Principles of Policy in Secure Groups

Hugh Harney Andrea Colgrove
SPARTA, Inc.
Columbia, MD
hh{acc}@sparta.com

Patrick McDaniel[*]
EECS Dept.
University of Michigan, Ann Arbor
pdmcdan@eecs.umich.edu

Abstract

Security policy is increasingly being used as a vehicle for specifying complex entity relationships. When used to define group security, policy must be extended to state the entirety of the security context. For this reason, the policy requirements of secure groups are more complex than found in traditional peer communication; group policies convey information about associations greater and more abstract than their pair-wise counterparts. This paper identifies and illustrates universal requirements of secure group policy and reasons about the adherence of the Group Security Association Key Management Protocol (GSAKMP) to these principles.

1. Introduction

The use of widely distributed resources on the Internet has strained existing network infrastructures. Until recently, applications and services were targeted to environments spanning few administrative domains supporting a relatively static user community. However, the explosion of new forms of communication has invalidated many of the basic assumptions upon which these systems were built. Thus, the design of these systems, and of their security in particular, has recently come under considerable scrutiny.

An approach addressing the requirements of these emerging applications and services is the use of policy. Through policy, a system may address the (sometimes conflicting) needs of all communication participants in real time. Each session occurs within the context of a shared policy defining the acceptable behavior and requirements of its participants. Thus, rather than relying solely on the system designers or network administration to define service behavior, the interests of all parities are considered at the point at which communication occurs. This paper considers a number of principles for the construction of one kind of policy, *secure group communication policy*.

We define a group security policy as a statement of the entirety of security relevant parameters and facilities used to implement the group. This best fits the viewpoint of policy as defining how security directs group behavior, who are the entities allowed to participate, and which mechanisms will be used to achieve mission critical goals. Note that this definition is not restricted to electronically distributed statements, policy is often the result of system design and configuration.

This paper considers the definition and requirements of security policies in groups. A number of principles for the design of policy services in group communication are identified. These principles are the result of a systematic analysis of the policy requirements unique to secure groups and those common to both peer and group communication. We reason about the security of arbitrary policy through the application of known principles and the reduction of group behavior to pair-wise operations.

We explicitly do not attempt to propose an approach for the formal analysis of group systems. We do, however, seek to identify universal requirements of policy management in secure groups. In developing these requirements, we investigate where known and accepted principles of (peer oriented) secure system construction and analysis are applicable to groups.

Secure groups protect content through the application of cryptographic methods on shared secrets. There are two predominant approaches used to establish and main-

[*] This work is supported in part under the National Science Foundation Grant #ATM-9873025 and by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-2-0508. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.

tain these secrets; collaborative group management [28] and authorized download of group data [17, 15, 22, 21].

Collaborative groups build trust through the inclusion of all group members in security relevant actions. Because each action requires the participation of the membership, management costs grow with group size. Hence, collaborative groups are appropriate for small groups with moderate to high communication and computational resources.

Conversely, authorized download groups relegate security relevant functions to trusted group entities. Trusted entities enforce the group policy and distribute session keys. However, authorized download groups are limited by trust; each member must trust the stated authorities or defer participation. Because of their relative low cost, these groups are appropriate for larger or more dynamic groups.

Much of the work relevant to security policy in group communication systems falls into two categories. In trust management systems [3, 4, 7], policy is specified and evaluated within a well-defined and provably correct framework. However, enforcement of policy is largely outside the scope of these systems. Conversely, policy directed secure group communication systems [17, 16, 10, 21] are chiefly designed to allow the definition and implementation of policies for security mechanisms (e.g., parameters and algorithms for session keying). While these latter systems provide correct enforcement of specific policies, it is not immediately clear that the distribution and composition of these policies always results in a secure group.

We attempt to reconcile existing policy approaches through the study of the requirements of groups; systems require the correct specification, distribution, evaluation, and enforcement of policy to be secure. Incomplete specification or incorrect implementation of any one of these processes can lead to an insecure solution.

The principals described in the following section were used to guide the design of the Group Security Association Key Management Protocol (GSAKMP) architecture [15, 14]. Summarized in the latter sections, the analysis of GSAKMP seeks to show that GSAKMP groups not only enforce a given policy, but do so consistently and securely.

GSAKMP defines an architecture and protocol used to implement secure multicast groups. Growing out of the GKMP [17, 16] protocol, GSAKMP focuses on the implementation of secure groups through the definition, evaluation, and enforcement of group policy. The physical manifestation of a group policy in GSAKMP is a *policy token*.

A GSAKMP policy token is a highly flexible data structure used to define the behavior of a group. The token defines an exhaustive list of policies (there are over 150 different fields supporting a wide range of policies and mechanisms). With small exception, the group is free to define policies containing as many (or as few) of these fields as is desirable. Thus, groups with varying abilities and requirements can be defined through the policy token.

