A Framework for Establishing Decentralized Secure Coalitions

Hongbin Zhou
Computer Science Department
University College Cork, Ireland
zhou@cs.ucc.ie

Simon N. Foley
Computer Science Department
University College Cork, Ireland
s.foley@cs.ucc.ie

Abstract

A coalition provides a virtual space across a network that allows its members to interact in a transparent manner. Coalitions may be formed for a variety of purposes. These range from simple spaces used by individuals to share resources and exchange information, to highly structured environments in which businesses and applications operate and may be governed according to regulation and contract (security policy). Coalitions may spawn further coalitions and coalitions may come-together and/or merge.

This paper describes a logic-based language that provides a foundation for coalition regulation and contract in a manner that avoids authorization subterfuge and has a number of novel features that make it applicable to open systems. The language provides inter- and intra- coalition delegation, including identity, role and threshold based delegation operations. The logic is used to describe a decentralized infrastructure for establishing and regulating these coalitions. Coalitions are formed with the involvement of founders, constructors and oversight. Constructors are responsible for properly creating a coalition; this service can be provided by a third party. If the service is improperly provided then the constructor is subject to a penalty, which may be collected by another third party providing oversight.

1 Introduction

Modern network environments are intended to support collaborations between large numbers of autonomous entities. Examples include GRID, Web Services, Peer-to-Peer systems, and so forth. We view these systems as providing frameworks for supporting coalitions. A *coalition* is a virtual space to organize and control a number of entities across a network. For example, a simple coalition may be used by individuals to share resources and exchange information. A complex coalition may provide a structured virtual space that supports the ordinary operations within an organization. Such a coalition may spawn further coalitions which represent the organization's departments. A number of these coalitions may come together and form a new coalition for a business-to-business relationship.

Many coalition-supporting frameworks provide not only operational mechanisms to ensure correct operation, but also provide security mechanisms to ensure that only authorized entities may participate in a coalition. However, with regard to authorization, we argue that existing coalition security mechanisms are limited in many respects.

Existing frameworks, such as [2, 12, 13, 18, 19], rely on a "super" administrator who has unlimited authority within the coalition. However, in many cases such flexibility is not desirable. For example, when several entities establish a coalition to share resources, the concern may be that such an administrator can arbitrarily authorize entities outside of the coalition. It is preferable that coalition security mechanisms do not rely on the notion of a "super" administrator.

In addition, existing coalition frameworks rely on the appointment of the "super" security administrator outside of the mechanics of the coalition framework, and he must be accepted by all coalition participants before a coalition can be established. Regulations concerning the resources under the control of this administrator should be carefully issued by the administrator, and well understood by all participants in advance. Different coalitions may require different establishment and regulations, and thus a high degree of expertise is required for an administrator to properly form and manage a coalition. We believe that coalition establishment should not be done in this ad-hoc manner, rather, it should be formalized as an integral part of the coalition framework.

Open delegation is a further challenge within coalition frameworks. We define open delegation to mean that a permission can be delegated in a decentralized way from one coalition to another without ambiguity or subterfuge [11]. It may happen that entities from different coalitions may use the same (permission) name in different ways. Thus, no entity has a complete picture of the entire name schema for all of the resources and services that are available; any entity that makes authorization/delegation decisions, does

so based on its incomplete view of the world. This can result in *authorization subterfuge* [11], whereby, a principal receiving a permission in one domain, can misuse the permission in another domain via some unexpected circuitous but apparently authorised route.

We argue that existing Trust Management/authorization systems, in practice, provide for *closed delegation*, whereby, a permission may be safely delegated only between coalitions that are effectively coordinated by the same security administrator [20]. For example, Keynote [3] relies on the Internet Assigned Number Authority (IANA) [15], RT [16] relies on Application Domain Specification Documents (ADSDs), and X509 [14] relies on the X500 name service, to ensure that different parties use the right name for resources, conditions, other participants, and so forth. However, global name providers are not coalition security administrators; they only provide each name with a unique meaning. Entities from different coalitions may still use arbitrary names to represent their own resources. For example, the cross-domain delegation that is used by the the payment systems [4, 5, 9] based on Keynote, are vulnerable to authorisation subterfuge if care is not taken to properly identify the 'permissions' indicating the payment authorisations when multiple banks and/or provisioning agents are possible. To our knowledge, there is no existing authorization language that supports true open delegation. We believe that without the proper underlying support for open delegation, cross-coalition delegation is unreliable.

This paper describes a logic-based language (Distributed Authorization Language, DAL) that supports coalition regulation and contract in large scale distributed systems without requiring pre-agreed global naming services or "super" administrators. In designing DAL, our motivation has been to provide a safe language that is specialized for open delegation in coalitions. The language is subterfuge-safe in the sense that properly encoded credentials are not vulnerable to subterfuge. While other languages such as [17, 7, 6, 16] offer comparable levels of expressiveness to DAL, credentials written in these languages require formal analysis [20] and/or pre-agreed global naming services to ensure subterfuge-safe delegation. Like type-safe languages, DAL is intended to provide flexibility while preventing classes of unsafe formulae from being encoded. Using DAL, a framework for establishing secure coalitions is proposed whereby coalitions can be dynamically formed in a fully distributed manner without relying on a "super" security administrator or any particular threshold cryptography algorithms.

The remainder of this paper is organized as follows. Section 2 describes DAL and this is used in Section 3 to explore delegation in open systems. A decentralized framework to support coalitions that are specified using DAL is described and discussed in Section 4. Appendix B provides examples that demonstrate the use of DAL in coalition establishment.

2 DAL: Distributed Authorisation Language

This section describes a simple yet expressive logic-based distributed authorisation language. Authorisation decisions are based on its logic axioms and on statements made by principals. While DAL focuses on providing open delegation, due to space limitations in this paper, we do not explore in depth the expressiveness of DAL versus other authorization languages. Subterfuge aside, DAL is generally comparable in terms of expressiveness to existing authorization languages, such as RT [16] and SPKI [7].

2.1 Notation

In the 'real' world, various entities, such as computers, people, organisations, and so forth, should be identified as globally unique. *Identifiers* represent these global unique entities and are denoted by a triple (K, N, $\mathcal{T}$), where K specifies the entity's signature key; text string N is the entity's user-friendly self-description; and $\mathcal{T}$ is a type flag (value $\mathcal{I}$ or $\mathcal{C}$). We assume that no two identifiers share the same public key and thus are globally unique and verifiable by others. For the sake of simplicity, the shorthand $ID^{\mathcal{T}}$ is used to replace the triple (K, N, $\mathcal{T}$), where ID is the global unique identifier replacing K and N. DAL defines two types of identifiers: individual (denoted as $\mathcal{I}$) and coalition (denoted as $\mathcal{C}$). An *individual* is a single entity who may make decisions by itself and use its own signature key to sign credentials. Examples include computers, people, and so forth. A *coalition* is a virtual space which may not make any decisions by itself: all of its decisions are made by its participants. Examples include departments, organisations, and so forth. If no ambiguity can arise, we refer to a globally unique entity as an identifier.

Roles are principals and are useful for managing the delegation of different permissions to groups of principals. A role is denoted as $ID^{\mathcal{T}}.\mathsf{n}$, where $ID^{\mathcal{T}}$ is an identifier, and n is an auxiliary local name defined by the identifier $ID^{\mathcal{T}}$. For example, $UnivA^{\mathcal{C}}.\mathsf{student}$ represents the role $\mathsf{student}$ of coalition $UnivA^{\mathcal{C}}$. An identifier controls, and may discretionarily define, all of its own roles, but may only define roles of other identifiers when explicitly authorised.

Threshold principals are used in DAL to provide threshold-based delegation. For example, professor $A^{\mathcal{I}}$ has three teaching assistants $B^{\mathcal{I}}$, $C^{\mathcal{I}}$, $D^{\mathcal{I}}$ for the course $\mathsf{CS101}$. To give any student an experimental score in $\mathsf{CS101}$, two of the three teaching assistants are required to agree with the score. Like the SPKI/SDSI 'k-of-n' threshold [7], we use a *static threshold structure* to express this kind of compound principal, denoted as $\mathsf{threshold}(k, [p_1, p_2, \ldots, p_{\mathsf{n}}])$, which means that at least k principals from the list of n principals are required. The threshold of the above example is represented as $\mathsf{threshold}(2, [B^{\mathcal{I}}, C^{\mathcal{I}}, D^{\mathcal{I}}])$. As a further

example, in order to combat fraud, a company (coalition) $ComA^{\mathcal{C}}$ has a policy that two managers are required to sign purchase orders for goods with a value over $5,000. We may not a-priori know the number of managers in the company (this number may change with time), and thus, a *dynamic threshold structure*[1] is required to express such compound principals. This is denoted as $\mathsf{threshold}(k,\ r)$ which means that at least k principals in the role r are required. The threshold of the above example is represented as $\mathsf{threshold}(2,\ \mathsf{ComA}^{\mathcal{C}}.\mathsf{manager})$.

Let G and R represent the set of all identifiers and roles respectively. The symbols g, g_0, g_1, ..., range over G, and the symbols r, r_0, r_1, ..., range over R. The set of *principals* P is the set of all identifiers, roles, and their combinations using the logic operators $\wedge$ (and) and $\neg$ (negation). The symbols p, p_0, p_1, ..., range over P.

We assume that statements may contain temporal information to indicate their validity periods. For the sake of simplicity, we do not consider the time related issues in this paper. Therefore, all statements are understood with respect to an arbitrary but fixed current time.

We write $g \,|\!\sim\, s$ to indicate a 'directly-says' statement; this means that a statement s is directly signed by the signing key of identifier g. The 'says' statement, denoted $p \mid\!\sim\, s$, is used to reflect deductive results from a chain of credentials. This means that statement s is directly signed by principal p, or that principal p indirectly makes statement s via a chain of relating statements. We write $(p_1 \wedge p_2) \mid\!\sim\, s$ as a shorthand of $p_1 \mid\!\sim\, s$ and $p_2 \mid\!\sim\, s$ in the remainder paper, and similarly for 'directly-says' statements.

