LONG-LIVED AUTHENTICATION PROTOCOLS FOR CRITICAL INFRASTRUCTURE PROCESS CONTROL SYSTEMS

By

RASIKA MUDUMBAI CHAKRAVARTHY

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Washington State University

School of Electrical Engineering and
Computer Science

May 2009

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of RASIKA MUDUMBAI CHAKRAVARTHY find it satisfactory and recommend that it be accepted.

_____
Carl H. Hauser (Chair)

_____
David E. Bakken

_____
Min Sik Kim

ACKOWLEDGEMENT

I am extremely grateful to my advisor Dr. Carl Hauser from whom I have learnt more than just Computer Science. I thank him for his support and guidance. I thank Dr. David Bakken for his guidance to constantly improve my work. I thank Dr. Min Sik Kim for his advice and for taking the time to be on my thesis committee.

I thank WSU, NSF (CNS 05-24695 (CT-CS: Trustworthy Cyber Infrastructure for the Power Grid(TCIP)) and PNNL (The U.S. Department of Energy, Office of Electricity Delivery, via subcontract 49944 with Pacific Northwest National Laboratory) for providing financial support during my studies which would not have been possible otherwise. I thank Loren Hoffman for helping me numerous times with questions related to GridStat. I also thank Dave Anderson for taking time to help me out technically.

I thank all my teachers so far in my life for what I am today. I thank Dr. Diane Cook and Dr. Larry Holder for teaching two of the best courses I have ever taken. I thank Mrs. Daphne Lopez for her great faith in me, Mr. Saleem Durai for helping to make this degree possible. I thank Sr. Anna and Ms. Irene for all their inspiration and guidance. I thank Mr. Jagadeesan for helping me take my first steps towards Computer Science. I thank Mrs. Saleena Williams for expecting the best.

Any amount of thanking will not suffice for my mom's love and patience, my dad's generosity, my grandparents' love. I thank my sister for being my first role model, my brother for setting high standards. I thank Murugan uncle, Shanthi aunty and family for all the educational material and support. I thank Mui, Jin, Aishu for their encouragement and giving me another home. I thank all my friends for being supportive in my endeavors so far.

PUBLICATIONS

Erik Solum, Carl Hauser, Dave Bakken, and **Rasika Chakravarthy**: Modular Over-The-Wire security in Managed Publish-Subscribe Systems: Submitted to International Conference on Distributed Event Based Systems (DEBS'09).

LONG-LIVED AUTHENTICATION PROTOCOLS FOR CRITICAL INFRASTRUCTURE PROCESS CONTROL SYSTEMS

ABSTRACT

by Rasika Mudumbai Chakravarthy, M.S.
Washington State University
May 2009

Chair: Carl H. Hauser

Process Control Systems monitor and control processes that manage critical infrastructure systems. To enable these systems to continue working uninterrupted, situational awareness is imperative. Communication systems that provide situational awareness pose challenges such as low latency, high availability and security. Being a modern life supporting system, critical infrastructures such as electric power grids have very high risk and may potentially result in enormous economic and social impact if attacked.

Authentication is the fundamental step towards security. The goals of other security services can be attained only if they are based on successful authentication. This thesis presents an authentication framework that authenticates nodes to ensure that they are genuine. A set of authentication protocols that employ authentication modules that can be changed at runtime

to support long-lived systems such as Process Control Systems are introduced in this work. The protocols use a pre-loaded key set as identification material. The pre-loaded key set is used minimally and only for authentication purposes. Keys for encryption are generated and exchanged between authenticated nodes thus enabling the security architecture to function longer.

GridStat, a publish/subscribe middleware system offers mechanisms that enable low latency and high availability for operational data delivery. GridStat has been designed specifically to improve situational awareness in the electric power grid. GridStat Security Management System protects the data exchanged using GridStat with confidentiality, integrity and availability. These capabilities would be able to serve their purpose only if authentic GridStat entities are exchanging the information being protected. This thesis presents the authentication framework and protocols in the context of GridStat.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Dedication

To Johnny for guiding me at every step.

CHAPTER ONE

INTRODUCTION

Critical infrastructures (CIs) such as electric power grids, water infrastructures, and oil and gas pipelines make use of technological advances in computer networks to enable timely and reliable information exchange. The Process Control System (PCS) that manage these critical infrastructures face unique demands. PCSs have to be available continuously without being shut down completely, they may have nodes which once deployed cannot be easily accessed and they need to be functional for a long lifetime. Because CIs consist of elements operated by many companies, their PCS must provide both information isolation and information sharing. Since the loss due to disruption of a CI is potentially huge and since CIs are under threat ranging from extortion to terrorism, it is necessary to keep the security services up to current standards over many years of deployment (Markus and Martin 2008).

Advances in the security field are rapid and techniques trusted only a few years ago may be considered broken today. Thus, security services must be able to dynamically accommodate new advances. Among security services, authentication is the first step to initiate secure communication. In this research an authentication protocol that can meet the security requirements of CIs over

long periods of time has been designed. The protocol leverages pre-shared key information to allow secure replacement of security algorithms multiple times over the life of the system. The protocol has been designed for GridStat, which is a PCS communication system for the electric power grid.

In this chapter PCS security issues and GridStat are introduced. A brief discussion on how the protocols designed in this thesis are evaluated is presented. The goals and challenges of this thesis and research contributions are also presented.

1.1 Security in Long-Lived Process Control Systems

In a Critical Infrastructure Process Control System, data is the fundamental element and system security is necessary to protect data. The reliable operation of the entire system depends on this. The consequences of security violations in PCS include potentially high economic damage, safety impacts to the general public in addition to physical impacts (Jason Stamp, John Dillinger et al. 2003). Given the extent of damage and the importance of data in PCSs, authentication protocols need to be carefully designed to lay a strong foundation for security.

Communication Systems for long-lived PCS have a specific challenge due to their longevity. On one hand the security field is constantly evolving, replacing trusted techniques quickly. Algorithms may become easier to break with new advances in technology and computing power. At the same time, newer and stronger algorithms may be introduced to remedy the flaws in the older ones. On the other hand, the communication system component of a long-lived PCS has to last a long time. Using an obsolete security system in a PCS is certainly disadvantageous. Hence the challenge in this context is that the PCS communication system has to last a long time and it has to keep up with the changes in the security field.

The authentication framework presented in this thesis accommodates the long life time and also the evolution of security. The protocols presented in this thesis sustain the key material for a longer period of time and facilitate replacing old algorithms with newer ones in a safe manner.

1.2 GridStat

Process Control Systems for Critical Infrastructures collect thousands of measurements either to take automated action or to present them to human operators (Markus and Martin 2008). This is done for situational awareness in

a PCS. Poor situational awareness has been one of the contributing causes to the major blackouts in the US (U.S.-Canada 2004).

GridStat is a new kind of middleware framework that supports what are known as Periodically Updated Variables (PUVs). The PUVs are used to provide better situational awareness as they represent source variables and various remote caches where PUV values can be retrieved. A PUV is implemented by a Periodically Updated Flow (PUF) abstraction which is a flow that feeds into a PUV cache (Dave Bakken, Carl Hauser et al. 2009).

GridStat components are organized into the Data Plane (DP), Management Plane (MP) and the Security Management Plane (SMP). Each of them addresses specific functionality to support timely and efficient information exchange. The following sections provide an overview of these planes and their functionalities.

1.2.1 Data Plane

GridStat's data plane consists of publishers, subscribers and forwarding elements (FEs) called status routers (SRs) that facilitate PUFs to feed data into PUVs. The publishers and subscribers use a network of SRs to support PUFs from publishers to subscribers. Though SRs are involved in routing messages from publishers to subscribers, they use static routing. Given the relatively

static and completely planned power grid environment, it is possible to employ static routing. The routers merely forward the packets and do not have to update their routing tables often.

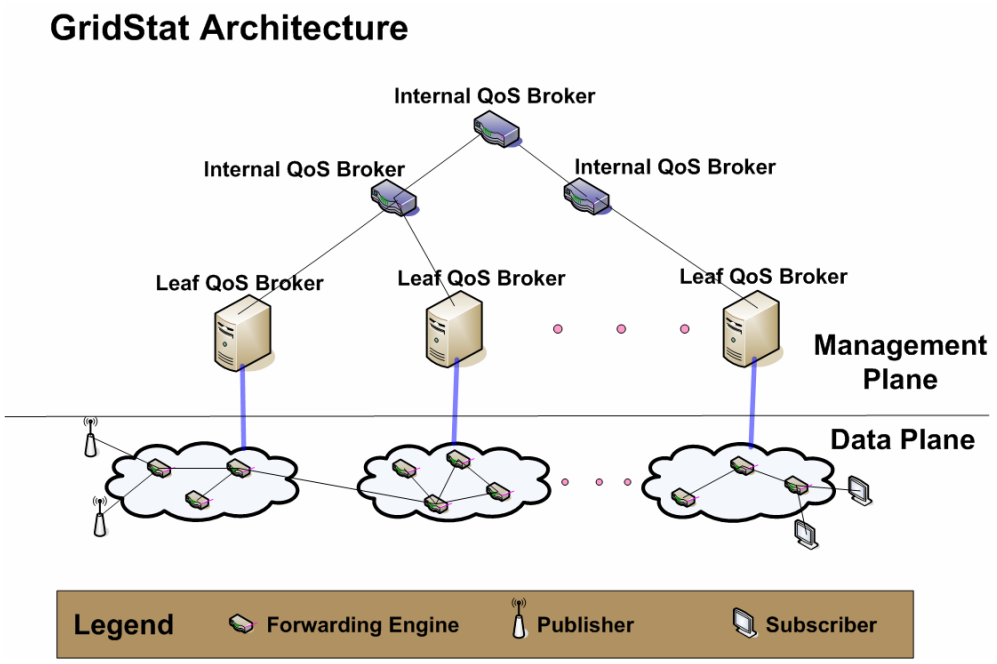

Figure 1.1 GridStat Architecture

Apart from forwarding the SRs are also responsible for rate filtering and multicast. Subscribers may request for different rate of updates and rate filtering facilitates this. Multicast is also efficiently supported by never repeating an update on a link. To ensure that updates reach the PUV caches, they are sent over multiple different paths thus employing redundancy to improve availability (Dave Bakken, Carl Hauser et al. 2009). Figure 1.1 shows

GridStat Architecture depicting the data plane and management plane components.

1.2.2 Management Plane

GridStat's management plane consists of quality of service brokers (QoS brokers) that are responsible for control decisions. Among their most important responsibilities are granting or denying subscription and publication requests, configuring the forwarding elements' forwarding tables and ensuring that QoS requirements are met.

The QoS brokers are organized in a hierarchy which reflects the organizational control levels in the electric power grid. The leaf nodes in the QoS broker hierarchy are the leaf QoS brokers. Leaf QoS brokers have jurisdiction over a cloud of data plane nodes. The leaf QoS brokers interact with other internal QoS brokers to aid decision making in case of inter-cloud publications/subscriptions. When publishers need to publish a variable, they contact their own leaf QoS broker that verifies if the publication can be supported. If the publication can be supported the Leaf QoS broker registers it. Similarly, subscribers contact their leaf QoS brokers for subscription requests and the leaf QoS broker grants or rejects a request according to subscriber permissions and QoS requirements.

GridStat uses a publish/subscribe communication model. In typical publish/subscribe systems, the management and routing operations are performed by the same node. But in GridStat the QoS brokers are free of filtering and store-and-forward responsibilities. This design decision is due to the fact that the information exchange in critical infrastructures have to be QoS compliant (quick and reliable usually). Though the QoS brokers do not participate directly in the information exchange, they have management responsibilities. The responsibilities of the QoS brokers include computing routing tables for the Status Routers and processing subscribe/unsubscribe requests. The QoS brokers coordinate with each other to allocate paths to be followed by the SRs (computation of the routing tables) while forwarding updates.

1.2.3 Security Management Plane

The Security Management Plane (SMP) consists of Security Management Servers (SMS) that control the security of communication in GridStat. The SMS are organized in a hierarchical fashion like the management plane. SMP controls the security policies followed by the communicating nodes in GridStat. The SMP also controls and secures the communication that takes place among the SMSs.

Each pair of parent and child nodes communicating with each other has its own key set that is pre-loaded (loaded into the system at the time when the child node is introduced in the system). This key set is used in combination with the policy decided by the SMP to communicate securely. SMP facilitates group communication by providing the data plane nodes with group policies. SMS nodes also follow policies to communicate with their parent or child SMS nodes. With its introduction in (Solum 2007), SMP provides services such as encryption and integrity for securing the DP communication and SMP communication.

The SMP uses symmetric key cryptography. Symmetric key cryptography uses the same keys for encryption and decryption. If not the same key, one key would be trivially derivable from the other. The alternative to symmetric key cryptography is public key cryptography, where the keys used for encryption and decryption are different (Smith and Marchesini 2007).

In case of public keys, the keys are expected to have a particular mathematical structure according to the algorithm used. For instance RSA requires the public keys to be a product of two prime numbers. So, a key that works for one algorithm that employs public key cryptography may not be suitable for another. However, in case of symmetric key cryptographic algorithms, no such structure is expected out of the keys (Schneier 1996). The keys are merely a

sequence of random bits that can be utilized with any symmetric key cryptography algorithm.

Hence, given a symmetric key it is possible to change the algorithm used with any key in the pre-loaded key set. This property of symmetric key cryptography is extremely important as this is used to change algorithms over a pre-distributed set of keys. The same cannot be done with public keys because to change the algorithm being used the keys being used have to be changed too since the keys need to exhibit different properties for different algorithms.

Symmetric key algorithms may require specific key sizes or may have different key size ranges. However, they are still a stream of bits. It is worth noting that the security of the cryptosystem depends on its key length. Again for a given key size the strength of the cryptosystem differs from one algorithm to another (Schneier 1996).

1.3 Protocol Verification

It is important to ensure that the authentication protocols function correctly. This thesis presents a formal analysis of the protocols using the BAN logic

(Michael Burrows, Martin Abadi et al. 1990). The authentication protocols support replacement of security algorithms with new ones. However, the BAN logic does not have a method to represent modules and reason about them. This thesis presents a way to represent modules in the BAN logic, to reason about the modules and to express beliefs about modules.

A protocol to switch safely from one key to another is also presented in this thesis. Since BAN logic does not define specific goals for this type of protocol, a set of goals to be verified for a key switch protocol are presented.

1.4 Goals and Challenges in Designing an Authentication Framework for GridStat

The goal of this thesis is to develop an authentication framework that can be incorporated into the Data Plane Security Architecture. The framework should provide authentication protocols that will authenticate GridStat components such as publishers and subscribers to the security management plane components and vice versa. The protocols should also authenticate the SMP components to each other.

In GridStat some nodes are unmanned and need to be managed remotely. It is a challenge to ensure that remote configurations and module changes take place safely. The modules for authentication need to be dynamically and securely distributed to GridStat nodes. An authentication framework where algorithms can be changed remotely during the course of operation allows authentication protocols to make use of newer technological advances in the security field while supporting unmanned nodes.

Keys for encryption and for authentication need to be organized and maintained in such a way that data plane security architecture and its features are still supported. At the same time, the pre-loaded key set should be used minimally to sustain the system as long as possible within reasonable limits. Since the pre-loaded keys are the identification material, the protocols should ensure that two participating nodes maintain the same beliefs in the pre-loaded key set during the operation of the system.

There is a hierarchy of control represented by the QoS broker hierarchy in GridStat. The authentication protocols need to adhere to this level of control and maintain it for authentication. They should also support the presence of multiple private companies and their interest in preserving market sensitive information. The authentication protocols presented in this thesis extend

GridStat's security architecture to authenticate the important entities in data exchange while meeting these requirements.

1.5 Research Contributions

The research contributions of this thesis are:

- Design and implementation of long-lived modular authentication protocols for Critical Infrastructure Process Control Systems
- Design and implementation of protocols that allow safe distribution of security modules, safe change and distribution of keys over an unsafe connection
- Analytical evaluation of the authentication, key switch and module change protocols
- Introduction of modules into BAN logic to facilitate reasoning about modules

CHAPTER TWO
SURVEY OF SECURITY ARCHITECTURES

In this chapter, overviews of security architectures in publish-subscribe systems and some security standards are presented. The security architecture used in GridStat is discussed. To begin with, the goals of security services (services provided by security architectures) and the common approaches to address them are explored.

