Reasoning about Joint Administration of Access Policies for Coalition Resources

Himanshu Khurana
University of Maryland
College Park, MD
hkhurana@eng.umd.edu

Virgil Gligor
University of Maryland
College Park, MD
gligor@eng.umd.edu

John Linn
RSA Laboratories
Bedford, MA
jlinn@rsasecurity.com

Abstract

We argue that joint administration of access policies for a dynamic coalition formed by autonomous domains requires that these domains set up a coalition authority that distributes attribute certificates authorizing access to policy objects (e.g., ACLs). Control over the issuance of such certificates is retained by member domains separately holding shares of the joint coalition authority's private key with which they sign the attribute certificates. Hence, any (proper) subset of the member domains need not be trusted to protect the private key. However, application servers that implement joint administration of access policies based on attribute certificates must trust all the signers of those certificates, namely all member domains of the coalition. To capture these trust relations we extend existing access control logics and show that the extensions are sound. To reason about joint administration of access policies, we illustrate an authorization protocol in our logic for accessing policy objects using threshold attribute certificates.

1. Introduction

Management of shared resources (e.g., objects, applications, and services) among autonomous domains that form a coalition or an alliance has recently become important in both military and commercial areas. Prior work in this area [11, 12, 22, 23, and 26] presents architectures and protocols for administering access control policies for coalition resources among autonomous domains using Public-Key Infrastructures (PKIs) that support attribute certificates [20]. This previous body of work focuses on resources owned by individual domains and shared with other member domains, which means that individual domains retain unilateral control over coalition resources they own.

In contrast, we address the problem of joint administration of access policies for coalition resources that are *jointly owned* by the member domains and therefore cannot be unilaterally controlled. Instances of this problem are found in both military coalitions [11] and commercial alliances. For example, a private genetics research company that has discovered the gene sequence for a particular disease wishes to form an alliance with a private hospital and a pharmaceutical company for research into finding a cure for the disease using the gene sequence. Given the financial commitments and expected eventual impact of finding such a cure, all three domains would like to jointly own and jointly administer access to all research data generated. To manage the resources comprising the research data, member domains establish a general-purpose web server. They agree that no single member domain should unilaterally administer access policies of the web server (e.g., set or modify ACLs, distribute membership certificates to access-authorized groups or roles) and that they all must reach consensus on any access policy decision. Given the high sensitivity of the research data, it is impractical to expect that member domains could out-source the administration of access policies jointly defined to any outside domain. Even if an outside domain that is equally trusted by all coalition members could be found, such a domain would represent a single point of *trust failure* and an attractive attack target. Therefore, the member domains, which jointly own the web server's resources, must jointly administer the access policies for these resources to satisfy the agreed-upon alliance requirements. The alliance is *dynamic*: domains representing other organizations may want to join this alliance after it is set up, and some of the initial member domains may leave the alliance before its mission ends. All alliance joins and leaves by autonomous domains may occur without prior arrangements.

In this paper, we argue that joint administration of access policies for coalition-owned resources formed by multiple autonomous domains requires that these domains set up a joint coalition authority that distributes attribute certificates authorizing access to policy objects (e.g., ACLs). Such an authority can be implemented in a self-contained manner, or its implementation can be distributed among participating domains. We also argue that shared public key systems, where one public key is owned by multiple principals each having a share of the corresponding private key, enable member domains to retain control over the issuance of these attribute certificates. That is, for joint administration of access policies, member domains separately hold shares of the

coalition authority's private key with which they sign the certificates. These distributed private key shares also enable effective protection of the private key from compromise by the coalition authority or by any proper subset of member domains (e.g., by coalition authority or domain penetration). Thus, a member domain need not trust any subset of other member domains for managing the private key that signs the attribute certificates. This is in contrast with conventional public-key systems (i.e., one public key owned by exactly one principal) where trust in the certificate signer would typically be required. However, application servers that implement joint administration of access policies based on attribute certificates must trust all the signers of those certificates, namely all member domains of the coalition. Hence, to reason about joint administration of access policies, we need an access control logic that captures these trust relations and an authorization protocol for accessing policy objects using attribute certificates. We present such a logic, which includes distributed private key shares, multi-principal jurisdiction, and selective distribution of access privileges with threshold attribute certificates. (A threshold attribute certificate has multiple principals (i.e., subjects) and a specified subset of these principals must agree in order to use or to delegate authority given by the certificate.) We show that our logic, which builds upon existing logics, [1, 18, 25], is sound.

The rest of this paper is organized as follows. In Section 2 we discuss the requirements of joint administration of access policies and show how these requirements can be satisfied. In Section 3 we discuss the use of shared public key techniques. In Section 4 we present an authorization protocol in our logic. In Section 5 we discuss related work and in Section 6 we conclude our work. In Appendices A-C we present the syntax and semantics of our logic and in Appendix D we show that our logic is sound. Due to space constraints, the appendices are provided in [17].

2. Joint Administration of Access Policies

In this section we specify requirements for joint administration of access policies for jointly owned dynamic coalition resources and show how these requirements can be satisfied. These resources may include applications that manage route communication systems, purchase orders for equipment, financial assets, and research data. In some coalitions, jointly owned resources may include auditing applications that are used to ensure that all domains are adhering to predefined access policies.

2.1. Requirements for Joint Administration of Access Policies

Requirement I: Joint coalition authority for administration of access policies. In general, the administration of resources uses identity certificate authorities for authentication purposes and attribute authorities for authorization purposes. Each autonomous domain will typically have its own identity certificate authority (*CA*) for distributing and revoking identity certificates to users registered in that domain. Since each domain will have its own policies for registering users and issuing identity certificates, it is impractical for the coalition to establish a coalition identity *CA* for registering all coalition users and issuing them identity certificates. Instead, all coalition application servers trust each domain's pre-established identity *CA* for distributing identity certificates to users of that domain. Similarly, for the administration of access policies for local domain resources, which are not essential for the continuity of coalition operations, each domain may have its own attribute authority that issues and revokes attribute certificates to users of that domain. In contrast, for the administration of access policies for coalition resources that are essential for coalition operations, and therefore are *jointly owned*, the coalition needs to establish a *joint coalition authority*. The jointly owned resources are deemed essential for coalition operations as any (proper) subset of these resources is insufficient for accomplishing the coalition mission. The joint coalition authority administers these resources in a manner that prevents any single domain from breaking up the coalition (or changing its mission) simply by unilaterally withdrawing from the coalition. This allows continuity of coalition operations even if a member domain leaves the coalition.

Requirement II: Coalition-closed administration of access policies. We require that coalition domains do not out-source the administration of jointly defined access policies to any domain outside the coalition. This is because given the sensitivity of jointly owned coalition resources (such as sensitive research data) it is impractical to expect the member domains to trust an outside domain for administering access to the resources. Furthermore, even if the member domains can find an outside domain that they trust equally, such a domain will become vulnerable to attacks and represent a single point of trust failure.

Requirement III: Consensus for administration of access policies. We require that no single domain should be able to unilaterally define and modify access policies of a jointly owned resource, namely the setting and updating of resource policy objects and the distribution and revocation of privileges for the resource, without consent of all other resource owner-domains. We assume that once member domains jointly define access policies they do not

compromise the coalition operations by refusing to co-operate in the administration of access policies, for example, by refusing to sign a joint access request.

2.2. Satisfying the Requirements

The first requirement discussed above, namely that for a joint coalition authority, may be satisfied by implementing a joint coalition authority that is separate from the member domains using replicated services and databases (Replication helps avoid single point of failures and slow response time.) Alternatively, the joint authority may be implemented in a fully distributed manner among the coalition members. If the joint coalition authority were implemented separately from the members' domains, then supporting coalition dynamics (domain departures and joins) would require modification of access policies only by the separate (replicated) administration services while in the case of a fully distributed authority, access policies at all member domains would have to be modified and some resources might have to be redistributed. Furthermore, in a fully distributed coalition authority, all member domains would have to distinguish between individually and jointly owned resources and administer them separately based on local domain or coalition-wide policies. This would complicate resource management in each member domain significantly.

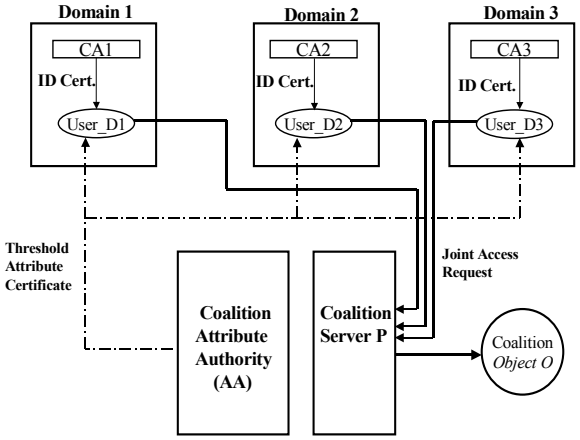

Figure 1: Joint Administration of Access Policies

The first two requirements discussed above can be satisfied in a straightforward manner as illustrated in Figure 1 above. Here Domains *D1*, *D2*, and *D3* establish a dynamic coalition to enable sharing of resources. Each domain has its own *CA* for distributing and revoking identity certificates to its users. For the joint administration of access policies for jointly owned resources such as *Object O* (managed by coalition server *P*), the domains establish a coalition authority called the *coalition Attribute Authority* (*AA*). The coalition *AA* operates effectively by distributing threshold attribute certificates to coalition users granting them privileges for setting and updating policy objects and for accessing resources. Threshold *k*-of-*n* attribute certificates or similar threshold structures [4, 6, 10, 19] distribute privileges to *n* principals in a manner that requires at least *k* of the *n* principals to sign an access request to be granted (such access requests are called *joint access requests*). For example, in a jointly owned database, it may be required that three specified coalition users, one from each coalition domain, must sign an access request to write to the database. Alternate mechanisms for distributing privileges such as attribute certificates issued to a group of users that own a shared public key can also be devised. Such alternate mechanisms do not impact the requirements of joint administration (and can be supported by our logic).

In Figure 1, the coalition *AA* distributes a threshold attribute certificate to coalition users *User_D1*, *User_D2* and *User_D3* granting them privileges for access to *Object O* (and to *Object O*'s ACL). The coalition users can consequently get access to *Object O* by sending joint access requests with the threshold attribute certificate to *Server P* (*Server P* trusts the coalition *AA* for distributing threshold attribute certificates). We note that both the coalition *AA* and *Server P* may be implemented separately from the member domains or fully distributed among the member domains.

To satisfy Requirement III, namely that of consensus for joint administration of access policies, the coalition *AA*'s private key must be protected such that no domain can exercise unilateral control over it. This is because access to *AA*'s private key will allow a domain to violate Requirement III by unilaterally modifying jointly defined access policies; e.g., by unilaterally distributing threshold attribute certificates for access to resources. To illustrate this with Figure 1, suppose that domain *D1* establishes the coalition *AA* and controls access to its private key. An administrator of Domain *D1* would program the coalition *AA* with trust relations that allow domains *D2* and *D3* to administer access policies at *AA* and with an access authorization protocol that would require signed requests from *all* three domains before any resource access policies can be defined or modified. Despite requiring signed requests from all three domains, the scenario would fail to satisfy Requirement III because domain *D1* could choose to unilaterally issue an attribute certificate to a user for access to *Object O*. This problem would only be compounded if the scenario is modified by distributing the coalition *AA*'s private key to all member domains; in that case, all domains would be able to unilaterally modify access policies.

We present two cases for management of *AA*'s public-private key pair that satisfy Requirement III albeit with different trust liabilities. The first case uses conventional public keys and results in a significant trust liability associated with the coalition *AA*. The second case uses

shared public keys and minimizes this trust liability.

Case I: A conventional public key for the coalition AA. In this case the coalition *AA* has a conventional public-private key pair. One way to manage this key pair for the scenario in Figure 1 that satisfies our requirement is as follows. Three administrators, one from each of the coalition domains *D1*, *D2*, and *D3* jointly create the coalition server *AA*. They program the server to generate its own public-private key pair and immediately store the private key in a hardware lock box so that it is inaccessible to the coalition domains (the public key is released to all administrators). The administrators then program the domains to trust the coalition *AA*'s private key for signatures on threshold attribute certificates. They also program *AA* with an authorization protocol that requires all owner-domains to sign a request for distribution and revocation of these threshold attribute certificates. Further, the coalition *AA* is programmed such that *AA*'s private key operations can be performed by cryptographic transaction sets implemented by the hardware lock box when a joint cryptographic request is made by all domain administrators (e.g., with a sequence of passwords).

This case satisfies our requirements as no domain can unilaterally modify resource access policies without consent of all three domains (as no domain has unilateral access to the coalition *AA*'s private key). However, this case has a significant trust liability associated with the protection of coalition *AA'*s private key even with a hardware lock box, such as the IBM 4758 [14], as a formal verification of the cryptographic transaction sets[1] implemented by such devices remains a significant research challenge [2]. On one hand, compromise of coalition *AA*'s private key by external penetrations would result in the *AA* being a single point of trust failure as this would lead to failure of access policy enforcement. (We also note that replication of the coalition *AA* for robustness reasons would only amplify this trust liability, as the private key would have to be replicated as well). On the other hand, compromise of coalition *AA*'s private key by an insider, such as a privileged administrator of a member domain who has access at the coalition *AA* for maintenance purposes, would enable the insider to violate the joint administration requirements in a repudiable manner.

Case II: A shared public key for the coalition AA. In this case, the coalition *AA* has a shared public key whose corresponding private key is split among the member domains. That is, in Figure 1, employing the distributed shared key generation algorithm of [8], domains *D1*, *D2*, and *D3* generate a public key for the coalition *AA* while retaining distributed shares of the corresponding private key such that no single domain has unilateral access to the coalition *AA*'s private key. All threshold attribute certificates distributed by the coalition *AA* must be signed by *all* domains using their distributed private key shares (such joint signature algorithms are discussed in Section 3). Similar to Case I above, this case also satisfies our requirements for joint administration as no domain can unilaterally modify resource access policies without consent of all other member domains. Furthermore, this case minimizes the trust liability of Case I as the private key is distributed among all coalition domains. That is, for external penetrations to succeed, all domains would have to be compromised to obtain the coalition *AA*'s private key; and for insider attacks to succeed, a domain would have to compromise all other member domains to get unilateral access to *AA*'s private key.

Hence, we use a shared public key for the coalition *AA* in Figure 1 to satisfy all requirements for joint administration of access policies.

3. Using Shared Public-Key Techniques

In this section we discuss the use of shared public key techniques, which have recently been used for intrusion-tolerant applications [27]. We give a brief overview of the shared public key generation algorithm of [8] and discuss the costs of using shared key techniques and the usefulness of the *m*-of-*n* private-key sharing scheme.

