International Journal of Science Engineering and Advance Technology

# Advances the Forward Security in Parallel Network File System using An Efficient and Reliable Key Exchange Protocols

Kona H S Venkat Vinay[1], K Prameela [2]
#1. M.Tech (CSE) in Department of Computer Science Engineering,
#2. Assosc.Prof, Department of Computer Science and Engineering,
Gonna Institute Of Information Technology & Sciences Aganampudi,
Visakhapatnam AP, INDIA.

**Abstract:**

In parallel file system we can disperse data over numerous nodes to permit simultaneous access by various errands of a parallel application. By play out the numerous errands in a parallel system is commonly to reduce execution and solid access to substantial data sets. In this work, we research the issue of secure many to numerous correspondences in substantial scale network file systems that bolster parallel access to different storage devices. That is, we consider a correspondence model where there are countless getting to various remote and conveyed storage devices in parallel. Especially, we concentrate on the most proficient method to trade key materials and set up parallel secure sessions between the clients and the storage devices in the parallel Network File System the present Internet standard in a productive and versatile way. In this paper we are proposed convention for performing key trade and furthermore build up parallel secure session amongst clients and storage devices. In this paper we are likewise proposed another idea for encryption and decoding of put away data. By actualizing this procedure we are utilizing mixture encryption and decoding calculation. By actualizing those ideas we can enhance execution of network and furthermore give greater security of put away data in the put away devices.

**Keywords** -- Parallel sessions, Authenticated Exchange, Network file systems, forward secrecy and key escrow.

## I. Introduction

In a similar classification Network, which executed hybrid symmetric sort also for uneven key technique, enable the opportunity to traverse a few storage items, while controlling of commonsense productivity security proportion. In parallel file system, file realities are circulated all through various storage items permitting simultaneous get to acquiring two or three assignments of parallel application. This truly is every now and again introduce in vital bunch registering that focus on high complete notwithstanding dependable utilization of colossal datasets. Outside of group advancement notwithstanding high get done with processing, appearance of mists and Map Reduce programming model components to make systems. This successively has lifted broad utilization of circulated notwithstanding parallel calculation on tremendous datasets in numerous associations. Our expectation ought to be to outline proficient notwithstanding secure way to deal with confirmed key trade which will get together specific needs of parallel Network File System [1]. We endeavor to fulfill taking after satisfying attributes, which moreover were not magnificently expert or aren't achievable by current Kerberos-based arrangement. Adaptability metadata server encourages get to requests from customer to a considerable measure of storage items need to hold up under as little workload as you can to guarantee that server won't absolutely be an execution blockage, however has the ability to bolster extensive figures of clients. Forward mystery: convention must confirmation security of past session keys when augmented standing mystery key of customer generally hard drive is bargained. Sans escrow: metadata server shouldn't contemplate data concerning any session key used by customer and troublesome plate, offered there's no conspiracy together. Our objective ought to be to abatement workload of metadata server. The computational notwithstanding correspondence straightforwardness for customer notwithstanding hard drive must stay with for all intents and purposes low. Our techniques, made to accomplish every above trademark, uncover tradeoffs among proficiency notwithstanding security. Our techniques can diminish workload of metadata server by method for around 50 % when contrasted and give Kerberos-based convention, though accomplishing required security attributes notwithstanding keeping computational overhead at clients and storage items at basically low-level.

