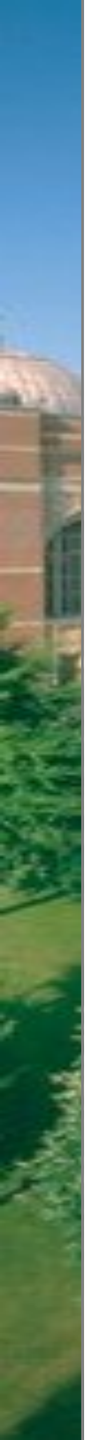


Automated Formal Analysis of Side-Channel Attacks on Probabilistic Systems

Chris Novakovic Dave Parker

University of Birmingham

ESORICS 2019, Luxembourg, September 2019

Motivation

- Side-channel attacks
  - potential leakage of secret data via e.g. time/power info
  - side channel attacks increasingly viable

- Probabilistic systems
  - information leakage is naturally expressed probabilistically
  - security systems often employ randomisation or operate in uncertain environments

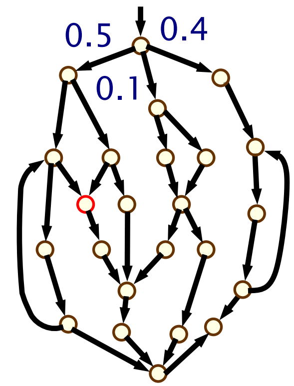
- Automated verification techniques
  - build on recent advances in techniques & tools (probabilistic model checking & PRISM)
  - meaningfully quantify the severity of potential attacks
  - synthesise optimal attacks and analyse defences/trade-offs

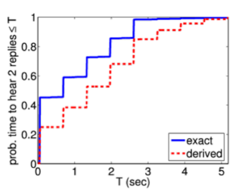
Overview

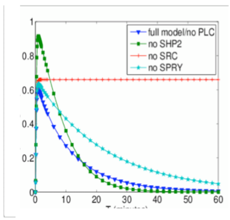
- Probabilistic model checking
  - discrete-time Markov chains
  - partially observable Markov decision processes (POMDPs)
- The SCH-IMP language
- Automated side channel analysis
  - via probabilistic model checking on POMDPs
- Tool support
- Examples and results

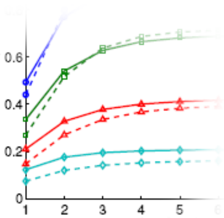
Probabilistic model checking

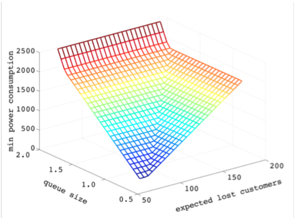
- Probabilistic model checking
  - formal construction/analysis of probabilistic models
  - “correctness” properties expressed in temporal logic
  - e.g. trigger $\rightarrow P_{\geq 0.999}$ [$F^{\leq 20}$ deploy]
  - mix of exhaustive & numerical/quantitative reasoning









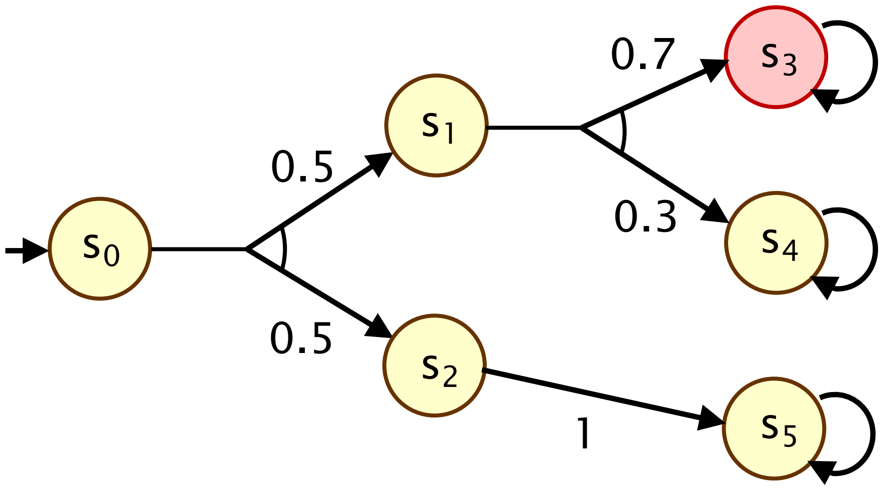


- Trends and advances
  - improvement in scalability to larger models
  - increasingly expressive/powerful model classes
  - from verification problems to control/synthesis problems
  - (i.e., synthesis of optimal strategies/policies)

