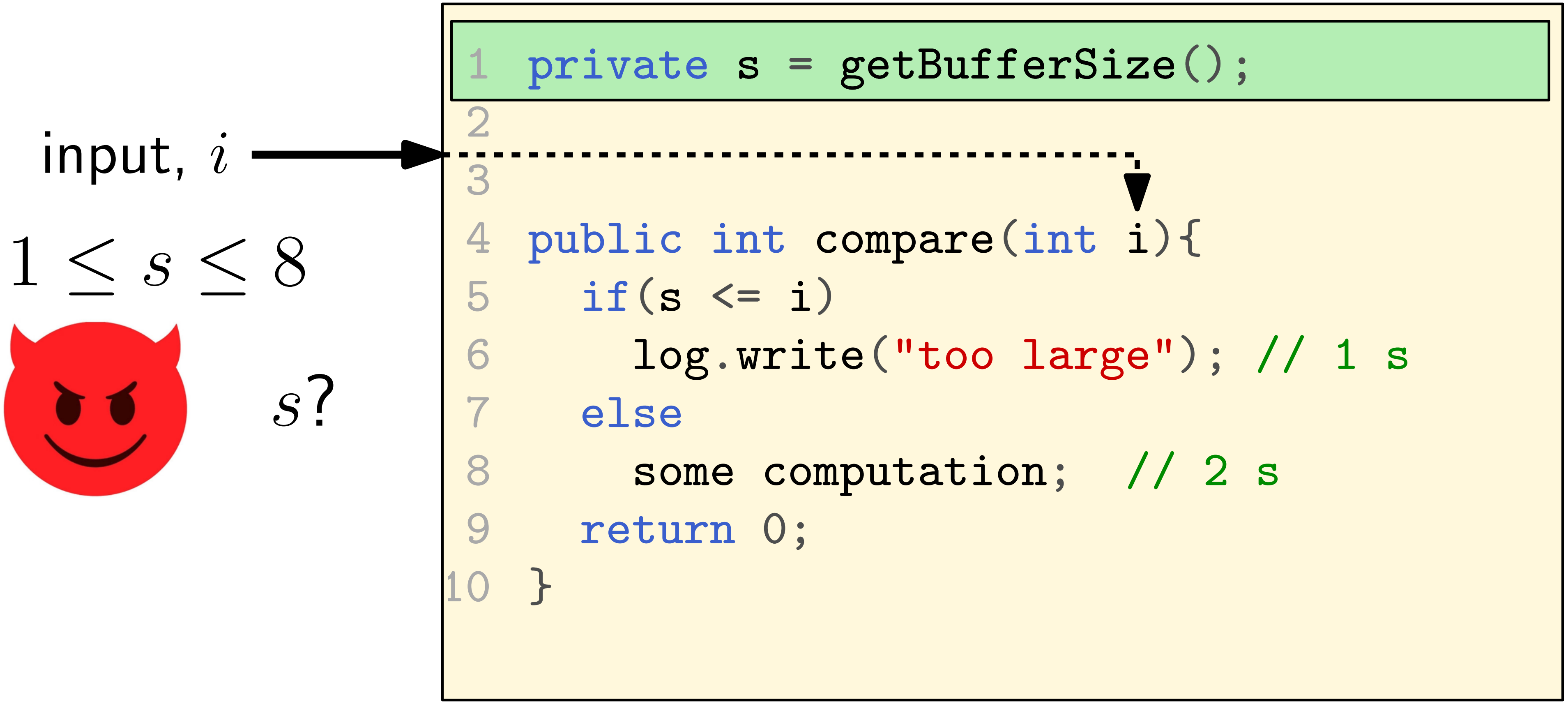
s?

~~output~~, 0

```java
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

$$s \le i \Rightarrow o = 1$$

$$s > i \Rightarrow o = 2$$



input, i

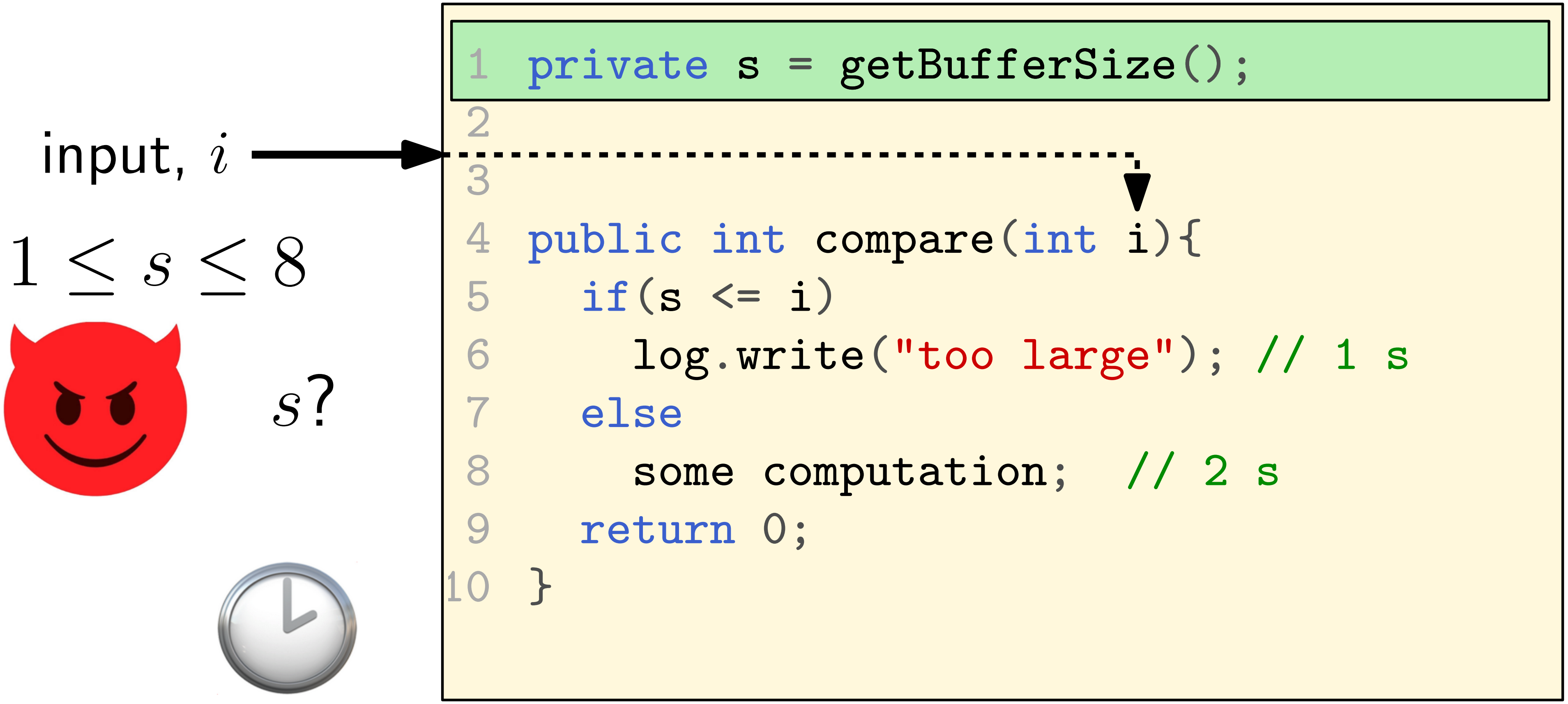
$1 \leq s \leq 8$

😈 $s?$

```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

$$s \leq i \Rightarrow o = 1$$

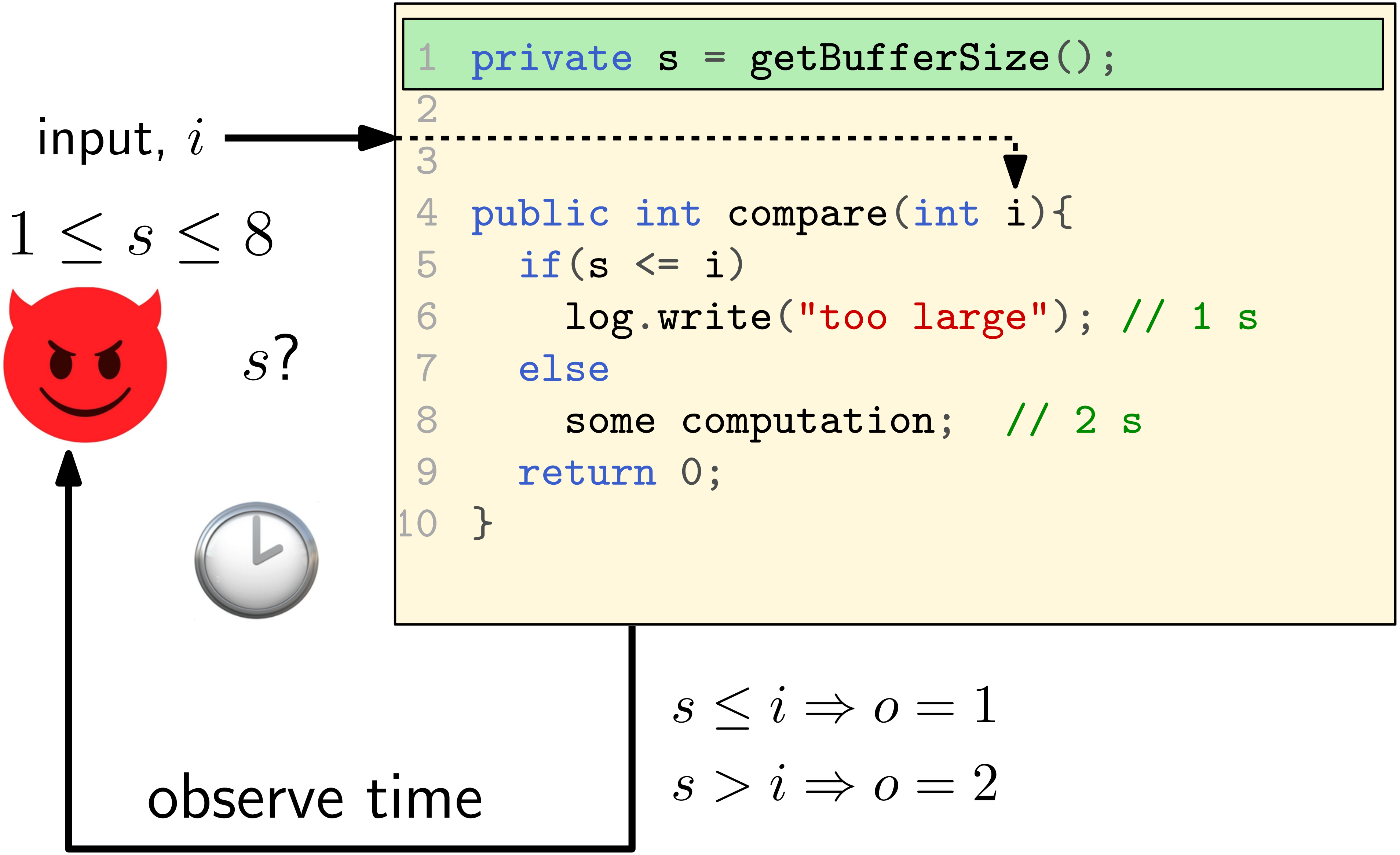
$$s > i \Rightarrow o = 2$$



```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

$$s \leq i \Rightarrow o = 1$$

$$s > i \Rightarrow o = 2$$



input, i

$1 \leq s \leq 8$

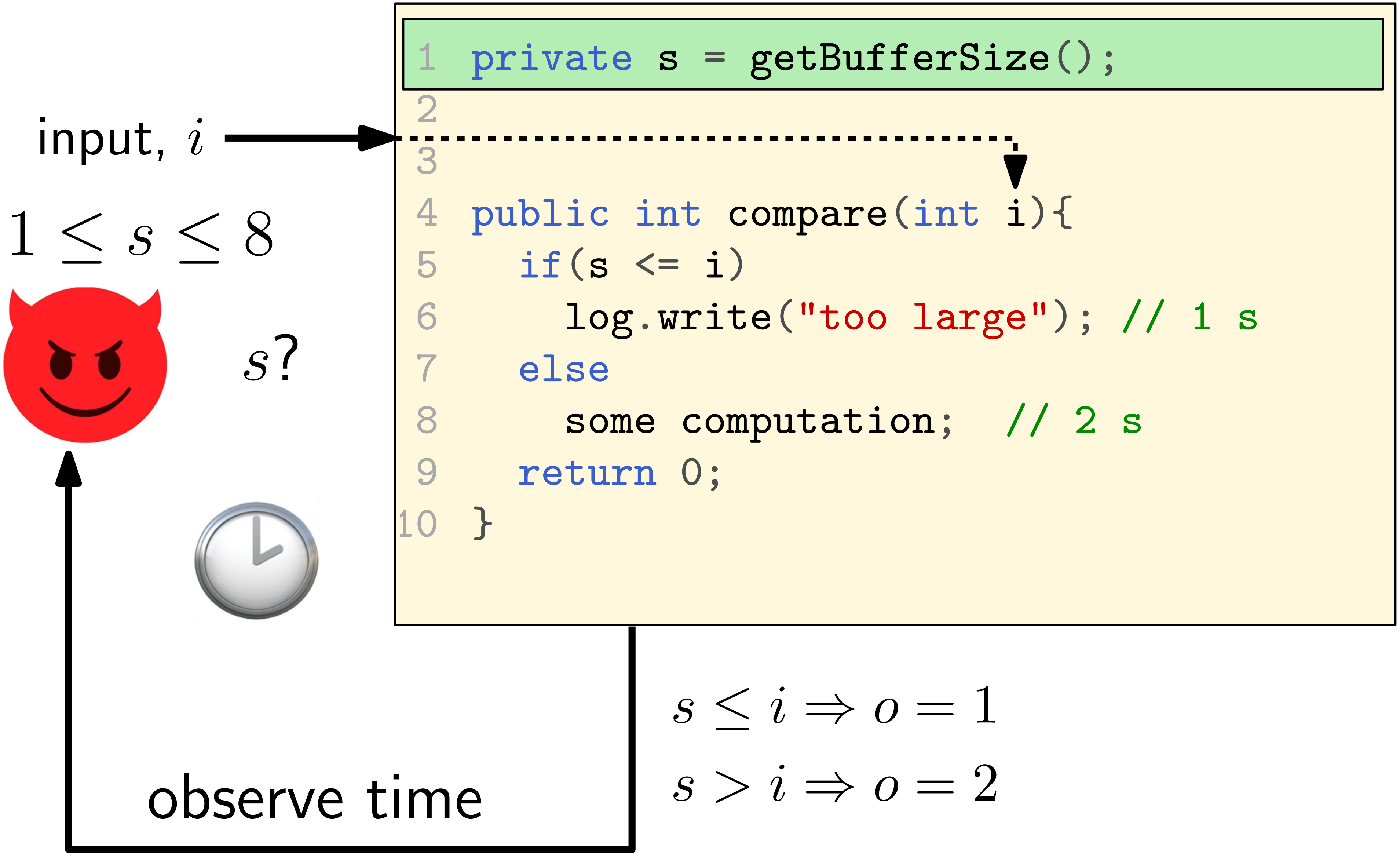
s?

```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

$s \leq i \Rightarrow o = 1$

$s > i \Rightarrow o = 2$

observe time



input, i

$1 \leq s \leq 8$

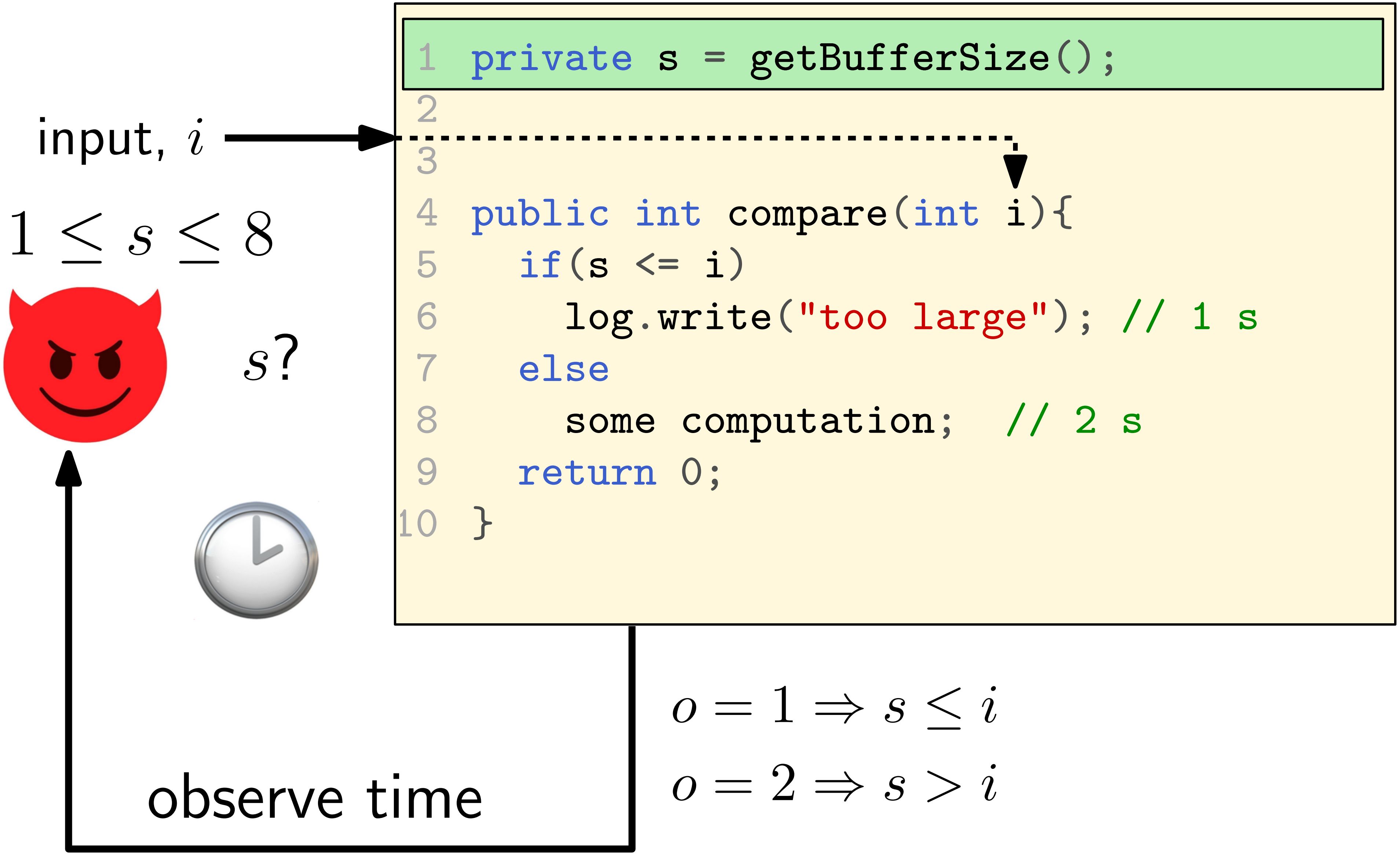
s?

```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

$s \leq i \Rightarrow o = 1$

$s > i \Rightarrow o = 2$

observe time



input, i

$1 \leq s \leq 8$

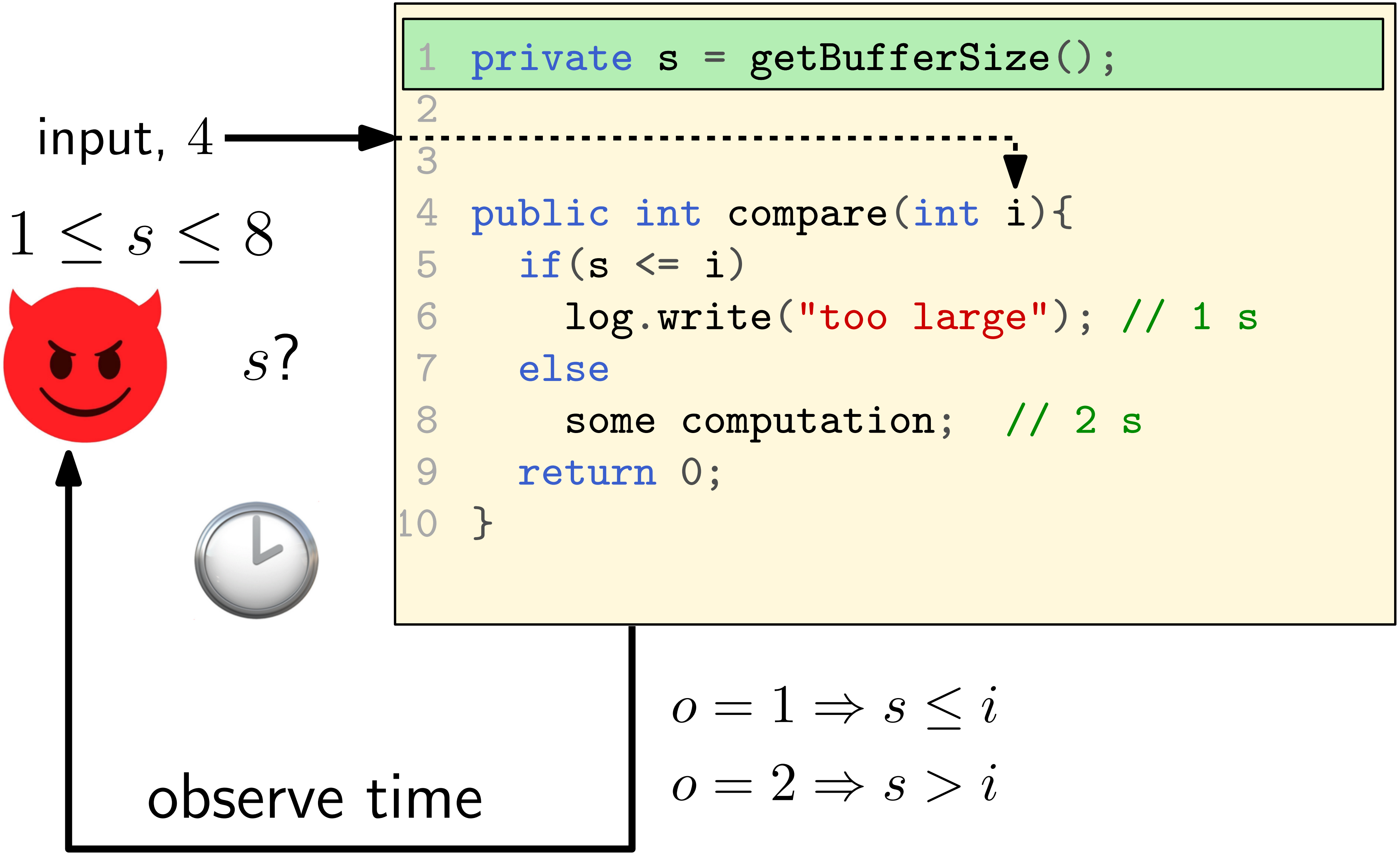
s?

```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

observe time

$o = 1 \Rightarrow s \leq i$

$o = 2 \Rightarrow s > i$



input, 4

$1 \leq s \leq 8$

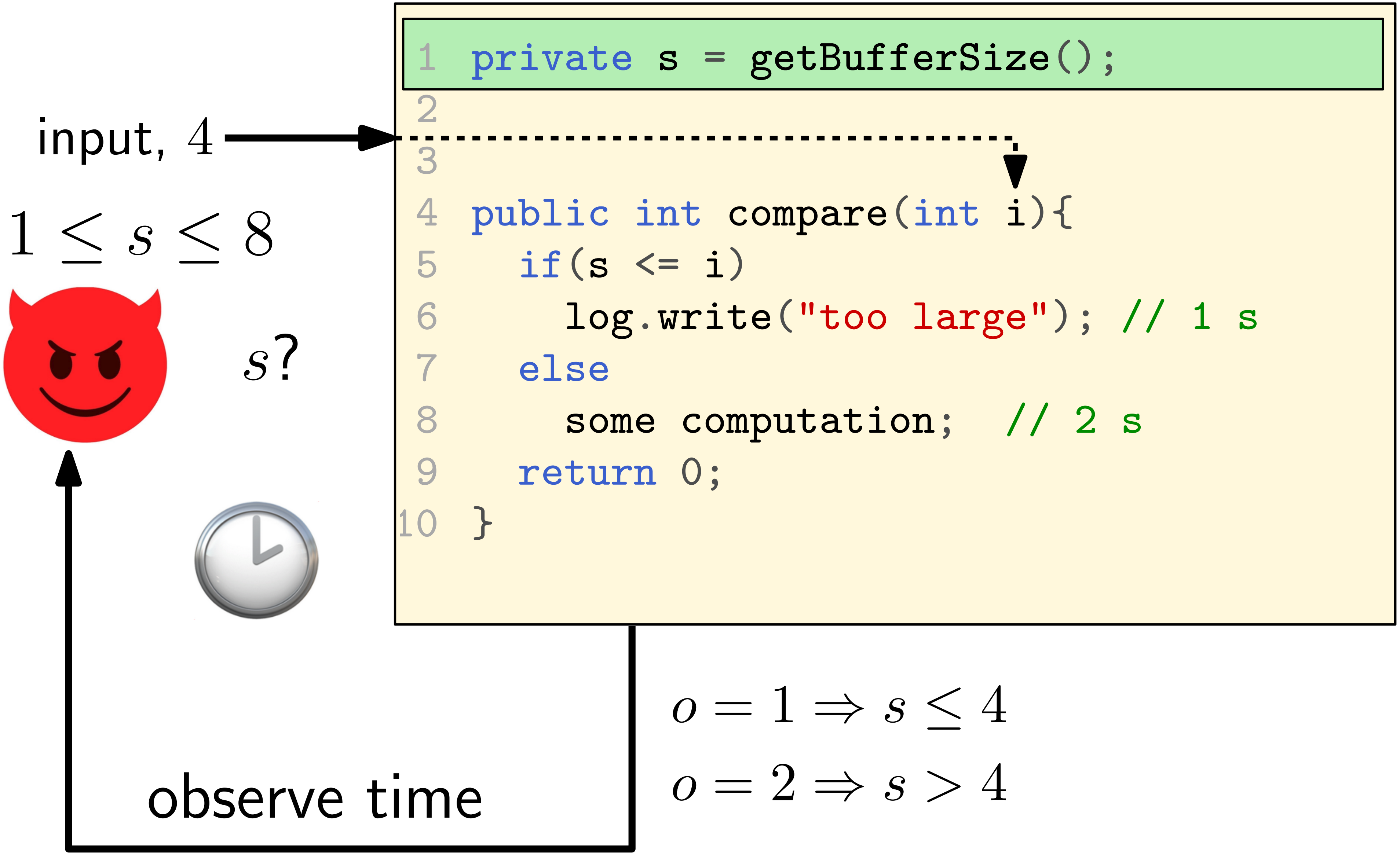
$s?$

```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

observe time

$o = 1 \Rightarrow s \leq i$

$o = 2 \Rightarrow s > i$



input, 4

$1 \leq s \leq 8$

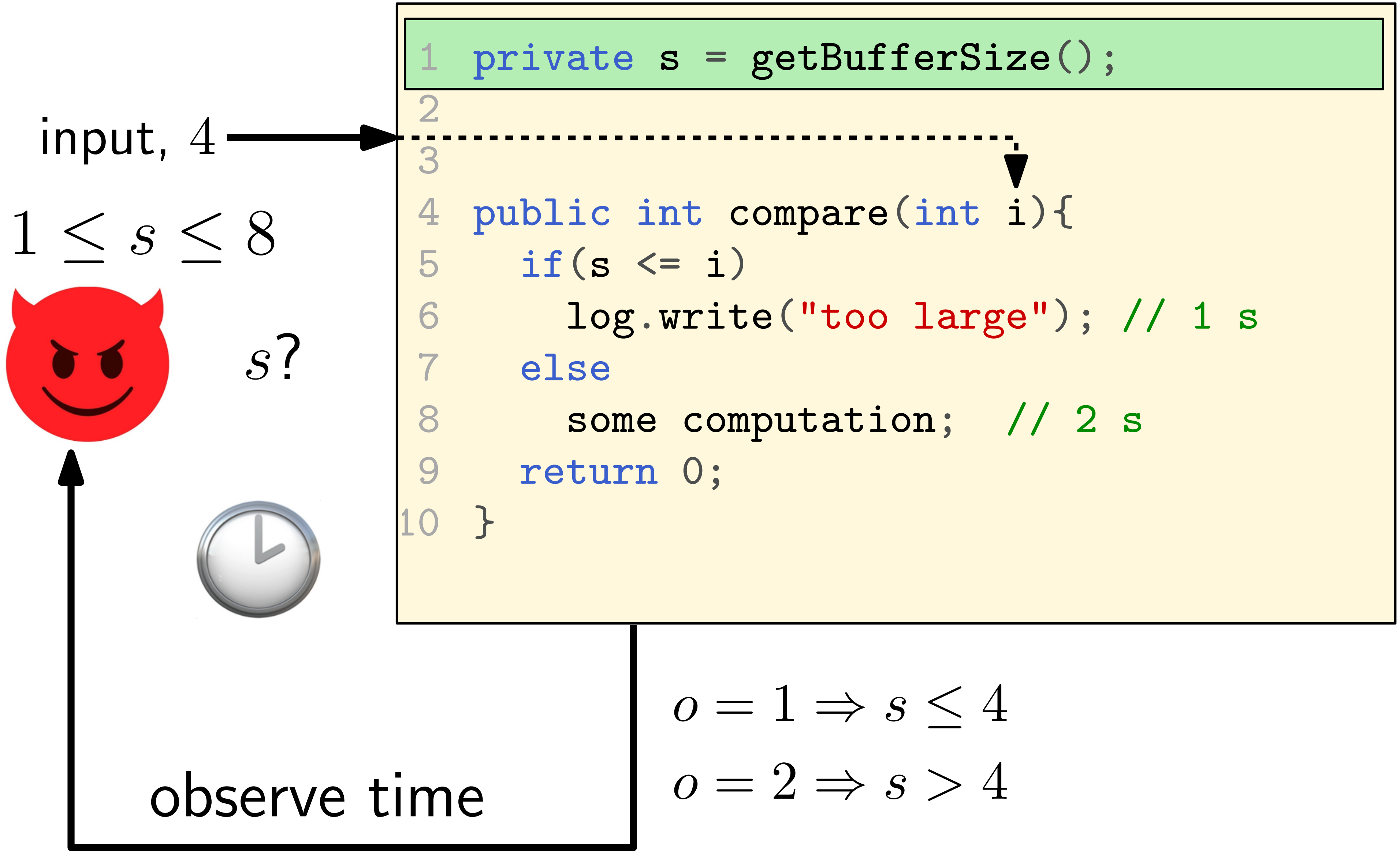
$s?$

```java
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

observe time

$$o = 1 \Rightarrow s \leq 4$$

$$o = 2 \Rightarrow s > 4$$



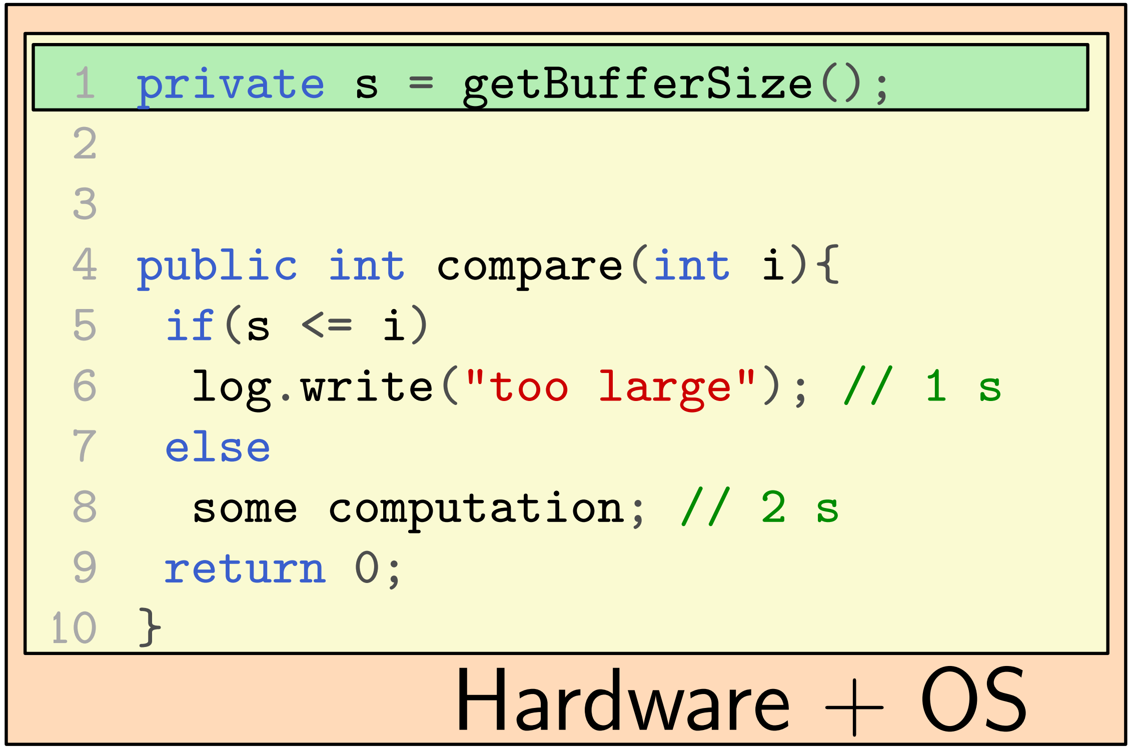
```
private s = getBufferSize();


public int compare(int i){
  if(s <= i)
    log.write("too large"); // 1 s
  else
    some computation;  // 2 s
  return 0;
}
```

