“How Did They Know?” — Model-checking for Analysis of Information Leakage in Social Networks

Louise A. Dennis[1], Marija Slavkovik[2], and Michael Fisher[1]

[1] Department of Computer Science, University of Liverpool
[2] Department of Information Science and Media Studies, University of Bergen

Abstract. We examine the use of model-checking in the analysis of information leakage in social networks. We take previous work on the formal analysis of digital crowds and show how a variation on the formalism can naturally model the interaction of people and groups of followers in intersecting social networks. We then show how probabilistic models of the forwarding and reposting behaviour of individuals can be used to analyse the risk that information will leak to unwanted parties. We illustrate our approach by analysing several simple examples.

1 Introduction

Can we use formal verification to check whether the privacy settings for accessing posted content in social media are effective? In this work we make the first steps towards answering this question in the positive.

The proliferation of social network services has made it possible for vast amounts of contributed content to be shared online by users who simultaneously are members of more than one social network service (SNS). Consider, for simplicity, one SNS user; let us call him Bob. Most social network services allow for various privacy settings to be specified, which should allow Bob to control who can access or further propagate the content he contributes. We say “should allow control” instead of “does allow control” because, in reality, it is not Bob’s privacy settings that ultimately determine accessibility to his shared content, but the combination of the privacy settings of Bob and the privacy settings of all of the users to whom Bob has allowed access to his shared content, *i.e.*, Bob’s *followers*. In the same vein let us call Bob’s *followees* all the users who have allowed access to their shared content to Bob. What is worse with respect to Bob’s control over the privacy of his shared content, is that many of his followers may be users of more than one SNS, with automated interfacing set to synchronise their activities among all the mediums either because one social network allows direct linkage with the API of another (e.g., Livejournal[3] allows posts to be automatically reposted as a link to Facebook[4]) or via third party synchronisation services such as IFTTT[5] and Zapier[6] which allow users to create customised rules to link their SNS accounts to each

[3] `livejournal.com`
[4] `facebook.com`
[5] `ifttt.com`
[6] `zapier.com`

other (and often to additional services and devices such as home automation tools, calendars, alerts and emails). It is thus very difficult for Bob to track *information leakage* – information that Bob shares with his followers, but reach other agents who are not directly authorised to share it. We give a very simple example of information leakage.

Let Bob and his friend Cathy both be members of social network service SN1. Cathy and Bob are within each others' networks on SN1, meaning they are both each other's followers and followees. In turn Bob's boss, Jim, is neither a follower nor a followee of Bob. Bob regularly posts content on SN1 and has chosen to make his content visible only to his followers, believing that his boss cannot access them. Bob makes really sure of this, he checks Cathy's followers and makes sure Jim is not among them. However Cathy and Jim are within each others networks on SN2 and Cathy automatically synchronises her posts between these two SNSs. Bob, having a hard day, complains about his boss on SN1. Cathy, sympathising with Bob acknowledges Bob's message thus making it visible to her followers on SN1, but due to her content synchronisation with SN2, Bob's message becomes also visible to Cathy's followers on SN2. As a result Jim finds out what Bob really thinks of him and rescinds his planned promotion.

It is not simple for one user such as Bob to keep track of all possible combinations of privacy settings within his network and their ultimate effect on content accessibility. Therefore we propose that this task of checking the effective content visibility, *i.e.*, the risk of information leakage occurring, should be automated. As a possible means to accomplish such automation, we propose formal verification. Our aim is to make it feasible for social network services to regularly model-check [4] user settings to ensure that the content privacy settings are effective and efficient, although we are aware that this is a very hard theoretical and engineering problem.

Formal verification is the process of establishing, typically via techniques based on formal logic, that a designed system has its intended properties. Such approaches have become widespread, enabling deep and (semi) automated formal analysis of both software and hardware systems so providing greater clarity concerning reliability and correctness. While logical proof techniques can be used, it is exhaustive state-space exploration, in the form of *model-checking* [4], that is the predominant approach. As we wish to formally model SNSs, our aim here is to utilise formal verification tools to automatically verify their behaviour. In particular, we wish to establish formal properties concerning information leakage using automatic *model-checking* systems.

Consequently we begin, in §2 and §3 by considering the general class of systems and a specific formal model for these based on similar work for namely digital crowds [22] Indeed, the formal model here provides a simplification of that in [22] in that agents have much more limited capabilities. We then consider how model-checking can be used to analyse information leakage properties within this framework. This we do in §4, utilising the PRISM probabilistic model-checker [11]. Finally, in §5, we provide concluding remarks, incorporating both related and future work.

2 System Representation

A *rational* agent is an agent that is capable of obtaining information about her environment, including other agents, and using this information to select actions in order to achieve her goals [24]. A multi-agent system (MAS) is a system of agents that share the same environment and can cooperate or compete within it, as well coordinate their actions. A system of social network services (SNSs) and their users is not a "traditional" MAS, foremost because the networks are not considered to be agents. We propose that since the SNS does obtain information about the users it hosts, and adapts its services and information to the particular needs of specific users, it can be modelled as a rational agent. We use the catch-all phrase "social agent" to refer to both SNSs and their users. We now discuss how to represent a social agent, so that we can formally analyse her properties.

