MeFoSyLoMa

November 10th, 2023
Évry, France

Preventing Timing Leaks using Parametric Timed Model Checking

Dylan Marinho

Télécom SudParis, Institut Polytechnique de Paris

Based on join works with Étienne André, Shapagat Bolat, Engel Lefaucheux, Didier Lime, and Sun Jun

These works are partially supported by the ANR-NRF research program ProMiS (ANR-19-CE25-0015) and the ANR research program BisoUS (ANR-22-CE48-0012).

General context: side-channel attacks

- Threats to a system using non-algorithmic weaknesses

General context: side-channel attacks

- Threats to a system using non-algorithmic weaknesses
  - Cache attacks
  - Electromagnetic attacks
  - Power attacks
  - Acoustic attacks
  - Timing attacks
  - Temperature attacks
  - etc.

General context: side-channel attacks

- Threats to a system using non-algorithmic weaknesses
  - Cache attacks
  - Electromagnetic attacks
  - Power attacks
  - Acoustic attacks
  - Timing attacks
  - Temperature attacks
  - etc.

- Example
  - Number of pizzas (and order time) ordered by the white house prior to major war announcements [1]

[1] `http://home.xnet.com/~warinner/pizzacites.html`

General context: side-channel attacks

- Threats to a system using non-algorithmic weaknesses
  - Cache attacks
  - Electromagnetic attacks
  - Power attacks
  - Acoustic attacks
  - Timing attacks
  - Temperature attacks
  - etc.
- Example
  - Number of pizzas (and order time) ordered by the white house prior to major war announcements [1]

[1] http://home.xnet.com/~warinner/pizzacites.html

A simple example of timing attack

```
# input pwd     : Real password
# input attempt: Tentative password
for i = 0 to min(len(pwd), len(attempt)) − 1 do
    if pwd[i] ≠ attempt[i] then
        return false
done
return true
```

A simple example of timing attack

```
# input pwd      : Real password
# input attempt: Tentative password
for i = 0 to min(len(pwd), len(attempt)) − 1 do
    if pwd[i] ≠ attempt[i] then
        return false
done
return true
```

pwd	c	h	i	c	k	e	n
attempt	c	h	e	e	s	e	

Execution time:

A simple example of timing attack

```
# input pwd     : Real password
# input attempt: Tentative password
for i = 0 to min(len(pwd), len(attempt)) − 1 do
    if pwd[i] ≠ attempt[i] then
        return false
done
return true
```

pwd	c	h	i	c	k	e	n
attempt	c	h	e	e	s	e	

Execution time: ϵ

A simple example of timing attack

```
# input pwd     : Real password
# input attempt: Tentative password
for i = 0 to min(len(pwd), len(attempt)) − 1 do
    if pwd[i] ≠ attempt[i] then
        return false
done
return true
```

pwd	c	h	i	c	k	e	n
attempt	c	h	e	e	s	e	

Execution time: $\epsilon + \epsilon$

A simple example of timing attack

```
# input pwd     : Real password
# input attempt: Tentative password
for i = 0 to min(len(pwd), len(attempt)) − 1 do
    if pwd[i] ≠ attempt[i] then
        return false
done
return true
```

pwd	c	h	i	c	k	e	n
attempt	c	h	e	e	s	e	

Execution time: $\epsilon + \epsilon + \epsilon$

A simple example of timing attack

```
# input pwd     : Real password
# input attempt: Tentative password
for i = 0 to min(len(pwd), len(attempt)) − 1 do
    if pwd[i] ≠ attempt[i] then
        return false
done
return true
```

pwd	c	h	i	c	k	e	n
attempt	c	h	e	e	s	e	

Execution time: $\epsilon + \epsilon + \epsilon$

- Problem: The execution time is proportional to the number of consecutive correct characters from the beginning of `attempt`

Timing attacks

- Principle: deduce private information from timing data (execution time)

Issues:

- May depend on the implementation (or, even worse, be introduced by the compiler)
- A relatively trivial solution: make the program last always its maximum execution time
  Drawback: loss of efficiency

$\leadsto$ Non-trivial problem

Detection

Need to detect timing-leak vulnerabilities

Detection

Need to detect timing-leak vulnerabilities

We want formal guarantees $\rightarrow$ formal methods

- Various methods:
  - Abstract interpretation
  - Static analysis
  - Model checking
  - Theorem proving

Detection

Need to detect timing-leak vulnerabilities

We want formal guarantees $\rightarrow$ formal methods

- Various methods:
  - Abstract interpretation
  - Static analysis
  - Model checking
  - Theorem proving

Methodology

A program

A specification

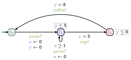
“The program must be secure”

Methodology

A program

→

A model



A specification

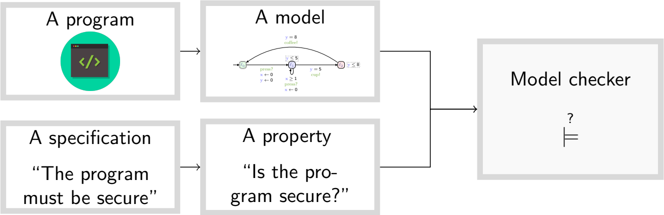
"The program must be secure"

→

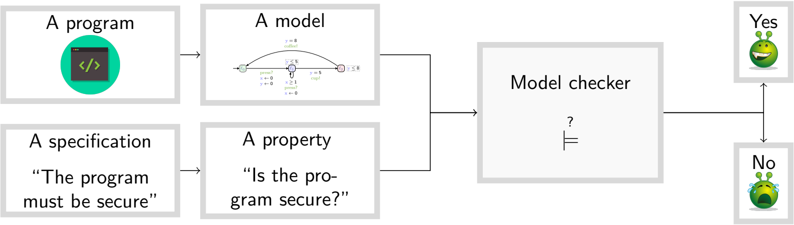
A property

"Is the program secure?"

