Journal of Applied and Advanced Research 2017, 2(3): 161–169
doi.: 10.21839/jaar.2017.v2i3.89
http://www.phoenixpub.org/journals/index.php/jaar
ISSN 2519-9412 / © 2017 Phoenix Research Publishers

Research Article – Computer Science Engineering

Parallel network file systems using authenticated key exchange protocols

S. Sathya*, M. Ranjith Kumar, K. Madheswaran

Department of Computer Science and Engineering, SRG Engineering College, Aniyapuram, Namakkal – 637017, Tamil Nadu, India

Abstract

The key establishment for secure many-to-many communications is very important nowadays. The problem is inspired by the proliferation of large-scale distributed file systems supporting parallel access to multiple storage devices. In this, a variety of authenticated key exchange protocols that are designed to address the issues. This shows that these protocols are capable of reducing 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. This proposed three authenticated key exchange protocols for parallel network file system (pNFS). The protocols offer three appealing advantages over the existing Kerberos-based protocol. First, the metadata server executing these protocols has much lower workload than that of the Kerberos-based approach. Second, two of these 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, designed a protocol which not only provides forward secrecy, but is also escrow-free.

Key words: Parallel sessions, Network file systems, Authenticated key exchange, Forward secrecy, Excrow-free

Introduction

This work, investigate the problem of secure many to many communications in large-scale network file systems that support parallel access to multiple storage devices. That is, 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, this 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) in 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.

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 high performance 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.

Received: 06-05-2017; Accepted 08-06-2017; Published Online 10-06-2017

*Corresponding Author

S. Sathya, *Department of Computer Science and Engineering, SRG Engineering College, Aniyapuram, Namakkal – 637017, Tamil Nadu, India*

Independent of the development of cluster and high performance computing, the emergence of clouds and the Map Reduce 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!

This paper, propose a variety of authenticated key exchange protocols that are designed to address the issues. The 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.This proposed three authenticated key exchange protocols for parallel network file system (pNFS). The protocols offer three appealing advantages over the existing Kerberos-based pNFS protocol. First, the metadata server executing these protocols has much lower workload than that of the Kerberos-based approach. Second, two of these 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, designed a protocol which not only provides forward secrecy, but is also escrow-free.

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.

Block-Level Security for Network-Attached Disks

This paperpropose a practical and efficient method for adding security to network-attached disks (NADs) [1]. In contrast to previous work, this design requires no changes to the data layout on disk, minimal changes to existing NADs, and only small changes to the standard protocol for accessing remote block-based devices. Thus, existing NAD file systems and storage-management software could incorporate in this scheme very easily. This design enforces security using the well-known idea of self-describing capabilities, with two novel features that limit the need for memory on secure NADs: a scheme to manage revocations based on *capability groups*, and a replay-detection method using Bloom filters. This have implemented a prototype NAD file system, called *Snapdragon*, that incorporates those ideas. It evaluated Snapdragon's performance and scalability. The overhead of access control is small: latency for reads and writes increases by less than 0.5 ms (5%), while bandwidth decreases by up to 16%. The aggregate throughput scales linearly with the number of NADs.

Network Storage Security

Network-storage Security is storage devices that accept block read/write requests over the network. They can be used to build file systems that provide better performance than traditional distributed file systems such as NFS. In traditional systems, disks are attached directly to a file server, which provides file access to clients across a network [9]. Because all data must pass through the server, it quickly becomes a bottleneck as the system scales, limiting the achievable bandwidth.

Authenticated Key Exchange Secure Against Dictionary Attacks

Password-based protocols for authenticated key exchange (AKE) are designed to work despite the use of passwords drawn from a space so small that an adversary might well enumerate, off line, all possible passwords. While several such protocols have been suggested, the underlying theory has been lagging. I begin by defining a model for this problem, one rich enough to deal with password guessing, forward secrecy, server compromise, and loss of session keys. The one model can be used to define various goals [2]. This take AKE (with "implicit" authentication) as the "basic" goal, and give definitions for it, and for entity-authentication goals as well. Then prove correctness for the idea at the center of the

Encrypted Key-Exchange (EKE) protocol of Bellovin and Merritt: prove security, in an ideal-cipher model, of the two-flow protocol at the core of EKE.