A rational agent can be represented by representing her mental attitudes, in particular her dynamic, informational and motivation aspects. This is exemplified by the popular BDI paradigm for representing agents via mental attitudes [18,17]. "BDI" denotes *Beliefs*, *Desires*, and *Intentions*. In terms of the analysis of information leakage we are primarily interested in the informational aspects of rational agency and so in what follows we will ignore the issue of an agent's desires and intentions[7].

As flexible and powerful as the BDI paradigm is, it is not completely suited for representing social agents since the mental attitudes of these agents, particularly if they are a SNS, are not available or they may not be visible. *E.g.*, a SNS may not have access to what Bob truly believes about his boss, only to what Bob has posted about his boss. Bob can know who Cathy's followers are on the SNS they share, but not on the SNSs they do not have in common. For reasons such as these, work in [22] introduces a new mental state, the communicational attitudes to describe the information about herself an agent shares with the world; $M^{\uparrow i}\varphi$ is used[8] to describe that the modelled agent has communicated φ to i, while $M^{\downarrow i}\varphi$ is used to describe that the modelled agent has received communication φ from agent i.

An agent can be modelled by only using communicational attitudes, when nothing of the private beliefs or goals of the agent is known. The agent representation in [22] builds upon formal agent organisational structures introduced in [8] and further studied in [7,9]. An extended agent, as given in [8], is one for which in addition to the agent's mental attitudes, two further sets of agents (or agent identifiers) are added, *content* and *context*, allowing for both simple agents and a system of agents to be represented using the same model. Content and context sets of agents are related, specifically if agent A_1 is in the content set of agent A_2 then A_2 is in the context set of A_1.

An extended agent, as defined in [7,9], can further include an agent's specification that is visible, or accessible, to the agent's content or context respectively. This paradigm of extended agents is particularly suitable for modelling the visibility of posted content. We thus arrive at our model of social agents. Social agents model the individuals who use social networks and the avatars they maintain on each network. An individual has all the avatars of their followees in their context set and their own avatars in their content set. Each avatar's content contains the agent's followers on that social

[7] Though note that these could be included.

[8] In [22], the formulas $M^{\uparrow i}\varphi$ and $M^{\downarrow i}\varphi$ have also subscripts that denote the nature of the communication, *i.e.*, whether it expresses a question, a statement, or an order, but we here only use statements and thus omit subscripts.

network while its context contains the individual who owns the avatar and any other agents or services to whom they have given posting access.

The model of a social agents and avators is given in Fig. 1. The mental attitudes of the social agent are private and it is not necessary to include any information in this agent part in order to specify a social agent. The information the agent shares to the avatar is made accessible by the avatar to the agents that are her followers.

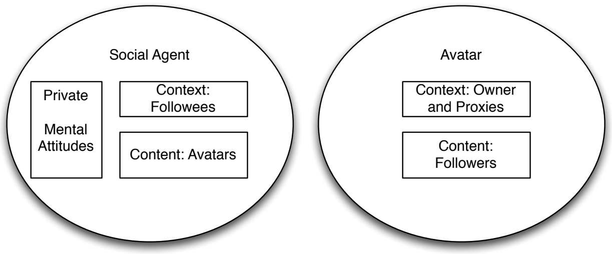

Fig. 1: Basic structure of socal agents and avatars.

Using this social agent structure, we can construct a model for the simple information leakage example outlined in §1. This model is given on Fig. 2.

3 Formal System Specification

The systems we need to specify are the SNS and their users. We represent both networks and users as extended agents using a simplification of the extended agent representation given in [22]. In [22], additional modalities were used to express language abilities as well as the type of the message that the agent sends or receives, linguistic structures that we do not have need for here.

Let Agt be a set of unique agent identifiers, let $Prop$ be a set of atomic propositions and constants, and $Pred$ be a set of a first-order predicates of arbitrary arity. We begin by defining a language $\mathcal{L}_p$ to be a set of grounded first order logic formulas without function symbols, namely the set of all φ_p such that

$$\varphi_p ::= p \mid \neg\varphi_p \mid \varphi_p \wedge \varphi_p \mid P(x_1, \ldots, x_m)$$

where $p \in Prop$, $P \in Pred$ and $x_1, \ldots, x_m \in Agt$.

Depending on the specific needs for a specification, different BDI operators can be used but, for demonstrating our specification approach, we use only the modal operator B which denotes the agent's informational attitudes. $\mathcal{L}_{BDI}$ is then the set of all formulas φ such that

$$\varphi ::= \varphi_p \mid \neg\varphi \mid \varphi \wedge \varphi \mid B\varphi_p$$

where $\varphi_p \in \mathcal{L}_p$.