We use *functions* to idealise atomic statements in the language, such as facts, regulations, and permissions. A function name may have different meanings for different principals, and, therefore, function names are meaningful only when bound to a specific identifier. A function $n(\mathsf{a}_1, \ldots, \mathsf{a}_n)@g$, has name n, arguments $\mathsf{a}_1, \ldots, \mathsf{a}_n$, and g is the identifier that defines this function and controls the operation described by the function. A function argument may be a constant or a variable. To distinguish a variable, its name is decorated by a "?". For example, function invocation $Read(\mathsf{fileB},\ ?\mathsf{X})@A^{\mathcal{I}}$ specifies that any (authorised) principal may read fileB. Evaluating statement $Read(\mathsf{fileB},\ \mathsf{C}^{\mathcal{I}})@A^{\mathcal{I}}$ to true means that individual $C^{\mathcal{I}}$ may read fileB by $A^{\mathcal{I}}$'s local function Read. This statement is true only when the permission provider $A^{\mathcal{I}}$ makes the statement as $A^{\mathcal{I}} \mid\!\sim\, Read(\mathsf{fileB},\ \mathsf{C}^{\mathcal{I}})@A^{\mathcal{I}}$, and $C^{\mathcal{I}}$ makes the request $C^{\mathcal{I}} \mid\!\sim\, Read(\mathsf{fileB},\ \mathsf{C}^{\mathcal{I}})@A^{\mathcal{I}}$.

A small number of widely used *global functions* are defined by the provider identified as $DAL^{\mathcal{I}}$. These are similar in intent to the standard classes for programming languages, such as Java. When no ambiguity can rise, the identifier $DAL^{\mathcal{I}}$ is omitted from 'global' functions, and a global function may be denoted as $n(a_1, \ldots, a_n)$.

Global function $actAs(p_0,\ [p_1, \ldots, p_j])$ relates principals (identifiers and roles): each principal $p_i,\ (i \in [1, \ldots, j])$ acts as principal p_0 (a role or an identifier). If $j = 1$, the bracket is omitted in the expression. The function expression $actAs(p_0,\ [p_1, \ldots,\ p_i])$ is a shorthand for the conjunction $actAs(p_0,\ p_1) \wedge \ldots \wedge actAs(p_0,\ p_j)$. For example, statement $actAs(UG^{\mathcal{C}}.\mathsf{member},\ [UA^{\mathcal{C}},\ UB^{\mathcal{C}},\ UC^{\mathcal{C}}])$ means that each of coalitions $UA^{\mathcal{C}}$, $UB^{\mathcal{C}}$, and $UC^{\mathcal{C}}$ is a member of the coalition $UG^{\mathcal{C}}$. The reader should note that identifier $DAL^{\mathcal{I}}$ provides only schema explanations for global functions: actual role relationships are constructed by participants' statements in the logic and can, therefore, be fully decentralized.

2.2 DAL Syntax

DAL statements represent facts that are held by principals (identifiers, roles, and threshold principals). DAL statements are made using basic logic operators, functions and the says and directly says operators. Statements involving principal conjunction, such as $(p_1 \wedge p_2) \mid\!\sim\, s$, are treated as a shorthand for $p_1 \mid\!\sim\, s$ and $p_2 \mid\!\sim\, s$. Threshold principals also represent a shorthand and their encoding in terms of the basic logic operators is described in Appendix A.

Following [1], given principal p, identifier g, statements s, s_1 and s_2, then DAL statements are defined inductively as follows.

- a function is a statement;
- $\neg s_1$ and $s_1 \wedge s_2$ are statements; we write the implication $s_1 \supset s_2$ as an abbreviation for $\neg(s_1 \wedge \neg s_2)$; we also write $s_2 \leftarrow s_1$ as an equivalence for $s_1 \supset s_2$, and $s_1 \vee s_2$ as an equivalence for $\neg(\neg s_1 \wedge \neg s_2)$;
- $g \,|\!\sim\, s$ and $p \mid\!\sim\, s$ are statements.

Operators $|\!\sim$ (directly says) and $\mid\!\sim$ (says) have the same precedence in a statement and have higher precedence than $\wedge$. Operator $\mid\!\sim$ is right-associative. Since both identifiers and roles are principals in DAL, the statement $C^{\mathcal{C}}.\mathsf{manager} \mid\!\sim\, s$ is valid. However, $C^{\mathcal{C}}.\mathsf{manager} \,|\!\sim\, s$ is not a valid statement as 'directly-says' statements reflect credentials directly signed by a key.

2.3 DAL Examples

This section demonstrates how DAL is used to express common security policies.

- Decentralized authority. Authorization is delegated by the issuing of certificates. For example, Alice signs a certificate stating that Bob is a student at University A:

$$Alice^{\mathcal{I}} \mid\!\sim\, actAs(UnivA^{\mathcal{C}}.\mathsf{student},\ Bob^{\mathcal{I}})$$

[1]This contrasts with SDSI which requires the members of a role in a threshold scheme to be a-priori defined.

- Role-based Authority. A principal may take on different roles within the same or different coalitions. These roles can have different authority. In order to prevent accidental misuse of authority, a principal should indicate which role it is acting as, when issuing a credential. For example, $Alice^{\mathcal{I}}$ (a $UnivA^{\mathcal{C}}$ staff member) asserts that $Bob^{\mathcal{I}}$ is a student at $UnivA^{\mathcal{C}}$.

$$Alice^{\mathcal{I}} \mid\sim UnivA^{\mathcal{C}}.\mathsf{staff} \\ \mid\sim actAs(UnivA^{\mathcal{C}}.\mathsf{student},\ Bob^{\mathcal{I}})$$

  Notes that operator $\mid\sim$ is right-associative.

- Identity-based delegation. A principal can delegate rights to other specific principals. For example, $UnivB^{\mathcal{C}}$ trusts $UnivA^{\mathcal{C}}$ to identify students.

$$UnivB^{\mathcal{C}} \mid\sim (actAs(UnivB^{\mathcal{C}}.\mathsf{student},\ ?X) \\ \leftarrow UnivA^{\mathcal{C}} \mid\sim \\ actAs(UnivB^{\mathcal{C}}.\mathsf{student},\ ?X))$$

- Role-based delegation. A principal can delegate rights to certain roles. This kind of delegation can be done across coalitions. For example, $UnivB^{\mathcal{C}}$ trusts $UnivA^{\mathcal{C}}.\mathsf{staff}$ to identify students.

$$UnivB^{\mathcal{C}} \mid\sim (actAs(UnivB^{\mathcal{C}}.\mathsf{student},\ ?\mathsf{X}) \\ \leftarrow UnivA^{\mathcal{C}}.\mathsf{staff} \mid\sim \\ actAs(UnivB^{\mathcal{C}}.\mathsf{student},\ ?\mathsf{X}))$$

  Another example, $UnivA^{\mathcal{C}}$ trusts any coalition that is a university to identify a student.

$$UnivA^{\mathcal{C}} \mid\sim (actAs(UnivA^{\mathcal{C}}.\mathsf{student},\ ?\mathsf{X}) \\ \leftarrow actAs(UnivA^{\mathcal{C}}.\mathsf{univ}, ?\mathsf{Y}) \\ \wedge\ ?\mathsf{Y} \mid\sim actAs(UnivA^{\mathcal{C}}.\mathsf{student}, ?\mathsf{X}))$$

- Conjunction inference. A principal may require another principal to satisfy several separate conditions to be authorised for some action. For example, $UnivA^{\mathcal{C}}$ allows anyone who is both a lecturer and a manager to access a document.

$$UnivA^{\mathcal{C}} \mid\sim (access(\mathsf{fileB},\ ?\mathsf{X})@UnivA^{\mathcal{C}} \\ \leftarrow actAs(UnivA^{\mathcal{C}}.\mathsf{lecture},\ ?\mathsf{X}) \\ \wedge\ actAs(UnivA^{\mathcal{C}}.\mathsf{manager},\ ?\mathsf{X}))$$

- Assert Inference. A principal authorises anyone who satisfies some condition to do a particular action. For example, $Alice^{\mathcal{I}}$ allows anyone who is a student to access a course file.

$$Alice^{\mathcal{I}} \mid\sim (Access(\mathsf{fileA},\ ?\mathsf{X})@Alice^{\mathcal{I}} \\ \leftarrow actAs(\mathsf{student},\ ?\mathsf{X}))$$

- Static threshold structures. For example, professor $Harry^{\mathcal{I}}$ requires two out of three students $Bob^{\mathcal{I}}$, $Carl^{\mathcal{I}}$, and $David^{\mathcal{I}}$ to cooperate in order to to give student $Alice^{\mathcal{I}}$ a score for course CS101 of University A.

$$Harry^{\mathcal{I}} \mid\sim \\ (Score(\mathsf{Alice}^{\mathcal{I}}, \mathsf{CS101}, ?\mathsf{score})@UnivA^{\mathcal{C}} \\ \leftarrow \mathsf{threshold}(2,\ [Carl^{\mathcal{I}}, David^{\mathcal{I}}, Bob^{\mathcal{I}}]) \mid\sim \\ Score(\mathsf{Alice}^{\mathcal{I}}, \mathsf{CS101}, ?\mathsf{score})@UnivA^{\mathcal{C}})$$

- Dynamic threshold structures. Professor $Harry^{\mathcal{I}}$ requires two students from $sec^{\mathcal{C}}$ cooperate to approve $Alice^{\mathcal{I}}$'s score for course CS101 of University A.

$$Harry^{\mathcal{I}} \mid\sim \\ (Score(\mathsf{Alice}^{\mathcal{I}}, \mathsf{CS101}, ?\mathsf{score})@UnivA^{\mathcal{C}} \\ \leftarrow \mathsf{threshold}(2,\ sec^{\mathcal{C}}.\mathsf{student}) \mid\sim \\ Score(\mathsf{Alice}^{\mathcal{I}}, \mathsf{CS101}, ?\mathsf{score})@UnivA^{\mathcal{C}})$$

- Limited role-based delegation. A principal may require a role from a specified coalition to do a job. See the example for dynamic threshold structures.