Security architectures are required to provide all or some of the following security services: authentication, access control, confidentiality, integrity and non-repudiation (Mogollon 2007). In addition to these, some publish-subscribe systems are required to support accountability (Ying Liu and Plale).

Each of these security services are discussed, focusing on their goals and the common approaches to provide them, individually. Since GridStat is a communication system for Process Control Systems (PCS), PCS Security requirements are also presented. Security standards that are candidates for PCS security are also presented.

2.1 Security Services

2.1.1 Authentication

The goals of an authentication service is to provide entity authentication and message authentication (Armin 1993). While entity authentication establishes an entity's identity, message authentication ensures that message content is un-tampered and message origin is an authenticated entity (Chenxi Wang, Carzaniga et al. 2002; Mogollon 2007).

An entity authentication service defines a sequence of steps in which an entity is challenged based on a secret. The entity wishing to establish the identity of another entity should be able to challenge it and verify its response. To verify the response, both entities need to have some common knowledge. In case of symmetric key cryptography, the common knowledge is based on secret keys which not known to any other entity. In case of public key cryptography, the common knowledge is the public keys that may be known to other entities. Using the public keys and private keys (that are known only to the entity to which it belongs) it is possible to verify an entity's response to a challenge.

An authentication protocol is typically executed when two entities initiate conversation. It may also be executed at other times depending on the application scenario. Hence the very first step to providing useful security

services is to perform authentication. Once authenticated, the participating entities can use other security services to securely communicate with each other.

The goal of the authentication protocols is to check whether the other entity possesses the secret. While checking, it is also necessary that the secret is not revealed to any other entity. This could be done in a challenge-response type of exchange until the challenger is satisfied about the other entity's identity. There are many authentication protocols, for instance Challenge Handshake Protocol (CHAP) (Simpson 1996) and S/Key (Haller 1994).

In CHAP, the participating entities can challenge each other regularly to check their possession of the shared secret. The proof of possession of the secret proves their authenticity. The concept behind S/Key is that it is difficult to reverse cryptographic hash functions. In S/Key a one-time password is used for authentication, that is every time, a different secret is used. A single secret is used to generate the passwords for S/Key. A cryptographic hash function is applied n times on this secret. So it results in a list of n one-time passwords. The user uses these passwords in the reverse order, i.e the n_{th} password is used first, then $n\text{-}1_{th}$ password and so on. To respond to a challenge, the entity performs a hash function on the challenge based on the secret that it possesses (Haller 1994). S/Key is not safe to be used on its own as it is vulnerable to

man in the middle attacks, some race conditions and pre-play attacks (A. Menezes, P. van Oorschot et al. 1996).

There are variations in the way entities are challenged too. The common goal is to reveal as little as possible about the secret. For example, in a Zero-knowledge proof challenge, the goal is to reveal no knowledge about the secret even over time. One way functions are easy to compute but difficult to reverse. An authentication protocol can hence use a combination of protocols, secrets and challenges.

2.1.2 Access Control

The process of restricting access is access control. The access control service is intended to perform this role of permitting or denying access of resources to entities. To make this decision, the access control service has to follow the policy of the system (Smith and Marchesini 2007).

Any entity requesting access to some resource has to be authenticated first. If the system policy is consulted without authenticating the entity first, it can be exploited to perform a guessing attack. The attacker could impersonate any entity (since authentication is not done first), and gather information about the resources to which the entity has access. Depending on the application revealing such information can be expensive. It is meaningful to perform

authentication first and then grant (or deny) access to resources (Charlie Kaufman 2002).

In case of a distributed system, permission has to be granted for sending and receiving data. Only authenticated receivers who are permitted to access information should be able to access it. Similarly, only an authenticated entity that has permission to send information should be able to send messages. The former is receiver access control and the latter sender access control. Commonly used approaches for access control include role based access control and access control lists. Trust based systems can also be used to provide access control (Seamons, Winslett et al. 2002; Dionysiou 2006).

2.1.3 Confidentiality

When two entities communicate, their communication is confidential if other entities or intruders cannot comprehend it. To understand the conversation, some special information or secret is necessary. Without this secret, it is very difficult to decipher what is being conveyed. The degree of difficulty of deciphering what is conveyed is such that the effort spent in retrieving the encrypted information is much greater than the information's worth. Confidentiality is a security service that ensures that the communication is understood by none other than the authentic intended entities.

Just as in the case of authentication, some secret is known only by the participating entities. Secret key cryptography and public key cryptography are the standard approaches for achieving confidentiality. In a publish/subscribe system, the goals of confidentiality include information confidentiality, subscription confidentiality and publication confidentiality (Chenxi Wang, Carzaniga et al. 2002). Information confidentiality i.e. keeping the communication between two entities confidential is what has been described previously in this section.

The goal of subscription confidentiality is that other entities do not know the subscription details of a subscriber. Other entities do not know what publications a subscriber has subscribed to. This is to ensure that sensitive information about the subscriber is not revealed. Similarly, publication confidentiality is achieved when entities who are not legitimate subscribers, do not know what topics (subjects) are being published by various publishers. This is to make sure that sensitive information about the publisher is not revealed (Chenxi Wang, Carzaniga et al. 2002).

Many systems use confidentiality service to secure their communication. However, not all systems authenticate the entities prior to securing their communication. For example, very often, only the server is authenticated in Transport Layer Security (discussed later). Though there may be many

applications where only the server needs to be authenticated, it does not make sense to participate in confidential conversation when only one of the entities is authenticated or when neither entities are authenticated due to the following reasons. Firstly, sending encrypted information to an unauthenticated entity is wasteful as resources are spent on sending information to the wrong entity. Secondly, the actual target will not receive this information. Thirdly, an intruder can almost effortlessly access information by directly participating in confidential information exchange with one-sided or no authentication (Charlie Kaufman 2002). For confidentiality to be purposeful, it is necessary that the entities are authenticated first.

2.1.4 Integrity

Integrity is defined as the condition of being tamper-free. When a message has not been tampered with, and it reaches the receiver as it was sent, then message integrity is maintained. Integrity goals in publish-subscribe systems include subscription integrity and service integrity, apart from message integrity. Subscription integrity is achieved when the access control information of subscribers is not maliciously altered. Service integrity ensures that the infrastructure level components are not mimicked (Chenxi Wang, Carzaniga et al. 2002).

Integrity is usually achieved by using signature based algorithms. Message integrity is a communication issue because message integrity is lost when a message is tampered with in transit from one entity to another. Service integrity and subscription integrity are node level issues because they can be tampered where they are stored in the entity that controls service and subscription integrity.

2.1.5 Non-repudiation

Non-repudiation means that the origin of data can be proved. So an entity that once sent data should not be able to deny that it sent the data at a later point of time. It should not be possible for an entity to maliciously impersonate another entity and still prove the origin. Digital signatures and hashing are used to assert the origin of data (R. Housley, W. Ford et al. 1999). Digital signatures are assumed to be unique to entities. If this assumption is maintained at the protocol level, it can be used to prove (or disprove) that data originated from that entity.

2.1.6 Accountability

Some publish-subscribe systems may need to keep track of the information that each subscriber receives. This may be for charging them for the subscriptions or for other book-keeping purposes. When a subscriber registers

a new subscription, it is issued a key to decrypt the published data. So, this key can be purchased by the subscriber, thus being accountable for information subscribed to. If payment is not required, the subscriptions can still be kept track of when they are registered (Chenxi Wang, Carzaniga et al. 2002).

2.2 Process Control System Security

GridStat has been specifically designed as a Process Control System (Dave Bakken, Carl Hauser et al. 2009) for the electrical power grid. To support this, the requirements of GridStat's security architecture include PCS security system goals in addition to general security architecture goals and publish-subscribe security goals.

Process control systems prioritize availability more than confidentiality or integrity. This is due to the inherent nature of critical infrastructure systems. For instance, consider a power company. Two examples of its goals would be, power outage avoidance and secrecy of power consumption levels in a town. The company would prioritize power outage avoidance more. This is because, the service is essential for daily life and it is also great revenue loss if this is not taken care of. It is hence obvious that critical infrastructure systems prioritize availability more than security. Yet, the importance of security is

even higher in PCS compared to standard information technology systems because the threat is against essential modern life supporting services (Bessani, Sousa et al. 2008).

In emerging EPInets (Electrical Power Information networks - PCS in power grid scenario), multiple organizational entities - utilities, regulators, independent generators etc., are involved. These organizations wish to protect their business sensitive information from other organizations. The security architecture needs to accommodate inter-operable security services and data-exchange with private information protection.

The security architecture has to meet other specific requirements which result from some EPInet features. EPInets have long life expectancies, possibly several decades. The security architecture will also have to meet similar life expectancies. Since shutting down nodes for maintenance or due to failure is expensive, the security architecture should not depend on shutting down nodes maintenance or updates. It is also the case that in EPInets, the nodes may be in remote locations. Due to this, they may be inaccessible once they are deployed. The security architecture should ensure that once deployed it is expected to be efficient years later, though many nodes may not be physically accessible.

Other characteristics of EPInets which play an important role in designing the security architecture for them include relatively small network size compared to the Internet, completely managed traffic, infrequent topology changes and very fast achievable latency level (Dave Bakken, Carl Hauser et al. 2009). These characteristics result in different requirements for GridStat's security architecture compared to other publish/subscribe systems and other PCSs.

2.3 Publish-Subscribe systems

The communication model of a system influences the way its security architecture is built. It is important to study the communication model to design the security architecture and the services it provides. To this end, some publish-subscribe systems and their security services are discussed.

Content-Based Publish-Subscribe (CBPS) are the most commonly used publish-subscribe flavor (Khurana 2005). Hence, most publish-subscribe systems and security architectures designed for them are for CBPSs (Khurana 2005; Mudhakar and Ling 2005; Mudhakar and Ling 2007). Another common flavor is Topic-based publish-subscribe systems (Shrideep Pallickara, Marlon Pierce et al. 2006). In CBPS, the contents of the messages need to be known to the routers to enable them to route the messages. (Khurana 2005) proposes a

partial message content encryption technique where part of the message is encrypted. The part that the brokers need not know is encrypted and the rest is sent in clear. This may not work for all scenarios is GridStat because the status routers perform multicast, rate filtering (see 1.2.1 above) and other functionalities (such as condensation functions (Dave Bakken, Carl Hauser et al. 2007)) which require them to know the contents of the updates.

Using predicate graphs or hierarchical key derivation algorithms (Mudhakar and Ling 2007), or per-topic keys (Mudhakar and Ling 2005; Shrideep Pallickara, Marlon Pierce et al. 2006) are not applicable for GridStat's communication. This is because, in case of data plane communication, the subscriptions differ not only on the content but also in the frequency in which the updates are sent (see 1.2.1 above). But in case of these approaches, they depend only on the content or topic.

To enable secure communications using encryption, the participating entities need to be provided with secret keys or some methodology to derive secret keys. In case of distribution, the keys have to be distributed to the entities in a secure manner. Third party based key distribution is a commonly used technique. Third parties such as Proxy Security and Accounting Services (PSAS) (Khurana 2005), Key Management Center (Shrideep Pallickara,

Marlon Pierce et al. 2006) and Key Distribution Center (Mudhakar and Ling 2007) serve this purpose.

Before distributing the keys, it is necessary to ensure that the entities are authenticated. The main used for this is Certificate Authorities (CA) (Shrideep Pallickara, Marlon Pierce et al. 2006). However, systems such as (Khurana 2005; Mudhakar and Ling 2005; Mudhakar and Ling 2007) concern themselves about encrypted communication while authentication - the first step towards it, is not discussed. It is futile to apply best possible techniques to encrypt communication if the entities are not authenticated.

2.4 Security standards

Security architectures used in distributed systems, even those with specific needs, use security standards as a guideline or evaluation measure. Security standards are used because they are well tested, avoid redesign and allow interoperability.

Standards can define the system policies, scope of security functions used, techniques to assess ongoing security and also serve as an evaluation criteria to measure the effectiveness of other systems. The services addressed in these

standards include authentication, confidentiality and integrity (Stallings 2007). This section gives an overview of some widely used security standards that are candidates for PCS security architecture. In section 3.2.2, the applicability of these standards are presented in GridStat context.

2.4.1 Public Key Infrastructure

Public Key Infrastructure (PKI) makes use of public key cryptography (Perlman 1999). In PKI, certificates are used to identify individual entities. A digital certificate consists of a digital signature binding a public key to some identity, for instance an IP address. These digital certificates and the private/public key pair are obtained and shared through a trusted authority.

The authority that issues and verifies digital certificates is called a Certificate Authority (CA). The CA hence plays an important role in PKI. There can be multiple CAs working together if necessary. When a CA signs a certificate, it is guaranteeing the identity of the entity to which the certificate belongs. The CAs themselves have certificates to prove their identities. So, in case of a hierarchy of CAs, a CA higher up in the hierarchy signs the certificate of one below it. However, there is a root CA that cannot be endorsed by a CA above it. So, the root CA signs its own certificate (root-certificate). But, the root certificate needs to be provided out of band to the client and not over the wire

because the client has no means of verifying the root certificate if sent over the network. The security of the entire system depends on the safety of the root-certificate. If the root-certificate is compromised, then the entire system can be compromised (Charlie Kaufman 2002).

When PKI is used, a server can send its certificate to a client and the client can authenticate the server. To do this, the client should be preconfigured with the CA's public key. The client verifies that the server's certificate is valid using the CA's public key. Then, the client verifies that the server is authentic by using the server's public key.

2.4.2 IPSEC

IPSEC provides security at the IP layer (S. Kent and Atkinson 1998). The implementations cannot run from the application layer and have to be built into the OS. IPSEC helps specify what traffic to protect, how it is to be protected and where it is to be sent. Security services such as access control, integrity, data origin authentication and confidentiality can be provided by IPSEC. These services are provided at the IP layer. This allows various applications to make use of one common security architecture without custom designing it for their specific needs. The applications may choose to provide their own security service on top of IPSEC.

Since IPSEC is at the IP layer, any higher layer protocol (TCP, UDP, etc.) can use it. Different systems can be configured to use different modules without losing inter-operability. This is because two systems can continue to inter-operate using common modules even if they are configured differently. This possibility to use different security modules is a very useful feature because, algorithms that are not used at the time of deployment, can be added in future. However, the possibility to be up to date with new advances in secure algorithms is not available because the modules have to be known at compile time of the IPSEC implementation. IPSEC provides choices in the security services to be used, the granularity at which a service is applied and how the secret keys are managed (Naganand Doraswamy and Harkins 2003).

Security Associations are fundamental to the functioning of an IPSEC based security service. A Security Association (SA) determines the parameters to configure IPSEC for a particular connection. For instance, a SA will have a particular configuration for the algorithm to be used for communication on that particular connection. Before two nodes start communication, they have to negotiate and decide on a SA that will work for both of them. Both nodes need to know the algorithm beforehand to be able to use the SA. However, a node may acquire an unavailable algorithm from some distribution point if the vendor supports this capability.

A SA is sufficient to determine the way communication will take place between two nodes. The secret keys used in the communication need to be managed by a key management system. Internet Key Exchange (Charlie Kaufman) is a key management system used by default. The users may choose their own key management system (S. Kent and Atkinson 1998).

IPSEC is a point-to-point protocol and it is not possible for it to support multicast with protocols such as IKE for key management. Since source authentication and anti-replay are originally supported by IPSEC, a key distribution and management protocol for IPSEC multicast would also have to support them.

There are many proposed solutions to this problem, such as Multicast Key Management Protocol, Group Key Distribution Center and Group Key Management Protocol. However, they have problems such as high latency, non-scalability etc. Due to this, research is ongoing and applications may need to customize it to their needs (Naganand Doraswamy and Harkins 2003).

IPSEC provides various choices to the users and thus enables a wide range of applications. The independence of higher layer protocols is also advantageous.

2.4.3 TLS

Transport Layer Security (TLS) (T. Dierks and C.Allen 1999) provides security over a reliable channel. It is a commonly used security mechanism like IPSEC. The security services provided by TLS include authentication, confidentiality and message integrity. As in the case of IPSEC, TLS also enables negotiation of algorithmic parameters and key establishment.