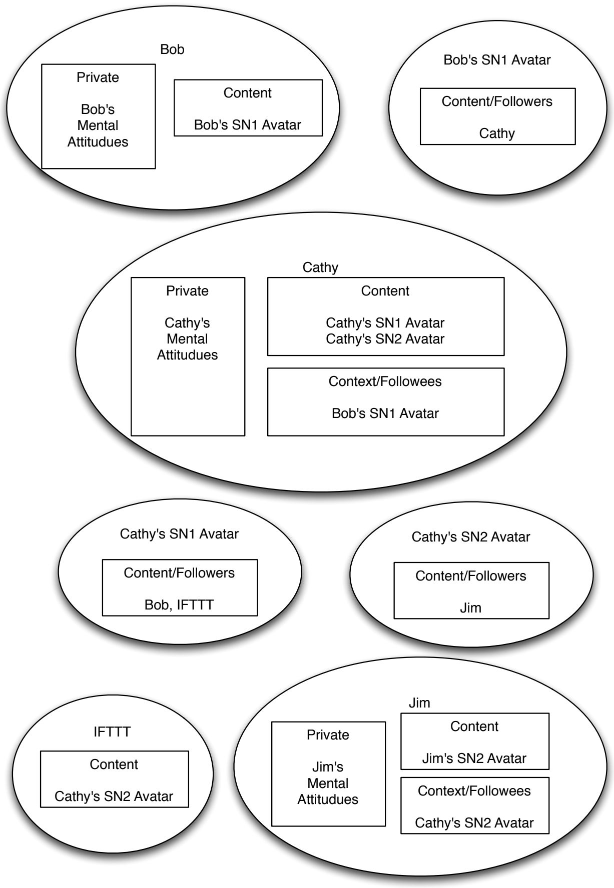

Fig. 2: The system of social agents from §1

Finally, we define the language for specifying communication among agents, $\mathcal{L}_M$. For this language we add operators to indicate the sending and receiving of messages. The language $\mathcal{L}_M$ is the set of all formulas θ such that

$$\theta ::= \varphi \mid M^{\downarrow j}\varphi \mid M^{\uparrow j}\varphi \mid \neg\theta \mid \theta \wedge \theta$$

where $i, j \in Agt$ and $\varphi \in \mathcal{L}_{BDI}$. In [22], temporal information can be included in message formulas but we ignore that possibility here.

The messages are sent to an agent j, however either the *context* set CX or the *content* set CN as a whole can be the target of message broadcast (in the general model, both are agents). We use the shorthand[9]

[9] **Note:** We define the messages with individual agents, not sets as in [8,7,9], because a message can be broadcast to many agents, but it can be sent from one agent, otherwise the sender

$$M^{\uparrow CN}\varphi \equiv \bigwedge_{j\in CN} M^{\uparrow j}\varphi, \qquad M^{\uparrow CX}\varphi \equiv \bigwedge_{j\in CX} M^{\uparrow j}\varphi.$$

The language $\mathcal{L}_{BDI}$ restricts the nesting of modal operators, while $\mathcal{L}_M$ forbids the use of BDI operators outside of the scope of a message operator and does not allow nesting of M operators. Nested messages express meta communication, allowing agents to communicate about what was communicated to them or by them. However, such nesting is not meaningful in our work here.

We can now give the following definition of an agent.

Definition 1. *Let Agt be a set of unique agent identifiers. An agent is a tuple $\langle ID, Bel, Com, CN, CX\rangle$, where $ID \in Agt$ is a unique agent identifier, $Bel \subset \mathcal{L}_p$ is the set of beliefs the agent holds about the world, $Com \subset \mathcal{L}_M$ is the set of messages the agent has received and sent, $CN \in \mathcal{P}(Agt\backslash\{ID\})$ is the set of agents contained and lastly $CX \in \mathcal{P}(Agt\backslash\{ID\})$ is the set of agents in which the agent is contained, i.e., its set of contexts, where $\mathcal{P}(S)$ is the power set of S. The set Bel is consistent and simplified.*

Given an agent $i \in Agt$, an agent specification is a set $SPEC(i) \subset \mathcal{L}_M$, where $B\varphi$ is true iff $\varphi \in Bel$, $cn(j)$ is true iff $j \in CN$, $cx(j)$ is true iff $j \in CX$ and $M^{\downarrow i}\varphi$ is true if $M^{\downarrow i}\varphi \in Com$ and $M^{\uparrow i}\varphi$ is true if $M^{\uparrow i}\varphi \in Com$.

Lastly when specifying the behaviour of a system we combine probabilistic and temporal operators.

$$\varphi ::= \mathtt{P}^{=n}\theta \mid \theta \mid \varphi\mathbf{U}\varphi \mid \bigcirc\varphi \mid \Diamond\varphi$$

where $\theta \in \mathcal{L}_M$, $0 \leqslant n \leqslant 1$ and $\varphi \in \mathcal{L}_{BDI}$. In the intuitive interpretation of our probabilistic operator: $P^{=n}\theta$ means that there is a probability of n that θ is true. For our temporal logic operators $p\mathbf{U}q$ means that p is continuously true up until the point when q becomes true; $\bigcirc r$ means that r is true in the next moment in time; while $\Diamond s$ means that s will be true at some moment in the future. Note that temporal operators can not be nested within the probabilistic operator which, therefore, refers only to the probability of messages being sent or formulas in $\mathcal{L}_{BDI}$ becoming true.