2.4 Proof System

The proof system has three kinds of axioms:

1. The standard axioms and rules of propositional logic, such as the axiom $(s_1 \wedge s_2) \supset s_1$, the rules $(s_2 \leftarrow s_1) \equiv (s_1 \supset s_2)$, and

$$\frac{s_1, s_1 \supset s_2}{s_2}$$

2. The standard axioms and rules of modal logic:

$$(p \mid\sim (s_1 \supset s_2)) \supset ((p \mid\sim s_1) \supset (p \mid\sim s_2)) \\ (p \mid\sim (s_1 \supset s_2)) \supset ((p \mid\sim s_1) \supset (p \mid\sim s_2))$$

$$\frac{s}{p \mid\sim s}$$

   These are useful for manipulating 'directly-says' and 'says' statements. The axioms define that the statements that a principal says are closed under consequence; the rule expresses that every principal says all provable statements.

3. A small number of additional intuitive axioms, **R1 - R8**, are listed in Figure 1.

R1 reflects the fact that a 'says' statement may be deduced from a single credential. **R2 – R5** define that only when both the role appointor g_1 and role acceptor g_2 explicitly confirm the role binding that g_2 (or g_2's role $g_2.\mathsf{n}_2$)

is bound to g_1 (or g_1's role $g_1.\mathsf{n}_1$), then the role binding becomes a fact. **R6** and **R7** define that the role binding relation is transitive and reflexive. **R8** defines that, if p_1 is bound to p_2, then a statement by p_1 is taken as a statement by p_2 only when p_1 is explicitly representing p_2 in the statement. We informally consider the soundness of this proof system.

Proposition 1 *If statement s is provable (that is, deducible by the axioms) in the logic, then s is valid (in the sense that the authorization request is granted).*

Proof sketch (following the approach in [1]). The propositional axioms and rules are valid, since the propositional connectives are interpreted in the usual manner. So too are the standard axioms and rules of modal logic. **R1**, **R6** and **R7** are straightforwardly valid. We argue that the remaining axioms are valid using the following comparison with SPKI/SDSI [7], a well-known name schema to link roles(names) between principals.

SPKI/SDSI principal expression $p_1's \ \ldots \ 's \ p_n$ means that p_n is a local name in ... in p_1's local name space, where each of p_i is an identifier or an auxiliary local name. A DAL principal may be regarded as SPKI/SDSI principal with the restrictions: p_1 must be an identifier; p_2 must be an auxiliary local name; and the linking name length $n \leq 2$. Intuitively, when $n = 1$, the principal is an identifier; when $n = 2$, the principal is a role. Thus the set of DAL principal expression is a subset of SPKI/SDSI principal expression.

Given this restriction and the fact that an identifier can be recognised by all principals, the SPKI/SDSI name linking rule may be represented in DAL as

$$(g_1 \mid\sim actAs(g_1.\mathsf{n}_1, g_2)) \supset actAs(g_1.\mathsf{n}_1, g_2)$$
$$(g_1 \mid\sim actAs(g_1.\mathsf{n}_1, g_2.\mathsf{n}_2)) \supset actAs(g_1.\mathsf{n}_2, g_2.\mathsf{n}_2)$$

R2 – **R5** are constructed from these SPKI/SDSI axioms together with the restriction that not only the role appointer is required to make the name linking statement, but also the role acceptor. Similarly, **R8** and **R9** are constructed from SPKI/SDSI's 'speaks-for' axiom $actAs(p_2,\ p_1) \supset ((p_1 \mid\sim s) \supset (p_2 \mid\sim s))$ with the restriction that, to represent another principal, a principal must explicitly represent that principal in the statement. Therefore, DAL's provable statements form a subset of SPKI/SDSI provable statements. The soundness of SPKI/SDSI— *that all provable statements are valid*— has been proven [1]. Therefore, all provable statements in DAL are also valid.

Soundness implies consistency: *if s is provable in the logic, then $\neg s$ is not provable.*

3 Delegation in Open Systems

Authorisation subterfuge [11, 20] is the vulnerability whereby a principal receiving a permission in one domain, can somehow misuse the permission in another domain via some unexpected circuitous but apparently authorised route. In this paper we use an example to illustrate the subterfuge problem. [20] provides a formal treatment of the problem.

The SPKI/SDSI axioms can be characterized in terms of the DAL syntax as follows.

- SPKI/SDSI Name Linking.

$$(g_1 \mid\sim actAs(g_1.\mathsf{n}_1, g_2)) \supset actAs(g_1.\mathsf{n}_1, g_2)$$

- SPKI/SDSI Speaks For.

$$actAs(p_2,\ p_1) \supset ((p_1 \mid\sim s) \supset (p_2 \mid\sim s))$$

- Modal Logic K Axiom.

$$(p_1 \mid\sim(s_1 \leftarrow s_2)) \supset ((p_1 \mid\sim s_2) \supset (p_1 \mid\sim s_2))$$

Consider the following DAL-like statements.

$$s_1 : ComA^{\mathcal{C}} \mid\sim actAs(ComA^{\mathcal{C}}.\mathsf{member},\ Bob^{\mathcal{I}})$$
$$s_2 : ComA^{\mathcal{C}} \mid\sim(T \leftarrow ComA^{\mathcal{C}}.\mathsf{member} \mid\sim T)$$
$$s_3 : ComB^{\mathcal{C}} \mid\sim(T \leftarrow Bob^{\mathcal{I}} \mid\sim T)$$
$$s_4 : Bob^{\mathcal{I}} \mid\sim(T \leftarrow Alice^{\mathcal{I}} \mid\sim T)$$

Statement s_1 states that $ComA^{\mathcal{C}}$ appoints $Bob^{\mathcal{I}}$ as $ComA^{\mathcal{C}}$'s member. Statement s_2 represents that $ComA^{\mathcal{C}}$ allows its member to issue purchase orders T, and they may also delegate this right to others. The content of permission T may be specified as Example 2.6 in SPKI/SDSI [7]. Note that T is not a DAL function: the permission originator is not specified in this permission. Suppose that $Bob^{\mathcal{I}}$ also works for another company $ComB^{\mathcal{C}}$ which also uses the T to issue purchase orders. Statement s_3 states that $ComB^{\mathcal{C}}$ allows $Bob^{\mathcal{I}}$ to issue purchase orders T, and $Bob^{\mathcal{I}}$ may also delegate this right to others. Suppose that $Bob^{\mathcal{I}}$ only receives s_3 from $ComB^{\mathcal{C}}$. Therefore, $Bob^{\mathcal{I}}$ issues s_4 to delegate this permission (issuing a purchase order) to a $ComB^{\mathcal{C}}$'s employee $Alice^{\mathcal{I}}$. $Bob^{\mathcal{I}}$ intends that $Alice^{\mathcal{I}}$ use s_3 and s_4 to obtain the permission from $ComB^{\mathcal{C}}$. However, $Alice^{\mathcal{I}}$ knows s_1 and s_2 and thus combines s_1, s_2, s_4 to prove that she may place a purchase order for $ComA^{\mathcal{C}}$. Both delegation chains (s_1, s_2, s_4) and (s_3, s_4) are provable using SPKI/SDSI axioms, but the intended permission originator is unclear. Note that if permission T was properly encoded as a DAL function then the DAL statements $s_1 \ldots s_4$ would be subterfuge-safe.

DAL provides a framework for specifying and reasoning about delegation and authorization in a manner that avoids subterfuge and has a number of novel features that make it particularly applicable to open systems. This is explored in the remainder of this section.

Atomic DAL permissions are globally unique. Existing Trust Management languages, such as SPKI/SDSI and

$\mathbf{R1}$: $(g \,|\!\sim s) \supset (g \mid\sim s)$
$\mathbf{R2}$: $(g_1 \mid\sim actAs(g_1,\ g_2)) \supset ((g_2 \mid\sim actAs(g_1,\ g_2)) \supset (actAs(g_1,\ g_2)))$
$\mathbf{R3}$: $(g_1 \mid\sim actAs(g_1.\mathsf{n}_1,\ g_2)) \supset ((g_2 \mid\sim actAs(g_1.\mathsf{n}_1,\ g_2)) \supset (actAs(g_1.\mathsf{n}_1,\ g_2)))$
$\mathbf{R4}$: $(g_1 \mid\sim actAs(g_1,\ g_2.\mathsf{n}_2)) \supset ((g_2 \mid\sim actAs(g_1,\ g_2.\mathsf{n}_2)) \supset (actAs(g_1,\ g_2.\mathsf{n}_2)))$
$\mathbf{R5}$: $(g_1 \mid\sim actAs(g_1.\mathsf{n}_1,\ g_2.\mathsf{n}_2)) \supset ((g_2 \mid\sim actAs(g_1.\mathsf{n}_1,\ g_2.\mathsf{n}_2)) \supset (actAs(g_1.\mathsf{n}_1,\ g_2.\mathsf{n}_2)))$
$\mathbf{R6}$: $(actAs(p_2,\ p_1)) \supset (actAs(p_3,\ p_2) \supset actAs(p_3,\ p_1))$
$\mathbf{R7}$: $actAs(p_1,\ p_1)$
$\mathbf{R8}$: $(actAs(p_2,\ p_1)) \supset ((p_1 \mid\sim p_2 \mid\sim s) \supset (p_2 \mid\sim s))$

Figure 1. New DAL axioms.

Keynote are designed to express arbitrary permissions. Writing a subterfuge-safe permission is dependent on the user's experience. Even though designed by experts, the KeyNote based payment systems [4, 5, 9] are vulnerable to authorisation subterfuge if care is not taken to properly identify the 'permissions' indicating the payment authorisations when multiple banks and/or provisioning agents are possible. Atomic DAL permissions are expressed by *globally unique functions*, whereby the identifier is the permission's originator (or the permission authority). The originator has the full authority to explain and manage its own permissions. Together with its own defined local function name, the identifier ensures that the permission is globally unique and the relationship between the permission and its originator is globally verifiable. For example, in statement $access(\mathsf{fileB}, \mathsf{B}^{\mathcal{I}})@A^{\mathcal{I}}$, $A^{\mathcal{I}}$ is the permission authority, and when this permission is delegated to other principals, all delegatees can verify that the permission is from $A^{\mathcal{I}}$.

