

Practical 2 – Formal Analysis of Privacy and Security Scenarios and Protocols

In this practical you will learn to use the probabilistic model checker PRISM [1,2] to model and analyse security scenarios and protocols.

Probabilistic model checking is a mathematically-based technique for establishing the correctness, security, performance and reliability of systems that exhibit stochastic behaviour. Example properties that can be established using the technique include the probability that an attacker will guess a password within a specified time period, and the expected response time of a software system under a given workload. The technique operates with models comprising:

- a finite set of states that correspond to different configurations of the analysed system;
- transitions between these states.

The models used in this practical are *discrete-time Markov chains* (DTMCs), i.e. models in which the state transitions are annotated with their occurrence probabilities in the real system. Note that the probabilities of all outgoing transitions of a state must add up to 1.

Given a DTMC model, PRISM can analyse properties of the model that are expressed in a variant of logic called probabilistic computation tree logic (PCTL). You will see concrete examples of (simple) PCTL formulae in the following exercises.

References

- [1] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: verification of probabilistic real-time systems. In 23rd International Conference on Computer Aided Verification (CAV'11), 585-591. Springer, 2011.
- [2] PRISM probabilistic model checker website – <http://www.prismmodelchecker.org>.

Exercise 1

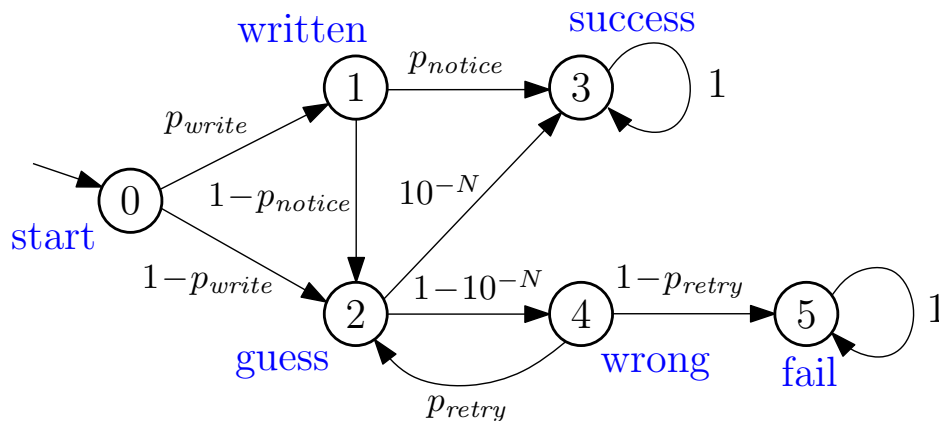
In this exercise you will learn the basics of probabilistic model checking with PRISM.

Consider the following simple scenario. A bank wants to decide the length its credit card PINs. The bank uses randomly allocated PINs (which its customers cannot change). The possible PIN lengths are $N=4$, $N=5$ and $N=6$. The bank knows that using a longer PIN increases the probability p_{write} that customers will write down their PINs and will keep them in the same place as their cards. In fact, a customer survey revealed that

$$p_{write} = 0.001 N^2$$

Suppose now that a criminal gets hold of a credit card. Two things may happen. If its owner wrote down the PIN, the criminal will notice it with (unknown) probability p_{notice} . Otherwise, i.e. if the PIN is not written down or the criminal does not spot it (because it is written in a non-obvious way), the criminal will use brute force to guess the PIN. Assume that after each unsuccessful guess the criminal retries using a new random PIN with (also unknown) probability p_{retry} . Note that p_{retry} is likely to be much smaller than 1.0, as the criminal will avoid staying too long in front of an ATM.

This scenario is modelled by the DTMC below.



In this model, state 0 (labelled ‘start’) corresponds to a customer having received his or her PIN. The transition from state 0 to state 1 models the customer writing down the PIN with probability p_{write} , in which case a criminal who acquired the credit card can notice the written-down PIN with probability p_{notice} – this outcome is modelled by state 3. If the customer does not write down the PIN or the criminal does not notice the written-down PIN, state 2 is reached. In state 2, the criminal attempts to guess the (random) PIN. The guess has a 1 in 10^N probability of success (since there are 10^N different PINs). If the PIN tried out by the criminal is wrong (state 4) then the criminal will either retry to guess the PIN with probability p_{retry} (as modelled by the transition between states 4 and 2) or give up and fail (state 5). The states labelled ‘success’ and ‘fail’ are final states, so their only transition keeps the DTMC in the current state (and has probability 1).