Finally, we assume, via (1), that if a message is sent then it will eventually be received. This is a property of communication among agents that should hold in the environment, for communication to be meaningful.

$$\exists i, M^{\uparrow j}\varphi \in SPEC(i) \Rightarrow \exists j, \Diamond M^{\downarrow i}\varphi \in SPEC(j) \tag{1}$$

Note that we do not develop an axiomatisation for $\mathcal{L}_M$ and do not intend to prove soundness for this language, because we aim ultimately to use it to create specifications for model checking, where soundness is not necessary. The above, together with standard modal and temporal logic semantic structures [23], provides a formal basis for describing agents and SNSs, communication and, hence, behaviour.

In order to consider communication among social networks, let us define the concept of *reachability* between two agents i and j. The agent i can reach agent j if, and

is unknown, which cannot happen here — if your contexts sends you a message it is from exactly one context.

only if, a message sent from i is eventually forwarded to j, under the assumption that avatar agents relay messages from one of their context agents to their followers. To help analyse this in social networks we define an *avatar context*. This is one which broadcasts to all its content agents the messages received from its context agents (*i.e.*, the agent for which it is an avatar or proxies or services authorised by that agent).

Definition 2. *Let i be an agent s.t. $CX(i) \neq \emptyset$. i is an* avatar context *when all the messages sent to i by an agent in its conext are sent on to all of its content agents:*

$$\forall j \in CX(i).\ (M^{\downarrow j}\varphi \rightarrow M^{\uparrow CN}\varphi) \in SPEC(i)$$

To show that information leakage to agent j does not happen to content posted by agent i we need to show that $SPEC(i)$ satisfies property (2):

$$\neg\Diamond(M^{\uparrow CN}\varphi \wedge \neg CN(j)) \rightarrow M^{\downarrow j}\varphi)) \qquad (2)$$

Recall that CN are the avatars of i which relay her messages to her followers, while CX are her followees. The property (2) states that it is not possible that what is posted to followers of i on any network where she has an avatar can be received by j who is not among i's followers.

Upon this basic framework we will now consider formal verification of key properties. To explain this, we will work through a relatively simple series of examples, showing the properties that can be formally established via model-checking.

4 Model Checking Information Leakage

PRISM [11] is a probabilistic symbolic model-checker in continuous development since 1999, primarily at the Universities of Birmingham and Oxford. Typically a model of a program (or in our case a network of agents) is supplied to PRISM in the form of a probabilistic automaton. This can then be exhaustively checked against a property written in PRISM's own probabilistic property specification language, which subsumes several well-known probabilistic logics including PCTL, probabilistic LTL, CTL, and PCTL*. PRISM has been used to formally verify a variety of systems in which reliability and uncertainty play a role, including communication protocols, cryptographic protocols and biological systems [16]. In this paper we use PRISM version `4.1.beta2`.

PRISM is an attractive option for modelling agents and social networks in our formalism since its probabilistic aspects allow us to reason not only about which messages are definitely sent and received, but also about the chance, or risk, that information leakage may occur.

We use a simple set of examples in order to illustrate our approach.

4.1 Basic Scenario

Alice, Bob, and Charlie share two social networks, SN1 and SN2. Alice is a follower of Bob on SN1 but Charlie is not. Charlie is a follower of Bob on SN2 but Alice is not. We treat all three agents, Alice, Bob and Charlie as *modules* in PRISM. Following our formalism we also treat the avatars Bob on the two networks as agents and so also as PRISM modules. The avatars of Bob on SN1 and SN2 are both 'avatar' contexts as defined in Definition 2 – i.e. all information from Bob is automatically transmitted to all content members.

The syntax of prism commands is `[?label] guard -> prob_1:update_1 + ...+ prob_n:update_n` where `label` is an optional keyword used for synchronisation, `guard` is a logical formula over the values of global and local variables, `prob_1` to `prob_n` are probabilities which sum to 1 and `update_1` to `update_n` specify changes to the global and local variables.

We modelled our scenario as a *Discrete Time Markov Chain* in PRISM[10]. Therefore '`->`' indicates a transition from one discrete time step to another. Synchronisation labels force commands in several modules to make a transitions at the same time.

We show the model for Bob's avatar on SN1, `SN1Bob`, in Fig.3. In this model

```
module SN1Bob
sn1bob_relays_message: bool init false;

[bobmessagetosn1] bob_sent_message_to_sn1 = true ->
                          1.0:(sn1bob_relays_message' = true);
[sn1bobmessage] sn1bob_relays_message = true ->
                          1.0:(sn1bob_relays_message' = false);

endmodule
```

Fig. 3: A PRISM model of Bob's followees on SN1.

`bob_sent_message_to_sn1` is a variable in the Bob module that is true if Bob has sent a message to SN1. `sn1bob_relays_message` is a variable in `SN1Bob` that is true if SN1 relays a message from bob to all his followees on SN1. `SN1Bob` contains two PRISM commands, both with synchronisation labels. The first specifies that if Bob has sent a message to SN1 then, with a probability of 1.0, sn1 will relay the message. This transition is synchronised with commands in other modules labelled `bobmessagetosn1` (specifically it synchronises with a command in the Bob module that sends the message). The second specifies that if sn1 relays a message then a synchronised transition will take place after which this variable is set to false (pending receipt of a new message from Bob).