$o = 1 \Rightarrow s \leq 4$

$o = 2 \Rightarrow s > 4$

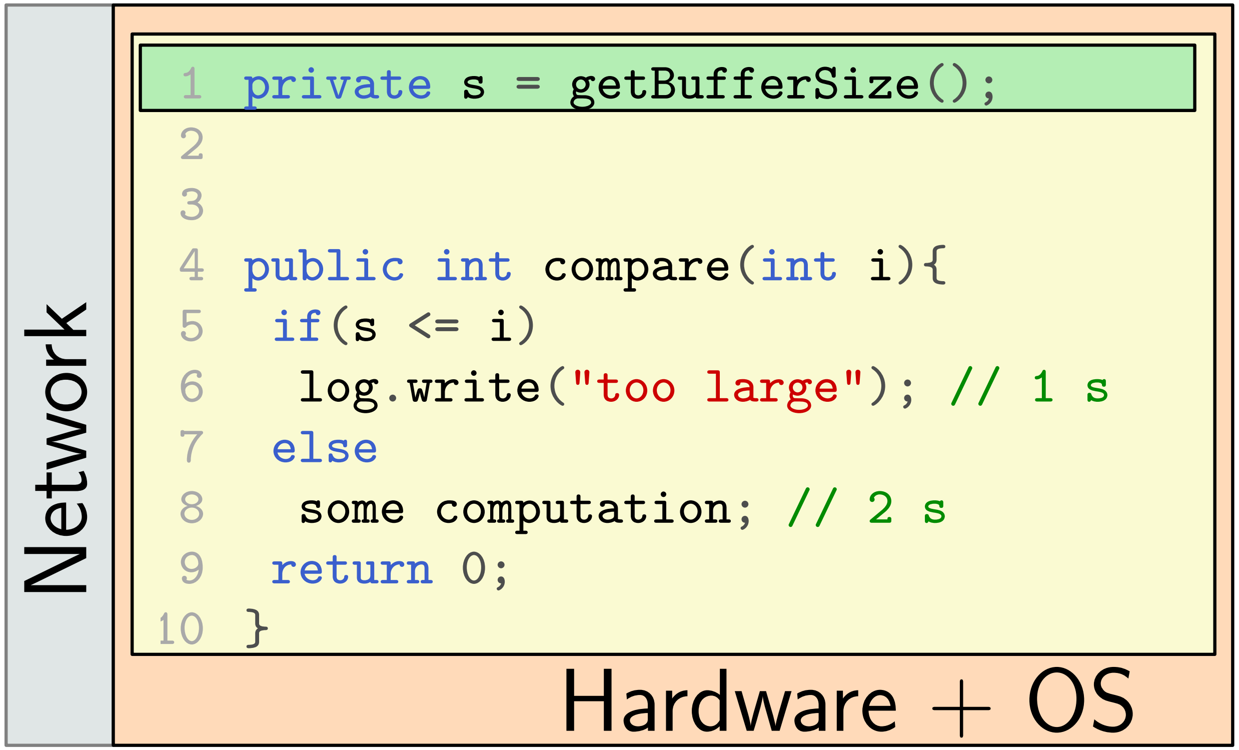
Attacker can binary search on s using i and o.

```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```



```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```

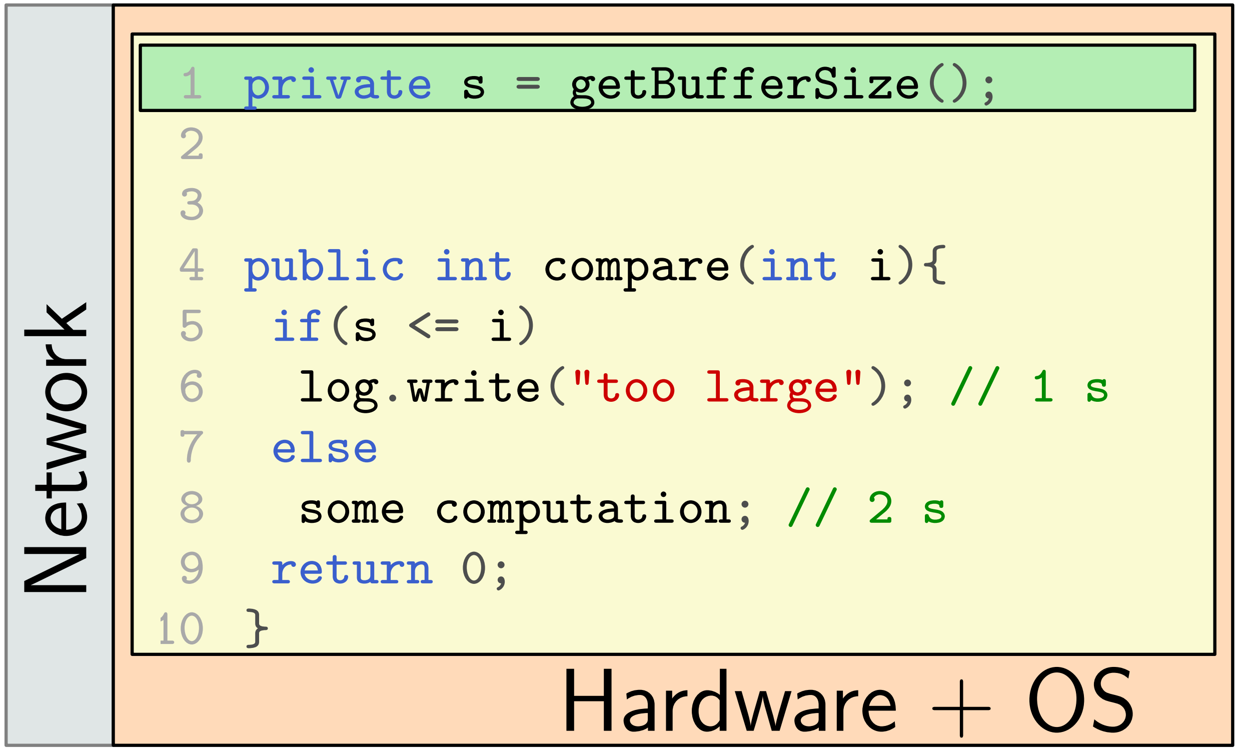
Hardware + OS



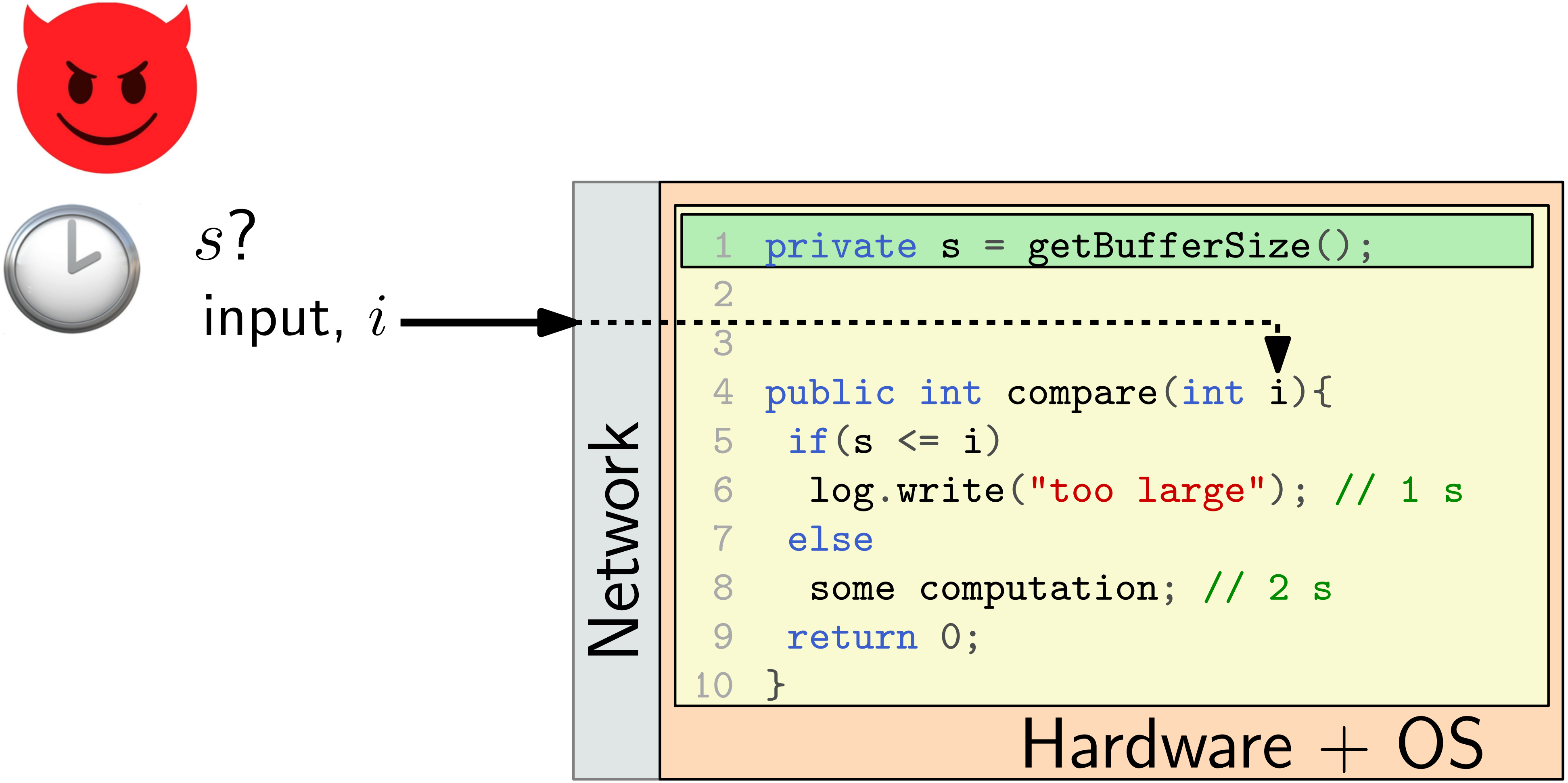
```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```

😈

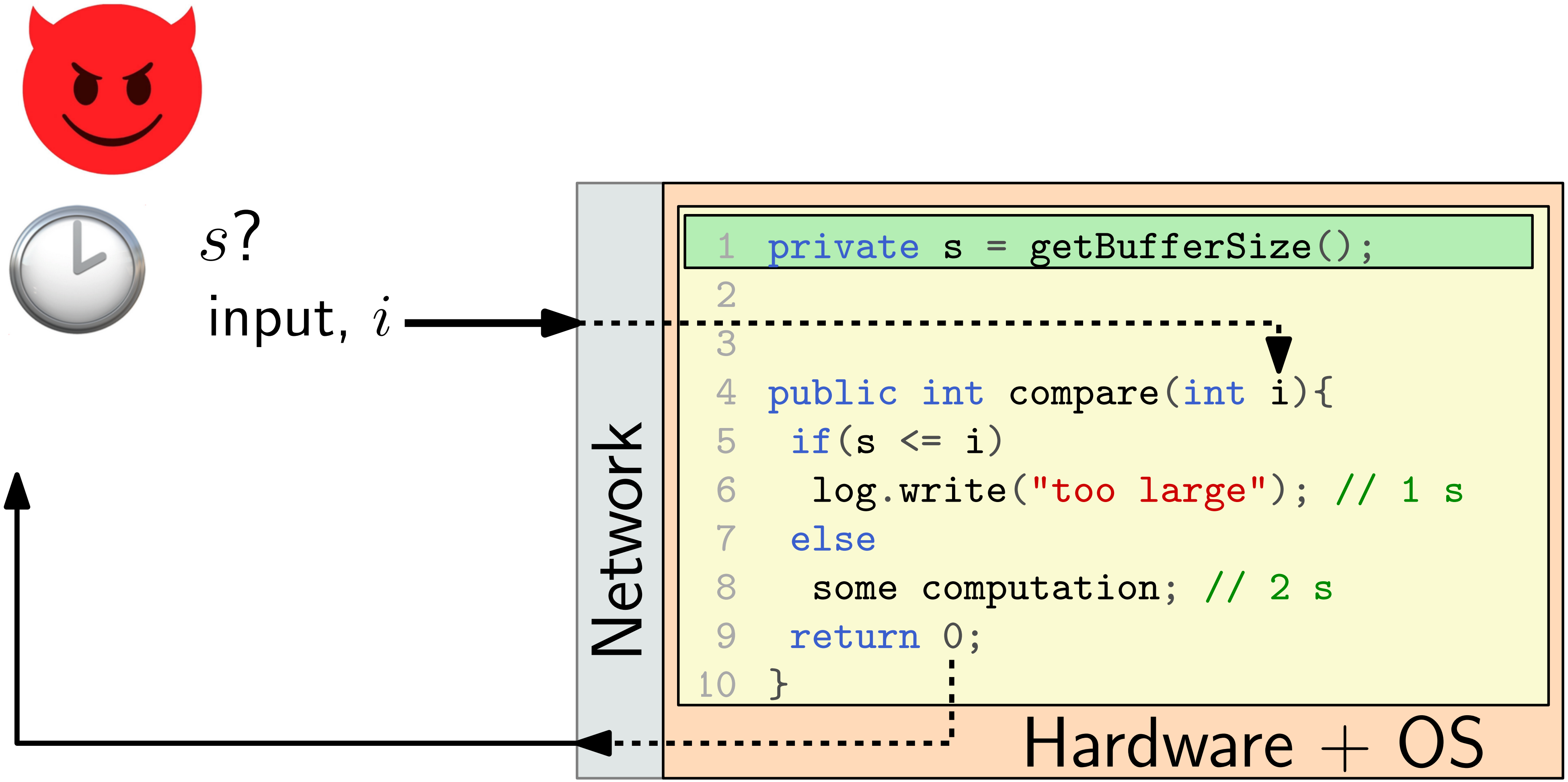
🕑 s?



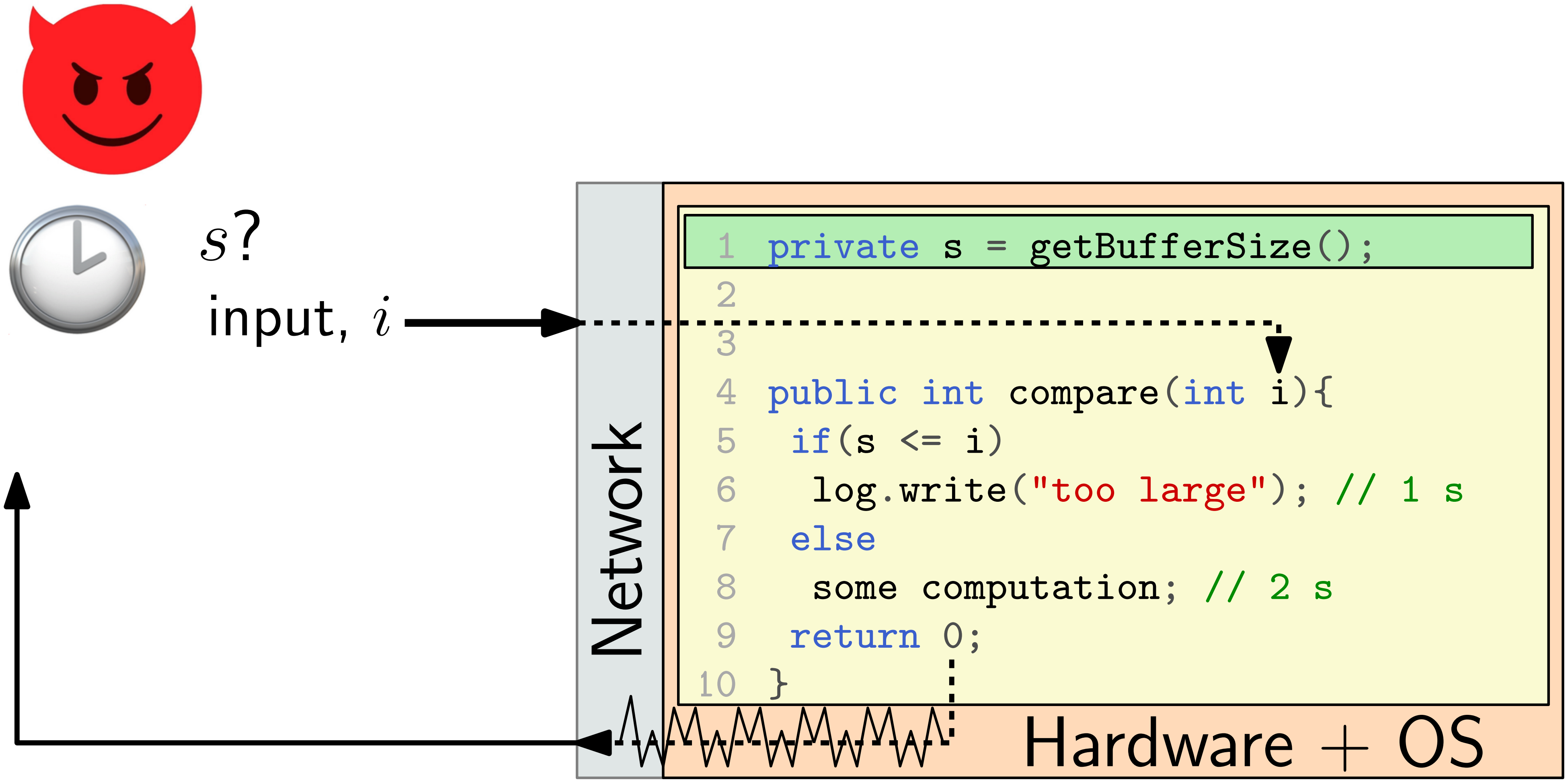
```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```



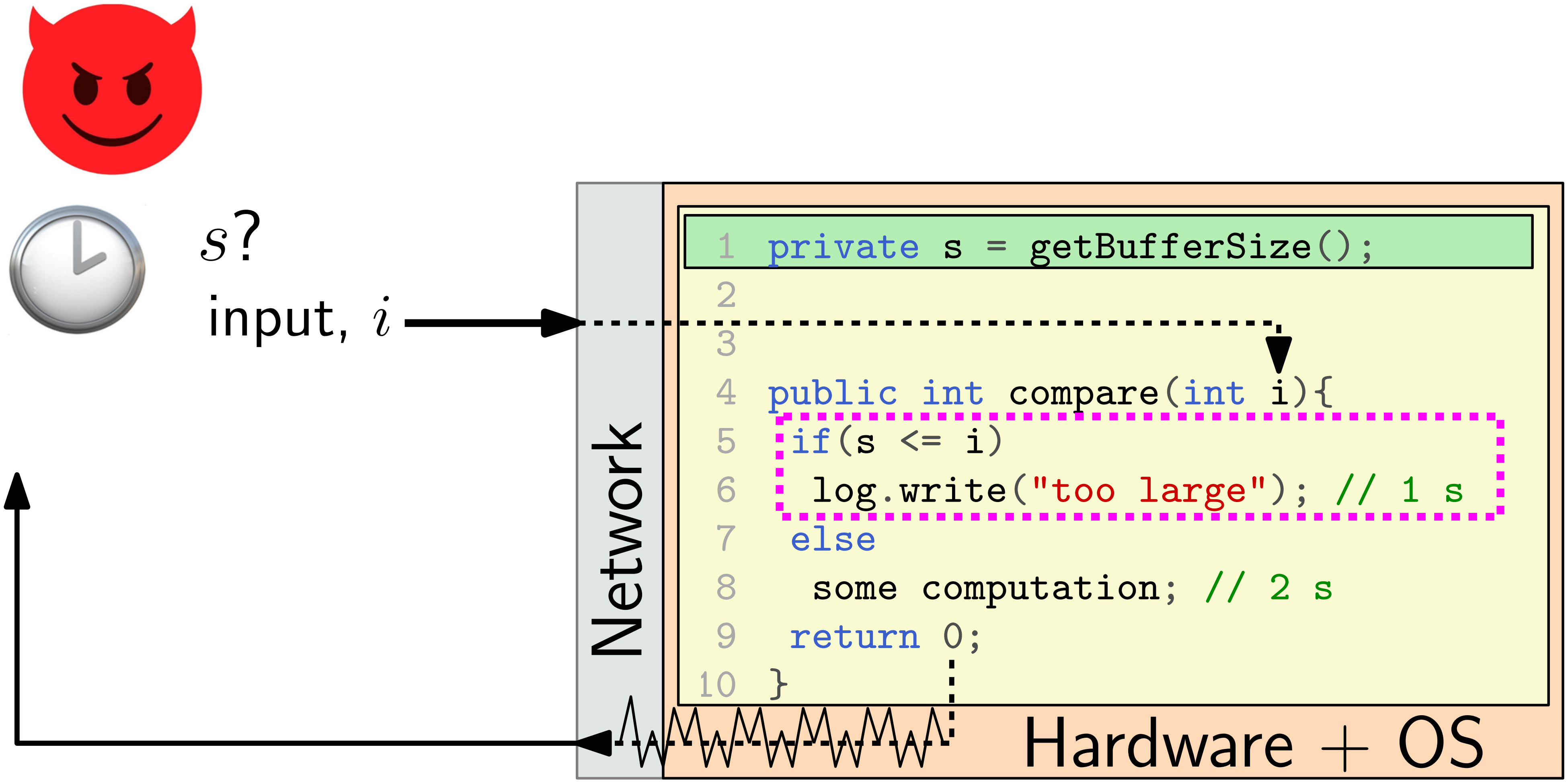
```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```



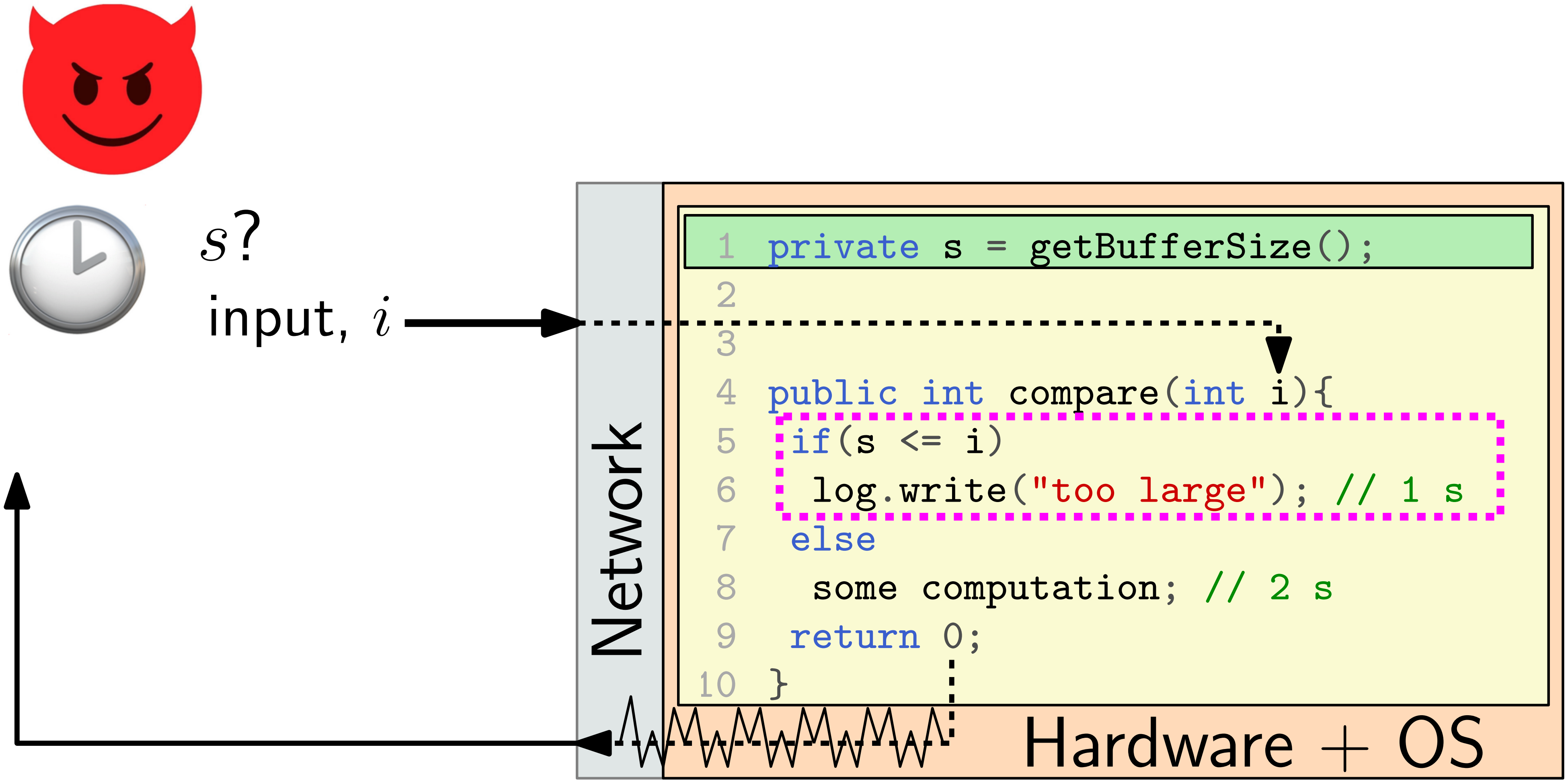
```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```



```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```

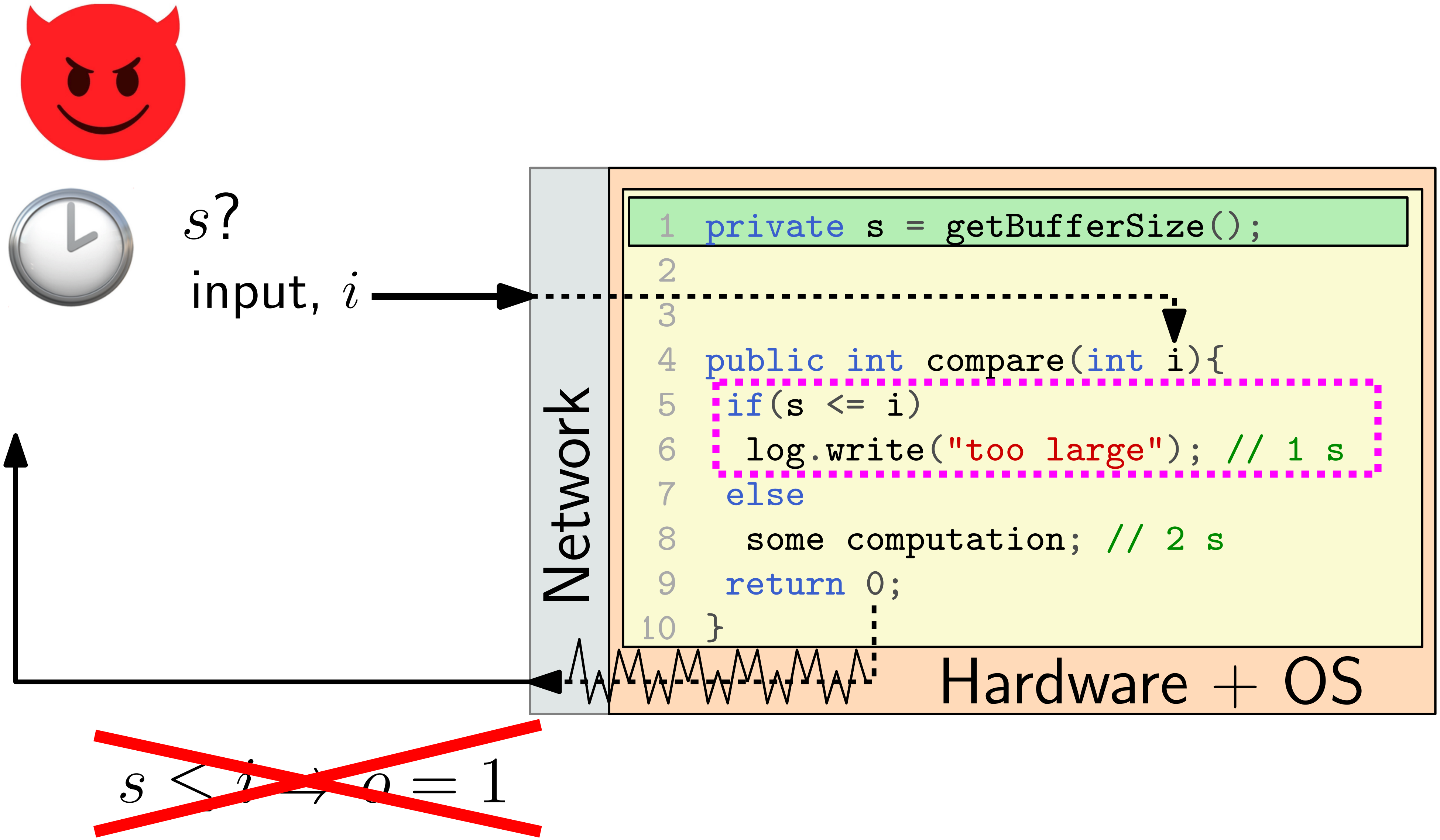


```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```



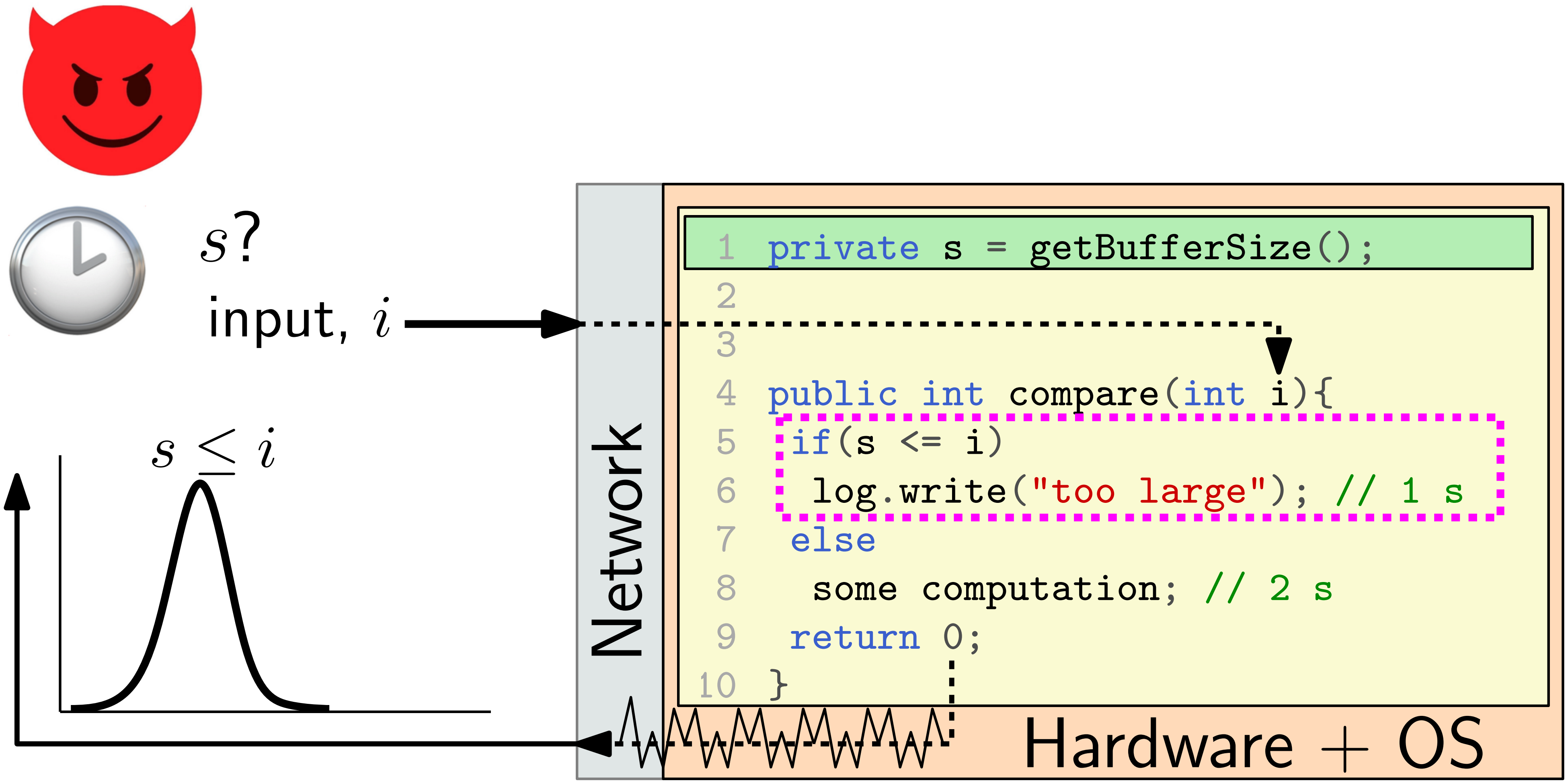
```java
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```

$$s \leq i \Rightarrow o = 1$$



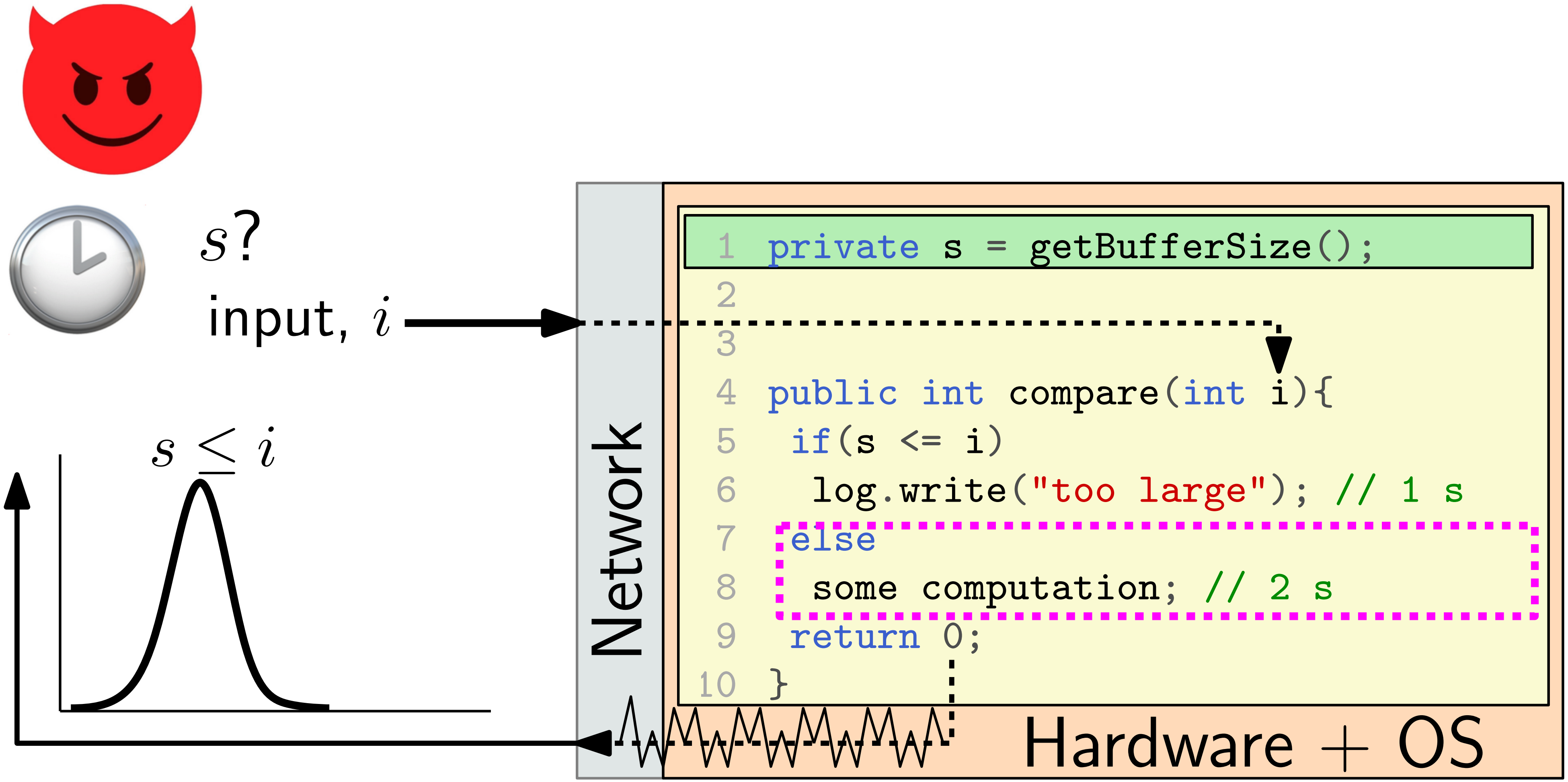
```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```