Carry out the following tasks.

Task 1. Start PRISM (look for ‘PRISM GUI’ in the list of applications on Windows, or type ‘xprism’ in a terminal window on GNU/Linux) and use the menu command ‘Model/Open model’ to load the DTMC probabilistic model PIN-attack.pm (available under ‘Practical 2’ on the module VLE). Compare the description of the model in the PRISM high-level modelling language with the graphical representation of the DTMC. The commands of a PRISM DTMC model have the generic form:

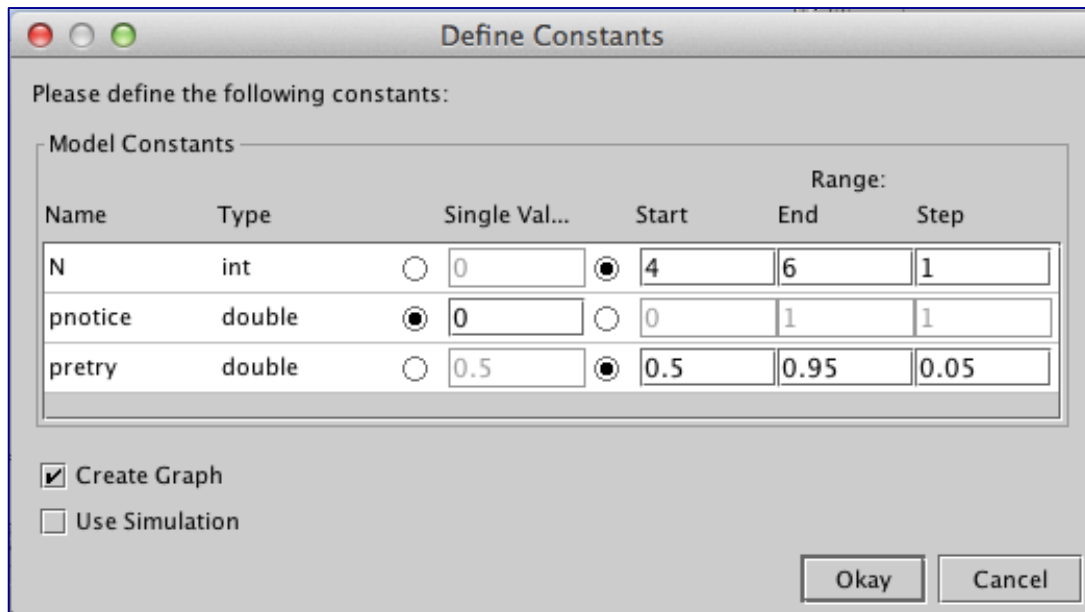
[action] guard -> e1 : update1 + e2 : update2 + ... + en : updaten;

In this command, guard is a boolean expression over the model variables. If guard evaluates to true, the arithmetic expressions e1, e2, ..., en give the probabilities with which the changes update1, update2, ..., updaten of the module variables occur, respectively.

Task 2. After you understand how the DTMC is encoded into PRISM, select the ‘Properties’ tab from the lower left corner of the model checker, right-click in the ‘Properties’ area of this view, and ‘Add’ the property

$P=? [F \text{ state}=3]$

When analysed by PRISM, this property will tell us the probability that the criminal ends up successfully guessing the customer’s PIN (which corresponds to the DTMC reaching state 3 in the **Future/Finally**). Right-click on the property you’ve just added, and select ‘New experiment’ to get PRISM to establish the value of the property for the model parameters shown next:



Please define the following constants:

Model Constants

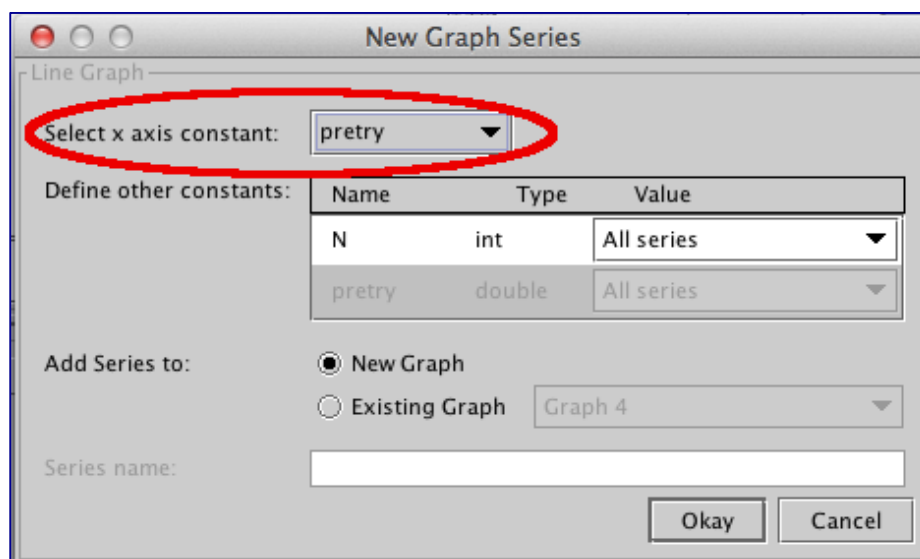
Name	Type	Single Val...	Start	End	Step
N	int	<input type="radio"/> 0	<input checked="" type="radio"/> 4	6	1
pnotice	double	<input checked="" type="radio"/> 0	<input type="radio"/> 0	1	1
pretry	double	<input type="radio"/> 0.5	<input checked="" type="radio"/> 0.5	0.95	0.05

☒ Create Graph
☐ Use Simulation

Okay Cancel

(We start by assuming that the criminal is unable to notice a written-down PIN, i.e. $p_{notice}=0$, and we examine the effect of varying p_{retry} between 0.5 and 0.95 on PINs of different lengths.)

Select p_{retry} as the x axis “constant” when asked by PRISM:



New Graph Series

Line Graph

Select x axis constant: pretry

Define other constants:

Name	Type	Value
N	int	All series
pretry	double	All series

Add Series to:

☒ New Graph
☐ Existing Graph Graph 4

Series name:

Okay Cancel

Analyse the results and suggest the best PIN length to use for this scenario – this may be different for different values of p_{retry} .

Task 3. Repeat the experiment and analysis from the previous task for the following (more realistic) values of p_{notice} : $p_{notice}=0.02$, $p_{notice}=0.05$, $p_{notice}=0.1$, $p_{notice}=0.5$ and $p_{notice}=1.0$. What choices should the bank make each time?

Task 4. The bank carried out a study and identified that $p_{notice}=0.1$ is the correct value of the model parameter analysed in Task 3. Supposed you are hired by the bank to provide expert advice on the best PIN length, and that you are allowed to suggest additional security controls that the bank could use. What will your advice be?

Exercise 2

In this exercise you will use probabilistic model checking to analyse the Crowds anonymity protocol [3] using a simplified version of a DTMC PRISM model originally introduced in [4].

The Crowds protocol makes it impossible to identify which of N parties participating in the protocol is the originator of a web request. The protocol works as follows:

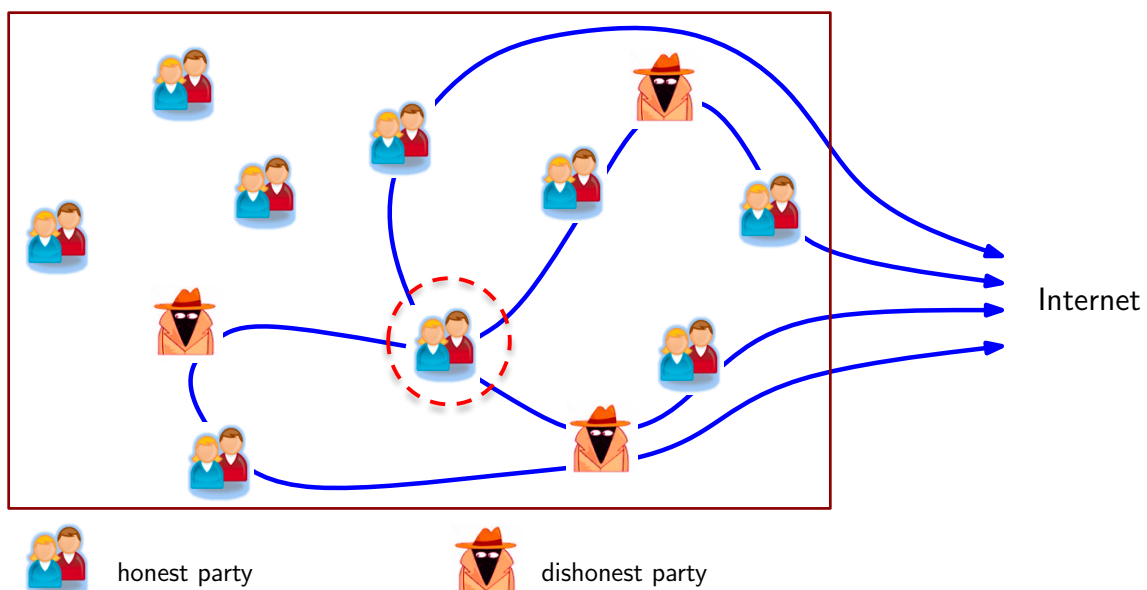
1. With probability p_{forward} , the originator of the request randomly selects one of the N parties (including itself) and forwards the request to this party. The probability of selecting each party is equal (i.e., $1/N$), and the request is forwarded to this party encrypted with a pre-agreed symmetric key known only to the originator and this party. With probability $1-p_{\text{forward}}$, the originator is sending the request to the actual recipient (e.g. a website on the Internet).
2. When an intermediate party receives an encrypted request from another party, it first decrypts it, and then implements the same procedure as the originator. Thus, the request will be forwarded to another randomly chosen party with probability p_{forward} and will be sent to the intended recipient on the Internet with probability $1-p_{\text{forward}}$.