Compared to IPSEC it is easier to deploy TLS as it can be inserted between the application layer and the transport layer whereas, IPSEC has to be deployed at the network layer. But, TLS relies on reliable transport and cannot be used for datagram transport.

Once a session is established, the server lets the client know its public key. The client can now communicate to the server securely. The server decrypts the information sent by the client and this information is used to decide a shared secret key. The client and sever can communicate securely using this shared secret key. For authentication of the server, digital certificates are used. The client can also be authenticated with its own certificate (T. Dierks and C.Allen 1999).

2.4.4 DTLS

Transport Layer Security (TLS) is widely used for network traffic security. However, TLS relies on reliable transport channel and hence cannot be used for datagram traffic security. Datagram TLS (DTLS) (Nagendra Modadugu and Rescorla 2004) has been designed to remedy this i.e., to enable datagram traffic to use TLS. DTLS makes minimal changes to TLS and borrows techniques for securing datagram traffic from IPSEC. However, some algorithms for instance RC4, are not supported by DTLS because DTLS does not maintain state.

2.4.5 Kerberos

Kerberos was designed by MIT and is a network authentication protocol that uses secret key cryptography. It is partly based on Needham and Shroeder's trusted third party authentication protocol. A Key Distribution Center (KDC) is a trusted third-party that knows the keys for all other nodes. When two nodes need to communicate, the KDC provides them with a session key in a secure manner to enable private communication. The primary goal of Kerberos is to eliminate the need to transmit a user's password in the clear across a network.

Kerberos based authentication is used both for users and for network servers (principals). Support for a large network is provided with multiple KDCs. Similarly, inter-organizational communication is supported using realms. A realm is established by each organization. Inter-realm keys are used by clients to communicate across realms (J. Kohl and Neuman 1993).

Kerberos uses a credential called a ticket. A ticket contains the name of the server and client, the IP address of the client, a timestamp, a lifetime and a session key. All this information is encrypted using the key of the server for which the ticket is issued. So, once issued, the client can use it to talk to that particular server, multiple times until the ticket expires. A client has to authenticate itself and obtain such tickets for accessing the services offered by any server. The client has to get a ticket for each service that it needs to access (Mogollon 2007).

Apart from authentication, Kerberos also supports confidentiality and integrity of messages exchanged. Version 5 provides some additional features such as the possibility of using different algorithms, delegation of rights to other users, public keys for users (J. Kohl and Neuman 1993).

2.4.6 GridStat Security Management System

GridStat's Security Management System (SMS) was developed to secure GridStat communication protocols. With its capabilities before the work described here, the SMS addresses two of the security goals discussed earlier namely, confidentiality and integrity. It also addresses availability to a lesser degree.

The management plane and data plane of GridStat have different communication protocols. Services provided by SMS have been applied only to data plane communications. To distinguish between the data plane and management plane security services, the current capabilities of SMS will be referred to as data plane security architecture (DPSA).

Figure 2.1 shows the Security Management Plane and the data plane which comprise the Data Plane Security Architecture. It is worth noting that this is very similar to the GridStat Architecture presented in Figure 1.1 except that the Management plane in the earlier figure is replaced by the Security Management Plane here. This illustrates the key aspect that the SMP is simply a extension to the MP hierarchy performing security functions while MP performs management functions. The MP and SMP co-exist with each other but do not communicate with each other (we have proposed that they should

be coupled together in section 4.2). The MP and SMP are not shown together in these figures to avoid complexity in presentation.

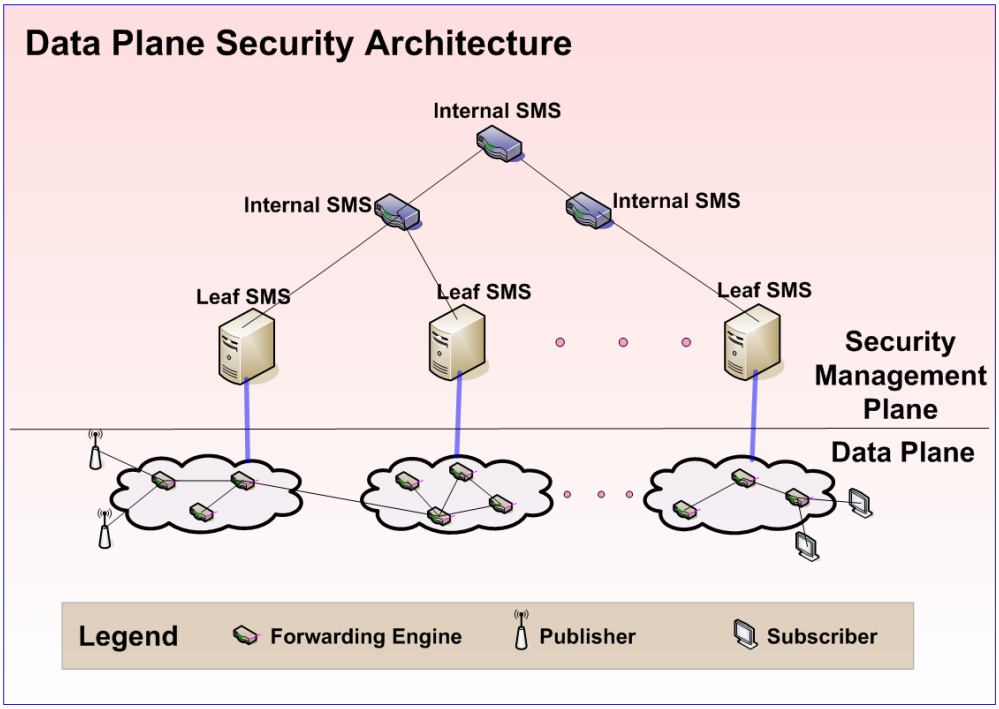

Figure 2.1 Data Plane Security Architecture

DPSA uses transparent interchangeable modules to achieve its security goals. Consequently, the security architecture is not bound to a fixed set of encryption algorithms. The system administrators can change over to newer modules that are introduced in the security field. The choice of modules can be adapted to unique requirements of the application by using different modules or a combination of modules. DPSA assigns modules and keys on a per PUV

granularity. This provides flexibility to the needs of different PUFs (Solum 2007).

One of the main requirements of PCS is that data is made available on time. GridStat has been designed with specific goals, one of them being situational awareness (Dave Bakken, Carl Hauser et al. 2009). DPSA caters to this need by using end-to-end security instead of time consuming hop-by-hop encryption/decryption.

DPSA is a solution designed specially for GridStat's unique security requirements. But it provides only confidentiality and integrity services and only for the data plane. Authentication, access control, and accountability are some of the services need to be supported. Authentication is addressed in this work.

2.5 Conclusion

DPSA provides few focused services that contribute to the complete security architecture. It has been designed to accommodate PCS specific needs and is flexible to future changes in the security field. Real-time communication requirements have been given foremost priority. Other security services such

as authentication, access control and accountability need to be incorporated to DPSA. Among them, authentication being the fundamental step for security needs foremost attention. Though the security solutions used for other publish/subscribe systems and security standards cannot be applied as is to GridStat, there are techniques used in them that can be applied to GridStat. Section 3.2 is a discussion of the applicability of the techniques discussed in this chapter.

CHAPTER THREE
DESIGN

This chapter presents features of GridStat and the applicability of security standards to GridStat. This leads up to the techniques used by this thesis to solve the problem at hand, thus laying down a foundation for the design. The design of authentication, key change and module change protocols are presented. The incorporations done to SMP are discussed followed by the XML policies required for the authentication protocols.

3.1 Key features affecting the design of security services in SMP

GridStat's unique features such as managed publish/subscribe communication model, relatively static topology, potentially inaccessible nodes, importance of availability, potentially long lifetime and presence of multiple administrative domains need to be carefully accommodated while designing security services for GridStat. It is important to understand the influence that these features have on the design. Table 3.1 shows some of GridStat’s features and the consequence or leverage they provide for our design.

Being a managed publish/subscribe system, GridStat has separate set of nodes in a hierarchy that accept or reject requests for subscriptions according to the quality of service constraints. The actual data is distributed over the data plane that is managed by the management plane. The communication among the management nodes has to benefit from the encryption and integrity provided by the SMS as discussed earlier. It is very important to protect this communication because, the decision making and configuration of the routes used in the data plane relies on this communication. It is insufficient to just protect the data and to leave the communication that controls it to be clear.

#	GridStat Feature	Consequence/Leverage
1	Managed publish-subscribe	Central topology and routing knowledge
2	Well-planned topology changes	Possible to pre-load keys
3	Hierarchical management plane	Can be used to reflect the level of control in administration (Allow configuration changes from top to bottom and not vice versa)

Table 3.1 GridStat Features

However, the security constraints for the management plane are different from the data plane. The real-time constraints in the data plane can be relaxed in the management plane and the communication is sporadic. The hierarchical arrangement of the nodes in the management plane also indicates a level of

control that the nodes higher up in the hierarchy have over the ones below them. So, the design should also permit configuration changes in security from a parent to its child but not vice versa.

The topology in this scenario is relatively static compared to the Internet. Every change in the topology is planned and known in advance. Due to this, it is possible to put some information that uniquely identifies nodes in them when they are deployed. There is no necessity of providing nodes with identification information such as certificates or keys after they are on the network. Security attacks on the key material through the network can be eliminated by not exchanging them over the network. SMS's design of using pre-loaded secret keys leverages this possibility. Since it is known when and where every node is deployed, the pre-loaded secret keys are loaded at the time of deployment of the node. The management plane can also use the same pre-loaded secret keys design, because this eliminates the need to exchange keys over the network.

Another important aspect is that after deployment, some nodes may be physically inaccessible. So, just pre-loading one key and using it for a while and loading another one physically is infeasible. Due to this, SMS proposes pre-loading a set of keys that can be changed without exchanging them over the network.

Availability has utmost importance in PCS. This is because PCSs support life sustaining services. Due to this, it is not possible to completely shutdown the system for maintenance. The system has to be in service continuously. Some nodes may be physically inaccessible once deployed and yet they have long life expectancies. Security architectures for PCS should meet all these requirements together.

#	GridStat Feature	Requirement	Solution
1	Physically inaccessible nodes	Allow safe key change remotely	Pre-loaded key set allows moving safely form one key to another
2	Availability has utmost importance	Cannot bring down the entire system	Switch keys safely and change modules with possibility to pass modules safely while in service
3	Long life expectancies	Key material should not be exhausted during the system lifetime	Use pre-loaded keys for authentication and using them distribute keys for encryption
4	Evolution in security	Cannot risk being outdated	Keep up to current advances by introducing and passing on new modules safely
5	Multiple administrative domains	Keep business-sensitive information safe	Pair-wise set of keys safe even within a single administrative domain

Table 3.2 Requirements in GridStat

The field of security evolves very fast and techniques trusted a few years ago may be discovered to be broken today. So, the security architecture, though continuously available and not having physically accessible nodes, has to be up to date with technological advances. The possibility of changing modules after deployment introduced in for encryption is hence a very useful feature and needs to be adapted for authentication. Though the actual protocol will not be the same, the underlying concept of changing modules to accommodate newer protocols is the same.

Though pre-loaded keys are ideal for this situation, it is not possible to say how many of them have to be pre-loaded. However, it is important to sustain the pre-loaded keys for as long as possible. So, it is necessary to spend some effort on ways to preserve the keys for a very long lifetime.

The presence of multiple administrative domains implies that each of them require to protect business sensitive information. The pre-loaded keys in SMS are such that each set of keys is only known by the pair of nodes that are communicating with each other. For instance if a parent has children A and B, then the parent shares one set of keys with A and a completely different set of keys with B. Hence, even though multiple domains are interacting with each other, it is still possible to keep information protected. Similarly, each parent

can influence what modules are used by its children and here too, each administrative domain can control the security level that is used.

Table 3.2 summarizes the key features and their effect (requirement) on the design and how they can be accommodated.

3.2 Applicability of standards and techniques

The communication model of GridStat, and candidate security standards contribute to the design of authentication protocols and may eventually be used for the design of the MPSA. The standards and techniques discussed in the previous chapter have varying levels of applicability to GridStat.

3.2.1 Publish/Subscribe systems

GridStat has relatively static nodes and any change in the topology is well known and well planned. This unique feature can be leveraged to securely distribute keys. Hence a set of pre-shared keys are distributed to the nodes at the time of their deployment (Solum 2007). Though trusted third party based key distribution can be used for GridStat, the pre-shared key option is a better one because the keys need not be shared through the network and hence cannot be detected through the communication. While trusted third party based

key distributions are adopted by other systems (Khurana 2005; Shrideep Pallickara, Marlon Pierce et al. 2006; Mudhakar and Ling 2007), it is not necessary for GridStat to do the same. It is possible that the key material may be compromised by attacks on the nodes themselves. So, it cannot be claimed that the keys are completely safe. Yet it can be claimed that possible attacks on the keys, through communication have been avoided altogether.

GridStat is different from publish/subscribe systems that are content-based or topic-based. So, techniques that work well for those systems may not be suitable for GridStat. The difference is that since GridStat is a managed publish/subscribe system, the brokers in GridStat do not participate in routing. They exist in a separate plane and have different communication styles compared to brokers in the other system. Due to this, securing communication in the management plane cannot use any technique that uses key derivation based on content or topic (Mudhakar and Ling 2005; Shrideep Pallickara, Marlon Pierce et al. 2006; Mudhakar and Ling 2007).

3.2.2 Security standards

Security standards such as PKI, IPSEC, TLS and Kerberos are candidates for PCS security architectures. However, they cannot be used as is. The rest of this

section discusses each of these standards separately with the focus on what features can be applied to GridStat and what features cannot.

PKI

To apply PKI to GridStat, it is possible to use a set of pre-shared root certificates similar to sharing pre-shared keys. Doing this will enable switching the root-certificate in case of a breach since the entities already have the next key to be used by the CA in the pre-shared certificate. But the ability to switch certificates still has one problem unsolved, which is how to stop using the old certificates. Regardless, (Solum 2007) observes that this solution is not suitable for GridStat's data plane communication as the data plane requires multicast communication and needs communication to be real-time.

In the MP and SMP, where these multicast requirements and real-time requirements do not exist, it is still more suitable to use pre-loaded keys per pair of parent-child nodes. This is because a compromise of a PKI root certificate in case of PKI would render the entire system that is under it to be unsafe. However, a compromise in case of using pre-loaded set of keys for each pair of nodes means that a compromise will render only the pair of nodes insecure and will not affect other parts of the system.

Consider the case where PKI is used in the same fashion as pre-shared secret keys, in order to reduce the vulnerability due to wide use of a single root certificate. Such usage of PKI introduces the same key complexity and PKI essentially becomes a private key system. The actual need for PKI is absent is this scenario. Another important aspect is that the use of public keys in certificates introduces dependence on algorithms. So, longevity is affected as it is not possible to change public key algorithms on top of pre-shared keys.

IPSEC

IPSEC is a widely used and well tested standard. These characteristics are important for any security system as it adds credibility. Since IPSEC is in the IP layer, all applications can share it and do not need their own security architectures. But applications may use their own architecture in addition. IPSEC has been designed with the idea that the application data needs encryption and the configuration of security is concerned only with IPSEC. Currently, it is not possible for the application to control the configuration of security. This is because, interaction of applications with IPSEC would require special interfaces as they are on different layers altogether. If an interface is designed for application logic to trigger something in IPSEC, it may need to be redone if another requirement arises. IPSEC is platform dependent and this poses yet another problem.

In case communication is strictly between one point to another, IPSEC needs one encryption at the sender and a decryption at the receiver. To perform rate-filtering (See 1.2.1 above), the status routers in GridStat need to be aware of the contents of the packet. So, if IPSEC is used, decryption and encryption is required at every hop and so a considerable amount of time is spent on it at every hop. This affects the real-time requirements of GridStat. Another basic problem is due to the fact that IPSEC design was not made with multicast in mind. Work has been done to extend multicast support to IPSEC. However, there is no known widely used or tested standard.

