





3 June 2026 | 1er Congrès des Sciences du Logiciel | Invitation GDR IFM | Lille, France

Detecting Timing Leaks of Programs

using Parametric Timed Model Checking

Dylan Marinho

Sorbonne Université, CNRS UMR 7606, LIP6

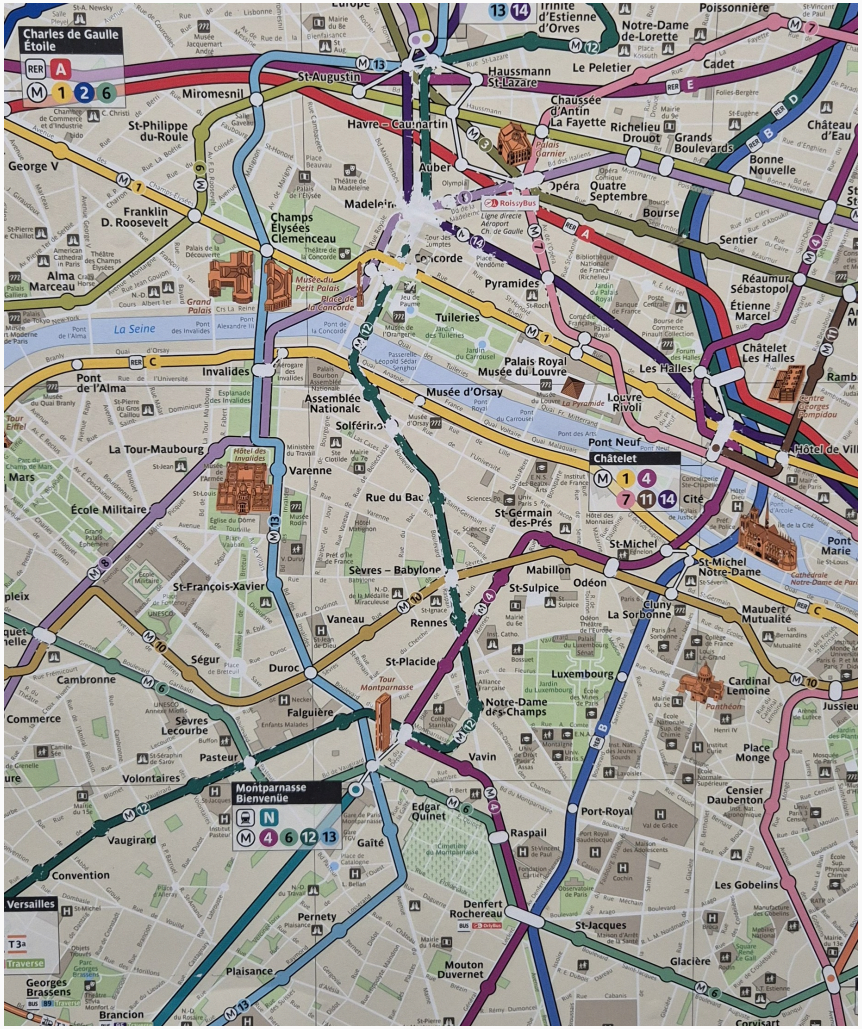
Based on joint work with Étienne André, Jean-Luc Béchennec, Sudipta Chattopadhyay, Sébastien Faucou, Didier Lime, Olivier H. Roux, Jun Sun





CC BY NC SA

To warm up: where was this picture taken?



Revealing hidden information

Real-world example



Number of pizzas (and order time) ordered by the White House prior to major war announcements[1]

[1] home.xnet.com/~warinner/pizzacites.html (1990s)

Revealing hidden information

Real-wo



Nu
wa
major

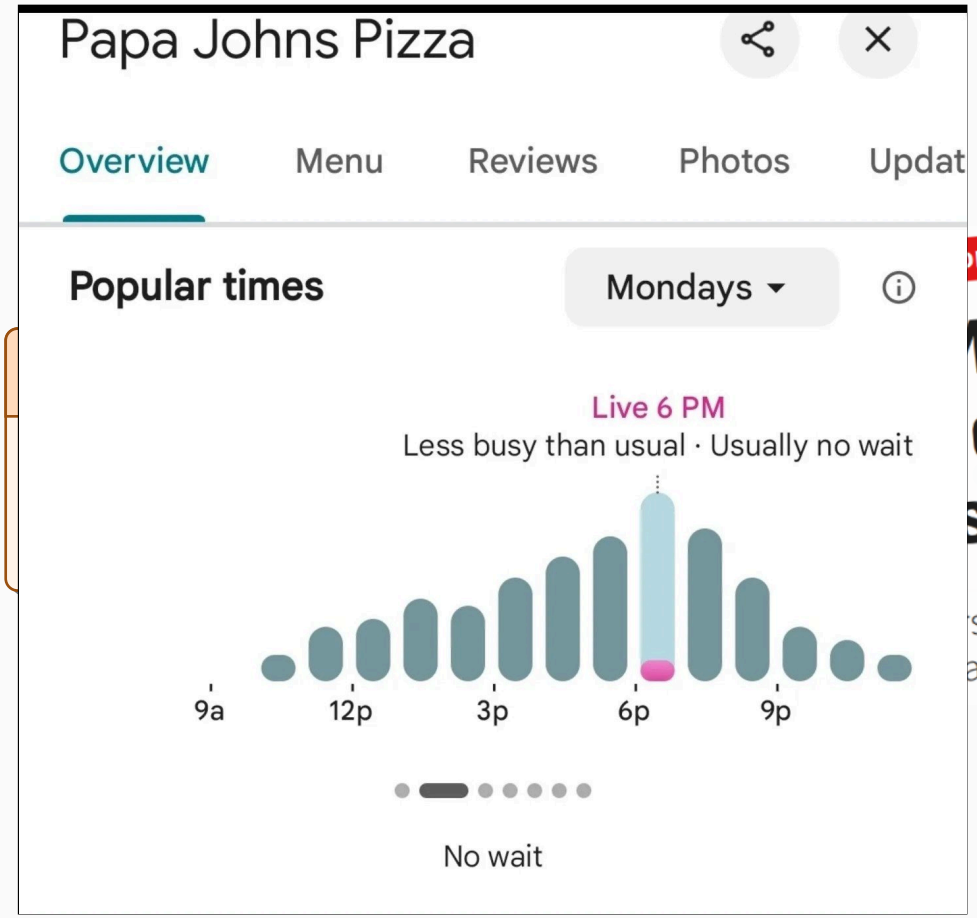
MIDDLE EAST

What is the Pizza Meter? The signal that spiked on Saturday during Iran's attack on Israel

On April 13 there was a surge in pizza orders from U.S. government buildings such as the White House, the Pentagon, and the Department of Defense.

[1] home.xnet.com/~warinner/pizzacites.html (1990s)

Revealing hidden information

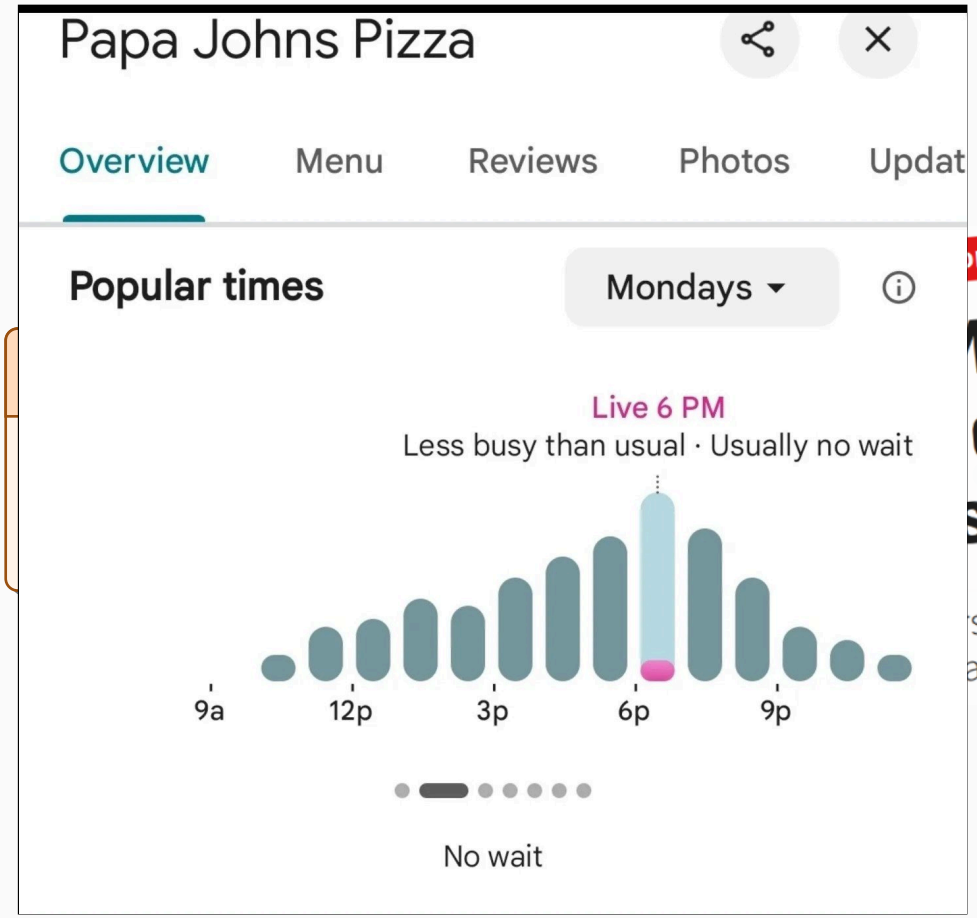


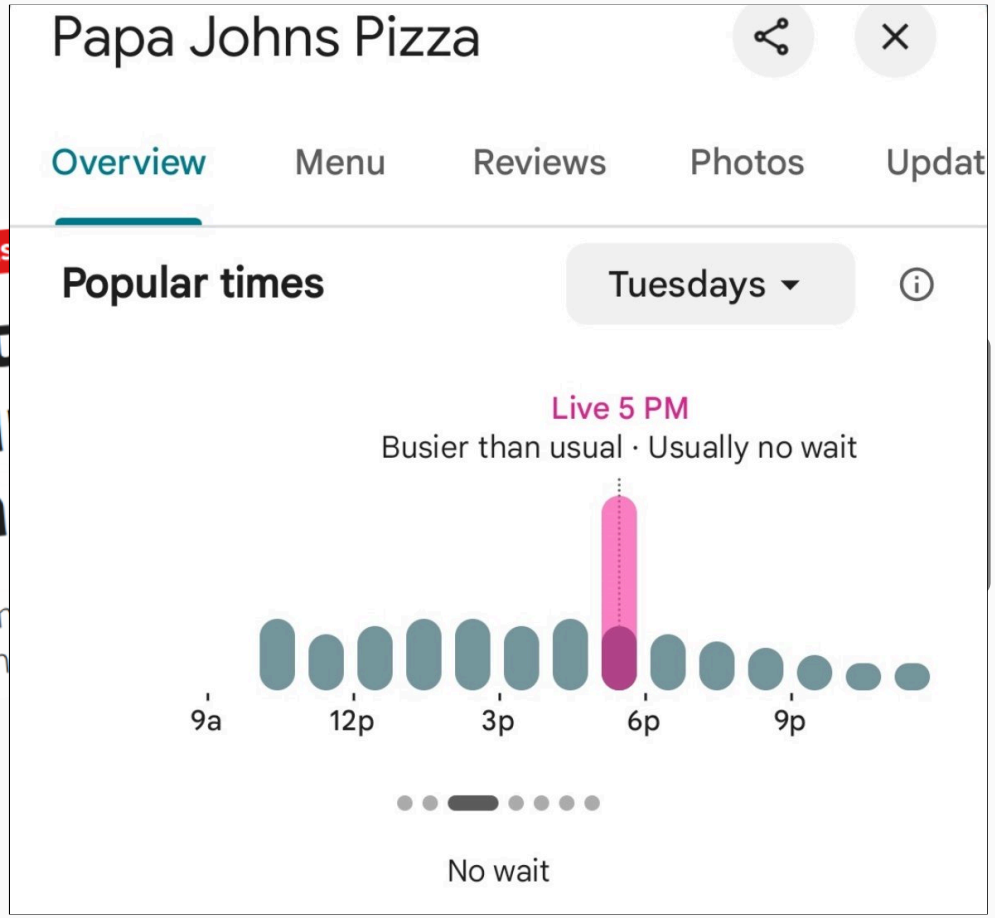


major

[1] home.xnet.com/~warinner/pizzacites.html (1990s)

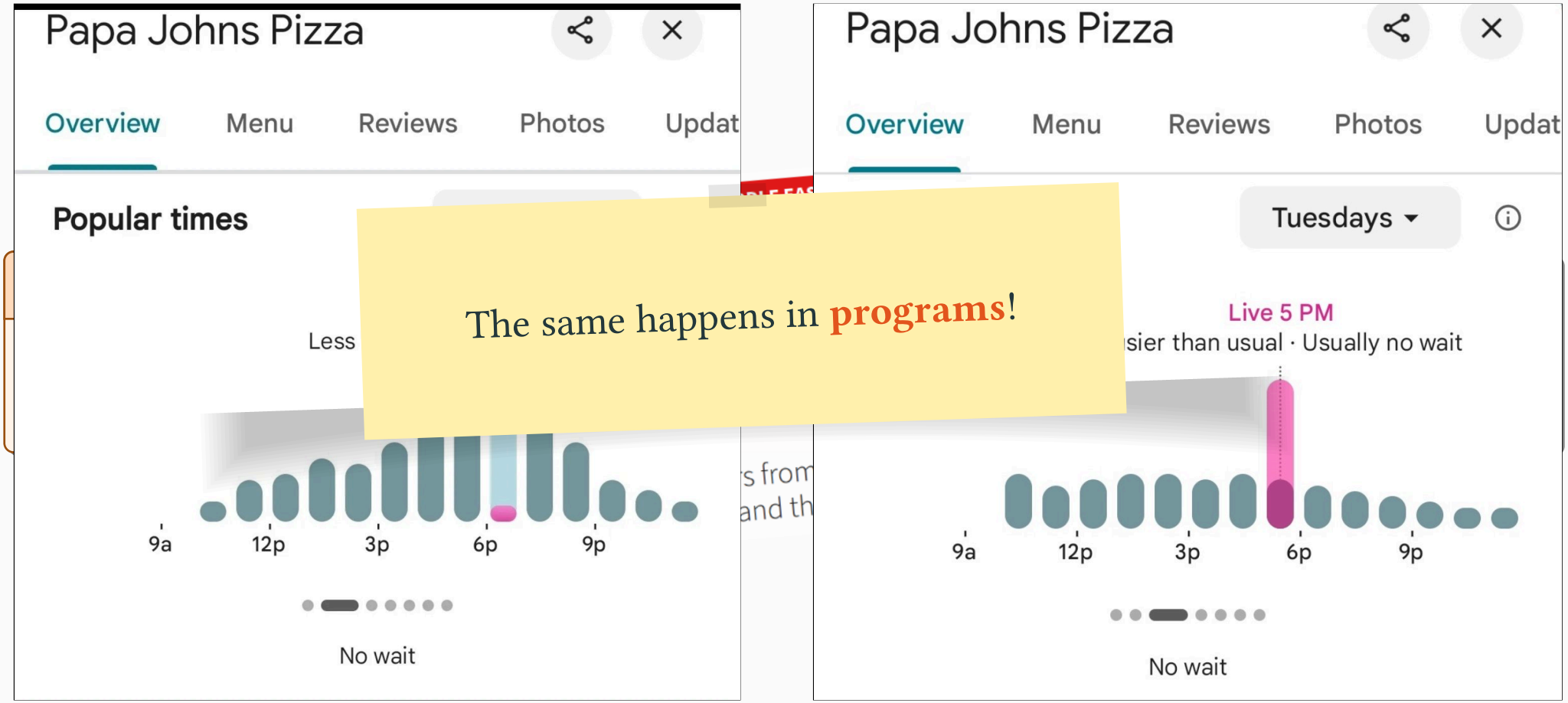
Revealing hidden information





[1] home.xnet.com/~warinner/pizzacites.html (1990s)

Revealing hidden information



The same happens in **programs**!