3.1. Shared RSA public key generation algorithm

Here we review some of the features of the shared RSA public-key generation algorithm of [8, 21]. The algorithm enables n domains to generate a modulus $N = pq$ and exponents e and d. At the end of the computation all domains are convinced that N is the product of two primes, however none of them know the factorization of N. The public exponent e is made public while d is shared among the domains in a way that enables m-out-of-n threshold signature generation. That is, m domains are able to issue a certificate without reconstructing the key d. This also implies that an attacker who penetrates at most m-1 domains is unable to obtain any information about the private key. From the point of view of collusion the algorithm is $\lfloor (n-1)/2 \rfloor$ private. That is, even if $\lfloor (n-1)/2 \rfloor$ domains share the information they learn during the protocol, they will still not be able to recover the factorization of N or the private key d.

Shared public key techniques have some technological costs associated with them, namely, deployment costs of a relatively new technology and operational costs of key generation time and joint signature application time. (An analysis of deployment costs is outside the scope of this paper). The operational costs are incurred on infrequent events of access policy specification and modification, such as certificate issuance and revocation, and hence deemed to be inconsequential relative to the frequency of

[1] For example, Anderson and Kuhn [3] and Bond [7] discuss protocol failures where the application of a sequence of cryptographic transactions leads to exposure of a clear key protected by a hardware security module.

subsequent accesses to be made using the certificate once issued. (Malkin *et al*. [21] show that it may take 1.5 mins to 5 mins on average to generate a shared public-key between three servers but it takes these servers only between 1.2 s to 2 s to apply a joint signature).

Though the idea of shared public-keys has also been discussed by others [9, 24], we use the algorithm of Boneh and Franklin [8] because it allows for the generation of the shared public key without a trusted server. The ability to generate the shared public key without a trusted server outside the coalition is of essence as discussed in Section 2.1 (Requirement II).

3.2. Joint Signatures with Distributed Private Key Shares

For joint administration of access policies, the public key K_{AA} of the coalition *Attribute Authority* (see Figure 1) is generated using the shared key generation algorithm resulting in private key shares that are distributed among all member domains (i.e., a *n*-of-*n* threshold sharing of the private key K_{AA}^{-1}). Once the public-key K_{AA} has been generated, all domains must apply a joint signature algorithm with their private key shares in order to sign any object with the private key K_{AA}^{-1}. The joint signature algorithm involves the *requestor* (one of the domains) sending a message to all the *co-signers* (the remaining member domains) with the message M to be signed and a key ID comprising the hash of N and the public exponent e. Each of the co-signers then apply their corresponding private key shares d_i to compute $S_i = M^{di} \bmod N$ and send the computations back to the requestor. The requestor then computes the message signature $S = \prod_{i=1}^{t} S_i \bmod N$. This joint signature protocol is illustrated in [27]. Using this joint signature algorithm, the domains sign threshold attribute certificates distributed by the coalition *AA*.

3.3. Threshold *m*-of-*n* sharing

Threshold *m*-of-*n* sharing [8] offers the advantage of increased domain server availability for joint signatures. Since only *m* out of the total *n* domains need to be on-line for application of joint signatures, threshold sharing increases domain availability as up to (*n*-*m*) domains can be down for maintenance or error recovery. This is probably not very useful in our example where n=3 but it may be useful in cases where n is larger. However, allowing a threshold *m*-of-*n* sharing requires a corresponding modification of the requirements of jointly owned coalition resources as the consent of *all* resource owner-domains is no longer necessary.

4. An Authorization Protocol and a Logic for its Specification

In this section we present an authorization protocol that uses shared public key techniques for joint administration of access policies. Verification of access requests that include threshold attribute certificates requires the application servers to trust signatures on the certificates. This implies that authorization protocols must capture trust relations between the servers and the domains that sign threshold attribute certificates with distributed private key shares. Current access control logics and trust management systems [1, 6, 10, 19] cannot capture these trust relations and hence cannot be used for joint administration of access policies with shared public keys. (They can, however, be used for administration of access policies for shared coalition resources with conventional public keys as discussed in [23]). We extend existing logics to capture the trust relations supported by shared public keys and develop an authorization protocol for accessing policy objects using threshold attribute certificates. We first present an example of joint administration of access policies and then present the authorization protocol in our logic.

4.1. An Example of Joint Administration of Access Policies

For simplicity we limit our discussion to object resources such as *Object O* in Figure 1 and the joint administration of *Object O*'s access control policies, namely, the setting and updating of policy objects of *Object O*, and the distribution and revocation of privileges for *Object O*. Other types of jointly owned resources such as applications or services and application-oriented policies such as privilege inheritance, time-constrained access, etc. are not discussed here (however, they will not pose any additional fundamental design problems).

In Figure 2 we present an example of joint administration of access policies for *Object O* based on the scenario illustrated in Figure 1. Here coalition domains *D1*, *D2* and *D3* establish the coalition *AA* and hold shares of *AA*'s private key. *User_D1* (U1), *User_D2* (U2) and *User_D3* (U3) are coalition users of domains *D1*, *D2* and *D3* respectively. Threshold Attribute Certificates (ACs) distributed by the coalition *AA* permit these coalition users to send access requests to *Server P* for access to *Object O*. For joint administration of access policies, these threshold attribute certificates are signed by all member domains using the joint signature algorithm (Section 3.2) with their private key shares. We show mediated access based on a jointly administered policy of privilege distribution that is represented by the joint signature on the threshold attribute certificate.

In this example, we support two operations on *Object O*, namely *write*, which includes the acts of creation and modification, and *read*. For a *write* operation a threshold of 2-out-of-3 signatures is required while for a *read* operation one signature suffices. When multiple principals send a joint access request, all principals making the request must sign the request before it is sent to the server.

The principal requesting the operation is called the *requestor* while the principal(s) attesting the request is called the *co-signer*(s). The requestor generates a request, obtains all necessary signatures from the co-signers and then sends the request to *Server P* as shown in Figure 2(b). We discuss verification of message freshness in Section 4.3. We note that a similar example for joint administration of *Object O*'s policy object (ACL) can easily be constructed.

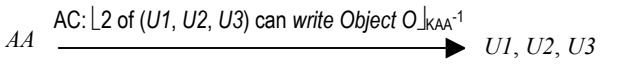

Figure 2(a): Threshold ACs for *write* operations

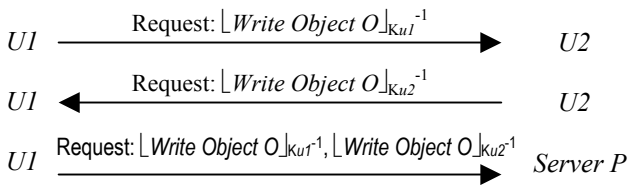

Figure 2(b): *Write* request approved by *Server P*

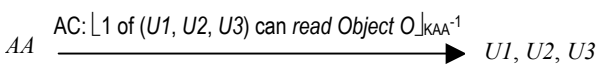

Figure 2(c): ACs for *read* operations

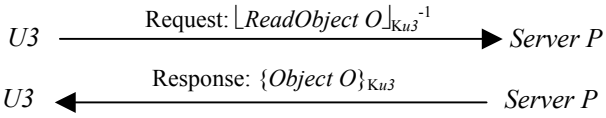

Figure 2(d): *Read* request approved by Server P

4.2. Introduction to the Logic

Here we introduce our logic primitives and concepts that are relevant to the authorization protocol presented in Section 4.3. The complete syntax and semantics of our logic is provided in Appendices A-C. Our logic builds on the authentication logics of [18, 25] and access control logics of [1]. We extend these logics to include (1) the notion of compound principals that own distributed private key shares of a public-key, (2) compound principal jurisdiction over formulae, (3) access control formulae and axioms that enable reasoning about distribution and revocation of attribute certificates, (4) selective distribution of access privileges to principals bound with public-keys, and (5) distribution and revocation of threshold attribute certificates.

We introduce the notion of a compound principal $CP = \{P_1, \ldots, P_n\}$, which is a set of n system principals $P_1, \ldots, P_n$ that collectively send and receive encrypted messages with n distributed private key shares of the public key K_{CP}. Alternatively, a compound principal CP can also be the set of principals (i.e., subjects) referenced in a threshold attribute certificate. Furthermore, $CP_{m,n}$ denotes a threshold m-of-n construct where either m principals are sufficient to sign on behalf of the n principals (via a threshold distribution of the private key shares) or m principals can send access requests against a threshold attribute certificate issued to $CP_{m,n}$.

In our logic, an encrypted message X is represented as $\lfloor X \rfloor_K^{-1}$ where K^{-1} is the private key that encrypts the message, which can then consequently be decrypted using the corresponding public-key K. We make statements such as Q *says*$_t$ $\lfloor X \rfloor_K^{-1}$, P *received*$_t$ $\lfloor X \rfloor_K^{-1}$, and P *believes*$_t$ X where P and Q are principals, and t is the time at which these statements were made. We say $\overset{K}{\Rightarrow}_t P$ to represent ownership of a public-key K by a principal P and we say S *controls*$_t$ $\overset{K}{\Rightarrow}_t P$ to represent that S is the principal (possibly a *CA*) that has jurisdiction over P's public-key. We provide an extension to previous logics to enable a public-key to represent a set of principals that own distributed shares of the corresponding private key. The generation of such shared keys was discussed in Section 3.1. We say $\overset{K}{\Rightarrow}_t CP$ where $CP = \{P_1, P_2, \ldots, P_n\}$ and P_1, P_2, …P_n are n principals. If the private key shares are distributed in a threshold manner we say that $\overset{K}{\Rightarrow}_t CP_{m,n}$ where m is an integer that is $\leq n$. Distributed private key shares allow us to make statements such as CP *says*$_t$ $\lfloor X \rfloor_K^{-1}$, CP *received*$_t$ $\lfloor X \rfloor_K^{-1}$, and CP *believes*$_t$ X. Furthermore, we allow multi-principal jurisdiction over formulae by making statements such as CP *controls*$_t$ $Q \Rightarrow_t G$, that is, all systems principals comprising the CP control Q's membership to group G.

For access control we allow system principals (and compound principals) to represent groups, which can be found on policy objects (e.g., ACLs). We make statements such as $P \Rightarrow_t G$, or $CP \Rightarrow_t G$ where G is a group to represent group membership. Since principals are more easily identified by names for the purpose of access control but must be bound to public-keys for appropriate selective distribution of privileges [16], we allow statements such as $P|K \Rightarrow_t G$ or $CP|K' \Rightarrow_t G$ where K is P's and K' is CP's public-key. Here "$P|K$" simply means that principal P is cryptographically bound to a public-key K in a verifiable identity certificate. This form of selective distribution of access privileges ensures that P must sign access requests with the private key K^{-1} in order to access privileges granted by this attribute certificate. This resolves any unauthorized access privilege retention problems that may occur in coalition scenarios as discussed in [16]. However, we do not support the *selective revocation* method for selective distribution of privileges discussed in [15, 16], as the method requires maintenance of dynamic links across the identity and attribute *CA*s. The maintenance of this state information is expensive and we avoid this cost. For distribution and revocation of threshold attribute certificates, we make

statements such as $CP_{m,n} \Rightarrow_t G$ where *m*-of-*n* principals in *CP* can send access requests for resources controlled by group *G*. For selective distribution of privileges in threshold attribute certificates, we allow *CP* to be comprised of system principals that are bound to specific public keys whose corresponding private keys must be used for signing access requests for these privileges; e.g., $CP = \{P_1|K_1, P_2|K_2, P_3|K_3, \ldots\}$.

Certificate Distribution and Revocation

Using formulae of the logic we represent public-key certificates by idealized time-stamped certificates. Consider a public-key identity certificate where *P* is a principal, K_P is *P*'s public signature verification key, t_b and t_e are begin and end times of the certificate validity period, *CA* is the issuing certificate authority with K_{CA}^{-1} being its private signature key and t_{CA} is the timestamp indicating the time when the certificate information was deemed accurate by the *CA*. Then an idealized time-stamped identity certificate issued by *CA* to *P* can be represented as,

Message $CA \rightarrow_{t1} P : \lfloor CA\ says_{t_{CA}} \overset{K_P}{\Rightarrow}_{[t_b,t_e]} P \rfloor_{K_{CA}}^{-1}$

Similarly, the revocation of *P*'s identity certificate will be represented as follows:

Message $CA \rightarrow_{t2} P : \lfloor CA\ says_{t_{CA}} \neg \overset{K_P}{\Rightarrow}_{t'} P \rfloor_{K_{CA}}^{-1}$ [2]

Attribute certificates define group membership. Consider a certificate authority *CA'* that controls membership to a group *G*, which can be found on certain ACLs. The following represents an idealized attribute certificate issued by *CA'* to *P* where $[t_b, t_e]$ denote the certificate's validity period.

Message $CA' \rightarrow_{t1} P : \lfloor CA'\ says_{t_{CA'}}\ P|K_P \Rightarrow_{[t_b,t_e]} G \rfloor_{K_{CA'}}^{-1}$ where K_P is *P*'s public-key as specified in the above identity certificate.

Similarly, the revocation of *P*'s attribute certificate will be represented as follows:

Message $CA' \rightarrow_{t2} P : \lfloor CA'\ says_{t_{CA'}} \neg P|K_P \Rightarrow_{t'} G \rfloor_{K_{CA'}}^{-1}$

Threshold attribute certificates are useful when privileges need to be distributed in a manner that requires multiple principals to send joint access requests. Consider a compound principal $CP = \{P_1|K_1, P_2|K_2, P_3|K_3\}$ where system principals P_1, P_2, and P_3 are bound to keys K_1, K_2, and K_3 respectively. *AA* (an Attribute Authority) is another compound principal that is established by three domains *D1*, *D2,* and *D3*. *AA*'s public key is K_{AA} while the corresponding private key K_{AA}^{-1} is distributed among the three domains that established *AA*. The following is a threshold attribute certificate for membership to group *G* issued to *CP* by *AA,* which requires that at least two of the three principals comprising *CP* must sign access requests in order to access resources available to group *G*. The compound principals comprising *AA* will sign this certificate using the joint signature algorithm discussed in Section 3.2. The threshold attribute certificate includes the set of principals comprising *CP* but for ease of reading we do not include it here.

Message $AA \rightarrow_{t1} CP : \lfloor AA\ says_{t_{AA}}\ CP_{2,3} \Rightarrow_{[t_b,t_e]} G \rfloor_{K_{AA}}^{-1}$

The revocation of *CP*'s threshold attribute certificate is represented as follows:

Message $AA \rightarrow_{t2} CP : \lfloor AA\ says_{t_{AA}} \neg\ CP_{2,3} \Rightarrow_{t'} G \rfloor_{K_{AA}}^{-1}$

4.3. The Authorization Protocol

Here we present the authorization protocol for joint administration of access policies. (Omitted details are provided in Appendix E). The protocol is developed in our logic and is based on the sound axioms of the logic. The authorization protocol is applied to *access requests* such as those illustrated in Figure 2 for the coalition scenario illustrated in Figure 1. We refer to Figure 1 and apply the authorization protocol on an access request using initial beliefs and our logic axioms to approve or deny the request. The protocol thus enables us to reason about the joint administration of access policies for coalition resources such as *Object O*.