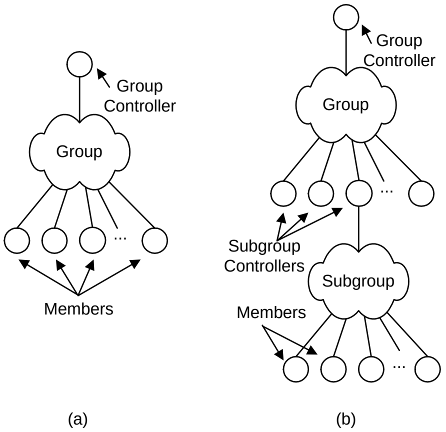

Figure 1. Secure group - A group is a collection of cooperating entities operating under a shared security policy. Groups may be organized into a flat structure (a), or as a collection of distinct subgroups (b).

The remainder of this paper is organized as follows. Section 2 develops a definition and set of universal principles for the specification and subsequent use of policy in secure groups. Section 3 reasons about the compliance of the GSAKMP with these policy principles. Section 4 reviews several works relevant to the definition and implementation group security policy. We conclude in Section 5.

2. Group Security Policy

This section considers the requirements of policy management in secure group communication systems. A set of principles derived from these requirements is developed and illustrated. We begin in the next subsection by stating a definition of secure groups and their policies.

2.1. Secure Groups and Policy

Described in Figure 1, we define a secure group as the collection of cooperating entities operating under a shared security policy. Each group contains a *group controller* from which keying material logically emanates. Groups can be organized into logical subgroups, with distinct entities serving as *subgroup-controllers* (for example, as defined in Iolus [22]). Group members may join, leave, or become compromised at any time during the session. In particular, we define the participants of a group as follows:

- **Group Owner** (GO): Also known as the *policy issuer*, the GO specifies the group security policy.

Action	Description
policy creation	create/assert a group policy
policy modification	modify the group policy, policy modification occurs after it is detected that a current policy in inappropriate or unimplementable
grant rights	grant rights to members/entities external to the policy
key creation	create a session key, or to generate rekeying material
group destruction	disband the group
key dissemination	distribute session keys and supporting keying material
rekey action initiation	initiate the group rekey process. This is often used to eject failed or compromised members.
authorize member	authorize/state authenticity of group member
admit member	admit a member to the group
eject member	remove a member from the group
audit group	monitor access control messages or membership information
key access	gain access to the session key

Table 1. Security relevant group actions - group is maintained through the application of security relevant actions. An access control policy will map these actions to authorized parties.

The policy is specified in accordance with the expected content value and operating environment. The issuing authority is trusted by all potential group members to state an appropriate policy.

- **Group Controller** (GC): The GC acts as a key dissemination and access control authority. The GC enforces group access control policy by creating and distributing group keying material to authorized entities, and initiating rekeying and member ejection as events dictate.
- **Subordinate Group Controller** (SGC): A subordinate controller performs all group controller functions, with the exception of session key creation.
- **Member** (M): The group member is the consumer of the group keying material. The member verifies the policy as correct, and enforces the authorization policies as defined by the policy specification (i.e., by only accepting appropriately authorized group messages).

Throughout, we assume that members and controllers are mutually trusted, i.e., entities receiving a policy accept and enforce (the authenticated) policy as directed by its specification. However, members who become compromised may diverge from the specification arbitrarily. We assert entities external to the group and compromised members may intercept messages, modify messages, or prevent messages from being delivered.

We note many other, more complicated, models of group exist. For example, groups can converge on a single policy through negotiation [10] or assume members are untrusted [24]. For brevity, we defer discussion of these groups. However, many of the principles identified in the following section are applicable to these groups.

Each policy is initially stated as sets of conditional statements defining the possible authorizations and mechanisms used to implement a group. The conditional statements indicate environment-specific constraints and requirements of potential sessions. The group owner creates the initial policy. An instantiation of the policy[1] results from the leader evaluation of the conditional policy statements. The instantiation defines the security relevant properties of the group. However, some aspects of the group policy are implicitly defined; decisions about how the group security is implemented can be the result of system design and configuration. Whether explicitly or implicitly defined, we assert that any group must specify the following:

- **Identification** - Each participant and group must be unambiguously identified. Failure to correctly identify the group policies, messages, and participants can lead to incorrect and insecure operation.
- **Authorization** - A group policy must identify the entities allowed to perform protected actions. Group authorization partially determines the trust embodied by the group.
- **Access Control** - Allowable access to group action must be stated by policy. An access control policy defines a mapping between the authorized parties and secure actions in the group, and indirectly, the per-

[1] Throughout, where unambiguous, we will refer to an instantiation of a policy as the *group policy*.

missions for group information. We present the set of group security relevant actions in Table 1.

- **Mechanism** - Each policy must state how the security requirements of the group are to be addressed. This includes the identification of the approaches used to achieve security guarantees and the parameters of their operation. Thus, a mechanism policy defines the provisioning of group software and often the operation of its component protocols.
- **Verification** - Each policy must present evidence of its validity. The means by which the origin, integrity, and freshness of the policy is asserted (for example, via digital signature) must be known by each group member prior to its acquisition.

