# Policy templates for relationship-based access control

Evangelos Aktoudianakis*, Jason Crampton†, Steve Schneider*, Helen Treharne* and Adrian Waller‡
*Department of Computing, University of Surrey
†Information Security Group, Royal Holloway, University of London
‡Thales UK Research & Technology
Email: e.aktoudianakis@surrey.ac.uk

***Abstract*—Social Networks were created to allow users to maintain circles of friends and acquaintances. Over time, they have come to be used to share data objects such as pictures between friends. There have been several approaches to formalize social networks in order to specify complicated access control policies for such objects. These approaches have involved a combination of existing logic with custom languages, with new operators being introduced when more complex policies needed to be expressed. In this paper we demonstrate that set theoretic notation provides a convenient syntax for specifying a social network and the associated access control policies. We demonstrate that our notation enables us to extend the range of policies that can be articulated. We also demonstrate that our notation is simpler and more concise than existing approaches.**

## I. Introduction

Online Social Networks (SNs), such as Facebook or Google+, were created to allow users to connect with their friends, form online communities and share their news and information. During the early days of SNs, a user was able to secure her data in a simplistic yet effective manner – by only making it available to her direct connections. However, as SNs grew in size and popularity, users realised they wanted to make different parts of their data available to different groups of their direct connections.

SNs, having been created with the average user in mind, had to address security in a manner the average user would understand. For example, Facebook expanded on its ‘Friends’ system by introducing categories of friendship (e.g., close friends) and allowing users to create groups of their own. Google+ introduced social circles that would allow users to organize their acquaintances according to their direct connection type (e.g., work, school and family).

The research community have realised the scalable and popular nature of SNs and have taken a strong research interest in its security mechanisms [5], [3], [6]. The research has focused on defining formal models for SNs using relationship based access control (ReBAC) [6] as the basis for defining access policies for use in the SN. In essence, an access policy defines the rules according to which authorization to resources is regulated [5] so that users define clear control on their data without compromising security. A policy’s rules are defined in terms of single or combined user relations.

Fong *et al.*, Cheng *et al.* and Carminati *et al.* have each defined formal notation to express a SN and to define the access policies. The notable difference between each of these approaches is that there is no agreed standard notation for expressing the SN and policies. Additionally, since each of them tries to address a slightly different research problem this results in a slightly different representation of their policies. Moreover, the user relationships that are mainly of interest in these works are restricted to those of single relationships between users and in Carminati’s work [3] specialisation of these relationships.

Our overall research aim is to provide a consistent and unified notation that defines a range of policies for regulating access within the context of a SN. Our overall aim is to provide decision support based on the policies in force within a particular social network. In this paper, as a first step towards our research aim, we define *policy templates* using set theory notation. Throughout the paper we will refer to our approach as set theoretic ReBac (STReBAC). These templates provide a structure for defining different kinds of relationship based access control policies that can be instantiated in many different ways by different policy owners.

We believe that this provides a simple and scalable notation to reaffirm that the notions captured by Fong *et al.*, Carminati *et al.* and Cheng *et al.* can be expressed simply. This simplicity will be particularly evident when we examine policies associated to *abstract paths*. The added benefit is that our model builds on Carminati *et al.*’s work, by allowing users to be involved in more than one type of relationship.

Furthermore, policy templates provide a more general framework for describing policies. Policy templates are a way to parameterize policies so that they can be formally represented and validated using set theoretic notation, and are split in two parts: policy parameters, which always includes the relationship constraints, and policy evaluation. These templates enable users to create well formed policies and to focus on the relationship constraints of the policies rather than their structure.

The paper is structured as follows: Section II provides the set theory notation used in this paper. Section III defines the policy templates for the STReBAC approach. Section IV illustrates an example to demonstrate the use of our policy templates. Section V illustrates the policies of our example using the notation of existing approaches. Section VI compares STReBAC against related work. Finally, Section VII reflects on our results and discusses future work.

## II. Set theory and notation

In this section we present the set theory notation that underpins our STReBAC approach. A Cartesian product of two sets $S$ and $T$ is defined as: $S \times T = \{(s,t) \mid s \in S \wedge t \in T\}$. A binary relation $R$ is defined as: $R \subseteq S \times T$. We define the identity relation $\Delta$ on $S$ to be $\Delta(S) = \{(s,t) \mid s \in S, t \in S, s = t\}$.

