ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

Authenticated Key Exchange Protocols for Parallel Network File Systems

Nikhil U. Lolage[1], Nikhil P. Jadhav[1], Priyanka M. Mhase[1], Revati S. Jare[1], Prof. Sinare P. D[2]

B. E Student, Dept. of Computer Engineering SCSCOE, Rahuri Factory, Maharashtra, India[1]

HOD, Dept. of Computer Engineering SCSCOE, Rahuri Factory, Maharashtra, India[2]

ABSTRACT: We study the problem of key establishment for secure many-to-many communications. The problem is inspired by the proliferation of large-scale distributed file systems supporting *parallel access* to multiple storage devices. Our work focuses on the current Internet standard for such file systems, *i.e.*, parallel Network File System (pNFS), which makes use of Kerberos to establish parallel session keys between clients and storage devices. Our review of the existing Kerberos-based protocol shows that it has a number of limitations: (i) a metadata server facilitating key exchange between the clients and the storage devices has heavy workload that restricts the scalability of the protocol; (ii) the protocol does not provide forward secrecy; (iii) the metadata server generates itself all the session keys that are used between the clients and storage devices, and this inherently leads to key escrow. In this paper, we propose a variety of authenticated key exchange protocols that are designed to address the above issues. We show that our protocols are capable of reducing up to approximately 54% of the workload of the metadata server and concurrently supporting forward secrecy and escrow-freeness. All this requires only a small fraction of increased computation overhead at the client.

KEYWORDS*:* Parallel sessions, authenticated key exchange, networkfile systems, forward secrecy, key escrow.

I. Introduction

In a parallel file system, file data is distributed across multiple storage devices or nodes to allow concurrent access by multiple tasks of a parallel application. This is typically used in large-scale cluster computing that focuses on highperformance and reliable access to large datasets. That is, higher I/O bandwidth is achieved through concurrent access to multiple storage devices within large compute clusters; while data loss is protected through data mirroring using fault-tolerant striping algorithms. Some examples of highperformance parallel file systems that are in production use are the IBM General Parallel File System (GPFS), Google File System (GoogleFS), Lustre, Parallel Virtual File System (PVFS), and Panasas File System; while there also exist research projects on distributed object storage systems such as Usra Minor, Ceph, XtreemFS, and Gfarm. These are usually required for advanced scientific or data-intensive applications such as, seismic data processing, digital animation studios, computational fluid dynamics, and semiconductor manufacturing. In these environments, hundreds or thousands of file system clients share data and generate very high aggregate I/O load on the file system.

Independent of the development of cluster and high performance computing, the emergence of clouds, and the MapReduce programming model has resulted in file systems such as the Hadoop Distributed File System (HDFS), Amazon S3 File System , and Cloud- Store. This, in turn, has accelerated the wide-spread.
use of distributed and parallel computation on large datasets in many organizations. Some notable users of the HDFS include AOL, Apple, eBay, Facebook, Hewlett-Packard, IBM, LinkedIn, Twitter, and Yahoo!

In this work, we investigate the problem of secure manyto- many communications in large-scale network file systems that support parallel access to multiple storage devices. That is, we consider a communication model where there are a large number of clients (potentially hundreds or thousands) accessing multiple remote and distributed storage devices (which also may scale up to hundreds or thousands) in parallel. Particularly, we focus on how to exchange key materials and establish parallel secure sessions between the clients and the storage devices in the parallel Network File System (pNFS) —the current Internet standard—in an efficient and scalable manner. The development of pNFS is driven by Panasas, Netapp, Sun, EMC, IBM, and UMich/CITI, and thus it shares many common features and is compatible with many existing commercial/proprietary network file systems.

Our primary goal in this work is to design efficient and secure authenticated key exchange protocols that meet specific requirements of pNFS. Particularly, we attempt to meet the following desirable properties, which either have not been satisfactorily achieved or are not achievable by the current Kerberos-based solution (as described in Section II):

• Scalability – the metadata server facilitating access requests from a client to multiple storage devices should bear as little workload as possible such that the server will not become a performance bottleneck, but is capable of supporting a very large number of clients;

• Forward secrecy – the protocol should guarantee the security of past session keys when the long-term secret key of a client or a storage device is compromised ; and

• Escrow-free – the metadata server should not learn any information about any session key used by the client and the storage device, provided there is no collusion among them.

