




A L O P A



Case Study: Deployment Considerations for a KDC in a Secure PacketCable Network

A White Paper

By Sumanth Channabasappa

Copyright Information

This material is protected by the laws of the United States and other countries. No part of this document may be reproduced, distributed, stored in a retrieval system, or altered in any form, by any means, electronic, mechanical, photocopying, or otherwise, by any entity; nor may it be used to make derivative work (such as translation, transformation, or adaptation) except in accordance with applicable agreements, contracts or licensing, without the express written permission of Alopa Networks.

Note: This document is provided for informational purposes only and may be subject to change without notice. Any mention to specifications is for reference only.

Notice

Alopa Networks has made every attempt to ensure the accuracy and completeness of information at the time of publication. As we continuously improve and add features to our products, Alopa Networks reserves the right to revise this document without prior notification of such revision or change.

Disclaimer: The contents of this case study are specific views of the author and Alopa Networks, and are not endorsed by CableLabs®.

Trademarks

"Alopa," "Alopa MetaServ™," the Alopa logo, and "Real-Time Any-Time™" are trademarks of Alopa Networks, Inc. ("Alopa"). All other trade names or trademarks are the property of their respective holders.

CableLabs® and DOCSIS(R) are registered trademarks of Cable Television Laboratories, Inc. PacketCable and CableHome are trademarks of Cable Television Laboratories, Inc.

Ordering Information

To order copies of this document, contact your Alopa Networks representative.

Technical Support

Support services can be reached via phone at 408-331-1750 ext.116, or e-mail at support@alopa.com.

Feedback

Alopa Networks appreciates your comments about this document. Please e-mail comments to documentation@alopa.com.

Version: 3/5/03 Rev. 2

© 2003 Alopa Networks, Inc. All Rights reserved.

Introduction:

Setting up a Key Distribution Center (KDC) for a large-scale PacketCable™ network deployment can present operational and security considerations that create hours of additional, manual labor. While the industry in general does not provide an automated solution to address these tasks, Alopa Networks' new Smart Server Technology™ does. The Alopa technology saves Multiple Service Operators (MSOs) time, and prevents configuration and security problems further down the line.

This paper addresses the operational overheads and additional features required for large-scale, secure PacketCable network deployment, and the thought behind our new solution. While it refers to PacketCable specifically, it also is applicable to CableHome™ and other, future technologies that may embrace similar systems.

Securing the Security Server Itself

PacketCable provides a variety of ways to protect its network with security protocols, depending on the interface. The adoption of the Kerberos Protocol [2] provides the core of the security infrastructure because Kerberos voluntarily exposes itself to suspect elements. In addition, other services rely on the authentication/rejection judgments of the Kerberos protocol and the provisioning server for normal operation.

Kerberos as a protocol assumes the KDC is a trusted server, running approved software, in a physically secure location or machine with limited access to unauthorized personnel or potentially compromised networks. The reason for this constraint is that a compromise of the KDC (or the means of authentication, which in this case are the PacketCable certificates and the service keys) leads to the compromise of the whole realm in which that KDC operates. That could impact the entire network's security, which could compromise the services and lead to theft or disruption.¹

In simple terms, this means the KDC, itself, must be installed on a different machine, independent of other application servers and disconnected from any network with which it does not directly interact.

The reasoning behind keeping the KDC isolated is that application servers open up various interfaces that can introduce security problems. For example, at a network layer, they may open up various ports such as TFTP, HTTP, which can have security holes. Or they may contain untested software such as shareware, which may give access control to users such as technicians or customer service representatives, who are not aware of the security constraints. As any network security analyst would point out, if you reduce the number of potentially un-trusted interfaces, it's easier to secure the network.

¹ Contingency measures for the same are dealt with at a later stage.

There is an argument that such a system can still be secured. But doing so increases maintenance and operational costs by requiring monitoring tools, physical access restrictions or more logging, rather than simple, physical security. In addition, the restrictions imposed can become a hindrance, themselves, such as limits on access privileges. Or, they too may become susceptible to compromise. For example: In case of emergencies, physical or network security might be eased to permit troubleshooting. In addition, for external applications, providing access to external vendors (physical or otherwise) is a potential security threat. Furthermore, relying purely on any contingency plans like Certificate Revocation Lists (CRLs)² might be disastrous because they could increase run-time costs while better, simpler mitigation plans can avoid that.

Generating the Secure Data

Assuming a KDC has been acquired and installed in a secure location, configuring the server itself and the various application servers is not a trivial task outside the lab environment.

