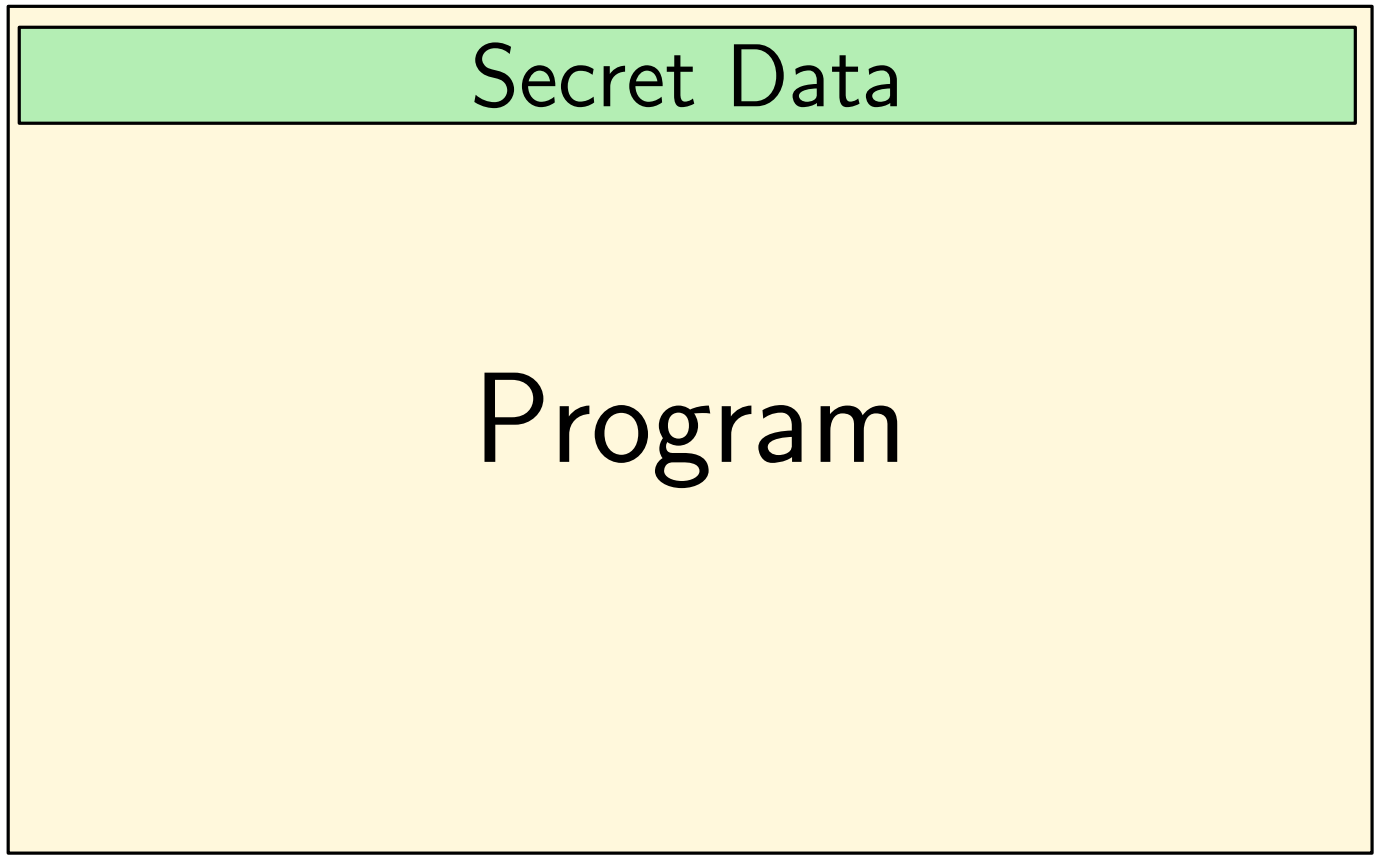
Online Synthesis of Adaptive Side-Channel Attacks Based On Noisy Observations

Lucas Bang, Nicolás Rosner, Tevfik Bultan

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

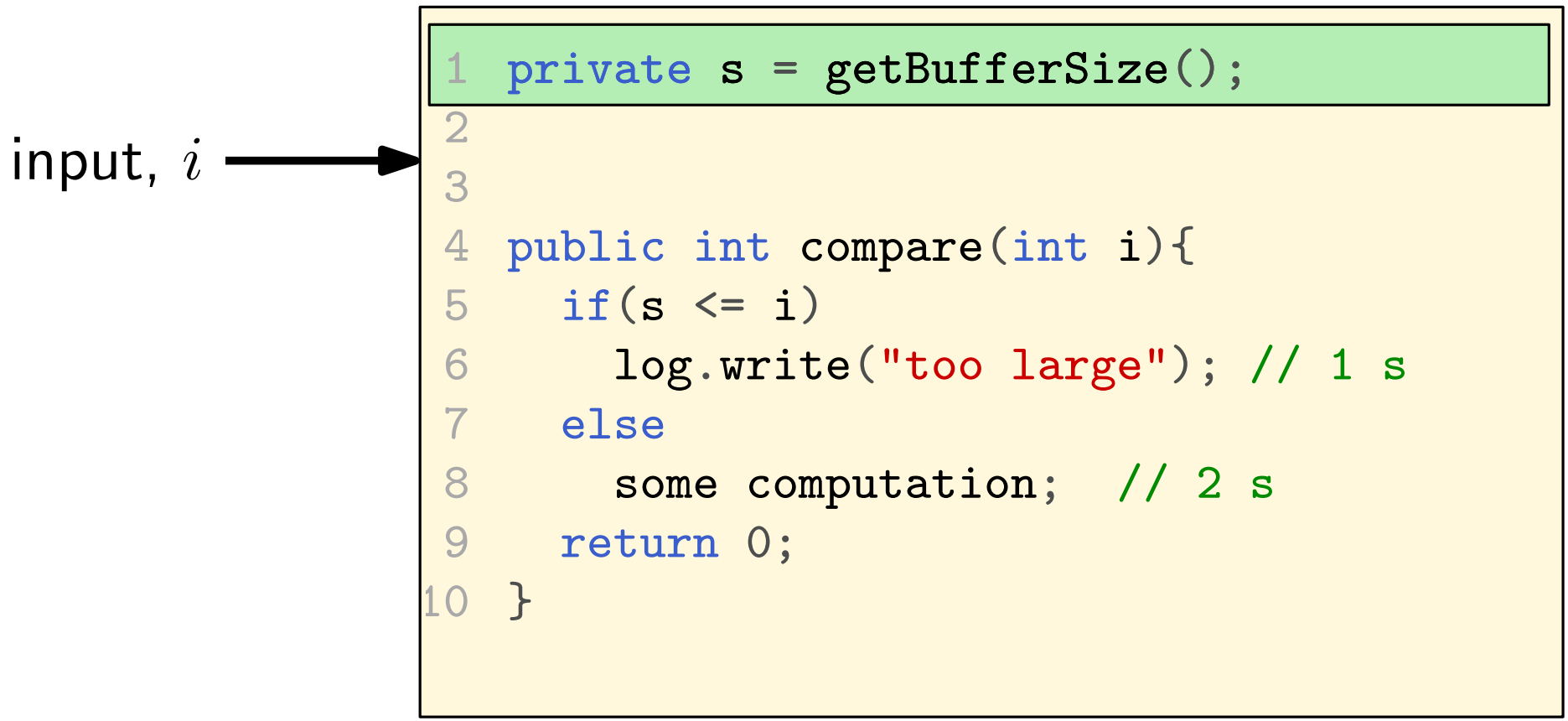
Adaptive Side-Channel Attacks

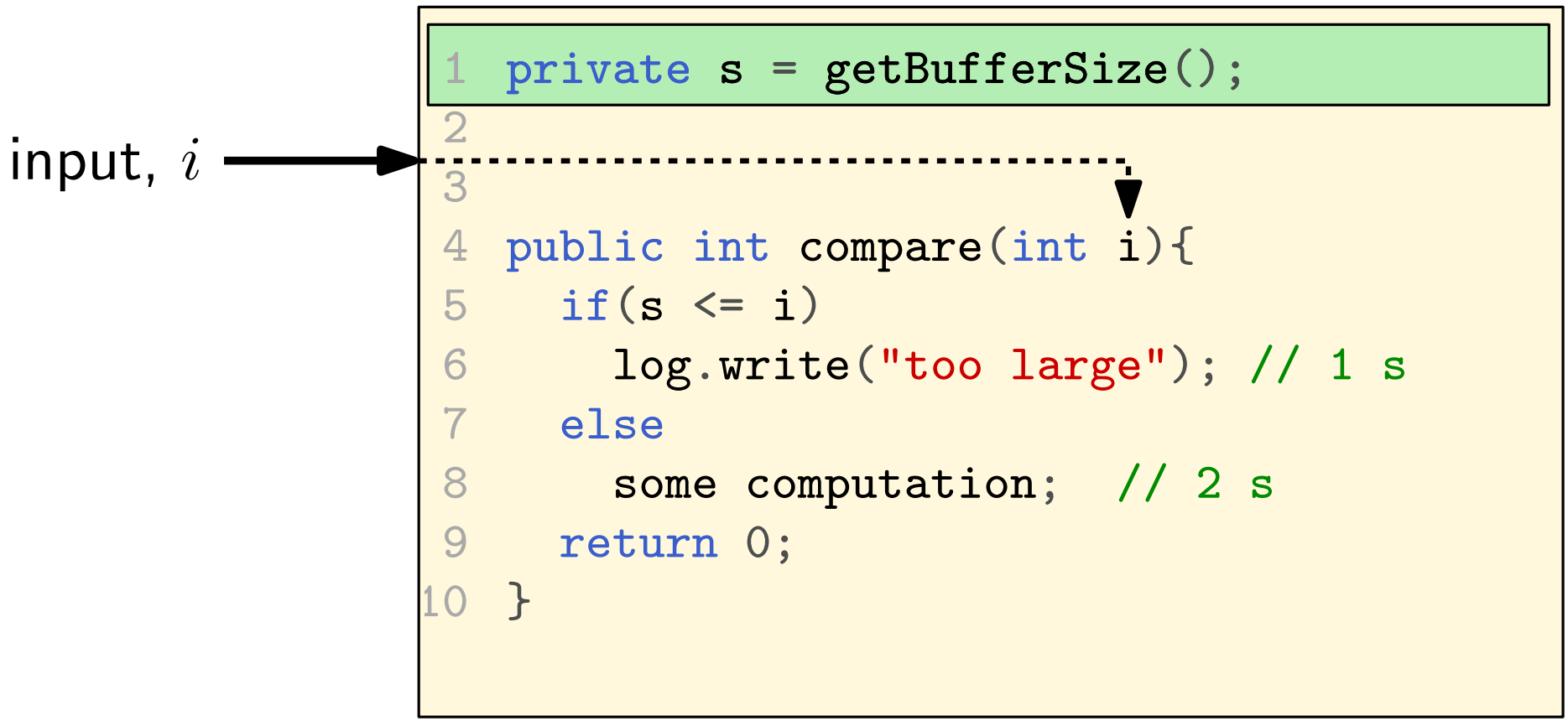

Secret Data
Program

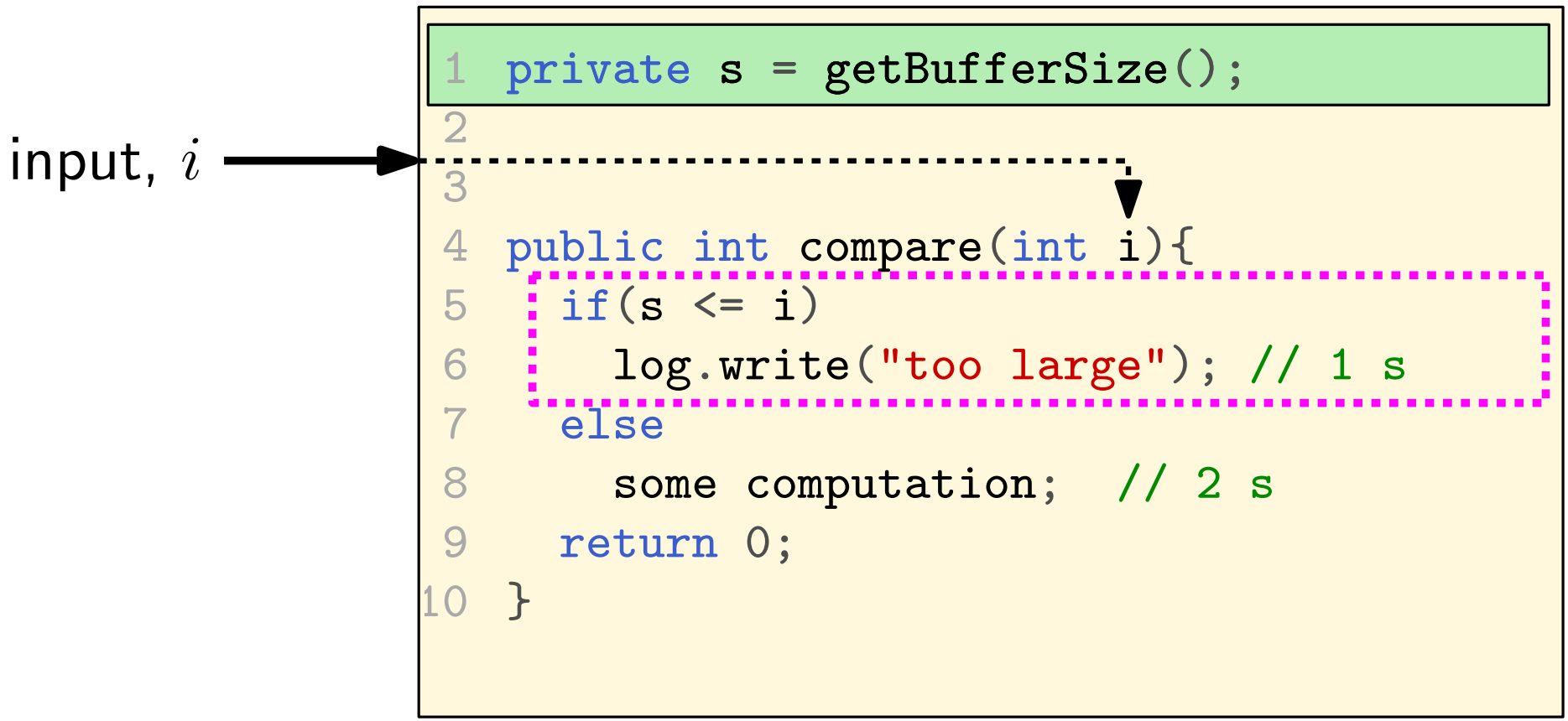
```
1  private s = getBufferSize();
```

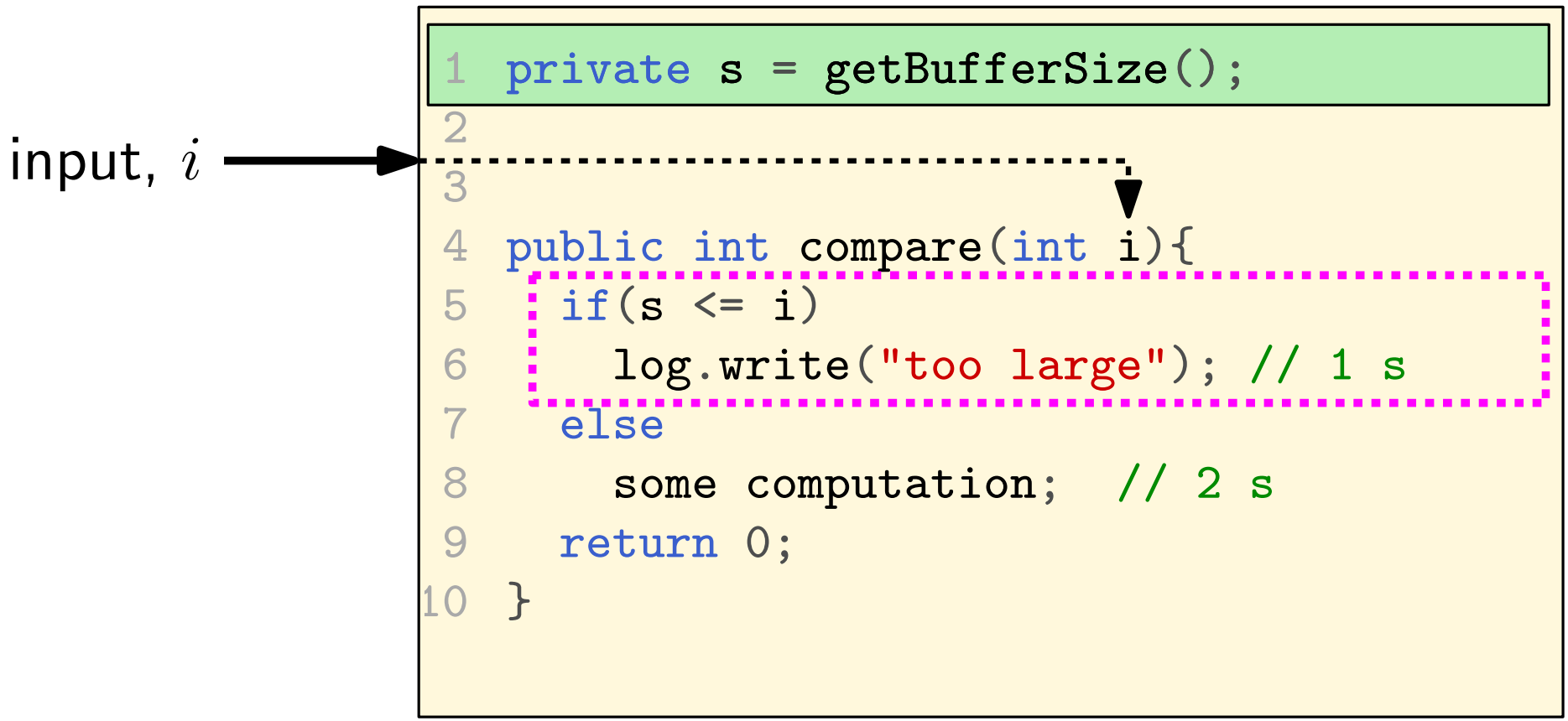
Program

```
1  private s = getBufferSize();
2
3
4  public int compare(int i){
5    if(s <= i)
6      log.write("too large"); // 1 s
7    else
8      some computation;  // 2 s
9    return 0;
10 }
```

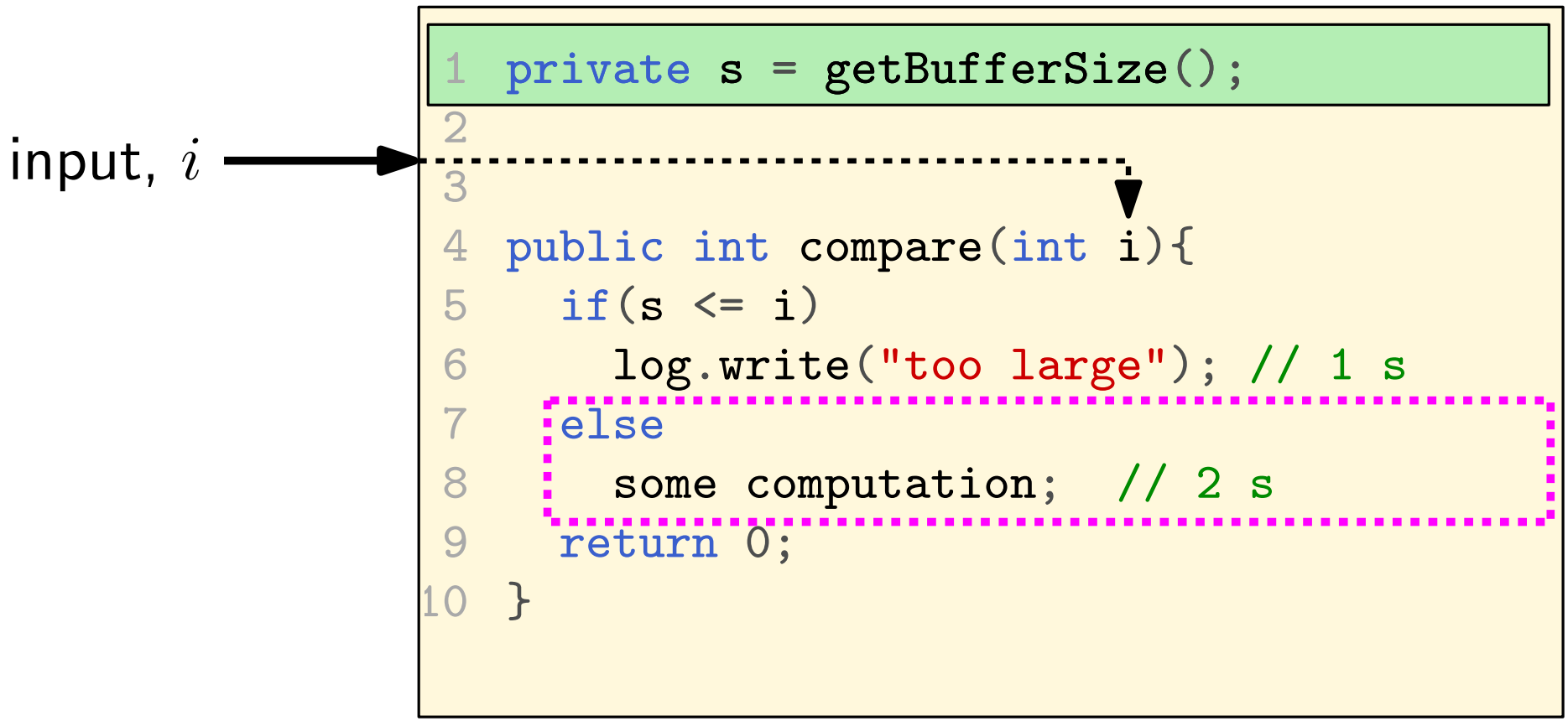

input, i
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }


input, i
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }

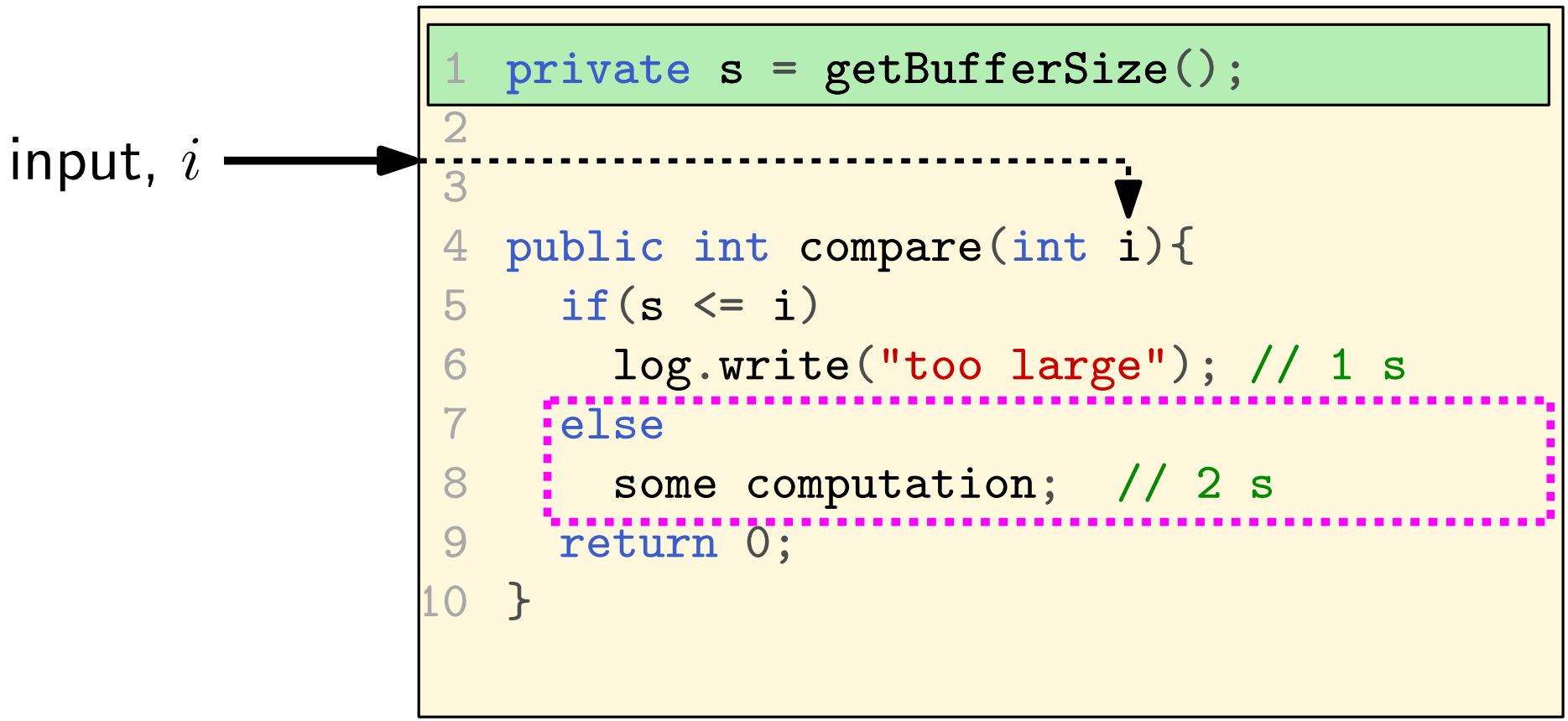

input, i
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }



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



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



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

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

input, i

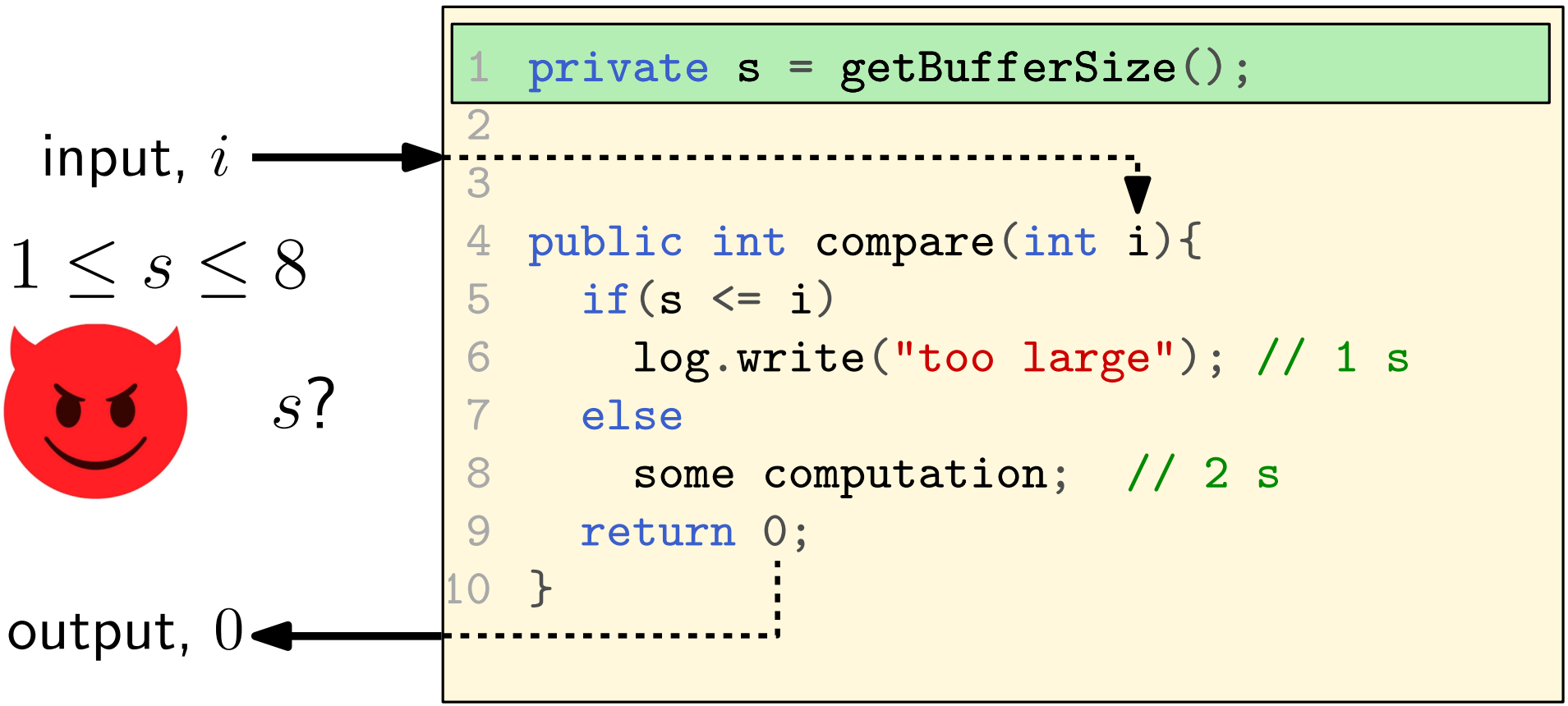
$1 \leq s \leq 8$

s?

```
1  private s = getBufferSize();
2
3
4  public int compare(int i){
5    if(s <= i)
6      log.write("too large"); // 1 s
7    else
8      some computation;  // 2 s
9    return 0;
10 }
```

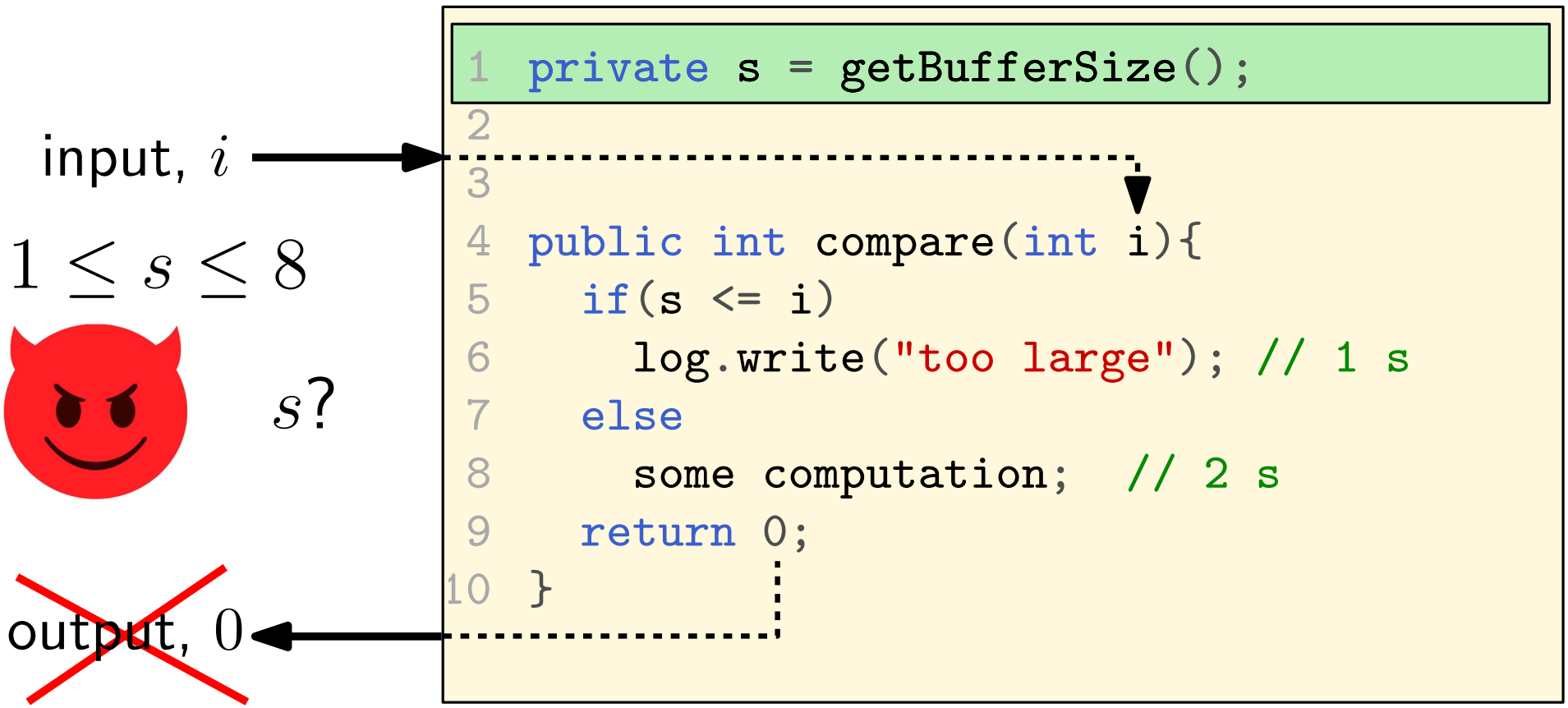
$$s \leq i \Rightarrow o = 1$$

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



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

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



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

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

input, i

$1 \leq s \leq 8$

s?

```
1  private s = getBufferSize();
2
3
4  public int compare(int i){
5    if(s <= i)
6      log.write("too large"); // 1 s
7    else
8      some computation;  // 2 s
9    return 0;
10 }
```