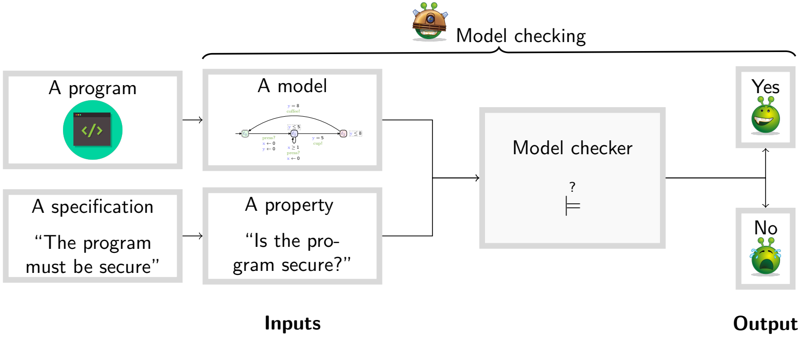
Methodology



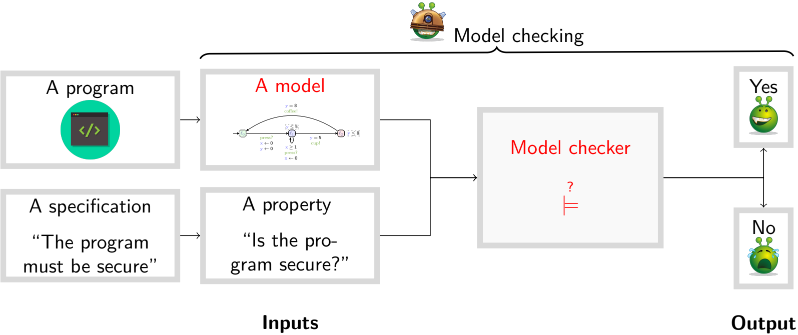
Methodology



Methodology



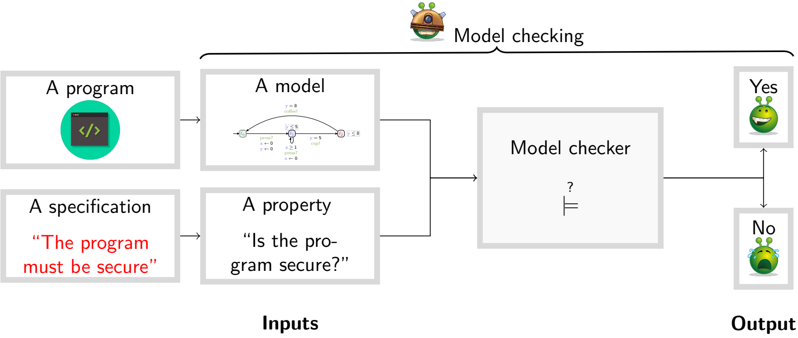
Outline



Outline

1. Preliminaries: Timed model checking

Outline



Outline

1. Preliminaries: Timed model checking
2. Execution-time opacity

Outline

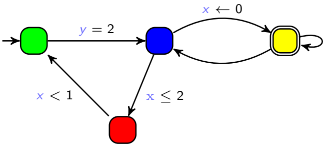
Preliminaries: (Parametric) Timed model checking

Execution-time opacity

Conclusion & Perspectives

Outline

Timed model checking

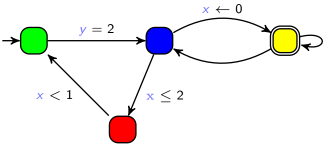


A model of the system

 is unreachable

A property to be satisfied

Timed model checking



A model of the system

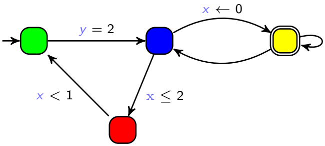
$\overset{?}{\models}$

 is unreachable

A property to be satisfied

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

Timed model checking



A model of the system

$\overset{?}{\models}$

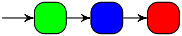
is unreachable

A property to be satisfied

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

Yes

No

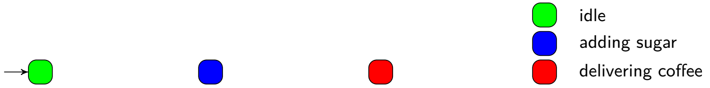


Counterexample

Timed automaton (TA)

[AD94]

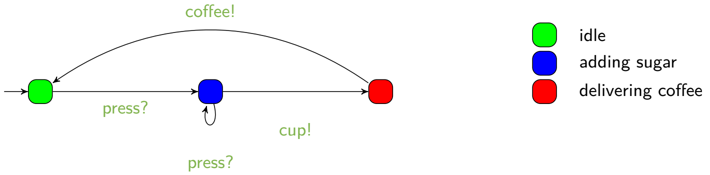
- Finite state automaton (sets of locations)



Timed automaton (TA)

[AD94]