Password-Only Authenticated Key Exchange Using Distributed Server

Authentication using Password authenticated key exchange using distributed server is done where a cryptographic key - exchange of messages. Database of all passwords to authenticate clients are stored in a distributed server. If the server is compromised, the attacker cannot act like a client with the information from the compromised server. Solution produced for distributed-server PAKE is by having parallel two peer servers which have equal contribution to authentication or asymmetric solution for distributed-server PAKE, where the client can establish different cryptographic key with the control server [8].

Fully Collusion Secure Dynamic Broadcast Encryption with Constant- Size Cipher texts or Decryption Keys

This paper puts forward new efficient constructions for public-key broadcast encryption that simultaneously enjoy the following properties: receivers are stateless; encryption is collusion-secure for arbitrarily large collusions of users and security is tight in the standard model; new users can join dynamically i.e. without modification of user decryption keys nor cipher text size and little or no alteration of the encryption key. This also shows how to permanently revoke any subgroup of users. Most importantly, these constructions achieve the optimal bound of O(1)-size either for cipher texts or decryption keys, where the hidden constant relates to a couple of elements of a pairing-friendly group [6]. The broadcast-KEM trapdoor technique, which has3 independent interest, also provides a dynamic broadcast encryption system improving all previous efficiency measures (for both execution time and sizes) in the private-key setting.

The NFS Version 4 Protocol

The Network File System (NFS) Version 4 is a new distributed file system similar to previous versions of NFS in its straightforward design, simplified error recovery, and independence of transport protocols and operating systems for file access in a heterogeneous network [10]. Unlike earlier versions of NFS, the new protocol integrates file locking, strong security, operation coalescing, and delegation capabilities to enhance client performance for narrow data sharing applications on high-bandwidth networks. Locking and delegation make NFS stateful, but simplicity of design is retained through well-defined recovery semantics in the face of client and server failures and network partitions.

NFS Version 3 Design and Implementation

This paper describes a new version of the Network File System (NFS) that supports access to files larger than 4GB and increases sequential write throughput seven fold when compared to unaccelerated NFS Version 2 [3]. NFS Version 3 maintains the stateless server design and simple crash recovery of NFS Version 2, and the philosophy of building a distributed file service from cooperating protocols. This describes the protocol and its implementation, and provides initial performance measurements. Then describe the implementation effort. Finally, contrast this work with other distributed file systems and discuss future revisions of NFS.

Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels

The study showed the formalism for the analysis of key-exchange protocols that combines previous definitional approaches and results in a definition of security that enjoys some important analytical benefits: (i) any key-exchange protocol that satisfies the security definition can be composed with symmetric encryption and authentication functions to provide provably secure communication channels (as defined here); and (ii) the definition allows for simple modular proofs of security: one can design and prove security of key-exchange protocols in an idealized model where the communication links are perfectly authenticated, and then translate them using general tools to obtain security in the realistic setting of adversary-controlled links [4].

This exemplify the usability of the results by applying them to obtain the proof of two classes of key-exchange protocols, Diffie-Hellman and key-transport, authenticated via symmetric or asymmetric techniques.

Map Reduce: Simplified Data Processing on Large Clusters

Map Reduce is a programming model and an associated implementation for processing and generating large data sets [5]. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

TheXtreem FS architecture – a case for object-based file systems in Grids

The file abstraction is one of the success stories of system architecture, and the current computing world is unthinkable without file systems. Files are the technology of choice for any unstructured data, and provide an efficient container for abstractions with more structure. However, conventional network file systems are ill-adapted to Grid-like environments.