## II. Related Work

A portion of the most punctual work in securing substantial scale circulated file systems, for instance [1], [2], have officially utilized Kerberos for performing validation and implementing access control. Kerberos, being founded on for the most part symmetric key procedures in its initial arrangement, was for the most part accepted to be more appropriate for rather shut, very much associated dispersed conditions. Then again, data networks and file systems, for example, OceanStore [3], LegionFS and FARSITE [7], make utilization of open key cryptographic procedures and open key framework (PKI) to perform cross-area client confirmation. Freely, SFS, likewise in light of open key cryptographic methods, was intended to empower between operability of various key administration plans. Every client of these systems is accepted to have an ensured open/private key combine. Be that as it may, these systems were not composed particularly in view of adaptability and parallel get to. With the expanding organization of exceptionally dispersed and network-joined storage systems, consequent work, for example, [8], focussed on adaptable security. All things considered, these recommendations accepted that a metadata server shares a gathering mystery key with each conveyed storage gadget. The gathering key is utilized to create abilities as message verification codes. In any case, trade off of the metadata server or any storage gadget enables the enemy to imitate the server to whatever other substances in the file system. This issue can be reduced by requiring that every storage gadget imparts an alternate mystery key to the metadata server. By the by, such an approach confines an ability to approving I/O on just a solitary gadget, instead of bigger gatherings of pieces or protests which may live on numerous storage devices. Later recommendations, which received a half and half symmetric key and topsy-turvy key strategy, enable an ability to traverse any number of storage devices, while keeping up a sensible efficiencysecurity. For instance, Maat incorporates an arrangement of conventions that encourage (i) validated key foundation amongst clients and storage devices, (ii) ability issuance and restoration, and (iii) designation between two clients. The validated key foundation convention enables a customer to build up and re-utilize a mutual (session) key with a storage gadget. Notwithstanding, Maat and other late proposition don't accompany thorough security investigation. Likewise with NFS, confirmation in Hadoop Distributed File System (HDFS) is additionally in light of Kerberos through GSS-API. Each HDFS customer gets a TGT that goes on for 10 hours and inexhaustible for 7 days of course; and get to control depends on the Unix-style ACLs. Be that as it may, HDFS makes utilization of the Simple Authentication and Security Layer (SASL), a system for giving an organized interface between association situated conventions and replaceable instruments. Keeping in mind the end goal to enhance the execution of the KDC, the engineers of HDFS utilized various tokens for correspondence secured with a RPC process conspire. The Hadoop security configuration makes utilization of Delegation Tokens, Job Tokens, and Block Access Tokens. Each of these tokens is comparative in structure and in light of HMAC-SHA1. Designation Tokens are utilized for clients to speak with the Name Node keeping in mind the end goal to access HDFS data; while Block Access Tokens are utilized to secure correspondence between the Name Node and Data Nodes and to uphold HDFS file system authorizations. Then again, the Job Token is utilized to secure correspondence between the MapReduce motor Task Tracker and individual assignments. Take note of that the RPC process plot utilizes symmetric encryption and relying on the token sort, the common key might be dispersed to hundreds or even a huge number of hosts.

## III. Methodology

*Kerberos-Based pNFS Protocol*

The pNFS convention that exchanges document metadata, otherwise called a layout, 1 between the metadata server and a customer hub. For fulfillment, we depict the key foundation convention prescribed for pNFS in RFC 5661 between a customer C and n stockpiling gadgets Si, through a metadata server M. Since the session keys are created by M and transported to Si through C, no association is required amongst C and Si (as far as key trade) so as to concur on a session key. This keeps the correspondence overhead between the customer and every capacity gadget to a base in examination with the situation where key trade is required. In addition, the computational overhead for the customer and every capacity gadget is low since the convention is for the most part in light of symmetric key encryption. The message fills in as key affirmation that is to persuade C that Si is in control of a similar session key that C employments.

*Security Model with Forward Secrecy*

In this module, we execute security demonstrate with forward mystery. We initially present some documentation required for our conventions. Let F(k;m) mean a protected key inference work that takes as information a mystery key k and some assistant data m, and yields another key. Let sid signify a session identifier which can be utilized to

extraordinarily name the following session. Let additionally N be the aggregate number of capacity gadgets to which a customer is permitted to get to. We are presently prepared to depict the development of our conventions. We now utilize a Diffie-Hellman key assertion strategy to both give forward mystery and avoid key escrow. In this convention, every Si is required to pre-appropriate some key material to M at Phase I of the convention.

*Security Analysis*

We work in a security model that empowers us to show that an enemy attacking our traditions won't prepared to take in any information about a session key. Our model likewise proposes certain check, that is, as of late the correct convention part can learn or choose a session key The above security display for pNFS-AKE does not consider forward mystery (i.e., the debasement of a gathering won't imperil his/her past correspondence sessions). Underneath we at first portray a frail kind of forward secret we call inadequate forward puzzle (PFS).

**IV. The concept model of pNFS**

Our conventions, continuously intended to accomplish proficiency and security. Demonstrate that the conventions can diminish the workload of the metadata server by roughly half contrasted with the current Kerberos-based convention, while accomplishing the coveted security properties and keeping the computational overhead at the customers and the capacity gadgets at a sensibly low level. A proper security demonstrate and demonstrate that our conventions are secure in the model.