DAL does not require global name services. A number of existing coalition frameworks rely on some form of global name provider to ensure that different parties get the right name for resources, and so forth. For example, Keynote [3] relies on IANA; RT [16] relies on ADSD; and X509 relies on the X500 name service. However, global name providers are not coalition security administrators; they only provide each name with a unique meaning and have no control over how names are used. Entities from different coalitions may still use arbitrary names to represent their own and the resources of others. While SPKI/SDSI does not rely on any global name provider, its name schema is subject to authorisation subterfuge. This can be explained as follows. When identifier g_1 binds its local name n_1 to g_2's local name n_2, the SPKI/SDSI 'linking' axiom does not require g_2 to be notified. Therefore, g_2 might not know that its n_2 is bound to g_1's local name, and may accidentally use n_2 for other purposes. DAL axioms **R2** – **R5** are used to relate principals. With these axioms, g_2 in the above example is required to accept a given name(role) binding. Therefore, g_2 may not use n_2 for other purpose. Thus, in DAL, binding roles(names) between different name spaces does not require global name providers, and can avoid authorisation subterfuge. For example,

$$A^{\mathcal{I}} \mid\sim (Read(\mathsf{fileB}, ?\mathsf{X})@A^{\mathcal{I}} \leftarrow {C_1}^{\mathcal{C}} \mid\sim actAs({C_1}^{\mathcal{C}}.\mathsf{manager}, ?\mathsf{X}) \wedge {C_2}^{\mathcal{C}} \mid\sim actAs({C_2}^{\mathcal{C}}.\mathsf{manager}, ?\mathsf{X}))$$

does not require that C_1 and C_2 agree on the meaning of the role manager. The meanings of "manager" in these separate coalitions C_1 and C_2 are decided within their own coalitions. $A^{\mathcal{I}}$ only needs to know the respective meanings of "manager" in these two separate coalitions C_1 and C_2.

DAL requires a principal to explicitly indicate its current role within the statement. Existing authorisation languages, such as SPKI/SDSI and Keynote use the 'speak-for' axiom, whereby if p_2 is bound to p_1, then any statement by p_1 is taken as a statement by p_2. However, a principal may represent different principals. For example, consider the following statements,

$$\begin{array}{l} actAs(C^{\mathcal{C}}.\mathsf{manager}, A^{\mathcal{I}}), \\ actAs(B^{\mathcal{I}}, A^{\mathcal{I}}), \\ C^{\mathcal{C}} \mid\sim (Read(\mathsf{fileB}, ?\mathsf{X})@C^{\mathcal{C}} \\ \qquad \leftarrow C^{\mathcal{C}}.\mathsf{manager} \mid\sim Read(\mathsf{fileB}, ?\mathsf{X})@C^{\mathcal{C}}), \\ A^{\mathcal{I}} \mid\sim Read(\mathsf{fileB},\ \mathsf{B}^{\mathcal{I}})@C^{\mathcal{C}} \end{array}$$

Using the SPKI/SDSI 'speak-for' axiom only, we can deduce: $C^{\mathcal{C}} \mid\sim Read(\mathsf{fileB},\ \mathsf{B}^{\mathcal{I}})@C^{\mathcal{C}}$, and $B^{\mathcal{I}} \mid\sim Read(\mathsf{fileB},\ \mathsf{B}^{\mathcal{I}})@C^{\mathcal{C}}$. The former statement specifies the policy that $C^{\mathcal{C}}$ authorises $B^{\mathcal{I}}$ to read fileB by $C^{\mathcal{C}}$'s local function. The latter statement defines the request made by $B^{\mathcal{I}}$ to read fileB by $C^{\mathcal{C}}$'s local function. In this scenario, it is not clear what $A^{\mathcal{I}}$ intends to do.

In order to provide more precise permissions, we use the axiom **R8** instead of the SPKI/SDSI 'speak-for' axiom. When a permission is delegated to a certain role, in order to validate the permission, DAL requires that principals must explicitly indicate that role in the statement. Otherwise, even though the permission is signed by the principal who obtains that role, the certificate is useless. It also means that an entity may only speak

in roles it knows. This requirement prevents principals from issuing ambiguous permissions. For example, in a coalition, all permissions for use only in the coalition should be delegated to coalition roles. If a principal wants to write a valid 'inner' coalition permission statement, it should know which role it is representing. Since this 'inner' coalition statement is invalid in other coalitions, the principal obtains other roles from other coalitions, and this statement is useless for use in other coalitions. Therefore, cross-coalition subterfuge may be prevented in DAL. Applying the DAL axioms to this example, it is not possible to derive statements $C^{\mathcal{C}} \mid\sim Read(\mathsf{fileB},\ \mathsf{B}^{\mathcal{I}})@C^{\mathcal{C}}$ and $B^{\mathcal{I}} \mid\sim Read(\mathsf{fileB},\ \mathsf{B}^{\mathcal{I}})@C^{\mathcal{C}}$. Presenting $A^{\mathcal{I}}$'s extended statement $A^{\mathcal{I}} \parallel\sim (C^{\mathcal{C}}.\mathsf{manager} \mid\sim Read(\mathsf{fileB},\ \mathsf{B}^{\mathcal{I}})@C^{\mathcal{C}})$, results in a policy that coalition $C^{\mathcal{C}}$ authorises $B^{\mathcal{I}}$ to read fileB by $C^{\mathcal{C}}$'s local function. Presenting $A^{\mathcal{I}} \parallel\sim (B^{\mathcal{I}} \mid\sim Read(\mathsf{fileB},\ \mathsf{B}^{\mathcal{I}})@C^{\mathcal{C}})$ results in a request by $B^{\mathcal{I}}$ to read fileB by $C^{\mathcal{C}}$'s local function.

DAL supports coalition delegation in addition to individual delegation. We give several statements that describe different categories of inner- and outer- coalition delegation.

- *Closed delegation*: only coalition participants may work within the coalition. Entities are required to obtain local roles in a closed coalition before they may use that coalition's resources. All 'inner' coalition permissions are delegated only to local roles of that coalition. For example, a closed delegation of coalition $C^{\mathcal{C}}$ may be defined by the statement

  $$C^{\mathcal{C}} \mid\sim (read(\mathsf{f_B}, ?\mathsf{X})@C^{\mathcal{C}} \leftarrow actAs(C^{\mathcal{C}}.\mathsf{member}, ?\mathsf{X}) \wedge ?\mathsf{X} \parallel\sim C^{\mathcal{C}}.\mathsf{member} \mid\sim read(\mathsf{f_B}, ?\mathsf{X})@C^{\mathcal{C}})$$

  This means that if a principal wants to operate file $\mathsf{f_B}$ by coalition $C^{\mathcal{C}}$'s local function $read$, it must be in the role member of $C^{\mathcal{C}}$, and must explicitly indicate the role that it is representing in its request (a 'directly-says' statement). Otherwise, the principal may not use that coalition's resource.

- *Semi-open delegation*: the resource of coalition ${C_1}^{\mathcal{C}}$ may be used by a group of principals from another coalition ${C_2}^{\mathcal{C}}$. These principals are not required to obtain any local role of coalition ${C_1}^{\mathcal{C}}$. However, they must obtain a specified role from coalition ${C_2}^{\mathcal{C}}$. In addition, these principals may not delegate their authority to others. In this case, all delegation decisions are still made by the resource owner coalition ${C_1}^{\mathcal{C}}$. The coalition ${C_2}^{\mathcal{C}}$ decides who may obtain its specified local role to grant the permission. An example of a semi-open delegation is given by the following statement.

  $${C_1}^{\mathcal{C}} \mid\sim (read(\mathsf{f}, ?\mathsf{X})@{C_1}^{\mathcal{C}} \leftarrow actAs({C_2}^{\mathcal{C}}.\mathsf{member}, ?\mathsf{X}) \wedge ?\mathsf{X} \parallel\sim {C_2}^{\mathcal{C}}.\mathsf{member} \mid\sim read(\mathsf{f}, ?\mathsf{X})@{C_1}^{\mathcal{C}})$$

  This means that ${C_2}^{\mathcal{C}}$'s members are authorised to use ${C_1}^{\mathcal{C}}$'s local resource when representing its membership of ${C_2}^{\mathcal{C}}$, but may not delegate it to others. Note that the directly-says sub-statement is made by ?X. This restricts ?X to identifiers because only identifiers (with their associated public key), may sign permissions.

- *Open delegation*: permissions are delegated from coalition ${C_1}^{\mathcal{C}}$ to a group of principals from another coalition ${C_2}^{\mathcal{C}}$. Unlike semi-open delegation, these principals may delegate their authority to others. For example, statement

  $${C_1}^{\mathcal{C}} \mid\sim (read(\mathsf{f}, ?\mathsf{X})@{C_1}^{\mathcal{C}} \leftarrow actAs({C_2}^{\mathcal{C}}.\mathsf{member}, ?\mathsf{X}) \wedge ?\mathsf{X} \mid\sim {C_2}^{\mathcal{C}}.\mathsf{member} \mid\sim read(\mathsf{f}, ?\mathsf{X})@{C_1}^{\mathcal{C}})$$

  means that ${C_1}^{\mathcal{C}}$ allows ${C_2}^{\mathcal{C}}$'s members to access file f by ${C_1}^{\mathcal{C}}$'s local function $read$, and ${C_2}^{\mathcal{C}}$'s members may also delegate the permission to others. For example, assuming that $actAs({C_2}^{\mathcal{C}}.\mathsf{member}, A^{\mathcal{I}})$ is true, then issuing the statement

  $$A^{\mathcal{I}} \mid\sim ({C_2}^{\mathcal{C}}.\mathsf{member} \mid\sim read(\mathsf{f}, ?\mathsf{X})@{C_1}^{\mathcal{C}} \leftarrow B^{\mathcal{I}} \parallel\sim read(\mathsf{f}, ?\mathsf{X})@{C_1}^{\mathcal{C}})$$

  means that ${C_2}^{\mathcal{C}}.\mathsf{member}$ $A^{\mathcal{I}}$ delegates the above statement to another principal $B^{\mathcal{I}}$. However, $A^{\mathcal{I}}$ still holds accountability for $B^{\mathcal{I}}$'s behavior regarding this delegated permission. This is because, given the request made by $B^{\mathcal{I}}$, we may deduce that $A^{\mathcal{I}}$ makes the request in the role ${C_2}^{\mathcal{C}}.\mathsf{member}$. Note that ${C_1}^{\mathcal{C}}$'s policy statement has a 'says' sub-statement, which is unlike the 'directly-says' used in semi-open delegation.