Since PacketCable relies on both PKINIT for authentication within Kerberos, and the concept of ‘shared secret keys,’ this means:

- Generating the appropriate certificates, which involves:
 - Creating certificates with the right information (For example: REALM and FQDN)
 - Managing and distributing these certificates across multiple KDCs
 - Securing the certificates and having a process in place for distributing them
- Generating the appropriate service keys for various application servers and their distribution
- Versioning the service keys to prevent well-known hacking algorithms from breaking into the network

Let’s look at each of these, in turn.

Generating Certificates

While the generation of certificates required for a PacketCable network appears to be a non-trivial task, the maintenance and secure distribution for large-scale deployments, if not planned ahead, might prove even more daunting.

To begin with, the hierarchy looks like:

²Certificate Revocation List is described later on

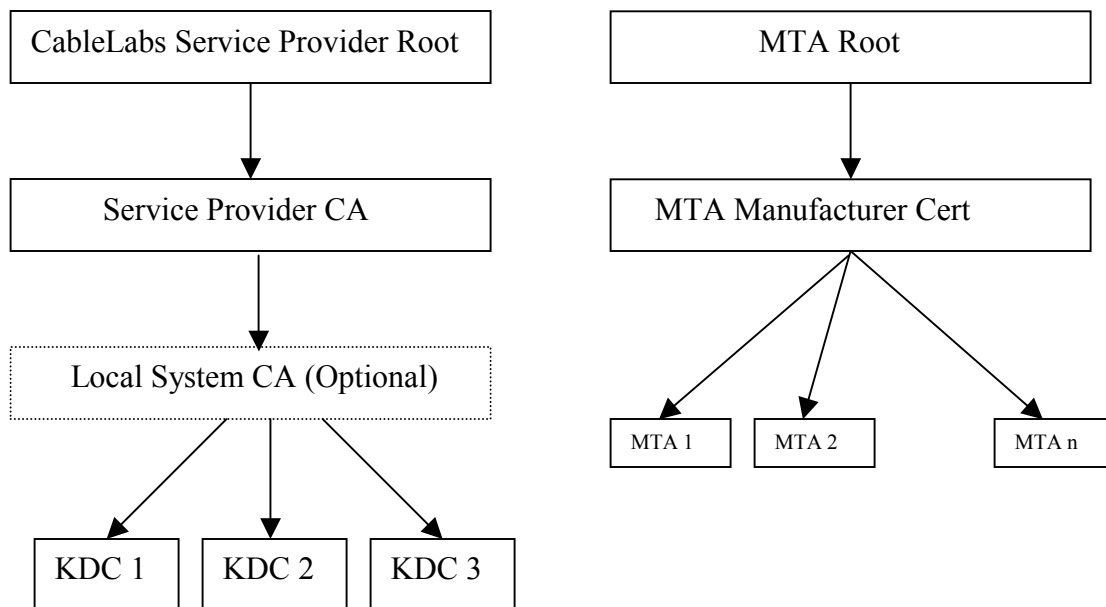


Figure 0: Relevant PacketCable Certificate Hierarchies

The hierarchy on the right is self-contained because the manufacturers embed the appropriate certificates. The hierarchy on the left, however, is under the control of the MSO (from the Service Provider CA downwards). Service Provider CA certificate maintenance is an arduous task because there are various constraints:

- The MSO has to secure the Service Provider CA certificate (specifically the private keys) to prevent compromise. If it doesn't, the entire deployment on that certificate is at risk
- While security is a major concern, the certificates cannot be made physically secure in a central location. That's because if the certificates/private keys are lost, the hierarchy becomes less useful for future expansion or regeneration of certificates. So a process must be put in place
- The certificates generated below the hierarchy have to be secured too, because compromise of any certificates can potentially lead to theft or denial of services

So the MSO needs a secure way of:

- Generating certificates
- Storing the certificates
- Distributing the certificates

The solution can be simple and standardized if the MSO has the right tools to generate the certificates and store them in standard, secure formats like PKCS#12. That format can

effectively store certificates with passwords to prevent compromise. Also, if the KDC can accept those certificates in PKCS#12 format, the distribution becomes an easier task. Current implementations mostly have proprietary formats that often expose private keys, making it easier to steal them during transit or storage.³

Maintaining the Service Keys

MSOs are not alien to the concept of sharing passwords or other secretive information. This is typically done by controlling them all at a central location and changing them on various different servers with remote access tools (SSH, Secure Telnet etc.). However, given that each KDC in a realm might be associated with a number of application servers such as provisioning servers or Call Management Systems (CMSs), a simply linear complication can grow exponentially.