Relational composition enables new relations to be formed, e.g., if $R_1 \subseteq S \times T$ and $R_2 \subseteq T \times U$ then $R_1 \mathbin{;} R_2$ is the composed relation between $S$ and $U$: $R_1 \mathbin{;} R_2 = \{(s,u) \mid s \in S \wedge u \in U \wedge \exists t.(t \in T \wedge (s,t) \in R_1 \wedge (t,u) \in R_2)\}$. For $n \geq 1$, we define $R^{n+1}$ to be $R^n \mathbin{;} R$. We define $R^*$ to be the reflexive transitive closure of $R$, given by $R^* = \Delta \cup \bigcup_{i=1}^{\infty} R^i$.

We assume the existence of a set of basic binary relations, such as friend, colleague, etc. We define a grammar for *relational expression* $R$ as follows:

$$R = B \mid R \mathbin{;} R \mid R^* \mid R_1 \cup R_2 \mid R_1 \cap R_2$$

where $B$ is a *b*asic binary relation (defined over $U$). Given a set of basic binary relations $\{B_1, \ldots, B_n\}$, we write $\beta$ to denote the binary relation $B_1 \cup \cdots \cup B_n \cup \Delta$.

Sometime we need to consider functions which are special kinds of relations $R$ where for any $s$ there is at most one $t$ such that $(s,t) \in R$.

## III. Policy Templates

Access control policies exist to identify which interactions between users and data objects are authorized and which are not. In this paper we do not distinguish between different types of objects. As is customary in the literature, we will refer to an attempted interaction as a *request* and assume that a request is modelled as a pair $(x,y)$, where $x$ and $y$ are users. (In other words, $y$ is either authorized to access all data objects belonging to $x$ or authorized for none.) Hence, determining whether a request to access a resource is authorised or not is based on the relationship(s) between the owner of the resource $x$ and the requester, henceforth called the *accessor*. Each policy is associated with an *owner*, which may be a user or the system. Our STReBAC model assumes a finite set $U$ represents all users in our social network.

In Section I we introduced the notion of policy templates. Policy templates provide the framework for evaluating relationship based access control. In this section we will present our templates and briefly discuss how each one can be used to recreate solutions to existing works' ReBAC problems. A template can be generally defined as follows:

$$\boxed{\textit{policy}\text{, where } \textbf{request } (x,y) \textbf{ is authorised when } Pred(x,y)}$$

where *policy* is the syntatic name for the template, $x$ is the owner of the policy template, $y$ is the accessor of the policy template and $Pred(x,y)$ is a policy predicate, which is parameterised by users and must be true in order for the request to be authorised. Depending on the type of predicate, we can categorise our policy templates in three categories: Relational, Abstract Path and Neighbourhood.

### A. Relational template

This simple yet flexible template allows us to formulate policies of varying relational complexity. The template for this policy type is simply a relational expression:

$$\boxed{R\text{, where } \textbf{request } (x,y) \textbf{ is authorised if } (x,y) \in R}$$

The predicate $(x,y) \in R$ simply means that accessor $y$ is related to $x$ by $R$. Therefore, a relation policy template is simply a relation expression from the grammar. In practice, $R$ can be a complex relation, such as $F;F;(F \cap C)$, representing a chain of three relations, where $F$ is the friend relation and $C$ is the colleague relation. The relational template is not concerned with defining explicit constraints within the policy on particular connections paths or the lengths of the paths through a social network that can be traversed. A particularly common use of this template is for owners to define policies regulating access to their respective data objects. We write $u.R$ to denote a policy specified by user $u \in U$; request $(x,\ y)$ is authorised if $x = u$ and $(x,y) \in R$.

### B. Abstract Path template

The Abstract Path template takes into consideration relational compositions, hop lengths and connection paths. The template allows an owner to define a policy based on relational compositions to be satisfied by remote user pairs, provided that these user pairs lie within a particular range of influence. In more detail, it allows an abstract exploration of connected users to find those close enough to be of interest. Then those of interest will need to be in a particular relationship with another user. That way, the owner's policy takes into consideration remote user-pairs relational compositions rather than relational compositions between the policy owner and the accessor.

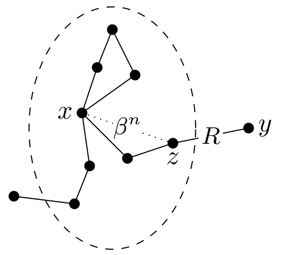
This template is novel because it allows the creation of policies that have not been proposed by previous related work.