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

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

input, i

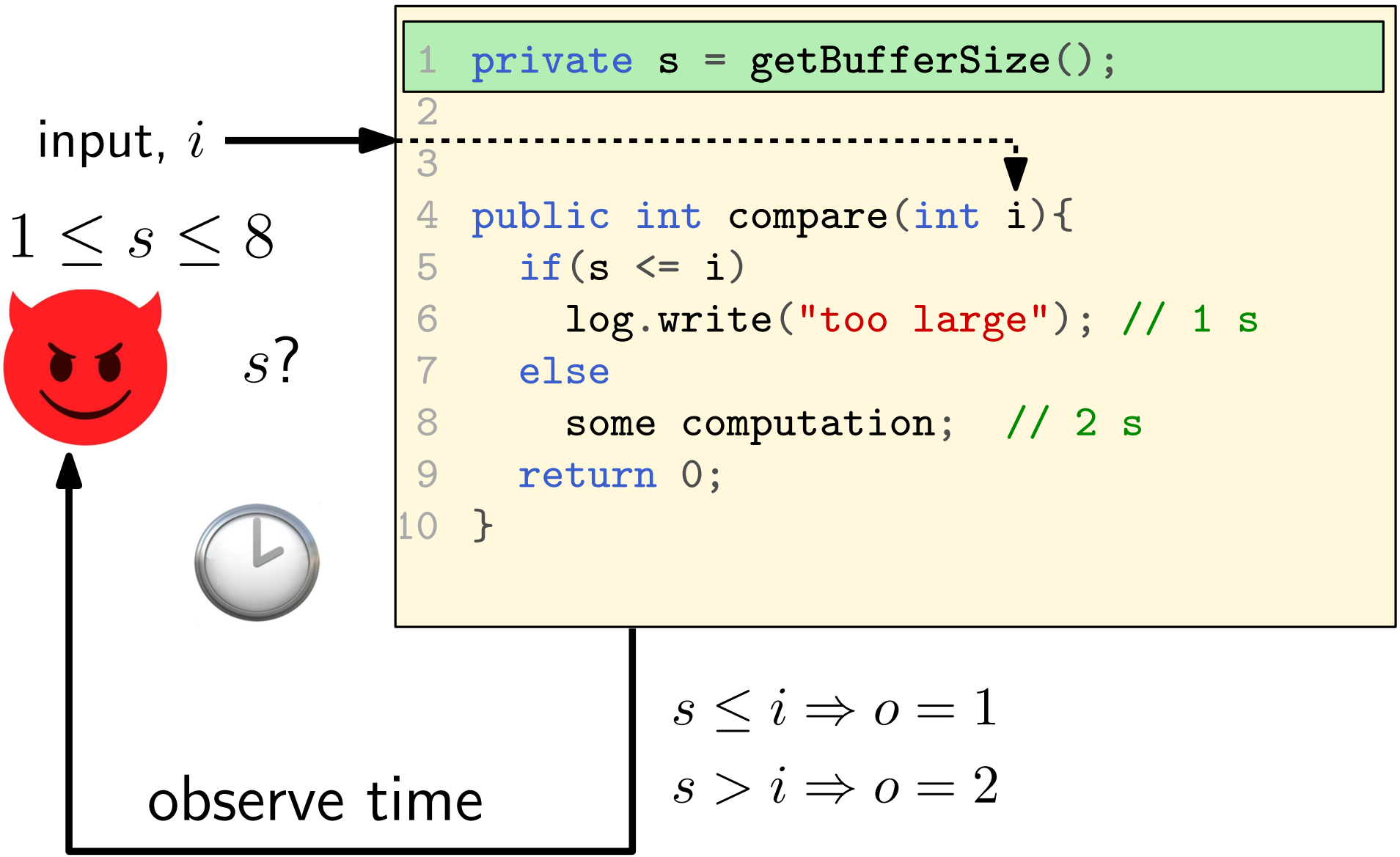
$1 \leq s \leq 8$

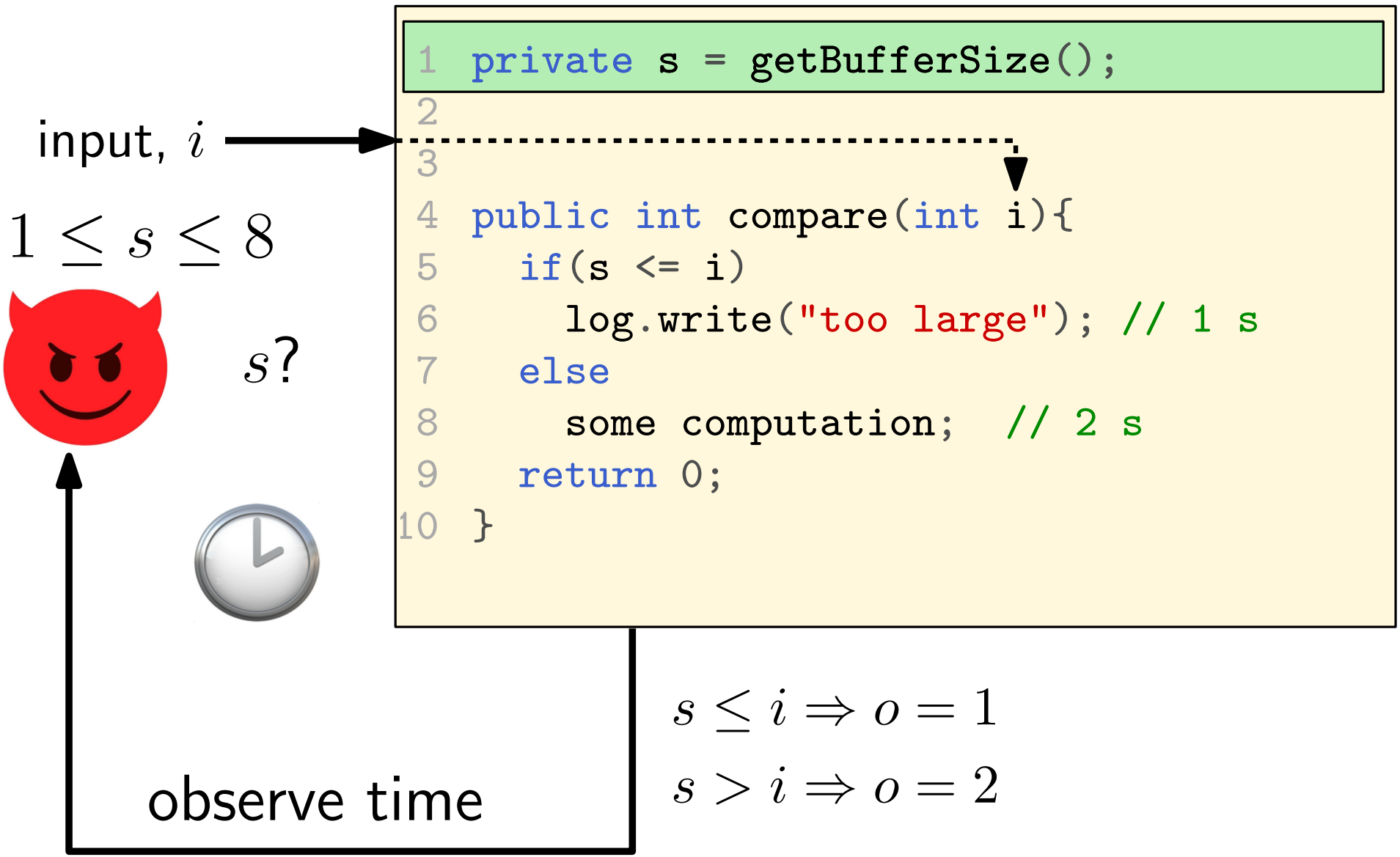
s?

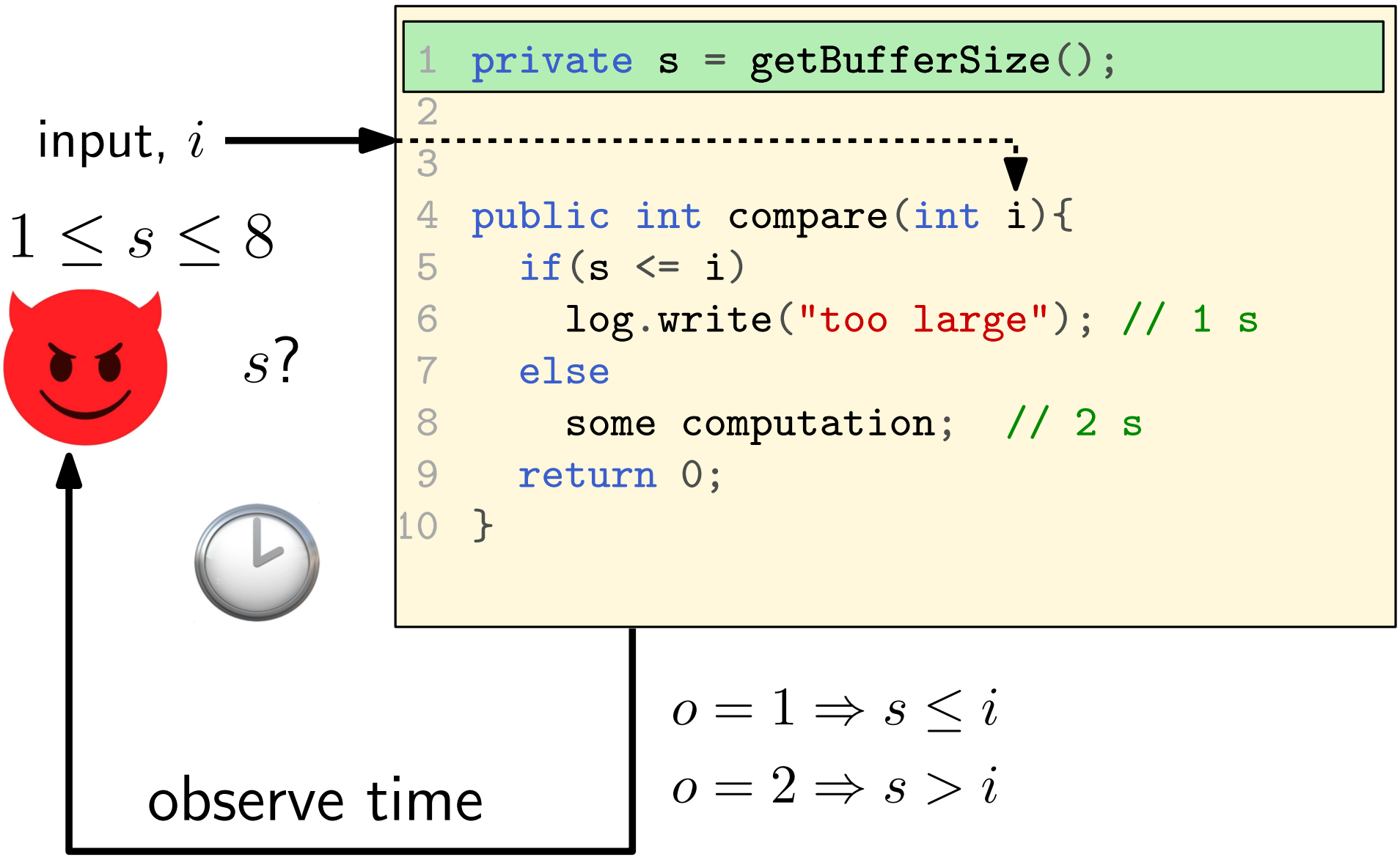
```
1  private s = getBufferSize();
2
3
4  public int compare(int i){
5    if(s <= i)
6      log.write("too large"); // 1 s
7    else
8      some computation;  // 2 s
9    return 0;
10 }
```

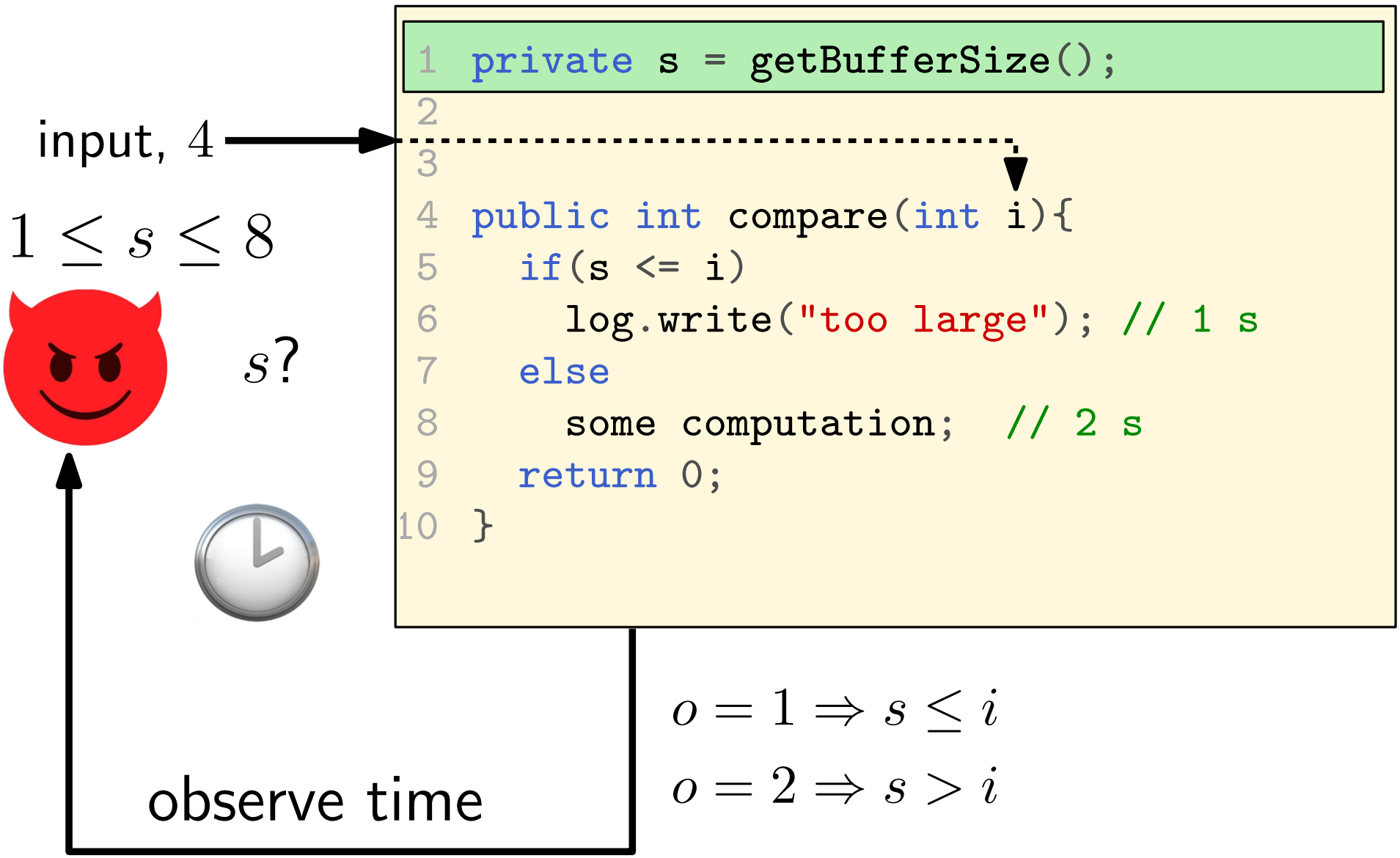
$$s \leq i \Rightarrow o = 1$$

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

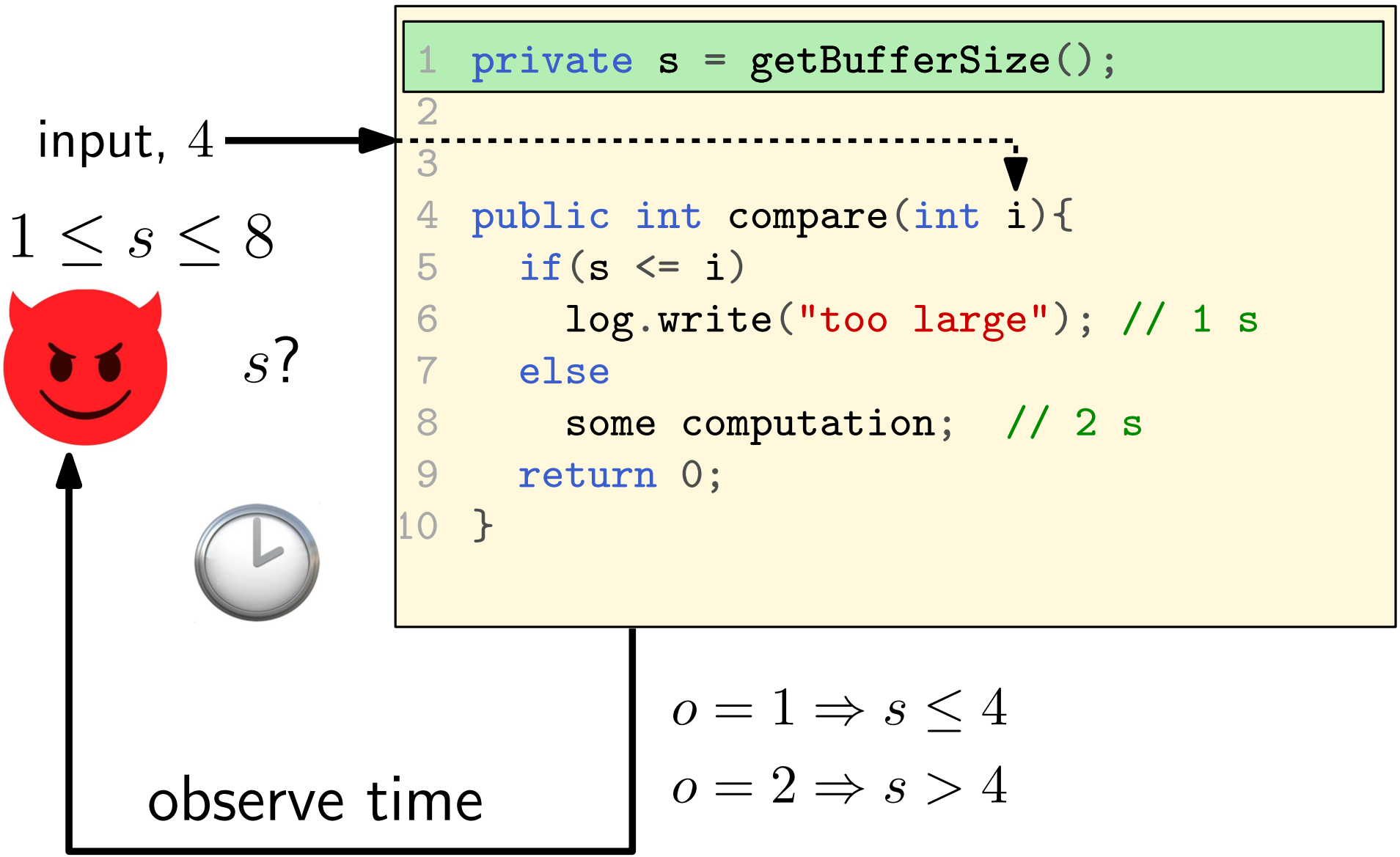

input, i
$1 \leq s \leq 8$
s?
observe time
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }
$s \leq i \Rightarrow o = 1$
$s > i \Rightarrow o = 2$

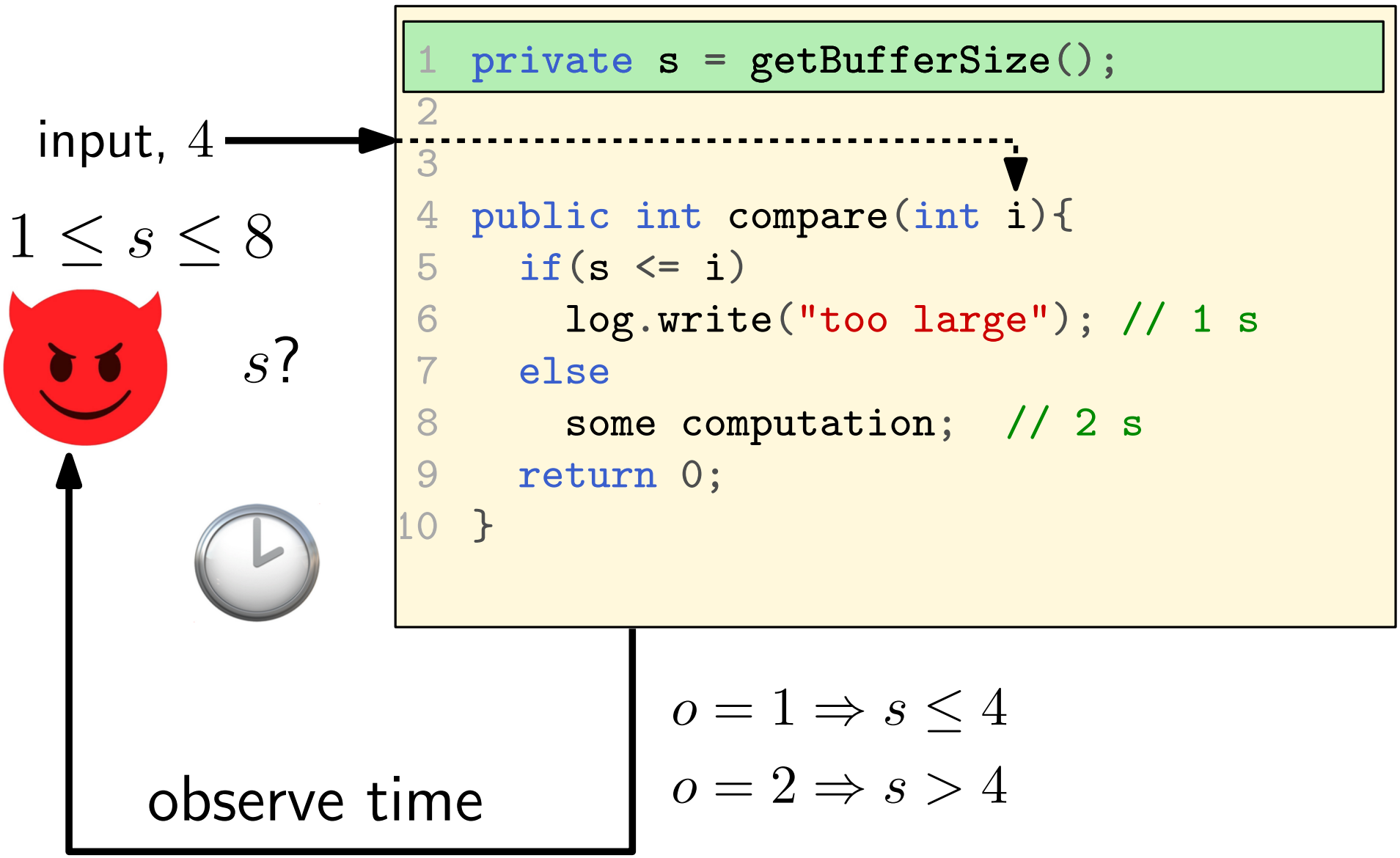

input, i
$1 \leq s \leq 8$
s?
observe time
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }
$s \leq i \Rightarrow o = 1$
$s > i \Rightarrow o = 2$


input, i
$1 \leq s \leq 8$
s?
observe time
$o = 1 \Rightarrow s \leq i$
$o = 2 \Rightarrow s > i$
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }




```
1  private s = getBufferSize();
2
3
4  public int compare(int i){
5    if(s <= i)
6      log.write("too large"); // 1 s
7    else
8      some computation;  // 2 s
9    return 0;
10 }
```


input, 4
$1 \leq s \leq 8$
s?
observe time
$o = 1 \Rightarrow s \leq 4$
$o = 2 \Rightarrow s > 4$
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }



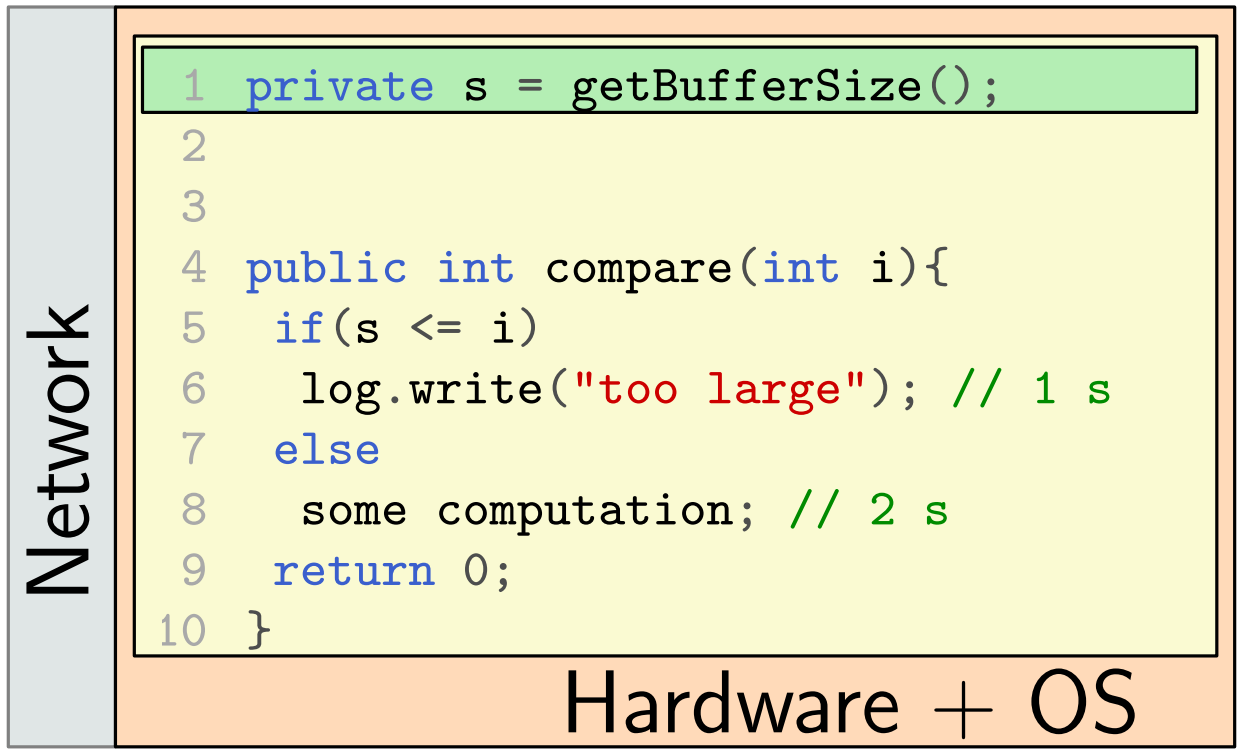

```
1  private s = getBufferSize();
2
3
4  public int compare(int i){
5    if(s <= i)
6      log.write("too large"); // 1 s
7    else
8      some computation;  // 2 s
9    return 0;
10 }
```

Attacker can binary search on s using i and o.

```
1  private s = getBufferSize();
2
3
4  public int compare(int i){
5   if(s <= i)
6    log.write("too large"); // 1 s
7   else
8    some computation; // 2 s
9   return 0;
10 }
```

```
1  private s = getBufferSize();
2
3
4  public int compare(int i){
5   if(s <= i)
6    log.write("too large"); // 1 s
7   else
8    some computation; // 2 s
9   return 0;
10 }
```

Hardware + OS

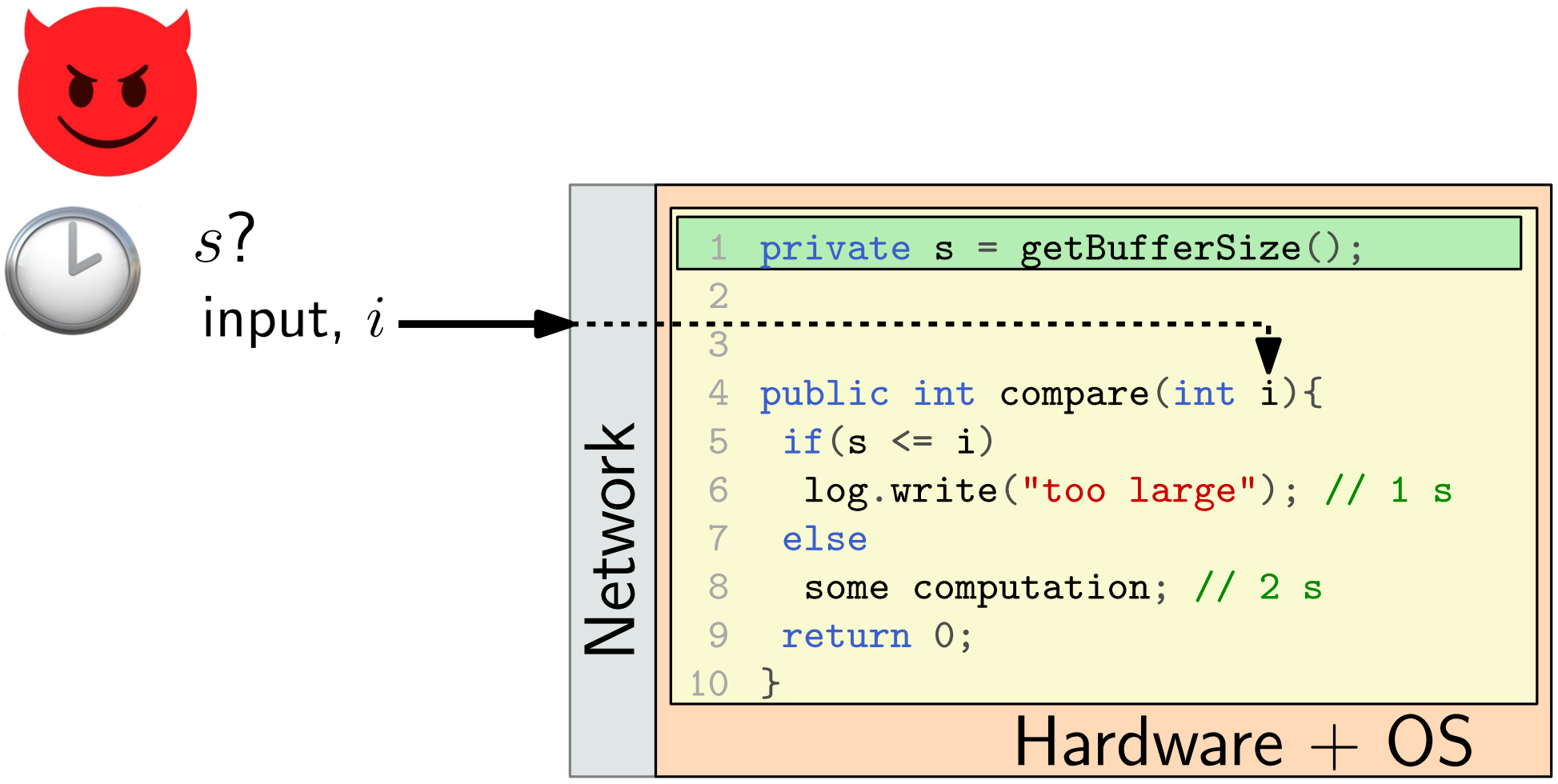

Network
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }
Hardware + OS

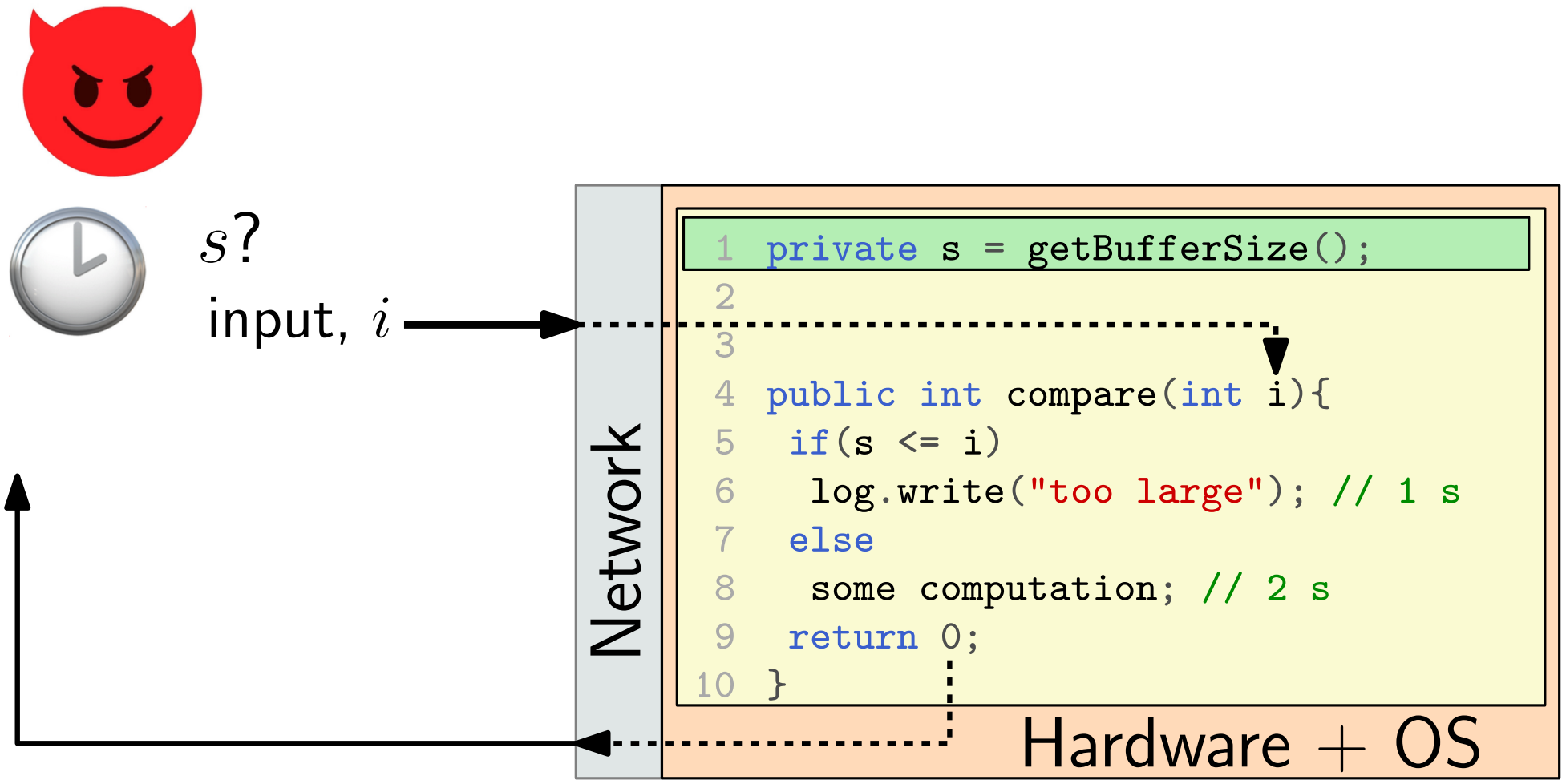
s?