These file systems are usually heavily geared toward centralized installations in a single data center and lack reliable support for remote access over wide-area networks (WANs) across multiple organizations [7]. For Grid data management an approach was needed to compensate these weaknesses of installed local, network or distributed file systems. Instead of extending file system architectures with the necessary features, Grid data management systems are imposed on the existing file system.

Existing and Proposed System

Existing System

Key establishment for secure many-to-many communications. The proliferation of large-scale distributed file systems supporting parallel accessto multiple storage devices.

Parallel Network File System (pNFS) makes use of Kerberos to establish parallel session keys between clients and storage devices. File data is distributed across multiple storage devices or nodes to allow concurrent access by multiple tasks of a parallel application.

Drawbacks in Existing System

- A metadata server facilitating key exchange between the clients and the storage devices has heavy workload that restricts the scalability of the protocol.
- The protocol does not provide forward secrecy.

Proposed System

Variety of authenticated key exchange protocols that are designed to address the existing issues. Protocols are capable of reducing up to approximately 54% of the workload of the metadata server and concurrently supporting forward secrecy and escrow-freeness. Metadata server executing this protocols has much lower workload than that of the Kerberos-based approach.

Diffie-Hellman key agreement technique to both provide forward secrecy and prevent key escrow. In this protocol, each *Si* is required to pre-distribute some key material to *M* at Phase I of the protocol.

Advantages in Proposed System

- The protocol not only provides forward secrecy, but is also escrow-free.
- The metadata server executing this protocols has much lower workload than that of the Kerberos-based approach.

Diffie-Hellman key exchange, also called exponential key exchange, is a method of digital encryption that uses numbers raised to specific powers to produce decryption keys on the basis of components that are never directly transmitted, making the task of a would-be code breaker mathematically overwhelming. The Figure 1 shows the overall system architecture.