An abstract path template has two forms:

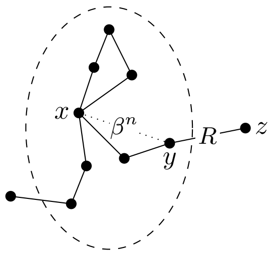
$$\boxed{\begin{array}{l}(ran,R,n)\text{, where } \textbf{request } (x,y) \textbf{ is authorised if}\\ (z,y) \in R \text{ and } (x,z) \in \beta^n \text{ for some } z \in U\end{array}}$$

$$\boxed{\begin{array}{l}(dom,R,n)\text{, where } \textbf{request } (x,y) \textbf{ is authorised if}\\ (x,y) \in \beta^n \text{ and there exists } z \in U \text{ such that } (y,z) \in R\end{array}}$$

The two forms arise because the predicate $R$ representing the relational composition needs to be considered from those users within a particular range of influence and depends on the direction of the relationships in the SN. This is particularly evident when the relationships are themselves functions. The first form defines that the relation of interest $R$ emanates from the accessor. The second form defines the accessor as the target of the relation $R$. These differences are shown in Figure 1. These definitions are illustrated in Section IV. It is true that the first form could be viewed as a simple relational policy template, namely $\beta^n;R$. However, this abstract form means that we do not have to be explicit about $\beta^n$ when a user instantiates a policy template. Thus, when instantiating



(a) $(ran, R, 2)$



(b) $(dom, R, 2)$

Fig. 1. Illustration of $R$ in Abstract Path templates

a $(ran, R, n)$ policy template, the basic relationships will be discovered during policy evaluation of a particular social network.

### C. Neighbourhood templates

At this point, we will provide templates for two policy types introduced by Fong *et al.* [1], [6], the $k$ common conectors and the $k$ clique. In his work, Fong displayed the need to amend his notation so as to include these two complex policy examples. It is worth noting that none of the other existing work can replicate these policy formats introduced by Fong. We will use Fong's terminology for policies for our templates as well.

*1) k common connectors:* Fong defines a problem where an accessor must share a number $k$ different common friends with the policy owner so that the accessor may be allowed access. Using disjunction and conjunction connectors, Fong tried to solve the problem by defining the following policy: $(\langle R_1\rangle\langle R_2\rangle \mathsf{a} \wedge \langle R_1\rangle\langle R_2\rangle \mathsf{a})$ however, there is nothing restricting $\langle R_1\rangle\langle R_2\rangle \mathsf{a} \wedge \langle R_1\rangle\langle R_2\rangle \mathsf{a}$ from referring to the same intermediate common friends twice. As such, Fong introduced $\otimes$ as new notation to replace $\wedge$ for this particular example, asserting that $\otimes$ certifies that the intermediate friend nodes are different. We have used set theory notation to achieve the same result without needing to introduce new symbols to our notation:

A neighborhood template has the following form:

$(R_1, R_2, k)$, where **request** $(x, y)$ **is authorised if** there are exactly $k$ users $z$ such that $(x, z) \in R_1$ and $(z, y) \in R_2$

For an accessor $y$ and an owner $x$ access is granted if $x$ is related to $y$ by $R_1 \,;\, R_2$, via a certain number, $k$, of common connections. The connections may be reached via different relationship compositions, which is why we have $R_1$ and $R_2$ in the template. This template could easily be extended to allow at least $k$ users or at most $k$ users by the inclusion of a binary predicate in the policy definition, for example $(R_1, R_2, k, \geq)$ and $(R_1, R_2, k, \leq)$.

*2) k clique:* Fong defines a problem where an accessor must be part of a closed connection circle of at least $k$ members one of which is the policy owner. As with the common connectors problem, Fong tried to solve the problem using disjunction and conjunction connectors but found it impossible to ensure that each of the $k$ users was different to all the others. He introduced a new symbol @ to amend his definitions and managed to solve the problem at the expense of making his language more complex. We define a clique[1] of $k$ members without needing to introduce new symbols and associated semantics to our notation:

$(clique, k)$, where **request** $(x, y)$ **is authorised if** there is a set of $k$ distinct elements $\{a_0, \ldots, a_{k-1}\}$ such that $(a_i, a_{i+1}) \in \beta$ for $0 \leq i < k-1$, and $(a_{k-1}, a_0) \in \beta$, and $x, y \in \{a_0, \ldots, a_{k-1}\}$.

