More Choices, More Control:
Extending Access Control by Meta-Rights Reallocation

Published as: A. Ahmad, B. Whitworth and L. Janczewski, *More Choices, More Control: Extending Access Control by Meta-Rights Reallocation*, International Workshop on Trust, Security and Privacy in e-Government, e-Systems & Social Networking (eGSSN-12), in IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2012), Liverpool, United Kingdom, June 25-27, 2012

Adnan Ahmad, Brian Whitworth
Massey University, Auckland, New Zealand

Lech Janczewski
The University of Auckland, Auckland, New Zealand

***Abstract*— Online Social Networks (OSN) are platforms that let users build relationships by interacting with each other and adding objects. They differ from simple technical systems in having to satisfy social as well as technical requirements, so OSN access control is both more complex and more subtle than traditional. Currently, it is managed by local management of individual domains and local roles like friend. But making friend gives them rights, raising the issue of meta-rights, the right to issue a right. As user move from friend dyads to groups to communities, a systematic scheme to handle meta-rights (e.g. transferring, delegating, multiplying and dividing rights) is required. This paper outlines a general model to manage meta-rights for OSN in particular and socio-technical systems in general. The model's validity derives from socio-technical design, where social requirements like ownership and fairness give technical axioms.**

Keywords – Rights reallocation; multiple ownership; transfer; delegation; rights division, social networks;

I. INTRODUCTION

Socio-technical systems today represent a subtle but profound shift of software towards becoming more sociable [1]. They arise when social interaction is mediated by information technology rather than the physical world [2]. During the last decade, we have witnessed the emergence of online social networks (OSN), where millions of users interact with each other to share billions of resources [3].

In computing systems, managing resources is done by access control, which grants authorized users permissions to act upon information objects. Every computer application has an access control system (ACS) but its role has changed as computing has evolved. The traditional security aim was to prevent unauthorized user access to the system as a whole, to avoid system failure or data theft. Today, each individual has the same concerns, e.g. privacy as the theft of personal data. A social "error" gives public outrage and rebellion. The community fails not the software, but the effect is the same - the system doesn't work and the code doesn't run. Access control is critical to social networks because even one access mishandling can cause a community to reject a site, as loss of privacy can result in family feuds, marriage breakdowns, spam, scams and even physical attacks [4].

Online communities where users share and talk about personal posts need a richer rights set than traditional read and write operations. OSN now let book authors launch marketing campaigns, companies recruit employees, people propagate visions and celebrities run fan clubs. Such acts require many users to share access to many objects, including the persona itself. Systems with only basic access control struggle to meet such social needs [3], e.g. for an ACS to manage many people writing a paper who share rights, to reallocate the original creator's rights needs meta-rights logic. No current ACS model for OSN covers the meta-rights of transferring, delegating, multiplying and dividing rights.

Our previous works [5-8] discuss how basic rights can be legitimately incorporated in access control. This paper does the same for meta-rights. It again uses the socio-technical approach, to define social requirements first. This ensures that technical design doesn't impede social needs, i.e. it avoids a socio-technical gap. The technical design may also support or enhance social rules and needs [9].

This paper is organized as follows: section II discusses related work. Section III sketches the requirements, section IV outlines the core access control model, section V formalizes the meta-rights model, and section VI concludes.

II. RELATED WORK

In traditional access control models, only delegation has got some attention but other modes of reallocation are hardly explored. Traditional models support three types of rights delegation: machine to machine, user to machine and user to user. Machine to machine delegation is the secure authorization of one object to act on the other's behalf [10]. User to machine delegation is when a user needs to securely authorize a system to access resources on his behalf, plus the ability to terminate the delegation [11]. Finally, there is user to user delegation, as users delegate roles to others [12]. Such traditional access control models cannot be used for OSN because they:

- Map every user to every resource in the system, and support system-wide roles, while OSN objects have local visibility only as well as support user oriented social circles.

- Assume single ownership of objects, whether a person or an organization (no multiple ownership).
- Consider delegation as a right moving from one person to another. But OSN access control works with local autonomous domains, so is domain based rather than role based, i.e. privileges are associated with objects not users.