IPSEC supports different cryptographic algorithms. This is a feature that GridStat's SMS emulates and needs to be carried over to authentication algorithms also. This is because GridStat has a long life expectancy. So supporting different cryptographic algorithms and allowing it to be changed after deployment would accommodate newer algorithms that are possibly developed after deployment of the security architecture. This enables the security architecture to use the technological advances in the security field.

IPSEC allows choice of algorithms based on the nodes' knowledge of those algorithms. So, it is left to the negotiating entities to decide which algorithm to use for their communication. GridStat would benefit from the ability to change algorithms. However, it has to be ensured that the safety of the modules is

reasonably guaranteed. The approach proposed in this thesis can be extended to a similar service for IPSEC that would facilitate a safe way to obtain unavailable algorithms.

TLS and DTLS

Like IPSEC, TLS is also application protocol independent. It has a standard implementation available. However, it has not been designed for datagram traffic. This places limitations on the type of cryptographic algorithms that can be used with it. DTLS extends TLS and adds on datagram capability to it. DTLS has referred to IPSEC for designing this extension.

Even after this capability was added, some algorithms like RC4 are not supported. DTLS cannot be depended upon for support of new algorithms developed after deployment, as the security architecture is expected to have a long lifetime.

To use different algorithms than those that are already supported by TLS, they need to be registered as cipher suites by publishing and RFC specification (T. Dierks and C.Allen 1999). Clearly, depending on TLS or DTLS for support of different algorithms does not support over-the-wire defines way to deploy new modules.

Kerberos

In GridStat the management communication takes place between two brokers that have a parent-child relationship and their communication will be infrequent. However, they will have to get a new ticket or renew the ticket they possess. Since the tickets have lifetimes associated with them, and since the communications are infrequent, the nodes might end up getting a ticket every time they need to communicate. There will be many tickets obtained for communication. An intruder can collect these tickets and attempt a password-guessing attack (Steven M. Bellovin and Merritt 1991). This makes Kerberos unsuitable for the problem at hand.

Kerberos supports different cryptographic algorithms, but integrity-only algorithms are limited to five algorithms specified in Kerberos Version 5 documentation. Kerberos also lacks a standard password change mechanism which will be a serious problem for supporting applications with long life expectancy (J. Kohl and Neuman 1993). In Kerberos, there is heavy dependence on the KDC and hence the level of availability required for PCS cannot be easily supported with Kerberos. Any communication that has to take place depends on the tickets that are issued and hence if the KDC is down, the rest of the system cannot carry forward any communication until it is restored.

Conclusion

Security standards being well tested are the very first candidates considered when a security architecture is to be designed. But security standards may go out of date. One of the reasons for this is their dependence on specific algorithms. Consider the case of MD5 cryptographic hash function. It has been widely used since its publication in 1991. Flaws have been discovered in it since 1996. Researchers have recently published a theoretical attack on it (Schneier 2009). Similarly, it is possible that other algorithms widely used today may be discovered to be broken in future. It is also possible that new cryptographic algorithms designed in future, that are considered better than current ones. It is essential for a system with long life expectancy to be flexible enough to accommodate these changes.

3.3 Pre-loaded key set

According to the design of GridStat SMP, a pre-loaded key set is used for encryption of the messages passed between the node pair. When a new node is added to GridStat, an initial encryption module and a set of keys are provided to them and their parents. The parent shares different sets of keys with each of its children. Hence, one set of keys are used only between a pair of nodes.

Between each node pair, keys can be switched a maximum of k times if the key-set size is k.

Our choice of identification material for the authentication service is the pre-loaded key set, because the pre-loaded key set is safe from attacks through communication. It is important to consider how these keys are consumed, as the number of keys is proportional to the time during which the security services can be sustained. If the keys are consumed fast, an enormous number of keys would be required to sustain a long-lived system. When key-set size is large, it is difficult for the parent nodes to maintain multiple sets of keys. It is hence necessary to make optimal use of each key in the key set.

Notice that keys get consumed not only when keys are switched but also when modules are switched. Suppose the authentication service and the encryption service both make use of the pre-loaded key set, keys need to be switched more often because the keys are used more often. In addition to this, each authentication module switch protocol would also consume keys. It is evident that when two services operate using the same set of keys, the keys get exhausted at possibly double the rate compared to one service operating on them.

It is standard practice that an authentication protocol concludes in a session key that can be used for encryption (Michael Burrows, Martin Abadi et al. 1990; Charlie Kaufman 2002). To solve the key consumption problem, it is proposed that this approach be adopted for SMP. Hence, the authentication service should alone use the pre-loaded key set and the encryption service would use the keys provided at the conclusion of the authentication protocol. To continue using the module change capabilities introduced in (Solum 2007), the authentication protocol will conclude with a set of keys instead of a single key for encryption. When this set is exhausted, the authentication protocol can be run again to generate a new set of encryption keys.

The benefit of layering encryption on top of authentication is that this scales better than allowing encryption and authentication to borrow keys from the same pool. In the description of the authentication protocol, it is evident that it is possible to further conserve the pre-loaded keys by using them strictly for identification purposes.

It is important to be aware of and handle issues that arise in mutual authentication protocols used with symmetric keys such as reflection attack and password guessing attack. This can be handled by introducing asymmetry and by requiring the initiator to sign first as discussed below.

3.3.1 Introducing asymmetry

In the pre-loaded key set of size k, the k keys are symmetric. So, a value RND signed by the parent will have the same result as the value RND signed by the child. In this case, reflection attacks are possible. A reflection attack is one where an intruder tricks an authentic node into signing its own challenge (Charlie Kaufman 2002). Figure 3.1 illustrates how this attack can take place.

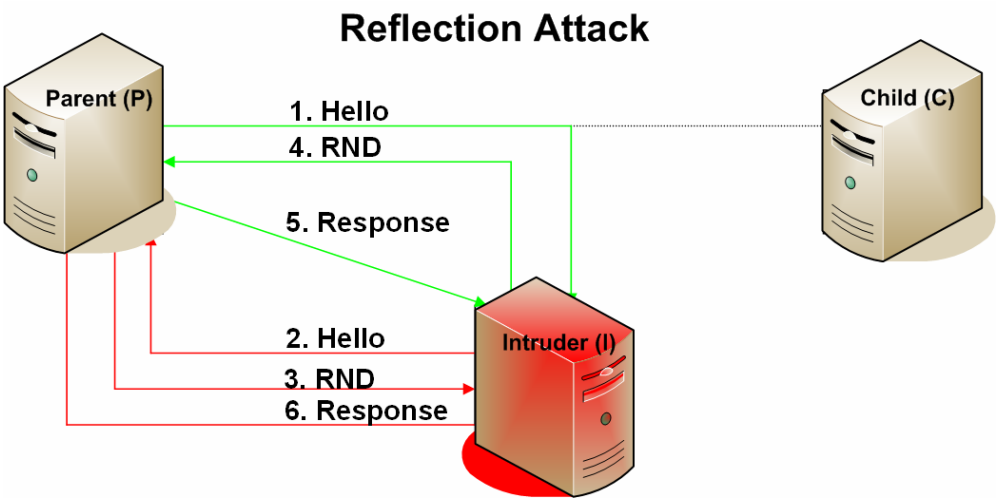

Figure 3.1 Reflection Attack

The intruder (I) waits until Parent (P) contacts the child (C) and intercepts this message (Step 1). In another session, ‘I’ sends a ‘Hello’ message to the parent and gets back a challenge RND (Steps 2 and 3). 'I' does not know the correct signature for RND and so it sends this same value RND to 'P' in the first session (Step 4). The parent signs its own challenge (Step 5). Now, in the other

session, ‘I’ can resend the same response as its own response (Step 6) and successfully communicate with ‘P’ maliciously.

This problem arises due to the fact that both signatures of the parent and the child look alike. If that is not the case, then reflection attacks can be prevented. One straightforward approach is to use different keys for different directions, in our case different key set for Parent to Child and for Child to Parent communication (Charlie Kaufman 2002). The parent will use one set 'Parent keys' for signing and use another set 'Child keys' for checking the child's responses. Similarly, child will use 'Child keys' for signing and use 'Parent keys' for checking the parent's responses.

3.3.2 Initiator always signs first

A common way of deducing the key used for authentication is to make an authentic node sign many challenges. The intruder knows the plain text of the challenges and also knows the signed values. The algorithm used for signature is well known. If the intruder can collect many such signatures it may be possible to find the key used for signing. So, the intruder can analyze these signatures and attempt to get the key used for signing by doing a password-guessing attack (Charlie Kaufman 2002; Smith and Marchesini 2007).

This can be prevented by ensuring that legitimate participants do not sign any value that the intruder provides. The way to do this is to make the initiator of a conversation prove that it is authentic first, and then proceed to sign the initiator’s challenge. This ensures that the intruder cannot perform an offline password-guessing attack. The number of values that an intruder will have to carry out a guessing attempt is reduced. So, in all the protocols presented in this chapter, the initiator always signs first. Only if their signature is authentic, the receiver will sign the initiator's challenge. It is still possible for a man in the middle attack to take place where the MITM can intercept an authentic node's attempt to connect to another node. The MITM can send its challenge to the initiator to sign in this case. The usage of Diffie Hellman for authentication discussed later in this chapter further makes the reflection attacks and password guessing attacks infeasible.

3.4 Authentication

While mutual authentication is the fundamental step towards security in our scenario, other protocols that help maintain the longevity of the system are also needed. The key switch protocol and module switch protocols serve this purpose. These three protocols are presented in the following sections.

3.4.1 Authentication and key exchange protocol

When two nodes need to communicate with each other, they start with an authentication protocol. The authentication protocol design is similar to the mutual authentication steps of the key management protocol for IPSEC, namely Internet Key Exchange (Charlie Kaufman). Like in IKE, Diffie-Hellman is used to exchange the proofs of identity and also to exchange keys for performing encryption.

The main difference comes with fact that in GridStat the nodes have a control hierarchy. The nodes in higher levels determine which algorithms are used by the child nodes. So, a need to propose and accept cryptographic suites to be used does not exist. The parent node simply uses an algorithm known to the child. Otherwise, it provides the necessary algorithm to the child node.

Diffie-Hellman key exchange method has been extensively used in literature. It allows two entities to establish secure communication over an insecure channel. The two communicating entities decide upon a prime number $\boldsymbol{p}$ and a base $\boldsymbol{g}$. Let the entities be $\boldsymbol{A}$ and $\boldsymbol{B}$. Now, $\boldsymbol{A}$ decides a secret number $\boldsymbol{a}$, known only to itself and sends ($\boldsymbol{g^a \bmod p}$) to $\boldsymbol{B}$. Similarly, $\boldsymbol{B}$ decides a secret number $\boldsymbol{b}$ and sends ($\boldsymbol{g^b \bmod p}$) to $\boldsymbol{A}$. $\boldsymbol{A}$ can calculate $\boldsymbol{(g^b \bmod p)^a \bmod p}$ using what it received and what it knows. $\boldsymbol{B}$ can calculate $\boldsymbol{(g^b \bmod p)^a \bmod p}$. Now, both

entities have arrived at the same value. This value is the Diffie-Hellman secret key. This key exchange method is based on the discrete logarithm problem. It is not possible to calculate the Diffie-Hellman key using just the values exchanged on the communication channel (Diffie and Hellman 1976).

<u>Authentication Protocol</u>

Phase 1:

1. Child: {Hello, DH parameters}

[Parent: Calculate DH-Key]

2. Parent: {Hello, DH parameters, $\{R1\}_{DH\text{-}Key}$}

[Child: Calculate DH-Key]

Phase 2:

3. Child: $\{\{\{R1\}_{DH\text{-}Key}\}_{Child\text{-}Authentication\text{-}Key}, \{R2\}_{DH\text{-}Key}\}$

4. Parent: $\{\{\{R2\}_{DH\text{-}Key}\}_{Parent\text{-}Authentication\text{-}Key}, \{Session\ Keys\}_{DH\text{-}Key}\}$

Protocol 1 Authentication Protocol

Before the authentication protocol starts, each of the communicating nodes will have their pre-loaded key set. They will also be provided with an initial module at the time their keys are pre-loaded. So, the nodes have some module to start their communication. Protocol 1 is the authentication protocol.

In phase 1 of the protocol, the communicating nodes exchange the Diffie-Hellman parameters required for phase two. At the end of phase one, both the nodes have calculated the Diffie-Hellman secret key based on the exchanges just made.

In phase 2, both nodes sign their challenges to prove their identity. Once both nodes verify the other node's identity by checking the responses, the authentication process of the protocol completes. The target node also generates a set of keys necessary for encryption and sends it along with its signature on R2. The protocol thus completes after mutual authentication and generating keys for encryption.

It is important to ensure the correctness of the authentication protocol especially since the correctness of rest of the security services is based on the authentication protocol's correctness. (Michael Burrows, Martin Abadi et al. 1990) states the goals to be verified to prove the correctness of the protocol in the form of beliefs held by participating entities at the conclusion of the protocol. The goals can be informally stated as:

The participating entities believe that

- the session key exchanged at the conclusion of the protocol is safe.

- the other entity holds the above belief.

These goals are verified in section 5.2.1 that presents the evaluation of this protocol.

3.4.2 Key switch protocol

When keys are used for a long time, they become vulnerable. So, continuing to use the key until it is compromised will be dangerous. Hence there is a need to be able to switch from one key to another safely when necessary. The key switch protocol is used to change keys when such a need arises. The protocol utilizes the current encryption key and module for encryption of the exchanged values as seen in Protocol 2.

The parent controls the keys and modules and so the parent initiates the key switch command. But as always, the initiator is authenticated first. So, the child returns a value to be signed by the parent. To sign this challenge, the parent uses the next parent key and not the current one. This ensures that the initiator knows the encryption key and also the new authentication key. There is no possibility for an intruder to know the new authentication key through the communication as it has never been used. So, if the parent is able to authenticate itself with the new key, then it is sufficient. Similarly, the child uses the new authentication key to sign.

Key Switch Protocol

1. Parent: {Switch Key}

2. Child: {R1} `Session-Key`

3. Parent: {{{R1} `Next-Parent-Authentication-Key`} `Session-Key`, {R2} `Session-Key`}

4. Child: {{{R2} `Next-Child-Authentication-Key`} `Session-Key`}

5. Parent: {Acknowledgement, {{R2+1} `Next-Parent-Authentication-Key`} `Session-Key`}

Protocol 2 Key Switch Protocol

The correctness of the key switch protocol should also be verified to ensure that the system will be in a good state throughout its lifetime. The goals of an authentication protocol specified in (Michael Burrows, Martin Abadi et al. 1990) are specifically for the session key exchanged at the end of the protocol. Since the key switch protocol does not conclude in a session key, the same goals cannot be used to verify this protocol. So, the same goals are stated for the next authentication key instead of the session key and they are verified as presented in section 5.2.2.

3.4.3 Module change protocol

The module change protocol is crucial to longevity because it is not just sufficient to manage keys well to sustain them longer. It should be possible to

change to newer algorithms if necessary. As in the case of key switch, the module change can also be initiated only by the parent. It is possible that the module to which the nodes need to change is not available at the child node. If so, the protocol behaves slightly differently than if the child has the new module with it already. Only one step in the protocol differs in the two cases.

Module Change Protocol (With Module Request)

1. Parent: {Change Module}
2. Child: {R1, Request}$_{\text{Session-Key}}$
3. Parent: {{{R1}$_{\text{Next-Parent-Authentication-Key (Old Module)}}$,
   {R1}$_{\text{Next-Parent-Authentication-Key (New Module)}}$,
   New Module, R2}$_{\text{Session-Key}}$}
4. Child: {{{R2}$_{\text{Next-Child-Authentication-Key (New Module)}}$}$_{\text{Session-Key}}$}
5. Parent: {Acknowledgement}
   [Parent & Child: Switch to next authentication key]

Protocol 3 Module Change Protocol with Module Request

The current encryption module and key are utilized for encryption in the protocol. Protocol 3 illustrates the module change protocol where the child requests a copy of the new module. The parent sends the change authentication module command to the child node. The child checks if it already has the

module and if not requests the module from the parent node. So as a reply to change module command, child sends back a challenge R1, to the parent and also requests the new module.