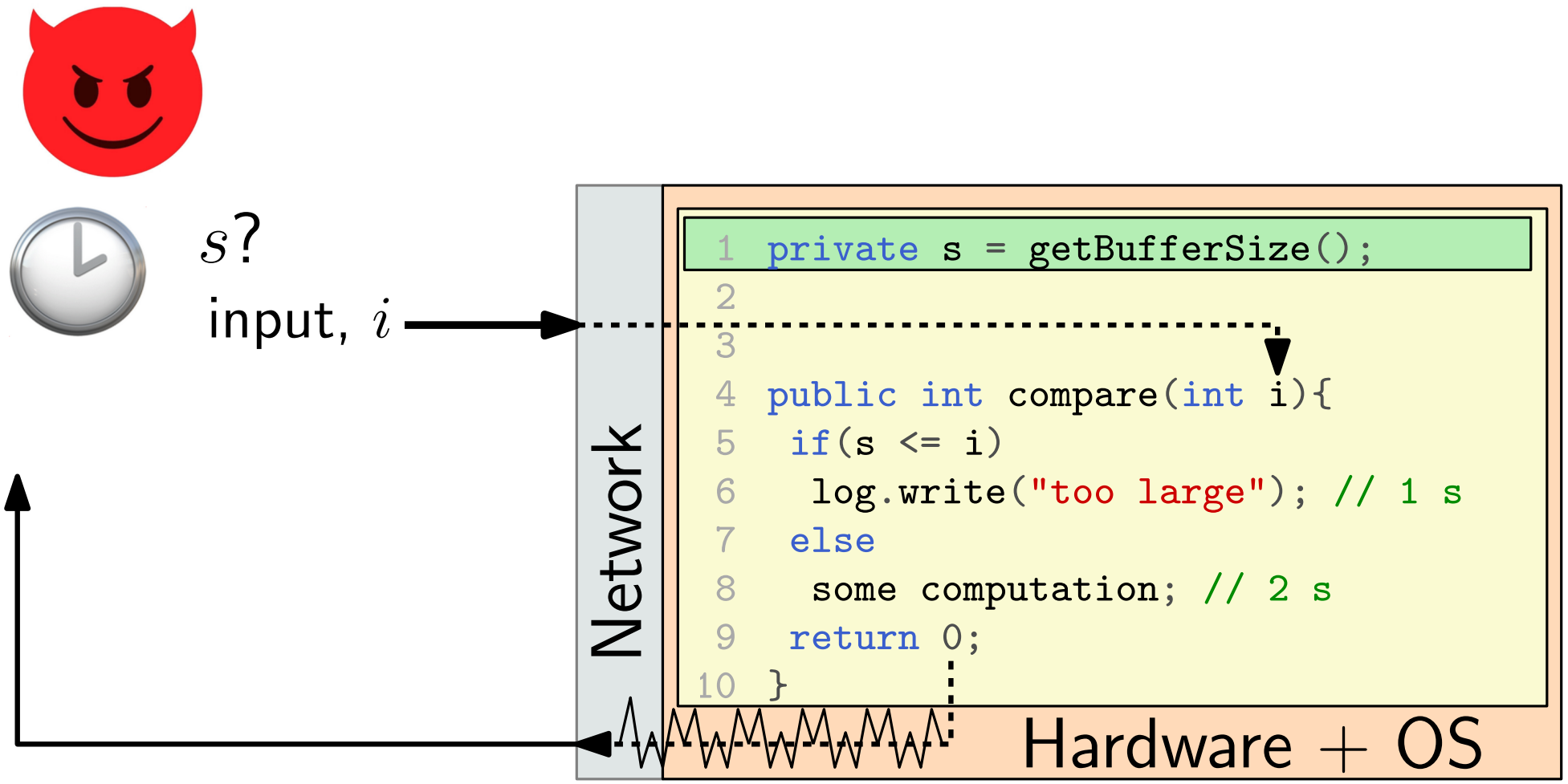
Network

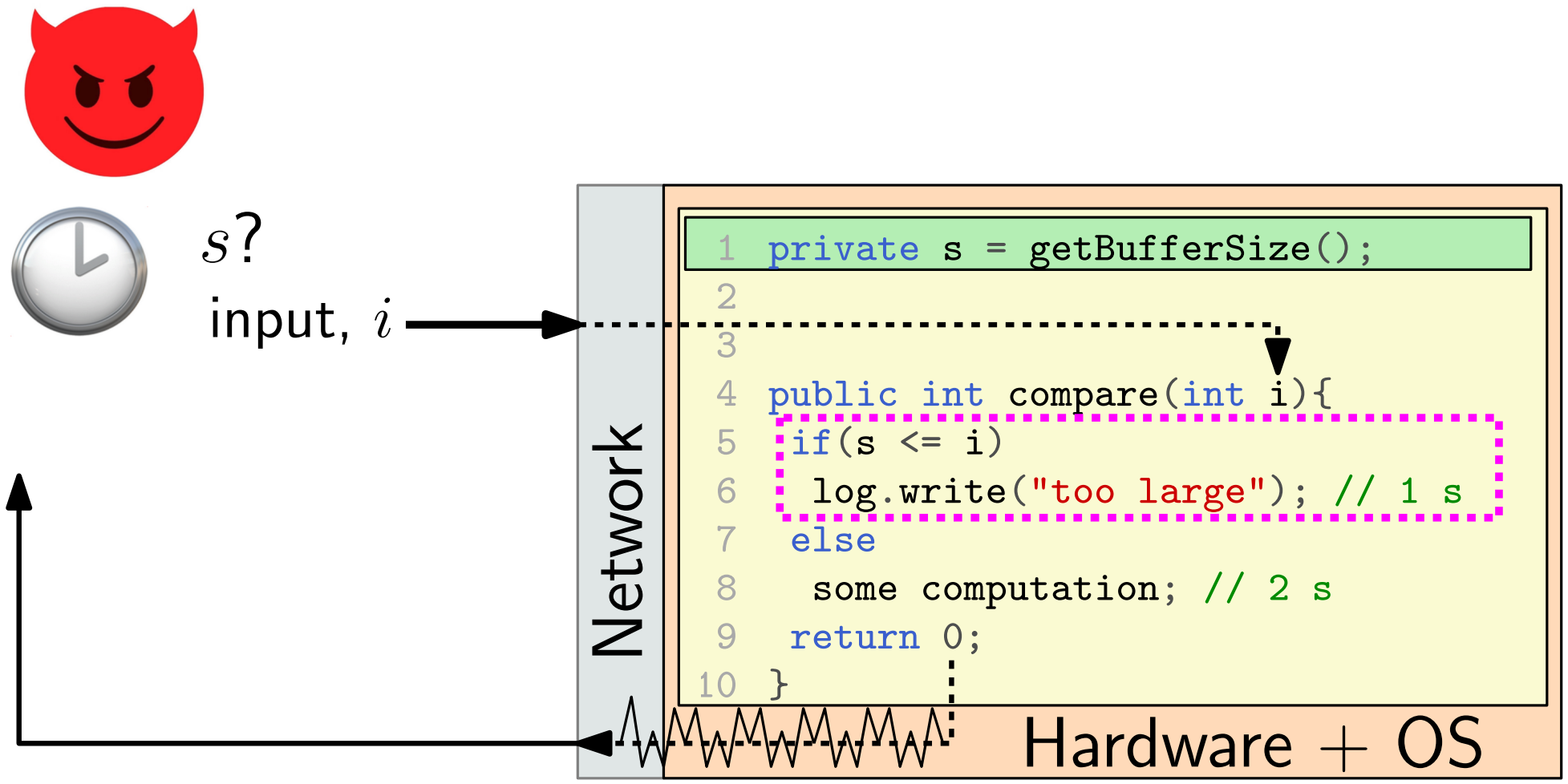
```
1  private s = getBufferSize();
2
3
4  public int compare(int i){
5   if(s <= i)
6    log.write("too large"); // 1 s
7   else
8    some computation; // 2 s
9   return 0;
10 }
```

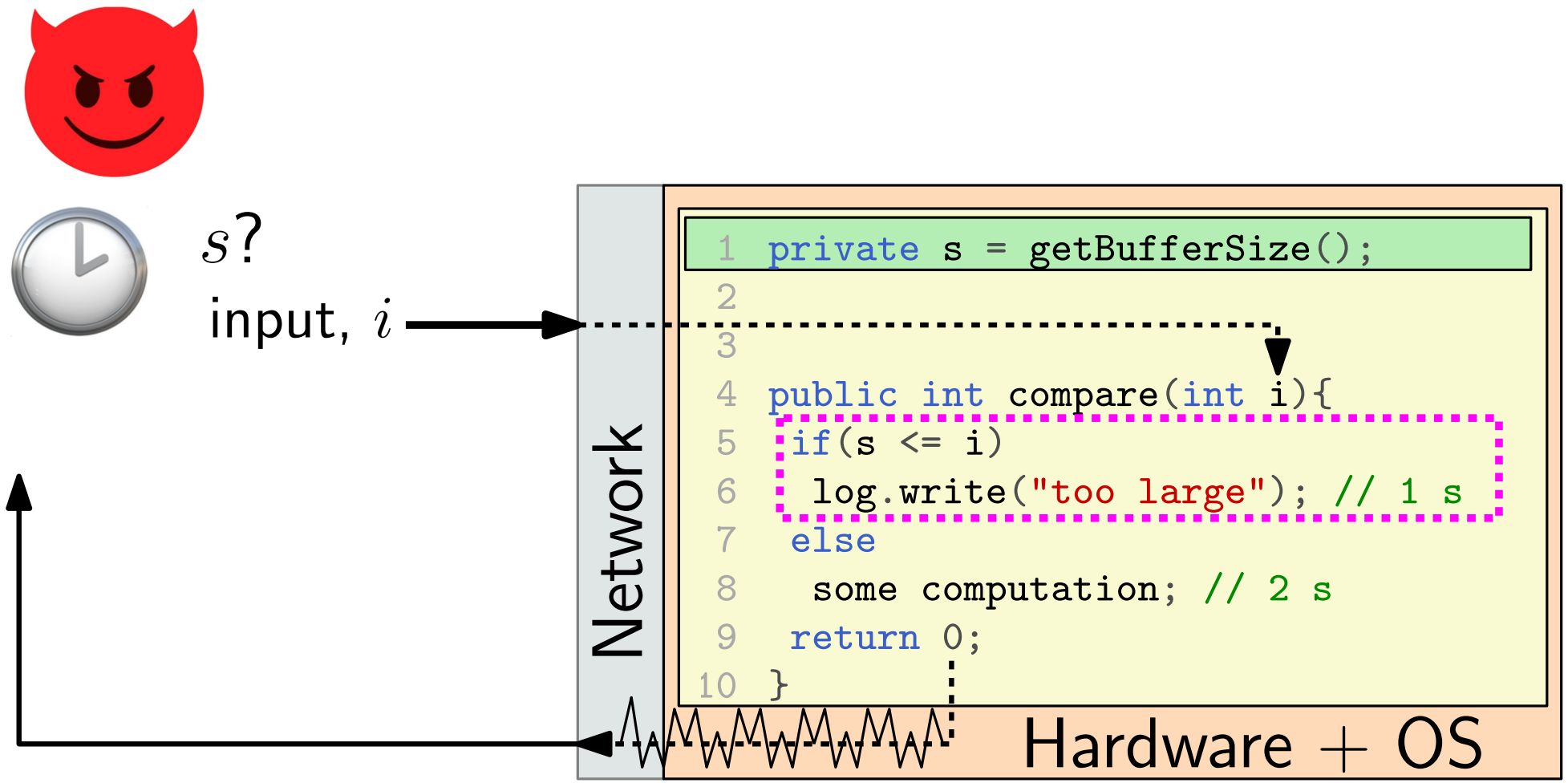
Hardware + OS


s?
input, i
Network
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }
Hardware + OS

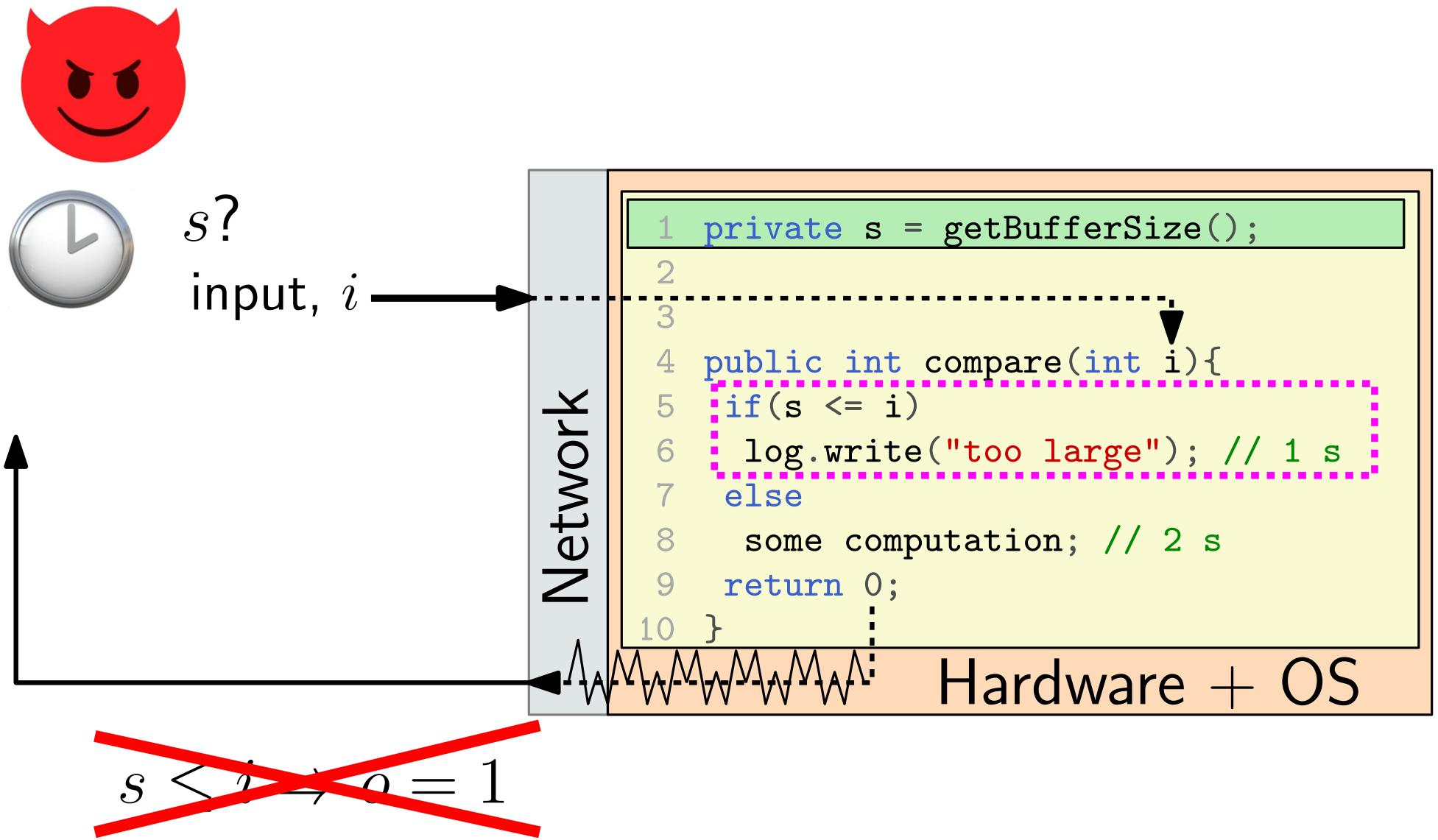

s?
input, i
Network
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }
Hardware + OS

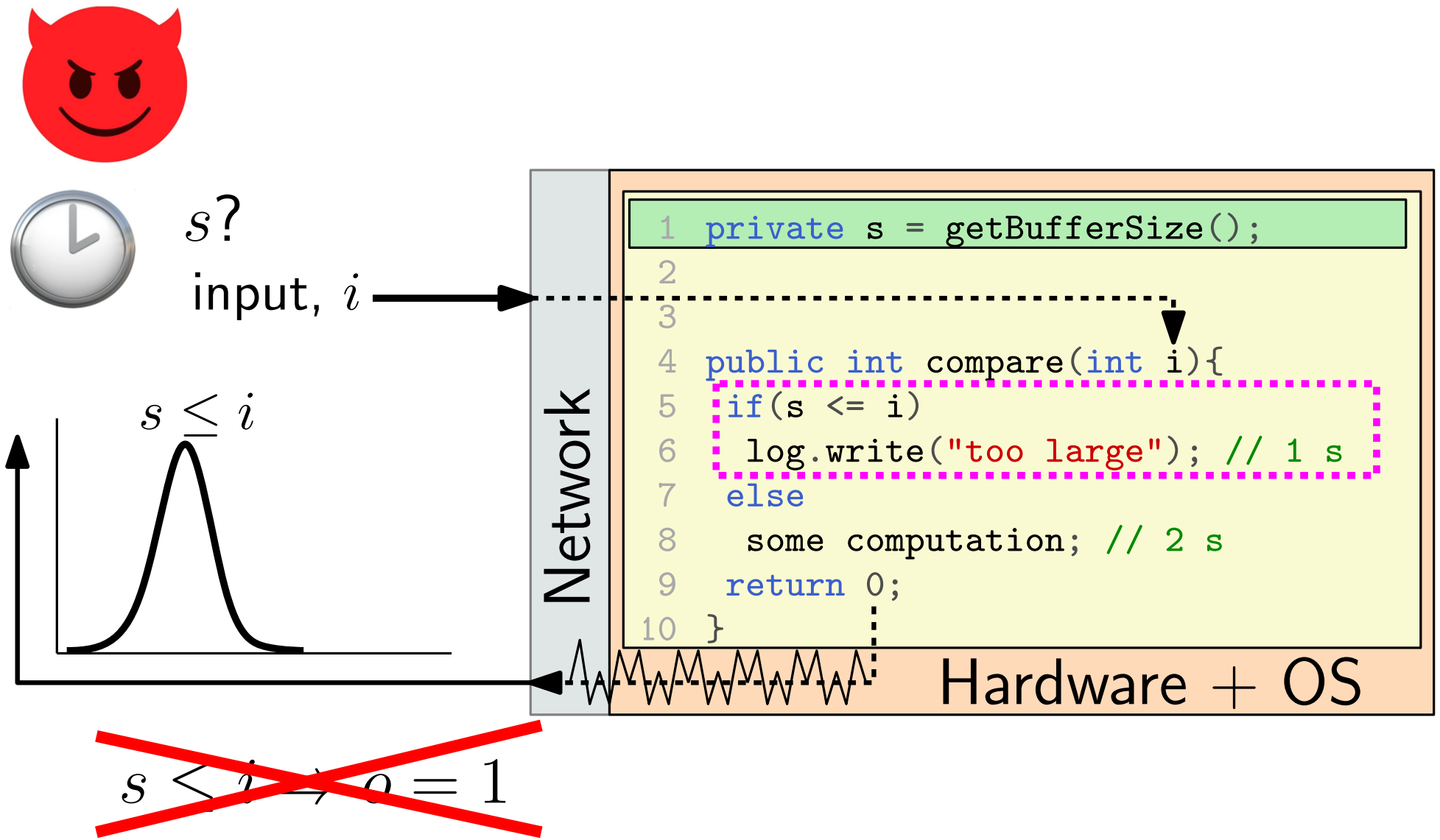

s?
input, i
Network
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }
Hardware + OS

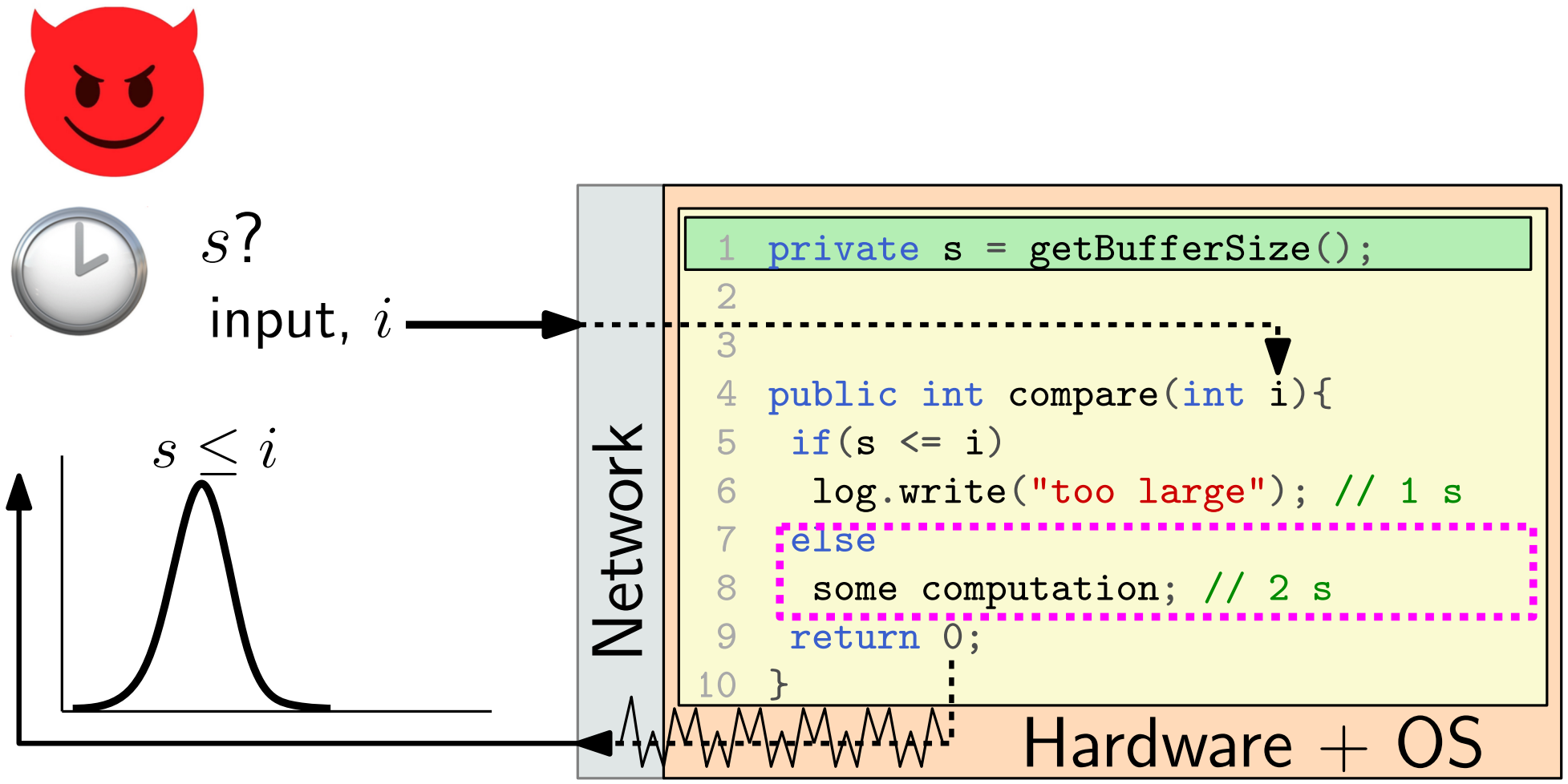

s?
input, i
Network
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }
Hardware + OS

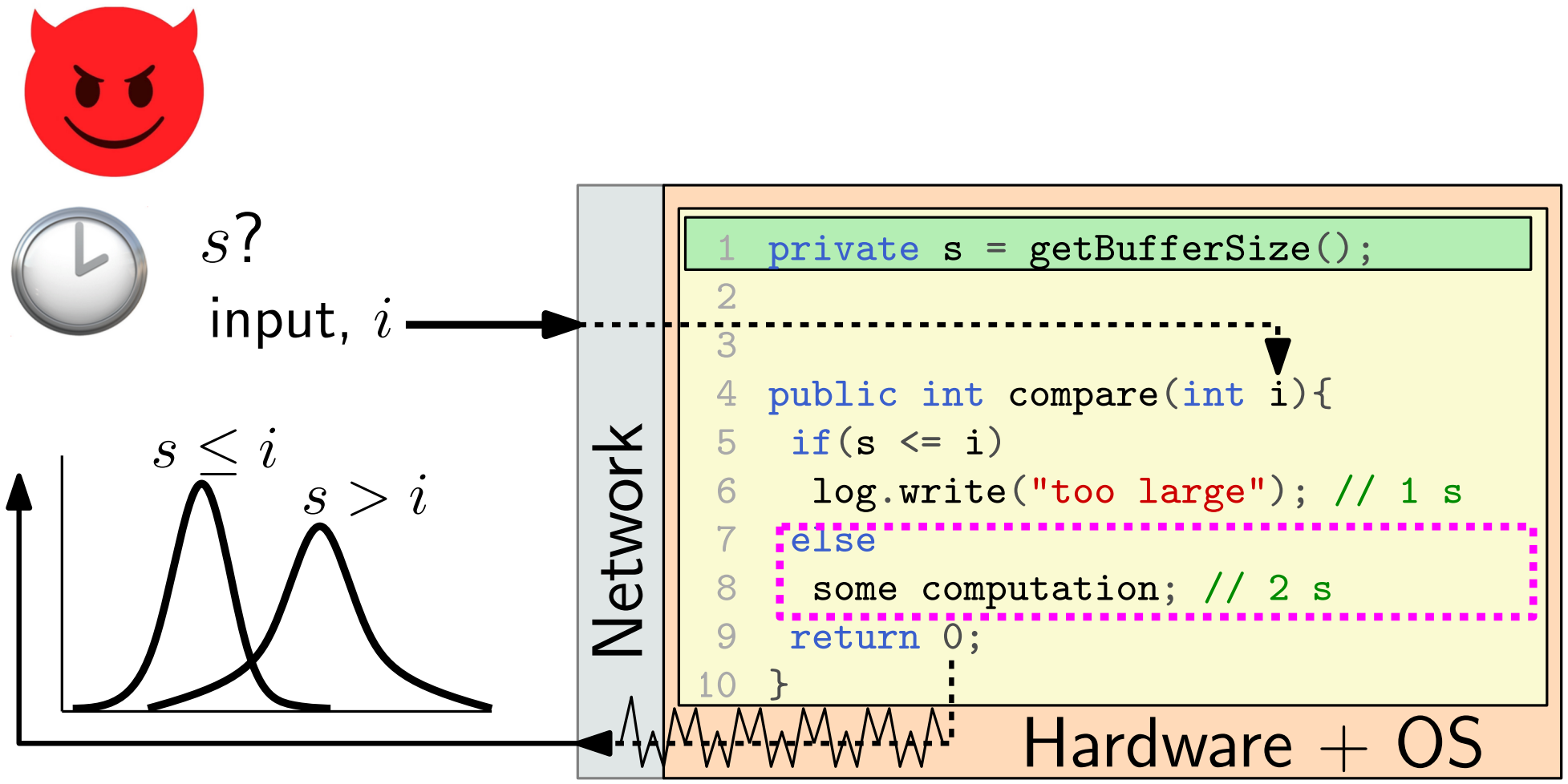


$$s \leq i \Rightarrow o = 1$$


s?
input, i
Network
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }
Hardware + OS
$s \leq i \rightarrow o = 1$


s?
input, i
$s \leq i$
Network
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }
Hardware + OS
$s \leq i \rightarrow o = 1$

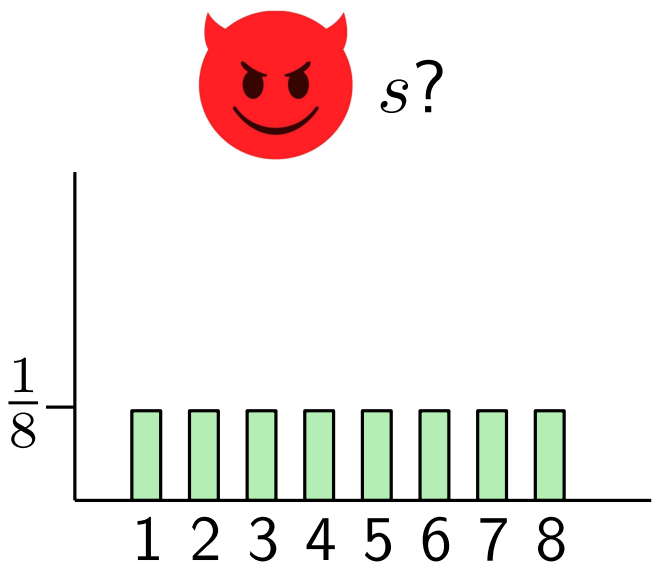

s?
input, i
$s \leq i$
Network
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }
Hardware + OS


s?
input, i
$s \leq i$
$s > i$
Network
1 private s = getBufferSize();
2
3
4 public int compare(int i){
5 if(s <= i)
6 log.write("too large"); // 1 s
7 else
8 some computation; // 2 s
9 return 0;
10 }
Hardware + OS

Attacker Belief?

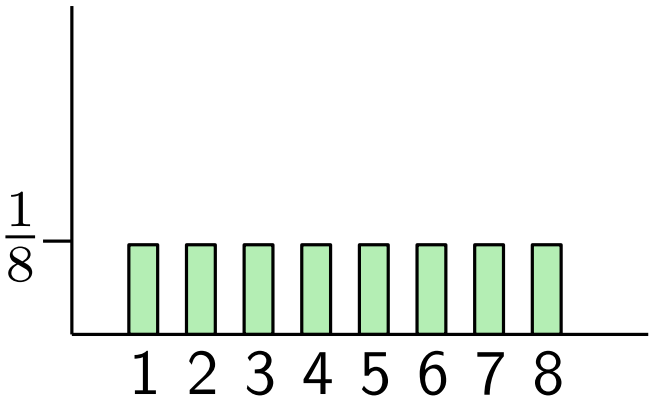
 s?

Attacker Belief?



Attacker Belief?

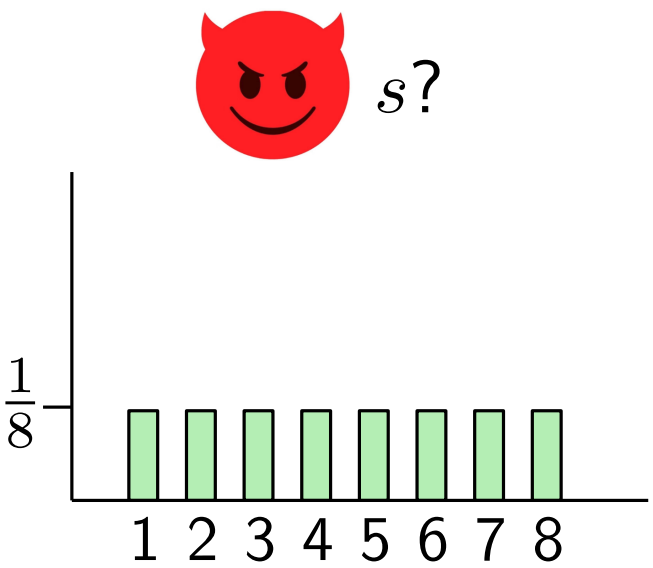
s?



Input Choice?

i^*

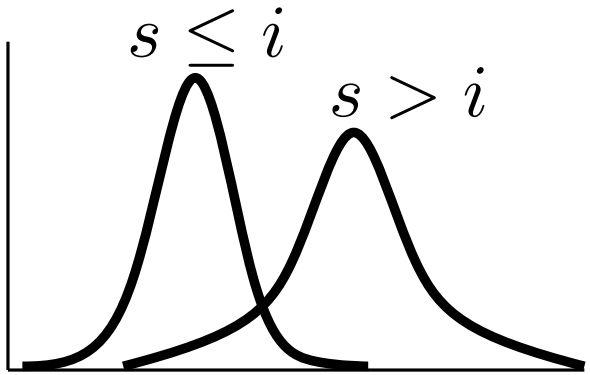
Attacker Belief?



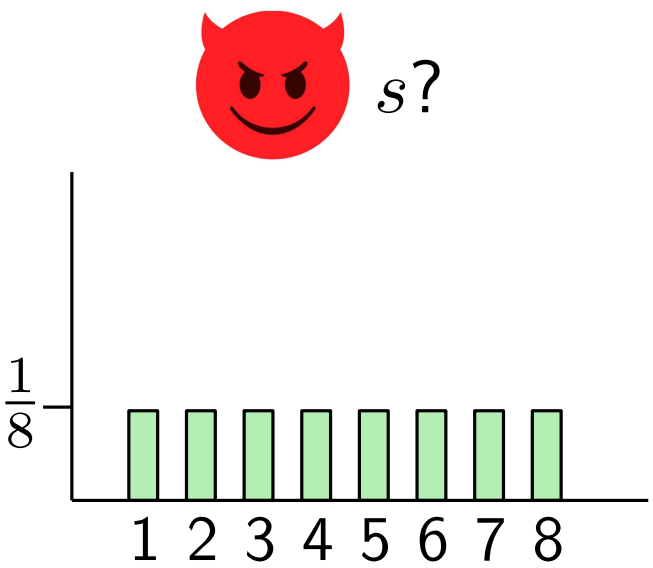
Input Choice?

i^*

Observation?



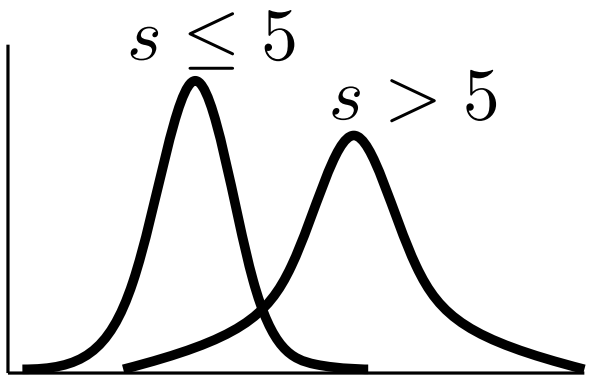
Attacker Belief?



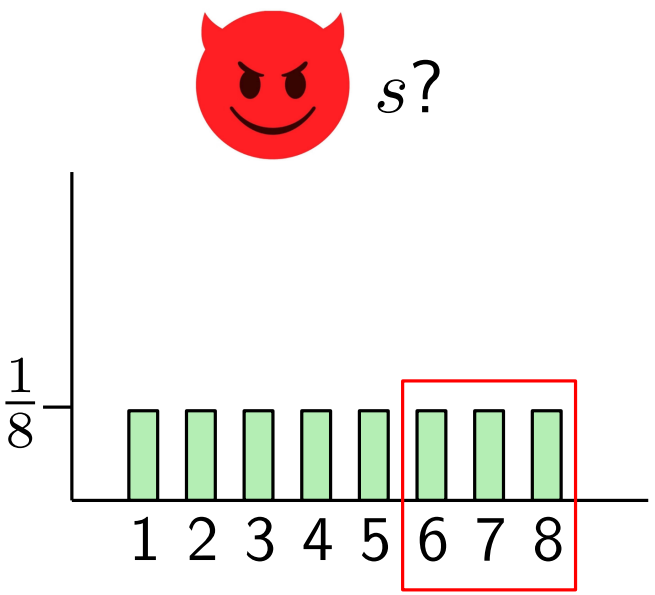
Input Choice?

$$i^* = 5$$

Observation?