Figure 1. System Architecture

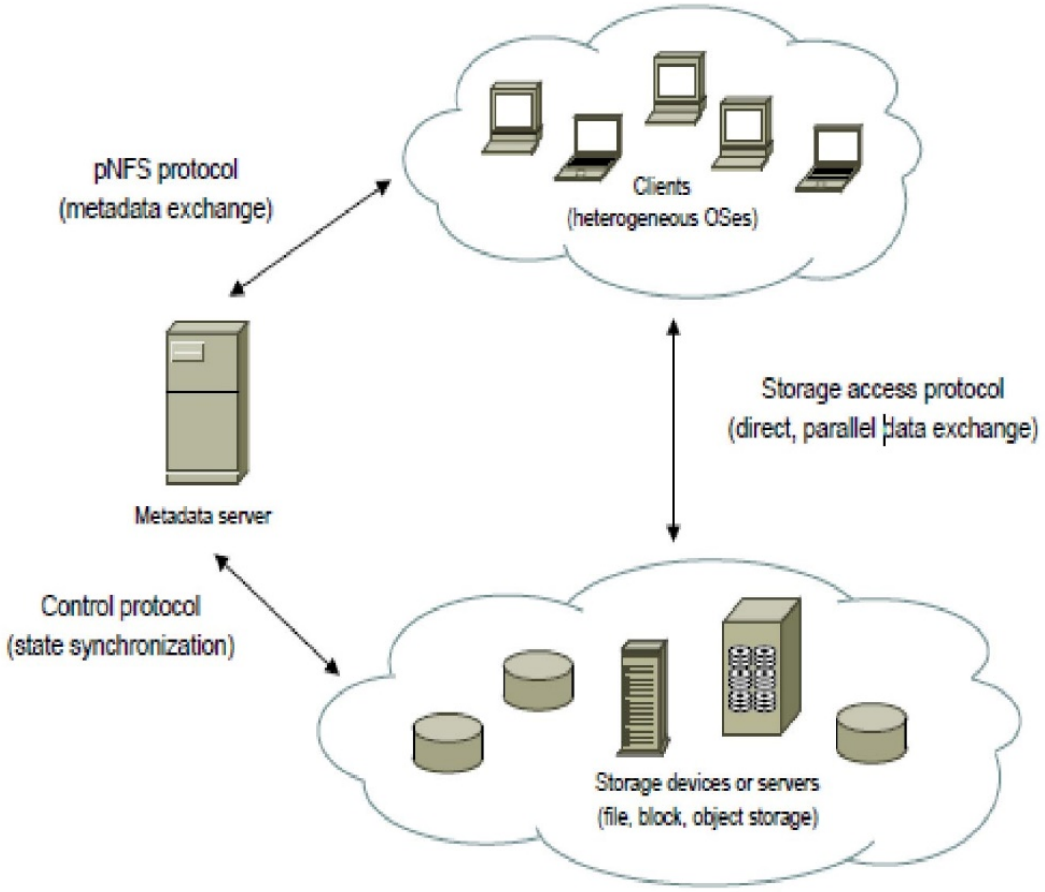


Figure 2. Specification of pNFS-AKE-I.

Phase I – For each validity period v:

(1) $C \rightarrow M: \quad ID_C, \mathcal{E}(K_{CM}; K_{CS_1}, \ldots, K_{CS_N})$

(2) $M \rightarrow C: \quad \mathcal{E}(K_{MS_1}; ID_C, ID_{S_1}, v, K_{CS_1}), \ldots, \mathcal{E}(K_{MS_N}; ID_C, ID_{S_N}, v, K_{CS_N})$

Phase II – For each access request at time t:

(1) $C \rightarrow M: \quad ID_C, ID_{S_1}, \ldots, ID_{S_n}$

(2) $M \rightarrow C: \quad \sigma_1, \ldots, \sigma_n$

(3) $C \rightarrow S_i: \quad \sigma_i, \mathcal{E}(K_{MS_i}; ID_C, ID_{S_i}, v, K_{CS_i}), \mathcal{E}(sk_i^0; ID_C, t)$

(4) $S_i \rightarrow C: \quad \mathcal{E}(sk_i^0; t+1)$

Description of Our Protocols

(i) pNFS-AKE-I: The first protocol can be regarded as a modified version of Kerberos that allows the client to generate its own session keys.

(ii). pNFS-AKE-II: To address key escrow while achieving forward secrecy simultaneously, this incorporate a Diffie-Hellman key agreement technique into Kerberos-like pNFS-AKE-I. Particularly, the client C and the storage device Si each now chooses a secret value (that is known only to itself) and pre-computes a Diffie-Hellman key component. A session key is then generated from both the Diffie-Hellman components.

(iii). pNFS-AKE-III: The third protocol aims to achieve full forward secrecy, that is, exposure of a long-term key affects only a current session key (with respect to t), but not all the other past session keys.

Figure 3. Specification of pNFS-AKE-II (with partial forward secrecy and escrow-free).

Phase I – For each validity period v:

(1) $S_i \rightarrow M: \quad ID_{S_i}, \mathcal{E}(K_{MS_i}; g^{s_i})$

(2) $C \rightarrow M: \quad ID_C, \mathcal{E}(K_{CM}; g^c)$

(3) $M \rightarrow C: \quad \mathcal{E}(K_{CM}; g^{s_1}, \ldots, g^{s_N}),$
$\tau(K_{MS_1}; ID_C, ID_{S_1}, v, g^c, g^{s_1}), \ldots, \tau(K_{MS_N}; ID_C, ID_{S_N}, v, g^c, g^{s_N})$

Phase II – For each access request at time t:

(1) $C \rightarrow M: \quad ID_C, ID_{S_1}, \ldots, ID_{S_n}$

(2) $M \rightarrow C: \quad \sigma_1, \ldots, \sigma_n$

(3) $C \rightarrow S_i: \quad \sigma_i, g^c, \tau(K_{MS_i}; ID_C, ID_{S_i}, v, g^c, g^{s_i}), \mathcal{E}(sk_i^0; ID_C, t)$

(4) $S_i \rightarrow C: \quad \mathcal{E}(sk_i^0; t+1)$

Figure 4. Specification of pNFS-AKE-III (with full forward secrecy and escrowfree).