Initial Beliefs. All beliefs held or deduced by *Server P* stem from the coalition *AA*'s shared public key K_{AA}. *Server P* trusts the coalition *AA* for distribution of threshold attribute certificates to all coalition users and trusts each domain's Identity *CA* for distribution of identity certificates to users of that domain. In Statement 1 below, *P* believes that *AA*'s public key K_{AA} is owned by all three domains *D1, D2* and *D3* ($CP = \{D1, D2, D3\}$) that have shares of the private key K_{AA}^{-1}. Though the *AA* only distributes this signed message, for ease of reading we say that *AA* signs messages with key K_{AA}^{-1} as well.

P believes that *AA* has jurisdiction over all group membership certificates for all groups (*G'*) at *AA* and that *AA* also has jurisdiction over the time that time-stamped certificates are believed accurate for all times after *t**. These beliefs are represented in our logic as follows:

1. $P\ believes_{t0}\ (\forall t \geq t^*)\ \overset{K_{AA}}{\Rightarrow}_{[t^*,\ t],\ P}\ CP_{3,3}$

2. $P\ believes_{t0}\ (\forall t)\ AA\ controls_t\ (\forall G', CP', t'_b, t'_e)\ CP' \Rightarrow_{[t'_b,t'_e],\ AA} G'$

3. $P\ believes_{t0}\ (\forall t \geq t^*)\ AA\ controls_{\ [t^*,\ t],\ P}\ (\forall G', CP', t'_b, t'_e, t'_{AA})\ AA\ says_{t'_{aa}}\ CP' \Rightarrow_{[t'_b,t'_e],\ AA} G'$

P believes that *CA1*, *CA2*, and *CA3* of domains *D1*, *D2,* and *D3*, respectively, have jurisdiction over the public-key identity certificates for users in their domains and that the *CA*s also have jurisdiction over the time that time-stamped certificates are believed accurate for all times after *t**. Furthermore, *P* believes that keys K_{CA1}, K_{CA2}, and K_{CA3} are the public-keys of *CA1*, *CA2*, and *CA3* respectively.

For access control of jointly owned resource *Object O*, *AA* establishes two groups *G_write* and *G_read*.

[2] In our analysis all revocation certificates have an upper bound of infinity.

Membership to group *G_write* allows a principal (or a compound principal) write privileges for *Object O* and membership to group *G_read* grants read privileges for *Object O*. These groups appear on *Object O*'s ACL managed at *Server P*. The ACL is a simple disjunction of expressions associated with *Object O*. That is, ACL_O: $\{E_0, E_1, \ldots, E_n\}$ where each expressions E_i = (G, access permissions) for a group G. Setting and updating policy objects is handled in a manner similar to that of accessing objects. That is, threshold attribute certificates are distributed that grant certain coalition users the authority to modify policy objects.

Access Request Verification. We now illustrate our authorization protocol by verifying an *access request* that comprises a signed request, identity certificates and a threshold attribute certificate. The access request is for a write operation on *Object O* and is sent by *User_D1* as illustrated in Figure 2 (b). In the idealized threshold attribute certificate below (issued to users *User_D1*, *User_D2*, and *User_D3* by *AA* granting them group membership in a 2-of-3 threshold manner to group *G_write*) *CP'* is a compound principal and *CP'* = $\{User\_D1|K_{user\_D1}, User\_D2|K_{user\_D2}, User\_D3|K_{user\_D3}\}$. The access request is as follows:

Message 1-1 (Identity certificate)

$User\_D1 \rightarrow_{t1} P: \lfloor CA1\ says_{tCA1} \overset{K_{user\_D1}}{\Rightarrow}_{[tb,te]} User\_D1 \rfloor_{K_{CA1}}^{-1}$

Message 1-2 (Identity certificate)

$User\_D1 \rightarrow_{t1} P: \lfloor CA2\ says_{tCA2} \overset{K_{user\_D2}}{\Rightarrow}_{[tb,te]} User\_D2 \rfloor_{K_{CA2}}^{-1}$

Message 1-3 (Threshold Attribute Certificate)

$User\_D1 \rightarrow_{t1} P: \lfloor AA\ says_{taa}\ CP'_{2,3} \Rightarrow_{[tb',te']} G\_write \rfloor_{K_{AA}}^{-1}$

Message 1-4 (Signed Request)

$User\_D1 \rightarrow_{t1} P: \{ \lfloor User\_D1\ says_{tu1}$ "write" $O \rfloor_{Kuser\_D1}^{-1}$,
$\lfloor User\_D2\ says_{tu2}$ "write" $O \rfloor_{Kuser\_D2}^{-1} \}$

We note that principal P can easily verify the freshness of these time-stamped messages using the freshness axiom of our logic (axiom A21) in a manner similar to that illustrated by [25]. The following access authorization protocol will be applied to these messages:

Step 1. *Verify the signing keys* K_{user_D1} and K_{user_D2}. We refer to techniques developed in [25] to authenticate users. We apply the originator identification axiom A10, the jurisdiction axiom A22, and the initial beliefs on message 1-1 to obtain,

4. $P\ believes_{t2}\ (\overset{K_{User\_D1}}{\Rightarrow}_{[tb,te],CA1}\ User\_D1)$

Similarly, for *User_D2* we deduce from message 1-2 that,

5. $P\ believes_{t3}\ (\overset{K_{User\_D2}}{\Rightarrow}_{[tb,te],CA2}\ User\_D2)$

Step 2. *Establish group membership*.

We apply the originator identification axiom A10 to message 1-3 and deduce that,

6. $P\ believes_{t4}$

$AA\ said_{t2,P} \lfloor AA\ says_{taa}\ CP'_{2,3} \Rightarrow_{[tb',te'],AA} G\_write \rfloor_{K_{AA}}^{-1}$

From Statement 3 we can deduce that,

7. $P\ believes_{t4}\ AA\ controls_{[t*,t],P}$

$AA\ says_{taa}\ CP'_{2,3} \Rightarrow_{[tb',te'],AA} G\_write$, where [$t*$, t] denotes the particular time interval in question.

We apply the jurisdiction axiom A23 on 7.

8. $P\ believes_{t4}$

($AA\ says_{taa}\ CP'_{2,3} \Rightarrow_{[tb',te'],AA} G\_write\ at_P \langle t*, t_2 \rangle$)

To remove $at_P \langle t*, t_2 \rangle$, we apply the reduction axiom A9,

9. $P\ believes_{t4}$ ($AA\ says_{taa}\ CP'_{2,3} \Rightarrow_{[tb',te'],AA} G\_write$)

We apply the access control group membership axiom A25 on 9 and 2 to obtain,

10. $P\ believes_{t4}\ CP'_{2,3} \Rightarrow_{[tb',te'],AA} G\_write$

Step 3. *Verify signed Request*. By applying the originator identification axiom and jurisdiction axiom on message 1-4 we get,

11. $P\ believes_{t5}$

$User\_D1\ says_{tu1} \lfloor User\_D1\ says_{tu1}$ "write" $O \rfloor K_{user\_D1}^{-1}$

12. $P\ believes_{t5}$

$User\_D2\ says_{tu2} \lfloor User\_D2\ says_{tu2}$ "write" $O \rfloor K_{user\_D2}^{-1}$

Applying access control axiom A38 on 10, 11 and 12,

13. $P\ believes_{t6}$ ($G\_write\ says_{t6}$ "write" O)

Step 4. *Verify ACL*. If $t_{b'} \leq t_1$, $t_6 \leq t_{e'}$ and (G_write, "write" O) $\in ACL_O$, access is approved.

Reasoning about revocation. We use our logic to illustrate how one can reason about revocation of the threshold attribute certificate in the example discussed above. (Revocation of identity certificates is discussed in [25]). Let *RA* be a revocation authority that is authorized to provide revocation information on behalf of *AA*. At time t_7 *P* receives the following revocation message from *RA*:

Message 2

$RA \rightarrow_{t7} P: \lfloor RA\ says_{tRA} \neg\ CP'_{2,3} \Rightarrow_{t',RA} G\_write \rfloor_{K_{RA}}^{-1}$

Applying steps 7-10 above on message 2, we can deduce the following at time t_8

14. $P\ believes_{t8}$ ($\neg CP'_{2,3} \Rightarrow_{t',RA} G\_write\ at_P\ t*$)

where $t* \geq t_P$. If we interpret message 1 subject to a "believe until revoked" condition regarding the goodness of $CP'_{2,3}$'s threshold attribute certificate, then, we will be unable to obtain this belief for $t_4 \geq t_8$.

5. Related Work

In this section we discuss related work in the areas of access control in dynamic coalitions and trust management systems.

Access control in dynamic coalitions. Winsboro *et al*. [26] and Seamons *et al*. [22] have developed a trust negotiation protocol whereby clients that were previously unknown to servers can get access to server resources after presenting credentials (certificates) that match those required by server access policies. This work presents a credential negotiation process that allows the client to present its credentials gradually based on the incremental disclosure of the server's access policies. Herzberg *et al*. [12] have developed a protocol that allows a foreign user

to be placed in a local role if that user can present necessary credentials (attribute certificates). This work defines a Trust Policy Language and a Trust Establishment Engine to manage foreign user mappings to local roles. Unlike our work, this interesting body of research does not address the possibility that multiple autonomous domains (the clients and servers in the models above) may wish to collaborate in a peer-to-peer fashion and jointly own resources.

Shands *et al*. [23] have developed a Secure Virtual Enclave environment where domains with Role Based Access Control instantiations can share resources. Although, this work addresses important concerns in dynamic coalitions, it assumes that all resources are unilaterally owned and administered by single coalition domains. Gibson [11] has studied the need for forming coalitions in military environments and has proposed an architecture for enabling the formation of such coalitions. His work motivates the need for jointly owned resources in a military coalition but does not aim at providing a specific solution to the problem of joint administration of those resources.

Trust Management Systems and Access Control Logics. Lampson *et al.* [18] and Abadi *et al*. [1] present an access control logic that enables reasoning about distribution of identity and attribute certificates. This logic can express typical authorization protocols for access control using public-key certificates though they do not address the revocation of attribute certificates assuming that attribute certificates will be short-lived. We extend their work to include time-based distribution and revocation of both identity and attribute certificates, distributed private key shares, multi-principal jurisdiction, and threshold attribute certificates. Further, we enable selective distribution of privileges in an attribute certificate to a public-key in addition to a name. The logic of Stubblebine and Wright [25] reasons about time-based distribution and revocation of identity certificates. We refer to their authentication logic and extend it to include access control formulae and axioms, distributed private key shares, and multi-principal jurisdiction.

SPKI [10] and Aura [4] discuss the notion of *authorization* certificates where privileges are directly issued to public-keys. They also allow distribution of privileges with threshold certificates. Our logic supports threshold attribute certificates and similar selective distribution to principals bound to specific public-keys. Furthermore, we support distributed private key shares and multi-principal jurisdiction, which are not addressed by SPKI.

PolicyMaker [5] and Keynote [6] are trust-management systems that grant privileges to public-keys and include threshold distribution of privileges. They can be used to specify access control policies more flexibly than is possible in our logic. However, in contrast to our logic, certificate distribution and revocation are considered external to trust management limiting the ability of compliance checkers to reason about privilege revocation. They also do not support distributed private key shares. DL [19] is a tractable trust management system for authorization in large-scale, open, distributed systems. DL supports multi-principal jurisdiction and can be used to express access control policies more flexibly than by using our logic. DL also supports reasoning about certificate revocation. However, DL does not support distributed private key shares but it may be possible to extend the logic to do so along the lines of our logic.

6. Conclusions and Future Work

We argued that joint administration of access policies for a dynamic coalition formed by autonomous domains requires that these domains set up a coalition authority that distributes attribute certificates yet retain control over the issuance of such certificates. Shared public keys have been applied to building intrusion tolerant applications and we have shown that they enable joint administration of access policies in a way that minimizes trust liabilities when compared to solutions with conventional public keys. Furthermore, we have developed an authorization protocol to reason about the joint administration of access policies in our logic. Our logic extends previous logics to include distributed private key shares and multi-principal jurisdiction over formulae and we have shown that our logic is sound.

Several problems of coalition resource management related to our work remain to be solved. Coalitions can be dynamic in that member domains may leave and new ones may join. In our scenario this would require re-keying the *Attribute Authority* whenever coalition dynamics occur. Wu *et al*. [27] describe a *refresh* operation that allows re-distribution of private key shares of an existing shared public key among the coalition domains. However this does not easily extend to coalition dynamics where member domains can join or leave. Therefore, in our scenario, coalition dynamics would require establishing a new, shared public-key and consequently would require large-scale revocation and re-distribution of certificates. Further work is required to find a reasonable cost for coalition dynamics. A potential related problem is that of possible collusion among domains. The shared key generation algorithm is $\lfloor (n+1)/2 \rfloor$ private, which means that if $\lfloor (n+1)/2 \rfloor$ domains collude, they can determine the private key. Whether this problem materializes in practice or whether this type of collision effectively implies that a new separate coalition must be formed, remains to be determined by the rules of coalition set up and tear down.

Acknowledgements

We would like to thank Radostina Koleva, Laurent Eschenauer, Omer Horvitz, and Vijay Bharadwaj for their insightful comments on earlier drafts of this paper. The first two authors' work is supported by the Defense Advanced Research Projects Agency and managed by the U.S. Air Force Research Laboratory under contract F30602-00-2-0510. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, U.S. Air Force Research Laboratory, or the United States Government.