[10] PRISM allows models to be created as Discrete Time Markov Chains (DTMCs), Continuous Time Markov Chains (CTMCs) and Markov Decisions Procedures (MDPs). Since our models had no continuous or non-deterministic aspects that would have required more complex models we opted to use the simplest of these (DTMCs) in modelling. We opted for a representation based on Markov Chains since they capture stochastic processes well and it seemed plausible that models of information leakage in social networks might need to be cyclic. If the possibility of cyclic models could be ruled out then Bayesian Networks would also be a plausible candidate formalism.

To represent the receipt of messages by Bob's followers we use the synchronisation label `sn1bobmessage`. All the commands with this label in all modules make transitions together. In practice this means all Bob's followers receive a message in the same time step. So, for instance, in the representation of Alice in the model, when SN1 relays Bob's message she, with probability 1.0, has a message.

```
[sn1bobmessage] sn1bob_relays_message = true &
                              1.0:(alice_has_message' = true);
```

If there were a second agent, Debbie say, among Bob's SN1 followers then Debbie would contain a similar command.

```
[sn1bobmessage] sn1bob_relays_message = true &
                              1.0:(debbie_has_message' = true);
```

Taken together the synchronised commands in the content agents and the relaying command in `SN1Bob` ensure that SN1Bob meets the specification of an avatar context.

4.2 Example 1

In our first, and simplest, example Alice, Bob and Charlie are the only relevant actors on each network. Bob posts a message to SN1. With the simple model and probabilities PRISM tells us that there is a probability of 1 that eventually Alice will receive the message[11]:

$$P^{=1}\Diamond M_{tell}^{\downarrow sn1bob}\ message \in SPEC(alice) \tag{3}$$

This is expressed as `P>=1 [F(alice_has_message = true)]` in PRISM's property specification language.

We can also prove that there is probability of zero that Charlie will eventually know the message, since the message was relayed only to Bob's followers on SN1 and not to those on SN2.

$$P^{=0}\Diamond M_{tell}^{\downarrow sn1bob}\ message \in SPEC(charlie) \tag{4}$$

4.3 Example 2

We now expand our example to consider the addition of a synchronisation agent, SYNC. Bob has set SYNC up so that when he posts a message to SN1 it is forwarded to the SN2 *as if it was Bob doing so – i.e.*, he has placed SYNC in the context of his avatar agent on SN2. We use a *global variable* `sync_sends_as_bob` to represent that sync can send a message as if it were Bob. When this variable is true then the Bob module sends the message to SN2 using the command

```
[] sync_sends_as_bob = true ->
         1.0: (bob_sent_message_to_sn2' = true) &
              (sync_sends_as_bob' = false);
```

The synchronisation agent is shown in Fig.4.

So, on receipt of a message from Bob's avatar on the first network, the SYNC agent forwards it to SN2 *as if it was Bob doing so*. Under these circumstances we can use PRISM to show that the probability that eventually Charlie receives the message is 1.

[11] We use the notation $P^{=n}$ to indicate that there is a probability of n that something will occur.

```
module SYNC
sync_has_message: bool init false;

[sn1bobmessage] sn1bob_relays_message = true &
                             sync_has_message = false ->
                   1.0:(sync_has_message' = true);

[] sync_has_message = true ->
                   1.0: (sync_has_message' = false) &
                        (sync_sends_as_bob' = true);
endmodule
```

Fig. 4: PRISM model of a simple synchronisation service

4.4 Example 3

Let us now remove the synchronisation agent and consider the possibility that Bob's followers on SN1 may forward the message to their avatars. Assume both Alice and Debbie follow Bob and that Charlie follows both Alice and Debbie. With both Alice and Debbie there is a possibility of 0.1 that they may forward a message to their avatars.

$$\forall j \in \{alice, debbie\}, \forall i, M_{tell}^{\downarrow i}\varphi \in SPEC(j) \Rightarrow P^{=0.1} M_{tell}^{\uparrow SN1}\varphi \tag{5}$$

The PRISM model for Debbie's behaviour is shown in Fig.5 (Alice's module is identical except for variable names and labels). We also add new synchronisation commands to Charlie's model to indicate a receipt of messages from Alice or Debbie's SN1.

```
module Debbie
debbie_has_message: bool init false;
debbie_sent_message_to_sn1: bool init false;

[] debbie_has_message = true ->
                       0.9:(debbie_has_message' = false)
                       + 0.1:(debbie_has_message' = false) &
                             (debbie_sent_message_to_sn1' = true);

[sn1bobmessage] sn1bob_relays_message = true ->
                       1.0:(debbie_has_message' = true);
[debbiemessagetosn1] debbie_sent_message_to_sn1 = true ->
                       1.0:(debbie_sent_message_to_sn1' = false);

endmodule
```

Fig. 5: PRISM model for Debbie

In this network PRISM tells us there is a probability of 0.19 that Charlie will eventually receive the message having had it forwarded to him by either Alice or Debbie (or by both of them).