We could equally define a policy based on cliques of at least $k$ elements as follows:

$(clique, k, \geq)$, where **request** $(x, y)$ **is authorised if** there is a set of $m \geq k$ distinct elements $\{a_0, \ldots, a_{m-1}\}$ such that $(a_i, a_{i+1}) \in \beta$ for $0 \leq i < m-1$, and $(a_{m-1}, a_0) \in \beta$, and $x, y \in \{a_0, \ldots, a_{m-1}\}$.

## IV. A simple example

In our example we use the following binary relations:

- $F \subseteq U \times U$ models the friend relation;
- $C \subseteq U \times U$ models the colleague relation;
- $MB \subseteq U \times U$ models the managed-by relation;
- $\Delta \subseteq U \times U$, where $\Delta$ is the identity relation.

We write $\beta \subseteq U \times U$ to denote $F \cup C \cup MB \cup \Delta$.

We will present a case study in the context of a multi-site business organisation. The relationships in the case study are typical of those exhibited in a SN. Using these relationships we illustrate the application of all of our policy templates. Note that the policy templates abstract away from the detail of whether they are assigned to the user of a resource or a resource itself.The focus of the paper is on the instantiation and the evaluation of the templates.

Let us assume that every employee is required to design an access control policy that controls access to her calendar. Alice is a new employee to the company although she already knows Denise. She decides to add Denise as a friend of hers and is also a colleague to Denise, James, Dave, Bob, and Mary. Jordan and Chris are company managers from different offices but visit Alice's office once per week, to manage Denise and Dave respectively.

Our network and relationships are defined in Figure 2. The relationships between the employees of the company are depicted in Figure 3.

Let us assume that, at first, Alice only wants to grant her friends direct access to her calendar. Accordingly, she defines the following (relational) policy:

$$Alice.F \tag{1}$$

Now, if Denise attempts to access Alice's resources, the policy is evaluated by replacing $y$ with Denise. Since $(Alice, Denise)$ indeed belongs to $F$, Denise is authorised access to Alice's resources. The only other person that would be authorised access to Alice's resources in our example would

[1]Note that this is not the standard graph-theoretic notion of clique; it is more accurately called a cycle.

$$
\begin{aligned}
F &= \{(Alice, Denise), (Denise, Alice), (Alice, James), (James, Alice), (Lora, Denise), (Denise, Lora),\\
&\quad (James, Lora), (Lora, James)\}\\
C &= \{(Dave, James), (James, Dave), (Dave, Alice), (Alice, Dave), (Dave, Bob), (Bob, Dave), (James, Alice),\\
&\quad (Alice, James), (James, Denise), (Denise, James), (Alice, Denise), (Denise, Alice), (Bob, Alice),\\
&\quad (Alice, Bob), (Bob, Mary), (Mary, Bob), (Denise, Mary), (Mary, Denise), (Mary, Alice), (Alice, Mary),\\
&\quad (Lora, Joe), (Joe, Lora), (Joe, Jordan), (Jordan, Joe),\\
&\quad (Colin, George), (George, Colin), (George, Charles), (Charles, George),\\
&\quad (Charles, Bob), (Bob, Charles)\}\\
MB &= \{(Dave, Chris), (Denise, Jordan), (Joe, Jordan), (Colin, Chris)\}
\end{aligned}
$$

Fig. 2. Binary relationships for Example

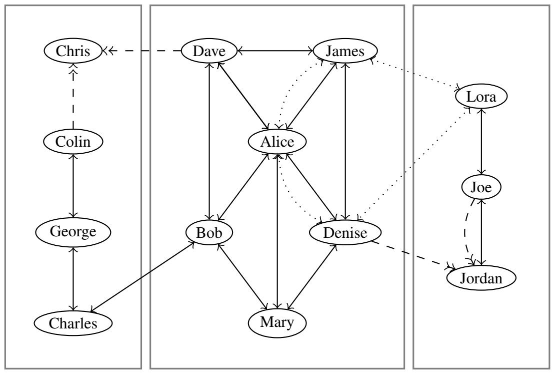


Fig. 3. Visual representation of user binary relations. The rectangles separate different physical locations of the same company. The lines represent binary relations: dotted line($F$), solid line($C$) and dashed line($MB$).

be James since he is the only other person in the friend relation.

Subsequently, Alice receives an email that asks her to share her calendar with all her colleagues. Alice amends her policy and changes the instantiation of relational parameter $R$ from the friend relation $F$ with $F \cup C$ to represent granting access to both friends and colleagues.