2.2. Principles of Group Policy

The direct application of policy approaches used in peer-communication is unlikely to meet the needs of groups. This is due in large part to fundamental differences between peer and group policies; group policy conveys information about an association greater and more abstract than its pair-wise counterpart. The following text identifies and illustrates universal principles resulting from our analysis of group and peer communication policies.

Principle 1: Enforcement of group policy must be consistent across a group

While it may evolve over the course of a session, the group requires a singular policy definition. This implies a shared view of the participants and the security of application content. Failure to operate under the same security context can lead to vulnerable or incompatible solutions.

Similarly, policy implicitly requires trust among the membership. Each member trusts that all participants have been admitted and enforce the policy specification correctly. If a consistent view of policy cannot be established, members will have no way to infer this trust. We have identified two facets of policy consistency: *mechanism equivalence* and *synchronization*.

Two mechanisms are *equivalent* if $a)$ they implement the same service (e.g., data confidentiality), and $b)$ the security of the mechanisms is not qualitatively different. For example, Figure 2 describes a group implementing a confidentiality policy. Subgroup a (in the figure) implements confidentiality using a strong data encryption algorithm. Furthermore, a cryptographic gateway gw coexists in both Subgroup a and a second Subgroup b. Subgroup b contains mobile devices with limited computing resources. The gw translates all communication between the strong algorithm implemented by a to a weaker algorithm implemented by the mobile devices in b.

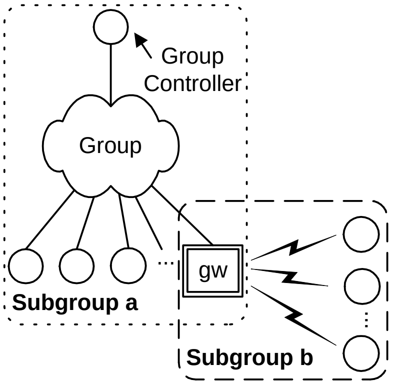

Figure 2. Mechanism equivalence - Policies implemented by members of Subgroup a must be equivalent to those implemented by Subgroup b. Failure to implement equivalent policies may result in undetected vulnerabilities.

Clearly, an adversary attempting to uncover group content will mount an attack against data transmitted under the weaker algorithm. Thus, for this group, the confidentiality is only as strong as provided by the weaker algorithm. Because the algorithms are not equivalent, the security of the group as a whole is weakened. Worse yet, members of Subgroup a may be unaware of the use of the weak algorithm.

Session rekeying in peer communication is well defined. The peer end-points actively participate in an exchange resulting in acceptance of the new key. Because both participants assert acceptance, subsequent use of the key can be unambiguous. The issues, design, and vulnerabilities of peer key exchanges have been thoroughly researched and are well understood.

Session rekeying in group communication is inherently more difficult. As defined by the group threat model, rekeying is triggered by security relevant events. Rekeying is often initiated, for example, when a session key lifetime is reached, following member joins and leaves, and to complete recovery from the compromise of a group member [19]. However, knowledge of these events is not often universally available.

Rekeying of the group is required to be *synchronized*. An arbitrary number of end-points must reach agreement not only on the new secret key, but synchronize its subsequent use. For example, consider a group which has recently distributed a new session key. A member receiving a message encrypted under an old session key is faced with a dilemma; in the absence of synchronized delivery, the message may represent $a)$ delayed delivery of a correct message encrypted under the old session key, or b) a message generated by an adversary who has gained access to the previous key.

Rekeying, and the synchronization of policy in general, are instances of *distributed consensus*. Agreement on the new session key or policy is reached through via group protocols. However, in the general case, distributed consensus algorithms are both complex and expensive [12, 23]. Many existing group systems attempt to avoid these costs by relaxing synchronization requirements.

One way to relax key synchronization requirements is to allow several session keys to be simultaneously valid. For example, suspending transmission of data during rekeying in a video-conference is highly undesirable. Thus, a group may wish to continue to use (accept packets encrypted under an old session key) previous keys until consensus on the new key has been reached. A policy supporting this environment should dictate the amount of time an old session key may be used (and by direct corollary the minimal freshness of received messages). This approach and other relaxations to the general case of this *key transition* problem are considered in [21].

Policy is also required to be synchronized. At any point during the session, the policy a group member enforces must be identical to the one intended by the controller. If the enforced policy is not both fresh and correct, then the member may diverge from the session specification arbitrarily. Obviously, the divergent member may introduce any number of vulnerabilities.

Synchronization directly requires all received policies be fresh, authentic, and unmodified. The means by which policy freshness is assessed must conform to some *a priori* policy. For example, group members could verify that a policy revision number increases monotonically. The group member would never accept a policy update with an unexpected revision number.

The group members must be able to verify that the policy has not been modified during dissemination (e.g., integrity of received policies is preserved). This reduces to a requirement stating each member must be able to verify that a policy originated from the group controller.

Of course, any policy must contain some evidence of its authenticity (i.e., policy verification). For example, a keyed message authentication code (HMAC) [18] or digital signature [9] can be used to assert the authenticity of a policy.