4.5 Example 4

Suppose at the same time that Bob sends his message he requests that it not be reposted. We view this request as the establishment of a norm and assume this further modifies the chance that Alice or Debbie will forward the message to 0.01. We represent this by modifying the behaviour of agents when they have a message as show in figure 6:

```
[] debbie_has_message = true & do_not_repost_norm = false ->
                    0.9:(debbie_has_message' = false)
                    + 0.1:(debbie_has_message' = false) &
                          (debbie_sent_message_to_sn1' = true);
[] debbie_has_message = true & do_not_repost_norm = true ->
                    0.99:(debbie_has_message' = false)
                    + 0.01:(debbie_has_message' = false) &
                           (debbie_sent_message_to_sn1' = true);
```

Fig. 6: PRISM command showing Debbie's behaviour when a norm is in place

Under these circumstances, PRISM tells us that the probability of Charlie receiving drops to 0.0199.

4.6 Example 5

Lastly we combine our various scenarios as follows: Bob is followed by Alice and Debbie on SN1 and by Charlie on SN2. Debbie and Alice are followed by Charlie on SN1. Debbie has a synchronisation agent set up on SN2 to forward her message automatically to SN1. Debbie is not followed by Charlie on SN2. This set up is shown in figure 7

If Bob asks that his message *not* be forwarded to Charlie then both Alice and Debbie have a 0.01 probability of reposting the message to SN1. However there is a 0.09 probability that Debbie will forward the message to SN2 since Charlie does not follow her there, forgetting that she has a synchronisation agent set up. In these circumstance the probability that Charlie receives the message is 0.109, either because Alice or Debbie has forwarded it directly to SN1, or because Debbie forwarded it to SN2 and then SYNC reposted it to SN1.

4.7 Results Summary

We summarise the results of our examples in the table below, in each case showing the probability, $P^{=?}$ that Alice, Charlie, Debbie or sync eventually receive Bob's message.

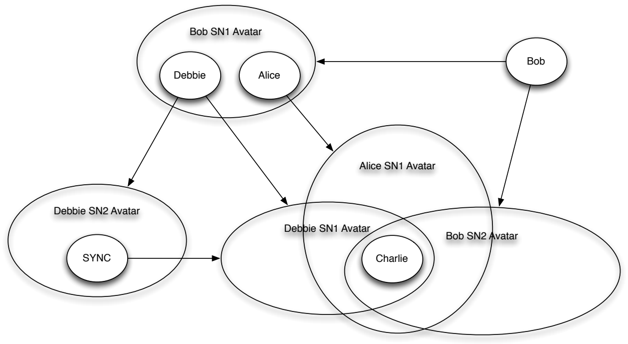

Fig. 7: Social Network for Example 5. Arrows indicate content/context relationships between social agents and their avatars (*i.e.*, "posting privileges"). Content/Context relationships between avatars and followers are shown by inclusion; a social agent appears within the avatars it follows.

	Example				
	1	2	3	4	5
$P^{=?}\Diamond M^{\downarrow}_{tell}\varphi \in SPEC(alice)$	1	1	1	1	1
$P^{=?}\Diamond M^{\downarrow}_{tell}\varphi \in SPEC(charlie)$	0	1	0.19	0.0199	0.109
$P^{=?}\Diamond M^{\downarrow}_{tell}\varphi \in SPEC(debbie)$	n/a	n/a	1	1	1
$P^{=?}\Diamond M^{\downarrow}_{tell}\varphi \in SPEC(sync)$	n/a	1	n/a	n/a	0.09

5 Discussion

The analyses of information leakage that we have presented assume that it is possible to gain some information about the composition of interlinked social networks in order to construct a model for analysis. In particular we assume that we can model the probability with which a user will forward messages; that we can gather information about the followers of users on different social networks (and identify users across social networks); and that we can tell when a user is using a synchronisation agent. We will briefly discuss each of these assumptions.

How likely is a user to forward a message? A decision made by an individual user over whether or not to repost a message to their own followers on a Social Network is obviously highly dependent upon the user, the content of the message, and external factors such as the time of day. However some work already exists in modelling the chances that a message becomes disseminated within a social networks [14] so it reasonable to assume that realistic probabilities could be generated to assess both the risk of messages in general, and of some specific message being forwarded within a network. Adding in assumptions about normative behaviour clearly makes such modelling harder however work also exists in modelling the norms of behaviour on social networking sites [3].

Can we gather information about a user's followers on different social networks and identify users across social networks. While some social networks make the list of a user's followers public, many do not and this obviously presents considerable difficulty in modelling the intersection of these networks. Moreover, for practical reasons the depth of exploration — i.e. the number of forwards — will need to be limited. However, it would not be unreasonable to assume a model in which once a message has been forwarded n times it can count as having "gone viral" and the information therein has irrevocably leaked. We have not considered this possibility here. Typically forwarding of messages happens primarily within the network where the message was generated. In this instance the network itself could choose to offer information leakage analysis from its vantage point of access to all follower groups.