A purely technical analysis of RBAC role delegation allows multi-step delegation [13], i.e. friends of friends of friends, but a socio-technical approach does not as a friend's friend needn't be mine. Delegatees further delegating rights also raises accountability issues, as will be seen. Conversely, traditional access control models don't support multiplying and dividing rights, which is socially common, e.g. conferences usually have more than one chair.

Current access control models for OSN are based on ownership and relationships [14-16], but do not specify the dynamic reallocation of distributed rights found in social networks [17], where everyone can give rights away. This research revisits the problems of OSN access control to suggest an access control model to manage not only ownership and local roles but also meta-rights – the right to give a right. There is currently no access control model for OSN that supports any type of rights reallocation.

III. REQUIREMENT ANALYSIS

Technical architectures that don't support social norms and expectations give a socio-technical gap, between what society wants and what technology does [18]. This gap can't be resolved by the usual social means of physical society, as Internet technologies are fast, variable and global while legislation is slow, fixed and local. Online, code *is* the law, far more than the directives of society [19]. To bridge the socio-technical gap we must integrate social requirements into technical design, i.e. it needs socio-technical design.

The first requirement of any community is that only people are held accountable, e.g. if a car crashes we blame the driver not the car. It follows that the access control system, being software, must allocate all rights over all entities to actors all the time. If not, a rights access request would require the system to decide yes or no, which is socially unacceptable.

Secondly, social evolution requires rights reallocation as socio-technical systems can evolve from an initial state of one administrator with all rights to a community with delegated and shared rights [20]. It allows role backups, work collaboration and decentralization of authority [21].

The social requirement is also that rights reallocation be done in a fair, consistent and understandable way. For example, knowledge exchange systems are platforms for people to share knowledge, e.g. academic conference or journal systems. Social interaction (people relating to people) is important in knowledge sharing, and the ability to reallocate rights is the key to social interaction. If many authors contribute to a paper, it makes sense to share rights to view, edit and delete, but how? A many author paper online can let the *one* submitting author alone edit it, let the *one* submitting author delegate edit to another, let edits proceed only if confirmed by *all* authors, or let *any* author do any edit. Similarly, for conference chairs to delegate rights and responsibilities to track chairs, the technology must let them. Knowledge exchange systems must share rights as well as knowledge.

If use rights are all rights except meta-rights, the rights of an entity can be reallocated as follows:

1) *Transfer.* Allocate all rights.
2) *Delegate.* Allocate use rights only.
3) *Multiply.* Allocate rights severally to an actor set:
   a) All rights
   b) Use rights
   c) Meta-rights
4) *Divide.* Allocate rights jointly to an actor set:
   a) All rights
   b) Use rights
   c) Meta-rights

Transfer gives ownership away irreversibly. It reallocates all rights, including meta-rights, so can't be undone. Delegation gives use rights but not meta-rights. The delegatee can use the object but the rights can be taken back. In general, a right reallocation is *revocable* if the initiating party keeps the meta-rights. One can also give meta but keep use rights, as illustrated by feudal barons pledging fealty to a king. A right can also be allocated to an actor set. In multiply, it is copied to many actors, so they can severally exercise the entire right. In divide, the right is allocated to an actor set who must jointly activate it. Table 1 summarizes the possible reallocation end states for a giver starting with all rights and a receiver with initially none (for a given entity).

In society, to own an object is to be accountable for it, e.g. to own a gun and not secure it is negligent. So the social requirement is that to allocate a right to act on an existing object makes one accountable for it, and so needs consent, e.g. to add a paper coauthor requires their consent. If a right holder grants a use right to another, the ACS needs to confirm that they agree to take it, along with its obligations, except for acts that don't change their target, like view and enter, or that don't reference an existing object, like create.