For each session involving communication with a website on the Internet, the protocol is used to establish a *path* involving zero or more intermediate parties.

However, the protocol has a vulnerability. When parties on an established path leave the system (and sometimes when a new session is initiated) new paths need to be established. As a result, an originator typically establishes $N_{\text{Paths}} > 1$ paths over time. Suppose that a fraction $c > 0$ of the N parties are dishonest and colluding. These cN parties share the network traffic they all receive and the identity of the party from which they received each request.

Note now that it is in general possible to identify which requests come from the same originator (by observing web traffic cookies, session IDs, web browser details, etc.). Over time, the colluding parties will therefore observe more than $1/N$ of the requests with the same originator actually coming from the originator.

This vulnerability is illustrated in the diagram below, where two of the three blue paths that reach a dishonest party come from the request originator (circled), compared to a single blue path reaching a dishonest party via another honest party. (A fourth blue path is not seen by any dishonest party.)

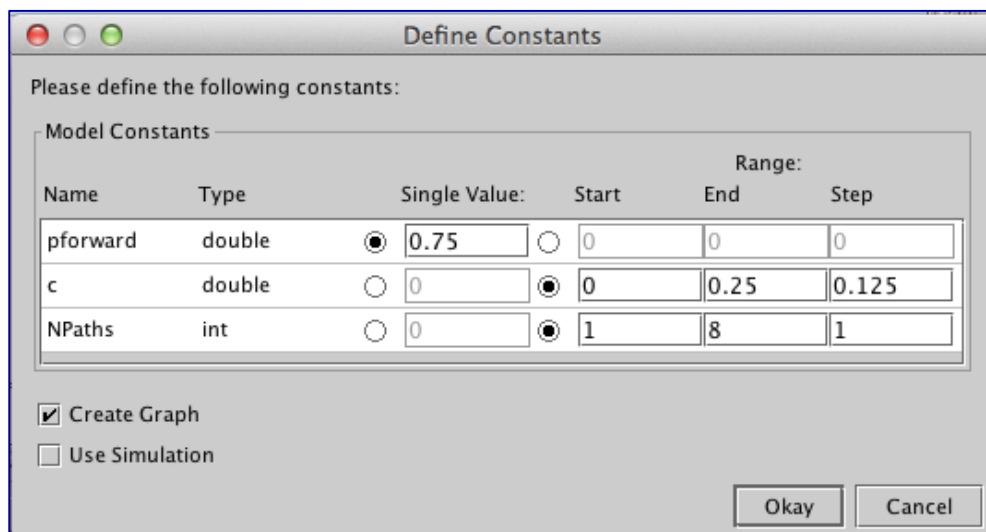


Carry out the following tasks.