As an example, Figure 1 describes a simple network involving 'n' number of servers, each having two accounts, and the password for which is changed every month.

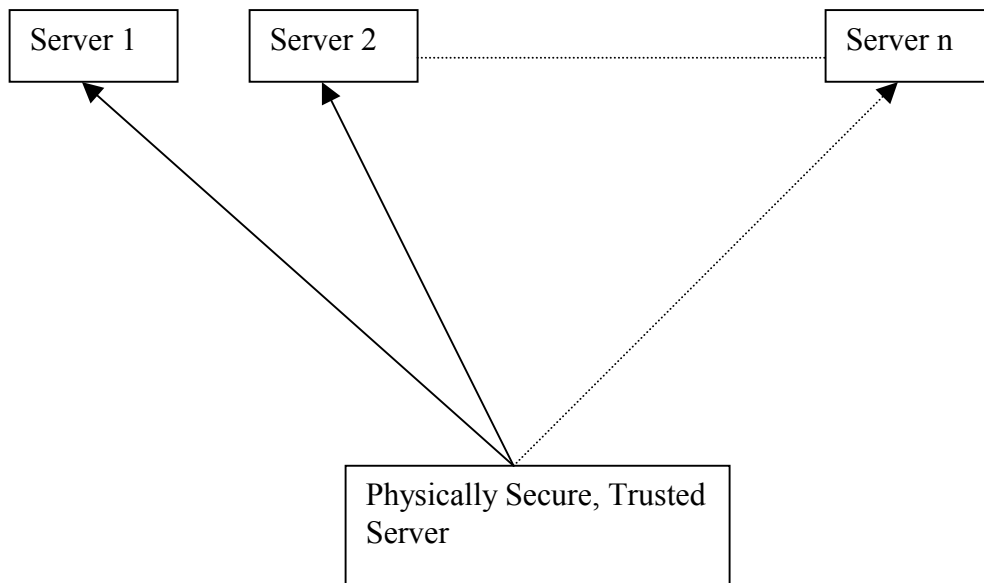


Figure 1: Updating of Secure Information

This process is easier today because the servers themselves (CMTSs or application servers) only interact via man-machine interfaces such as telnet or HTTP, or through inbuilt interfaces. They don't have to interact with each other or a centralized server. So

³ The Alopa KDC Server is among the first (and perhaps the only one currently) in the industry to incorporate pure PKCS#12 based certificate acceptance.

information update is typically passwords for various users and the kind ⁴ and is generally specific to each device.

However, this changes with a Kerberized environment. Not only do we have to update information across the network, we also have to synchronize the data between the KDC and the application servers as depicted in Figure 2.