Now, the parent signs R1 using the next authentication key. Again this is to ensure that any intruder will not know the next authentication key through communication as it is unused still. The parent uses both the currently used authentication module and the new module to sign R1. This is shown in step 3 of the protocol in Protocol 3. Usually, the usage of only the new module should establish the initiator's identity since the next authentication key is to be used with it. However, it is possible that the initiator sends a malicious module or a dummy module as the new module with which all the authentication tests will pass. So, if a signature on the challenge with the old module and new key is also obtained, it will make sure that the initiator knows the new key and so is genuine. It the initiator is genuine, it is assumed that the module sent is also genuine.

Consider the case when the initiator is genuine and the module as it was found to be broken is changed. In this case, the new authentication key is revealed by using the old module. Perhaps it is too difficult to deduce the new key from one exchange. But it is safer to move to another key after this. So, at the end of the protocol, keys are switched again to move to an unused authentication key.

In the step 3 where parent sends its signatures, it also sends a random value R2 for the child to sign and it sends a copy of the module. On receipt of this, the child checks the parent's response and verifies that it is genuine. Then the child signs R2 with the next child authentication key and new module. Now it is sufficient for the child to send the signature only with the new module because the new module itself was sent by the parent. Finally the parent verifies the child's signature and sends an acknowledgement. Once this is done, both nodes switch their authentication key because the keys were used with the old authentication module.

Changing a module when the child already has the target module differs from the previous one slightly. In step 3 of the protocol (Protocol 4), the child simply sends a challenge R1 for the parent and does not request for the module. The parent now needs only to sign the challenge R1 with the new module and the next authentication key. This is because the child knows the new module and so the risk of it being a dummy one is eliminated. The parent also sends a random value R2 for the child to sign. Step 4 and 5 of the protocol are same as Protocol 3.

Finally, the nodes do not change keys again because the new key was not revealed in any way. The new key was not used with the old module. Hence,

once both nodes authenticate each other, they can continue using the key they used for signing the challenges in the protocol and need not change once more.

In the second version of the protocol where there was no request for the new module, if at any point authentication fails, use of the old module and old authentication key is continued. This is because there is no evidence that necessitates this change.

Module Change Protocol (No Module Request)

1. Parent: {Change Module}

2. Child: {R1} `Session-Key`

3. Parent: {{{R1} `Next-Parent-Authentication-Key (New Module)`, R2} `Session-Key`}

4. Child: {{{R2} `Next-Child-Authentication-Key (New Module)`} `Session-Key`}

5. Parent: {Acknowledgement}

Protocol 4 Module Change Protocol without Module Request

In the first version of the protocol, the scenario is different. In the first protocol, there are basically three cases of failure. Table 3.3 illustrates them. In the first case, both the signatures using old module and new module fail. In this case, it is evident that the old module and key can be used as the intruder

did not authenticate itself successfully. So there is no evidence that the old module or key is broken and their use is continued.

Signature with Old Module	Signature with new module	Implication
Fail	Fail	No evidence of compromised modules or keys. Evidence of an attempted attack
Fail	Pass	No evidence of compromise on old module or keys. Evidence of an attempted attack - New module is malicious.
Pass	Fail	Bug in new module or possible breach in old module and/or key
Pass	Pass	Initiator successfully authenticated. Switch to new module

Table 3.3 Implications of Challenge-Response in Module Change Protocol

The second case is when the signature with old module is incorrect but the one with new module is correct. The correctness of signature using new module is useful only when the signature with the old module is correct. This simply shows a case where the new module is a dummy one. So again there is no evidence that the old module or key is broken and hence the system can continue using them for authentication. But it is possible that this was an attempted attack. This attack is reported and has to be handled by a human or automated monitoring system.

The third case shown in the table is one where authentication with old module passed but with new module failed. This is a complicated case. This clearly shows that first and foremost, the encryption module and/or keys are not safe anymore and they need to be changed. This also shows that the old module is not safe any more, even with the new key. This may indicate a bug in the implementation on the initiator's side. Another possibility is that the initiator is malicious. The initiator if malicious has not made use of the fact that it can use its own dummy module for the new module. It cannot be reasoned why the intruder did not make use of this opportunity. But if this scenario is true, it shows the need for changing the authentication module. Again, this is reported and needs to be handled by a human or an automated monitoring system.

The fourth case is the simplest one where the initiator is authenticated using both the old and new modules. Hence, both nodes change to the new module. On comparing these module change protocols to the encryption module change protocol (Solum 2007), it is worthwhile to notice that there are no time constraints in the authentication protocol. However, one step in the module change protocol has strict time constraints. This is a result of the fact that the old authentication module is used to authenticate the initiator before the new one is used. If the authentication of the initiator fails with the old module, the target node terminates the protocol without testing the authenticity of the

initiator with the new module. But in case of the encryption module change, only the new module is used to prove the possession of the next encryption key.

3.5 Incorporations to GridStat

The three protocols discussed above have been added to the capabilities of GridStat. The protocols are used to authenticate the communication between a parent SMS and its child SMS. It is also used to authenticate communication between a leaf SMS and a data plane node.

Figure 3.2 shows the added functionalities and the interfaces to which these functions have been added. The Child SMS Communicator interface is used by a parent SMS to talk to its child SMS. The key switch and module change protocols initiate from the parent via the Child SMS Communicator interface. Similarly, in the leaf SMS these two commands originate from the leaf SMS's Data Plane Communicator interface.

The Parent SMS Communicator is used by a child SMS to communicate with its parent. The authentication protocol initiates from this interface as the child

connects to its parent. For the data plane nodes, Management Communicator interface plays the same role.

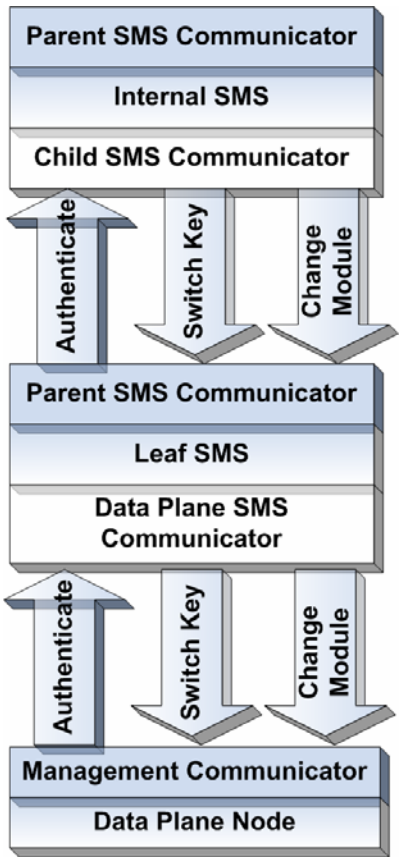

Figure 3.2 Incorporations to GridStat

The SMSs also have user interfaces (not shown in Figure 3.2) that currently control when the keys and modules are changed. All other interactions within the SMSs, data plane nodes are not shown here and they can be found in detail in (Solum 2007).

3.6 Security policies

The modules and keys to be used and other details necessary for communication between two nodes in GridStat are stored in a Security Management Communication Policy (SMCP). The format of this policy is shown in Figure 3.3. This format is the same as the format used for storing details about encryption.

```
<GridStatPolicy>
   <Type>SMCP</Type>
   <Parent />
   <Child />
   <Modules />
   <OutputKeys />
   <InputKeys />
   <KeyIndex />
</GridStatPolicy>
```

Figure 3.3 Security Policy

For encryption SMCP policies, both nodes make use of the same keys on either side. But in case of authentication, there is a need to use one pre-loaded key set for outgoing messages and one pre-loaded key set for incoming messages. So these sets are stored as separate SMCP policy files.

CHAPTER FOUR
MANAGEMENT PLANE SECURITY

This chapter presents the current capabilities of GridStat Security Management System (SMS) with focus towards MPSA. A discussion on extending SMP services to Management Plane Security Architecture (MPSA) is presented. The relationship between the management plane and SMP is discussed.

4.1 Extending services of the SMP to the Management Plane

The types of communication that are secure with confidentiality, integrity and authentication are SMS to data plane nodes and SMS to SMS. The communication between the QoS brokers and the communication between the QoS brokers and the data plane nodes are currently sent in the clear. Figure 4.1 shows communication that does not use encryption, integrity and authentication. The existing security architecture could be applied to the management plane by extending its services to the management communication and the communication between the management plane and the data plane.

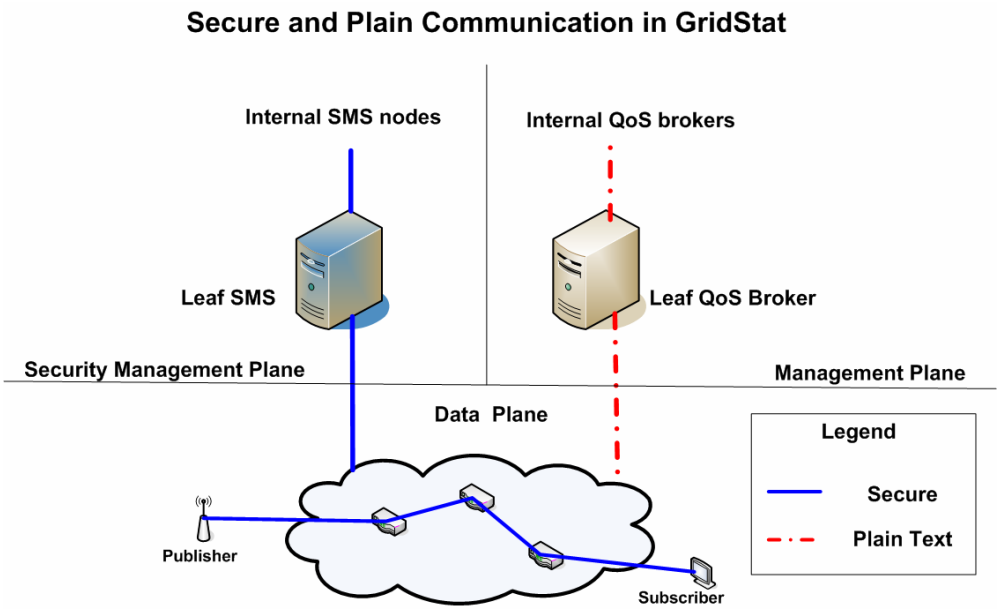

Figure 4.1 Secure and Plain Communication in GridStat

In Section 2.1 the security services that security architectures are expected to provide, were discussed. Our work incorporated the authentication service into GridStat. There are other security services that need to be added to the SMS. However, they are left for the future. Though the SMS provides efficient ways to change modules or keys when a breach is detected, currently there is no mechanism to detect a breach. This is also an important feature to be added to the SMS that is left for future work. Since the management plane has control over the data plane and makes strategic decisions for the data plane, it is important for this communication to be secure. Extending services of SMP to the management plane is essential. But there are some challenges in achieving this goal.

Though SMP is designed to be an extension of the management plane, the management plane is disconnected from the SMP currently. Firstly, this disconnect has to be solved. The management communication can be secured only after this is done. To this end, in the next chapter, a discussion on the relationship between QoS brokers and SMS and how they need to work with each other is presented.

QoS brokers are responsible for decision making in mechanisms such as rate filtering, flooding, redundant paths, mode change and RPC (Dave Bakken, Carl Hauser et al. 2007). So the QoS brokers will communicate with each other for decision making for all these mechanisms and other purposes. Before MPSA is implemented, an analysis of which of these communications need to be secure, and which do not has to be in place.

The QoS brokers have multiple access points to them. There are numerous CORBA interfaces to which security services need to be associated. So, very careful planning is necessary to design and implement the security service extension to management plane without adding vulnerabilities. There are multiple methods used in QoS broker communication and each of them has different set of values to be encrypted. There is not a single point where the security interface can be inserted. This adds complexity and calls for careful analysis of the functions of the QoS brokers.

Hence there are three important tasks to be carried out - combining the QoS brokers and the SMS, deciding which part of the communication needs to be secure and finally implementing the security extension without adding any vulnerabilities. This process will require a considerable amount of time and hence cannot be included in this thesis due to time constraints. However, while designing the authentication protocol for GridStat, it is important to consider the characteristics of management plane communication. It is necessary that SMP and its services should be designed in such a way that it can be extended to the management plane in future. So, this chapter presented ideas on the design of MPSA which would essentially be a logical extension to the SMP used for the management plane.

4.2 Coupling Management Plane and Security Management Plane

The organization of SMP with respect to all other components in GridStat is important. According to (Solum 2007), the Management Plane (MP) and the SMP are designed as two processes on the same hardware. The SMP sets up security policies for the data plane nodes to communicate. But the decision making process happens in the MP. There is an inter-dependency that makes it important for the MP and SMP to work hand in hand. Yet, the interfaces

between the QoS brokers and the data plane and between the SMSs and the data plane are different.

If the QoS brokers continue to reside on a plane different from the SMSs, then they need to have an additional interface to talk to their corresponding SMSs. They also need to use a pre-loaded key set to communicate securely among themselves.

The security policies used for the QoS brokers will be similar to the ones used by the SMSs to communicate among themselves. Yet, the QoS brokers need to be provided with their own policies. Similarly, the communication between the MP and the SMP needs policies and keys. As a result of this, there is complexity in terms of the number of communication interfaces and the sets of keys and the policies used. Figure 4.2 (a) illustrates the number of keys, policies and communication links that exist in this scenario.

It is proposed that the QoS brokers and SMSs be coupled as a single unit. The SMS acts as a middle man between the data plane nodes and QoS brokers and also between the QoS brokers and its children. SMS handles the security and communication aspects. The QoS brokers will be the decision makers but communicate through the SMSs thereby using SMSs' security capabilities.

Figure 4.2 (b) shows the significant reduction in the number of keys, policies and communication links compared to the earlier scheme Figure 4.2 (a).

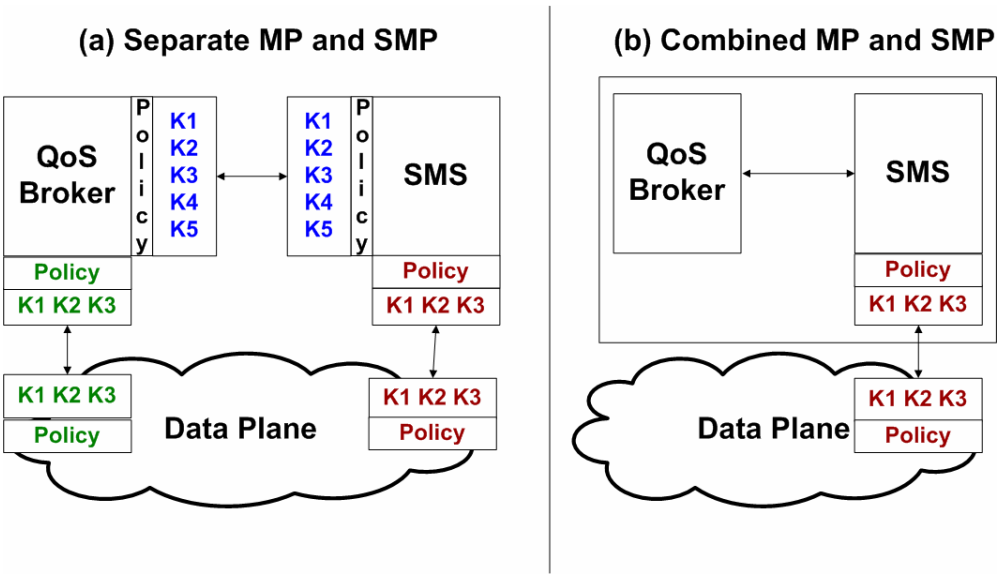

Figure 4.2 Coupling MP and SMP

In the proposed approach, responsibility is added to the SMS. However, vulnerabilities due to complexity in the first approach are avoided. An entire level of issues related to sending out policies for communication among the QoS brokers, between the management plane and data plane are solved. The existing policies, keys in SMP for security are also reused. Thus there is no need to add any more keys or policies to extend security services to the management plane.

CHAPTER FIVE
EVALUATION

Authentication protocols are fundamental to security. It is necessary to ensure that the protocols work as expected. To achieve this, the protocols design has to be analyzed and verified. Flaws due to cryptographic attacks, incorrect implementation or vulnerabilities in the deployed system are not considered. The Burrows, Abadi and Needham logic (BAN logic), which is a logic to formally express and verify authentication protocols (Michael Burrows, Martin Abadi et al. 1990) is used for analysis. The BAN logic has been widely used as a standard approach to prove other security protocols (Mao and Boyd 1993).