TABLE 1. REALLOCATION END STATES FOR A GIVER AND A RECEIVER WITH ALL RIGHTS

	Giver		**Receiver**	
	Meta rights	*Use rights*	*Meta rights*	*Use rights*
1. Transfer			√	√
2. Delegate	√			√
3. Multiply				
a. All	√	√	√	√
b. Use	√	√		√
c. Meta	√	√	√	
4. Divide				

a. All	½√	½√	½√	½√
b. Use	√	½√		½√
c. Meta	½√	√	½√	

IV. CORE MODEL

This research models the access control for rights reallocations based on the core access control model for OSN [5], thereby reusing several of the basic primitives. Before the extended module for meta-rights reallocation is presented, we briefly outline relevant aspects of the core access control model. Table 2 defines the constructs of the model.

TABLE 2. ABBREVIATIONS AND THEIR DEFINITIONS

	Definition
SH	*Stakeholder:* A user who posts online resource objects, e.g. papers, reviews, comments or votes.
NS	*Namespace:* The set of objects a stakeholder creates.
VU	*Virtual user:* A user, from the social circle of stakeholder, seeking a NS resource access.
LR	*Local role:* A VU group with defined access to NS resources.
OC	*Object class:* An object group, based on security clearance, whose access is mapped to LRs.
AC	*Attestation certificates:* Permission objects encapsulated various access rights and map LR to OC objects.

These components are used to define an access control model independent of the policy. Each *SH* manages its own policy by allocating *VU*s to *LR*s with predefine access to *OC*s. No global administration is required, as *SH*s administer their *NS* resources.

The *VU*s are not mapped to the resources rather the entry point to a *NS* is the abstraction of local roles. All the *VU*s in *SH NS* are assigned some *LR* and access is managed on the basis of *LR* membership. Likewise, objects *O* in *SH NS* are categorized in security labeled *OC* with respect to their clearance level. Additionally, attestation certificates *(AC)* are introduced to add another protection layer [22] and are assigned to every *LR*. The access is granted on the encapsulation of requested right in *AC* for the requested *OC* label. The system architecture of the core access control model is illustrated in Figure 1.

The access control model can be described as a state transition system {δ, γ, σ, Λ} where δ is a set of states, γ is a set of rights that include privileged requests considered by the system, σ is the entailment relation that determines whether a given right request is true or not in a given state, and Λ is the set of state-transition rules.

The implementation computes a function σ_i: $\delta_i \times \gamma \rightarrow$ *{true, false}*, where δ_i is the set of local states of domain *i*, and γ is the set of specific access requests. In general, δ comprises of five different states namely, *Virtual users* (*VU), Member (M), Non-Member (Nm), Allow (a)* and *Deny (d)*. σ has four set of functions, including mapping of *VU* to *M* or *Nm*, mapping of *objects* to *OC*, allocation of *AC* to *M* and *OC*, and mapping of *LR* to *OC* to decide the outcome of request Υ_i. Λ comprises of the following access rules for every namespace request:

- If a *VUid* is in NS_i and maps to some LR_j, the *VU* state changes to M_i, else it becomes Nm_i.
- Object belongs to an object class under some label (default $L_1(\tau)$), i.e. $O \rightarrow OC_\tau$, where, τ is the set of all security labels used for confidentiality levels. These labels are hierarchical and form a lattice under a partial order > such that $L_1 > L_2$ if and only if $L_2 \in L_1$.
- If *VU* is in *M* state and requests some object *O* from *OC*, and there is a mapping of LR_i to OC_i then the request Υ_i is granted, else it is denied.
- If *VU* is in *Nm* state and requests some object *O*, then the request Υ_i is always denied.

$$\sigma = \begin{cases} VU \rightarrow M/Nm \; \forall \; M \in \; LR \\ O \; \rightarrow \; OC_\tau \\ LR \; \rightarrow \; OC \; (\gamma_i) \\ M \rightarrow a/d_r \; \forall \; r \; \in \gamma \end{cases} \quad \ldots (i)$$

Given a namespace *i* in OSN, an access condition *con* against NS_i is a tuple *(VU, LR, OC, AC)*, where $VU \in \sum_{i=1}^{N} M_{LR_i} \cup \{*\}$ is the requestor in domain *i*, $obj \in OC_\tau$ is the object privacy clearance and $AC \in AC \cup \{*\}$ is the attestation certificate for *LR*. If *VU* =*, *VU* corresponds to any user in OSN but is not active in namespace *i*, whereas if *AC*=*, there is no mapping exist for LR_i to OC_τ.

