Parametric Timed Model Checking for Guaranteeing Timed Opacity

Étienne André\textsuperscript{1,2,3} and Sun Jun\textsuperscript{4}

\textsuperscript{1} Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
\textsuperscript{2} National Institute of Informatics, Japan
\textsuperscript{3} JFLI, UMI CNRS, Tokyo, Japan
\textsuperscript{4} Singapore Management University

Supported by the ANR national research program PACS (ANR-14-CE28-0002), and by JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603).
Context: side-channel attacks

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

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

\[^{1} \text{http://home.xnet.com/~warinner/pizzacites.html} \]

Étienne André

PTA for Guaranteeing Timed Opacity

29 October 2019
Context: side-channel attacks

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

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

---

\(^1\) [http://home.xnet.com/~warinner/pizzacites.html](http://home.xnet.com/~warinner/pizzacites.html)
Context: 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

Non-trivial problem
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

```python
# 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
```

Execution time:

```
pwd  ch o u d o u f u
attempt  c h e e s e
```

Problem: The execution time is proportional to the number of consecutive correct characters from the beginning of `attempt`.
A simple example of timing attack

```plaintext
# 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
```

Execution time: $\epsilon$

Problem: The execution time is proportional to the number of consecutive correct characters from the beginning of attempt.
A simple example of timing attack

```plaintext
# 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
```

Execution time: $\epsilon + \epsilon$
A simple example of timing attack

```python
# 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
```

Execution time: $\epsilon + \epsilon + \epsilon$
A simple example of timing attack

```java
# 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
```

Example:

- `pwd`: choose dodo fu
- `attempt`: choose esse

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

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

1. Problems
2. Timed automata
3. Timed-opacity computation
4. Timed-opacity synthesis
5. Experiments
6. Conclusion and perspectives
Informal problems

Question: can we exhibit secure execution times?

Time-opacity computation

Exhibit execution times for which it is not only possible to infer information on the internal behavior.
Informal problems

Question: can we exhibit secure execution times?

Time-opacity computation

Exhibit execution times for which it is not only possible to infer information on the internal behavior

Further question: can we also tune internal timing constants to make the system resisting to timing attacks?

Time-opacity synthesis

Exhibit execution times and internal timing constants for which it is not only possible to infer information on the internal behavior
Outline

1 Problems

2 Timed automata

3 Timed-opacity computation

4 Timed-opacity synthesis

5 Experiments

6 Conclusion and perspectives
Timed automaton (TA)

- Finite state automaton (sets of locations)

---

Etienne André

PTA for Guaranteeing Timed Opacity

29 October 2019
Timed automaton (TA)

- Finite state automaton (sets of locations and actions)

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
Timed automaton (TA)

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

---

Press:

- Coffee!
- Press?
- Cup!

States:

- Idle
- Adding sugar
- Delivering coffee
Timed automaton (TA)

- 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

---

Diagram:

- Green state: idle
- Blue state: adding sugar
- Red state: delivering coffee
- Initial state: press?
- Transition conditions:
  - $y \leq 5$
  - $y \leq 8$
Timed automaton (TA)

- 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

![Diagram of a timed automaton](image)

```plaintext
y \leq 5
```

```
y = 8
cup!
```

```
x \geq 1
press?
```

```
y = 5
cup!
```

```
y \leq 8
delivering coffee
```

```
idle
```

```
adding sugar
```

```
press?
```
Timed automaton (TA)

- 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

$y \leq 5$
$y = 8$
$y = 5$
$x \geq 1$
$x := 0$
$y := 0$
$y := 0$
Press? press?
Press? press?
Cup! cup!
Coffee! coffee!

idle
adding sugar
delivering coffee
Concrete semantics of timed automata

- **Concrete state** of a TA: pair \((\ell, w)\), where
  - \(\ell\) is a location,
  - \(w\) is a valuation of each clock

Example: \((\square, (x=1.2, y=3.7))\)

- **Concrete run**: alternating sequence of concrete states and actions or time elapse
The most critical system: The coffee machine