5.1 BAN Logic

The BAN logic can formally express beliefs that the participating entities hold. In the logic, objects can be differentiated according to their type for instance, principals (entities or nodes), encryption keys and formulae (statements). The basic constructs used to express a protocol in BAN are listed below:

1. P **believes** X: P believes that X is true.

2. P **sees** X : Some principal sent X to P

3. P **said** X : P said X at some point of time (could be in the past or present)

4. P **controls** X: P has jurisdiction over X. That is, P is the authority in the truth of X and should be trusted in this aspect.

5. **fresh**(X) : X has been sent in the 'present' epoch and not in the 'past'

6. $P \xleftrightarrow{K} Q$: P and Q have a shared key K and it is not known to any other principal except P and Q

7. $\{X\}_K$: X is encrypted with key K

The logical postulates are below:

1. **Message-meaning rule**: For shared keys, if a principal P believes that key K is shared with principal Q alone, and if it sees a message encrypted with K, it concludes that Q said the contents of the message at some point of time.

$$\frac{P\ believes\ (P \xleftrightarrow{K} Q),\ P\ sees\ \{X\}_K}{P\ believes\ Q\ said\ X} \qquad (1)$$

2. **Nonce-verification rule**: If P believes that X was said only recently, then if Q said X, P concludes that Q believes X.

$$\frac{P\ believes\ \ fresh(X),\ P\ believes\ Q\ said\ \ X}{P\ believes\ Q\ believes\ \ X} \qquad (2)$$

3. **Jurisdiction rule**: If P believes that Q controls X and believes that Q believes X, then P believes X.

$$\frac{P\ believes\ Q\ controls\ \ X,\ P\ believes\ Q\ believes\ \ X}{P\ believes\ \ X} \qquad (3)$$

(Note that the following two rules are not named as referred here by BAN logic authors. We name them as follows for easy reference)

4. **Freshness rule**: If one part of the formula is fresh, then the entire formula is fresh

$$\frac{P\ believes\ \ fresh(X)}{P\ believes\ \ fresh(X,Y)} \qquad (4)$$

5. **Belief rule**: Only when individual parts of a formula are believed, the entire formula is believed

$$\frac{P\ believes\ \ X,\ P\ believes\ Y}{P\ believes\ (X,Y)} \qquad (5)$$

To analyze a protocol, the assumptions of the principals' beliefs are first stated. Then the protocol is expressed in idealized form, which is a representation that is independent of implementation encodings. The idealized form of a protocol can be obtained by deleting plaintext, replacing key exchanged by its equivalent formula, replacing messages by its semantic equivalent and inserting new information which is semantically implied by the messages or formulas in the protocol (Michael Burrows, Martin Abadi et al. 1990; Mao and Boyd 1993). Then the logical postulates message-meaning rule, nonce-verification rule and jurisdiction rules are applied. The goal of an authentication protocol is to prove that at the completion of the protocol, it concludes in a session key K such that the following four beliefs hold:

$$\begin{aligned}
&A\ believes\ A \xleftrightarrow{K} B \\
&B\ believes\ A \xleftrightarrow{K} B \\
&A\ believes\ B\ believes\ A \xleftrightarrow{K} B \\
&B\ believes\ A\ believes\ A \xleftrightarrow{K} B
\end{aligned} \tag{6}$$

The first two goals indicate that principals A and B on their own are convinced that K is a good shared between A and B (it will not be discovered by any principal except A and B). These two goals only establish each principal trusts the key but does not suggest anything about the other one. These goals are only weak authentication goals. However, the last two goals also establish that the other entity also believes in K. This is a convincing situation to start using

K because both parties are convinced about it and they are also convinced about the other party's belief. Hence, together if all these four goals are met then it is a strong state.

5.2 Analysis of the Protocols

A verification of correctness of the authentication protocols is presented in this section. The authentication, key switch and module change protocols are analyzed in the following sections.

5.2.1 Analysis of the Authentication Protocol

The authentication protocol for mutual authentication in GridStat (Protocol 1) is verified using the goals in (6). The symbols P and C denote the Parent and Child respectively. DHP_C is the Diffie-Hellman parameter sent by the child and DHP_P is the Diffie-Hellman parameter sent by the parent. DHK is the derived Diffie-Hellman key. KPC is a pre-loaded key used for messages from Parent to Child while KCP is used from Child to Parent. Since there is a set of pre-loaded keys, KPC and KCP denote the currently used key pair in that set. Finally K denotes the set of session keys generated by P to be used for

encryption in later conversations by P and C. Any belief stated for K applies to every key in the session key set.

The initial part of the proof is a standard proof for verifying beliefs about the Diffie-Hellman key (Wessels 2001; Yang and Li 2006). The later part of the proof expresses beliefs about the session keys exchanged using the protocol. So, to prove the mutual authentication protocol, beliefs stated in (6) have to be proved for the Diffie-Hellman key DHK and for all keys in the session key set denoted by K.

The concrete protocol below, restates Protocol 1 in a brief manner:

$$
\begin{aligned}
&Msg1.\ C \rightarrow P : DHP_C \\
&Msg2.\ P \rightarrow C : DHP_P,\ \{R1\}_{DHK} \\
&Msg3.\ C \rightarrow P : \{\{R1\}_{DHK}\}_{KCP}, \{R2\}_{DHK} \\
&Msg4.\ P \rightarrow C : \{\{R2\}_{DHK}\}_{KPC}, \{K\}_{DHK} \\
&Msg5.\ C \rightarrow P : \{X\}_K
\end{aligned}
$$

Note that Msg5 is the first message from Child to Parent after the session key has been exchanged. After the authentication protocol completes, the Child communicates with the Parent proving its possession of keys in the set K. This step is a representation of the proof-of-possession step followed according to (Solum 2007). Though it is not a part of the protocol, it is shown here as it is needed to complete the verification of the protocol.

Initial assumptions:

There are some initial assumptions about the pre-loaded authentication key set and the challenges (R1, R2) generated by the Child and Parent. They are listed below.

Since P generated R1 in the current epoch,

$$P \ believes \ \ fresh(R1) \tag{7}$$

Since KPC and KCP are pre-loaded keys,

$$P \ believes \ P \xleftrightarrow{KPC} C \tag{8}$$

$$P \ believes \ P \xleftrightarrow{KCP} C \tag{9}$$

Since C generated R2 in the current epoch,

$$C \ believes \ \ fresh(R2) \tag{10}$$

Since KPC and KCP are pre-loaded keys,

$$C \ believes \ P \xleftrightarrow{KPC} C \tag{11}$$

$$C \ believes \ P \xleftrightarrow{KCP} C \tag{12}$$

After the protocol begins, there are some more assumptions that result. From the Diffie-Hellman parameter exchange Msg1 and Msg2, we have the assumptions

Since P generated DHP_P in the current epoch,

$$P\ believes\ \ fresh(DHP_P) \tag{13}$$

Since C generated DHP_C in the current epoch,

$$C\ believes\ \ fresh(DHP_C) \tag{14}$$

$$P\ believes\ \overline{C}\ controls\ DHP_C,\ P\ controls\ DHP_P \tag{15}$$

where $\overline{C}$ is some node that sent DHP_C to P.

$$C\ believes\ \overline{P}\ controls\ DHP_P,\ C\ controls\ DHP_C \tag{16}$$

where $\overline{P}$ is some node that sent DHP_P to C.

Since DHP_P and DHP_C were used by P and C to generate DHK, the freshness belief that they had for DHP_P and DHP_C results in the following two beliefs

$$P\ believes\ \ fresh(DHK) \tag{17}$$

$$C \text{ believes } fresh(DHK) \tag{18}$$

Since P generates the session key set and is responsible for it, the following are assumed (these are also assumptions that are held before the protocol begins but are stated here in the order that they appear in the protocol):

$$C \text{ believes } P \text{ controls } P \xleftrightarrow{K} C \tag{19}$$

$$P \text{ believes } P \xleftrightarrow{K} C \tag{20}$$

$$P \text{ believes } fresh(K) \tag{21}$$

Idealized Protocol:

Section 5.1 specifies how the process of idealization is done. This has been used to idealize Protocol 1. Msg1 and Msg2 only serve the purpose of calculating the Diffie-Hellman key (DHK). They do not appear in idealized protocols.

In Msg3, since R1 was encrypted using DHK, it means that the initiator of this message believes in DHK. So, after Msg3 we get,

$$P \text{ sees } \left\{P \xleftrightarrow{DHK} \overline{C}\right\}_{KCP} \tag{22}$$

Applying freshness rule (4) to the above using (7) (P believes that R1 is fresh), we get,

$$P\ believes\ fresh\left(\left\{P \xleftrightarrow{DHK} \overline{C}\right\}_{KCP}\right) \tag{23}$$

In Msg4, R2 is encrypted using DHK. So the initiator of the message believes in DHK. So, after Msg4 we get,

$$C\ sees\ \left\{\overline{P} \xleftrightarrow{DHK} C\right\}_{KPC} \tag{24}$$

In Msg4, we also have R2 encrypted using K. Again, the initiator of the message believes in K. So we get,

$$C\ sees\ \left\{\overline{P} \xleftrightarrow{K} C\right\}_{DHK} \tag{25}$$

Applying freshness rule (4) to the above using (10) (C believes that R2 is fresh), we get,

$$C\ believes\ fresh\left(\left\{\overline{P} \xleftrightarrow{DHK} C\right\}_{KPC}\right) \tag{26}$$

$$C\ believes\ fresh\left(\left\{\overline{P} \xleftrightarrow{DHK} C\right\}\right) \tag{27}$$

$$C\ believes\ fresh\left(\left\{\overline{P} \xleftrightarrow{K} C\right\}_{DHK}\right) \tag{28}$$

$$C \ believes \ \ fresh\left(\left\{\overline{P} \xleftrightarrow{K} C\right\}\right) \tag{29}$$

Since a message was encrypted by K in Msg5, it is evident that the initiator believes in key k. So after Msg5 we have,

$$P \ sees \ \left\{P \xleftrightarrow{K} \overline{C}\right\}_K \tag{30}$$

Protocol Verification:

$$\frac{P \ believes \ P \xleftrightarrow{KCP} C, \ P \ sees \ \left\{P \xleftrightarrow{DHK} \overline{C}\right\}_{KCP}}{P \ believes \ C \ said \ \left\{P \xleftrightarrow{DHK} C\right\}} \tag{31}$$

Note that $\overline{C}$ has been replaced by C because they have been found equivalent in the above step since the initiator of the message is C as seen from the result of the above result. Since $P \ sees \ \left\{P \xleftrightarrow{DHK} \overline{C}\right\}_{KCP}$ (22) recognized $\overline{C}$ as the initiator, both are equivalent. Similarly, all $\overline{C}$ can be replaced with C in (22), (23) and (30).

$$\frac{P \ believes \ \ fresh\left(\left\{P \xleftrightarrow{DHK} C\right\}\right), \ P \ believes \ C \ said \ \left\{P \xleftrightarrow{DHK} C\right\}}{\boldsymbol{P \ believes \ C \ believes \ \left\{P \xleftrightarrow{DHK} C\right\}}} \tag{32}$$

Also, $P\ believes\ \overline{C}\ controls\ DHP_C,\ P\ controls\ DHP_P$ is replaced with $P\ believes\ C\ controls\ DHP_C,\ P\ controls\ DHP_P$ since $\overline{C}$ and C are the same. So we get,

$$\frac{\begin{array}{l} P\ believes\ C\ controls\ DHP_C, \\ P\ controls\ DHP_P,\ P\ believes\ C\ believes\ \left\{P \xleftrightarrow{DHK} C\right\} \end{array}}{\boldsymbol{P\ believes\ \left\{P \xleftrightarrow{DHK} C\right\}}} \qquad (33)$$

Similarly, after Msg4 on the Child side, using (11) and (24) we have,

$$\frac{C\ believes\ P \xleftrightarrow{KPC} C,\ C\ sees\ \left\{\overline{P} \xleftrightarrow{DHK} C\right\}_{KPC}}{C\ believes\ P\ said\ \left\{P \xleftrightarrow{DHK} C\right\}} \qquad (34)$$

Similar to the results for the Parent's side, it is found that $\overline{P}$ has been replaced by P in the above result. The reasoning is also same that is, the initiator of the message is P (result of the above).

Since (24) and (25) recognized $\overline{P}$ as the initiator, both are equivalent. So all occurrences of $\overline{P}$ can be replaced with P in (26) to (29).

Using (27) and (34),

$$\frac{C\ believes\ fresh\left(\left\{P \xleftrightarrow{DHK} C\right\}\right),\ C\ believes\ P\ said\ \left\{P \xleftrightarrow{DHK} C\right\}}{\boldsymbol{C\ believes\ P\ believes\ \left\{P \xleftrightarrow{DHK} C\right\}}} \qquad (35)$$

$C\ believes\ \overline{P}\ controls\ DHP_P, C\ controls\ DHP_C$ is also replaced with $C\ believes\ P\ controls\ DHP_P, C\ controls\ DHP_C$.

Using (16) and (35),

$$\frac{\begin{array}{l} C\ believes\ P\ controls\ DHP_P, \\ C\ controls\ DHP_C, C\ believes\ P\ believes\ \left\{P \xleftrightarrow{DHK} C\right\} \end{array}}{\boldsymbol{C\ believes\ \left\{P \xleftrightarrow{DHK} C\right\}}} \qquad (36)$$

Hence from (32), (33), (35) and (36) we have the following four results:

$$\begin{array}{l} P\ believes\ P \xleftrightarrow{DHK} C \\ C\ believes\ P \xleftrightarrow{DHK} C \\ P\ believes\ C\ believes\ P \xleftrightarrow{DHK} C \\ C\ believes\ P\ believes\ P \xleftrightarrow{DHK} C \end{array}$$

The four goals about the key DHK have been verified. Hence, the Diffie-Hellman key is safe. The next thing to verify is whether the session key that is exchanges at the end of the protocol is safe. Continuing with the remaining part of Msg4,

After Msg4 using (36) and (25) we have,

$$\frac{C\ believes\ P \xleftrightarrow{DHK} C, C\ sees\ \left\{P \xleftrightarrow{K} C\right\}_{DHK}}{C\ believes\ P\ said\ \left\{P \xleftrightarrow{K} C\right\}} \qquad (37)$$

Using (29) and (37),

$$\frac{C\ believes\ \ fresh\left(P \xleftrightarrow{K} C\right),\ C\ believes\ P\ said\ \left\{P \xleftrightarrow{K} C\right\}}{\boldsymbol{C\ believes\ P\ believes\ \left\{P \xleftrightarrow{K} C\right\}}} \qquad (38)$$

Using (19) and (38),

$$\frac{C\ believes\ P\ controls\ P \xleftrightarrow{K} C,\ C\ believes\ P\ believes\ \left\{P \xleftrightarrow{K} C\right\}}{\boldsymbol{C\ believes\ \left\{P \xleftrightarrow{K} C\right\}}} \qquad (39)$$

Working from Msg5 on Parent's side using (20) and (30), we get,

$$\frac{\boldsymbol{P\ believes\ P \xleftrightarrow{K} C},\ P\ sees\ \left\{P \xleftrightarrow{K} C\right\}_K}{P\ believes\ C\ said\ \ P \xleftrightarrow{K} C} \qquad (40)$$

Using (21) and (40),

$$\frac{P\ believes\ \ fresh\left(K\right),\ P\ believes\ C\ said\ \ P \xleftrightarrow{K} C}{\boldsymbol{P\ believes\ C\ believes\ P \xleftrightarrow{K} C}} \qquad (41)$$

So, we now have the following results from (20), (38), (39) and (41):

$P\ believes\ P \xleftrightarrow{K} C$

$C\ believes\ P \xleftrightarrow{K} C$

$P\ believes\ C\ believes\ P \xleftrightarrow{K} C$

$C\ beleives\ P\ believes\ P \xleftrightarrow{K} C$

The session keys exchanged at the end of the protocol have been verified to exhibit a strong final state as required by an authentication protocol in BAN logic.

