**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 *high performance* 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 (Google FS), 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 animationstudios, 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 supporting petabyte- or terabyte-scale storage capacities. Independent of the development of cluster and high performance computing, the emergence of cloud, 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 with secure 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. In Section III, 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 in Section IX, we conclude and discuss some future work.