The main results of this paper are three new provably secure authenticated key exchange protocols. Our protocols, progressively designed to achieve each of the above properties demonstrate the trade-offs between efficiency and security. We show that our protocols can reduce the workload of the metadata server by approximately half compared to the current Kerberos-based protocol, while achieving the desired security properties and keeping the computational overhead at the clients and the storage devices at a reasonably low level. We define an appropriate security model and prove that our protocols are secure in the model. In the next section, we provide some background on pNFS and describe its existing security mechanisms associated withsecure communications between clients and distributed storage devices. Moreover, we identify the limitations of the current Kerberos-based protocol in pNFS for establishing secure channels in parallel. We describe the threat model for pNFS and the existing Kerberos-based protocol. In Section IV, we present our protocols that aim to address the current limitations. We then provide formal security analyses of our protocols under an appropriate security model, as well as performance evaluation in Sections VI and VII, respectively. In Section VIII, we describe related work, and finally inSection IX, we conclude and discuss some future work.

LITERATURE SURVEY

We address the problem with a multiple metadata server(M-MDS) design using standard parallel NFS.Password based protocols for AKE are designed to work despite the use of passwords drawn from a space small than a adversary might well enumerate offline all passwords.In a broadcast encryption scheme a broadcaster encrypts a message for some subsets S users who are listening on a broadcast channel.Describes a new version o NFS that supports access to file larger than 4GB.We present a formalism for the analysis for the key exchange protocols that combines previous definitional approaches and results in definition of security.

II. PROPOSED ALGORITHM

Fault-tolerant striping algorithms.

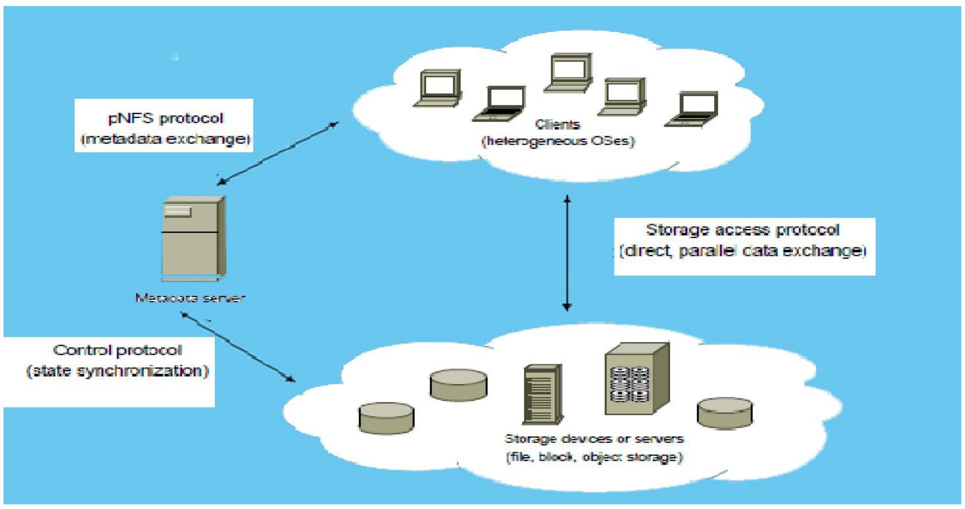

Fig. 1 System Architecture

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

A. *METHODOLOGIES*

It works in a client-server model, in which each domain (also known as realm) is governed by a Key Distribution Center (KDC), acting as a server that authenticates and provides ticket-granting services to its users (through their respective clients) within the domain. Each user shares a password with its KDC and a user is authenticated through a password-derived symmetric key known only between the user and the KDC. However, one security weakness of such an authentication method is that it may be susceptible to an off-line password guessing attack, particularly when a weak password is used to derive a key that encrypts a protocol message transmitted between the client and the KDC. Furthermore, Kerberos has strict time requirements, implying that the clocks of the involved hosts must be synchronized with that of the KDC within configuredlimits.

Hence, LIPKEY is used instead to authenticate the clientwith a password and the metadata server with a public key certificate, and to establish a secure channel between the client and the server. LIPKEY leverages the existing Simple Public- Key Mechanism (SPKM) and is specified as an GSSAPI mechanism layered above SPKM, which in turn, allows both unilateral and mutual authentication to be accomplishedwithout the use of secure time-stamps. Through LIPKEY, analogous to a typical TLS deployment scenario that consists of a client with no public key certificate accessing a server with a public key certificate, the client in NFS:

- obtains the metadata server’s certificate;
- verifies that it was signed by a trusted CertificationAuthority (CA);
- generates a random session symmetric key;
- encrypts the session key with the metadata server’s publickey; and
- sends the encrypted session key to the server.