Probabilistic models

- Models used here:
  - discrete-time Markov chains
  - partially observable Markov decision processes (POMDPs)

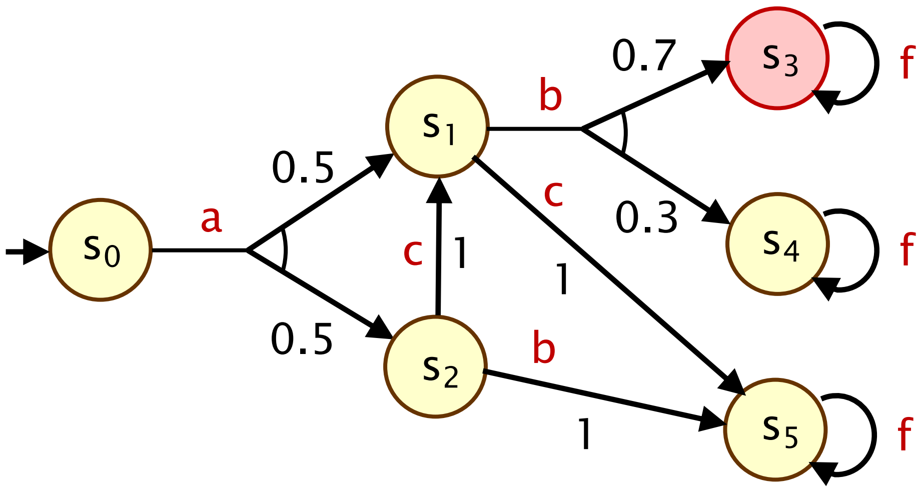
- Discrete-time Markov chains



  - e.g. what is the probability of reaching s_3?

Probabilistic models: MDPs

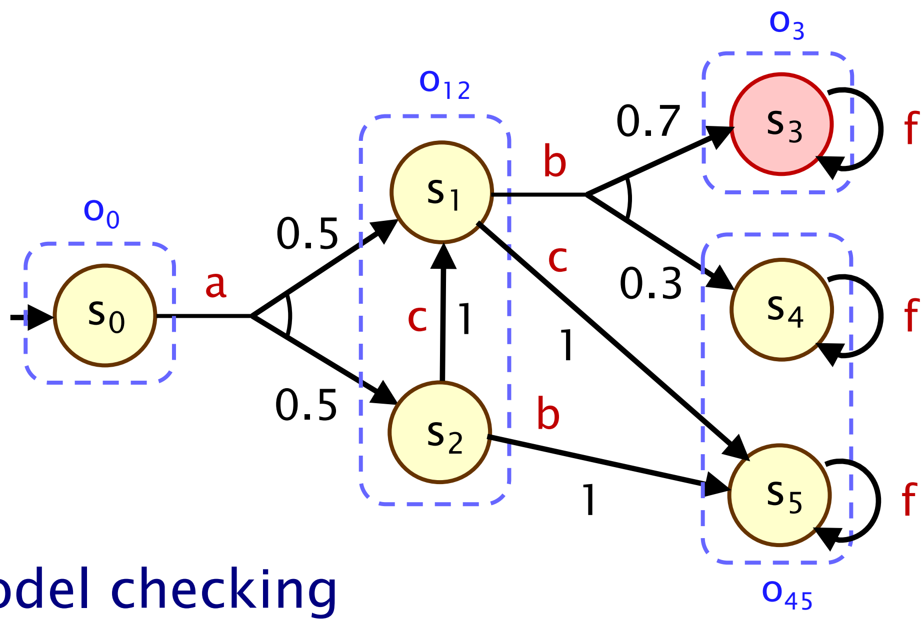
- Markov decision processes (MDPs)
  - mix nondeterministic and probabilistic choice
  - strategies (or policies) resolve actions based on history



- MDP model checking
  - e.g. what is the maximum probability of reaching s_3 achievable by any strategy?