The simplicity of this permission change is easily reflected in the updated instantiation of the Relational policy template as follows:

$$Alice.(F \cup C) \tag{2}$$

This simplicity also demonstrates the difference between ReBAC and role based access control. In role based access control, each colleague associated to Alice would need to be explicitly added to an authorised list. With ReBAC, Alice simply modified the relational expression associated with her policy to reflect the changing constraints.

Now, when Denise asks for calendar access the predicate evaluation on Alice's update relational policy yields the following: $(Alice, Denise) \in (F \cup C) = true$.

Alice then receives an email that asks her to make her calendar available to her manager, adding that if she has not yet been assigned a manager, she must make her calendar available to anyone in the company that manages a colleague of hers. Alice changes her policy once more. This time to maintain her existing accessor list and add all managers as well she defines a policy to be the disjunction of two instantiated templates: (2) $\vee$ (3), where (3) is defined as the following instantiation of the Abstract Path template:

$$Alice.(ran, MB, 3) \tag{3}$$

For this example, we consider connections that are at most three connections away. The template requires her to be explicit about which colleagues of hers she considers to be close enough to her so that access may be granted to the managers of those colleagues. Our motivation is that a connection which is further away from the user may be less known to them and therefore it is appropriate to be clear and to define a limit on this. For example, she may not wish a manager which is related via the $10^{th}$ colleague to be granted access because that connection is too tenuous or untrustworthy.

This is what happens when Chris is the accessor $y$. When he tries to access Alice's calendar, (2) does not permit an authorised request since Chris is neither Alice's friend nor a colleague. Therefore, access can only be authorised if the policy evaluation in (3) holds.

The users that are at most three connections away from Alice are Dave, Bob, Mary, Denise, James, Chris, Jordan, Lora, Charles, George and Joe. Therefore, when $z$ is Dave then $(Dave, Chris) \in MB$ and $(Alice, Dave) \in \beta^3$ and hence permission is granted access to Alice's resources.

It is important to note that we were able to search for relationships between distant users without caring about the policy owner's local relationships. All we were interested in was to have a connection path from the policy owner to the target range relationships.

The elegance of the abstract path template can be seen when we write (3) as a relational template instantiation:

$$Alice.(((C \cup F); C; MB) \cup ((C \cup F); MB) \cap \Delta(MB)) \tag{4}$$

Next we investigate the policy scenarios from Fong's latest work [6] – $k$ common connectors and $k$ clique – to demonstrate the flexibility of STReBAC. Let us suppose that Alice wants to grant access to her calendar to anyone who shares at least two friends with her. We instantiate the $k$ common connectors template with both $R_1 = R_2 = F$, hence we obtain:

$$Alice.(F, F, 2, \geq) \tag{5}$$

When Lora tries to access Alice's profile we can validate that $(Alice, Denise) \in F$ and $(Denise, Lora) \in F$ and $(Alice, James) \in F$ and $(James, Lora) \in F$ therefore there are at least two users that satisfy the policy predicate and therefore the request is authorised.

As a final example, Alice changes her policy yet again. This time, she chooses to grant access to anyone who shares a clique with her through three connections. For this we instantiate the 3-clique template:

$$Alice.(clique, 3) \tag{6}$$

One clique of size 3 is $\{Alice, Dave, James\}$ through $C$ relations. Thus a request from accessor $Dave$ would be authorised. On the other hand a request from accessor $Jordan$ would not be authorised.

If Alice had instead chosen to grant access to anyone who shares a clique with her through six connections, then the 6-clique template would be appropriate:

$$Alice.(clique, 6) \tag{7}$$

One clique of size 6 is the set of users $\{Alice, Denise, Jordan, Joe, Lora, James\}$, and so any of the users in that clique can have access to Alice's profile. Conversely, George is not in any clique of size 6 with Alice, so cannot access Alice's profile under this policy.

## V. Comparison to existing work

The purpose of this section is to compare the use of our STReBAC notation in the example of Section IV with expressing the same policies using the notation of related work. By doing so, we will be able to compare such notations to our own and exemplify our belief that our model offers greater simplicity.