$\cancel{s < i \to o = 1}$

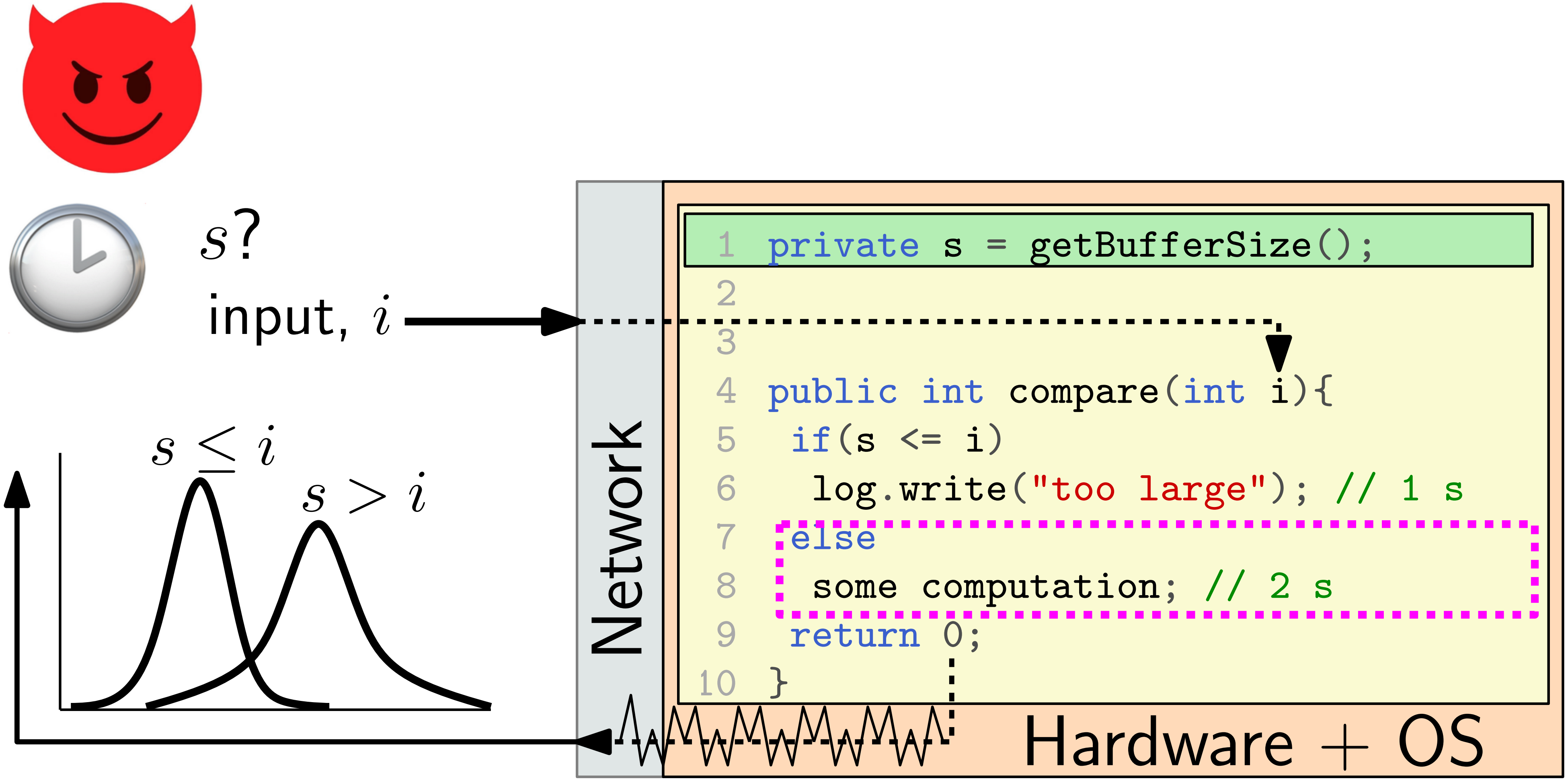


```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```

~~$s \leq i \rightarrow o = 1$~~



```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```



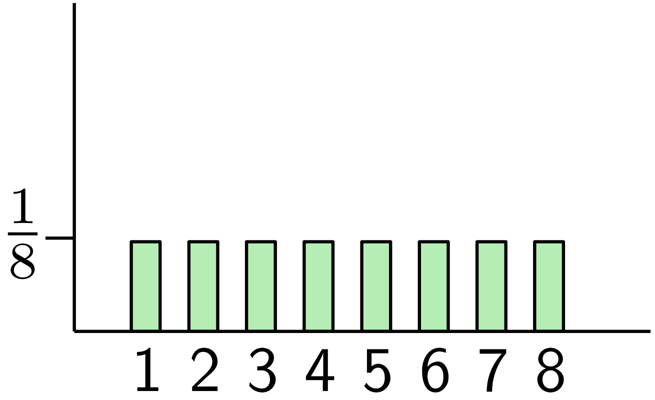
```
private s = getBufferSize();


public int compare(int i){
 if(s <= i)
  log.write("too large"); // 1 s
 else
  some computation; // 2 s
 return 0;
}
```

Attacker Belief?

😈 s?

Attacker Belief?

😈 s?

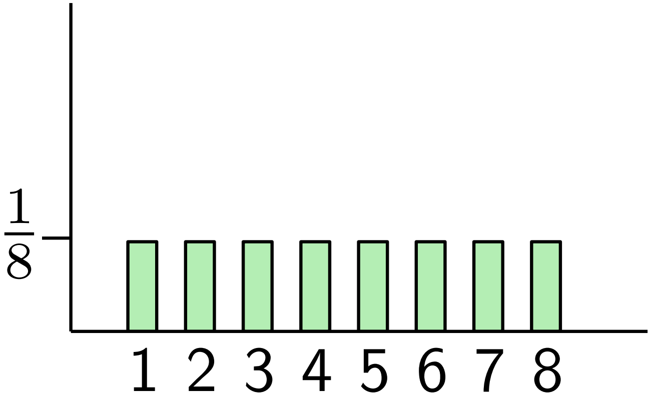


Attacker Belief?

Input Choice?

😈 s?

i^*

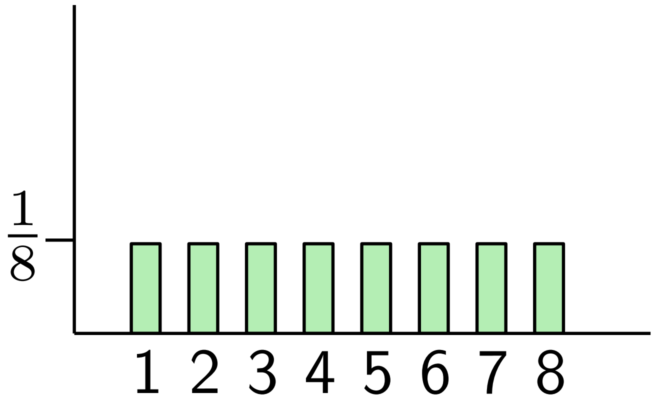


Attacker Belief?

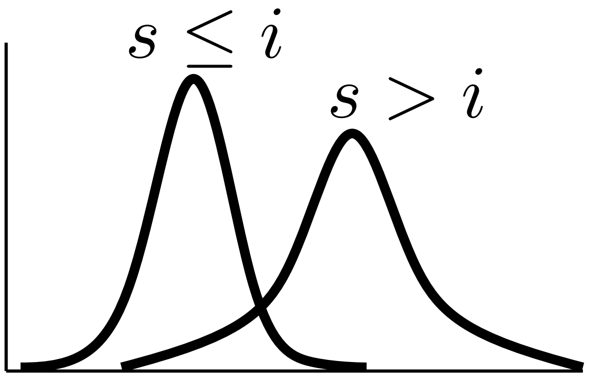
Input Choice?

Observation?

😈 s?

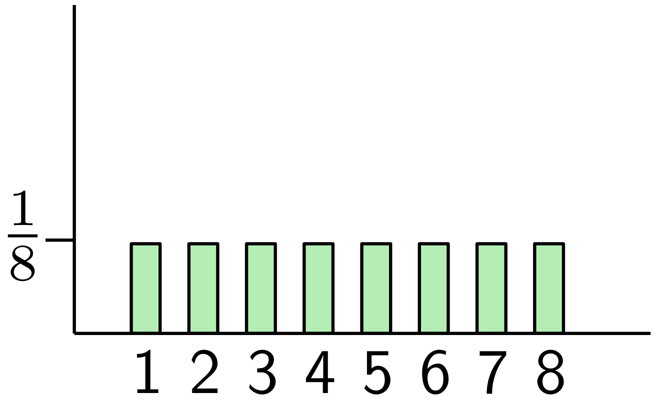


i^*



Attacker Belief?

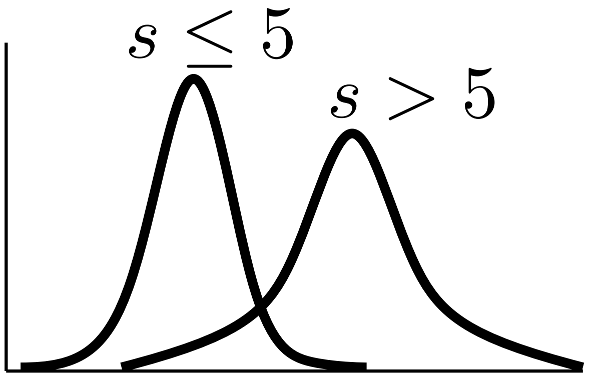
😈 s?



Input Choice?

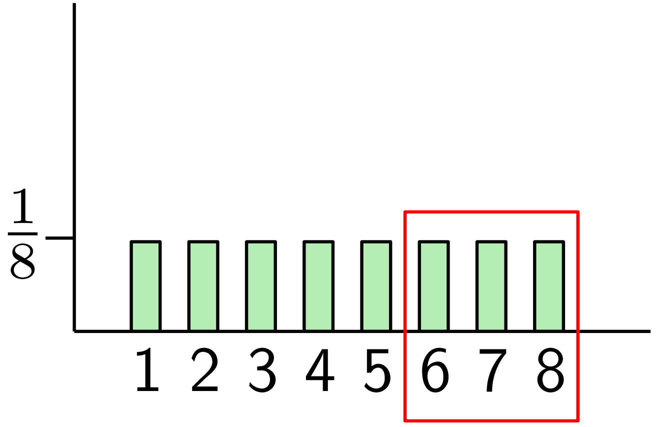
$i^* = 5$

Observation?



Attacker Belief?

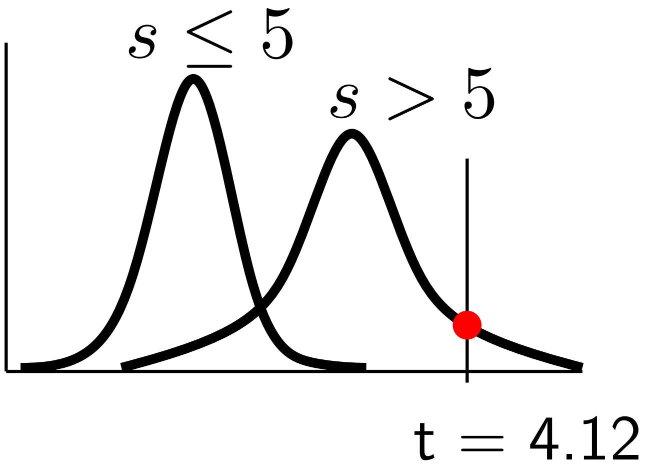
s?



Input Choice?

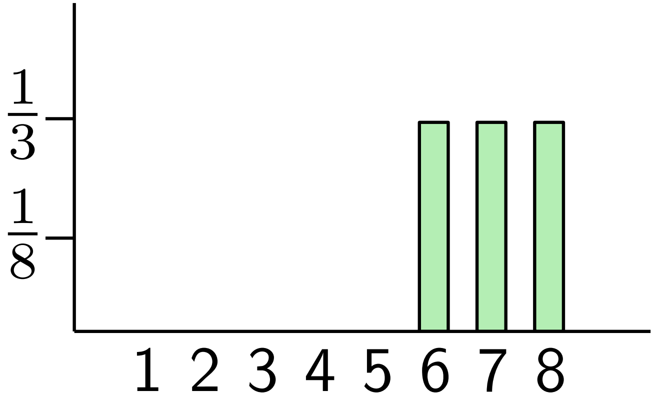
$i^* = 5$

Observation?



Attacker Belief?

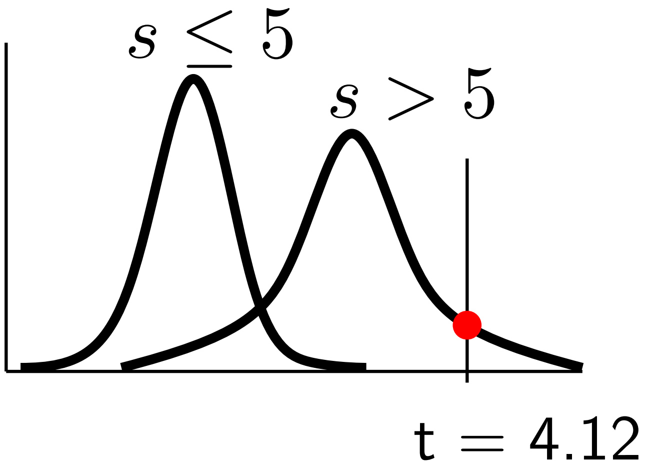
s?



Input Choice?

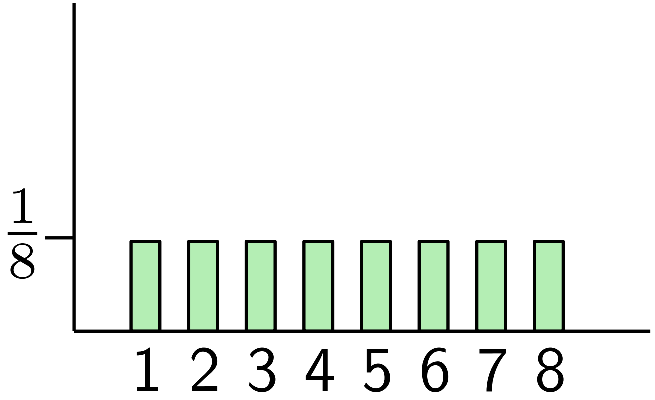
$i^* = 5$

Observation?

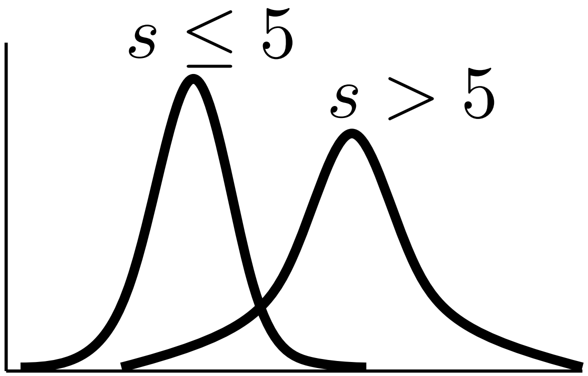


Attacker Belief? Input Choice? Observation?

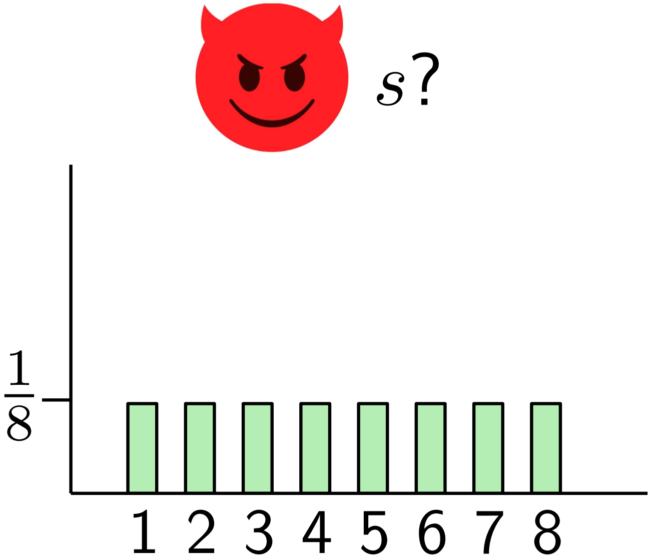
s?



$i^* = 5$



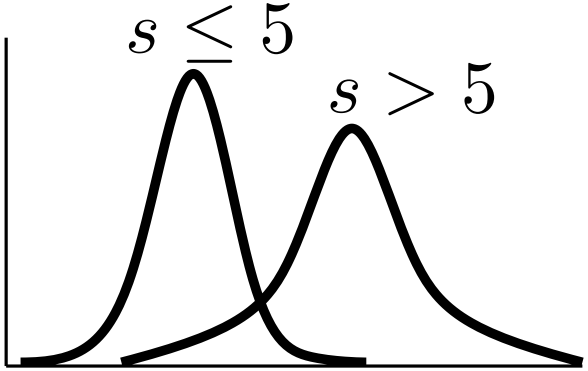
Attacker Belief?



Input Choice?

$i^* = 5$

Observation?

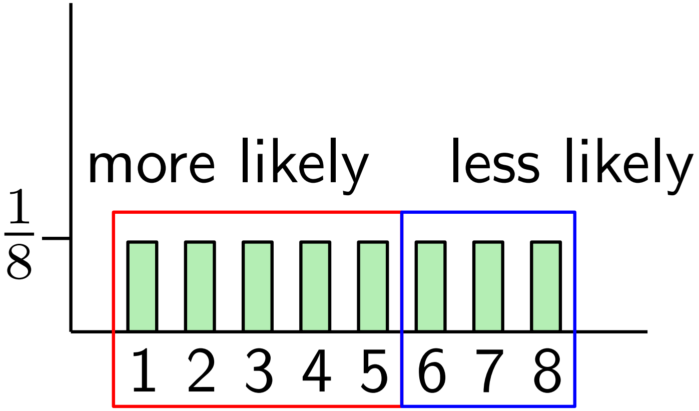


Attacker Belief?

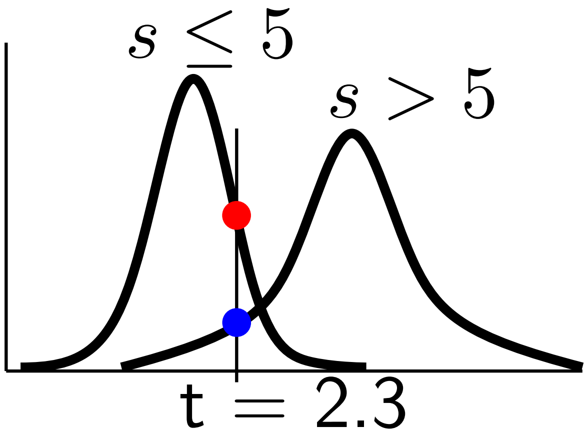
Input Choice?

Observation?

😈 s?

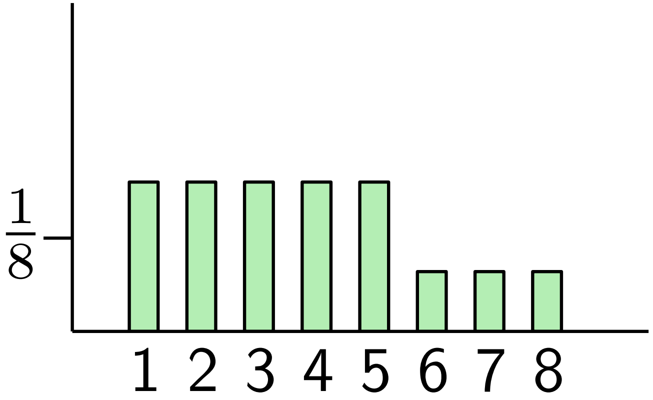


$i^* = 5$



Attacker Belief?

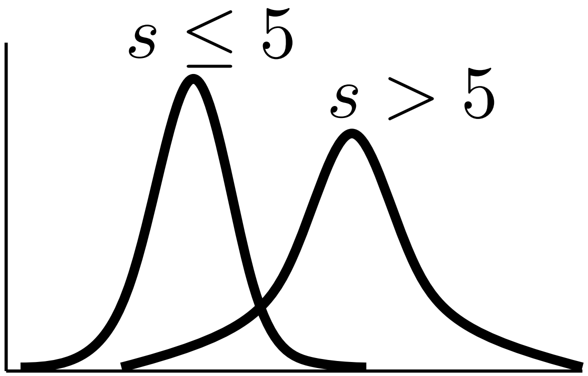
😈 s?



Input Choice?

$i^* = 5$

Observation?

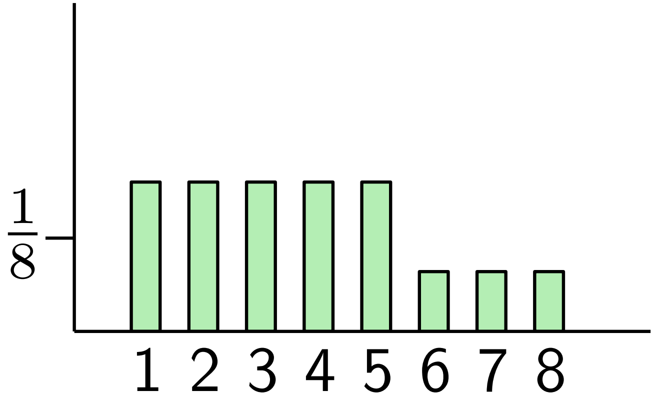


Attacker Belief?

Input Choice?

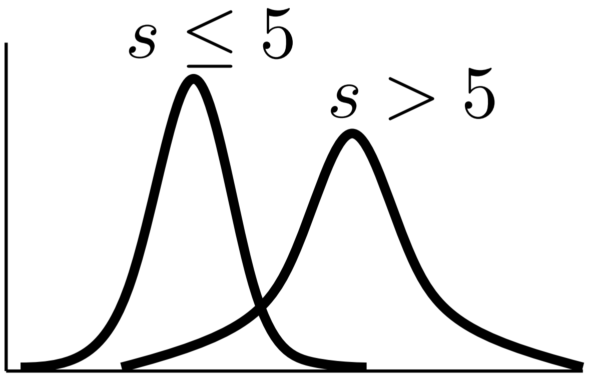
Observation?

😈 s?



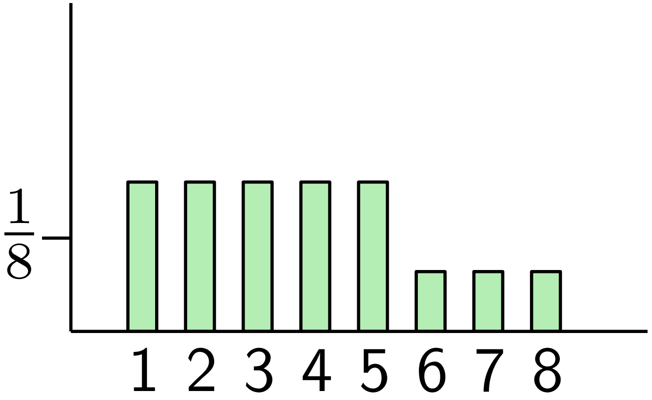
$p(s|o, i^{*}\ )$

$i^{*}\ = 5$



Attacker Belief?

😈 s?

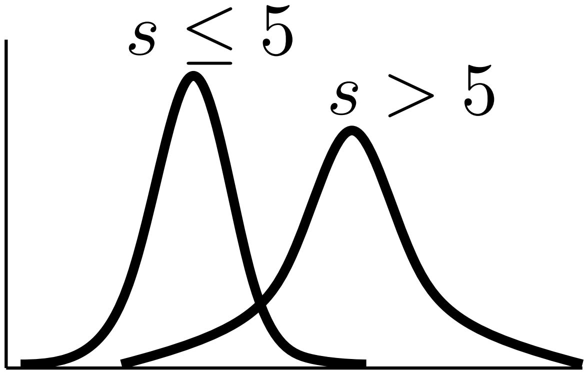


$p(s|o, i^*\ )$

Input Choice?

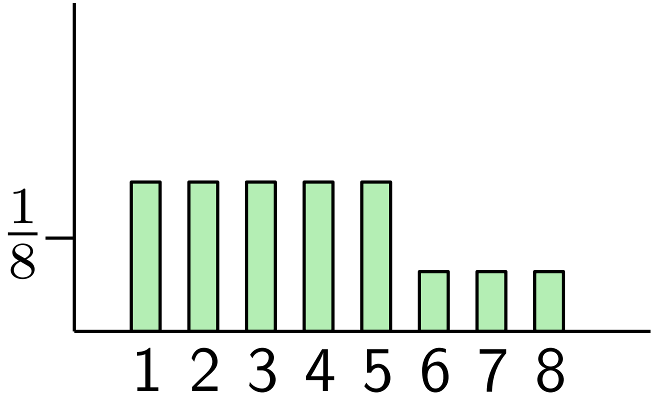
$i^* \ = 5$

Observation?



Attacker Belief?

😈 s?

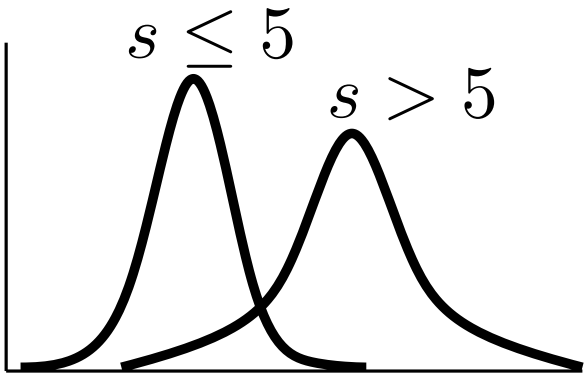


$p(s|o, i^*\ )$

Input Choice?

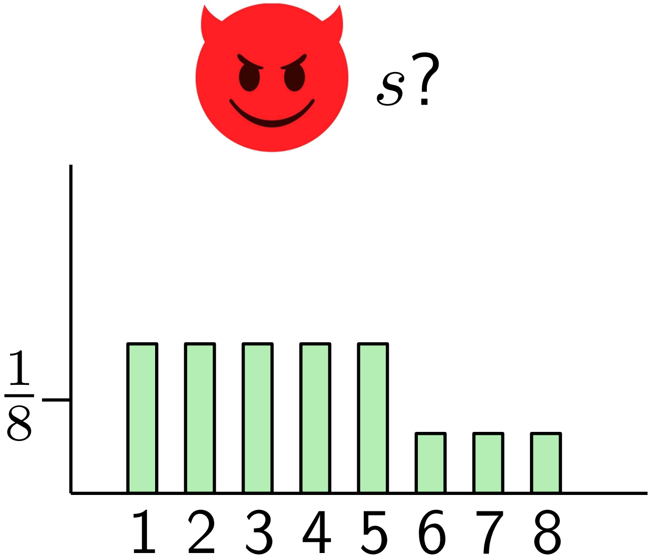
$i^*\ = 5$

Observation?

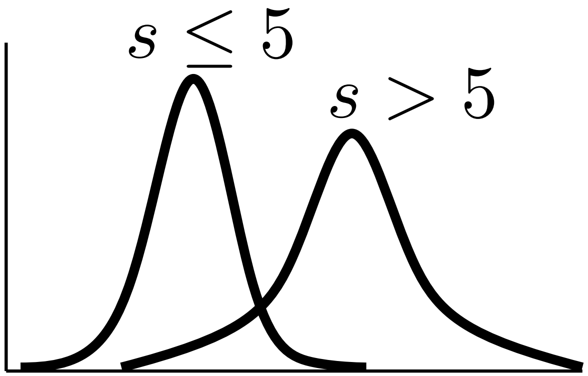


$p(o|s, i)$

Attacker Belief? Input Choice? Observation?



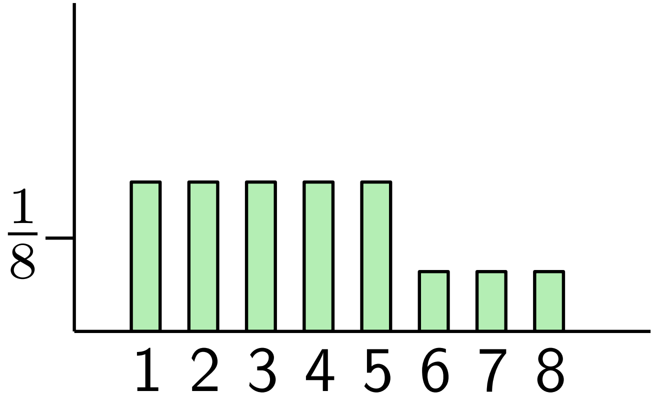
$i^* = 5$



$p(s|o, i^*) \longleftarrow p(o|s, i)$

Attacker Belief? Input Choice? Observation?

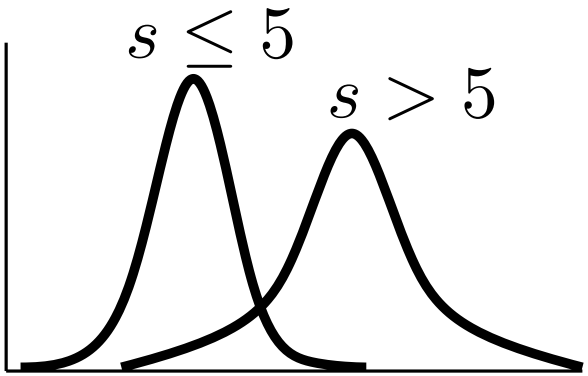
😈 s?



$p(s|o, i^*)$

$i^* = 5$

$p(o|s, i)$

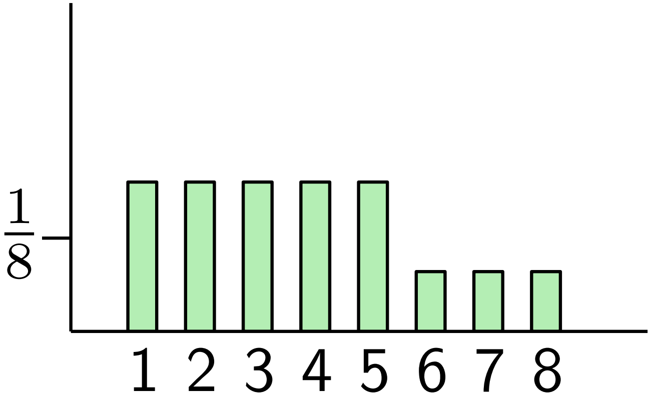


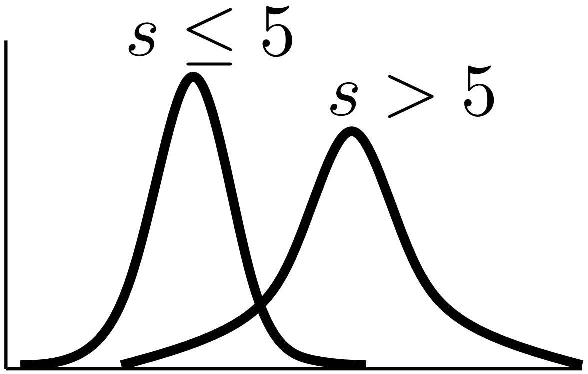
$p(o|s, i)$

Attacker Belief? Input Choice? Observation?