Probabilistic models: POMDPs

- Partially observable Markov decision processes (POMDPs)
  - an observation function limits what a strategy can observe
  - strategies make the same choice in equivalent states



- POMDP model checking
  - basic verification problems are undecidable
  - but techniques exist to approximate optimal strategies
  - and tool support is now available, e.g. PRISM-pomdps

The SCH-IMP language

- SCH-IMP: Simple imperative programming language, with:
  - secret variables, probabilistic assignment, resource usage
  - extension of CH-IMP [CSF'13] for information leakage
- Key language components:
  - variables (finite-ranging)
    - assigned via discrete probability distributions
    - either "initial" (secret) or "regular" variables
  - output statements + functions + basic control flow

The SCH-IMP language

- SCH-IMP: Simple imperative programming language, with:
  - secret variables, probabilistic assignment, resource usage
  - extension of CH-IMP [CSF'13] for information leakage
- Key language components:
  - variables (finite-ranging)
    - assigned probabilistically (via discrete probability distributions)
    - either "initial" (secret) or "regular" variables
  - output statements + functions + basic control flow

$$
\begin{aligned}
\mathbb{P} \;::=\;& [\text{initial } V := \rho;]^* \\
& [\text{new } V := \rho;]^* \\
& [\text{function } F([V]^*) \;\{\; C;\; [\text{output } [A]^+;]^? \text{ return } \};]^+ \\
& F([A]^*);\ \text{end} \\
C \;::=\;& \text{skip} \;\mid\; \text{new } V := \rho \;\mid\; V := \rho \;\mid\; F([A]^*) \\
\mid\;& \text{if } (B) \;\{\; C \;\}\; [\text{else } \{\; C \;\}]^? \\
\mid\;& \text{while } (B) \;\{\; C \;\} \;\mid\; C;\; C
\end{aligned}
$$

SCH-IMP example

- Simple example program

```
initial i := { 0→¼, 1→¼, 2→¼, 3→¼ };
function f(x) {
        new o := 1;
        if (x>0) { o := x/x };
        output o;
        return
};
f(i);
end
```

- Note:
  - secret/non-secret variables declared with **initial** and **new**
  - variable o is always set to 1, so no leakage via **output**

SCH-IMP: Resources

- Resource usage
  - we focus here on just two: time and power consumption
  - defined at the level of functions, not individual commands
  - a resource function gives, for a subset of program functions, a mapping from function arguments to discrete probability distributions over time/power usage

```
initial i := { 0→¼, 1→¼, 2→¼, 3→¼ };
function f(x) {
      new o := 1;
      if (x>0) { o := x/x };
      output o;
      return
};
f(i);
end
```

```
f → {
      (0) → { (5,7) → ½, (6,7) → ½ },
      (_) → { (6,7) → ½, (7,7) → ½ }
}
```

SCH-IMP semantics

- Semantics for execution of a SCH-IMP program
  - defined as a discrete-time Markov chain
  - see the paper for a formal definition
- A SCH-IMP **state** is a tuple (F, I, t, p, Δ)
  - F is the current command stack (+ associated variable values)
  - $I : \mathrm{Var} \rightarrow \mathrm{Val}$ is the initial variables and their values
  - $t : \mathbb{N}$ is the total time elapsed so far
  - $p : \mathbb{N}$ is the power consumed so far
  - $\Delta : \mathbb{N} \rightarrow \mathbb{N} \times \mathrm{seq}(\mathrm{Val})$ is an observation function defining, for certain time points, the cumulative power consumed and values that were output from the program