Principle 2: Only authorized entities can affect the security posture of the group

Authorization in peer communication can easily be inferred from knowledge of the session key. Because there are only two entities participating in the communication, the correctness of any action can be directly assessed. Conversely, because of the many roles which members may perform, group communication requires a more complex authorization model.

Groups commonly designate one or more entities to act as authorities within the group. For example, an entity wishing to join the group will communicate with an entity authorized to admit members. Once the admittance authority verifies that the potential group member possesses the appropriate credentials, it allows the member into group. However, unless otherwise specified by policy, the admitted member should not have the authority to admit other members. The new group resulting from the admittance of the member represents a new security context; there is a new group member trusted with the group key.

As identified in Table 1, there are many actions that affect the group security context. Because each of these actions can affect the security of all group members, they must be associated with the set of entities that are authorized to perform them. A group allowing an adversary to perform security relevant actions would be, among others, vulnerable in:

- Policy creation: The unauthorized entity can modify policy in arbitrary ways. Thus, the group may be manipulated into operating in an insecure way.
- Key dissemination: An unauthorized group controller can create a false group.
- Initiate rekey: An unauthorized entity performs a denial of service attack in which the group would continually rekey. The group would expend considerable resources performing key management functions.
- Group destruction: If an unauthorized group destruction command is accepted, the group will disband prematurely. Clearly, this represents a serious denial of service attack.

Principle 3: Group content must be protected Generally, data security mechanisms provide a means by which content confidentiality, authenticity, and integrity can be protected. These mechanisms implement protection through the application of cryptographic algorithms on session keys. Thus, the security of the group is predicated on the security of the processes restricting access to session keys.

As stated indirectly by principle 2, access to session keys must be restricted to entities with authority to receive them. The access control policy must be defined as part of the larger group policy. Similarly, the means by which the potential member meets the criteria must be specified.

Consider an example group policy stating that a member must prove possession of company X credentials (in the form of a certificate) before being admitted to the group. Thus, an example access control policy states 1) the entity must possess the private key of a certificate, 2) the certificate must state that the organization of the entity is the desired company, and 3) the certificate must be issued from the company's certificate authority.

An admittance authority enforces the access control policy (on a signed join request containing the certificate) by verifying the certificate organization and issuer fields, validating the signature, and checking the certificate has not been revoked (e.g., through an appropriate certificate revocation list). If this process is successful, the member receives and subsequently uses the session key to communicate with the group.

Because the group policy is enforced correctly, and the underlying cryptographic algorithms are secure[2], group content protection is ensured. However, if any of these authorization, access control, or data security policies is incorrectly enforced, then the security of the group as a whole may be lost. This demonstrates the fragility of security; incorrect implementation of any one function can invalidate guarantees provided by others.

Principle 4: Groups must be capable of recovery from security relevant failures to a secure state

It is necessary for groups to recover to a secure operating state when a subset of its membership is found to be untrustworthy. Thus, a policy must state the way in which compromise is to be detected and, if available, the mechanisms used for recovery.

There are a myriad of ways groups may recover from member compromise. Early systems, in an effort to restrict insecure access to content, disbanded immediately following compromise [17, 16]. More recently, group systems employ sophisticated rekeying approaches [31, 32, 21] to recover from member compromise. In these latter systems, compromised members are ejected by their exclusion from the subsequent rekey process.

A group may also require recovery from non-compromise failures. The effect of network partitions [11], process crashes [20], and other failures on the group security context is an open area of research. We note that the mechanisms used for failure detection and recovery will have unique security requirements. For example, the heartbeat-based failure detection mechanism in [20] requires heartbeats be authentic. In the absence of authentic failure detection, an adversary may be able to mask the failure of group members through forged heartbeats.

3. Policy Specification and Enforcement in GSAKMP

This section describes the Group Secure Association Key Management Protocol and presents arguments illustrating its compliance with the principles presented in the preceding section.

3.1. Description

The Group Secure Association Key Management Protocol (GSAKMP) dictates and manages the security of a communications group. GSAKMP manages group security throughout the life-cycle of the group: group initiation, maintenance, compromise recovery, and deletion. Policy is defined in the *policy token* data structure, which is distributed and enforced over specified protocol exchanges. Group content is protected by appropriate security mechanisms and their associated session keys that are known only the current membership.

In GSAKMP, group responsibilities are decomposed into authorized roles. Roles are defined for Group Owner, Group Controller, Subgroup Controller, and Member. The rights attributed to these roles are presented in Table 2. The authorization criteria, as well as the mechanisms used to verify authorizations, are defined in the policy token. In one instantiation of GSAKMP, security relevant messages must be signed by the authorized entity.

Signed by the Group Owner, the policy token contains authorization, key management, and data protection rules for the group, and defines the mechanisms used for verification. Using the defined mechanisms, potential group members receiving a token verify the token is authentic, fresh, and unmodified. Once verified, the token is used to direct the behavior of the member.