References

1. M. Abadi, M. Burrows, B. Lampson, and G.D. Plotkin, "A calculus for access control in distributed systems", *ACM Transactions on Programming Languages and Systems*, vol. 15, no. 4, September, 1993, pp. 706-734.
2. R. Anderson, "The Correctness of Crypto Transaction Sets", 8th International Workshop on Security Protocols, LNCS, Vol. 2133, Springer-Verlag, 2000, pp 125-141.
3. R. Anderson and M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices", 5th International Workshop on Security Protocols, LNCS, Vol. 1361, Springer-Verlag, 1997, pp 125-136.
4. T. Aura, "On the Structure of Delegation Networks", Proceedings of the 11th IEEE Computer Security Foundations Workshop, Rockport, MA, June 1998.
5. M. Blaze, J. Feigenbaum, and J. Lacy, " Decentralized Trust Management", Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA, May 1996.
6. M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, "The Keynote Trust Management System", Version 2., RFC-2704, IETF, September 1999.
7. M. Bond, "Attacks on Cryptoprocessor Transaction Sets", Proceedings of the CHES 2001 Workshop, Paris 2001, Springer-Verlag LNCS 2162, pp 220-234.
8. D. Boneh and M. Franklin, "Efficient Generation of Shared RSA Keys", *Advances in Cryptology - Crypto' 97, Lecture Notes in Computer Science*, Vol. 1233, Springer-Verlag, 1997, pp. 425--439.
9. Y.Desmedt and Y. Frankel, "Shared Generation of Authenticators and Signatures", *Advances in Cryptology – Crypto '91*, Springer-Verlag LNCS 576, 457-469, 1992.
10. C.M.Ellison, "SPKI Certificate documentation" (See http://world.std.com/~cme/html/spki.html), 1998.
11. T. Gibson, "An Architecture for Flexible, High Assurance, Multi-Security Domain Networks", Proceedings of the Network and Distributed Systems Security Symposium, San Diego, February 2001.
12. A. Herzberg, Y. Mass, J. Michaeli, D. Naor and Y. Ravid, "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers", Proceeding of the IEEE Symposium on Security and Privacy, Oakland, California, May 2000.
13. R.Housley, W.Ford, W.Polk, D.Solo, "Internet X.509 Public Key Infrastructure: Certificate and CRL Profile", Work in Progress - PKIX Working Group, Internet Draft, July 2001.
14. IBM, '*IBM 4758 PCI Cryptographic Coprocessor – CCA Basic Services Reference and Guide'*, Release 1.31 for the IBM 4758-001, available through http://www.ibm.com/security/cryptocards/
15. H.Khurana and V.D. Gligor, "Review and Revocation of Access Privileges Distributed with PKI Certificates", 8th International Workshop on Security Protocols, LNCS, Vol. 2133, Springer-Verlag, 2000, pp 100-124.
16. H.Khurana and V.D. Gligor, "Enforcing the Dependency Between PKI Certificate in *ad-hoc* Networks", Proceedings of the IEEE International Conference on Telecommunications, Bucharest, Romania, June 2001.
17. H.Khurana, V.D. Gligor, and J. Linn, "Reasoning About Joint Administration of Access Policies for Coalition Resources", (full length manuscript), March 15th, 2002, available at *http://glue.umd.edu/~gligor*.
18. B.Lampson, M.Abadi, M.Burrows and E.Wobber, "Authentication in distributed systems: Theory and Practice", *ACM Transactions on Computer Systems*, vol. 10, no. 4, November 1992, pp. 265-310.
19. N. Li, B. Grosof, and J. Feigenbaum, "A Practical Implementable and Tractable Delegation Logic", Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, May 2000.
20. J. Linn and M. Nystrom, "Attribute certification: an enabling technology for delegation and role-based controls in distributed environments", Proceedings of the Fourth ACM Workshop on Role-Based Access Control, 1999, p. 160.
21. M. Malkin, T. Wu and D. Boneh, "Experimenting with Shared Generation of RSA keys", Proceedings of the Internet Society's Symposium on Network and Distributed System Security, Feb. 1999, pp. 43—56.
22. K. E. Seamons, M. Winslett, and T. Yu, "Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation", Proceedings of the Internet Society's Symposium on Network and Distributed System Security, San Diego, CA, February 2001.
23. D. Shands, R. Yee, J. Jacobs, "Secure Virtual Enclaves: Supporting Coalition Use of Distributed Application Technologies", Proceedings of the Network and Distributed Systems Security Symposium, San Diego, February 2000.
24. V. Shoup, "Practical Threshold Signatures", *Advances in Cryptology – Eurocrypt 2000*, Springer-Verlag LNCS 1807, pp.207-220, 2000.
25. S. Stubblebine and R. Wright, "An Authentication Logic Supporting Synchronization, Revocation, and Recency", Third ACM Conference on Computer and Communications Security, New Delhi, India, March, 1996, pp. 95-105.
26. W. H. Winsborough, K. E. Seamons, and V. E. Jones, "Automated Trust Negotiation", DARPA Information Survivability Conference and Exposition, Hilton Head, January 2000.
27. T. Wu, M. Malkin, and D. Boneh, "Building Intrusion Tolerant Applications", Proceedings of the 8th USENIX Security Symposium, 1999, pp. 79--91.

Appendix A. Messages and Formulas

In Appendices A – C we present the syntax and semantics of our logic. In Appendix D we present a proof of the soundness of our logic. Our logic extends the authentication logics of [1, 18, 25] to include (1) the notion of compound principals that own distributed private keys shares of a public-key, (2) compound principal jurisdiction over formulae, (3) access control formulae and axioms that enable reasoning about distribution and revocation of attribute certificates, (4) selective distribution of access privileges to principals bound with public-keys, and (5) distribution and revocation of threshold attribute certificates. These extensions are reflected in Axioms 10, 24 – 38.

We assume that each principal has a local concept of time, and that different principals' times may not agree. However, we assume that the times of all the principals comprising a compound principal are synchronized. We use $[t_1, t_2]$ to indicate that a formula holds at all times between t_1 and t_2, and we use $\langle t_1, t_2 \rangle$ to indicate that a formula holds at some time between t_1 and t_2.

Γ is the set of primitive terms. We assume Γ contains real numbers and constant symbols called *principals*, *public keys*, *times*, *data constants* and *primitive propositions*. We assume that the set of times is ordered. Each key has an inverse. If K is a public key then K^{-1} is a private key. A digital signature on a message X is represented as $\lfloor X \rfloor_K^{-1}$.

Given the set of terms Γ, we define the set of messages, M_Γ, to be the smallest set satisfying the following conditions:

M1. φ is a message if φ is a formula.

M2. X is a message if $X \in \Gamma$.

M3. $F(X_1, \ldots, X_n)$ is a message if $X_1, \ldots, X_n$ are messages and F is any n-ary function in F_Γ and is defined below.

In particular, it follows from M3 that $\lfloor X \rfloor_K^{-1}$ is a message if X is a message and K is a public key. A message is usually sent from one principal to another at a particular time on the sender's clock. This transmission of a signed message X from principal A to B at time t is represented as, $A \rightarrow_t B: \lfloor X \rfloor_K^{-1}$.

F_Γ is the set of functions on primitive terms. Some messages have truth-values while others, such as times, principal names and nonces, do not. We identify a subset F_Γ of M_Γ that is the set of *formulas*. All formulas have truth-values. Messages and formulas in the logic are defined by mutual induction.

The set of formulas, F_Γ, is the smallest set satisfying the following conditions.

F1. p is a formula if p is a primitive proposition.

F2. $\neg\varphi$ and $\varphi \wedge \psi$ are formulas if φ and ψ are formulas.

F3. $t_1 \leq t_2$ and $t_1 \geq t_2$ are formulas if t_1 and t_2 are times or real numbers.

F4. If P is a principal and φ is a formula, then

a. P *believes*$_t$ φ and P *controls*$_t$ φ are formulas if t is a time.

b. P *belives*$_{[t_1, t_2]}$ φ and P *controls*$_{[t_1, t_2]}$ φ are formulas if $t_1 \leq t_2$ are times.

c. P *believes*$_{\langle t_1, t_2 \rangle}$ φ and P *controls*$_{\langle t_1, t_2 \rangle}$ φ are formulas if $t_1 \leq t_2$ are times.

F5. If $P_1, P_2, \ldots P_n$ are n principals, and $CP = \{P_1, P_2, \ldots P_n\}$ is a compound principal then

a. CP *believes*$_t$ φ and CP *controls*$_t$ φ are formulas if t is a time.

b. CP *belives*$_{[t_1, t_2]}$ φ and CP *controls*$_{[t_1, t_2]}$ φ are formulas if $t_1 \leq t_2$ are times.

c. CP *believes*$_{\langle t_1, t_2 \rangle}$ φ and CP *controls*$_{\langle t_1, t_2 \rangle}$ φ are formulas if $t_1 \leq t_2$ are times.

F6. If P is a principal and X is a message, then

a. P *received*$_t$ X, P *said*$_t$ X, and P *says*$_t$ X are formulas if t is a time

b. P *received*$_{[t_1, t_2]}$ X, P *said*$_{[t_1, t_2]}$ X, and P *says*$_{[t_1, t_2]}$ X are formulas if $t_1 \leq t_2$ are times.

c. P *received*$_{\langle t_1, t_2 \rangle}$ X, P *said*$_{\langle t_1, t_2 \rangle}$ X, and P *says*$_{\langle t_1, t_2 \rangle}$ X are formulas if $t_1 \leq t_2$ are times.

F7. If $P_1, P_2, \ldots P_n$ are n principals, $CP = \{P_1, P_2, \ldots P_n\}$ is a compound principal and X is a message, then

a. CP *received*$_t$ X, CP *said*$_t$ X, and CP *says*$_t$ X are formulas if t is a time

b. CP *received*$_{[t_1, t_2]}$ X, CP *said*$_{[t_1, t_2]}$ X, and CP *says*$_{[t_1, t_2]}$ X are formulas if $t_1 \leq t_2$ are times.

c. CP *received*$_{\langle t_1, t_2 \rangle}$ X, CP *said*$_{\langle t_1, t_2 \rangle}$ X, and CP *says*$_{\langle t_1, t_2 \rangle}$ X are formulas if $t_1 \leq t_2$ are times

F8. If P is a principal and K is a public key, then

$\overset{K}{\Rightarrow}_t P$, $\overset{K}{\Rightarrow}_{[t_1, t_2]} P$ and $\overset{K}{\Rightarrow}_{\langle t_1, t_2 \rangle} P$ are formulas if t is a time and $t_1 \leq t_2$ are times.

F9. If $P_1, P_2, \ldots P_n$ are n principals, $CP = \{P_1, P_2, \ldots P_n\}$ is a compound principal, and K is public key, then

$\overset{K}{\Rightarrow}_t CP$, $\overset{K}{\Rightarrow}_{[t_1, t_2]} CP$ and $\overset{K}{\Rightarrow}_{\langle t_1, t_2 \rangle} CP$ are formulas if t is a time and $t_1 \leq t_2$ are times.

F10. If $P_1, P_2, \ldots P_n$ are n principals, $CP = \{P_1, P_2, \ldots P_n\}$ is a compound principal, K is public key, and m is an integer such that $m \le n$, then

$\overset{K}{\Rightarrow}_t CP_{m,n}$, $\overset{K}{\Rightarrow}_{[t_1, t_2]} CP_{m,n}$ and $\overset{K}{\Rightarrow}_{\langle t_1, t_2\rangle} CP_{m,n}$ are formulas if t is a time and $t_1 \le t_2$ are times.

F11. If P is a principal, CP is a compound principal and K is a public key, then

a. $P\ has_t\ K$ and $CP\ has_t\ K$ are formulas if t is a time
b. $P\ has_{[t_1, t_2]}\ K$ and $CP\ has_{[t_1, t_2]}\ K$ are formulas if $t_1 \le t_2$ are times
c. $P\ has_{\langle t_1, t_2\rangle}\ K$ and $CP\ has_{\langle t_1, t_2\rangle}\ K$ are formulas if $t_1 \le t_2$ are times

F12. If P is a principal and G is a group, then

$P \Rightarrow_t G$, $P \Rightarrow_{[t_1, t_2]} G$, and $P \Rightarrow_{\langle t_1, t_2\rangle} G$ are formulas if t is a time and $t_1 \le t_2$ are times.

F13. If P is a principal, K is a public-key, and G is a group, then

$P|K \Rightarrow_t G$, $P|K \Rightarrow_{[t_1, t_2]} G$, and $P|K \Rightarrow_{\langle t_1, t_2\rangle} G$ are formulas if t is a time and $t_1 \le t_2$ are times. Here "|" is a concatenation symbol and the formulas imply that principal P, which is bound to public-key K, *speaks for* group G.

F14. If $P_1, P_2, \ldots P_n$ are n principals, $CP = \{P_1, P_2, \ldots P_n\}$ is a compound principal, and G is a group, then

$CP \Rightarrow_t G$, $CP \Rightarrow_{[t_1, t_2]} G$, and $CP \Rightarrow_{\langle t_1, t_2\rangle} G$ are formulas if t is a time and $t_1 \le t_2$ are times.

F15. If $P_1, P_2, \ldots P_n$ are n principals, $K_1, K_2, \ldots K_n$ are public-keys, $CP = \{P_1|K_1, P_2|K_2, \ldots P_n|K_n\}$ is a compound principal, m is an integer such that $m \le n$ and G is a group, then

$CP_{m,n} \Rightarrow_t G$, $CP_{m,n} \Rightarrow_{[t_1, t_2]} G$, and $CP_{m,n} \Rightarrow_{\langle t_1, t_2\rangle} G$ are formulas if t is a time and $t_1 \le t_2$ are times

F16. If $P_1, P_2, \ldots P_n$ are n principals, $CP = \{P_1, P_2, \ldots P_n\}$ is a compound principal, K is a public-key, and G is a group, then

$CP|K \Rightarrow_t G$, $CP|K \Rightarrow_{[t_1, t_2]} G$, and $CP|K \Rightarrow_{\langle t_1, t_2\rangle} G$ are formulas if t is a time and $t_1 \le t_2$ are times.

F17. If X is a message and P is a principal, then

a. $fresh_{t,P}\ X$ is a formula if t is a time.
b. $fresh_{[t_1, t_2],P}\ X$ is a formula if $t_1 \le t_2$ are times.
c. $fresh_{\langle t_1, t_2\rangle,P}\ X$ is a formula if $t_1 \le t_2$ are times.

F18. If X is a message and CP is a compound principal, then

d. $fresh_{t,CP}\ X$ is a formula if t is a time.
e. $fresh_{[t_1, t_2],CP}\ X$ is a formula if $t_1 \le t_2$ are times.
f. $fresh_{\langle t_1, t_2\rangle,CP}\ X$ is a formula if $t_1 \le t_2$ are times.

F19. If P is a principal, t is a time, and φ is a formula, then $\varphi\ at_P\ t$ is a formula. This formula denotes presence of φ at the principal P at time t on P's clock.

F20. If CP is a principal, t is a time, and φ is a formula, then $\varphi\ at_{CP}\ t$ is a formula. This formula denotes presence of φ at the compound principal CP at time t on the clock of any principal $P_i \in CP$. We assume that the clocks of all principals comprising a compound principal are synchronized.

F21. If φ is a formula and t is a time, then

a. $(\exists t : t_1 \le t \le t_2)\ \varphi$ is a formula if t_1 and t_2 are times.
b. $(\exists t \ge t')\ \varphi$ and $(\exists t \le t')\ \varphi$ are formulas if t' is a time.

F22. If φ is a formula and t is a time, then

a. $(\forall t : t_1 \le t \le t_2)\ \varphi$ is a formula if t_1 and t_2 are times.
b. $(\forall t \ge t')\ \varphi$ and $(\forall t \le t')\ \varphi$ are formulas if t' is a time.

In addition, any time t that appears in a formula can be replaced by t, P to yield another formula, which denotes the principal at whose clock t is measured. Similarly $[t_1, t_2]$ can be replaced by $[t_1, t_2], P$ and $\langle t_1, t_2\rangle$ can be replaced by $\langle t_1, t_2\rangle, P$.

Appendix B. Axioms

Our axiom system includes two inference rules. For any formulas φ and ψ, and time t,

R1. Modus Ponens: From φ and $\varphi \supset \psi$ infer ψ.