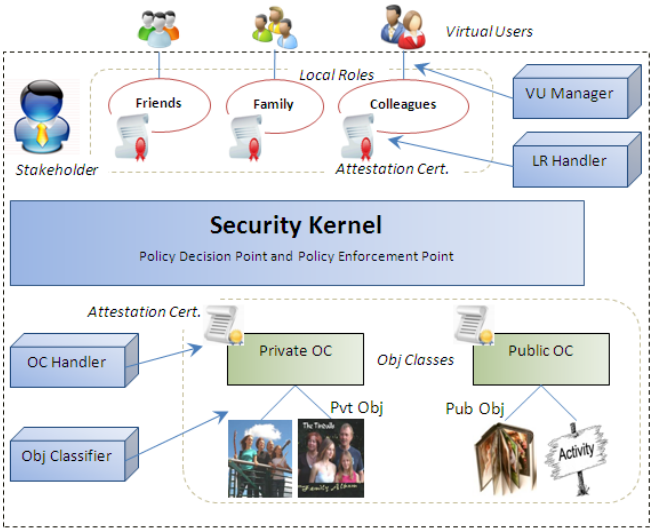

Figure 1. Core access control model system architecture

V. META-RIGHTS MODEL

Basic access control operations can be expressed using a ternary function *Grant-Right (A, E, M)* which lets actor *A* perform method *M* over entity *E*, for E outside the access control system. So, the owner of *text* can grant *Alice* right to *view* it by saying to the ACS:

Grant-Right (Alice, text, View)… (ii)

Meta-rights then define who can operate on rights entities within the access control system. The general form is:

$$Meta\text{-}Right_E = (Owner, AnyRight_E, Allocate) \ldots (iii)$$

This is the same form as (ii) but *the entity acted on is now a right*, and *allocate* is any operation on a right, e.g. transfer, delegate, multiply or divide (Table 1).

Generalizing *equation (ii)*, replacing Alice with an actor set and text with a namespace gives a role:

$$Friend_{NS} = (\{*\}, NS, View) \ldots (iv)$$

where {*} is an unspecified actor set, and *NS* a namespace with objects in it like *text*. The friend role statement permits its actor set to view any object in the namespace. Allocating Alice to the friend role lets her view not only *text* but any other entities in the namespace now or in the future. Yet it is now an act upon an entity inside the ACS:

$$Change_R\ (Alice, Friend_{NS}, Add) \ldots (iv)$$

$Change_R$ as an act upon a role requires meta-rights to define who *owns* the friend role. A role is a variable right attached to an entity, here a local namespace. In general, it is a triplet of any actor set, any entity set and any method set. So the friend role could let friends edit as well as view:

$$Friend_{NS} = (\{*\}, NS, \{View, Edit\}) \ldots (v)$$

Such authorizations are also modeled by formulae of the form *A says Ω*, where Ω is any rights statement [23], e.g. a rights assignment or security policy condition. The form *A says* Ω can be used in any system as *A* and Ω are general, not just restricted to the *admin says* Ω of a central system. This form also requires meta-rights, to specify who can say what, so this model would write (ii) above as:

$$Owner_{text}\ says\ Grant\text{-}Right\ (Alice, text, View) \ldots (vii)$$

where $Owner_{text}$ has the view meta-right, and (iii) as:

$$Owner_{NS}\ says\ Friend_{NS} = (\{*\}, NS, View) \ldots (viii)$$

Equation (iv), adding a friend, becomes:

$$Owner_{Friend}\ says\ Change_R\ (Alice, Friend_{NS}, Add)\ (ix)$$

or equivalently:

$$Meta\text{-}Owner_{NS}\ says\ Change_R\ (Alice, Friend_{NS}, Add) \ldots (x)$$

The right can also be revoked:

$$Meta\text{-}Owner_{NS}\ says\ Change_R\ (Alice, Friend_{NS}, Subtract) \ldots (xi)$$

The above can be generalized to any role on any domain namespace. The details of the allocate operations permitted upon rights entities are now discussed and the system architecture of the proposed model is presented in Figure 2.