**Problem Description**

The attacker can decide at any point to corrupt a party, in which case the attacker learns all the internal memory of that party including long-term secrets (such as private keys or master shared keys used across different sessions) and session specific information contained in the party's memory (such as internal state of incomplete sessions and session keys corresponding to completed sessions). Since by learning its long term secrets the attacker can impersonate a party in all its actions then a party is considered completely controlled by the attacker from the time of corruption and can, in particular, depart arbitrarily from the protocol specifications. Hence key escrows occurs. Key escrow is an arrangement in which the keys needed to decrypt encrypted data are held in escrow so that, under certain circumstances, an authorized third party may gain access to those keys. The protocol does not provide forward secrecy.

**V. Proposed Methodology**

In our proposed plot, the primary point is to lessen the heap of key dissemination server and to give solid verification. Here, numerous customers' web administration can get to the application server all the while. All in all, key circulation server is utilized to make all the administration tickets and session keys between customer web administration and cloud server by putting overwhelming burden on it. In our answer utilizing .Net we will utilize web. config file for sending the key to the Key distribution server for generating the session key for file transfer. The System architecture is shown below: Fig 1: System Architecture The metadata server is trusted to work as a kind of perspective screen, issue substantial formats containing access consents, and in some cases even create session keys (for instance, on account of Kerberos-based pNFS) for secure correspondence between the customer and the capacity gadgets. The capacity gadgets are trusted to store information and just perform I/O operations upon approved solicitations. In any case, we expect that the capacity gadgets are at a significantly higher danger of being traded off contrasted with the metadata server, which is normally less demanding to screen and ensure in a unified area.

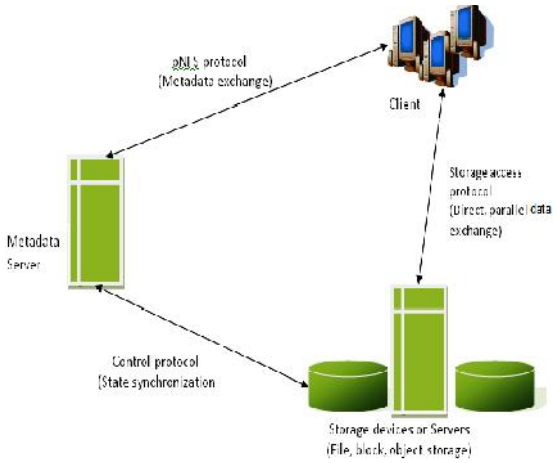


Fig.2 Proposed System Architecture

**ALGORITHM**

**Encrypted Key- Exchange (EKE) protocol**

Scrambled Key Exchange (otherwise called EKE) is a group of secret word confirmed key understanding strategies depicted by Steven M. Bellovin and Michael Merritt. Albeit a few of the types of EKE were later observed to be defective, the surviving, refined, and upgraded types of EKE adequately make this the principal strategy to enhance a common watchword into a mutual key, where the mutual key may in this manner be utilized to give a zero-learning

secret word evidence or different capacities. In the most broad type of EKE, no less than one gathering encodes a vaporous (one-time) open key utilizing a watchword, and sends it to a moment party, who decodes it and utilizations it to arrange a mutual key with the primary party.

Expanded techniques have the additional objective of guaranteeing that secret word check information stolen from a server can't be utilized by an assailant to take on the appearance of the customer, unless the aggressor initially decides the watchword.

**Executing the protocol:** Formally, a protocol is just a probabilistic algorithm taking strings to strings. This algorithm determines how instances of the principals behave in response to signals (messages) from their environment.

Our communications model places the adversary at the centre of the universe. The adversary A can make queries to any instance: she has an endless supply of $\Pi^i_U$ oracles (U ϵ ID and i ϵ N). There are all together six types of queries that A can make. The responses to these queries are specified in Figure 5.2. We now explain the capability that each kind of query captures.

**Send (U, *i*, M) |** This sends message M to oracle $\Pi^i_U$. The oracle computes what the protocol says to, and sends back the response. Should the oracle accept, this fact, as well as the SID and PID, will be made visible to the adversary. Should the oracle terminate, this too will be made visible to the adversary. To initiate the protocol with client A trying to enter into an exchange with server B the adversary should send message M = B to an unused instance of A. A Send-query models the real-world possibility of an adversary A causing an instance to come into existence, for that instance to receive communications fabricated by A, and for that instance to respond in the manner prescribed by the protocol.