In our example, equation (1) Alice used a Relational template to create a 'Friends' only policy and we illustrated this by demonstrating that Denise's request was authorised since the policy was satisfied. Fong *et al.* use modal logic to define policies [6], [8] using an extended ReBAC policy language referred to as $\mathsf{E}$. Fong then uses a satisfiability relation to evaluate the policy predicate, which compares to our $Pred(x, y)$. The satisfiability relation has the form $G, \Sigma, x \models_{\mathsf{E}} \phi$, where $G$ is a social network, $x$ is the owner, $\phi$ is the policy and $y$ is the accessor which is associated with $\Sigma(\mathsf{a})$. It uses a special atomic formula $\mathsf{a}$ which is part of the syntax of the modal language used which asserts that the accessor is the owner herself. A friends only policy is represented as $\langle F \rangle \mathsf{a}$. The policy evaluation would be achieved using the following satisfiability relation: $G, x, y \models_{\mathsf{E}} \langle F \rangle \mathsf{a}$ iff $G, \emptyset[\mathsf{a} \mapsto y], x \Vdash \langle F \rangle \mathsf{a}$. It asserts that the owner $x$ is the friend's neighbour of some individual that is related to the accessor $y$. The semantics of the evaluation is given as follows:

$$\begin{aligned} G, \Sigma, x \quad &\Vdash p \text{ iff } \Sigma(p) = x \\ G, \Sigma, x \quad &\Vdash \langle F \rangle \mathsf{a} \text{ iff there exists } x' \in V \text{ such that} \\ &\quad (x, x') \in F \text{ and } G, \Sigma, x' \Vdash \mathsf{a}. \end{aligned}$$

Authorisation of a request to access a resource in a social network is then defined as a sequent. For example the following sequent represents the accessor Denise requesting access to the calendar resource. The predicate $P(Alice, Denise, G)$ is true provided the above satisfiability relation holds.

$$\frac{\begin{array}{c} N = \langle \_, \_, \_, \_, \_, policy, owner \rangle \qquad \gamma = \langle C, \_, \_ \rangle \\ c \in C \\ P = policy(calendar) \;\; Alice = owner(calendar) \\ G = esn_{\gamma}(c) \\ P(Alice, Denise, G) \end{array}}{\gamma \vdash_N acceses \;\; calendar \;\; in \;\; c}$$

Carminati *et al.* [2], [3] would use the following notation to define the same access policy.

$$?F(Alice, Denise) \wedge ?Owns(Alice, Calendar) = \\ ?Access(Denise, Calendar)$$

Here again we can see the use of the friend relation constraining the policy. There is also clarity that it is Alice who owns the calendar resource and Denise wishes to access it. The notation used is the Semantic Web Rule Language (SWRL) and OWL models. As with Fong, it allows for multiple type and directional relations and it also explores user-to-resource policy scenarios, where policies are assigned directly to user resources (see Section VI for a description of these terms). In our current notation we do not include details of the resources in the policy templates, since our current focus is on defining parameterised templates. It will be essential in the future to associate owners with particular resources explicitly and to augment the templates with this information.

Cheng *et al.*'s [5] model is based on regular expressions and modal conjuction/disjunction connectors that form actions and policies. It allows for path pattern policy compositions through the use of regular expressions. In their notation access control is handled by evaluating both an accessor's and an owner's policies for a specific action. Reproducing (1) using the User-to-user relationship based access control (UURAC) notation in [5] yields the following:

$$\langle access^{-1}(Alice, (F, 1)) \rangle \wedge \langle access, (Denise, (F, 1)) \rangle$$

The $(F, 1)$ in the policies represents the path rule, and therefore, reproducing policy (2) would mean changing the path rule to $((F, 1) \vee (C, 1))$ for both the target user policy ($access^{-1}$ predicate) and the accessing user policy ($access$ predicate).

The model appears more flexible than both Fong's and Carminati's - through the use of regular expressions, it allows for path patterns when evaluating relationship paths against policies. Additionally it allows the policy owner to specify a maximum hop length for a policy. Our notation is similar to UURAC since we use relational composition to define paths and wildcards in UURAC is akin to $R^*$ in our grammar.

Neither Carminati *et al.* nor Cheng *et al.* can recreate Fong's $k$ common connector and $k$ clique examples. As we have already discussed Fong discovered a flaw in his initial approach when the $\wedge$ and $\vee$ connectors were not adequate to ensure the individuality of common connectors:

$$(\langle F\rangle\langle F\rangle \mathsf{a}) \wedge (\langle F\rangle\langle F\rangle \mathsf{a}$$

The above policy ensures that anyone who shares at least two common friends with the owner may access the owner's resources. However there is no constraint against the policy evaluation traversing the same path twice, thus returning a wrong result. The new notation introduced by Fong handles the issue and we can reproduce the policy evaluation predicate in (5) as follows:

$$(\langle F\rangle\langle F\rangle \mathsf{a}) \otimes (\langle F\rangle\langle F\rangle \mathsf{a})$$

Fong's use of $\otimes$ is meant to ensure that the common connectors between owner and accessor were, in fact, different users.