Phase I – For each validity period v:

(1) $S_i \rightarrow M: \quad ID_{S_i}, \mathcal{E}(K_{MS_i}; g^{s_i})$

(2) $C \rightarrow M: \quad ID_C, \mathcal{E}(K_{CM}; g^c)$

(3) $M \rightarrow C: \quad \mathcal{E}(K_{CM}; g^{s_1}, \ldots, g^{s_N})$

(4) $M \rightarrow S_i: \quad \mathcal{E}(K_{MS_i}; ID_C, ID_{S_i}, v, g^c, g^{s_i})$

Phase II – For each access request at time t:

(1) $C \rightarrow M: \quad ID_C, ID_{S_1}, \ldots, ID_{S_n}$

(2) $M \rightarrow C: \quad \sigma_1, \ldots, \sigma_n$

(3) $C \rightarrow S_i: \quad \sigma_i, \mathcal{E}(sk_i^{j,0}; ID_C, t)$

(4) $S_i \rightarrow C: \quad \mathcal{E}(sk_i^{j,0}; t+1)$

Table 1: Comparison In Terms of Cryptographic Operations For W Access Requests From C To Si Via M Over Time Period V, For All $1 \leq i \leq n$ And Where $n \leq N$.

Protocol	M	C	all S_i	Total
Kerberos-pNFS				
– Symmetric key encryption / decryption	$w(n+5)$	$w(2n+3)$	$3wn$	$w(6n+8)$
– MAC generation / verification	wn	0	wn	$2wn$
pNFS-AKE-I				
– Symmetric key encryption / decryption	$N+1$	$2wn+1$	$3wn$	$5wn+N+2$
– MAC generation / verification	wn	0	wn	$2wn$
– Key derivation	0	$2wn$	$2wn$	$4wn$
pNFS-AKE-II				
– Symmetric key encryption / decryption	$N+2$	$2wn+2$	$2wn+1$	$4wn+N+5$
– MAC generation / verification	$wn+N$	0	$2wn$	$3wn+N$
– Key derivation	0	$2wn$	$2wn$	$4wn$
– Diffie-Hellman exponentiation	0	$N+1$	$N+wn$	$2N+wn+1$
pNFS-AKE-III				
– Symmetric key encryption / decryption	$2N+2$	$2wn+2$	$2wn+1$	$4wn+2N+5$
– MAC generation / verification	wn	0	wn	$2wn$
– Key derivation	0	$3wn+N$	$3wn+N$	$6wn+2N$
– Diffie-Hellman exponentiation	0	$N+1$	$2N$	$3N+1$

VII. Performance Evaluation

Computational Overhead

We consider the computational overhead for w access requests over time period v for a metadata server M, a client C, and storage devices Si for i ϵ [1, N]. We assume that a layout σ is of the form of a MAC, and the computational cost for authenticated symmetric encryption E is similar to that for the non-authenticated version E.10 Table I gives a comparison between Kerberos-based pNFS and our protocols in terms of the number of cryptographic operations required for executing the protocols over time period v. To give a more concrete view, Table II provides some estimation of the total computation times in seconds (s) for each protocol by using the Crypto++ benchmarks obtained on an Intel Core 2 1.83 GHz processor under Windows Vista in 32-bit mode. We choose AES/CBC (128-bit key) for encryption, AES/GCM (128-bit, 64K tables) for authenticated encryption, HMAC (SHA-1) for MAC, and SHA-1 for key derivation. Also, Diffie-Hellman exponentiations are based on Table 1.

- ✓ DH 1024-bit key pair generation. Our estimation is based on a fixed message size of 1024 bytes for all cryptographic operations, and we consider the following case:
- ✓ N = 2n and w = 50 (total access requests by C within v).
- ✓ C interacts with 103 storage devices concurrently for each access request, i.e. n = 103.
- ✓ M has interacted with 105 clients over time period v; and
- ✓ Each Si has interacted with 104 clients over time period v.