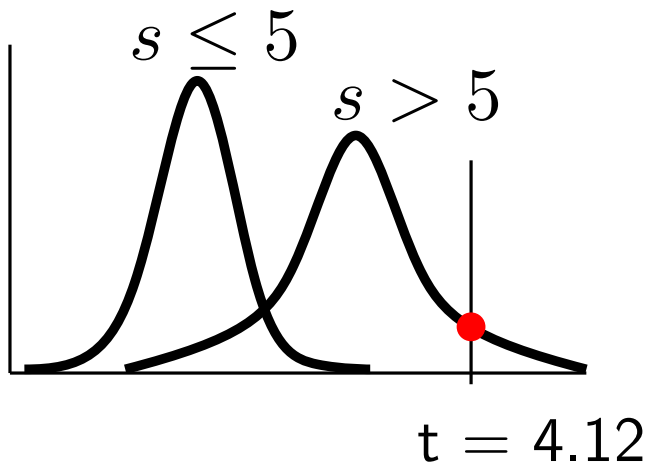
Attacker Belief?



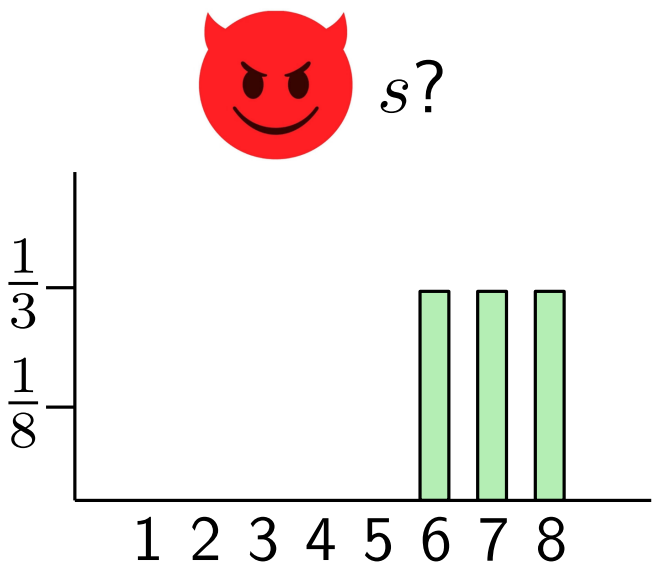
Input Choice?

$$i^* = 5$$

Observation?



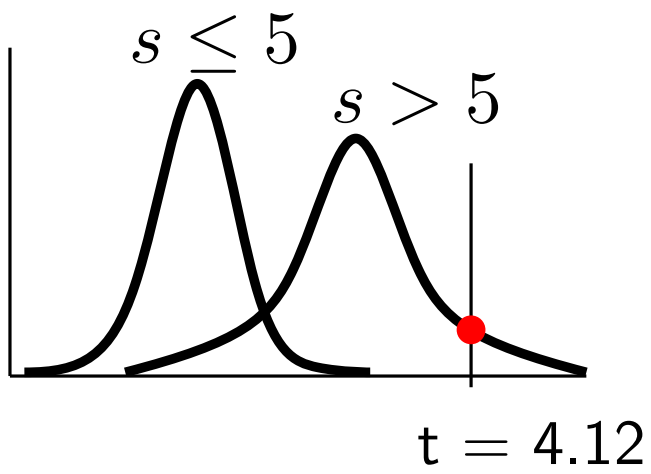
Attacker Belief?



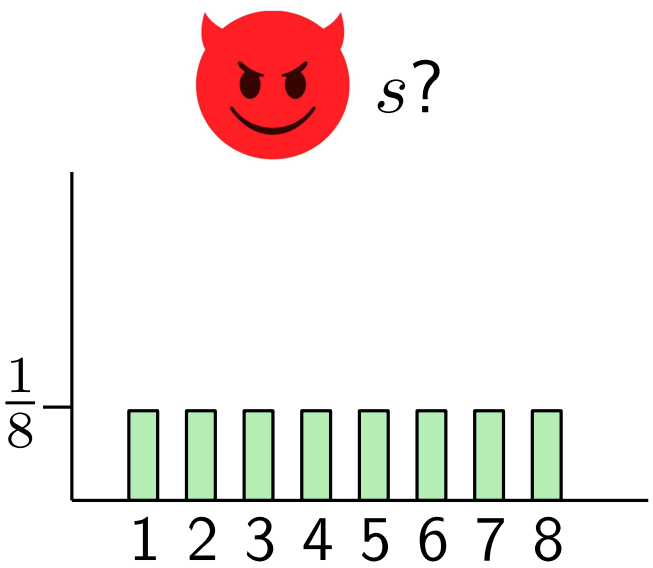
Input Choice?

$i^* = 5$

Observation?



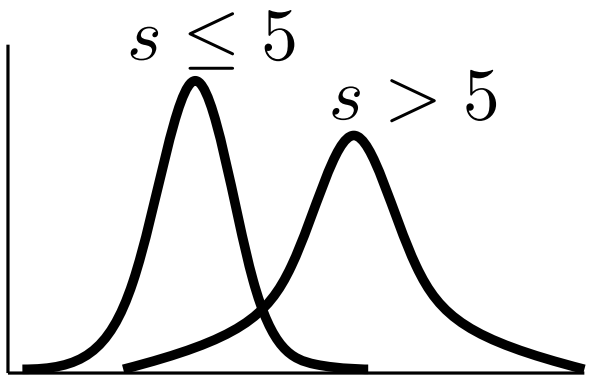
Attacker Belief?



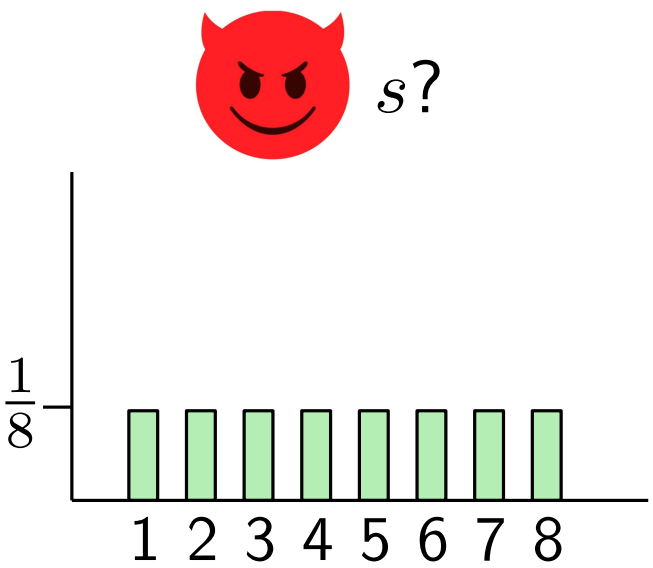
Input Choice?

$$i^* = 5$$

Observation?



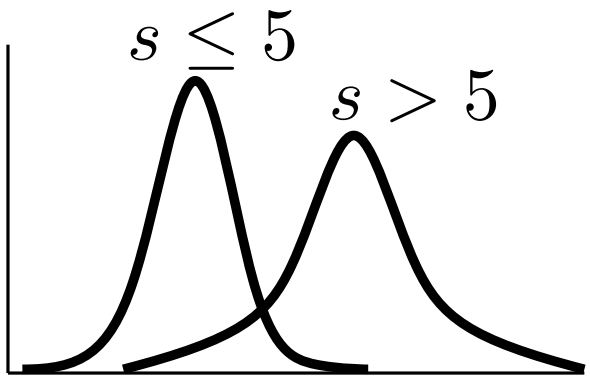
Attacker Belief?



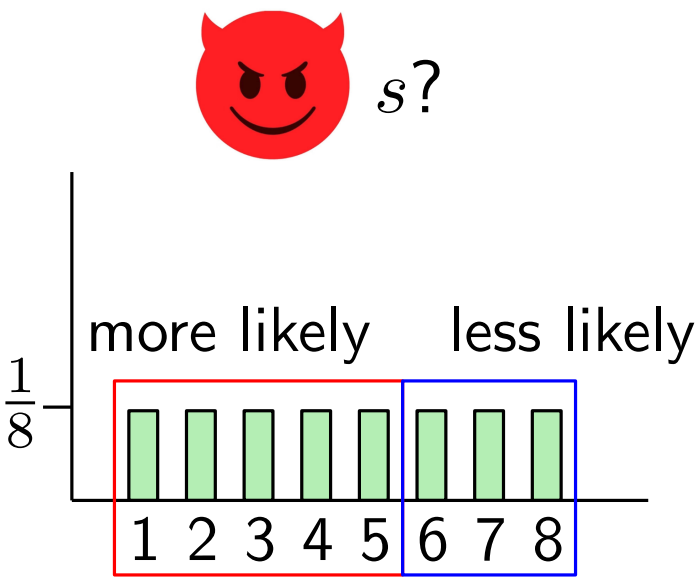
Input Choice?

$$i^* = 5$$

Observation?



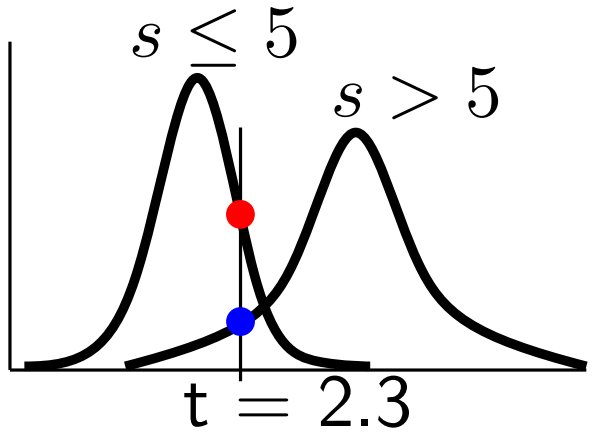
Attacker Belief?



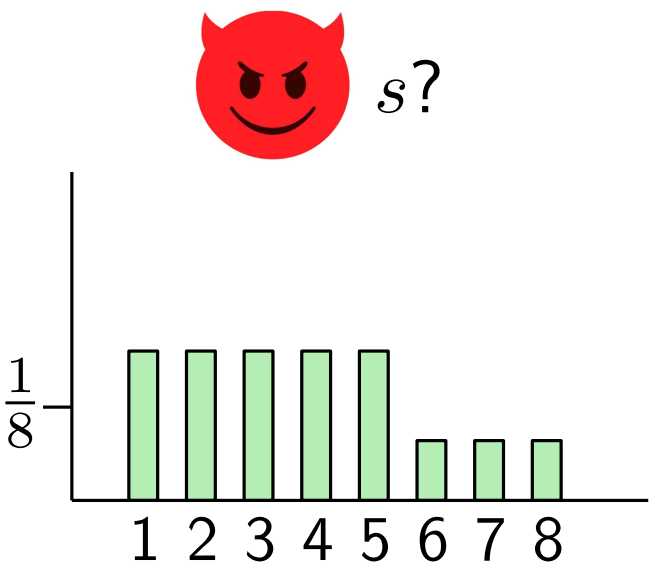
Input Choice?

$i^* = 5$

Observation?



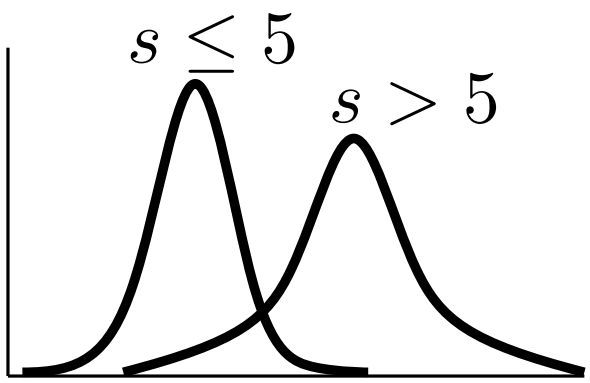
Attacker Belief?



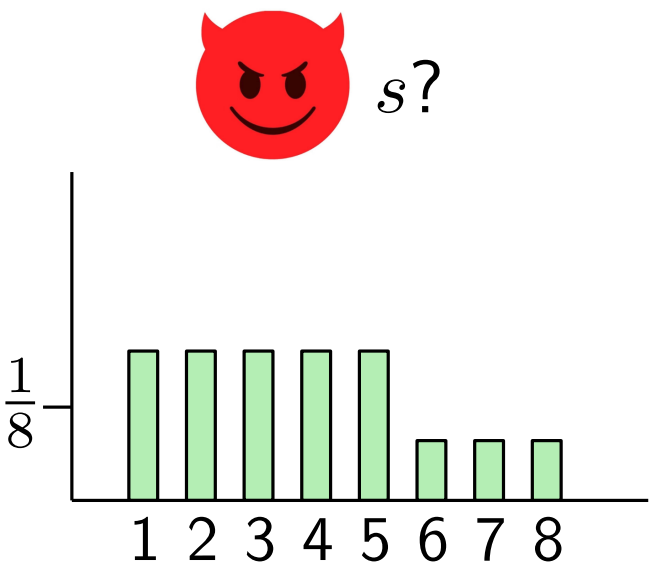
Input Choice?

$$i^* = 5$$

Observation?



Attacker Belief?

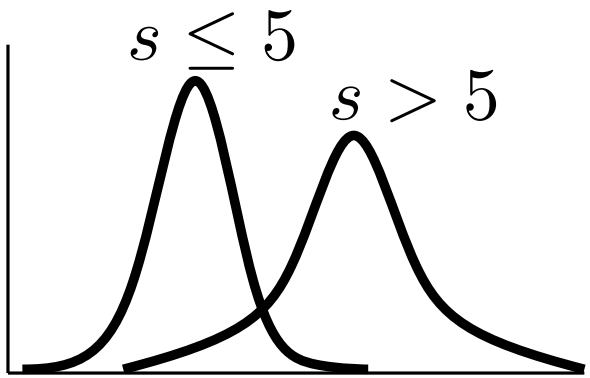


$p(s|o, i^*)$

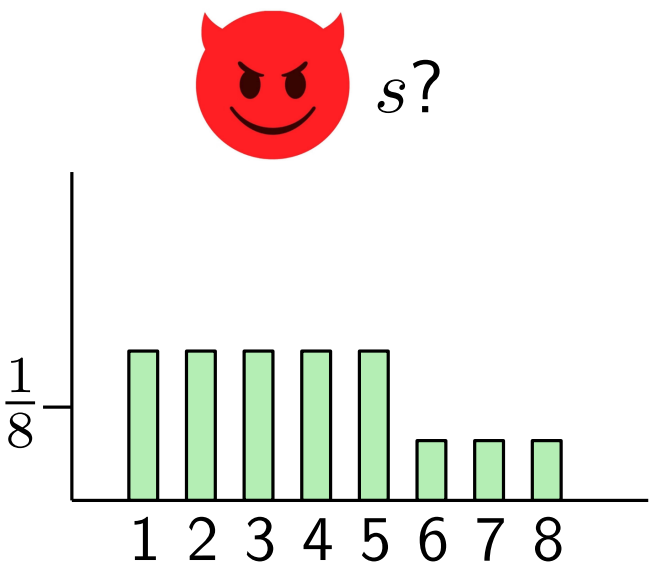
Input Choice?

$i^* = 5$

Observation?



Attacker Belief?

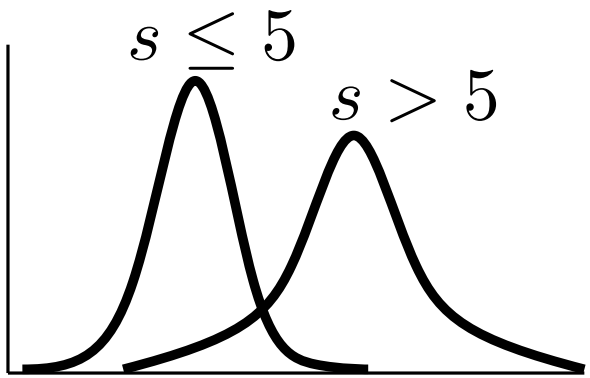


$p(s|o, i^*)$

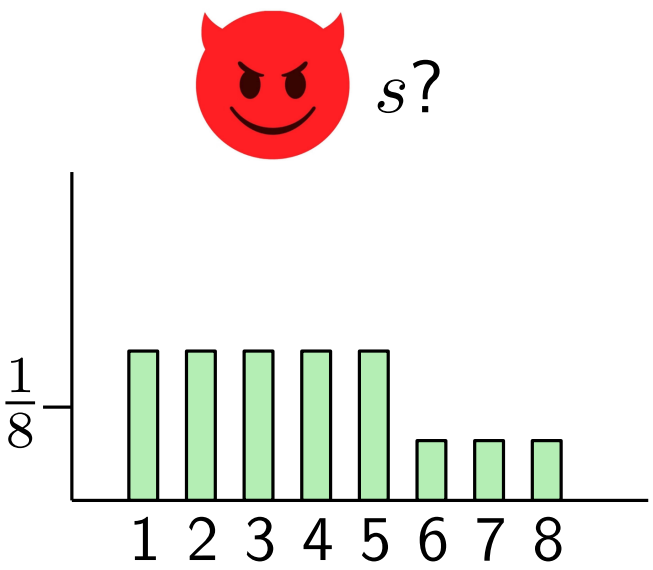
Input Choice?

$i^* = 5$

Observation?



Attacker Belief?

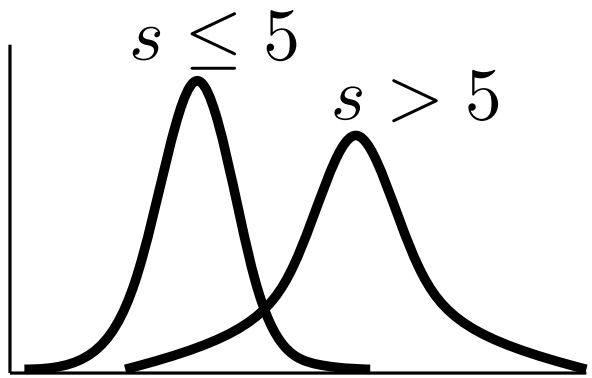


$p(s|o, i^*)$

Input Choice?

$i^* = 5$

Observation?



$p(o|s, i)$

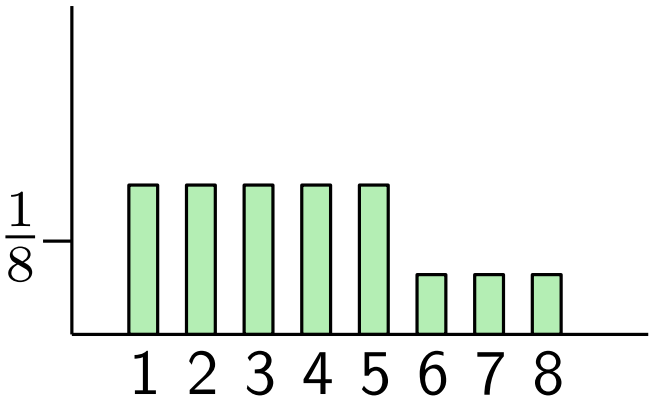
Attacker Belief?

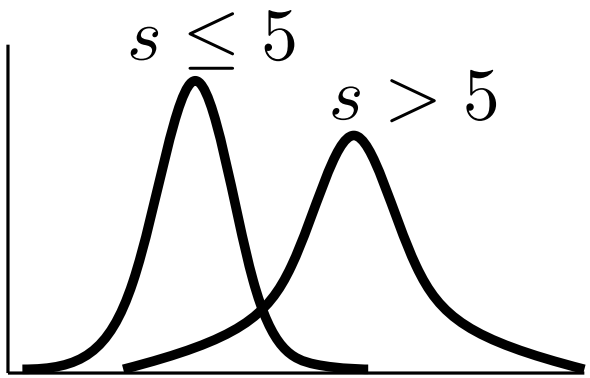
Input Choice?

Observation?

s?

$i^* = 5$





$p(s|o, i^*)$ ← $p(o|s, i)$

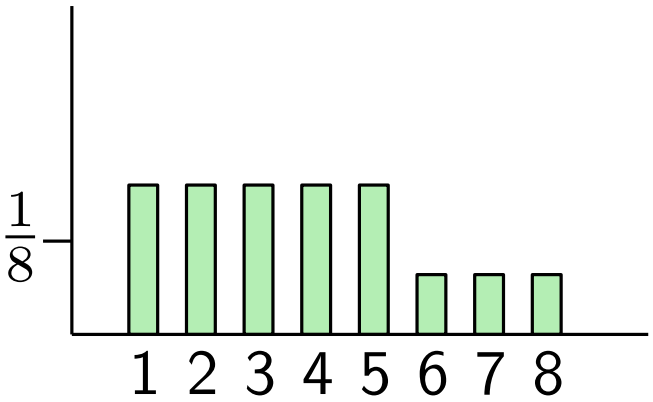
Attacker Belief?

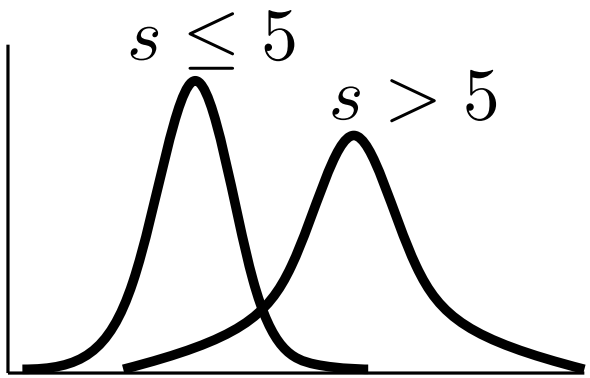
Input Choice?

Observation?

s?

$i^* = 5$





$p(o|s,i)$

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

$p(o|s,i)$

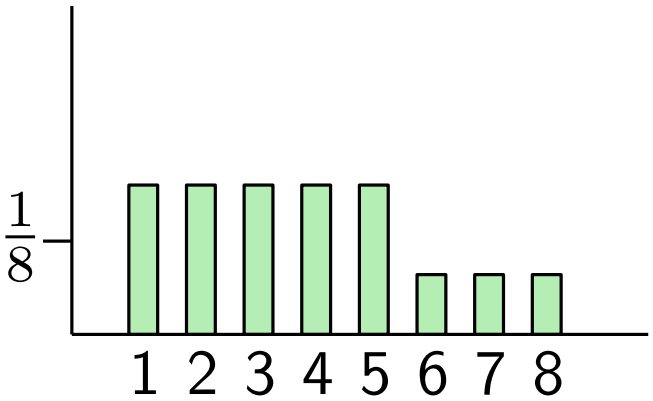
Attacker Belief?

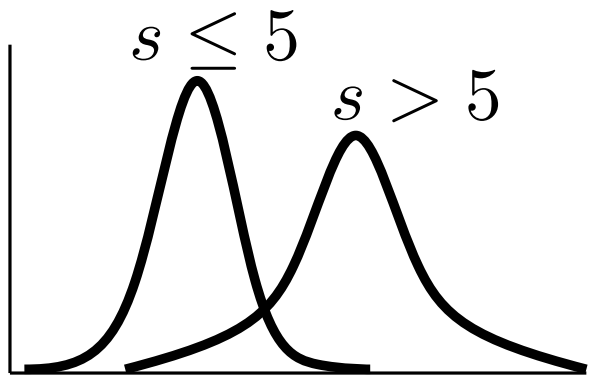
Input Choice?

Observation?

s?

$i^* = 5$





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

$p(s|o, i^*)$ ← $p(o|s, i)$

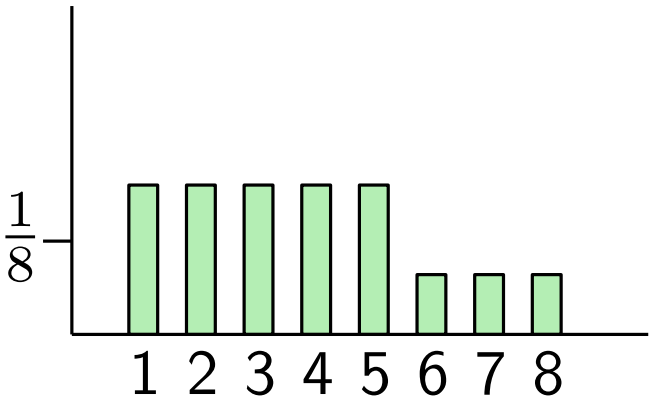
Attacker Belief?

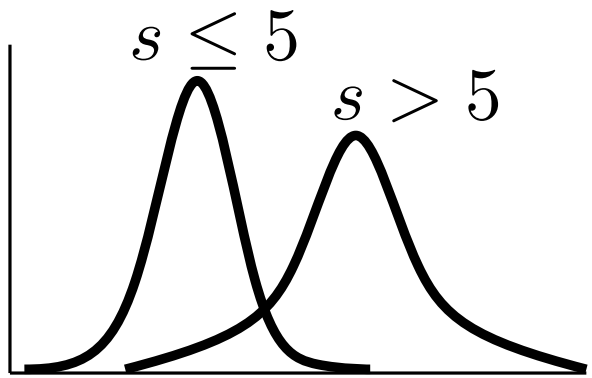
Input Choice?

Observation?

s?

$i^* = 5$





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

$p(s|o, i^*)$ ← $p(o|s, i)$

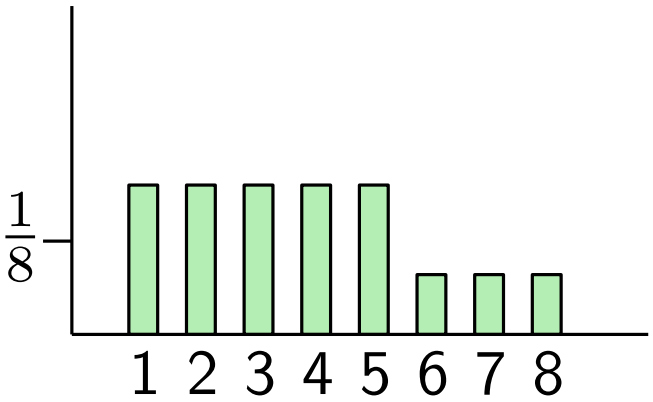
Attacker Belief?

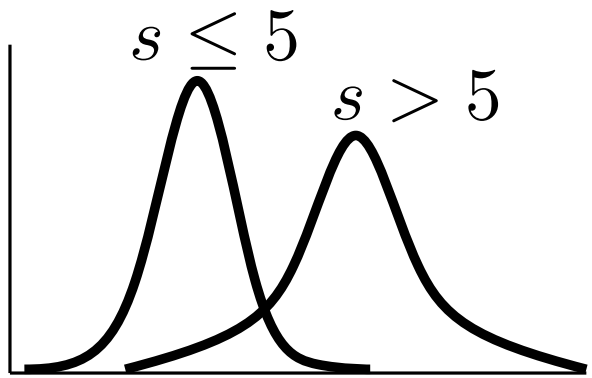
Input Choice?

Observation?

s?

$i^* = 5$





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

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

$p(o|s, i)$

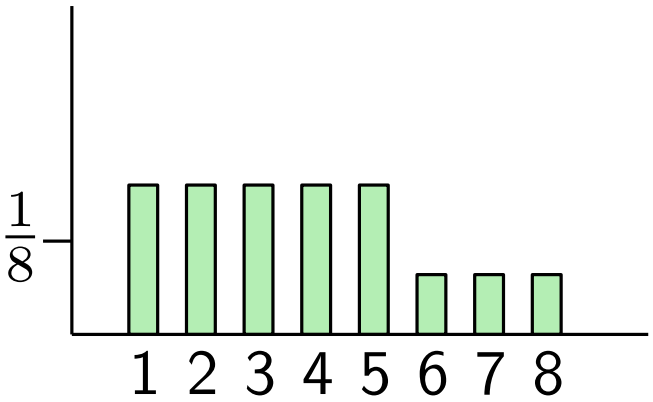
Attacker Belief?

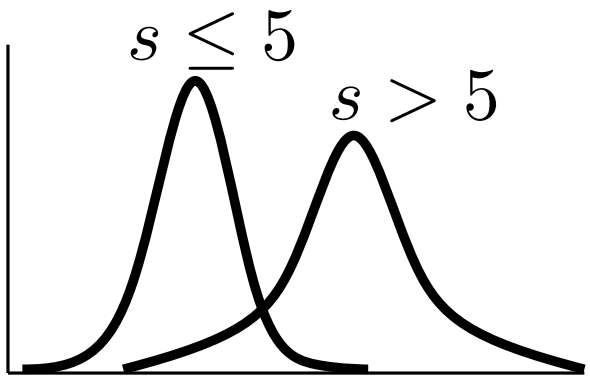
Input Choice?

Observation?

$s?$

$i^* = 5$





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

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

Bayes' Rule

$p(o|s, i)$

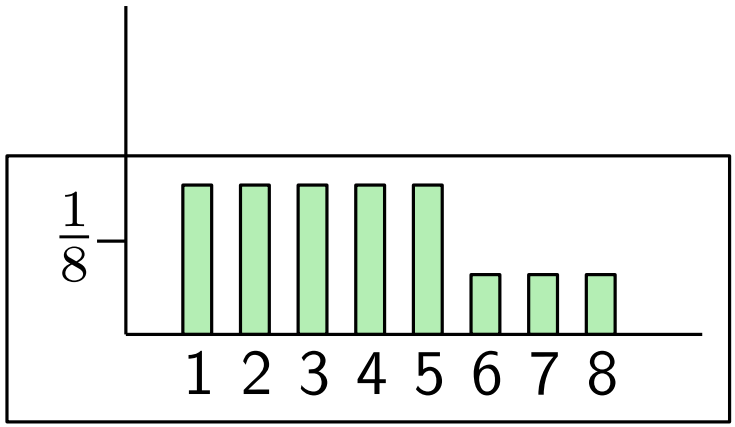
Attacker Belief?

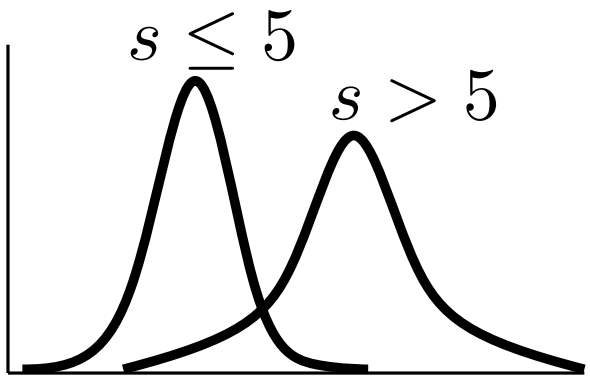
Input Choice?

Observation?

s?