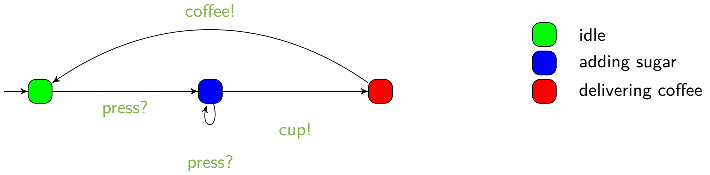
- Finite state automaton (sets of locations and actions)



Timed automaton (TA)

[AD94]

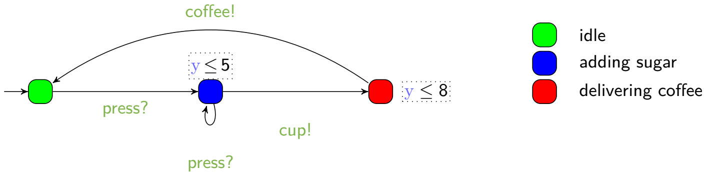
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks
  - Real-valued variables evolving linearly at the same rate



Timed automaton (TA)

[AD94]

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants
- Features
  - Location invariant: property to be verified to stay at a location

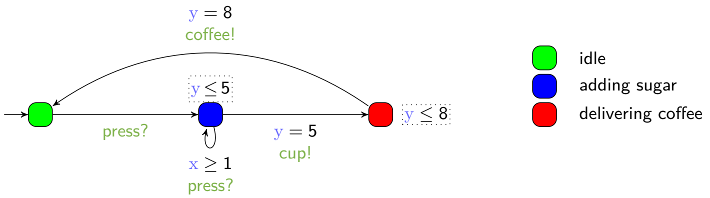


Timed automaton (TA)

[AD94]

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

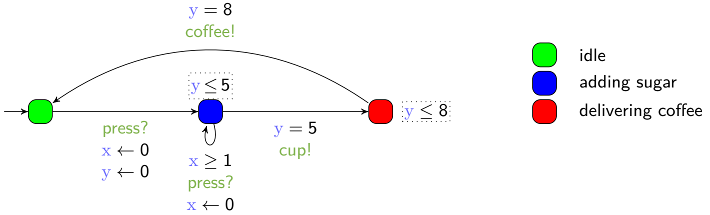
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition



Timed automaton (TA)

[AD94]

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 along transitions



Outline

Preliminaries: (Parametric) Timed model checking

Timed model checking and Timed automata

Parametric timed model checking and Parametric timed automata

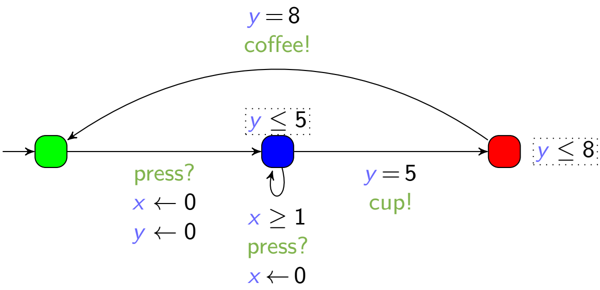
Execution-time opacity

Conclusion & Perspectives

Timed Automaton (PTA)

[AHV93]

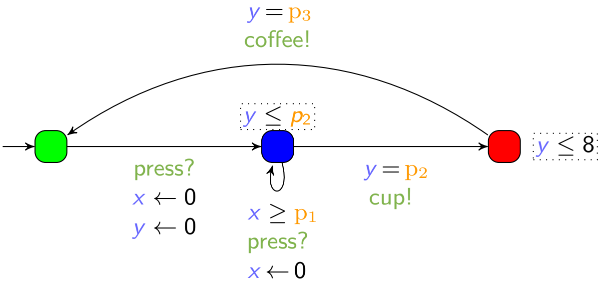
- Timed automaton (sets of locations, actions and clocks)



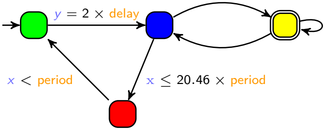
Parametric Timed Automaton (PTA)

[AHV93]

- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters
  - Unknown constants compared to a clock in guards and invariants



timed model checking



A model of the system

$\overset{?}{\models}$

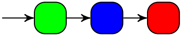
 is unreachable

A property to be satisfied

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

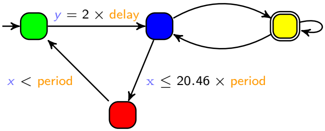
Yes

No



Counterexample

Parametric timed model checking



A model of the system

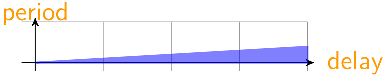
$\models^{?}$

 is unreachable

A property to be satisfied

- Question: for what values of the parameters does the model of the system satisfy the property?

Yes if...



$2 \times \text{delay} > 20.46 \times \text{period}$

Outline

Execution-time opacity

- How to detect timing-leak vulnerabilities?

Execution-time opacity

- How to detect timing-leak vulnerabilities?

Goal

- Propose a formalization of the private information and attacker model
- Check whether a model is secure or not

Execution-time opacity

- How to detect timing-leak vulnerabilities?