R2. Necessitation: If $\vdash \varphi$, then from φ infer $P\ believes_t\ \varphi$

$\vdash \varphi$ means that φ is a *theorem,* i.e., it is derivable from the axioms alone. Since the symbol $\vdash$ is a metalinguistic symbol, it does not actually appear in any proofs. We write

$\Gamma \vdash \varphi$ if from the set Γ of formulas, it is possible to derive φ.

We take all tautologies of propositional logic as axioms. In addition, we take as axioms all instances of the following axiom schemas.

Belief

A1. $P\ believes_t\ \varphi \wedge P\ believes_t\ (\varphi \supset \psi) \supset P\ believes_t\ \psi$
A2. $P\ believes_t\ \varphi \equiv P\ believes_t\ P\ believes_t\ \varphi$
A3. $P\ believes_t\ \varphi \equiv P\ believes_t\ (\varphi\ at_P\ t)$
A4. $CP\ believes_t\ \varphi \wedge CP\ believes_t\ (\varphi \supset \psi) \supset CP\ believes_t\ \psi$
A5. $CP\ believes_t\ \varphi \equiv CP\ believes_t\ CP\ believes_t\ \varphi$
A6. $CP\ believes_t\ \varphi \equiv CP\ believes_t\ (\varphi\ at_{CP}\ t)$
Multiple levels of belief are equivalent to a single level. Since principals can be mistaken,
$P\ believes_t\ \varphi \supset \varphi\ at_P\ t$ and $CP\ belives_t\ \varphi \supset \varphi\ at_{CP}\ t$ does not hold.

Time and Reduction

A7. Time interval axioms

a. $P\ belives_{[t1,\ t2]}\ \varphi \equiv (\forall t\colon t_1 \le t \le t_2)\ P\ believes_t\ \varphi$
b. $P\ belives_{\langle t1,\ t2\rangle}\ \varphi \equiv (\forall t\colon t_1 \le t \le t_2)\ P\ believes_t\ \varphi$
c. $CP\ belives_{[t1,\ t2]}\ \varphi \equiv (\forall t\colon t_1 \le t \le t_2)\ CP\ believes_t\ \varphi$
d. $CP\ belives_{\langle t1,\ t2\rangle}\ \varphi \equiv (\forall t\colon t_1 \le t \le t_2)\ CP\ believes_t\ \varphi$

We also include analogous axioms for *controls*, *received*, *says*, *said*, *has*, and $\Rightarrow$.
A8. Monotonicity axioms

a. $P\ received_t\ X \wedge t' \ge t \supset P\ received_{t'}\ X$
b. $P\ said_t\ X \wedge t' \ge t \supset P\ said_{t'}\ X$
c. $P\ has_t\ K \wedge t' \ge t \supset P\ has_{t'}\ K$
d. $fresh_{t,\ P}\ X \wedge t' \le t \supset fresh_{t',P}\ X$
e. $(\varphi\ at_P\ t_1)\ at_P\ t_2 \wedge t_1 \le t_2 \supset \varphi\ at_P\ t_1$

We also include analogous axiom for compound principals *CP*.
A9. Reduction axiom

$(\varphi\ at_P\ t_1)\ at_P\ t_2 \wedge t_2 \ge t_1 \supset \varphi\ at_P\ t_2$, where φ is $\psi\ at_P\ t$ for some ψ, P, and t, or φ is $Q\ says_t\ X$, $Q\ said_t\ X$, or $Q\ received_t\ X$ for some Q, X, and t.

We also include an analogous axiom for compound principals *CP*.
A10. Originator Identification

a. $\overset{K}{\Rightarrow}_{t,\ P}\ Q \wedge P\ received_t \lfloor X \rfloor_K^{-1} \supset Q\ said_{t,\ P}\ X \wedge Q\ said_{t,\ P} \lfloor X \rfloor_K^{-1}$
b. $\overset{K}{\Rightarrow}_{t,\ P}\ CP \wedge P\ received_t \lfloor X \rfloor_K^{-1} \supset CP\ said_{t,\ P}\ X \wedge CP\ said_{t,\ P} \lfloor X \rfloor_K^{-1}$
c. $\overset{K}{\Rightarrow}_{t,\ P}\ CP_{m,n} \wedge P\ received_t \lfloor X \rfloor_K^{-1} \supset CP\ said_{t,\ P}\ X \wedge CP\ said_{t,\ P} \lfloor X \rfloor_K^{-1}$

Principals use signatures to deduce the originator of messages.

Receiving

A11. $P\ received_t\ \{X\}_K \wedge P\ has_t\ K^{-1} \supset P\ received_t\ X$
A12. $P\ received_t \lfloor X \rfloor_K^{-1} \supset P\ received_t\ X$
A13. $CP\ received_t\ \{X\}_K \wedge CP\ has_t\ K^{-1} \supset CP\ received_t\ X$
A14. $CP\ received_t \lfloor X \rfloor_K^{-1} \supset CP\ received_t\ X$
Principals and compound principals can decrypt encrypted messages if they have decryption keys, and they can read signed messages with or without the signature verification key.

Saying

A15. $P\ said_t\ (X_1, \ldots, X_n) \supset P\ said_t\ X_i$
A16. $P\ says_t\ (X_1, \ldots, X_n) \supset P\ says_t\ X_i$
A17. $P\ said_t \lfloor X \rfloor_K^{-1} \supset P\ said_t\ X$
A18. $P\ says_t \lfloor X \rfloor_K^{-1} \supset P\ says_t\ X$
A19. $P\ said_t\ X \supset (\exists t' \ge t)\ P\ says_{t'}\ X$
A20. $P\ says_t\ X \supset P\ said_t\ X$

A17 and A18 state that principals are responsible for the contents of signed messages that they send, even if they did not sign them. If instead, P wants to forward a signed message $\lfloor X \rfloor_K^{-1}$ from Q without being responsible for the contents, P would send the messages "Q said$_t$ $\lfloor X \rfloor_K^{-1}$".

We also include analogous axioms for compound principal CP.

Freshness

A.21 $fresh_t\, X \supset fresh_t\, F(X,Y)$
It is assumed that the function F actually depends on the argument X.

Jurisdiction

A.22 $P\ controls_t\ \varphi \wedge P\ says_t\ \varphi \supset \varphi\ at_P\ t$
A.23 $CP\ controls_t\ \varphi \wedge CP\ says_t\ \varphi \supset \varphi\ at_{CP}\ t$

Access Control

Here P and Q are principals, $CP = \{P_1, \ldots, P_n\}$ and CP' are compound principals, m and n are integers, K is a public key, and G is a group.

Group Membership

A.24 $P\ controls_t\ Q \Rightarrow_{t'} G \wedge P\ says_t\ Q \Rightarrow_{t'} G \supset Q \Rightarrow_{t'} G\ at_P\ t$
A.25 $P\ controls_t\ CP' \Rightarrow_{t'} G \wedge P\ says_t\ CP' \Rightarrow_{t'} G \supset CP' \Rightarrow_{t'} G\ at_P\ t$
A.26 $P\ controls_t\ Q|K \Rightarrow_{t'} G \wedge P\ says_t\ Q|K \Rightarrow_{t'} G \supset Q|K \Rightarrow_{t'} G\ at_P\ t$
A.27 $P\ controls_t\ CP'|K \Rightarrow_{t'} G \wedge P\ says_t\ CP'|K \Rightarrow_{t'} G \supset CP'|K \Rightarrow_{t'} G\ at_P\ t$
A.28 $P\ controls_t\ CP'_{m,n} \Rightarrow_{t'} G \wedge P\ says_t\ CP'_{m,n} \Rightarrow_{t'} G \supset CP'_{m,n} \Rightarrow_{t'} G\ at_P\ t$

A.29 $CP\ controls_t\ Q \Rightarrow_{t'} G \wedge CP\ says_t\ Q \Rightarrow_{t'} G \supset Q \Rightarrow_{t'} G\ at_{CP}\ t$
A.30 $CP\ controls_t\ CP' \Rightarrow_{t'} G \wedge CP\ says_t\ CP' \Rightarrow_{t'} G \supset CP' \Rightarrow_{t'} G\ at_{CP}\ t$
A.31 $CP\ controls_t\ Q|K \Rightarrow_{t'} G \wedge CP\ says_t\ Q|K \Rightarrow_{t'} G \supset Q|K \Rightarrow_{t'} G\ at_{CP}\ t$
A.32 $CP\ controls_t\ CP'|K \Rightarrow_{t'} G \wedge CP\ says_t\ CP'|K \Rightarrow_{t'} G \supset CP'|K \Rightarrow_{t'} G\ at_{CP}\ t$
A.33 $CP\ controls_t\ CP'_{m,n} \Rightarrow_{t'} G \wedge CP\ says_t\ CP'_{m,n} \Rightarrow_{t'} G \supset CP'_{m,n} \Rightarrow_{t'} G\ at_{CP}\ t$

Note that these axioms are a direct result of the jurisdiction axiom.

A.34 $Q \Rightarrow_t G \wedge Q\ says_t\ X \supset G\ says_t\ X$
A.35 $Q|K \Rightarrow_t G \wedge \overset{K}{\Rightarrow}_{t,P} Q \wedge Q\ says_t \lfloor X \rfloor_K^{-1} \supset G\ says_t\ X$
A.36 $CP \Rightarrow_t G \wedge CP\ says_t\ X \supset G\ says_t\ X$
A.37 $CP|K \Rightarrow_t G \wedge \overset{K}{\Rightarrow}_{t,P} CP \wedge CP\ says_t \lfloor X \rfloor_K^{-1} \supset G\ says_t\ X$

A.38 $CP_{m,n} \Rightarrow_t G \wedge P_1\ says_t \lfloor X \rfloor_{K1}^{-1} \wedge P_2\ says_t \lfloor X \rfloor_{K2}^{-1} \wedge \ldots \wedge P_m\ says_t \lfloor X \rfloor_{Km}^{-1} \supset G\ says_t\ X$

Appendix C. Model of Computation - Semantics

We now present our model of computation. We consider a finite collection of *system principals* $\mathbf{P} = \{P_1, \ldots, P_k\}$ that communicate by sending messages to each other. We also consider a finite collection of *compound principals* $\mathbf{CP} = \{CP_1, \ldots, CP_l\}$ where each element $CP_i = \{P_1, \ldots, P_n\}$, that is, a set of n system principals that collectively send messages to each other and to system principals. There is an additional environmental principal P_e that represents relevant aspects external to the system principals. P_e's clock represents global time and each system principal has a clock corresponding to the principal's local time.

Formulas, messages and some basic values are as defined in Appendix B. Given a message M and a set $\boldsymbol{K}$ of keys, we define submsgs$_K$(M) to be the union of {M} and

a. submsgs$_K(X_1) \cup \ldots \cup$ submsgs$_K$ (Xn) if $M = (X_1, \ldots, X_n)$
b. submsgs$_K(X)$ if $M = \{X\}_K$ and $K \in \boldsymbol{K}$,
c. subsmsgs$_K(X)$ if $M = \lfloor X \rfloor_K^{-1}$
d. submsgs$_K(X)$ if $M = X\ at_P\ t$

Thus $\text{submsgs}_K(\text{M})$ constitutes the set of messages derivable from M by reading submessages and using keys in $\boldsymbol{K}$. The following are several basic events that are included in a principal's or a compound principal's set of events.

- send (X, P_j): send message X to P_j
- send (X, CP_j): send message X to CP_j
- receive(X): receive message X, and
- generate(X): generate message X.

In terms of a basic event v and a time t, the pair (v, t) describes a *timestamped event.* A *history* is a sequence of timestamped events that is possible infinite. If (v, t) is a timestamped event in the history $\boldsymbol{H}$, we write (v, t) $\in \boldsymbol{H}$. $\boldsymbol{H}$ is a sequential history at t_i if (1) $t_1 < t_2$ for every t_1, t_2 such that (v_1, t_1), (v_2, t_2) $\in \boldsymbol{H}$ and (v_1, t_1) appears earlier in the sequence than (v_2, t_2), and (2) $t \leq t_i$ for every (v, t) $\in \boldsymbol{H}$.

Each principal P_i ($1 \leq \text{i} \leq \text{n}$) is associated with a local state that contains all the information P_i has access to. This information includes P_i's identity, the current local time, keys P_i can use, and a history of events that have taken place for P_i. A tuple $s_\text{i} = (i, t_i, \boldsymbol{K}_\text{i}, \boldsymbol{H}_\text{i})$, where t_i is a time, $\boldsymbol{K}_\text{i}$ is a set of keys, and $\boldsymbol{H}_\text{i}$ is a sequential history at t_i formally defines a *local state*.

Each compound principal $CP_j = \{P_1, \ldots, P_\text{n}\}$ comprising of n principals ($1 \leq \text{j} \leq \text{l}$) is associated with a local state that contains all the information CP_j has access to. We assume that the clocks of all k principals comprising CP_j are synchronized. The local state includes CP_j's identity, the local time of CP_j (that is, the local time at any principal $P_\text{i} \in CP_\text{j}$), keys CP_j can use, and a history of events that have taken place for CP_j. A tuple $cs_\text{i} = (j, t_j, \boldsymbol{K}_\text{j}, \boldsymbol{H}_\text{j})$, where t_j is the local time at any one of the n principals which comprise CP_j , $\boldsymbol{K}_\text{j}$ is a set of keys, and $\boldsymbol{H}_\text{j}$ is a sequential history at t_j formally defines a *local state*.

The *environment* P_e captures relevant aspects of the system not captured in the local states of the principals, including message modification, replay, loss, and delay. The environment state contains

a. a time t_e, called the *real time*,
b. a sequential history $\boldsymbol{H}_\text{e}$ at t_e, and
c. for $1 \leq \text{i} \leq \text{n}$, a *message buffer* b_i for P_i containing the set of messages sent to P_i but not yet delivered.

The *global state* of a system with n principals is a (n+l+1)-tuple of the form (s_e, s_1, …, s_n, cs_1, cs_2, …, cs_l), where s_e is the environment state, each s_i ($1 \leq i \leq \text{n}$) is the local state of the principal P_i, and each cs_j ($1 \leq j \leq \text{l}$) is the local state of the compound principal CP_j. A *run* is a function r: $\text{R} \rightarrow \boldsymbol{G}$, associating a global state with every point of real time. Given a run r and a time t, we denote the local state of P_i in the global state $r(t)$ by $r_i(t)$. Similarly we denote the local state of a compound principal CP_j in the global state $r(t)$ by $r_\text{j}(t)$. We sometimes write (r, t) to denote $r(t)$, and call (r, t) a point.

Given a run r, a time t, and a principal Pi, $\text{Time}_{Pi}(r, t)$ is defined to be P_i's local time, i.e. the time in P_i's local state $r_i(t)$. Similarly, $\text{History}_{Pi}(r, t)$ is defined to be the history in the local state $r_i(t)$, and $\text{Keyset}_{Pi}(r, t)$ is defined to be the key set in the local state $r_i(t)$. We also define $\text{Msgs}_{Pi}(r, t) = \{M\text{: (receive(M),t')} \in \text{History}_{Pi}(r, t) \text{ for some t'}\}$, so $\text{Msgs}_{Pi}(r, t)$ is the set of messages received by P_i at or before time t in the run r, the set $\{t\text{: Time}_{Pi}(r, t) = \text{t}_i\}$ generally contains more than a single element. We write $\text{Start}_{Pi}(r, t_i)$ and $\text{End}_{Pi}(r, t_i)$ to denote the minimum and the maximum element, respectively, of this set.