Example of concrete run for the coffee machine

0
0
x =
y =
x ≥ 1
press?
y ≤ 5
x := 0
press?
y = 8
coffee!
y = 5
cup!
y ≤ 8
cup!  

idle
adding sugar
delivering coffee
The most critical system: The coffee machine

Example of concrete run for the coffee machine

- Coffee with 2 doses of sugar

\[ x = 0 \\
\]
The most critical system: The coffee machine

Example of concrete run for the coffee machine

- Coffee with 2 doses of sugar
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th>$x$</th>
<th>0</th>
<th>0</th>
<th>1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
</tr>
</tbody>
</table>
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>0</th>
<th>1.5</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

\[\begin{align*}
x &= 0 & 0 & 1.5 & 0 & 2.7 \\
y &= 0 & 0 & 1.5 & 1.5 & 4.2
\end{align*}\]
The most critical system: The coffee machine

```latex
\begin{align*}
y &= 8 \\
\text{coffee!}
\end{align*}
```

- idle
- adding sugar
- delivering coffee

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

```
x = 0 \quad 0 \quad 1.5 \quad 0 \quad 2.7 \quad 0 \\
y = 0 \quad 0 \quad 1.5 \quad 1.5 \quad 4.2 \quad 4.2
```

Étienne André

PTA for Guaranteeing Timed Opacity
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1.5</td>
</tr>
<tr>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2.7</td>
</tr>
<tr>
<td>2.7</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.8</td>
</tr>
<tr>
<td>0.8</td>
<td>5</td>
</tr>
</tbody>
</table>
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>0</td>
<td>2.7</td>
<td>0.8</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>1.5</td>
<td>4.2</td>
<td>4.2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar
Outline

1. Problems
2. Timed automata
3. Timed-opacity computation
4. Timed-opacity synthesis
5. Experiments
6. Conclusion and perspectives
Formalization

Hypotheses:

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

Definition (timed opacity)

The system is opaque w.r.t. $ℓ_{priv}$ on the way to $ℓ_f$ if

1. for any run to $ℓ_f$ of duration $d$ passing by $ℓ_{priv}$, there exists another run to $ℓ_f$ of duration $d$ not passing by $ℓ_{priv}$, and
2. conversely
Problem 1: timed-opacity computation

Timed-opacity computation problem

Find durations \( d \) (“execution times”) of runs from \( l_0 \) to \( l_f \) such that the system is opaque w.r.t. \( l_{priv} \) on the way to \( l_f \)
Problem 1: timed-opacity computation

Timed-opacity computation problem

Find durations $d$ ("execution times") of runs from $l_0$ to $l_f$ such that the system is opaque w.r.t. $l_{priv}$ on the way to $l_f$

Example:
Problem 1: timed-opacity computation

Timed-opacity computation problem

Find durations $d$ ("execution times") of runs from $\ell_0$ to $\ell_f$ such that the system is opaque w.r.t. $\ell_{priv}$ on the way to $\ell_f$

Example:

Here, only computations with durations $d \in [2, 3]$ are opaque
Problem 1b: timed-opacity

Definition (Timed-opacity)

A system is **timed-opaque** if the answer to the timed-opacity computation problem is the set of all possible execution times for this system.

Intuition: is a system timed-opaque for all its execution times?
Problem 1b: timed-opacity

Definition (Timed-opacity)

A system is timed-opaque if the answer to the timed-opacity computation problem is the set of all possible execution times for this system.

Intuition: is a system timed-opaque for all its execution times?
**Problem 1b: timed-opacity**

**Definition (Timed-opacity)**

A system is **timed-opaque** if the answer to the timed-opacity computation problem is the set of all possible execution times for this system.

Intuition: is a system timed-opaque for all its execution times?