How can we tell if a user is using a synchronisation agent? The main danger of information leakage between networks arises when a user is employing a synchronisation agent. While it is generally easy to tell if a person you follow on a social network is using an agent to repost to that network from some other network, it is considerably harder to tell if they have a synchronisation agent that posts from the network you share to one that you don't. It may be that the existence of such agents for other users will need to be modelled as part of user behaviour. However it is easy to obtain information about synchronisation agents owned by the user wishing to perform a risk analysis. Since users can easily forget that they have set up synchronisations and the synchronisation rules they have may interact in unexpected ways, explicit analysis of these agents remains valuable.

Nevertheless, *in spite of* the difficulty in gaining accurate probabilistic data for the behaviour of humans in the social networks we believe that model-checking does provide a tool which would allow some understanding of the risks of privacy violations and information leaks in social networks. Services which allowed networks to be evaluated on a regular basis in order to asses general risk could be of significant value. While only applied here to very simple examples, we believe the approach described could form the basis for exactly these services.

5.1 Related Work

To the best of our knowledge, this is the first work considering information leakage in the general sense as any information shared on a social network service accessible to persons not directly authorised to access it. Furthermore this is the first attempt to apply formal verification to determine whether, and how often, information leakage occurs. There is a large literature on security models which is obviously of relevance. Most of this literature is focused on access permissions within a single enterprise sytems (e.g., [10]) but [15] introduces socio-technical aspects into the models via the use of obligations and prohibitions and analyses the models for attacks using answer set programming and graph-based models.

"Information leakage" is a term typically used in the context of software engineering, to denote the event when a software system designed to be closed for unauthorised parties reveals some information to them nonetheless. In [13] the use of an agent-based approach to facilitate software information leakage is proposed.

Involuntary information leakage within the context of social network services has been considered for sensitive information, such as personal data and location. A study showed that even if people do not directly reveal their personal information in a social networking service, this may happen indirectly with personal information becoming either directly accessible or inferable from accessible information [12]. Multi-agent system (MAS) technology use is proposed in [1] to assess the vulnerability of particular user profiles on a social network service. Specifically, a software agent is associated with each user profile to extract the user's updates and send them to a controller agent which saves the history of each user and analyses it for possible vulnerabilities.

The DEPNET and DEPINT systems [21,20] reason about social dependency in multi-agent systems. They allow agents to reason about the goals, actions, resources and plans of other agents in order to decide questions such as coalition formation, or where to send requests. This framework doesn't explicitly model information flow among the agents but represents early work reasoning about social structures in mult-agent systems.

Logic-based representation of social network service users and their interactions is an increasing area of research, although work is mainly aimed at studying the information diffusion in a social network. In particular, [19] proposes a two-dimensional modal logic for reasoning about the changing patterns of knowledge and social relationships in networks. Model-checking as a method for verifying properties of information diffusion in open networks has been studied in [2]. The authors, however, focus on modelling the entire (open dynamic agent) network whereas we are modelling a software agent in a social network service system.

5.2 Further Work

As this paper simply sets out a broad direction, and gives quite simple examples, there is much further work to be done.

We would be interested in extending our system to look at, for instance, how information through different routes (e.g. location information sent to one social network service and information about companions sent to another) can be combined to leak key information in unanticipated ways (*e.g.*, someone can now know the location of your companion). Formal verification would surely be more complex but still viable.

The examples we have provided have been built "by hand" and so it would be advantageous to provide a route whereby (some at least) social networks could be automatically extracted into our formalism.

Finally, we here use a relatively standard model-checker, namely PRISM, as we are not primarily concerned with anything more than the beliefs of our agents. As we move to more complex systems it would be ideal to verify complex BDI behaviours. An agent model-checker capable of this exists [6], and indeed this can also be configured to export models to PRISM [5] if probabilistic results are desired. However, it would be ideal to enhance the agent model-checker with explicit content/context constructs in order to facilitate a more direct relationship between our formalism and the model analysed by the tool than we could achieve via a direct translation into PRISM. This would also allow for the practical verification of higher-level properties.

Acknowledgments This work was partially funded through EPSRC Grants EP/L024845 ("Verifiable Autonomy") and EP/N007565 ("Science of Sensor System Software"). The authors would also like to thank Dagstuhl for their facilities and hospitality, something that provided the impetus for this work.