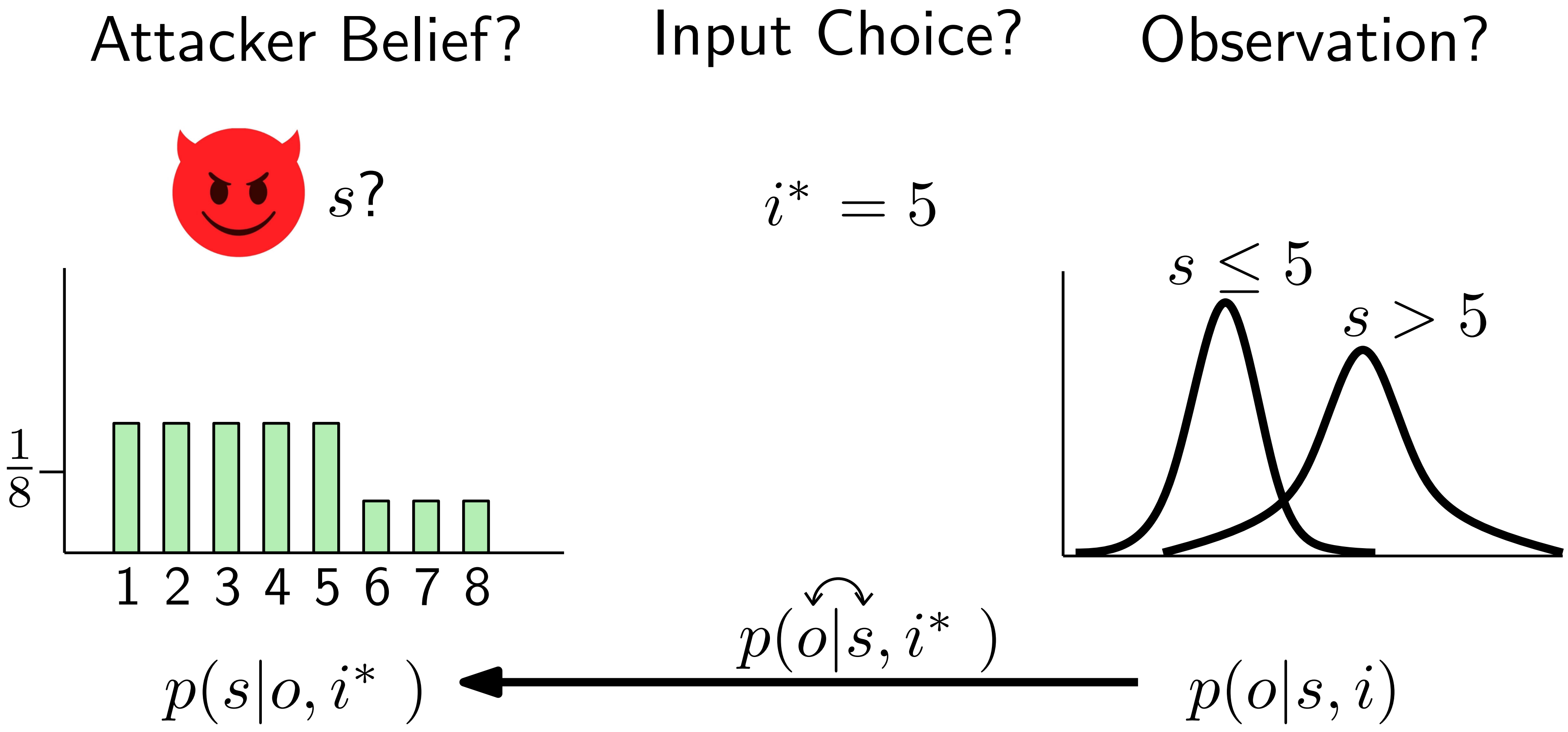
s?

$i^* = 5$





$$p(s|o, i^*) \xleftarrow{p(o|s, i^*)} p(o|s, i)$$



Attacker Belief?

Input Choice?

Observation?

s?

$i^* = 5$

$p(\overset{\curvearrowright}{o|s}, i^*)$

$p(s|o, i^*)$

$p(o|s, i)$

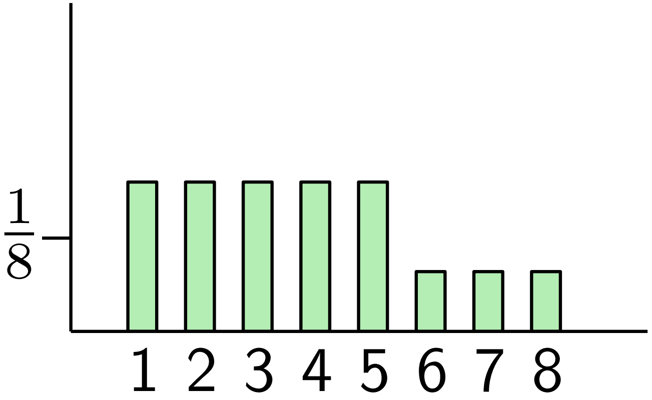
Attacker Belief?

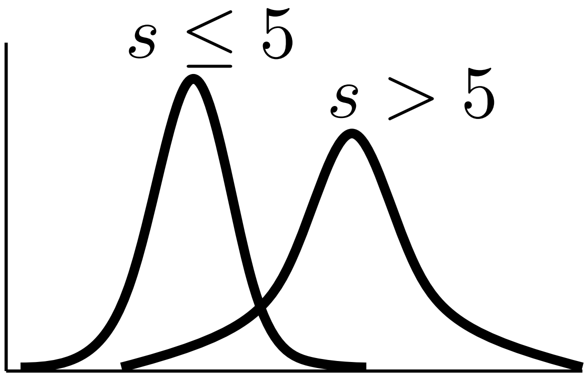
Input Choice?

Observation?

s?

$i^* = 5$



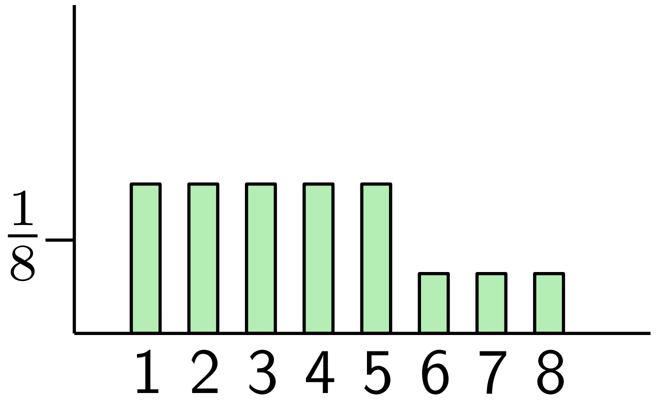


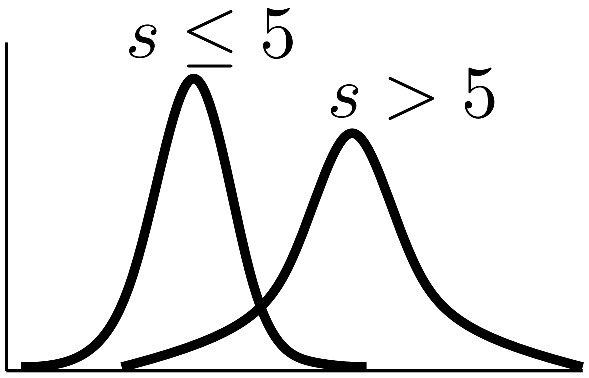
$$p(s|o, i^*) \xleftarrow{p(\overset{\curvearrowright}{s|o}, i^*)} p(o|s, i)$$

Attacker Belief? Input Choice? Observation?

s?

$i^* = 5$





$$p(s|o, i^*) \xleftarrow[\text{Bayes' Rule}]{p(\overset{\curvearrowright}{s|o}, i^*)} p(o|s, i)$$

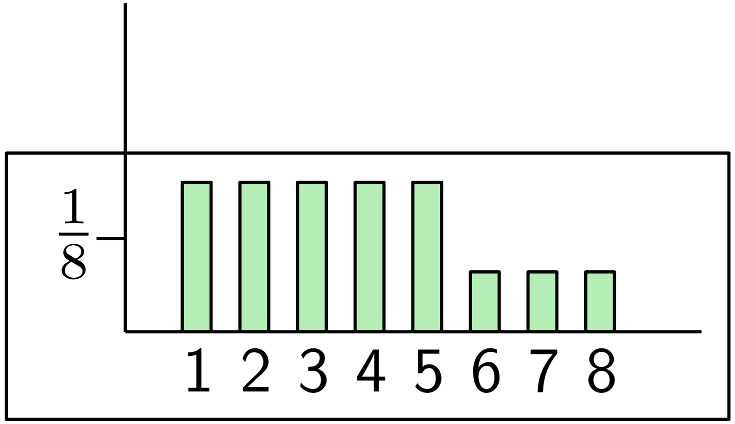
Attacker Belief?

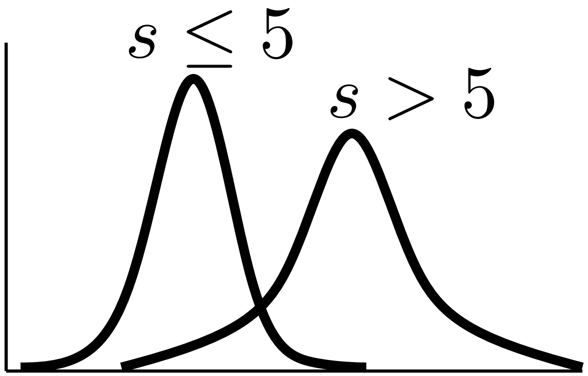
Input Choice?

Observation?

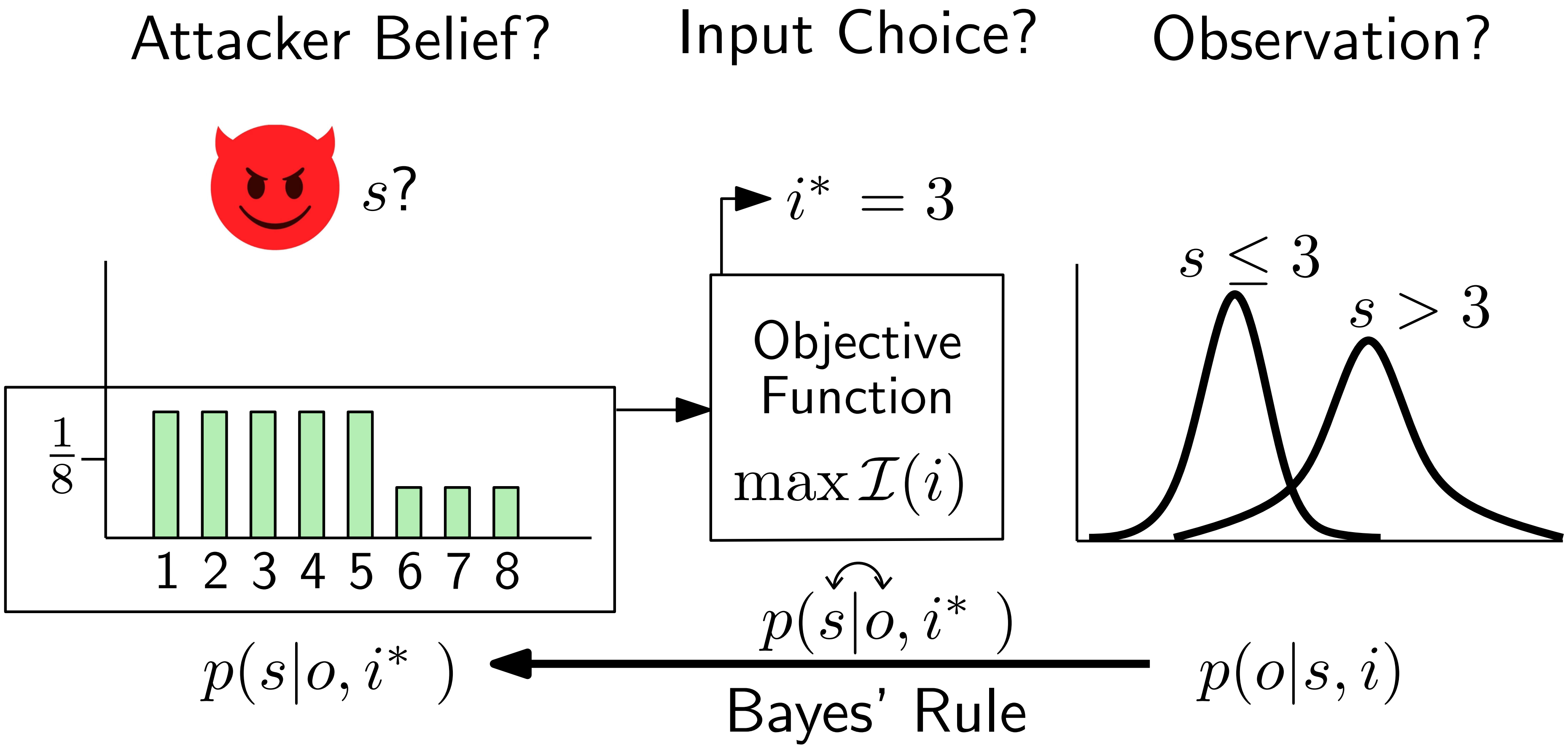
$s?$

$i^* = 5$

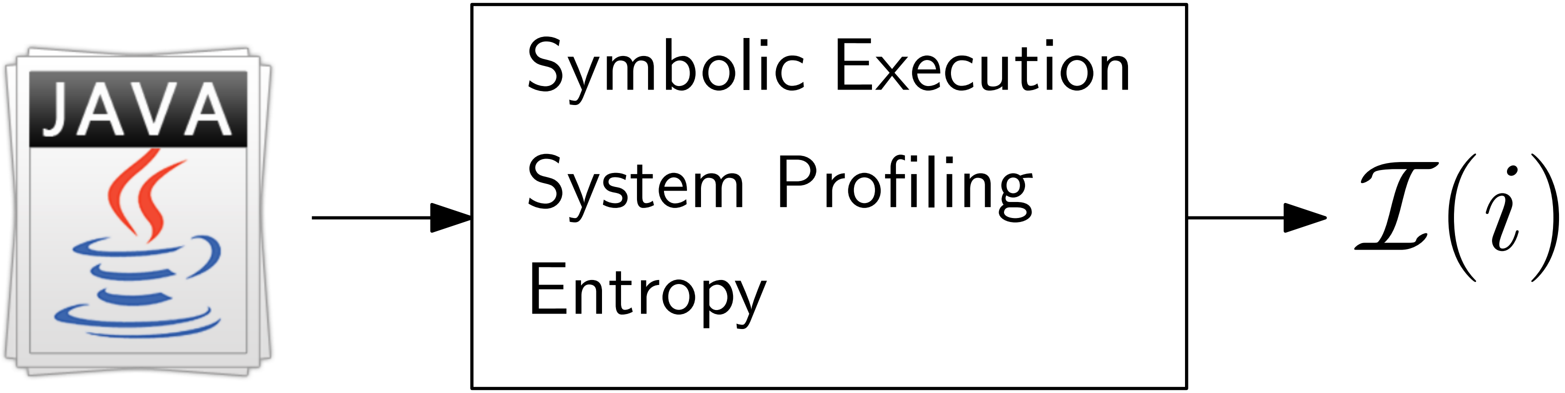




$$p(s|o, i^*) \xleftarrow[\text{Bayes' Rule}]{p(\overset{\curvearrowright}{s|o}, i^*)} p(o|s, i)$$


Attacker Belief?
Input Choice?
Observation?
s?
$i^* = 3$
$s \leq 3$
$s > 3$
$\frac{1}{8}$
1 2 3 4 5 6 7 8
Objective Function
$\max \mathcal{I}(i)$
$p(s|o, i^*)$
$p(s|o, i^*)$
$p(o|s, i)$
Bayes' Rule

Our Approach



Symbolic Execution

System Profiling

Entropy

$\mathcal{I}(i)$

$\mathcal{I}(i)$ is a symbolic expression over program inputs i that measures how much information is gained by an attacker when making input i.

Find i that maximizes $\mathcal{I}(i)$ to get the attacker's best input at every step.

1. Offline Static Analysis

1. Offline Static Analysis

2. Offline Dynamic Analysis

1. Offline Static Analysis

2. Offline Dynamic Analysis

3. Online Attack Synthesis

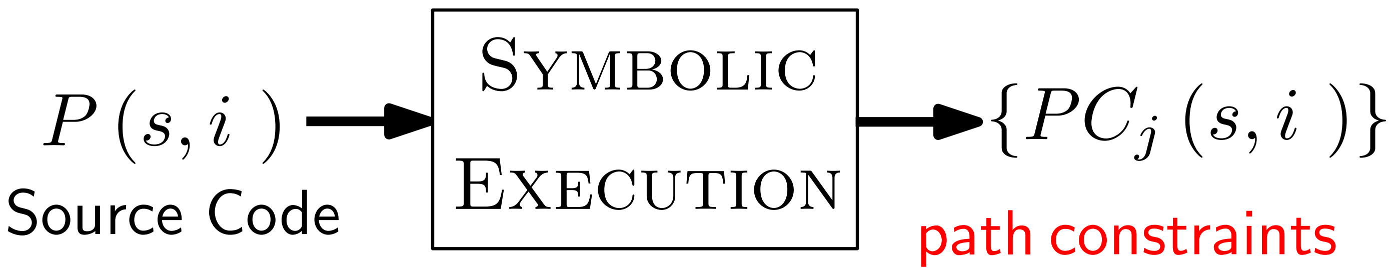
1. Offline Static Analysis

2. Offline Dynamic Analysis

3. Online Attack Synthesis

$P\,(s, i\ \ )$

Source Code



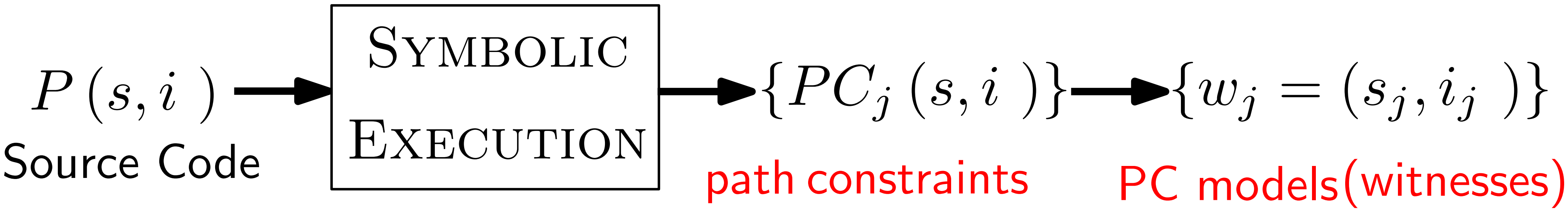
$P(s,i)$

Source Code

Symbolic Execution

$\{PC_j(s,i)\}$

path constraints



$P\,(s, i\ )$

Source Code

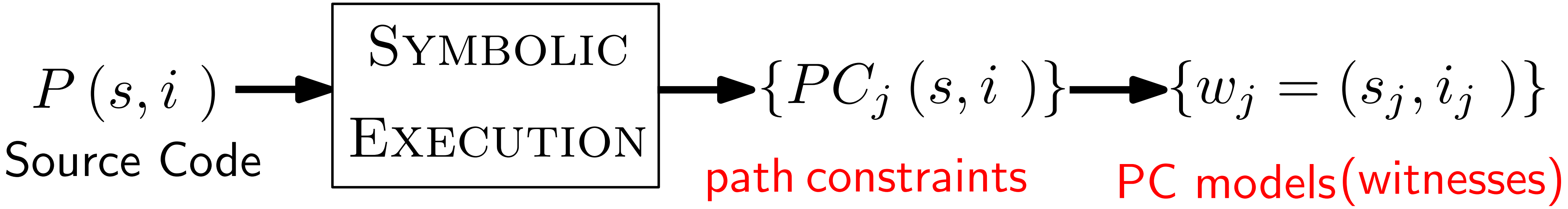
SYMBOLIC EXECUTION

$\{PC_j\,(s, i\ )\}$

path constraints

$\{w_j = (s_j, i_j\ )\}$

PC models(witnesses)



$P(s,i)$

Source Code

SYMBOLIC EXECUTION

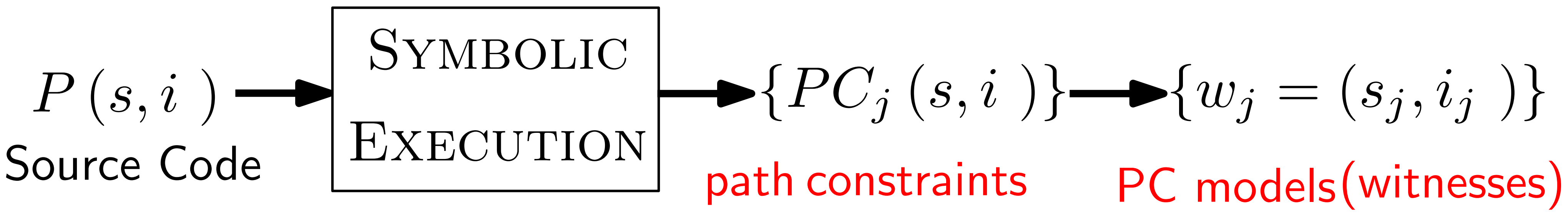
$\{PC_j(s,i)\}$

path constraints

$\{w_j=(s_j,i_j)\}$

PC models(witnesses)

Each PC characterizes an observable program behavior



$P(s, i)$

Source Code

Symbolic Execution

$\{PC_j(s, i)\}$

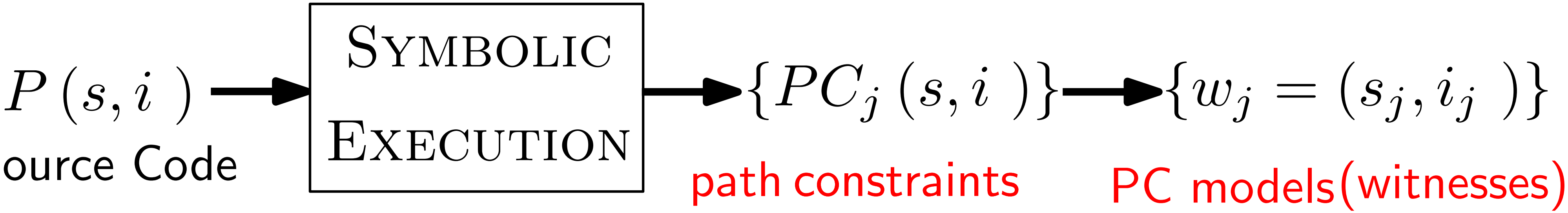
path constraints

$\{w_j = (s_j, i_j)\}$

PC models(witnesses)

Each PC characterizes an observable program behavior

$$(s, i) \models PC_j \qquad (s', i') \models PC_j$$



$P(s, i)$

Source Code

SYMBOLIC EXECUTION

$\{PC_j(s, i)\}$

path constraints

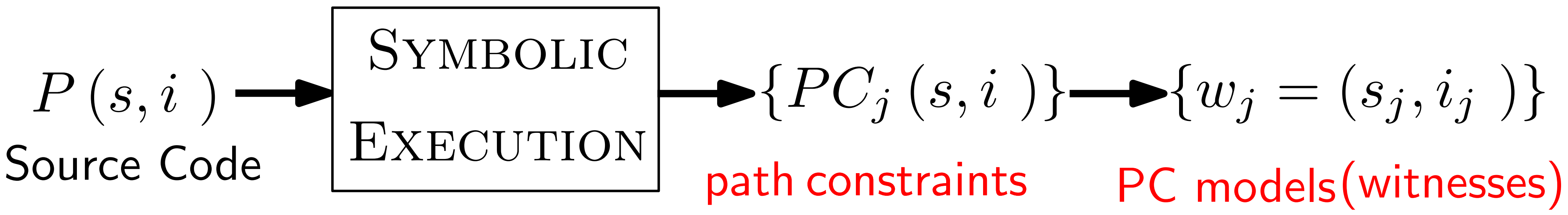
$\{w_j = (s_j, i_j)\}$

PC models(witnesses)

Each PC characterizes an observable program behavior

$(s, i) \models PC_j \qquad (s', i') \models PC_j$

$P(s, i) \qquad P(s', i')$



$P(s,i)$

Source Code

SYMBOLIC EXECUTION

$\{PC_j(s,i)\}$

path constraints

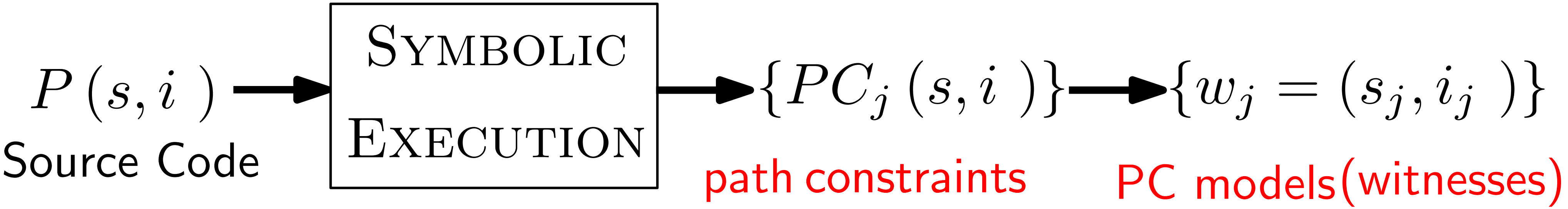
$\{w_j = (s_j, i_j)\}$

PC models(witnesses)

Each PC characterizes an observable program behavior

$(s,i) \models PC_j \qquad (s',i') \models PC_j$

$P(s,i)$? 😈 ? $P(s',i')$



$P(s, i\ )$

Source Code

SYMBOLIC EXECUTION

$\{PC_j(s, i\ )\}$

path constraints

$\{w_j = (s_j, i_j\ )\}$

PC models(witnesses)

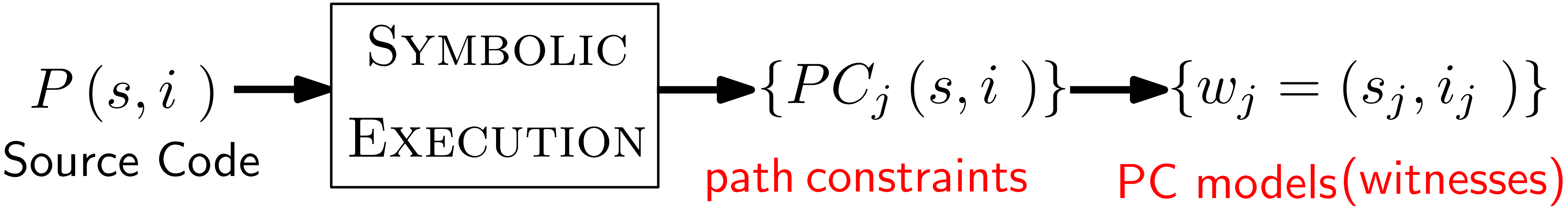
Each PC characterizes an observable program behavior

$(s, i\ ) \models PC_j \qquad\qquad (s', i'\ ) \models PC_j$

$P(s, i\ )$? 😈 ? $P(s', i'\ )$

$PC_j(s, i\ )$ characterizes indistinguishable behaviors

$P(s, i\ )$ is a representative of all behaviors in that class



$P(s, i)$

Source Code

Symbolic Execution

$\{PC_j(s, i)\}$

path constraints

$\{w_j = (s_j, i_j)\}$

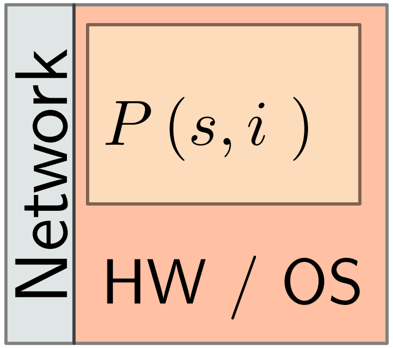
PC models(witnesses)

1. Offline Static Analysis
2. Offline Dynamic Analysis
3. Online Attack Synthesis

1. Offline Static Analysis
2. Offline Dynamic Analysis
3. Online Attack Synthesis

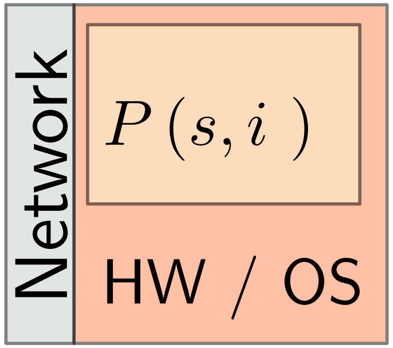
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

Characterize effect of noise on each class of program behaviors using the witness for that behavior.

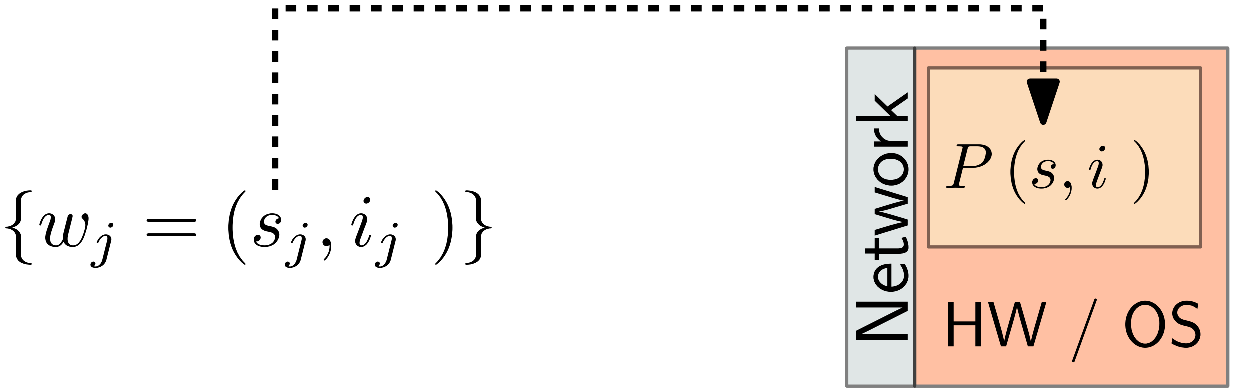


Characterize effect of noise on each class of program behaviors using the witness for that behavior.

$$\{w_j = (s_j, i_j\ )\}$$



Characterize effect of noise on each class of program behaviors using the witness for that behavior.



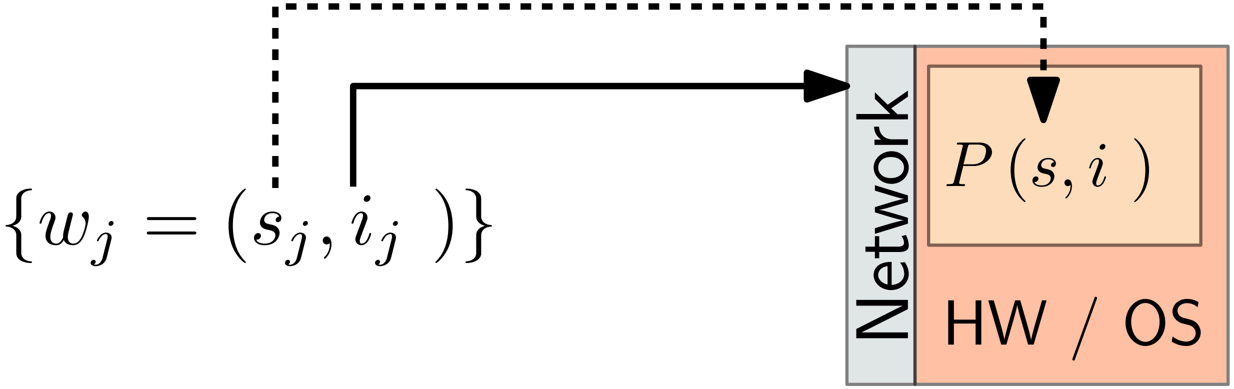
$\{w_j = (s_j, i_j\ )\}$

$P\,(s, i\ )$

Network

HW / OS

Characterize effect of noise on each class of program behaviors using the witness for that behavior.



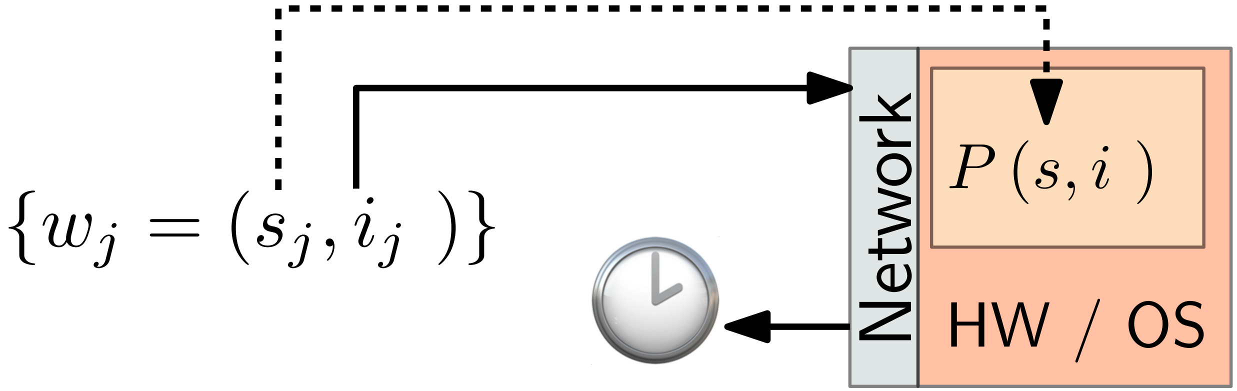
$\{w_j = (s_j, i_j\ )\}$

Network

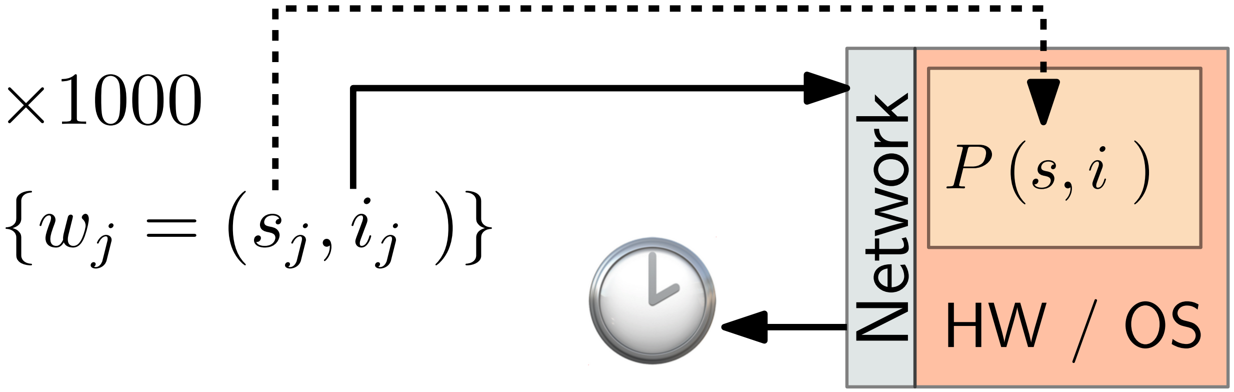
$P\,(s, i\ )$

HW / OS

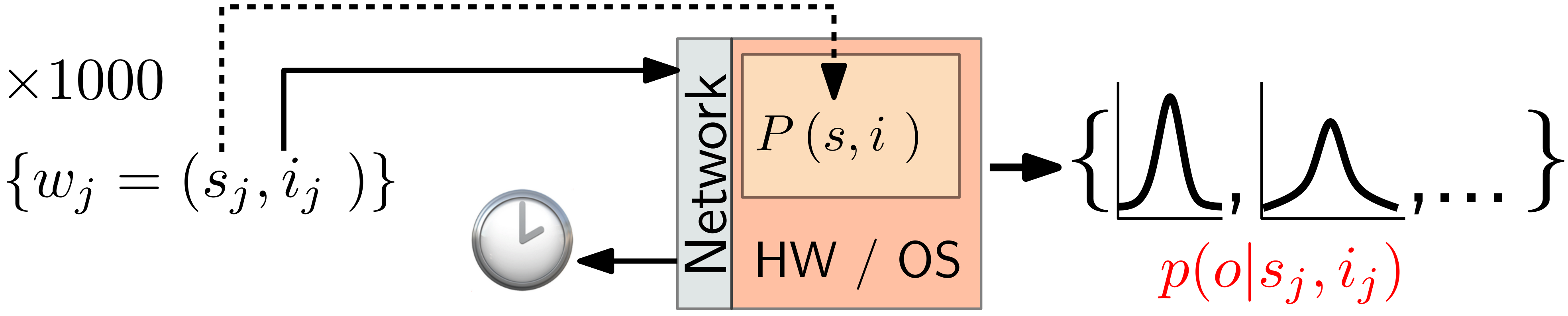
Characterize effect of noise on each class of program behaviors using the witness for that behavior.