Given a run r, a time t, and a compound principal CP_j, $\text{Time}_{CPj}(r, t)$ is defined to be CP_j's local time, i.e. the time in CP_j's local state $r_j(t)$. Similarly, $\text{History}_{CPj}(r, t)$ is defined to be the history in the local state $r_j(t)$, and $\text{Keyset}_{CPj}(r, t)$ is defined to be the key set in the local state $r_j(t)$. We also define $\text{Msgs}_{CPj}(r, t) = \{M\text{: (receive(M),t')} \in \text{History}_{CPj}(r, t) \text{ for some t'}\}$, so $\text{Msgs}_{CPj}(r, t)$ is the set of messages received by CP_j at or before time t in the run r, the set $\{t\text{: Time}_{CPj}(r, t) = \text{t}_i\}$ generally contains more than a single element. We write $\text{Start}_{CPj}(r, t_i)$ and $\text{End}_{CPj}(r, t_i)$ to denote the minimum and the maximum element, respectively, of this set.

A run r is legal if certain monotonicity and consistency conditions hold. Namely, r is legal if for every principal P,

a. if $t \leq t'$, then $\text{Time}_P(r, t) \leq t'$.
b. if $t \leq t'$, then $\text{Keyset}_P(r, t) \subseteq \text{Keyset}_P(r, t')$,
c. if $K \in \text{Keyset}_P(r, t)$ then either
   (a) (generate(K),t') $\in \text{History}_P(r, t)$ for some $t' \leq t$, or
   (b) there exists a sequence $\boldsymbol{K}_0, \boldsymbol{K}_1, \boldsymbol{K}_2, \ldots, \boldsymbol{K}_\text{m}$ such that $\boldsymbol{K}_m = \text{Keyset}_P(r, t)$, $\boldsymbol{K}_o = \text{Keyset}_P(r, t')$ for some $t' \leq t$, and if $\text{K} \in \boldsymbol{K}_l$, then either $\text{K} \in \boldsymbol{K}_{\text{l-1}}$ or
   $\text{K} \in \text{submsgs}_{\boldsymbol{K}_{\text{l-1}}}(\text{Msgs}_\text{P}(r, t))$
d. if (receive(X),t') $\in \text{History}_\text{P}(r, t)$, then (send(X, P),t'') $\in \text{History}_\text{Q}(r, t)$ for some Q and t'' such that $\text{End}_\text{Q}(r, t'') \leq \text{Start}_\text{P}(r, t')$,

and for every compound principal CP,

e. if $t \leq t'$, then $\text{Time}_{CP}(r, t) \leq t'$.
f. if $t \leq t'$, then $\text{Keyset}_{CP}(r, t) \subseteq \text{Keyset}_{CP}(r, t')$,
g. if $K \in \text{Keyset}_{CP}(r, t)$ then either
   (c) $(\text{generate}(K), t') \in \text{History}_{CP}(r, t)$ for some $t' \leq t$, or
   (d) there exists a sequence $\boldsymbol{K}_0, \boldsymbol{K}_1, \boldsymbol{K}_2, \ldots, \boldsymbol{K}_m$ such that $\boldsymbol{K}_m = \text{Keyset}_{CP}(r, t)$, $\boldsymbol{K}_o = \text{Keyset}_{CP}(r, t')$ for some $t' \leq t$, and if $K \in \boldsymbol{K}_l$, then either $K \in \boldsymbol{K}_{l-1}$ or $K \in \text{submsgs}_{\boldsymbol{K}_{l-1}}(\text{Msgs}_{CP}(r, t))$.
h. if $(\text{receive}(X), t') \in \text{History}_{CP}(r, t)$, then $(\text{send}(X, P), t'') \in \text{History}_{Q}(r, t)$ for some Q and t'' such that $\text{End}_{Q}(r, t'') \leq \text{Start}_{CP}(r, t')$.

A *system* is a set of legal runs. We will typically be interested in a set of runs consisting of relevant events occurring before the start of a protocol followed by possible executions of that protocol.

Truth Conditions

In this section, we inductively define the truth of formulas in our model. An *interpretation* π is a function π: $\boldsymbol{G} \times \varphi \rightarrow \{T, F\}$ assigning truth-values to primitive propositions at each global state. A pair $\boldsymbol{I} = (\boldsymbol{R}, \pi)$ is an interpreted system. The truth of the formulas is evaluated with respect to a given point (r, t) in an interpreted system I. We write $(\boldsymbol{I}, r, t) \models \varphi$ to indicate that φ is true at the point (r, t) of $\boldsymbol{I}$, and $(\boldsymbol{I}, r, t) \models \neg\varphi$ to indicate that φ is not true at the point (r, t) of $\boldsymbol{I}$. Where $\boldsymbol{I}$ is clear from context, we write simply $(r, t) \models \varphi$ and $(r, t) \models \neg\varphi$.

We define our truth conditions in such a way that for nonnegated basic formulas, only formulas about the past can be true. We do this in order to maintain a certain kind of stability of formulas that is important to the soundness of the logic. In particular, we only define statements that can be revoked, such as statements about belief, control, group membership, delegation and keys, to be true at a given time on a principal's clock if the principal's clock has changed from that time, since otherwise it would be possible to construct runs in which the same statement is both true and false at the same time.

Primitive Propositions $(r, t) \models p$ if and only if $(r, t) \in \pi(p)$ for primitive proposition p.

Logical Connectives

$(r, t) \models \varphi \wedge \psi$ if and only if $(r, t) \models \varphi$ and $(r, t) \models \psi$.
$(r, t) \models \neg\varphi$ if and only $(r, t) \not\models \varphi$.
$(r, t) \models (\exists t' : t_1 \leq t' \leq t_2)\varphi$ if and only if $(r, t) \models \varphi$ for some t' such that $t_1 \leq t' \leq t_2$.
$(r, t) \models (\exists t_1 \geq t_2)\varphi$ if and only if $(r, t) \models \varphi$ for some t_1 such that $t_1 \geq t_2$.
$(r, t) \models (\exists t_1 \leq t_2)\varphi$ if and only if $(r, t) \models \varphi$ for some t_1 such that $t_1 \leq t_2$.
$(r, t) \models (\forall t' : t_1 \leq t' \leq t_2)\varphi$ if and only if $(r, t) \models \varphi$ for every t' such that $t_1 \leq t' \leq t_2$.
$(r, t) \models (\forall t_1 \geq t_2)\varphi$ if and only if $(r, t) \models \varphi$ for every t_1 such that $t_1 \geq t_2$.
$(r, t) \models (\forall t_1 \leq t_2)\varphi$ if and only if $(r, t) \models \varphi$ for every t_1 such that $t_1 \leq t_2$.

Relations

$(r, t) \models t_1 \geq t_2$ if and only if $t_1 \geq t_2$.
$(r, t) \models t_1 \leq t_2$ if and only if $t_1 \leq t_2$.

Receives P derives X from the set of received messages and P's current key set

$(r, t) \models P\ received_{t'}\ X$ if and only if $t' \leq \text{Time}_P(r, t)$ and $X \in \text{submsgs}_{\text{Keyset P}(r, t)}(\text{Msgs}(r, t))$.
$(r, t) \models P\ received_{[t_1, t_2]} X$ if and only if $(r, t) \models P\ received_{t'}\ X$ for every $t_1 \leq t' \leq t_2$, and
$(r, t) \models P\ received_{\langle t_1, t_2 \rangle} X$ if and only if $(r, t) \models P\ received_{t'}\ X$ for some $t_1 \leq t' \leq t_2$.

CP derives X from the set of received messages and CP's current key set

$(r, t) \models CP\ received_{t'}\ X$ if and only if $t' \leq \text{Time}_{CP}(r, t)$ and $X \in \text{submsgs}_{\text{Keyset CP}(r, t)}(\text{Msgs}(r, t))$.
$(r, t) \models CP\ received_{[t_1, t_2]} X$ if and only if $(r, t) \models CP\ received_{t'}\ X$ for every $t_1 \leq t' \leq t_2$, and
$(r, t) \models CP\ received_{\langle t_1, t_2 \rangle} X$ if and only if $(r, t) \models CP\ received_{t'}\ X$ for some $t_1 \leq t' \leq t_2$.

Says

$(r, t) \models P\ says_{t'}\ X$ if and only if $t' \le \text{Time}_P(r, t)$ and there is some message M and principal $Q \in P \vee Q \in CP$ such that $(\text{send}(M, Q), t') \in \text{History}_P(r, t)$ and $X \in \text{submsgs}_{\text{Keyset P}(r, t')}(M)$.

$(r, t) \models P\ says_{[t_1, t_2]} X$ if and only if $(r, t) \models P\ says_{t'}\ X$ for every $t_1 \le t' \le t_2$, and

$(r, t) \models P\ says_{\langle t_1, t_2 \rangle} X$ if and only if $(r, t) \models P\ says_{t'}\ X$ for some $t_1 \le t' \le t_2$.

$(r, t) \models CP\ says_{t'}\ X$ if and only if $t' \le \text{Time}_{CP}(r, t)$ and there is some message M and principal $Q \in P \vee Q \in CP$ such that $(\text{send}(M, Q), t') \in \text{History}_{CP}(r, t)$ and $X \in \text{submsgs}_{\text{Keyset CP}(r, t')}(M)$.

$(r, t) \models CP\ says_{[t_1, t_2]} X$ if and only if $(r, t) \models CP\ says_{t'}\ X$ for every $t_1 \le t' \le t_2$, and

$(r, t) \models CP\ says_{\langle t_1, t_2 \rangle} X$ if and only if $(r, t) \models CP\ says_{t'}\ X$ for some $t_1 \le t' \le t_2$.

Said

$(r, t) \models P\ said_{t'}\ X$ if and only if $t' \le \text{Time}_P(r, t)$ and there is some $t'' \le t'$ such that $P\ says_{t''}\ X$.

$(r, t) \models P\ said_{[t_1, t_2]} X$ if and only if $(r, t) \models P\ said_{t'}\ X$ for every $t_1 \le t' \le t_2$, and

$(r, t) \models P\ said_{\langle t_1, t_2 \rangle} X$ if and only if $(r, t) \models P\ said_{t'}\ X$ for some $t_1 \le t' \le t_2$.

$(r, t) \models CP\ said_{t'}\ X$ if and only if $t' \le \text{Time}_{CP}(r, t)$ and there is some $t'' \le t'$ such that $CP\ says_{t''}\ X$.

$(r, t) \models CP\ said_{[t_1, t_2]} X$ if and only if $(r, t) \models CP\ said_{t'}\ X$ for every $t_1 \le t' \le t_2$, and

$(r, t) \models CP\ said_{\langle t_1, t_2 \rangle} X$ if and only if $(r, t) \models CP\ said_{t'}\ X$ for some $t_1 \le t' \le t_2$.

Controls Principals neither lie about formulas they control, nor make contradictory statements about formulas they control using the same timestamp.

$(r, t) \models P\ controls_{t'}\ \varphi$ if and only if (1) $t' \le \text{Time}_P(r, t)$ and (2) $(r, t) \models P\ says_{t'}\ \varphi$ implies $(r, t) \models \varphi\ at_P\ t'$.

$(r, t) \models P\ controls_{[t_1, t_2]}\ \varphi$ if and only if $(r, t) \models P\ controls_{t'}\ \varphi$ for every $t_1 \le t' \le t_2$, and

$(r, t) \models P\ controls_{\langle t_1, t_2 \rangle} \varphi$ if and only if $(r, t) \models P\ controls_{t'}\ \varphi$ for some $t_1 \le t' \le t_2$.

$(r, t) \models CP\ controls_{t'}\ \varphi$ if and only if (1) $t' \le \text{Time}_{CP}(r, t)$ and (2) $(r, t) \models CP\ says_{t'}\ \varphi$ implies $(r, t) \models \varphi\ at_P\ t'$.

$(r, t) \models CP\ controls_{[t_1, t_2]}\ \varphi$ if and only if $(r, t) \models CP\ controls_{t'}\ \varphi$ for every $t_1 \le t' \le t_2$, and

$(r, t) \models CP\ controls_{\langle t_1, t_2 \rangle} \varphi$ if and only if $(r, t) \models CP\ controls_{t'}\ \varphi$ for some $t_1 \le t' \le t_2$.

Fresh A message is fresh if it has not been said before in the run.

$(r, t) \models P\ fresh_{t'}\ X$ if and only if $t' \le \text{Time}_P(r, t)$ and $(r, t) \models \neg Q\ said_{t'}\ X$ for all principals (and compound principals) Q.

$(r, t) \models CP\ fresh_{t'}\ X$ if and only if $t' \le \text{Time}_{CP}(r, t)$ and $(r, t) \models \neg Q\ said_{t'}\ X$ for all principals (and compound principals) Q.

Public Signature Verification Keys Signature checking keys are good if they properly identify signatures.

$(r, t) \models (\overset{K}{\Rightarrow}_{t', Q} P)$ if and only if (1) $t' \le \text{Time}_P(\text{r})$ and (2) $Q\ received_{t'} \lfloor X \rfloor_K^{-1}$ implies $(r, t) \models P\ said_{t'}\ X$.

$(r, t) \models (\overset{K}{\Rightarrow}_{t', Q} P)$ if and only if $(r, t) \models (\overset{K}{\Rightarrow}_{t', Q} P)$ for every $t_1 \le t' \le t_2$, and

$(r, t) \models (\overset{K}{\Rightarrow}_{t', Q} P)$ if and only if $(r, t) \models (\overset{K}{\Rightarrow}_{t', Q} P)$ for some $t_1 \le t' \le t_2$.

For compound principals the private key is shared among the principals comprising the compound principal. The corresponding public-key is then good if it properly identifies signatures on behalf of the compound principal.

$(r, t) \models (\overset{K}{\Rightarrow}_{t', Q} CP)$ if and only if (1) $t' \le \text{Time}_{CP}(\text{r})$ and (2) $Q\ received_{t'} \lfloor X \rfloor_K^{-1}$ implies $(r, t) \models CP\ said_{t'}\ X$.

$(r, t) \models (\overset{K}{\Rightarrow}_{t',Q} CP)$ if and only if $(r, t) \models (\overset{K}{\Rightarrow}_{t',Q} CP)$ for every $t_1 \le t' \le t_2$, and

$(r, t) \models (\overset{K}{\Rightarrow}_{t',Q} CP)$ if and only if $(r, t) \models (\overset{K}{\Rightarrow}_{t',Q} CP)$ for some $t_1 \le t' \le t_2$.

We further allow the private key to be shared in a threshold manner such that for $CP = \{P_1, \ldots, P_n\}$ any m of the n system principals comprising the compound principal can sign messages on behalf of the compound principal.

