Bootstrapping Secure Multicast using Kerberized Multimedia Internet Keying

by

Jeffrey Lok Tin Woo

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2012

Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

We address bootstrapping secure multicast in enterprise and public-safety settings. Our work is motivated by the fact that secure multicast has important applications in such settings, and that the application setting significantly influences the design of security systems and protocols.

This document presents and analyzes two designs for the composition of the authentication protocol, Kerberos, and the key transport protocol, Multimedia Internet KEYing (MIKEY). The two designs are denoted to be KM_1 and KM_2. The main aspect in which the objective impacts the design is the assumption of an additional trusted third party (called a Key Server) that is the final arbiter on whether a principal is authorized to receive a key.

Secure composition can be a challenge, and therefore the designs were kept to be simple so they have intuitive appeal. Notwithstanding this, it was recognized that even simple, seemingly secure protocols can have flaws. Two main security properties of interest called safety and availability were articulated. A rigorous analysis of KM_1 and KM_2 was conducted using Protocol Composition Logic (PCL), a symbolic approach to analyzing security protocols, to show that the designs have those properties.

The value of the analysis is demonstrated by a possible weakness in KM_1 that was discovered which lead to the design of KM_2. A prototype of KM_1 and KM_2 was implemented starting with the publicly available reference implementation of Kerberos, and an open-source implementation of MIKEY. This document also discusses the experience from the implementation, and present empirical results which demonstrate the inherent trade-off between security and performance in the design of KM_1 and KM_2.

Acknowledgements

I want to express my deepest gratitude to my supervisors, Dr. Mahesh Tripunitara and Dr. Gordon Agnew. The creation of this thesis would not have been possible without Dr. Tripunitara's close guidance and insight in this research topic. His enthusiasm in research encouraged and motivated me to refine my knowledgebase and be creative in deriving solutions. Also without Dr. Gordon Agnew's introduction to the Cisco Graduate Internship Program, I may not have had the resources which allowed me to pursue graduate studies. It was truly an honour to have both opportunities bestowed upon me.

I would also like to thank all the members of the Cisco team that aided me throughout the internship. I am especially grateful for their support as it was an exceptionally difficult period for them. In particular, I would like to thank my manager, Matthias Loeser for his mentorship, my mentor, Fangjin Yang who never hesitated in providing me with helpful advice in various aspects of life and career, and of course Suran De Silva for organizing this program.

Lastly, I wish to thank all my friends for helping me through all the difficult times and acted as my family away from home. They provided the environment that filled me with entertainment, emotional support and care throughout my undergraduate and graduate studies. In alphabetical order by surname: Yuki Cheung, Kenneth Lam, Henry Pang, Alice Tsang, Ronald Wan, Abraham Wong, Lily Wong, Noreen Wong, Osman Wong, and Wallace Wu. Thank you all for everything.

Dedication

I would like to thank my dad, Sak Fai Woo, my mom, Lillian Woo, my brother, Tim Woo, and my other significant half, Connie Ling for their endless and altruistic support. Their love and care reached far from the other side of the world. I dedicate this thesis to them.

Table of Contents

List of Tables

List of Figures

Glossary

API	Application Programming Interface
GCKS	Group Controller and Key Server
GSA	Group Security Associations
IETF	Internet Engineering Task Force
IP	Internet Protocol
KAS	Kerberos Authenticaion Server
KDC	Key Distribution Centre. In Kerberos's context, this is both the KAS and TGS
Kerberos	A single sign-on centralized authentication protocol
KM$_1$	First proposed protocol presented in this document
KM$_2$	Second proposed protocol presented in this document
MAC	Message Authentication Code
MIKEY	Multimedia Internet KEYing protcol
MIME	Multipurpose Internet Mail Extenseions
MITKerberos	Open-sourced implementation of Kerberos protocol [26]
MSec	Multicast Security
PC	Personal Computer
PCL	Protocol Composition Logic

RAM	Random Access Memory
RFC	Request For Comment, a memorandum published by IETF
SIP	Session Initiation Protocol
SPI	Security Parameter Index
TCP	Transmission Control Protocol
TEK	Traffic Encryption Key
TGK	Traffic Encryption Key Generation Key
TGS	Kerberos Ticket Granting Sever
TGT	Kerberos Ticket Granting Ticket
TKT	Kerberos session Ticket

Chapter 1

Introduction

1.1 Problem Description

Multimedia group communication is becoming a common channel for everyday work within enterprises and institutes. Protecting the privacy and confidentiality of its content is paramount in network security.

In group communication, two or more principals (e.g., humans or IP addresses) form an entity called a group. When the principals communicate in the context of the group, anything transmitted by a member of the group is received by all others in the group. Group communication, and its underlying technology, multicast, have important applications in both Internet-scale, and enterprise and public-safety settings.

In this document's context, bootstrapping is the transport of an initial cryptographic key which can be used to secure future communications, including the transport of other keys. Group key management protocols are examples of security protocols that allow the establishment of keying materials that can be used by its underlying transport layer's encryption mechanism.

This essentially is what the Internet Engineering Task Force (IETF)'s multicast security working group (MSec) [20] calls a registration protocol [4, 16]. Its intent is to transport an initial cryptographic key to authorized members of a group so they can then communicate securely with other members of the group. All members of the group possess this key. However, the process of bootstrapping for such a protocol or system is not always trivial. This thesis attempts to address this problem.

1.2 Contributions

The contribution of this thesis are the two possible designs of bootstrapping for secure group communication in the enterprise and public-safety settings. Both designs were rigorously analyzed and validated through prototype implementation. There is extensive prior work on authenticated key agreement and transport and also considerable work on group key transport and agreement (related work will be discussed in Section 7). However, prior work typically designs new protocols "from scratch." The designs proposed in this thesis instead leverages existing de facto standard protocols, and ones that are deployed widely, which is arguably more realistic.

Also, discussion of the implementation will be provided. The starting points for the implementation are the publicly available reference implementation of Kerberos [26] and an open-source implementation of MIKEY [25] . Empirical results will be presented based on the implementation that demonstrate the inherent trade-offs between security and performance in the designs. Additionally, the implementation using open-sourced software libraries will provide insights in how existing systems can adapt to the protocol proposed.

1.3 Design Overview

Kerberos [29] was adopted as the authentication protocol as it is the de facto standard for authentication in enterprise and public-safety settings. Kerberos is a protocol for centralized authentication. It was based originally on a well-known authentication protocol with a trusted third-party [28]. Kerberos will be discussed in more detail in Section 2.1.

Multimedia Internet Keying or MIKEY [1] was adopted as the key transport protocol for two reasons. One is that it is a proposed standard for key transport from the IETF. The other reason is that an express intent of Arkko et al. [1] is for MIKEY to be useful as a registration protocol to transport group keys as characterized by IETF MSec. Based on the findings of this thesis, there is no prior work on how exactly this would work. This thesis addresses this gap by proposing two ways in which a group key is transported to only authenticated and authorized group members. Details of MIKEY that are relevant to this work will be discussed in Section 2.3.

The manner in which the objective of transporting a group key impacts the designs as there is an assumption on the existence of a trusted third-party, separate from the authentication mechanism, that is the final arbiter on whether a principal is authorized to a key. In keeping with terminology from IETF MSec, this trusted third-party is called the Group Controller and Key Server (GCKS) [4, 16].

It can be argued that the composition presented can be used for the authenticated transport of any key, and not just group keys. This is certainly true. However, it may be more efficient to use other approaches in such cases, such as authenticated Diffie-Hellman [14] or the user-to-user mode of Kerberos [29], that do not involve the additional trusted third-party, the GCKS.

1.4 Methodology

Compositions of the nature needed by this work can be a technical challenge [9]. Consequently, simplicity is a design criterion in the process of composition. The intent is that this results in a composed protocol that is amenable to intuiting desired security properties. (However, too much simplicity, as presented in the initial design, KM_1, can in itself result in security problems. This will be discussed this in Section 4.3.)

Notwithstanding the design philosophy of simplicity, it was recognized that history is replete with seemingly correct security protocols being broken. Therefore, it behooves the necessity to rigorously analyze the designs. Indeed, it is such analysis that lead to the discovery of the weakness in KM_1 as forementioned. The properties of interest were expressed and analyzed using Protocol Composition Logic (PCL) [12, 15, 32] .

There are two reasons that PCL is an appropriate choice in this context. One is that the problem addressed is one of secure composition. Two protocol suites that exist, and that can be shown or are widely thought to be secure, are composed into a single protocol suite. Prior work on PCL provides constructs and terminology to characterize the kinds of compositions that the proposed designs use. The two kinds are sequential and parallel composition [32, 33].

The other reason that PCL is a good choice is convenience. With PCL, security properties of interest can be proved by considering the actions of honest principals only [12, 32]. The actions of adversaries do not have to explicitly be modelled. A principal is said to be honest if it faithfully executes the steps that correspond to a role in the protocol. In the proofs, it was shown that honest principals send out "safe" messages only. Roy et al. [32] provide a precise characterization of safe messages; informally, a message is safe if any data that is to be kept secret that appears as part of the message is protected by a cryptographic key that is a member of a set of keys that was initially prescribed. Once it can be shown that honest principals send out safe messages only, there is no longer a need to explicitly model adversaries. This can then be the basis for proving other security properties of interest.

To demonstrate the feasibility of the proposed design, an implementation of the protocol using publicly available third-party libraries. This also allows the collection of empirical measurements that will be used for efficiency and performance analysis.

1.5 Outline

The remainder of this thesis is organized as follows. In the next section, details of IETF Msec, Kerberos, MIKEY and Protocol Composition Logic that are relevant will be discussed. Section 3 will present the proposed protocol, KM_1 and KM_2. Section 4, discusses the analysis of KM_1 and KM_2 using PCL. This also includes a possible weakness in KM_1 that KM_2 does not have, and as associated trade-off in the total number of keys for each design. Section 5 discusses the implementation of KM_1 and KM_2. Also, some empirical results will be presented based on the implementation in that section. Related work will be discussed in Section 7, and conclude with Section 8.

Chapter 2

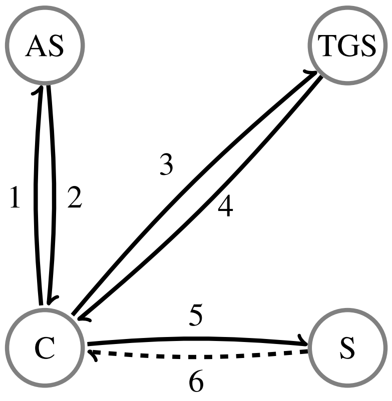
Background

2.1 Kerberos

This section will provide a limited discussion on Kerberos. Reader should be referred to Neuman et al. [29] for a more comprehensive treatment.

Kerberos [29] is a protocol for centralized authentication. By centralized, it means that it relies on a trusted third-party. Its original design was based on the work of Needham and Schroeder [28]. It has since been changed to incorporate other advances, such as the work of Denning and Sacco [13]. It is a de facto standard for authentication in enterprise and public-safety settings [23].

In Figure 2.1, the Kerberos protocol and its participants are shown. C is a client that seeks to authenticate itself, and prove this to a server S, from whom it seeks some service. KAS is the Kerberos Authentication Server that is able to accept and validate C's credentials. TGS is the Ticket Granting Server that hands out authentication tokens called tickets. The KAS and TGS together are the trusted third-party; in deployments, they are sometimes implemented together



1. **AS_REQ**
2. **AS_REP**
3. **TGS_REQ**
4. **TGS_REP**
5. **AP_REQ**
6. **AP_REP**

Figure 2.1: The Kerberos authentication protocol. Step 6 is shown dotted as it is optional. Steps 1–2 are used by C to authenticate itself to the KAS and acquire a ticket granting ticket, TGT . Steps 3–4 are used by C to leverage the TGT to authenticate itself to the TGS and acquire a ticket, TKT. Step 5 is used by C to leverage TKT and authenticate itself to S. Step 6 is used to authenticate S to C.

as a Key Distribution Center (KDC). S relies on the KAS and TGS to have authenticated C. C demonstrates this to S by presenting a valid ticket issued by the TGS.