SCH-IMP semantics: Example

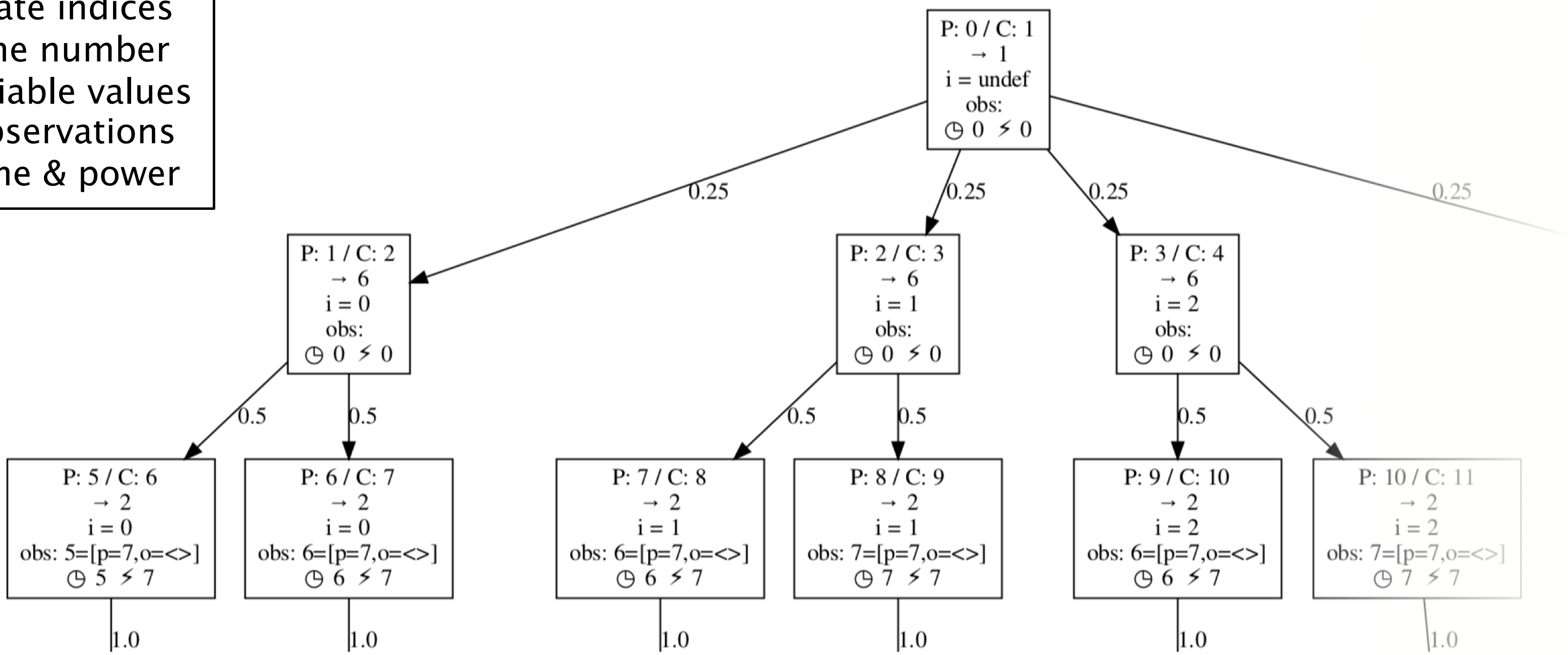
- Initial fragment of semantics for example

```
1: initial i := { 0→¼, 1→¼, 2→¼, 3→¼ };
   function f(x) {
      ...
5: };
6: f(i);
7: end
```

```
f → {
      (0) → { (5,7) → ½, (6,7) → ½ },
      (_) → { (6,7) → ½, (7,7) → ½ }
}
```

Key:

state indices
line number
variable values
observations
time & power



Side channel analysis

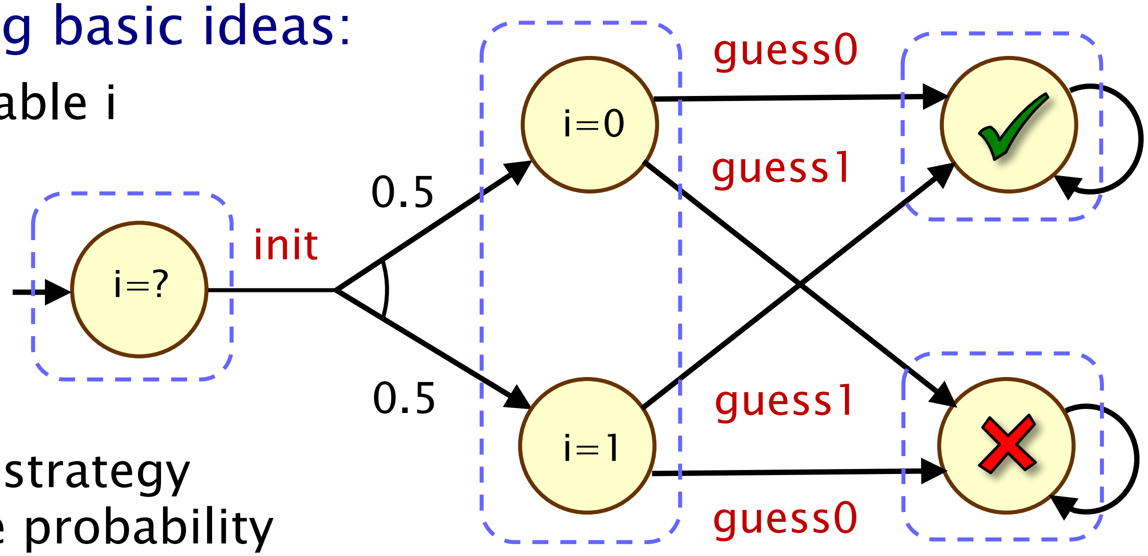
- Attack model
  - attacker has full access to program source code
  - sees program outputs & power usage at fixed time points