Goal

- Propose a formalization of the private information and attacker model
- Check whether a model is secure or not

Contributions

- ET-opacity definition, decidability results and experiments [TOSEM22]
- Expiring ET-opacity definition and decidability results [ICECCS23]
- Untimed control [FTSCS22]

Our attacker model

Attacker capabilities

- Has access to the model (white box)
- Can only observe the total execution time

Our attacker model

Attacker capabilities

- Has access to the model (white box)
- Can only observe the total execution time

Attacker goal

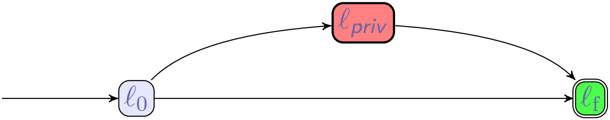
- Wants to deduce some private information based on these observations
  $\rightarrow$ visit of a private location

Outline

Formalization

Hypotheses: [AS19][TOSEM22]

- A start location ℓ_0 and an end location ℓ_f
- A special private location ℓ_{priv}

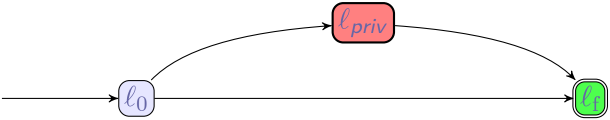


[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *ACM TOSEM* (2022)

Formalization

Hypotheses: [AS19][TOSEM22]

- A start location ℓ_0 and an end location ℓ_f
- A special private location ℓ_{priv}



Definition (execution-time opacity)

The system is ET-opaque for a duration d if there exist two runs to ℓ_f of duration d

1. one visiting ℓ_{priv}
2. one *not* visiting ℓ_{priv}

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *ACM TOSEM* (2022)

Three levels of ET-opacity

Existential ($\exists$)

There exist a duration d and two runs of duration d,
one visiting ℓ_{priv},
one not visiting ℓ_{priv}

Three levels of ET-opacity

Existential ($\exists$)

private durations $\cap$ public durations $\neq \emptyset$

Three levels of ET-opacity

Existential ($\exists$)

private durations $\cap$ public durations $\neq \emptyset$

Weak

For all durations d,
There exists a run of duration d visiting ℓ_{priv}
$\Rightarrow$
There exists a run of duration d not visiting ℓ_{priv}

Three levels of ET-opacity

Existential ($\exists$)

private durations $\cap$ public durations $\neq \emptyset$

Weak

For all durations d,
There exists a run of duration d visiting ℓ_{priv}
$\Rightarrow$
There exists a run of duration d not visiting ℓ_{priv}

Full

For all durations d,
There exists a run of duration d visiting ℓ_{priv}
$\Leftrightarrow$
There exists a run of duration d not visiting ℓ_{priv}

Three levels of ET-opacity

Existential ($\exists$)

private durations $\cap$ public durations $\neq \emptyset$

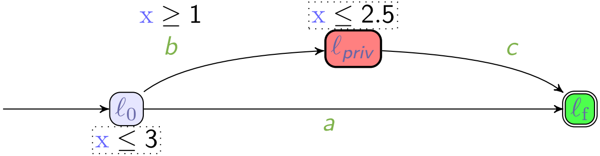
Weak

private durations $\subseteq$ public durations

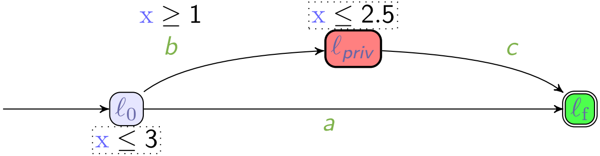
Full

private durations $=$ public durations

Example

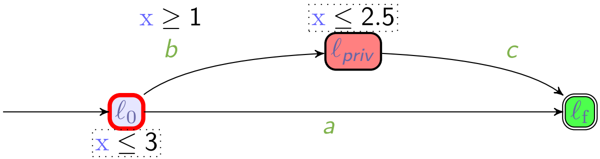


Example



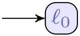
- There exist *(at least)* two runs of duration $d = 2$:

Example

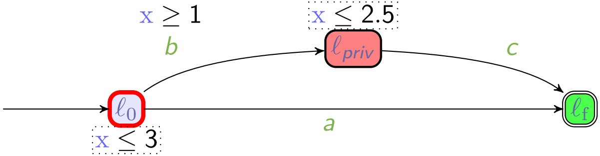


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

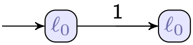


Example

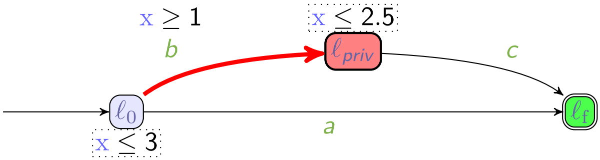


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

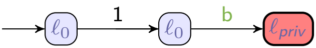


Example

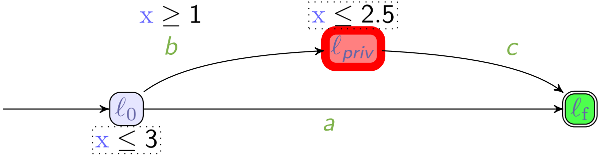


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

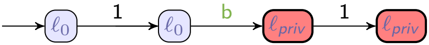


Example

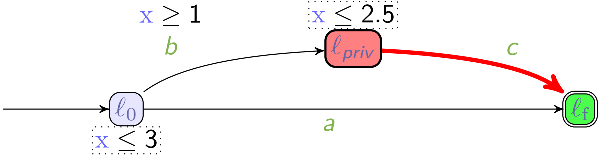


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

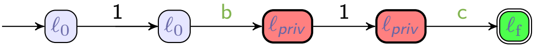


Example

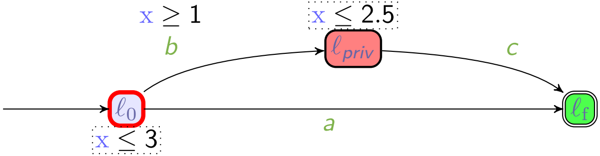


- There exist *(at least)* two runs of duration $d = 2$:

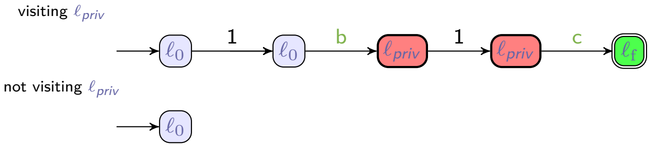
visiting ℓ_{priv}



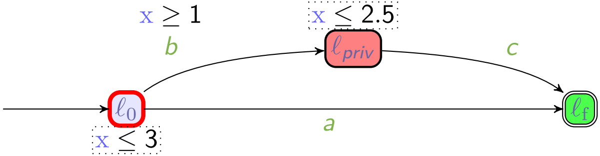
Example



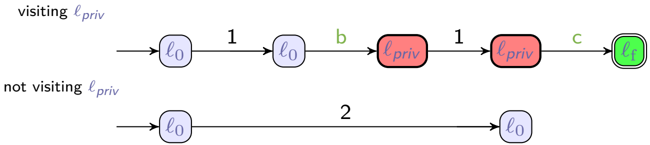
- There exist *(at least)* two runs of duration $d = 2$:



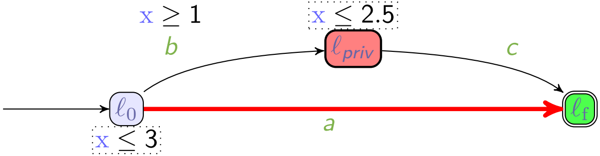
Example



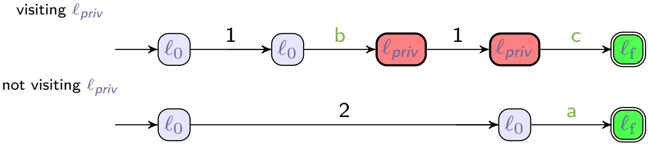
- There exist *(at least)* two runs of duration $d = 2$:



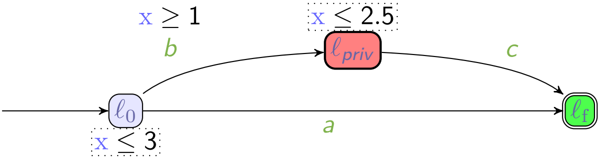
Example



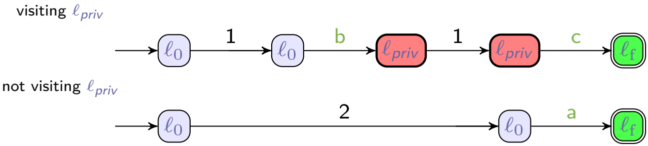
- There exist *(at least)* two runs of duration $d = 2$:



Example



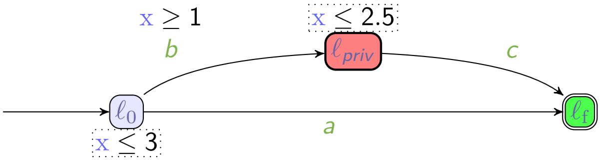
- There exist *(at least)* two runs of duration $d = 2$:



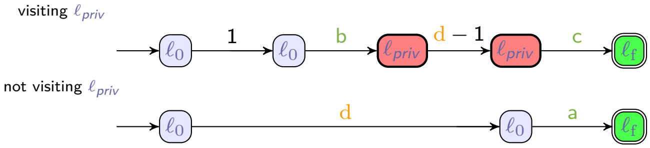
The system is ET-opaque for a duration $d = 2$

The system is ∃-ET-opaque

Example



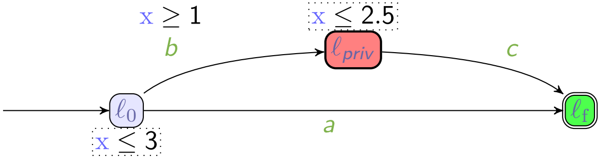
- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$:



The system is ET-opaque for all durations in $[1, 2.5]$

The system is ∃-ET-opaque

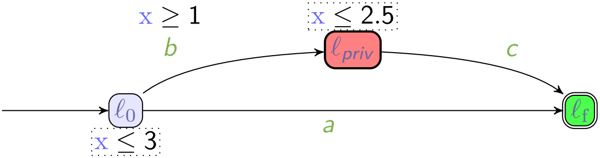
Example



- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is ∃-ET-opaque

Example

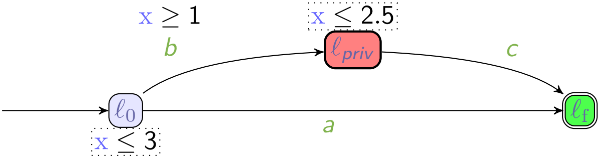


- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is ∃-ET-opaque

- private durations are $[1, 2.5]$
  public durations are $[0, 3]$

Example

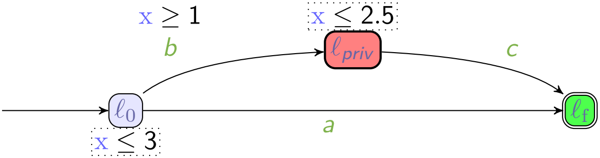


- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is ∃-ET-opaque

- private durations are $[1, 2.5]$
  public durations are $[0, 3]$
- private durations $\subseteq$ public durations