The details of Steps 1–4 are not important to the work in this thesis. However, it is assumed that the KAS and TGS successfully authenticate C, and provide it with a valid ticket to contact S in those steps. That assumption is well-founded — these properties have been established for Kerberos in prior work [6, 8, 32]. Furthermore, the longevity of Kerberos gives confidence that it is a sound protocol for authentication [23].

In Step 5, C sends the following **AP_REQ** message to S.

AP_REQ: $AUTH_{C,S}, \{TKT\}_{K_S}$

$$AUTH_{C,S} = \{S, C, T\}_{K_{C,S}}$$
$$TKT = C, S, T, \text{Lifetime}, K_{C,S}$$

TKT is the ticket issued by TGS in Step 4. It is encrypted with K_S, the long-term key known to S and TGS only, and is therefore intended to be opaque to C. $AUTH$ is called an authenticator; it is constructed by C. S validates the contents of $AUTH$ against the contents of TKT. T is a timestamp, Lifetime indicates the validity of the ticket, and $K_{C,S}$ is a session key generated by TGS for use by C and S.

In Step 6, S may optionally authenticate itself to C. This message is called **AP_REP**. its format is not shown as the protocol uses a different **AP_REP** in KM from what is used in Kerberos (see Section 3).

2.2 IETF MSec

Multicast enables group communication and an example of an associated group security architecture is the Multicast Group Security Architecture. The architecture is discussed by Hardjono and Weis [16] and Baugher et al. [4]. This document will refer the architecture as "IETF MSec." While the focus of IETF MSec appears to be Internet-scale settings, it can be argued that it is also applicable in enterprise and public-safety settings. Furthermore, based on the knowledge at the creation of this document, there is no alternative that is as widely accepted.

A comprehensive discussion of IETF MSec is beyond the scope of this document. Figure 2.2, shows its main components and processes. The focus of IETF MSec is a cryptographic approach to securing multicast. There are several cryptographic keys that a client needs to acquire before it is able to communicate as part of a group. These keys are derived or transported with protection from an initial key. As the problem addressed in this document is that of bootstrapping secure

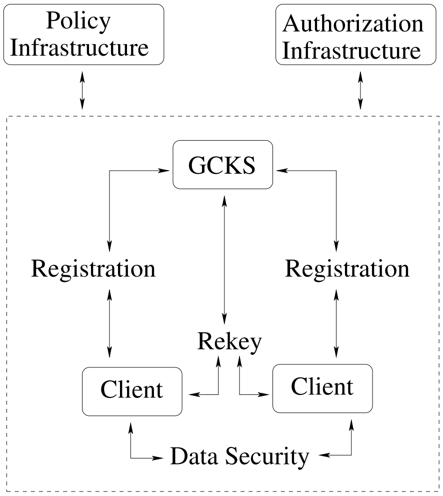


Figure 2.2: IETF Msec. This figure is adapted from Baugher et al. [4]. Our focus is the registration protocol.

multicast, the focus is the transport of this initial key for the group; called the "group key", where K_G denotes the group key of the group G along with the Group Security Associations (GSA). In the context of IETF MSec, a Security Association usually contains the following attributes:

- selectors, such as source and destination transport addresses.
- properties, such as an security parameter index (SPI) or cookie pair, and identities.
- cryptographic policy, such as the algorithms, modes, key lifetimes, and key lengths used for authentication or confidentiality.
- keys, such as authentication, encryption and signing keys.

A client acquires K_G using what IETF MSec calls a registration protocol with an entity that it calls the Group Controller and Key Server (GCKS) (see Figure 2.2). The GCKS "represents the entity and functions relating to the issuance and management of cryptographic keys used by a multicast group" [16]. Once the client acquires K_G, it is able to communicate with the others in the group. The GCKS entity is responsible in accessing the policy and authorization infrastructures to determine how it should handle client requests.

The following are the aspects and components of IETF MSec that are relevant to this work.

- Whether a client is authorized to a group. This is enforced by the GCKS. The policy that underlies such client authorization is beyond the scope of this thesis. For the purposes of this thesis, authorization is modelled at the level of abstraction that a client is either authorized or is not authorized to a group at a given instant.

  In the context of bootstrapping with Kerberos, the division of authority between Kerberos and components of IETF MSec needs to be considered. One of the main design choices that was made was to ensure that the GCKS is the final arbiter on authorization to groups, as is required by IETF MSec. Consequently, this also is an aspect from IETF MSec that was captured in the proposed designs.

- The goal is to transport K_G. It is acquired by a client as a consequence of running the registration or re-registration protocol. In the context of this work, the problem of bootstrapping multicast is the problem of transporting K_G to a client that is authorized to be a member of a group.

2.3 Multimedia Internet KEYing (MIKEY)

MIKEY [1] is an IETF proposed standard protocol for the transport of cryptographic keys. Two of MIKEY's design goals make it a particularly good choice as the key transport protocol to use in bootstrapping secure multicast.

One is that MIKEY is intended to be lightweight. Indeed, the default in the pre-shared key mode of MIKEY is to transport a key in a single round of protocol communication. The second design aspect of MIKEY that makes it a good fit for us is that one of its explicit intents is to be useful for key transport as part of the registration protocol in IETF MSec. However, how exactly MIKEY should be used as part of IETF MSec has, not been specified prior to this work. KM_1 and KM_2 can be seen as two proposals for how this should happen.

In the remainder of this section, only details of MIKEY that are relevant to this work will be presented. Both designs focuses on MIKEY's pre-shared key mode; however, KM is compatible with its other modes as well. MIKEY's other modes are public-key and Diffie-Hellman [1].

MIKEY defines two roles for its participants: Initiator and Responder. The Initiator has a key that it wants to transport to the Responder. When used in the pre-shared key mode, MIKEY assumes that the Initiator and Responder share a symmetric key with which the MIKEY payload can be protected. This key is different from the key that is to be transported.

The goal of MIKEY is to deliver sufficient keying materials for the Responder to derive the Traffic Encryption Key (TEK) , the key used for the multimedia traffic encryption. This includes the encrypted TEK Generation Key (TGK) along with other security policies as parameters to generate the TEK.

The Initiator initiates communication by sending what Arkko et al.[1] call an **I_MESSAGE**, whose format is the following.

I_MESSAGE: $\text{HDR}, T, \text{RAND}, [ID_i], [ID_r], \{\text{SP}\}, \text{KEMAC}$

KEMAC: $E(\text{pre-shared\_key}, \{\text{TGK}\})||\text{MAC}$

In the above message, $[\cdot]$ indicates an optional component of the message, $E(k, m)$ indicates the encryption of message m using key k, $||$ denotes concatenation and $\{\cdot\}$ indicates zero or more occurrences of the component. The HDR is called a general MIKEY header. It is used to transport data such as the version number of MIKEY that is being used, and an identifier for the message. It is also used to communicate some other important information, such as what pseudo-random function is to be used to generate keys, and whether the Initiator expects the Responder to send a response. T is a timestamp; MIKEY relies on synchronized clocks to detect replay. This is one of the design choices that makes MIKEY lightweight.

RAND is a random seed for key generation. ID_i is the identity of the Initiator, and ID_r is the identity of the Responder. The reason they are optional is that one of the participants may already have state regarding the other; MIKEY allows this, and accounts for it by making those fields optional. The SP stands for "security policy." Its intent is to communicate the policy that the participants should use in their choices regarding security, such as the encryption algorithm to be used. KEMAC contains the key (TGK) being transported. It is encrypted with the pre-shared key. KEMAC also contains a Message Authentication Code (MAC) for the receiver to verify authenticity and integrity.

The design presented in this docment use the **I_MESSAGE** only. Therefore, other kinds of MIKEY messages will not be discussed. Based on the findings of this work, there has been no rigorous security analysis of MIKEY. However, it was adopted for the reasons cited at the start of this section, and because it is a proposed standard of the IETF. Arkko et al. [1] include a section titled "Security Considerations" as required by the IETF, in which they argue the security of MIKEY. The proofs in this document for the security properties of KM assume the security of MIKEY in the "safe message" sense as discussed in Section 1.4 as part of the discussions on PCL. That is, it was assumed that if a message is safe, then transporting that message between honest participants using MIKEY keeps that message safe.

2.4 Protocol Composition Logic

PCL is sound in a sense that is customary in mathematical logic [12]. If a property is entailed syntactically, then it is entailed semantically. It should be pointed out, however, that there are known limitations in the use of PCL in proving security properties. Probably the most significant is that it is a symbolic approach, and does not give us any assurances about the computational difficulty that an attacker would face in compromising (an implementation of) the designs. Proofs in PCL idealize cryptographic functions such as encryption and randomization. Nevertheless, it can be argued that the proofs provide confidence that the designs are sound. Leveraging more recent work that bridges the computational and symbolic approaches (see, for example, [10]) to prove that the compositions are sound is a topic for future work.

A comprehensive description of PCL is again, beyond the scope of this document. Sufficient background on PCL will be provided on demand so the work presented is understandable.

Security Properties There are two security properties of primary interest; they are shown in PCL notation in Figure 2.3. These are instantiated for the designs in Propositions 1 and 2 in Section 4.

In the figure, the group that seeks to transport the key is denoted as G, and the key associated with it as Γ. $\widehat{X}, \widehat{KAS}, \widehat{TGS}$ and $\widehat{GCKS}$ are called principals in PCL. The principal $\widehat{X}$ is used to represent a client that seeks to acquire a group key. The principals $\widehat{KAS}$ and $\widehat{TGS}$ are the Authentication Server and Ticket Granting Server in Kerberos (see Section 2.1).

A principal instantiates a thread of execution that plays a role in a protocol. A thread that corresponds to the principal $\widehat{X}$ is written as X or $(\widehat{X}, \eta)$. An example of a role from this work is in Figure 4.1. A role specifies sequential protocol operations such as “receive message,” “encrypt” and “send message.” In Figure 2.3, the initial condition, “For $\widehat{X} \neq \ldots$” specifies that it

For $\widehat{X} \neq \widehat{KAS} \wedge \widehat{X} \neq \widehat{TGS} \wedge \widehat{X} \neq \widehat{GCKS}$,

$$\textbf{Safety}: \ \exists\eta.\, \mathsf{Has}((\widehat{X},\eta),\Gamma) \supset \mathsf{GroupAuth}(\widehat{X},G)$$

$$\textbf{Availability}: \mathsf{Honest}(\widehat{X}) \supset (\mathsf{GroupAuth}(\widehat{X},G) \supset (\forall\eta.\, \mathsf{Has}((\widehat{X},\eta),\Gamma)))$$

Figure 2.3: The two main security properties for KM, expressed in PCL. The safety property asserts that if some thread of a principal $\widehat{X}$ has the key for the group G, then $\widehat{X}$ is an authorized member of G. The availability property asserts that if $\widehat{X}$ is an authorized member of G then it acquires the key for G.

does not consider cases that $\widehat{X}$ is one of those principals. The reason is that the $\widehat{KAS}$, $\widehat{TGS}$ and $\widehat{GCKS}$ are trusted third-parties in the protocol. The first two are from Kerberos (see Section 2.1) and the last is from IETF MSec [20].

In PCL, assertions involve predicates and formulas. Corresponding to each protocol action (e.g., "encrypt with symmetric key") is a predicate (e.g., "SymEnc.") Such a predicate is used to indicate that the action occurred. There may also be other predicates and formulas that do not correspond directly to an action. None of the predicates that appears in Figure 2.3 corresponds to a protocol action.

In the figure, the Has, Honest and GroupAuth predicates is used to express the security properties. The first two have been used extensively in prior work on PCL; the last was introduced to express authorization to a multicast group. The Has predicate is used to indicate that a thread possesses a particular piece of data. The Honest predicate indicates that a principal is honest; that is, its threads faithfully follow the steps that correspond to a role in the protocol. The GroupAuth predicate is used to indicate that a principal is authorized to a multicast group.