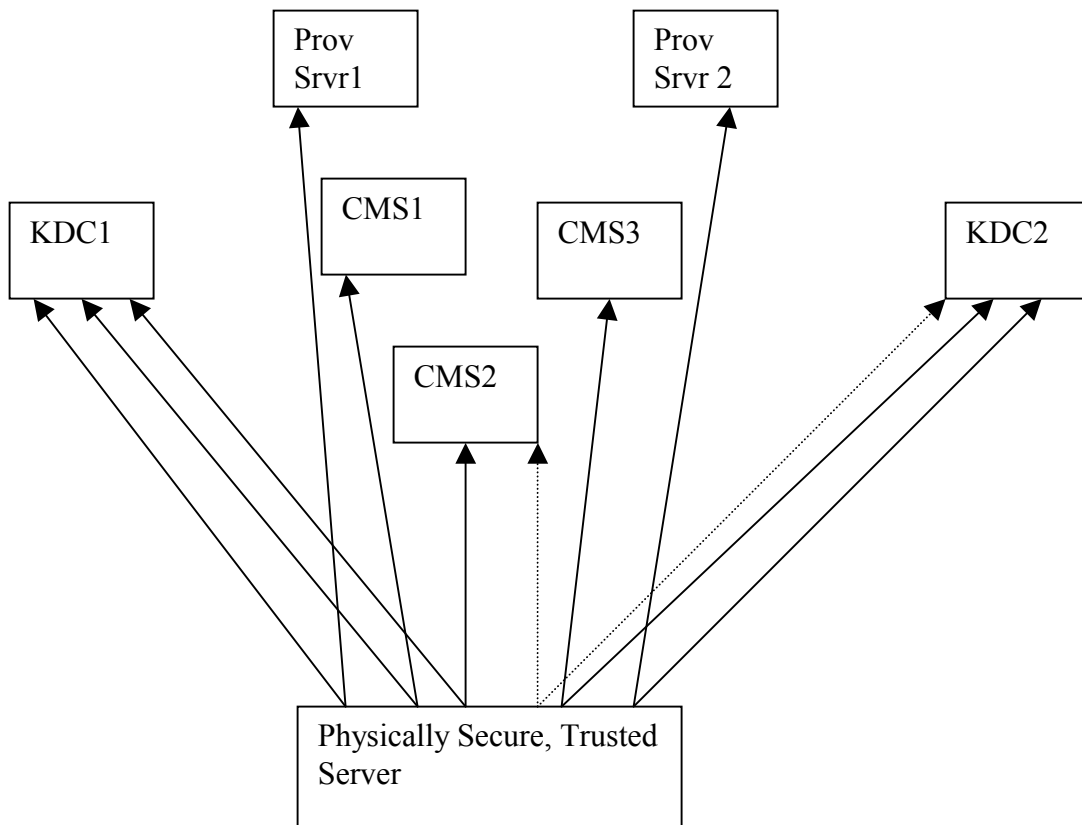


Fig 2: Update of secure information in a Kerberized network

The reason for the complexity is the need to sync all service keys. While this can be done with service key versioning, an easier way is Automated Service Key Versioning, using concepts similar to Smart Server TechnologyTM. ⁵

This would mean that:

⁴ Honorary exceptions exempted.

⁵Deal with in another paper, Smart Server TechnologyTM is a trademark of Alopa Networks, Inc.

- Servers using the Smart Server Technology automatically sync up with different, dynamically-generated service keys. The situation in Figure 2 then changes as per Figure 3.

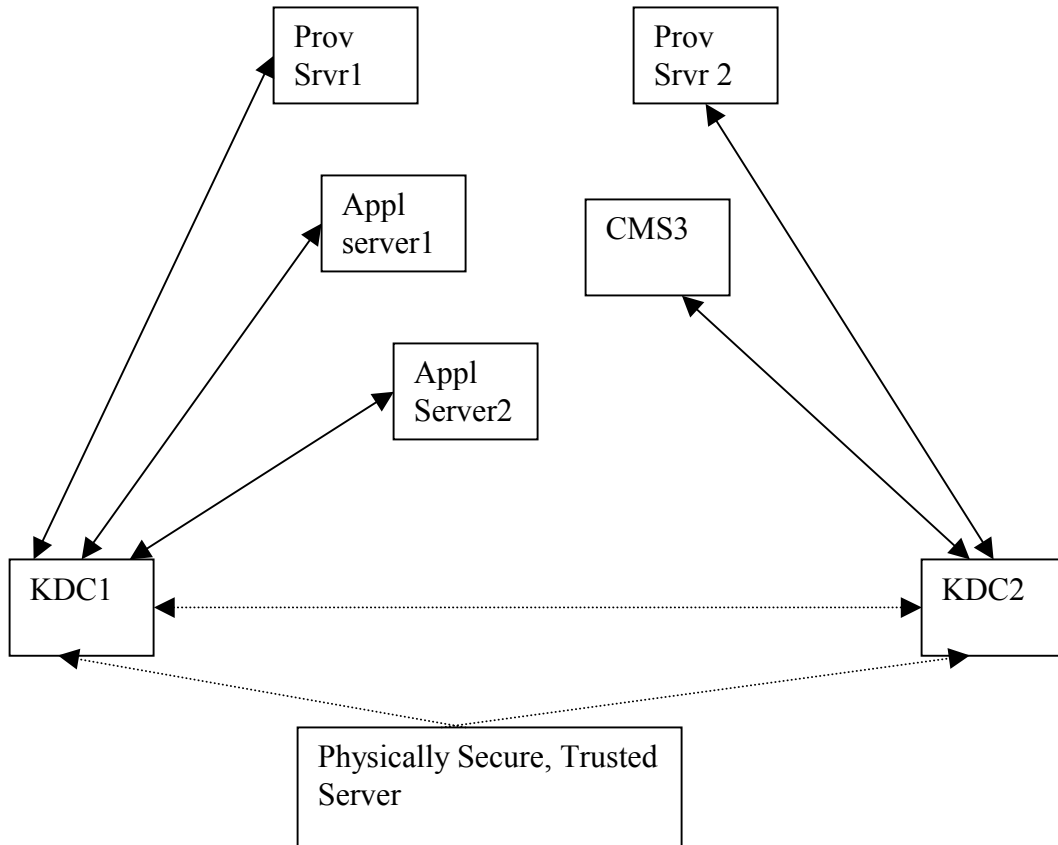


Fig 3: Update of secure information using Smart Server Technology™

As Figure 3 shows, this eliminates the need to sync all the servers from an external server. Meanwhile, the external server can still speak to the individual elements to:

- Change certificates
- Add to CRLs in case of potential compromises
- Update the periodicity of change
- Reset the seeds for service keys
- Correct conflicts
- Configure failover etc.