5.2.2 Analysis of the Key Switch Protocol

Verifying the key switch protocol (Protocol 2) is slightly different than the authentication protocol. BAN logic is designed specifically to address an authentication protocol that ends in generating a session key for encryption and the participating entities' beliefs about the session key. But the key switch protocol does not conclude in a session key. It simply moves from one authentication key to another. The beliefs hence need to be about the new authentication key pair and not about a session key. So we will prove

$$\begin{array}{l} P \textit{ believes } P \xleftrightarrow{NPK} C \\ C \textit{ believes } P \xleftrightarrow{NPK} C \\ P \textit{ believes } C \textit{ believes } P \xleftrightarrow{NPK} C \\ C \textit{ believes } P \textit{ believes } P \xleftrightarrow{NPK} C \end{array} \quad \text{and} \quad \begin{array}{l} P \textit{ believes } P \xleftrightarrow{NCK} C \\ C \textit{ believes } P \xleftrightarrow{NCK} C \\ P \textit{ believes } C \textit{ believes } P \xleftrightarrow{NCK} C \\ C \textit{ believes } P \textit{ believes } P \xleftrightarrow{NCK} C \end{array}$$

instead of proving the goals stated in (6).

The symbols P and C denote the Parent and Child respectively. SK is the current session key used for encryption between P and C. NPK is the next pre-loaded key used for authentication from Parent to Child while NCK is the next key to be used from Child to Parent.

Following is a concrete protocol representing Protocol 2 briefly:

$$Msg1.\ P \rightarrow C : Change\ Key$$
$$Msg2.\ C \rightarrow P : \{R1\}_{SK}$$
$$Msg3.\ P \rightarrow C : \{\{R1\}_{NPK}, R2\}_{SK}$$
$$Msg4.\ C \rightarrow P : \{\{R2\}_{NCK}\}_{SK}$$
$$Msg5.\ P \rightarrow C : \{\{R2+1\}_{NPK}\}_{SK}$$

Initial assumptions:

The initial assumptions about the pre-loaded authentication key set, the session key and the challenges (R1, R2) generated by the Child and Parent are as follows.

From the result at the end of authentication protocol,

$$P\ believes\ P \xleftrightarrow{SK} C \tag{42}$$

Since NPK and NCK are pre-loaded keys,

$$\boldsymbol{P\ believes\ P \xleftrightarrow{NPK} C} \tag{43}$$

$$\boldsymbol{P\ believes\ P \xleftrightarrow{NCK} C} \tag{44}$$

Since P generated R2 in the current epoch,

$$P\ believes\ fresh(R2) \tag{45}$$

From the result at the end of the authentication protocol,

$$C\ believes\ P \xleftrightarrow{SK} C \tag{46}$$

Since NPK and NCK are pre-loaded keys,

$$\boldsymbol{C\ believes\ P \xleftrightarrow{NPK} C} \tag{47}$$

$$\boldsymbol{C\ believes\ P \xleftrightarrow{NCK} C} \tag{48}$$

Since C generated R1 in the current epoch,

$$C\ believes\ fresh(R1) \tag{49}$$

Idealized Protocol:

Msg1 and Msg2 contribute to initiating the protocol and to freshness beliefs respectively. The other messages appear in the idealized protocol below:

In Msg3, NPK was used to sign R1. So it expresses P's belief about NPK implicitly expressed. Hence after Msg3,

$$C\ sees\ \left\{P \xleftrightarrow{NPK} C\right\}_{SK} \tag{50}$$

Applying freshness rule (4) to the above using (49),

$$C\ believes\ \ fresh\left(P \xleftrightarrow{NPK} C\right) \tag{51}$$

In Msg4 since R2 was signed using NCK, it expresses C's belief implicitly expressed. So we get,

$$P\ sees\ \left\{P \xleftrightarrow{NCK} C\right\}_{SK} \tag{52}$$

Applying freshness rule (4) to the above using (45), we get,

$$P\ believes\ \ fresh\left(P \xleftrightarrow{NCK} C\right) \tag{53}$$

Since P sent a message encrypted with NPK in the Msg3 and C decrypted it and sent back Msg4 it means that C used NPK in the previous step. Hence, P also sees the following:

$$P\ sees\ \left\{P \xleftrightarrow{NPK} C\right\}_{SK} \tag{54}$$

Again, applying freshness rule (4) using (45), we get,

$$P\ believes\ \ fresh\left(P \xleftrightarrow{NPK} C\right) \tag{55}$$

From Msg5 what C sees directly in the proof is not directly used. Msg5 indicates that P was able to decrypt the message sent by C in the previous step. So we have the following,

$$C\ sees\ \left\{P \xleftrightarrow{NCK} C\right\}_{SK} \tag{56}$$

Applying freshness rule using (49), we get,

$$C\ believes\ \ fresh\left(P \xleftrightarrow{NCK} C\right) \tag{57}$$

Protocol Verification:

After Msg3, (46) and (50) we have,

$$\frac{C\ believes\ P \xleftrightarrow{SK} C,\ C\ sees\ \left\{P \xleftrightarrow{NPK} C\right\}_{SK}}{C\ believes\ P\ said\ \left\{P \xleftrightarrow{NPK} C\right\}} \tag{58}$$

Using (51) and (58),

$$\frac{C\ believes\ \ fresh\left(P \xleftrightarrow{NPK} C\right),\ C\ believes\ P\ said\ \left\{P \xleftrightarrow{NPK} C\right\}_{SK}}{\boldsymbol{C\ believes\ P\ believes\ \left\{P \xleftrightarrow{NPK} C\right\}}} \tag{59}$$

After Msg4, (42) and (52) we get,

$$\frac{P\ believes\ P \xleftrightarrow{SK} C,\ P\ sees\ \left\{P \xleftrightarrow{NCK} C\right\}_{SK}}{P\ believes\ C\ said\ \left\{P \xleftrightarrow{NCK} C\right\}} \tag{60}$$

Using (53) and (60),

$$\frac{P\ believes\ \ fresh\left(P \xleftrightarrow{NCK} C\right),\ P\ believes\ C\ said\ \left\{P \xleftrightarrow{NCK} C\right\}_{SK}}{\boldsymbol{P\ believes\ C\ believes\ \left\{P \xleftrightarrow{NCK} C\right\}}} \quad (61)$$

Using (42) and (54) we get,

$$\frac{P\ believes\ P \xleftrightarrow{SK} C,\ P\ sees\ \left\{P \xleftrightarrow{NPK} C\right\}_{SK}}{P\ believes\ C\ said\ \left\{P \xleftrightarrow{NPK} C\right\}} \quad (62)$$

Using (55) and (62),

$$\frac{P\ believes\ \ fresh\left(P \xleftrightarrow{NPK} C\right),\ P\ believes\ C\ said\ \left\{P \xleftrightarrow{NPK} C\right\}_{SK}}{\boldsymbol{P\ believes\ C\ believes\ \left\{P \xleftrightarrow{NPK} C\right\}}} \quad (63)$$

After Msg5, (46) and (56) we get,

$$\frac{C\ believes\ P \xleftrightarrow{SK} C,\ C\ sees\ \left\{P \xleftrightarrow{NCK} C\right\}_{SK}}{C\ believes\ P\ said\ \left\{P \xleftrightarrow{NCK} C\right\}} \quad (64)$$

Using (57) and (64),

$$\frac{C\ believes\ \ fresh\left(P \xleftrightarrow{NCK} C\right),\ C\ believes\ P\ said\ \left\{P \xleftrightarrow{NCK} C\right\}_{SK}}{\boldsymbol{C\ believes\ P\ believes\ \left\{P \xleftrightarrow{NCK} C\right\}}} \quad (65)$$

The results obtained are:

From (43), (47), (59) and (63),

$$P \text{ believes } P \xleftrightarrow{NPK} C$$
$$C \text{ believes } P \xleftrightarrow{NPK} C$$
$$P \text{ believes } C \text{ believes } P \xleftrightarrow{NPK} C$$
$$C \text{ believes } P \text{ believes } P \xleftrightarrow{NPK} C$$

From (44), (48), (61) and (65),

$$P \text{ believes } P \xleftrightarrow{NCK} C$$
$$C \text{ believes } P \xleftrightarrow{NCK} C$$
$$P \text{ believes } C \text{ believes } P \xleftrightarrow{NCK} C$$
$$C \text{ believes } P \text{ believes } P \xleftrightarrow{NCK} C$$

At the end of the key switch protocol it is evident that both Parent and Child maintain their beliefs about the next pre-loaded key pair. It has been verified that the key switch protocol maintains strong beliefs about the authentication keys.

As an example of a situation where all these goals cannot be attained, consider the case where Msg5 was not part of the protocol. In this case, the results (56) and (57) are not achieved. That is the results $C \text{ sees } \left\{P \xleftrightarrow{NCK} C\right\}_{SK}$ and $C \text{ believes } fresh\left(P \xleftrightarrow{NCK} C\right)$ do not exist. So, without these results, the results stated in (64) and (65) are unavailable. Consequently

C *beleives* P *believes* $P \xleftrightarrow{NCK} C$ cannot be proved. Hence, this is an example of a situation where a protocol will not be able to prove all the goals. Without the last message (Msg5), the protocol will not succeed in acknowledging to the Child about the success or failure about the completion of the key switch. Hence, Child does not get a proof of whether or not the Parent believes NCK. Since all the goals for Protocol 2 have been proved, its correctness has been verified.

5.2.3 Analysis of the Module Change Protocol

Like in the case of key switch protocol the module change protocols are also different because they do not conclude with the exchange of a session key either. Beliefs about the authentication modules are verified instead. Some concepts that are not originally in the BAN logic are introduced to facilitate expressing beliefs about modules.

Special types of statements that express the safety of a module are introduced here. The statements can be *safe(Module)* or *unsafe(Module)*. All the constructs defined in BAN logic can be applied to these statements just like applying them to an arbitrary statement. In particular, since *A sees X* is a valid construct, *A sees safe(Module)* is also a valid construct.

Though modules have been used with authentication protocols involving cryptography, beliefs about them have not been expressed explicitly. So, the message-meaning rule is re-stated to include beliefs about modules.

The original message-meaning rule as given in (1) is

$$\frac{P\ believes\ \left(P \xleftrightarrow{K} Q\right),\ P\ sees\ \{X\}_K}{P\ believes\ Q\ said\ \ X}$$

When re-stated to include beliefs about modules the message-meaning rule as follows:

$$\frac{P\ believes\ \left(P \xleftrightarrow{K} Q\right),\ P\ believes\ safe(M),\ P\ sees\ \{X\}_{<M,K>}}{P\ believes\ Q\ said\ \ X} \quad \text{(I)}$$

For P to believe that Q said X when a module M is used, P's belief about the safety of module M is explicitly stated.

The above rule however cannot be used in case X itself is a statement that expresses the safety of module M. Similarly, the above rule cannot be used in case P does not have any beliefs about M (when M is unknown to P and M was supplied to P only using the protocol). In such cases, it is postulated that the message-meaning rule involving encryption is as follows:

$$\frac{P\ believes\ \left(P \xleftrightarrow{K} Q\right),\ P\ believes\ \ fresh(K),\ P\ sees\ \{X\}_{<M,K>}}{P\ believes\ Q\ said\ \ X} \text{ (II)}$$

When any belief about M's safety cannot be expressed, the result $P\ believes\ Q\ said\ \ X$ can be obtained only if the key used is fresh. Irrespective of the safety of the module, if the key is fresh it automatically proves the initiator's identity. Hence, rule (I) can be used when the statement X is not about the safety of module M and in other cases rule (II) can be used as the message-meaning rule.

It is worth noting that keys that are used with unsafe modules may become unsafe. So, once a key is used with an unsafe module, no assumption can be made about its safety.

Now, the verification of the module change protocols is presented. The symbols P and C denote the Parent and Child respectively. SK is the current session key used for encryption between P and C. NPK is the next pre-loaded key used for messages from Parent to Child while NCK is the next key to be used from Child to Parent. OM refers to the module being currently used for authentication. NM is the new module proposed by the parent for authentication in future. Among the two module change protocols, Protocol 3 is first analyzed.

Concrete version of Protocol 3 is:

$$Msg1.\ P \rightarrow C : Change\ Module$$
$$Msg2.\ C \rightarrow P : \{R1,\ Module\ \mathrm{Re}\,quest\}_{SK}$$
$$Msg3.\ P \rightarrow C : \{NM,\ R2,\ \{R1\}_{<NM,NPK>},\ \{R1\}_{<OM,NPK>}\}_{SK}$$
$$Msg4.\ C \rightarrow P : \{\{R2\}_{<NM,NCK>}\}_{SK}$$
$$Msg5.\ P \rightarrow C : Ack$$

Initial assumptions:

Following are some initial assumptions about the pre-loaded authentication key set, the current/new authentication modules, the session key and the challenges (R1, R2) generated by the Child and Parent (The reasoning for each of these assumptions are omitted as they are similar to the ones presented for the other two protocols):

$$P\ believes\ P \xleftrightarrow{NPK} C \tag{66}$$

$$P\ believes\ P \xleftrightarrow{NCK} C \tag{67}$$

$$\boldsymbol{P\ believes\ \{unsafe(OM),\ safe(NM)\}} \tag{68}$$

$$P\ believes\ fresh(R2) \tag{69}$$

$$P\ believes\ fresh(NPK) \tag{70}$$

$$P\ believes\ fresh(NCK) \tag{71}$$

$$C\ believes\ P \xleftrightarrow{NPK} C \tag{72}$$

$$C\ believes\ P \xleftrightarrow{NCK} C \tag{73}$$

$$C\ believes\ safe(OM) \tag{74}$$

$$C\ believes\ fresh(R1) \tag{75}$$

$$C\ believes\ fresh(NPK) \tag{76}$$

$$C\ believes\ fresh(NCK) \tag{77}$$

Since P believes that R2 is fresh and C believes that R1 is fresh, the responses to those challenges are also fresh:

$$C\ believes\ fresh(unsafe(OM)),\ C\ believes\ fresh(safe(NM)) \tag{78}$$

where these beliefs are held on the receipt of Msg3 by C

$$P\ believes\ fresh(unsafe(OM)),\ P\ believes\ fresh(safe(NM)) \tag{79}$$

where these beliefs are held on the receipt of Msg4 by P

The idealized protocol for these messages further clarify the above two beliefs.

It is known that P controls the modules. So, C has the following beliefs for any module M

$$C\ believes\ P\ controls\ safe(M) \tag{80}$$

$$C\ believes\ P\ controls\ unsafe(M) \tag{81}$$

Idealized Protocol:

Msg1 and Msg2 contribute to initiating the protocol and to freshness beliefs respectively. The other messages appear in the idealized protocol below. Note that SK has been ignored altogether because it provides an extra layer of protection but it is not depended upon for authentication in this protocol. So this protocol is proved without using the session key.

After Msg3 we get,

$$C\ sees\ \{unsafe(OM)\}_{<OM,NPK>} \tag{82}$$

$$C\ sees\ \{safe(NM)\}_{<NM,NPK>} \tag{83}$$

After Msg4 we get,

$$P\ sees\ \{safe(NM)\}_{<NM,NCK>} \tag{84}$$

and since C sent a message using module NM, it implies the following

$$P \text{ sees } \{unsafe(OM)\}_{<NM,NCK>} \tag{85}$$

Notice that the above result expresses that OM is unsafe but uses NM to sign this message unlike in the case of (82). This is a result implied from the previous one (85). Since NM was used with (85), the same (NM) can only be used with (86).