Table II shows that our protocols reduce the workload of M in the existing Kerberos-based protocol by up to approximately 54%. This improves the scalability of the metadata server considerably. The total estimated computational cost for M for serving 105 clients is 8.02× 104 s (≈ 22.3 hours) in Kerberos-based pNFS, compared with 3.68 × 104 s (≈ 10.2 hours) in pNFS-AKE-I and 3.86 × 104 s (≈ 10.6 hours) in pNFS-AKE-III. In general, one can see from Table I that the workload of M is always reduced by roughly half for any values of (w; n; N). The scalability of our protocols from the server's perspective in terms of supporting a large number of clients is further illustrated in the left graph of Fig.3 when we consider each client requesting access to an average of n = 103 storage devices. Moreover, the additional overhead for C (and all Si) for achieving full forward secrecy and escrow-freeness using our techniques are minimal. The right graph of Fig.3 shows that our pNFS-AKE-III protocol has roughly similar computational overhead in comparison with Kerberos-pNFS when the number of accessed storage devices is small; and the increased computational overhead for accessing 103 storage devices in parallel is only roughly 1/500 of a second compared to that of Kerberos-pNFS—a very reasonable trade-off between efficiency and security. The small increase in overhead is partly due to the fact that some of our cryptographic cost is amortized over a time period v (recall that and for each access request at time t, the client runs only Phase II of the protocol). On the other hand, we note that the significantly higher computational overhead incurred by Si in pNFS-AKE-II is largely due to the cost of Diffie- Hellman exponentiations. This is a space-computation trade-off as explained-(see Section IV-C for further discussion on key storage). Nevertheless, 256 s is an average computation time for 103 storage devices over time period v, and thus the average computation time for a storage device is still reasonably small, i.e. less than 1/3 of a second overtime period v. Moreover, we can reduce the computational cost for Si to roughly similar to that of pNFS-AKE-III if C pre-distributes its gcvalue to all relevant Si so that they can pre-compute the gcsivalue for each time period v.

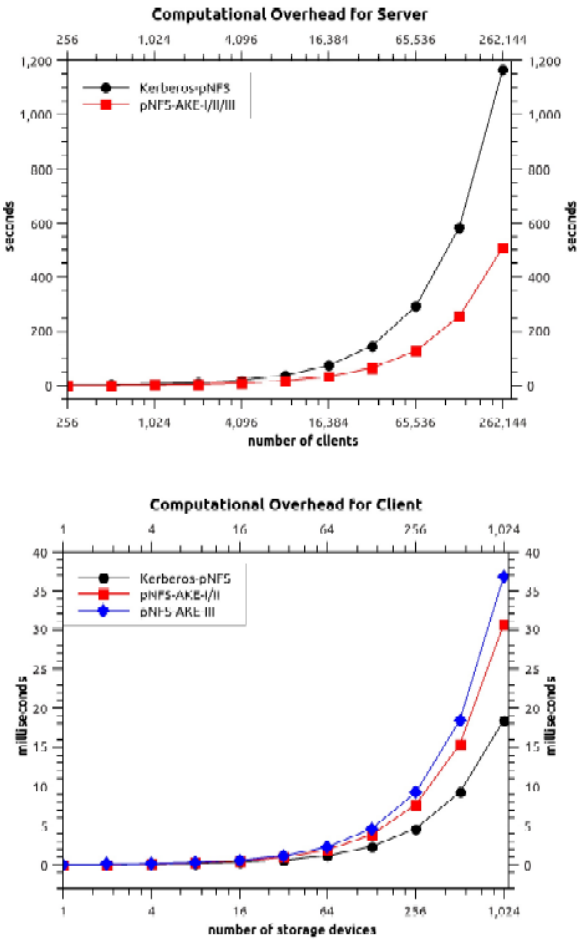
Table 2: Comparison In Terms of Computation Times In Seconds (S) Over Time Period V Between Kerberos-PNFs And Our Protocols. Here FFs Denotes Full Forward Secrecy, While EF Denotes Escrow-Freeness