Reproducing the policy predicate for a clique of three in (6) in Fong's notation gives us (as is shown in [6]):

$$\mathsf{a} \vee (\neg \mathsf{a} \wedge \langle F\rangle \mathsf{a} \quad \wedge \quad @p.\langle F\rangle(\neg p \wedge \neg \mathsf{a} \wedge \langle F\rangle \mathsf{a}))$$

The above cases present a good example as to why STReBAC could be considered a simpler notation. As shown in section IV, we were able to design solutions to the problems Fong introduced using a well established language, and we did so without needing to add to its notations. Furthermore, we went a step further by introducing abstract path discovery. To the best of our knowledge, existing work cannot produce a clean and scalable example of how to handle abstract path discovery - Fong, Carminati and Cheng's notations would need to explicitly define the paths leading to finding the managers in Section IV's example. Carminati would need to define three different policies, one for each number of hops up to three to explicitly capture every path for every manager within the number of hops:

$?C(owner, subject1) \vee$
$?F(owner, subject1)$
$?Owns(owner, resource) = ?Access(subject1, resource)$

$(?C(owner, subject1) \wedge ?MB(subject1, subject2)) \vee$
$(?F(owner, subject1) \wedge ?MB(subject1, subject2))) \wedge$
$?Owns(owner, resource) = ?Access(subject2, resource)$

$((?C(owner, subject1) \wedge ?C(subject1, subject2) \wedge$
$?MB(subject2, subject3)) \vee$
$(?F(owner, subject1) \wedge ?C(subject1, subject2) \wedge$
$?MB(subject2, subject3))) \wedge$
$?Owns(owner, resource) = ?Access(subject3, resource)$

Cheng *et al.* and Fong *et al.* would experience the same verbose issue; the main reason behind this, is their need to explicitly define policy paths.

## VI. Related Work

In the literature Cheng *et al.* [5] categorise policies according to their creator:

1) **user policies** are defined by users to manage access to their own profile and/or resources.
2) **super-user policies** are defined by users to target other users in a system. Such policies are used in disjunction/conjunction with the target users' policies in order to reach an access decision.
3) **system policies** are defined by the system and are applied to user groups. They are evaluated in disjunction/conjunction with user policies and, where applicable, super-user policies.

Our policy template are system templates because they apply across all owners and accessors of the SN. The templates can be considered as user policies when they are of the form $u.R$.

Table I is a version of Cheng *et al.*'s [5] categorisation of existing work that we have modified to include our own model. The **Relationship Category** identifies the different ways a SN system may handle relationships - multiple relationship types (the ability have more than just one type of relationship between users e.g., colleagues, friends and siblings), directional relationships (e.g., a friendship would be a bi-directional relationship between two users, a parenthood would be a function between two users), User to User relationships (U2U) which is the model most SNs employ and UserToResources relationships (where resources have policies of their own). In this paper we have illustrated all types of relationships apart from the latter one, since this was not the focus of the paper.

The **Model Characteristics** identify the various recipients of policies in a SN system - Policy individualization (the ability for users to create policies to police other users), User & Resource as a target (enabling resources to have policies of their own), outgoing/incoming policies (enabling policies that police a user's outgoing traffic). In Section V we showed that the fine granularity of the policies in UURAC when the latter two model characteristics are employed could be considered notationally cumbersome.

The **Relationship Composition** identifies relational depth (the maximum number of users considered from the policy owner's location) and relationship composition (the level of complexity of each model's policy language). We believe that our STReBAC approach provides a rich language for defining a wide range of policies using minimal standard set theory notation and the table shows that we are able to encompass all of the compositions that can be supported by related work.

We have extended the original table of Cheng *et al.* by including a row for **Policy Types**. It is here that the versatility of STReBAC can be most clearly seen. The policy types specified in the later work of Fong, which has had the greatest influence on the design of our policy templates, can all be expressed using our policy templates, unlike other proposals