- *Cross coalition delegation*: permissions may be delegated from one coalition ${C_1}^{\mathcal{C}}$ to another coalition ${C_2}^{\mathcal{C}}$ which, in turn, may independently decide who may use the permission. For example,

  $${C_1}^{\mathcal{C}} \mid\sim (read(\mathsf{f}, ?\mathsf{X})@{C_1}^{\mathcal{C}} \leftarrow {C_2}^{\mathcal{C}} \mid\sim read(\mathsf{f}, ?\mathsf{X})@{C_1}^{\mathcal{C}})$$

DAL does not support complex principal expressions, such as SPKI/SDSI compound names. The compound name `UnivA's professor's student` represents the local name `student` in all the local name spaces of `UnivA`'s `professor`. This kind of local name requires that all `UnivA's professors` must have the same meaning for their local role `student` based on `UnivA`'s understanding. We argue that this is unreasonable in many scenarios. For example, before joining `UnivA` as a `professor`, Alice may already define her local name `student` with its own meaning. According to the conflicting local name meanings, Alice may not join `UnivA`, or may have to

adapt the meaning of `student` according to `UnivA`'s local name space and withdraw all certificates related to the meaning of her own `student`. On the other hand, when Alice joins `UnivA`, she may not be aware of the name conflict, which may lead to further unexpected authorisations/delegations subterfuge. If `UnivA` wishes to delegate its permission to `UnivA's professor's student`, it should ensure that the name `student` has the same meaning for all related principals' local name space. In practice, these are unreasonable solutions. DAL, in avoiding the use of complex role principals, permits a local name to resolve only to an identifier.

4 The Basic Coalition Framework

DAL is used to make statements regarding relationships between principals including coalition membership. In this section we consider the process by which new coalitions are negotiated and formed. To establish a purpose-independent coalition, intended participants should understand and obey a small number of basic coalition regulations. To interact with such coalitions, principals should know how the coalition works in order to correctly negotiate with the coalition. The regulations that are used to characterise the nature of a coalition and its normal operation are specified in terms of DAL. In designing DAL, our motivation has been to provide a safe language that has been tailored for open delegation in coalitions. While other languages such as [17, 7, 6, 16] offer comparable levels of expressiveness to DAL, credentials written in these languages require formal analysis [20] and/or pre-agreed global naming services to ensure subterfuge-safe delegation. Like type-safe languages, DAL is intended to provide flexibility while preventing classes of unsafe formulae from being encoded.

In existing frameworks [2, 12, 13, 18, 19], coalitions may or may not have an associated signing key. If a coalition does not have its own signature key, then the coalition may not be uniquely identified over the network. In this case an outsider cannot verify whether a statement is from the coalition and cross-coalition operations may not be possible. When a coalition has a signature key, its "super" security administrator controls that key. Other coalition participants cannot prevent the administrator from arbitrarily signing any statement using that key. Furthermore, if the coalition signing key is compromised then the coalition has to be reformed.

In our coalition framework, a coalition has a unique identifier that includes its signature key as defined in Section 2. The purpose of the coalition key is to sign the initial coalition regulations during the establishment of the coalition and is not intended for any other purpose. The coalition key is generated and initially held by a trusted principal (the constructor) of the proposed coalition. The constructor is selected by the coalition founders and may be a trusted external third party or intended member of the coalition.

Once the initial coalition regulations have been signed and the coalition established, the coalition key should not be used for further signing. In establishing a coalition, the constructor signs a penalty contract accepting responsibility for the proper use of the signing key. If the key is misused then the constructor becomes liable under the terms of the contract. The coalition regulations are such that it is not possible to establish a coalition without signing this contract. In practice, it is expected that having established the coalition, the constructor will destroy the (ephemeral private) coalition key in order to avoid accidental compromise. This coalition establishing framework does not depend on any particular threshold cryptography scheme and, therefore, the users of this framework are free to use the cryptographic algorithms of their choice.

The following predefined global functions are used in establishing coalitions. The reader is reminded that it is only the function schema that are global; their (decentralized) definitions are provided by DAL statements made by participating principals. The coalition establishing process relies on this small number of global functions to bootstrap subterfuge-safe coalitions. DAL ensures that all other names and permissions that are used within the coalitions are managed in a subterfuge-safe way and do not require a global name service such as X.500.

- Function $Pay(amount,\ unit,\ payer,\ payee)$ means that the principal payer is willing to pay a certain amount of money to another principal payee. For example, $Pay(500, USD, A^{\mathcal{I}}, B^{\mathcal{I}})$ means that $A^{\mathcal{I}}$ is willing to pay \$500 to $B^{\mathcal{I}}$. When the payee is a coalition role then the mechanism does not specify how the payment should be divided among coalition partners.
- Given two valid statements s_1 and s_2, then function $neq(s_1, s_2)$ represents a fact within the logic as to whether s_1 is not equal to s_2.

4.1 Core Coalition Regulations

Every valid coalition must include two core statements which act as the basic regulation for the coalition. These core statements have the following Coalition Formation (**CF**) patterns.

CF$_1$ A coalition ${id_1}^{\mathcal{C}}$ is formed by an individual ${id_2}^{\mathcal{I}}$ in the role ${\mathsf{id}_1}^{\mathcal{C}}.\mathsf{constructor}$ with the agreement of all founding principals. The responsibility of ${id_2}^{\mathcal{I}}$ extends to the formation of the coalition, but no further.

$$
\begin{aligned}
{id_1}^{\mathcal{C}} \mid\!\sim (& actAs({id_1}^{\mathcal{C}}.\mathsf{constructor}, {id_2}^{\mathcal{I}}) \\
& \wedge actAs({id_1}^{\mathcal{C}}.\mathsf{r_1}, [p_1, \ldots, p_\mathsf{n}]) \\
& \wedge (?\mathsf{X} \leftarrow \mathsf{threshold}(n,\ {id_1}^{\mathcal{C}}.\mathsf{r_1}) \mid\!\sim ?\mathsf{X}))
\end{aligned}
$$

This statement from $id_1{}^{\mathcal{C}}$, is interpreted as follows,

1. $id_1{}^{\mathcal{C}} \mid\sim actAs(id_1{}^{\mathcal{C}}.\mathsf{constructor},\ id_2{}^{\mathcal{I}})$. This statement indicates that the coalition $id_1{}^{\mathcal{C}}$ claims the individual $id_2{}^{\mathcal{I}}$ is its constructor and holds its coalition key. After obtaining this information, the principal who wants to join $id_1{}^{\mathcal{C}}$, or collaborate with the coalition, may require further insurance information ($\mathbf{CF_2}$) from $id_2{}^{\mathcal{I}}$.
2. $id_1{}^{\mathcal{C}} \mid\sim actAs(id_1{}^{\mathcal{C}}.\mathsf{r}_1, [p_1, \ldots, p_n])$. Here, $id_1{}^{\mathcal{C}}$ defines the role $id_1{}^{\mathcal{C}}.\mathsf{r}_1$ in $id_1{}^{\mathcal{C}}$ when $id_1{}^{\mathcal{C}}$ established, and $p_i, i \in [1, n]$ are the only initial principals for this role. The role $id_1{}^{\mathcal{C}}.\mathsf{r}_1$ represents the most important role in the coalition, for example, a role to which founders belong.
3. $id_1{}^{\mathcal{C}} \mid\sim(?\mathsf{X} \leftarrow \mathsf{threshold}(n,\ \ id_1{}^{\mathcal{C}}.\mathsf{r}_1) \mid\sim ?\mathsf{X})$. When a principal accepts the role $id_1{}^{\mathcal{C}}.\mathsf{r}_1$, it needs to be sure that it does not delegate any unintended rights, that no one can force it to accept conditions it does not want to accept, such as sharing its own resources without its permission and so forth, as a consequence of joining a coalition.

   Only when n initial principals of this coalition agree on a decision, may the coalition make this decision. In other-words, if any one playing the role $id_1{}^{\mathcal{C}}.\mathsf{r}_1$ does not agree on a decision, then the coalition may not be established. If the coalition key does not sign any other credentials, then this statement guarantees that the $id_1{}^{\mathcal{C}}$ authority is distributed to all initial principals who are playing the role $id_1{}^{\mathcal{C}}.\mathsf{r}_1$.

$\mathbf{CF_2}$ Coalition constructor $id_2{}^{\mathcal{I}}$ agrees to a penalty contract regarding proper use of the coalition key.

$$id_2{}^{\mathcal{I}} \mid\sim(Pay(AMT, U, id_2{}^{\mathcal{I}}, id_1{}^{\mathcal{C}}.\mathsf{r}_2) \leftarrow neq(id_1{}^{\mathcal{C}} \mid\sim ?\mathsf{Y},\ \mathbf{CF_1}))$$

This is a necessary part of a properly formed coalition and provides evidence that, if $id_1{}^{\mathcal{C}}$ signs any statement other than founding regulation $\mathbf{CF_1}$ above, then $id_2{}^{\mathcal{I}}$ is willing to pay a penalty amount AMT in the currency U to the principals in the oversight role $id_1{}^{\mathcal{C}}.\mathsf{r}_2$.

Note that the oversight role $id_1{}^{\mathcal{C}}.\mathsf{r}_2$ may be different to the founding role $id_1{}^{\mathcal{C}}.\mathsf{r}_1$. The presence of some penalty regulation clause is sufficient in a valid coalition; whether it is acceptable is part of the coalition establishment process with the founders and is considered in Section 4.2.

The signature keys for $id_1{}^{\mathcal{C}}$ and $id_2{}^{\mathcal{I}}$ are known only by $id_2{}^{\mathcal{I}}$. Therefore, $id_2{}^{\mathcal{I}}$ is the only principal who can generate the necessary regulation credentials above. The signature key for $id_1{}^{\mathcal{C}}$ should only be used for establishing the coalition; to use it for any other action results in a penalty on $id_2{}^{\mathcal{I}}$ according to $\mathbf{CF_2}$.