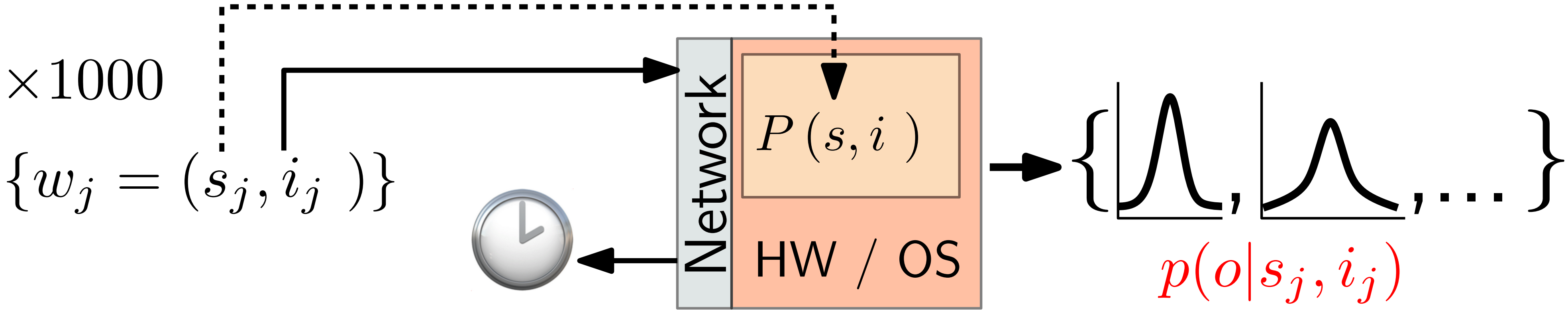
Characterize effect of noise on each class of program behaviors using the witness for that behavior.



Characterize effect of noise on each class of program behaviors using the witness for that behavior.



Characterize effect of noise on each class of program behaviors using the witness for that behavior.



$\times 1000$

$\{w_j = (s_j, i_j)\}$

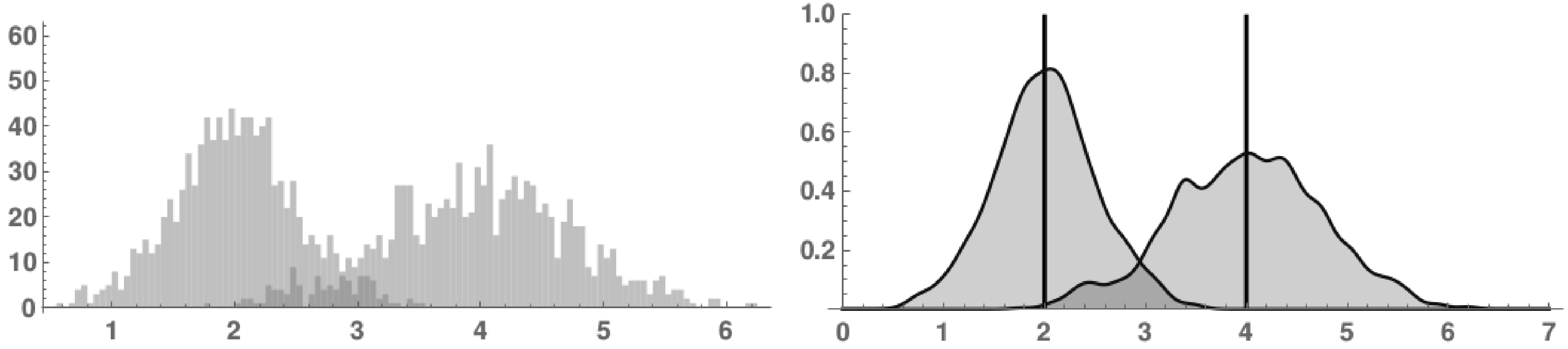
Network

$P(s, i)$

HW / OS

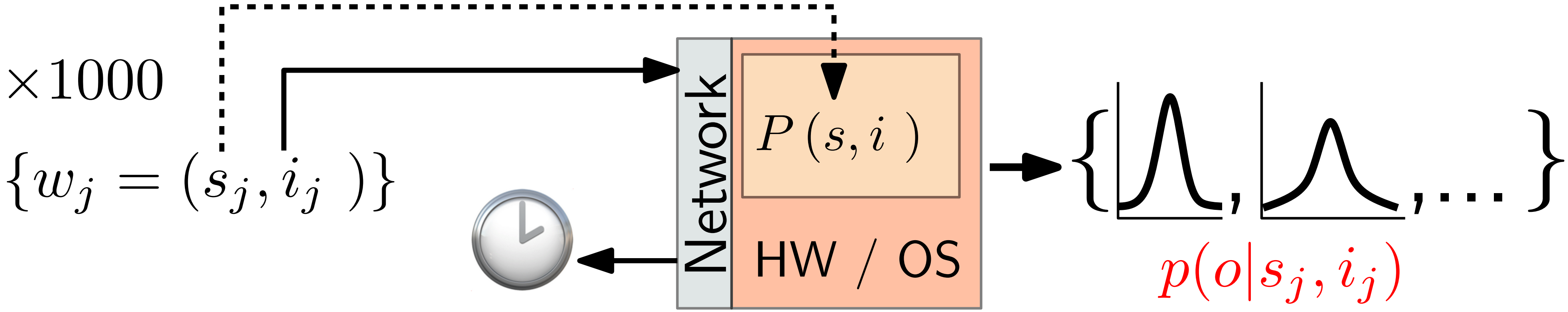
$\{ \ldots \}$

$p(o|s_j, i_j)$



Smooth Kernel Density Estimation

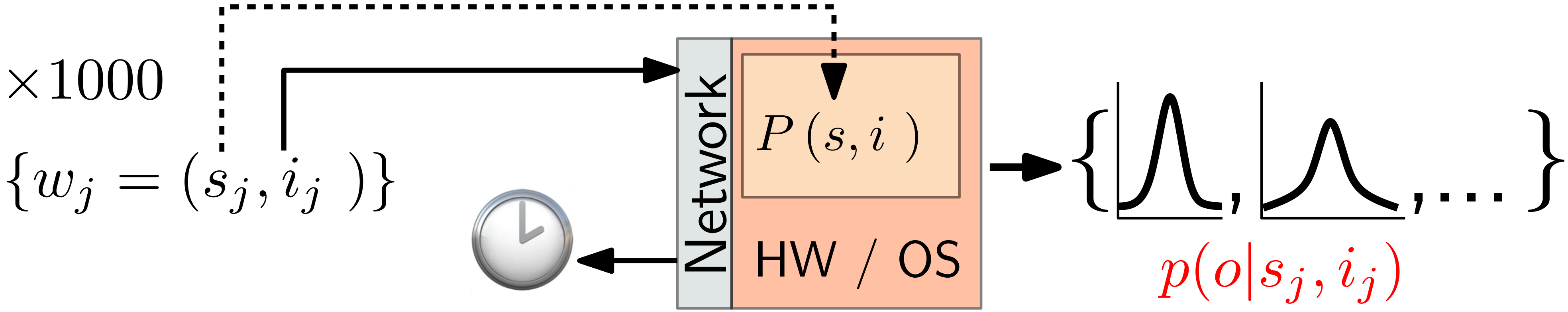
Characterize effect of noise on each class of program behaviors using the witness for that behavior.



$$p(o|PC) = \frac{1}{n\delta} \sum_{k=1}^{N} K\left(\frac{o - o_k}{\delta}\right)$$

Smooth Kernel Density Estimation

Characterize effect of noise on each class of program behaviors using the witness for that behavior.



$\times 1000$

$\{w_j = (s_j, i_j)\}$

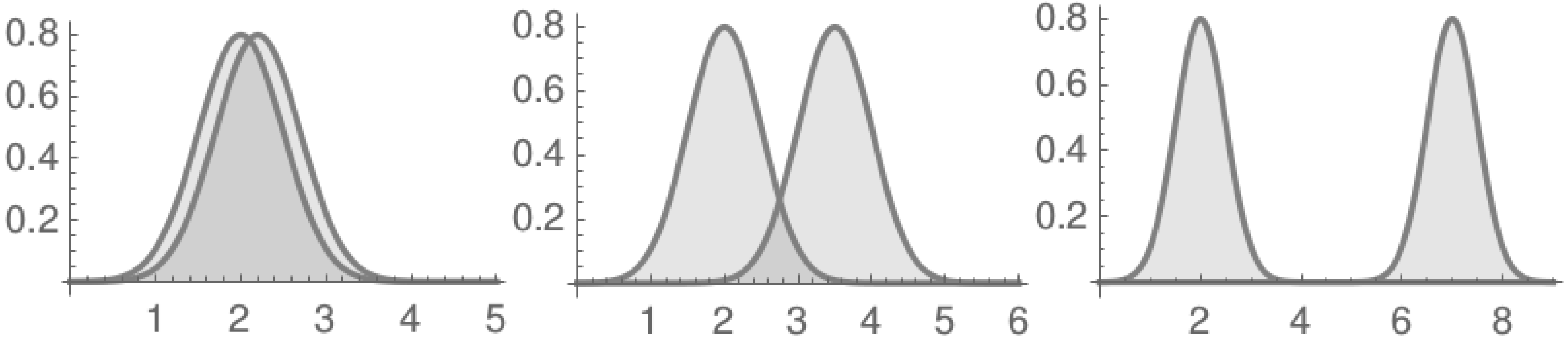
Network

$P(s, i)$

HW / OS

$\{$ ⋀ , ⋀ , … $\}$

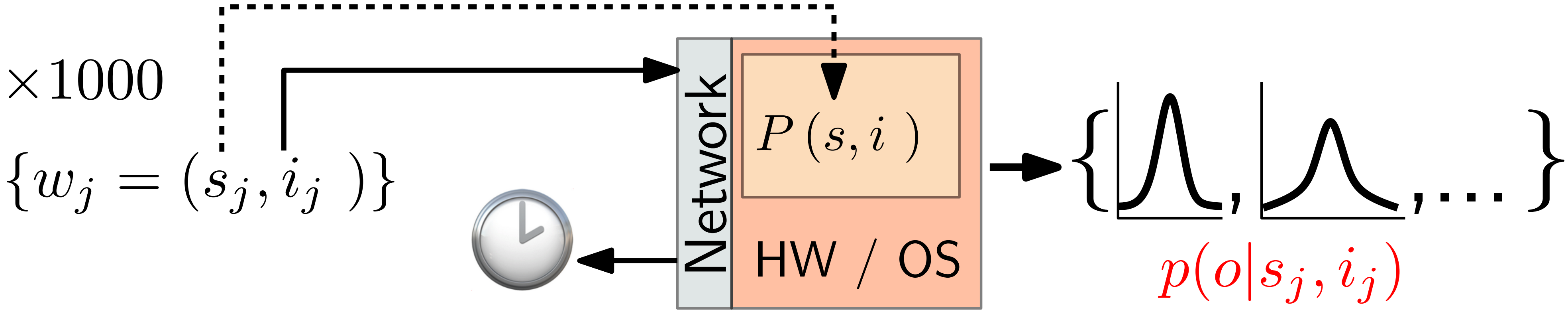
$p(o|s_j, i_j)$



(a) $d_H = 0.068$ (b) $d_H = 0.491$ (c) $d_H = 0.978$

Merging via Hellinger Distance

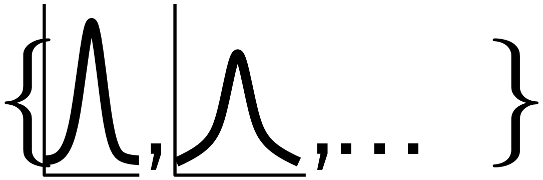
Characterize effect of noise on each class of program behaviors using the witness for that behavior.



$$d_H(p,q) = \sqrt{\frac{1}{2}\int_{-\infty}^{\infty}\left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 \, dx}$$

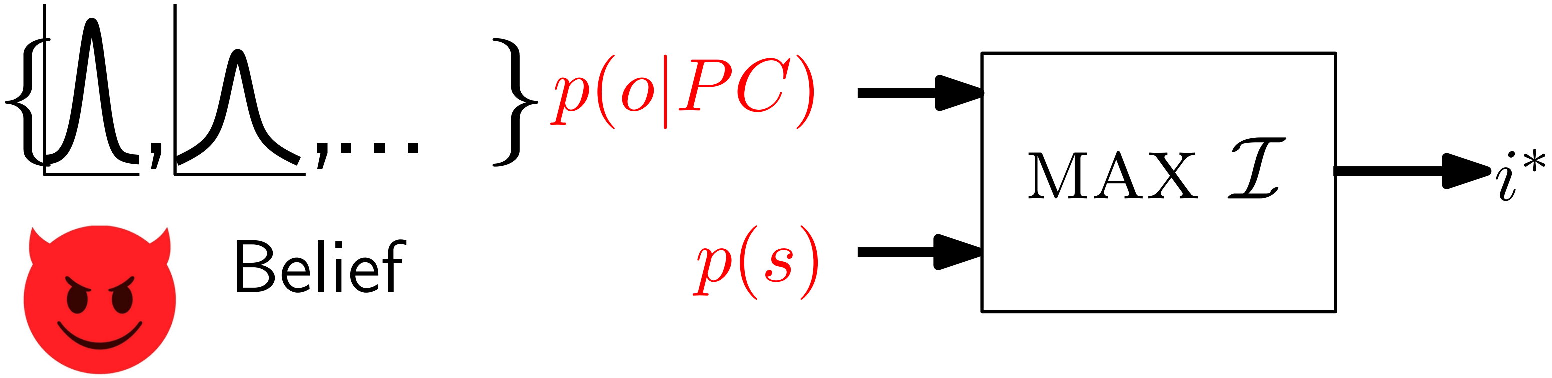
Merging via Hellinger Distance

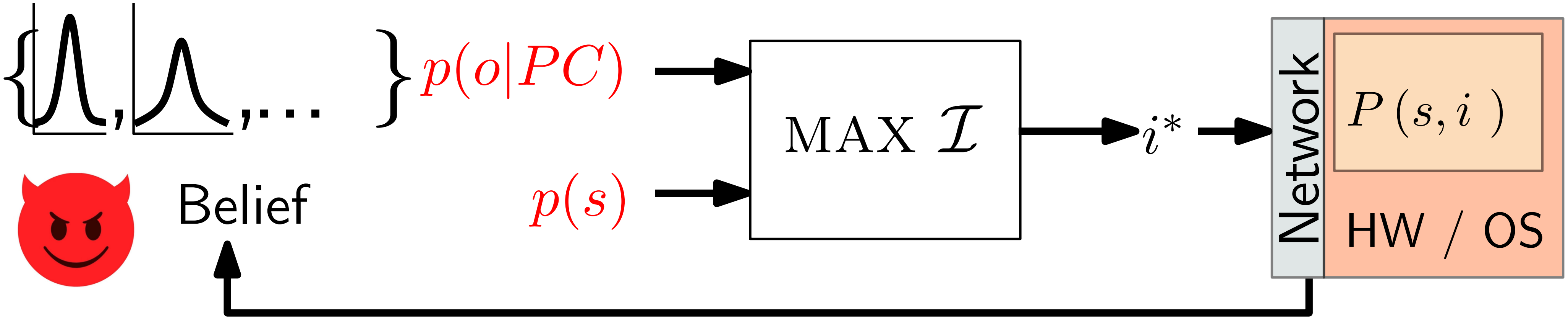
1. Offline Static Analysis
2. Offline Dynamic Analysis
3. Online Attack Synthesis

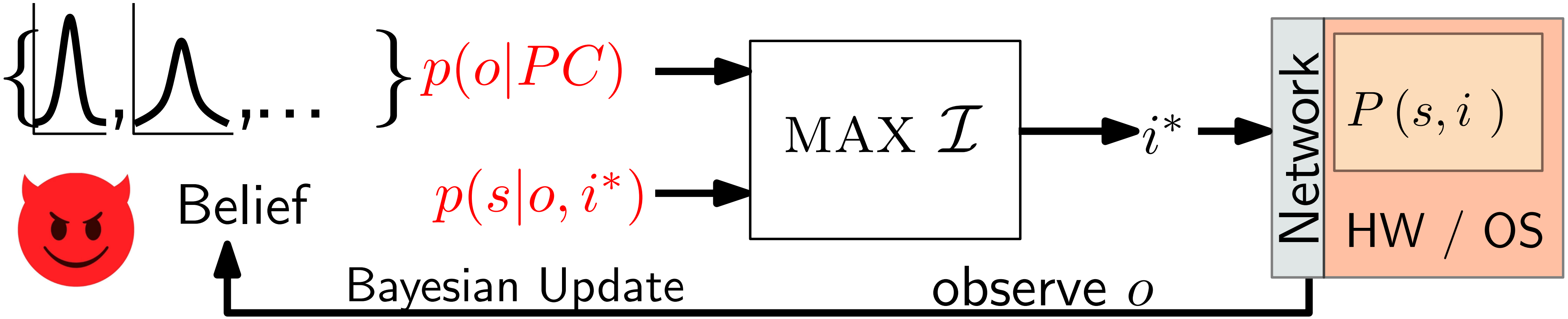


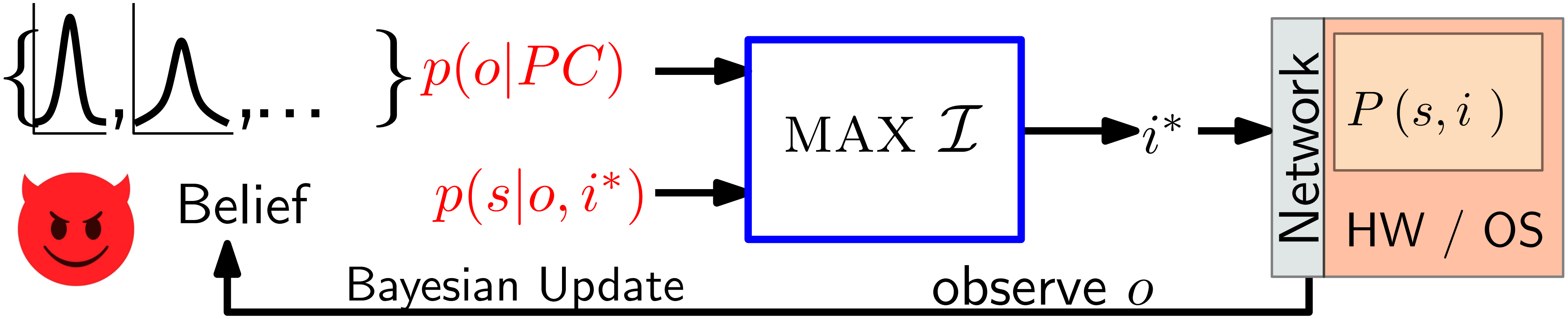
$p(o|PC)$

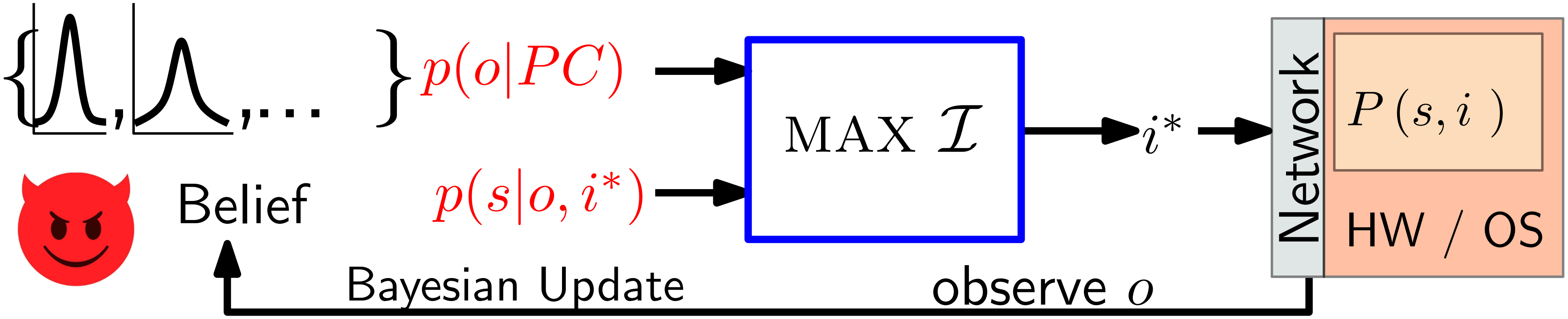
😈 Belief $p(s)$

$\{\ ,\ ,\ldots\ \}$ $p(o|PC)$
Belief
$p(s)$
MAX $\mathcal{I}$
i^*


$\{$, , … $\}$ $p(o|PC)$
Belief
$p(s)$
MAX $\mathcal{I}$
i^*
Network
$P(s,i)$
HW / OS


$\{\ ,\ ,\dots\ \}$ $p(o|PC)$
Belief
$p(s|o,i^*)$
MAX $\mathcal{I}$
i^*
Network
$P\,(s,i\ )$
HW / OS
Bayesian Update
observe o

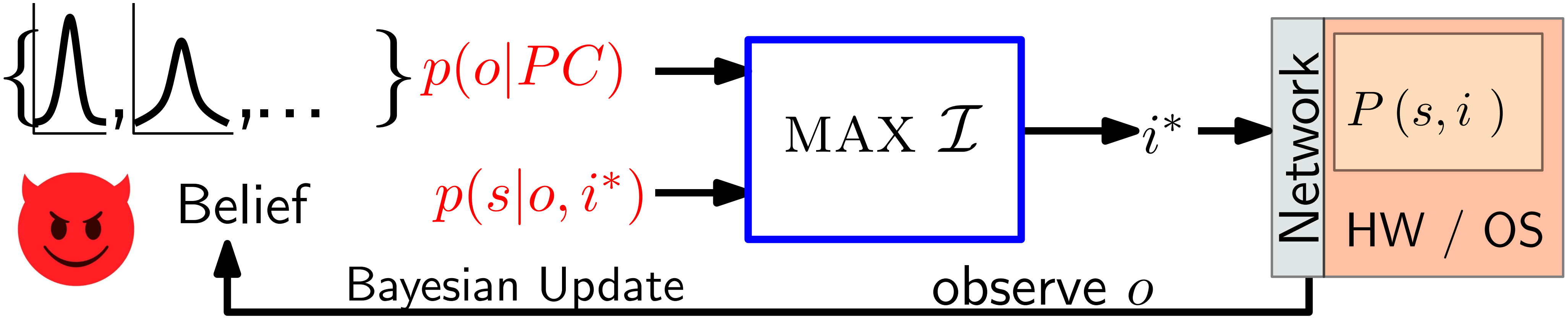

$p(o|PC)$
$p(s|o, i^*)$
Belief
MAX $\mathcal{I}$
i^*
Network
$P(s, i)$
HW / OS
Bayesian Update
observe o



$$\mathcal{I}(s;PC_j|i) = -\sum_{j=1}^{n} \underline{p(PC_j|i)} \log_2 \underline{p(PC_j|i)}$$

Expected info gain
given attacker input

Path constraint probabilities

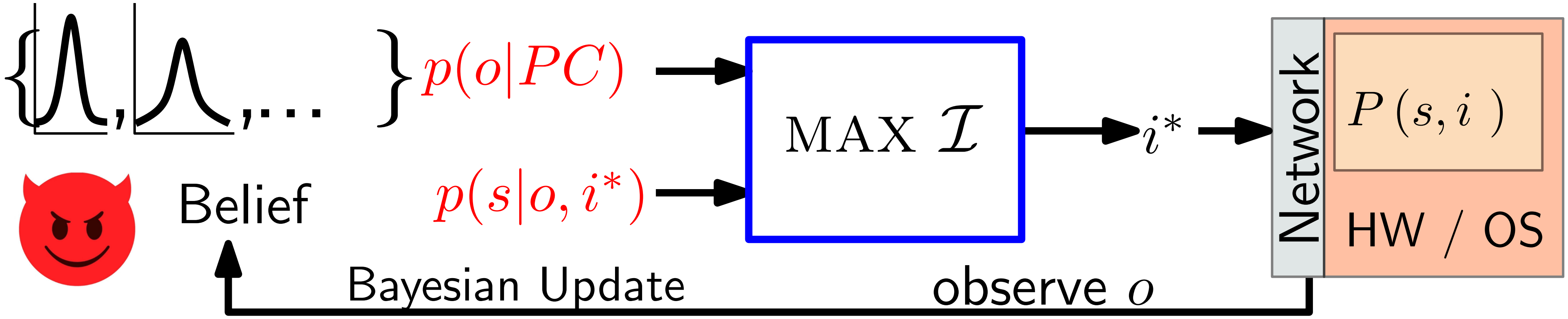


$$\mathcal{I}(s;PC_j|i) = -\sum_{j=1}^{n} \underline{p(PC_j|i)} \log_2 \underline{p(PC_j|i)}$$

Expected info gain
given attacker input

Path constraint probabilities

$$\#(PC_j|i) = \sum_{s\in S} \begin{cases} 1 & \text{if } (s,i) \models PC_j \\ 0 & \text{otherwise} \end{cases}$$



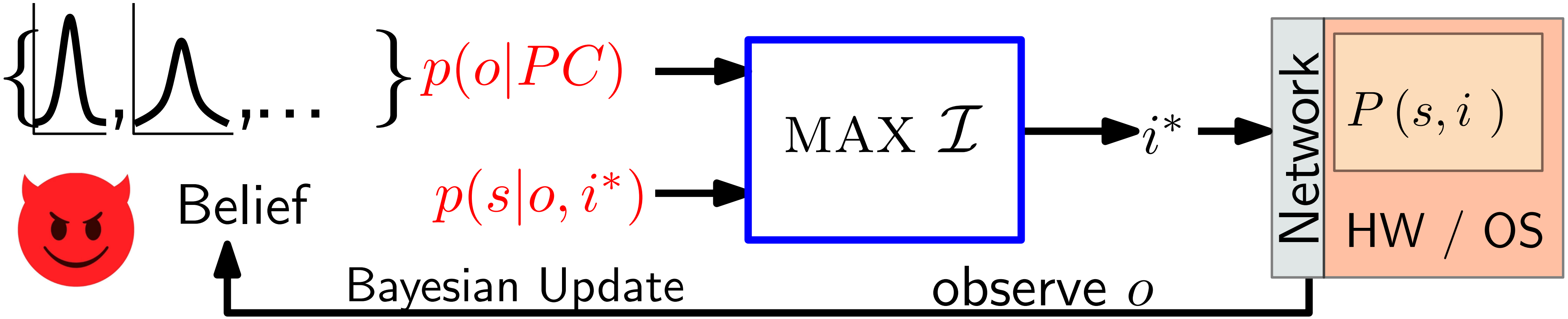
$$\mathcal{I}(s; PC_j|i) = -\sum_{j=1}^{n} \underline{p(PC_j|i)} \log_2 \underline{p(PC_j|i)}$$

Expected info gain
given attacker input

Path constraint probabilities

$$\#(PC_j|i) = \sum_{s\in S} \begin{cases} 1 & \text{if } (s, i) \models PC_j \\ 0 & \text{otherwise} \end{cases}$$

Model Counting



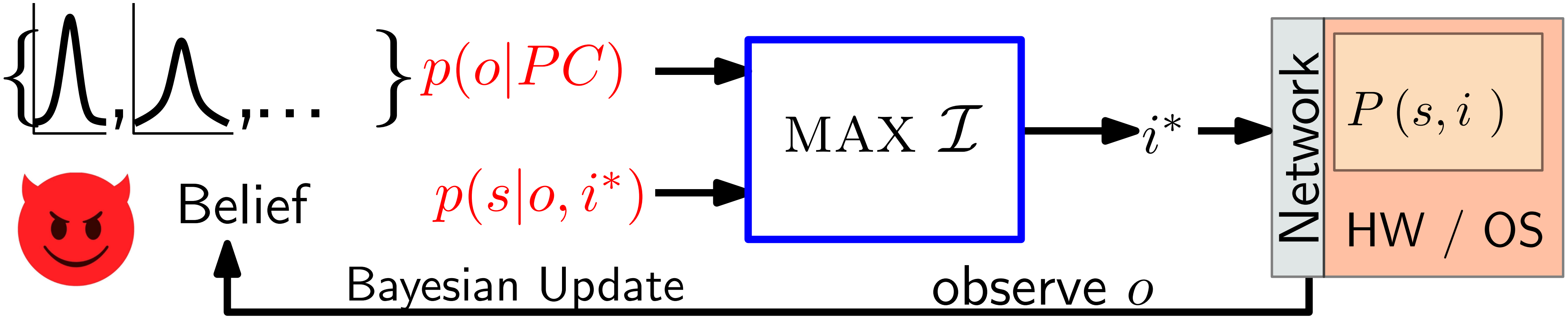
$$\mathcal{I}(s;PC_j|i) = -\sum_{j=1}^{n} \underline{p(PC_j|i)} \log_2 \underline{p(PC_j|i)}$$

Expected info gain given attacker input

Path constraint probabilities

$$p(PC_j|i) = \sum_{s\in S} p(s) \times \begin{cases} 1 & \text{if } (s,i) \models PC_j \\ 0 & \text{otherwise} \end{cases}$$

Weighted Model Counting



$$\mathcal{I}(s; PC_j|i) = -\sum_{j=1}^{n} \underline{p(PC_j|i)} \log_2 \underline{p(PC_j|i)}$$

Expected info gain given attacker input

Path constraint probabilities

$$p(PC_j|i) = \sum_{s\in S} p(s) \times \begin{cases} 1 & \text{if } (s,i) \models PC_j \\ 0 & \text{otherwise} \end{cases}$$

Weighted Model Counting

Barvinok

1. Offline Static Analysis

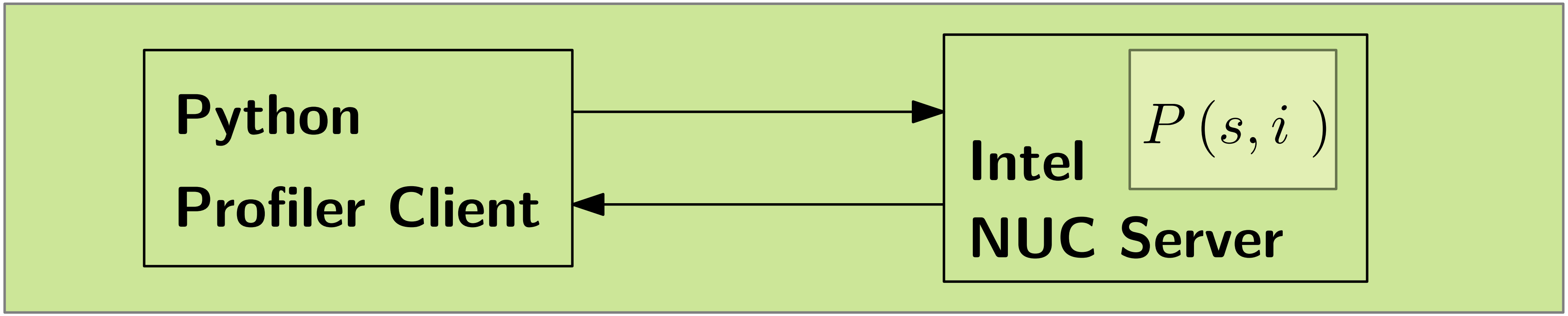
2. Offline Dynamic Analysis

3. Online Attack Synthesis

Implementation

NASA Symbolic PathFinder (SPF)

Z3 Constraint Solver



Python Profiler Client

Intel NUC Server

$P(s, i)$

Barvinok

Mathematica

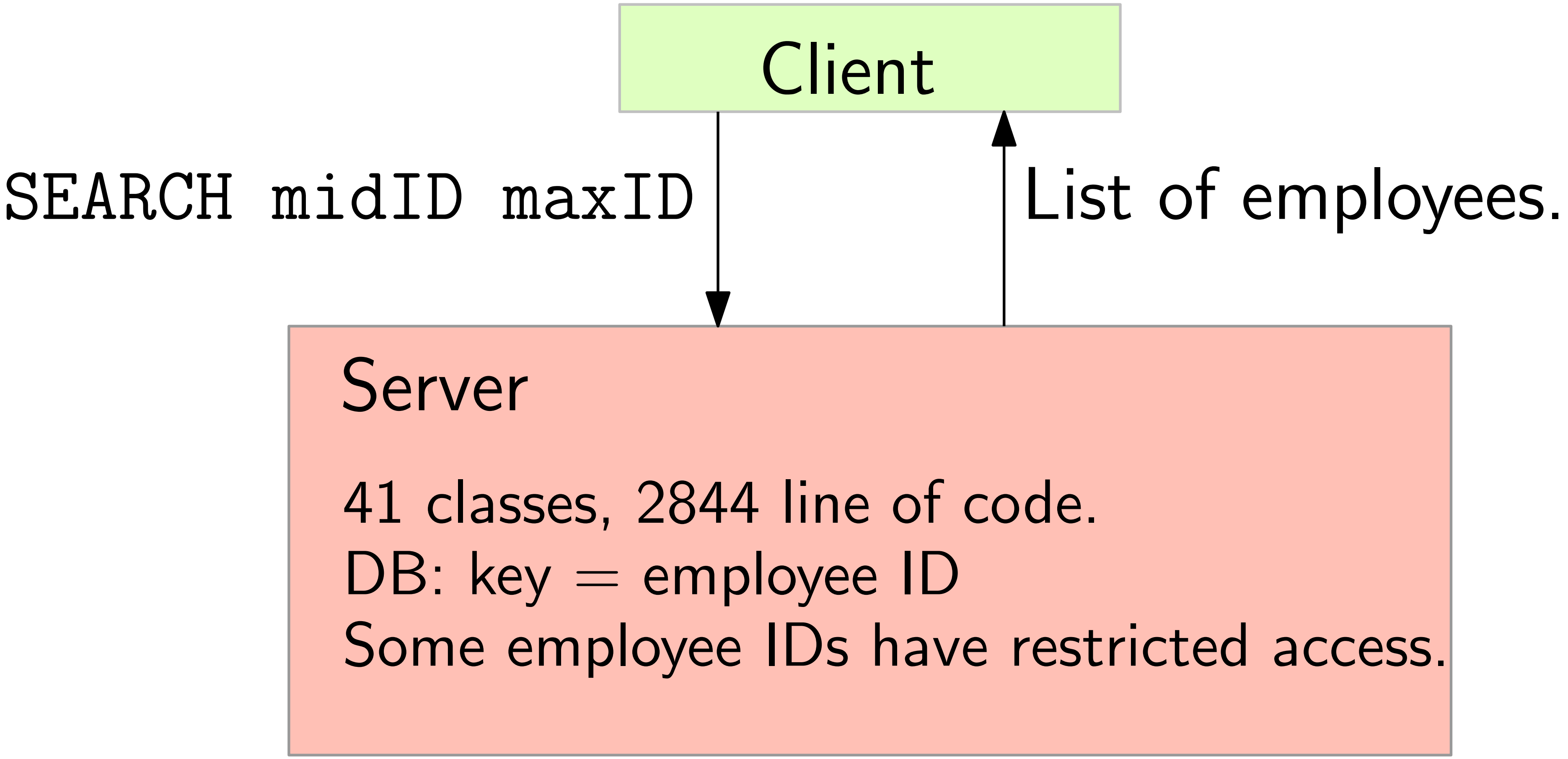
Weighted Symbolic Model Counting

Symbolic Entropy Computation

Numeric Maximization

Case Study: LawDB

From Defense Advanced Research Projects Agency (DARPA)
Space-Time Analysis for Cybersecurity (STAC) Project



Client

`SEARCH midID maxID`

List of employees.