A. Transfer

To transfer the ownership of an entity is to change the actor property of all rights upon it. It transfers use and meta-rights to another user [20], which always requires the consent of the new owner. The rights are irrevocably given, e.g. after selling a house the old owner has no rights to it. This is represented by function

$$Meta\text{-}Owner_{Space}\ says\ Change_R\ (A, Right_{All}, Add) \ldots (xii)$$

To maintain the consistency of the system state, the transfer of an entity, as opposed to a right, may take several steps. First, moving the object from one namespace to another, second is the transfer of *says* method to the new owner so the transfer of use rights to the object, and finally the transfer of meta-rights, after which no more activity upon it is possible.

The transfer model $(\delta_{Transfer}, \gamma_{Transfer}, \sigma_{Transfer}, \Lambda_{Transfer})$ extends the core model by $\sigma_{Transfer}$ and $\Lambda_{Transfer}$, where $\sigma_{Transfer}$ comprises of two functions: *(i)* addition of use and meta rights to the new owner, and *(ii)* removal of use and meta-rights from the old owner, and $\Lambda_{Transfer}$ consists of the following set of rules:

- The requestor VU_j owning NS_j is mapped to some LR_i in NS_i which belongs to VU_i using *equation (i).*
- The requested object *O* is classified into the same OC_τ.
- The LR_i has an *AC* with access to that *OC*.
- The transfer of object *O* to NS_j.
- The addition of use rights as well as meta-rights to VU_j.
- The removal of use and meta-rights from VU_i.

$$\sigma_{Transfer} = \begin{cases} VU_j \rightarrow M \,/\, Nm \;\forall\, M \in LR_i \\ O \rightarrow OC_\tau \\ LR_i \rightarrow OC\ (\gamma_i) \\ O \rightarrow NS_j(VU_j) \\ \Upsilon_{(Use)} \rightarrow VU_j \,;\, * \;\Upsilon_{(Meta)} \rightarrow VU_j \\ \neg\, \Upsilon_{(Use)} \rightarrow VU_i \,;\, \neg * \;\Upsilon_{(Meta)} \rightarrow VU_i \end{cases} \quad \ldots (xiii)$$

B. Delegate

The owner of an object delegates the use rights over his object to some other user, who can exercise the rights on the owner's behalf. Delegating a right changes the active actor for use rights but not for entity meta-rights, so can be taken back, To delegate edit rights, as a conference chair to a track chair, means the conference chair cannot directly edit the track, but can dismiss the track chair to take it over. Socially, full accountability for a track increases effort, and likewise a "free" community with delegated rights will participate more. Delegating edit rights to A to a space is the statement:

$$Meta\text{-}Owner_{Space}\ says\ Change_R\ (A, Right_{Edit}, Add) \ldots (xiv)$$

which requires delegatee consent. The delegatee cannot then let a third actor edit because they don't own the meta-rights, i.e. the right to give a right, e.g. renting an apartment gives no right to sub-let. If they give up the edit right, it reverts back to the meta-owner, by the principle that all rights must be allocated. Similarly, lending a book to another doesn't give them the right to on-lend it. The social principle that delegatees can't on-delegate is being consistent to maintain accountability, e.g. if one loans a book to a person who loans it to another person, who loses it, who is accountable?

Apart from the core model components, the delegation model further extends by the delegate relation $\gamma_{Delegate}$, the set of states for delegation $\delta_{Delegate}$, the delegation function

$\sigma_{Delegate}$ and the set of state transition rules $\Lambda_{Delegate}$. $\delta_{Delegate}$ consists of two states, *Delegator (Dlg)* and *Delegatee (Dge)*, $\sigma_{Delegate}$ extends by two functions: *(i)* addition of use rights to *Dge*, and *(ii)* removal of use rights from the *Dlg*, while the revocation of the delegation method remains with the *Dlg*. $\Lambda_{Delegate}$ comprises of the following set of rules:

- The requestor VU_j belongs to the LR_i for the requested *OC* in the NS_i using *equation (i)*.
- The requested object *O* is classified into the same OC_τ.
- The LR_i has an *AC* with access to that OC_τ.
- The assignment of *Dge* role to VU_j, and *Dlg* to VU_i.
- The addition of use rights to the *Dge*, and their removal from *Dlg*.
- Meta-rights remain with the *Dlg*.

$$\sigma_{Delegate} = \begin{cases} VU_j \rightarrow M \;/\; Nm \;\forall\; M \in LR_i \\ O \rightarrow OC_\tau \\ LR_i \rightarrow OC\,(\gamma_i) \\ VU_j \rightarrow Dge_i \;;\; VU_i \rightarrow Dlg_i \\ \gamma_{(Use)} \rightarrow Dge_i \\ \neg\,\gamma_{(Use)} \rightarrow Dlg_i \end{cases} \quad \ldots (xv)$$

C. *Multiply*

This reallocation multiplies the entire right completely, so any party can act alone, as if they owned it exclusively, e.g. a couple's bank account where both can withdraw all the money. Entity operations like view are usually multiplied rather than transferred. The meta-owner of an entity gives view rights to others as well as keeps them, which in effect copies the rights. Now the meta-owner and the beneficiaries can all exercise that right. Multiplying use rights is revocable but multiplying meta-rights is a dictator's dream, as anyone can immediately allocate all rights to themselves (Table 1). Multiplying view rights to a space to A is the statement:

$$Meta\text{-}Owner_{NS}\ says\ Change_R\ (A, Right_{View}, Multiply) \ldots (xvi)$$

This adds A to a viewing set, as in *(ix)*.

Multiply can be modeled using the formula $A \vee B$ *says* Ω to mean that principal *A* or *B says* Ω, where Ω can be any arbitrary operation legal in the settings of access control

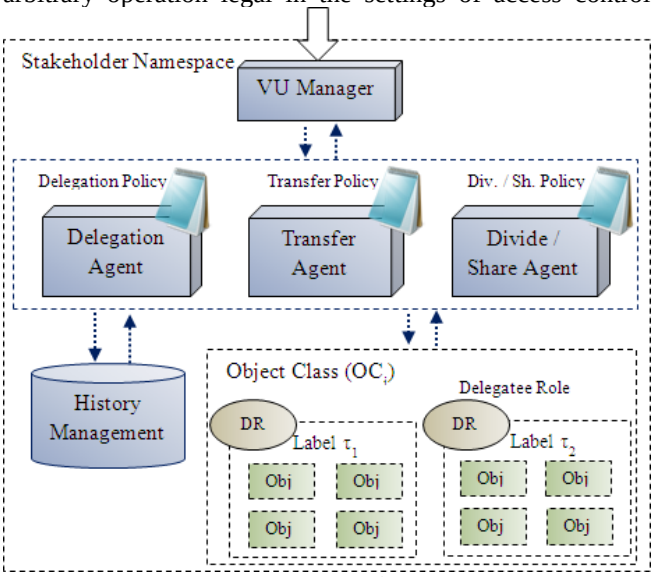

Figure 2. System architecture of meta-rights model

model instance. In multiply, Ω needs to be explicitly defined for the object as all the actors can execute the multiplied operation alone on their own behalf.

Apart from the core model components, the multiply model extends $\delta_{Nultiply}$ by one state, *Secondary owner (SO)* under the multiply relation $\gamma_{Multiply}$. $\sigma_{Multiply}$ extends by two functions: *(i)* addition of use rights to *SO*, and *(ii)* addition of meta rights to *SO*, while $\Lambda_{Multiply}$ comprises of the following set of rules:

- The VU_j belongs to the LR_i for the requested *OC* in the NS_i using *equation (i)*.
- The requested object *O* is classified into the same OC_τ.
- The LR_i has an *AC* with access to that OC_τ.
- The assignment of SO_i role to VU_j.
- The multiplication of use and/or meta-rights to SO_i.