[1] home.xnet.com/~warinner/pizzacites.html (1990s)

A program that betrays its secrets

```C
// input pwd    : Real password
// input attempt: Tentative password
for (i = 0; i < len(attempt); i++) {
    if(pwd[i] != attempt[i]){
        return false
    }
}
return len(attempt) == len(pwd)
```

A program that betrays its secrets

```c
// input pwd    : Real password
// input attempt: Tentative password
for (i = 0; i < len(attempt); i++) {
    if(pwd[i] != attempt[i]){
        return false
    }
}
return len(attempt) == len(pwd)
```

pwd	G	D	R	S	C	I	L	O	G
attempt	G	D	R	I	F	M			

A program that betrays its secrets

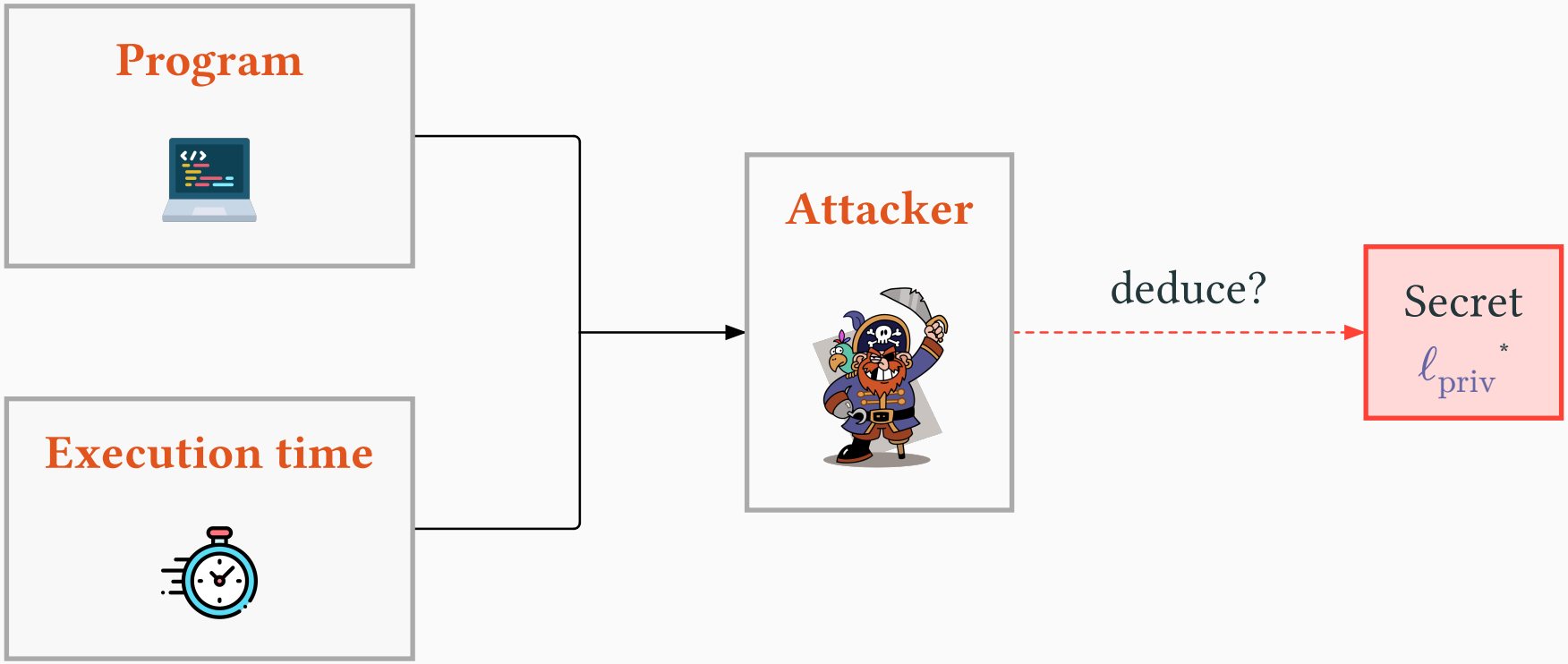
```c
// input pwd    : Real password
// input attempt: Tentative password
for (i = 0; i < len(attempt); i++) {
    if(pwd[i] != attempt[i]){
        return false
    }
}
return len(attempt) == len(pwd)
```

pwd	G	D	R	S	C	I	L	O	G
attempt	G	D	R	I	F	M			

🕓 Execution time (ET): $t \quad t \quad t \quad t \qquad = 4t \qquad \Rightarrow$ 3 correct characters

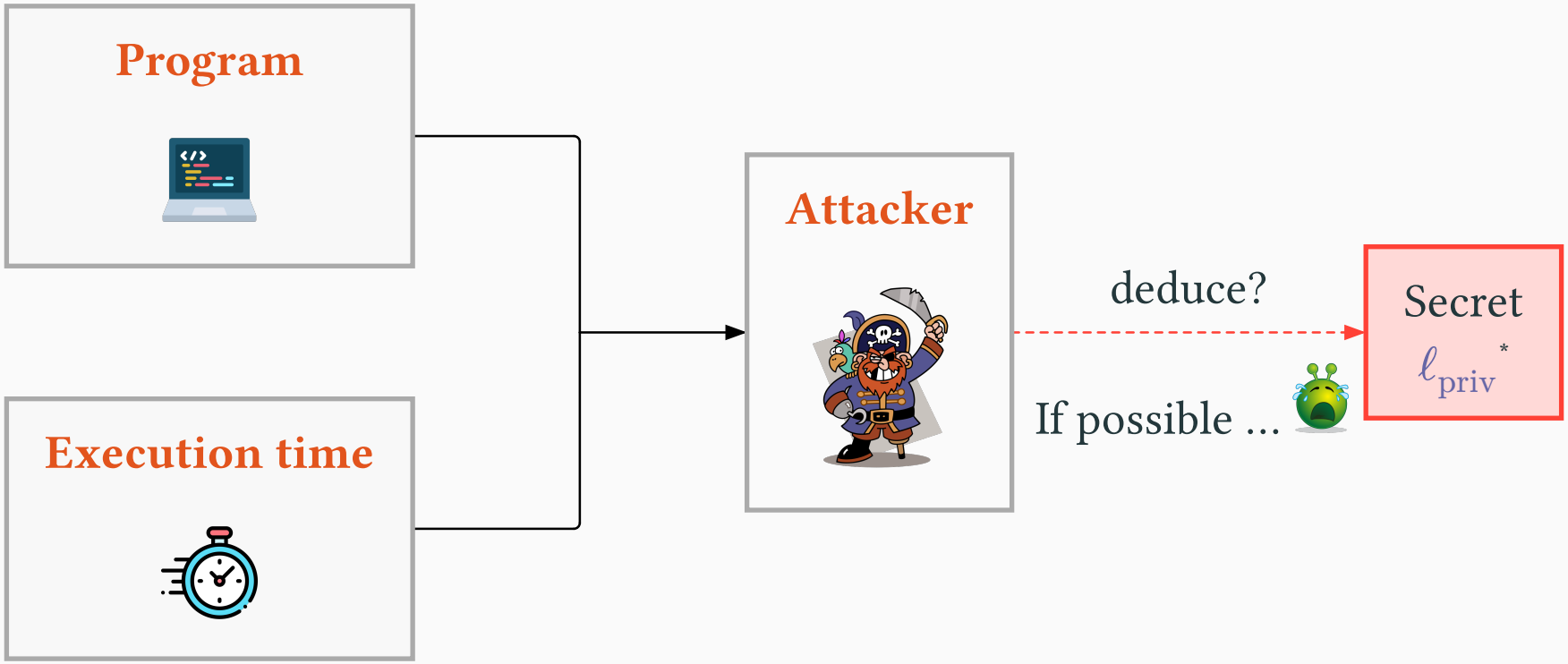
Problem: The ET is proportional to the **number of consecutive correct** characters from the beginning of attempt

The attacker



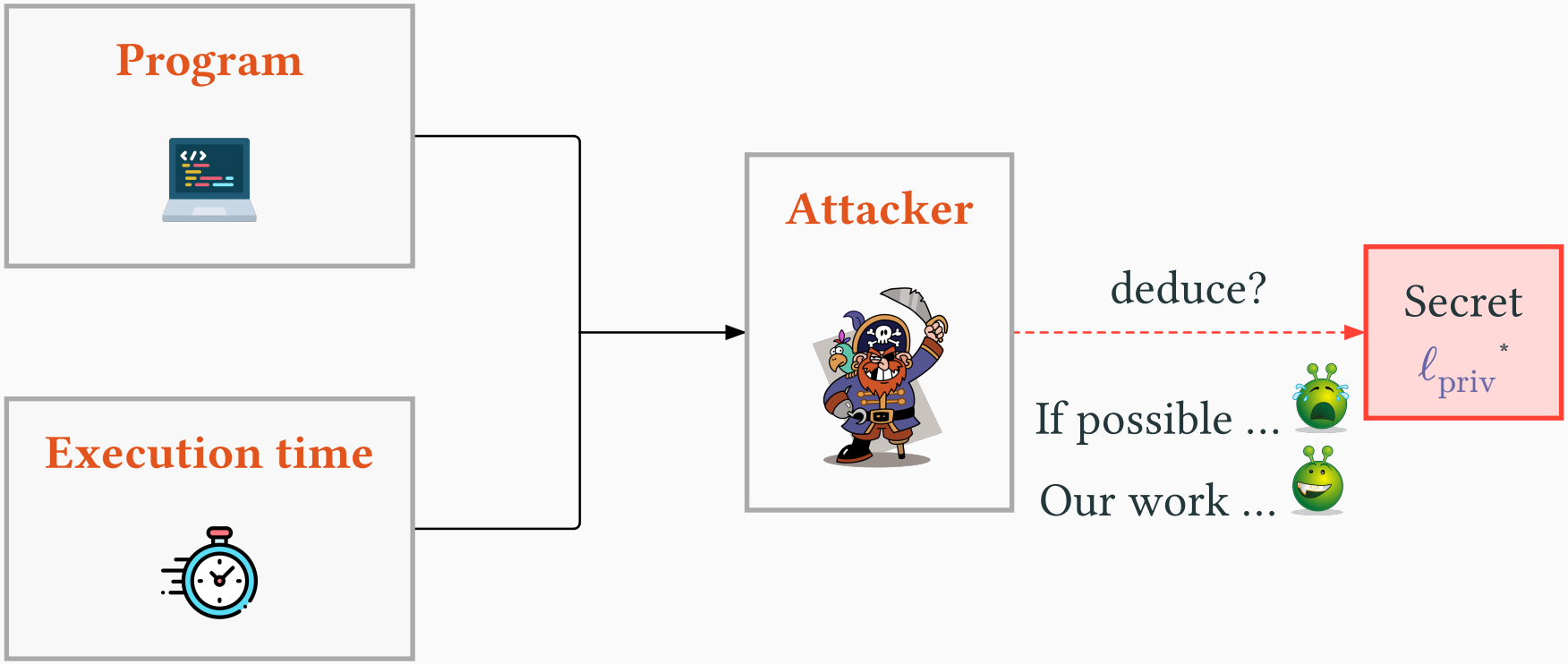
*Deduce whether a **private location** was visited, from the observed timing

The attacker



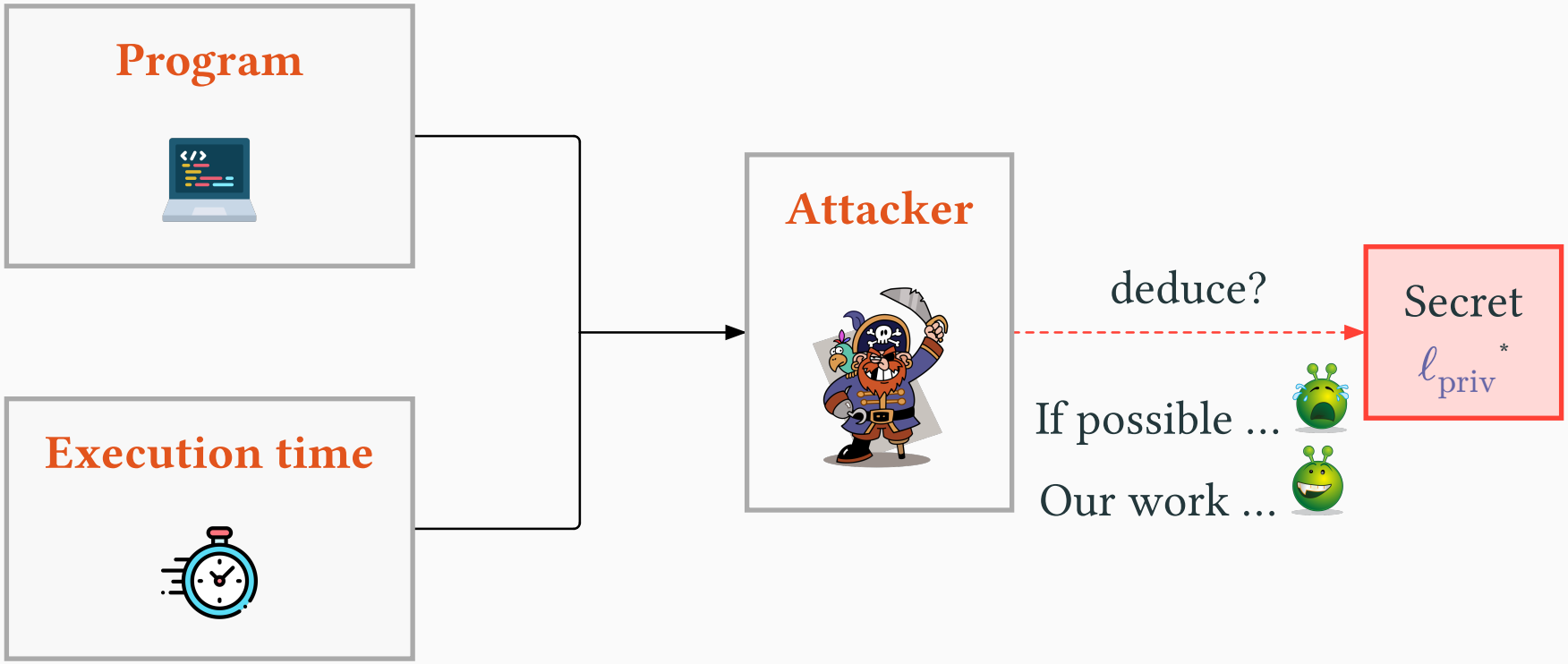
*Deduce whether a **private location** was visited, from the observed timing

The attacker



*Deduce whether a **private location** was visited, from the observed timing

The attacker



⇝ We want to provide **formal guarantees** of security

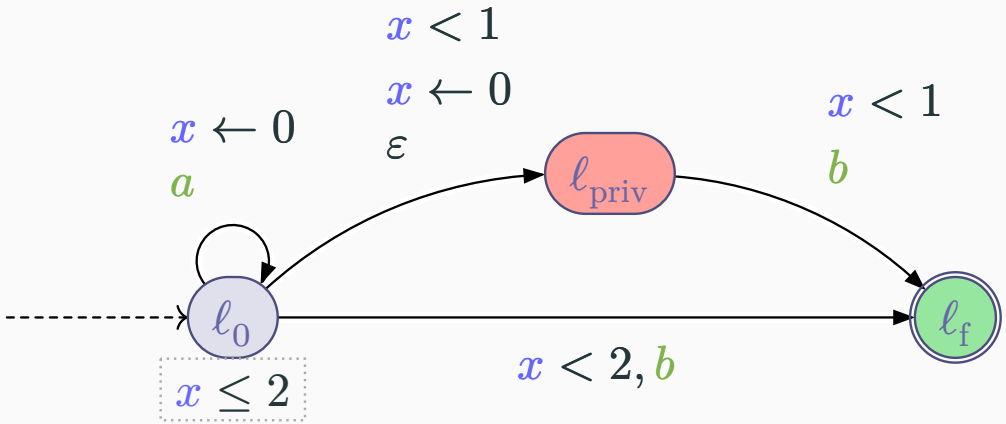
⟹ We use formal methods and **model checking**

---

*Deduce whether a **private location** was visited, from the observed timing

How can we prove a program is secure?

Can the attacker tell what happened?