Example 1 $A^{\mathcal{I}}$, $B^{\mathcal{I}}$, $C^{\mathcal{I}}$ wish to form a coalition $M^{\mathcal{C}}$. They will be the $M^{\mathcal{C}}$'s founders. They all trust a third party $TTP^{\mathcal{I}}$ to generate the basic coalition regulations so long as $TTP^{\mathcal{I}}$ is willing to promise that if $TTP^{\mathcal{I}}$ misuses the coalition key, then it will pay \$50 collective penalty to principals in the role $\mathsf{M}^{\mathcal{C}}.\mathsf{oversight}$[2]. The regulations generated are as follows.

$$\mathbf{CF_1} \triangleq M^{\mathcal{C}} \mid\sim(actAs(M^{\mathcal{C}}.\mathsf{constructor}, TTP^{\mathcal{I}}) \wedge actAs(M^{\mathcal{C}}.\mathsf{founder},\ [A^{\mathcal{I}}, B^{\mathcal{I}}, C^{\mathcal{I}}])) \wedge (?\mathsf{X} \leftarrow \mathsf{threshold}(3,\ M^{\mathcal{C}}.\mathsf{founder}) \mid\sim ?\mathsf{X}))$$

$$\mathbf{CF_2} \triangleq TTP^{\mathcal{I}} \mid\sim(Pay(50, USD, TTP^{\mathcal{I}}, M^{\mathcal{C}}.\mathsf{oversight}) \leftarrow neq(M^{\mathcal{C}} \mid\sim ?\mathsf{Y}, \mathbf{CF_1}))$$

△

4.2 Coalition Establishment Process

Establishing a new coalition requires cooperation and agreement between the coalition constructor and all of the coalition founders. This Coalition Establishment (**CE**) process involves three steps. Given $\mathbf{i}, \mathbf{j} \in [1, n]$:

$\mathbf{CE}_1$ $id_2{}^{\mathcal{I}} \rightarrow p_i : \mathbf{CF_1} \wedge \mathbf{CF_2}$;

Each principal p_i invited to join the coalition's founding role $id_1{}^{\mathcal{C}}.\mathsf{r}_1$ receives and checks the two regulations generated from $id_2{}^{\mathcal{I}}$. After signing $\mathbf{CF_1}$ it will not be in the interest of $id_1{}^{\mathcal{C}}$ to sign further credentials, due to the penalty signed by $id_2{}^{\mathcal{I}}$ (in $\mathbf{CF_2}$).

$\mathbf{CE}_2$ $p_i \rightarrow id_2{}^{\mathcal{I}}$: $p_i \mid\sim(actAs(id_1{}^{\mathcal{C}}.\mathsf{r}_1, p_i) \wedge \mathbf{CF_1} \wedge \mathbf{CF_2})$;

Each founder p_i that accepts the coalition regulations signs an agreement on the regulations and its role membership. Before engaging this step, coalition founders may communicate to consider and informally agree the regulations.

$\mathbf{CE}_3$ $p_i \rightarrow p_j$: $p_i \mid\sim id_1{}^{\mathcal{C}}.\mathsf{r}_1 \mid\sim actAs(id_1{}^{\mathcal{C}}.\mathsf{r}_2, id_1{}^{\mathcal{C}}.\mathsf{r}_1)$.

The final step in the process requires the founders to agree the enforcement role, that is, the oversight role to which payment will be made if the coalition constructor breaks the regulations of the coalition.

After establishing a coalition, anyone can trust $id_1{}^{\mathcal{C}}$ or ask its constructor $id_2{}^{\mathcal{I}}$ to provide all above credentials as evidence for the coalition establishment.

[2]Note the founders may also be willing to pay for the service of a properly generated coalition. For reasons of space, we do not consider this form of coalition regulation; it is the topic of a further paper

Example 2 (Security Group) This is a centralized coalition, in which the constructor $Alice^{\mathcal{I}}$ is also the founder and controls all authority of the coalition $sec^{\mathcal{C}}$. Alice follows the three protocol steps defined above.

Alice acting as coalition constructor sends the founding regulations to herself (in role $sec^{\mathcal{C}}.\mathsf{head}$).

$$\begin{aligned}&\mathbf{CF_{sec1}} \hat{=}\\ &\quad sec^{\mathcal{C}} \,|\!\sim\!(actAs(sec^{\mathcal{C}}.\mathsf{constructor},\ Alice^{\mathcal{I}})\\ &\qquad\quad \wedge\, actAs(sec^{\mathcal{C}}.\mathsf{head},\ Alice^{\mathcal{I}})\\ &\qquad\quad \wedge\, (?\mathsf{X} \leftarrow \mathsf{threshold}(1,\ sec^{\mathcal{C}}.\mathsf{head}) \,|\!\sim ?\mathsf{X}))\end{aligned}$$

$$\begin{aligned}&\mathbf{CF_{sec2}} \hat{=}\\ &\quad Alice^{\mathcal{I}} \,|\!\sim\!(Pay(500,\ USD,\ Alice^{\mathcal{I}},\ sec^{\mathcal{C}}.\mathsf{oversight})\\ &\qquad\quad \leftarrow neq(sec^{\mathcal{C}} \,|\!\sim ?\mathsf{Y},\ \mathbf{CF_{sec1}}))\end{aligned}$$

Alice accepts founding role $sec^{\mathcal{C}}.\mathsf{head}$ defined by herself:

$$Alice^{\mathcal{I}} \,|\!\sim\!(actAs(sec^{\mathcal{C}}.\mathsf{head},\ Alice^{\mathcal{I}}) \wedge \mathbf{CF_{sec1}} \wedge \mathbf{CF_{sec2}})$$

Alice specifies that role $sec^{\mathcal{C}}.\mathsf{oversight}$ is the oversight role:

$$Alice^{\mathcal{I}} \,|\!\sim sec^{\mathcal{C}}.\mathsf{head} \,|\!\sim actAs(sec^{\mathcal{C}}.\mathsf{oversight}, sec^{\mathcal{C}}.\mathsf{head})$$

At this point $Alice^{\mathcal{I}}$ is the constructor, founder and sole member of the coalition $sec^{\mathcal{C}}$. She introduces a new role $sec^{\mathcal{C}}.\mathsf{member}$ for members and assigns $Bob^{\mathcal{I}}$ to that role:

$$Alice^{\mathcal{I}} \,|\!\sim sec^{\mathcal{C}}.\mathsf{head} \,|\!\sim actAs(sec^{\mathcal{C}}.\mathsf{member},\ Bob^{\mathcal{I}})$$

$Alice^{\mathcal{I}}$ assigns all group members to the oversight role:

$$Alice^{\mathcal{I}} \,|\!\sim sec^{\mathcal{C}}.\mathsf{head} \,|\!\sim actAs(sec^{\mathcal{C}}.\mathsf{oversight}, sec^{\mathcal{C}}.\mathsf{member})$$

If $Bob^{\mathcal{I}}$ is satisfied with the regulation on oversight then he is willing to participate within the coalition and signs

$$Bob^{\mathcal{I}} \,|\!\sim actAs(sec^{\mathcal{C}}.\mathsf{member},\ Bob^{\mathcal{I}})$$

and $Bob^{\mathcal{I}}$ can now speak as a member of the coalition. $\triangle$

Appendix B provides further examples of coalition establishment, coalition merging, and coalition spawning.

4.3 Security Analysis

The coalition establishment process is secure, in the sense that no principal can cheat or mis-represent the coalition. We provide a security analysis of coalition establishment; for reasons of space, the analysis is informal.

This coalition establishment process relies on three roles: constructor, founder, and oversight. Role constructor is a predefined role name for all coalitions, making it clear who creates the coalition. The founders and oversight are user-definable role names. We analyse the accountability of each role as follows.

Regulation $\mathbf{CF_1}$ is a prearranged agreement by all founders regarding coalition structure and initial participants. $\mathbf{CF_2}$ is a prearranged penalty contract agreed by the coalition constructor and all of the coalition founders. All founders knows the constructor's public key and agree that the constructor generates the basic coalition regulations.

Only the constructor knows the coalition and constructor signature (private) keys. He is therefore the only principal that can generate and accept accountability for $\mathbf{CF_1}$ and $\mathbf{CF_2}$. If a valid constructor generates just one version of $\mathbf{CF_1} \wedge \mathbf{CF_2}$ then it is not possible to deduce $neq({id_1}^{\mathcal{C}} \,|\!\sim ?X, \mathbf{CF_1})$ and consequently it is not possible to deduce $id_2{}^{\mathcal{I}} \,|\!\sim pay(AMT, U, id_2{}^{\mathcal{I}}, {id_1}^{\mathcal{C}}.\mathsf{r}_2)$, that is, a valid constructor cannot be penalised using $\mathbf{CF_2}$.

Founders are willing to accept some pre-agreed $\mathbf{CF_1}$ and $\mathbf{CF_2}$ when establishing the coalition. If a constructor (or any principal masquerading as constructor) generates regulations $\mathbf{CF_1}$ and $\mathbf{CF_2}$ that are not acceptable to the founders, then the coalition establishment process stops at Step $\mathbf{CE}_2$.

If the coalition constructor attempts to mislead principals by generating different versions of $\mathbf{CF_1}$ for founders and other participants, then upon detection the penalty regulation can be applied. Similarly, once a coalition has been established, if the constructor attempts to speak for the coalition (using the coalition key), then upon detection the penalty regulation may be applied. Thus, a constructor cannot misrepresent a coalition without the application of the penalty regulation. On the other hand, a valid constructor cannot be penalised.

Having completed Step $\mathbf{CE}_2$, the founders can collectively speak for the coalition. However, each founder is not willing to participate until it has received acceptable declaration concerning the oversight role (Step $\mathbf{CE}_3$). This agreement is also necessary before the penalty regulation can be properly applied and effectively protects the constructor if a coalition is not properly established.

Before accepting the role founder, coalition founders may informally negotiate with each other to determine whether they received the same message. Any founder may stop the establishment process, if it received an unexpected $\mathbf{CF_1}$ or $\mathbf{CF_2}$.

The threshold structure in $\mathbf{CF_1}$ ensures that agreement is required between all founders before Step $\mathbf{CE}_2$ can be successfully completed: it is not possible to establish a coalition without the agreement of all founders. Once established all the founders share authority of the coalition.

The oversight role does not have any inherent authority other than authority over the penalty. However, when a principal accepts this role, it may protect itself by keeping necessary evidence to carry out the penalty contract.

4.4 Discussion

The coalition forming framework has a number of characteristics.