- 3-phase process
  - systematic construction of Markov chain model
    - following SCH-IMP semantics (including optimisations)
  - conversion to POMDP model
    - encoding (side channel) attacker knowledge and choices
  - solution of POMDP
    - construct optimal strategy for maximising info leakage

POMDP construction

- POMDP extends Markov chain model of program
  - observation function hides program internals
  - attacker “guess” actions added
  - (for now, these occur on termination)

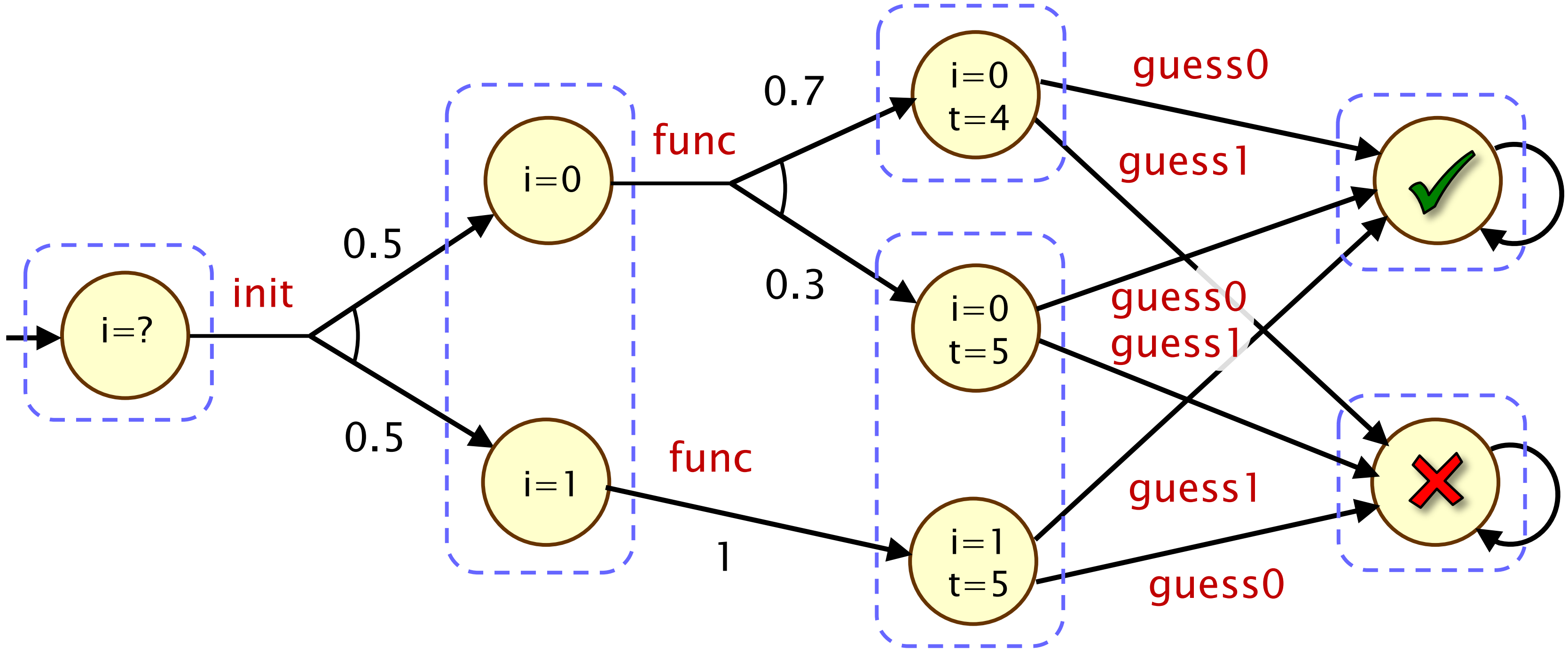
- Example showing basic ideas:
  - for secret variable i