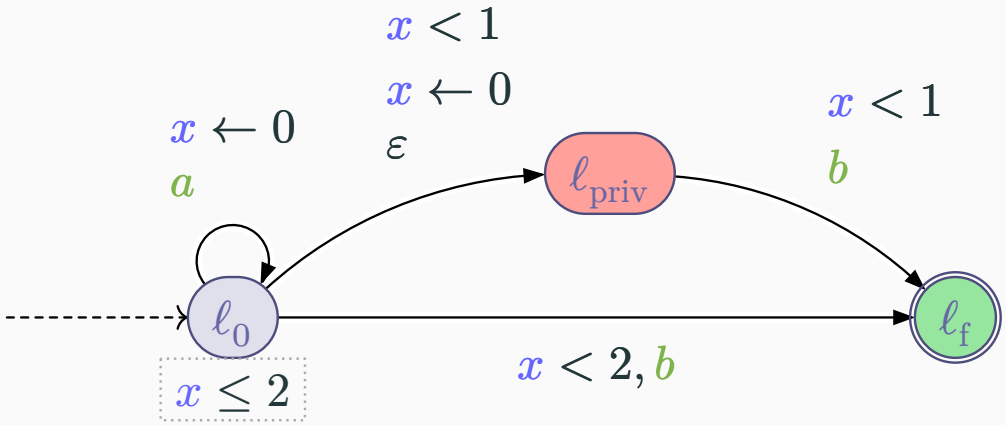


Observed trace

$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

Can the attacker tell what happened?



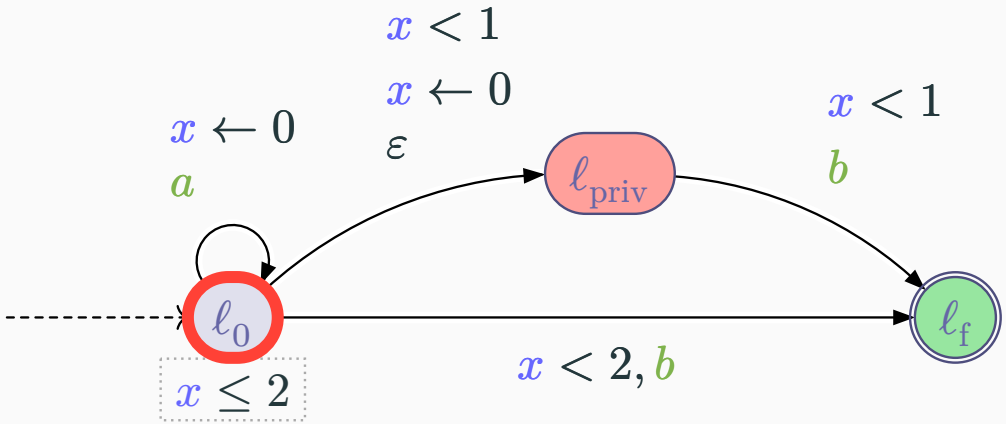
Observed trace

$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.

Can the attacker tell what happened?



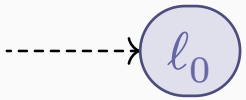


Observed trace

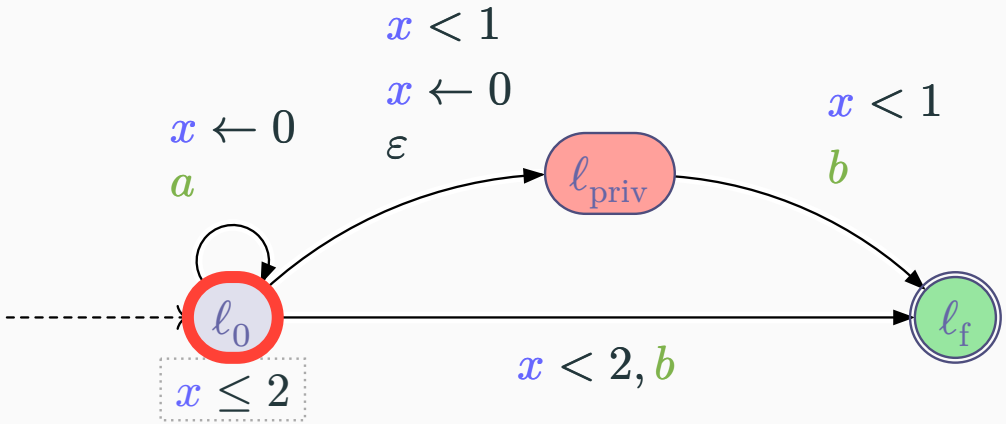
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



Can the attacker tell what happened?



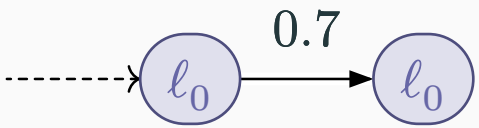


Observed trace

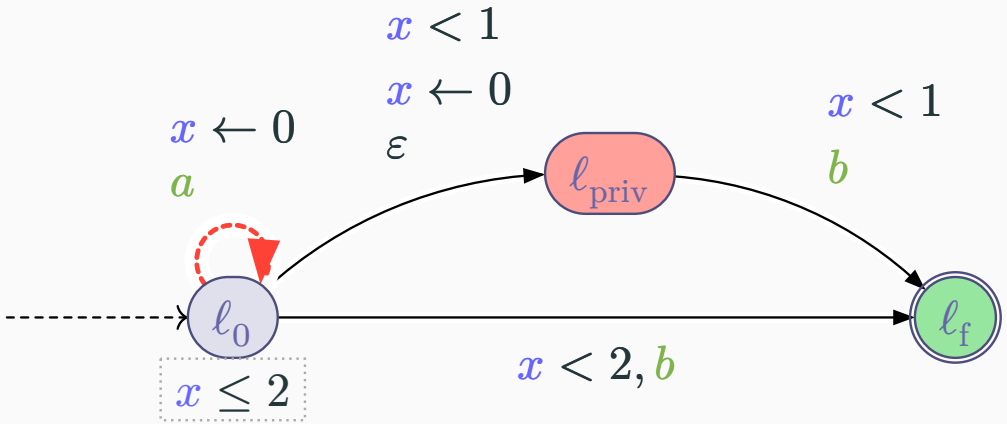
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



Can the attacker tell what happened?



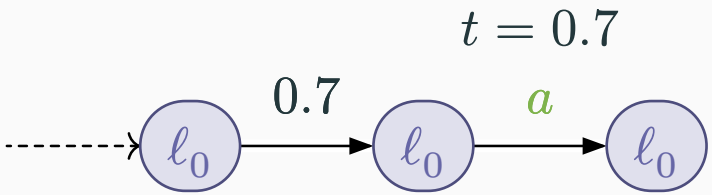


Observed trace

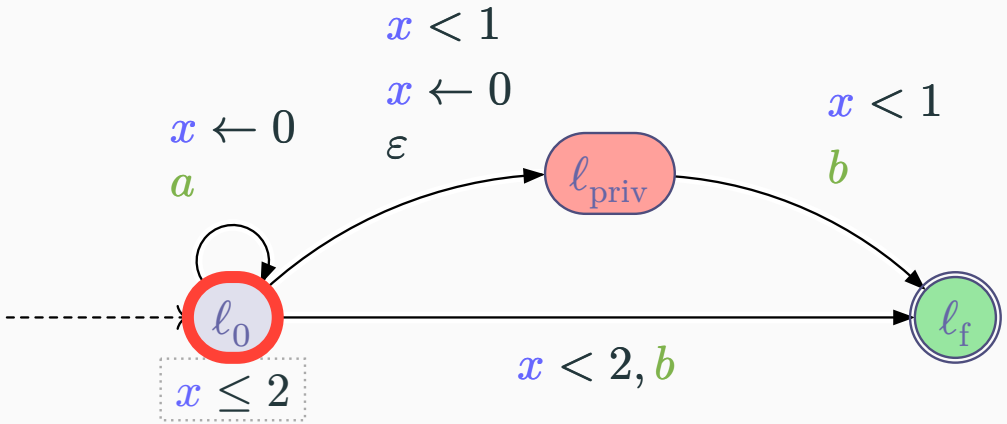
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



Can the attacker tell what happened?



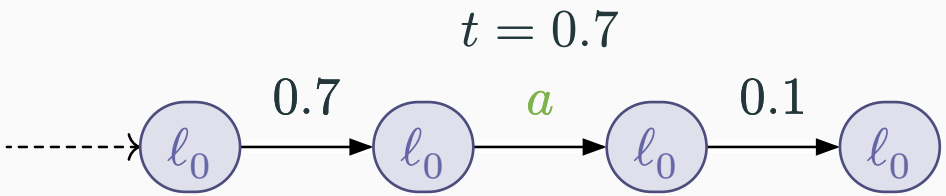


Observed trace

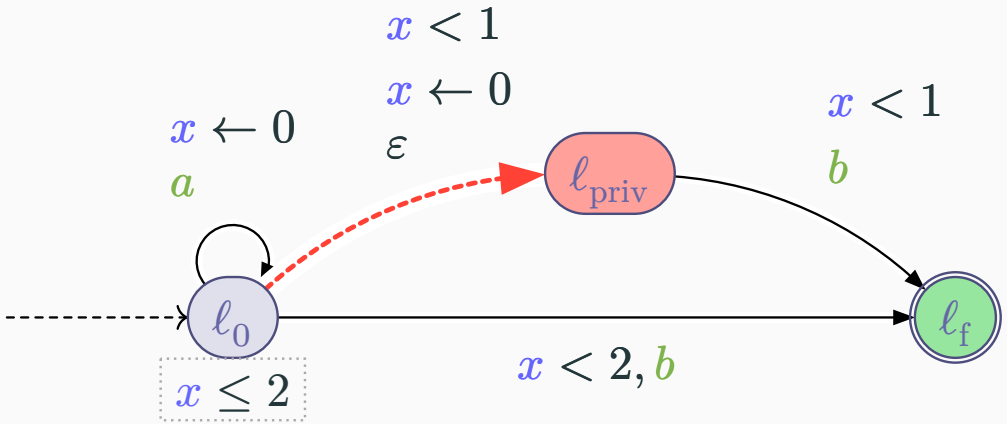
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



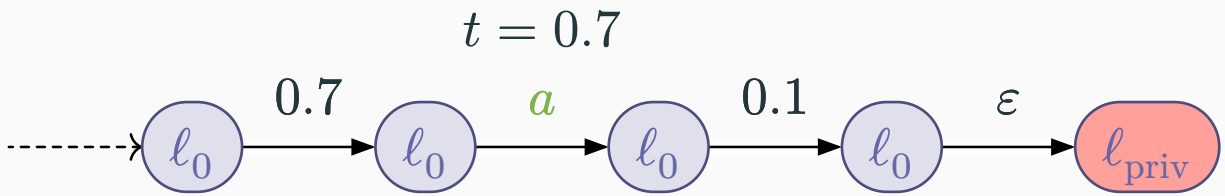
Can the attacker tell what happened?



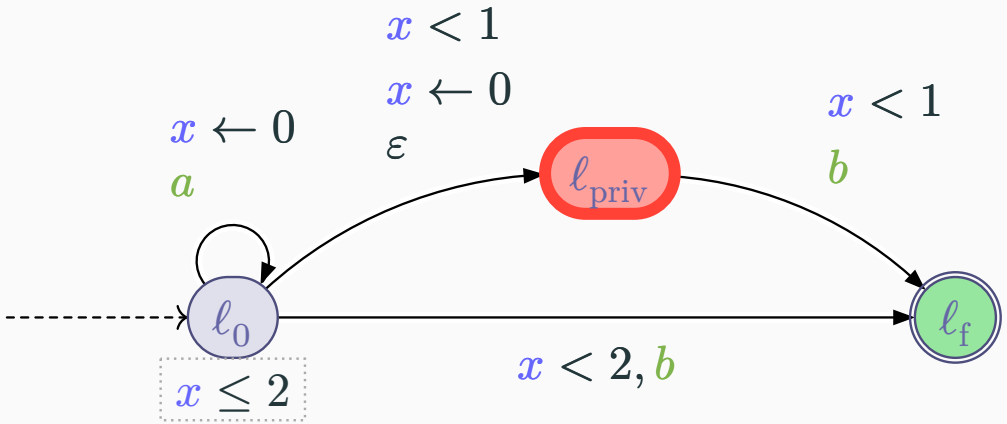
Observed trace
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



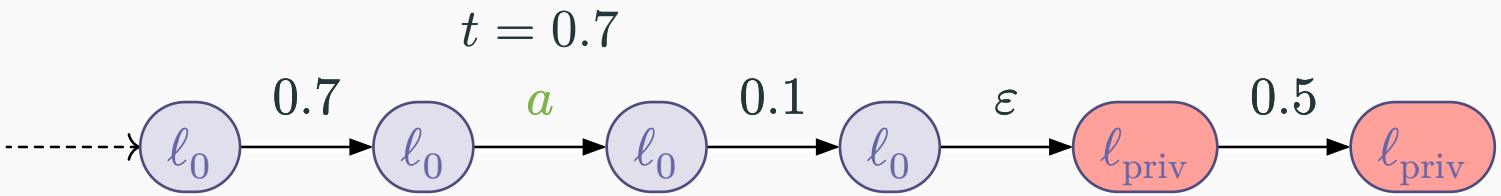
Can the attacker tell what happened?



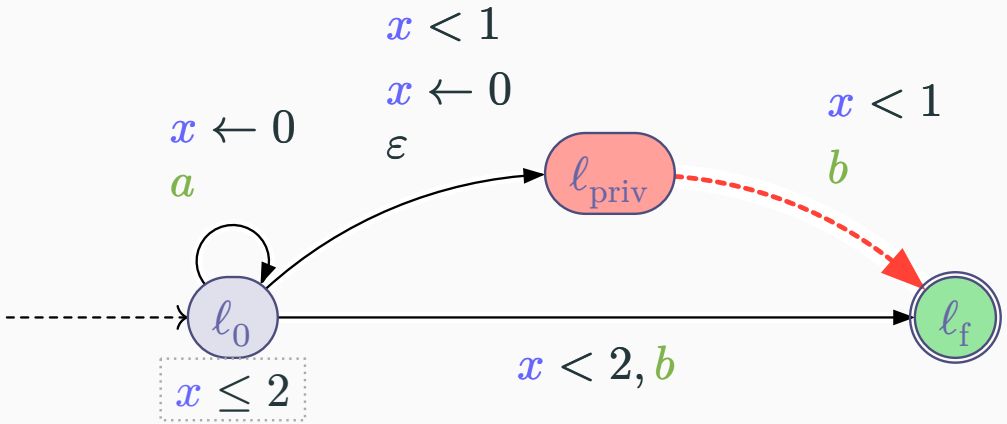
Observed trace
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



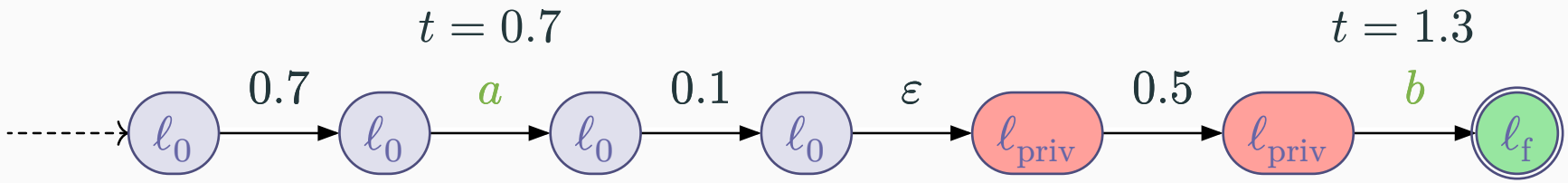
Can the attacker tell what happened?



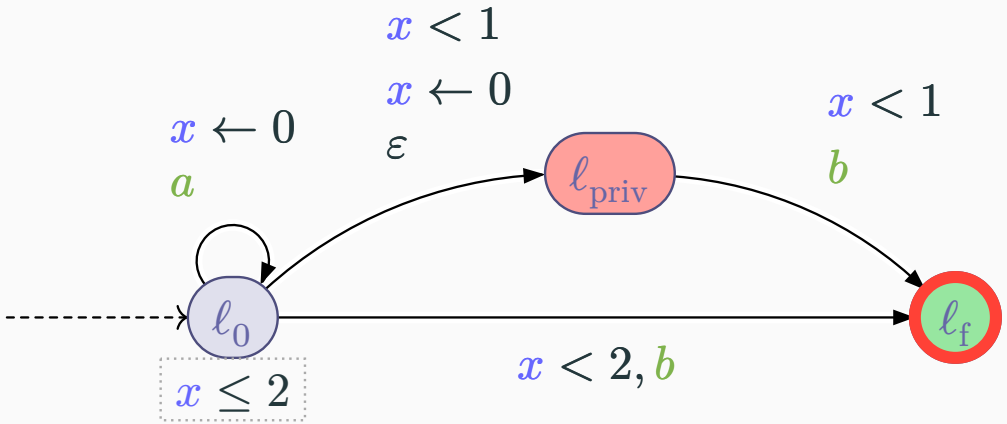
Observed trace
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



Can the attacker tell what happened?