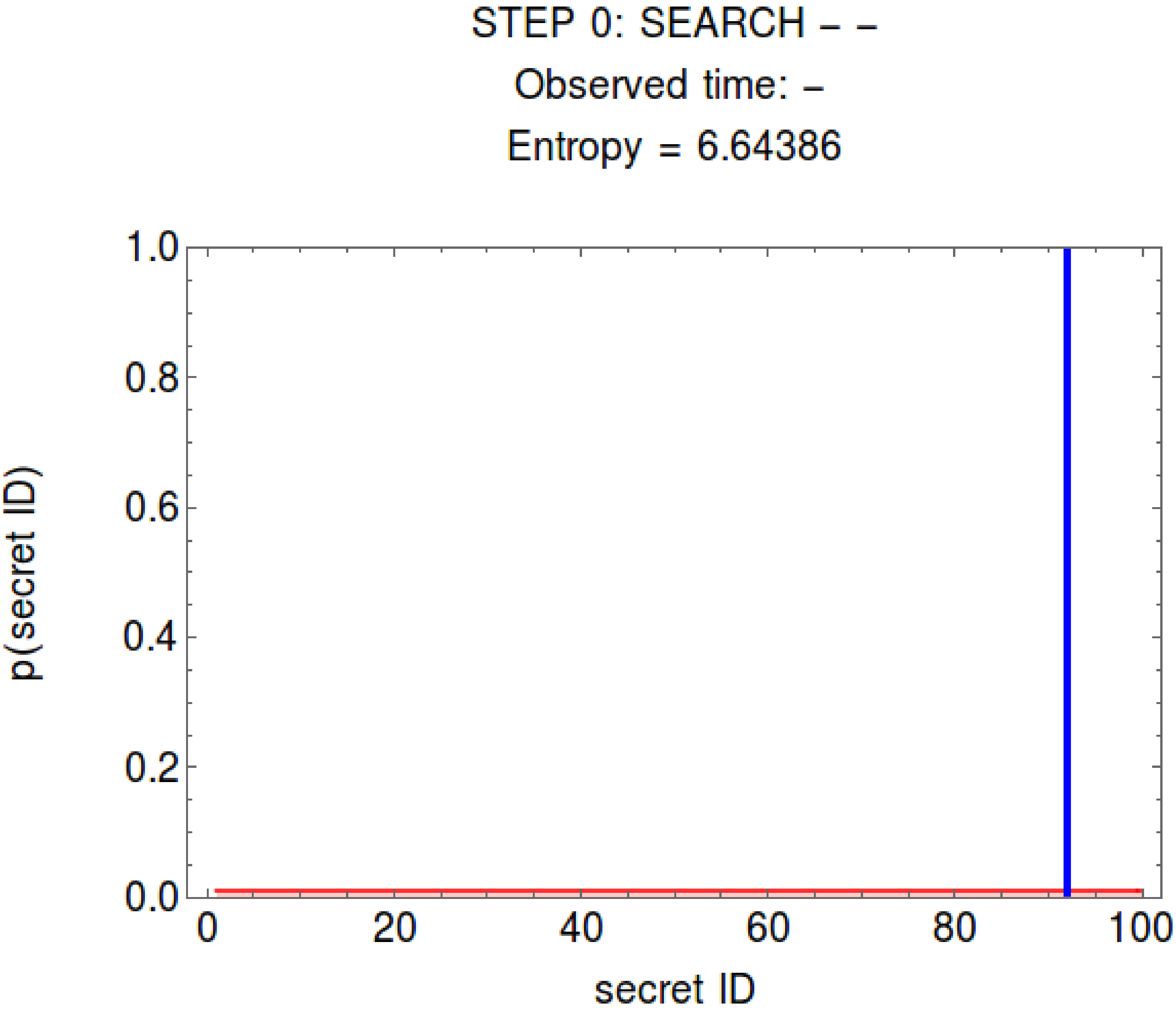
Server

41 classes, 2844 line of code.
DB: key = employee ID
Some employee IDs have restricted access.