Example



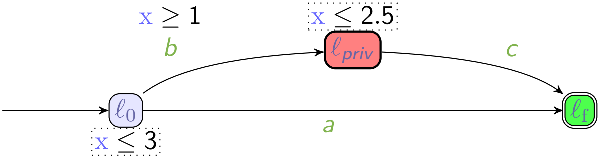
- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is ∃-ET-opaque

- private durations are $[1, 2.5]$
  public durations are $[0, 3]$
- private durations $\subseteq$ public durations

The system is weakly ET-opaque

Example



- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

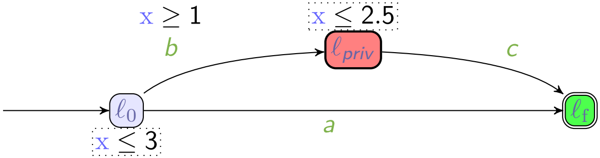
The system is ∃-ET-opaque

- private durations are $[1, 2.5]$
  public durations are $[0, 3]$
- private durations $\subseteq$ public durations

The system is weakly ET-opaque

- private durations $\neq$ public durations

Example



- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is ∃-ET-opaque

- private durations are $[1, 2.5]$
  public durations are $[0, 3]$
- private durations $\subseteq$ public durations

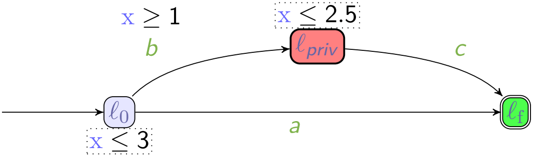
The system is weakly ET-opaque

- private durations $\neq$ public durations

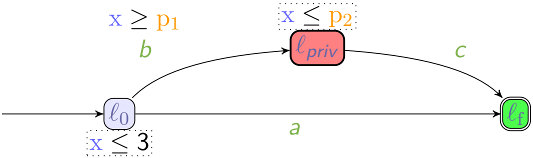
The system is *not* fully ET-opaque

Outline

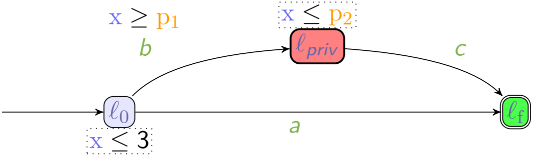
Example



Example

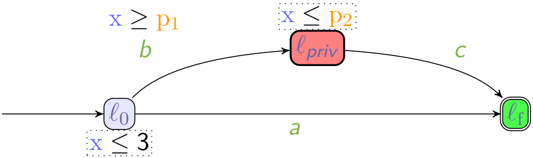


Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

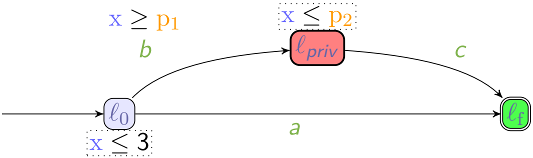
Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	**Private**	**Public**	**Answer**
$p_1 = 1 \land p_2 = 2.5$			
$\exists$			$\surd$
weak	$[1, 2.5]$	$[0, 3]$	$\surd$
full			$\times$

Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	**Private**	**Public**	**Answer**
$p_1 = 1 \wedge p_2 = 2.5$			
∃			$\surd$
weak	$[1, 2.5]$	$[0, 3]$	$\surd$
full			$\times$
$p_1 = 0 \wedge p_2 = 3$			
∃			$\surd$
weak	$[0, 3]$	$[0, 3]$	$\surd$
full			$\surd$

Two classes of parametric problems

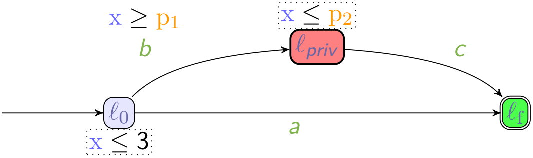
p-Emptiness problem

Decide the emptiness of the set of parameter valuations v
s. t. $\mathrm{v}(\mathcal{P})$ is ET-opaque

p-Synthesis problem

Synthesize the set of parameter valuations v
s. t. $\mathrm{v}(\mathcal{P})$ is ET-opaque

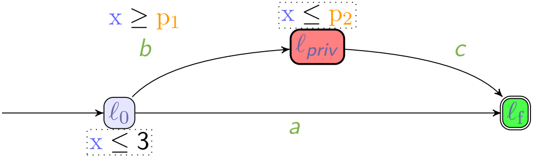
Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	∃	Weak	Full
p-Emptiness			
p-Synthesis			

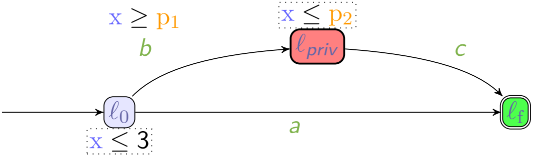
Example



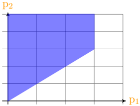
Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	$\exists$	**Weak**	**Full**
p-Emptiness	$\times(\exists v)$	$\times(\exists v)$	$\times(\exists v)$
p-Synthesis			