$(r, t) \models (\overset{K}{\Rightarrow}_{t',Q} CP_{m,n})$ if and only if (1) $t' \le \text{Time}_{CP}(r)$ and (2) Q *received*$_{t'}$ $\lfloor X \rfloor_K^{-1}$ implies $(r, t) \models CP$ *said*$_{t'}$ X.

$(r, t) \models (\overset{K}{\Rightarrow}_{t',Q} CP_{m,n})$ if and only if $(r, t) \models (\overset{K}{\Rightarrow}_{t',Q} CP_{m,n})$ for every $t_1 \le t' \le t_2$, and

$(r, t) \models (\overset{K}{\Rightarrow}_{t',Q} CP_{m,n})$ if and only if $(r, t) \models (\overset{K}{\Rightarrow}_{t',Q} CP_{m,n})$ for some $t_1 \le t' \le t_2$.

Believes We define the possibility relation $\sim_{Pi}$ for a principal P_i in a state (r, t) by $(r, t) \sim_i (r', t')$ if $r_i(t) = r'_i(t')$. Hence $\sim_i$ defines an equivalence relation for P_i among global states. We define belief relative to this equivalence class.

$(r, t) \models P$ *believes*$_{t'}$ φ if and only if $t' \le \text{Time}_P(r, t)$ and $(r',t') \models \varphi$ *at*$_P$ t' for all (r', t') such that $(r', t') \sim_P (r, t)$.

$(r, t) \models P$ *believes*$_{[t_1, t_2]}$ φ if and only if $(r, t) \models P$ *believes*$_{t'}$ φ for every $t_1 \le t' \le t_2$, and

$(r, t) \models P$ *believes*$_{\langle t_1, t_2 \rangle}$ φ if and only if $(r, t) \models P$ *believes*$_{t'}$ φ for some $t_1 \le t' \le t_2$.

Similarly, we define the possibility relation $\sim_{CPi}$ for a compound principal CP_i in a state (r, t) by $(r, t) \sim_i (r', t')$ if $r_i(t) = r'_i(t')$. Hence $\sim_i$ defines an equivalence relation for CP_i among global states. We define belief relative to this equivalence class.

$(r, t) \models CP$ *believes*$_{t'}$ φ if and only if $t' \le \text{Time}_{CP}(r, t)$ and $(r',t') \models \varphi$ *at*$_{CP}$ t' for all (r', t') such that $(r', t') \sim_{CP} (r, t)$.

$(r, t) \models CP$ *believes*$_{[t_1, t_2]}$ φ if and only if $(r, t) \models CP$ *believes*$_{t'}$ φ for every $t_1 \le t' \le t_2$, and

$(r, t) \models CP$ *believes*$_{\langle t_1, t_2 \rangle}$ φ if and only if $(r, t) \models CP$ *believes*$_{t'}$ φ for some $t_1 \le t' \le t_2$.

At $(r, t) \models \varphi$ *at*$_P$ t' if and only if $t' \le \text{Time}_P(t)$ and $(r', t'') \models \varphi$ for all t'' such that $\text{Start}_P(r, t') \le t'' \le \text{End}_P(r, t')$.

$(r, t) \models \varphi$ *at*$_P$ $[t_1, t_2]$ if and only if $(r, t) \models \varphi$ *at*$_P$ t' for every $t_1 \le t' \le t_2$, and

$(r, t) \models \varphi$ *at*$_P$ $\langle t_1, t_2 \rangle$ if and only if $(r, t) \models \varphi$ *at*$_P$ t' for some $t_1 \le t' \le t_2$.

$(r, t) \models \varphi$ *at*$_{CP}$ t' if and only if $t' \le \text{Time}_{CP}(t)$ and $(r', t'') \models \varphi$ for all t'' such that $\text{Start}_{CP}(r, t') \le t'' \le \text{End}_{CP}(r, t')$.

$(r, t) \models \varphi$ *at*$_{CP}$ $[t_1, t_2]$ if and only if $(r, t) \models \varphi$ *at*$_{CP}$ t' for every $t_1 \le t' \le t_2$, and

$(r, t) \models \varphi$ *at*$_{CP}$ $\langle t_1, t_2 \rangle$ if and only if $(r, t) \models \varphi$ *at*$_{CP}$ t' for some $t_1 \le t' \le t_2$.

Access Control We define a principal G that denotes a group, which can be found on an object's ACL. If a principal P or a compound principal CP *speaks for* a group G, then P (CP) can claim to be present on the object's ACL as well. In other words, P (CP) is then a group member of group G.

$(r, t) \models P \Rightarrow_{t',R} G$ if and only if (1) $t' \le \text{Time}_R(t)$, and (2) $(r, t) \models (P$ *says*$_t$ $X)$ *at*$_R$ t' implies $(r, t) \models (G$ *says*$_t$ $X)$ *at*$_R$ t'

$(r, t) \models P \Rightarrow_{[t_1, t_2],R} G$ if and only if $(r, t) \models P \Rightarrow_{t',R} G$ for every $t_1 \le t' \le t_2$, and

$(r, t) \models P \Rightarrow_{\langle t_1, t_2 \rangle,R} G$ if and only if $(r, t) \models P \Rightarrow_{t',R} G$ for some $t_1 \le t' \le t_2$.

$(r, t) \models CP \Rightarrow_{t',R} G$ if and only if (1) $t' \le \text{Time}_R(t)$, and (2) $(r, t) \models (CP$ *says*$_t$ $X)$ *at*$_R$ t' implies $(r, t) \models (G$ *says*$_t$ $X)$ *at*$_R$ t'

$(r, t) \models CP \Rightarrow_{[t_1, t_2],R} G$ if and only if $(r, t) \models CP \Rightarrow_{t',R} G$ for every $t_1 \le t' \le t_2$, and

$(r, t) \models CP \Rightarrow_{\langle t_1, t_2 \rangle,R} G$ if and only if $(r, t) \models CP \Rightarrow_{t',R} G$ for some $t_1 \le t' \le t_2$.

We further allow a threshold subset of the compound principal to *speak for* a group. In this case the compound principal need not have a public-key. Instead, the system principals comprising the compound principal can make statements on behalf of the compound principal for access to objects via group *G*. The privileges distributed in this threshold structure are bound to specific public-keys that are initially specified in the definition of the compound principal (one public-key per system principal).

For $CP = \{P_1|K_1, P_2|K_2, \ldots P_n|K_n\}$

$(r, t) \models CP_{m,n} \Rightarrow_{t',R} G$ where $(m \leq n)$

if and only if (1) $t' \leq \text{Time}_R(t)$, and

(2) $(r, t) \models \{(P_1 \text{ says}_t \lfloor X \rfloor_{K1}^{-1})\ at_R\ t' \wedge (P_2 \text{ says}_t \lfloor X \rfloor_{K2}^{-1})\ at_R\ t' \wedge \ldots \wedge (P_m \text{ says}_t \lfloor X \rfloor_{Km}^{-1})\ at_R\ t'\}$ implies $(r, t) \models (G \text{ says}_t X)\ at_R\ t'$

$(r, t) \models CP \Rightarrow_{[t_1, t_2],R} G$ if and only if $(r, t) \models CP \Rightarrow_{t',R} G$ for every $t_1 \leq t' \leq t_2$, and

$(r, t) \models CP \Rightarrow_{\langle t_1, t_2\rangle,R} G$ if and only if $(r, t) \models CP \Rightarrow_{t',R} G$ for some $t_1 \leq t' \leq t_2$.

To enable selective distribution of access privileges we allow a principal (or a compound principal) to be bound to a specific key when being a group member.

$(r, t) \models P|K \Rightarrow_{t',R} G$

if and only if (1) $t' \leq \text{Time}_R(t)$, (2) $(r, t) \models (\overset{K}{\Rightarrow}_{t',R} P)$, and (3) $(r, t) \models (P \text{ says}_t \lfloor X \rfloor_K^{-1})\ at_R\ t'$ implies $(r, t) \models (G \text{ says}_t X)\ at_R\ t'$

$(r, t) \models P \Rightarrow_{[t_1, t_2],R} G$ if and only if $(r, t) \models P \Rightarrow_{t',R} G$ for every $t_1 \leq t' \leq t_2$, and

$(r, t) \models P \Rightarrow_{\langle t_1, t_2\rangle,R} G$ if and only if $(r, t) \models P \Rightarrow_{t',R} G$ for some $t_1 \leq t' \leq t_2$.

$(r, t) \models CP|K \Rightarrow_{t',R} G$

if and only if (1) $t' \leq \text{Time}_R(t)$, (2) $(r, t) \models (\overset{K}{\Rightarrow}_{t',R} CP)$, and (3) $(r, t) \models (CP \text{ says}_t \lfloor X \rfloor_K^{-1})\ at_R\ t'$ implies $(r, t) \models (G \text{ says}_t X)\ at_R\ t'$

$(r, t) \models CP \Rightarrow_{[t_1, t_2],R} G$ if and only if $(r, t) \models CP \Rightarrow_{t',R} G$ for every $t_1 \leq t' \leq t_2$, and

$(r, t) \models CP \Rightarrow_{\langle t_1, t_2\rangle,R} G$ if and only if $(r, t) \models CP \Rightarrow_{t',R} G$ for some $t_1 \leq t' \leq t_2$.

Appendix D. Soundness

Our logic is sound, in the sense that any derivation allowed by the logic corresponds to a truth in the model. For any formula φ and a set Γ of formulas, we write $\Gamma \models \varphi$ if $(r, t) \models \varphi$ for all (r, t) such that $(r, t) \models \psi$ for every $\psi \in \Gamma$.

Theorem *Let* Γ *be a set of formulas and* φ *be a formula. If* $\Gamma \vdash\!\!\!- \varphi$ *then* $\Gamma \models \varphi$.

Proof: This is a tedious proof that involves demonstrating (1) the validity of all axioms, and (2) preservation of truth in the derivation. We first show that these axioms are true on all worlds. This is an easy conclusion reached by inspection of the truth conditions in Appendix C. We then show that all the ways of deriving a formula φ from Γ preserve truth.

For any axiom if the antecedent is false at (r, t) then, the conditional is true. We need to show that if the antecedent is true then from our truth conditions we must be able to conclude that the conditional is always true. This requires a case-by-case analysis of all our axioms. Specifically, for each axiom $\psi \supset \psi'$, we show that if $(r, t) \vdash\!\!\!- \psi'$, then $(r, t) \vdash\!\!\!- \psi$. It therefore follows that if φ was obtained by modus ponens from ψ, then since $\Gamma \models \psi$, $\Gamma \models \varphi$.

We show this for the axioms that we have added to those presented in [25]. This includes the axioms for distributed private key shares and access control.

Distributed Private Key Shares

A10. Originator Identification

a. $\overset{K}{\Rightarrow}_{t, P} \text{CP} \wedge \text{P received}_t \lfloor \text{X} \rfloor_K^{-1} \supset \text{CP said}_{t, P} \text{X} \wedge \text{CP said}_{t, P} \lfloor \text{X} \rfloor_K^{-1}$

If the antecedent is true at (r, t) then both its conjuncts are true. But, from our truth conditions

$(r, t) \models \overset{K}{\Rightarrow}_{t, P} CP$ implies that if $(r, t) \models P \text{ received}_t \lfloor X \rfloor_K^{-1}$ then $(r, t) \models CP \text{ said}_{t, P} X$. Hence axiom A10a is true on all worlds

b. $\overset{K}{\Rightarrow}_{t,P} CP_{m,n} \wedge P\ received_t \lfloor X \rfloor_K^{-1} \supset CP\ said_{t,P}\ X \wedge CP\ said_{t,P} \lfloor X \rfloor_K^{-1}$

If the antecedent is true at (r, t) then both its conjuncts are true. But, from our truth conditions $(r, t) \models \overset{K}{\Rightarrow}_{t,P} CP_{m,n}$ implies that if $(r, t) \models P\ received_t \lfloor X \rfloor_K^{-1}$ then $(r, t) \models CP\ said_{t,P}\ X$. Hence axiom A10b is true on all worlds

Access Control (Group Membership)

A.24 $P\ controls_t\ Q \Rightarrow_{t'} G \wedge P\ says_t\ Q \Rightarrow_{t'} G \supset Q \Rightarrow_{t'} G\ at_P\ t$

If we let X = ($Q \Rightarrow_{t'} G$), then this axioms follows directly from the jurisdiction axiom A22.
Similarly, Axioms A25 – A28 follow directly from the jurisdiction axiom A22.

A.29 $CP\ controls_t\ Q \Rightarrow_{t'} G \wedge CP\ says_t\ Q \Rightarrow_{t'} G \supset Q \Rightarrow_{t'} G\ at_{CP}\ t$

If we let X = ($Q \Rightarrow_{t'} G$), then this axioms follows directly from the jurisdiction axiom A23
Similarly, Axioms A30 - A33 follow directly from the jurisdiction axiom A23.

A.34 $Q \Rightarrow_t G \wedge Q\ says_t\ X \supset G\ says_t\ X$

If the antecedent is true at (r, t) then both its conjuncts are true. But, from our truth conditions $(r, t) \models Q \Rightarrow_t G$ implies that if $(r, t) \models Q\ says_t\ X$ then $(r, t) \models G\ says_t\ X$. Hence axiom A34 is true on all worlds.

A.35 $Q|K \Rightarrow_t G \wedge \overset{K}{\Rightarrow}_{t,P} Q \wedge Q\ says_t \lfloor X \rfloor_K^{-1} \supset G\ says_t\ X$

If the antecedent is true at (r, t) then both its conjuncts are true. But, from our truth conditions $(r, t) \models Q|K \Rightarrow_t G$ implies that if $(r, t) \models Q\ says_t \lfloor X \rfloor_K^{-1}$ and $\overset{K}{\Rightarrow}_{t,P} Q$ then $(r, t) \models G\ says_t\ X$. Hence axiom A35 is true on all worlds.

A.36 $CP \Rightarrow_t G \wedge CP\ says_t\ X \supset G\ says_t\ X$

If the antecedent is true at (r, t) then both its conjuncts are true. But, from our truth conditions $(r, t) \models CP \Rightarrow_t G$ implies that if $(r, t) \models CP\ says_t\ X$ then $(r, t) \models G\ says_t\ X$. Hence axiom A36 is true on all worlds.

A.37 $CP|K \Rightarrow_t G \wedge \overset{K}{\Rightarrow}_{t,P} CP \wedge CP\ says_t \lfloor X \rfloor_K^{-1} \supset G\ says_t\ X$

If the antecedent is true at (r, t) then both its conjuncts are true. But, from our truth conditions $(r, t) \models CP|K \Rightarrow_t G$ implies that if $(r, t) \models CP\ says_t \lfloor X \rfloor_K^{-1}$ and $\overset{K}{\Rightarrow}_{t,P} CP$ then $(r, t) \models G\ says_t\ X$. Hence axiom A37 is true on all worlds.