Groups are formed by multiple individual (peer) admittances. In the GSAKMP join exchange, the Group Controller presents the group's token and verifies the potential member's credentials. If the member credentials are accepted, the current group keys are securely downloaded to the newly accepted member. The member, in turn, inspects the token to determine if the group policy is appropriate for the information that they wish to share.

Compromise recovery methods such as the Logical Key Hierarchy (LKH) [31, 32] allow GSAKMP to securely and rapidly eject compromised members. Because of the properties of these keying techniques, any keys possessed by compromised members will not be valid after ejection.

The GSAKMP specifications are quickly reaching maturity. Two IETF drafts have been submitted [15, 14] and a license free reference architecture and associated documentation are available at:

`ftp://ftp.sparta.com/pub/columbia/gsakmp`

[2] There is significant debate on the correct design of secure group data transforms. For the purposes of this discussion, we assume that all mechanisms are fundamentally secure; the cryptographic algorithms and data transforms are sound.

Action	Rights
policy creation	GO
policy modification	GO
grant rights	GO
key creation	GC
group destruction	GC
key dissemination	GC,SGC
rekey action initiation	GC,SGC
authorize member	GC,SGC
admit member	GC,SGC
eject member	GC,SGC
audit group	GC,SGC
key access	GC,SCG,M

Table 2. GSAKMP rights assignment - assignment of rights to group owners (GO), group controllers (GC), subgroup-controllers (SGC), and members (M) within a GSAKMP group. A definition of these rights is presented in Table 1.

3.2. Analysis

In this section, we informally argue that the framework provided by GSAKMP adheres to the group policy principles discussed in the previous section.

GSAKMP specifies a group security policy through the policy token. The token is passed to each group member as part of the join exchange. Each group member verifies the policy token and enforces the stated policy. As events dictate, the group policy token can be updated over the life of the group. Each received update is verified in the same way as the initial token.

To ensure the current policy token is correctly disseminated across the group, GSAKMP sends the current token along with each security relevant action. Each member checks the token's freshness indicators and verifies its origin and integrity.

Principle 1: Enforcement of group policy must be consistent across a group

a) *GSAKMP enforces the use of equivalent mechanisms*

Allowable cryptographic and key management techniques are specified in the policy token.

b) *GSAKMP provides methods for key and policy synchronization.*

Based on the authorizations defined in the policy token, each joining member is given the keys needed to fulfill their assigned roles. Because the group is formed by multiple individual joins, each member will initially have the keys needed for authorized participation. Via LKH, keying due to stale session keys or compromise recovery are accomplished using a single rekey message. Similarly, if the policy changes such that a current member is no longer allowed to participate, the group must be rekeyed. In this case, rekeying must be handled in the same way as in compromise recovery.

Similarly, any member joining the group receives the most current policy token. Subordinate group controllers not providing the correct (fresh) policy token are ejected from the group forcing a rekey. The ejected subordinate group controller would be incapable of providing a valid key to the new member. Hence, freshness of the initial token may be verified by the ability to access the group by the new member. The GSAKMP Policy Token contains freshness dates and sequence numbers. Any subsequent token will only be accepted if it contains later freshness indicators. Thus, stale policies are detectable. As reasoned before, as the group consists of individual member joins, the group only will accept the most current token. Thus, the group policy is synchronized.

c) *From 1 and 2, enforcement of group policy is consistent across the group.*

The GSAKMP policy token is issued and signed by an entity responsible for the group security policy. It further defines who may perform security relevant actions. Both the rules for determining permissions and the mechanism used to verify the rules themselves are defined in the policy token.

Upon receipt of a security relevant message, group members review the current policy token. The message is initially verified using a mechanism defined in the token. Using the roles and authorizations defined in the token, the member checks the permissions of the sending party. If the verification succeeds and the sender can assume a role allowed to perform the action implied by the message, then the message is accepted.

Principle 2: Only authorized entities can affect the security posture of the group

a) *The policy token is verified as coming from a trusted source and as being authentic. (1-1)*

The signature payload contains the identity of the signer of the token. This signature is verified according to the parameters for verification of the token and the stated identity.

b) *Authorized entities are identified in the policy token. (GSAKMP specification)*

c) *Messages affecting the security posture of the group must be issued by an authorized entity. (GSAKMP specification)*

d) *Each member can verify that the message affecting the security posture was issued by a trusted source indicated by the verified policy token.*

The verification mechanism and parameters are indicated in the policy,

e) *Each member can verify that the message was freshly signed.*

GSAKMP join exchange messages contain nonces. The rekey message contains a timestamp. The group delete message is only generated once, so no replay potential exists.

f) *Each member can verify that they are intended recipients of the message.*

GSAKMP message header's contain group identifiers. Security relevant messages to individuals (e.g., Key Download) contain individual identifiers.

g) *Therefore, each member will only act on fresh messages intended for them from an authorized source.*

h) *Groups are comprised of individual members.*

i) *The group will only act on fresh messages intended for them from an authorized source.*

j) *Only authorized entities can affect the security posture of the group.*

GSAKMP protects the group by controlling access to group keys. The access control rules and compliance mechanisms are defined in the policy token.