Protocol Verification:

Using Msg3 (83), (72) and (74) applying the new message-meaning rule (I) we have,

$$\frac{\begin{array}{l} C \text{ believes } P \xleftrightarrow{NPK} C,\ C \text{ believes } safe(OM), \\ C \text{ sees } \{unsafe(OM)\}_{<OM,NPK>} \end{array}}{C \text{ believes } P \text{ said } unsafe(OM)} \tag{86}$$

Using (72), (76) and (84) applying the new message-meaning rule (II) we also have

$$\frac{\begin{array}{l} C \text{ believes } P \xleftrightarrow{NPK} C,\ C \text{ believes } fresh(NPK), \\ C \text{ sees } \{safe(NM)\}_{<NM,NPK>} \end{array}}{C \text{ believes } P \text{ said } safe(NM)} \tag{87}$$

Since Msg3 has R1 signed using OM and NM, and since C believes that R1 is fresh (75) it is easy to derive that (83) and (84) are also fresh (As given in (79)). On applying the nonce-verification rule using (75), (86) we get,

$$\frac{C\ believes\ \ fresh(unsafe(OM)),\ C\ believes\ P\ said\ unsafe(OM)}{C\ believes\ P\ believes\ (unsafe(OM))} \quad (88)$$

Similarly using (75) and (88) we get,

$$\frac{C\ believes\ \ fresh(safe(NM)),\ C\ believes\ P\ said\ \{safe(NM)\}}{C\ believes\ P\ believes\ (safe(NM))} \quad (89)$$

From the above two results we get

$$\boldsymbol{C\ believes\ P\ believes\ (unsafe(OM),\ safe(NM))} \quad (90)$$

Using the above with (81) and (82) we get,

$$\frac{\begin{array}{c} C\ believes\ P\ controls\ (unsafe(OM),\ safe(NM)), \\ C\ believes\ P\ believes\ \{unsafe(OM),\ safe(NM)\} \end{array}}{\boldsymbol{C\ believes\ \{unsafe(OM),\ safe(NM)\}}} \quad (91)$$

After Msg4 (85), using (68), (67) and applying message-meaning rule (I),

$$\frac{\begin{array}{c} P\ believes\ P \xleftrightarrow{NCK} C,\ P\ believes\ safe(NM), \\ P\ sees\ \{safe(NM)\}_{<NM,NCK>} \end{array}}{P\ believes\ C\ said\ \{safe(NM)\}} \quad (92)$$

Similarly after (85) using (68), (67) and applying message-meaning rule (I),

$$\frac{\begin{array}{l} P\ believes\ P \xleftrightarrow{NCK} C,\ P\ believes\ safe(NM), \\ P\ sees\ \{unsafe(OM)\}_{<NM,NCK>} \end{array}}{P\ believes\ C\ said\ \{unsafe(OM)\}} \quad (93)$$

Since Msg4 has R2 signed using OM and NM, and since P believes that R2 is fresh (69) it can be derived that (85) and (86) are also fresh as given by (79). On applying nonce-verification rule using (79), and (93) we get,

$$\frac{P\ believes\ \ fresh(\{safe(NM)\}),\ P\ believes\ C\ said\ \{safe(NM)\}}{P\ believes\ C\ believes\ (safe(NM))} \quad (94)$$

Similarly (79) and (94) we get,

$$\frac{P\ believes\ \ fresh(\{unsafe(OM)\}),\ P\ believes\ C\ said\ \{unsafe(OM)\}}{P\ believes\ C\ believes\ (unsafe(OM))} \quad (95)$$

From the above two results we have

$$\boldsymbol{P\ believes\ C\ believes\ (unsafe(OM),\ safe(NM))} \quad (96)$$

Hence now there are four beliefs about the modules from the results (70), (91), (92) and (96). They are

$$P \text{ believes } \{unsafe(OM), safe(NM)\}$$
$$C \text{ believes } \{unsafe(OM), safe(NM)\}$$
$$P \text{ believes } C \text{ believes } \{unsafe(OM), safe(NM)\}$$
$$C \text{ believes } P \text{ believes } \{unsafe(OM), safe(NM)\}$$

It has been verified that there are strong beliefs about the modules. However, since NPK and NCK were used with the unsafe module OM, no assumptions can be made about them. At the end of this protocol, both nodes change to the next keys in the list after NPK and NCK. Hence the new keys are good since beliefs about them hold. To verify Protocol 4 the same procedure as for the verification of Protocol 3 has to be followed. However, the child now possesses a new belief

$$C \text{ believes } safe(NM)$$

This is because the child already has a copy of NM with it. It is possible to obtain the results obtained for Protocol 3, with the same steps as the previous proof. Since this proof is trivial, it is not presented.

5.3 Conclusion

All three protocols have been evaluated and verified to result in the goals stated by BAN logic or in equivalent states. Beliefs about modules were

explicitly stated to facilitate analyzing the modules using BAN logic. BAN logic's logic postulates namely, message-meaning rule and nonce-verification rule were also re-stated to express beliefs about modules. Hence the correctness of the protocols has been proved using a widely used standard formal analysis method.

CHAPTER SIX
FUTURE WORK

6.1 Future Work

6.1.1 QoS-Broker Security

As proposed in this thesis, it is important to combine the QoS broker and SMS node as one unit. This will help extending the SMS's security features to the QoS brokers. The QoS broker communication needs to be secure. This will maintain the original design of having the SMS control security policies while the QoS brokers control the data dissemination policies. The number of communication links and keys are also reduced considerably.

6.1.2 Intrusion Detection

The security architecture that was presented in (Solum 2007) and the authentication framework presented in this thesis address the needs of responding to intrusions and to maintain security on the detection of a security breach. However, there is a need to have an intrusion detection system that would trigger these measures in the system.

6.1.3 End Point Security

GridStat security architecture has been designed to protect communication between GridStat nodes. But the end-points are not protected from any attacks. Securing the endpoints is essential among other needs to secure the key material used for securing the communication. The physical inaccessibility of some nodes poses additional challenges in securing them physically.

The Status Routers now merely participate in routing information and in working without interfering with security of the data. However, the level of security that needs to be associated with them have not been identified or addressed yet.

6.1.4 Access Control

Configuration of GridStat nodes' capabilities have been manually done so far. There is no structure of control followed for access control in GridStat. Work needs to be done to explore what policies need to be followed, how the policies are designed or structured, where and how the configurations need to be performed and so on.

A related issue is that of a monitoring system. Though SMS is equipped to report attempted attacks or any other security issues, there needs to be a

monitoring and control system that monitors the system state and reacts according to a defined system policy. The intrusion detection system and the SMP should report potential breaches to the monitoring system for decision making.

6.2 Conclusions

The authentication protocols presented in this thesis specifically address longevity needs in critical infrastructure process control systems. Such information systems are required to keep up to demands for timely critical information. GridStat's QoS guarantees and highly controlled management system with reliability and security features and caters to these needs. This thesis has contributed to incorporating strong authentication protocols as a foundation for GridStat's security management system. Verifying that an entity is who it claims to be is the first step towards securing a system.

The authentication protocol presented in this thesis makes use of the pre-loaded key set as key material to perform mutual authentication that concludes in strong beliefs. The key switch protocol and module change protocol are vital to maintaining the authentication capabilities of the system. The key switch protocol facilitates moving to fresher and stronger keys safely. The

module change protocol facilitates moving to more secure modules and to distribute secure modules in a secure manner. The protocols utilize the pre-loaded key set minimally and efficiently. The protocol also overcomes man-in-the-middle attacks that Diffie-Hellman suffers from by employing simple techniques from the literature.

This thesis presents an overview of security standards and other security architectures used in information systems with details on their applicability to this problem. Since these systems are not suitable to be applied as is to GridStat, this thesis provides an authentication framework that is tailored to meet requirements for GridStat.

The authentication framework has also been made available to the security management plane for the SMSs to mutually authenticate each other. Thus, the SMP is equipped to provide authentication to data plane nodes and is also self-sufficient in authenticating its own nodes.

The protocols presented in the thesis are evaluated using the BAN logic. The authentication protocol has been verified to have strong authentication final states as required for authentication protocols in conventional proofs (Michael Burrows, Martin Abadi et al. 1990). The key switch and module change protocols address different purpose than the authentication protocol. The BAN

logic does not directly provide necessary constructs to verify the beliefs of these protocols. This thesis also introduces new constructs that help verify the final states of these protocols. Though modules are used in authentication protocols, BAN logic does not explicitly express them. Modules are silent or implicit in BAN logic. However, for verifying the module change protocol it is important to be able to express beliefs about modules. This thesis has incorporated statements that express such beliefs and new rules to analyze these beliefs into BAN logic.

REFERENCES

A. Menezes, P. van Oorschot, et al. (1996). Handbook of Applied Cryptography, CRC Press.

Armin, L. (1993). "Authentication in distributed systems: a bibliography." SIGOPS Oper. Syst. Rev. 27(4): 31-41.

Bessani, A. N., P. Sousa, et al. (2008). "The Crutial Way of Critical Infrastrcuture Protection." IEEE Security & Privacy 6(6): 44-51.

Charlie Kaufman, R. P., Mike Speciner, Michael Speciner (2002). Network Security: Private Communication in a Public World, Prentice Hall PTR.

Chenxi Wang, A. Carzaniga, et al. (2002). Security issues and requirements for Internet-scale publish-subscribe systems. Proceedings of the 35th Annual Hawaii International Conference on System Sciences, 2002. HICSS. Hawaii.

Dave Bakken, Carl Hauser, et al. (2009). Periodically Updated Variables: Wide-Area Publish-Subscribe Middleware Supporting Electric Power Monitoring Control and Protection. Sumitted to: DEBS'09. URL: http://gridstat.net/publications/periodically-updated.pdf

Dave Bakken, Carl Hauser, et al. (2007). Towards More Flexible and Robust Data Delivery for Monitoring and Control of the Electric Power Grid. URL: http://gridstat.net/publications/TR-GS-009.pdf, Technical Report: TR-EECS-GS-009.

Diffie, W. and M. E. Hellman (1976). "New Directions in Cryptography." IEEE Transactions on Information Theory 22(6): 644-654.

Dionysiou, I. (2006). Dynamic and Composable Trust for Indirect Interactions. EECS. Pullman, WA, Washington State University. Ph.D Dissertation.

Haller, N. M. (1994). The S/KEY TM one-time password system. Internet Society Symposium on Network and Distributed Systems.

J. Kohl and C. Neuman. (1993). "The Kerberos Network Authentication Services (v5)." from http://www.isi.edu/in-notes/rfc1510.txt.

Jason Stamp, John Dillinger, et al. (2003). Common Vulnerabilities in Critical Infrastructure Control Systems. Alburquerque, NM, Sandia National Laboratories. URL: http://www.oe.netl.doe.gov/docs/prepare/vulnerabilities.pdf, Report Number: SAND2003-1772C

Khurana, H. (2005). Scalable security and accounting services for content-based publish/subscribe systems. Proceedings of the 2005 ACM symposium on Applied computing. Santa Fe, New Mexico, ACM.

Mao, W. and C. Boyd (1993). Towards Formal Analysis of Security Protocols. Computer Security Foundations Workshop VI, Franconia, NH, USA.

Markus, B. and N. Martin (2008). "Security for Process Control Systems: An Overview." IEEE Security and Privacy 6(6): 24-29.

Michael Burrows, Martin Abadi, et al. (1990). "A logic of authentication." ACM Transactions on Computer Systems 8(1): 18-36.

Mogollon, M. (2007). Cryptography and Security Services: Mechanisms and Applications, IGI Publishing.

Mudhakar, S. and L. Ling (2005). Securing publish-subscribe overlay services with EventGuard. Proceedings of the 12th ACM conference on Computer and communications security. Alexandria, VA, USA, ACM.

Mudhakar, S. and L. Ling (2007). Secure Event Dissemination in Publish-Subscribe Networks. International Conference on Distributed Computing Systems, 2007. ICDCS '07, Toronto, ON.

Naganand Doraswamy and D. Harkins (2003). IPSec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks, Prentice Hall PTR.

Nagendra Modadugu and E. Rescorla (2004). The Design and Implementation of Datagram TLS. Network and Distributed System Security Symposium.

Perlman, R. (1999). An Overview of PKI Trust Models. IEEE Network. 13(6): 38-43.

R. Housley, W. Ford, et al. (1999). "Internet X.509 Public Key Infrastructure Certificate and CRL Profile." from http://www.ietf.org/rfc/rfc2459.txt.

S. Kent and R. Atkinson. (1998). "Security Architecture for the Internet Protocol." from http://www.ietf.org/rfc/rfc2401.txt

Schneier, B. (1996). Applied Cryptography, Wiley.

Schneier, B. (2009). "Forging SSL Certificates." Crypto-Gram Newsletter, from http://www.schneier.com/crypto-gram-0901.html.

Seamons, K., M. Winslett, et al. (2002). Requirements for Policy Languages for Trust Negotiation. Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02), IEEE Computer Society.

Shrideep Pallickara, Marlon Pierce, et al. (2006). A Framework for Secure End-to-End Delivery of Messages in Publish/Subscribe Systems. 7th IEEE/ACM International Conference on Grid Computing (GRID 2006), Barcelona, Spain.

Simpson, W. (1996). "PPP Challenge Handshake Authentication Protocol (CHAP)." from http://www.ietf.org/rfc/rfc1994.txt.

Smith, S. and J. Marchesini (2007). The Craft Of System Security, Addison Wesley Professional.

Solum, E. (2007). Acheiving Over-the-wire Configurable Confidentiality, Integrity, Authentication and Availability in GridStat's Status Dissemination. EECS. Pullman, WA, Washington State University. M.S dissertation.

Stallings, W. (2007). "Standards for Information Security Management." The Internet Protocol Journal 10(4). URL: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_10-4/104_standards.html

Steven M. Bellovin and M. Merritt (1991). Limitations of the Kerberos Authentication System.ACM SIGCOMM Computer Communication Review 20(5): 119-132.

T. Dierks and C.Allen. (1999). "The TLS Protocol." from http://www.ietf.org/rfc/rfc2246.txt.

U.S.-Canada, P. S. O. T. F. (2004). Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations. URL: http://www.iwar.org.uk/cip/resources/blackout-03/

Wessels, J. (2001). "Applications of BAN-Logic." from http://www.win.tue.nl/ipa/archive/springdays2001/banwessels.pdf.

Yang, S. and X. Li (2006). "A Limitation of BAN Logic Analysis on a Man-in-the-middle Attack." Journal of Information and Computing Science 1(3): 131-138.

Ying Liu and B. Plale Survey of Publish Subscribe Event Systems. Bloomington, IN, Indiana University. URL: http://www.cs.indiana.edu/pub/techreports/TR574.pdf, Technical Report: TR 574

APPENDIX ONE

We present significant additions to the implementation to enable the authentication protocols below:

Parent SMS Communicator

The ParentSMSCommunicator class is meant for an SMS node to communicate with its parent. When considering communication between the data plane and the SMS, the equivalent class is the ManagementCommunicator class. To support authentication protocols, the child side of the communication has been added using methods in these classes.

The authenticateHello method is used to initiate the authentication protocol for mutual authentication of child and parent. recvAuthenticateResponse receives a challenge from the parent and returns back the response and also sends its own challenge to the parent. In recvChallengeResponse, the child gets to verify the authenticity of the parent and continues communication if the parent is authenticated.

The recvMoveToNectAuthKey method is invoked when the child receives a key switch command from the parent. recvAuthKeyChallenge and recvAuthKeyConfim perform verification steps according to the key switch protocol to conclude or discard the key switch command.

Similarly, recvChangeAuthModule, recvChangeAuthChallengeWithModule, recvAuthChallenge, recvChangeAuthModuleConfirm methods let the child participate in the module change protocol.

Child SMS Communicator

This class is used by a parent SMS node to communicate with its child SMS nodes. The leaf SMS nodes do not have this class's capabilities. The SMP to data plane equivalent to this class is the DataPlaneCommunicator class. Both classes have the same functionalities.

recvConnectHello is invoked at the parent when a child initiates the authentication protocol. recvConnectResponse is the subsequent step for the parent to verify that the child is authentic and to prove its authenticity to the child.

The moveToNextAuthenticationKey method is responsible for carrying out the parent's side of communication according to the key switch protocol. The parent initiates this protocol using this method.

The changeAuthenticationModule, changeAuthModuleResponse and changeAuthModuleResponseWithModule methods are used by the parent to carry out the module change protocol with its child node.

NodeOrganizer

An important method that has been added to the NodeOrganizer is the generateKeys method. It is used to generate encryption keys for communication between two nodes as soon as they have been successfully mutually authenticated. The NodeOrganizer acts on the parent node's side to generate keys and also to initialize the modules with these keys.

Authenticator

This class has been designed to enable the nodes to use Diffie-Hellman key exchange during the authentication protocol. It only serves to generate the

Diffie-Hellman parameters and calculate the keys each time the authentication protocol takes place. It also generates the challenges that are used to verify the authenticity of the nodes during the authentication protocol.