A.38 $CP_{m,n} \Rightarrow_t G \wedge P_1\ says_t \lfloor X \rfloor_{K1}^{-1} \wedge P_2\ says_t \lfloor X \rfloor_{K2}^{-1} \wedge \ldots \wedge P_m\ says_t \lfloor X \rfloor_{Km}^{-1} \supset G\ says_t\ X$

If the antecedent is true at (r, t) then both its conjuncts are true. But, from our truth conditions $(r, t) \models CP_{m,n} \Rightarrow_t G$ implies that if $(r, t) \models (P_1\ says_t \lfloor X \rfloor_{K1}^{-1} \wedge P_2\ says_t \lfloor X \rfloor_{K2}^{-1} \wedge \ldots \wedge P_m\ says_t \lfloor X \rfloor_{Km}^{-1})$ then $(r, t) \models G\ says_t\ X$. Hence axiom A38 is true on all worlds.

There are three ways of deriving a formula φ from Γ. (1) If φ is a theorem or if $\varphi \in \Gamma$, then $\Gamma \models \varphi$ trivially. (2) If φ is obtained by modus ponens, then it appears in the derivation from Γ. In this derivation there is some ψ and $\psi \supset \varphi$ that appears earlier. By induction on the structure of the derivation, the truth conditions and our above proof of validity of all axioms, $\Gamma \models \varphi$. (3) If φ was obtained by necessitation from some ψ then $\vdash \psi$ and $\varphi = P\ believes_{t,P}\ \psi$ for some P. Since ψ is a theorem, $\models \psi$. Hence for any r, t, and P, $(r, t) \models \psi$. From the truth conditions for believe it follows that $(r, t) \models P\ believes_{t,P}\ \psi$ for every (r, t). Since $\varphi = P\ believes_{t,P}\ \psi$, $\Gamma \models \varphi$.

Appendix E. An Authorization Protocol

Here we present the authorization protocol for joint administration of access policies. The protocol is developed in our logic and is based on the sound axioms of the logic. The authorization protocol is applied to *access requests* such as those illustrated in Figure 2 for the coalition scenario illustrated in Figure 1. We refer to Figure 1 and apply the authorization

protocol on an access request using initial beliefs and our logic axioms to approve or deny the request. The protocol thus enables us to reason about the joint ownership of coalition resources such as *Object O*.

Initial Beliefs

All beliefs held or deduced by *Server P* stem from the central *AA*'s shared public key K_{AA}. *Server P* trusts the central *AA* for distribution of threshold attribute certificates to all coalition users and trusts each domain's Identity *CA* for distribution of identity certificates to users of that domain. In Statement 1 below, *P* believes that *AA*'s public key K_{AA} is owned by all three domains *D1*, *D2* and *D3* that have shares of the private key K_{AA}^{-1}. *CP* is a compound principal and *CP* = {*D1*, *D2*, *D3*}. Therefore, all messages that carry signatures of key K_{AA}^{-1} are signed by the compound principal *CP* (i.e., by all three domains) using the joint signature algorithm. Though the *AA* only distributes this signed message, for ease of reading we say that *AA* signs messages with key K_{AA}^{-1} as well.

P believes that *AA* has jurisdiction over all group membership certificates (for system principals *Q'* and compound principals *CP'*) for all groups (*G'*) at *AA* and that *AA* also has jurisdiction over the time that time-stamped certificates are believed accurate for all times after *t**. These beliefs are represented in our logic as follows:

1. $P\ believes_{t0}\ (\forall t \geq t^*)\ \overset{K_{AA}}{\Rightarrow}_{[t^*,\,t],\,P}\ CP_{3,3}$
2. $P\ believes_{t0}\ (\forall t)\ AA\ controls_{t}\ (\forall G', Q', t'_b, t'_e)\ Q' \Rightarrow_{[t'_b,t'_e],\,AA} G'$
3. $P\ believes_{t0}\ (\forall t)\ AA\ controls_{t}\ (\forall G', CP', t'_b, t'_e)\ CP' \Rightarrow_{[t'_b,t'_e],\,AA} G'$
4. $P\ believes_{t0}\ (\forall t \geq t^*)\ AA\ controls_{\,[t^*,\,t],\,P}\ (\forall G', Q', t'_b, t'_e, t'_{AA})\ AA\ says_{t'aa}\ Q' \Rightarrow_{[t'_b,t'_e],\,AA} G'$
5. $P\ believes_{t0}\ (\forall t \geq t^*)\ AA\ controls_{\,[t^*,\,t],\,P}\ (\forall G', CP', t'_b, t'_e, t'_{AA})\ AA\ says_{t'aa}\ CP' \Rightarrow_{[t'_b,t'_e],\,AA} G'$

P believes that *CA1*, *CA2*, and *CA3* of domains *D1*, *D2,* and *D3*, respectively, have jurisdiction over the public-key identity certificates for users in their domains and that the *CA*s also have jurisdiction over the time that time-stamped certificates are believed accurate for all times after *t**. Furthermore, *P* believes that keys K_{CA1}, K_{CA2}, and K_{CA3} are the public-keys of *CA1*, *CA2*, and *CA3* respectively.

6. $P\ believes_{t0}\ (\forall t)\ CA1\ controls_{t}\ (\forall Q', K_{Q'}, t'_b, t'_e)\ \overset{K_{Q'}}{\Rightarrow}_{[t'_b,t'_e],\,CA1}\ Q'$
7. $P\ believes_{t0}\ (\forall t \geq t^*)\ CA1\ controls_{[t^*,\,t],\,P}\ (\forall Q', K_{Q'}, t'_b, t'_e, t'_{ca1})\ CA1\ says_{t'ca1} \overset{K_{Q'}}{\Rightarrow}_{[t'_b,t'_e],\,CA1}\ Q'$
8. $P\ believes_{t0}\ (\forall t)\ CA2\ controls_{t}\ (\forall Q', K_{Q'}, t'_b, t'_e)\ \overset{K_{Q'}}{\Rightarrow}_{[t'_b,t'_e],\,CA2}\ Q'$
9. $P\ believes_{t0}\ (\forall t \geq t^*)\ CA2\ controls_{[t^*,\,t],\,P}\ (\forall Q', K_{Q'}, t'_b, t'_e, t'_{ca2})\ CA2\ says_{t'ca2} \overset{K_{Q'}}{\Rightarrow}_{[t'_b,t'_e],\,CA2}\ Q'$
10. $P\ believes_{t0}\ (\forall t)\ CA3\ controls_{t}\ (\forall Q', K_{Q'}, t'_b, t'_e)\ \overset{K_{Q'}}{\Rightarrow}_{[t'_b,t'_e],\,CA3}\ Q'$
11. $P\ believes_{t0}\ (\forall t \geq t^*)\ CA3\ controls_{[t^*,\,t],\,P}\ (\forall Q', K_{Q'}, t'_b, t'_e, t'_{ca3})\ CA3\ says_{t'ca3} \overset{K_{Q'}}{\Rightarrow}_{[t'_b,t'_e],\,CA3}\ Q'$

For access control of jointly owned resource *Object O*, *AA* establishes two groups *G_write* and *G_read*. Membership to group *G_write* allows a principal (or a compound principal) write privileges for *Object O* and membership to group *G_read* grants read privileges for *Object O*. These groups appear on *Object O*'s ACL managed at *Server P*. The ACL is a simple disjunction of expressions associated with *Object O*. That is, ACL_O: {E_0, E_1, …, E_n} where each expressions E_i = (*G*, access permissions) for a group *G*. Setting and updating policy objects is handled in a manner similar to that of accessing objects. That is, threshold attribute certificates are distributed that grant certain coalition users the authority to modify policy objects.

Access Request Verification

We now illustrate our authorization protocol by verifying an *access request* that comprises a signed request, identity certificates and a threshold attribute certificate. The access request is for a write operation on *Object O* and is sent by *User_D1* as illustrated in Figure 2 (b). The identity certificates in this access request are those issued to *User_D1* and *User_D2* by their respective domain *CA*s - *CA1* and *CA2*. The threshold attribute certificate in the access request is the one issued to users *User_D1*, *User_D2*, and *User_D3* by *AA* granting them group membership (in a 2-of-3 threshold manner) to group *G_write*. In the idealized threshold attribute certificate below, *CP* is a compound principal and *CP* = {*User_D1*|K_{user_D1}, *User_D2*|K_{user_D2}, *User_D3*|K_{user_D3}}. The threshold attribute certificate includes the set of principals comprising *CP* but for ease of reading we do not include it here. The access request with idealized certificates is as follows:

Message 1-1 (Identity certificate) $User\_D1 \rightarrow_{t1} P: \lfloor CA1\ says_{tCA1} \overset{K_{user\_D1}}{\Rightarrow}_{[tb,te]} User\_D1 \rfloor_{K_{CA1}}^{-1}$

Message 1-2 (Identity certificate) $User\_D1 \rightarrow_{t1} P: \lfloor CA2\ says_{tCA2} \overset{K_{user\_D2}}{\Rightarrow}_{[tb,te]} User\_D2 \rfloor_{K_{CA2}}^{-1}$

Message 1-3 (Threshold Attribute Certificate) $User\_D1 \rightarrow_{t1} P: \lfloor AA\ says_{taa}\ CP_{2,3} \Rightarrow_{[tb',te']} G\_write \rfloor_{K_{AA}}^{-1}$

Message 1-4 (Signed Request)

$User\_D1 \rightarrow_{t1} P: \{ \lfloor User\_D1\ says_{tu1}$ "write" $O \rfloor_{Kuser\_D1}^{-1}, \lfloor User\_D2\ says_{tu2}$ "write" $O \rfloor_{Kuser\_D2}^{-1} \}$

We note that principal *P* can easily verify the freshness of these time-stamped messages using the freshness axiom of our logic (axiom A21) in a manner similar to that illustrated by [25]. The following access authorization protocol will be applied to these messages:

Step 1. Verify the signing keys K_{user_D1} and K_{user_D2}. We use techniques developed in [25] to authenticate *User_D1*, including reasoning about revocation. We give a brief outline of the steps involved and refer to their work for details.

We have the initial assumption that K_{CA1} is *CA1*'s public key and that *CA1* has jurisdiction over *User_D1'*s key K_{user_D1} (Statement 6). Using the originator identification axiom A10 we can deduce that *CA1* generated the certificate sent in message 1-1, that is,

12. $P\ believes_{t2}\ CA1\ said_{t1,P} \lfloor CA1\ says_{tca1} \overset{K_{User\_D1}}{\Rightarrow}_{[tb,te],CA1} User\_D1 \rfloor_{K_{CA1}}^{-1}$

From *CA1*'s jurisdiction over the time that *CA1* sends any message (Statement 7) we can deduce that,

13. $P\ believes_{t2}\ CA1\ controls_{[t*,t],P}\ CA1\ says_{tca1} \overset{K_{User\_D1}}{\Rightarrow}_{[tb,te],CA1} User\_D1$, where [*t*, t*] denotes the particular time interval in question.

Applying jurisdiction axiom A22 we can deduce that,

14. $P\ believes_{t2}\ CA1\ says_{tca1} \overset{K_{User\_D1}}{\Rightarrow}_{[tb,te],CA1} User\_D1$

From our initial assumption of *CA1*'s jurisdiction over *User_D1'*s (Statement 6) public key we can deduce that,

15. $P\ believes_{t2}\ CA1\ controls_{tca1} \overset{K_{User\_D1}}{\Rightarrow}_{[tb,te],CA1} User\_D1$

and therefore by applying the jurisdiction axiom we obtain,

16. $P\ believes_{t2}\ (\overset{K_{User\_D1}}{\Rightarrow}_{[tb,te],CA1} User\_D1)$

Similarly, for *User_D2* we can deduce from message 1-2 that,

17. $P\ believes_{t3}\ (\overset{K_{User\_D2}}{\Rightarrow}_{[tb,te],CA2} User\_D2)$

Step 2. Establish group membership.

We apply the originator identification axiom A10 to message 1-3 and deduce that,

18. $P\ believes_{t4}\ AA\ said_{t2,P} \lfloor AA\ says_{taa}\ CP_{2,3} \Rightarrow_{[tb',te'],AA} G\_write \rfloor_{K_{AA}}^{-1}$

From *AA*'s jurisdiction over the time that *AA* sends any message (Statement 5) we can deduce that,

19. $P\ believes_{t4}\ AA\ controls_{[t*,t],P}\ AA\ says_{taa}\ CP_{2,3} \Rightarrow_{[tb',te'],AA} G\_write$, where [*t*, t*] denotes the particular time interval in question.

We apply the jurisdiction axiom A23 on 19.

20. $P\ believes_{t4}\ (AA\ says_{taa}\ CP_{2,3} \Rightarrow_{[tb',te'],AA} G\_write\ at_P \langle t*, t_2 \rangle)$

To remove $at_P \langle t*, t_2 \rangle$, we apply the reduction axiom A9,

21. $P\ believes_{t4}\ (AA\ says_{taa}\ CP_{2,3} \Rightarrow_{[tb',te'],AA} G\_write)$

We apply the access control group membership axiom A25 on 21 and 4 to obtain,

22. $P\ believes_{t4}\ CP_{2,3} \Rightarrow_{[tb',te'],AA} G\_write$

Step 3. Verify signed Request. By applying the originator identification axiom and jurisdiction axiom on message 1-4 we get,

23. $P\ believes_{t5}\ User\_D1\ says_{tu1} \lfloor User\_D1\ says_{tu1}$ "write" $O \rfloor K_{user\_D1}^{-1}$

24. $P\ believes_{t_5}\ User\_D2\ says_{tu2} \lfloor User\_D2\ says_{tu2}$ "write" $O \rfloor K_{user\_D2}^{-1}$
Applying access control axiom A38 on 22, 23 and 24,
25. $P\ believes_{t_6}$ ($G\_write\ says_{t_6}$ "write" O)

Step 4. Verify ACL. If $t_{b'} \leq t_1$, $t_6 \leq t_{e'}$ and (G_write, "write" O) $\in$ ACL_O, access is approved.

Reasoning about revocation

It is essential to verify the most recent available revocation information before granting access to an object. We can use our logic to illustrate how one can reason about revocation of the threshold attribute certificate in the example discussed above. (Revocation of identity certificates is discussed in [24]). Let *RA* be a revocation authority that is authorized to provide revocation information on behalf of *AA*. At time t_7 *P* receives the following revocation message from *RA*:

Message 2 $RA \rightarrow_{t7} P : \lfloor RA\ says_{tRA} \neg CP_{2,3} \Rightarrow_{t',RA} G\_write \rfloor_{K_{RA}}^{-1}$

Applying steps 11-14 above on message 2, we can deduce the following at time t_8
26. $P\ believes_{t_8}$ ($\neg CP_{2,3} \Rightarrow_{t',RA} G\_write\ at_P\ t^*$)
where $t^* \geq t_P$. If we interpret message 1 subject to a "believe until revoked" condition regarding the goodness of $CP_{2,3}$'s threshold attribute certificate, then, we will be unable to obtain this belief for $t_4 \geq t_8$.