Proper planning in this case can reduce operational costs of:

- Improper configuration

- Compromise of the keys during configuration
- Costs of a secure interface to configure the KDCs and the various application servers to which they interact
- Non-repudiation recovery costs

It also aids in:

- Auto recovery of servers in case of software, hardware or network failure
- Increases in the security level, since automation would lead to more frequent changes to the keys involved, eliminating traditional brute-force attacks

Maintaining and Administering the Network

The planned (and unplanned) increase in services provided by the broadband industry relies heavily on the underlying security protocols and the servers themselves. This requires that all underlying servers be up 24/7 to prevent service loss.⁶ Maintaining a network of security servers with mostly encrypted messages requires savvy operators and makes it harder to troubleshoot and diagnose problems across numerous locations. Automation of recovery and fault-tolerance among the servers themselves is key for effective, transparent network operation, without compromising security requirements set by the specifications.

Thus, it is desirable that a KDC, as well as any application server incorporating Kerberized services, include as part of its architecture:

- A strong basic architecture to prevent ‘break-ins’
- Minimal intelligence⁷ to identify and isolate errors due to improper configuration⁸
- Appropriate reporting levels to indicate probable network problems⁹
- Different levels of logging capability to cater to the level of the troubleshooter, whether a technician or security expert

Protocol Overheads, Future Considerations

While Kerberos is an effective, proven technology, it has traditionally been utilized in networks that involve client-server architecture, which are mostly limited in number as well as in real-time requirements. (For example: Corporate networks.) Now, with the introduction of Kerberos into devices like Message Transfer Agents (MTAs), which are

⁶ Example: Primary Line Service.

⁷ Like Alopa Smart Server Technology™.

⁸ For Example: If there are invalid certificates configured on the server, it should flag the same.

⁹ For Example: A valid MTA, which is repeatedly sending a request for an erroneous CMS (invalid FQDN).

out in an un-trusted environment, and involving costly PKINIT operations, the processor overheads indicate that each MTA would take up a couple of seconds for each provisioning cycle. As a result, the time to bring up a whole network from scratch rises with the number of MTAs a KDC has to cater to.

In the future, it would be better to take into consideration the fact that an MTA, once authenticated, can be recognized without going through the whole PKINIT process.

Future implementations should also include ways of decreasing the KDC-Prov interactions. This would reduce interaction time for every provisioning cycle.

Alopa's Approach and Differentiator

The bottom line is: Every KDC, like any other PacketCable element that caters to these specifications, behaves the same way functionally.

However, because the cable industry is taking the leap into creating secure networks from an Access Network Standpoint,¹⁰ most of the thought development is focusing on proving and deploying the technology, rather than the operational and maintenance considerations raised here.¹¹

At Alopa, we have proven this technology in the lab and taken it from its infancy to its teenage years. Given our position in the standards arena, in general, and PacketCable in particular, we feel it is time to look forward. As a result, we are incorporating these considerations into our product, and in doing so, setting a new standard for security and KDC deployments. While these are technical requirements, they also act as differentiating factors that set Alopa's KDC offering apart from other KDCs in the market, current and the future.

Some factors worth considering are:

- The KDC was developed from scratch and is not based on the MIT KDC code¹²
- The KDC incorporates standards like PKCS#12, which aids in maintenance and distribution of the Certificates/Private Keys

¹⁰ DOCSIS1.1 is on the Rf side.

¹¹ Specifically, the KDC. Other papers will deal with other elements.

¹² An argument can be made that MIT KDC code is tested, but PacketCable makes changes within Kerberos that are easier to manage when developed independently. Besides, Alopa's solution for security is based on well-tested and reliable software libraries underneath.

Alopa KDC Software

The upcoming releases of the KDC software will include:

- PKI certificate generation tools
- PKI management tools
- Alopa Smart Server Technology™ for automatic service key versioning
- Increased efficiency through reduced time per MTA reset cycle
- An increased number of MTAs that a KDC can handle
- Failover incorporation

References

[1] PacketCable™ Security Specification, PKT-SP-SEC-I07-021127
<http://www.packetcable.com/downloads/specs/PKT-SP-SEC-I07-021127.pdf>

[2] Kerberos Network Authentication Service, draft-ietf-cat-kerberos-revisions-07.txt
Appendix B in <http://www.packetcable.com/downloads/specs/PKT-SP-SEC-I07-021127.pdf>