$$\sigma_{Multiply} = \begin{cases} VU_j \rightarrow M \;/\; Nm \;\forall\; M \in LR_i \\ O \rightarrow OC_\tau \\ LR_i \rightarrow OC\,(\gamma_i) \\ VU_j \rightarrow SO_i \\ [\,\gamma_{(Use)} \rightarrow VU_j\,] \;/\; [\,*\;\gamma_{(Meta)} \rightarrow VU_j] \end{cases} \quad \ldots (xvii)$$

D. *Divide*

Division of rights requires multiple actors to collaborate to complete a task. A right divided among actors means all must consent to exercise it, and no actor can act alone, e.g. a couple who jointly own a house must both agree to sell it. In rights division, any party can stop an act but all are needed to activate it. Dividing use rights is revocable but dividing all rights is not, as reverting would require joint agreement. For example, dividing edit rights among two authors would require them both to consent to any edit, as word processing track-change functions currently try to do. This can be modeled as:

$$Meta\text{-}Owner_{Document}\ says\ Change_R\ (A, Right_{Edit}, Divide) \ldots (xviii)$$

After rights division, it requires a statement of the form $A \wedge B$ *says* Ω. This would require the consent of both *A* and *B* to execute the function Ω. One can argue that democracy is the division of community meta-rights among its citizens, where the division is not absolute, i.e. a majority of over 50% agreement allows an action to proceed.

Along with the core model components, the division model includes the divide relation γ_{Divide}, and extends δ Divide by one state, *Primary Owner (PO)*. σ_{Divide} also extends by three functions including: *(i)* addition of use rights to *PO*, *(ii)* addition of meta rights to *PO* and *(iii)* the restriction on *PO* and *Owner* to use them jointly. Λ_{Divide} comprises of the following set of rules:

- The VU_j belongs to the LR_i for the requested *OC* in the NS_i using *equation (i)*.
- The requested object *O* is classified into the same OC_τ.
- The LR_i has an *AC* with access to that OC_τ.
- The assignment of PO_i role to VU_j.
- The addition of use and/or meta-rights to the PO_i.
- Restricting PO_i and VU_i to act jointly.

$$\sigma_{Divide} = \begin{cases} VU_j \rightarrow M \,/\, Nm \;\forall\, M \in LR_i \\ O \rightarrow OC_\tau \\ LR_i \rightarrow OC\ (\gamma_i) \\ VU_j \rightarrow PO_i \\ 1/2 \begin{Bmatrix} [\,\gamma_{(Use)} \rightarrow VU_j\,] \,/\, [*\,\gamma_{(Meta)} \rightarrow VU_j] \\ [\,\gamma_{(Use)} \rightarrow VU_i\,] \,/\, [*\,\gamma_{(Meta)} \rightarrow VU_i] \end{Bmatrix} \end{cases} \ldots (xix)$$

VI. CONCLUSION

This paper defines a mathematical framework for meta-rights reallocation in access control for OSN based on socio-technical design. An architecture for the model is also provided to work with existing security structures. The aim of using socio-technical design approach is to also satisfy social needs to avoid social errors that give community outrage. The framework is generic enough to model a large variety of applications. The next step is to apply this meta-access control model on various current scenarios of OSN to generalize its semantics, and evaluate it against fairness and efficiency.

Socio-technical design is the future of information security, as online communities can't survive without participation. Access control today is more about access than control, i.e. about letting people in rather than keeping them out. This model avoids social errors at source to increase the chance of online social success. The evolution of access control towards the allocation of rights by meta-rights will open up new research dimensions.

ACKNOWLEDGMENT

This work has been sponsored by National Science Foundation (NSF), USA, under award number 0968445. "OKES: An open knowledge exchange system to promote meta-disciplinary collaboration based on socio-technical principles".

REFERENCES

[1] F. Fu, L. Liu, and L. Wang, "Empirical analysis of online social networks in the age of web 2.0". Physica A, Vol. 387, pp.675-684, 2007.

[2] B. Whitworth, "Socio-technical Systems". In C. Ghaoui (Ed.), Encyclopedia of Human Computer Interaction (pp. 533-541). Hershey: Idea Group, 2006.