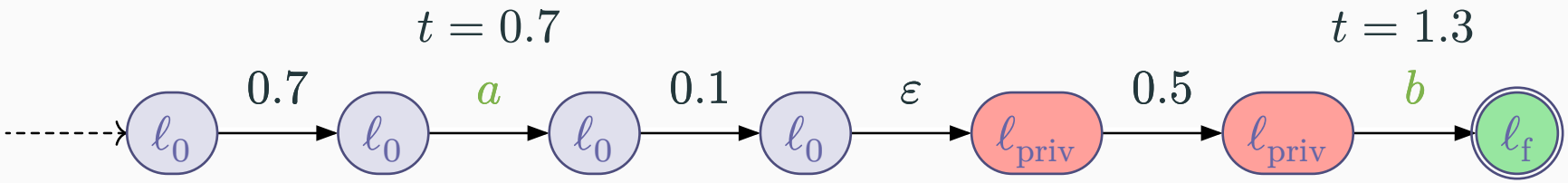




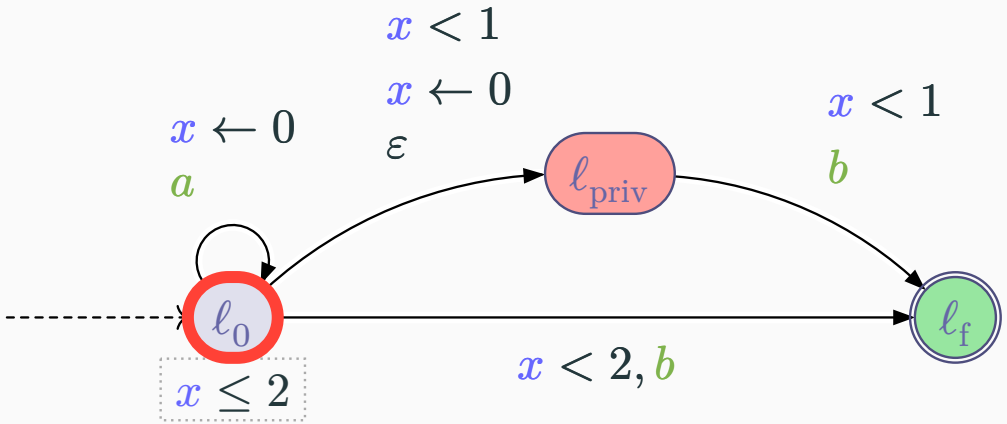
Observed trace
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



Can the attacker tell what happened?



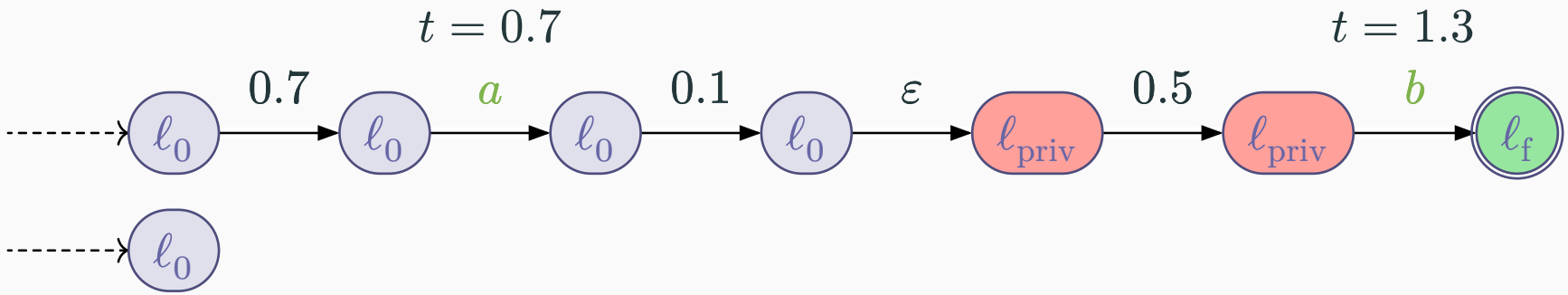


Observed trace

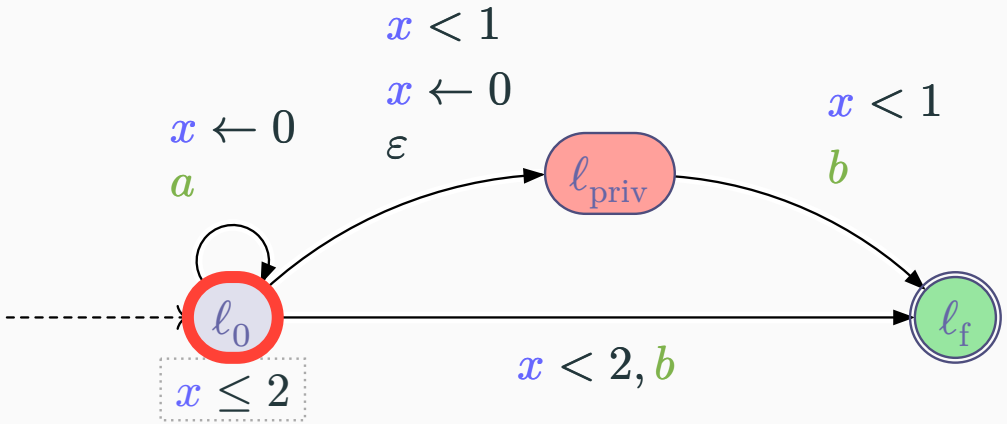
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



Can the attacker tell what happened?



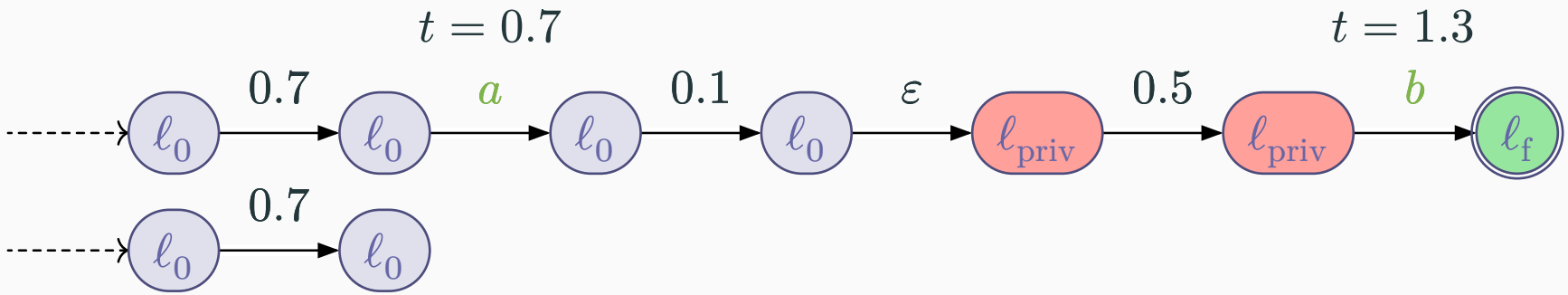


Observed trace

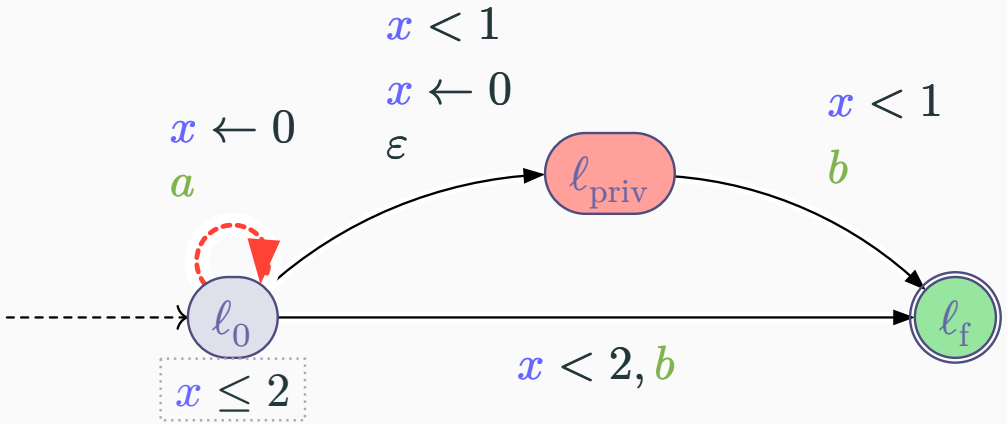
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



Can the attacker tell what happened?

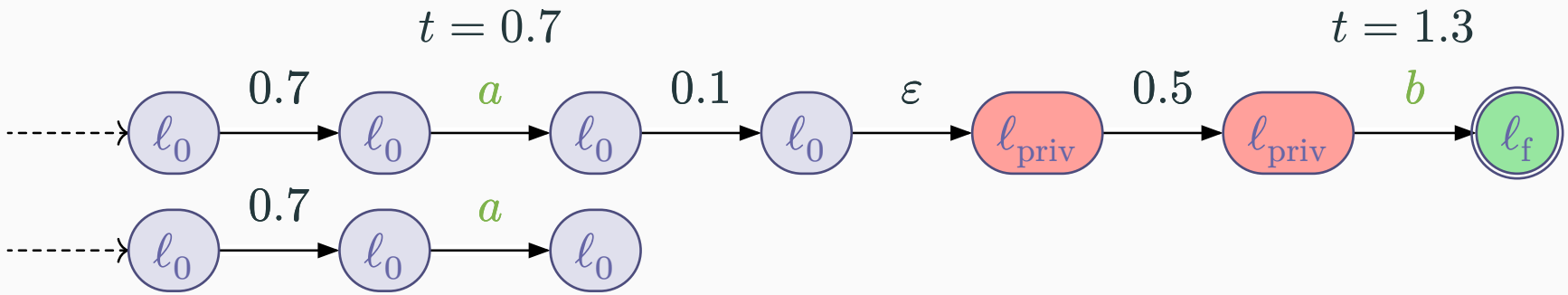




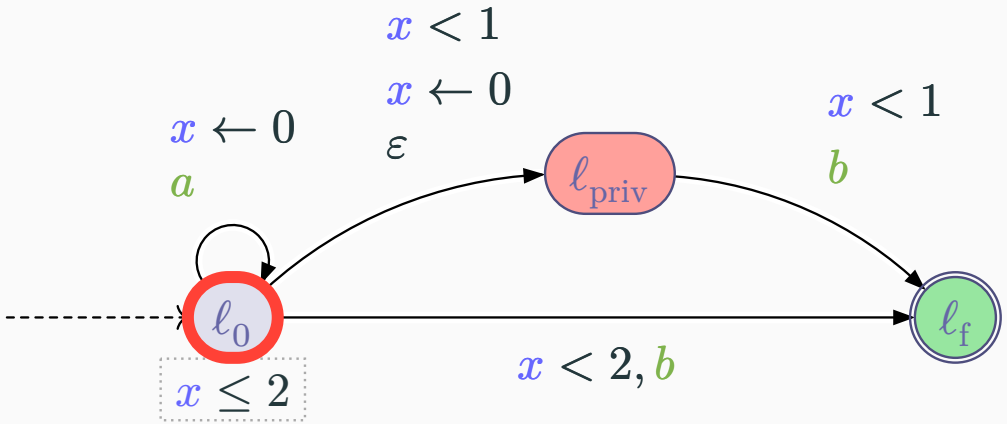
Observed trace
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



Can the attacker tell what happened?

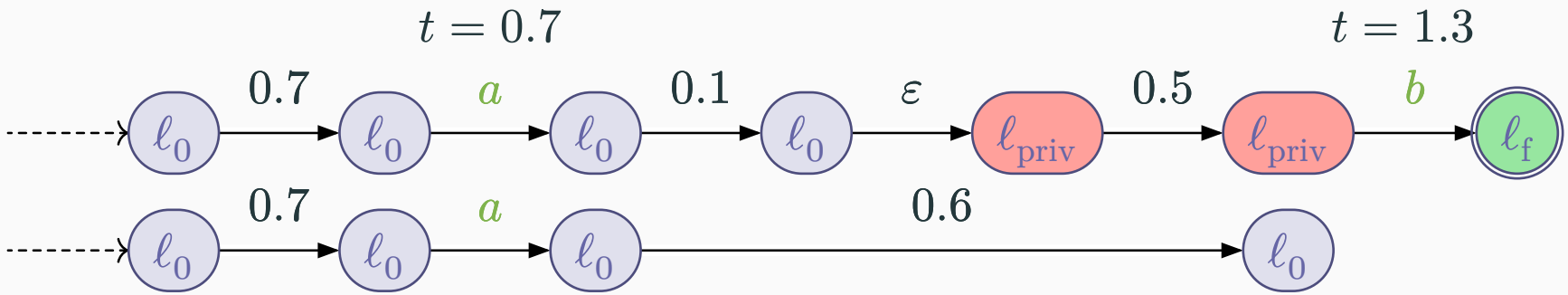




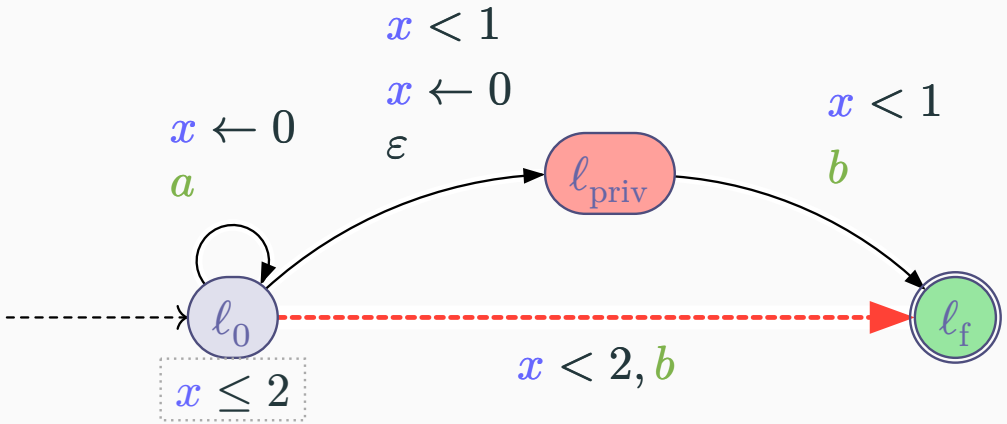
Observed trace
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



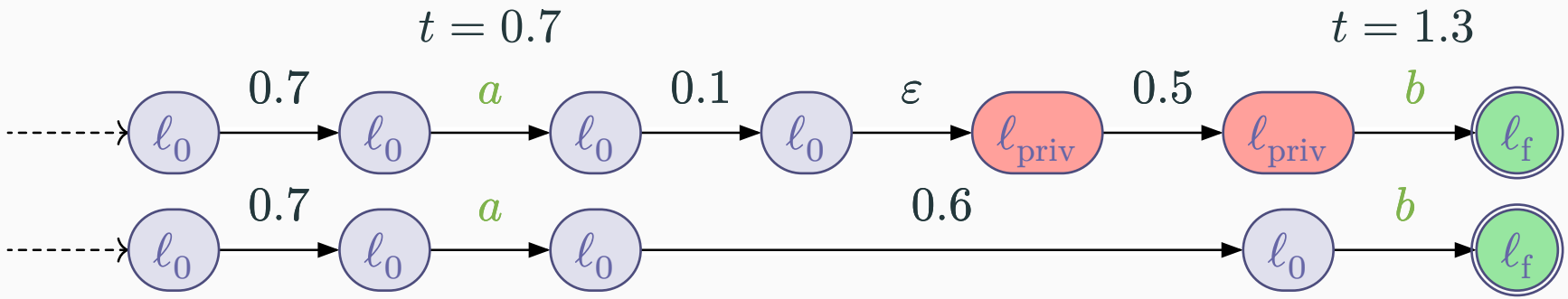
Can the attacker tell what happened?



Observed trace
$(a, 0.7)(b, 1.3)$

Question: Can the attacker infer if ℓ_{priv} has been visited?

No, eg.



The general problem is... impossible to solve

Timed Opacity



A system is **timed opaque** if for every observable trace, the attacker cannot determine whether ℓ_{priv} was visited

[Cas09] Franck Cassez, "The Dark Side of Timed Opacity," in *ISA*, 2009.

The general problem is... impossible to solve

Timed Opacity 

A system is **timed opaque** if for every observable trace, the attacker cannot determine whether ℓ_{priv} was visited

Franck Cassez, The Dark Side of Timed Opacity (2009)

→  **Opacity is undecidable for timed automata!**

[Cas09] Franck Cassez, "The Dark Side of Timed Opacity," in *ISA*, 2009.