$i^* = 5$



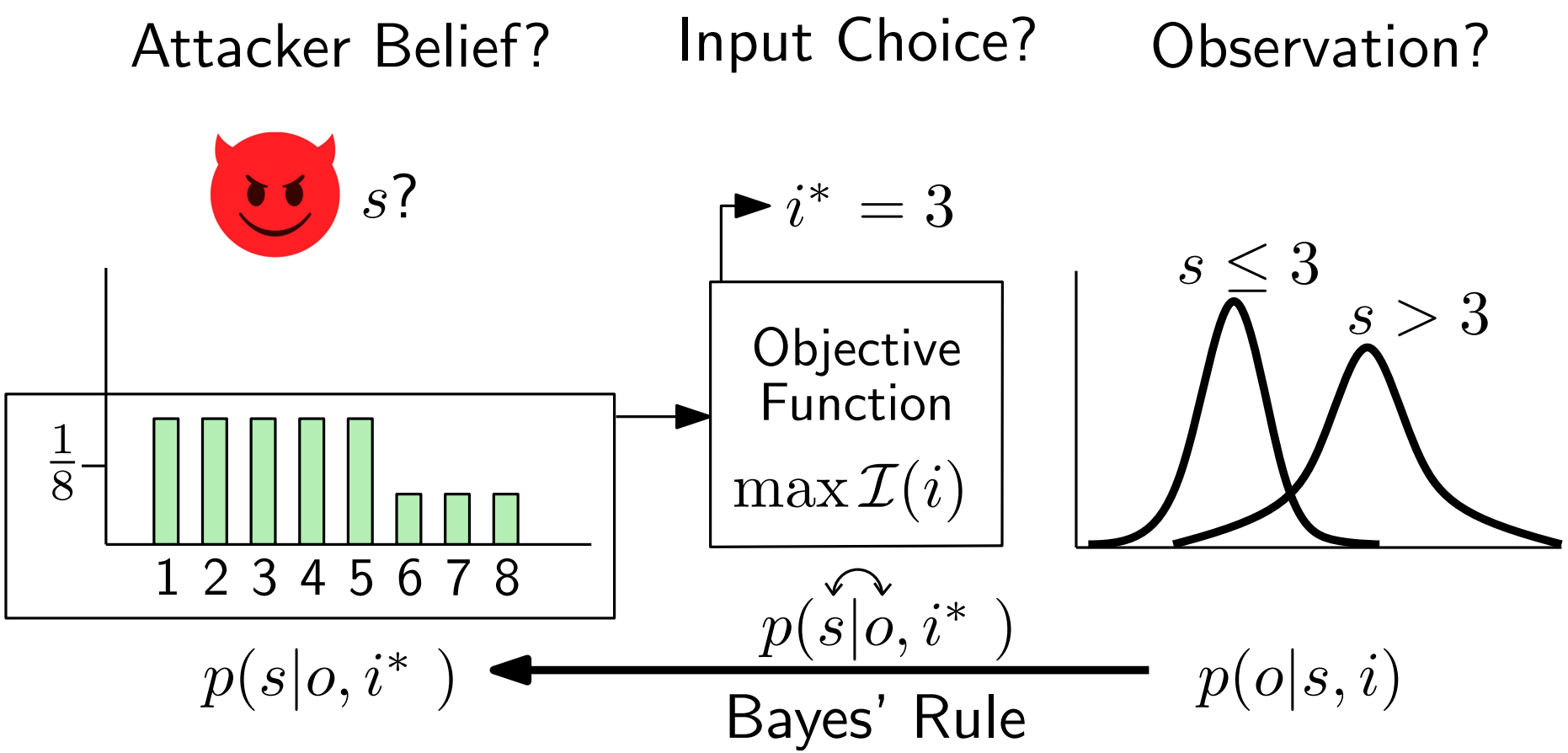


$p(s|o, i^*)$

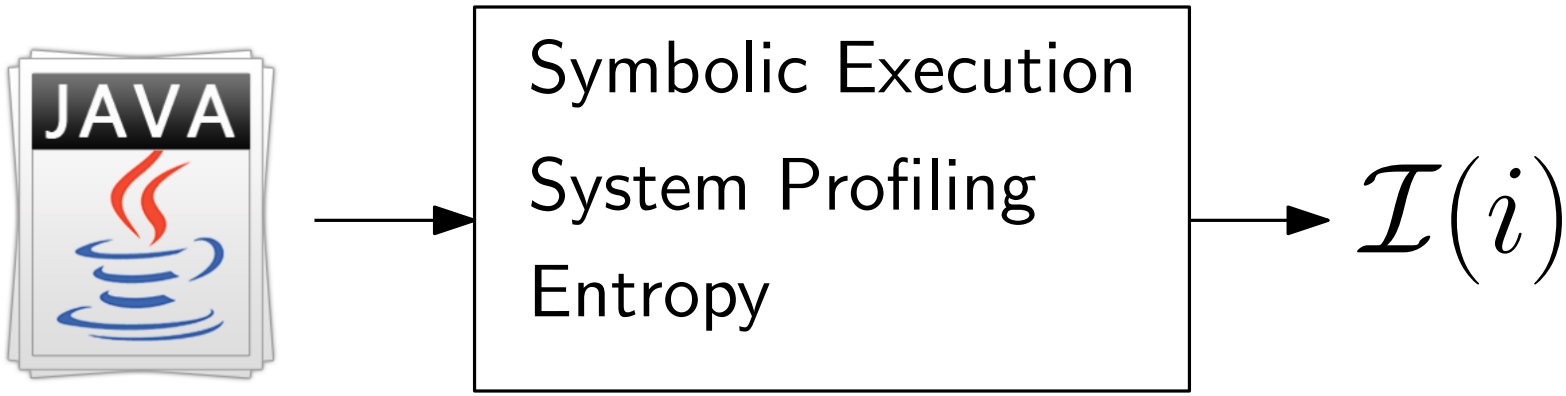
$p(s|o, i^*)$

Bayes' Rule

$p(o|s, i)$


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

Our Approach



$\mathcal{I}(i)$ 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

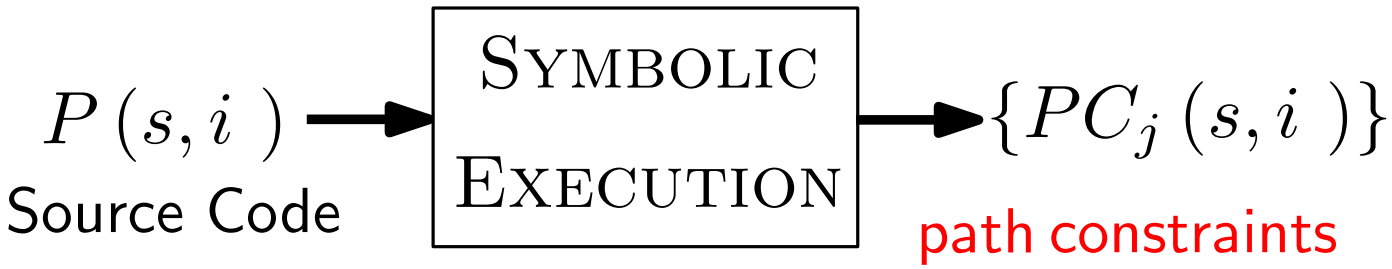
1. Offline Static Analysis

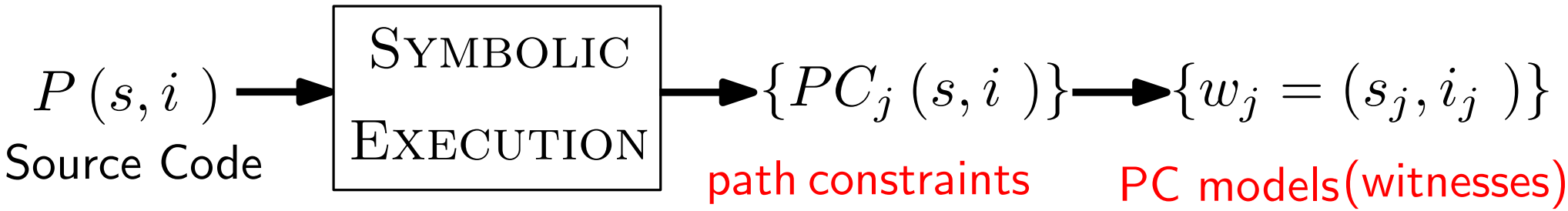
2. Offline Dynamic Analysis

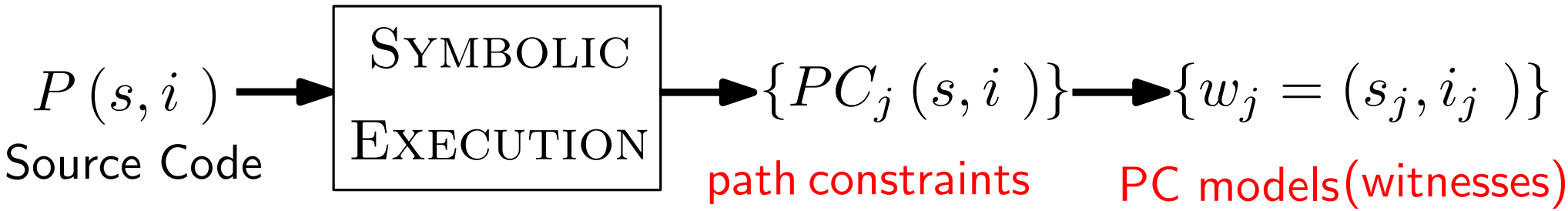
3. Online Attack Synthesis

$P\,(s, i\ )$

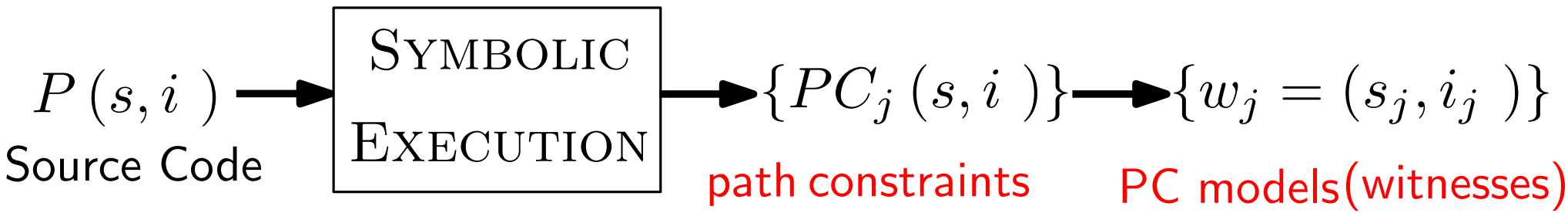
Source Code


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

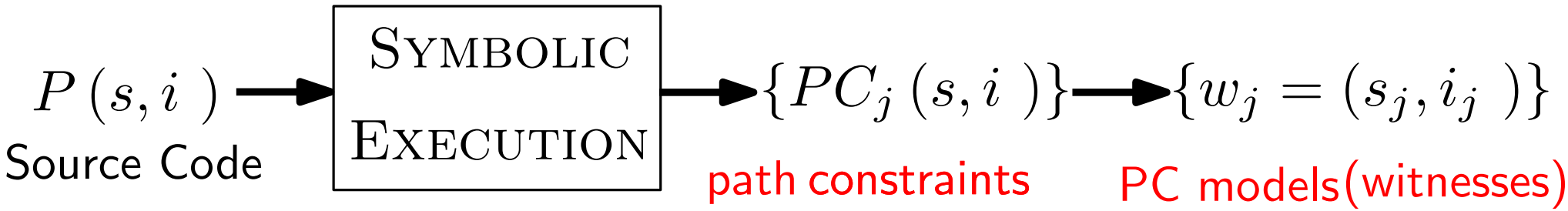


Each PC characterizes an observable program behavior



Each PC characterizes an observable program behavior

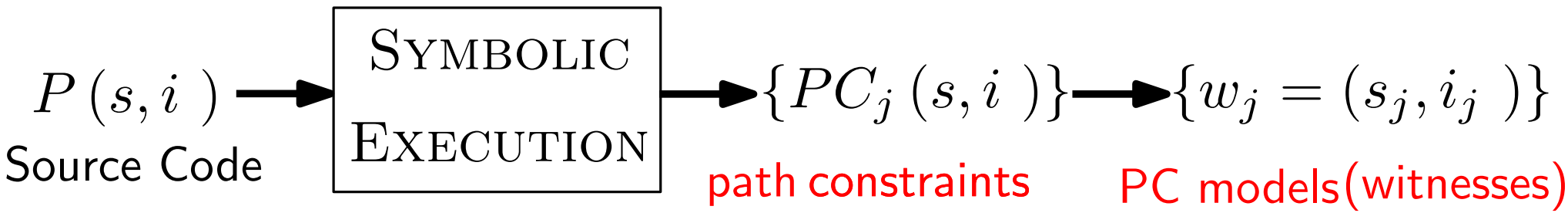
$$(s, i\,) \models PC_j \qquad\qquad (s', i'\,) \models PC_j$$



Each PC characterizes an observable program behavior

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

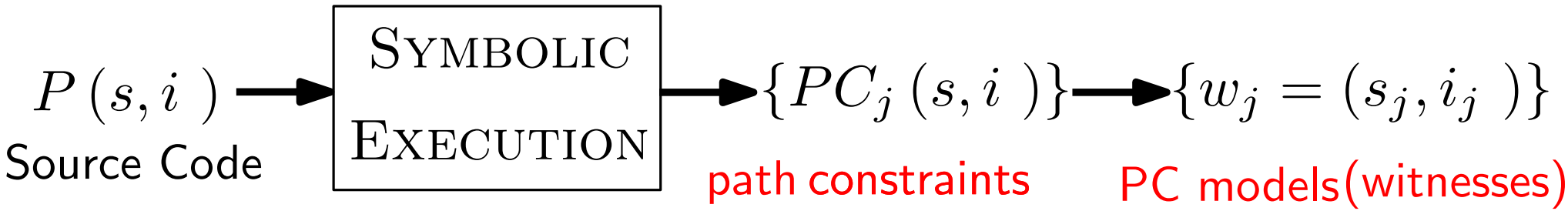
$$P\,(s, i\,) \qquad\qquad P\,(s', i'\,)$$



Each PC characterizes an observable program behavior

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

$$P\,(s, i\,) \quad ? \quad P\,(s', i'\,)$$



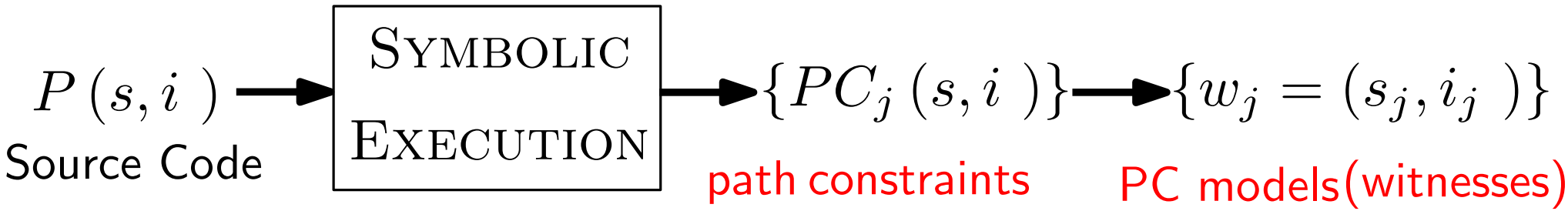
Each PC characterizes an observable program behavior

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

$$P\,(s,i\,) \quad ? \quad ? \quad P\,(s',i'\,)$$

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

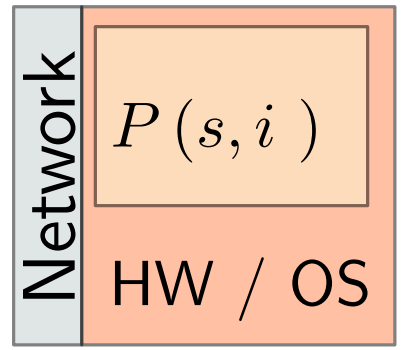
1. Offline Static Analysis

2. Offline Dynamic Analysis

3. Online Attack Synthesis

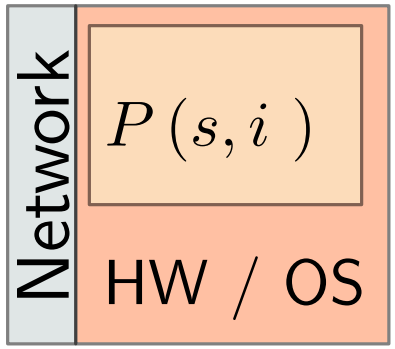
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

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



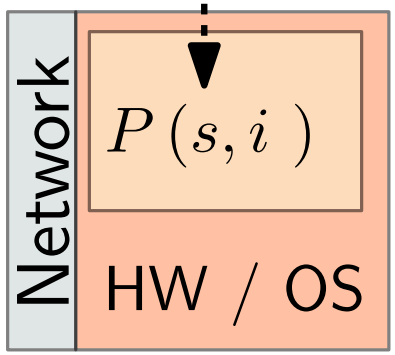
Characterize effect of noise on each class of program behaviors using the witness for that behavior.

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

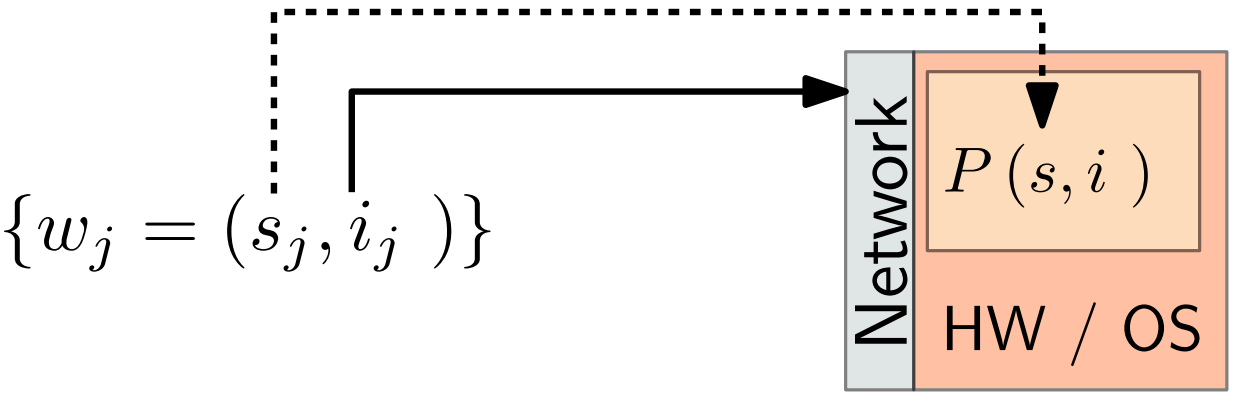


Characterize effect of noise on each class of program behaviors using the witness for that behavior.

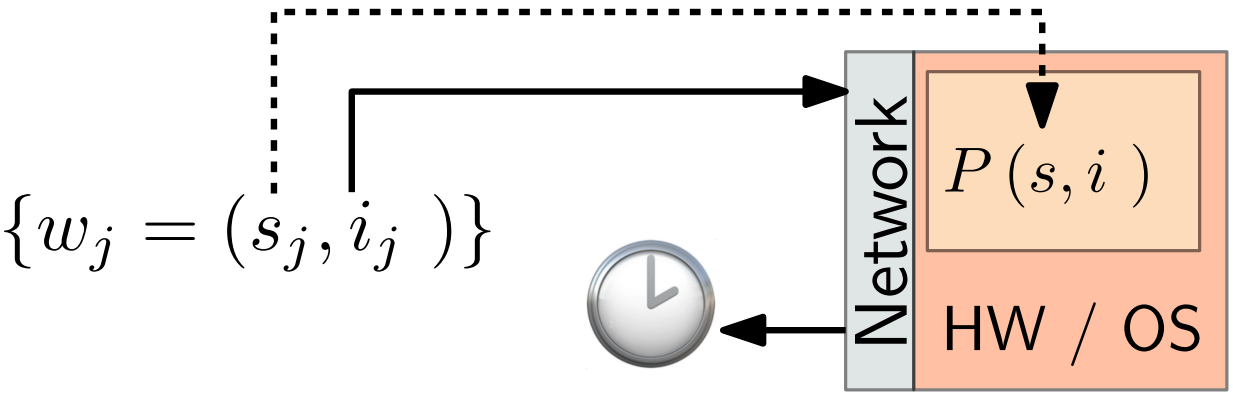
$\{w_j = (s_j, i_j\ )\}$



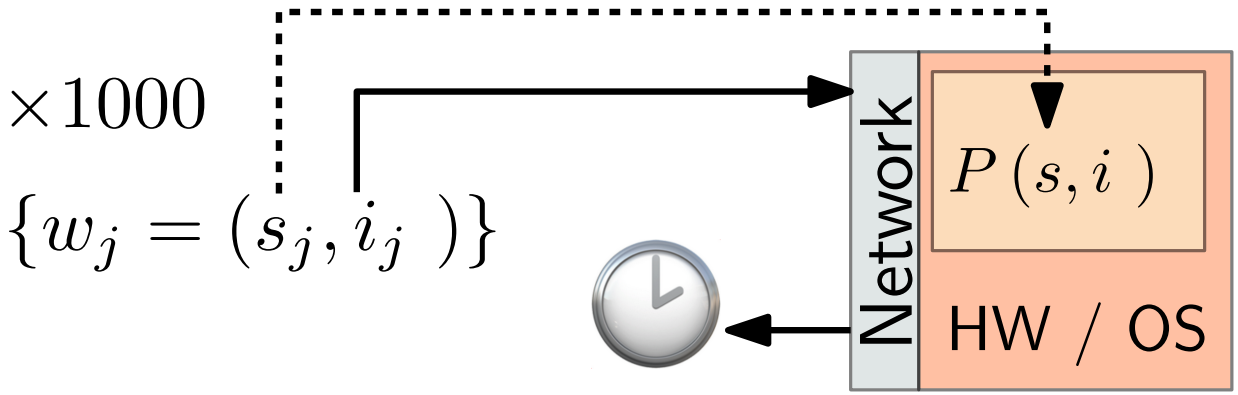
Characterize effect of noise on each class of program behaviors using the witness for that behavior.



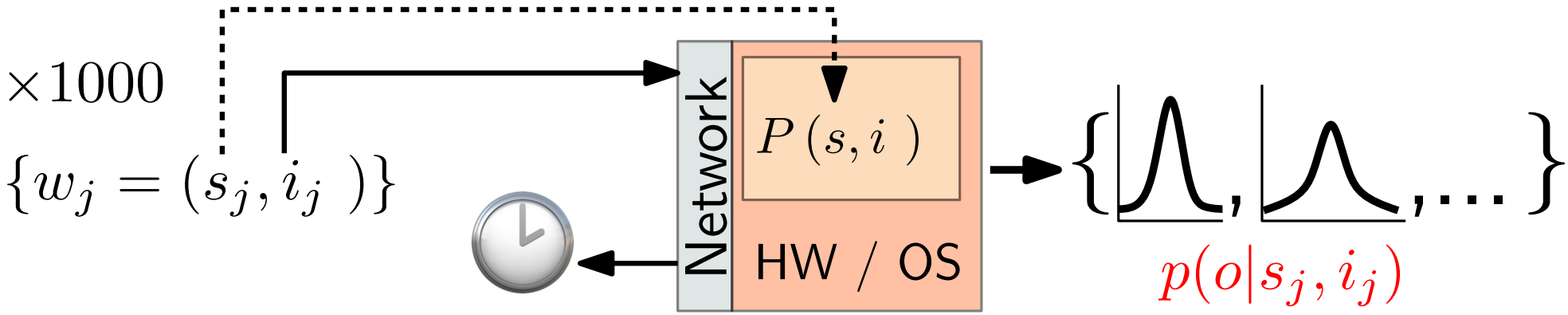
Characterize effect of noise on each class of program behaviors using the witness for that behavior.



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



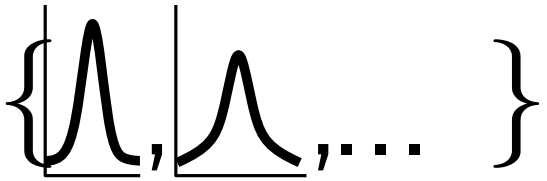
Characterize effect of noise on each class of program behaviors using the witness for that behavior.



1. Offline Static Analysis

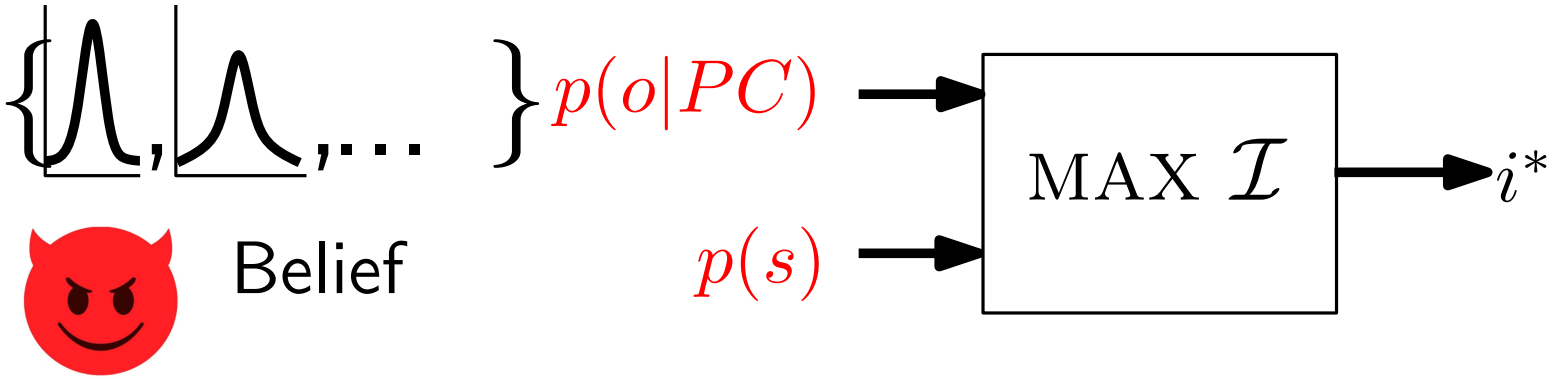
2. Offline Dynamic Analysis

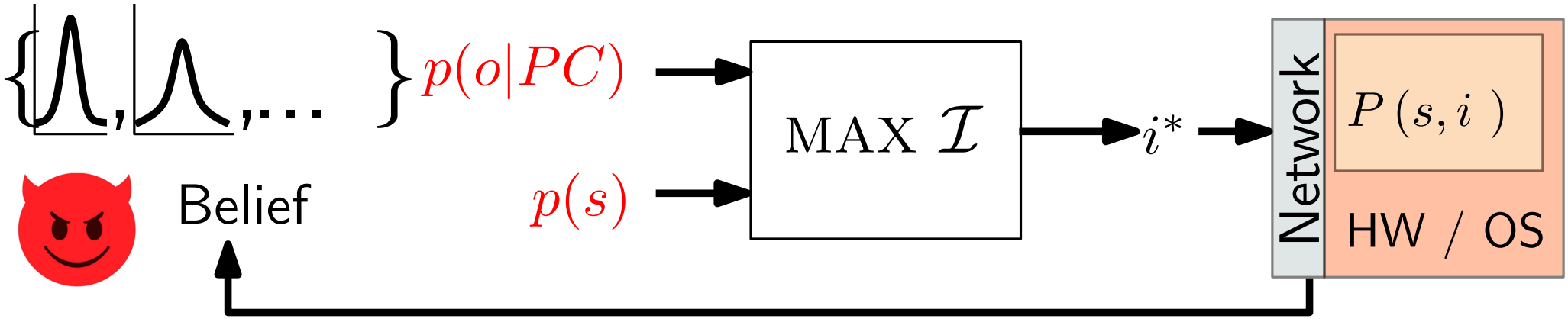
3. Online Attack Synthesis

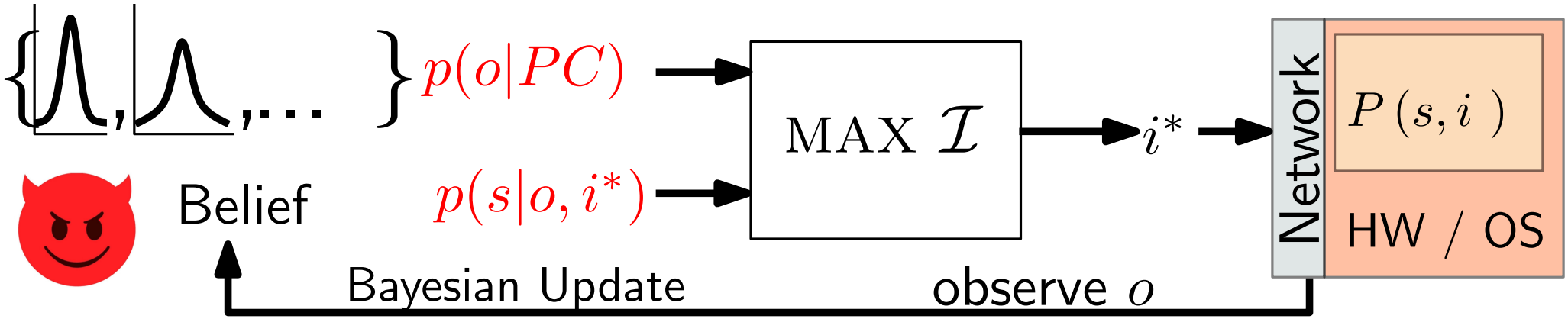


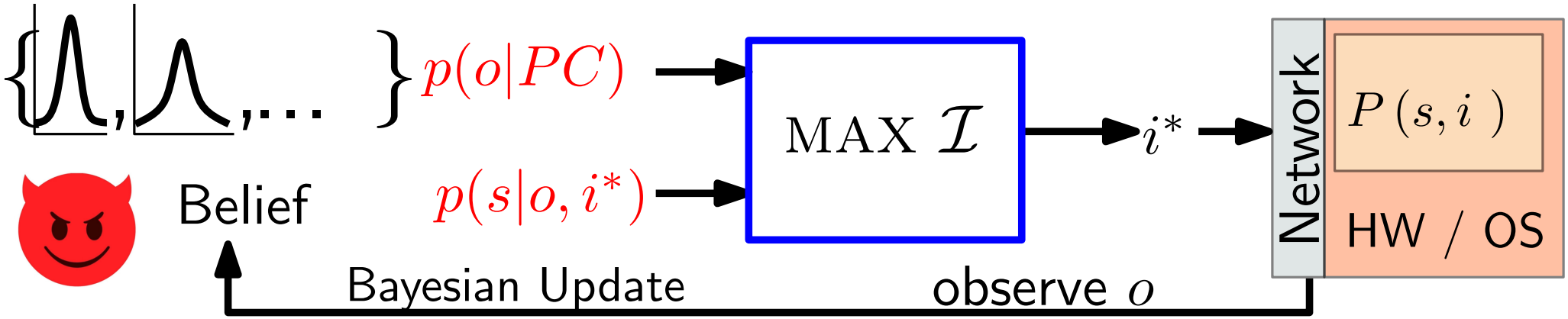
$p(o|PC)$

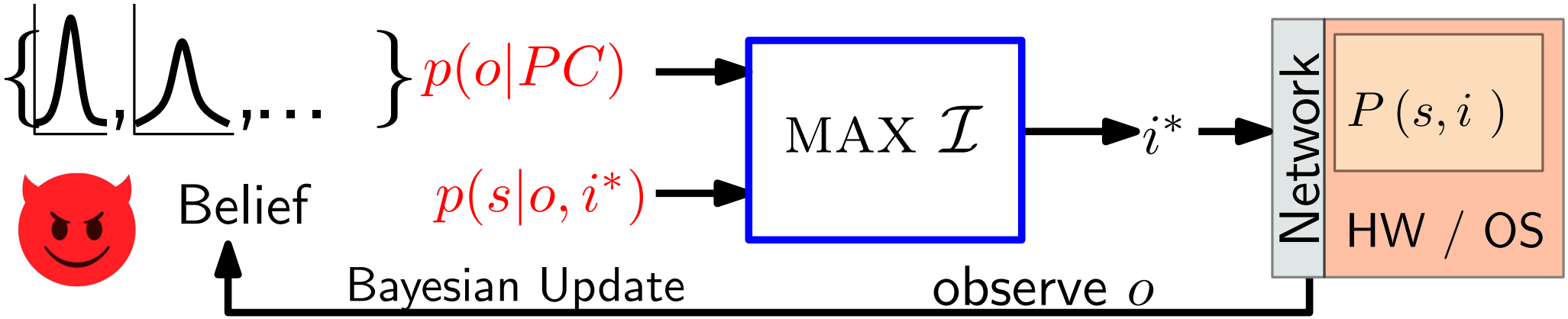
Belief $p(s)$


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


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


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

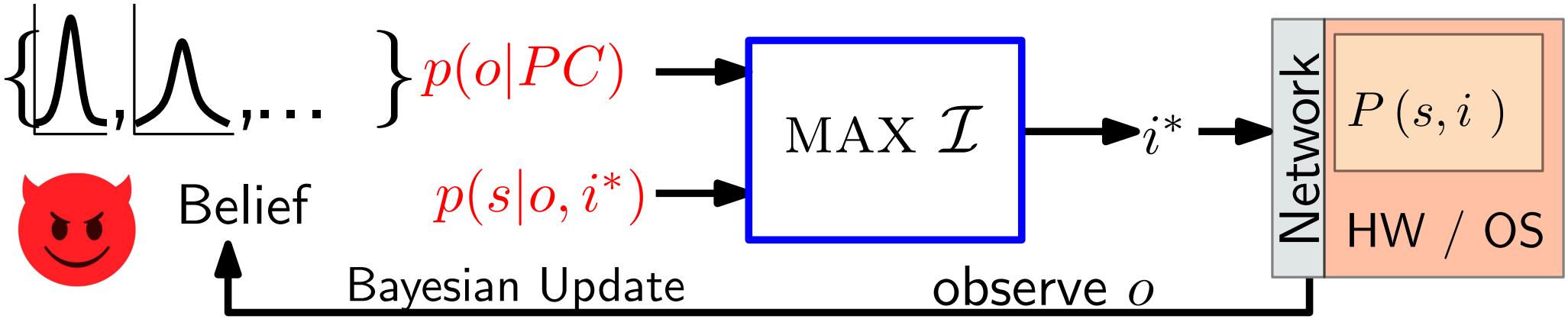

$p(o|PC)$
$p(s|o,i^*)$
MAX $\mathcal{I}$
i^*
Network
$P\,(s,i\,)$
HW / OS
Belief
Bayesian Update
observe o



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

Expected info gain given attacker input

Path constraint probabilities

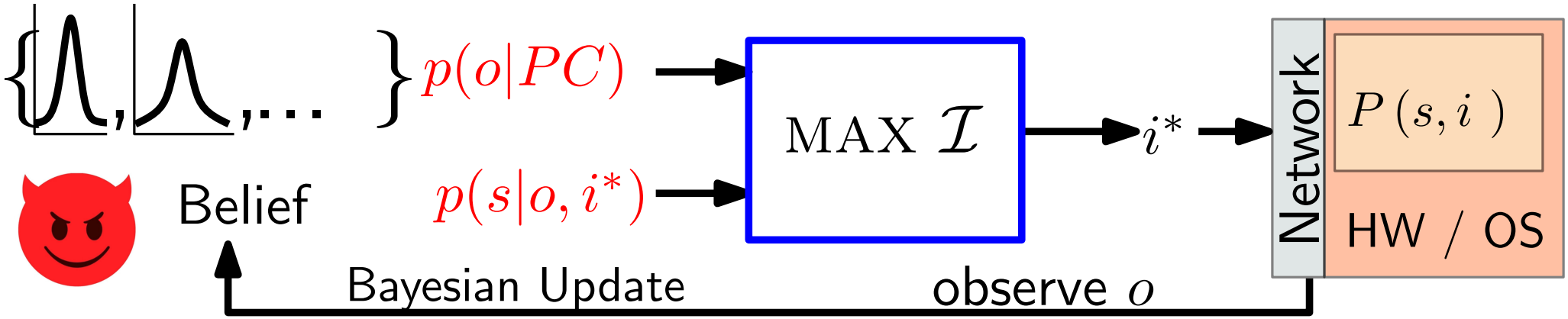


$$\mathcal{I}(s; PC_j|i) = -\sum_{j=1}^{n} p(PC_j|i) \log_2 p(PC_j|i)$$

Expected info gain given attacker input

Path constraint probabilities

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



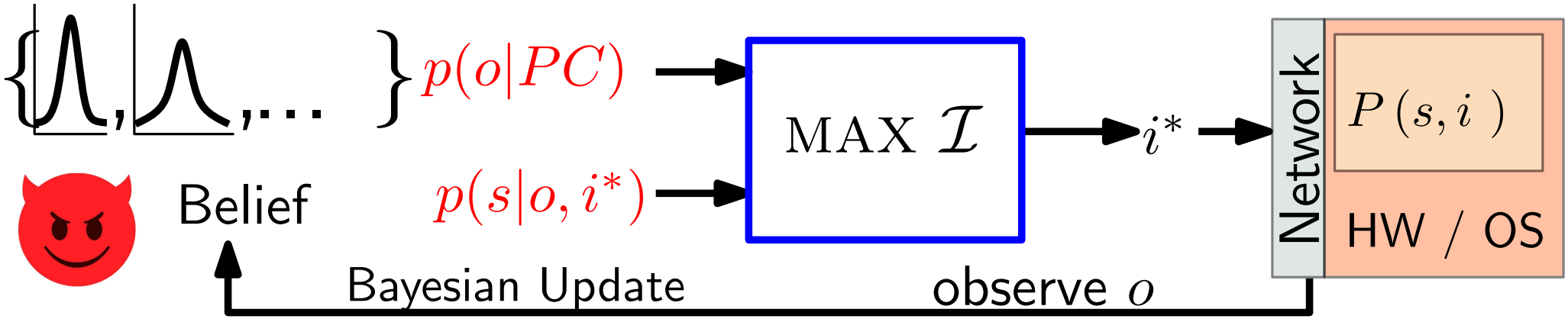
$$\mathcal{I}(s; PC_j|i) = -\sum_{j=1}^{n} p(PC_j|i) \log_2 p(PC_j|i)$$