Here, only computations with durations $d \in [2, 3]$ are **opaque**

All execution times: $[1, 3]$ ∼ **not timed-opaque**
Theorem (Computability of timed-opacity)

The answer to the timed-opacity computation problem can be effectively be computed in the form of a finite union of intervals

Proof: based on the region graph (see paper)

Exact complexity: unproved (EXPSACE upper bound proved, but exponential hardness seems likely)

Remark: to be put in perspective with [Cassez, 2009]
- undecidability for a less expressive class, for a stronger notion of opacity
Outline

1. Problems
2. Timed automata
3. Timed-opacity computation
4. Timed-opacity synthesis
5. Experiments
6. Conclusion and perspectives
Towards a more abstract framework...

Problems

- Can we tune some timing constants to guarantee opacity?

- Verification for one set of constants does not usually guarantee the correctness for other values

- Robustness [Bouyer et al., 2013]: What happens if 50 is implemented with 49.99?
Towards a more abstract framework...

Problems

■ Can we tune some timing constants to guarantee opacity?

■ Verification for one set of constants does not usually guarantee the correctness for other values

■ Robustness [Bouyer et al., 2013]: What happens if 50 is implemented with 49.99?

A solution:

■ Parameter synthesis
  ■ Consider that timing constants are unknown constants (parameters)
Parametric Timed Automaton (PTA)

Timed automaton (sets of locations, actions and clocks)

- $y \leq 5$
- $y = 8$
- $x = 0$
- $y = 0$
- $y = 5$
- $x \geq 1$
- coffee!
- cup!
Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks) augmented with a set $P$ of parameters [Alur et al., 1993]
- Unknown constants compared to a clock in guards and invariants

$$y = p_3$$
coffee!

$$y \leq p_2$$
press?

$$x \geq p_1$$
cup!

$y := 0$

$y := 0$

$x := 0$

$x := 0$
Notation: Valuation of a PTA

Given a PTA $\mathcal{A}$ and a parameter valuation $v$, we denote by $v(\mathcal{A})$ the (non-parametric) timed automaton where each parameter $p$ is valuated by $v(p)$. 

\[
\begin{align*}
\text{press?} & : x := 0 \\
\text{cup!} & : x \geq p_1 \\
\text{coffee!} & : x := 0 \\
\text{cup!} & : y = 5 \\
\text{coffee!} & : y = 8
\end{align*}
\]

\[
\begin{align*}
\text{press?} & : x := 0 \\
\text{cup!} & : y = 8 \\
\text{coffee!} & : y = 5
\end{align*}
\]
Notation: Valuation of a PTA

Given a PTA $\mathcal{A}$ and a parameter valuation $v$, we denote by $v(\mathcal{A})$ the (non-parametric) timed automaton where each parameter $p$ is valuated by $v(p)$

$$v:\begin{cases}p_1 & \rightarrow & 1 \\ p_2 & \rightarrow & 5 \\ p_3 & \rightarrow & 8 \end{cases}$$

with $v$:
Problem 2: timed-opacity synthesis

Timed-opacity synthesis problem

Find parameter valuations $v$ and durations $d$ ("execution times") of runs of $v(A)$ from $\ell_0$ to $\ell_f$ such that the system is opaque w.r.t. $\ell_{priv}$ on the way to $\ell_f$.
Problem 2: timed-opacity synthesis

Timed-opacity synthesis problem

Find parameter valuations \( v \) and durations \( d \) (“execution times”) of runs of \( v(A) \) from \( \ell_0 \) to \( \ell_f \) such that the system is opaque w.r.t. \( \ell_{\text{priv}} \) on the way to \( \ell_f \).