The general problem is... impossible to solve

Timed Opacity

A system is **timed op** er cannot determine whether ℓ_{priv} was visi

Do we give up?

of Timed Opacity (2009)

e for timed automata!

[Cas09] Franck Cassez, "The Dark Side of Timed Opacity," in *ISA*, 2009.

The general problem is... impossible to solve

Timed Opacity

A system is **timed op** er cannot determine whether ℓ_{priv} was visi

Do we give up?

No - we refocus on what matters

ie. we change the definition!

of Timed Opacity (2009)

e for timed automata!

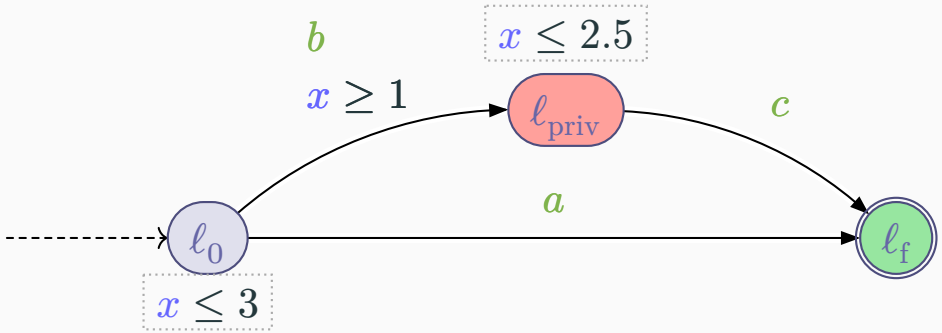
[Cas09] Franck Cassez, "The Dark Side of Timed Opacity," in *ISA*, 2009.

A tractable notion: execution-time opacity

Execution-time Opacity

The attacker only observes the **total execution time** *(not the sequence of actions)*

The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not



🕵️ **Observation:** ⏱ $= 2$

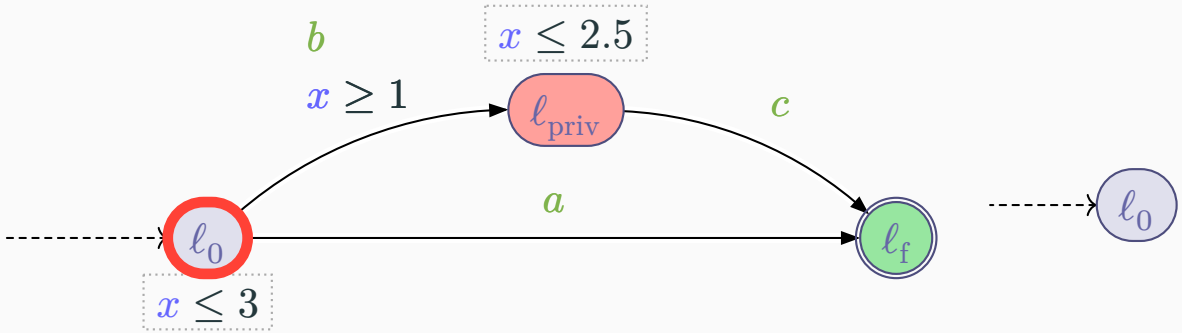
[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

A tractable notion: execution-time opacity

Execution-time Opacity

The attacker only observes the **total execution time** *(not the sequence of actions)*

The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not



🕵️ **Observation:** ⏱ $= 2$

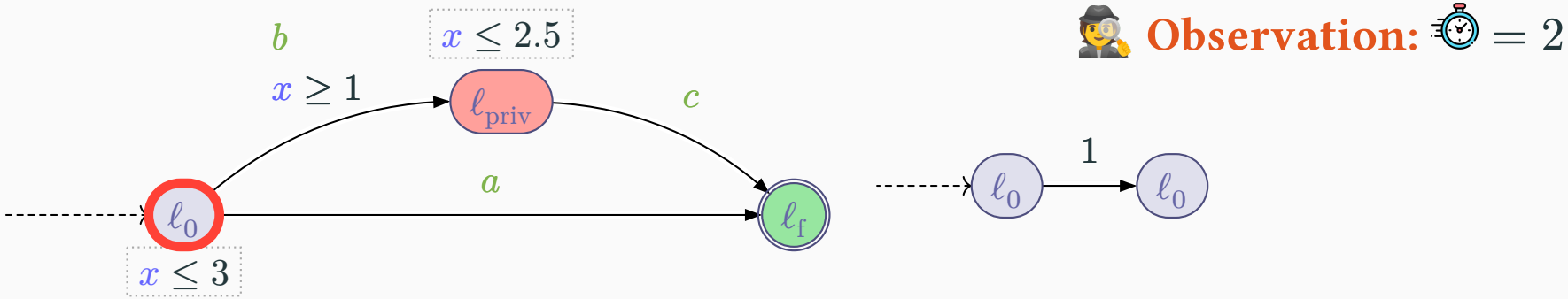
[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

A tractable notion: execution-time opacity

Execution-time Opacity

The attacker only observes the **total execution time** *(not the sequence of actions)*

The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not



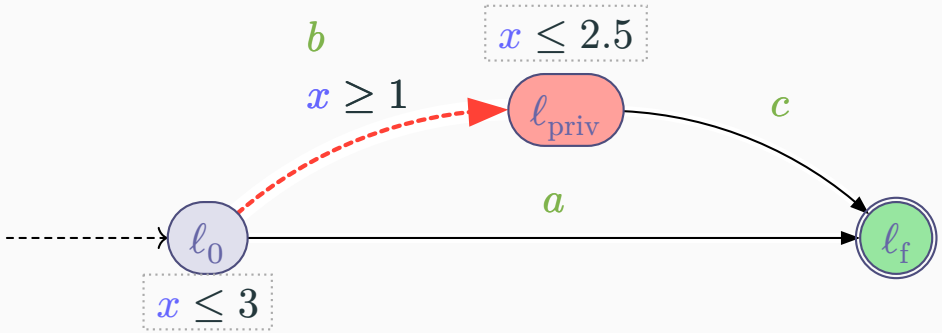
[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

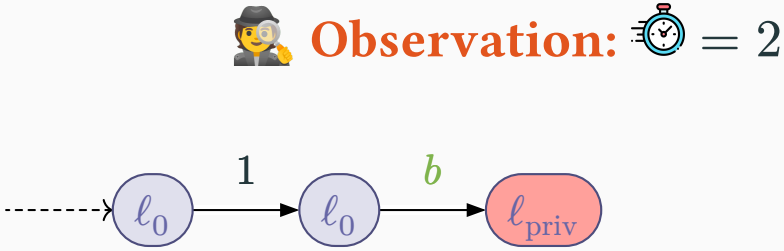
A tractable notion: execution-time opacity

Execution-time Opacity

The attacker only observes the **total execution time** *(not the sequence of actions)*

The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not





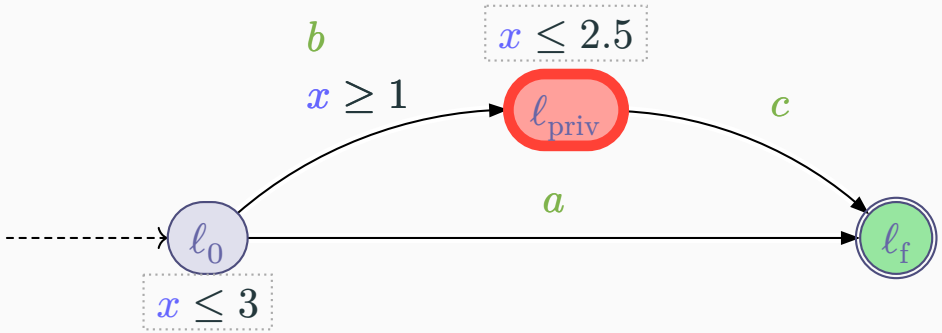
[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

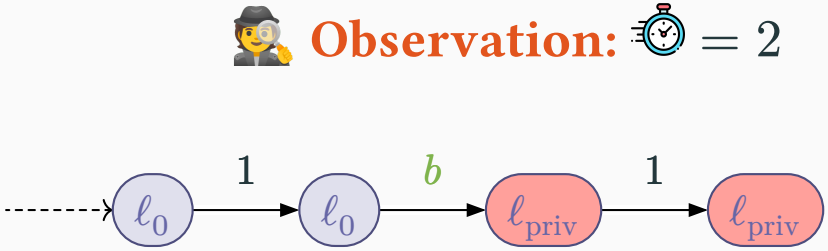
A tractable notion: execution-time opacity

Execution-time Opacity

The attacker only observes the **total execution time** *(not the sequence of actions)*

The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not





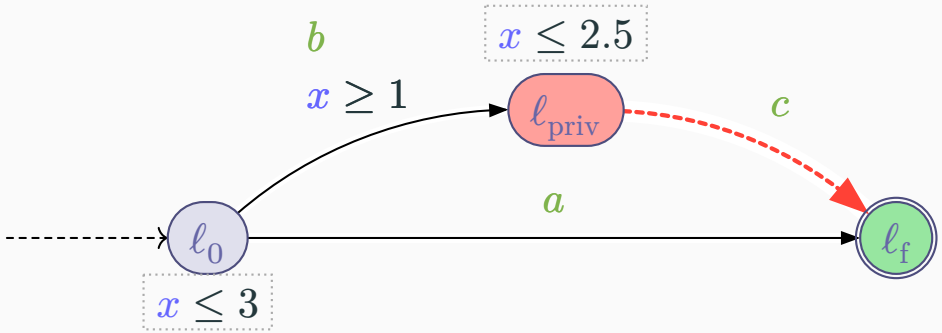
[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

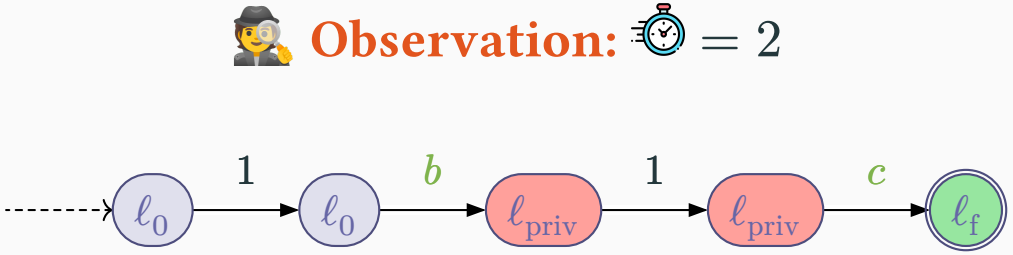
A tractable notion: execution-time opacity

Execution-time Opacity

The attacker only observes the **total execution time** *(not the sequence of actions)*

The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not





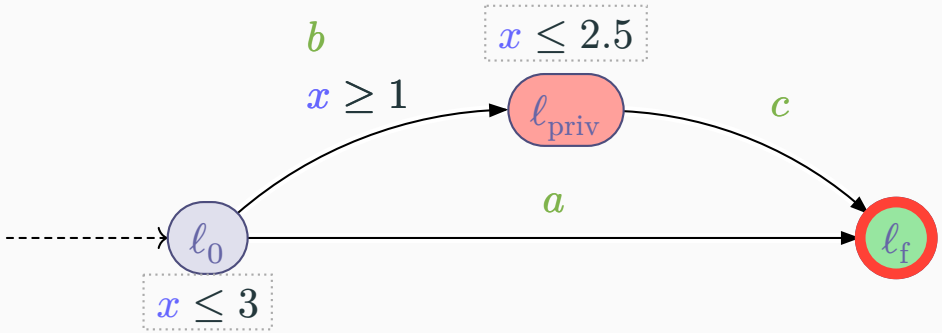
[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

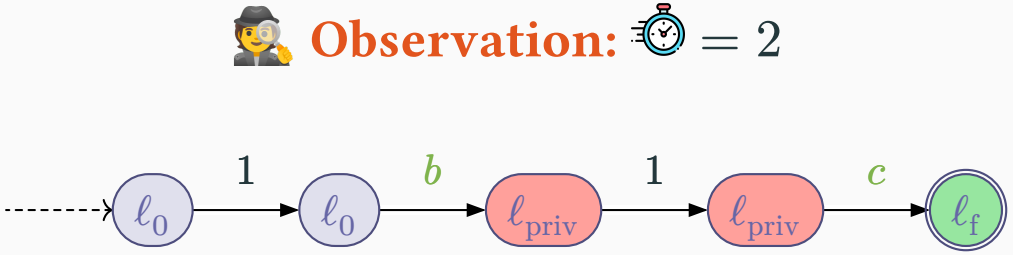
A tractable notion: execution-time opacity

Execution-time Opacity

The attacker only observes the **total execution time** *(not the sequence of actions)*

The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not





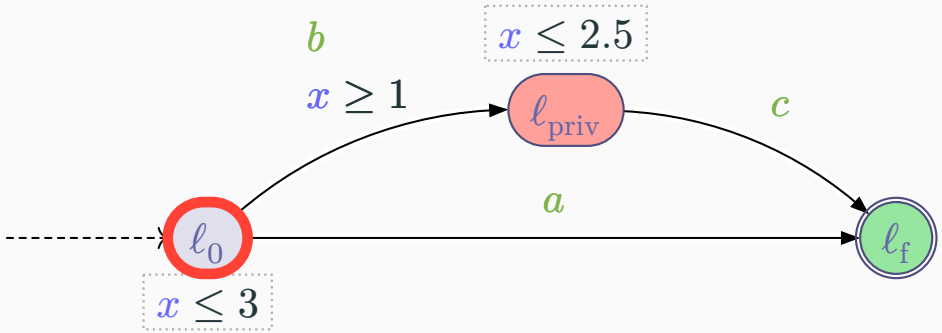
[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

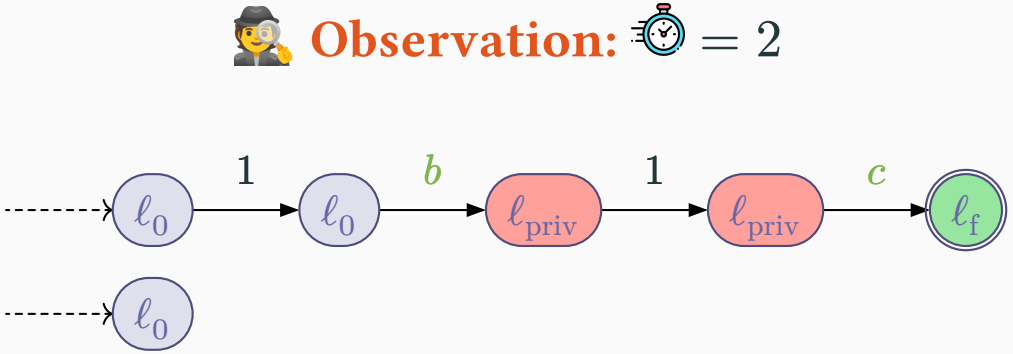
A tractable notion: execution-time opacity

Execution-time Opacity

The attacker only observes the **total execution time** *(not the sequence of actions)*

The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not





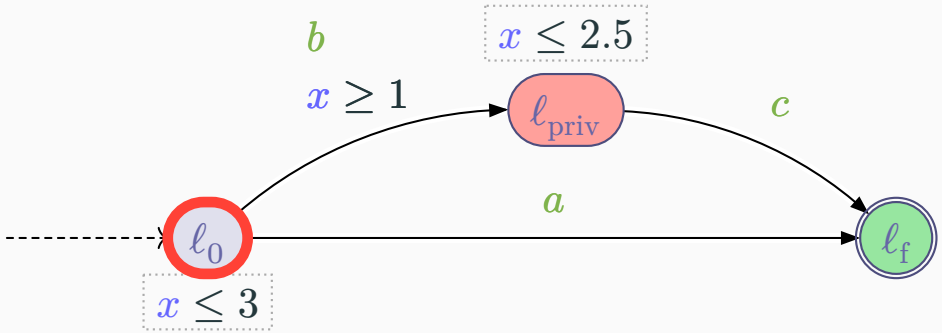
[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