Writes to log file depending on

$$ID_{res} \in [minID, maxID]$$

$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

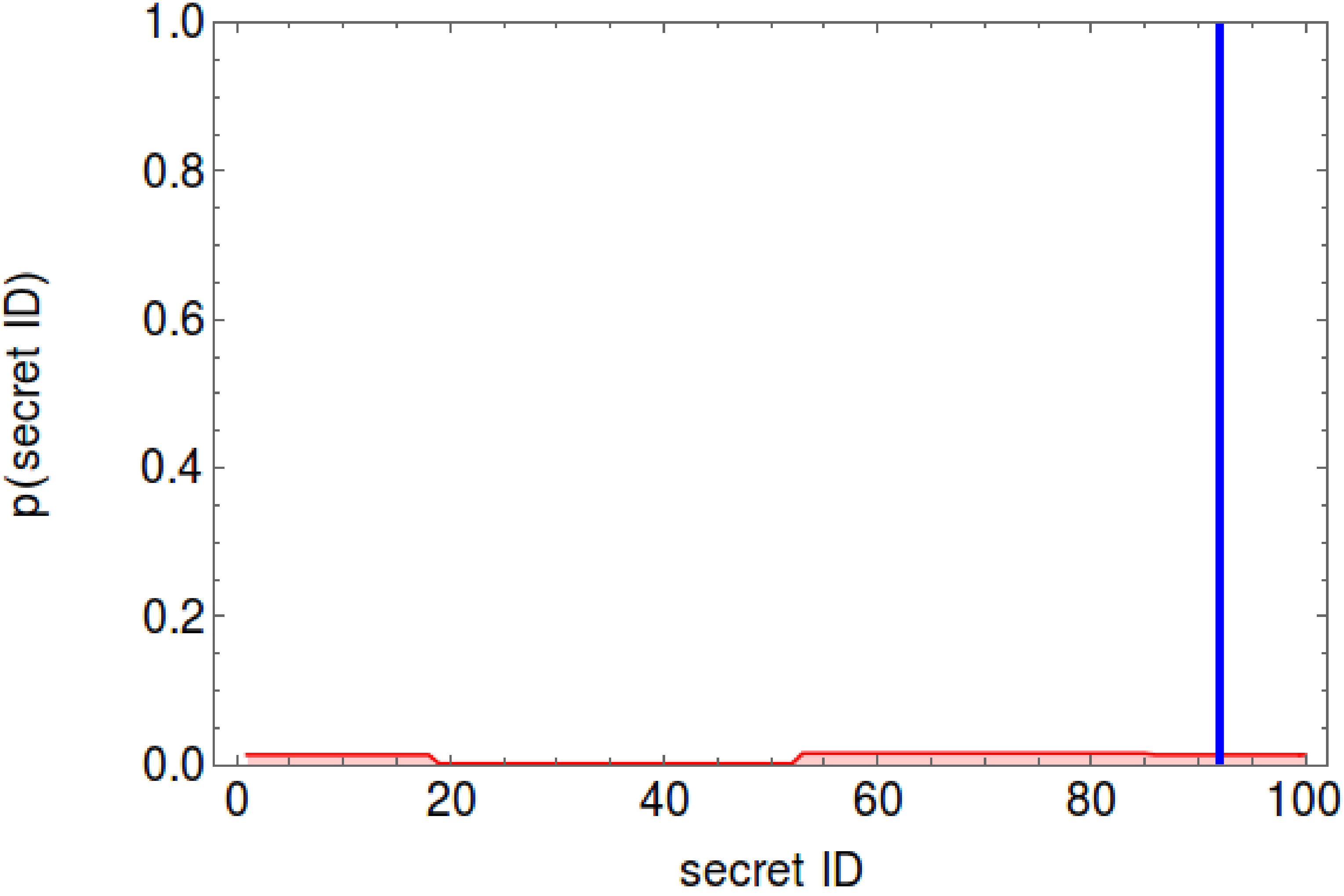


$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

STEP 1: SEARCH 19 52

Observed time:0.00444

Entropy = 6.27408

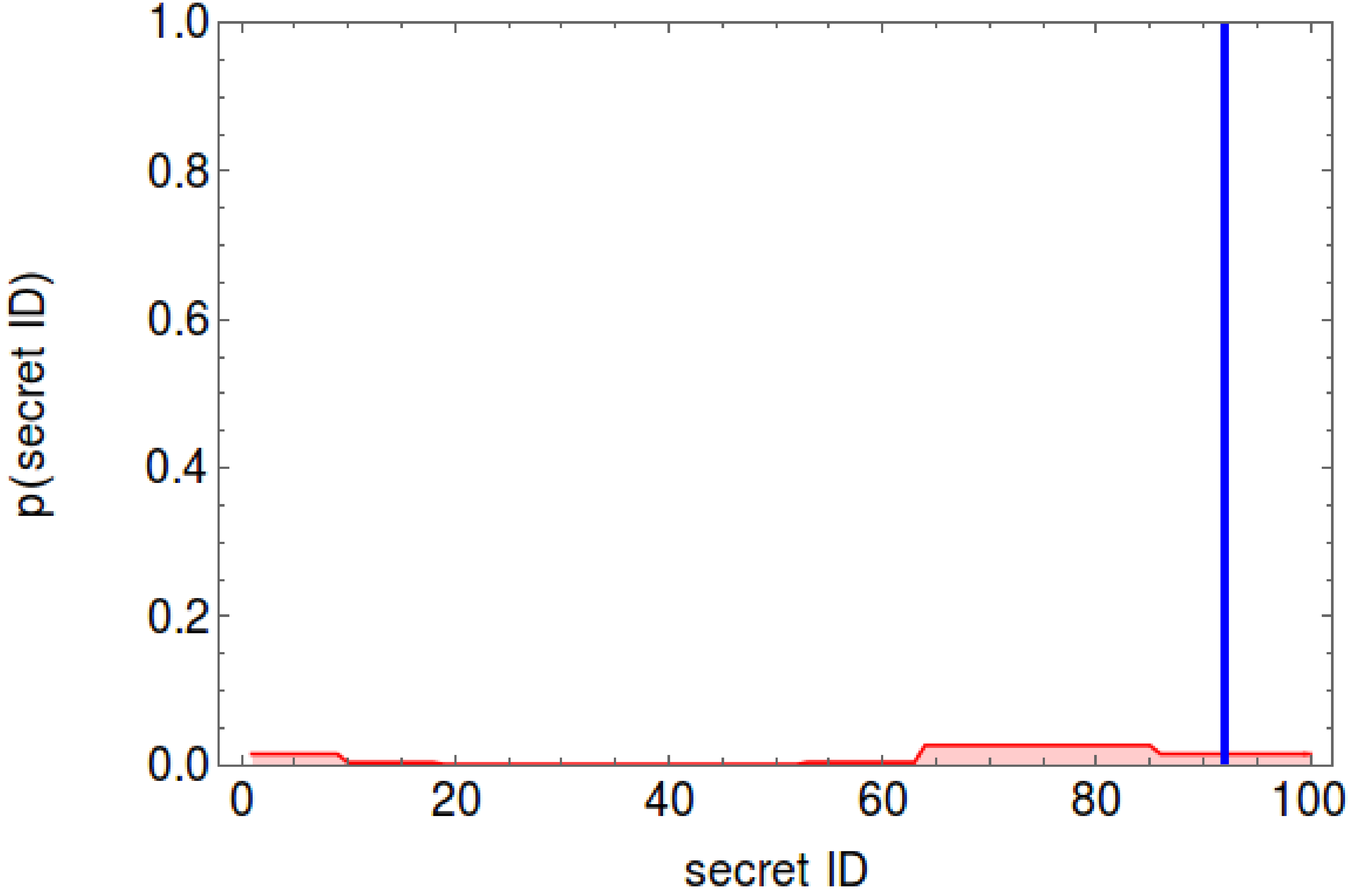


$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

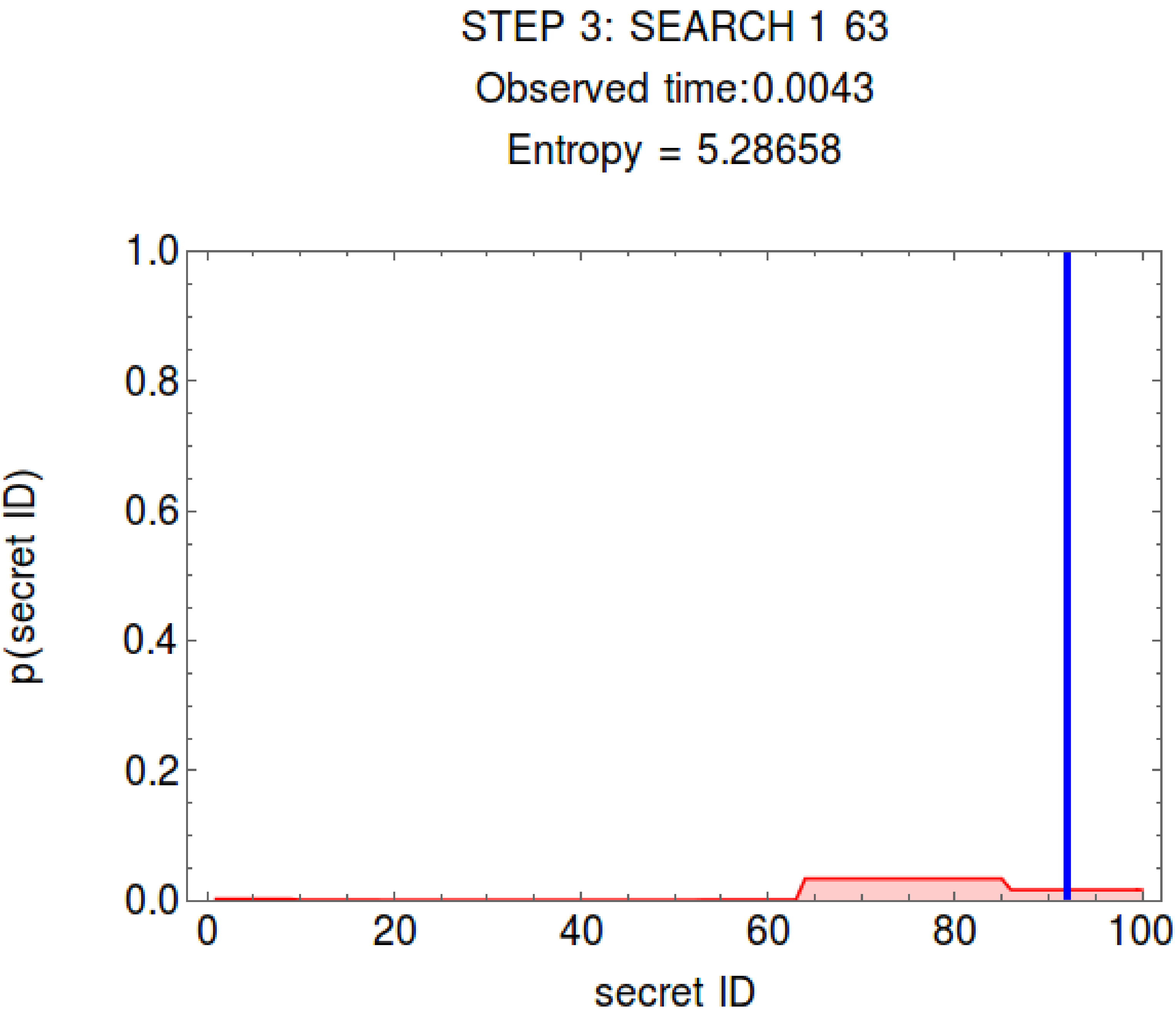
STEP 2: SEARCH 10 63

Observed time:0.00436

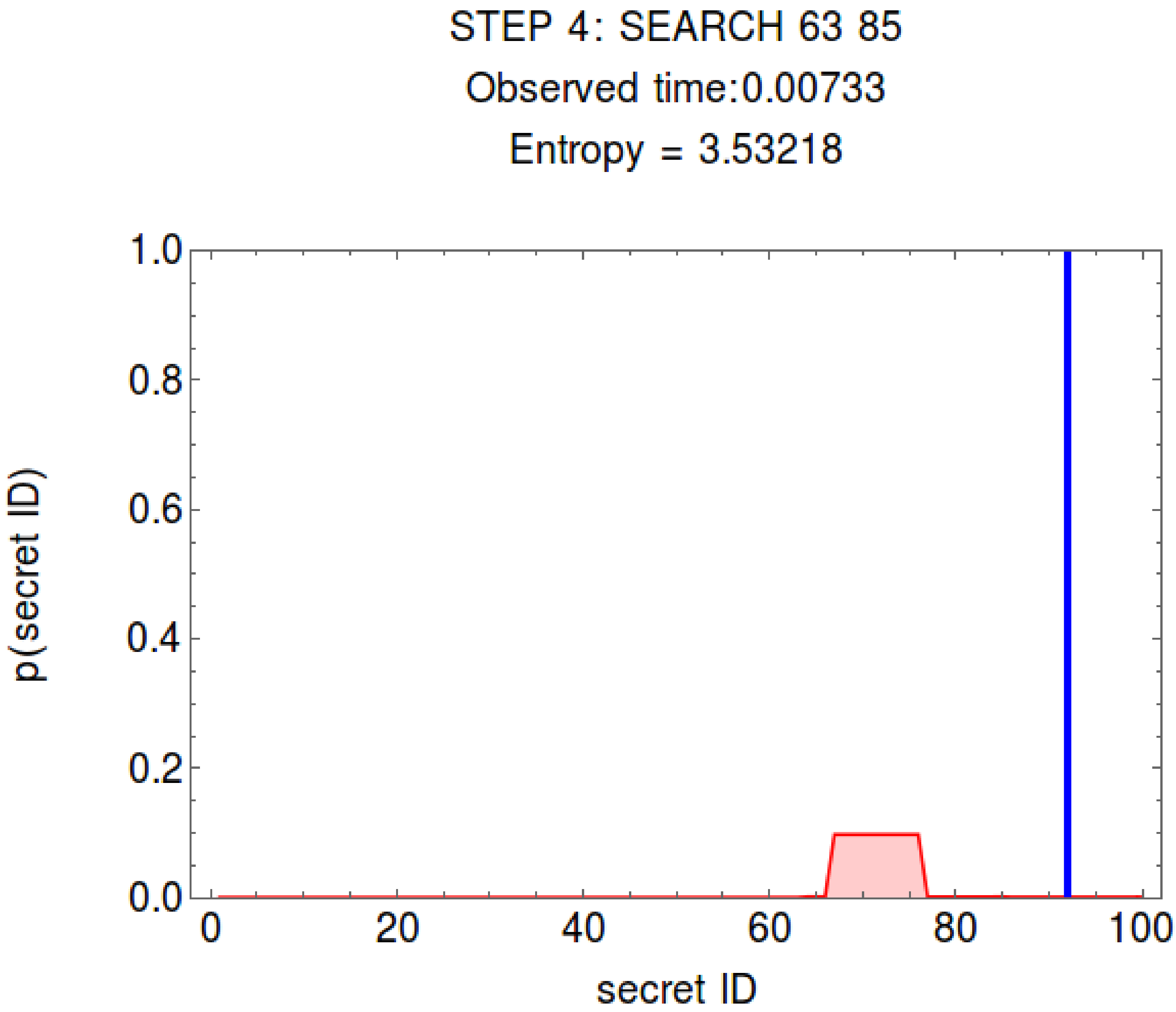
Entropy = 5.81014



$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$



$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

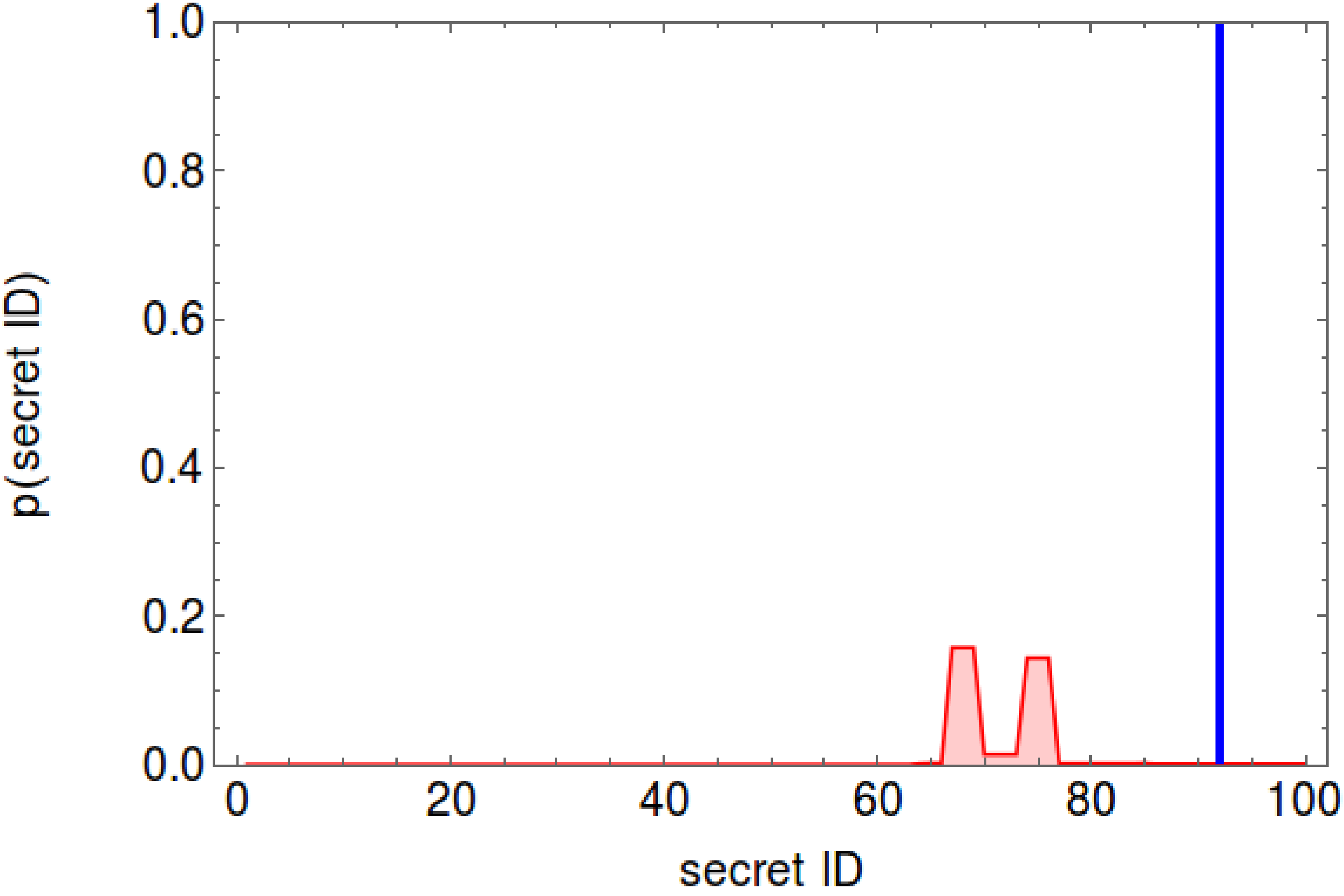


$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

STEP 5: SEARCH 70 73

Observed time:0.00447

Entropy = 3.19249

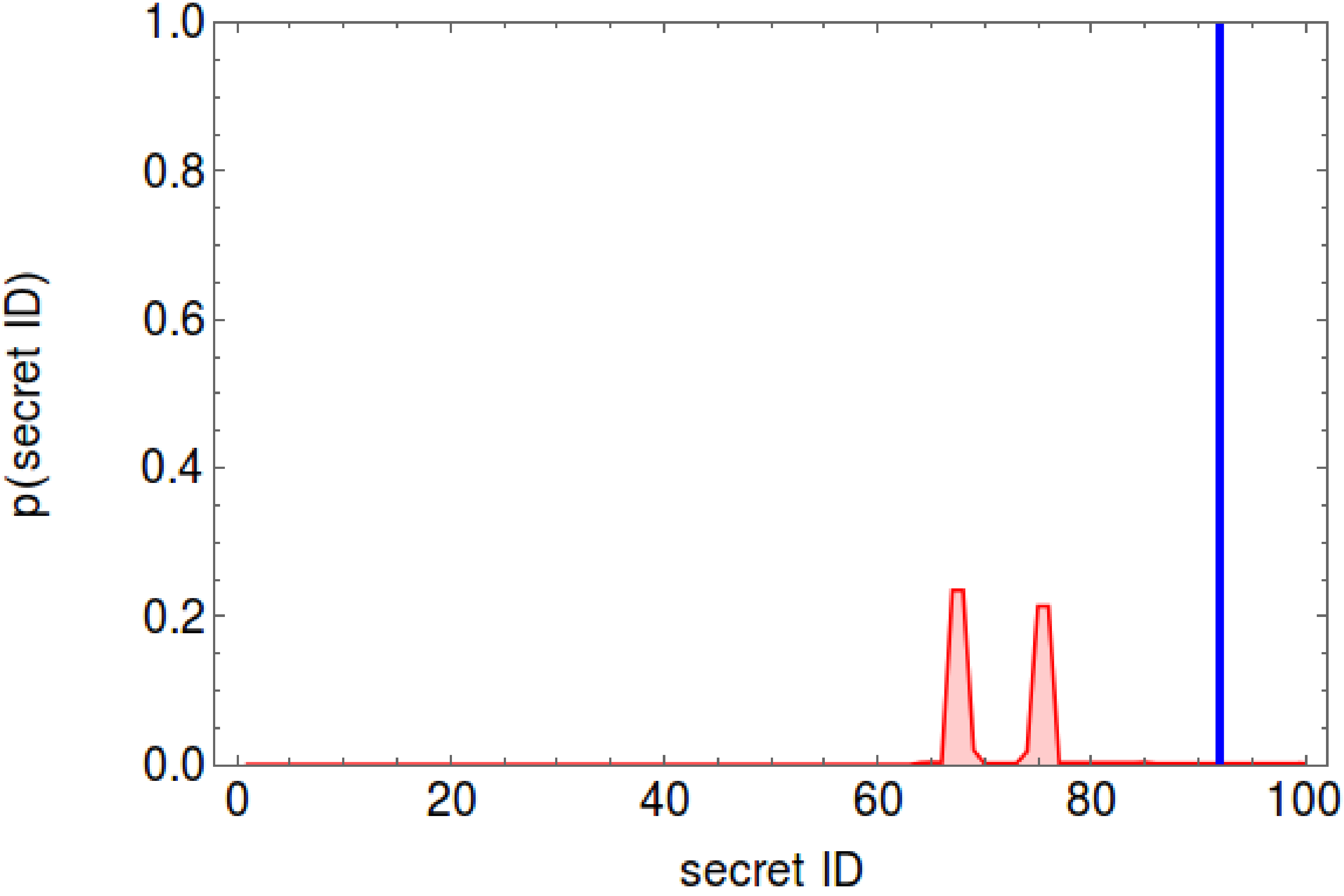


$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

STEP 6: SEARCH 67 74

Observed time:0.00427

Entropy = 2.74012

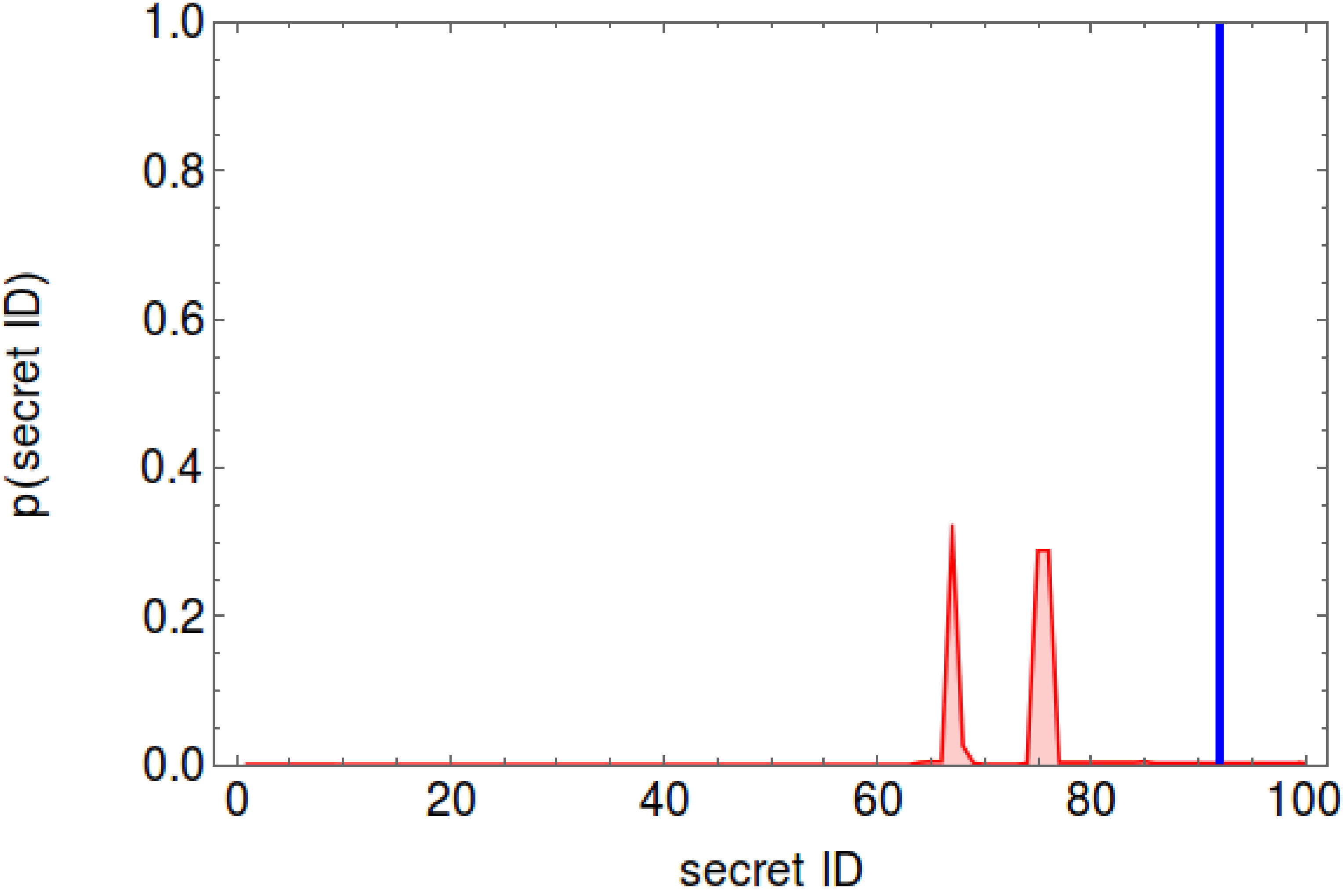


$$1 \le ID \le 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

STEP 7: SEARCH 63 74

Observed time:0.00452

Entropy = 2.41548

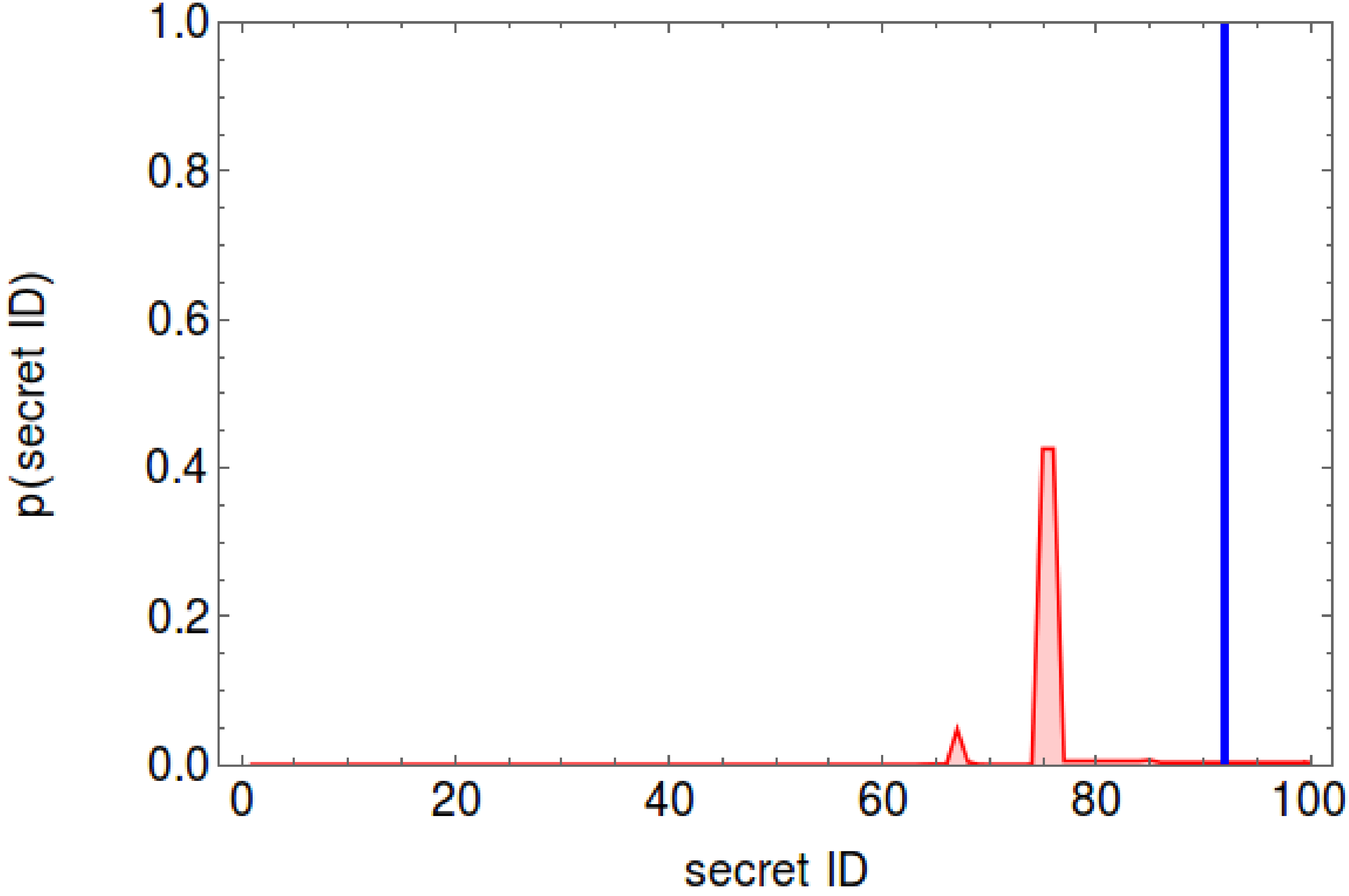


$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

STEP 8: SEARCH 63 70

Observed time:0.00435

Entropy = 2.07286

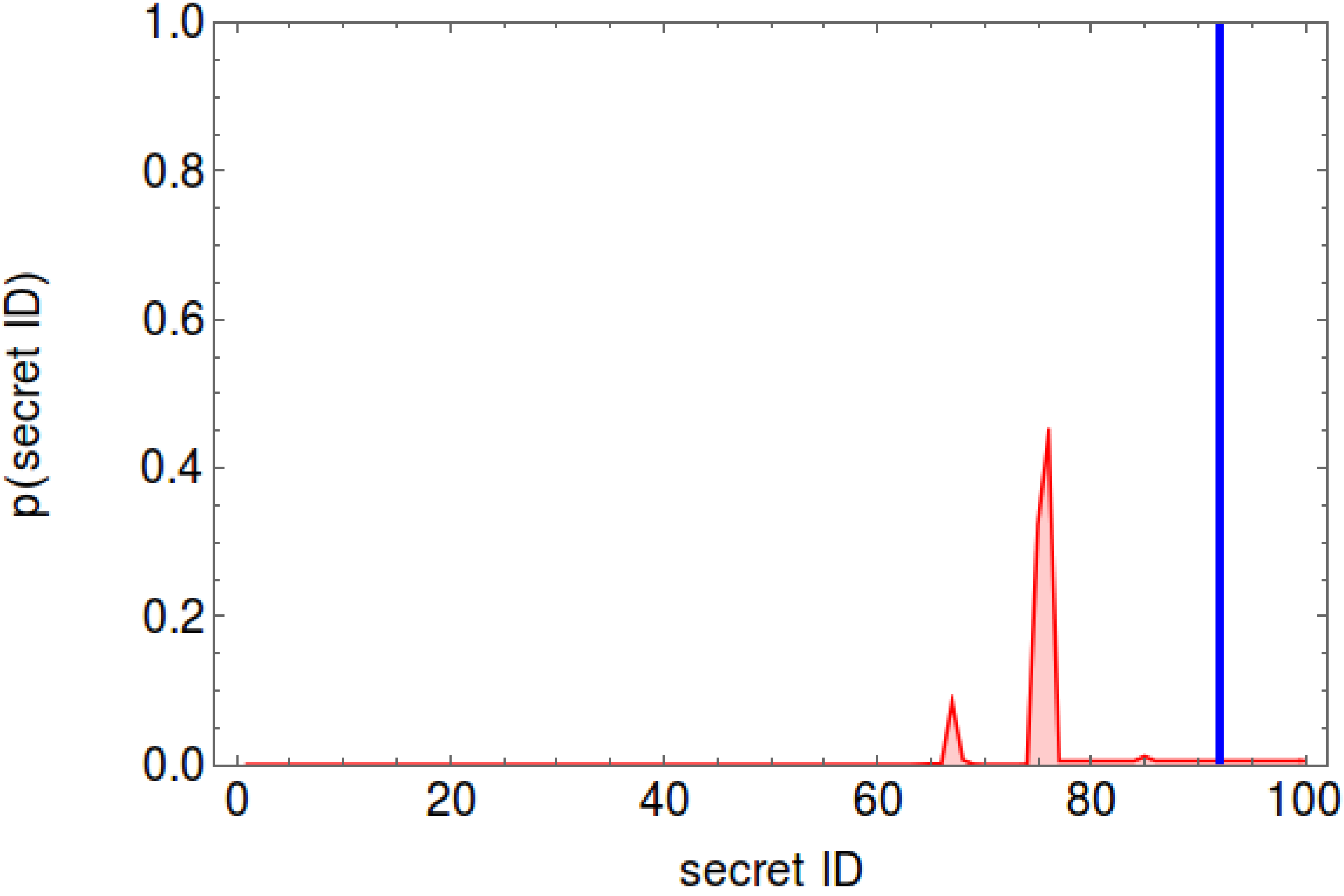


$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 9: SEARCH 74 75

Observed time:0.00431

Entropy = 2.46103

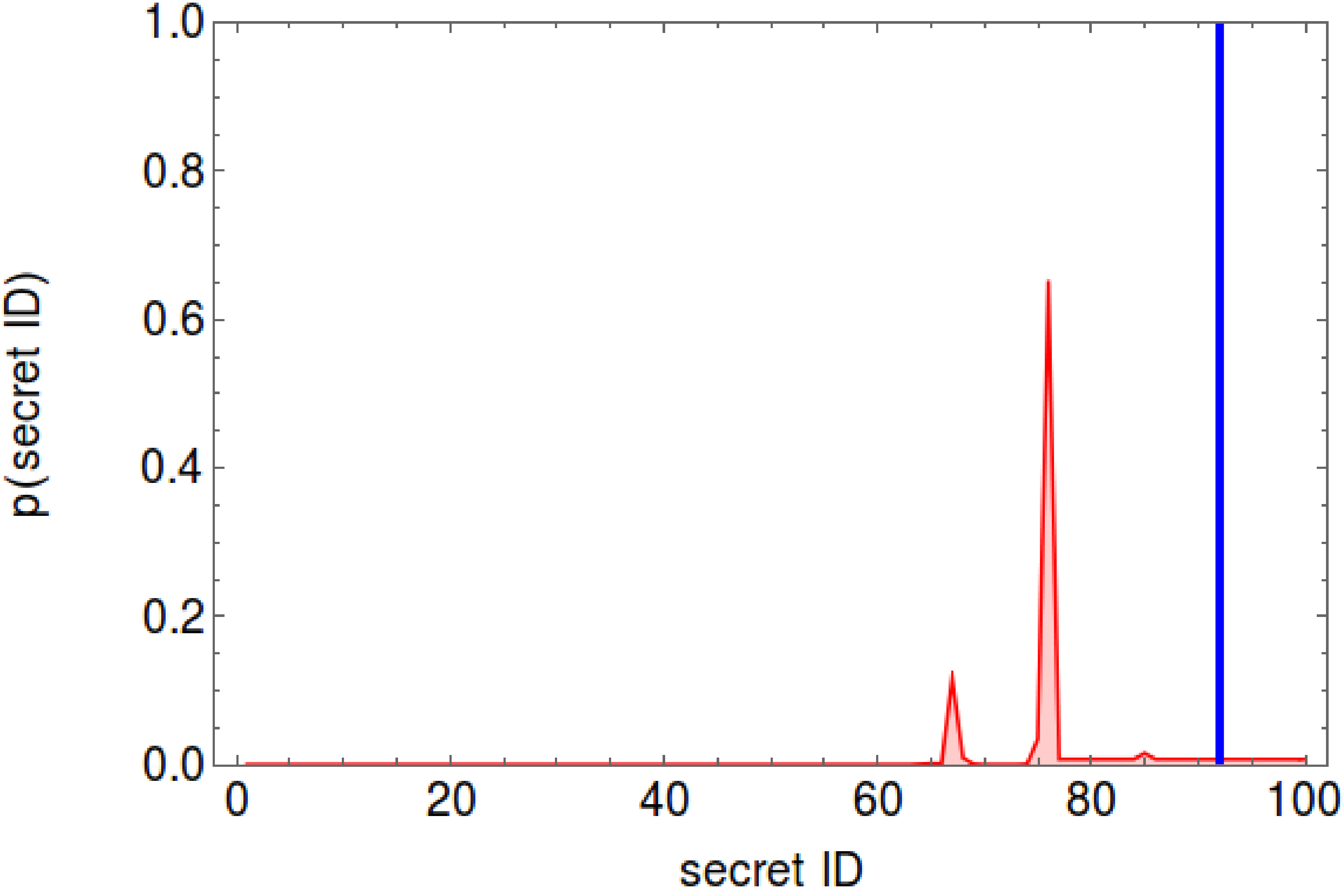


$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

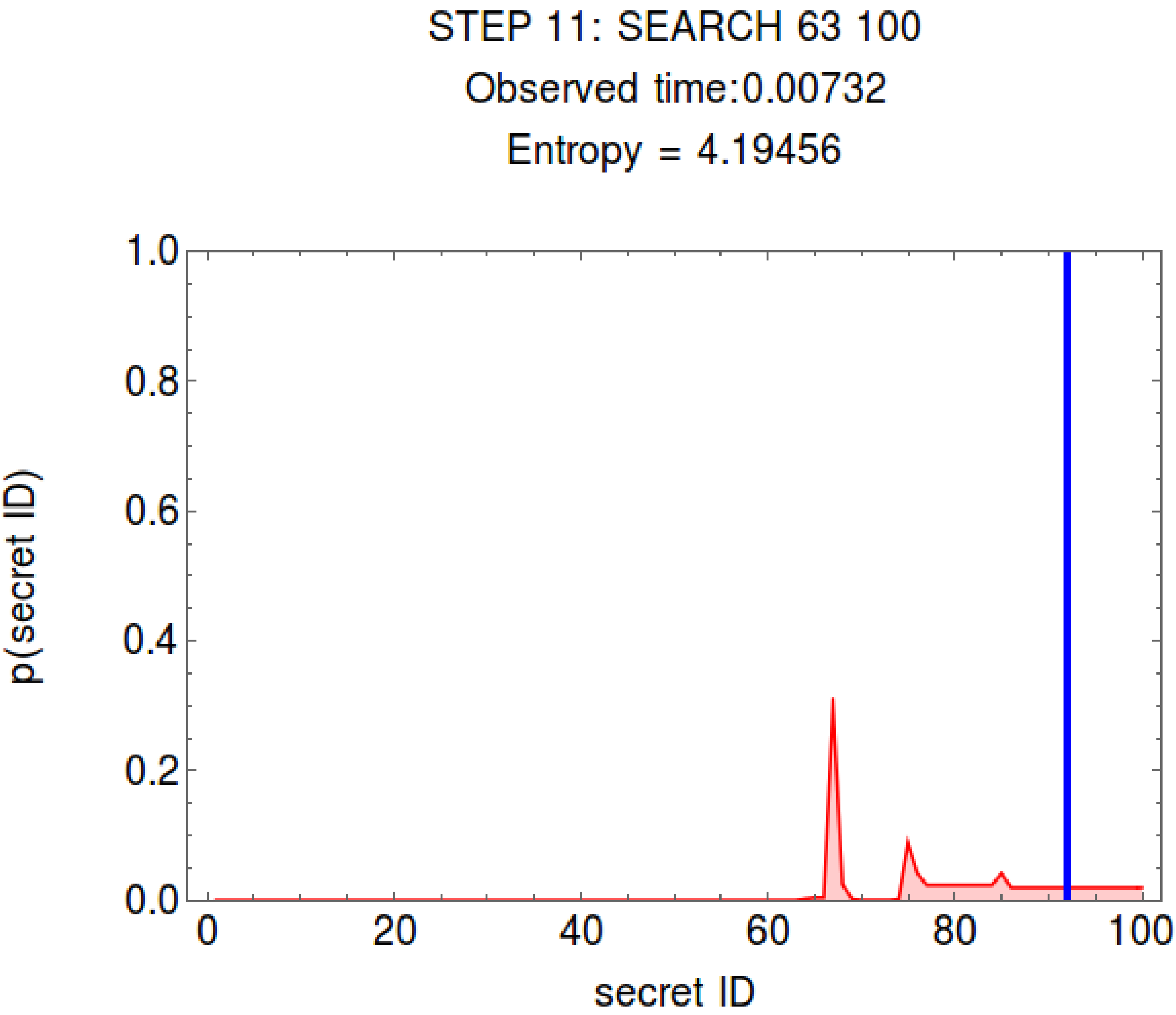
STEP 10: SEARCH 74 75

Observed time:0.00435

Entropy = 2.39414



$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

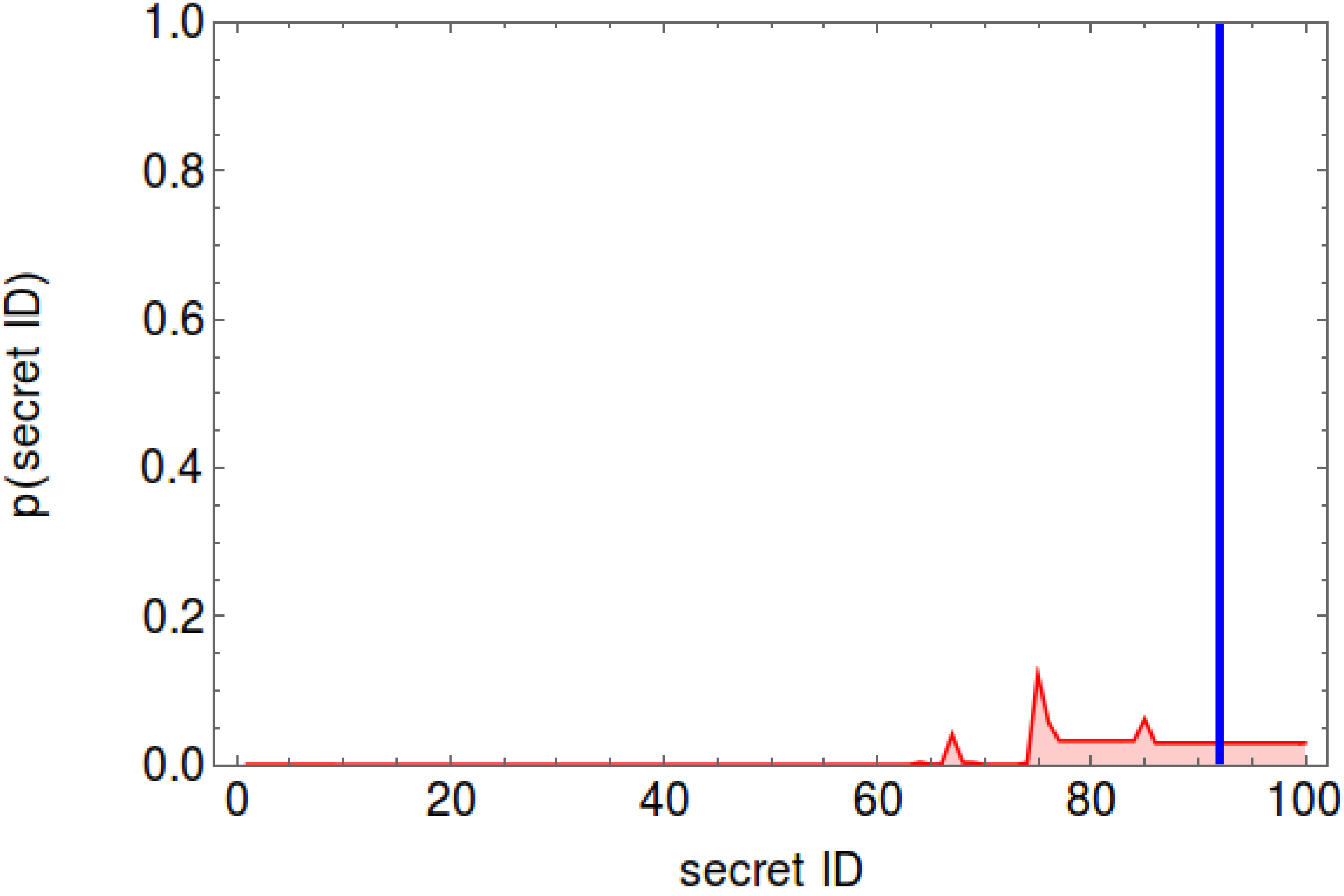


$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

STEP 12: SEARCH 74 100

Observed time:0.00743

Entropy = 4.73142

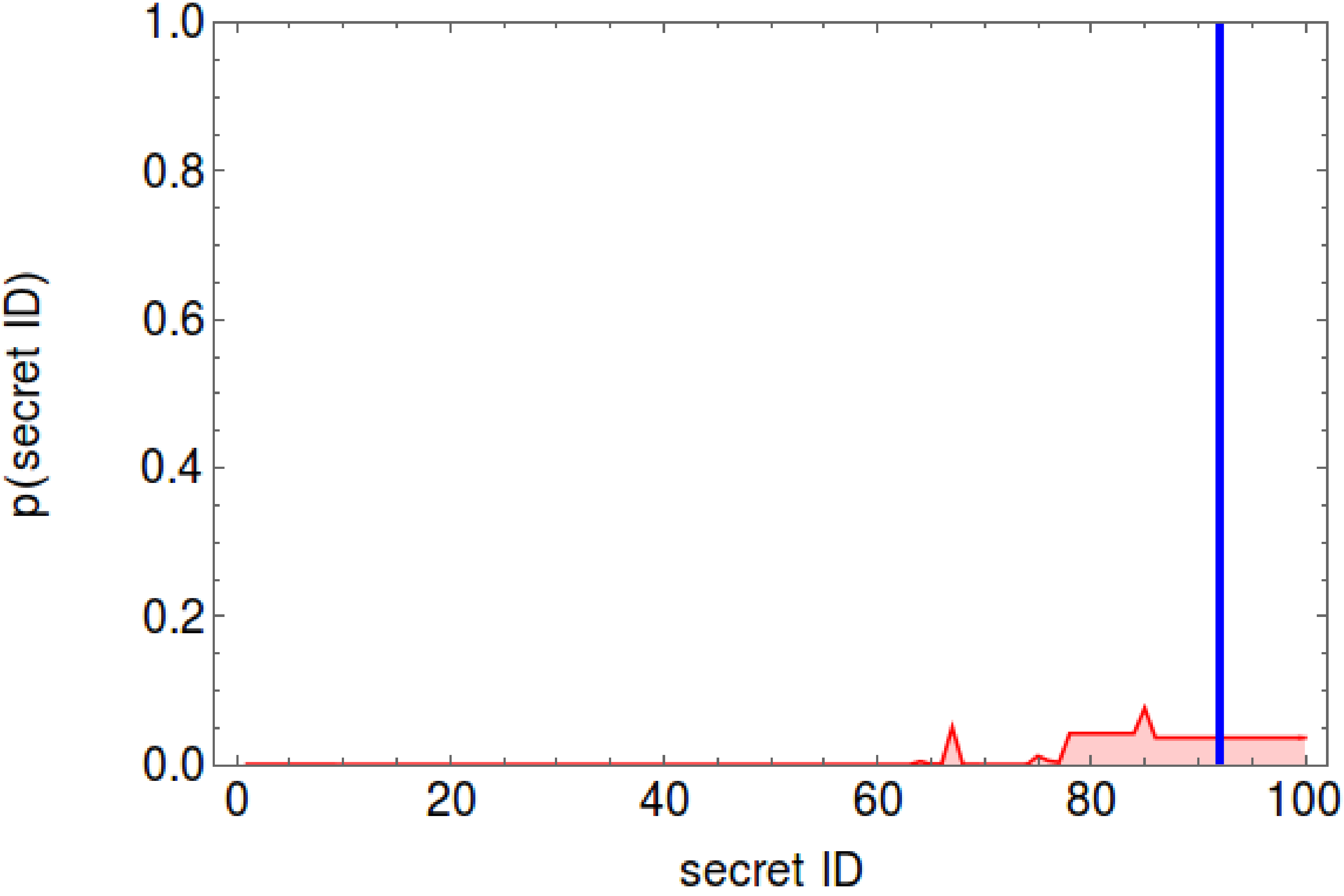


$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 13: SEARCH 78 100

Observed time:0.00733

Entropy = 4.70767

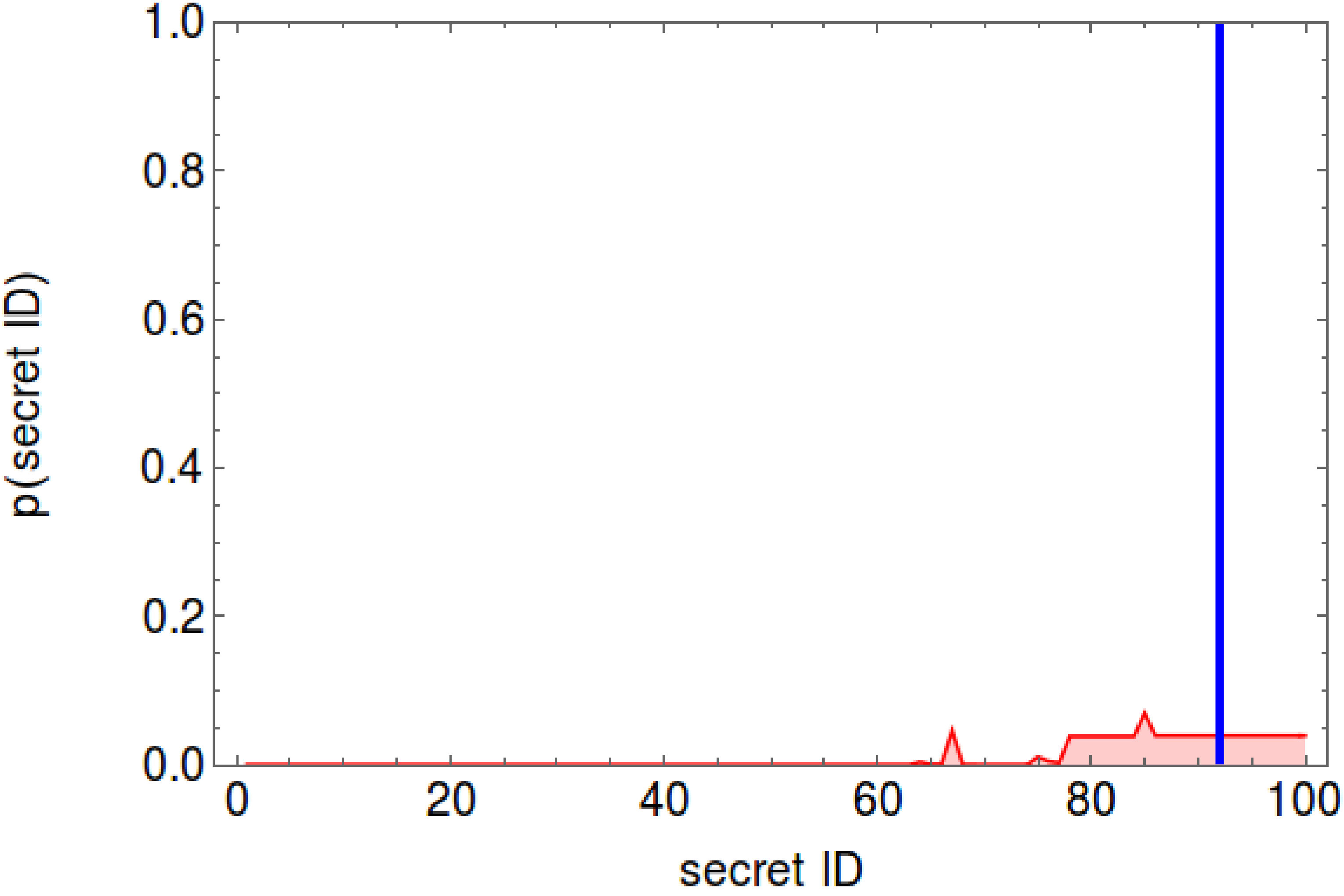