Initially, the policy issuance authority (Group Owner) passes the token to the group controller (GC). The GC uses the token's access control rules to restrict access to the group key. In all instances, there will be a member of the group who is authorized to disseminate keys and perform access control.

The group member presents required credentials to the GC. Using these credentials as proof, the GC ensures that the access control criteria is met prior to release of the group key.

Equally important, the potential group member verifies the authority of GC or SGCs to make access control decisions. In verifying the authenticity of messages received from controllers, a group member is protected from accepting false keys and inadvertently revealing confidential group information.

Principle 3: Group information must be protected

a) *Cryptographic mechanisms specified in the GSAKMP token protect group data.*

b) *Access to the group is granted by issuing the group cryptographic key(s).*

c) *Keys can be obtained only through legitimate distribution or from compromise.*

Initial key distribution is secure, which is reasoned as follows. Keys are distributed in a pairwise manner. The pairwise join protocol is secure: 1) its messages are signed by authorized entities and subsequently verified; 2) the sender and intended recipient of the messages are explicit; 3) the messages freshness are indicated by nonces; and 4) the download of the keys are protected by a group-appropriate confidentiality mechanism as indicated by the GSAKMP policy token. The design of the initial exchange is based on well known principles of peer authentication and key distribution protocols [6, 1, 2].

Keys are distributed to authorized individuals only. The authorization rules are listed in the policy token. These rules are trusted (1-1). Because the pairwise distribution to each individual member is done securely and because groups are comprised of individual members, the initial key distribution to the group is secure.

The first rekey under LKH is sent encrypted in keys obtained through initial distribution, which is secure. Only authorized members received these keys through initial distribution. Subsequent rekeys are sent encrypted either in initial keys or in keys distributed through previous rekeys.

d) *If keys are obtained through compromise and that compromise is discovered, the group is rekeyed such that any compromised key is no longer valid.*

GSAKMP provides a framework for advanced recovery mechanisms; LKH is implemented in the reference software. GSAKMP distributes LKH key arrays during group establishment and uses the key tree to rekey following the detection of member compromise. Over the course of the session, each member receives an array of keys that correspond to a unique path in the key tree.

GSAKMP defines compromise recovery as a security relevant action. Thus, policy defines the rules and mechanisms needed to issue a recovery message. However, compromise detection and reporting is outside of the scope of the GSAKMP protocol.

Upon receipt of a compromise report, the GC creates an LKH recovery message identifying the current policy token. The GC instigates recovery by signing and transmitting the recovery message.

All group members verify the authenticity of the recovery message. Once the LKH message saturates the group[3], the compromise member is excluded from the new security association. Thus, a secure state of operation is recovered.

A member not receiving the recovery message will no longer have access to the group. In this case, the mem-

[3] For brevity, these arguments do not discuss faults, accidental or induced, related to unreliable nature of group communication.

ber will be forced to rejoin through the authenticated join exchange.

Principle 4: Groups must be capable of recovery from security failures to a secure state

a) *A security failure can occur when an adversary attempts to masquerade as an authorized individual.*

b) *Any failure in verification of the join exchange (signature, nonce, identification field, or inadequate credentials) are detectable by both parties. In this case, the join exchange is aborted.*

c) *Counterfeit rekey messages are detected. Bad rekey message will fail contain incorrect signatures, timestamps, or fail to be authorized.*

d) *Counterfeit group delete messages are also detected. Bad group delete message will contain incorrect signatures or fail to be authorized.*

e) *Group access via key compromise is remedied via LKH as discussed previously; therefore, the group will rapidly be rekeyed causing all compromised keys to be invalid. This will result in denial of access to any entity possessing only the compromised keys.*

4. Related Work

The Internet Engineering Task Force (IETF), Policy Framework working group is chartered with the development of an architecture supporting the management of network devices through abstract policies. In the policy architecture [29], the desired behavior of each device is stated through sets of *policy rules*. A policy rule is a conditional statement identifying a set of (possibly abstract) actions that are to be executed in environments when/where the conditions are satisfied. However, the security of the distribution and enforcement of these rules has yet to be addressed.

The Security Policy System [26] is an architecture supporting flexible definition and distribution of security policies for IPSec security associations (SA)s. The central specification documents include; an architecture overview [26], a policy specification language [8], and a policy distribution protocol definition [27]. To simplify, clients of the policy system query the policy database for connection policies. Connections are accepted only if they meet the requirements of policies obtained from a policy database in the server domain. The mechanisms used to secure communication are defined in the connection policies.

Developed by Branstad et. al., the Dynamic Cryptographic Context Management (DCCM) system [10] is used to define and enforce security policies within very large groups (100,000+ members). A principle contribution of DCCM is its use of policy as entirely defining the context in which a group operates. Policy may be negotiated or stated by an initiating member, and flexible mechanisms for policy representation and interpretation are defined.