  - which POMDP strategy maximises the probability of reaching the success state?

POMDP construction

- Extended example
  - incorporating time observations (t)



  - optimal strategy can now use time observations

Tool support

- Prototype tool, including
  - language parser and processor
  - model construction via PRISM (model generator API)
  - POMDP model checking via PRISM-pomdps
  - source/binaries/examples available from: www.cs.bham.ac.uk/research/projects/schimp/

- Sample of performance results for experiments

	States		Result		
Example	DTMC	POMDP	p	Error	Time (min)
DC-net	93333	20003	0.527	0.017	63
Uni. network: $\mathsf{h1} = 3$, $\mathsf{m} \leq 15$	142547	36009	0.991	0.000	13
Uni. network: $\mathsf{h1} = 10$, $\mathsf{m} \leq 5$	43461	12403	0.997	0.003	1
Square-and-multiply: naive	60749	27003	0.964	0.000	16

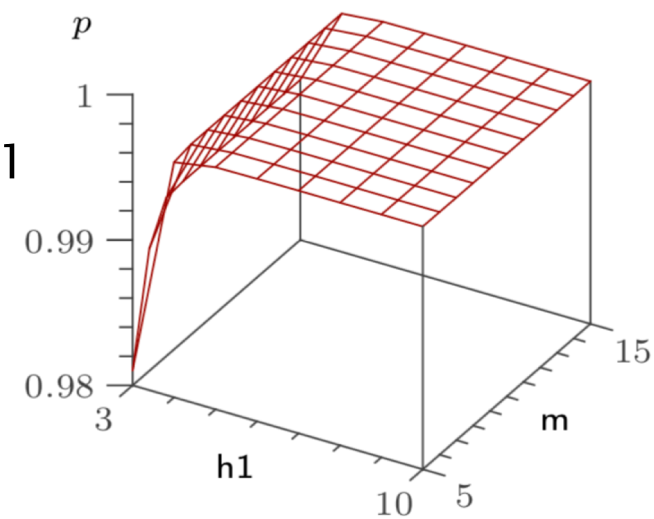
Case studies

- Three case studies considered for evaluation
  - investigate severity/details/trade-offs of <u>known</u> side-channels

1. Covert information flow: the NRL pump
   - network messages only sent from "low" to "high" hosts
   - probabilistic network delays inserted to prevent deliberate side-channel via delays in message acknowledgements

Our analysis:

- max m attempts to send a message
- ack delays h0, h1 for secret values 0, 1
- for a given m, we find ack delays h0, h1 that maximise the probability of a successful leak
- study trade-off between network performance and security



[fix h0=2]

Case studies

2. Square and multiply algorithms
   - more efficient implementation of modular exponentiation
   - known power analysis side channel [Messerges et al.'99] since multiply is more expensive and only needed for some bits
   - we measure the increase in the probability of a successful attack for a variety of different power usage schemes
   - and verify that several alternative schemes indeed fix this

3. Anonymous communication networks: DC-net
   - based on dining cryptographers protocol [Chaum'88]
   - protocol successfully preserves anonymity of the sender
   - study leakage due to timing of extra ops performed by sender

Conclusions

- Automated formal analysis of side channel attacks
  - using an imperative probabilistic language SCH-IMP
  - and building on probabilistic model checking of POMDPs
- Benefits
  - meaningful quantification of attack severity (and generation of the details of the worst-case attack)
  - study of trade-offs involved in defending against attacks
- Limitations
  - custom modelling language; scalability remains a challenge
- Future work
  - optimising POMDP construction and analysis
  - use POMDP to extend attack search (e.g. considering cost)
  - more structured representation of strategies