$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

STEP 14: SEARCH 86 100

Observed time:0.00728

Entropy = 4.68363

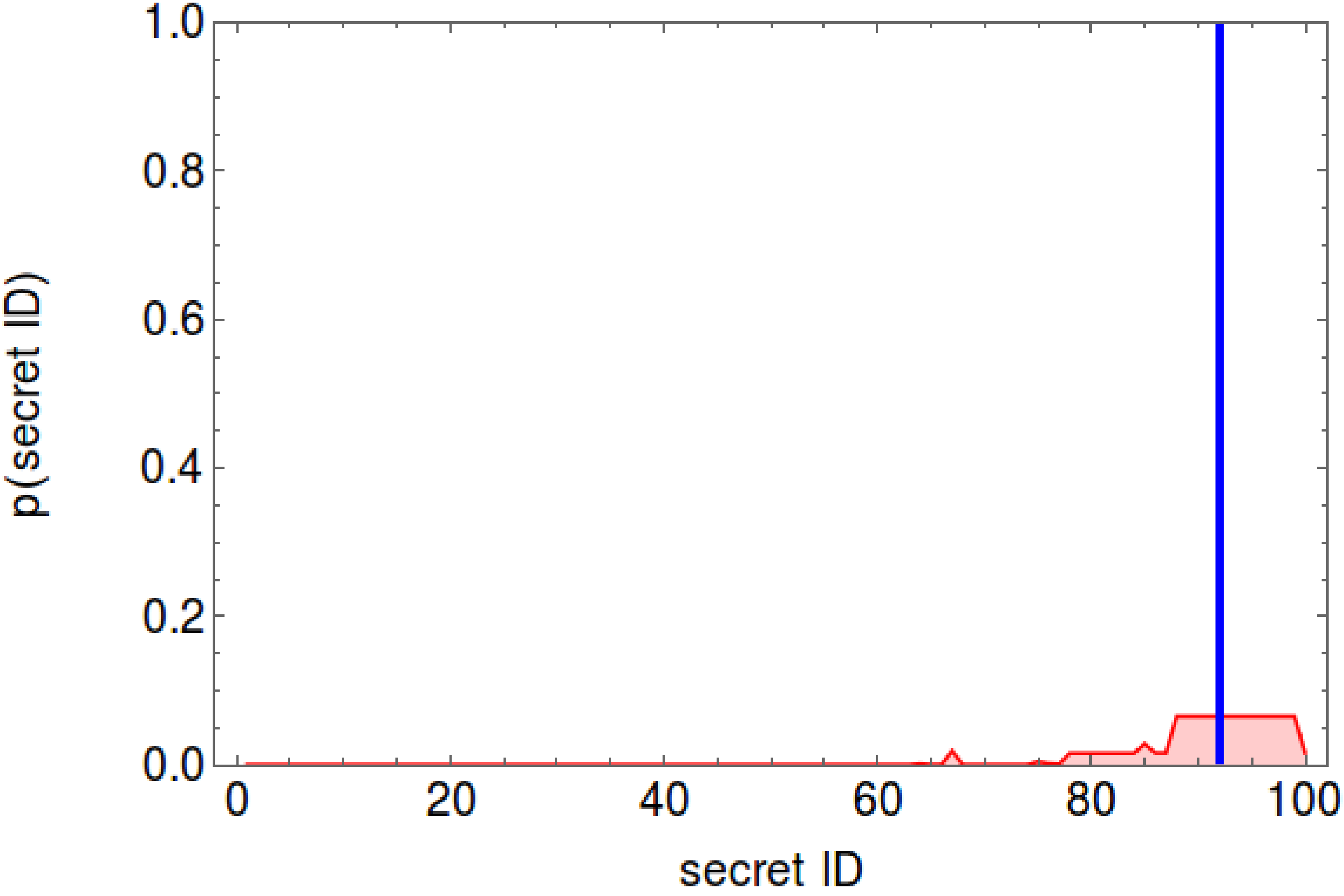


$$1 \leq ID \leq 100 \qquad ID_1 = 64 \qquad ID_2 = 85 \qquad ID_{res} = 92$$

STEP 15: SEARCH 87 99

Observed time:0.00716

Entropy = 4.37901

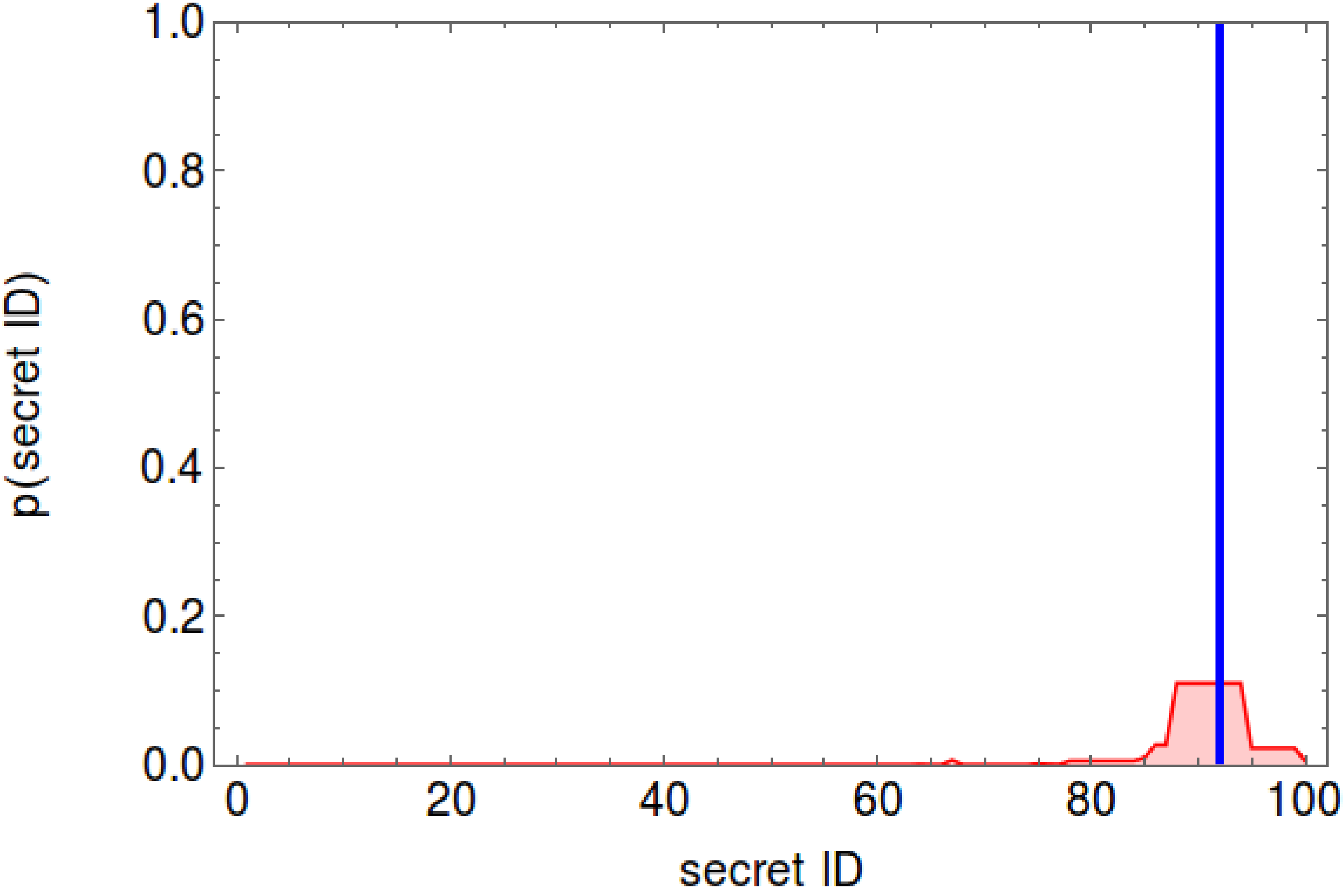


$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

STEP 16: SEARCH 87 95

Observed time:0.00727

Entropy = 3.83405

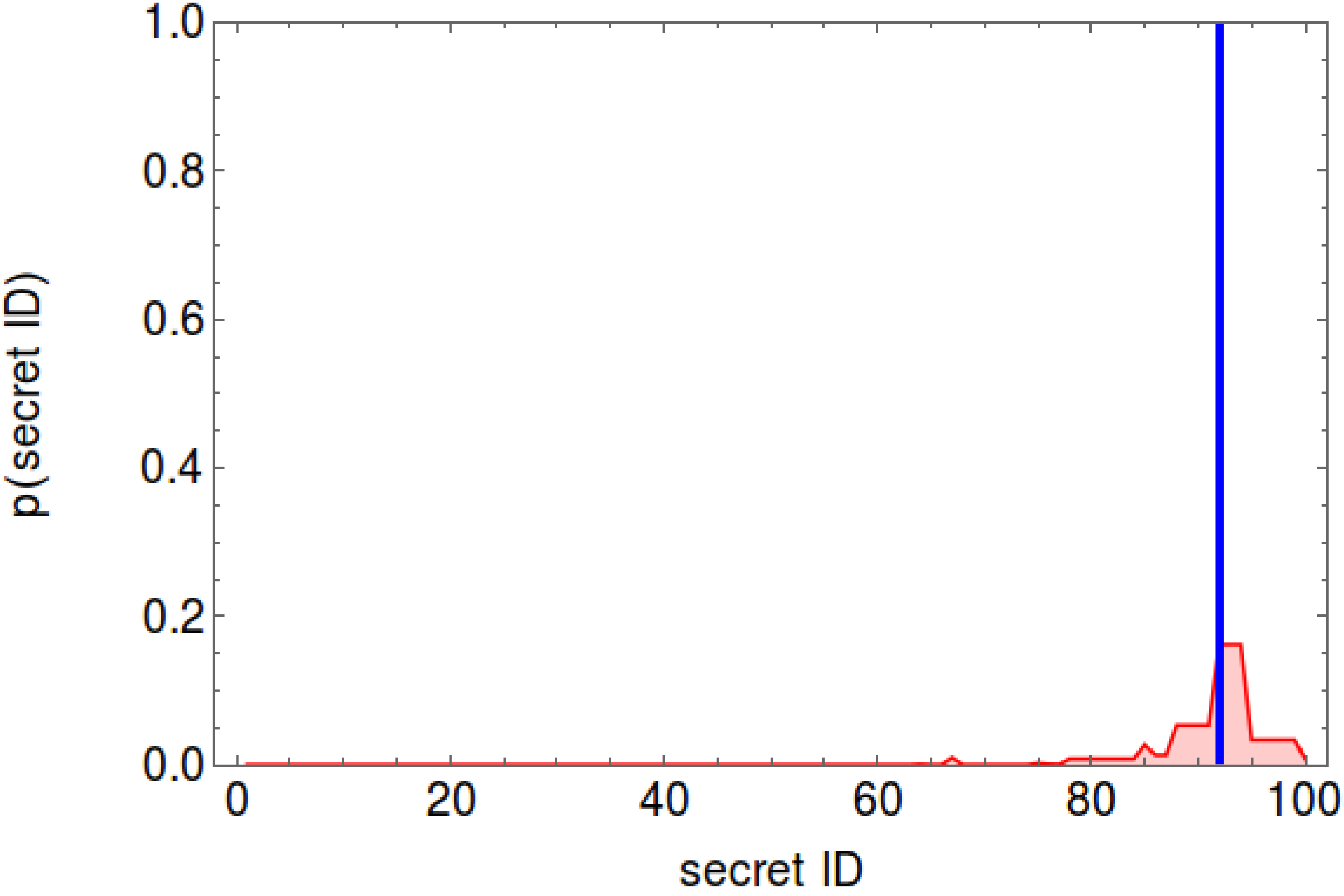


$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

STEP 17: SEARCH 91 95

Observed time:0.00731

Entropy = 3.87438

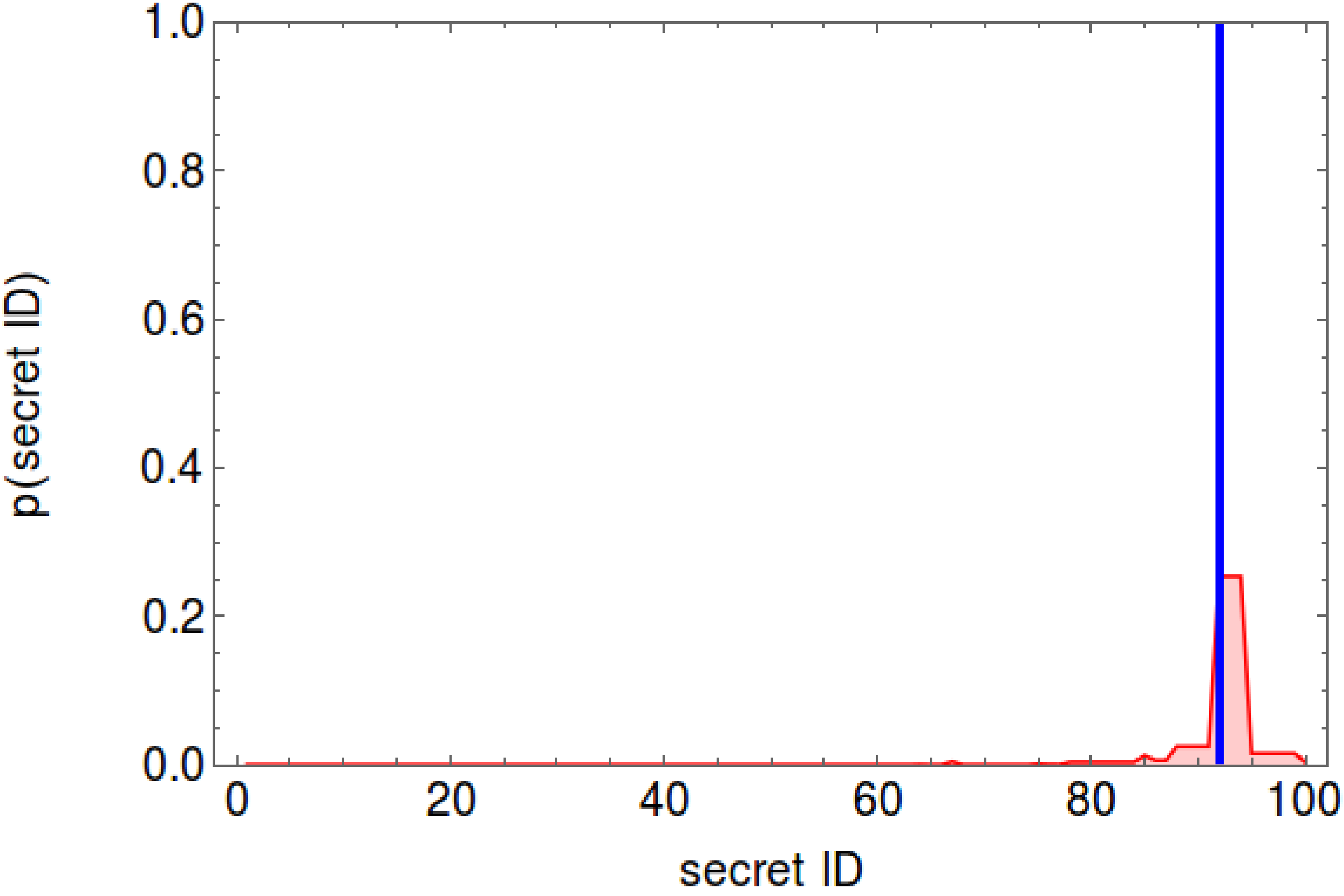


$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

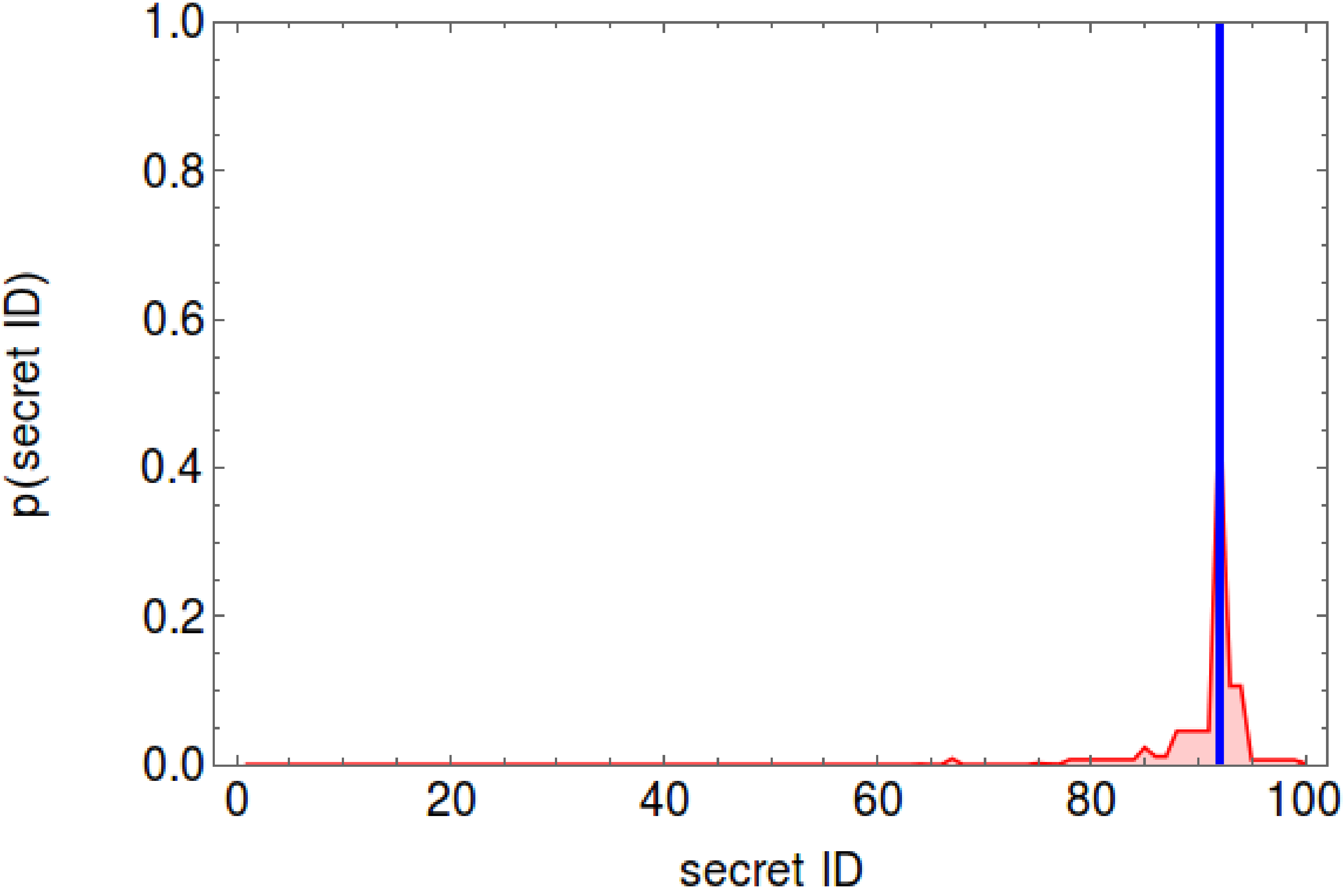
STEP 18: SEARCH 92 95

Observed time:0.0072

Entropy = 2.9822



$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

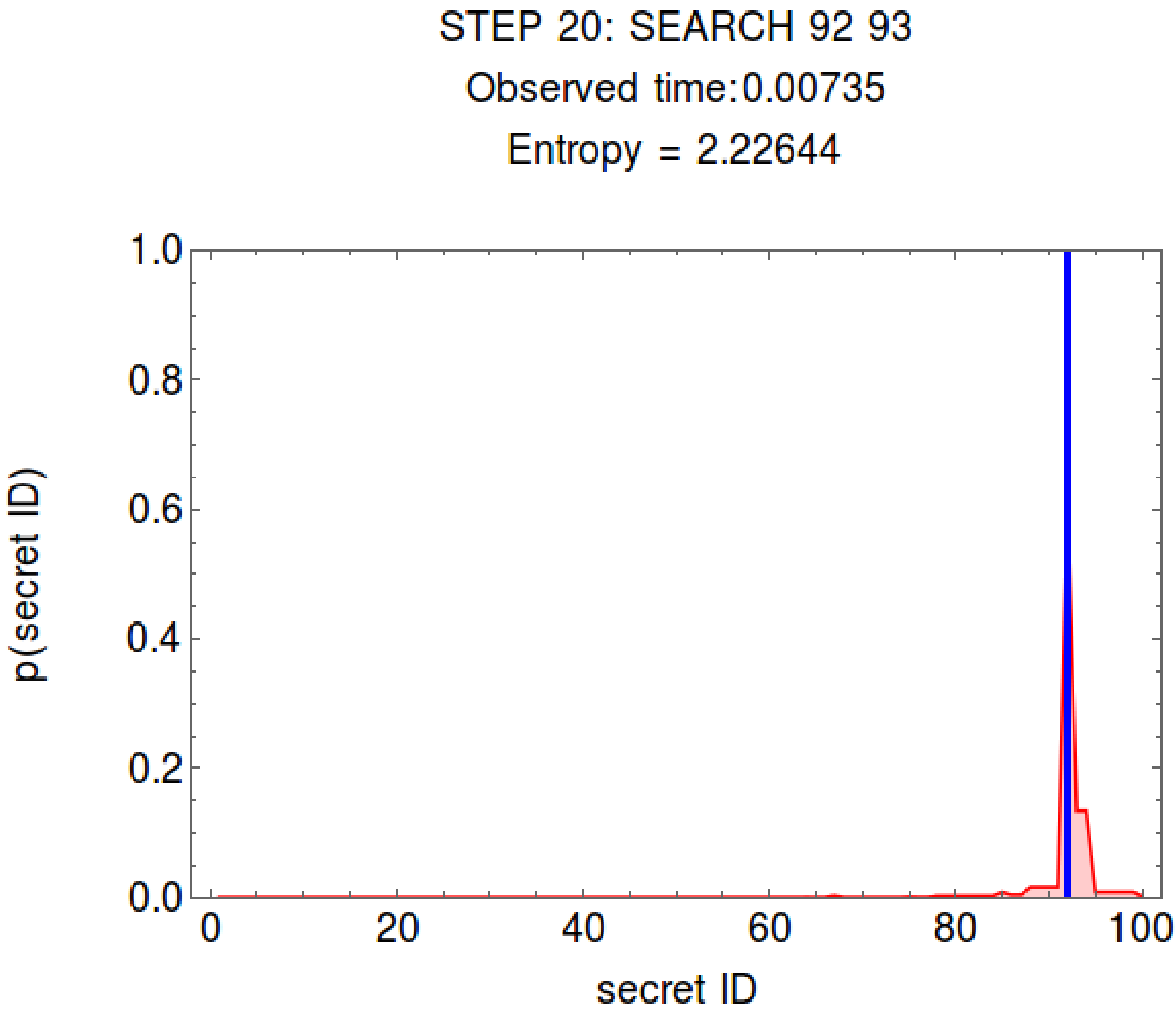


STEP 19: SEARCH 92 94

Observed time:0.00729

Entropy = 2.98878

$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

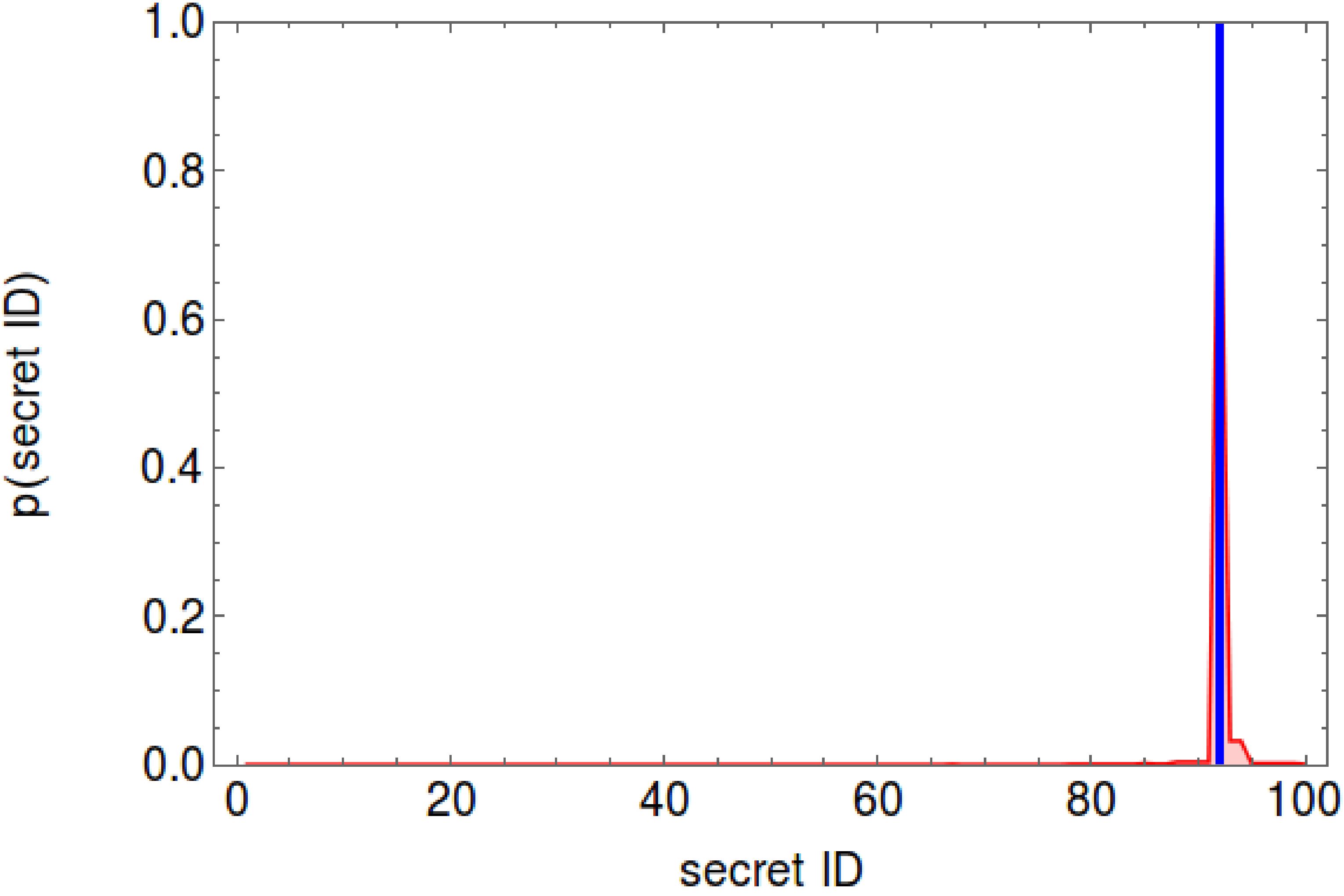


$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

STEP 21: SEARCH 92 92

Observed time:0.00739

Entropy = 0.767476

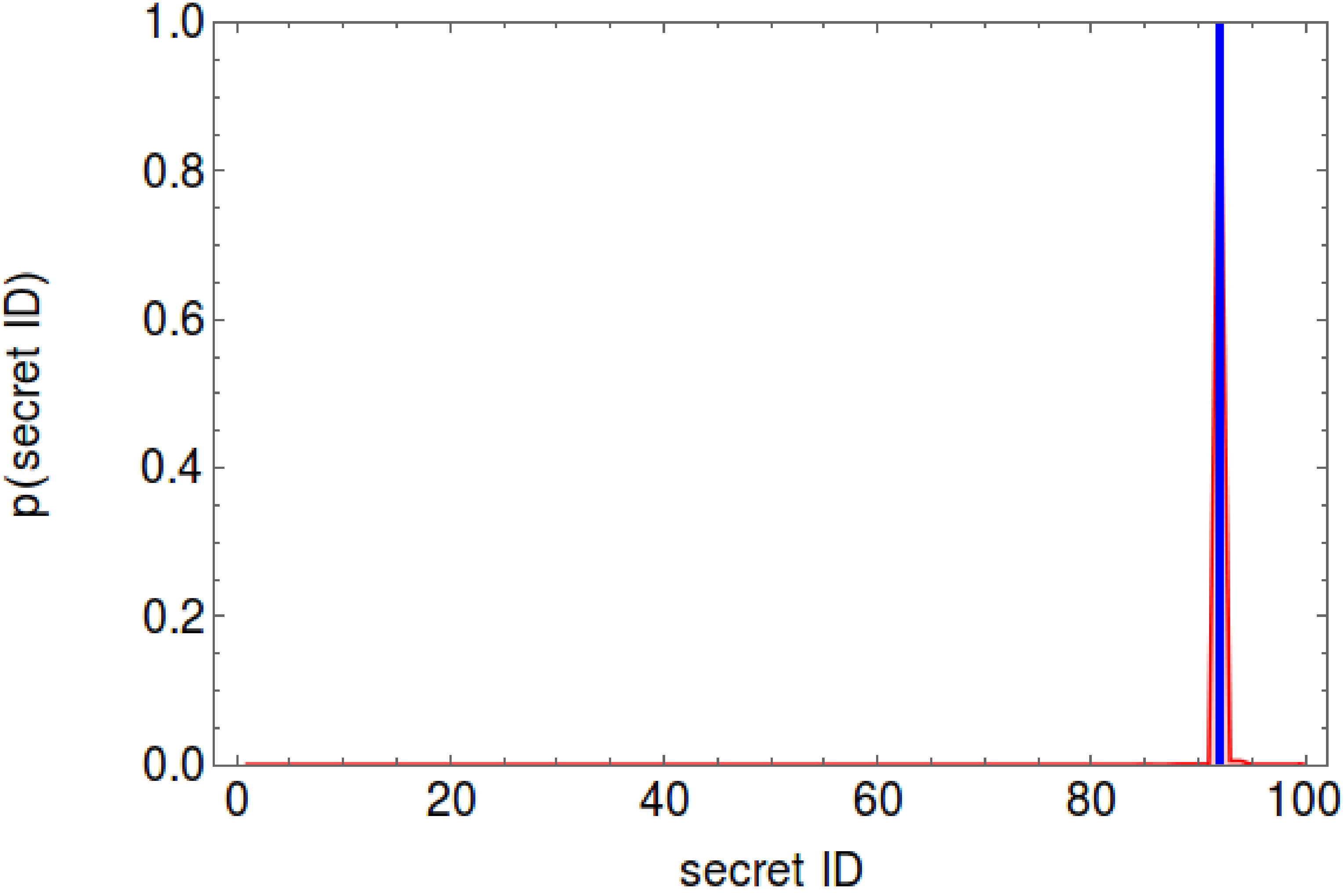


$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

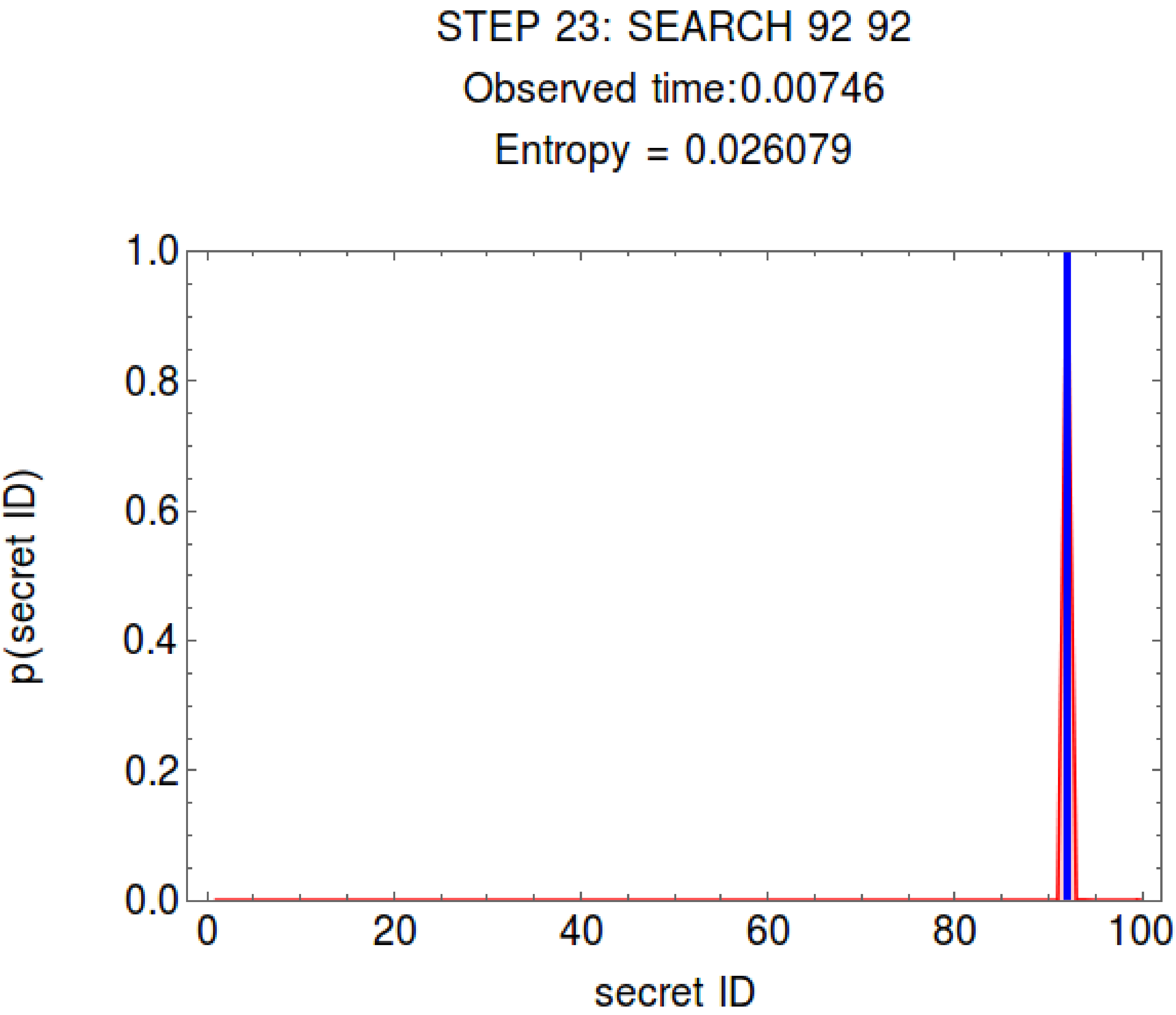
STEP 22: SEARCH 92 92

Observed time:0.00715

Entropy = 0.170871



$$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$$

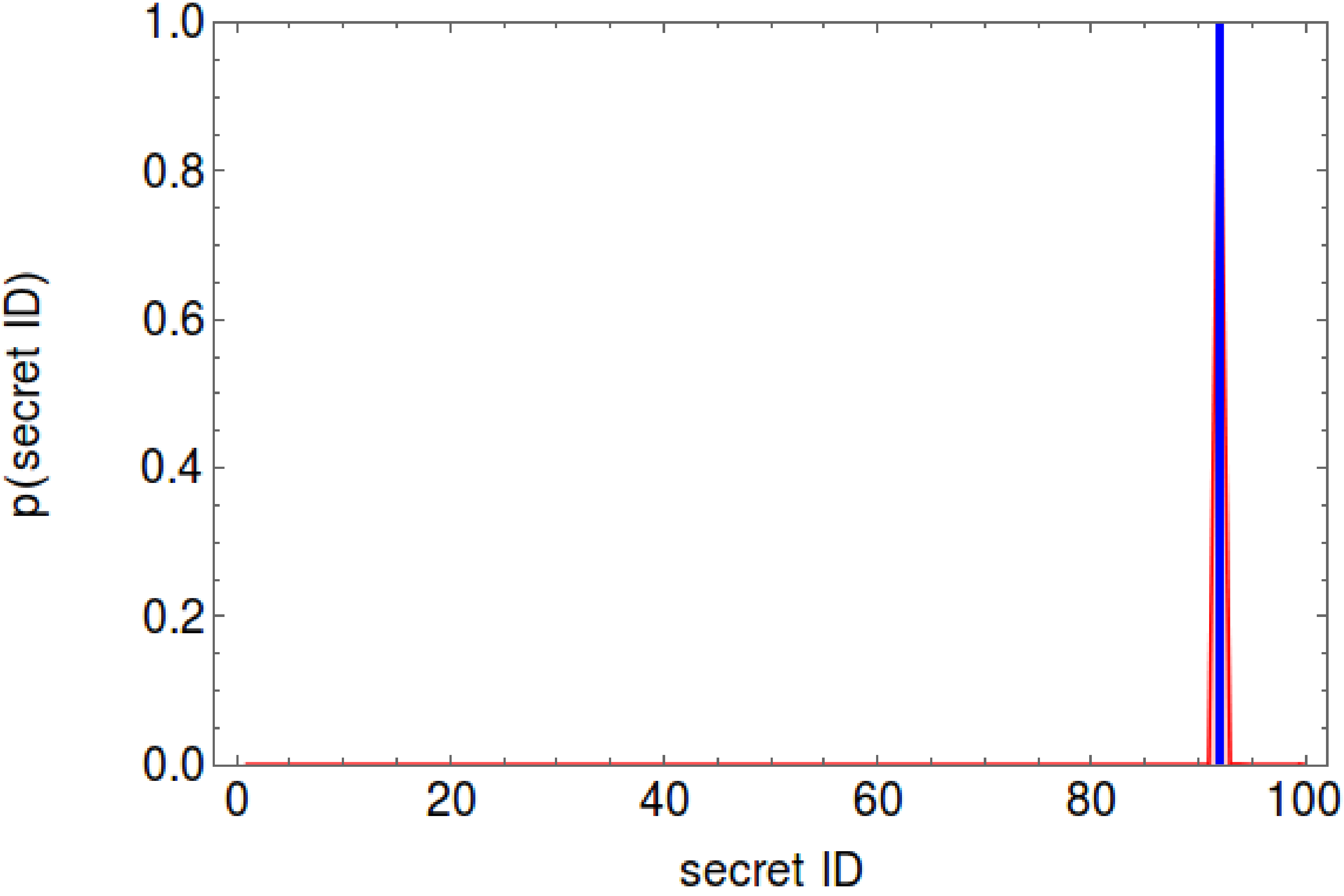


$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$

STEP 24: SEARCH 92 92

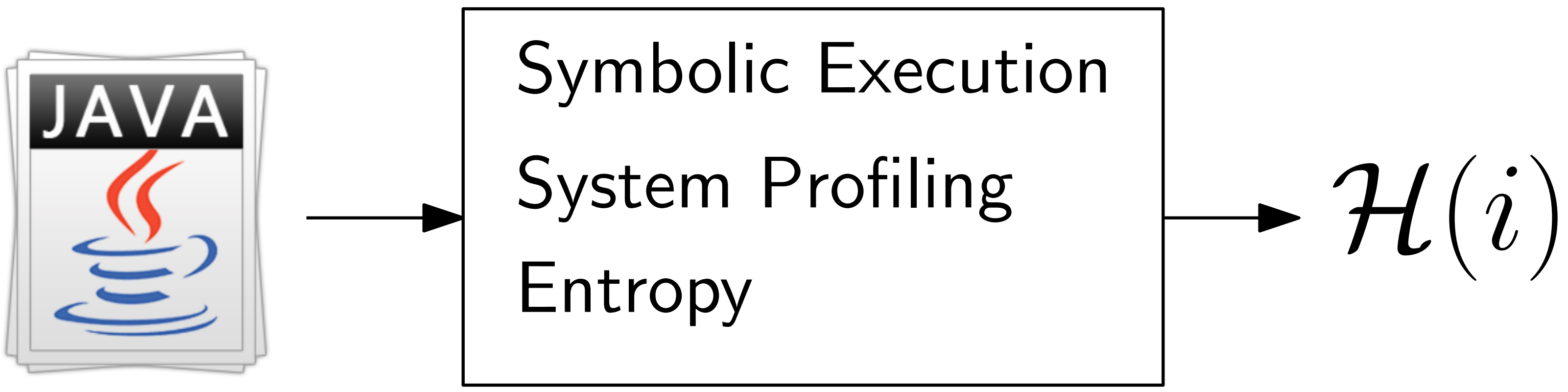
Observed time:0.00721

Entropy = 0.026084



ID Range	# Employees	Offline Analysis	Attack	
			time (m)	# steps
1-100	3	57s	2m38s	25
1-10000	4	2m21s	2m43s	45
1-10000	5	6m30s	3m08s	48
1-10000	10	42m09s	4m31s	77

Our Approach

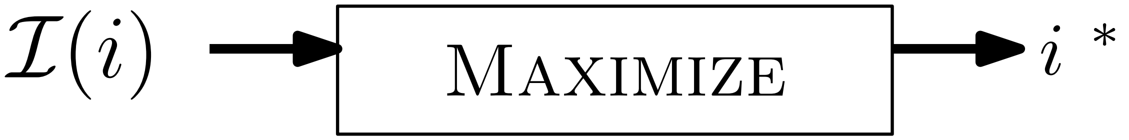


Symbolic Execution

System Profiling

Entropy

$\mathcal{H}(i)$



$\mathcal{I}(i)$

Maximize

$i^{\,*}$

Automatically synthesize side-channel attacks!

Thanks!

Questions?