At this point, the client and the authenticated metadata server have set up a secure channel. The client can then provide a user name and a password to the server for user authentication. Single Sign-on. In NFS/pNFS that employs Kerberos, each storage device shares a (long-term) symmetric key with the metadata server (which acts as the KDC). Kerberos then allows the client to perform single sign-on, such that the client is authenticated once to the KDC for a fixed period of time but may be allowed access to multiple storage devices governed by the KDC within that period. This can be summarized in three rounds of communication between the client, the metadata server, and the storage devices as follows:

1) the client and the metadata server perform mutual authentication through LIPKEY (as described before), and the server issues a ticket-granting ticket (TGT) to the client upon successful authentication;

2) the client forwards the TGT to a ticket-granting server (TGS), typically the same entity as the KDC, in order to obtain one or more service tickets (each containing a session key for access to a storage device), and valid layouts (each presenting valid access permissions to a storage device according to the ACLs);

3) the client finally presents the service tickets and layouts to the corresponding storage devices to get access to the stored data objects or files.

B. *ADVANTAGES:*

4) Finally, in the last augmented game, we can claim that the adversary has no advantage in winning the game since a random key is returned to the adversary.
5) Our protocols offer three appealing advantages over the existing Kerberos-based pNFS protocol.

C. *APPLICATIONS:*

- Used in Banking sector
- Used in Industrial sector
- Used in various colleges schools where sensational data is considered
- Used in telecare medical information system

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

III. Conclusion

We proposed three authenticated key exchange protocols for parallel network file system (pNFS). Our protocols offer three appealing advantages over the existing Kerberos-based pNFS protocol. First, the metadata server executing our protocols has much lower workload than that of the Kerberos-based approach. Second, two our protocols provide forward secrecy: one is partially forward secure (with respect to multiple sessions within a time period), while the other is fully forward secure (with respect to a session). Third, we have designed a protocol which not only provides forward secrecy, but is also escrow-free.

References

[1] M. Abd-El-Malek, W.V. Courtright II, C. Cranor, G.R. Ganger, J. Hendricks, A.J. Klosterman, M.P. Mesnier, M. Prasad, B. Salmon, R.R. Sambasivan, S. Sinnamohideen, J.D. Strunk, E. Thereska, M. Wachs, and J.J. Wylie. Ursa Minor: Versatile cluster-based storage. In Proceedingsof the 4th USENIX Conference on File and Storage Technologies (FAST), pages 59–72. USENIX Association, Dec 2005.
[2] C. Adams. The simple public-key GSS-API mechanism (SPKM). The Internet Engineering Task Force (IETF), RFC 2025, Oct 1996.
[3] A. Adya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J.R. Lorch, M. Theimer, and R. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In Proceedings of the 5th Symposium on OperatingSystem Design and Implementation (OSDI). USENIX Association, Dec 2002.
[4] M.K. Aguilera, M. Ji, M. Lillibridge, J. MacCormick, E. Oertli, D.G. Andersen, M. Burrows, T. Mann, and C.A. Thekkath. Blocklevel security for network-attached disks. In Proceedings of the 2ndInternational Conference on File and Storage Technologies (FAST). USENIX Association, Mar 2003.
[5] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. Communications of the ACM, 53(4):50–58. ACM Press, Apr 2010.
[6] Amazon simple storage service (Amazon S3). http://aws.amazon.com/ s3/.
[7] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In Advances in Cryptology– Proceedings of EUROCRYPT, pages 139–155. Springer LNCS 1807, May 2000.
[8] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Advances inCryptology – Proceedings of CRYPTO, pages 258–275. Springer LNCS 3621, Aug 2005.
[9] B. Callaghan, B. Pawlowski, and P. Staubach. NFS version 3 protocol specification. The Internet Engineering Task Force (IETF), RFC 1813, Jun 1995.
[10] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Advances in Cryptology – Proceedings of EUROCRYPT, pages 453–474. Springer LNCS 2045, May 2001.