Example:

\[
\begin{align*}
\ell_0 & \quad x \leq 3 \\
\ell_f & \quad x \geq p_2 \\
\ell_{\text{priv}} & \quad x \leq 3 \\
\end{align*}
\]

\[
\begin{align*}
\ell_0 & \quad x \geq p_1 \\
\ell_{\text{priv}} & \quad x \leq 3 \\
\end{align*}
\]
Problem 2: timed-opacity synthesis

Timed-opacity synthesis problem

Find parameter valuations $v$ and durations $d$ (“execution times”) of runs of $v(A)$ from $l_0$ to $l_f$ such that the system is opaque w.r.t. $l_{priv}$ on the way to $l_f$

Example:

$$x \leq 3$$

$$x \geq p_1$$

$$x \geq p_2$$

Expected result:

$$p_1 \leq 3 \land p_2 \leq 3 \land d \in [p_2, 3]$$
Outline

1 Problems

2 Timed automata

3 Timed-opacity computation

4 Timed-opacity synthesis
   - Theory: undecidability
   - A practical approach

5 Experiments

6 Conclusion and perspectives
Timed-opacity synthesis is (very) difficult

Theorem (Undecidability of timed-opacity-emptiness)

The mere existence of a parameter valuation for which there exists a duration for which timed-opacity is achieved is undecidable.

Proof idea: reduction from reachability-emptiness for PTAs [Alur et al., 1993]

Remark: decidable subclass

(see Theorem 1 in the paper)
Timed-opacity synthesis is (very) difficult

Theorem (Undecidability of timed-opacity-emptiness)

The mere existence of a parameter valuation for which there exists a duration for which timed-opacity is achieved is undecidable.

Proof idea: reduction from reachability-emptiness for PTAs [Alur et al., 1993]

Remark: decidable subclass (see Theorem 1 in the paper)

In the following, we adopt a “best-effort” approach

- Approach not guaranteed to terminate in theory
Outline

1 Problems
2 Timed automata
3 Timed-opacity computation
4 Timed-opacity synthesis
   ▪ Theory: undecidability
   ▪ A practical approach
5 Experiments
6 Conclusion and perspectives
Reducing timed-opacity synthesis to reachability synthesis

Big picture:

- Formalism: parametric timed automata
- Our approach:
  1. Perform a (mild) transformation of the PTA
  2. Perform self-composition
  3. Apply parametric timed model checking (reachability-synthesis)
- Tool support: IMITATOR

[André et al., 2012]
Our transformation of the PTA in 4 overlays

\[
\begin{align*}
\ell_0 & \quad x \leq 3 \\
\ell & \quad x \geq p_2 \\
\ell_{priv} & \quad x \leq 3 \\
\end{align*}
\]
Our transformation of the PTA in 4 overlays

1. Add a Boolean flag $b$ to remember whether $l_{priv}$ was visited

\[\begin{array}{l}
x \leq 3 \\
\ell_0 \\
x \geq p_1 \\
\ell_f \\
x \leq 3 \\
\ell_{priv} \\
b \leftarrow \text{true}
\end{array}\]
Our transformation of the PTA in 4 overlays

1. Add a Boolean flag \( b \) to remember whether \( \ell_{\text{priv}} \) was visited
2. Add a synchronization action \texttt{finish} on any transition to \( \ell_f \)
Our transformation of the PTA in 4 overlays

1. Add a Boolean flag $b$ to remember whether $l_{priv}$ was visited
2. Add a synchronization action $\text{finish}$ on any transition to $l_f$
3. Measure the (parametric) duration to $l_f$ thanks to a new clock $x_{abs}$ and a new parameter $d$
Our transformation of the PTA in 4 overlays

1. Add a Boolean flag $b$ to remember whether $\ell_{priv}$ was visited
2. Add a synchronization action `finish` on any transition to $\ell_f$
3. Measure the (parametric) duration to $\ell_f$ thanks to a new clock $x_{abs}$ and a new parameter $d$
4. Perform self-composition (i.e., a synchronization on shared actions of the PTA with a copy of itself)
timed model checking

A model of the system

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

Yes

No

Counterexample
Parametric timed model checking

A model of the system

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

Yes if...

\[ 2\text{delay} > \text{period} \wedge \text{period} < 20.46 \]
Applying reachability-synthesis

We then synthesize all parameter valuations (including $d$) for which the following discrete state is reachable:

- the original automaton is in $\ell_f$ with $b = \text{true}$
- the copy automaton is in $\ell_f$ with $b' = \text{false}$