The Antigone framework [21] provides flexible interfaces for the definition and implementation of a wide range of secure group policies. A central element of the Antigone architecture is a set of mechanisms providing the basic services needed for secure groups. Policies are implemented by the composition and configuration of these mechanisms. Thus, Antigone does not dictate the available security policies to an application, but provides high-level mechanisms for implementing them.

Many recent advances in the use of policy in distributed systems have occurred in the realm of authorization and access control. Introduced in [3] by Blaze et. al., *trust management* provides a unified approach for the specification and evaluation of security policies. Trust management focuses centrally on development of access control policy through the specification of evaluation of trust relationships. At the core of any trust management system is a domain independent language used to specify the capabilities, policies, and relationships between entities. Each application subscribing to a trust management service provides policy specifications to a central component called the trust management engine. An application consults the engine for access control decisions at run-time. The engine evaluates the access control request using the policy specification and environmental data. Therefore, applications need not evaluate access control policies directly, but defer analysis to the trust management engine.

Through rigorous analysis, the PolicyMaker [3] trust management engine has been proven to be correct. Thus, with respect to the policy specification, any application using PolicyMaker is guaranteed to evaluate each policy decision correctly. However, policy enforcement is left to the application. Recent systems (e.g KeyNote [4] and REFEREE [7]) have extended the trust management approach by simplifying application interfaces and introducing a limited set of enforcement facilities. Other approaches addressing access control policy in distributed environments (e.g., Akenti [30], GAA API [25]) approach the management of trust in similar ways, but with somewhat different goals, requirements, and architectures.

The Internet Engineering Research Force (IRTF), Secure Multicast Research Group (SMuG) is researching a suite of standards and associated reference architecture [13] upon which secure multicast applications can be built. In addition to the specification of cryptographic transforms [5] (i.e., content security) and key management protocols [14], this work will define a policy management infrastructure appropriate for secure multicast. The policy team has recently published a taxonomy of group policies [19] and a requirements statement is forthcoming.

5. Conclusions

This paper considers the requirements of policy in secure group communication. We have identified a set of universal principals that can be used to guide the design of security policy in secure group communication systems. These principles not only address the pragmatic requirements for the correct and secure operation of standard services, but identify requirements for the secure distribution and synchronization of policy specifications.

In our investigation, we found that it is necessary, but not sufficient, to correctly specify policy. Once a policy is specified, it must be securely distributed, authenticated, and enforced. This illustrates the fragility of security in general; incorrect implementation of any one security function can invalidate guarantees provided by others.

Over the lifetime of the group, consensus surrounding the definition and interpretation of policy must be maintained by all participants. Failure to correctly synchronize a policy definition can lead to undetected vulnerabilities. If a policy is not uniformly interpreted by group members, the security of the group as a whole can be compromised.

The attribution of authority in groups is of paramount importance. Failure to correctly identify and authorize secure action can lead to any number of vulnerabilities. Thus, the protection of the group content is largely defined by the verification and enforcement of rules defining authorization and access control.

Finally, we note that member compromise is an inevitability for many applications. Thus, a resilient group must be able to recover from compromise to a secure state. This often requires the ejection of compromised members in a way that does significantly affect the performance or security of the group as a whole.

We have evaluated the Group Security Association Key Management Protocol under the identified principles. We have investigated where known and accepted principles of (peer oriented) secure system construction and analysis are applicable to groups. Furthermore, we have demonstrated GSAKMP compliance with these principles.

In the end, policy supported communication is only going to be as successful as the abilities of entities specifying policy permit. We, as researchers and engineers, can develop rigorous frameworks in which policy can be specified, distributed, and enforced. However, preventing the specification of bad policies is a much harder problem. Environmental issues are likely to determine the correctness of a given policy. Thus, education of the user community will play a large role in the correct operation of the supported applications.

References

[1] M. Abadi and R. Needham. Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.

[2] R. Anderson and R. Needham. Robustness Principles for Public Key Protocols. In *Lecture Notes in Computer Science, Don Coppersmith (Ed.), Advances in Cryptology – CRYPTO '95*, volume 963, pages 236–247. Springer-Verlag, 1995.

[3] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173, November 1996. Los Alamitos.

[4] M. Blaze, J. Feignbaum, J. Ioannidis, and A. Keromytis. The KeyNote Trust Management System - Version 2. *Internet Engineering Task Force*, September 1999. RFC 2704.

[5] R. Canetti, P. Rohatgi, and P.-C. Cheng. Multicast Data Security Transformations: Requirements, Considerations, and Proposed Design. *Internet Engineering Task Force*, June 2000. `draft-irtf-smug-data-transforms-00.txt`.

[6] U. Carlsen. Cryptographic Protocol Flaws: Know Your Enemy. In *Proceedings of 7th IEEE Computer Security Foundations Workshop*, pages 192–200. IEEE, 1994.

[7] Y. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss. REFEREE: Trust Management for Web Applications. In *Proceedings of Financial Cryptography '98*, volume 1465, pages 254–274, Anguilla, British West Indies, February 1998. Springer-Verlag.