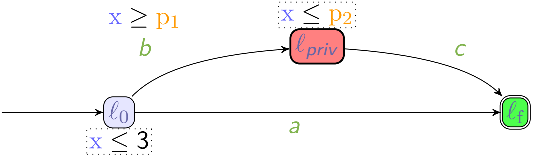
Example



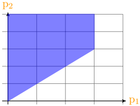
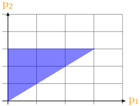
Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	$\exists$	Weak	Full
p-Emptiness	$\times(\exists v)$	$\times(\exists v)$	$\times(\exists v)$
p-Synthesis	$0 \leq p_1 \leq 3$ $\wedge\ p_1 \leq p_2$ 		

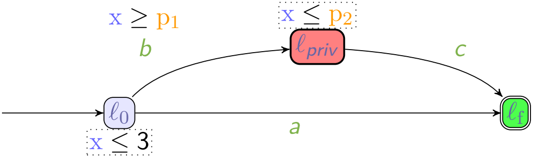
Example



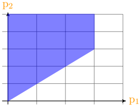
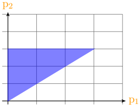
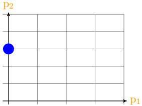
Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	$\exists$	Weak	Full
p-Emptiness	$\times(\exists v)$	$\times(\exists v)$	$\times(\exists v)$
p-Synthesis	$0 \leq p_1 \leq 3$ $\wedge\ p_1 \leq p_2$ 	$0 \leq p_1 \wedge p_2 \leq 3$ $\wedge\ p_1 \leq p_2$ 	

Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	$\exists$	Weak	Full
p-Emptiness	$\times(\exists v)$	$\times(\exists v)$	$\times(\exists v)$
p-Synthesis	$0 \leq p_1 \leq 3 \wedge p_1 \leq p_2$ 	$0 \leq p_1 \wedge p_2 \leq 3 \wedge p_1 \leq p_2$ 	$p_1 = 0 \wedge p_2 = 3$ 

Decidability results for ET-opacity

		∃-ET-opaque	weakly ET-opaque	fully ET-opaque
Decision	TA	√	√	√
p-emptiness	L/U-PTA	√	×	×
	PTA	×	×	×
p-synthesis	L/U-PTA	×	×	×
	PTA	×	×	×

- L/U-PTA (*Lower/Upper-PTA*): subclass of PTA where the parameters are partitioned into two sets (either compared to clocks as upperbound, or as lower bound) [Hun+02]
- *Proofs are based on the region automaton (for TAs) and by reduction from EF-emptiness (for PTAs).*

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *ACM TOSEM* (2022)

Decidability results for ET-opacity

		∃-ET-opaque	weakly ET-opaque	fully ET-opaque
Decision	TA	√	√	√
p-emptiness	L/U-PTA	√	×	×
	PTA	×	×	×
p-synthesis	L/U-PTA	×	×	×
	PTA	×	×	×

- L/U-PTA (*Lower/Upper-PTA*): subclass of PTA where the parameters are partitioned into two sets (either compared to clocks as upperbound, or as lower bound) [Hun+02]
- *Proofs are based on the region automaton (for TAs) and by reduction from EF-emptiness (for PTAs).*

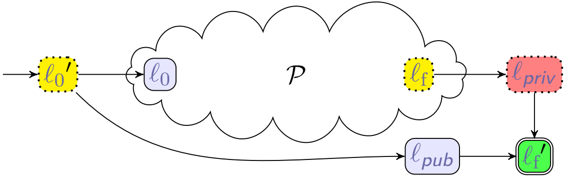
[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *ACM TOSEM* (2022)

ET-opacity synthesis is (very) difficult

Theorem (Undecidability of ∃-ET-opacity p-emptiness)

Given $\mathcal{P}$, the mere existence of a parameter valuation v s. t. $v(\mathcal{P})$ ∃-ET-opacity is undecidable.

Proof idea: reduction from reachability-emptiness for PTAs



Remark: L/U-PTA is a decidable subclass

Outline

Experiments: Computing ET-opaque durations

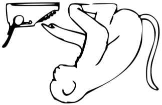
- Benchmark library + Library of Java programs [2]
  - Manually translated to PTAs
  - User-input variables → (non-timing) parameters
- Algorithms
  1. "Is the TA ET-opaque for all execution times?"
  2. "Synthesize parameter valuations and durations ensuring ET-opacity of a given PTA"

[2] https://github.com/Apogee-Research/STAC/

Experiments: Computing ET-opaque durations

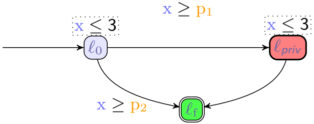
- Benchmark library + Library of Java programs [2]
  - Manually translated to PTAs
  - User-input variables → (non-timing) parameters
- Algorithms
  1. "Is the TA ET-opaque for all execution times?"
  2. "Synthesize parameter valuations and durations ensuring ET-opacity of a given PTA"

- Problems are undecidable → best-effort approach
- Algorithms based on parameter synthesis



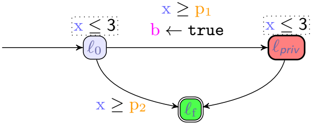
[2] https://github.com/Apogee-Research/STAC/

Our transformation of the PTA in 4 overlays



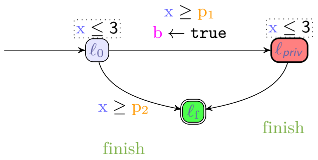
Our transformation of the PTA in 4 overlays