[3] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu and B. M. Thuraisingham, "A semantic web based framework for social network access control". Proc. ACM Symposium on Access Control Models and Technologies, SACMAT 2009.

[4] ABC News, 17th May 2010. http://www.abc.net.au/news/2010-05-17/teens-murder-sparks-facebook-privacy-plea/829850.

[5] A. Ahmad, and B. Whitworth, "Distributed Access Control for Social Networks", Proc. International conference of information assurance and security (IAS'11)2011.

[6] A. Ahmad, and B. Whitworth, "Access Control Taxonomy for Social Networks", Proc. International conference of information assurance and security (IAS'11) 2011.

[7] B. Whitworth, A. de Moor, and T. Liu, "Towards a Theory of Online Social Rights", in R. Meersman, Z. Tari, P. Herrero et al. (Eds.): OTM Workshops LNCS 4277, pp. 247 – 256, Springer-Verlag Berlin Heidelberg, 2006.

[8] B. Whitworth, and A. deMoor, "Legitimate by design: Towards trusted virtual community environments". Behaviour & Information Technology Journal, 22:1, p31-51, 2003.

[9] N. V. Patel., "Theory of Deferred Action: Exploring the Boundaries of and Between Socio-Technical Systems Design", Design Principles & Practices 3(4), 285-296, 2009.

[10] V. Varadharajan, P. Allen and S. Black, "An Analysis of the Proxy Problem in Distributed systems". IEEE Symposium on Research in Security and Privacy. Oakland, CA 1991.

[11] M. Gasser and E. McDermott, "An Architecture for practical Delegation in a Distributed System". IEEE Symposium on Research in Security and Privacy. Oakland, CA, 1990.

[12] E. Barka, and R. S. Sandhu, "Framework for role-based delegation models", 16th Annual Computer Security Applications Conference (ACSAC 2000) New Orleans, La. IEEE Computer Society Press, Los Alamitos, Calif., 168–177.

[13] L. Zhang, G. Ahn, and B. Chu, "A Rule-based framework for role based delegation", Proc. 6th ACM Symposium on Access Control Models and Technologies, Chantilly, VA, 2001.

[14] B. Ali, W. Villegas, and M. Maheswaran, "A trust based approach for protecting user data in social networks". Proc. Conference of the Center for Advanced Studies on Collaborative research (CASCON'07), pages 288–293, 2007.

[15] B. Carminati, E. Ferrari, and A. Perego, "Rule-based access control for social networks", In On the Move to Meaningful Internet Systems 2006: OTM Workshops 2006.

[16] A. Tapiador, D. Carrera, and J. Salvachúa, "Tie-RBAC: an application of RBAC to Social Networks". Web 2.0 Security and Privacy , Oakland, California, 2011.

[17] A. Simpson, "On the need for user-defined fine-grained access control policies for social networking applications", In SOSOC '08: Proc. of the Work-shop on Security in Opportunistic and social networks, New York, USA, 2008.

[18] C. Dwyer, "Digital Relationships in the 'MySpace' Generation: Results From a Qualitative Study" Proc. 40th Hawaii International Conference on System Sciences, 2007.

[19] J. Mueller, B. Renzl, and A. Kaar, "It's not my community'? insights from social identity theory explaining community-

failure". International Journal of Learning and Change, 3(1), 23-37, 2008.

[20] K. Gaaloul, A. Schaad, and U. Flegel, "A secure task delegation model for workflows", Proc. Second International Conference on Emerging Security Information, Systems and Technologies, 2008.

[21] D. G. Park, and Y. R. Lee, "A Flexible Role-Based Delegation Model Using Characteristics of Permissions" 16th International conference on databases and expert system, (pp. 310-323), 2005.

[22] M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari, "Certificate-based access control for widely distributed resources", Proc. 8th Usenix Security Symposium, pages 215–228, 1999.

[23] V. Genovese, L. Giordano, V. Gliozzi, G. L. Pozzato, "A constructive conditional logic for access control: a preliminary report". Proc. 19th European Conference on artificial intelligence, ECAI'2010. pp.1073~1074., 2010.