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



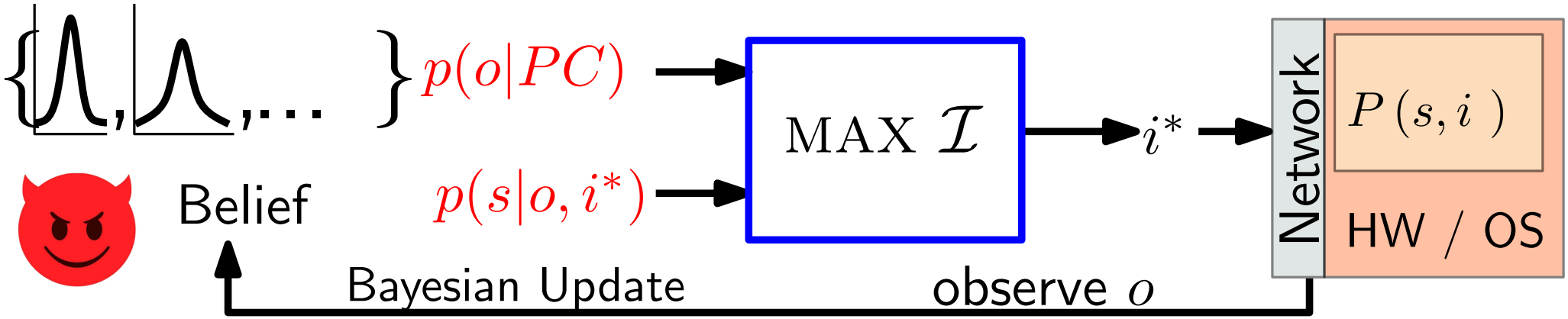
$$\mathcal{I}(s; PC_j|i) = -\sum_{j=1}^{n} p(PC_j|i) \log_2 p(PC_j|i)$$

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



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

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

Barvinok

Weighted Symbolic Model Counting

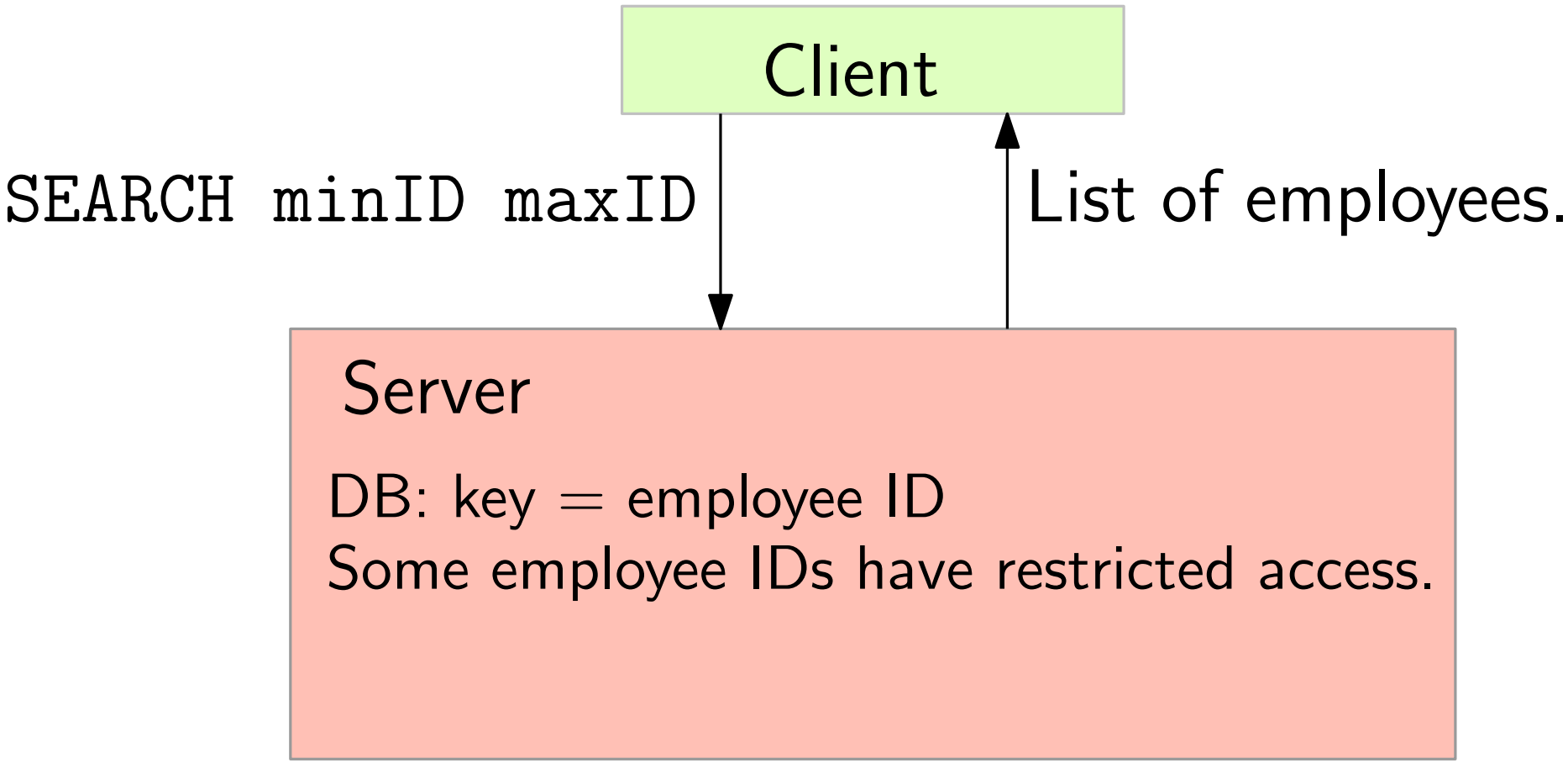
Mathematica

Symbolic Entropy Computation

Numeric Maximization

Case Study: LawDB

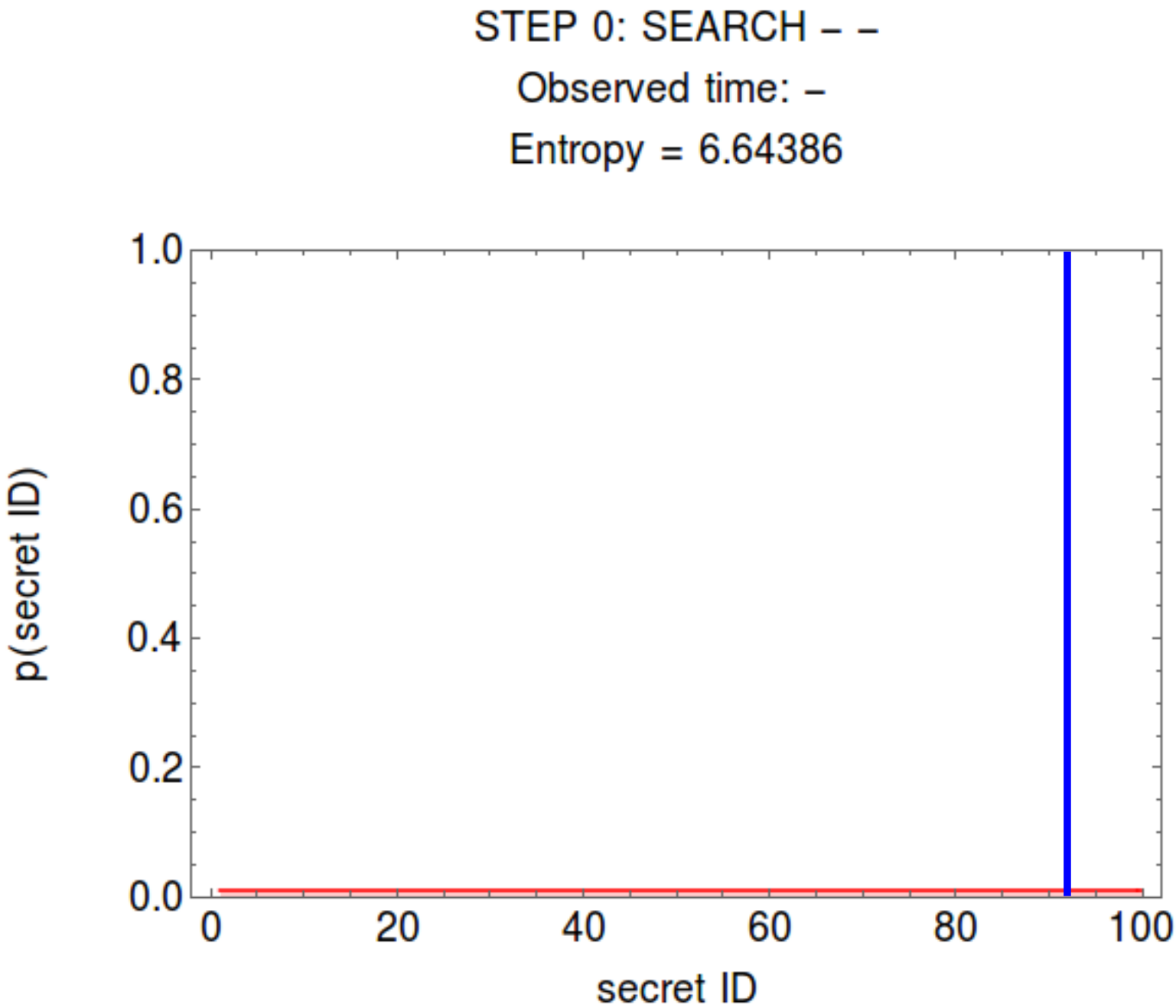
From Defense Advanced Research Projects Agency (DARPA)
Space-Time Analysis for Cybersecurity (STAC) Project



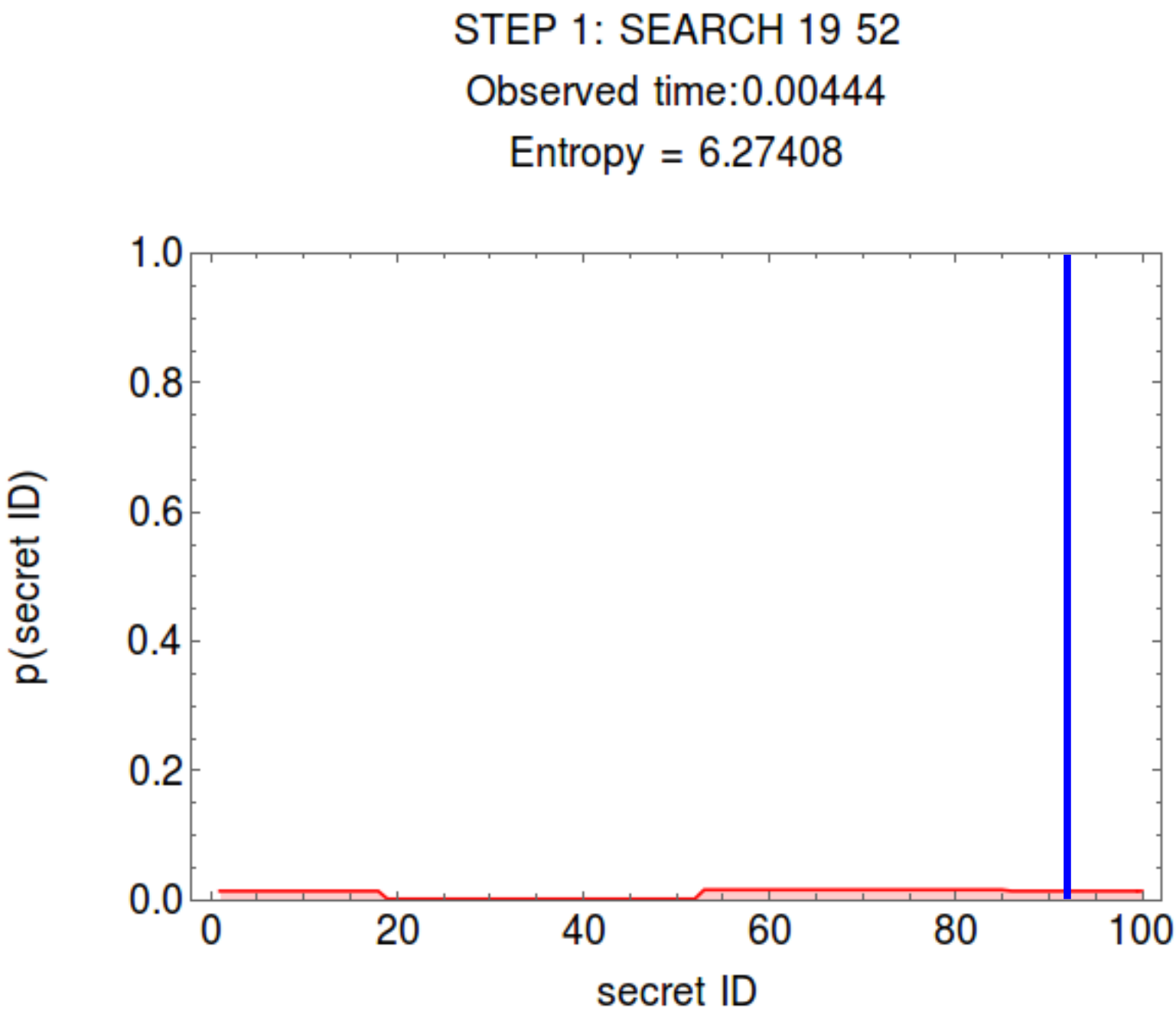
Writes to log file depending on

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

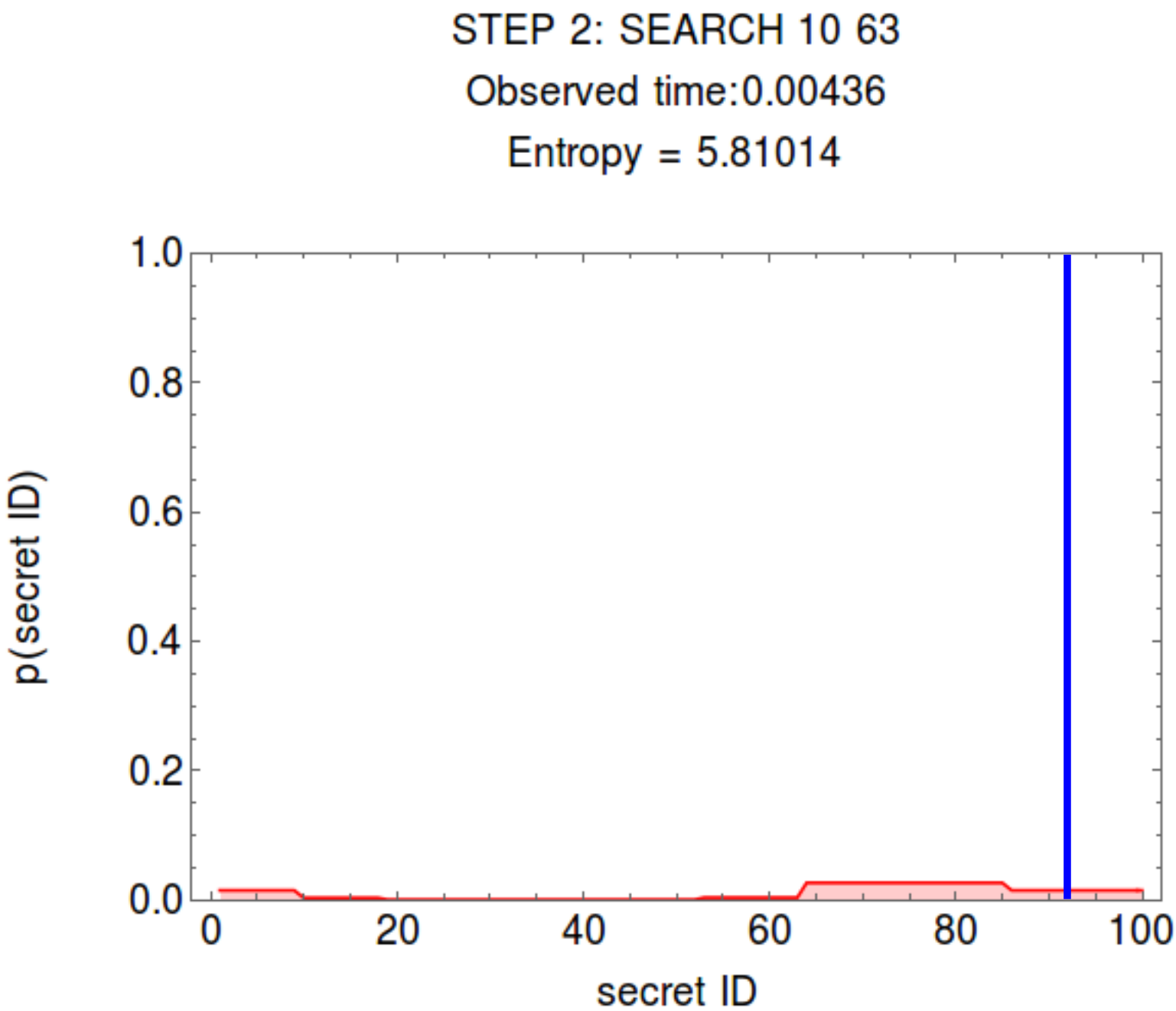
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



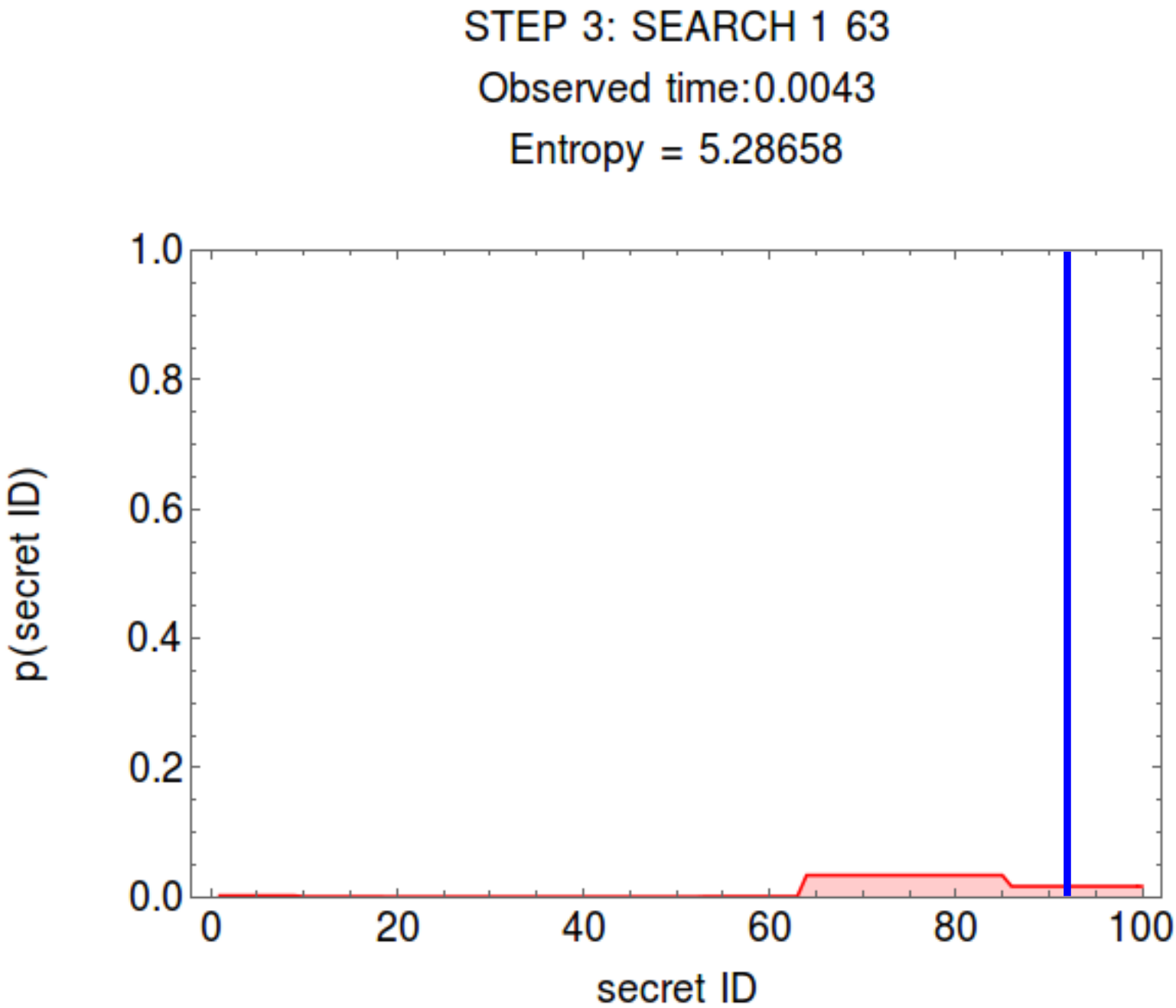
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



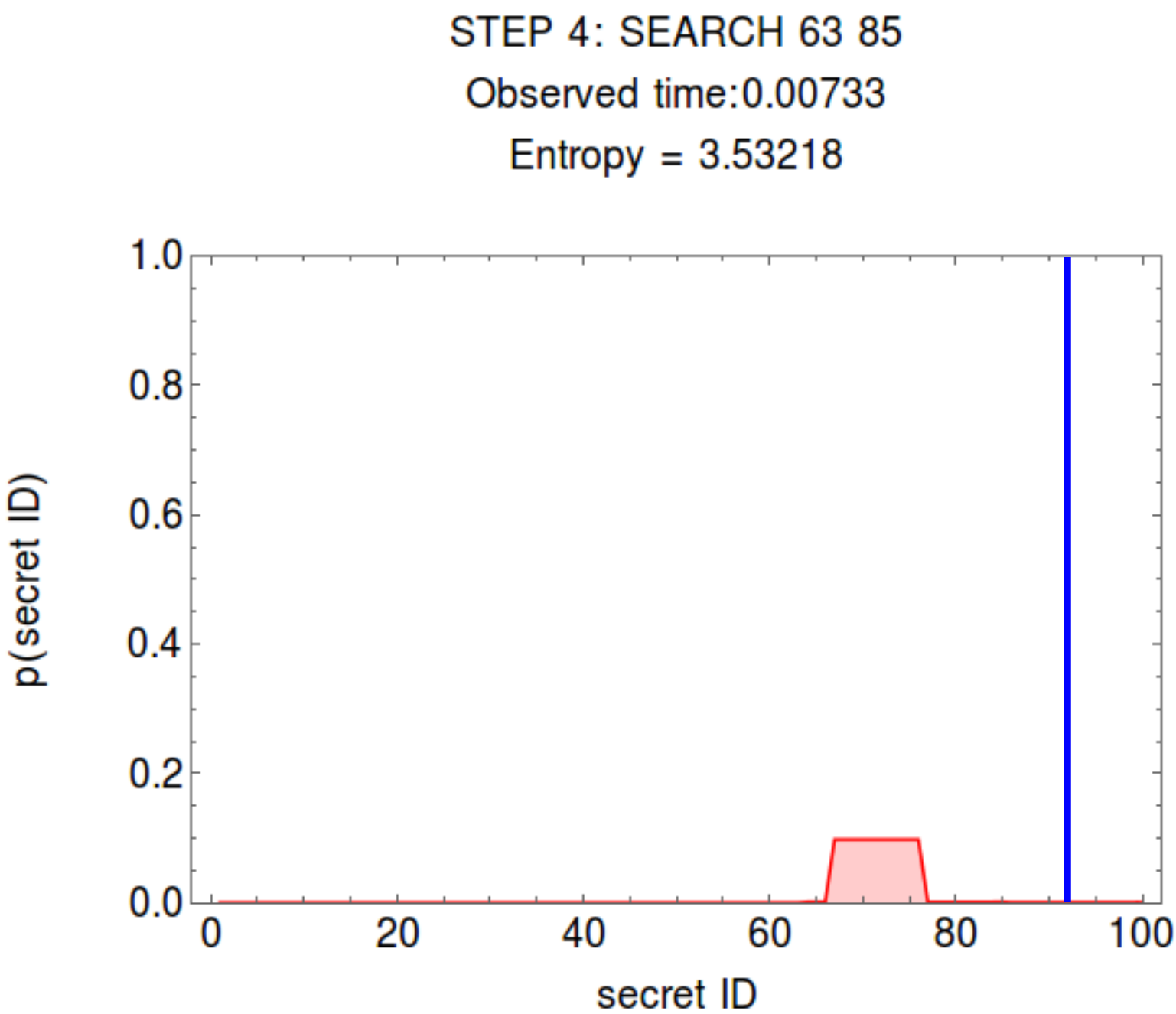
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



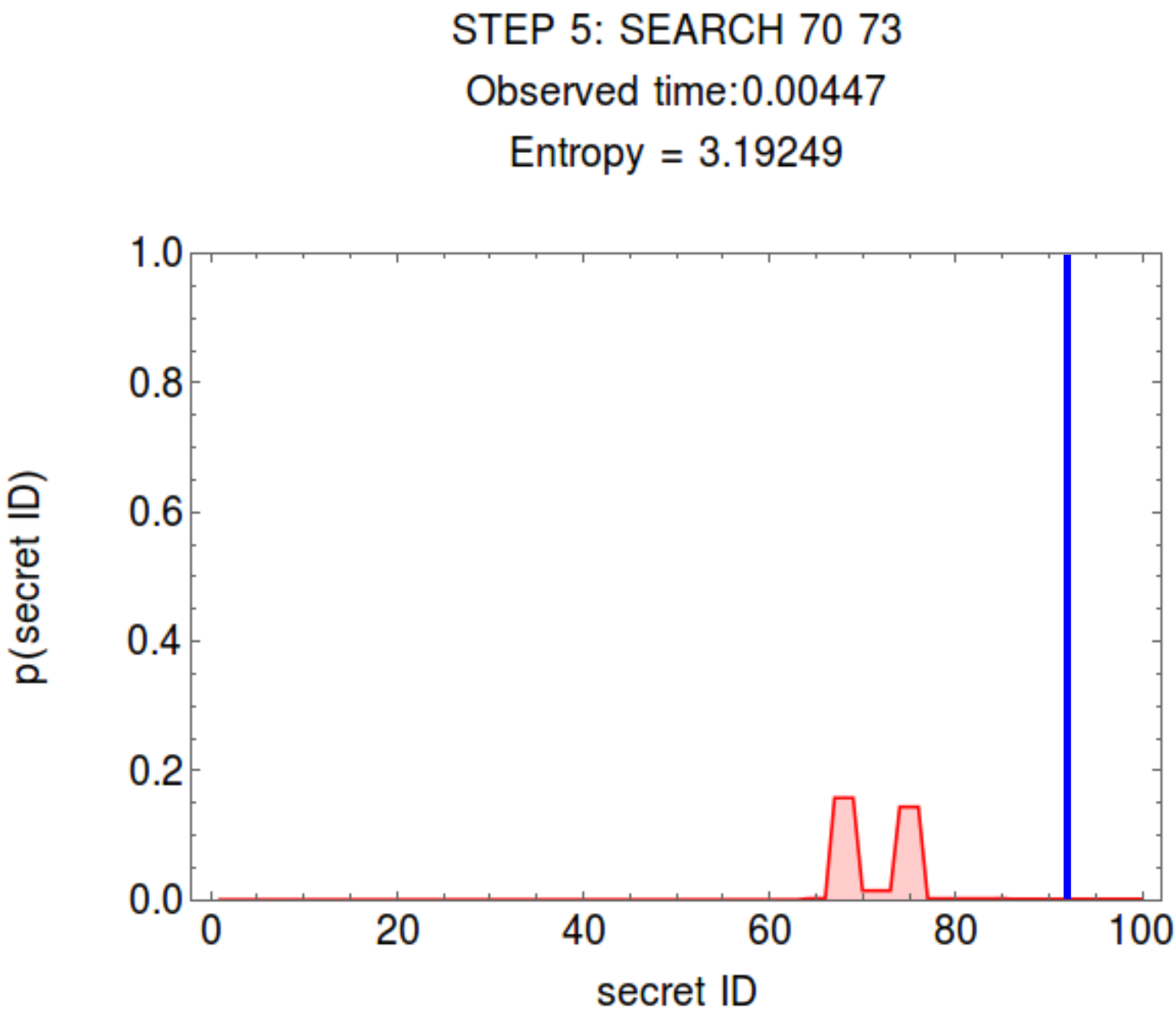
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



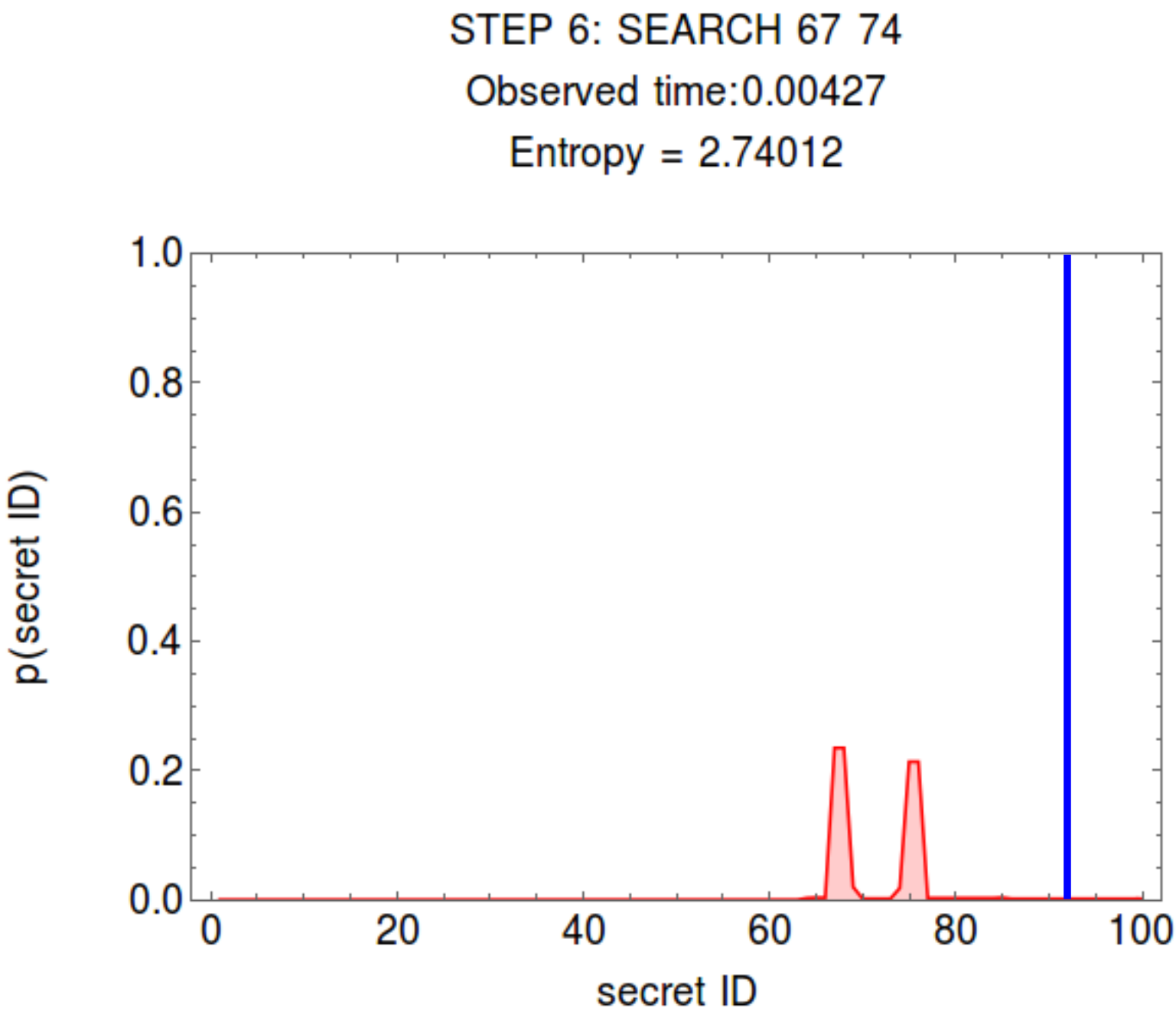
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



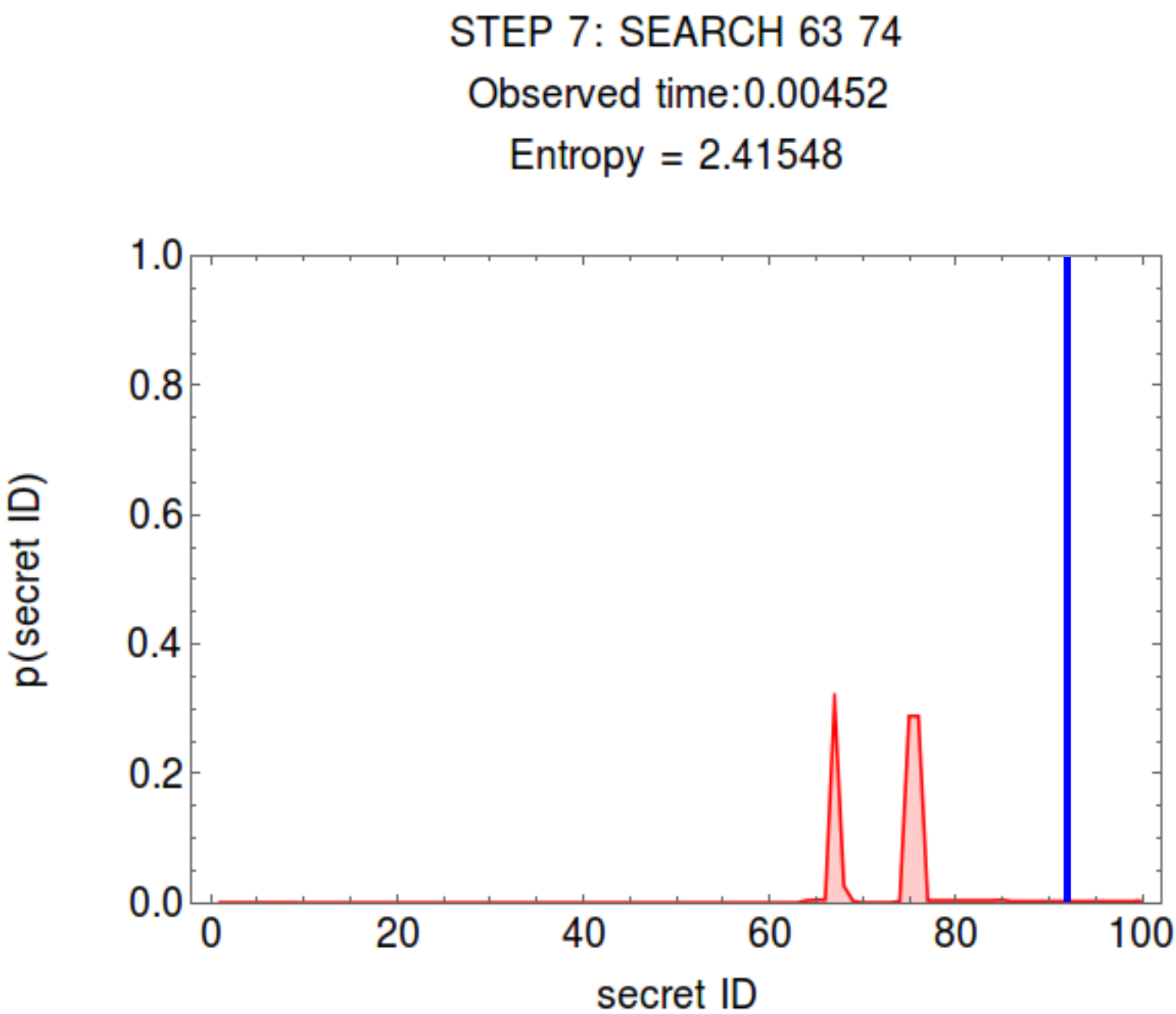
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