- *Authentication.* Each individual and coalition has a unique signature key. Every coalition and individual can be uniquely authenticated.
- *Delegation*. DAL can express a wide range of delegation actions, including, identity-based, role-based, conjunction, cross-coalition and static and dynamic threshold-based delegation.
- *Decentralisation.* Our framework does not need a centralized or 'super' coalition administrator. Once a coalition is established, then all the authority of the coalition lies with the founders who can create and regulate their own coalition structure. If some of the coalition founders' signature keys are compromised, then the other coalition founders may choose to no longer issue further regulations in order to protect the coalition. However, revocation of compromised keys and revocation of regulations that have been issued with the participation of compromised keys remain an issue. The focus in this paper is on providing a logic and framework for establishing coalitions and we do not consider revocation in this paper.
- *Accountability*. Principals wishing to negotiate with a coalition check the coalition's two regulations. Following delegation of coalition authority, principals determine whether they are engaging with the right role/principal in that coalition. If incorrect regulations are received then they can be kept as evidence. The constructor has accountability for incorrect regulations. This provides autonomy and self-determination.
- *Maintainability.* Once the coalition is created the coalition (signing) key should be destroyed by the constructor; the coalition key provides a unique identity and is used to validate the basic regulations. Founders can introduce further regulations (speaking for the coalition) that control existing and new participants. Principals, including the founders, may also leave a coalition under regulations that were agreed upon by the other founders.
- *Fairness.* No principal has advantages over any other during or after the coalition establishment process. This fair coalition establishing process does not require any trusted third party who has total power over issuance of certificates.
- *Non-repudiation.* We consider only the simple meaning of non-repudiation, that is, once a statement is signed by a principal then it may not subsequently deny that statement. We do not consider non-repudiation of recipient.
- *Dynamic establishment.* Since coalitions are principals then the establishment process can be used by coalitions to form further coalitions.

5 Conclusion

In this paper a logic-based language Distributed Authorization Language (DAL) is proposed that supports open delegation in large scale distributed systems. ¿From the outset, DAL has been designed with open systems in mind; flexible cross-domain delegation can be achieved without subterfuge and without having to rely on the proper use of a global name service.

Using DAL, a formal framework for regulating the establishment of dynamic coalitions is proposed. Coalitions are formed with the involvement of founders, constructors and oversight and do not rely on the traditional notion of a "super" administrator. Constructors are responsible for properly creating a coalition; this service can be provided by a third party. If the service is improperly provided then the constructor is subject to a penalty, which may be collected by another third party providing oversight. With this framework, a coalition can be dynamically formed in a fully distributed manner without relying on a "super" security administrator or any particular threshold cryptography algorithms. We are currently using DAL and the coalition establishment process to develop support for Virtual Organizations in GRIDs.

As with any protection framework the challenge is to make sure that it provides some useful and consistent notion of security. Assurance is required that a principal cannot bypass security via some unexpected but authorised route. In the case of DAL, we seek formal proof that it is a subterfuge-safe language. It is argued in [8] that verifying whether a particular configuration of access controls is effective can be achieved by analysing its consistency, that is, whether it is possible for a malicious principle to interfere with the the normal operation of the system. This type of analysis [8, 10] is not unlike the analysis carried out on authentication protocols. In the case of mechanisms based on trust management schemes, such as DAL, it is a question of ensuring consistency between potential delegation chains. We are currently developing a non-interference verification for subterfuge-safety in DAL based on [8, 10].

Acknowledgements

We are grateful for helpful feedback from the anonymous referees. This work is supported by Enterprise Ireland Basic Research Grant (SC/2003/007).

References

[1] M. Abadi. On sdsi's linked local name spaces. In *Proceedings of the 10th Computer Security Foundations Workshop (CSFW '97)*, page 98, Washington, DC, USA, 1997. IEEE Computer Society.

[2] T. Aura and S. Mäki. Towards a survivable security architecture for ad-hoc networks. In *Proc. Security Protocols, 9th International Workshop*, volume 2467 of *LNCS*, pages 63–79, Cambridge, UK, Apr. 2001. Springer.

[3] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The keynote trust-management system, version 2, September 1999.

[4] M. Blaze, J. Ioannidis, S. Ioannidis, A. Keromytis, P. Nikander, and V. Prevelakis. Tapi: Transactions for accessing public infrastructure. In *Proceedings of the 8th IFIP Personal Wireless Communications (PWC) Conference*, 2003.

[5] M. Blaze, J. Ioannidis, and A. D. Keromytis. Offline micropayments without trusted hardware. In *Financial Cryptography*, Grand Cayman, February 2001.

[6] J. DeTreville. Binder, a logic-based security language. In *Proceedings of the 2002 IEEE Symposium on Research in Security and Privacy*, pages 105–113. IEEE Computer Society Press, 2002.

[7] C. Ellison, B. Frantz, B. Lampson, R. L. Rivest, B. Thomas, and T. Ylonen. Spki certificate theory, September 1999.

[8] S. Foley. A non-functional approach to system integrity. *IEEE Journal on Selected Areas in Communications*, 21(1), Jan 2003.

[9] S. Foley. Using trust management to support transferable hash-based micropayments. In *Proceedings of the 7th International Financial Cryptography Conference*, Gosier, Guadeloupe, FWI, January 2003.

[10] S. Foley. Believing in the integrity of a system. In *IJCAR Workshop on Automated Reasoning for Security Protocol Analysis*. Springer Verlag Electronic Notes in Computer Science, 2004.

[11] S. N. Foley and H. Zhou. Authorisation subterfuge by delegation in decentralised networks. In *International Security Protocols Workshop*, Cambridge, UK, April 2005.

[12] I. Foster, C. Kesselman, and G. Tsudik. A security architecture for computational grids. In *Proceedings of ACM Conference on Computers And Security*, pages 83–91. ACM Press, Oct. 1998.

[13] L. Gong. Enclaves: Enabling secure collaboration over the internet. *IEEE Journal on Selected Areas in Communications*, 15(3):567–575, 1997.

[14] R. Housley, W. Polk, W. Ford, and D. Solo. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile, April 2002.

[15] Internet Assigned Numbers Authority. Internet corporation for assigned names and numbers, May 1999.

[16] N. Li and J. C. Mitchell. RT: A role-based trust-management framework. In *The Third DARPA Information Survivability Conference and Exposition (DISCEX III)*, pages 201–212, Washington, D.C., April 2003. IEEE Computer Society Press, Los Alamitos, California.

[17] N. Li, W. H. Winsborough, and J. C. Mitchell. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security (TISSEC)*, 6(1):128–171, February 2003.

[18] N. H. Minsky and V. Ungureanu. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *ACM Transactions on Software Engineering and Methodology*, 9(3):273–305, 2000.

[19] L. Pearlman, C. Kesselman, V. Welch, I. Foster, and S. Tuecke. The community authorization service: Status and future. In *CHEP03*, La Jolla, California, March 2003.

[20] H. Zhou and S. N. Foley. A logic for analysing subterfuge in delegation chains. In *Workshop on Formal Aspects in Security and Trust (FAST2005)*, Newcastle upon Tyne, UK, July 2005.

A Encoding Threshold Principals in DAL

The static threshold principal $\mathsf{threshold}(k,\ [p_1, \ldots, p_n])$ can be simply encoded in the usual way in terms of the logic operators $\wedge$ and $\neg$. For example, given principals p_1 and p_2, then $\mathsf{threshold}(2,\ [p_1, p_2])$ and $\mathsf{threshold}(1,\ [p_1, p_2])$ may be represented by $p_1 \wedge p_2$ and $p_1 \vee p_2$, respectively.

A dynamic threshold principal $\mathsf{threshold}(n,\ g.\mathsf{r})$ means that at least n principals in the role $g.\mathsf{r}$ are required. A threshold may appear only as a principal in a statement, that is, in a statement of the form $\mathsf{threshold}(n,\ g.\mathsf{r}) \mid\sim s$ for some arbitrary statement s. A statement $\mathsf{threshold}(n,\ g.\mathsf{r}) \mid\sim s$ is encoded by the logic statement

$$(?X_1 \mid\sim g.\mathsf{r} \mid\sim s) \wedge \ldots \wedge (?X_n \mid\sim g.\mathsf{r} \mid\sim s)$$

where each of the $?X_i$ $(1 \leq i \leq n)$ are distinct principals. A function that is similar to $neq(s_1, s_2)$ defined in Section 4 can be defined to distinguish principals and this function can be used to encode $neq(?X_i, ?X_j)$ for the above formula with $0 \leq i, j \leq n$ and $i \neq j$.

The global uniqueness of signature keys ensure that identifiers and roles are distinguishable from each other. Roles are distinguishable between each other in statements because DAL does not have a SPKI-like rule of the form $actAs(p_2,\ p_1) \supset (p_1 \mid\sim s \supset p_2 \mid\sim s)$. Instead, DAL uses axiom **R8** to force a principal to provide an unambiguous role name in the statements clarifying the current role that it is acting as. For any two different roles $g_1.\mathsf{r}_1$ and $g_2.\mathsf{r}_2$, the statement $g_1.\mathsf{r}_1 \mid\sim s$ may not be deduced from $g_2.\mathsf{r}_2 \mid\sim s$. For example, given $actAs(B^{\mathcal{C}}.\mathsf{student}, A^{\mathcal{I}})$, then statements $A^{\mathcal{I}} \mid\sim s$ and $A^{\mathcal{I}} \mid\sim B^{\mathcal{C}}.\mathsf{student} \mid\sim s$ are different, since from the latter we may deduce $B^{\mathcal{C}}.\mathsf{student} \mid\sim s$, which cannot be deduced from the former $A^{\mathcal{I}} \mid\sim s$ in DAL.