**Algorithm**

Initialization()

{ h ←R Ω h pwA, pwBi A∈Client,B∈Server ←R

PWh () for i ∈ N and U ∈ ID do statei U ← ready acci U ← termi U ← usedi U ← false

sidi U ← pidi U ← ski U ← false } Send(U, i, M)

{ usedi U ← true if termi U = true then return invalid

hmsg-out, acc, termi U , sid, pid, sk, statei U i ← P h (hU, pwU , statei U , Mi) if (acc = true and ¬acci U = true) then sidi U ← sid; pidi U ← pid; ski U ← sk; acci U ← true

return hmsg-out, sid, pid, acc, termi U i }

Reveal(U, i)

{ return ski U }

Execute(A, i, B, j)

{ if A ∈/ Client or B ∈/ Server or usedi A = true or usedj B = true

then return invalid msg-in ← B for t ← 1 to ∞ do

hmsg-out, sid, pid, acc, termAi ←R Send(A, i,

msg-in) αt ← hmsg-out, sid, pid, acc, termAi if termA and termB then return hα1, β1, α2, β2, ..., αti

hmsg-out, sid, pid, acc, termBi ←R Send(B, j,

msg-in) βt ← hmsg-out, sid, pid, acc, termBi if termA and termB then return hα1, β1, α2, β2, ..., αt, βti }

Oracle(M)

{ return h(M) }

**Reveal (U, *i*) -** If oracle $\Pi^i_U$ has accepted, holding some session key *sk*, then this query returns *sk* to the adversary. This query models the idea (going back to Denning and Sacco) that loss of a session key shouldn't be damaging to other sessions. A session key might be lost for a variety of reasons, including hacking, cryptanalysis, and the prescribed-release of that session key when the session is torn down.

**Corrupt (U, *pw*) -** The adversary obtains $pw_U$ and the states of all instances of U. This query models the possibility of subverting a principal by, for example, witnessing a user type in his password, installing a "Trojan horse" on his machine, or hacking into a machine. Obviously this is a very damaging type of query. Allowing it lets us deal with forward secrecy and the extent of damage which can be done by breaking into a server. A Corrupt query directed against a client U may also be used to replace the value of $pw_B$[U] used by server B. This is the role of the second argument to Corrupt. Including this capability allows a dishonest client A to try to defeat protocol aims by installing a strange string as a server B's transformed password $pw_B$ [A].

**Execute (A, *i*, B, *j*) -** Assuming that client oracle $\Pi^i_A$ and server oracle $\Pi^j_B$ have not been used; this call carries out an honest execution of the protocol between these oracles, returning a transcript of that execution. This query may at first seem useless since, using Send queries; the adversary already has the ability to carry out an honest execution between two oracles. Yet the query is essential for properly dealing with dictionary attacks. In modeling such attacks the adversary should be granted access to

plenty of honest executions, since collecting these involves just passive eavesdropping. The adversary is comparatively constrained in its ability to actively manipulate °ows to the principals, since bogus flows can be auditied and punative measures taken should there be too many.

**Test (U, *i*) -** If $\Pi^{i}_{U}$ has accepted, holding a session key *sk*, then the following happens. A coin b is flipped. If it lands b = 0, then sk is returned to the adversary. If it lands b = 1, then a random session key, drawn from the distribution from which session keys are supposed to be drawn, is returned. This type of query is only used to measure adversarial success it does not correspond to any actual adversarial ability. You should think of the adversary asking this query just once.

**Oracle (M)** - Finally, we give the adversary oracle access to a function h, which is selected at random from some probability space Ω. As already remarked, not only the adversary, but the protocol and the LL-key generator may depend on h. The choice of - determines if we are Woking in the standard model, ideal-hash model, or ideal-cipher model.

## VI. Performance Evaluation

### *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. The total delay can be calculated as

$$(n-2)\ell+s/b$$

Where *n* is the number of messages sent; *l* is the latency in seconds; *s* is the total size of all messages; and *b* is the bandwidth in bytes per second.

### Key Storage

We note that the key storage requirements for Kerberos pNFS 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 is 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, pNFSAKE-I and pNFS-AKE-II that are not required to maintain any client key information.