Protocol	FFS	EF	M	C	S_i
Kerberos-pNFS			8.02×10^4	0.90	17.00
pNFS-AKE-I			3.68×10^4	1.50	23.00
pNFS-AKE-II		✓	3.82×10^4	2.40	256.00
pNFS-AKE-III	✓	✓	3.86×10^4	2.71	39.60

Communication Overhead: Assuming fresh session keys are used to secure communications between the client and multiple storage devices, clearly all our protocols have reduced bandwidth

requirements. This is because during each access request, the client does not need to fetch the required authentication token set from M. Hence, the reduction in bandwidth consumption is approximately the size of n authentication tokens.

Figure 5. Comparison in terms of computation times for M (on the left) and for C (on the right) at a specific time.



Key Storage: We note that the key storage requirements for Kerberosp NFS and all our described protocols are roughly similar from the client's perspective. For each access request, the client needs to store N or N + 1 key materials (either in the form of symmetric keys or Diffie-Hellman components) in their internal states. However, the key storage requirements for each storage device are higher in pNFS-AKE-III since the storage device has to store some key material for each client in their internal state. This is in contrast to Kerberos-pNFS, pNFS-AKE-I and pNFS-AKE-II that are not required to maintain any client key information.

Conclusion

In this work, proposed three authenticated key exchange protocols for parallel network file system (pNFS). The protocols offer three appealing advantages over the existing Kerberos-based pNFS protocol. First, the metadata server executing these protocols has much lower workload than that of the Kerberos-based approach. Second, two of these protocols provide for 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, designed a protocol which not only provides forward secrecy, but is also escrow-free.

References

1. Aguilera M.K., Ji M., Lillibridge M., MacCormick J., Oertli E., Andersen D.G., Burrows M., Mann T. and. Thekkath C.A. (2003). 'Block level security for network-attached disks', In Proceedings of the 2nd International Conference on File and Storage Technologies (FAST). USENIX Association, pp. 1-16.
2. Bellare M., Pointcheval D. and Rogaway P. (2000). 'Authenticated key exchange secure against dictionary attacks', In Advances in Cryptology – Proceedings of EUROCRYPT, pp. 139–155.
3. Callaghan B., Pawlowski B. and Staubach P. (1995). 'NFS version 3 protocol specification', The Internet Engineering Task Force (IETF), RFC 1813.
4. Canetti R. and Krawczyk H. (2001). 'Analysis of key-exchange protocols and their use for building secure channels', In *Advances in Cryptology – Proceedings of EUROCRYPT*, Springer. Pp. 453–474.
5. Dean J. and Ghemawat S. (2004). 'MapReduce: Simplified data processing on large clusters', In Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI), USENIX Association, pp. 137–150.

6. Delerablée C., Paillier P. and Pointcheval D. (2007). 'Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Keys',In: Pairing-Based Cryptography – Pairing 2007 (Takagi et al. Eds.). Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, pp. 39-59.
7. Hupfeld F., Cortes T., Kolbeck B., Stender J., Focht E., Hess M., Malo J., Marti J. and Cesario E. (2008). 'The Xtreem FS architecture – a case for object based file systems in grids', Concurrency and Computation: Practice and Experience (CCPE), 20(17), pp. 2049–2060.
8. Narmadha N. and Rajathi S. (2014). 'Password-only authenticated key exchange using distributed server', International Journal of Computer Trends and Technology. 9(6), pp. 305-309.
9. Riedel E., Kallhalla M., and Swaminathan R. (2002). 'A framework for evaluating storage system security', Proceedings of 1st Conference on File and Storage Technologies (FAST), pp. 15–30.
10. Shepler S., Callaghan B., Robinson D., Thurlow R., Beame C., Eisler M., and Noveck D. (2003). 'Network File System (NFS) version 4 Protocol', Network Appliance, Inc. pp. 1-257.