Intuition: for the same duration (thanks to the synchroniation on \textit{finish}), we can reach $\ell_f$ "both" with passing by $\ell_{\text{priv}}$ (i.e., $b = \text{true}$) or without (i.e., $b = \text{false}$)
Applying reachability-synthesis

We then synthesize all parameter valuations (including $d$) for which the following discrete state is reachable:

- the original automaton is in $\ell_f$ with $b = \text{true}$
- the copy automaton is in $\ell_f$ with $b' = \text{false}$

Intuition:

- for the same duration (thanks to the synchronisation on $\text{finish}$), we can reach $\ell_f$ “both” with passing by $\ell_{\text{priv}}$ (i.e., $b = \text{true}$) or without (i.e., $b = \text{false}$)
Outline

1 Problems

2 Timed automata

3 Timed-opacity computation

4 Timed-opacity synthesis

5 Experiments

6 Conclusion and perspectives
Experimental environment

Algorithms

1. Timed-opacity: “for a non-parametric TA, is the TA opaque for all execution times?”

2. Timed-opacity synthesis: “for a PTA, synthesize parameter valuations and execution times ensuring timed opacity”

Benchmarks

- Common PTA benchmarks
- Library of Java programs
  - Manually translated to PTAs
  - User-input variables translated to (non-timing) parameters (supported by IMITATOR)

See experiments at doi.org/10.5281/zenodo.3251141 and imitator.fr/static/ATVA19/
Outline

1 Problems

2 Timed automata

3 Timed-opacity computation

4 Timed-opacity synthesis

5 Experiments
   ■ IMITATOR in a nutshell
   ■ Non-parametric timed-opacity computation
   ■ Parametric timed-opacity synthesis

6 Conclusion and perspectives
**IMITATOR**

- A tool for modeling and verifying **timed concurrent systems** with unknown constants modeled with **parametric timed automata**
  - Communication through (strong) broadcast synchronization
  - Rational-valued shared discrete variables
  - **Stopwatches**, to model schedulability problems with preemption

- **Synthesis algorithms**
  - (non-Zeno) parametric model checking (using a subset of **TCTL**)
  - Language and trace preservation, and robustness analysis
  - Parametric deadlock-freeness checking
IMITATOR

Under continuous development since 2008

A library of benchmarks

- Communication protocols
- Schedulability problems
- Asynchronous circuits
- …and more

Free and open source software: Available under the GNU-GPL license

[André et al., FM’12]
IMITATOR

Under continuous development since 2008

A library of benchmarks

- Communication protocols
- Schedulability problems
- Asynchronous circuits
- …and more

Free and open source software: Available under the GNU-GPL license

Try it!

www.imitator.fr
Outline

1. Problems
2. Timed automata
3. Timed-opacity computation
4. Timed-opacity synthesis
5. Experiments
   - IMITATOR in a nutshell
   - Non-parametric timed-opacity computation
   - Parametric timed-opacity synthesis
6. Conclusion and perspectives
### Experiments: (non-parametric) timed opacity