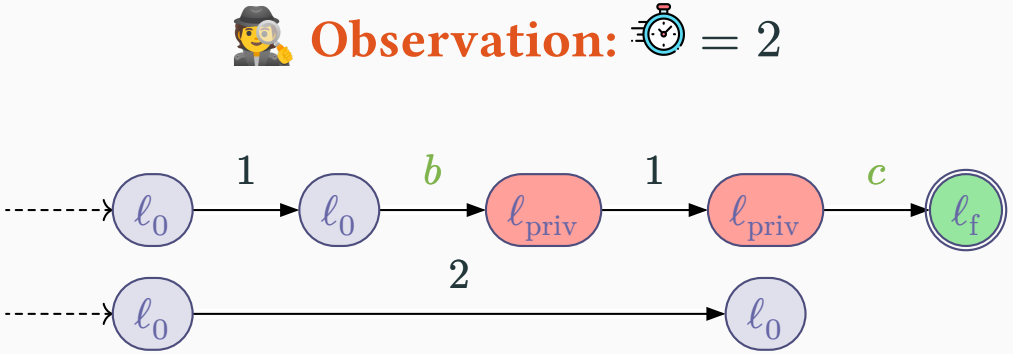
A tractable notion: execution-time opacity

Execution-time Opacity

The attacker only observes the **total execution time** *(not the sequence of actions)*

The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not





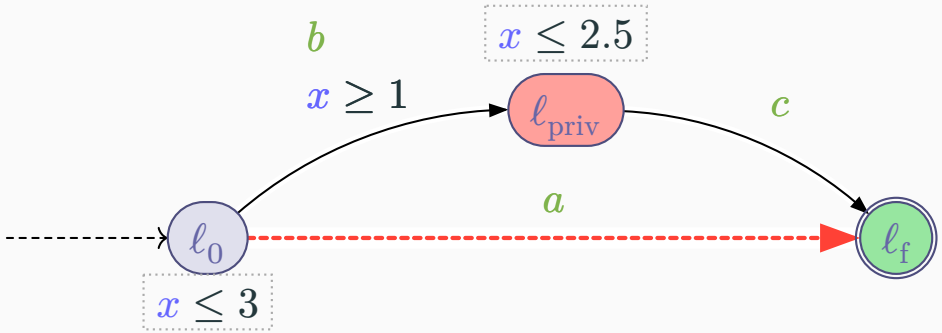
[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

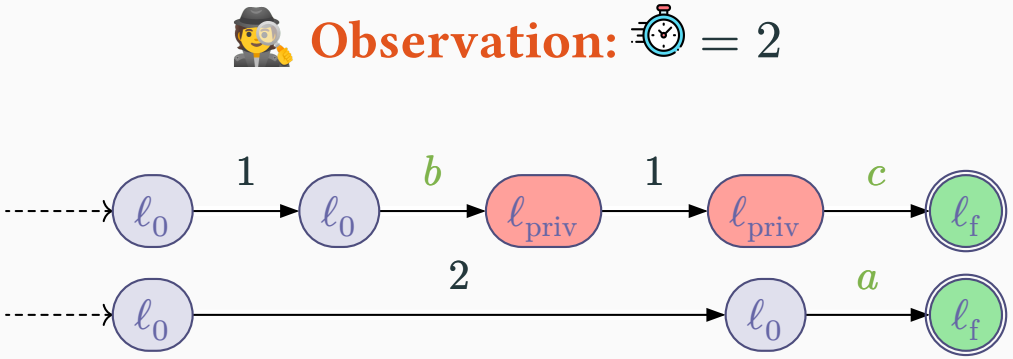
A tractable notion: execution-time opacity

Execution-time Opacity

The attacker only observes the **total execution time** *(not the sequence of actions)*

The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not





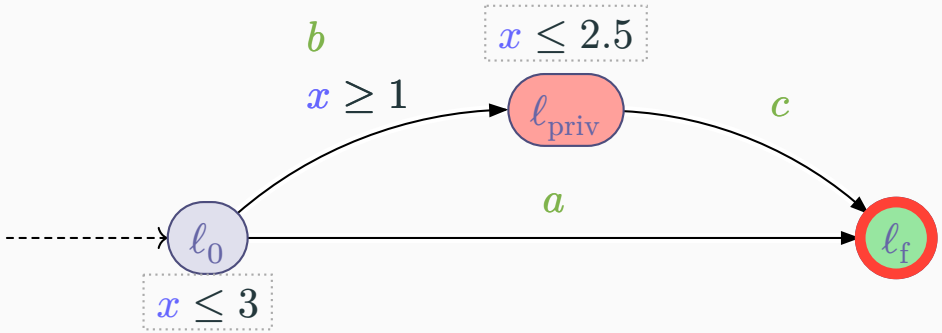
[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

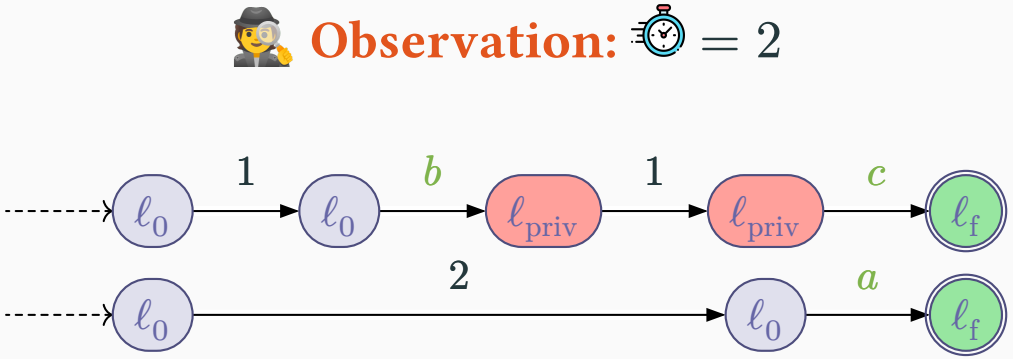
A tractable notion: execution-time opacity

Execution-time Opacity

The attacker only observes the **total execution time** *(not the sequence of actions)*

The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not





[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

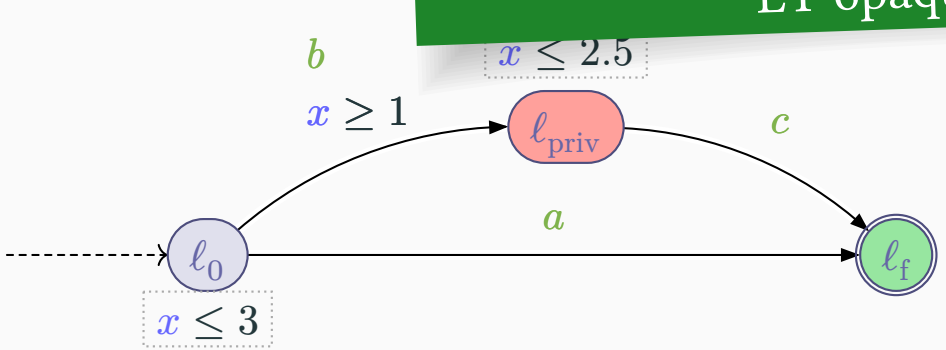
A tractable notion: execution-time opacity

Execution-time Opacity

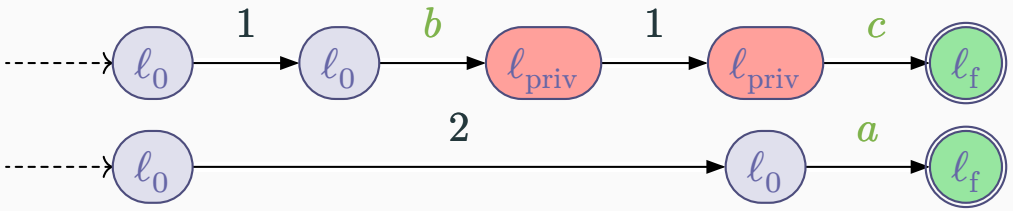
The attacker only observes the **total execution time** *(not the sequence of actions)*

The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not

ET-opaque for $d = 2$



🕵️ **Observation:** ⏱ $= 2$



[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

A tractable notion: execution-time opacity

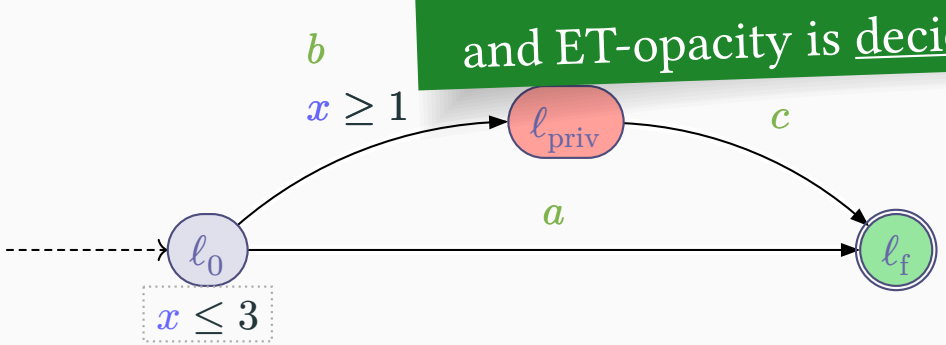
Execution-time Opacity

The attacker only observes the **total execution time** *(not the sequence of actions)*

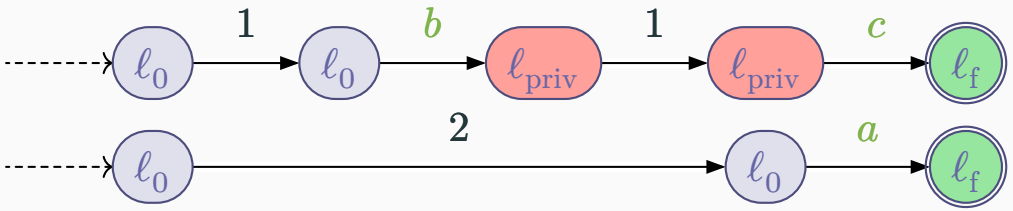
The system is **ET-opaque** for duration d if there exist two runs of duration d:
one visiting ℓ_{priv}, one not

ET-opaque for $d = 2$

and ET-opacity is decidable for timed automata!



Observation: ⏱ $= 2$



[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

Around the definition of ET-opacity

Levels of ET-opacity

- There exists a duration d for which the system is ET-opaque $DV \cap \overline{DV} \neq \emptyset$

Around the definition of ET-opacity

Levels of ET-opacity

- There exists a duration d for which the system is ET-opaque $DV \cap \overline{DV} \neq \emptyset$
- The system is ET-opaque for every duration d $DV = \overline{DV}$

Around the definition of ET-opacity

Levels of ET-opacity

- There exists a duration d for which the system is ET-opaque $DV \cap \overline{DV} \neq \emptyset$
- The system is ET-opaque for every duration d $DV = \overline{DV}$
- For every *private* duration d, there exists a *public* run of duration d $DV \subseteq \overline{DV}$

Around the definition of ET-opacity

Levels of ET-opacity

- There exists a duration d for which the system is ET-opaque $DV \cap \overline{DV} \neq \emptyset$
- The system is ET-opaque for every duration d $DV = \overline{DV}$
- For every *private* duration d, there exists a *public* run of duration d $DV \subseteq \overline{DV}$

Expiring ET-opacity



- The secret can *expire*: the attacker need to know the secret *early* enough

Around the definition of ET-opacity

Levels of ET-opacity

- There exists a duration d for which the system is ET-opaque $DV \cap \overline{DV} \neq \emptyset$
- The system is ET-opaque for every duration d $DV = \overline{DV}$
- For every *private* duration d, there exists a *public* run of duration d $DV \subseteq \overline{DV}$

Expiring ET-opacity



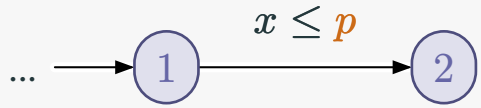
- The secret can *expire*: the attacker need to know the secret *early* enough

Parametric systems



- Parameters can be included inside the system

  eg. model unknown delays, uncertainties or to design the system



Around the definition of ET-opacity

Levels of ET-opacity

- There exists a duration d for which the system is ET-opaque $DV \cap \overline{DV} \neq \emptyset$
- The system is ET-opaque for every duration d $DV = \overline{DV}$
- For every *private* duration d, there exists a *public* run of duration d $DV \subseteq \overline{DV}$

Expiring ET-opacity

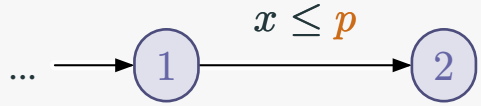


- The secret can *expire*: the attacker need to know the secret *early* enough

Parametric systems



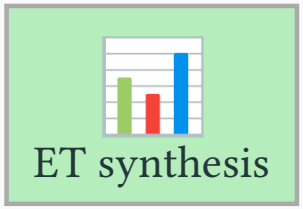
- Parameters can be included inside the system
  eg. model unknown delays, uncertainties or to design the system



↝ Approx. 10 papers studying ET-opacity and its variants:
decidability results, algorithms, applications, ...

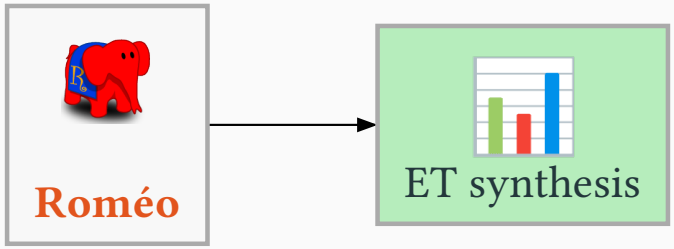
Our tool: from code to verdict

From C code to a formal verdict



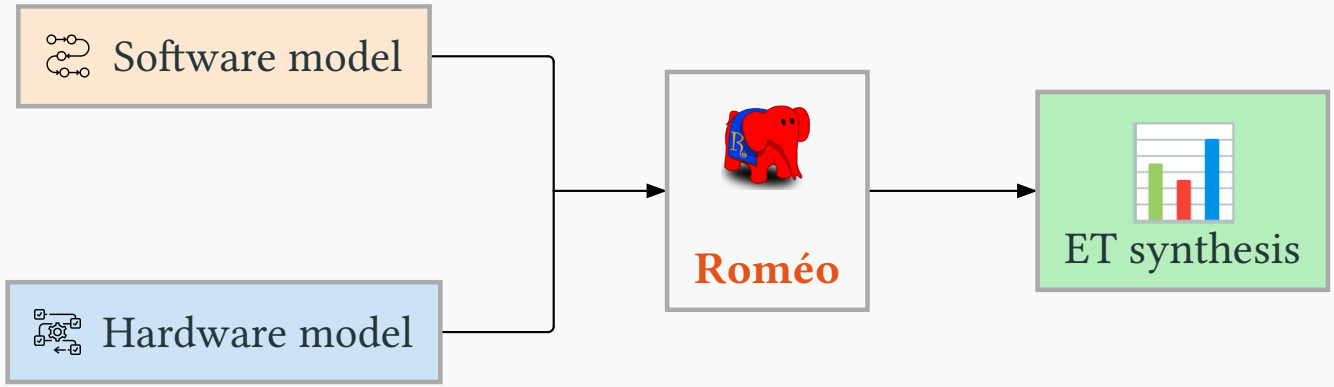
[And+25] Étienne André *et al.*, "Verifying Timed Properties of Programs in IoT nodes using Parametric Time Petri Nets," in *SAC 2025*, 2025.