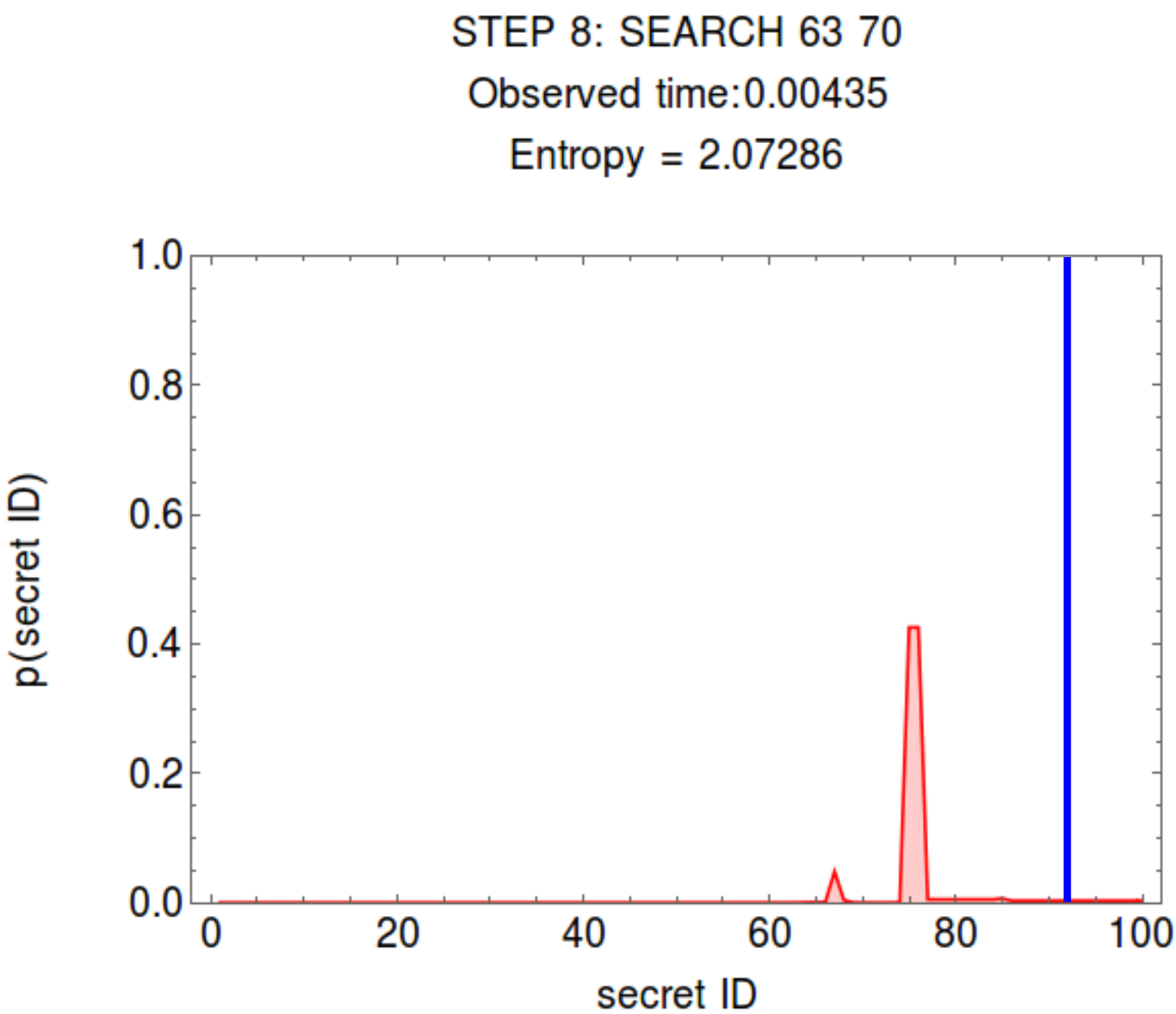
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



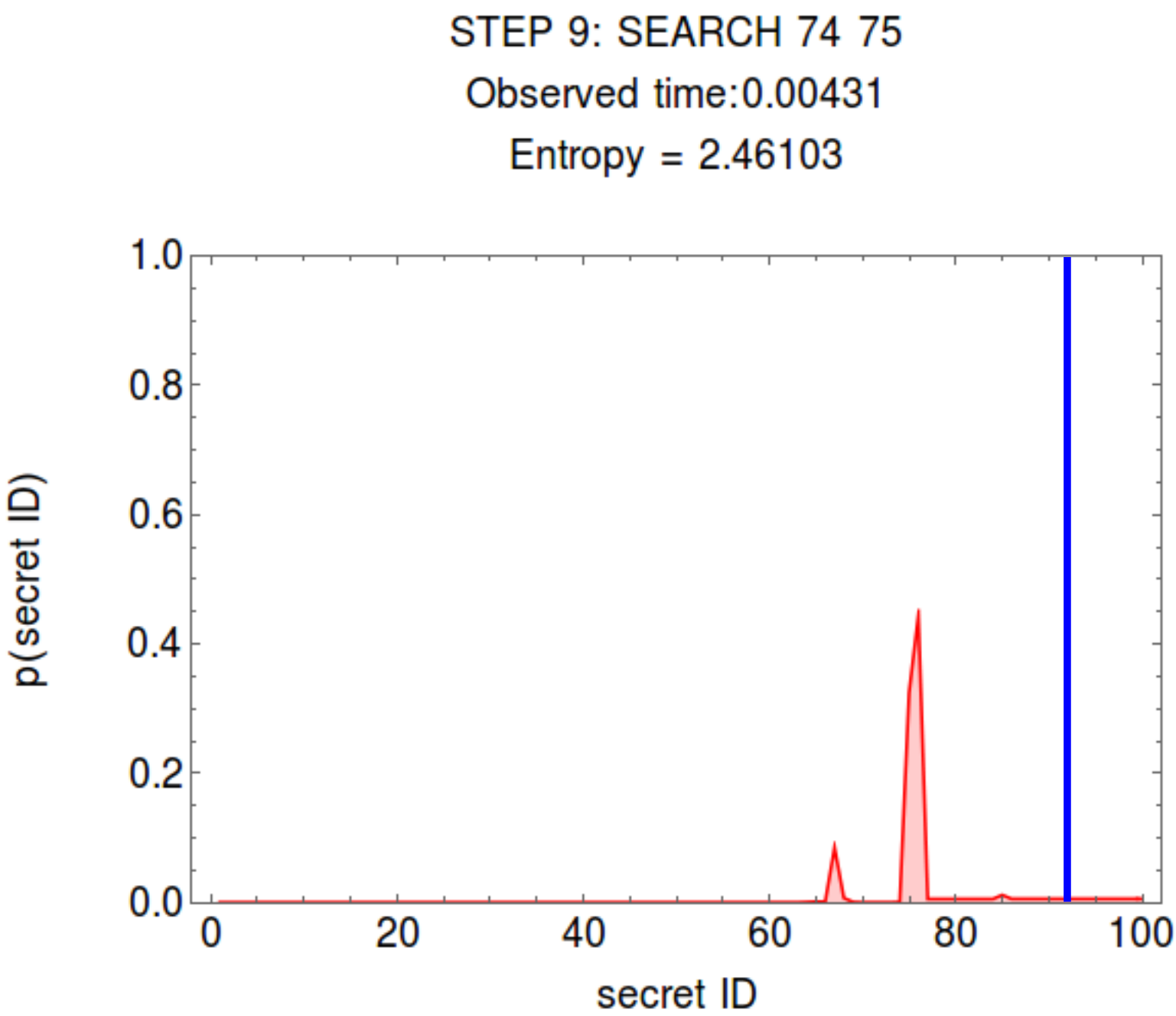
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



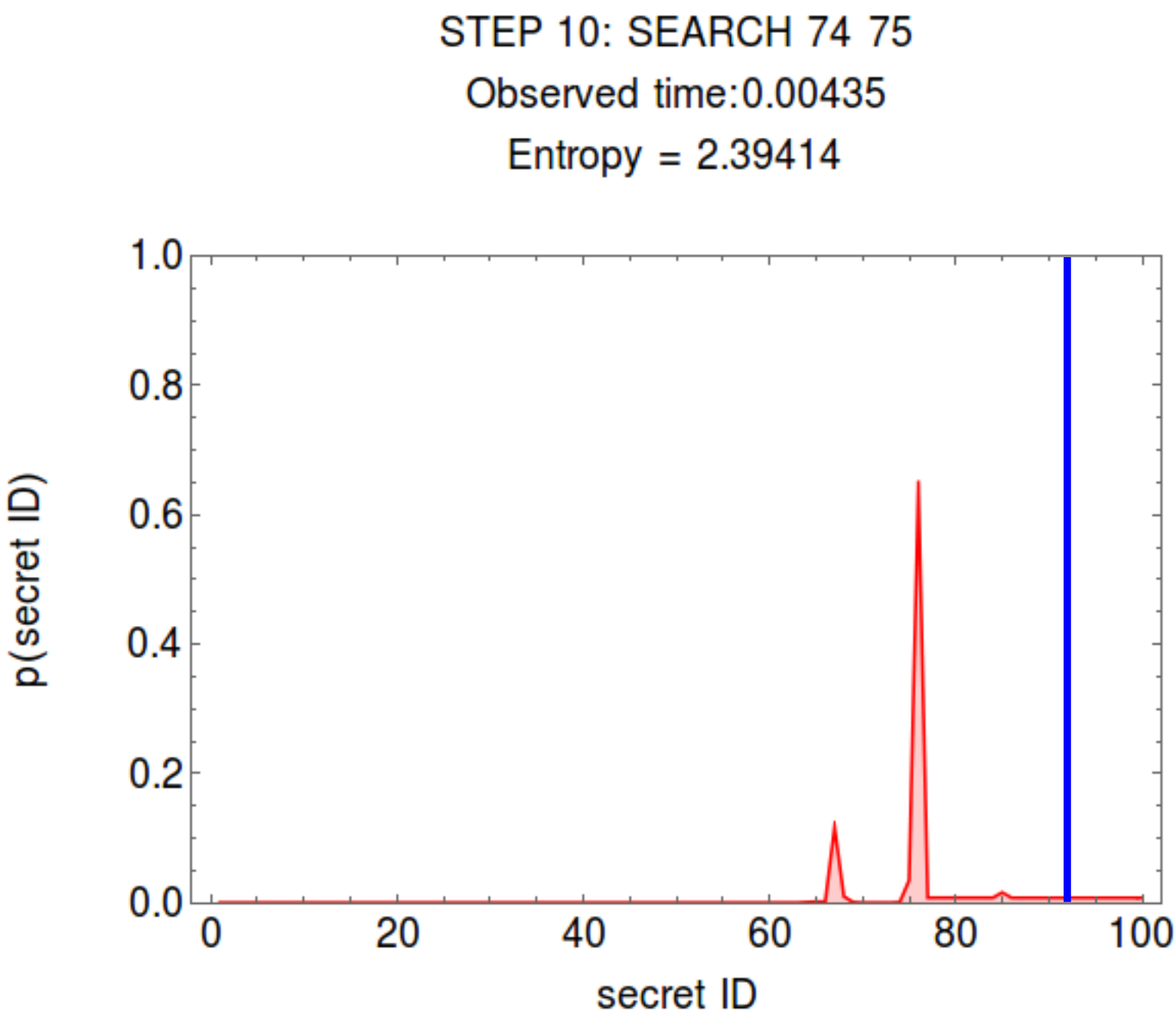
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



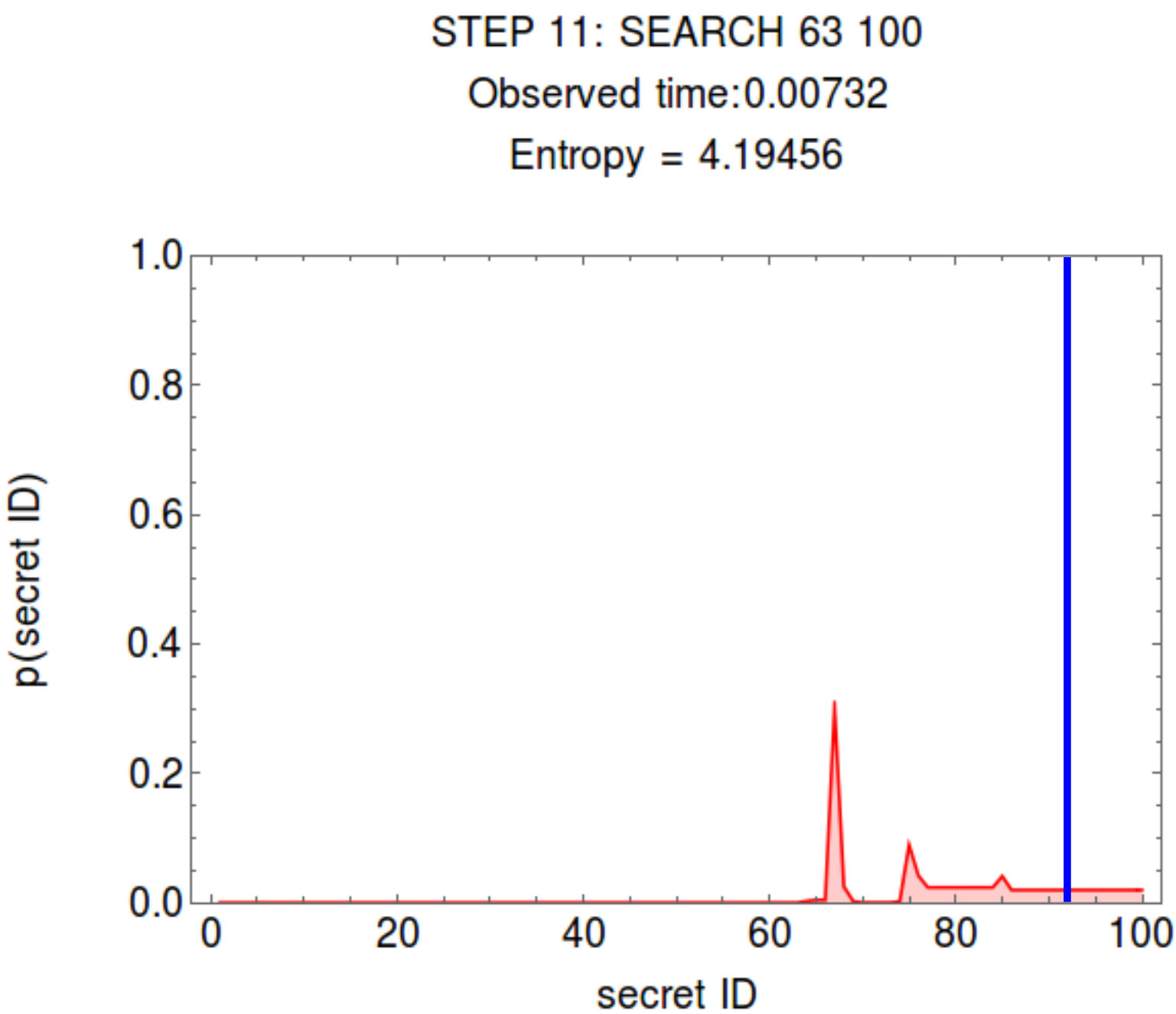
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



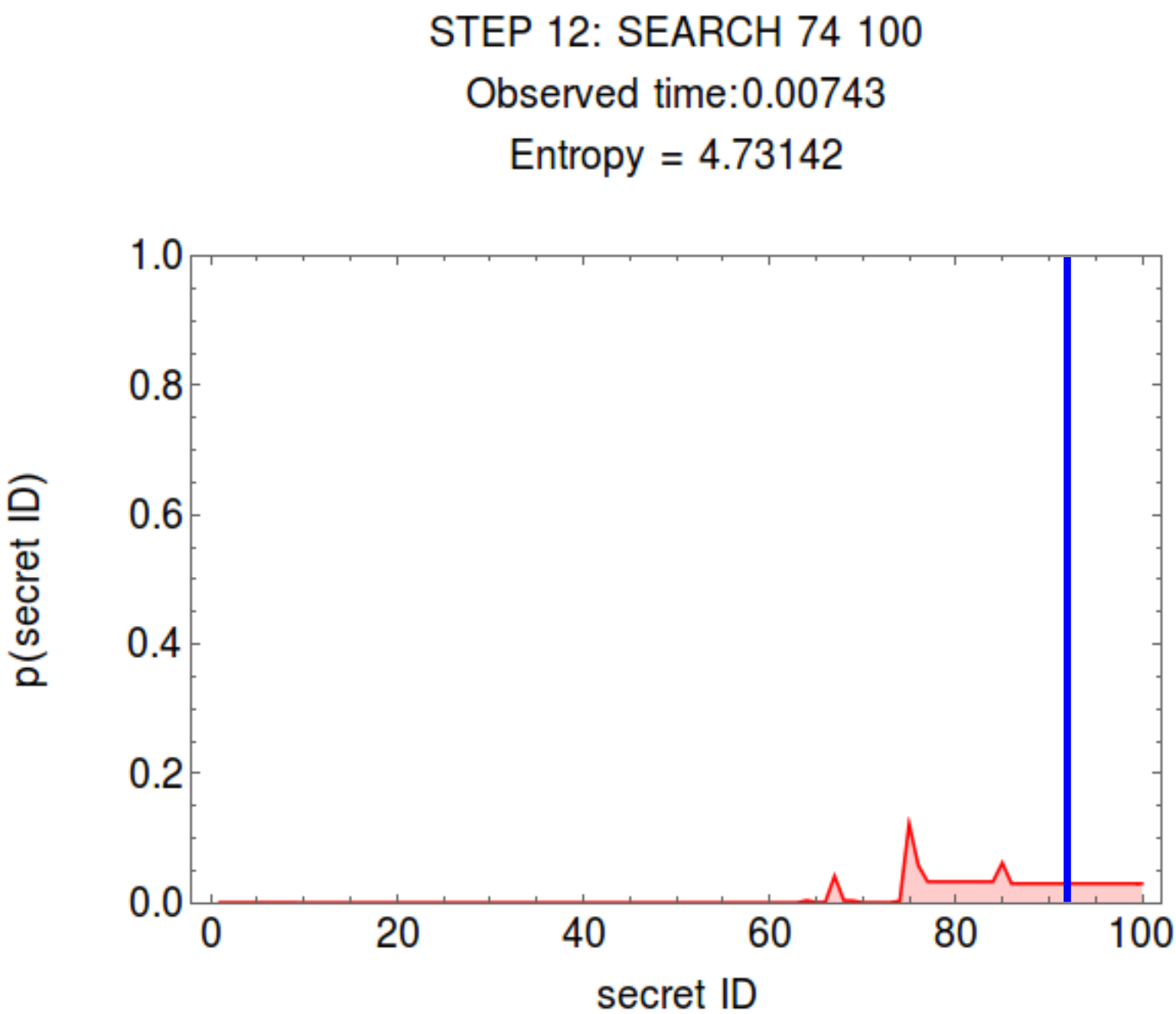
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



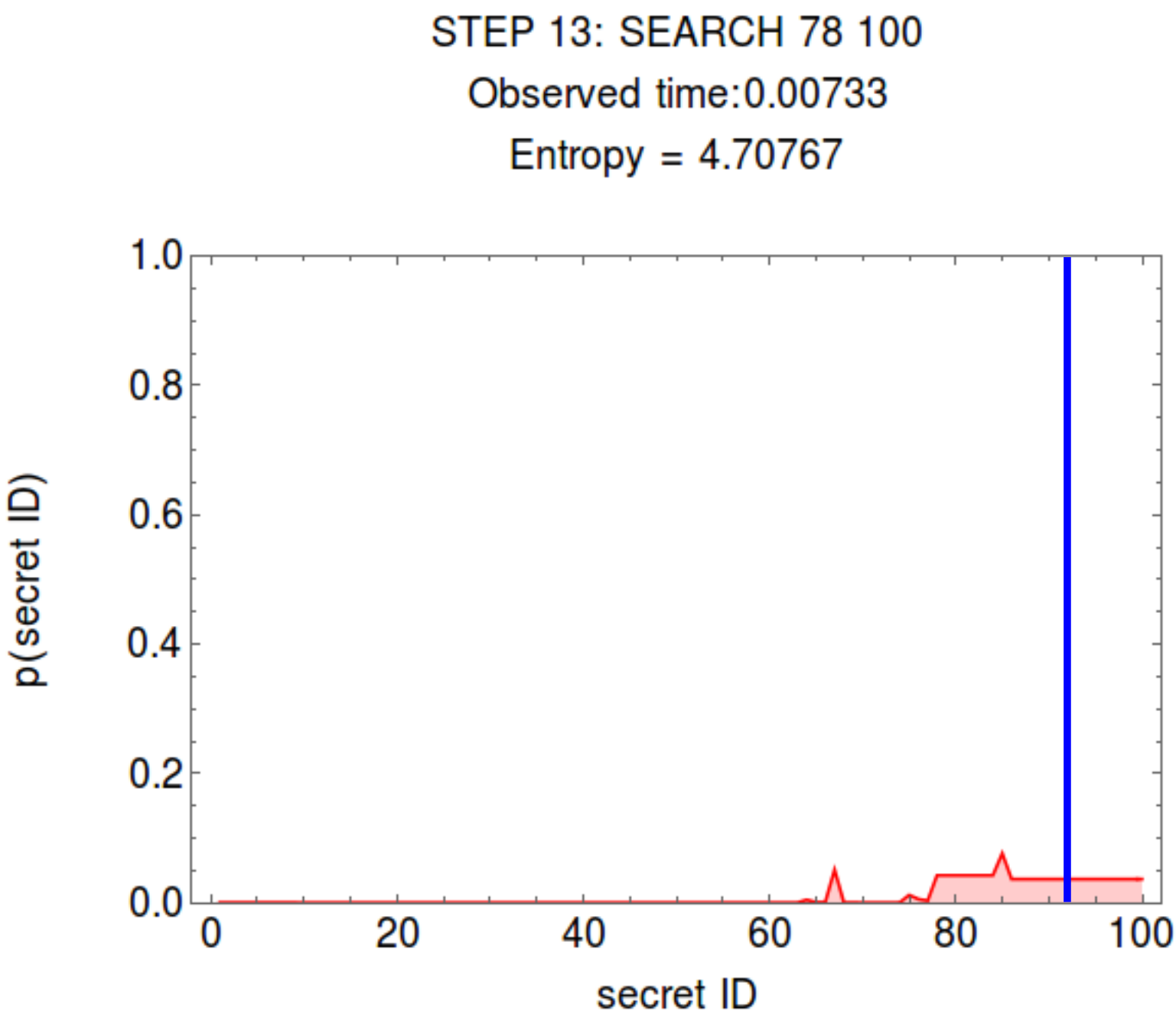
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



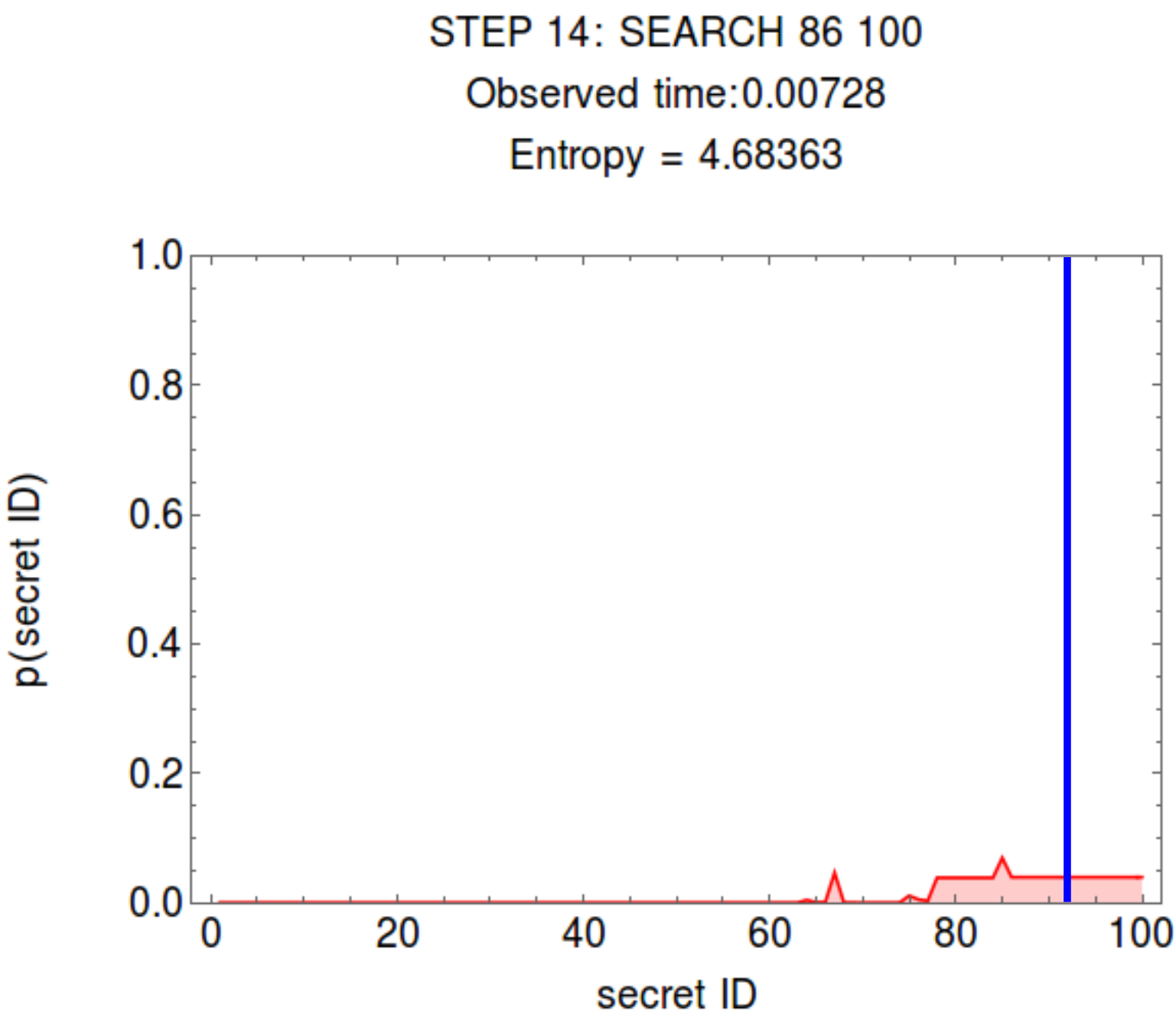
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



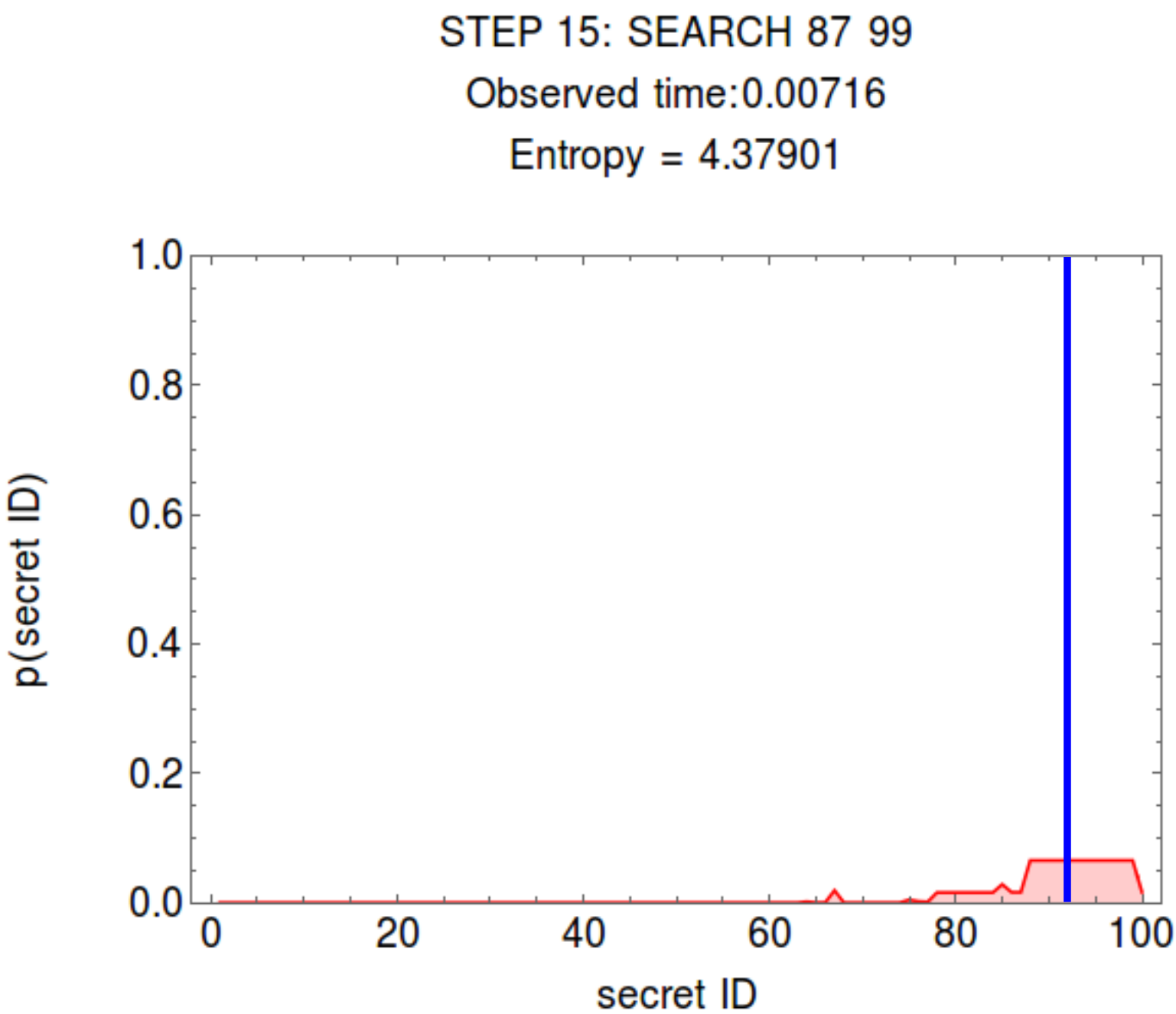
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



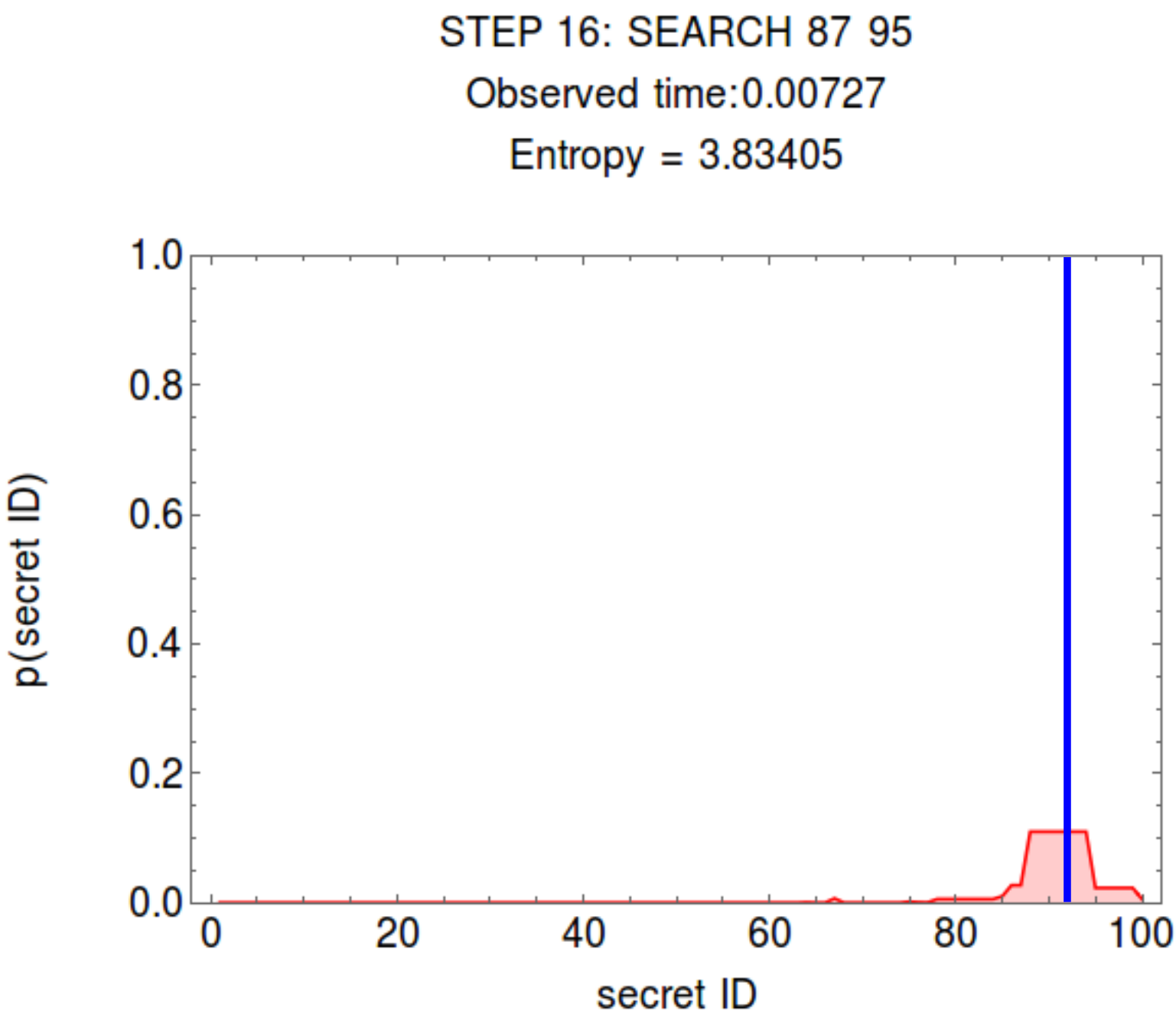
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