B Coalition Establishing Examples

GRID Group This is a decentralized coalition, in which the constructor $John^{\mathcal{I}}$, together with other two individuals $Ellen^{\mathcal{I}}$ and $Alice^{\mathcal{I}}$, are the founders of the coalition $grid^{\mathcal{C}}$. These three individuals control all $grid^{\mathcal{C}}$'s authority. We have: $John^{\mathcal{I}} \rightarrow Ellen^{\mathcal{I}}, Alice^{\mathcal{I}} : \mathbf{CF_{grid1}} \wedge \mathbf{CF_{grid2}}$, where,

$$\begin{aligned}\mathbf{CF_{grid1}} \;\hat{=}\; & grid^{\mathcal{C}} \,|\!\sim\!(actAs(grid^{\mathcal{C}}.\mathsf{constructor},\ John^{\mathcal{I}}) \wedge actAs(grid^{\mathcal{C}}.\mathsf{committee}, [Ellen^{\mathcal{I}}, John^{\mathcal{I}}, Alice]^{\mathcal{I}}) \\ & \wedge (?\mathsf{X} \leftarrow \mathsf{threshold}(3,\ grid^{\mathcal{C}}.\mathsf{committee}) \,|\!\sim\, ?\mathsf{X}))\end{aligned}$$

$$\mathbf{CF_{grid2}} \;\hat{=}\; John^{\mathcal{I}} \,|\!\sim\!(Pay(500, USD, John^{\mathcal{I}}, grid^{\mathcal{C}}.\mathsf{oversight}) \leftarrow neq(grid^{\mathcal{C}} \,|\!\sim\, ?\mathsf{Y}, \mathbf{CF_{grid1}}))$$

Alice, Ellen, and John accept the founding role $grid^{\mathcal{C}}.\mathsf{committee}$ of coalition $grid^{\mathcal{C}}$:

$$\begin{aligned}Ellen^{\mathcal{I}} \rightarrow John^{\mathcal{I}} \;&:\; Ellen^{\mathcal{I}} \,|\!\sim\!(actAs(grid^{\mathcal{C}}.\mathsf{committee},\ Ellen^{\mathcal{I}}) \wedge \mathbf{CF_{grid1}} \wedge \mathbf{CF_{grid2}}) \\ John^{\mathcal{I}} \rightarrow John^{\mathcal{I}} \;&:\; John^{\mathcal{I}} \,|\!\sim\!(actAs(grid^{\mathcal{C}}.\mathsf{committee},\ John^{\mathcal{I}}) \wedge \mathbf{CF_{grid1}} \wedge \mathbf{CF_{grid2}}) \\ Alice^{\mathcal{I}} \rightarrow John^{\mathcal{I}} \;&:\; Alice^{\mathcal{I}} \,|\!\sim\!(actAs(grid^{\mathcal{C}}.\mathsf{committee},\ Alice^{\mathcal{I}}) \wedge \mathbf{CF_{grid1}} \wedge \mathbf{CF_{grid2}})\end{aligned}$$

Alice, Ellen, and John specify that role $\mathsf{grid}^{\mathcal{C}}, \mathsf{oversight}$ is the oversight role:

$$\begin{aligned}Ellen^{\mathcal{I}} \rightarrow John^{\mathcal{I}}, Alice^{\mathcal{I}} \;&:\; Ellen^{\mathcal{I}} \,|\!\sim\, grid^{\mathcal{C}}.\mathsf{committee} \,|\!\sim\, actAs(grid^{\mathcal{C}}.\mathsf{oversight}, grid^{\mathcal{C}}.\mathsf{committee}) \\ John^{\mathcal{I}} \rightarrow Alice^{\mathcal{I}}, Ellen^{\mathcal{I}} \;&:\; John^{\mathcal{I}} \,|\!\sim\, \mathsf{grid}^{\mathcal{C}}, \mathsf{committee} \,|\!\sim\, actAs(grid^{\mathcal{C}}.\mathsf{oversight}, grid^{\mathcal{C}}.\mathsf{committee}) \\ Alice^{\mathcal{I}} \rightarrow Ellen^{\mathcal{I}}, John^{\mathcal{I}} \;&:\; Alice^{\mathcal{I}} \,|\!\sim\, grid^{\mathcal{C}}.\mathsf{committee} \,|\!\sim\, actAs(grid^{\mathcal{C}}.\mathsf{oversight}, grid^{\mathcal{C}}.\mathsf{committee})\end{aligned}$$

Everyone can verify that $grid^{\mathcal{C}}$ has been established properly. The committee member role $\mathsf{grid}^{\mathcal{C}}, \mathsf{committee}$ share all $grid^{\mathcal{C}}$'s authority. All new regulations must be certificated by all of them.

Spawning a New Web-Grid Group $grid^{\mathcal{C}}$ wants to spawn a further group $web^{\mathcal{C}}$ to manage the Web GRID project. $grid^{\mathcal{C}}$ director holds all authority of $web^{\mathcal{C}}$.
Firstly, John acting as the constructor sends the founding regulations to himself (in role $web^{\mathcal{C}}.\mathsf{founder}$):

$$\begin{aligned}\mathbf{CF_{web1}} \;\hat{=}\; & web^{\mathcal{C}} \,|\!\sim\!(actAs(web^{\mathcal{C}}.\mathsf{constructor},\ John^{\mathcal{I}}) \wedge actAs(web^{\mathcal{C}}.\mathsf{founder},\ grid^{\mathcal{C}}.\mathsf{director}) \\ & \wedge (?\mathsf{X} \leftarrow \mathsf{threshold}(1,\ web^{\mathcal{C}}.\mathsf{founder}) \,|\!\sim\, ?\mathsf{X}))\end{aligned}$$

$$\mathbf{CF_{web2}} \;\hat{=}\; John^{\mathcal{I}} \,|\!\sim\!(Pay(500,\ USD,\ John^{\mathcal{I}},\ web^{\mathcal{C}}.\mathsf{oversight}) \leftarrow neq(web^{\mathcal{C}} \,|\!\sim\, ?\mathsf{Y},\ \mathbf{CF_{web1}}))$$

Then, John accepts the founding role $web^{\mathcal{C}}.\mathsf{founder}$ defined by himself:

$$John^{\mathcal{I}} \,|\!\sim\, grid^{\mathcal{C}}.\mathsf{director} \,|\!\sim\!(actAs(web^{\mathcal{C}}.\mathsf{founder},\ grid^{\mathcal{C}}.\mathsf{director}) \wedge \mathbf{CF_{web1}} \wedge \mathbf{CF_{web2}})$$

Finally, John specifies that role $web^{\mathcal{C}}.\mathsf{oversight}$ is the oversight role:

$$John^{\mathcal{I}} \,|\!\sim\, grid^{\mathcal{C}}.\mathsf{director} \,|\!\sim\, web^{\mathcal{C}}.\mathsf{founder} \,|\!\sim\, actAs(web^{\mathcal{C}}.\mathsf{oversight}, web^{\mathcal{C}}.\mathsf{founder})$$

Web Security Group Coalitions $sec^{\mathcal{C}}$ and $web^{\mathcal{C}}$ merge to a new coalition, which is called $websec^{\mathcal{C}}$. We have: $Alice^{\mathcal{I}} \rightarrow Bob^{\mathcal{I}}, Philip^{\mathcal{I}} : \mathbf{CF_{websec1}} \wedge \mathbf{CF_{websec2}}$;

$$\begin{aligned}\mathbf{CF_{websec1}} \;\hat{=}\; & websec^{\mathcal{C}} \,|\!\sim\!(actAs(websec^{\mathcal{C}}.\mathsf{constructor},\ Alice^{\mathcal{I}}) \wedge actAs(websec^{\mathcal{C}}.\mathsf{cmtte},\ [sec^{\mathcal{C}}, grid^{\mathcal{C}}]) \\ & \wedge (?\mathsf{X} \leftarrow \mathsf{threshold}(2,\ [websec^{\mathcal{C}}.\mathsf{cmtte}]) \,|\!\sim\, ?\mathsf{X}))\end{aligned}$$

$$\mathbf{CF_{websec2}} \;\hat{=}\; Alice^{\mathcal{I}} \,|\!\sim\!(Pay(500,\ USD,\ Alice^{\mathcal{I}},\ websec^{\mathcal{C}}.\mathsf{oversight}) \leftarrow neq(websec^{\mathcal{C}} \,|\!\sim\, ?\mathsf{Y},\ \mathbf{CF_{websec1}}))$$

Then, Bob and Philip accept the founding role $websec^{\mathcal{C}}.\mathsf{cmtte}$ defined by Alice:

$$\begin{aligned}Bob^{\mathcal{I}} \rightarrow Alice^{\mathcal{I}} \;&:\; Bob^{\mathcal{I}} \,|\!\sim\, sec^{\mathcal{C}}.\mathsf{wscmtte} \,|\!\sim\!(GFactAswebsec^{\mathcal{C}}.\mathsf{cmtte},\ sec^{\mathcal{C}} \wedge\ \mathbf{CF_{websec1}} \wedge\ \mathbf{CF_{websec2}}) \\ Philip^{\mathcal{I}} \rightarrow Alice^{\mathcal{I}} \;&:\; Philip^{\mathcal{I}} \,|\!\sim\, web^{\mathcal{C}}.\mathsf{wscmtte} \,|\!\sim\!(actAs(websec^{\mathcal{C}}.\mathsf{cmtte},\ web^{\mathcal{C}}) \wedge\ \mathbf{CF_{websec1}} \wedge\ \mathbf{CF_{websec2}})\end{aligned}$$

Finally, Bob (speaking for $sec^{\mathcal{C}}$) and Philip (speaking for $web^{\mathcal{C}}$) specify that role $websec^{\mathcal{C}}.\mathsf{oversight}$ is the oversight role.

$$\begin{aligned}Philip^{\mathcal{I}} \rightarrow Bob^{\mathcal{I}} \;&:\; Philip^{\mathcal{I}} \,|\!\sim\, web^{\mathcal{C}}.\mathsf{wscmtte} \,|\!\sim\, websec^{\mathcal{C}}.\mathsf{cmtte} \,|\!\sim\, actAs(websec^{\mathcal{C}}.\mathsf{oversight}, websec^{\mathcal{C}}.\mathsf{cmtte}) \\ Bob^{\mathcal{I}} \rightarrow Philip^{\mathcal{I}} \;&:\; Bob^{\mathcal{I}} \,|\!\sim\, sec^{\mathcal{C}}.\mathsf{wscmtte} \,|\!\sim\, websec^{\mathcal{C}}.\mathsf{cmtte} \,|\!\sim\, actAs(websec^{\mathcal{C}}.\mathsf{oversight}, websec^{\mathcal{C}}.\mathsf{cmtte})\end{aligned}$$