The safety property in Figure 2.3 expresses that if any thread of $\widehat{X}$ possesses the group key

Γ, then $\widehat{X}$ is authorized to the group G. (The symbol "$\supset$" may be read informally as "implies.") The availability property expresses that provided $\widehat{X}$ is honest, if it is authorized to G, then all of its threads possess Γ. In Propositions 1 and 2 in Section 4, the expected situation in which these properties to hold was specified. The situation is that a thread of $\widehat{X}$ faithfully executes a role in KM.

It is argued that the safety and availability properties that is articulated above are meaningful as the main security properties of interest in the current context. Underlying the argument is the observation that in the current context, ultimately the goal is about transporting Γ to authorized group members. The safety and availability properties capture an "if and only if" condition – a client should have Γ if and only if it is authorized by the GCKS to it, and its threads faithfully play the role of a client.

We acknowledge that there may be other security considerations that are of interest beyond the safety and availability properties. For example, the confidentiality of group memberships may be important in some settings. Properties other than the safety and availability properties are not considered in this work. An articulation and consideration of other properties is topic for future work.

Chapter 3

Composition of Kerberos and MIKEY

In this section, the description is provided for KM_1 (Section 3.1) and KM_2 (Section 3.2). The design criteria for KM are simplicity, efficiency and scalability. It can be argued that in each case that the designs have adhered to the first criterion of simplicity. This is essential to the designs since it utilizes existing standards.

Our initial approach is to consider extending the existing protocols. This is directly reflected by the sequential composition of Kerberos and MIKEY for KM_1. KM_2 examines the concept of the group entities being as Kerberos principals. This serves as an mean for authority delegation. This also leverages the existing Kerberos principal management infrastructure to include group management.

Efficiency and scalability will be addressed as part of the empirical assessment of the implementations in Section 6.

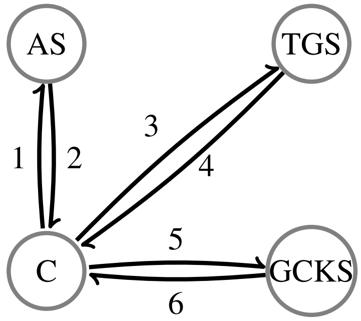


Figure 3.1: KM$_1$ – GCKS as a Kerberos principal.

3.1 Design of KM$_1$

- Steps 1 – 4 are the same as in Figure 2.1 with $S = GCKS$.
- Step 5, $S = GCKS$ in TKT, and $AUTH_{C,GCKS} = \{G, C, T\}_{K_{C,GCKS}}$, where G is the identity of the group that C wants to join.
- Step 6 is the following MIKEY message – **MIKEY-INIT:** $C, G, T, \{K_G\}_{K_{C,GCKS}}$.

KM$_1$ is conceptually simple: $GCKS$ is treated as the server, S, in Kerberos. Steps of KM$_1$ are shown in Figure 3.1. Steps 1–4 are from Kerberos. Step 5 is also from Kerberos; it is the **AP_REQ** message (see Section 2.1) with S instantiated to $GCKS$ in TKT, and S instantiated to the identity of the group G whose key C seeks, in $AUTH$. Unlike in Kerberos, however, Step 6 is not optional. It is a MIKEY message that is used to transport the group key, Γ from the $GCKS$ to the client, C.

Consequently, the registration protocol from IETF MSec can be seen as Steps 5 and 6: a sequential composition of Step 5 from Kerberos, and a new step, with MIKEY for key transport.

The simplicity of KM$_1$ belies an important nuance: the session. The session key, $K_{C,GCKS}$, is used for two purposes. It is used by C to authenticate itself to the $GCKS$, and further by the $GCKS$ to authorize C to the key Γ. This is indicated by the mismatch between TKT and $AUTH$; the former identifies the $GCKS$, and the latter, G. Even if C is not authorized to any group, it passes the first test, that of authenticating itself to the $GCKS$.

An immediate consequence is a possible denial-of-service attack by unauthorized clients on the $GCKS$. However, given that the scenario considered is closed enterprise and public-safety settings, we argue that this is not a major threat. C still needs to be able to acquire a valid ticket to authenticate itself to the $GCKS$.

There is still an over-reliance of $K_{C,GCKS}$, however. Indeed, in Kerberos, this key is used only to authenticate C to the server. Our observation regarding this over-reliance is related to a possible weakness in KM$_1$ that is discussed in Section 4.3.

3.2 Design of KM$_2$

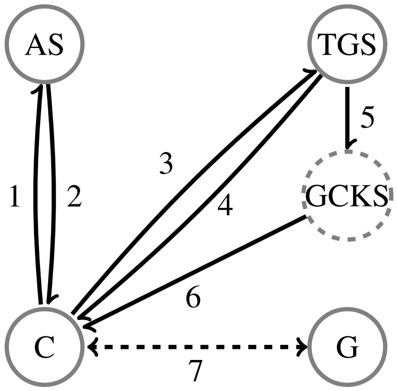


Figure 3.2: KM$_2$ – Multicast group as a Kerberos principal.

- Steps 1 – 4 are the same as in Figure 2.1 with $S = G$, where G is the identity of the group that C wants to join.

- Step 5 is the following MIKEY message – **MIKEY-INIT-TGS:** $C, G, T, \{K_{C,G}\}_{K_G}$. T is the timestamp.

- Step 6 is the following MIKEY message – **MIKEY-INIT-GCKS:** $C, G, T, \{\Gamma\}_{K_{C,G}}$.

- Step 7 was shown as a dotted line as there is no authentication or authorization step that corresponds to it.

KM$_2$ is somewhat more complex than KM$_1$. In KM$_2$, rather than using Kerberos for only authentication, the $GCKS$ delegates a portion of the authorization to groups to the TGS. However, we emphasize that in keeping with IETF MSec, the $GCKS$ remains the final arbiter on group authorizations.

The $GCKS$ now no longer has a role from amongst the traditional roles in Kerberos. Rather, the group G that C seeks to join is identified as a Kerberos principal; it is perceived as the server, S.

Figure 3.2 shows the components and steps in KM$_2$. As the figure indicates, Step 5 is now communication from the TGS to the $GCKS$. Step 6 is communication from the $GCKS$ to C. Both these steps involve the transport of keys, and both use MIKEY. That is, KM$_2$ as composing Steps 1–4 of Kerberos with the two new steps, 5 and 6, that use MIKEY. The registration protocol from IETF MSec corresponds to Steps 4–6.

The $GCKS$ is shown in a dotted circle as it does not correspond to any roles in Kerberos. The group G is shown in a solid circle as it is perceived as the server in Kerberos to which C seeks to authenticate itself.

The contents of the MIKEY messages in Steps 5 and 6 are shown in the caption of Figure 3.2. In Steps 3 and 4, C identifies the group G as the server to which it wants to authenticate itself. The TGS determines whether C is authorized to G. If so, it transports $K_{C,G}$ to C in Step 4. This session key is intended to protect the transport of the group key Γ that is sent in Step 6 by the $GCKS$ to C. Notwithstanding the issuance of $K_{C,G}$ to C by the TGS, the $GCKS$ may withhold the transport of Γ to C for policy reasons. In this manner, the $GCKS$ remains the final arbiter on group authorizations.

It may seem somewhat curious that the TGS was chosen to communicate directly with the $GCKS$ in Step 5. Instead, it could have been Step 5 to be communication from C to the $GCKS$, with the TKT communicated by the TGS in Step 4 used by C to demonstrate to $GCKS$ that it is authentic. We conjecture that this alternative approach is largely equivalent to KM$_2$. However, it has a conceptual problem. The TKT identifies G as the server, and not $GCKS$. We argue that our design of KM$_2$ is cleaner.

Chapter 4

PCL Analysis

As mention in Section 1, KM_1 and KM_2 has been analyzed using PCL [12, 15, 32]. Here, we present one of our proofs that illustrates our approach. Also, the discussion of our analysis explicates the assumption that lead us to discover the possible weakness in KM_1 that is discussed in Section 4.3.

4.1 KM_1

Figure 4.1 shows KM_1 expressed in PCL. There are two roles, **Client** and **GCKS**. The **Client** is assumed to be composed sequentially with the client operations from Steps 1–4 of Kerberos. We refer the reader to Roy et al. [32, 33] for an expression of those steps in PCL. As Figure 4.1 indicates, the **Client** first constructs the $AUTH$, which involves encrypting $G.\hat{C}.t$ with the key $K_{\hat{C},\widehat{GCKS}}$. The operation "." is used to indicate concatenation in PCL. As mentioned in Section 1 in the context of the security properties of interest, $\hat{C}$ and $\widehat{GCKS}$ are principals, and t is the timestamp. The symbol ":=" is used for instantiating a variable to the left with the value to its

Client$(C, \widehat{KAS}, \widehat{TGS}, \widehat{GCKS}, t)$ [	**GCKS**(S) [
$\cdots$ *Steps 1–4 of Kerberos* $\cdots$	
$\cdots$ *tkt is received in Step 4* $\cdots$	
	receive $\;tkt.\, enc_{c,s}$
$enc_{c,s} :=$ symenc $\;G.\, \widehat{C}.\, t, K_{\widehat{C},\widehat{GCKS}}$	$text_{tkt} :=$ symdec $\;tkt, K_{\widehat{GCKS}}$
send $\;tkt.\, enc_{c,s}$	match $\;text_{tkt}\;$ as $\;\widehat{C}.\, \widehat{GCKS}.\, t'.\, K_{\widehat{C},\widehat{GCKS}}$
	$text_{c,s} :=$ symdec $\;enc_{c,s}, K_{\widehat{C},\widehat{GCKS}}$
receive $\;t.\, enc_{s,c}$	match $\;text_{c,s}\;$ as $\;G.\, \widehat{C}.\, t$
$text_{s,c} :=$ symdec $\;enc_{s,c}, K_{\widehat{C},\widehat{GCKS}}$	$enc_{s,c} :=$ symenc $\;\Gamma, K_{\widehat{C},\widehat{GCKS}}$
match $\;text_{s,c}\;$ as $\;\Gamma$	send $\;t.\, enc_{s,c}$
] C	] S

Figure 4.1: KM_1 expressed in PCL.

There are two roles, **Client** and **GCKS**. Some details were omitted, such as the crytographically strong checksum used in MIKEY, which would appear in the send step of the **GCKS** and the receive step of the **Client**.

right.

The mnemonics `symenc, send, receive, symdec` and `match` are used to indicate operations that a thread that executes a role carries out. As their names suggest, they are used to encrypt, send, receive, decrypt and compare strings, respectively. Each corresponds to a predicate which becomes true after the action is executed. For example, once "send m" is executed by a thread X, the predicate $\mathsf{Send}(X, m)$ is true.

The following proposition asserts the correctness of KM$_1$. The symbol "$\vdash$" can be read informally as "entails." Roles in $[\cdot]_X$ with the subscript X identifies the thread that executes it. Part *(a)* asserts that if the **GCKS** role is executed by a thread faithfully, then the safety property is satisfied. Clearly, execution of the **GCKS** role only cannot guarantee the availability property. Part *(b)* of the assertion is that if a thread executes the **Client** role faithfully, then the safety and availability properties from Figure 2.3 are satisfied.

Proposition 1 *The following are true.*

(a) $KM_1 \vdash [\textbf{\textit{GCKS}}]_S\ \mathbf{Safety}$

(b) $KM_1 \vdash [\textbf{\textit{Client}}]_C\ \mathbf{Safety}, \mathbf{Availability}$

We present a proof in PCL for the safety property of Part *(b)* in Figure 4.3. For our proof, we need to adopt some definitions, axioms and inference rules. We show the ones that we need for the proof we present in Figure 4.2.