[8] M. Condell, C. Lynn, and J. Zao. Security Policy System Language (*Draft*). *Internet Engineering Task Force*, July 1999. `draft-ietf-ipsec-spsl-01.txt`.

[9] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[10] P. Dinsmore, D. Balenson, M. Heyman, P. Kruus, C. Scace, and A. Sherman. Policy-Based Security Management for Large Dynamic Groups: A Overview of the DCCM Project. In *Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX '00)*, pages 64–73. DARPA, January 2000. Hilton Head, S.C.

[11] D. Dolev and D. Malki. The Transis Approach to High Availibility Cluster Communication. *Communications of the ACM*, 39(4), April 1996.

[12] M. Fischer, N. Lynch, and M. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM*, 32(2):374–382, 1985.

[13] T. Hardjono, R. Canetti, M. Baugher, and m P. Dinsmore. Secure Multicast: Problem Areas, Framework, and Building Blocks (*Draft*). *Internet Engineering Task Force*, October 1999. `draft-irtf-smug-framework-00.txt`.

[14] H. Harney, M. Baugher, and T. Hardjono. GKM Bukld-ing Block: Group Security Association (GSA) Definition (*Draft*). *Internet Engineering Task Force*, February 2000. `draft-irtf-smug-gkmbbb-gsadef-00.txt`.

[15] H. Harney, A. Colegrove, E. Harder, U. Meth, and R. Fleischer. Group Secure Association Key Management Protocol (*Draft*). *Internet Engineering Task Force*, April 2000. `draft-harney-sparta-gsakmp-sec-01.txt`.

[16] H. Harney and C. Muckenhirn. Group Key Management Protocol (GKMP) Architecture. *Internet Engineering Task Force*, July 1997. RFC 2094.

[17] H. Harney and C. Muckenhirn. Group Key Management Protocol (GKMP) Specification. *Internet Engineering Task Force*, July 1997. RFC 2093.

[18] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. *Internet Engineering Task Force*, April 1997. RFC 2104.

[19] P. McDaniel, H. Harney, P. Dinsmore, and A. Prakash. Multicast Security Policy (*Draft*). Internet Research Task Force, Secure Mutlicast Research Group (SMuG), June 2000. `draft-irtf-smug-mcast-policy-00.txt`.

[20] P. McDaniel and A. Prakash. Lightweight Failure Detection in Secure Group Communication. Technical Report CSE-TR-428-00, Electrical Engineering and Computer Science, University of Michigan, June 2000.

[21] P. McDaniel, A. Prakash, and P. Honeyman. Antigone: A Flexible Framework for Secure Group Communication. In *Proceedings of the 8th USENIX Security Symposium*, pages 99–114, August 1999.

[22] S. Mittra. Iolus: A Framework for Scalable Secure Multicasting. In *Proceedings of ACM SIGCOMM '97*, pages 277–278. ACM, September 1997.

[23] S. Mullender. *Distributed Systems*. Addison-Wesley, First edition, 1993.

[24] M. Reiter. Secure Agreement Protocols: Reliable and Atomic Group Multicast in Rampart. In *Proceedings of 2nd ACM Conference on Computer and Communications Security*, pages 68–80. ACM, November 1994.

[25] T. Ryutov and C. Neuman. Representation and Evaluation of Security Policies for Distributed System Services. In *Proceedings of DARPA Information Survuvability Conference and Exposition*, pages 172–183, Hilton Head, South Carolina, January 2000. DARPA.

[26] L. Sanchez and M. Condell. Security Policy System (*Draft*). *Internet Engineering Task Force*, November 1998. `draft-ietf-ipsec-sps.txt`.

[27] L. Sanchez and M. Condell. Security Policy Protocol (*Draft*). *Internet Engineering Task Force*, July 1999. `draft-ietf-ipsec-spp-00.txt`.

[28] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A New Approach to Group Key Agreement. In *International Conference on Distributed Computing Systems (ICDCS'98)*. IEEE, May 1998.

[29] M. Stevens, W. Weiss, H. Mahon, B. Moore, J. Strassner, G. Waters, A. Westerinen, and J. Wheeler. Policy Framework (*Draft*). *Internet Engineering Task Force*, September 1999. (`draft-ietf-policy-framework-00.txt`).

[30] M. Thompson, W. Johnson, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari. Certificate-based Access Control for Widely Distributed Resources. In *Proceedings of 8th USENIX UNIX Security Symposium*, pages 215–227. USENIX Association, August 1999. Washington D. C.

[31] D. M. Wallner, E. J. Harder, and R. C. Agee. Key Management for Multicast: Issues and Architectures (*Draft*). *Internet Engineering Task Force*, September 1998. `draft-wallner-key-arch-01.txt`.

[32] C. K. Wong, M. Gouda, and S. S. Lam. Secure Group Communication Using Key Graphs. In *Proceedings of ACM SIGCOMM '98*, pages 68–79. ACM, September 1998.