Task 1. Load the DTMC model `crowds.pm` (available under 'Practical 2' on the module VLE) into PRISM, and observe how the steps of the Crowds protocol are modelled for $N=8$ parties. Note how the variables `log1`, `log2`, etc. are used to count how many times the coalition of dishonest parties observes a request coming from party 1, party 2, etc. Analyse the probability that these dishonest parties observed the actual originator of the requests more often than any other party after N paths were established (i.e., the probability that `log1` is larger than each of `log2`, `log3`, etc.):

$$P=? [F \text{ path}=N\text{Paths} \ \& \ \text{log1}>\text{log2} \ \& \ \text{log1}>\text{log3} \ \& \ \text{log1}>\text{log4} \ \& \ \text{log1}>\text{log5} \ \& \ \text{log1}>\text{log6} \ \& \ \text{log1}>\text{log7} \ \& \ \text{log1}>\text{log8}]$$

for the following combinations of the model parameters ([select \$N\$ Paths as the "x axis constant"](#)):



Define Constants

Please define the following constants:

Model Constants

Name	Type	Single Value:	Start	End	Step
pforward	double	<input checked="" type="radio"/> 0.75	<input type="radio"/> 0	0	0
c	double	<input type="radio"/> 0	<input checked="" type="radio"/> 0	0.25	0.125
NPaths	int	<input type="radio"/> 0	<input checked="" type="radio"/> 1	8	1

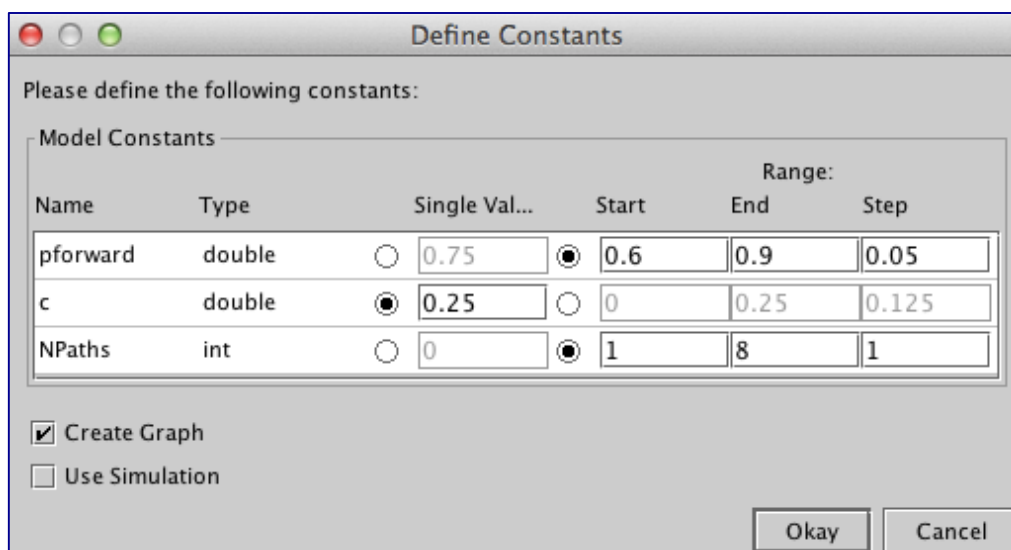
☒ Create Graph
☐ Use Simulation

Okay Cancel

(Note that $c=0.125$ corresponds to one dishonest party and $c=0.25$ to two dishonest parties.)

Analyse the results.

Task 2. Run a similar PRISM experiment, this time keeping c constant and varying p_{forward} as follows:



Define Constants

Please define the following constants:

Model Constants

Name	Type	Single Val...	Start	End	Step
pforward	double	<input type="radio"/> 0.75	<input checked="" type="radio"/> 0.6	0.9	0.05
c	double	<input checked="" type="radio"/> 0.25	<input type="radio"/> 0	0.25	0.125
NPaths	int	<input type="radio"/> 0	<input checked="" type="radio"/> 1	8	1

☒ Create Graph
☐ Use Simulation

Okay Cancel

How do you explain the pattern that you obtained?

Task 3. Analyse the probability that another party (e.g., party 2) is observed forwarding the requests more often than everyone else (including the actual originator). Compare the results with those from the previous two tasks.

Hint: You will need to use a PRISM property very similar to the one from Task 1.

References

- [3] A M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, **1**(1):66–92, 1998.
- [4] V. Shmatikov. Probabilistic Model Checking of an Anonymity System. *Journal of Computer Security* **12**(3/4):355-377, 2004.

Exercise 3 (optional)

Modify the DTMC model PIN-attack.pm from Exercise 1 so that the criminal can try to guess the PIN precisely three times (before the customer's credit card is blocked).

Run PRISM experiments similar to those from Exercise 1. How does the change impact the probability that the criminal will successfully obtain the PIN?