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Transf. PTA</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 5 [Vasilikos et al., 2018]</td>
<td>1 1</td>
<td>2 3 3</td>
</tr>
<tr>
<td>Fig. 1b, [Gardey et al., 2007]</td>
<td>1 1</td>
<td>2 3 1</td>
</tr>
<tr>
<td>Fig. 2a, [Gardey et al., 2007]</td>
<td>1 1</td>
<td>2 3 1</td>
</tr>
<tr>
<td>Fig. 2b, [Gardey et al., 2007]</td>
<td>1 1</td>
<td>2 3 1</td>
</tr>
<tr>
<td>Web privacy problem [Benattar et al., 2015]</td>
<td>1 2</td>
<td>2 4 1</td>
</tr>
<tr>
<td>Coffee</td>
<td>1 2</td>
<td>2 5 1</td>
</tr>
<tr>
<td>Fischer-HSRV02</td>
<td>3 2</td>
<td>6 5 1</td>
</tr>
<tr>
<td>STAC:1:n</td>
<td></td>
<td>2 3 6</td>
</tr>
<tr>
<td>STAC:1:v</td>
<td></td>
<td>2 3 6</td>
</tr>
<tr>
<td>STAC:3:n</td>
<td></td>
<td>2 3 8</td>
</tr>
<tr>
<td>STAC:3:v</td>
<td></td>
<td>2 3 8</td>
</tr>
<tr>
<td>STAC:4:n</td>
<td></td>
<td>2 3 8</td>
</tr>
<tr>
<td>STAC:4:v</td>
<td></td>
<td>2 3 8</td>
</tr>
<tr>
<td>STAC:5:n</td>
<td></td>
<td>2 3 6</td>
</tr>
<tr>
<td>STAC:11A:v</td>
<td></td>
<td>2 3 8</td>
</tr>
<tr>
<td>STAC:11B:v</td>
<td></td>
<td>2 3 8</td>
</tr>
<tr>
<td>STAC:12c:v</td>
<td></td>
<td>2 3 8</td>
</tr>
<tr>
<td>STAC:12e:n</td>
<td></td>
<td>2 3 8</td>
</tr>
<tr>
<td>STAC:12e:v</td>
<td></td>
<td>2 3 8</td>
</tr>
<tr>
<td>STAC:14:n</td>
<td></td>
<td>2 3 8</td>
</tr>
</tbody>
</table>

× = not vulnerable; (✓) = vulnerable, can be repaired; √ = vulnerable, cannot be repaired
Outline

1. Problems
2. Timed automata
3. Timed-opacity computation
4. Timed-opacity synthesis
5. Experiments
   - IMITATOR in a nutshell
   - Non-parametric timed-opacity computation
   - Parametric timed-opacity synthesis
6. Conclusion and perspectives
Experiments: (parametric) timed-opacity synthesis

<table>
<thead>
<tr>
<th>Model</th>
<th>Transf. PTA</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fig. 5, [Vasilikos et al., 2018]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Fig. 1b, [Gardey et al., 2007]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Fig. 2, [Gardey et al., 2007]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Web privacy problem [Benattar et al., 2015]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Coffee</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Fischer-HSRV02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>STAC:3:v</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$K = \text{some valuations make the system non-vulnerable;} \quad T = \text{all valuations make the system non-vulnerable}$
Conclusion

Context: vulnerability by timing-attacks

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

Solution: parametric timed model checking

- Formalism: parametric timed automata
- Our approach:
  1. Perform a (mild) transformation of the PTA
  2. Perform self-composition
  3. Apply parametric timed model checking (reachability-synthesis)

- Toolkit: IMITATOR

- Benchmarks: concurrent systems and Java programs
Perspectives

- Theoretical open problems
  - Decidability of timed-opacity emptiness remains open for a clock case of U-PTAs or L-PTAs
  - [Bozzelli and La Torre, 2009]

- Automated translation of Java programs
  - Our translation required non-trivial creativity
  - How to automate it?
  - Finer grain needed for “untimed” instructions: probabilistic timings?

- Repairing a non-opaque system
  - “From PTA parameter tuning back to the original system”
  - In programs: using Wait or Sleep
  - Preliminary ideas in the paper
We hire!

- **What**: Project on **quantitative formal methods + security** *(2020-2023)*

- **Where**: France *(Nancy / Nantes)*, Singapore

- **Who**: Master students, PhD, post-docs
  
  ...starting anytime!
Bibliography
References I


References II

Robustness in timed automata.
Invited paper.

Decision problems for lower/upper bound parametric timed automata.

The dark side of timed opacity.

Non-interference control synthesis for security timed automata.

Secure information release in timed automata.
Licensing
Source of the graphics used I

Title: Power attack
Author: Audriusa
Source: https://commons.wikimedia.org/wiki/File:Power_attack.png
License: GNU GPL

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
License of this document

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

(\LaTeX{} source available on demand)

Author: Étienne André

https://creativecommons.org/licenses/by-sa/4.0/