### Security

First, whenever it happens we have a way to \embed" instances of the DH problem into the protocol so that adversarial success leads to our obtaining a solution to the DH problem. Second, absence of the bad event leads to an inability of the adversary to obtain information about the password at a better rate than eliminating one password per reveal or test query to a manipulated oracle. Bounding the probability of the bad event involves a \simulation" argument as we attempt to \plant" DH problem instances in the protocol. Bounding adversarial success under the assumption the bad event does not happen is an information-theoretic argument. Indeed, the difficulty of the proof is in choosing the bad event so that one can split the analysis into an information-theoretic component and a computational component in this way.

## VII. Conclusion and Future Work

We are proposed authenticated key exchange protocol for performing authentication process in parallel network file system. In this paper we are build parallel network contains storage server, metadata server, data owner and clients. By implementing this architecture we can improve efficiency in a parallel network file system. In this architecture we can perform authentication encryption and decryption of stored data. The authentication process can be done metadata server and metadata server will perform the authentication process. In the authentication process the metadata server will generate session key for individual clients for performing authentication process. After completion of authentication process the data owner will encrypt data and stored into storage server. Here the data owner will perform the encryption process for converting plain format data into cipher format. After completing encryption process the data owner will stored data into storage server. The storage server will maintain information related clients and lists of file available in the server. If any user want that files it will retrieve and perform decryption process it will get original plain text. By implementing those concepts we can improve efficient and privacy of stored data into storage server.

In the future work we can enhance the protocol to avoid the possible security attacks in the parallel network environment which will mainly focus on the key escrow. The performance of the protocol will be increased.

**References**

[1] Hoon Wei Lim and Guomin Yang, "Authenticated Key Exchange Protocols for Parallel Network File Systems" in vol. 27, no. 1, Jan. 2016

[2] M. Abd-El-Malek, W. V. Courtright II, C. Cranor, E. Thereska, M. Wachs, and J. J. Wylie, "Ursa minor: Versatile cluster-based storage," in Proc. 4th USENIX Conf. File Storage Technol., Dec. 2005,pp. 59–72.

[3] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, M. Theimer, and R. Wattenhofer, "FARSITE: Federated, available, and reliable storage for an incompletely trusted environment," in Proc. 5th Symp. Oper. Syst. Des.Implementation, Dec. 2002, pp. 1–14.

[4] M. K. Aguilera, M. Ji, M. Lillibridge, T. Mann, and C. A. Thekkath, "Block-level security for network-attached disks," in Proc. 2nd Int. Conf. File Storage Technol., Mar. 2003, pp. 159–174.

[5] A. W. Leung, E. L. Miller, and S. Jones, "Scalable security for petascale parallel file systems," in Proc. ACM/IEEE Conf. High Perform. Netw. Comput., Nov. 2007, p. 16.

[6] J. Kubiatowicz, D. Bindel, Y. Chen, and B. Y. Zhao, "OceanStore: An architecture for global scale persistent storage," in Proc. 9th Int. Conf. Arch. Support Program. Language Oper. Syst., Nov. 2000, pp. 190–201.

[7] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in Proc. 19th Int. Conf. Theory Appl. Cryptographic Techn., May 2000, pp. 139–155.

[8] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in Proc. 25th Annu. Int. Conf. Adv. Cryptol., Aug. 2005, pp. 258–275.

[9] B. Callaghan, B. Pawlowski, and P. Staubach, "NFS version 3 protocol specification," Internet Eng. Task Force (IETF), RFC 1813,Jun. 1995.

[10] F. Hupfeld, T. Cortes, B. Kolbeck, J. Stender, J.Malo, J. Marti, and E. Cesario, "The XtreemFS architecture—A case for object-based file systems in grids," Concurrency Comput.: Prac. Exp., vol. 20, no. 17, pp. 2049–2060, Dec. 2008.

**Authors**

**KONA H S VENKAT VINAY** Pursuing M.Tech (CSE) From GONNA INSTITUTE OF INFORMATION TECHNOLOGY & SCIENCES Aganampudi-530046 Visakhapatnam Dist (A.P**)**.His area of interest includes Computer Networks and Information Security.

**Ms. K.PRAMEELA**, Received M.Tech from NIT DURGAPUR, working as an Assistant Professor, Department of CSE, Gonna Institute of Information Technology And Sciences. She has 3 Years of Teaching Experience. Her Area of Interest includes COMPUTER NETWORKS.