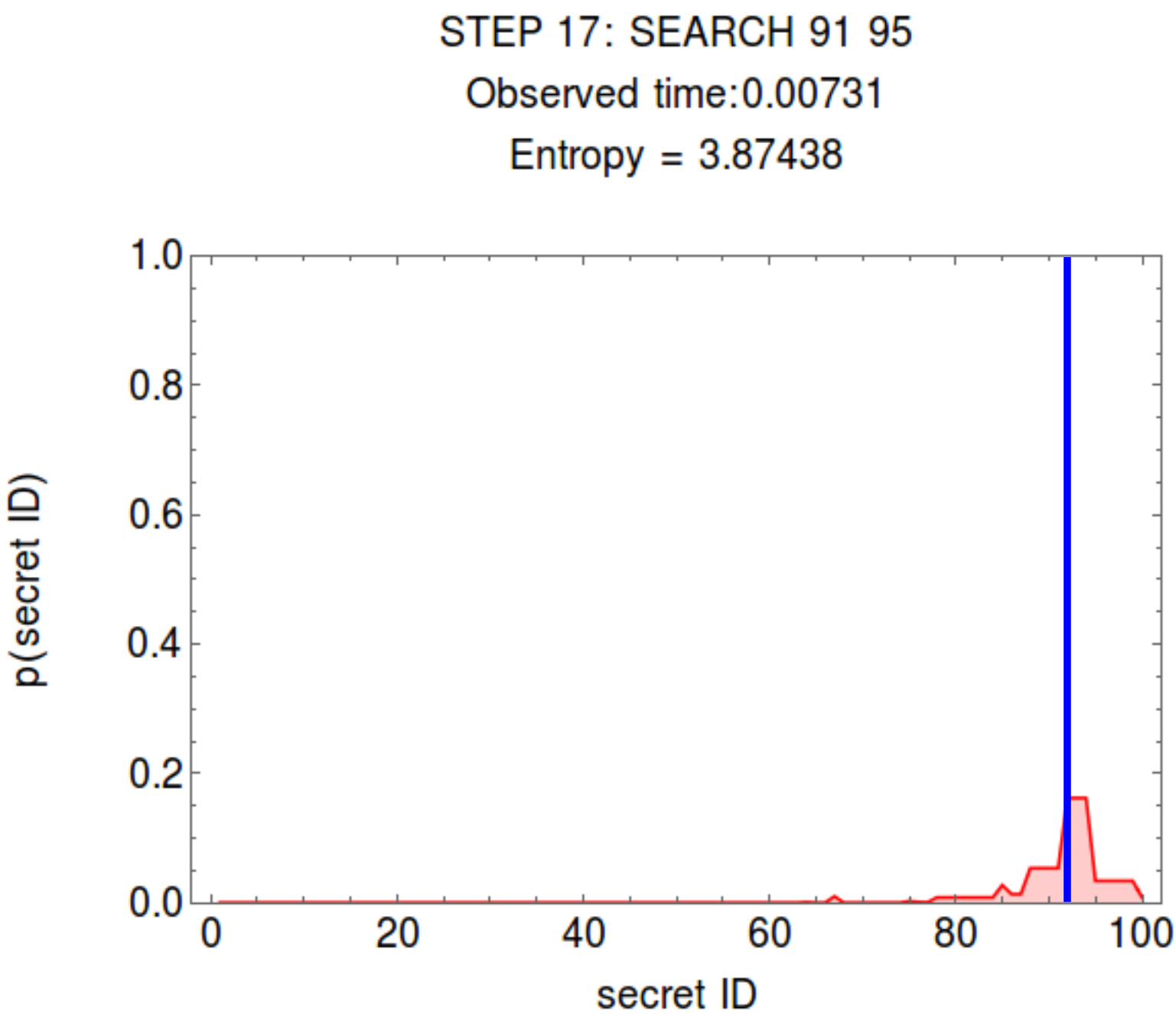
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



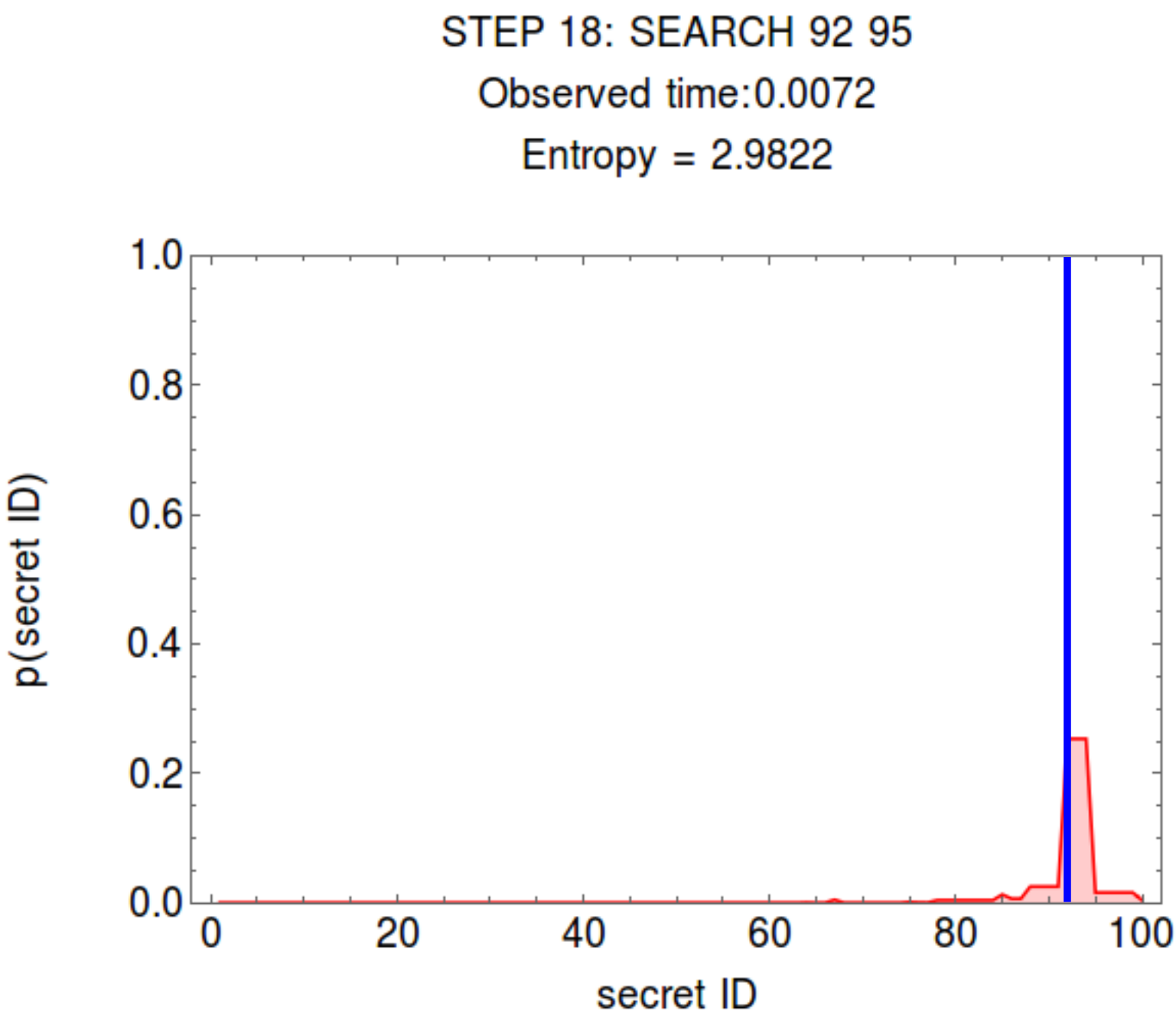
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



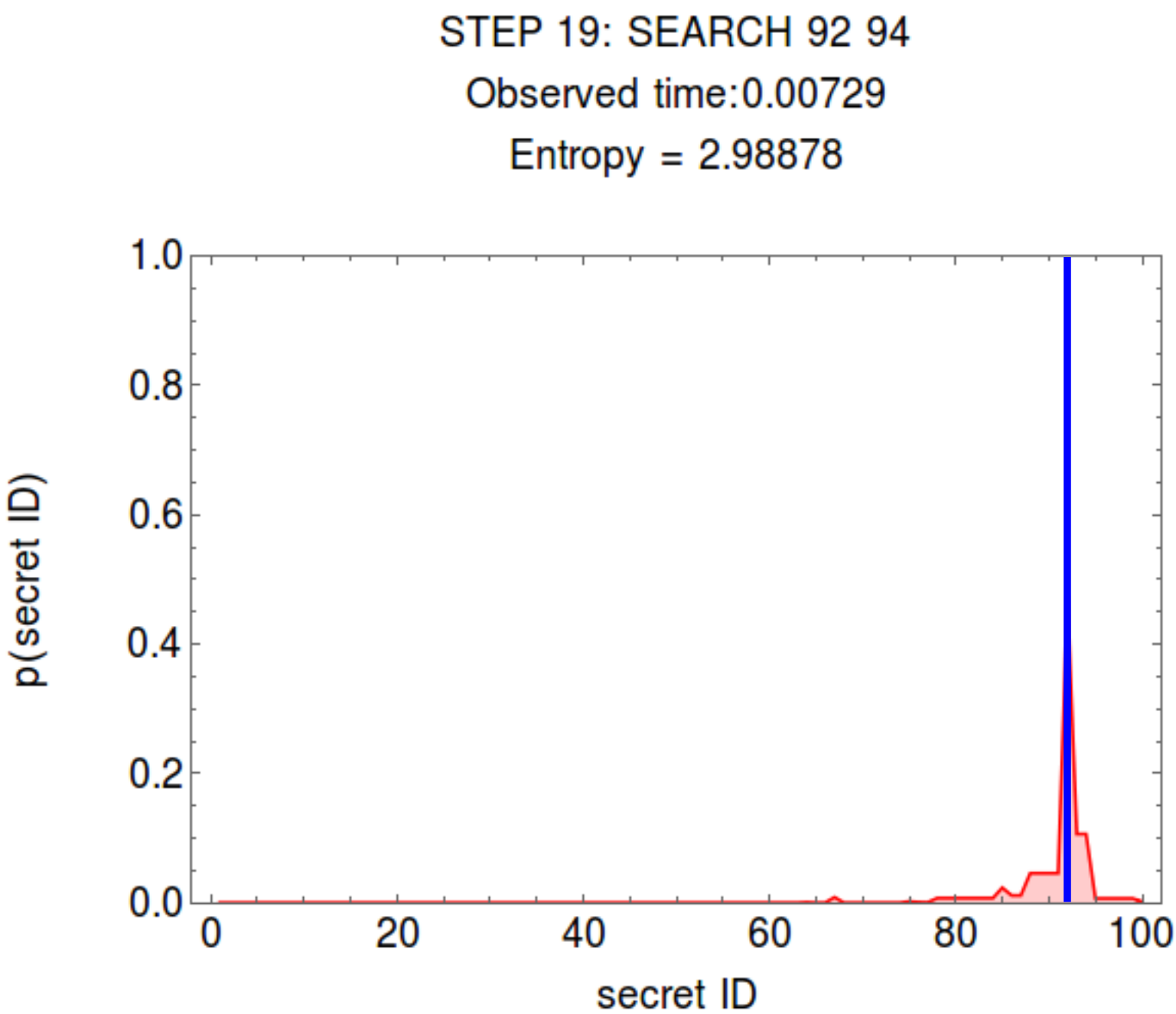
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



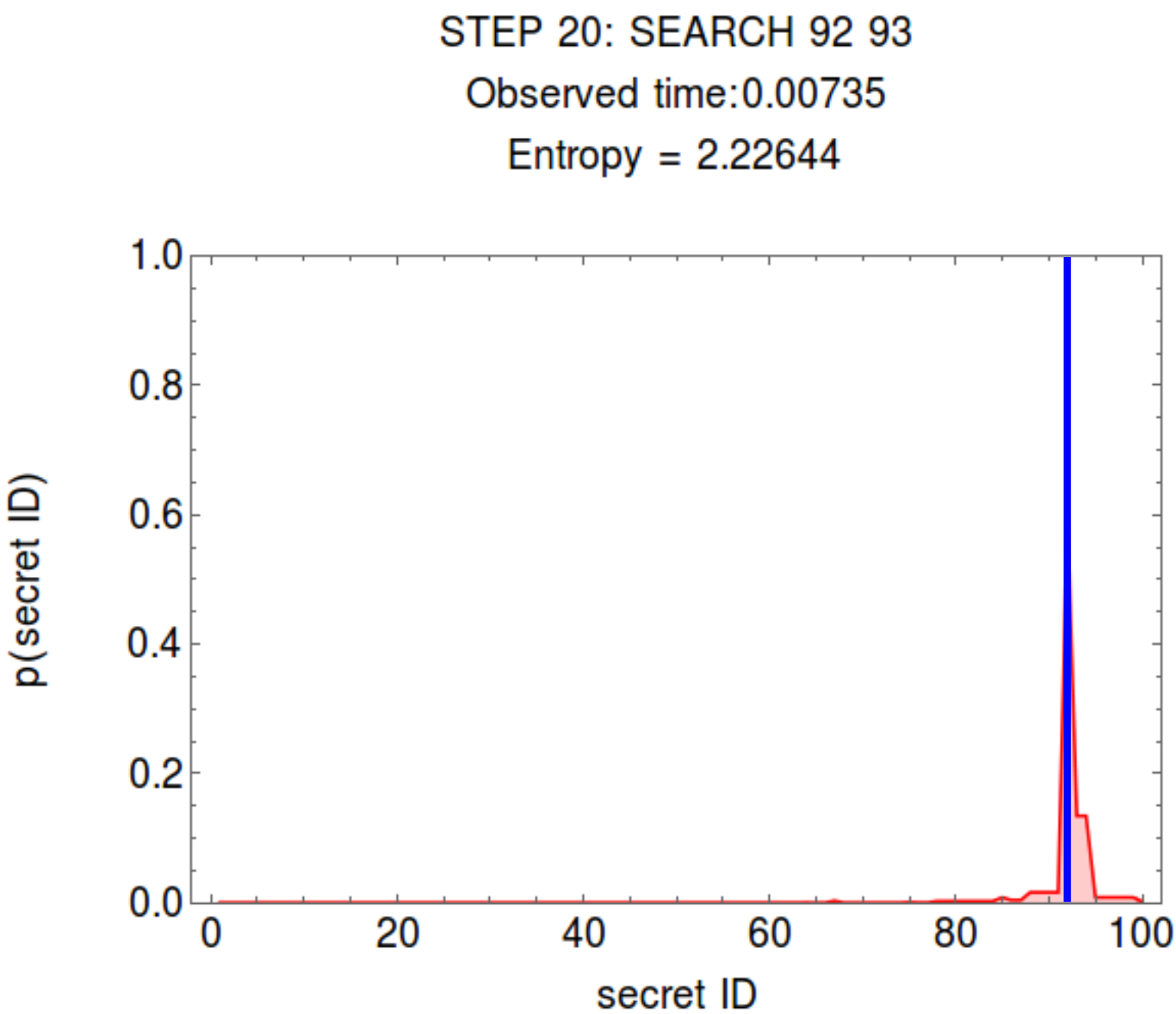
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



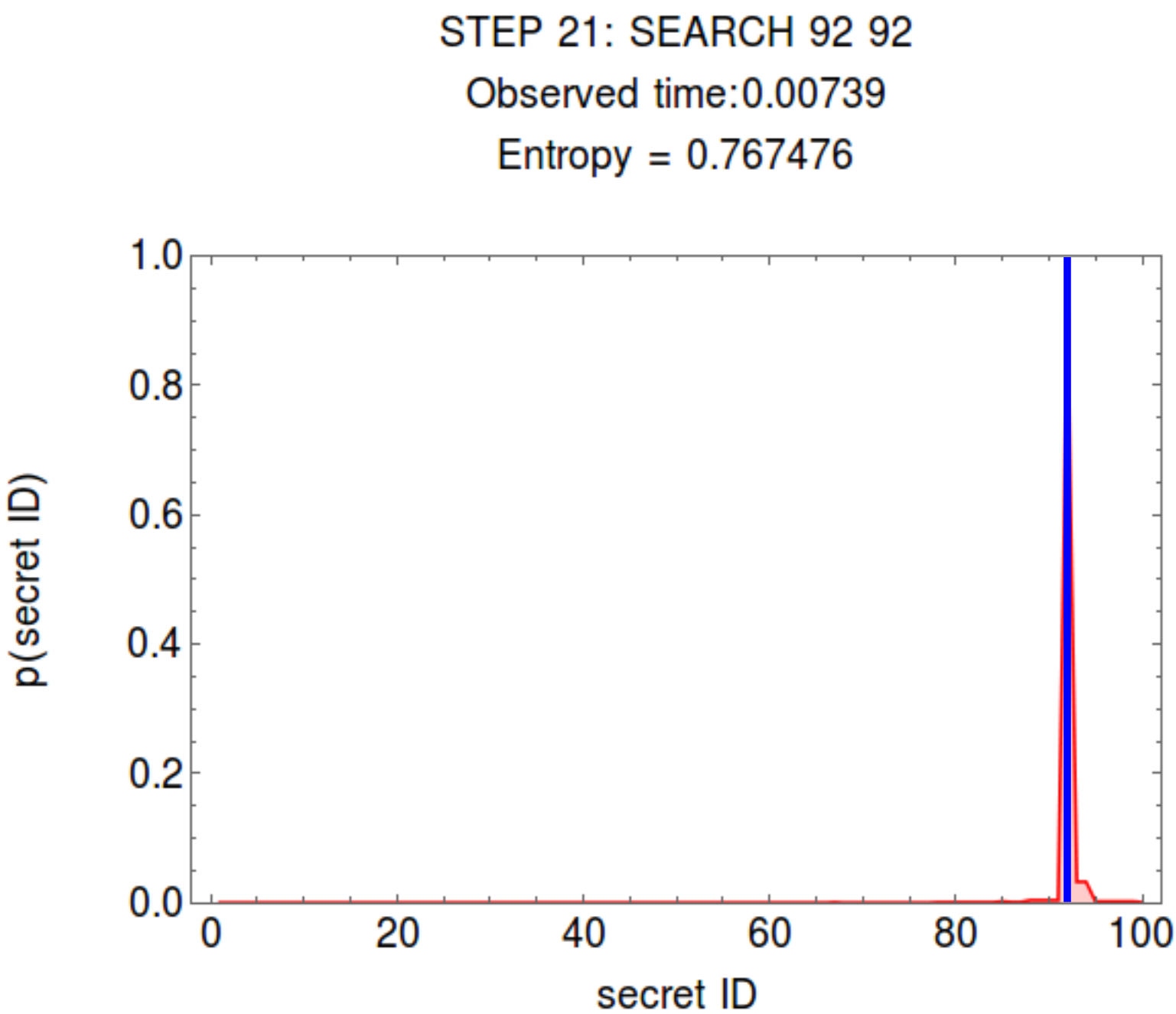
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



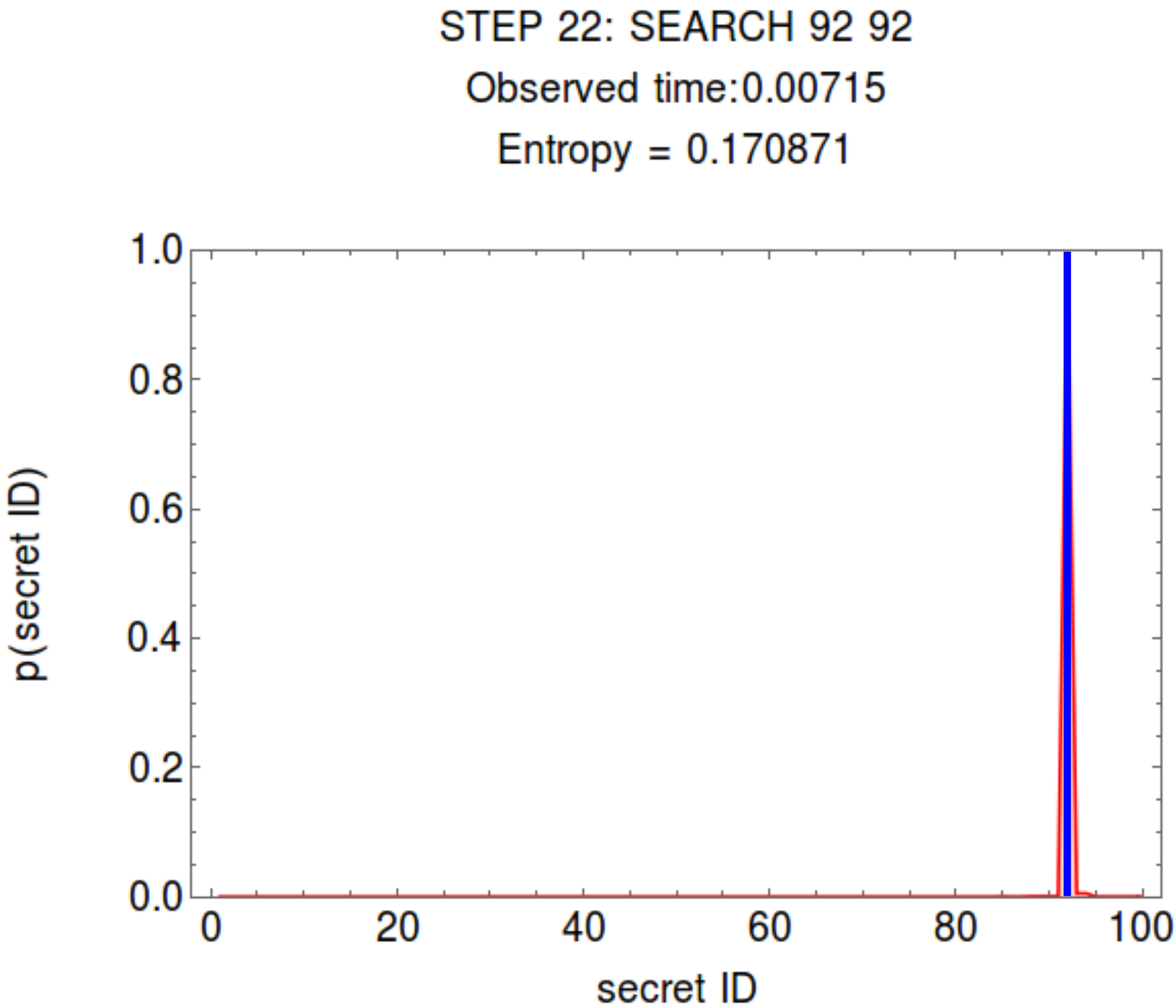
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



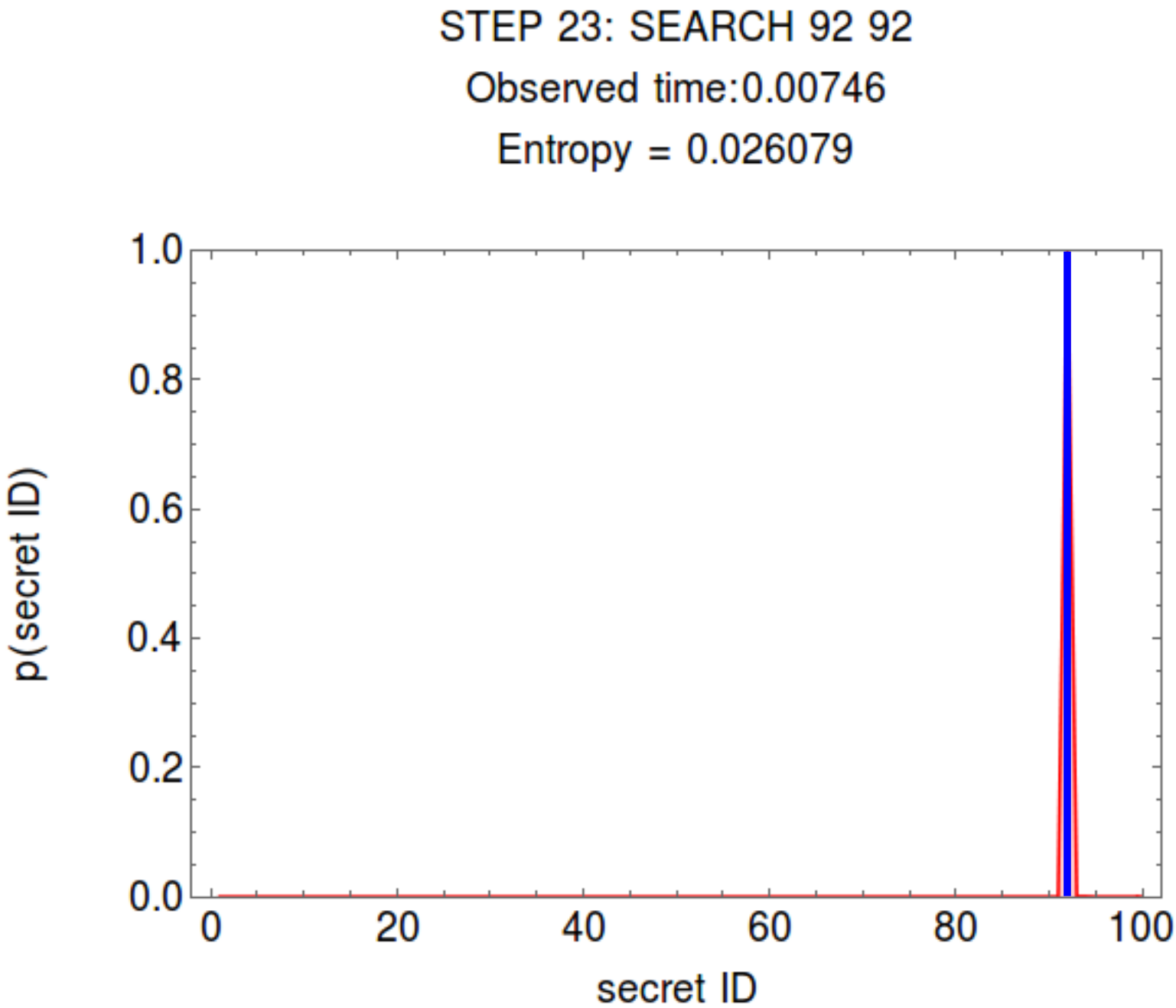
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



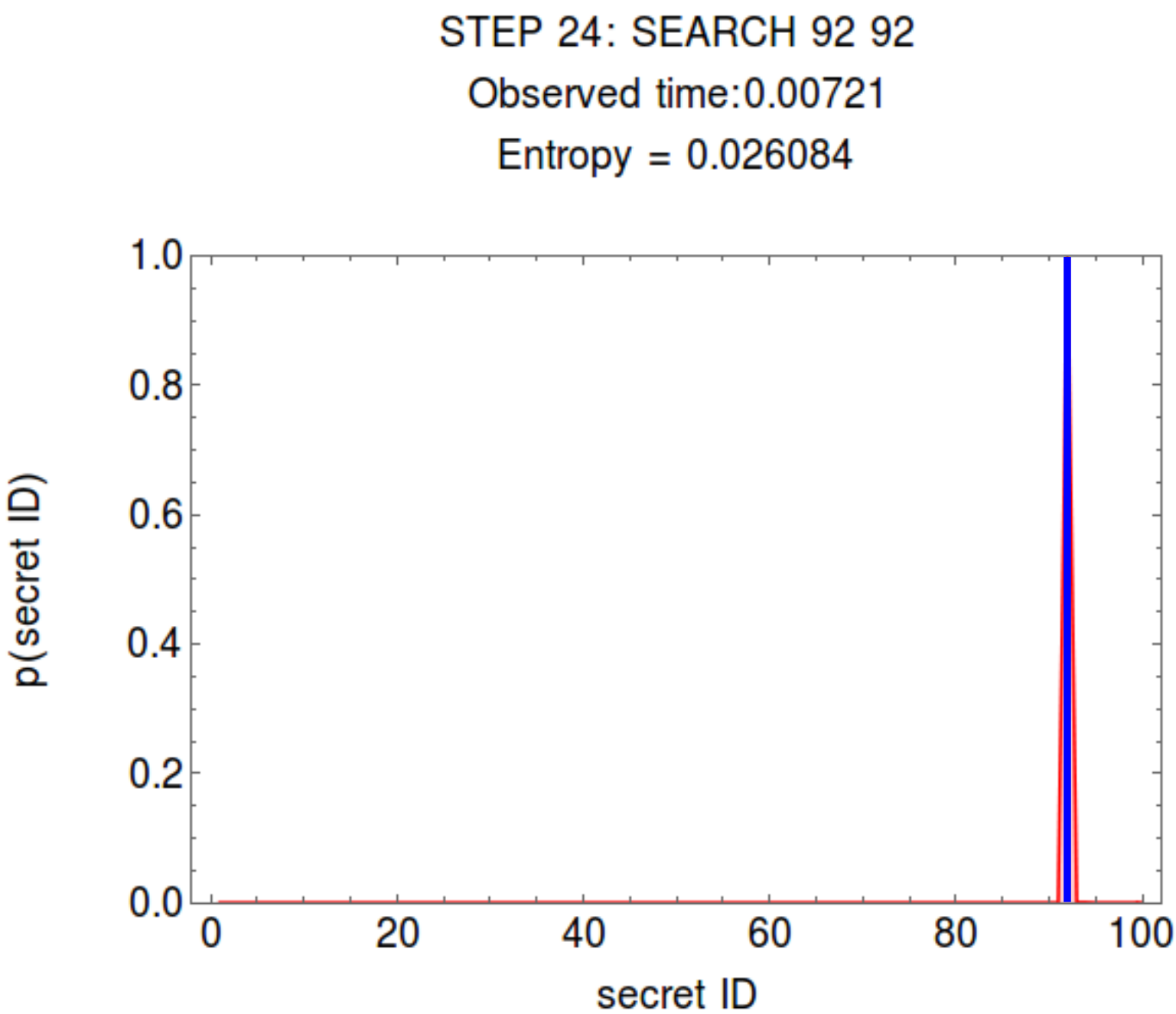
$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



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

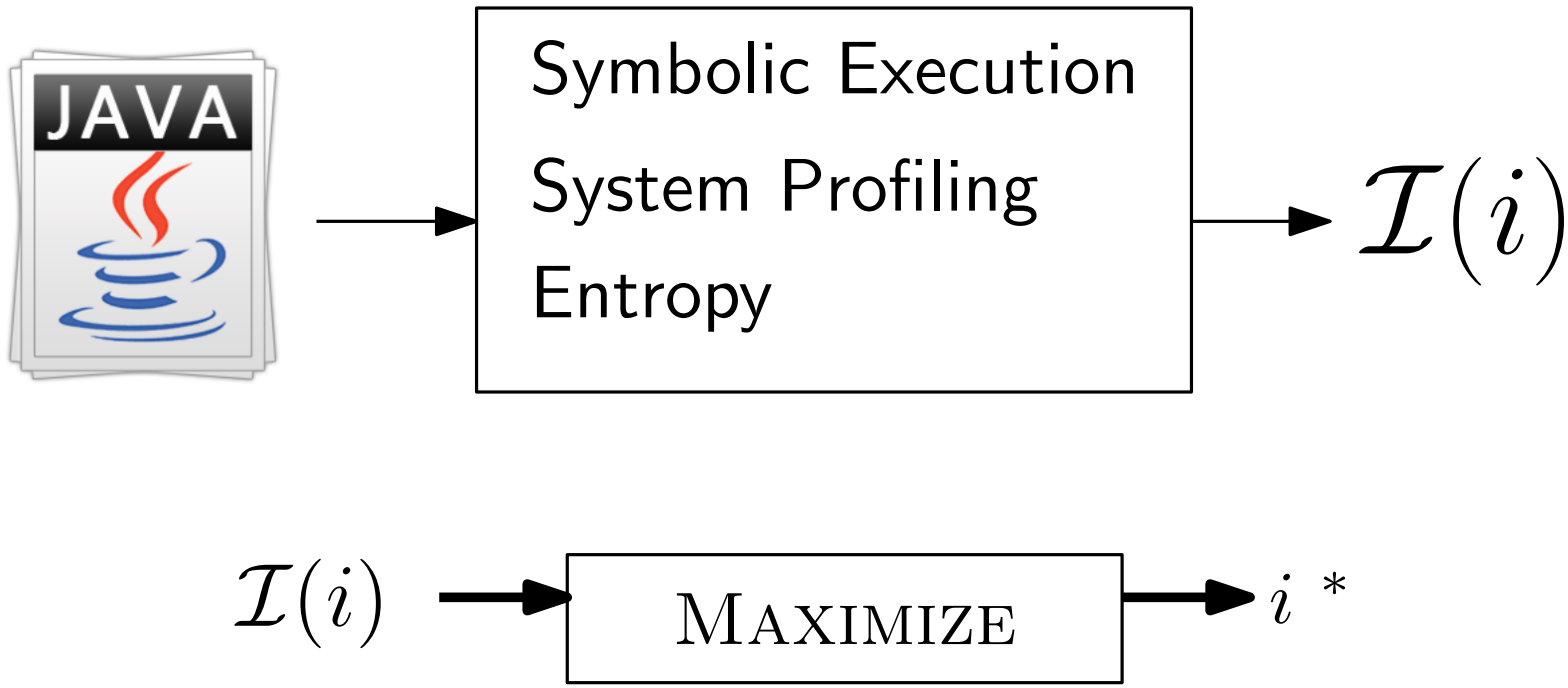


$1 \leq ID \leq 100 \quad ID_1 = 64 \quad ID_2 = 85 \quad ID_{res} = 92$



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



Automatically synthesize side-channel attacks!

Thanks!

Questions?