Our starting premise, that is expressed in Line (13) of Figure 4.3 is that before execution of the thread C, $\mathsf{SafeNet}(\Gamma, \mathcal{K})$ is true, and the safety property holds for every principal. The predicate $\mathsf{SafeNet}$ is discussed extensively by Roy et al. [32, 33]. Informally, it holds if across all threads of all principals, any occurrence of Γ is a message is protected by some key from the

$$\textbf{GA} \qquad \mathsf{SymEnc}(\widehat{GCKS}, \Gamma, K_{\widehat{X},\widehat{GCKS}}) \supset \mathsf{GroupAuth}(\widehat{X}, G) \tag{4.1}$$

$$\textbf{HON} \qquad \frac{[\,]_X\ \phi \quad \forall \rho \in \mathcal{Q}. \forall P \in BS(\rho).\ \phi\ [P]_X\ \phi}{\mathcal{Q} \vdash \mathsf{Honest}(\widehat{X}) \supset \phi} \tag{4.2}$$

$$\textbf{GI} \qquad \phi\ [a_1; \ldots ; a_n]_X\ \phi$$

$$\text{where } \phi \in \{\mathsf{GroupAuth}(\widehat{Y}, G), \neg\mathsf{GroupAuth}(\widehat{Y}, G)\}, \tag{4.3}$$

$$\text{and } \forall i.\ a_i \neq \mathit{symenc}\ \ \Gamma, K,\ \text{ for some } K \tag{4.4}$$

$$\textbf{DSN} \qquad \mathsf{SafeNet}(s, \mathcal{K}) \equiv \forall X.\ \mathsf{SendsSafeMsg}(X, s, \mathcal{K}) \tag{4.5}$$

$$\textbf{SAF0} \qquad \neg\mathsf{SafeMsg}(s, s, \mathcal{K}) \tag{4.6}$$

$$\textbf{POS} \qquad \mathsf{SafeNet}(s, \mathcal{K}) \wedge \mathsf{Has}(X, M) \wedge \neg\mathsf{SafeMsg}(M, s, \mathcal{K}) \supset \exists k \in \mathcal{K}.\ \mathsf{Has}(X, k) \vee \mathsf{New}(X, s) \tag{4.7}$$

$$\textbf{KSEC} \qquad \mathsf{Has}(X, K_{\widehat{A},\widehat{B}}) \supset (\widehat{X} = \widehat{A}) \vee (\widehat{X} = \widehat{B}) \tag{4.8}$$

$$\textbf{ENC4} \qquad \mathsf{SymDec}(X, E[k](m), k) \supset \exists Y.\ \mathsf{SymEnc}(Y, m, k) \tag{4.9}$$

Figure 4.2: Axioms, definitions and rules used in our proof

Used in Figure 4.3. **GA** and **GI** are new to our work and regard group authorization. **KSEC** can be shown from the results of Roy et al. [32], and the remainder are from prior work [12, 32, 33].

$$\text{Premise, } \mathcal{K} = \{K_{\widehat{C}_1,\widehat{GCKS}}, \ldots, K_{\widehat{C}_n,\widehat{GCKS}}\} \qquad \mathsf{Start}(C)\ [\,]_C\ \mathsf{SafeNet}(\Gamma,\mathcal{K}) \wedge \forall \widehat{X}.\, \mathbf{Safety} \tag{4.10}$$

$$\text{Case 1 :} \qquad \mathsf{Start}(C)\ [\,]_C\ \mathsf{GroupAuth}(\widehat{C},G) \tag{4.11}$$

$$\mathbf{GI} \qquad \mathsf{GroupAuth}(\widehat{C},G)\ [\mathbf{Client}]_C\ \mathsf{GroupAuth}(\widehat{C},G) \tag{4.12}$$

$$(14),(15) \qquad [\mathbf{Client}]_C\ \mathbf{Safety} \tag{4.13}$$

$$\text{Case 2 :} \qquad \mathsf{Start}(C)\ [\,]_C\ \neg\mathsf{GroupAuth}(\widehat{C},G) \tag{4.14}$$

$$\mathbf{DSN} \qquad \mathsf{SafeNet}(\Gamma,\mathcal{K})\ [\mathbf{Client}]_C\ \mathsf{SafeNet}(\Gamma,\mathcal{K}) \tag{4.15}$$

$$\text{Premise} \qquad [\mathbf{Client}]_C\ \mathsf{Has}(C,\Gamma) \tag{4.16}$$

$$\mathbf{SAF0} \qquad \neg\mathsf{SafeMsg}(\Gamma,\Gamma,\mathcal{K}) \tag{4.17}$$

$$(13),(18),(19),(20) \qquad [\mathbf{Client}]_C\ \mathsf{SafeNet}(\Gamma,\mathcal{K}) \wedge \mathsf{Has}(C,\Gamma) \wedge \neg\mathsf{SafeMsg}(\Gamma,\Gamma,\mathcal{K}) \tag{4.18}$$

$$(21), \mathbf{POS}, \neg\mathsf{New}(C,\Gamma) \qquad [\mathbf{Client}]_C\ \exists k \in \mathcal{K}.\, \mathsf{Has}(C,k) \tag{4.19}$$

$$(22), \mathbf{KSEC} \qquad [\mathbf{Client}]_C\ \mathsf{Has}(C,k) \wedge (k = K_{\widehat{C},\widehat{GCKS}}) \tag{4.20}$$

$$(23), \mathbf{ENC4} \qquad [\mathbf{Client}]_C\ \exists Y.\, \mathsf{SymEnc}(Y,\Gamma,K_{\widehat{C},\widehat{GCKS}}) \tag{4.21}$$

$$(24), \mathbf{KSEC}, \neg\mathsf{SymEnc}(C,\Gamma,K_{\widehat{C},\widehat{GCKS}}) \qquad [\mathbf{Client}]_C\ \mathsf{SymEnc}(GCKS,\Gamma,K_{\widehat{C},\widehat{GCKS}}) \tag{4.22}$$

$$\mathbf{GA} \qquad [\mathbf{Client}]_C\ \mathsf{GroupAuth}(\widehat{C},G) \tag{4.23}$$

$$(19),(26) \qquad [\mathbf{Client}]_C\ \mathbf{Safety} \tag{4.24}$$

Figure 4.3: Proof for the part of Assertion (b) from Proposition 1 that pertains to safety.

set of keys $\mathcal{K}$. That is, even though Γ may have been transmitted, it was done so only under the protection of some key from $\mathcal{K}$. The set $\mathcal{K}$, in our context, is all the session keys issued by the TGS for use by the client and the $GCKS$.

The proof is broken down into two cases. Case 1 assumes that the principal $\widehat{C}$ is already authorized to G prior to the execution of the thread C. In this case, all that is needed to show is that $\mathsf{GroupAuth}(\widehat{C}, G)$ continues to hold after the execution of **Client** by C. We use the axiom **GI** as the basis to infer this. The axiom asserts that the validity or invalidity of group authorization is not affected by a sequence of actions, none of which is the encryption of Γ with some key K.

For Case 2, it is assumed that $\widehat{C}$ is not authorized to G before the thread C executes. In this case, we adopt the premise in Line (19) that after execution of **Client** by C, C has Γ. We then need to show that this implies $\mathsf{GroupAuth}(\widehat{C}, G)$. We infer, in Line (21) that the conditions for Axiom **POS** are met, and therefore, in Line (22) we assert that C has a key k from the set $\mathcal{K}$. In Line (23) we infer, from the axiom **KSEC** about Kerberos, that this key must be a session key issued to a thread of $\widehat{C}$.

Then, in Lines (24) and (25), we infer that given that C has an encryption of Γ, some thread must have performed the encryption. Again, **KSEC** allows us to infer that that thread belongs to the principal $\widehat{GCKS}$. An application of the axiom **GA** then allows us to conclude that $\widehat{C}$ is authorized to the group G.

4.2 KM$_2$

The two roles **TGS_CLIENT** and **TGS_CLIENT** in KM$_2$ are composed in parallel. They are both executed at the Kerberos TGS. The role **TGS_CLIENT** transports the key $K_{\widehat{C},\widehat{G}}$ that is shared by the client and the group identified as the Kerberos principal $\widehat{G}$ to the client. The role

```
**TGS_CLIENT**$(A, \widehat{G}, \widehat{C}, t)$ [
    $tkt$ := symenc $\widehat{C}.\widehat{G}.t.K_{\widehat{C},\widehat{G}}, K_{\widehat{G},\widehat{TGS}}$
    $enc_{a,c}$ := symenc $\widehat{G}.t.K_{\widehat{C},\widehat{G}}, K_{\widehat{C},\widehat{TGS}}$
    send $\widehat{C}.\ tkt.\ enc_{a,c}$
    ] $A$
```

```
**TGS_GCKS**$(B, \widehat{G}, \widehat{S}, t)$ [
    $enc_{b,s}$ := symenc $K_{\widehat{C},\widehat{G}}, K_{\widehat{G},\widehat{TGS}}$
    $sign_{b,s}$ := sign $(G.\,\widehat{C}.\,\widehat{S}.\,t.\ enc_{b,s}), K_{\widehat{G},\widehat{TGS}}$
    $mikey_{b,s}$ := $G.\,\widehat{C}.\,\widehat{S}.\,t.\ enc_{b,s}.\ sign_{b,s}$
    send $mikey_{b,s}$
    ] $B$
```

```
**CLIENT**$(C, \widehat{A}, \widehat{S}, t)$ [
    receive $tkt.\ enc_{a,c}$
    receive $mikey_{s,c}$
    $text_{a,c}$ := symdec $enc_{a,c}, K_{\widehat{C},\widehat{TGS}}$
    match $text_{a,c}$ as $\widehat{G}.t.K_{\widehat{C},\widehat{G}}$
    match $mikey_{s,c}$ as $G.\,\widehat{C}.\,\widehat{S}.\,t.\ enc_{s,c}.\ sign_{s,c}$
    verify $sign_{s,c},\ (G.\,\widehat{C}.\,\widehat{S}.\,t.\ enc_{s,c}),\ K_{\widehat{C},\widehat{G}}$
    $text_{s,c}$ := symdec $enc_{s,c}, K_{\widehat{C},\widehat{G}}$
    match $text_{s,c}$ as $K_{tgk}$
    ] $C$
```

```
**GCKS**$(S)$ [
    receive $mikey_{b,s}$
    match $mikey_{b,s}$ as $G.\,\widehat{C}.\,\widehat{S}.\,t.\ enc_{b,s}.\ sign_{b,s}$
    verify $sign_{b,s},\ (G.\,\widehat{C}.\,\widehat{S}.\,t.\ enc_{b,s}),\ K_{\widehat{G},\widehat{TGS}}$
    $text_{b,s}$ := symdec $enc_{b,s},\ K_{\widehat{G},\widehat{TGS}}$
    match $text_{b,s}$ as $K_{\widehat{C},\widehat{G}}$
    new $K_{tgk}$;
    $enc_{s,c}$ := symenc $K_{tgk},\ K_{\widehat{C},\widehat{G}}$
    $sign_{s,c}$ := sign $(G.\,\widehat{C}.\,\widehat{S}.\,t.\ enc_{s,c}),\ K_{\widehat{C},\widehat{G}}$
    $mikey_{s,c}$ := $G.\,\widehat{C}.\,\widehat{S}.\,t.\ enc_{s,c}.\ sign_{s,c}$
    send $mikey_{s,c}$
    ] $S$
```

Figure 4.4: Roles of KM$_2$ protocol.

TGS_GCKS transports that key to the GCKS. Note that the GCKS implements the Kerberos principal $\widehat{G}$. As in Figure 4.1, we omit some details, such as the checksum that is be used in the message sent in **TGS_GCKS** that uses MIKEY for key transport.

In the following proposition, we assert that KM$_2$ has the same properties as KM$_1$ with regards to the security properties of interest. A difference is that for KM$_2$, we need to make the assertion for the TGS as well, as Step 4 is considered to be part of the registration protocol of IETF MSec. Also, there is a parallel composition [32] of Steps 4 and 5. That is, they may occur in any order with respect to one another. Of course, one can simply choose not to model this in PCL, and mandate that the two steps occur in a particular order.