1. Add a Boolean flag b



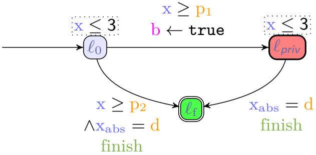
Our transformation of the PTA in 4 overlays

1. Add a Boolean flag b
2. Add a synchronization action finish



Our transformation of the PTA in 4 overlays

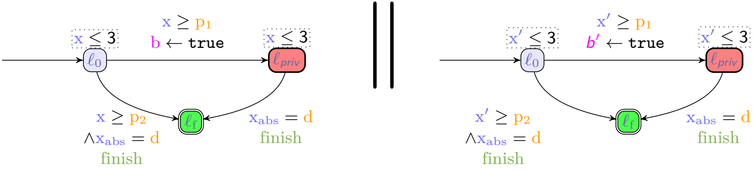
1. Add a Boolean flag b
2. Add a synchronization action finish
3. Measure the (parametric) duration to ℓ_f



Our transformation of the PTA in 4 overlays

1. Add a Boolean flag b
2. Add a synchronization action finish
3. Measure the (parametric) duration to ℓ_f
4. Perform self-composition

   (a synchronization on shared actions of the PTA with a copy of itself)

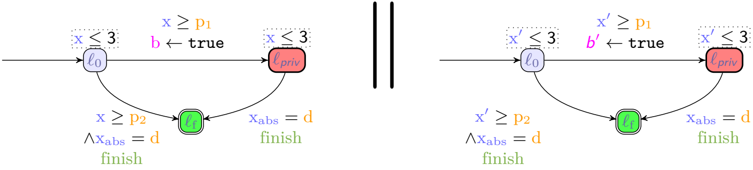


Applying reachability-synthesis

Synthesize all parameter valuations (including d) with a particular reachable state:

- ℓ_f with $b = \texttt{true}$
- ℓ_f with $b' = \texttt{false}$

$(\ell_f, b = \texttt{true})$ $(\ell_f, b' = \texttt{false})$



Formal proof of correctness: see [TOSEM22]

Outline

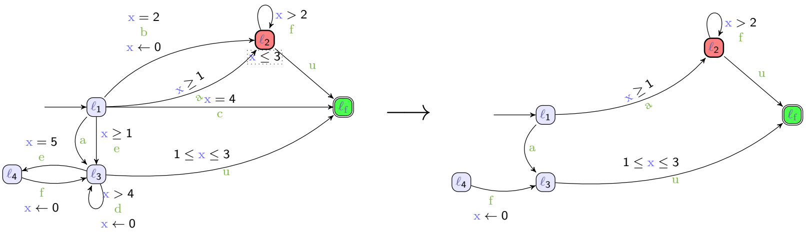
Extension 1: Expiring ET-opacity

- How to deal with outdated secrets?
  e. g., cache values, status of the memory, ...

Idea

The secret can expire: beyond a certain duration, knowing the secret is useless to the attacker (e. g., a cache value) [Amm+21]

Extension 2: Untimed control



- Restrict the behavior of the system to ensure ET-opacity
- Development of an open-source tool strategFTO ($\approx$ 1200 lines of code, Java)
  - Enumeration of transition sets

[FTSCS22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. "strategFTO: Untimed control for timed opacity". In: *FTSCS* (2022). ACM, 2022

Outline

Conclusion

Context: vulnerability by timing-attacks

- Attacker model: observability of the global execution time
- Goal: avoid leaking information on whether some discrete state has been visited

Several problems studied for timed automata

- ☺ Mostly decidable

Extension to parametric timed automata

- ☹ Quickly undecidable
- ☺ One procedure for one synthesis problem
- Toolkit: IMITATOR
- Benchmarks: concurrent systems and Java programs

Perspectives

Theoretical perspectives

- Existential version of expiring ET-opacity
- Δ-synthesis for full expiring ET-opacity

Algorihtmic perspectives

- Synthesis for weak and full ET-opacity
- Synthesis for expiring problems

Automatic translation of programs to PTAs

- Our translation required non-trivial creativity

  $\rightarrow$ Preliminary translation with Petri nets including cache system

References I

[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *TCS* 126 (Apr. 1994).

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC* (1993). ACM, 1993.

[Amm+21] Ikhlass Ammar, Yamen El Touati, Moez Yeddes, and John Mullins. "Bounded opacity for timed systems". In: *Journal of Information Security and Applications* 61 (Sept. 2021).

[AS19] Étienne André and Jun Sun. "Parametric Timed Model Checking for Guaranteeing Timed Opacity". In: *ATVA* (2019). LNCS. Springer, 2019.

[FTSCS22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. "strategFTO: Untimed control for timed opacity". In: *FTSCS* (2022). ACM, 2022.

References II

[Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. "Linear parametric model checking of timed automata". In: *Journal of Logic and Algebraic Programming* 52-53 (2002).

[ICECCS23] Étienne André, Engel Lefaucheux, and Dylan Marinho. "Expiring opacity problems in parametric timed automata". In: *ICECCS* (2023). To appear. Springer, 2023.

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *ACM TOSEM* 31 (2022).

Licensing

Source of the graphics used I

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

Title: Piratey, vector version
Author: Gustavb
Source: https://commons.wikimedia.org/wiki/File:Piratey,_vector_version.svg
License: CC by-sa

Title: Expired
Author: RRZEicons
Source: https://commons.wikimedia.org/wiki/File:Expired.svg
License: CC by-sa

License of this document

Authors: **Étienne André** and **Dylan Marinho**