Access to Data The PRISM models used in this work will are available in the University of Liverpool's Data Catalogue at DOI: `10.17638/datacat.liverpool.ac.uk/163`

References

1. Abdulrahman, R., Alim, S., Neagu, D., Holton, D.R.W., Ridley, M.: Multi Agent System Approach for Vulnerability Analysis of Online Social Network Profiles over Time. International Journal of Knowledge and Web Intelligence 3(3), 256–286 (Dec 2012), `http://dx.doi.org/10.1504/IJKWI.2012.050854`
2. Belardinelli, F., Grossi, D.: On the Formal Verification of Diffusion Phenomena in Open Dynamic Agent Networks. In: Proc. International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 237–245 (2015), `http://dl.acm.org/citation.cfm?id=2772912`
3. Bryant, E.M., Marmo, J.: The Rules of Facebook Friendship: A two-stage examination of interaction rules in close, casual, and acquaintance friendships. Journal of Social and Personal Relationships 29(8), 1013–1035 (2012), `http://spr.sagepub.com/content/29/8/1013.abstract`
4. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (1999)
5. Dennis, L.A., Fisher, M., Webster, M.: Two-stage agent program verification. Journal of Logic and Computation (2016), `http://logcom.oxfordjournals.org/content/early/2015/02/16/logcom.exv002.abstract`
6. Dennis, L.A., Fisher, M., Webster, M., Bordini, R.H.: Model Checking Agent Programming Languages. Automated Software Engineering 19(1), 5–63 (2012)
7. Fisher, M., Dennis, L., Hepple, A.: Modular Multi-Agent Design. Tech. Rep. ULCS-09-002, Department of Computer Science, University of Liverpool (2009), `http://www.csc.liv.ac.uk/research`
8. Fisher, M., Kakoudakis, T.: Flexible Agent Grouping In Executable Temporal Logic. In: Proc. 12th Int. Symposium on Languages for Intensional Programming (ISLIP). World Scientific Press (1999)
9. Hepple, A., Dennis, L., Fisher, M.: A Common Basis for Agent Organisation in BDI Languages. In: Languages, Methodologies and Development Tools for Multi-Agent Systems, LNAI, vol. 4908, pp. 71–88. Springer-Verlag (2008)
10. Holm, H., Sommestad, T., Ekstedt, M., Nordström, L.: CySeMol: a tool for cyber security analysis of enterpises. In: Electricity Distribution (CIRED 2013), 22nd International Conference and Exhibition. pp. 1–4. IET (2013)
11. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of Probabilistic Realtime Systems. In: Proc. 23rd Int. Conf. Computer Aided Verification (CAV). LNCS, vol. 6806, pp. 585–591. Springer (2011)
12. Lam, I.F., Chen, K.T., Chen, L.J.: Involuntary Information Leakage in Social Network Services. In: Advances in Information and Computer Security: Third International Workshop on Security, IWSEC 2008, Kagawa, Japan, November 25-27, 2008. Proceedings. pp. 167–183. Springer, Berlin, Heidelberg (2008), `http://dx.doi.org/10.1007/978-3-540-89598-5_11`

13. Lee, Y.C., Bishop, S., Okhravi, H., Rahimi, S.: Information Leakage Detection in Distributed Systems using Software Agents. In: Proc. IEEE Symposium on Intelligent Agents. pp. 128–135 (2009)
14. Lu, X., Yu, Z., Guo, B., Zhou, X.: Predicting the Content Dissemination Trends by Repost Behavior Modeling in Mobile Social Networks. Journal of Network and Computer Applications 42, 197–207 (2014), `http://www.sciencedirect.com/science/article/pii/S1084804514000599`
15. Pieters, W., Padget, J., Dechesne, F., Dignum, V., Aldewereld, H.: Effectiveness of qualitative and quantitative security obligations. Journal of Information Security and Applications 22, 3 – 16 (2015), `http://www.sciencedirect.com/science/article/pii/S2214212614000805`, special Issue on Security of Information and Networks
16. PRISM: Probabilistic Symbolic Model Checker: `http://www.prismmodelchecker.org`. Accessed 2013-05-31.
17. Rao, A.S., Georgeff, M.P.: Modelling Agents within a BDI-Architecture. In: Proc. Int. Conf. Principles of Knowledge Representation and Reasoning (KR). Morgan Kaufmann (1991)
18. Rao, A.S., Georgeff, M.P.: BDI Agents: from Theory to Practice. In: Proc. 1st Int. Conf. Multi-Agent Systems (ICMAS). pp. 312–319 (1995)
19. Seligman, J., Liu, F., Girard, P.: Facebook and the Epistemic Logic of Friendship. In: Proc. 14th Conf. Theoretical Aspects of Rationality and Knowledge (TARK) (2013), `http://www.tark.org/proceedings/tark_jan7_13/p229-seligman.pdf`
20. Sichman, J.S.: DEPINT: Dependence-based coalition formation in an open multi-agent scenarios. J. Artif. Soc. Social Sim. 1(2) (1998)
21. Sichman, J.S., Conte, R., Demazeau, Y., Castelfranchi, C.: A social reasoning mechanism based on dependence networks. pp. 188–192. John Wiley and Sons (1994)
22. Slavkovik, M., Dennis, L., Fisher, M.: An Abstract Formal Basis for Digital Crowds. Distributed and Parallel Databases 33(1), 3–31 (2015), `http://dx.doi.org/10.1007/s10619-014-7161-y`
23. Stirling, C.: Modal and Temporal Logics. In: Handbook of Logic in Computer Science. Oxford University Press (1992)
24. Wooldridge, M., Jennings, N.R.: Intelligent Agents: Theory and Practice. Knowledge Engineering Review 10(2), 115–152 (1995)