Proposition 2 *The following are true.*

(a) $KM_2 \vdash [\textbf{\textit{TGS}}\ (\textbf{\textit{TGS\_CLIENT}} \mid \textbf{\textit{TGS\_GCKS}})]_S\ \mathbf{Safety}$

(b) $KM_2 \vdash [\textbf{\textit{GCKS}}]_S\ \mathbf{Safety}$

(c) $KM_2 \vdash [\textbf{\textit{Client}}]_C\ \mathbf{Safety}, \mathbf{Availability}$

However, this does not capture the flexibility that no ordering needs to be imposed. In practice, not imposing an ordering may have efficiency benefits; for example, the TGS may be able execute several instances of Step (5) in batch, and incur the cost of only a single session with the $GCKS$.

The manner in which we reconcile this parallel composition is to first split the actions of the TGS into three roles: **TGS**, which executes all actions of the TGS prior to the two send actions of Steps (4) and (5), **TGS_CLIENT** that executes the send action of Step (4) and **TGS_GCKS** that executes the send action of Step (5). In the proposition below, "|" is used to indicate parallel composition.

$$\text{Premise}, \mathcal{K} = \left\{K_{\widehat{C},\widehat{TGS}}, K_{\widehat{G},\widehat{TGS}}\right\} \qquad \mathsf{Start}(S)\ \ [\mathbf{TGS}]_S\ \ \mathsf{SafeNet}(K_{\widehat{C},\widehat{G}}, \mathcal{K}) \tag{4.25}$$

$$(29) \qquad \left[\ \textit{enc}_{c,g} := \text{symenc}\ \ \widehat{G}.t.K_{\widehat{C},\widehat{G}}, K_{\widehat{C},\widehat{TGS}}\ \right]_S$$

$$\mathsf{SafeMsg}(\textit{enc}_{c,g}, K_{\widehat{C},\widehat{G}}, \mathcal{K}) \tag{4.26}$$

$$(30) \qquad \Big[\ \textit{enc}_{c,g} := \text{symenc}\ \ \widehat{G}.t.K_{\widehat{C},\widehat{G}}, K_{\widehat{C},\widehat{TGS}};$$

$$\textit{tkt} := \text{symenc}\ \ \widehat{C}.\widehat{G}.t.K_{\widehat{C},\widehat{G}}, K_{\widehat{G},\widehat{TGS}}\Big]_S$$

$$\mathsf{SafeMsg}(\textit{tkt}.\textit{enc}_{c,g}, K_{\widehat{C},\widehat{G}}, \mathcal{K}) \tag{4.27}$$

$$(31) \qquad [\mathbf{TGS\_CLIENT}]_S\ \ \mathsf{SendsSafeMsg}(S, K_{\widehat{C},\widehat{G}}, \mathcal{K}) \tag{4.28}$$

$$(29), (32), \mathbf{DSN} \qquad [\mathbf{TGS}; \mathbf{TGS\_CLIENT}]_S\ \ \mathsf{SafeNet}(K_{\widehat{C},\widehat{G}}, \mathcal{K}) \tag{4.29}$$

Figure 4.5: KM$_2$ PCL Proof for Safety property

Proof for the maintenance of the invariant we need to prove safety in KM$_2$ by the parallel composition of the roles **TGS_CLIENT** and **TGS_GCKS**, for the role **TGS_CLIENT**. The invariant is $\mathsf{SafeNet}(K_{\widehat{C},\widehat{G}}, \mathcal{K})$ for $\mathcal{K} = \{K_{\widehat{C},\widehat{TGS}}, K_{\widehat{G},\widehat{TGS}}\}$. The proof for **TGS_GCKS** is similar.

We then employ the parallel composition theorem of Roy et al. [32, 33]. That theorem asserts that given two protocols $\mathcal{Q}_1$ and $\mathcal{Q}_2$, such that $\mathcal{Q}_1 \vdash \Psi$ and $\mathcal{Q}_2 \vdash \Psi$, and $\Psi \vdash \Phi$, then the parallel composition of $\mathcal{Q}_1$ and $\mathcal{Q}_2$, $\mathcal{Q}_1 \mid \mathcal{Q}_2 \vdash \Phi$. What we seek, then, is some property Ψ that helps us prove our safety property. The property Ψ, in our case, is that $\mathsf{SafeNet}(K_{\hat{C},\hat{G}}, \mathcal{K})$ is true for $\mathcal{K}$ that contains the long-term keys of principals in Kerberos (K_C and K_G, in particular).

We show the proof for the role **TGS_CLIENT** in Figure 4.5. We start in Line (29) with the premise that before execution of the role, the invariant we seek to prove holds. Then, in Lines (30) and (31), we infer that the key $K_{\hat{C},\hat{G}}$ remains safe as any message in which it is contained is protected by a key from $\mathcal{K}$. We infer in Line (32) that the role **TGS_CLIENT** sends out $K_{\hat{C},\hat{G}}$ in safe messages only, and then we are able to conclude in Line (33) that the sequential composition of **TGS** and **TGS_CLIENT** maintains our invariant. The proof for **TGS_GCKS** is similar, and we are able to infer the invariant for the parallel composition as we seek.

4.3 A weakness in KM$_1$

Our work on the proof for Proposition 1 reveals a potential weakness in KM$_1$. Indeed, this was our motivation for designing KM$_2$ as an alternative.

In our work, it is assumed that the Kerberos infrastructure is secure. That is, the principals $\widehat{KAS}$ and $\widehat{TGS}$ are trusted to be honest, keys are kept secret by principals that possess them, only safe messages are sent out, and clients are authenticated correctly. However, we are sensitive to the interface between Kerberos, and the MIKEY steps of KM that we add to Kerberos. We argue that weaknesses that lie in this interface make the composed protocol particularly vulnerable. The session key, $K_{\hat{C},\widehat{GCKS}}$, is used at this interface between Kerberos and the MIKEY steps.

In KM$_1$, $K_{\hat{C},\widehat{GCKS}}$ is used to protect all group keys to which the client is authorized. In other words, the security (secrecy) of all the group keys to which a client is authorized relies on the security of $K_{\widehat{C},\widehat{GCKS}}$. We make this observation precise in the following proposition.

Proposition 3 $\mathit{KM}_1 \vdash \exists\eta.\, \mathsf{Has}((\widehat{X},\eta), K_{\widehat{C},\widehat{GCKS}}) \supset$
$(\mathsf{GroupAuth}(\widehat{C}, G) \supset \forall\gamma.\, \mathsf{Has}((\widehat{X},\gamma), \Gamma))$

KM$_2$ does not have this weakness. A consequence of the partial delegation of group authorization to the TGS by the $GCKS$ is that session keys are specific to each group. The compromise of a session key leads to the compromise of one group key only.

Chapter 5

Implementation

We have implemented KM$_1$ and KM$_2$. The starting points for our implementations were the open-source, reference implementation of Kerberos [26], and an open-source implementation of MIKEY [25]. We have made our implementation available publicly as open-source [37].

We used the sample code (`sclient.c` and `sserver.c`) delivered as part of the MITKerberos source package [26] as a reference to develop the Kerberos client and server code. In the MIKEY implementation with which we started, MIKEY messages are carried as part of the MIME payload in SIP [30]. However, we wanted our implementation to be independent of SIP. The associations with SIP related objects were removed in the extracted MIKEY library. This allowed us to isolate the MIKEY library from other MiniSIP source code.

The sample code demonstrates a generic test that allows a Kerberos client to acquire service tickets and authenticate itself to a Kerberos service host. The sample applications were changed to C++ application to easily invoke the C++ MIKEY library in the MiniSIP project [25]. Changes made to the KDC source code are designed to be compatible with existing Kerberos clients.

For both prototypes, the client needs to pre-acquire the TGT. MITKerberos has a tool `kinit` that allows the end-user to perform this task. It performs Steps 1 and 2 of Kerberos and saves the TGT into a cache at the client.

5.1 KM$_1$ Prototype

The prototype for KM$_1$ consists of two files, `sclient.cxx` and `gcks1.cxx`. The sclient is a Kerberos client principal that attempts to contact the GCKS. The GCKS listens on a specified TCP port for any incoming messages. The client uses the `krb5_sendauth` Kerberos API call to send an authenticator to authenticate itself to the GCKS and awaits a reply.

To allow the client to indicate the multicast group, we attach this information to the `krb5_authdata` data structure as part of the credentials supplied to `krb5_sendauth`. This `krb5_authdata` with become part of the authenticator that is sent in the AP_REQ. As a result, the GCKS uses a modified version of the `krb5_recvauth` Kerberos API call, `krb5_recvauth_gcks` to allow the GCKS to directly retrieve the associated authorization data.

This method is of utilizing the authorization-data field of the authenticator is justified and is compliant with Kerberos. The authorization data ”is optional and will only appear when additional restrictions are to be placed on the use of a ticket, beyond those carried in the ticket itself” [29]. There are existing proposed standards that utilizes this optional field such as Kerberos Cryptosystem Negotiation Extension [38]. Within our context, we argue that the GCKS must know the target group to determine whether the client is authorized to that intended group.

KM$_1$ required no code changes to the MITKerberos KDC. The GCKS implemented in `gcks1.cxx` is just like any other Kerberos server from the standpoint of setup. The administrator is required to first create a Kerberos service principal for the GCKS and generate a keytab file associated to that principal. Then the generated keytab file needs to be transferred onto the GCKS host machine.

The keytab file allows the GCKS to retrieve its own credentials. It essentially contains the long term key K_{GCKS}. This key is used to decrypt the ticket that contains the session key shared between the client and the GCKS (refer to Figure 4.1). This keytab file can be created using the `local/sbin/kadmin.local` executable and the `ktadd [-k keytab_filename] <service name>` command [27].

5.2 KM$_2$ Prototype

A copy of the sclient from KM$_1$ was modified to adapt to KM$_2$. Similarly, we have `gcks2.cxx` for KM$_2$ GCKS. Instead of using the sendauth API call, the client uses the `krb5_get_credentials` MITKerberos API call. This function attempts to retrieve the session ticket credentials for the client with a particular Kerberos service principal (multicast group) from the client's local cache. If the session ticket does not exist in the local cache, it will use a cached Kerberos TGT (obtained through the `kinit` tool) to acquire a new session ticket.

This prototype also requires an additional KDC request flag in the TGS_REQ request to notify the Kerberos KDC to contact the GCKS. The client was modified to also wait for the GCKS's reply separately. However, this creates a possible race condition since the TGS_REP is needed before the message from the GCKS can be decrypted. This issue remains future work for further investigation.

An additional KDC request flag (named `KDC_OPT_MIKEY`) was added to the existing list of KDC options for KDC request located in the MITKerberos krb5.h (global header) file. This method of extending KDC request flag complies with the RFC [29]. The client simply performs a bitwise OR operation with the other request flags. For example, this can be done using the following line of code:

```
krb5_flags req_flags =
        AP_OPTS_MUTUAL_REQUIRED | KDC_OPT_MIKEY;
```

This allows the client to distinguish its requests to the KDC server.

On the KDC server, our code injection is placed after the TGS_REQ is authenticated and when the session key $SK_{C,G}$ is generated. The KDC is assumed to have a mechanism to determine the mapping of the multicast group to a specific GCKS to forward its request to. Afterwards, the KDC will retrieve the multicast group's private key (K_G) between the TGS and the chosen GCKS from its database. This key would have been created and dsitrbuted to the GCKS when the multicast group principal entry was created in the KDC database.

The TGS will send the I_MESSAGE encrypted by K_G through its own pre-defined channel to the delegated GCKS to forward the $SK_{C,G}$. An additional payload is needed for the I_MESSAGE to indicate the client's address and the intended multicast group. This is necessary for the GCKS to identify the client-to-group mapping. Similarly, the second I_MESSAGE encrypted by $SK_{C,G}$ will carry the TGK along with the additional payload for the client to ensure the request is properly mapped to the correct multicast group.

5.3 Confidentiality of Group Memberships

The new payload introduced in KM_2 is added as an GeneralExtension MIKEY payload as supported by the original MIKEY protocol (section 6.15 [1]). This GeneralExtension payload has integrity protection as it is included as part the calculation of the MIKEY message's MAC.