From C code to a formal verdict



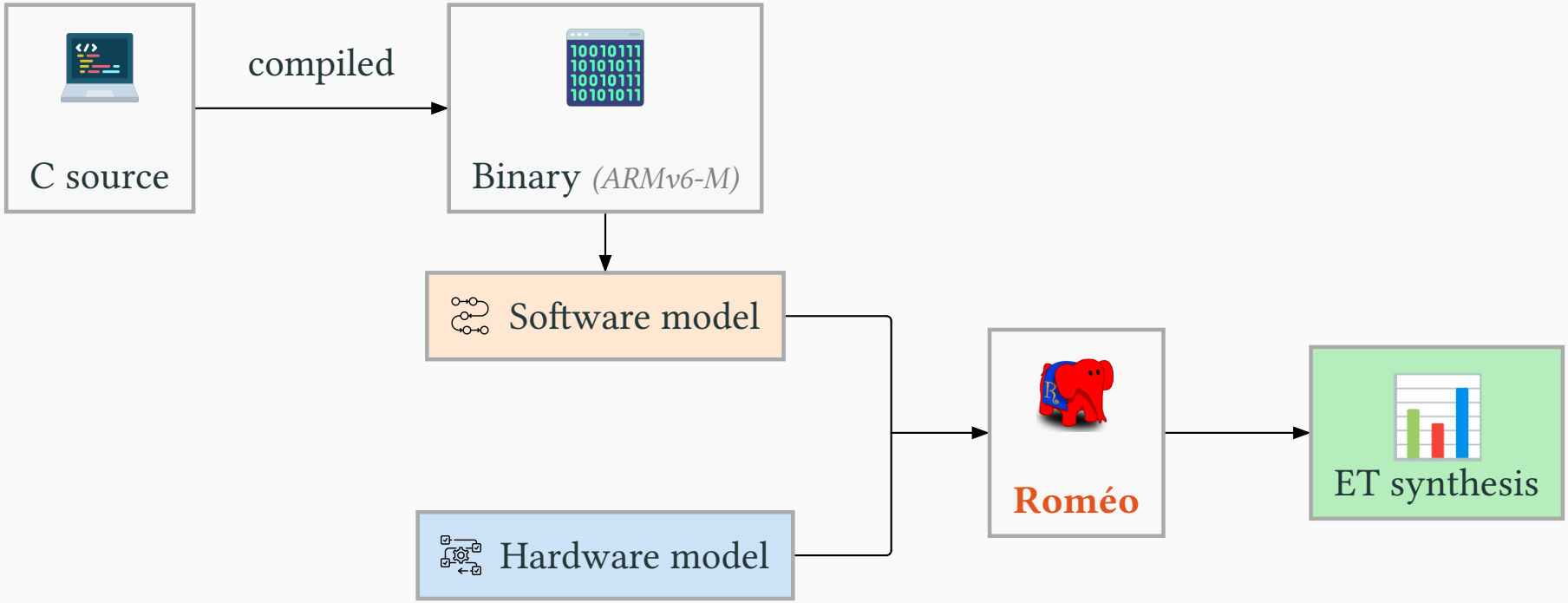
[And+25] Étienne André *et al.*, "Verifying Timed Properties of Programs in IoT nodes using Parametric Time Petri Nets," in *SAC 2025*, 2025.

From C code to a formal verdict



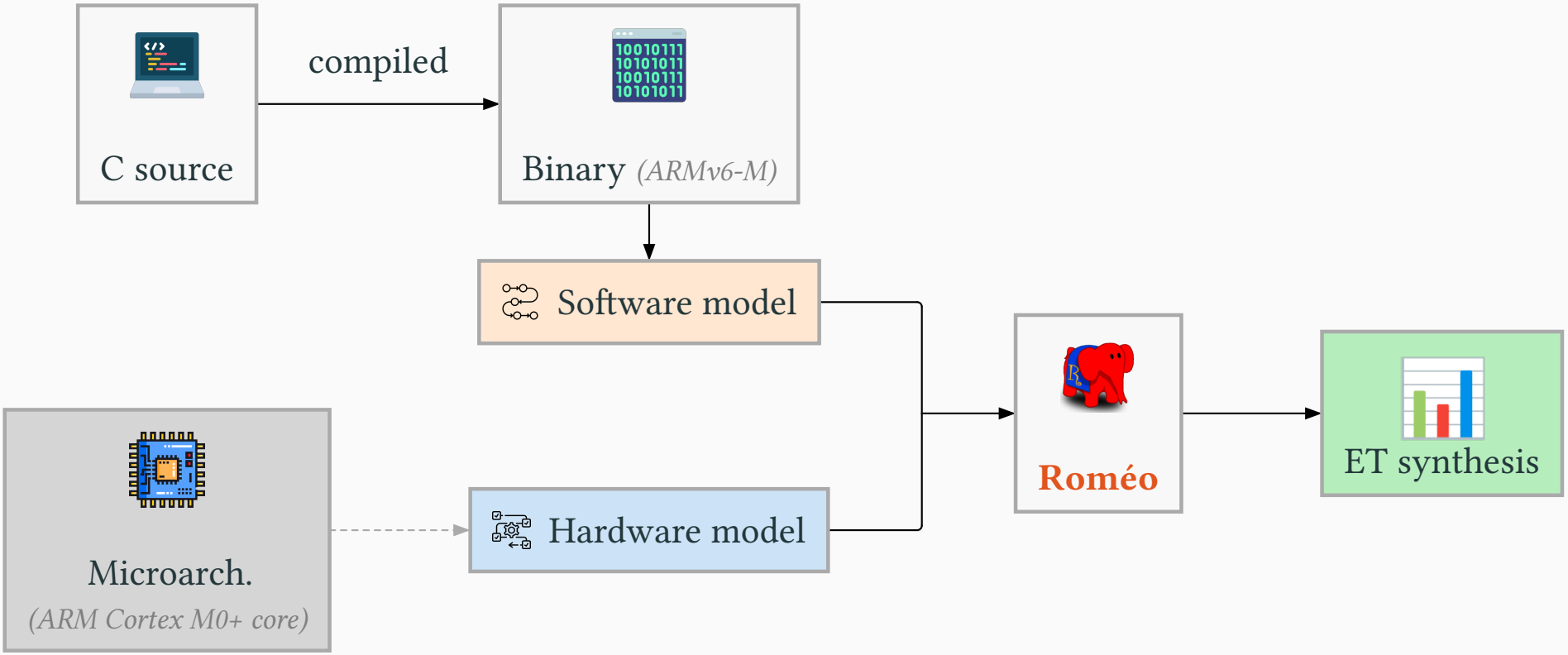
[And+25] Étienne André *et al.*, "Verifying Timed Properties of Programs in IoT nodes using Parametric Time Petri Nets," in *SAC 2025*, 2025.

From C code to a formal verdict



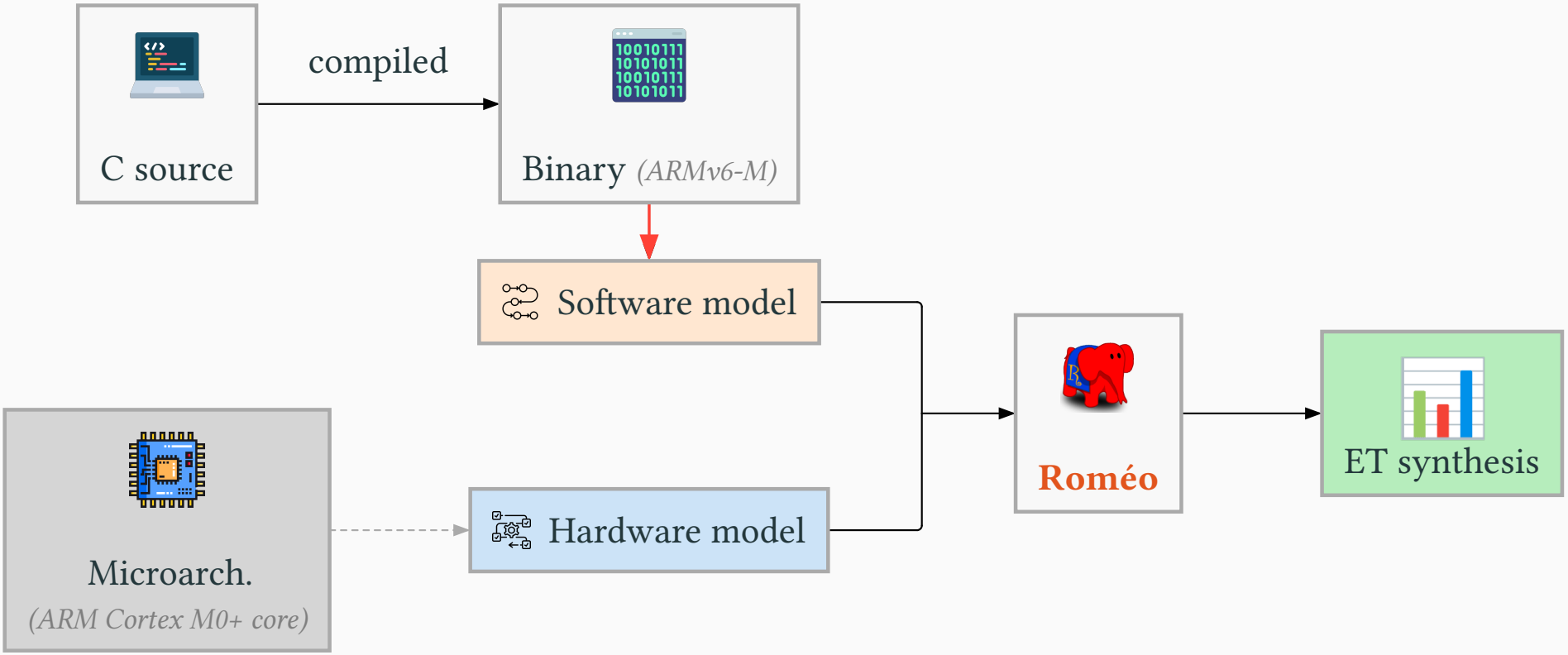
[And+25] Étienne André *et al.*, "Verifying Timed Properties of Programs in IoT nodes using Parametric Time Petri Nets," in *SAC 2025*, 2025.

From C code to a formal verdict



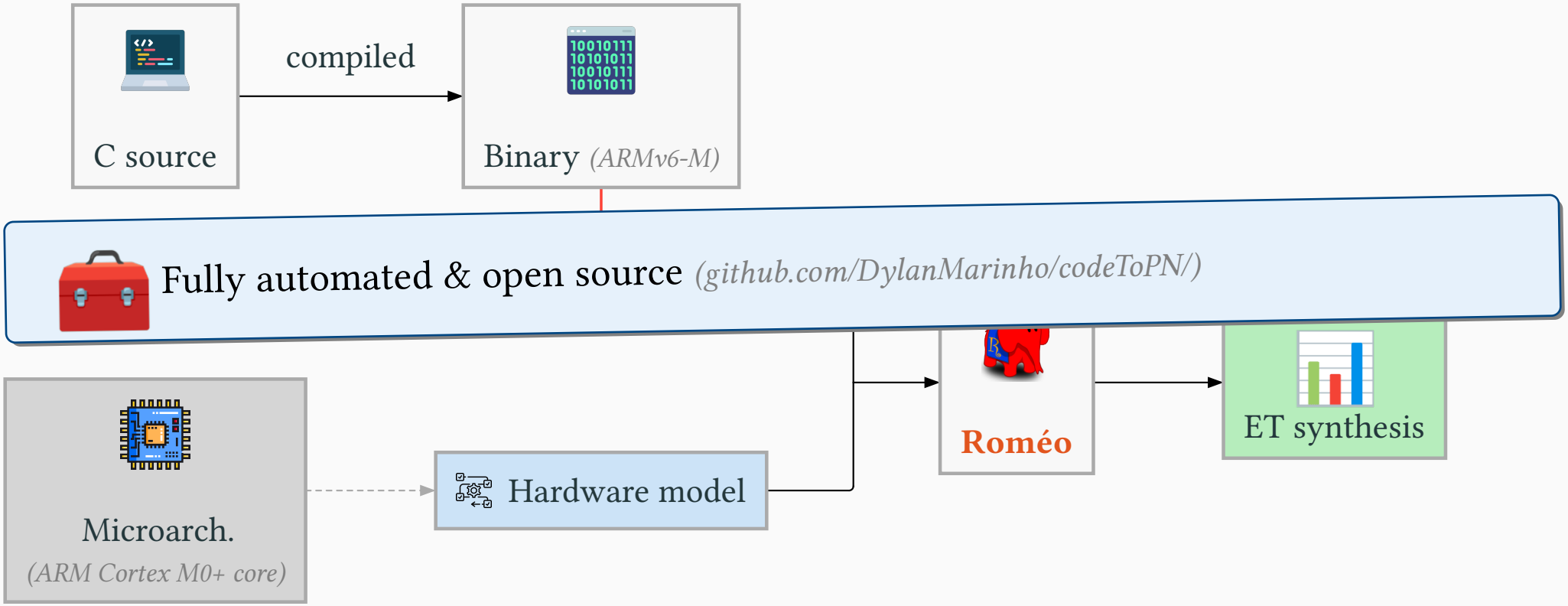
[And+25] Étienne André *et al.*, "Verifying Timed Properties of Programs in IoT nodes using Parametric Time Petri Nets," in *SAC 2025*, 2025.

From C code to a formal verdict



[And+25] Étienne André *et al.*, "Verifying Timed Properties of Programs in IoT nodes using Parametric Time Petri Nets," in *SAC 2025*, 2025.

From C code to a formal verdict



[And+25] Étienne André *et al.*, "Verifying Timed Properties of Programs in IoT nodes using Parametric Time Petri Nets," in *SAC 2025*, 2025.

Analysis of programs

Which of the following two programs is not secure?

Program 1

```c
int main() {
    int i; int length = 10;
    char ca[11] = "patehenaff";
    char cb[11] = "pasta";

    for (i = 0; i < length; i++){
        if (ca[i] != cb[i]) {
            return 0; // false
        }
    }
    return 1; // true
}
```

Program 2

```c
int main() {
    int i;  int length = 10;
    char ca[11] = "patehenaff";
    char cb[11] = "pasta";

    int result = 1; // true

    for (i = 0; i < length; i++){
        result &= (ca[i] == cb[i]);
    }
    return result;
}
```

We consider C programs compiled without optimizations, `gcc -O0`

Which of the following two programs is not secure?

Program 1

```c
int main() {
    int i; int length = 10;
    char ca[11] = "patehenaff";
    char cb[11] = "pasta";

    for (i = 0; i < length; i++){
        if (ca[i] != cb[i]) {
            return 0; // false
        }
    }
    return 1; // true
}
```

Unsecure

Program 2

```c
int main() {
    int i;  int length = 10;
    char ca[11] = "patehenaff";
    char cb[11] = "pasta";

    int result = 1; // true

    for (i = 0; i < length; i++){
        result &= (ca[i] == cb[i]);
    }
    return result;
}
```

Secure

We consider C programs compiled without optimizations, `gcc -O0`

Which of the following two programs is not secure?

Program 1

```c
int main() {
    int i; int length = 10;
    char ca[11] = "patehenaff";
    char cb[11] = "pasta";

    for (i = 0; i < length; i++){
        if (ca[i] != cb[i]) {
            return 0; // false
        }
    }
    return 1; // true
}
```

Unsecure - ET sensitive

- 758 for the secret password
- {362, 404, 446, 488, 530, 572, 614, 656, 698, 740} for any other password

Program 2

```c
int main() {
    int i;  int length = 10;
    char ca[11] = "patehenaff";
    char cb[11] = "pasta";

    int result = 1; // true

    for (i = 0; i < length; i++){
        result &= (ca[i] == cb[i]);
    }
    return result;
}
```

Secure - Constant ET: 876

Is this third program secure?

Program 3

```c
int main () {
  int i ;
  int length = 10; // length of the strings
  char ca[11] = " patehenaff " ;
  char cb[11] = " pasta " ;

  int result = 1; // true
  for (i=0; i<length ; i++) {
    if (ca[i] == cb[i]) {
      result &= 1;
    } else {
      result &= 0;
    }
  }
  return result ;
}
```

Is this third program secure?

Program 3

```c
int main () {
  int i ;
  int length = 10; // length of the strings
  char ca[11] = " patehenaff " ;
  char cb[11] = " pasta " ;

  int result = 1; // true
  for (i=0; i<length ; i++) {
    if (ca[i] == cb[i]) {
      result &= 1;
    } else {
      result &= 0;
    }
  }
  return result ;
}
```

- It seems so: very close to the former secure program

Is this third program secure?

Program 3

```c
int main () {
  int i ;
  int length = 10; // length of the strings
  char ca[11] = " patehenaff " ;
  char cb[11] = " pasta " ;

  int result = 1; // true
  for (i=0; i<length ; i++) {
    if (ca[i] == cb[i]) {
      result &= 1;
    } else {
      result &= 0;
    }
  }
  return result ;
}
```

- It seems so: very close to the former secure program
- But it is not due to the **instruction cache**
  - 876 for the secret password
  - {816, 822, 828, 834, 840, 846, 852, 858, 864, 870} for any other password

Is this third program secure?