|  | **Fong [7]** | **Fong [6], [8]** | **Carminati [4]** | **Carminati [3], [2]** | **UURAC [5]** | **STReBAC** |
|---|---|---|---|---|---|---|
| **Relationship Categories** |  |  |  |  |  |  |
| Multiple Relationship Types |  | ✓ | ✓ | ✓ | ✓ | ✓ |
| Directional Relationship |  | ✓ | ✓ |  | ✓ | ✓ |
| U2U Relationship | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| U2R Relationship |  |  |  | ✓ |  |  |
| **Model Characteristics** |  |  |  |  |  |  |
| Policy Individualisation | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| User & Resource as a Target |  |  |  | (partial) | ✓ |  |
| Outgoing/Incoming Action Policy |  |  |  | (partial) | ✓ |  |
| **Relationship Composition** |  |  |  |  |  |  |
| Relationship Depth | 0 to 2 | 0 to $n$ | 1 to $n$ | 1 to $n$ | 0 to $n$ | 0 to $n$ |
| Relationship Composition | $F$, $F$ of $F$ | exact type sequence | path same type | exact type sequence | path pattern of different types | exact type sequence, path pattern of different types, depth-search pattern |
| **Policy Types** |  |  |  |  |  |  |
| Relational | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Abstract path |  |  |  |  |  | ✓ |
| Common connectors |  | ✓ |  |  |  | ✓ |
| Cliques |  | ✓ |  |  |  | ✓ |

TABLE I: Comparison of our approach, extension of Table 1 in [5]

in the literature. Moreover, we have developed a new policy template, using abstract paths, that provide support for policies with practical utility, as we demonstrated in Section IV.

## VII. Conclusions and Future work

We proposed a STReBAC model based on set theoretic notation. We identified a new type of policy (*abstract path* template) that cannot be implemented by existing models and we compared the simplicity of our policy construction with that of existing work.

As with existing related work our policies are currently defined by hand and our ongoing work is to develop decision support tools in order to be able to evaluate policies for particular SNs. In future work we will also increase the granularity of detail that we can capture in the models of our SNs by differentiating between users and resources. Currently, we have not separated policies from resources but we plan to create a triple (Policies - Permissions - Resources), adding an extra layer of abstraction to the existing models.

In our example we introduced multiple-relationships between users and categorised the SN based on the users' physical location. The benefit of being able to model multiple-relationships will become valuable when we explore cross-SN communication by allowing the combination of policies from different SNs, and introducing cross SN relationship mappings. For example, a relationship in one SN will have one meaning whilst in another SN a different relationship may have a comparable meaning. Therefore, we will expand our notation to allow different SNs relationship types to map to one another, and we may be able to achieve this using functional renaming. This will allow users to create policies for cross SN communication.

An interest in future work will be to investigate whether there is an increased cost in terms of the time required to evaluate a wider range of policies. We have seen in this paper that SNs are modelled as edge-labelled directed graphs and policies are really concerned with paths (and perhaps the existence of sub-graphs) within those graphs. The computation complexity will come from finding sub-graphs. We will investigate the complexity analysis of evaluating policies in our future work.

## Acknowledgment

We would like to thank Thales UK Research & Technology for supporting this research through an EPSRC CASE KTN Award.

## References

[1] Glenn Bruns, Philip W. L. Fong, Ida Siahaan, and Michael Huth. Relationship-based access control: its expression and enforcement through hybrid logic. In Elisa Bertino and Ravi S. Sandhu, editors, *CODASPY*, pages 117–124. ACM, 2012.

[2] Barbara Carminati, Elena Ferrari, Raymond Heatherly, Murat Kantarcioglu, and Bhavani M. Thuraisingham. A semantic web based framework for social network access control. In *SACMAT*, pages 177–186, 2009.

[3] Barbara Carminati, Elena Ferrari, Raymond Heatherly, Murat Kantarcioglu, and Bhavani M. Thuraisingham. Semantic web-based social network access control. *Computers & Security*, 30(2-3):108–115, 2011.

[4] Barbara Carminati, Elena Ferrari, and Andrea Perego. Enforcing access control in web-based social networks. *ACM Trans. Inf. Syst. Secur.*, 13(1), 2009.

[5] Yuan Cheng, Jaehong Park, and Ravi S. Sandhu. A user-to-user relationship-based access control model for online social networks. In Nora Cuppens-Boulahia, Frédéric Cuppens, and Joaquín García-Alfaro, editors, *DBSec*, volume 7371 of *Lecture Notes in Computer Science*, pages 8–24. Springer, 2012.

[6] Philip W. L. Fong. Relationship-based access control: protection model and policy language. In *CODASPY*, pages 191–202, 2011.

[7] Philip W. L. Fong, Mohd M. Anwar, and Zhen Zhao. A privacy preservation model for facebook-style social network systems. In *ESORICS*, pages 303–320, 2009.

[8] Philip W. L. Fong and Ida Siahaan. Relationship-based access control policies and their policy languages. In *SACMAT*, pages 51–60, 2011.