However, some environment may require such mappings (group membership assignment) to remain confidential. For example, we may not want to reveal the identities of agents that are assigned to a particular investigation even within the same institute. In such scenarios, we recommend the payload should also be encrypted using the $SK_{C,G}$. This additional encryption step was not implemented in our prototype as it is an optional requirement and believed to have limited impact on the overall analysis.

KM$_1$ does not have this concern. Since the membership information is contained within the authenticator, it is encrypted as part of its original design. No extra steps are needed to accomplish this.

5.4 Perturbation

Both KM$_1$ and KM$_2$ are designed to have minimal effects on existing architecture. Apart from administrative tasks, KM$_1$ allows existing KDC to operate without any modifications. This is a strong advantage over KM$_2$. Especially in enterprise and public-safety settings, any modifications to existing systems can impose large administrative overhead. The costs and potential risk factors involved to update all instances of master and slave servers must be carefully examined.

From a broader perspective, however, KM$_2$ does not affect existing KDC algorithms. As mentioned in Section 5.2, after the KDC identifies the KDC_OPT_MIKEY flag, it will send the MIKEY I_MESSAGE to the GCKS. If the flag is not set, the execution flow will be exactly the same as original Kerberos. As recommended in Kerberos V5 RFC [29], the KDC *should* return appropriate error message if the KDC cannot recognize certain option flags. If the system supports this feature, the client can identify whether the server supports the KM$_2$ protocol.

5.5 Administrative Authority

For administrative authority, both designs has clear distinctions. KM_2 requires each group to be a Kerberos principal. Hence, the KDC administrators has authority on group creation/removal but group membership management is still managed by the GCKS. In KM_1, the GCKS has full authority over group management. Depending on the organization's administrative structure, this may influence one's preference for each design.

For an organization that requires centralized administrative control over all group management, then the difference between KM_1 and KM_2 is minimal. However, for organizations that prefer isolating multicast group management from KDC administration (notion of least privileged security design principle [35]), then KM_1 is more suitable if the weakness in KM_1 is acknowledged (single point of failure, see Section 4.3).

5.6 Usability

Usability takes a predominant role in the acceptability of a security system design (or *psychological acceptability*, as explained in early works by Saltzer, J.H. and Schroeder, M.D. [35]). For a thorough analysis, a discussion is provided to show some brief insight on usability based on our implementation experience.

Kerberos is a Single Sign-on authentication service. This means a Kerberos user is only required to enter their secret key once so all subsequent requests are processed using the TGT. A Kerberos client may already have a TGT in its ticket cache from previous Kerberos-related actions (e.g. logging into a corporate account at system startup). This implies as long as there's a valid TGT, there will be no extra security-related steps needed for the user to initiate group communication. A Kerberos-authenticated user will not be required to enter a password to access the group communication service. This seamless experience applies to both KM_1 and KM_2.

As an administrator, the management of Kerberos principals is no different from the existing mechanism. For KM_1, only one extra Kerberos principal is required for the GCKS. For KM_2, the number of Kerberos principal managed is proportional to the number of groups created and destroyed. Given that the number of groups are relatively static, the amount of administrative work is also bound to a constant factor. The intuition is that Kerberos principals can be created for groups prior to communication and should instead be enabled/disabled dynamically when necessary by the GCKS, the final arbiter. We argue that the usability of the system for GCKS to manage group member assignment or access control is out of the scope of this document as it is recognized to be independent from our protocol's design.

In the following section, we present some empirical results we have collected from our implementation. The results reveal interesting trade-offs between performance and the choice between KM_1 and KM_2.

Chapter 6

Empirical Results

6.1 Experiment Design

We ran our experiments on a 2.4GHz Intel PC with 4 GB of RAM that runs the Ubuntu Linux operating system. The test suite used for the experiments was written in C. It involves executing scripts that will automatically handle the creation/removal of Kerberos principals, ticket caches and executing instances of the client and server programs (as discussed in Section 5).

There are several scenarios of interest that we examined. We consider the following as possible parameters within the context of group communication:

- Varying the number of clients in total
- Varying the number of groups
- Varying the number of clients assigned per group
- Varying group diversity

The results are shown graphically in Figures 6.1–6.4. Each data point is a mean of as many samples as it took for the coefficient of variation to be less than 5%. We then calculated a 95% confidence interval. These are shown with the vertical error bars in the graphs. The reason for computing confidence intervals is to check whether the confidence intervals overlap. If they do, then we cannot consider the two values to be statistically distinct (with that confidence).

In all cases, the total time that it takes clients to acquire group keys for all groups to which they are authorized was measured. This is the vertical axis. The clients attempt to acquire keys only to those groups to which they are authorized.

6.2 Results

In Figure 6.1–6.3, 3 curves were plotted for each of KM_1 and KM_2. In Figure 6.4, there are 2 curves for each of KM_1 and KM_2.

6.2.1 Test 1: Number of clients

In Figure 6.1 we measure the relationship between the number of clients and time to acquire keys. For one curve, there is only one group, and all clients are authorized to that group. For another, there are three groups of which all clients are members. And for the third, 5 groups were created in total; a client was authorized and assigned to 3 of the 5 groups uniformly at random. In the latter two cases, each client will make three requests. The difference between those two cases will determine whether number of groups within the database will affect the relationship with number of clients given a constant number of groups assigned to a client.

In all 3 cases, there's a linear relationship between the number of clients and the time required to process their requests. If we examine when the number of groups are increased, the

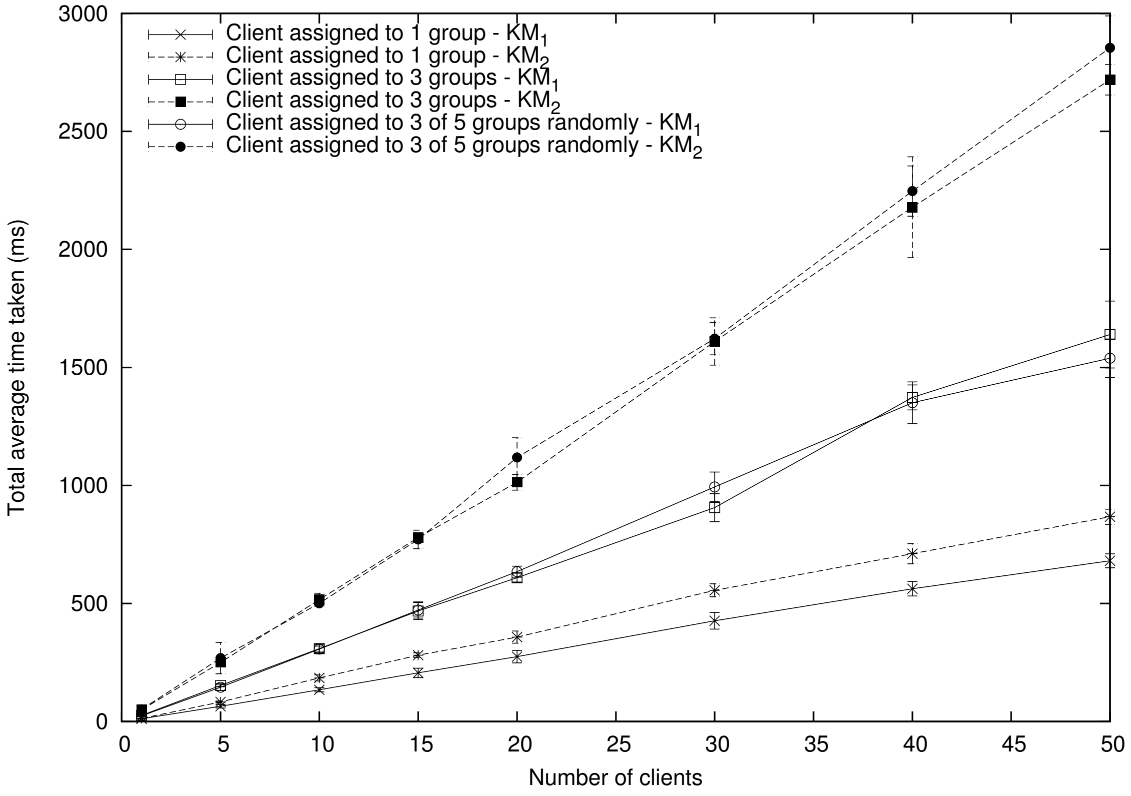


Figure 6.1: Average total time to complete client requests with varying number of clients.

relationship remains the same. In fact, for KM_1 when the groups assigned were multiplied by 3, the total time required also increased by the same factor. This is not the case for KM_2 as the time required increased by a greater factor. However, this relates to the impact of the number of groups instead of clients, as explained in Section 6.2.2.

This result is as expected as the testing script was implemented to process client requests sequentially. The only overhead that might be caused the number of clients is the lookup time for the client's record in the KDC database. From the results, it suggests the effects are minimal.

6.2.2 Test 2: Number of groups

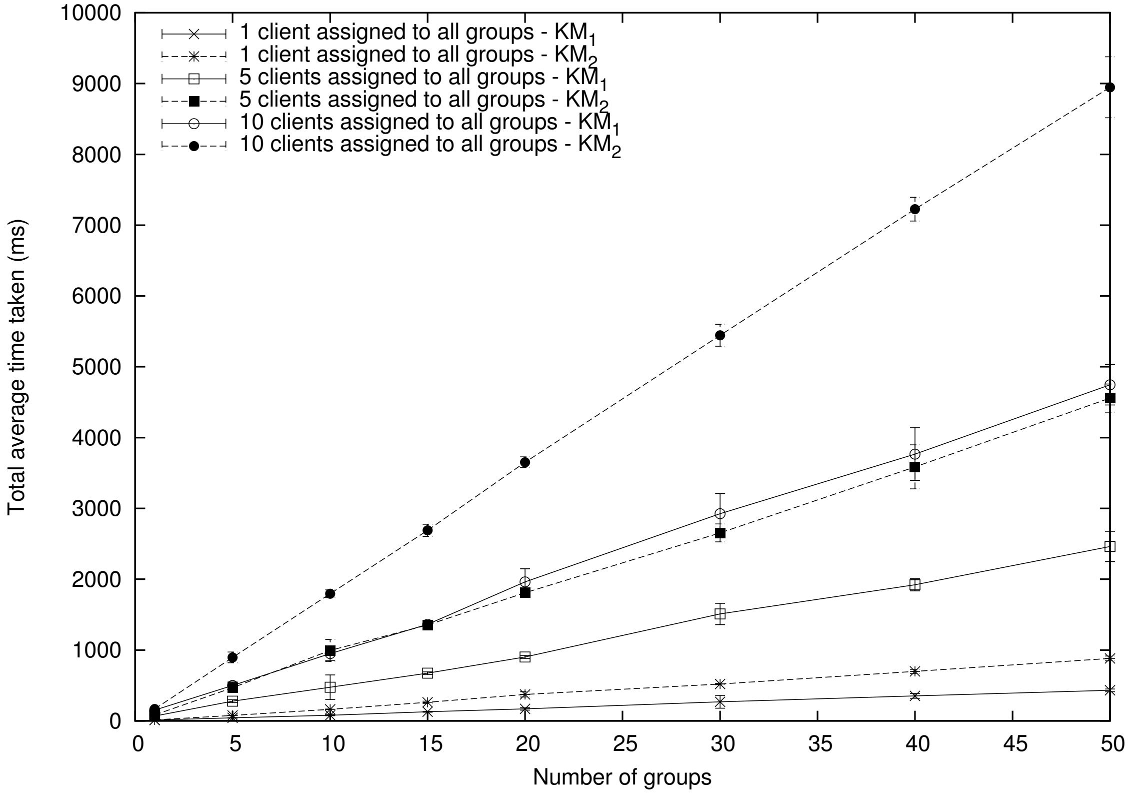


Figure 6.2: Average total time to complete client requests with varying number of clients authorized to groups.