Program 3

```c
int main () {
  int i ;
  int length = 10; // length of the strings
  char ca[11] = " patehenaff " ;
  char cb[11] = " pasta " ;

  int result = 1; // true
  for (i=0; i<length ; i++) {
    if (ca[i] == cb[i]) {
      result &= 1;
    } else {
      result &= 0;
    }
  }
  return result ;
}
```

- It seems so: very close to the former secure program
- But it is not due to the **instruction cache**
  - 876 for the secret password
  - {816, 822, 828, 834, 840, 846, 852, 858, 864, 870} for any other password



We can **reconfigure** the program, by **making it opaque**

- *adding 6 nop instructions at the end of one branch*
- *(see paper)*

Conclusion

Summary and perspectives

Contributions



- **Automated** detection of timing leaks in C programs
- End-to-end: C source → formal model *(SW + HW)* → ET computation
- Demonstrated on password comparison – including a **non-obvious** vulnerability (instruction cache)
- Open-source tool *(github.com/DylanMarinho/codeToPN/)*

Future work



- – **Tool** with views of executions, extending to **data cache** - *Internships ongoing*
- Programs on **multicore** architectures
- **Automatic** reconfiguration to restore opacity
- More attacks: fault-injection, energy-based, cache side-channels







3 June 2026 | 1er Congrès des Sciences du Logiciel | Invitation GDR IFM | Lille, France

Detecting Timing Leaks of Programs

using Parametric Timed Model Checking

Dylan Marinho

Sorbonne Université, CNRS UMR 7606, LIP6

Based on joint work with Étienne André, Jean-Luc Béchennec, Sudipta Chattopadhyay, Sébastien Faucou, Didier Lime, Olivier H. Roux, Jun Sun





CC BY NC SA

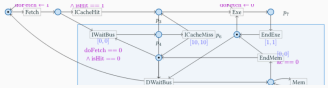
Other slides

Model checking

System



Formal model



Specification

"The system must be safe"

Property

$AG \neg$ ○

Model checking

?

$\models$

Yes

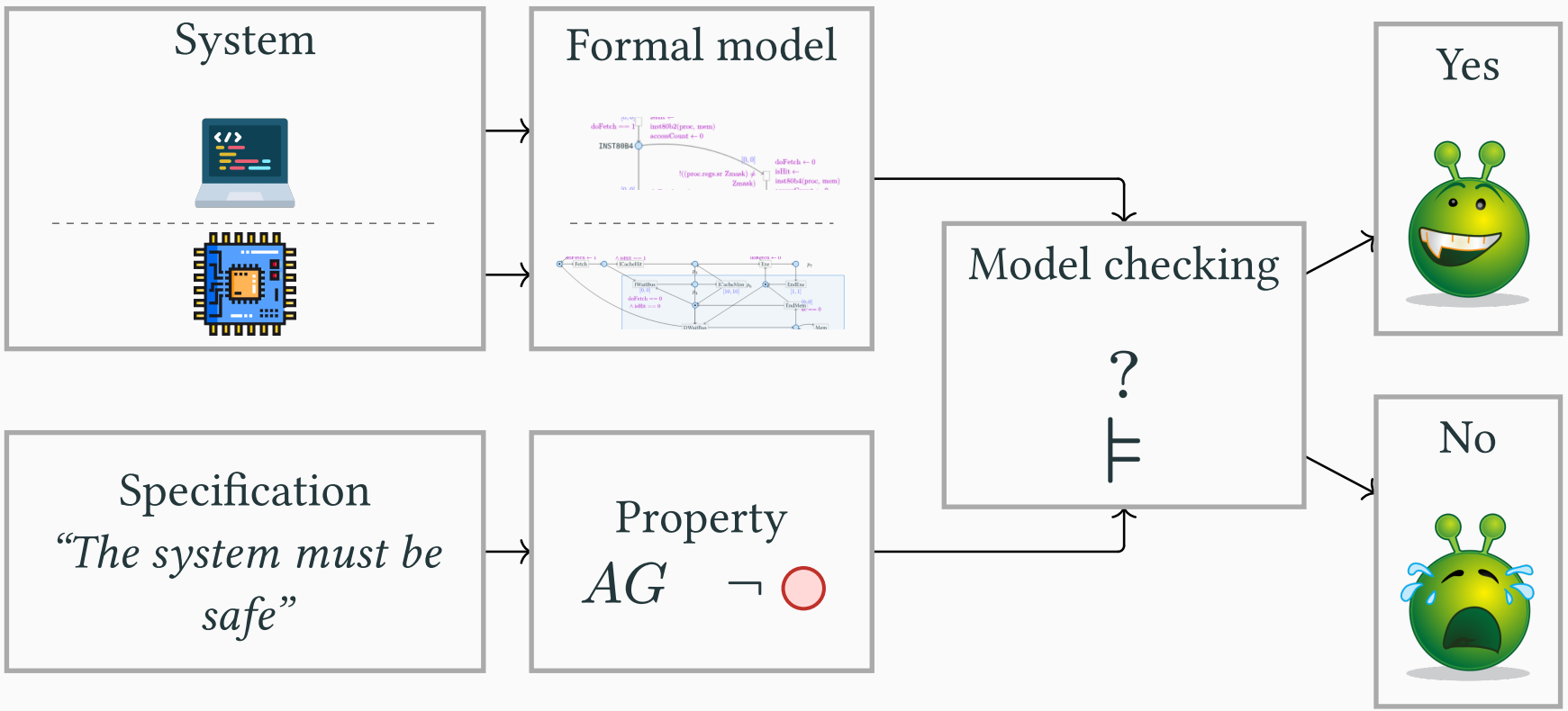


No



Question: does the model of the system **satisfy** the property?

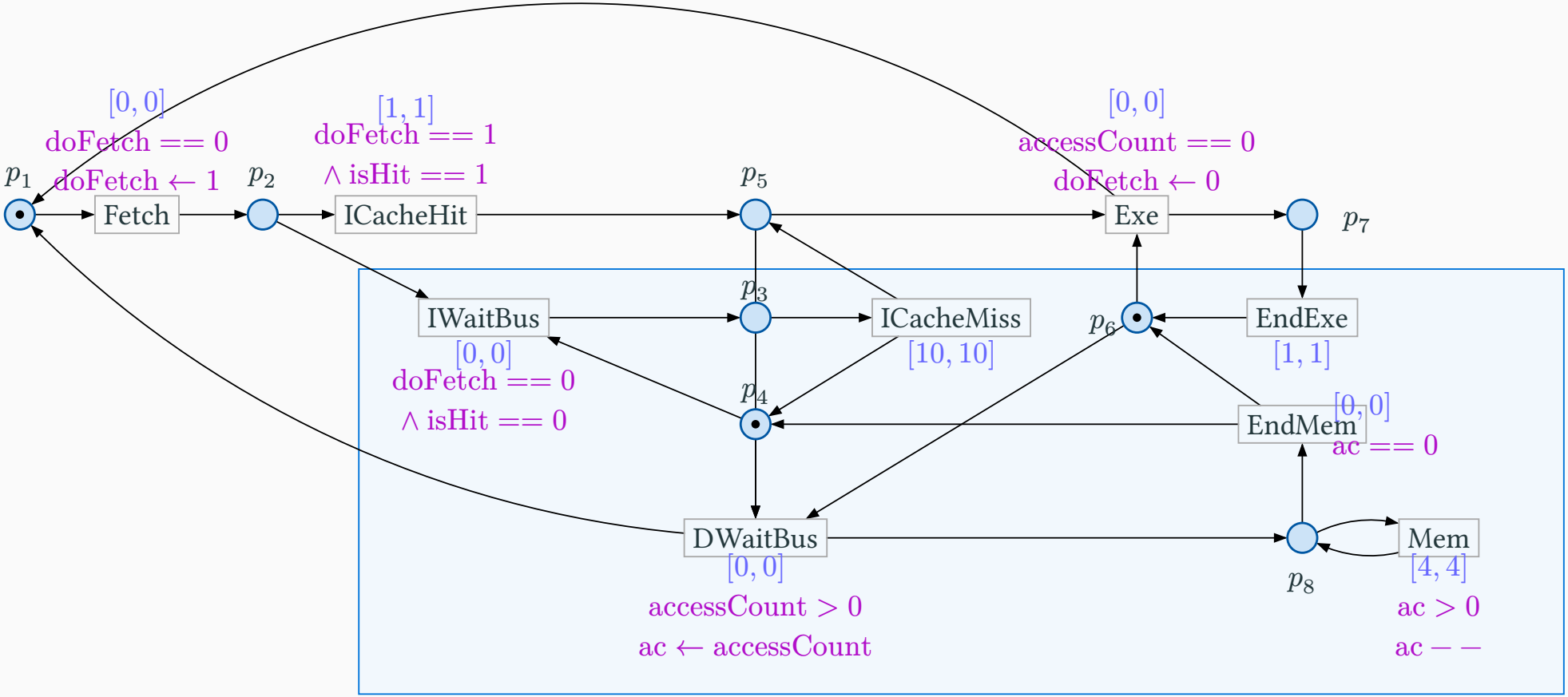
Model checking



Question: does the model of the system **satisfy** the property?

PTPN hardware model

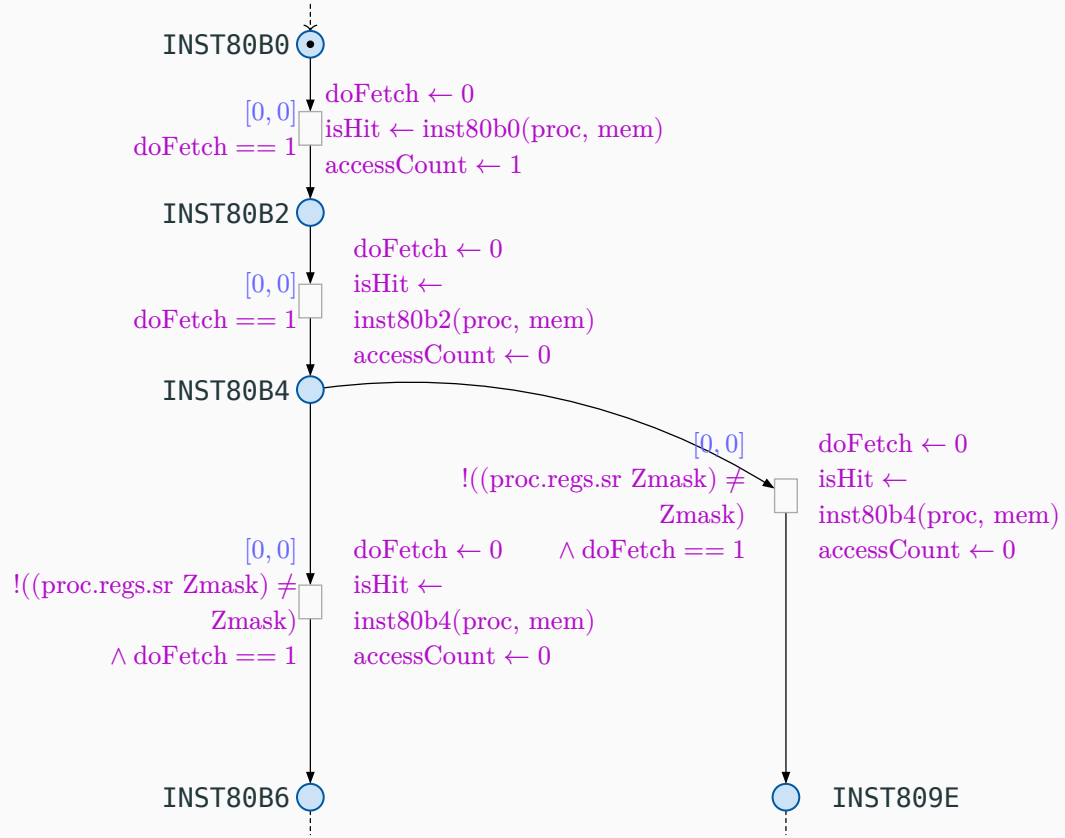
- doFetch, isHit and accessCount: variables used to synchronize with the software



PTPN software model

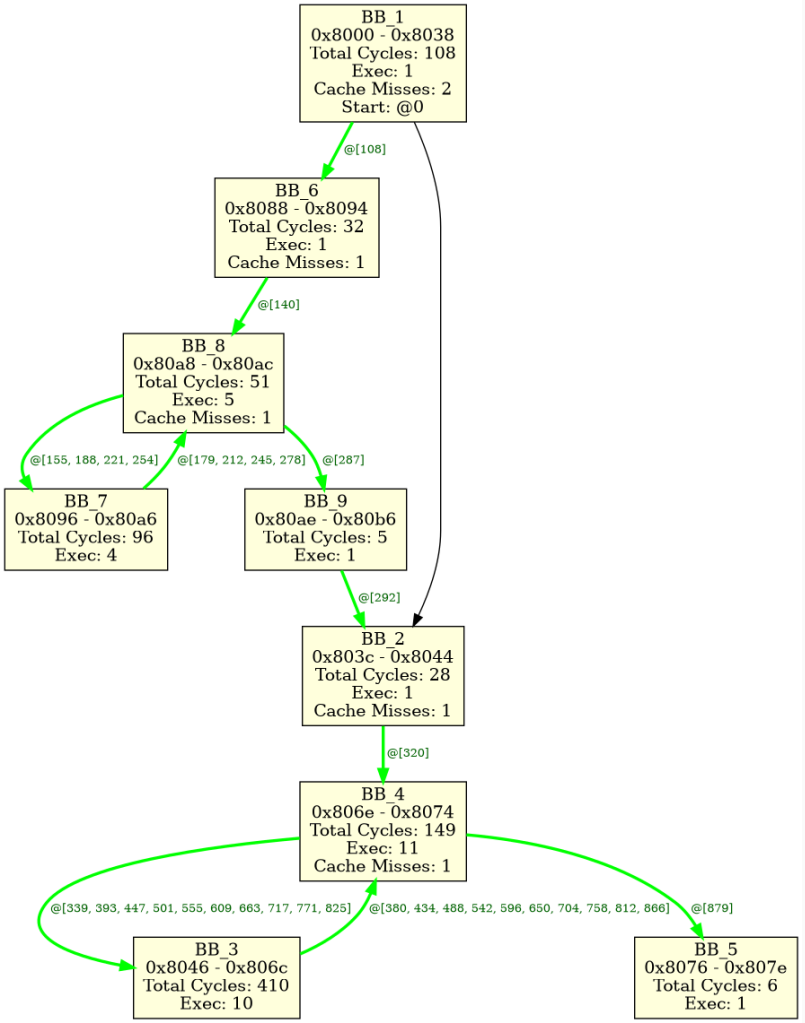
Software

- Captures the **binary code** of the program *(ARMv6-M)*
  - Firing a transition corresponds to executing the instruction
  - Pipeline fetch: doFetch
  - Memory access: accessCount and isHit
- Structurally identical to the **control flow graph**



```
$ ./cchronos benchmarks/secure_string_comparison_bitwise.c --mode time --cfg
[ >> ] Compiling benchmarks/secure_string_comparison_bitwise.c ...
[ >> ] [...]
[ >> ] Model built
[ >> ] CTS file generated
[ >> ] Running romeo-cli on [...] ...
[ >> ] Extracting timed trace for CFG overlay...
[>>>>] p in [885, 886]
[ >> ] Generating Timed CFG...
[ >> ] Timed CFG successfully overlaid
```

Shell



Bibliography

Bibliography

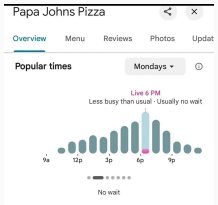
[Cas09] Franck Cassez, "The Dark Side of Timed Opacity," in *ISA*, 2009.

[And+22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun, "Guaranteeing Timed Opacity using Parametric Timed Model Checking," *TOSEM*, 2022.

[And+25] Étienne André *et al.*, "Verifying Timed Properties of Programs in IoT nodes using Parametric Time Petri Nets," in *SAC 2025*, 2025.

Additional information

Explanation of the pictures



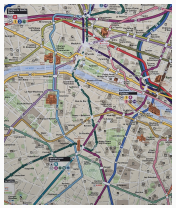
- USA, June 2025
- Empty bars during Iran's riposte against US military bases.

(Dr. Dominic Ng)



- USA, 24 June 2025
- After the Israel-Iran ceasefire

(Dr. Dominic Ng)



- Paris Metro map, Madeleine station (Paris, France)

Licensing

Sources of the graphics



- Title : Smiley green alien big eyes (aaah)
- Author : LadyofHats
- Source : https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
- License : Public domain



- Title : Smiley green alien big eyes (cry)
- Author : LadyofHats
- Source : https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
- License : Public domain

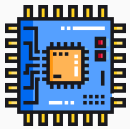


- Title : Smiley green alien exterminate
- Author : LadyofHats
- Source : https://commons.wikimedia.org/wiki/File:Smiley_green_alien_exterminate.svg
- License : Public domain

Sources of the graphics



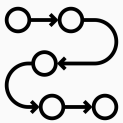
- Source: Flaticon.com



- Source: Flaticon.com

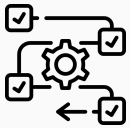


- Source: Flaticon.com



- Source: Flaticon.com

Sources of the graphics



- Source: Flaticon.com

License of this document

This presentation can be published, reused and modified under the terms of the license Creative Commons **Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**



creativecommons.org/licenses/by-nc-sa/4.0/

Author: **Dylan Marinho**

Version: 2026-06-03