In Figure 6.2, we study how the performance of KM_1 and KM_2 is affected by an increase in the number of groups. As with Figure 6.1, performance is quantified as the total time it takes all clients to acquire groups keys for all the groups to which they are authorized. 3 curves were plotted for each of KM_1 and KM_2. For one curve, there was only 1 client, and this client is authorized to all groups. For another, there were 5 clients, each of which is authorized to all groups. And for the third, there were 10 clients, each of which is authorized to all groups.

This graph also shows there's a linear relationship between the number of groups and the time taken for all the clients to be completed. However, the number of groups has a greater impact on KM$_2$ than KM$_1$ as indicated by the gradient of the line.

One explanation is after a client's first group request in KM$_1$, all subsequent group requests will not require the KDC's involvement. That removes 4 out of 6 steps of the protocol. Although KM$_1$ theoretically has only $\frac{1}{3}$ of the steps required by KM$_2$, not all step's computation time are equal. Hence, the total execution time for KM$_1$ may not be $\frac{1}{3}$ of KM$_1$. This effect is directly related to caching mechanism of the system implemented as mentioned in Section 6.2.5.

6.2.3 Test 3: Group size

In Figure 6.3, we study performance as the size (number of members) of groups increases. As with the prior cases, performance is measured as the total time for clients to acquire group keys. Again, 3 curves were polotted for each of KM$_1$ and KM$_2$. A constant number of clients was chosen; 50 in this case. Then uniformly at random, authorized the clients to groups to ensure a certain size for each group. For one curve, there was one group only. For another curve, there were 5 groups, and for the third curve there were 10 groups.

Figure 6.3 results are somewhat similar to Figure 6.2. The distinction lies again in the magnitude of increase for increasing group size. Comparing Figure 6.2 with 10 clients assigned to 50 groups to Figure 6.3 with 10 groups each with 50 clients assigned, KM$_2$ reached almost 15000ms for Test 3 whilst it only reached 9000ms for Test 2. For KM$_1$, it reached around 6000ms for Test 3 and less than 5000ms for Test 2.

The difference between the two cases is that in Test 2, the client's ticket cache is reused for all subsequent group requests. However, Test 3 clears client's ticket cache after each group's

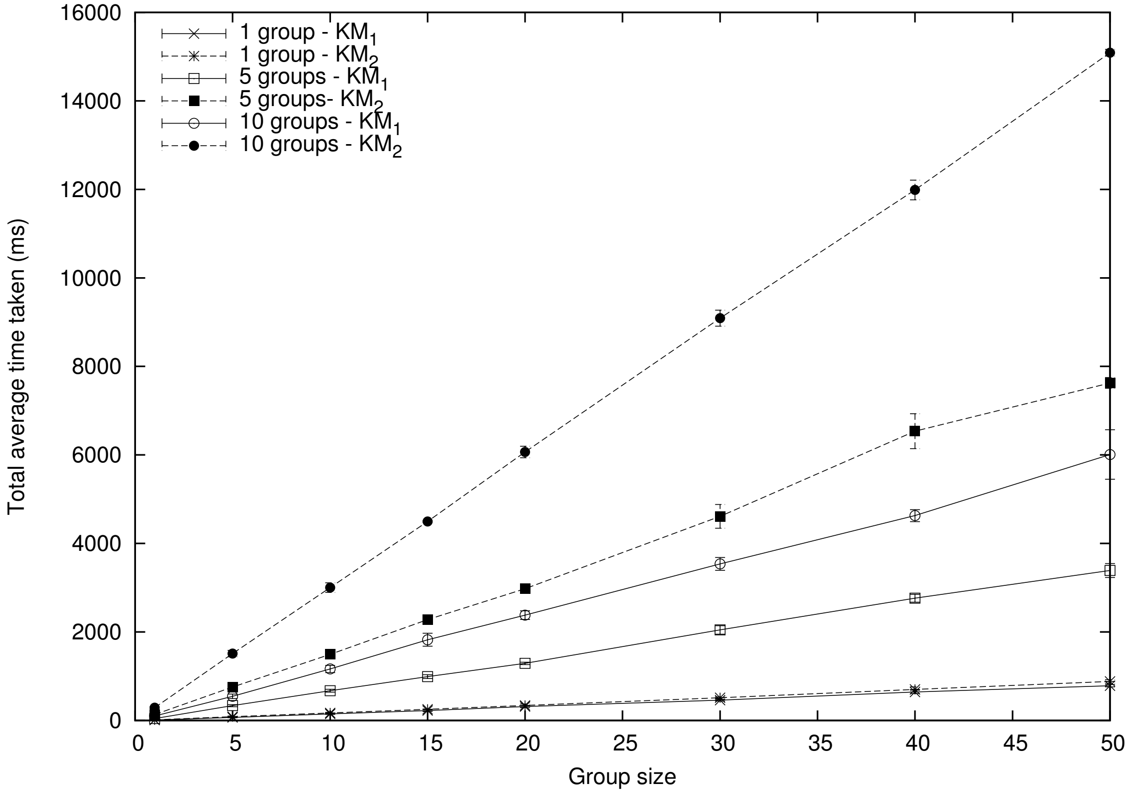


Figure 6.3: Average total time to complete client requests with varying group sizes.

client finishes their requests. Although this related directly to the implementation details of the tests, it again demonstrates the critical impact that the ticket cache has.

6.2.4 Test 4: Group Diversity

In Figure 6.4, we plot the impact on performance of what we call "group diversity." Intuitively, groups are diverse when they do not share common members. For the case of an unbounded number of clients and a constant number of groups, for example, the assignment of a client to exactly one group results in the highest diversity. The assignment of all clients to all groups results in the lowest diversity.

To better illustrate the group diversity defined for our test case, Tables 6.1 and 6.2 are presented for the case where the group size is 2 and there are 5 groups. The X in the table marks that a client is assigned to a particular group.

In the case of high group diversity, chances of client being in the same group as another client is low. For the case of low group diversity, all the clients are assigned to the same two groups.

Table 6.1: Illustration of high group diversity

		Client				
		a	b	c	d	e
Group	1	x	x			
	2		x	x		
	3			x	x	
	4				x	x
	5	x				x

Table 6.2: Illustration of low group diversity

		Client				
		a	b	c	d	e
Group	1	x	x			
	2	x	x			
	3	x	x			
	4	x	x			
	5	x	x			

Based on Figure 6.4, it can be seen that there are no effects on group diversity. This is as expected as the number of client requests being processed are the same. In a client's view, it needs to perform two group requests. Each client's request is independent of other clients' group requests. Unless there's optimization related to client lookup, there should also be no difference on the server side. Hence, there should not be a correlation between the two.

6.2.5 Summary

We make two observations from the graphs. One is that both KM_1 and KM_2 scale linearly with the number of clients, groups, group size and group diversity. This is promising for both approaches and suggests that they will scale well for large deployments. This linear scaling was fully expected. One way to reason about it is to examine the number of keys that the protocol

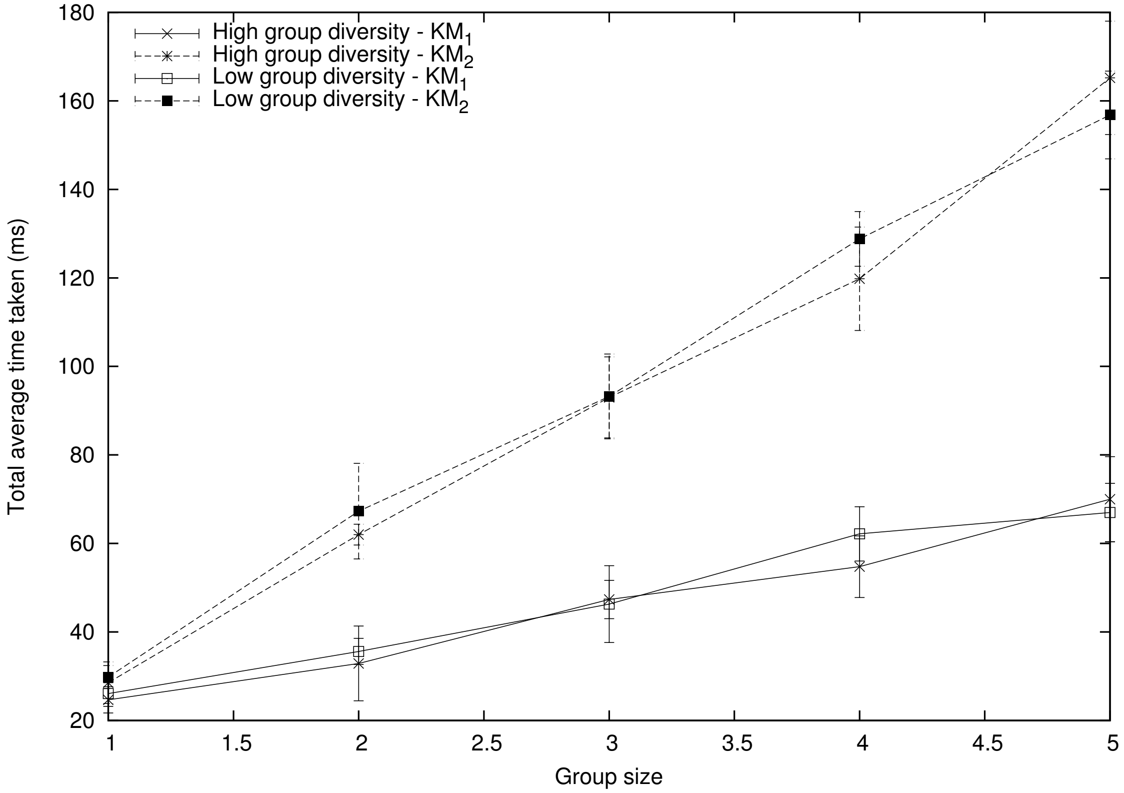


Figure 6.4: Average total time to complete client requests with varying group assignment with fixed number of clients and groups.

infrastructure must manage and transport. In each case, the number of keys increases linearly with each of the inputs.

The number of keys can also be used to reason about the second observation. The second observation is that KM_1 consistently outperforms KM_2. This can be seen as a trade-off between security and performance that is inherent to KM_1 and KM_2. In particular, KM_1 has the possible weakness that was discussed in Section 4.3 that KM_2 does not have.

The implementation of Kerberos that was adopted [26] has a somewhat sophisticated caching mechanism. Repeated requests for the same key do not necessarily result in the same time overhead, as the key may be in the cache. Our implementation of the $GCKS$ is somewhat rudimentary in that while it supports all the protocols as needed, it does not have a cache similar to Kerberos.

While we argue that our empirical results provide meaningful insights about the scaling properties of KM$_1$ and KM$_2$, and their relative performance characteristics, It should be pointed out that such implementation issues can significantly affect the results. A more comprehensive assessment is beyond the scope of this document.

Chapter 7

Related Work

We have explicitly incorporated Kerberos [29] and MIKEY [1] in our work. Therefore, those pieces of work are closely related to ours. As we discuss them elsewhere in this document, we do not discuss them further here. Also, we use PCL [12, 15, 32, 33] for our analysis. Consequently, that work is related as well. We provide a limited discussion on PCL as needed elsewhere in our document. Also, as our work does not propose new techniques for such analysis, we do not discuss other approaches from the literature for it.

The motivation for us to consider the composition of Kerberos and MIKEY is the transport of keys for secure group communication. There has been considerable work on authenticated key agreement and distribution or transfer. Work that does not specifically address group keys, such as those of Diffie et al. [14] and Bellare and Rogaway [7] is related to ours mostly in the authentication part. We adopt Kerberos for authentication as it is the de facto standard in the settings we consider.

There is also work that is focussed on group keys. Key agreement schemes such as the work of Harn and Lin [17] are not similar to our work as they require group participants to contribute to the key. We assume that the enterprise wants control over the generation of keys. There is also some work from IETF MSec that proposes new protocols for group key establishment for secure multicast. Such work is different from ours in that they are designed “from scratch.” Our work is on reusing an existing de facto and a proposed standard. Furthermore, to our knowledge, unlike our work, no rigorous analysis has been done for such prior work. Examples of such work are the Group Domain of Interpretation (GDOI) [5] and the Group Security Association and Key Management Protocol [18]

The work of Jordan and Medina [21], is somewhat closer to ours. That work uses Kerberos to establish group keys. We argue that the use of “pure” Kerberos, and not Kerberos per se is less practical than what we have done. Our work is backward-compatible in settings in which Kerberos is already used for authentication.

The work of Crescenzo and Kornievskaia [11] addresses secure multicast and Kerberos. However, their focus is on what is called the cross-realm scenario, in which the main problem is that of an enterprise authorizing principals from another enterprise to multicast groups. While this problem is certainly interesting, it is different from the problem that we address. Also, they do not leverage MIKEY as we do.

There is also work on key management in the context of secure group communication. Rafaeli and Hutchison [31] survey such work; an example is the work of Balenson et al. [3]. Such work is focussed on how the communication and key overhead can be reduced for efficient rekeying. We do not address such issues in our work, and deal only with the initial transport of the group key.

There has also been work on what is called “Kerberizing” applications and protocols that is somewhat related to ours. The work of Mattsson and Tian [24], for example, Kerberizes MIKEY by introducing new modes of operation for MIKEY. These new incorporate a ticket mechanism similar to that of Kerberos. Their technique can potentially be used for authenticated group key transport. However, they do not leverage an existing Kerberos infrastructure like we do.

Other examples of such work are those of Hildrum [19], which addresses Kerberized rlogin, and [36], which is a deployment of Kerberized ftp. Rather than Kerberizing a single application, our work can be seen as Kerberizing a class of applications – group communication.

In this respect, our work is similar to the work on Kerberized Internet Negotiation of Keys (KINK) [34]. The idea there, as the name suggests, is an approach based on Kerberos for key agreement for IPSec. As such, it is more similar to key agreement for IPSec, such as IKE [22]. Their focus is not group keys. However, the existence of their work suggests that our perspective on Kerberizing the transport of group keys is well-motivated.

Chapter 8

Conclusions and Future Work

8.1 Future Work

8.1.1 Empirical Analysis

There are a number of topics for future work in this context. One is a more comprehensive empirical evaluation of our implementation, and a deeper study on how the GCKS can be implemented to be faithful to IETF MSec, and bound more tightly to the components of Kerberos to make the overall system more efficient. In particular, the experience and longevity of Kerberos implementations may offer useful lessons in building portions of the GCKS.

8.1.2 PCL Analysis

There has been research on using theorem solvers to prove PCL assertions [2]. Unfortunately, the source provided was outdated. The PCL axioms definitions needs to be updated to be compatible with the latest syntax of Isabelle. This work was left outside the scope of this document.

However, if more resources are provided, leveraging theorem solvers can provide a concrete validation and contribution to the analysis. More lemmas and assertions can be proven once the protocol is formally defined.

8.1.3 Group Rekeying

The Rekeying process provides a mean to support groups with dynamic members, as specified by IETF MSec [16]. Its purpose is to maintain key freshness when security policy adjustments may occur or if members leave or join the session. Although MIKEY does not fully provide a mechanism for rekeying, it can be achieved by issuing a new MIKEY I_MESSAGE (as suggested by its RFC [1]). However, this would require unicast communication from the GCKS to each active client. There will be a linear relationship between the cost of rekey and number of clients per group.

Further analysis is needed to determine the cost and benefit in allowing such a feature. For scenarios where the group sessions are relatively short, then rekeying may not be necessary. Same applies if the group members are relatively static for each session.

8.1.4 Authenticated Cross-Realm Group Communication

As menionted before in Section 7, the problem of Cross-Realm Authentication is very different from the issue addressed by this document. However, since our design adopts Kerberos, it might be interesting to see its effects in the case for group communication. In the context of group communication with Cross-Realm Authentication, one can imagine the interesting problem of allowing authenticated Kerberos clients to communicate with clients from other realms.

For KM_1, this can be achieved if the GCKS is a service principal under multiple KDC. Two authenticated Kerberos clients from different realms can be authenticating to the same ”multi-realm GCKS” that maps each client to the same group. This method bypasses the traditional model with Cross-Realm KDCs requiring shared keys[29]. One might evaluate the possible security threat this might impose. However, the same arguments can be applied to whether the KDC should have authority over all group management as discussed in Section 5.5.

For KM_2, the GCKS is tightly bound to the KDC so a trivial solution is not apparent. This will be left as future work.

8.1.5 Alternative Methods

Another topic for future work regards other designs that may be viable alternatives to KM_1 and KM_2, and a study of the properties that those new designs have that may be superior to our designs. We allude to one such design in Section 3.2. Of course, as is our mindset in this work, any new design must also be rigorously analyzed as part of the future work.

8.2 Conclusions

We have proposed two designs, KM_1 and KM_2, for bootstrapping secure multicast with Kerberos by composing it with MIKEY [1]. Our motivation is the authenticated transport of group keys in environments that already deploy Kerberos for authentication. Indeed, Kerberos is the de facto standard for authentication in enterprise and public-safety settings. Our choice of protocol for key transport is MIKEY, which is an IETF proposed standard. One of the explicit intents of MIKEY is for it to be usable in the registration protocol of IETF MSec.

We have rigorously analyzed KM_1 and KM_2 using PCL. We have pointed out that the analysis was valuable to us — it revealed a possible weakness in KM_1, and it is the discovery of this weakness that lead us to design KM_2. We have also discussed an implementation of KM_1 and KM_2 that we have carried out, starting with the open-source reference implementation of Kerberos, and an open-source implementation of MIKEY. We have presented empirical results based on our implementation that demonstrate that our designs scale well with the number of clients, groups, group size and group diversity. They also demonstrate the inherent trade-offs between security and performance in KM_1 and KM_2.

References

[1] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman. MIKEY: Multimedia Internet KEYing. RFC 3830, August 2004. Available from `http://tools.ietf.org/html/rfc3830`.

[2] D. Auerbach, C. Kempston, A. Datta, A. Derek, and J. C. Mitchell. Isabelle Implementation of Protocol Composition Logic, October 2004. Available from `http://www.cis.upenn.edu/~spyce/oct04/posters/anupam_Isabelle_PCL.pdf`.

[3] D. Balenson, D. McGrew, and A. Sherman. Key management for large dynamic groups: One-way function trees and amortized initialization, 2000.

[4] M. Baugher, R. Canetti, L. Dondeti, and F. Lindholm. Multicast Security (MSEC) Group Key Management Architecture. RFC 4046 (Informational), April 2005.

[5] M. Baugher, B. Weis, T. Hardjono, and H. Harney. The Group Domain of Interpretation, July 2003. Available from `http://tools.ietf.org/html/rfc3547`.

[6] Giampaolo Bella and Lawrence C. Paulson. Kerberos version 4: Inductive analysis of the secrecy goals. In *Proceedings of the 5th European Symposium on Research in Computer Security*, pages 361–375, London, UK, 1998. Springer-Verlag.

[7] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, pages 232–249, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[8] Frederick Butler, Iliano Cervesato, Aaron D. Jaggard, Andre Scedrov, and Christopher Walstad. Formal analysis of kerberos 5. *Theor. Comput. Sci.*, 367:57–87, November 2006.

[9] Ran Canetti. Security and composition of cryptographic protocols: A tutorial. Cryptology ePrint Archive, Report 2006/465, 2006. `http://eprint.iacr.org/`.

[10] Veronique Cortier and Bogdan Warinschi. A composable computational soundness notion. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 63–74, New York, NY, USA, 2011. ACM.

[11] Giovanni Di Crescenzo and Olga Kornievskaia. Efficient kerberized multicast in a practical distributed setting. In *Proceedings of the 4th International Conference on Information Security*, ISC '01, pages 27–45, London, UK, UK, 2001. Springer-Verlag.

[12] Anupam Datta, Ante Derek, John C. Mitchell, and Arnab Roy. Protocol Composition Logic (PCL). *Electronic Notes in Theoretical Computer Science*, 172:311–358, April 2007.

[13] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Commun. ACM*, 24:533–536, August 1981.

[14] Whitfield Diffie, Paul C. Van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptography*, 2:107–125, June 1992.

[15] Nancy Durgin, John Mitchell, and Dusko Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11(4):677–721, 2003. Available from `http://seclab.stanford.edu/pcl/papers/dmp-jcs03.pdf`.

[16] T. Hardjono and B. Weis. The Multicast Group Security Architecture. RFC 3740, March 2004. Available from http://tools.ietf.org/html/rfc3740.

[17] Lein Harn and Changlu Lin. Authenticated group key transfer protocol based on secret sharing. *IEEE Transactions on Computers*, 59:842–846, 2010.

[18] H. Harney, A. Colegrove, and G. Gross. GSAKMP: Group Secure Association Key Management Protocol, June 2006. Available from http://tools.ietf.org/html/rfc4535.

[19] Kirsten Hildrum. Security of encrypted rlogin connections created with Kerberos IV. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2000.

[20] IETF. Multicast security (msec), September 2011. http://datatracker.ietf.org/wg/msec/charter/.

[21] Francisco Jordan and Manuel Medina. A complete secure transport service in the internet. In *PROCEEDINGS INTERNET SOCIETY SYMPOSIUM ON NETWORK AND DISTRIBUTED SYSTEM SECURITY*, pages 67–76, 1993.

[22] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet Key Exchange Protocol Version 2 (IKEv2), September 2010. Available from http://tools.ietf.org/html/rfc5996.

[23] Kerberos Consortium. Why is Kerberos a credible security solution?, 2008. Available from www.kerberos.org/software/whykerberos.pdf.

[24] J. Mattsson and T. Tian. MIKEY-TICKET: Ticket-Based Modes of Key Distribution in Multimedia Internet KEYing (MIKEY). RFC 6043, mar 2011. Available from http://tools.ietf.org/html/rfc6043.

[25] Minisip. MiniSIP, February 2011. http://www.minisip.org/.

[26] MIT Kerberos Team. MITKeberos: Kerberos – The Network Authentication Protocol, February 2011. http://web.mit.edu/Kerberos/.

[27] MIT Kerberos Team. MITKerberos: Admin Guide, February 2011. http://web.mit.edu/Kerberos/krb5-1.8/krb5-1.8.3/doc/krb5-admin.html.

[28] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21:993–999, December 1978.

[29] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120, July 2005. Available from http://tools.ietf.org/html/rfc4120.

[30] J. Orrblad. Alternatives to mikey/srtp to secure voip. Master's thesis, KTH Microelectronics and Information Technology, 2005.

[31] Sandro Rafaeli and David Hutchison. A survey of key management for secure group communication. *ACM Comput. Surv.*, 35:309–329, September 2003.

[32] Arnab Roy, Anupam Datta, Ante Derek, and John C. Mitchell. Secrecy analysis in protocol composition logic. In *Proceedings of 11th Annual Asian Computing Science Conference*, 2006.

[33] Arnab Roy, Anupam Datta, Ante Derek, John C. Mitchell, and Jean-Pierre Siefert. Secrecy analysis in protocol composition logic. Technical report, Carnegie Mellon University, 2006. Available from http://www.andrew.cmu.edu/user/danupam/secrecy-pcl-mbdf.pdf.

[34] S. Sakane, K. Kamada, M. Thomas, and J. Vilhuber. Kerberized Internet Negotiation of Keys (KINK). RFC 4430, March 2006. Available from http://tools.ietf.org/html/rfc4430.

[35] J.H. Saltzer and M.D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278 – 1308, sept. 1975.

[36] MIT Information Services and Technology. Kerberized File Transfer Protocol (FTP) at MIT, September 2011. http://ist.mit.edu/services/software/filetransfer/ftp.

[37] Jeffrey Woo. Kerberized multimedia internet keying, March 2012. https://github.com/jltwoo/Kerberized-Multimedia-Internet-Keying.

[38] L. Zhu, P. Leach, and K. Jaganathan. Kerberos Cryptosystem Negotiation Extension. RFC 4537 (Proposed Standard), June 2006.